Unimeter Serial Protocol

for

Autoplex International Pty. Ltd.

www.unimeter.com

Contents

1.0 Overview

Unilink, the proprietary serial protocol used with the Unimeter range of products has been optimised for performance and speed. It features a method of floating point compression. Furthermore, redundancy checking is performed on all communicated data. To refine operations, an extra bit, known as a wake-up bit is implemented. A high security option is also implemented, to provide a more secure means of operation.

Communications with the Unimeter range of products is purely responsive, with all devices acting as slaves in the RS485 multi-drop network. This is with exception to the IBM compatible computer that runs the Unisoft software. Also when function numbers 170 and 172 is implemented within a Unimeter or if PLC communications are utilised. This may also change with the modbus version of the unimeter, but such is irrelevant in this discussion.

2.0 the protocol

2.1 master and slave

One complete transmission consists of a number of packets that are negotiated between master and slave. All slaves on the network receive the same information, but only one slave responds at any one time. This is defined by the identification number (ID) of the particular device.

2.2 Initiating Communications

The first byte sent to a slave contains this ID. It is comprised of the following:

Bit Number	**Details**	**Value**
1	Start bit	Zero
2 to 9	ID number in binary	ID
10	Wake-up bit	One
11	Stop bit	One

The master must release the half duplex RS485 line within two bit lengths after the stop bit.

2.3 The Initial Response

Each slave device determines if it is intended to respond the transmission from the master. If the ID matches to that of its own, it sends the following acknowledgement:

Bit Number	Details	Value
1	Start bit	Zero
2 to 9	ID number in binary	Slave ID
10	Wake-up bit	Zero
11	Stop bit	One

If however there is more than one slave device with the same ID number, collisions will occur and no subsequent communications will proceed.

2.4 The Instruction Code

Once the master has verified that a valid slave exists at the designated ID number, it sends the instruction code. This packet is comprised of the following information:

Bit Number	Details	Value
1	Start bit	Zero
2 to 5	checksum nibble	*See below and 2.5
6 to 9	Function number	see 2.4.1
10	Wake-up bit	Zero
11	Stop bit	One

This byte must follow the slave's initial response within 40ms. No other packets apart from this one may be sent from the master until a response is received. Programmers may want to provide a timeout here in the order of milliseconds.

*The checksum is calculated on the initial ID number packet and extended to the second instruction code packet. Here, only the function number is used for the checksum calculations and deviates slightly to the instructions of section 2.5 in this manner.

2.4.1 Function Codes

Here is a list of function codes that can be sent to a Unimeter XQL version 6.58 and above. Please note that this san excerpt from the Unimeter program code and that 'equ' is an assembly directive meaning 'equal to'. The numbers are in decimal except where 'h' appears, indicating the value is in hexadecimal.

2.4.1.1 General Function Codes

```
send_value                      equ   1
receive_novram                  equ   2
send_novram                     equ   3
receive_value                   equ   4
send_ram                        equ   5
receive_time                    equ   6
send_logain_values              equ   7 (no longer supported)
send_medgain_values             equ   8 (no longer supported)
send_higain_values              equ   9 (no longer supported)
receive_linear_values           equ   10
receive_pcval                   equ   11
mod_setpoint_1                  equ   12
mod_setpoint_2                  equ   13
send_setup                      equ   7
send_eram                       equ   14
receive_eram                    equ   15
```

2.4.1.2 Extended Function Codes

```
send_version_number             equ   1
receive_profile_values          equ   2
backup_cal_values               equ   3 (no longer supported)
send_new_ram_values             equ   4
receive_new_ram_values          equ   5
send_sp1_value                  equ   6
send_sp2_value                  equ   7

receive_proportional_band_value equ   8
receive_pid_terms               equ   9
send_pid_terms                  equ   10
```

To submit an extended code, a general function number of zero needs to be sent. This is to be then followed by three packets of data. The first of these three bytes is the extended function code as listed in this section. The second is an exclusive-OR of the first byte with itself. The third byte is spare. Here, each of the start and wake-up bits are zero, with the stop bit being one.

2.5 The CRC

The CRC is only a nibble, calculated on an entire message. This includes all packets sent by the master to the slave. This does not include the slave's response (if any) as a separate CRC is calculated for that. The method of calculation is the same for both.

2.5.1 The Checksum Process

The checksum is calculated by the following process:

1. Load a temporary byte variable (temp) with zero
2. Exclusive-OR the low nibble of a packet with temp
3. Store the result in temp
4. Move the high nibble of a packet to a low nibble position
5. Exclusive-OR this low nibble with temp
6. Store the result in temp
7. Steps 2 to 6 are repeated for all subsequent packets originating from a device

please note that all checksum calculations do not include the start, stop or wake-up bits.

2.5.2 Checksum Code Fragment

Here is an example of how to generate the checksum. This is written for the Intel 8051 family.

```
      mov   a,sbuf_2          ;load packed data into ACC
      anl   a,#0fh            ;extract low nibble
      xrl   temp,a            ;XOR with current checksum
      mov   a,sbuf_2          ;load packed data into ACC
      swap  a                 ;extract low nibble by reversing nibbles
      anl   a,#0fh
      xrl   temp,a            ;XOR with current checksum
```

Legend:

```
      mov   - move byte variable
      anl   - logical AND for bit variables
      xrl   - logical XOR for byte variables
      swap  - swap nibbles within the accumulator (ACC)
```

2.6 The Slave's Final Response

There are various responses that various slaves will provide, depending upon the requested function number.

This information is only guaranteed to be correct for version 6.58 and above. If in doubt, ask for your Unimeter XQL firmware to be updated.

2.6.1 Send Value

For the most part, it is desirable to know the response of a Unimeter XQL. The final response from such a Unimeter is at the end of a message from the master device. Each byte contains start, stop and wakeup bits. It takes the following form:

Byte Number	Details	Value
1	Two BCD nibbles	High order
2	Two BCD nibbles	Middle order
3	Two BCD nibbles	Low order
4	Checksum and scaling	*see below

*The fourth bit is of the following format:

Bit Number	Details	Value
1	Start bit	Zero
2 to 5	checksum	**see below
6	spare	Ambiguous
7	Value is negative	Zero or one
8	Divide final value by 10	Zero or one
9	Divide final value by 100	Zero or one
10	Wake-up bit	Zero
11	Stop bit	One

**The checksum located in the fourth bit is calculated on all other 7 nibbles of these four packets. This includes bytes 1 to 3 and the high order nibble of the fourth packet.