PMBus Slave Example Project
1.00

Features

- Demonstrates basic usage of the PMBus Slave component in a power supervisor application

General Description

This example project demonstrates how to get started using the PMBus Slave component in a simulated power supervisor application. In place of external DC-DC power converters, voltage DACs (VDACs) internal to PSoC are used. These are combined with the Power Monitor and Voltage Sequencer components to create a simple 4-rail power supervisor that can run on a PSoC development board without any special hardware. The example supports a small subset of the PMBus command set to configure and monitor the simulated power supervisor.

Development Kit Configuration

The following configuration instructions provide a guideline to test this design. For simplicity, the instructions describe the stepwise process to be followed when testing this design with the PSoC Development Kit (CY8CKIT-001) board, but can be generalized for the PSoC 3 Development Kit (CY8CKIT-030) and PSoC 5 Development Kit (CY8CKIT-050) as well.

1. Connect the PMBus host SDA signal to P0[0] and SCL signal to P0[1]. Any compliant PMBus host may be used. The PSoC Programmer hardware (MiniProg3) combined with the Cypress Bridge Control Panel software application can also be used to emulate a PMBus host
2. For demonstration purposes, the Voltage Sequencer enable signals EN[1:4] that would connect to external DC-DC power converters in a real power supervisor design are made available on port pins P0[2:5]. These pins may optionally be connected to a Logic Analyzer or Oscilloscope
3. Connect an external 12V power supply to the Development Kit power connector

Project Configuration

The PSoC Creator design schematic is shown in **Figure 1** below. Note that most of the schematic in this example is occupied by the simulated power supervisor application. The PMBus slave component only requires two bi-directional pin connections for I^2C clock (SCL) and data (SDA).

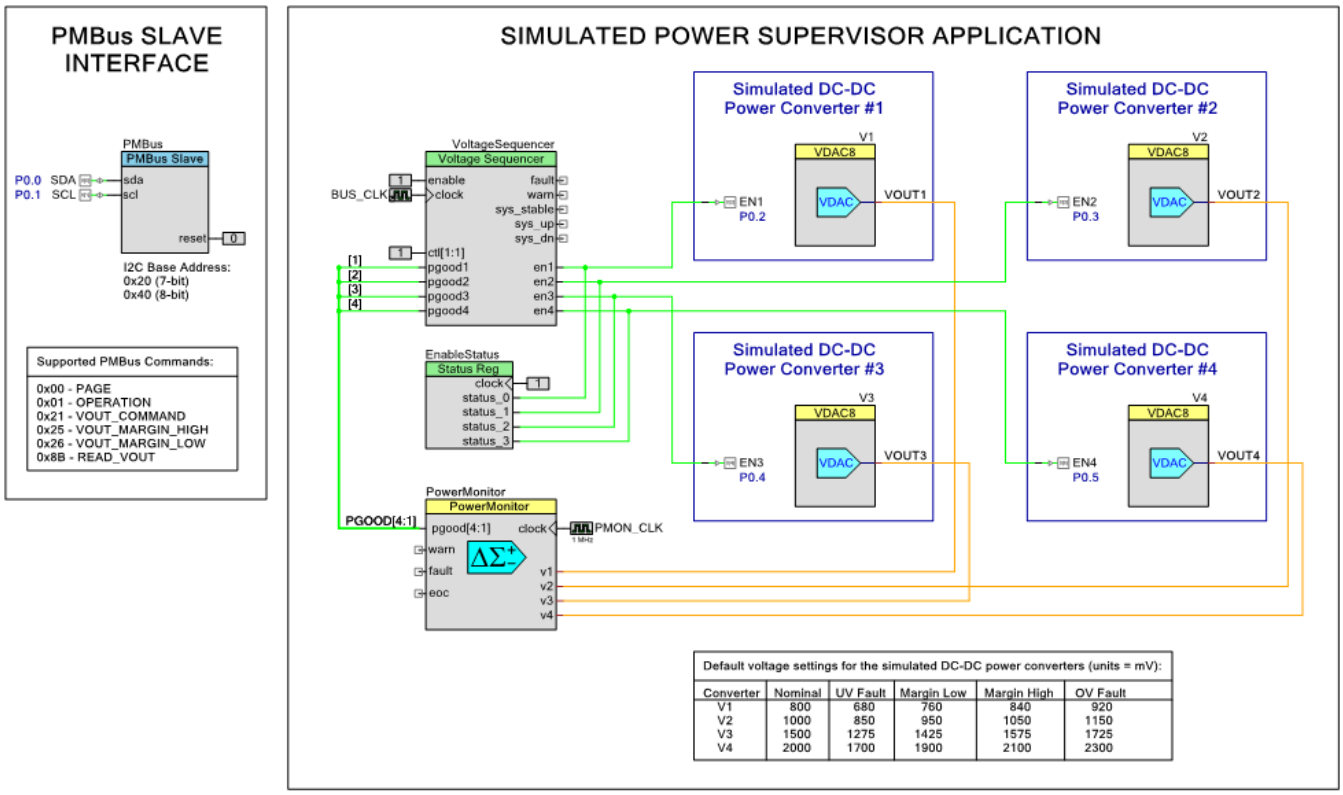

Figure 1 – PMBus Slave Component Example Project Schematic

The simulated power supervisor application is comprised of four VDACs to generate voltages, the Power Monitor Component for voltage measurement and fault detection, and the Voltage Sequencer component to handle power converter power-up sequence and timing. Rather than connecting external power converters, the sequencer outputs are connected to a Status Register Component. Firmware monitors this register and turns the VDACs on/off as instructed by the Voltage Sequencer. Additional example projects exist for the Power Monitor and Voltage Sequencer components.

Open the Configure Dialog for the PMBus Slave by double clicking on the component. Figure 2 shows the General tab that enables users to set the desired data rate, I^2C address, and optional PMBus features. In this example, the data rate is set to 400 KHz and the I^2C address (7-bit) is 0x20. The number of PMBus pages is also set here. Pages allow access to multiple logical units within a single physical PMBus device. In this example, there are 4 simulated power converter rails and each one is accessed through a separate page. Therefore, the **Paged commands array size** parameter is set to 4.

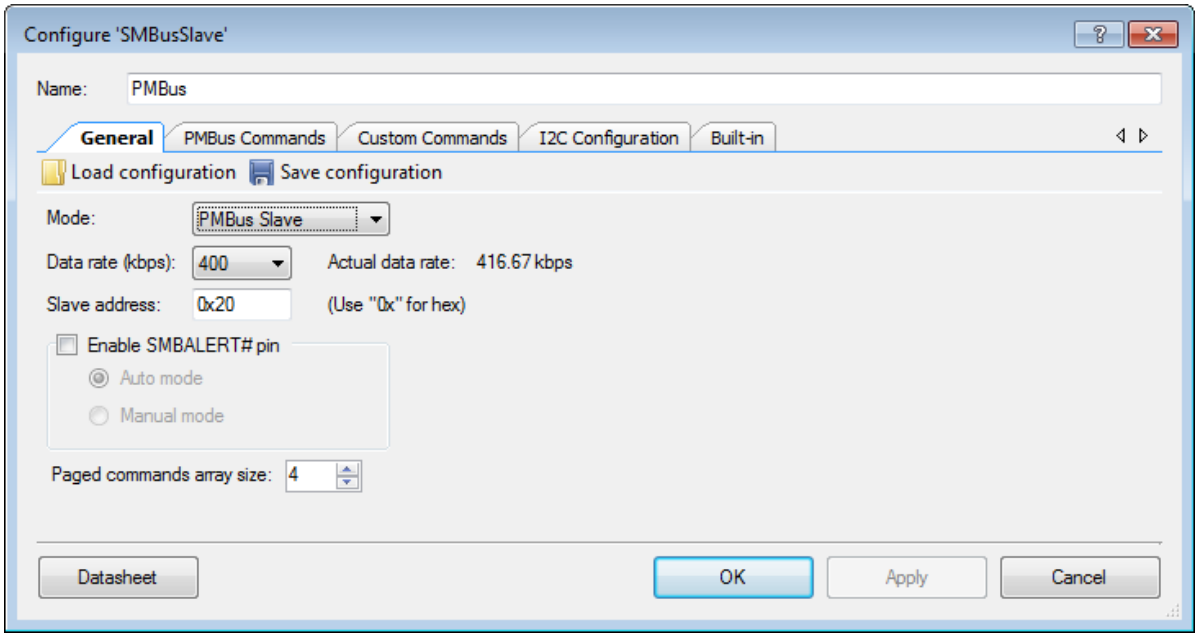

Figure 2 – General Tab

Figure 3 shows the PMBus Commands tab which enables users to select which PMBus commands will be supported by their design. The component handles all of the low level communications, protocol, command verification, and memory allocation for the chosen set of PMBus commands.

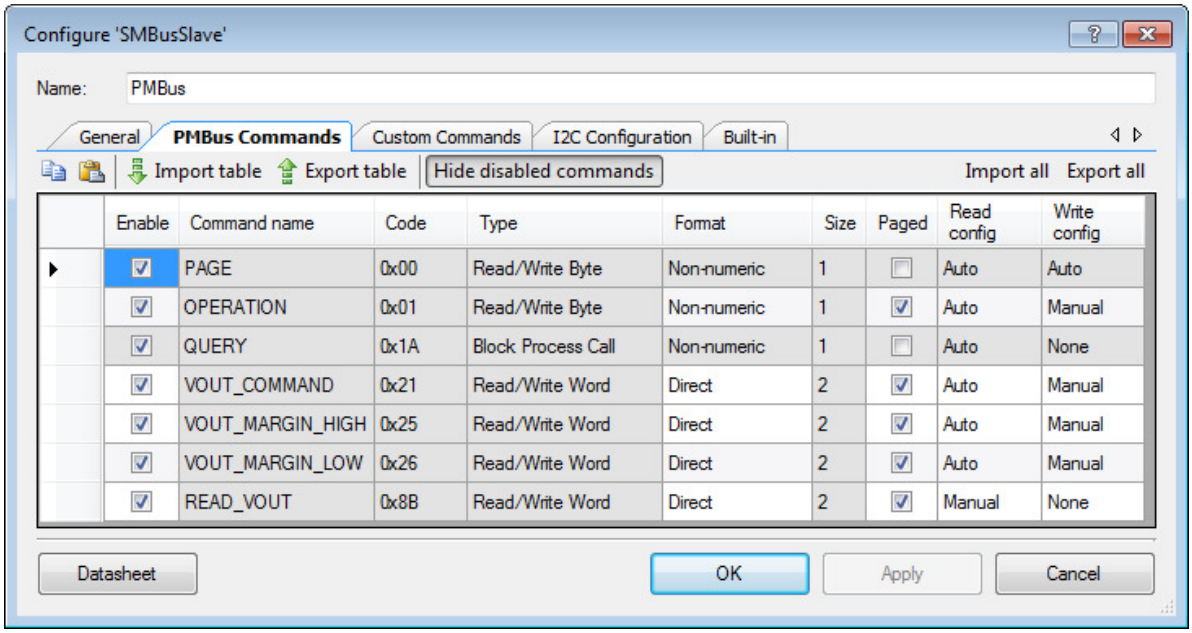

Figure 3 – PMBus Commands Tab

P

The Custom Commands tab allows the entry of manufacturer specific PMBus commands. This example does not include any custom commands, so that tab remains empty. Finally, the I2C Configuration tab sets low level I^2C hardware block properties. This example uses the default values on this tab.

Project Description

The simulated power supervisor application is setup in this example project with the voltage parameters as shown in Table 1. The Power Monitor component is configured to generate a fault signal if the converter nominal voltages are outside the range specified by the under-voltage (UV) and over-voltage (OV) thresholds. Any fault condition will cause the Voltage Sequencer component to power-down all converters.

Converter	Nominal Voltage	UV Fault Threshold	Margin Low Voltage	Margin High Voltage	OV Fault Threshold
V1	800 mV	680 mV	760 mV	840 mV	920 mV
V2	1000 mV	850 mV	950 mV	1050 mV	1150 mV
V3	1500 mV	1275 mV	1425 mV	1575 mV	1725 mV
V4	2000 mV	1700 mV	1900 mV	2100 mV	2300 mV

Table 1 – Simulated Power Supervisor Power Converter Settings

Notice that the Margin Low and Margin High voltage levels are inside the OV/UV Fault Thresholds, so it is safe margin any of the rails low or high without generating a fault condition.

Users can interact with this example project by sending PMBus commands over the I^2C physical layer connection to the PMBus Slave component. The PMBus specification defines a detailed and thorough command set for interfacing to power, thermal and other common system management systems. However, the way in which a particular system carries out a PMBus command is likely to be application specific. PMBus defines the protocol and the command set, but it is up to the individual system designer to implement each command. The PMBus Slave component follows this model.

When a command is enabled in the component, the user selects the access configuration for both Reads and Writes of that command. A selection of “None” is used to specify read-only or write-only commands. “Auto” means that the component will automatically handle the command without any user firmware required. “Manual” means that user firmware will handle the command. Any command may mix both Auto and Manual modes. This makes sense for commands where writing requires some kind of additional processing or further action (Manual), but reading just needs to return the value previously written (Auto).

The example project supports the commands shown in Table 2. See the PMBus Specification Part II for more detailed explanations of these and other commands.

Command	Code	Type	Description
PAGE	0x00	Byte R/W	Sets the current PMBus page. In this example, each simulated power converter rail is accessed through a different page
OPERATION	0x01	Byte R/W	OPERATION is used to control the power state of individual rails. The byte of payload determines the function. Supported codes are: 0x00=Rail Off, 0x80=Rail On, 0x94=Margin Low, 0xA4=Margin High
QUERY	0x1A	Process Call	Reports which commands are supported along with basic command properties
VOUT_COMMAND	0x21	Word R/W	The nominal voltage setting of the power converter (page indexed)
VOUT_MARGIN_HIGH	0x25	Word R/W	Voltage setting of the power converter when Margin High is active (page indexed)
VOUT_MARGIN_LOW	0x26	Word R/W	Voltage setting of the power converter when Margin Low is active (page indexed)
READ_VOUT	(0x8Bu)	Word Read Only	Measured power converter voltage (page indexed)

Table 2 – PMBus Commands Supported in the Example Project

For commands that are configured as Manual, user firmware must participate in handling the command. In this example, the user firmware command handler is implemented in the function `HandlePMBusCommand()` located in main.c. This function is called periodically from the main program loop, and when called, it must execute quickly and must not block. This is necessary because the PMBus component may disable interrupts and stretch the PMBus clock signal (SCL) until user code has handled the command.

Some PMBus component functionality is enabled through “merge code” sections. Merge code sections are areas of the automatically generated component firmware API’s where a user can insert custom code. During the build process, PSoC Creator preserves user code that has been added to these sections. The component datasheet documents which PMBus API’s include merge sections. This example project includes custom merge code in the following API function:

PMBus_StoreComponentAll()

This API is responsible for updating the PMBus register/command values with values that come from other components in the design (e.g. the Power Monitor and Voltage Sequencer) at startup.

Expected Results

Run the Cypress Bridge Control Panel software application by clicking on Start → All Programs →Cypress → Bridge Control Panel 1.6 → Bridge Control Panel 1.6. Figure 4 shows the Bridge Control Panel software application when successfully launched.

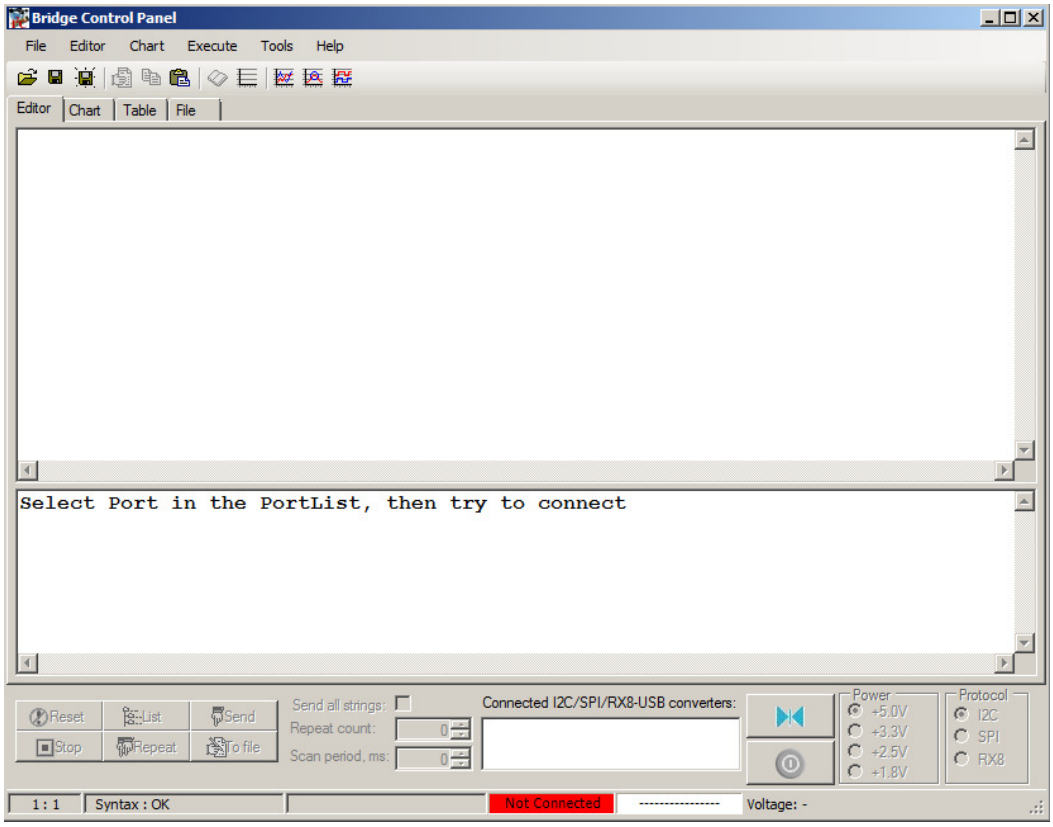

Figure 4 – Bridge Control Panel Application

Connect a MiniProg3 to the USB port on your computer. The Bridge Control Panel will automatically detect the MiniProg3 and connect to it. Connect the ground reference on the PSoC Development kit to the GND connection on the 5-wire white connector on the MiniProg3. Ensure also that the SDA data signal from PSoC (P0[0]) is connected to SDAT and that the SCL clock signal from PSoC (P0[1]) is connected to SCLK on the 5-wire white connector on the MiniProg3. Check Power = +5.0V and click on the Power icon on the Bridge Control Panel. When everything is correcty configured, the status display boxes at the bottom of the application will both be highlighted in green color and display “Connected” and “Powered”. To confirm the communication channel to PSoC is working properly, click on the “List” button. The status window of the Bridge Control Panel should display the I2C address as shown below. If the

display shows “No devices found”, double check power, SCL and SDA connections to the MiniProg.

```
Devices list:  8bit   7bit
     address:  40     20
```

To setup the Bridge Control Panel, click on Chart→Variable Settings. Click on Load and navigate to the location where you saved the PMBus Example project to your hard drive. Go into the PMBusExample.cydsn folder and select the PMBus_Commands_BCP.ini file. Click on OK.This will configure the graphical charting functionality of the Bridge Control Panel. Next, in the PMBus Example project inside PSoC Creator, go to the Workspace Explorer and open the PMBus_Commands_BCP.iic file by double clicking on it. Select all of the contents of this file and then copy. Go back to the Bridge Control Panel and paste in the copied text into the Editor Tab. When you have done this successfully, the Bridge Control Panel should look like this:

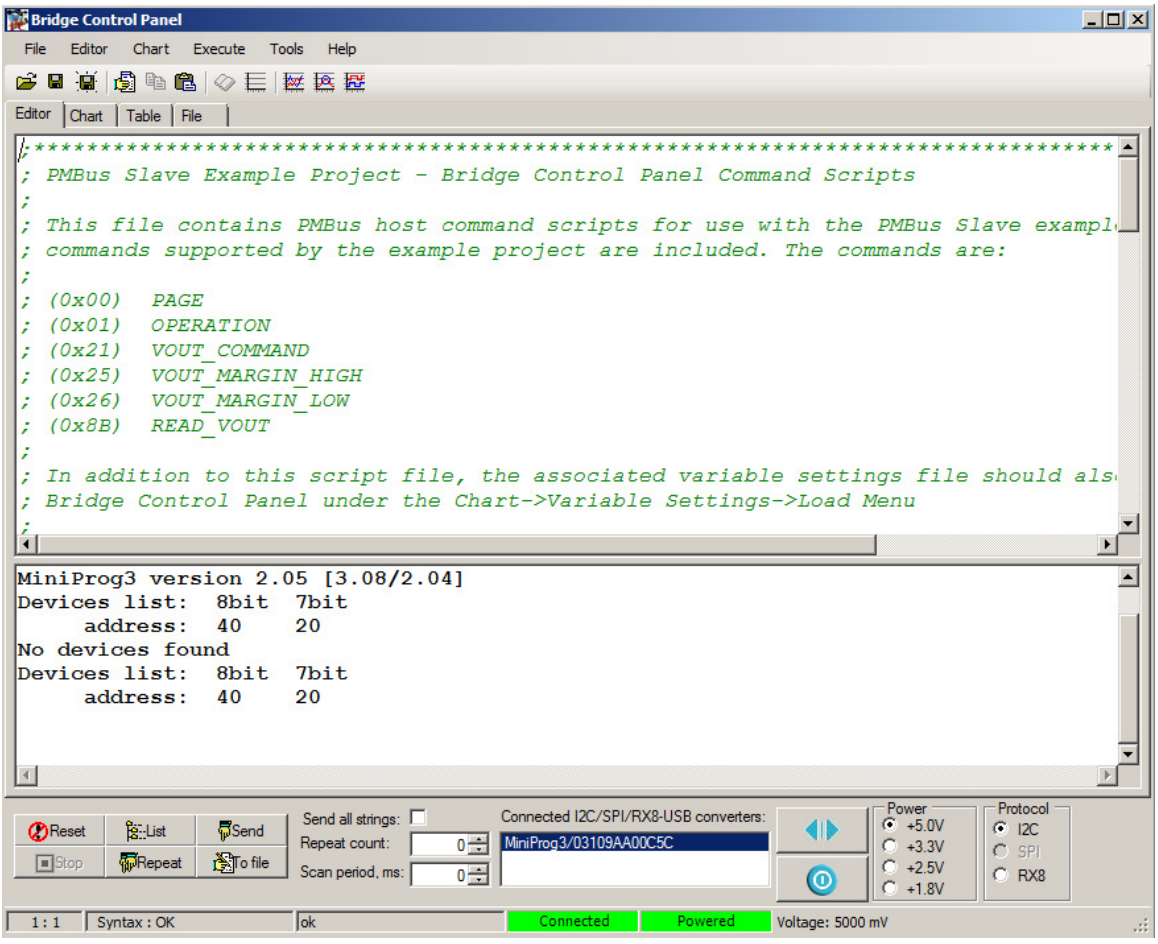

Figure 5 – Configured Bridge Control Panel Application

Now that communications has been established, PMBus commands can be sent. There are five pre-configured PMBus command scripts included in the .iic file that was pasted into the Editor tab as follows:

1. Read all simulated power converter voltages for graphical charting
2. Change simulated power converter output voltages
3. Margin simulated power converter output voltage high
4. Margin simulated power converter output voltage low
5. Set simulated power converter output voltage back to nominal level

Each script includes several PMBus commands sent back-to-back to PSoC over I^2C. To send the script, click on any of the lines associated with the script of interest and click on Enter. The commands that are sent to PSoC will be displayed in the Status Window. Click on the Repeat button to continually send the script over and over. This is very useful when using script #1 above. When the script is set to Repeat mode, click on the Chart tab to see a real-time graphical display of the simulated power converter voltages. The units on the Y-axis are in mV.

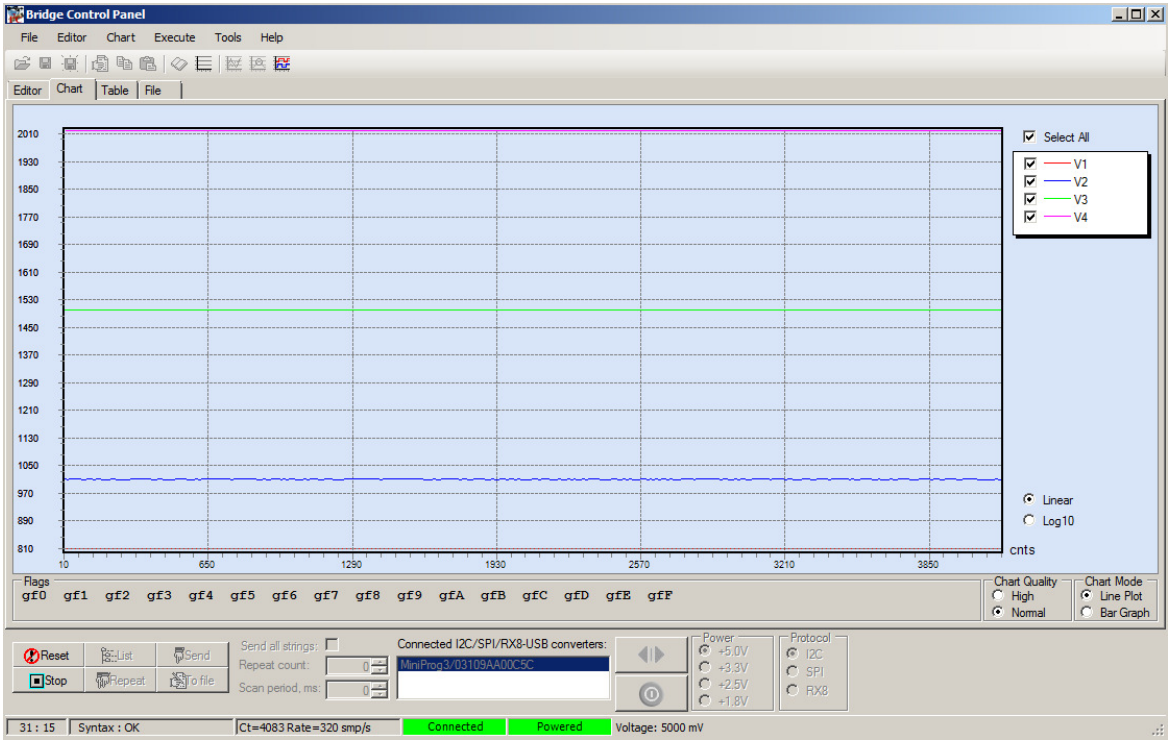

Figure 6 – Bridge Control Panel Application Grpahical Charting Display

Click on Stop to terminate the PMBus command script. Refer to the comments in the .iic file for a more detailed description on what the supplied scripts do and which specific PMBus commands are used.

Related Material

AN76474 - PSoC® 3 Power Supervisor

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709

Phone : 408-943-2600
Fax : 408-943-4730
Website : www.cypress.com

© Cypress Semiconductor Corporation, 2012. The information contained herein is subject to change without notice. Cypress Semiconductor Corporation assumes no responsibility for the use of any circuitry other than circuitry embodied in a Cypress product. Nor does it convey or imply any license under patent or other rights. Cypress products are not warranted nor intended to be used for medical, life support, life saving, critical control or safety applications, unless pursuant to an express written agreement with Cypress. Furthermore, Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress products in life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges. PSoC® is a registered trademark, and PSoC Creator™ and Programmable System-on-Chip™ are trademarks of Cypress Semiconductor Corp. All other trademarks or registered trademarks referenced herein are property of the respective corporations.

This Source Code (software and/or firmware) is owned by Cypress Semiconductor Corporation (Cypress) and is protected by and subject to worldwide patent protection (United States and foreign), United States copyright laws and international treaty provisions. Cypress hereby grants to licensee a personal, non-exclusive, non-transferable license to copy, use, modify, create derivative works of, and compile the Cypress Source Code and derivative works for the sole purpose of creating custom software and or firmware in support of licensee product to be used only in conjunction with a Cypress integrated circuit as specified in the applicable agreement. Any reproduction, modification, translation, compilation, or representation of this Source Code except as specified above is prohibited without the express written permission of Cypress.

Disclaimer: CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Cypress reserves the right to make changes without further notice to the materials described herein. Cypress does not assume any liability arising out of the application or use of any product or circuit described herein. Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress' product in a life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.

Use may be limited by and subject to the applicable Cypress software license agreement.