

ViewPort

**Local Displays for
LabVIEW applications**

VP420L / VP420V

User Manual Version 1.00



Viewpoint Systems, Inc. does not warrant that the Program will meet Customer's requirements or will operate in the combinations which may be selected by the Customer or that the operation of the Program will be uninterrupted or error free or that all Program defects will be corrected.

VIEWPOINT SYSTEMS, INC. DOES NOT AND CANNOT WARRANT THE PERFORMANCE OR RESULTS THAT MAY BE OBTAINED BY USING THIS SOFTWARE. ACCORDINGLY, THE SOFTWARE AND ITS DOCUMENTATION ARE SOLD "AS IS" WITHOUT WARRANTY AS TO THEIR PERFORMANCE, MERCHANTABILITY, OR FITNESS FOR ANY PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE RESULTS AND PERFORMANCE OF THE PROGRAM IS ASSUMED BY YOU.

NEITHER VIEWPOINT SYSTEMS, INC. NOR ANYONE ELSE WHO HAS BEEN INVOLVED IN THE CREATION, PRODUCTION, OR DELIVERY OF THIS SOFTWARE SHALL BE LIABLE FOR ANY DIRECT, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, SUCH AS, BUT NOT LIMITED TO, LOSS OF ANTICIPATED PROFITS OR BENEFITS, RESULTING FROM THE USE OF THE PROGRAM OR ARISING OUT OF ANY BREACH OF ANY WARRANTY. SOME STATES DO NOT ALLOW THE EXCLUSION OR LIMITATION OF DIRECT INCIDENTAL OR CONSEQUENTIAL DAMAGES, FOR THE ABOVE MAY NOT APPLY TO YOU.

LabVIEW[®], FieldPoint[®], and Compact FieldPoint are registered trademarks of National Instruments Inc.

All other brand and product names are trademarks or registered trademarks of their respective companies.

Copyright © 2003, Viewpoint Systems, Inc.

All Rights Reserved. Reproduction or adaptation of any part of this documentation beyond that permitted by Section 117 of the 1976 United States Copyright Act without permission of the Copyright owner is unlawful.

Viewpoint Systems Inc.
800 West Metro Parkway
Rochester, NY 14623
June 2003
VS-VUPRT-420-0100

Table of Contents

Overview	1
System Requirements.....	1
Getting Started	2
Installing the Software	2
Software Updates	2
Installing the Hardware.....	2
System Setup Verification	2
Removing the software	2
Theory of Operation.....	3
Overview	3
Main Concepts	3
Display Commands:.....	3
Screen:.....	4
Display Manager:.....	4
Display Engine:.....	4
Static Screen:.....	4
Dynamic Screen:	4
Creating Applications	5
Some preliminaries	5
ViewPort “ <i>Hello World!</i> ” Screen	5
Creating a simple application.....	5
Introducing the ViewPort Engine	6
Simplifying with Dynamic Pages	7
Controlling the ViewPort Engine.....	7
Adding a Keypad	8
User I/O Screens	9
Tips on creating a new application	10
VI Reference	11
ViewPort 420 Commands	12
ViewPort 420 Screen Manager	17
ViewPort 420 Screen Engine	18
ViewPort 420 Xlat Keypad.....	19
ViewPort 420 Edit Double	20
ViewPort 420 Edit List	21

Performance Hints.....	22
General Considerations	22
Cable issues.....	22
Additional LabVIEW VIs	23
ViewPort 420 Matrix Effect.....	23
ViewPort 420 Slide Effect	23
Diagnostics.....	25
Specifications	26
Dimensions	26
Power Requirements	27
Environment.....	27
Physical.....	27
Connectors	27
Contacting Us.....	29

Overview

ViewPort displays allow embedded applications to get information to the outside world. The devices are easy to install and even easier to program. The ViewPort LabVIEW library provides all the tools necessary for adding output to your applications.

ViewPort features include:

- **Simple devices** Serial port based displays can be attached to a wide variety of systems. Use ViewPort with FieldPoint, Compact FieldPoint, or PXI based controllers.
- **Code portability** The same ViewPort VIs and examples will run on LabVIEW system with a spare serial port. Code can be developed and tested on a desktop PC and then re-targeted to a FieldPoint controller.
- **Universal Power** ViewPort can work with 7-30 Volt DC sources (same as FieldPoint and Compact FieldPoint)
- **Driver software** drivers and LabVIEW VIs provided
- **Examples** LabVIEW examples show how to make the most of the ViewPort displays

System Requirements

FieldPoint or Compact FieldPoint controller with available serial port

OR

PXI or CompactPCI chassis with available serial port

OR

IBM compatible PC with available serial port

LabVIEW 6i (6.1) or higher

Getting Started

Installing the Software

The ViewPort software is installed through an installation program found on the accompanying CD-ROM. The installation program will run automatically when the CD-ROM is inserted into the CD-ROM drive. If *AutoPlay* has been disabled on your computer, you can manually start the installation program by running the *setup.exe* program found in the CD-ROM's root directory.

Software Updates

Software updates will be made available through the Viewpoint Systems web site. Check the ViewPort web page at <http://www.ViewpointUSA.com/ViewPort> periodically for the latest driver software and new examples.

Installing the Hardware

It is important to follow appropriate electrostatic precautions when handling the ViewPort board. The board is shipped in an electrostatic bag. Be sure that you have removed any electrostatic hazard by attaching a grounding wrist strap or touching the computer chassis before removing the ViewPort board from its bag. Save the bag for future use when the ViewPort board is removed from the system.

ViewPort displays require power from a 7-30 Volt DC power supply. Connect the power supply to the screw terminal power connector supplied. Be sure to follow the indicated polarity (see connectors section – later in this manual).

System Setup Verification

One of the powerful features of the LabVIEW environment is that the same code runs on multiple target devices. The ViewPort examples we provide, and ViewPort VIs for applications you write, will run just as well on your desktop as your LabVIEW RT FieldPoint or PXI controller system.

The easiest way to verify proper operation of your ViewPort display is to simply connect to a spare serial port on your desktop PC.

Removing the software

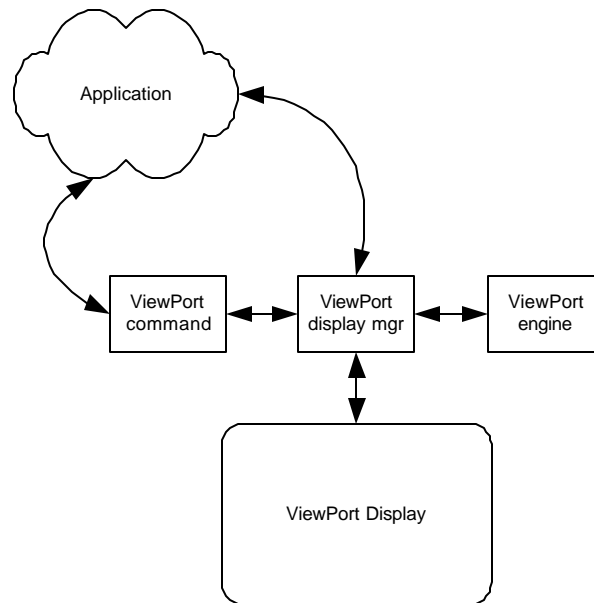
Please use the *Add/remove Software* applet found in the control panel to remove the ViewPort software. This ensures the complete and proper removal of the software from your system.

Theory of Operation

Overview

ViewPort serial displays allow you to add local display to any LabVIEW application with a spare serial port. These displays are especially useful when combined with LabVIEW RT applications deployed on Fieldpoint and Compact FieldPoint systems. The ViewPort software provides the application with everything it needs to create simple, useful, informative displays.

All interaction with the ViewPort display is through the serial port. Command sequences (strings) are built up by the application and transmitted to the display through VIs in the ViewPort library. The ViewPort library consists of low-level and high-level routines that allow you to create ViewPort display screens.



Main Concepts

Display Commands:

Each type of display has a number of commands it recognizes. The ViewPort library contains a VI that allows the application to generate these commands easily. These commands include simple actions like setting brightness/contrast or positioning the cursor to more complicated commands such as drawing a bar graph.

Screen:

A ViewPort *screen* is a predefined page of display information that can be displayed at will. A typical application will consist of a number of screens of information that is displayed sequentially on the ViewPort display.

Display Manager:

The ViewPort *display manager* is the main mechanism for interfacing with the ViewPort displays. The manager has actions that:

- Provides the path for all serial communication with the display/keypad.
- Caches screens for display by the display engine.
- Provides routines to control the actions of the display engine.

Display Engine:

The ViewPort *display engine* is a VI that runs independent of the main application and is responsible for automatically sequencing screens on the ViewPort display. The engine cycles through the screens according to its settings. The application can switch the Display Engine to manual mode and take control over the ViewPort Display at any time. This is especially useful when the system needs to alert an operator of a problem or particular system condition.

Static Screen:

A *static screen* is a page of information that the application creates and doesn't change. A static page is presented to the Display Manager and it will hold a copy of the screen (cache) for replay whenever it is called for. The application can overwrite a *static screen* at any time to freshen its content. This is a *push* type of operation.

Dynamic Screen:

A *dynamic screen* is a page whose content changes frequently. A separate VI is responsible for formatting the information for the display. The application then registers this VI with the *display manager* as the source for the content of a particular screen. The *display engine* will then call the VI anytime it needs to display that particular screen on the ViewPort display. The VI would gather any information needed for that screen from other sources in the application. This is a *pull* type of operation.

Creating Applications

Some preliminaries

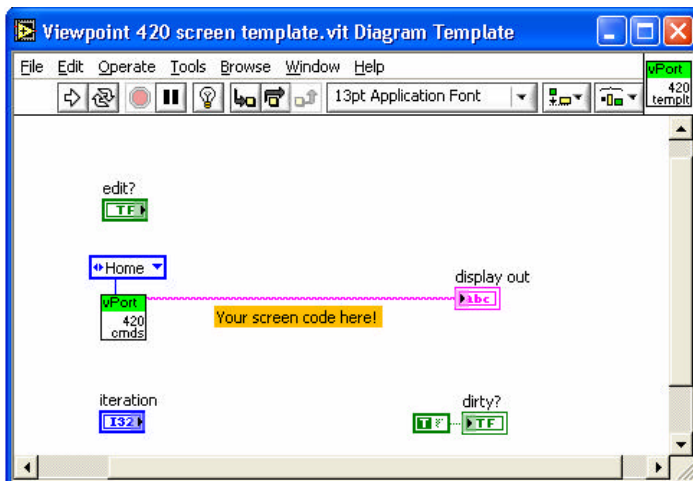
The following LabVIEW examples assume that you have successfully connected and powered your ViewPort display.

The manual assumes that you have a good working of Labview and Labview RT applications.

We recommend developing the framework for the ViewPort application under windows to allow easier access to the software. When the framework is working as expected any application specific interfacing can be done and then downloaded in the RT system for final verification.

ViewPort “Hello World!” Screen

The ViewPort driver works by supplying a screen VI. The screen VI is generated from the screen template shown below.

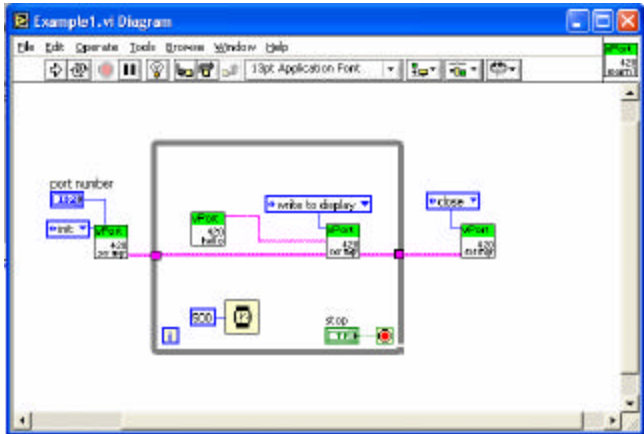


All screens take on the form of this template. The screens themselves are actually very simple; they have an output for the text to display and a control to when to update the screen. Looking at the screen VI template you can see that a call to the command VI with the “Home” action is called to put the display cursor at the home position. Also since the iteration is zero the flag “dirty” is raised to show the Screen manager that the screen has changed.

Creating a simple application

The first example is a very basic look into using a screen template in a very simple application using the screen manager.

The first call to the screen manager is to initialize the manager. All ViewPort Applications must be initialized. This application loops while writing directly to the display.

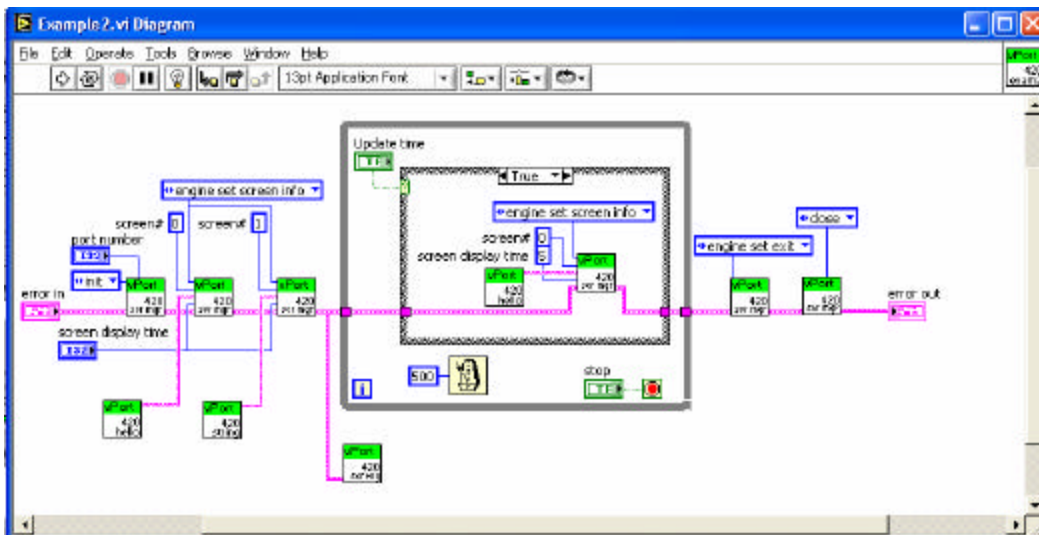


Introducing the ViewPort Engine

In the previous example the code just used one screen and simply just sent the information directly to the screen. But what if you want multiple screens? How is this handled?

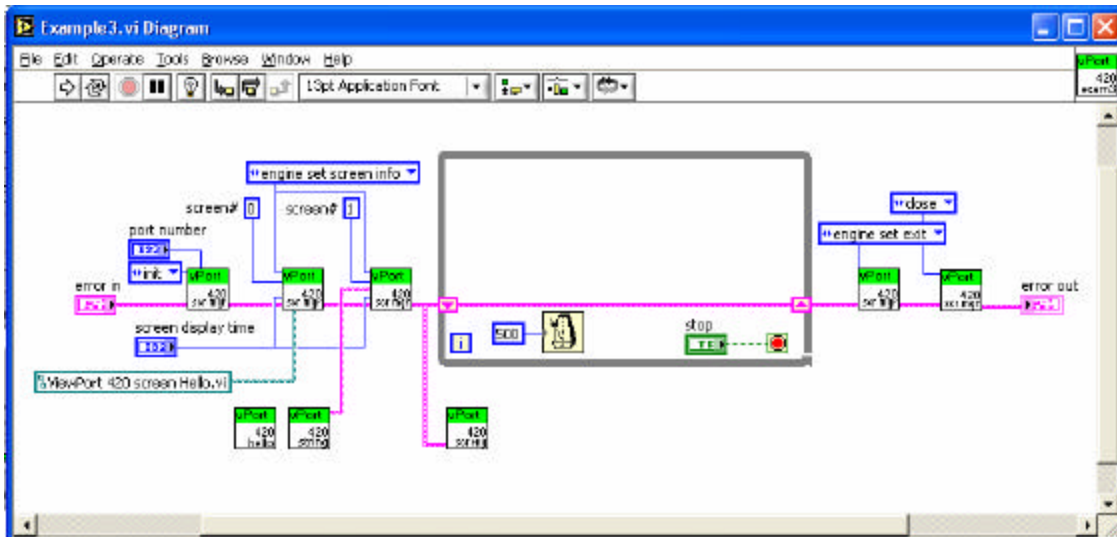
The ViewPort Screen Engine simplifies the use of multiple screens. The engine will automatically update screen information and settings by calling the screen manager. The Screen Engine can be thought of as the behind the scenes application that insures all screens get updated in a regular timeframe.

The application shown in example2.vi uses the screen engine with multiple screens and updates screen zero at an independent rate.



Simplifying with Dynamic Pages

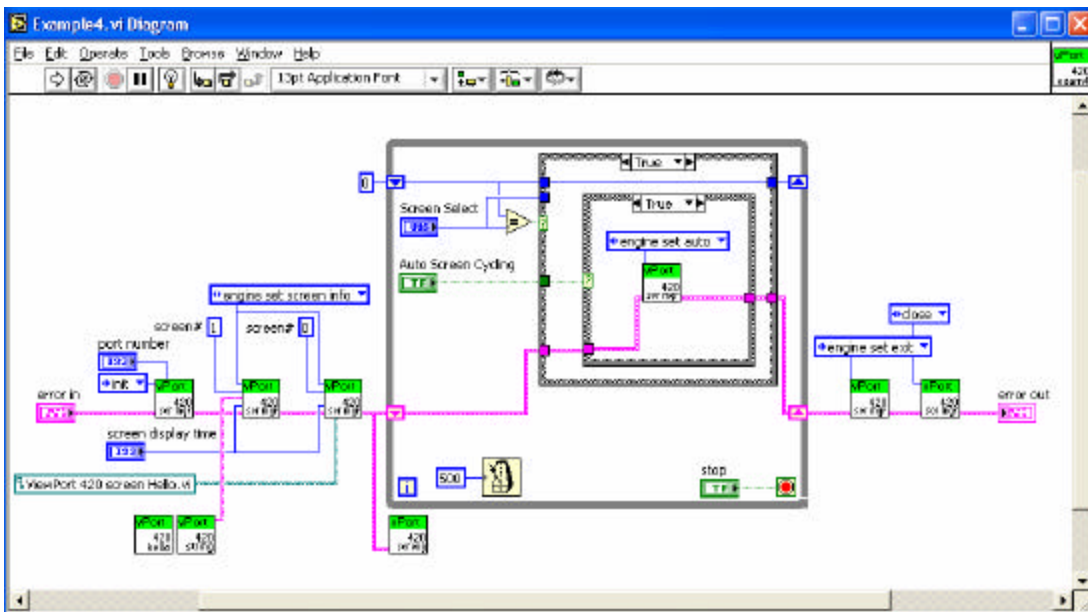
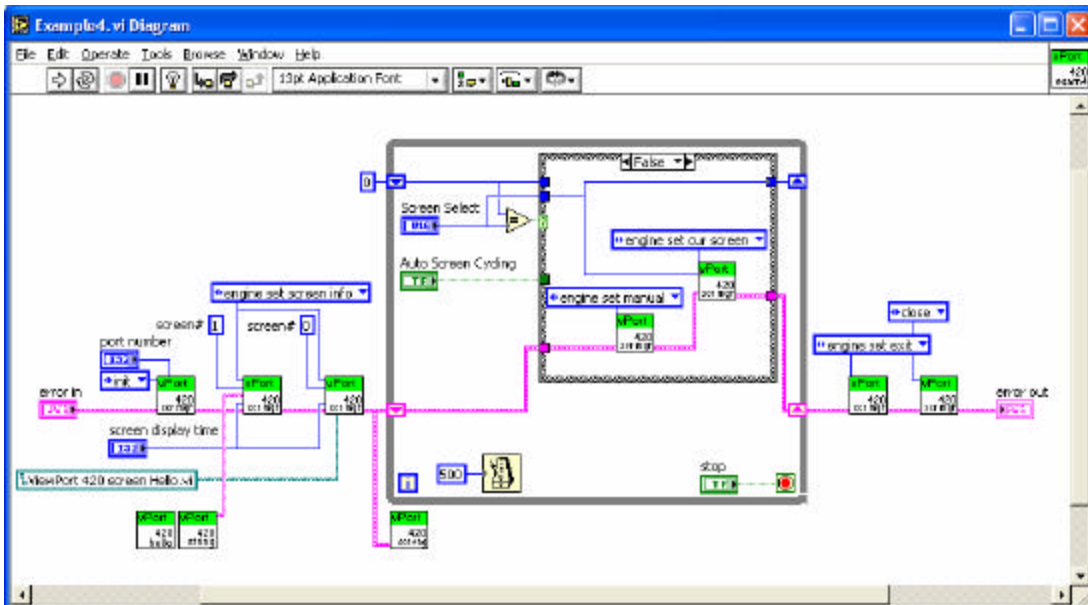
The main loop below was simplified by using dynamic calls to the hello screen VI. The concept of dynamic pages can be confusing but they allow the screen engine to call the VI without programming the application to call the screen VI to update it. The example below, Example3.VI works functionally the same as the code in example2.



Controlling the ViewPort Engine

The ViewPort Engine doesn't always need to automatically update the screens, the engine can be placed in a manual mode allowing the user to force the display order, change the current screen or modify the timing of screens. From manual mode the Application can also enable or disable a screen on the display.

Example4.VI forces the engine in manual mode (when Auto Screen Cycling is false) allowing the application to control which screen is the currently being displayed.



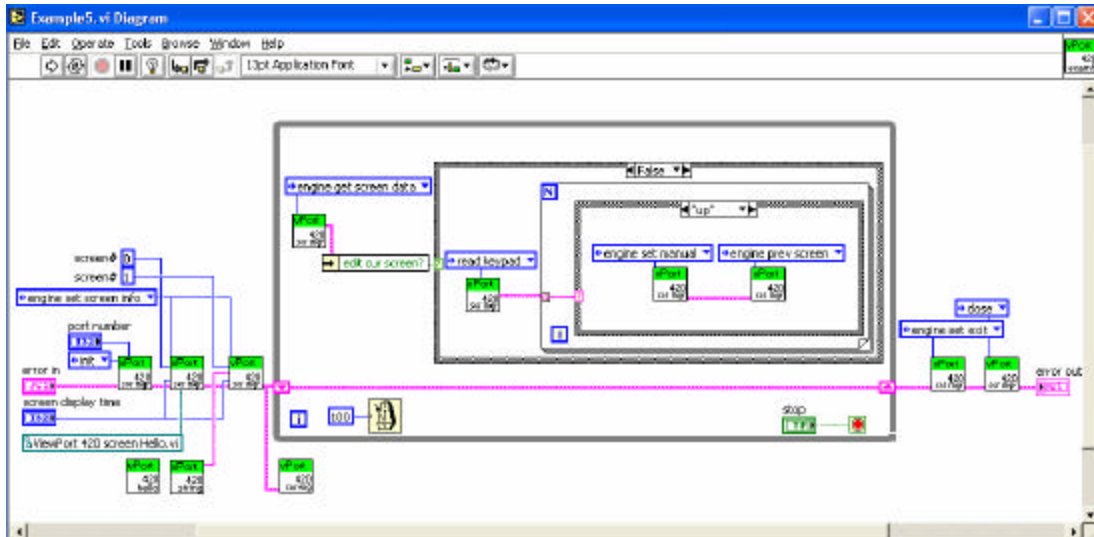
Adding a Keypad

ViewPorts included with keypads may allow for user input. The screen engine looks for keys being pushed and stores them until the application can request the manager for the latest commands available.

The main loop must now look for the keypad sequences and handle these events through an event handler structure as shown below. Any keypad key that is to be handled needs a case in which the required sequence or action occurs based on that particular keypad event.

The example below looks for the up and down arrow keys to manually override the engine's cycling of the screens. The Handler also looks at the 0 and 1 characters to force the display of screens 0 and 1 respectively. Once the engine is in the manual mode the escape key is used to return the engine back to operating in the automatic mode.

This simple keypad handler can be used in a variety of applications further extending the display capabilities of the ViewPort.



User I/O Screens

To further extend the functionality of a ViewPort, the screen VI's have the capability to supply more complex I/O interactions by the use of the Edit flag in the screen VI's. The concept of using the edit control enables anything from selecting which screens to view to allowing input from the keypad to do some physical thing to the system. The idea here is that the event structure shown in example 5 can be further extended per screen. The event handler looks for the enter key to be pressed. When the enter key is pressed and if the current screen allows editing, the main loop doesn't allow the event handler in the application to run, instead the screen event handler (user supplied) overrides this handler and does something different than the default behavior.



The example screen 6.vi installs an edit handler that simply allows the user to move between three different string lines to be output on the screen until enter or cancel are pressed, saving or restoring the string used for that screens display.

Tips on creating a new application

Some of the simplest uses of the ViewPort is for just sending messages out for either debugging or user status. A good percentage of applications will fall into the example 2 and 3 scope. The data screen will be static but the data itself may or may not be updated. The ViewPort Screen template should always be used as a starting point. The screens themselves can be pretty much debugged on a standalone basis without the use of any ViewPort calls. Once the screens operate as expected, the screen can be easily added into the top ViewPort application, keeping in mind whether or not the screen is dynamic (appear to change while being displayed) or static (doesn't change as being displayed) and having the screen manger call the VI dynamically or statically as required.

VI Reference

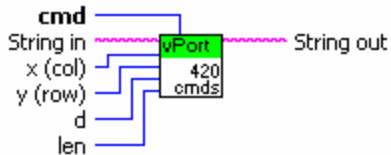
Some of the ViewPort VIs share common parameters. These parameters are defined once below, and are omitted from the individual VI descriptions.

	<p>error in (no error) error in is a cluster that describes the error status before this VI executes. If error in indicates that an error occurred before this VI was called, this VI may choose not to execute its function, but just pass the error through to its error out cluster. If no error has occurred, then this VI executes normally and sets its own error status in error out. Use the error handler VIs to look up the error code and to display the corresponding error message. Using error in and error out clusters is a convenient way to check errors and to specify execution order by wiring the error output from one subVI to the error input of the next.</p>
	<p>error out error out is a cluster that describes the error status after this VI executes. If an error occurred before this VI was called, error out is the same as error in. Otherwise, error out shows the error, if any, that occurred in this VI. Use the error handler VIs to look up the error code and to display the corresponding error message. Using error in and error out clusters is a convenient way to check errors and to specify execution order by wiring the error output from one subVI to the error input of the next.</p>


ViewPort 420 Commands


This VI appends a ViewPort display command to the string being built up. Each type of screen has different commands available to it. Commands control a variety of aspects of the display --- from cursor control to bar graph commands.

Connector Pane



Controls and Indicators


 **String in** The input string. The command will be appended to this string and passed out the string out terminal.


 **cmd** Specifies the command to be appended to the input string. These commands are particular to the type of ViewPort display. See the table below for the particulars.

 **x (col)** Parameters used in certain commands. See the table below for the particulars.

 **y (row)** Parameters used in certain commands. See the table below for the particulars.

 **d** Parameters used in certain commands. See the table below for the particulars.

 **len** Parameters used in certain commands. See the table below for the particulars.

 **String out** The resultant string combining the input string and the command indicated

Command	Parameters	Description
Auto line wrap on		Enables automatic line wrapping. Note that this is not "word wrapping" and wraps may occur in the middle of a word.
Auto line wrap off		Disables automatic line wrapping. Characters beyond the end of a line will be lost.
Auto scroll on		When auto scrolling is on, it causes the VK204-25 to shift the entire display's contents up to make room for a new line of text when the text reaches the scroll position (the bottom right character position).

Command	Parameters	Description
Auto scroll off		When auto scrolling is disabled, text will wrap to the top left corner of the display area. Existing text in the display area is not erased before new text is placed. A series of "spaces" followed by a Cursor Home command may be used to erase the top line of text.
Set cursor position	X (column), Y (row)	This command sets the cursor position (text insertion point) to the [column] and [row] specified. Columns have values from 1 to 20 (0x01 to 0x14) and rows have values of 1 and 2 (0x01 and 0x02).
Home		This command moves the cursor position (text insertion point) to the top left of the display.
Block cursor on		Turns on the blinking block cursor. The cursor shows the current text insertion point. Both blinking and underline cursors may be turned on or off independently. The cursor is off by default
Block cursor off		Turns off the blinking block cursor. Does not affect the underline cursor.
Cursor left		Moves the cursor one position to the left but does not erase any character that may be in that position. Note that this command moves the text insertion point even if the cursor is turned off.
Cursor right		Moves the cursor one position to the right but does not erase any character that may be in that position. Note that this command moves the text insertion point even if the cursor is turned off
Auto Repeat mode on		
Auto repeat mode off		This command turns off auto repeat mode.
Auto transmit keys on		In this mode, all key presses are sent immediately to the host system without the use of the poll keypad command. This is the default mode on power up.
Auto transmit keys off		In this mode, up to 10 key presses are buffered until the unit is polled by the host system via the poll keypad command. Issuing this command places the unit in polled mode.

Command	Parameters	Description
Clear key buffer		This command clears any unread key presses. In a menuing application, if the user presses a key which changes the menu context, any following key presses may be inaccurate and can be cleared out of the buffer between menu changes to prevent jumping around the menu tree. It may also be used to, in effect, reset the keypad in case the host application resets for whatever reason.
Poll Keypad		This command returns any unbuffered key presses via the RS - 232 interface. The host system must be set up to receive the key codes. When the VK204-25 receives this command it will immediately return any unbuffered key presses which may have not been read already. If there is more than one key press buffered, then the high order bit (MSB) of this returned key code will be set (1). If this is the only buffered key press, then the MSB will be reset (0). If there are no buffered key presses, then the returned code will be 0x00. Please note to make use of this command the "Auto Transmit Keypress" mode should be off.
Set debounce time	X (time)	<p>[time] is in increments of 6.554 milliseconds.</p> <p>This command sets the time between key press and key read. All key types with the exception of latched piezo switches will "bounce" for a varying time, depending on their physical characteristics. The default debounce time for the module is about 52 mS, which is adequate for most membrane keypads. This time equates to a setting of 8 using this command as there is a debounce time resolution of 6.554 milliseconds.</p>
Initialize thick vert bar graph		This command defines the 8 special/user characters to be blocks suitable for use in drawing wide (5 pixel) vertical bar graphs. Any previously existing definitions will be lost. Once this command has been issued, any number of vertical bar graphs may be drawn unless the characters are redefined by another command.

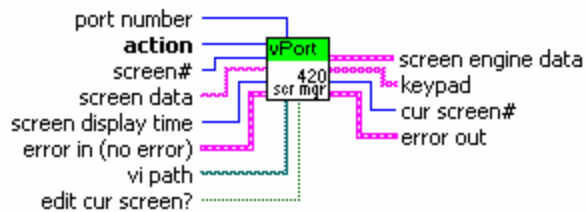
Command	Parameters	Description
Initialize thin vert bar graph		This command defines the 8 special/user characters to be blocks suitable for use in drawing narrow (2 pixel) vertical bar graphs. Any previously existing definitions will be lost. Once this command has been issued, any number of vertical bar graphs may be drawn unless the characters are redefined by another command
Initialize horiz bar graph		This command defines the 8 special/user characters to be blocks suitable for use in drawing horizontal bar graphs. Any previously existing definitions will be lost. Once this command has been issued, any number of horizontal bar graphs may be drawn unless the characters are redefined by another command.
Define custom character		
Draw vertical bar graph	X (column), Y (height)	Draws a vertical bar graph in [column] having a height of [height] pixels. The height may range from 0 to 20 (0x00 to 0x14) pixels. The necessary characters must first be initialized by either of the commands shown in section 5.1.1 or 5.1.2, which will determine the width of the graph drawn. Graph may be erased by drawing a bar graph of height = 0 in the same column.
Draw horizontal bar graph	X(column), Y(row), d, len	Draws a horizontal bar graph in [row] starting at [column] with a length of [length] pixels. [row] may have a value of 0x01 or 0x02, column may range from 0x01 to 0x14 and length may be from 0x00 to 0x64 (0 to 100) if the graph can extend the full width of the screen. Each column is 5 pixels wide (spaces between the columns don't count). [dir] specifies the direction: 0x00 goes from left to right, 0x01 goes from right to left.
Initialize large digits		This command defines the 8 special/user characters to be blocks suitable for use in drawing large digits. Any previously existing definitions will be lost. Once this command has been issued, any number of large characters may be placed until the characters are redefined by another command.
Place large digits	X(column), Y(row)	

Command	Parameters	Description
Clear Display		This command clears the display and resets the text insertion point to the top left of the screen.
Set brightness	X	This command sets the display's brightness to [brightness], where [brightness] is a value between 0x00 and 0x03 (between 0 and 3) according to the table below. Hex Values Brightness 0x00 100% 0x01 75% 0x02 50% 0x03 25%
Display on	X(time)	This command turns on the display for a time of [minutes] minutes. If [minutes] is zero (0), the display will remain on indefinitely. Note: the factory default for display is on.
Display off		This command turns the display of the VK204-25 off.
General purpose output off	X	This command turns OFF any of the General Purpose Outputs (see section 2.2 for a description of the GPOs). [gpo #] is 1 to 6. Note that OFF means that the output floats.
General purpose output on	X	This command turns ON any of the General Purpose Outputs. [gpo #] is 1 to 6. ON means that the output is pulled low (ground via 240 ohms).
Read module type		This command will return, over the RS-232 interface, the 1-byte model type value of the module. This command will return a 1-byte hex value. Values for various modules at the time of this publication are as follows:
Read version number		This command will return the firmware version number of the VK204-25. This command will return a 1-byte hex value.
Set Contrast	X	











ViewPort 420 Screen Manager

The ViewPort Screen Manager is the main interface to the ViewPort Display hardware. The manager provides direct access to the serial communication between the application and the display. The manager is the primary interface to the Screen Engine, providing control over the engine's mode of operation. The manager also maintains the parameters for the Screen Engine, including information for each screen registered being displayed.

Connector Pane



Controls and Indicators

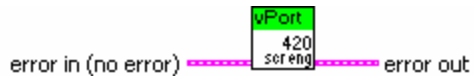
-  **action** The action the manager is to perform.
-  **screen data** A string representing the content for a ViewPort page.
-  **screen#** Used with certain actions to apply action to a particular screen.
-  **screen display time** An integer representing the number of seconds that this page should be displayed.
-  **vi path** If this page is to be a dynamic page, this path specifies the path to a VI that will provide the page content to the ViewPort Engine.
-  **edit cur screen?** An input that is used by the "xxxx" action, specifies whether or not this page should be marked as being edited.
-  **port number** port number varies by platform. The parameters for serial port numbers depend on the whether you use Macintosh, Windows, or Unix.
-  **screen engine data** This cluster contains the internal state of the ViewPort Engine.
-  **keypad** This is an array of strings that represent the keypad information returned by the ViewPort keypad interface
-  **cur screen#** This indicates the current screen being manipulated by the ViewPort Manager.

ViewPort 420 Screen Engine

The ViewPort Screen Engine handles the automatic display of screens for the application. The engine will cycle through enabled screens based on the timing information the application provides.

The engine is implemented as an independent loop, and will only exit when the application set the engine's mode to exit via the ViewPort Screen Manager. The application can take manual control of the engine, and either force a particular screen to be displayed OR directly write to the ViewPort display.

Connector Pane



ViewPort 420 Xlat Keypad

The Xlat Keypad VI converts the raw keyboard codes to a common set of entries.

Connector Pane



Controls and Indicators



raw data Raw keypad data received from the hardware.



keypad



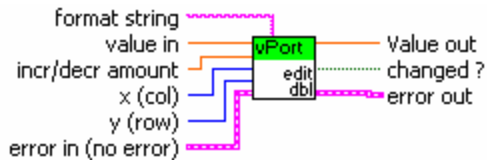
raw data out Remaining Raw Keypad data after A Xlat has been run. This data is still valid and contains any command after the current translation.

Keyboard entry	
Up	0
Down	1
Right	2
Left	3
Enter	4
Clear	5
Help	6
2 nd	7
<unknown>	8
	9


ViewPort 420 Edit Double


The Edit Double functions as a keypad handler that allows the user to increment or decrement a double value on the display. The calling routine supplies the increment/decrement value to apply when a user presses the Up/Down keys. The initial value of the display is based on the Format string and the double initially supplied.


Connector Pane





Controls and Indicators


-  **incr/decr amount** The amount that the initial value is incremented or decremented for each corresponding up or down key pressed.


-  **value in** The Double value to be incremented or decremented.

-  **format string** The Labview format string for the 2nd line of the labview double.

-  **x (col)** X position to place the block cursor.

-  **y (row)** Y position to place the block cursor.

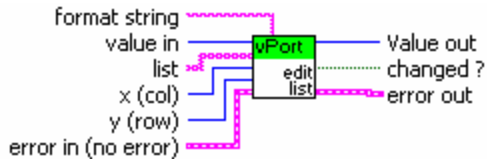
-  **changed ?** Boolean to test to see if value changed.

-  **Value out** The resultant value of the user interaction with the keypad and display. If the user hit the Clear key the value will revert back to the input value, otherwise it will be the updated double value.

ViewPort 420 Edit List

The Edit List VI allows the Display of a string list. It manages user keypad entry to scroll through the list and allows the user to either Enter (accept changes) or cancel (abort changes) of the string list selection. The Up and down Arrow keys are used for scrolling through the list.

Connector Pane




Controls and Indicators


 **value in** Startup index into string list


 **format string** string Labview Format String to Format string list

 **x (col)** X Location for block cursor.

 **y (row)** Y location for block cursor.

 **list** list List of strings to display (up and down arrow move through the string list)

 **changed ?** Will be true if user accept the changes by pressing the accept key or false if the user presses the cancel key.

 **Value out** Sting List index after Edit list runs. The value will be the same as input if cancel entered. If enter is pressed the value will be the index of the selected string.

Performance Hints

General Considerations

In RT applications always make sure to keep the ViewPort code out of the Time Critical loop. Avoid using dynamic variables such as arrays that require relatively slow memory manager support and are sometimes limited to the memory available on board.

Any dynamically called VI's need be either downloaded to the target with the VI path on the target specified in the application or include a case statement that never runs that calls all the Dynamic Screen VI's (and their custom subVIs) so that the VIs are included automatically in the download.

Cable issues

Serial Cable shall be a Male to Female type, 1 to 1

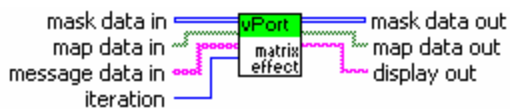
Additional LabVIEW VIs

Included on the installation disk are additional VI's that exercise many of the ViewPort Screen manager options. The "Test ViewPort 420.VI" includes screen calls that utilize all the functionality mentioned in this manual plus it adds some additional VI's just for reference and to show other capabilities of the ViewPort. Listed below are just some of the additional ways the ViewPort 420 can be utilized.

ViewPort 420 Matrix Effect

This VI is used in a screen VI that is supposed to mimic the opening to the movie The Matrix. It was created just for fun, but can be used as an example for other types of screens that can be accomplished using the ViewPort.

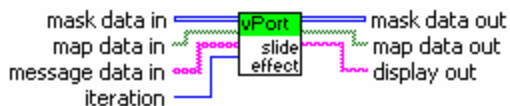
Connector Pane



ViewPort 420 Slide Effect

The VI is used in a screen VI that demonstrates how to generate a moving slide bar on the ViewPort.

Connector Pane



ViewPort Error Codes

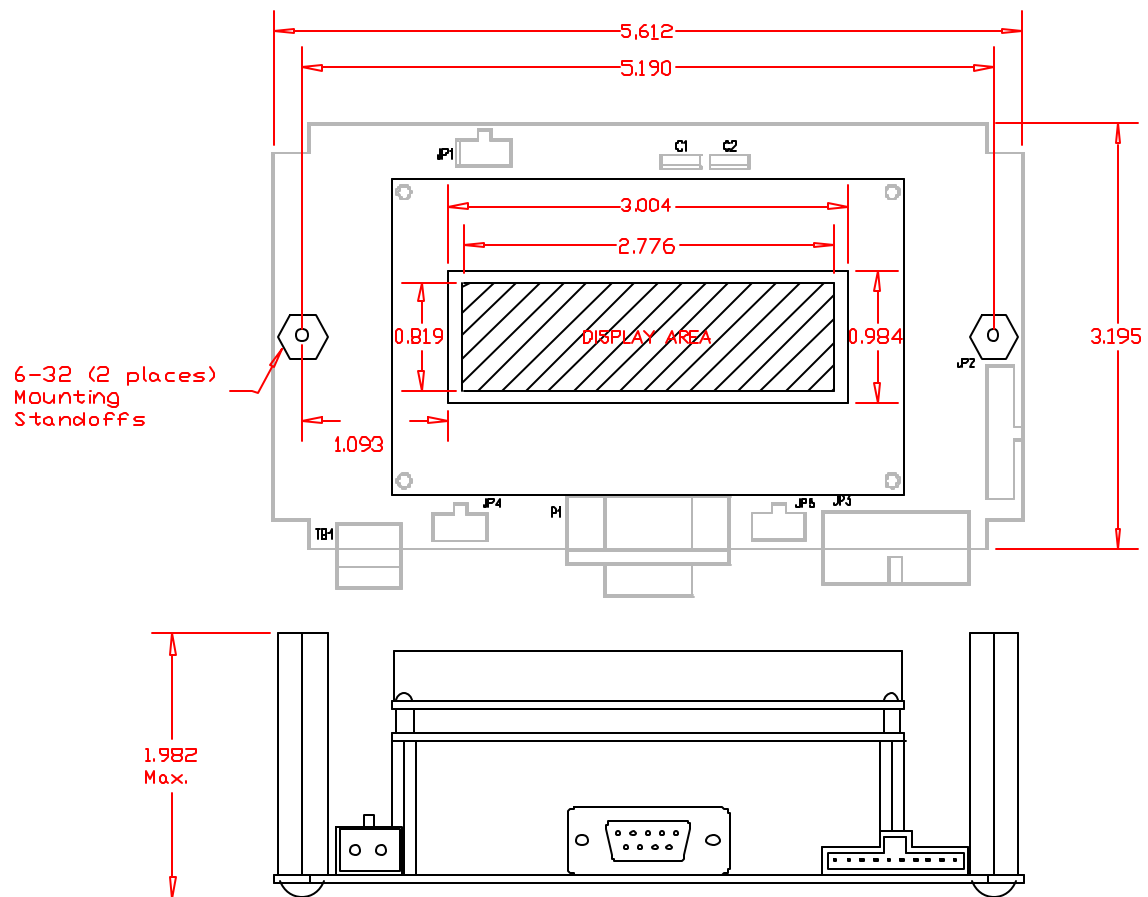
Error Code	Error String	Error Description
Set by Labview Serial init	ViewPort 420 Screen Manager – serial init failed	There is a problem with the Serial port initialization. Either the port does not exist or is in use already.
Set by Labview Open Reference	ViewPort 420 Screen Manager – cannot locate screen callback: <screen vi>	The Vi reference supplied to the screen manager could not be found or does not exist.
100	ViewPort 420 Screen Manager – invalid screen #	A call was made with an out of range screen number.

Diagnostics

Problem	Help
No Display	Check power connections' making sure power is indeed at terminal block and polarity is correct.
No communication	Be sure that a 1 to 1 cable is being used and that the right com port number is specified.
Screen doesn't dynamically update	Be sure that the Screen VI is either being called inside the application while loop or that the application is setup to call the screen VI by reference. If the VI is called by reference, be sure that the VI is downloaded with the correct path information to the target.
Dynamic Screen doesn't show	Be sure the VI path is correct in the application. Also be sure VI is downloaded to target as required.
Parts of the display show <i>blocks</i> instead of the expected text.	Be sure the strings that are sent to the display are defined as 20 characters long per line.

Specifications

Dimensions



Power Requirements

Input Power.....7-30V DC; 1.0 watts

Environment

Operating Temperature (VP420L)0-50 °C (extended ranges available consult factory)

Operating Temperature (VP420V)-20-70 °C (extended ranges available consult factory)

Storage Temperature (VP420L)-20-70 °C

Storage Temperature (VP420V).....-40-80 °C

Operating Relative Humidity 90% max non-condensing

Physical

Characters80 (4 lines by 20 characters)

Display Area3.00 x 1.00 inches [76.3 x 25.20 mm] (XxY)

Matrix format 5 x 7 with underline

Dot Pitch 0.60 x 0.60 mm (X x Y)

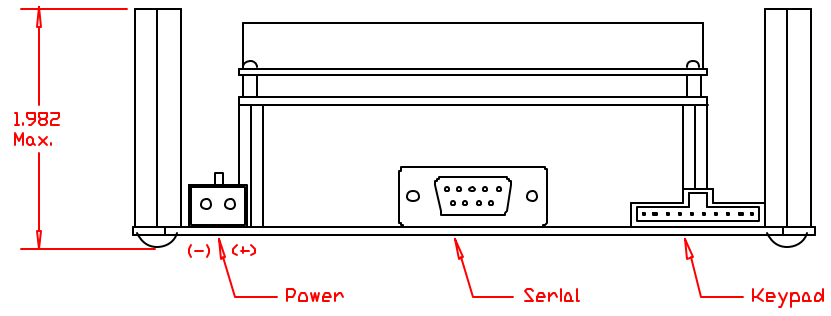
Dot Size 0.55 x 0.55 mm (X x Y)

Connectors

SerialDB9 – Female

PowerTerminal Block, 2 position Compression Screw, 16 AWG max

Keypad.....10 Position Ribbon



Contacting Us

Viewpoint Systems, Inc.
800 West Metro Parkway
Rochester, NY 14623

voice: 585-475-9555

fax: 585-475-9645

e-mail: support@ViewpointUSA.com

Technical support is available any business day from 9:00 AM to 5:00 PM Eastern time. Of course, you may fax or e-mail questions at any time.

