EnviroTech LLC
1517 Technology Drive, Suite 101
Chesapeake, VA 23320, USA
Tel: (757) 549 8474
Fax: (757) 382 5012
Web: www.n-virotech.com
Email: info@n-virotech.com

Title	**Eco-Script Manual**
Issue	**2.9**
Status	**Approved**
Dist.	**General**
Circ.	**Commercially Confidential**
Author	**Paul Oakham / Mark Rawlinson / Quentin Lister**
Date	**26 January 2005**

Rev	**Change**	**Author**	**Date**	**Status**
1.0	Original	NAS	19 Mar 2002	Draft
1.1	Updated for version 2.0.27 SPS motor commands added	NAS	16 Apr 2002	Approved
1.2	Updated for addition of ESM Micro Mk II. Version 2.0.29	NAS	30 Apr 2002	Approved
1.3	Updated for version 2.1.3 firmware	NAS	31 May 2002	Approved
1.4	Updated for version 2.1.4 firmware	GFPO	31 May 2002	Approved
1.5	Updated for version 2.1.6 firmware	GFPO	28 Jun 2002	Approved
1.6	Updated for version 2.1.7 firmware	GFPO	4 Jul 2002	Approved
1.7	Updated for version 2.1.8 firmware	GFPO	22 Nov 2002	Approved
2.0	Reformatted with additional text	MBR	19 May 2003	Approved
2.1	Updated for version 2.1.14 firmware	GFPO	30 Oct 2003	Approved
2.2	Updated for version 2.1.17 firmware	GFPO	6 Jan 2004	Approved
2.3	Minor formatting	MBR	15 May 2004	Approved
2.4	Updated to include error messages	QL	28 Sept 2004	Approved
2.5	Updated initserial command	MBR	12 October 2004	Approved
2.6	Updated reference to variables scope and lifetime	QL	15 Dec 2004	Approved
2.7	Updated to include saving and loading of variables	QL	6 Jan 2005	Approved
2.8	New SPS encoder commands added	MBR	7 Jan 2005	Approved
2.9	Script editor sections added to 1.0	MBR	26 Jan 2005	Approved

Copyright EnviroTech LLC

Contents

1.0 Introduction to Eco-Script

ESM based products are defined as the ESM-1 Plus & Micro, EcoLAB, AquaLAB, AutoLAB Mk II, Aqua Monitor Mk II and IUC-2. Each of these produced contains the ESM-1 electronic board set and firmware.

The ESM-1 series of devices take a new approach to data logging and control applications. Their high quality and high specification hardware design is uniquely coupled with a new system control language, Eco-Script. This is a significant change from the traditional and limiting set of user programmable variables. Eco-Script enables the ESM-1 logger to be used in a wide range of applications with infinitely programmable sampling and system control. This can include irregular timing within burst samples, many burst samples of different types within a sampling regime and incredible variability of timing and control functions. However, the Eco-Script can also implement any regime that can be configured via a conventional data logger.

Eco-Script is a simple control language designed specifically for remote and unattended data acquisition systems. A series of commands are compiled into a script and then placed in the script memory of the ESM. At least one script is required, but many can be uploaded. These may then be invoked manually at different times by the user or may “call” each other. This “nesting” allows the system to perform an infinitely variable and irregular burst cycle. For example the application may require a simple analogue burst sample every ten minutes, but once an hour burst sampling from a power-hungry device is also necessary. Three scripts would be required to solve this problem: Script 1 would call Script 2 five times and then Script 3 once at ten minute intervals. Script 2 would be the normal analogue sensor burst sample and Script 3 would include the power hungry device. If the data from the power-hungry device were now required every 30 minutes the modification is a simple one line edit of Script 1. Further scripts could be readily added and either called by Script 1 or be implemented as sub-tasks of Scripts 2 and 3.

The ESM has what can be considered a hard-drive in the form of a Compact Flash (CF) card or flash memory in the case of the Micro version. This “drive” is a logical DOS drive in that it supports DOS file naming conventions, directories, etc., and for the compact flash card can be read directly as a DOS drive if plugged into a PC card slot on a laptop PC – for example. The ESM stores data on the CF card or flash memory chip, but as well as this the chemistry scripts or sampling routines are also stored in memory. These scripts are stored in a directory called

\store\cf\scripts	for the Plus version
\store\flash\scripts	for the Micro version

Eco-Script also includes a number of built-in interfaces for “industry standard” sensors. These include our own ***NAS-2E*** In-situ Nutrient Analyzer. Using ESM and Eco-Script technology comprehensive environmental monitoring systems can be rapidly assembled and reconfigured to suit changing application requirements.

1.1 Editing Eco-Script files

Eco-Script routines can be either edited while resident on the ESM CF card (or flash memory) or while on your PC hard-drive. The recommended method is to edit your scripts using a PC text editor application, such as Notepad or ConTEXT, and save them as text format files, with the (extention *.eco) on your PC hard drive (or another logical PC drive, such as a shared server drive).

Eco-Script routines are always run from The ESM’s CF card or flash memory (i.e. never from your PC hard drive). Therefore you need to upload your script to the CF card or flash memory. This is done via Hyper Terminal. Once uploaded to the CF card or flash memory directory (given above) The ESM can be told to run or “execute” the script from the CF card or flash memory.

While editing your script it should be saved with a meaningful file name of up to eight characters. As the ESM file names are eight bytes you must designate an eight-byte name or it will be truncated.

While editing it is recommended that you use a header to enable you to keep track of differences between versions as you develop your routines. It is also highly recommended that you comment your files fully so that they are easy to maintain. Everything following the # character on any single line will be treated as a comment rather a script command.

You can edit scripts on a PC using a text editor and ConTEXT is recommended. Then upload the script to the \scripts directory on the CF card or flash memory using ESM Manager 2 Software.

Once the script has been uploaded the script may be run from ESM Manager 2.

1.2 Installing ConTEXT

ConTEXT is a language sensitive programmer’s editor, distributed with a freeware licence. We have written an Eco-Script language definition file that will highlight the ESM script files. Main controller commands are **blue**, variables and switches are **red**, SPS commands are **purple** and comments are **green**. All other text is black. A screen shot of a script loaded in ConTEXT is shown below.

Install ConTEXT and then copy the “**Eco-Script.chl**” file to

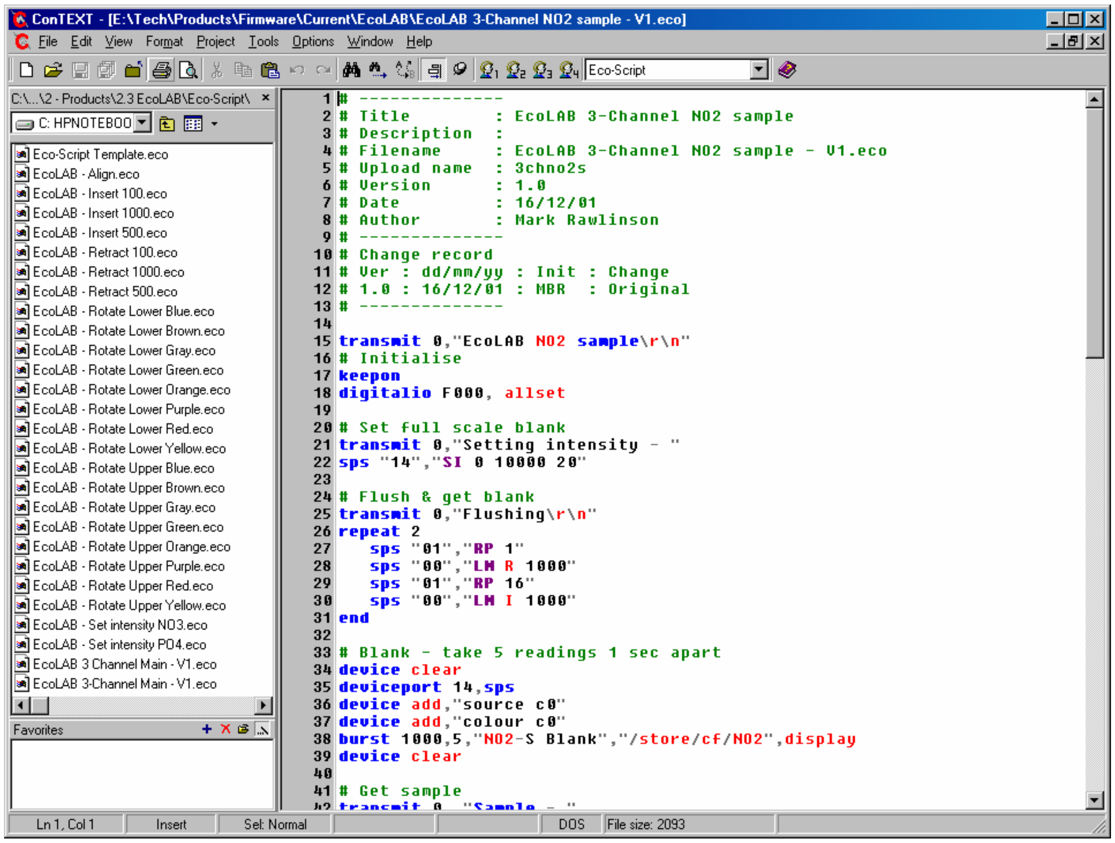
C:\Program Files\ConTEXT\Highlighters

Run ConTEXT and open a sample ESM script file (Eco-Script) for editing. From the Tools menu select Set Highlighter and from the drop-down list select Eco-Script. The script file in the main screen should be highlighted just like the screen shot below.

You can then edit and save the file to suit your own needs and files will be saved with the default extension is *.eco.

Once uploaded to the /scripts directory one script may “call” another. For example a flushing routine that is common to all routines could be uploaded as “comflush” and then called from each script as:

```
execute “/store/cf/scripts/comflush”
```


ConTEXT - [E:\Tech\Products\Firmware\Current\EcoLAB\EcoLAB 3-Channel NO2 sample - V1.eco]
File Edit View Format Project Tools Options Window Help
Eco-Script
C:\...\2 - Products\2.3 EcoLAB\Eco-Script\
C: HPNOTEBOO
Eco-Script Template.eco
EcoLAB - Align.eco
EcoLAB - Insert 100.eco
EcoLAB - Insert 1000.eco
EcoLAB - Insert 500.eco
EcoLAB - Retract 100.eco
EcoLAB - Retract 1000.eco
EcoLAB - Retract 500.eco
EcoLAB - Rotate Lower Blue.eco
EcoLAB - Rotate Lower Brown.eco
EcoLAB - Rotate Lower Gray.eco
EcoLAB - Rotate Lower Green.eco
EcoLAB - Rotate Lower Orange.eco
EcoLAB - Rotate Lower Purple.eco
EcoLAB - Rotate Lower Red.eco
EcoLAB - Rotate Lower Yellow.eco
EcoLAB - Rotate Upper Blue.eco
EcoLAB - Rotate Upper Brown.eco
EcoLAB - Rotate Upper Gray.eco
EcoLAB - Rotate Upper Green.eco
EcoLAB - Rotate Upper Orange.eco
EcoLAB - Rotate Upper Purple.eco
EcoLAB - Rotate Upper Red.eco
EcoLAB - Rotate Upper Yellow.eco
EcoLAB - Set intensity NO3.eco
EcoLAB - Set intensity PO4.eco
EcoLAB 3 Channel Main - V1.eco
EcoLAB 3-Channel Main - V1.eco
Favorites
Ln 1, Col 1
Insert
Sel: Normal
DOS
File size: 2093

```
# --------------
# Title         : EcoLAB 3-Channel NO2 sample
# Description   :
# Filename      : EcoLAB 3-Channel NO2 sample - V1.eco
# Upload name   : 3chno2s
# Version       : 1.0
# Date          : 16/12/01
# Author        : Mark Rawlinson
# --------------
# Change record
# Ver : dd/mm/yy : Init : Change
# 1.0 : 16/12/01 : MBR  : Original
# --------------

transmit 0,"EcoLAB NO2 sample\r\n"
# Initialise
keepon
digitalio F000, allset

# Set full scale blank
transmit 0,"Setting intensity - "
sps "14","SI 0 10000 20"

# Flush & get blank
transmit 0,"Flushing\r\n"
repeat 2
   sps "01","RP 1"
   sps "00","LM R 1000"
   sps "01","RP 16"
   sps "00","LM I 1000"
end

# Blank - take 5 readings 1 sec apart
device clear
deviceport 14,sps
device add,"source c0"
device add,"colour c0"
burst 1000,5,"NO2-S Blank","/store/cf/NO2",display
device clear

# Get sample
transmit 0,"Sample - "
```

1.3 Scripting tips

Scripts can be used to simplify the ESM control language while preserving flexibility and direct hardware control. For example the linear stepper motor of EcoLAB, AutoLAB and AquaLAB may be retracted 500 steps using the Direct Command:

```
      sps "00","LM R 500"
```

But this is quite long-winded. However, there is a solution – uploading the following file allows you to simply run:

```
      r500
```

```
# r500
# Retract 500
# Upload name  : r500
transmit 0,"Retracting 500 - "
digitalio F000
wait 500
sps "00","LM R 500"
transmit 0,"done\n\r"
```

The script also forces The ESM to acknowledge when the program (i.e. motor movement) is complete

With EcoLAB the following file will rotate the valve to the upper red port – run it with “ured”

```
# ured
# Rotate port Upper Red
# Upload name  : ured
transmit 0,"Rotating - "
digitalio F000
wait 500
sps "01","RP 11"
transmit 0,"Upper Red (11)\n\r"
```

These “short-hand” files may also be called from other Eco-Script files:

e.g. `execute “/……/r500”`

1.4 Step-by-step scripting

Go ahead and design a few simple scripts (as above), upload them, run them and ensure you are happy with the edit, save, upload, run (and debug) cycle.

Then put The ESM to one side and design your logging, sampling or chemistry routines on paper, including calling names. A hierarchy diagram always helps here.

Then implement them as scripts via ConTEXT. Check them carefully as a print out.

Next upload the scripts from the lowest hierarchy – usually utilities – and test them one by one.

Do the same for the next level of hierarchy – usually chemistry routines – and test them too, including the calling to the utility scripts

Finally upload and test the highest level – normally the main logging loop.

2. ESM Scripting Language Reference

2.1 Procedure for entering and editing scripts

There are three ways to enter a script into the ESM:

1) Upload a finished scripts or set of scripts with ESM Manager 2 software

2) Writing the scripts directly to the Compact Flash card, by inserting it into a PC using a suitable adapter. Standard text editors, such as ConTEXT or Windows Notepad can then be used to edit them. Be sure not to use long file names.

3) Use the edit command in a terminal

   Open a terminal window and proceed as follows:

   *edit /store/eep/test … *OR* … edit test*

   if you are in the correct local directory

   A text editor will be displayed, which behaves in a similar way to the DOS edit command.

   To execute a script, type its name. e.g. to execute the script uploaded in the example above, type

   */store/eep/test … *OR* … test*

   if you are in the correct local directory.

2.2 Scriptable Commands

repeat

Parameters

reps	Number of times to execute the loop	0 - 9999999

Description

This statement executes a block of code multiple times. *reps* specifies the number of times that the repeat loop is executed. The special value of 0 indicates that the loop is executed indefinitely. Each repeat statement must have a matching *end* statement. E.g. *repeat 10*

if

Parameters

condition	The condition	Any valid condition

Description

This statement executes a block of code providing the *condition* is true. Each if statement must have a matching *end* statement. It may have an optional *else* statement.

else

Parameters

none

Description

When used in conjunction with the *if* statement, a block of code may be executed if the *condition* of the *if* statement is false.

while

Parameters

condition	The condition	Any valid condition

Description

This statement executes a block of code multiple times providing the *condition* is true. Each *while* statement must have a matching *end* statement.

end

Parameters

none

Description

Specifies the end of a block such as *repeat*, *if*, and *while*. E.g. *end*

return

Parameters

none

Description

Immediately returns from a script. This is useful for breaking out of a subscript in the event of a conditional.

variable

Parameters

name	The name of the variable to define

Description

Defines a variable. Variable *names* must be letters only and are case sensitive. Although only the first 8 letters are used to identify a variable, the names may be longer. The scope of the variable is restricted to the particular script and to that particular instance. E.g. subscripts called with execute do not have access to a parent scripts variables. E.g. *variable psupply* creates a variable called psupply for later use.

savevars

Parameters

filename	all current variables saved to *filename*

Description

This command saves all current variables to a file and allows them to be preserved through a power cycle. The value of variables persist through sleep mode, but not power-off. Also see “loadvars” command.

loadvars

Parameters

filename	variables loaded from *filename*

Description

This command loads all variables previously stored in the *filename* by the “savevars” command. The “loadvars” and “savevars” commands may appear in different scripts. A variable read from the *filename* will overwrite existing variables of the same name.

execute

Parameters

subscript	The filename of the subscript to execute

Description

Executes the subscript specified by *subscript*. It is important to note that when a subscript is called, everything is preserved, except for the device selection list. E.g. a subscript cannot affect its parent’s checkpoints, but it can change which devices are selected. E.g. *execute "/store/eep/example.scp"*

Parameters
cancel

Description
keepon sets the ESM to stay on through commands which should put it to sleep. E.g. *sleep*, *checkpoint*. This is useful if the power outputs are required at certain times. *keepon cancel* reverses the *keepon* command. At the start of a script, *keepon* is not enabled. A subscript can modify the *keepon* setting for its parent script. E.g. *keepon cancel*

checkpoint

Parameters

delay	The minimum delay, in seconds	1 - 9999999

Description
This command halts execution until the amount of seconds specified by *delay* has passed since the last time a *checkpoint* command was executed. This command is very useful for triggering something to happen every n seconds. The other time control commands are not accurate for this, since the time taken to process a command is unpredictable. If *delay* is set as 0, then *checkpoint* returns immediately, but a *checkpoint* is still triggered. E.g. *checkpoint 60*

force checkpoint

Parameters
none

Description
This command causes the next *checkpoint* command to return immediately, regardless of the amount of time since a checkpoint was passed.
E.g. *force checkpoint*

wait

Parameters

delay	The number of milliseconds to delay for	1 - 9999999

Description
This command blocks execution for the number of milliseconds specified by *delay*. E.g. *wait 1000*

sleep

Parameters

sleep	The number of seconds to sleep for	1 - 9999999

Description
This command blocks execution for the number of seconds specified by *sleep*. If the delay is long enough to put the ESM into standby mode without affecting timings, then the ESM is switched off for the sleep period. E.g. *sleep 10*

waituntil

Parameters

time	The time to wait until	HH:MM:SS
date	The optional date to wait until	DD/MM/YYYY

Description

waituntil 14:56:57 will block until 14:56:57 every day. *waituntil 13:00:00 15/07/2003* will block until 1PM on 15th July 2003. It will never block if that date and time has already passed.

shutdown

Parameters

none

Description

This command puts the ESM into standby mode indefinitely. When the ESM is powered on, scripts will not be executed, unless an *AUTOSCRIPT* is programmed.

digitalio

Alternative:
dio

Parameters

iobit	The i/o bit to change, or hex value to set	1 – 16 or HHHH
change	How to change the i/o bit	set, clear, toggle, allset

Description

See the section on Digital IO commands for a detailed description.

deviceport

Parameters

port	The port to use for devices	Depends on *type*
type	The type of port to use for the device	sps or com

Description

This command sets which port to use for subsequent devices. *Type* specifies the type of device. If *type* is *com*, then *port* represents the serial port number to include a serial device on. If *type* is *sps*, then *port* represents the sps address of the device to use. If a device is on a port, then this command must be called before the call to the device command. E.g. *deviceport 2, com*.

device

Parameters

command	What to do with the device selection list	clear, add, remove
device	The name of the device	

Description

This command modifies the device selection list (the list that specifies which devices are included in *samples* and *bursts*). If *command* is *clear*, then the device selection list is emptied. No *device* parameter is required for this. If *command* is *add* or *remove*, then *device* specifies the name of the device to add to, or remove from the device selection list. E.g. *device add, [Analog 0]* If a device requires once of the digital outputs to be turned on for it to work, then *device remove* or *device clear* must be used before turning the devices off, to prevent error messages being displayed.

sample

Parameters

samples	Number of samples to do	1 - 9999999
samplename	The name of the sample group	100 chars
storagepath	The full path where the data is stored	100 chars
display	Sends data via serial (optional)	display
noburst	Inhibits writing burst data (optional)	noburst

Description

This command takes samples from the devices in the device selection list, as soon as quickly as data becomes available. The string specified by *samplename* is saved along with the data, so that the sample may be identified. *storagepath* indicates where the data for the sample group is stored. This is the name of a directory. If the directory doesn't exist, then it will be created. The *noburst* parameter prevents the ESM from writing burst data, however synopsis data is still stored in the usual way.
E.g. *sample 100, "example sample", "/store/cf/test", display*

burst

Parameters

interval	Milliseconds between samples	20 - 9999999
samples	Number of samples to do	1 - 9999999
samplename	The name of the sample group	100 chars
storagepath	The full path where the data is stored	100 chars
display	Sends data via serial (optional)	display
buffered	Selects buffered burst mode (optional)	buffered
noburst	Inhibits writing burst data (optional)	noburst

Description

This command takes samples from the devices in the device selection list, as quickly as specified by *interval*. The string specified by *samplename* is saved along with the data, so that the sample may be identified. *storagepath* indicates where the data for the burst group is stored. This is the name of a directory. If the directory doesn't exist, then it will be created. If the *buffered* parameter is specified, then burst data is written to the location specified by *STREAMBUFFER* in */cfg/main.cfg* whilst the burst is being carried out. Once the burst is finished, the data is then written to the location specified by *storagepath*. With default settings, 6000 samples can be buffered (on all 8 channels). For example 3 channels could be sampled at 60Hz for 133 seconds. The *noburst* parameter prevents the ESM from writing burst data, however synopsis data is still stored in the usual way. Noburst and buffered cannot be used together.
E.g. *burst 100, 10, "example sample", "/store/cf/test", display, buffered*

dumpburst

Parameters

storagepath	The full path where the data is stored	100 chars

Description

Dumps the oldest sample to the UI port, and then erases it from the stream.
E.g. *dumpburst "/store/cf/test"*

initserial

Parameters

port	The serial port to initialise	Plus: 1 – 5 Micro: 1 – 2
baud	The baud rate to initialise at	110, 150, 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200
Parity	Parity setting n, e or o	For No parity, Even parity, Odd parity
Bits	Bits per character	8 or 7
Stop bits	Stop bits per character	1 or 2

Description

Initialises serial port specified by port, to the baud rate specified by baud. Optionally the parity, bits and stop bits may be specified
e.g. *initserial 5, 9600*
or *initserial 5,9600,7,e,1*
Caution should be used when initialising ports used for UI as specified in */cfg/main.cfg*.

transmit

Parameters

port	The port to transmit to	Plus: 0 – 5 Micro: 0 – 2
string	The string to transmit	100 chars

Description

Transmits *string* on *port*. String may contain chars \r, \n, \t, \e, \c, \xHH.
E.g. *transmit 1, "Test\r\n".* The special value of 0 for the port will cause the string to be transmitted on whichever port is specified by UIPORT in the main configuration file.

transmitvar

Parameters

port	The port to transmit to	Plus: 0 – 5 Micro: 0 – 2
var	The name of the variable to transmit	

Description

Transmits the variable *var* on *port*.

waitfor

Parameters

port	The port to wait on	Plus: 0 – 5 Micro: 0 – 2
timeout	The maximum seconds to wait	0-9999999
string	The string to wait for	100 chars

Description

Waits for a *string* to be received on a serial port. The special value of 0 for *timeout* indicates that the command should never timeout. E.g. *waitfor 2, 60, "hello\r\n"*

capture

Parameters

port	The port to capture from	Plus: 0 – 5 Micro: 0 – 2
path	The full path to store the capture	100 chars
chars	The maximum characters to capture	0-99999999
timeout	The maximum milliseconds to wait	0-99999999
pattern	The pattern to match	100 chars

Description

Logs raw data from a serial port to a file. This command can be called many times with the same path. A new file will be generated in that directory each time. The three parameters specify the conditions for stopping the capture. If *chars* is 0, then the capture is not limited by the number of characters received. If *timeout* is 0 then the capture is not limited by the amount of time elapsed. If the *pattern* parameter is omitted, then no terminating sequence is looked for.
E.g. *capture 2, "/store/cf/sercap", 10, 0, "\r\n"* will capture either 10 characters or until a carriage return – line feed pair is received from port 2 to */store/cf/sercap*. There is no limit on the amount of time this will wait.

sps

Parameters

address	The address of the SPS device	Any valid SPS address
command	The command to transmit	100 chars

Description

Transmits *command* to SPS device at *address*. E.g. *sps "0D", "RP 8"*

sps auto

Parameters

none

Description

Puts the SPS mode into automatic. The SPS mode is automatic by default at the start of each subscript. In automatic mode, when a command such as *sps "1F","ST 40000 10 100"* is sent, the script execution is stopped until the SPS device reports that either it succeeded, or it gave up trying.

sps manual

Parameters

none

Description

Puts the SPS mode into manual. In manual mode, the command returns immediately. In this mode, care should be taken not to send a command to the SPS device until it has finished.

waitsps

Parameters

address	The address of the SPS device	Any valid SPS address

Description

Waits until an SPS command has completed. This has no effect if the SPS mode is set to automatic, which is the default. E.g. *waitsps "01"* will wait until the SPS device with address 01 has completed its command. This command is very useful in circumstances where multiple SPS devices are used at the same time. For example if two colourimeters were attached, they could both be given their target temperatures immediately one after the other, then the script could wait for them both to finish, thus allowing them to seek to their temperatures in parallel.

adcrate

Parameters

rate	The ADC rate	1 – 100

Description

Sets the rate at which ADC samples are placed into the device managers buffers, overriding ADCRATE in /cfg/main.cfg. The rate at which samples can be taken by the burst and sample commands are directly limited by this value. Care should be taken not to increase this higher than is needed, since the processor is more heavily loaded when this value is increased. The maximum safe value will depend on how the ADC over-sampling is configured, how many devices are included and the actual sample or burst rate.

adcoversample

Parameters

rate	The oversampling rate	1 – 1000

Description

Sets the number of ADC readings which are averaged to generate an ADC reading, overriding ADCOVERSAMPLE in /cfg/main.cfg. The maximum value to set this too, depends on ADCRATE, the number of devices included, and the actual sample or burst rate.

3.0 ESM Command Line Reference

3.1 Data manipulation commands

allocate

Parameters

path	The data store path	Any path
stream	The type of stream to allocate	Synopsis or burst
size	The number of bytes to allocate	0 - 999999999
wrap	Specifies the stream should be wrapped	wrap

Description

Sets a restriction on the maximum size a stream may be. When the stream becomes as big as the maximum amount allowed. If the optional *wrap* parameter is included, then the oldest entries are removed from the stream. Otherwise storing of data is skipped. If the allocated size is 0, then the stream is allowed to grow indefinitely. Care should be used with this however, as when the card becomes full, the script will not be able to continue. E.g. *allocate /store/cf/test, synopsis, 1000000* Sets a cap of 1MB on the synopsis stream in */store/cf/test* without wrapping when the limit is reached. *allocate /store/cf/test, burst, 5000000, wrap* sets a cap of 5MB on the burst stream in */store/cf/test* old data is deleted to allow storing to continue when the 5MB limit is reached.

extract

Parameters

path	The data store path	Any path
stream	The type of stream to allocate	synopsis, burst or graph
mode	Optional mode parameter	compact, new

Description

Extracts data from a stream in a storage directory. If the *compact* switch is present, then headers are omitted. This is very useful when importing the data into Microsoft Excel. E.g. *extract /store/cf/test, burst, compact* Displays the burst stream in */store/cf/test* without displaying headers. *extract /store/cf/test, burst* Displays the burst stream in */store/cf/test* including headers. If the *graph* option is specified, synopsis data is output in a format convenient for importing into Excel. The *compact* flag is not compatible with graph output. If the *new* switch is present, then only data what has not previously been downloaded is retrieved. This is not compatible with the *compact* switch.

discard

Parameters

path	The data store path	Any path
stream	The type of stream to allocate	Synopsis or burst
amount	The number of entries to discard	0 – 999999 or all

Description

Discards the number of entries as specified by *amount*. *amount* may be *all*, in which case all data in the stream is discarded. E.g. *discard /store/cf/test, burst, 5* Gets rid of the results of the last 5 sample or burst commands output to */store/cf/test*. *discard /store/cf/test, synopsis, all* Gets rid of all synopsis data in */store/cf/test.*

streaminfo

Parameters

path	The data store path	Any path
stream	The type of stream to get info for	synopsis or burst

Description

Gets the number of entries in a stream. E.g. *streaminfo /store/cf/test, burst* gets the number of entries in the burst stream for */store/cf/test*. *streaminfo /store/cf/test, synopsis* gets the number of entries for the synopsis stream.

3.2 Download mode commands

timeleft

Parameters

none

Description

Displays the amount of time available in download before the power MUST be switched off.

stop

Parameters

none

Description

Aborts execution of the running script and returns to the command prompt.

off

Parameters

none

Description

Shuts down the ESM so that the script may resume when necessary. This should be done as soon as possible after entering download mode to prevent script timings from being affected.

3.3 File manipulation commands

ls
Alternative: dir

Parameters

directory	The directory to list	Any directory

Description

Displays a directory listing relative to the current directory. If no parameter is entered, the current directory is listed. If the directory name starts with a /, then the directory name is relative to root. Otherwise, it is relative to the current directory. The special '.' and '..' directories behave in the same was as DOS/UNIX. E.g. the directory */store/cf/subdir/../../eep* is equivalent to */store/eep* (even if pointless). This is useful if a listing of a higher level directory is required. E.g. the current directory is */store/cf* and a directory listing of the EEPROM volume is required. *../eep* can be used instead of the fully qualified name of */store/eep*.

cd

Parameters

directory	The directory to change to	Any directory

Description

Changes the current directory. See the ls command for details about how to use the parameter.

mkdir

Parameters

directory	The directory to create	Any directory

Description

Creates a subdirectory. Multiple directories can be created at the same time. E.g. mkdir /store/cf/sub1/sub2/sub3 will create sub1 and sub2 and sub3 if necessary. Directories can be relative to root or the current directory. See the ls command for details.

rm
Alternative: del

Parameters

file/dir	The file or directory to remove	Any directory or file

Description

Deletes a file or directory. Only empty directories may be deleted. Special directories such as */store* cannot be removed

deltree

Parameters

directory	The directory tree to remove	Any directory

Description

Recursively removes an entire directory tree. E.g. *deltree /store/cf/subdir* will delete all of subdir's contents, and then remove the directory subdir.

cp

Alternative:
copy

Parameters

source	The filename of the file to copy	Any file
destination	The filename of the file to copy to	Any file

Description

Copies source to destination. E.g. *cp /store/cf/blah.txt /store/eep/blah.txt*

mv

Alternative:
move

Parameters

source	The filename of the file to move	Any file
destination	The filename of the file to move to	Any file

Description

Moves source to destination. E.g. *mv /store/cf/blah.txt /store/eep/blah.txt*

format

Parameters

volume	The volume to format	Any volume

Description

Extreme caution should be used when executing this command – all data will be permanently lost, and the ESM may behave in unexpected ways. The format command should always take a fully qualified directory name (see the ls command for more details). The following directories may be formatted:
/store/cf – on an ESM Plus
/store/flash – on an ESM Micro
/ram
Do NOT attempt to format any other directory.

freespace

Parameters

path	The path to get the free space for.	Any path

Description

Gets the number of bytes of freespace at the path specified by path. E.g. *freespace /store/cf* will give the amount of freespace on the compact flash card. *freespace /store/cf/sub1* will also give the amount of space on the compact flash card.

type

Parameters

file	The filename of the file to display	Any filename

Description

Displays the contents of file on the screen. This command should only be used with ASCII files. See the df command for information about displaying binary files.

df

Parameters

file	The filename of the file to display	Any filename

Description

Performs a hex dump of file. This will display any file, ASCII and binary.

upload

Parameters

file	The filename of the file to upload to	Any filename

Description

Accepts text from serial, and inserts it straight into the file specified by file. Press Ctrl+C when you have finished uploading.

edit

Parameters

file	The filename of the file to edit	Any filename

Description

Launches a text editor. The editor should only be used on ASCII files. The Terminal program's dimensions must match the settings in the main configuration file, and escape characters must be enabled. See the section on configuring the ESM for more information. Using the Ctrl key and the highlighted letter on the menu, selects the option. E.g. Ctrl + E exits the program.

3.4 Real Time Clock commands

setdate

Parameters
none

Description
This launches a program for setting the date in the real time clock.

settime

Parameters
none

Description
This launches a program for setting the time in the real time clock.

time

Parameters
none

Description
Displays the date and time in the format *DD/MM/YYYY HH:MM:SS*

3.5 Digital IO commands

digitalio

Parameters

iobit	The i/o bit to change or hex value to set	1 – 16 or HHHH
change	How to change the i/o bit	set, clear, toggle, allset

Description

This command changes the levels of the I/O bits, and turns on the power outputs associated with the I/O bits if available. If change is set, then the I/O bit is turned on, if it is clear, then the I/O bit is turned off. If change is toggle, then the I/O bit is set to the opposite that it was. E.g. *digitalio 15, set* If change is allset, then wherever a bit is 1, the output is set. Once all outputs are set, then any values where the bit is 0, the output is cleared. E.g. *digitalio F39B, allset* sets the digital outputs to F39B (1111001110011011). The least significant bit is bit 1. The upper 4 bits control the SPS ports on the ESM Plus.

getio

Parameters

none

Description

Displays the current output levels on each I/O bit, in the form: Digital IO Pattern: 0000000000000000 [0000] Where the least significant bit of the binary number is I/O bit 1. The number in square brackets is the Hexadecimal representation of the number

Digital IO Bit Assignment

ESM Plus		
Bit	**Usage**	**Output Type**
1	J18 Pin 2	TTL
2	J18 Pin 4	TTL
3	J18 Pin 6	TTL
4	J18 Pin 8	TTL
5	J18 Pin 1	TTL
6	J18 Pin 3	TTL
7	J18 Pin 5	TTL
8	J17 Pin 1	TTL
9	J23 Pin 8	Power output
10	J23 Pin 6	Power output
11	J23 Pin 4	Power output
12	J23 Pin 2	Power output
13	J15	SPS
14	J14	SPS
15	J13	SPS
16	J12	SPS

ESM Micro		
Bit	**Usage**	**Output Type**
1	J5 Pin 1	TTL
2	J5 Pin 3	TTL
3	J5 Pin 5	TTL
4	J5 Pin 7	TTL
5	J5 Pin 9	TTL
6	J5 Pin 11	TTL
7	J5 Pin 13	TTL
8	J5 Pin 15	TTL
9	J3 Pin 8	Power output
10	J3 Pin 6	Power output
11	J3 Pin 4	Power output
12	J3 Pin 2	Power output

3.6 Serial commands

initserial

Parameters

port	The serial port to initialise	Plus: 1 – 5 Micro: 1 – 2
baud	The baud rate to initialise at	110, 150, 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200

Description

Initialises serial port specified by port, to the baud rate specified by baud. . E.g. *initserial 5, 9600* Caution should be used when initialising ports used for UI, as specified in */cfg/main.cfg*

openpipe

Parameters

port	The port to open a pipe to	Plus: 1 – 5 Micro: 1 – 2

Description

Opens up a two-way pipe between the UI port and another serial port. Any characters received on the UI port are transmitted on the other serial port. Any characters received on the other serial port are transmitted on the UI port. The abort character specified in */cfg/main.cfg* (Ctrl+C by default) terminates the pipe.

sps

Parameters

address	The address of the SPS device	Any SPS Address
data	The string to send to the SPS device	Any string

Description

Sends a string to an SPS device, and displays the response from the device. E.g. *sps "00", "VQ"*

waitsps

Parameters

address	The address of the SPS device	Any valid SPS address

Description

Waits until an SPS command has completed. E.g. *waitsps “01”* will wait until the SPS device with address 01 has completed its command.

3.7 Miscellaneous commands

cls
Alternative: clear

Parameters
none

Description
Clears the screen. Escape characters must be enabled to use this option. See the section on configuring the ESM for more information.

info

Parameters
none

Description
Displays information about the ESM firmware and available hardware.

devicelist

Parameters
none

Description
Displays a list of all available devices that may be used with the device scripting command

powersupply

Parameters
none

Description
Displays the current voltage being supplied to ESM before regulation. This is useful for checking the state of the battery powering the ESM.

uid

Parameters
none

Description
Displays the unique identification number for this ESM.

3.8 Variables and conditionals

The ESM can be supplied with an optional firmware module that adds some conditional commands to the main command set. This module can also be added to any ESM system at anytime as a serial port uploadable upgrade. The conditional commands significantly enhance the functionality of the ESM and allow so called “Smart” sampling regimes to be easily implemented. Please ask for further details.

Variable definition

```
variable pear
variable min = -3000
variable max = 12.34
variable sply = [Power Supply]
```

This example defines *pear*, *min*, *max* and *sply*. *min* is initialised to -3000 and *max* to 12.34. *pear* is initialised to 0 and *sply* is initialised with the current voltage supplied to the ESM.

It is not possible to use certain variable names, which are reserved by the scripting language. These included device names, such as *VIn* and *Ch0* and commands such as *repeat* and *transmit*.

Handling variables

Variables may take values from installed devices as instantaneous readings. The format for getting the values from installed devices is the name of the device enclosed in square brackets. The names of devices supported, can be found by using the *devicelist* command.

```
pear = [Analog 0]
sply = [Power Supply]
```

In the example, *pear* takes the value of a single reading from analogue channel 0 and *sply* takes the value of a single reading from the power supply. Variables may take the most recent value from a synopsis. In the following example, *pear* will take the synopsis average of the 100 samples from channel 0 analogue

```
device add,[Analog 0]
sample 100, "Test 1", "/store/cf/an0"
pear = [Analog 0]
```

To revert to taking instant readings, use the device remove, or device clear command.

Variables may be transmitted as part of a serial string. In the following example, a single sample from channel 0 analogue and the value in *sply* will be sent to the UI port.

```
transmitvar 0, [Analog 0]
transmitvar 0, sply
```

Special variables

A special variable is defined after an SPS command has been sent. This takes the form *SPS:XX* where *XX* is the address of the SPS device. This variable is defined as 1 if the last command sent to the SPS device was successful, and 0 if the command failed. This is useful for error checking. E.g.

```
sps "1F","ST 40000 10 100"                              # Attempt to set the temperature
if sps:1F = 0                                           # If the last command failed
   transmit 0, “Colorimeter failed to get to temperature!\r\n”  # Output a debug message
   return                                               # abort execution of the subscript
end
```

If statements

An *if* statement can be used to perform conditional execution. In the following example, the script “special” is executed if the value of *pear* is greater than 12.34, and ‘normal’ at all other times.

```
if pear > 12.34
   execute "/store/cf/scripts/special"
else
   execute "/store/cf/scripts/normal"
end
```

While statements

A *while* statement can be used to perform conditional loops. The following example takes a sample from Analog 0 and Analog 1, and compares the values with 5000 and 3000. Providing the overall condition is true, then the statements within the *while* loop are continuously executed.

```
while ( [Analog 0] < 5000 ) and ( [Analog 1] > 3000 )
   device add, [Analog 0]
   sample 100, "Test", "/store/cf/test"
   device clear
   sps "1A","RF A 100"
end
```

Note that the *device clear* statement is needed to prevent the synopsis [Analog 0] value from being used in the condition, rather than the instantaneous value.

Conditions

Conditions in an *if* and *while* statement can consist of
<   >   =   and   or

Parenthesis *must* be used to force precedence.

Parameters to commands

Certain variables may be used as parameters to certain commands. Only whole integers may be used.

```
var serport = 2                  # Initialise the variable serport with the value 2
var serbaud = 9600               # Initialise the variable serbaud with the value 9600
initserial serport, serbaud      # Initialise the serial port specified by serport to the baud rate specified by serbaud
transmit serport, “Banana”       # Transmit the string “Banana” to the specified serialport
```

The above script is particularly useful, since a script can be written once, and then if the configuration changes such that a device is attached to a different serial port, only one value in the script needs to be modified. The following commands can take variables as parameters:

Command	**Parameters for which variables can be used.**
adcoversample	1st
adcrate	1st
burst	1st and 2nd
capture	1st, 3rd and 4th
checkpoint	1st
deviceport	1st but only if the second parameter is *COM*
initserial	1st and 2nd
repeat	1st
sample	1st
senddate	1st
sendtime	1st
sleep	1st
transmit	1st
wait	1st

Variable scope and lifetime

A variable name is local to a script. A variable within one script may not be referred to in a nested (executed) script.

The values of variables persist through cycles into power save mode.

Preserving variables in a file

Values of variables may be preserved by saving them to a file using the 'savevars' command

```
savevars “vars”
```

This script statement will store all variables into a file called 'vars'.

The values of the variables may be retrieved by using the 'loadvars' command.

```
loadvars “vars”
```

This script statement will load all variables from a file called 'vars'. The 'loadvars' and 'savevars' statements may appear in different scripts.

NOTE: A variable read from a file will overwrite existing variables of the same name. Variables with different names will not be affected.

3.9 Maths operations

ESMs with the optional conditional command module also have support for basic maths operations.

The following operators are supported:

Operator	Purpose
+	Addition operator
-	Subtraction operator
*	Multiplication operator
/	Division operator
(	Open parenthesis operator
)	Close parenthesis operator

All maths operations must be assigned to a variable – it is not possible to use a maths operation as a parameter to a command, or as part of a conditional. However once the operation has been stored in a variable, the variable may be used as a parameter of a conditional. For example:

```
var orange = 0.5                    # Initialise the variable orange with the value 0.5
var apple = 20                      # Initialise the variable apple with the value 20
var result                          # Declare the variable result
result = orange * ( 50 +  apple ) – 5   # Perform the maths operation, and store the result in the variable result
transmit 0, “The result is: “       # Output something to the UI serial port
transmitvar 0, result               # Output the result of the operation to the UI serial port
transmit 0, “\r\n”                  # Send the end of line characters
```

The above script will output the string: “The result is: 30”

Device variables such as [Analog 0] may be used in maths operations, and they behave as they do in conditionals.

4.0 ESM Configuration Instructions

4.1 Administrative commands

reset

Parameters

none

Description

Simulates switching off and switching on the ESM. Useful for refreshing after changing configuration files.

cfgraw

Parameters

configfile	The file to modify (usually */cfg/main.cfg*)	Any valid filename
setting	The setting to change	Max 17 chars
contents	The new value to write	Max 40 chars

Description

Waits for a new value against a configuration setting.
E.g. *cfgraw /cfg/main.cfg, "ESCAPECHARS", "ENABLED"*

initsys

Parameters

none

Description

Rebuilds the */store/eep* tree, the */cfg* tree, and several internally used trees. All data will be lost. To be used as a last resort!

firmwareupgrade

Alternative:
fw

Parameters

none

Description

Puts the ESM into a mode where the ESM Firmware Upgrader utility may be used. Only a successful code update, power cycling or pressing the reset button (if fitted) can get out of this mode.

reloadsps

Parameters

none

Description

Reloads SPS device information from */cfg/devs.cfg*. See the section on SPS Device Configuration.

There are two user modifiable files in */cfg*. These are:

/cfg/main.cfg	-	The main configuration file

/cfg/devs.cfg - The device configuration file

There may be an additional file (*/cfg/inter.cfg*). This file should not be modified.

The configuration files are changed by using the *edit* command (see the section titled 'File manipulation command reference' for more information). E.g. type *edit /cfg/main.cfg* from the command line to edit the main configuration file.

4.2 The "main" configuration file

Most of these settings in the main.cfg file are only read after power on, so after modifying this file it is recommended that either the power is cycled, or the *reset* command is used.

The ESM will check for the presence of */store/cf/main.cfg* first. If this file is detected, then it will be used in precedence over */cfg/main.cfg*. This is useful if */cfg/main.cfg* is invalid.

DEFAULTHOME	Sets the default home directory (i.e. the current directory when the ESM is powered on). This may be any valid directory. It is not recommended to use a directory that may not be present, such as a subdirectory of the Compact Flash card, or */cfg*.
INTERMEDIATESTACK	This option specifies where the program execution stack should be stored during power off. This must be a file in a non-volatile tree such as */store/cf*. The recommended setting for an ESM Plus is: */store/cf/stack.run*. for an ESM Micro: */store/flash/stack.run*
DMPERSIST	This option is not used in this release of Firmware.
UISERIALPORT	Sets the serial port that should be used for the User Interface. This may be any value between COM1 and COM6 for the ESM Plus or between COM1 and COM3 on the ESM Micro. It is important to note that the serial ports are not mapped internally in the same way as for the rest of the interface. COM3 is the SPS hardware port, not the third port on the external serial connector. The default is COM1, which is the isolated serial port.
UIBAUD	Specifies the baud rate to use for the User Interface. This may be any value that can be used by the *initserial* command. The ESM ships with this set to 115200.
ABORTCHAR	Specifies the character that should be treated as Ctrl+C (in decimal) e.g. 3 would be Ctrl+C.
TERMWIDTH	Sets the number of columns that the terminal program has. This is very important for use by the *edit* command. Acceptable values are in the range of 80-132.
TERMHEIGHT	Sets the number of rows that the terminal program has. Acceptable values are in the range of 20-50.
DUPLEX	Specifies the duplex mode of the ESM. Half-duplex serial uses a proprietary format. Acceptable values: *FULLDUPLEX*, *HALFDUPLEX*.
ACCESSLEVEL	This option is not used in this release of Firmware.
DEFAULTDEVPORT	This option is not used in this release of Firmware.
SPSMODE	Acceptable values are: *NORMAL*, *DEBUG*. If *DEBUG* is specified, then extremely verbose diagnostic information is displayed whilst accessing SPS devices. The recommended setting is *NORMAL*.
SPSPORT	Specifies which serial port to use for SPS communication. This should always be COM3, since that is the port the SPS isolation hardware is on. This option is intended to allow future SPS style communication over standard RS232.
SPSBAUD	The baud rate to use for SPS communication. The SPS v1 specification requires this to be 9600.
SPSTIMEOUT	Sets the timeout (in milliseconds) before assuming the SPS device is not going to respond. The recommended value for this is 500.
SPSRETRIES	Specifies the number of retries before giving up completely with a single SPS communication. The recommended value is 5.
ESCAPECHARS	Specifies whether escape characters should be sent. Acceptable values are: *ENABLED*, *DISABLED*. Escape chars must be enabled to use the edit command. The recommended value is *ENABLED*.
LINETERMINATION	Specifies what the line termination should be. Acceptable values are: *CR*, *CRLF*. Escape chars will automatically be disabled when line termination is set to *CR*. They can be re-enabled by using the *crlf* command (see section titled ‘Administrative command reference’ for more details).
LSTHRESHOLD	A stream is split up into many files to allow rolling addition when the size of a stream is restricted through use of the *allocate* command. This specifies the file size threshold where a new file is created. The recommended value is 250000 on an ESM Plus or 50000 on an ESM Micro.

UNITSERIALNUMBER	This option is not used in this release of Firmware**.**
MASTERTIMEOUT	Sets the number of minutes of inactivity before the ESM will put itself into power down mode. The special value of 0 indicates no timeout. If a script is specified for *AUTOSCRIPT*, then the ESM will not put itself into power down mode.
DOWNLOADTIME	Sets the number of milliseconds of download time to be given before shutting down whilst executing a script. The special value of 0 indicates that the ESM will shutdown without entering download mode.
AUTOSCRIPT	Specifies the script that will automatically be run at start up. If this is blank, then no script will be run. E.g. */store/eep/autoscp*.
DELAYAUTOSCRIPT	Sets the number of milliseconds to spend in command mode after power up before executing the script specified by *AUTOSCRIPT*. Any key press before this time will abort execution of the script.
ASRESETS	This variable is decremented every time the ESM starts up. The script specified by AUTOSCRIPT is only run if ASRESETS is 0. The *firmwareupgrade* command increments this when it is executed.
ADCRATE	Specifies the rate at which the ADC converter provides samples to the device manager. This defines the maximum burst rate. (In Hz)
ADCOVERSAMPLE	Specifies the number of actual ADC readings that are taken to generate the value provided to the device manager. A higher value increases ADC accuracy, but reduces the maximum ADCRATE
STREAMBUFFER	Specifies the location where burst data should be stored during buffered burst mode. The default is */store/ram/stream.buf.*

devs.cfg

This file contains all of the definitions for SPS devices and future serial devices. It is a text file in Device Specification Language (DSL). Most SPS devices will be supplied with a suitable DSL file for merging into *devs.cfg* to add support for them. The ESM ships with a DSL file, which allows access to the SPS colorimeter device.

After changing devs.cfg, either cycle the power, use the *reset* command, or the *reloadsps* command. The new devices will now appear when you use the *devicelist* command.

5.0 Error Messages

5.1 Script errors

The following are the possible errors that may be reported during scripting.

ff is the script filename
nn is the line number of the error

Synopsis stream full	The allocated synopsis memory is full Allocate more space using 'allocate'
Script Error: [ff:nn] Bad token xxx	ff is the script file name nn is the line number of the error xxx is the line token that was not recognised
SPS: Unable to initialise Port	Internal error. SPS port could not be initialised.
SPS: rx timeout on attempt nn	No response from SPS device
Script Error: [ff:nn] Undefined variable: xxx	xxx is the variable that is not defined
Script Error: [ff:nn] Bad parameter: xxx	xxx is the parameter in error
Script Error: [ff:nn] Invalid time: xxx	xxx is an invalid time
Script Error: [ff:nn] Invalid date: xxx	xxx is an invalid date
Script Error: [ff:nn] Port xx is not initialised	xx is the port number is not initialised
Script Error: [ff:nn] Unable to init serial	
Script Error: [ff:nn] Failed to add device xx	xxx is the device that cannot be added to sampling
Script Error: [ff:nn] Unknown parameter	
Script Error: [ff:nn] No Devices Included	No devices have been added when sampling
Script Error: [ff:nn] invalid parameter combination	Burst parameters are wrong
Unable to write: " xxx "	Unable to open a file for writing.
Stream: " xxx " full	When recording the burst data, out of space
Script Error: [ff:nn] Invalid digitalio	Digitalio setting is wrong
Script Error: [ff:nn] Invalid Port xx	Invalid Port when setting device port
SPS command failed	
SPS device rejected command	
SPS device failed to respond	
SPS device responded with: " xx "	
Script Error: [ff:nn] : Bad ADC rate	ADC rate is invalid
Script Error: [ff:nn] : Token 'xxx' is reserved	Xxx cannot be used as a token in a script
Script Error: [ff:nn] : Bad variable name	Variable name cannot be used
Script Error: [ff:nn] : Variable name is already in use	Variable name cannot be re-used
Script Error: [ff:nn] : Failed to allocate space for the variable	Variable storage space has been filled
Script Error: [ff:nn] : Bad expression: " xxx "	Error in mathematical expression
Script Error: [ff:nn] : Bad token " xxx "	Bad token in script
Script Error: [ff:nn] : Bad conditional " xxx "	Error in mathematical expression
Script Error: [ff:nn] : Else without matching if	Improperly formed if statement.
Bad operand: "nn"	Bad operand in mathematical expression

5.2 File system errors

The following errors may be reported directly as a result of performing file system commands, or scripting commands.

Cache Error	Internal error when initialising the filing sytem
No directory	Specified directory is invalid when requesting to delete it
No source	Source filename invalid when performing file copy or move
No destination	Destination filename invalid when performing file copy or move
xxx : command not found	Invalid command specified
ACCESS DENIED	File system access error
DIRECTORY CONFLICT	File system access error
DIRECTORY FULL	DOS file system limit exceeded (should not be possible on the ESM
DISK FORMAT ERROR	Unable to format drive
DRIVE ERROR	Unable to read from drive
No such file or directory	Bad filename
INVALID SPEC	Invalid drive specification (internal error)
MODE ERROR	File system access error (internal error)
File not found	Bad filename
RENAMING ERROR	File system access error (internal error)
SEEK OUT OF RANGE	File system access error (internal error)
UNRECOVERABLE FILE	File has bad sectors – file is not readable
Dir not found	Unable to change into requested directory
No such dir	Unable to perform directory listing of requested directory
File not found	Unable to find file when 'TYPE'ing a file
Unable to make directory	Unable to create the requested directory
File not found	Unable to delete the requested file
Access Denied	Unable to delete the requested file – perhaps the file is open for writing.
Device not found	Unable to format the selected device (name not found)
Volume full	Volume has filled when copying a file.
Invalid date	Invalid date format when setting the date – use format (dd/mm/yyyy)
Invalid time	Invalid time format when setting the time – use format (hh:mm:ss)
No path	No path (or invalid path) specified when extracting data
No extraction type	Extraction type not specified when extracting data
Checking FAT... BAD	Checking the FAT (and trying to fix it) has failed – serious internal error
Checking Cluster Chains... BAD	Checking the cluster chains (and trying to fix them) has failed – serious internal error
Checking Directories... BAD	Checking directory entries (and trying to fix them) has failed – serious internal error
No path	Allocate – no path specified
No stream	Allocate – must specify a stream
No size	Allocate – no size specified
Stream not found	Allocate – Stream not found

Unable to set a limit on size	Allocate – Unable to set a limit on the specified stream
No path	Discard – No path specified
No stream type	Discard – Stream type needs to be specified
Amount to discard not specified	Discard – Must specify the amount to discard
No path	Upload – specify path
Bad param	Upload – invalid filename
RAM disk error	Unable to create a RAM disk. This is an internal error
*** WARNING *** The filing system on the current volume is corrupted. Information follows …………………	Filing system is corrupted. This is an internal error.
Critical Error: Volume to large to format	We are limited to volume size of 512MBytes
Warning: Cluster size exceeds 32K. There may be compatibility problems on some systems	The format of the CF card may not be compatible with other systems. This is extremely unlikely.
Unable to write recovery file	Unable to create the recovery file – unable to open file for writing
LS: Corrupted recovery file!	Recovery file is corrupt – unable to recover.

5.3 Serial device errors

No serial port	InitSerial – no serial port specified
No baud rate	Initserial – no baud rate specified
Invalid port number	Initserial – specified COM port number is invalid
Invalid port	OpenPipe – invalid port number
Port not initialised	OpenPipe – port not initialised
Unable to open a circular pipe	OpenPipe – trying to connect to yourself!
No address	SPS - command did not have an address specifier
No command	SPS - command did not have a command
Failed to get a response	SPS - command – SPS device did not reply within timeout period.
Bad parameter	waittsps - Wait for SPS device – Bad SPS address in command
SPS command failed	waittsps - Wait for SPS device – Command failed (SPS device responded with a failure message)
Wait aborted	waittsps - Wait for SPS device – Command was aborted (SPS device did not respond in timeout period)

5.4 I/O command errors

No bit pattern	Digital I/O command specified with no bit pattern
Bit doesn't exist!	Digital I/O command specified a bad bit reference (only 16 bits are available)
Unrecognised parameter:	Digital I/O command – unrecognised parameter

5.5 Duplex errors

No serial port	SetDuplex – no serial port specified
Duplex settings may only be configured on COM1 and COM2	SetDuplex – serial port for duplex may only be COM1 or COM2
Port not initialised	SetDuplex – port has not been initialised
No duplex setting	SetDuplex – No duplex setting specified
Failed to set duplex	SetDuplex – Set duplex command failed (when all parameters are apparently correct) – internal error

5.6 Miscellaneous error messages

Drive x not installed in cache	An internal error indicating that a requested drive is not being cached.
CF: Failed to read	Failed to read a CF sector – hardware failure
Unable to create file!	Unable to create the configuration file – serious internal error
Didn't write enough bytes!	Writing the configuration failed to write enough bytes
Failed to sample atod(s)	A to D did not respond when taking a reading
No COM port selected	No COM port selected when adding a device (when one is explicitly required)
Failed whilst starting device: xxxxx - serial port not initialised (yy)	Failed to start sampling since the specified serial port is not initialised for device xxxx
Not enough space to include the device	Too many devices added for burst sampling (128!)
File specified for AUTOSCRIPT not found	The file specified as the autoscript does not exist.
Null Spec Passed!	Edit – No filename supplied
Unknown VKEY:	Key code not recognised
FS Err(R)	Unable to read the temporary stack from E2PROM
FS Err(W)	Unable to write the temporary stack from E2PROM
EEPROM Failure, or first boot Initialising...	EEPROM was corrupt – (Happens on the first boot, and once the EEPROM has been erased).
BATTERY FAILURE	Non volatile memory has been lost and the time will need to be set
EASYPARSE	This message should NEVER appear. The comms has switched to binary mode.

5.7 Administrative error messages

No config file specified, no entry specified or no value specified	Cfgraw – One of the three parameters is not set
No config file specified, or no entry specified	CfgrawRead – config file cannot be found (This is an internal error since all config entries should be available)
Entry not found	CfgrawRead – entry in the config file cannot be found (This is an internal error since all config entries should be available)

6.0 SPS Command Summary

6.1 Motor Driver Command Set

Command	**AL**
Description	Align to zero position in a clockwise direction
Format	AL <C>
Argument	Direction to turn = C (clockwise)
Output	{xx:00}OK **or** {xx:00}ERROR

Command	**LM**
Description	Linear move
Format	LM <direction> <steps>
Argument	Direction = direction to move (I)nsert or (R)etract. Steps = number of steps to move (1 – 65535)
Output	{xx:00}OK **or** {xx:00}ERROR

Command	**RF**
Description	Rotates the motor by n steps in a specified direction. When ST = 0 RF relates to motor steps and when ST = 2 RF relates to encoder steps.
Format	RF <Direction> <Steps>
Argument	Direction: 'A' for anticlockwise or 'C' for clockwise
Output	{xx:00}OK
Example	Example: RF A 520 : Rotates the motor 520 steps anticlockwise

Command	**RP**
Description	Rotate to Port
Format	RP <port> **or** RP <direction> <port>
Argument	Number of port between 1 and 16 Direction of rotation (A = anticlockwise, C = clockwise)
Output	{xx:00}OK **or** {xx:00}ERROR

Command	**RS**
Description	Return status for previous operation
Format	RS
Argument	N/A
Output	{xx:00}SUCCESS **or** {xx:00}FAILURE

Command	**SB**
Description	Set backlash value for motor driver
Format	SB <steps>
Argument	Steps is the number of steps to add when changing motor drive direction (0 – 2000). The SPS motor driver stores the setting through power-off. Set to 0 for ST = 2 (encoder mode).
Output	{xx:00}OK **or** {xx:00}ERROR

Command	**SC**
Description	Set clockwise value for motor driver
Format	SC <mode>
Argument	Mode = motor direction (1 = one way, 0 = the other). This sets the meaning of ‘clockwise’. The SPS motor driver stores the setting through power-off.
Output	{xx:00}OK **or** {xx:00}ERROR

Command	**SD**
Description	Set Delay
Format	SD <delay>
Argument	Delay in ms between motor steps
Output	{xx:00}OK **or** {xx:00}ERROR

Command	**SE** (Ver 2.03 and later)
Description	Set encoder steps
Format	SE <steps>
Argument	Steps is number of discrete encoder positions in one revolution
Output	{xx:00}OK **or** {xx:00}ERROR

Command	**SH**
Description	Set half-step mode
Format	SH <mode>
Argument	Mode is half or full steps (1 = use half steps, 0 = use full steps)
Output	{xx:00}OK **or** {xx:00}ERROR

Command	**SO**
Description	Set offset
Format	SO <offset>
Argument	Number of steps from rotor index to logical 0 position in anti-clockwise direction.
Output	{xx:00}OK **or** {xx:00}ERROR

Command	**SP**
Description	Set Ports
Format	SP <ports>
Argument	Number of ports on rotor (must be equally spaced)
Output	{xx:00}OK **or** {xx:00}ERROR

Command	**SS**
Description	Set steps
Format	SS <steps>
Argument	Number of steps in a full rotation
Output	{xx:00}OK **or** {xx:00}ERROR

Command	**ST** (Ver 2.03 and later)
Description	Set type
Format	ST <type>
Argument	Set type of operation (0 = optical switch, 2 = encoder). The SPS board must be power cycled after the *type* has been changed
Output	{xx:00}OK **or** {xx:00}ERROR

Command	**SU** (Ver 2.03 and later)
Description	Set microseconds
Format	SU <delay>
Argument	Set step delay in microseconds (for use with encoders) where *delay* = 500 to 65536 microseconds. This command overrides SD.
Output	{xx:00}OK **or** {xx:00}ERROR

Command	**SW** (Ver 2.03 and later)
Description	Set way-up
Format	SU <mode>
Argument	*mode* is 0 or 1. This sets the vertical orientation of encoder. The code expects encoder pulses in a certain direction for clockwise and counter-clockwise movement. If the way mode is wrong the motor will move “the long way round” when rotating to ports.
Output	{xx:00}OK **or** {xx:00}ERROR

Command	**VQ**
Description	Version
Format	VQ
Argument	N/A
Output	{xx:00}SPS Motor v1.2

Command	**XX**
Description	Poll status
Format	XX
Argument	N/A
Output	{xx:00}BUSY – If currently performing an operation **or** {xx:00}INACTIVE – if not doing anything

6.2 SPS Detector Command Set

Command	**VQ**
Description	Version
Format	VQ
Argument	N/A
Output	{xx:00}SPS Colorimeter v1.02

Command	**GS**
Description	Get Sample
Format	GS <cell>
Argument	Cell being sampled (0 or 1)
Output	{xx:00}23675 65335

Command	**GT**
Description	Get temperature
Format	GT
Argument	N/A
Output	{xx:00}65335

Command	**SI**
Description	Set Intensity
Format	SI <cell> <intensity> <tolerance>
Argument	Cell to set (0 or 1) Intensity = ADC value to be set (0 – 65535) Tolerance = No. ADC counts allowed outside target (0 – 255)
Output	{xx:00}OK

Command	**ST**
Description	Set Temperature
Format	ST <temp> <timeout> <tolerance>
Argument	Temp = ADC value for temperature (0 – 65535) Timeout = No. minutes to achieve temperature (0 – 300) Tolerance = No. ADC counts allowed outside target (0 – 65535)
Output	{xx:00}OK

Command	**XX**
Description	Poll status
Format	XX
Argument	N/A
Output	{xx:00}BUSY – If currently performing an operation **or** {xx:00}INACTIVE – if not doing anything

Command	**RC**
Description	Recalibrate the intensity profile The re-calibration must be performed after gain changes to the colorimeter are made
Format	RC
Argument	N/A
Output	{xx:00}SUCCESS **or** {xx:00}FAILURE

Command	**RS**
Description	Return status for previous operation
Format	RS
Argument	N/A
Output	{xx:00}SUCCESS **or** {xx:00}FAILURE