RASPBERRY PI COMPATIBLE COMPILER FOR LABVIEW

GUI Operating Manual

Abstract

This document includes important information on how to operate the Raspberry Pi Compatible Compiler for LabVIEW Main GUI. It allows the developer to take the most out of the compiler.

Filipe Altoe
lvforpisupport@tsxperts.com

Contents

Raspberry Pi Compatible Compiler for LabVIEW GUI

Introduction

The Raspberry Pi Compatible Compiler for LabVIEW is a product based on LabVIEW (Laboratory Virtual Instrument Engineering Workbench) by National Instruments. LabVIEW is a graphical programming language that uses icons instead of lines of text to create applications. In contrast to text-based programming languages that use instructions to determine the order of program execution, LabVIEW uses dataflow programming. In data flow programming, the flow of data through the nodes on the block diagram determines the execution order of the VIs and functions. VIs, or virtual instruments, are LabVIEW programs that imitate physical instruments. This document assumes basic familiarity with LabVIEW. For more information about LabVIEW, visit www.ni.com/labVIEW.

The Raspberry Pi Compatible Compiler for LabVIEW is a true compiler product that allows one to compile, deploy and run stand alone, full fledge LabVIEW applications embedded in Raspberry Pi targets. All flavors of official Raspberry Pi (or RasPi for short) single board computer boards released by the Raspberry Pi organization are supported. However, performance differences may be noticed based on which version of Pi is selected and the complexity of the LabVIEW VI downloaded. The Raspberry Pi is a single board computer; therefore, the same performance considerations are valid in the selection of regular computers to run LabVIEW applications. The Raspberry Pi boards that have higher horse power CPUs and more RAM memory will outperform their counterparts with lower end CPUs and less RAM memory. As an example, the Raspberry Pi 2 Model B will outperform its older brother, the Raspberry Pi 1 Model B+. The former includes a 900MHz quad-core Arm Cortex-A7 CPU and 1GB of RAM memory while the latter is powered by a 700MHz single-core ARM11 CPU and 512M of RAM memory.

Another point that is fundamental to highlight is that the Raspberry Pi Compatible Compiler for LabVIEW requires the RasPi target to have an Ethernet/Wifi port and an IP address for communication with the host development computer for initial download of the compiled VI. Once the download is complete, the RasPi can operate standalone and disconnected from the network. Furthermore, the setup of the RasPi, as it will be seen in a future section, requires the RasPi to have Internet access, so all the correct packages can be installed upon execution of the setup script.

This requirement doesn't constraint the use of the compiler with RasPi boards that include an Ethernet port as part of the board. The user is free to elect using any Raspberry Pi, as long as it has Ethernet connectivity. As an example, the user can setup a Raspberry Pi Zero with an Ethernet to USB dongle.

The Raspberry Pi Compatible Compiler for LabVIEW product works in combination with any of the available editions of LabVIEW for Windows; Base, Full, Professional and Home. The oldest version of

LabVIEW supported will be LabVIEW 2014. One doesn't need to have the LabVIEW Application Builder installed on the machine, which allows the use of both the LabVIEW Base and Home editions. The compiler also allows the compilation of full GUIs or Console Application. Refer to the section named Supported Compilation Types for more information on the differences between the two compilation types.

Installing the Raspberry Pi Compatible Compiler for LabVIEW

Now that your RasPi target is ready; if you haven't already done so, the next step is to install the Raspberry Pi Compatible Compiler for LabVIEW on your development PC running LabVIEW for Windows. The compiler product has been packaged as a VI Package Manager vip file. Download the VIP file from this link.

One important detail about downloading the file that one needs to pay attention to. It is important to click of the Download ZIP button on Github, as shown by figure below. This will download all VIP package files for the multiple versions of LabVIEW supported. Make sure to select one appropriate to the version of LabVIEW you are using.
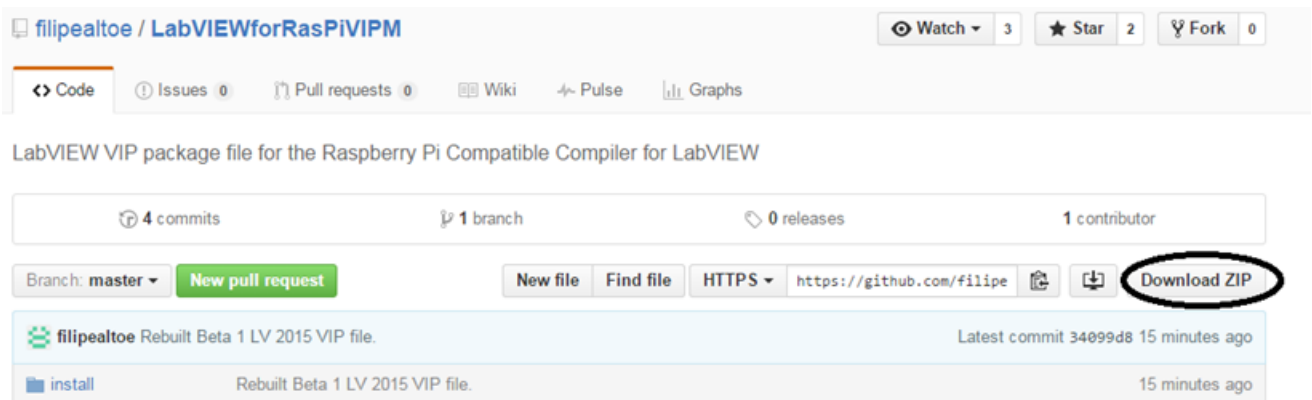

Before you proceed with the installation process, make sure your user account has administrator privileges. Installing the VIP package file in LabVIEW from an account without administrator privileges will prevent the compiler VIs to run. Once the file has been downloaded to your development computer, open VI Package Manager from LabVIEW->Tools Menu, as shown by figure below.
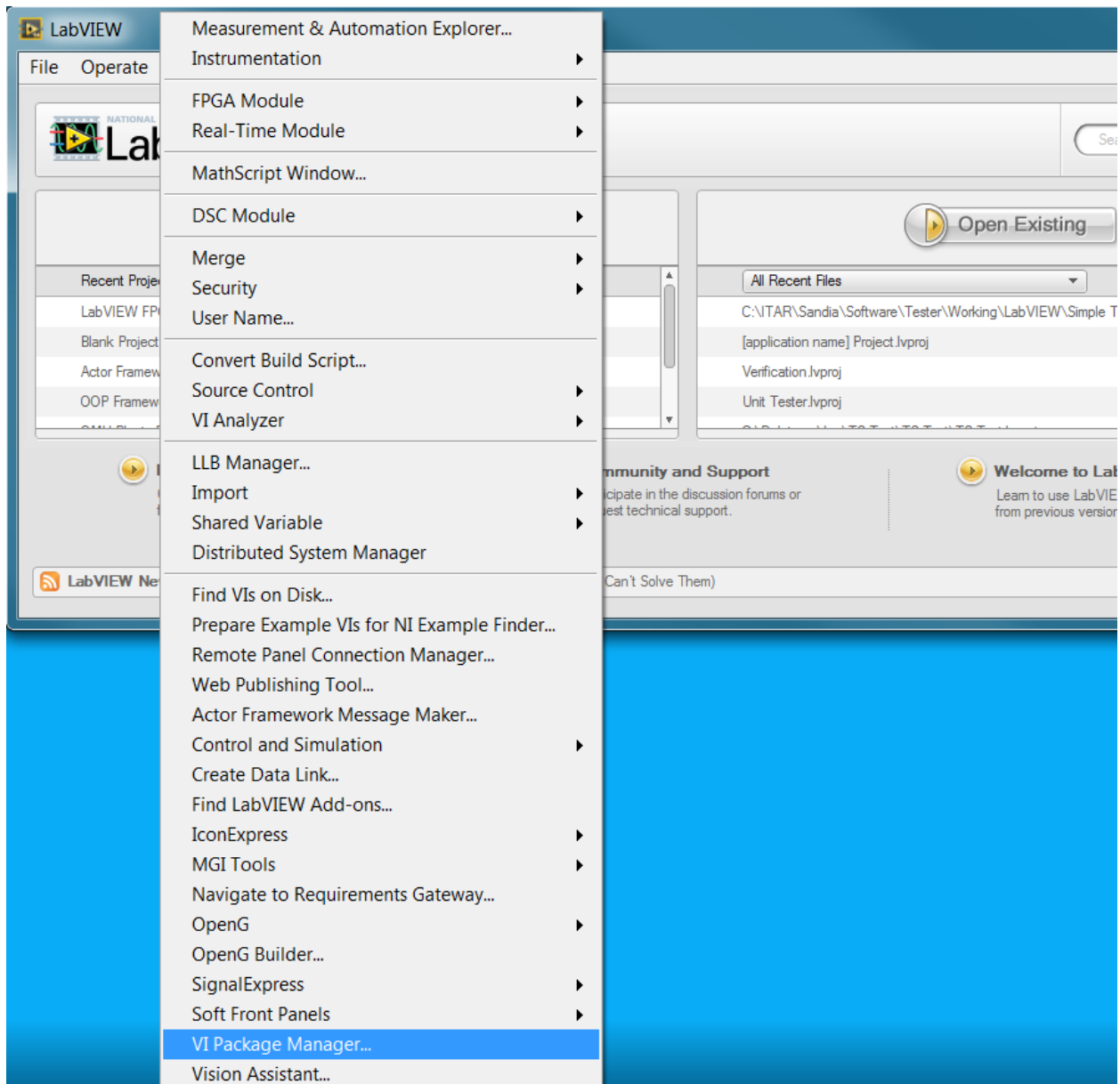
From within VIPM, open the corresponding RasPi Compiler for LabVIEW.vip file; as shown by the following figure.
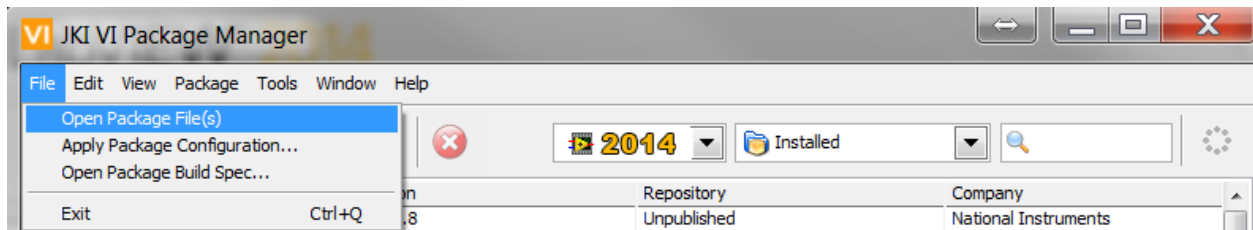
Follow the instructions on VIPM throughout the installation. Once that is complete; the VI Package Manager will show an item named Raspberry Pi Compatible Compiler for LabVIEW on the list of Installed packages, as shown below.
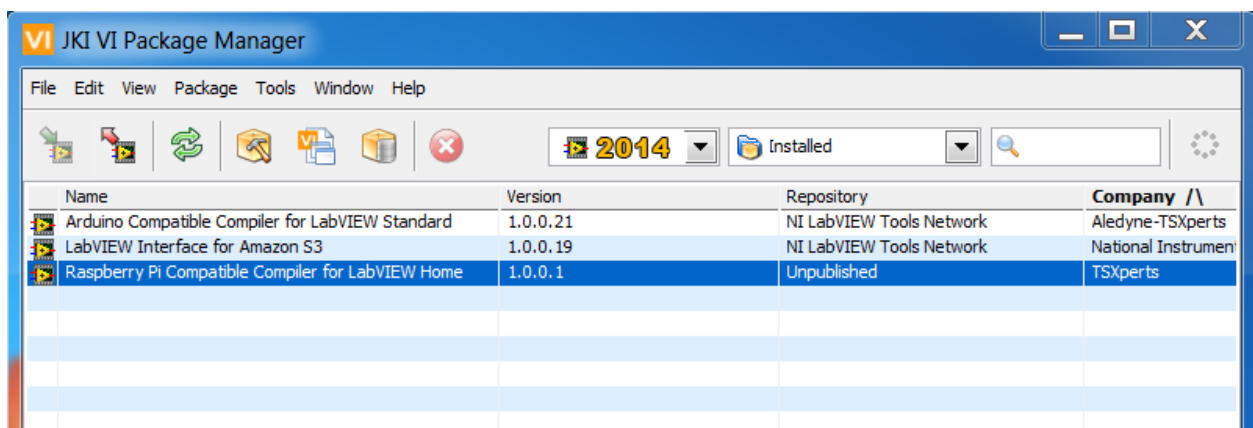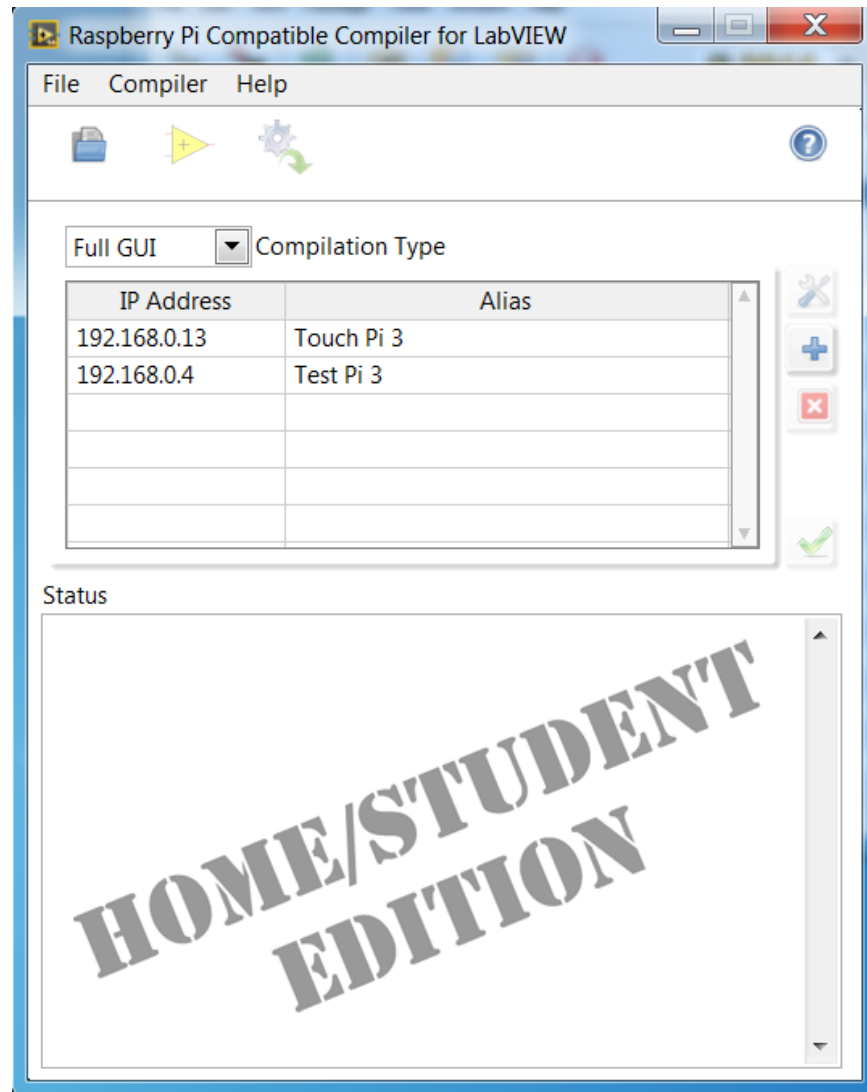

Moreover, a menu items named Raspberry Pi Compatible Compiler for LabVIEW will be added to your LabVIEW Tools menu, as shown by figure below.
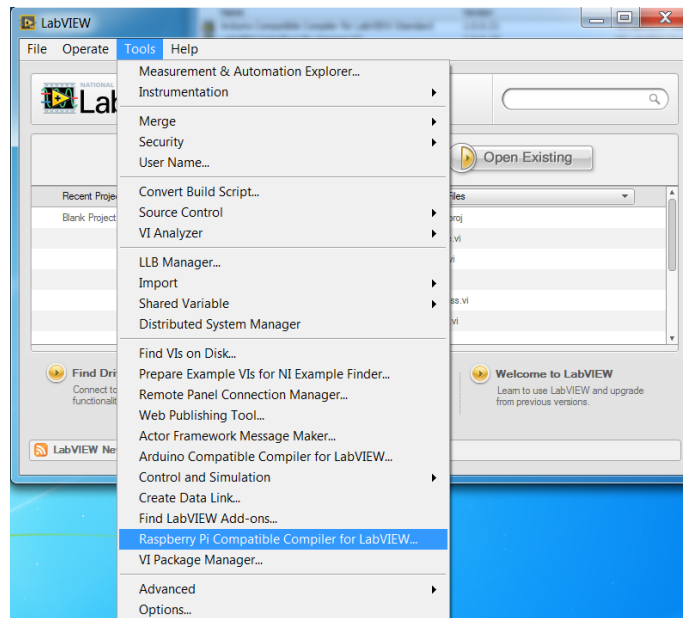
Introduction

Once you click on the item named Raspberry Pi Compatible Compiler for LabVIEW on the LabVIEW Tools menu, the following screen will be launched.

The snapshot above shows the Home edition of the Compiler. The Home edition displays a corresponding watermark as part of the compiler main window. The Standard edition doesn't include this watermark. The compiler screen has three main areas that are worth detailing; the RasPi connectivity area, the RasPi Compilation Tools area and the RasPi Compilation Status area. The first one we will focus our attention to is the area in the middle, which we will call the RasPi connectivity area. This area is highlighted in the following figure.
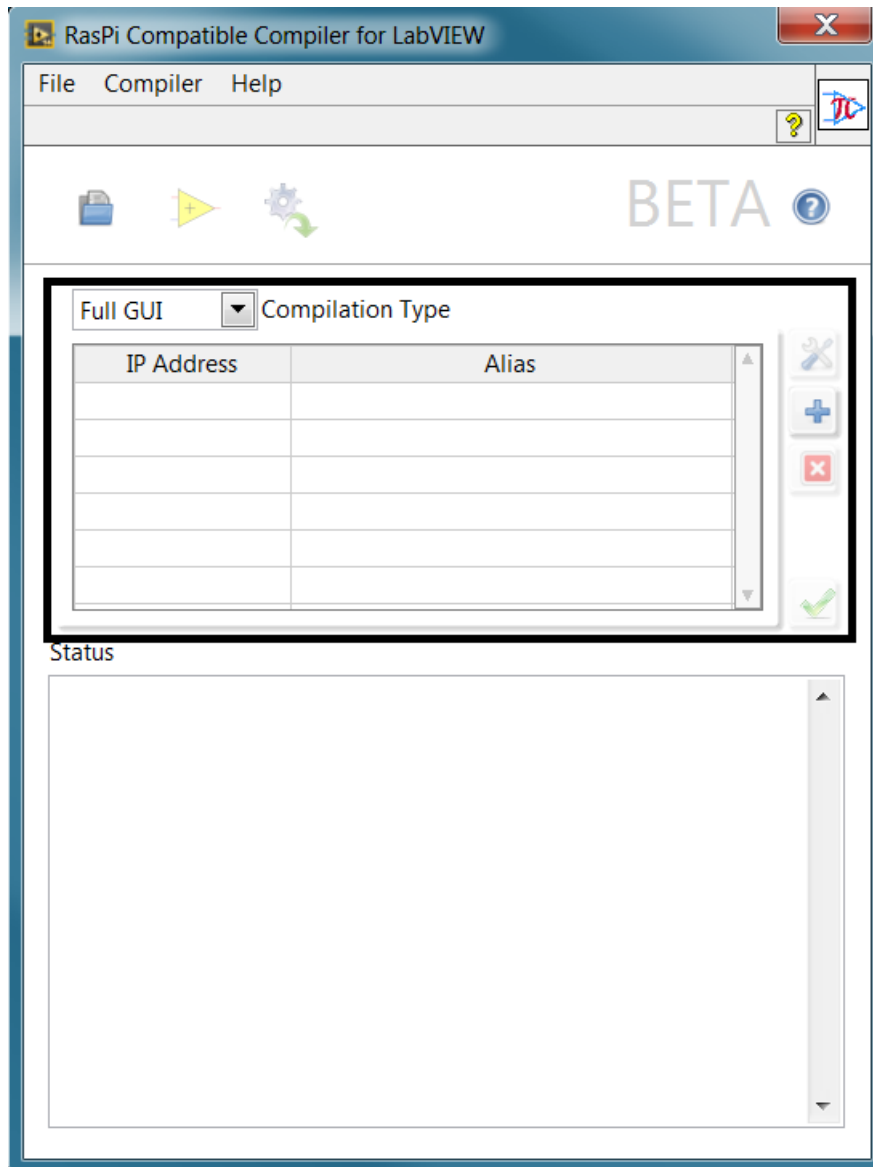
This is the area the user should start with when running the compiler for the first time. This section of the screen is where the user will configure one or multiple RasPi targets to receive compiled LabVIEW code from the Compiler product. The first step is for the user to enter a valid IP address for a RasPi target that has been setup to work with the compiler. In order to do that, the user should click on the "+" button of this area. This will launch the following screen.
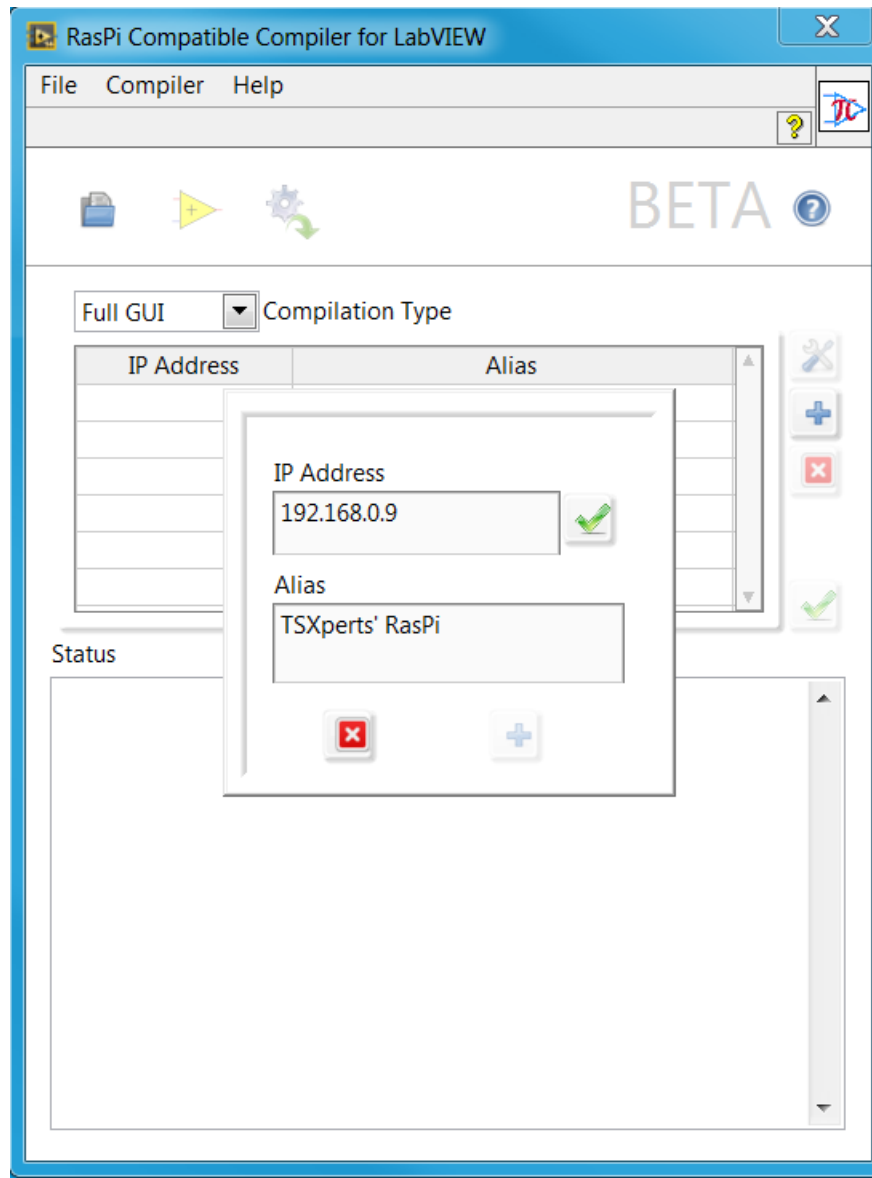
As one can see, this window allows the user to type a valid IP address for a previously configured RasPi target, as well as an Alias that will help the user in identifying which RasPi target the corresponding IP address relates to. The Alias is an optional field and can be left blank. It is important to notice that the "+" button will remain disabled until the user has clicked on the verification button; the green checkmark on the right hand side of the IP address field.

The verification button will trigger the compiler to attempt a connection with the RasPi IP Address, as well as verify if the corresponding RasPi target has been properly setup to receive compiled LabVIEW VIs

from the compiler. Once this activity is complete, the user can add the target to the list by simply clicking the "+" button.

If you get an error at this step; either the IP address is not valid, you have a network issue that is preventing the connection between your development machine and the Pi or the Raspberry Pi you have specified hasn't been properly setup to work with the compiler.
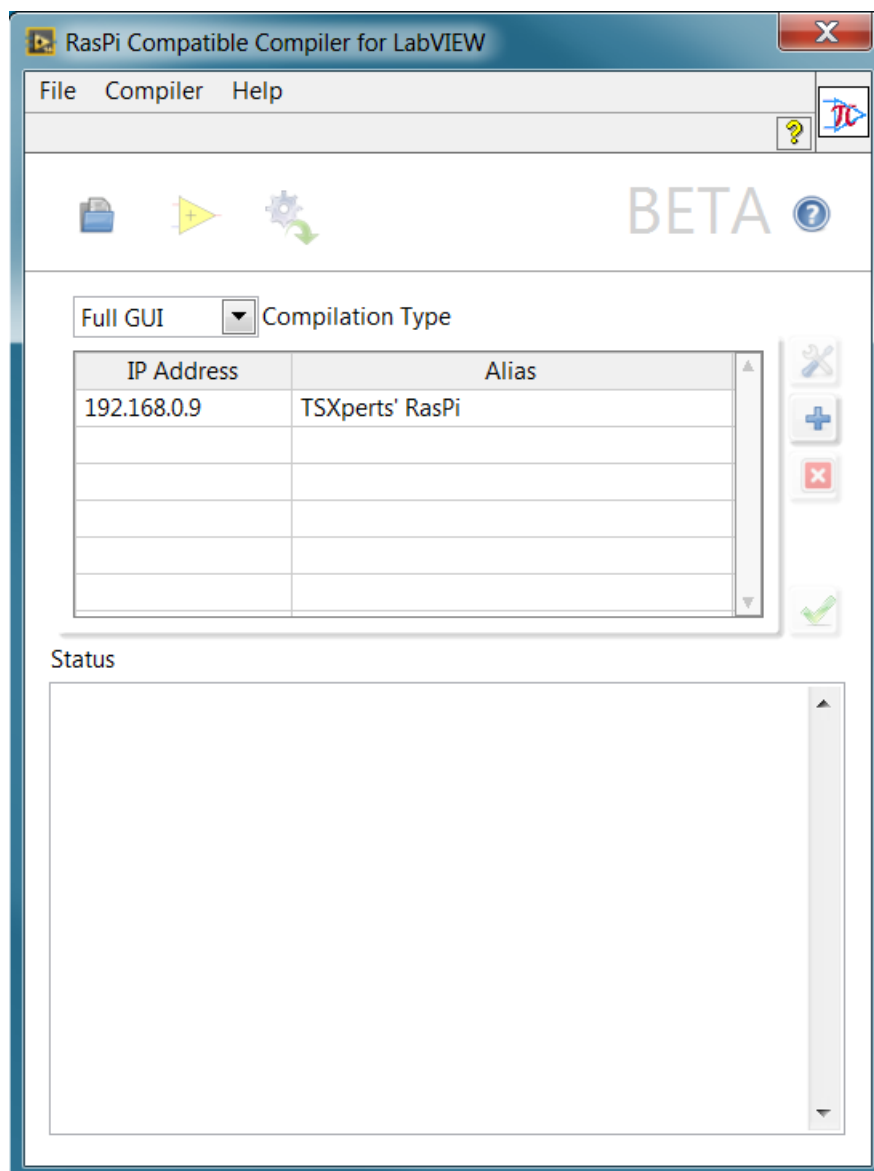
Once that is complete, the configured RasPi will show up on the list of configured IP addresses, as illustrated above. This activity only needs to be performed once per each new target. The information will be remembered by the compiler application even after it has been restarted.

This area of the compiler also offers the option for the user to edit an existing configured RasPi target's IP address and/or Alias; to remove an existing target and to run the connectivity verification on a previously configured RasPi target. Once a target is selected in the list, as shown by the following figure, the corresponding buttons for Editing, Deleting and Verifying Connectivity are enabled.
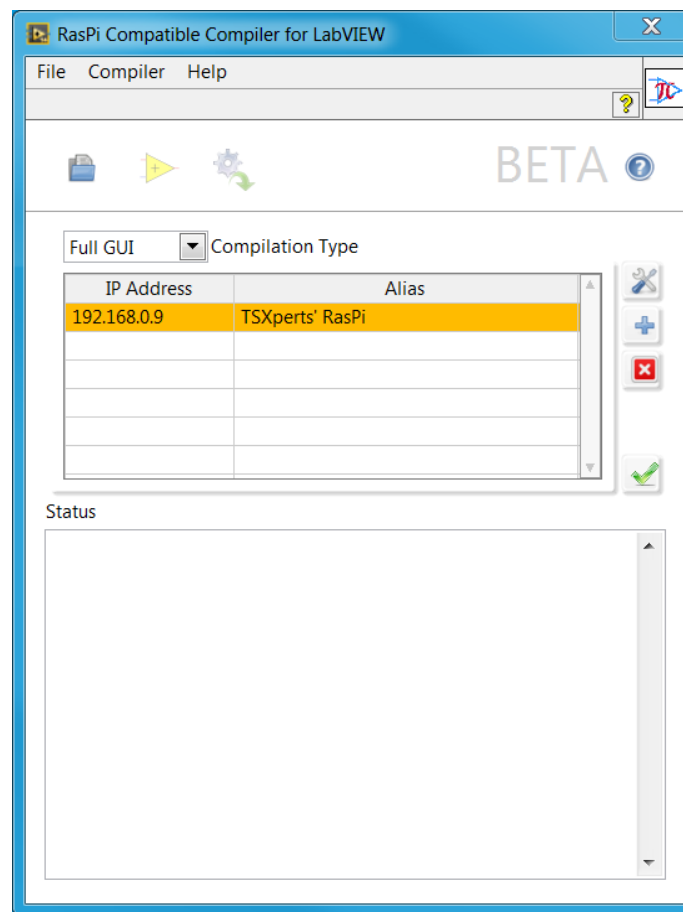
Clicking the editing button will open a separate screen the user can change either the target IP address, alias or both.
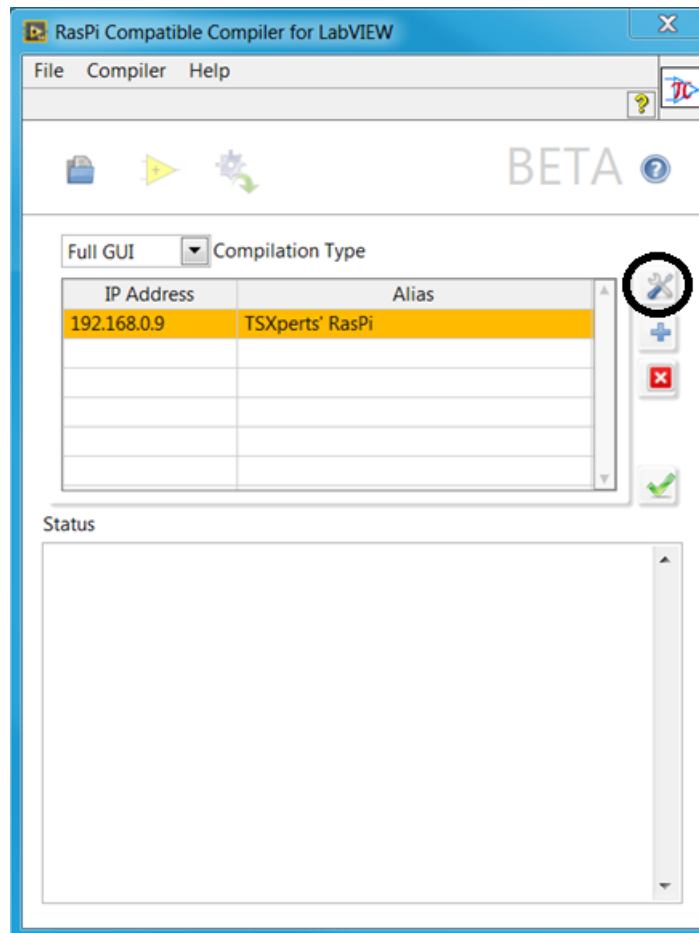
The user can have multiple Raspberry Pi targets setup to work with the compiler. To do that, just click the "+" button as illustrated below and configure the next Raspberry Pi target by following the steps you have done in setting up the first target.
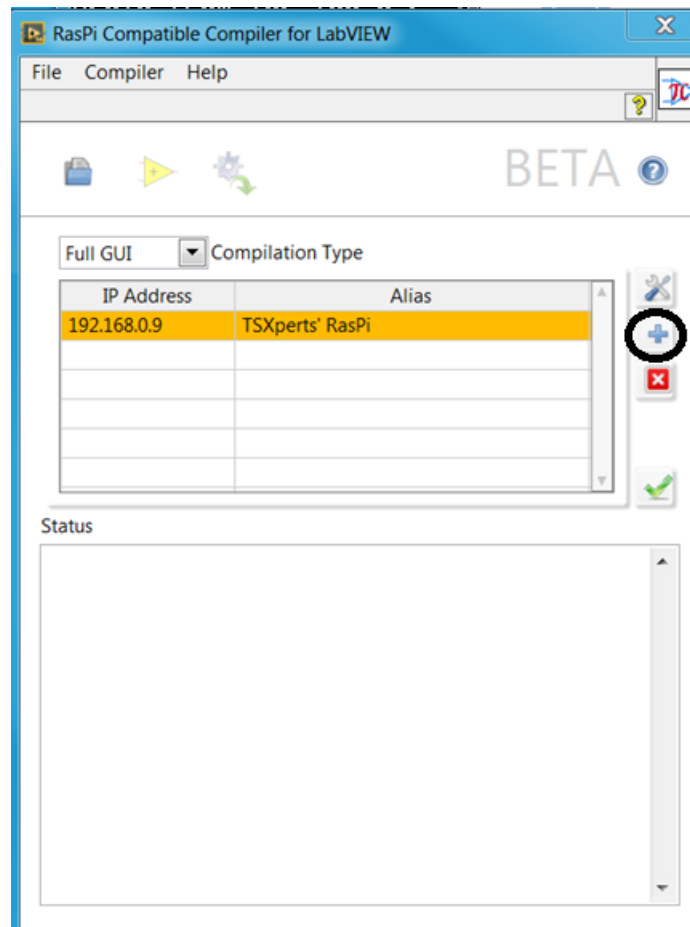
The next button below the one to add new targets can be used to delete the selected target from the list.

The final button is the one that allows verification of the connectivity between the development computer and the selected target from the list.
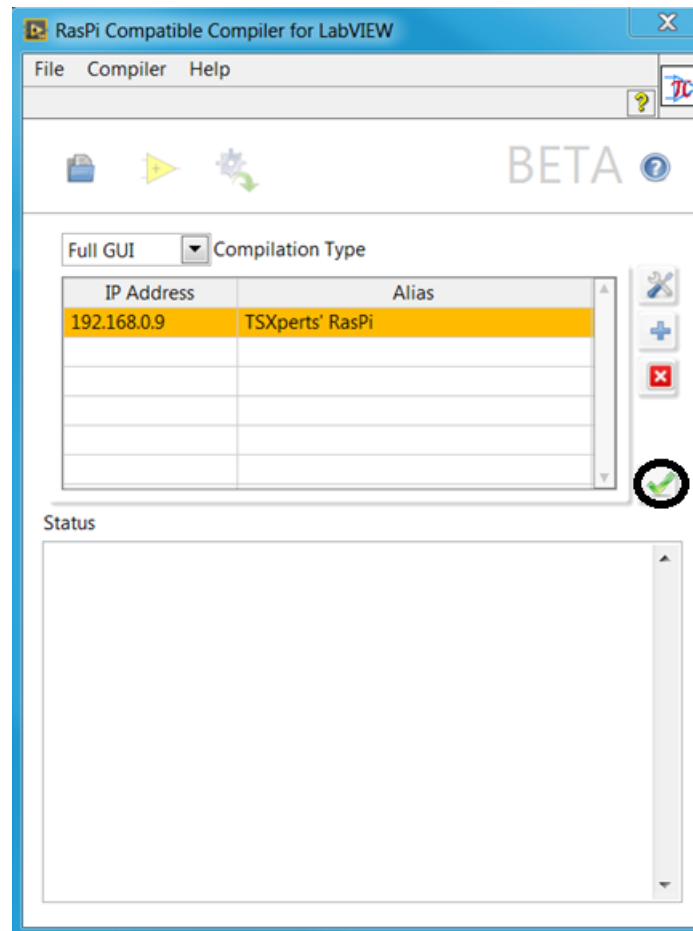
The last control in this area is the Compilation Type drop down. Let's postpone the description of this feature as it deserves a separate section to best describe what if offers the users.

The next area of interest of the main compiler application that deserves attention is the RasPi Compilation Tools area, highlighted in the next figure.
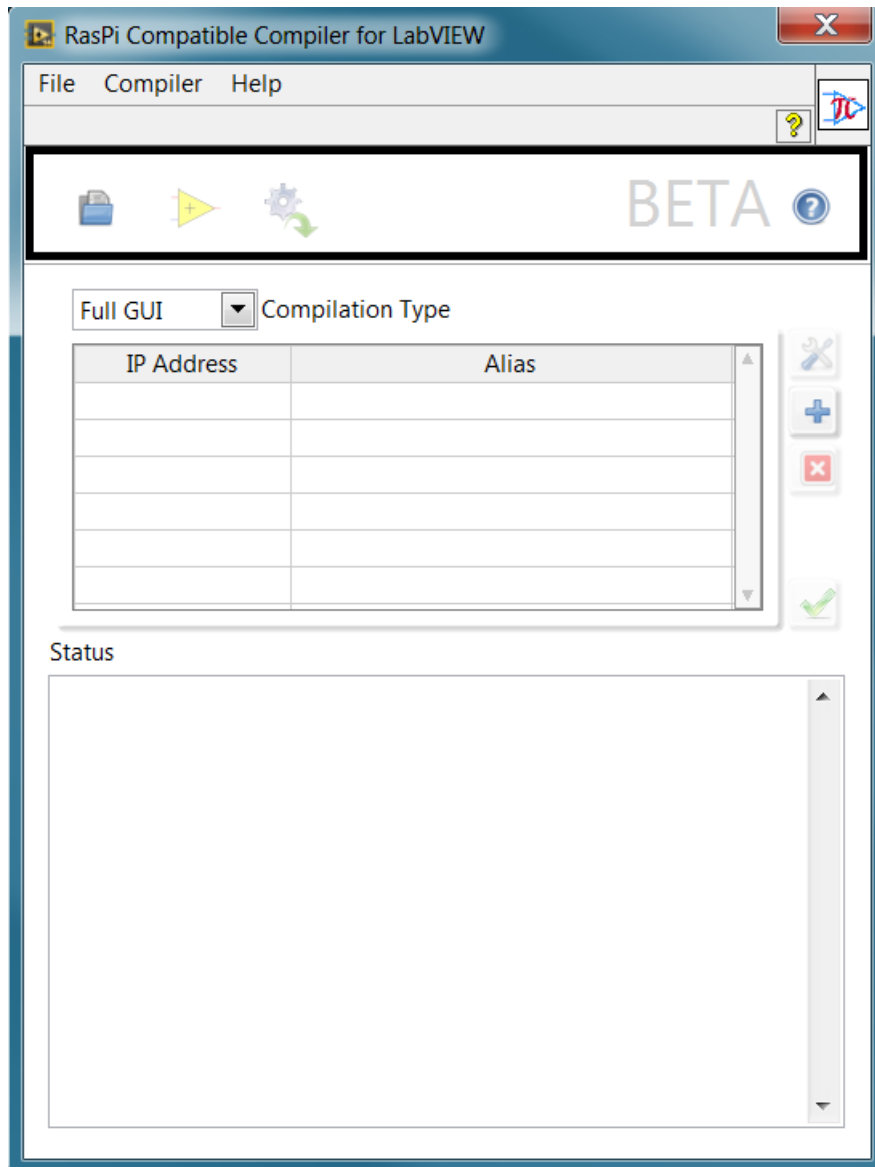
This area has four buttons, from left to right, the Load VI button, the Open Front Panel button, the Compile and Download button and the Open User Manual button.

The Load VI button allows the user to select the top level VI that will be compiled and downloaded to the selected and verified RasPi target. Once a VI is loaded, if it is in runnable state, the Open Front Panel button will be enabled and the compiler main window will show the loaded VI name as part of its title bar, as shown in figure below that has a VI named Edit Lists.vi loaded. If the loaded VI is not in a runnable state, an error will be generated and the Open Front Panel button will not be enabled.
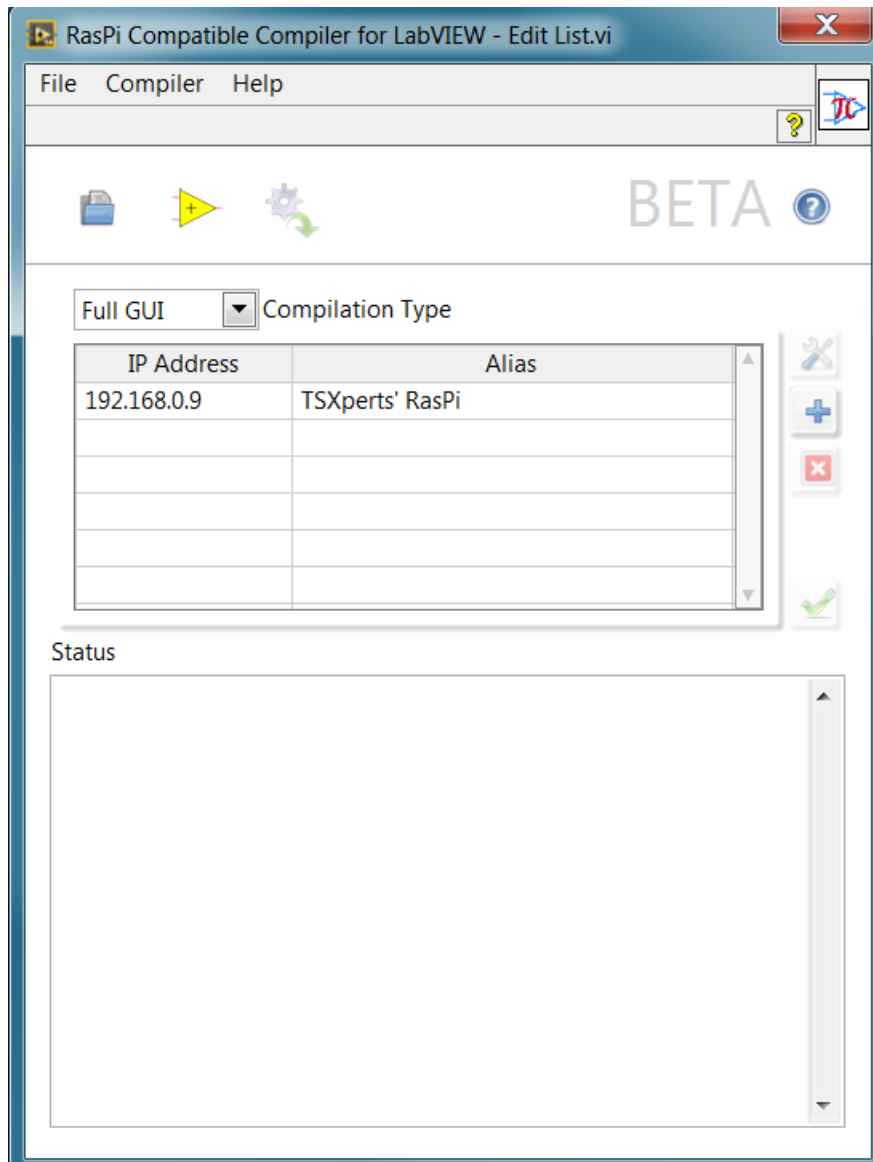
The Open Front Panel button is a convenience for the user. Clicking at that button will open the loaded VI front panel so the user can inspect it if needed.

Once a VI is successfully loaded, the last step prior for a compilation and download task is the selection of a RasPi target from the list of targets. Once a target is selected, the Compile and Download button will be enabled, as illustrated by the following figure.
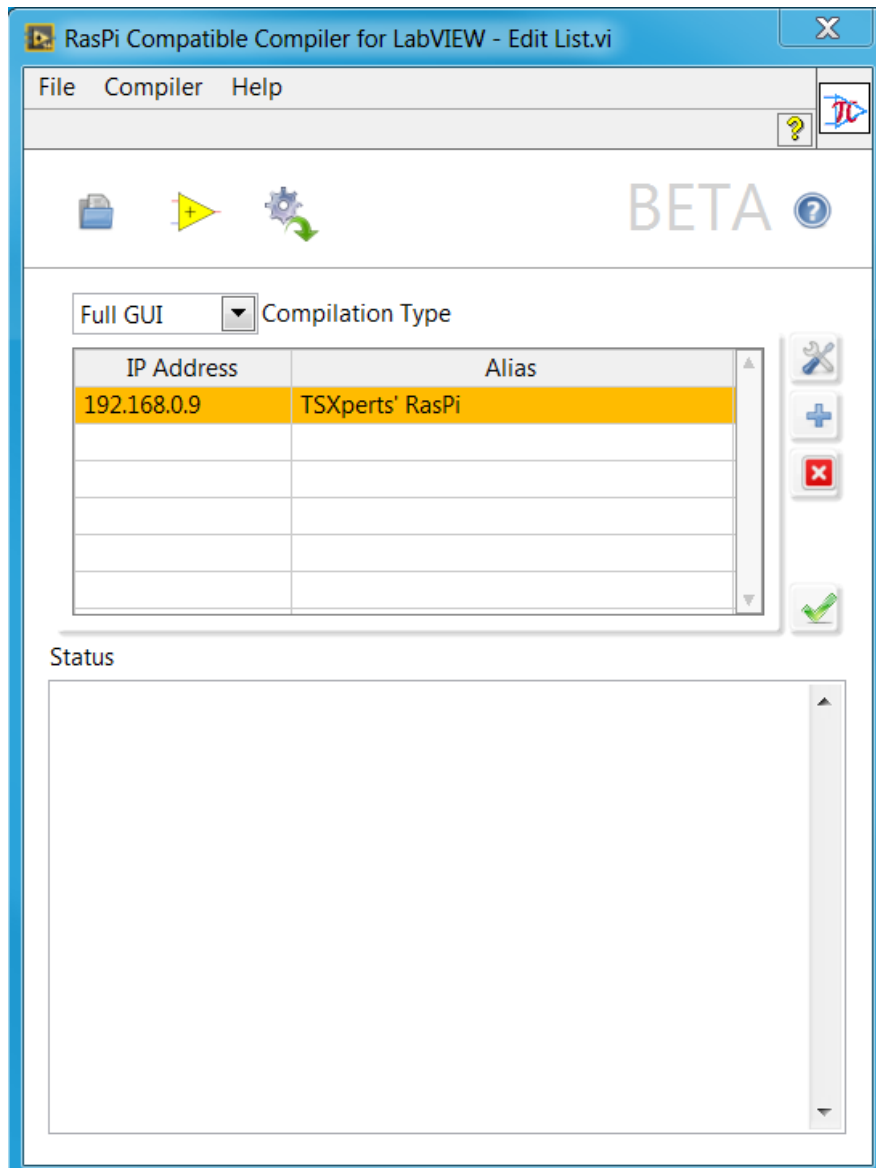
At this point, clicking at the Compile and Download button will trigger a compilation. Once the compilation is complete, the compiled VI will be downloaded to the selected RasPi target.

The last of the four buttons of this area is the help button, on the right hand corner. Clicking at this button will launch the online Wiki page that complements this document as part of the product documentation.

This brings us to the analysis of the last of the three areas of interest of the main compiler window; the RasPi Compilation Status area; as illustrated below.
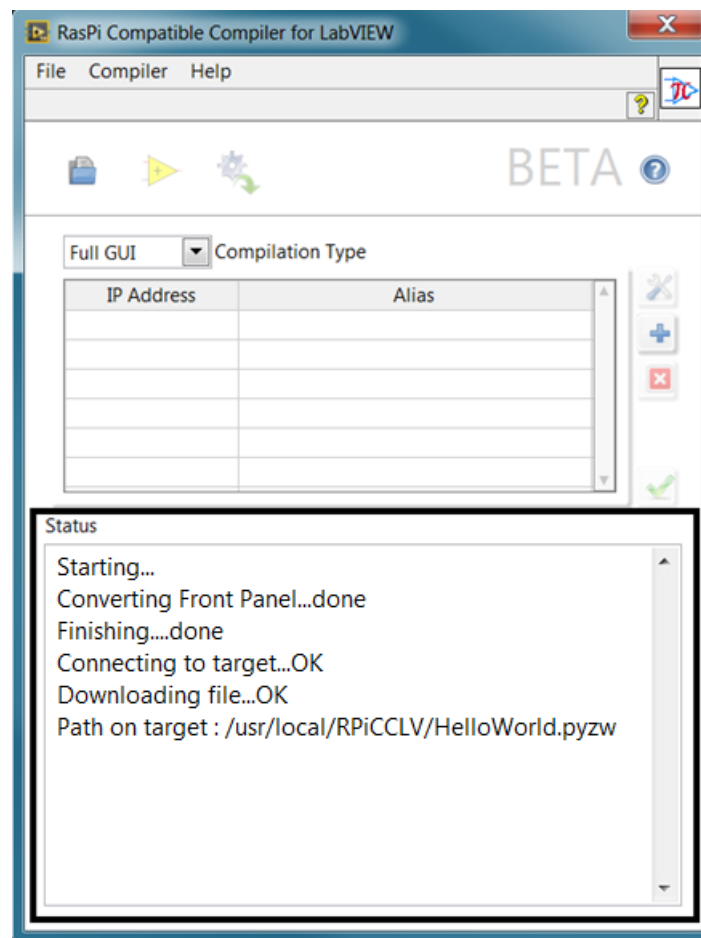
The Status area is where the messages generated by the compiler, during the process of compilation and download of the loaded VI to the target will be displayed. Also, any potential errors generated during the compilation process will be displayed on this indicator.

Compiling and Downloading an Example VI

Now that you understand all resources of the compiler application, it is time to download your first LabVIEW VI to your configured RasPi target. From the application File menu, navigate to Examples->Hello World GUI.vi, as illustrated below.
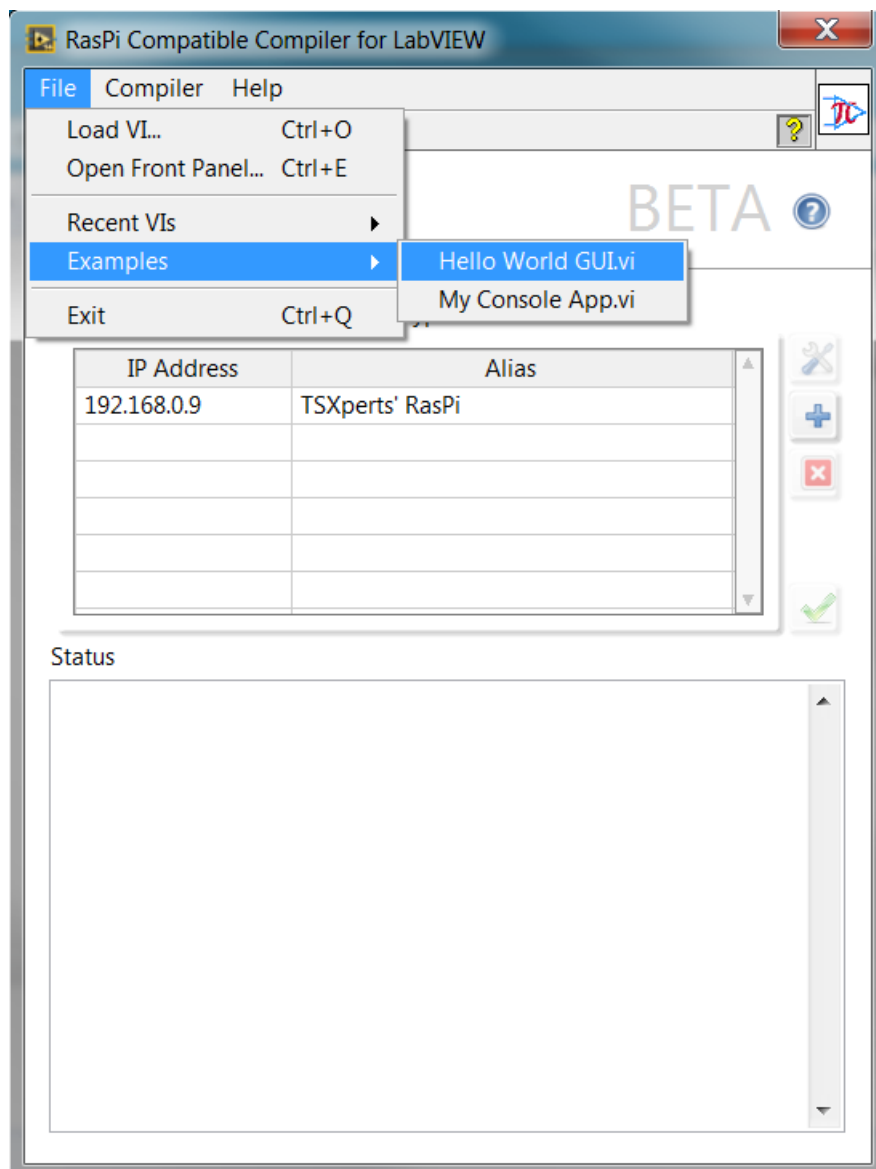
This will open the Hello World GUI.vi example and load the VI to be compiled and downloaded. Inspect the Hello World GUI.vi block diagram to get familiar with its code. Run the example VI on your development machine and notice how its user interface behaves. This will be useful in comparing with the behavior of the downloaded compiled VI, once we run it on the RasPi target.

Select your verified RasPi target from the list of targets. This will enable the Compile and Download button. Click the Compile and Download button and wait until the compilation finishes. Once it is complete, the compiled VI is downloaded to your RasPi target. Additional information on how to run the deployed VI on the Pi can be found on the Wiki page referred to above.

Running a Deployed VI on your RasPi

Once you press the Compile and Download button on the Raspberry Pi Compatible Compiler for LabVIEW main screen, the application will compile and download the selected VI to the Raspberry Pi target you have selected from the list of valid targets. At the end of the process; you will notice a message on the Status window indicating the Path on the target where the VI got downloaded to. That is where you will need to navigate to, on your Raspberry Pi, in order to run your deployed VI. This guide will assume the VI got deployed to /usr/local/RPiCCLV; but you can follow the same steps on any other directory the VI got deployed to.

On you Raspberry Pi, click on the File Manager icon on the top left of your Raspberry Pi screen as shown below.
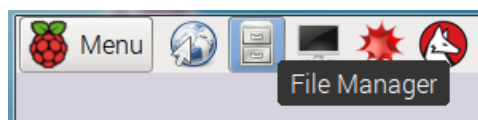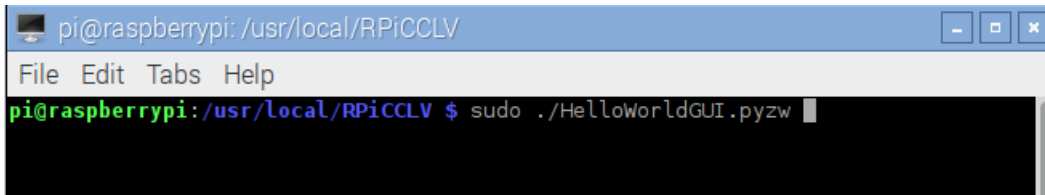

This will open the RasPi File Manager, an application similar to Windows Explorer on a Windows computer. Using the File Manager, navigate to the directory your VI got deployed to, as illustrated below.
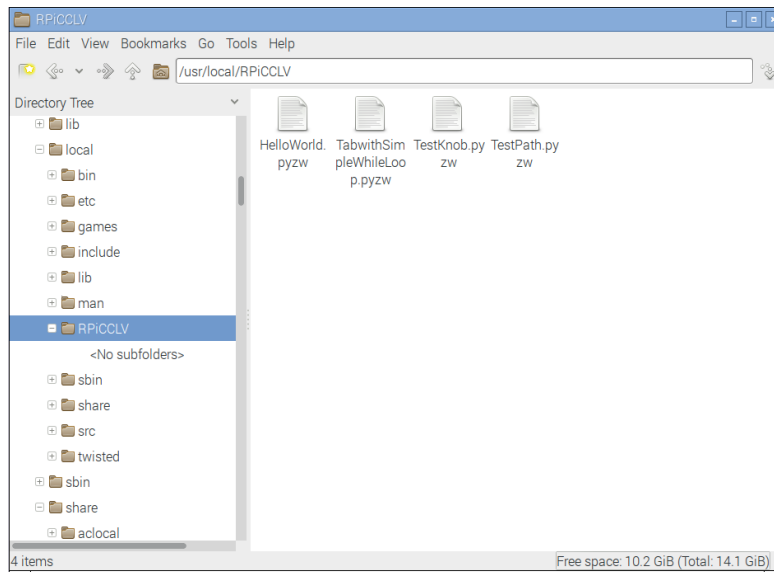
Next, right click at the RPiCCLV folder and select Open in Terminal as illustrated below.
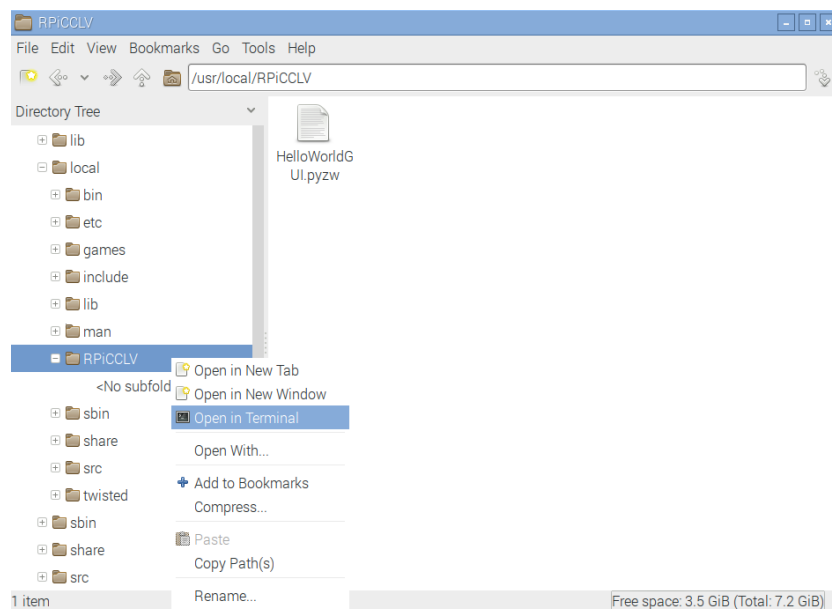
This will open a Shell Terminal screen on your Pi. On that screen, type the following command: **sudo ./<VI Name>.pyzw &**. The illustration below runs a HelloWorld VI that got deployed to the RasPi target.

```
pi@raspberrypi: /usr/local/RPiCCLV
File  Edit  Tabs  Help
pi@raspberrypi:/usr/local/RPiCCLV $ sudo ./HelloWorldGUI.pyzw
```

Once you press Enter to run the command, the HelloWorld deployed VI will open.

Simply clicking at its Run button will run the VI, exactly like you would do on regular LabVIEW.

Updating the Current VIP Package and the Raspberry Pi Runtime

The full Raspberry Pi Compatible Compiler for LabVIEW environment includes three main components. The Raspberry Pi released image, the LabVIEW VIP file that is to be installed on the development machine and the Raspberry Pi Runtime that resides in the Raspberry Pi itself. The Raspberry Pi image includes a Runtime as part of the image itself. However, there will be potential upgrades to the Runtime that are separate from upgrades of the image file itself.

The image will only be upgraded once there are significant changes to the Raspberry Pi OS made by the Raspberry Pi organization. We have incorporated an automated process for upgrading both your installed VIP file as well as the Pi Runtime.

Upon start of the Compiler GUI, the public repository where the latest versions of the VIP and Runtime is hosted are checked; this assuming the development machine is connected to the Internet. If the development machine is not connected to the Internet at that time, the status indicator will display a message stating that the update check couldn't be performed due to lack of connectivity.

If a newer version of the VIP file is verified in the repository, a window will popup indicating the installed version, the latest version on the repository as well as the release notes showing what changed from the version you have installed to the latest release.
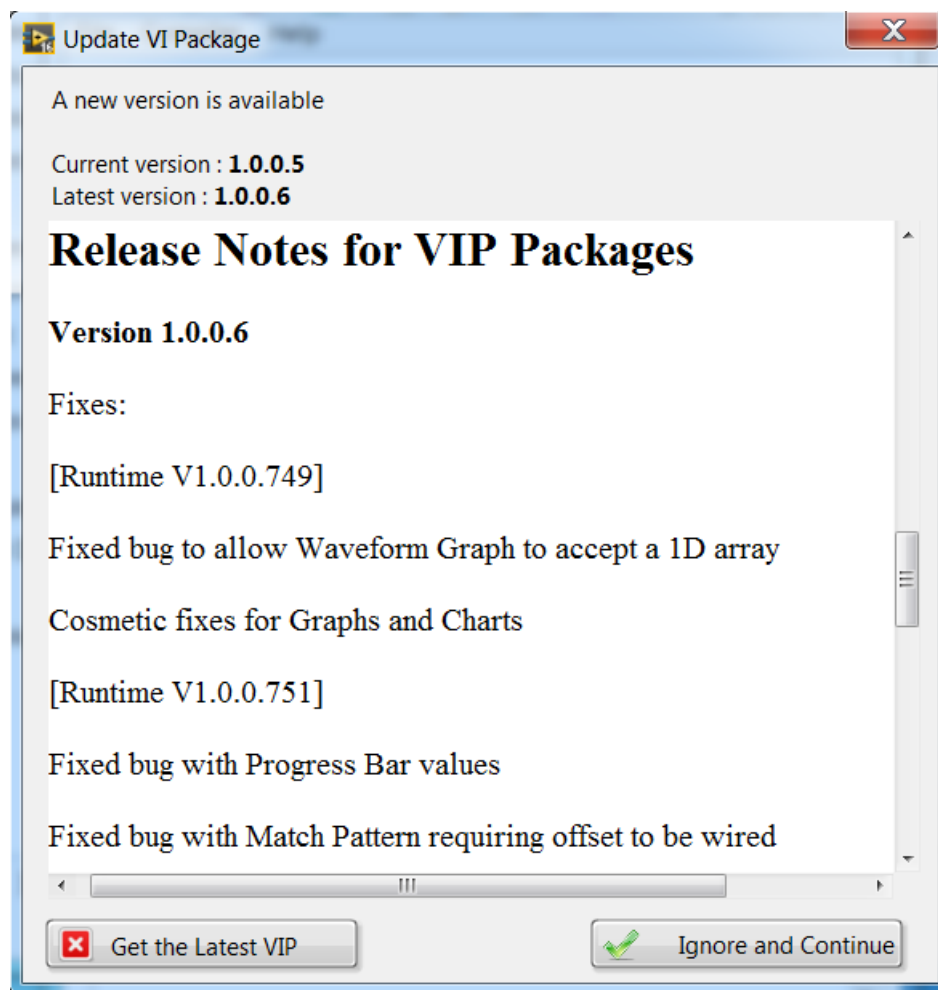
The user has the option to either accept the upgrade or cancel the upgrade and continue using the Compiler GUI. Should the user cancel the upgrade, the Compiler will remind the user once a day, upon launching, until the upgrade is performed.

Should the user decides to accept the upgrade, the Compiler GUI will exit, VIPM will launch and start the upgrade process for the user. Upon completion of the upgrade, the compiler GUI can be launched by the user regularly.

The Pi Runtime version update is done behind the scenes, every time a newer version is detected on the repository. This process doesn't require user interaction and it is completely seamless.

Checking for Updates

Besides the automated update checks as explained on the previous section, the user has the option to perform manual update checks via the Compiler GUI Menu item, as shown by figure below.
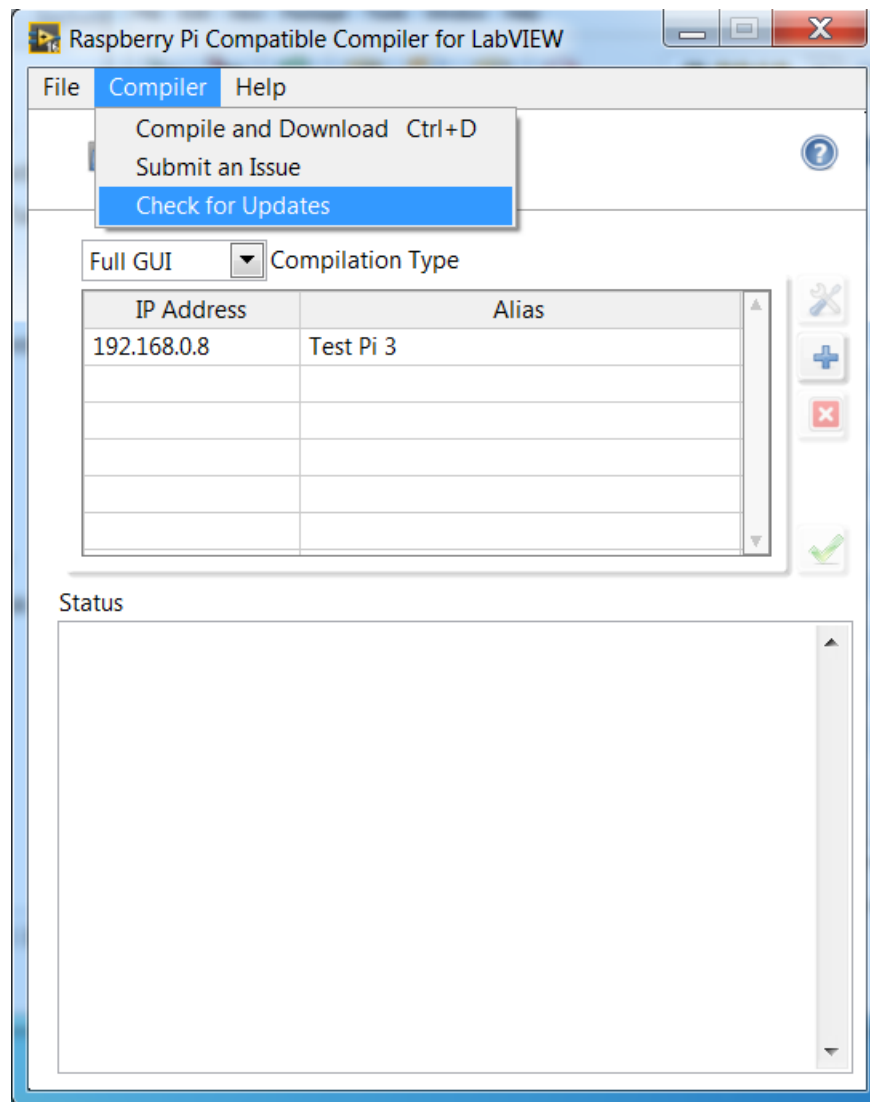
The selection of this menu item by the user triggers the same automated upgrade process described above, should the Compiler GUI detects a more recent version of the VIP package in the public repository. Since the Compiler GUI performs daily checks for updates, it is not expected the user to need to perform this manual check; but the option was given just in case the user doesn't want to wait for the daily check to occur.

Supported Compilation Types

The last important feature of the main compiler application window to discuss is the supported compilation types. There are two types of compilation types: Full GUI and Console App. The selection of which type to be executed by the compiler is done via the Compilation Type drop down control.

The Full GUI compilation type, as the name indicates, will compile both the VI block diagram and front panel as part of the compiled VI to be downloaded to the RasPi target. This will allow the RasPi to function basically as a regular computer running LabVIEW VIs. Full-fledged GUI VIs will run on the RasPi if that compilation type is selected. One thing that is important to make clear is that the RasPi will run what would be the corresponding version of a compiled executable VI. It would be the equivalent of a computer without the LabVIEW development environment running an Executable VI without its block diagram. The user will not be able to see the VI block diagram when the VI is deployed to the RasPi target, only its front panel.

The Console App compilation type strips the VI front panel and deploy only the VI block diagram code. One important feature to highlight as part of this compilation type is that the resulting console app created from the compilation will allow the user to call the app via command line and also pass arguments to it. The arguments would be the equivalent of the VI input terminals.

One interesting use case for this compilation type would be the creation of a Web service, in LabVIEW, to execute in the RasPi target. Some Web services don't require a user interface, but to respond to communication requests from other applications. Another example of use for such compilation type would be the creation of blocks of code in LabVIEW that only need to process data, without the requirement to present any user interface. In such cases, the selection of the Console App type of compilation is preferable, as it strips the VI code to its most fundamental components, saving target resources and executing in a more optimal manner.

The Compiler makes available two Console API VIs

Raspberry PI GPIO Support

The Raspberry Pi Compatible Compiler for LabVIEW includes a full API set dedicated to GPIO manipulation. There are three main areas of support for the RasPi GPIO: GPIO, I2C and SPI. Not surprisingly; the GPIO API palette has three subpalettes, one for each main area of support.
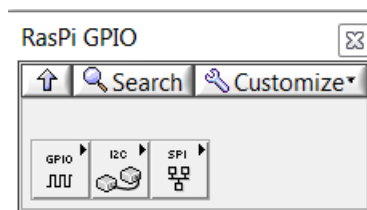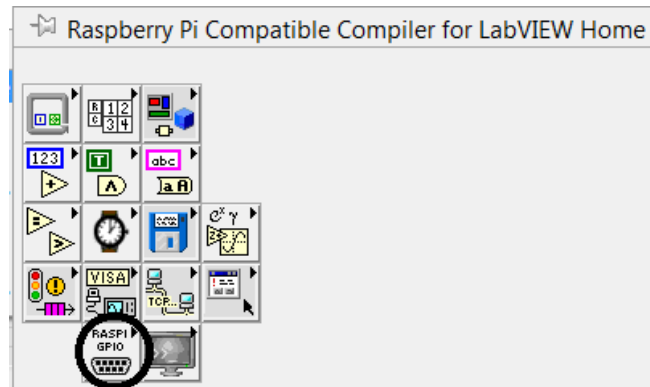
Furthermore, the GPIO API includes the installation of three groups of shipping examples, matching the names of the three main GPIO support group. The user is encouraged to start the learning curve on the GPIO API set from the shipping examples.
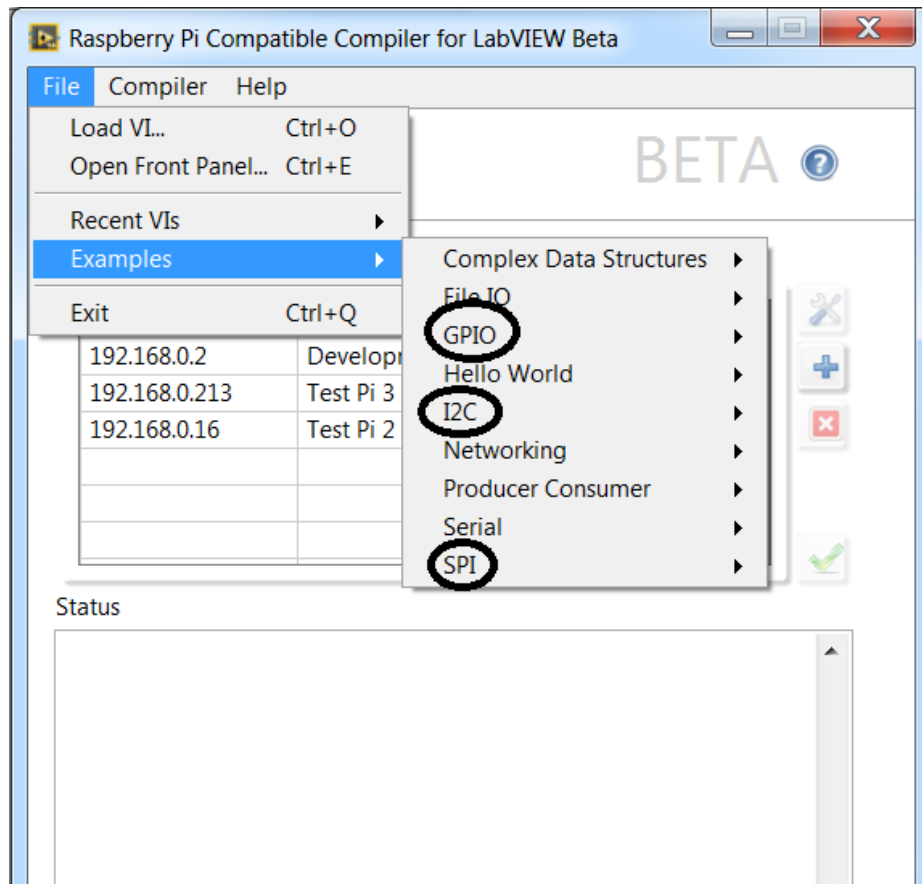
The GPIO API Set

The GPIO API set includes API VIs to manipulate the digital lines exposed to the Raspberry Pi GPIO connector. Make sure to read through the documentation for Raspberry Pi GPIO before start using this API set. This API Set has three layers; configuration VIs, Read/Write VIs and Event VIs, as illustrated by the following figure.
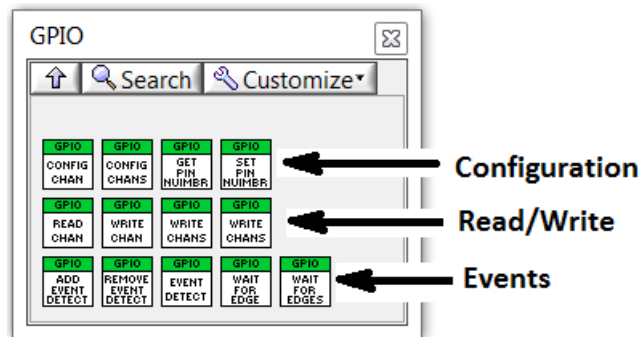
Refer to each VI LabVIEW documentation for a functionality overview.

The I2C API Set

The I2C API set includes API VIs to facilitate the implementation of I2C communication via the Raspberry Pi I2C GPIO pins. The I2C palette is divided into three main areas; I2C Open/Close, I2C Read and I2C Write, according to the following figure.
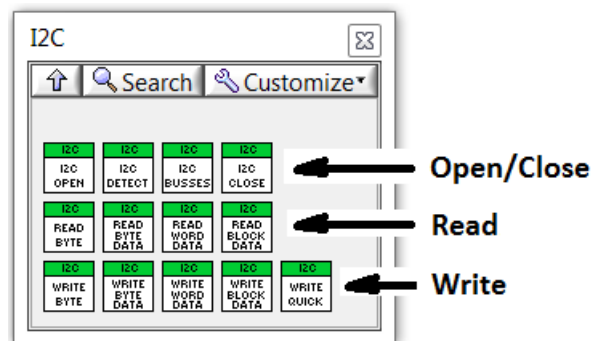
The Read and Write groups include the ability for the user to read/write to a single byte, word, or a block of data at a time. Refer to each VI LabVIEW documentation for a functionality overview.

The SPI API Set

The I2C API set includes API VIs to facilitate the implementation of SPI communication via the Raspberry Pi I2C GPIO pins. The SPI palettes also divided into three main areas: SPI Open/Close, Read/Get and Write/Set, as illustrated below.
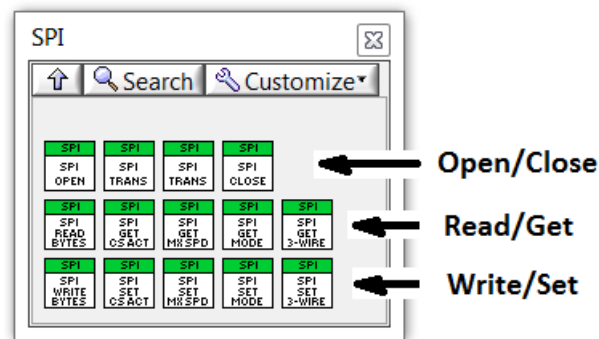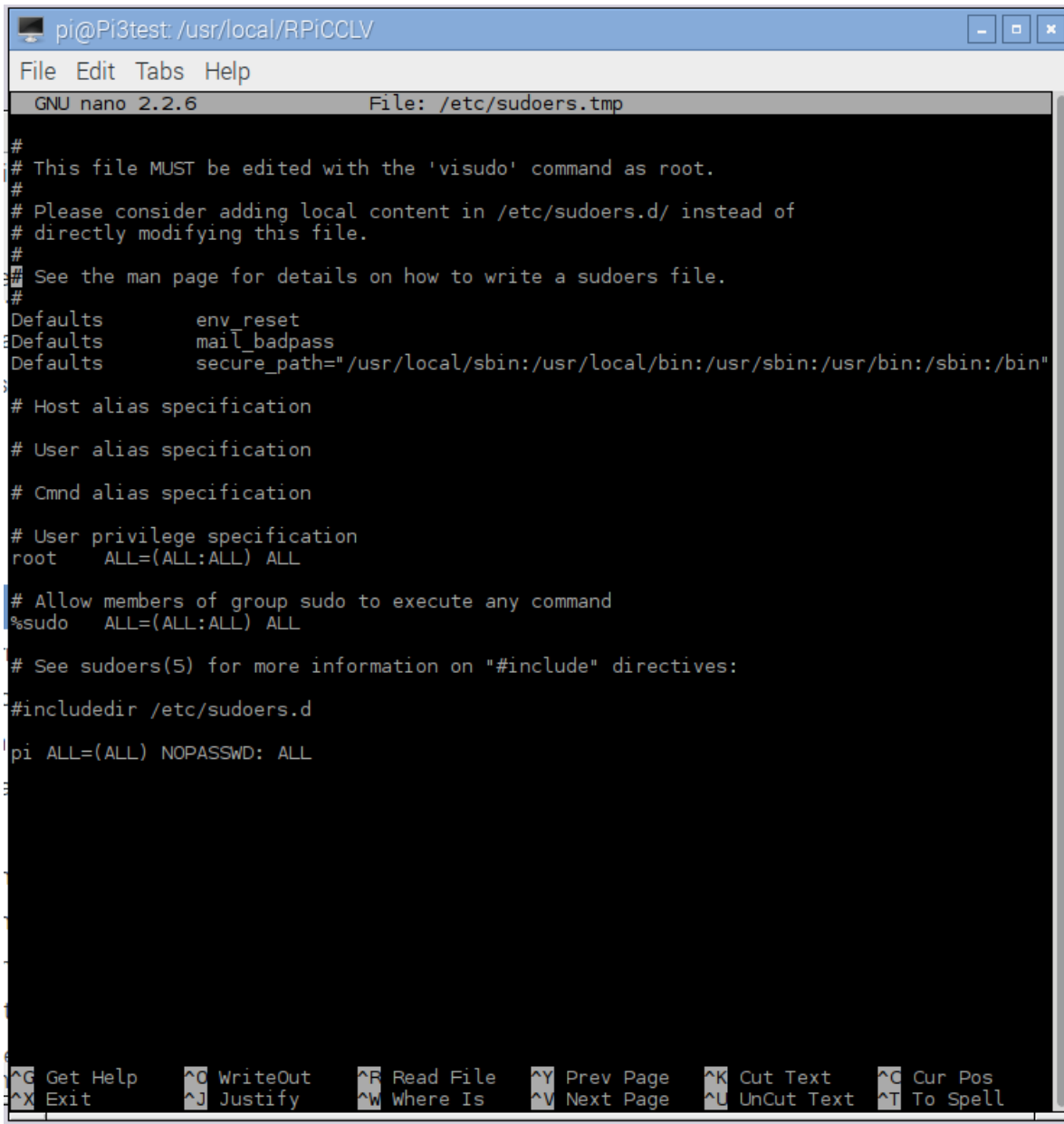

The Read/Get area, besides providing the ability for the user to read incoming SPI bytes, it also allows the user to retrieve parameters of the SPI bus, such as the state of the Chip Select, the Maximum speed for the SPI device, the configure SPI and whether or not the SI/SO lines for the SPI device are shared. The Write/Set area allows, besides the API VI that writes bytes to the SPI bus, the ability for the user to set the same SPI parameters that are exposed by the Read/Get API VIs.

Linux GPIO User Privilege

If you are logged into your Raspberry Pi with the standard pi sudo user; you can skip this section. However, if you create another user account and will be logged in as this user while deploying LabVIEW VIs that use the Pi's GPIO capability, you will need to make sure to add this user to be part of the GPIO group on your Pi.

In order to do that, if the user was previously added to the sudoer list on your Pi, all you need to do is to open a terminal and type: **sudo adduser user_name gpio**; where user_name is the name of the user that is currently logged in.

In the event you have created another user account and the new user is not part of the sudoers list; type the following command on a terminal; **sudo visudo**. This command will open the sudoers file as illustrated below.

The terminal has opened the file using the nano editor of your Jessie distribution. You will need to add the user account below the line that reads pi ALL=(ALL) NOPASSWD: ALL. You need to copy this line and replace the word pi with the name of the user you wish to add to the sudoers list.

Furthermore, you will need to delete the "%" character from the line %sudo ALL=(ALL:ALL) ALL.

Figure below shows the addition of the filipe user account to the sudoers list.

Once you are done with the edits, type Ctrl+X to Exit nano, and make sure you save your changes.

How to Report an Issue

This section describes the steps for submission of issues found with the compiler that are specific to a given VI. These types of issues include compilation errors as well as unforeseen functionality of a deployed VI that doesn't match the functionality of the same VI when running in regular LabVIEW for Windows. It is important though to make sure that, prior to sending an issue, the developer makes sure it is only using the primitive set provided in the Raspberry Pi Compatible Compiler for LabVIEW palette.

From the Raspberry Pi Compatible Compiler for LabVIEW GUI menu, select Compiler->Submit an Issue; as illustrated below.
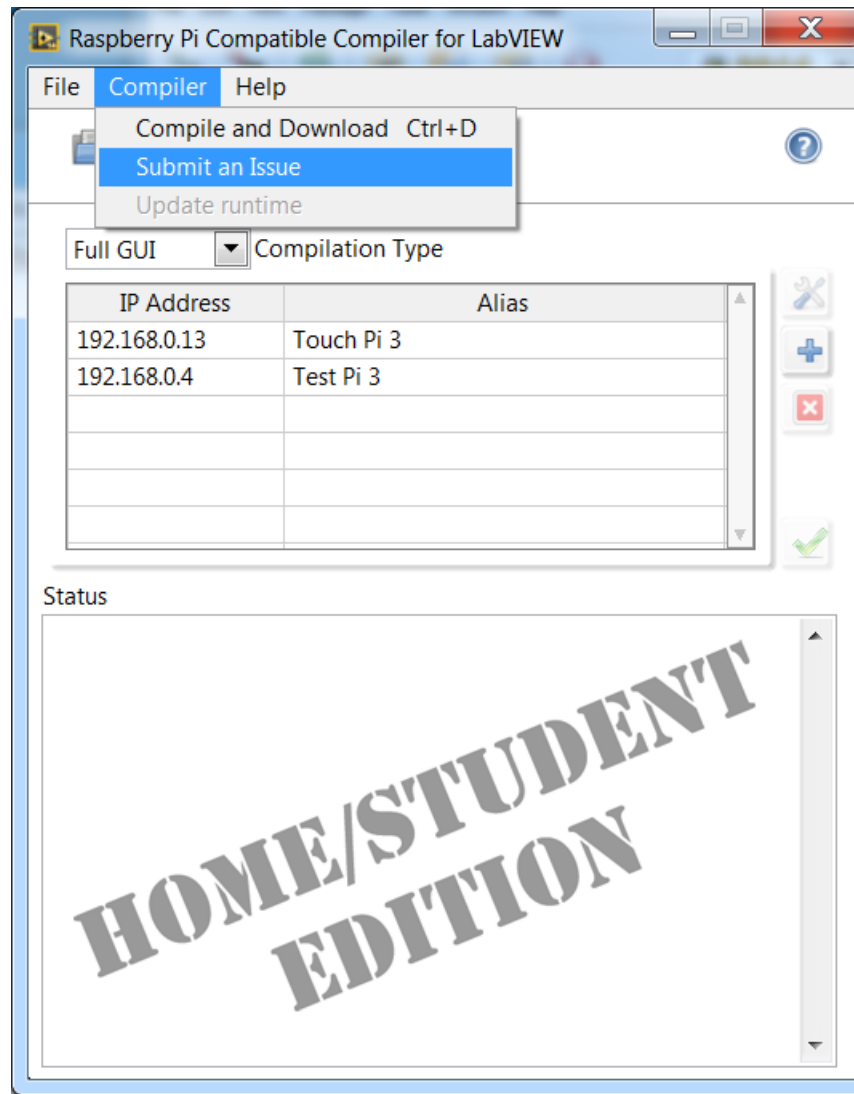
A separate screen will launch the following application.

This application allows the user to point to the top level VI that generated the issue as well as enter a thorough description of what the issue is as well as the steps to reproduce it.

Once the information above is filled in, the Submit button is enabled.

Clicking the Submit button triggers the application to package all needed information related to the VI being reported and send that information to the Raspberry Compatible Compiler for LabVIEW support channels. It is required the development PC that is running the compiler product has internet access for the submission to be successful.

Supported Controls and LabVIEW Primitives

All the LabVIEW function primitives and front panel controls that are currently supported by the compiler are packaged into specific LabVIEW palettes; for the convenience of the user. The following figure illustrates where to find the control palette for the RasPi compiler product.
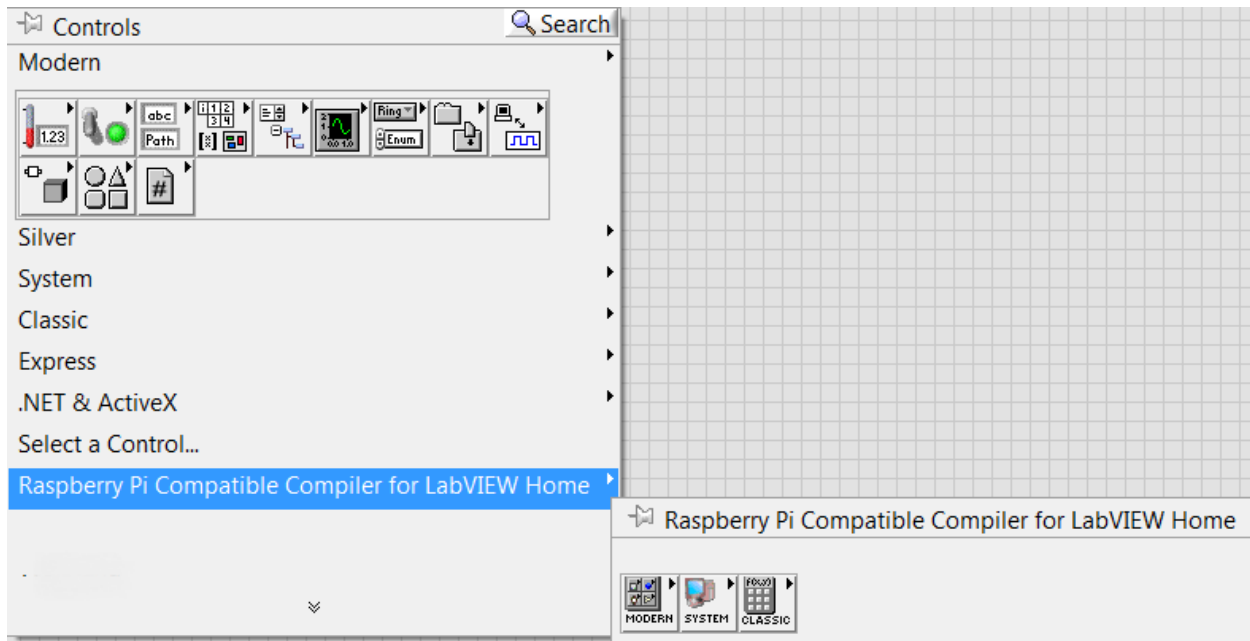
As you can see; there are three control subpalettes available for the user; Modern, System and Classic. The user is free to use any control included in these three sub-palettes. The use of controls not included in these palettes will generate a compilation error.

Similarly, a special palette containing all LabVIEW functions supported by the compiler can be accessed from the LabVIEW block diagram. The following figure shows the location of the RasPi compiler function palette.
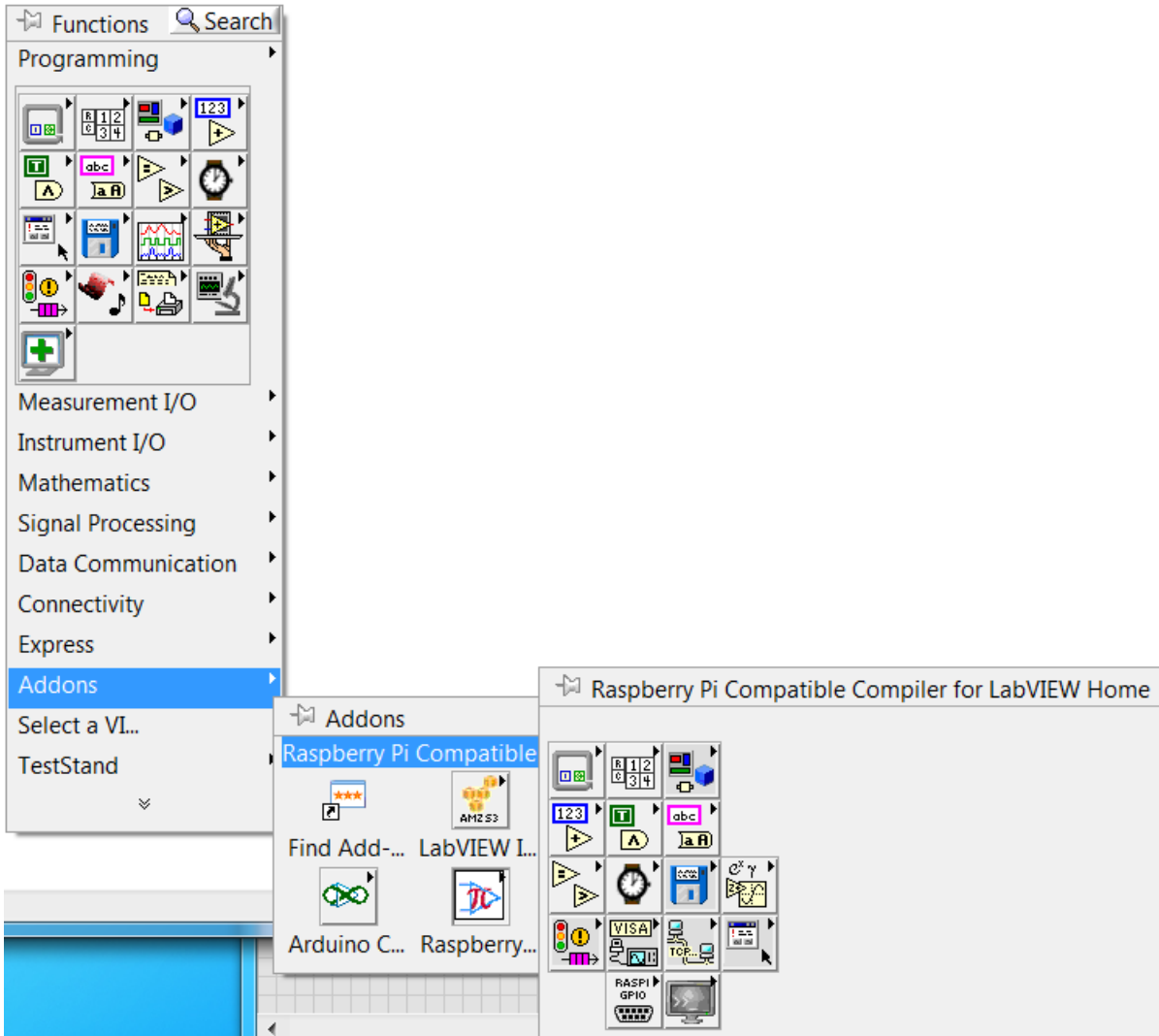
The use of any LabVIEW primitive not included in the Raspberry Pi Compatible Compiler for LabVIEW will generate a compilation error.

Visit This Wiki Page for up to date list of all supported functions and front panel objects.

Differences with Regular LabVIEW and Features not Currently Supported

It is important to highlight that the deployment of LabVIEW VIs to a Raspberry Pi target involves the migration of an application that natively runs on one operating system onto another operating system. Extreme care has been taken in order to maintain the look and feel of LabVIEW front panels as much as possible, once the operating system migration is complete. The vast majority of LabVIEW front panel controls, once deployed, will look exactly as they do on the development machine. However, it is possible that some controls may have subtle differences once deployed to the Raspberry Pi target.

Another important point to make is that LabVIEW is a mature product that has been improved upon for over thirty years. The Raspberry Pi Compatible Compiler for LabVIEW development team worked extremely hard to implement all right click options and features that are part of the supported LabVIEW primitives that have been included as part of the compiler. However, some of these options and features weren't implemented at the time this user guide is being created. This Wiki Page includes a comprehensive list of all known differences and exceptions one can expect when programming a Raspberry Pi target with LabVIEW using the Raspberry Pi Compatible Compiler for LabVIEW.

Support

If the support request is related to a problem with a specific VI, the most efficient way to request support is by using the Submit an Issue Screen, as described in the Section named How to Report an Issue.

If you have a question, comment or suggestion, we would love to hear from you at our community support forum. Feel free to drop us a line at the e-mail: lvforpisupport@tsxperts.com.