```
                    ======================
                    D E L T A   L O G G E R
                    ======================


              ===================================
              P R O G R A M M E R S '   G U I D E
              ===================================


                 ------------------------------
                        Version 2.02
                 Applying to PROM versions 2.xx
                 ------------------------------


                     ---------------------
                      Delta-T Devices Ltd
                     128 Low Road, Burwell
                       Cambridge CB5 0EJ
                     ---------------------



REVISION HISTORY --------------------------------------------------------
Rev Date        By      Reason
-------------------------------------------------------------------------
 01 28 Mar 90   kaz     Creation
 02 23 Mar 99   kaz     Instr 65: memory full flags
                        Instr 65: corrected timestamp definitions
                        Instr 67, 68: reversed table and point numbers
                        Instr 80: corrected byte count value
                        Instr 105: corrected pos'n of timestamp bytes
                        #TYPE.FLAGS: added AC channel bit
                        #INTERVAL: corrected data trigger bits
  03 19 Jan 11   Chris F BYTE COUNTS AND CHECKSUMS: corrected checksum
description
-------------------------------------------------------------------------
```

=================
TABLE OF CONTENTS
=================

1. COMMUNICATING WITH THE LOGGER
   1.1 Establishing communications
   1.2 Auto-sleeping
   1.3 Sending instructions to the logger
   1.4 The command protocol

2. SUMMARY OF LOGGER OPERATIONS
   2.1 Setting up
   2.2 Interrogating
   2.3 Starting / stopping logging
   2.4 Collecting data
   2.5 Erasing data

3. THE INSTRUCTIONS

4. INTERPRETATION OF LOGGED DATA

5. THE CHANNEL CONFIGURATION ARRAYS
   5.2 Input channels
   5.3 Event trigger channel
   5.4 Warm up channel
   5.5 Control output channel

6. LINEARISATION TABLES

7. DATA FILE STRUCTURES
   7.1 .HFD files
   7.2 .BIN files

================================
1. COMMUNICATING WITH THE LOGGER
================================

1.1 Establishing communications
===============================

Proceed as follows:

1. The logger should be either asleep, or in the keypad main menu ("press any key if required.." message).
2. Send an RS232 level signal to the logger, any character. The logger wakes if asleep, and sends RDY$ (see below).
   NOTE: On powering up, the logger may take up to 100ms to establish a correct RS232C level. Noise on the output lines occurs during this period, and communications software may need to take account of such spurious signals.
3. Send an instruction code to the logger within 2s to confirm that the signal sent in 2. above was intentional. Otherwise the logger interprets the signal as noise and sleeps.
4. Send further instructions to the logger according the command protocol described below.

1.2 Auto-sleeping

The logger normally auto-sleeps if two consecutive minutes elapse without any input signals. Communications software may need to take account of auto-sleeping, especially if pauses are anticipated for operator input, for example:

- = instruction 88 disables the auto-sleep feature, but should be used with caution to avoid accidentally leaving the logger awake, and running down its batteries
- = instruction 13 (a null instruction) can be sent to the logger at regular intervals of less than one minute, to prevent the logger autosleeping
- = be prepared to wake logger after a long pause

1.3 Sending instructions to the logger

Instructions are single character codes. They must be sent to the logger in accordance with the command protocol described in the flowchart overleaf.

SENDING DATA TO THE LOGGER
Some instructions operate on data in the logger's "input buffer". Instruction 70 instructs the logger to prepare to receive data into the input buffer; its use is defined in the command protocol flowchart.
The data required in the input buffer is listed for each instruction in chapter 3.

REQUESTING DATA FROM THE LOGGER
Several instructions command the logger to output data. In general, the protocol for receiving data is as described under "other data request instructions" in the flowchart. The exceptions are instructions 86, 98, and 105 which are described in separate branches of the flowchart.

PROTOCOL CODES
In addition to the instruction codes, the following "protocol codes" have a special significance, as defined in the flowchart:

```
    OK$   = 13 (0Dh)
    NOK$  = 14 (0Eh)
    RDY$  = 15 (0Fh)
    XON$  = 17 (11h)
    XOFF$ = 19 (13h)
    BSY$  = 64 (40h)
```

1.4 The command protocol

The following notes refer to the flowchart below:

1. The sleep flag cleared on establishing communications. It is set by instruction 90, which causes the logger to power down into sleep mode at this stage.

2. The logger echoes the instruction code it has received to allow the host PC to check for transmission errors and take corrective action (ie abort instruction and re-send, see note 3 below).

3. OK$ causes instruction to be executed; NOK$ or (any character other than OK$) aborts an instruction.

4. The logger may be interrupted (eg to log data, or to reset its auto-sleep time-out) while data is being transmitted to it, and lose some incoming data while servicing the interrupt. BSY$ indicates that some data may have been lost, and re-transmission is required.

5. The logger echoes received data, enabling the host to check for transmission errors and take corrective action if appropriate.

6. Byte count and checksum information attached to data (see chapter 3) allows the host PC to confirm correct transmission of data, and take corrective action.

7. OK$ instructs the logger to move its data pointers and prepare to transmit new data when it next receives instruction 105.
   NOK$ (or any other character) instructs the logger to retain its current data pointers; the effect is that the same line of data is transmitted when instruction 105 is next executed.

COMMAND PROTOCOL FLOWCHART

```
host PC sends any character, to establish communications
 |
 |<<--------------------------------------------------------------------------+
 |                                                                            |
logger sends RDY$, to indicate it is ready to receive an instruction          |
 |->>CASE: sleep flag clear (note 1)                                          |
 |    host PC sends INSTRUCTION                                               |
 |    logger sends ECHO (note 2)                                              |
 |    host PC compares INSTRUCTION with ECHO                                  |
 |      |->>CASE: INSTRUCTION <> ECHO                                         |
 |      |    host PC sends NOK$ (note 3)                                      |
 |      |    logger aborts INSTRUCTION >>-------------------------------------|
 |      +->>CASE: INSTRUCTION = ECHO                                          |
 |           host PC sends OK$ (note 3)                                       |
 |             |->>CASE: INSTRUCTION NOT RECOGNISED BY LOGGER                 |
 |             |    logger takes no action >>---------------------------------|
 |             |->>CASE: INSTRUCTION 70, "PREPARE TO RECEIVE DATA" (note 7)   |
 |             |    logger sends RDY$                                         |
 |             |    host PC sends DATA$ followed by OK$                       |
 |             |      |->>CASE: LOGGER INTERRUPTED (note 4)                   |
 |             |      |    logger sends BSY$                                  |
 |             |      |    host PC sends OK$ (must re-send DATA$) >>----------|
 |             |      +->>CASE: DATA RECEIVED OK                              |
 |          |        logger sends ECHO$ followed by OK$ (note 5)           |
 |          |        host PC compares DATA$ with ECHO$                      |
 |          |        host PC sends OK$ >>-----------------------------------|
 |          |->>CASE: INSTRUCTION 86, "DIAGNOSTIC DATA DUMP"                |
 |          |   logger sends specified number of bytes >>-------------------|
 |          |->>CASE: INSTRUCTION 98, "SEND DATA FILE"                      |
 |          |   logger sends RDY$                                           |
 |          |   logger sends specified number of data                       |
 |          |     |->>CASE: .BIN format                                     |
 |          |     |   host PC sends XOFF$ to pause data transmission        |
 |          |     |     .. and XON$ to resume data transmission             |
 |          |     |   logger sends end of file block (.BIN format)          |
 |          |   host PC sends OK$ >>----------------------------------------|
 |          |->>CASE: INSTRUCTION 105, "SEND LINE OF LOGGED DATA"           |
 |          |   logger sends RDY$                                           |
 |          |   logger sends DATA$ followed by OK$                          |
 |          |   host PC performs byte-count/checksum calculation (note 6)   |
 |          |     |->>CASE: BYTE-COUNT/CHECKSUM ERROR DETECTED              |
 |          |     |   host PC sends NOK$ (note 12) >>-----------------------|
 |          |     |->>CASE: NO TRANSMISSION ERROR DETECTED                  |
 |          |        computer sends OK$                                     |
 |          |        logger increments data pointers (note 7) >>------------|
 |          |->>CASE: OTHER INSTRUCTION REQUESTING DATA                     |
 |          |   logger sends RDY$                                           |
 |          |   logger sends DATA$ followed by OK$                          |
 |          |   host PC performs byte-count/checksum calculation (note 11)  |
 |          |   host PC sends OK$ or NOK$ >>--------------------------------|
 |          +->>CASE: OTHER INSTRUCTION, REQUESTING NO DATA FROM LOGGER     |
 |              logger executes INSTRUCTION >>------------------------------+
 +->>CASE: sleep flag set (note 3)
     logger powers down to sleep mode >>-------------------------------->> END
```

```
                    ===============================
                    2. SUMMARY OF LOGGER OPERATIONS
                    ===============================
```

2.1 Setting up

CONFIGURING THE LOGGER
 1. INITIALISATION Instruction 72 resets the logger's variables in preparation for a new configuration, and reads in an experiment name and password from the input buffer
 2. CHANNEL CONFIGURATION Instruction 73 reads in a set of channel configuration variables for a specified channel from the input buffer; repeat instruction 73 for each non NUL channel
 3. USER DEFINED LINEARISATION TABLES Instruction 67 loads linearisation tables one element at a time.
 4. FINALISING CONFIGURATION Instruction 74 performs calculations on the channel configuration variables in preparation for logging.

SETTING THE LOGGER'S CLOCK   Instruction 71.

PRINTER INITIALISATION SEQUENCE   Instruction 82.

OVERWRITE MODE - Instruction 96.

2.2 Interrogating

GENERAL STATUS REPORT
 1. Instruction 65 returns the following information:
    - PROM version
    - Battery voltage

- Memory allocated and number of stored readings for each data type (compressed format)
- Minimum data storage interval for timed data (ie time interval between lines of timed data)
- Malfunction reports: battery failure and memory filled
- Experiment name and password
- Date-times: started logging, stopped logging, first stored timed data, next timed data to be output
- Logger's date-time format, US or European
- Overwrite mode, whether enabled
- Current date and time

2. Instruction 78 returns the following information for each data type:
   - Number of stored data
   - Number of data previously output by the logger
   - Date-time of first stored data
   - Date-time of next data to be output (timed data), or last data which has been output (event triggered data)

CHANNEL STATUS AND SENSOR MALFUNCTION REPORT
1. Instruction 81 returns the numbers of all non NUL channels.
2. Instruction 101 returns a channel status report. In the case of input channels, a sensor malfunction report accompanies the reading.
3. Instruction 87 turns on and resets warm-up relays.

LOGGING CONFIGURATION
1. Instruction 81 returns channel numbers of all non NUL channels.
2. Instruction 80 returns the channel configuration variables for a selected channel.
3. Instruction 68 returns a linearisation table element.

DELETING MALFUNCTION REPORT   Instruction 68.

2.3 Starting / stopping logging

IMMEDIATE START - Instruction 75.

TIMED START - Instruction 76.

STOP LOGGING - Instruction 78.

2.4 Collecting data

Logged data can be retrieved from the logger either as a file dump (binary .BIN or ASCII .PRN), or in discrete hex format (.HFD) packets with attached byte counts and checksums, allowing data integrity to be checked and re-transmission to be requested in the event of transmission error.
  Refer also to chapter 4 for details on how to reconstruct the data output from the logger.

INITIALISATION
Before requesting data (file dump or HFD), the logger's data pointers must be initialised:
1. Instruction 106 selects the TIMED, TRIG/61 or TRIG/62 data
2. Instruction 84 sets data output pointers to first data resident in memory if re-send of previously collected data is required; otherwise data output follows on from any previously transmitted data.

HEX FORMAT DATA PACKETS
1. HEADER INFORMATION use the following intructions:

65	general logger status, including basic TIMED data storage interval
78	data status, date-times of first TIMED data and last previously transmitted TIMED data,
79	data sequence
103	conversion factors

```
       104  zero offset
       108  label, engineering unit, sensor type code
       110  maximum recorded values
       111  minimum recorded values
 2. LOGGED DATA  Instruction 105 to request a line of data; send OK$ to
    advance data pointers after each line or NOK$ if re-send required.

FILE DUMP, .BIN or .PRN FORMAT
 1. FORMAT and NUMBER OF DATA  Instruction 97
 2. REQUEST DATA  Instruction 98
 3. HANDSHAKING  XON/XOFF for .BIN format only
 4. INTEGRITY CHECK  Instruction 99 returns bytecount and checksum for
    previously transmitted .BIN format data


2.5 Erasing data
================

ERASING ALL DATA
Instruction 83; should not be used during the course of logging.

ERASING DAT WHILE LOGGING
 1. SELECT DATA TYPE  Instruction 106
 2. ERASE DATA  Instruction 107:
     - TIMED DATA  Erases data as far as the last successfully output
       line of data
     - EVENT TRIGGERED DATA  Erases all data of selected data type,
       irrespective of whether it has been already output; take care !
```

3. THE INSTRUCTIONS

Instructions are single character codes, which are sent to the logger in accordance with the communications protocol described in chapter 1.
In this section, each instruction is described by its code in decimal, hex, ASCII, and the FORTH word executed by the logger on receipt of the command.

Some instructions operate on data previously sent to the logger and stored in its "input buffer". This is accomplished by means of instruction 70 ("F"), in accordance with the protocol described in chapter 1. The data required in the input buffer is listed for each instruction.

Some instructions request data from the logger. The logger outputs data in accordance with the protocol described in chapter 1 and the format of output data is listed for each instruction.

BYTE COUNTS AND CHECKSUMS
Byte count is represented by 2 hex digits at the beginning of a data string, and equals the number of data bytes in the string, excluding the 2 characters of the byte count itself, and the 4 character checksum at the end of the string.
Checksum is represented by 4 hex digits at the end of the data string; its value is equal to the arithmetical sum of the ASCII codes of all the preceding characters in the string, including the characters representing the bytecount, but NOT the characters representing the checksum itself.

DATA REPRESENTATIONS
Unless otherwise indicated:
```
  = STRINGS comprise sequences of printable ASCII characters (ie codes
    32 to 126)
  = INTEGERS may be 8- 16- or 32- bit; unless otherwise indicated,
    represented by 2, 4 or 8 hex digits respectively.
      16-bit integers may represent compressed format values, as
    described in chapter 4.
  = DATE-TIMES are represented by strings of 12 or 14 decimal digits,
    representing date and time as follows:
     BYTES
      1-2     month, 1-12
      3-4     day of month, 1-31
      5-6     unused
      7-8     hour, 0-24
      9-10    minute, 0-59
     11-12    second, 0-59
     13-14    .01 second (optional), 0-99
  = CHANNEL NUMBERS 1 to 64, are represented by 0 to 3Dh
```
Exceptions are:
Instruction 86 (56h, "V", BIN.DUMP)
Instruction 98 (62h, "b", SERIAL.DUMP)
Instruction 101 (65h, "e", SERIAL.DISPLAY)

INSTRUCTION 13 (0Dh)
- a null instruction; the logger takes no action, the only effect being to reset the auto sleep time-out and keep the logger awake.

```
INSTRUCTION 65 (41h, "A", .STAT)
 - send general status information.
OUTPUT, 166 bytes:
   1-2    integer; byte count (A0)
   3-6    integer; 0000
   7-10   integer; PROM version number
  11-14   integer; PROM revision number
  15-18   integer; battery voltage
          bits 0-11 = battery voltage x 409.6
          bit 12, 1 => battery voltage > 10 volts
                  0 => battery voltage < 10 volts, calculate as above
  19-22   integer; logging status
          A1B2 => logging
          0000 => not logging
  23-34   3 x integer, compressed format;
          RAM installed and allocated to TIMED, TRIG/61, TRIG/62 data
  35-46   3 x integer, compressed format;
          number of stored TIMED, TRIG/61, TRIG/62 data
  47-50   integer; minimum sampling interval (interval between
          lines of data) for TIMED data:
          0001 => 1s,   0002 => 5s,   0003 => 10s, 0004 => 30s,
          0005 => 1m,   0006 => 5m,   0007 => 10m, 0008 => 30m,
          0009 => 1h,   000A => 2h,   000B => 4h,  000C => 12h,
          000D => 24h
  51-58   2 x integer;
          number of scanned channels for TRIG/61, TRIG/62 data
  59-60   integer; battery failed flag
          00 => battery OK
          01 => battery failed
  61-62   integer; memory full flags
          bits 0,1,2 => TIMED, TRIG/61, TRIG/62
          0 => memory OK
          1 => memory filled
  63-70   text; experiment name
  71-78   text; password
  79-90   date-time; started logging (if applicable)
  91-102  date-time; stopped logging (if applicable)
 103-114  date-time; first stored timed data stored in RAM (if
          applicable)
 115-126  date-time; next timed data to be output (if applicable)
 127-128  integer; logger's date-time format
          00 => European
          01 => US
 129-130  integer; overwrite mode
          00 => disabled
          01 => enabled
 131-142  date-time: time of next timed data to be logged
          (date digits are unused)
 143-146  unused
 147-158  date-time; current time
 159-162  unused
 163-166  integer; checksum

INSTRUCTION 66 (42h, "B", SET.DEFAULT)
 - install default configuration.




INSTRUCTION 67, (43h, "C", !TABLE)
 - install linearisation table value(s) from data in input buffer (refer to
chapter 6).
INPUT BUFFER, 20 bytes:
   1-4    integer;
          linearisation table number (permitted values 1-7)
   5-8    integer;
          FFFF => bytes 9-20 contain increment and bottom of table
   9-12   integer; increment
  13-20   integer; bottom of linearisation table
   1-4    integer; data point number (0-32)
   5-8    integer; linearisation table number (permitted values 0-7)
   9-16   integer; data point value

INSTRUCTION 68 (44h, "D", .TABLE)
 - send linearisation table value(s) specified by input buffer.
INPUT BUFFER, 8 bytes:
   1-4    integer;
          linearisation table number (permitted values 1-7)
   5-8    integer;
          FFFF => increment and bottom of table to be output
          otherwise => data point number
OUTPUT, 18 bytes:
   1-2    integer; byte count (10)
   3-6    integer; increment
   7-14   integer; bottom of table
  15-18   integer; checksum
   1-2    integer; byte count (0B)
   3-8    integer; data point value
   9-12   integer; checksum

INSTRUCTION 68 (45h, "E", .DATA.STATUS)
 - send data status information.
OUTPUT, 126 bytes:
   1-2    integer; byte count (78)
   3-26   3 x integer; numbers of stored TIMED, TRIG/61, TRIG/62

          data
  27-50   3 x integer; numbers of previously output TIMED,
          TRIG/61, TRIG/62 data
  51-122  3 x (2 x date-time); date-times of 1st-stored-data and
          next-data-to-be-output, for TIMED, TRIG/61, TRIG/62 data
          (for TRIG/61 and TRIG/62 data, the date-time of next-data-to-
          be-output is in fact the date-time of the last data already
          output)
 123-126  integer; checksum

INSTRUCTION 70 (46h, "F", $IN)
 - load data string to input buffer; refer to chapter 1 for details.

INSTRUCTION 71 (47h, "G", !TIME)
 - set real time clock from data in input buffer.
INPUT BUFFER, 12 bytes:
   1-12   date-time; current time to be written to clock.

INSTRUCTION 72 (48h, "H", START.CONF)
 - initialise for new configuration and install experiment name and password
from input buffer.
INPUT BUFFER, 16 bytes:
   1-8    text; experiment name
   9-16   text; password

INSTRUCTION 73 (49h, "I", !CCT)
 - configure a channel according to input buffer.
INPUT BUFFER,56 bytes:
   1-4    integer; channel number
   5-21   text; #STRING
  22-24   text;
  25-28   integer; #TYPE.FLAGS
  29-32   integer; #INTERVAL
  33-36   integer; #CONTROL.O/P
  37-40   integer; #FACTOR
  41-44   integer, compressed format; #OFFSET
  45-48   integer, compressed format; #LIMIT.MIN
  49-52   integer, compressed format; #LIMIT.MAX
  53-56   integer, compressed format; #LIMIT.CTRL

INSTRUCTION 74 (4Ah, "J", END.CONF)
```

```
 - finalise configuration.

INSTRUCTION 75 (4Bh, "K", TALK.START)
 - start logging immediately (equivalent to pressing START button).

INSTRUCTION 76 (4Ch, "L", TIMED.START)
 - start logging at date and time specified in input buffer.
INPUT BUFFER, 12 bytes:
   1-12   date-time; date-time when logging is to start.

INSTRUCTION 77 (4Dh, "M", (TRIG.START))
 - no useful function in PROM 2.

INSTRUCTION 78 (4Eh, "N", STOP.LOG)
 - stop logging.

INSTRUCTION 79 (4Fh, "O", .*ORDER)
 - output data sequence for data type (TIMED, TRIG/61, TRIG/62)
previously specified by instruction 106.
OUTPUT, 2n+6 bytes:
       1 - 2        integer; byte count
       2 - 2n+2     n x integer; channel numbers
    2n+3 - 2n+6     checksum
(n = number of channels configured for selected data type)

INSTRUCTION 80 (50h, "P", .CCT)
 - output channel configuration of channel specified in input buffer.
INPUT BUFFER, 4 bytes:
   1-4    integer; channel number
OUTPUT, 58 bytes:
   1-2    integer; byte count (34)
   3-19   text; #STRING
  20-22   text; ; spaces
  23-26   integer; #TYPE.FLAGS
  27-30   integer; #INTERVAL
  31-34   integer; #CONTROL.O/P
  35-38   integer; #FACTOR
  39-42   integer, compressed format; #OFFSET
  43-46   integer, compressed format; #LIMIT.MIN
  47-50   integer, compressed format; #LIMIT.MAX
  51-54   integer, compressed format; #LIMIT.CTRL
  55-58   integer; checksum

INSTRUCTION 81 (51h, "Q", .CCTORDER)
 - output channel numbers of all active channels.
OUTPUT, 2n+6 bytes:
       1 - 2        integer; byte count
       3 - 2n+2     n x integer; channel numbers of non NUL channels
    2n+3 - 2n+6     integer; checksum
(n = number of non NUL channels):

INSTRUCTION 82 (52h, "R", !CTRL.SEQ)
 - store printer page width and printer initialisation sequence from
input buffer.
INPUT BUFFER - 4n+8 bytes:
       1 - 4        integer; printer page width
       5 - 8        integer; number of control characters for printer

                    initialisation, n
       9 - 4n+8     n x integer; control code sequence

INSTRUCTION 83 (53h, "S", INITIALIZE)
 - erase all data in RAM.

INSTRUCTION 84 (54h, "T", *PAGE.INITIALISE)
 - initialise pointers for data transmission to first non-erased item of
data; data type (TIMED, TRIG/61, TRIG/62) previously specified by
instruction 106.
```

```
INSTRUCTION 85 (55h, "U", DELETE.REPORT)
 - delete malfunction report.

INSTRUCTION 86 (56h, "V", BIN.DUMP)
 - diagnostic binary memory dump; page, address and quantity of data
according to input buffer.
INPUT BUFFER, 12 bytes:
   1-4    integer; page
   5-8    integer; start address
   9-12   integer; number of bytes, n
OUTPUT, n bytes:
   1-n    binary image of logger's memory contents.

INSTRUCTION 87 (57h, "W", SET.RELAYS)
 - reset or force warm-up relays on, for channels status report.
INPUT BUFFER, 4 bytes:
   1-4    0000 => resets warm-up relays
          0001 => forces warm-up relays on
Not available in PROM version prior to 2.08.

INSTRUCTION 88 (58h, "X", SET.AUTOSLEEP)
 - disables or enables autosleep function according to input buffer.
INPUT BUFFER, 4 bytes:
   1-4    0000 => disable autosleep
          0001 => enable autosleep

INSTRUCTION 90 (5Ah, "Z", SET.SLEEP)
 - set sleep flag; logger powers down to sleep mode after transmitting
RDY$.

INSTRUCTION 96 (60h, "`", !CYCLE)
 - enable or disable overwrite mode according to input buffer.
INPUT BUFFER, 4 bytes:
   1-4    0000 => disable overwrite mode
          0001 => enable overwrite mode

INSTRUCTION 97 (61h, "a", SELECT.FORMAT)
 - specify format and quantity of data for continuous data output
according to input buffer.
INPUT BUFFER, 12 bytes:
   1-4    0000 => .PRN format
          0001 => .BIN format
   5-12   integer; number of data to be output

INSTRUCTION 98 (62h, "b", SERIAL.DUMP)
 - data file output; format and number of data previously specified
using instruction 97.
OUTPUT, n data in .BIN or .PRN format; see 7.2 for structure of .BIN
format data.

INSTRUCTION 99 (63h, "c", .CHECKSUM)
 - send byte count and checksum for previous .BIN format data
transmission (either using instruction 98, or logger's keypad, or auto-
printing).
OUTPUT, 18 bytes:
   1-2    integer; byte count, 0B
   3-10   integer; byte count for previous .BIN transmission
  11-14   integer; checksum for previous .BIN transmission

INSTRUCTION 100 (64h, "d", SET.AUTO.PRINT)
 - enable or disable auto-printing, according to input buffer. Format of
data output as previously specified using instruction 97.
INPUT BUFFER, 4 bytes:
   1-4    integer;
          0000 => disable auto-printing
          0001 => enable auto-printing

INSTRUCTION 101 (65h, "e", SERIAL.DISPLAY)
 - send channel status and malfunction report for channel specified in
```

```
input buffer.
INPUT BUFFER, 4 bytes:
   1-4    integer; channel number
OUTPUT, text string
   1-32   string;
          channel number, label, sensor type code, current value,
          engineering units
  32-..   string;
          sensor malfunction report, if appropriate

INSTRUCTION 102 (66h, "f", .TYPE.FLAGS)
 - output #TYPE.FLAGS in data sequence order; data type (TIMED, TRIG/61,
TRIG/62) previously specified by instruction 106.
OUTPUT, 4n+6 bytes:
      BYTES         INTERPRETATION
      1 - 2         integer; byte count
      3 - 4n+2      n x 16-bit integers;
                    #TYPE.FLAGS for channels in data sequence order
   4n+3 - 4n+6      integer; checksum
(n = number of channels configured for selected data type)




INSTRUCTION 103 (67h, "g", .FACTOR)
 - output #FACTOR in data sequence order; data type (TIMED, TRIG/61,
TRIG/62) previously specified by instruction 106.
OUTPUT, 4n+6 bytes:
      1 - 2         integer; byte count
      3 - 4n+2      n x integer;
                    #FACTOR for channels in data sequence order
   4n+3 - 4n+6      integer; checksum
(n = number of channels configured for selected data type)

INSTRUCTION 104 (68h, "h", .OFFSET)
 - output #OFFSET in data sequence order; data type (TIMED, TRIG/61,
TRIG/62) previously specified by instruction 106.
OUTPUT, 4n+6 bytes:
      1 - 2         integer; byte count
      3 - 4n+2      n x integer, compressed format;
                    #OFFSET for channels in data sequence order
   4n+3 - 4n+6      integer; checksum
(n = number of channels configured for selected data type)

INSTRUCTION 105 (69h, "i", .DATA)
 - output next row of data in data sequence; if OK$ received at end of
transmission increment data pointer, otherwise otherwise keep previous
position (ie re-transmit same data next time instruction 105 is
executed).
OUTPUT, (timed data) 4n+6 bytes:
      1 - 2         integer; byte count
      3 - 4n+2      n x integer, compressed format;
                    n data in data sequence order
   4n+3 - 4n+6      integer; checksum
      1 - 2         integer; byte count
      3 - 4         unused;
      5 - 18        date-time; date-time of line of data
     19 - 4n+18     n x integer, compressed format;
                    n data in data sequence order
  4n+19 - 4n+22     integer; checksum
(n = number of data in the line of data)

INSTRUCTION 106 (6Ah, "j", DATA.SOURCE.SELECT)
 - select data type and set data page according to data type specified
in input buffer.
INPUT BUFFER, 4 bytes:
   1-4    integer;
```

```
          0000 => TIMED
          0001 => TRIG/61
          0002 => TRIG/62

INSTRUCTION 107 (6Bh, "k", DATA.READ.OK)
 - partially erase data, of data type previously specified by
instruction 106.
  For timed data, the block of data most recently collected is erased.
  For event triggered data, all data of the specified type is erased.







INSTRUCTION 108 (6Ch, "l", .STRING)
 - output 4-byte sections (specified by input buffer) of #STRING in data
sequence order (see instruction 79); data type (TIMED, TRIG/61, TRIG/62)
previously specified by instruction 106.
INPUT BUFFER, 4 bytes:
   1-4    integer; specifies the section of #STRING
          0000 => bytes 1-4 of #STRING
          0001 => bytes 5-8
          0002 => bytes 9-12
          0003 => bytes 13-16
          0004 => byte 17 of #STRING and 3 spaces
OUTPUT, 4n+6 bytes:
      1 - 2         integer; byte count
      3 - 4n+2      n x 4-byte text;
                    #STRING sections for channels in data sequence order
   4n+3 - 4n+6      integer; checksum
(n = number of channels configured for selected data type)

INSTRUCTION 110 (6Eh, "n", .MAX)
 - output maximum logged values in data sequence order (see instruction
79); data type (TIMED, TRIG/61, TRIG/62) previously specified by
instruction 106.
OUTPUT, 4n+6 bytes:
      1 - 2         integer; byte count
      3 - 4n+2      n x integer, compressed format;
                    #MAX for channels in data sequence order
   4n+3 - 4n+6      integer; checksum
(n = number of channels configured for selected data type)

INSTRUCTION 111 (6Fh, "o",  .MIN)
- output minimum logged values in data sequence order (see instruction
79); data type (TIMED, TRIG/61, TRIG/62) previously specified by
instruction 106.
OUTPUT, 4n+6 bytes:
      1 - 2         integer; byte count
      3 - 4n+2      n x integer, compressed format;
                    #MIN for channels in data sequence order
   4n+3 - 4n+6      integer; checksum
(n = number of channels configured for selected data type)
```

4. INTERPRETATION OF LOGGED DATA

Data output from the logger reflects the way in which logged data is stored in the logger's memory. The terms defined below are used in descriptions of the instructions for retrieving data.

Logged data is stored sequentially in the logger's memory, with TIMED, TRIG/61 and TRIG/62 data in separate compartments.

LINES of DATA
Data output from the logger is presented in "lines". Each line contains data recorded nominally at a single point in time.

DATE-TIMES
Date-times are stored for each line of event triggered data.
  For TIMED data, date-times are not stored in the data stream; they must be reconstructed, using an initial date-time and basic data storage interval.

DATA SEQUENCE
Channel numbers are not stored in the data stream. Instead, data in each line is stored and output in fixed order by channel. "Data sequence" is the sequence of channel numbers, which determines which channel each item of data in a line belongs to.
  There is a separate data sequence for each of the three data types. For event triggered data, the data sequence contains channels in increasing numerical order.
  For timed data, the data sequence is calculated by the logger when it is configured: in general, frequently logged channels precede less frequent channels, and digital channels precede analogue channels.

COMPRESSED FORMAT
The logger stores all logged data as integer values in 2-byte (16-bit) compressed format words:

```
  bits 0-11    unsigned 12-bit integer (range 0 to 4095)
               NOTE: if bit 15 =1 (see below), then bits 0-1 are used to
               denote the type of fault detected:
                    00 => over-run
                    01 => noisy
                    10 => outside limits
                    11 => over-range
  bits 12-13   octal range; integer value in bits 0-11 to be multiplied by a
               power of 8:
                    00 => x 1
                    01 => x 8
                    10 => x 64
                    11 => x 512
  bit 14       sign: 0 => -ve
                     1 => +ve
  bit 15       flag: 0 => good data
                     1 => suspect data, type of fault indicated by bits 0-1
```

Compressed format is derived from the architecture of the logger's analogue circuitry, enabling a reading from a LINEAR ANALOGUE sensor to be stored economically with no loss of resolution. The output of the 12-bit A-D converter is stored in bits 0-11 and bit 14, bits 12-13 represent the octal gain setting of the preamplifier, and the compressed format word represents an integer number of microvolts.

For consistency of representation, all other data is stored in compressed format:

- = COUNTER and FREQUENCY: a compressed format word represents a number of counts
- = NON-LINEAR SENSORS: linearisation (and cold-junction referencing of thermocouples) performed using 32-bit integer arithemetic, and the resulting value converted into compressed format
- = AVERAGES: calculated using 32-bit integer arithmetic, and resulting value converted to compressed format

RANGE and RESOLUTION of COMPRESSED FORMAT
Compressed format represents integers in the range -2096640 to +2096640.

Resolution is nominally 12-bit (ie 1 part in 4096). Actual resolution changes by a factor of 8 at the octal range thresholds (4096, 32768, 262144), and numbers just exceeding an octal range threshold may be said to be represented with a resolution of only 1 part in 512.

When selecting or setting up sensors, it may be worth bearing in mind that the logger's performance is optimised for sensors whose output matches the octal ranges of the analogue circuitry.

CONVERSION OF INTEGER DATA INTO ENGINEERING UNITS
#FACTOR and #OFFSET convert stored data into engineering units according to the following formula:

$$\text{Value\_in\_engineering\_units} = \frac{\text{Stored\_integer\_value + \#OFFSET}}{\text{\#FACTOR}}$$

The formula applies equally to data from non-linear sensors. Each linearisation table requires associated #FACTOR and #OFFSET values. The logger performs a linearisation and the resulting value, representing engineering units according to the above formula, is stored in compressed format. For the logger's on-board linearisation tables, #OFFSET=0 and #FACTOR=100, giving the tables a resolution of 0.01 deg C.

5. THE CHANNEL CONFIGURATION ARRAYS

The logging instructions for each channel are coded in nine 64-element variable arrays (one element per channel). The FORTH names of these variables are: #STRING, #TYPE.FLAGS, #INTERVAL, #CONTROL.O/P, #FACTOR, #OFFSET, #LIMIT.MIN, #LIMIT.MAX, #LIMIT.CTRL.

```
5.1 Channel functions
=====================

#TYPE.FLAGS - bits 0-5 code for the channel function:
     0 => NUL channel
     1 => MALFUNCTION WARNING relay
     2 => CONTROL OUTPUT relay
     3 => WARM-UP relay
     4 => EVENT TRIGGER channel
    >4 => INPUT channel; bits 6-15 contain additional information for
input and warm-up channels. See 5.2 and 5.4 for further details.

#STRING is a 17 byte string of printable ASCII characters for
identifying each non NUL channel, comprising:
  bytes
   1-3    SENSOR TYPE CODE
   4-11   LABEL
  12-17   ENGINEERING UNIT

A NUL channel is completely specified by #TYPE.FLAGS.

A MALFUNCTION WARNING channel is completely specified by #TYPE.FLAGS
and #STRING.

The additional information required to specify other channel functions
is described in 5.2 - 5.5.


5.2 Input channels
==================

#TYPE.FLAGS - flags denoting input type etc:
bits 0-1       channel group number:
                00 => group: 1-15
                01 => group: 16-30
                10 => group: 31-45
                11 => group: 46-60
bits 2-3       CASE: bits 4-5 = 11
                    excitation current for resistance reading:
                     00 => 2uA
                     01 => 20uA
                     10 => 200uA
                     11 => 2000uA
               CASE: bits 4-5 = 01
                    counter type channel:
                     00 => Counter
                     01 => Frequency
                     10 => Digital
bits 4-5       input channel type:
                00 => not permitted
                01 => counter type channel, see above
                10 => voltage
                11 => resistance, see above
bits 6-7       data-compression:
                00 => no data compression, store all data
                01 => store average
                10 => store highest reading
                11 => store lowest reading
bit 8          1 => AC channel
bit 9          1 => thermocouple
               #CONROL.O/P indicates whether or not cold-junction
               reference is required, and from which channel
bit 10         1 => control output required
               #CONTROL.O/P contains number of associated output relay

               and polarity of switching
bit 11         1 => suppress autoranging
```

```
               octal range automatically selected to accommodate values
               of #LIMIT.MIN and #LIMIT.MAX
bits 12-15     linearisation table number:
                   0 => linear sensor
                 1-4 => number of user defined linearisation table
                 5-7 => illegal values
                 8-F => on-board linearisation tables
                 8-B => thermistors
                   8 => Fenwall 2K, -20 to +60 deg C
                   9 => Fenwall 2K252, -20 to +60 deg C
                   A => Fenwall 10K, -20 to +60 deg C
                   B => Fenwall 100K, -20 to +60 deg C
                 C-E => thermocuples
                   C => Iron/Constantan, -120 to +200 deg C
                   D => Chromel/Alumel, -120 to +200 deg C
                   E => Copper/Constantan, -120 to +200 deg C
                   F => Platinum resistance transducer PRT100,
                        -200 to +600 deg C

#INTERVAL
 - specifies conditions for data sampling and storage:
bits 0-3    storage frequency for TIMED data
bits 8-11   sampling frequency for TIMED data ( = bits 0-3 if data
            compression is not required)
bit 4       1 => reading triggered by event trigger channel 60
bit 5       1 => reading triggered by event trigger channel 61
 - coding of sampling and storage frequencies:
            0 => channel not scanned at regular intervals
            1 => 1s,  2 => 5s,  3 => 10s, 4 => 30s
            5 => 1m,  6 => 5m,  7 => 10m, 8 => 30m
            9 => 1h, Ah => 2h, Bh => 4h, Ch => 12h,  Dh => 24h

#CONTROL.O/P
  - specifies associated channels:
  = If control output is required (#TYPE.FLAGS bit 10 = 1):
    bits 0-5       number of associated control output channel
    bit 7          0 => close on rising input
                   1 => close on falling input
  = If input is a thermocouple (#TYPE.FLAGS bit 9 = 1):
    bits 8-13      number of cold junction reference channel if required
    bit 15         0 => cold junction compensation not required
                   1 => see bits 8-13 for cold junction channel number

#FACTOR
 - a +ve conversion factor for converting stored data to engineering
units as in chapter 4; must be an integer value between 1 and 32767.
A #FACTOR value must be supplied for non-linear sensors:
  = on-board linearisation tables: 100
  = thermocouples: same #FACTOR value as cold-junction reference
    (assuming both are to be linearised in the logger)
  = user defined non-linear sensors: see chapter 6.

#OFFSET
 - zero offset for converting stored data to engineering units, see
chapter 4; a compressed format integer.
  A #OFFSET value should be supplied for non linear sensors; normally
this should be set to 0, but see chapter 6.

#LIMIT.MAX and #LIMIT.MIN
 - limits of acceptable readings; compressed format integer, derived
from engineering unit value using #FACTOR and #OFFSET as in chapter 4.
  Any readings outside the specified range are flagged by the logger as
"outside limits", are reported in the logger's malfunction report, and
switch the malfunction warning relay (if configured).

#LIMIT.CTRL
 - threshold for switching control output relay if control output is
required (#TYPE.FLAGS bit 10 = 1); compressed format integer, derived
from engineering unit value using #FACTOR and #OFFSET as in chapter 4.
```

5.3 Event trigger channel

The channels which constitute the data sequence for event triggered data are flagged in #INTERVAL of the respective input channels; see 5.2.
  The value of #FACTOR (between 1 and 32767) denotes the number of repetitions of an event triggered data sequence that are required per event.

5.4 Warm up channel

#INTERVAL determines the length of warm-up and interval between warm-ups:
```
     bytes 0-3   interval between warm-ups
     bytes 8-11  length of warm-up
```
The permitted intervals are coded as in 5.2 above.

5.5 Control output channel

The switching threshold and polarity are determined by the values of #LIMIT.CTRL and #CONTROL.O/P of the controlling input channel.

6. LINEARISATION TABLES

A non-linear input channel and the appropriate linearisation table are defined by bits 12-15 of #TYPE.FLAGS (see 5.2).
  The logger converts a number of "input units" (microvolts or counts) into an integer value by linear interpolation on a look-up table using 32-bit integer arithmetic.
  The resulting integer value is stored in compressed format. Conversion to engineering units as described in chapter 4. Appropriate #FACTOR and #OFFSET values must be supplied with non-linear sensors for this purpose.

A linearisation in the logger represents a +ve or -ve sloping monotonic curve:
  = DATA POINTS (D0 - D32): 33 data points are equally spaced in terms of data units, 32-bit integer values of input units.
  = BOTTOM of table (B): the number of data units which corresponds to the smallest number of input units, 32-bit integer value.
  = INCREMENT (I): the spacing of data points along the data unit axis, 16-bit integer value of data units.

```
     data units
            |
     B+32I -|- - - - - - - - - - - - - - - - - - - - - x
            |                                          |
            |                                          |
            |                                          |
     B+24I -|- - - - - - - - - - - - - - - - - - - x   |
            |                                      |   |
            |                                      |   |
            |                                      |   |
     B+16I -|- - - - - - - - - - - - - - x       |   |
            |                            |       |   |
            |                            |       |   |
            |                            |       |   |
      B+8I -|- - - - - - - - x           |       |   |
            |                |           |       |   |
            |                |           |       |   |
            |                |           |       |   |
        B -x                 |           |       |   |
            |                |           |       |   |
            |                |           |       |   |
           -|----------------|-----------|-------|---|---input units
            D0               D8          D16     D24 D32
```

SELECTION of #FACTOR and #OFFSET VALUES
On-board linearisation tables require a #FACTOR value of 100 and #OFFSET value of 0.
  Thermocouples require a #FACTOR value equal to #FACTOR of the corresponding cold junction reference.
  #FACTOR for other user defined linearisation tables can be any integer value in the range 1 to 32767, selected to give the required arithmetical resolution.
  #OFFSET for user defined linearisation tables can in most cases be set to 0 by selecting an appropriate value for bottom.

NEGATIVE SLOPING LINEARISATION CURVES (eg thermistors)
Increment I is -ve; bottom B is the maximum number of data units in the table (highest temperature in the case of a thermistor).

```
        B -x
            |
            |
            |
      B+8I -|- -x
            |   |
            |   |
            |   |
     B+16I -|- -|- - - -x
            |   |       |
            |   |       |
            |   |       |
     B+24I -|- -|- - - -|- - - - - -x
            |   |       |           |
            |   |       |           |
            |   |       |           |
     B+32I -|- -|- - - -|- - - - - -|- - - - - - - -x
            |   |       |           |               |
            |   |       |           |               |
           -|---|-------|-----------|---------------|--- input units
            D0  D8      D16         D24             D32
```

TABLES WITH FEWER THAN 33 DATA POINTS
The table must be truncated by filling up unused data points with the highest data point value in the table. A linearisation curve containing 25 data points:

```
     B+32I -|- - - - - - - - - - - - - - - - - - x
            |
            |
            |
     B+24I -|- - - - - - - - - - - - - - - - - - x
            |                                    |
            |                                    |
            |                                    |
     B+16I -|- - - - - - - - - - - - - - x       |
            |                            |       |
            |                            |       |
            |                            |       |
      B+8I -|- - - - - - - - - x         |       |
            |                  |         |       |
            |                  |         |       |
            |                  |         |       |
         B -x                  |         |       |
            |                  |         |       |
            |                  |         |       |
           -|------------------|-----------|-------|--- input units
            D0                D8          D16  D24-D32
```

```
                      =======================
                      7. DATA FILE STRUCTURES
                      =======================
```

7.1 .HFD files
==============

The Delta Logger software creates .HFD files by sending the following sequence of instructions to the logger, and writing the output strings directly to disk file (except for one modification). For detailed information about the contents of each line, refer to the instruction which generates it:

```
LINE 1: instruction 65 .STAT
 - general header information for file; thefollowing bytes modified:
    19-22   data type, 0001 => TIMED, 0002 => TRIG/61, 0003 => TRIG/62
   103-124  date-time of first data in file inserted, if data does not
            start at first data in RAM
   163-166  checksum modified in accordance with above changes
LINE 2: instruction 79 .*ORDER
 - data sequence for selected data type
LINES 3-7: instruction 108 .STRING, (0000 - 0004 in input buffer)
 - consecutive sets of 4-byte sections of #STRING for each channel in
   data sequence
LINE 8: instruction 103 .FACTOR
 - #FACTOR for each channel in data sequence
LINE 9: instruction 104 .OFFSET
 - #OFFSET for each channel in data sequence
LINE 10: instruction 111 .MIN
 - #MIN for each channel in data sequence
LINE 11: instruction 110 .MAX
 - #MAX for each channel in data sequence
LINES 12 to end of file: instruction 105 .DATA
 - consecutive lines of data, for channels in data sequence
```

7.2 .BIN files
==============

The following elements make up a .BIN file:

```
  = file header block:
     - fixed length section, 47 bytes
     - variable length section, 8 + 30n bytes, n defined at byte 48
  = any number of data blocks: each data block starting with a channel
    count byte n (in range 1-3Dh), followed by 2n bytes of data
  = if event triggered data, each data block is preceded with a date-
    time block, 8-bytes starting with FFh
  = end of file block, 64 ASCII 0's
```

Note that after the file header block, the different types of block can be differentiated by their first byte.

Date-times are represented by 7 bcd digits:

1	month, 1-12h
2	day of month, 1-31h
3	unused
4	hour, 0-24h
5	minute, 0-59h
6	second, 0-59h
7	.01 second, 0-99h

Numerical values are 8- or 16-bit binary integers.

All channel configuration variables and logged data presented in data sequence, as defined in chapter 4.

FILE HEADER BLOCK
55+30n bytes, where n is the number of channels, specified at byte 48:

1	integer; PROM version number
2	integer; PROM revision number
3 - 10	text; experiment name
11 - 16	date-time; start time
17 - 22	date-time; current time
23 - 28	date-time; first timed data in file
29	integer; interval between lines of timed data, coded as in 5.2, #INTERVAL
30	integer; data type: 1 => TIMED, 2 => TRIG/61, 3 => TRIG/62
31	integer; logger's date-time format: 0 => European, 1 => US
32 - 47	unused
48	integer; channel count, n
49 - 48+n	n x integer; channel numbers in data sequence
49+n	integer; channel count, n
50+n - 49+18n	n x (17 bytes text); #STRING
50+18n	integer; channel count, n
51+18n - 50+20n	n x integer; #TYPE.FLAGS
51+20n	channel count, n
52+20n - 51+22n	n x integer; #INTERVAL
52+22n	channel count, n
53+22n - 52+24n	n x integer; #FACTOR
53+24n	channel count, n
54+24n - 53+26n	n x integer, compressed format; #OFFSET
54+26n	channel count, n
55+26n - 54+28n	n x integer, compressed format; minimum logged data, conversion to engineering units using #FACTOR and #OFFSET
55+28n	channel count, n
56+28n - 55+30n	n x integer, compressed format; maximum logged data, conversion to engineering units using #FACTOR and #OFFSET

DATA BLOCK
A "line" of data, comprising 2n+1 bytes, for a data block containing data from n channels.

1	integer; channel count
2 - 1+2n	n x integer, compressed format; data from n channels in data sequence order, conversion to engineering units using #FACTOR and #OFFSET

DATE-TIME BLOCK
Occurring in event triggered data only, 8 bytes:

1	integer; FFh
2 - 8	bcd date-time; date-time for line of triggered data

END OF FILE BLOCK
64 bytes:

1-64	nul (ASCII 0)

←@