Tutorial

Visual Supervisor

User Guide

EUROTHERM

invensys

About

Contents

VISUAL SUPERVISOR TUTORIAL/USER GUIDE

Intentionally left blank

CHAPTER 1 GETTING STARTED

This chapter offers instructions about how to get started with the Eycon™10 and Eycon20 Visual Supervisors, and perform the basic operations needed to benefit from the rest of this user guide.

The topics covered are:

- Front-panel keys

- Using keyboards

- First-time access

- Managing files

- Managing applications

1.1 FRONT-PANEL KEYS

Table 1-1 summarises the functions of the front-panel navigation touchkeys on both the 12.1" XGA and 5.5" ¼ VGA model, in the order they appear from left to right across the foot of the screen. These keys only work in the correct context, except for the 'Menu' key, which *always* displays the top-level menu.

TouchKey	Name	Units
	'Up' key	Steps UP a menu hierarchy, or closes current popup
	'Down' key	Steps DOWN a menu hierarchy, or cycles to next page
	'Left' key	Steps LEFT in a hierarchy or in a table
	'Right' key	Steps RIGHT in a hierarchy or in a table
	'Option' key	Pops up OPTIONs specific to the current page
	'Menu' key	Pops up the top-level main MENU
	'Fn' keys	The function key area on both the Eycon10, and Eycon20 is fully configurable using User Screen Editor software running in a Computer. This allows the user to redesign this area to suit the specfic requirements.

Table 1-1   Touchkeys

1.2 USING THE KEYBOARDS

Popup keyboards appear whenever you have to type in passwords, filenames, LIN block field values, etc. Figure 1-1 summarises how to use them.
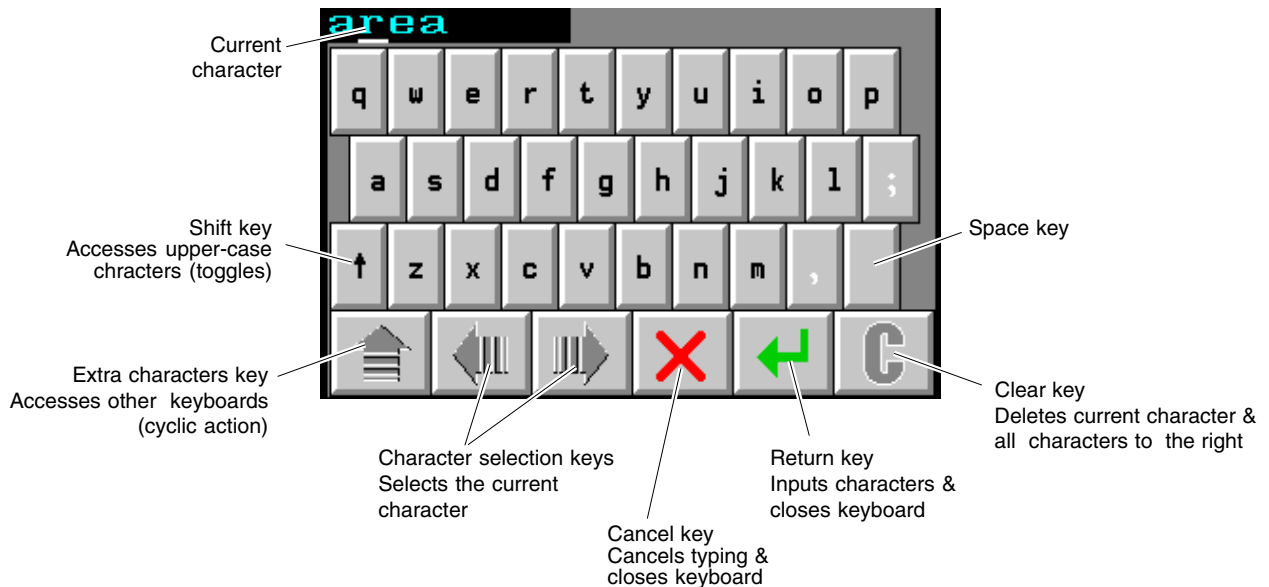
Figure 1-1   Standard keyboard

1.3 FIRST-TIME ACCESS

When first powering up, you must log in using the built-in default password. To log in at 'ENGINEER' access level, do this:

1   Press the **Menu** key to pop up the main set of keys. (Press **Menu** again, or the **Up** key, to clear the popup immediately if required.)

2   Press **ACCESS** to pop up the **Security Access** window.

3   Press the yellow text in **New Level: LOCKED**, and select **ENGINEER** from the list of levels. Press the green 'return' key to enter the selected level. (The red cross 'cancel' key closes the list and cancels the operation.)

4   Press the yellow asterisks in **Password: ********** to pop up an alphanumeric keyboard. Type in the default 8-character password, which is **[space]default**.   Press the green 'return' key to enter.

> *Note*   **Passwords are case-sensitive!**

5   Press the **CHANGE** button to make ENGINEER the current access level.

> *Note*   *If the current level remains 'LOCKED', try again ensuring that you type in the case-sensitive password correctly. If this does not work, the default password must have been altered by an earlier user. Find out what the correct password is, or contact the supplier for help.*

1.4 MANAGING FILES

To benefit from this Guide you will have to manipulate the files on the instrument's E: drive - creating, deleting, copying, uploading/downloading to/from the Computer, or USB bulk storage device, etc. However, creating specific file types relating to any additional software can also be launched via LINtools Engineering Studio.

There are four ways to manage files - you can use:

- **File Manager.** The instruments inbuilt *File Manager* utility combined with a removable storage media device. This is the slowest method, but requires no external equipment.

- **Cloning.** The instruments system for copying sets of files to or from the instrument via storage media devices. You can opt to transfer selected *application* files, e.g. programs, *system* (instrument) files, or both.

- **ELIN.** The Computer fitted with a standard *ELIN* (e.g. Ethernet) communications card, running the Explore Networks application, communicating with the Instrument using an ELIN RJ45/RJ45 cable. This is the fastest method.

1.4.1 Using File Manager

Before using the File Manager utility you must stop and unload the currently-running application - if one is running and requires **ENGINEER** login status. If not, you can skip steps 1 and 2 in the following.

1   Press the **Menu** key, then **SYSTEM**, **APPLN**, **APP MGR**, to access the Application Manager window. This shows the current application file and its state (e.g. 'RUNNING').

> *Note   Application files are stored in the instruments E: drive (EPROM), and must be copied (loaded) to RAM for running. When you unload an application you are deleting its application files from RAM.*

2   Press **STOP** to stop the running application, then **UNLOAD** to unload it.

You are now ready to use the File Manager.

3   Press the **Menu** key, then **MAINT**, **FILE MGR**, to see the File Manager window, showing data about the instrument's E: drive.

4   To select a different drive, press the yellow **E:** and select and enter the required drive.

5   You can specify a filename filter to limit the number of filenames seen. To do this, press the yellow **Filter: *.*** to pop up a keyboard. Type in the required file format using asterisks as wildcard characters, e.g. ***.dbf** will show only LIN database (.dbf) continuous control files in the **File:** menu.

> *Note   Press the **Extra Characters key** (Figure 1-1) then the **Shift key** to get to the asterisk character (and other symbols).*

6   Press the yellow **File: <filename>** to pop up a menu of filenames on the selected drive (subject to any file filter you have specified). Select and enter a file. Its size is displayed in the **Size:** field, followed by the **Free Space** on the drive.

7   To copy the selected file to another drive, press **COPY** to pop up the **Copy File** window. The destination drive and filename to be copied are displayed (and could be edited if required). Hit **OK** to execute the copy, or **CANCEL** to abort.

8   To delete the selected file, press **DELETE**, and then **OK** to confirm or **CANCEL** to abort.

1.4.2 Using ELIN communications

To set up the Computer for ELIN comms:

1   Click 🏁 **Start** , **Settings**, **Control Panel**, then double-click **LINOPC** to pop up the **Configured LIN Ports dialog**.

2   If needed, add a new ELIN port. To do this click **Add...**, select **ELIN**, and hit **OK** to see the **Edit ELIN Port** dialog.
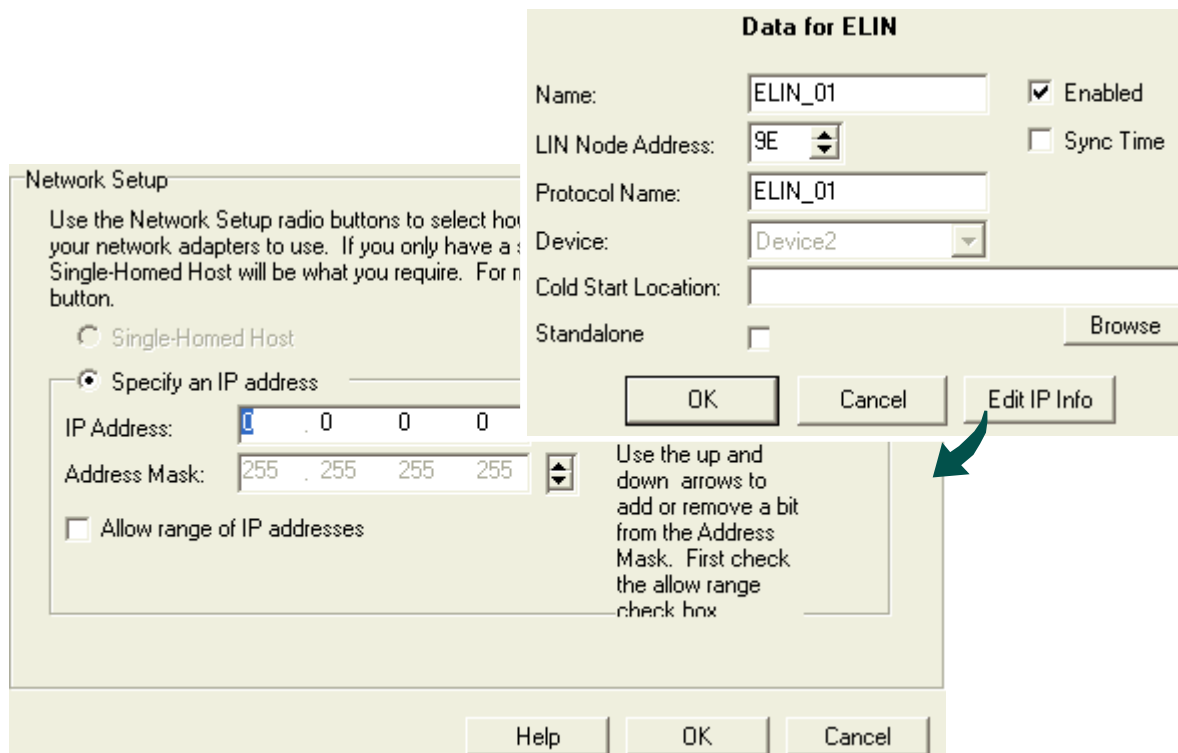
3   Enter a **Name** for the new port, and edit the **Protocol Name** fields. If necessary edit the LIN Node Address and the Edit IP Address, but leave the other fields at their default values. Hit **OK**, and close all dialogues.

   a.   The **Name**, see Data for ELIN, identifies the Network used by this instrument to communicate with other devices. It is also defined as the Online Port in Network Properties dialog.

   b.   The **Protocol Name**, see Data for ELIN, identifies the Network in the LINOPC.

   c.   The **Edit IP Info** button, reveals a dialogue that allows you to define the IP parameters relating to a specific network adapter.

   Note   The Network Port is also considered a LIN Node on the Network and MUST be configured with unique LIN Node Address and associated IP Address numbers.

 In the Instrument:

4   Press **Menu**, **SYSTEM, SETUP**, **COMMS**, to access the **Comms Setup** window, see Editing the network settings using On-Screen Menus.

5   In the **ELIN** column, press the **Protocol** cell and select **ELIN** from the popup menu. Set the **Node No.** in the range 1 to 254. Save the configured communication parameters using the **Save** key on the **Comms Setup** page in the Eycon20, or the **Save** key available when the **Option** key is pressed in the Eycon10.

6   Power the Instrument off/on to activate the settings.

7   Connect the Computer's ELIN port directly to, the Instruments ELIN Port or a Switching Unit/HUB using the correct RJ45/RJ45 cable.

   Note   The instrument communications can also be configured via the Network Settings and Instrument Options pages in the Instrument Properties dialogue displayed in the LINtools application.

To use the ELIN comms:

8   To do this, click ▥ **Start** > **Programs** > **Eurotherm > Eurotherm Network**.

> *Note*   *This application is an extension of the Windows Explorer on the Computer and can be used to view the entire control network as a hierarchy of nodes and folders. It will also allow you to transfer configuration files from between node drives, shown as folders and control the database files from the Computer desktop.A shortcut icon may already exist on the Computer desktop.*

Explore to see the **<Instrument Folder Name_*nn*> E:** folder by opening the relevant Network folder, i.e. ELIN_01, and the Instrument Folder. When the Instrument Folder is open the 'E:' folder is shown. This is where the application files are stored.

> *Tip!*   *Using the **_nn** suffix for an Instrument Folder is good practice, and gives instant Node recognition.*

9   Now you can use drag-&-drop, download, and other Explorer facilities to manage your Computer and Instrument files.

> *Note*   *You can also use LINtools Engineering Studio to download the strategy or selected files to the Instrument. Use the 'Files to be downloaded' command to show the files that will be downloaded on request.*

1.4.3 Setting the IP Address

> *Note   For a more comprehensive description of IP Addresses, refer to the ELIN User Guide*
> *(Part no. HA082429) for details.*

An instrument (IP host) will always need an IP Address, this can be allocated either automatically or manually. Which method (and the allocated IP Address used) will depend on any existing (or planned) networks.

Each instrument uses a one-to-one mapping of, LIN Node Number to a single IP Address, defined in the 'network.unh' file.

IP ADDRESS

ELIN runs over Ethernet using IP (Internet Protocol). Instruments (IP hosts) are identified by an 'IP Address',

■   Expressed in 'dotted decimal' notation

Example   192.168.111.222

■   Actually represents a 4 byte (32-bit) number

Example   0xC0 A8 6F DE

PORT NUMBER

By default, all ELIN instruments automatically use Port Number 49152. This is a configured port number in the Dynamic, or Private, port range that allows private access to the LIN.

> *Note   For a more comprehensive description of Port Numbers, refer to the ELIN User Guide*
> *(Part no. HA082429) for details.*

ALLOCATION OF IP ADDRESS

IP Addressses can be allocated in a manner appropriate to the local company network policy.

■   **DHCP** - This is a method whereby the instrument (IP host) will ask a DHCP server to provide it with an IP Address. Typically this happens at start-up, but can be repeated during operation. DHCP includes the concept of 'leases', i.e. the assigned value will 'expire'.

A DHCP server is required that can respond to the request. The DHCP server will need to be configured to correctly respond to the request. This configuration will depend on the company network policy.

■   **Link-Local** - Link-Local is used as a fallback to either DHCP or BootP, or can be used on its own as the only IP Address configuration method. Link-Local will always assign an IP Address in the range 169.254.X.Y. This IP Address range is reserved for use by Link-Local and is explicitly defined as private and non-routable.

The Link-Local algorithm ensures that an instrument (IP host) on a network will chose a unique IP Address from the Link-Local range.

This is supported by Windows 98 and onwards, and was originally specified as a fallback from DHCP.

■   **Manual** (or **Fixed**) - The IP Address is explicitly defined in the 'network.unh' file. This is the recommended method of IP address allocation.

■   **BootP** - BootP or Bootstrap Protocol (Internet (TCP/IP protocol)) is used by a network computer to obtain an IP Address and other network information such as server address and Default Gateway. Upon startup, the client station sends out a BOOTP request to the BOOTP server, which returns the required information. A BootPtimeout period can be configured. If this period elapses before the IP Address, Subnet mask, and Default Gateway address are obtained, the values will automatically reset to 0.0.0.0.

EDITING THE NETWORK SETTINGS USING ON-SCREEN MENUS

The editing procedure for Communications Parameters consists of displaying the Comms Setup page and setting up or editing the parameters for each port fitted to the unit.

The SAVE button is used to save the changes, or to cancel the changes before saving them, the CANCEL button is used.

Before any saved changes can take effect, the application must be stopped and then restarted, or the instrument must be powered off and on again. Generally, 'parameter' changes (such as baud rate) require only a stop and restart of the application, whereas 'hardware' changes (such as changing a Modbus master port to a slave port) require a power cycle.

1. Press the Menu key.

2. Press SYSTEM from the Pop-up menu.

3. Press SETUP

4. Press COMMS.
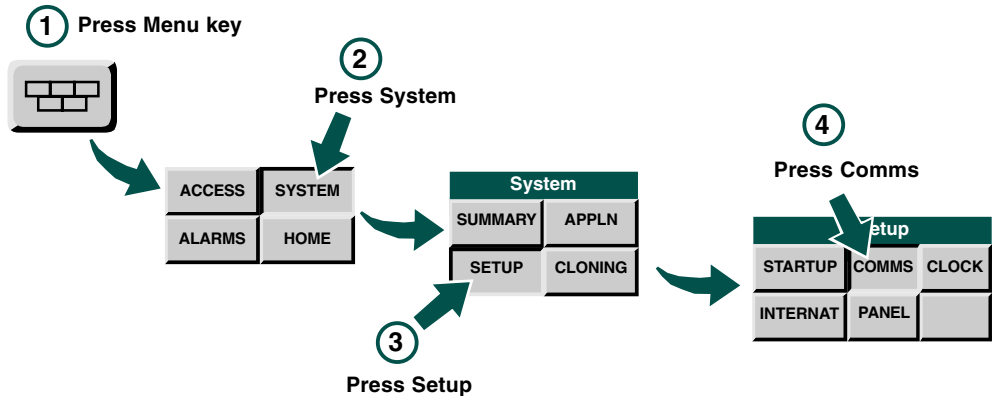
The Comms Setup page appears.

For each port (COM1, COM2, ENET1, ENET2…) there is a column of parameters (Hardware, Protocol, Mode No…). If necessary, the vertical slidebar can be used to display more parameters hidden further down the page. The full list is:

■ Hardware Standard (for example, RS485)

■ Protocol (for example, Modbus Slave)

■ Node Number (decimal)

■ Baud (rate)

■ Parity

■ Data bits (number of)

■ Stop bits (number of)

■ Timeout (Modbus Master only, in milliseconds)

■ TalkThru (Modbus slave only)

5. Press the ETHERNET key.

The Comms - Ethernet page appears.

6. Set the fields as shown. Locate the IP Address fields, and enter the required 3 digit numeric in each of the 4 fields using the numeric keypad displayed when each field is pressed.

> *Note   While editing this page ensure the Protocol Name field shows the Network connected via this instrument. Remember, if more than one user is setting up ELIN Networks on a shared Ethernet Network, a conflict of LIN Node numbers could occur if the Default Protocol name, NET, is used by everyone.*
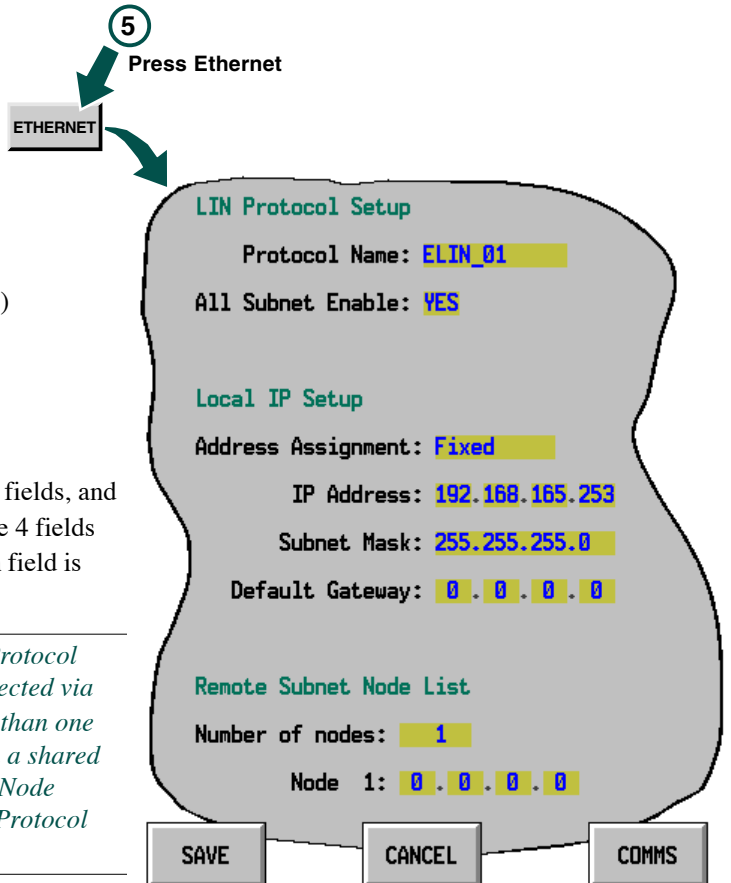
7. Press SAVE.

EDITING THE NETWORK SETTINGS USING INSTRUMENT PROPERTIES

The Instrument Properties dialog is simply a window used to display the information stored in the Network.unh and _system.opt files.

Each instrument uses a one-to-one mapping of, LIN Node Number to a single IP Address, defined on the Network Settings page and the Instrument Options page in the Instrument Properties dialog, stored in the 'network.unh' file.

> *Note The Compact Flash card is accessed using a standard Compact Flash card reader. The 'network.unh' file MUST be edited using the Instrument Properties dialog.*

When despatched from the factory, the instrument properties are configured for using DHCP with Link-Local Fallback, and a default LIN Protocol Name, 'NET'.

Important If more than one user is setting up ELIN Networks on a shared Ethernet Network, a conflict of LIN Node numbers could occur if the Default Protocol name, NET, is used by everyone.

However, if the instrument is to have a fixed IP Address, i.e. 192.168.111.2, and use the LIN Protocol Name, 'ELIN_01', the Instrument Properties dialog can be used to modify these parameters.

■ The Protocol Name shows the name of the Network used by this instrument to communicate with other instruments and is also defined as the Online Port in Network Properties dialog.

> *Note The IP Address must correspond to the local company Network Policy.*
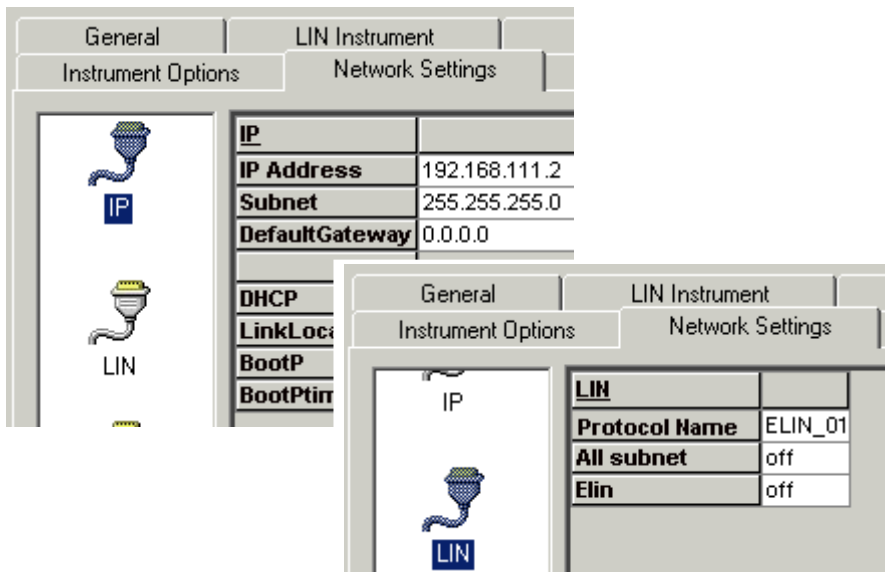
To display the Instrument Properties dialog, select the **Properties** command after selecting the Instrument Folder in an appropriate Explorer view.

Select the Network Settings page, and press the Upload Current Settings button. This will connect to the 'Live' instrument and automatically populate all the parameters on this and the Instrument Options page.

When the applicable parameters have been edited, use the 'OK' button to confirm the changes. This will also display a request to update the Instrument parameters.

1.5 MANAGING APPLICATIONS

Once you have created a LIN strategy in your Computer and downloaded it to the Instrument, you must load and run the application. To carry out these, and other, operations on an application you use the inbuilt **Application Manager**.

> *Note   Applications can also be managed via LINtools, by simply connecting to the instrument, and using the Online Instrument Status dialog to stop the application, open a different strategy, and download and run it in the instrument.*

To access and use this utility:

1   *If no application is loaded*, press the **Menu** key, then **APP MGR**, to see the **Application Manager** window. *Otherwise*, press the **Menu** key, then **SYSTEM**, **APPLN**, then **APP MGR**.

2   To select an application file for loading, running, or deleting, press the yellow **File: ????????** field to pop up a menu of the available application files. Select and enter the required file. Its name appears in the **File:** field.

You can now operate on the selected file, or on a running application, according to Table 1-2, which summarises the actions of the various Application Manager buttons.

Press...	To...
STOP	Stop the running application
SAVE	Save the stopped* application to the E: drive under its current name
SAVE AS	Save the stopped application to the E: drive under a new name
UNLOAD	Unload the stopped application from RAM
LOAD	Load the selected application to RAM
START	Run the loaded application
LD+RUN	Load the selected application to RAM and then run it
DELETE	Delete the selected unloaded application
	*Note **Press Save.** You can save an application under its current name while it is still running, by setting the Options.FullSave bit to TRUE in the instruments LIN database header block. However, **stopping the application and using the SAVE button** is the recommended method.*

Table 1-2   Application Manager window buttons

Intentionally left blank

CHAPTER 2 A PID LOOP

This chapter shows you how to build a simple PID loop control application for the Instrument, using the LINtools Engineering Studio. Using this application you will be able to:

■ Explore the Group display and PID block faceplate

■ Look at realtime and historic trend displays

■ Inspect database fields

> *Note* *In this example the PID loop runs locally in the Instrument, using a PID (LIN) function block. However, you may prefer to run the loop remotely in a T2550 I/O Subsystem. In this case you must configure I/O category function blocks, AI_UIO and AO_UIO blocks, to run in the Visual Supervisor instrument communicating with the configured T2550 loop(s).*

The main topics covered in this chapter are:

■ Opening the LINtools Engineering Studio

■ Creating the PID control loop strategy

■ Testing the PID loop simulation strategy

■ Looking at alarms

2.1 OPENING LINTOOLS ENGINEERING STUDIO

The LINtools Engineering Studio window is where you create and edit LIN Databases, LIN Sequences, and LIN Actions. From this window you can access the LIN Database on both the Computer or the 'Live' instrument.

> *Note* *The strategies created during this tutorial are created for the Eycon10, but instructions and principles are also applicable to the Eycon20.*

1　In the ⊞ **Start** menu, select **Programs** > **...** > **LINtools Engineering Studio**. A blank configuration window opens.

> *Note* *The '...' denotes the default installation path.*

If launching a new instance of this software, a 'Get Me Started' wizard will appear offering a selection of commonly required operations, if not disabled during a previous session.

2　On the 'Get me Started' wizard, click the **Create a new LINtools file** to display a **Select File Type for New Editing Session** dialog.

3　In the dialog, select **New LIN Database [FBD]** and hit **OK**. LINtools Engineering Studio opens, with blank default windows displayed, ready for you to start the configuration.
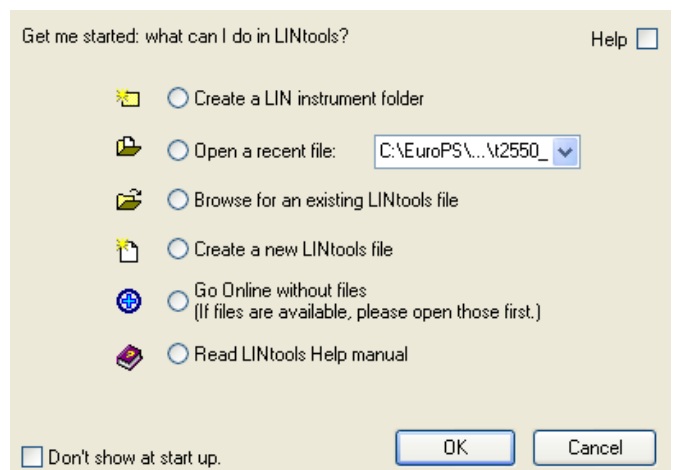
> *Note* *If you try to create a new configuration while an unsaved configuration is already open, you will be asked if you want to save the current file.*

2.2 CREATING THE PID CONTROL LOOP STRATEGY

Figure 2-2 shows the simple strategy that you will create in LINtools Engineering Studio and summarises the LIN block functions.

Once you have accessed the LINtools Engineering Studio you will be able to get help on all aspects of building up the strategy, by pressing the <F1> 'help' softkey. What follows is therefore only a brief outline of the steps you need to take.

A full working system requires you to create two databases, the PID Control Loop strategy should be created in a T2550 database and a strategy for displaying Trends, PV, SP, and OP, in the Eycon. For this to operate correctly the PID block must be cached from the T2550 to the Eycon database.

For clarity the cached PID Control Loop strategy has be shown, however, to simplify this tutorial, the PID block and SIM block can be created in the Eycon database, because the I/O is controlled via the T2550.

PID control block
Generates a PID (proportional-integral-derivative) control output (OP) from a resultant setpoint (SP) & a process variable input (PV)

EYCON-10/EYCON-20 header block
A header block is required in all LIN instrument strategies

AI_UIO
An Analogue input block Allows communication with a single analogue input channel running in a remote T2550 I/O module. (Other I/O blocks can be used depending on application - see Table 2-1)

AO_UIO
Analogue output block Allows communication with a single analogue output channel running in a remote T2550 I/O module. (See also Table 2-1)

AREA block
Associates up to 16 GROUP blocks with an Area, and enables the Instruments' Overview display. AREA & GROUP blocks together are needed for display navigation & data recording.

GROUP block
Associates up to 16 display-type & 16 recorder channel blocks with a Group, and enables the instrument Group display.

DR_ANCHP
A data recording analogue channel point block Allow flash recording (trending) of a database floating-point value

Note   For more detailed LIN Block information refer to the LIN Blocks Reference Manual.

Figure 2-2a   Simple PID control loop strategy

2.2.1 Placing the function blocks

CREATE THE T2550 PID CONTROL LOOP STRATEGY

1   In the **Function Block Template Palette** pull down the **Library** menu and select **T2550**. Ensure the defined version corresponds to your instrument. The Configuration (Header) blocks associated with this instrument appear. Drag the header block onto the worksheet and position approximately as shown in Figure 2-2.

> *Note   If the instrument folder was created using the New Instrument wizard, the Header block will already exist.*

2   Select the **Control** function block category and drag a **PID** block onto the worksheet.

3   Now select the **I/O** (Input/Output) category, and place the **AI_UIO** and **AO_UIO** blocks. In a real situation you may want to select different I/O blocks, depending on your particular application and T2550 hardware.

Change each block name, as shown in the image below.

> *Note   If the T2550 LIN Database file already exists these blocks can be dragged to the Eycon .dbf file, automatically creating cached blocks. Other means of creating cached blocks are described in the LINtools Engineering Studio Help file.*
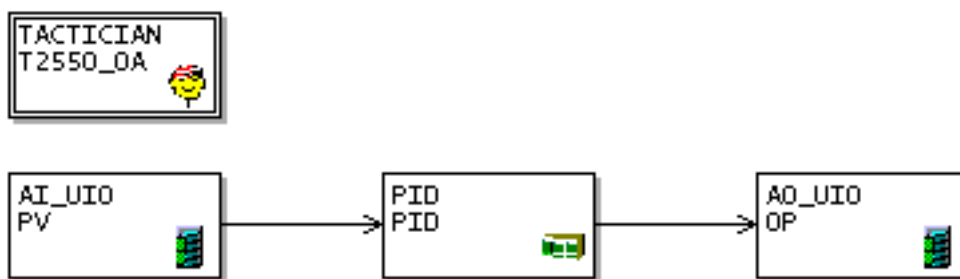
CREATE THE EYCON STRATEGY

    IMPORTANT   *It is good practice to wire from a cached block to local blocks.*

1   Create a new database file for the Eycon.

2   In the **Function Block Template Palette** pull down the **Library** menu and select either the **EYCON-10** or **EYCON-20**. Ensure the defined version corresponds to your instrument. The Configuration (Header) blocks associated with this instrument appear. Drag the header block onto the worksheet and position approximately as shown in Figure 2-2.

> *Note   If the instrument folder was created using the New Instrument wizard, the Header block will already exist.*

2   Select the **Control** function block category and drag a **PID** block onto the worksheet. Now select the **I/O** (Input/Output) category, and place the **AI_UIO** and **AO_UIO** blocks. In a real situation you may want to select different I/O blocks, depending on your particular application and T2550 hardware.

3   Rename each of these blocks, as shown above, and change the DBase field to identify the T2550 database. This will automatically change these to cached blocks, indicated by the green arrow symbol.

> *Note   If the T2550 LIN Database file already exists these blocks can be dragged to the Eycon .dbf file, automatically creating cached blocks. Other means of creating cached blocks are described in the LINtools Engineering Studio Help file.*

4   From the **Recorder** category, place the three **DR_ANCHP** blocks. (You can place the first block from the list then use the configurator's copy-&-paste facility to create the rest.)

5   From the **Organise** category, drag and drop the **AREA** and **GROUP** blocks to the worksheet.

2.2.2 Wiring up the blocks

Start with the *PV* input from the AI_UIO to the PID block:

1   On the source AI_UIO block, click the *category icon* in the lower right corner of the box. This pops up a menu of wiring source fields. Double-click **PV** to close the menu and activate 'wiring mode'. The 'wiring' cursor appears.

2   Now click this cursor on the destination PID block to pop up a menu of wiring destination fields. Double-click **PV** to select it and close the popup. The new wire appears between the two blocks.

Note   To get help with wiring, press <F1> while a connections menu is open.
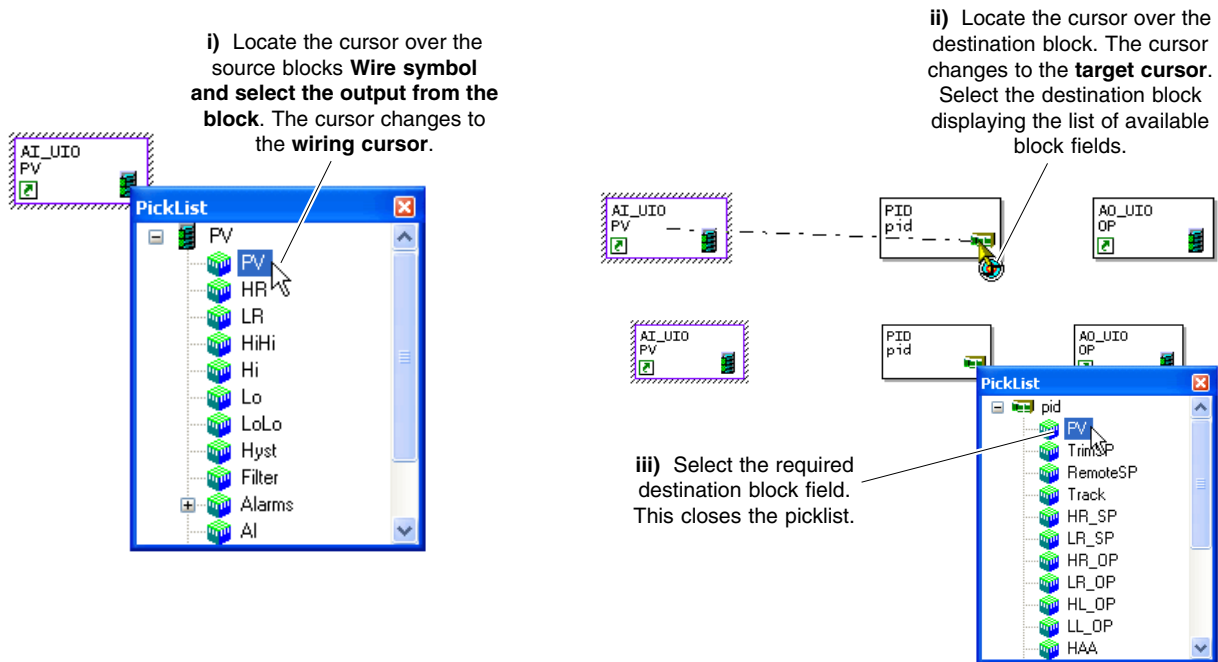
Figure 2-2b   Wiring data

3   Add the remaining wiring connections shown in Figure 2-2. Table 2-1 lists all the source and destination fields that should be interconnected in this strategy, and summarises their functions.

Note   Multiple wires connecting two blocks appear as a single wire.

Source*	Destination*	Function
(AI_UIO) PV.PV	(PID) pid.PV	Process Variable (PV) from the remote T2550 input module into the PID block
(PID) pid.OP	(AO_UIO) OP.OP	PID control output OP to the remote T2550 output module
(PID) pid.OP	(DR_ANCHP) OP_rcrd.CurrVal	PID control output OP into the OP_rcrd (DR_ANCHP) block to enable data recording onto instrument's local flash storage
(PID) pid.PV	(DR_ANCHP) PV_rcrd.CurrVal	PID process variable PV into the PV_rcrd block to enable data recording
(PID) pid.SP	(DR_ANCHP) SP_rcrd.CurrVal	PID setpoint SP into the SP_rcrd block to enable data recording
*Note *(Block type) blockname.field.subfield format, referring to Figure 2-2*		

Table 2-1   Strategy wiring data

Note   The Configuration (Header) block and the AREA and GROUP blocks do not need to be wired up. They are associated with other blocks by setting field values.

4   You may want to tidy up and reroute the wiring to look more like that in Figure 2-2!

LINtools wiring help (press <F1>) tells you how to do this.

2.2.3 Editing the block fields

Many of the block fields in this strategy can be left at their default values, but some should as a minimum be edited. Table 2-2 lists these fields (apart from the **Name** fields).

Start by naming the blocks:

1 Click (or double-click) on a block symbol in the worksheet to access the block's BLOCK pane. (If a BLOCK pane is already open a single click will do.)

2 Click the white "**NoName**" field once to highlight it, then type in the name of the block (Figure 2-2 suggests block names). Alternatively, double-click the white field to insert a text cursor, and use this to edit the existing name. Hit **<Return>** to enter the new name.

3 Continue editing the other block fields, according to Table 2-2 or to your particular application.

4 Save the completed strategy. (LINtools help gives information on the *Save* operation.). You will notice that the strategy automatically builds. This automatically adds these changes to the Project Database.

> *Note* *You can get help with editing function block fields by pressing <F1> while a text cursor appears in a field.*

Field *Value	Function
(AI_UIO) PV.DBase=**<remote database name>**	Identifies the origin of this cached block, creating a link from the values in this block on the T2550 to this instrument
(AO_UIO) OP.DBase=**<remote database name>**	Identifies the origin of this cached block, creating a link from the values in this block on the T2550 to this instrument
(PID) pid.TI = **5.00**	Sets the PID integral time constant to 5 seconds, to improve control
(PID) pid.LAA = **20.00**	Sets the pid block's PV output 'low absolute' alarm at 20%
(PID) pid.Alarms.LowAbs = **1**	Enables the 'low absolute' alarm on the pid block's PV output
(AREA) area.Id = **1**	Allocates area number (only Area '1' currently implemented). No OVERVIEW appears in the Instrument unless an area number has been allocated
(AREA) area.Group1 = **pidloop**	Assigns the 'pidloop' GROUP block to this area. This specifies that pidloop will appear in the OVERVIEW and enables data recording for this group
(GROUP) pidloop.Update = **10**	Specifies the recording sample rate (seconds) for this group. The default rate of zero disables recording
(GROUP) pidloop.Disp1 = **pid**	Specifies the PID block 'pid' as the source of the data that will be displayed in the faceplate and trends for the 'pidloop' group
(GROUP) pidloop.Chan1 = **PV_rcrd**	Assigns the recorder channel block 'PV_rcrd' to channel 1 of the 'pidloop' recording group
(GROUP) pidloop.Chan2 = **SP_rcrd**	Assigns the recorder channel block 'SP_rcrd' to channel 2 of the 'pidloop' recording group
(GROUP) pidloop.Chan3 = **OP_rcrd**	Assigns the recorder channel block 'OP_rcrd' to channel 3 of the 'pidloop' recording group
(DR_ANCHP) OP_rcrd.CurrVal[units] = **%**	(*Engineering units for a field are entered in the white cell to the right of its 'value' cell*.) Assigns '%' engineering units to the PID output field OP, which will appear on the corresponding trend faceplate
(DR_ANCHP) PV_rcrd.CurrVal[units] = **degC**	Assigns 'degC' engineering units to the PID process variable field PV, which will appear on the corresponding trend faceplate
(DR_ANCHP) SP_rcrd.CurrVal[units] = **degC**	Assigns 'degC' engineering units to the PID setpoint field SP, which will appear on the corresponding trend faceplate

> *Note* *(Block type) blockname.field.subfield format, referring to Figure 2-2*

Table 2-2   Values for non-default block fields

2.3 TESTING THE PID LOOP SIMULATION STRATEGY

A simple way to get an idea of how the strategy will work in a Instrument, without having to configure and connect up a T2550 I/O Subsystem, is to use a SIM block to simulate plant behaviour. In the slightly modified strategy shown in Figure 2-3, a SIM block simulates a process variable and its response to the changing control output.

To simplify the Eycon tutorial, the I/O blocks can be ignored. These I/O blocks are applicable to a system that includes a T2550,. However, by locally adding the PID block and wiring the SIM block in the Eycon database as specified, a load can be simulated, and used to produce an output.
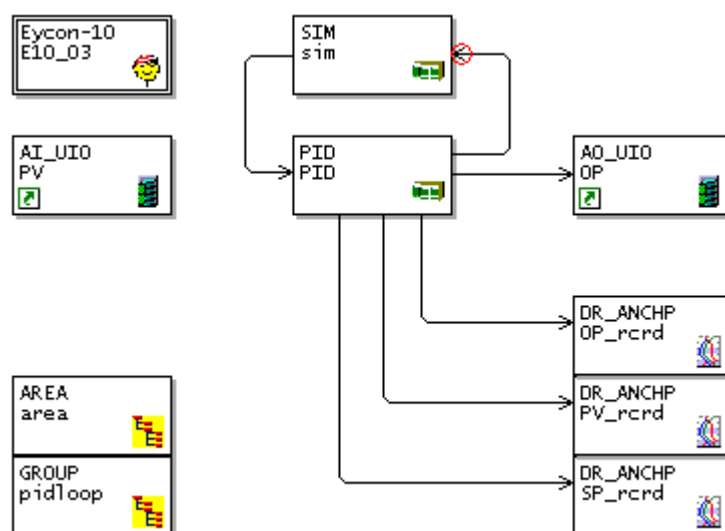
Figure 2-3a   Testing the PID control loop strategy

2.3.1 Modifying the strategy

1   Delete the wire between the AI_UIO 'PV' block and the PID block. To do this, simply click on the wire to highlight it then press the **<delete>** key.

2   Place a SIM simulation block on the worksheet, as shown in Figure 2-3a. (SIM blocks are in the **Control** function block category.)

3   Wire the **SIM.OP** field into the **PID.PV** input, and **PID.OP** into **SIM.PV**. This forms a closed loop, indicated by the small red circle.

4   Name the SIM block, 'sim', and edit both its **NoiseMax** and **Lag1** fields to **25.0**.

> *Note*   *The SIM block generates pseudo-random noise at its output* OP, *which simulates a varying process variable input to the PID control block's* PV *field. The PID block's resulting control output* OP *is fed back to the SIM block's* PV *input, where it is delayed and added to noise to simulate the plant's response to the control input.*

5   Save the modified strategy (under a different name).

2.3.2 Running the application

To run the PID loop simulation strategy and try out some instrument features, do the following:

1   Download the saved modified strategy file to the instrument. Only the **.dbf** file is required by the instrument, the other two files that appear when you save a strategy (**.grf**, and **.dtf,**) are only used by LINtools.

> *Note   If you need help with downloading, see Chapter 1,* Getting started with the instrument.

2   Select the application. To do this, press the **Menu** key (see Figure 2-3), then **APP MGR** to pop up the **Application Manager** window. Press **File: ????????** (the yellow area) and select the application from the list, which will have the same filename as the .dbf file. Press 'return' (the green arrow key) to enter your selection.

3   Load and run the selected application, by pressing the **LD+RUN** key. The 'pidloop' group display appears, with its single 'pid' faceplate representing the PID control block (Figure 2-3b, top left). The 'LED' shows the loop alarm status, and the current PV, SP, and OP readouts.

> *Note   As there is only one group associated with this area, an 'area overview' display of the groups is not necessary and does not appear.*
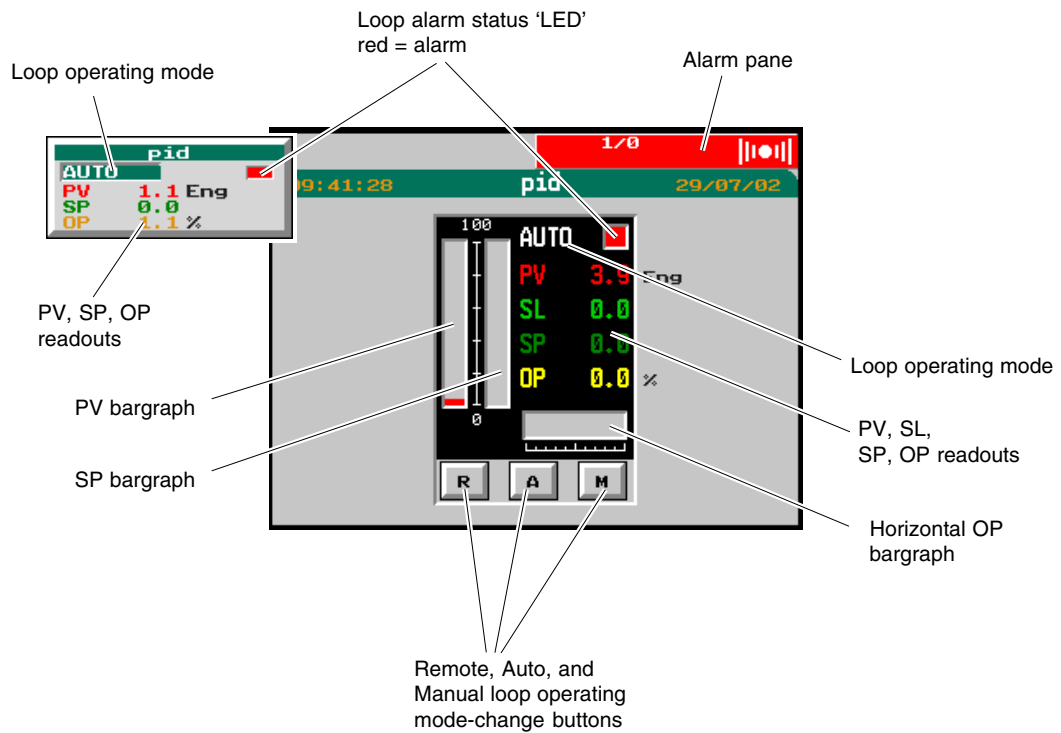
Note   A Touch Screen keypad is available below the display screen area

Figure 2-3b   PID faceplate (top left) and larger function block faceplate

2.4 EXPLORING THE PID FUNCTION BLOCK FACEPLATE

The larger PID function block faceplate gives a more detailed view of the loop parameters, and allows you to change the loop operating mode, local setpoint (SL), and output (OP).

1   Press the 'pid' faceplate to see the **PID function block faceplate** showing the loop parameters (Figure 2-3b). Note the PV, SL, SP, and OP numeric displays, PV and SP vertical bargraphs, horizontal output bargraph, alarm status 'LED', operating mode legend, and the three operating mode-change buttons, **R** (remote), **A** (automatic), and **M** (manual).

2   Press the bright-green **SL** readout, the numerals, not the letters, to pop up an edit keyboard. Enter a new local setpoint value. The loop attempts to control to the new resultant setpoint SP.

3   Now press the **M** button to put the loop into Manual mode. The operating mode legend at the top of the faceplate confirms your selection.

4   Press the yellow **OP** readout and enter a new output value in the popup edit keyboard. You can do this because you are in Manual mode. After a while, return the loop to auto mode by pressing the **A** button. The output reverts to automatic control.

5   Finally, press the **Up** key to return to the 'pidloop' group display.

2.4.1 Looking at realtime trends

Remember, three analogue points are being trended, OP, PV, and SP. This is because you wired them to DR_ANCHP data recording blocks and assigned these blocks to channels in the 'pidloop' recording group. To explore the configured trends:

1   In the 'pidloop' group display, press the **Down** key repeatedly to cycle round the configured realtime trend formats, eventually getting back to where you started.

> *Note*   Faceplate, Horizontal Bars, *and* Vertical Trend *are the default formats, but you can configure others by setting the AREA block's* DispMode *fields appropriately, see* Inspecting LIN function blocks *below for how to do this.*

2   Cycle to the vertical trend (with faceplates). Note the colour-coding and the engineering units, that you configured earlier, displayed on each faceplate. Press a faceplate to see its source block faceplate ('pid'). Press **Up** to return to the trend.

2.4.2 Looking at trend history

Various options are available for viewing historical trends.

1   Access the vertical trend (press the **Down** key as needed). Note that latest data is at the top, and the display span in **mins:secs**, **hrs:mins**, or **days-hrs** is shown at the bottom-left of the trend. See Figure 2-4, top left.

2   Touch the trend to see a dotted line cursor. The cursor date and time now appear below the trend, and the faceplates show the (three) cursor values, for OP, PV, and SP in this case.

3   Press the **Option** key to pop up the available options, then press the **VIEW** option. **Zoom** and **Pan** controls appear below and to the side of the trend, respectively (Figure 2-4).

These controls allow the amount of data displayed on the screen to be varied, either continuously, using the slider, or in steps (using the keys). The expansion/contraction of the trends is centred on the cursor position.

> *Note*   *Changing the PID.TimeBase field from Secs to Mins increases the amount of data displayed. The amount of data displayed can then be decreased by changing the same field back to Secs.*

4   Press the **Option** key, then **VIEW**, to restore the standard view of the trend.

2.4.3 Inspecting LIN function blocks

You can inspect and edit the fields in the LIN blocks running in your application.

1   Press the **Menu** key at the foot of the screen, then **SYSTEM**, **APPLN**, **FB MGR**. The **Function Block Manager** screen appears, showing the (10) LIN blocks in the application as an array of named boxes.

2   Press the '**area**' block to pop up its fields. You can press yellow fields to inspect and in some cases edit them. Try editing the Area block's **DispMode** field to include the horizontal trend (option **H_Trend**) in the available trend formats. Press the **Up** key to close the subfield menu and return to block field window.

3   From the block field view, press the **Down** key to see the block's **Name** and **Type**. Press **Down** again to inspect any incoming connections to this block. From here, pressing **Up** or **Down** returns you directly to the Function Block Manager window.

4   To inspect other blocks, without having to return to the Function Block Manager window, press the **Left** or the **Right** key to cycle round all the blocks in the application.
Pressing **Up** again returns you to the Function Block Manager window.

> *Note*   *If you want to edit a block name (and certain other parameters) you must first stop the application.*
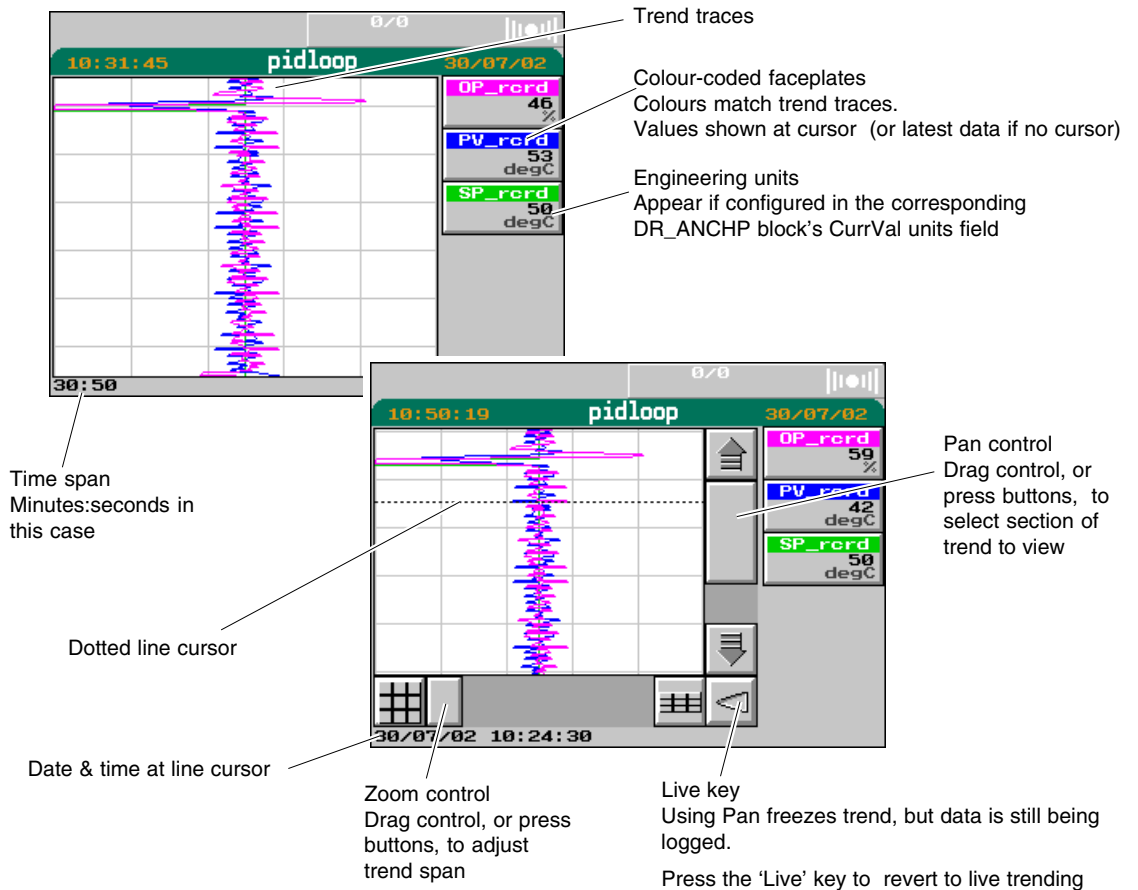
Figure 2-4   Vertical trend with faceplates

2.5 LOOKING AT ALARMS

In this section you take a quick look at how alarm conditions are displayed on the instrument's screen, and how to archive alarms and add notes to the alarm history.

For comprehensive details on alarms you should refer to the *Eycon-10/Eycon-20 Handbook,*Part No. HA029280.

2.5.1 The Alarm pane

Figure 2-3b showed the alarm pane at the top right corner of the screen. It is red, meaning that there is at least one active alarm present, in this case a *low absolute alarm* (set in the PID blocks *Alarms.LowAbs* field - Priority 1) because PV is below 20, the low absolute alarm threshold (set in the PID block's *LAA* field). The legend '**1/0**' means that there is currently **one** active alarm and **zero** unacknowledged alarms.

> *Note* *Alarms with priorities from 1 to 5 are 'self-acknowledging' — i.e. they are indicated only when the alarm state is active, and clear themselves when the alarm state is over. Alarms with priorities of 6 or greater are 'acknowledging' alarms. They continue to be indicated, even after the alarm state is over, until acknowledged by an operator.*

To see some activity in the alarm pane, try the following:

1   Get the detailed 'pid' function block faceplate on view, as shown in Figure 2-3. (Press **Menu**, **OVERVIEW**, and then the small 'pid' faceplate.)

2   Press the green **SL** readout ('**0.0**') and edit the local setpoint to '**50**' in the numeric pad popup. As the PV rises above 20 the red alarm pane flickers and when the low absolute alarm state has cleared, the pane becomes light grey and indicates '**0/0**'.

3   Now create some 'acknowleding' alarms. Press **Menu**, **SYSTEM**, **APPLN**, **FB MGR**. Select the **pid** block and edit its **Alarms** field. Set the **LowAbs** priority to '**6**', i.e. an *acknowledging* alarm, see note above. Press **Up** twice to return to the **FB Manager** screen.

4   Similarly, set the **area** block's **Area** alarm priority to say, '**10**', and the **pidloop** block's **Group** alarm to, '**15**'. Return to the function block faceplate (via **Menu**, **OVERVIEW** etc.).

5   Now reset SL to zero and watch the alarm pane. A confusing amount of activity is seen as PV oscillates its way down through the '20' threshold and the three alarms arise and clear in rapid succession.

> *Note* 'Flashing' *indicates unacknowledged alarm(s), and* 'red' *indicates active alarm(s).*

6   When PV has neared zero the alarm pane settles down to a flashing red/black showing '**3/3**', i.e. 3 *active* alarms and 3 *unacknowledged* alarms (as expected). The block name and alarm name of one of the alarms (the last to have occurred) is displayed in the pane.

2.5.2 The Alarm Summary screen

1   To see individual alarms, press the alarm pane, then **SUMMARY**. The **Alarm Summary** screen appears, listing the three current alarms with the latest at the top.

2   Now acknowledge one of the still-active alarms. Touch the '**pid LowAbs**' alarm anywhere over its first two columns. It highlights in yellow. Press the **Option** key to see a row of softkeys along the bottom of the screen for working with the Alarm Summary screen. Press **ACK** to acknowledge the **LowAbs** alarm, see Figure 2-5. This alarm item stops flashing (but remains red because the alarm condition still exists), and the alarm pane displays '**3/2**' and the name of the most recent unacknowledged alarm, the next one in the list.
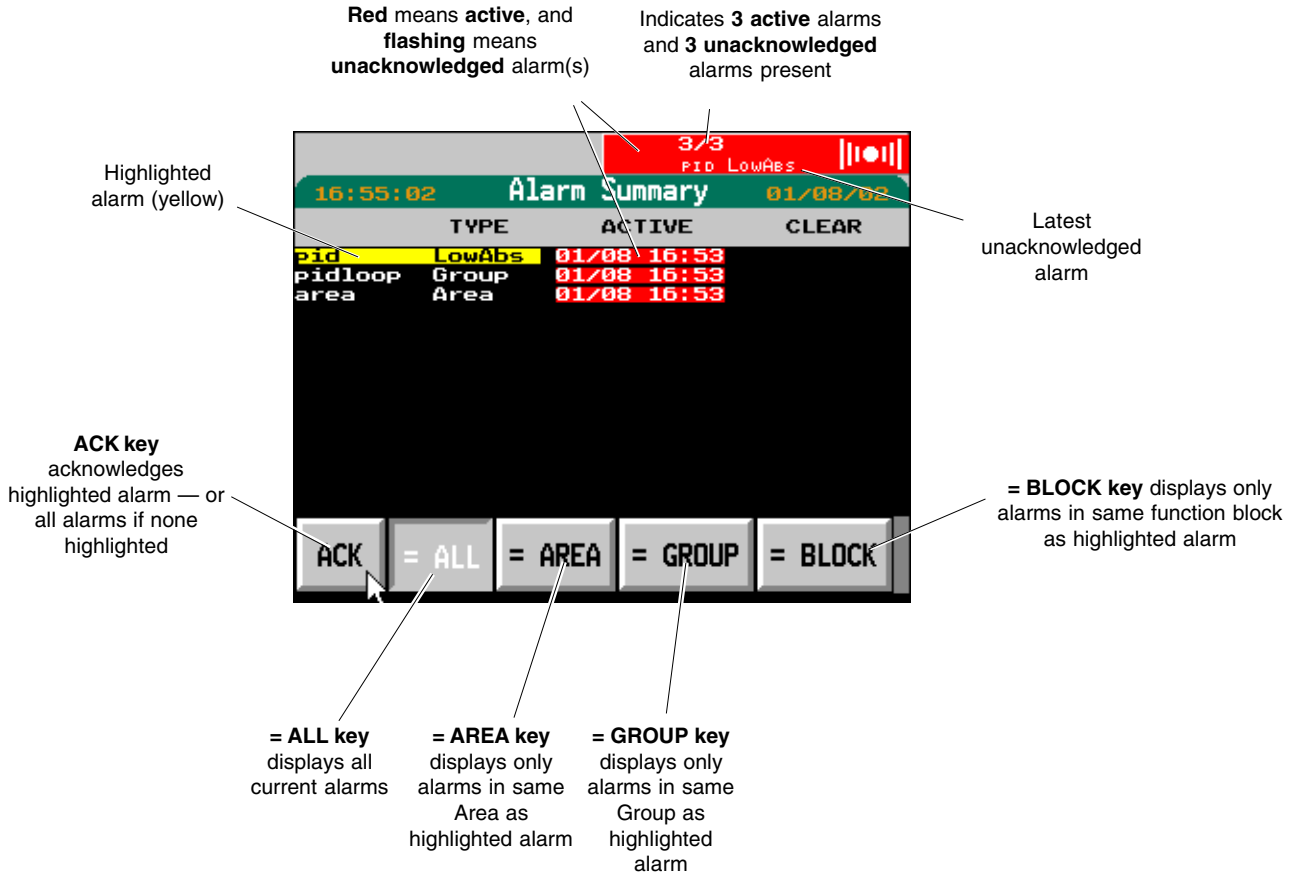
Figure 2-5   Alarm Summary screen

3   Acknowledge the remaining two unacknowledged alarms by simply pressing the **ACK** button, without highlighting any individual alarm. An **ACK ALL** dialog pops up asking for confirmation. Press **OK** to complete the acknowledgement process. All flashing stops and '**3/0**' appears in the alarm pane.

2.5.3 The Alarm History screen

1   To see a chronological list of all alarms (and events and messages) generated since the application started running, press the alarm pane, then **HISTORY**. The first page of the **Alarm History** screen appears, showing the newest alarms/events/messages. Press the **Left** and **Right** keys to access any other history pages, if appropriate.

> *Note*   *The history list usually runs to several pages and can store up to 250 or 500 items. When the history is full, new items cause the oldest items to be erased.*

2   Press the **Option** key to pop up one of two sets of Alarm History softkeys. (Press **Option** again to access the other softkeys, and once again to hide the softkeys.)

3   Highlight one of the history items, e.g. a '**pid LowAbs**' alarm and press the **= BLOCK** softkey. All alarms/ events/messages other than those associated with the 'pid' block are filtered out of the list. The **= ALL**, **= AREA**, and **= GROUP** keys work in an analogous way. Try them!

4   Try pressing the **= ALARMS** key. All items other than alarms are filtered out of the displayed history list. The **= MSGS** key works similarly for messages (but you probably won't have any messages!). Press **= ALL** to restore the full history list.

> *Note*   *Pressing the **Down** key toggles each history item between occupying one line and two lines. The second line displays the user ID current at the time of the entry.*

2.5.4 Archiving the alarm history

Now try archiving the alarm history to a USB bulk storage device. To do this, fit a USB bulk storage device into the instrument's USB port, and press the **ARCHIVE** softkey. The **Alarm Archive** dialog appears, where you can edit the default archive filename, YYMMDDHH, and choose a **Date Format**. You cannot edit the .ALH filename extension. Press **OK** to start archiving.

(See Chapter 4, *Data Logging*, Table 4-3 in this tutorial, for details of the available date formats.)

> *Note*   *You should switch off normal data logging while archiving the alarm history to avoid the possibility of losing logging data.*

2.5.5 Adding notes to the alarm history

If you want to add a short 'note' to the alarm history, do this. Press the alarm pane then **NOTE** to pop up the **Add Note** dialog. Press the yellow 16-character field and enter your message. Press **OK**. The note appears in the alarm history along with its data and time of entry.

CHAPTER 3 SETPOINT PROGRAMMING

This chapter shows you how to add *Setpoint Programming* functionality to the simple PID loop strategy you built in Chapter 2. You also test-run a simulation version of the strategy and try out some of the Instrument's Programmer features.

The main topics covered are:

■ Adding programming to the PID strategy

■ Creating a setpoint program

■ Testing the programmer strategy

The *Setpoint Programming* editor is a graphical programming tool that lets you create setpoint programs and recipes for the ICP (Industrial Control Package) setpoint programmer. It is part of the LINtools Engineering Studio suite of software for configuring LIN instruments.

The editor outputs a complete setpoint program in compact binary format as a single .UYS file, ready for direct downloading to the target ICP.

3.1 WHAT IS SETPOINT PROGRAMMING ?

Many applications need to vary temperature, or a process value, with time. Such applications need a control that varies a setpoint as a function of time.

The setpoint is varied by using a *setpoint program* that drives the setpoint according to the selected program. The program is stored as a series of 'ramp' and 'dwell' segments, as shown below.
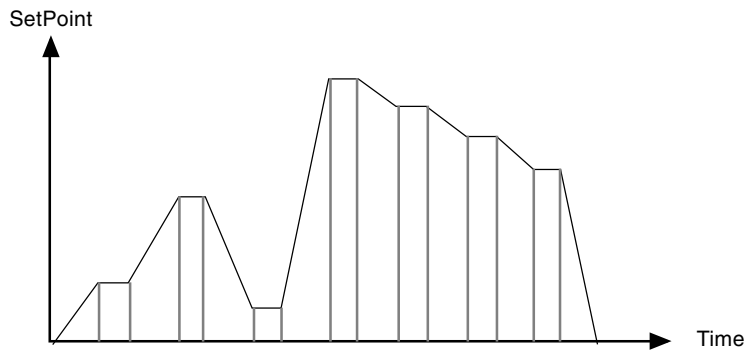
In each segment you can define the state of up to two outputs, each can be used to trigger external events.

A program is executed either, once, repeated a set number of times, or repeated continuously. If repeated a set number of times, then the number of cycles must be specified as part of the program.

> *Note   The list of segment types used in the Visual Supervisor may differ from the list available in the Setpoint Programme Editor*

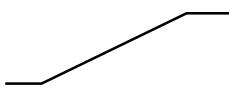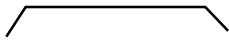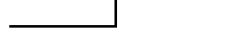| Segment Type	Graphic	Function
Ramp		**The setpoint ramps linearly,** from its current value to a new value, either at a set rate (called *ramp-rate programming*), or in a set time (called *time-to-target programming*). You must specify the ramp rate, or the ramp time, and the target setpoint, when creating or modifying a program.
Dwell		**The setpoint remains constant** for a specified period.
Step (Set to)		**The setpoint steps instantaneously** from its current value to a new value.
Servo to SP		The unit reads the current setpoint value, and sets the setpoint to that value (that is, it does not change it). Similar to 'Dwell' except that the instrument carries out the instruction automatically (without operator intervention). As there is no change, power output remains constant. May be used only in the first segment.
Servo to PV		The instrument reads the current process value and sets the setpoint to that value. Because the current process value normally differs from the current setpoint value, this option usually results in a change in the power consumption of the process.
	Note	*If the first segment is a Servo to PV or SP, the instrument assumes that it starts from an SP of 0.0. This is unlikely to be the actual SP or PV. Therefore the Preview profile displayed for the first segment will differ from the actual programmed profile. For the same reason, if the second segment is a ramp-at-rate, the segment duration in Preview will differ from the actual duration; and if the second segment is ramp-to-target then the slope in Preview will differ from the actual slope.*
End		A program either ends in this segment, or **repeats.** You specify which is the case when you create, or modify, a program (see the final topic in this chapter). When a program ends, the programmer is put into either, a continuous Dwell state with all outputs staying unchanged, or the Reset state.

Each Program can operate in the following states:- *Reset, Run, Hold, Holdback* and *End.*

State	Function
Reset	**In Reset,** the program is inactive, with the setpoint deteremined by the value set in the lower readout.
Run	**In Run,** the program is varies the setpoint according to the active program.
Hold	**In Hold,** the program is frozen at its current point. In this state you can make temporary changes to any program parameter, e.g. a target setpoint, a dwell time, or the time remaining in the current segment. These changes only remain effective until the program is reset and run again, when they are overwritten by the stored program values.
	Note When a program is running., you are not permitted to edit a 'called' program until it becomes active within that program.
Holdback	**In Holdback,** the program is waiting for the process to catch up, because the measured value has deviated from the setpoint by more than the preset amount and the program is in Hold.
End	**In End,** the program has completed.

3.2 ADDING PROGRAMMING TO THE PID STRATEGY

Figure 3-1 shows the PID loop strategy created in Chapter 2, but with extra function blocks added, at the bottom of the figure, to allow a setpoint program to be run.

SPP_RAMP
setpoint programmer
local ramp block
[Programmer Category]
*Allows local analogue setpoint
(generated by setpoint
program) to be ramped locally
in the Instrument.*

**DR_ANCHP data recording
analogue channel point blocks**
*Allow trending of database
floating-point values, and also use
of Instrument's Preplot facility.*

SPP_CTRL
**setpoint programmer
control block**
[Programmer Category]
*Accesses Instrument's status &
programming/scheduling
facilities. Must be run for full
programming functionality.*

SPP_DIG
setpoint programmer digital block
[Programmer Category]
*Allows local digital setpoints generated by
setpoint program to be wired locally (to
DR_DGCHP blocks), or to remote target
instruments (e.g. T2550).*

**DR_DGCHP
data recording digital channel
point block**
[Recorder Category]
*Allow trending of database digitals,
and also their inclusion in
Instrument's Preplot facility.*

GROUP block
*Organises the two DR_ANCHP &
linked SPP_RAMP blocks, the
SPP_DIG & linked DR_DGCHP
blocks, into a 'preplot' group —
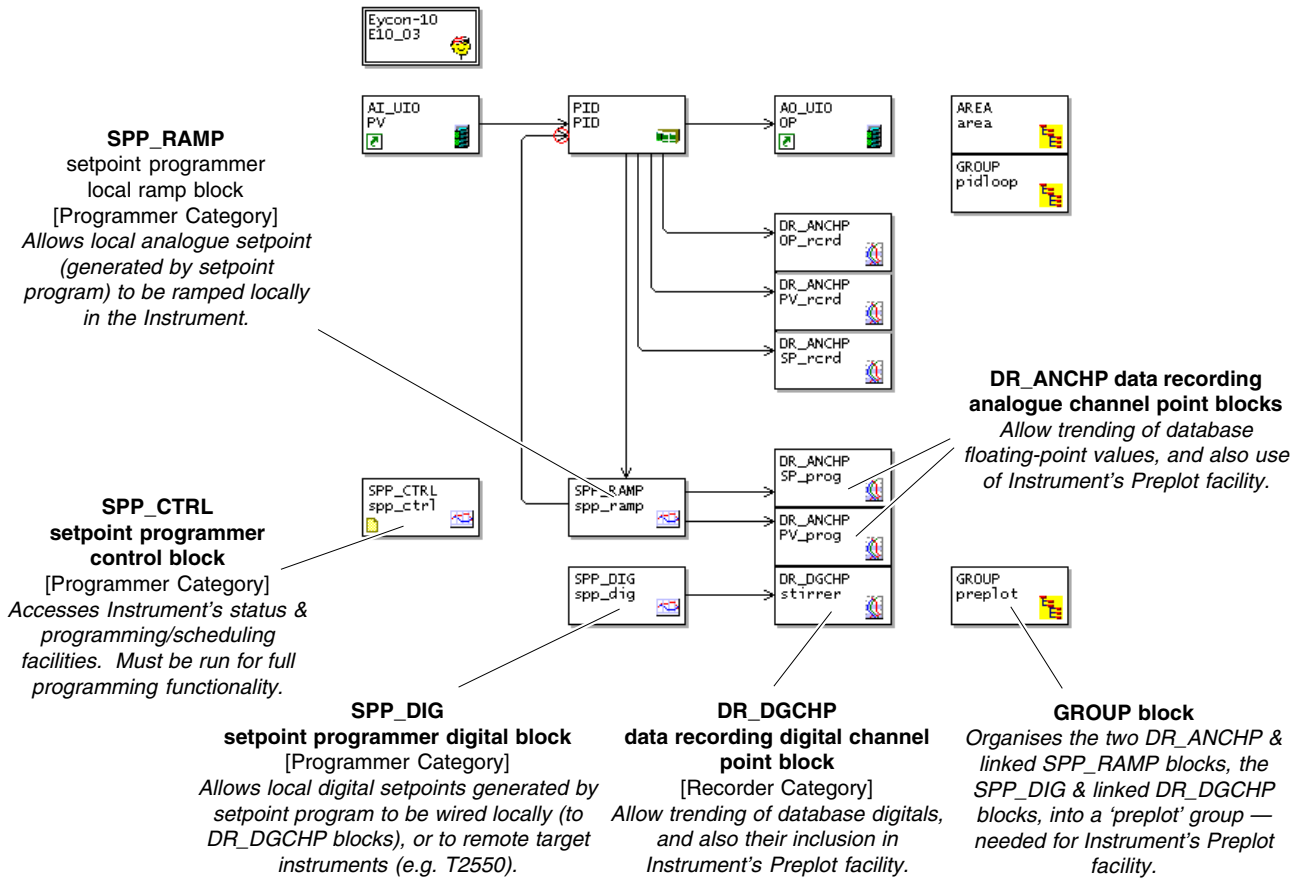needed for Instrument's Preplot
facility.*

Figure 3-1   PID loop strategy with added Setpoint Programming

3.2.1 Building the programmer strategy

1   In the LINtools Engineering Studio, locate the required blocks as shown in Figure 3-1. The quickest way to do this is to open up the existing PID loop strategy as your starting-point — by double-clicking its icon in Explorer view, then simply add the seven extra 'programming' blocks.

> *Note*   *If you need help with LINtools Engineering Studio, Chapter 2 explained how to access and use the configurator, and the context-sensitive Help system (press <F1>) provides instructions on every operation.*

2   Wire up the new blocks according to Table 3-1, which lists all the strategy connections for completeness.

> *Note*   *In the table, the wires below the dashed line are new, so you need only add these if you are starting from the PID loop strategy created in Chapter 2.*

Source*	Destination*	Function
(AI_UIO) PV.PV	(PID) pid.PV	process variable PV from the remote input module into the PID block
(PID) pid.OP	(AO_UIO) OP.OP	PID control output OP to the remote output module
(PID) pid.OP	(DR_ANCHP) OP_rcrd.CurrVal	PID control output OP into the OP_rcrd (DR_ANCHP) block to enable data recording onto Instrument's local flash storage
(PID) pid.PV	(DR_ANCHP) PV_rcrd.CurrVal	PID process variable PV into the PV_rcrd block to enable data recording
(PID) pid.SP	(DR_ANCHP) SP_rcrd.CurrVal	PID setpoint SP into the SP_rcrd block to enable data recording
(PID) pid.PV	(SPP_RAMP) spp_ramp.PV	PID process variable PV into the spp_ramp block to include PV in the 'preplot' group
(SPP_RAMP) spp_ramp.Out	(PID) pid.RemoteSP	setpoint generated by the locally-running setpoint program into the PID block's remote setpoint input
(SPP_RAMP) spp_ramp.Out	(DR_ANCHP) SP_prog.CurrVal	setpoint generated by the locally-running setpoint program into DR_ANCHP block to enable trending & Instrument's 'Preplot' facility
(SPP_RAMP) spp_ramp.PV	(DR_ANCHP) PV_prog.CurrVal	PID process variable PV (sourced from the PID block) into DR_ANCHP block to enable trending & Instrument's 'Preplot' facility
(SPP_DIG) spp_dig.Out.Bit0	(DR_DGCHP) stirrer.CurrVal	digital setpoint generated by the locally-running setpoint program into DR_DGCHP block to enable trending & Instrument's 'Preplot' facility

> *Note*   *(Block type) blockname.field.subfield format, referring to Figure 3-1*

Table 3-1   Strategy wiring data

> *Note*   *The Configuration (Header) block and the AREA, GROUP, and SPP_CTRL blocks do not need to be wired. They are associated with other blocks by setting field values (see next).*

3.2.2 Editing the block fields

Table 3-2 lists the fields that should, as a minimum, be edited from their default values. For completeness, all such fields are shown in the table — including those edited in the original PID loop strategy. Note that you must also edit the **Name** fields in all blocks, according to Figure 3-1 if you wish.

> *Note   In the table, the fields below the dashed line are new, so you need only edit these if you are starting from the PID loop strategy created in Chapter 2. You can get help with editing function block fields in LINtools by pressing <F1> while a text cursor appears in a field.*

Field Value*	Function
(PID) pid.TI = **5.00**	Sets the PID integral time constant to 5 seconds, to improve control
(PID) pid.LAA = **20.00**	Sets the pid block's PV output 'low absolute' alarm at 20%
(PID) pid.Alarms.LowAbs = **1**	Enables the 'low absolute' alarm on the pid block's PV output
(AREA) area.Id = **1**	Allocates area number (only Area '1' currently implemented). No OVERVIEW appears in the Instrument unless an area number has been allocated
(AREA) area.Group1 = **pidloop**	Assigns the 'pidloop' GROUP block to this area. This specifies that pidloop will appear in the OVERVIEW and enables data recording for this group
(GROUP) pidloop.Update = **10**	Specifies the recording sample rate (seconds) for the 'pidloop' group. The default rate of zero disables recording
(GROUP) pidloop.Disp1 = **pid**	Specifies the PID block 'pid' as the source of the data that will be displayed in the faceplate and trends for the 'pidloop' group
(GROUP) pidloop.Chan1 = **PV_rcrd**	Assigns the recorder channel block 'PV_rcrd' to channel 1 of the 'pidloop' recording group
(GROUP) pidloop.Chan2 = **SP_rcrd**	Assigns the recorder channel block 'SP_rcrd' to channel 2 of the 'pidloop' recording group
(GROUP) pidloop.Chan3 = **OP_rcrd**	Assigns the recorder channel block 'OP_rcrd' to channel 3 of the 'pidloop' recording group
(AREA) area.Group2 = **preplot**	Assigns the 'preplot' GROUP block to this area. This specifies that preplot will appear in the OVERVIEW and enables data recording for this group
(GROUP) preplot.Update = **10**	Specifies the recording sample rate (seconds) for the 'preplot' group. The default rate of zero disables recording
(GROUP) preplot.Disp1 = **spp_ramp**	Specifies the SPP_RAMP block 'spp_ramp' as the first source of data that will be displayed in the faceplate and trends for the 'preplot' group
(GROUP) preplot.Disp2 = **spp_dig**	Specifies the SPP_DIG block 'spp_dig' as the second source of data that will be displayed in the faceplate and trends for the 'preplot' group
(GROUP) preplot.Chan1 = **PV_prog**	Assigns the recorder channel block 'PV_prog' to channel 1 of the 'preplot' recording group
(GROUP) preplot.Chan2 = **stirrer**	Assigns the recorder channel block 'stirrer' to channel 2 of the 'preplot' recording group
(GROUP) preplot.Chan3 = **SP_prog**	Assigns the recorder channel block 'SP_prog' to channel 3 of the 'preplot' recording group
(DR_ANCHP) SP_prog.ZoneA_LO = **11**	Specifies '11' (%) as the bottom of the chart zone or band occupied by all SP plots. The default ZoneA_HI of 100% specifies the top of SP's plotting zone.
(DR_ANCHP) PV_prog.ZoneA_LO = **11**	Specifies '11' (%) as the bottom of the chart zone occupied by the PV plots. The top of the zone = 100% (default).
(DR_DGCHP) stirrer.ZoneA_HI = **10**	Specifies '10' (%) as the top of the chart zone occupied by the 'stirrer' digital plots. The bottom of the zone = 0% (default).
(SPP_CTRL) RqNxtPrg = **program1**	The yellow page icon shows an additional file is required for this block to function correctly. Entering the filename allows LINtools to offer the option to create this file using the defined filename via the context menu.

> *Note   *(Block type) blockname.field.subfield format, referring to Figure 3-2*

Table 3-2   Values for non-default block fields

3.2.3 Using GROUP block channels to organise the preplot display

An important function of the GROUP block's *Chan1* to *Chan16* parameters is to organise setpoints for display in the Instrument's preplot view, via DR_ANCHP and DR_DGCHP recorder channel blocks. Figure 3-2 shows a preplot display with one digital and two analogue setpoints.

> *Note*   **Zoning** *has been used to confine the three setpoints to their own horizontal (in this case) bands on the 'chart'. Table 3-2 gave you the zone specifications for the current setpoint program application. You will see what effect this has later in the tutorial.*
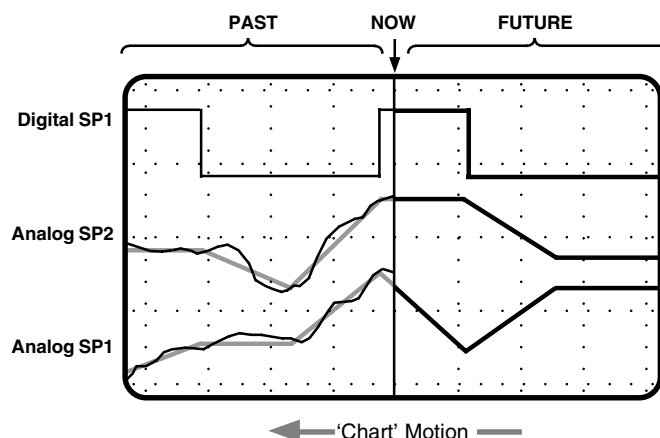
Figure 3-2   Example Instrument 'preplot' display

For a valid preplot display, the GROUP block channels must be associated in strict numerical order with the setpoints generated by the running setpoint program — which you will create in the next section. This is done as follows:

1   First, the ***analogue PVs*** must be associated in numerical order as defined by the setpoint program. That is, Analogue PV1 (resulting from Analogue Setpoint 1) must be associated with *Chan1*, Analogue PV2 with *Chan2*, etc., until all the analogue PVs have been dealt with.

■   In the 'preplot' group there is only one analogue PV resulting from the single analogue setpoint generated by the setpoint program. So the DR_ANCHP block **PV_prog** is allocated to the 'preplot' GROUP block's *Chan1* field.

2   Then, any ***digital setpoints*** must be associated in program order with the next of the GROUP block's channels.

■   The 'preplot' group has one digital setpoint. So the DR_DGCHP block **stirrer** is allocated to the 'preplot' GROUP block's *Chan2* field.

3   When all the analogue PVs and digitals have been dealt with, the ***analogue SPs*** are associated with the remaining channels in order.

■   In the 'preplot' group, the DR_ANCHP block **SP_prog** is therefore associated with *Chan3* to complete the set.

3.3 CREATING A SETPOINT PROGRAM

This section tells you how to create a very simple program to provide one analogue and one digital setpoint for the strategy, using the **Setpoint Program Editor**. The analogue setpoint will start off holding at zero in a 'ZeroOP' segment, then ramp up to 100% ('Heat Up' segment), then hold at 100% ('Soak' segment), and finally ramp back to zero ('Cool Down' segment). The digital setpoint could be used to switch a 'stirrer' on (digital HIGH) and off (digital LOW). The stirrer is to operate during the 'Heat Up' and 'Cool Down' segments only.

> *Note   Full details on using this editor are given in the online* Setpoint Program Editor Handbook, *Part No. RM 261 134 U005. The present section provides guidance only on what you need to do in this particular example.*
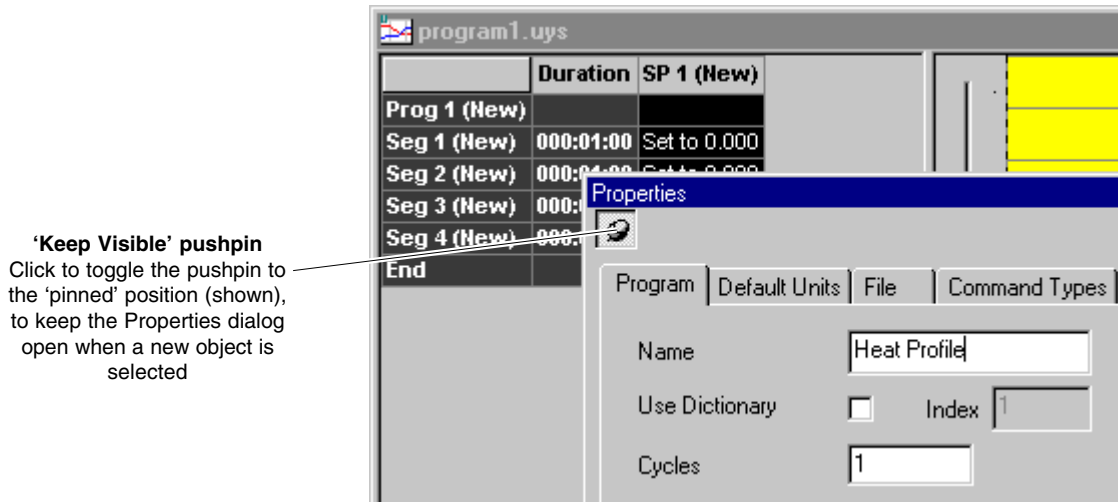

'**Keep Visible' pushpin**
Click to toggle the pushpin to the 'pinned' position (shown), to keep the Properties dialog open when a new object is selected

Figure 3-3   Properties dialog for 'Prog 1 (New)'

Each application used to create a related file type can be launched from within LINtools. Any function block that displays a Yellow page icon requires an additional file before for it will function correctly.

1   The filename, **program1**, in *SPP_CTRL.RqNxtPrg*. does not need a .uys file extension reference because the block in LINtools knows that this file MUST be a Setpoint Program. Right-click this field to reveal the context menu and select **Open Setpoint Program (.uys)**. This will launch the application.

> *Note   Additional Setpoint Program files are created by accessing the Explorer folder where these tutorial files are being stored, then right-click on an empty area of the window to pop up a context menu. Select **New** > **Setpoint Program**. A new setpoint file (**.uys**) appears. Rename the file, e.g. as '**program1.uys**'.*

2   When the **Setpoint Program Editor** window is initially displayed it will contain a **New SPE File** dialog. Leave all its fields at their default values, except for the **Initial number of segments per program** field, which you should set to **4**. Hit **OK** to close the dialog and open a setpoint file window, ready for configuration.

3   Double-click the default **Prog 1 (New)** cell near the top-left corner of the window to pop up the **Properties dialog** for this object (Figure 3-3). On the **Program** page of the dialog, edit the program **Name** to '**Heat Profile**' and hit the Computer's <Return> key to enter the data. The dialog should stay open if you have set the 'Keep Visible' pushpin to 'pinned' (as in the figure).

4   Now single-click the **SP 1 (New)** cell. The Properties dialog changes to a '**Setpoint/Holdback**' dialog. Edit the setpoint **Name** field to '**profile1**', and the **Hardware reference** field to '**spp_ramp**'. This is the name of the SPP_RAMP block that transfers the generated setpoint to the strategy (see Figure 3-1).

5   Click the **Holdback** tab and enter a **Holdback Value** of '20'. In the **Holdback Mode** pulldown, select 'Band' from the list. Hit <Return> to enter the data.

> *Note*   *'Holdback' is a way to extend the duration of a program segment to allow a measured process variable PV to 'catch up' with its setpoint SP, if it has deviated from it by more than a certain amount. If at any time during the segment a PV differs from its SP by more than the specified* **Holdback Value***, in a direction specified by* **Holdback Mode***, then the program enters a 'hold' state until the condition clears. In this state, all SP ramps in the segment are 'frozen' so that they stay in step. Setting* Holdback Mode *to 'Band' triggers holdback action when any PV is above or below its SP by more than the Holdback Value — 20% in this case.*
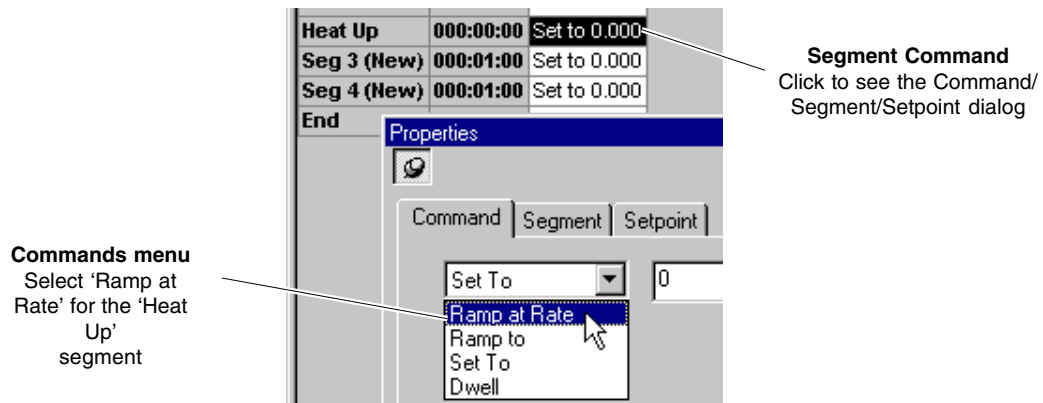

Figure 3-4   Configuring the 'Heat Up' segment

6   Next, single-click the **Seg 1 (New)** cell. The Properties dialog changes to a '**Segment**' dialog, where you can edit the **Profile Name** field to '**ZeroOP**'. Leave the **Duration** field at its default value of 1 minute, and hit <Return> to enter the data.

7   Access the **Seg 2 (New)** cell Properties dialog. Edit **Profile Name** to '**Heat Up**', leaving **Duration** at its default. Click the default **Set to 0.000** cell in this row of the table to access the **Command/Segment/Setpoint** set of properties pages (Figure 3-4).

8   Select **Ramp at Rate** from the pulldown menu of available commands, and enter '**0.25**' in the **per sec** box, and '**100**' in the **to** box. This will cause the 'Heat Up' segment to ramp the setpoint up to 100% at the rate of 0.25% per second.

9   Continue specifying the segment names, durations, and commands according to the data given in Table 3-3.

> *Note*   *Notice how the shape of the setpoint profile is building up in the setpoint file window, according to your specifications so far.*

Segment	Segment Name	Duration (mins:secs)	Function
1	ZeroOP	01:00	Set to 0.000 (default)
2	Heat Up	[06:40]*	Ramp at Rate, 0.25 per sec, to 100.00
3	Soak	03:00	Dwell
4	Cool Down	[03:20]*	Ramp at Rate, 0.50 per sec, to 0.00

> *Note*   **These values are automatically calculated by the editor, and should not be entered.*

Table 3-3   Specifications for the 'Heat Profile' segments

10   Now add the digital setpoint. Click the **Insert Digital Setpoint** toolbutton ⊞ (or press **Edit > Insert > Digital Setpoint**). A new digital setpoint faceplate appears at the right of the editor window.

11   In the **Properties** dialog **Setpoint** page, type 'Stirrer' in the **Name** field, and enter a **Hardware Reference** of 'spp_dig.Out.Bit0'. (This is the LIN block field that will collect the digital setpoint from the running setpoint program and pass it into the DR_DGCHP block 'stirrer' for trending and preplot.) Hit <return> to enter the data.

12   In the program pane at the left of the editor window, click the '**Dwell @OFF**' cell in the **Heat Up** segment (row) of the **Stirrer** profile (column) to see its properties. In the **Command** page of the **Properties** dialog, pull down the menu and select 'Set To', then click the 'On' radio button. The digital setpoint steps up to 'ON' in the chart pane.

13   Now click the '**Dwell @ON**' cell in the **Soak** segment of the **Stirrer** profile. In the **Command** page, select 'Set To' 'Off'. The setpoint in the chart returns to 'OFF.

14   Lastly, set the digital setpoint to be 'On' in the **Cool Down** segment.
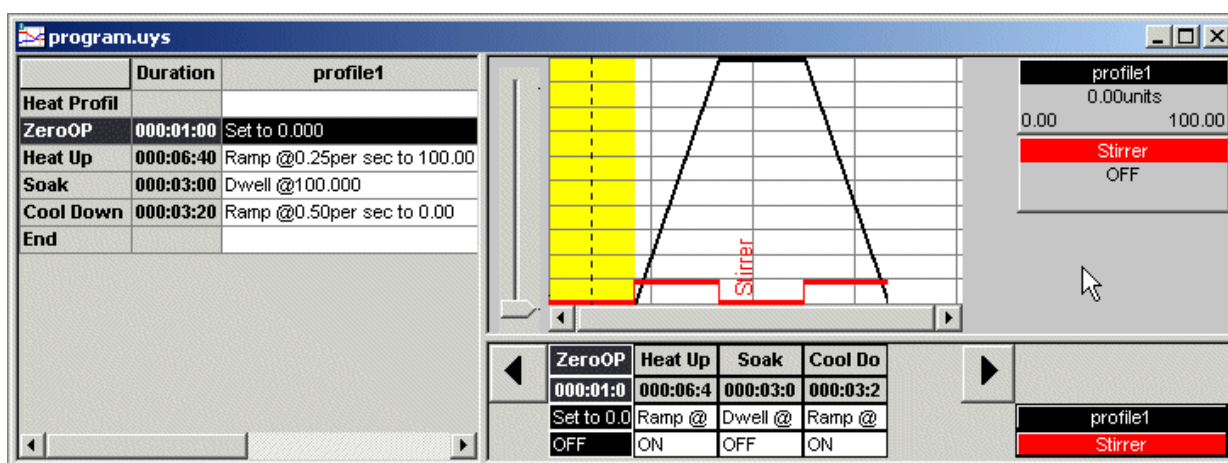
Figure 3-5   The completed 'Heat Profile' setpoint program

15   Save the setpoint file by clicking **File > Save** (or the **Save File** icon in the tool bar). Your completed setpoint file window should resemble Figure 3-5.

> *Note   This graphical view of your setpoint program resembles what you will see when you run the application in a Instrument later on in this chapter, and view its PREVIEW and PRE-PLOT screens. However, the instrument generates its own displays from the setpoint data which are completely independent of the setpoint program displays and, in particular, may have different colour schemes.*

3.4 TESTING THE PROGRAMMER STRATEGY

As for the PID loop in Chapter 2, you can quickly test out the programmer strategy using a SIM block to simulate plant behaviour. Figure 3-6 shows the slightly modified strategy.

3.4.1 Modifying the programmer strategy

1   Delete the wire between the AI_UIO 'PV' block and the PID block.

2   Place a SIM simulation block on the worksheet, as shown in Figure 3-6.

3   Wire the **SIM.OP** field into the **PID.PV** input, and **PID.OP** into **SIM.PV**. This forms a closed loop, indicated by the small red circle.

4   Name the SIM block, and edit both its **NoiseMax** and **Lag1** fields to **25.0**.

5   Save the modified programmer strategy (under a different name).
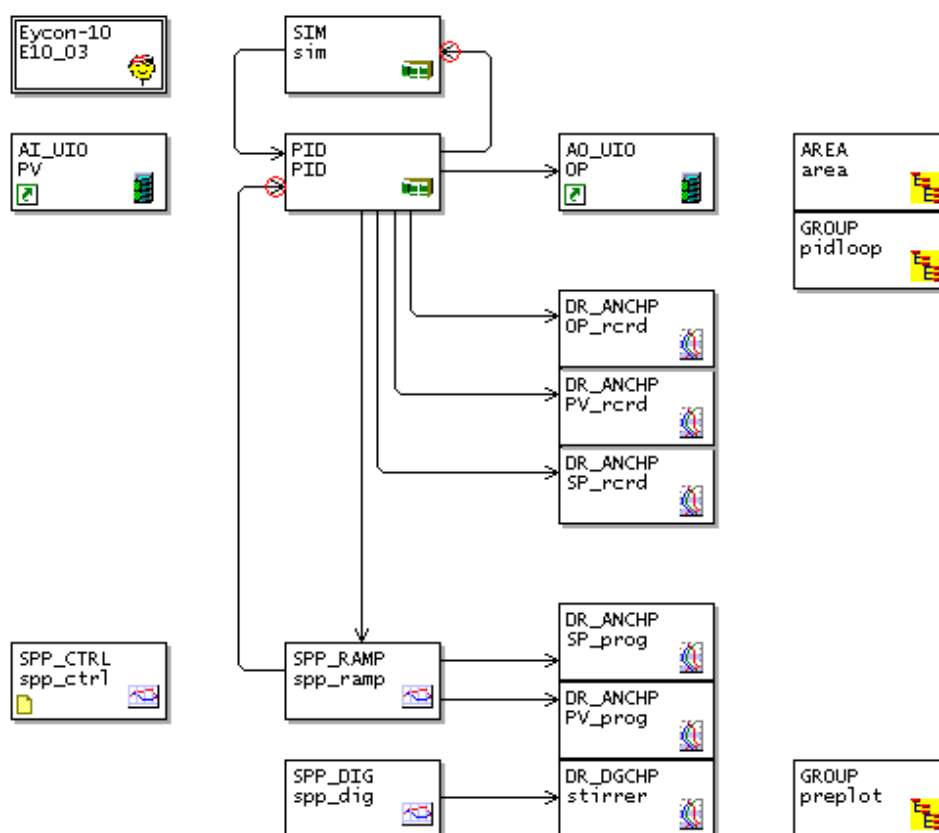
Figure 3-6   Modified programmer strategy for testing

3.4.2 Running the application

To run the programmer simulation strategy and try out some of the Instrument's programmer features, do the following:

1   Copy the saved modified 'simulation' strategy file to the Instrument's E: drive. Only the **.dbf** file is required by the instrument — the other two files that appear when you save a strategy in LINtools (**.grf** and **.dtf**) are not used in the instrument.

> *Note   If you need help with downloading, see Chapter 1,* Getting started.

2   Also download the setpoint program file, e.g. **program1.uys**, that you configured in the previous section.

3   In the instrument, load and run the 'simulation' programmer application. The area overview appears, containing the 'pidloop' and 'preplot' group faceplates that you specified in the AREA block.

■   If you press the 'pidloop' faceplate you will see the 'pid' faceplate that you explored in Chapter 2, *Running the application*. (You can get back to the area overview at any time by pressing the **Menu** key, then **OVERVIEW**.)

■   Pressing the 'preplot' faceplate accesses faceplates representing the SPP_RAMP and SPP_DIG blocks that you allocated to the 'preplot' group. Press these to explore further.

Note   *If you need reminding, Chapter 1,* Getting started, *describes how to select and run applications.*

Program pane
Shows the current program status, expected completion time, and name of segment currently executing.
[Press to access the Programmer menu]

Analogue setpoint faceplate
Represents the 'profile1' analogue setpoint. The upper bold figure shows the current process variable value, and the lower feint figure shows the current setpoint value.

Digital setpoint faceplate
Represents the 'Stirrer' digital setpoint. ('OPEN' equals 'OFF')

'Now' line
Marks the current moment in the preplot view — 00:00. The past is to the left of this line, the future to the right.

Programmed setpoints
'Preplot' of the setpoint profiles contained in the running setpoint program ('Heat Profile'). Note the 'zoning' of the two traces
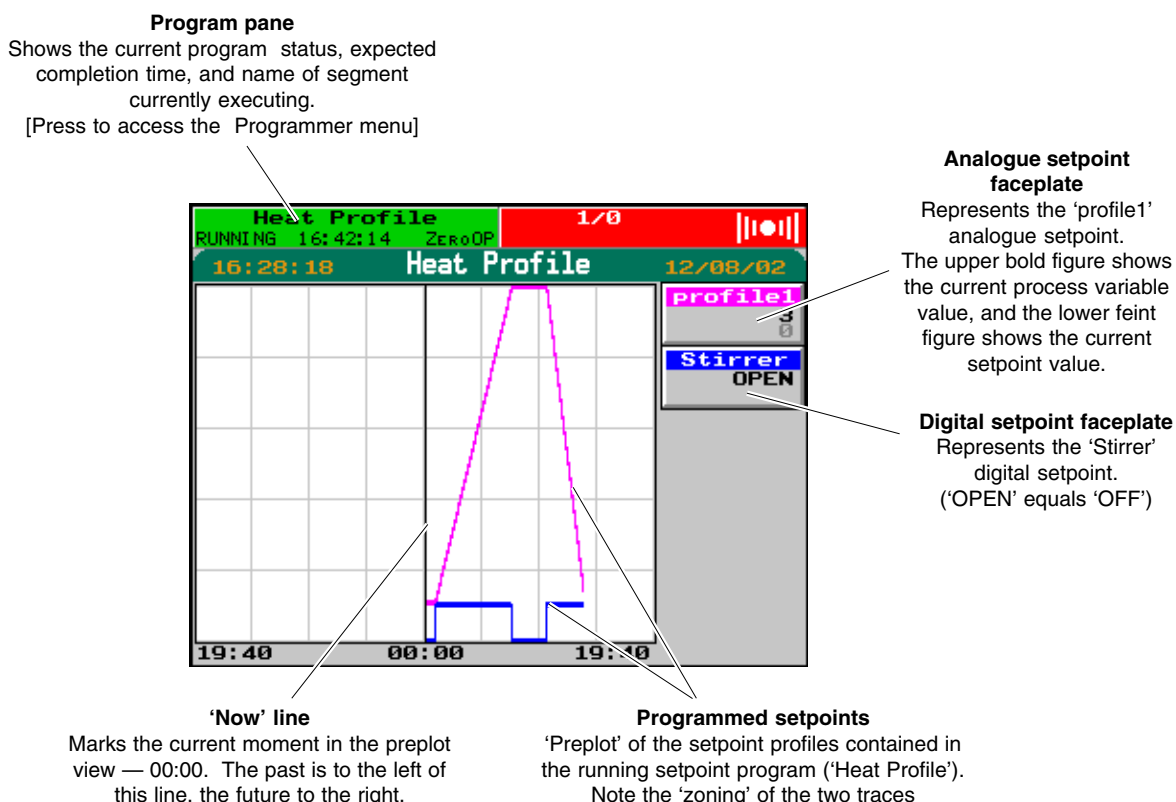
Figure 3-7   Preplot view of 'Heat Profile' (ZeroOP segment)

4   Press the pane at the top left of the screen (labelled 'RESET') to pop up the **Programmer** menu. (You can also access this via the **Menu** key, then **PROGRAMMER**.) Press **PROGRAMS** to see the **Load/Save Program** window, then **File Name: ????????** for a list of setpoint program files. Select and enter the file you downloaded in step 2 ('PROGRAM1'), then **LOAD** it to RAM.

5   In the **Programmer** menu, press **PREVIEW** to see a preview of the **Heat Profile** setpoint program. You can move the dotted cursor over the plot to see values and times, and also the name of each segment displayed.

6   Access the **Programmer** menu again, and press **RUN** to start the setpoint program. Then press **PRE-PLOT** to see the 'Heat Profile' preplot (Figure 3-7). To the left of the central vertical 'now' line is a realtime plot of PV and the programmed SP. To the right is that part of the programmed setpoint still to be applied. The preplot makes it very clear how closely (or not) PV is following SP, and what to expect next.

7   Watch the progress of the preplot for a while. You will see after a minute or so that something is wrong. The actual PV is not following the programmed setpoint! The reason is that *remote mode* has not yet been enabled in the PID block, and so the remotely-generated setpoint is being ignored.

> *Note*   *You will also see the* holdback *action you configured. The (green) programmed setpoint initially rises at the expected rate but soon starts to flatten out to a horizontal line. The analogue setpoint freezes when it deviates by 20 units from the PV-value — which at the moment is remaining on the zero line. The digital setpoint 'Stirrer' also freezes CLOSED. This state of affairs continues, with the 'Heat Up' segment extending indefinitely, until PV can follow its programmed SP.*

8   To put this right, press the **Menu** softkey, then **SYSTEM**, **APPLN**, **FB MGR** to access the **FB Manager** window. Press the '**pid**' block box to see the PID block parameters, scroll down to the **SelMode** field, press the yellow value pop up the **SelMode** menu. Set **SelRem** to TRUE to select remote mode and press return. Repeat for **EnaRem** to enable remote mode. Finally, press the program pane, then **PRE-PLOT**, and watch the progress. PV now shoots up rapidly to follow the programmed SP, which is released from holdback mode and resumes its intended path — see Figure 3-8.

> *Note*   *If you need reminding, Chapter 2,* Inspecting LIN function blocks, *gives more information on inspecting and editing block fields.*
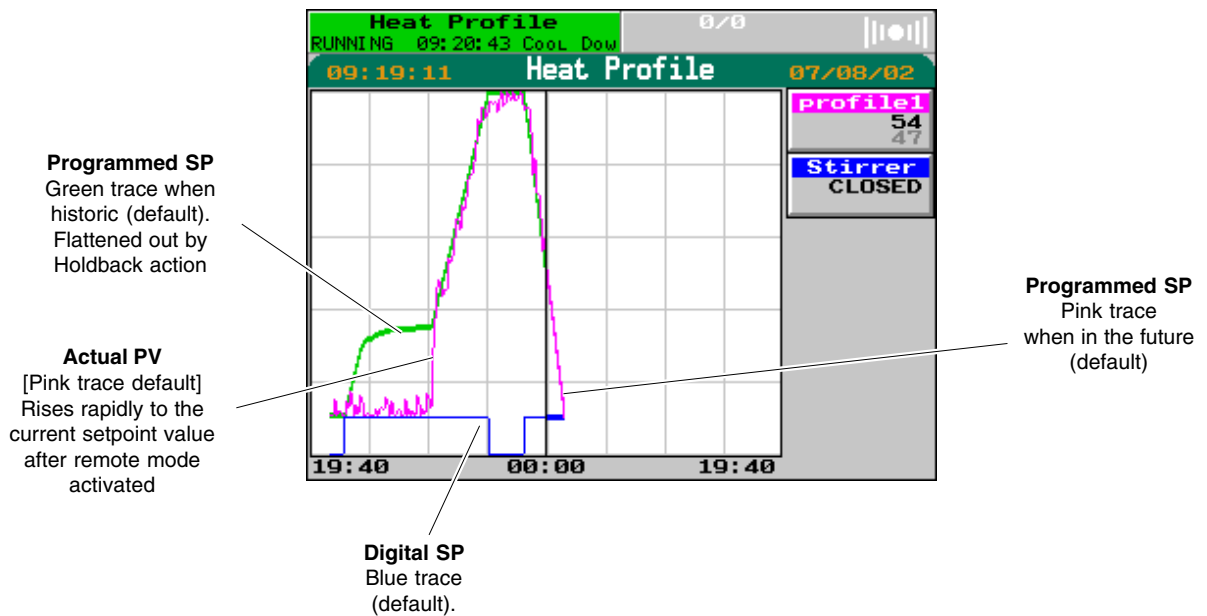

Figure 3-8   Preplot view of 'Heat Profile' (Cool Down segment)

3.4.3 Exploring the Programmer menu

Try using some of the available programmer functions:

1   While the program is running, press the program pane to access the **Programmer** window, then **MONITOR** to see the detailed status of the current program. You can hold or abort the program from this window. Try pressing **HOLD** and watch the progress in the preplot view, the setpoint holds at its current value until you press **RUN**, or **ABORT** the program.

2   The **SCHEDULE** button lets you run a program at some future date and time, a specified number of times. Press it to access the **Schedule Program** window, where you can select a program file, start date and start time, and the number of iterations required. Press **ACCEPT** to enter the schedule and see a summary. (Leaving the date and time fields blank executes the schedule as soon as any currently-running program has finished.)

> *Note   If a program is already scheduled, pressing **SCHEDULE** accesses the summary screen directly.*

3   The **EDIT** button lets you edit the loaded setpoint progam within the instrument. Press **EDIT** to see the '**Heat Profile**' edit window (in this case). Pressing any of the yellowed fields pops up an edit window where you can edit program parameters. Editable fields are yellowed. Press them to pop up applicable keypads for entering the new values, or selecting from picklists. You can also delete, insert, and rename segments.

> *Note   You cannot edit a running program. If the program is 'held', you can edit some of the parameters, e.g. ramp rates and target values, but for full editability you must abort the program or wait for it to reach the 'idle' state.*

4   When the program has finished running, try doing some edits. Press the idle program pane, then **EDIT** in the **Programmer** window. In the 'Stirrer' profile's 'ZeroOP' segment, press the 'OPEN' cell to pop up its edit dialog. Edit the yellow 'OPEN' field to 'CLOSED' and press **DONE** to apply the edit.

5   Press the program pane, then **PREVIEW**, to inspect the result. The (blue) digital setpoint is now CLOSED during the 'ZeroOP' segment.

6   Return to the **EDIT** window and press the yellow 'Soak' segment cell in the SP row of the table. Edit the **Duration** field to '00:00:30', i.e. 30 seconds soak time. Press **DONE**. Inspect the result in the **PREVIEW** window, and note that the 'Soak' segment has shortened to your new specification.

7   If you want to save your edited program, return to the edit window and press the **Option** softkey below the screen. Press the **SAVE AS** key and enter a new filename in the **File Name** field. Press **OK** to carry out the save.

> *Note   For a more complete list of programmer functions, please refer to the* Eycon-10/Eycon-20 Handbook, *Part no. HA 029 280.*

Intentionally left blank

CHAPTER 4 DATA LOGGING

This chapter shows you how to add *Data Logging* functionality to the setpoint programming strategy you built in Chapter 3. You test-run simulations of the strategy and try out some of Instrument's logging features.

Note *Data Logging can be used in many applications, setpoint programming is chosen here as just one (convenient) example.*

The main sections are:

- ■   Adding data logging to the programming strategy
- ■   Testing the data logging strategy
- ■   Adding a second log group
- ■   Charting logged data

4.1 WHAT IS DATA LOGGING ?

Data Logging, or *Archiving*, is a method of automatically storing incoming recorded data to an archive device, for Performance, Quality Control, and Audit Record purposes. Within the Visual Supervisor incoming data is logged to a 45MB internal flash memory directory, H: drive and saved in .asc (ASCII) or .uhh files (recommended).

Note *.uhh files are recommended and can only be read using Review software. These .uhh files are secure data logging files, appropriate for any application, but are 21 CFR pt 11 application compliant.*

Recording data is the process of saving the history of a set of data values in the instrument's non-volatile memory. The data can survive a power outage, and can be replayed on the instrument.

The recorded analogue or digital values stored in the local flash memoy, archive device, are displayed on Video Chart Recorders. An archive device providing a secure storage area for files holding the logged data. This can be internal to the Instrument, or via the relevant hardware configuration.

4.2 ADDING DATA LOGGING TO THE PROGRAMMING STRATEGY

Figure 4-2 shows the setpoint programming strategy created in Chapter 3, but with three extra function blocks added to allow data logging to be performed.
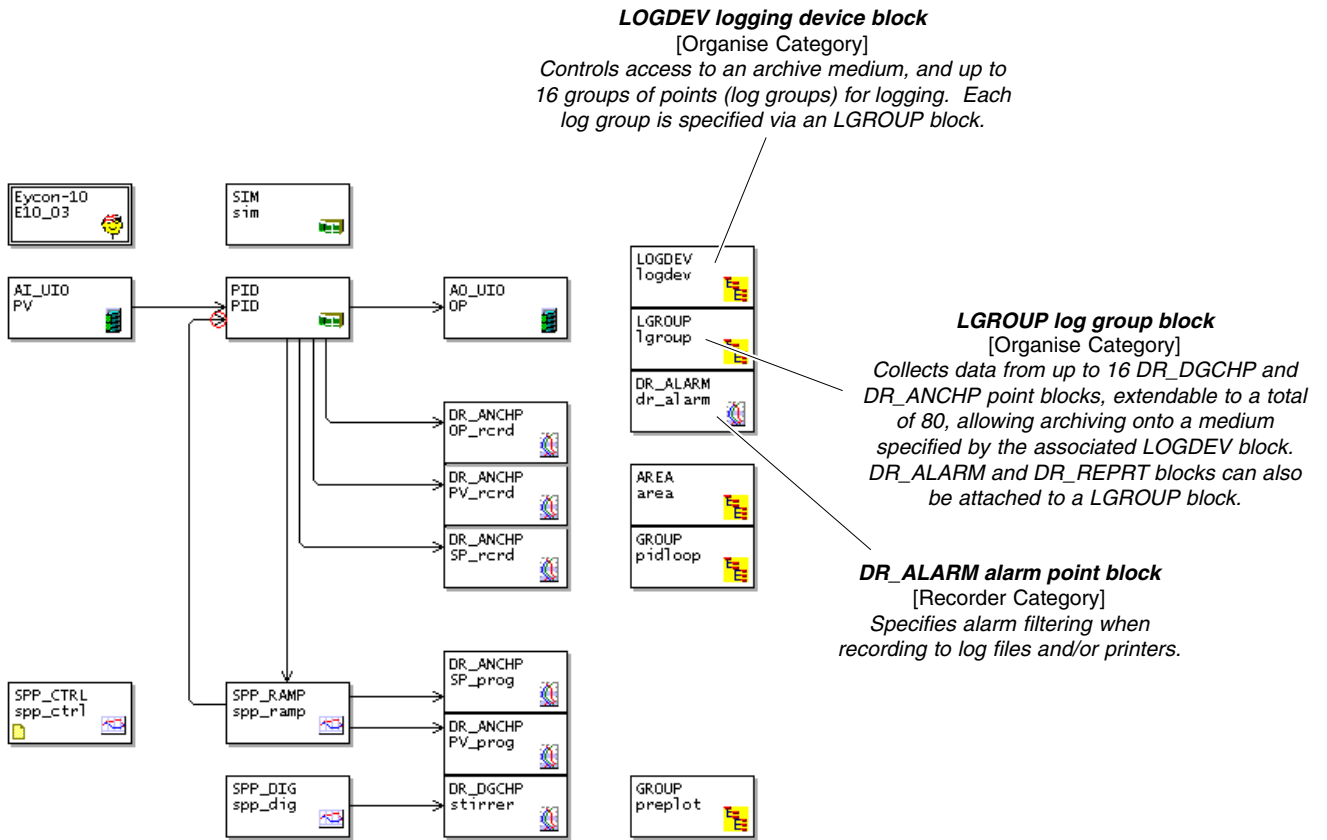

LOGDEV logging device block
[Organise Category]
*Controls access to an archive medium, and up to
16 groups of points (log groups) for logging. Each
log group is specified via an LGROUP block.*

LGROUP log group block
[Organise Category]
*Collects data from up to 16 DR_DGCHP and
DR_ANCHP point blocks, extendable to a total
of 80, allowing archiving onto a medium
specified by the associated LOGDEV block.
DR_ALARM and DR_REPRT blocks can also
be attached to a LGROUP block.*

DR_ALARM alarm point block
[Recorder Category]
*Specifies alarm filtering when
recording to log files and/or printers.*

Figure 4-2   Setpoint programming strategy with added data logging

4.2.1 Building the data logging strategy

1   In the LINtools database configurator, locate the required blocks as shown in Figure 4-2. The quickest way to do this is to open up the existing setpoint programming strategy as your starting-point, by double-clicking its icon in Explorer view, then simply add the three extra 'logging' blocks (LOGDEV, LGROUP, and DR_ALARM).

> *Note*   *If you need help with LINtools, Chapter 2 explained how to access and use the configurator, and the context-sensitive Help system (press <F1>) provides instructions on every operation.*

2   Table 4-2a lists all the strategy connections in case you are starting from scratch. You won't need to do any wiring if you started from the existing programmer strategy.

Source*	Destination*	Function
(AI_UIO) PV.PV	(PID) pid.PV	PID process variable PV from the remote input module into the PID block
(PID) pid.OP	(AO_UIO) OP.OP	PID control output OP to the remote output module
(PID) pid.OP	(DR_ANCHP) OP_rcrd.CurrVal	PID control output OP into the OP_rcrd (DR_ANCHP) block to enable data recording onto Instrument's local flash storage
(PID) pid.PV	(DR_ANCHP) PV_rcrd.CurrVal	PID process variable PV into the PV_rcrd block to enable data recording
(PID) pid.SP	(DR_ANCHP) SP_rcrd.CurrVal	PID setpoint SP into the SP_rcrd block to enable data recording
(PID) pid.PV	(SPP_RAMP) spp_ramp.PV	PID process variable PV into the spp_ramp block to include PV in the 'preplot' group
(SPP_RAMP) spp_ramp.Out	(PID) pid.RemoteSP	setpoint generated by the locally-running setpoint program ('program1.uys') into the PID block's remote setpoint input
(SPP_RAMP) spp_ramp.Out	(DR_ANCHP) SP_prog.CurrVal	setpoint generated by the locally-running setpoint program ('program1.uys') into DR_ANCHP block to enable trending & Instrument's 'Preplot' facility
(SPP_RAMP) spp_ramp.PV	(DR_ANCHP) PV_prog.CurrVal	PID process variable PV (sourced from the PID block) into DR_ANCHP block to enable trending & Instrument's 'Preplot' facility
(SPP_DIG) spp_dig.Out.Bit0	(DR_DGCHP) stirrer.CurrVal	digital setpoint generated by the locally-running setpoint program ('program1.uys') into DR_DGCHP block to enable trending & Instrument's 'Preplot' facility
Note *(Block type) blockname.field.subfield format, referring to Figure 4-2*		

Table 4-2a   Strategy wiring data

> *Note*   *The Configuration (Header) block and the AREA, GROUP, LOGDEV, LGROUP, DR_ALARM, and SPP_CTRL blocks don't need to be wired up. They are associated with other blocks by setting field values, see Editing the block fields.*

4.2.2 Editing the block fields

Table 4-2 lists the fields that should, as a minimum, be edited from their default values. For completeness, all such fields are shown in the table, including those edited in the original setpoint program strategy. Note that you must also edit the **Name** fields in all blocks, according to Figure 4-2 as you wish

Note   If you are starting from the existing programmer strategy, you need only edit the block fields below the dotted line in Table 4-2.

Field Value*	Function
(PID) pid.TI = **5.00**	Sets the PID integral time constant to 5 seconds, to improve control
(PID) pid.LAA = **20.00**	Sets the pid block's PV output 'low absolute' alarm at 20%
(PID) pid.Alarms.LowAbs = **1**	Enables the 'low absolute' alarm on the pid block's PV output
(AREA) area.Id = **1**	Allocates area number (only Area '1' currently implemented). No OVERVIEW appears in the Instrument unless an area number is allocated
(AREA) area.Group1 = **pidloop**	Assigns the 'pidloop' GROUP block to this area. This specifies that pidloop will appear in the OVERVIEW and enables data recording for this group
(AREA) area.Group2 = **preplot**	Assigns the 'preplot' GROUP block to this area. This specifies that pidloop will appear in the OVERVIEW and enables data recording for this group
(GROUP) pidloop.Update = **10**	Specifies the recording sample rate (seconds) for the 'pidloop' group. The default rate of zero disables recording
(GROUP) pidloop.Disp1 = **pid**	Specifies the PID block 'pid' as the source of the data that will be displayed in the faceplate and trends for the 'pidloop' group
(GROUP) pidloop.Chan1 = **PV_rcrd**	Assigns the recorder channel block 'PV_rcrd' to channel 1 of the 'pidloop' recording group
(GROUP) pidloop.Chan2 = **SP_rcrd**	Assigns the recorder channel block 'SP_rcrd' to channel 2 of the 'pidloop' recording group
(GROUP) pidloop.Chan3 = **OP_rcrd**	Assigns the recorder channel block 'OP_rcrd' to channel 3 of the 'pidloop' recording group
(GROUP) preplot.Update = **10**	Specifies the recording sample rate (seconds) for the 'preplot' group. The default rate of zero disables recording
(GROUP) preplot.Disp1 = **spp_ramp**	Specifies the SPP_RAMP block 'spp_ramp' as the first source of the data that will be displayed in the faceplate and trends for the 'preplot' group
(GROUP) preplot.Disp2 = **spp_dig**	Specifies the SPP_DIG block 'spp_dig' as the second source of the data that will be displayed in the faceplate and trends for the 'preplot' group
(GROUP) preplot.Chan1 = **PV_prog**	Assigns the recorder channel block 'PV_prog' to channel 1 of the 'preplot' recording group
(GROUP) preplot.Chan2 = **stirrer**	Assigns the recorder channel block 'SP_prog' to channel 2 of the 'preplot' recording group
(GROUP) preplot.Chan3 = **SP_prog**	Assigns the recorder channel block 'stirrer' to channel 2 of the 'preplot' recording group
(GROUP) preplot.Chan4 = **HiAbs_rc**	Assigns the recorder channel block 'HiAbs_rc' to channel 3 of the 'preplot' recording group
(GROUP) preplot.Chan5 = **LoAbs_rc**	Assigns the recorder channel block 'LoAbs_rc' to channel 4 of the 'preplot' recording group
(DR_ANCHP) SP_prog.ZoneA_LO = **11**	Specifies '11' (%) as the bottom of the chart zone or band occupied by all SP plots. The default ZoneA_HI of 100% specifies the top of SP's plotting zone.
(DR_ANCHP) PV_prog.ZoneA_LO = **11**	Specifies '11' (%) as the bottom of the chart zone occupied by the PV plots. The top of the zone = 100% (default).
(DR_DGCHP) stirrer.ZoneA_HI = **10**	Specifies '10' (%) as the top of the chart zone occupied by the 'stirrer' digital plots. The bottom of the zone = 0% (default).

Continued...

Continued...

Field Value*	Function
(PID) pid.Alarms.HighAbs = **1**	Enables the 'high absolute' alarm on the pid block's PV output
(PID) pid.HAA = **80.00**	Sets the pid block's PV output 'high absolute' alarm at 80%
(PID) pid.SelMode.SelRem = **TRUE**	Selects Remote mode for the pid block, to allow it to use the remote setpoint generated by the running setpoint program
(PID) pid.SelMode.EnaRem = **TRUE**	Enables the pid block to operate in Remote mode
(LOGDEV) logdev.Group1 = **lgroup**	Assigns the 'lgroup' LGROUP block to Group 1 of the 'logdev' logging device block
(LOGDEV) logdev.State = **OFF**	Prevents access to the storage media device when application starts running
(LGROUP) lgroup.UpdateA = **10**	Specifies the 'lgroup' block's standard logging interval as 10 seconds (the minimum time for a single group). The default rate of zero disables logging
(LGROUP) lgroup.Point1 = **PV_rcrd**	Assigns the 'PV_rcrd' block's analogue value (CurrVal) to the log group, for logging as Point 1
(LGROUP) lgroup.Point2 = **SP_rcrd**	Assigns the 'SP_rcrd' block's analogue value (CurrVal) to the log group, for logging as Point 2
(LGROUP) lgroup.Point3 = **OP_rcrd**	Assigns the 'OP_rcrd' block's analogue value (CurrVal) to the log group, for logging as Point 3
(LGROUP) lgroup.Point4 = **dr_alarm**	Assigns the 'dr_alarm' alarm point block to Point 4 of the log group, to provide alarm filtering and logging for the log group

Note   *(Block type) blockname.field.subfield *format, referring to Figure 4-2a.*

Table 4-2   Values for non-default block fields in the logging strategy

4.3 TESTING THE DATA LOGGING STRATEGY

As for the setpoint programming strategy in Chapter 3, you can test out the data logging strategy using a SIM block to simulate plant behaviour. Figure 4-2a shows the slightly modified strategy.
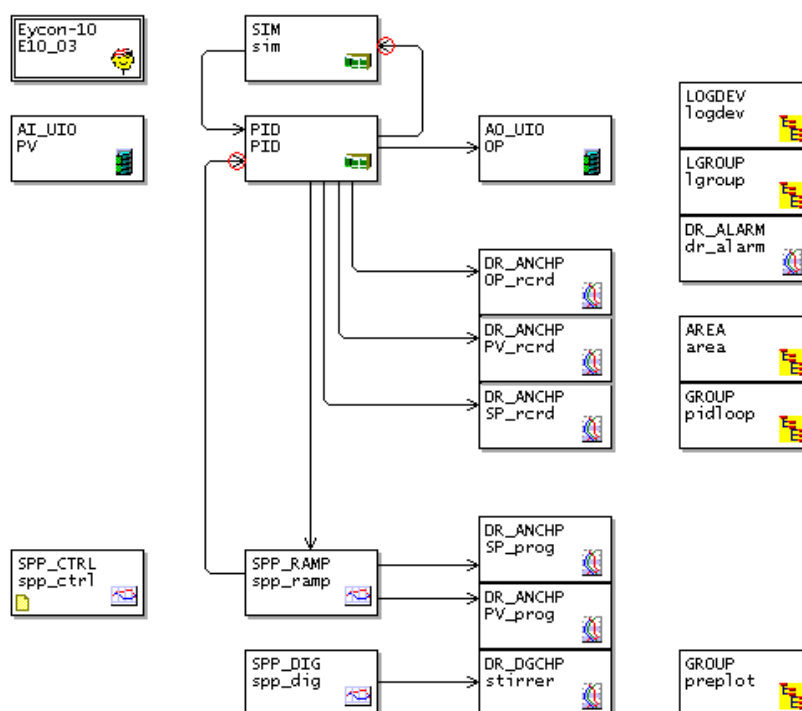
Figure 4-3a   Modified data logging strategy for testing

4.3.1 Modifying the data logging strategy

1   Delete the wire between the AI_UIO 'PV' block and the PID block.

2   Place a SIM simulation block on the worksheet, as shown in Figure 4-3a.

3   Wire the **SIM.OP** field into the **PID.PV** input, and **PID.OP** into **SIM.PV**. This forms a closed loop, indicated by the small red circle.

4   Name the SIM block, edit its **Lag1** field to **25.0**, and save the modified data logging strategy, e.g. as '**logging1.dbf**'. (Leave **NoiseMax** at the default **0.0**.)

4.3.2 Running the logging application

To run the logging simulation strategy and try out some of instrument's data logging features, do the following:

1   Download the saved modified 'simulation' strategy file to the instrument's E: drive. Only the **.dbf** file is required by the instrument, the other two files that appear when you save a strategy in LINtools (**.grf** and **.dtf**) are not used.

> *Note   If you need help with downloading, see Chapter 1,* Getting started.

2   If you haven't already done so, download the setpoint program file (e.g. **program1.uys**) that you configured in Chapter 3.

3   In the instrument, load and run the 'simulation' data logging application. After a delay the area overview appears, containing the 'pidloop' and 'preplot' group faceplates, exactly as in Chapter 3 for the programmer application.

> *Note   If you need reminding, Chapter 1,* Getting started, *describes how to select and run applications.*

4   Press the **Menu** key, then **LOGGING**, **GROUPS** to see the **Logging Groups** window. This shows you parameters associated with the log group you set up in the logging strategy. Most of the fields in the window access corresponding fields in the LGROUP log group block and have similar names. They can be conveniently edited 'on the fly' via this window. Table 4-3 lists the Instrument fields and equivalent LGROUP block fields, together with their functions.

> *Note*   *Refer to the* Eycon-10/Eycon-20 Handbook, *Part no. HA028209, if you need more detailed information on these fields.*

5   To edit Logging Groups fields, the **Logging** field must be set to 'OFF'. To do this, press the yellow **Logging** field 'ON' legend to pop up a menu of log group states. Select 'OFF' and hit return. Now you will see that all the 'Configuration' fields have become editable (yellowed).

6   Edit the blank **File Name** field via its pop-up keyboard with a suitable log file name (e.g. use the group name '**LGROUP**').

Logging Groups Field	LGROUP block Field	Function
Group Name	Name	Names the log group and the LIN block
Logging ON OFF TRIGGER	State ON OFF TRIGGER	Determines when log group is logged: Enables logging Disables logging Allows LGROUP's digital input to enable/disable logging
Archive Int	UpdateA/UpdateB	Logging interval (seconds)
File Type ASCII Binary Binary	FileType ASCII PACKED UHH	Specifies the file format generated by the log group: Tabulated text files for spreadsheets Compressed Propietary format .uhh file (recommended)
Name Type Text Hourly Daily	NameType TEXT HOURLY DAILY	Specifies the log file partitioning: Continuous file Data logged in a series of hourly files Data logged in a series of 24h files
File Name	FileName	Basename of the generated log files. 8 DOS characters max. for Text files; 2 characters max. for Hourly and Daily
Column Titles Present Absent	ColTitle Standard None	[ASCII file type only] Includes/excludes default column titles in log files: Column titles included in log files No column titles included in log files
Date Format Date Time Spreadsheet Integer Duration Days DHMS	DateFmt DateTime Sprdsht Integer Duration Days D,H,M,S	[ASCII file type only] Specifies format of date/time/duration in the logged files: E.g. 16/01/99,16:27:28 E.g. 37278.504190 (days since midnight 31/12/1899. For export to Excel etc.) E.g. 020122093201 (YYMMDDhhmmss) E.g. 00:01:02:26 (DD:hh:mm:ss since start) E.g. 0.1875 (days since start) E.g. 0,0,6,24 (D,h,m,s since start)
Compr Ratio Normal High	ComRatio NORMAL HIGH	[Binary/PACKED file type only] Specifies whether data is compressed: No data compression Data compressed (*not implemented*)

Table 4-3   Logging Groups and LGROUP block equivalent fields

7 Press the **SAVE** key to write your edits to the database. *If you don't save them your edits will be lost as soon as you leave the window.* Restore the **Logging** field to 'ON'.

Logging does not start yet because the LOGDEV block's *State* field is still 'OFF', disabling access to the logging medium.

*Note Some of instrument's **Logging Monitor** fields access corresponding fields in the LOGDEV function block. Table 4-4 shows this.*

Logging Groups Field	LGROUP block Field	Function
Disk	Status	Reports current status of logging to storage media
(*No equivalent*)	INACTIVE	No log files are open
Active	ACTIVE	Log files are open
Flushing	FLUSHING	Actual flushing of buffered data to storage media
Off-Line	OFF_LINE	Logging/recording terminated
Logging	State	Determines when the medium may be accessed for logging:
ON	ON	Enables logging to medium
OFF	OFF	Disables logging to medium
On Event	TRIGGER	Allows LOGDEV's Trigger digital input to enable/disable logging
Media Size	(*No equivalent*)	Capacity of the storage media (storage media dependant)
Free Space	FreeSpac	Free space on medium [kB in Logging Monitor window, % in LOGDEV block]
Free Time	FreeTime	Logging time remaining at current rate [hh:mm:ss in Logging Monitor window; hours in LOGDEV block]

Table 4-4   Logging Monitor and LOGDEV block equivalent fields

8 To give your Instrument something more interesting to log, start the setpoint program by pressing the top-left programming pane to pop up the **Programmer** window, then **PROGRAMS**. In the **File Name** field, select your setpoint program, e.g. 'PROGRAM1', and press **LOAD**. In the **Programmer** window again, press **RUN**, then **PRE-PLOT** to watch the program's progress.
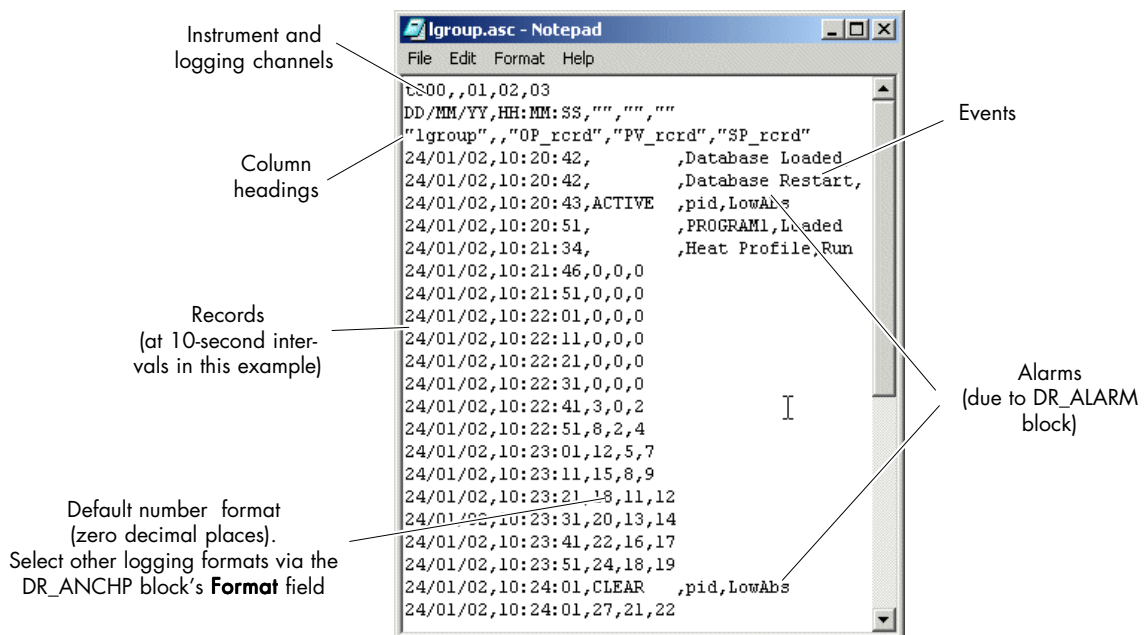
Figure 4-3b   Example log file (ASCII, Text, Date Time format)

9   Leave the logging going for a minute or two, then press **Menu**, **LOGGING**. Make a note of the current time shown near the top-left of the Instrument screen, then press **OFF**-**LINE** and remove the storage device.

10   While data logging continues the data is held in the internal buffer store. The exact amount depends on how much data is being logged, in this case several minutes of data will be buffered. Insert the storage device into your Computer and examine the contents so far. You should see a single log file, called, e.g. **lgroup.asc**. Open the file in a text editor such as 'Notepad' or as a .csv file in a spreadsheet (e.g. 'Excel'). Figure 4-3b shows the sort of contents you should see.

11   When you've finished examining the log file, re-insert the storage device into the Instrument and press **MONITOR**, to resume logging to storage device. You may have to wait a few moments to see any storage device activity. Buffered data is flushed onto the storage device.

> *Note*   *Should logging to storage device fail to resume, press **OFFLINE** then **MONITOR** to restart the process.*

12   Leave the logging going for a further period, e.g. until the setpoint program has finished, then set the **Logging Monitor** window's **Logging** field to 'OFF'. Press **OFFLINE** and remove the storage device.

> *Note*   *The **OFFLINE** keys that appear in the **Logging Monitor** window and also in the **Logging** window do the same job.*

13   Examine the contents of the storage device again in the Computer. You will now see two files, **lgroup.asc** and **lgroup.as1**. Open the **.as1** file, which is a logical continuation of the original 'asc' file, created because you interrupted logging by removing the disk. Further continuation files would keep the same root filename with extensions 'asc2', 'asc3', etc. You should find that despite removing the storage device for a few minutes, that no logged data has been lost.

4.4 ADDING A SECOND LOG GROUP

In this section you will add a second log group to the strategy (to log some digitals), and also control the existing log group using the 'On Event' facility.

Figure 4-4 shows the modified (simulation) strategy, with another LGROUP block added to collect data from a pair of DR_DGCHP digital channel blocks linked to the loop's high and low absolute PV alarms. An SPP_EXT block provides a trigger input to control the logging of the first log group 'lgroup', in this case only when the setpoint program is running.

Note   The 'spp_dig' and 'stirrer' blocks have been deleted from the strategy for simplicity.

4.4.1 Modifying the simulated data logging strategy

1   Start by opening the simulated data logging strategy in LINtools, and deleting the SPP_DIG block ('spp_dig') and the DR_DGCHP block ('stirrer').

2   Select and place the four new blocks on the worksheet, two DR_DGCHP blocks, an LGROUP and an SPP_EXT block.

3   Wire up the new blocks according to Table 4-5, which lists only the new connections.

4   Edit the new block fields and the existing LOGDEV and LGROUP blocks, according to Table 4-6, save the modified strategy under a new name (e.g. 'logging2.dbf'), and download it to the instrument's E: drive.
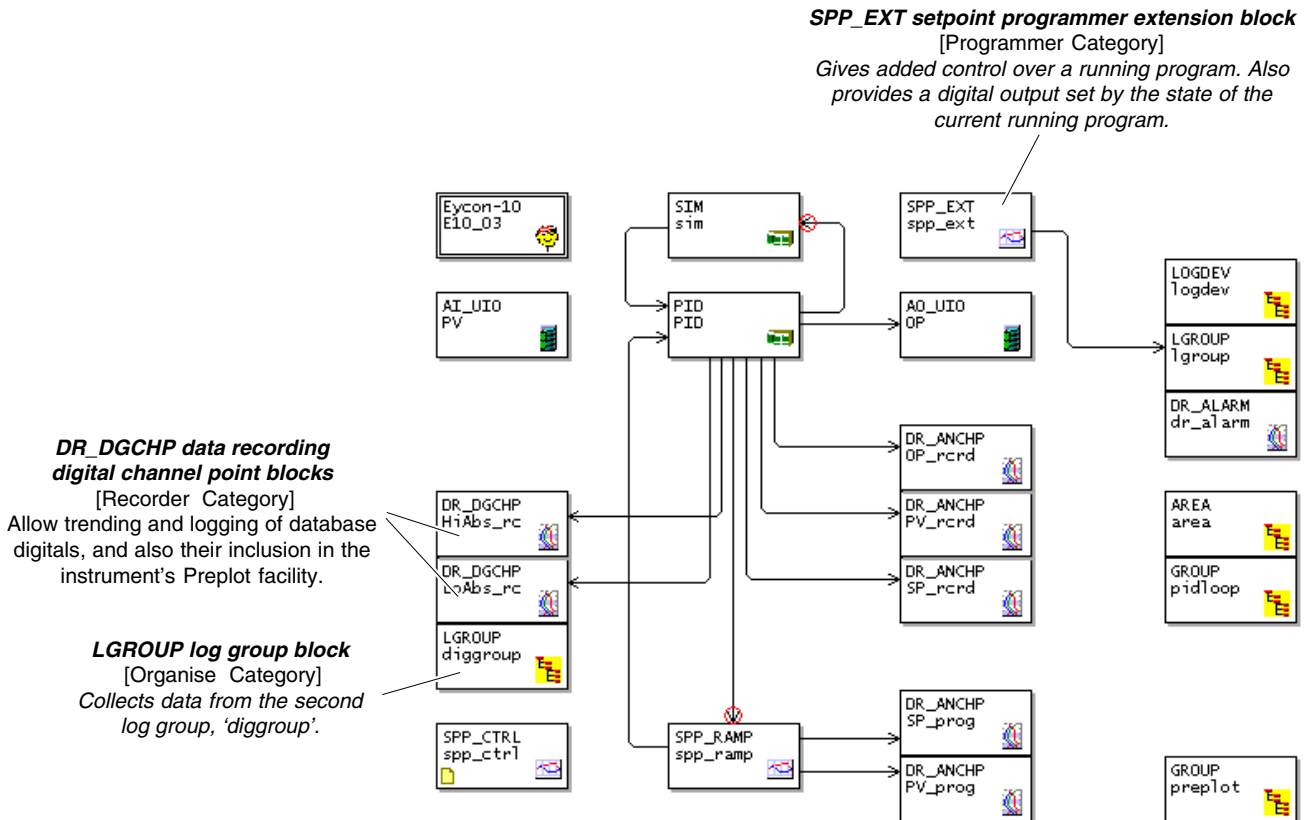
Figure 4-4   Second log group added to the (simulated) strategy

Source*	Destination*	Function
(SPP_EXT) spp_ext.Triggerd	(LGROUP) lgroup.Trigger	Triggerd digital output from the SPP_EXT block into the 'lgroup' block's Trigger input. Triggerd is TRUE only when the setpoint program is running. And for 'On Event' logging, data is logged only when Trigger is TRUE
(PID) pid.Alarms.HighAbs	(DR_DGCHP) HiAbs_rc.CurrVal	PID block's High Absolute alarm (on PV) into the HiAbs_rc (DR_DGCHP) block to enable the alarm to be data logged, and also included in Instrument's 'Preplot' facility
(PID) pid.Alarms.LowAbs	(DR_DGCHP) LoAbs_rc.CurrVal	PID block's Low Absolute alarm (on PV) into the LoAbs_rc (DR_DGCHP) block to enable the alarm to be data logged, and also included in Instrument's 'Preplot' facility
Note *(Block type) blockname.field.subfield *format, referring to Figure 4-1*		

Table 4-5   Extra wiring data for the two-group logging strategy

Field Value*	Function
(SPP_EXT) spp_ext.Trigger = **TRUE**	Causes the block's Triggerd output to become TRUE only when the Setpoint program is in the 'Running' state
(LOGDEV) logdev.Group2 = **diggroup**	Adds the new log group 'diggroup' to the logging device block, as Group 2
(LGROUP) lgroup.State = **TRIGGER**	Causes the 'lgroup' log group to perform data logging only when its Trigger input is TRUE
(LGROUP) lgroup.UpdateA = **20.00**	Specifies the 'lgroup' group logging interval as 20 seconds. This is the minimum time for each group in a two-group strategy (10s is the minimum for one group, 20s for two, 30s for three, etc.)
(DR_DGCHP) HiAbs_rc.ColourA = **red**	Specifies 'red' as the trace colour for the PID block's high absolute alarm. This will appear in trends and in the setpoint programmer 'Preplot' window
(DR_DGCHP) HiAbs_rc.ZoneA_HI = **19.00**	Defines the upper limit of the zone occupied by the 'HiAbs' trace as 19% of full chart scale
(DR_DGCHP) HiAbs_rc.ZoneA_LO = **11.00**	Defines the lower limit of the zone occupied by the 'HiAbs' trace as 11% of full chart scale
(DR_DGCHP) LoAbs_rc.ColourA = **blue**	Specifies 'blue' as the trace colour for the PID block's low absolute alarm. This will appear in trends and in the setpoint programmer 'Preplot' window
(DR_DGCHP) LoAbs_rc.ZoneA_HI = **9.00**	Defines the upper limit of the zone occupied by the 'LoAbs' trace as 9% of full chart scale
(DR_DGCHP) LoAbs_rc.ZoneA_LO = **0.00**	Defines the lower limit of the zone occupied by the 'LoAbs' trace as 0% of full chart scale — i.e. at the bottom (or left-hand) edge of the chart
(LGROUP) diggroup.UpdateA = **20.00**	Specifies the 'diggroup' group logging interval as 20 seconds. This is the minimum time for each group in a two-group strategy (10s is the minimum for one group, 20s for two, 30s for three, etc.)
(LGROUP) diggroup.Point1 = **HiAbs_rc**	Assigns the 'HiAbs_rc' block's digital value (CurrVal) to the 'diggroup' log group, for logging as Point 1
(LGROUP) diggroup.Point2 = **LoAbs_rc**	Assigns the 'LoAbs_rc' block's digital value (CurrVal) to the 'diggroup' log group, for logging as Point 2
(GROUP) preplot.Chan2 = **SP_prog**	Assigns the recorder channel block 'SP_prog' to channel 2 of the 'preplot' recording group
(GROUP) preplot.Chan3 = **[null]**	Unassigns channel 3 of the 'preplot' recording group (now unused)
Note *(Block type) blockname.field.subfield *format, referring to Figure 4-4*	

Table 4-6   Field edits for blocks in the two-group logging strategy

4.4.2 Modifying the setpoint program

For the new two-group strategy, you can slightly modify the existing setpoint program to result in a more interesting set of logged data.

To do this:

1   Double-click on the setpoint program file in Explorer (e.g. **program1.uys**) to open up its configuration window.

2   Delete the digital setpoint. To do this, right-click the 'Stirrer' faceplate to the right of the chart and select **Delete** in the popup context menu.

3   Add a new segment at the end of the program called 'Stand', configured as a 2-minute 'Dwell'. If you've forgotten how: at the left of the window, click on the blank 'End' segment to highlight it, then click the **Insert Profile Segment** toolbutton. To edit a column in the new segment, double-click on it to pop up its **Properties** window.

   (Chapter 3, *Creating a Setpoint Program* section, tells you how to edit setpoint programs.)

4   Modify the 'Heat Up' segment to Ramp @ 4.00 per second to a value of 80.00%.

5   Modify the 'Cool Down' segment to ramp down to a value of 20.00 instead of 0.00%. Save the edited program under a new name, e.g. '**program2.uys**', and download it to the Instrument.

Your finished setpoint program should look like Figure 4-5.
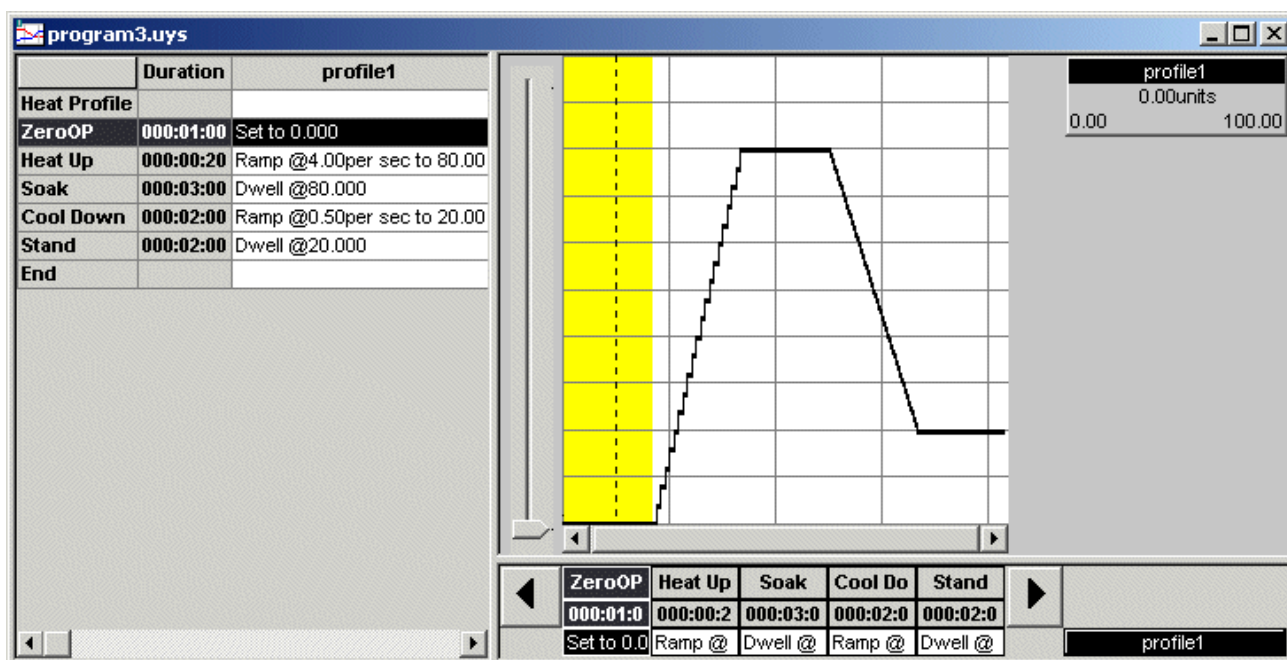
Figure 4-5   Modified setpoint program

4.4.3 Running the modified logging aplications

1   Stop and unload the currently-running application — press the **Menu** key, then **SYSTEM**, **APPLN**, **APP MGR**, **STOP**, **UNLOAD**.

2   In the Application Manager window, select the new two-group logging application then load and run it (press **LD+RUN**). The usual area overview appears with its two faceplates.

3   Access the **Logging Groups** window by pressing **Menu**, **LOGGING**, **GROUPS**. Press the ◁ or ▷ keys below the screen to cycle round all the log groups in the application — two in this case. Look at the 'lgroup' group first. Note that the **Logging** field now says 'TRIGGER' because you configured the 'lgroup' block's *State* field as TRIGGER (which is equivalent, see Table 4-3).

4   Edit 'TRIGGER' to 'OFF' temporarily, then enter a **File Name**, e.g. 'ANALOGS'. Press **SAVE** then return the **Logging** field to 'TRIGGER'.

5   Move to the 'diggroup' log group window and enter a **File Name** value, e.g. 'DIGITALS'. Just for a change, edit the **Name Type** field to 'Hourly'. Note that the file name automatically truncates to two characters followed by '??????'. Press **SAVE** and restore the **Logging** field to 'ON'. (Logging does not start yet because the LOGDEV block's *State* field is 'OFF', disabling access to the recording medium.)

> *Note*   *The truncation allows for six time-stamp characters to be incorporated into the series of log filenames generated hourly, e.g. 'DI092503.asc' for a file generated on the 9th month (September), 25th day, at 0300 hours. The possibility of truncation is why every log group file name must be unique in its first two characters.* **The SAVE operation will fail if this rule is broken.** *Try this if you wish! (Refer to the* Eycon-10/Eycon-20 Handbook, *Part no. HA 028 209, for detailed information on log filenames.)*

6   Inspect both log groups to make sure that they have been correctly edited and saved, and that neither of them have been left switched 'OFF'. Insert the storage media device into Instrument's drive and press **Menu**, **LOGGING**, **MONITOR**. In the **Logging Monitor** window, start the logging by setting **Logging** to 'ON'. Activity will soon start as the digital logging begins. (The analog logging is awaiting a trigger signal.)

7   Now press the grey 'program' pane at the top left of the screen (labelled 'RESET') to pop up the **Programmer** window. Press **PROGRAMS**, then select your newly modified setpoint program from the **File Name** list, e.g. 'PROGRAM2'. Press **LOAD**.

8   Back in the **Programmer** window press **RUN**, then **PRE-PLOT** to watch the program's progress. Figure 4-6 shows the pre-plot window as it might appear towards the end of the program. The high absolute alarms when PV overshoots the setpoint are clearly visible in the preplot (red trace).
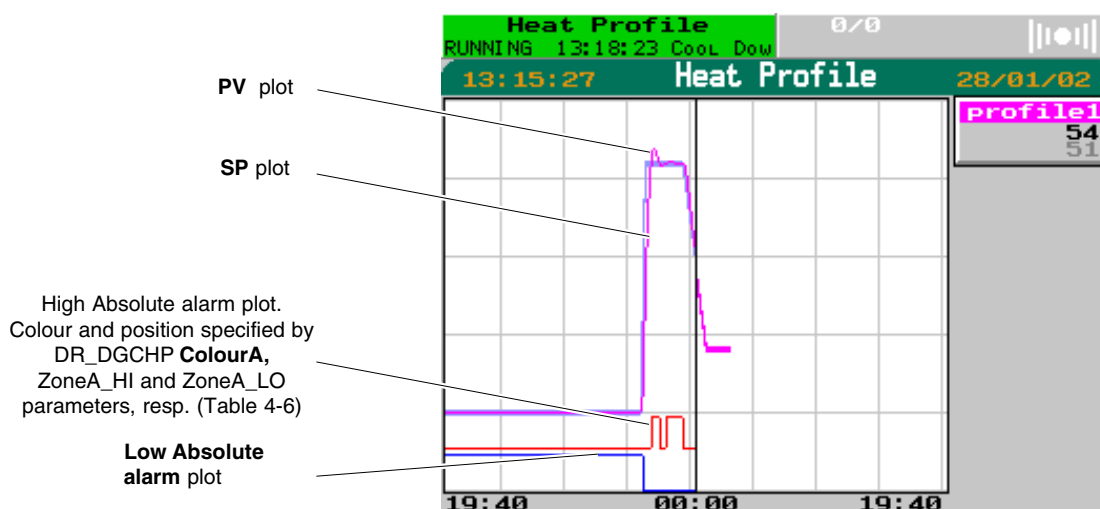

Figure 4-6   Pre-plot view of the two-group application

9   A short while after the setpoint program has stopped running, 'IDLE' is indicated in the top-left program pane, press **Menu**, **LOGGING**, **MONITOR** and reset the **Logging** field to 'OFF' to stop the logging. Press **OFFLINE** and remove the storage device. Examine the log files in your Computer using Notepad or a similar text editor. Note the 'Hourly' file-naming convention that you specified in step 5, and the contents of the logs.

10 Look at the 'analogs.asc' file contents and confirm that data logging of the log group 'lgroup' has taken place only while the setpoint program was in the 'RUNNING' state, automatically switching off when the program state was 'IDLE'. Your trigger input to the 'lgroup' block enabled this.

> *Note   Event and alarm messages do appear while the program is 'IDLE'.*

11 Also look at the 'digitals.asc' log file. The default enumerations for the HiAbs and LoAbs alarm states are 'OPEN' (= no alarm) and 'CLOSED' (= in alarm). Try specifying your own more meaningful enumerations as described in the next section.

SPECIFYING YOUR OWN DIGITAL ENUMERATIONS, .UYN FILE

Enumerations are usually Boolean two-state variables, such as TRUE/FALSE and OPEN/CLOSED. They appear in the Programmer graphics (PREVIEW, PRE-PLOT, and EDIT), and also in logging files, as in the present example.

Enumerations are assigned default text values by the Instrument, but you can override these by creating and downloading a text file called **<application name>.uyn**.

Try this:

1   Open up 'Notepad' or a similar text editor and type in the two lines of text *exactly* as shown in Figure 4-7. The format used for each line in the file is:

<p align="center">,<Block Name>.<Block Field>, "<User Text>,<User Text>"</p>
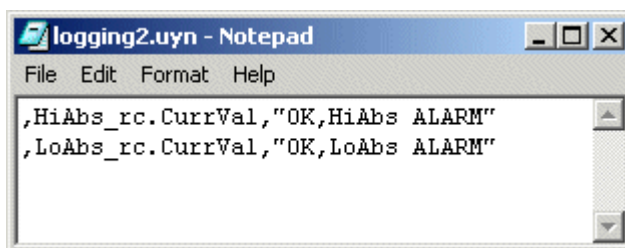
<p align="center">Figure 4-7   .uyn file assigning user enumerations — example</p>

In this example the high and low absolute alarm states are logged via the 'HiAbs_rc' and 'LoAbs_rc' DR_DGCHP (Stirrer) block *CurrVal* fields, respectively.

This will result in the substitution of 'OK' for the default 'OPEN' enumeration, and 'HiAbs ALARM' or 'LoAbs ALARM' for the default 'CLOSED' enumeration, in the two logged digital fields, which makes more sense!

2   Save the text file with the root name of your application (LIN database), e.g. '**logging2**', and edit its extension to **.uyn**. Download the file to the instrument's E: drive.

3   To activate the downloaded .uyn file you must now stop the application, **Menu** key, **SYSTEM**, **APPLN**, **APP MGR**, **STOP**, then start it again, **START**.

> *Note   You do not have to* unload *the stopped application.*

4   Examine the resulting log files. You will see the alarm enumerations you specified in the .uyn file appear instead of the default ones. Data Logging to External Archive directories is possible if the instrument is connected to an Ethernet Network. If Data Logging to External Archive directory, ensure the network configuration is correct, and run the setpoint program and logging session agai, as per steps 6 to 9 in the previous section, '*Running the modified logging application*'.

> *Note   Refer to the* Eycon-10/Eycon-20 Handbook, *Part no. HA 028 209, for .uyn files and their use details.*

4.5 CHARTING LOGGED DATA

If the Visual Supervisor is connected, as part of a Ethernet Network, the Archive files, .uhh file (recommended) or .pkd file, in the Visual Supervisor can be used to display your logged analogue data as a trend chart using the Review software, launched from,

<p align="center">🏁 Start > Programs > Eurotherm > Review</p>

Note   The Review Software allows the display and printing of archive data files from suitable recorders, data acquisition units, etc., see Review Help file, Part no.HA 027 962 for full details.

Data is transferred from the .uhh file (recommended), or .pkd files to a database on the Computer. Data is stored in the database, by instrument tag, descriptor, group name and point identifier.
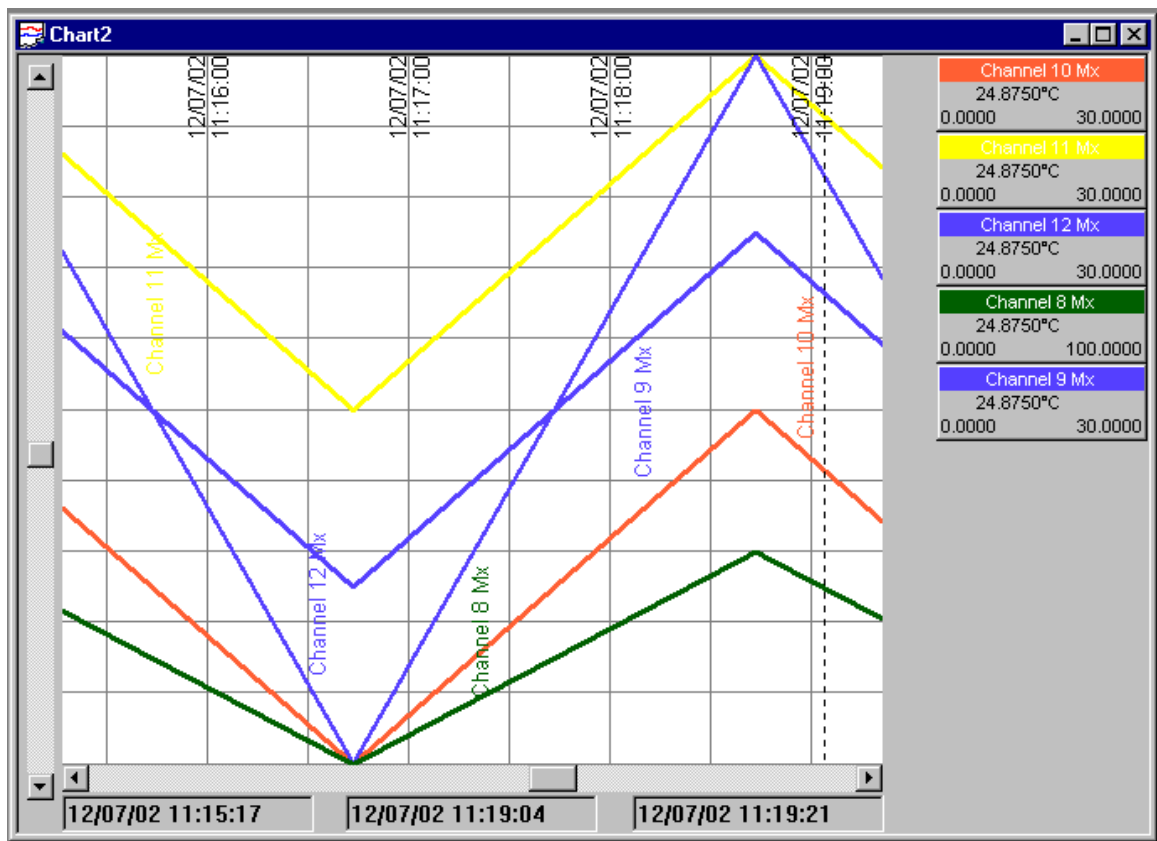
<p align="center">Figure 4-8   Review Software showing '.uhh' log file</p>

Intentionally left blank

CHAPTER 5 RECIPE CONTROL

This chapter assumes you have purchased the Instrument's *Recipe* option and shows you how to add 'Recipe' functionality to the data logging strategy you built in Chapter 4. You test-run simulations of the strategy and try out some recipe features. Recipes can be used in many applications, the data logging application is chosen here simply as a convenient starting-point.

The Recipe functionality is used to control the parameters that change based on the product or formula, e.g. reaction time.

The main sections are:

■   Adding Recipes to the data logging strategy

■   Running the recipe application

■   Adding a second line to the application

■   Running the two-line recipe application

Recipe functionality provides the Instrument with a set of LIN function block field names ('variables') with a particular value assigned to each, e.g. if the recipe is for one or more identical paint-mixing lines, it would include setpoint values representing the amounts of each pigment in the mixture needed for a particular colour. In use, the recipe is 'downloaded' to the LIN database controlling a particular plant line, i.e. each recipe value is copied to the corresponding LIN block field to achieve the required control.

You will usually want to run several related recipe on a given plant line at different times, with the same set of variables but with different assigned values, e.g. for different colour mixes. These recipes constitute a *recipe set*, that are stored in a single comma-separated text file with extension **.uyr**.

5.1 ADDING RECIPES TO THE DATA LOGGING STRATEGY

Figure 5-1 shows the (simulation) data logging strategy you created in the first part of Chapter 4, but with the setpoint program blocks deleted and some extra function blocks added to allow recipes to be used.

RCP_SET block
[Batch Category]
Controls the recipe set (.UYR file) to be used for one or more plant (recipe) lines. Each line is represented by an attached RCP_LINE block. (Up to 4 RCP_SET blocks per database)

RCP_LINE block
[Batch Category]
Controls the downloading of recipes to a single plant line, from a recipe set specified in the associated RCP_SET block. Up to 8 RCP_LINE blocks may attach to a RCP_SET block.

DR_ANCHP data recording analogue channel point blocks
[Recorder Category]
Edited to allow trending of the PID block's High and Low Absolute alarm levels.

Figure 5-1   Data logging plus recipe strategy (simulated)

5.1.1 Building the recipe strategy

1   In the LINtools Engineering Studio, locate the required blocks as shown in Figure 5-1. The quickest way to do this is to open up the existing simulated data logging strategy as your starting-point, by double-clicking its icon in Explorer view.

> *Note*   *In this chapter you build the* simulated *recipe strategy directly, starting from the simulated data logging strategy you created in Chapter 4, shown in Figure 4-2.*

2   In the existing strategy, delete the blocks associated with setpoint programming — which you won't be needing for this part of the tutorial. These are the five blocks named 'spp_ctrl', 'spp_ramp', 'spp_dig', 'stirrer', and 'preplot'. Then add a RCP_SET and a RCP_LINE block (from the Batch category) as shown in Figure 5-1.

3   Rename the two 'spare' DR_ANCHP blocks, which will now be used to trend the PID block's high and low absolute alarm levels.

4   Wire up the new blocks according to Table 5-1a, which as usual lists all the necessary strategy connections for completeness, in case you are starting from scratch.

Source*	Destination*	Function
(PID) pid.OP	(AO_UIO) OP.OP	PID control output OP to the remote output module
(PID) pid.OP	(DR_ANCHP) OP_rcrd.CurrVal	PID control output OP into the OP_rcrd (DR_ANCHP) block to enable data recording onto Instrument's local flash storage
(PID) pid.PV	(DR_ANCHP) PV_rcrd.CurrVal	PID process variable PV into the PV_rcrd block to enable data recording
(PID) pid.SP	(DR_ANCHP) SP_rcrd.CurrVal	PID setpoint SP into the SP_rcrd block to enable data recording
(PID) pid.OP	(SIM) sim.PV	PID control output OP into the SIM block's PV input to simulate output to the plant
(SIM) sim.OP	(PID) pid.PV	SIM block's lagged output OP into the PID block's PV input to simulate the plant's PV
[PID] pid.HAA	(DR_ANCHP) HAA_rcrd.CurrVal	PID High Absolute Alarm level HAA into the HAA_rcrd block to enable data recording
(PID) pid.LAA	(DR_ANCHP) LAA_rcrd.CurrVal	PID Low Absolute Alarm level LAA into the LAA_rcrd block to enable data

Table 5-1a   Strategy wiring data

> *Note*   *The Configuration (Header) block and the AREA, GROUP, LOGDEV, LGROUP, DR_ALARM, RCP_SET and RCP_LINE blocks don't need to be wired up. They are associated with other blocks by setting field values (see next).*

5.1.2 Editing the block fields

Table 5-2 lists the fields that should, as a minimum, be edited from their default values. For completeness, all such fields are shown in the table, including those edited in the original data logging strategy. Note that you must also edit the **Name** fields in all blocks, according to Figure 5-1 if you wish.

Finally, save the modified recipe strategy under a new name, e.g. '**recipe.dbf**'.

> *Note*   *If you are starting from the existing logging strategy, you need only edit the fields listed below the dashed line in Table 5-2.*

Field Value*	Function
(PID) pid.Alarms.HighAbs = **1**	Enables the 'high absolute' alarm on the pid block's PV output
(PID) pid.Alarms.LowAbs = **1**	Enables the 'low absolute' alarm on the pid block's PV output
(PID) pid.HAA = **80.00**	Sets the pid block's PV output 'high absolute' alarm at 80%
(PID) pid.LAA = **20.00**	Sets the pid block's PV output 'low absolute' alarm at 20%
(PID) pid.TI = **5.00**	Sets the PID integral time constant to 5 seconds, to improve control in this simulated strategy
(SIM) sim.Lag1 = **25.00**	Sets the SIM block's 1st filter time constant to 25 seconds, to simulate lag in the plant's process variable
(LOGDEV) logdev.Group1 = **lgroup**	Assigns the 'lgroup' LGROUP block to Group 1 of the 'logdev' logging device block
(LOGDEV) logdev.State = **OFF**	Prevents access to the floppy disk when application starts running
(LGROUP) lgroup.UpdateA = **10**	Specifies the 'lgroup' block's standard logging interval as 10 seconds (the minimum time for a single group). The default rate of zero disables logging
(LGROUP) lgroup.Point1 = **PV_rcrd**	Assigns the 'PV_rcrd' block's analogue value (CurrVal) to the log group, for logging as Point 1
(LGROUP) lgroup.Point2 = **SP_rcrd**	Assigns the 'SP_rcrd' block's analogue value (CurrVal) to the log group, for logging as Point 2
(LGROUP) lgroup.Point3 = **OP_rcrd**	Assigns the 'OP_rcrd' block's analogue value (CurrVal) to the log group, for logging as Point 3
(LGROUP) lgroup.Point4 = **dr_alarm**	Assigns the 'dr_alarm' alarm point block to Point 4 of the log group, to provide alarm filtering and logging for the log group
(AREA) area.Id = **1**	Allocates area number (only Area '1' currently implemented). No OVERVIEW appears in the Instrument unless an area number has been allocated
(AREA) area.Group1 = **pidloop**	Assigns the 'pidloop' GROUP block to this area. This specifies that pidloop will appear in the OVERVIEW and enables data recording for this group
((GROUP) pidloop.Update = **10**	Specifies the 'pidloop' group's recording sample rate (seconds). The default rate of zero disables recording
(GROUP) pidloop.Disp1 = **pid**	Specifies the PID block 'pid' as the source of the data that will be displayed in the faceplate and trends for the 'pidloop' group
(GROUP) pidloop.Chan1 = **PV_rcrd**	Assigns the recorder channel block 'PV_rcrd' to channel 1 of the 'pidloop' recording group
(GROUP) pidloop.Chan2 = **SP_rcrd**	Assigns the recorder channel block 'SP_rcrd' to channel 2 of the 'pidloop' recording group
(GROUP) pidloop.Chan3 = **OP_rcrd**	Assigns the recorder channel block 'OP_rcrd' to channel 3 of the 'pidloop' recording group
(AREA) area.DispMode.H_Trend = **TRUE**	Enables the horizontal trend display, needed for this application
(AREA) area.DispMode.V_Trend = **FALSE**	Disables the vertical trend display, which is not required in this application
(GROUP) pidloop.Chan4 = **HAA_rcrd**	Assigns the recorder channel block 'HAA_rcrd' to channel 4 of the 'pidloop' recording group
(GROUP) pidloop.Chan5 = **LAA_rcrd**	Assigns the recorder channel block 'LAA_rcrd' to channel 5 of the 'pidloop' recording group
(RCP_SET) mixers.Filename = **mixers**	Specifies the recipe set filename as 'mixers.uyr'
(RCP_SET) mixers.Line1 = **line1**	Associates the RCP_LINE block called 'line1' with the 'mixers' RCP_SET block
(DR_ANCHP) HAA_rcrd.ColourA = **Black**	Specifies 'black' as the colour of the High Absolute Alarm level trend, to contrast with the other traces
(DR_ANCHP) LAA_rcrd.ColourA = **Black**	Specifies 'black' as the colour of the Low Absolute Alarm level trend, to contrast with the other traces
(PID) pid.SelMode.SelAuto = **TRUE**	Selects automatic mode of operation for the PID block (overrides Remote mode settings)
(LGROUP) lgroup.FileName = **MIXERLOG**	Specifies 'MIXERLOG' as the root filename of any data logging files output from this log group

Note *(Block type) blockname.field.subfield *format, referring to Figure 5-2*

Table 5-2   Values for non-default block fields in the recipe strategy

5.2 RUNNING THE RECIPE APPLICATION

To run the (simulated) recipe application and try out some of instrument's recipe features, do the following:

1   Download the saved modified 'simulation' strategy file to the instrument's E: drive. (Only the **.dbf** file is required by the Instrument).

> *Note   If you need help with downloading, see Chapter 1,* Getting started.

2   In the Instrument, load and run the simulation recipe application. After a delay the 'pidloop' group display appears, containing the single 'pid' faceplate representing the PID control block, exactly as in Chapter 2 for the PID loop application.

> *Note   If you need reminding, Chapter 1,* Getting started, *describes how to select and run applications.*

3   Press the **Down** key until you see the pidloop horizontal trend display. This should show a pair of black traces representing the PID high and low absolute alarm levels — which you set to 80 and 20% respectively. The loop will be in low absolute alarm because PV is currently zero (the green trace at the bottom of the trend).

5.2.1 Creating a recipe set

You will now create a very simple recipe set consisting of (initially) three recipes for a 'mixing vat'. The vat is to maintain its contents at any one of three temperatures.

To create the recipe set in the Instrument:

1   Press the **Menu** key, then **RECIPE**, or simply press the grey blank box at the top-left of the screen, the 'recipe pane'. In the **Recipe** popup, press **RECIPES** to see the **Load/Save Recipe** page.

> *Note   The recipe options appear because you have configured a RCP_SET block in the running application.*

2   Press the **CREATE** button, then the **File Name** field in the **CREATE AS...** popup. Type in the filename 'MIXERS' and press return. **OK** your entry. A recipe editor table entitled 'MIXERS' appears. The first column, headed **RCP**, will contain the variable names in this recipe set, and the remaining columns will specify the particular recipe values assigned to each variable. Each column represents a recipe in the set, and the first one is named recipe '**1**' by default.

3   Press the **RCP** column heading. It turns yellow and pops up a recipe file properties dialog showing recipe set filename, version, edit details, and a **Timeout** value of 30 seconds (default).

> *Note   **Version** is incremented each time the file is saved. **Timeout** lets you specify the maximum time allowed*
> *for the successful download of a recipe, before the download aborts and a warning is displayed.*

4   Press **INSERT** to pop up the **Insert Variable** dialog. Edit the **Variable Name** field to 'temp', the temperature setpoint for recipe '1'. Edit the **Tag References SP** field to 'pid.SL', the PID block's local setpoint. Note that the tag reference entry is not case-sensitive. Press **OK** to return to the recipe editor table, which shows your first variable 'temp' with a default value of **0.0** under recipe '1'.

> *Note*   *You assign 'temp' to the* pid.SL *field because in this simulation the PID block is to control the temperature of the vat.*

5   Press the 'temp' cell to look at its **Properties** dialog. Check that the tag reference is correct. '**YES**' in the **Verify** field means the variable value is checked after download.   ('**NO**' is used for self-resetting variables, etc.) Press **OK** to return to the edit table.

6   To edit the default recipe name to something more suitable, press the cell containing '**1**' and type in 'hotmix' in the **Recipe Name** field. (This recipe will control the high-temperature mixture.) **OK** your edit. The new recipe name appears in the table.

7   To edit the default 'temp' value, press the '**0.0**' cell and enter a value of '80' as the 'hotmix' vat temperature.

8   Append another variable by pressing the 'temp' cell, then **INSERT**. Edit the variable name to 'hiabs' and the tag reference SP to 'pid.HAA'. Hit **OK**, and edit the default 'hiabs' value to '90'. This is the high absolute alarm level (temperature) for the 'hotmix' recipe.

9   Repeat step 8 to append and configure the last variable, 'loabs', using values given in Table 5-3. (Start by pressing the 'hiabs' cell, then **INSERT**.)

Variable (RCP)	Tag Reference	Hotmix	Warmmix	Coolmix
temp	pid.SL	80.00	50.00	30.00
hiabs	pid.HAA	90.00	55.00	31.00
loabs	pid.LAA	70.00	45.00	29.00

Table 5-3   'MIXERS' recipe set data (initial)

10   To append the next recipe column, press the 'hotmix' cell, then **NEW** to see the **New Recipe** dialog. Enter 'warmmix' as the new recipe name and press **OK**. Note that the default values in the new recipe column copy those of the previous column, leaving you to edit them as required. Edit these values according to Table 5-3.

11   Append the last ('coolmix') recipe and configure it as per Table 5-3. (Start by pressing the 'warmmix' cell, then **NEW**.) Finally, press **SAVE** to save the completed recipe set under its current name ('MIXERS').

Your completed 'MIXERS' recipe set edit table should look like Figure 5-2.
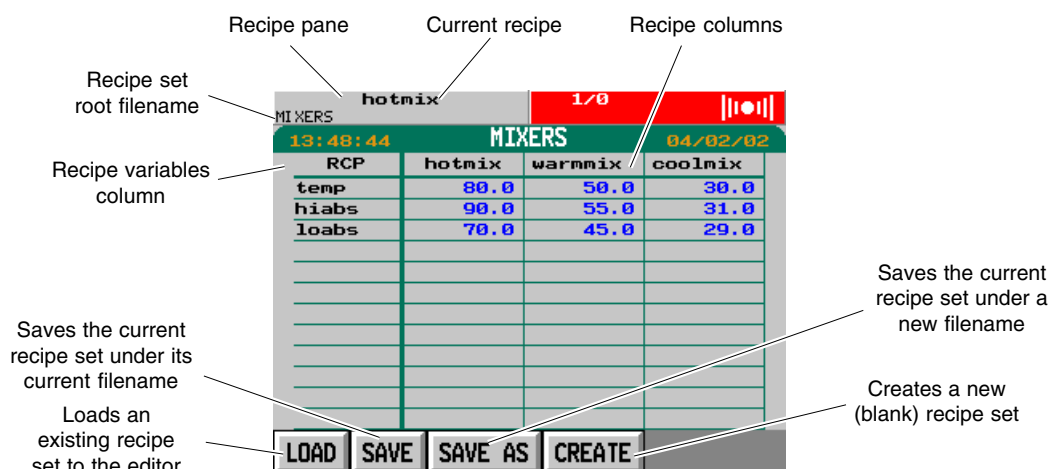
> *Note*   *Press the Options key for scroll bar to access any hidden recipe columns [none in this case].*

Figure 5-2   The completed 'MIXERS' recipe set edit table

5.2.2 Downloading recipes

Now try downloading a recipe to the running application. To do this:

1   Access the overview by pressing the **Menu** key, then **OVERVIEW**. Look at the horizontal trend overview.

2   Now press the recipe pane to pop up the **Recipe** window, then **MONITOR**. The recipe monitor page appears, entitled with the currently-selected recipe. It shows the recipe values (in blue) and also the actual 'live' values of the corresponding LIN block fields in the running strategy, in green, prior to download. These will not yet be the same.

3   Before downloading the recipe, it's useful at this point to try out the 'capture' facility. To copy the current live SP values into a new recipe, press the **CAPTURE AS** key. In the **Capture New Recipe** dialog, enter the name 'reset' and hit **OK**. Press **SAVE** to save the enlarged 'MIXERS' recipe set, overwriting the current version.

4   To see the newly-captured recipe, press the recipe pane, then **RECIPES**, **Recipe Name**. You will see 'reset' in the picklist of recipes. Leave the selection at 'hotmix'.

5   Press the recipe pane, **MONITOR**, **DOWNLOAD**, to download the recipe values to the associated tags (LIN block fields). The green 'SP live' values adopt the 'SP' recipe values and turn blue to indicate what has happened — they would turn *red* if the download failed to verify. Press the **Menu** key then **OVERVIEW** to see the effect.

6   After a short while, download another recipe from the 'MIXERS' set, e.g. 'warmmix'. To do this, press the recipe pane, **STATUS**, **Recipe Name** field, and select 'warmmix' from the picklist. Note that the recipe pane now shows 'warmmix' as the current recipe. Hit the **DOWNLOAD** button, and look at the 'pidloop' overview trend to verify that the setpoint and alarm levels have changed to the 'warmmix' recipe specifications.

7   Try selecting and downloading the 'coolmix' and 'reset' recipes. Figure 5-3 shows how your overview trend might look at this stage.
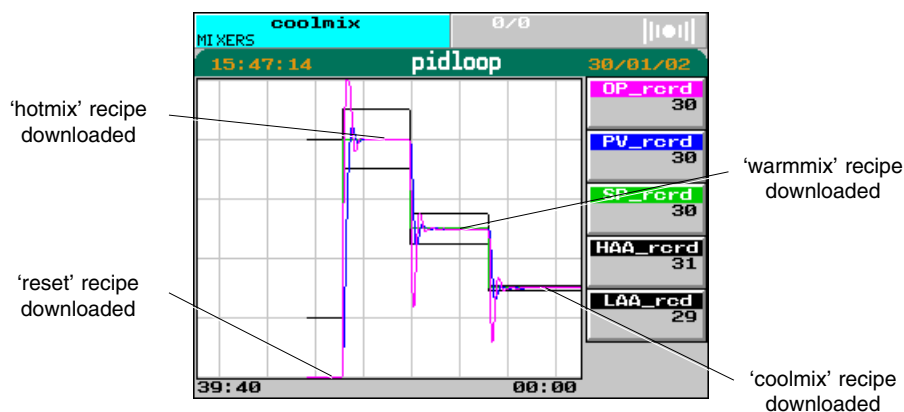
Figure 5-3   Trend showing the recipes in action

5.2.3 Looking at the recipe set .uyr file

The recipe set you just created resides as a comma-separated text file, 'MIXERS.UYR', on the instrument E: drive. Copy the file from the E: drive to your Computer, using Network Explorer if you are connected, or via a storage media device, and open it up in a text editor or spreadsheet.

Figure 5-4 shows the file opened up in Microsoft Excel, after conversion from text to columns. The recipe variables, tag references, and recipe values are clearly identifiable.

> **Note** *Refer to the* Eycon-10/Eycon-20 Handbook *(Part no. HA 029 280), if you need more detailed information on the structure of recipe set files.*

	A	B	C	D	E	F
1	UYR	1				
2	3	04/02/2002	15:56:38	ENGINEER		
3		30				
4		Setpoint:1	hotmix	warmmix	coolmix	reset
5	temp	pid.SL	80	50	30	0
6	hiabs	pid.HAA	90	55	31	80
7	loabs	pid.LAA	70	45	29	20

Figure 5-4   The MIXER.UYR recipe set file (in Excel spreadsheet)

5.3 ADDING A SECOND LINE TO THE APPLICATION

You will often want to run your recipe set with more than just one plant line at a time, so that processes can occur simultaneously, e.g. you may want to mix several paint colours at once in several identical plant lines, all using the same or a different recipe from the recipe set.

In this section of the tutorial you add a second 'line' to the recipe application, to allow you to control two 'mixing vats'. To do this, start by modifying the LIN database to include a second control loop and associated RCP_LINE block.

5.3.1 Adding a second loop to the strategy

Figure 5-5 shows the modified (simulated) strategy, with a second set of control loop blocks added. Also added are a RCP_LINE block and a GROUP block. To avoid unnecessary clutter, the data logging blocks and the AI_UIO and AO_UIO blocks are not shown in the figure. You can delete them from the strategy if you wish, as they are not needed for this section of the tutorial.

1   Start by opening the simulated recipe strategy in LINtools, shown in Figure 5-1.

2   For simplicity, delete the data logging blocks LOGDEV, LGROUP, and DR_ALARM, (or move them well out of the way on the worksheet). Delete the AI_UIO and AO_UIO blocks.

3   Select both the AREA and GROUP blocks and relocate them to be next to the RCP_SET and RCP_LINE blocks, see Figure 5-1.

4   Select the set of blocks linked to the PID block, i.e. the SIM, PID, and the five DR_ANCHP blocks, e.g. drag a dotted box around the blocks. Copy (Ctrl+C) and paste (Ctrl+V) the selection, then move the superimposed copy, while still selected, to the right of the original blocks, as in Figure 5-5.

5   Select the RCP_LINE and GROUP blocks, then copy, paste, and position the copies just below the existing pair of blocks. Rename the two GROUP blocks with more sensible names, 'vat1' and 'vat2', as in Figure 5-5.
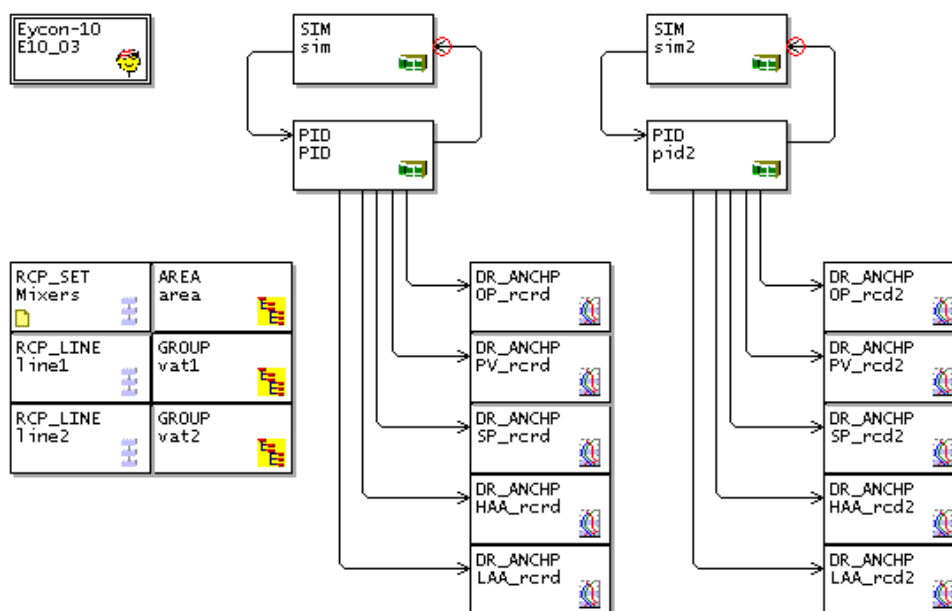
Figure 5-5   The modified two-line (simulation) strategy

6   Edit the new block fields and the existing AREA and RCP_SET blocks, according to Table 5-4, then save the modified strategy under a new name, e.g. 'recipe1.dbf', and download it to the instrument's E: drive.

> *Note*   *Most of the copied block fields won't need editing because parameter values are copied as well, except for block names and certain channel/display fields listed in Table 5-4.*

Source*	Function
(RCP_SET) Mixers.Line2 = **line2**	Associates the new RCP_LINE block 'line2' with the RCP_SET recipe set block 'Mixers'
(AREA) area.Group1 = **vat1**	LINtools automatically assigns the 'vat1' GROUP block to this area. This specifies that vat1 will appear in the OVERVIEW and enables data recording for this group.
(AREA) area.Group2 = **vat2**	Assigns the 'vat2' GROUP block to this area. This specifies that vat2 will appear in the OVERVIEW and enables data recording for this group. [LINtools assigns the 'vat1' group automatically]
(GROUP) vat2.Disp1 = **pid2**	Specifies the PID block 'pid2' as the source of the data that will be displayed in the faceplate and trends for the 'vat2' group
(GROUP) vat2.Chan1 = **PV_rcd2**	Assigns the recorder channel block 'PV_rcd2' to channel 1 of the 'vat2' recording group
(GROUP) vat2.Chan2 = **SP_rcd2**	Assigns the recorder channel block 'SP_rcd2' to channel 2 of the 'vat2' recording group
(GROUP) vat2.Chan3 = **OP_rcd2**	Assigns the recorder channel block 'OP_rcd2' to channel 3 of the 'vat2' recording group
((GROUP) vat2.Chan4 = **HAA_rcd2**	Assigns the recorder channel block 'HAA_rcd2' to channel 4 of the 'vat2' recording group
(GROUP) vat2.Chan5 = **LAA_rcd2**	Assigns the recorder channel block 'LAA_rcd2' to channel 5 of the 'vat2' recording group

Note   *(Block type) blockname.field.subfield *format, referring to Figure 5-5*.

Table 5-4   Field edits for blocks in the two-line recipe strategy

5.3.2 Adding a second line to the MIXERS recipe set

The next stage is to add a second line to the existing MIXERS recipe set. This will use the same set of variables and recipes as does line 1, but the variables will have different tag references as they are to be downloaded to a different part of the LIN blocks strategy. Specifically, the line 1 recipes download to 'pid' fields (controlling 'vat1'), and the line 2 recipes will download to the corresponding 'pid2' fields (controlling 'vat2').

To edit the recipe set:

1   Stop and unload the currently-running application — press the **Menu** key, then **SYSTEM**, **APPLN**, **APP MGR**, **STOP**, **UNLOAD**.

2   In the Application Manager window, select the new two-line recipe application then load and run it (press **LD+RUN**). The area overview appears with two faceplates — 'vat1' and 'vat2'. Press one of the faceplates, then the **Down** key until you reach the horizontal trend display. This is now the 'set' view.

3   Press the blank recipe pane (top-left corner of screen), then press **RECIPES** in the **Recipe** popup and select the **MIXERS** filename from the **Load/Save Recipe** page. **LOAD** the file.

4   In the **Recipe** popup press **EDIT** to see the **MIXERS** edit table, with the four recipes you configured earlier. To add another line to the recipe set, press the **Option** key repeatedly until **ADD LINE** appears at the foot of the page.

*Note   The **ADD LINE** key is active only if there is an unassigned RCP_LINE block in the associated LIN database. Otherwise it is 'greyed out' (unavailable).*

5   Press **ADD LINE**. A figure '**1**' appears at the bottom right corner of the recipe pane denoting that line '1' is now on display in the edit table. ('1' is the default line name.) Press the cell containing '**RCP**' to pop up the recipe file properties dialog and edit the default **Line** '**1**' to a more meaningful name, e.g. '**vat1**'. **OK** your edit — 'vat1' now appears in the recipe pane.

6   Press the **Down** key to scroll to the line '2' edit table. This is blank (apart from the variable and recipe names) because you have not yet assigned tag references to the variables. Edit the default line name to 'vat2', as in step 5.

7 Now assign the new tag references. Press the '**temp**' cell to pop up its **Properties** dialog. Edit **Verify** to 'YES', and edit the blank **Tag Reference SP** to 'pid2.sl'.

Remember, tag references are not case-sensitive. **OK** your edits. Note that the **temp** variable's existing recipe values now appear in the table(Figure 5-6).

Figure 5-6   'temp' variable tag references assigned

8 Assign the remaining variable tag references as in step 7, according to Table 5-5. (Start by pressing the '**hiabs**' cell.)

Variable (RCP)	Tag Reference	Hotmix	Warmmix	Coolmix	Reset
temp	pid.SL	80.00	50.00	30.00	0.00
hiabs	pid.HAA	90.00	55.00	31.00	80.00
loabs	pid.LAA	70.00	45.00	29.00	20.00

Table 5-5   'vat2' data in the 'MIXERS' recipe set

9 When you have finished, press the **Option** key to cycle the softkeys until you see the **SAVE** key, then press it to save your edited recipe set file. **OK** the save, which will overwrite the existing MIXERS.UYR file.

10 If you wish, take a look at the MIXERS.UYR file in 'Notepad', to see what difference the additional line has made.

5.4 RUNNING THE TWO-LINE RECIPE APPLICATION

Now that you've added the second line to your recipe set, try using it to control the two 'vats'.

1   Start by downloading the 'coolmix' recipe to 'vat1'. To do this, press the recipe pane then **STATUS** to access the **Recipe Status** page. This shows you the recipe set file currently loaded, 'MIXERS', the current line, 'vat1' or 'vat2', and the recipe currently selected for download to that line, 'hotmix', 'warmmix', 'coolmix', or 'reset'. Note that this information is also displayed in the recipe pane.

2   Press the **Line** field and select and enter '**vat1**' from the picklist. Similarly, select the '**coolmix**' recipe via the **Recipe Name** field. The recipe pane legends change accordingly.

3   Now press **DOWNLOAD** to copy the 'coolmix' recipe values to 'vat1'. Notice that the recipe pane turns from grey to cyan, signifying that a download has occurred. Also note that the **Recipe Status** page now displays the download time and date, and the **Status** of the download, '**COMPLETE**' if successful.

> *Note   **Status** can be **DOWNLOADING** if a download is in progress,**COMPLETE** if the latest download was completed successfully, or **FAILED** if the previous download was unsuccessful or aborted.*

4   Look at the 'vat1' horizontal trend to check that the vat is indeed running 'coolmix' — press **Menu**, **OVERVIEW**, then the '**vat1**' faceplate.

5   Now get 'vat2' going. Press the recipe pane, then **STATUS**. This time move to the 'vat2' line by pressing the **Down** key instead of editing the **Line** field, it's quicker! You can scroll round all the lines using the **Down** key.

6   Select '**warmmix**' via the **Recipe Name** field and press **DOWNLOAD** to download it to 'vat2'. Verify that the **Status** is 'COMPLETE'. Look at the 'vat2' trend to check that 'warmmix' is running.

You are now in control of both 'vats'!

Intentionally left blank

CHAPTER 6 BATCH CONTROL

This chapter gets you using some of the Instrument's *Batch* functionality, for which you must have purchased the *LO2 Batch* option.

You start off by adapting the setpoint program strategy you built in Chapter 3, to make it 'drive' the instruments' batch engine. That is, you use the setpoint program segments to define the batch phases. You also include a recipe set, based on the one you configured in Chapter 5, so that you can vary batch parameters.

After that you add batch reporting, and customise some of the batch screens to enhance the application. This involves editing the *batch file* and creating a *user dictionary* and a report *form file*.

The main sections in this chapter are:

- Modifying the programmer strategy for batch

- Creating batch files

- Running the batch application

- Enhancing the batch application

- Running the enhanced batch application

The Batch functionality is used to control the parameters that change based on the product or formula. Each Batch is the single process that produces the required product.

6.1 MODIFYING THE PROGRAMMER STRATEGY FOR BATCH

Figure 6-1 shows the (simulation) programmer strategy you created in Chapter 3, but with a Batch Control block and two recipe blocks, Recipe Set and Recipe Line, added to give the strategy some batch and recipe functionality.
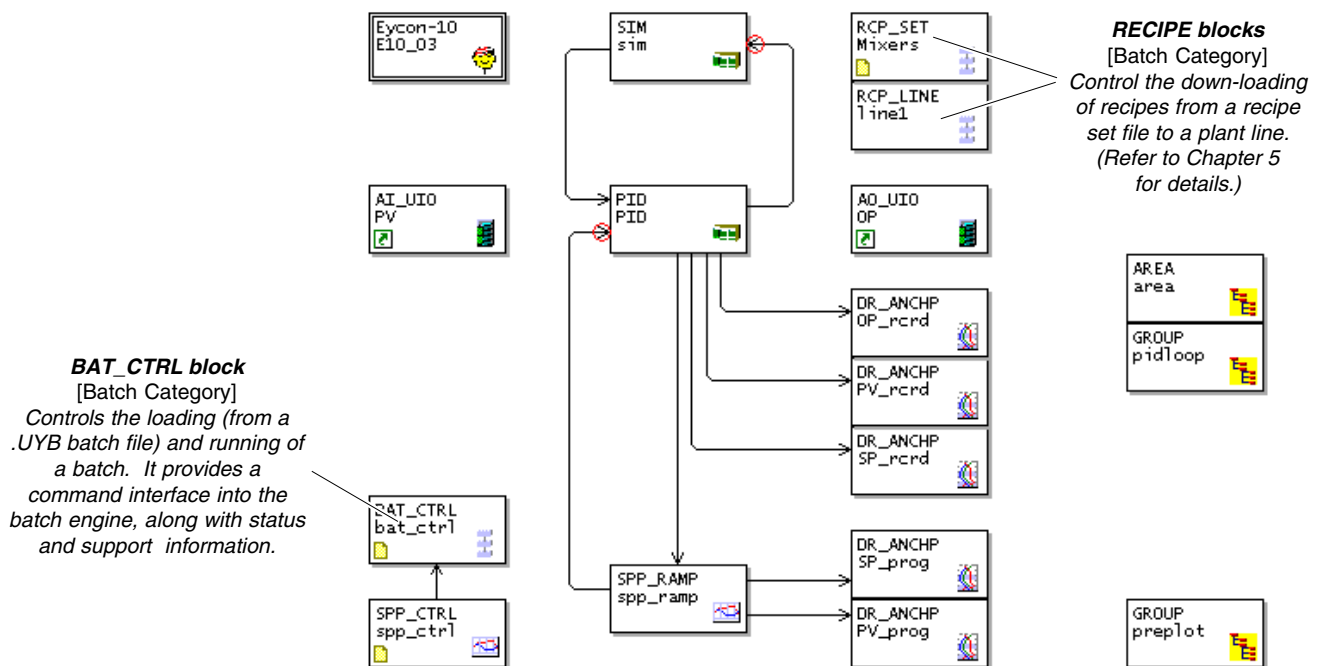
Figure 6-1   The program-driven batch strategy (simulated)

6.1.1 Building the batch strategy

1   In the LINtools Engineering Studio, locate the required blocks as shown in Figure 6-1. The quickest way to do this is to open up the existing simulated setpoint programmer strategy as your starting-point, by double-clicking its icon in Explorer view. Save the strategy under a new name, e.g. '**batch.dbf**'.

2   Delete the AI_UIO, AO_UIO, SPP_DIG, and DR_DGCHP blocks. They are not strictly needed in the simulation and will clutter up the worksheet when you add more blocks later. Add a BAT_CTRL block from the Batch category.

3   Now add an RCP_SET and an RCP_LINE block.

> *Note*   *A neat way of doing this is to open up the recipe strategy, created in Chapter 5, in a second, smaller, LINtools window, so that you can see both windows at once. Select the two recipe blocks and simply 'drag' them across into the batch strategy window. They are copied along with their field values. Alternatively, you can copy and paste the blocks.*

4   Wire up the new blocks according to Table 6-1, which as usual lists all the necessary strategy connections for completeness, in case you are starting from scratch.

Source*	Destination*	Function
(PID) pid.OP	(AO_UIO) OP.OP	PID control output OP to the remote output module
(PID pid.OP	(DR_ANCHP) OP_rcrd.CurrVal	PID control output OP into the OP_rcrd (DR_ANCHP) block to enable data recording onto instrument's local flash storage
(PID) pid.PV	(DR_ANCHP) PV_rcrd.CurrVal	PID process variable PV into the PV_rcrd block to enable data recording
(PID) pid.SP	(DR_ANCHP) SP_rcrd.CurrVal	PID setpoint SP into the SP_rcrd block to enable data recording
(PID) pid.PV	(SPP_RAMP) spp_ramp.PV	PID process variable PV into the spp_ramp block to include PV in the 'preplot' group
(PID) pid.OP	(SIM) sim.PV	PID control output OP into the SIM block's PV input to simulate output to the plant
(SIM) sim.OP	(PID) pid.PV	SIM block's lagged output OP into the PID block's PV input to simulate the plant's PV
(SPP_RAMP) spp_ramp.Out	(PID) pid.RemoteSP	setpoint generated by the locally-running setpoint program (e.g. 'prog1.uys') into the PID block's remote setpoint input
(SPP_RAMP) spp_ramp.Out	(DR_ANCHP) SP_prog.CurrVal	setpoint generated by the locally-running setpoint program into DR_ANCHP block to enable trending & instrument's 'Preplot' facility
(SPP_RAMP) spp_ramp.PV	(DR_ANCHP) PV_prog.CurrVal	PID process variable PV (sourced from the PID block) into DR_ANCHP block to enable trending & instrument's 'Preplot' facility
- - - - - - - - - - - - - -	- - - - - - - - - - - - - -	- -
(SPP_CTRL) spp_ctrl.CurrSeg	(BAT_CTRL) bat_ctrl.CurPhase	SPP_CTRL block's currently-active segment field CurrSeg into the BAT_CTRL block's CurPhase field, to make the setpoint program drive the phase of the batch engine

> *Note*   **(Block type) blockname.field.subfield format, referring to Figure 6-1. New wiring is listed below the dashed line.*

Table 6-1   Strategy wiring data

> *Note*   *The Instrument Configuration (Header) block and the AREA, GROUP, RCP_SET and RCP_LINE blocks don't need to be wired up. They are associated with other blocks via field values (see next). If you are starting from the existing programmmer strategy created in Chapter 3 you need only wire up the connection below the dashed line in Table 6-1.*

6.1.2 Editing the block fields

Table 6-2 lists the fields that should, as a minimum, be edited from their default values. For completeness, all such fields are shown in the table, including those edited in the original programmer strategy.

> *Note* *If you are starting from the programmer strategy created in Chapter 3, you need only edit the fields below the dashed line in Table 6-2.*

You must also edit the Name fields in all blocks, according to Figure 6-1 if you wish.

When you've carried out all necessary field edits, save the modified batch strategy again.

Field Value*	Function
(PID) pid.TI = **5.00**	Sets the PID integral time constant to 5 seconds, to improve control in this simulated strategy
(PID) pid.HAA = **80.00**	Sets the pid block's PV output 'high absolute' alarm at 80%
(PID) pid.LAA = **20.00**	Sets the pid block's PV output 'low absolute' alarm at 20%
(PID) pid.Alarms.LowAbs = **1**	Enables the 'low absolute' alarm on the pid block's PV output
(PID) pid.Alarms.HighAbs = **1**	Enables the 'high absolute' alarm on the pid block's PV output
(PID) pid.Alarms.HighDev = 1	Enables the 'high deviation' alarm on the pid block's PV output
(PID) pid.Alarms.LowDev = 1	Enables the 'low deviation' alarm on the pid block's PV output
(SIM) sim.Lag1 = **25.00**	Sets the SIM block's 1st filter time constant to 25 seconds, to simulate lag in the plant's process variable
(AREA) area.Id = **1**	Allocates area number (only Area '1' currently implemented). No OVERVIEW appears in the instrument unless an area number has been allocated
(AREA) area.Group1 = **pidloop**	Assigns the 'pidloop' GROUP block to this area. This specifies that 'pidloop' will appear in the OVERVIEW and enables data recording for this group
(AREA) area.Group2 = **preplot**	Assigns the 'preplot' GROUP block to this area. This specifies that 'preplot' will appear in the OVERVIEW and enables data recording for this group
(GROUP) pidloop.Update = **10**	Specifies the 'pidloop' group's recording sample rate (seconds). The default rate of zero disables recording
(GROUP) pidloop.Disp1 = **pid**	Specifies the PID block 'pid' as the source of the data that will be displayed in the faceplate and trends for the 'pidloop' group
(GROUP) pidloop.Chan1 = **PV_rcrd**	Assigns the recorder channel block 'PV_rcrd' to channel 1 of the 'pidloop' recording group
(GROUP) pidloop.Chan2 = **SP_rcrd**	Assigns the recorder channel block 'SP_rcrd' to channel 2 of the 'pidloop' recording group
(GROUP) pidloop.Chan3 = **OP_rcrd**	Assigns the recorder channel block 'OP_rcrd' to channel 3 of the 'pidloop' recording group
(GROUP) preplot.Update = **10**	Specifies the recording sample rate (seconds) for the 'preplot' group. The default rate of zero disables recording
(GROUP) preplot.Disp1 = **spp_ramp**	Specifies the SPP_RAMP block 'spp_ramp' as the source of the data that will be displayed in the faceplate and trends for the 'preplot' group
(GROUP) preplot.Chan1 = **PV_prog**	Assigns the recorder channel block 'PV_prog' to channel 1 of the 'preplot' recording group
(AREA) area.DispMode.H_Trend = **TRUE**	Enables the horizontal trend display, needed for this application
(AREA) area.DispMode.V_Trend = **FALSE**	Disables the vertical trend display, which is not required in this application
(PID) pid.SelMode.SelRem = **TRUE**	Selects remote mode of operation for the PID block, needed to allow use of the program's remote setpoint
(PID) pid.SelMode.EnaRem = **TRUE**	Enables remote mode of operation for the PID block
(GROUP) preplot.Chan2 = **SP_prog**	Assigns the recorder channel block 'SP_prog' to channel 2 of the 'preplot' recording group
(GROUP) preplot.Chan3 = **[null]**	Unassigns channel 3 of the 'preplot' recording group (now unused)
(RCP_SET) mixers.FileName = **mixers**	Entering the filename allows LINtools to offer a context menu to create the required file using this filename.
(RCP_SET) mixers.Line1 = **line1**	Associates the RCP_LINE block called 'line1' with the 'mixers' RCP_SET block
(SIM) sim.NoiseMax = **0.00**	Sets the SIM block's pseudo-random noise maximum amplitude to zero
(DR_ANCHP) SP_prog.ZoneA_LO = **0.00**	Resets zone to its default value
(DR_ANCHP) PV_prog.ZoneA_LO = **0.00**	Resets zone to its default value

> *Note* **(Block type) blockname.field.subfield format, referring to Figure 6-1*

Table 6-2   Values for non-default block fields in the batch strategy

6.2 CREATING BATCH FILES

Having configured a batch strategy you must now create a *batch file*. This comma-separated text file has a .UYB extension and specifies how the batch will operate. The easiest way to create a basic batch file is via instrument's CREATE batch facility. Later you will go on to customise this file using a text editor, e.g. 'Notepad' to add useful features to the batch.

To create the basic batch file:

1   Download the batch strategy file ('**batch.dbf**') to the instrument's E: drive. (Only the **.dbf** file is required.) If you need reminding, Chapter 1 describes how to download, select, and run applications.

2   In the instrument, load and run the batch application. After a delay the area overview appears, with two group faceplates, 'pidloop' and 'preplot, exactly as in Chapter 3 for the setpoint program application.

> *Note*   *As soon as you run up the batch application, the instrument automatically creates an empty default setpoint program file on its E: drive, named after the strategy, '***batch.uys***' in this case. It assumes that you will need this because you have included an SPP_CTRL block in the strategy.*

3   Press the **Menu** key, then **BATCH**, **BATCHES**, to see the **Load Batch** screen.
(The batch keys appear because you included a BAT_CTRL block in the strategy.)

4   Press the **CREATE** button to pop up the **CREATE AS...** dialog. Enter the name 'BATCH' in the **Filename** field.

> *Note*   *You don't* have *to name your batch file after the LIN database, but it makes life simpler.*

5   In the **Recipe Line** field, select 'line1', which is the name of the RCP_LINE block controlling the downloading of recipes to the strategy. You want these recipes (from the recipe set) to be used when a batch runs.

6   In the **Display Group** field, select 'pidloop' to associate this group with the batch. The remaining dialog fields must be left as '**<NONE>**' because the strategy does not (yet) have the necessary function blocks to support the extra functionality.

7   **OK** your entries. A **SAVE** popup warns you that the automatically-created BATCH.UYS file will be overwritten, which is okay because you won't be using it anyway! Press **OK** to create and save the new batch file and return to the **Load Batch** screen.

8   Now inspect instrument's E: drive to see what new files have appeared.

   ■   You can do this most conveniently on the Computer via Network Explorer if you are linked up. If Network Explorer is already running, remember to refresh the window, press the <F5> key.

   ■   Otherwise you will have to stop and unload the application, and then use instrument's resident **File Manager** utility and a storage media device. (If you need reminding, Chapter 1, *Getting Started*, tells you how to do this.)

9   Three new files have been created, BATCH.UYB (the batch file), BATCH.UYS (the empty setpoint program file), and BATCH.UYR (an empty recipe file). The recipe file was auto-created because you opted to link a recipe block to the batch (in step 5 above). You now have to edit this default recipe file to suit the batch application.

6.2.1 Editing the default recipe file

As in the previous recipe tutorial we will simulate ramping the temperature of a vat according to a set of recipes. Because the batch phases will be defined by a setpoint program, the recipes cannot change ramp rates or segment durations. But they can, for example, alter setpoints (via setpoint trim values) and vary alarm levels to suit each recipe in the set.

Table 6-3 lists the recipe values you should now configure for this part of the tutorial.

To do this:

1   Press the **Menu** key, then **RECIPE**. In the **Recipe** popup press **RECIPES** to see the **Load/Save Recipe** screen.

2   Press the **File Name** field and select the automatically-created 'BATCH' recipe set file from the picklist. Press **LOAD** then **EDIT** to see the empty 'BATCH, recipe editor table.

3   Press the **RCP** column heading, then press **INSERT** in the popup to see the **Insert Variable** dialog. Enter a **Variable Name** of 'trimsp' and link it to the 'pid.trimsp' LIN database tag. **OK** your entries.

> *Tip!   Remember, tags are not case-sensitive.*

4   Press the '**1**' column heading and rename it 'hotmix'. **OK** your edit. Press the default '**0.0**' value and change it to '20.0'.

5   Continue adding variables and editing the recipe set according to the values given in Table 6-3. If you need reminding how to do this, see the *Creating a recipe set* section in Chapter 5.

6   Save the completed 'BATCH' recipe set by pressing **SAVE**. Confirm the overwrite with **OK**.

Variable (RCP)	Tag Reference	Hotmix	Warmmix	Coolmix
trimsp	pid.TrimSP	20	10	0
hiabs	pid.HAA	100	90	80
loabs	pid.LAA	40	30	20
hidev	pid.HDA	10	5	2
lodev	pid.LDA	10	5	2
id	Mixers.Filepath	hotmix	warmmix	coolmix
Note The 'id' variable will be used when you customise the batch later on.				

Table 6-3   'BATCH' recipe set data for batch application

6.2.2 Replacing the default setpoint program file

You now have everything in place except a valid setpoint program file. All you must do is replace the empty program file automatically created by the instrument, 'Batch.UYS', with the one you modified in Chapter 4, *Data logging*. You may have named this file 'program2.uys'.

> *Tip!   Rename the file 'batch.uys' so that it will be recognised by the batch program.*

To do this using Network Explorer:

1   In the Computer, display the contents of instrument's E: drive in Network Explorer. Delete the empty 'BATCH.UYS' file by right-clicking it and selecting *Delete*.

2   Rename the 'program2.uys' setpoint program file as 'batch.uys' and download it to the E: drive.

If you don't have a working Computer-instrument link, stop and unload the application then use **File Manager** and a storage media device to carry out the file operations.

6.3 RUNNING THE BATCH APPLICATION

To run the (simulated) batch application and try out some of instrument's batch features, do the following:

1   If you had to stop and unload the 'BATCH' application for filing operations, reload and run it now. Otherwise go straight on to step 2. The area overview appears with its two group faceplates, 'pidloop' and 'preplot'.
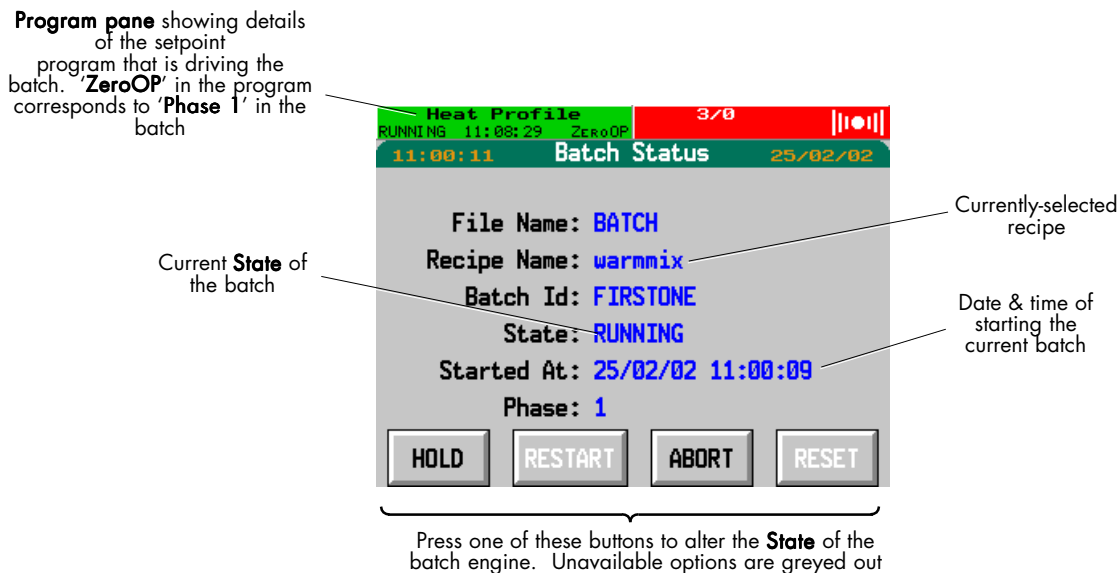
Figure 6-2   Batch Status screen (phase 1 in the 'FIRSTONE' batch)

2   Press the **Menu** key, then **BATCH**, **BATCHES**. In the **Load Batch** screen press the **File Name** field and select 'BATCH'. Press **LOAD** to see the **Batch Start** screen. This tells you the filename of the loaded batch, the recipe currently selected (defaults to the first in the set), and an automatically-generated batch identity, '**00000001**' if this is your very first batch.

3   You can select another recipe if you wish, and also edit the Batch Id. Press the **Recipe Name** field ('hotmix') and select the 'warmmix' recipe instead. Edit the **Batch Id** field to 'FIRSTONE'.

4   Press **START** to start the batch. The **Batch Status** screen appears, telling you what is currently happening with the batch. Figure 6-2 shows what you might see at this initial stage.

> *Note   The batch state shows RUNNING and it is at Phase 1, corresponding to the first segment of the setpoint program called 'ZeroOP'. The program pane tells you what is happening with the setpoint program.*

5   Press the **Menu** key, then **OVERVIEW** to see the Area display. Press the 'pidloop' faceplate then the **Down** key until you see the horizontal trend display. Watch this for a while. You should confirm that the 'vat' temperatures correspond to the 'warmmix' recipe. The 'ZeroOP' phase setpoint should be at 10 degrees (programmed zero plus setpoint trim of 10), the 'soak' phase at 90 degrees (80 + 10), and the 'stand' phase at 30 degrees (20 + 10).

6   Return to the **Batch Status** screen by pressing **Menu**, **BATCH**, **STATUS**. As the batch continues running, note the phase-changes and their correspondence to the setpoint program segments.

7   To hurry things along try pressing the top-left program pane, then **SKIP** to the next program segment. As the batch is program-driven, it also skips a phase when you do this. When the batch has run its course, its state becomes **COMPLETE** (and the grey-coloured program pane shows that the program is **IDLE**). The Batch Status screen now reports the end date and time.

8   Take a look at the preplot screen, press the program pane then **PRE-PLOT**. You will clearly see the action of the 'warmmix' recipe 10% setpoint trim.

6.3.1 Exploring batch states

If you have let the batch run its course your screen should look like Figure 6-3, with the batch engine state at COMPLETE. Now only the **RESET** button is active.
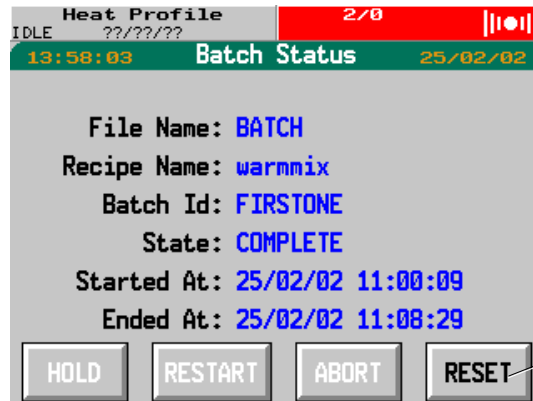
In the COMPLETE state, RESET is the only active button — see Figure 6-4

Figure 6-3   Batch Status screen — batch in the COMPLETE state

Figure 6-4 shows the instrument batch engine state diagram. The different states are shown in boxes linked by labelled arrowed lines. In most cases you can get from one state to another by pressing the button identified by the relevant label. E.g. the only way to get from RESET to IDLE is to press the LOAD button. In the RESET state, no other button is active.

> *Note*   *Some states are* transitory, *i.e. they move on to another state spontaneously after a delay (which is often too small to notice). That's why not all arrowed lines are labelled. Strictly, the RUNNING state is transitory, but it usually lasts a considerable time — as long as the batch.*
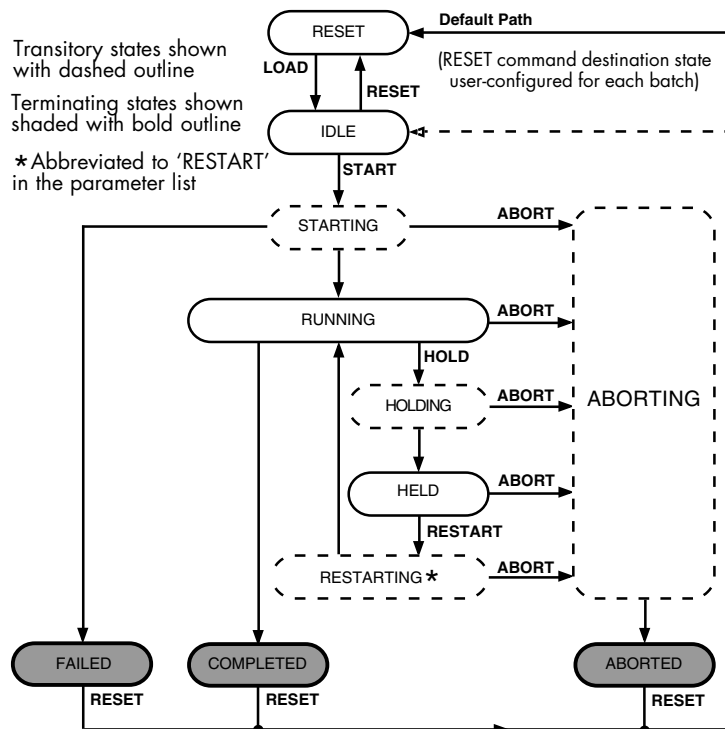
Figure 6-4   Batch engine state diagram

Try running the batch again, to check out the operation of the state diagram.

1   From the COMPLETE state, press **RESET** to get the batch engine into the RESET state. This is the only possibility, see Figure 6-4. The **Batch Status** screen appears.

2   You can see from the state diagram that the only route out of the RESET state is to **LOAD** the batch, again. To do this, press the **Menu** key, then **BATCH**, **BATCHES**, **File Name** field, and select 'BATCH'. Now you can press the **LOAD** button, which gets the batch into the IDLE state according to the diagram. In this state the **Batch Start** screen appears.

3   In the **Batch Start** screen you have two possibilities, **START** and **RESET**. **SAVE AS** does not affect the batch state. Figure 6-4 shows where each of these leads. Press **START** to move to the RUNNING state.

4   Now try pressing **HOLD**. The batch passes very rapidly through the transitory HOLDING state, you won't notice this, into the HELD state. Note that the program pane shows that the setpoint program has also gone into HELD as well.

5   From HELD you can either 'abort' or 'restart' the batch. Try either option and notice that in this application the relevant transitory states are passed through too quickly to be observed.

> *Note*   *You can see from Figure 6-4 that pressing RESET from any of the terminating states (FAILED, COMPLETE, and ABORTED) moves the batch state either to RESET (the default) or to IDLE (if you configure it so). An advantage of the IDLE state is that you don't have to reload the batch to run it again. But the advantage of returning to RESET is that all data is re-read from disk, ensuring that any online modifications are discarded, that may be important. How to configure this option is explained in Chapter 7.*

6.4 ENHANCING THE BATCH APPLICATION

In this section you add three blocks to the LIN strategy and customise the batch in a number of ways to provide additional functionality, specifically:

■ Batch reporting, using a DR_REPRT block

■ Customised Batch Start and Status screens, via the batch file and user dictionary

■ Batch start confirm dialog, via the batch file

■ Customised batch phase names, via the batch file

6.4.1 Modifying the batch strategy

Figure 6-5 shows the batch strategy you created earlier in this chapter, but with three additional blocks to provide some of the enhancements and customisation listed above.
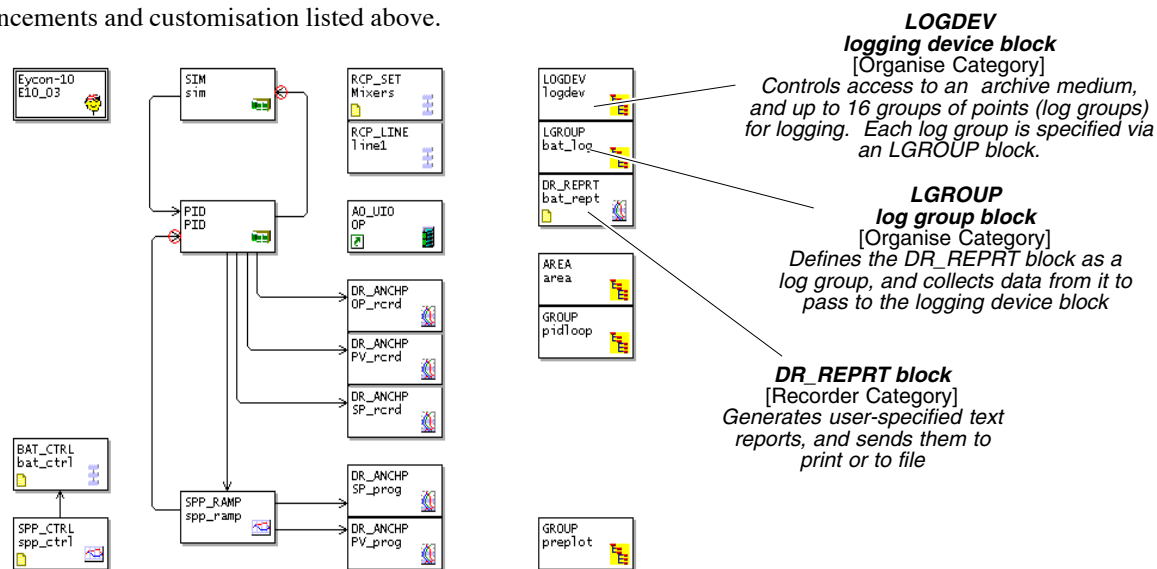
Figure 6-5   The enhanced batch strategy

To enhance the strategy, do the following:

1   In LINtools, open up the existing batch strategy, add the three new blocks and parameterise them according to Table 6-4. Note that you must also edit the **Name** fields in all blocks, according to Figure 6-5 if you wish.

Source*	Function
(LOGDEV) logdev.State = **OFF**	Prevents access to the storage media device when application starts running. The batch engine itself enables access when the batch is started
(LOGDEV) logdev.Group1 = **bat_log**	Assigns the 'bat_log' LGROUP block to Group 1 of the LOGDEV logging device block
LGROUP) bat_log.UpdateA = **10**	Specifies the 'bat_log' block's standard logging interval as 10 seconds (the minimum time for a single group). The default rate of zero disables logging
(LGROUP) bat_log.Point1 = **bat_rept**	Assigns the 'bat_rept' report block to Point 1 of the 'bat_log' log group, to provide text reports for the group
(LGROUP) bat_log.ColTitle = **None**	Suppresses unwanted column heads in batch reports
(DR_REPRT) bat_rept.FormFile = **batch**	Specifies the 'batch.uyf' form file as the formatting source for the generated group reports
*Note *(Block type) blockname.field.subfield format, referring to Figure 6-5.*	

Table 6-4   Values for non-default block fields in the enhanced batch strategy

2   When you've carried out all necessary field edits, save the modified batch strategy under its original filename, i.e. '**batch.dbf**'. Download it to the instrument, overwriting the existing database file.

6.4.2 Modifying the batch file

Most of the batch customisation is done by editing the comma-separated **batch.uyb** file in a text editor or spreadsheet. To do this:

1 Open up the 'batch.uyb' file in 'Notepad' or similar editor/spreadsheet. (Right-click the file in Explorer and select *Open With*. Then select *Notepad*.) The file should look something like the text window shown in Figure 6-6.

2 Each line of the file does a different job, as indicated in the figure. The selections you made when you first created the batch file determined the contents of lines 4 to 6.

> *Note* *The* Eycon-10/Eycon-20 Handbook *gives full details on batch file syntax. Please refer there if you need to know more.*
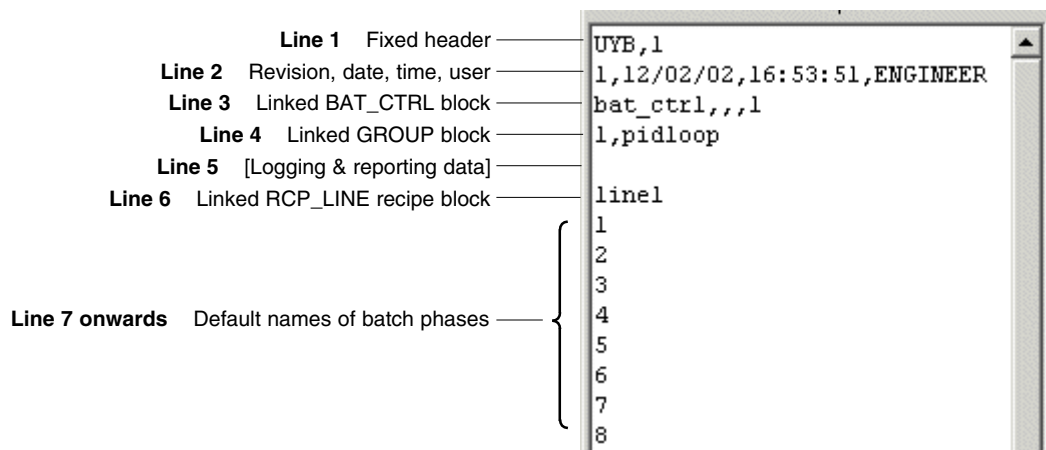
Figure 6-6   The batch.uyb file before editing

3 Start by editing line 3 of the file, according to Figure 6-7, which shows the fully-edited batch file. The three commas are vital as they ensure that the '2' occupies the *fourth* field in the line. This field determines what user confirmation is required when the batch is started.

```
'0' [or blank field] = No confirmation needed (the default option),
'1' = Batch Start Confirm dialog appears with OK/CANCEL buttons,
'2' = Batch Start Confirm dialog with Password entry field and OK/CANCEL.
```

4   In line 4, add fields 4, 5, 6, and 7 exactly according to Figure 6-7. Each of these fields defines an extra line in the **Batch Start** screen. Each field consists of three parts separated by colons (**:**), as follows:

■   Title.   This refers to a line in a *User Dictionary* containing the text you want to act as a title or fieldname. E.g. **#U3** refers to the text string '**Contact:** ' in the user dictionary that you will create in the next section of this tutorial. Dictionary references starting with '#U' always refer to read-only field titles.

Note   You should always append a colon *and a* space *to your field titles, so that they appear nicely formatted on the screen.*

Confirm level (2 = Password en-
try)
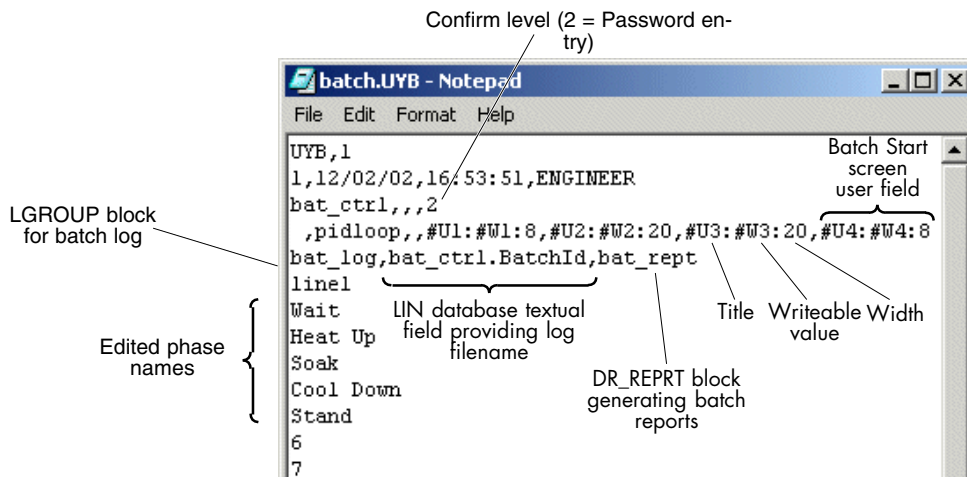
Figure 6-7   The edited batch.uyb file

■   Writeable value.   This refers to a line in the User Dictionary containing the text you want displayed as the field value. E.g. **#W3** refers to the text string '**Jim Bond**' in the user dictionary. Dictionary references starting with '#W' always refer to operator-writeable field values.

■   Width.   This number specifies the number of characters you want allowed for the field value, e.g. '20' results in the **Contact** field being 20 characters wide.

Note   References to user dictionary entries rather than to fixed text allow you to switch display language simply by swapping the active user dictionary.

---

5   Edit line 5 (currently blank) by entering the name of the log group block for the batch ('bat_log') in field 1. Field 2 contains a LIN block reference, the *bat_ctrl* block's *BatchId* field, to be the source of the batch log filename. So each batch report file will be named after the ID of the batch producing it, which makes sense. Field 3 names the DR_REPRT block generating the batch reports.

6   Finally, edit lines 7 to 11 to show meaningful phase names corresponding to each segment of the setpoint program. Use the ones suggested in Figure 6-7 if you wish, they need not be the same as the setpoint program's segment names. If you delete ***all*** the default numeric phase names, instrument uses the program segment names instead.

7   Carefully check that you have edited the batch file correctly (a single misplaced comma could invalidate it), then save it under its current name, '**batch.uyb**'. Download the file to the instrument.

---

*Note   Some editors automatically append the '**.txt**' extension to saved text files. Restore the batch filename to the correct '**.uyb**' format if this happens.*

6.4.3 Creating a user dictionary

You now create the *user dictionary* mentioned in the previous section of this tutorial.

User dictionaries let you enter your own custom texts for display on Home pages, User screens, Batch screens, etc. Additional dictionaries to hold versions in other languages can be configured if required. User dictionaries have filenames based on '**_user.uyl**', and can be created in any suitable text editor or spreadsheet

> *Note*   *The 'home language' dictionary (e.g. English) is named **_user.uyl**. Foreign language versions are named **_user0.uyl**, **_user1.uyl**, **_user2.uyl**, and so on for up to ten alternative languages.*

To configure a user dictionary for this batch:

1   Open a blank text editor window, e.g. in 'Notepad', and type in the text shown in Figure 6-8. Each entry occupies one line and has the format **<entry type><index number>,<text string>**. The text string may be blank, if no (initial) value is to be displayed.
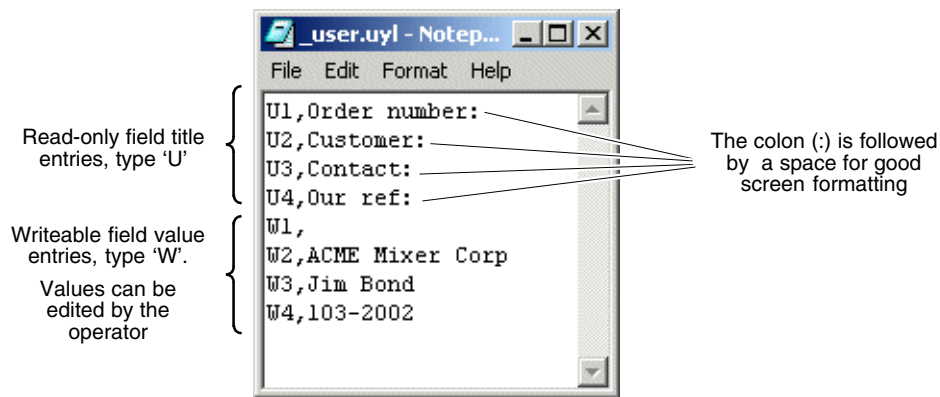
Figure 6-8   The configured '_user.uyl' user dictionary file

2   Save the file as text and edit its filename to '**_user.uyl**'. Download the dictionary file to the instrument.

6.4.4 Creating a form file for batch reports

The last thing you need do before testing out the enhanced batch application is to create a report '*form file*'. This is a comma-separated text file with a filename that is the same as the .uyb batch file and extension '**.uyf**'. One use of a form file is to customise the layout and formatting of information in the group reports generated by a DR_REPRT block. The form file can specify the inclusion of LIN database variables, system variables, e.g. current date and time, and text, optionally internationalised.

One of the fields parameterised in the DR_REPRT block, *FormFile,* referred to the '**batch.uyf**' file, see Table 6-4.

To create this file:

1   Open a blank text editor window (e.g. in 'Notepad') and type in the text exactly as shown in Figure 6-9.

To successfully format a batch report, the file must be partitioned into three sections, identified by 'report numbers' ***I1**, ***I2**, and ***I3**. The first section formats a report generated when the batch engine enters the STARTING state (the 'Start Report'). The second section ('Stop Report') activates when the COMPLETE state is entered, and the third ('Abort Report') if and when ABORTED is entered. ***This arrangement is mandatory for batch report form files.*** The *contents* of each report section is optional. In the example in Figure 6-9:
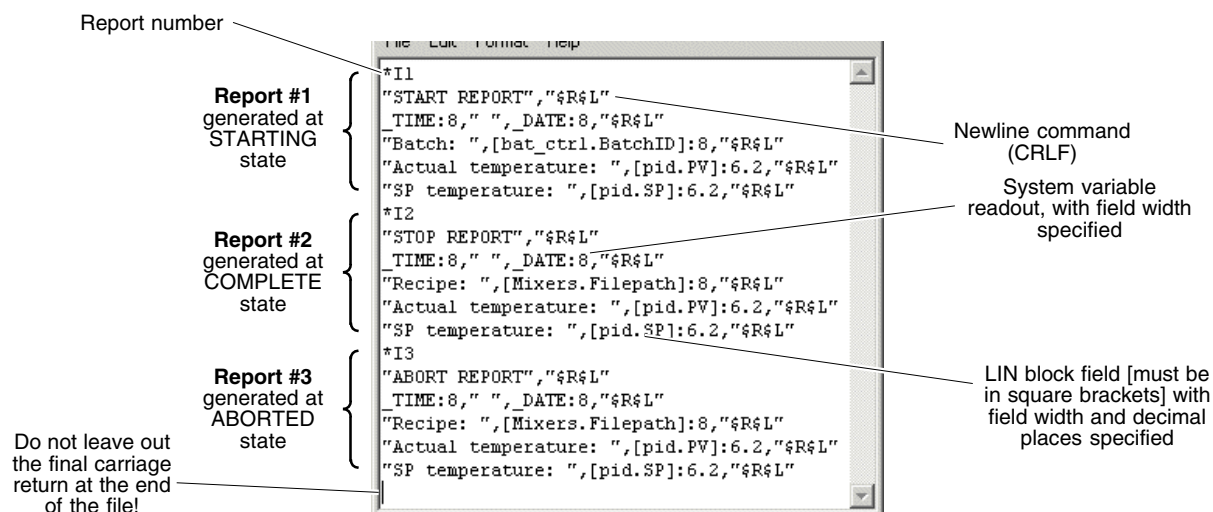
Figure 6-9   The configured 'batch.uyf' report form file

- ◼ **"START REPORT","RL"** generates a text string followed by a newline (carriage return + line feed) command. For some printers the CRLF command can take the simpler form '**$N**'. Note that commas without spaces must be used to separate the fields in a line.

- ◼ **_TIME:8,** generates a display of the current time from the system variable _TIME. The ':8' specifies a field width of 8 characters, sufficient for the hh:mm:ss time display. Variables must always be followed by a field width figure.

- ◼ **" ",_DATE:8,"RL"** generates a 'space' character followed by a dd/mm/yy current date display (from the _DATE system variable) eight characters wide. Another CRLF ends this line of the report.

- ◼ **"Batch: ",[bat_ctrl.BatchID]:8,"RL"** generates the string 'Batch: ' followed by an 8-character wide readout of the LIN block variable *bat_ctrl.BatchID*. This variable holds the current batch identification number. Note that in form files LIN block variables must always be enclosed by square brackets. As usual, a CRLF ends this line of the report.

- ◼ **"Actual temperature: ",[pid.PV]:6.2,"RL"** generates the string 'Actual temperature: ' followed by a readout of the LIN block variable *pid.PV* (the PID block's process variable). The '6.2' specification means 6 characters wide (including decimal point) with 2 decimal places displayed, e.g. '**123.45**'.

The rest of the form file coding follows this syntax

> *Note*   *The* Eycon-10/Eycon-20 Handbook *gives full details on form file syntax. Please refer there for more information.*

2   When you have checked that the form file is correct, save it as text and edit its filename to '**batch.uyf**'. Download the report form file to the instrument.

6.5 RUNNING THE ENHANCED BATCH APPLICATION

Now that you have configured the extra files and downloaded them to the instrument, you are ready to try out the enhanced batch application.

1   To force the instrument to recognise the new user dictionary ('**_user.uyl**') you must power down and then power up the unit.

> *Note*   *An easier alternative that works in this case, is to use the **Internationalise** screen. To do this, press the **Menu** key, then **SETUP**, **INTERNAT**, **CHANGE**.*

2   Get the application running by pressing **Menu**, **APP MGR**, selecting the 'BATCH' application, then **LD+RUN**. The area overview appears.

3   Press **Menu**, **BATCH**, **BATCHES**, and select the 'BATCH' file. Press **LOAD** to see the customised **Batch Start** screen, Figure 6-10.

4   You can select an alternative recipe, edit the Batch Id, fill in an Order number, or change any of the yellow (writeable) fields, before you run the batch. Try some of these options, then press **START** to start the batch.

5   A **Batch Start Confirm** dialog pops up (which you configured in the batch file). Enter your password (e.g. **[space]default**) into the **Password** field, then hit **OK**. The **Batch Status** screen appears, which is also customised. Notice that now the batch remains in the transitory STARTING state for an appreciable time. (This is because you have configured batch reporting.)

> *Note*   *If the batch then enters the FAILED state it is probably because you have forgotten to connect a storage media device, ready for data logging. Connect the device, then press **RESET**, **Menu**, **BATCH**, **BATCHES** to return to the **Load Batch** screen and start again.*
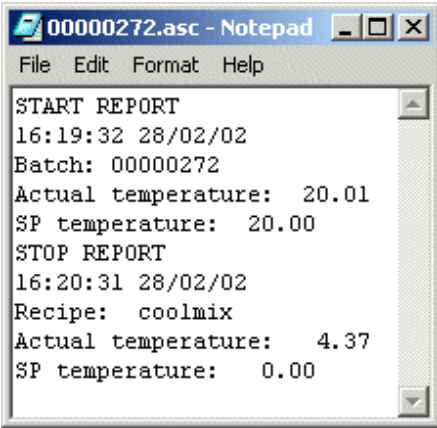
Figure 6-10   The customised Batch Start screen

6   You will notice some storage media device activity (as the 'Start report' is being generated and written to disk). After a delay the batch enters the RUNNING state, and the start date/time and customised phase name is displayed. If you let the batch progress you will see the different phase names appear in the status screen.

7   Skip to the end of the batch, press the top-left setpoint program pane, then **SKIP** repeatedly until the program is IDLE. The batch state is now COMPLETE.

8   At the Computer, explore the report contents. Open up the report in 'Notepad'. Figure 6-11 shows the sort of thing you might see. If you compare this with the form file, see Figure 6-9, you will see that the 'Start report' and the 'Stop report' sections have been used. No 'Abort report' appears because you did not abort the batch. Try this next.

> *Note*   *The 'Recipe' field is read from the LIN block* Mixers.Filepath *field. This is effectively a spare string field, to which the recipe downloads its name, 'id' in the recipe set, shown in Table 6-3.*

Figure 6-11   A batch report with 'Start' and 'Stop' sections

9   Now at the Visual Supervisor, on the **Batch Status** screen, press **RESET** to reset the batch, then **Menu**, **BATCH**, **BATCHES**. Select and **LOAD** the 'BATCH' file, and press **START**. Confirm the batch start with your password and **OK**.

10   When the batch is running, press **ABORT** to get the batch state to ABORTED. Now inspect the batch report for this batch in the Computer. You will see that this time it includes a 'Start report' and an 'Abort report', but no 'Stop' report.

6.6 WHAT NEXT ?

In Chapter 7, *User Screens*, you will add a user screen to your batch application, so that you can see graphically what is happening to the batch. After that, in Chapter 8, *Sequences*, you can have a go at creating a simple sequence (SFC) to drive the batch engine, instead of using a setpoint program. In this approach the sequence 'steps' define the batch phases. This is more work but gives you a lot more flexibility, very useful in certain applications.

CHAPTER 7 USER SCREENS

In this chapter you try out the *User Screen Editor* and create a simple user screen to enhance the batch application developed in the previous chapter. You also configure an *operator message* linked to the batch operation, and further customise the batch interface.

User screens are graphics displays that you can design to appear on instrument's screen. As well as the relatively simple objects you draw using the package, you can also include bitmap images. Screen objects can be linked to block fields in the LIN database, and to system variables, to obtain animation effects, e.g. touch-sensitive buttons, dynamic bargraphs, colour changes, numeric and text readouts, and many others.

> *Note* *This chapter explores only a few of the User Screen Editor capabilities. You should consult the* User Screen Editor Handbook, *for full details of all the editor's features.*

The main sections in this chapter are:

- Modifying the batch strategy for messaging

- Creating a user screen

- Adding messages to the user dictionary

- Modifying the batch file

- Running the batch application with user screen

7.1 MODIFYING THE BATCH STRATEGY FOR MESSAGING

Figure 7-1 shows the (simulation) batch strategy you created in Chapter 6, but with three extra blocks, AND4, PNL_MSG, and PULSE, added to give the strategy some panel messaging capability. Panel messaging is the presentation to the operator of on-screen messages triggered by events in the application. These messages can be configured to offer the operator a choice of responses.

In this case we want an operator message to appear when the batch enters a 'dangerous' condition, i.e. when the measured 'vat temperature' (PV) is in both high absolute alarm and high deviation alarm. The message dialog is to display buttons allowing the operator to either ignore the message or abort the batch.

7.1.1 Configuring the strategy

1   Open up the batch strategy you built in Chapter 6 ('**batch.dbf**') in the LINtools database configurator, by double-clicking its icon in Explorer view. Add the three extra blocks as shown in Figure 7-1.

2   Wire up the new blocks according to Table 7-1, which as usual lists all the necessary strategy connections for completeness, in case you are starting from scratch.

AND4 block
[Logic Category]
Performs the logical AND function on up to four digital inputs.

PNL_MSG block
[Operator Category]
Configures and queues an operator message, and acts on any response.

PULSE block
[Logic Category]
Outputs a rectangular pulse when triggered by a digital input.
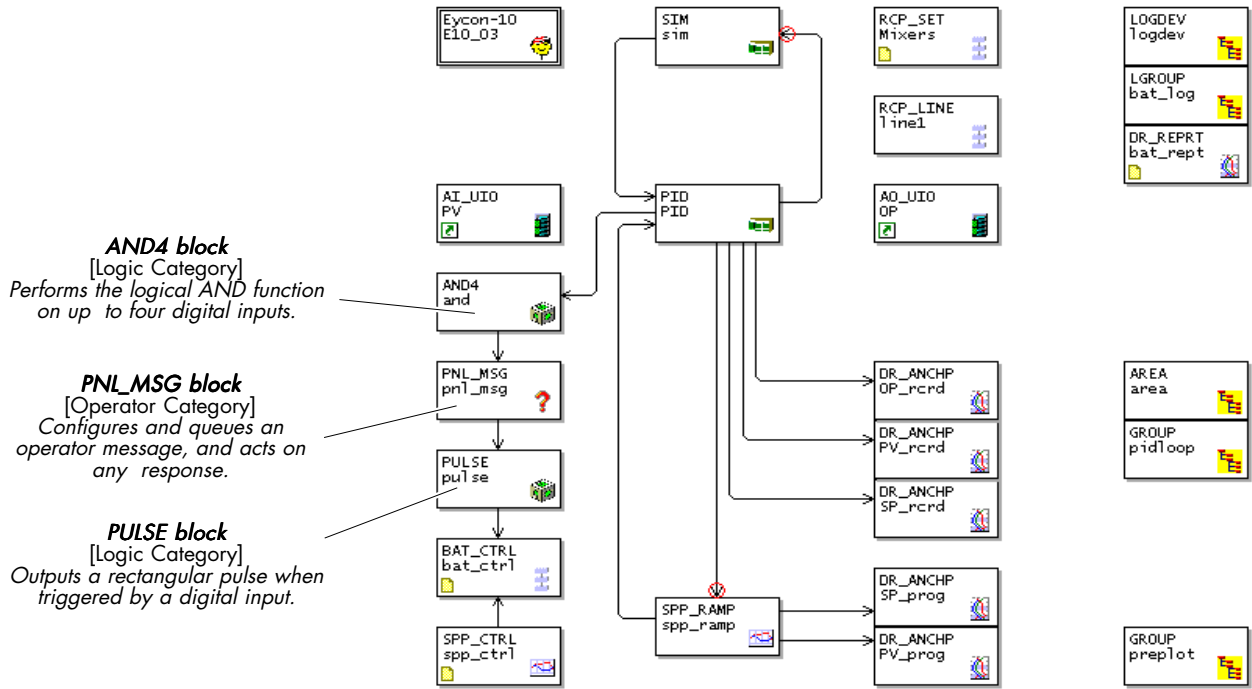
Figure 7-1   The batch strategy modified for messaging

Note   In the table, the wires below the dashed line are the new ones, so you need only configure these if you are starting from the existing batch strategy created in Chapter 6.

Source*	Destination*	Function
(PID) pid.OP	(DR_ANCHP) OP_rcrd.CurrVal	PID control output OP into the OP_rcrd (DR_ANCHP) block to enable data recording onto instrument's local flash storage
(PID) pid.PV	(DR_ANCHP) PV_rcrd.CurrVal	PID process variable PV into the PV_rcrd block to enable data recording
(PID) pid.SP	(DR_ANCHP) SP_rcrd.CurrVal	PID setpoint SP into the SP_rcrd block to enable data recording
(PID) pid.PV	(SPP_RAMP) spp_ramp.PV	PID process variable PV into the spp_ramp block to include PV in the 'preplot' group
(PID) pid.OP	(SIM) sim.PV	PID control output OP into the SIM block's PV input to simulate output to the plant
(SIM) sim.OP	(PID) pid.PV	SIM block's lagged output OP into the PID block's PV input to simulate the plant's PV
(SPP_RAMP) spp_ramp.Out	(PID) pid.RemoteSP	setpoint generated by the locally-running setpoint program (e.g. 'prog1.uys') into the PID block's remote setpoint input
(SPP_RAMP) spp_ramp.Out	(DR_ANCHP) SP_prog.CurrVal	setpoint generated by the locally-running setpoint program into DR_ANCHP block to enable trending & Visual Supervisor's 'Preplot' facility
(SPP_RAMP) spp_ramp.PV	(DR_ANCHP) PV_prog.CurrVal	PID process variable PV (sourced from the PID block) into DR_ANCHP block to enable trending & Visual Supervisor's 'Preplot' facility
(SPP_CTRL) spp_ctrl.CurrSeg	(BAT_CTRL) bat_ctrl.CurPhase	SPP_CTRL block's currently-active segment field CurrSeg into the BAT_CTRL block's CurPhase field, to make the setpoint program drive the phase of the batch engine
(PID) pid.Alarms.HighAbs	(AND4) and.In_1	PID block's high absolute alarm output into the AND4 block's first logic input
(PID) pid.Alarms.HighDev	(AND4) and.In_2	PID block's high deviation alarm output into the AND4 block's second logic input. If and only if both inputs are high (TRUE), the block's output is high.
(AND4) and.Out	(PNL_MSG) pnl_msg.Trigger	AND4 block's logic output into the PNL_MSG block's Trigger input. A high input triggers a predefined message which joins a message queue.
(PNL_MSG) pnl_msg.Response.ABORT	(PULSE) pulse.In	PNL_MSG block's ABORT response output into the PULSE block's input, which triggers a 1-second rectangular output pulse. (A pulse is used because the ABORT bit latches high until 'Trigger' is re-asserted)
(PULSE) pulse.Out	(BAT_CTRL) bat_ctrl.CmndFlg.ABORT	PULSE block's 1-second output pulse into the BAT_CTRL block's ABORT command input. This puts the batch engine into the transitory ABORTING state

Note   *(Block type) blockname.field.subfield *format, referring to Figure 7-1. New wiring is listed below the dashed line.*

Table 7-1   Batch (messaging) Strategy wiring data

Note   *The instrument header block and the AREA, GROUP, RCP_SET, RCP_LINE, LOGDEV, LGROUP, and DR_REPRT blocks don't need to be wired up. They are associated with other blocks via field values (see next).*

7.1.2 Editing the block fields

Table 7-2 lists the fields that should, as a minimum, be edited from their default values. For completeness, all such fields are shown in the table — including those edited in the original batch strategy.

> *Note* *In the table, the fields below the dashed line are the new ones, so you need only edit these if you are starting from the existing batch strategy created in Chapter 6.*

You must also edit the **Name** fields in all blocks, according to Figure 7-1 if you wish.

When you've carried out all necessary field edits, save the modified batch strategy under its original name — '**batch.dbf**'.

Field Value*	Function
(PID) pid.SelMode.SelRem = **TRUE**	Selects remote mode of operation for the PID block, needed to allow the program's remote setpoint to be used
(PID) pid.SelMode.EnaRem = **TRUE**	Enables remote mode of operation for the PID block
(PID) pid.Alarms.HighAbs = **1**	Enables the 'high absolute' alarm on the pid block's PV output
(PID) pid.Alarms.LowAbs = **1**	Enables the 'low absolute' alarm on the pid block's PV output
(PID) pid.Alarms.HighDev = **1**	Enables the 'high deviation' alarm on the pid block's PV output
(PID) pid.Alarms.LowDev = **1**	Enables the 'low deviation' alarm on the pid block's PV output
(PID) pid.TI = **5.00**	Sets the PID integral time constant to 5 seconds, to improve control in this simulated strategy
(PID) pid.HAA = **80.00**	Sets the pid block's PV output 'high absolute' alarm at 80%
(PID) pid.LAA = **20.00**	Sets the pid block's PV output 'low absolute' alarm at 20%
(SIM) sim.Lag1 = **25.00**	Sets the SIM block's 1st filter time constant to 25 seconds, to simulate lag in the plant's process variable
(AREA) area.Id = **1**	Allocates area number (only Area '1' currently implemented). No OVERVIEW appears in the instrument unless an area number has been allocated
(AREA) area.Group1 = **pidloop**	Assigns the 'pidloop' GROUP block to this area. This specifies that 'pidloop' will appear in the OVERVIEW and enables data recording for this group
(AREA) area.Group2 = **preplot**	Assigns the 'preplot' GROUP block to this area. This specifies that 'preplot' will appear in the OVERVIEW and enables data recording for this group
(AREA) area.DispMode.H_Trend = **TRUE**	Enables the horizontal trend display, needed for this application
(AREA) area.DispMode.V_Trend = **FALSE**	Disables the vertical trend display, which is not required in this application
(GROUP) pidloop.Update = **10**	Specifies the 'pidloop' group's recording sample rate (seconds). The default rate of zero disables recording
(GROUP) pidloop.Disp1 = **pid**	Specifies the PID block 'pid' as the source of the data that will be displayed in the faceplate and trends for the 'pidloop' group
(GROUP) pidloop.Chan1 = **PV_rcrd**	Assigns the recorder channel block 'PV_rcrd' to channel 1 of the 'pidloop' recording group
(GROUP) pidloop.Chan2 = **SP_rcrd**	Assigns the recorder channel block 'SP_rcrd' to channel 2 of the 'pidloop' recording group
(GROUP) pidloop.Chan3 = **OP_rcrd**	Assigns the recorder channel block 'OP_rcrd' to channel 3 of the 'pidloop' recording group
(GROUP) preplot.Update = **10**	Specifies the recording sample rate (seconds) for the 'preplot' group. The default rate of zero disables recording
(GROUP) preplot.Disp1 = **spp_ramp**	Specifies the SPP_RAMP block 'spp_ramp' as the source of the data that will be displayed in the faceplate and trends for the 'preplot' group
(GROUP) preplot.Chan1 = **PV_prog**	Assigns the recorder channel block 'PV_prog' to channel 1 of the 'preplot' recording group
(GROUP) preplot.Chan2 = **SP_prog**	Assigns the recorder channel block 'SP_prog' to channel 2 of the 'preplot' recording group

Continued...

Field Value*	Function
(RCP_SET) mixers.Line1 = **line1**	Associates the RCP_LINE block called 'line1' with the 'mixers' RCP_SET block
(RCP_SET) mixers.FileName = **mixers**	Entering the filename allows LINtools to offer a context menu to create the required file using this filename
(LOGDEV) logdev.State = **OFF**	Prevents access to the floppy disk when application starts running. The batch engine itself enables access when the batch is started
(LOGDEV) logdev.Group1 = **bat_log**	Assigns the 'bat_log' LGROUP block to Group 1 of the LOGDEV logging device block
(LGROUP) bat_log.UpdateA = **10**	Specifies the 'bat_log' block's standard logging interval as 10 seconds (the minimum time for a single group). The default rate of zero disables logging
(LGROUP) bat_log.Point1 = **bat_rept**	Assigns the 'bat_rept' report block to Point 1 of the 'bat_log' log group, to provide text reports for the group
(LGROUP) bat_log.ColTitle = **None**	Suppresses unwanted column heads in batch reports
(DR_REPRT) bat_rept.FormFile = **batch**	Specifies the 'batch.uyf' form file as the formatting source for the generated group reports
(LOGDEV) logdev.State = **OFF**	Prevents access to the floppy disk when application starts running. The batch engine itself enables access when the batch is started
(LOGDEV) logdev.Group1 = **bat_log**	Assigns the 'bat_log' LGROUP block to Group 1 of the LOGDEV logging device block
(LGROUP) bat_log.UpdateA = **10**	Specifies the 'bat_log' block's standard logging interval as 10 seconds (the minimum time for a single group). The default rate of zero disables logging
(LGROUP) bat_log.Point1 = **bat_rept**	Assigns the 'bat_rept' report block to Point 1 of the 'bat_log' log group, to provide text reports for the group
(LGROUP) bat_log.ColTitle = **None**	Suppresses unwanted column heads in batch reports
(GROUP) preplot.Chan3 = **[null]**	Unassigns channel 3 of the 'preplot' recording group (now unused)
(DR_REPRT) bat_rept.FormFile = **batch**	Specifies the 'batch.uyf' form file as the formatting source for the generated group reports
(PID) pid.SelMode.SelRem = **TRUE**	Selects remote mode of operation for the PID block, needed to allow use of the program's remote setpoint
(PID) pid.SelMode.EnaRem = **TRUE**	Enables remote mode of operation for the PID block
(SIM) sim.NoiseMax = **0.00**	Sets the SIM block's pseudo-random noise maximum amplitude to zero
(DR_ANCHP) SP_prog.ZoneA_LO = **0.00**	Resets zone to its default value
(DR_ANCHP) PV_prog.ZoneA_LO = **0.00**	Resets zone to its default value
(PNL_MSG) pnl_msg.Title = **1**	Identifies a reference to a dictionary entry of up to 20 characters
(PNL_MSG) pnl_msg.Body = **10**	Identifies a reference to a dictionary entry for message text
(PNL_MSG) pnl_msg.UButton1 = **20**	Identifies a reference to a dictionary legend entry for user button 1
(PNL_MSG) pnl_msg.Alarms.OK = **False**	Acknowledges the warning, but intend to continue the Batch until complete
(PNL_MSG) pnl_msg.Alarms.Abort = **True**	Acknowledges the warning, and terminates the Batch process

Note   *(Block type) blockname.field.subfield format, referring to Figure 7-1

Table 7-2   Values for non-default block fields in the batch (messaging) strategy

7.2 CREATING A USER SCREEN

In this section of the tutorial you create a user screen that presents a schematic picture of the vat and its contents, reflecting its current status. When the application is running you will be able to view this screen at any time. The user screen will include:

- A simple representation of the vat and its contents, plus a temperature readout. The colour of the contents changes to suggest changes in temperature.

- Four 'lamps' indicating high and low absolute, and high and low deviation alarms.

- Horizontal bargraphs indicating the relative PV and SP values.

- Readouts showing the current batch number, state, phase, and recipe.

- A 'time now' readout.

> *Note   User Screen Editor can also be launched from LINtools.*

Figure 7-2 shows how the finished user screen might look, near the beginning of a batch run.

Figure 7-2   The user screen in action

To create this user screen, do the following:

1   In your Computer, access the Explorer view of the working instrument folder, preferably where the '**batch.dbf**' file is located. Right-click in the list pane and in the popup menu select **New**, **Eurotherm User Screen Files**. A **.uxp** file appears with a default filename.

2   Edit the filename to '**batch.uxp**'. Double-click it to open up a **User Screen Editor** window with a **Set Initial PageSet Values** dialog.

> *Note   You can get specific **help** for an active dialog by pressing the <F1> key. For most menu items press <F1> when the cursor is over the item. To get help for a toolbutton, locate the cursor over the tool, hold down the left mouse button and press <F1>. Release the mouse button. This works even for 'greyed out' toolbuttons.*

3   If the '**batch.uxp**' file was created in the working instrument folder, all required fields are automatically configured, but if created outside the working instrument folder, set the **Target panel** field to the appropriate value for your instrument target, e.g. 'qvga' or 'Eycon-10', and then if necessary change the **Page Set Name** field. Use the Browse button to configure the **Location of DB file for Page Set** field. Leave other fields at their defaults and **OK** your edits. A blank '**batch.uxp**' window now appears.

> *Note   If you wish to change the Target Panel, it is recommended that the Target Panel is specified before the Name is edited.*

4   Click the 'New Page' icon ⬒ to pop up a **New Page** dialog. Edit the dialog fields according to Figure 7-3, most of the fields are left at their defaults. **OK** your edits. A blank '**Page1**' window appears, ready for building the user screen.
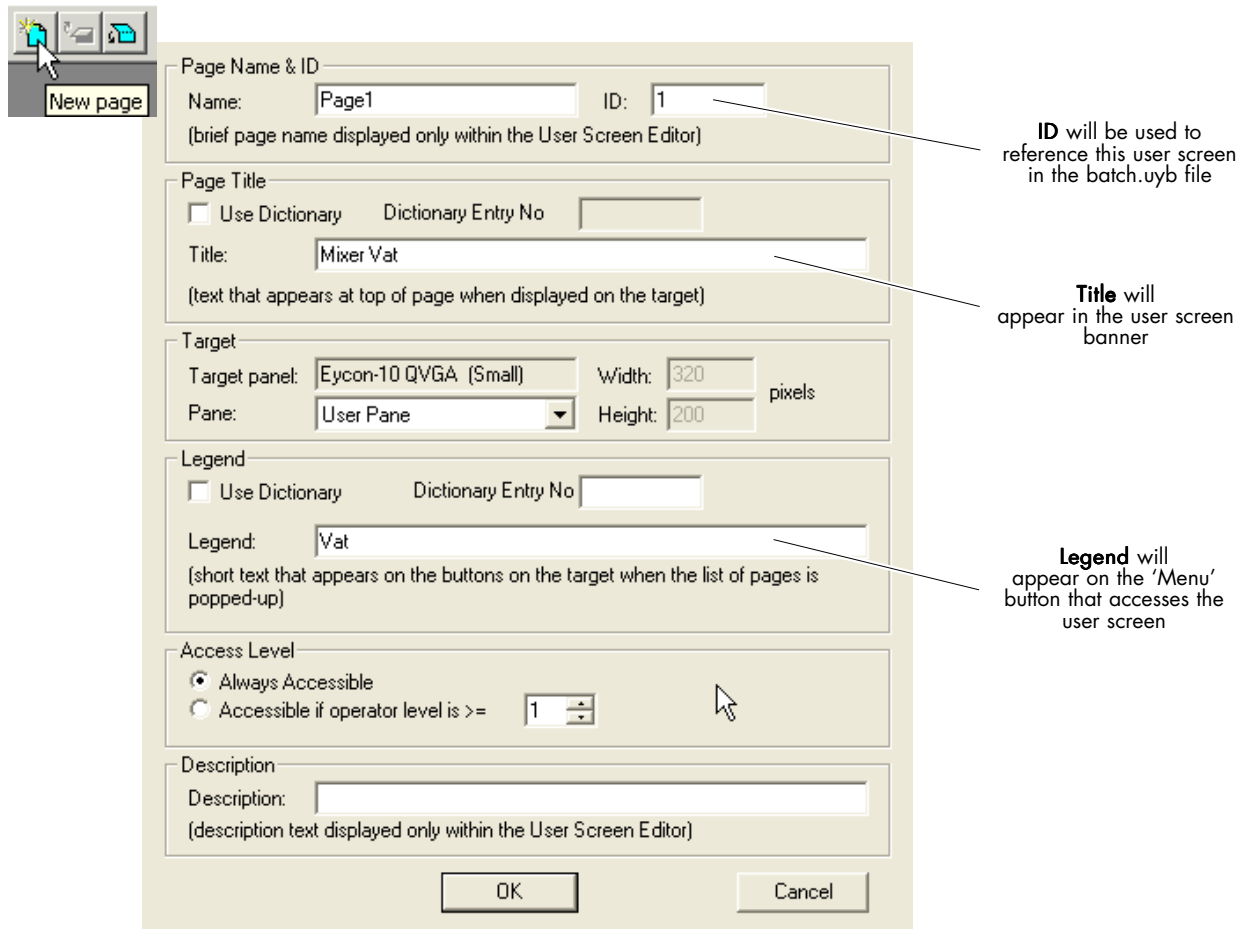
Figure 7-3   Accessing and filling in the 'New Page' dialog

7.2.1 Creating the 'vat' graphic

1   If the Target panel is set to 'qvga' or 'Eycon-10', start by zooming the scale to 200% using the zoom box ▢200% ▾ at the top of the editor window to make drawing easier. Drag the editor window sides/corners if necessary to see the whole gridded screen area.

> *Note*   *Hovering the cursor over a toolbutton shows a 'tool tip' box with the name of the tool. The status bar at the foot of the window summarises the tool's action.*

2   Click the **Ellipse** tool ◯ and draw (drag) a circle of diameter equal to four grid squares. The 'snap to' grid should make this operation very easy.

> *Note*   *You can toggle the visibility of the grid lines by clicking the **Toggle Gridlines** button ▦. But even with the grid invisible its 'snap to' action still works. To switch off 'snap to' use the **Grid Settings** button ▢.*

3   Click the **Selection** tool ▱ at the bottom left corner of the editor window, then right-click in the circle you have just drawn to pop up a context menu. Select **Properties...** to see the **Ellipse Item Properties** dialog.

4   In the **Appearance** page, set the **Line/Text Weight** to '3'. Leave the other fields at their defaults and click **OK**.

5   Now select the **Line** tool ◸ and draw a vertical line 3 grid squares high, to form the left side of the vat. Repeat for the other side of the vat, then draw the base of length 4 grid squares. Figure 7-4 shows some stages in the drawing of the 'vat'.

With the 'snap to' grid switched on (the default), add three lines to the circle to form the vat outline.

With the 'snap to' action now off, adjust the width of the circle to fit nicely to the vat sides.

Drag a handle to change the shape of the circle/ellipse.

Draw a rectangle just inside the lower part of the vat, to represent the vat 'contents'.

Link the rectangle to **pid.PV** to make it dynamic.

Place a text variable field on the graphic and link it to **pid.PV** to act as a 'temperature' readout.

Add a 'degC' text legend below the graphic.

Figure 7-4   Drawing the 'vat'

6   Click the **Selection** tool, then right-click a line and in its **Properties** dialog set the line weight to '3'. Repeat for the other two lines.

7   To make the circular 'top' of the vat fit the sides better, first click the **Grid Settings** toolbutton ⊡ and uncheck the **Snap to grid** tickbox. Hit **OK**. Then click on the circle and drag the side handle to the right or left to adjust its width. Nudge the selected object horizontally or vertically using the Computer's arrow keys, to get a satisfactory result.

7.2.2 Creating the vat 'contents'

1   Select the **Rectangle** tool ▢ and draw a rectangle to lie just inside the lower part of the vat graphic, see Figure 7-4. Click the **Selection** tool then right-click the rectangle and select **Properties...**.

2   In the **Appearance** page, set the **Weight** to '0', tick the **Object is filled** box, and set the **Background/Fill** to 'Pastel green'.

3   In the **Colour attribute** page's 'Variable selection area', tick the **Colour change** box, select the **LIN variable** radio button and type 'pid.PV' in the **Name** box. Note that the red cross becomes a green tick to show that your choice of LIN variable is valid.

4   In the 'Colour selection area', set the **Value** box to **Value < lo_limit**. Tick the 'Change' box for the **Background/Fill** field and leave its colour at the default grey.

Now set **Value** to **lo <= Value <= hi**, tick the **Background/Fill** field's 'Change' box and select 'Orange' as the fill colour for this condition. Repeat for the **Value > hi_limit** condition but set the fill colour to 'Red'.

5   In the **Colour Limits** page's 'Low Colour Limit' area, select the **Constant** radio button and enter a **Value** of '81'. In the 'High Colour Limit' area, select **Constant** and enter a **Value** of '91'. Leave all other fields at their defaults. Finally, click **OK** to input your edits.

> *Note*   *The result of this configuration is that the 'vat contents' will adopt a grey colour when PV is less than 81 ('degrees Celsius'), an orange colour when PV lies between 81 and 91, and a red colour when the temperature exceeds 91.*

7.2.3 Adding text to the vat

1   Select the **Text** tool 𝐴𝑎 and click near the top of the 'vat' to place a text cursor there. Type 'degC'. Click the **Selection** tool ▹ then right-click the text and select its **Properties**. In the **Text** page, set the **Font name** to 'SmallBold' and the **Alignment** to 'Left Centre'. Click **OK**. The **Fill** adopts the 'Pastel green' colour by default.

2   Move the text block into the correct position using the arrow keys.

3   Click the **Text Variable** tool ▫ᵃᵇ , position the cursor above the 'degC' text block, and drag to the right to produce a row of four question marks '**????**'. Click the **Selection** tool, right-click the '**????**' and select **Properties...**.

4   In the **Variable** page, select the **LIN Variable** radio button and enter 'pid.PV' in the **Name** box. In the **Format** page, select '0' **Decimal places**, and in the **Text** page set the **Alignment** to 'Left Centre'. The **Font name** defaults to 'SmallBold'. Click **OK**.

5   If necessary, nudge the elements of the '**vat**' graphic into their correct positions, see Figure 7-4. Drag a box around the graphics to select them all, then click the **Group** button ⊞ to combine them into a single object. Drag the 'vat' to near the top centre of the screen.

7.2.4 Creating the alarm indicator 'lamps'

There are four of these. Start with the 'HiAbs' lamp:

1   Using the **Ellipse** tool, draw a circle a bit smaller than a grid square. ('Snap to' must be off to do this.) Open up its **Properties** dialog and in the **Appearance** page, tick the **Object is filled** box. The **Background/Fill** colour should be set to 'Pastel green'.

2   In the **Colour attribute** page, tick the **Colour change** box and the **LIN Variable** radio button. Enter '*pid.Alarms.HighAbs*' in the **Name** box (or use the **Browse...** button).

3   In the **Colour selection** section, select **Value** as 'Inactive Acknowlegded', tick the **Change Background/Fill** box and select 'Pastel green' as the fill colour. Then set **Value** to 'Active Ack', tick the **Change Background/Fill** box and select 'Red'. This will mean that when the high absolute alarm is inactive, the circle's fill colour will be pastel green. But when the alarm is active the fill colour will be red.

4   Now use the **Text** tool to create the 'HiAbs' legend, and position it alongside the 'lamp', see Figure 7-2. Locate the graphics near the top left of the vat.

5   Create the three other lamps in the same way, linking them to the relevant *pid.Alarms* output, *pid.Alarms.LowAbs*, *pid.Alarms.HighDev*, and *pid.Alarms.LowDev*.

> *Note*   *It is often quicker to copy and paste an existing graphic, then edit its properties, rather than draw a series of similar graphics from scratch. Use the copy* 📄 *and paste* 📋 *toolbuttons (or <Ctrl+C> and <Ctrl+V>).*

7.2.5 Creating the PV and SP bargraphs

Use the **Bar Chart** tool 📊 for this:

1   Click the **Bar Chart** toolbutton, locate the cursor on the screen and drag out a horizontal rectangle about 12 grid units long by about a half unit high. Release the button and click on a blank area of the grid to deselect the bar, then right-click on it and open up its **Properties** dialog.

2   In the **Variable** page, link the bar to the LIN block field 'pid.PV'. In the **Appearance** page, select the **Foreground** colour as 'Red' and the **Background/Fill** colour as 'Black'. In the **Bar Chart** page, set the **Fill direction** as 'Left to right'. Click **OK**.

3   Add a text legend, 'PV', to the left of the bargraph, and position the graphics just below the vat. Adjust the shape of the bar if required, by dragging its handles.

4   Copy the bargraph plus legend and paste the copy just below the original (see Figure 7-2).

> *Note*   *An easy way to do this is to select the bar plus legend, then do a copy-and-paste. This pastes the copy exactly over the original so that it doesn't look like you've done anything. Then, while it's still selected, nudge the copy downwards to the required position.*

5   Edit the second bargraph properties to link it to the *pid.SP* LIN variable, and make the foreground colour 'green'. Also, edit the text legend to 'SP'.

7.2.6 Creating the batch status readouts

These are simply text items plus text variables linked to the appropriate LIN or instrument variables. To create them:

1   Click the **Text** tool **Aa** and type in the 'Batch:' legend below the bargraphs. Open the legend's **Text Item Properties** dialog and fill in the fields according to the first item in Table 7-3. **OK** your selections.

2   Now use the **Text Variable** tool ▪ab to produce a row of ten question marks. Open the readout's **Text Variable Item Properties** dialog and fill in the fields according to the second item in Table 7-3. **OK** your selections.

Graphic	Object*	Properties		
'Batch:'	Text Item	Appearance:	Line/Text: Fill/Background:	Dark green Pastel green
		Text:	Font name: Alignment:	Medium Right
'Batch' readout	Text Variable Item	Variable: Appearance:	LIN Variable: Line/Text: Fill/Background:	Name: bat_ctrl.BatchId Blue Pastel green
		Text:	Font name: Alignment:	Medium Left

Table 7-3   Text Item Properties for the 'Batch:' graphic

Note   Giving every object a pastel green fill/background will match the pastel green background you will be adding to the whole user screen.

3   Position both legend and readout below the bargraphs and align them as shown in Figure 7-5, *item 1*. Drag a box round both items to select them, then do a copy-and-paste (Ctrl+C, then Ctrl+V). The copy is exactly superimposed over the original.

4   Nudge the still-selected copy downwards, using the 'down-arrow' cursor key, to create a second line — which will be nicely aligned beneath the first (*item 2* in the figure).

5   Edit the new line's text item to 'State:'. To do this, deselect the items by clicking on a clear area of the screen, then click the **Text** tool and click in the text item to insert a text cursor to the left of the colon character (*item 3* in the figure). Backspace out the 'Batch' legend and type in 'State' instead.

6   Repeat this procedure to create the remaining text items and readouts. Configure their **Text Item** properties (font, colour, etc.) as for the 'Batch' line (Table 7-3), and their **Variable** properties according to Table 7-4.

Note   You may want to make the 'Time now' text a different colour, e.g. black - to distinguish this readout from the others.
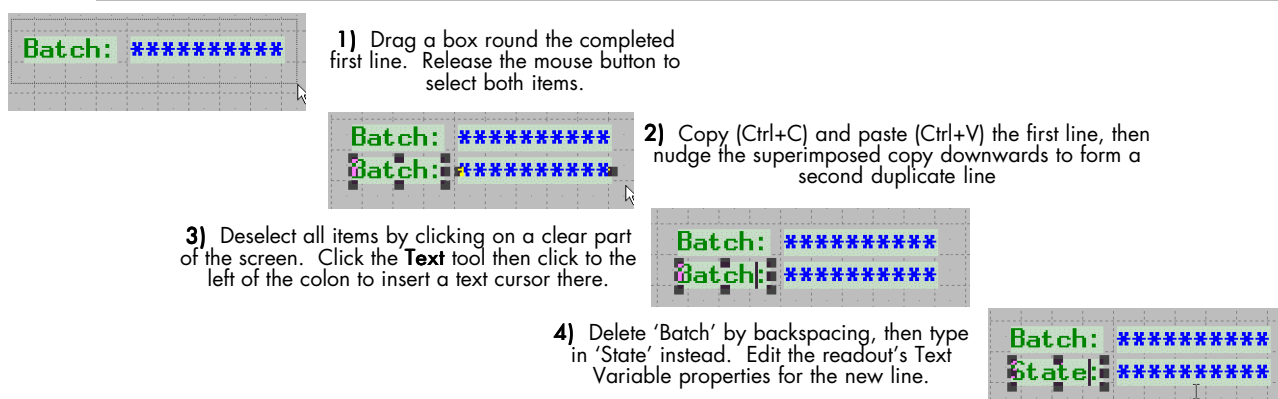
Figure 7-5   Creating a second readout line using copy-and-paste

Graphic	Variable	Name
'State' readout	LIN Variable:	bat_ctrl.State
'Phase' readout	Instrument Variable:	_SPP_SEG_S
'Recipe' readout	LIN Variable:	Mixers.Filepath
'Time now' readout	Instrument Variable:	_TIME

Table 7-4   The user screen batch status readouts

7.2.7 Completing the user screen

To complete the '**Mixer Vat**' user screen you need only add a pastel green background:

1 Ensure that the 'snap to grid' action is turned off (click the **Grid Settings** toolbutton, and uncheck the **Snap to grid** checkbox if necessary.)

2 Using the **Rectangle** tool, draw a rectangle that fills the whole screen. Drag out the edges if necessary. Open up the **Rectangle Item Properties** dialog.

3 In the **Appearance** page, set the **Line/Text Weight** to '0', tick the **Object is filled** checkbox and select 'Pastel green' for the fill colour.

4 In the **3D Styling** page, leave the **Style** as 'Bevelled, Raised' but edit the **Depth** to '3'. **OK** your edits. The green rectangle now overlays and obscures the other graphic objects.

5 Send the still-selected rectangle to the back by clicking the **Move to Back** toolbutton 🔲. Your finished user screen should now look something like Figure 7-6.
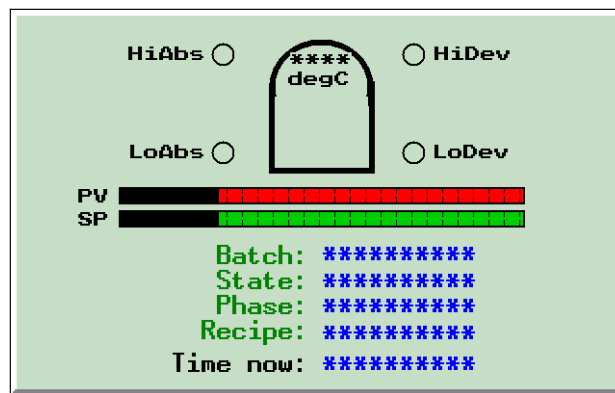
Figure 7-6   The finished 'Mixer Vat' user screen

6 Save the finished user screen by clicking the **Save PageSet** toolbutton 💾.

7 During the save process a **Build Window** appears, reporting on the progress of the save. Figure 7-7 shows a window for a successful build.
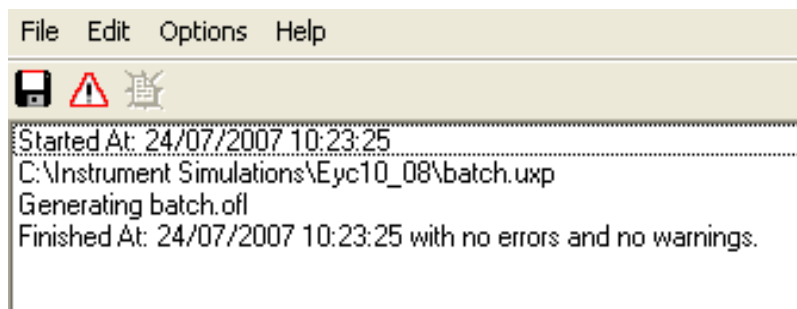
Figure 7-7   User screen editor build window — successful build

Note   If you have followed the steps in this tutorial you should not get any error messages or warnings. If you do get errors reported, they are usually self-explanatory. If in doubt, look carefully through the steps to see if you have left anything out. Check the User Screen Editor window to see if you have inadvertently created any unwanted graphics that have been left unconfigured. Do an Edit > Select All *to show up any rogue objects.*

8 Finally, download the '**batch.ofl**' file that your 'save' created to the instrument's E: drive. This text file contains all the code needed to drive the user screen. Do not download the '**batch.uxp**' file that was also created, keep this in case you want to make future edits to the user screen, or add more screens to the pageset.

7.3 ADDING MESSAGES TO THE USER DICTIONARY

You will recall that you specified several *user dictionary* references when you were parameterising the PNL_MSG block in the LIN database (see Figure 7-8). These dictionary items were to appear in an operator message. You must now add these references to the existing user dictionary — stored in the file '**_user.uyl**' — so that they can be accessed when the application is run.

To do this:

1   Open up the user dictionary file, '**_user.uyl**', in a text editor such as 'Notepad'.

2   Add three lines to the end of the existing text, referring to Figure 7-8. Note that they all begin with 'M', which tells the system that these are for *messages*. The number after the 'M' is as usual the reference number of the text string following the comma.

3   Save the edited user dictionary with its existing name, checking that the name has not been automatically appended with the '**.txt**' extension. Rename the file if necessary, then download it to the instrument.
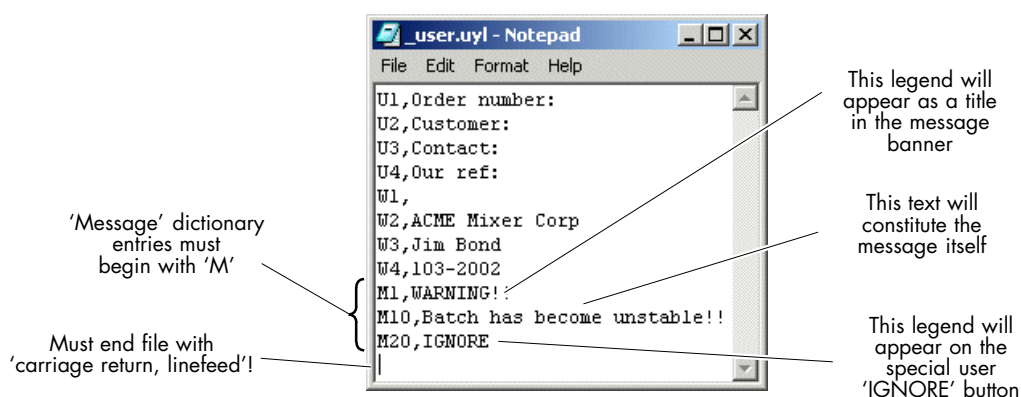
Figure 7-8   The modified user dictionary '_user.uyl'

7.4 MODIFYING THE BATCH FILE

In this section you slightly modify the batch file you created in Chapter 6, to include the user screen you have just built, and also to alter the batch's action on reset.

Do the following:

1   Open up the batch file, '**batch.uyb**', in a text editor such as 'Notepad'.

2   Edit lines 3 and 4 as highlighted in Figure 7-9.

> *Note*   *For comprehensive information on batch file syntax please refer to the* Eycon-10/Eycon-20 Handbook

3   Save the edited batch file as '**batch.uyb**' and download it to the instrument.
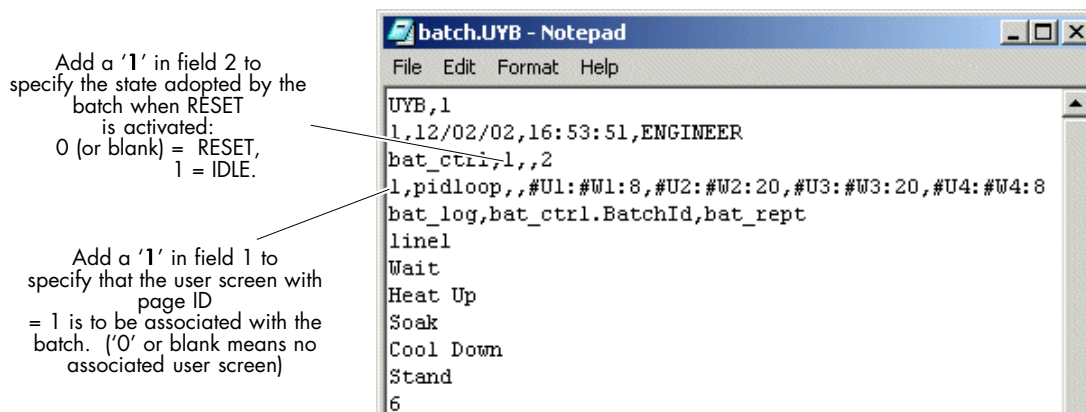
Figure 7-9   The modified batch file 'batch.uyb'

7.5 RUNNING THE BATCH APPLICATION WITH USER SCREEN

To run the (simulated) batch/user screen application and see the results of your labours, do the following:

1 Stop and unload any currently-running application.

2 Force the edited user dictionary to be recognised by power cycling the instrument.

Alternatively, press **Menu**, **SYSTEM, SETUP**, **INTERNAT**, **CHANGE**, which works in this case.

3 In the **Appl'n Manager** screen, select the 'BATCH' file and **LD+RUN** it. After a short delay your 'Mixer Vat' user screen appears instead of the usual Area overview.

4 To get a batch going, press the **Menu** key, then **BATCH**, **BATCHES**. In the **Load Batch** screen select the 'BATCH' file and press **LOAD**. The customised **Batch Start** screen appears.

5 Edit any of the yellow editable fields if you wish, but leave the **Recipe Name** set to 'hotmix', then press **START**. **OK** the confirmation popup to start the batch. The customised **Batch Status** screen tells you that the batch state is 'STARTING', which may persist for a while.

> *Note* *If the batch then adopts the 'FAILED' state it is probably because there is no storage media device connected. As you have configured batch reporting, a media storage device is expected and the batch is not allowed to proceed without one! In this case, press **RESET**, **Menu**, **BATCH**, **BATCHES**, to return to the **Batch Start** screen. Insert a media storage device, then press **START** again.*

6 Press the **Menu** key, then **Vat**, to view your 'Mixer Vat' user screen and watch the progress of the batch. You will see the alarm lamps operating with PV/SP changes, and the various readouts reporting on the batch state, phase, and so on. Note the vat 'contents' colour-changes as the temperature rises and falls. Note also the occasional disk activity as batch reporting occurs.

7 Let the batch run to its conclusion, when the COMPLETE state is adopted. You can always speed things up by skipping phases via **Program pane**, **SKIP**.

> *Note* *Skipping a phase (program segment) can have unexpected effects. If you 'skip' while ramping up or down to the next 'dwell' segment, the setpoint never reaches its intended target. It instead dwells at the value it had when you executed the skip command.*

8 When the batch is in the COMPLETE state, press **RESET**. Note that the state now adopted is IDLE, not RESET. This is because, in the previous section, you configured the 'batch.uyb' file to make the batch behave in this way. A consequence of this configuration is that in the IDLE state the current batch is not unloaded, and therefore does not need to be re-loaded to get it running again.

9 Return to the **Batch Start** screen by pressing **Menu**, **BATCH**, **BATCHES**. From here you can start another batch run, without having to select and reload the batch file.

7.5.1 Testing operator messaging

To do this:

1   In the **Batch Start** screen, edit the **Recipe Name** field to 'coolmix'. Press **START** and confirm with **OK**.

2   Press **Menu**, **Vat** to view the user screen, and follow the progress of the batch. The '**Recipe**' readout should now display 'coolmix'.

Figure 7-10   The operator message

3   Some time after the batch phase has reached 'Soak' an orange/black flashing question mark appears at the left-hand end of the alarm pane. This is the 'message queued' flag and it appears because the batch has gone into a PV high absolute alarm and high deviation alarm simultaneously.

4   Press the alarm pane to pop up the **Alarms** menu, then press **MESSAGES**. The flashing '**WARNING!**' dialog that you configured in Table 7-2 appears, telling you that the batch has become 'unstable'. You can either abort the batch or ignore the warning, at your peril! (See Figure 7-10.)

5   Select one of these alternatives. The message is cleared from the queue, but will reappear if the batch again enters an 'unstable' condition from a 'stable' condition. (This may happen at the start of the 'Cool Down' phase, if you chose to ignore the warning.)

> *Note*   *By default, popup dialogs disappear from instrument's screen after the relatively short time of 10 seconds. If you want to lengthen this period, to give you more time to think, press the **Menu** key, then **SYSTEM**, **SETUP**, **PANEL**. In the **Panel Setup** screen, edit the **Pop-up** field to the required value. Editing it to '0' seconds disables the timeout and makes popups stay visible indefinitely or until you clear them. Hit **SAVE** to store your new settings.*

Intentionally left blank

CHAPTER 8 SEQUENCES

This chapter guides you through the creation of a simple *sequence* for controlling the simulated 'vat' via the PID loop configured in Chapter 2. You then go on to adapt the sequence to drive the batch phases you configured in Chapter 6, instead of using the setpoint program, which gives you a lot more flexibility. Finally, you enhance the user screen created in Chapter 7 to give more integrated control over the batch.

By combining and building on what you have configured in previous chapters of this tutorial, you finish up with a sequence-driven batch control application that includes an interactive user screen, batch reporting, recipes, and panel messaging.

The Sequence. A sequence is the control of specific states, *Steps*, in an ordered manner using *Transitions*. It is always run in conjunction with a LIN database,and is particularly applicable to processes that can adopt one or more distinct 'states', e.g. 'Heating Up', 'Waiting', 'Cooling Down', etc. Each *Step* is associated with one or more *Actions* that are configured using additional SFC's, Structured Text or Ladder diagrams. Actions are user-specified operations on LIN database fields that occur when the Step is active. If the *Action* is successfully completed, the *Transition* allows the activity to pass to the next *Step*, repeating or continuing until the sequence is complete.

The sequence determines:

- The initial state(s) adopted by the process at startup
- The conditions triggering state-changes (events)
- The new state(s) adopted when changes are triggered
- The way the LIN database controls the process in each of the states.

In LINtools a sequence is represented and configured graphically as a *Sequential Function Chart* (SFC). An example of an SFC, the one you are about to build, is shown in Figure 8-2. The *Steps* in the SFC represent states, and the *wires* between them determine which state(s) can be adopted from a given state, and the events that will trigger these state-changes.

> *Note* *This chapter explores only a very few of the Sequence Editor capabilities. Consult the* LINtools *Engineering Studio Help file, Part no. RM 263 001 U055, for full details of all the features.*

The main sections in this chapter are:

- Modifying the PID loop strategy for sequence
- Creating a sequence
- Running the sequence application
- Adding batch and recipe to the sequence strategy
- Running the sequence/batch application
- Adding Hold, Abort, and Skip to the application
- Running the enhanced sequence/batch application

RATE_LMT block
[Timing Category]
Limits the rate of change of an output as it tries to attain a target input value. Ramp-up and ramp-down rates are individually specified.

TIMER block
[Timing Category]
Resettable incremental up-timer, with digital flag set when timing is complete.

SFC_CON block
[Batch Category]
SFC control block lets sequences be selected, loaded, initialised, held, and run by the control strategy.

SFC_MON block
[Batch Category]
SFC monitor block monitors up to eight steps of a running sequence. Reports step name, activity status, and active time.

SFC_DISP block
[Batch Category]
SFC display block displays active step and action name(s) in the running sequence.

Figure 8-1   The simple PID loop strategy adapted for sequence

8.1 MODIFYING THE PID LOOP STRATEGY FOR SEQUENCE

Figure 8-1 shows the (simulation) PID loop strategy you created in Chapter 2, but with five new blocks, SFC_CON, SFC_MON, SFC_DISP, RATE_LMT, and TIMER, added to give the strategy sequence and timed ramping capability. The AI_UIO and AO_UIO blocks have been deleted as they are not needed in this simulation strategy.

8.1.1 Configuring the strategy

1   Open the simulated PID loop strategy you built in Chapter 2 in LINtools, by double-clicking its icon in Explorer view. Delete or ignore the AI_UIO and AO_UIO blocks, then add the five new blocks as shown in Figure 8-1.

2   Wire up the strategy according to Table 8-1, which as usual lists all the necessary connections for completeness, in case you are starting from scratch.

> *Note   In the table, the single wire below the dashed line is the only new one. So you need only configure that if you are starting from the existing PID loop strategy created in Chapter 2.*

Source*	Destination*	Function
(PID) pid.OP	(SIM) sim.PV	PID control output OP into the SIM block's PV input to simulate output to the plant
(SIM) sim.OP	(PID) pid.PV	SIM block's lagged output OP into the PID block's PV input to simulate the plant's PV
(PID) pid.OP	(DR_ANCHP) OP_rcrd.CurrVal	PID control output OP into the OP_rcrd (DR_ANCHP) block to enable data recording onto Visual Supervisor's local flash storage
(PID) pid.PV	(DR_ANCHP) PV_rcrd.CurrVal	PID process variable PV into the PV_rcrd block to enable data recording
(PID) pid.SP	(DR_ANCHP) SP_rcrd.CurrVal	PID setpoint SP into the SP_rcrd block to enable data recording
(RATE_LMT) rate.OP	(PID) pid.RemoteSP	RATE_LMT block's ramped output into the PID block's remote setpoint to ramp the setpoint up or down

> *Note   *(Block type) blockname.field.subfield format, referring to Figure 8-1. New wiring is listed below the dashed line.*

Table 8-1   Sequence strategy wiring data

> *Note   The Configuration (Header) block and the AREA, GROUP, SFC_CON, SFC_MON, SFC_DISP, and TIMER blocks don't need to be wired up. They are associated with other blocks via field values (see next).*

8.1.2 Editing the block fields

Table 8-2 lists the fields that should, as a minimum, be edited from their default values. For completeness, all such fields are shown in the table — including those edited in the original PID loop strategy.

> *Note   In the table, the fields below the dashed line are the new ones, so you need only edit these if you are starting from the existing PID loop strategy created in Chapter 2.*

You must also edit the **Name** fields in all blocks, see Figure 8-1.

When you've carried out all necessary field edits, save the modified PID loop strategy under a new name, e.g. '**seq.dbf**'.

Field *Value	Function
(PID) pid.TI = **5.00**	Sets the PID integral time constant to 5 seconds, to improve control in this simulated strategy
(PID) pid.HAA = **80.00**	Sets the pid block's PV output 'high absolute' alarm at 80%
(PID) pid.LAA = **20.00**	Sets the pid block's PV output 'low absolute' alarm at 20%
(AREA) area.Id = **1**	Allocates area number (only Area '1' currently implemented). No OVERVIEW appears in the instrument unless an area number has been allocated
(AREA) area.Group1 = **pidloop**	Assigns the 'pidloop' GROUP block to this area. This specifies that 'pidloop' will appear in the OVERVIEW and enables data recording for this group
(GROUP) pidloop.Update = **10**	Specifies the 'pidloop' group's recording sample rate (seconds). The default rate of zero disables recording
(GROUP) pidloop.Disp1 = **pid**	Specifies the PID block 'pid' as the source of the data that will be displayed in the faceplate and trends for the 'pidloop' group
(GROUP) pidloop.Chan1 = **PV_rcrd**	Assigns the recorder channel block 'PV_rcrd' to channel 1 of the 'pidloop' recording group
(GROUP) pidloop.Chan2 = **SP_rcrd**	Assigns the recorder channel block 'SP_rcrd' to channel 2 of the 'pidloop' recording group
(GROUP) pidloop.Chan3 = **OP_rcrd**	Assigns the recorder channel block 'OP_rcrd' to channel 3 of the 'pidloop' recording group
(SIM) sim.Lag1 = **25.00**	Sets the SIM block's 1st filter time constant to 25 seconds, to simulate lag in the plant's process variable
(AREA) area.DispMode.H_Trend = **TRUE**	Enables the horizontal trend display, needed for this application
(AREA) area.DispMode.V_Trend = **FALSE**	Disables the vertical trend display, which is not required in this application
(PID) pid.SelMode.SelRem = **TRUE**	Selects remote operating mode for the PID block, to allow the remote setpoint generated by the RATE_LMT block to be used
(PID) pid.SelMode.EnaRem = **TRUE**	Enables remote operating mode for the PID block
(PID) pid.Alarms.HighAbs = **1**	Enables the 'high absolute' alarm on the pid block's PV output
(PID) pid.Alarms.LowAbs = **1**	Enables the 'low absolute' alarm on the pid block's PV output
(PID) pid.Alarms.HighDev = **1**	Enables the 'high deviation' alarm on the pid block's PV output
(PID) pid.Alarms.LowDev = **1**	Enables the 'low deviation' alarm on the pid block's PV output
(SFC_CON) sfc_con.init = **False**	Holds the loaded sequence waiting for initialisation
(SFC_CON) sfc_con.Run = **True**	The loaded sequence is waiting for init = True
(SFC_CON) sfc_con.load = **TRUE**	Loads the sequence when the LIN database starts running
(SFC_CON) sfc_con.FileName = **seq**	Specifies the sequence file 'seq.sdb' as the sequence to be controlled by this SFC_CON block
(SFC_CON) sfc_con.DispBlk = **sfc_disp**	Assigns the SFC display block 'sfc_disp' to the SFC_CON block
(SFC_MON) sfc_mon.SfcBlock = **sfc_con**	Assigns the SFC control block 'sfc_con' to the SFC_MON block
(SFC_MON) sfc_mon.Step_A = **Init**	Specifies 'Init' as the name of a step in the sequence to be monitored by the SFC_MON block
(SFC_MON) sfc_mon.Step_B = **Wait**	Specifies 'Wait' as the name of a step in the sequence to be monitored by the SFC_MON block
(SFC_MON) sfc_mon.Step_C = **Heat_up**	Specifies 'Heat_up' as the name of a step in the sequence to be monitored by the SFC_MON block
(SFC_MON) sfc_mon.Step_D = **Soak**	Specifies 'Soak' as the name of a step in the sequence to be monitored by the SFC_MON block
(SFC_MON) sfc_mon.Step_E = **Cooldown**	Specifies 'Cooldown' as the name of a step in the sequence to be monitored by the SFC_MON block
(SFC_MON) sfc_mon.Step_F = **Stand**	Specifies 'Stand' as the name of a step in the sequence to be monitored by the SFC_MON block
(SFC_DISP) sfc_disp.SfcBlock = **sfc_con**	Links the SFC control block 'sfc_con' to the SFC_DISP block
(SFC_DISP) sfc_disp.Running = **TRUE**	Enables the SFC_DISP block update routine

Note *(Block type) blockname.field.subfield *format, referring to Figure8-1*

Table 8-2   Values for non-default block fields in the sequence strategy

8.2 CREATING A SEQUENCE

Figure 8-2 shows the sequence (SFC) you are about to build. You can see that each stage or *state* in the vat-heating process is represented by a sequence *step*, one of the labelled boxes in the SFC. Apart from the first (top-most) step, an initialisation step where nothing much happens, each step corresponds to one of the batch phases you defined in Chapter 6, *Batches*.

Initial step. Indicated by a double-outlined border. An initial step becomes active as soon as the sequence starts to run.

Wire. A directed link between steps. Represents the path the sequence may take through the steps.

Wire symbol. Click this to start a wire. Click <Escape> to abort the wire operation.

Step. Represents a discrete stage in the process, i.e. one of its possible 'states'. Associated with each step are 'Actions'.

Actions are user-specified operations on LIN database fields that happen while the step is active. (Actions are not shown on the SFC)

Transition. Represents a logical TRUE/FALSE condition or 'event'. When TRUE, a change of state occurs, i.e. activity passes to a new step.

Step name. You usually name steps to reflect the states they represent, as in this case.

In this sequence, activity returns to the initial step when the last transition becomes TRUE

Figure 8-2   The sequential function chart (SFC) — initial version

To create this sequence, do the following:

1   In LINtools, right-click the SFC_CON block FileName field displayed in the Object Properties Pane, reading **seq**, to show a selection of commands.

2   Select the Open Sequence file (.SDB) to launch the LIN Sequence Editor.

Alternatively,

i   In your Computer, access the Explorer view of the working folder containing the 'seq.dbf' file. Right-click in the list pane and in the popup menu select **New**, **Eurotherm LIN Sequence**. A **.sdb** file appears with a default filename.

ii   Edit the filename to 'seq.sdb'. Double-click it to launch the LINtools **Specific Sequence Editor**. A blank editor window appears. Notice that the seq.dbf file automatically associates itself with the new sequence because it shares the same root filename and folder.

*Note   You can get specific **help** for an active dialog by pressing the <F1> key. For most menu items press <F1> when the cursor is over the item. To get help for most toolbuttons, locate the cursor over the tool, hold down the left mouse button and press <F1>. Release the mouse button. This works even for 'greyed out' toolbuttons.*

8.2.1 Placing the sequence steps

Start your sequence by placing the six steps, defining the process states:

1   To place the first step, click the **Step** tool ⊡ to 'load' the cursor with a step symbol. Click the loaded cursor near the top of the editor window to paste down a step. The first step to be pasted is always an 'initial' step with a double-outlined border, and is labelled 'STEP1' by default. You'll rename the steps later.

2   Place five more steps below the first, spaced out a bit more generously than those shown in Figure 8-2. An invisible grid helps you align the steps.

To move a step, simply drag it with the mouse but avoid clicking on its wire symbol (see Figure 8-2). To delete a step, click it to highlight it then press **<Delete>**.

> *Tip!*   *To place a series of steps, hold down the <Ctrl> key as you paste a step down. This leaves the cursor still loaded and ready to paste more steps. <Esc> 'unloads' the cursor.*

3   If required, adjust the positions of the steps so that they align nicely in a vertical column. Leave a gap of just over one step-height between each of your steps, to make sufficient room for the wire and transition symbol.

> *Note*   *The invisible 'snap-to' grid allows the gap between steps to be either just under or just over one step-height. Use the larger spacing to avoid a mess later on.*

8.2.2 Wiring up the steps

Now connect up the steps with 'wires' to specify the order in which the states are adopted:

1   Locate the cursor over the first (top-most) step's **wire symbol** ↵ at the lower right corner of the step box. The regular arrow cursor changes to a special **wiring cursor**.

2   Click the wire symbol then locate the cursor over any part of the destination step, 'STEP2' in this case. A **target symbol** 🐾 appears below the cursor, signifying that this is a legitimate wire destination.

3   Click over the destination step to complete the wire connection. A brown (unconfigured) transition bar ⊞ appears on the new wire. Drag the highlighted transition bar up the connecting wire to centre it if you wish. Figure 8-3 summarises these wiring operations.

1) Locate the cursor over the source step's **wire symbol**. The cursor changes to the special **wiring cursor**.

2) Click the wire symbol then locate the cursor anywhere over the destination step. The cursor changes to a **target cursor**.

3) Click on the destination step. A connecting wire appears with a highlighted **transition bar** on it. Drag the bar to adjust its position if required.

Figure 8-3   Wiring up 'STEP1' to 'STEP2'

> *Note*   *If your wire turns into a tangle it's probably because the steps are too close. To cure this, drag the lower step downwards one grid space at a time until the tangle disappears. Then centre the transition bar on the wire if you wish.* **It pays to keep your sequence layout as neat as possible** *so that you can see what's going on as the sequence gets more complicated later on!*

4   Wire up the rest of the steps in the same way. Don't forget to wire the last step, 'STEP6', all the back to 'STEP1'. We want the sequence to return to the initial ('IDLE') state when it has run its course.

5   Tidy up the layout so that it looks like the one shown in Figure 8-2. To move a wire just drag it perpendicular to its length. To delete a wire with its transition, highlight the transition and press **<Delete>**. If you move a transition, the associated wire moves with it. Zoom the window by clicking ⊕ ⊖ if you need a closer view.

> *Note*   *A short 'stub' of wire always remains attached to the step symbol. This may be dragged to different positions but not removed. Wiring can only be drawn vertically and horizontally.*

8.2.3 Naming the steps

When you placed the steps they adopted default names. To rename them:

1 Double-click the first step (avoiding its wire symbol). A **step pane** appears at the bottom of the editor window, entitled with the current (default) step name.

2 Click the white **Name** field and type in the required step name, 'IDLE' in this case. Hit **<Return>** to enter the new name, which appears on the step. Note that the **Initial** field's tickbox is ticked TRUE (by default), making this step an 'initial' step. Leave this setting as is.

> *Note* *The column headings in the step pane,* Qualifier, Duration, Action, Type, *refer to the* actions *that you will configure after naming the steps.*

3 Click the next step to see its step pane, and overwrite the default name with 'Wait'. Leave the **Initial** field at FALSE. Rename the remaining steps according to Figure 8-2.

8.2.4 Configuring Actions

As each step (state) becomes active, you want the sequence to write values to certain LIN block fields in the strategy to control the 'vat' in ways appropriate to that particular state, e.g. while the 'Heat_up' step is active the PID block's remote setpoint must be ramped up to the required temperature at the required rate. While the 'Soak' step is active the vat's temperature must be held at a constant value for the required soak time, and so on throughout the sequence.

You achieve these requirements by writing *actions* in a simple language called *Structured Text* (ST). They're called actions because they act on the LIN database. When you have created and named the necessary actions you then associate them with the relevant steps. Note that a given action can be associated with more than one step if it's needed again, as is the case in this tutorial.

Creating the 'zerotime' action. Start by writing an action that zeroes and disables the TIMER block.

1 In the sequence editor window, click the **Make Action** tool 🎬 to pop up a **Make Action** dialog. Type in a **Name** for the action — 'zerotime'. (Action names can have up to eight characters.) Leave the **Type** as 'Structured Text' and hit **OK** to close the popup. A blank action window appears for entering the structured text, and the **Contents pane** on the left lists the name of the new text action.

> *Note* *The contents pane lists all the actions in the sequence in order of creation. The sequence itself is also actually an action — a 'chart' (SFC) action.*
> *If you cannot see the contents pane for any reason, click **Contents** in the **View** menu.*

2 Type the following text, exactly as shown, into the action window:

```
timer.Reset:=1;    (*zero and inhibit timer*)
```

This has the effect of assigning the value '1' (TRUE) to the *Reset* field of the block called 'timer' — i.e. the TIMER block. This resets the timer function and inhibits any timing action.

> *Note* *You must use ':=' in an assignment statement (not just '='), and end the assignment with ';'. Optional comments can be added enclosed by brackets and asterisks, and are ignored when the sequence is run.*

3 In the action window, right-click to pop up a context menu and select **Compile zerotime (Text)**. This causes the sequence editor to run a compilation check on the 'zerotime' action's ST to see if contains any detectable errors. You should see a '**zerotime (Text) - No Errors**' message in the pane below the action window — assuming you've made no mistakes. Check carefully if any errors are flagged up.

Using the ST prompter — the 'wait' action. Now create another action, using the editor's built-in *ST prompter* to help you. This action will set the PID block's remote setpoint to zero (°C), and set the timer period to 60 seconds — the 'wait' time:

4 Click the **Make Action** tool again and create a text action called 'wait'. Right-click in the blank action window to pop up a context menu, then click **Variable...**. A pick-list of LIN block fields appears, from the associated the LIN database '**seq.dbf**' (see Figure 8-4).

5 Double-click the **rate** item. The text '**rate.**' appears in the action window, and the pick-list now shows all the 'rate' block's fields. Double-click **PV** to add it to the action as '**rate.PV**'.

i. Right-click in the blank action window to pop up a context menu, then click **Variable...**.

A pick-list of LIN block appears, from the '**seq.dbf**' database.

Select the '+' symbol beside the **rate** *block to reveal the fields in this block.*
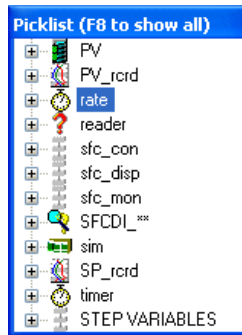
ii. Select *PV* to add it to the action as '**rate.PV**'.
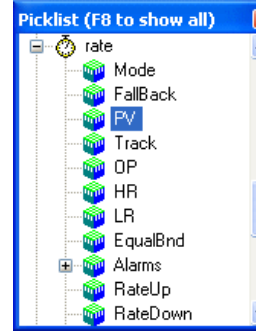
iii. Right-click and highlight **Operator** in the context menu. Then click **:= (assign)** to insert the assignment symbol.

Type in '0;' to finish the statement as 'rate.PV:=0;'.

Add comments if you wish.
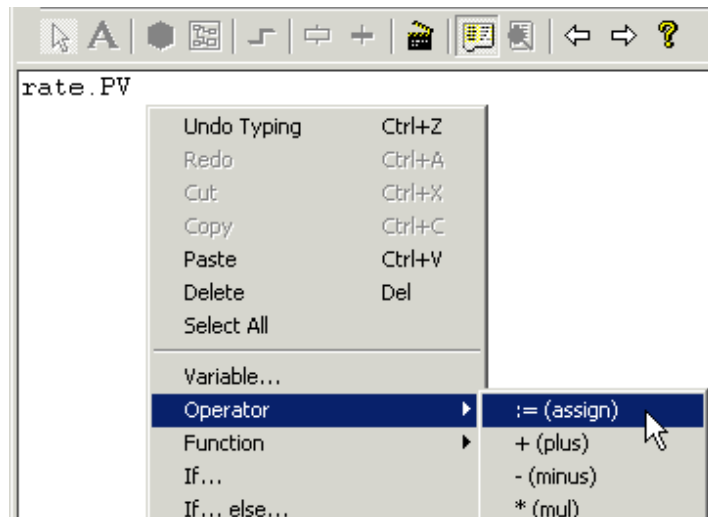
Figure 8-4   Writing the 'wait' action with the help of the ST prompter

6 Now right-click again and highlight **Operator**, then click **:= (assign)**. This inserts the assignment operator. Finish the statement by typing in '**0;**' by hand. Add the comment '(***set target SP to zero***)' — which is what this assignment does.

7 Add the remaining three lines of ST to the action window, according to Figure 8-5. Use the ST prompter to save time and to ensure that you do not mistype any fields.
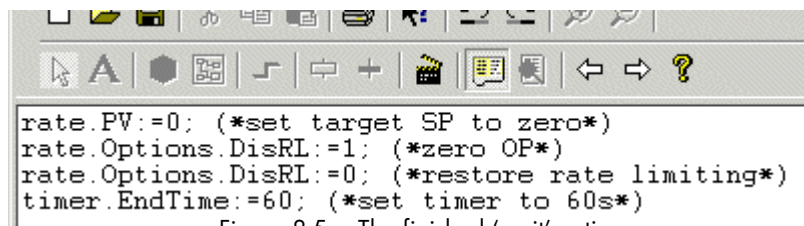
```
rate.PV:=0; (*set target SP to zero*)
rate.Options.DisRL:=1; (*zero OP*)
rate.Options.DisRL:=0; (*restore rate limiting*)
timer.EndTime:=60; (*set timer to 60s*)
```
Figure 8-5   The finished 'wait' action

Note   The LIN Blocks Reference Manual, *Part no. HA 082 375 U003, explains fully how these fields work, as do the* LINtools Engineering Studio *individual field help popups (click a field and press<F1>).*

8 Compile the action to check for errors.

Creating the 'starttim' action.   Create one more action before you go on to associate actions with steps. Enter the 'starttim' action as:

```
timer.Reset:=0;     (*start timer*)
```

This action makes the 'timer' block's *Reset* field FALSE, which allows the *CurrTime* field to start incrementing in seconds until it reaches the *EndTime* value. When it does, the block's *Out* field goes from '0' (FALSE) to '1' (TRUE).

8.2.5 Associating actions with steps

Now you have created some actions, try associating them with the first two steps in the sequence — 'IDLE' and 'Wait'. To do this:

1   Double-click the '**ROOT (Chart)**' item in the contents pane to bring back the SFC to the editor window. The main SFC appears by default as 'page 1'.

2   (Double-)click the 'IDLE' step to display its step pane at the foot of the window, in the *Object Properties Pane*. Your editor window should resemble Figure 8-6.

3   (Double-)click the cell in the **Action** column ('ROOT') to pull down a menu of all actions currently configured. Select the **zerotime** item to associate it with this step.

Figure 8-6   Associating an action with the 'IDLE' step

4   Now click the cell in the **Qualifier** column (labelled 'N-Normal') and select **P - Initial**. The corresponding **Type** column cell now reads **TEXT**, because 'zerotime' is a Structured Text Action. Selecting a 'P' type qualifier means that this action will be executed as soon as the step activates, and will be executed only once. After all, we need only zero and disable the timer clock *once* in this step.

> *Note*   *The default qualifier, **N** - (Normal), runs an action repeatedly as long as the associated step is active, i.e. once every database iteration. If you want to see what the other qualifiers do, access the LINtools help file by pressing* **<F1>** *and look up 'Action qualifiers' in the Index.*

Figure 8-7   Associating an action with the 'IDLE' step

5   That's the only action associated with the 'IDLE' step. Now click the 'Wait' step to display its step pane. Append an action row to the table and select the **wait** action for association. Assign the **P - Initial** qualifier to this text action.

Other items that can appear in the popup allow you to *Make New Action*, *Delete* the current row, *Move Up/Down* the row in the table, or *Goto* an action, i.e. display it in the editor window.)

6   Add the **starttim** action to the table as the second item, and qualify it as a '**P**' action. Finally, append the **zerotime** action to the 'Wait' step, qualified as an '**F - Final**' action. '**F**' actions execute once only, immediately after the step de-activates. We don't want to zero and disable the timer until the 'wait' period is over.

> *Note*   *The order of the actions in the step pane can be important, as well as their individual qualifiers. This is because they execute in this order when the step activates, with the top action going first.*

8.2.6 Creating the remaining actions

You can now go on to configure and compile the rest of the actions needed for this sequence, using the methods described above. Refer to Table 8-3 for your data, which lists all the actions — including the three you've already configured. Don't forget to use the ST prompter to speed up ST writing and avoid spelling mistakes.

Note   You can 'copy-&-paste' text freely between similar actions if helpful. Double-click a contents pane item to display it in the main editor window.

Action	Structured Text and (*Comment)*	Function
zerotime	timer.Reset:=1 (*zero & inhibit timer*)	Reset bit latches on
wait	rate.PV:=0; (*set target SP to zero*) rate.Options.DisRL:=1; (*zero OP*) rate.Options.DisRL:=0; (*restore rate limiting*) timer.EndTime:=60; (*set timer to 60s*)	Copies PV to OP
starttim	timer.Reset:=0; (*start timer*)	Unlatches Reset bit
heatup	rate.RateUp:=4; (*set uprate to 4/s*) rate.PV:=80; (*set target to 80 & start ramp*)	OP ramps to PV at RateUp/s
soak	timer.EndTime:=120; (*set timer to 120s*)	
cooldown	rate.RateDown:=0.5; (*set downrate to 0.5/s*) rate.PV:=20; (*set target to 20*)	
stand	timer.EndTime:=60; (*set timer to 60s*)	
stopsequ	sfc_con.Run:=0; (*stop & initialise sequence*)	Init bit works when Run = 0

Table 8-3   Text actions used in the sequence (first version)

8.2.7 Associating the remaining actions to steps

Associate the rest of the actions to appropriate steps, according to Table 8-4.

Step	Qualifier	Associated actions
IDLE	P - Initial	zerotime
Wait	P - Initial P - Initial F - Final	wait starttim zerotime
Heat_up	P - Initial	heatup
Soak	P - Initial P - Initial F - Final	soak starttim zerotime
Cooldown	P - Initial	cooldown
Stand	P - Initial P - Initial F - Final	stand starttim stopsequ

Table 8-4   Steps and associated actions (first version)

8.2.8 Writing transitions

So, you have now laid out and wired up the sequence steps and associated them with actions, executed according to action qualifiers. To complete the sequence you must tell each step what must happen for it to deactivate and pass its activity on to the next step. That is, you must write the TRUE/FALSE transition expressions.

> *Tip!   Remember, transitions are represented on the SFC by the short horizontal bars crossing each wire between steps.*

Start with the transition from 'IDLE' to 'Wait':

1   (Double-)click on the brown-coloured transition bar to pop up a transition pane at the foot of the window, entitled **Transition: IDLE to Wait**.

2   Enter the following ST expression in the transition pane, either by hand or with the help of the ST prompter:

> `sfc_con.Run = 1  (*sequence is running so move to Wait step*)`

> *Note   A transition expression is **not** an assignment — it is an equality test. Use the ordinary equals sign '=' and do **not** terminate a transition expression with a semicolon ';'.*

When the SFC_CON block's Run bit is set to '1' (TRUE), which you will do when you want the sequence to start running, this transition expression becomes TRUE. So the 'IDLE' step deactivates and the 'Wait' step activates, automatically starting the 60-second wait time.

3   Check the validity of the expression by right-clicking in the transition pane and selecting **Compile** from the popup. A **Compiling...** sub-pane appears which should report **'No Errors'**.

> *Note   Transition bars are* brown *when unconfigured, but turn* black *when configured. You may have to click on the transition bar to see this colour-change.*

4   Now configure and error-check the rest of the transitions according to Table 8-5. Speed up configuration by using copy-&-paste for similar transitions.

Action	Expression Text and (*Comment)*	Function
IDLE to Wait	sfc_con.Run = 1 (*sequence is running so move to Wait step*)	You will set Run manually
Wait to Heat_up	timer.Out = 1 (*timed period over*)	Out sets if CurrTime = EndTime
Heat_up to Soak	rate.OP >= 80 (*output has reached target*)	OP ramps up to PV
Soak to Cooldown	timer.Out = 1 (*timed period over*)	
Cooldown to Stand	rate.OP <= 20 (*output has reached target*)	OP ramps down to PV
Stand to IDLE	timer.Out = 1 (*timed period over*)	

Table 8-5   Transition expressions used in the sequence (first version)

8.2.9 Saving the completed sequence

To save your completed sequence under its current filename, click the **Save** toolbutton 💾. The sequence is automatically compiled and checked for errors.

Three files are produced when you save a 'specific' sequence (like this one), '**seq.sdb**', '**seq.sdt**', and '**seq.sgx**'. The .sdb file contains the runtime data and structured text, and is the only one you need to download to the Visual Supervisor. The .sdt (textual comments) and .sgx (SFC layouts) files should be kept in case you want to modify the sequence, which you will later on in this tutorial

8.3 RUNNING THE SEQUENCE APPLICATION

To run the (simulated) sequence application, do the following:

1   Download the '**seq.dbf**' strategy file and the '**seq.sdb**' sequence file to the Visual Supervisor's E: drive.

2   In the Visual Supervisor, load and run the simulation sequence application. The 'pidloop' group display appears, containing the single 'pid' faceplate representing the PID control block.

3   Press the **Menu** key, then **SYSTEM**, **APPLN**, **FB MGR**, to access the **FB Manager** screen. Press the **sfc_con** cell to see the **SFC_CON** block's fields. Notice that the **Load** field is TRUE but **Run** is FALSE. This is why the sequence was loaded when you started the application but is not yet running.

4   Get the sequence running by pressing the yellow **Run** field and editing it to TRUE. Then navigate to the **sfc_mon** block's field page. You can do this either by pressing the **Right** (or **Left**) key until you reach the page, or by pressing **Up** and selecting the block from the **FB Manager** screen.

5   Look at the **SFC_MON** block's fields. Each step name in the sequence appears in the appropriate **Step_A**, **Step_B**, ... field, with its activity state (TRUE/FALSE) and running time (seconds) displayed in the corresponding **A_X ...** and **A_T ...** fields respectively. Follow the progress of the running sequence by observing these fields.

> *Note*   *The 'IDLE' step is active for only a fraction of a second once you start the sequence running. This is because as soon as you set* sfc_con.Run *= TRUE the 'IDLE to Wait' transition operates (Table 8-5).*

6   Take a quick look at the **SFC_DISP** block page while the sequence is still running. Notice that the currently-active step-name is displayed in either the **ActStepA** or **ActStepB** field near the top of the page. Also, the **ROOT** action (the SFC itself) always appears in the **AActionA** (Active Action) field so long as the sequence runs. When the sequence stops these fields clear.

7   After a few minutes you will see the last step, **Step_F = Stand**, deactivate, and the sequence appears to stop running. Check this by returning to the SFC_CON block page and noting that **Run** is now FALSE again.

> *Note*   *The sequence switches itself off when the last step '**Stand**' deactivates, thanks to the '**stopsequ**' action that executes as an '**F - Final**' action. The SFC_CON block's **Run** field is reset to '0' (FALSE) by this action (see Table 8-3).*

8   Try running the sequence again by editing **sfc_con.Run** to TRUE. This time press **Menu**, **OVERVIEW** to access the overview screen. Press the **Down** key to scroll through the various trend displays and confirm that the 'vat' is being controlled in the intended manner.

9   You might want to take a look at how some of the other LIN blocks work while the sequence is running, especially the **TIMER** and **RATE_LMT** blocks.

Now that you have a working sequence, you can confidently modify it to control the batch/recipe/user screen application you developed in Chapter 7.

8.4 ADDING BATCH & RECIPE TO THE SEQUENCE STRATEGY

To enable the sequence to drive the batch engine you will add batch (and recipe) functionality to the existing sequence strategy. The quickest way to do this is to copy and paste the required blocks from an earlier strategy into the new one. Figure 8-7 shows the result you are aiming for.
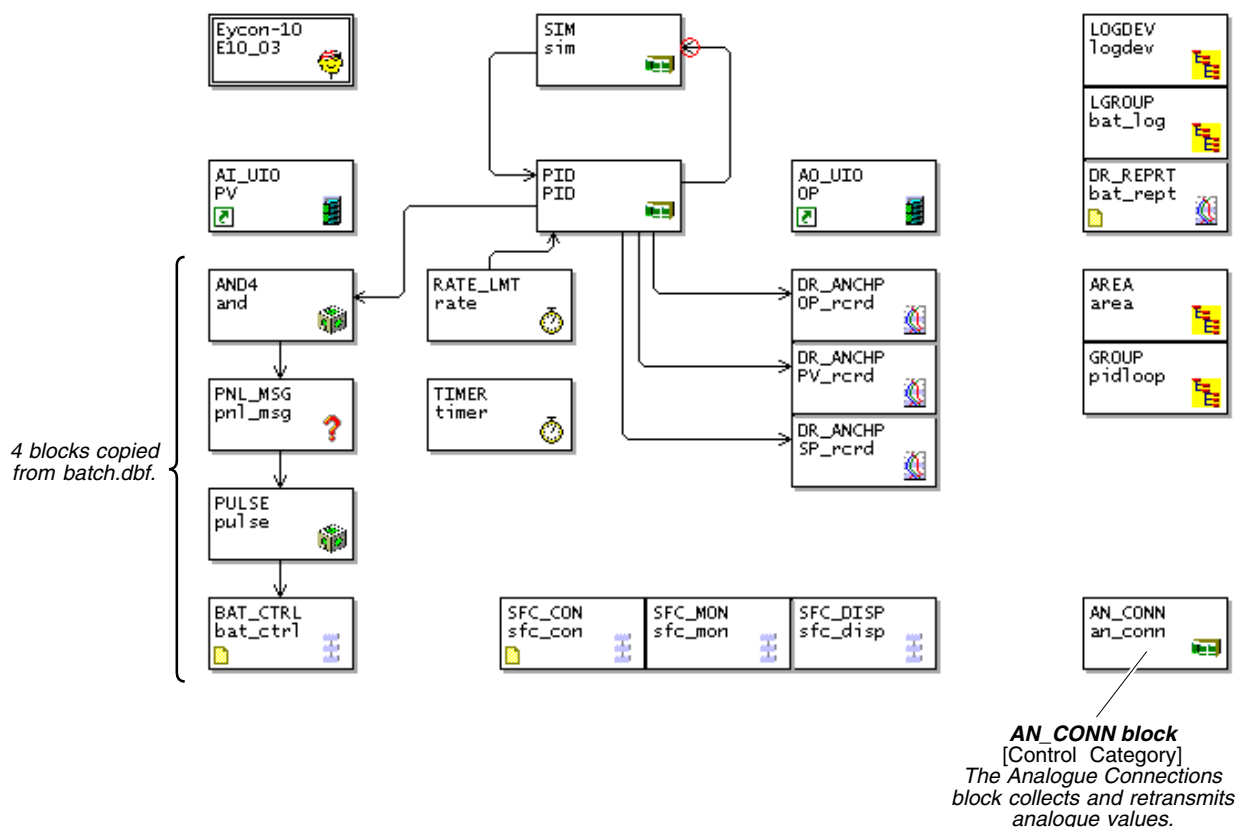
Figure 8-7   Adding batch and recipe to the sequence strategy

1   Open up the '**seq.dbf**' file you created early in this chapter (shown in Figure 8-1). Reduce the size of the LINtools window so that it occupies about one half of the screen.

2   Also open up the '**batch.dbf**' file you developed in Chapter 7 (shown in Figure 7-1). Reduce its window so that you can place it alongside 'seq.dbf' and see both at once — at least partially.

3   In the 'batch.dbf' window, multiple-select the **LOGDEV**, **LGROUP**, **DR_REPRT**, **RCP_SET**, and **RCP_LINE** set of five blocks. Copy and paste them onto the 'seq.dbf' window in the positions shown in Figure 8-7. You can do this either by using **<Ctrl-C>** (copy) and **<Ctrl-V>** (paste), or simply by dragging the selection from one window to the other.

4   Now select the **AND4**, **PNL_MSG**, **PULSE**, and **BAT_CTRL** blocks in 'batch.dbf' and drag the four blocks across to the 'seq.dbf' window where indicated.

5   Finally, place a new block, **AN_CONN** (Control category), in the lower right corner of the layout. This block will be used to store a few values for access by the sequence.

6   Complete the layout by adding the one missing wire (bus), from the PID block's **pid.Alarms.HighAbs** output to the AND4 block's **and.In_1** input, and also from **pid.Alarms.HighDev** to **and.In_2**. (You recall that this connection triggers the 'Batch has become unstable' operator message.)

7   All the blocks already have suitable field values copied over from the earlier strategy, or defaults, so no field edits need be made. (If you need to know field values, please refer to the original strategy in Chapter 7.) Save the edited database as 'seq.dbf'.

8.4.1 Modifying the sequence to run a batch

We are going to modify the sequence in two ways:

■ Enable it to drive the batch engine phases

■ Implement different temperatures and times for each recipe in the 'Mixers' set.

BATCH PHASES

For the sequence to be able to 'drive the batch engine' it must be able to tell the batch what phase it has reached, i.e. the currently-active step. To do this, all that is needed is an action included in each step that writes the number of the current phase to the BAT_CTRL block's **CurPhase** field.

> *Note* *This corresponds to what the setpoint program did in order to drive the batch phases in Chapter 6.* *There, the SPP_CTRL block's* CurrSeg *output was the source of the phase number.*

After the sequence has run its course, it must also be able to put the batch into the COMPLETE state. Otherwise the batch will continue in the RUNNING state indefinitely. This is easily done by adding an action to the 'Stand' step that writes a zero to the BAT_CTRL block's **CurPhase** field as the step deactivates, i.e. an 'F - Final' action. This is accepted by the batch as an indication of the COMPLETE state, which it immediately adopts.

RECIPE-DRIVEN TIMES AND TEMPERATURES

Now that a sequence is controlling the batch, we can take advantage of the flexibility offered by this option. It would be nice to have the various dwell times and temperatures tailored to suit each recipe in the set, rather than be fixed by a setpoint program. To do this we will edit the 'Mixers' recipe set, in the next section, to have different target temperatures and times during the 'Heat_up', 'Soak', 'Cooldown', and 'Stand' steps. When the batch runs, these recipe values will be downloaded to fields in the AN_CONN block that are read by the sequence.

Some of the sequence actions must be modified slightly to read these recipe values, rather than be assigned fixed values.

So, to modify the sequence in these ways, do the following:

1 Open up the sequence file, '**seq.sdb**', and create the six new text actions listed in Table 8-6. These will let the sequence drive the batch engine phases.

Action	Structured Text and (*Comment)*	Function
phase1	bat_ctrl.CurPhase:=1; (*this is phase 1*)	Executed in 'Wait' step
phase2	bat_ctrl.CurPhase:=2; (*this is phase 2*)	Executed in 'Heat_up' step
phase3	bat_ctrl.CurPhase:=3; (*this is phase 3*)	Executed in 'Soak' step
phase4	bat_ctrl.CurPhase:=4; (*this is phase 4*)	Executed in 'Cooldown' step
phase5	bat_ctrl.CurPhase:=5; (*this is phase 5*)	Executed in 'Stand' step
stopbat	bat_ctrl.CurPhase:=0; (*sets batch to COMPLETE*)	

Table 8-6   New text actions required by the modified sequence

2   Now modify the existing actions listed in Table 8-7. This will allow the sequence to work with recipe variables, rather than using fixed temperatures and times.

Action	Structured Text and (*Comment)*
heatup	rate.RateUp:=4; (*set uprate to 4/s*) rate.PV:=an_conn.PV1; (*set target to 'soaktmp' recipe value & start ramp*)
soak	timer.EndTime:=an_conn.PV3; (*set timer to 'soaktim' recipe value*)
cooldown	rate.RateDown:=0.5; (*set downrate to 0.5/s*) rate.PV:=an_conn.PV2; (*set target to 'standtmp' recipe value and start ramp*)
stand	timer.EndTime:=an_conn.PV4; (*set timer to 'standtim' recipe value*)

Table 8-7   Modifications to existing actions

3   Now associate the new actions with the corresponding steps, as shown in Table 8-8, which lists all the action associations in the sequence.

> *Note*   *It is important to get the action associations in the right order, especially those in the final 'Stand' step. Since it is the sequence that is to stop the batch, this must happen **before** the sequence stops itself!*

4   Save the edited sequence.

Step	Qualifier	Associated actions
IDLE	P - Initial	zerotime
Wait	P - Initial P - Initial P - Initial F - Final	phase1 wait starttim zerotime
Heat_up	P - Initial P - Initial	phase2 heatup
Soak	P - Initial P - Initial P - Initial F - Final	phase3 soak starttim zerotime
Cooldown	P - Initial P - Initial	phase4 cooldown
Stand	P - Initial P - Initial P - Initial F - Final F - Final	phase5 stand starttim stopbat stopsequ

Table 8-8   Steps and associated actions (modified sequence)

8.4.2 Updating the recipe file

As explained in the previous section, the 'Mixers' recipe set is to have different target temperatures and times during the 'Heat_up', 'Soak', 'Cooldown', and 'Stand' steps.

Also, it would be preferable to have the *batch* start the sequence rather than having to start it 'by hand', as you did earlier in this Chapter. This is easy to arrange: just have all the recipes download a 'TRUE' value to the SFC_CON block's **Run** field.

To achieve these aims, you can modify the recipe set in two ways:

■ Edit the **.uyr** file you created in Chapter 6 directly in a text editor (e.g. 'Notepad').

■ Edit the recipe set in the Visual Supervisor, via the Recipe Edit utility. This way may take longer but is less likely to introduce errors.

In either case you must save the edited recipe file as '**seq.uyr**' so that it will be automatically called by the batch file '**seq.uyb**'.

EDITING THE RECIPE FILE DIRECTLY

To do this:

1 Open up the recipe file in a text editor, it was probably called '**batch.uyr**' in Chapter 6.

2 Carefully edit the file to get the result shown in Figure 8-8 (the version number, date, time, and user name contained in line two don't matter) . Save the file as '**seq.uyr**'.

> *Note* *If you get a recipe error message when you run the application in the Visual Supervisor it will almost certainly be due to a typing error in the edited recipe file.*

```
UYR,1
21,15/03/02,10:10:21,ENGINEER
,30
,Setpoint:1,hotmix,warmmix,coolmix
hiabs,pid.HAA,100.0000,76.00000,51.00000
loabs,pid.LAA,98.00000,74.00000,49.00000
hidev,pid.HDA,10.00000,5.000000,2.000000
lodev,pid.LDA,10.00000,5.000000,2.000000
soaktmp,an_conn.PV1,99.00000,75.00000,50.00000
standtmp,an_conn.PV2,20.00000,15.00000,10.00000
soaktim,an_conn.PV3,120.00000,90.00000,60.00000
standtim,an_conn.PV4,90.00000,60.00000,30.00000
id,Mixers.Filepath,hotmix,warmmix,coolmix
seq,sfc_con.Run,TRUE,TRUE,TRUE
```

Figure 8-8   The edited recipe file 'seq.uyr' viewed in 'Notepad'

EDITING THE RECIPE SET IN THE VISUAL SUPERVISOR

This can only be done when you are running a recipe application in the Visual Supervisor, i.e. later on in this Chapter. Please refer to the '*Running the Sequence/Batch Application*' section below for details.

8.4.3 Modifying the batch and form files

The *batch file* you developed in Chapter 7 (shown in Figure 7-9) does not need changing, except for its filename. Edit this to '**seq.uyb**' to be uniform with the batch filename.

Similarly, the *report form file* you created in Chapter 6 (shown in Figure 6-9) needs only its filename changing, to '**seq.uyf**'.

8.4.4 Modifying the user screen

In Chapter 7, *User Screens*, you built a screen to represent the simulated 'vat'. This needs very little modification to run with the sequence application, just a change of linked variable because the setpoint program is no longer present.

Figure 8-9 shows the 'Vat' user screen modified for the sequence application. Notice that the '**Phase**' readout now has *two* text variable readout lines instead of one. In the setpoint program application the readout was linked to a single instrument variable (associated with the setpoint program) called **_SPP_SEG_S** which reported the current program segment (i.e. batch phase).
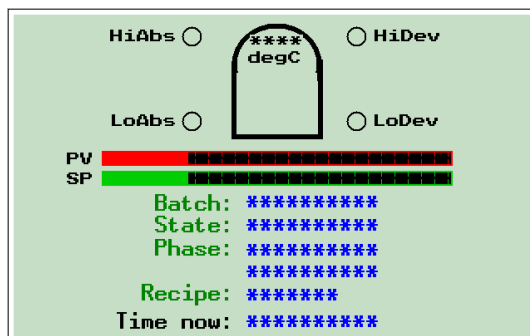
Figure 8-9   The modified user screen

In the absence of a setpoint program we can use two fields in the SFC_DISP block, **ActStepA** and **ActStepB**. Owing to the way this LIN function block works, the currently-active step (i.e. phase) is displayed in either of these two fields, the other field being blank. So to be sure of displaying the phase we include *both* fields as alternative readouts.

To modify the user screen, do the following:

1   Find the user screen file you created in Chapter 7, probably called '**batch.uxp**', and copy it to the directory holding your latest '**seq.dbf**' file. (Ignore the **.ofl** file.)

2   Double-click on the '**batch.uxp**' file in the sequence directory to open up its User Screen Editor window. This contains a '**batch.uxp**' sub-window with a blue '**Page1**' icon.

3   Right-click a blank area of the sub-window to pop up a context menu. Select **Properties...** to see the **Page Set Properties** dialog. Edit the **Name of Page Set** field to 'seq'.

4   Edit the **Location of DB file for Page Set** field using the **Browse...** button. Double-click '**seq.dbf**' in the **Open** dialog to make this the new source database for the user screen. Click **OK** to close the **Page Set Properties** dialog.

5   Now double-click on the blue '**Page1**' icon to open up the user screen, ready for editing. If the Target panel allows, increase the zoom to 200% for easier working.

6   To make room for the extra line, select all objects down to and including the
     '**Phase:** ********' line, and move them upwards.

> *Note*   *The easiest way to do this is to position the cursor just* outside *the top-left corner of the (green) user*
>   *screen, drag a box around the objects using the left mouse button, and then release the button. Hit the*
>   *'**Up**' arrow key repeatedly to increment the selection towards the top of the screen, to open up a space*
>   *below the 'Phase' readout.*

7   Move the '**Recipe**' and '**Time now**' readouts downwards in the same way.

8   Click the 'Phase' text variable object (the eight asterisks) to select it, then **<Ctrl-C>**, **<Ctrl-V>** to copy and paste
     it over itself. Nothing appears to have changed.

9   Now hit the '**Down**' arrow key to increment the still-selected text variable downwards to a suitable position (see
     Figure 8-9). This forms the second 'Phase' readout field.

10   Right-click the new field and select **Properties...** from the context menu to pop up the **Text Variable Item**
     Properties dialog. In the **Variable** page, select the **LIN Variable** radio button. Hit **Browse...** and click the '**+**'
     box next to '**sfc_disp**' in the **LIN Database Browser** window. Double-click the '**ActStepB**' item. This selects it
     as the source of the second 'Phase' readout. Hit **OK** to close the dialog.

11   Repeat the previous step for the first 'Phase' text variable readout, but link it to the **sfc_disp.ActStepA** field.

12   Save the finished user screen under its new name of '**seq.uxp**'. To do this, select **Save As...** in the **File** menu, edit
     the filename to 'seq.uxp' and click **Save**. A **Save OIFL?** dialog pops up. Click **Yes** to output a '**seq.ofl**' file. This
     is the file needed by the Visual Supervisor.

> *Note*   *You should not see any errors in the **Build Window** that appears. If you do get errors reported, they*
>   *are usually self-explanatory. Carefully check through the above steps to see if you have omitted*
>   *anything.*

8.5 RUNNING THE SEQUENCE/BATCH APPLICATION

To test run your new sequence/batch application:

1   Download the latest '**seq.dbf**' (database), '**seq.sdb**' (sequence), '**seq.uyr**' (recipe), '**seq.uyb**' (batch), '**seq.uyf**' (report form), and '**seq.ofl**' (user screen) files to the Visual Supervisor's E: drive.

2   In the Visual Supervisor's **Appl'n Manager** screen, load and run the sequence application, called '**SEQ**'. Instead of the usual Overview, the '**Mixer Vat**' user screen appears, containing the representation of the 'vat' and the various readouts that you configured.

8.5.1 Editing the recipe set

If you correctly updated the recipe file 'seq.uyf' earlier on in this tutorial to run with the sequence application, you won't have much to do now. In any case, access the recipe set as follows:

3   Press the **Menu** key, then **RECIPE**, **RECIPES** to see the **Load/Save Recipe** screen. Select the appropriate recipe file in the **File Name** field. If you updated it earlier and saved it as '**seq.uyr**', select the 'SEQ' recipe file. Otherwise select the old 'BATCH' recipe file for updating.

4   Press **LOAD** to pop up the **Recipe** window, then **EDIT** to open the recipe table for inspection/editing. Figure 8-10 shows the SEQ recipe table as it should look after editing.

Figure 8-10   The fully updated recipe set edit table

5   If you need to, edit the recipe set according to the data given in Table 8-9. If you want reminding how to do this, refer to the *Creating a recipe set* section in Chapter 5.

Variable (RCP)	Tag Reference	Hotmix	Warmmix	Coolmix
hiabs	pid.HAA	100	76	51
loabs	pid.LAA	98	74	49
hidev	pid.HDA	10	5	2
lodev	pid.LDA	10	5	2
soaktmp	an_conn.PV1	99	75	50
standtmp	an_conn.PV2	20	15	10
soaktim	an_conn.PV3	120	90	60
standtim	an_conn.PV4	90	60	30
id	Mixers.Filepath	hotmix	warmmix	coolmix
seq	sfc_con.Run	TRUE	TRUE	TRUE

Table 8-9   'SEQ' recipe set data for the sequence/batch application

6   When you have completed the edits, or are satisfied that your recipe set was already correct, press the **SAVE AS** button and save the file with the name 'SEQ'. This will overwrite any existing seq.uyr file.

8.5.2 Running a batch

Now that you have a correctly edited recipe set, start running a batch:

1   Press the grey pane at the top-left of the screen, labelled '**RESET**'. This pops up the **Batch** menu.

> *Note*   *When you were running an application containing both setpoint programming and batch (in Chapters 6 and 7), this pane accessed the **Programmer** menu, not the **Batch** menu. This is because a setpoint program takes priority over a batch for the use of this pane.*

2   Press **BATCHES**, and load the **SEQ** batch from the **Load Batch** screen. The **Batch Start** screen appears. Leave the recipe selection as 'hotmix'. Press the **START** button and hit **OK** to confirm the batch start.

3   Press the **Menu** key, then **Vat** to return to the 'Mixer Vat' user screen. Notice that the readouts are now active, and the batch state is RUNNING. Follow the progress of the batch and confirm that the sequence is correctly driving the batch phases.

4   When the batch has reached the COMPLETE state the sequence should have stopped as well. (Check this by inspecting the SFC_CON block. It's **Run** field should now be **FALSE** again.)

5   Now try running the 'coolmix' recipe. To get the batch from the COMPLETE state back to the RESET state, press the top-left batch pane, then **RESET**, **RESET**. Press **BATCHES** and reload the 'SEQ' batch. (If you need reminding about the various batch states, take a quick look at the batch state diagram in Figure 6-4.)

6   Select the 'coolmix' recipe and start the batch running. Follow its progress in the '**Mixer Vat**' user screen. Towards the end of the 'Heat_up' phase an operator message flags up in the Alarm banner as a flashing question mark. Press the banner to pop up the **Alarms** window, and press **MESSAGES**. You are warned that the '**batch has become unstable**' and are invited to abort the batch. Press the **ABORT** button!

7   The batch pane at the top-left goes red and indicates 'ABORTED' but you will quickly see that the sequence has not stopped running! The vat temperature continues to be controlled according to the current sequence step (phase) despite the fact that the batch state is reported as 'ABORTED' by the 'State' readout.

> *Note*   *This is not too surprising because we have not (yet) arranged for the batch engine to tell the sequence that the batch has been aborted.*

8   Abandon the batch/sequence by pressing the batch pane, then **RESET**, **RESET**. Load and start the batch again, watch it in the user screen, and wait for it to reach the RUNNING state.

9   Now put the batch into the HOLD state, press the batch pane then **HOLD**. The batch pane turns yellow and reports the HELD state, but again you will see that the sequence continues running in blissful ignorance.

> *Note*   *You will also find that there is no simple mechanism allowing you to 'skip' a phase, as there was when you were driving the batch with a setpoint program.*

These deficiencies can easily be made good by some editing of the sequence, and a very minor change to the LIN database. This is what you will now do to complete this chapter of the tutorial.

8.6 ADDING HOLD, ABORT, & SKIP TO THE APPLICATION

To make the application more user-friendly, you will now:

■ Edit the LIN Database by adding a wire to put the sequence into hold

■ Edit the LIN Sequence by adding an 'Abort' step, and 'skip' wiring

■ Edit the User Screen to include a 'skip' button, and also a 'progress bar' for operator information.

Tip!   Remember, Sequences run top to bottom.

8.6.1 Editing the database — Hold

Open up the '**seq.dbf**' file in the LINtools and add a wire from the **bat_ctrl** block's **StateFlg.HELD** field into the **sfc_con** block's **Hold** field. Save the database and download it to the Visual Supervisor.

The *StateFlg.HELD* bit goes high whenever the batch enters the HELD state, and this high signal passes to the sfc_con block's *Hold* field and puts the sequence into 'hold'. The sequence resumes running where it left off as soon as the batch is returned to the RUNNING state.

8.6.2 Editing the sequence — Abort

Figure 8-11 shows how the sequence will look after being modified to add 'abort' functionality. To do this:

1   Double-click the '**seq.sdb**' file to open up the SFC in the LINtools sequence editor.

2   Click the **Step** tool ⊟ and position a new step near the bottom right of the existing sequence (see Figure 8-11). Double-click the new step and name it 'ABORT' in the step pane.

3   Run a wire from 'IDLE' to 'ABORT'. To do this, click the IDLE step's **wire symbol** ↵, then click the ABORT step. The wire plus its highlighted transition appears.
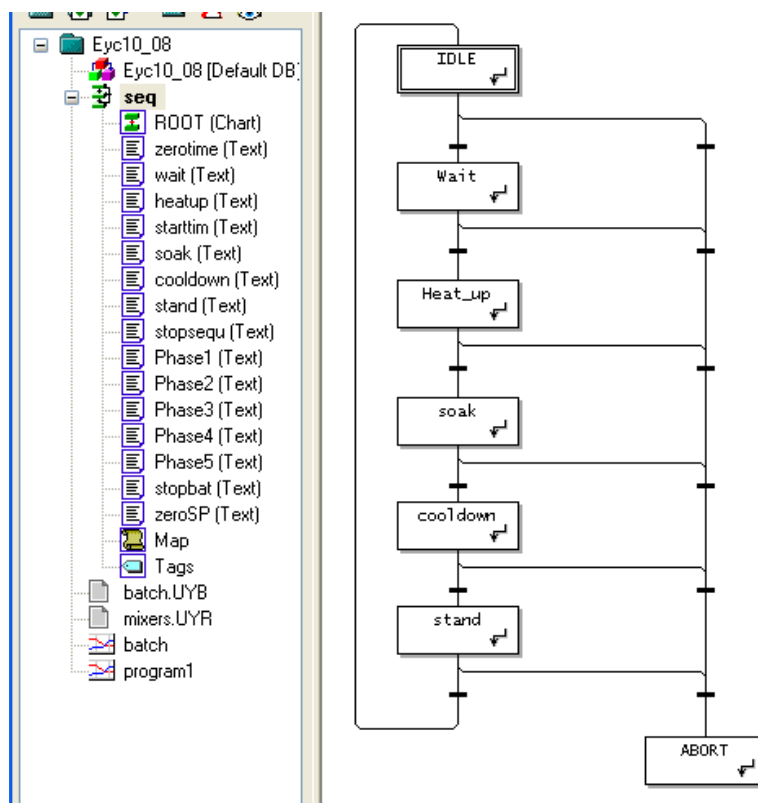
Figure 8-11   The modified sequence — 'abort' added

4   Straighten out the long vertical part of the new wire by dragging the upper wire segment to the right to remove the 'dog-leg'. Drag the new transition bar upwards to be level with the existing IDLE-to-Wait transition.

5   Now wire from 'Wait' to 'ABORT'. Tidy up the wire as in the previous step, so that its vertical segment lies exactly over the previous wire, and its transition bar is about level with the existing Wait-to-Heat_up transition.

6   Wire up the remaining steps to the ABORT step. After tidying, your 'abort' wiring should look something like that in Figure 8-11.

7   Double-click one of the new transitions and type in the following expression (or use the ST prompter to help you):

(bat_ctrl.State = "ABORTED") OR (bat_ctrl.State = "FAILED")

This expression becomes TRUE if the batch enters the ABORTED or the FAILED state. So under these conditions the sequence will jump to the ABORT step from any of the other six steps — which is what we require.

8   Copy and paste this transition expression into the other five new transitions.

ACTIONS ASSOCIATED WITH THE ABORT STEP

When the ABORT step is entered we want to zero and inhibit the timer, and stop and initialise the sequence. It would also be useful to zero the remote PID setpoint. We already have actions that do two of these things ('zerotime' and 'stopsequ'), but we need a new action to zero the remote SP.

To create this new action:

1   In the sequence editor window, click the **Make Action** tool 🎬 to pop up a **Make Action** dialog. Type in a **Name** for the text action, 'zeroSP'. Type in (or use the ST prompter) the following assignment:

rate.PV:=0;   (*Zero remote SP*)

Setting *PV* to zero ramps the RATE_LMT block's target output *OP* to zero. And *OP* feeds into the PID block's remote setpoint, which has the required effect.

2   Now associate these three actions with the ABORT step. Start with the 'zerotime' action. In the ABORT step pane, right-click a cell in the **Action** column and select 'zerotime'. Click the white **Qualifier** cell and select **P - Initial** as the qualifier. We want these actions to happen once only, as soon as the ABORT step becomes active.

3   Associate the 'zeroSP' and 'stopsequ' actions with the ABORT step, qualifying them as 'P - Initial' actions. Make sure that the 'stopsequ' action is last in the list, otherwise the sequence will stop itself before it has done what it is supposed to do!

8.6.3 Editing the sequence — Skip

Figure 8-12 shows how the sequence will look after being modified to add 'Skip' functionality. It consists simply of a short wire linking adjacent steps, via a transition that detects when the 'skip' button on the user screen is being pressed (you'll add this button to the user screen in the next section). Note that you don't need 'skip' wiring between 'IDLE' and 'Wait' because the IDLE step is very short-lived. And if you want to skip the 'Stand' step you can abort the batch, which comes to the same thing at that stage of the sequence.

To add the 'skip' wiring:

1   If necessary, restore the ROOT SFC to the editor window (double-click it in the contents pane).

2   Run a wire from 'Wait' to 'Heat_up'. It will almost certainly look a mess, so increase the zoom by clicking the 🔍 zoom button repeatedly to get maximum magnification. Drag the highlighted transition bar upwards and to the left until it is level with the other two transitions and nicely spaced out. You will find that the 'skip' wire has magically become neat and tidy!

3   Repeat step 2 for the remaining three 'skip' wires.

4   Enter the following transition expression into one of the 'skip' transition panes:

<p align="center"><code>an_conn.PV5 = 1   (*Skip button pressed*)</code></p>

You will configure the user screen to set the AN_CONN block's **PV5** field to '1' when the skip button is pressed. The transition going TRUE will cause activity to pass from the current step to the next step, i.e. a 'skip' to the next transition (phase).

But note that there is a potential problem here. By changing the AN_CONN block's **PV5** field to '1' each transition will be affected. Since *all* the 'skip' transitions will become TRUE when the skip button is pressed, the sequence will skip very rapidly through all its steps in succession until it reaches 'Stand', rather than just skipping one step. This is certainly not what we want!

To avoid this happening we need to create one last text action, 'rsetskip'. This will reset **an_conn.PV5** to zero just as the current step deactivates, so further skipping will not occur (unless the operator presses the skip button again).

To create and associate this ST action:

5   Click the **Make Action** tool 🎬 and enter the name 'rsetskip' in the **Make Action** dialog. Enter the assignment:

<p align="center"><code>an_conn.PV5:=0;   (*Reset 'skip' bit*)</code></p>

6   Return to the ROOT chart, double-click the IDLE step and associate it with the new 'rsetskip' action using a '**P-Initial**' qualifier. We want skip to be reset as soon as the sequence starts, just for safety.

7   Now associate the 'rsetskip' action with the 'Wait', 'Heat_up', 'Soak', and 'Cooldown' steps as the last in their action lists, but this time qualify 'rsetskip' as an '**F - Final**' action.

> *Note   Remember, 'F' qualified actions execute immediately* after *the step deactivates. We want the resetting action to occur only* after *the transition has become TRUE,* not *while the step is still active. (An alternative place for the 'rsetskip' action would be at the* start *of a step, qualified as a 'P' action.)*

8   Save the edited sequence under its existing name, and download it to the instrument.

> *Note   Skip wiring between Steps has been added to the left.*
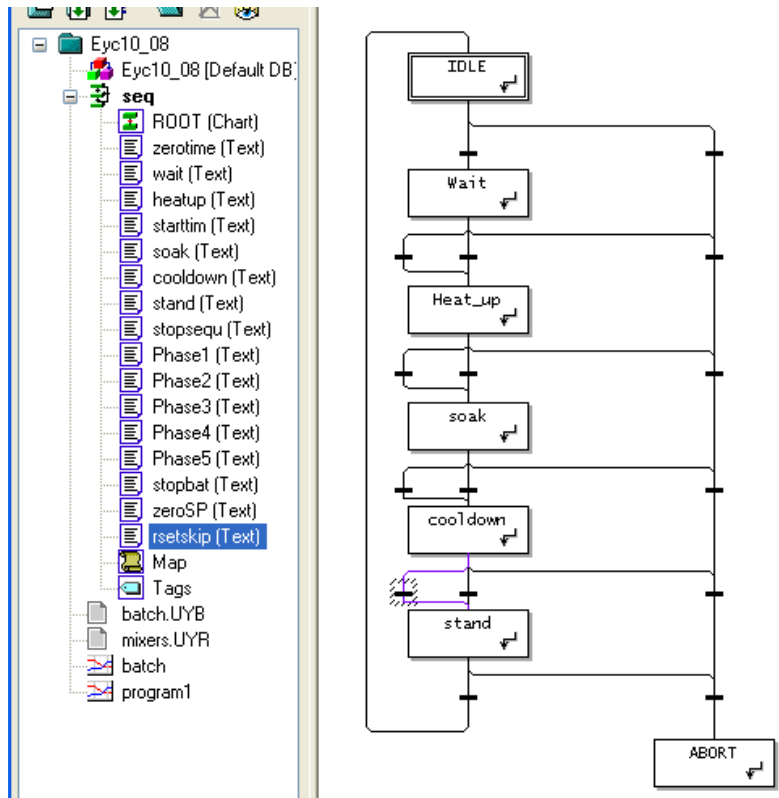
Figure 8-12   The modified sequence — 'skip' added

8.6.4 Editing the user screen — 'SKIP button' & 'progress bar'

We will now add the 'skip' button to the user screen, and also, while we are about it, a convenient 'progress bar' that lets the operator see approximately how much longer a timed phase will take to reach completion. Figure 8-13 shows what we are aiming at.

To do this:

1   Double-click the user screen file '**seq.uxp**' to open the User Screen Editor window. This contains a '**seq.uxp**' subwindow with a blue '**Page1**' icon.

2   Double-click the '**Page1**' icon to open up the user screen, ready for editing. Increase the zoom to 200% for easier working.

3   Click the **Button** tool ▢ and draw a rectangular button about 40 pixels wide by 20 high in a blank area of the screen. Use the **W:** and **H:** indicators in the status bar at the foot of the window to size the button.
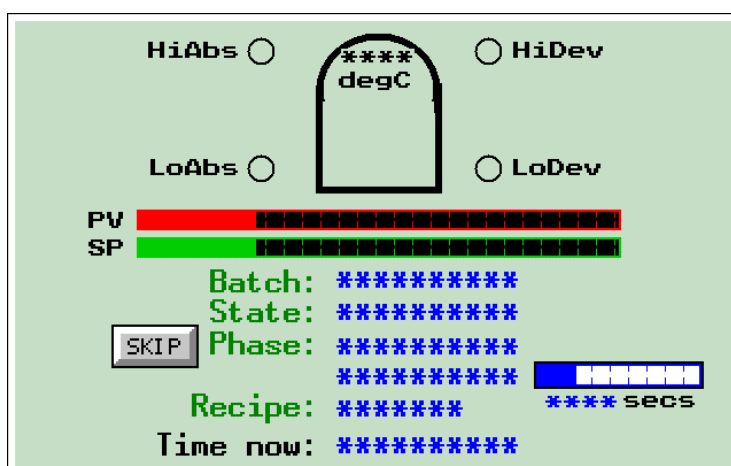
Figure 8-13   The modified user screen — 'skip' button & progress bar added

4   Click the arrow cursor, then right-click the button and select **Properties...** to pop up its **Button Item Properties** dialog. In the **Text** page, select **SmallFont** for the **Font name**, and type 'SKIP' in the **Text** field (i.e. the button legend).

5   In the **Touch Area** page, click the **Action** radio button in the **Touch Area Type** section. In the **Action** field, enter the following 'structured text user screen action'.

$$\text{ST:"[an\_conn.PV5]:=1;"}$$

This will have the effect of setting the AN_CONN block's *PV5* field to '1' when the button is pressed on instrument's screen.

> *Note*   Structured text user screen actions *resemble the sequence actions you have already written. But LIN block field references must be enclosed in square brackets, and each action must be preceded by '**ST:**' and enclosed in double quotation marks.*
> *If you want to know more about this subject, press the Computer's <**F1**> key while the **Touch Area** page is open to pop up a context-sensitive help window.*

6   Hit **OK** to confirm your edits and close the dialog. Drag the skip button to its correct location, to the left of the '**Phase**' legend.

7   Now create a 'progress bar'. Click the **Bar Chart** tool 📊 and draw a horizontal bar 72 pixels wide by 10 high on a blank area of the screen. Position the bar to the right of the second 'Phase' readout (see Figure 8-13).

8   Access the **Bar Chart Item Properties** dialog. In the **Variable** page, select the LIN variable named **timer.CurrTime**. In the **Appearance** page, select a **Blue** Foreground and a **White** Background/Fill. Close the dialog with **OK**.

9   Next, click the **Rectangle** tool ▢ and draw a horizontal 74 by 12 rectangle in a blank area of the screen. This will form a border for the progress bar. The rectangle should be black, of line-weight '1', and unfilled (the defaults). Position the rectangle to frame the progress bar, using the arrow keys to nudge it into the correct position.

10   Now create a readout for the progress bar to display the target time period in seconds. Click the **Text Variable** tool ▣ and drag out a four-character field beneath the progress bar. In the **Appearance** page, make the Foreground **Blue** and the Background/Fill **Pastel Green**. In the **Text** page, select **SmallBold** font, right-aligned. In the **Variable** page, link to the **timer.EndTime** LIN Variable. In the **Format** page, select **0** decimal places. Hit **OK** to apply your edits.

11   Finally, use the **Text** tool *Aa* to create a legend for the readout, 'secs', and position it to the right of the readout. Make the legend **Black**, **SmallBold**, and left-aligned.

12   Save the completed user screen under its existing name, and download the resulting '**seq.ofl**' file to the Visual Supervisor.

8.7 RUNNING THE ENHANCED SEQUENCE/BATCH APPLICATION

Having modified and downloaded the strategy ('seq.dbf'), the sequence ('seq.sdb'), and the user screen ('seq.ofl'), you are ready to test out the enhanced sequence/batch application.

1   In the instrument, stop and unload the unedited application if necessary, and then load and run the enhanced 'SEQ' application. The 'Mixer Vat' user screen appears, with its new 'SKIP' button and progress bar displayed.

2   Make sure that a storage device is connected, then press the batch pane to pop up the **Batch** menu. Press **BATCHES**, and select the **SEQ** batch. Press **LOAD** to see the **Batch Start** screen. Leave the 'hotmix' recipe selected and press **START**, **OK**.

3   Press the **Menu** key then **Vat** to see the user screen. The batch remains in the STARTING state for a short time, then adopts the RUNNING state. In the 'Wait' phase the blue progress bar indicates the passing of the 60-second wait time. After about 30 seconds, try pressing the 'SKIP' button to interrupt the wait time (Figure 8-14). The batch should jump immediately on to the 'Heat_up' phase.
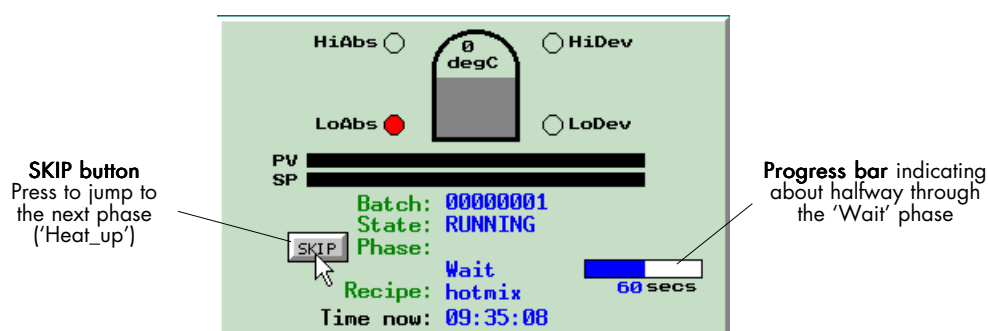
Figure 8-14   Halfway through the 'Wait' phase

4   When the batch is well into the 'Soak' phase, press the batch pane, then **HOLD**. The batch pane turns yellow and indicates the HELD state. Notice that the batch remains indefinitely in this state, even after the 120-second soak time is over. Check that the sequence has also been put into its Hold state, by looking at the sfc_con block's *Hold* field, which should be TRUE. (Press **Menu**, **SYSTEM**, **APPLN**, **FB MGR**, **sfc_con**.)

5   Restart the batch by pressing the batch pane, then **RESTART**. The sequence restarts as well and the batch resumes running at the 'cooldown' phase. Press the 'SKIP' button to jump to the 90-second 'stand' phase, and let the batch proceed to the COMPLETE state.

6   Now test the abort function. Reset the batch by pressing the batch pane, then **RESET**, **RESET**, **BATCHES**. Load the SEQ batch again, then select and start the 'coolmix' recipe. View the user screen.

7   Skip the 'Wait' phase if you are impatient, and watch the progress of the batch. During the 'Soak' phase the usual operator message is flagged by the flashing '?' in the Alarm banner. Press the banner, then **MESSAGES** to see the '**Batch has become unstable!!**' warning. Press the **ABORT** button. The batch aborts, as indicated in the red batch pane. Confirm that the sequence has also stopped running and has been initialised. (Inspect the sfc_con block, *Run* is FALSE and *Init* is TRUE.)

Congratulations! You have created a fully functional batch/sequence
application and have completed this part of the tutorial.

Index

VISUAL SUPERVISOR TUTORIAL/USER GUIDE

Inter-Company sales and service locations

AUSTRALIA Sydney
Eurotherm Pty. Ltd.
Telephone (+61 2) 9838 0099
Fax (+61 2) 9838 9288
E-mail info.au@eurotherm.com

AUSTRIA Vienna
Eurotherm GmbH
Telephone (+43 1) 7987601
Fax (+43 1) 7987605
E-mail info.at@eurotherm.com

BELGIUM & LUXEMBURG Moha
Eurotherm S.A/N.V.
Telephone (+32) 85 274080
Fax (+32) 85 274081
E-mail info.be@eurotherm.com

BRAZIL Campinas-SP
Eurotherm Ltda.
Telephone (+5519) 3707 5333
Fax (+5519) 3707 5345
E-mail info.br@eurotherm.com

DENMARK Copenhagen
Eurotherm Danmark AS
Telephone (+45 70) 234670
Fax (+45 70) 234660
E-mail info.dk@eurotherm.com

FINLAND Abo
Eurotherm Finland
Telephone (+358) 22506030
Fax (+358) 22503201
E-mail info.fi@eurotherm.com

FRANCE Lyon
Eurotherm Automation SA
Telephone (+33 478) 664500
Fax (+33 478) 352490
E-mail info.fr@eurotherm.com

GERMANY Limburg
Eurotherm Deutschland GmbH
Telephone (+49 6431) 2980
Fax (+49 6431) 298119
E-mail info.de@eurotherm.com

HONG KONG & CHINA
Eurotherm Limited North Point
Telephone (+85 2) 28733826
Fax (+85 2) 28700148
E-mail info.hk@eurotherm.com

Guangzhou Office
Telephone (+86 20) 8755 5099
Fax (+86 20) 8755 5831
E-mail info.cn@eurotherm.com

Beijing Office
Telephone (+86 10) 6567 8506
Fax (+86 10) 6567 8509
E-mail info.cn@eurotherm.com

Shanghai Office
Telephone (+86 21) 6145 1188
Fax (+86 21) 6145 1187
E-mail info.cn@eurotherm.com

INDIA Chennai
Eurotherm India Limited
Telephone (+9144) 2496 1129
Fax (+9144) 2496 1831
E-mail info.in@eurotherm.com

IRELAND Dublin
Eurotherm Ireland Limited
Telephone (+353 1) 4691800
Fax (+353 1) 4691300
E-mail info.ie@eurotherm.com

ITALY Como
Eurotherm S.r.l
Telephone (+39 31) 975111
Fax (+39 31) 977512
E-mail info.it@eurotherm.com

KOREA Seoul
Eurotherm Korea Limited
Telephone (+82 31) 2738507
Fax (+82 31) 2738508
E-mail info.kr@eurotherm.com

NETHERLANDS Alphen a/d Rijn
Eurotherm B.V.
Telephone (+31 172) 411752
Fax (+31 172) 417260
E-mail info.nl@eurotherm.com

NORWAY Oslo
Eurotherm A/S
Telephone (+47 67) 592170
Fax (+47 67) 118301
E-mail info.no@eurotherm.com

POLAND Katowice
Invensys Eurotherm Sp z o.o
Telephone (+48 32) 218 5100
Fax (+48 32) 217 7171
E-mail info.pl@eurotherm.com

SPAIN Madrid
Eurotherm España SA
Telephone (+34 91) 661 6001
Fax (+34 91) 661 9093
E-mail info.es@eurotherm.com

SWEDEN Malmo
Eurotherm AB
Telephone (+46 40) 384500
Fax (+46 40) 384545
E-mail info.se@eurotherm.com

SWITZERLAND Wollerau
Eurotherm Produkte (Schweiz) AG
Telephone (+41 44) 787 1040
Fax (+41 44) 787 1044
E-mail info.ch@eurotherm.com

UNITED KINGDOM Worthing
Eurotherm Limited
Telephone (+44 1903) 268500
Fax (+44 1903) 265982
E-mail info.uk@eurotherm.com
Web www.eurotherm.co.uk

U.S.A Leesburg VA
Eurotherm Inc.
Telephone (+1 703) 443 0000
Fax (+1 703) 669 1300
E-mail info.us@eurotherm.com
Web www.eurotherm.com

ED52

**invensys
EUROTHERM**

EUROTHERM LIMITED
Faraday Close, Durrington, Worthing, West Sussex, BN13 3PL
Telephone: +44 (0)1903 268500 Facsimile: +44 (0)1903 265982
e-mail: info.uk@eurotherm.com
Website: http://www.eurotherm.co.uk

HA029587/1 (CN23659)