PMBus Slave Example Project
1.0

Features

- Demonstrates basic usage of PMBus Slave component in thermal management application

General Description

This example project demonstrates how to get started using the PMBus Slave component in a simulated thermal management application. In place of an external temperature sensor, the on-chip temperature sensor is used. It is combined with the ADC SAR Sequencer component to create a simple thermal manager that can run on a PSoC 4 Pioneer kit without any special hardware. The example supports a small subset of the PMBus command set to provide the voltage and temperature reading.

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 4 Pioneer kit (CY8CKIT-042).

1. Set jumper J9 to 3.3V position.
2. Connect the PMBus host SDA signal to P0[5] and SCL signal to P0[4] and P0[7]. 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.
3. Connect a source voltage to P0[1] on CY8CKIT-042 kit. The voltage range should be between 0V and 1.024 V for correct reading.
4. Connect a USB cable to the PSoC 4 Pioneer Kit DVK and a PC

Project Configuration

The PSoC Creator design schematic is shown in **Figure 1**. Note that most of the schematic in this example is occupied by the simulated thermal manager application. The PMBus slave component only requires two bi-directional pin connections for I^{2}C clock (SCL) and data (SDA) and one input pin connection for 25 ms SCL timeout detection.

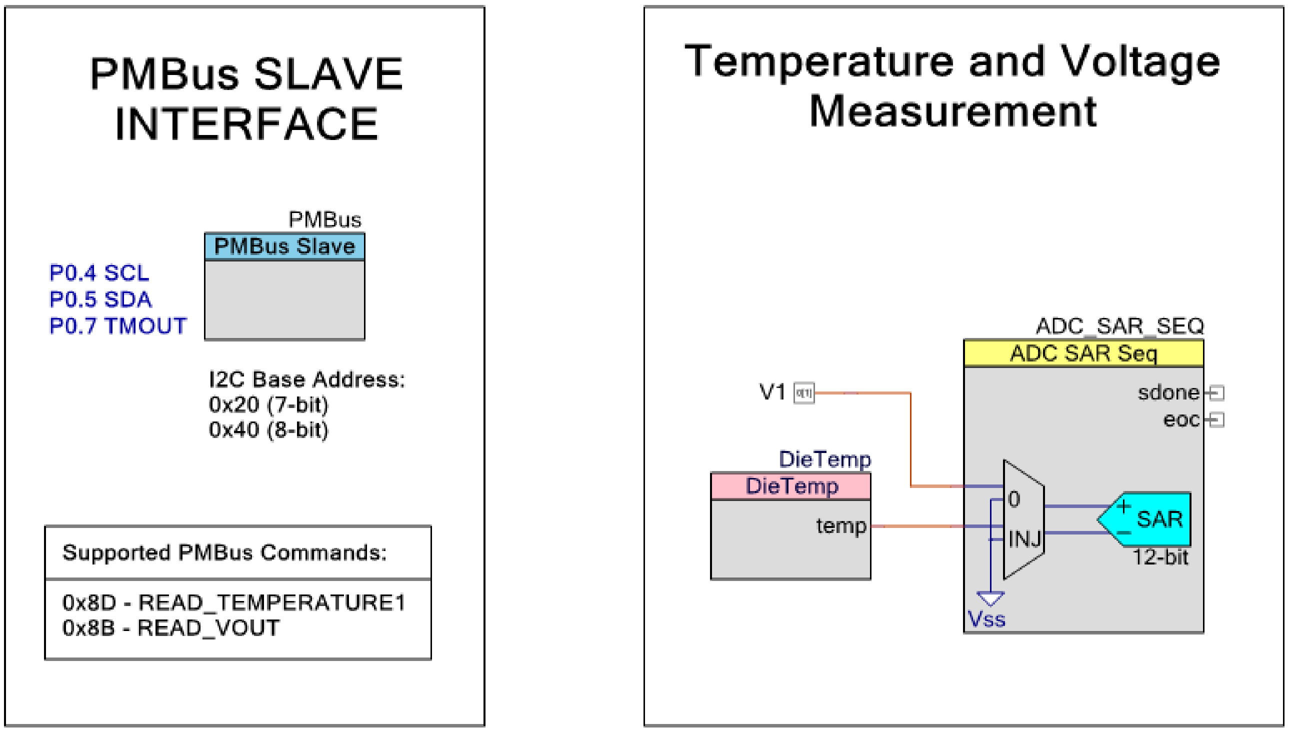
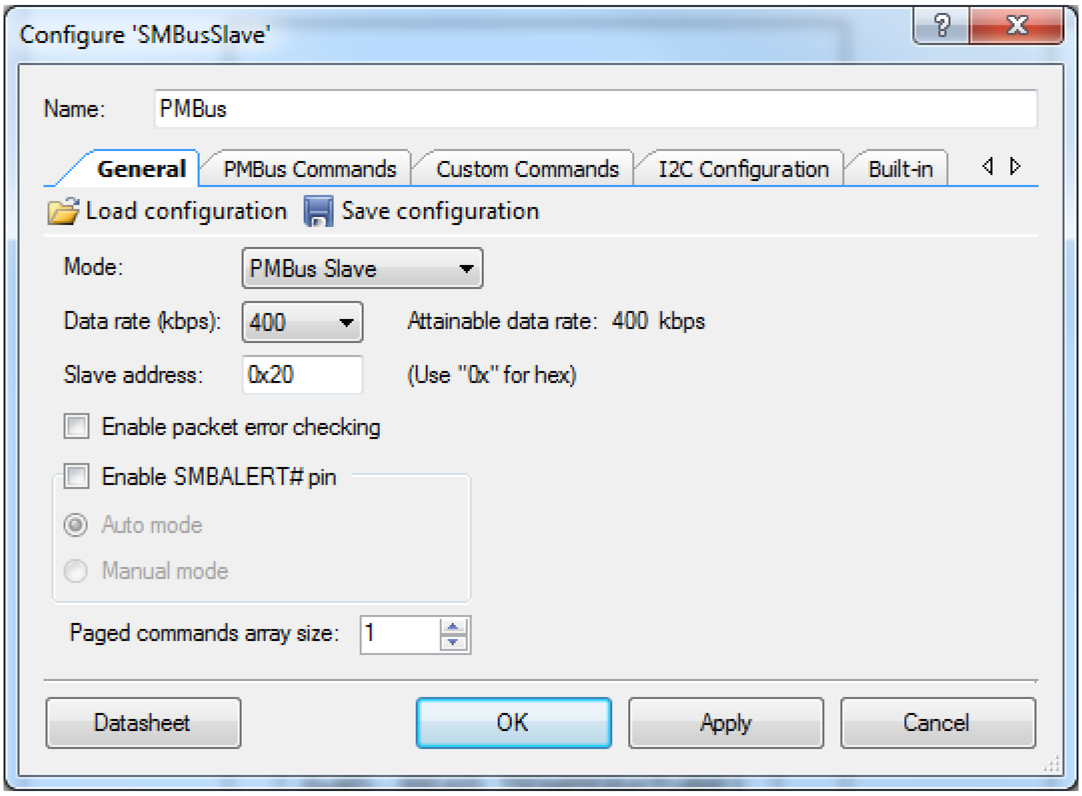


Figure 1. PMBus Slave Component Example Project Schematic

The simulated thermal manager application is comprised of the DieTemp connected to the ADC SAR Sequencer which digitizes the reading. Then the reading is converted to a temperature value. Additionally channel 0 of the ADC is connected to an analog pin and may be used for measuring voltage in the range of 0..1.024V.

An additional example project exists for the PMBus component and is targeted to PSoC 3 and PSoC 5LP architectures.

Open the Configure Dialog for the PMBus Slave by double clicking on the component. Figure 2 shows the General tab that enables the user 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 is only one page since the project consists of one logical unit. Therefore, the **Paged commands array size** parameter is set to 1.



Name: PMBus

Mode: PMBus Slave

Data rate (kbps): 400 Attainable data rate: 400 kbps

Slave address: 0x20 (Use "0x" for hex)

☐ Enable packet error checking

☐ Enable SMBALERT# pin

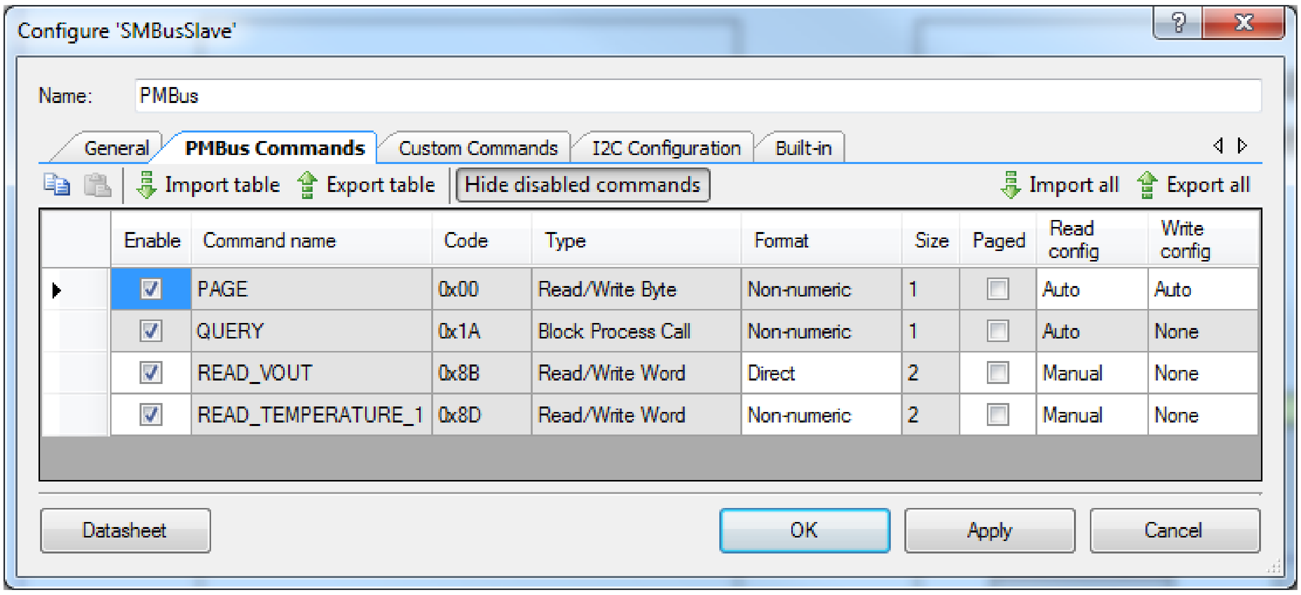
● Auto mode

○ Manual mode

Paged commands array size: 1

Figure 2. General Tab

Figure 3 shows the PMBus Commands tab which enables the user 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.



Name: PMBus

Enable	Command name	Code	Type	Format	Size	Paged	Read config	Write config
✓	PAGE	0x00	Read/Write Byte	Non-numeric	1	☐	Auto	Auto
✓	QUERY	0x1A	Block Process Call	Non-numeric	1	☐	Auto	None
✓	READ_VOUT	0x8B	Read/Write Word	Direct	2	☐	Manual	None
✓	READ_TEMPERATURE_1	0x8D	Read/Write Word	Non-numeric	2	☐	Manual	None

Figure 3. PMBus Commands Tab

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^{2}C hardware block properties. This example uses the default values on this tab.

Project Description

In the main function all the components are started, the SAR Sequencer is configured, and ADC conversion is started. The for loop in main.c checks for a pending PMBus transaction and handles it.

The user can interact with this example project by sending PMBus commands over the I^{2}C 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 the application specific. The 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. Selecting "None" is for specifying read-only or write-only commands. "Auto" means that the component will automatically handle the command without any user firmware required. "Manual" means that the user's 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 1. 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, only one page is enabled.
QUERY	0x1A	Process Call	Reports which commands are supported along with basic command properties.
READ_TEMPERATURE_1	0x01	Word Read Only	Measured temperate value in Celsius
READ_VOUT	(0x8Bu)	Word Read Only	Measured input voltage in mV

Table 1. PMBus Commands Supported in Example Project

For commands that are configured as "Manual", the user's firmware must participate in handling the command. In this example, the user's 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 the user's code has handled the command.

Expected Results

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

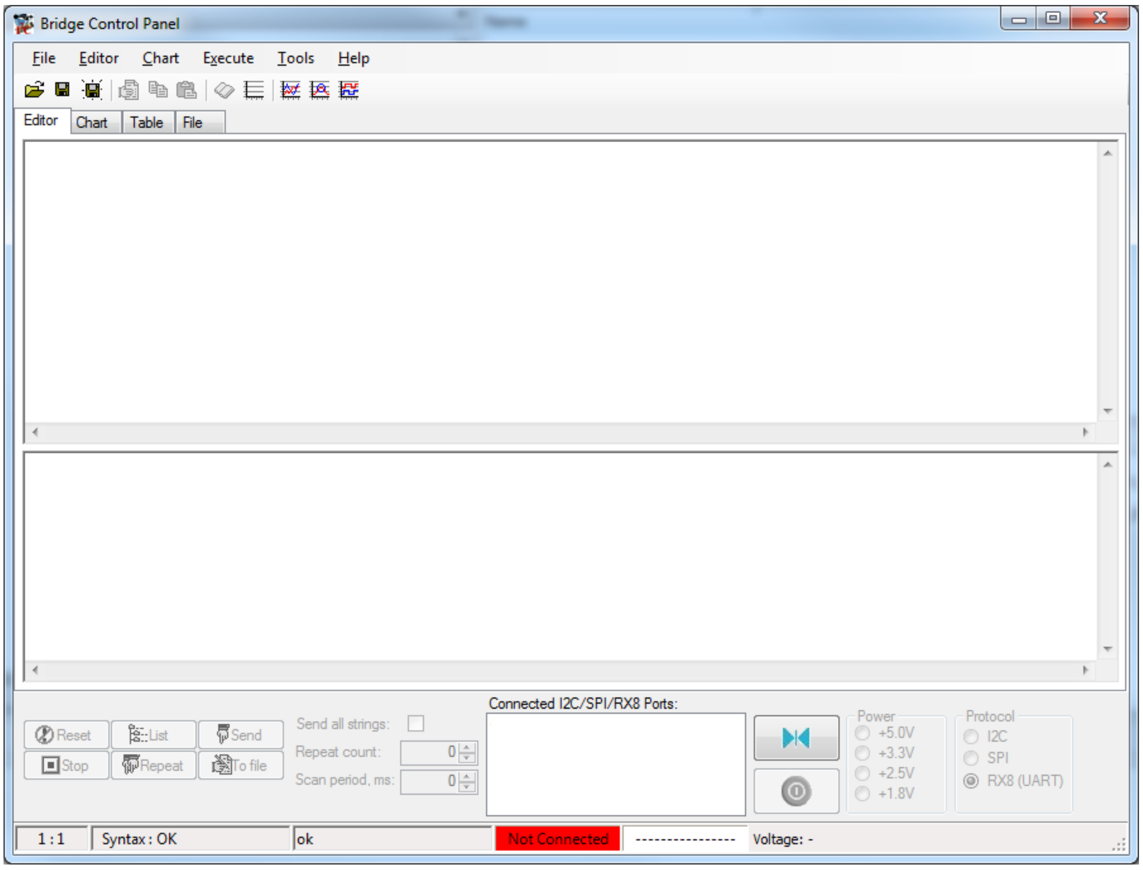
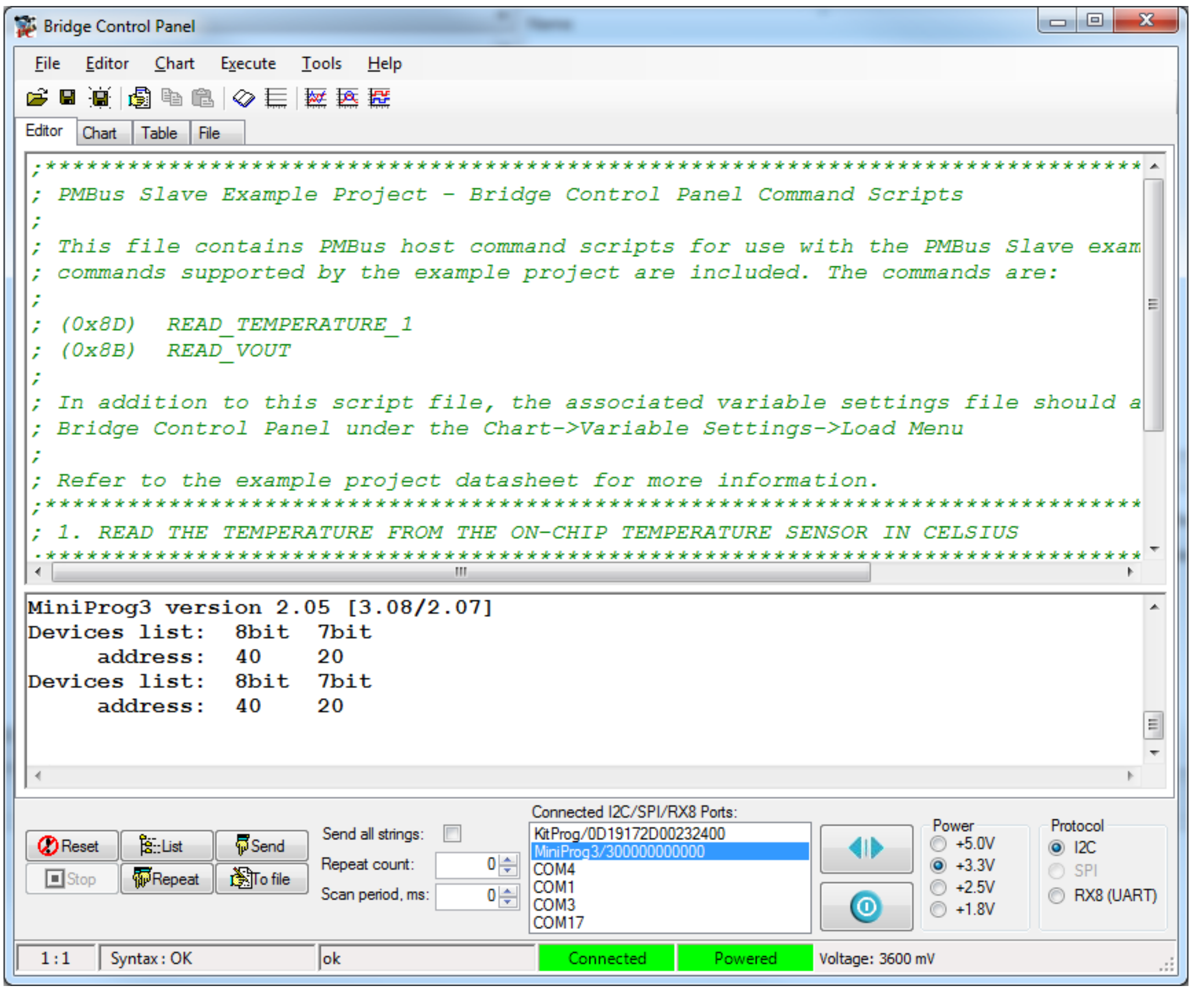

Bridge Control Panel
File Editor Chart Execute Tools Help
Editor Chart Table File
Reset List Send
Stop Repeat To file
Send all strings:
Repeat count: 0
Scan period, ms: 0
Connected I2C/SPI/RX8 Ports:
Power +5.0V +3.3V +2.5V +1.8V
Protocol I2C SPI RX8 (UART)
1 : 1 Syntax : OK ok Not Connected ---------------- Voltage: -

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[5]) is connected to SDAT and that the SCL clock signal from PSoC (P0[4]) along with the SCL timeout detection signal (P0[7]) are connected to SCLK on the 5-wire white connector on the MiniProg3. Check Power = +3.3V and click on the Power icon on the Bridge Control Panel. When everything is correctly configured, the status display boxes at the bottom of the application will be both highlighted green 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 PMBusThermExample.cydsn folder and select the PMBus_Commands_BCP.ini file. Click 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:


Bridge Control Panel
File Editor Chart Execute Tools Help
Editor Chart Table File
Reset List Send Stop Repeat To file

```
;***********************************************************************
; PMBus Slave Example Project - Bridge Control Panel Command Scripts
;
; This file contains PMBus host command scripts for use with the PMBus Slave exam
; commands supported by the example project are included. The commands are:
;
; (0x8D)   READ_TEMPERATURE_1
; (0x8B)   READ_VOUT
;
; In addition to this script file, the associated variable settings file should a
; Bridge Control Panel under the Chart->Variable Settings->Load Menu
;
; Refer to the example project datasheet for more information.
;***********************************************************************
; 1. READ THE TEMPERATURE FROM THE ON-CHIP TEMPERATURE SENSOR IN CELSIUS
;***********************************************************************
```

```
MiniProg3 version 2.05 [3.08/2.07]
Devices list:  8bit  7bit
      address:  40    20
Devices list:  8bit  7bit
      address:  40    20
```

Send all strings: ☐
Repeat count: 0
Scan period, ms: 0

Connected I2C/SPI/RX8 Ports:
KitProg/0D19172D00232400
MiniProg3/300000000000
COM4
COM1
COM3
COM17

Power: +5.0V [✓ +3.3V] +2.5V +1.8V

Protocol: [✓ I2C] SPI RX8 (UART)

1:1 | Syntax : OK | ok | Connected | Powered | Voltage: 3600 mV

Figure 5. Configured Bridge Control Panel Application

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

1. Read the temperature from the on-chip temperature sensor in Celsius
2. Read measured voltage from the channel_0 of the ADC in units of mV

To send the command, click on the command and press Enter. The command that is sent to PSoC will be displayed in the Status Window. Click on the Repeat button to continually send the command over and over. When the command is set to Repeat mode, click on the Chart tab to see a real-time graphical display of the simulated temperature sensor reading or the voltage measured on channel 0 of the ADC. The units on the Y-axis are Celsius for the temperature and in mV for the voltage. Click on Stop to terminate the PMBus command.

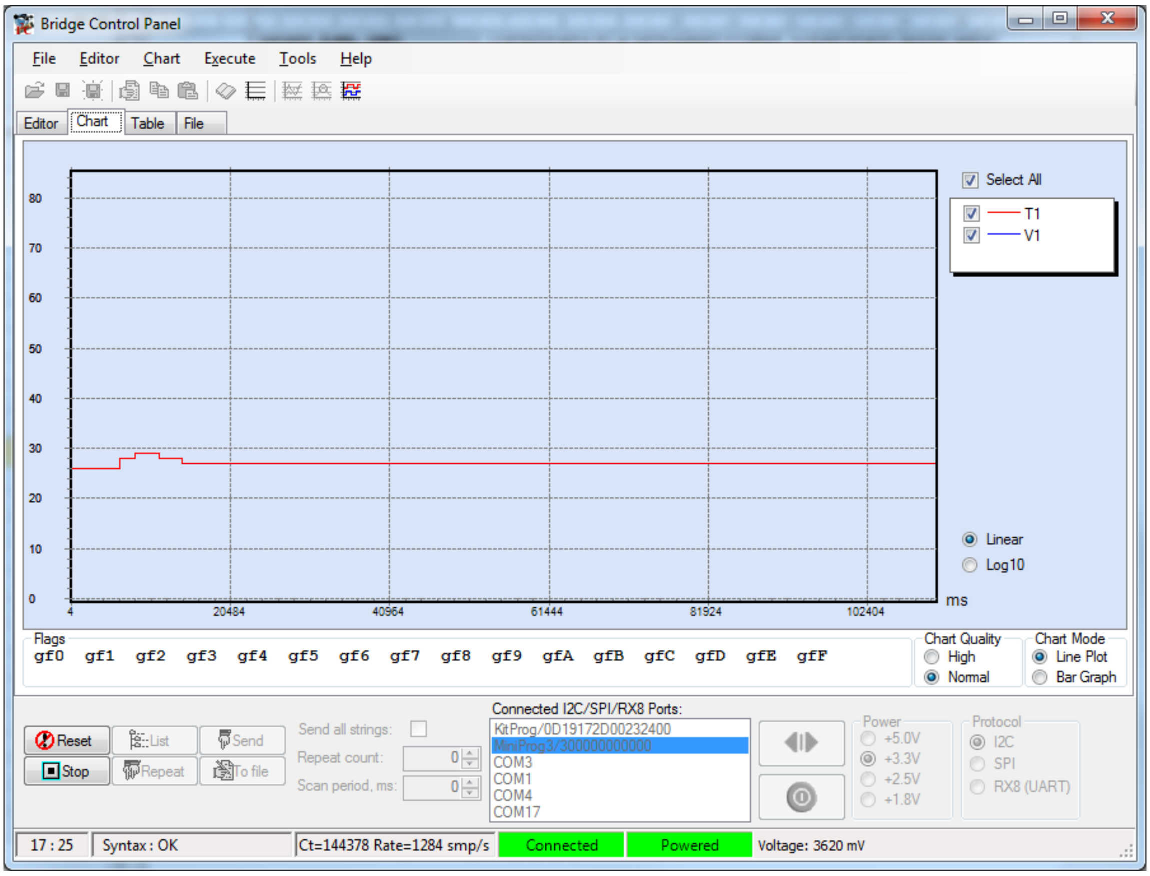


Figure 6. Bridge Control Panel Application Grpahical Charting Display