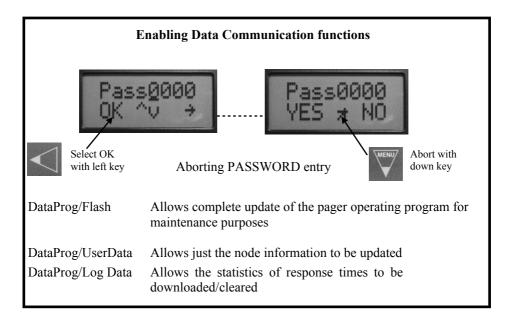# Programmers Handbook

One of a range of Alert-it Care Alarms available from:

This handbook is intended for engineers who are communicating with the P137 pager using the optional P146 USB interface.

Enabling Data Communication functions

Select OK with left key

Aborting PASSWORD entry

Abort with down key

DataProg/Flash	Allows complete update of the pager operating program for maintenance purposes
DataProg/UserData	Allows just the node information to be updated
DataProg/Log Data	Allows the statistics of response times to be downloaded/cleared

---

Data output formats (DataOut function enabled)

Communication Parameters

All data is communicated via a serial data stream of 9600baud, 8bit, no parity. XON/XOFF handshake The P146 and P152 USB interfaces all emulate serial ports with that setting, and can there fore be easily accessed via Hyperterminal or bespoke serial communication software.

Alarm Data Output

The occurance of an alarm produces an output string (in hexidecimal format)

Annvv<CR>   nn=00 when an alarm is cleared

(nn=node number, vv=alarm value, <CR> Carriage return 0x0D)

Node Test Data Output

In data output mode, the pager continuously output the data received from each node, in hexadecimal format as follows:

Signal Strength, Node address, Node Type, Node Data

Query Mode

All the store data (node and alarm) and pager information can be interrogated. Such query commands all start with a ?

Data Programming Overview

All programming uses the industry standard Intel format to transmit blocks of hexidecimal code (see below), with various characters being sent in response to confirm activity. Any errors produce the character Q followed by a hexidecimal value to define the error. For security all programming must be enabled by the Supervisor within the pager menu, and this can be restricted to prevent accidental changing of sensitive data. All programming begins with a command that starts with $. See below for a full description of coding.

Commands for Control of Pager via P146 USB Interface

Uploading data into Pager

$M   download memory block(s) (intel data follows but control characters ignored before:)
$U   download user data block(s) (intel data follows but control characters ignored before:)
$L   Log file (EEPROM) data follows (address 0x3000-0x3FF)
$R   Reboot program from RESET vector
$O   Programming data (only sent not received- see? below)

Answer-back

*   each time a full page is programmed (64 bytes)
%   sent when end of data block recieved and acted on
@   sent when a block of EEPROM has been correctly programmed

    Q   Error followed by 2 digit hex code for error
    •D0   Programming mode not enabled
    •E0   wrong data type in Intel word (not 0,1,2)
    •E1   non hex character in Intel word (>F)
    •E2   Intel Checksum failed
    •E3   Overflow error in cyclic input buffer
    •E4   Attempt to write into protected area
    •00   End of programming, system rebooted

Programming Node Block

sent to nodes in response to menu item "Program Node"
$O:04aaaa00D1D2D3D4CC<CR>
aaaa is an internal address of the registry
D1,D2   node address & type in hex
D2D4 16bit CRC seed
The programming mode displays "Saving" if an @ is recieved in response

Query informtion from Pager

?I   Send the product ID block (8 hex characters)
?V   Verbose information on the product
?U   download user data block as Intel code (includes node names and properties)
?L   download EEPROM data as Intel code (includes log data)

Alarm data

If the feature enabled (menu 1.6 DataOut)the alarm information is sent as:
Annxx<CR> nn=node number in hex, xx=alarm value in hex
Ann00<CR> is sent when the node alarm is cleared

P137 Data in response to ?I

-      Manufacturer
-      Product Code
-      Optional
-      Optional
-      Software Version
-      Software Issue
-      CRC seed (2 bytes)
-      Serial Number (3bytes)

Intel Data Block Format

:LLAAAATTD1D2D3........CC
L=length of data stream (D1...) excluding checksum
A=address of block start (0x3000 = start of EEPROM)
T=type (00=data, 01=end, 02=extend address)
D1..Dn Data bytes
C= Checksum (all data+checksum add to zero)
Any text not preceeded by a : will be ignored and hence the use of comment lines is possible, provided a colon never appears.

Serial Comms

XON-XOFF (3 character latency), 1 stop, 8bit, no parity, 9600baud

XOFF response only needed above address 0x1800 due to microprocessor halting for 9msec on block writing after 0x183F, 0x187F, 0x18BF, 0x18FF etc. 10 padding characters (any character except :) after these (or all) lines is another possibility.

End Protocol

After recieving data type 01 Intel Data or $R the pager will return to normal operation as if just powered and requires $M or $U to restart more data downloads. Note this will also reset the Program Enable flags so no further programming will be possible until re-enabled through the Supervisor menu

EEPROM Programming

For details of EEPROM usage see "US1095 P137 EEPROM Setup Definitions.doc"
The EEPROM is made to appear as memory in the address range 0x3000 onwards.
To create this EEPROM offset (0x3000) from baseline Intel code (start 0x0000) insert following line before the code.
:020000023000CC   ;sets base to 0x3000
Remember to reset the Base Address for any subsequent code.
:020000020000FC ;reset base to 0x0000

Engineering Error Codes

The following codes are shown on the front panel if the "SysError" function is enabled from the Supervisor Menu. This can help trace problems with a network and would normally only be useful to a trained engineer.

00	No error
F0	ASCII (a valid data but no code available yet)
F1	New node found (address not in registered list)
F2	Duplicated node (address registered but type is different)
F3	Input buffer overrun DV=1 when new data ready in buffer
F4	Unaddressed node detected (address=FF)
F5	DV was not cleared within the preamble time allocated (warning only)
F6	Watchdog timeout occured
F7	0.5 sec overrun compensated (due to RF noise)
F8	Display fault. Timeout waiting for busy signal
F9	Cyclic buffer over-run
FE	Test value only

For full details seeUS1080 Alert-it Serial Data Protocol.doc