C-Bus™ Basic Logic

C-Bus™ Products Training Course

Training Guide

1250SM0905R10/09

Retain for future use.

Schneider Electric

HAZARD CATEGORIES AND SPECIAL SYMBOLS

Read these instructions carefully and look at the equipment to become familiar with the device before trying to install, operate, service, or maintain it. The following special messages may appear throughout this bulletin or on the equipment to warn of potential hazards or to call attention to information that clarifies or simplifies a procedure.

The addition of either symbol to a “Danger” or “Warning” safety label indicates that an electrical hazard exists which will result in personal injury if the instructions are not followed.

This is the safety alert symbol. It is used to alert you to potential personal injury hazards. Obey all safety messages that follow this symbol to avoid possible injury or death.

⚠ DANGER
Danger indicates an immediately hazardous situation which, if not avoided, will result in death or serious injury.

⚠ WARNING
Warning indicates a potentially hazardous situation which, if not avoided, can result in death or serious injury.

⚠ CAUTION
Caution indicates a potentially hazardous situation which, if not avoided, can result in minor or moderate injury.

CAUTION
Caution, used without the safety alert symbol, indicates a potentially hazardous situation which, if not avoided, can result in property damage or improper operation.

NOTE: Provides additional information to clarify or simplify a procedure.

PLEASE NOTE

Electrical equipment should be installed, operated, serviced, and maintained only by qualified personnel. This document is not intended as an instruction manual for untrained persons. No responsibility is assumed by Square D for any consequences arising out of the use of this manual.

CLASS B FCC STATEMENT

This equipment has been tested and found to comply with the limits for a Class B digital device, pursuant to Part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference in a residential installation. This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instructions, may cause harmful interference to radio communications. However, there is no guarantee that interference will not occur in a particular installation. If this equipment does cause harmful interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one or more of the following measures:

- Reorient or relocate the receiving antenna.
- Increase the separation between the equipment and receiver.
- Connect the equipment into an outlet on a circuit different from that to which the receiver is connected.
- Consult the dealer or an experienced radio/TV technician for help.

SAFETY PRECAUTIONS

Carefully read and follow the safety precautions below before attempting to install or maintain electrical equipment.

⚠ DANGER

HAZARD OF ELECTRIC SHOCK, EXPLOSION, OR ARC FLASH

- Apply appropriate personal protective equipment (PPE) and follow safe electrical work practices. See NFPA 70E.
- This equipment must be installed and serviced by qualified electrical personnel.
- Turn off all electrical power supplying this equipment before working on or inside the equipment.
- Always use a properly rated voltage sensing device to confirm that power is off.
- Replace all devices, doors, and covers before turning on power to this equipment.

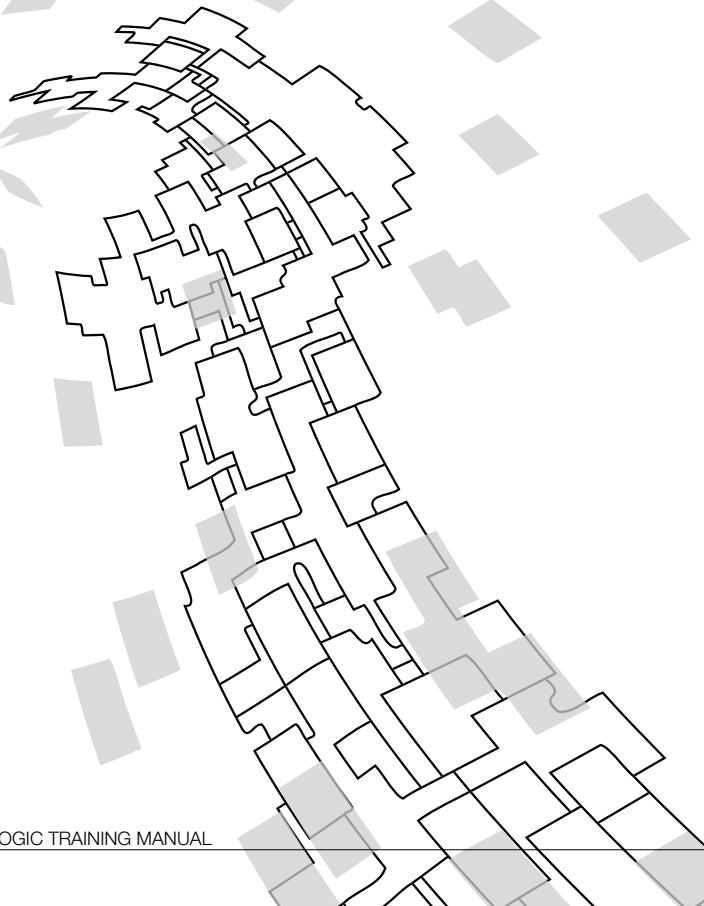
Failure to follow these instructions will result in death or serious injury.

The information in this manual is provided in good faith. Whilst Clipsal Australia Pty Ltd (CAPL) has endeavoured to ensure the relevance and accuracy of the information, it assumes no responsibility for any loss incurred as a result of its use. CAPL does not warrant that the information is fit for any particular purpose, nor does it endorse its use in applications which are critical to the health or life of any human being. CAPL reserves the right to update the information at any time without notice.

V2.0 Jan 2008

Contents

C-Bus Logic Training



Scope

This manual aims to provide the installer with the basic skills needed to program

C-Bus Logic in PICED, HomeGate or Schedule Plus. A fundamental technical background is required.

This manual covers:

- PICED File Management
- Colour Basics
- C-Bus Logic Engine
- Logic Engine Mechanics
- Logic Fundamentals
- PAC Logic Limitations.

The following prerequisites must be carried out before attending this course:

- C-Bus Basic
- C-Bus Touch Screens.

Learning Outcomes

By the end of this training course, you should have an understanding of:

- Various colour applications on a Colour Touch Screen
- Mechanics of the C-Bus Logic Engine
- Various logic theory and terminologies
- Various logic functions and applications.

1.0 PICED

PICED is used to configure the following devices to meet the user's requirements:

1) Black & White Touch Screen
2) Colour Touch Screen
3) Pascal Automation Controller (PAC).

NOTE  **The PAC does not have a physical screen, hence it is unable to display pages. However you can still use pages in your project for test purposes.**

The PICED software basic features include:

- Display of many components on many pages
- Scenes for the control of many loads together
- Schedules for the automatic control of loads
- Access Control to provide security
- Irrigation Control.

1.1 Software Limitation

Parameters	Black & White MkII	Colour	PAC
Maximum number of Pages	100	100	100
Maximum number of Components per Page	100	250	250
Maximum number of Schedules	250	250	250
Maximum number of Scenes	250	250	250
Maximum number of Components per Scene	100	250	100*
Maximum number of Users	20	20	N/A
Maximum Password Length	6	8	N/A
Maximum number of Access Levels	6	10	N/A
Maximum number of Irrigation Program	N/A	8	N/A
Maximum number of Irrigation Zones	N/A	16	N/A
Maximum number of Master Load Zones	N/A	2	N/A
Maximum number of Special Days	150	250	100
Maximum number of System I/O variables	100	250	N/A
Maximum number of Logic Timers	50**	50	20
Maximum number of Logic Modules	50**	100	50
Maximum number of Networks	10	20	10
Maximum number of C-Bus Applications	10	20	10

Table 1: Software Limitations.

NOTE  *** If large amounts of logic code are being used and scenes in the PAC are being triggered by logic, the maximum number of components per scene may be significantly reduced.**

**** Only available on Black & White MkII Touch Screen with logic enabled.**

1.2 Hardware Limitations

Feature	Black & White MkII	Colour	PAC
Display	320 x 240 pixels	640 x 480 pixels	No
Colours	Black & White	16 million colours	N/A *
Daylight Savings	Requires two schedules	Automatic	Requires two schedules
Selector Component	Yes	Yes	Yes *
HTML Component	No	Yes	Yes *
Web Cam Image	No	Yes	Yes *
Sliders	Vertical & Horizontal	Vertical & Horizontal	Vertical & Horizontal *
Bar Graphs	Vertical & Horizontal	Vertical & Horizontal	Vertical & Horizontal *
Alpha Blending	No	Yes	Yes *
Animated Images	No	Yes	Yes *
Special Functions	Unique Set	Unique Set	Unique Set
Remote Control	Yes	Yes	No
Running Programs	No	WAV files only	No
Access Control	Partial	Yes	No
Special Days	Yes	Yes	Yes
Irrigation Control	No	Yes	No
Logic	Yes**	Yes	Yes
System I/O	Yes	Yes	No
Network Modelling	Fully Connected Only	Choice	Fully Connected
Add / Delete Scenes	No	Yes	No
Add / Delete Schedules	No	Yes	No
Add / Delete Users	No	Yes	No
Edit Passwords	No	Yes	No
Load Monitor	No	Yes	No
Log	No	Yes	Yes *

Table 2: Hardware Limitations

NOTE 

*** These features are available in the PAC, only for testing purposes. They are not available in the PAC itself.**

**** Only available on Black & White MkII Touch Screen with logic enabled.**

1.3 Colour Options

The Colour Touch Screen offers a number of different ways to utilise colours in accordance with C-Bus buttons. The most commonly used colour features are:

- Colour Status Indication
- Colour Blending
- Alpha Blending.

1.3.1 Colour Status Indication

The Colour Status Indication is a feature of the Colour Touch Screen, which allows the colour of a component to change when it is turned on or off. This can be used with C-Bus and Text buttons, Sliders and Bar Graphs. It can be applied to the border and background and text of a component.

To select Colour Status Indication on a component:

1) Double click on the component and click on the Visual Properties tab
2) Select the Border or Background tab
3) Select the Background or Border Status indicator tick box
4) Select the Active and Inactive colours and press OK.

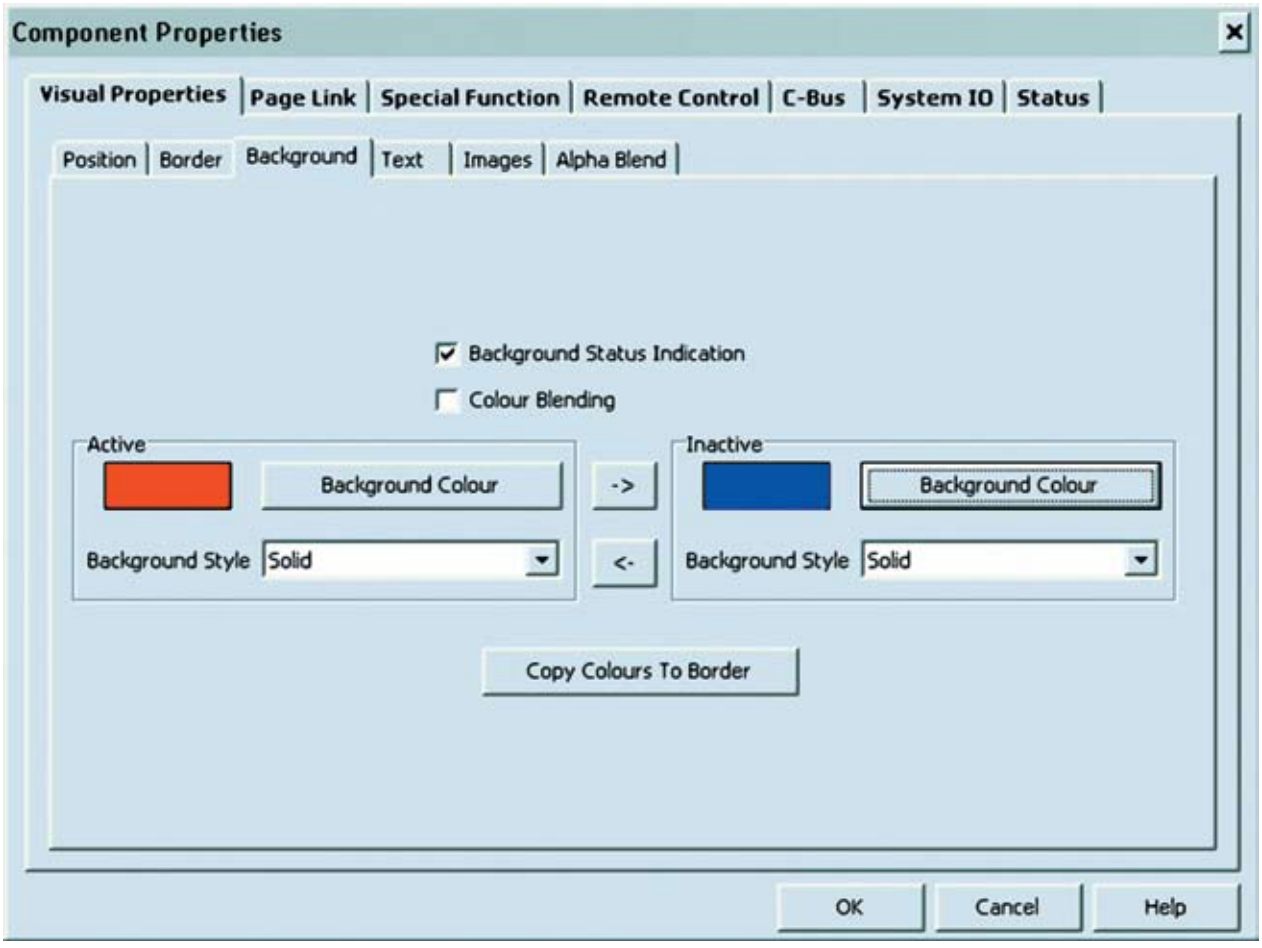


Figure 1: Selecting the Background Status Indicator.

1.3.2 Colour Blending

The Colour Blending is a feature of the Colour Touch Screen, which allows a smooth colour transition (between the Active and Inactive colours) as the Group Address is ramped up and down. This can be used with C-Bus and Text buttons, Sliders and Bar Graphs. It can be adapted to the border and background of a component.

To select Colour Status Indication on a component:

1) Double click on the component and click on the Visual Properties tab
2) Select the Border or Background tab
3) Select the Background or Border Status indicator tick box
4) Select the Background or Border Colour Blending tick box
5) Select the Active and Inactive colours and press OK.

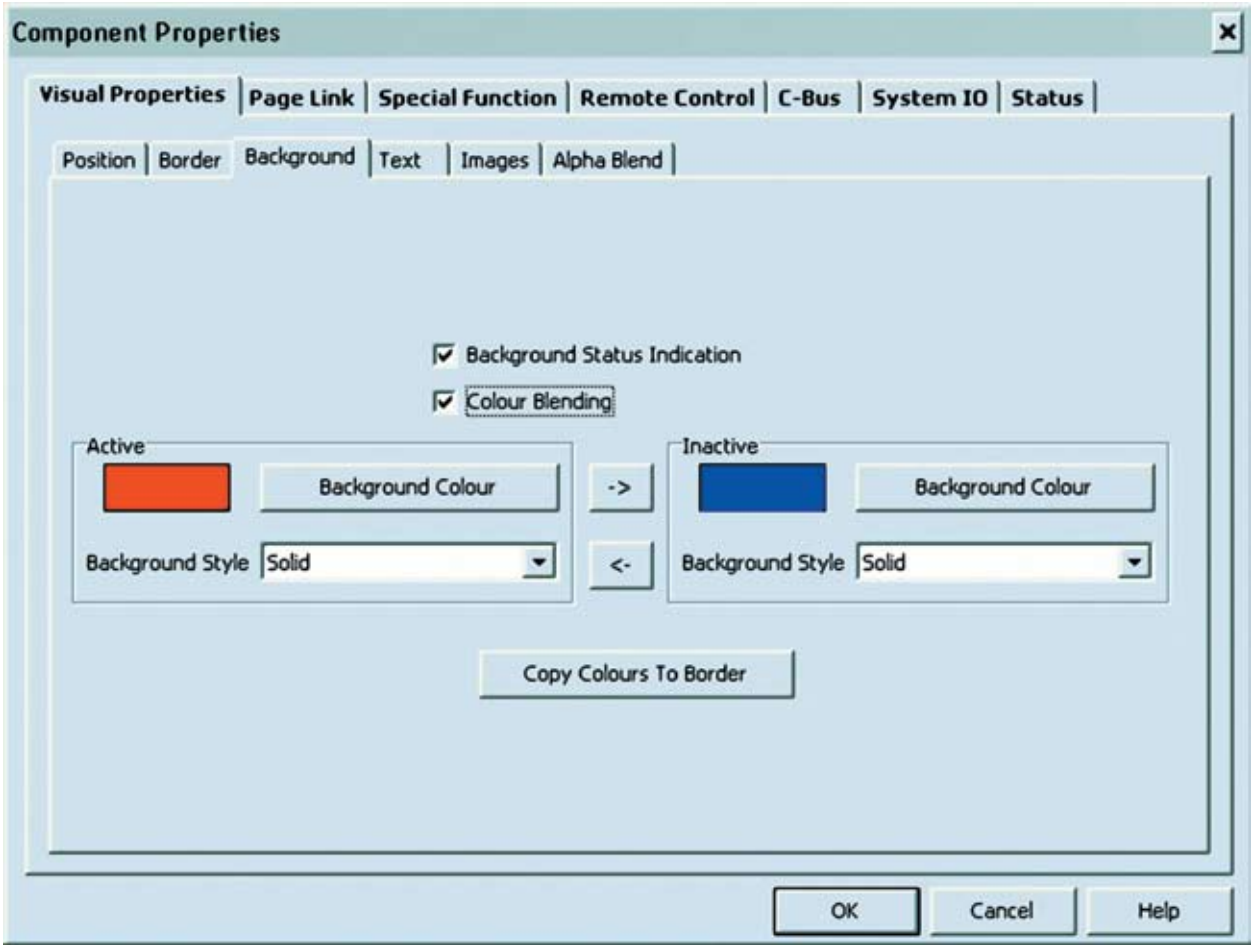


Figure 2: Selecting the Colour Blending parameter.

1.3.3 Alpha Blending

Alpha Blending is a function of the Colour Touch Screen that allows a component to become partially transparent. Alpha Blending can be applied to images and components. It also allows the selection of what properties of the component will be Alpha Blended eg. Border, Background, Text etc.

Figure 3: Example of an Alpha Blended button.

To apply Alpha Blending to a component:

1) Double click on the component and click on the Visual Properties tab
2) Select the Alpha Blend tab
3) Select which properties of the component you want to Alpha Blend from the Alpha Blend options drop down list
4) Select the Alpha Blend Level (to select the level of transparency)
5) Press the OK button.

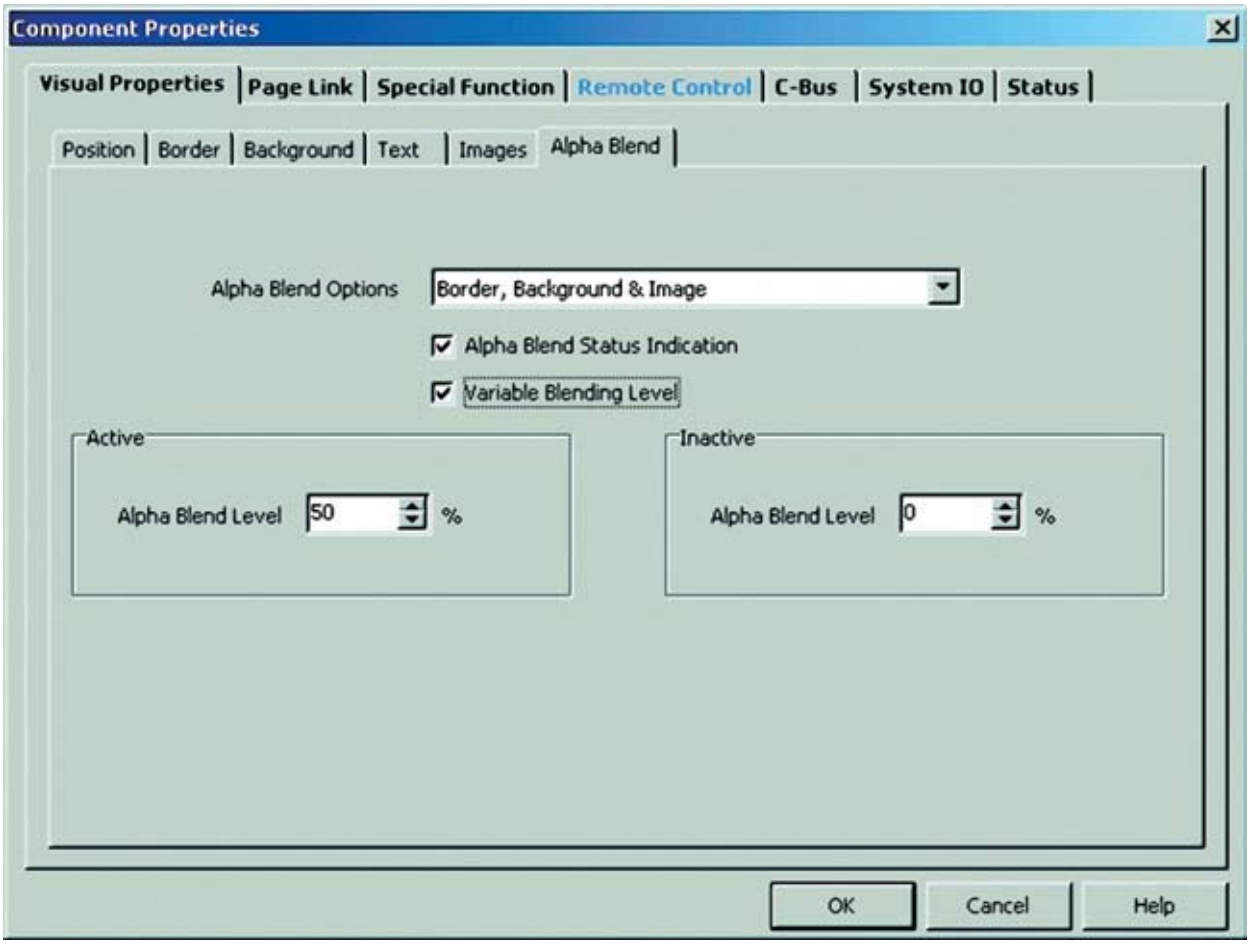


Figure 4: Selecting Alpha Blending.

WARNING **Excessive use of the Alpha Blending feature may cause the Colour Touch Screen to operate slowly.**

1.4 File Management

The PICED software allows the user to simply import and export projects with a single, easy to use file called an Archive. Archived projects are identified by the cta file extension.

1.4.1 Importing An Archive

The following steps will guide you through the correct process of successfully restoring a backed-up PICED, Schedule Plus or HomeGate project. Please follow the steps below.

1) Open the PICED, Schedule Plus or HomeGate software
2) On the first page of the Project Details Wizard, select the 'Import Archive' option and click next
3) You will then be prompted to locate the archived file. Once you have selected the archive you wish to import click next
4) You will then be prompted to allocate a folder for where all the associated project files will be installed to. Once you have selected a folder click next
5) You will then be asked to install the Toolkit project. This will automatically select the relevant folders. Click next to continue
6) You will then be prompted if any fonts need to be restored. If this is the case restore them and click Finish
7) Finally, you will be prompted to open the project file. Locate the ctd file and click Open.

The import process is now complete. You may find that the Toolkit project does not exist in the project list. If this is the case restart the software (including C Gate).

1.4.2 Exporting An Archive

The following steps will guide you through the correct process to successfully backup a PICED, Schedule Plus or HomeGate project. Please follow the steps below.

1) Open the project that you want to backup
2) Click on File
3) Click on Archive
4) Click on Export to Archive
5) Select 'Just the files used by the project' and click next
6) Ensure 'Include C-Bus Project' is ticked and click next
7) Ensure 'Include Windows Fonts' is ticked and click next

NOTE  **If there are any 'non standard' fonts used in the project, you will need to copy them into the '<Project Directory>\fonts' folder first.**

8) Select the location of where you want the Archive file to be saved and click Finish.

The project is now backed up with all its associated files. The Archive file can now be backed up onto a CD or it can be emailed.

2.0 Logic Theory

The Logic Engine language is based on the Pascal programming language. Pascal was chosen because it is one of the easiest languages to learn, and is one of the most straight forward to read for someone with no experience in the language.

NOTE **It is not necessary to learn the full language in order to do basic logic functions.**

2.1 Logic Mechanics

The Logic Engine has an Editor that allows the user to enter their programs. The user's program is run every time that an input to the Logic Engine changes. This includes:

- The time
- C-Bus levels
- System I/O settings
- Selected page.

When the user program is executed, it is referred to as a "scan." Since the time is updated every second, the user program will be executed (scanned) at least once per second.

Certain actions can be performed when the Logic Engine first starts. These are put in the Initialisation section. Other actions are executed every time something changes. These are put in the Modules section.

The data flow of the Logic Engine is summarised by the flow chart below.

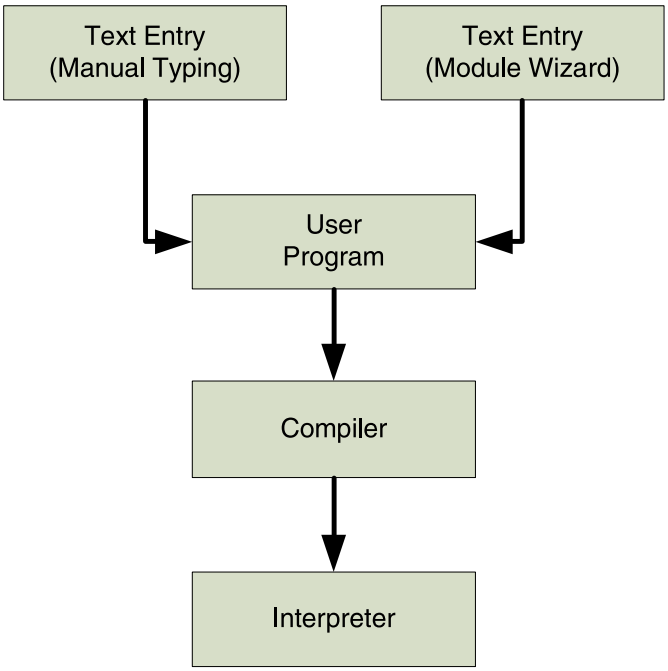


Figure 5: Logic Engine data flow.

The steps in the operation of the Logic Engine are as follows:

1) Creation of the user program, either by the direct entry of the program text, or by means of the Module Wizard
2) Compilation of the program
3) Running the logic in an Interpreter.

The operation of the Logic Engine is shown in the flow chart below:

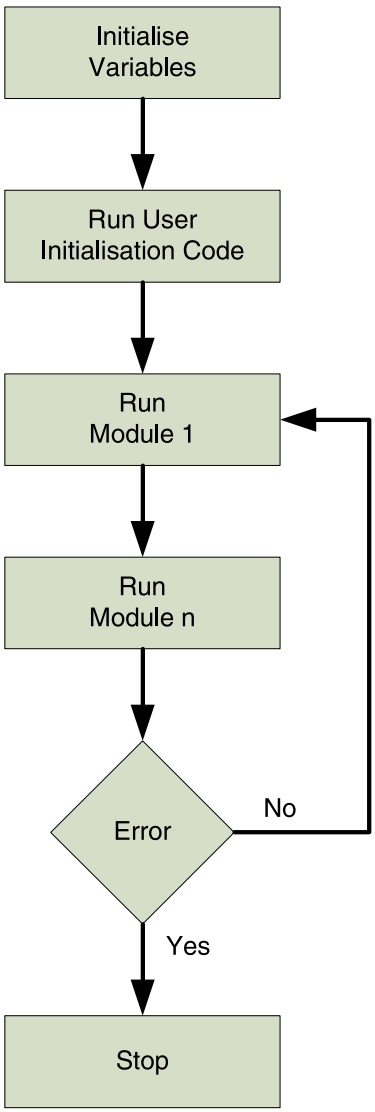


Figure 6: Logic engine operation.

The first step in the process is to initialise all variables. The users Initialisation Code is then executed. The Modules are then all executed in order. If there are no errors, then the Logic Engine waits for 0.2 seconds and then it runs the Modules again. When an error occurs, the Logic Engine stops.

2.2 Modules

Modules contain the main part of the user program. Modules separate the code into separate sections to:

- Place related parts of code together to make the code easier to understand
- Allow a section of code to be enabled or disabled
- Control which parts of the code gets suspended with a delay procedure.

Modules are created with the Logic Editor. When the logic program is compiled, a Pascal program is generated containing the code from each Module. This program is then wrapped in some automatically generated code, to implement the necessary features for Module operation.

The Module code is also wrapped in a repeat loop so that the Modules get executed on a regular basis. The result is a Pascal program similar to the one below.

```
program LogicEngine;
begin
        {Initialisation Code goes here}
        repeat
        if ModuleEnabled(“Module 1”) then
                begin
                {Module 1 code goes here}
                end;
                if ModuleEnabled(“Module n”) then
                 begin
                {Module n code goes here}
                end;
        WaitUntilSomethingHappens;
        until LogicEngineRunTimeError;
        end;
```

Figure 7: Logic program operation.

NOTE  **The user does not have to get involved with the complexities of how Modules get enabled or disabled. This is all handled by the compiler.**

To an extent, Modules create behaviour similar to a multi-tasking / multi-process / multi-threaded system. The main difference is that each Module does not have its own memory, as all modules share Global Variables.

Although there are many benefits in modularizing your code, each additional Module does require extra processing due to the "wrapping" process described above. For example, creating hundreds of separate Modules, each with just a few lines of code would be very inefficient.

For each scan, the Modules all get executed one after the other, in the order that they are listed in the Logic Tree. Any Modules which are Disabled or are Delayed are skipped until they are Enabled again.

2.3 Statements

A logic statement is a section of logic code that can be found in each Module. There may be a number of different statements in each Module that contain conditional logic or an action. The figure below shows two statements in a Module.

```
Module 1;
{Statement 1}
                once something happens then
                begin
                  turn something on;
                end;
{Statement 2}
                once something happens then
                begin
                  turn something off;
                end;
```

Figure 8: Example of two logic statements in a Module.

Most logic statements usually comprise of two sections:

1) A condition
2) An action.

2.3.1 Conditions

A Condition is simply a part of a logic statement that asks a question. Some examples of more commonly used conditions are:

- Once a C-Bus Group Address is on
- Once the time equals 11:00pm
- Once the date is > 1/1/2007 <1/6/2007
- Once a scene has been set etc.

2.3.2 Actions

An Action is a task that the logic engine carries out when a condition becomes true. Some examples of commonly used actions are:

- Turn off a Group Address
- Set a scene
- Enable a logic module
- Start a timer
- Show a page.

NOTE  **In some cases, actions may also be used without the use of a condition.**

So once a statement has been completed, you should be able to easily identify the condition and action of each statement. The figure below shows a complete logic statement where the condition is written in red, and the action is written in blue.

```
once (GetLightingState("Group Address 1") = OFF) then
begin
  SetScene("Master Off");
end;
```

Figure 9: A logic statement showing a condition and action.

2.4 Parenthesis

Parenthesis refers to the use of brackets in the logic engine. Brackets are used to group conditions together to achieve a desired outcome.

To help understand parenthesis, think of mathematics and the way that brackets are used in a simple maths problem.

- 10 - 2 x 4 + 6 = 8
- 10 - 2 x (4 + 6) = -10
- (10 - 2) x 4 + 6 = 38
- (10 - 2) x (4 + 6) = 80.

So as you can see, the placement of brackets gives a series of different outcomes or actions. This means that if you are not careful with the placement of brackets, you may not get the desired action as a result of your conditions.

Parenthesis is particularly useful when using a combination of 'And' and 'Or' logic.

2.5 Compile Errors

Compile Errors occur during the compiling process, when there is a syntax error in the logic code.

To find where there is an error in the code, just double click on the error message, and the cursor will be positioned at the position in the code where the compiler detected the error. It may be necessary to look either side of the cursor position, or even on the lines above to find the root cause of the error.

A single compile error may result in more than one error message, depending on whether the compiler can determine what was intended. If more than one error message is reported, it is always best to fix the first error in the list of errors first. If the following error messages make sense, then fix them at the same time, otherwise compile the code again and fix the first error in the new list.

Compilation Errors must be fixed before the program can be compiled and run.

3.0 Logic Editor

To open the Logic Editor, simply click on 'Project' and then select 'Edit Logic.' This can also be achieved by clicking on the logic button, found on the toolbar.

The Logic Editor is shown below and consists of several sections:

- A toolbar at the top, which provides access to common functions
- A logic tree on the left which provides access to various parts of the code
- A code window on the right where the users program is entered
- An output window at the bottom, where results are displayed
- A resources window at the bottom left showing the resources used.

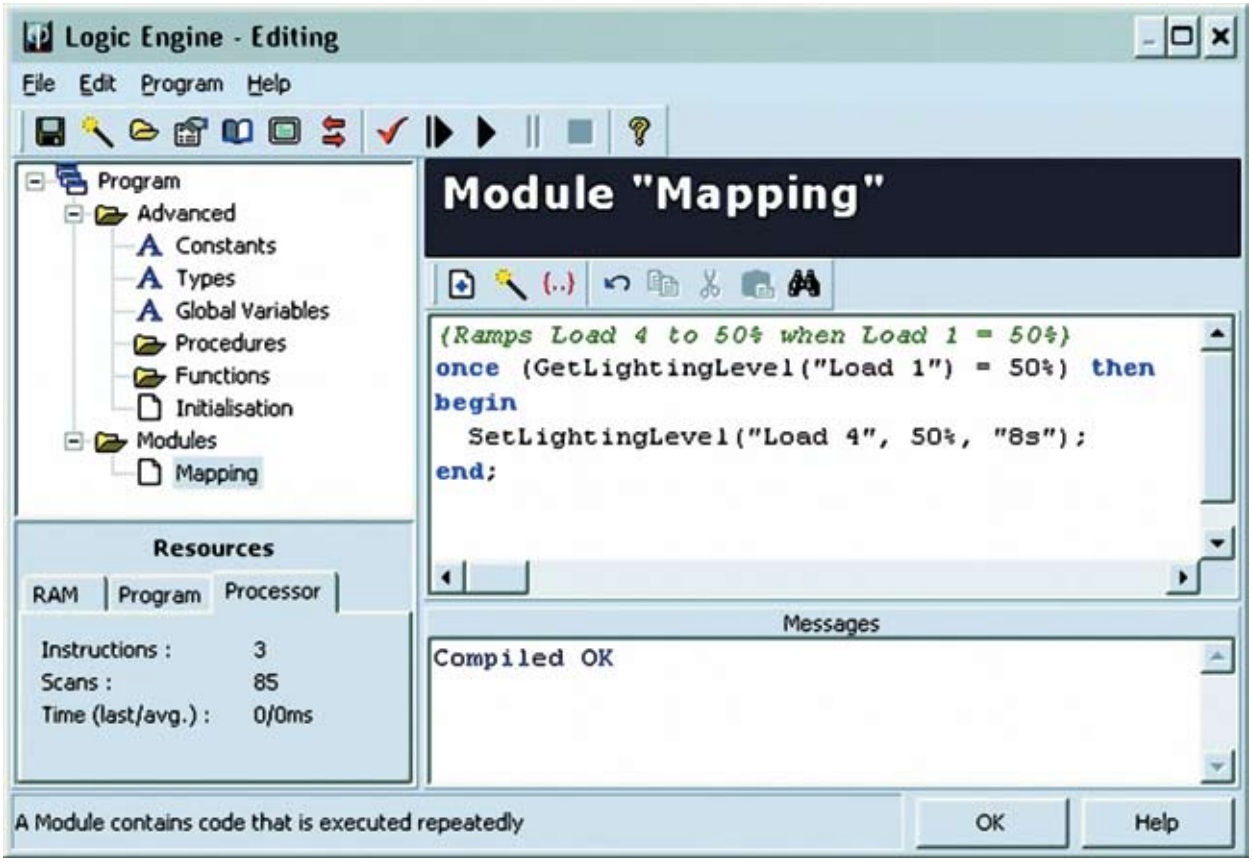


Figure 10: The Logic Editor.

3.1 Toolbar

The toolbar at the top of the Logic Editor allows quick access to commonly used features of the Logic Editor.

Figure 11: The toolbar in the Logic Editor.

The features available via the toolbar are:

- Save Project
- Module Wizard
- Adding Module Groups
- Logic Engine Options
- Logic Engine Report
- Edit System I/O Variables
- Compile
- Running Logic
- Help.

3.2 Logic Tree

The Logic Tree on the left of the Logic Editor provides access to the various parts of the code. The two main parts of the Logic Tree are:

- Advanced - this provides access to the advanced parts of the Logic Engine
- Modules - this is where most of the user program is written.

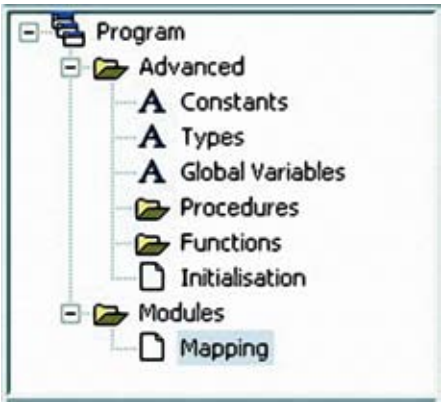


Figure 12: The Logic Tree.

The Advanced section contains the following sections, which (with the exception of Initialisation) correspond to sections of standard Pascal Code:

- Constants
- Types
- Global Variables
- Procedures
- Functions
- Initialisation.

To create new code sections:

- Constants: select the Constants node. Click on the Add button on the toolbar or pop-up menu. Enter the Constant name and value. Click on OK
- Types: select the Types node. Type the new type definition in the Code Window
- Global Variables: select the Variables node. Click on the Add button. Enter the Variable(s) name and type. Click on OK
- Procedures: select the Procedures node. Click on the Add button. Enter the Procedure name. Click on OK. Type the procedure body in the Code Window
- Functions: select the Functions node. Click on the Add button. Enter the Function name. Click on OK. Type the function body in the Code Window
- Initialisation: select the Initialisation node. Type the Initialisation Code in the Code Window
- Modules: select the Modules node. Click on the Add button. Enter the Module name. Click on OK. New Modules can also be added with the Module Wizard by clicking on the Wizard button
- Module Groups: click on the New Module Group button on the toolbar. Enter the name and click on OK.

If you right click on the Logic Tree, you will see a pop-up menu appear:

Figure 13: Logic Tree pop-up menu.

This allows you to perform functions like:

- Adding new code sections (see above)
- Inserting a new code section
- Deleting a code section
- Changing the order of the code sections (Move Up or Move Down)
- Manually Enabling and Disabling Modules.

If the Logic Engine is running, the state of Modules can be seen by the icons:

- A green tick indicates that the Module is enabled
- A red cross indicates that the Module is disabled
- A blue timer indicates that the Module is waiting for a Delay Procedure or a WaitUntil Procedure.

3.3 Code Window

The Code Window is the part of the Logic Editor that is primarily used for editing user code, as shown below.

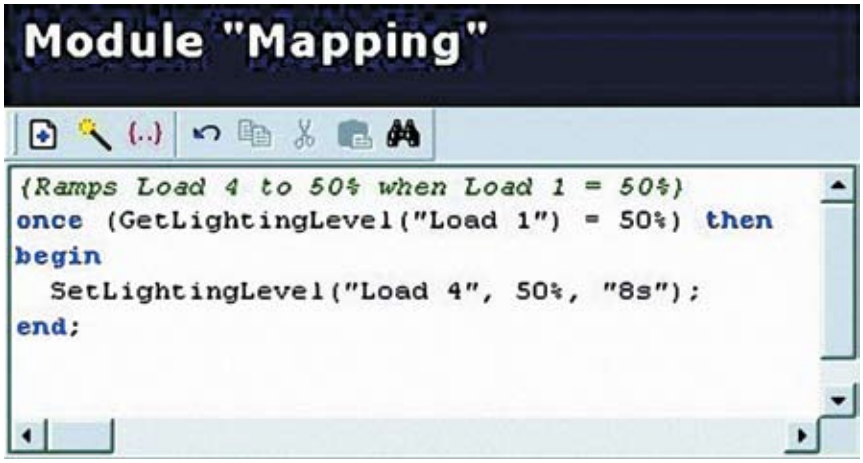


Figure 14: The Logic Code Window.

To enter code in a section of the program, click on the appropriate node in the Logic Tree, and enter text in the Code Window. If you cannot edit the code or it is greyed out, this means that the Logic Engine is running. The program must be stopped before code can be edited.

There are essentially four ways of generating code:

- The Module Wizard can also be used to automate the creation of code for Modules
- The Statement Wizard can be used for creating sections of code for a Module
- The pop-up menu can be used for automatically generating small sections of code
- Code can be typed in directly.

3.3.1 Logic Wizard

New Modules can be added with the Module Wizard by following this process:

1) Click on the Wizard button on the toolbar
2) Enter the Wizard Details
3) Click on the Next button.

Figure 15: The Module Wizard.

4) enter the Wizard Conditions

5) click on the Next button

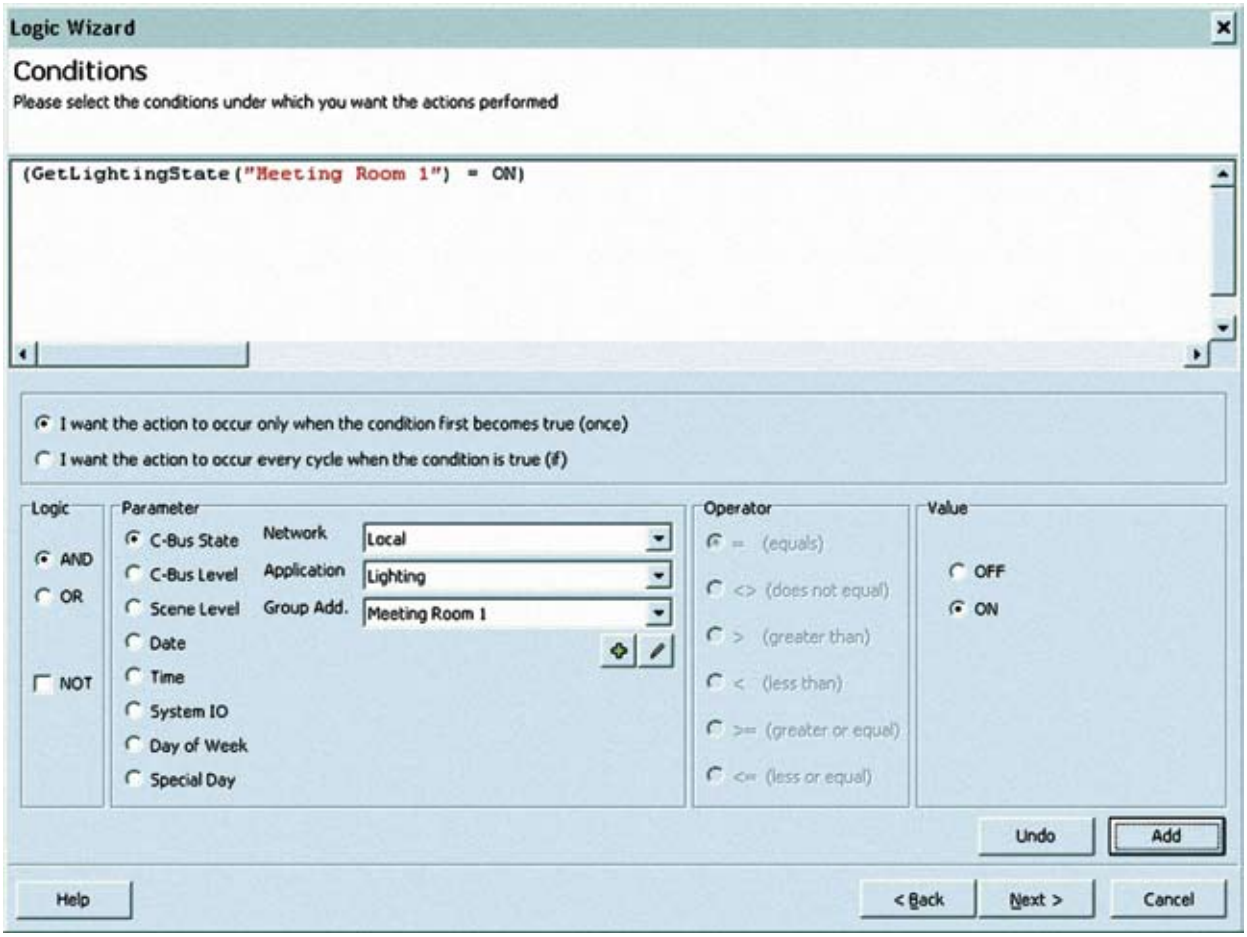


Figure 16: Adding conditions to a logic statement.

WARNING **The incorrect use of 'Once' and 'If' statements may cause errors in the Logic Engine. 'Once' must be used when the action of the logic statement is sending a command to the C-Bus network eg. Turn On Group Address 46.**

If you were to use an 'If' statement, then while the condition is true, the Logic Engine will continue sending the action onto the C Bus network, rather than a single time (when the condition first became true). This can potentially flood the network with commands.

6) enter the Wizard Actions

7) click on the Finish button.

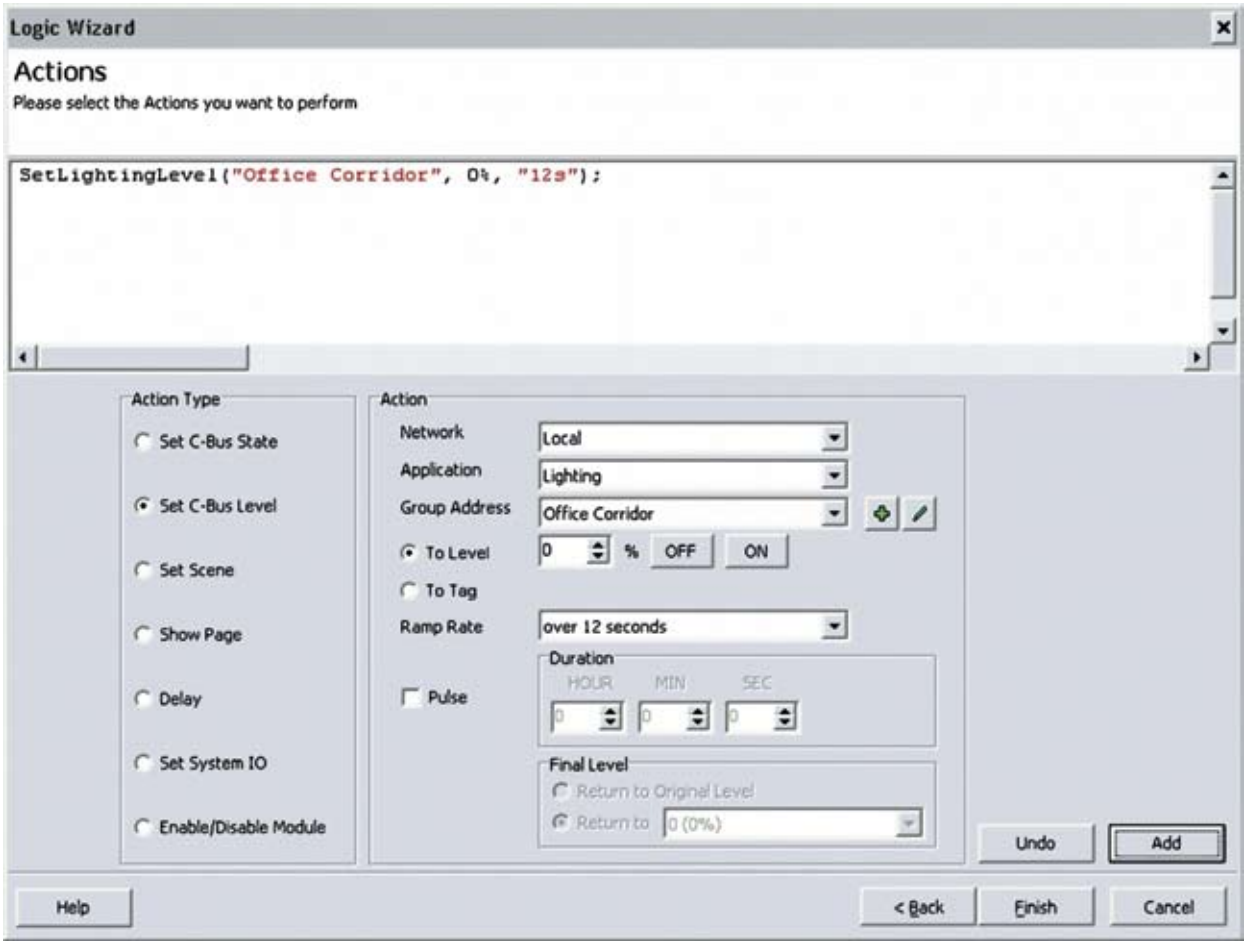


Figure 17: Adding actions to a logic statement.

8) The Logic Wizard will then generate the logic statement (using correct syntax) as shown below

```
once (GetLightingState(“Meeting Room 1”) = ON) then
begin
  SetLightingLevel(“Office Corridor”, 100%, “12s”);
end;
```

Figure 18: An example of a logic statement generated using the wizard.

9) If needed, changes or any additions can be made manually in the Code Window.

3.3.2 Line Insert

To enter code in a section of the program using the Line Insert feature, click on the appropriate node in the Logic Tree, and position the cursor at the point where you want the code to be inserted. If you right click on the code window, a pop-up menu will appear.

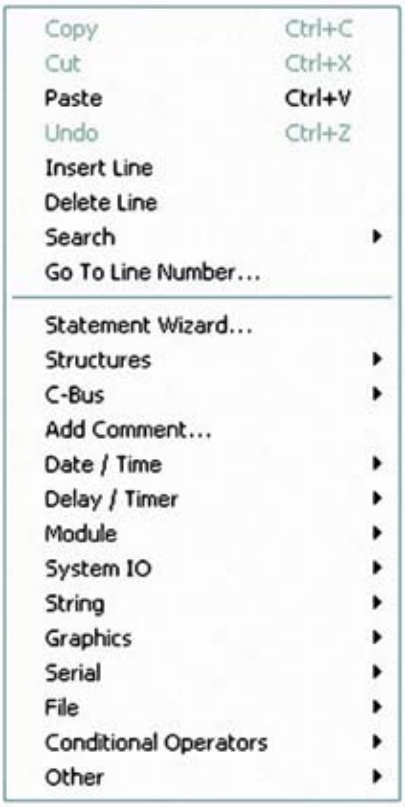


Figure 19: Line Insert parameters.

This provides access to most of the Logic Engine language elements. Many dialogue boxes are included to automate the generation of code.

You can also use the pop-up menu for editing to copy, cut and paste code, as well as undoing changes. A search facility is included to find words within the code window.

3.4 Message Window

The Output Window at the bottom on the Logic Editor is used to display:

- Results of the Compilation
- Error Messages
- Data written from the user program.

Figure 20: The Message Window.

3.5 Resources

The Logic Engine has finite resources, primarily memory. The Resources Window on the Logic Editor shows the following data:

- Stack Size: the Stack is where all Static (regular) variables are stored. The Stack Size is how much memory has been used
- Heap Size: the Heap is where Dynamic Variables are stored. The Heap Size is how much memory has been used
- Spare: this is the amount of data memory spare
- Program : this is the amount of program memory used
- Int: this is how much of the Integer constants memory has been used
- Real: this is how much of the Real constants memory has been used
- String: this is how much of the String constants memory has been used
- Instructions: this is the number of Interpreter instructions executed in the last scan
- Scans: this is the number of scans run since the Logic Engine was started
- Time: this is the time / average time taken by the Logic Engine scans. This should be less than 100ms for stable operation. For PAC (Pascal Automation Controller) projects, this shows an estimate of the time and maximum time as a percentage of the PAC capacity. This should not exceed 75%.

NOTE **Resource Data are only updated when the Logic Engine is Run.**

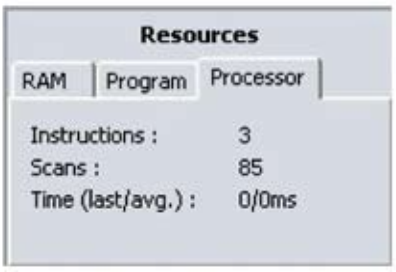


Figure 21: The Resources View.

3.6 Logic Help

The Logic Engine has a large amount of help files associated with the operation of the software, and all of the logic functions and parameters. The Logic Help files provide detailed information with useful examples of how logic code can be written.

To open the Logic Help files, either:

1) In the main view of PICED, click on Help and then select Logic, or
2) Open the Logic Editor and click on the Help icon.

NOTE **Examples in the Logic Help files may also be copied and pasted into the Code Window.**

4.0 PAC Logic Limitations

There is no definite answer to how much Logic Code is possible, as it depends on many factors including:

- The speed of your computer
- The logic functions used
- How the code has been written.

You can use the Resource Window to see how much of the resources are being used. There are several basic limits to the amount of code:

- The amount of memory available
- The amount of code storage space available
- How long the code takes to run (in the PAC, if it takes too long to run, it will be terminated).

NOTE  **Note that it is possible to run a lot more logic on your computer than in a Colour C-Touch or particularly in a PAC. For more information please consult the Logic Help files.**

You can get an idea of how much of a scan typical statements take in a PAC by looking at the table below:

C-Bus Commands		
Statement	**% of Scan**	**Comment**
SetLightingLevel("light", 100%, 4);	2%	
SetCBusLevel("local", "lighting", "light", 100%, 4);	2.2%	The SetLightingLevel is faster
SetScene("All On");	1.4%	Per Scene Component. Much faster than sending individual commands.
TrackGroup("Local", "Lighting", "Group 1", "Group 2");	0.4% to 2.4%	Larger value is if the tracking group needs to be changed

Conditions		
Statement	**% of Scan**	**Comment**
Once / If	0.1%	If is slightly faster than once.
GetLightingLevel("light") = 100%	0.3%	
GetCBusLevel("local", "lighting", "light") = 100%	0.4%	
GetLightingState("light") = ON	0.3%	
GetLightingState("light")	0.2%	This is quicker than above because no comparison (= ON) is used
GetSceneLevel('floor 1") = 0%	0.1%	Per scene item. Much faster than a series of GetCBusLevel commands.
Time = "7:00 PM"	0.2%	
ConditionStaysTrue(condition, "0:01:00")	0.4%	Not including the evaluation of the condition

Variables		
Statement	**% of Scan**	**Comment**
Flag := true;	0.1%	
Counter := 0;	0.1%	
Counter := Counter + 1;	0.2%	

Other		
Statement	**% of Scan**	**Comment**
Each additional module used	1%	
Begin / End	0%	
Comments or Blank Lines	0%	

By adding up the individual items that occur in the worst case scan, you can get an idea of whether a particular amount of logic is likely to fit in a PAC. Consider the code :

```
{This Module}                                    {1 %}
once (GetLightingLevel("Start") = 100%) then     {0.4 % The Once & the GetLightingLevel}
begin                                            {0 %}
        SetLightingLevel("Room 1", 100%);        {2 %}
        SetLightingLevel("Room 2", 100%);        {2 %}
        SetLightingLevel("Room 3", 100%);        {2 %}
end;                                             {0 %}
```

Figure 22: C-Bus Logic.

On the scan when "Start" goes to 100%, this will take approximately 7.4% of the PAC capacity (including 1% for the module it is in). On the other scans, it will only take 1.4% of the PAC capacity.

When the logic is running, the Resource Window shows a rough estimate of how much of PAC capacity the logic will use. You will need to leave the logic running and test that it doesn't exceed 75% (to allow some spare capacity). The Logic Engine Options form allows you to select whether you want to receive warnings if the PAC usage becomes excessive. To be confident that the PAC has the capacity to run your logic, you will need to test all aspects of the logic. This requires exercising every function of the logic and ensuring that the PAC usage does not exceed 75%. To be completely sure, you will need to transfer the project to the PAC and repeat the tests on the actual site.

Measuring the Actual Usage

The actual usage in the PAC can be monitored by connecting to the PAC and selecting the Transfer | Control Unit | Log PAC Messages option. This will log the PAC usage whenever it exceeds 75%. It is necessary to test the worst-case conditions to be sure that the PAC will be OK under all circumstances. See Controlling the PAC in the main help file.

SUPPORT AND TRAINING

Contact the Customer Information Center for technical support by phone at 1-888-778-2733 or e-mail at lightingcontrol.support@us.schneider-electric.com.

You may also find helpful information on our web site at www.Schneider-Electric.us.

Schneider Electric, USA
320 Tech Park Drive, Suite 100
La Vergne, TN, 37086
1-888-778-2733
www.schneider-electric.us

1250SM0908R10/09 **10/2009**