TPS6598x Utilities Tool User's Guide

ABSTRACT

The TPS6598x Utilities Tool is used to interact with the TPS6598x USB Type-C™ and Power Delivery (PD) controller using the host interface of the device and an external Microsoft Windows® based computer. This document provides instructions for installing and executing a set of Python-generated scripts. The interface provides the user a rich variety of functions for both controlling the TPS6598x device and acquiring device information.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Overview

The *TPS6598x Utilities Tool* is used to interact with the TPS6598x USB Type-C™ and PD controller using the host interface of the device and an external Microsoft Windows based computer.

The Python scripts can be used in two primary modes: *TPS6598x Utilities Command Line* and *TPS6598x Utilities GUI*. The *Utilities Command Line* allows the user to invoke one of the text-based Python application scripts through a command window. The *Utilities GUI* allows the user to interface with a front-end Python application, which presents the various debugging functions in a convenient format.

The *TPS65981, TPS65982, and TPS65986 Host Interface Technical Reference Manual* is implemented through I^{2}C. To communicate with a Windows system, one of three supported USB-to-I^{2}C/SPI adaptors is required. The three explicitly supported options are the Total Phase Aardvark™ USB-to-SPI/I^{2}C adaptor, an FTDI-based adaptor board, and the USB2.0 Low-Speed Endpoint (USB EP) with a USB Type-C to Type-A adaptor cable. For a complete understanding of the behavior of the TPS65982 firmware, refer to the *TPS65981, TPS65982, and TPS65986 Firmware User's Guide*.

1.2 Hardware Requirements

The required hardware is listed as follows:

NOTE: The hardware and setup is different depending on the adaptor selected. See Section 5.2 for more information on hardware set up.

- Windows-based PC with at least one USB2.0 (or later) port
- TPS6598x-EVM
- Barrel-jack laptop charger power-supply AC adaptor (20 V)
- USB Standard-A to Type-C passive adaptor cable for USB2.0 Low-Speed Endpoint (USB EP) option

  Texas Instruments recommends purchasing a quality-brand cable, such as Belkin.
- Total Phase Aardvark USB-to-SPI/I^{2}C Master adaptor + USB B to A cable + jumper wires or LaunchPad™ EVM to Aardvark adaptor (see Section 5)

1.3 Software Requirements

The TPS6598x debugging scripts are implemented using the Python programming language. The Python language was selected because it is highly portable and well supported. Python allows users to update and execute custom application scripts without the need to recompile.

The required software packages, in addition to the TPS6598x debugging utility files, are bundled into a single windows installer. This installer uses a relocatable distribution of Python (based on WinPython) for simplicity of installation. See Section 1.4 for download instructions using the bundled installer.

If the user chooses to not use the bundled installer, the required components can be downloaded separately. For reference, the software components required for running the TPS6598x Utilities Python scripts are listed as follows:

- Python version 2.7.9 (see Section 4 for separate installation instructions)
- Python modules:
  - crcmod
  - intelhex
  - simplejson
  - cherrypy
  - WinPython distribution (subset)
    - Relocatable WinPython command prompt
    - Relocatable IDLE editor
- Windows driver and dynamic-link library support for one or both of:

- FTDI-based adaptor
- Total Phase Aardvark USB-to-I^{2}C/SPI Master adaptor
- USB endpoint

1.4 Installation Using Bundled Installer

Download the software from TI's website, located on the USB PD Landing Page and the TPS65982 Product Folder. Use the following steps to install the software:

Step 1. Navigate to either of the links provided and click on the *Tools & Software* tab.

Step 2. Click the *TPS6598x Host Interface Utility Tool* link.

Step 3. Click the *Get Software* button on the TPS6598X Host Interface Utility Tool page.

Step 4. Complete all the requested information for the U.S. Government export approval, review the terms of agreement, and click *Submit* for approval.

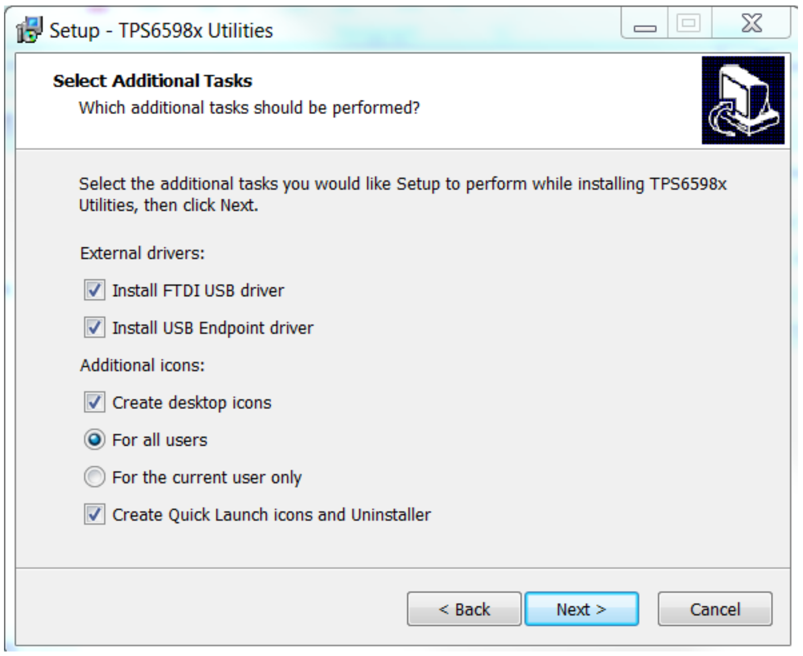
NOTE: Incomplete information will not be approved.

Step 5. If approved, the user will receive an email with a download link. Click the link to *Download* the configuration tool.

Step 6. Select the appropriate installer (.exe file) for the operating system of the user's computer.

Step 7. Run the .exe file and follow the onscreen instructions:

- The installer defaults to placing all files in the user's *Program Files* directory. The installed files are relocatable and will run from any location. If a user intends to view and modify the Python scripts, the files can be relocated to a more accessible location such as the PC desktop. The installer adds shortcuts on the *Start Menu* and *Desktop* for accessing both the GUI and command line, regardless of location. See Table 1 for folder descriptions.
- The bundled installer gives the user the opportunity to launch the installers for the FTDI and USB endpoint drivers. Before installation, TI recommends to first uninstall any previous version of these drivers. See Figure 1 for driver installer options.
- The bundled installer does not install the Aardvark USB-to-I^{2}C/SPI driver. If the user intends to use the Aardvark, see Section 5.1 for Total Phase Aardvark driver and software installation instructions.


Setup - TPS6598x Utilities

Select Additional Tasks

Which additional tasks should be performed?

Select the additional tasks you would like Setup to perform while installing TPS6598x Utilities, then click Next.

External drivers:

- [✓] Install FTDI USB driver
- [✓] Install USB Endpoint driver

Additional icons:

- [✓] Create desktop icons
- [●] For all users
- [ ] For the current user only
- [✓] Create Quick Launch icons and Uninstaller

Figure 1. FTDI and USB EP Driver Installation Options

Table 1. TPS6598x Host Interface Python Utility Files

Folder or File	Description
drivers	The Windows USB driver installers for FTDI and USB EP, selectable in the installer (Figure 1).
TPS6598x utilities	Primarily comprised of TI-provided Python scripts. The main folder also includes .dll files for Total Phase (see Section 5), .dll files for FTDI. The subfolder html contains the jquery package, which is not TI generated.
WinPython-64bit-2.7.10.3	This is the Python installation used to interpret the TPS6598x utilities Python scripts.
readme.txt	Information about the installation of the TPS6598x utilities.
uninst000.exe	This file uninstalls the TPS6598x utilities.

1.5 Features

When installation is complete, two desktop shortcuts are available as shown in Figure 2: *TPS6598x Utilities GUI* and *TPS6598x Utilities Command Line*.

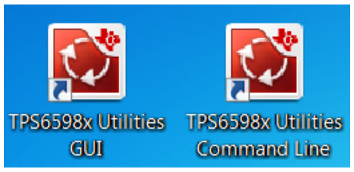


Figure 2. GUI and Command Line Shortcuts

Texas Instruments provides a collection of Python-based scripts for interacting with the TPS6598x device using the host interface accessed through I^{2}C. These scripts can be used in two primary modes:

TPS6598x Utilities Command Line — Gives user the ability to invoke one of the text-based Python application scripts from a command window.

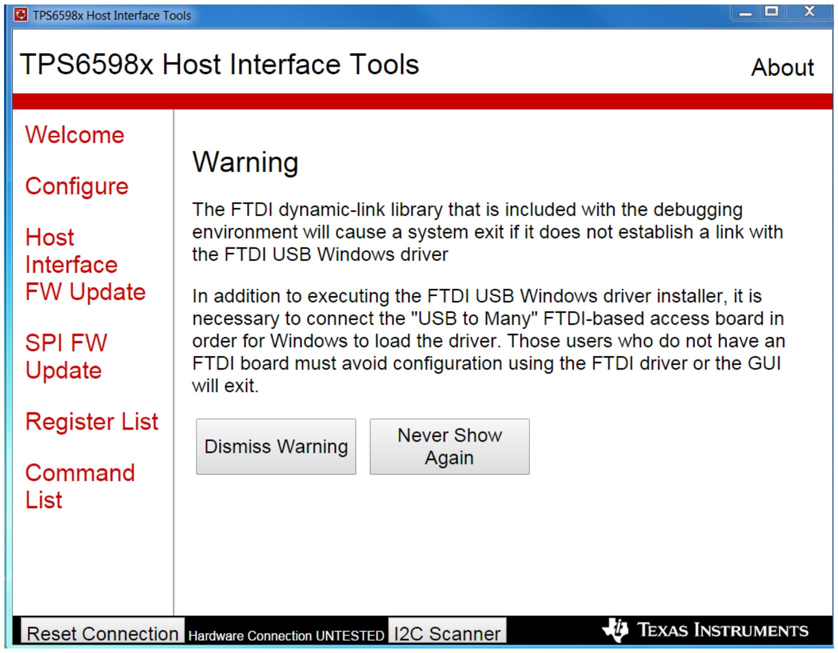
TPS6598x Utilities GUI — Allows the user to interface with a front-end Python application.

The GUI offers a simple, standalone application for viewing and modifying firmware settings through the host interface. The Utilities Command Line and Python scripts allow the user to directly customize behaviors or view examples for performing common tasks. A user seeking a utility for updating firmware will likely find the GUI more useful than the command window. A developer wanting an example for updating firmware, that can be ported to a C-application for an external micro-controller in a custom design, will likely find the scripts more useful.

2 Using the TPS6598x Utilities GUI

2.1 GUI Overview

When opening the TPS6598x Utilities GUI, the user immediately sees the *Welcome* page (see Figure 3). The GUI contains three primary regions: the footer, the page selection pane, and the page display. The footer primarily contains connection status information. The page selection pane, located along the left side of the GUI window, contains buttons that direct the user to pages with various functions. The page display, which represents the majority of the GUI window, shows the page linked to each button within the page selection pane. The location and functionality of each button in the GUI window are listed in Table 2.


TPS6598x Host Interface Tools

TPS6598x Host Interface Tools

About

Welcome

Configure

Host Interface FW Update

SPI FW Update

Register List

Command List

Warning

The FTDI dynamic-link library that is included with the debugging environment will cause a system exit if it does not establish a link with the FTDI USB Windows driver

In addition to executing the FTDI USB Windows driver installer, it is necessary to connect the "USB to Many" FTDI-based access board in order for Windows to load the driver. Those users who do not have an FTDI board must avoid configuration using the FTDI driver or the GUI will exit.

Dismiss Warning

Never Show Again

Reset Connection Hardware Connection UNTESTED I2C Scanner

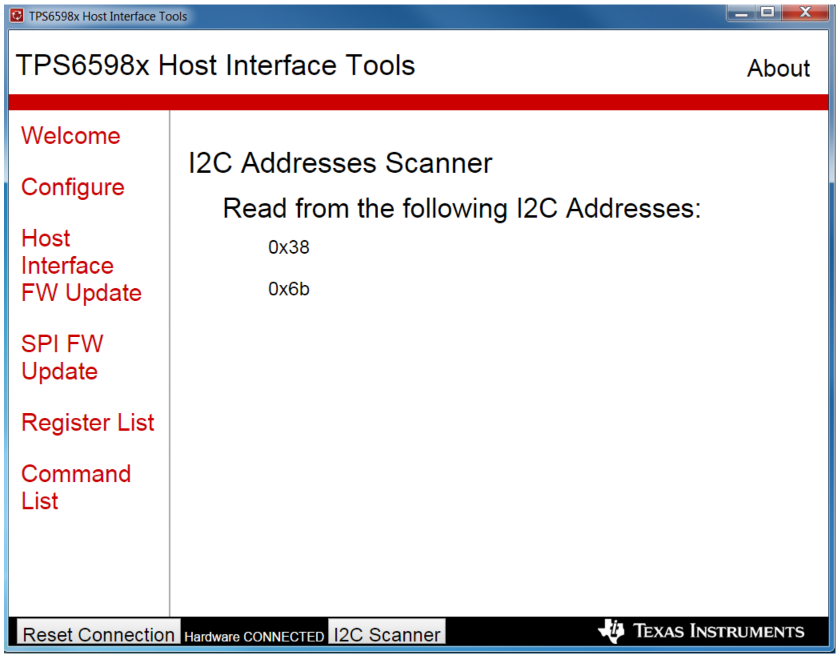
TEXAS INSTRUMENTS

Figure 3. Utilities GUI Application

Table 2. Descriptions of Buttons in GUI Window

Button	Location	Functionality
About	Upper-right corner of window, above red Texas Instruments banner	Details the version and disclaimer information. Contains the forum link to the E2E™ online community.
Welcome	Page selection pane	Page display seen when opening GUI. Must click the *Dismiss Warning* button to see more information.
Configure	Page selection pane	Contains functionality to edit configuration settings and test for successful interface. See Section 2.2 for more information.
Host Interface FW Update	Page selection pane	Contains functionality to update firmware through I^{2}C. See Section 2.3.1 for more information.
SPI FW Update	Page selection pane	Contains functionality to update firmware through SPI. See Section 2.3.2 for more information.
Register List	Page selection pane	Contains functionality to read and write to various registers in the TPS6598x device. See Section 2.4 for more information.
Command List	Page selection pane	Contains functionality to execute and read various commands for the TPS6598x device. See Section 2.5 for more information.
Reset Connection	Lower-left corner of window in footer	Push this button to test the connection status (located to the right of *Reset Connection*)[(1)]. Connection status possibilities include: • *Hardware UNTESTED*: button has not been pressed. • *Hardware DISCONNECTED*: last test indicated no connection. • *Hardware CONNECTED*: last test indicated successful connection.
I2C Scanner	Bottom of window in footer	Sweeps through all possible (7-bit) I^{2}C addresses and returns which ones respond (See Figure 4). User uses returned addresses in the configuration page. Refer to *TPS65981, TPS65982, and TPS65986 Firmware User's Guide* for more information regarding determining I^{2}C address.

(1) The user must select *USB-to-I2C/SPI Adaptor* in the *Configure* page before testing connection.



I2C Addresses Scanner

Read from the following I2C Addresses:

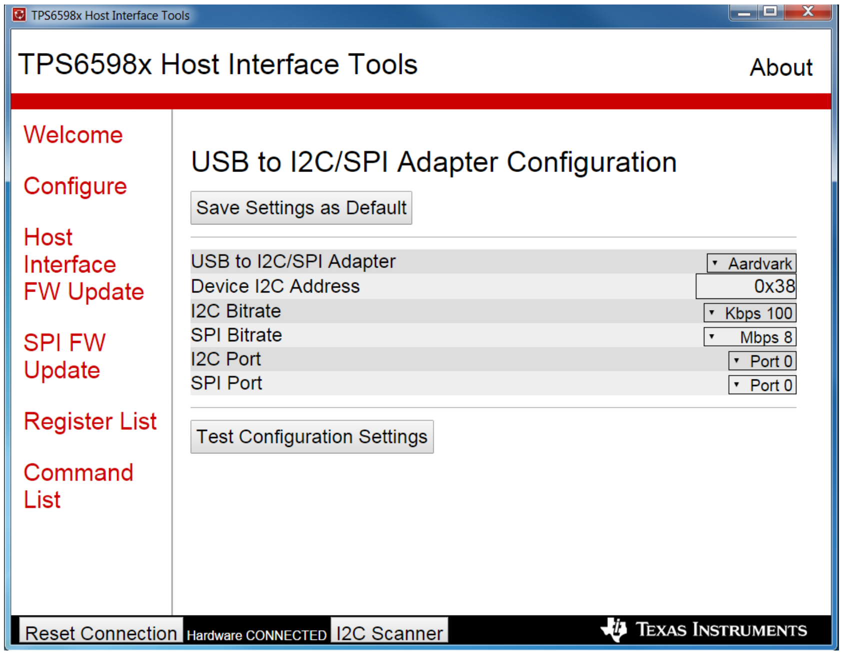
0x38

0x6b

Figure 4. Successful I^{2}C Scan

2.2 *Configure*

The TPS6598x Utilities Tool can be configured to use the Total Phase Aardvark USB-to-I^{2}C/SPI adaptor, an FTDI-based adaptor, or USB EP (with USB Type-A to Type-C cable). The configuration settings are located in the *Configure* page, which is the second option in the page selection pane (see Figure 5). Settings for each adaptor option are listed in Table 3.



USB to I2C/SPI Adapter Configuration

USB to I2C/SPI Adapter	Aardvark
Device I2C Address	0x38
I2C Bitrate	Kbps 100
SPI Bitrate	Mbps 8
I2C Port	Port 0
SPI Port	Port 0

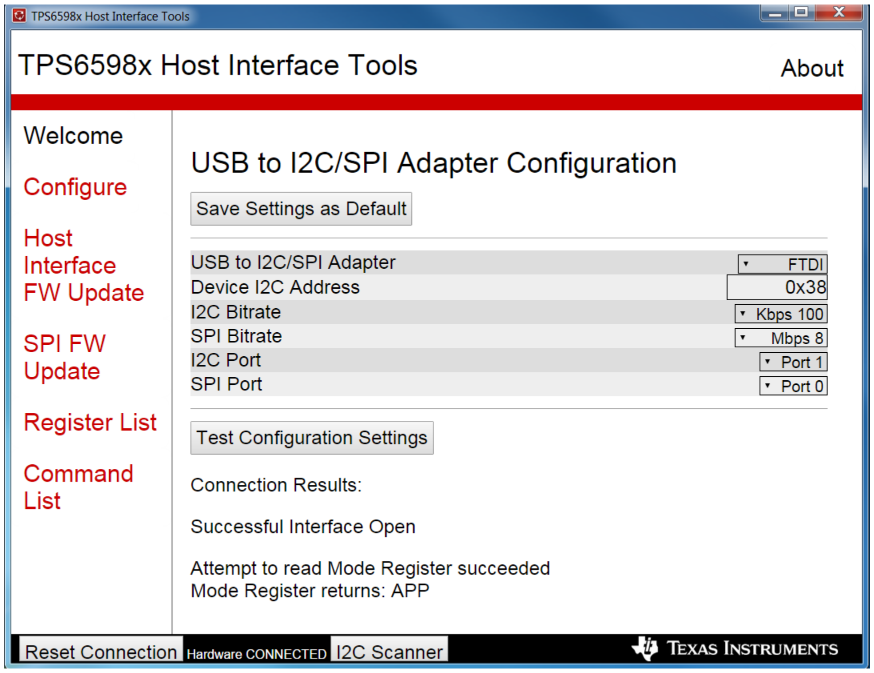
Figure 5. GUI Configuration Page

Table 3. Configuration Settings

Setting	FTDI	Aardvark	USB EP
Device I^{2}C address	0x38 (Board dependent)	0x38 (Board dependent)	0x38 (Board dependent)
I^{2}C bitrate (Kbps)	400	400	Any Option
SPI bitrate (Mbps)	8	8	Any Option
I^{2}C Port	1	0	0 or 1
SPI Port	0	0	0 or 1

Settings that are changed on the *Configuration* page are effective immediately. However, the settings are not preset during the next session unless the *Save Default Settings* button is clicked. This button actually overwrites the *config.py* file. See Section 3.1 for more detail regarding *config.py* file.

When opening the GUI or changing configuration settings, verify that the configuration settings have resulted in a successful interface. Clicking the *Test Configuration Settings* button can verify a successful interface and read the mode of the TPS6598x device. For more information about modes, refer to the *TPS65981, TPS65982, and TPS65986 Firmware User's Guide*. Examples of successful configuration tests for each adaptor are shown in Figure 6, Figure 7, and Figure 8.



USB to I2C/SPI Adapter Configuration

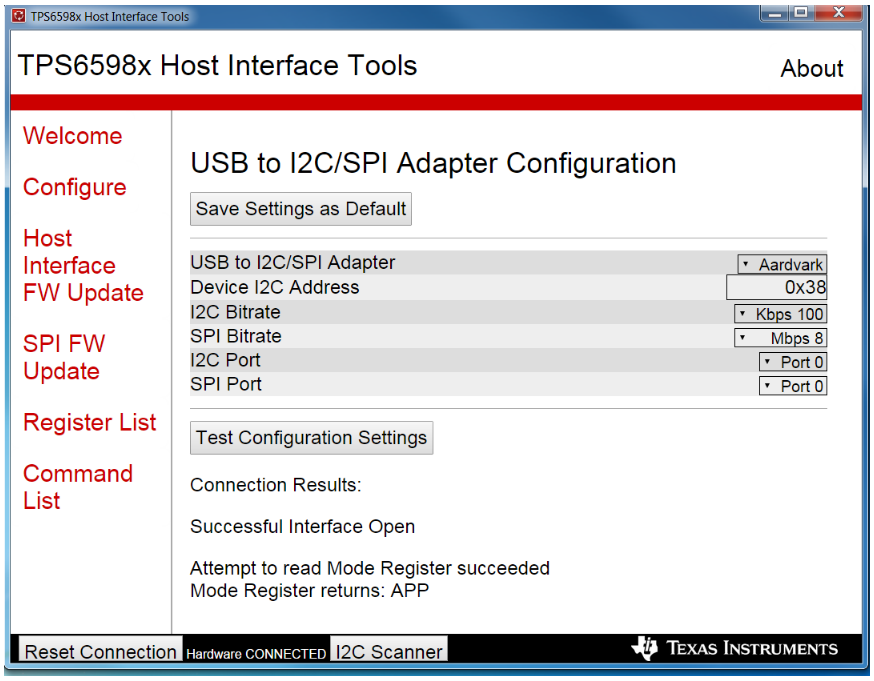
USB to I2C/SPI Adapter	FTDI
Device I2C Address	0x38
I2C Bitrate	Kbps 100
SPI Bitrate	Mbps 8
I2C Port	Port 1
SPI Port	Port 0

Connection Results:

Successful Interface Open

Attempt to read Mode Register succeeded
Mode Register returns: APP

Figure 6. Successful Configuration for FTDI



USB to I2C/SPI Adapter Configuration

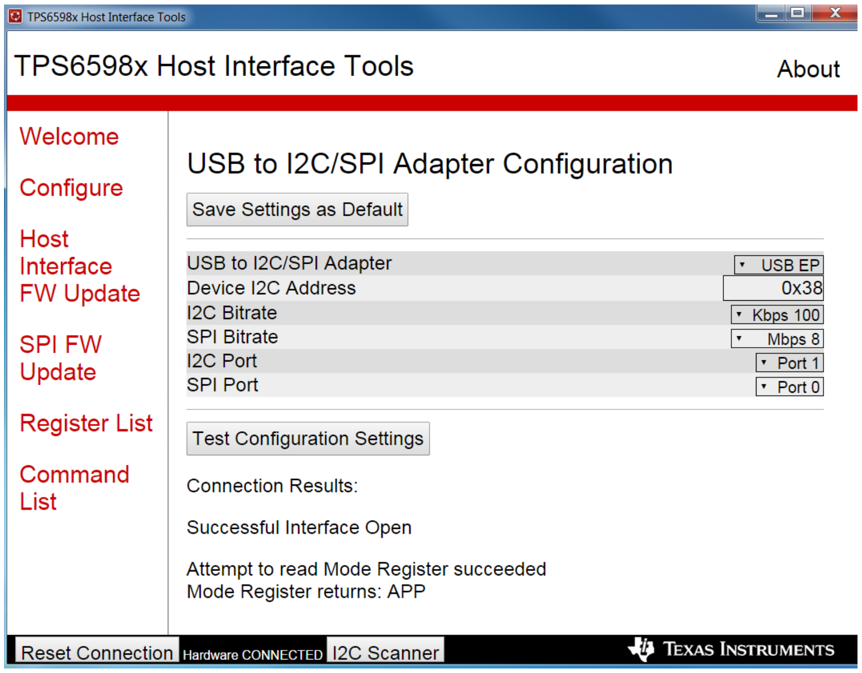
USB to I2C/SPI Adapter	Aardvark
Device I2C Address	0x38
I2C Bitrate	Kbps 100
SPI Bitrate	Mbps 8
I2C Port	Port 0
SPI Port	Port 0

Connection Results:

Successful Interface Open

Attempt to read Mode Register succeeded
Mode Register returns: APP

Figure 7. Successful Configuration for Aardvark



USB to I2C/SPI Adapter Configuration

USB to I2C/SPI Adapter	USB EP
Device I2C Address	0x38
I2C Bitrate	Kbps 100
SPI Bitrate	Mbps 8
I2C Port	Port 1
SPI Port	Port 0

Connection Results:

Successful Interface Open

Attempt to read Mode Register succeeded
Mode Register returns: APP

To use the USB EP option, the TPS6598x must have firmware that is v. 1.7.5, or later.

Figure 8. Successful Configuration for USB EP

2.3 Firmware Update

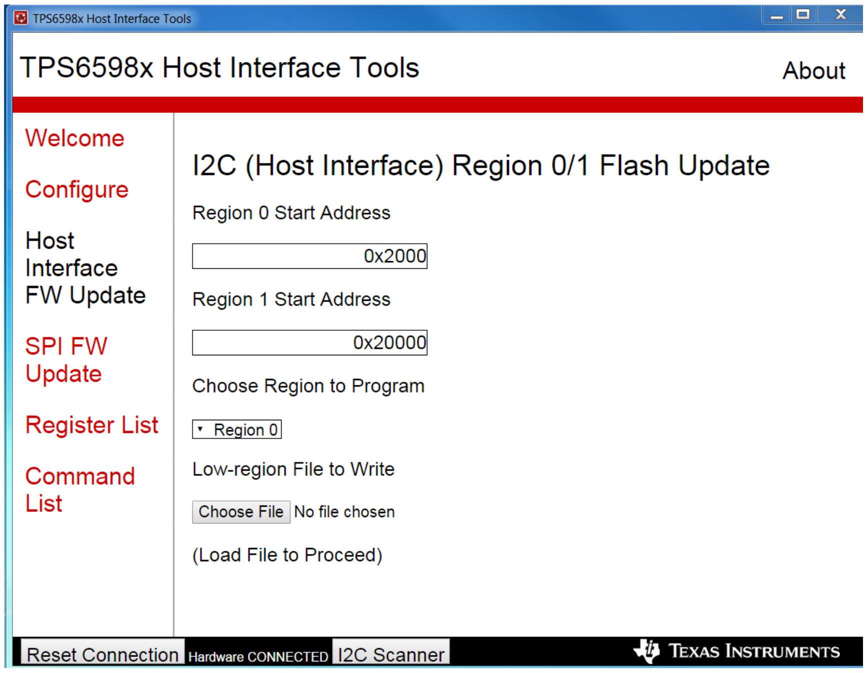
The TPS6598x Utilities GUI provides two mechanisms for performing a firmware (FW) update: the host interface FW update and a SPI FW update. Both updates are accessed by clicking the corresponding button on the page selection pane. The host interface FW update mechanism uses the FLxx 4CC commands through the I^{2}C host interface to reprogram the system SPI flash using the TPS6598x device. The SPI FW update mechanism uses the SPI bus to directly program the SPI flash. Table 4 summarizes which adaptors can be used to perform each type of flash update.

Table 4. Adaptor and Firmware Update Compatibility

Update	Aardvark	FTDI	USB EP
Host interface FW update	√	√	√
SPI FW update	√	√	x

2.3.1 Host Interface FW Update (Aardvark, FTDI, or EP)

To use the host interface update, a valid application image must already be loaded onto the TPS6598x device, and a live I^{2}C connection must be made. Upon selecting the *Host Interface FW Update* page from the left-hand pane, the *I2C (Host Interface) Region 0/1 Flash Update* page appears which displays the offset of the high-region and low-region images, the inputs for selecting which region to overwrite, and a button to upload a file (see Figure 9). A low-region file is a raw application file with a preappended 4K boot header. For more information on boot regions and the low region files, refer to the *TPS65981, TPS65982, and TPS65986 Firmware User's Guide*.



I2C (Host Interface) Region 0/1 Flash Update

Region 0 Start Address

0x2000

Region 1 Start Address

0x20000

Choose Region to Program

Region 0

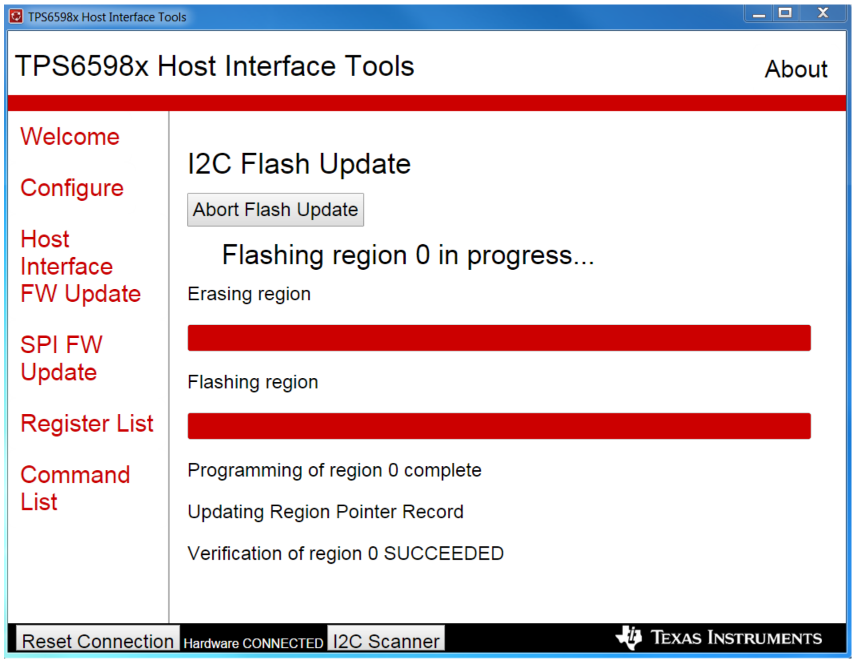
Low-region File to Write

Choose File No file chosen

(Load File to Proceed)

Figure 9. *Host Interface FW Update* **Page**

Click the *Choose File* button to browse and select a low-region binary image (.bin file only), which was created using the TPS6598x Configuration GUI (see *TPS6598x Application-Customization Tool User Guide*). After a valid file is loaded, the *Program the Region* button is available. Click this button to begin the firmware update. During the flash update, the user is unable to navigate away from the page. To move to another page, the user must click the *Abort Flash Update* button or wait for the flash update to conclude. Upon successful completion of the flash update, the page in Figure 10 is displayed in the page display.



I2C Flash Update

Flashing region 0 in progress...

Erasing region

Flashing region

Programming of region 0 complete

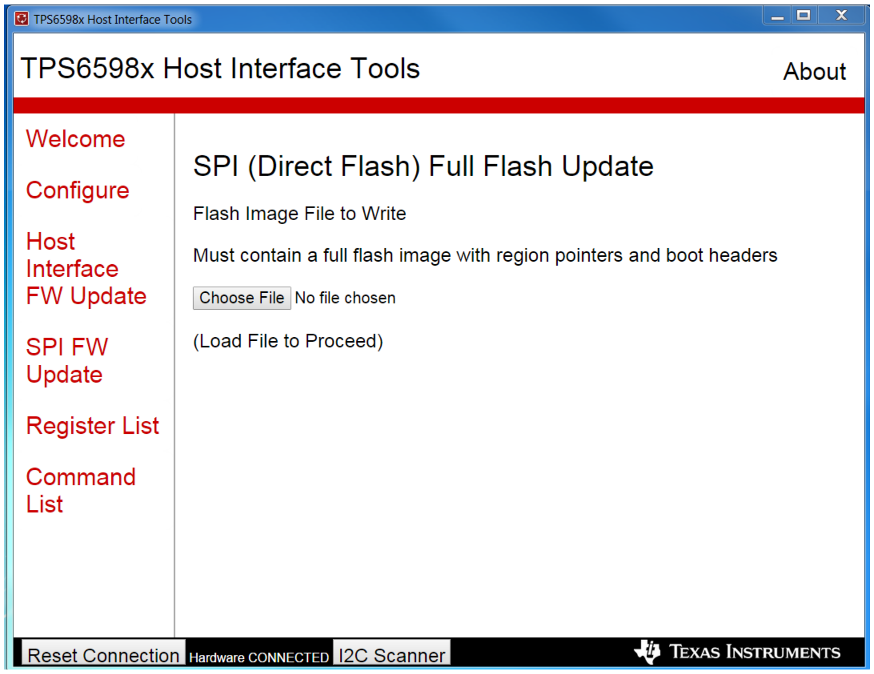
Updating Region Pointer Record

Verification of region 0 SUCCEEDED

Figure 10. Successful Host Interface Flash Update Using USB EP

2.3.2 SPI FW Update (FTDI or Aardvark)

The *SPI FW Update* page allows the user to directly program a SPI flash image using the SPI bus. Upon selecting the *SPI FW Update* page in the left-hand pane, the *SPI (Direct Flash) Full Flash Update* page appears which allows the user to select and program a full flash image (see Figure 11). This image is written onto the SPI flash at offset 0; therefore, it must be a bootable image with high boot regions, low boot regions, and boot headers. For more information on bootable flash images, refer to the *TPS65981, TPS65982, and TPS65986 Firmware User's Guide*.



SPI (Direct Flash) Full Flash Update

Flash Image File to Write

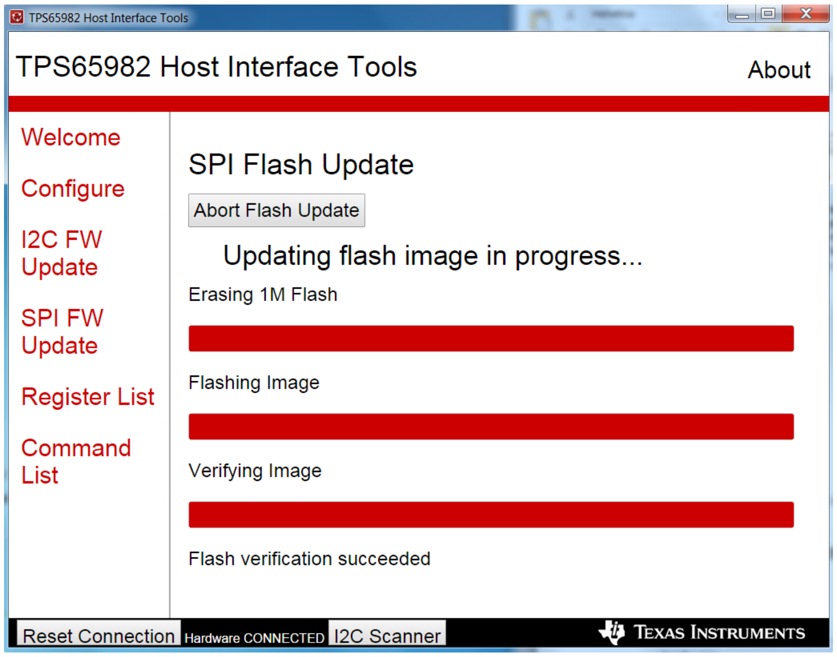
Must contain a full flash image with region pointers and boot headers

No file chosen

(Load File to Proceed)

Figure 11. SPI Flash Update Selection

Click the *Choose File* button to browse and select a full flash image (.bin file only), which was created using the TPS6598x Configuration GUI (see the *TPS6598x Application-Customization Tool User Guide*). After a valid file is loaded, the *Program Flash Image* button is available. Click this button to begin the firmware update. During the flash update, the user is unable to navigate away from the page. To move to another page, the user must click the *Abort Flash Update* button or wait for the flash update to conclude. Upon successful completion of the flash update, the page in Figure 12 is displayed in the page display.



SPI Flash Update

Updating flash image in progress...

Erasing 1M Flash

Flashing Image

Verifying Image

Flash verification succeeded

Figure 12. Successful SPI Flash Update

2.4 Register List

The *Register List* page of the TPS6598x Utilities GUI allows the user to access the virtual registers of a TPS6598x device through an I^{2}C connection to the host interface of the device. Upon selecting the *Register List* page in the left-hand pane, the set of buttons is displayed in the page display. This page includes one button for each available virtual register as shown in Figure 13.

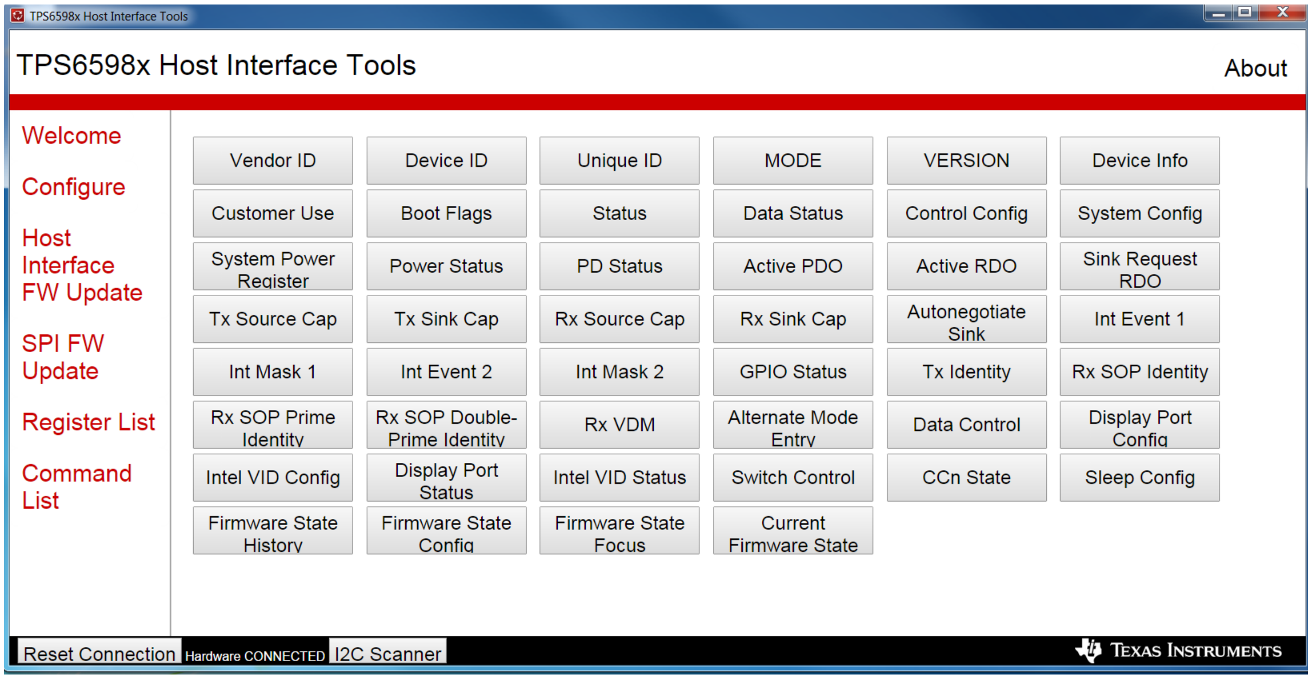
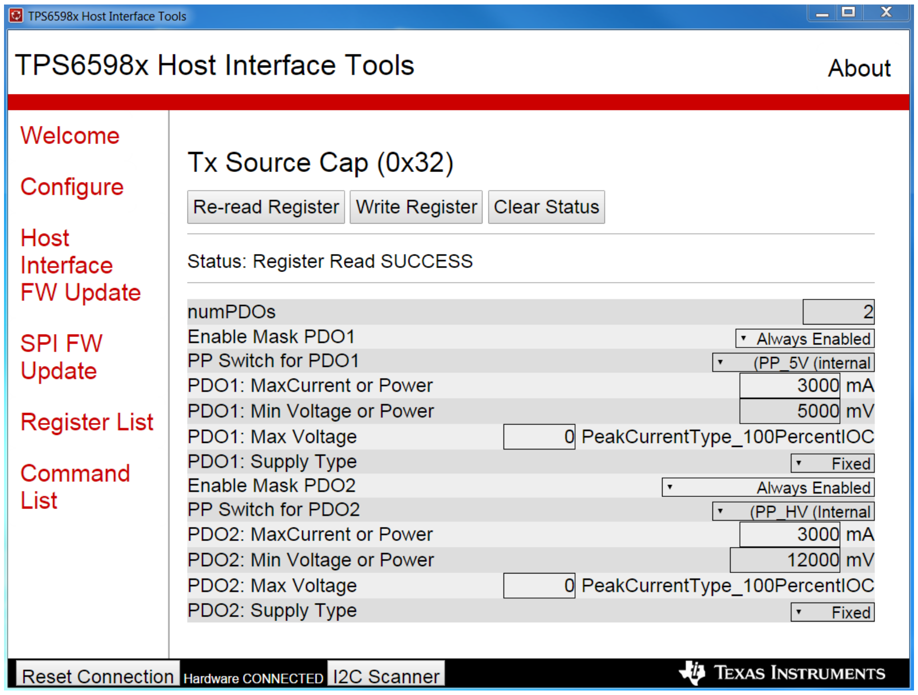


Figure 13. Register Selection List

When the user selects one of the registers from the button list, the TPS6598x Utilities GUI loads a page that displays the name and I^{2}C address of the register. Each field in the register is listed in a readable translation, if applicable. Two types of fields are present: read-only and read-write. Read-write fields can be modified and, when clicked, show a button called *Write Register* which can be executed after a register field is modified using the appropriate user input (check-box, combo box, text box, etc). Read-Only fields are display-only, show a *Re-Read Register* button, and display the most recently read value for each field in a decoded format (true/false, enabled/disabled, decimal conversions, interpreted text, etc).

The selected register (read-write and read-only) is read automatically when pressing the button displaying the register's name. On each register page, a *Re-read Register* button and *Clear Status* button is available. The *Re-read Register* button manually re-reads the register at any time. The *Clear Status* button clears the register read status, which is located below the mentioned buttons.

A *Write Register* button is available for certain registers. The *Write Register* button must be pressed for manual changes to be written to the device.



Tx Source Cap (0x32)

Status: Register Read SUCCESS

numPDOs	2
Enable Mask PDO1	Always Enabled
PP Switch for PDO1	(PP_5V (internal
PDO1: MaxCurrent or Power	3000 mA
PDO1: Min Voltage or Power	5000 mV
PDO1: Max Voltage	0 PeakCurrentType_100PercentIOC
PDO1: Supply Type	Fixed
Enable Mask PDO2	Always Enabled
PP Switch for PDO2	(PP_HV (Internal
PDO2: MaxCurrent or Power	3000 mA
PDO2: Min Voltage or Power	12000 mV
PDO2: Max Voltage	0 PeakCurrentType_100PercentIOC
PDO2: Supply Type	Fixed

Hardware CONNECTED

Figure 14. Register Read-Write Page

NOTE: Writing to the virtual registers at run-time is writing to RAM (volatile memory). Essentially, when the TPS6598x device is disconnected from power, it returns to the original firmware. To permanently change the settings, use the host interface FW update or SPI FW update mechanism to flash (non-volatile memory) the TPS6598x-EVM.

2.5 Command List

The *Command List* page contains a list of 4CC Commands that can be sent from the external controller (Aardvark, for example) to the TPS6598x by using the ASCII code as defined by the *TPS65981, TPS65982, and TPS65986 Host Interface Technical Reference Manual*. Each command allows the user to send these commands and, if applicable, display the return value. Upon selecting the *Command List* page in the left-hand pane, the set of buttons is displayed in the page display, This page includes one button for each available command as shown in Figure 15.

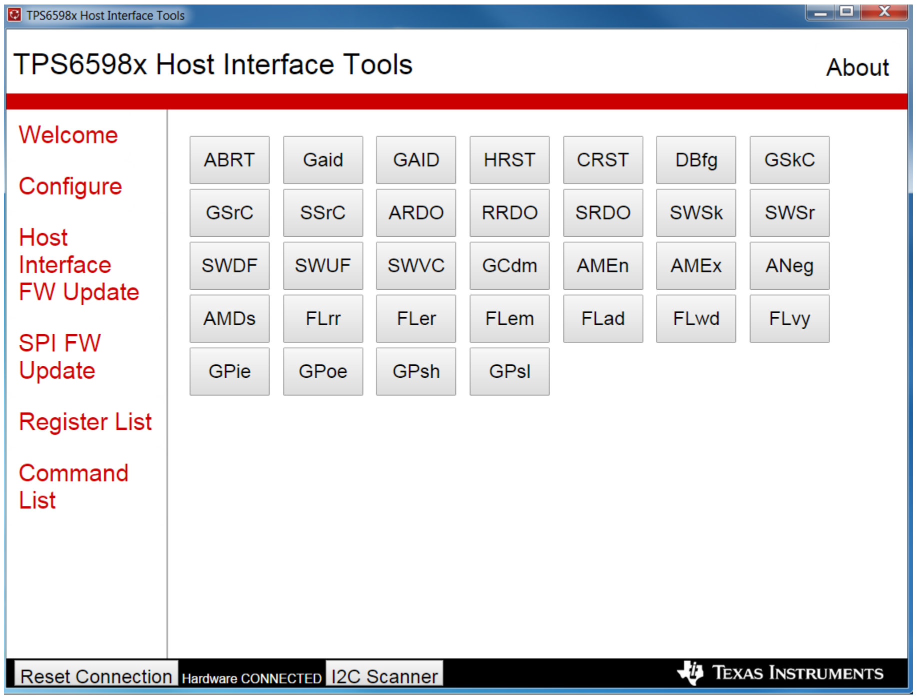
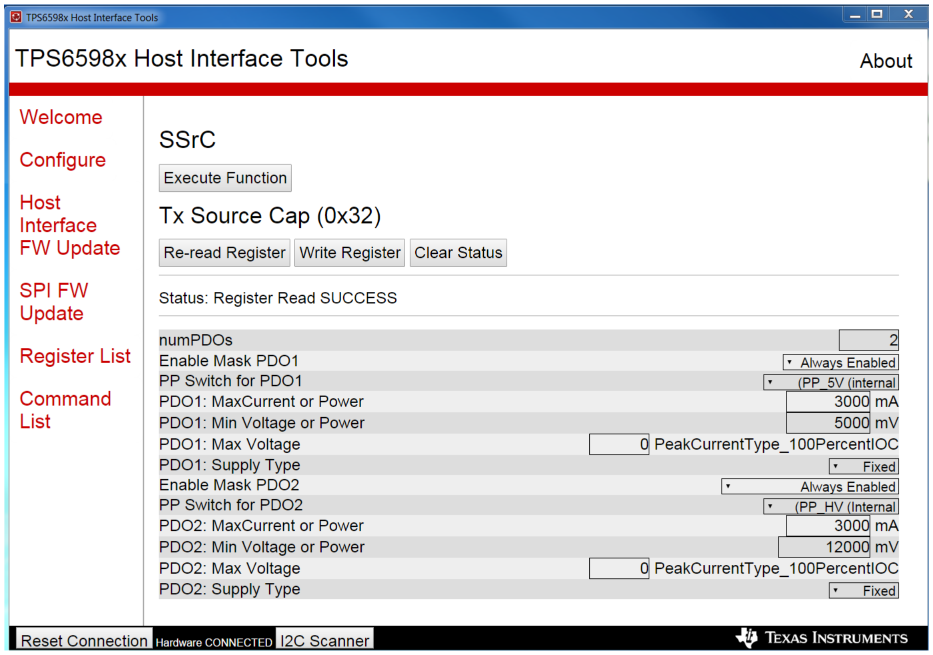


Figure 15. Command List

When a command is selected from the button list, a page is loaded that displays the name of the command and each of the input variables (see Figure 16). Set the input variables and click the *Execute Function* button to send the command.



SSrC

Tx Source Cap (0x32)

Status: Register Read SUCCESS

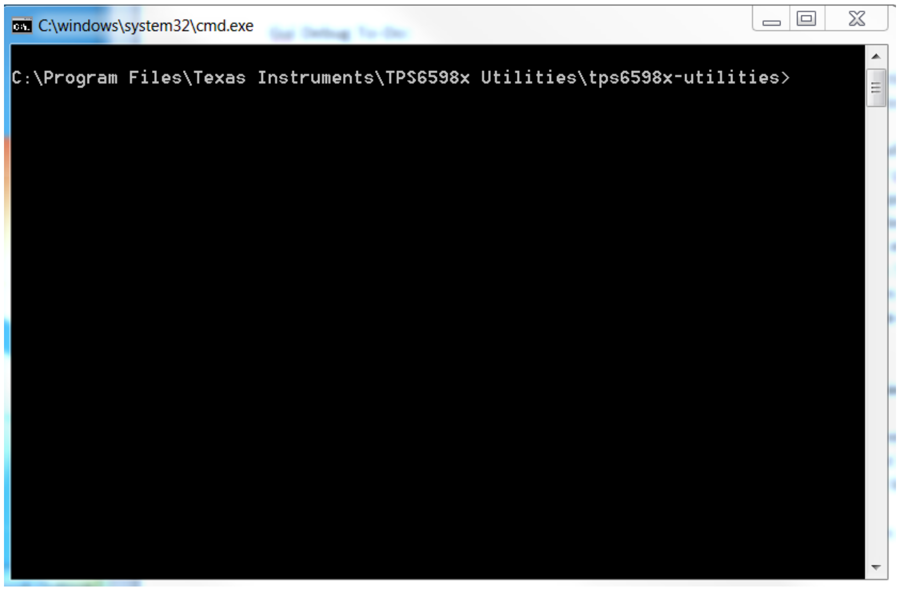
numPDOs	2
Enable Mask PDO1	Always Enabled
PP Switch for PDO1	(PP_5V (internal
PDO1: MaxCurrent or Power	3000 mA
PDO1: Min Voltage or Power	5000 mV
PDO1: Max Voltage	0 PeakCurrentType_100PercentIOC
PDO1: Supply Type	Fixed
Enable Mask PDO2	Always Enabled
PP Switch for PDO2	(PP_HV (Internal
PDO2: MaxCurrent or Power	3000 mA
PDO2: Min Voltage or Power	12000 mV
PDO2: Max Voltage	0 PeakCurrentType_100PercentIOC
PDO2: Supply Type	Fixed

Figure 16. Execute Function Page

3 Using the TPS6598x Utilities Python Scripts

TI provides a collection of Python-based scripts for interacting with the TPS6598x using the host interface accessed through I^{2}C. By opening the TPS6598x Utilities Command Line, one of the text-based Python application scripts is invoked by typing the corresponding filename. The primary advantage to accessing the scripts through the command line is the ability to modify scripts for a custom application. The scripts can also serve as templates for developing host interface access through an external micro-controller on custom hardware.

When opening the TPS6598x Utilities Command Line, a window similar to Figure 17 is displayed. Using this command window as it has been configured to use the relocatable Python distribution that is installed with the bundled installer is important.



```
C:\Program Files\Texas Instruments\TPS6598x Utilities\tps6598x-utilities>
```

Figure 17. Command Window

The bundled installer places the base Python scripts of the TPS6598x utilities in the following directory:

C:\Program Files\Texas Instruments\TPS6598x Utilities\TPS6598x-utilities

Some Python scripts are callable in the TPS6598x Utilities Command Line, while some are intended to be only edited in the Python IDLE program or a third-party Python integrated development environment (IDE). Table 5 lists descriptions of the included Python scripts and other files. Because the Utility Command Line is using a relocatable Python script, the scripts must be called in the following format:

```
> python <Insert Python Script> | more
```

NOTE: The > symbol should not be typed and indicates the command prompt. The | *more* portion of the command is not required, but it is recommended because of the length of most scripts which allows the user to press the *Spacebar* button on the keyboard to forward view the scripts one screen at a time.

Table 5. TPS6598x Host Interface Python Utility Files

File	Description
aardvark.dll	The dynamic-link library used to interact with the Total Phase Aardvark adaptor. See Section 5 for installation instructions.
aardvark_py.py	This file contains low-level Python functions for interacting with the Total Phase Aardvark. It is not meant to be called. See Section 5 for installation instructions.
aardvark_rw.py	This file builds upon aardvark_py.py to provide basic read and write commands using I^2C. It is not meant to be called.
aardvark_spi.py	This file builds upon aardvark_py.py to provide basic read and write commands using SPI. It is not meant to be called.

Table 5. TPS6598x Host Interface Python Utility Files (continued)

File	Description
alternate_modes.py[1]	This file builds upon the 4CC function definitions in hi_functions.py. It reports support for Intel, Display Port, and TI SVIDs and attempts to enter any alternate modes from supported SVIDs in this list.
config.py	This file is used to configure the communication layer to use either FTDI or Aardvark drivers. It is not meant to be called.
debug_trace.py	This file builds upon the register class definitions of register_class.py to define the trace registers of the host interface. It is not meant to be called (use pd_trace.py for an example of using these registers).
device_rw.py	This file provides functions for accessing device registers through the *hw_interface* abstraction layer. It is not meant to be called directly.
flash_update_image_spi.py[1]	This file builds upon the flash programming 4cc function definitions defined in hi_functions.py. It is used to update the SPI flash attached to TPS6598x using the host interface and a full flash binary image.
flash_update_region_0.py[1]	This file builds upon the flash programming 4cc function definitions defined in hi_functions.py. It is used to update region 0 of the SPI flash attached to TPS6598x using the host interface and an application binary.
flash_update_region_1.py[1]	This file builds upon the flash programming 4cc function definitions defined in hi_functions.py. It is used to update region 0 of the SPI flash attached to TPS6598x using the host interface and an application binary.
ftdi.py	This file is the counterpart to aardvark_rw.py and provides functions for accessing the I^{2}C capabilities of FTDI-based boards. It is not meant to be called.
ftdi_spi.py	This file is the counterpart to aardvark_spi.py and works alongside ftdi.py to provide SPI access via FTDI-based boards. It is not meant to be called.
function_class.py	This file defines an object class for the host interface 4CC commands of the TPS6598x devices. This class is used to abstract these functions for presentation in the GUI (see gui.py).
gui.py	This file defines the front-end for the utilities GUI. While it is provided as source code, it is provided as a tool as opposed to an example, and it is not expected that many users will examine or modify this file.
hi_functions.py	This file defines a number of Python functions to call corresponding 4cc host interface functions over the TPS6598x I^{2}C interface. It is not meant to be called.
hw_interface.py	This file helps adapt the scripts to a different hardware interface.
intrusive_PD_example.py[1]	This script provides an example of using the TPS6598x host interface to operate the device in intrusive mode. Intrusive mode disables many of the automatic responses of the TPS6598x device, providing an external micro-controller a larger degree of freedom in the decision making of exposing and accepting power contracts and alternate modes
libMPSSE.dll	This dynamic-link library contains the low-level functions for accessing an FTDI-based debugging board. It is accessed by ftdi.py and ftdi_spi.py.
pd_trace.py[1]	This file builds upon the register definitions of debug_trace.py. It is used to read and display a circular buffer of 64 PD status change messages maintained by the TPS6598x for debugging purposes.
pdst_dump.py	This file uses the TPS6598x host interface to dump a detailed, low-level trace of the internal state machine. Neither the commands used by this script nor the output are documented. Customers should use “pd_trace.py” to achieve similar functionality when debugging, but in some cases, TI support personnel may request the output of this script to aid in debugging systems.
read_registers.py[1]	This file builds upon the register definitions of register_defintions.py. It is used to read and display the basic (non-trace) registers of the TPS6598x host interface.
register_class.py	This file defines an object class for the virtual registers of the TPS6598x host interface and various interaction methods for reading, writing, and displaying the contents of these registers. It is not meant to be called.
register_definitions.py	This file builds upon the register class definitions of register_class.py to define the basic registers of the host interface. It is not meant to be called (see read_registers.py for an example of using these registers).
send_commands.py[1]	This file allows the user to perform various commands including PD swaps, GPIO enables / disable, Alternate Mode, and register read / write.
set_dp_cap.py[1]	This script provides an example for updating the display port capabilities of a system dynamically using the host interface.
set_sleep_level.py	This script provides an example for updating the low power mode (sleep) configuration of a system dynamically using the host interface.
set_sourcesink_cap.py[1]	This script provides an example for updating the power source and sink capabilities of a system dynamically using the host interface.
usbep.py	This file is the counterpart to aardvark_rw.py and ftdi.py and provides functions for accessing the TPS6598x device using the USB2.0 endpoint host interface.
version.py	This file contains the version of the current utilities release.

[1] Callable Scripts

3.1 Configure USB to I^{2}C/SPI Adaptor With *config.py* Script

When initially using the TPS6598x Utilities Tool, check the system configuration as set in the *config.py* file. The contents of this file can be modified using either the *Configure* page of the GUI or the IDLE Python Editor included in the bundled installer at:

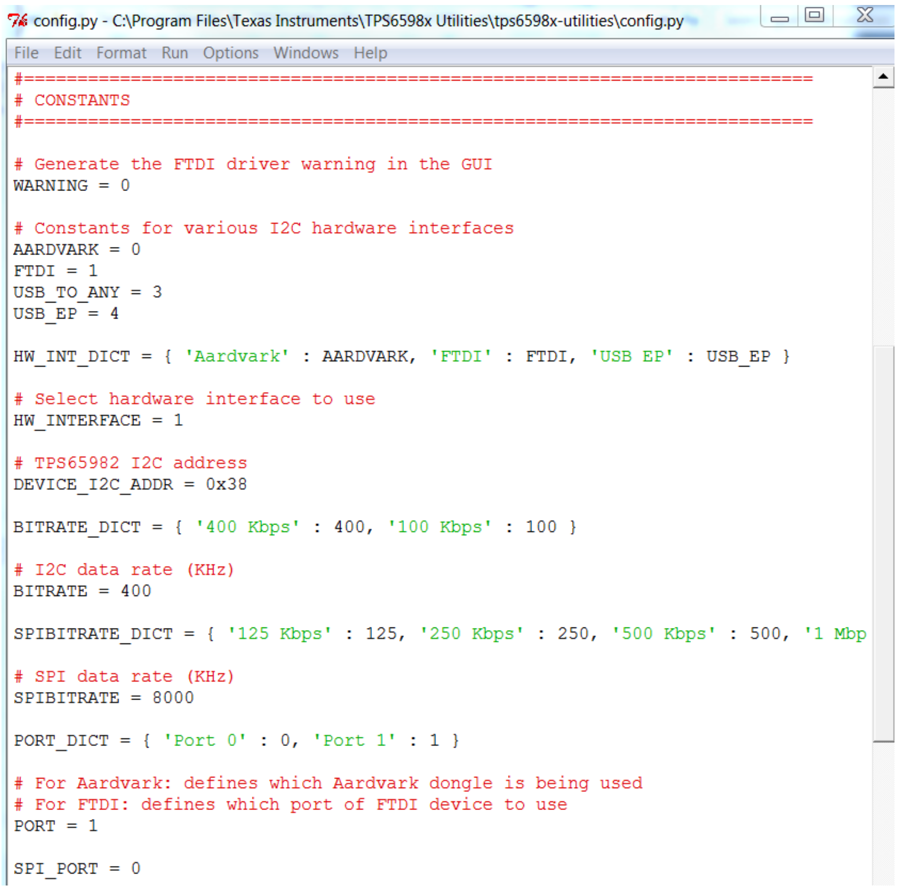
C:\Program Files\Texas Instruments\TPS6598x Utilities\WinPython-64bit-2.7.10.3

The configuration file is easily accessed in IDLE with the following method:

Step 1. Locate the folder containing the Python scripts (bundled installer places scripts in C:\Program Files\Texas Instruments\TPS6598x Utilities\TPS6598x-utilities)

Step 2. Locate the *config.py* file, right-click the file, and select *Edit with IDLE*.

Selecting *Edit with IDLE* opens the IDLE window with the *config.py* contents (see Figure 18). The configuration settings can be directly edited in this window. If saved, the settings will become the new default on the GUI.


config.py - C:\Program Files\Texas Instruments\TPS6598x Utilities\tps6598x-utilities\config.py
File Edit Format Run Options Windows Help

```
#==========================================================================
# CONSTANTS
#==========================================================================

# Generate the FTDI driver warning in the GUI
WARNING = 0

# Constants for various I2C hardware interfaces
AARDVARK = 0
FTDI = 1
USB_TO_ANY = 3
USB_EP = 4

HW_INT_DICT = { 'Aardvark' : AARDVARK, 'FTDI' : FTDI, 'USB EP' : USB_EP }

# Select hardware interface to use
HW_INTERFACE = 1

# TPS65982 I2C address
DEVICE_I2C_ADDR = 0x38

BITRATE_DICT = { '400 Kbps' : 400, '100 Kbps' : 100 }

# I2C data rate (KHz)
BITRATE = 400

SPIBITRATE_DICT = { '125 Kbps' : 125, '250 Kbps' : 250, '500 Kbps' : 500, '1 Mbp

# SPI data rate (KHz)
SPIBITRATE = 8000

PORT_DICT = { 'Port 0' : 0, 'Port 1' : 1 }

# For Aardvark: defines which Aardvark dongle is being used
# For FTDI: defines which port of FTDI device to use
PORT = 1

SPI_PORT = 0
```

Figure 18. Example *config.py* Settings in Python IDLE Program

The I^{2}C address that is specified in the *config.py* file is board dependent. The TPS6598x evaluation modules (EVMs) from TI use the I^{2}C address of 0x38. The I^{2}C Scanner tool in the TPS6598x Utilities GUI scans through all possible addresses and reports successful connections. Details regarding the method used, to determine the I^{2}C address for a given device, are specified in the *TPS65981, TPS65982, and TPS65986 Firmware*.

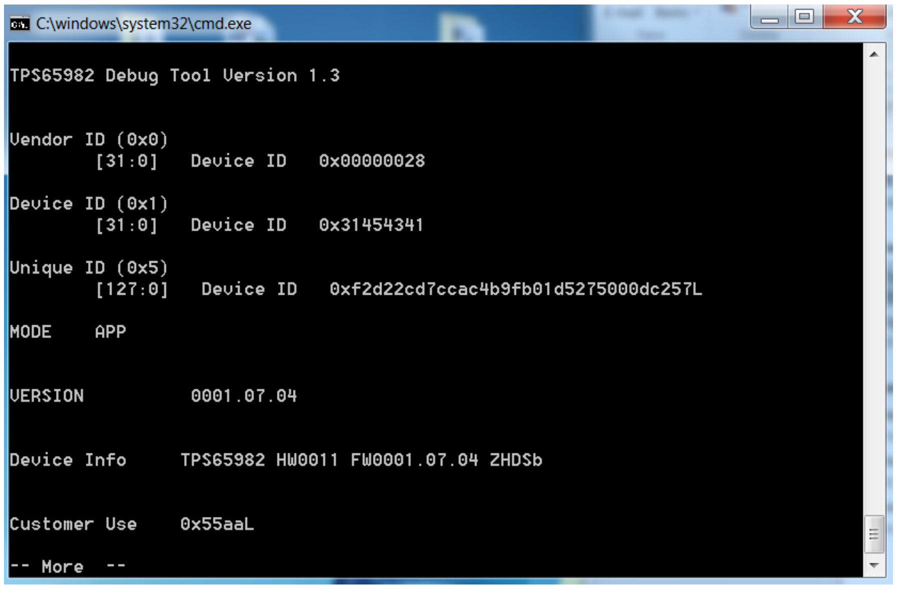
3.2 Execute the *read_registers.py* Script

Invoke callable scripts in the TPS6598x Utilities Command Line using the Python command as follows:

> python read_registers.py | more

Upon execution of the previous command, output similar to that in Figure 19 is displayed.

NOTE: Figure 19 is only a partial output because of the use of | *more* in the command.



```
TPS65982 Debug Tool Version 1.3


Vendor ID (0x0)
        [31:0]   Device ID   0x00000028

Device ID (0x1)
        [31:0]   Device ID   0x31454341

Unique ID (0x5)
        [127:0]   Device ID   0xf2d22cd7ccac4b9fb01d5275000dc257L

MODE    APP


VERSION          0001.07.04


Device Info     TPS65982 HW0011 FW0001.07.04 ZHDSb


Customer Use    0x55aaL

-- More  --
```

Figure 19. Execution of *read_registers.py* Script (Partial)

Users can also consider redirecting the output of the script into a text file by typing the > redirection symbol followed by the file path for the text file. The following is an example of the syntax:

> python read_registers.py > C:\User\Desktop\outfile.txt

Some users will notice that the *read_registers.py* script can be executed from the command line without prepending the Python command. When a file name, such as *read_registers.py*, is typed at the command line, Windows executes the default program handler for the file extension, which may or may not be the Python command parser for the *.py* extension on a given system.

3.3 FW Update With *flash_update_image.py* and *flash_update_region_0.py* Scripts

Three flash programming scripts are available which program the SPI flash attached to the TPS6598x using the TPS6598x host interface. The files are the following:

- flash_update_image_spi.py
- flash_update_region_0.py
- flash_update_region_1.py

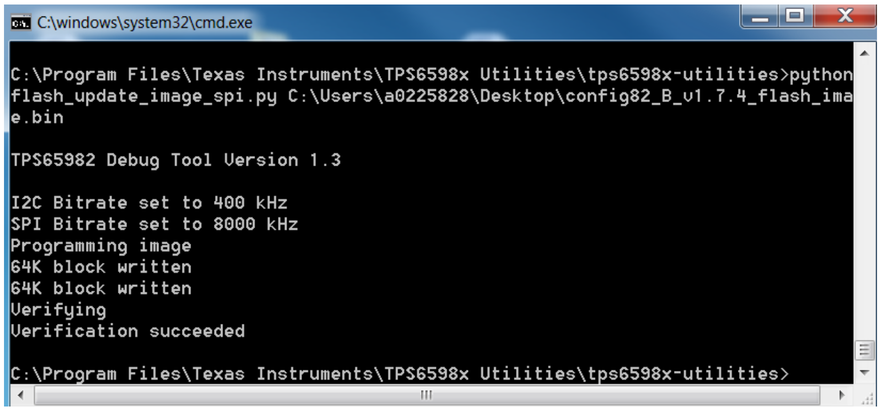
The TPS6598x bootloader specifies two regions in which an application image may reside. These regions provide redundancy in case the flash is corrupted through a field update or another mechanism. The device first attempts to boot from region 0. If the first boot fails, the device attempts to boot from region 1.

The *flash_update_image_spi.py* script reprograms the entire flash using a full-flash image. This image is written into flash offset 0. To use this utility, the flash image programmed must contain region records as well as boot headers for the images in each region. The full-flash script requires a filename argument as shown in the following:

> python flash_update_image_spi.py <Insert Binary File>

For example:

> python flash_update_image_spi.py C:\User\Desktop\full_flash_im.bin



```
C:\Program Files\Texas Instruments\TPS6598x Utilities\tps6598x-utilities>python
flash_update_image_spi.py C:\Users\a0225828\Desktop\config82_B_v1.7.4_flash_ima
e.bin

TPS65982 Debug Tool Version 1.3

I2C Bitrate set to 400 kHz
SPI Bitrate set to 8000 kHz
Programming image
64K block written
64K block written
Verifying
Verification succeeded

C:\Program Files\Texas Instruments\TPS6598x Utilities\tps6598x-utilities>
```

Figure 20. Execution of *flash_update_image.py* Script

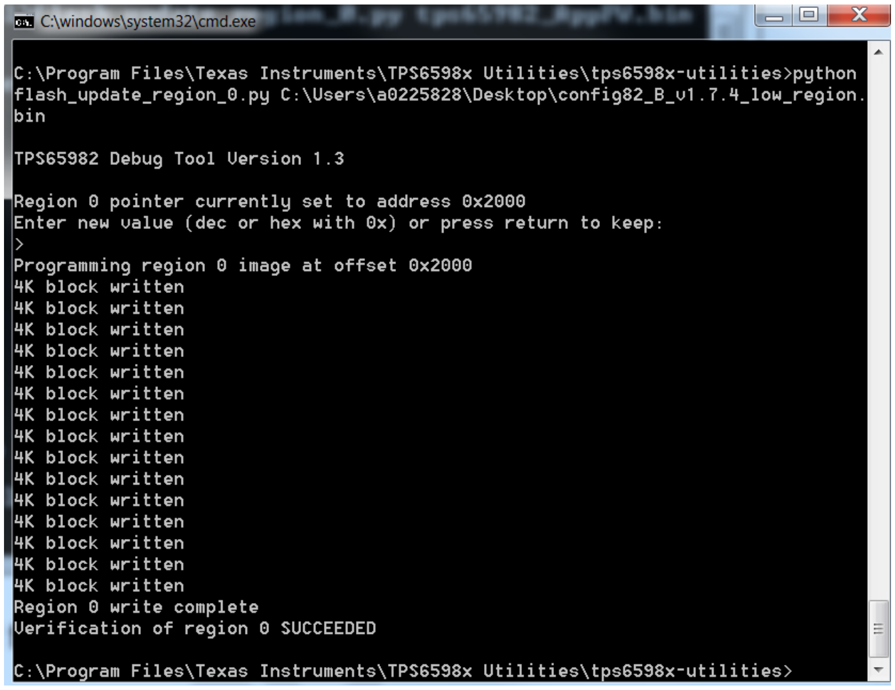
The *flash_update_region_0.py* and *flash_update_region_1.py* scripts update a single region (0 or 1, respectively). Both region-specific scripts expect an application binary which does not contain region records or a boot header. The region-specific scripts require a filename argument as shown in the following:

> python flash_update_region_0.py <Insert Binary File>

For example:

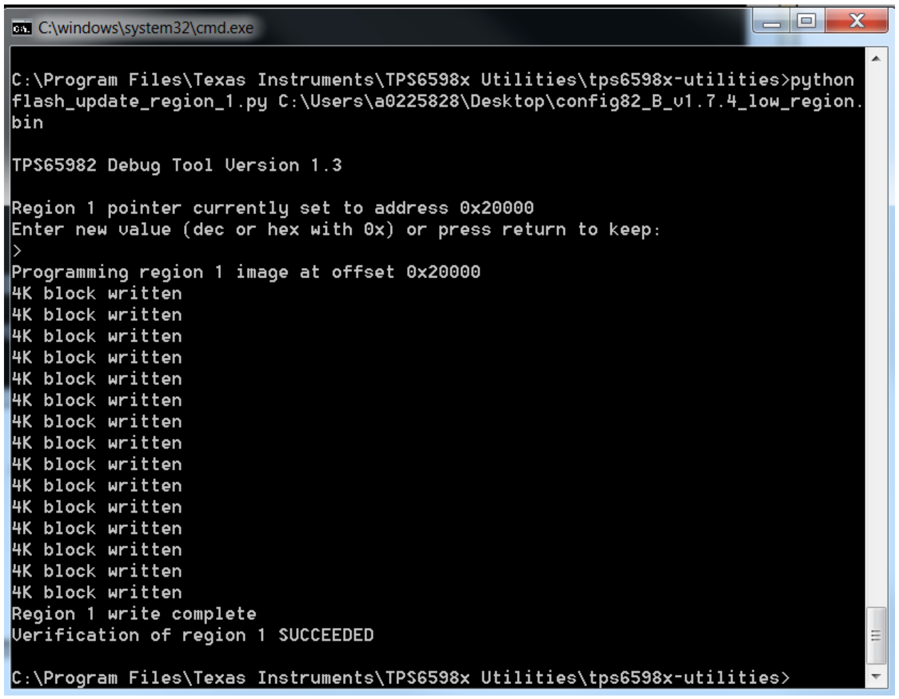
> python flash_update_region_0.py C:\User\Desktop\low_region.bin

Both scripts prompt the user to input a new address to point to or keep the default pointed address. The region 0 and region 1 scripts point to addresses 0x2000 and 0x20000, respectively, by default. See Figure 21 and Figure 22 for successful region-specific flash updates.



```
C:\Program Files\Texas Instruments\TPS6598x Utilities\tps6598x-utilities>python
flash_update_region_0.py C:\Users\a0225828\Desktop\config82_B_v1.7.4_low_region.
bin

TPS65982 Debug Tool Version 1.3

Region 0 pointer currently set to address 0x2000
Enter new value (dec or hex with 0x) or press return to keep:
>
Programming region 0 image at offset 0x2000
4K block written
4K block written
4K block written
4K block written
4K block written
4K block written
4K block written
4K block written
4K block written
4K block written
4K block written
4K block written
4K block written
4K block written
4K block written
Region 0 write complete
Verification of region 0 SUCCEEDED

C:\Program Files\Texas Instruments\TPS6598x Utilities\tps6598x-utilities>
```

Figure 21. Execution of *flash_update_region_0.py* Script



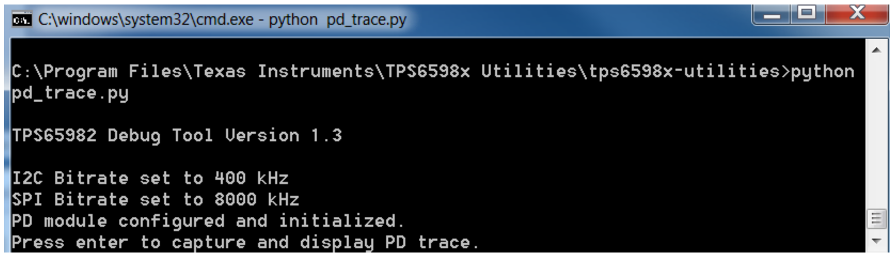
```
C:\Program Files\Texas Instruments\TPS6598x Utilities\tps6598x-utilities>python
flash_update_region_1.py C:\Users\a0225828\Desktop\config82_B_v1.7.4_low_region.
bin

TPS65982 Debug Tool Version 1.3

Region 1 pointer currently set to address 0x20000
Enter new value (dec or hex with 0x) or press return to keep:
>
Programming region 1 image at offset 0x20000
4K block written
4K block written
4K block written
4K block written
4K block written
4K block written
4K block written
4K block written
4K block written
4K block written
4K block written
4K block written
4K block written
4K block written
4K block written
Region 1 write complete
Verification of region 1 SUCCEEDED

C:\Program Files\Texas Instruments\TPS6598x Utilities\tps6598x-utilities>
```

Figure 22. Execution of *flash_update_region_1.py* Script

3.4 Test the *pd_trace.py* Script

The *pd_trace.py* script uses the trace registers defined in the *debug_trace.py* script to read a circular buffer of the PD state transitions that are captured on the TPS6598x device when connected to a device through the Type-C cable.

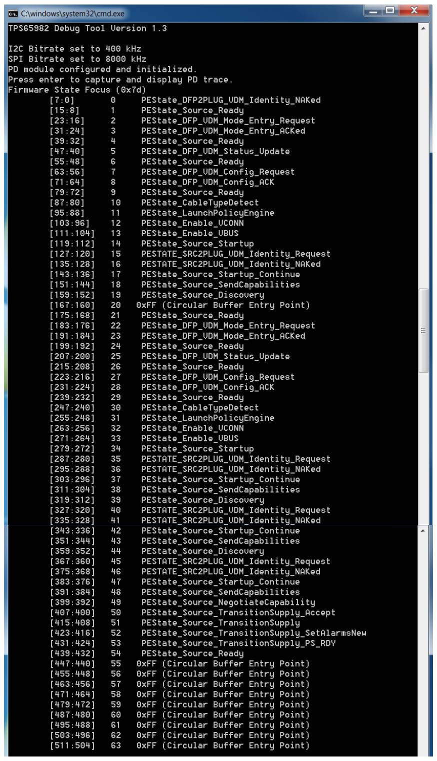
Before the script can read this data, it must configure the trace module and initialize the circular buffer to 0xFF, the circular buffer marker. After the buffer is initialized, the script prompts the user to plug in the Type-C cable to initiate PD negotiation as shown in Figure 23.



```
C:\Program Files\Texas Instruments\TPS6598x Utilities\tps6598x-utilities>python
pd_trace.py

TPS65982 Debug Tool Version 1.3

I2C Bitrate set to 400 kHz
SPI Bitrate set to 8000 kHz
PD module configured and initialized.
Press enter to capture and display PD trace.
```

Figure 23. Partial Execution of pd_trace.py Script

The script is paused at this point (see Figure 23). Before continuing, attach the Type-C cable (with the other device attached) to begin PD negotiation. When negotiation is complete (1 s or 2 s), press the *Enter* button on the keyboard to continue. The PD state transitions populate in the command window as shown in Figure 24.



```
TPS65982 Debug Tool Version 1.3

I2C Bitrate set to 400 kHz
SPI Bitrate set to 8000 kHz
PD module configured and initialized.
Press enter to capture and display PD trace.
Firmware State Focus (0x7d)
        [7:0]       0     PEState_DFP2PLUG_VDM_Identity_NAKed
        [15:8]      1     PEState_Source_Ready
        [23:16]     2     PEState_DFP_VDM_Mode_Entry_Request
        [31:24]     3     PEState_DFP_VDM_Mode_Entry_ACKed
        [39:32]     4     PEState_Source_Ready
        [47:40]     5     PEState_DFP_VDM_Status_Update
        [55:48]     6     PEState_Source_Ready
        [63:56]     7     PEState_DFP_VDM_Config_Request
        [71:64]     8     PEState_DFP_VDM_Config_ACK
        [79:72]     9     PEState_Source_Ready
        [87:80]     10    PEState_CableTypeDetect
        [95:88]     11    PEState_LaunchPolicyEngine
        [103:96]    12    PEState_Enable_VCONN
        [111:104]   13    PEState_Enable_VBUS
        [119:112]   14    PEState_Source_Startup
        [127:120]   15    PESTATE_SRC2PLUG_VDM_Identity_Request
        [135:128]   16    PESTATE_SRC2PLUG_VDM_Identity_NAKed
        [143:136]   17    PEState_Source_Startup_Continue
        [151:144]   18    PEState_Source_SendCapabilities
        [159:152]   19    PEState_Source_Discovery
        [167:160]   20   0xFF (Circular Buffer Entry Point)
        [175:168]   21    PEState_Source_Ready
        [183:176]   22    PEState_DFP_VDM_Mode_Entry_Request
        [191:184]   23    PEState_DFP_VDM_Mode_Entry_ACKed
        [199:192]   24    PEState_Source_Ready
        [207:200]   25    PEState_DFP_VDM_Status_Update
        [215:208]   26    PEState_Source_Ready
        [223:216]   27    PEState_DFP_VDM_Config_Request
        [231:224]   28    PEState_DFP_VDM_Config_ACK
        [239:232]   29    PEState_Source_Ready
        [247:240]   30    PEState_CableTypeDetect
        [255:248]   31    PEState_LaunchPolicyEngine
        [263:256]   32    PEState_Enable_VCONN
        [271:264]   33    PEState_Enable_VBUS
        [279:272]   34    PEState_Source_Startup
        [287:280]   35    PESTATE_SRC2PLUG_VDM_Identity_Request
        [295:288]   36    PESTATE_SRC2PLUG_VDM_Identity_NAKed
        [303:296]   37    PEState_Source_Startup_Continue
        [311:304]   38    PEState_Source_SendCapabilities
        [319:312]   39    PEState_Source_Discovery
        [327:320]   40    PESTATE_SRC2PLUG_VDM_Identity_Request
        [335:328]   41    PESTATE_SRC2PLUG_VDM_Identity_NAKed
        [343:336]   42    PEState_Source_Startup_Continue
        [351:344]   43    PEState_Source_SendCapabilities
        [359:352]   44    PEState_Source_Discovery
        [367:360]   45    PESTATE_SRC2PLUG_VDM_Identity_Request
        [375:368]   46    PESTATE_SRC2PLUG_VDM_Identity_NAKed
        [383:376]   47    PEState_Source_Startup_Continue
        [391:384]   48    PEState_Source_SendCapabilities
        [399:392]   49    PEState_Source_NegotiateCapability
        [407:400]   50    PEState_Source_TransitionSupply_Accept
        [415:408]   51    PEState_Source_TransitionSupply
        [423:416]   52    PEState_Source_TransitionSupply_SetAlarmsNew
        [431:424]   53    PEState_Source_TransitionSupply_PS_RDY
        [439:432]   54    PEState_Source_Ready
        [447:440]   55   0xFF (Circular Buffer Entry Point)
        [455:448]   56   0xFF (Circular Buffer Entry Point)
        [463:456]   57   0xFF (Circular Buffer Entry Point)
        [471:464]   58   0xFF (Circular Buffer Entry Point)
        [479:472]   59   0xFF (Circular Buffer Entry Point)
        [487:480]   60   0xFF (Circular Buffer Entry Point)
        [495:488]   61   0xFF (Circular Buffer Entry Point)
        [503:496]   62   0xFF (Circular Buffer Entry Point)
        [511:504]   63   0xFF (Circular Buffer Entry Point)
```

Figure 24. Completed Execution of pd_trace.py Script

4 Installation of Python

As discussed in Section 1.3, Python and the required components can be downloaded separately. Go to the python.org 2.7 Release Download Page to download the Python .msi file. Make sure to select the correct download file for the correct PC operating system. Total Phase, the maker of the Aardvark adaptor, recommends using **Python version 2.7**.

NOTE: Python version 2.7 is not the latest version of Python.

This procedure is tested for a 64-bit x86 platform. The "*Windows X86-64 MSI Installer (2.7)*" option was selected to download the*Python-2.7.amd64.msi* file.

Follow the prompts of the .msi file installer to instal Python.

Use the following steps to add the Python executable to the Windows path variable:

Step 1. From the *Windows Start Menu*, select the *Control Panel*.

Step 2. From the *Control Panel* window, click the *System* icon (see Figure 25).

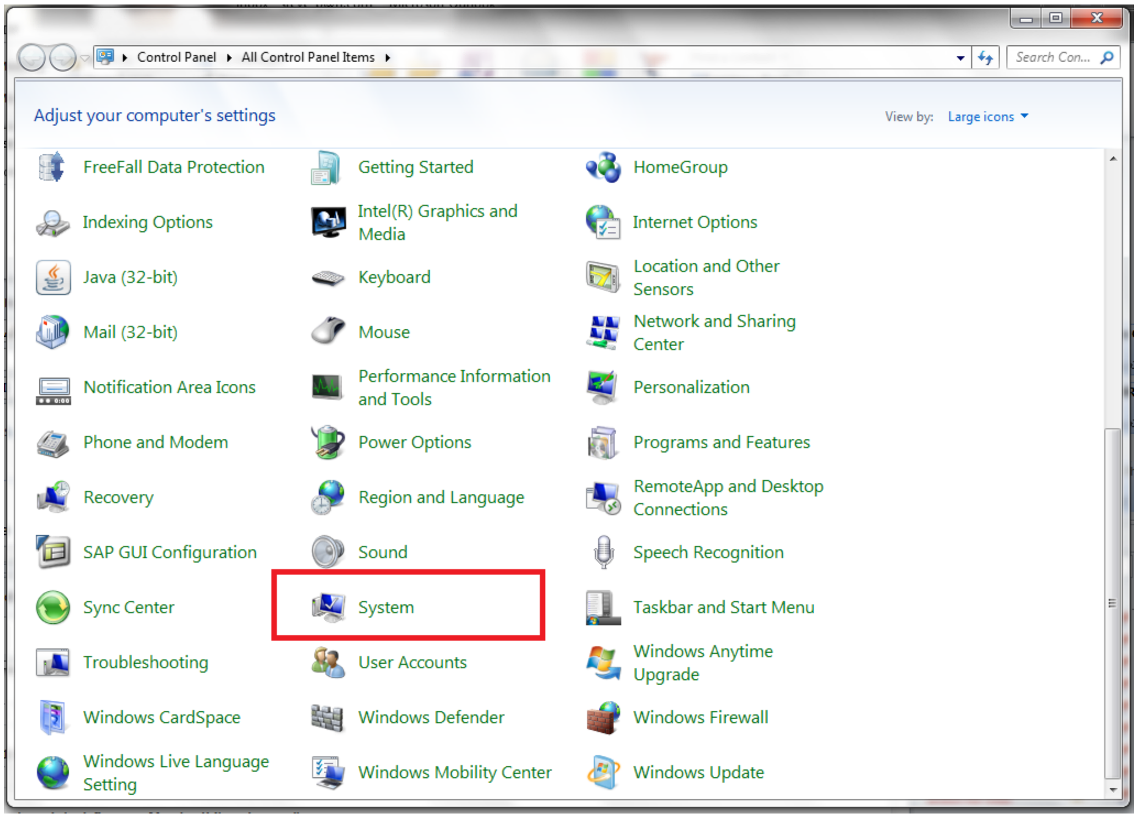
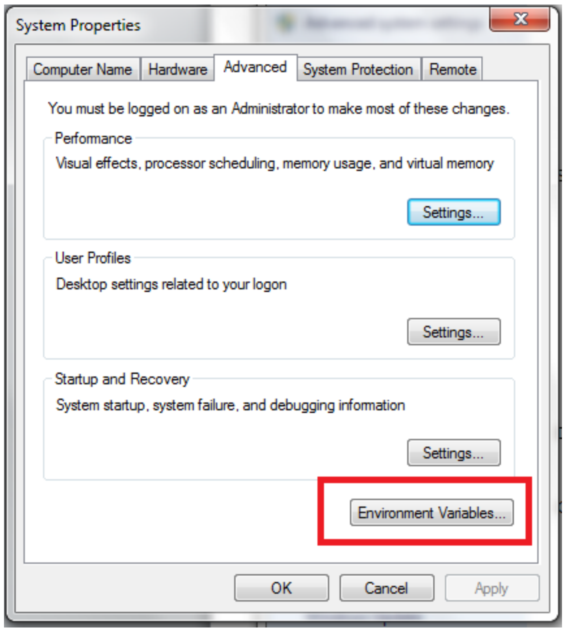


Figure 25. Windows® Control Panel Window

Step 3. From the *System* window, select the *Advanced system settings* option from the left-hand navigation pane.

Step 4. In the *System Properties* window, click the *Environment Variables* button on the *Advanced* tab (see Figure 26).


System Properties

Computer Name | Hardware | Advanced | System Protection | Remote

You must be logged on as an Administrator to make most of these changes.

Performance
Visual effects, processor scheduling, memory usage, and virtual memory
Settings...

User Profiles
Desktop settings related to your logon
Settings...

Startup and Recovery
System startup, system failure, and debugging information
Settings...

Environment Variables...

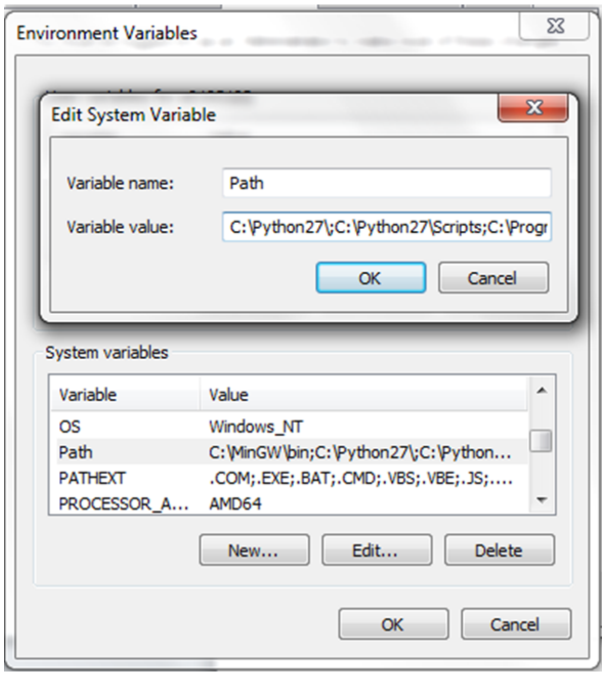
OK Cancel Apply

Figure 26. System Properties Window

Step 5. In the *Environment Variables* window, select *Path* in the *System variables* section.

Step 6. Click the *Edit...* button.

Step 7. Add the following in the *Variable value:* field: *C:\Python27\;C:\Python27\scripts* (see Figure 27).



Variable name: Path

Variable value: C:\Python27\;C:\Python27\Scripts;C:\Progr

Variable	Value
OS	Windows_NT
Path	C:\MinGW\bin;C:\Python27\;C:\Python...
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;....
PROCESSOR_A...	AMD64

Figure 27. Editing the Path

Step 8. Click the *OK* button in the *Environment Variables* window to save the changes.

Step 9. Click the *Apply* button in the *System Properties* window and then click the *OK* button.

5 Using the Aardvark

5.1 *Software and Driver Installation*

As discussed in Section 1.3, the Aardvark driver and associated software are not included in the bundled installer. Use the following steps to install the software and driver:

Step 1. Go to the *Total Phase USB Drivers* website (www.totalphase.com/products/usb-drivers-windows) to download the Aardvark driver.

Step 2. Click the *USB Drivers – Windows v2.12* link and fill out the requested registration information if an account has not been created.

Step 3. Install the Aardvark drivers using the .exe which is downloaded from the Total Phase website by following these steps:

(a) Extract the .zip file to obtain the Total PhaseUSB-v2.12.exe file.

NOTE: The file must execute the installer with administrative privileges.

(b) In Windows 7, right click the .exe file and select *Run as Administrator* from the menu.

(c) Follow the instructions in the installer window to complete the installation.

Step 4. Go to the *Total Phase Aardvark Software* website (www.totalphase.com/products/aardvark-software-api).

Step 5. Click the *Aardvark Software API v5.15 (Windows 64-bit)* link to download the aardvark-api-windows-x86_64-v5.15.zip file.

Step 6. De-archive the Aardvark Software API to open the *aardvark-api-windows-x86_64-v5.15* folder.

Step 7. Search in this folder for the *Python* folder. Place the *aardvark.dll* and *aardvark_py.py* files into the *Configuration Tool Python* scripts folder (C:\Program Files\Texas Instruments\TPS6598x Utilities\TPS6598x-utilities).

NOTE: The rest of the downloaded files are not required for the Aardvark to function.

5.2 *Connecting Aardvark to TPS6598x-EVM*

5.2.1 Direct Connection

The Aardvark connector has an I^{2}C line that must be physically connected to the I^{2}C1 of the TPS6598x device. Refer to the Aardvark User’s Guide (www.totalphase.com/support/articles/200468316) for the pin connections and descriptions of the Aardvark connector.

Figure 28 is from the design files of the TPS6598x-EVM and shows the pin-out of the J2 connector for I^{2}C (pins 16, 18, 20) and the J3 connector for SPI (not discussed in this document) and an alternate ground (GND) pin. Figure 28 shows the position of the three I^{2}C lines required on the Total Phase Aardvark adaptor: serial clock (SCL), serial data (SDA), and ground (GND). For more details and the full schematic of the TPS65982-EVM, refer to the TPS65982-EVM User's Guide.

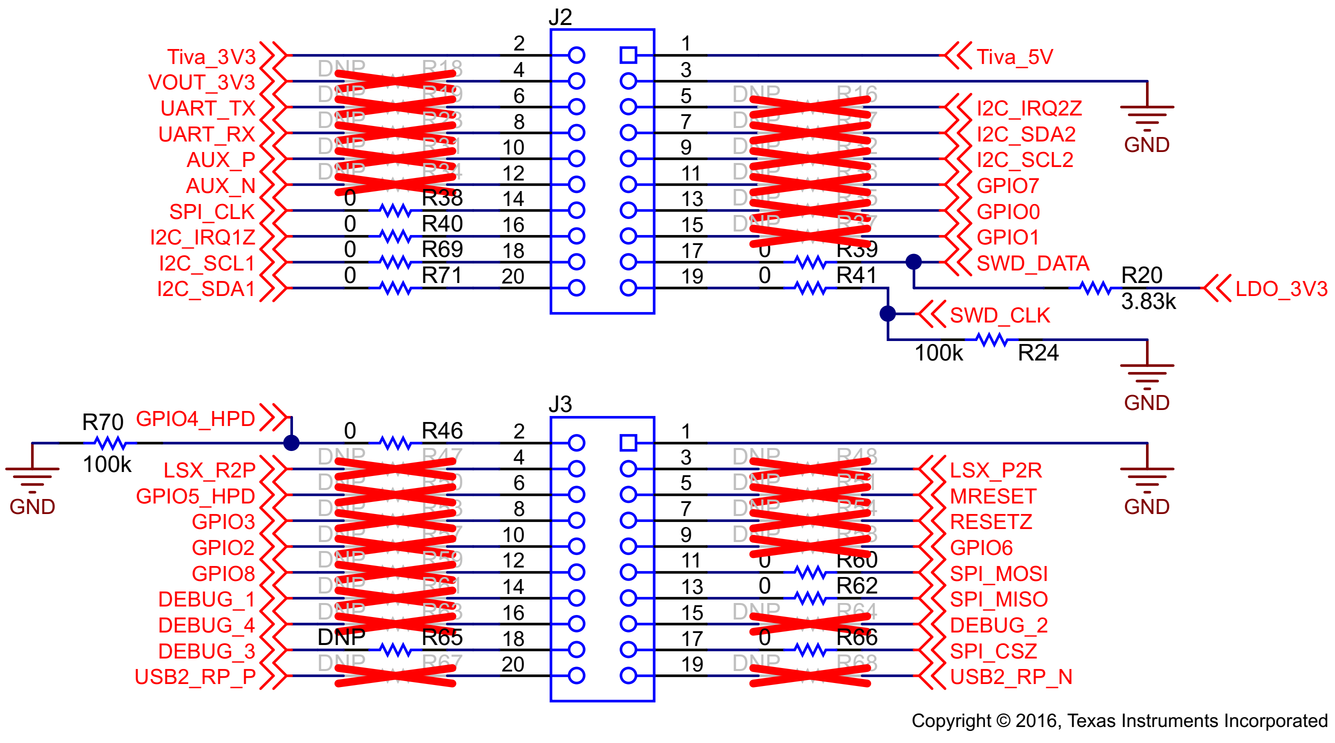
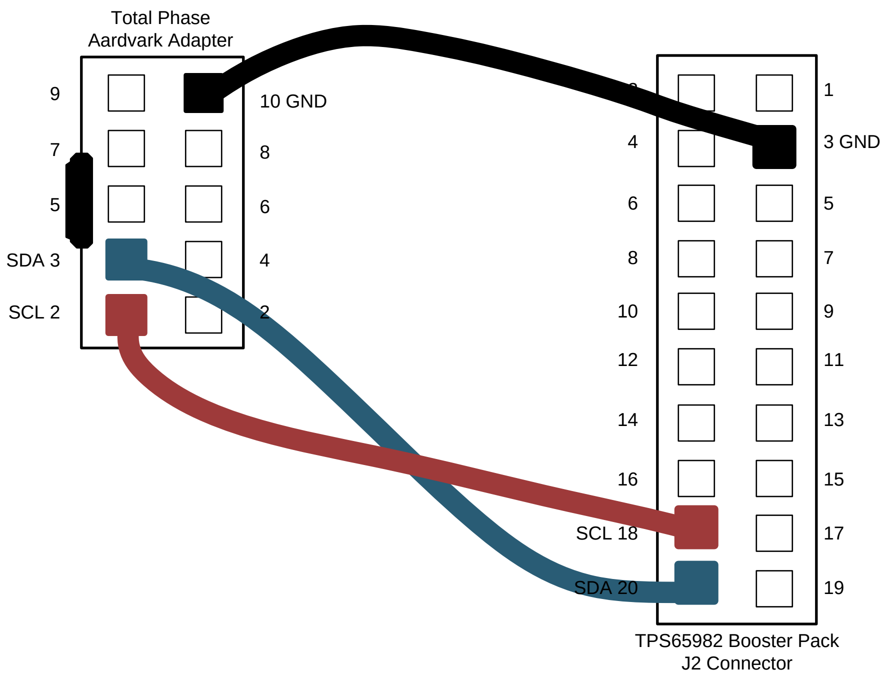


Figure 28. Texas Instruments TPS6598x Booster Pack J2 and J3 Pin Connections for I²C and SPI

Figure 29 shows the wiring diagram to wire the I²C lines of the Aardvark connector into the J2 connector of the TPS6598x-EVM. The wiring diagram represents the connections for using the I²C port 1; however, the host interface also recognizes commands sent over I²C port 2.



the position of the pins is shown as looking down at the board from above.

Figure 29. Aardvark to Booster Pack Wiring Diagram

Figure 30 shows an Aardvark adaptor wired into the TPS6598x Booster Pack. Notice that the orientation of the TPS6598x board and the Aardvark connector match the diagram in Figure 29.

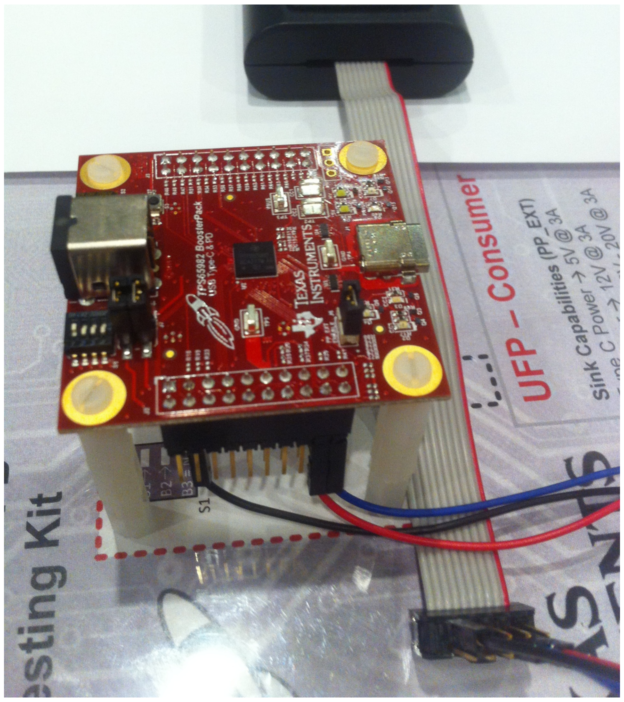


Figure 30. Correctly-Wired System

5.2.2 Aardvark Using LaunchPad EVM to Aardvark Adaptor PCB

The Aardvark can also be connected the standard 10-pin male header connection on a LaunchPad EVM to Aardvark adaptor PCB. See Figure 31 for the location of the connector.

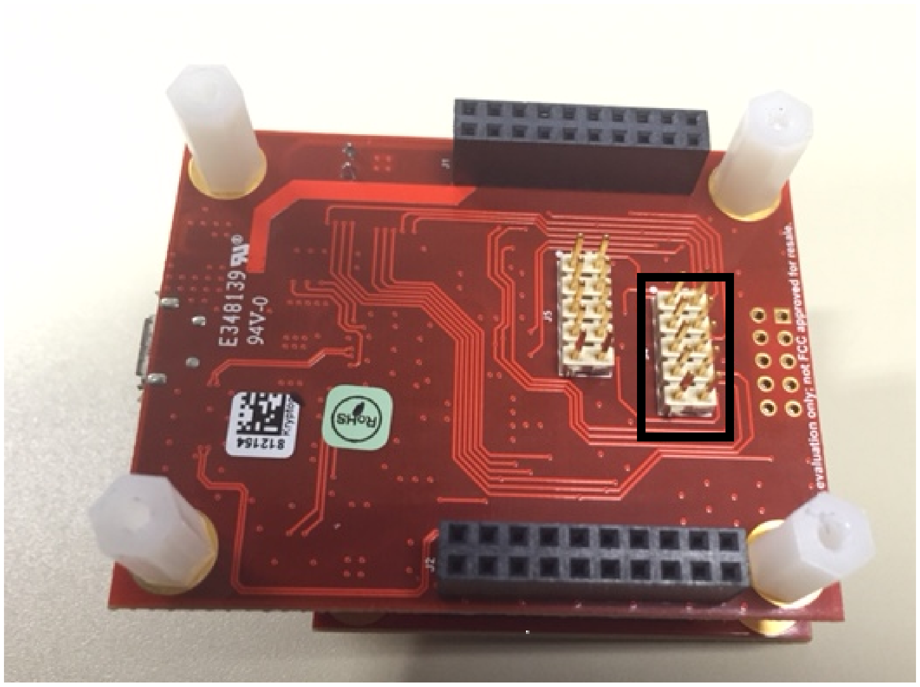


Figure 31. Aardvark 10-pin Header on LaunchPad EVM to Aardvark Adaptor PCB

This feature offers the ability to directly connect the Aardvark connector, as shown in Figure 32.

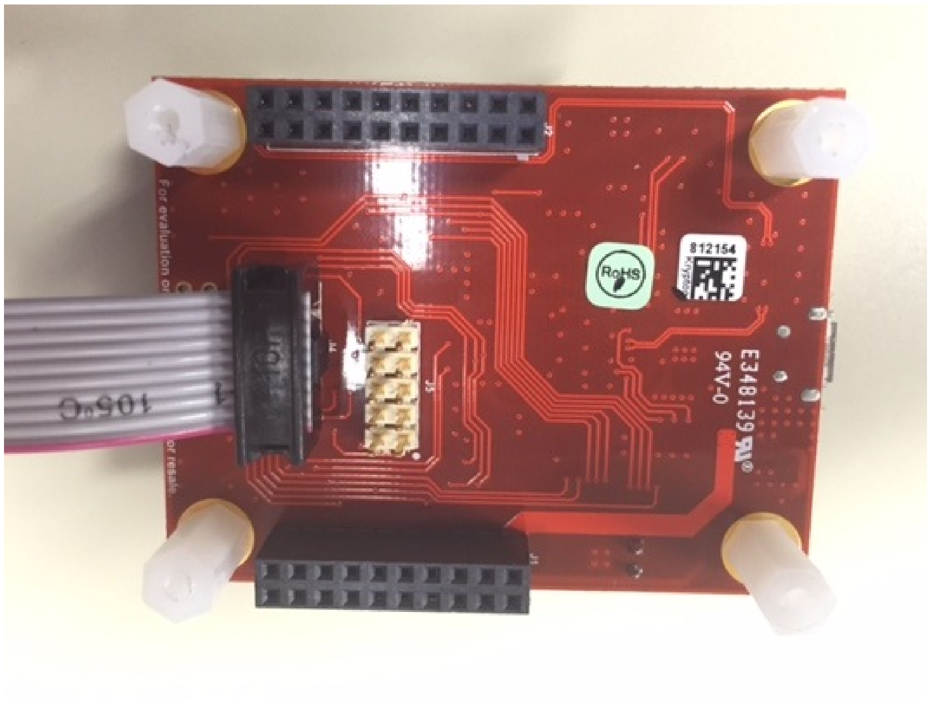


Figure 32. Aardvark 10-pin Header Attached to LaunchPad EVM to Aardvark Adaptor PCB

Revision History

NOTE: Page numbers for previous revisions may differ from page numbers in the current version.

Changes from B Revision (July 2016) to C Revision	Page
• Updated the titles of supporting documentation	3
• Updated the installation instructions for the Aardvark driver	26

Products		**Applications**	
Audio	www.ti.com/audio	Automotive and Transportation	www.ti.com/automotive
Amplifiers	amplifier.ti.com	Communications and Telecom	www.ti.com/communications
Data Converters	dataconverter.ti.com	Computers and Peripherals	www.ti.com/computers
DLP® Products	www.dlp.com	Consumer Electronics	www.ti.com/consumer-apps
DSP	dsp.ti.com	Energy and Lighting	www.ti.com/energy
Clocks and Timers	www.ti.com/clocks	Industrial	www.ti.com/industrial
Interface	interface.ti.com	Medical	www.ti.com/medical
Logic	logic.ti.com	Security	www.ti.com/security
Power Mgmt	power.ti.com	Space, Avionics and Defense	www.ti.com/space-avionics-defense
Microcontrollers	microcontroller.ti.com	Video and Imaging	www.ti.com/video
RFID	www.ti-rfid.com		
OMAP Applications Processors	www.ti.com/omap	**TI E2E Community**	e2e.ti.com
Wireless Connectivity	www.ti.com/wirelessconnectivity		