User Manual

Communicating with the Compact Power Line (CPL) modules Using PMBus™ protocols over the I²C bus

imagination at work

July 6, 2015

Table of Contents

2 Introduction

The Compact Power Line continues to evolve since its introduction over 8+ years ago. The latest modules are significantly more efficient, provide a lot more power, and offer enhanced communications features such as reading input voltage or computing input power consumption, all in a retrofit able package. This application note concentrates on two GE Digital Power Insight™ software development tools that demonstrate the I²C based PMBus™ protocol compliant communications capabilities of the modules.

The first tool is an interactive Graphical User Interface (GUI) that can communicate to and display the status of all 8 power supplies that could get connected to a single I²C bus.

The second tool is the Command Line Interface (CLI) that issues individual commands and receives back data from the power supply. The CLI supports a powerful scripting capability that can be used to automatically test a set of features.

3 The GE Digital Power Insight™ development tool

This User Manual starts with detailed information on installing the configuration software that places all drivers and executable programs into a directory structure consistent with the Windows environment. The section ends with describing how to connect the various elements to initiate communications.

The second element describes in detail the simple, graphically-oriented User Interface, referred to here as the Digital Power Insight GUI. This tool, designed for users who do not want to do the detailed level of programming needed to communicate with one or more modules, is easy to learn. It is ideal for the Power Design Engineer who wants to access digital functionality in a limited number of power modules without getting into the details of PMBus commands and programming. Finally, the GUI enables a user to communicate with a system containing multiple modules (up to a total of eight) by supporting various functions such as loading commands into the power supplies, "on-the-fly" adjustment of module parameters and functions, and real-time display of status, alarm, and measured data.

The third element is a low level command line interface tool called the Digital Power Insight Command Line Interface or CLI. The CLI provides a host of capabilities ranging from invoking simple PMBus commands to scripting complex test programs with multiple PMBus commands that can be used to control and acquire data from the power supply. By providing a lower level interface to multiple power supplies, this tool delivers comprehensive support of programming activities as well as testing/debugging capability for users who are developing software to control the power supplies or want to examine specific commands or controller-module interactions in detail. The CLI also provides a means to query and change settings of the USB Interface Adapter.

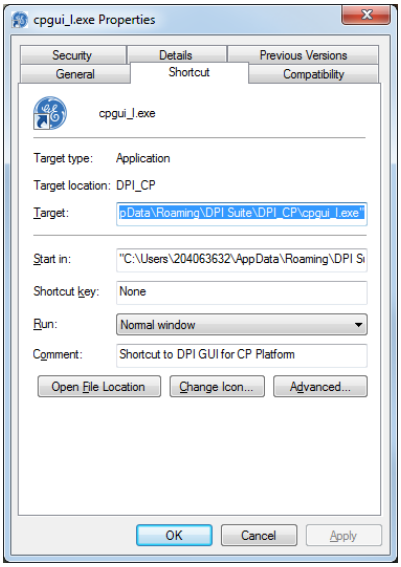
4 Installing the DPI Software

The DPI software is installed using a self-executable install file. To install the software double click on the file **dpi_CP_CAR.msi**. This results in the screen shown to the right. Click on <next> and follow the successive installation instructions.

The DPI Software installation automatically places icons for the **dpi_cli.exe, dpi_car.exe**, and **cpgui.exe** programs on the computer's Windows Desktop**.**

This **Start In** location is driven by the windows operating system and it has been our experience that it may not show the correct file structure. Click on the <next> button and on the following screen click <install>

The setup program finishes with the statement **DPI CAR CP has been installed successfully.** Click on <finish>.



The installed location of the DPI Program files vary depending on the computer, permissions, and file structure of the specific machine. To determine the actual location right click on the icon that was created by the installation and click on properties. The screen to the left will appear. The Target area shows the correct location of the executable.

DPI programs create log files every time the program is executed, in the same directory where the actual executable resides. The user may either choose to have the DPI Program files installed in an alternate directory that can be specified during the installation, or move the program files into a directory that is simpler to access after the installation.

The software installation also attempts to install a driver file that allows the computer to communicate to the GE USB Interface Adapter.

If the GE USB Interface Adapter driver file could not be installed automatically, a manual installation process may be necessary in order to use the programs. Different driver files are available for 32 bit and 64 bit systems.

4.1 Starting the DPI software

The programs can also be started by clicking on the **windows** icon and entering the name of the executable in the *search program and files* box right above the windows icon, provided that the directory location of the executable is known.

The location of the executable can be found by right clicking on the executable icon, and clicking on the 'properties' feature in the pull down menu. The resulting properties screen shows the **Start In** location where the executable is installed.

4.2 USB Interface Adapter driver setup

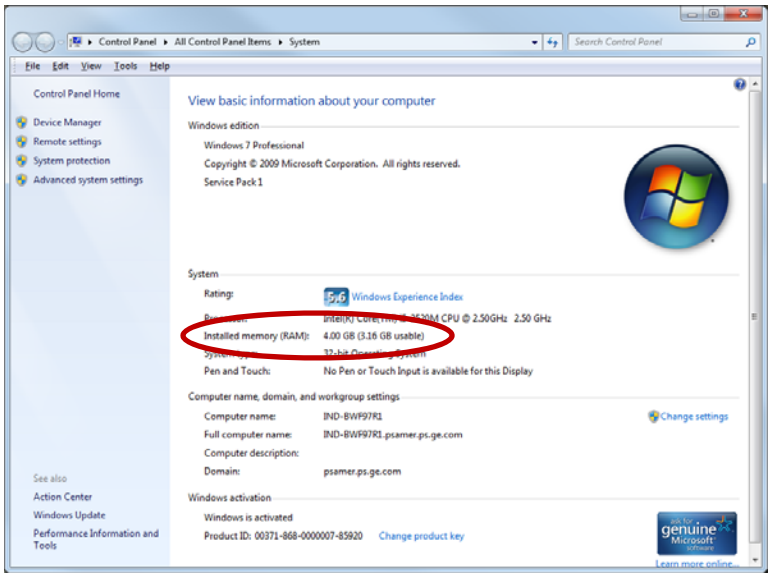
The files ***GE_Power_Electronics_USB_Interface_Adapter.inf*** and ***GE_Power_Electronics_USB_Interface_Adapter_64bit.inf*** contain the Interface adapter driver for Windows 32-bit and 64-bit operating systems, respectively. During the DPI software installation, the file ***GE_Power_Electronics_USB_Interface_Adapter.inf*** Is automatically placed in the C:\Windows\System32 directory for 32 bit systems.

The file ***GE_Power_Electronics_USB_Interface_Adapter_64bit.inf*** is placed in the C:\Windows\SysWOW64 directory for 64 bit systems. Please verify that the appropriate driver file has been installed on your PC operating system.

4.3 32 bit or 64 bit

To determine whether your computer is 32 bit or 64 bit follow the following steps in Windows7 environments. (Similar steps should be available in previous Windows based systems).[Note: You can skip this task if the computer locates the .inf file as evidenced by a normal start of either the GUI or the CLI]

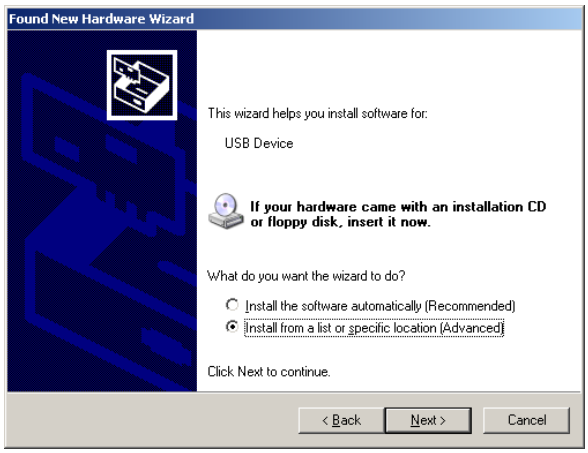
- On the desktop locate the 'Computer' icon and right click on it.
- A number of selections should appear, the last selection is 'properties' click on it
- Under the System subsection, the System Type: tells you whether your computer is either a 32 or 64 bit machine.



4.4 Recognition and configuration of the USB Interface Adapter

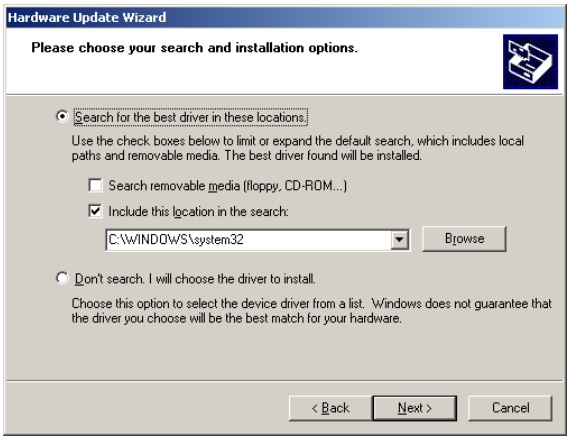
Plug the USB Interface Adapter into an open USB port on your computer. The computer should recognize that new hardware has been connected and it will ask you for a location for the driver for this new hardware.

In the screen shown above, select **No, not this time**, and then click on the **Next>** box. This will bring up the screen shown below:



Select **Install from a list or specific location (Advanced)** and click on **Next>** to proceed to the next screen.

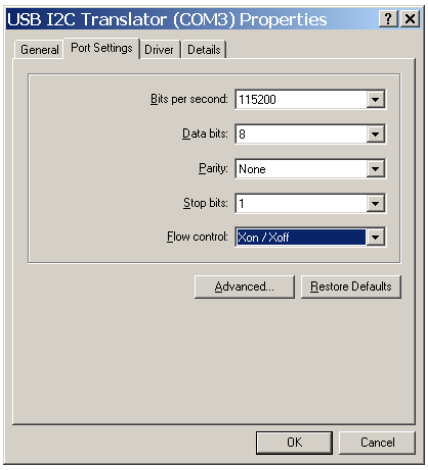
The resulting screen is shown below



Select **Search for the best driver in these locations,** followed by checking the **Include this location in the search** and enter C:\Windows\System32 or C:\Windows\SysWOW64 as appropriate for your PC Operating System.

Click on the **Next>** button and the computer should now install the driver for the USB Interface Adapter. Wait until the driver installation completes before proceeding to the next step.

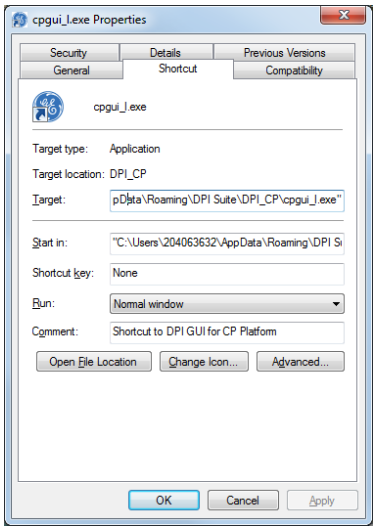
4.5 Communication Port Setup



In order to improve communication speed, settings of the COM port that is used by the USB Interface Adapter should be changed. This step is not necessary, but may result in faster communication speeds. In Microsoft Windows, from the **Start** button, select **Settings** and then **Control Panel**. Within the **Control Panel** window, double click on **System,** select the **Hardware** tab on the **System Properties** window, and then select **Device Manager**. Under **Device Manager** navigate down to **Ports** and double click on the **USB Interface Adapter** icon. The port properties should be displayed. Select the **Port Settings** tab and change the transmission speed to ***115200*** and flow control to ***Xon/Xoff*** by selecting and clicking on the drop down menu. The screen shot for this step is shown below.

Once these steps are completed, the DPI Tool Set is ready to use.

4.6 Changing the location of files Created by DPI



The GUI and CLI tools within DPI create files that keep logs of commands, states and readings. These log files are created in the directory where the DPI executable software is installed. To determine the location of these files right click on any of the dpi icons that have been placed on the desktop, then click on 'properties' in the window that appeared after the right click. The properties window shows the installed location in the **Target:** area. Navigate into the DPI_CP directory where you can locate both the executable and the created .txt log files.

The **Target** location for the executable file can be moved if rapid access to the log files are desired. The location could be the desktop or any other convenient directory. You may want to delete all the DPI shortcut icons created by the installer on the desktop, but this is not necessary. Move, or copy, the DPI_CP directory to the desktop. (you need to move this directory in its entirety, since it includes many subroutines used by the executable), On the dektop click on the DPI_CP icon. Either create shortcuts for the desired executables, or simply execute the desired program from the DPI_CP directory. The .txt files are now accessible from the desktop/DPI_CP directory.

Filenames are automatically created by the tools. Those created by the GUI have the filename

"cpgui_log_MM_DD_HH_MM.txt"

where "MM_DD" refers to the month and day and "HH_MM" refers to the hour and minute when the file was created. These files can be safely deleted, or renamed as desired.

The contents of the GUI log file can be modified by the user. The following GUI section reviews the steps for modifying the contents of the logfile.

The log file created by the CLI has similar annotations

"dpi-cli-YYYYMMDD-HHMM.txt"

with "YY YYMMDD" referring to the year, month and day when the file was created and "HHMM" referring to the time.

The contents of the log file created by the CLI tool can be truncated as outlined in the <SUPPRESS> command description in the CLI command set. The user therefore has a choice between keeping the 'longer' style or 'abbreviated' style of records. The <SUPPRESS> command can be invoked at any time within CLI. Commands executed after the <suppress> command are recorded in the style directed by the command. The default content is the 'longer' style of recordkeeping. The 'suppress' command provides the only permitted modification of the logfile.

5 The GE Interface Adopter

This adapter is the interface between the USB port available in all computers/notebooks and the I²C port of the modules. The interface is bi-directional either transmitting commands to, or receiving data from, the modules. The adapter receives its operational power from the 5V pin of the USB port. GE Digital Power Insight™ development tools are designed to work in conjunction with the GE Interface Adapter. Other commercially available interfaces will likely not work.

5.1 Pull-up Resistors

The translator has internal pull-up resistors connected to 3V that source a default value of 3.3mA for the clock and data lines. Other possible values are 0.9mA, or 0.44mA. The SMBAlert# signal is pulled up to 3V via a 7.5kΩ resistor.

5.2 Changing the Pull-up resistor of the USB Interface Adopter

The pull-up resistor values can be changed with the **P** command.

Possible options are 3.3mA (default), 0.9mA, 0.44mA, or 0 (none). The desired pull-up current capability is specified in mA as an argument for the function.

Examples:

```
>P 0.9
```

```
>P 3.3mA
```

`>P` If no argument is specified, the current value is returned.

5.3 LED Status Indicator

The USB Interface Adapter has a three-color (Green/Orange/Red) LED on one end where the USB interface cable plugs into the adapter. The status of the adapter is displayed as follows:

- Green ON – Normal operation
- Green – Fast Blinking – The translator is communicating on the bus.
- Orange – Blinking or ON – The SMBAlert# line is active (LO). Blinking indicates active communications.
- Red – Internal fault

5.4 I²C Clock Speed Settings

The default value of the I²C clock speed is set at 100kHz. It can also be programmed to 400kHz.

5.5 Changing the I²C Clock Speed Settings

The clock speed setting can be changed by invoking the **Command Line Interface** program. The command used for this function is the **K** command.

This function is used to read or change the I²C clock setting in the USB Interface Adapter. The default value is 100kHz but the clock rate can be changed to 400kHz. The desired clock rate can be specified as 100 or 100kHz or 400 or 400kHz. If this function is used without an argument, the current I²C clock rate in the USB Interface adapter is returned.

Examples:

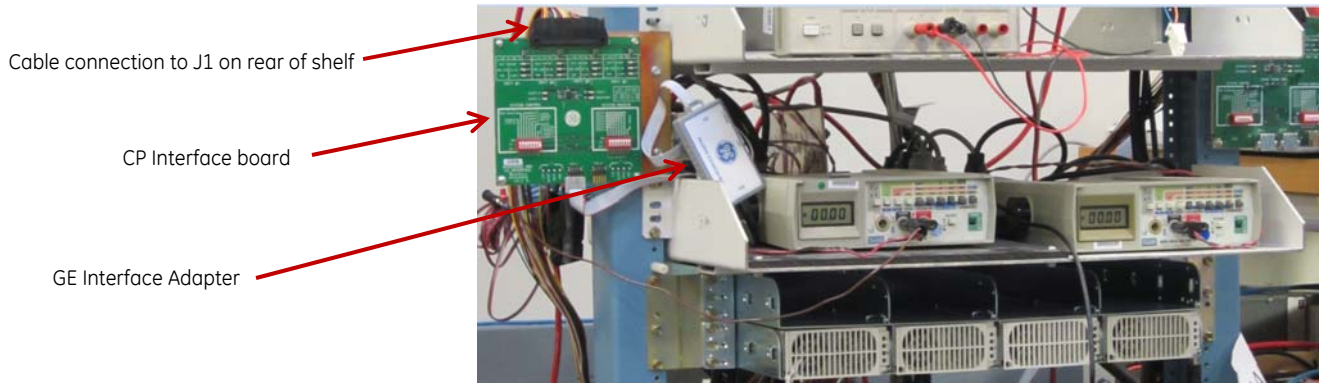
`>K 400Khz`	(sets the USB Interface Adapter Clock speed to 400kHz)
`>K 100`	(sets the USB Interface Adapter Clock speed to 100kHz)
`>K`	(reads the USB Interface Adapter Clock speed setting)

6 Setup

The following components are required to set up a working system

- A desktop or laptop PC running Microsoft Windows XP, Vista or the Windows 7 operating systems
- An open USB interface slot on the computer
- The USB Interface Adapter provided by GE
- The CPL Interface Board
- A USB to Mini-USB cable to connect the USB Interface Adapter to the computer
- A 10-conductor ribbon cable that mates the USB Interface Adopter to the CPL Interface Board
- A GE Critical Power J85480 shelf, or an equivalent mating mechanism to the UUT
- A cable set connected between the P1 connector of the Interface Board and the J1 connector of the shelf

The figure below shows a lab setup using the interface board, the USB Interface Adapter, the interconnection cables, and a CPL shelf with four rectifiers. A number of other instruments are also included in this setup. The interface board is cabled to the J1 connector interface on the rear of the CPL shelf.



7 Using the DPI-CPGUI

The DPI-GUI is a graphically-oriented tool for interfacing to up to eight modules. The tool capabilities include

- Identifying and configuring the addresses of up to eight modules
- Retrieving and displaying status and alarm register information for the eight modules.
- Display of bus status information for the eight modules.
- Acquiring and showing analog data including; input voltage, input power, output voltage, output current, unit temperature, fan speed settings and fan speeds, micro controller firmware revisions, run time
- Setting of key parameters such as; output voltage, output voltage shutdown level, fan speed, or issuing a custom command
- Control of key parameters; who should be controlled (one or all modules), output ON/OFF, LED test routine, read a single status, fan speed control, EEPROM write control, shutdown control (latch or restart), i2c bus control, isolation test, inhibit droop output voltage control, issue bad command or a bad PEC number.
- Setting up periodic polling and display of key information from the modules
- An automated communications log saving all the commands issued on the I^2C bus and status changes with associated time stamps
- Saving to and retrieving data from the EEPROM
- Acquiring which of the two i2c busses has control and executing commands to take over control either manually or automatically at pre-set periods of time.

7.1 Starting the CPGUI Tool

Once the USB Interface Adapter is plugged into the USB port on the Personal Computer, the Green LED of the Adapter should be ON. Apply input power to the module and start the GUI application. Note that **dpi_car.exe** and **cpgui.exe** applications cannot run simultaneously as they share the same COM port connection to the USB Interface Adapter on the personal computer.

The GUI tool can be started by double clicking on the ***cpgui.exe*** or ***dpi_car.exe*** icons located on the desktop, or using the standard ***Windows Start – Programs – DPI – CP GUI*** or ***CAR GUI*** option. In Windows_7, the programs can also be started by clicking on the **windows** icon and entering the name of the executable in the *search program and files* box right above the windows icon. The tool starts up and displays the following screen, (Note: it may take up a few seconds for this screen to appear. This is normal)

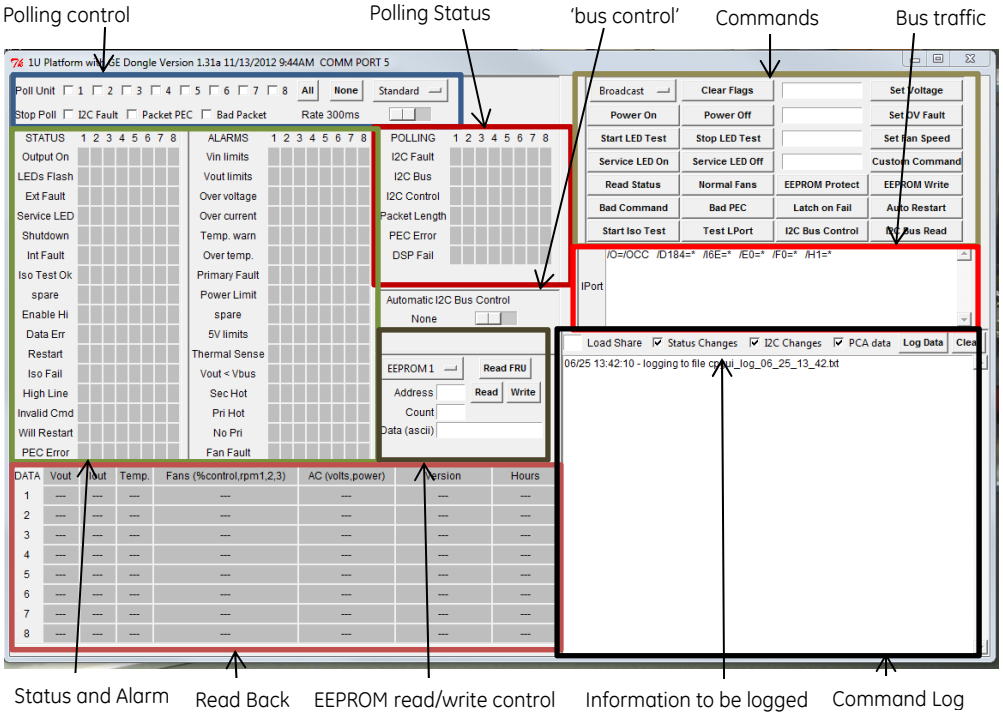
This tool has only the one screen shown below. The various functions of the tool will be explained in the following sections.

Note that the GE USB Interface adapter needs to be plugged into the computer for the GUI program to work properly. If the GE USB Interface adapter cannot be accessed by the GUI software for any reason, it will either display the following window during startup, or it may not start up at all.

In some cases, this can be corrected by removing and re-inserting the USB cable connected to the GE USB Interface adapter, and restarting the GUI program. Correcting this condition may require at least two reinsertions .To exit the GUI program, just click on the X in the top right corner of the tool display window.

8 CPGUI Description

The CPGUI tool display may be divided into the areas shown in the figure below as an aid to describing its functionality. On the first line the executable revision number, the date of creation of the executable and the communications port being used are displayed.



8.1 Polling control

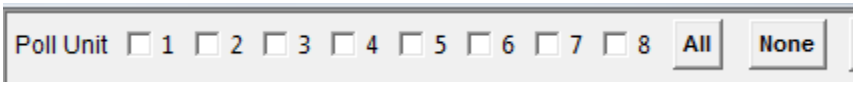
This area is used to select which modules are to be polled, what rate they should be polled at, and under what conditions should polling be automatically stopped.

Module addresses are set by the unit_ID and shelf_ID signal pins of the power supplies. The internal DSP translates these two signal pins to three bits A2, A1, and A0 components of the address field. .....000 is unit #1 and .....111 is unit #8. This is all done automatically. There are three devices that can be addressed, the micro controller that communicates power supply related information and control, the EEPROM that contains FRU_ID information, and the PCA9541 multiplexer that controls which i2c line is being communicated to. The GUI automatically sets the correct address depending on which feature is selected by the user.

8.1.1 Initiation of and manual termination of Polling

The tool can be set up to poll any of the 8 modules by clicking on the individual areas to the left of the module number. If all modules are to be pulled the user can alternatively click the **All** button once. To turn OFF polling simply click on

the **None** button. The polling rate can be changed by moving the scrolling bar to the right or left. The chosen polling rate is displayed to the left of the scrolling bar.



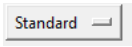
8.1.2 Automatic Poll termination

Polling can be automatically stopped for any or all of the three failure modes, **I2C fault**, **Packet error**, or **Bad Packet** detected by the tool if the appropriate box is selected by clicking on the box area. A bad packet may exist if the number of data bytes requested and sent do not agree.

8.1.3 Changing the rate of Polling

The **polling rate** can be changed by dragging the bar to the right of the displayed polling rate. The selected rate is displayed. The rate between polls can vary from 200ms to 5 seconds.

8.1.4 Choosing what to Poll and which product is being Polled



The **Standard** labeled button above the scrolling bar determines what is being polled. Clicking on the little square to the right of the 'standard' label displays a drop down menu that selects other possible features beside the 'standard' polling feature. The following features are currently available:

Standard: Polls the digital STATUS and ALARM registers and output voltage, output current, and temperature.

Enhanced: Invokes the polling of all analog read backs in addition the 'standard' poll. These include input voltage and power, fan speeds, Controller revisions, and run time. Default numbers are set in registers that are not supported by the module being polled.

CDC2100 and **CAC3000:** These are two custom codes that to tool can support.

CP2500DC: Clicking on this icon changes the GUI display to communicate with the latest CP2500DC module. In this mode only the 'enhanced' mode of read back is supported.

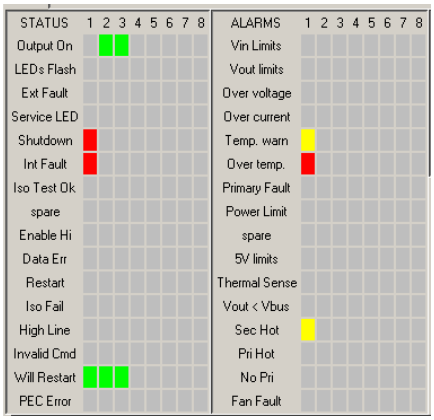
8.2 Returned data Display Area

There are three areas in the tool that display the status and key module parameters as a combination of numerical values and LED-like status displays. These are the **Status & Alarm**, **Read Back**, and **Polling Status** areas.

The LED-like status screen of the **Status & Alarm, Polling Status,** and the analog data displayed in the **Read Back** area display the latest reported state of the monitored modules. These registers are not 'frozen' as specified by the PMBus specification. Instead the registers display the updated state being read back every time the unit is polled.

8.2.1 Status & Alarm

The **Status & Alarm Area** displays status of up to eight power supplies is being presented here. The user selects in the Polling and Stop Control area which power supplies are to be monitored and at what time rate they should be polled. The displayed color assesses event severity. Green – normal, Yellow – warning, Red – fault which requires immediate attention. Since these registers are continually being updated, it is not unusual to see changes in the displayed states as the module goes through a number of event changes prior to settling down to the final state.



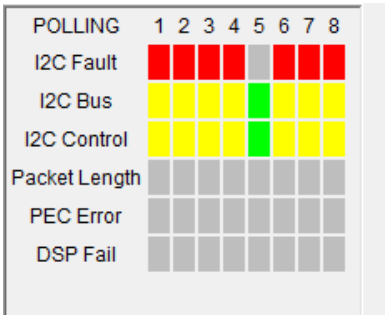
In the display to the left, module #1 has been shut down because of an over-temperature event on the secondary side. The module assesses that this is an internal fault which is the predictably likely cause but not necessarily correct. The thermal condition could have been caused by either an excessive thermal ambient or airflow blockage. Unfortunately, the determination of these two 'external' events is beyond the assessment capability of the module electronics. Thus the assessment of the internal fault has a likelihood of being incorrect. This is a real example of the limitations of fault prediction.

Modules #2 and #3 are working correctly. Their outputs are ON.

The 'will restart' indicator shows that the three modules are in the restart mode.

8.2.2 Polling Status

The **Polling Status area** displays the status of communications of the modules being read or polled.



I2C Fault – RED Communications failed with the I2C µController.

I2C Bus – YELLOW Communications are not established on the bus.
GREEN Module at address #5 Established communications

I2C Control – YELLOW Does not have control of this I2C line.
GREEN Module at address #5 Has control of the connected bus

Packet Length – RED Length of the data packet is incorrect.

PEC error – RED The calculated PEC does not agree with the transmitted PEC.

DSP Fail – RED Internal communications between the DSP and the I2C µC failed

8.3 Read Back

In the lower left corner analog register information is read from the selected modules. The data read back from all 8 possible modules (the maximum capacity of a single bus line) can be displayed at once. There are two modes of read back, **standard** and **extended**.

The **standard** read back records the output voltage, output current, and temperature of the units.

The **extended** read back also records the commanded fan control percentage and the speed of up to three fans in RPM, input voltage and power, firmware revision of the three µCs, and operational running time. The extended read back is supported by the latest revisions of the power supplies.

DATA	Vout	Iout	Temp.	Fans (%control,rpm1,2,3)	AC (volts,power)	Version	Hours
1	-----	-----	-----	---	---	---	---
2	-----	-----	-----	---	---	---	---
3	-----	-----	-----	---	---	---	---
4	-----	-----	-----	---	---	---	---
5	53.99	0.0	29	0% (0.0k,0.0k,0.0k)	117V 17W	FE, 73, 25	105
6	-----	-----	-----	---	---	---	---
7	-----	-----	-----	---	---	---	---
8	-----	-----	-----	---	---	---	---

Vout, Iout, Temp. – standard & extended mode reads

Fans – commanded duty cycle(RPM#1,RPM#2,RPM#3)

AC – input voltage, input power

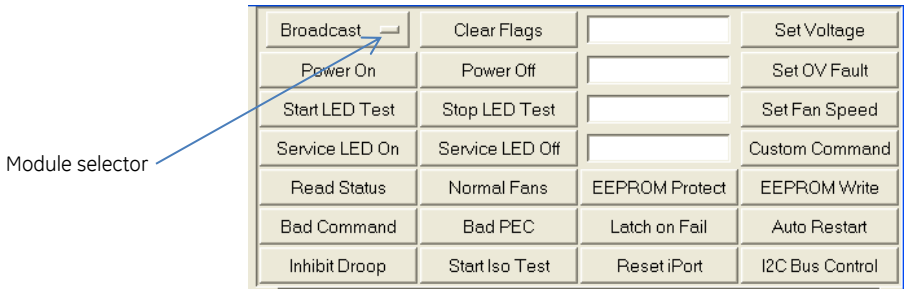
Version –primary µC, secondary DSP,I²C µC in succession

Hours – Cumulated run time hours

The **digital STATUS and ALARM register information** area shows the last read state of all polled modules. The registers are cleared by either clicking on the **Clear Flags** button or by clicking on the **Read Status** button. If the fault or alarm is still persistent after clearing the register flags are reset again. The communication fault bits are an exception. These bits must be cleared using the **Clear Flags command**.

8.4 Commands area

This area of the tool provides access to a number of commands. Each of the commands is summarized below



8.4.1 Broadcast/module:

The user may select between addressing an individual module or all modules simultaneously as indicated by the 'broadcast' label. In the view above, the GUI indicates that 'broadcast' is the present addressing choice. To change to a specific address click on the **module selector** tab that opens a selection menu showing all possible combinations. Select the module to be addressed. The GUI responds by showing the new address in the box, i.e. unit 1 or unit 2 etc. Only a single module, or all modules, can be selected..

8.4.2 Clear_Flags :

The alarm states of the four status and alarm registers can be cleared by clicking on this button. Note that the cleared states of the modules will not be observed on the screen until **Read Status** is clicked to refresh the module status in the tool. An alternate approach is to poll the status of all modules periodically to maintain a continuously refreshed status display. If the fault/warning condition in the module persists the refreshed display will get reinstated into its alarm state to indicate the continuing fault/warning condition.

8.4.3 Power On and Power Off:

The main output of the power supply is commanded to turn ON or OFF via these commands.

8.4.4 Start LED Test and Stop LED Test:

These buttons command the power supply to execute the LED test which starts/stops the blinking of all LEDs simultaneously. The ON time is appreciably longer than the OFF time. The ON time is chosen to enable the user to scan through a large number of power supplies operating in parallel.

8.4.5 Service LED ON and Service LED OFF:

These buttons exercise the Service LED commands.

8.4.6 Read Status:

This button executes a single **read** to the module selected to be commanded to.

8.4.7 Normal Fans:

This button relinquishes control of the power supply fan speed to the power supply controller.

8.4.8 Bad command and Bad PEC:

These commands purposely issue either a bad command or an incorrect PEC so that the user could verify that the power supply properly sets the correct STATUS flags identifying the incorrect instruction. An executed read back should verify the correct behaviour .

8.4.9 Inhibit droop:

This command is only applicable to modules such as the PEM that incorporate droop regulation where the output voltage decreases proportionately to output current. The command tells the module to ignore drooping. There isn't a command to revert back to droop. The only way to reinstate droop is to unplug and reinsert the unit in a multiple unit arrangement. (This is required in order to remove the bias voltage from the control DSP and thus reset it into its default mode).

8.4.10 Start Iso Test:

This command instructs the selected module to commence an isolation test. The isolation test can only be performed when multiple modules are being tested. Prior to initiating this command the user needs to verify that the other modules on-line have sufficient capacity to provide the output power without the unit being tested. If sufficient power is not available the modules on-line will go into a power /current limit mode that may negate the validity of the test.

Note: Since the verification of sufficient power capacity is a contingency for validity of this test, a test failure is provided as 'information only' and does not trigger an automatic power fault. The using system must determine whether the module is in fact faulty.

8.4.11 Set Voltage and Set OV fault:

The output voltage of the module and the OV shutdown set point can be changed with these two buttons. The changes are only temporary. Removal of the bias voltage to the DSP would reset these register values into their defaults.

8.4.12 Set Fan Speed:

This button can be used to increase the speed of the fans beyond what the power supply requires for its internal operation. The entry is in % duty cycle.

8.4.13 Custom Command:

This button enables the exercising of a command that is not listed in this command area. The command needs to be entered in Hex and it should be supported by the module.

8.4.14 EEPROM Protect and EEPROM Write:

These commands instructs the module to either 'enable' or 'disable' the ability to write into the upper ¼ of memory space of the EEPROM. The EEPROM in the CP platform contains 64kbits. Memory locations 0 x 1800 and above are in the protected section of the EEPROM.

8.4.15 Latch on Fail and Auto Restart:

These buttons instruct the module to either 'latch' after a OV, OC, or OT fault or 'restart' after an OV, OC, OT fault. The instruction cannot be commanded to individual functions. These commands do not change the default state of the power supply.

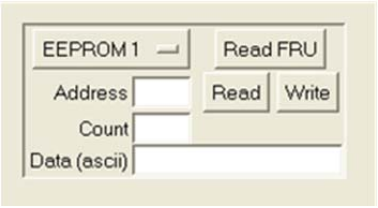
8.4.16 Reset Iport:

This function is no longer supported. It was used prior to the introduction of the **GE Interface Adapter.**

8.4.17 I2C Bus Control:

The power supply has two independent and isolated I²C ports. This button instructs the module to take over control to the I²C port this application is connected to. This feature is further discussed in the next section.

8.5 Communicating with the External EEPROM



This section of the tool enables writing to and reading data from the external EEPROM (Atmel AT64C64 or ST M34D64). Besides storing and reading data the 'write protect' feature protection the upper ¼ of memory can be demonstrated.

The tool can select which of the 8 possible modules is being communicated to. Clicking on the little square to the right of the EEPROM1 labeled button brings up a dropdown menu that can select any of the 8 modules.

8.5.1 Reading content in the EEPROM

There are two ways to extract content from the EEPROM. The first was to extract data is using the **Read_FRU** command that extracts all the FRU_ID information in one command. The second way provides the starting memory location address and telling the gui how many bytes of data are to be read.

8.5.1.1 Read FRU:

This button extracts the FRU_ID information that is stored in the EEPROM. The information is displayed in the Command Log area of the tool. It is also stored in the test record file that has been created automatically by the tool.

8.5.1.2 Read content

The procedure below reads content in any memory location of the EEPROM

8.5.1.2.1 Address:

The starting hex memory location for either the **Read** or **Write** functions needs to be entered next to the address box prior to clicking either of these two functions. Note that a **Read** also needs the count area to be populated.

8.5.1.2.2 Count:

Tells the tool how many bytes should be read back. The entry is a number indicating how many bytes are to be read back. The count number should not exceed 32 bytes and it needs to be populated prior to clicking on the **Read** function.

8.5.1.2.3 Data (ascii):

The read back information is displayed in the **Command Log** box. The Command Log displays both the hex and ascii equivalent of the read back information. This information gets logged into the test record file.

8.5.2 Writing content into the EEPROM:

Writing content into the EEPROM requires sending the starting memory address and the content to be written

8.5.2.1 Address

The starting hex memory location needs to be entered next to the address box

8.5.2.2 Count

Leave this box empty

8.5.2.3 Data - 32 byte content limit

The external EEPROM is partitioned into 32 byte pages. For a write operation only the starting address is required and subsequent data is automatically incremented by the EEPROM hardware. If the 32 byte limit is exceeded the device executes a wrap-around that will start rewriting from the first address specified. Thus byte 33 will replace the first byte written, byte 34 the second byte and so on. One needs to be careful therefore not to exceed the 32 byte page limitation of the device address incrementing feature.

8.5.2.4 Write protect

Make sure that the **write_protect** feature is enabled when executing a write command to the upper ¼ of memory. Without enabling **write_protect** the write command will be ignored by the EEPROM.

8.6 Demonstration of dual I²C communications

The tool can be used to demonstrate the dual I²C communications capability of the power supplies. The Interface board connected to the GE shelf brings out both I²C lines of the power supply. There are two ways to demonstrate the capability.

8.6.1 Using a single adapter and laptop

The first way is to connect the GE Interface Adapter into I2C-0 and set up the tool to communicate to the power supply. After the establishment of communications (note the communications information displayed by the LEDs in the polling section of the display area) , disconnect the ribbon cable from I²C-0 and insert it into I²C-1. Observe that the polling section of the display area will indicate that the I2C-1 side does not have control if the power supplies are polled. Issue any operational commands and observe that the power supplies do not respond to the commands. Next, click on the **I2C Bus Control** button in the commands area to take over control. Note that LED indicator

changes showing that the I²C-1 side now has control. Issue any operational commands and observe that the power supplies respond to the instructions. This method is not the best alternative because of the switching spikes that are observed when the Interface Adapter is removed and inserted into the I²C ports. Albeit, in our development labs, numerous switching between the ports did not cause any glitches or damage.

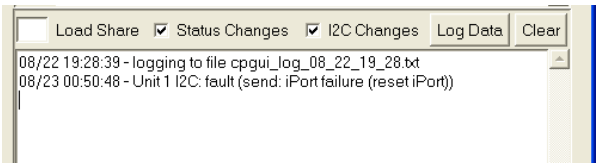
8.6.2 Using two adapters and two laptops

The second, and preferred way, of demonstrating dual communications capability is to connect two GE Interface Adapters, one into I²C-0 and the other into I²C-1, each adapter connected to a separate computer. Start the CPGUI in both computers and observe that the computer connected into I²C-0 has control. The computer connected to I²C-1 should indicate that I²C-1 has no control. Control can be taken over manually by either computer by clicking on the **I2C Bus Control** button. Alternatively, drag the cursor below the **Automatic I2C Bus Control** labeled area in the middle of the tool to set the time delay between automated bus takeover on both computers. The time delay can be as long as 30 minutes. Set different times so that you can recognize which side is being used. (a good suggestion is 1 minute and 2 minutes). The set time delay is displayed instead of the **None** that is initially configured. . Set both computers to the poll mode, and observe that control is automatically being taken over by the two applications.

8.7 Command/Data log

Another feature of the CP GUI is its automated time stamp driven capture of all changes in the STATUS and ALARM registers, including the execution of all instructions.

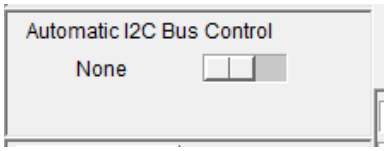
Four different types of events can be time stamp recorded and these are;



Load share – an entry in this table compares the difference in current readings between the modules. If the difference is greater than the value entered, this data will be recorded.

Status changes – records changes in STATUS and ALARM registers

I2C changes – records initiated commands and communications errors



8.7.1 Log Data:

Records the analog data captured on the screen. This is a one- time stamped event. The time stamp captures the date, time of the screen capture, and the unit # transmitting the information. If the Standard read is selected then three parameters, output voltage, output current and temperature are captured. If the Enhanced read is selected, then all accessible parameters, such as fan speed control, fan speed, input voltage, input power, firmware revision numbers and accumulated running time are also captured. In either mode the date, time, and unit # are also recorded.

Standard read capture

06/27 13:41:13 - Unit 5 Vout=53.99 Iout=0.0 Temp.=34

Enhanced read capture

06/27 13:46:44 - Unit 5 Vout=53.99 Iout=0.0 Temp.=35 Fans (%control,rpm1,2,3)=0% (0.0k,0.0k,0.0k) AC (volts,power)=119V 17W Version=FE, 73, 25 Hours=35

8.7.2 Clear:

Clicking this button clears the Log screen, however, it does not erase the data that has already been captured in the Log file.

8.8 Log File

The Log File is recorded automatically in the same directory where the executable .exe file resides. If the downloaded executable is located on the desktop than the Log file is also automatically placed on the desktop. It is named automatically by date i.e. 6_27 and time(hours_minutes) recorded 13_25 in military time (1:25PM),

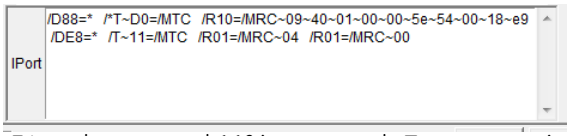
The log file can be renamed and located into any directory just like any other file. Or, if not needed, simply delete it.

8.9 Bus Traffic

The latest bus traffic transmitted over the I²C bus is shown above the Command Log. This detailed information of bus activity is not trivial to decode. The commands are a conglomerate of the Interface Adapter command set, the I²C communications protocol of the CPL platform and the command set of the PCA9541 multiplexer. This data is normally for information only. The contents have already been translated and displayed by the GUI into the various sections.

Never the less, the information presented may be useful for programmers who are attempting to communicate to the power supply but have trouble doing so. The data here shows what is being transmitted over the bus, albeit, commands between the adapter and the computer overshadow some of the information.

Translation of the recorded single poll shown below is as follows: µC of the module at address 44[1] is requested to read status [D0], 10 bytes of data are received, the first byte tells the host that 9 data bytes follow, and these are: STATUS – 2 bytes, ALARM – 2 bytes, output voltage – 2 bytes, output current – 1 byte, temperature – 1 byte, followed by the PEC – 1 byte. (The last data byte of each data packet is always the computed PEC). When this instruction is completed the GUI addresses the multiplexer of the module, 0xE8, at device address 74 and command 11[2] is executed. Two successive reads follow obtaining control and interrupt status.



[1] Unit #1 in the 2nd shelf; device 0x88 has the bit pattern 10001000, however, since the LSB is the write/read bit, the actual address is 1000100 which is 0x44. The last three bits are A2, A1, and A0. In the next section **Starting the CPGUI** we explain how we created this address
[2] Command hex byte 0x11 reads the control register, increments by one and reads the interrupt status register of the multiplexer

9 DPI-CLI (Command Line Interface)

The DPI-CLI is a low-level command-line based tool that is useful for decoding and analyzing basic communications with the module. The low level capability facilitates debugging when specific command interactions need to be examined in detail. The DPI-CLI also has a powerful scripting and data capture/logging capability that supports diagnostics-type debugging of a system with one or more modules.

9.1 Starting the DPI-CLI Tool

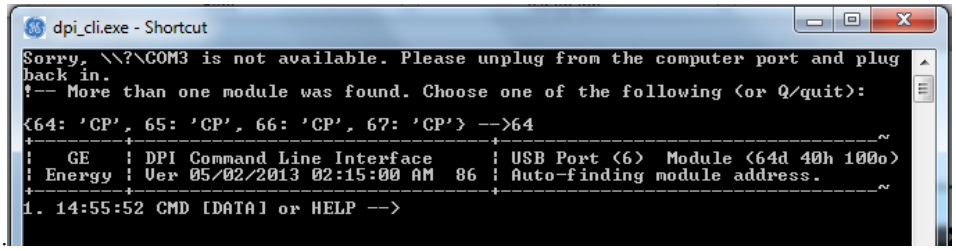
Unlike the CP_GUI, the CLI requires connectivity to a powered up set of modules prior to start up. CLI automatically finds all modules with valid addresses on the bus. These modules may include POLs, bus converters, CAR front ends or CP modules. When multiple devices are found, the tool will indicate that and ask the user to select the device to communicate with (in the CLI, only one device can be addressed at a time).

Make sure that the modules are powered up and that all connections are properly made. CP is designed to issue an SMBAlert signal until the module is addressed by the host controller. The Interface Adapter should indicate that the Alert has been set by the continuous blinking of its LED in Amber color. This is a clear indication from the adapter that communications with the power supply has been established.

The DPI-CLI tool can now be started by double clicking on the dpi_cli icon placed by the installation program on the desktop. The tool can also be started by directly executing the executable file from its stored location.

In the example below the tool starts up stating that COM3 is not available. We believe that this error occurs on occasion if the port was not disconnected cleanly from the computer the last time the Adapter was used. The computer indicates that the Adapter is not available on COM3 but then it looks for other ports to see if it can find the Adapter. Once the adapter is found, the program starts. The tool identifies four modules, at the decimal addresses indicated by the tool. The tool also indicates that all four modules are CP power supplies. The tool then asks to choose which CP module should be chosen.

Once the module address is selected, the CLI starts by displaying some basic information including which address is being used, followed by the command prompt. The tool is now ready to communicate with the selected module.



9.2 Standard instruction

The instruction normally contains a function such as 'write' or 'read' followed by the command to be executed and then a data field, if required, each separated by a space.

FUNCTION <command> <data>

Below is a list of supported functions and their brief description:

- H or HELP – provides help within the program with a detailed list of functions, brief descriptions and examples of usage
- A or ALERT – reads the status of the SMBAlert Line
- D or DELAY – allows a specified delay to be inserted before the next function is executed
- G or GROUP – allows for the execution of multiple read commands in a comma separated list. Works only with text commands (cannot use the hex value of the command)
- I or INPUT – used to load a text file (.txt) containing a series of commands to be executed by DPI-CLI
- K – for displaying or changing the I^2C clock rate between 100kHz (default) and 400kHz in the USB Interface Adapter
- L or LIVE – allows continuous, repetitive running of one or more commands
- M or MODULE – changes the module being addressed
- N or NOTE – supports insertion of notes for logging
- O or OUTPUT – allows for saving all results to a file as well as the screen
- P – for displaying or changing pull-up resistor values in the USB Interface Adapter
- Q or QUIT – exit the DPI-CLI tool
- R or READ – allows data to be extracted from the power module
- REGINFO – tabulates the supported functions and commands of DPI-CLI
- S or STOP – supports stopping the saving of results to a file
- SHOWALL – displays a summary of all commands within DPI-CLI
- SUPPRESS_X – used to abbreviate the display returned by the tool, X is either Y (for yes}, or N (for No)
- V – shows version of firmware in the USB Interface Adapter
- W or WRITE – allows data to be written to the module

All functions are case-insensitive, i.e. they can be entered as lower case, upper case or a mix of the two. Also note that except for the REGINFO and SHOWALL functions, it is not necessary to type the entire function name, just the first letter is sufficient.

REGINFO provides additional information such as the hex command byte corresponding to each command and the default value of each command.

Below are two examples of standard instructions. The first changes the value of the output voltage of the power supply and the second turns OFF the output of the power supply. (note: A table showing all supported commands and their hex register equivalents is summarized further down in this document)

>write vout_command 50

>W 21 50

Both commands execute the same instruction. The 'write' function can be abbreviated and entered in upper case, if desired.

>write OPERATION on:n

>W 01 on:n

Some commands, as seen above, follow with a verbalized data field. These data fields may include multiple instructions, each separated by a ',' . However, the CP platform command set does not support commands with multiple data field instructions.

Detailed instructions and examples for each of the functions follow.

9.2.1 H or HELP Function

This function can be used in two forms, with and without an argument. If no argument is used, the function returns a screen with the following:

- Examples of how the DPI-CLI tool can be invoked and the purpose of each command line argument
- Description of how module addresses can be specified (in hex, octal or decimal format)
- Examples of how to use functions
- How to exit the tool

The HELP function can also be used with an argument, where the argument is the specific function on which help is needed. For example,

```
>HELP ON_OFF_CONFIG or
>H 02
```

results in the DPI-CLI tool displaying help information specific to the ON_OFF_CONFIG PMBus command which also has the hex value 02.

9.2.2 A or ALERT Function

The ALERT or A function will read the SMBALERT line status. If the line is asserted it will return the value 'asserted' and if the line is not asserted the value 'normal' will be returned.

Example:

```
>ALERT
```

'normal' or 'asserted' follows the command indicating the status of the SMBAlert line

9.2.3 D or Delay Function

This function is used to insert a specified delay in seconds and is commonly used in a list with multiple functions. If the Delay function is used without any specified delay, the tool will pause until the user presses the <ENTER> key to continue.

Example:

```
>DELAY 1.2
```

results in a 1.2 second delay, before the next function is executed.

9.2.4 G or GROUP Function

This function allows multiple commands to be placed in a comma separated list for execution. Note that there should be no spaces between multiple commands.
Example:

```
>GROUP VOUT,VIN,IOUT
```

reads the output voltage, input voltage and output current and return all three values.

Note: The CP command set does not use this capability.

9.2.5 I or INPUT Function

This function is used to load a text file containing one or more functions. Only one function may appear in each line of the text file. This function supports a scripting capability where a sequence of multiple functions with numerous commands can be loaded and run using the DPI-CLI tool.

Example:

>INPUT test_sequence.txt

reads in the file test_sequence.txt and execute functions contained within the file.

9.2.6 K - Clock Setting Function

This function is used to read or change the I^2C clock setting in the USB Interface Adapter. The default value is 100kHz. The desired clock rate can be specified as 100 or 100kHz or 400 or 400kHz. If this function is used without an argument, the current I2C clock rate in the USB Interface adapter is returned.

Examples:

>K 400Khz (sets the USB Interface Adapter Clock speed to 400kHz)

>K 100 (sets the USB Interface Adapter Clock speed to 100kHz)

>K (reads the USB Interface Adapter Clock speed setting)

9.2.7 L or LIVE Function

This function supports the continuous and repetitive running of one or more commands. The delay between the execution of a set of commands can also be specified in seconds, as well as the number of repetitions desired.

Example:

>L vin,vout 2.5 30

reads Vin and Vout from the UUT 30 times with a 2.5 second delay between each set of read operations (read Vin and Vout). Note that there should be no spaces between the entered commands (e.g. between vin and vout in the example above). but a space must be entered between the delay and the number of repetitions. If the number of repititions is not specified, the comparison continues indefinitely until it is stopped by using the DOS command (<CNTL>C).

9.2.8 M or MODULE Function

This command allows switching to either a different device address or transitioning to the Global address 0 x 00, if supported, to issue simultaneous commands to all devices on the bus. Changing the output voltage of a system of paralleled power supplies needs to be accomplished simultaneously. Switching to the Global address accomplishes that simultaneous command transmission. The default address is assumed to be decimal, but hex or octal addresses are also supported if the letters 'h' or 'o' are placed before or after the address numbers.

Examples:

>module 25

>module 19h

>module 31o

are all equivalent and change the module addressed to 25 (decimal), 19 (hex) or 31 (octal).

>m 0

Changes the communicating address to the Global 0 x 00. Note that the CLI warns that the 0 x 00 address is in a special category and should not be used as a 'device' address.

9.2.9 N or NOTE Function

This command allows a note to be put out by the tool – primarily useful for logging purposes. If spaces are to be included, enclose the enter text string within single quotes.

Example:

>N 'this is a test'

places the text string "this is a text" in the output from the tool.

9.2.10 O or OUTPUT Function

This command supports saving results to a specified file. The file using the specified name is saved in the same directory where the executable DPI-CLI tool resides. The STOP command terminates saving results into the specified file. So the feature can be used to capture everything from when OUTPUT is enabled to when STOP is entered.

Example:

>OUTPUT testing_results.txt

places results from the DPI-CLI tool into the file named testing_results.txt located in the directory from where the DPI-CLI tool is run.

9.2.11 P Function

This command is used to set the clock and data line pull-up resistors in the USB Interface Adapter. Possible options are 3.3mA (default), 0.9mA, 0.44mA, or 0 (none). If no argument is specified, the current value is returned.

Examples:

```
>P 0.9

>P 3.3mA

>P   [This command returns the current sourced current value. ]
```

9.2.12 Q or QUIT Function

This function is used to exit the DPI-CLI tool.

9.2.13 R or READ Function

This function can be the PMBus command entered as a command string or a command byte in hex format.

Examples:

```
>READ READ_STATUS
>R D0
```

The CLI tool responds with a detailed response, an example of which is shown below with explanations.

Digital bit representation of the data read back – high byte first

```
{ 'BINARY': '00011110000000000101010001011110000000000000000000000010100000000001001',
  'CMDBYTE': 'D0',
  'CMDNAME': 'READ_STATUS',
  'DATA': { '1 STATUS2': '01000000(WILL RESTART)',
            '2 STATUS1': '00000001(OUTPUT ON)',
            '3 ALARM2': '00000000()',
            '4 ALARM1': '00000000()',
            '5 VOLTAGE': '53.99',
            '6 CURRENT': 0,
            '7 TEMP': 30},
  'ERRORS': [],
  'MODADDR': '68d',
  'RAW': '/O=/OCC\r  /D44=*  /*T~D0=/MTC\r  /R0A=/MRC~09~40~01~00~00~5e~54~00~1e~fb\r  /C=/CCC\r  '}
```

The command: hex byte representation and its meaning

Shows the four digital registers, their bit pattern, and the meaning of bits with a '1' state. In this example the STATUS2 register returned a '1' in bit 6 which indicates 'will restart' and the STATUS1 register returned a '1' in bit 0 which indicates 'output ON'.

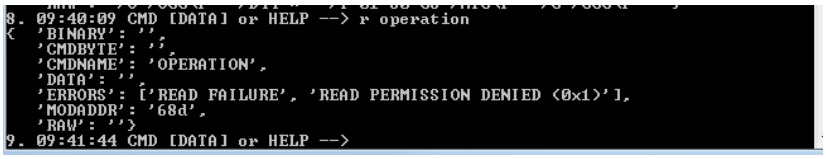
Three readings are returned in the 'standard format'; output voltage – 53.99, output current – 0 , and internal temperature - 30

No transmission errors were recorded

The device address in decimal format

RAW shows the transmitted communication. Interpretation is as follows: /O – open , /OCC – open command complete, /D44 – device address in hex, /*T~ command is followed by a restart, do not issue a STOP followed by command D0 , /MTC –master transmit complete, /R0A – read back ten bytes, /MRC – master receive complete, followed by the ten data bytes. (~ separates each bytes). The first byte '09' indicates how many data bytes are following. The last byte 'fb' is the calculated PEC number of the transmitted data. The last instruction is /C – close followed by /CCC – close command completed by the translator.

Note: The CP command set does not support the reading of 'writeable' commands. The OPERATION command is writeable, but simply reading the state of the OPERATION command is not supported. See example below,



9.2.14 REGINFO Function

This command is used with no arguments and provides a compact listing of all supported PMBus commands.

Example:

```
>REGINFO
```

9.2.15 S or STOP Function

This command is used along with the OUTPUT command to stop saving results from DPI-CLI to a file.

Example:

```
>STOP
```

9.2.16 SHOWALL Function

This function displays all commands with initial default values, the default bit string and the interpreted meaning of the command. An example of the command with a partial view of the response is shown here.

```
05/18/2010 12:38:54 --- R/W/C/H CMD [WDATA] --> showall
| HX | PARAMETER         | CURRENT BINARY    | INTERPRETED RESULT
| 01 | OPERATION         | 00000000          | {'On': 'N', 'Margin': 'Off'}
| 02 | ON_OFF_CONFIG     | 00010111          | {'CPA': 'Y', 'CPR': 'Y', 'CMD': 'N',
'POL': 'Y', 'PU': 'Y'}
| 10 | WRITE_PROTECT     | 00000000          | {'ALLOW': 'ALL'}
| 20 | VOUT_MODE         | 00010110          | {'VOUTEXP': -10, 'VOUTMODE': 0}
| 22 | VOUT_TRIM         | 0000000000000000  | 0.0
```

9.2.17 SUPPRESS_Y or SUPPRESS_N Function

Instructs DPI-CLI to suppress or turn back display of some of the message content in order to reduce the amount of information being displayed and saved. The SUPPRESS_Y function will suppress part of the displayed content while SUPPRESS_N will turn back ON the full display.

Example:

For example the instruction to read STATUS will display the following:

{ 'BINARY': '00011110000000000101010001011110000000000000000000000000101000000000001001',
  'CMDBYTE': 'D0',
  'CMDNAME': 'READ_STATUS',
  'DATA': { '1 STATUS2': '01000000(WILL RESTART)',
          '2 STATUS1': '00000001(OUTPUT ON)',
          '3 ALARM2': '00000000()',
          '4 ALARM1': '00000000()',
          '5 VOLTAGE': '53.99',
          '6 CURRENT': 0,
          '7 TEMP': 30},
  'ERRORS': [],
  'MODADDR': '68d',
  'RAW': '/O=/OCC\r /D44=* /*T~D0=/MTC\r /R0A=/MRC~09~40~01~00~00~5e~54~00~1e~fb\r /C=/CCC\r '}

When the SUPPRESS_Y function has been exercised,

>SUPPRESS_Y

the same instruction will display as follows:

READBACK: STATUS|
            {'1 STATUS2': '01000000(WILL RESTART)',
             '2 STATUS1': '00000001(OUTPUT ON)',
             '3 ALARM2': '00000000()',
             '4 ALARM1': '00000000()',
             '5 VOLTAGE': '53.99',
             '6 CURRENT': 0,
             '7 TEMP': 35}

The user can invoke this command set multiple times if desired during a scripting program.

9.2.18 V or Version Function

This function reads back the software (firmware) revision of the USB Interface Adapter.

Example:

>V

returns the firmware revision of the software in the USB Interface Adapter.

9.2.19 W or WRITE Function

The WRITE function can have multiple syntaxes. The basic syntax consists of a single argument and data value, as follows:

WRITE <argument> <value>

For example,

`>WRITE vout_command 50` commands the module to set the Output Voltage to 50Vdc.

The WRITE function can also execute successive arguments in a single command line. Note that this successive arguments function is not used in the CP platform.

9.3 Summary of Supported PMBus Commands for the CP Platform

Command	Code	Capability	Name	Value	Function
OPERATION	01h	w	On	Y	Turns Module ON
Example: `> w 01 On:Y`				N	Turns Module OFF
CLEAR_FAULTS **Example:** `> w 03`	03h	w			Clears the Status registers and resets the SMBAlert# signal.
VOUT_COMMAND **Example:** > w 21 52 > write vout_... 52	21h	w		2 Bytes	Changes the main output voltage . The command example instructs the module output to be set to 52Vdc,
VOUT_OV_FAULT_LIMIT **Example:** `> w 40 56`	40h	w		2 Bytes	Changes the overvoltage shutdown level of the main output voltage. The example changes to OV shutdown level to 56Vdc.
READ_STATUS **Example:** > r D0 > read read_status	D0h	r		10 bytes	Returns the state of the power supply (STATUS and ALARM registers, output voltage, output current, and internal temperature levels).
FLASH_LEDS_ON **Example:** > w D2 > write flash_LEDs...	D2h	w		none	ALL LED test, 0.7sec ON, 0.2sec OFF
FLASH_LEDS_OFF **Example:** > w D3 > write flash_LEDs...	D3h	w		none	Turn OFF LED testing
SERVICE_LED_ON **Example:** > w D4 > write service_...	D4h	w		none	Turn ON service LED, 0.5s ON and 0.5sec OFF
SERVICE_LED_OFF **Example:** > w D5 > write service_...	D5h	w		none	Turn OFF service LED
EEPROM_WRITE_ON **Example:** > w D6 > write EEPROM_.....	D6h	w		none	Enables writing into the upper ¼ of EEPROM
EEPROM_WRITE_OFF **Example:** > w D7 > write EEPROM_...	D7h	w		none	Disables writing into the upper ¼ of EEPROM
INHIBIT_RESTART **Example:** > w D8 > write inhibit_...	D8h	w		none	Sets fault levels into latched mode.

Command	Code	Capability	Name	Value	Function
AUTO_RESTART **Example:** > w D2 > write auto_...	D9h	w		none	Sets fault levels into restart mode.
ISOLATION TEST **Example:** > w DA > write isolation...	DAh	w		none	Tests the Or'ing function in paralleled modules
INHIBIT_VOUT_DROOP **Example:** > w D2 > write auto_...	DBh	w		none	Turns OFF the 'droop' feature, if supported
READ_INPUT_STRING **Example:** > r D2 > read auto_...	DCh	r		3 Bytes	Reads input voltage and input power
READ_FIRMWARE_REV **Example:** > r DD > read read_...	DDh	r		3 Bytes	Reads firmware revision of the 3 processors
READ_RUN_TIMER **Example:** > r DE > read read_run_...	DEh	r		3 Bytes	Reads the accumulated ON time of the PS
FAN_HI_SPEED **Example:** > w DF 75 > write fan_HI_... 75	DFh	w		1 Byte	Increases speed to the specified % of maximum speed capability of the fan. The example sets the duty cycle of the fans to 75%.
FAN_NORMAL_SPEED **Example:** > w E0 > write fan_...	E0h	w		none	Reverts fan speed back to power supply control
READ_FAN_SPEED **Example:** > r E1 > read read_...	E1h	r		4 Bytes	Returns % control, fan1, fan2, fan3 speed in RPM

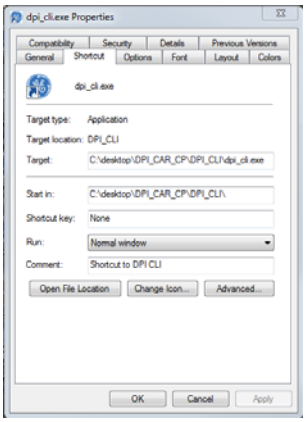
9.4 Scripting support

One of the more powerful tools of CLI is the ability to sequentially execute a set of commands that were created in a text file and record in the output file created by the tool the test results of these commands. Supporting the scripting feature is the data comparison capability described in the next section. Data comparison allows the tool to validate the value of the digital word or the analog value obtained from the module under test. The output file created with the data comparison capability is a complete test record of the UUT. Review of this record is simplified by the error reporting feature of the comparison tool which reports between asterisks unexpected or incorrect behavior. Simply looking for the asterisk is sufficient to detect all errors in the test report that could become quite lengthy.

Scripting is being used to test the capability of the firmware within the modules. Appendix A demonstrates how scripting is used to test the high voltage shutdown capability of the CP power supplies.

9.4.2 Location of the script file

The script file must be located in the same directory as the CLI executable. To find the location of the executable, point the mouse to the shortcut  executable and right click the mouse. This action will bring up another menu screen that displays a number of selectable options. Click on the last item; 'properties'. Clicking on 'properties' brings up the screen shown below.. The **start in** location, as seen below, shows the actual location of the executable file. The scripting file must be located either in this directory or it can be located in a directory that gets placed (or copied) onto the desktop.



The entire directory must be copied, not only the executable because there are numerous files in this directory that are linked together. Note: this directory can be relocated to the desktop as described in the previous section **Changing the location of files Created by DPI**

9.5 Data comparison capability

CLI has enhanced features supporting testing operations that indicate whether the data returned is within specified limits (for analog variables) or matches expected settings (for digital data). [Note that in the examples that follow SUPPRESS_Y is commanded to limit the display to data returned.]

9.5.1 Analog read back

For reading back analog variables the READ command enhanced functionality can be invoked by using the following format:

READ [command] [parameter] (nominal,%high,%low),

9.5.2 Digital read back

For reading back digital comparisons the READ command enhanced functionality can be invoked using the following format:

READ [command] [parameter] (byte)

Only one comparison can be invoked at any one time,

9.5.3 Multi-parameter read back

The CP protocol uses batch read back in order to improve the transmission efficiency of the required information across the bus. The batches are partitioned into various operational segments. Below, for example, is the read back of the READ_STATUS command which returns the digital states of the status and alarm registers, and the analog reading of output voltage, output current, and temperature of the unit. Suppression has been invoked.

```
READBACK: STATUS|
        {'1 STATUS2': '01000000(WILL RESTART)',
         '2 STATUS1': '00000001(OUTPUT ON)',
         '3 ALARM2': '00000000()',
         '4 ALARM1': '00000000()',
         '5 VOLTAGE': '53.99',
         '6 CURRENT': 0,
         '7 TEMP': 35}
```

9.5.4 Multi-parameter read back comparison

Note that each parameter read back in the READ_STATUS command example above is identified within quotes [i.e. '1 STATUS2']. In the subsequent example below the read_status multi-level read is compared to expected values. Since only a single comparison can be invoked at any one time, the multi-level read instruction must be executed in parts in order to compare each data separately. Note in the example below the difference between digital comparisons and analog comparisons. Each comparison is a separate command.

```
r d0 '1 status2' (40)    compares the status2 register to 0x40
r d0 '2 status1' (01)    compares the status1 register to 0x01
r d0 '3 alarm2' (00)     compares the alarm2 register to 0x00
r d0 '4 alarm1' (00)     compares the alarm1 register to 0x00
r d0 '5 voltage' (54,3,3)   compares the voltage read to 54V ±3% tolerance.
r d0 '6 current' (0.3,50,100)   compares the current read to 0.3A +50% and -100% tolerance.
r d0 '7 temp' (38,10,10)  compares the temperature read to 38V ±10% tolerance.
```

If a digital comparison fails, the program informs the user what the expected value was, what value came back and lists the functions that have been set.

In the example below the READ_STATUS read back sends back 9 bytes of data. The first byte returned is the digital value of the status2 register. The content of this register is examined and is being compared to 0x10. As can be seen below the actual register value is 0xC0. The difference in the comparison is noted and the actual bits that are set to 1 are also reported by the tool in parenthesis.

```
3. 14:16:59 CMD [DATA] or HELP --> r d0 '1 status2' (10)
**  EXPECTED :  0x10 - RETURNED : 0xc0  **
READBACK: STATUS - 1 STATUS2!11000000(PEC ERROR, WILL RESTART)  ]
```

If an analog comparison fails, the program returns to the user the reading within '*'

In the example below the output of the power supply is 54Vdc, but the comparison is being made on purpose to 50Vdc ±3%, in order to demonstrate the incorrect data reporting of the tool

```
4. 14:17:51 CMD [DATA] or HELP --> r d0 '5 voltage' (50,3,3)
READBACK: STATUS - 5 VOLTAGE!* 53.98 *                          ]
```

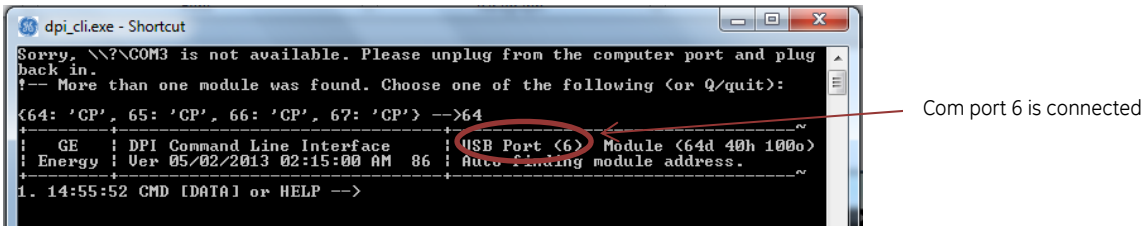
9.5.5 Implementation of a scripting routine

Appendix A shows the scripting test routine implemented for overvoltage shutdown.

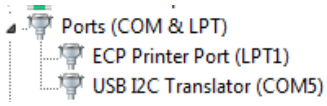
10 CLI issues

10.1 Communications port reporting error

The tool starts up stating that COMx is not available. This announcement appears on occasion. It is reported because even though the USB Adapter is plugged into a specific com.port, the computer incorrectly remembers that the adaptor was plugged into another com.port, in this case COMx in this computer some time in the past. In other words, the computer does not recognize that the Adapter is no longer connected to a specific port. To verify and correct this condition in Windows 7, right click on the Computer icon on the desktop, click on 'properties', expand the Ports (COM &LPT) section. In this section the USB I2C Translator is shown connected to two communications ports, even though in actuality only one of the ports is being used. Note in the example below that com port 3 is shown as not available and the Translator appears on com port 6. Delete com port 3 that is superfluous in order to ensure that you will no longer receive this error. (Make sure that you verified which port is not used. Do not disconnect in error the port that is actually being used.)



When properly identified the ports section should list only a single com port for the Translator.



11 Using the 1u_Interface_Board

11.1 Concept

The CPL 1u_Interface_Board is designed to interface to the **J85480** type GE CPL shelves and to the **GE USB Interface Adapter** that converts I²C signals into USB and vice versa.

On one end of the board a 40-pin ribbon cable terminates the interface board the CPL shelf. On the other end the interface board provides inputs to two I²C lines. The **GE USB Interface Adapter** connects to either of these two connectors. Note on the picture the orientation of the ribbon cable. This side is not keyed and so care needs to be taken to ensure that the connection is correct. The key is the location of the red line identifier.

The board provides two 8-position dip switches for controlling the analog functions of the CPL system and a number of LEDs that display drive signals from the CPL system.

Power for the Interface_Board is derived from the +5V output of the power supply. The board is operational as soon as one of the power supplies receives AC power.

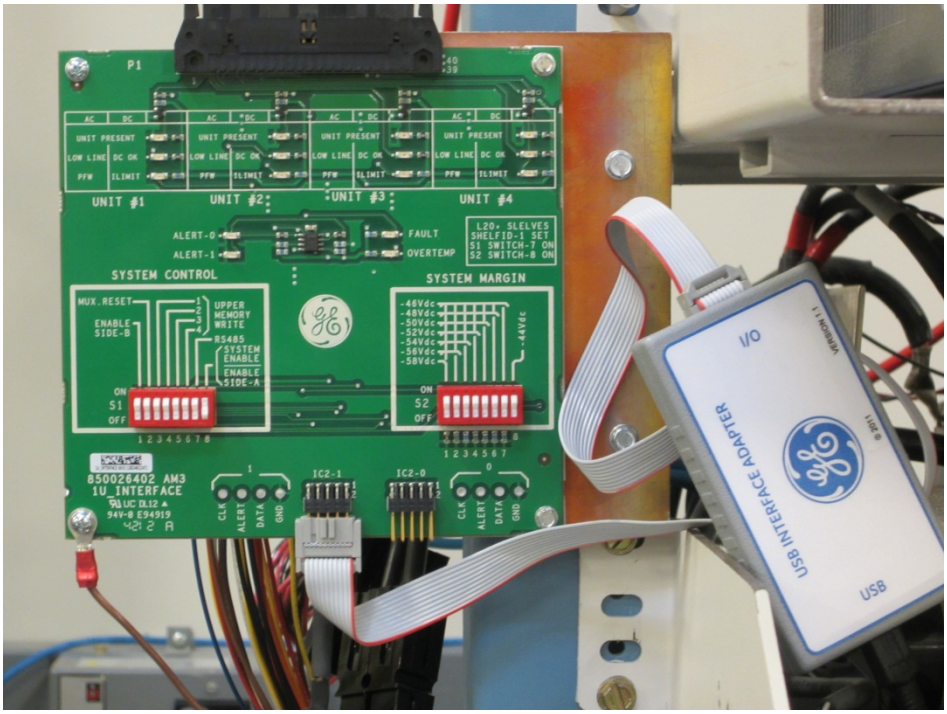


Figure 1: The latest CPL Interface Board and USB Interface Adapter

Note that in the above picture the adapter is connected to i2c-1 and not i2c-0. To communicate from this position the power supply needs to be switched to side 1 because at default side 1 does not have control. Side i2c-0 is a better position to start from because the multiplexer does not have to be exercised prior to the start of communications.

11.2 LED Annunciation

The top of the board shows four boxes, each one representing power supplies in the shelf under test. The three LEDs in each box are;

- **UNIT PRESENT** – This indicator is ON when a module is physically plugged in and at least one of the power supplies has input power (thus generating 5Vdc for the interface board).
- **LOW LINE/DC OK** – In rectifiers this indicator is ON when Input voltage is below 135Vac. In PEMs this indicator shows that the output voltage is present and normal.
- **PWR FAIL WARNING/Ilimit** – This indicator shows **PFW** in rectifiers and in **Current Limit** in PEMs

Below the four sets of LED indicators there are two sets of LEDs, and these are;

- **FAULT** – This indicator is ON when one or more of the power supplies issues a fault warning.
- **OVERTEMP** – This indicator is ON when one or more of the power supplies gets into an over-temperature warning or shutdown condition.
- **INTERRUPT-0 / INTERRUPT – 1** – These LEDs are functional with I^2C communications. The indicators are ON whenever the power supply requests service from the host controller.

11.3 Dip Switch Operation

There are two independent dip switches, one for **System Control** and the other for **System margin.**

The **System Margin** dip switch manually changes the output voltage of the power supply from its 54Vdc factory setting. Switch 1 ON margins the power supply output to 58Vdc. Switches 1 and 2 ON margin the power supply output to 56Vdc and so on. Switch 8 margins the power supply output to 44Vdc whenever it is ON, independent of the position of the other switches.

The **System Control** dip switch has the following functions;

- **MUX.RESET** – The CPL platform offers two independent, redundant, i_2C busses labeled i2C-0 and i2C-1. Selection of which bus the power supply will communicate with is accomplished by a Philips PCA9541 2-to-1 Master Selector. The PCA9541 is configured such that i2C-0 is active on power up. Changing the communications port is done by firmware. The PCA9541 also offers a RESET pin. In case of bus hang-up for any reason, RESET reconfigures the device into its initial state. In addition, the device issues a set of reset commands to the downstream devices as well. For additional information please consult the application notes and data sheet for the PCA9541.
- **UPPER MEMORY WRITE 1 – 4 –** Each of the four power supplies in the shelf contains an internal EEPROM. Turning these switches ON enables a *write* instruction into the upper ¼ of memory. Although this feature is available, it should be used with caution because a re-write will erase factory provided information. For further information see the *PMBus Compliant Digital Interface* technical requirements document.
- **RS485** – When ON, this setting tells the power supply to revert into RS485 mode.
- **ENABLE side A/ENABLE side B –** This switch turns ON output power. ENABLE side A operates all power supplies in a single output shelf. Sides A and B are used in split shelf applications. Side A turns ON the two left-most power supplies. Side B turns ON the two right-most power supplies.

11.5 Components of the demonstration tool

Component	Description	Part number
1U Interface Board (IB)	Exercises features through the J85480 shelves	150027074
Interface cable	Between the IB and the J85480 L20 through L30 shelves	CC848848960
GE Interface Adapter	PMBus™ to USB protocol converter	CC109155636
USB cable assembly	Commercial part Molex 88732-8602	CC408649511
10position ribbon cable	Between the IB and Interface Adapter commercial part Molex 1110620228	CC408650477

11.6 Two shelf operation

Up to two shelves can be paralleled and communicated to. Although the interface board can be used to communicate to the two shelves, its display of operational status only shows the modules plugged into the first shelf.

12 Attachment A: a scripting example

Below is a scripting example of testing the high voltage shutdown level of the power supply.

```
n 'CP rectifier i2c testing'
n 'Vout control and OV protection - rev1.0'
n ' '
n '#1'
n ' '
n 'suppress printout'
n ' '
suppress_y
n ' '
n 'recover power supply from a possible latched shutdown'
n ' '
w operation on:n
d 2
w operation on:y
d 5
n ' '
n 'verify that alert has retired to normal'
n ' '
w vout_ov_fault_limit 60
d 2
n 'Alert state'
a (normal)
n ' '
n 'check revision and time in service'
n ' '
r read_firmware_rev
r read_run_timer
n ' '
r read_fan_speed
n ' '
n 'verify normal start-up'
n ' '
r read_status '1 status2' (40)
r read_status '2 status1' (01)
r read_status '3 alarm2' (00)
r read_status '4 alarm1' (00)
r read_status '5 voltage' (54,1,1)
r read_status '6 current' (0.3,50,100)
r read_status '7 temp' (24,60,20)
d 1
n 'Alert state'
a (normal)
n ' '
r read_input_string 'power' (17,20,20)
r read_input_string 'voltage' (120,5,5)
n ' '
```

```
n 'verify that the output voltage can be changed within limits'
n ' '
w vout_command 42
d 5
n ' '
r read_status '1 status2' (40)
r read_status '2 status1' (01)
r read_status '3 alarm2' (00)
r read_status '4 alarm1' (00)
r read_status '5 voltage' (42,1,1)
r read_status '6 current' (0.3,50,100)
r read_status '7 temp' (24,60,20)
d 1
n 'Alert state'
a (normal)
n ' '
w vout_command 48
d 4
n ' '
r read_status '1 status2' (40)
r read_status '2 status1' (01)
r read_status '3 alarm2' (00)
r read_status '4 alarm1' (00)
r read_status '5 voltage' (48,1,1)
r read_status '6 current' (0.3,50,100)
r read_status '7 temp' (24,60,20)
d 1
n 'Alert state'
a (normal)
n ' '
w vout_command 58
d 4
n ' '
r read_status '1 status2' (40)
r read_status '2 status1' (01)
r read_status '3 alarm2' (00)
r read_status '4 alarm1' (00)
r read_status '5 voltage' (58,1,1)
r read_status '6 current' (0.3,50,100)
r read_status '7 temp' (24,60,20)
d 1
n 'Alert state'
a (normal)
n ' '
w vout_command 54
d 4
n ' '
r read_status '1 status2' (40)
r read_status '2 status1' (01)
r read_status '3 alarm2' (00)
```

```
r read_status '4 alarm1' (00)
r read_status '5 voltage' (54,1,1)
r read_status '6 current' (0.3,50,100)
r read_status '7 temp' (24,60,20)
d 1
n 'Alert state'
a (normal)
n ' '
n ' '
n 'change the high voltage shutdown level and test OV'
n ' '
w vout_ov_fault_limit 55
d 4
r read_status '1 status2' (40)
r read_status '2 status1' (01)
r read_status '3 alarm2' (00)
r read_status '4 alarm1' (00)
r read_status '5 voltage' (54,1,1)
r read_status '6 current' (0.3,50,100)
r read_status '7 temp' (24,60,20)
d 1
n 'Alert state'
a (normal)
n ' '
w vout_command 56
n ' '
d 30
n 'Alert state'
a (asserted)
n ' '
r read_status '1 status2' (40)
r read_status '2 status1' (30)
r read_status '3 alarm2' (00)
r read_status '4 alarm1' (04)
r read_status '5 voltage' (0.2,1,1)
r read_status '6 current' (0.3,50,100)
r read_status '7 temp' (24,60,20)
d 3
n 'Alert state'
a (asserted)
n ' '
w vout_command 54
d 2
n 'Alert state'
a (asserted)
n ' '
r read_status '1 status2' (40)
r read_status '2 status1' (30)
r read_status '3 alarm2' (00)
r read_status '4 alarm1' (04)
```

```
r read_status '5 voltage' (0.2,50,100)
r read_status '6 current' (0.3,50,100)
r read_status '7 temp' (24,60,20)
d 3
n 'Alert state'
a (asserted)
n ' '
n 'recover from latched state'
n ' '
w operation on:n
d 2
w operation on:y
d 2
n ' '
r read_status '1 status2' (40)
r read_status '2 status1' (01)
r read_status '3 alarm2' (00)
r read_status '4 alarm1' (00)
r read_status '5 voltage' (54,1,1)
r read_status '6 current' (0.3,50,100)
r read_status '7 temp' (24,60,20)
d 2
n ' '
n 'Alert state'
a (normal)
n ' '
n 'this is the end of the ov test'
```