



Flex.CFE V3.1 Help Manual

TABLE OF CONTENTS:

PREFACE	7
RELATED DOCUMENTS	7
SOFTWARE INFORMATION	7
CONVENTIONS	8
MOUSE CONVENTIONS	8
1 INTRODUCTION	9
2 USING THE FLEX.CFE PROGRAM	11
2.1 STARTING THE FLEX.CFE PROGRAM	11
2.1.1 Connecting to the RealFlex server	12
2.1.2 User Logon	14
2.1.3 Changing the Language	14
2.2 ABOUT THE FLEX.CFE APPLICATION WINDOW.....	18
2.2.1 Title bar	18
2.2.2 Status Bar	18
2.2.3 Toolbar	20
2.2.4 Opening and closing the toolbar	21
2.2.5 Moving the Toolbar	21
2.2.6 Using Flex.CFE Windows	21
2.2.6.1 Resizing and moving windows	21
2.2.6.2 To resize or move a window	22
2.2.6.3 Using scroll bars	23
2.2.6.4 Splitting the window	23
2.2.6.5 Closing files and other windows.....	23
2.2.6.6 Minimizing and maximizing the Main Flex.CFE window	23
2.2.6.7 Minimizing and maximizing files and other windows	24
2.2.6.8 Entering data	24
2.2.7 Output window	25
2.2.8 Setting font sizes	25
2.2.9 Windows menu	26
2.2.10 Exiting the Flex.CFE application	26
2.3 HANDLING FILES	27
2.3.1 Creating a new file	27
2.3.2 Opening an existing file	30
2.3.3 Closing a file	31
2.3.4 Saving a file	33
2.3.5 Saving all files	33
2.3.6 Saving files with a new name	33
2.3.7 Sending files to the RealFlex server.....	33
2.3.8 Deleting a file	35
2.3.9 Checking file syntax.....	35
2.3.10 Print functions	35
2.3.10.1 Print a file	35
2.3.10.2 Print Preview	37
2.3.10.3 Printer Setup	37
2.4 EDITING FILES	38
2.4.1 Selecting all text.....	38
2.4.2 Cut text.....	38
2.4.3 Copy text.....	38
2.4.4 Paste text.....	38



2.4.5	Undo	39
2.4.6	Redo	39
2.4.7	Find and replace text	39
2.4.7.1	Find.....	39
2.4.7.2	Find next.....	41
2.4.7.3	Find previous.....	41
2.4.7.4	Replace	42
2.4.8	Bookmarking.....	43
2.4.8.1	Add a Bookmark	43
2.4.8.2	Goto next bookmark	43
2.4.8.3	Goto previous bookmark.....	43
2.4.8.4	Clear all bookmarks	45
2.5	RESTART REALFLEX AND CSL PROCESS ACTIONS	46
2.5.1	Restart a CSL process.....	46
2.5.2	Restart RealFlex	46
2.6	HELP INFORMATION	47
2.6.1	Accessing Help information	47
2.6.2	About Flex.CFE.....	47
3	MENUS	50
3.1	FILE MENU COMMANDS	50
3.2	EDIT MENU COMMANDS.....	51
3.3	VIEW MENU COMMANDS	51
3.4	OPTIONS MENU COMMANDS	52
3.5	ACTION MENU COMMANDS	52
3.6	WINDOW MENU COMMANDS.....	52
3.7	HELP MENU COMMANDS	52
4	CONFIGURATION FILES.....	54
4.1	CSL FILES	54
4.1.1	CSL Procedures	54
4.1.1.1	CSL Procedure File Organization	56
4.1.1.2	CSL Statement Syntax	56
4.1.1.3	CSL Symbol Tables	57
4.1.1.4	CSL Monitor and Debug Modes.....	57
4.1.1.5	CSL Monitor and Debug Options.....	57
4.1.1.6	Referencing RealFlex Data.....	58
4.1.1.7	Referencing the Realtime Database	59
4.1.1.8	Referencing Historical Data	61
4.1.1.9	Referencing RealFlex Internal Flags.....	61
4.1.2	CSL Keywords	62
4.1.3	CSL Statements.....	64
4.1.3.1	Detailed Explanations of CSL Statements	65
4.1.4	CSL Operators.....	69
4.1.4.1	Detailed Explanation of CSL Operators	70
4.1.5	CSL Calculation functions.....	75
4.1.5.1	The History Calculation Function	75
4.1.5.2	AGA3 Calculation Functions.....	78
4.2	SUPERKEY FILES	82
4.2.1.1	Superkey Procedure Files	82
4.2.1.2	Testing Superkey Logic	83
4.2.1.3	Controlling Superkey Access from the System Summary Screen	84
4.2.1.4	Superkey Status Records	84
4.2.2	Superkey Procedures	85
4.2.2.1	Superkey Language Statement Syntax	85
4.2.2.2	Referencing the Real-Time Database.....	85
4.2.2.3	RealFlex Internal Flags Accessible by Superkey Procedure.....	88
4.2.3	Superkey Keywords and Statements	89
4.2.3.1	Statements.....	89
4.2.3.2	Detailed Explanations of Superkey Statement Types	90
4.2.4	Superkey Operators.....	97
4.2.4.1	Detailed Explanations of Superkey Operators	97
4.2.5	Superkey Examples.....	103



4.3 ALARM DISPLAY CONFIGURATION LAYOUT FILE (ALARM_CONFIG)..... 106

4.4 CHANNELS CONFIGURATION FILE (DRIVER.CHN)..... 109

5 DIALOG BOXES..... 111

5.1 SAVE AS DIALOG BOX 111





Preface

Flex.CFE is part of the Flex.Win suite of programs and is a "Configuration File Editor" application allowing you to create and edit Flex.Win and RealFlex configuration files.

Related documents

The following documents are related to the use of the Flex.CFE program.

- Flex.View Getting Started. Document reference number 5000-0001-0050.
- Flex.View Help Manual. Document reference number 5000-0001-0040.
- Flex.Builder Help Manual. Document reference number 5000-0001-0041.
- Flex.Gallery Help Manual. Document reference number 5000-0001-0042.
- Flex.Start Help Manual. Document reference number 5000-0001-0043.
- Flex.Language Help Manual. Document reference number 5000-0001-0044.
- Flex.Converter Help Manual. Document reference number 5000-0001-0045.

Software information

This Help Manual describes facilities contained in the Flex.CFE application program, version 3.1.1.



Conventions

The following conventions are used throughout this document:

➔ **The beginning of a sequence of instructions:**

1, 2, 3 etc. A set of steps in a sequence of instructions.

□ A single step in an instruction.

Highlight This term defines the action of moving the cursor to illuminate an alphabetic or numeric character, word or phrase to initiate a procedure.

OR: In a sequence of instructions the text OR: is used to indicate a choice of steps. Either execute the one step before the OR: or the one step after it.

Mouse conventions

Within the text the left mouse button is assumed for all mouse operations unless otherwise stated.

Click Press and release the left-hand mouse button without moving the pointer. This action is used to select an object or perform an action.

Double-click Press and release the left-hand mouse button twice in quick succession.

Drag Press the mouse left-hand button without releasing it and then move the pointer. This action tracks the position of the mouse pointer. The action ends when the mouse button is released.

Press Press the mouse left-hand button without releasing it. This action is generally used to select an object for action.

Release Release the mouse button after pressing it. This action is generally used to conclude an action initiated by a press or a drag.



1 Introduction

In order to configure RealFlex there are some "configuration" files which need to be created or edited. The Flex.CFE (Configuration File Editor) application allows a user to configure these files from within Flex.View or Flex.Builder. A user is able to setup his/her system totally independent of the RealFlex/QNX system.

Configuration files that can be created or edited via the "Configuration File Editor" are:

System files

coldstart - this file is run when RealFlex starts up. Within this file a user can tell RealFlex what tasks to run when RealFlex is started. (Changes only take effect when RealFlex has been restarted).
startrf - a file from RealFlex6 which contains the list of processes to be started.
rptcron - this file is a scheduler which allows a user to run automated tasks, (RealFlex does not have to be restarted for changes to take effect).

The editing of the Project "coldstart", "startrf" and "rptcron" files are not discussed within these Help topics. For the "Coldstart" and "startrf" files, users should refer to the RealFlex help file. Within the RealFlex/QNX system, click on the "?" in the menu banner. Select "RealFlex Help" and click on the topic "RealFlex coldstart file" or "RealFlex startrf file".

For the "rptcron" file, users should refer to the "QNX Utilities Reference - A to M", and select the topic "crontab" & "cron".

Alarm configuration files

alarm_config - this file is an Alarm Display Configuration layout file used for building the alarm messages.
event.dial - this file is an alarm states description file.
 Please refer to Section 4.3, "Alarm Display Configuration layout file (alarm_config)" for details about creating and editing the "alarm_config" and "event.dial" files.

Script files

***.csl** - a Control Sequence Language file.
***.prc** - a Superkey process file, e.g., default Superkey procedure file "superkey.prc".
***.cld** - a "calcdo" file used with the "Calcdo" application running on QNX.

Please refer to Section 4.1, "CSL files" for details about creating and editing "*.csl" Control Sequence Language files.

Superkey process files are normally created when adding Superkey Dynamic elements within the Flex.Builder application program. For details, please refer to the "Flex.Builder" Help information. Additional information about *.prc" Superkey process files are included in Section 4.2, "Superkey files".

The creation and editing of "*.cld" Script files are not discussed within these Help topics.

Driver configuration files

***.cfg** - driver configuration files, e.g., Channel information as stored in the "[scanner_name].cfg" file.
driver_chn - the Channels configuration file.
***.ctl** - menu configuration files for pop-up menus in the RealFlex4 System Summary window.



The creation and editing of "*.cfg" and "*.ctl" Driver configuration files are not discussed within these Help topics. Please refer to Section 4.4, "Channels configuration file (driver.chn)" for details about creating and editing the "driver.chn" file.

Editor configuration files

editor.cfg - a user configuration file.

The creation and editing of Editor configuration file is not discussed within these Help topics.

The "Configuration File Editor" option allows a Flex.View or Flex.Builder user to:

- Create or import System files from the RealFlex server, edit them, save them and export them back to the RealFlex server.
- Create or open files for the currently opened Project, edit them, save them and send them back to the Project.
- Open local files stored on the hard-drive of your PC or from a floppy disc or CD, edit them and save them.
- Restart RealFlex on Node 1 or Node 2.
- Restart a CSL process.



2 Using the Flex.CFE program

2.1 Starting the Flex.CFE program

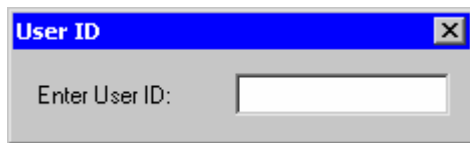
➔ To start Flex.CFE:

- *From within Flex.View:* From the **Configuration** menu, click on the **Configuration File Editor** option.

From within Flex.Builder: From the **File** menu, click on the **Configuration File Editor** option.

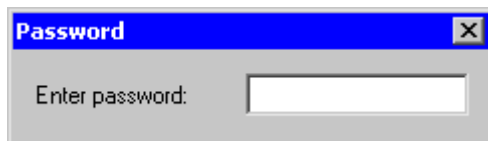
Note: When starting Flex.CFE from within Flex.Builder and you are not connected to the RealFlex server, you must first connect to the server before you can complete the Flex.CFE opening sequence. See Section 2.1.1.

If no one is currently logged on, the "Flex.CFE" application will automatically start to open and a "User ID" dialog box will appear:



Enter your user ID in the "Enter User ID:" field, and press **Return**. User ID's are not echoed to the screen while being entered.

If the "Configuration File Editor" option is protected by a password for the user attempting to login, the "Password" dialog box will appear:



Enter your user password in the "Enter password:" field, and press **Return**. Passwords are not echoed to the screen while being entered.

The Flex.CFE application will then complete the opening sequence.

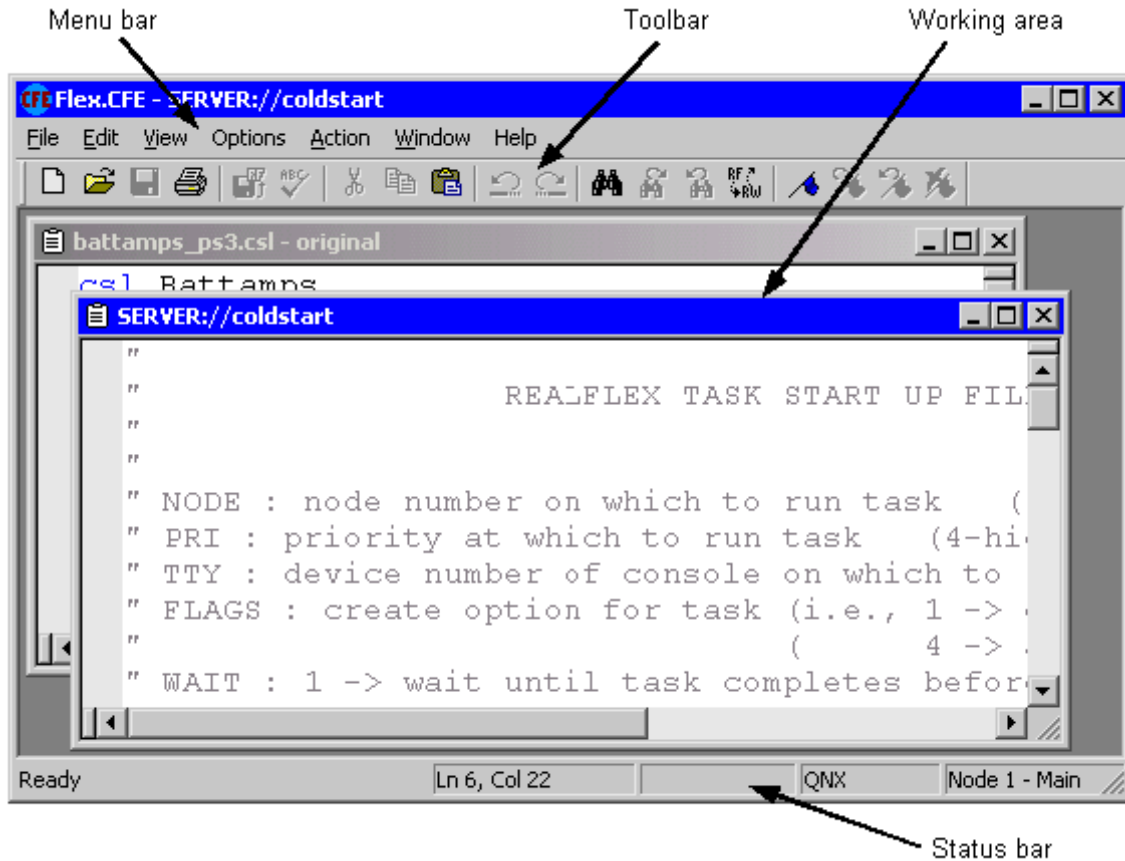


Figure 1: Flex.CFE Application window

2.1.1 Connecting to the RealFlex server

When using Flex.Builder and you are not connected to the server, the "Connect" option from the "File" menu will become active.

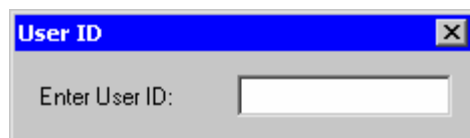
Note: You must be connected to the RealFlex server before you can use the Flex.CFE program.

➔ To connect to the RealFlex server:

- From the **File** menu, select the **Connect** option.

Flex.Builder will attempt to connect you to the Node 1 IP address. Connection progress is reported in the Status bar.

After establishing a connection to the RealFlex server, the "User ID" dialog box will appear:



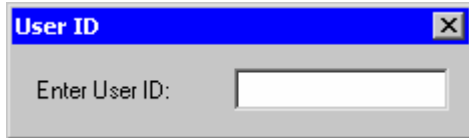
Enter your user ID in the "Enter User ID:" field, and press **Return**. User ID's are not echoed to the screen while being entered.

The Flex.CFE application will then complete the opening sequence.



2.1.2 User Logon

When you try to access the "Configuration File Editor" option from within Flex.View or Flex.Builder, and no user is currently logged on within those applications, the Flex.CFE application will automatically start to open and a "User ID" dialog box will appear:



Enter your user ID in the "Enter User ID:" field, and press **Return**. User ID's are not echoed to the screen while being entered.

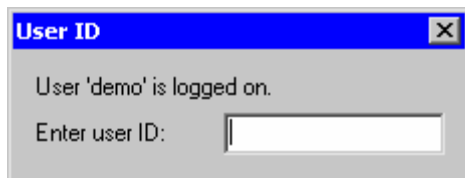
The Flex.CFE application will then complete the opening sequence.

Note: You must be connected to the RealFlex server before you can log on.

➔ To change the logged on user:

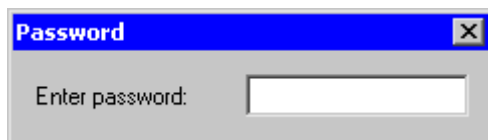
1. From the **File** menu, select the **Login User...** option.

The "User ID" dialog box will appear:



2. Enter your user ID in the "Enter user ID:" field, and press **Return**. User ID's are not echoed to the screen while being entered.

If the "Configuration File Editor" option is protected by a password for the user attempting to login, the "Password" dialog box will appear:



Enter your user password in the "Enter password:" field, and press **Return**. Passwords are not echoed to the screen while being entered.

If no error message appears, the LOGON was successful.

2.1.3 Changing the Language

The "Language..." option from the "Options" menu, allows you to choose and specify a product specific language translation for use within Flex.CFE.

Product specific language translations are produced using the "Flex.Language" program and saved on your Windows PC as a custom translation file in your "C:\Program Files\Datac\Flex.View" directory. For details, please refer to the "Flex.Language" program Help documentation.



➔ **To access the Language window:**

1. From the **Options** menu, select the **Language...** option.

The "Language" window will appear:




The "Language" window is divided into two sections; "Enable product interface language translation" and "Advanced settings".

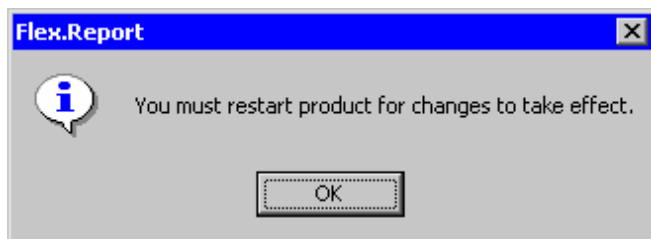
Enable product interface language translation section:

Note: If the "Enable product interface language translation" option is not active, no tick in check-box, then Flex.CFE will use the default language as specified in the "Default language used for all products" field of the "Translation Language Settings" window within the "Flex.Language" program.

➔ **To enable a Product specific language:**

1. Click on the **Enable product interface language translation** check-box. A tick will appear when enabled and the "Use product specific language:" option listed below will become active. Click again to disable (no tick).
2. For the "Use product specific language:" option, choose a user specified language setting for use within Flex.CFE by clicking on the  button to the right of this field and choosing required language from the drop-down list.

If you reset the language for the Flex.CFE program, after choosing the required language from the drop-down list, the following message will appear:



You will need to restart the program before the changes can take effect.

The character set on your PC will change to the selected language. For example, if you have "Spanish" and "English" character sets installed on your PC, you will be able to choose "Spanish" from the drop-down list. Now the character set used will be "Spanish". You are then able to construct a new menu or edit an existing menu in "Spanish". The same applies to all other aspects, e.g., a new text box in a display will be in the "Spanish" character set.

Note: The "Flex.Language" program can be used to remotely set or override the "Enable product interface language translation" settings.

Advanced settings section:

Selecting the **Advanced settings** option, tick in check-box, will reveal the **I want language settings to affect current Windows® user only** option.

Selecting the **I want language settings to affect current Windows® user only** option, tick in check-box, allows you to set the Flex.CFE character set for the user currently logged onto the Windows PC.

For example, if you were to log onto a Windows 2000 PC as "John" and opened Flex.CFE (logged in as "USERA") and have selected "Spanish" as the chosen language, then the character set in Flex.CFE will be Spanish.

If you were to log off the Windows 2000 PC and another user logged on as "David", and opens Flex.CFE (logged in as "USERA"), the character set will revert to the default character set of "English" and not your preferred language of "Spanish".

After all changes have been made to the "Language" window, click on the **OK** button to save your changes, or click on the **Cancel** button to close window and not save changes.



2.2 About the Flex.CFE Application window



2.2.1 Title bar




The Title bar is located along the top of a Flex.CFE application window. It contains the name of the application program and the name of the currently active file.

On the right-hand side of the title bar there are the following buttons:

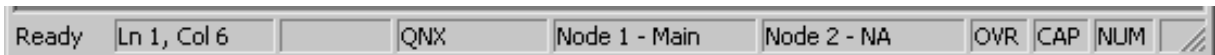
 Minimize button

 Maximize button, or  Restore button

 Close button

2.2.2 Status Bar

Use the "Status Bar" option from the "View" menu to display or hide the Status Bar. The status bar is displayed at the bottom of the Flex.CFE application window.



The left area of the status bar describes actions of menu items as you use the arrow keys to navigate through menus. This area similarly shows messages that describe the actions of toolbar buttons as the pointer is moved over them.

The area on the right of the status bar displays boxes with the following information:

Ln 1, Col 1 The Line of text (Ln) where the insertion point is located and the distance (Col), in number of characters, from the left margin to the insertion point. No measurement is displayed if the insertion point is not in the window.

QNX Two options can be displayed, **DOS** shows that the open file uses DOS format for end of line hidden characters (When used with Real.Win), and **QNX** when used with RealFlex4 or RealFlex6.

Node 1 - Main and **Node 2 - NA** Indicates the status of the Server Nodes. When connected to a RealFlex server, this can be "Node 1 - Main" or "Node 1 - Standby" and the same for Node 2. When connected to the Real.Win server, this will only be "Node 1 - Main".

OVR Indicates the overtype mode status. Press the **Insert** key to turn the mode on or off. When this mode is on, **OVR** is displayed.

CAP Indicates the Caps Lock mode status. Press the **Caps Lock** key to turn the mode on or off. When this mode is on, **CAP** is displayed.

NUM Indicates the Number Lock key status. Press the **NUM LOCK** key to turn the mode on or off. When this is on, **NUM** is displayed.






















➔ **To display or hide the status bar:**

- From the **View** menu, select the **Status Bar** option. A check mark appears next to the menu item when the Status Bar is displayed.

2.2.3 Toolbar

A toolbar is displayed along the top of the Flex.CFE application window. The toolbar provides quick mouse access to many editing functions.

Button	Function
	New file
	Open file
	Save file
	Print
	Send to Server
	Check file syntax
	Cut
	Copy
	Paste
	Undo
	Redo
	Find
	Find next
	Find previous
	Replace
	Set/clear bookmark
	Goto next bookmark
	Goto previous bookmark
	Clear all bookmarks

2.2.4 Opening and closing the toolbar

By default, the toolbar of the Flex.CFE application window is shown. You can show or hide the toolbar whenever necessary.

➔ To show the toolbar:

- From the **View** menu, select the **Toolbar** option. (Open toolbars display a check mark at the left side).

➔ To hide the toolbar:

- From the **View** menu, select the **Toolbar** option. (Closed toolbars do not display a check mark at the left side).

2.2.5 Moving the Toolbar

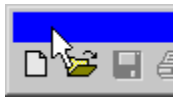
You can move the toolbar to any location in the window.

➔ To move a toolbar:

1. Position the pointer somewhere over an area of the toolbar that does not display a button or drop-down list as shown below.



If the toolbar has already been moved, point to the title bar as shown below.



2. Press and hold the mouse button while you drag the toolbar to a suitable location in your window.
3. Release the mouse button to drop the toolbar to its new location. It will remain in this new place until you move it again or close it.

2.2.6 Using Flex.CFE Windows

Depending on the tasks you do, you may need know about the basic window operations when using Flex.CFE. The following subjects can help you.

2.2.6.1 Resizing and moving windows

During the operation of Flex.CFE, you may find it useful to reduce the size of a window, and/or to move it to a different location on the screen. For example, it may be desirable to simultaneously view two different files, in which case, you could reduce the size of each, and move each to a different side of the screen.

2.2.6.2 To resize or move a window

1. Move the mouse pointer over the edge or corner of the window. The pointer becomes a double-pointed arrow.



2. Drag the window border until the window is the desired size, then release the button.
3. Move the window by clicking on an open area of the Title Bar, and drag the window to the required position in the screen.

2.2.6.3 Using scroll bars



Some of the windows used by Flex.CFE have scroll bars. These are graphical objects along the side and/or bottom of a window, which can be used to move the contents of the window so that a different portion of the contents may be viewed.

To scroll a line or column at a time, click on one of the small triangles in the control button of the scroll bar.



To scroll a page at a time, click on the bar itself.

To scroll to a specific portion of the display, place the cursor on the slider in the middle of the bar, press and hold down the select button on the mouse, and drag the slider button along the scroll bar, releasing the select button when the desired location is reached.


2.2.6.4 Splitting the window

1. With a file open, at the top of the vertical scroll bar or at the left end of the horizontal scroll bar, point to the split box,





2. When the pointer changes to a split pointer,  or , drag to split window down or to the right to the position required. The window will split into two or four panes.

2.2.6.5 Closing files and other windows


- Click on the  (Close) button in the right-hand side of the Title Bar.


Note: Alternative methods of closing files and other windows are:

- Click on the  icon in the Title Bar of the window and select **Close** from the pop-up menu.
- Double-click the windows Control menu button .
- With the window active, key **Ctrl+F4**.

2.2.6.6 Minimizing and maximizing the Main Flex.CFE window


➔ **To minimize the Main Flex.CFE window:**

- Click on the  button in the right-hand side of the applications Title Bar.

Use this command to reduce the application window to a  icon at the bottom of your screen.




➔ **To maximize the Main Flex.CFE window:**

- Click on the  icon at the bottom of your screen, or right-click on the icon and select **Maximize** from the pop-up menu.

2.2.6.7 Minimizing and maximizing files and other windows

A Flex.CFE file, and the operation executing within it can often be set aside while another window is being viewed, then rapidly redisplayed when it is needed.

➔ **To minimize a window:**

- Click on the  button in the right-hand side of the files Title Bar.

Use this command to reduce the window to an icon similar to the one shown below. This icon can be moved anywhere on the screen.



If you have a number of files minimized, you can arrange the icons neatly along the bottom of the Flex.CFE application window using the "Arrange Icons" option from the "Windows" menu. See Section 2.2.9.

➔ **To maximize (or Restore) a window:**

- Click on the icon and select **Maximize** or **Restore** from the pop-up menu, or click on the  or  button.

When Maximized, the window will fill the Flex.CFE application window. When Restored, the window will return to the size and location it had before it was minimized.

A minimized window can also be restored by placing the pointer on the icon and clicking twice in rapid succession (i.e., double-clicking).


2.2.6.8 Entering data

You will enter data and make choices in a number of different ways while using Flex.CFE. The most common methods are described here. For specific details of data entry refer to the help information corresponding to the required task.

There are two buttons that appear on the data input windows:

OK- If you click on "OK" button after you have input data to a window the data will be saved.

Cancel - If you click on the "Cancel" button after you have input data into a window the data is not saved and you are returned to the previous window.

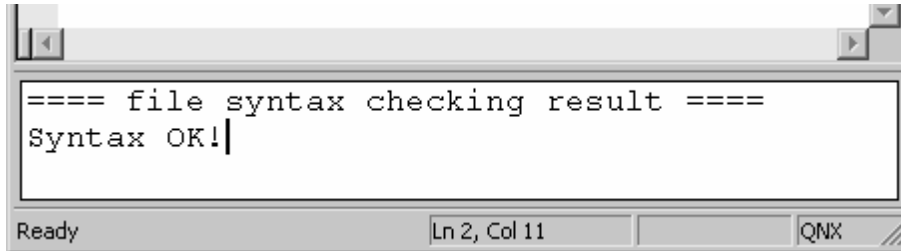
Some fields have a down arrow  button. Clicking on this button will display a drop-down list of choices.



2.2.7 Output window

When you are connected to a RealFlex4 or RealFlex6 server and you have a CSL or Superkey file open, the "Check syntax" option from the "Action" menu becomes active.

When you check syntax, an "Output" window is automatically opened at the bottom of the Flex.CFE application window to report on the Syntax checking. An example is shown below:



➔ To open/close the Output window:

- From the **View** menu, select the **Output** option. A check mark appears next to the menu item when the Output window is displayed.

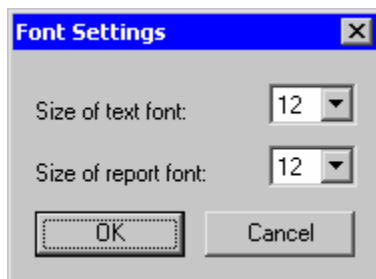
2.2.8 Setting font sizes


Use the "Font settings..." option from the "Options" menu to change the size of fonts used in the currently active window and the Output window.

➔ To change the size of a font:

1. From the **Options** menu, select the **Font settings...** option.

The "Font Settings" window will appear:



2. Choose the font option to be changed by clicking on its  button and selecting required font size from the drop-down list. Options are:

Size of text font: - The font size used in the active window.

Size of report font: - The font size used in the "Output" window.

3. After selecting the font size, click on the **OK** button.

2.2.9 Windows menu

By use of the "Windows" menu, the "Configuration File Editor" application allows you to view files in different ways. Options available are:

Cascade - Use this command to arrange multiple opened files in an overlapped fashion.

Tile - Use this command to arrange multiple opened files in a horizontal non-overlapped fashion.

Arrange Icons - Use this command to arrange the icons of minimized files neatly along the bottom of the main window. If there is an open file covering the bottom of the main window, then some or all of the icons may not be visible because they will be underneath this open file.

Close - Use this command to close the currently active file. If the file has been edited, a dialog box will appear asking if you want to save changes. Select **Yes** to save the changes and close the file. Select **No** to close the file without saving. Select **Cancel** to return to the active file without saving.

Close All - Use this command to close all opened and minimized files. If any files have been edited, a dialog box will appear for each edited file asking if you want to save changes. Select **Yes** to save the changes and close the file. Select **No** to close the file without saving. Select **Cancel** to return to the active file without saving.

1, 2, ... - Flex.CFE displays a list of currently open files at the bottom of the "Window" menu. A check mark appears in front of the file name of the active file. Choose a file from this list to make it active and click on it to switch to the corresponding file.

2.2.10 Exiting the Flex.CFE application

Use the "Exit" option from the "File" menu to end your "Configuration File Editor" session. You can also use the Close command on the application Control menu. Flex.CFE prompts you to save files with unsaved changes.

Shortcuts

Mouse: Double-click the application's Control menu button.



OR:

Click the Close window button  on the right-hand side of the title bar.

Keys: **Alt+F4**

2.3 Handling files

2.3.1 Creating a new file

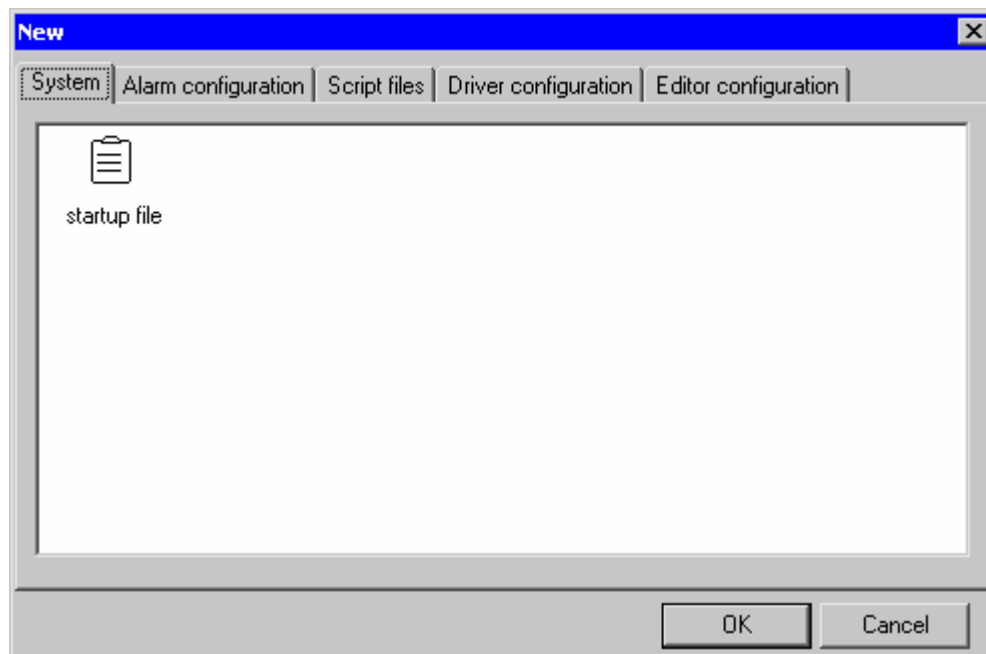
The "New..." option from the "File" menu allows you to create a new configuration file.

➔ **To create a new file:**

1. From the **File** menu, select the **New** option.

Shortcuts: Toolbar  Keypad **Ctrl+N**

The "New" dialog box will appear:



In this dialog box, tabs will be displayed grouping together the different file types. The number of tabs displayed will depend on the type of server you are connected to. In the example dialog box shown above the tabs displayed are for a RealFlex4 or RealFlex6 server.

2. Click on required tab, select file type required then click on the **OK** button.

File types available are:

Tab	File type	Description
System	coldstart	The RealFlex4 startup file
	starttrf.local	The RealFlex6 startup file
	startup file	The rptcron scheduler which allows a user to run automated tasks
Alarm configuration	Alarm Config File	The alarm_config "Alarm Display Configuration layout" file used for building the alarm messages.

Continued on next page

Continued from previous page

Script files	CSL File	A *.csl "Control Sequence Language" file.
	Superkey File	A *.prc "Superkey process" file, e.g., default Superkey procedure file "superkey.prc".
	calcdo File	A *.cld file used with calcdo application running on QNX.
Driver configuration	Telemetry Config File	A driver configuration file, e.g., Channel information as stored in the "[scanner_name].cfg" file.
	driver.chn	The Channels configuration file. Note: If this file is not displayed, then it should already exist on the RealFlex server. Use the "Open" option to edit this file.
	Comm Menu	A file for driver-context menu. If a driver supports special commands that can be issued from the RealFlex4 Communication Summary window, then such commands should be defined in "driver#.ctl" file. Where # - is channel number for PCU.
Editor configuration	Local Config CFE-file	A user configuration file.

For the "alarm_config" file type, a template will be opened in the Flex.CFE application window. For information on how to edit this template and create the "alarm_config" file, please refer to Section 4.3, "Alarm Display Configuration layout file (alarm_config)".

For all other file types, a blank document will be opened in the Flex.CFE application window.

For information on how to create a CSL file, please refer to Section 4.1, "CSL files".

For information on how to create a Superkey file, please refer to Section 4.2, "Superkey files".



2.3.2 Opening an existing file

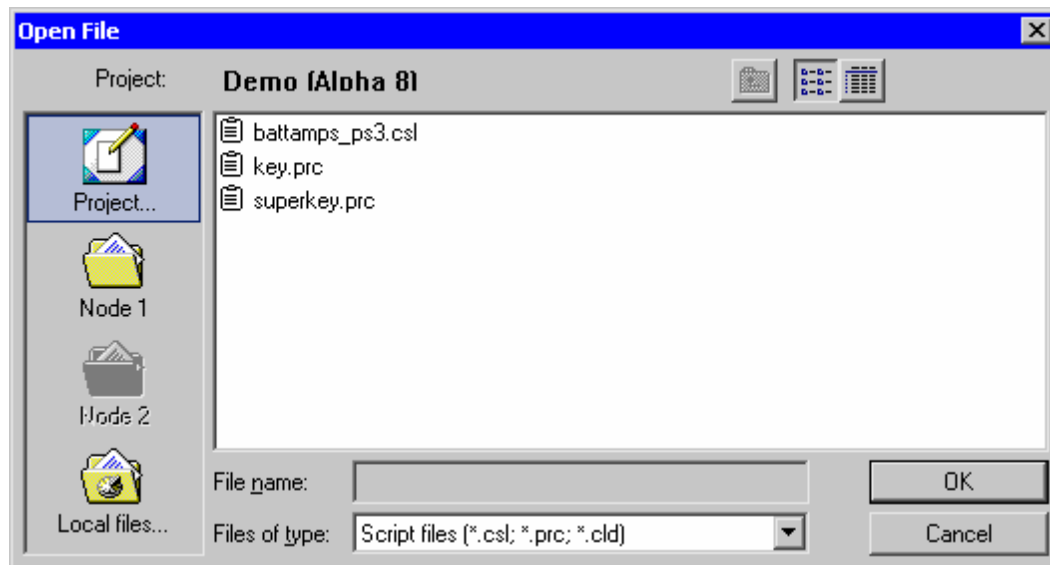
The "Open..." option from the "File" menu allows you to browse for and open existing files saved on the RealFlex server and files saved on your PC for the currently opened Project. You can also browse for and open configuration files stored on the hard-drive of your PC.

➔ To open a file:


1. From the **File** menu, select the **Open...** option.


Shortcuts: Toolbar  Keypad **Ctrl+O**

The "Open File" dialog box will appear:



On the right-hand side of this dialog box, just below the title bar, there are three buttons:

Go to level up:  Moves the "Look in" folder up one level in the directory hierarchy. **Note:** This button is only active when "Local files..." is selected.

List view:  Changes the appearance of items in a folder to a List view.


Details view:  Changes the appearance of items in a folder to a Details view.

2. On the left-hand side of the dialog box, select the required file location:

Project... – Displays files on your PC for the currently opened Project.

Node 1 (Node 2) - Displays files on the RealFlex server.

Local files... – Allows you to browse for and display files on the hard-drive of your PC or a floppy disc or CD. For example, to look at files from other Projects.


3. In the "Files of type:" field, click on the  button and select required file type from the drop-down list.

The file types displayed will depend on the type of server you are connected to. Possible choices are:



Location	Files of type
Project...	Alarm configuration (alarm_config; event.dial)
	Script files (*.csl; *.prc; *.cld)
	Editor configuration (editor.cfg)
Node 1 (Node 2)	System (coldstart; startrf; rptcron)
	Driver configuration (*.cfg; driver_chn; *.ctl)
Local files...	System (coldstart; startrf; rptcron)
	Alarm configuration (alarm_config; event.dial)
	Script files (*.csl; *.prc; *.cld)
	Driver configuration (*.cfg; driver_chn; *.ctl)
	Editor configuration (editor.cfg)

For "Local files...", the following field will appear along the top of the dialog box, just below the title bar. The "File name:" field will also become active:

Look in: Displays the current folder and its list of available folders and files. Double-click on the folder you want to open. To see where the current folder is located in the hierarchy of folders, click on the  button. The resulting drop-down list displays folders above the selected location.

File name: Displays files of the type specified in the "Files of type" field.

4. Select required file then click on the **Open** button.

The selected file will be opened in the Flex.CFE application window.

Current state of files:

For "Project" and "Node 1 (Node 2)" files, a symbol to the left of the file name signifies the current state of the file. Symbols are as follows:



File is an original file and is registered on the RealFlex server or Project.



File is a draft or has been edited, but not yet registered on the RealFlex server or Project.

2.3.3 Closing a file

Use the "Close" option from the "File" menu to close the currently active file without saving.

➔ To close a file:

1. From the **File** menu, select the **Close** option.

Note: If the file has been edited, changes will not be saved.

2.3.4 Saving a file

Use the "Save" option from the "File" menu to save the currently active file.

➔ To save a file:

- From the **File** menu, select the **Save** option.

Shortcuts: Toolbar  Keypad **Ctrl+S**

Files are saved as Draft files until you send the file to the RealFlex server using the "Send to Server" option from the "File" menu.

2.3.5 Saving all files

Use the "Save all" option from the "File" menu to save all the currently opened configuration files.

➔ To save all files:

- From the **File** menu, select the **Save all** option.

Files are saved as Draft files until you send the file to the RealFlex server using the "Send to Server" option from the "File" menu.

2.3.6 Saving files with a new name

Use the "Save As..." option from the "File" menu to save the currently active file under a new name.

➔ To save a file under a new name:

1. From the **File** menu, select the **Save As...** option.

The "Save As" dialog box will appear.


2. Indicate the location and the name of the file to be saved, the file name can differ from the original name, then click on the **Save** button.

The "Save as type" will be automatically set to the file type used for the original file.

2.3.7 Sending files to the RealFlex server

Use the "Send to Server" option from the "File" menu to send edited files to the RealFlex server, i.e., update the Project.


➔ To send a file:

1. Open the files you wish to send to the RealFlex server. These are the files marked with a  symbol in the "Open" dialog box.
2. From the **File** menu, select the **Send to Server** option.

Shortcut: Toolbar 



Flex.View will communicate with the RealFlex server and send the edited file(s) to the appropriate folder(s) on the RealFlex server. Any connection problems or errors are reported.

When the file(s) have been successfully registered on the RealFlex server, a  symbol will now be displayed to the left of the file name in the "Open" dialog box.

2.3.8 Deleting a file

Use the "Delete" option from the "File" menu to remove files.

➔ To delete a file:

1. Open the file you want to delete.
2. From the **File** menu, select the **Delete** option.

You will be prompted to confirm deletion of the file with a "Yes/No" selection.

2.3.9 Checking file syntax

Use the "Check syntax" option from the "Action" menu to check the syntax of files.

When you are connected to a RealFlex4 or RealFlex6 server and you open a certain type of file (CSL or a SUPERKEY), the "Check syntax" option becomes active and you can check the syntax of the file for errors.

➔ To check the syntax of a file:

- From the **Action** menu, select the **Check syntax** option.

Shortcut: Toolbar 

An "Output" window is automatically opened at the bottom of the Flex.CFE application window to report on the Syntax checking. Any errors are reported to the user.

2.3.10 Print functions

2.3.10.1 Print a file

Use the "Print" option from the "File" menu to print the contents of the currently active file.

➔ To print the contents of a file:

- From the **File** menu, select the **Print** option.

Shortcuts: Toolbar  Keypad **Ctrl+P**

The "Print" dialog box will appear where you may specify the destination printer, the range of pages to be printed, the number of copies, and other printer setup options.

2.3.10.2 Print Preview




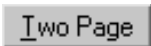
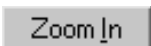
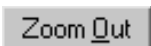
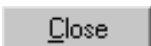
Use the "Print Preview" option from the "File" menu to display the contents of the currently active file as it would appear when printed.

➔ To preview the file contents before printing:

- From the **File** menu, select the **Print Preview** option.

The main window will be replaced with a print preview window in which a page or pages will be displayed in their printed format. The buttons on the print preview toolbar, shown below, offers you options to move back and forth through the pages; zoom in and out of pages; initiate a print job; close the print preview window.

Print Preview toolbar:

	Brings up the Print dialog box, to start a print job.
	Preview the next printed page.
	Preview the previous printed page.
	Preview two printed pages.
	Take a closer look at the printed page.
	Take a larger look at the printed page.
	Exits Print Preview and returns you to the previous Flex.CFE window.

2.3.10.3 Printer Setup

Use the "Print Setup..." option from the "File" menu to change your printing preferences. This command presents a "Print Setup" dialog box, where you may specify the destination printer, the page size and orientation and other printer setup options.

2.4 Editing files

2.4.1 Selecting all text

Use the “Select All” option from the “Edit” menu to select all the text in the currently active file.


- From the **Edit** menu, select the **Select All** option, or click right mouse button and select **Select All** from the pop-up menu.

Shortcut: Keypad **Ctrl+A**

2.4.2 Cut text

Use the “Cut” option from the “Edit” menu to remove text and copy it to the Clipboard.

1. Select the text that you want to cut.
2. From the **Edit** menu, select the **Cut** option, or click right mouse button and select **Cut** from the pop-up menu.

Shortcuts: Toolbar  Keypad **Ctrl+X**

2.4.3 Copy text

Use the “Copy” option from the “Edit” menu to copy text to the Clipboard.

1. Select the text that you want to copy.
2. From the **Edit** menu, select the **Copy** option, or click right mouse button and select **Copy** from the pop-up menu.

Shortcuts: Toolbar  Keypad **Ctrl+C**

2.4.4 Paste text

Use the “Paste” option from the “Edit” menu to paste the contents of the Clipboard (Text only).

1. Position the cursor where you want to paste the text.
2. From the **Edit** menu, select the **Paste** option, or click right mouse button and select **Paste** from the pop-up menu.

Shortcuts: Toolbar  Keypad **Ctrl+V**

2.4.5 Undo

If you decide to remove the last edits you made, you can easily change the text back using the "Undo" command.

➔ To undo the last edits:

- From the **Edit** menu, select the **Undo (Delete or Typing)** option, or click right mouse button and select **Undo (Delete or Typing)** from the pop-up menu.

Shortcut: Toolbar  Keypad **Ctrl+Z**

Continue clicking **Undo** to remove as many of the previous edits as necessary.

Note: When the "Undo" command is unavailable (dimmed), that means you cannot undo the last edit you made.

Tip: If you decide you didn't want to undo an action, select **Redo** from the **Edit** menu.

2.4.6 Redo

The "Redo" command allows you to reverse the action of the "Undo" command.

➔ To redo:

- From the **Edit** menu, select the **Redo (Delete or Typing)** option, or click right mouse button and select **Redo (Delete or Typing)** from the pop-up menu.

Shortcuts: Toolbar  Keypad **Ctrl+Y**

2.4.7 Find and replace text

2.4.7.1 Find

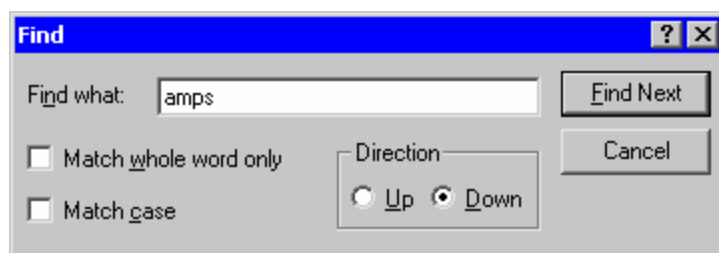
The "Find..." option from the "Edit" menu allows you to search for occurrences of specified text in the current document.

➔ To find text:

1. From the **Edit** menu, select the **Find...** option, or click right mouse button and select **Find...** from the pop-up menu.

Shortcut: Toolbar  Keypad **Ctrl+F**

The "Find" dialog box will appear:



2. In the **Find what:** field, enter the text you want to search for. You can type or paste text into this field.
3. Set the **Match whole word only** and **Match case** choices as required.

When **Match whole word only** is selected, tick in check-box, Flex.CFE searches for occurrences that are whole words and not part of a longer word.

When **Match case** is selected, tick in check-box, Flex.CFE searches for occurrences in which the capitalization matches the text you entered in the **Find what:** field.

4. In the **Direction** section, set the direction for the search.

Up - Searches up from the current insertion point.

Down - Searches down from the current insertion point.

5. Click on the **Find Next** button.

Flex.CFE finds and selects the next occurrence of the text in the **Find what:** field.

2.4.7.2 Find next

The "Find next" option from the "Edit" menu allows you to find and select the next occurrence of the text specified in the "Find what:" field of the "Find" dialog box.

➔ To find the next occurrence:

- From the **Edit** menu, select the **Find next** option, or click right mouse button and select **Find next** from the pop-up menu.

Shortcut: Toolbar  Keypad **F3**

2.4.7.3 Find previous

The "Find previous" option from the "Edit" menu allows you to find and select the previous occurrence of the text specified in the "Find what:" field of the "Find" dialog box.

➔ To find the previous occurrence:

- From the **Edit** menu, select the **Find previous** option, or click right mouse button and select **Find previous** from the pop-up menu.

Shortcut: Toolbar  Keypad **Shift+F3**

2.4.7.4 Replace

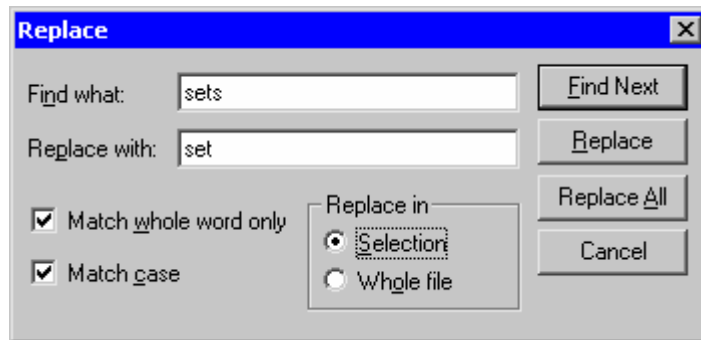
The "Replace" option from the "Edit" menu allows you to replace existing text used in a search. It automates the process of searching for and replacing text manually. You can replace all occurrences at once or confirm the ones you want to replace by viewing each occurrence individually.

➔ To replace text:

1. From the **Edit** menu, select the **Replace...** option, or click right mouse button and select **Replace...** from the pop-up menu.

Shortcut: Toolbar  Keypad **Ctrl+H**

The "Replace" dialog box will appear:



2. In the **Find what:** field, enter the text you want to search for. You can type or paste text into this field.
3. In the **Replace with:** field, enter the replacement text. You can type or paste text into this field.
4. Set the **Match whole word only** and **Match case** choices as required.

When **Match whole word only** is selected, tick in check-box, Flex.CFE searches for occurrences that are whole words and not part of a longer word.

When **Match case** is selected, tick in check-box, Flex.CFE searches for occurrences in which the capitalization matches the text you entered in the **Find what:** field.

5. In the "Replace in" section, choose **Selection** or **Whole file** as required.

The **Selection** option is only active after text has been selected (highlighted) and Flex.CFE will only search for occurrences of the text you entered in the **Find what:** field within the highlighted text.

When **Whole file** is selected, Flex.CFE searches the whole file for the text you entered in the **Find what:** field.

6. Click on the **Find Next**, **Replace**, or **Replace All** button as required.

To cancel a search in progress, click on the **Cancel** button.

2.4.8 Bookmarking


2.4.8.1 Add a Bookmark

Bookmarks within a Flex.CFE configuration file allow you to quickly jump to a specific location within the file.

➔ To add a bookmark:

1. Position the cursor in the line of text where you want to add a bookmark.
2. From the **Edit** menu, select the **Bookmark** option, or click right mouse button and select **Bookmark** from the pop-up menu.

Shortcuts: Toolbar  Keypad **Ctrl+F2**

A Bookmark symbol  will be inserted into the left-hand margin.

2.4.8.2 Goto next bookmark

The "Goto next" option from the "Edit" menu allows you to move to the next bookmark within the currently active file.

➔ To move to the next bookmark:

- From the **Edit** menu, select the **Goto next** option, or click right mouse button and select **Goto next** from the pop-up menu.

Shortcut: Toolbar  Keypad **F2**

Repeat action to move to the next bookmark. The search for the next bookmark is circular. That is, if **Goto next** is selected while at the end of the file, the user will be moved to the beginning of the file.

2.4.8.3 Goto previous bookmark

The "Goto previous" option from the "Edit" menu allows you to move to the previous bookmark within the currently active file.

➔ To move to the previous bookmark:

- From the **Edit** menu, select the **Goto previous** option, or click right mouse button and select **Goto previous** from the pop-up menu.

Shortcut: Toolbar  Keypad **Shift+F2**

Repeat action to move to the previous bookmark. The search for the previous bookmark is circular. That is, if **Goto previous** is selected while at the beginning of the file, the user will be moved to the end of the file.

2.4.8.4 Clear all bookmarks

The "Clear all" option from the "Edit" menu allows you to remove all bookmarks from the currently active file.

➔ **To remove all bookmarks:**

- From the **Edit** menu, select the **Clear all** option, or click right mouse button and select **Clear all** from the pop-up menu.


Shortcut: Toolbar  Keypad **Ctrl+Shift+F2**

2.5 Restart RealFlex and CSL process actions

2.5.1 Restart a CSL process

Use the "Restart CSL process" option from the "Action" menu to restart a CSL program on the RealFlex machine. A CSL is a "Command Sequence Language" which executes a set of instructions on the RealFlex server and returns the result to the Flex.View console.

➔ **To restart a CSL process:**

1. Open the edited *.csl file you wish to restart. This is a file marked with a  symbol in the "Open" dialog box.
2. From the **Action** menu, select the **Restart CSL process** option.

2.5.2 Restart RealFlex

Use the "Restart RealFlex on xxx Node" option from the "Action" menu to restart RealFlex after you have made changes to the "Coldstart" file. This option has the same effect as the command "Restart RF" in the Flex.View "Configuration - Options - Main Menu" option.

➔ **To restart RealFlex:**

- From the Action menu, select the **Restart RealFlex on Main Node** or **Restart RealFlex on BackUp Node** option as required.



2.6 Help information

2.6.1 Accessing Help information

The "Help" option from the "Help" menu links you to the on-line Help and provides Help topics and tips to help you accomplish your tasks.

➔ To access the on-line Help:

- From the **Help** menu, select the **Help** option.

Shortcut: Keypad F1

2.6.2 About Flex.CFE

Selecting "About Flex.CFE" from the "Help" menu will display Flex.CFE program information similar to the display below.

Use this command to display information about the Flex.CFE software you are using, such as; software version number and build number; licence type (Development or Runtime); registration ID and copyright. This information can be given to the support engineer in the event of a problem with Flex.CFE.

- From the **Help** menu, select the **About Flex.CFE** option.

The "About Flex.CFE" window will appear:



From the "About Flex.CFE" window, you can visit the Datac WEB site, or send an e-mail to the Datac Sales department to enquire about other Datac or Flex.Win products.

➔ To send our Sales department an e-mail:

1. Click on the **sales@datac-technologies.com** hyperlink. Your e-mail application will be opened where the Datac contact address will be entered automatically.
2. Type in your query or request and send the e-mail to us in the normal way.



➔ **To visit the Datac Technologies Limited WEB site:**

- Click on the **www.datac-technologies.com** hyperlink.

Your WEB browser will be opened and the Datac WEB site will be dialed.

Click on the **OK** button to close the "About Flex.CFE" window.



3 Menus

3.1 File menu commands

The "File" menu offers the following commands:

New - Allows you to create a new configuration file. For details, please refer to Section 2.3.1.

Open... - Allows you to browse for and open existing files saved on the RealFlex server and files saved on your PC for the currently opened Project. For details, please refer to Section 2.3.2.

Close - Allows you to close the currently active file without saving. For details, please refer to Section 2.3.3.

Save - Allows you to save the currently active file. For details, please refer to Section 2.3.4.

Save As... - Allows you to save the currently active file under a new name. For details, please refer to Section 2.3.6.

Save all - Allows you to save all the currently opened configuration files. For details, please refer to Section 2.3.5.

Send to Server - Allows you to send edited files to the RealFlex server, i.e., update the Project. For details, please refer to Section 2.3.7.

Delete - Allows you to remove files. For details, please refer to Section 2.3.8

Connect - Allows you to connect to the RealFlex server while in Flex.Builder. For details, please refer to Section 2.1.1.

Login User... - Allows a user to log on. For details, please refer to Section 2.1.2.

Print - Allows you to print the contents of the currently active file. For details, please refer to Section 2.3.10.1.

Print Preview - Allows you to display the contents of the currently active file as it would appear when printed. For details, please refer to Section 2.3.10.2.

Print Setup... - Allows you to change your printing preferences. For details, please refer to Section 2.3.10.3.

Exit - Allows you to end your "Configuration File Editor" session. For details, please refer to Section 2.2.10.



3.2 Edit menu commands

The "Edit" menu offers the following commands:

Undo - Allows you to remove the last edits you made. For details, please refer to Section 2.4.5.

Redo - Allows you to reverse the action of the "Undo" command. For details, please refer to Section 2.4.6.

Cut - Allows you to remove text and copy it to the Clipboard. For details, please refer to Section 2.4.2.

Copy - Allows you to copy text to the Clipboard. For details, please refer to Section 2.4.3.

Paste - Allows you to paste the contents of the Clipboard. For details, please refer to Section 2.4.4.

Find... - Allows you to search for occurrences of specified text in the current document. For details, please refer to Section 2.4.7.1.

Find next - Allows you to find and select the next occurrence of the text specified in the "Find what:" field of the "Find" dialog box. For details, please refer to Section 2.4.7.2.

Find previous - Allows you to find and select the previous occurrence of the text specified in the "Find what:" field of the "Find" dialog box. For details, please refer to Section 2.4.7.3.

Replace - Allows you to replace existing text used in a search. For details, please refer to Section 2.4.7.4.

Select All - Allows you to select all the text in the currently active file. For details, please refer to Section 2.4.1.

Bookmark - Allows you add a bookmark in the currently active file. For details, please refer to Section 2.4.8.1.

Goto next - Allows you to move to the next bookmark within the currently active file. For details, please refer to Section 2.4.8.2.

Goto previous - Allows you to move to the previous bookmark within the currently active file. For details, please refer to Section 2.4.8.3.

Clear all - Allows you to remove all bookmarks from the currently active file. For details, please refer to Section 2.4.8.4.

3.3 View menu commands

The "View" menu offers the following commands:

Toolbar - Shows/Hides the Toolbar. For details, please refer to Section 2.2.3.

Status Bar - Shows/Hides the Status Bar. For details, please refer to Section 2.2.2.

Output - Shows/Hides the Output area. For details, please refer to Section 2.2.7.



3.4 Options menu commands

The "Options" menu offers the following commands:

Font settings... - Allows you to change the size of fonts used in the currently active window and the Output window. For details, please refer to Section 2.2.8.

Language... - Allows you to choose and specify a product specific language translation for use within Flex.CFE. For details, please refer to Section 2.1.3.

3.5 Action menu commands

The "Action" menu offers the following commands:

Restart CSL process - Allows you to restart a CSL program on the RealFlex machine. For details, please refer to Section 2.5.1.

Restart RealFlex on Main Node - Allows you to restart RealFlex on the Main Node after you have made changes to the "Coldstart" file. For details, please refer to Section 2.5.2.

Restart RealFlex on BackUp Node - Allows you to restart RealFlex on the BackUp Node after you have made changes to the "Coldstart" file. For details, please refer to Section 2.5.2.

Check syntax - Allows you to check the syntax of files. For details, please refer to Section 2.3.9.

3.6 Window menu commands

For details, please refer to Section 2.2.9.

3.7 Help menu commands

The "Help" menu offers the following commands:

Help - Links you to the on-line Help. For details, please refer to Section 2.6.1.

About Flex.CFE - Displays Flex.CFE program information. For details, please refer to Section 2.6.2.





4 Configuration files

4.1 CSL files

The CSL processor executes programmatic sequences that are written by the user in a simple, easy-to-understand programming language. Command sequences can be executed independently from other RealFlex real-time processes to effect automatic monitoring and controlling of devices based on events, time, logical changes of states, or values received from input devices.

This optional product runs as an independent task under RealFlex, providing the user the flexibility to customize control sequences based on application-specific criteria.

CSL command sequences are defined within the RealFlex system and organized by the user in files. Command sequences are further organized in terms of procedures or routines, where commands performing one logical function are grouped together. CSL procedures are delineated by CSL commands indicating the start of each procedure (the CSL command) and the end of each procedure (the **endcsi** command).

CSL procedures are entered into any editable text file that is named according to the standard QNX system convention and that can be edited using the Configuration File Editor application. Other editing software packages may be used if they are compatible with the QNX Operating System and if the text files are maintained in a plain ASCII format (i.e., formatting characteristics must not be stored in the files).

4.1.1 CSL Procedures

CSL can process an unlimited number of procedures in a single CSL file. Individual CSL procedure execution is controlled by the following items:

- The CSL procedure status point. To change a CSL status point, from within Flex.View, click on the **Main** button, select **RealFlex Summaries**, select **System - PCU** and select the PCU containing the control point. To change the status of a controllable point, click on the value listed to the right of the point. A menu will appear listing the control options for that status point. Click on the desired control option and click EXECUTE, to execute the control change.
- By other CSL procedures setting the status point current value for a specific CSL procedure to active or inactive (ON or OFF).
- By a CSL procedure including and executing a CSL exit statement.

Although CSL procedures may exist in several files, only the CSL procedures in the specified (f) file will be executed by a particular CSL task. This provides the user with a facility to develop a library of CSL files with selective execution. In addition, more than one CSL task can execute concurrently, each with different procedure files.

Please note: CSL uses about 90K of RAM each time it loads into memory. CSL procedures should be kept in the same file where possible.



4.1.1.1 CSL Procedure File Organization

CSL procedure files are organized in a linear block progression of CSL procedures. The command "csl procedure_name" is used to mark the beginning of each procedure, and the command "endcsl" is used to mark the end of each. (One could say that each "csl procedure_name" command serves as the "entry point" of a CSL routine.) Each procedure_name is a unique identifier following the standard RealFlex "tag" naming convention, which allows up to 12 alphanumeric characters. (The underscore (_), period (.), and dash (-) are also allowed.) The procedures are entered into the file sequentially, adhering to the general format shown in the example below. In the example, comments are placed after the semicolon (;) following each applicable line for explanation purposes; they are also used the same way in actual CSL procedure files.

Example: CSL control file block diagram for **monday.tsk**

```
csl amtasks
; Statements in CSL procedure
; for execution/monitoring
endcsl
csl pmtasks
; Statements in CSL procedure
; for execution/monitoring
endcsl
csl endday
; Statements in CSL procedure
; for execution/monitoring
endcsl
; Repeat sequence until
; all CSL procedures defined
etc.
```

When a CSL task is started, the entire procedure file is compiled, the first procedure in the file is executed until a "wait" statement is encountered, and then each of the subsequent procedures is executed one at a time (skipping those procedures whose status control points are set to state 0). CSL continues to execute the procedures in this fashion until a "return" or "exit" statement is executed, or the task is otherwise halted (i.e., by someone "slaying" the task).

4.1.1.2 CSL Statement Syntax

CSL statements may start at any column position on the line. By using a standard style of programming language indentation, the user can create his own CSL formatting standards and enhance readability of CSL procedures. Statements may be terminated with a ";" to allow insertion of comments to the right of the semicolon. This promotes readability and provides documentation for easy maintenance of CSL procedures.

Some expressions require enclosing parentheses () or square brackets [].

When a CSL task is started, all the CSL procedures defined in the specified procedure file are parsed against the allowable CSL syntax for database references, commands and operators. In the event of a parsing error, the CSL task prints the line number where the error occurred and CSL execution halts. If no errors are encountered, the CSL task begins immediate and continuous execution of the parsed CSL procedures.

Although CSL statements may be up to 200 characters long, CSL statements that extend past 80 columns or need to be continued on the next line for clarity may be split using the backslash operator '\'. Do not place comments (;) after a '\' within a CSL statement.



Example:

```
set [an, total_temp, device_1] = [an, temp, device_2] + \
[an, temp, device_3]
```

4.1.1.3 CSL Symbol Tables

The CSL processor allows the user to declare variables (i.e., symbols) as temporary storage units for use within CSL procedures. The symbols for each CSL procedure (csl line to endcsl line) are maintained in a double precision, floating point, symbol table. Each symbol must be declared (using the FLOAT statement defined later in this manual) before, but not necessarily immediately before, it is used in a calculation.

4.1.1.4 CSL Monitor and Debug Modes

The CSL processor provides a monitor mode, two verbose modes and a limited interactive debugger mode. For example, if you were building a new CSL procedure in file test, and you suspected that something was not working properly you could run the CSL procedure by entering the following command:

```
/realflex/bin> Ccsl -m -d -f/realflex/data/test
```

This command will allow you to step through with the key.

Then you could use the debug commands, listed under "DEBUG MODE" below, to control the execution of the CSL procedure. The above command will first parse the csl against the debug mode, and then require the user to hit the key to execute each line of code.

4.1.1.5 CSL Monitor and Debug Options

MONITOR MODE

Monitor mode (-m) prints each numbered line exactly as it appears in the file during the parsing phase of CSL and then prints each numbered executable line as it is reconstructed with each new iteration. For lines that result in setting a datapoint or variable to a value, the reconstructed line includes the identifier of the datapoint or variable in question, any resolved numeric references and operators in postfix notation, and the calculated value to which the datapoint or variable is set. Running monitor mode without any of the other debug options does not require that the user hit the key to execute each line of the CSL. If monitor mode is used in conjunction with other options the user must depress the key in order to execute each line of the CSL.

VERBOSE MODE

Verbose mode (-v) describes each operation in detail.

EXTREMELY VERBOSE MODE

Extremely verbose mode (-E) describes each operation in even greater detail than does verbose mode. Its purpose is to help the developers of the CSL processor to track down any software errors that might be occurring. If you are having problems with CSL that are not easily resolved, Customer Support personnel will often request that you send in a printed copy of your CSL file with the -E mode enabled.

DEBUG MODE

Debug mode (-d) provides a way to control the line by line operation of the CSL file. The DEBUG COMMANDS are single characters entered at the debug mode prompt (CSL:) and are as follows:



ASSIGN

Sets the value of a RealFlex datapoint or CSL symbol (created using the FLOAT command in the CSL procedure) to the value specified. For example, while running CSL in the debug mode, at the "CSL:" prompt, one could enter these commands:

```
CSL: a [an, pipeline, an1] 34
CSL: a counter 100
```

The first command sets the current value of analog datapoint an1, of PCU "pipeline" to 34. The second command sets the value of a variable named "counter" (that has been created via the command "float counter" in the CSL procedure file) to 100.

BREAK (bx)

Sets an execution break at line x (for example, CSL: b 19 sets an execution break at line 19). The step command will automatically execute CSL commands until line x is encountered. The CSL processor will stop execution immediately before beginning line x. Once the break line is encountered, the break condition is cleared.

DEBUG (d)

Toggles debug mode off. When debug mode is turned off the CSL processor will operate as if the debug option had not been selected.

HELP (h)

The help command gives a brief list of CSL debug commands.

MONITOR (m)

Toggles monitor mode on and off.

QUIT (q)

Terminates execution of this CSL.

STEP (s)

Steps through the CSL file line-by-line or proceeds from the current line to the break line if the break condition is set. This is the default command if is pressed with no other command selected.

VERBOSE (v)

Toggles verbose mode on and off.

WHERE (w)

Displays current location in CSL file.

4.1.1.6 Referencing RealFlex Data

Three kinds of RealFlex data can be referenced from within CSL:

1. User-defined data (including the "SYSTEM" PCU).
2. Historical data.
3. RealFlex internal flags.



4.1.1.7 Referencing the Realtime Database

Status, analog, meter and tank records are referenced by using the following syntax:

[DD, PCU, TAG] FIELD, where:

and, PCU records are referenced as follows:

[DD,PCU] FIELD

DD = 2 character mnemonic for the database type.

Possible values are:

"rt" PCU database record
 "an" Analog database record (input and output)
 "mt" Meter database record
 "st" Status database record
 "tk" Tank database record

PCU = 12 character name of associated PCU

TAG = 12 character name for the scanpoint in the RealFlex database. (Not applicable when DD="rt".)

FIELD = A character mnemonic or identifier for a particular field within a scanpoint type. The possibilities are as follows:

If database type is PCU (i.e., DD = "rt")

ACTIVE = PCU active state (default)
 CHAN = PCU control queue channel
 ADDR = PCU physical address
 TSCANS = PCU total scans
 VSCANS = PCU valid scans
 RETRIES = PCU scan retries
 NO RESP = PCU no response
 ERRORS = PCU data errors
 COMMEFF = PCU communications efficiency

If database type is ANALOG (i.e., DD="an")

CVAL = Current EU value (default)
 HI = The high alarm threshold
 HIHI = The high high alarm threshold
 LO = The low alarm threshold
 LOLO = The low low alarm threshold
 MAX_EU = The maximum EU value
 MIN_EU = The minimum EU value

If database type is METER (i.e., DD = "mt")

CVAL = current net accumulation (default)
 LHOURL = last hour's accumulation
 HOURL = current hourly accumulation
 DAY = current daily accumulation
 MONTH = current monthly accumulation
 YEAR = current yearly accumulation
 GROSS = current gross accumulation
 FACTOR = meter multiplication factor



YESTERDAY = yesterday's accumulation
 LAST_GOOD = last good meter reading from device

If database type is STATUS (i.e., DD = "st")

CVAL = current value (default)

If database type is TANK (i.e., DD = "tk")

PVOL = product volume (default)
 AVOL = available volume
 TEMP = temperature
 PROD = product code
 FEET = tank level in feet
 INCH = tank level in inches
 FRAC = tank level in fractions (8ths or 16ths)
 GRAV = product specific gravity

Any referenced scan points must have been defined via the RealFlex database configuration processor prior to starting the CSL tasks. Invalid TAG name, PCU name, database type or field references will be flagged at startup and will cause the entire CSL process to terminate. The RealFlex system must also be running, because the CSL task accesses the database through regular RealFlex data handling routines.

Using "ALL" and "ANY" - The CSL processor allows the user to use the special datapoint tags "all" and "any" to perform database-wide or pcu-wide processing. In "if" statements, the "all" tag requires that every tag of the appropriate type pass the conditional processing. When the "any" tag is used, if any tag of the appropriate type passes the conditional processing, the condition is satisfied. The "all" tag can be used to assign all points of a given type to a particular value. Using "all" and "any" can simplify CSL programming, but there is a significant price to pay in terms of execution speed, and "any" and "all" tags should be avoided when more explicit expressions will work as well.

"ALL" and "ANY" tags may only be used in "left expressions". That is, they must always be to the left of the binary operator.

SYNTAX:

```
[an, all, all]
[an, pcu_name, all]
[an, any, any]
[an, pcu_name, any]
```

Example:

```
csl test
  if ([an, test, any] > 0) or ([an, power, all] > 0)
    set [st, test, all] = off
  endif
  if ([an, test, any] > 0)
    set [an, test, all] = 0
  else
    stuff [an, test, all] = 1
  endif
  wait 3
endcsl
```



In this example, if any analog point in PCU "TEST" has a value greater than zero, or if all analog points in PCU "POWER" have values greater than zero, then all status points in PCU "TEST" will be "set" to their "off" states. Also, if any analog point in PCU "TEST" has a value greater than zero, then all analog points in PCU "TEST" will be "set" to zero; otherwise, the value 1 will be "stuffed" into all of the analog points in PCU "TEST".

Error examples:

("All" and "any" are not allowed to the right of the operator.)

```
if (0 > [an, test, any])
if ([an, test, any] > [an, test, all])
```

("Any" is not allowed to the left of the operator on a "STUFF" statement.)

```
stuff [an, test, any] = 100.5
```

4.1.1.8 Referencing Historical Data

The CSL processor gives users automatic access to the RealFlex historical database, via a predefined "calc function".

4.1.1.9 Referencing RealFlex Internal Flags

In addition to the external, user definable database types, the CSL task also may be used to manipulate the internal RealFlex control flags associated with each defined scan point in the system. These flags provide the user with a means to perform manual overwrite of a scan point, take a scan point out of scan processing, and to monitor the PCU communications health status of messages sent and received.

Please Note: The following alarm flags are read (or monitor) only flags: lo_alarm, hi_alarm, lolo_alarm, hihi_alarm, un_ack and alarm. CSL will not clear or set the alarm flags.

RealFlex Internal Flags Accessible by CSL:

To set any of the following flags that are not write protected, you must use the STUFF command.

NO_REPLY PCU no-reply flag
 INVALID Data point invalid (not updated) flag
 INSTR_FAIL Instrument failure (over/under range) flag
 for analog datapoints
 COLD_START RealFlex cold start flag (i.e., first time
 data used)
 OVER_WRITE Data point manual overwrite flag
 LO_ALARM Low alarm flag for analog and tank points
 (Read ONLY)
 HI_ALARM High alarm flag for analog and tank points
 (Read ONLY)
 LOLO_ALARM Low low alarm flag for analog and tank points
 (Read ONLY)
 HIHI_ALARM High high alarm flag for analog and tank
 points (Read ONLY)
 UN_ACK Data point alarm unacknowledged flag
 (Read ONLY)
 ALARM Global alarm flag for the specific database
 point referenced (Read ONLY)
 CONTROL Data point control pending flag



CTRL_TAGGED Control tag for status points (Read ONLY)
 INFO_TAGGED Information tag for status, analog, meter
 and tank points (Read ONLY)

Table 1.1

FLAG PCU STATUS ANALOG METER TANK

FLAG	PCU	STATUS	ANALOG	METER	TANK
NO_REPLY	R/W	N/A	N/A	N/A	N/A
INVALID	N/A	R/W	R/W	R/W	R/W
INSTR_FAIL	N/A	N/A	R/W	N/A	N/A
COLD_START	N/A	R/W	R/W	R/W	R/W
OVER_WRITE	N/A	R/W	R/W	R/W	R/W
LO_ALARM	N/A	N/A	R	N/A	R
HI_ALARM	N/A	N/A	R	N/A	R
LOLO_ALARM	N/A	N/A	R	N/A	R
HIHI_ALARM	N/A	N/A	R	N/A	R
UN_ACK	R	R	R	R	R
ALARM	R	R	R	N/A	R
CONTROL	N/A	R/W	R/W	R/W	N/A
CTRL_TAGGED	N/A	R	N/A	N/A	N/A
INFO_TAGGED	N/A	R	R	R	R

4.1.2 CSL Keywords

Keywords used in CSL statements to represent status record attained or controlled states are the same as those defined for the scan point during the database building process. Any of the six letter "state descriptor" mnemonics specified for a status point in the database configuration process may be used in CSL statements.

In addition, CSL translates a few useful keywords into a constant value as an aid to CSL processing.

CSL_OFF

Used as a symbolic constant for the value 0.

CSL_ON

Used as a symbolic constant for the value 1.

Example:

```
if ([an, system, cancel] over_write = csl_on)
  stuff [an, system, csl_start] no_reply = csl_off
endif
```

TIME

Time is returned in a floating point format as the total number of minutes in the day with seconds as a fraction of minutes. For example, a time of 90 seconds past midnight would be returned by time as 1.5. The value for the keyword time can be captured and stored in an analog database record.

Example: stuff [an, system, time_holder] = TIME

Any given time of day may be referenced in CSL as hours and minutes with a colon separating the two numbers (hh:mm) or as hours, minutes, seconds with colon separators (hh:mm:ss). Time is specified in terms of a 24 hour clock (from 00:00:00 to 23:59:59).



Example:

```
if ((time > 12:00) and (time < 13:00))
  set [st, system, lunch] = now
endif
```

MINUTE

Current minute of hour - ranges from 0 to 59

HOUR

Current hour of day - ranges from 0 to 23 (0 = midnight)

MONTH_DAY

Current day of month - ranges from 1 to 31

MONTH

Current month of year - ranges from 1 to 12

YEAR_DAY

Current Julian day of year - ranges from 1 to 366

WEEK_DAY

Current day of week - ranges from 1 to 7 (1 = Sunday)

Example:

```
if ((MONTH = 1) and (MONTH_DAY = 1) \
  and (HOUR = 0) and (MINUTE = 0))
  set [st, SYSTEM, NEW_YEAR] = TRUE
endif
```

4.1.3 CSL Statements

CSL is a simple language in which users can write "programs" which monitor and control individual devices based on events, time or values of database points. Programming experience is not required to use CSL; however, the user must have a basic understanding of the task(s) which need to be performed in the particular application, and enough knowledge to enter the text into a file in the correct syntax.

Note: { } denotes optional parameter

```
CALC MONTH_DAY
CONTINUE WHEN (expression) PCU_OFF/PCU_ON
CSL procedure name PRINT_SCREEN
CSL_OFF RECIPE recipe_name
CSL_ON REMOVE_DISPLAY
DEMAND_SCAN REPORT report_type
DISPLAY format_name {node} RETURN
DSAVE database_type SEND a = b
DSTUFF a = b SET a = b
ENDCSL SHELL QNX_command
EXIT STEP n
FLOAT STUFF a = b
```



GOTO n TIME
 GOTO [an,PCU,TAG] n1 n2 .. n9 WAIT n
 HOUR WARMSTART
 IF with ELSE and ENDIF WEEK_YEAR
 MINUTE YEAR_DAY
 MONTH

(Portions in italics are replaced with user-supplied expressions and values.)

4.1.3.1 Detailed Explanations of CSL Statements

CALC

The CSL calc command provides a C language subroutine call to a predetermined set of C functions. In general these functions, written in the C programming language, provide a mechanism for complex mathematical calculations in CSL. Each named calc function has a set of named arguments that must be present in the calc invocation. Each named argument must be followed by an equals sign (=) and either a constant or a RealFlex datapoint reference. The end of the argument list is signaled by the reserved word endcalc. The Endcalc argument is followed by the RealFlex datapoint or CSL float variable that will store the calculation answer (see example next page). The answer for the calculation is placed directly in the CSL variable or passed to RealFlex via the raw data queue in the same way as a CSL set command (i.e., as if being reported from a field device).

If the calc function name or any of the calc arguments are not found, or the argument is inappropriately assigned, it is a compile time error. The order of the calc arguments between calc and endcalc is not important. The line continuation symbol '\ ' is not required between the calc and the endcalc commands. Character case is ignored in the calc call.

SYNTAX

```
calc Name
  arg1 =
  arg2 =
  ...
  argn =
endcalc
```

Examples:

```
calc nx19
  Pressure = [an, pcu, pressure]
  Temperature = [an, pcu, temperature]
  Gravity = 0.6
  Co2 = [an, pcu, co2]
  Nitrogen = [an, pcu, nitrogen]
endcalc [an, system, nx19]
```

```
float press
float f_supercomp
stuff press = [an, pcu, pressure]*1.234
calc nx19
  Pressure = press
  Temperature = [an, pcu, temperature]
  Gravity = 0.6
  Co2 = [an, pcu, co2]
  Nitrogen = [an, pcu, nitrogen]
endcalc f_supercomp
```



CONTINUE WHEN (*expression*)

Used to hold execution of the CSL procedure until the expression is evaluated as TRUE. After that time, execution proceeds. Use parentheses to specify and/or clarify order of evaluation in a compound expression.

CSL *tag_name*

Must be the first statement of a CSL procedure. The "tag_name" represents a unique, maximal 12 character identification for the procedure.

CSL_OFF and **CSL_ON**

Keywords used as a symbolic constants for the values 0 and 1, respectively.

DEMAND_SCAN

Sends "demand scan" control to the applicable I/O driver for the specified PCU. Does not change the communications summary, active on/off state to reflect a demand scan status. Note for users of the Custom Development Package: DEMAND_SCAN is a named constant for 9. (See the Custom Development Package file dbtype.h).

DISPLAY *format_name* {*node_number*}

Used to automatically display a screen format on the specified node. If node number is not specified, the current node is assumed. If you are not sure what node number is, type in the command "who" at the "\$" prompt and it will return node number.



WARNING: Calling up multiple displays may exhaust available RAM. If this occurs, you should begin closing windows in order to ease the strain on memory. (For automatic removal of displays, see the description of the REMOVE_DISPLAY command).

DSAVE *database type*

Forces a file-based save of the entire database type to hard disk. The RealFlex database types that can be saved by this command are specified as PCU, analog, meter, status, tank, and crt_fmt. (All writes to disk are buffered and handled by task memdb every 30 seconds; therefore, there is a delay of up to 30 seconds after this command is executed before the save to disk begins). Please note that placing a DSAVE command in a tight logic loop is CPU intensive and may place undue stress on your hard drive.

DSTUFF A=B

Used to "stuff" data directly into the RealFlex system bypassing any normal RealFlex data processing and additionally save the individual destination record to hard disk. The hard disk save versus RAM disk only save is the difference between the "DSTUFF" and the "STUFF" command. (As with DSAVE, there will be a delay of up to 30 seconds after this command is executed, before the write to disk begins). DSTUFF syntax sets the specified database point on left side of the "=" to the result of the expression on the right side of the "=". Please note that placing a DSTUFF command in a tight logic loop can place undue stress on CPU performance and on your hard drive.

Example:

```
; Store evaluation of B directly to A in RAM memory and additionally, write A to hard disk.
dstuff A = B
```

ENDCSL

Used to mark the end of a CSL procedure.



EXIT

End processing of current CSL procedure and turn this CSL procedure off. Subsequent tests for procedure execution will fail until some action is taken to activate the CSL procedure again.

FLOAT

Used to declare variables (temporary storage units for double precision, floating point numbers). A variable can be used only within the individual procedure in which it is declared. (The same name may be used for variables in several different CSL procedures, but changing the value of one of these variables does not change the value of the others). Each variable must be declared before, but not necessarily immediately before, it is used in a calculation. Each variable is automatically initialized to 0.0 upon creation. Only the STUFF command and the CALC command (via the "endcalc") can be used to place data in a variable.

Please Note: The FLOAT command treats a "-" as a minus operation. Do not use a dash in the name of your float variable.

SYNTAX:

FLOAT variable_name

Example:

```

csl counter
  float count
  step 10
  if ([st, pipeline, pump]) = run)
    stuff count = count + 1
  endif
  wait 60
  goto 10
endcsl

```

GOTO n

Used to control execution flow. Proceed at STEP number "n" of the procedure. It is strongly recommended that users execute a WAIT between STEP and GOTO commands whenever infinite loop processing is a possibility.

Example:

```

step 1
  ;statements
  wait 1
  goto 1

```

GOTO [an, pcu, tag] n1 n2 .. n9

Performs a goto step based on the value found in the referenced analog database point. This command compares the value found in the analog database point with each of the numbers n1 n2 .. n9. If the analog value matches one of these numbers then execution continues at the STEP statement with the same number. If no match is found, execution continues at the STEP statement with the first number in the list (n1). The list may contain up to nine numbers, all of which must appear in STEP statements.

HOUR

Keyword - Current hour of day, ranges from 0 to 23, where 0 is midnight



IF with ELSE and ENDIF

Use this statement to control logic flow within a CSL procedure. Every IF must have a corresponding ENDIF to mark the boundary of the statement(s) executed if the expression is TRUE. Simple IF expressions do not need parentheses. Compound IF expressions, which are those IFs with more than one check, need to be grouped by parentheses to specify and/or clarify order of evaluation. The second IF statement structure provides an optional ELSE statement(s) to be executed when the IF statement expression(s) proves to be FALSE. Nesting IF and ELSE and the statements are often used to send the pointer to another line of code based on the procedure which needs to be performed.

Example 1:

```
IF (expression)
  statement(s)
ENDIF
or
IF (expression)
  statement(s)
ELSE
  statement(s)
ENDIF
```

Example 2:

```
IF [st, pipeline, an1] = on
  goto 10
ELSE
  if [st, pipeline, an1] = off
  goto 5
ENDIF
```

>MINUTE

Keyword - Current minute of hour, ranges from 0 to 59 More CSL Keywords

4.1.4 CSL Operators

CSL provides limited logical and mathematical processing through the use of reserved operators. Some operators function as both logical and arithmetic operators. Use parentheses to develop compound mathematical statements. Operators provided in the CSL language are listed below and detailed explanations are presented below.

```
"=" equal (or set equal to)
"<>" not equal to
">" greater than
">=" greater than or equal to
"<" less than
"<=" less than or equal to
OR logical "or"
AND logical "and"
"+" addition
"-" subtraction
"*" multiplication
"/" division
%" modulus arithmetic
"^" exponentiation (including ^2 for squaring)
SQRT square root
LOG natural log
ABS absolute value (integer)
```



FABS absolute value (float)
 SIN trigonometric sin (degrees)
 COS trigonometric cosin (degrees)
 TAN trigonometric tangent (degrees)
 RSIN trigonometric sin (radians)
 RCOS trigonometric cosin (radians)
 RTAN trigonometric tangent (radians)

Items located at the bottom of the above list have higher precedence in the evaluation order than items located at the top. In other words, evaluation precedence is at its highest for RTAN and at its lowest for =. Proper placement of parentheses may be used to override the default precedence.

4.1.4.1 Detailed Explanation of CSL Operators

"="

Evaluate or set expression equal

Examples:

1. Compare the value of status point "amtasks" in PCU "SYSTEM" with the user defined state mnemonic "start".

```
if ([st, system, amtasks] = start)
```

2. Set analog point "an1" in PCU "SYSTEM" equal to a numeric constant.

```
set [an, system, an1] = 1
```

"<>"

Evaluate expression not equal

Example:

Evaluate status point "st1" in PCU "SYSTEM" against user defined state mnemonic "start".

```
if ([st, system, st1] <> start)
```

">"

Evaluate expression or constant greater than

Examples:

1. Evaluate analog point "an1" in PCU "PCU1" against numeric constant

```
if ([an, PCU1, an1] > 10)
```

2. Compare two analog points

```
if ([an, pcu1, an1] > [an, pcu2, an1])
```



">="

Evaluate expression or constant greater than or equal to

Example:

Evaluate analog point "an1" in PCU "PCU1" against numeric constant

if ([an, PCU1, an1] >= 10)

"<"

Evaluate expression or constant less than

Example:

Evaluate analog point "an1" in PCU "PCU1" against a numeric constant.

if ([an, PCU1, an1] < 10)

"<="

Evaluate expression or constant less than or equal to

Example:

Evaluate analog point "an1" in PCU "PCU1" against a numeric constant.

if ([an, PCU1, an1] <= 10)

OR

Evaluate two expressions and perform conditional instructions if either expression is true.

Example:

Evaluate analog points "an1" and "an2" in PCU "PCU1" against numeric constants

if (([an,PCU1,an1] < 1) or ([an,PCU1,an2] < 99))

AND

Evaluate two expressions and perform conditional instructions only if both expressions are true.

Example:

Evaluate analog points "an1" and "an2" in PCU "PCU1" against numeric constants

if (([an, PCU1, an1] < 1) and ([an, PCU1, an2] < 99))

"+"

Perform arithmetic addition operation

Example:

Perform addition operation on analog point "an1" in PCU "PCU1"

set [an, PCU1, an1] = [an, PCU1, an1] + 100



"-"

Perform arithmetic subtraction operation

Example:

Perform subtraction operation on analog point "an1" in PCU "PCU1"

```
set [an, PCU1, an1] = [an, PCU1, an1] - 100
```

"*"

Perform multiplication operation

Example:

Perform multiplication operation on analog point "an1 " in PCU "PCU1"

```
set [an, PCU1, an1] = [an, PCU1, an1] * 100
```

"/"

Perform division operation

Example:

Perform division operation on analog point "an1" in PCU "PCU1"

```
set [an, PCU1, an1] = [an, PCU1, an1] / 100
```

"%"

Perform modulus arithmetic

Example:

Perform modulus 10 arithmetic on analog point "an3 " in PCU "PCU2"

```
set [an, PCU2, an3] = [an, PCU2, an3] % 10
```

"^"

Perform exponentiation operation (use "^2" to square a value)

Example:

Perform exponentiation operation of cube on analog point "an3" in PCU "PCU2"

```
set [an, PCU2, an3] = [an, PCU2, an3] ^ 3
```

SQRT

Perform square root operation

Example:

Perform operation of square root on analog point "an3 " in PCU "PCU3"

```
set [an, PCU3, an3] = SQRT [an, PCU3, an3]
```



LOG

Perform natural log operation

Example:

Perform operation of natural log on analog point "an3" in PCU "PCU3"

```
set [an, PCU3, an3] = LOG [an, PCU3, an3]
```

ABS

Perform absolute value operation (integer argument)

Example:

Perform operation of absolute value on the value -5 and put the result (5) in analog point "an3" in PCU "PCU3"

```
set [an, PCU3, an3] = ABS(-5)
```

FABS

Perform absolute value operation (floating point argument)

Example:

Perform operation of absolute value on analog point "an3" in PCU "PCU3" (if the point contains (for example) -5.2, it will be changed to 5.2)

```
set [an, PCU3, an3] = FABS [an, PCU3, an3]
```

SIN

Calculate trigonometric sin (argument in degrees)

Example:

Calculate the sin of analog point "an3" in PCU "PCU3"

```
set [an, PCU3, an3] = SIN [an, PCU3, an3]
```

COS

Calculate trigonometric cosine (argument in degrees)

Example:

Calculate the cosin of analog point "an3" in PCU "PCU3"

```
set [an, PCU3, an3] = COS [an, PCU3, an3]
```



TAN

Calculate trigonometric tangent (argument in degrees) Note: Zero (0) is returned for values (such as 90 and 270 degrees) for which this function is undefined.

Example:

Calculate the tangent of analog point "an3" in PCU "PCU3"

```
set [an, PCU3, an3] = TAN [an, PCU3, an3]
```

RSIN

Calculate trigonometric sin (argument in radians)

Example:

Calculate the sin of analog point "an3" in PCU "PCU3"

```
set [an, PCU3, an3] = RSIN [an, PCU3, an3]
```

RCOS

Calculate trigonometric cosine (argument in radians)

Example:

Calculate the cosin of analog point "an3" in PCU "PCU3"

```
set [an, PCU3, an3] = RCOS [an, PCU3, an3]
```

RTAN

Calculate trigonometric tangent (argument in radians).

Please Note: Zero (0) is returned for values (such as pi/2 and 3pi/2 radians) for which this function is undefined.

Example:

Calculate the tangent of analog point "an3" in PCU "PCU3"

```
set [an, PCU3, an3] = RTAN [an, PCU3, an3]
```



4.1.5 CSL Calculation functions

The CSL calc command provides, essentially, a C language subroutine call to a predetermined set of C functions. In general these functions, written in the C programming language, provide a mechanism for complex mathematical calculations in CSL. Each named calc function has a set of named arguments that must be present in the calc invocation. Each named argument must be followed by an equal sign (=) and either a constant or a RealFlex datapoint reference. The end of the argument list is signaled by the reserved word endcalc which is immediately followed by the RealFlex datapoint or CSL float variable that will store the calculation answer.

The answer for the calculation is placed in the CSL variable or passed to RealFlex via the raw data queue in the same way as a CSL set command (i.e., as if being reported from a field device). If the calc function name or any of calc arguments are not found, or the argument is inappropriately assigned, it is a compile time error. The order of the calc arguments between calc and endcalc is not important. The line continuation symbol \ is not required between the calc and the endcalc commands. Character case is ignored in the calc call.

SYNTAX

```
calc Name
  arg1 =
  arg2 =
  ...
  argn =
```

```
endcalc
```

Examples:

```
calc nx19
  Pressure = [an, pcu, pressure]
  Temperature = [an, pcu, temperature]
  Gravity = 0.6
  Co2 = [an, pcu, co2]
  Nitrogen = [an, pcu, nitrogen]
```

```
endcalc [an, system, nx19]
```

```
float press
float f_supercomp
set press = [an, pcu, pressure]*1.234
calc nx19
  Pressure = press
  Temperature = [an, pcu, temperature]
  Gravity = 0.6
  Co2 = [an, pcu, co2]
  Nitrogen = [an, pcu, nitrogen]
```

```
endcalc f_supercomp
```

4.1.5.1 The History Calculation Function

The CSL processor gives users access to the RealFlex historical database via a predefined "calc function" called "history". This function, called via the CSL CALC statement is used to make read-only requests similar to requests accepted by the RealFlex Report Generator. The following historical read requests are supported:

```
ANALOG READ_EXACT, READ_AVG, READ_MIN, READ_MAX
METER READ_NET, READ_HOUR, READ_DAY, READ_MON, READ_YEAR
```



STATUS READ_EXACT

When the CSL processor processes one of these requests, it checks its contents against the contents of the latest historical request. If the tag, read type, time and span are the same, it does not read the database again. It does update the answer datapoint or float variable with the value from the last historical read. Accessing the historical database can be very time consuming, so care should be taken to minimize use of this feature, and to carefully group the commands to avoid unnecessary duplication.

Please Note: The span parameter in the historical calculation function is subject to the following constraints:

1. Span can range from a minimum of 1 second to a maximum of 31622400 seconds (equivalent to 366 days or 1 leap year).
2. Span is maintained as a double precision floating point value and is subject to normal float rounding at higher values.
3. The RealFlex historical data processor retrieval algorithm is designed to first read the start time, convert this time to seconds and then report the historical reading at that time. This design makes assigning span value inconsequential for the following calculation types:

```
ANALOG = READ_EXACT
METER = "all" READ TYPES
STATUS = READ_EXACT
```

In the cases listed directly above, a span value of zero is automatically forced to 1. Setting the span to 1 (or a value greater than 1) will return an exact value. The above read types will return the value which is received at the designated start time.

4. The RealFlex historical retrieval algorithm uses a specified span time for the following historical calculations:

ANALOG - READ_AVG, READ_MIN, READ_MAX

In these read types the span time will be the "slice" of data in seconds which we want to pull out of the historical database (the amount of time which we want historical retrieval to cover, i.e., span=3600 will return a "slice" of data which is one hour long). The day, month, hour, minute and second fields will be the start time for the requested span interval of historical retrieval.

SYNTAX

calc history

```
tag = A database identifier
type = Must be one of the requests listed above
span = A constant, database identifier,
      or arithmetic expression
day = A constant, database identifier,
     or arithmetic expression
month = A constant, database identifier,
      or arithmetic expression
hour = A constant, database identifier,
      or arithmetic expression
minute = A constant, database identifier,
       or arithmetic expression
second = A constant, database identifier,
        or arithmetic expression
```

endcalc -> [database identifier] or float variable for the result.



Example 1

```

csl test
  calc history
    tag = [an, pipeline, an1]
    type = READ_EXACT
    span = 1
    day = 1 (i.e. month_day)
    month = month
    hour = hour - 1
    minute = 0
    second = 0

    endcalc [an, test, a01]
  wait 3
endcsl

```

Note that setting the span time here is inconsequential, this CSL will return the exact value of the analog point at the beginning of the last hour. Please also note that "month_day", "month" and "hour" used to the right of the "=" sign in these examples are CSL keywords for the current month and current hour, respectively.

Example 2

```

csl test
  float histdata
  calc history
    tag = [an, pipeline, an1]
    type = READ_AVG
    span = 3600
    day = month_day
    month = month
    hour = hour - 1
    minute = 0
    second = 0

    endcalc histdata
    if (histdata > 100)
      ....
    endif
  wait 3

endcsl

```

In the above example, we have created the variable histdata and requested that the last hour's average of the analog point be placed into this variable. The span of 3600 seconds pulls an average of all the data archived over the last hour. The day, month, hour, minute and second fields sets the start time to the present day, present month, the previous hour, the "0" minute and "0" second, respectively.

4.1.5.2 AGA3 Calculation Functions

Each component of the American Gas Association AGA3 calculation is a CSL calc command. All AGA3 and component calculations were taken from the "Orifice Metering of Natural Gas and Other Related Hydrocarbon Fluids." American Gas Association Report No. 3, ANSI/API 2530, API 2530, GPA 8185-85.



Several of these calculations have arithmetic expressions raised to a very large power or to a fractional power that requires the use of the C math library call, $\text{pow}(x, y)$; where x must be greater than 0. As a result, in most cases if one or more calc arguments is equal to or less than 0 the calculation result returns 0.

CALC:

nx19 - Supercompressibility Factor (also Fpv)

ARGUMENTS:

pressure flowing pressure (Pf)
 $0.000 \leq x \leq 5000.0$
 temperature flowing temperature (TF)
 $-40.000 \leq x \leq 239.0$
 gravity specific gravity (g)
 $0.554 \leq x \leq 1.0$
 co2 mole percent carbon dioxide (Mc)
 $0.000 < x \leq 15.0$
 nitrogen mole percent nitrogen (Mn)
 $0.000 < x \leq 15.0$

CALC:

aga3_fr - Reynolds Number Factor

ARGUMENTS:

tap flange or pipe tap type
 0 is flange
 1 is pipe
 tube inside meter diameter (D)
 $0.000 < x \leq 30.0$
 orifice orifice diameter (d)
 $0.000 < x \leq 21.5$
 pressure flowing pressure (Pf)
 $0.000 \leq x \leq 5000.0$
 difpres differential pressure (hw)
 $0.000 \leq x \leq 5000.0$

CALC:

aga3_y - Expansion Factor

ARGUMENTS:

location upstream or downstream tap location
 0 is upstream
 1 is downstream
 tap flange or pipe tap type
 0 is flange
 1 is pipe
 difpres differential pressure (hw)
 $0.000 \leq x \leq 5000.0$
 pressure flowing pressure (Pf)
 $0.000 \leq x \leq 5000.0$
 tube inside meter diameter (D)
 $0.000 < x \leq 30.0$
 orifice orifice diameter (d)
 $0.000 < x \leq 21.5$



CALC:

aga3_fb - Basic Orifice Factor

ARGUMENTS:

tap flange or pipe tap
0 is flange
1 is pipe
tube inside meter diameter (D)
 $0.000 < x \leq 30.0$
orifice orifice diameter (d)
 $0.000 < x \leq 21.5$

CALC:

aga3_fpb - Pressure Base Factor

ARGUMENTS:

pressure contract base pressure (Pb)
 $14.4 \leq x \leq 16.70$

CALC:

aga3_ftb - Temperature Base Factor

ARGUMENTS:

temperature contract base temperature (Tb)
 $40.0 \leq x \leq 90.0$

CALC:

aga3_ftf - Flowing temperature Factor

ARGUMENTS:

temperature flowing temperature (Tf)
 $-20.0 \leq x \leq 150.0$

CALC:

aga3_fgr - Real Gas Relative Density Factor

ARGUMENTS:

gravity specific gravity (g)
 $0.550 \leq x \leq 1.0$

CALC:

aga3 - Volume Flow Rate of Gases in Cubic Feet per
Hour at Base Conditions:
 $Q_v = C_{prime} * \text{sqrt}(H_w(P_{fl} + P_{atm}))$



ARGUMENTS:

cprime Fb*Fr*Y*Fpb*Ftb*Ftf*Fgr*nx19
difpres differential pressure (hw)
 0.000 <= x <= 5000.0
pressure flowing pressure (Pf)
 0.000 <= x <= 5000.0
atmpres atmospheric pressure (Patm)
 0.000 < x <= 20.0



4.2 Superkey files

The Superkey processor allows definition of control sequences that may be initiated by clicking (with the mouse or trackball) on on-screen buttons (Superkeys). These Superkeys are defined and placed in Displays using the Flex.Builder application program.

The Superkey control sequences are written in a simple language and placed, either singly or in groups, in QNX text file(s). Within these files, the command sequences are further organized into "procedures". Clicking on a single Superkey starts execution of a corresponding procedure. Superkey procedure files are text files named according to the standard QNX system convention. They are created and edited by the system implementor, with the Configuration File Editor application or other editing software packages that are compatible with the QNX Operating System and that produce plain ASCII format files (i.e., formatting characteristics cannot be stored in the files).

4.2.1.1 Superkey Procedure Files

Individual Superkey procedure execution is started when the operator clicks on a Superkey button. The Superkey processor accesses the file associated with the Superkey and finds within the file the procedure whose name is part of the Superkey's definition and executes the procedure. More than one Superkey can execute at the same time. All Superkey procedure files should reside in directory **/realflex/data**, or a subdirectory thereof.

Note to LanFlex Users: If a Superkey procedure is developed on a network node, the procedure will only be available on the network node where it was created. To make the procedure available to the entire network, develop the Superkey procedure on the prime node or copy the procedure from the network node to the prime node after development has been completed.

The organization of Superkey procedures in multiple user-named files can simplify the task of keeping track of numerous Superkey functions. All Superkey procedures of a given type can be kept in a single text file, the name of which reflects the type of procedure included. Each Superkey procedure is uniquely identified by a user defined name that can contain up to 12 characters (alphanumeric characters plus the dash (-), the underscore (_) and the period (.)). The procedures should be entered into the file in a sequential format, adhering to the general format outlined below. Comments can be placed in Superkey procedures, provided they are placed after semicolons. (i.e., that part of any line that follows a ";" is treated as a comment and is ignored by the Superkey processor.)

For example:

```
superkey name1
  Statements defining Superkey procedure "name1".
endkey
```

```
superkey name2
  Statements defining Superkey procedure "name2".
endkey
```

```
superkey name3
  Statements defining Superkey procedure "name3".
endkey
```

Please note: A carriage return (blank line) must be placed at the end of any file edited with the QNX text editor (vedit).



4.2.1.2 Testing Superkey Logic

The Superkey processor may be started from the command line using the [-m] option while in the directory **/realflex/bin**. The examples later in this section show how to start the Superkey processor from the command line and execute a specific procedure. If no Superkey filename is provided, the Superkey process will search for the default Superkey procedure file **superkey.prc**, in the **/realflex/data** directory. The file's complete pathname must be specified if the file is not in the **/realflex/data** directory.

When specifying the filename in the Flex.Builder, the name must be specified relative to the **/realflex/data** directory. For example, if the complete pathname is **/realflex/data/skeys/shut_downs**, the file would be specified in the Flex.Builder as **"skeys/shut_downs"**. A maximum of 16 characters may be used for this specification.

Because Superkey procedures access the RealFlex database, RealFlex must be running when the Superkey processor is started.

To step through the logic, run the superkey file with both a [-m] and [-d] option appended to the run statement. This will debug the file and provide feedback in verbose mode. When you have completed the debug, try using your superkey.



Warning: If your superkey sends controls to the field (i.e., opening a valve, starting pumps, etc.) you may want to be careful testing your superkey.

If your superkey fails, you will need to check that the superkey file name and procedure name in the Configuration File Editor screen matches those you have specified in your superkey procedure.

Example 1

To view the options for the Superkey processor, open a QNX shell, change to the **/realflex/bin** subdirectory and enter:

```
/realflex/bin> Csuperkey -?

-s Superkey procedure name
-f Filename of the superkey procedure file
-r Status control PCU for procedure status points
-e If the selected superkey is started with a -e, the
  superkey procedure will not be logged to the
  Historical Alarm/Event Summary.
-p This parameter will send reports to the default
  printer
-U This parameter will echo updates to the screen
-S Calculation stack size
-t This parameter is used with the PRINT_SCREEN
  command to set a "wait" for printing complex
  displays.
-m This parameter will activate the superkey and print
  informative messages to the screen during the
  execution process.
-d Runtime debugger
-v This will print verbose messages concerning the
  execution of the chosen superkey.
-E This parameter will print extremely verbose
  messages during the execution of the chosen
  superkey.
```



Example 2

Invoking superkey to execute procedure "checklist" in the default superkey procedure file **superkey.prc**:

```
/realflex/bin> Csuperkey -schecklist -m -d
```

Example 3

Invoking superkey from the command line to execute procedure "report_6" in a user specific superkey procedure file (suprky.rpts) that resides in directory **/realflex/data**:

```
/realflex/bin> Csuperkey -sreport_6 -fsuprky.rpts -m -d
```

Example 4

Invoking superkey to execute procedure "reports" in a user specific superkey procedure file (Monday) that resides in directory **/realflex/data/skeys**:

```
/realflex/bin> Csuperkey -sreports -f/realflex/data/skeys /Monday -m -d
```

4.2.1.3 Controlling Superkey Access from the System Summary Screen

When superkeys are created, a line is added in the "SYSTEM" PCU in the System-PCU Summary that will allow controls to be sent to make the superkey active or inactive (ON/OFF). To access the System-PCU Summary, from within Flex.View, select the **Main** menu, select **RealFlex Summaries** and then select "System-PCU". A menu will appear allowing you to choose which PCU to view, select "SYSTEM".

As referenced earlier, if your superkey is named "skeys/shut_downs" in the Flex.Builder, you will be able to view that name on the "SYSTEM" PCU. If you click on the control field for that superkey, you will be given the option to turn the superkey "ON" or "OFF". If you select "OFF", the superkey will not perform its specified control when selected in the Display that it has been placed in. To reactivate the superkey, from within Flex.View, select the **Main** menu, select **RealFlex Summaries**, select "System-PCU", click on the superkey value and click on the "Yes" button.

Keep in mind that if you disable a superkey from the System-PCU Summary screen that the superkey may still be activated by accessing it through a command line argument.

4.2.1.4 Superkey Status Records

The first time a Superkey procedure is executed, a superkey status record with the same name is created in the System-PCU Summary PCU "SYSTEM". This record can be set to "OFF" to disable execution of the Superkey. (It would be possible to create a Superkey procedure that would enable, disable, or toggle other Superkeys).

Example:

Superkey "toggle" would toggle Superkeys "pump_on" and "pump_off" between being operable (ON) and being inoperable (OFF). A dynamic display of the database point "pump_on" in pcu "system" could be used to indicate whether the superkeys were ON or OFF.

```
superkey pump_on
  send [st, tank_farm, pump_1] = on
endkey
;
superkey pump_off
  send [st, tank_farm, pump_1] = off
```



```

endkey
;
superkey toggle
  if [st, system, pump_on] = on
    stuff [st, system, pump_on] = off
    stuff [st, system, pump_off] = off
  else
    stuff [st, system, pump_on] = on
    stuff [st, system, pump_off] = on
  endif
endkey

```

4.2.2 Superkey Procedures

Creating Superkeys involves installing on-screen buttons in Displays and building and testing Superkey procedure files. This section goes into more detail about both of those activities.

4.2.2.1 Superkey Language Statement Syntax

Superkey statements may start at any column position on the line. Using a standard style of indentation is not required but is recommended to enhance readability of the Superkey procedures. Semicolons (;) are used to delimit comments. If there is a semicolon on a line, anything to the right of it is considered to be a comment, and is ignored by the Superkey processor (including entire lines with the semicolon in column 1). Plenty of carefully worded comments promotes readability and facilitates maintenance of the Superkey procedures.

Some expressions require parentheses () or square brackets []. These are further described in the following subsections.

When the Superkey processor is started, the Superkey procedure specified, in the specified (or default) procedure file, is compiled against the allowable Superkey syntax for database references, commands and operators. In the event of a compilation error, the Superkey task prints the line number where the error occurred and aborts. If no errors are encountered, the Superkey task begins immediate execution of the specified Superkey procedure.

Although Superkey statements may be up to 200 characters long, Superkey statements that extend past 80 columns or that need to be continued on the next line to enhance readability may be split using the backslash operator "\".

Example:

```

[an, total_temp, device_1] = \
[an, temp, device_2] + \
[an, temp, device_3]

```

4.2.2.2 Referencing the Real-Time Database

Individual scan points may be referenced as follows:

[DD, PCU, TAG] FIELD, where:

DD = 2 character mnemonic for the database type.
Possible values are:
"rt" PCU database record
"an" Analog database record (input and output)
"mt" Meter database record
"st" Status database record



"tk" Tank database record
PCU = 12 character name of associated PCU

TAG = 12 character name for the scanpoint in the
RealFlex database. (Not applicable when DD="rt".)

FIELD = A character mnemonic or identifier for a
particular field within a scanpoint type.
The possibilities are as follows:

If database type is PCU (i.e., DD = "rt")

ACTIVE = PCU active state (default)
CHAN = PCU control queue channel
ADDR = PCU physical address
TSCANS = PCU total scans
VSCANS = PCU valid scans
RETRIES = PCU scan retries
NO RESP = PCU no response
ERRORS = PCU data errors
COMMEFF = PCU communications efficiency

If database type is ANALOG (i.e., DD="an")

CVAL = Current EU value (default)
HI = The high alarm threshold
HIHI = The high high alarm threshold
LO = The low alarm threshold
LOLO = The low low alarm threshold
MAX_EU = The maximum EU value
MIN_EU = The minimum EU value

If database type is METER (i.e., DD = "mt")

CVAL = current net accumulation (default)
LHOUR = last hour's accumulation
HOUR = current hourly accumulation
DAY = current daily accumulation
MONTH = current monthly accumulation
YEAR = current yearly accumulation
GROSS = current gross accumulation
FACTOR = meter multiplication factor
YESTERDAY = yesterday's accumulation
LAST_GOOD = last good meter reading from device

If database type is STATUS (i.e., DD = "st")

CVAL = current value (default)

If database type is TANK (i.e., DD = "tk")

PVOL = product volume (default)
AVOL = available volume
TEMP = temperature
PROD = product code
FEET = tank level in feet
INCH = tank level in inches
FRAC = tank level in fractions (8ths or 16ths)
GRAV = product specific gravity



The RealFlex realtime system must be operating before the Superkey processor is started, so that any database references may be resolved. If the Superkey processor fails to find a referenced PCU/Tag name combination in the RealFlex database, the Superkey processor will abort. Therefore, all referenced PCUs and datapoints (Tag names) must be entered into the database using the database editor before the Superkey processor is started. Also, any invalid database type or field specifications found in the Superkey file will cause the processor to abort.

4.2.2.3 RealFlex Internal Flags Accessible by Superkey Procedure

In addition to the external, user definable database types, the Superkey task also may be used to manipulate the internal RealFlex control flags associated with each defined scan point in the system. These flags provide the user with a means to perform manual overwrite of a scan point, take a scan point out of scan processing, and to monitor the PCU communications status of the last message sent or received.

Please note: The alarm flags: lo_alarm, hi_alarm, lolo_alarm, hihi_alarm, un_ack and alarm, are "read" or "monitor only" flags. Superkeys cannot be used to directly clear or set the alarm flags.

Accessible Internal Flags:

NO_REPLY PCU no reply flag

INVALID PCU invalid reply flag for all database point types

INSTR_FAIL Instrument failure (over/under range) flag for analog datapoints

COLD_START RealFlex coldstart flag (i.e., first time data used)

OVER_WRITE Scan point manual overwrite flag

LO_ALARM Low alarm flag for analog and tank points

HI_ALARM High alarm flag for analog and tank points

LOLO_ALARM Low low alarm flag for analog and tank points

HIHI_ALARM High High alarm flag for analog and tank points

UN_ACK ALARM unacknowledged flag for all database point types

ALARM Global alarm flag for all database point types

CONTROL Control pending flag

CTRL_TAGGED Control tagged flag for status points

INFO_TAGGED Information tagged flag for status, analog, meter and tank points



4.2.3 Superkey Keywords and Statements

Keywords used in Superkey statements to represent awaited/controlled states are the same as those defined for the scan point during the database build process. Any of the six letter mnemonics specified for a status point in the database configuration process may be used in Superkey statements. Currently, RealFlex supports up to four control outputs for an associated status point. This is important to remember during the database definition process.

In addition, Superkey translates the following keywords into constant values as an aid to Superkey processing:

SK_OFF - Used as a symbolic constant for the value 0.

SK_ON - Used as a symbolic constant for the value 1.

Example:

```
if ([an, system, cancel] over_write = sk_on)
stuff [an, system, cancel] over_write = sk_off
```

4.2.3.1 Statements

The Superkey language is a subset of CSL language and allows users to write logic to perform monitoring and controlling of individual devices based on events or values of points. Programming experience is not required to use the Superkey language, however, the user must have a basic understanding of the task(s) to be performed. The following example is provided to give an idea of how the Superkey language works:

Statement of what the procedure (named "control_1") is designed to do:

```
If status point "st1" of PCU "PCU23" is ON,
    send a control output of "10" to analog point "an14" of "PCU23",
else
    send a control output of "25" to analog point "an14" of "PCU23".
```

Procedure written in Superkey language:

```
superkey control_1
if ([st, pcu23, st1] = on)
    SEND [an, pcu23, an14] = 10
else
    SEND [an, pcu23, an14] = 25
endif
endkey
```

This example is obviously very simple, and control logic can be very complex; however, the same principles apply regardless of the complexity of a procedure. The Superkey processor has been designed to help deal with the complexity of control logic by making the language as close as possible to natural language while providing ways of precisely specifying the steps, database points, values, etc., that are involved.

Superkey Statements:

```
CONTINUE WHEN (expression) RECIPE recipe_name
DISPLAY REMOVE_DISPLAY
DSAVE database_type REPORT report_type
DSTUFF a = b RETURN
ENDKEY SEND a = b
EXIT SET a = b
GOTO n SHELL QNX_command
```



GOTO [an, PCU, TAG] n1 n2..n9 STEP n
 IF with ENDIF STUFF a = b
 IF with ELSE and ENDIF SUPERKEY proc_name
 OPERATOR text_string WAIT n
 PRINT_SCREEN

(Portions in italics are replaced with user-supplied expressions and values.)

4.2.3.2 Detailed Explanations of Superkey Statement Types

CONTINUE WHEN (*expression*)

Used to hold execution of the Superkey procedure until the expression is evaluated as TRUE. After that time, execution proceeds. Use parenthesis to specify and/or clarify order of evaluation in a compound expression.

DISPLAY *format_name* {*node_number*}

Used to automatically display a screen format on the specified node. If node number is not specified, the current node is assumed. If you are not sure what node number is, type in the command "sin net" at the QNX prompt and it will return node number.



WARNING: Calling up multiple displays may exhaust available RAM. If this occurs, you should begin closing windows in order to ease the strain on memory. (For automatic removal of displays, see the description of the REMOVE_DISPLAY command.)

DSAVE *database_type*

Force file-based save on entire database type to hard disk. The RealFlex database types that can be saved by this command are PCU, ANALOG, METER, STATUS, TANK and graphic Displays. There will be a delay of up to 30 seconds after this command is executed before write to disk begins.

Example:

```
superkey save
  dsave analog
  dsave meter
  dsave status
endkey
```

This example will force a file based save of all analog, meter and status points in the database on warmstart. These are the values which will come up before information is received from the field.

DSTUFF a = b

Used to "stuff" the value of expression "b" directly into the RealFlex record identified by "a", bypassing any normal RealFlex data processing and to save the individual record to hard disk. The hard disk save versus RAM-only save is the difference between the DSTUFF and STUFF commands.

Essentially a manual overwrite facility. As with DSAVE, there will be a delay of up to 30 seconds after this command is executed before write to disk begins. DSTUFF syntax sets the specified database point on the left side of the "=" to the result of the expression on the right side of the "=". Please note that placing a DSTUFF command in a tight logic loop can place undue stress on CPU performance on your hard disk.

Example:

```
DSTUFF [an, system, an4] = [an, pcu6, an3] + 15.6
```



ENDKEY

Used to mark the end of the procedure. Each Superkey procedure must end with an ENDKEY statement.

EXIT

End processing of the current superkey procedure and turn this Superkey procedure off. Subsequent tests for procedure execution will fail until the Superkey is reactivated either via the "SYSTEM" PCU in the System-PCU Summary or via command line argument.

Example:

```
superkey test
  if [st,power,st_1]=off
    exit
  endkey
```

GOTO n

Used to control the order in which statements are executed. Proceed at STEP number "n" of the Superkey procedure. The GOTO statement can only be used to branch forward within a Superkey Procedure. (See STEP statement for examples).

GOTO [an, PCU, TAG] n1 n2 .. n9

Performs a GOTO based on the value found in the referenced analog database point. This command compares the value found in the analog database point with each of the numbers n1 n2 .. n9. If the analog value matches one of these numbers then execution continues at the STEP statement with the same number. If no match is found, execution continues at the STEP statement with the first number in the list (n1). The list may contain up to nine numbers, all of which should appear in STEP statements. The GOTO statement can only be used to branch forward within a Superkey Procedure. (See STEP statement for examples). It is strongly recommended that users execute a WAIT between STEP and GOTO commands whenever infinite loop processing is a possibility.

Example:

```
GOTO [an, pcu15, an16] 99 5 10 12 15
STEP 5
  ;Statements to execute if point an16 equals 5
  GOTO 99
STEP 10
  ;Statements to execute if point an16 equals 10
  GOTO 99
STEP 12
  ;Statements to execute if point an16 equals 12
  GOTO 99
STEP 15
  ;Statements to execute if point an16 equals 15
STEP 99
  ;Skip to here if value of an16 not on list
```

IF with ENDIF

Use this statement to control logic flow within a Superkey procedure. Every IF must have a corresponding ENDIF to mark the boundary of the statement(s) to be executed if the expression is TRUE. Simple IF expressions do not need parentheses. Compound IF statements, those IFs with more than one check, need to be grouped by parenthesis.



Example:

```
if ([an, PCU_3, an_pt_2] > 95.5)
  Statement(s) to be executed if value exceeds 95.5
endif
```

Superkey supports nested IF-ENDIF and IF-ELSE-ENDIF statements to three levels.

IF with ELSE and ENDIF

This statement also controls logic flow within a Superkey procedure, but provides an optional ELSE to identify statement(s) to be executed if the expression is found to be FALSE.

Example:

```
if (expression)
  Statement(s) to execute if expression is true
else
  Statement(s) to execute if expression is false
endif
```

Superkey supports nested IF-ENDIF and IF-ELSE-ENDIF statements to three levels.

OPERATOR *text_string*
NOOPERATOR *text_string*
POPERATOR *text_string*

This statement is used to ask the RealFlex operator yes/no and numerical-input questions. When an OPERATOR statement is executed, a pop-up window appears on the screen and displays the specified *text_string* along with YES and NO on-screen buttons and a data entry field. (Everything in the line after the word "operator" is taken as the text string. Quotation marks are not required).

The operator's "yes" or "no" response, if any, is placed in variable "reply" (if only a numerical value is entered, "reply" is set to "no"). The operator's numerical input, if any, is placed in variable "nreply" (if none, "nreply" is set to 0). If no response is made before the timeout interval elapses, the variable reply is set to "timeout", and the variable nreply is set to zero (0). The default timeout interval is 20 seconds, but the timeout interval can be modified by placing a WAIT statement immediately after the OPERATOR statement.

Example:

```
operator enter value or select YES to use 10.
-> wait 5 ( if wait is 0 then a numeric value can be entered )
if (reply = timeout)
endkey
```

Example for numeric input:

```
noperator enter value.
wait 5
if (reply = timeout)
endkey
```

Example for numeric input, no echo of numbers on screen:

```
poperator enter value.
wait 5
if (reply = timeout)
endkey
```

In this example, the pop-up window will display the message "Enter value or select YES to use 10.", and the operator will be given 5 seconds to respond. When an on-screen button is selected, when the key is pressed, or after 5 seconds, the pop-up window will disappear.

It is not necessary to test for replies for which nothing is to be done.



Example:

```
OPERATOR Ready to start?
if (REPLY = YES)
    Statements to be executed if answer is YES
endif
```

Notice that in this example, the window will disappear after 20 seconds.

PRINT_SCREEN *format_name* {*node_number*} {-L}

Displays the requested graphics format and then prints the screen. In networked systems you may specify a node number for format display, absence of a node number assumes the current node. (If you are not sure what the node number is, type in the command "sin net" at the QNX shell prompt and it will return the node number). The -L (or -l) option forces the printer to landscape mode; otherwise, portrait mode is the print default.

In order to completely display historical trend displays, a delay greater than 5 seconds may be required. This is defined as a command line argument for Superkey. In order to determine the amount of time to wait, display the trend via Display and use this as a reference. This time-out value will only affect PRINT_SCREEN commands.



WARNING: Calling up multiple displays may exhaust available RAM. If this occurs, you should begin closing windows in order to ease the strain on memory. (For removal of displays, see the description of the REMOVE_DISPLAY command).

RECIPE *recipe_name*

Superkey interface to the optional RealFlex Recipe Loader package. Loads the requested recipe. (See the Recipe Loader documentation for details).

REMOVE_DISPLAY *format_name* {*node_number*}

Removes the named display brought up via DISPLAY or PRINT_SCREEN command on the specified node. If node number is omitted, the current node is assumed.

REPORT *report_type* [*options*]

Prints the requested report, either one of the RealFlex predefined reports or a report defined by the report generator. Allowable report type values are SYSTEM [all or PCU name], COMM, ACTIVE, HISTORY, METER or RG {-p -d -R -r -s} [report file name] for any report created by the user.

Examples:

```
REPORT comm
REPORT system PCU5
report RG -p -d -r0 my_rpt_file
```

The command line parameters specified when using the RG command are pulled from **Crg_exec** which is used to print reports generated by the Report Generator. To view the command line arguments for **Crg_exec**, open a shell, change to the **/realflex/bin** subdirectory and enter the command "Crg_exec -?".

RETURN

Used to terminate the Superkey tasks (and return to the QNX operating system prompt if running in the foreground).





WARNING: When a RETURN statement is executed, the Superkey task is terminated, not just the Superkey procedure containing the RETURN statement. This means that all Superkey procedures in the same procedure file will terminate, as well. If you wish to terminate only the Superkey procedure, use the EXIT statement. The RETURN statement should be included in any procedure invoked on a scheduled basis via an entry in the RealFlex rptcron file or via the printing of a user defined report.

Example:

```
superkey test
  if [st,power,st_1]=off
  return
endkey
```

This example will leave the currently running procedure and return the pointer to the beginning of the code.

SEND a = b

Used to send control commands to the application system's data acquisition units. SEND syntax directs to the PCU specified via the database point on the left side of the "=" the result of the expression on the right side of the "=";

Example: Send the value 10 to analog output point an14 of pcu23.
SEND [an, pcu23, an14] = 10

SET a = b

Used to enter calculated data into the RealFlex system through the normal RealFlex data processors and alarm checks. SET syntax directs to the specified "pseudo" database point on left side of the "=" the result of the expression on the right side of the "=". The SET command should only be used to place values in pseudo (unscanned) database points.

Example:

```
SET [an, system, an14_adj] = [an, pcu23, an14] * 10
```

SHELL QNX_command

Superkey interface to the QNX operating system. This Superkey command issues the given QNX command to the operating system. This allows the user access to all the QNX commands found in the QNX /bin32 directory, such as slay, which, etc. It also allows the user access to user defined QNX shell files that have the "executable" attribute set. No error checking is done on the QNX command. If the QNX command is entered incorrectly in the Superkey file then the QNX operating system will issue the appropriate error message, possibly "command not found", when the Superkey procedure requests the QNX command.

Example: to execute the **user_function** QNX shell file:
shell user_function

STEP n

Used to define a label for use by GOTO statement(s) when logic flow must be transferred from one location to another, later in the procedure. (A STEP statement may not be placed between an IF statement and the corresponding ENDIF statement).



Example:

```
GOTO [an, pcu15, an16] 99 5 10 12 15
STEP 5
  Statements to execute if point an16 equals 5
  GOTO 99
STEP 10
  Statements to execute if point an16 equals 10
  GOTO 99
STEP 12
  Statements to execute if point an16 equals 12
  GOTO 99
STEP 15
  Statements to execute if point an16 equals 15
STEP 99
  ; Skip to here if value of an16 not on list
```

In this example, if the value of analog point "an16" of PCU "pcu15" in the RealFlex database is 10, the statements immediately after the "STEP 10" statements will be executed, down to the "GOTO 99" statement, then processing will skip to the "STEP 99" statement. If the value of the point is not 5, 10, 12, or 15, processing will skip directly from the first GOTO statement to the "STEP 99" statement.

STUFF *a = b*

Used to "stuff" data directly into the RealFlex system, bypassing any normal RealFlex data processing and alarm checking. This command should only be used to adjust state/value of pseudo database records or float variables. Essentially a manual overwrite facility. STUFF syntax forces the specified database point on left side of the "=" to the result of the expression on the right side of the "=";

Example:

```
STUFF [an,system,total]=[an,pcu15,an31]+[an,pcu15,an32]
```

SUPERKEY *proc_name*

Must be the first statement of each Superkey procedure. The procedure name (replaces "proc_name") must be unique within the procedure file, but may be duplicated in other files.

WAIT *n*

Execute a delay of "n" seconds before resuming execution of the procedure at the next statement. The WAIT statement has a maximum value of 9,999,999 seconds (roughly 155 days). The WAIT statement has a special use in conjunction with the OPERATOR statement (described previously).



4.2.4 Superkey Operators

Superkey provides limited logical and mathematical processing through the use of reserved operators. Use parentheses to develop compound mathematical statements. Operators provided in the Superkey language are listed below with detailed explanations presented immediately following.

"=" equal (or set equal to)
 "<>" not equal to
 ">" greater than
 ">=" greater than or equal to
 "<" less than
 "<=" less than or equal to
 OR logical "or"
 AND logical "and"
 "+" addition
 "-" subtraction
 "*" multiplication
 "/" division
 "%" modulus arithmetic
 "^" exponentiation (including ^2 for squaring)
 SQRT square root
 LOG natural log
 ABS absolute value (integer)
 FABS absolute value (float)
 SIN trigonometric sin (degrees)
 COS trigonometric cosin (degrees)
 TAN trigonometric tangent (degrees)
 RSIN trigonometric sin (radians)
 RCOS trigonometric cosin (radians)
 RTAN trigonometric tangent (radians)

4.2.4.1 Detailed Explanations of Superkey Operators

"="

Evaluate or set expression equal

Examples:

1. Compare the value of status point "amtasks" in PCU "SYSTEM" with the user defined state mnemonic "start".

if ([st, system, amtasks] = start)

2. Set analog point "an1" in PCU "SYSTEM" equal to a numeric constant.

set [an, system, an1] = 1

"<>"

Evaluate expression not equal

Example:

Evaluate status point "st1" in PCU "SYSTEM" against user defined state mnemonic "start".

if ([st, system, st1] <> start)



">"

Evaluate expression or constant greater than

Examples:

1. Evaluate analog point "an1" in PCU "PCU1" against numeric constant

```
if ([an, PCU1, an1] > 10)
```

2. Compare two analog points

```
if ([an, pcu1, an1] > [an, pcu2, an1])
```

">="

Evaluate expression or constant greater than or equal to

Example:

Evaluate analog point "an1" in PCU "PCU1" against numeric constant

```
if ([an, PCU1, an1] >= 10)
```

"<"

Evaluate expression or constant less than

Example:

Evaluate analog point "an1" in PCU "PCU1" against a numeric constant.

```
if ([an, PCU1, an1] < 10)
```

"<="

Evaluate expression or constant less than or equal to

Example:

Evaluate analog point "an1" in PCU "PCU1" against a numeric constant.

```
if ([an, PCU1, an1] <= 10)
```

OR

Evaluate two expressions and perform conditional instructions if either expression is true.

Example:

Evaluate analog points "an1" and "an2" in PCU "PCU1" against numeric constants

```
if (([an,PCU1,an1] < 1) or ([an,PCU1,an2] < 99))
```



AND

Evaluate two expressions and perform conditional instructions only if both expressions are true.

Example:

Evaluate analog points "an1" and "an2" in PCU "PCU1" against numeric constants

if (([an, PCU1, an1] < 1) and ([an, PCU1, an2] < 99))

"+"

Perform arithmetic addition operation

Example:

Perform addition operation on analog point "an1" in PCU "PCU1"

set [an, PCU1, an1] = [an, PCU1, an1] + 100

"-"

Perform arithmetic subtraction operation

Example:

Perform subtraction operation on analog point "an1" in PCU "PCU1"

set [an, PCU1, an1] = [an, PCU1, an1] - 100

"*"

Perform multiplication operation

Example:

Perform multiplication operation on analog point "an1" in PCU "PCU1"

set [an, PCU1, an1] = [an, PCU1, an1] * 100

"/"

Perform division operation

Example:

Perform division operation on analog point "an1" in PCU "PCU1"

set [an, PCU1, an1] = [an, PCU1, an1] / 100

"%"

Perform modulus arithmetic

Example:

Perform modulus 10 arithmetic on analog point "an3" in PCU "PCU2"

set [an, PCU2, an3] = [an, PCU2, an3] % 10



"^"

Perform exponentiation operation (use "^2" to square a value)

Example:

Perform exponentiation operation of cube on analog point "an3" in PCU "PCU2"

```
set [an, PCU2, an3] = [an, PCU2, an3] ^ 3
```

SQRT

Perform square root operation

Example:

Perform operation of square root on analog point "an3 " in PCU "PCU3"

```
set [an, PCU3, an3] = SQRT [an, PCU3, an3]
```

LOG

Perform natural log operation

Example:

Perform operation of natural log on analog point "an3" in PCU "PCU3"

```
set [an, PCU3, an3] = LOG [an, PCU3, an3]
```

ABS

Perform absolute value operation (integer argument)

Example:

Perform operation of absolute value on the value -5 and put the result (5) in analog point "an3" in PCU "PCU3"

```
set [an, PCU3, an3] = ABS(-5)
```

FABS

Perform absolute value operation (floating point argument)

Example:

Perform operation of absolute value on analog point "an3" in PCU "PCU3" (if the point contains (for example) -5.2, it will be changed to 5.2)

```
set [an, PCU3, an3] = FABS [an, PCU3, an3]
```



SIN

Calculate trigonometric sin (argument in degrees)

Example:

Calculate the sin of analog point "an3" in PCU "PCU3"

```
set [an, PCU3, an3] = SIN [an, PCU3, an3]
```

COS

Calculate trigonometric cosine (argument in degrees)

Example:

Calculate the cosin of analog point "an3" in PCU "PCU3"

```
set [an, PCU3, an3] = COS [an, PCU3, an3]
```

TAN

Calculate trigonometric tangent (argument in degrees) Note: Zero (0) is returned for values (such as 90 and 270 degrees) for which this function is undefined.

Example:

Calculate the tangent of analog point "an3" in PCU "PCU3"

```
set [an, PCU3, an3] = TAN [an, PCU3, an3]
```

RSIN

Calculate trigonometric sin (argument in radians)

Example:

Calculate the sin of analog point "an3" in PCU "PCU3"

```
set [an, PCU3, an3] = RSIN [an, PCU3, an3]
```

RCOS

Calculate trigonometric cosine (argument in radians)

Example:

Calculate the cosin of analog point "an3" in PCU "PCU3"

```
set [an, PCU3, an3] = RCOS [an, PCU3, an3]
```

RTAN

Calculate trigonometric tangent (argument in radians).

Please Note: Zero (0) is returned for values (such as pi/2 and 3pi/2 radians) for which this function is undefined.



Example:

Calculate the tangent of analog point "an3" in PCU "PCU3"

```
set [an, PCU3, an3] = RTAN [an, PCU3, an3]
```

4.2.5 Superkey Examples

The following sample Superkey procedure files illustrate how to create Superkey procedures.

Example 1 The following example procedures will work within the demonstration database and graphic Displays included with RealFlex:

```
;This is the default Superkey procedure file.
;It contains a few short demonstration procedures.
;-----
;This procedure asks a yes/no question, and echoes the reply
;-----
superkey yesno
operator Sample Superkey query. Pick yes or no.
wait 5
if (reply=timeout)
operator You waited too long.
goto 99
endif
if (reply=yes)
operator You selected YES.
goto 99
endif
if (reply=no)
operator You selected NO.
endif
step 99
endkey
;
;-----
;This procedure overwrites the status of point "DEMO_STATUS"
;pcu "SYSTEM" with the value "CLOSED" after asking the
;operator if the valve is really closed
;-----
superkey set_closed
operator Is the valve really closed?
if (reply = yes)
stuff [st, system, demo_status] = closed
endif
endkey
;
;-----
;This procedure is the same as the one above, except that
;it sets the value to "OPEN", and doesn't ask first.
;-----
superkey set_open
stuff [st, system, demo_status] = open
endkey
;
;-----
;This procedure adds 25 to a pseudopoint or subtracts 25
;from it.
;-----
```



```

superkey delta
  operator Click YES to add 25, NO to subtract
  if (reply = yes)
    set [an, system, demo_analog] = \
      [an, system, demo_analog] + 25
  endif
  if (reply = no)
    set [an, system, demo_analog] = \
      [an, system, demo_analog] - 25
  endif
endkey
;
;-----
;These procedures add and subtract (respectively) 10
;-----
superkey plus_10
  set [an,system,demo_analog] = [an,system,demo_analog] + 10
endkey
;
superkey minus_10
  set [an,system,demo_analog] = [an,system,demo_analog] - 10
endkey
;
;-----
;These procedures changes all four alarm limits at one time
;-----
superkey adjust
  shell /realflex/bin/showhelp adjustments
endkey
superkey narrow
  stuff [an, system, demo_analog] HIHI = 270
  stuff [an, system, demo_analog] HI = 240
  stuff [an, system, demo_analog] LO = 60
  stuff [an, system, demo_analog] LOLO = 30
endkey
;
superkey wide
  stuff [an, system, demo_analog] HIHI = 285
  stuff [an, system, demo_analog] HI = 275
  stuff [an, system, demo_analog] LO = 25
  stuff [an, system, demo_analog] LOLO = 10
endkey

```

Example 2 Example of how Superkeys can be used to execute a group of "commands" simultaneously:

```

;-----
;Superkey to perform a sequenced shutdown
;-----
superkey shutdown
  operator Perform Pipeline shutdown?
  wait 5
  if (reply = NO)
    goto 99
  endif
; These statements are executed if the operator selects YES,
; or if he doesn't answer at all:
  send [st, pipeline, s15] = closed
  send [st, pipeline, s1] = stop
  send [st, pipeline, s2] = stop

```



```
send [st, pipeline, s3] = stop
send [st, pipeline, s4] = stop
send [st, pipeline, s5] = stop
send [st, pipeline, s6] = stop
send [st, pipeline, s7] = closed
send [st, pipeline, s8] = closed
send [st, pipeline, s9] = closed
send [st, pipeline, s10] = closed
send [st, pipeline, s11] = closed
send [st, pipeline, s12] = closed
send [st, pipeline, s13] = closed
send [st, pipeline, s14] = closed
step 99
endkey
```



4.3 Alarm Display Configuration layout file (alarm_config)

The Alarm Display Configuration layout file "alarm_config" is used for building the alarm messages.

The Active Alarm and Historical Event Summaries (available via the Flex.View **Main** menu, selecting **RealFlex Summaries** and then choosing "Active Alarm" or "Historical Alarm/Event"), Alarm Lines (displayed at the bottom of the RealFlex base window) and printed alarm/event messages may be configured to display data in any way the system implementor chooses.

In the "/realflex/data" directory, create a file with the name **alarm_config**.

This is the layout file that will be used for building the alarm messages. When this file is present it will be used; otherwise, default alarm/event formatting will be used. There is no command line option on any of the processes that will turn this feature on or off.

A separate entry for date/time stamp must be entered into the file. If no entry is found, the date/time stamp will be displayed using default conventions. A separate entry for each database type must be entered into the file. If no entry is found for a given database type, the alarms/events for that database type will be displayed using default conventions.

Each template line should be no more than 94 characters including length of date/time stamp. Recognized tokens (as described below) will be resolved as required. Any other text on the line will be displayed as static text. For example, the colons (:) in the time stamp are interpreted as static text. RealFlex resolved information fields that are longer than the length reserved by the token will be truncated. For example, if the token for the PCU name is PPP and the SYSTEM PCU is applicable, it will be displayed as "SYS".

The following keywords should be placed on a single line to identify the format for that database type on the next line.

KEYWORD	DESCRIPTION
TAG_STAMP:	Date/time stamp
ANALOG_DB:	Analog database type
STATUS_DB:	Status database type
METER_DB:	Meter database type
TANK_DB:	Tank database type
PCU_DB:	PCU database type
DAYS:	Names for days of week Seven names started from Sunday Name cannot be longer then 3 characters
MONTHS:	Names for months of year Twelve names started from January Name cannot be longer then 3 characters
LIMITS:	Names for alarm limits Should be in following order: HI HIHI LO LOLO ROC INSTR Name cannot be longer then 5 characters



The following are the valid tokens that will be resolved for the specified alarm/event message types.

TOKEN	VALID DB	LENGTH	DESCRIPTION
YY	ALL	2	Year (2 digits)
MO	ALL	2	Month (2 digits)
DD	ALL	2	Day
HH	ALL	2	Hour
MM	ALL	2	Minute
SS	ALL	2	Second
AT	ALL	2	Alarm Type
P	ALL	<=12	PCU Name
T	ALL (except PCU)	<=12	Tag Name
X	ALL	<=20	Point Description
V	Analog, Tank	<=15	Value
U	Analog, Tank, Meter	<=10	Units
A	Analog, Tank	<=5	Alarm State
L	Analog, Tank	<=15	Alarm Limit
LASTSA	Status	6	Last State
CURRSA	Status	6	Current State
NO REPLY	PCU	8	PCU Comm. State
DAY	ALL	3	Day (3 letters)
MTH	ALL	3	Month (3 letters)
YEAR	ALL	4	Year (4 digits)

For tokens V and L, it is possible to use '?' after '.' to allow the system to determine where the decimal place goes.

Note: All keywords and tokens must be in UPPERCASE letters. It is impossible to assign different layouts for date/time, alarm type, PCU name, tag name and point description for different database types. Layout for date/time stamp, alarm type, PCU name, tag name and point description from first defined database type will be used for all following database types.

You can comment out any line if you start it with a semi-colon (;) character.

A customized example **alarm_config** file is as follows:

```
; Default system layout for date/time stamp and point description:
; YY/MM/DD HH:MM:SS AT PPPPPPPPPPP XXXXXXXXXXXXXXXXXXXXX
; Layout for date/time stamp
;
TAG_STAMP:
DD/MO/YY HH:MM:SS AT PPPPPPPPPPP XXXXXXXXXXXXXXXXXXXXX
;
; Layout for analog database
;
ANALOG_DB:
VVVV.? UUUUUUUUUU AAAA LL.LL
;
; Layout for status database
;
STATUS_DB:
CURRSA <- LASTSA
;
; Layout for meter database
;
METER_DB:
UUUUUUUUUU
```



```

;
; Layout for tank database
;
TANK_DB:
VVVVV.V UUUUUUUUUU AAAA LL.LL
;
; Layout for PCU database
;
PCU_DB:
NO REPLY
;
DAYS:
Sun Mon Tus Wed Thu Fri Sat
;
MONTHS:
Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
;
LIMITS:
Hi HiHi Lo LoLo Roc Instr

```

To change alarm states description file, the **/realflex/dial/event.dial** must be edited.

Default **event.dial** file is as follows:

```

AK - ACKNOWLEDGE;
A - ALARM;
EV - EVENT;
C - CRITICAL;
U - URGENT;
OK - RETURNED TO NORMAL;

```

A customized example **event.dial** file is as follows:

```

Ac - ACKNOWLEDGE;
AI - ALARM;
Ev - EVENT;
Cr - CRITICAL;
Ur - URGENT;
Ok - RETURNED TO NORMAL;

```

Note: Only first two characters can be changed. Order of lines must be the same as in above examples.

Please also note: If the default alarm configuration is modified for an existing project, the Selective Alarm Print utility may yield unpredictable or undesirable results.

What this means is the default Alarm Display Configuration will create event files with a set of columns. These event files are read by the Selective Alarm Print utility.

If you change the Alarm Display Configuration at the beginning of a project then all event files have the same format. However, if you change this format on an existing project, then the Selective Alarm Print will have to cope with old format data and new format data, and may have difficulty accessing both formats, e.g., if for 5 days you are using American format date 01/25/2003 and then you change to European format 25/01/2003, then the Selective Alarm Print may get confused when retrieving this data from the event files.



4.4 Channels configuration file (driver.chn)

The Telemetry Editor included in Flex.View V3.x, can be used for any RealFlex scanner that supports remote editing of telemetry information.

Telemetry information for each scanner consists of two parts – Channel information and PCU information. Channel information is stored in the "[scanner_name].cfg" file. The PCU configuration file must have a "scanner_name].pcu.cfg" file. The scanner developer must provide these two files. Some scanners might have only one configuration file. Configuration files must be stored in the "/realflex/data/crt/fwd/tel" folder.

The Channels Configuration file "driver.chn" must be created manually using the Configuration File Editor.

➔ To create the "driver.chn" file:

1. In Flex.View, from the **Configuration** menu, click on the **Configuration File Editor** option.

The "Flex.CFE" application will be opened.

2. From the **File** menu, select the **New** option.

Shortcuts: Toolbar  Keypad **Ctrl+N**

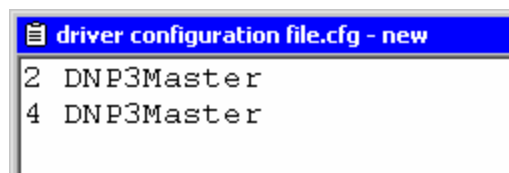
The "New" dialog box will appear:

3. Click on the "Driver Configuration" tab and select the "driver.chn" icon then click on the **OK** button.

Note: If the "driver.chn" icon is not visible in the "New" dialog box, then this file should already exist in the "/realflex/data/crt/fwd/tel" folder. You will need to open and edit this file via the "Open File" dialog box. See "To update the "driver.chn" file" below.

4. Create or edit the "driver.chn" file.

Each line in the "driver.chn" file must have a channel number and a scanner name running on that channel. In the example shown below, we have channels 2 and 4 using the "DNP3Master" scanner name:



```

2 DNP3Master
4 DNP3Master

```

5. After you have created or edited your file, send the file directly to the RealFlex server by choosing the **Send to Server** option from the **File** menu.


The file will be sent to the RealFlex server and stored in the "/realflex/data/crt/fwd/tel" folder.

Because this file always stores on the RealFlex server you only need one "driver.chn" file. Every Flex.View console will use it as soon as this file has been created.

➔ To create the binary files for each channel:

Using the "driver.chn" file, the Telemetry Editor will be able to select the appropriate configuration file for a PCU. First it gets the channel number from the PCU record, then it reads the "driver.chn" file and finds what configuration files contain channel and PCU structure definitions.



1. Within Flex.View, from the **Configuration** menu, select the **Database Editor** option. The "Database Editor - List of project PCU:" window will appear listing all PCU's on the system.
2. Click on and highlight the first PCU name, then from the "Telemetry Editing" toolbar, click on the  button.

An "Operation in progress" window will appear while Flex.View receives the "Channel description" file, "driver XXXX channel configuration" file and the "channel driver XXX settings".

When this process has completed, the "Telemetry Editor - Driver: XXX" window will appear.

3. Click on the **OK** button.

Two files will be created on the RealFlex server and stored in the "/realflex/data/crt/fwd/tel" folder. These files are:

[rtu_name].tel - The binary file with "PCU_XX" telemetry settings.
[scanner_name].1 - The binary file with "Channel 1" telemetry settings.

4. Repeat Steps 2 and 3 above for all remaining PCUs and channels listed and defined in the "driver.chn" file.

Once you have created your binary files, proceed to edit the Telemetry settings as required. For details, please refer to the "To edit Telemetry settings" section within the Flex.View Help information.

➔ To update the "driver.chn" file:

1. Within Flex.View, from the **Configuration** menu, click on the **Configuration File Editor** option.
The "Flex.CFE" application will be opened.
2. From the **File** menu, select the **Open...** option.
3. In the "Open File" dialog box, select "Node 1" on the left side of dialog box and select "Driver configuration...." in "Files of type:" field.

For example if we have a DNP3Master scanner running on channels two and four, then the "/realflex/data/crt/fwd/tel" folder should contain the following files:

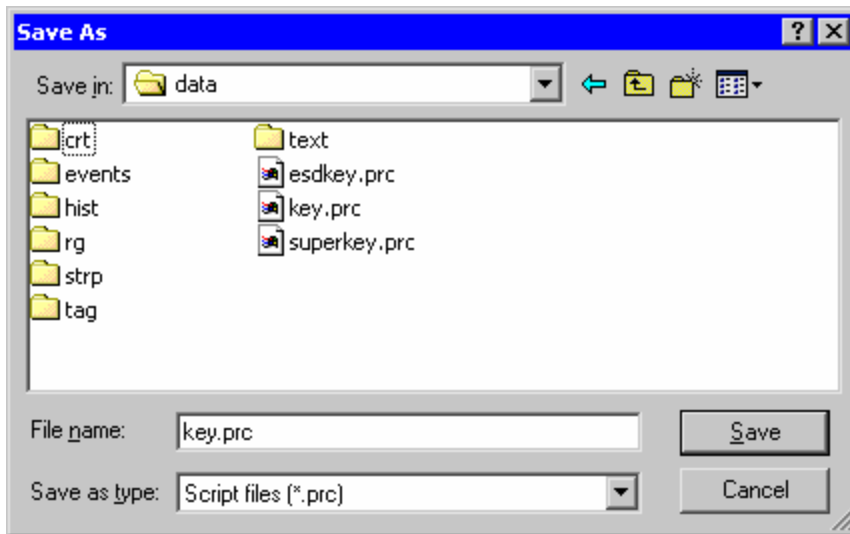
DNP3Master.cfg	- channel configuration definition
DNP3Master.pcu.cfg	- PCU configuration definition
[PCU name].tel e.g., RTU_1.tel, RTU_2.tel etc.	- binary file(s) with "PCU_XX" telemetry settings, created by Flex.View
DNP3Master.2	- binary file with channel 2 settings, created by Flex.View
DNP3Master.4	- binary file with channel 4 settings, created by Flex.View
driver.chn	- file with 2 lines: 2 DNP3Master 4 DNP3Master

4. Select the "driver.chn" file and click on the **Open** button.
5. Edit the "driver.chn" file as required then send the file directly to the RealFlex server by choosing the **Send to Server** option from the **File** menu.




5 Dialog boxes

5.1 Save As dialog box




The following options allow you to specify the name and location of the file you're about to save:

Save in: Displays the current folder and its list of available folders and files. Double-click on the folder you want to open. To see where the current folder is located in the hierarchy of folders, click on the  button. The resulting drop-down list displays folders above the selected location.

Go To Last Folder Visited:  Moves to the last folder you visited.

Up One Level:  Moves the "Save in" folder up one level in the directory hierarchy.

Create New Folder:  Creates a new folder in the current directory hierarchy.

View Menu:  Allows you to change the appearance of items in a folder. Click the View menu button, and then click on **Large Icons**, **Small Icons**, **List**, **Details**, or **Thumbnails** as required from the drop-down list.

File name: Displays files of the type specified in the "Save as type" box. To save a file, choose from the list, or type a path and filename in the box.

Save as type: Displays the file types you can save as, identified by their filename extension. To change the type, choose from the drop-down list.

Save button: Saves the selected file.





Index

A

About Flex.CFE	37
ABS - CSL Operator	55
Superkey Operator.....	80
Action menu - commands.....	41
Check syntax	28
Restart CSL process.....	36
Restart RealFlex on xxx Node	36
Alarm_config file - Creating and editing	88
New	23
Opening.....	25
Alt+F4 - Exit Flex.CFE shortcut	22
AND - CSL Operator.....	55
Superkey Operator.....	80
Arrange Icons	22
ASSIGN (CSL Procedure - DEBUG MODE)	43

B

Bookmark - Adding	34
Clear all.....	35
Goto next.....	34
Goto previous.....	34
BREAK (CSL Procedure - DEBUG MODE)	43

C

CALC (CSL Statement)	51
calcdo file - New	23
CAP	16
Caps Lock.....	16
Cascade windows.....	22
Cfg files - Opening.....	23, 25
New	23
Change - Font sizes used in Configuration File Editor	21
Change language	13
Channels Configuration file - driver.chn file.....	91
Check syntax (Action menu).....	28
Cld files - Opening	25
Clear all (Edit menu).....	35
Close.....	22
Flex.CFE application.....	22
Close (File menu)	26
Close All.....	22
Closing windows.....	18
Coldstart file - New	23
Opening.....	25
Restart RealFlex on Main or BackUp Node.....	36
Command Sequence Language (CSL files).....	36
Connect to the RealFlex server	12
CONTINUE WHEN - CSL Statement	51
Superkey keyword/statement.....	73
Copy (Edit menu).....	30
COS - CSL Operator	55
Superkey Operator.....	80
Create - Alarm_config file	88
CSL files.....	43
driver.chn file.....	91



Event.dial file.....	88
Superkey files	67
CSL (CSL Statement).....	51
CSL files - AGA3 Calculation functions	61
Calculation functions.....	61
Keywords	50
Monitor and debug modes and options	43
New	23
Opening.....	25
Operators	55
Overview	43
Procedure file organization	43
Procedures.....	43
Statement syntax	43
Statements.....	51
Symbol tables	43
CSL process - Restart	36
CSL_OFF - CSL Keyword	50
CSL Statement.....	51
CSL_ON - CSL Keyword.....	50
CSL Statement.....	51
Ctl files - Opening	25
Ctrl+F4 - Close window shortcut.....	18
Ctrl+A - Select All shortcut.....	30
Ctrl+C - Copy shortcut.....	30
Ctrl+F - Find shortcut.....	31
Ctrl+F2 - Add bookmark shortcut	34
Ctrl+H - Replace shortcut	33
Ctrl+N - New file shortcut	23
Ctrl+O - Open file shortcut.....	25
Ctrl+P - Print shortcut	28
Ctrl+S - Save file shortcut.....	27
Ctrl+Shift+F2 - Clear all bookmarks shortcut	35
Ctrl+V - Paste shortcut	30
Ctrl+X - Cut shortcut.....	30
Ctrl+Y - Redo shortcut.....	31
Ctrl+Z - Undo shortcut	31
Cut (Edit menu)	30
D	
Datac Sales	37
Datac WEB site	37
DEBUG (CSL Procedure - DEBUG MODE).....	43
CSL Procedure	43
Delete - Files	28
Delete (File menu)	28
DEMAND_SCAN (CSL Statement).....	51
Dialog boxes - Find.....	31
New	23
Open File.....	25
Password	13
Print Preview	29
Print Setup	29
Replace.....	33
Save As.....	93
User ID.....	13
DISPLAY - CSL Statement.....	51
Superkey keyword/statement.....	73
Display - Toolbar	18
Driver Configuration file (driver.chn) - Creating and changing.....	91



New	23
Driver_chn file - Opening.....	25
DSAVE - CSL Statement.....	51
Superkey keyword/statement.....	73
DSTUFF (Superkey keyword/statement)	73
DSTUFF A=B (CSL Statement).....	51

E

Edit - Alarm_config file.....	88
CSL files.....	43
driver.chn file.....	91
Event.dial file.....	88
Superkey files	67
Edit menu - commands.....	40
Bookmark	34
Clear all.....	35
Copy.....	30
Cut.....	30
Find	31
Find next	32
Find previous	32
Goto next.....	34
Goto previous.....	34
Paste.....	30
Redo.....	31
Replace	33
Select All	30
Undo.....	31
Editor.cfg file - Opening.....	25
New	23
ELSE (CSL Statement).....	51
E-mail - Datac Sales.....	37
ENDCSL (CSL Statement)	51
ENDIF (CSL Statement).....	51
ENDKEY SEND (Superkey keyword/statement).....	73
Event.dial file - Creating and editing.....	88
Opening.....	25
Example - Superkey procedure files	85
EXIT (CSL Statement).....	51
Exit (File menu)	22
EXIT SET (Superkey keyword/statement).....	73
EXTREMELY VERBOSE MODE (CSL Procedure)	43

F

F1 - On-line Help shortcut	37
F2 - Goto next bookmark shortcut.....	34
F3 - Find next shortcut.....	32
FABS - CSL Operator.....	55
Superkey Operator.....	80
File menu - commands	39
Close.....	26
Connect.....	12
Delete.....	28
Exit	22
Login User.....	13
New	23
Open	25
Print.....	28
Print Preview	29
Print Setup	29



Save	27
Save all	27
Save As.....	27
Send to Server	27
Files - Close without saving.....	26
Deleting	28
New	23
Open existing	25
Print preview	29
Printing	28
Saving	22, 27
Saving all.....	22, 27
Saving under a new name	27
Sending to the RealFlex server	27
Find (Edit menu)	31
Find and replace text	33
Find next (Edit menu)	32
Find previous (Edit menu)	32
Find text.....	31
Flex.CFE - About information	37
Application window	11
Operation overview	9
Starting.....	11
Title bar	16
FLOAT (CSL Statement)	51
Font settings (Options menu)	21

G

GOTO - CSL Statement	51
Superkey keyword/statement.....	73
Goto next (Edit menu)	34
Goto previous (Edit menu).....	34

H

Help	37
HELP (CSL Procedure - DEBUG MODE)	43
Help menu - commands	41
About Flex.CFE.....	37
Help.....	37
Hide - Status Bar	16
Output window	21
Toolbar	17, 18
HOUR - CSL Keyword.....	50
CSL Statement.....	51

I

IF - CSL Statement.....	51
Superkey keyword/statement.....	73
Insert key	16

L

Language (Options menu).....	13
Language setting	13
Local files.....	25
LOG - CSL Operator.....	55
Superkey Operator.....	80
Login User (File menu)	13



M

Match - Case (Find dialog box)	31
Whole word (Find dialog box)	31
Maximizing windows	18
Minimizing windows	18
MINUTE - CSL Keyword	50
CSL Statement	51
MONITOR (CSL Procedure - DEBUG MODE)	43
MONITOR MODE (CSL Procedure)	43
MONTH (CSL Keyword)	50
MONTH_DAY (CSL Keyword)	50
Moving - Toolbar	18
Windows	18

N

New (File menu)	23
NEW_YEAR (CSL Keyword)	50
NOOPERATOR (Superkey keyword/statement)	73
NUM	16
Number Lock key	16

O

Open File (File menu)	25
Opening and closing windows	18
Operations overview	9
OPERATOR (Superkey keyword/statement)	73
Options menu - commands	41
Font settings	21
Language	13
OR - CSL Operator	55
Superkey operator	80
Output window - open/close	21
OVR	16

P

Paste (Edit menu)	30
POPERATOR (Superkey keyword/statement)	73
Prc files - Opening	25
Print (File menu)	28
Print Preview (File menu)	29
Print Setup... (File menu)	29
PRINT_SCREEN (Superkey keyword/statement)	73
Product specific language	13
Project files - Opening	25

Q

QUIT (CSL Procedure - DEBUG MODE)	43
---	----

R

RCOS - CSL Operator	55
RCOS - Superkey Operator	80
RealFlex - Connecting to the server	12
RealFlex Internal Flags Accessible (CSL Procedure)	43
RECIPE (Superkey keyword/statement)	73
Redo (Edit menu)	31
Referencing - Historical Data (CSL Procedure)	43
RealFlex Internal Flags (CSL Procedure)	43
REMOVE_DISPLAY (Superkey keyword/statement)	73



Replace... (Edit menu).....	33
REPORT (Superkey keyword/statement).....	73
Resizing and moving windows	18
Restart CSL process (Action menu).....	36
Restart RealFlex on xxx Node (Action menu)	36
Restore - Windows	18
RETURN (Superkey keyword/statement).....	73
Rptcron configuration file - New	23
Opening.....	25
RSIN - CSL Operator.....	55
Superkey Operator.....	80
RTAN - CSL Operator	55
Superkey Operator.....	80

S

Sales@datac-technologies.com hyperlink	37
Save - A file	27
All files.....	27
Files under a new name.....	27
Save (File menu)	27
Save all (File menu).....	27
Save As (File menu)	27
Save As dialog box.....	93
Scanner configuration files	91
Script files - Opening	25
New	23
Scroll bars.....	18
Search for words	31
Select All (Edit menu)	30
Selecting - All text.....	30
SEND (Superkey keyword/statement).....	73
Send to Server (File menu)	27
Server - Sending updated files	27
Connecting to.....	12
SET (Superkey keyword/statement).....	73
Setting - Language	13
SHELL (Superkey keyword/statement)	73
Shift+F2 - Goto previous bookmark shortcut.....	34
Shift+F3 - Find previous shortcut	32
Show - Status Bar.....	16
Output window	21
Toolbar	17, 18
SIN - CSL Operator	55
Superkey Operator.....	80
Size - Windows.....	18
Splitting windows	18
SQRT - CSL Operator	55
Superkey Operator.....	80
Starting - Flex.CFE	11
Startrf file - Opening	23, 25
New	23
Status Bar	16
STEP - CSL Procedure - DEBUG MODE	43
Superkey keyword/statement.....	73
STUFF (Superkey keyword/statement)	73
Superkey - Accessible internal flags	70
Controlling Superkey access from the System Summary screen.....	67
Examples	85
Keywords	73
Language Statement Syntax.....	70



Operators	80
Procedure files	67
RealFlex Internal Flags accessible by superkey procedure	70
Referencing the Real-Time Database	70
Statements	73
Status records	67
Testing Superkey logic	67
SUPERKEY (Superkey keyword/statement)	73
Superkey files - Overview	67
New	23
Procedures	70
Syntax - Check syntax of files	28
SYNTAX (CSL Calculation function)	61
System files - Opening	25
T	
TAN - CSL Operator	55
Superkey Operator	80
Telemetry Config File	23
Telemetry Editor - Creating binary files for each channel	91
Creating or editing the driver.chn file	91
Tile - Horizontally	22
TIME (CSL Keyword)	50
Title bar	16
Toolbar	17
Moving	18
Show/hide	18
Translation Language Settings	13
U	
Undo (Edit menu)	31
User - ID's	13
Password	13
Using - Flex.CFE windows	18
Scroll bars	18
V	
VERBOSE (CSL Procedure - DEBUG MODE)	43
VERBOSE MODE (CSL Procedure)	43
View menu - commands	40
Output	21
Status Bar	16
Toolbar	17
W	
WEEK_DAY (CSL Keyword)	50
WHERE (CSL Procedure - DEBUG MODE)	43
Windows - Moving	18
Sizing	18
Splitting	18
Windows menu	22
Www.datac-technologies.com hyperlink	37
Y	
YEAR_DAY (CSL Keyword)	50



