
INTRODUCTION TO VYSTA

Bardac drives



CONTENTS

| | |
|------------------------------------|-----------|
| Introduction to VYSTA | 5 |
| About Vysta | 5 |
| Application Range | 5 |
| PC Requirements | 6 |
| The VYSTA Platform | 7 |
| Drop-Down Menu | 7 |
| The Screen List | 9 |
| Adding a Screen | 10 |
| Edit Screen Dialog Box | 11 |
| Display Mode sub-block | 11 |
| Attributes sub-block | 12 |
| Edit Screen - Advanced Dialog Box | 12 |
| Edit Variable Reference Dialog Box | 13 |
| Numerical Type | 13 |
| Enumerated Type | 16 |
| Bargraph Type | 17 |
| The Schematic Editor | 19 |
| The Oval Menu | 20 |
| Oval Menu Options | 22 |
| Building a Schematic | 23 |
| Edit / Zoom Window | 25 |
| Floating Text | 26 |
| Variables | 27 |
| New Variable Function Block | 28 |
| Configuring Function Blocks | 29 |
| Function Block Parameters | 30 |
| Edit Parameters Dialog Box | 30 |
| Value sub-block | 31 |
| Flags sub-block | 31 |
| Modbus sub-block | 32 |
| Output Variable | 33 |
| Standard Program Function Block | 34 |
| E-Series Programming | 36 |
| Memory Constraints | 36 |
| Timing | 37 |
| Reading Function Block Outputs | 38 |
| PID Setup | 39 |
| Standard Program | 42 |
| Screen List | 42 |

REVISION HISTORY

| C | 20/02/2002 | Upgraded from Vista 1.1.0.0 to Vysta 1.5 |
|---|------------|--|
| B | 6/10/2000 | Serial Comms section removed and diagrams updated to Ver 1.1.0.0 |
| A | 14/02/2000 | Created |

Introduction to Vysta

The purpose of this document is to provide a preliminary reference for the user to modify existing Vysta programs, and to create simple Vysta programs. Vysta Virtual Automation is a graphical programming software platform produced by Bardac Drives for use with the E-Series and RVSx Series range of motor controllers, however this document will primarily discuss the E-Series. Please contact Bardac Drives if an RVSx compatible Vysta program is required.

The standard motor controller can be configured and commissioned from the Display Unit, as described in the appropriate Technical Manual (4201-180 for the E-Series). On a higher level, the motor controller can be customised using Vysta. This enables process control function blocks to be assembled and interconnected to perform custom applications.

Development of the Vysta Virtual Automation programming software platform is ongoing and therefore some aspects of Vysta may change without notice.

About Vysta

Three stages can be identified in the development of a Vysta program:

- **The Screen List**
The motor controller's Display Unit is a two line, 16 character LCD display. Vysta can be used to edit existing screens and / or add new screens, creating a customised Operator interface for a specific application.
- **The Schematic**
A graphical interface is used to create a function block based logic schematic that provides a customised real-time digital control system for the motor controller.
- **The Compiler**
The Vysta program is created and saved as a netlist file that contains the Schematic and the Screen List. The netlist file must be compiled, which creates a visual language object file. The compiled Vysta program, as a vlo file, is loaded into the motor controller through an RS232 serial link. The Vysta program can then be activated in the motor controller via screen Y3.

Application Range

Vysta was developed to allow the motor controller user to add functionality without the need for external control equipment. The ability of Vysta to perform any function is dependent on the constraints of the motor controller's

analogue and digital I/O, the available program functionality, and the application requirement.

The Schematic is very flexible and contains the major process control function blocks such as PID loops, logic functions, counters, comparators, rate limiters etc. The standard arithmetic operators such as addition, division, multiplication, integration and trigonometric functions are also available.

Vysta allows access to many of the motor controller's internal control variables (System variables) such as frequency, current and torque. Vysta has the ability to control any of the motor controller's parameters accessed by the Display Unit, such as the speed and torque references, the acceleration and deceleration rates and the stop modes etc.

The motor controller's analogue and digital I/O can be monitored and controlled totally independent of the standard program functionality.

From the outset, the range of applications envisaged for Vysta is to supplant external control equipment in common motor controller applications where typically a small PLC or external PID or other loop controller would be used. Applications such as closed loop pump control, lead pump systems, tension control, multispeed modes, crane modes etc can be configured with Vysta. As the motor controller uses Flash ROM for program storage, one of the main advantages over other drive manufacturers' programming products is the ability to provide ongoing program development without having to change EPROM's.

Position control is possible, but the limitation at this stage is due to the absence of a zero-reference input terminal in the E-Series encoder inputs.

Hence the limitation of Vysta to control an application will depend on a number of items...

- Input/Output requirement
- Speed of program execution
- Maximum size of program
- Functionality required

PC Requirements

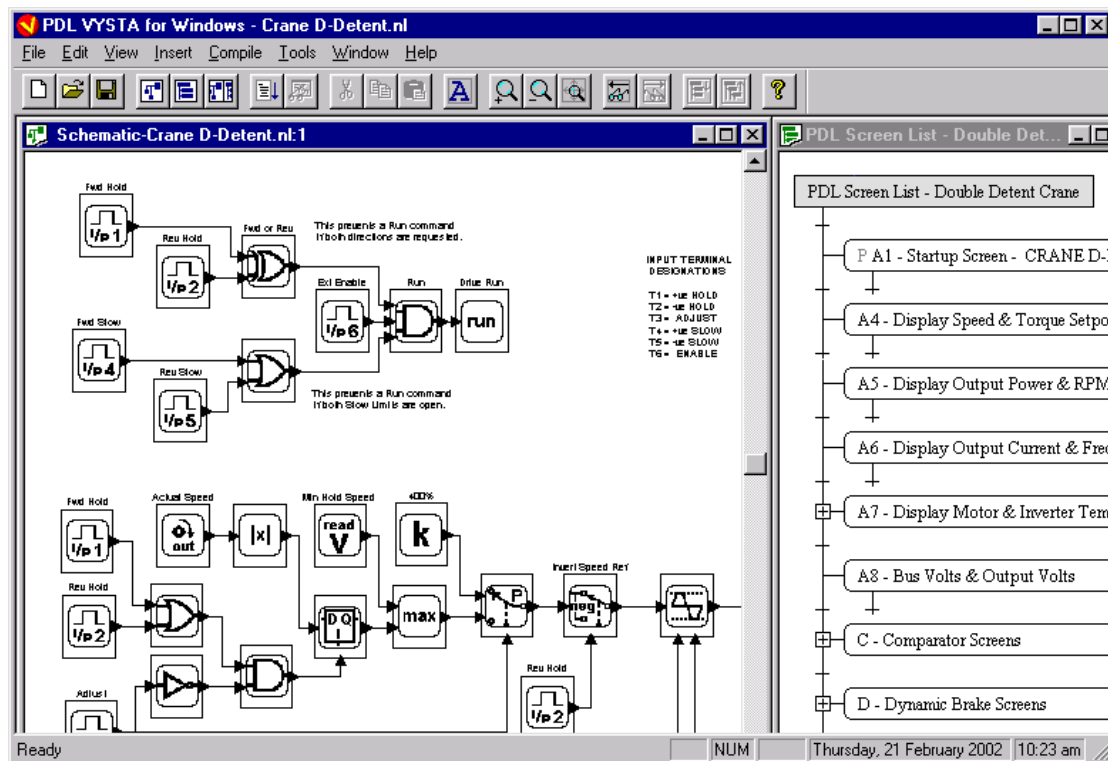
Vysta is designed to operate within the Windows environment and operates best on a PC using Windows XP/ME/2000/NT/98/95(version B SP1 or better). Hardware requirements - IBM compatible PC. Equivalent of 64Mb Pentium 166 or better, running a screen resolution of 640 x 480 or better. A pointing device such as a mouse or trackball. A 9/25 pin serial communications port is needed for loading of Vysta programs.

Supporting software required - Internet Explorer version 4.01 SP1 or better. A PDF reader (ie. Adobe Acrobat 3.0 or better).

It is strongly recommended that the 16 bit colour setting is used.

The Vysta Platform

A Vysta program has two main components; the Screen List and the Schematic.



When a new Vysta program is created or an existing program is opened, both the Screen List and the Schematic are opened by Vysta as separate windows in the Vysta platform.

A Title Block is also available to record applicable program details.

Moving around the Schematic can be accomplished by using the mouse right click menu, toolbar or keyboard.

Moving around the Screen List can also be accomplished by using the mouse right click menu, toolbar or keyboard although there are no zoom functions available.

The mouse right click menu, toolbar or keyboard navigating functions will be described in detail in the Vysta online help and only a brief outline of the drop-down menu functions is given here.

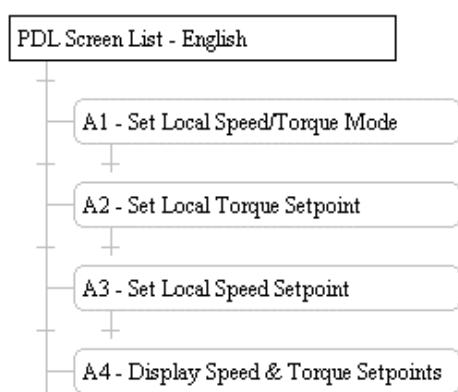
Drop-Down Menu Functions

- **File**
Provides commands for creating, opening and saving netlist files, printing and page formatting.

-
- **Edit**
Provides the typical Cut/Copy/Paste/Delete/Undo commands. Also provides a Find function.
 - **View**
Allows manipulation of the View State of the current window.
 - **Insert**
Allows a new Screen List to be created and also provides a Floating Text function that allows remarks to be inserted into the Schematic.
 - **Compile**
Compiles the Vysta netlist into a Visual Language Object file (.vlo) ready for loading into the motor controller and also displays any error messages associated with the compilation process.
 - **Tools**
Provides a Screen List or Schematic report to be generated and also selects colour or monochrome display mode and the toolbar display mode.
 - **Window**
Arranges opened Windows.
 - **Help**
Contains the Vysta on line Help file.

The Screen List

The motor controller's Display Unit is a two line LCD display with a width of 16 characters. The Vysta programming software is used to design a menu system for the display in a similar format to the standard Screen List. The menu system has a tree structure. Each branch shown in the Screen List view window stores a set of information to be displayed on the Display Unit. The information can be for indication/display only, or parameters can be entered. By using the Display Unit buttons, the various screens, and screen branches may be accessed and displayed.



Each screen may be configured to use either both lines of the two-line Display Unit or just one line. If a screen only uses one line, the Status screen will be displayed on the other line. The Status screen is the screen that appears at the top of the Screen List tree in the Screen List window.

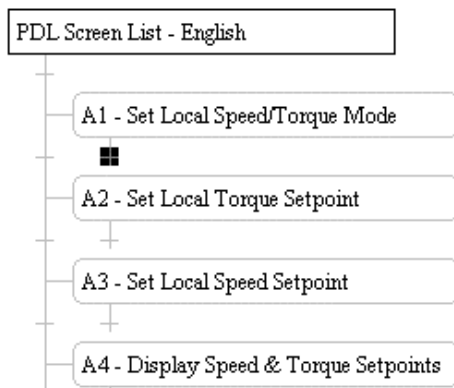
Screens may be used to display and manipulate either System variables or Vysta variables. System variables are the motor control related variables that exist in the motor controller's System Code. We may read them and may change the value of some of them, but we cannot create new ones, nor alter their parameters. Vysta variables are variables that can be created under the Vysta platform. We are able to create them, delete them and change their parameters to suit our needs.

It is good practise to create a Screen List for a Vysta program by modifying an existing default Screen List. The netlist containing the default Screen List is of the form **0410aaXX.nl** for the E-Series, where **XX** refers to the software release version. With this method a minimum of effort is required to develop a customised Screen List while still maintaining many of the useful characteristics of the motor controller's standard program screen structure.

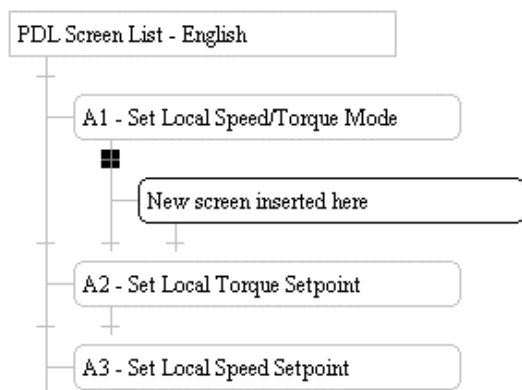
Note: You should ensure that the default Screen List that you start with is correct for the software version of motor controller that you are going to load the Vysta program into. Refer to the Screen List update table that outlines Screen List changes between software revisions.

Adding a Screen

To add a screen to the Screen List, the mouse pointer is positioned on one of the horizontal tabs below a screen at the position the new screen is to be placed.



When the mouse pointer is on the tab, it indicates correct position by highlighting a small box. Click the left mouse button, drag the pointer down and a new screen is inserted. Multiple screens can be inserted by continuing to drag the mouse pointer downwards.



An **Undo Insert Screens** function is available under the **Edit** pull down menu.

In the example above, the newly inserted screen is highlighted. This highlight indicates this screen is selected for editing. Positioning the mouse pointer on the screen location and clicking the left mouse button or scrolling with the up/down arrow keys can highlight any screen.

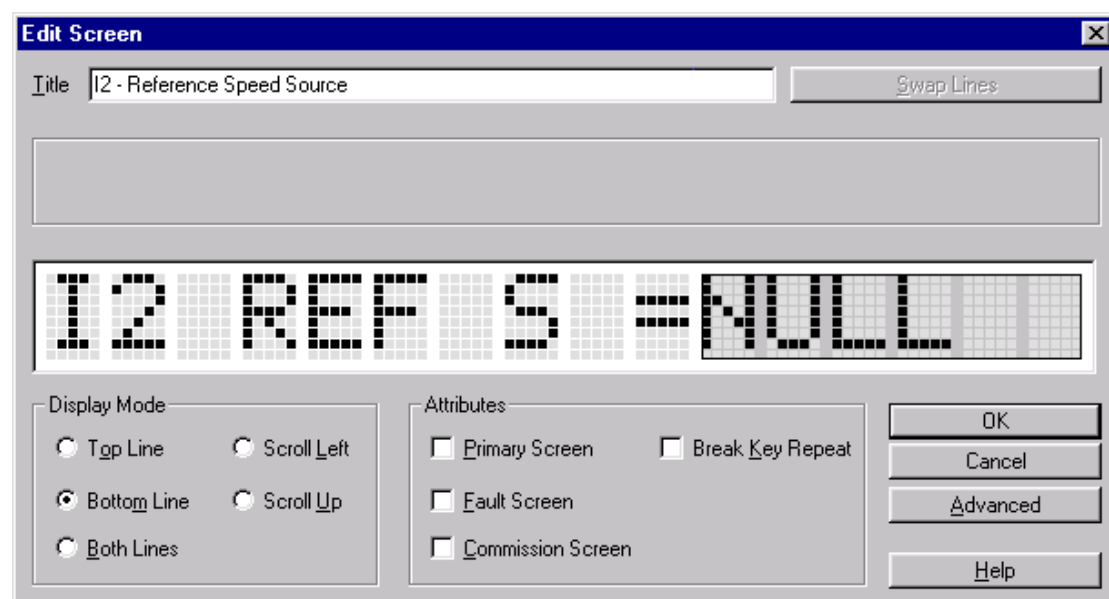
Multiple screens can be highlighted using the shift button function. This feature is used for copying multiple screens.

Edit Screen Dialog Box

To display the Edit Screen dialog box, double click the left mouse button while the mouse pointer is on the screen to be edited. Pressing the enter key will also display the Edit Screen dialog box for the highlighted screen.

The **Title** field contains text that appears on the Screen List tree and printout. This is the descriptive name of the screen and should be unique to that particular screen.

The **Text** field contains the text that appears on the Display Unit. A highlighted region indicates a variable has been inserted into this field. Text characters are displayed as they appear in the text field.



Display Mode sub-block

The Display Mode parameter determines which line or lines the screen text will be displayed on, and how a modifiable enumerated variable is displayed.

- **Top Line**
The screen text will be displayed on the top line of the Display Unit and the status screen will be displayed on the bottom line.
- **Bottom Line**
The screen text will be displayed on the bottom line of the Display Unit and the status screen will be displayed on the top line. This is the default display mode.
- **Both Lines**
The screen text will be displayed on both lines of the Display Unit and the status screen will not be displayed.
- **Scroll Left**
The screen text will be displayed on the bottom line of the Display Unit with the status screen on the top line. When modifying an enumerated

variable on this screen, the displayed text will be shifted to the left to display the full enumeration text.

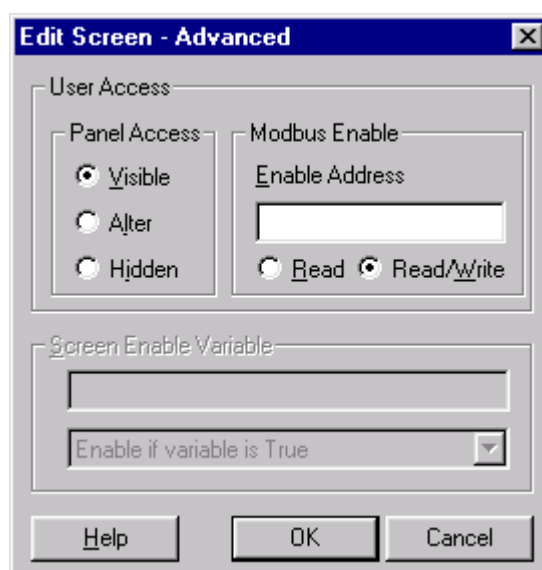
- **Scroll Up**
The screen text will be displayed on the bottom line of the Display Unit with the status screen on the top line. When modifying an enumerated variable on this screen, the displayed text will be shifted to the top line and the full enumeration text will be displayed on the bottom line.

Attributes sub-block

The screen attributes allow for the following options.

- **Primary Screen**
The primary screen is the power-on screen. On power-on the program searches for and displays the screen with the Primary Screen attribute set. Only one screen per Screen List should have this attribute set.
- **Fault Screen**
The Display Unit jumps to the fault screen when a fault occurs. Only one screen per Screen List is allowed to have this attribute set.
- **Break Key Repeat**
If this attribute is set then the automatic key repeat is stopped when this screen is displayed. This causes a delay to occur if scrolling quickly through the Screen List.
- **Commission Screen**
The screen with this attribute set contains the password variable and is the only screen that may contain two modifiable variables. The first modifiable variable is the commission mode flag and the second is the password variable. Because the variables on this screen have special significance, it is best to copy this screen from the default Screen List.

Edit Screen - Advanced Dialog Box



The **Panel Access** selections set the default screen access attribute that applies when the motor controller is not in commission mode. When the motor controller is in commission mode, all of the screens are Read / Write (Alter).

- **Visible**
The screen is visible but any modifiable variable on that screen cannot be changed from the Display Unit.
- **Alter**
The screen is visible and any modifiable variable on that screen may be changed from the Display Unit.
- **Hidden**
The screen does not appear on the Display Unit.

Note: This screen access attribute may be altered by the User by accessing the 'Z' screens Menu Setup Mode as described in the E-Series Technical Manual (Document 4201-180), section 7.3.3. The screen access attribute may also be altered via serial communications as described below.

The **Modbus Enable** selection determines whether the screen display attribute can be modified by serial communications.

- **Read**
Allows the screen access attribute to be read via Modbus using the assigned Modbus address.
- **Read/Write**
Allows the screen access attribute to be read and altered via Modbus using the assigned Modbus address.
- **Unavailable**
Prevents the screen access attribute from being read or altered via Modbus. Not assigning a Modbus address will always make this function unavailable, regardless of the radio button selections.

The **Screen Enable Variable** selection is an RVSx specific function. Please contact Bardac Drives if and RVSx compatible Vysta program is required.

Edit Variable Reference Dialog Box

An existing variable reference in the screen may be edited by double-clicking on the highlighted variable. The Edit Variable Reference dialog box will then appear. The dialog box differs dependant on the **Type** of variable display, Numerical, Enumerated or Bargraph.

Numerical Type

The **Variable** text field specifies which variable is being referred to. A screen may display multiple variables, however only one of the variables may be modifiable. System variables are always identified by **System.** prefixing the

variable name. Vysta variables are identified only by the names assigned to them.

Width determines the number of character places used for displaying the variable. This width includes the sign, the decimal point and any text displayed as an enumeration.

The **Modifiable** check box determines whether the Display Unit is just for displaying the variable or also for changing the value of the variable. The following functions only apply if the Modifiable check box is selected...

- **Stop on Zero**
When changing the numerical value of the variable from the Display Unit, it will pause at zero.
- **Min/Max**
These are the limits (after the scaling) to which the variable can be modified. These limits may be a number or the name of another variable
- **Wrap Around**
When the scaled variable reaches the maximum limit, and it is increased further, the Wrap Around function will cause that variable to wrap around to its minimum value. If the Wrap Around option is not checked, attempting to increase the variable above the maximum limit will result in the variable remaining at the maximum value. This applies to the

minimum limit also. The wrap around function is mainly used with enumerated types.

- **Continuous**

To modify a variable from the Display Unit, the * button must be pressed. The + and - buttons are then used to change the value of the variable. When changing the value of the variable, the value of the variable will either be updated continuously or will hold it's present value until the * button is released. The Continuous function would not normally be used with enumerated types.

- **Stop to Modify**

The motor must be stopped (Status =OFF) before the variable can be adjusted from the Display Unit.

NOTE: If a Vysta variable has been designated as Stop to Modify at the function block level (in the Schematic) then the screen Stop to Modify parameter setting will not override this.

Type selects how the variable will be displayed. The options, *Binary*, *Decimal* and *Hexadecimal*, allow selection of the number system to be used if the value of the scaled variable is to be displayed or modified.

Places specifies the number of digits displayed after the decimal point.

The **Scaling** factor multiplies the actual value of the variable for display on the Screen. This enables a per-unit (normalised) System variable to be displayed as a percent.

To display a normalised System variable in percent, the Scaling factor should be 100. To display a de-normalised System Variable, the Scaling factor should be 1. To display Vysta variables and other values, set the Scaling factor to suit.

If the **Signed** box is selected, the variable will be preceded by a + if it has a positive value. The - sign will always be displayed if the variable has a negative value, irrespective of whether this box is checked. Note; the sign is included in the variable **Width** allocation and takes up one character space.

The **Normalise** option influences the way some real (scalar) System variables are displayed. A normalised System variable has a per-unit value (0-1 represents 0-100%) normalised to the Base value for that variable (e.g. normalised RPM for a 1500RPM motor = 1 at 1500RPM).

As an example, the E-Series Status screen on the top line of the Display Unit indicates Output Speed as a normalised value (with a scale factor of 100 to provide a display in %). In screen A5 the Output Speed is displayed as a de-normalised value, with scale factor of 1, and indicates RPM.

A **Special Value** can be assigned an enumeration when the **Numerical Type** is **Decimal**. The numerical value will be displayed except when the value equals the special value. The associated enumeration will then be displayed. The Engineering Units normally displayed after the variable can be configured

as a **Conditional Character** and therefore will not be displayed if the enumeration is presently being displayed.

Enumerated Type

| Number | Enumeration |
|--------|-------------|
| 0 | NULL |
| 1 | AIN1 |
| 2 | AIN2 |
| 3 | AIN1+2 |
| 4 | FIBRE |
| 5 | LOCAL |
| 6 | MREF |
| 7 | MTRPOT |
| 8 | PROCESS |

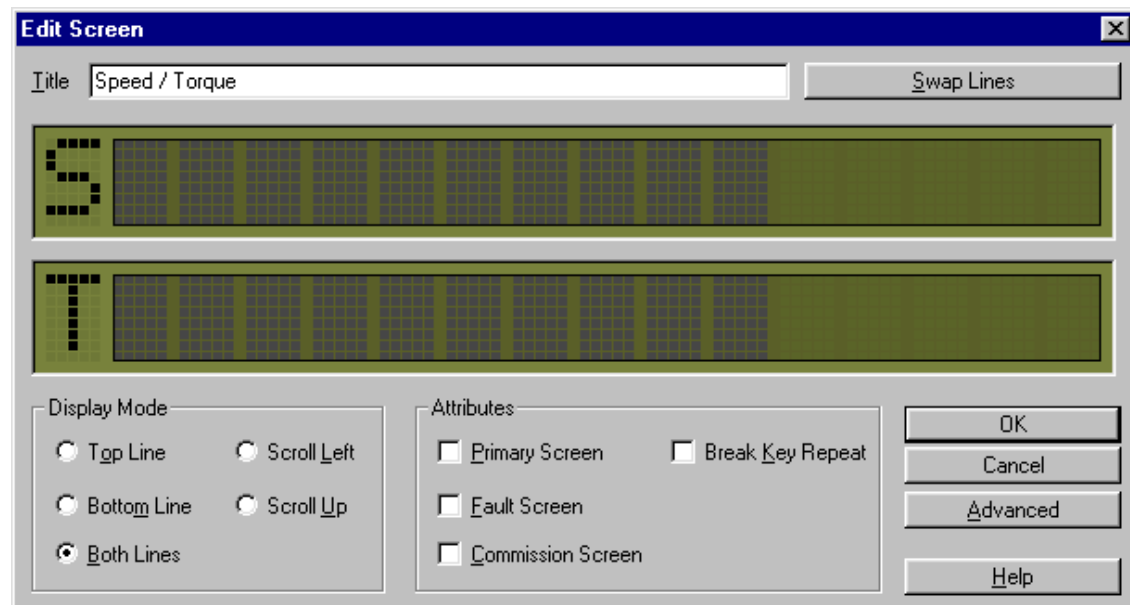
Width specifies how many letters wide the displayed character string will be if the variable is not presently being modified. **Expanded Width** specifies how many letters wide the character string can be for a modifiable variable while it is being modified from the Display Unit. If this width exceeds the available display space, the scroll left or scroll up function should be selected in the Edit Screen dialog box.

As an example, the E-Series Output Relay 1 Source screen enumeration is '10 AT SET SPEED' when the variable **system.relay_1_select** has a value of 10. The Width is 2 and therefore the screen will display 'O2a RELAY 1=10'. The 10 is first two characters of the enumeration.

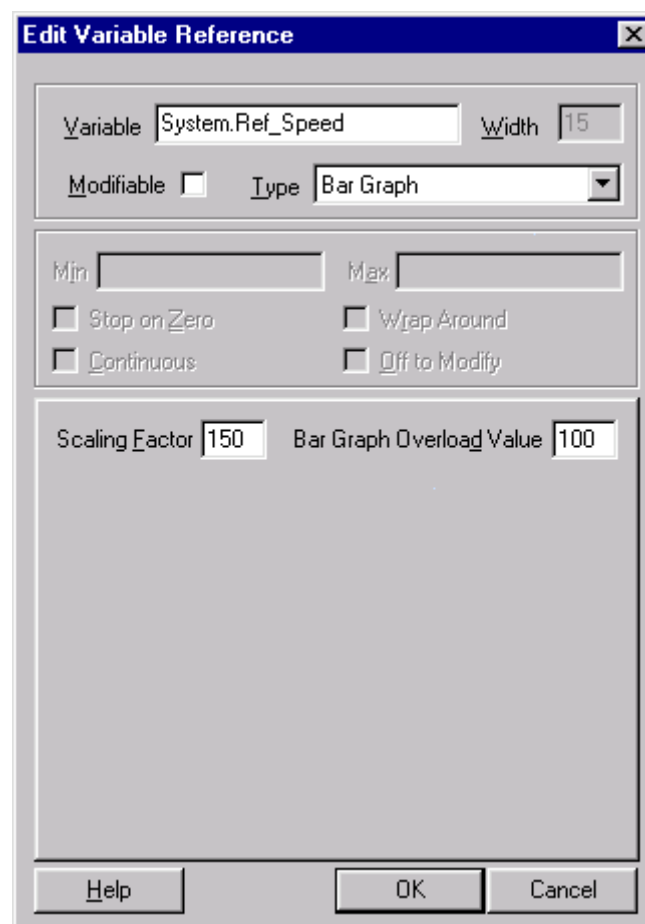
The **Expanded Width** is 16, therefore pushing the * Button on the Display Unit (and hence modifying the variable) will change the screen display to '10 AT SET SPEED'. This particular screen is designated as Scroll left and this means the expanded enumeration pushes the normally displayed text to the left leaving only the enumeration visible.

Bar Graph Type

This allows the value of the variable to be displayed as a bar graph.



A Bar Graph always occupies 15 of the 16 characters on the display and must always start on the second character in the screen text. The left most character space is reserved for an alphanumeric tag that can be used to identify the variable being displayed.



The bar graph full-scale value (after the scaling factor is applied) is 150. Therefore as the chosen variable's value reaches 150, the bar reaches the right hand side of the display.

The **Scaling Factor** should be set so that the variable's maximum value (after the scaling) equals 150.

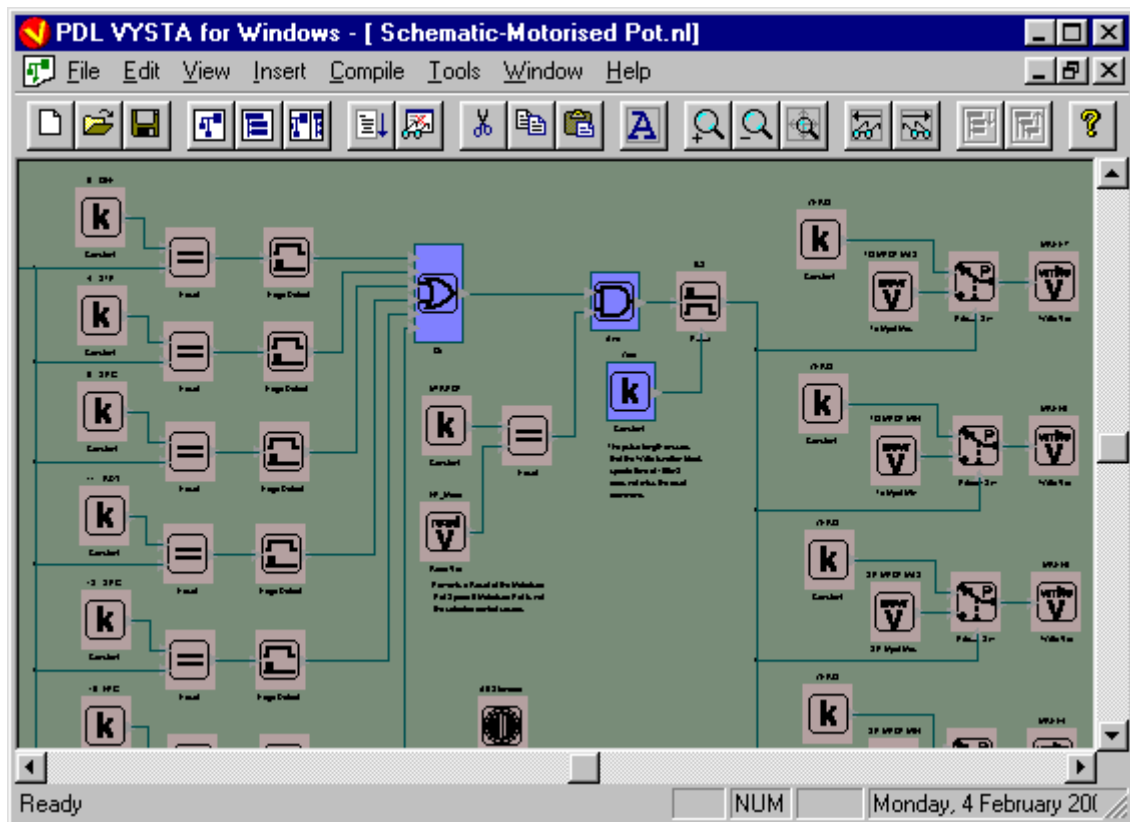
The standard bar is displayed 6 pixels thick (from the bottom of the display to the top). The section of the bar that represents values (after scaling) greater than the **Bar Graph Overload Value** utilise the full 8 pixels of character height.

The Schematic Editor

The Schematic Editor enables a function block based control configuration to be assembled. Function blocks are selected from Vysta's Oval Menu and interconnected using click and drag techniques.

Each Vysta program may only have one Schematic associated with it.

A wide range of standard logic and process control function blocks are available such as PID controllers, comparators, logic gates and arithmetic functions etc. A function block may be selected from the Oval Menu multiple times. Each function block has its own configuration dialog box(es) for the various parameters associated with that function block. Standard E and RVSx Series system variables can be accessed, and custom "Vysta variables" can be created for control purposes. Variable values can be displayed and entered via the motor controller's Display Unit and linked from the Screen List to the Schematic by the variable name.



The wire connections between blocks are organised into orthogonal lines. Various editing and zoom functions are available within the Schematic editor.

When writing a Vysta program that will run on a E-Series AC motor controller, the Schematic Program is required to interface with some of the standard motor controller functions. To ensure that no contention occurs between the Vysta program and the AC motor controller's System Code standard functions, there are some programming requirements dependant on which type of E-Series AC

motor controller is being utilised.

For the E-Series, the Standard Program function block is used once in each Vysta program's Schematic to select the control source for various E-Series functions. Incorrect or unpredictable operation of the E-Series may result if the Standard Program function block is either not used or configured incorrectly.

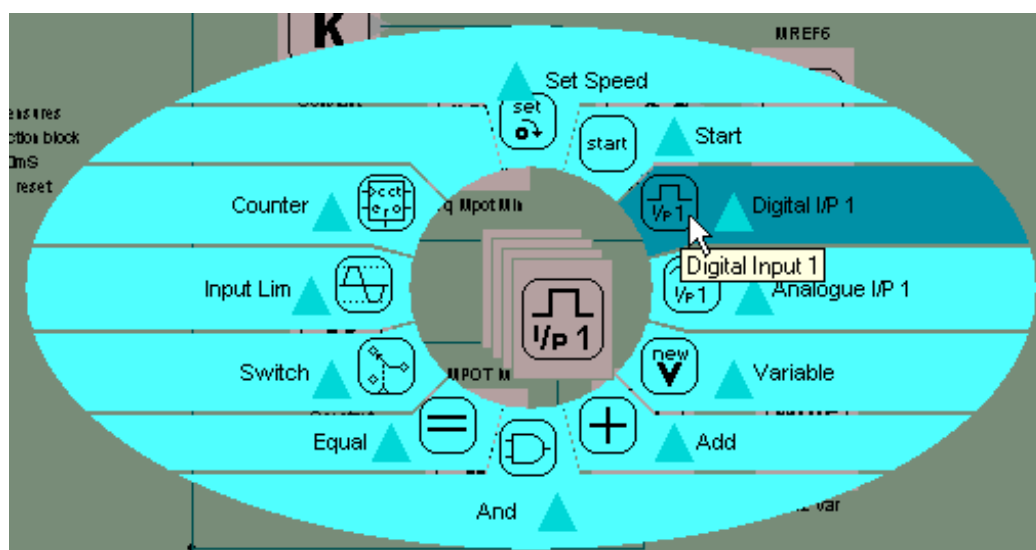
For the RVSx Series, various function blocks take precedence over some of the System Code's standard functions, but most of the System Code's standard functions are still active and can influence the response of the RVSx Series.

The Oval Menu

The Oval Menu can be used to place function blocks onto the Schematic. The Oval Menu can be invoked by double left-clicking on a blank section of the Schematic or by pressing the space bar while the Schematic has the focus. If the Oval Menu is invoked using the mouse, then it will first appear at the point of insertion. This is where the cursor was when the Oval Menu was invoked using the left mouse button.

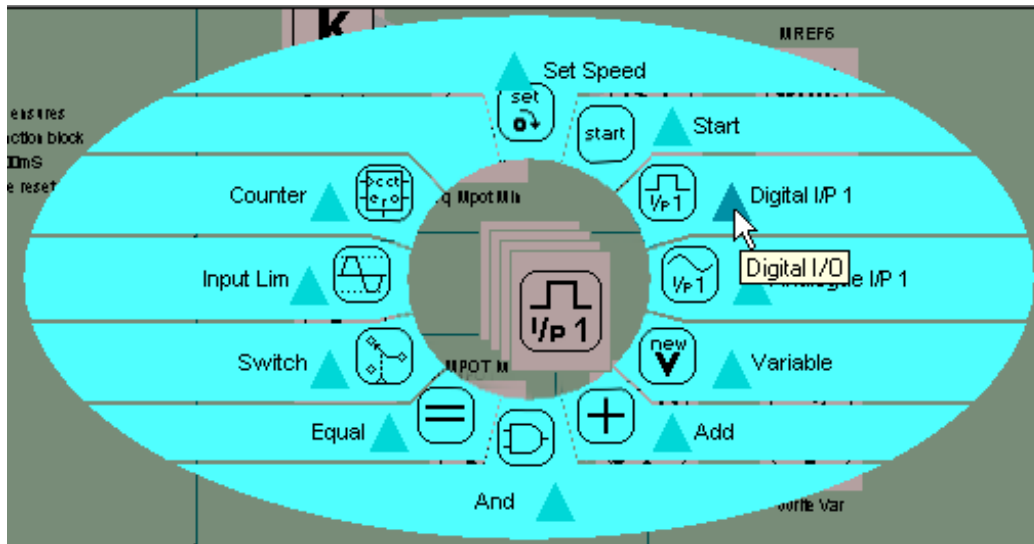
If the space bar is used to invoke the Oval Menu then it will appear in the centre of the Schematic.

The set of function blocks that appears in the Oval Menu is dependant on the type of E-Series AC motor controller that the Vysta program is being developed for, but in general the Oval Menu is organised into a hierarchically layered sub-menu structure. Each sub-menu is divided into a number of "function block segments" as can be seen below. The currently selected function block segment is shown in a darker colour. If the mouse cursor is left hovering over a function block for a short time a tool-tip will pop up describing the function block in question.

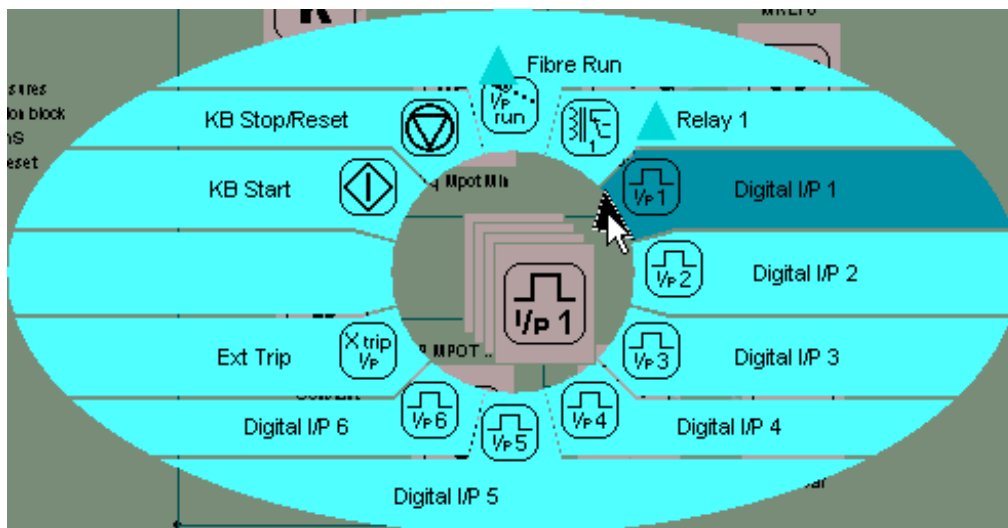


Segments that have sub-menus are indicated by a triangle within the segment. When the Oval Menu is first invoked the top layer of the menu structure will be

presented. Deeper layers of the Oval Menu may be accessed by clicking on a sub-menu triangle or by moving the selection to the required function block and pressing the down arrow key. The function block that appears alongside a sub-menu arrow is representative of the types of function blocks contained in that sub-menu. By hovering over a sub-menu arrow a tool-tip will pop up describing the sub-menu.



Each sub-menu has a “previous menu” arrow that can be used to return to the previous layer in the menu structure. The “previous menu” arrow is always attached to the function block that represents the sub-menu in the sub-menu layer above. In the example below, the representative function block is the Digital Input 1 function block.



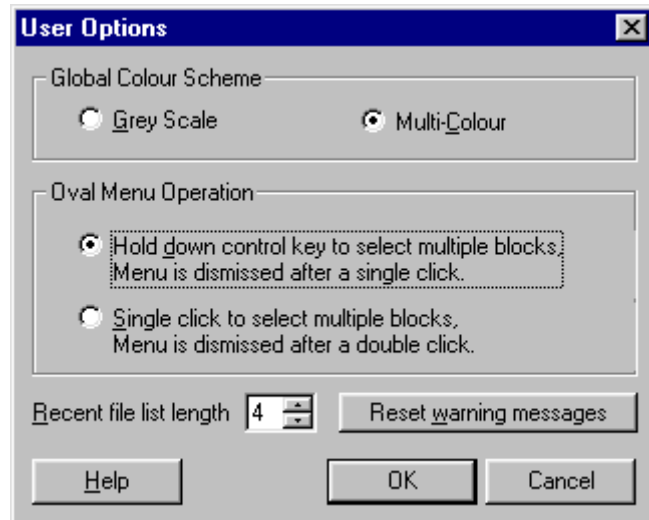
It is possible to select multiple function blocks for insertion onto the Schematic during one invocation of the Oval Menu. When multiple function blocks have been selected for insertion, they are stacked atop each other, each one offset slightly from the one before, as shown above.

Once the Oval Menu has been dismissed, the selection of function blocks can

be dragged to the positions required.

Oval Menu Options

They are two modes of operation for the Oval Menu and these can be selected from the User Options dialog box. The User Options dialog box can be activated from the Tools drop-down menu.



In the first mode of operation, a single or double left-click on a function block in the Oval Menu will immediately cause the Oval Menu to be dismissed and the selected function blocks to be placed in the Schematic. In order to select multiple blocks during one invocation of the Oval Menu, the Ctrl key can be held down while function block selections are made.

In the second mode of operation, a single click on a function block in the Oval Menu will select that function block but will not dismiss the Oval Menu (regardless of whether the Ctrl key is held down). A double left-click on a function block will select that function block, dismiss the Oval Menu and place onto the Schematic, all of the function blocks that were selected during the Oval Menu invocation.

In either mode of operation, holding down the Ctrl key while invoking the Oval Menu will cause the last active sub-menu layer to be shown rather than the top layer of the menu structure.

In either mode of operation the Oval Menu can be fully controlled by the use of the following keyboard keys:

- The space bar can be used to invoke or dismiss the Oval Menu. Using the space bar to dismiss the Oval Menu will cause all of the function blocks selected during the Oval Menu invocation to be placed onto the Schematic at the position where the Oval Menu was first invoked.
- The left and right arrow keys can be used to move the selection around the function blocks in the active sub-menu. The right arrow will move the selection in a clockwise fashion while the left arrow will move the selection in an anti-clockwise fashion.

-
- The up and down arrow keys can be used to move in and out of the sub-menus. The down arrow “drills” down into the menu structure, while the up arrow returns the Oval Menu to the next highest sub-menu in the hierarchy.
 - The return key can be used to add an instance of the currently selected function block to those selected for insertion in the schematic.
 - The escape key can be used to cancel the Oval Menu invocation. When the escape key is used to dismiss the Oval Menu, all function block selections are discarded.

Building a Schematic

Selected function blocks are depicted by a bold perimeter of the block. When a block is highlighted in this manner, a click and drag of the left mouse button can reposition it. The mouse determines the position of the block. If any connections are fixed to the function block, they will be dragged with the block. A double click of the left mouse button opens the function block’s parameter dialog box for editing of parameters.

- **Linking Function Blocks**

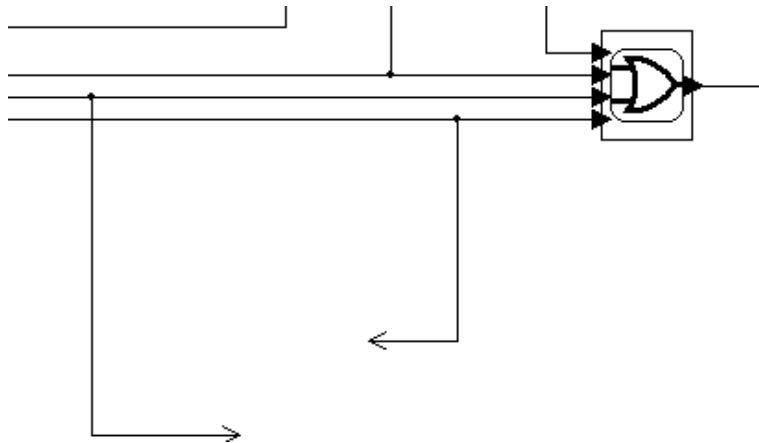
Function blocks can be connected together with wires by clicking and dragging (with the left mouse button held down) from one function block to another. E.g. A connection could be made by clicking on the output terminal of a function block (depicted by a small triangle on the function block which is highlighted when selected) and dragging to the input terminal of another function block.

All the wire connections are automatically adjusted between function blocks into orthogonal lines (e.g. all lines are at an angle of 90° or 180°).

An illegal connection will not latch (e.g. multiple outputs to a single input).

- **Open-ended link**

A wire may be drawn from a function block or wire junction, to terminate without necessarily connecting to another function block. Once the wire is drawn it is terminated with an arrowhead. Other valid connections can be linked to the terminated wire. When a valid link occurs the arrowhead disappears. When an arrowhead is highlighted for editing, it appears as a solid arrowhead.

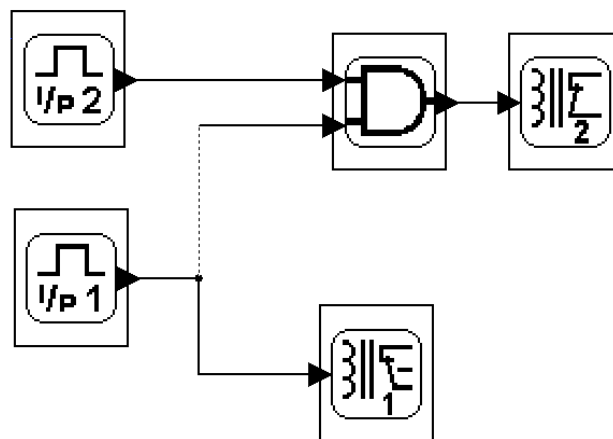


- **Wire Junction**

A branch or junction may be made along a wire. A valid junction is depicted by a dot. A junction can be produced by either clicking on a function block input or output and dragging with the left mouse button. Alternatively, a junction can be produced by positioning the cursor over a wire and dragging with the right mouse button.

- **Deleting Lines**

Selected wire segments can be deleted with the delete key or by the cut function (Ctrl + X or Shift + Delete). The cut function will insert the elements into the clipboard to be pasted elsewhere before deleting them.



Wire segments can be selected by creating an Edit Window around the wire segment, or by placing the cursor on the wire segment and left-clicking.

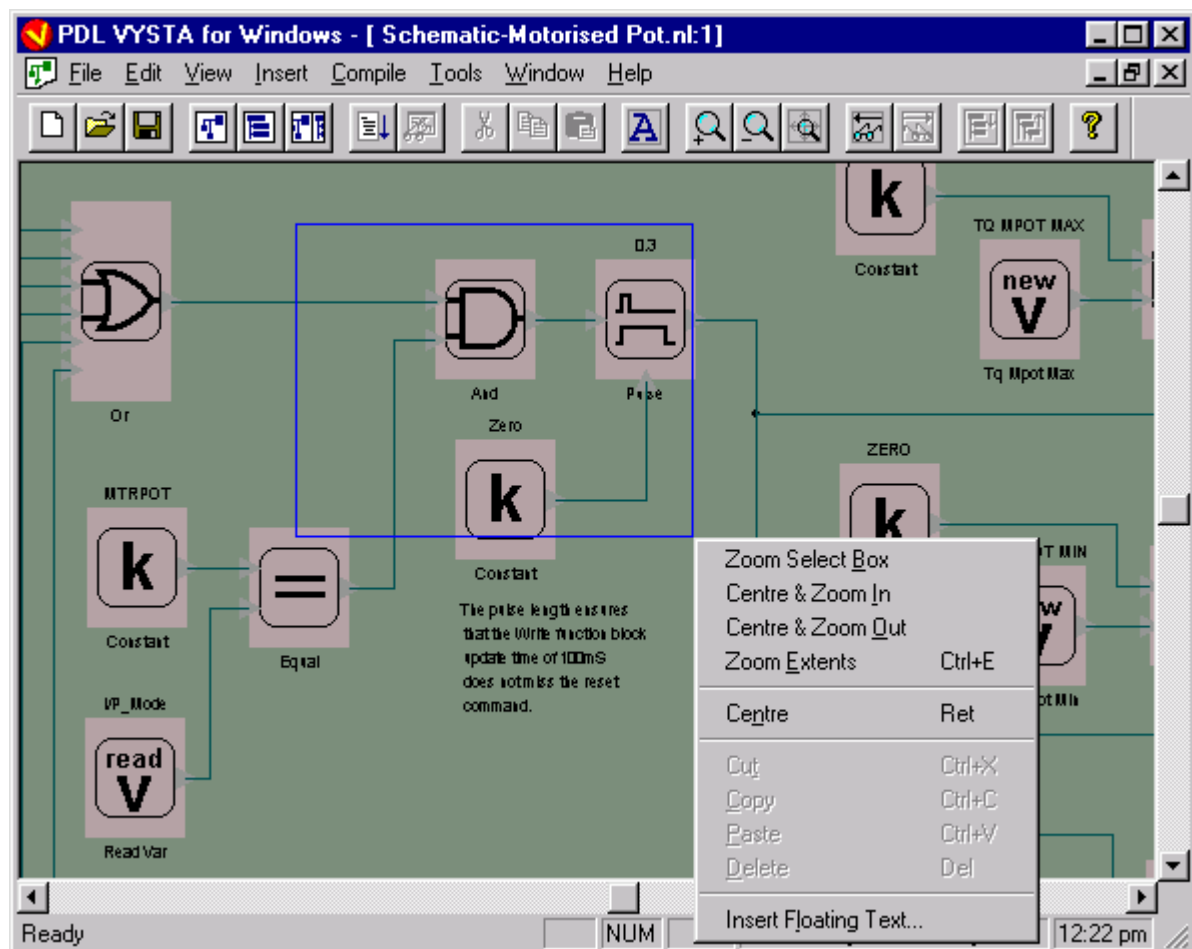
Vysta will maintain that all wires in the Schematic are connected at no less than one end.

Edit / Zoom Window

Selection of multiple wire segments, function blocks and floating text can be achieved by using a left-click and drag action to extend a selection box over any number of elements to be selected. This creates an Edit Window. No elements will be selected until the mouse button is released.

If the right mouse button is used during the drag operation as shown below, when the button is released, the user will be presented with the usual “right-click” context menu.

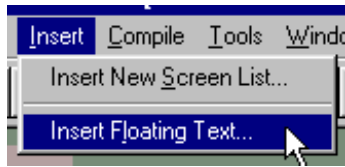
When a wire segment, function block or floating text is selected it is shown in a more prominent style and colour. The joins between segments are more prominently displayed as well. Once a set of elements has been selected, the usual set of edit operations can be performed on that set, such as move, cut, copy, delete etc.



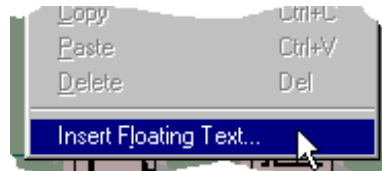
Selection of a single wire segment, function block or floating text can be achieved by single left clicking on that segment.

Floating Text

The “Insert Floating Text” command, accessible from the “Insert” drop-down menu or from the Schematic context menu, activates the “Edit Floating Text” dialog box. Text entered into the “Edit Floating Text” dialog box is inserted into the Schematic as a text comment.



Drop-down menu



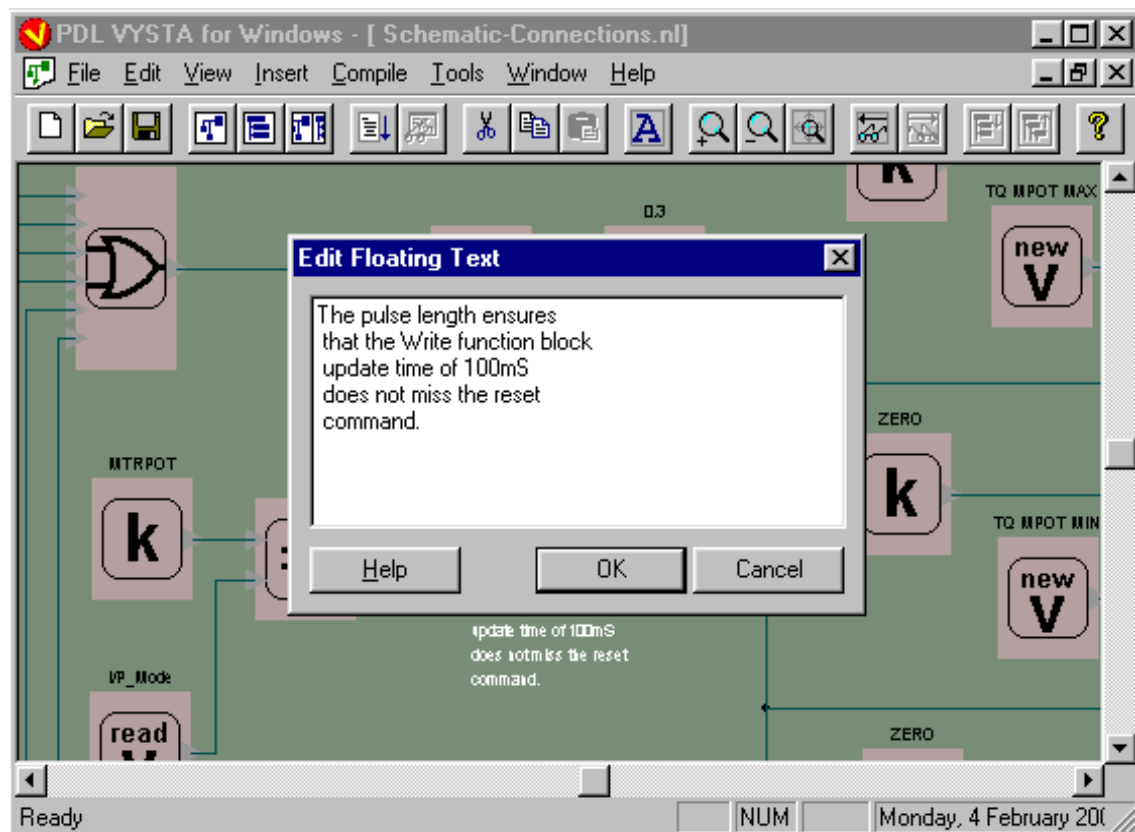
Context menu

Text inserted by means of the “Insert” drop-down menu is placed at the top left of the Schematic window when the dialog box is closed and can then be dragged to the desired location.

Text inserted by way of the Schematic context menu first appears at the point of insertion. This is where the cursor was when the context menu was invoked using the right mouse button.

Floating text can be selected and manipulated in the same way as the other Schematic elements such as function blocks and wires (cut, copy, move, delete etc.).

Existing floating text can be edited at any time using the “Edit Floating Text” dialog box, which can be activated by a double left-click over the floating text to be edited.



Variables

There are two classes of variables that can exist in a Vysta compatible E-Series AC motor controller.

System variables exist in the E-Series AC motor controller's System Code. These variables are used by the System Code to carry out the core functions of the E-Series AC motor controller. System variables are identified by the **system.** prefix.

The Vysta programmer cannot create new System variables, nor change their characteristics (eg, their default values), however, System variables may be read in the Vysta program and some may be written to by the Vysta program. When writing to a System variable from the Vysta program, the writing will only occur if the motor controller's System Code allows it to take place. The main constraints are:

- The System variable is modifiable from the Screen List, and therefore cannot also be modified from a Write function block.
- The System Variable is a **Stop To Modify** variable and will not allow its value to be changed if the motor controller is running.
- In the E-Series, the Standard Program function block causes the value of some System variables to be ignored by the System Code.

The two data types of System variables are Flag and Real.

A Flag is used for Boolean type logic and will hold a value of either **0** or **1**. A Real is a scalar and the range of values each can hold will be detailed in the relevant System variable Data Table. When a Real System variable is read in the Vysta Schematic, it will be converted to a floating-point format. When a Real System variable is written to from the Vysta Schematic, it will be automatically converted from floating-point to the format required by that System variable.

Vysta variables are variables that have been created by the Vysta programmer in the Vysta Schematic. The Vysta programmer can assign the variable's characteristics and must assign a unique name to the variable (but the name must not include the prefix **system.**)

A Vysta variable can be created by one of the following methods:

- By selecting a New Variable function block from the Oval Menu.
- By selecting any function block from the Oval Menu that has a **parameter** associated with it.
- By selecting any function block from the Oval Menu that has an output connection. This **Output variable** is a read-only variable.

The two data types of Vysta variables are Flag and Real and both types are initialised as User Parameters.

A Flag is used for Boolean type logic and will hold a value of either **0** or **1**. A Real is a scalar and is a floating-point (as defined by IEEE-754). A Vysta Real can hold a value of +/- 3.4 x 10E38.

New Variable Function Block



The New Variable function block by the Vysta programmer to create a new Vysta variable in the Vysta Schematic. The Vysta programmer can assign the variable's characteristics and must assign a unique name to the variable (but the name must not include the prefix **system**.)

The format of the variable is then **function block name**. In the example below, the function block is called **Filter 1** and the format of the variable will be **Filter 1**.

The two data types of Vysta variables are Flag and Real and both types are initialised as User Parameters.

A Flag is used for Boolean type logic and will hold a value of either **0** or **1**.

A Real is a scalar and is a floating-point (as defined by IEEE-754). A Vysta Real can hold a value of +/- 3.4 x 10E38.

The screenshot shows a dialog box titled "Variable" with the following fields and controls:

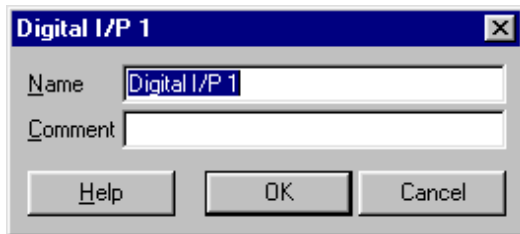
- Name:** Filter 1
- Comment:** Filter used during acceleration
- Default:** 2.7 (with an "Edit Parameters..." button next to it)
- Type:** Real (dropdown menu)
- Buttons:** Help, OK, Cancel

The variable can be given a default value as shown above. This is the value that the variable will have at turn-on unless another value has been saved to EEPROM. It is also the value that the variable will have after the E-Series AC motor controller has been initialised using the User Parameters initialising option.

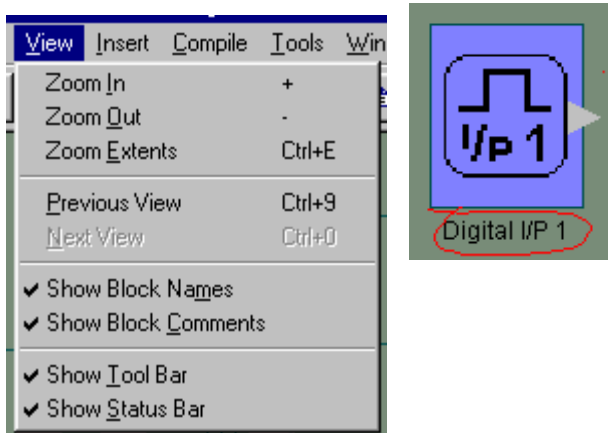
The Vysta programmer can also assign the variable's characteristics in the **Edit Parameters** dialog box.

Configuring Function Blocks

The text entered into the **Name** field of the function block dialog box becomes the name of the function block.



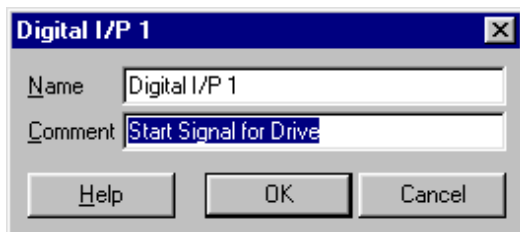
This field will appear in the schematic below the function block if “Show Block Names” is checked in the Schematic “View” menu.



Each function block should have a unique name. This unique name is used to address any parameters associated with the block. These parameters will have the form **function block name.parameter name**. The output of the function block is a parameter and has the form **function block name.output**.

If each block does not have a unique name, the compiler will assign a unique name during the compilation process. If a block parameter is called elsewhere in the program and the compiler has assigned a new name to the function block, the block parameter will no longer be found.

The **Comment** field allows for a text comment to be associated with a function block.



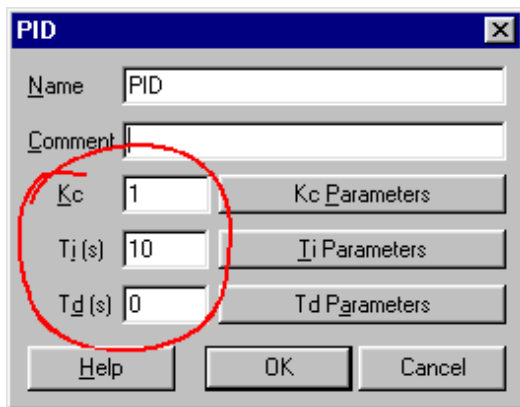
Block comments will appear in the schematic above their associated function block if the “Show Block Comments” option is checked in the Schematic “View” menu.

Function Block Parameters

Some function blocks have parameters associated with them that may be used to modify the output characteristics of the function block.

Function block parameters are Vysta variables that have been created by the Vysta programmer in the Vysta Schematic by selecting a function block from the Oval Menu that has a **parameter** associated with it.

The Vysta programmer must assign a unique name to the function block (but the name must not include the prefix **system.**). The format of the parameter is then **function block name.parameter name**. In the example below, the function block is called **PID** and the format of the Kc parameter will be **PID.Kc**



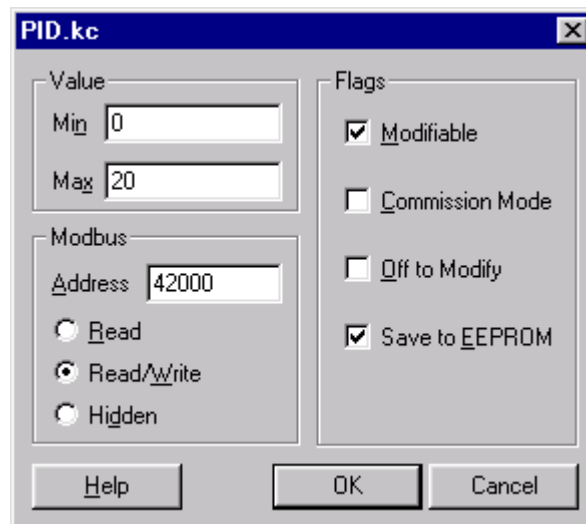
The parameter can be given a default value as shown above. This is the value that the parameter will have at turn-on unless another value has been saved to EEPROM. It is also the value that the parameter will have after the E-Series AC motor controller has been initialised using the User Parameters initialising option.

The Vysta programmer can also assign the variable's characteristics in the **Edit Parameters dialog box** and this can be done in the same fashion as the New Variable function block.

Any engineering units associated with the parameter are shown alongside the parameter's name.

Edit Parameters Dialog Box

Clicking on the **Edit Parameters..** button will cause the Edit Parameters Dialog Box to appear.



Value sub-block

- **Min/Max**

The Min and Max text fields specify the minimum and maximum values of data the variable will accept. The values entered into these text fields can only be numeric values.

If either of the text fields is left empty, then the value of the variable will only be limited to $\pm 3.4 \times 10^{E38}$.

Flags sub-block

- **Modifiable**

The Modifiable check box determines if the variable may be modified either from the screen, via Modbus Serial comms, or by a Write function block within the Schematic.

If the Modifiable check box is not checked then the variable will be a constant at the default value.

- **Commission Mode**

The Commission Mode check box determines if the modifiable variable may be modified *only* while the drive is in commission mode.

This restriction applies only if the modifiable variable is being modified from the screen, or by a Write function block within the schematic. This restriction is ignored if the modifiable variable is being modified via Modbus Serial comms.

- **Off to Modify**

The Off to Modify check box determines if the modifiable variable may be modified *only* while the motor controller is stopped.

This restriction applies regardless of where the modifiable variable is being modified from, either the screen, by a Write function block within the schematic or via Modbus Serial comms.

- **Save to EEPROM**

The Save to EEPROM check box determines if the last value of the variable entered either from the screen, or via the Clocked Write function block will be saved to EEPROM. This saved value overrides the default value at turn on.

A value changed as the result of a Write function block within the schematic may not be saved to EEPROM.

If the Save to EEPROM check box is not checked then the variable value at turn on will be the default value.

Modbus sub-block

- **Address**

The Address text field allows the variable to be assigned a unique Modbus address. The variable can then be interrogated and or manipulated via Modbus serial comms.

The Modbus address must be a 5 digit decimal no. in the 40000 to 49999 range. It is advisable to use no.'s above 41999 to separate from the E- and RVSx Series System Variable addresses (as listed in appropriate Serial Comms Manual).

- **Read**

The value of the variable may be read via Modbus at the given address, but may not be written to.

- **Read / Write**

The value of the variable may be written to and read via Modbus at the given address. A read will occur with no restrictions, but if the variable is to be written to, the variable must also be checked as Modifiable.

The Off to Modify restriction applies to a write via Serial communications if the Off to Modify flag is checked. Any Commission Mode restriction is ignored.

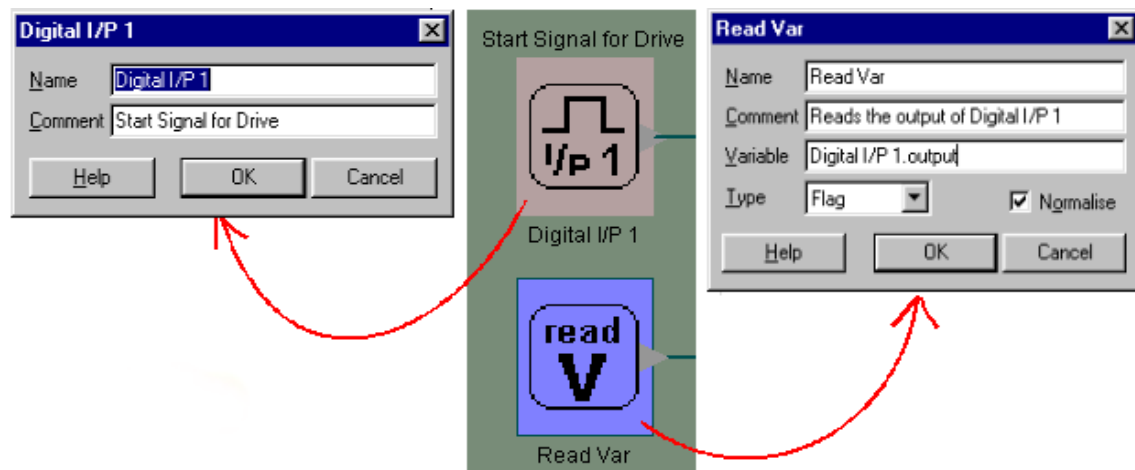
- **Hidden**

The variable is unavailable to Modbus

Output Variable

Any function block that has an output connection also has an associated read-only Output variable. The value of this Output variable can be read from within the Schematic with a Read function block.

It can also be used for display in the Screen List, or assigned a Modbus address and monitored via serial comms.



The format of the Output variable is ***function block name.output*** where the ***function block name*** is the unique **Name** given to the function block in the Block Configuration dialog box. In this example the output of function block **Digital I/P 1** is to be read within the Schematic using a Read function block.

Standard Program Function Block (E-Series only)

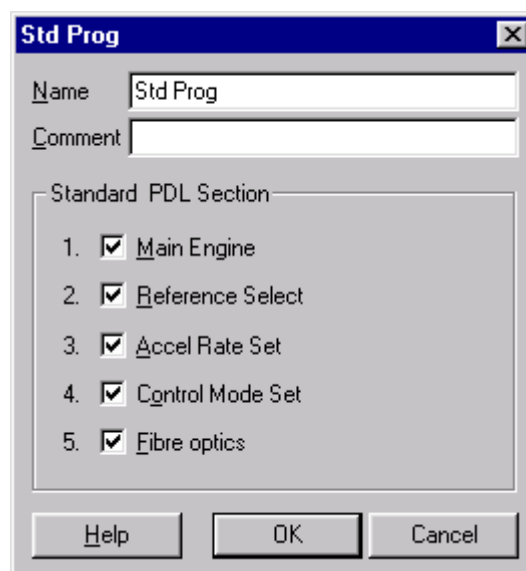


The Standard Program function block defines which of the E-Series functions, as described in the E-Series Technical Manual (4201-180), are controlled by the E-Series System Code, and which of the E-Series functions are controlled by the Vysta program.

This function block **must** be configured in the Vysta program if any of the E-Series functions are to be controlled by the System Code. Each Vysta program may have no more than one Standard Program function block.

The control functions are grouped in four sections, while the fifth section enables the Fibre-optics processor.

Each selection is checked [✓] to enable the E-Series System Code to control the selected function. For example, if the Vysta program just requires a Screen List only, with all of the standard drive functionality retained, all selections within the Standard Program function block must be checked as shown below.



1. Main Engine

Enables / Disables the E-Series System Code control over ...

- Multi-function Input mode selection

-
- Stop mode selection
 - Start/Reset/Run (but not commands via the fibre mode functions or Serial Comms)
 - Motorised Pot, Multi-references and Alt Speed/Torque References

2. Reference Select

Enables / Disables the E-Series System Code control over ...

- Speed and Alternative speed reference
- Torque and Alternative torque reference

Note that the Main Engine influences this function also.

3. Accel Rate Set

Enables / Disables the E-Series System Code control over ...

- Acceleration and Alternative Acceleration rates
- Deceleration and Alternative Deceleration rates

Note that the Main Engine influences this function also.

4. Control Mode Set

Enables / Disables the E-Series System Code control over ...

- Speed / Torque Mode

Note that the Main Engine influences this function also. Therefore checking Control Mode Set will not enable the E-Series System Code control over this function unless Main Engine is also checked.

5. Fibre optics

Enables / Disables the E-Series Fibre optics processor.

Disabling the processor will prevent any Fibre optic functions from occurring, either in the E-Series System Code or in Vysta.

It is normal to always check this option, and if the Fibre optics is to be used then Fibre optics must be checked. However, disabling the Fibre optics will provide more processor speed for complex Vysta programs.

E-Series Series Programming

The E-Series is primarily a motor controller and any programming that is done in Vysta must not inhibit in any way, the ability of the E-Series to control the motor.

To this end it is important that if the standard starting and stopping functions are to be taken over by the Vysta program, a stop command is included in the program. This is also important for any E-Series parameter and all important screens should be included in the Vysta program.

The Vysta platform has a Vysta program that contains the standard Screen List. It is often practical to use this as the base of a new Vysta program rather than start with a new program and then have to build up a Screen List.

The Vysta execution does not function in a similar fashion as a PLC. A typical PLC cycle starts with the Input status being read and then the program is scanned. Following this the Outputs are set. The larger the program, then the longer the scan time. With the nature of the PLC scan it is possible to introduce loops into the program to achieve the desired outcome.

The Vysta program execution cycle occurs at 4mS and each of the function blocks inputs and outputs, including the function blocks representing the E-Series I/O, are updated each cycle. This prevents the output of any function block being allowed to influence its own input.

The Timing and Reading Block Output sections explain this in greater detail.

A Vysta program may often only entail modifications to the Screen List. In this case the Schematic window will only contain the Standard Program function block which will have all of the control options set to the standard program. If the Standard Program function block is left out or not set correctly, the E-Series will not function correctly as a motor controller.

Memory Constraints

The E-Series is primarily a motor controller and most of the processing capabilities are used to this end. This limits the amount of memory that is available for the Vysta program or programs and hence the Vysta program size.

There are two aspects to the Vysta program size. The Flash EEPROM which is used for the program storage, and the Ram that is used to hold the current Vysta program's variables.

DriveLink reports the amount of Flash EEPROM available for program storage and this determines the number of Vysta programs that the E-Series can contain at any time.

The Vysta Screen List is the main factor in determining program size. The larger the Screen List, or the more languages that are attached, the larger the program size.

The standard single language Screen List program size is approximately 22k. The E-Series will typically hold two Vysta programs of this size before running out of Flash EEPROM for storage. The Schematic has minimal effect on the program size.

DriveLink the reports the amount of Ram available, and this is used by the current Vysta program only. Because only the current Vysta program uses the Ram, each Vysta program may use the full RAM available.

The Vysta Schematic is the main factor in determining the Ram requirements for the Vysta program. This is the restriction that will affect most programs. A function block parameter or output that is a Real data type will require 4 bytes of Ram. A function block parameter or output that is a Flag will require 1 byte.

As an example, the PID function block has 1 Real output and 3 Real parameters. This function block therefore requires 16 bytes of Ram.

The Screen List also has an effect on the Ram requirement with every 4 screens needing approximately 1 byte.

Timing

The Vysta program operates with two timing cycles. The bulk of the processing is done at 4mS and selected function blocks operate at 100mS.

The Schematic must be formatted without loops. These will be detected at the compilation stage. The execution of the Vysta program is such that every function block input and output is updated every 4mS. Hence a function block input cannot be dependent on its own output.

The exceptions are the following function blocks that are updated every 100mS:

- Read
- Write (with some exceptions)
- Variable
- Constant
- KYBD Start
- KYBD Stop

The result of interaction between the 100mS and 4mS cycles will be dependent on how the interaction occurs.

As an example of this, the Analogue Input function blocks are updated at 4mS and if one of these function blocks is connected to the speed reference, a change in the input will be reflected to the speed reference at 4mS.

If some arithmetic manipulation is to be carried out on the value of the Analogue Input prior to being connected to the speed reference, care must be

taken to ensure that the response is not adversely affected. For example, a Vysta Variable is added to the Analogue Input. Any change due to the Analogue input will still be reflected to the speed reference at 4mS, BUT any change due to the Vysta variable, will be reflected to the speed reference at 100mS.

Generally program execution is deterministic (e.g. fixed execution time). The minimum execution times for a Vysta program is 4 and 100mS, however this can increase slightly as a function of the program size and complexity.

Reading Function Block Outputs

Typically the output of a function block is connected to the input of another function block using the mouse Click and Drag function. However, the function block output also exists as a read-only variable that may be utilized elsewhere in the program. The output variable can also be given a Modbus address and its value read via serial comms. This will be a Floating-Point format.

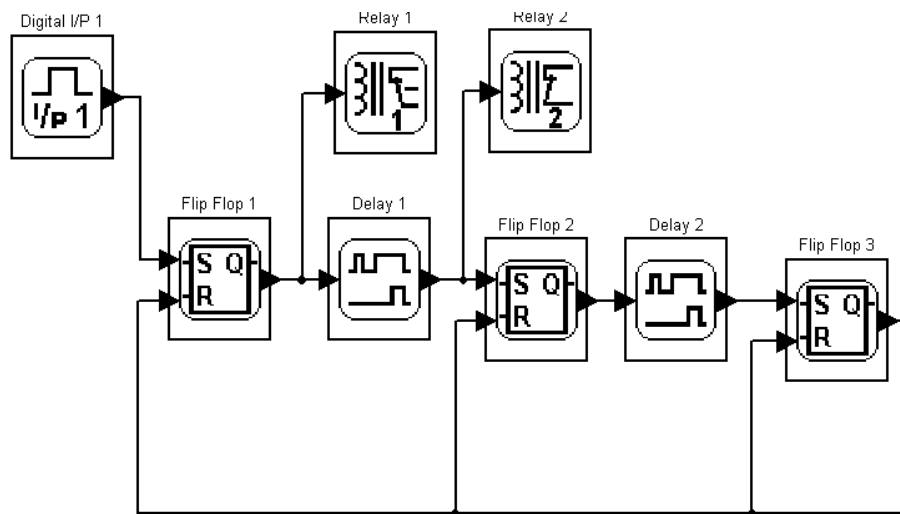
To utilize the output variable elsewhere in the program without using the mouse Click and Drag function, the Read function block is used to read the value of the output variable. The name of the output variable is **function block name.output**.

Note that the Read function block is updated at 100mS, not 4mS, and if timing is an issue then the mouse Click and Drag function should be used to link the function blocks.

The output variable may also be viewed on the Display Unit in the Screen List. If the variable is rapidly changing, it may be best to connect the output of the function block you wish to view to a Low Pass Filter. The output variable of the Low Pass Filter is then viewed on the Display Unit.

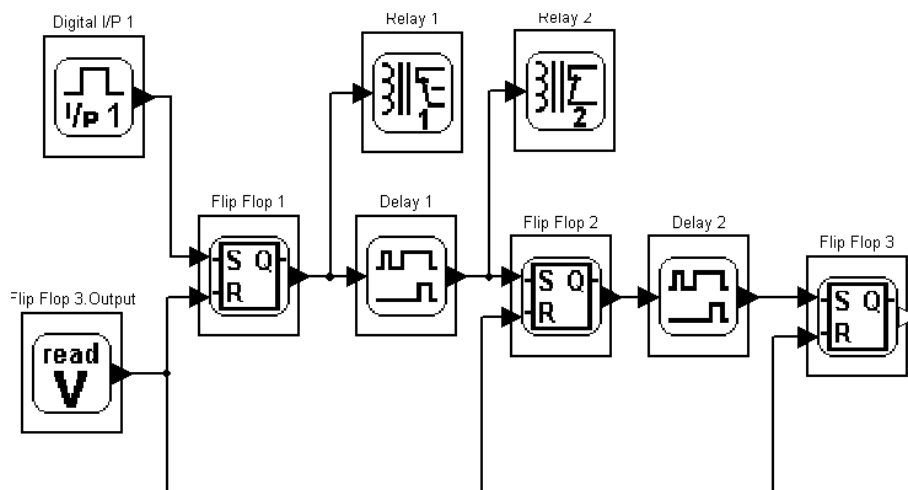
Another reason to read the output variable is the Schematic must be formatted without loops. These will be detected at the compilation stage. The execution of the Vysta program is such that every function block input and output is updated every 4mS. Hence a function block input cannot be dependent on its own output. There will be times when a loop in the logic is required. This cannot be achieved by connecting the function block with the mouse Click and Drag technique. Rather, the loop is created using the Read function block.

As an example, a series of S/R Flip-Flops are used to step through a sequence. At the completion of the sequence all of the S/R Flip-Flops are to be reset. Because the output of the last S/R Flip Flop cannot be looped back to reset all of the preceding S/R Flip Flops, the layout shown below cannot be used.



An Illegal configuration with a Loop in the program from 'Flip Flop 3' back to 'Flip Flop 1'

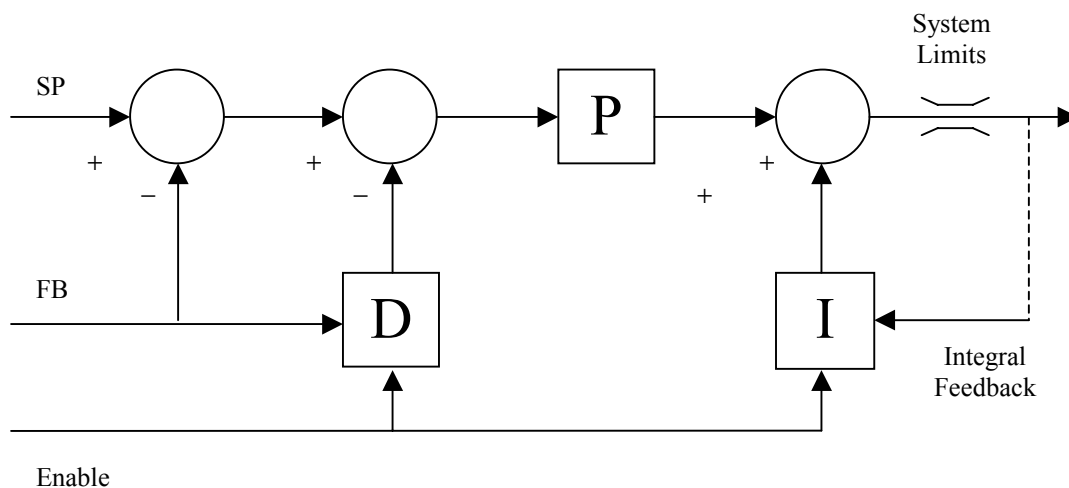
A better configuration is shown below. A Read function block is used to read the output of the last S/R Flip Flop. The output of the Read function block is then used to reset all of the S/R Flip-Flops.



An acceptable configuration with a Loop in the program from 'Flip Flop 3' back to 'Flip Flop 1' using a Read function block

PID Setup

The PID control is implemented as an interacting controller. Added to this is the ability to disable, which effectively removes the derivative and integral components of the controller.



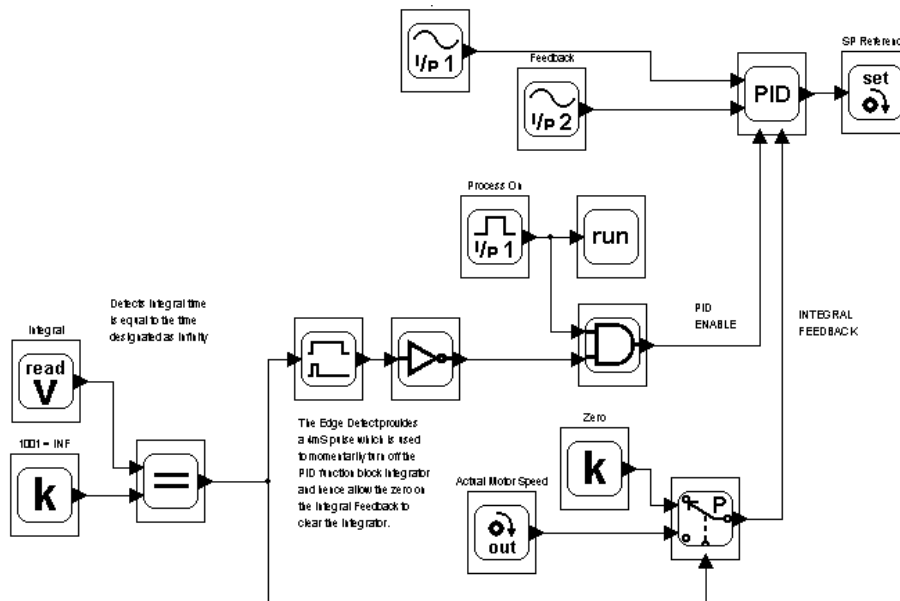
The Derivative action occurs on a change of feedback rather than a change of error. The Integrator requires an Integral Feedback signal that is the result of the PID action after any process limits. This is to avoid Integral Windup.

The Integral and Derivative action are disabled if the Enable is removed. The output will then only be a function of the Proportional action.

The Integrator has no Off state due only to the value of Integral Time selected. If the Integrator needs to be turned Off and cleared when a sufficiently large Integral Time is selected, then this must be done externally to the PID function block.

An example is shown below where the Integral Time may be set at a value of 1 second to 1000 seconds in the relevant screen and the Integrator will function normally if the PID block is enabled. The screen has an enumeration 'INF' associated with the value of 1001seconds, and it is at this value that the Integrator must be turned off and cleared.

The value of Integral Time is read and if equal to 1001, a value of 0 is applied to the Integral Feedback through a Switch. At the same time a short 'Off' pulse is applied to the PID function block that turns off the Integrator and clears its last value with the 0 being applied at the Integral Feedback. The PID function block is then turned back on and the Integrator functions as normal, but now has 0 applied to it. This results in no Integral action.



When providing a setpoint and feedback it must be ensured that the correct values are used to provide the desired output value.

If the Speed Reference function block is used on the output of the PID function block, the value of the PID output needs to be a per unit value (typically 0 to 1). Therefore the setpoint and feedback values and the Gain must be suitably scaled to avoid over ranging the Speed Reference function block.

It is often practical to copy the standard E-Series Process Screens for the Gain, Integral and Derivative adjustments for use with the PID function block. The variable names need to be changed using the Edit Placeholders dialog boxes, from **System.Kc**, **System.Ti** and **System.Td** to **function block name.Kc**, **function block name.Ti** and **function block name.Td** respectively. The Gain screen Scaling Factor should also be changed from 100 to 1 for use with the PID function block.

Standard Program

When disabling various selections within the standard program care must be taken to ensure that conflict does not arise between the standard operating functions and the Vysta program.

1. Main Engine – disabled
System.Ref_Stop_Mode must be included in the program to assign the desired stop mode.
A Reset function must be provided, along with a start and stop.
2. Reference Selection - disabled
Delete the Local Speed/Torque setpoint screens and provide both a speed and a torque reference if required by the program.
If the Reference Selection is enabled, unless the Main Engine is also enabled the Alternate Speed / Torque references cannot be selected and the Motorised Pot and Mrefs will not function.
3. Accel Rate Set - disabled
Provide Accel and Decel rates within the Vysta program. Delete the standard screens.
If the Accel Rate Set is enabled, unless the Main Engine is also enabled the Alternate Accel / Decel rates can only be selected using the break frequency functions.
4. Control Mode Set - disabled
Delete the local control mode Screen and select speed or torque mode within the Vysta program.
5. Fibre optics – disabled
This allows for faster execution of the Vysta program, but none of the drives Fibre functions operate.

Screen List

The Status screen must be included in the Screen List as it contains special information. It is advisable to copy it from the current Screen List. It is good practice to start with the full current Screen List and then prune and graft to suit.

For economy of memory usage, delete the languages that you do not require. This also means you only need to make changes in one of the Screen Lists.

The Vysta Program Screen List should always include a title screen to inform users of the program function that is included in the motor controller.

The Z screens are always useful for fault finding and shouldn't be removed unless redundant. i.e. the fibre Z screens can be removed if the Fibre is disabled. An additional Z screen should be included with the file reference number for program archive purposes.

