Process Eye
SECS Host
SECS II or GEM/SECS II
Equipment

SP104012 Rev 0.01
September 1998

As part of our continuous product improvement policy, we are always pleased to receive your comments and suggestions about how we should develop our product range. We believe that the manual is an important part of the product and would welcome your feedback particularly relating to any omissions or inaccuracies you may discover.

You can send your comments to:-

Leda-Mass Ltd.
Cowley Way
Crewe
Cheshire
CW1 6AG
U.K.

+44 1270 250150 Tel. International
+44 1270 251939 Fax. International

01270 250150 Tel. U.K.
01270 251939 Fax. U.K.

In North America you can send your comments to:-

Spectra Instruments
380 Woodview Ave.
Morgan Hill
CA 95037

(408) 778 6060 Office
(408) 776 8575 Fax

Contents

Errata and addenda ..... a

Section 1. Introduction 1

1.1. Overview 1
1.1.1. Pre-Programmed Actions 1
1.1.2. Grid display 1
1.1.3. ProcSECS Macros 2
1.1.4. Process Eye Macros 2

Section 2. Installation 3

2.1. Configuration 3
2.1.1. ProcSECS.Cfg – SECS I (RS232) 3
2.1.2. The ProcSECS.Ini file 6
2.2. Configure Quick Start 8
2.3. Head .Ini File settings 8

Section 3. Operation 11

3.1. Pre-Programmed Operation 11
3.2. Program Start 12
3.2.1. Action Type = 1 – Run Recipe: Fixed Name 12
3.2.2. Action Type = 6 – Stop Recipe 12
3.2.3. Action Type = 7 – Run External Program 13
3.2.4. Action Type = 12 – Write to SECSlog.Txt 13
3.3. Program Stop 14
3.3.1. Action Type = 12 – Write to SECSlog.Txt 14
3.3.2. Action Type = 13 – Stop RGA 14
3.3.3. Action Type = 14 – Restart Windows 14
3.4. SECS Link Fail 15

3.5. S1F4 Poll 16
3.6. Collection Events 18
3.7. GEM Alarm Events 23
3.8. Grid Display 26
3.9. Status Variable Polling 27

Section 4. Macros 29

4.1. Overview 29
4.2. Read Only Variables 30
4.2.1. EventID 30
4.3. Read Only Arrays 31
4.3.1. S1F4Data(Index) 31
4.3.2. AlarmData(Index) 31
4.3.3. CEReportData(Index,ReportID) 31
4.4. Subroutines 31
4.4.1. MessageToTool MessageString, MacroHandlesReply 31
4.4.2. MessageToRGA MessageString, RGA 32
4.4.3. WriteToGrid Row, Col, Data, FormatString 33
4.5. Functions 33
4.5.1. IsRGARegistered (RGA) 33
4.6. Process Eye Programming 33
4.7. Macro Language Extensions 33
4.7.1. Read Only Variables 34
4.7.2. Functions 34
4.8. Miscellaneous Notes 37

Appendix 1 Date type names 39

Errata and addenda

This page is deliberately left blank.

This page is deliberately left blank.

Section 1.
Introduction

1.1. Overview

There are two versions of ProcSECS. One is designed for use with tools having a generic SECS II or GEM/SECS II interface and is known as ProcSECS (G). The other is designed for use with Endura and Centura Process tools from Applied Materials and is known as ProcSECS (A). This manual covers ProcSECS (G).

The ProcSECS application gives Process Eye the ability to communicate with a piece of equipment (typically a process tool) that has a SECS II or GEM/SECS II interface. At its simplest it may just be used to ensure the correct RGA recipe is run at the appropriate time. A more ambitious link could run appropriate recipes, log tool data as part of the RGA data and perform statistical monitoring of the data and raise alarms on the Process Tool. The computer running Process Eye, i.e. the RGA or RGA's and ProcSECS is the Host, the tool with a SECS II or GEM/SECS II interface is the Equipment.

SECS II or GEM/SECS II data can be transmitted using either SECS I (RS232) or HSMS-SS (TCP/IP) protocols. ProcSECS can use either but the correct protocol must be defined at time of order.

There are four methods that can be used to utilize RGA(s) to Process Tool communications: Pre-Programmed, Grid Display, ProcSECS Macros and Process Eye Macros.

1.1.1. Pre-Programmed Actions

Certain messages sent over the SECS II or GEM/SECS II interface can be configured to cause one or more actions at the RGA computer.

1.1.2. Grid display

Status variables may be polled from the tool at a user defined interval and

displayed in a spreadsheet like grid. Row and column headings may be defined as well as the placement of the data.

1.1.3. ProcSECS Macros

Messages received by ProcSECS are pre-processed and then made available to the VBA compatible macro language. Additionally any or all messages may be configured to be intercepted by the Macro. Within the macro arbitrarily complex actions may be taken.

1.1.4. Process Eye Macros

Using the Macro facility of Process Eye virtually anything that can be done over the SECS II or GEM/SECS II interface may be integrated with RGA operation. Using the Macro method requires a detailed understanding of the SECS II or GEM/SECS II interface, VBA programming and Process Eye's implementation of the link. (Note that in Multi-head situations it may not be possible to direct the appropriate message to the appropriate head and ProcSECS macros should be used if the pre-programmed capabilities are not sufficient.)

Section 2.
Installation

Close any programs that are running, insert the ProcSECS install disk in floppy drive A: (or B:) and either click Start|Settings|Control Panel|Add/Remove Programs or double click the Setup.Exe file from explorer. Follow the on-screen instructions. It is suggested that you leave all settings at their default values. The only item that should be changed is the directory if the default is not the same as the directory in which Process Eye is installed as it is strongly recommended that Process Eye and ProcSECS are installed in the same directory.

2.1. Configuration

For the SECS II link to function the ProcSECS software must be correctly configured. There are two files that must be correctly set: ProcSECS.Ini and ProcSECS.Cfg. Both ProcSECS.Cfg and ProcSECS.Ini are text files and **must** be edited with a text editor such as NotePad. In addition Win.Ini must be modified to show the location of the ProcSECS software but this is handled automatically by the install program.

2.1.1. ProcSECS.Cfg – SECS I (RS232)

Below is a typical configuration for a SECS I (RS232) link.

```
PORT 0:
  LINE COM1
  PROTOCOL SECS
  BAUD 9600
  T1 0.5
  T2 3.0
  T3 30.0
  T4 10.0
  RETRY 3
```

```
  HOST

DEVICE 0:
  PORT 0
  DEVICEID 0x0000
```

Port 0: parameters
LINE – Defines the RS232 port the SECS link is using. Must be COM1 or COM2

PROTOCOL – Must be SECS

BAUD – defines the Baud rate. May be 110, 150, 300, 600, 1200, 2400, 4800, 9600 or 19200

T1 – The SECS T1 (Inter Character) timeout. Must be between 0.1 and 25.5 seconds

T2 – The SECS T2 (Block Protocol) timeout. Must be between 0.1 and 25.5 seconds

T3 – The SECS T3 (Reply) timeout. Must be between 1.0 and 120 seconds

T4 – The SECS T4 (Inter Block) timeout. Must be between 1.0 and 120 seconds

RETRY – The SECS Retry Limit (RTY). Must be between 0 and 31

HOST – Specifies that ProcSECS is going to act as the SECS host. Generally should be left alone but if the other end cannot be specified as Equipment it may be set to EQUIPMENT.

DEVICE 0: parameters

PORT – must match the Port number defined above. Should be left at 0

DEVICEID – The SECS Device ID. A number between 0 and 32767

ProcSECS.Cfg – HSMS-SS (TCP/IP)

Below is a typical configuration for an HSMS-SS (TCP/IP) link.

```
PORT 0:
   PROTOCOL HSMS94
   HOST
   ACTIVE ENTITY
   PASSIVE ENTITY IPADDRESS 234.123.234.123
   PASSIVE ENTITY TCPPORT 5000
   T3 30
   T5 5
   T6 20
   T7 5
   T8 6
   CONNECTION ESTABLISHMENT 5
   CIRCUIT ASSURANCE 0
   TGRACE 15
   MEMORY STALL 5
   WRITE STALL 15

DEVICE 0:
   PORT 0
   DEVICEID 0
```

Port 0: parameters

HOST – Specifies that ProcSECS is going to act as the SECS host. Generally should be left alone but if the other end cannot be specified as Equipment it may be set to EQUIPMENT.

PROTOCOL – Must be HSMS94

ACTIVE ENTITY – Specifies ProcSECS should establish the TCP/IP connection. On an HSMS link one end must be active, the other passive. If ProcSECS is to be passive this entry should be: PASSIVE ENTRY

PASSIVE ENTRY IPADDRESS – The IP address of the passive entry in standard dotted notation.

PASSIVE ENTITY TCPPORT – THE TCP port Number at which the

PASSIVE ENTITY waits for a connection on this HSMS link.

T3 – The HSMS-SS T3 (Reply) timeout. Must be between 1.0 and 120 seconds

T5 – The HSMS-SS T5 Connect Separation Timer. Typically 10

T6 – The HSMS-SS T6 Control Message Reply Timeout. Should be set to a value less than the T3 value, typically 20

T7 – The HSMS-SS T7 Connect Timeout. Typically 10

T8 – The HSMS-SS T8 Inter-Character Timeout. Typically 10

CONNECTION ESTABLISHMENT – Typically 20

CIRCUIT ASSURANCE – How often an HSMS Linktest is performed, typically 0 (no tests) or 15

TGRACE – Do not change

MEMORY STALL – Do not change

WRITE STALL – Do not change

DEVICE 0: parameters

PORT – must match the Port number defined above. Should be left at 0

DEVICEID – The SECS Device ID. A number between 0 and 32767

2.1.2. The ProcSECS.Ini file

In the [SYSTEM] section there are a number of possible configuration settings. In general they should be left at the default (i.e. nothing in the file) settings. If editing the ProcSECS.Ini file it is essential that a text editor that saves text in plain format (such as Notepad) be used.

App Is Active

When set to 0 will cause ProcSECS to wait for the other end to connect using either S1F13 or S1F1. When set to 1 will cause the ProcSECS application to send an S1F13 or S1F1 connect message. By default it will be 0,

S1F13 Connect
When ProcSECS is the active entity it can either use an S1F13 or S1F1 message to initiate communications. By default S1F1 will be used to connect. (S1F13 Connect =0). Set to 1 to use S1F13 to connect

Log Errors
All error messages can be written to a file called ProcSECS.Err. By default all errors are logged. (Log Errors = 1). Set to 0 to disable error logging.

The following configurations only apply if the Pre-Programmed method of handling SECS messages is to be replaced or augmented using the ProcSECS Macro, Process Eye Macros or another stand alone application.

ProcSECS Default Handler
When set to 1 the ProcSECS main application handles all messages that are not specified (see below). When 0 the Alternative Message Handler handles all messages by default.

Alternate Message Handler
Specifies the place messages not handled by ProcSECS are sent to in terms of the Serial number for Microvision Plus and HPQ 2 RGAs, Head number for Microvision RGAs and an arbitrary string for Stand Alone applications (See registering stand alone applications) or the ProcSECS macro code (set Alternative Message Handler = Macro)

Alternate Handler Count
Defines how many messages are handled by the Alternative Message Handler. Default is zero.

Alt Stream N
Alt Function N
For each message to handled by the Alternate Message Handler the message Stream and function are define. N = 1, 2, 3 …. Alternate Handler Count

Auto Drop Ticket

ProcSECS uses the SDR toolkit licensed from GW Associates to perform SECS I or HSMS-SS communications. The SDR uses the Ticket method to deal with multiple messages. When set to 1 ProcSECS drops all tickets except Primary messages with the W bit set (it is the Process Eye Macro or Stand Alone applications responsibility to deal with these tickets). Should be left at 1 unless you really know why you want to change it.

2.2. Configure Quick Start

RS232
Set LINE to COM1 or COM2
Set BAUD (typically 9600)
Set DEVICEID (typically 0)

TCP/IP
Set PASSIVE ENTRY IPADDRESS
Set PASSIVE ENTITY TCPPORT (typically 5000)
Set DEVICEID (typically 0)

2.3. Head .Ini File settings

To allow communications between ProcSECS and Process Eye the Enabled item in the [SECS] section of the SerialNumber.Ini file (for Process Eye V1.6x) or the ProcX.Ini (for Process Eye V1.3x and V1.4x) must be set as follows:

[SECS]
Enabled=1

Additionally the following items may be set (in the same file and section) to facilitate automatic start and stop of Process Eye for that head.

Skip the exit confirmation dialog box:
No Exit Confirmation = 1

Force the RGA's filament to be switched off if it is on:
Turn Fil Off On Exit = 1

If the RGA head stops communicating restart Windows (V1.6x only):
Restart Windows On Head Down = 1

If the RGA fails to communicate at startup retry (V1.6x only):
Retry RGA Startup = 1

This page is deliberately left blank.

Section 3.
Operation

3.1. Pre-Programmed Operation

In Pre-Programmed operation you define what action should be taken when a specific even occurs. Not all Actions are applicable to all events (see the table below for a summary)

Event	Program Start	Program Stop	SECS Link Fail	S1F4 Poll	Collection	GEM Alarm
1 – Run Recipe: Fixed Name	X	-	X	-	X	X
2 – Run Recipe: Event Defined	-	-	-	-	X	X
3 – Run Recipe: Event Defined with Lookup	-	-	-	-	X	X
4 – Reserved	-	-	-	-	-	-
5 – Set User Disk Header	-	-	-	-	X	-
6 – Stop Recipe	X	-	X	-	X	X
7 – Run External Program	X	-	X	-	X	X
8 – Add to RGA Recipe Comment	-	-	-	-	X	X
9 – Set Global Variable	-	-	-	X	X	X
10 – Switch On Filament	-	-	-	-	X	X
11 – Switch Off Filament	-	-	X	-	X	X
12 – Write Comment to disk	X	X	X	X	X	X
13 – Stop Process Eye program	-	X	X	-	-	-
14 – Restart Windows	-	X	X	-	-	-

"X" = available for that event
"-" = unavailable for that event

Note that the same action, e.g. Run an External Program, make be executed many times for a single event with different parameters for each instance of the action as well as multiple different actions.

All Actions are defined in the ProcSECS.Ini file. The section is determined by the Event that has the actions associated with it. Note that items are shown with quotation marks around the item name but the items should be entered in the ProcSECS.Ini file without the quotation marks.

3.2. Program Start

When the ProcSECS program runs there are four possible actions. The main use is to run the RGA(s) so that starting a single program runs both the ProcSECS application and the Process Eye application(s). If one or more RGAs are already running, they may have recipes started or stopped. Finally the log file SECSlog.Txt can have a line written to it.

All items go in the [STARTUP] section

The number of actions (default = 0) is defined in the "Action Count" item. For each Action an Action Type is defined in the "Action x" where x is 1,2,…Action Count. Depending on the Action Type, type specific parameters may be defined. See below for the parameters associated with the different Action Types.

3.2.1. Action Type = 1 – Run Recipe: Fixed Name

The RGA on which to run the Recipe is defined in the "RGA x =" as its serial number (V1.6x) or head number (V1.4x).
The Recipe name to run is defined in the "Fixed Recipe x" item. The Recipe name must match the text on the recipe button within Process Eye.
If DDE Filename has been selected as the file naming method in the recipe the DDE filename is defined using the "DDE Filename n" item. Note that Process Eye is a 16 bit application and the DDE filename must be in DOS 8.3 format. Also using an existing filename will overwrite the previous file with no warnings.

3.2.2. Action Type = 6 – Stop Recipe

The RGA on which to stop the currently running Recipe is defined in the "RGA x =" as its serial number (V1.6x) or head number (V1.4x).

3.2.3. Action Type = 7 – Run External Program

Starts an external program. The name (including path if required) is defined in the "Program Path n" item. Note that Process Eye is a 16 bit application and any path must be in DOS 8.3 format. Any arguments may be added to the "Program Path n" item with a space to separate the path/filename and arguments. Optionally the window style that the program is to start with can be defined in the "WindowStyle n" item. A WindowStyle of 1 = Normal (default if no WindowStyle is defined), 2 = Minimized and 3 = Maximized.

3.2.4. Action Type = 12 – Write to SECSlog.Txt

Appends "Date Time Process SECS host started" to the SECSlog.Txt file

Example 1 (Process Eye V1.6x):
To run heads with serial numbers LM70-00197005 and LM70-00197006 when ProcSECS starts:

[STARTUP]
Action Count = 2
Action 1 = 7
Program Path 1 = C:\Progra~1\Spectra\Proces~1\Process.exe /LM70-00197005
Action 2 = 7
Program Path 2 = C:\Progra~1\Spectra\Proces~1\Process.exe /LM70-00197006

Example 2 (Process Eye 1.4x):
To run heads 1 and 2 when ProcSECS starts:

[STARTUP]
Action Count = 2
Action 1 = 7
Program Path 1 = C:\Proceye\Process.exe /S1
Action 2 = 7
Program Path 2 = C:\ Proceye\Process.exe /S2

3.3. Program Stop

When the ProcSECS program runs there are only two possible actions. The main use is to stop the RGA(s) so that when ProcSECS is restarted the heads will be restarted and the link between ProcSECS and Process Eye(s) established. The other action is to write to the SECSlog.Txt file.

All items go in the [STOP] section

The number of actions (default = 0) is defined in the "Action Count" item. For each Action an Action Type is defined in the "Action x" where x is 1,2,…Action Count. Depending on the Action Type, type specific parameters may be defined. See below for the parameters associated with the different Action Types.

3.3.1. Action Type = 12 – Write to SECSlog.Txt

Appends "Date Time Process SECS host stopped" to the SECSlog.Txt file

3.3.2. Action Type = 13 – Stop RGA

The RGA to stop is defined in the "RGA x =" as its serial number (V1.6x) or head number (V1.4x).

3.3.3. Action Type = 14 – Restart Windows

Windows will be terminated and then restarted.

Example 1 (Process Eye V1.6x):
To stop heads with serial numbers LM70-00197005 and LM70-00197006 when ProcSECS stops:

[STOP]
Action Count = 2
Action 1 = 13
RGA 1 = LM70-00197005
Action 2 = 13
RGA 2 = LM70-00197006

Example 2 (Process Eye V1.4x):
To stop heads 1 and 2 and write to the SECSlog.Txt file when ProcSECS stops:

[STOP]
Action Count = 3
Action 1 = 13
RGA 1 = 1
Action 2 = 13
RGA 2 = 2
Action 3 = 12

3.4. SECS Link Fail

When the ProcSECS program runs there are seven possible actions. The main use is to restart windows if the and cause the computer to re-initialize and re-make the SECS link. If one or more RGAs are already running, they may have recipes started or stopped, the filament(s) turned off. An external program may be run and the log file SECSlog.Txt can have a line written to it. Finally Windows may be terminated and restarted.

All items go in the [FAIL] section

The number of actions (default = 0) is defined in the "Action Count" item. For each Action an Action Type is defined in the "Action x" where x is 1,2,…Action Count. Depending on the Action Type, type specific parameters may be defined. See below for the parameters associated with the different Action Types.

Action Type = 1 – Run Recipe: Fixed Name
The RGA on which to run the Recipe is defined in the "RGA x =" as its serial number (V1.6x) or head number (V1.4x).
The Recipe name to run is defined in the "Fixed Recipe x" item. The Recipe name must match the text on the recipe button within Process Eye.
If DDE Filename has been selected as the file naming method in the recipe the DDE filename is defined using the "DDE Filename n" item. Note that Process Eye is a 16 bit application and the DDE filename must be in DOS 8.3 format. Also using an existing filename will overwrite the previous file with no warnings.

Action Type = 6 – Stop Recipe
The RGA on which to stop the currently running Recipe is defined in the "RGA x =" as its serial number (V1.6x) or head number (V1.4x).

Action Type = 7 – Run External Program
Starts an external program. The name (including path if required) is defined in the "Program Path n" item. Note that Process Eye is a 16 bit application and any path must be in DOS 8.3 format. Any arguments may be added to the "Program Path n" item with a space to separate the path/filename and arguments. Optionally the window style that the program is to start with can be defined in the "WindowStyle n" item. A WindowStyle of 1 = Normal (default if no WindowStyle is defined), 2 = Minimized and 3 = Maximized.

Action Type = 11 – Switch Off Filament
The RGA to have its filament switched off is defined in the "RGA x =" as its serial number (V1.6x) or head number (V1.4x).

Action Type = 12 – Write to SECSlog.Txt
Appends {Date} {Time} "SECS link failed" to the SECSlog.Txt file

Action Type = 13 – Stop RGA
The RGA to stop is defined in the "RGA x =" as its serial number (V1.6x) or head number (V1.4x).

Action Type = 14 – Restart Windows
Windows will be terminated and restarted.

Example 1 To restart Window when the SECS link to the tool fails:

[FAIL]
Action Count = 1
Action 1 = 14

3.5. S1F4 Poll

ProcSECS can be configured to poll SVID data from the tool periodically. This data can then be displayed directly in the Grid Display, processed and displayed in the Grid Display or sent to an RGA (typically for inclusion with RGA data). See the Grid Display section below for details of how the SVID

polling is configured.

When data from an SVID poll is received by ProcSECS only a single action is configurable – send the data to a global variable within an RGA head.

All items go in the [S1F4] section

The number of actions (default = 0) is defined in the "Action Count" item. For each Action an Action Type is defined in the "Action x" where x is 1,2,…Action Count.

Action Type = 9 – Set Global Variable
The RGA on which to set a Global Variable is defined in the "RGA x =" as its serial number (V1.6x) or head number (V1.4x).
The Global Variable to set is defined in the "Global Variable Pointer x" item. Typically global variables are numbered 1 through 8 but this may be extended.
The data to store in the global variable is defined in the "Data Pointer x" item. This points to which Status Variable is to be stored in the Global Variable.

Example 1 (Process Eye V1.6x):
To store the first polled status variable in Global Variable #1 on both heads, the second polled status variable in Global Variable #2 on the first head (LM70-00197005) and the third polled status variable in Global Variable #2 on the second head (LM70-00197006)

[S1F4]
Action Count = 4
Action 1 = 9
Global Variable Pointer 1 = 1
Data Pointer 1 = 1
RGA 1 = LM70-00197005
Action 2 = 9
Global Variable Pointer 2 = 1
Data Pointer 2 = 1
RGA 2 = LM70-00197006
Action 3 = 9
Global Variable Pointer 3 = 2
Data Pointer 3 = 2

RGA 3 = LM70-00197005
Action 4 = 9
Global Variable Pointer 4 = 2
Data Pointer 4 = 3
RGA 4 = LM70-00197006

3.6. Collection Events

GEM/SECS II tools have many collection events defined that correspond to significant events at the tool. Collection Events typically have Reports associated with them. ProcSECS allows reports to be defined, reports to be associated with Collection Events, Collection Events to be enabled and Actions to be performed when a Collection Event occurs.

To create one or more Reports create a [REPORTS] section and define the total number of reports with the "Report Count" item.
Define the data type used to refer to reports in the "Report Type" item. See Appendix 1 for data types.
Define the data type used to refer to VIDs in the "VID Type" item. See Appendix 1 for data types.
For each report create a [REPORT X] section where X is the Report Number 1,2, … Report Count.
For each Report define the number of VIDs in the Report with the "Count" item.
For each VID define the VID number with a "VID x" item and the VID data type with the "Type x" item. See Appendix 1 for data types.

For each report define the number of VID's in the report. If there are two VID's in report 1 and 1 VID in report two then:

Example: To create two reports, Report ID #1 has two VID's, VID 3 an ASCII representation of the clock and VID 106 a 4 byte floating point number. Report ID #2 has 1 VID, VID 104 an ASCII representation of the tool Recipe name. This tool uses 4 byte unsigned integers to refer to VIDs and Report IDs

[REPORTS]
Report Type = S2_U4
VID Type = S2_U4

Report Count = 2

[REPORT 1]
Count = 2
VID 1 = 3
Type 1 = S2_A
VID 2 = 106
Type 2 = S2_F4

[REPORT 2]
Count = 1
VID 1 = 104
Type 1 = S2_A

Create a [CEIDS] section and define the total number of Collection Events to monitor with the "CEID Count" item.
Define the data type used to refer to Collection Event ID's in the "CEID Type" item. See Appendix 1 for data types.
For each collection event create a [CEID X] section where X is 1,2, … CEID Count.

For each [CEID X] define the collection event ID with the "CEID" item, the Report ID (in the range 1,2… Report Count) with the "Report ID" item. The number of actions (default = 0) is defined in the "Action Count" item. For each Action an Action Type is defined in the "Action x" where x is 1,2,…Action Count. Depending on the Action Type, type specific parameters may be defined. See below for the parameters associated with the different Action Types.

If DDE Filename is being used to define that name of the data file for a Process Eye recipe it can have embedded in it any of the following data items by inserting the text <n> in the DDE Filename. The ,n> will be substituted for the nth Status Variable in the report associated with the Collection Event.

It is important that the DDE filename keep to DOS 8.3 filename conventions. Note that the "\" character is permitted and this allows the path for data storage to be dynamically defined at run time.

Action Type = 1 – Run Recipe: Fixed Name

The RGA on which to run the Recipe is defined in the "RGA x =" as its serial number (V1.6x) or head number (V1.4x).
The Recipe name to run is defined in the "Fixed Recipe x" item. The Recipe name must match the text on the recipe button within Process Eye.
If DDE Filename has been selected as the file naming method in the recipe the DDE filename is defined using the "DDE Filename x" item. Note that Process Eye is a 16 bit application and the DDE filename must be in DOS 8.3 format. Also using an existing filename will overwrite the previous file with no warnings.

Action Type = 2 – Run Recipe: Event Defined
The RGA on which to run the Recipe is defined in the "RGA x =" as its serial number (V1.6x) or head number (V1.4x).
The Recipe name to run is defined by the contents of the first VID of the report. The Recipe name must match the text on the recipe button within Process Eye.
If DDE Filename has been selected as the file naming method in the recipe the DDE filename is defined using the "DDE Filename x" item. Note that Process Eye is a 16 bit application and the DDE filename must be in DOS 8.3 format. Also using an existing filename will overwrite the previous file with no warnings.

Action Type = 3 – Run Recipe: Event Defined with Lookup
The RGA on which to run the Recipe is defined in the "RGA x =" as its serial number (V1.6x) or head number (V1.4x).
The Recipe name to run is defined by taking the contents of the first VID of the report and using that to lookup a corresponding RGA recipe name. The lookup is defined in the [RECIPE LOOKUP] section. Each Tool Recipe name becomes an item. The looked up recipe name must match the text on the recipe button within Process Eye.
If DDE Filename has been selected as the file naming method in the recipe the DDE filename is defined using the "DDE Filename x" item. Note that Process Eye is a 16 bit application and the DDE filename must be in DOS 8.3 format. Also using an existing filename will overwrite the previous file with no warnings.

Action Type = 5 – Set User Disk Header
Uses the data in the last report linked to the collection event as the values for

Process Eye's User Disk Header. It is important that VIDs in the report match exactly the user disk header definition. Setting the user disk header will cause the next recipe that has User Disk Header enabled to use the information sent with this command without the usual screen dialog. This is true even if the recipe is run locally.

Action Type = 6 – Stop Recipe
The RGA on which to stop the Recipe is defined in the "RGA x =" as its serial number (V1.6x) or head number (V1.4x).

Action Type = 7 – Run External Program
Starts an external program. The name (including path if required) is defined in the "Program Path x" item. Note that Process Eye is a 16 bit application and any path must be in DOS 8.3 format. Any arguments may be added to the "Program Path n" item with a space to separate the path/filename and arguments. Optionally the window style that the program is to start with can be defined in the "WindowStyle x" item. A WindowStyle of 1 = Normal (default if no WindowStyle is defined), 2 = Minimized and 3 = Maximized.

Action Type = 8 – Add to Comment
The RGA on which to add the comment is defined in the "RGA x =" as its serial number (V1.6x) or head number (V1.4x).
The comment to add is defined by the Collection Event Date/Time and the text in the "Comment x" item. The selected RGA must be running a Bar Chart or Peak Jump recipe.

Action Type = 9 – Set Global Variable
The RGA on which to set a Global Variable is defined in the "RGA x =" as its serial number (V1.6x) or head number (V1.4x).
The Global Variable to set is defined in the "Global Variable Pointer x =" item. Typically global variables are numbered 1 through 8 but this may be extended.
The data to store in the global variable is defined in the "Data Pointer x =" item. This points to the n'th Status Variable in the Report.

Action Type = 10 – Switch On Filament

The RGA to have its filament switched on is defined in the “RGA x =” as its serial number (V1.6x) or head number (V1.4x).

Action Type = 11 – Switch Off Filament
The RGA to have its filament switched off is defined in the “RGA x =” as its serial number (V1.6x) or head number (V1.4x).

Action Type = 12 – Write to SECSlog.Txt
Appends “{Date} {Time} Collection Event {CEID}” to the SECSlog.Txt file

Example:
To define two Collection Events to be monitored, CEID 103 (Sequence Start) and CEID 108 (Step Finished), CE103 has Report ID #1 linked with it and performs the action of starting a recipe on the RGA with the same name as the recipe running on the tool. CEID 108 has Report ID #2 linked with it and performs the action of writing the comment “Step Finished” into the comment file associated with the RGA recipe.

[CEIDS]
CEID Count=1
CEID Type = S2_U4

[CEID 1]
CEID = 103
Report ID 1 = 1
Action Count = 1
Action 1 = 2
RGA 1 = LM70-00197006

[CEID 2]
CEID = 108
Report ID 1 = 2
Action Count = 1
Action 1 = 8
RGA 1 = LM70-00197006
Comment 1 =Step Finished

3.7. GEM Alarm Events

GEM/SECS II tools usually have alarm events defined that correspond to significant problems at the tool. Alarm Events have three data items associated with them:

ALCD – a 1 byte alarm code as defined in the SEMI SECS II standard
ALID – a unique number identifying the alarm
ALTX – an option text string of up to 40 characters detailing the alarm event

ProcSECS allows Alarms to be enabled and Actions to be performed when an Alarm Event occurs.

Create an [ALARMS] section and define the total number of Alarm Events to monitor with the "Alarm Count" item.
For each alarm event create an "Alarm XXX" item where XXX is the Alarm ID of an alarm to monitor.

For each alarm create an [ALARM XXX] section where XXX is the Alarm ID of an alarm to monitor.

In each [ALARM XXX] section define the number of actions (default = 0) is defined in the "Action Count" item. For each Action an Action Type is defined in the "Action x" item where x is 1,2,…Action Count. Depending on the Action Type, type specific parameters may be defined. See below for the parameters associated with the different Action Types.

Action Type = 1 – Run Recipe: Fixed Name
The RGA on which to run the Recipe is defined in the "RGA x =" as its serial number (V1.6x) or head number (V1.4x).
The Recipe name to run is defined in the "Fixed Recipe x" item. The Recipe name must match the text on the recipe button within Process Eye.
If DDE Filename has been selected as the file naming method in the recipe the DDE filename is defined using the "DDE Filename x" item. Note that Process Eye is a 16 bit application and the DDE filename must be in DOS 8.3 format. Also using an existing filename will overwrite the previous file with no warnings.

Action Type = 2 – Run Recipe: Event Defined
The RGA on which to run the Recipe is defined in the "RGA x =" as its serial number (V1.6x) or head number (V1.4x).
The Recipe name to run is defined by the contents of ALTX. The Recipe name must match the text on the recipe button within Process Eye.
If DDE Filename has been selected as the file naming method in the recipe the DDE filename is defined using the "DDE Filename x" item. Note that Process Eye is a 16 bit application and the DDE filename must be in DOS 8.3 format. Also using an existing filename will overwrite the previous file with no warnings.

Action Type = 3 – Run Recipe: Event Defined with Lookup
The RGA on which to run the Recipe is defined in the "RGA x" item x =" as its serial number (V1.6x) or head number (V1.4x).
The Recipe name to run is defined by taking the contents of ALTX and using that to lookup a corresponding RGA recipe name. The lookup is defined in the [RECIPE LOOKUP] section. Each Tool Recipe name becomes an item. The looked up recipe name must match the text on the recipe button within Process Eye.
If DDE Filename has been selected as the file naming method in the recipe the DDE filename is defined using the "DDE Filename x" item. Note that Process Eye is a 16 bit application and the DDE filename must be in DOS 8.3 format. Also using an existing filename will overwrite the previous file with no warnings.

Action Type = 6 – Stop Recipe
The RGA on which to stop the Recipe is defined in the "RGA x =" as its serial number (V1.6x) or head number (V1.4x).

Action Type = 7 – Run External Program
Starts an external program. The name (including path if required) is defined in the "Program Path n" item. Note that Process Eye is a 16 bit application and any path must be in DOS 8.3 format. Any arguments may be added to the "Program Path x" item with a space to separate the path/filename and arguments. Optionally the window style that the program is to start with can be defined in the "WindowStyle x" item. A WindowStyle of 1 = Normal (default if no WindowStyle is defined), 2 = Minimized and 3 = Maximized.

Action Type = 8 – Add to Comment
The RGA on which to add the comment is defined in the "RGA x =" as its serial number (V1.6x) or head number (V1.4x).
The comment to add is defined in the form: "Date Time Alarm {ALID} Code {ALCD} was set|cleared. {ALTX}". The selected RGA must be running a Bar Chart or Peak Jump recipe.

Action Type = 9 – Set Global Variable
The RGA on which to set a Global Variable is defined in the "RGA x =" as its serial number (V1.6x) or head number (V1.4x).
The Global Variable to set is defined in the "Global Variable Pointer x =" item. Typically global variables are numbered 1 through 8 but this may be extended.
The data to store in the global variable is defined in the "Data Pointer x =" item. 1 sets it to ALID, 2 sets it to ALCD and 3 sets it to ALTX.

Action Type = 10 – Switch On Filament
The RGA to have its filament switched on is defined in the "RGA x =" as its serial number (V1.6x) or head number (V1.4x).

Action Type = 11 – Switch Off Filament
The RGA to have its filament switched off is defined in the "RGA x =" as its serial number (V1.6x) or head number (V1.4x).

Action Type = 12 – Write to SECSlog.Txt
Appends "{Date} {Time} Alarm {ALID} Code: {ALCD} was set|cleared {ALTX}" to the SECSlog.Txt file

Example:
Enable Alarm IDs 6 and 1, configure alarm 6 to write a comment to the currently running recipe and Alarm ID 12 to run an external program.

[ALARMS]
Alarm Type = S2_U4
Alarm Count = 2

Alarm 1 = 6
Alarm 2 = 12

[ALARM 6]
Action Count = 1
Action 1 = 6
RGA 1 = LM75-40597009

[ALARM 12]
Action Count = 1
Action 1 = 7
Program Path 1 = Notepad.exe

3.8. Grid Display

Optionally, a spreadsheet like grid may be displayed in the ProcSECS window. The number of rows, columns and titles are user definable. All go in the [GRID] section.

Enable the grid display:
Grid Enabled = 1

Define the number of rows (excluding the title row)
Grid Rows = 2

Define the number of columns (excluding the title column)
Grid Cols = 3

Text above column 1
Text Col 1 = Title for Column 1

Text above column 2
Text Col 2 = Title for Column 2

Etc.

Text to the left of row 1
Text Row 1 = Title for Row 1

Text to the left of row 2

Text Row 2 = Title for Row 2

Etc.

Optionally a column may be defined for switching RGA(s) between Automatic (no user control of the RGA – all commands come via ProcSECS) and Manual (user controls the Process Eye application – no commands from ProcSECS). The column should be given a title similar to "RGA Auto/Manual" and will display either Automatic or Manual in the appropriate cell once the RGA has run. The column to use must be defined (the rows used will always be 1,2,… Max RGAs) and the RGA in each row must be defined in terms of serial number for Process Eye V1.6x or head number for Process Eye V1.4x, e.g. for two heads that will use column 3:

Manual RGA Column = 3
Max RGAs = 2
RGA 1 = LM70-00197005
RGA 2 = LM70-00197006

3.9. Status Variable Polling

All tools maintain a number of Status Variables giving details of the tools current status that may be polled using the standard SECS II S1F3 message with the data coming back in an S1F4 reply. The data may be displayed in the Grid Display by defining a row and column where it should be shown. All items go in the [SVID STATE] section.

The number of Status Variables to be polled:
SVID Count = 3

The period between polls in seconds (1 to 60):
Interval = 2

The data type for SVIDs (note: This is not the individual data types but the ID type)
SVID Type = S2_U2

For each Status Variable 1,2,… SVID Count

The Status Variables ID number (taken from the tools SECS manual)

SVID 1=48

The Row where the data should be displayed. A row of 0 will prevent the data being displayed. (see below for information on default setting if Row x = y is omitted)
Row 1 = 1

The Column where the data should be displayed. A column of 0 will prevent the data being displayed. (see below for information on default setting if Col x = y is omitted)
Col 1 = 1

If Row and Col are omitted the Row/Col settings will take the Row/Col of the previous variable with one added to the Col. If this causes it to “wrap” past the last column then the column will be set to one and the Row will be incremented by one. The only exception to this is if the previous Row or Col was 0 then the default value will also be 0.

Section 4.
Macros

4.1. Overview

The ProcSECS macro capability was added for two general reasons. Firstly the basic functionality of the pre-programmed operation may be augmented where it is not powerful enough and secondly some or all of the handling of SECS messages from the tool can be done via the macro.

The macro language is Visual Basic for Applications ™ (VBA) compatible with extensions that allow manipulation of SECS messages in addition to controlling RGA(s).

Each time one of a number of events occurs the macro will be called with a variable set that indicates which event has happened. The blank form of the macro that should be used is in MTSECS.BAS. It should be edited either using VB as a editor or a text editor such as Notepad.Exe and then renamed SECS.BAS and placed in the same directory as the ProcSECS.EXE file.

The form of a ProcSECS macro is:

```
Const PRIMARY_MSG  = -3
Const STATUS_MSG = -2
Const REPLY_MSG = -1
Const START = 1
Const STOP = 2
Const S1F4_POLL = 3
Const ALARM = 4   ' Generic GEM/SECS
Const CE = 5
Sub Main ()

Do
        Select Case EventID
                Case PRIMARY_MSG ‘ Macro is handling this primary message
                Case STATUS_MSG     ‘ RGA status message
                Case REPLY_MSG      ‘ Reply to message sent in macro
```

```
            Case START          ‘ ProcSECS started
            Case STOP           ‘ ProcSECS stopped
            Case S1F4_POLL      ‘ S1F4 Polled data message
            Case ALARM          ‘ Tool generated an alarm
            Case CE             ‘ Tool generated a Collection
Event
        End Select
        Stop
Loop
```

The appropriate code going in each Case section as required.
Most of the events occur with no additional configuration but to allow “raw” SECS II messages to be passed through the ProcSECS application to the macro running in ProcSECS the following items in the ProcSECS.Ini file must be set:

ProcSECS Default Handler
Alternate Message Handler
Alternate Handler Count
Alt Stream N
Alt Function N
Auto Drop Ticket

See the configuration section for more details.

To facilitate Macro to tool and Macro to RGA communication the follow language extensions have been made available.

4.2. Read Only Variables

4.2.1. EventID

An Integer giving the event that caused the macro to be called. Value will be from –3 to 6 with 0 excluded.

-3 = Primary SECS message that the macro is handling
-2 = Status message from an RGA
-1 = Reply message to a primary message sent from within the macro
1 = ProcSECS program start
2 = ProcSECS program stop

3 = SVID Poll data available
4 = S6F67 Collection Event
5 = S6F69 Recipe Start Event
6 = S6F71 Recipe Stop Event

4.3. Read Only Arrays

4.3.1. S1F4Data(Index)

A variant array that holds the data from the most recent S1F3/4 poll. Index must be an Integer in the range 1 to SVID Count. If Index is outside of this range 0 will be read for that array element.

4.3.2. AlarmData(Index)

A variant array of three elements that holds the data from the most recent S6F67 Event. Index must be an Integer in the range 1 to 3. If Index is outside of this range 0 will be read for that array element.
Index = 1 gives the ALCD
Index = 2 gives the ALID
Index = 3 gives ALTX

4.3.3. CEReportData(Index,ReportID)

A multi dimensional array giving the data received in reports. Index is an integer giving the item within a report. Report ID is an optional integer – Report ID 1 will be used if it is omitted.

4.4. Subroutines

4.4.1. MessageToTool MessageString, MacroHandlesReply

MessageToTool sends the message in MessageString to the tool. MessageString is a string consisting of a comma delimited series of parameters:
Stream – the SECS Stream – 0 to 255
Function – the SECS Function – 0 to 255
W bit – the SECS W bit – 0 or 1
Item Count – the number of SECS data items
Data Type Item 1 – the type of data for the first item (See Appendix 1 for

details)
Data Item 1 – the first data item
.
.
Data Type Item n – the type of data for the last item
Data Item n – the last data item

MacroHandlesReply is an optional boolean parameter – if omitted it will be taken as true which means if the W bit is 1 the macro must handle the reply message. If false then the reply will be handled by ProcSECS.

Example to request Equipment Constant 5 by sending an S2F13 and having the macro handle the reply:

MessageToTool "2,13,1,2,4,1,29,5"
(i.e. Stream = 2, Function = 13, W bit = 1, Item Count = 2, Data Type Item 1 = List = 2, Data Item 1 = 1, Data Type Item 2 = I2 = 29, Data Item 2 = 15)

4.4.2. MessageToRGA MessageString, RGA

Sends the MessageString to the RGA in the RGA string. For V1.6x RGA is the serial number, for V1.4 it is the head number (as a string)
Valid commands are:
END RECIPE – halts currently running recipe
LOCKOUT,{Lock} – disables/enables the head. Lock = 1 locks it, Lock = 0 unlocks it
RUN,{Recipe name},{DDE Filename} – runs a recipe. DDE filename is optional
SET UDH,{UDH} – defines the user disk header, UDH is a comma delimited string of the individual headers
SECS,{Message} – a SECS message that may be processed by the Process Eye macro
ADD COMMENT,{Comment} – adds a text comment to the comment file for a running recipe
FIL,{Setting} – turns the currently selected filament on (1) or off (0)
SET GLOBAL,{Global Variable Number},{Data} – stores the data in the selected global variable for use by a recipe macro.
TERMINATE – closes down the instance of Process Eye

4.4.3. WriteToGrid Row, Col, Data, FormatString

Write the Variant Data into Row,Col of the Grid. By default it will be formatted in scientific notation but the optional variant FormatString may be used and the data will be formatted using Format$(Data,FormatString)

4.5. Functions

4.5.1. IsRGARegistered (RGA)

Returns a Boolean to indicate if the RGA (serial number or head) in the String RGA has run up and registered itself with ProcSECS and is therefor ready to take commands.

4.6. Process Eye Programming

Messages not generated or handled within ProcSECS can either be generated and handled by macro code within Process Eye or by a stand alone program.

There are two types of message in to your application:
Responses to Primary messages with the W bit set sent by your application
Primary messages from the tool

To allow “raw” SECS II messages to be passed through the ProcSECS application to either a macro running in Process Eye or a stand alone application the following items in the ProcSECS.Ini file must be set:

ProcSECS Default Handler
Alternate Message Handler
Alternate Handler Count
Alt Stream N
Alt Function N
Auto Drop Ticket

See the configuration section for more details.

4.7. Macro Language Extensions

Process Eye hides some of the details of a SECS link with the following

Functions, Subroutines and Variables. These are in addition to the existing macro extensions detailed in the Process Eye manual

4.7.1. Read Only Variables

SECSMessageCount – an integer giving the number of messages currently waiting in the SECS message input buffer.

4.7.2. Functions

SendSECSMessage(Message As String) – Transmits a primary SECS message (for secondary messages use SendSECSReply) and returns an Integer to indicate the message was successfully transmitted to ProcSECS. This does not guarantee the message made it to the tool. The Message is a string of comma delimited values:
Stream – the SECS Stream – 0 to 255
Function – the SECS Function – 0 to 255
W bit – the SECS W bit – 0 or 1
SECS ID – a string used to allow your application to identify which of the incoming reply messages belongs with this primary.
Item Count – the number of SECS data items
Data Type Item 1 – the type of data for the first item (See Appendix 1 for details)
Data Item 1 – the first data item
.
.
Data Type Item n – the type of data for the last item
Data Item n – the last data item

SendSECSReply(Message As String) – Transmits a primary SECS message (for secondary messages use SendSECSReply) and returns an Integer to indicate the message was successfully transmitted to ProcSECS. This does not guarantee the message made it to the tool. The Message is a string of comma delimited values:
Stream – the SECS Stream – 0 to 255 – the same as the incoming message
Function – the SECS Function – 0 to 255
Ticket – the SDR ticket received with the incoming message
Item Count – the number of SECS data items
Data Type Item 1 – the type of data for the first item (See Appendix 1 for details)
Data Item 1 – the first data item

.
.
Data Type Item n – the type of data for the last item
Data Item n – the last data item

PollSECSData(SECS_ID As String) – searches the messages currently waiting in the SECS message input buffer for the reply to your primary message with the specified SECS ID, returns a Boolean, true if the message is waiting, false if it is not.

GetSECSData(SECS_ID As String) – searches the messages currently waiting in the SECS message input buffer for the reply to your primary message with the specified SECS ID. If the message is found it is returned as a string, if it is not found a null string is returned. The message string is a comma delimited string in the following format:
SECS ID the string sent by your application when transmitting the primary message.
Ticket: the SDR transaction ticket. It can usually be ignored except it is used in the SendSECSReply message or if, despite all our warnings, you disable Auto Drop Ticket feature.
Stream – the SECS Stream – 0 to 255 – the same as the incoming message
Function – the SECS Function – 0 to 255
W Bit – the SECS W bit – 0 or 1
Item Count – the number of SECS data items
Data Type Item 1 – the type of data for the first item (See Appendix 1 for details)
Data Item 1 – the first data item
.
.
Data Type Item n – the type of data for the last item
Data Item n – the last data item

PopSECSData() – returns a String that is the last message put into the SECS message input buffer. If no messages are in the buffer it returns a null. The message is in exactly the same format as the GetSECSData described above.

SendSECSDrop(Ticket As String) – if you have taken responsibility for

dealing with SDR tickets this function is used to drop tickets when they are no longer required. If tickets are not properly dropped the SDR application will eventually (after about 255 tickets are used) stop functioning. It is strongly recommended that the Auto Drop Ticket capability of ProcSECS is used.

RemoveSECSHeader(SECSMsg As String) – returns the Item Count (integer) in a message and the message in SECSMsg is returned with the SECS_ID, Ticket, Stream, Function, W Bit and Item Count data removed.

NthCommaValue(CommaDelimitedString As String, N As Integer) returns a variant which is the Nth position in a comma delimited string. Note each item consists of two values, the data type and the data. Multiply the item number by 2 and subtract 1 to point at the data type and multiply the item number by 2 to point at the data.

Example:
To poll Status Variable ID 4 once per scan and store it in Action Channel 1. (Note there are a number of different methods of implementing this particular feature and using the ProcSECS Macro capability should also be considered)

```
Global Result as Integer    ' general variable to store SECS command status in
Global SECS_ID as Long  ' incrementing variable that tracks message

Sub Main ()
        PreTrigger
        PauseMacro

        PostTrigger
        PauseMacro

        PreRun
        PauseMacro

        Do While Scanning
                EndOfScan
                PauseMacro
        Loop
```

```
        PostRun
End Sub

Sub PreTrigger ()
‘ Request value of SVID #4 for the first time using S1F3 with W set
        Result = SendSECSMessage(“1,3,1,” & Cstr(SECS_ID) & “2,4,1,29,4")
End Sub

Sub PostTrigger ()
End Sub

Sub PreRun ()
End Sub

Sub EndOfScan ()

Dim S1F4Message as String       ‘ Variable to hold the S1F4 reply to our S1F3 message
Dim S1F4ItemCount as Integer    ‘ Variable to hold number of data items in S1F4 reply
Dim SVID4 as Variant            ‘ Variable to hold SVID 4 value

        S1F4Message= GetSECSData(Cstr(SECS_ID))
        S1F4ItemCount = RemoveSECSHeader(S1F4Message)       ‘ Remove header, get ItemCount
SVID4 = NthCommaValue(S1F4ItemCount * 2, S1F3Message)       ‘ *2 as 2 values per Item
WriteHugeArrayData          (1,ScanNumber,ActionChannel(1),CSng(SVID4))
        ‘ Save
        SECS_ID = SECS_ID + 1           ‘ Use a different SECS ID
        Result = SendSECSMessage(“1,3,1,” & Cstr(SECS_ID) & “2,4,1,29,4") ‘ Request SVID 4 again
End Sub

Sub PostRun ()
End Sub
```

4.8. Miscellaneous Notes

For TCI/IP links it may be required to have the Passive end of the link

running first.
It is essential that any instances of Process Eye that are to communicate with ProcSECS are run after ProcSECS has initialized.

Appendix 1
Date type names

Names to use when defining data types

Name	Macro Value	SECS II Name
S2_L	4	List
S2_B	8	1 Byte Binary
S2_BOOLEAN	12	Boolean
S2_A	16	ASCII
S2_U1	24	1 Byte Unsigned Integer
S2_U2	25	2 Byte Unsigned Integer
S2_U4	26	4 Byte Unsigned Integer
S2_U8	27	8 Byte Unsigned Integer
S2_I1	28	1 Byte Signed Integer
S2_I2	29	2 Byte Signed Integer
S2_I4	30	4 Byte Signed Integer
S2_I8	31	8 Byte Signed Integer
S2_F4	34	4 Byte Floating Point Number
S2_F8	35	8 Byte Floating Point Number

Recipe Lookup syntax

[RECIPE LOOKUP]
ToolRecipeName1 = RGARecipeName1
ToolRecipeName2 = RGARecipeName2
ToolRecipeName3 = RGARecipeName3

This page is deliberately left blank.