A1700 COMMUNICATIONS – DATA STREAM MODE (DSM) SPECIFICATION

P Smith
SENIOR ENGINEER

P. Smith 14/6/07

SUMMARY

This document describes the Data Stream Mode (DSM) used to permit large data blocks to be retrieved from the A1700.

PRODUCT

A1700

CONTENTS

1.0 Overview 2
2.0 Sign on behaviour change: 3
3.0 Data identity support. 6
4.0 Modifications and additions to data identities 6
  4.1.0 Data identity 500 6
  4.2.0 Data identity 543 7
  4.3.0 Data identity 544 7
  4.4.0 Data identity 550 8
  4.5.0 Data identity 552 8
  4.6.0 Data identity 551 8
  4.7.0 Data identity 687 9
5.0 New read command. 9
  5.1.0 Notation 9
  5.2.0 Command structure 9
  5.3.0 Index & packet numbers 10
  5.4.0 Data packet structure. 10
  5.5.0 Data stream mode behaviour 13
  5.6.0 Data stream termination 13
6.0 Example communication sessions 14
7.0 CRC algorithm 17
8.0 Protocol timings 18
9.0 Related or generated documents 19
10.0 Appendices 20
  10.1.0 Appendix 1 – CRC method 20
  10.2.0 Appendix 2 - *Theoretical* communication timing 21
  10.3.0 Appendix 3 - Design decisions 22

Firmware Design Specification (Data Stream Mode)

1.0 Overview

This document defines the protocol changes that are to be made to the Elster A1700 meter to support “Data Stream Mode” (DSM). This mode permits large data blocks to be streamed from the A1700 without the requirement for the host to progress the data retrieval.

2.0 Sign on behaviour change:

Since it is necessary for the host software to determine whether the A1700 can support data stream mode, the following mechanism is proposed.

Sign on

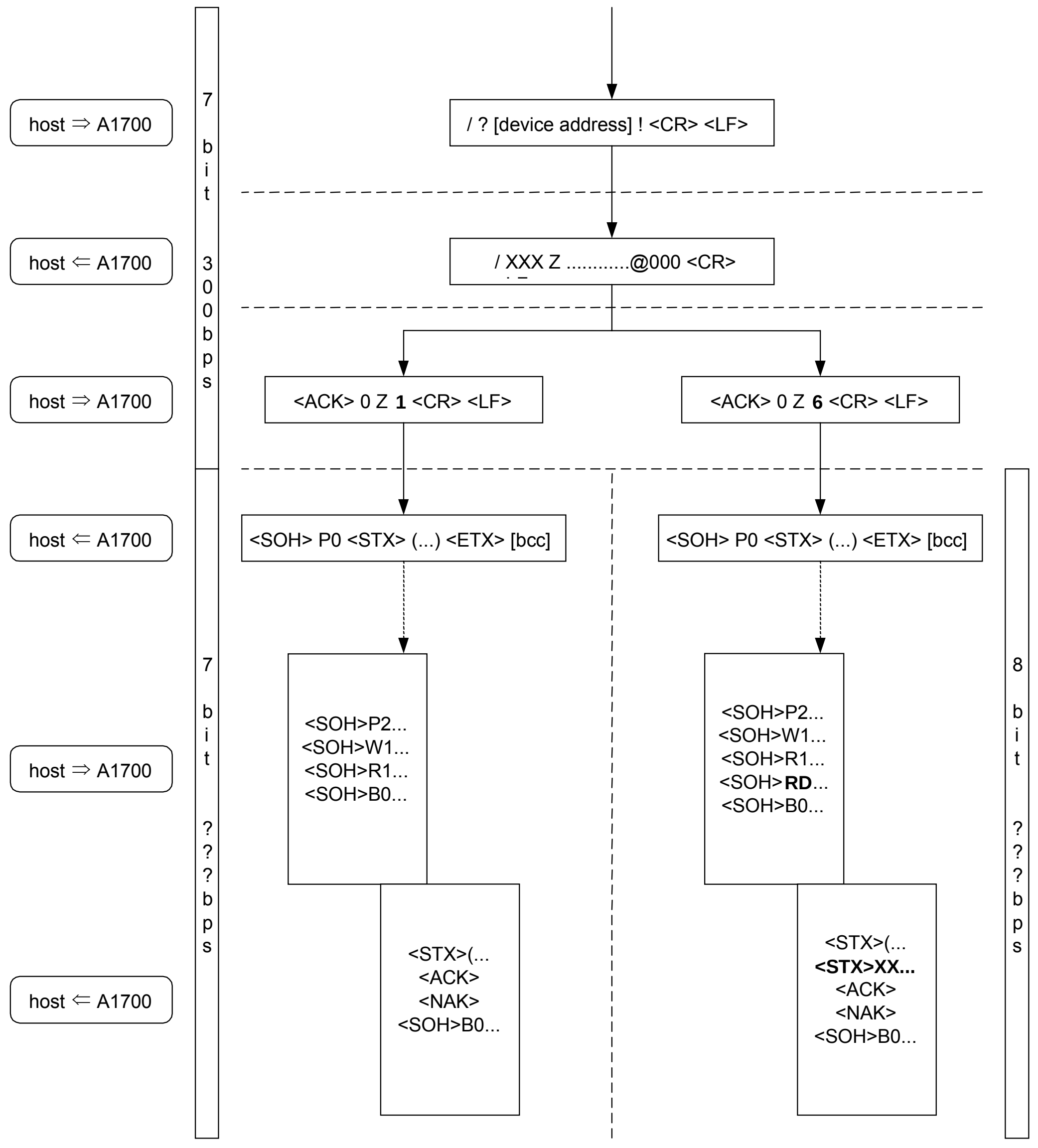

host ⇒ A1700
host ⇐ A1700
host ⇒ A1700
host ⇐ A1700
host ⇒ A1700
host ⇐ A1700
7 bit 300 bps
7 bit ??? bps
8 bit ??? bps
/ ? [device address] ! <CR> <LF>
/ XXX Z ....@000 <CR>
<ACK> 0 Z 1 <CR> <LF>
<ACK> 0 Z 6 <CR> <LF>
<SOH> P0 <STX> (...) <ETX> [bcc]
<SOH> P0 <STX> (...) <ETX> [bcc]
<SOH>P2... <SOH>W1... <SOH>R1... <SOH>B0...
<SOH>P2... <SOH>W1... <SOH>R1... <SOH>RD... <SOH>B0...
<STX>(... <ACK> <NAK> <SOH>B0...
<STX>(... <STX>XX... <ACK> <NAK> <SOH>B0...

Figure 1

Normal sign-on sequence:

HOST ⇒ A1700	/?!<CR><LF>	
HOST ⇐ A1700	/GEC5090100120400@000<CR><LF>	
HOST ⇒ A1700	<ACK>05**1**<CR><LF>	…[31]…
: : : :	(Mode change delay: 200 to 1500ms) Baud rate change from 300 to 9600bps at 7-bit, 1-stop, even parity.	
HOST ⇐ A1700	*<SOH>P0<STX>(974D640ADDF1A806)<ETX>e*	

Following a standard host sign-on message the A1700 will respond with its identification message.

The host software will then respond with the "Acknowledgement/option select message". For a normal IEC62056-21 session, the message would be "<ACK>051<CR><LF>", however to enter DSM, the host must reply with "<ACK>05**6**<CR><LF>", the 'Mode control character' being the digit '6' (36h). Digit '6' being the first available code representing 'Manufacturer-specific use'. ([1]6.4.5.3).

Sign on sequence to enter 'Manufacturer specific' mode.

HOST ⇒ A1700	/?!<CR><LF>	
HOST ⇐ A1700	/GEC5090100120400@000<CR><LF>	
HOST ⇒ A1700	<ACK>05**6**<CR><LF>	…[36]…
: : : : :	(Mode change delay: 200 to 1500ms) Baud rate change from 300bps at 7-bit, 1-stop, even parity, to 9600bps at 8-bit, 1-stop, no parity.	
HOST ⇐ A1700	*<SOH>P0<STX>(974D640ADDF1A806)<ETX>e*	

Then follows the 'baud rate change' delay. This is the standard protocol time of (Tr).

$$200\text{ms} \le Tr \le 1500\text{ms} \quad ([1]\ 6.4.2.5)$$

In this new mode, both host and A1700 will apply the following communications parameters:

- 9600bps (or as determined by 'Baud rate identification' code in the "Acknowledgement/option select message")
- 8-bit data.
- 1-stop bit.
- No parity.

Note that when communicating with the A1700 via its RS232 port no baud rate and data bit length changes are possible during signon. Therefore the signon must be performed at the configured baud rate and data bit length.

From this point onwards, all of the protocol messages will have the identical structure to those defined in [1], with the exception of the new data stream read command (see 5.0).

3.0 Data identity support.

The data stream read command will only be supported by the following data identities:

- 500 LIVE REGISTER DATA BLOCK
- 543 HISTORICAL VALUES
- 544 HISTORICAL EVENTS
- 550 LOAD PROFILE
- 552 LOAD PROFILE (RAW)

A data stream read command to any other data identity will result in the A1700 returning an “ERR2” error message.

4.0 Modifications and additions to data identities

In data stream mode data formats for existing data identities remain as specified in [3] and [4].

4.1.0 Data identity 500

This data identity is **new**. It must provide read access to the following data elements:

- o Cumulative total registers
    - Import and Export Wh
    - Q1 to Q4 vars
    - VA
    - Customer defined 1 to 3
    - Multi-utility registers 1 to 4
    - Time of Use registers 1 to 32
- o Maximum demand registers
- o Cumulative maximum demand registers

Data identity size:	728-bytes 6 packets.	
Packets:	1 of 80-bytes	(as data identity 507)
	1 of 32-bytes	(as data identity 516)
	1 of 256-bytes	(as data identity 508)
	1 of 256-bytes	(as data identity 510)
	1 of 32-bytes	(as data identity 510 cont.)
	1 of 72-bytes	(as data identity 509)

This data identity access must provide an access method that supports the data stream read command data packet size of 256-bytes.

4.2.0 Data identity 543

This is a current data identity and provides read access to the “HISTORICAL VALUES” data set.

This data identity access must provide an access method that supports the data stream read command data packet size of 256-bytes.

Access to each individual historical data set will be maintained.

Three packets per historical set: 2 of 256-bytes plus 1 of 231-bytes, 743-bytes per set, 12 sets.

Data identity size: 8916-bytes in total, 36 packets in total.

4.3.0 Data identity 544

This is a current data identity and provides read access to the “HISTORICAL EVENTS” data set.

This data identity access must provide an access method that supports the data stream read command data packet size of 256-bytes.

Access to each individual historical data set will be maintained.

One packet per historical set: 1 of 66-bytes, 12 sets.

Data identity size: 792-bytes in total, 12 packets in total.

4.4.0 Data identity 550

This is a current data identity and provides read access to the “LOAD PROFILE” data set.

This data identity access must provide an access method that supports the data stream read command data packet size of 256-bytes.

Data identity size: 90112-bytes in total (at max. capacity), 352 packets in total.

4.5.0 Data identity 552

This is a current data identity and provides read access to the “RAW LOAD PROFILE” data set.

This data identity access must provide an access method that supports the data stream read command data packet size of 256-bytes.

Data identity size: 90112-bytes in total (at max. capacity), 352 packets in total.

4.6.0 Data identity 551

Data identity 551, when written by the host software provides a mechanism for the number of days of load profile data to be configured. Data identity 551 is subsequently read by the host software in order to obtain the number of data packets storing the data. Currently the number of data packets is returned in 2 bytes in terms of the number of 64-byte packets to read. Data identity 551 will be extended to 4 bytes. Bytes 1 and 2 will contain the number of 64-byte packets, bytes 3 and 4 will contain the number of 256-byte packets.

Note that data identity 551 itself does not support data stream mode and as such is accessed using the normal command (W1/R1).

4.7.0 Data identity 687

Data identity 687 currently configures the fixed baud rate for the A1700 RS232 port (in its second byte). The identity will be extended by 1 further byte that will be used to select the bit rate of the RS232 port. Two settings will be possible:

- 7 bits, 1 stop bit, even parity.
- 8 bits, 1 stop bit, no parity.

Note that data identity 687 itself does not support data stream mode and as such is accessed using the normal command (W1/R1)

5.0 New read command.

5.1.0 Notation

<???>	Represents a standard ASCII control code in the range 00h to 1Fh. e.g. <NAK> is ASCII code 15h
[??]	Represents an application specific byte code mnemonic. e.g. [lsPP] is the least significant byte of the packet number in binary. [msPP] is the least significant byte of the packet number in binary.
?	Represents a 'literal' character. e.g. "(" is character code 8h

5.2.0 Command structure

The new data stream read command is 'RD'. [1]6.3.14-19 defines the 'R' for 'read command' and the codes that follow it. 0 to 4 are standard. 5 and 6 are reserved for 'national use' and 7 to 9 are reserved for 'future use. So, since our implementation permits 0-9 and A-F, 'D' has been chosen for 'data stream' mode.

The command will have the standard structure:

<SOH>R**D**<STX>550**xxx**(**yy**)<ETX>[bcc]

The data identity code will operate as normal.

5.3.0 Index & packet numbers

The 256-byte packets are accessed using an index number together with the number of packets required.

The index number (**xxx**) is hexadecimal and is valid in the range of 001h to FFFh.

The index number 000h has special meaning i.e. return ALL available data from the data identity. In this case the number of packets (see below) will be ignored.

The number of packets (**yy**) is hexadecimal and is valid in the range of 01h to FFh.

If an attempt is made to read more data than is available, all data will be read and the last packet will contain the 'end-of-data' packet marker. (5.4.0)

In the following cases, the A1700 will return "ERR2".

- Should there be no 'valid' data available or
- The data identity to be read is not accessible using DSM.

5.4.0 Data packet structure.

Following a data stream mode read request, the A1700 will commence transmitting data.

The new data packet will have the following structure:

Token	Description
<STX>	Start of packet marker (conforms to current implementation)
[lsPP]	Index number in binary (least significant byte)
[msPP]	Index number in binary (most significant byte)
[LL]	Data (Length – 1) in binary (single byte)
←1 to 256-bytes of data →	Between 1 to 256-bytes of data
<ETX> | <EOT>	End of data packet marker
[lsCRC]	Binary CRC of packet from <STX> to <E**> (least significant byte)
[msCRC]	Binary CRC of packet from <STX> to <E**> (most significant byte)

- Start of packet marker.

<STX>	Start of packet marker (conforms to current implementation)

The start of every data packet from the A1700, either in data stream or standard mode will start with the "<STX>" (02h) byte code.

(This byte is included in the CRC data range.)

- Index number.

[lsPP]	Index number in binary (least significant byte)
[msPP]	Index number in binary (most significant byte)

The index number. Its valid range is 1d to 65,535d. The data will be encoded in binary format. The 16-bit index is stored in the packet in little endian format i.e. least significant byte in the lowest memory location.

(These bytes are included in the CRC data range.)

- Data length minus one

[LL]	Data length minus one in binary (single byte)

This byte represents the number of data bytes minus one contained in the packet. Its valid range is 0d to 255d i.e. 1 to 256 bytes of data. The data will be encoded in binary format.

(This byte is included in the CRC data range.)

- Packet data

←1 to 256-bytes of data →	Between 1 to 256-bytes of data

The data will be encoded in binary format.

(These bytes are included in the CRC data range.)

- End of data packet marker.

<ETX> | <EOT>	End of data packet marker

The end of every data block within a packet from the A1700 will end with an "<ETX>" (02h) or "<EOT>" (04h) byte code.

Only the last packet of a data stream session will contain an "<EOT>" byte code.

(This byte is included in the CRC data range.)

- Packet validation CRC

[lsCRC]	CRC of packet from <STX> to <E**> (least significant byte)
[msCRC]	CRC of packet from <STX> to <E**> (most significant byte)

This CRC is generated from every byte in the packet from (and including) the "<STX>" byte code to the (and including) the "<ETX> or <EOT>" byte code. Refer to section 7.0 for details. The 16-bit CRC is stored in the packet in little endian format, least significant byte in the lowest memory location.

5.5.0 Data stream mode behaviour

Following the data stream read request the A1700 will transmit the requested data in a number of 256-byte packets.

The host software is not required to progress this transfer.

Any re-transmission of a packet must be requested upon the completion of the operation.

First and subsequent packet structure if request requires more that one packet.

<STX>[lsPP][msPP][LL]←1 to 256-bytes of data →**<ETX>**[lsCRC][msCRC]

Last (or only) packet structure.

<STX>[lsPP][msPP][LL]←1 to 256-bytes of data →**<EOT>**[lsCRC][msCRC]

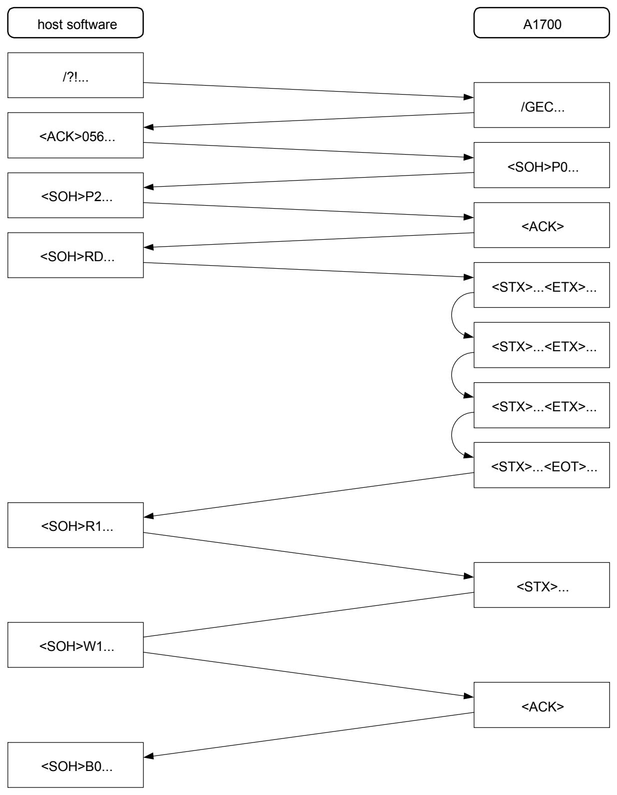
The “End of packet marker” in the last packet will be an ‘<EOT>’ code (04h)

5.6.0 Data stream termination

If, at any time during the transmission of packets, the A1700 receives an ‘<ESC>’ byte from the host software, the current data stream read command will be aborted by the A1700 following the transmission of the current packet.

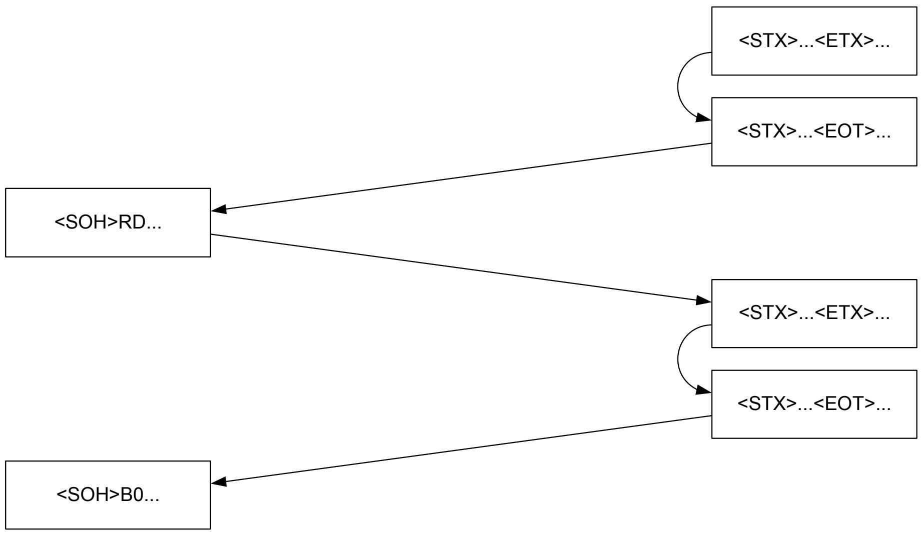
Note that this mechanism is only supported by the A1700’s remote communications port.

6.0 Example communication sessions

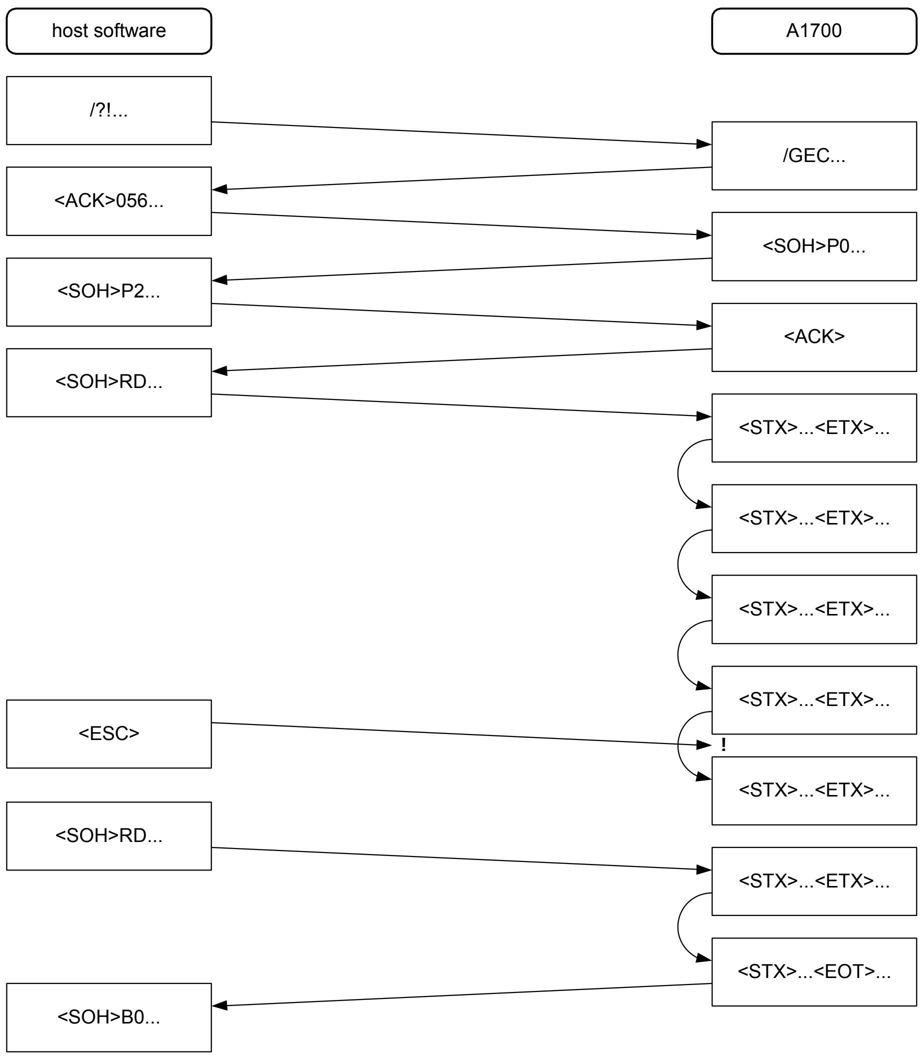


An example of a communications session using standard ‘R1’ and ‘W1’ commands along with the ‘RD’ command.

Upon the host software receiving the last packet denoted by the “<EOT>” both communication devices will have left data stream mode and reverted to standard mode.



If, whilst in data stream mode packets are lost or are discovered to be in error, the host software may request single or groups of packets from any point in the data identity by specifying the index and number of packets.



If during the data stream mode, the host software wishes to abort the transfer, it may send an asynchronous “<ESC>” (1Bh) code. The A1700 will exit the command following the completion of the current packets transmission and it will return to standard mode.

7.0 CRC algorithm

The data stream packet data integrity mechanism being used is a standard 16-bit CRC. The CRC-16 configuration that is currently used in the A1700 is as follows:

Name:	CRC-16
Width	16
Polynomial	A001h
Initial value	0000h
Reflected	Yes
XOR out with	0000h

The CRC is generated for each byte between and including the ‘Start of packet’ and the ‘End of data’ markers.

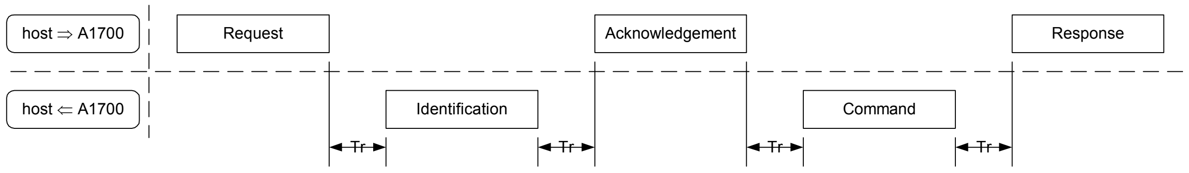
$$\underbrace{\textbf{<STX>[lsPP][msPP]}\text{[LL](1 to 256-bytes of data)}\textbf{<EOT>}}_{\text{CRC range}}\textbf{[lsCRC][msCRC]}$$

The 16-bit CRC is appended to the packet in little endian format i.e. least significant byte in the lowest memory location.

Refer to Appendix 1 for an example of the A1700 CRC generation routine.

8.0 Protocol timings

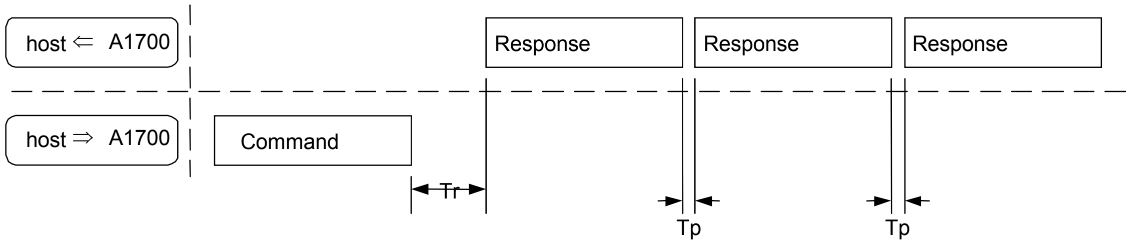
For communication timing, all of the [1] constraints will apply.

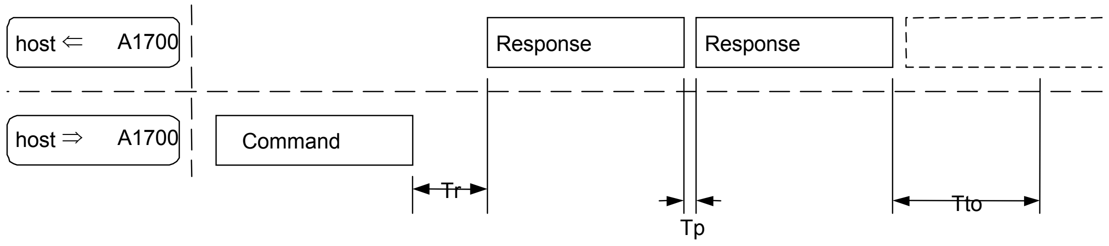


The additions to these rules are as follows:

The minimum response time to an ‘RD’ command will be as normal (Tr), however the time between the response data packets is (Tp)

$$60\text{ms} \le Tp \le 120\text{ms}$$





If the host software does not receive a response packet without having first received an <EOT> end-of-packet marker, after Tto, then it must consider that the command has either aborted or communications has been lost.

It is recommended that Tto be configurable in the host software and have a minimum value of 3000ms.

9.0 Related or generated documents

Reference	Title
[1]	IEC62056-21 Electricity metering – Data exchange for meter reading, tariff and load control – Part21: Direct local data exchange
[2]	102M050 – OPTICAL COMMUNICATIONS PROTOCOL
[3]	104M023 – DATA IDENTITIES FOR THE VISION METER
[4]	102M139 – LOAD PROFILE FORMAT FOR THE A1700 METER

10.0 Appendices

10.1.0 Appendix 1 – CRC method

Table Generation

```
#define MAX_CRC_TABLE_SIZE (256)

static uint16_t crcTable[MAX_CRC_TABLE_SIZE];

void GenerateCrcTable()
{
    uint16_t inx, entry16, carry16, i;
    for(inx = 0; inx < MAX_CRC_TABLE_SIZE; inx++)
    {
        entry16 = inx;
        for(i = 0; i < 8; i++)
        {
            carry16 = entry16 & 1;
            entry16 = entry16 >> 1;
            if(carry16 != 0)
                entry16 = entry16 ^ 0xA001;
        }
        crcTable[inx] = entry16;
    }
}
```

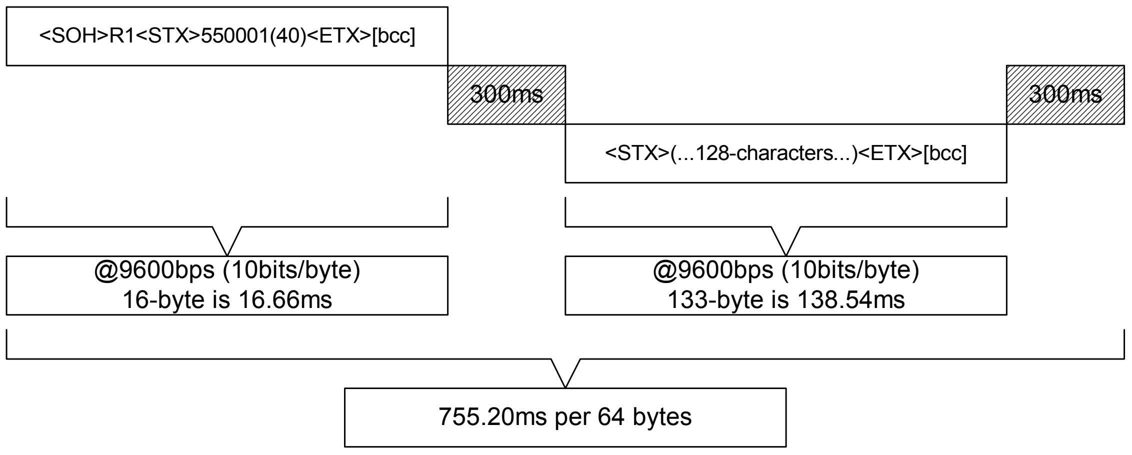
CRC Generation

```
typedef uint16_t    unsigned int;

uint16_t update_crc (uint16_t count, uint16_t crc16, char *ptr)
{
    uint16_t inx16;
    while (count)
    {
        inx16 = (*ptr ^ crc16) & 0x00FF;
        crc16 >>= 8;
        crc16 ^= crc16tab[inx16];
        ptr++;
        count--;
    }
    return (crc16)
}
```

10.2.0 Appendix 2 - *Theoretical* communication timing

(These timings 'exclude' any inter-characters delays and <u>assume</u> host software processing delays of 300ms. The specification states 200 to 15000ms.)

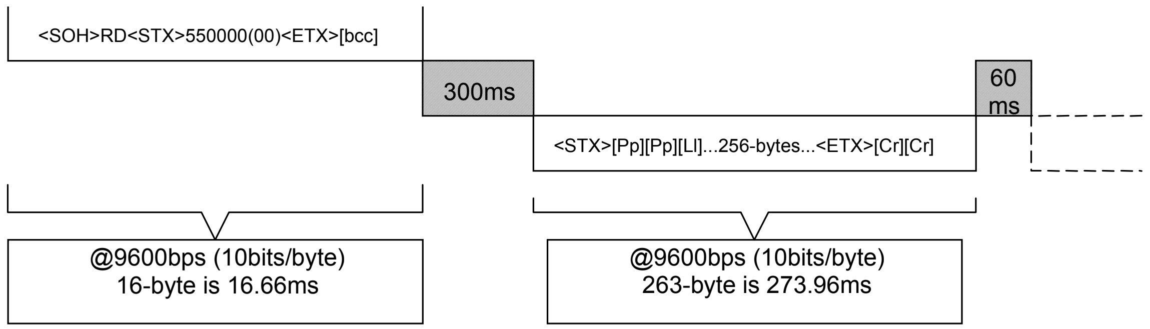


Current scheme:

For every 64-bytes read, the host software must send a sixteen-byte request. On top of this is a 400 to 600 ms turnaround delay.

Worst case: Read 64-bytes takes 755.20ms.

To read all of the load profile memory of 90,112-bytes would take 1408 packets of 64-bytes would take <u>17.72</u>-minutes. (Without retries)



Proposed scheme:

Since the communication configuration is now 8-data bits per byte, the packet can hold up to 256-bytes.

Worst case (single packet): Read 64-bytes takes 450.62ms.

Worst case (single packet): Read 256-bytes takes 650.62ms.

10.3.0 Appendix 3 - Design decisions

- Encoding all messages using 8-bit byte codes was not considered for the following reasons.

  1. To reduce the impact upon the code.
  2. Since 'standard' Read and Write packets do not contain length-data, the end of length marker is either ')' (29h) or the 128-character maximum. Since it would therefore be possible to have byte 29h or <ETX> (03h) in the data, an escape sequence would be required to permit the codes to be handled properly. This will require large modifications to the firmware to support standard commands in two different modes operating is different ways and is therefore high risk. Therefore 'standard' commands in 7 or 8-bit operating modes will only operate with 7-bit byte codes.

- The originally suggested scheme of having the new command have a set-up method to determine the number of packets to read was abandoned for the following reasons.

  1. This mechanism would only be of real benefit for the 550 read, for which there is already a mechanism i.e. 551 write then read. All other data identities will have a 'fixed' 128-byte packet geometry, which, as normal, would be known by the host software.
  2. This would add, unnecessary communication packets.

- Although the host software would know the number of 128-byte packets that it is to receive for any of the data identities it was considered 'useful' to have a 'last packet' marker.

- The original packet number (offset) structure was re-adopted since it would be easily capable of supporting the data ranges required.

- Through feedback, a packet range parameter has been adopted, so that a packet offset into the data block and number of packets can be specified in the data stream read command.

- Through feedback, an abort session command has been added to permit the host software to stop A1700 data transmission rather wait until the end of the session.

- Through feedback, encoding the length minus one of the data in the packet was thought acceptable as i) one or more bytes will always be read back and ii) using the length would incur an overhaed of one extra byte per packet.