MASsoft
Event Sequences
Guide

Hiden Analytical Limited
420 Europa Boulevard
Warrington
WA5 7UN
England

Tel: +44 (0)1925 445225
Fax: +44 (0)1925 416518

E-mail: info@hiden.co.uk
Web site: http://www.HidenAnalytical.com

Document Number: HA-085-660 **Issue: B**

MASsoft Events Sequence Guide

Manual Number: HA-085-660 **Issue: B**

Date: 20 April 2015

The Hiden Analytical Limited Website

Information on Hiden Analytical Limited and its wide range scientific instruments for research, development and production applications can be obtained from the Hiden Analytical Limited Website at http://www.HidenAnalytical.com.

Contents

Illustrations

Welcome

The purpose of this manual is to describe MASsoft Event Sequences.

Copyright notice

Licence and Warranty agreement

Hiden Analytical Limited licenses you to use the Personal Computer (PC)-based software on a single PC connected to the equipment, subject to the terms of this agreement. The software may be moved from one computer to another, as long as it can only be used by one person at a time. The transfer, sublicensing or assignment of the PC software to another party is permitted provided that party accepts these terms and conditions. In such a case, any copies of the software not transferred must be deleted. One copy of the software may be made for protection against loss or damage; a further copy of the software may be made for off-line data analysis purposes.

Acknowledgements

Warnings and Cautions

In this Manual, a **Warning** is an instruction that draws the operator's attention to the risk of injury or death; a **Caution** is an instruction that draws attention to the risk of damage to the product or process.

Warnings and Cautions are placed immediately before the text to which they refer; they are headed by **WARNING** or **CAUTION** respectively. The associated explanatory text is in **bold**. If several Warnings or Cautions apply at one point in the text, they are numbered with the most important appearing first.

Typographical conventions

For ease of identification the names of menu commands, keys, dialogue items and screen text are typographically distinct from the ordinary text of this Manual. These distinctions are as follows:

Menu commands, dialog box items, such as buttons and check boxes, and text that appears on the display screen are presented in bold typeface; thus **File** menu, **Enter** button.

Keys are presented in bold, italic text; thus ***Esc***, ***Return***, ***Space bar***.

Terminology

Terminology that accords with basic Windows principles is included in this Manual.

Instructions

For clarity, the instructions given in this Manual are presented in two columns. The left-hand column provides imperative instructions that are numbered sequentially to provide a step-by-step guide through the functions. The right-hand column describes the system's response (where appropriate) and gives any additional information that may be of relevance.

Importance of this Manual

This Manual should be regarded as part of the product described herein.

Technical assistance

Technical assistance can be obtained from the Hiden Analytical Limited Service Department which can be contacted on:

Email: service@hiden.co.uk

Tel: +44 (0)1925 445225

Fax: +44 (0)1925 416518

In the U.S.A. and Canada, technical assistance can be obtained from Hiden Analytical Inc.:

Email: service@hideninc.com

Tel: 603 924 5008

Fax: 603 924 5009

Toll-free phone: 1-888-96 HIDEN
Option 1 U.S.Sales Office
Option 2 U.S.A. & Canada Corporate Office & Service Department
Option 3 U.K. Manufacturing Facility

Amendments

This Manual will be updated, as necessary, to cover modifications to the product. Minor amendments may take the form of Addenda, which will normally be located at the back of the Manual, on coloured paper.

Amendment history

Issue: A Date: 22 March 2007

Issue: B 20 April 2015. M085-567. Section 5 added.

1 Introduction

1.1 What are Event Sequences?

Event sequences are a way to programme the Mass Spectrometer. Event sequences are run by the microprocessor in the instrument's Interface Unit (IU), not by the PC. The PC just acts as an editor. MASsoft downloads the event sequences to the Interface Unit as a series of IU commands. This means that Event Sequences can respond faster to data than if they were run in the PC because they do not have to wait for data to be sent from the IU to the PC. Event sequences have direct access to the resources of the IU, such as IO lines on the auxiliary and trip connectors. However, it also means that Event Sequences do not have access to the resources of the PC like disk storage, network connections and communication with other applications.

1.2 Understanding event sequences

Communication between an Event Sequence and MASsoft mainly takes place as messages sent to the Event Log window.

Event Sequences are not like a conventional programming language. In a conventional programming language, such as BASIC, the following might be written to print the result of multiplying 2 by 2:

```
B = 2 * 2
PRINT  "2 X 2 = "; B
```

These commands are typed into a file and either interpreted (in a language like BASIC) or compiled to a programme file which is then run (in a language like C).

The equivalent in an Event Sequence would be :

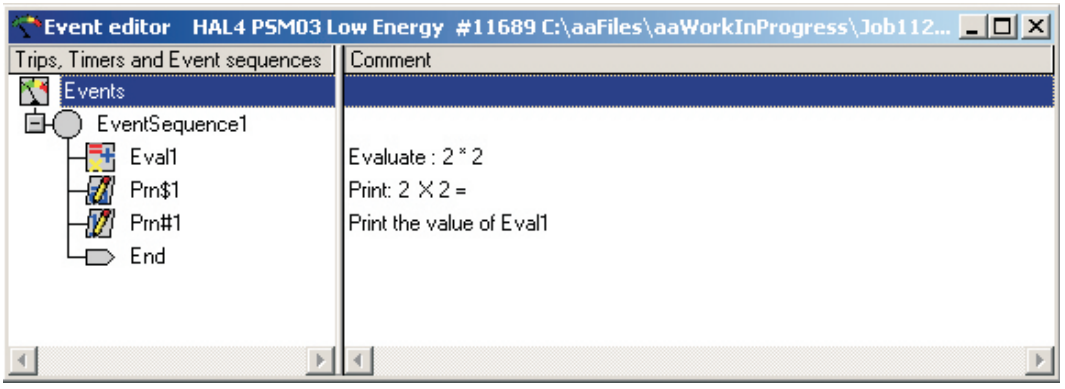

Figure 1 Print 2x2 Event Sequence

Understanding Event Sequences requires slightly different thinking than is needed to understand a conventional programming language. Each step in an Event Sequence is represented by an Event "object".

The Event objects have “properties” that can be set in each Event object dialog box. When the event sequence is run, how each step behaves is determined by the type of Event object at that step and by the properties set for the Event object.

In the above example the properties of Eval1 are:

Name: Eval1
Type: Evaluate event
Comment: Evaluate : 2 * 2
Output Format:
Source 0: 2
Operator 1: *
Source 1: 2
Operator 2:
Source 2:
Operator 3:
Source 3:
Operator 4:
Source 4:

The properties are shown like this when an event sequence is printed, or when it is copied and pasted into a document. These properties are shown below as they would appear in the editor dialog box.

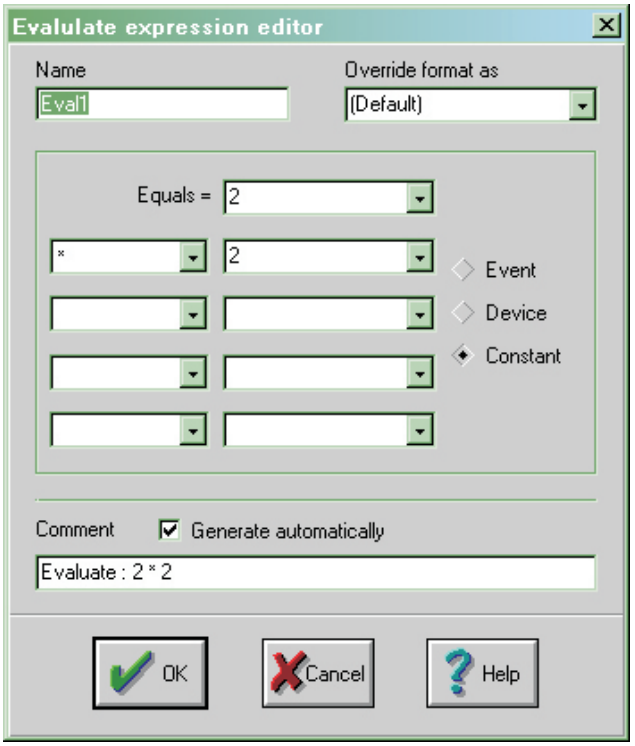

Figure 2 Evaluate expression editor

Not only do Event objects have properties but they also have an associated value and device type. To draw an analogy with spreadsheets. A cell in a spreadsheet may contain a value or a formula. If it contains a formula the value is calculated by evaluating the formula. If the cell is referred to in a second cell the value in the second cell in calculated based on the value in first. Likewise, a step in an event sequence can be referred to from another and the value of the first used in the calculation of the second. So the value of Eval1, after it has been run, is 4. This what Prn#1 prints.

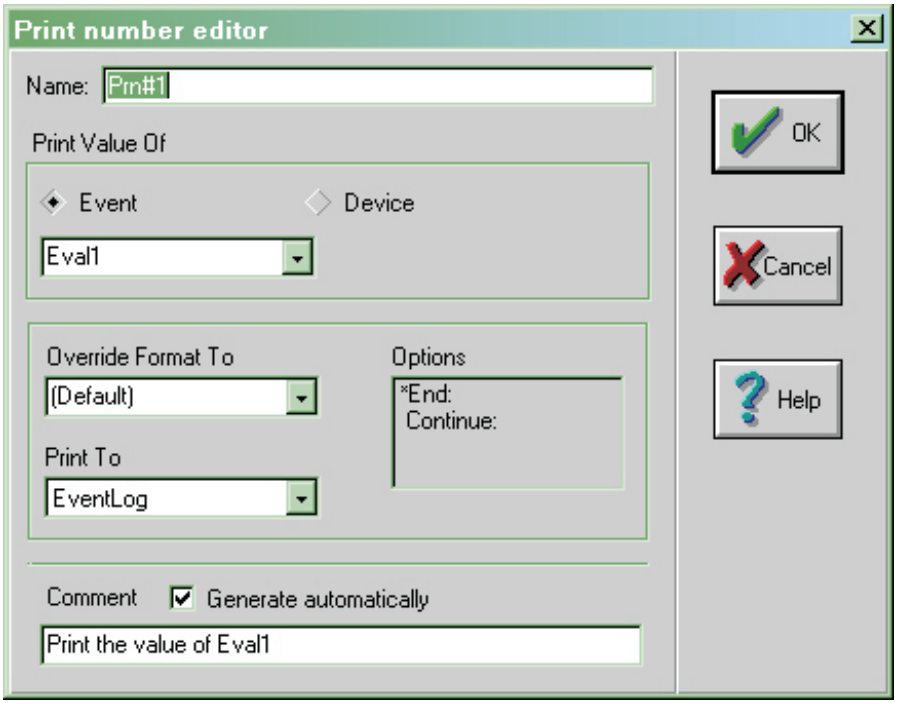

Figure 3 Print number editor, Prn#1

The device type associated with a value tells the Print Number event object how to format the value. For instance, the current multiplier HT voltage is formatted as an integer with three decimal places whereas the Faraday detector value is given in exponential format.

In the case of a constant (like 2 * 2) no device type can be associated with the value; in this case it defaults to the format of the main input device (Faraday or SEM). If this is not appropriate it can be over-ridden by selecting a device in the “Override Format To” property.

The associated device type also allows the Evaluate type events to convert different types of value (integer, fixed decimal place and floating point) and affects how values are rounded off when stored as an Event object value.

Many types of Event objects have an Options property. The Options for each type of Event object are read from the IU when the instrument is interrogated. The options available depend on the firmware version of the IU. The options shown in this manual are those for release R5.4.

An option is selected or de-selected by double-clicking it. Selected options have a * in front of them.

All Event objects have a Comment property. When the event object is edited MASsoft automatically generates a comment describing the main function of the Event object. The comment may be changed to something more meaningful. Comments are displayed in the right hand pane of the **Event Editor** window.

Summary

An Event Sequence consists of a sequence of Event objects.

An Event object has properties, a value and a device type.

The device type affects how values are stored, rounded and displayed.

Options depend on the IU's firmware version.

MASsoft can automatically generate comments when Event objects are edited.

2 Basics

2.1 The Set Event

The Set Event's primary function is to set a device to a value. The "device" is usually a piece of hardware like a power supply or a relay. The power supply might drive a lens in the analyser. The relay might be an output like trip1 or might switch some of the hardware on and off, like a filament.

It is possible, with suitably factory configured firmware, to set up dummy devices that have no physical existence or "logical" devices that affect the state of one or more other devices.

A sequence to turn Filament 1 off and Filament 2 on is shown in Figure 4.

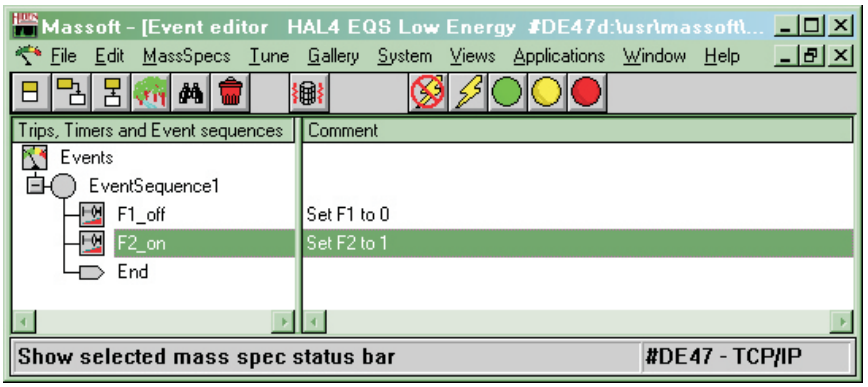

Figure 4 Filament sequence

If a device has two states, on and off, then usually setting the device to 1 turns it on and setting it to 0 turns it off. F1 is set to 0 to switch filament 1 off and F2 is set to 1 to switch filament 2 on.

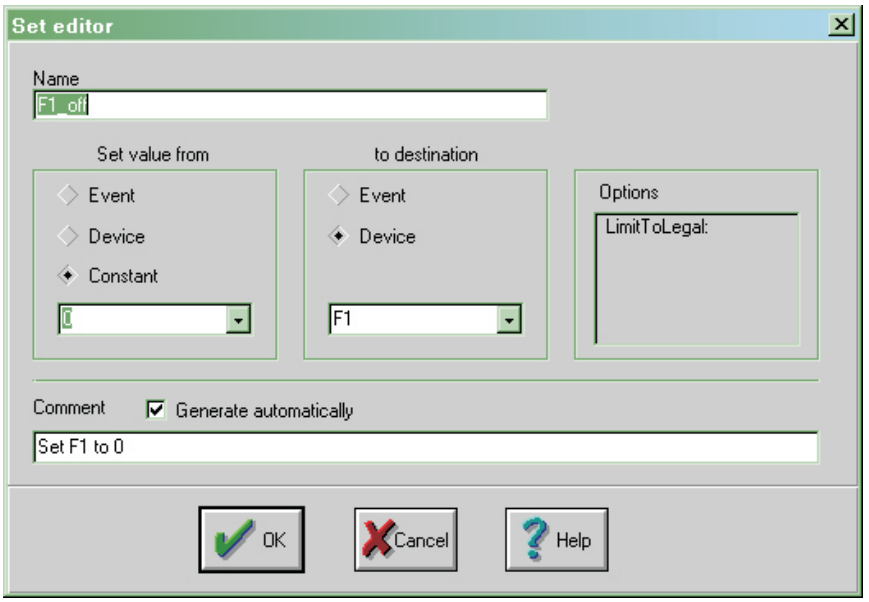

Figure 5 Set editor, filament 1 off

The value used to set the device need not be a constant, it could be a value calculated by an Evaluate event. For instance, the voltage on an analogue output could be proportional to the percentage concentration of a gas mixture component.

Figure 6 shows the Set editor dialog box to set the analogue output Channel_1 to the value of the event Eval1.

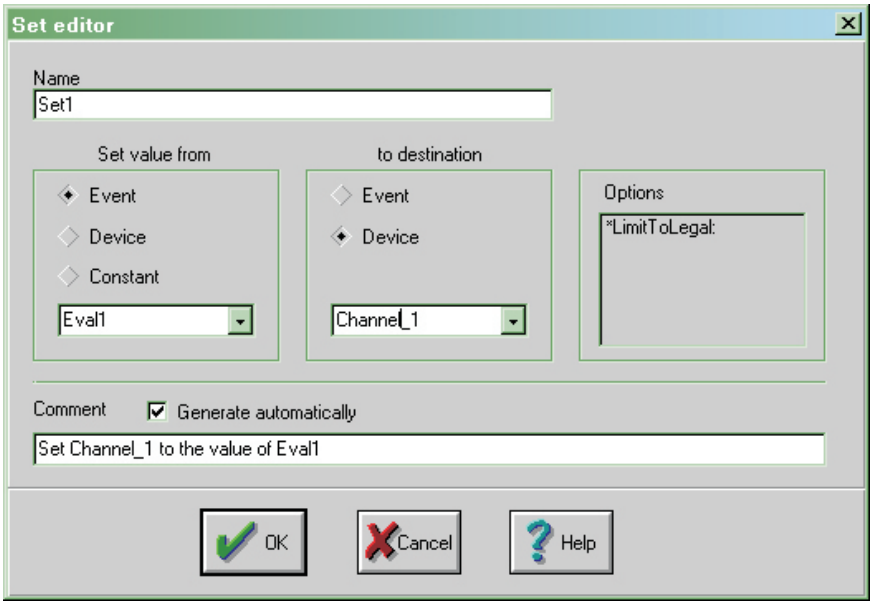

Figure 6 Set editor, Set1

The source property is Eval1, the destination property is Channel_1. The Option LimitToLegal is set, this ensures that the value written to Channel_1 does not exceed the devices limits of 10V. This will prevent an error message being generated if, for instance, 1% Argon is set to give 10V output and then is actually measured as 1.2%.

A Set Event can copy a value from one device to another if both the source and destination are similar devices.

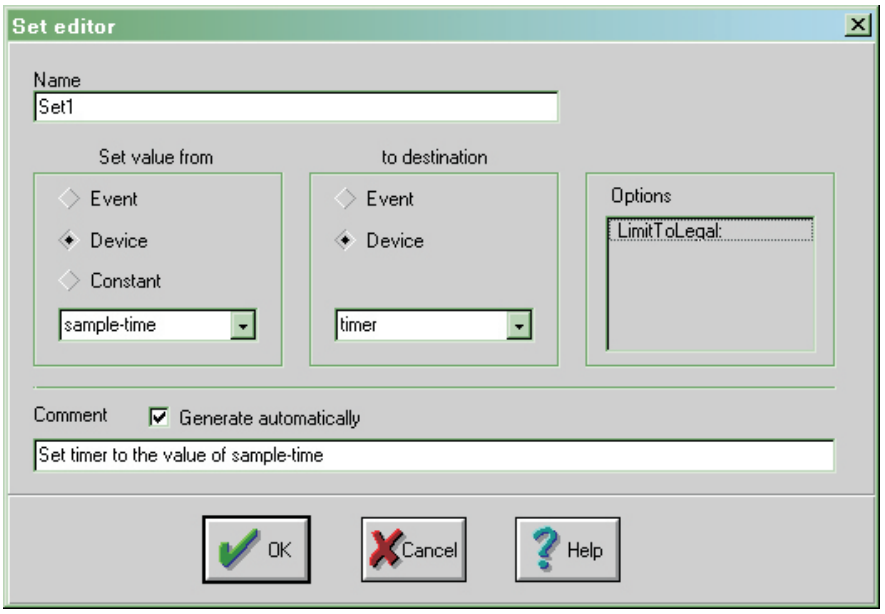

Figure 7 Set editor, copy values

Figure 7 shows the **Set editor** dialog box being used to copy the value from a dummy device called **sample-time** to the logical device called **timer**. **sample-time** exists solely to allow a user to enter a value which can be read by an event sequence. Setting the value in the Global Environment does not actually do anything at the time it is set. **timer** is an example of a logical device which has no corresponding existence in hardware. **timer** will cause the Event Sequence to pause for a number of seconds equal to the value of **timer**.

Finally, a Set Event can copy a value from an event, device or constant to another event. When a value is copied from a device to an event object the event-object's value is set to the value read from the device (i.e. the devices current setting if it is an output device or a reading from the device if it is an input device). The event's associated device type is set to the source device itself.

When a value is copied from one event to another the destination event's value and device type are made the same as the source event's value and device type.

A set event is also used to:

- copy a value to an event object when printing values using the #=value option in Print Text events
- to set Timer events
- to store the value in a "variable".

These are described later in this manual.

2.2 The Limit Event – Making decisions

The Limit Event object allows a value to be compared with upper and lower limits and an action taken depending on whether or not the limits were met. One of the actions a Limit Event can take is to branch to another Event object. The Limit Event is equivalent to an IF statement in high level programming languages.

A simple example of the use of the Limit event is this sequence that changes to filament 2 if filament 1 fails.

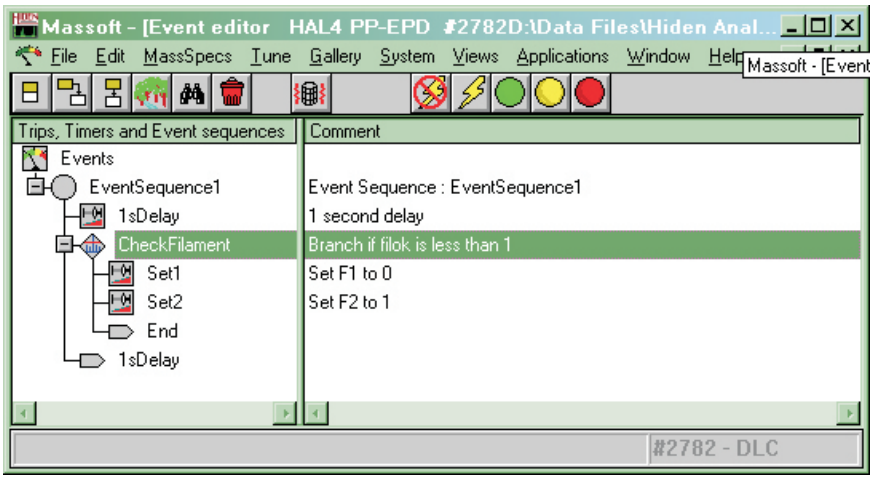

Figure 8 Limit event sequence

The sequence shown in Figure 8 runs after the scan has started, so the filaments will have been turned on.

The sequence reads the device *filok*. *filok* reads an input from the source control circuit which checks the current flowing through the filament, it returns 1 if the filament is intact, 0 if it is open circuit.

Action needs to be taken if the filament fails. When the filament is okay *filok* =1. Filament fail is indicated by *filok* being 0 (or rather less than 1). To test if *filok* is less than a value, the < *Less than lower limit* type of trip is chosen. < *Less than lower limit* always compares with the Lower limit, so 1 is entered as the lower limit :

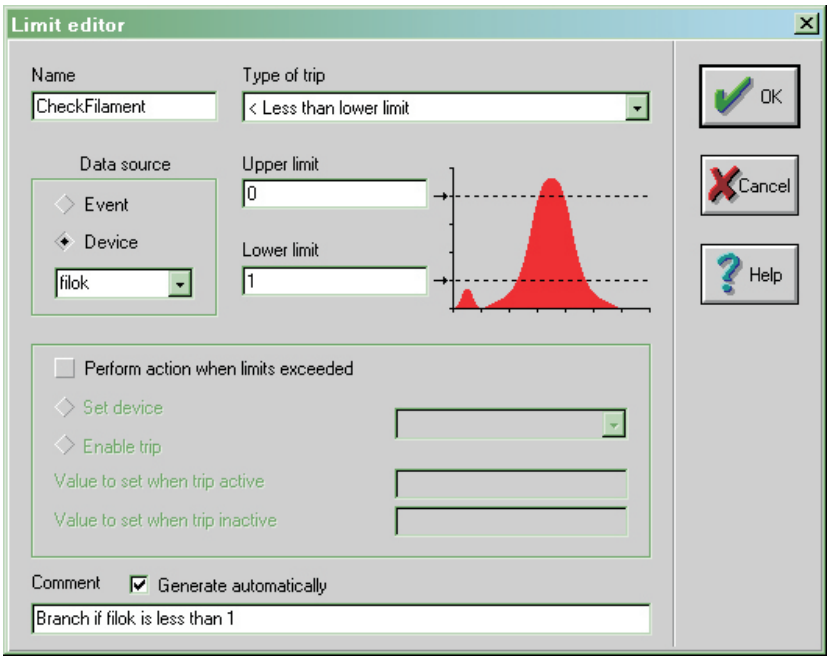

Figure 9 Limit editor dialog box

Note

Counter intuitively, the Lower limit value is higher than the, unused, Upper limit. This odd situation only exists for binary devices like filok that can only have the values 0 and 1. Other data sources like mSecs, the elapsed time in milliseconds, might have <30s or >60s with 30000 in the Lower limit and 60000 in the Upper limit.

It would be equally valid to write this event sequence taking the branch if the filament is OK.

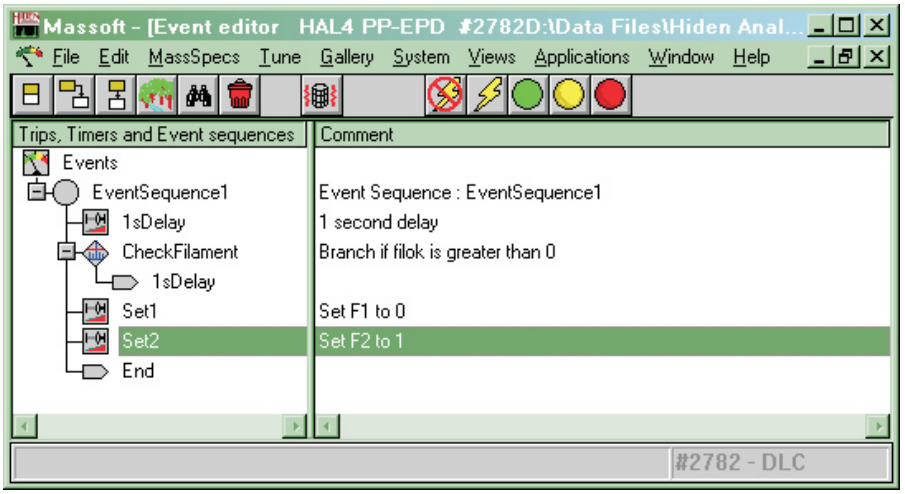

Figure 10 Event sequence, greater than upper limit

In this version of a check filament sequence (see Figure 10) **CheckFilament** uses > **Greater than upper limit** with 0 in the **Upper limit**, as shown in Figure 11.

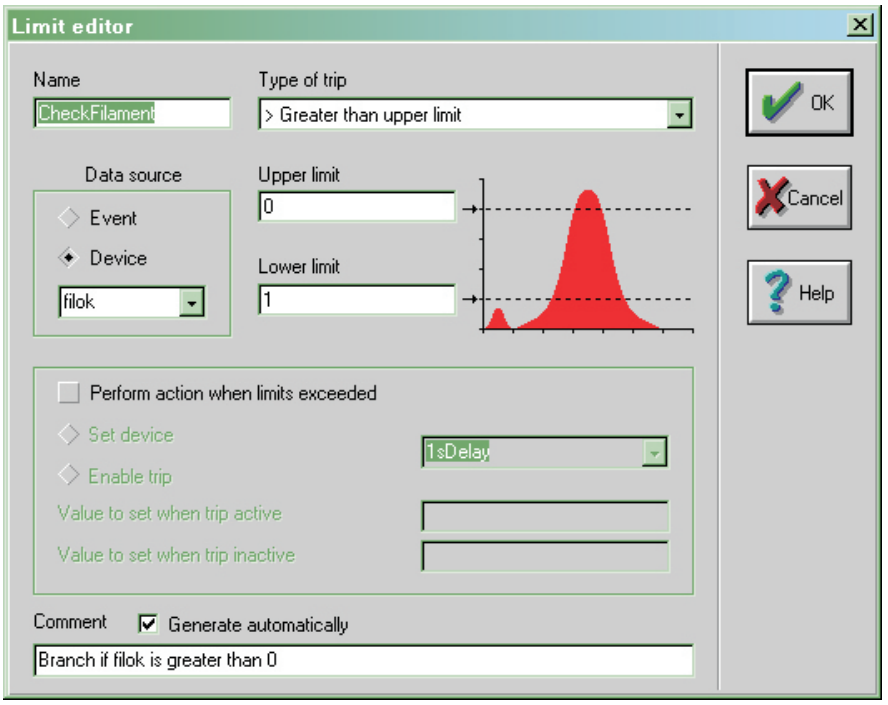

Figure 11 Limit editor, check filament, upper limit

Note

Both of the check filament sequences go to End after changing from F1 (filament 1) to F2 (filament 2). There is no point checking F2 because there is no further action that can be taken. There is no third filament to change to. The only thing a user might want to do is display a message to the log and stop the scan.

The full range of options for the **type of trip** property are described in Section 4.

2.3 Sources and Destinations

When thinking about event objects it can be helpful to think in terms of data sources and data destinations. A data source is somewhere to get a value from and a data destination is somewhere to store that value.

For instance the Limit event in the previous section has a Data source property set to filok. The radio buttons Event and Device determine if a list of Events or a list of Devices appears in the Data source drop down list. In the listing this is always described as Device Source. Whether this is a device or event is implicit in the name because events are not allowed to have the same name as devices.

Name: CheckFilament
Type: Limit event
    Comment: Branch if filok is greater than 0
    Trip Logic: >
    Device Source: filok
    Upper Limit: 0
    Lower Limit: 1
    Action: 1sDelay
    Active Value:
    Inactive Value:

Set events have a Source Value property:

Name: Set1
Type: Set event
    Comment: Set F1 to 0
    Source Value: 0
    To: F1
Options:

The Source Value is in a frame labelled “Set value from” in the Set Event editor dialog box because this better describes what the property does.

The frame labelled “To destination” in the Set Event editor dialog box is shortened to just To: in the property listing; this is the destination.

The Set event is the only event object that has a destination.

All event objects can be the destination of a Set Event. The Set Event will store the source value, with its associated device-type, in the destination event object. A value that has been stored in an event object can be retrieved by making that event object itself a source. Only Print Text events, with the #=value option, and Timer events make direct use of this value.

A source can be either a device or an event object, in Set and Evaluate events it can also be a constant. The source can not, directly, be data from a scan because of the need to specify other parameters. A Data event should be used to fetch data from a scan, the Data event can then be a source.

Set, Limit, Print Number and Evaluate events all have a Source. In an Evaluate event each term of the expression has a source.

2.4 Reading Data from a Scan

Readings can be read from a scan using the Data event. The Interface Unit buffers data until the memory fills up, at which point it discards the oldest value.

The Data event allows the cycle number from which to get data to be specified. Positive numbers specify an absolute cycle number; 0 specifies the current, most recent, value; -1 specifies the previous cycle; -2 the cycle before that and so on.

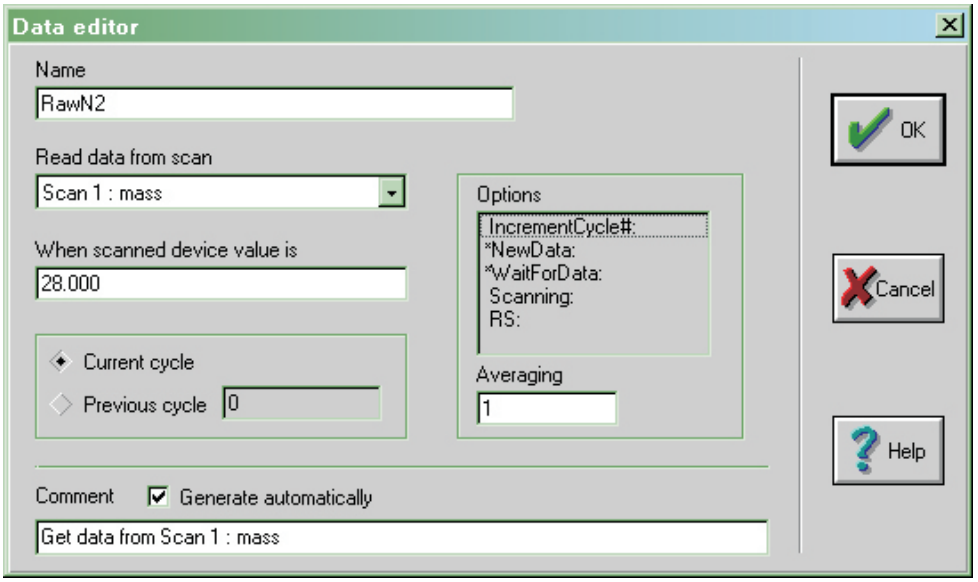

Figure 12 Data editor dialog box

Figure 12 shows a typical **Data editor** dialog box. The value of the mass 28 peak (Nitrogen / Carbon Monoxide) from a Bar mode scan is being read. The most recent value is being used.

The NewData option means the Data Event will only process each data point once. If the event sequence loops at a faster rate than the scan, which it probably will, the Data Event will wait until new data is available, pausing the Event Sequence.

The WaitForData option causes the Event Sequence to wait if mass 28 has not yet been measured, instead of producing a “No data” error.

The IncrementCycle# option is used when Previous cycle specifies an absolute cycle number. So if Previous cycle is initially 1, after cycle 1 has been read it increments to 2. This can be used to ensure every cycle is processed.

The Scanning option checks that the scan has actually started. It is better to use the *Start sequence - After start or restart of scan* option in the Event Sequence editor dialog box to run the event sequence.

The RS: option (relative sensitivity) is not supported by MASsoft yet. Use an Evaluate event to divide by the Relative Sensitivity.

Averaging calculates the mean of the specified number of values from previous cycles. If the specified number of cycles are not available then all the available cycles are averaged.

The Data Event also has an On error handler property to allow a branch to an error handler routine, but this is not directly supported by MASsoft. A Command event must be used to set the On error property.

After the value has been read by using the Data event it may be manipulated by using an Evaluate event. It may be checked to see that it falls between limits using the Limit event or displayed using the Print Number event.

2.5 Calculations, Variables and Constants

The Evaluate event can be used to perform calculations. In firmware versions before R5.4 it could only perform the basic arithmetic operations add, subtract, divide and multiply. Version R5.4 added Log and InvLog, Power and Root, Min and Max.

The Evaluate event calculates strictly left to right (or top to bottom as displayed in the **Evaluate expression editor** dialog box) with no operator precedence. Figure 13 shows the **Evaluate expression editor** dialog box.

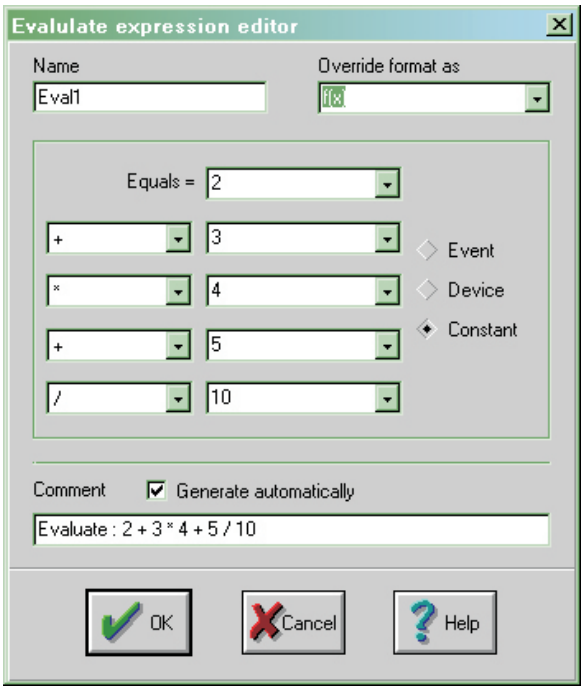

Figure 13 Evaluate expression event editor

The expression entered into the dialog box shown in Figure 13 is evaluated as:

2 + 3 = 5

5 * 4 = 20

20 + 5 = 25

25 / 10 = 2.5

not as 2 + (3 * 4) + (5 / 10) = 14.5

The evaluation of the operators + (add) - (subtract) * (multiply) and / (divide) is fairly obvious; these are binary operators, they take two values – one before the operator, another after.

Log, InvLog, Power, Root, Min and Max are not normally used in this way. They are usually used either as unary operators (e.g. Log 100 = 2) or as functions (e.g. log(100) = 2.

In an Evaluate event the value following a Log or InvLog "operator" indicates the base.

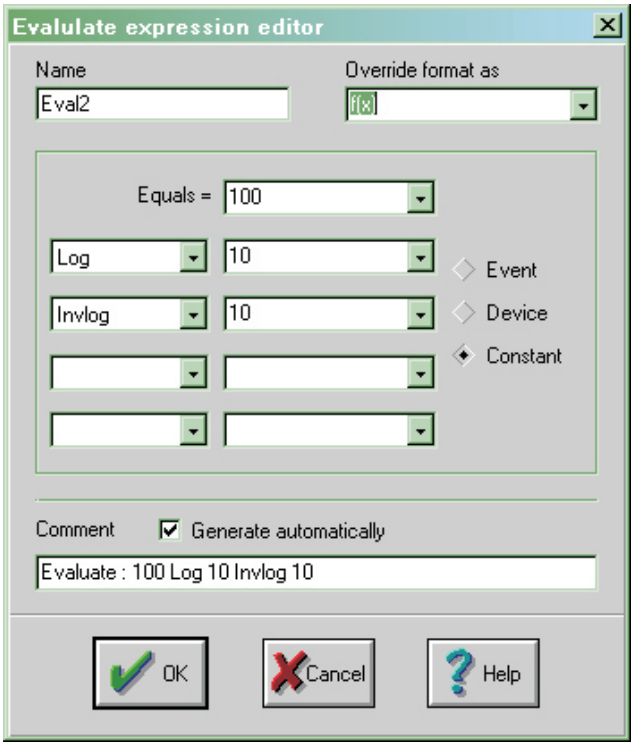

Figure 14 Evaluate expression editor, log invlog

Invlog is the inverse logarithm or anti-log. The anti-log of x to the base z is equivalent to z^x.

The expression entered into the dialog box shown in Figure 14 is:

$\log_{10} 100 = 2$

$10^2 = 100$

If a base of 0 or 1 is used the Evaluate event calculates a natural logarithm using base e. The Invlog with base 0 or 1 is equivalent to e^x.

In an Evaluate event using Power the value after the “operator” indicates the exponent. Likewise, the value after Root so, 2 is the square root, 3 the cube root etc.

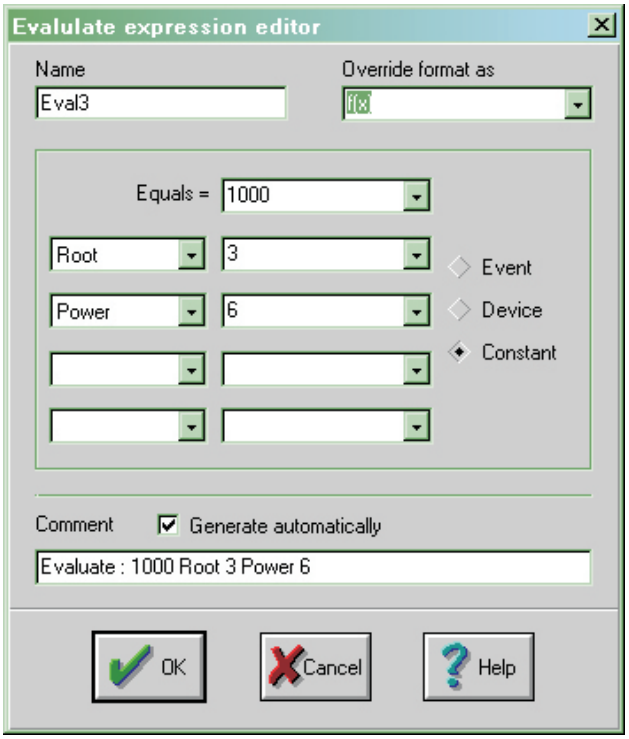

Figure 15 Evaluate expression editor, power and root

The expression entered into the dialog box shown in Figure 15 is:

$$\sqrt[3]{1000} = 10$$

$$10^6 = 1000000$$

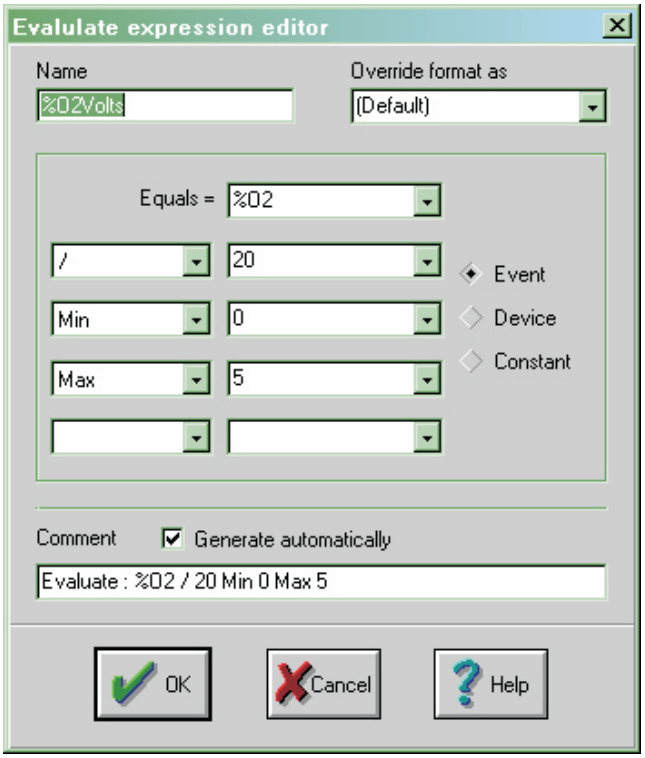

Figure 16 Evaluate expression editor, min and max

The Evaluate event shown in Figure 16 takes the percentage of oxygen calculated by another Evaluate event called %02 and converts it to a 0-5V analogue output. / 20 makes 100% equivalent to 5V. The Min and Max terms should be read as “a minimum of 0 and a maximum of 5"; so if %02 is less than zero (which it could be due to noise) then the result is 0V, if it is more than 5V (unlikely!) then the result is 5V.

Note

For a +/- 10V analogue output the Set event’s LimitToLegal option can be used to constrain the output to +/- 10V.

2.6 Using Evaluate Events

Evaluate events may be used as variables and constants.

Event Sequences do not have event types for constants and variables, instead an Evaluate event is used. The example shown in Figure 17 illustrates how an Evaluate event can be used as a constant or a variable.

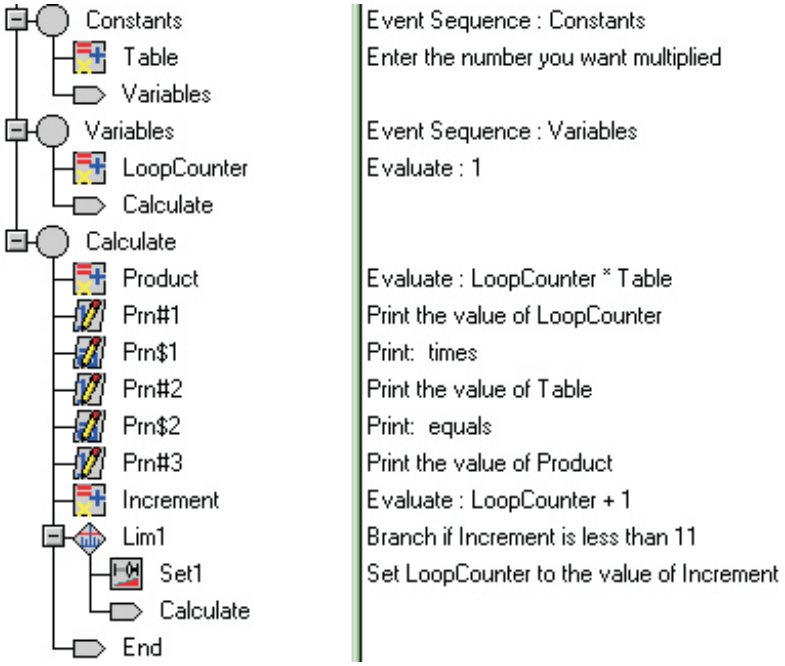

Figure 17 Evaluate event, variables and constants

Table is a constant that determines the times table that is displayed. This event object must be run to initialise it, otherwise the value 4 would remain just a property of the event and would not be stored in its value; for that reason this sequence starts running at *Constant*, jumps to *Variables* and then jumps to *Calculate*.

The event sequence is run by selecting the **Run event sequence** option in the **Event sequence editor** dialog box, see Figure 26. The dialog box can be opened by double clicking Constants or by pressing the ***Return*** key when Constants is selected.

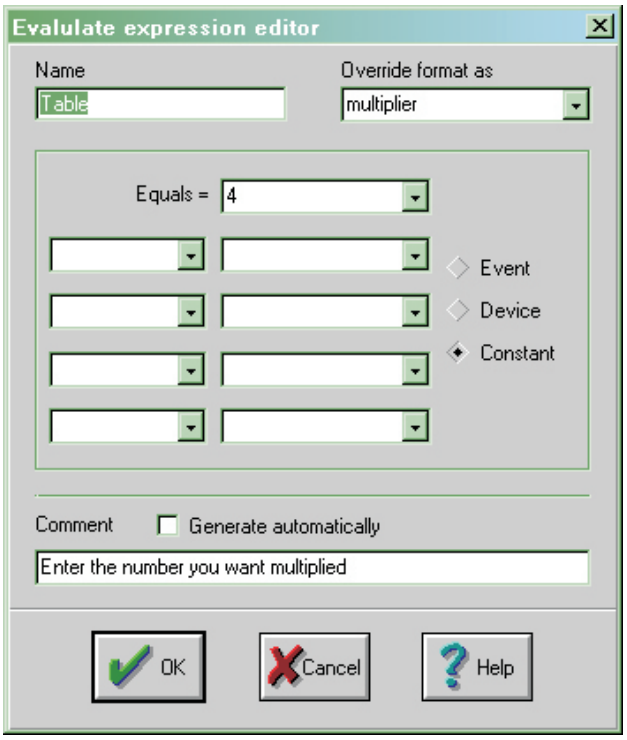

Figure 18 Evaluate expression editor, multiply

The “variable” LoopCounter is executed to initialise it to 1.

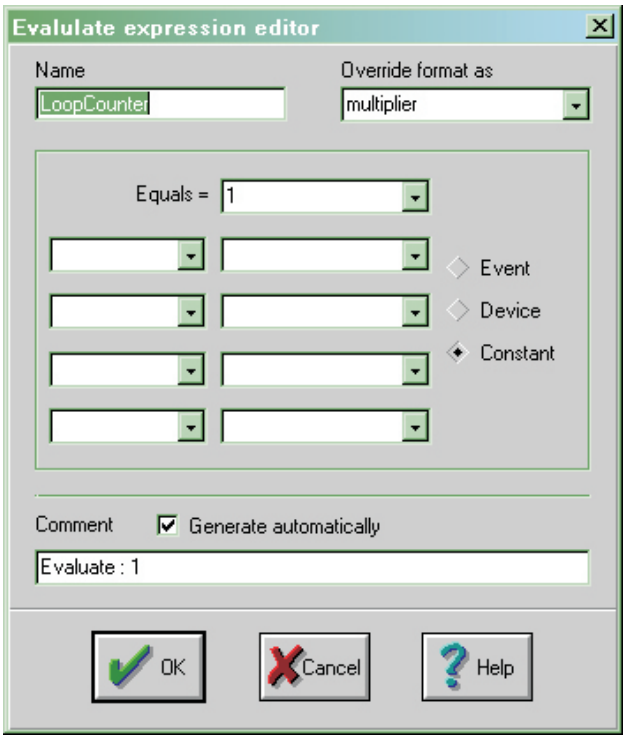

Figure 19 Evaluate expression editor, LoopCounter

Both **Table** and **LoopCounter** have the **Override format as** property set to the device **multiplier**. This results in printing the table with numbers as unsigned integers which happens to be the format used to display multiplier voltages (in this case multiplier has nothing to do with multiplication tables).

The remaining steps are well explained by the comments in the Event editor:
Product multiplies Table by LoopCounter
the next five steps display this
increment adds one to the value of LoopCounter
if the result of this is less than 11 (i.e. up to 10) the incremented value is stored in LoopCounter by Set1 and the sequence loops back to display the next product.

2.7 Displaying Values

There are two ways to display values, using a Print Text event with the #=value option or by using a Print Number event.

Print Number	Print Text with #=Value:
Gets data from source	Needs set event to get data
Can specify display format	Uses data source's format
Can use fixed decimal place formats	Must use source device's format
Just prints number	Embeds number in a message
Supported by all firmware	Only in firmware 5.4 or later

Table 1 Print Events

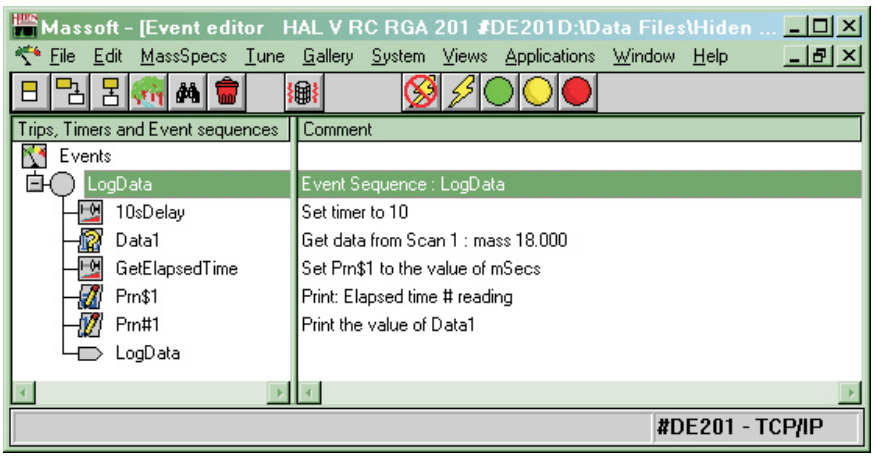

Figure 20 Log data event sequence

Sequence : LogData

The sequence logs data to the Event Log every 10 seconds.

Name: 10sDelay
Type: Set event
Comment: Set timer to 10
Source Value: 10
To: timer
Options:

The logged value is the average of the previous 10 readings. The *WaitForData:* option makes Data1 wait if the scan has not yet saved any data when it runs, *NewData:* option ensures that the same data is not logged twice.

Name: Data1
Type: Data event
    Comment: Get data from Scan 1: mass 18.000
    Source Scan: Scan 1: mass 18.000
    Device Value: 18.000
    Cycle: 0
    Average: 10
Options: WaitForData:,NewData:

A Set event is used to get the elapsed time from the device *mSecs* immediately after the Data event. It stores the value in *Prn$1* where it will be printed by the *#=value* option.

Name: GetElapsedTime
Type: Set event
    Comment: Set Prn$1 to the value of mSecs
    Source Value: mSecs
    To: Prn$1
Options:

The *Prn$1* event prints the text of the message, because the *#=value* option is set. The # is replaced by the value stored in *Prn$1* by *GetElapsedtime. Prn$1* has the *Continue* option set so that *Prn#1* will write to the same line as *Prn$1*.

BUFFER: is the Interface Unit's name for the output "stream" that is read by the Event Log. It would be easy to change this sequence to output to the IU's RS232 or RS485 port if these are available.

Name: Prn$1
Type: Print text
    Comment: Print: Elapsed time # reading
    Message: Elapsed time # reading
    Destination: BUFFER:
Options: Continue:,#=Value:

Prn#1 illustrates the other way of displaying values, using a Print Number event. A Print Number event does not need to have a Set event to fetch the value. It is also possible to change the display format of a Print Number event by setting the *Override format to* property in the editor (called Output Format in the listing).

Prn#1 has the *End:* option set so that the next message is started on a new line in the log.

Name: Prn#1
Type: Print number
    Comment: Print the value of Data1
    Output Format: Source: Data1
    Destination: BUFFER:
Options: End:

The *#=value* option can also be used to pass a value to a Command event. A command event can be used to set a value in a device, a parameter, a scan or even to modify the Event sequence, whereas a Set event can only set a value in a device.

For instance a Command event can be used to modify the averaging used by a Data event. This example is taken from the endpoint sequence for an Ion Milling Probe used as an Endpoint Detector, this instrument has a dummy device called *average* configured in the Global Environment; this fragment from the sequence reads *average* and uses it to modify the Data events.

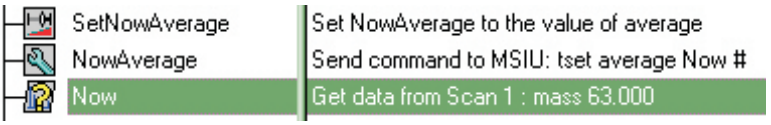

Figure 21 Command event sequence

Name: SetNowAverage
Type: Set event
Comment: Set NowAverage to the value of average
Source Value: average
To: NowAverage
Options:

Name: NowAverage
Type: Command event
Comment: Send command to MSIU: tset average Now #
MSIU Command: tset average Now #
Destination: NUL:
Options: #=Value:

Note

The name of the Data event Now *is embedded in the text of the Command event* NowAverage. *If the name of* Now *is changed it will not be changed automatically in the Command event.*

2.8 The Timer Event

The Timer event can be used as a simple timer or used to take periodic actions. A Timer event's value is set to zero when the event sequence is run and its value returns the time in seconds since it was first run (with the format of the device *timer*). A timer can be reset by using a Set event with the Timer event as its destination. A Timer event can be paused by setting its *enable* property to 0 using a Command event.

The sequence shown in Figure 22 will print a message and stop 60s after the sequence was started.

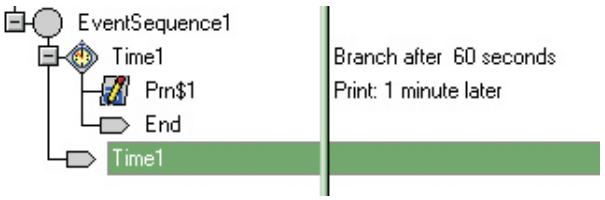

Figure 22 Stop after 60 seconds time event sequence

The Timer event editor was set to After, so the value is entered into the right hand edit box.

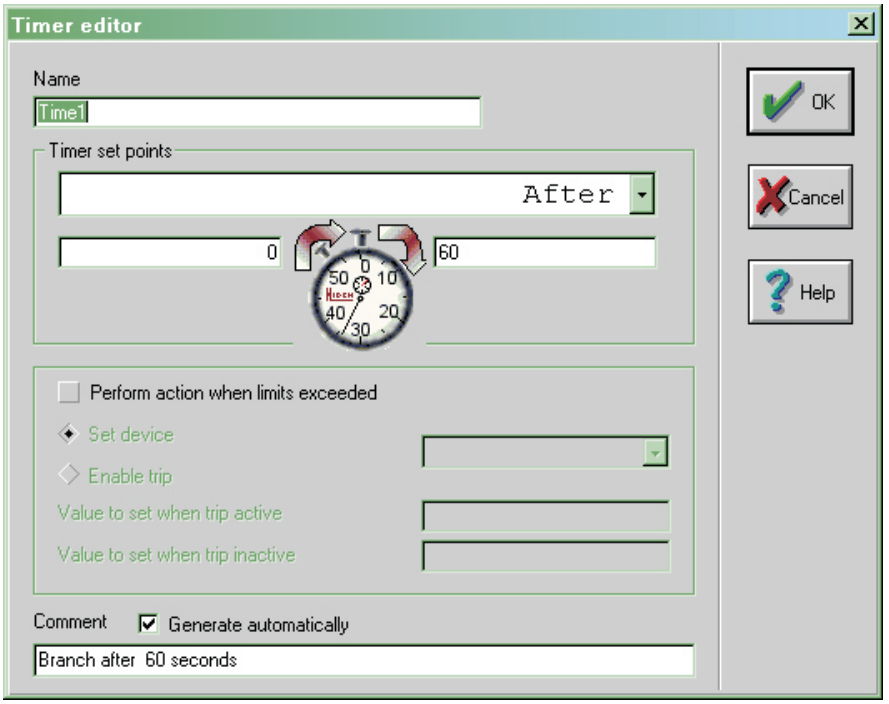

Figure 23 Timer editor dialog box

The actions allowed for a Timer event are exactly the same as a Limit event, it can either branch, set a device or enable another trip or event.

There may be occasions where it is desirable to start a timer at some point in a sequence and then read back the elapsed time at a later point. To do this create a sequence with just timers in it.

Figure 24 Timers event sequence

The timers should be set to Before 0, which will never branch.

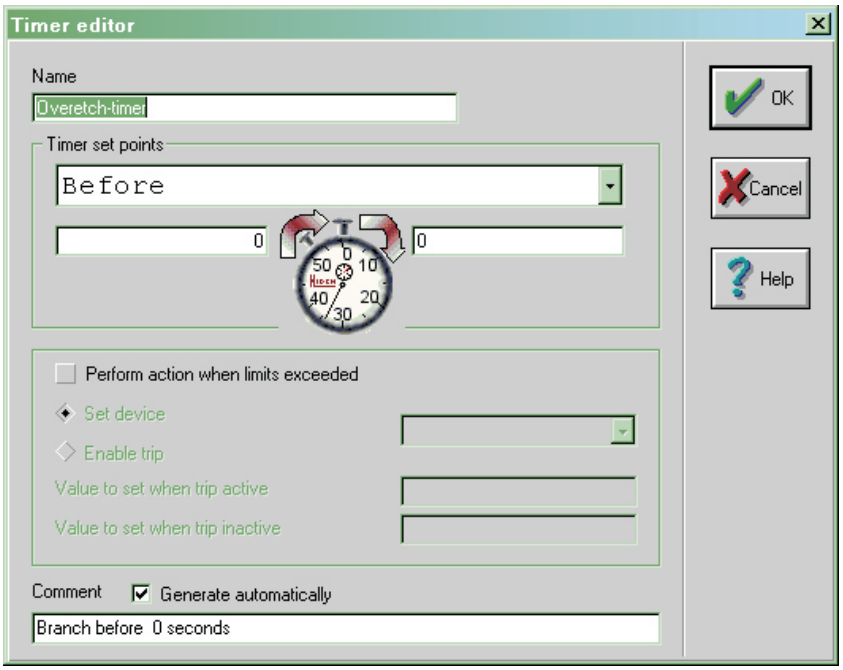

Figure 25 Timer editor, overstretch timer

Run the sequence shown in Figure 24 at the start of the scan to initialise the timers.

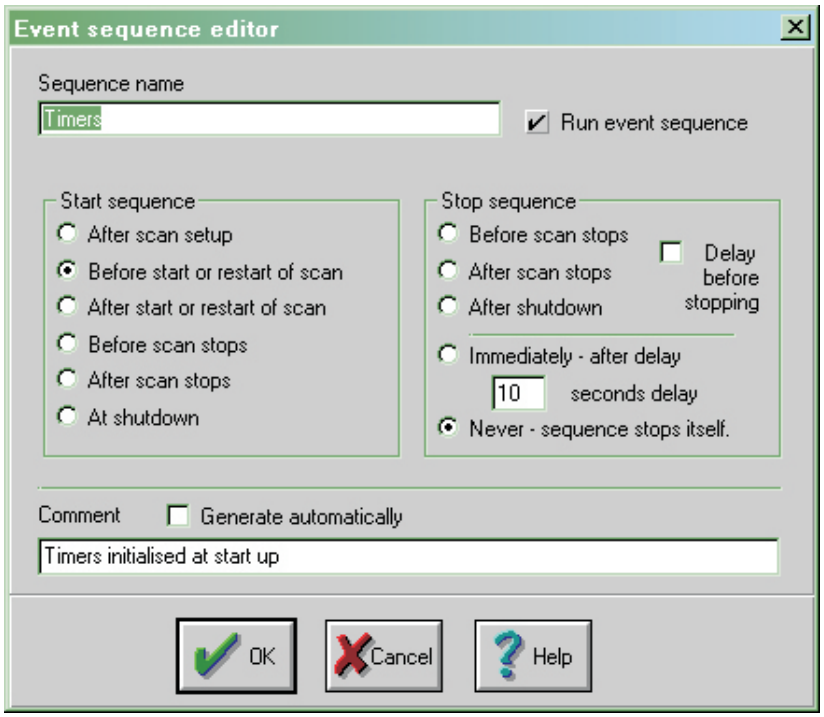

Figure 26 Event sequence editor, Timers

To start a timer use a Set event to set the timer to 0, see Figure 27.

Figure 27 Start Timer event sequence

Later the time can be read as the timer was set using an event that can read an Event's value as a data source; Set, Limit, Evaluate or Print Number.

Figure 28 Timer event sequence

Figure 29 Print event sequence

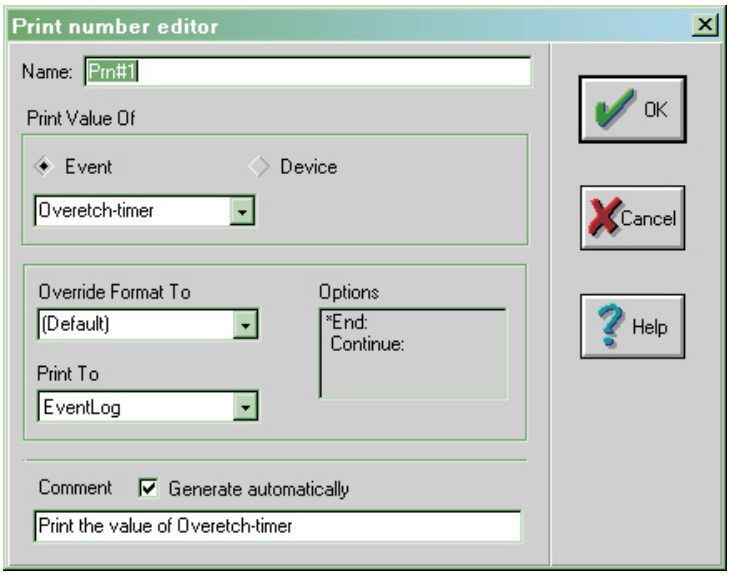

Figure 30 Print number editor

The (Default) format will be the format of the timer device.

A Command event can be used to pause a timer and resume a timer. This event sequence fragment shown in Figure 31 stops the timer for 10s.

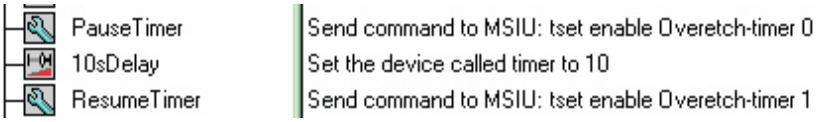

Figure 31 Timer pause event sequence

Note

If the name of the timer is changed it will not be changed automatically in the text of the Command event.

Timers can be used to perform periodic actions.

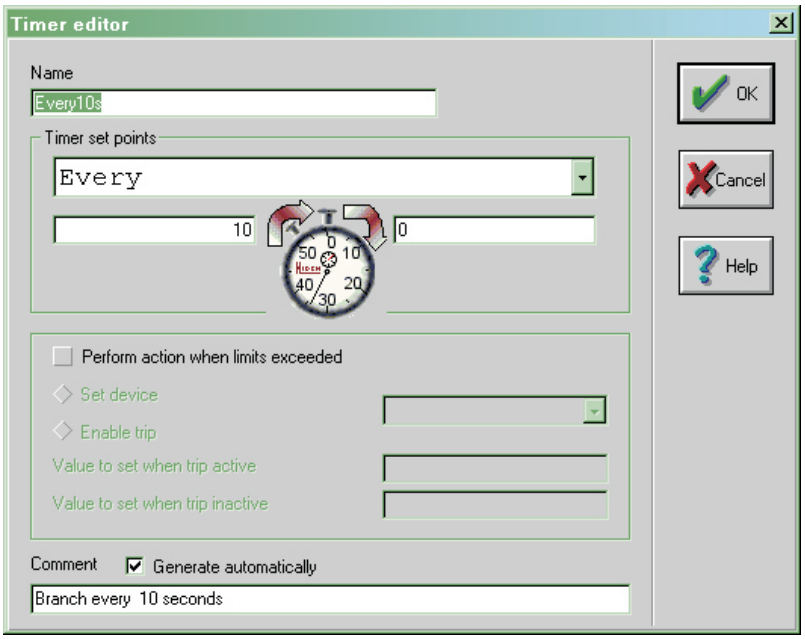

Figure 32 Timer editor, every 10 seconds

The timers should be added to a loop that runs continuously.

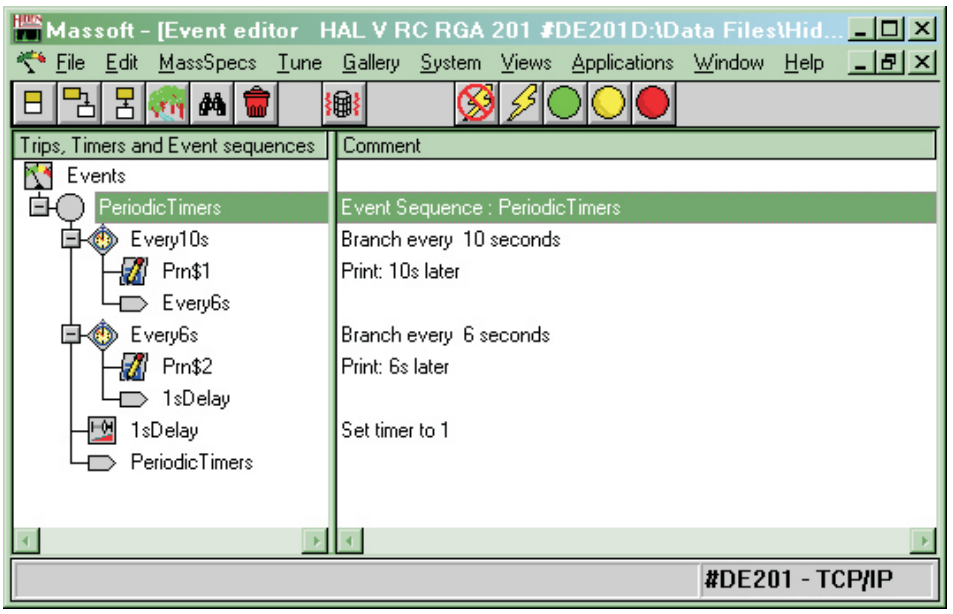

Figure 33 Periodic timers event sequence

Note

After the timer's action it jumps back to execute the next timer in the loop. Timer's actions should be kept short to avoid delaying other timers.

The sequence ends with a 1 second delay to prevent the loop using too much processor time when no timers have expired.

2.8.1 Tips and Suggestions

Use a command event with the command

pget filename

to log the current file name. This is especially useful when used with the *Automatic scan restart* option.

The command

sget cycles

can be used to log the current cycle number.

If an event sequence is used to pause a scan using the standard command

sset state Pause:

use an Evaluate event to read the device m*Secs*. *mSecs* holds the scans elapsed time.

Before restarting the scan using the command

sset state

a Set event can be used to copy the saved value from the Evaluate event back into the device mSecs.

2.9 Device Locking

When a scan is running its input and output devices are locked to prevent inadvertent use. An event sequence is not allowed to set the mass whilst mass is being scanned, because this would affect the value being read by the scan; it would also be pointless because the scan would set the mass again when it moved to the next step.

Likewise, input devices cannot be read when they are being used by the scan; this is to prevent changing the range and affecting the values read by the scan.

Note

Sequences run from a f(x) input or output are exempt from this locking because they run as part of the scan itself.

However, sometimes it is necessary to over-ride this locking, for example when implementing an interlock that must protect the mass spectrometer. To do this an IU command must be used to set the device instead of using a Set event.

The command *L999 <device> <value>* sets a device to a value and over-rides locking. This can be executed by a Command event.

For instance, the enable device is a logical device that functions like a master on/off switch. Setting enable to 0 will force all power supplies into a safe state by setting many individual devices to 0V, but this will cause “Device in use” errors if done while scanning. To avoid that use the command shown in Figure 34.

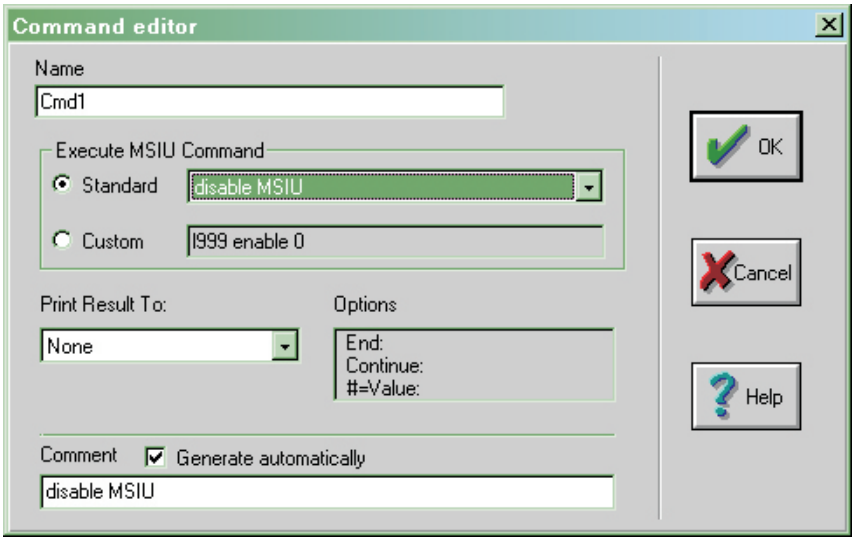

Figure 34 Disable MSIU command

This command was selected from the drop down list of standard commands.

2.10 Trips

2.10.1 Differences between Trips and Event Sequences

There are two ways to check that a value read by a scan has not exceeded a set of limit values:

1.	Use an Intensity Trip.	Tie the trip to the scan to be monitored. Set the limits in the trip. Define the trip's actions.
2.	Write an event sequence.	Use a Data event to fetch a value from the scan. Use a Limit event to check that value has not exceeded the limit values. Define the actions in the Limit event, or use the Limit event to branch to a sequence that performs the required actions. Make the event sequence loop to repeat the test.

Which approach should be used? How do they differ?

The main difference is that Trips are all checked by the Trip task, whereas Event sequences each run as independent programmes.

The Trip task follows behind the data as soon as it is stored in the buffer and checks if any trips apply to it. The Trip task will only check each data point against a trip once.

A Trip only takes an action when it changes state, from inactive to active or from active to inactive. A trip becomes active when its limits are exceeded and becomes inactive when the values return within limits. No action is taken whilst the data remains within the limit, only when it changes from being within the limits to being outside the limit. Likewise no action is taken whilst the data remains outside the limit, only when it changes from being outside the limits to being back within the limit. Trips only execute their actions when they change state from Active to Inactive, or vice versa. This means Trips are fast and will keep up with the speed at which data is acquired.

A Limit event takes an action every time it executes. Each time it executes it compares the data with the limits and takes the specified action. How often it does this depends on the time taken to execute the loop containing the Limit event. Therefore, a Limit event may test the same data point many times if the acquisition speed is slow and the loop is short or it may skip data points if the acquisition speed is fast and the loop is long.

If a Limit event's action is to set a device to 1 when it is active and 0 when it is inactive it will set the device to 1 when its limits are exceeded and keep on setting it to 1 whilst the data exceeds the limit. It will set the device to 0 when it changes from being outside the limits to being back within the limit, and will keep on setting it to 0 whilst it remains within the limits.

2.10.2 Trip Devices

There are three trip devices provided as standard on Hiden instruments: the relays *trip1* and *trip2* and the audible output *beep.*

These devices are specially designed to be used with Trips. They allow one output to be used by more than one Trip. Consider the following scenario to see why this would normally be a problem:

Two trips, Int1 and Int2, both use IO1 as an output. IO1 is initially 0. Int1 becomes active and sets IO1 to 1 (1 = on). Then Int2 becomes active and sets IO1 to 1 also. Next Int2 becomes inactive and sets IO1 to 0 (0 = off) , but Int1 is still active so IO1 should still be 1 (on) .

Trip devices overcome this problem by keeping a “reference count” of the number of times they have been set. Setting a trip device to 1 increases its reference count by 1, setting it to -1 decreases its reference count by 1. When its reference count is greater than 0 its output is on. Now consider the same scenario again:

Two trips, Int1 and Int2, both use *beep* as an output. *beep* is initially off and its reference count is 0. Int1 becomes active and sets *beep* to 1 so its reference count increases to 1 therefore its output is on. Then Int2 becomes active and sets *beep* to 1 also, this increases the reference count to 2, the output remains on. Next Int2 becomes inactive and sets *beep* to -1, this decreases the reference count to 1, but this time the output remains on. Finally, Int1 becomes inactive and sets *beep* to -1, this decreases the reference count to 0, so the output goes off.

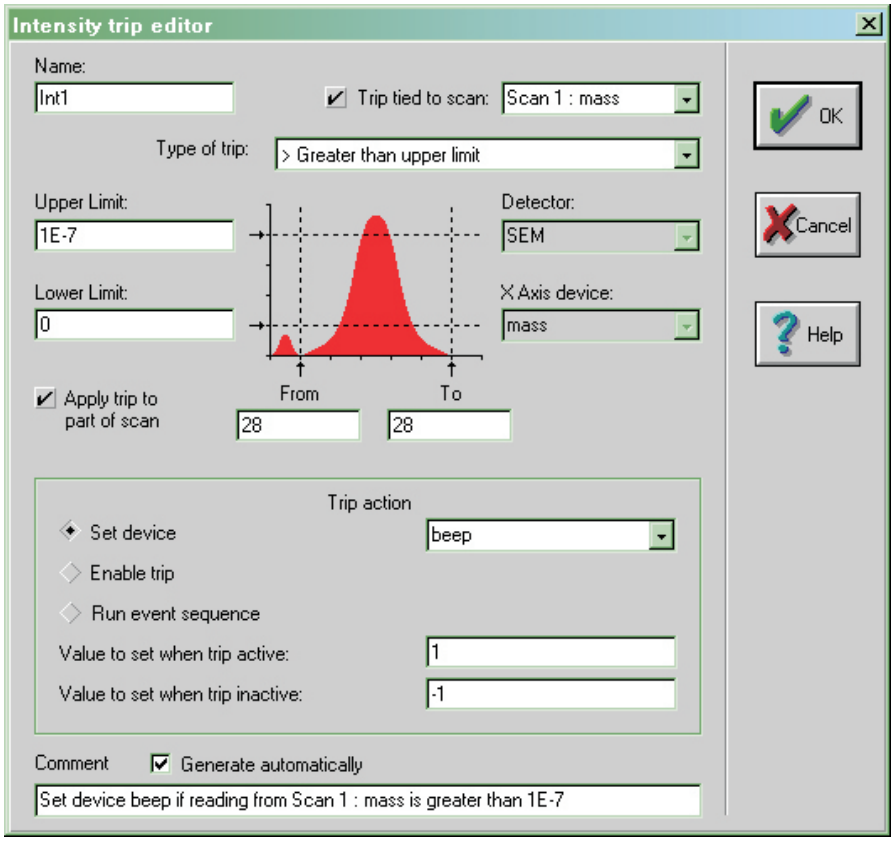

Figure 35 Using a Trip Device in an Intensity Trip

This technique only works with Trips because Trips only set their outputs when they change state from active to inactive, or vice versa.

A Limit event would increment (or decrement) the reference count each time it executed, i.e. on every loop. If a trip device is used in an Event Sequence it cannot be shared with a Trip. Logic must be incorporated in the code to turn the device on or off as required.

Trip devices have a third legal value, 0. Setting a trip device to 0 turns the device off and sets the reference count to 0. When used in an event sequence 1 should be used to turn a trip device on and 0 used to turn it off

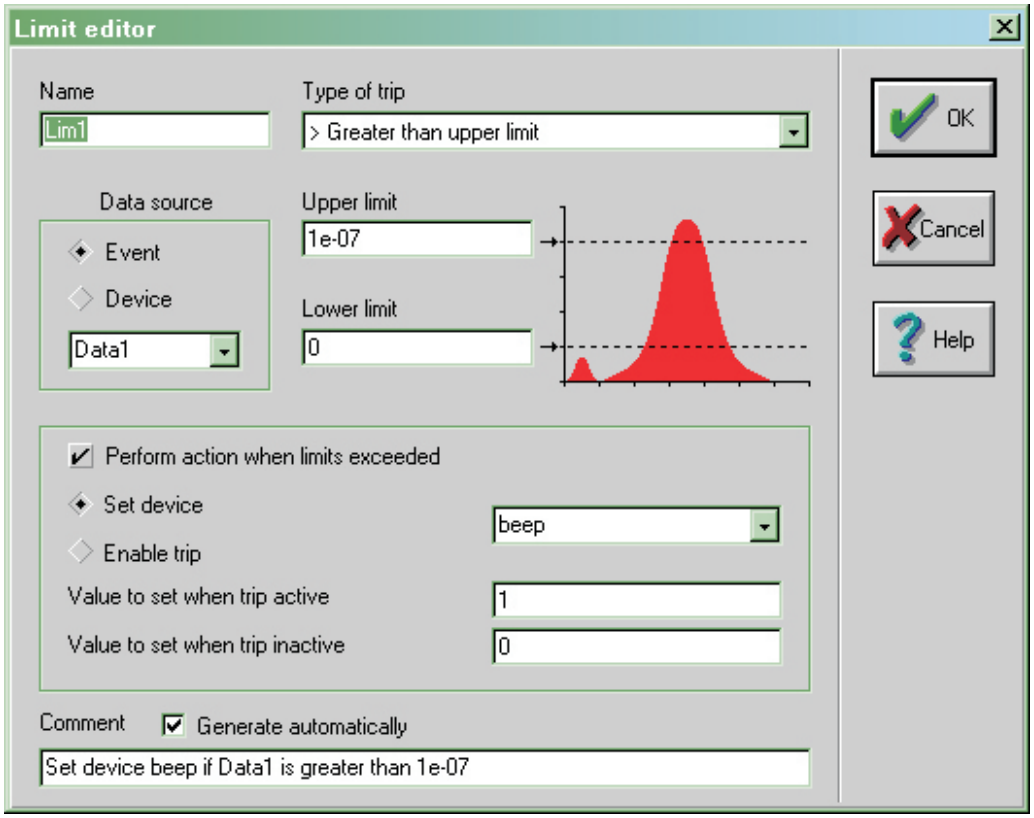

Figure 36 Using a Trip Device in a Limit Event

3 Examples

The files for the following examples are provided on the Hiden Software Suite CD.

The files were created on a system with R5.6 IU Firmware. Not all of the illustrated features will be available in instruments with earlier versions of firmware. The example files will generate errors if downloaded to instruments with earlier versions of firmware.

Firmware version R5.6 added the Options property to many event types. The *Options* list in the editor displays the available options; with firmware prior to R5.6 the *Options* list may be empty in some event types.

R5.6 also added the Timer event. If the Timer event is not available it will not be displayed in the Edit dialog box, nor in the pop-up displayed by right-clicking on an Event Sequence.

The example files were created on a HAL IV RGA 201 system. This is an RGA (Residual Gas Analyser) system with a mass range of 200 amu with analogue Faraday and SEM detectors. The system was configured to have 16 channels of analogue output.

The example files should only be run directly on systems that exactly match the above specification otherwise, “Unknown logical device” and “Device value out of range” errors can be expected during scanning.

Where systems are not an exact match use **File, New** to create a new file then duplicate the scan tree of the example file. Open the example file, open the Event Editor and select the entire Event Sequence tree. Copy the event sequences from the example file and paste them into the Event Editor in the new file.

3.1 Writing a message to the Event Log

This is the classic “Hello world” programme which introduces almost every book on almost all programming languages.

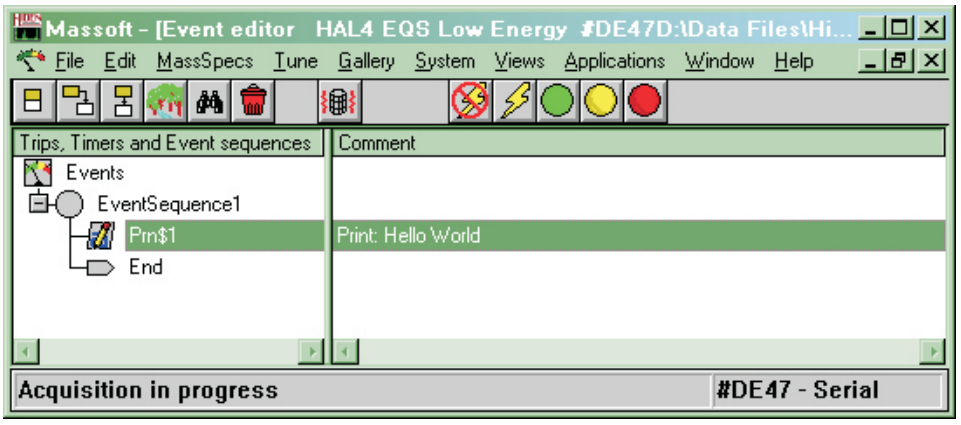

Figure 37 Hello world event sequence

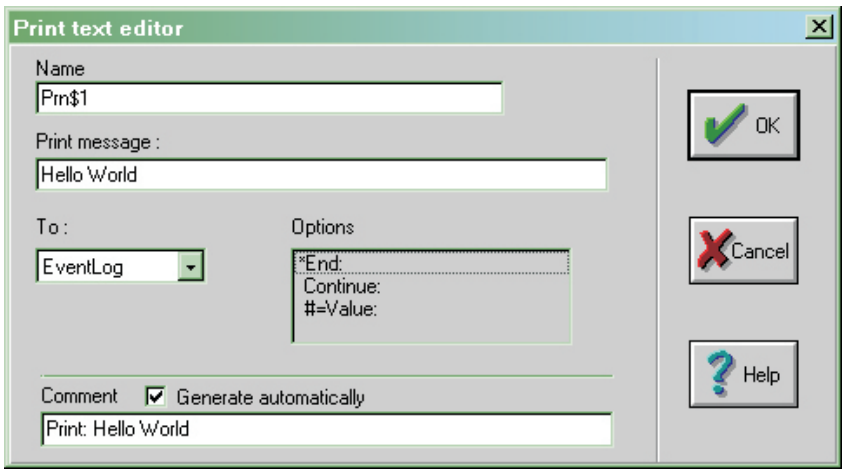

Figure 38 Print text editor

The To: property instructs Prn$1 to print ***Hello World*** to the Event Log. If the PC is connected to the instrument via a network the output could be the IU's serial port, COM2 (COM1: is the IU's RS485 port).

The *End option tells Prn$1 to send a newline character after the message. If this were omitted MASsoft would sit waiting for the end of the message to be sent – effectively hanging MASsoft. If the End option is not selected then the Print text event must be followed by another Print Text or Print Number event that does send a newline; this allows several print events to be built into a longer message line.

NOTE

For versions of firmware that do not have the End option the last character of a message must be . (full stop), :(colon), ? (question mark) or ! (exclamation mark) to ensure a newline is sent.

Double click on the Event Sequence label EventSequence1 to display the Event sequence editor dialog box, shown in Figure 39.

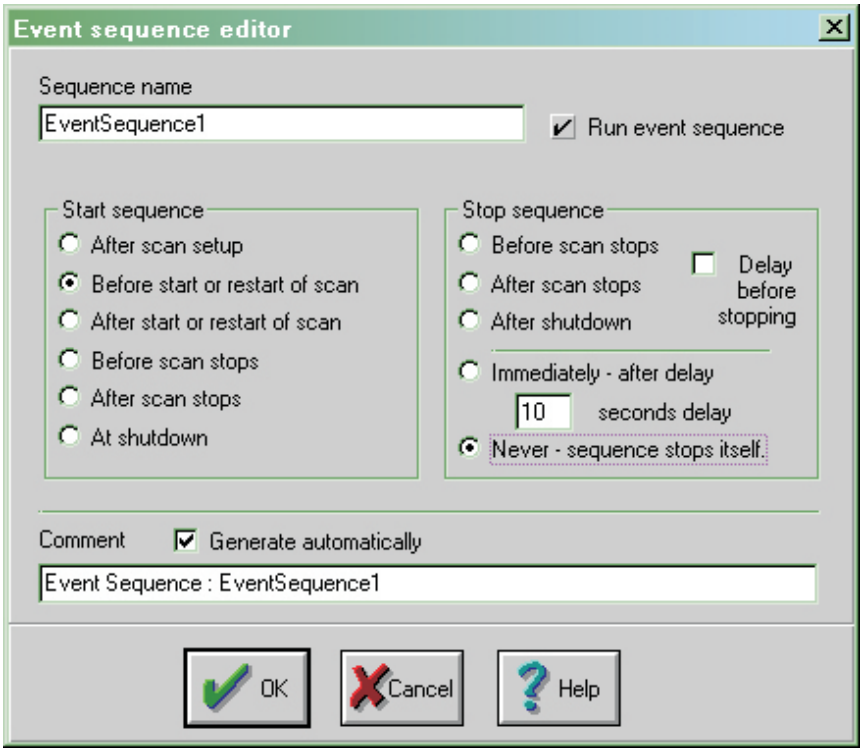

Figure 39 Event sequence editor, EventSequence1

The *Run event sequence* check box must be ticked for the Event sequence to run. Event sequence labels should be used to divide the event sequence into logical sections, so not all sections will be started automatically when the scan starts.

The *Start sequence* radio buttons give a choice as to when the sequence starts. The easiest way to see what each option does is to try them. To appreciate them fully set the file to run for a fixed number of cycles and select the *Automatic scan restart* option.

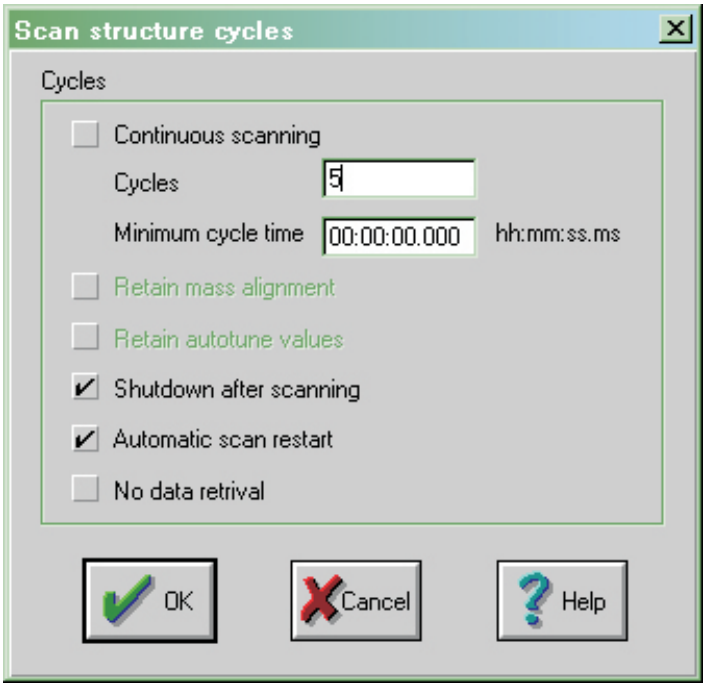

Figure 40 Scan structure cycles dialog box

NOTE

For At Shutdown *to work the file must still be open when the instrument is switched to shutdown. If the file is closed before switching to shutdown the sequence will not run.*

The *Stop sequence* radio button should be set to *Never – sequence stops itself* because this sequence just outputs one message and then ends.

To see the affect of the Stop sequence options change the sequence slightly. Add a Set event to add a delay and then make the sequence loop back to the beginning.

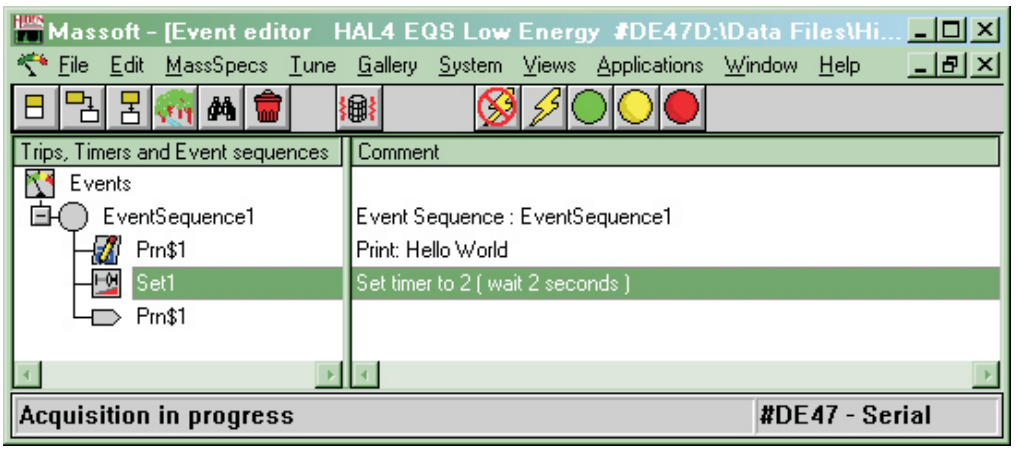

Figure 41 Event sequence, wait

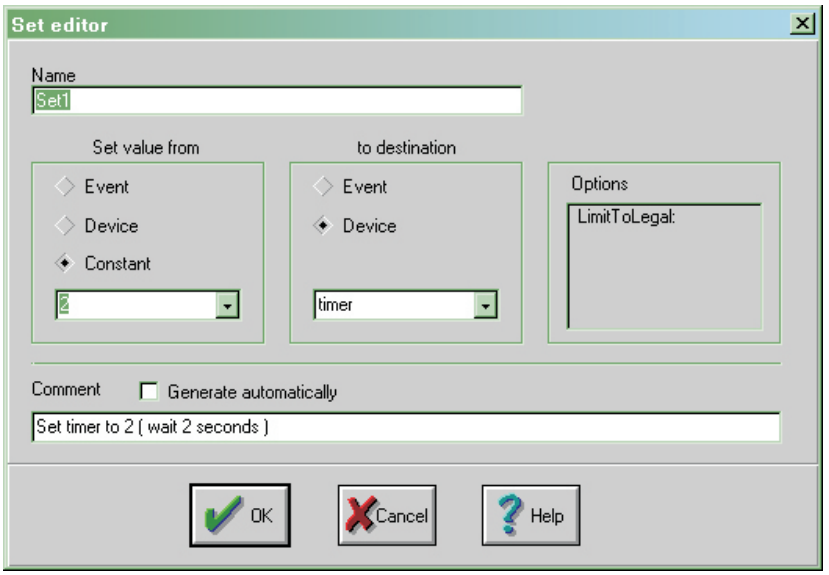

Figure 42 Set editor, wait 2 seconds

Setting the timer device to 2 makes the sequence delay for 2 seconds, the comment has been edited to explain this. There is a similar device called delay, which waits for milliseconds – setting timer to 2 or delay to 2000 are equivalent.

The delay is necessary to prevent the Event Sequence generating messages faster than MASsoft can read them.

Try this option for starting and ending the run:

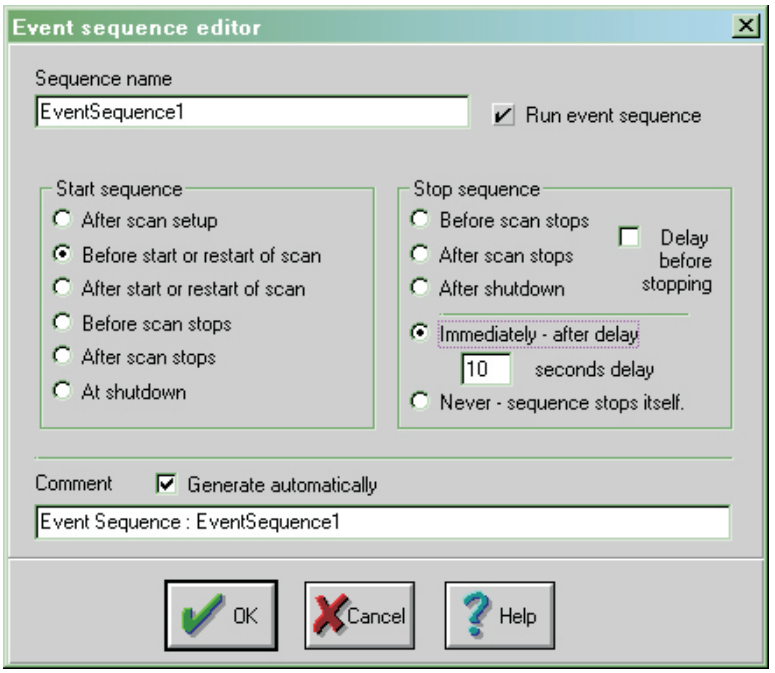

Figure 43 Event sequence 1, event editor

The event sequence will run before the start of the run – that is, after the scan has been setup but before the command to start scanning has been sent from MASsoft to the Interface Unit.

The option to stop the sequence is set to “Immediately – after 10s delay”. MASsoft will wait up to 10s for the event sequence to stop itself, if after 10s the sequence is still running then MASsoft will stop it. So, in this example the event sequence will output 5 or 6 messages and then be stopped by MASsoft.

Only after the event sequence has stopped will MASsoft start the scan.

The “immediately – after delay” option provides a way to start, run, and stop a sequence at the point specified by the “Start sequence” option; while this is happening MASsoft waits. The “immediately – after delay” option is always used with a delay, there would be no point in stopping the sequence without giving it some time to do something; it is up to the user to ensure the delay is long enough.

For an explanation of the other start and stop options please see the reference section of this manual.

3.1.1 Turning F2 on if F1 fails

This event sequence is part of the experiment file checkfilament.exp.

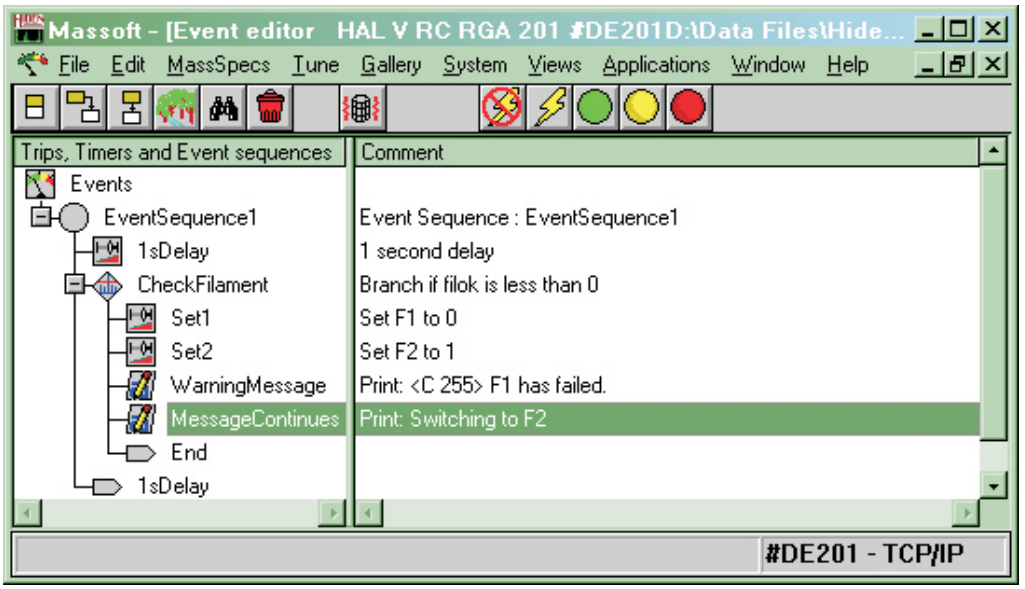

Figure 44 Check filament event sequence

Sequence : EventSequence1

Name: 1sDelay
Type: Set event
    Comment: 1 second delay
    Source Value: 1
    To: timer
Options:

Name: CheckFilament
Type: Limit event
Comment: Branch if filok is less than 0
Trip Logic: <
Device Source: filok
Upper Limit: 0
Lower Limit: 0
Branch to: Set1

Name: Set1
Type: Set event
Comment: Set F1 to 0
Source Value: 0
To: F1
Options:

Name: Set2
Type: Set event

Comment: Set F2 to 1
Source Value: 1
To: F2Options:

Name: WarningMessage
Type: Print text
Comment: Print:<C 255> F1 has failed.
Message:<C 255> F1 has failed.
Destination: BUFFER:
Options: Continue:

Name: MessageContinues
Type: Print text
Comment: Print: Switching to F2
Message: Switching to F2
Destination: BUFFER:
Options: End:

3.1.2 Using Command Events to configure auxiliary inputs

This sequence makes it easier to set up the auxiliary inputs.
Copy and paste this sequence into a new file then edit the steps labelled 1/ to 7/. Make sure the # after the device name in step 7/ is not removed.
The Constants depend on the version of the instrument.

V to F Slope	Intercept	Instrument Version
-90909.1	1000000	Some 6U and 7U Interface Units supplied prior to 1996 fitted with Analyser Control board revisions a to e. Contact Hiden Analytical for advice.
-90497.7	1000000	7U Interface Units fitted with Analyser Control board revision f or later with HA-061-408 PLD.
-45248.8	500000	7U IUs fitted with Analyser Control board revisions f or later with HA-061-410, HA-061-411, HA-061-412 or HA-061-413 PLD. These are systems with the “chopper” PLD with foreground and background gating inputs (though separate foreground and background inputs are not always configured). These systems have gating-invert or foreground-invert in the Global Environment.
-90497.7	1000000	All 2U HAL IV RC Interface units to date (7/1/2005) (all RGAs and SIMS systems with 2U SIMS interface units) fitted with HA-061-404 PLD.

Table 2 Slope and intercept values

If in any doubt please feel free to contact Hiden Analytical for assistance.

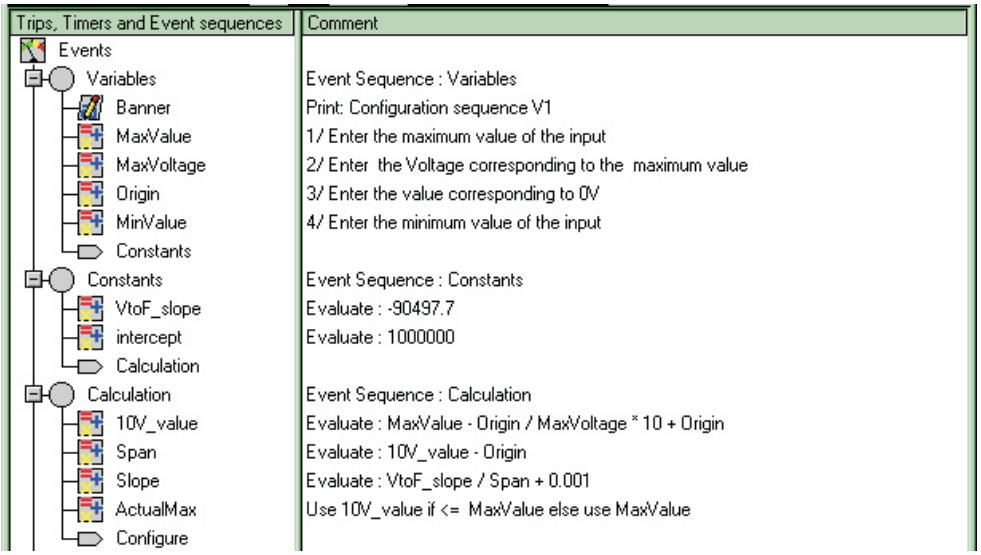

Figure 45 Command event, event sequence (1)

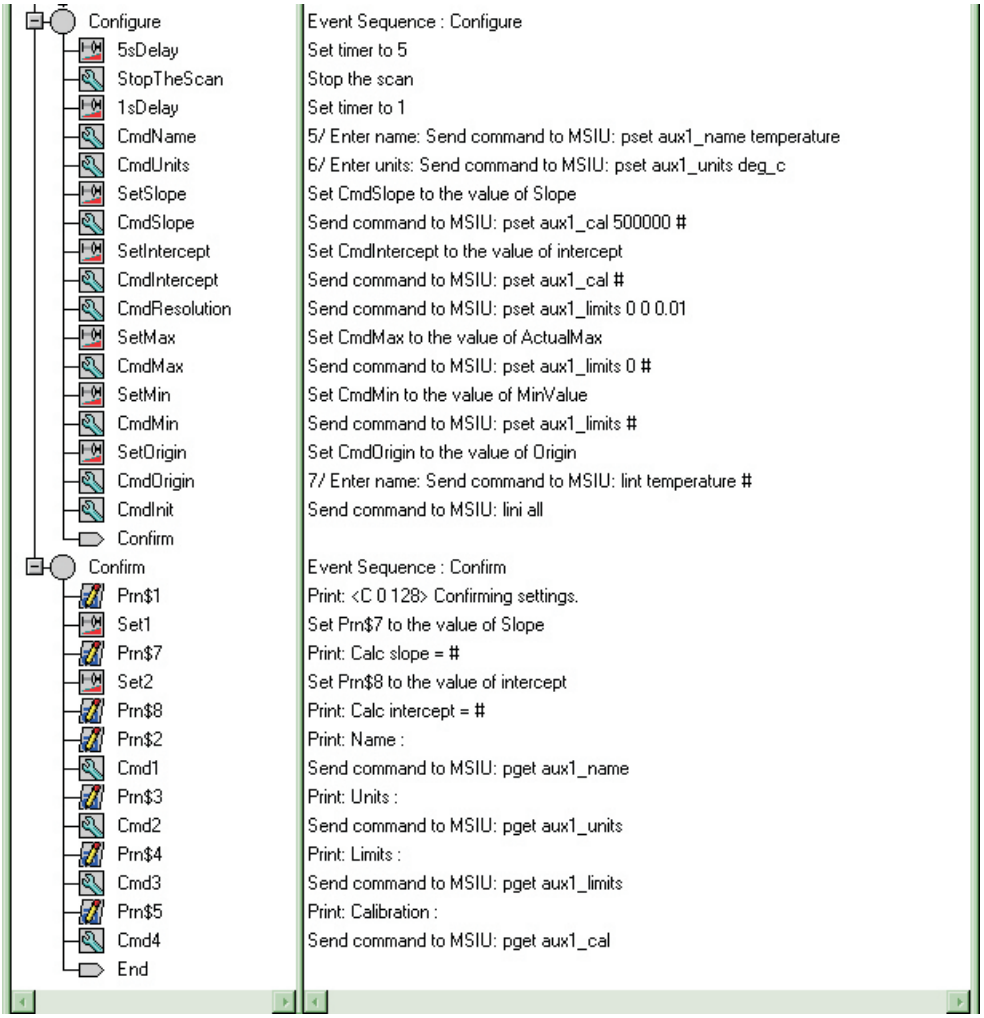

Figure 46 Command event, event sequence (2)

This sequence has been listed in full because the setting of options is crucial to the correct operation of this sequence.

This sequence is broken down into five logical sections. Variables holds the values entered by the user. Constants hold values which depend on the instrument. Calculate derives values from the entries in Variables and Constants. Configure actually sets up the auxiliary input. Confirm checks that the values were actually set and writes this to the Event log.

Breaking down the code into sections makes it more readable; long sections should be avoided because they can cause MASsoft to run out of memory when performing some operations.

This sequence makes use of the *#=value* option in Command events. The format in which the number is sent depends on the format of the source event. This is often an Evaluate event so the *Override format as* setting of some of these Evaluate events is crucial. This is especially true of the Intercept and Slope Evaluate events which are passed to the *pset aux1_cal* command. *pset aux1_cal* will not accept numbers in exponential format, so a

device whose format displays to 3 decimal places and that allows +ve and –ve values must be found; the device *timer* meets this requirement.

Note

The format specifiers 1 to 9 (or Ascans to IScans) should only be used in Print Number events; their use in other types of event will result in the value being incorrect by factors of ten.

This sequence stops the scan using the standard command sset state Abort. This command is available in the drop down list of commands in the Command event editor. The step StopTheScan stops the scan to avoid "Device in use" errors when issuing the *lini all* command later in the sequence.

The Event sequence editor for Variables has the *Delay before stopping* option set to ensure that the sequence has run to completion before MASsoft tries to stop it.

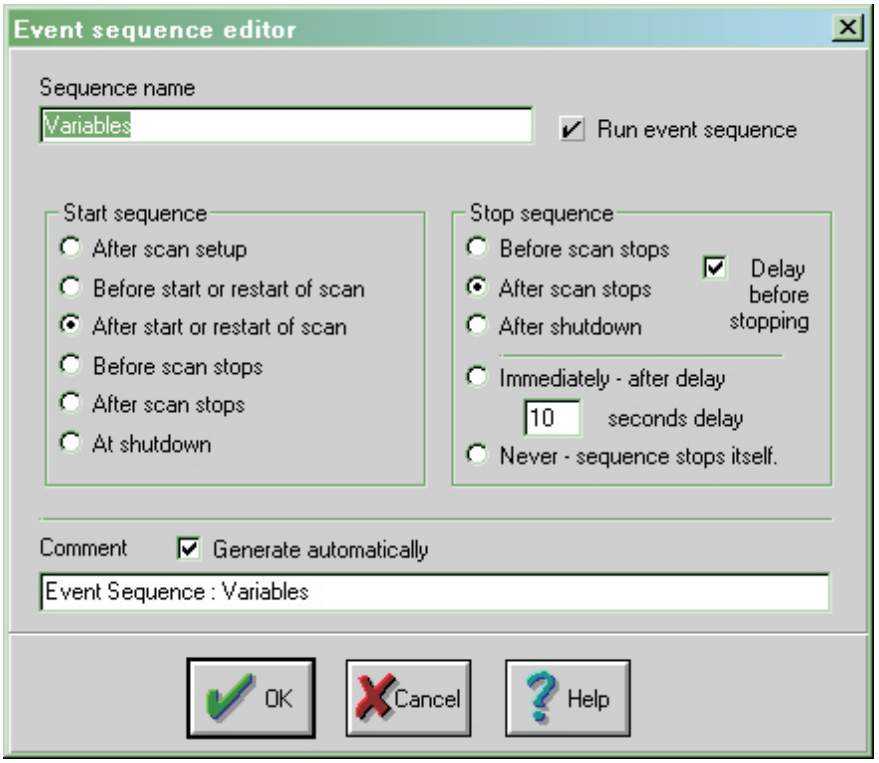

Figure 47 Event sequence editor, variables

This is the only sequence with *Run event sequence* checked, the other sequences are chained to this.

Sequence : Variables
Name: Banner
Type: Print text
Comment: Print: Configuration sequence V1
Message: Configuration sequence V1
Destination: BUFFER:

Options: End:
Name: MaxValue
Type: Evaluate event
Comment: 1/ Enter the maximum value of the input
Output Format: f(x)
Source 0: 1400
Operator 1:
Source 1:
Operator 2:
Source 2:
Operator 3:
Source 3:
Operator 4:
Source 4:
Name: MaxVoltage
Type: Evaluate event
Comment: 2/ Enter the Voltage corresponding to the maximum value
Output Format: f(x)
Source 0: 5
Operator 1:
Source 1:
Operator 2:
Source 2:
Operator 3:
Source 3:
Operator 4:
Source 4:
Name: Origin
Type: Evaluate event
Comment: 3/ Enter the value corresponding to 0V
Output Format: f(x)
Source 0: -100
Operator 1:
Source 1:
Operator 2:
Source 2:
Operator 3:
Source 3:
Operator 4:
Source 4:
Name: MinValue
Type: Evaluate event
Comment: 4/ Enter the minimum value of the input
Output Format: f(x)
Source 0: -100
Operator 1:
Source 1:
Operator 2:
Source 2:
Operator 3:
Source 3:
Operator 4:

Source 4:
Sequence : Constants
Name: VtoF_slope
Type: Evaluate event
Comment: Evaluate : -90497.7
Output Format: timer
Source 0: -90497.7
Operator 1:
Source 1:
Operator 2:
Source 2:
Operator 3:
Source 3:
Operator 4:
Source 4:
Name: intercept
Type: Evaluate event
Comment: Evaluate : 1000000
Output Format: timer
Source 0: 1000000
Operator 1:
Source 1:
Operator 2:
Source 2:
Operator 3:
Source 3:
Operator 4:
Source 4:
Sequence : Calculation
Name: 10V_value
Type: Evaluate event
Comment: Evaluate : MaxValue - Origin / MaxVoltage * 10 + Origin
Output Format: f(x)
Source 0: MaxValue
Operator 1: -
Source 1: Origin
Operator 2: /
Source 2: MaxVoltage
Operator 3: *
Source 3: 10
Operator 4: +
Source 4: Origin
Name: Span
Type: Evaluate event
Comment: Evaluate : 10V_value - Origin
Output Format: 3
Source 0: 10V_value
Operator 1: -
Source 1: Origin
Operator 2:
Source 2:
Operator 3:

Source 3:
Operator 4:
Source 4:
Name: Slope
Type: Evaluate event
Comment: Evaluate : VtoF_slope / Span + 0.001
Output Format: timer
Source 0: VtoF_slope
Operator 1: /
Source 1: Span
Operator 2: +
Source 2: 0.001
Operator 3:
Source 3:
Operator 4:
Source 4:
Name: ActualMax
Type: Evaluate event
Comment: Use 10V_value if <= MaxValue else use MaxValue
Output Format: f(x)
Source 0: 10V_value
Operator 1: Max
Source 1: MaxValue
Operator 2:
Source 2:
Operator 3:
Source 3:
Operator 4:
Source 4:
Sequence : Configure
Name: 5sDelay
Type: Set event
Comment: Set timer to 5
Source Value: 5
To: timer
Options:
Name: StopTheScan
Type: Command event
Comment: Stop the scan
MSIU Command: sset state Abort:
Destination: NUL:
Options:
Name: 1sDelay
Type: Set event
Comment: Set timer to 1
Source Value: 1
To: timer
Options:
Name: CmdName
Type: Command event
Comment: 5/ Enter name: Send command to MSIU: pset aux1_name temperature
MSIU Command: pset aux1_name temperature

Destination: NUL:
Options:
Name: CmdUnits
Type: Command event
Comment: 6/ Enter units: Send command to MSIU: pset aux1_units deg_c
MSIU Command: pset aux1_units deg_c
Destination: NUL:
Options:
Name: SetSlope
Type: Set event
Comment: Set CmdSlope to the value of Slope
Source Value: Slope
To: CmdSlope
Options:
Name: CmdSlope
Type: Command event
Comment: Send command to MSIU: pset aux1_cal 500000 #
MSIU Command: pset aux1_cal 500000 #
Destination: NUL:
Options: #=Value:
Name: SetIntercept
Type: Set event
Comment: Set CmdIntercept to the value of intercept
Source Value: intercept
To: CmdIntercept
Options:
Name: CmdIntercept
Type: Command event
Comment: Send command to MSIU: pset aux1_cal #
MSIU Command: pset aux1_cal #
Destination: NUL:
Options: #=Value:
Name: CmdResolution
Type: Command event
Comment: Send command to MSIU: pset aux1_limits 0 0 0.01
MSIU Command: pset aux1_limits 0 0 0.01
Destination: NUL:
Options:
Name: SetMax
Type: Set event
Comment: Set CmdMax to the value of ActualMax
Source Value: ActualMax
To: CmdMax
Options:
Name: CmdMax
Type: Command event
Comment: Send command to MSIU: pset aux1_limits 0 #
MSIU Command: pset aux1_limits 0 #
Destination: NUL:
Options: #=Value:
Name: SetMin
Type: Set event

Comment: Set CmdMin to the value of MinValue
Source Value: MinValue
To: CmdMin
Options:
Name: CmdMin
Type: Command event
Comment: Send command to MSIU: pset aux1_limits #
MSIU Command: pset aux1_limits #
Destination: NUL:
Options: #=Value:
Name: SetOrigin
Type: Set event
Comment: Set CmdOrigin to the value of Origin
Source Value: Origin
To: CmdOrigin
Options:
Name: CmdOrigin
Type: Command event
Comment: 7/ Enter name: Send command to MSIU: lint temperature #
MSIU Command: lint temperature #
Destination: NUL:
Options: #=Value:
Name: CmdInit
Type: Command event
Comment: Send command to MSIU: lini all
MSIU Command: lini all
Destination: NUL:
Options:
Sequence : Confirm
Name: Prn$1
Type: Print text
Comment: Print: <C 0 128> Confirming settings.
Message: <C 0 128> Confirming settings.
Destination: BUFFER:
Options: End:
Name: Set1
Type: Set event
Comment: Set Prn$7 to the value of Slope
Source Value: Slope
To: Prn$7
Options:
Name: Prn$7
Type: Print text
Comment: Print: Calc slope = #
Message: Calc slope = #
Destination: BUFFER:
Options: #=Value:,End:
Name: Set2
Type: Set event
Comment: Set Prn$8 to the value of intercept
Source Value: intercept
To: Prn$8

Options:
Name: Prn$8
Type: Print text
    Comment: Print: Calc intercept = #
    Message: Calc intercept =#
    Destination: BUFFER:
Options: #=Value:,End:
Name: Prn$2
Type: Print text
    Comment: Print: Name :
    Message: Name :
    Destination: BUFFER:
Options: Continue:
Name: Cmd1
Type: Command event
    Comment: Send command to MSIU: pget aux1_name
    MSIU Command: pget aux1_name
    Destination: BUFFER:
Options: End:
Name: Prn$3
Type: Print text
    Comment: Print: Units :
    Message: Units :
    Destination: BUFFER:
Options: Continue:
Name: Cmd2
Type: Command event
    Comment: Send command to MSIU: pget aux1_units
    MSIU Command: pget aux1_units
    Destination: BUFFER:
Options: End:
Name: Prn$4
Type: Print text
    Comment: Print: Limits :
    Message: Limits :
    Destination: BUFFER:
Options: Continue:
Name: Cmd3
Type: Command event
    Comment: Send command to MSIU: pget aux1_limits
    MSIU Command: pget aux1_limits
    Destination: BUFFER:
Options: End:
Name: Prn$5
Type: Print text
    Comment: Print: Calibration :
    Message: Calibration :
    Destination: BUFFER:
Options: Continue:
Name: Cmd4
Type: Command event
    Comment: Send command to MSIU: pget aux1_cal

MSIU Command: pget aux1_cal
Destination: BUFFER:
Options: End:

3.2 Calculating concentrations

In this example f(x) inputs are used to calculate percentage (%) and part per million (ppm) concentrations of argon and helium in air. The example can successfully be run on any atmospheric sampling mass spectrometer. The experiment file is available from Hiden Analytical and may be supplied on the System Information floppy disk or Hiden USB Flash Drive. The file may be copied and modified to use for other experiments.

The experiment file can be created as follows:

1. On the **File** menu click **New**.

   See Figure 48.

   A new scan tree will be displayed as shown in Figure 49.

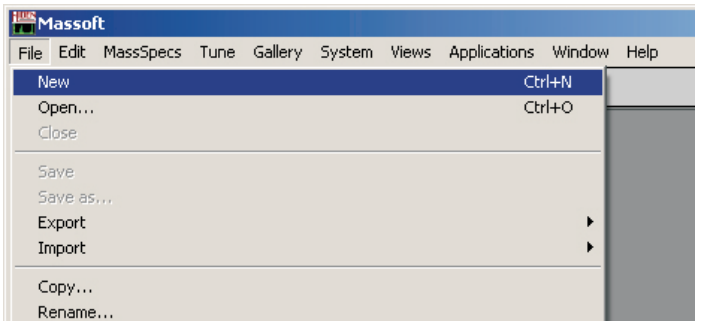

Figure 48 Select File, New

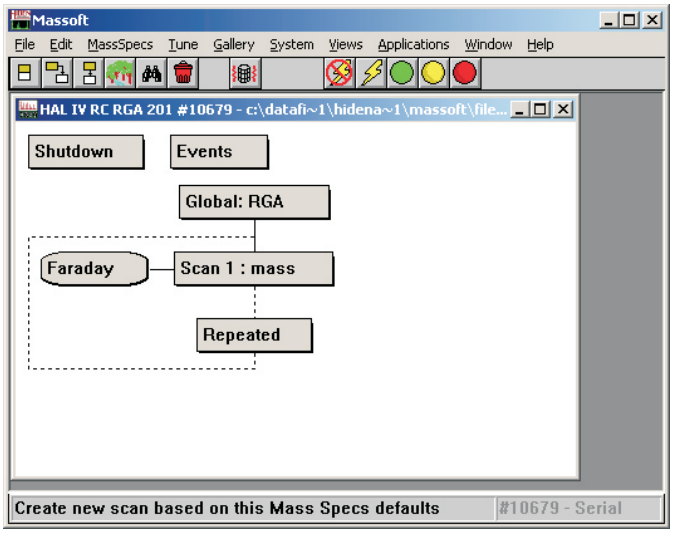

Figure 49 New scan tree

2. In the scan tree double click the **Global: RGA** box.

   The **Environment Editor** dialog box will be displayed as shown in Figure 50.

3. Click one of the Filament buttons. This will cause the filament to be switched on when the scan is run.

Other global settings such as emission and electron energy could be set at this stage but the default values are used in this experiment.

One of the most common causes for a lack of spectra is the failure to switch on a filament.

4. Click the **OK** button. The **Environment Editor** dialog box will be closed.

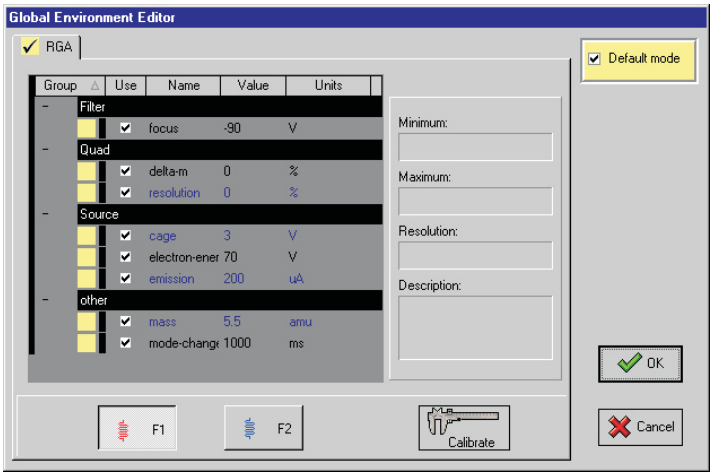

Figure 50 Environment editor dialog box

To calculate the concentration of argon in air the mass spectrometer must measure the partial pressure of argon and the partial pressure of another air gas from which the total pressure (of air) will be calculated. The argon measurement can then be displayed as a percentage concentration of air. The same will be done using the partial pressure measurement of helium. Nitrogen will be used to determine the total pressure. Therefore, three partial pressure measurements will be made by the MASsoft experiment.

As the helium signal will be quite small (helium is 5ppm in air) the electron multiplier detector will be used. It is good practise to use the same detector for all the measurements as this removes any discrepancies between detectors and time delays due to switching between detectors. Usually, the peak at mass 28 (due to N_2^{14+}) would be used to measure the partial pressure of nitrogen but this signal will be very large and if used would saturate the multiplier detector. The mass 29 peak (due to $N^{14}N^{15}+$) will be used instead. It is assumed that the mass 29 peak is entirely due to nitrogen, which may not always be the case.

5. In the scan tree double click the **Scan 1: mass** box. The **Scan Editor** dialog box will be displayed, see Figure 51.

6. In the **Scan Legend** text box type ***He.***

   In both the **Start Value:** and S**top Value:** boxes type ***4***.

   He is a suitable legend for a scan to determine the partial pressure of helium from the mass 4 peak it will produce in the mass spectrum.

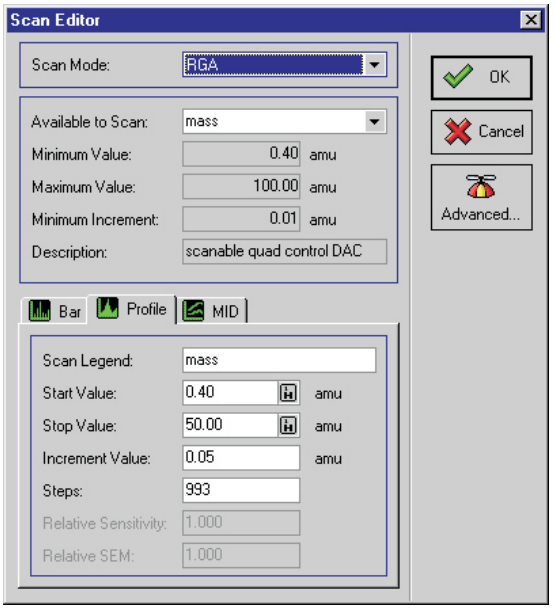

Figure 51 Scan Editor dialog for the helium scan

7. Click the **Advanced** button.

   The dialog box shown in Figure 52 is displayed.

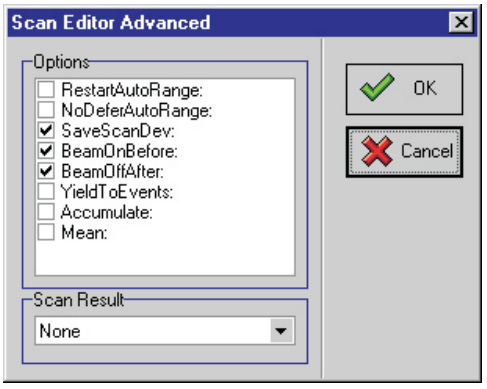

Figure 52 Scan advanced dialog box

8.	In the **Options** list double click **NoDeferAutoRange** to select it	A check mark will be displayed in front of **NoDeferAutoRange** when it is selected.
9.	Click the **OK** button.	The **Scan advanced** dialog will be closed.
10.	In the **Scan Editor** dialog box click the **OK** button.	The **Scan Editor** dialog box will be closed.
11.	In the scan tree click on the box that now has the legend **Scan 1: He 4.000.**	The legend in the box will be highlighted to show it is selected.
12.	Either press the ***Insert*** key on the PC keyboard or on the **Edit** menu click **Insert new Sequence** or click the **Next scan** tool on the tool bar.	All three methods insert a new scan after the first (mass 4) scan in the scan tree.
13.	Repeat the insert action to add another scan box.	The scan tree will now look like the one in Figure 53.

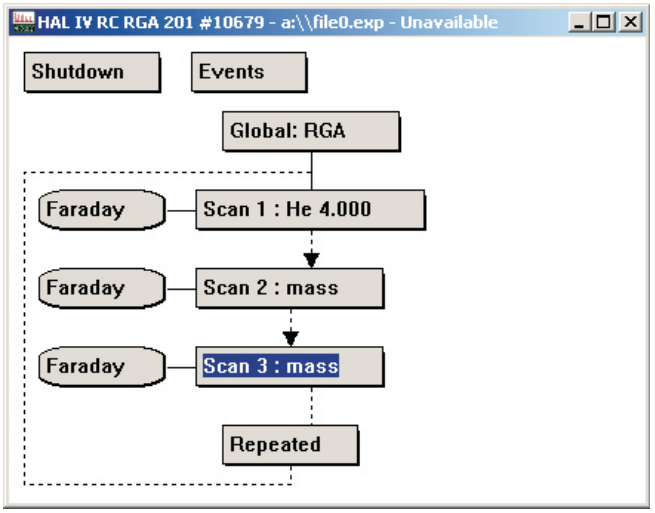

Figure 53 Three scans in the scan tree

14.	Double click the **Scan 2 : mass** box.	The **Scan Editor** dialog box will be opened.
15.	In the **Scan Legend** text box type ***N2_at_29.*** In both the **Start Value:** and S**top Value:** boxes type ***29***.	

16. Click the **Advanced** button and double click **NoDeferAutoRange** to enable it.

17. Close the **Scan advanced** and **Scan Editor** dialog boxes.

18. Repeat steps 14 to 17 for the **Scan 3 : mass** box setting **Scan Legend** to Ar and **Start Valve:** and **Stop Value:** to 40.

The scan will now look like the one shown in Figure 54.

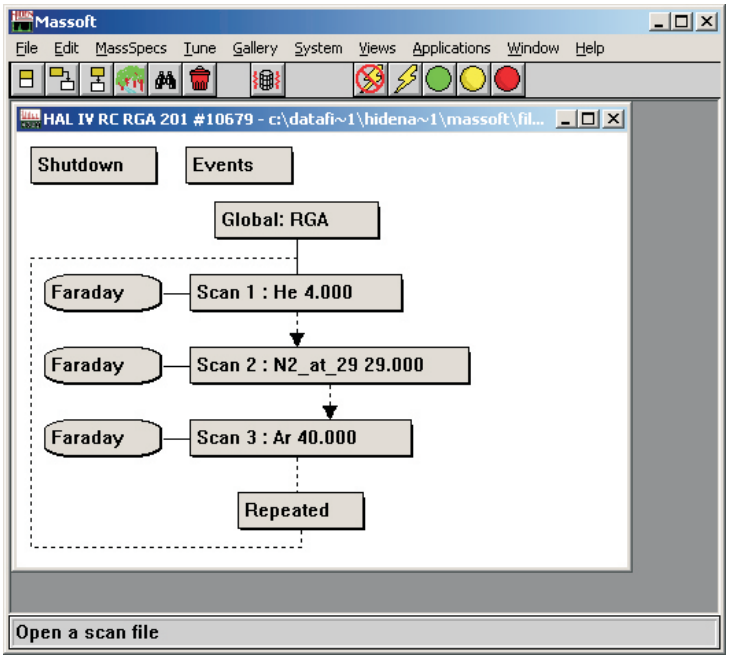

Figure 54 Scan tree, three mass scans configured

19. Double click the **Faraday** box to the left of the **Scan 1: He 4.000** box.

The **Input Selection** dialog box will be displayed, see Figure 55.

20. Select **SEM** in the **Available Inputs:** drop down list box. In the **Acquisition Range:** frame set **Start:** to **-11**, **Highest** to **-7**, **Lowest** to **-12** and enable **Auto Range**. Enable **Auto Zero**.

21. Click the **OK** button to close the **Input Selection** dialog box.

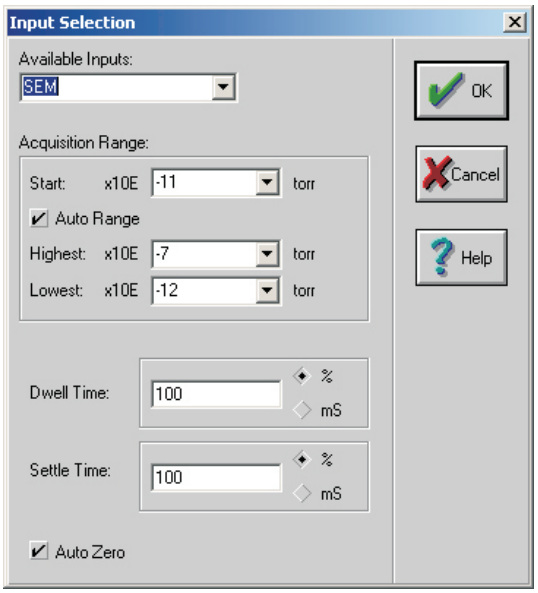

Figure 55 Input Selection dialog box

In the **Input Selection** dialog box the **SEM** detector is selected to measure the partial pressure of helium at mass 4. The first measurement will be made in the 10^{-11} Torr range. This means the electronic gain range of the instrument is set to measure in the 10^{-11} range. Auto range is enabled so if the reading is above $10x10^{-11}$ the instrument will automatically change the gain to the 10^{-10} range and re-measure the signal on the new range. If the reading is above $10x10^{-10}$ the gain range will be changed to 10^{-9} and the signal re-measured. Similarly, if the reading is less than $0.9x10^{-11}$ Torr the electronic gain range will be changed to the 10^{-12} range and the signal will be re-measured. If Auto range was not enabled the signal measured on the 10^{-11} range would be used even if it was very small or very large (over range). The auto ranging process will continue between the gain range limits entered in the **Highest** and **Lowest** drop down list boxes.

Enabling **NoDeferAutoRange** in the **Scan advanced** dialog box results in the instrument auto ranging to measure a signal within the limits ($0.9x10^{-n}$ to $10x10^{-n}$) before continuing with the next scan, in this experiment measuring the mass 29 peak. If **NoDeferAutoRange** had not been enabled the measurement would be made on the 10^{-11} range and, if it was outside the limits, autoranging would not change the gain range until the next scan of the mass peak. Enabling **NoDeferAutoRange** leads to an decrease in overall scan speed but reduces the chances of out of range measurements being recorded.

For any scan the **Start:** range should be set to the range where the measurement is expected to be made. Helium is 5ppm in air, the pressure in the system will be about $1x10^{-5}$ Torr. Therefore, 5ppm x $1x10^{-5}$ is $5x10^{-11}$, hence setting to the -11 range.

22. Double click the **Faraday** box to the left of the **Scan 2: N2_at_29 : 29.000** box.

    The **Input Selection** dialog box will be displayed.

23. Select **SEM** in the **Available Inputs:** drop down list box. In the **Acquisition Range:** frame set **Start:** to **-8, Highest** to **-8, Lowest** to **-10** and enable **Auto Range**. Enable **Auto Zero**.

24. Click the **OK** button to close the **Input Selection** dialog box.

25. Repeat steps 21. to 23. for the Ar scan setting SEM as the input **Start:** range as -7, **Highest:** as -7, **Lowest:** -10.

26. Click on the **Scan 3 : Ar 40.000** scan box to select it then insert two new scan boxes.

Refer to step 12.

The scan tree should now look like the one shown in Figure 56.

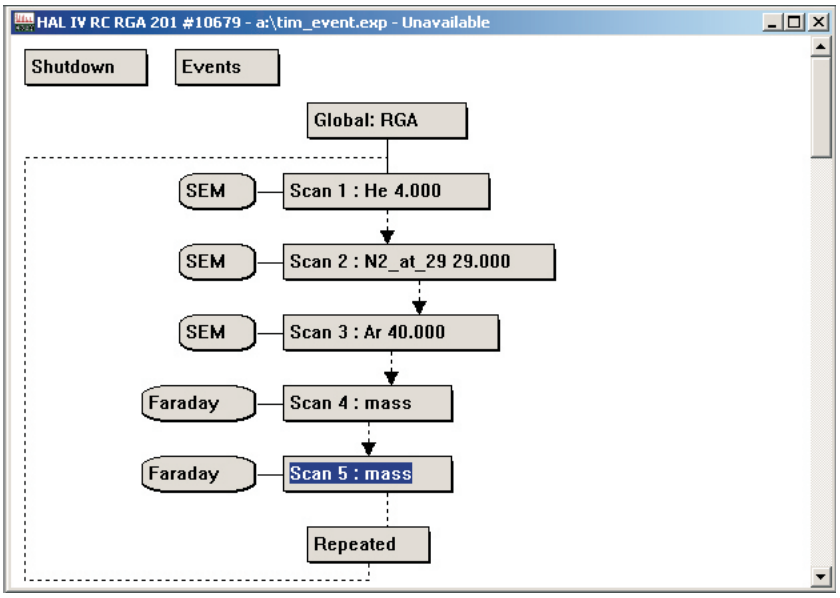

Figure 56 Scan with five scan boxes

The two new scan boxes, Scan 4 and Scan 5, will be used to receive the results of the concentration calculation that will be performed in the event sequence described in the following steps.

27. Double click the **Scan 4: mass** box.

The **Scan Editor** dialog box will be displayed.

28. Select **None** in the **Available to Scan** drop-down list box.

29. In the **Scan Legend** text box type ***He_ppm***.

30. In both the **Start Value:** and **Stop Value** boxes type ***0***.

31. Click the **OK** button to close the **Scan Editor** dialog box.

32. Repeat steps 27. to 31. for the **Scan 5: mass** box setting the **Scan Legend** to **Ar%**.

33. Double click the **Faraday** box to the left of the **Scan 4: He_ppm** box.

    The **Input Selection** dialog box will be displayed, see Figure 55.

34. Select **f(x)** in the **Available Inputs:** dropdown list box.

35. Click the **OK** button to close the **Input Selection** dialog box.

36. Repeat steps 34 to 36 for the **Faraday** box to the left of the **Scan 5: Ar%** box.

37. Double click the **Events** box.

    The **Event editor** window will be opened, see Figure 57.

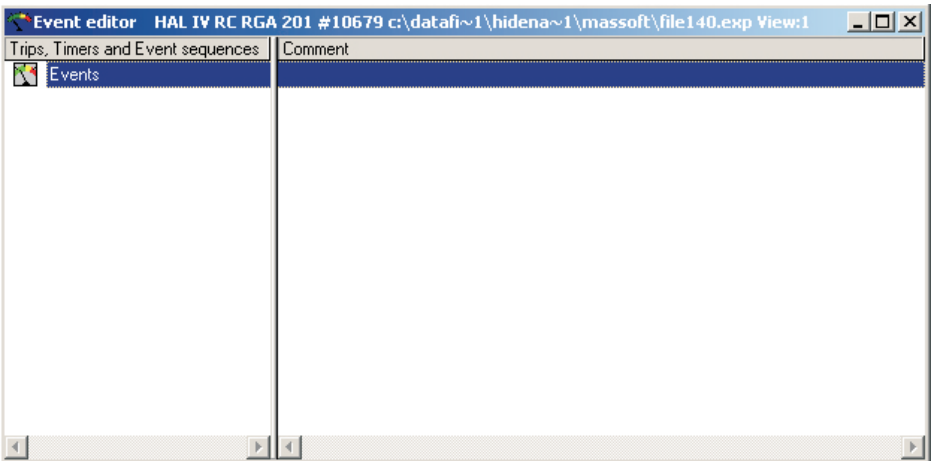

Figure 57 Event editor window

38. In the **Event editor** window right-click **Events**.

    The trip and event menu as shown in Figure 58 will be displayed.

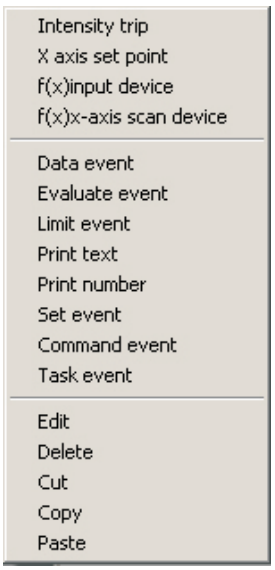

Figure 58 Trip and event menu (right click version)

39. In the trip and event menu click **Data event**.

A Data event will be added to the event sequence which will now look like the one shown in Figure 59.

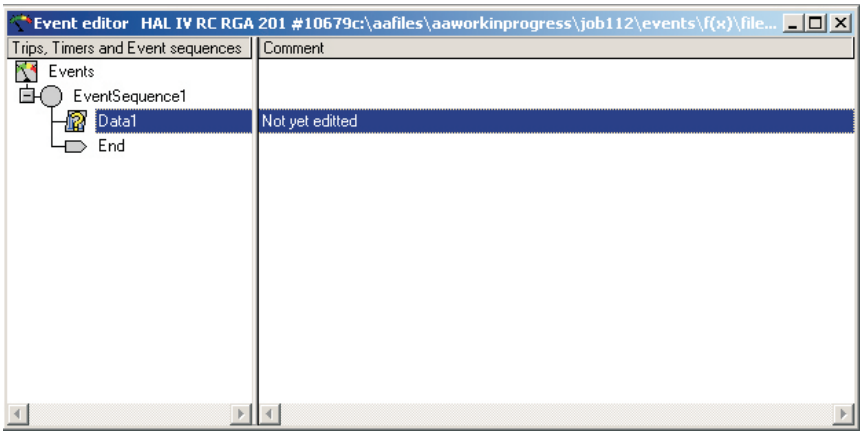

Figure 59 First Data event added to event sequence

40. In the **Event editor** double click **Data1** in **EventSequence1**.

The **Data editor** dialog box is displayed.

41. In the **Name** text box enter ***Get_He***.

Any name can be used but the first seven characters of the name must be unique within the event sequence. Fetch_data_1 and Fetch_data_2 would be read as the same and would cause a conflict.

This data event will retrieve the last partial pressure measurement for helium at mass 4.

42. Select **Scan 1 He 4.000** from the **Read data from scan** drop down list box.

See Figure 60.

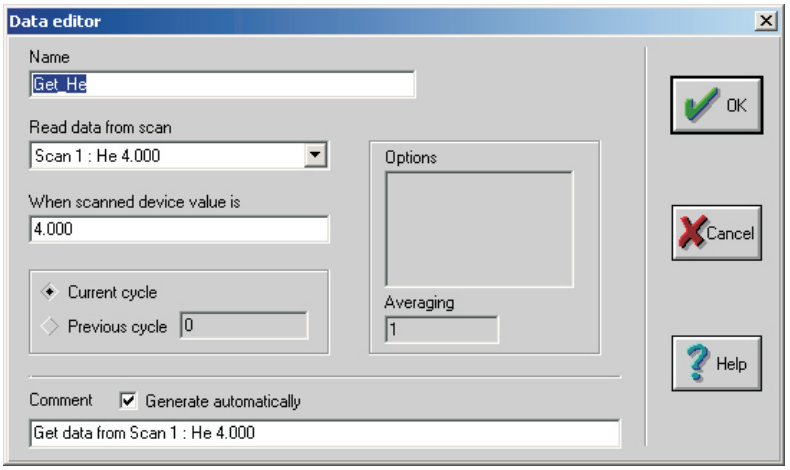

Figure 60 Data editor, helium reading

43. In the **Data editor** dialog box click the **OK** button.

The **Event editor** window will now look like the one in Figure 61.

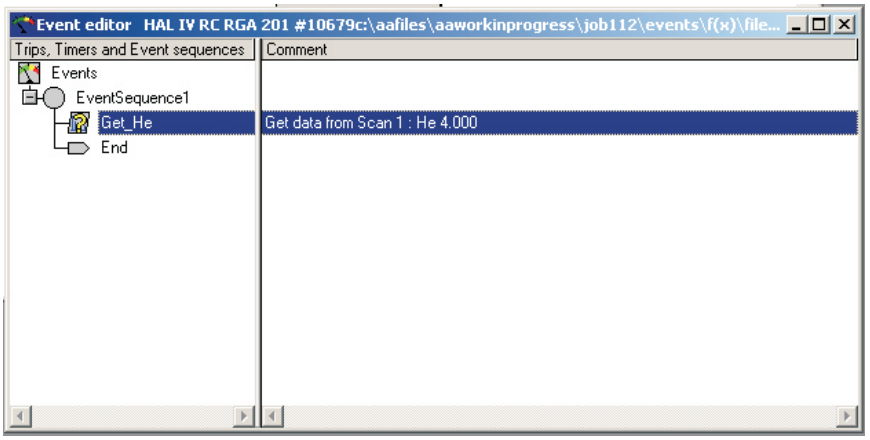

Figure 61 Event sequence with get helium event

44. Right-click **Get_He** then click **Data event** in the trip and event menu.

    Data1 event will be added to the event sequence.

45. Double click **Data1**.

    The **Data editor** dialog box will be displayed.

46. In the **Name** text box enter ***Get_N2*** and in the **Read data from scan** drop down list box select **Scan 2: N2_at 29 29.000**.

47. Click **OK**.

48. Right-click **Get_N2** then click **Data event** in the trip and event menu.

49. Repeat steps 33. to 35. to added a Get_Ar data event with the Scan 3: Ar 40.000 data read from the scan.

    The event sequence should now look like the one shown in Figure 62.

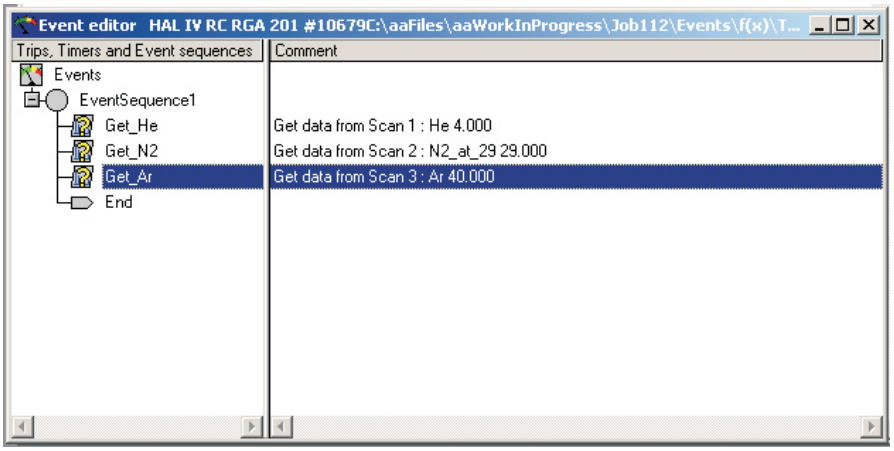

Figure 62 Event sequence with three get events

50. In the event sequence right-click **Get_Ar** and click **Evaluate event** in the trips and event menu.	An evaluate event, **Eval1**, will be added to the event sequence below the **Get_Ar** data event.
51. In the event sequence double click **Eval1**.	The **Evaluate expression editor** dialog box will be displayed.
52. Type ***He_correct*** in the **Name** text box.	
53. Select **Get_He** from the **Equals=** drop down list box.	Ensure the **Event** radio button is selected. The list of available events will not be displayed in the drop down list box if either **Constant** or **Device** is selected.
54. In the top operator box select / (division symbol).	
55. Type ***0.14*** in the box below the Equal= drop down list box.	The **Evaluate expression editor** dialog box will look like the one shown in Figure 63.

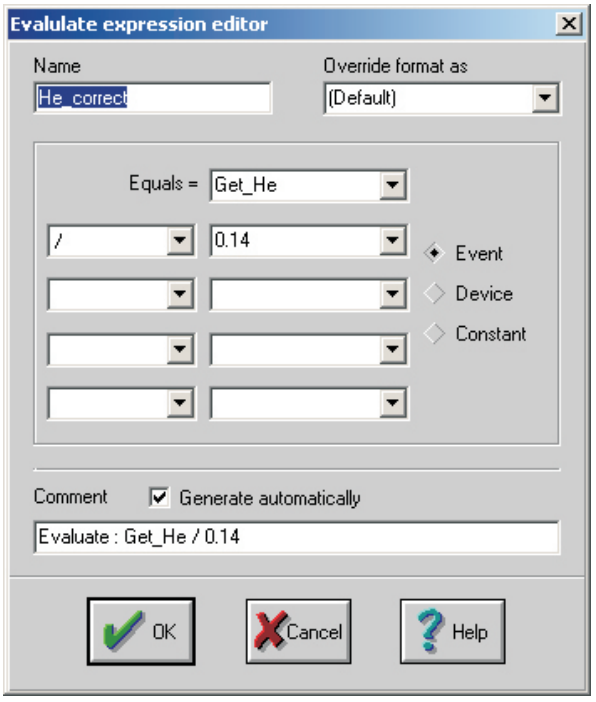

Figure 63 Evaluate expression editor, helium

The expression defined in the dialog box shown in Figure 63 will correct the mass 4 partial pressure measurement for the relative sensitivity of helium, 0.14. This means that a given pressure of helium will produce 14% of the signal of the equivalent pressure of nitrogen. It is assumed that all of the mass 4 peak is due to helium and helium only produces a peak at 4 in the mass spectrum. This is a fair assumption as the natural abundance of helium 3 is less than 0.001%.

56. In the **Evaluate expression editor** dialog box click **OK**.

The dialog box will close. The event sequence will now look like the one shown in Figure 64.

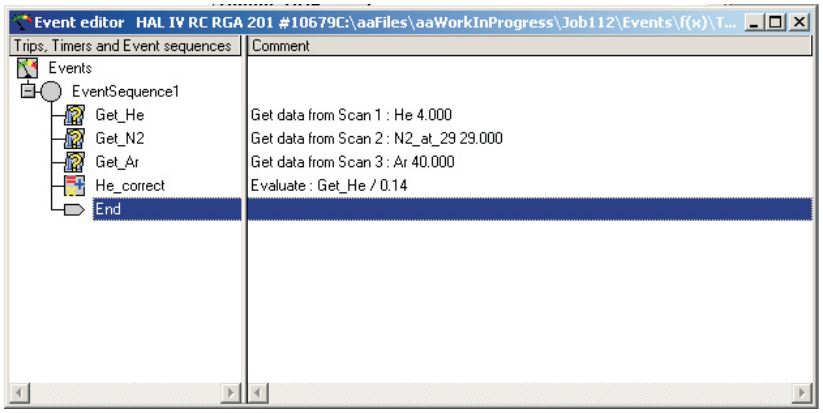

Figure 64 Event sequence, He_correct added

57. In the event sequence right-click **He_correct** and click **Evaluate event** in the trips and events menu.	A new evaluate event named **Eval1** will be added to the event sequence below the **He_correct** event.
58. Double click **Eval1**.	The **Evaluate expression editor** dialog box will be displayed.
59. Configure the event with the name N2_correct to evaluate: N2_correct=(Get_29 x 100)/1.	See Figure 65.

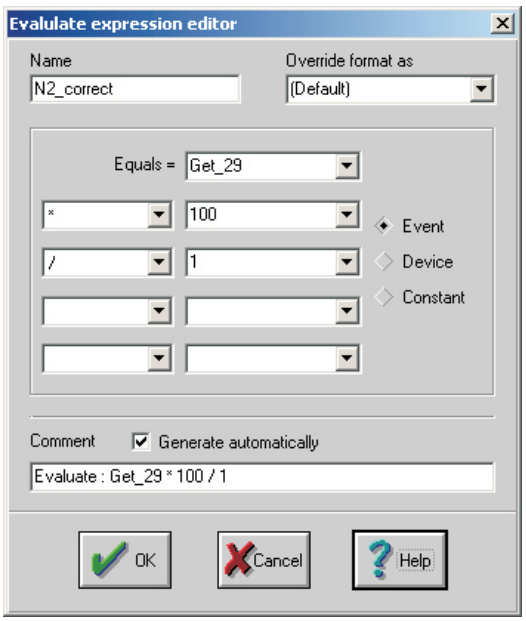

Figure 65 Evaluate expression editor, N2_correct

60. Click **OK**.

The nitrogen 29 peak is being used which is 1% of the major mass 28 peak. So, the measurement needs to be multiplied by 100. The relative sensitivity of nitrogen is 1.0.

61. Add a third evaluate event after **N2_correct** to correct the argon mass 40 measurement.

The relative sensitivity of argon is 1.2 and mass 40 is the major peak.

The Evaluate expression editor dialog box will look like the one shown in Figure 66 and the event sequence will look like the one shown in Figure 67.

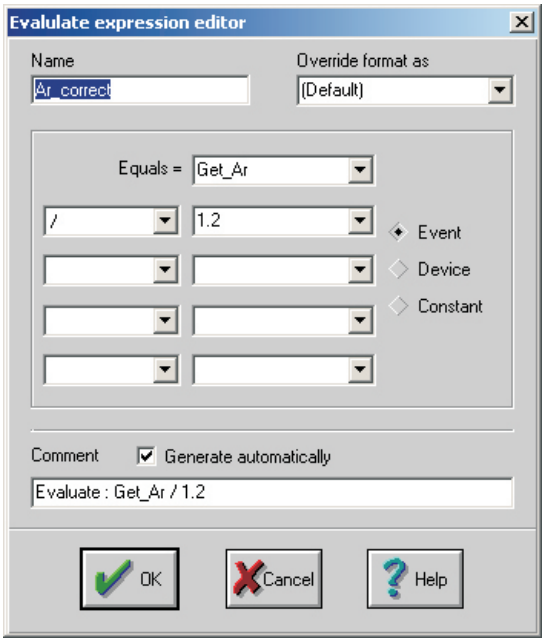

Figure 66 Evaluate expression editor, Ar_correct

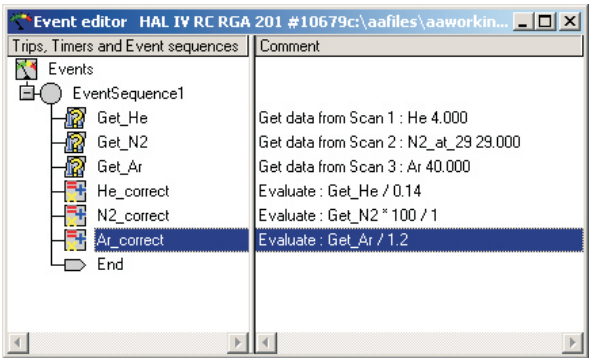

Figure 67 Event sequence, Ar_correct added

The mass 29 peak has been corrected to give an equivalent nitrogen mass 28 value but it is the pressure of air that is required. Nitrogen is 78% in air. A fourth evaluate event is needed to correct for air.

62. Right-click the **Ar_correct** event, click **Evaluate event** and enter the information for the Air_correct event.

See Figures 68 and 69.

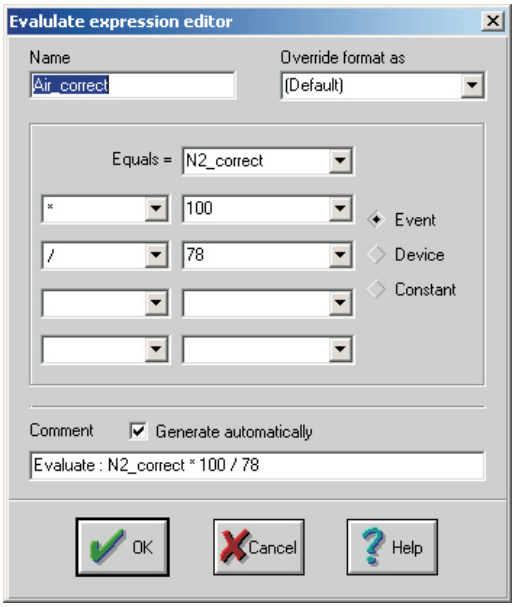

Figure 68 Evaluate expression editor, Air_correct

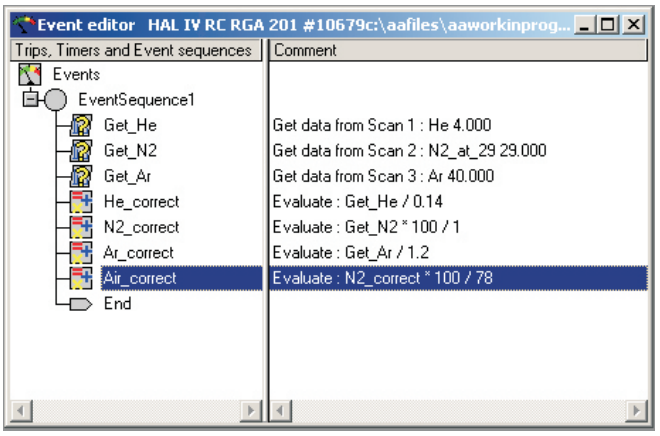

Figure 69 Event sequence, Air_correct

The ppm concentration of helium can now be calculated by dividing the corrected partial pressure of helium by the air (total) pressure and multiplying by 1000000 to convert to parts per million.

63. In the event sequence right-click on the **Air_correct** event and click **Evaluate event** in the trips and events menu.

A new evaluated event **Eval1** is added to the event sequence.

64. Double click **Eval1**.

The **Evaluate expression editor** is displayed.

65. Enter the name ***He_ppm*** and the expression **Equals= He_correct/Air_correct * 1000000**.

See Figure 70.

The resultant event sequence is shown in Figure 71.

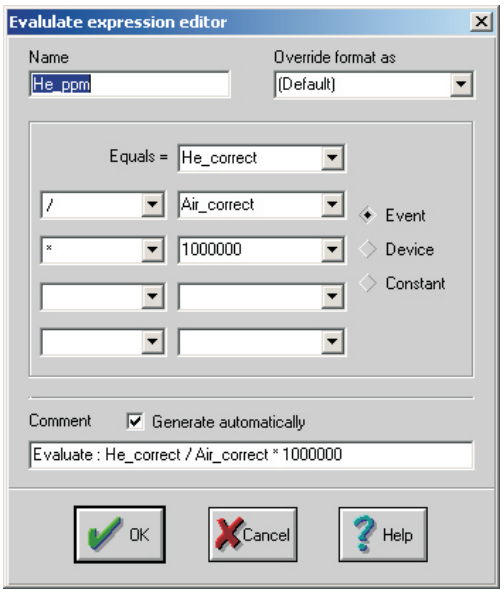

Figure 70 Evaluate expression editor, He_ppm

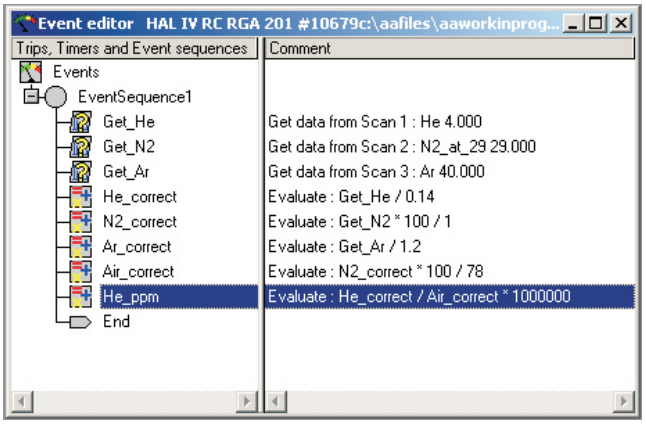

Figure 71 Event sequence, He_ppm

Another evaluation event can now be added to calculate the concentration of argon in air. This event will be added in a second event sequence. Each event sequence will correspond to an f(x) input.