

KITPFPGEVME Programmer for PF Series Devices



Figure 1. The PF Programmer (KITPFPGEVME)

This is the user guide for Rev. A of the KITPFPGEVME. Please upgrade to Rev. B of the KITPFPGEVME and follow the Rev. B user guide at: http://cache.freescale.com/files/analog/doc/user_guide/KITPFPGEVMEUG.pdf

Contents

1 Kit Contents / Packing List	2
2 Important Notice	4
3 Introduction	4
4 Software and Hardware Requirements	4
5 Software and Drivers Installation	8
6 Graphical User Interface Description	17
7 References	43
8 Revision History	44

1 Kit Contents / Packing List

- PF Series Programmer
- Flat ribbon cable (8-conductor)
- Mini USB cable
- Quick start guide
- Technical support brochure
- Warranty card

2 Important Notice

Freescale provides the enclosed product(s) under the following conditions:

This evaluation kit is intended for use of ENGINEERING DEVELOPMENT OR EVALUATION PURPOSES ONLY. It is provided as a sample IC pre-soldered to a printed circuit board to make it easier to access inputs, outputs, and supply terminals. This EVB may be used with any development system or other source of I/O signals by simply connecting it to the host MCU or computer board via off-the-shelf cables. This EVB is not a Reference Design and is not intended to represent a final design recommendation for any particular application. Final device in an application will be heavily dependent on proper printed circuit board layout and heat sinking design as well as attention to supply filtering, transient suppression, and I/O signal quality.

The goods provided may not be complete in terms of required design, marketing, and or manufacturing related protective considerations, including product safety measures typically found in the end product incorporating the goods. Due to the open construction of the product, it is the user's responsibility to take any and all appropriate precautions with regard to electrostatic discharge. In order to minimize risks associated with the customers applications, adequate design and operating safeguards must be provided by the customer to minimize inherent or procedural hazards. For any safety concerns, contact Freescale sales and technical support services.

Should this evaluation kit not meet the specifications indicated in the kit, it may be returned within 30 days from the date of delivery and will be replaced by a new kit.

Freescale reserves the right to make changes without further notice to any products herein. Freescale makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters can and do vary in different applications and actual performance may vary over time. All operating parameters, including "Typical", must be validated for each customer application by customer's technical experts.

Freescale does not convey any license under its patent rights nor the rights of others. Freescale products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Freescale product could create a situation where personal injury or death may occur.

Should the Buyer purchase or use Freescale products for any such unintended or unauthorized application, the Buyer shall indemnify and hold Freescale and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Freescale was negligent regarding the design or manufacture of the part. Freescale™ and the Freescale logo are trademarks of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners.

© Freescale Semiconductor, Inc. 2013-2015

3 Introduction

Freescale recommends the KITPFPGMMEVME development tool, also known as the PF Programmer, for configuring and programming the PF0100 Power Management Integrated Circuit. This PMIC is designed primarily for use with Freescale's i.MX6 series of application processors.

This document describes the specifications of the graphical user interface (GUI) for the PF0100 device. The primary features of this GUI are:

1. Read/write access to PF0100 registers
2. Intuitive control for programming the PF0100's start up configuration
3. Script to enable the 8.0 V Boost and 3.3 V LDO on the PF Programmer
4. Script to enable the configuration macros for fuse prototyping and fuse programming
5. Save or recall user defined configuration file

Note: This document provides updated information on the installation and use of the current PF0100 EVK Control GUI, revision 3.0.0.20. Some discrepancies may be found if using an earlier version of the GUI. To learn about the version of the GUI you are using, please refer to section "[Graphical User Interface Description](#)".

4 Software and Hardware Requirements

The KITPFPGMMEVME Programmer is a development tool designed to provide easy configuration of the PF Series devices and to facilitate prototyping and programming the device's (OTP) one-time programmable fuses. The programmer kit comes with a PF series programmer, a flat ribbon cable (8-conductor) and a mini-USB cable, as shown in [Figure 2](#).



Figure 2. KITPFPGMMEVME PF Series Programmer

The KITPFPGMMEVME incorporates a MC9S908JM60 8-bit MCU, a MAX686 boost regulator, and a MIC5202 LDO. The programming voltage is supplied by the boost regulator, which boosts the nominal 5.0 V from the USB port to the required 8.25 V.

The programmer is connected to the customer's board via a flat cable with a standard 8-pin (4x2), 100 mil pitch header plug. [Figure 3](#) shows the pinout and signals of the Programming Interface connector.

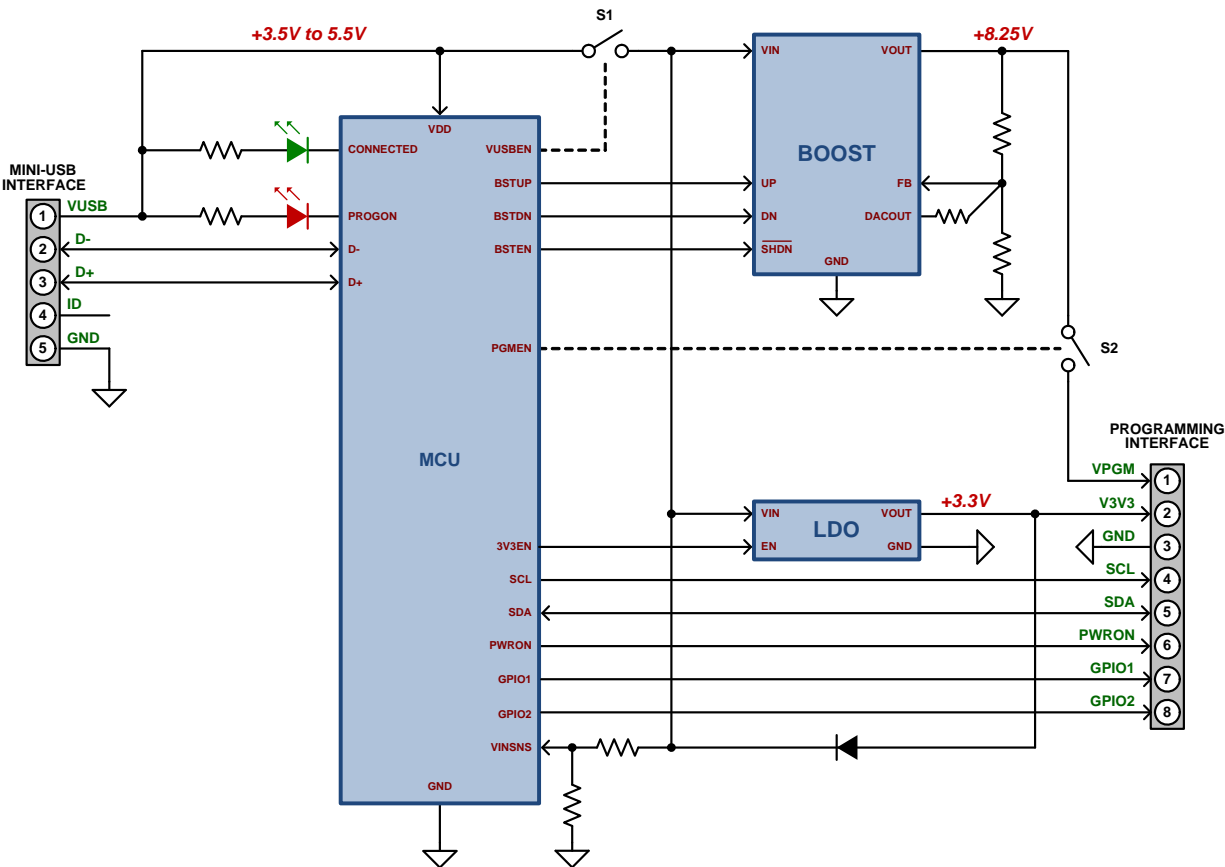


Figure 3. KITPFGMEVME Block Diagram

4.1 Designing the Target PF0100 Application

When connecting the KITPFGMEVME programmer to an application board with an i.MX processor and PF0100 sharing the I²C bus, it is required to disconnect the processor I²C lines from the communication bus.

Figure 4 shows a simplified diagram to be able to program the PF0100 with the KITPFGMEVME programmer. Since programming the OTP fuse on the PF0100 is a one-time event during production, the most simple and cost-effective solution for isolating the processor I²C lines is to add two 0 Ω resistors (R5 and R6) which should not be placed during OTP programming, but are soldered down for normal operation. Note that Figure 4 represents only a recommended configuration; it is the customer’s responsibility to make sure this proposal meets the requirements of a specific application.

PWRON pin is configured to be LOW so the processor will be in reset, since the PORB pin in typical application is connected to the RESETBMCU pin of PF0100.

Note: ICTEST should be grounded.

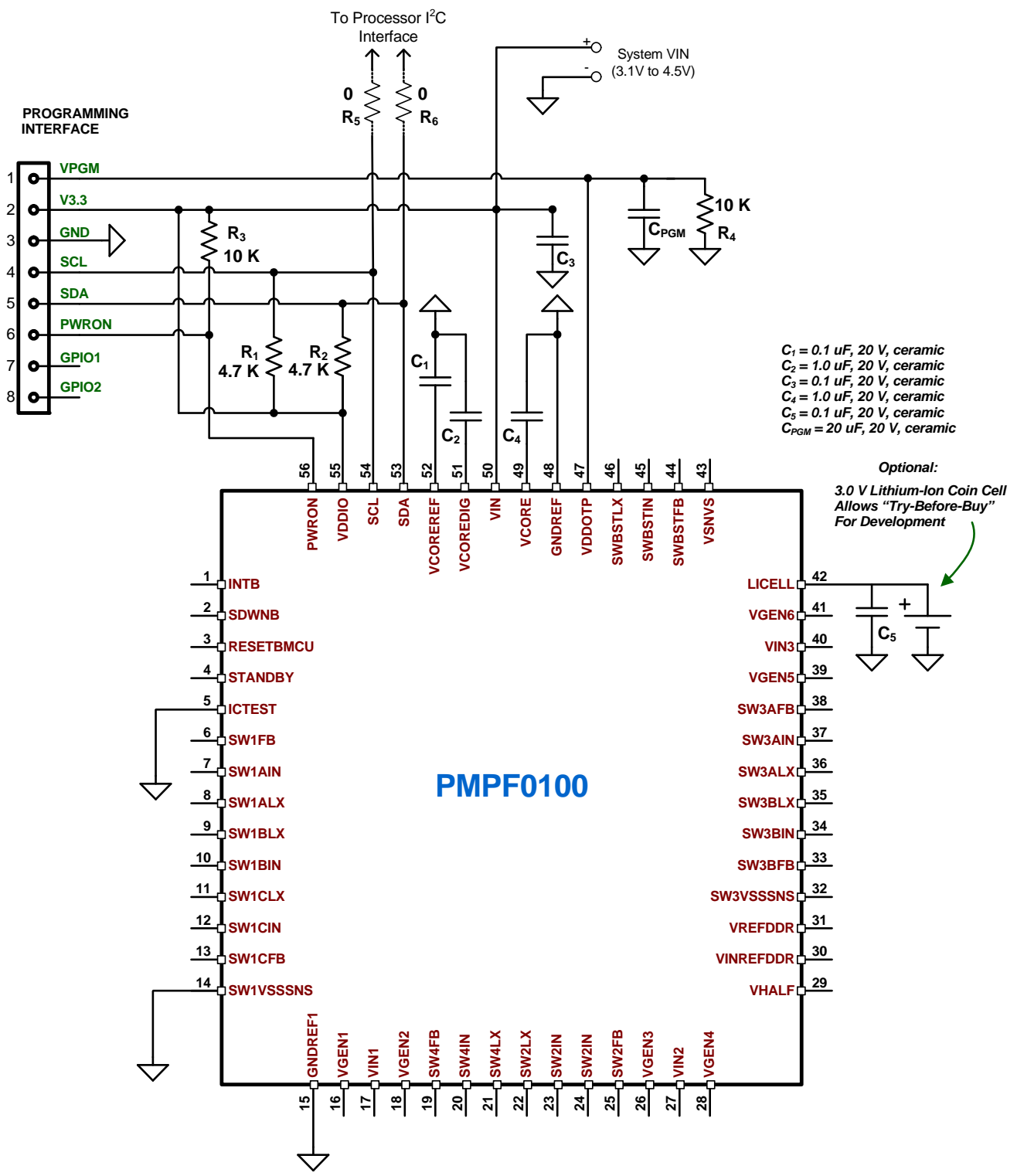


Figure 4. PF100 Minimum System for Programming

4.2 Hardware Requirements

- PC with Windows XP or Windows 7 operating system
- Standard USB port
- 8-pin (4x2) 100 mil IDC male connector on target board (recommended)

4.3 Software Requirements

Necessary software and drivers include:

- Microsoft .NET Framework 4.0
- NI-VISA 5.1.2 communication package plus development support (including .NET Framework 4.0 languages)
- KITPF0100GUI.zip: Graphical User Interface (GUI) for PF0100 Power Management IC.

To download the latest version of KITPF0100GUI.zip, go to the link:

http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=KITPFGMEVME&fpp=1&tab=Design_Tools_Tab

5 Software and Drivers Installation

- 1. Install Microsoft .NET Framework 4.0, download and run "dotNetFx40_client_x86_x84.exe". Click on link below <http://www.microsoft.com/en-us/download/details.aspx?id=24872>
- 2. Install Windows Installer 3.1 (Windows XP Only), download and run "WindowsInstaller-KB93803-v2-x86.exe". Onl Click on link below <http://www.microsoft.com/en-us/download/details.aspx?id=25>
- 3. Install NI-VISA 5.1.2, download and run "visa512.exe". Click on link below [NI-VISA 5.1.2 - National Instruments](#)

Note: It is the customer's responsibility to obtain any license files from National Instruments that are necessary for enabling the NI-VISA 5.1.2 drivers.

When installing the NI-VISA 5.1.2, make sure to select the .NET Framework 4.0 Language Support drivers as shown in Figure 5.

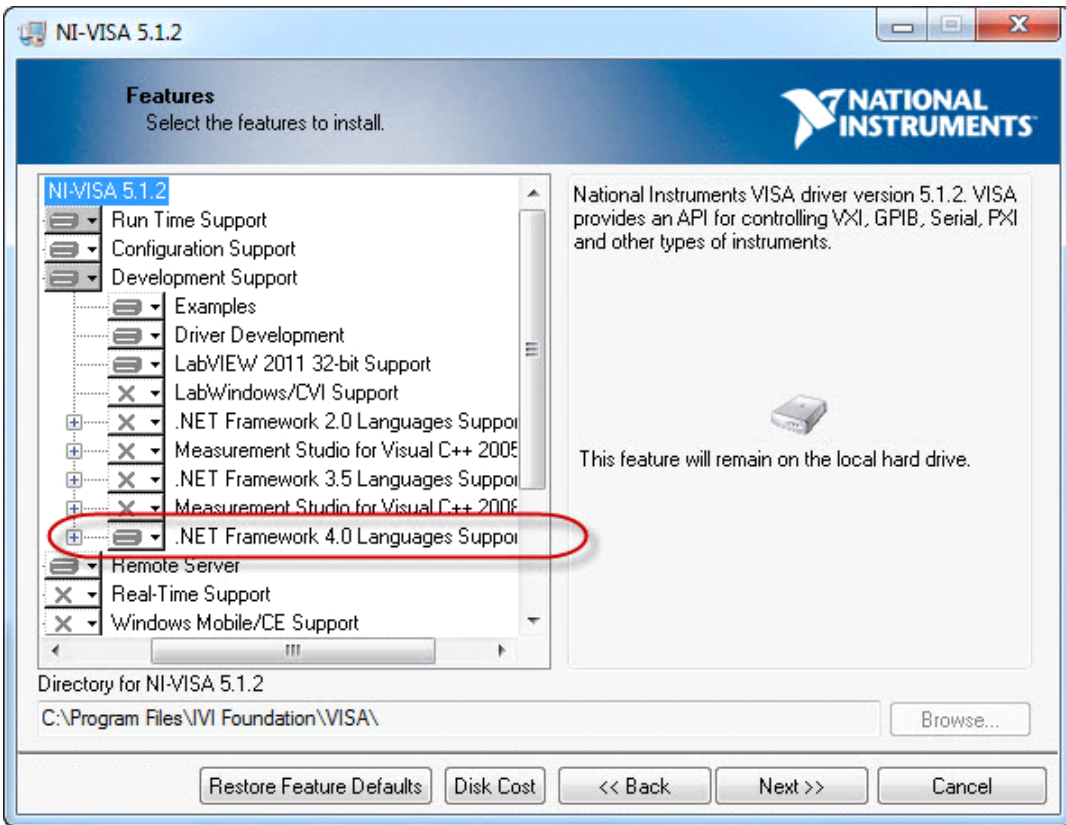


Figure 5. NI-VISA 5.1.2 Features Install window

5.1 Using NI VISA Driver Wizard to install .INF file

Before proceeding to generate and install the proper .INF files, connect the KITPFGMEVME programmer to the PC USB port using the standard type A-to-B mini USB cable. Windows will then detect the new hardware. Next, proceed with the installation. Do not connect to Windows Update to search for software.



Figure 6. Install New PF Programmer Hardware

On the following window, select the “Install Software Automatically” option and click next.

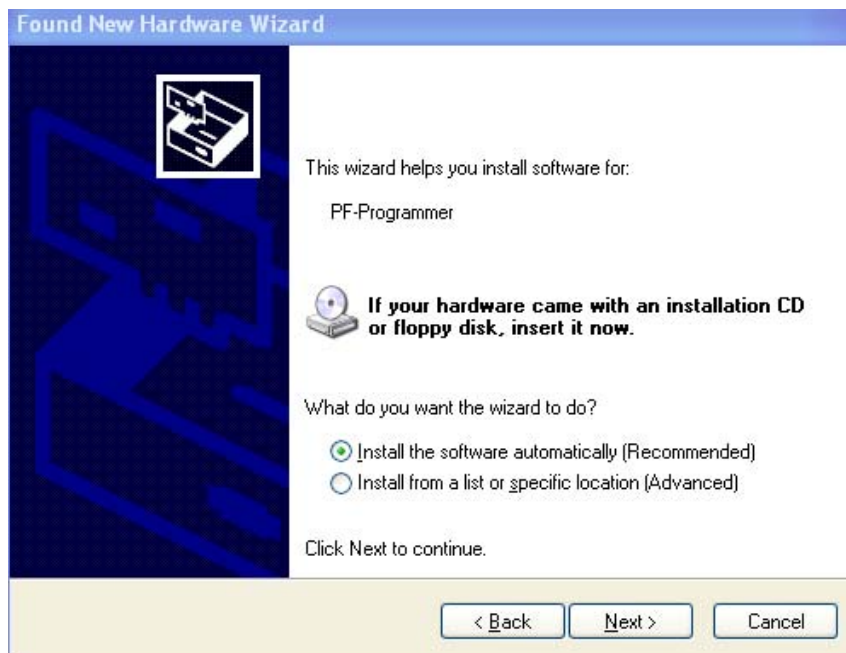


Figure 7. Automatic Hardware Installation

Verify on the Windows Device Manager, that the hardware was properly installed.

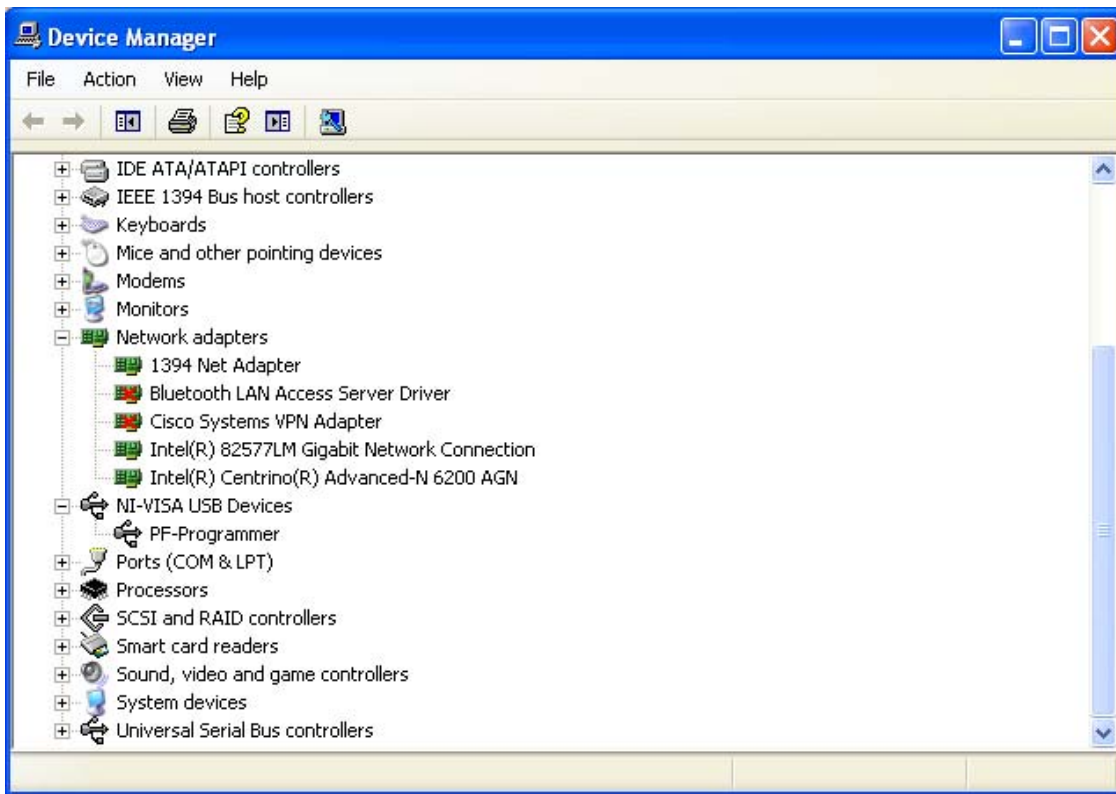


Figure 8. Windows Device Manager

Now proceed with the .INF creation with the NI-VISA Driver Wizard:

1. Go to Start > Programs > National Instruments > VISA > Driver Wizard
2. Select USB under Hardware Bus Selection and press "Next" button

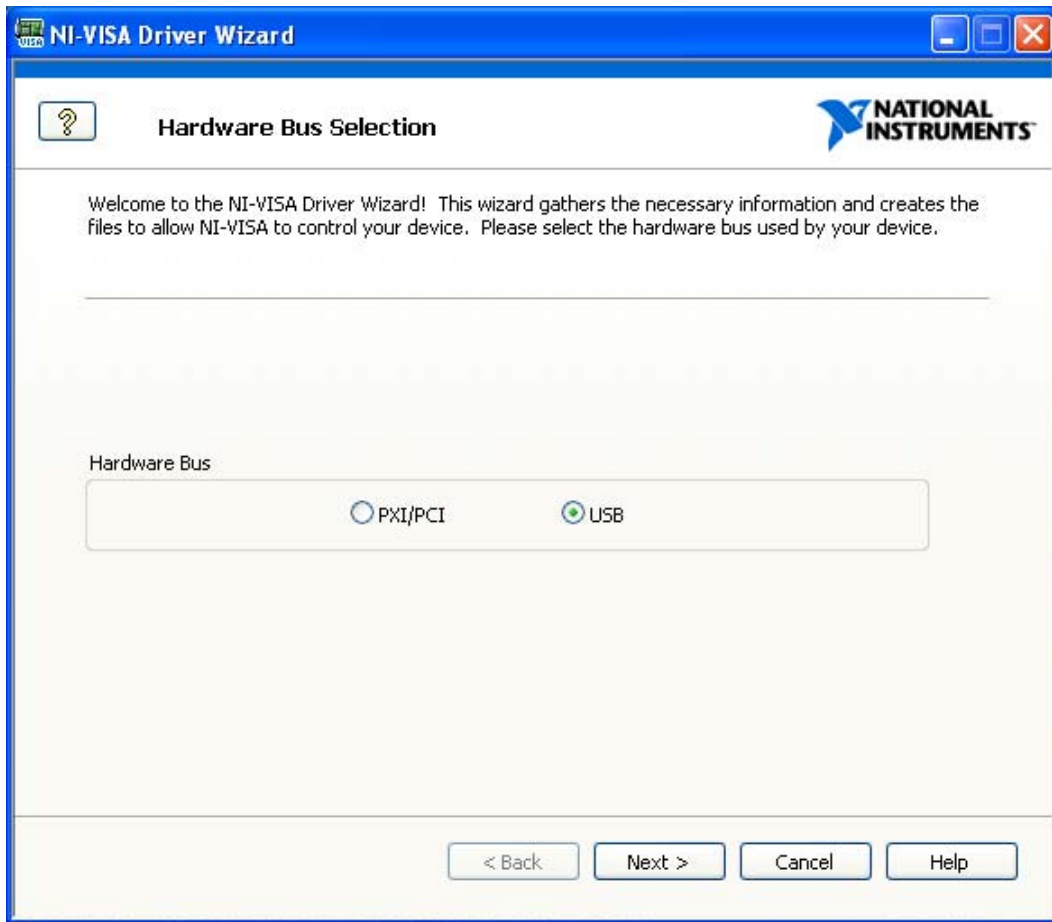


Figure 9. NI-VISA Hardware Bus Selection Window

If the application prompts an error message, disregard the error and click continue.

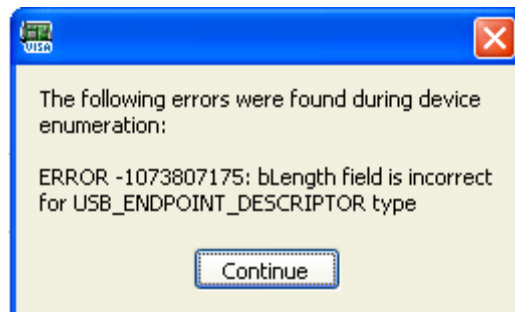


Figure 10. USB Error Message (Disregard)

3. Select the correct device from the device list, make sure the USB Manufacturer ID is "15A2" and the USB Mode Code (Product ID) is "0050", then press the "Next" button.

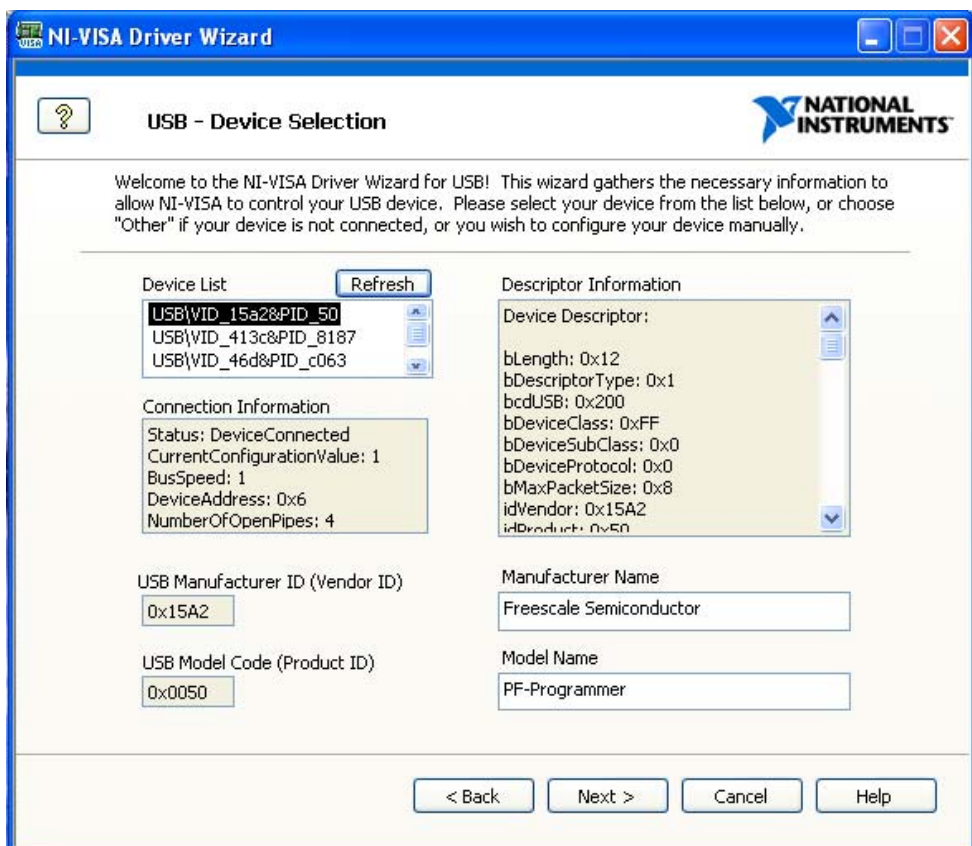


Figure 11. NI-VISA USB Device Selection Window

4. Enter "PF-Programmer" in the instrument prefix field, browse to a folder where you want to save the output file and press the "Next" button

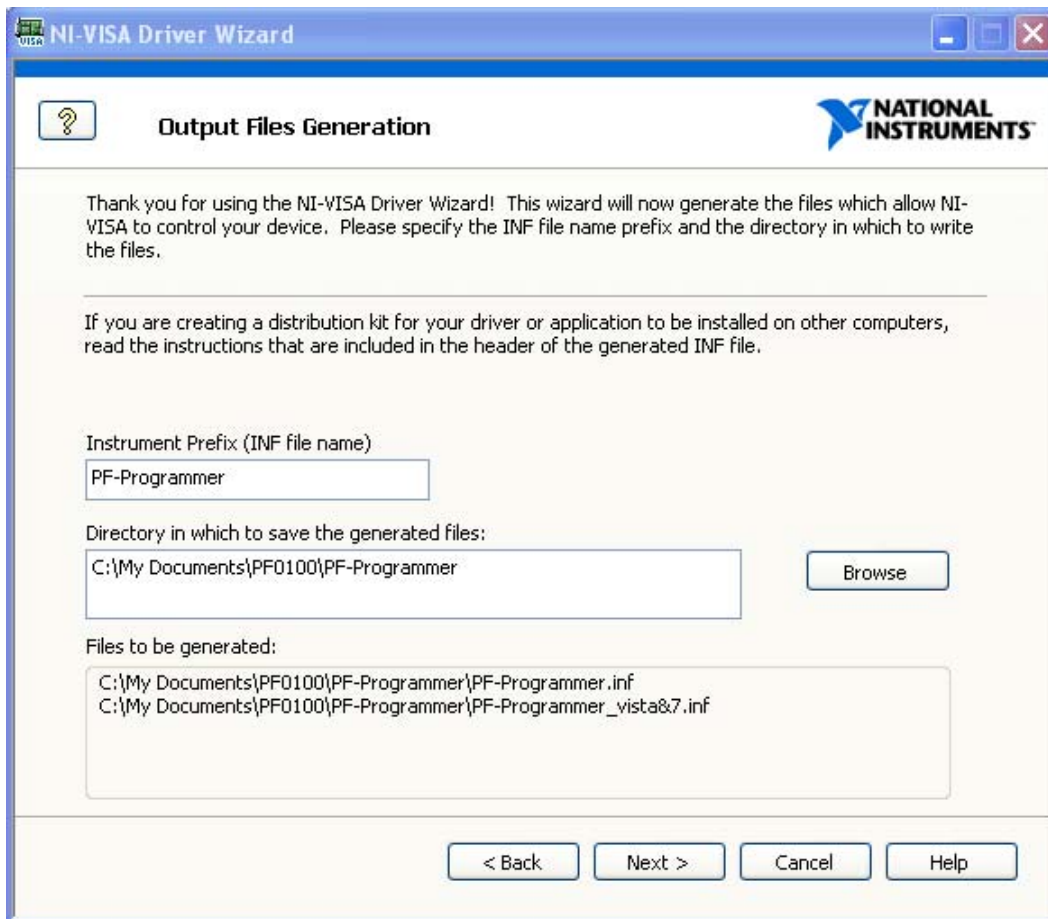


Figure 12. NI-VISA Output File Generation Window

5. Select the option of automatically installing the generated .INF driver and press the “Finish” button. If the software prompts an error message that the device could not be found, disconnect and reconnect the KITPFGMEVME programmer and click on the “Retry” button.

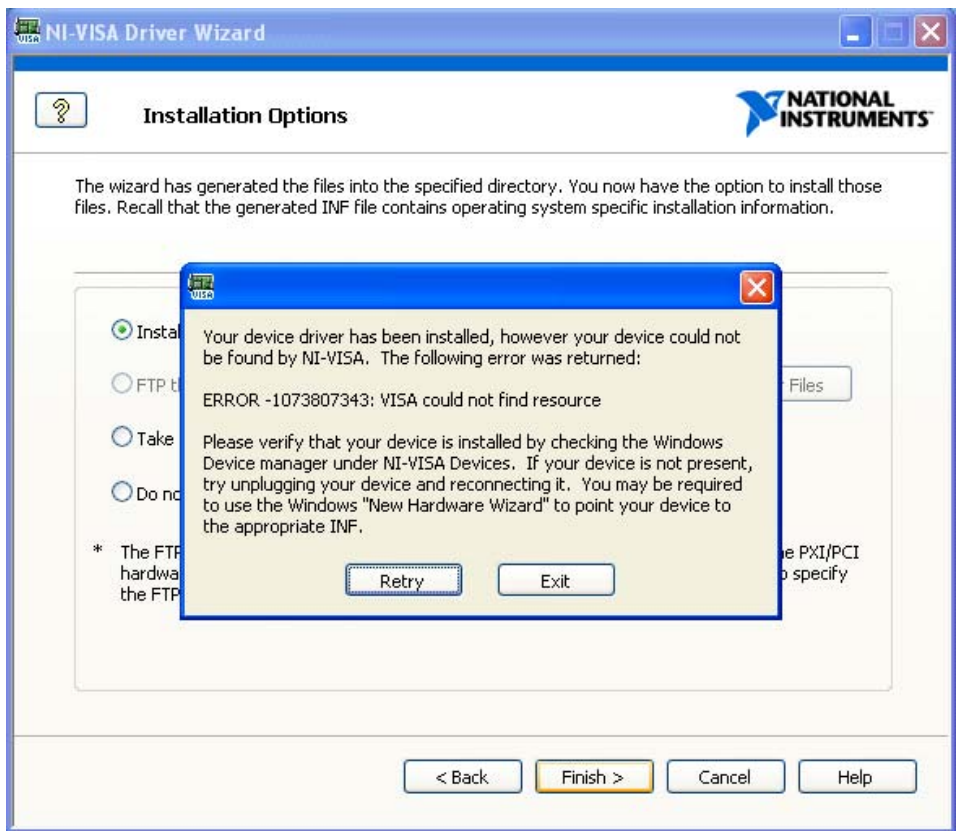


Figure 13. NI-VISA Installation Options Window

6. To check if the USB device driver was installed correctly, go to Start > Setting > Control Panels > System > Device Driver. You should see "PF-Programmer" under "NI-VISA USB Devices".

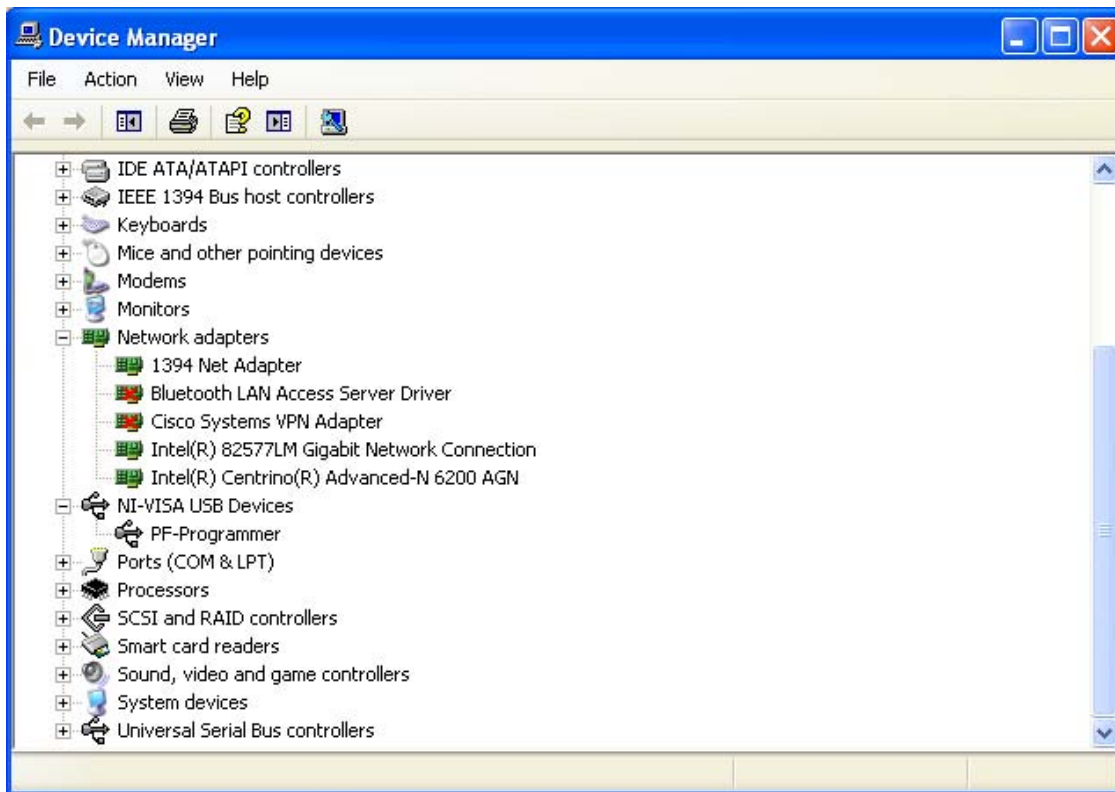


Figure 14. Microsoft Windows Device Manager

5.2 Connect the Target Board to KITPFGMEVME Programmer

Connect the KITPFGMEVME programmer to your target board using the supplied 8-conductor flat ribbon cable. The flat cable is keyed so that the correct header pins are connected on the programmer.

WARNING: Verify that the brown wire on the other end of the flat cable is connected to pin 1 on your target board header. Freescale recommends a shrouded pin header (TST-104-01-G-D) to be placed on the target board to prevent the cable from being incorrectly inserted into the header and damaging your board.

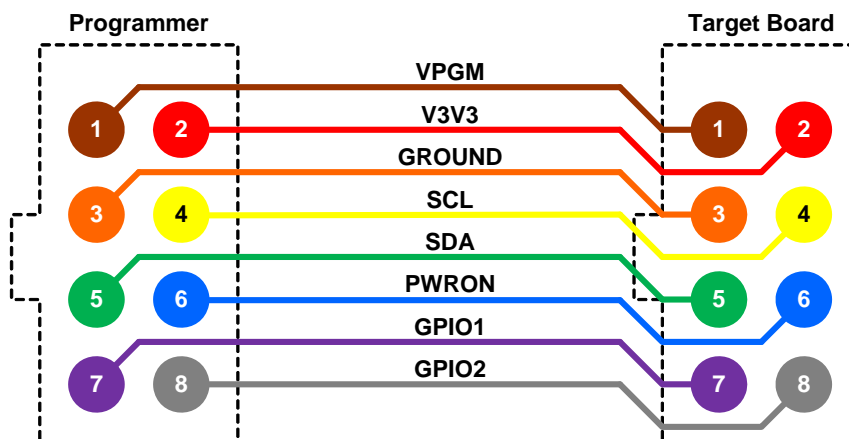


Figure 15. Ribbon Cable Wiring Diagram

Table 1. Pin Description

PIN	NAME	FUNCTION
1	VPGM	Programming voltage for OTP fuses
2	V3V3	Auxiliary 3.3V supply
3	GND	Ground
4	SCL	I ² C Master clock
5	SDA	I ² C Master data
6	PWRON	Logic output to turn-on/turn-off the PMIC
7	GPIO1	GPIO signal from MCU (Open drain)
8	GPIO2	GPIO signal from MCU (Logic output)

5.3 Installing the KITPF0100GUI

1. Create a directory on your PC as follows:
C:\Freescale\KITPF0100
2. Extract the KITPF0100GUI.zip file into that directory.
3. Launch the "setup.exe" program.
4. When the following pop-up dialog appears, press the "Install" button.



Figure 16. KITPF0100GUI installation Window

If everything installs correctly, the next screen you will see is the application GUI.

6 Graphical User Interface Description

The KITPF0100GUI is a graphical user interface created to control the KITPF0100EPEVBE evaluation board as well as to control any PF0100 based application board through the KITPFGMEVME programmer. The GUI is divided into tabs which can be selected to display a window associated with each functional aspect of the PF0100 device. Note that, depending on the device connected to the USB port, the GUI enables or disables sections not accessible to a specific device. The following section describes only the tabs used to control a PF0100 device with the KITPFGMEVME programming board.

6.1 Getting Started

1. Connect the PF Programmer to the target board using the 8-pin ribbon cable, then connect the USB cable from the Mini-B USB port to the computer.
2. Press the “Open Session” button to search for the PF0100 Programmer device.
3. The “Select Resource” dialog box should pop up, and you should see the KITPFGMEVME device listed. The USB Vendor ID is 0x15A2, and the Part ID is 0x0050 for the KITPFGMEVME.

Select the device and press the “OK” button.

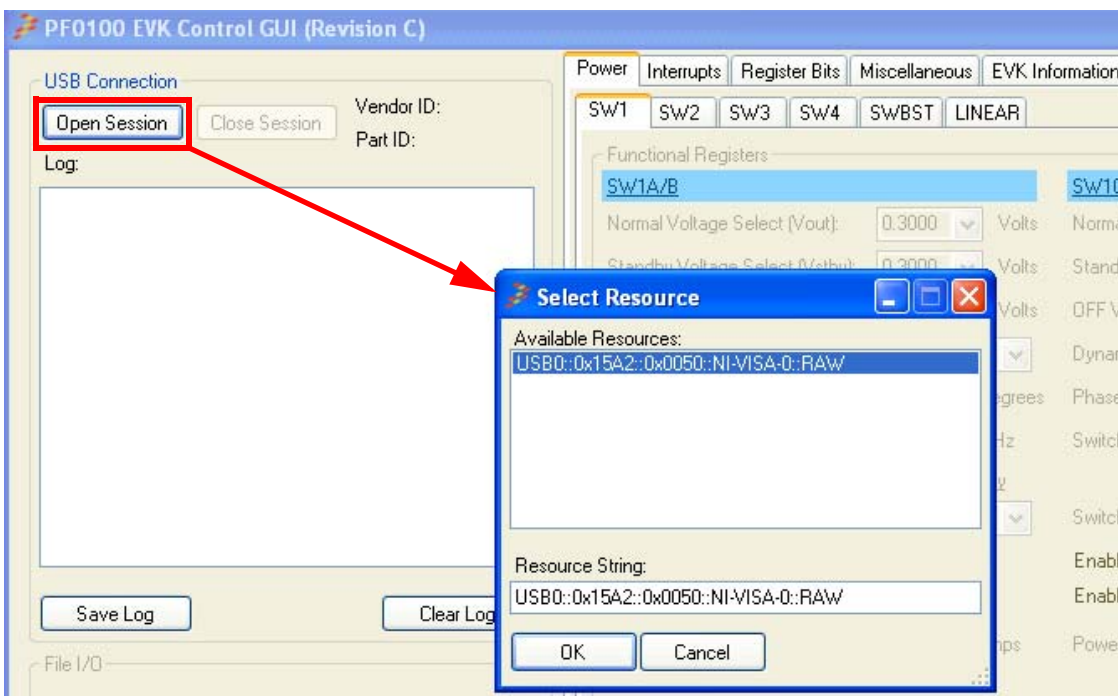


Figure 17. OPEN USB Connection

4. The “Select Resource” dialog should close, and the Log List will display a message that you are connected to the KITPFGMEVME. The Vendor ID and Part ID will also be displayed to let you know that you are connected to the right device.
 - The Log List can be saved to a text file at any time by pressing the “Save Log” button, and a “File Save” dialog box will pop up.
 - The Log List can also be cleared at any time by pressing the “Clear Log” button.

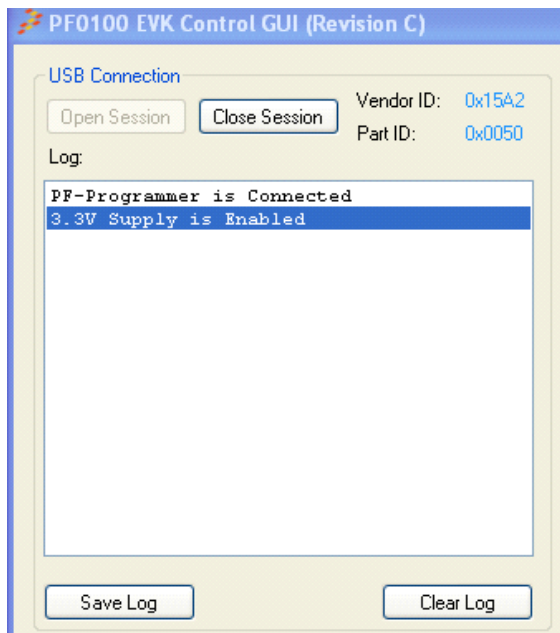


Figure 18. Log List Box.

- If prototyping is desired, the Try-Before-Buy mode can be enabled or disabled through the TBB On or TBB Off buttons respectively. The operating mode will be displayed on the lower left area of the KITPF0100GUI as shown in [Figure 19](#). The KITPF0100GUI version is shown in the lower right area of the graphical interface.



Figure 19. TBB Mode Enable/Disable

6.2 Verify I²C Communication to PF0100

Use the “Byte Write” button to write one byte of data to register 0x1C of PF0100 and use the “Byte Read” button to read back the register contents at address 0x1C to verify correct data was written. The Log List should also reflect what I²C transactions the GUI has processed.

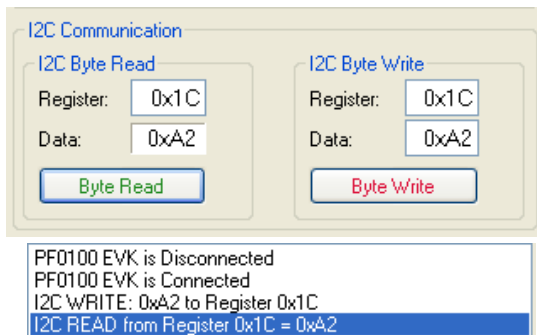


Figure 20. Verify I²C Communication.

6.3 GUI Features

There are five functional tabs in the KITPF0100GUI to allow programming PF0100 registers. They also permit one to perform either prototyping or one-time programming (OTP) of the fuse registers.

The *Power* tab provides user access to control both the switching as well as the linear regulators on the PF0100.

The switching regulators can be programmed by sub-tabs for SW1, SW2, SW3, SW4 and SWBST. For each of these tabs, the user has access to functional registers which configure the output voltage, the standby voltage, the off-mode voltage, the phase control, the switching frequency and more. The user can also configure the OTP register via the bottom half of the tab, which selects the start-up sequence of the switching regulators as well as phase configuration for SW1 and SW3.

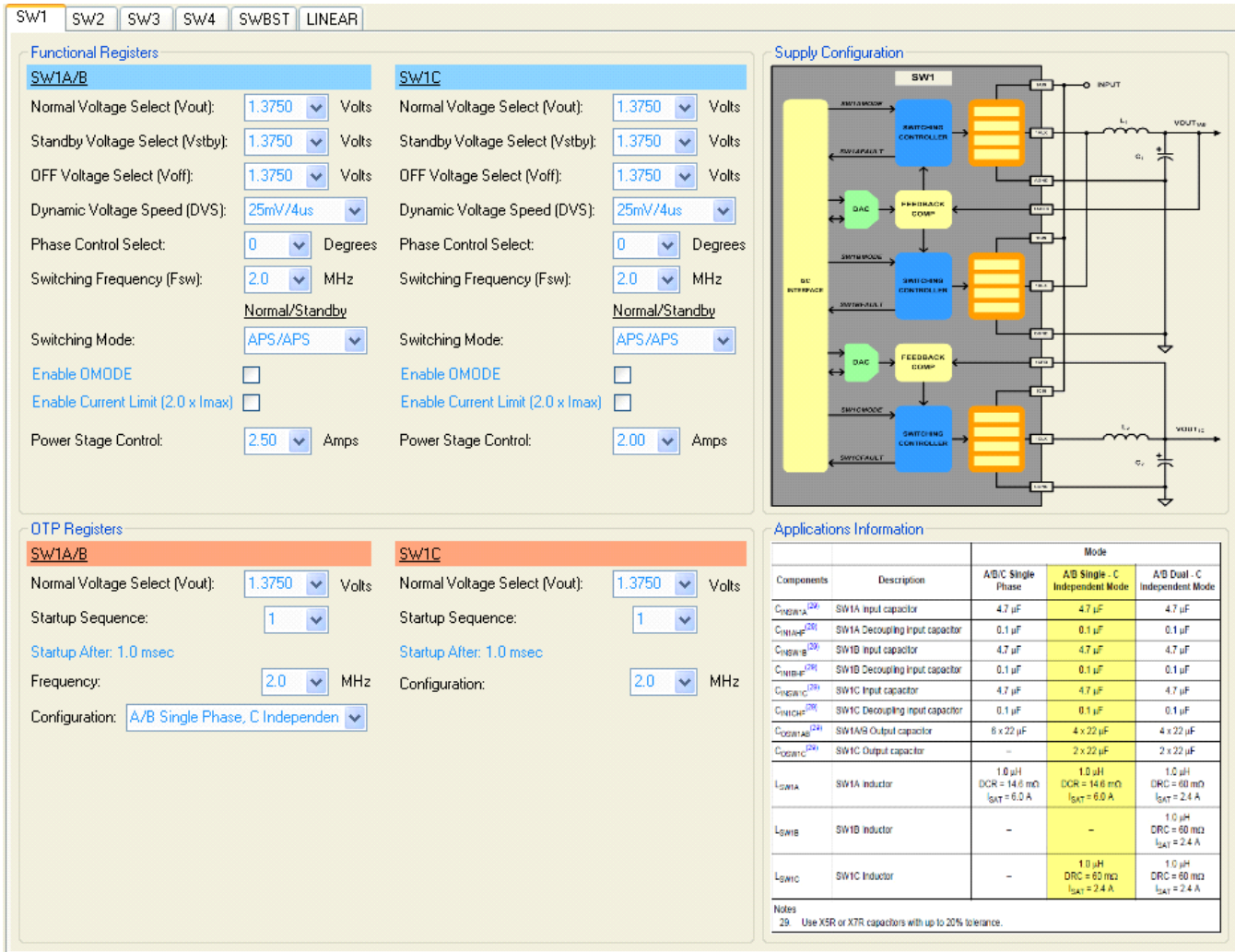


Figure 21. Switching Regulators Tab

Linear regulators can be programmed using the LINEAR sub-tab. It enables the user to control the output voltage and the start-up sequence of the regulators as well as an option to choose standby and low power mode.

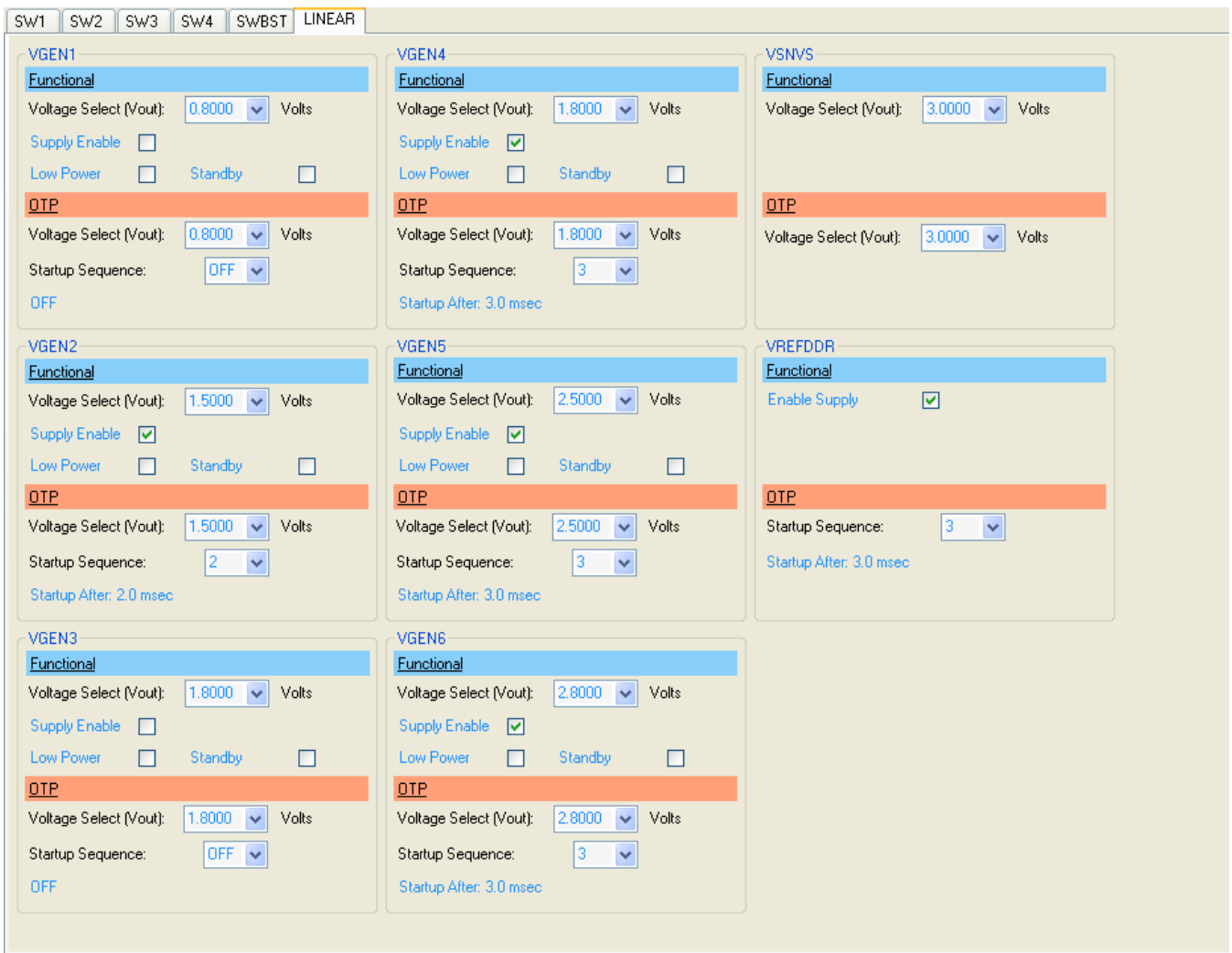


Figure 22. LDO Regulators Tab

The *Interrupts* tab provides user access to the four interrupt registers in PF0100 functional register map. The user can choose to read the interrupts by pressing on the “Read Interrupt x” button.

Each interrupt is latched so that even if the interrupt source becomes inactive, the interrupt will remain set until cleared. Each interrupt can be cleared by writing a “1” to the appropriate bit in the Interrupt Status register. This will also cause the INTB pin to go high.

Each interrupt can be masked by setting the corresponding mask bit to a 1. As a result, when a masked interrupt bit goes high, the INTB pin will not go low. A masked interrupt can still be read from the Interrupt Status register.

The sense registers contain status and input sense bits so the system processor can poll the current state of interrupt sources. They are read only, and neither latched nor clearable.

The user may choose to press the “Poll Interrupt x” control to read the *Interrupts* tab every 500ms.

Power Interrupts Register Bits Miscellaneous EVK TestScript Editor Tool Testbench

Interrupt 0

STATUS	MASK	SENSE	TRIGGER	DEBOUNCE TIME
<input checked="" type="checkbox"/> Power On	<input checked="" type="checkbox"/>	■	H to L	3.9 msec
<input type="checkbox"/> Low Voltage	<input checked="" type="checkbox"/>	■	H to L	31.25 msec
<input type="checkbox"/> 110°C Thermal	<input checked="" type="checkbox"/>	■	Dual	3.9 msec
<input type="checkbox"/> 120°C Thermal	<input checked="" type="checkbox"/>	■	Dual	3.9 msec
<input type="checkbox"/> 125°C Thermal	<input checked="" type="checkbox"/>	■	Dual	3.9 msec
<input type="checkbox"/> 130°C Thermal	<input checked="" type="checkbox"/>	■	Dual	3.9 msec
<input type="checkbox"/> Poll Interrupt 0	■	<input type="button" value="Read Interrupt 0"/>		

Interrupt 3

STATUS	MASK	SENSE	TRIGGER	DEBOUNCE TIME
<input type="checkbox"/> SWBST Over-current	<input checked="" type="checkbox"/>	■	L to H	8.0 msec
<input type="checkbox"/> OTP Error	<input checked="" type="checkbox"/>	■	L to H	8.0 msec
<input type="checkbox"/> Poll Interrupt 3	■	<input type="button" value="Read Interrupt 3"/>		

Interrupt 1

STATUS	MASK	SENSE	TRIGGER	DEBOUNCE TIME
<input type="checkbox"/> SW1A Over-current	<input checked="" type="checkbox"/>	■	L to H	8.0 msec
<input type="checkbox"/> SW1B Over-current	<input checked="" type="checkbox"/>	■	L to H	8.0 msec
<input type="checkbox"/> SW1C Over-current	<input checked="" type="checkbox"/>	■	L to H	8.0 msec
<input type="checkbox"/> SW2 Over-current	<input checked="" type="checkbox"/>	■	L to H	8.0 msec
<input type="checkbox"/> SW3A Over-current	<input checked="" type="checkbox"/>	■	L to H	8.0 msec
<input type="checkbox"/> SW3B Over-current	<input checked="" type="checkbox"/>	■	L to H	8.0 msec
<input type="checkbox"/> SW4 Over-current	<input checked="" type="checkbox"/>	■	L to H	8.0 msec
<input type="checkbox"/> Poll Interrupt 1	■	<input type="button" value="Read Interrupt 1"/>		

Interrupt 4

STATUS	MASK	SENSE	TRIGGER	DEBOUNCE TIME
<input type="checkbox"/> VGEN1 Over-current	<input checked="" type="checkbox"/>	■	L to H	8.0 msec
<input type="checkbox"/> VGEN2 Over-current	<input checked="" type="checkbox"/>	■	L to H	8.0 msec
<input type="checkbox"/> VGEN3 Over-current	<input checked="" type="checkbox"/>	■	L to H	8.0 msec
<input type="checkbox"/> VGEN4 Over-current	<input checked="" type="checkbox"/>	■	L to H	8.0 msec
<input type="checkbox"/> VGEN5 Over-current	<input checked="" type="checkbox"/>	■	L to H	8.0 msec
<input type="checkbox"/> VGEN6 Over-current	<input checked="" type="checkbox"/>	■	L to H	8.0 msec
<input type="checkbox"/> Poll Interrupt 4	■	<input type="button" value="Read Interrupt 4"/>		

Figure 23. Interrupts Tab

The *Register Bits* tab allows bit-programming of the PF0100 registers. This is an alternative to the I²C single byte write and read controls shown in section “[Verify I2C Communication to PF0100](#)”.

Power	Interrupts	Register Bits	Miscellaneous	EvScript Editorts	Script Tool	Testbench						
SW1	SW2	SW3	SW4 / SWBST	LINEAR 1	LINEAR 2	INTERRUPTS	RAM	tabPage22				
SW1AB				SW1C								
0x20	D7	D6	D5	D4	D3	D2	D1	D0	VOUT	0x2B	VOUT	0x2B
	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	1.3750 Volts			1.3750 Volts
	X	X							VOUT			VOUT
0x21	D7	D6	D5	D4	D3	D2	D1	D0	VSTBY	0x2B	VSTBY	0x2B
	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	1.3750 Volts			1.3750 Volts
	X	X							VSTBY			VSTBY
0x22	D7	D6	D5	D4	D3	D2	D1	D0	VOFF	0x2B	VOFF	0x2B
	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	1.3750 Volts			1.3750 Volts
	X	X							VOFF			VOFF
0x23	D7	D6	D5	D4	D3	D2	D1	D0	MODE	0x08	MODE	0x08
	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	APSIAPS			APSIAPS
	X	X	QM	X					MODE			MODE
0x24	D7	D6	D5	D4	D3	D2	D1	D0	CONFIG	0x44	CONFIG	0x44
	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	D: 25mV/4us			D: 25mV/4us
		DVS	PHASE		FWS		X	I	P: 0 *			P: 0 *
									F: 2.0 MHz			F: 2.0 MHz
SW1AB POWER STAGE (Ext Page 2)				SW1C POWER STAGE (Ext Page 2)								
0x81	D7	D6	D5	D4	D3	D2	D1	D0	PSEG	0x07	PSEG	0x07
	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	2.50			2.00
	X	X	X	X	X				PWR			PWR
SW1AB OTP REGISTERS (Ext Page 1)				SW1C OTP REGISTERS (Ext Page 1)								
0xA0	D7	D6	D5	D4	D3	D2	D1	D0	VOUT	0x2B	VOUT	0x2B
	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	1.3750 Volts			1.3750 Volts
	X	X							VOUT			VOUT
0xA1	D7	D6	D5	D4	D3	D2	D1	D0	SEQ	0x01	SEQ	0x01
	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	1			1
	X	X	X						SEQUENCE			SEQUENCE
0xA2	D7	D6	D5	D4	D3	D2	D1	D0	CONFIG	0x05	CONFIG	0x01
	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	AB Single.			F: 2.0 MHz
	X	X	X	X					C Indpt			
									F: 2.0 MHz			

Figure 24. Register Bits Tab

The *Miscellaneous* tab allows the user to read the silicon device ID, configure the coin cell charger, enable LDO short-circuit protection, set the de-bounce time of logic IO, and communicate with memory registers A-D.



Figure 25. Miscellaneous Tab

The *Script Editor* Tab allows one to create, load and save configuration scripts for the PF0100 device. For more information on how to create a configuration script, see section [“Using the Script Editor”](#).

6.4 Using the Script Editor

The Script Editor is a powerful tool that automates the PF0100 development process. Scripts are groups of commands that are executed sequentially. They can quickly load PF0100 registers with your desired configuration, or they can help you to determine the correct power-up sequence for your design. Scripts are stored as simple text files, and as such, can be edited with any text editor. Since scripts are driven by your PC, PMIC configurations can be explored and validated prior to connecting to a host i.MX processor.

The Script Editor work area is shown in [Figure 26](#). Script files are created in the large work area to the left. The blank area to the right-hand side is the Script Log, which displays the script output as it steps sequentially.

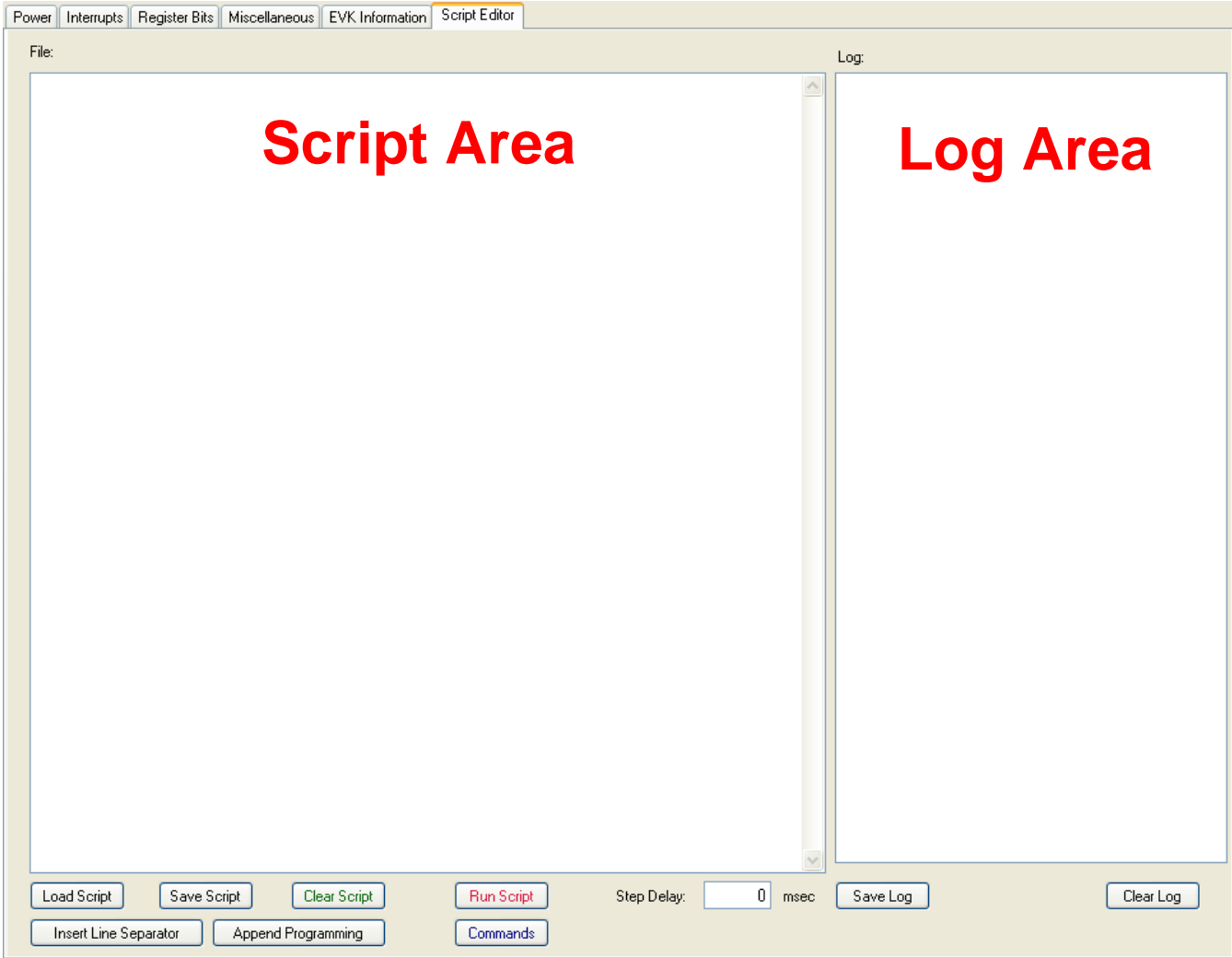


Figure 26. Script Editor Window

The following list describes all the available buttons on the Script Editor tab.

- **Load Script:** Launches the “File Load” dialog box allowing the user to select and load a stored script file.
- **Save Script:** Launches the “File Save” dialog box, allowing the user to save a script file to storage.
- **Clear Script:** Clears the current Script Editor work area to prepare for writing a new script.
- **Run Script:** Begins execution of the currently loaded script. Execution runs sequentially.
- **Step Delay:** Entered as an integer number between 0 and 1000 milliseconds. Double-click with the left mouse button over the text box to begin editing the value, then press the Enter key.
- **Insert Line Separator:** Inserts a comment at the current cursor position that represents a separating line. Used to organize long scripts.
- **Append Programming:** Inserts all the commands required to program the OTP memory into the Script Editor at the current cursor location.
- **Save Log:** Launches the “File Save” dialog box, allowing users to save the Script Log to a file.
- **Clear Log:** Clears the Script Log.
- **Commands:** Display a pop-up window shown in [Figure 31](#), with a graphical set of commands to add to the script.

6.4.1 Loading and Running a Script

To load a preexisting script file, press the “Load Script” Button. The “File Load” dialog box will appear, allowing you to navigate to the directory where your script file is located. Select the file you want and click on the “Open” button.

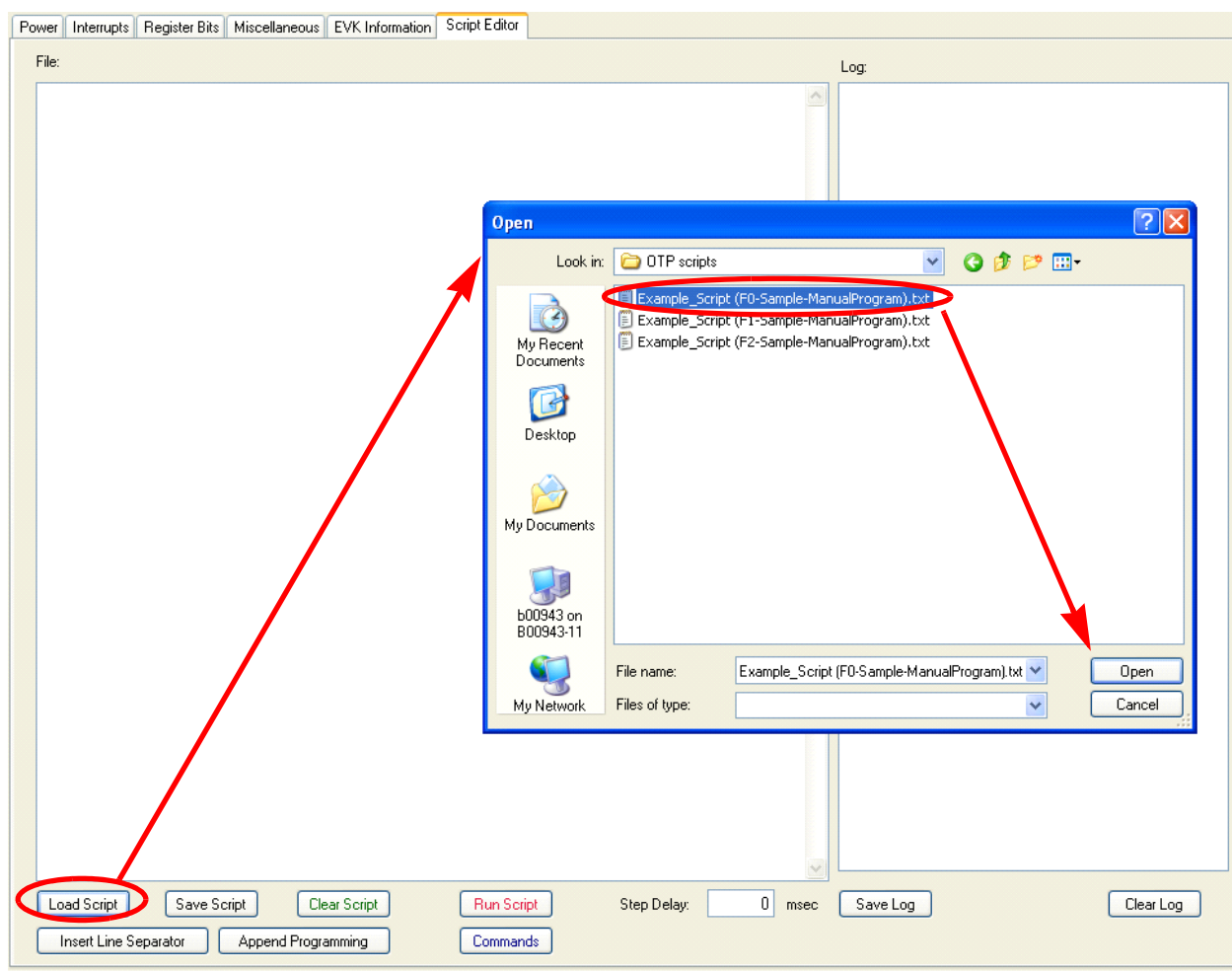


Figure 27. Loading a Script File

The Script Editor work area will now be filled with the file content, and the file name will appear next to the "File:" label and also as an entry in the Script Log.

Next, change the script’s Step Delay to allow delay between each command. For instance, 50 ms are used in the example shown in [Figure 28](#). To make this change, double-click with the left mouse button while pointing to the “Step Delay” text box. The text box background color will turn pink, indicating that the value is being changed, but has not yet been updated. Enter the desired delay value, and press the Enter key. Notice that the text box background color returns to white, indicating that your updated value has been accepted.

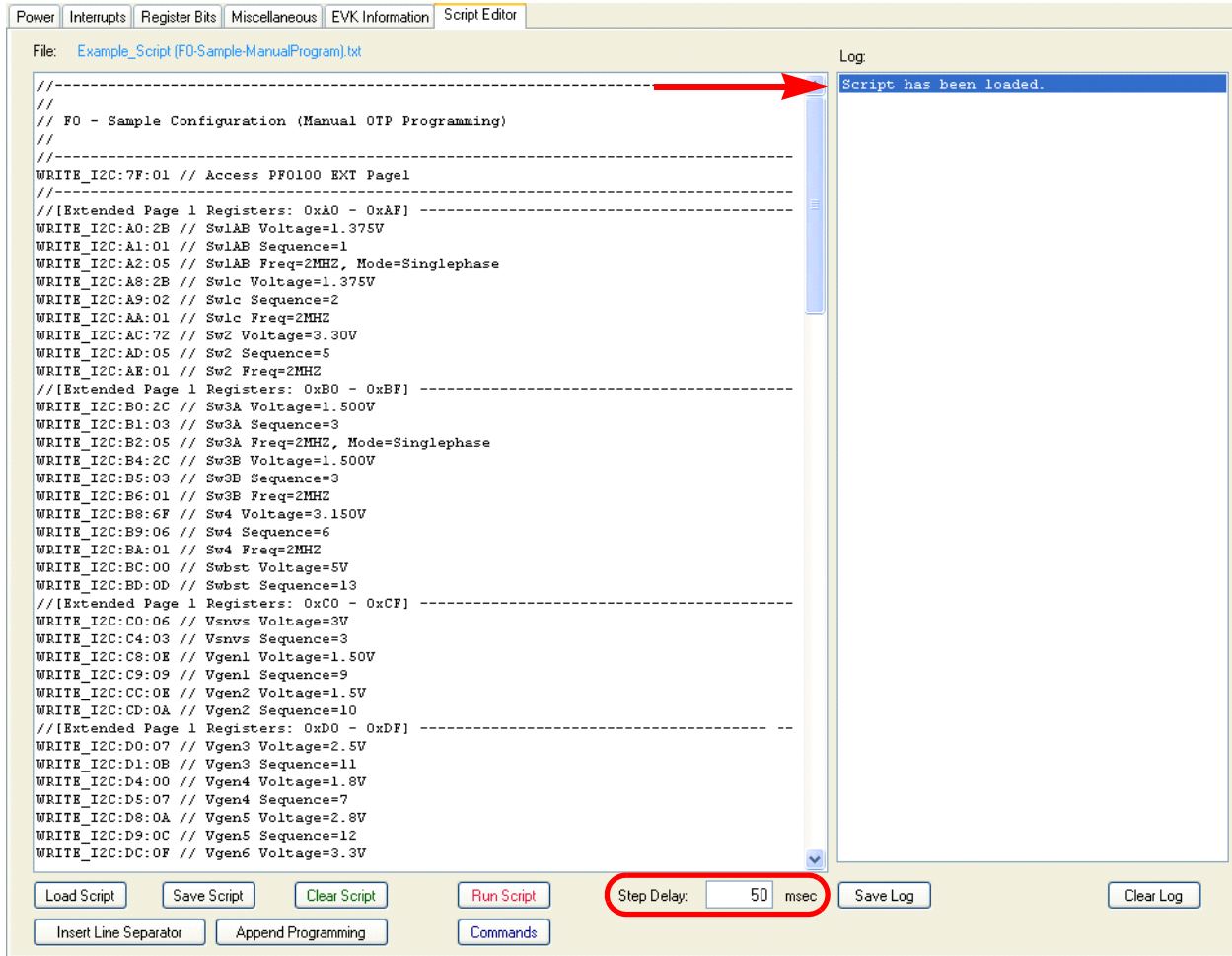


Figure 28. Script Dialog Box

Press the “Run Script” button to execute the script. As the script executes, each command will appear sequentially in the Script Log. Comments are ignored. When the script has completed, an entry in the Script Log will be made as shown in Figure 29.

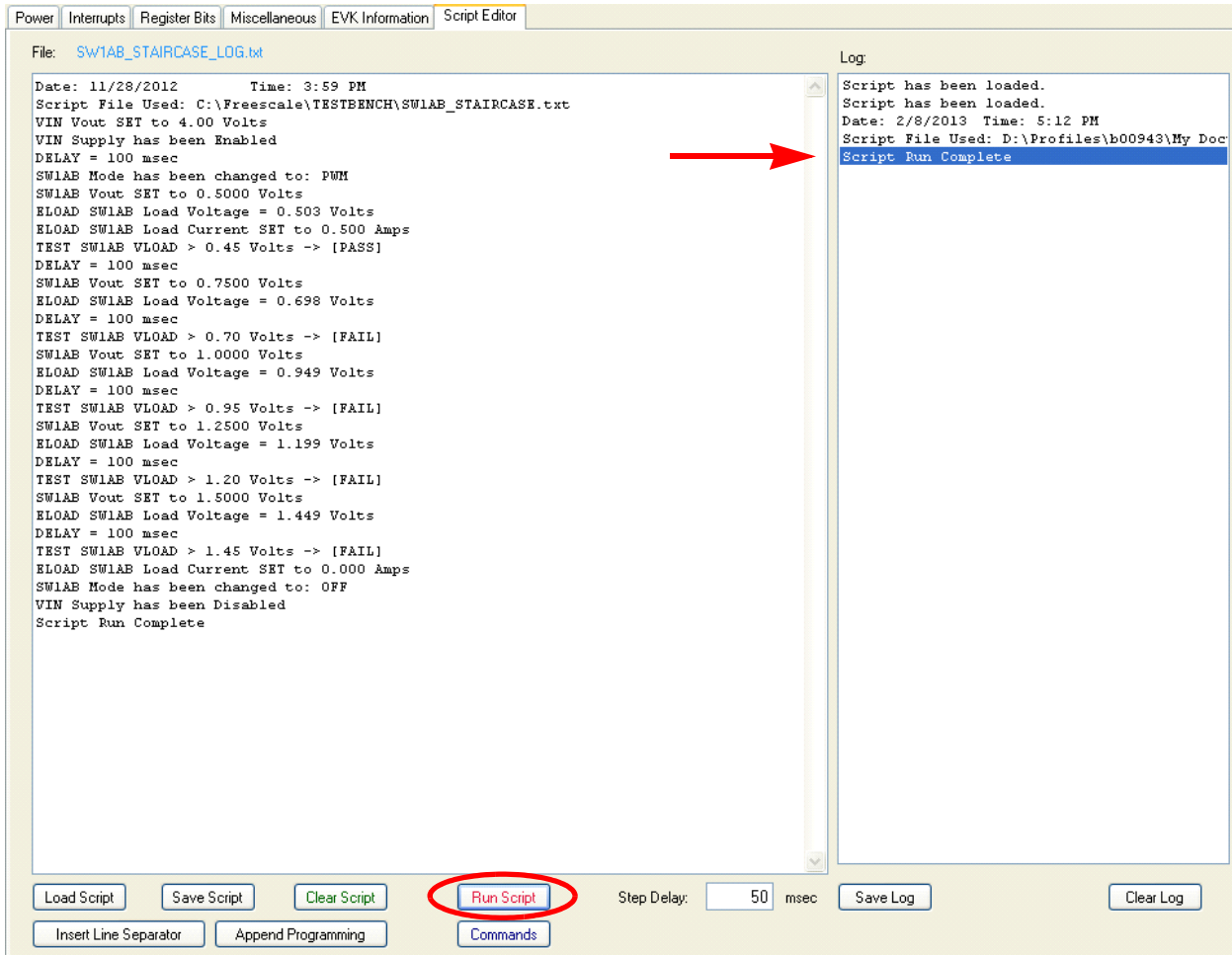


Figure 29. Running the Script

6.4.2 Writing a New Script

To write your own scripts, begin by creating a comment header using the “Insert Line Separator” button.

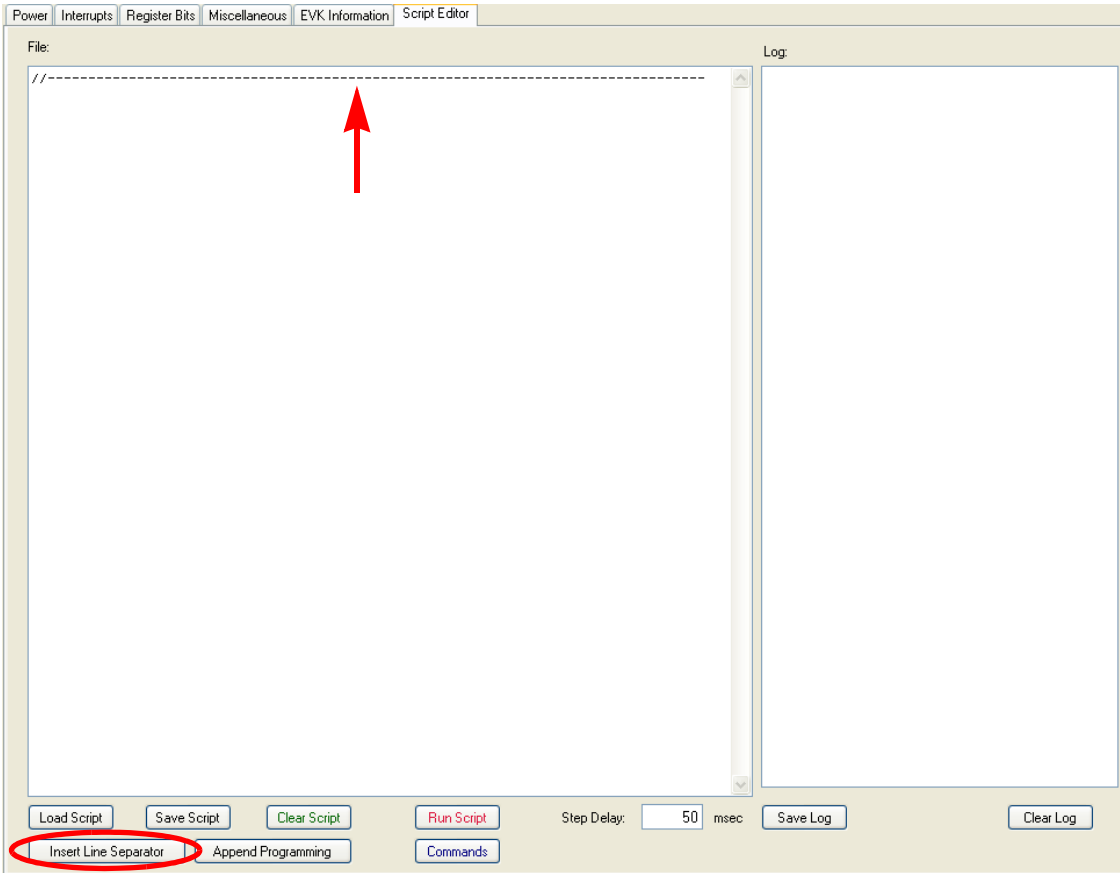


Figure 30. Inserting Line Separators

Proceed by manually writing the desired commands or use the “Command” button to display a graphical command selector in a new window as shown in Figure 31.

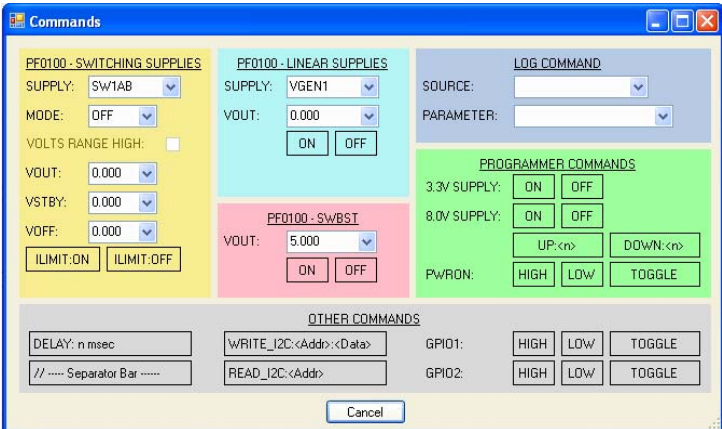


Figure 31. Command Selector Window

The “Command” window contains a set of commands that are useful for automatically sequencing the PF0100 power supplies, thereby emulating system behavior. The “Command” window is organized in six sections, as described by the following:

- PF0100 - SWITCHING SUPPLIES> place a single script command with the selected buck regulator and the desired function. The available functions are: mode selection, operating voltage setpoint, standby voltage setpoint, OFF voltage setpoint, enable current limit and disable current limit. Note that for SW2, SW3A/B and SW4, the VOLTS RANGE HIGH box need to be checked if the initial power up voltage is set to the high voltage operating range. If working in the lower voltage range, leave the box unchecked.
- PF0100 - LINEAR SUPPLIES> places a single script command with the selected LDO regulator and the desired function. The available functions are: operating voltage setpoint, enable and disable output.
- PF0100 - SWBST> Permits changing the operating voltage, as well as enabling and disabling the SWBST output.
- PROGRAMMER COMMANDS> Enables or disables the 3.3V supply as well as the ~8.0V boost supply on the KITPF0100EPEVBE. Allows a progressive step-up/step-down of the ~8.0V output and controls the general purpose output to set the PWRON terminal on the PF0100 high or set the same PWRON terminal low or toggle a pulse that triggers a PWRON event.
- OTHER COMMANDS> This section provides access to common instructions initiated by the control MCU. The possible commands include delay, add separator bar, generic I²C write/read, set GPIO1 and GPIO2 high, low or toggle.
- LOG COMMAND> provide a log report of the actual status of a specific configuration on the PMIC. Syntax for the log commands are shown in [Table 2](#).

6.4.2.1 Syntax and Command Set

Delimiters

- ':' - Is used as a separator
- '/' - Anything after a '/' will be ignored.
- White spaces will be truncated.

Table 2. Command List⁽¹⁾

Command	Description
WRITE_I2C:<Addr>:<Data>	Sends <Data> to I ² C register <Addr>. ⁽²⁾
READ_I2C:<Addr>	Reads the value of <Addr> and displays it in the Script Log. ⁽²⁾
VPGM:ON	Enables the 8.0 V OTP programming supply.
VPGM:OFF	Disables the 8.0 V OTP programming supply.
VPGM:UP:<n>	Increases the OTP programming voltage (VPGM) in <n> DAC steps.
VPGM:DOWN:<n>	Decreases the OTP programming voltage (VPGM) in <n> DAC steps.
V3V3:ON	Enables the 3.3 V system supply.
V3V3:OFF	Disables the 3.3 V system supply.
PWRON:HIGH	Releases the PWRON signal to a high-impedance state, allowing the PF0100 to start up.
PWRON:LOW	Asserts the PWRON signal LOW, forcing the PF0100 to shutdown.
PWRON:TOGGLE	Asserts the PWRON signal LOW, and then releases it to a high-impedance state, generating a power on event on the PF0100.
DELAY:<value>	Adds delay between script commands. Note that delays are cumulative with the Script Delay set on the Editor. delay is set in ms.
GPIO1:HIGH	Releases the GPIO1 signal to a high-impedance state.
GPIO1:LOW	Asserts the GPIO1 signal LOW.
GPIO1:TOGGLE	Asserts the GPIO1 signal LOW, and then releases it to a high-impedance state.
GPIO2:HIGH	Releases the GPIO2 signal to a high-impedance state.
GPIO2:LOW	Asserts the GPIO2 signal LOW.
GPIO2:TOGGLE	Asserts the GPIO2 signal LOW, and then releases it to a high-impedance state.
SW1x:MODE:<operator>	Sets the mode of operation of the SW1x regulator. The valid operators are as follows: <ul style="list-style-type: none"> • OFF • PFM • PWM • APS
SW1x:VOUT:<value>	Sets the SW1x output voltage in normal operation. Operating range from 0.300 V to 1.875 V in 0.025 V steps.
SW1x:VSTBY:<value>	Sets the SW1x output voltage to the STANDBY mode. Operating range from 0.300 V to 1.875 V in 0.025 V steps.
SW1x:OFF:<value>	Sets the SW1x output voltage to the OFF Mode. Operating range from 0.300 V to 1.875 V in 0.025 V steps.
SW1x:ILIM:<operator>	Enables/disables the SW1x current limit. Valid operators: <ul style="list-style-type: none"> • ON • OFF

Table 2. Command List⁽¹⁾ (continued)

Command	Description
SWx:MODE:<operator>	Sets the mode of operation of the SWx regulator. Following are valid operators: <ul style="list-style-type: none"> • OFF • PFM • PWM • APS
SWx:VOUT:<value>	Sets the SWx output voltage to normal operation. Full operating range from 0.300 V to 3.300 V divided into two operating ranges ⁽³⁾ : <ul style="list-style-type: none"> • Low Voltage Range > 0.300 V to 1.875 V in 0.025 V steps. • High voltage Range > 0.800 V to 3.300 V in 0.050 V steps.
SWx:VSTBY:<value>	Sets the SWx output voltage to the STANDBY mode. Full operating range from 0.300 V to 3.300 V divided in two operating ranges ⁽³⁾ : <ul style="list-style-type: none"> • Low Voltage Range > 0.300 V to 1.875 V in 0.025 V steps. • High voltage Range > 0.800 V to 3.300 V in 0.050 V steps.
SWx:OFF:<value>	Sets the SWx output voltage to the OFF mode. Full operating range from 0.300 V to 3.300 V divided into two operating ranges ⁽³⁾ : <ul style="list-style-type: none"> • Low Voltage Range > 0.300 V to 1.875 V in 0.025 V steps. • High voltage Range > 0.800 V to 3.300 V in 0.050 V steps.
SWx:ILIM:<operator>	Enables/disables the SWx current limit. Valid operators: <ul style="list-style-type: none"> • ON • OFF
SWBST:VOUT:<value>	Set the output voltage of the SWBST regulator. Valid output voltage: <ul style="list-style-type: none"> • 5.000 • 5.050 • 5.100 • 5.150
SWBST:ON	Enables SWBST regulator
SWBST:OFF	Disables SWBST regulator.
VGENx:ON	Enables the VGENx supply.
VGENx:OFF	Disables the VGENx supply.
VGENx:VOUT:<value>	Sets the output voltage for VGENx supply. <ul style="list-style-type: none"> • VGEN1/2 operating range: 0.800 V to 1.550 V with 50 mV steps. • VGEN3/4/5/6 operating range: 1.800 V to 3.3 V with 100 mV steps.
VREFDDR:ON	Enables the VREFDDR supply.
VREFDDR:OFF	Disables the VREFDDR supply.
VSNVS:ON	Enables the VSNVS supply
VSNVS:OFF	Disables the VSNVS supply.
PWRON:Float	Releases the PWRON signal to a high-impedance state, allowing the PF0100 to start up. (Legacy for Revision A scripts)

Table 2. Command List⁽¹⁾ (continued)

Command	Description
LOG Commands	
LOG:SWx:<log operator>	Shows the current value of the <log operator> for the SWx regulator. Log operators: <ul style="list-style-type: none"> • VOUT = Output voltage in normal operation. • STBY = Output voltage in STANDBY mode. • OFF = Output voltage in OFF mode. • MODE = Current switching mode set. • OTP_VOUT = Default power up voltage set through OTP. • OTP_SEQUENCE = Default power up sequence of regulator.
LOG:VGENx:<log operator>	Shows the current value of the <log operator> for the VGENx regulator. Log operators: <ul style="list-style-type: none"> • VOUT = Output voltage. • ENABLE = supply is ENABLED/DISABLED. • OTP_VOUT = Default power up voltage set through OTP. • OTP_SEQUENCE = Default power up sequence of regulator.
LOG:VSWBST:<log operator>	Shows the current value of the <log operator> for the VSWBST regulator. Log operators: <ul style="list-style-type: none"> • VOUT = Output voltage. • ENABLE = Supply is ENABLED/DISABLED. • OTP_VOUT = Default power up voltage set through OTP. • OTP_SEQUENCE = Default power up sequence of regulator.
LOG:VREFDDR:<log operator>	Shows the current value of the <log operator> for the VREFDDR regulator. Log operators: <ul style="list-style-type: none"> • ENABLE = supply is ENABLED/DISABLED. • OTP_SEQUENCE = Default power up sequence of regulator.
LOG:VSNVS:<log operator>	Shows the current value of the <log operator> for the VSNVS regulator. Log operators: <ul style="list-style-type: none"> • VOUT = Output voltage. • OTP_VOUT = Default power up voltage set through OTP.
LOG:OTP_PU_CONFIG:<log operator>	Shows the current value set as default by OTP. <ul style="list-style-type: none"> • SEQ_CLK_SPEED = programmed power up sequencing speed. • DVS_CLK_SPEED = programmed DVS speed. • PWRON_MODE = programmed PWRON pin active level. • PGOOD_ENABLE = Power good mode is on/off.

Table 2. Command List⁽¹⁾ (continued)

Command	Description
LOG:INT:<Int operator>	<p>Shows the status of the corresponding interrupt bit. Interrupt operators are as follows:</p> <ul style="list-style-type: none"> • 110_DEGREES • 120_DEGREES • 125_DEGREES • 130_DEGREES • SW1A_OVERCURRENT • SW1C_OVERCURRENT • SW2_OVERCURRENT • SW3A_OVERCURRENT • SW3B_OVERCURRENT • SW4_OVERCURRENT • SWBST_OVERCURRENT • VGEN1_OVERCURRENT • VGEN2_OVERCURRENT • VGEN3_OVERCURRENT • VGEN4_OVERCURRENT • VGEN5_OVERCURRENT • VGEN6_OVERCURRENT
<p>1. All characters (except for the "Float" subcommand) have to be entered in uppercase. 2. The register and data values should be entered as hexadecimal numbers, for example: 0x20 is entered as 20. 3. The output voltage operating range is set during OTP programming and cannot be changed under normal PMIC control.</p>	

Figure 32 shows a small sample script which enables the VGEN1 supply and reads the SW1AB Mode register. Run this script by pressing the “Run Script” Button. Note that the result for the READ_I2C:23 command shows up in the Script Log as 0x08.

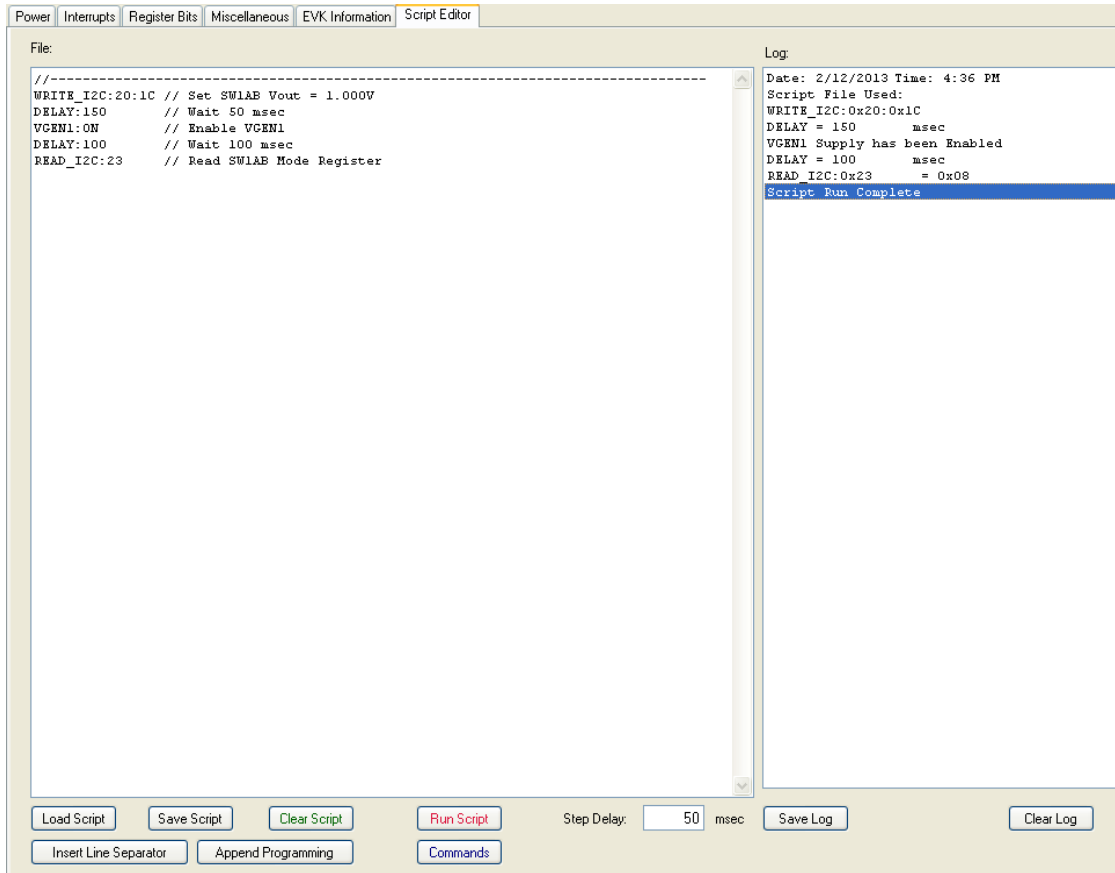


Figure 32. Sample Script

Confirm that the script has run correctly by checking the expected results in the GUI registers for the respective supplies.

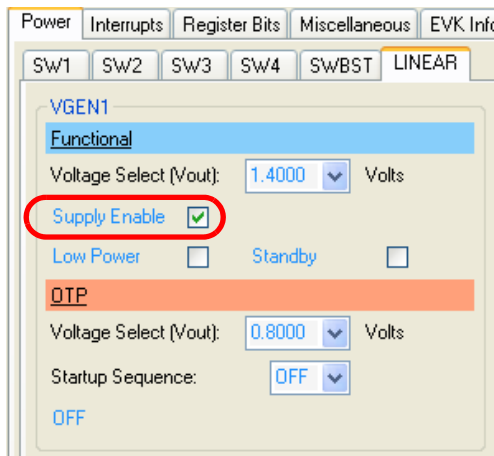
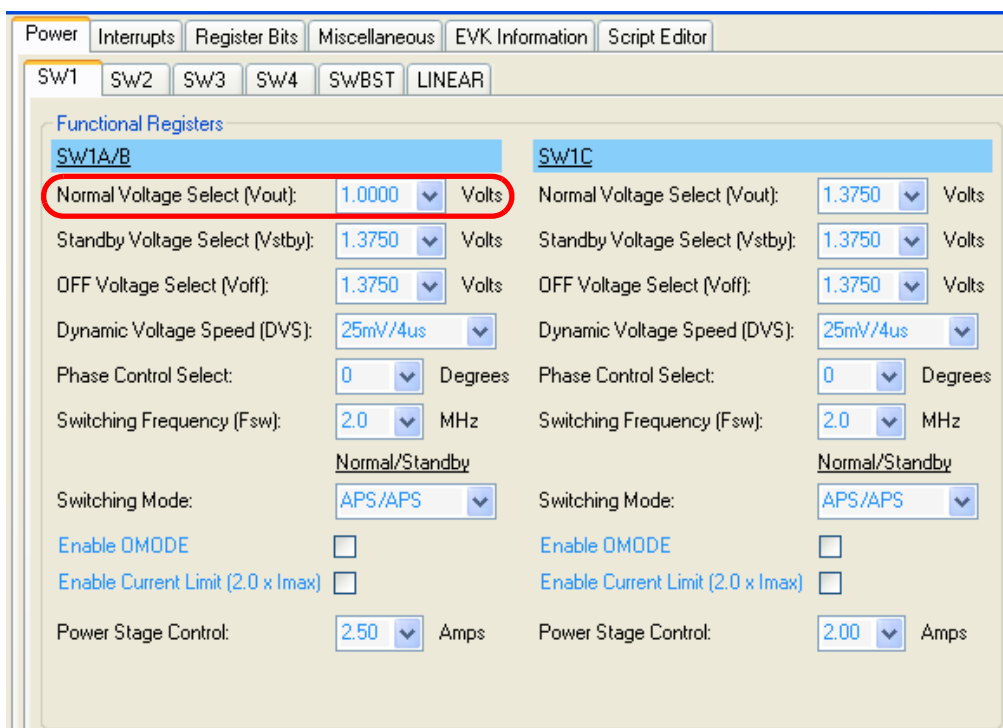


Figure 33. Verifying Script Results.

Power	Interrupts	Register Bits	Miscellaneous	EVK Information	Script Editor					
SW1	SW2	SW3	SW4 / SWBST	LINEAR 1	LINEAR 2	INTERRUPTS				
SW1AB										
0x20	D7	D6	D5	D4	D3	D2	D1	D0	VOUT	0x1C
	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		1.0000 Volts
	X	X	VOUT							
0x21	D7	D6	D5	D4	D3	D2	D1	D0	VSTBY	0x2B
	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		1.3750 Volts
	X	X	VSTBY							
0x22	D7	D6	D5	D4	D3	D2	D1	D0	VOFF	0x2B
	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		1.3750 Volts
	X	X	VOFF							
0x23	D7	D6	D5	D4	D3	D2	D1	D0	MODE	0x08
	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		APS/APS
	X	X	OM	X	MODE					
0x24	D7	D6	D5	D4	D3	D2	D1	D0	CONFIG	0x44
	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		D: 25mV/4us
	DVS		PHASE		FSW		X	I		P: 0 °
										F: 2.0 MHz

Figure 34. Verifying SW1 Mode Read Command

Note that the 0x08 SW1AB Mode value in the Script Log (Figure 32) correlates with that in the register bits, as shown directly above.

Finally, save the script so that it can be used again. Press the “Save Script” Button, then the “Save File” dialog box will appear. Enter the desired script file name, including the .txt file extension, then press the “Save” Button.

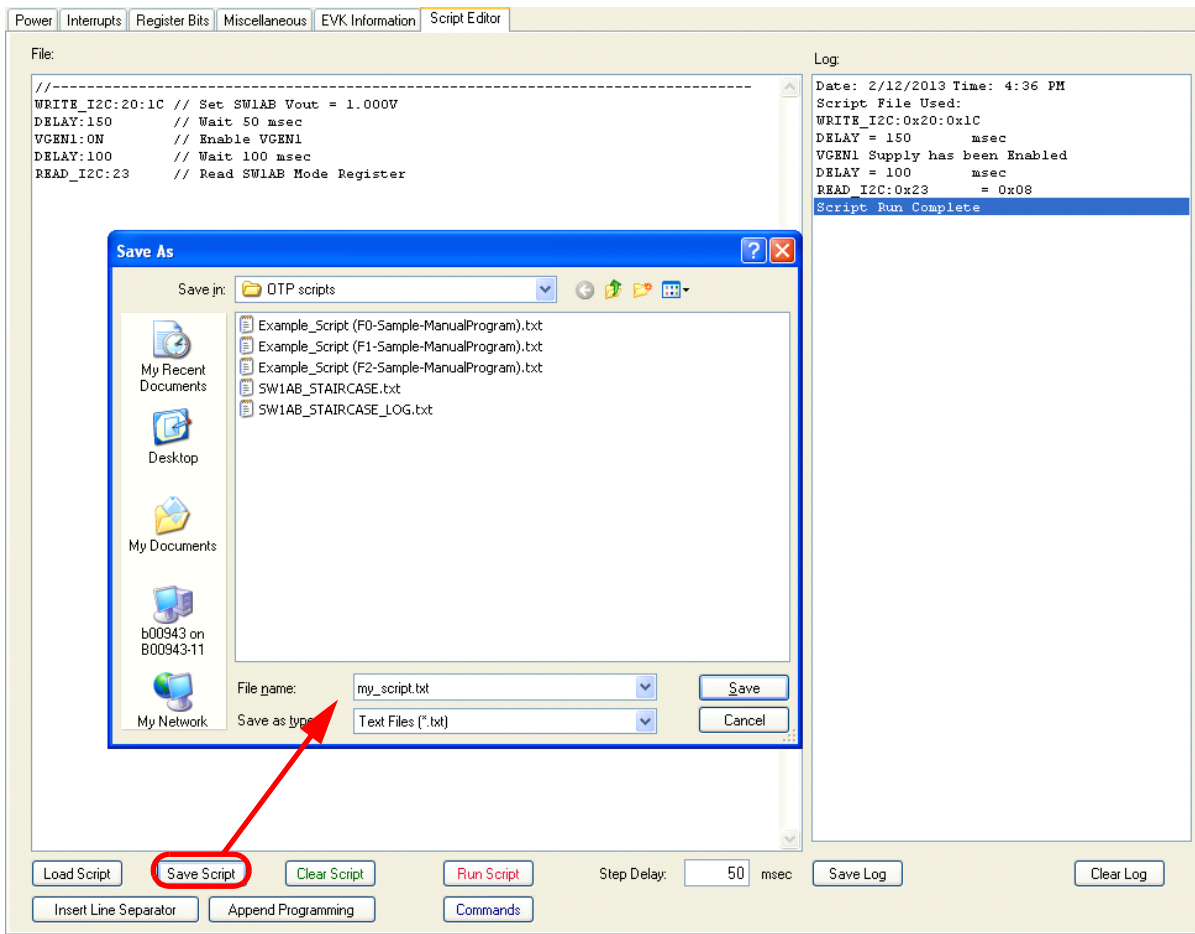


Figure 35. Saving a Script

6.5 Loading a Configuration File

A configuration file is a .txt file which contains specific OTP configuration instructions ready to be loaded and programmed into the OTP memory, thereby setting up a definitive power-up configuration for the PMIC. To load a configuration script file, press the “Load Configuration” button. An “Open File” dialog box will appear so that you can browse for and select your desired script file. Once you have selected the file you want, press the “Open” button.

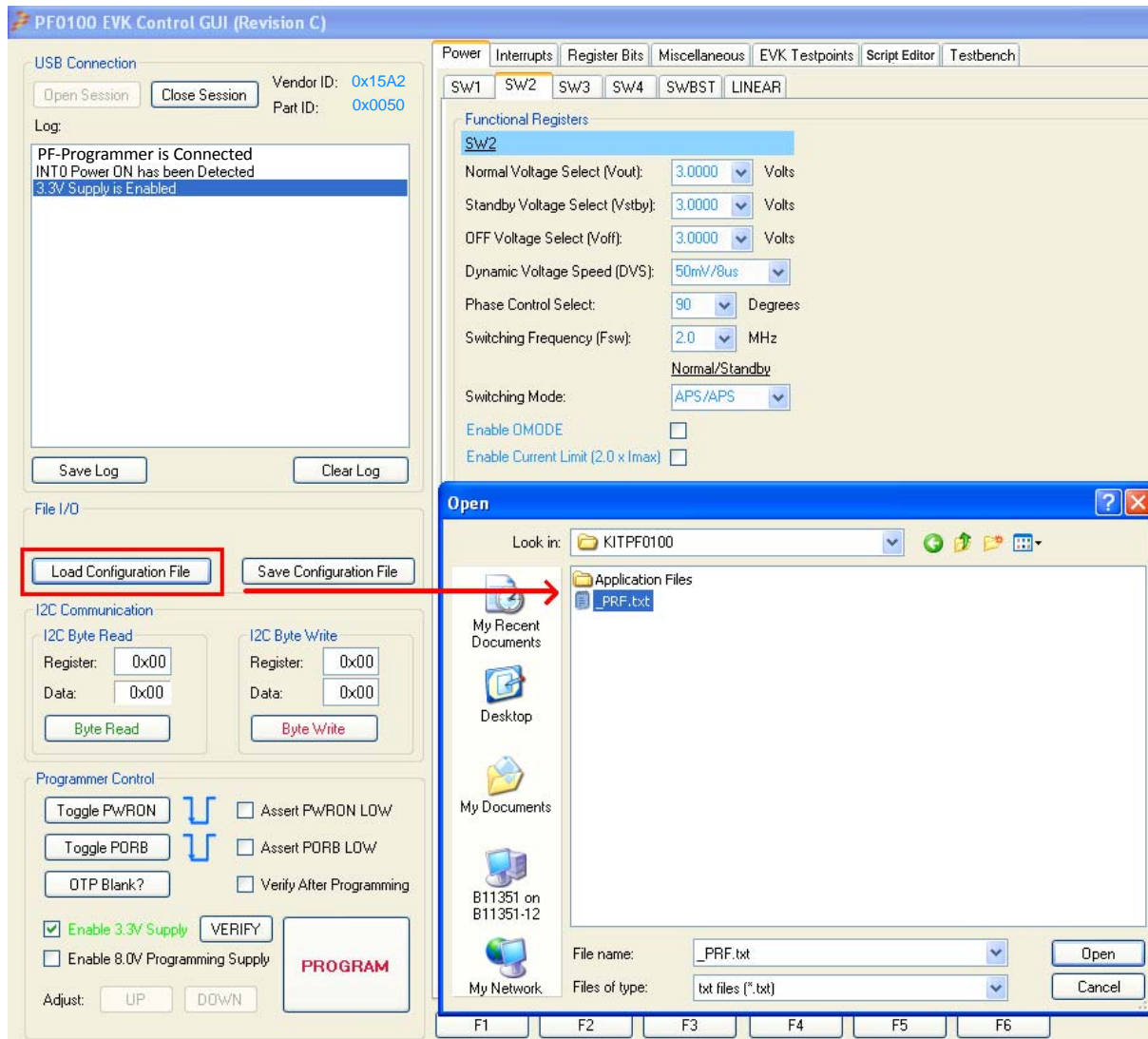


Figure 36. Loading A Script

When the file has loaded, an entry in the Log List will be made, and you should see the selected file displayed in the File I/O box.

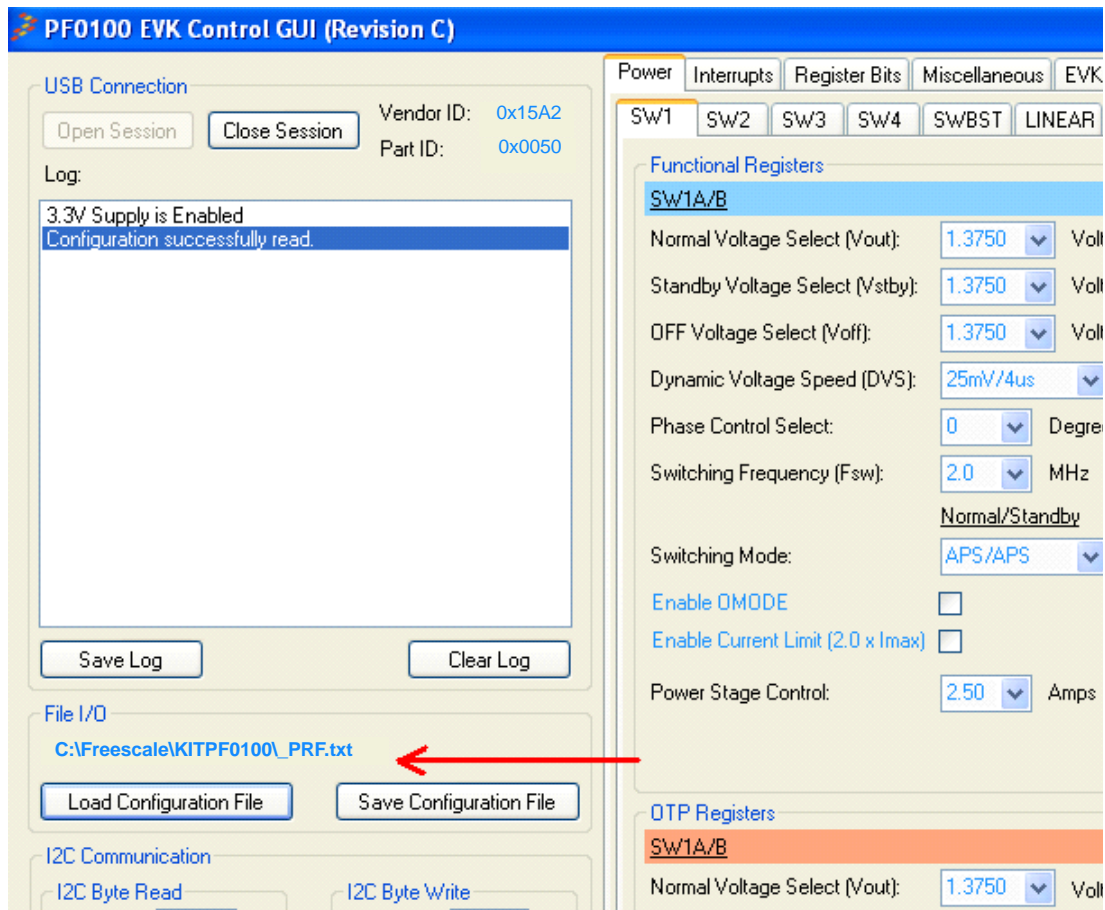


Figure 37. Scripting Tab

The “Save Configuration” button can be used to extract the current values of the PF0100 OTP Extended Page 1 registers from 0xA0 to 0xFF and place them in a text file, creating a mirror image of the OTP configuration of the PMIC in use. Remember to add the .txt file extension.

6.6 OTP Programming with the KITPFGMEVME

The KITPFGMEVME allows programming the OTP fuses on the target board attached. At this point it might be a good idea to check if the PF0100 part to be programmed is blank (not programmed). This can be easily done by pressing the “OTP Blank?” button.

If you would like the KITPF0100GUI to verify that a part has been programmed with the OTP data supplied by the programming script, click the “Verify After Programming” check box. This will direct the GUI to do a line-by-line comparison between the data written and the data actually programmed into the part.

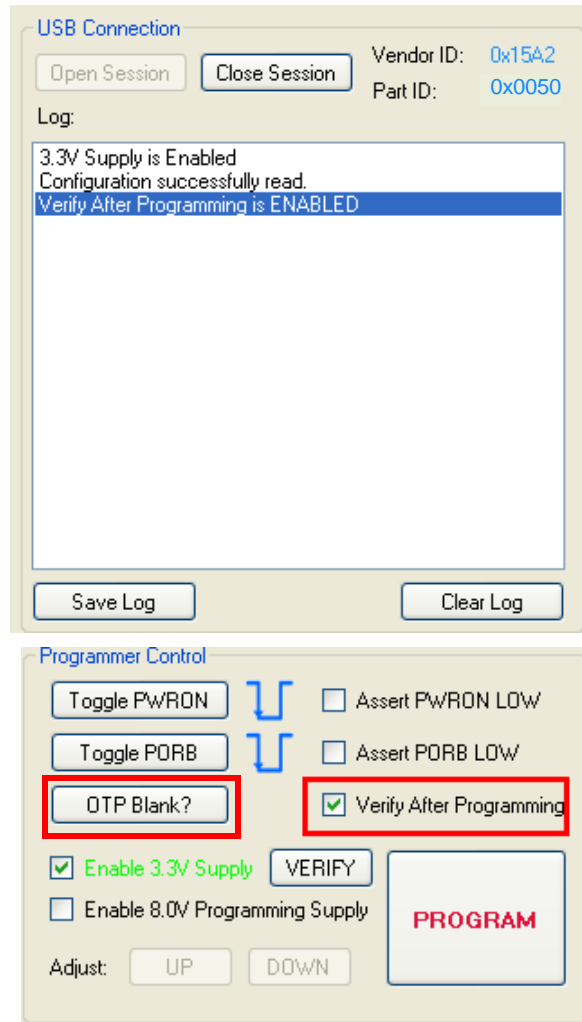


Figure 38. Verify After Programming and OTP Blank?

Then press the “PROGRAM” button. The KITPF0100GUI will verify that the script file has been loaded and the supply voltages have been applied before starting the programming sequence. Each step of the programming sequence will be displayed in the Log List.

When the programming sequence turns on the 8.0V VPGM programming supply, then the red LED of the KITPFGMEVME will be on.

After programming has been completed, the GUI will automatically check to see if the OTP memory has been programmed, and because we selected “Verify” afterwards, the verify algorithm will be run. The number of OTP programming errors will be reported in the Log List.

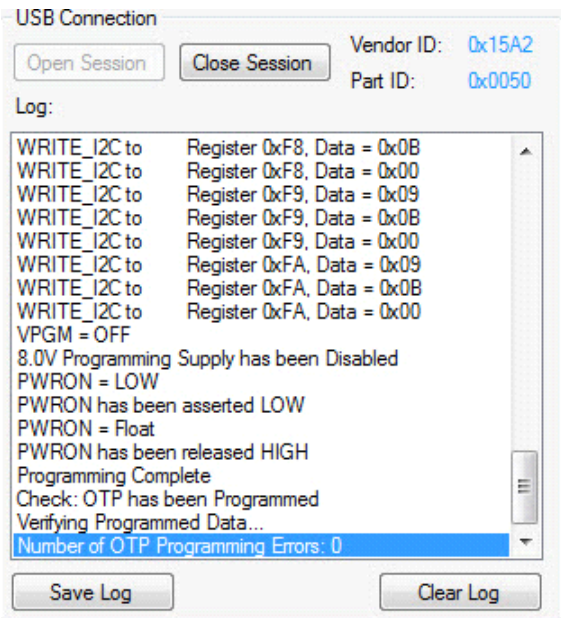


Figure 39. Scripting Log Session Example

7 References

Document Number	Description	URL
PF Programmer		
	Tool Summary Page	http://www.freescale.com/KITPFGMEVME
MMPF0100 PMIC		
MMPF0100	Data Sheet	http://cache.freescale.com/files/analog/doc/data_sheet/MMPF0100.pdf
MMPF0100ER	Errata	http://cache.freescale.com/files/analog/doc/errata/MMPF0100ER.pdf
PFSERIESFS	Fact Sheet	http://cache.freescale.com/files/analog/doc/fact_sheet/PFSeriesFS.pdf
AN4622	Layout Application Note	http://cache.freescale.com/files/analog/doc/app_note/AN4622.pdf
	Product Summary Page	http://www.freescale.com/MMPF0100
Other		
	Power Management Home Page	http://www.freescale.com/PMIC
	Analog Home Page	http://www.freescale.com/analog

7.1 Support

Visit [Freescale.com/support](http://www.freescale.com/support) for a list of phone numbers within your region.

7.2 Warranty

Visit [Freescale.com/warranty](http://www.freescale.com/warranty) for a list of phone numbers within your region.

8 Revision History

Revision	Date	Description of Changes
1.0	12/2012	<ul style="list-style-type: none"> Initial Release
2.0	2/2013	<ul style="list-style-type: none"> Updated document for the latest GUI Revision 3.0.0.20 Added TBB operation Mode. Updated section 6.4 Using the Script Editor Updated section 6.5 Loading a Configuration File
3.0	1/2015	<ul style="list-style-type: none"> Provided link to the upgraded version of this user guide

How to Reach Us:

Home Page:
freescale.com

Web Support:
freescale.com/support

Information in this document is provided solely to enable system and software implementers to use Freescale products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document.

Freescale reserves the right to make changes without further notice to any products herein. Freescale makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters that may be provided in Freescale data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including "typicals," must be validated for each customer application by customer's technical experts. Freescale does not convey any license under its patent rights nor the rights of others. Freescale sells products pursuant to standard terms and conditions of sale, which can be found at the following address: freescale.com/SalesTermsandConditions.

Freescale and the Freescale logo are trademarks of Freescale Semiconductor, Inc., Reg. U.S. Pat. & Tm. Off. All other product or service names are the property of their respective owners.

© 2015 Freescale Semiconductor, Inc.