Mixed Signal Power Manager for SmartFusion2 Reference Design

User's Guide

Microsemi

Table of Contents

1 – Introduction

This document explains how to use the mixed signal power manager (MPM) reference design for SmartFusion®2 system-on-chip (SoC) field programmable gate array (FPGA) device, by using the SmartFusion2 Digital MPM daughter board (SF2-DMPM-DB) that is connected to the SmartFusion2 Development Kit board. This assumes the use of SF2 MPM 6.1.100 firmware or a later version.

Refer to the *SmartFusion2 Development Kit User's Guide* for more information on the Development Kit board.

MPM is a reference design that is programmed into the SmartFusion2 SoC FPGA device and that can be controlled and configured by the MPM graphical user interface (GUI) through a standard 2-wire Philips Inter-Integrated Circuit (I2C) interface.

Based on the SmartFusion2 device, MPM delivers superior power monitoring, power sequencing, closed-loop trimming, and power-up and power-down control of up to 64 power supplies, which can be a mix of analog points of loads (APOLs) and digital points of loads (DPOLs). This adds more flexibility, reduces total parts count on the board level, and increases system reliability by eliminating single points of failure.

It is not required to use FPGA design tools to configure power management sequencing, levels, or thresholds; the MPM design is programmed into the device through an easy-to-use standalone GUI tool. The GUI enables to configure the power management and drive output signals as the monitored voltages meet or deviate from the user-programmed operating limits, all without opening Libero® System-on-Chip (SoC) tools.

The MPM GUI tool writes register values to on-chip embedded flash memory, which control the power sequencing and monitoring functionality of the MPM reference design. The MPM GUI tool programs the board and configuration settings by launching the FlashPro software, which communicates with the target over a USB port that is connected to an on-board or standalone FlashPro programmer device. In addition, the MPM GUI can configure, control, and monitor the MPM target over I2C using an appropriate USB to I2C dongle.

This document assumes some knowledge of MPM or similar application-specific standard product (ASSP) applications.

Note: All the images in this revision of the document are based on development boards and are not representative of the release hardware. This document is currently based on the Rev A SF2-DMPM-DB but will note any differences for the Rev B SF2-DMPM-DB.

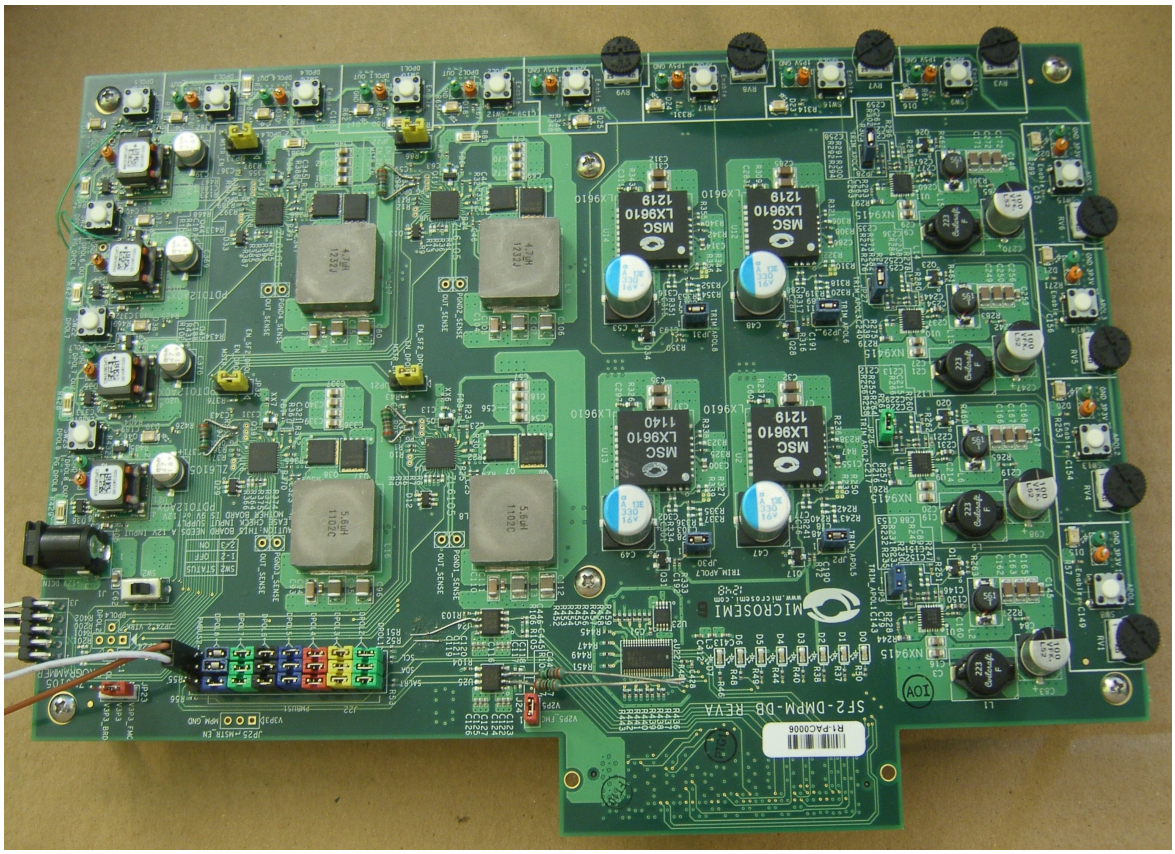

Figure 1-1 **• SmartFusion2 Daughter Board - Revision A**

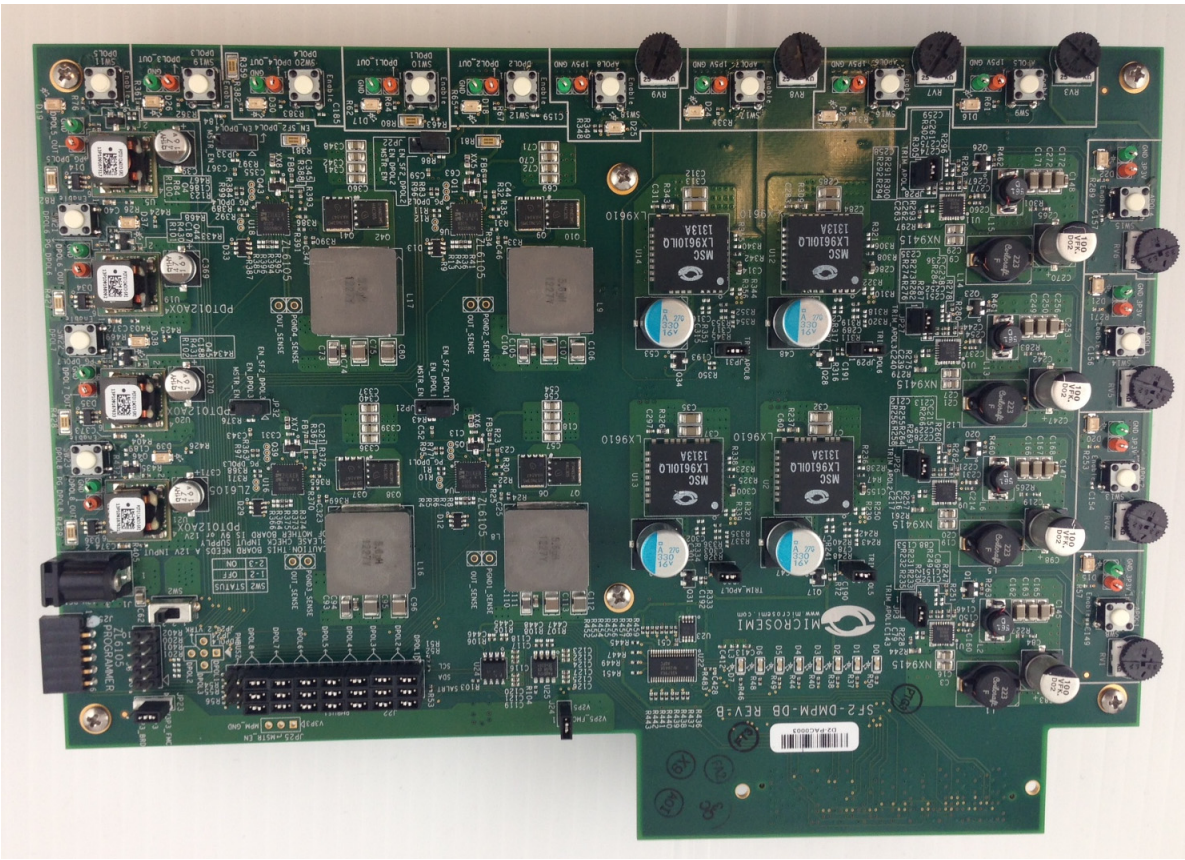

***Figure 1-2* • SmartFusion2 Daughter Board - Revision B**

2 – Using the Mixed Signal Power Manager

Introduction

Run the installer and follow the installation wizard instructions. By default the MPM installs into the **C:\Microsemi\SF2_MPM_RefDesign_v6.1** folder; and this is the recommended location. Avoid installing it to a folder that is very deeply nested or has a very long path name, some tools may encounter problems accessing files with names longer than 259 characters1.

Once installed, the MPM adds the following options to the Windows start menu: Start > All Programs > Microsemi SmartFusion2 MPM Reference Design v6.1

- Start
- All Programs
  - Microsemi SmartFusion2 MPM Reference Design v6.1
    - Browse Desizzgn Files - Opens in Windows Explorer the folder containing the MPM Libero SoC hardware and SoftConsole firmware projects
    - MPM GUI - Runs the MPM GUI
    - Uninstall - Uninstalls MPM

Hardware Setup

This section explains how to prepare the hardware for programming the MPM reference design and running the default demonstration.

The following tools are required:

- SmartFusion2 Development Kit Board
- SF2 Digital MPM Daughter Board (SF2-DMPM-DB)
- Devantech/Robot Electronics USB-ISS communications module

Note: Refer to this Microsoft MSDN article for more information on path length issues: http://msdn.microsoft.com/en-us/library/aa365247.aspx.

SmartFusion2 Development Kit Board Setup

The following section provides the basic setup information on setting-up the hardware and firmware for the DMPM 6.1.100 release. The setup is based on the initial porting of MPM to the SmartFusion2 platform with the Rev B Dev Kit and the Rev A SF2-DMPM-DB[1].

Before connecting power to the MPM system, ensure that the following jumpers are installed:

1. *If a Rev B SF2-DMPM-DB board is being used, J183 needs to be jumpered on pins 2 and 3. Remove the jumper lead between J145 and J199.*

Dev Kit Setup

The following jumper settings are required on the SmartFusion2 Development Kit board to route the signals required from the DMPM board to the FMC connector:

***Table 2-1* • Jumper Settings on the SmartFusion2 Development Kit Board**

J174 pins 2 and 3 jumpered	J172 pins 2 and 3 jumpered	J184 pins 2 and 3 jumpered
J175 pins 2 and 3 jumpered	J179 pins 2 and 3 jumpered	J195 pins 2 and 3 jumpered
J200 pins 2 and 3 jumpered	J194 pins 2 and 3 jumpered	J202 pins 2 and 3 jumpered
J210 pins 2 and 3 jumpered	J201 pins 2 and 3 jumpered	J209 pins 2 and 3 jumpered
J155 pins 2 and 3 jumpered	J146 pins 2 and 3 jumpered	J140 pins 2 and 3 jumpered
J138 pins 2 and 3 jumpered	J158 pins 2 and 3 jumpered	J154 pins 2 and 3 jumpered
J143 pins 2 and 3 jumpered	J141 pins 2 and 3 jumpered	J111 pins 2 and 3 jumpered
J133 pins 1 and 2 jumpered	J214 pins 2 and 3 jumpered	J213 pins 2 and 3 jumpered
J178 pins 2 and 3 jumpered	J188 pins 2 and 3 jumpered	J187 pins 2 and 3 jumpered
J197 pins 2 and 3 jumpered	J196 pins 2 and 3 jumpered	

J145 pin 2 connected through flying lead to J199 pin 3.

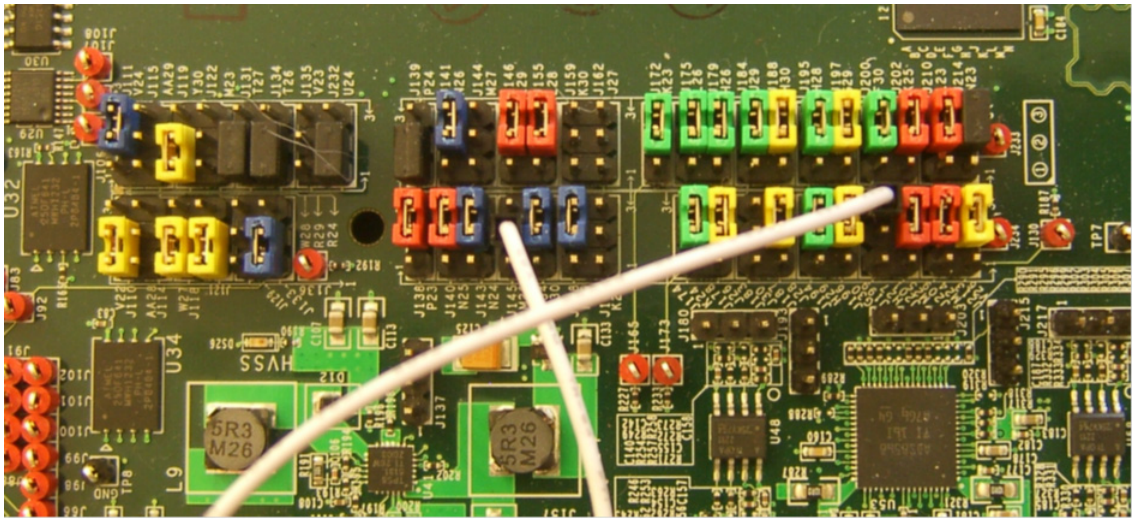

***Figure 2-1* • SmartFusion2 Development Kit Jumpers**

The DIP switch SW10 is used to provide four of the MPM inputs and also to select the optional operational modes for the MPM.

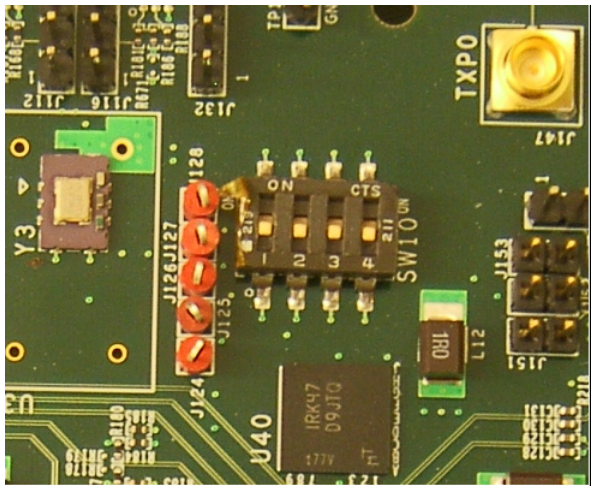

***Figure 2-2* • SmartFusion2 Development Kit SW10**

Table 2-2 explains the use of the SW10 DIP switches:

***Table 2-2* • SW10 DIP Switches**

DIP Switch	Description	Recommended Default Setting
SW10.1	Controls whether or not MPM automatically initiates power off sequencing when any configured channel deviates from Nominal to a critical state (OFF, UV2 or OV2). Can be changed on the fly while MPM is running. OFF: disable auto power off sequencing. ON: enable auto power off sequencing.	Whichever is preferred for demonstration purposes.
SW10.2	Controls whether or not MPM loads a fixed set of known good demo configuration settings superseding any configuration settings stored in ENVM by the GUI/I2C. OFF: don't load fixed known good demo configuration settings - load configuration settings stored in ENVM via GUI/I2C. ON: load fixed known good demo configuration settings.	OFF
Notes: If a Rev B SF2-DMPM-DB board is being used, the following changes apply: • *J183 needs to be jumpered on pins 2 and 3.* • *Remove the jumper lead between J145 and J199.*		

***Table 2-2* • SW10 DIP Switches (continued)**

DIP Switch	Description	Recommended Default Setting
SW10.3	Along with SW10.4 controls what DEADTIME setting to use for Intersil DPOLs. OFF: don't load any specific DEADTIME configuration setting - use whatever setting the DPOL was previously programmed with. ON: load the DEADTIME configuration setting indicated by the SW10.4 setting.	ON
SW10.4	If SW10.3 is ON then SW10.4 controls which DEADTIME setting to use for Intersil DPOLs, otherwise SW10.4 is ignored. OFF: 40 nanoseconds ON: 24 nanoseconds	ON
Notes: If a Rev B SF2-DMPM-DB board is being used, the following changes apply: • *J183 needs to be jumpered on pins 2 and 3.* • *Remove the jumper lead between J145 and J199.*		

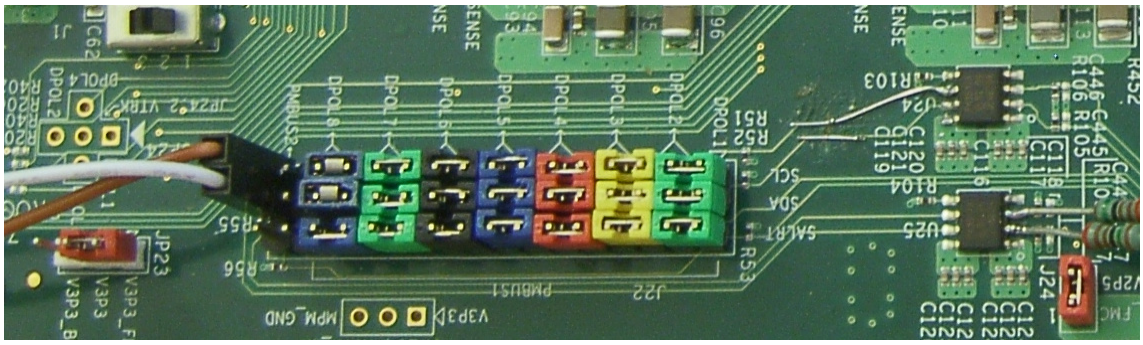

***Figure 2-3* • SF2 DMPM Daughter Board Setup**

SF2 DMPM Daughter Board Setup

JP23 should have pins 1 and 2 jumpered and the J24 pin should be jumpered to provide 3.3 V and 2.5 V supplies to the SF2-DMPM-DB board from the FMC connector.

J22 should have 7 x 3 jumpers installed, as shown in Figure 2-5, to link the eight DPOLs to PMBUS1. The last pair of pins labeled PMBUS2 are unjumpered but are used to connect the Devantech USB Adaptor, as shown in Figure 2-6. A suitable ground connection for this can be found on J3 - second pin from the left on the bottom row.

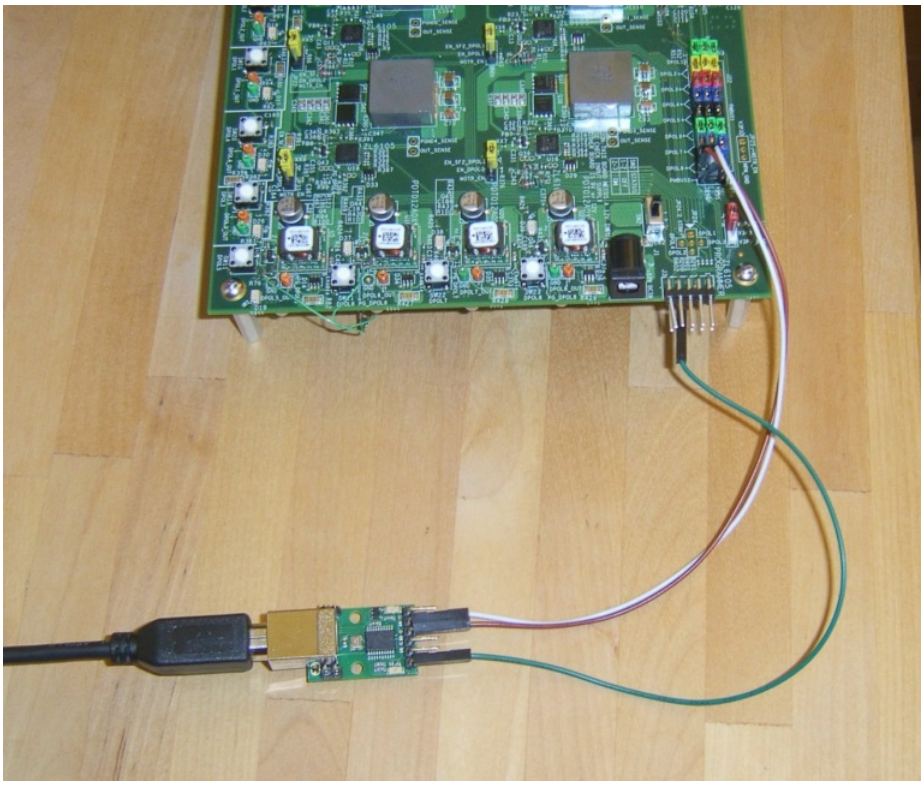

***Figure 2-4* • SF2-DMPM-DB Rev A Slave I2C Connection**

SF2-DMPM-DB Rev B boards have a dedicated connector for the Devantech USB Adaptor (J25).

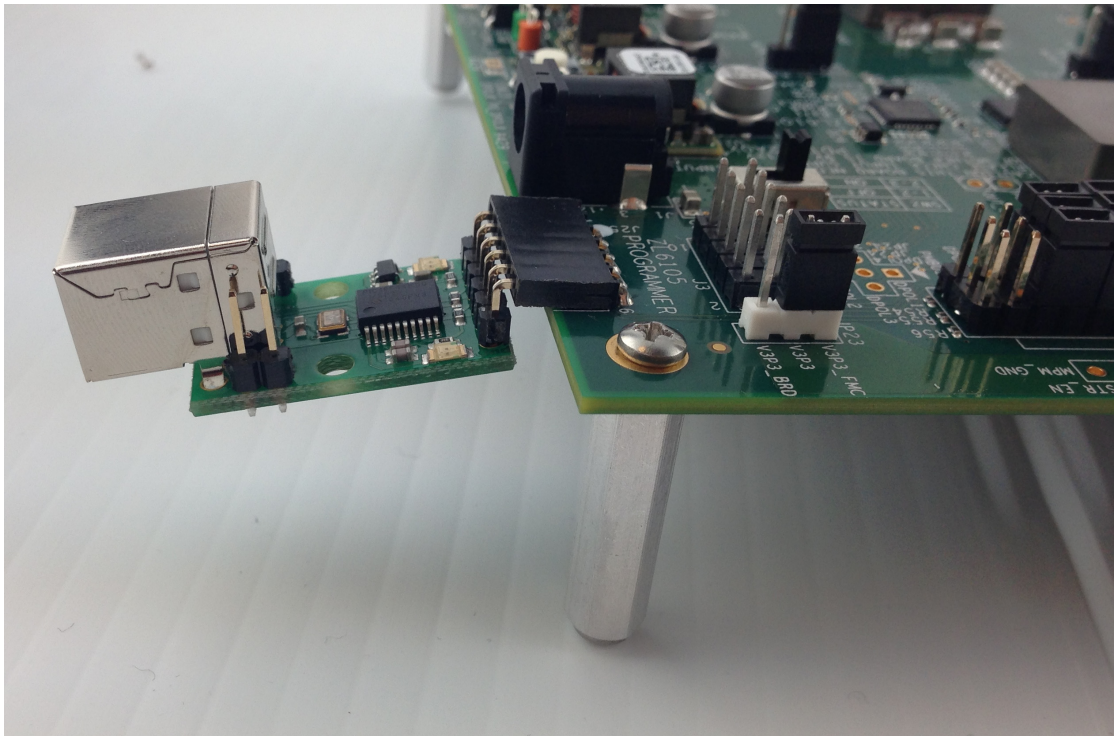

***Figure 2-5* • SF2-DMPM-DB Rev B Slave I2C Connection**

For the APOLs on the SF2 DMPM-DB: JP3, JP26, JP27, JP28, JP2, JP29, JP30, and JP31 must be jumpered to enable trimming.

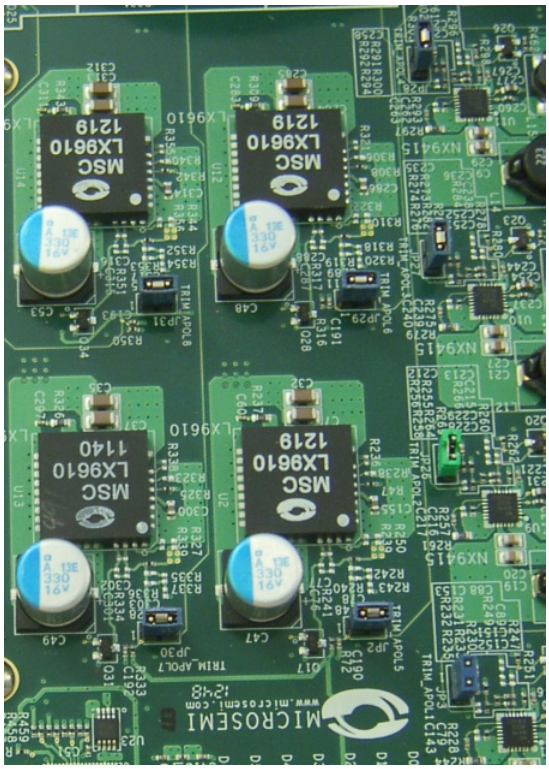

***Figure 2-6* • SF2-DMPM-DB Trim Jumpers**

For the DPOLs on the SF2-DMPM-DB board: JP21, JP22, JP32, and JP33 must be jumpered on pins 1 and 2 to route the regulator enables of the first 4 DPOLs to the FMC connector.

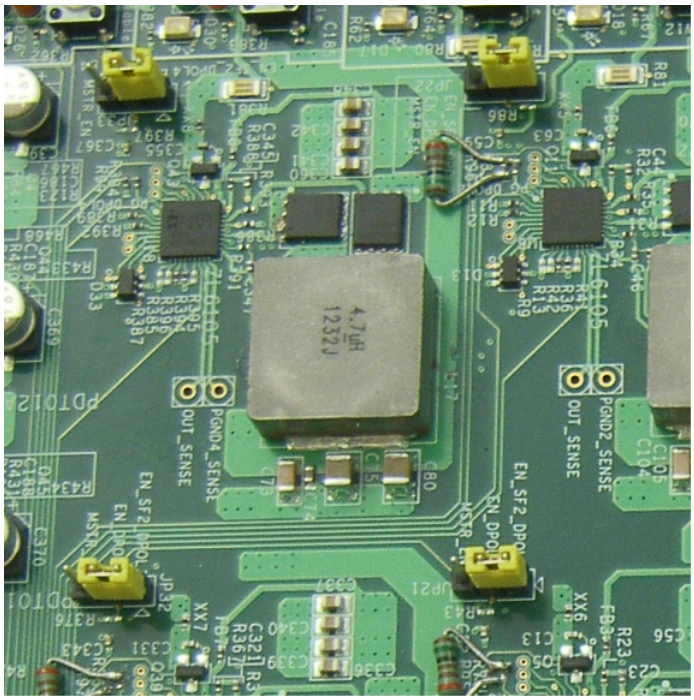

***Figure 2-7* • SF2-DMPM-DB DPOL Enable Jumpers**

Programming the Mixed Signal Power Manager

To use the MPM for the first time, follow the steps below to program the MPM design to the target hardware.

1. To set-up the SmartFusion2 Development Kit board:
   a. Connect the 12 V 6A power supply to the J18 12V INPUT SUPPLY connector
   b. Connect a FlashPro4 programmer to the J59 - FP4 Header
   c. Connect a mini USB cable between the PC and the FlashPro4
   d. Power-on the board using SW7 the board on using SW7
2. To run the MPM GUI:
   a. Select Data > FlashPro > Choose STAPL template and select the appropriate STAPL file for the target hardware from the C:\Microsemi\SF2_MPM_RefDesign_v6.1\template folder.
   b. Select Data > FlashPro > FlashPro Setup and browse to select the FlashPro software executable in the Libero SoC v1.1 or a later version.
   c. Select Data > FlashPro > Write NVM and Fabric, and the MPM GUI will launch the FlashPro and program the full design (MSS configuration, MPM firmware, MPM configuration data in ENVM along with the FPGA fabric logic) to the target hardware. The command shell "DOS Box" will be displayed showing the progress and can be closed when prompted. When prompted, close this. At this stage, the MPM target should be programmed with the MPM reference design.
3. Power-off the SmartFusion2 Development Kit board by disconnecting the USB cable and power supply cables.
4. Connect the SF2-DMPM-DB to the SmartFusion2 Development Kit board by connecting their respective FMC connectors.
5. Connect the 12 V power supply to the SF2-DMPM-DB J1 12V DC IN connector

6. Reconnect power and USB to the SmartFusion2 Dev Kit board.
7. Power the SF2-DMPM-DB on using SW2
8. Power on the SmartFusion2 Development Kit board.
9. Reset the SmartFusion2 Development Kit board by pressing SW9 RESET.
10. Once the initial programming of the MPM design has been performed, the MPM configuration setting changes can be programmed using **Data > I2C > Read Config Via I2C and Write Config Via I2C**.

I2C Setup

Use the MPM GUI to connect to the MPM target through I2C with the included Devantech/Robot Electronics USB-ISS communications module. Ensure that the USB-ISS Power Link has a jumper removed for 3.3 V operations for compatibility with the MPM target.

Use the included standard USB A/B cable to connect the USB-ISS to the PC. Install the drivers that are bundled with MPM GUI; install in the **C:\Microsemi\SF2_MPM_RefDesign_v6.1\Devantech_USB-ISS_drivers** folder.

After installing the drivers, and plugging the USB-ISS module into a spare USB port, confirm the COM port to where it has been assigned to. This will vary depending on the number of COM ports installed. To verify this:

1. Go to **Computer>Control Panel>Hardware and select Device Manager**.
2. Scroll-down and open the **Ports (COM & LPT)** tab. The USB serial port is listed - COM5, as shown in the example below.
3. To change the COM port number, right-click on it, select **Properties > Advanced > Port Settings > COM port number** from the available list. It is not required to change the COM port default settings.

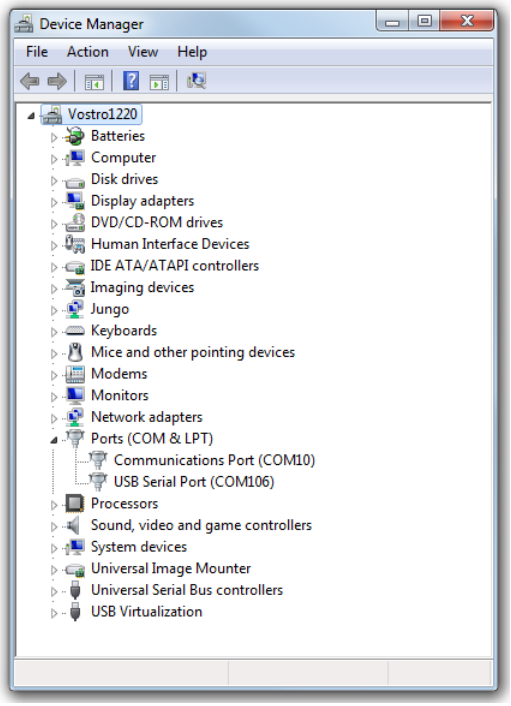

Figure 2-8 • **Device Manager**

To allow the MPM GUI to communicate with the MPM device manager, then connect the Devantech USB-ISS to MSS I2C_1 by installing female to female jumper cables between the following pins:

***Table 2-3* • Devantech/Robot Electronics USB-ISS Connections**

I²C Signal Board	**SCL**	**SDA**	**GND**
Devantech/Robot Electronics USB-ISS	SCL (I/O 3)	SDA (I/O 4)	0V/Ground
SF2 DMPM Board Rev A	J22P pin 16	J22P pin 32	3 pin 8

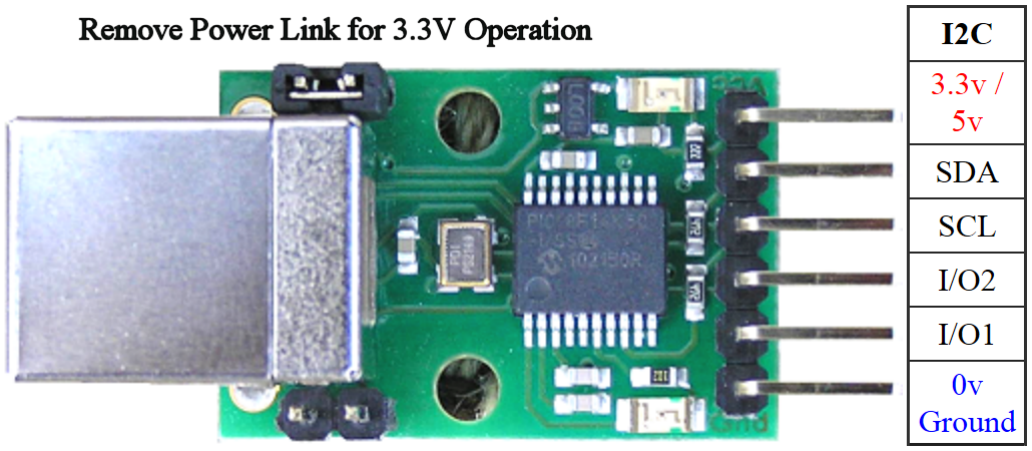

***Figure 2-9* • Devantech/Robot Electronics USB-ISS**

As explained above, note that the COM port is assigned to the USB-ISS communications module.

1. Run the MPM GUI, select **Data > I²C > Test I²C Dongle**.
2. Select the **USB-ISS COM** port from the **Communications > Port** drop-down list.
3. Select **Test Dongle** from the **Activity > Action** drop-down list and click Go. If the test is successful, the follwing message is displayed in the **Status** text-box.

```
Starting 'Test of Communications with Dongle'
I2C Speed set to 1 MHz
Communicated with dongle
Firmware Version: 07 02 80
Serial No: [00001425]
Test of Communications with Dongle Completed
If the test fails then you will see this and you need to review and correct the setup:
Starting 'Test of Communications with Dongle'
Failed to open I2C port
```

MPM Reference Design Demo

Without the MPM GUI

Once the MPM reference design is programmed to the target hardware, the demo can be viewed even without the MPM GUI, by using the SW3 on the Dev Kit to initiate power-up and power-down sequences and observing the current state of MPM on the Dev Kit light-emitting diodes (LEDs).

The central button (SW3) is used to start the MPM sequencing. If the MPM system is off, press **SW3** to start sequencing the POLs up. If the MPM system is fully started, press **SW3** to start sequencing the POLs down.

The remaining push buttons are connected to the MPM inputs and can be used to demonstrate the input functionality:

SW1 is connected to MPM input 1.

SW2 is connected to MPM input 2.

SW4 is connected to MPM input 3.

SW5 is connected to MPM input 4.

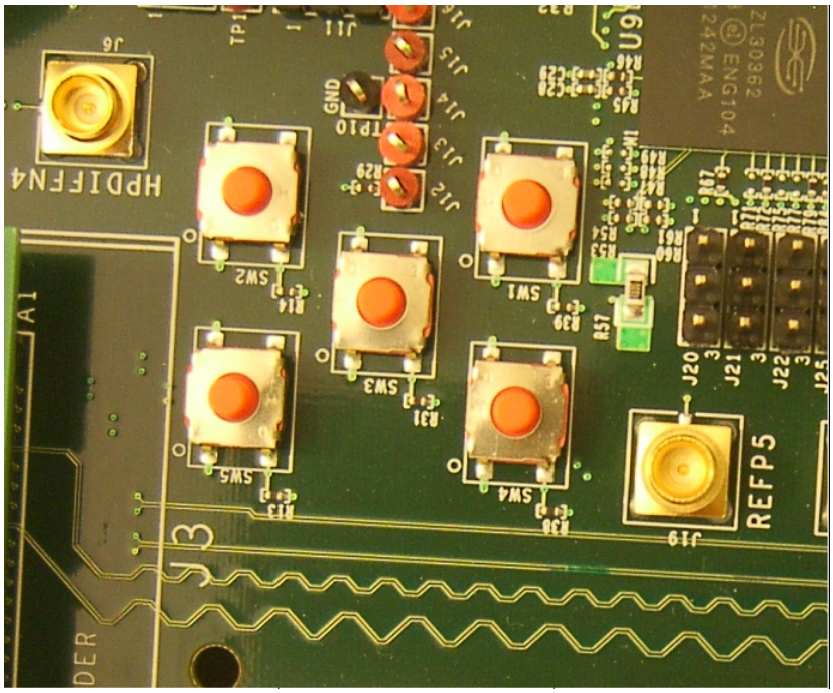

***Figure 2-10* • SF2 Dev Kit SW1 to SW5**

The first four of the eight LEDs on the Development Kit board are used to display the current state of the MPM system as shown below:

LED 1 - On, when MPM is in the off state.

LED 2 - On, when MPM is sequencing power-up.

LED 3 - On, when all POLs have sequenced on.

LED 4 - On, when MPM is sequencing power-down.

LEDs 5 to 7 are currently toggled by the `mpm_threshold_task()`, `mpm_i2c_slave_task()` and `mpm_timer2_task()` every time they run and provide visual indication of MPM activity.

LED 8 pulses once if there is a PMBus I2C timeout and twice if there is a PMBus error that provides a useful oscilloscope or logic analyzer trigger signal when debugging issues on the PMBus.

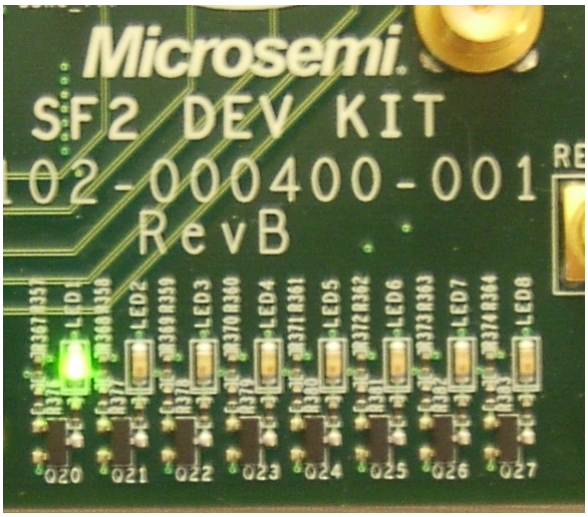

***Figure 2-11* • SF2 Development Kit Board LEDs**

***Table 2-4* • MPM Status Description**

Status	Description
Stopped	Power-off sequencing is successful and MPM is idle. None of the following are active: channel threshold monitoring, output flag generation, and open or closed trimming. Channel voltages can be read in any state.
Starting	Executing power-on sequencing during which open-loop trimming (if applicable), channel threshold monitoring, and output flag generation are active.
Started	Power sequencing is successful; MPM is now active and reading channel voltages on demand, monitoring channel thresholds, executing closed-loop trimming (if applicable), and generating output flags.
Stopping	Executing power-off sequencing before which closed-loop trimming (if applicable) is switched off but channel threshold monitoring and output flag generation remains operational.

The DIP switch, SW10 is used to provide another 4 of the MPM inputs and also to select optional operational modes for MPM.

- **SW10.1** is connected to MPM input 5
- **SW10.2** is connected to MPM input 6
- **SW10.3** is connected to MPM input 7
- **SW10.4** is connected to MPM input 8

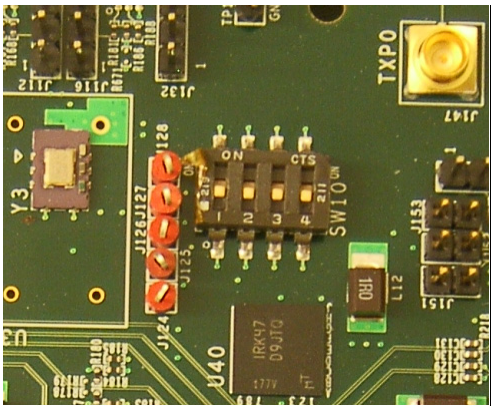

***Figure 2-12* • SF2 Dev Kit SW10**

Additionally, **SW10.1** selects whether or not MPM shuts down if a POL goes to OV2 or UV2. When OFF MPM does not shut down and when ON MPM does shut down .

- **SW10.2** selects whether or not MPM reloads the configuration in eNMV with known good values on power-up.
- **SW10.3** and **SW10.4** allow adjustment of the DEADTIME parameter for DPOLs 1 to 4 as follows:
  - If **SW10.3** is OFF, DEADTIME is not set by MPM and will either have the default values as defined in the ZL6105 documentation or f the DPOL has just powered up or the last value set by MPM has not been power cycled since the last write to DEADTIME.
  - If **SW10.3** is ON, **SW10.4** off selects a DEADTIME of 40 nS and **SW10.4** on selects a DEADTIME of 24 nS.

The eight LEDs on the DMPM board display the state of the first eight MPM outputs. With the default configuration, these LEDs represent the state of the first eight APOLs. When an LEDs is on, the corresponding APOL is in a non nominal state. When an LED is off, the corresponding channel is in the nominal state. This is useful when sequencing the APOLs as the pot can be adjusted for a channel until it enters the nominal state by watching for the corresponding LED to turn off.

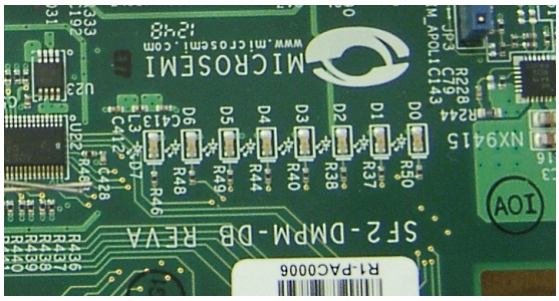

***Figure 2-13* • SF2-DMPM-DB LEDs**

When you press SW3 to initiate power-up sequencing, the MPM state changes to Starting and the various regulator enabled LEDs are shown on the SF2-DMPM-DB turning on in sequence. If the status does not change to Started and the power-on sequence restarts, adjust the voltage of the individual APOL channels through the associated potentiometers to complete the sequencing. If open-loop trimming is enabled, the open-loop trim pin voltage will only achieve nominal value if the potentiometer is suitably adjusted.

When the power sequencing is completed and all the regulators have reached nominal voltage, the status changes to **Started** and APOL closed-loop trimming is also enabled, if applicable. Closed-loop trimming keeps the APOL channel output voltage at the nominal value, as specified in the GUI, even when the potentiometer is adjusted. Remove the Trim Jumper for a given regulator to disable the trimming. The following list shows jumper configurations with APOL:

Table 2-5 • **MPM Status Description**

Jumper	APOL
JP3	1
JP26	2
JP27	3
JP28	4
JP2	5
JP29	6
JP30	7
JP31	8

Removing one of these jumpers disables closed-loop trimming for the associated APOL and the output voltage can be varied using the potentiometer for that channel. Reinstalling the jumper reactivates closed-loop trimming and resets it to nominal.

The characteristics of the channels configured in the reference design are as follows:

Note: The default demo setup for all APOLs is closed-loop trimming.

- Channel A1
  - SF2-DMPM-DB APOL1
  - Microsemi NX9415 3.3 V nominal regulator
  - POT range when switched on approximately 3100 - 3600 mV
  - Optionally open and closed-loop trimmed using CorePWM output 1 depending on the MPM GUI configuration and SF2-DMPM-DB jumper settings.
- Channel A2
  - SF2-DMPM-DB APOL2
  - Microsemi NX9415 3.3 V nominal regulator
  - POT range when switched on approximately 3100 - 3600 mV
  - Optionally open and closed-loop trimmed using CorePWM output 2 depending on the MPM GUI configuration and SF2-DMPM-DB jumper settings.
- Channel A3
  - SF2-DMPM-DB APOL3
  - Microsemi NX9415 3.3 V nominal regulator
  - POT range when switched on approximately 3100 - 3600 mV
  - Optionally open and closed-loop trimmed using CorePWM output 3 depending on the MPM GUI configuration and SF2-DMPM-DB jumper settings.
- Channel A4
  - SF2-DMPM-DB APOL4
  - Microsemi NX9415 3.3 V nominal regulator
  - POT range when switched on approximately 3100 - 3600 mV
  - Optionally open and closed-loop trimmed using CorePWM output 4 depending on the MPM GUI configuration and SF2-DMPM-DB jumper settings.
- Channel A5
  - SF2-DMPM-DB APOL5

  - Microsemi LX9610 1.5 V nominal regulator
  - POT range when switched on approximately 1340 - 1650 mV
  - Optionally open and closed-loop trimmed using CorePWM output 5 depending on the MPM GUI configuration and SF2-DMPM-DB jumper settings.
- Channel A6
  - SF2-DMPM-DB APOL6
  - Microsemi LX9610 1.5 V nominal regulator
  - POT range when switched on approximately 1340 - 1650 mV
  - Optionally open and closed-loop trimmed using CorePWM output 6 depending on the MPM GUI configuration and SF2-DMPM-DB jumper settings.
- Channel A7
  - SF2-DMPM-DB APOL7
  - Microsemi LX9610 1.5 V nominal regulator
  - POT range when switched on approximately 1340 - 1650 mV
  - Optionally open and closed-loop trimmed using CorePWM output 7 depending on the MPM GUI configuration and SF2-DMPM-DB jumper settings.
- Channel A8
  - SF2-DMPM-DB APOL8
  - Microsemi LX9610 1.5 V nominal regulator
  - POT range when switched on approximately 1340 - 1650 mV
  - Optionally open and closed-loop trimmed using CorePWM output 8 depending on the MPM GUI configuration and SF2-DMPM-DB jumper settings.
- Channel A9
  - SF2 MPM-DB DPOL1
  - Intersil ZL6105 3.3 V nominal DPOL
  - PMBus address 0x21
- Channel A10
  - SF2 MPM-DB DPOL2
  - Intersil ZL6105 3.3 V nominal DPOL
  - PMBus address 0x20
- Channel A11
  - SF2 MPM-DB DPOL3
  - Intersil ZL6105 3.3 V nominal DPOL
  - PMBus address 0x22
- Channel A12
  - SF2 MPM-DB DPOL4
  - Intersil ZL6105 3.3 V nominal DPOL
  - PMBus address 0x23
- Channel A13
  - MPM-DC DPOL5
  - Lineage PDT012A0X 1.5 V nominal DPOL
  - PMBus address 0x10
- Channel A14
  - MPM-DC DPOL6
  - Lineage PDT012A0X 1.5 V nominal DPOL
  - PMBus address 0x11
- Channel A15

  - MPM-DC DPOL7
  - Lineage PDT012A0X 1.5 V nominal DPOL
  - PMBus address 0x12
- Channel A16
  - MPM-DC DPOL8
  - Lineage PDT012A0X 1.5 V nominal DPOL
  - PMBus address 0x13

Note:

1. *The MPM DPOL channels are programmed over PMBus, switched on/off using the DPOL discrete enable digital input, and monitored using the DPOL Power Good (PG) or the equivalent digital output.*
2. *It is the MPM demo program (main.c), which implements SW3 control, LED status display, and the functionality underlying options selected with SW10.1 to SW10.4. The demo program is provided as a simple illustration of how the MPM and MPM API can be used but it is not part of the MPM reference design (for example, the core engine). Refer to the demo program code for more information on how the MPM can be used and deployed.*

Using the Mixed Signal Power Manager GUI

The MPM reference design demo can also be exercised using the MPM GUI communicating with the MPM target I^2C slave through the **Devantech/Robot Electronics USB-ISS I^2C** communications module. Ensure that the USB-ISS hardware and drivers are installed, configured, and functioning as described earlier.

The default I^2C slave address for MPM is 100 (decimal). All menu options under **Data > I^2C launch the MPM I^2C Communications** window with different default settings. For example, **Data > I^2C > Monitor Values ON/OFF** displays the window as shown in Figure 2-14.

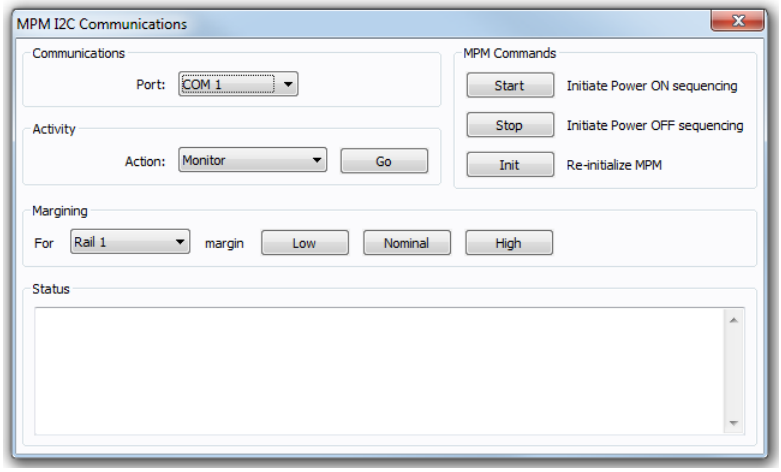

Figure 2-14 • **MPM I2C Communications Window**

Select the appropriate COM port for the USB-ISS device from the **Port** drop-down list under Communications and click **Go** under Activity to check that the MPM GUI communicates with the USB-ISS. If not, ensure that all hardware, drivers, and software configurations are set correctly.

Once the MPM GUI communications with the USB-ISS device are working, select **Find I2C Address** from **Action** drop-down list under Activity and click **Go**. For the default configuration with MPM I2C slave address 100, check the following:

```
Starting Looking for I2C Address
I2C Speed set to 1MHz
Trying I2C Address of 100
Success
```

If **Find I2C Address** is not used to dynamically scan for the MPM slave address then the GUI automatically uses the specified **I2C address** under Management Interface I2C .

Once GUI communication with the USB-ISS and the MPM target I2C slave has been established, it is possible to use the various other options in the MPM I2C Communications window to interact with the target, as shown below:

- Activity
  - **Test Dongle**: Tests communication with the USB-ISS dongle only. The USB-ISS does not need to be connected to the MPM target.
  - **Find I2C Address**: Dynamically scans to search for the MPM I2C slave address.
  - **Read**: Reads the configuration settings from the target through the I2C and populates the MPM GUI settings.
  - **Write**: Writes the configuration settings in the MPM GUI to the MPM target through the I2C. These settings will be effective when the MPM is reinitialized (in Stopped mode) and restarted.

    Note: The MPM configuration settings can be Read, reconfigured, and Write in order to change the configuration of the MPM target.
  - **Monitor**: Enters Monitoring mode and displays live updates of the target state using the Meters and Memory Map views.
  - **Go/Stop**: Click **Go** to run the chosen activity. While running, Go option changes to a Stop option. Press Stop to prematurely terminate the active activity.
- MPM Commands
  - **Start**: Initiates power-on sequencing same as using SW3 when the MPM is in Stopped mode.
  - **Stop**: Initiates power-off sequencing same as using SW3 when the MPM is in Started mode. Note that the MPM can be started using the GUI and stopped using SW3 or vice-versa.
  - **Init**: Reinitializes MPM - that is reloads the latest MPM configuration data from eNVM. It works only when the MPM is in Stopped mode.
- Margining
  - **For**: Select the channel/rail to which the following margining command will be applied.
  - **Margin**: Low, Nominal, and High cause the selected channel/rail to margin to its low, nominal, or high set voltage. There is no effect, if the MPM is not started or the relevant channel is not available.
- Status
  - Displays information about the progress and status of the most recent activity, command, or margining operation.

MPM Design Files

SoftConsole Firmware Project

A SoftConsole workspace contains the reference design firmware and the demonstration program application code is included at: C:\Microsemi\SF2_MPM_RefDesign_v6.1\design_files\SoftConsole_workspace\SF2_MPM_RefDesign.

Use the following steps to access this:

1. Make a backup or work copy of the original SoftConsole workspace.
2. Run SoftConsole v3.4.
3. Choose **File > Switch Workspace > Other….**
4. Browse to the reference design SoftConsole workspace folder containing the .metadata folder.
5. Click **OK**.

SoftConsole reopens using the reference designget reopened using the reference design firmware workspace, which contains a single *mpm_reference_design* project implementing the MPM driver and demo program.

The *main.c* file contains the implementation of the reference design demonstration program, which interacts with the MPM hardware design through the MPM driver bundled in the project's mpm folder.

The *mpm.h* and *mpm.c* files are also in the mpm folder.

Review main.c to find out how the MPM driver is used by the demonstration program.

Note: The demonstration program also includes other firmware cores used directly by the application code or the MPM driver.

The *mpm*/*mpm.h* file describes the public interface to the MPM driver.

The *mpm*/*mpm.c* and various files in *mpm* implement the actual MPM driver functionality that interfaces with the underlying MPM hardware design.

Note: The project settings include a number of manifest constant/symbol definitions under **Properties > C/C++ Build > Settings > Tool Settings > GNU C Compiler > Symbols** which configure and tune as the firmware operates.

The MPM driver public interface is described by *mpm/mpm.h*:

```
/* void mpm_task_init()
 *
 * Initializes the tasks, queues, semaphores etc for the MPM engine
 * with FreeRTOS. This call leaves the MPM tasks suspended and should
 * be called before enabling the task scheduler.
 *
 * This should be called before vTaskStartScheduler() and before mpm_init()
 * which will usually be called in the main application task.
 *
 */

void mpm_task_init(void);

/* void mpm_init()
 *
 * Initializes the MPM engine "driver" and must be called before any other
 * MPM driver methods below.
 */
void mpm_init();

/* void mpm_start()
 *
 * Starts the MPM engine.
 *
 * If MPM is not in state mpm_is_stopped then this method does nothing and
 * just returns immediately otherwise if MPM is in state mpm_is_stopped
 * then this method:
```

```
 *
 * - puts MPM into state mpm_is_starting
 * - starts channel threshold monitoring
 * - starts channel open loop trimming where applicable
 * - initiates power on sequencing
 * - waits until power on sequencing has successfully completed
 * - starts channel closed loop trimming where applicable
 * - puts MPM into state mpm_is_running
 * - channel threshold monitoring remains active
 */
void mpm_start();

/* void mpm_stop()
 *
 * Stops the MPM engine
 *
 * If MPM is not in state mpm_is_running then this method does nothing and
 * just returns immediately otherwise if MPM is in state mpm_is_running
 * then this method:
 *
 * - puts MPM into state mpm_is_stoping
 * - stops closed loop trimming where applicable
 * - initiates power down sequencing
 * - waits until power down sequencing has successfully completed
 * - stops open loop trimming where applicable
 * - stops channel threshold monitoring
 * - puts MPM into state mpm_is_stopped
 */
void mpm_stop();

/* void mpm_dev_show_state(void)
 *
 * Reflect the current state of MPM on the Dev Kit LEDs
 */
void mpm_dev_show_state(void);

void mpm_dev_debug_led_5(int state);
void mpm_dev_debug_led_6(int state);
void mpm_dev_debug_led_7(int state);
void mpm_dev_debug_led_8(int state);

/* uint32_t mpm_dev_get_inputs(void)
 *
 * return the input state of the internal mode switching and
 * debug display GPIO.
 */

uint32_t mpm_dev_get_inputs(void);

/* mpm_state_t mpm_get_state()
 *
 * Returns the state that the MPM engine is currently operating in.
 */
mpm_state_t mpm_get_state();

/* mpm_channel_state_t mpm_get_channel_state(mpm_channel_number_t)
 *
 * Returns the current threshold relative state for the channel
 * identified by the channel number passed in. If the channel number
 * passed in does not refer to a valid channel then
 * mpm_channel_is_off is returned.
 */
mpm_channel_state_t mpm_get_channel_state(mpm_channel_number_t);

/* int32_t mpm_get_channel_voltage_mv(mpm_channel_number_t)
```

```
 *
 * Returns the current channel voltage in mV for the channel identified
 * by the channel number passed in. If the channel number passed in does
 * not refer to a valid channel then 0mV is returned.
 *
 * If MPM is in state mpm_is_stopped then mpm_channel_is_off is returned
 * for all channels.
 */
int32_t mpm_get_channel_voltage_mv(mpm_channel_number_t);

/* bool mpm_is_valid_channel(mpm_channel_number_t)
 *
 * Returns true if the channel identified by the channel number passed in is
 * a valid channel recognized by the MPM engine and false otherwise.
 *
 * For a channel to be valid it must meet the following criteria:
 *
 * - Signal name in ACE configuration must be "MPM_Channel_<n>..." where
 *   <n> is a unique identification number between 1 and min(64,
 *   MPM_MAX_NUMBER_OF_CHANNELS) and "..." can be any other text.
 * - ACE configuration for this channel must include one "UNDER" threshold
 *   flag named "DOWN" and one "OVER" threshold flag named "UP" with any
 *   valid voltage level (the threshold voltage levels are dynamically
 *   adjusted at runtime by the MPM engine)
 * - None of the threshold/hysteresis/nominal voltage values configured
 *   through the MPM GUI is out of range of the underlying ACE (ABPS or
 *   direct analog input) voltage channel.
 */
bool mpm_is_valid_channel(mpm_channel_number_t);

/* uint32_t mpm_get_digital_inputs(int bank)
 * uint32_t mpm_get_digital_outputs(int bank)
 * uint32_t mpm_get_regulator_enable_outputs(int bank)
 *
 * Returns a 32 bit bitmask [31:0] representing the current state of the
 * relevant digital I/Os. In 32 channel builds bank is ignored but in 64
 * channel builds, bank == 0 is the first 32 channels and bank == 1 is the
 * second 32 channels.
 */
uint32_t mpm_get_digital_inputs(int bank);
uint32_t mpm_get_digital_outputs(int bank);
uint32_t mpm_get_regulator_enable_outputs(int bank);
```

Libero SoC Hardware Project

A Libero SoC v11.0 project implementing the MPM hardware is included at: **C:\Microsemi\SF2_MPM_RefDesign_v6.1\design_files\Libero_project\SmartFusion2_MPM_RefDesign**.

Use the following steps to access this:

1. Make a backup or work copy of the original Libero SoC project bundled with the package.
2. Run Libero SoC v11.0.
3. Browse to the Libero integrated design environment (IDE)project folder: **C:\Microsemi\SF2_MPM_RefDesign_v6.1\design_files\Libero_project\SmartFusion2_MPM_RefDesign**.
4. Select **SF2_MPM_RefDesign.prjx** and click **Open** .
5. It prompts if any IP cores are missing. download them from the repository to the vault using the Libero SoC Catalog.
6. The design comprises a top-level SmartDesign that instantiates the SmartFusion2 MSS and several fabric-based peripherals.

The SmartFusion2 MSS resources used are:

- MSS general purpose I/Os (GPIOs) used for interfacing to DPOL PG (Power Good) inputs.
- 3 x MSS eNVM data storage client placeholders for MPM firmware, MPM configuration data, and MPM logging
- 2 x MSS I2Cs for PMBus(I2C_0) and MPM I2C slave interfacing (I2C_1)
- Clock, reset, MSS/FPGA interrupt, and AMBA configuration.
- RTC for log record time stamping.

The fabric-based logic used are:

- CoreAPB3 for interfacing MSS to the fabric DirectCore peripherals.
- 2 x CoreGPIO (MPM_GPIO_Digital_IOs + MPM_GPIO_Digital_IOs_II) implementing up to 64 general digital inputs and 64 flag digital outputs. These connect to SW1, 2, 4, 5
- push-buttons, SW10, and the SF2-DMPM-DB LEDs.
- 2 x CoreGPIO (MPM_GPIO_Regulator_Enables + MPM_GPIO_Regulator_Enables_II) implementing up to 64 APOL/DPOL regulators enable digital outputs.
- CorePWM (MPM_PWM_Trimming_Outputs) implementing up to 16 APOL channel trimming PWM DAC outputs.
- CoreI2C is for MPM to DPOL PMBus connectivity - not supported in the initial release.
- CoreGPIO (MPM_GPIO_Mode) for mode selection to allow mapping of some POL enables to upper 32 regulator enables, Dev Kit LEDs and SW3.
- CoreInterrupt (MPM_CoreInterrupt) for routing I2C interrupts to fabric int.
- 8 x multiplexors to switch regulator enables between low and high 32-bit blocks.
- BIBUFs for PMBus bidirectional signal support.

Note: In the reference design, some of the output flag digital outputs are connected to LEDs. some of the MPM digital inputs are connected to switches and some of the regulator enable digital outputs are connected to regulator enables.

APOL Channels

APOL channels are assigned to ADC inputs in a sequence as it sequentially so that the order of received APOLs in the configuration matches the physical order on the SF2-DMPM-DB.

DPOL Channels

MPM DPOL channels require the following:

1. The DPOL for the channel must be connected to the MPM PMBus
2. Each DPOL on the MPM PMBus must have a unique I2C/PMBus slave address.
3. As with the APOL channels, the enable/switching input to the DPOL must be connected to the relevant MPM_GPIO_Regulator_Enable output (Channel <n> enable must be connected to MPM_GPIO_Regulator_Enable:GPIO_OUTS [n-1] where 1 <= n <= min(64, MPM_MAX_NUMBER_OF_CHANNELS). Note that in this release, DPOLs are switched only by using the discrete enable input and not using the PMBus OPERATION (0x01) command.
4. The DPOL PG output signal must be connected to a suitable MPM input and the MPM firmware modified to handle this. Refer to the reference design Libero SoC project for more information on how the SF2-DMPM-DB Channels 9-16 (DPOL1-8) are connected.

MPM GUI Overview

This section briefly describes the layout of the MPM GUI.

Property Page Tabs

Built-In Help

The main display of MPM has four property page tabs:

- Power
- Outputs [1-32]
- Outputs [33-64]
- Miscellaneous

All the configuration settings can be set in the tabs as shown in Figure 2-15 on page 30. A built-in help panel can be accessed using the **Help** option from the **Help** menu.

When a configuration parameter is selected (left panel), it will be highlighted in yellow with the relevant parameter description displayed on the right panel.

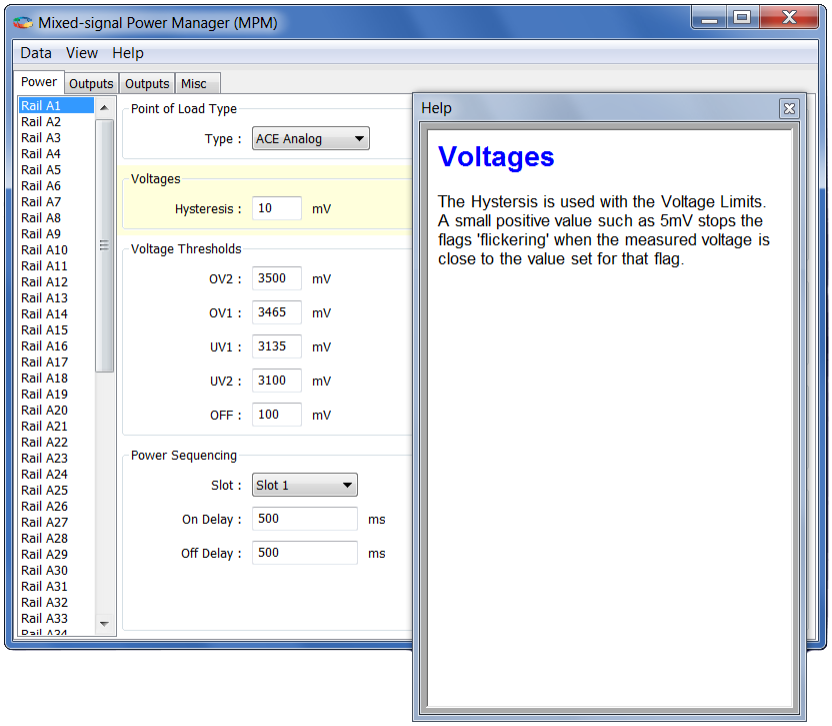

***Figure 2-15* • Built-In Help Window**

Power Property Page Tab

The Power tab is used to set the parameters for each voltage rail including:

- Channel/point of load type (ACE Analog, Generic Digital, Zilker Labs, and Lineage)
- Hysteresis, nominal, and threshold voltages
- Power sequencing slot and on/off delays
- Trimming/margining control

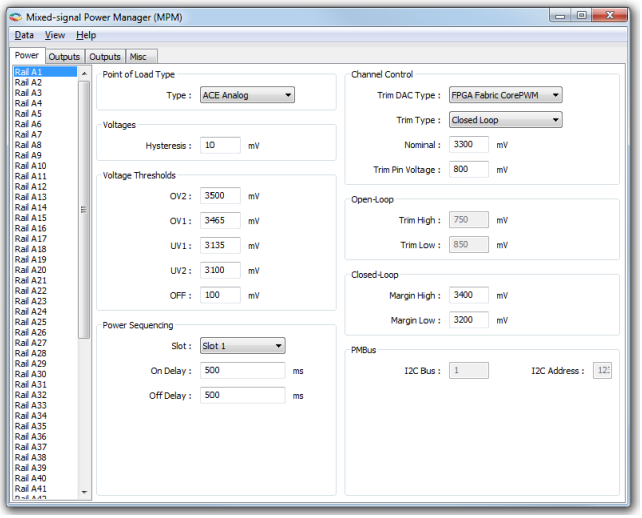

***Figure 2-16* • Power Property Page Tab Window**

Outputs Property Page Tabs

The Outputs tab is used to define the conditions and polarity of up to 64 digital outputs based on the states of various MPM channels. There are two such tabs, one for outputs 1 to 32 and the other for outputs 33 to 64. The tabs support the configuration of each digital output and how they are combined with a corresponding digital input. Also shown, is any output change and how it is logged.

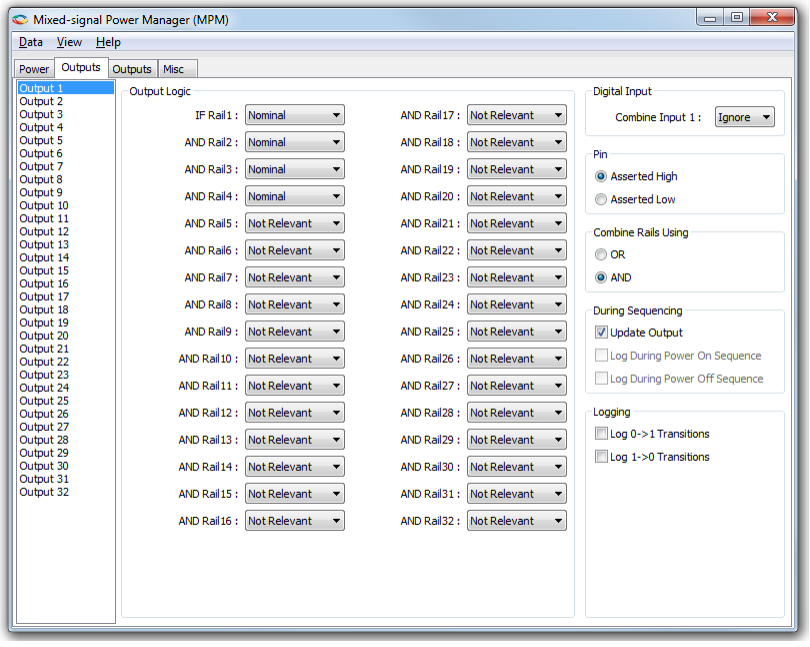

***Figure 2-17* • Outputs Property Page Tab Window**

Miscellaneous Property Page Tab

The Miscellaneous tab is used to configure various other aspects of MPM including:

- Power-on sequence slot timeout and failure action
- Power-off sequence direction
- MPM I2C slave interface address

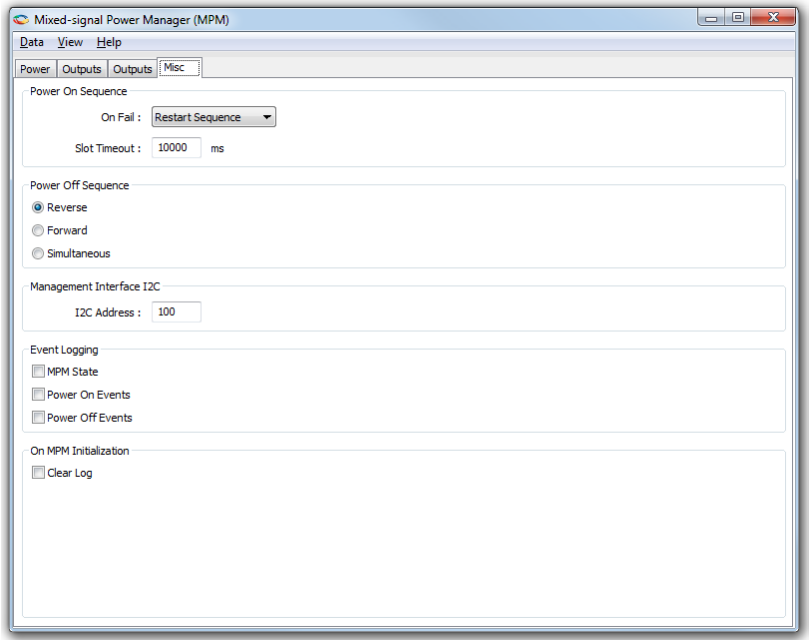

***Figure 2-18* • Miscellaneous Property Page Tab Window**

Menu Options

Figure 2-19 shows the activities that can be initiated from the MPM GUI menu.

- The Data submenu has activities relating to the transferring of data to and from the MPM and to and from the files.
- The View submenu shows or hides additional views of the configuration data.
- The Help submenu opens help on the application and gives information about the software version.

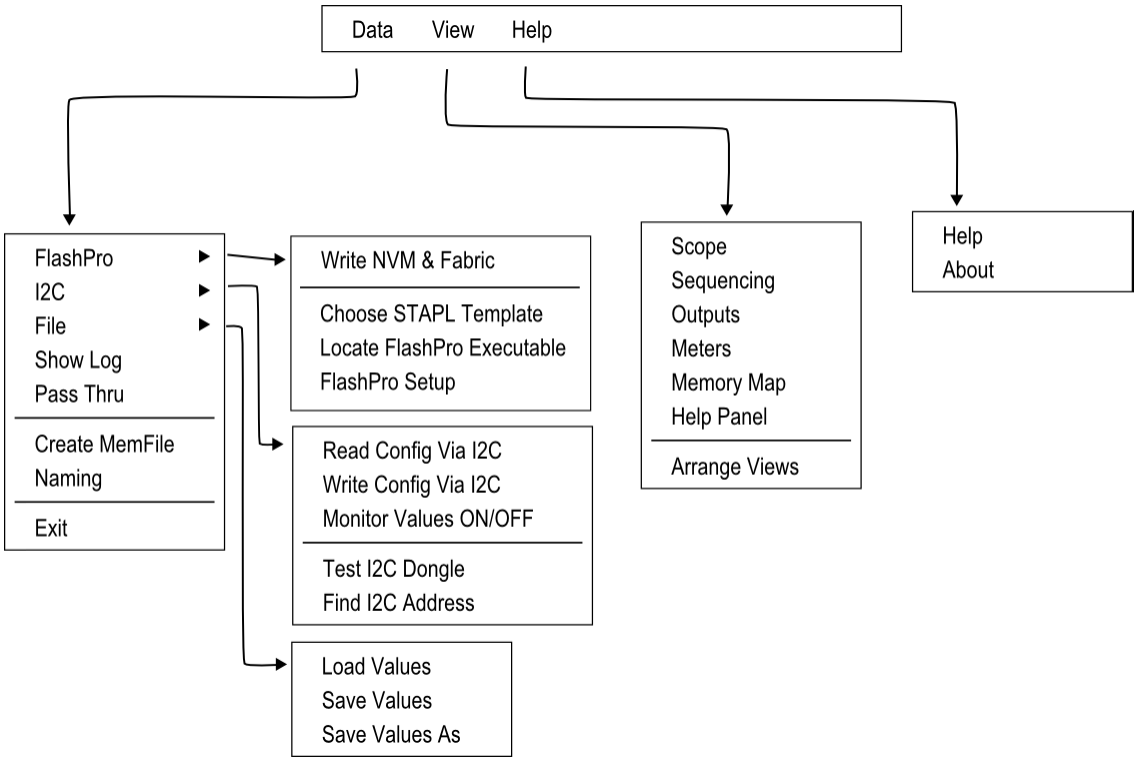

Figure 2-19 • **Menu Options**

I2C Communications

The MPM I^2C Communications dialog is for communication from the GUI to MPM using I^2C through the Devantech/Robot Electronics USB-ISS communications module/dongle. In the **Activity** section choose any activity and press **Go** to start that activity. A log will be displayed under the Status section.

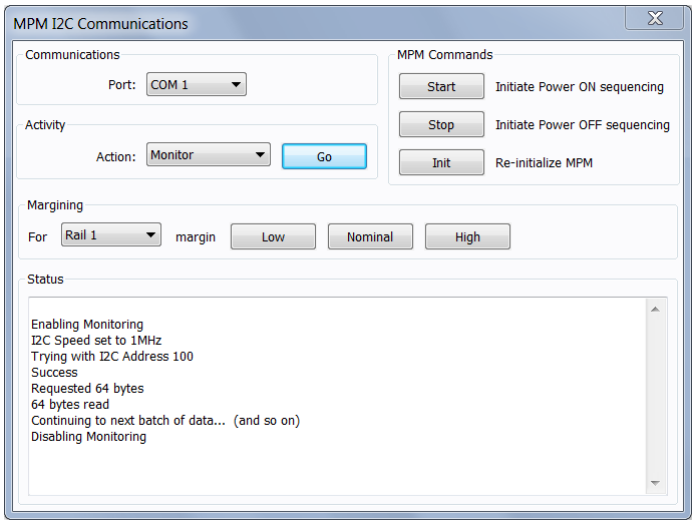

Figure 2-20 • **I^2C Communication**

Table 2-6 • **Status**

Activity	Description
Test Dongle	Tests communication between the PC and dongle only. The dongle does not need to be connected to the MPM.
Read	Reads the entire MPM configuration. The configuration values in the main tabs are updated, as are the values in the GUIs copy of the memory map.
Write	Writes the entire MPM configuration using values in the main tab.
Monitor	Repeatedly reads the live analog values at address 0x1900 to 0x197F. This activity continues until "Stop" is pressed.
Find I^2C Address	Scans the I^2C bus looking for an MPM device. The I^2C address in the Misc tab is the first address the GUI tries.

The **Go** option changes to **Stop** when there is I2C activity. The activity can be stopped by clicking Stop. The window allows for one-time actions - to power-on or power-off sequencing, to reinitialize MPM, and to perform margining on a particular rail.

Log Window

The log window displays the contents of the event log. Click **Fetch** to retrieve the most recent 128 records from the log. Each row in the log window shows the status of one log record, timestamp for it, and associated data.

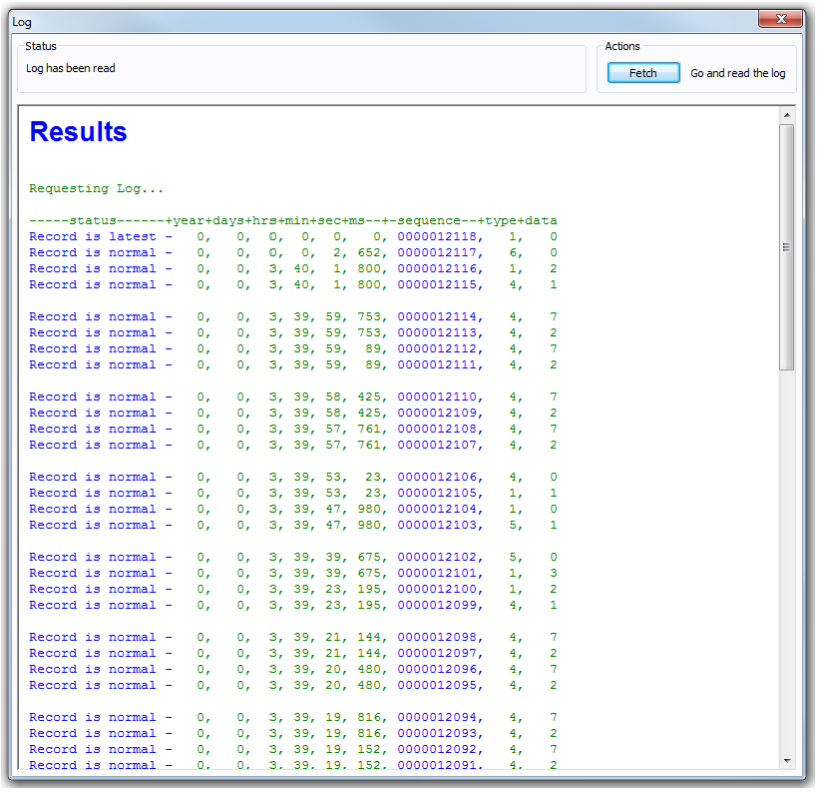

***Figure 2-21* • Log Window**

Views

Sequencing View

This gives a visual display of the configured power-on and power-off sequencing. Rails in the same slot are shown stacked above each other. The delay time for each rail within the slot is shown both numerically and as a bar below the rail's name. Clicking on a rail will open the main configuration window on the appropriate page.

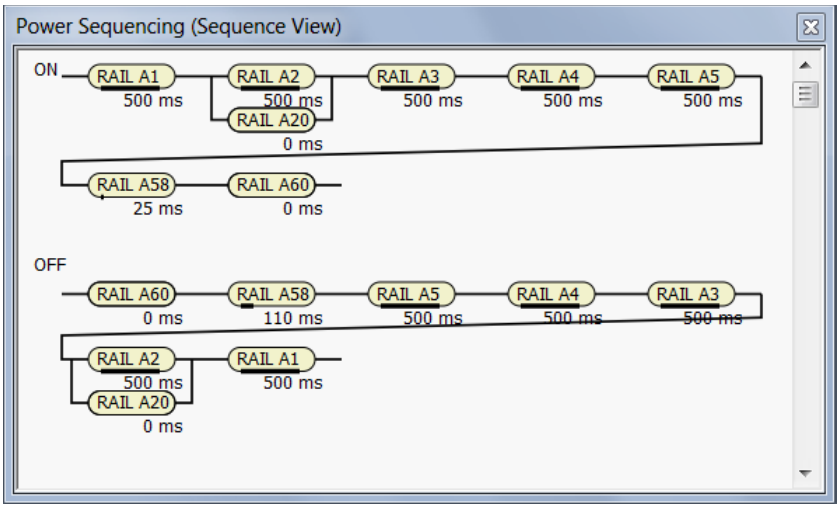

***Figure 2-22* • Sequencing View**

Scope View

This gives an alternative schematic display of the power-on and power-off sequencing as a graph. It is meant to be used as a visual guideline only, displaying only relative time frames of power-up and power-down, as a result of sequencing requirements entered in the **Power** tab field. Rise and fall times are not accurate and voltage levels are not taken into account.

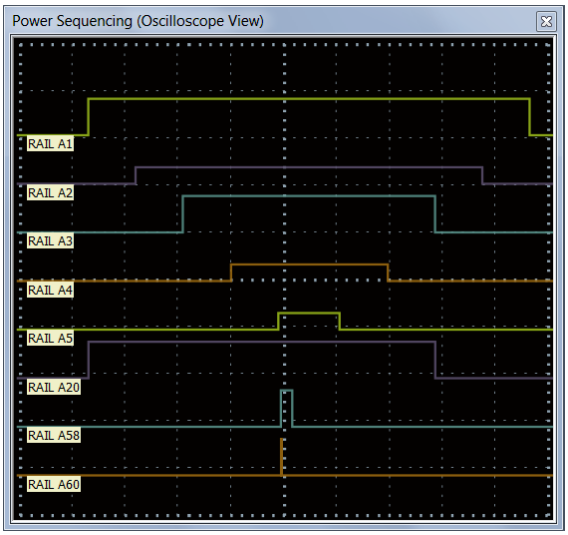

***Figure 2-23* • Scope View**

Output View

Figure 2-24 shows a schematic view of the output logic selected on the current output tab.

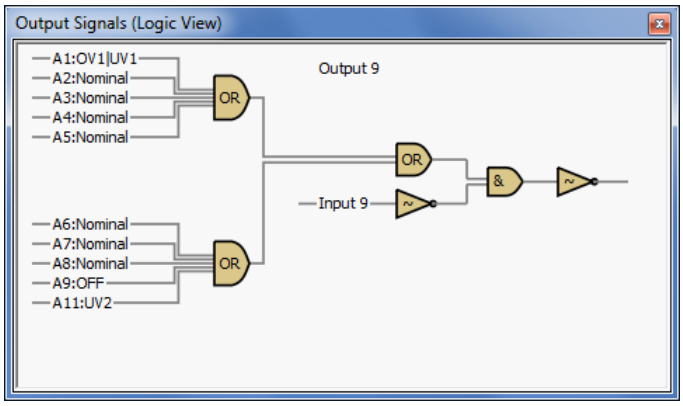

***Figure 2-24* • Output View**

Meters View

The meters view shows the values of the first twelve analog values read back by the MPM firmware at address 0x1900. For the meters to reflect live values, the USB-ISS module must be connected to the SmartFusion2 Development Kit board and monitoring must be active on the MPM I2C Communications window. The meters ranges automatically adjust to the voltage conditions. Values up to 50 V and the actual value are shown in Figure 2-25.

There are two rows of simulated LEDs above the meters view. These indicate:

- The state of each digital input used in generating the corresponding flag.
- The current state of the generated flag.

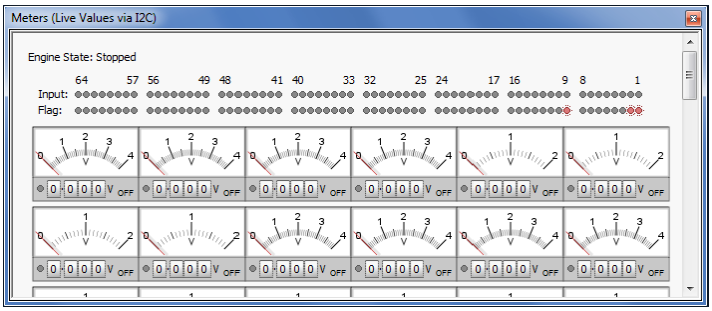

***Figure 2-25* • Meters View**

The details shown for each meter are annotated in Figure 2-26 below.

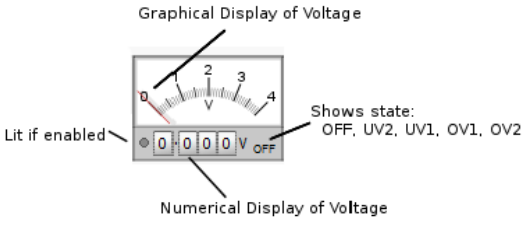

***Figure 2-26* • Details of Meter Information**

Memory Map

The Memory Map view shows a hexadecimal representation of the configuration data. Values with dark yellow backgrounds are relevant to the configuration. Values with light yellow backgrounds are not configuration values or are unused. When the help panel is open, clicking on a value gives more information about that value in the help panel.

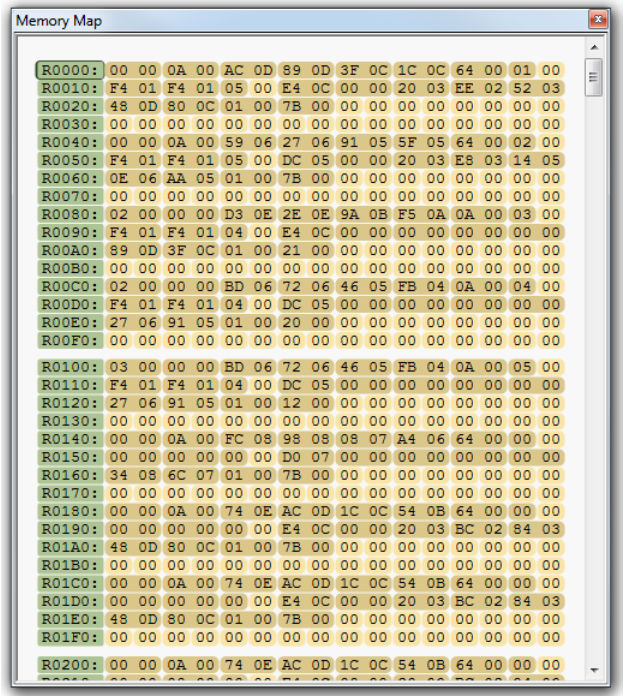

Figure 2-27 • **Memory Map View**

Pass Thru

Pass Thru allows the data to be sent from the GUI along the DMPM 1^2C through to the 1^2C for the DPOLs.

Select **Pass Thru** from the Data menu.

Click **File** and browse to select a file that can be passed to a particular DPOL.

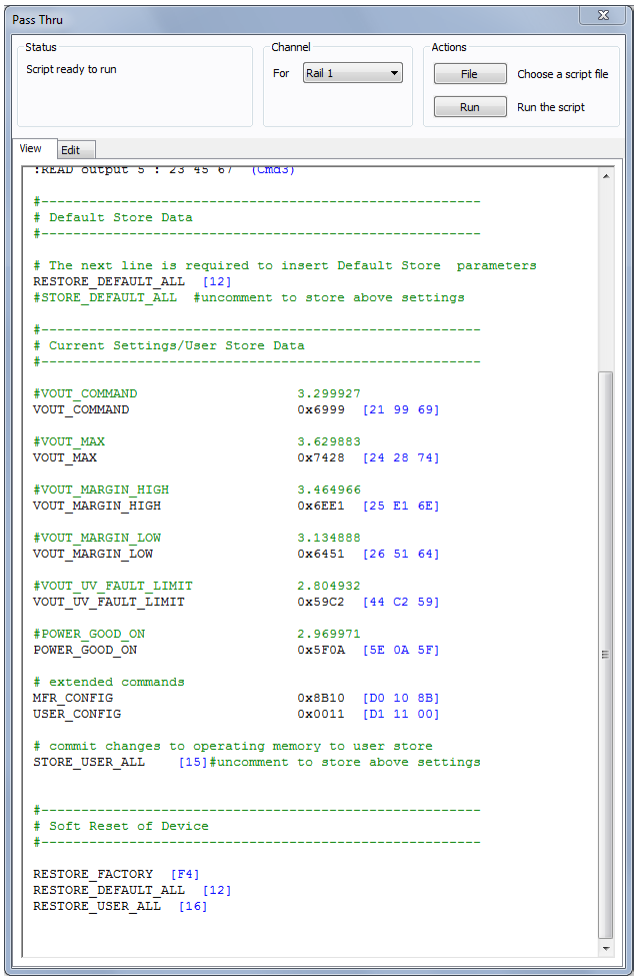

Figure 2-28 • **Pass Thru Window**

The file is parsed and the data to be sent is shown in blue. Invalid entries or entries which the software cannot parse are shown in red.

Run

Runs the configuration script by sending the values to the target DPOL. The target DPOL can be changed by selecting a different rail from the Channel drop -down menu.

Naming

The naming feature allows the default names of Rail A1, Rail A2... and Output 1, Output 2... to be changed.

Select **Naming of Rails** and **Outputs** from the Data menu.

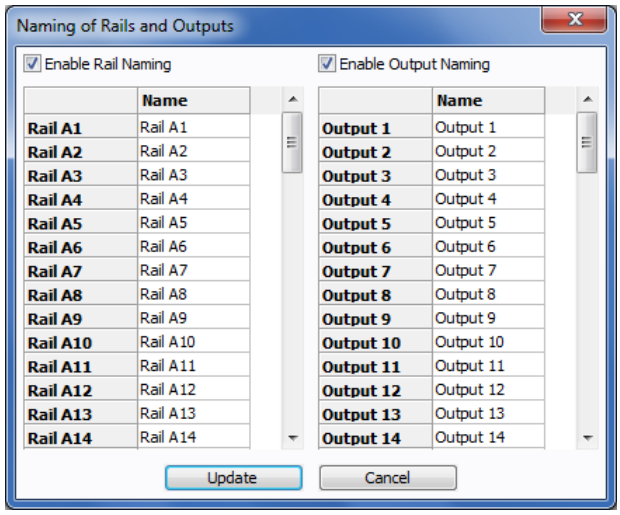

***Figure 2-29* • The Default Naming Dialog Window**

All the rail names can be edited. Modify the rail name and click **Update** to change the rail name. Ensure that **Enable Rail Naming** check-box is selected.

The names can also be saved and loaded to the configuration file and NVM along with the configuration.

The following are some restrictions on naming the rails:

- Names should be a maximum of 14 characters long. ASCII characters including 0..9 a..z and A..Z can be used. The " character will be changed to a #. If the name includes accented characters, the entire renaming resets to the default name.
- The output diagram does not update to take account of new naming. Also, the space in diagrams for the names is not increased if the name is too long.

After changing the names, the Naming of Rails and Outputs window will be as shown in Figure 2-30:

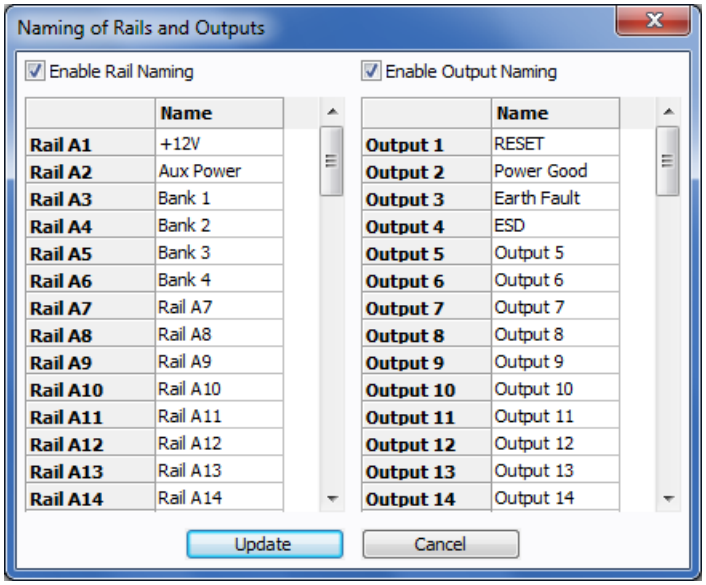

***Figure 2-30* • Naming Dialog After Renaming**

3 – Reference Design Development Flow

Purpose

The purpose of this document is to provide a detailed description of the flow of the SmartFusion2 MPM Reference Design: updating/compiling firmware, modifying the top-level FPGA design, synthesis/layout, updating NVM, generating STAPL/PDB programming file, and integrating the programming file into the GUI.

Tools and Resources Used

The following Microsemi resources were used to create the MPM Reference Design:

- Libero SoC v11(11.0.0.23)
- SoftConsole v3.4 (3.4.0.5-M20130329-1700)
- IP cores - MSS and DirectCores
  - SmartFusion2 MSS v1.0.100
  - CoreAPB3 v4.0.8
  - CoreGPIO v3.0.120
  - CoreI2C v7.0.102
  - CoreInterrupt v1.1.101
  - CorePWM v4.1.106
- Firmware drivers
  - SmartFusion2 MSS drivers.
  - SmartFusion2 CMSIS-PAL v2.1.101 - With modifications to the startup_m2sxxx.s file to support execution of NVM related code from RAM when the rest of the code is executing from NVM.
  - SmartFusion2 MSS GPIO driver v2.0.101.
  - SmartFusion2 MSS I2C driver v2.0.100:This has been modified from the stock driver to integrate with FreeRTOS when using a slave write handler.
  - SmartFusion2 MSS NVM driver v2.0.103.
  - SmartFusion2 MSS PDMA driver 2.0.102.
  - SmartFusion2 MSS RTC driver 2.0.101.
  - SmartFusion2 MSS SPI driver 2.0.103.
  - SmartFusion2 MSS Timer driver v2.0.101.
  - SmartFusion2 MSS Watchdog driver v2.0.102.
- DirectCore drivers
  - Hardware abstraction layer (HAL) v2.2.102
  - CoreGPIO driver v3.0.101
  - CoreI2C driver v2.0.103 (currently not used)
  - CorePWM driver v2.1.107
- Other drivers
  - Unified I2C driver abstraction layer (DAL) v1.0.101
  - PMBus driver v1.1.102: This has been modified to fix a bug in the linked list handling
  - SPI Flash driver: This has been modified from the version supplied with the SPI sample code to use dual channel DMA transfer.

  - SPI ADS7953 driver: New SPI based driver for Texas Instruments ADS7953 16 channel 12-bit ADC.
- Other software (Not Microsemi's)
  - FreeRTOS version 7.1.1

Building the SoftConsole Firmware Image

The SmartFusion2 MPM reference design release contains a SoftConsole v3.4 project, which can be modified and located in the SoftConsole workspace at:

C:\Microsemi\SF2_MPM_RefDesign_v6.1\design_files\SoftConsole_workspace\SF2_MPM_RefDesign

For more information on the MPM driver and how to use or modify it, refer to the "SoftConsole Firmware Project" section.

Once the project has been compiled and tested, a final version can be built for deployment in NVM by **Project > Build Configurations > Set Active > Release**. The Release target is configured to link using the mpm/mpm-production-run-from-envm.ld linker script. This linker script creates a firmware image suitable for storage in the eNVM which is mirrored to 0x00000000. Once stored in eNVM mirrored to 0x00000000, the program starts executing directly from eNVM, as shown in Figure 3-1.

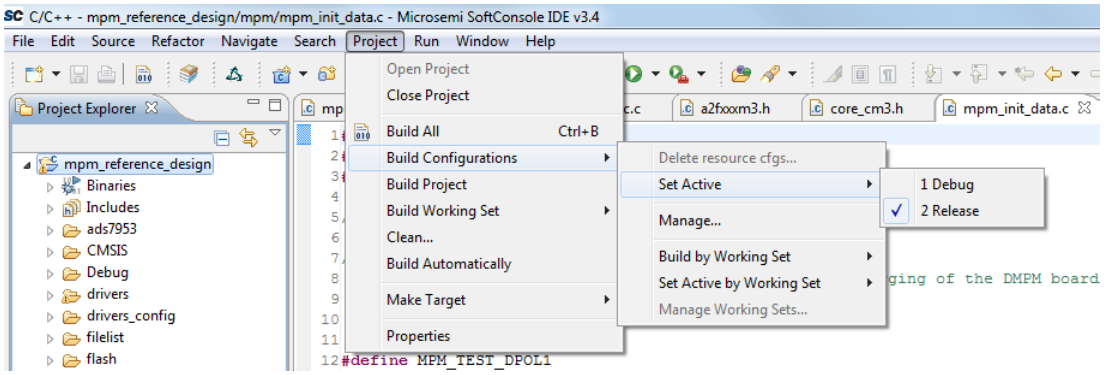

***Figure 3-1* • Setting Active Build to Release in SoftConsole**

At this point the project can be built and one of the steps is to generate a *hex file containing the firmware:

```
arm-none-eabi-objcopy -O ihex  mpm_reference_design  "mpm_reference_design.hex"
```

`mpm_reference_design.hex` will later be used to place the firmware into NVM. Refer to the "The SmartFusion2 MPM Reference design defines three MPM specific eNVM data storage clients as illustrated in Figure 3-2 on page 43.

MSS eNVM Data Storage Clients

The SmartFusion2 MPM reference design defines three MPM specific eNVM data storage, as shown in Figure 3-2.

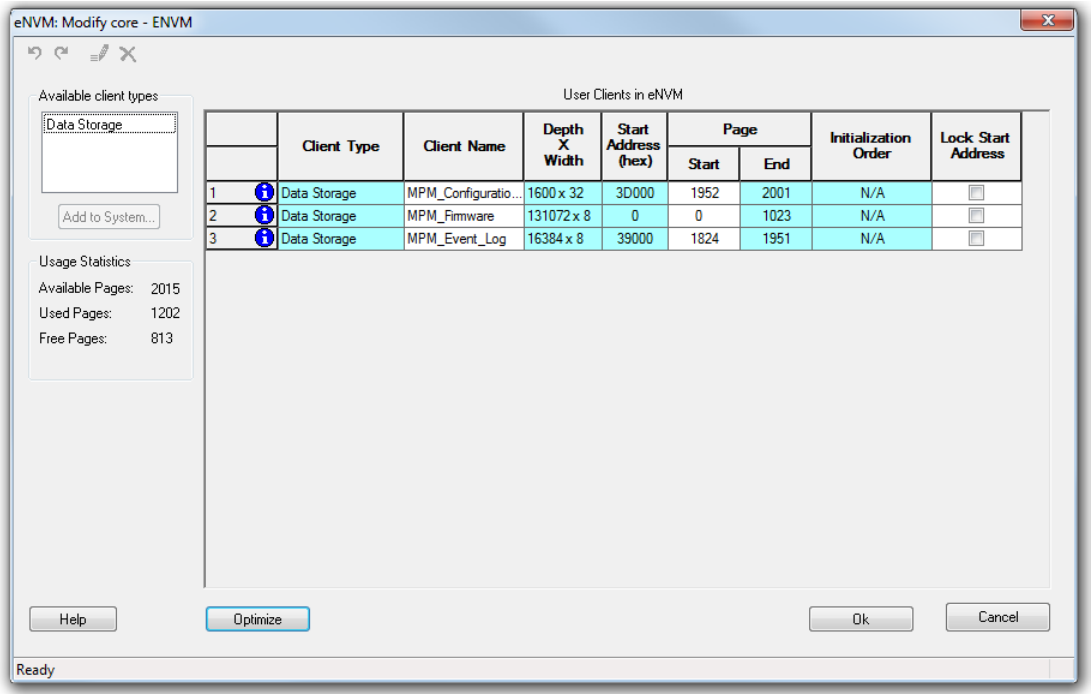

***Figure 3-2* • MSS eNVM Configuration GUI**

MPM_Configuration_Data

This is where the MPM configuration data is stored in eNVM. By default with the MPM firmware driver unchanged it has a base address of 0x3D000 from the beginning of eNVM (0x60000000) and consists of 1600 x 32-bit words or 6400 bytes.

The MPM_Configuration_Data eNVM data storage client configuration in the reference design is shown in Figure 3-3:

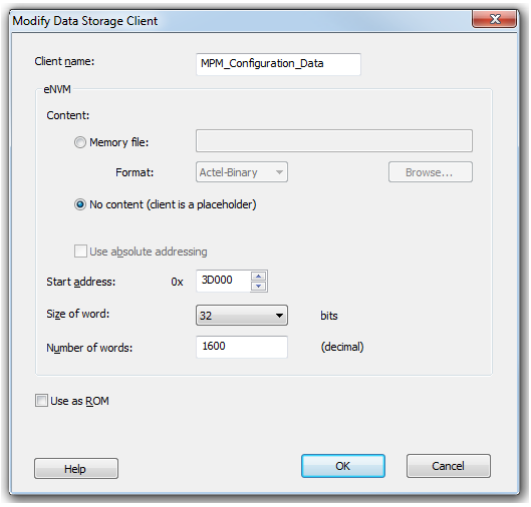

***Figure 3-3* • Data Storage Client for MPM Configuration Data**

Note: If the base address of the MPM_Configuration_Data eNVM data storage block needs to change (for example, the SmartFusion2 MSS. configuration data also stored in eNVM extends into this region) then the address must be changed consistently in both the eNVM data storage client configurator and the manifest constant in mpm.c:

```
/* Base address of MPM configuration data */
#ifndef MPM_CONFIGURATION_DATA_BASE_ADDRESS
#define MPM_CONFIGURATION_DATA_BASE_ADDRESS (0x6003D000)
#endif
```

MPM_Firmware

This is the region in NVM that stores the MPM firmware for execution out of reset. The MPM_Firmware eNVM data storage client configuration is shown in Figure 3-4.

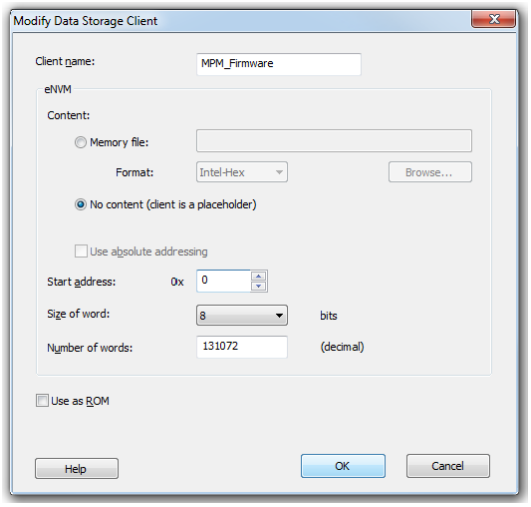

***Figure 3-4* • Data Storage Client for MPM Firmware**

To properly populate this region with the MPM firmware, select **Memory file** in the above menu and browse to the Intel-Hex file generated in the SoftConsole during the firmware build. For example,

```
C:\Microsemi\SF_MPM_RefDesign_v5.0\design_files\SoftConsole_workspace\SF_MPM_RefDesign
\mpm_reference_design\Release\mpm_reference_design.hex
```

The **Start address, Size of word, and Number of words** parameters should be detected from the .hex file and auto-populated.

MPM_Event_Logging

This region is needed only if the MPM is compiled to use a data logging event log in the internal NVM rather than the (default) external NVM.

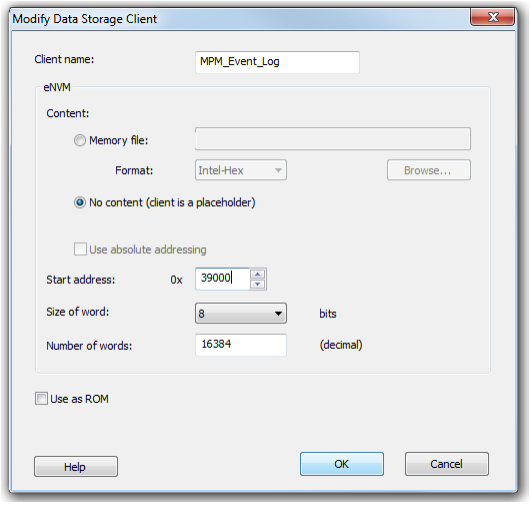

***Figure 3-5* • Data Storage Client for MPM_Event_Logging**

Note: The MPM Configuration Data, the base address, and size of this memory region must be consistent in both the eNVM data storage client configurator and the mpm.c file:

```
/* Base address and extent of MPM event logging area in eNVM.
 * These must match the configuration of the MPM_Event_Logging MSS eNVM
 * data storage client in the Libero project */
#ifndef MPM_EVENT_LOG_BASE_ADDRESS
#define MPM_EVENT_LOG_BASE_ADDRESS (0x6003A000)
#endif

#ifndef MPM_EVENT_LOG_LENGTH
#define MPM_EVENT_LOG_LENGTH (0x0004000)
#endif
```

Libero SoC Design Flow

If any changes have been made to the MPM Reference Design Libero project (for example, changing die, modification of MSS configuration settings, addition of FPGA fabric logic etc.), the project can be run through the normal Libero SoC design flow.

STAPL File Generation and GUI Integration

Once the design has been run through the design flow, it is necessary to generate the STAPL file for the design to use with the MPM GUI.

To do this, run FlashPro interactively from Libero SoC and export the STAPL file for the PDB file. Once generated, this STAPL file can be copied to the MPM GUI at:
C:\Microsemi\SF2_MPM_RefDesign_v6.1\template folder where it can be selected from the GUI using the **Data > FlashPro > Choose STAPL Template** menu.

4 – I2C Support

I2C Operation

Overview

The SmartFusion2 MPM Reference Design uses a standard 2-wire I2C slave interface, which is by default implemented using the SmartFusion2 MSS I2C_1 interface. The host can perform block reads and writes of up to 64 bytes at a time.

The SmartFusion2 MPM Reference Design I2C protocol conforms to the I2C v2.1 Specification utilizing 7-bit addressing format at effective 100 Kbps and 400 Kbps data rates2. For more information on I2C, refer to the *Philips/NXP I2C Specification Document*.

For more information on how this pertains to the MSS I2C or CoreI2C, refer to the appropriate IP/hardware and firmware driver datasheet/handbook.

Refer to section "I2C Register Map" section for specific configuration register information.

Board

I2C access from an I2C master to the MPM I2C slave is provided through the MSS I2C_1 SCL and SDA pins on the SmartFusion2 DMPM-DB as described in Table 4-1. The MPM GUI can act as an I2C master communicating with the MPM I2C slave through the Devantech/Robot Electronics USB-ISS USB to I2C dongle and the corresponding pins on this dongle are also listed.

SF2-DMPM-DB Rev B boards have a dedicated connector for the Slave I2C interface which the USB-ISS can plug directly into.

***Table 4-1* • Devantech/Robot Electronics USB-ISS Connections**

I2C Signal Board	SDL	SDA	GND
Devantech/Robot Electronics USB-ISS	SCL (I/O 3)	SDA (I/O 4)	0V/Ground
SF2-DMPM-DB Board Rev A	J22P pin 16	J22P pin 32	J3 pin 8
SF2-DMPM-DB Board Rev B	J25 pin 3	J25 pin 2	J25 pin 6
Note: Effective data rates can be less due to clock stretching. Also note that the I2C clock is configurable to other clock rates as well and is dependent on the I2C input clock(s).			

The SF2-DMPM-DB MSS I2C SCL and SDA signals have 1K Ohms pull-up resistors and have an operating voltage of 3.3 V[1] . For more detailed board specifications, refer to the SmartFusion2 Development Kit, SF2-DMPM-DB, and Devantech/Robot Electronics USB-ISS documentation.

GUI

Unlike previous releases of MPM, the GUI can only program the MPM design and original configuration from the Libero project by calling out to the FlashPro software and passing the relevant "template" STAPL file. It is not possible to include user specified MPM configuration settings in the STAPL file.

Once MPM is initially programmed to the target, the GUI can connect through the Devantech/Robot Electronics USB-ISS USB to I2C dongle to the MPM I2C slave on MSS I2C_1 in order to reprogram, control, and monitor the MPM.

1. *When using the Devantech/Robot Electronics USB-ISS, make sure to remove the Power Link jumper for 3.3 V rather than 5 V operations. Refer to the "Connections" section in: www.robot-electronics.co.uk/htm/usb_iss_tech.htm.*

I2C Protocol

This section describes the application-level protocol; that is, the meaning of specific transmitted and received bytes from/to the SmartFusion2 MPM I2C slave interface. For a description of low-level I2C protocol, refer to the I2C Specification.

Block Write

Using block writes, a host can write up to 64 bytes of data to MPM registers starting at any base address. An overview of the block write protocol is shown in Figure 4-1.

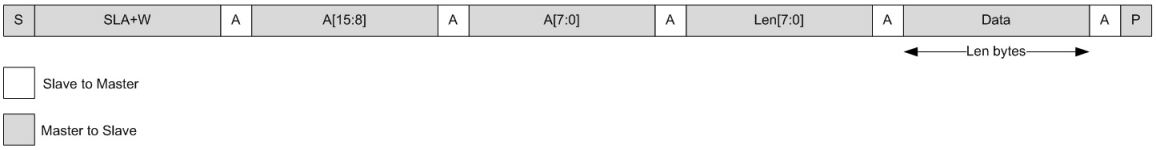

Figure 4-1 • I2C Block Write

Notes:

- *'S' denotes a start condition, 'A' denotes acknowledge (one bit), and 'P' denotes a stop condition.*
- *SLA+W refers to 8 bits: the 7-bit I2C slave address and 1-bit ('0') to indicate a write transaction.*
- *A[15:0] refers to the two-byte (16-bit) address of the MPM register being written (refer to the "I2C Register Map" on page 38 for specific addresses). The first data byte is written to the above address, and all subsequent data is written to incremental addresses.*
- *Len [7:0] is the number of data bytes that follow; the valid range is 1-64.*

Block Read

A host can read up to 64 bytes at a time from the SmartFusion2 MPM registers by issuing a block read, as shown in Figure 4-2.

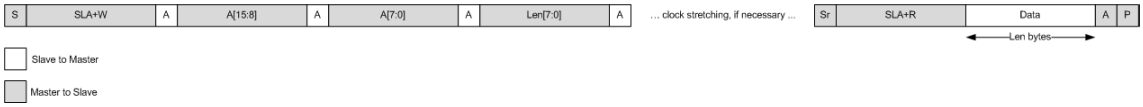

Figure 4-2 • I2C Block Read

Notes:

- *'S' denotes a start condition, 'A' denotes acknowledge (one bit), and 'P' denotes a stop condition, "Sr" denotes a repeated start condition (for change of direction).*
- *SLA+R refers to 8 bits: the 7-bit I2C slave address and 1-bit ('1') to indicate a read transaction*
- *A[15:0] refers to the two-byte (16-bit) address of the MPM register being read from (refer to the section I2C Register Map" on page 38 for specific addresses). The first data byte is read from the above address, and all subsequent data is read from incremental addresses.*
- *Len [7:0] is the number of data bytes that follow; the valid range is 1-64.*

- *Clock stretching[2] is employed by the I2C slave (MPM) for as long as is necessary to fetch the requested data.*

Commands

Command transfers, implemented as I2C writes to a fixed address (refer to the "I2C Register Map" section on page 38 for specific addresses), are used to issue real-time commands to MPM, as shown in Figure 4-3.

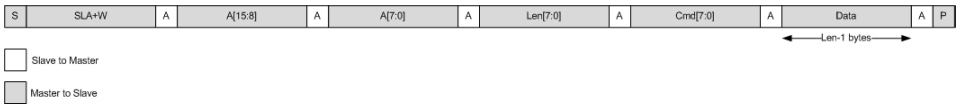

Figure 4-3 • I2C Command

Notes:

- *Cmd[7:0] is a command op-code.*
- *Len n[7:0] is the number of data bytes that follow the command op-code, the valid range is 0-64.*
- *A[15:0] in this case is fixed at I2C_CMD_0 (0x8000), but this can be expanded in firmware to support multiple command addresses.*

I2C Status

The I2C_STATUS register described in the section "I2C Register Map" reflects the result of the most recent I2C access.

Types of Access

The I2C has read and/or write access to the real or pseudo registers defined in the MPM register map. Three specific types of I2C access are supported:

- Read/write access to configuration registers - Write access to configuration registers is only allowed when the MPM is in the stopped state and each individual write operation is immediately committed to eNVM. Configuration registers changes will be applied only when the MPM is reinitialized or restarted.
- Read only access to monitoring pseudo registers - Monitoring pseudo registers can be read any time irrespective of the MPM state. Reads of rail/channel voltage pseudo registers will map to runtime calls within the MPM engine that retrieve the current rail/channel voltages.

Write access to pseudo command registers - Commands are executed by writing a command opcode and any associated data to the relevant I2C command pseudo register. Some commands may only be available in certain MPM states and ignored if the MPM is not in the appropriate state (for example, attempting to stop MPM when it has not been started). Command register writes map to runtime calls inside the MPM engine to code that implements the relevant command functionality.

Size of Transfers

Block accesses (read or write) of up to 64 bytes of data are supported. In this case, a read or write of up to 64 bytes starts at the specified MPM register map address and continues to subsequent register addresses. For example, a block read of 64 bytes from the MPM register address 0x1900 reads the 16-bit signed mV values for 64 of the MPM rails/channels and returns them in order of MPM rail/channel number (1..64)[3]. Individual 8-or 16-bit register reads or writes are also possible. Ensure that each individual configuration register write will be committed immediately to the eNVM and will be applied when the MPM is reinitialized or restarted.

2. *For more information on I2C clock stretching, refer to the 3.9 section of the I2C Rev 03 specification.*
3. *Contiguous reads may not be possible across all the register sub-blocks.*

I2C Register Map

MPM Memory Map Overview

***Table 4-2* • Memory Map Overview**

Name	Description	Start Address	End Address	Persistent Storage Space Used
Channel configuration	64 channels x 64 bytes of configuration data each	0x0000	0x0FFF	MPM_Configuration_Data eNVM data storage client. This is a 0x1900 (6400 decimal) byte sized memory block at eNVM offset 0x003D000 which is 0x603D000 in the SmartFusion2 MSS Cortex-M3 memory map.
Output flags configuration	64 flags x 32 bytes of configuration data each	0x1000	0x17FF	
Miscellaneous configuration data	256 bytes of miscellaneous configuration data - only 8 bytes actually used right now	0x1800	0x18FF	
Channel voltage monitoring registers	64 channels x 2 bytes each	0x1900	0x197F	None - these are pseudo registers that do not occupy any persistent data storage space.
I^2C status, log status, and RTC registers	7 bytes	0x1980	0x1986	
Input, Output, regulator enables, and MPM state	28 bytes	0x1990	0x19AB	
MPM channel state values	256 bytes	0x19b0	0x1AAF	
I^2C status, log status, and RTC registers	7 bytes	0x1980	0x1986	
I^2C command pseudo register	MPM I2C slave I2C command register	0x8000	0x8001	None - these are pseudo registers that do not occupy any persistent data storage space.

Base Address

The MPM register map base address is set in two places which must always be kept in sync:

1. In the SoftConsole MPM firmware project through the MPM_CONFIGURATION_DATA_BASE_ADDRESS manifest constant in mpm.c which specifies the Cortex-M3 processor eNVM base address of the MPM register map. By default this is set to 0x6003D000 which is offset 0x0003D000 from the start of eNVM at Cortex-M3 processor address 0x60000000.

   The default can be overridden by adding the following to the SoftConsole MPM firmware project properties: `-DMPM_CONFIGURATION_DATA_BASE_ADDRESS=<some-eNVM-address>`.

2. In the MPM Libero project, the MSS configuration is where the MPM Register Map eNVM data storage client's eNVM offset address is specified. By default this is set to 0x0003D000, which is implicitly with respect to the eNVM base address of 0x60000000 in the Cortex-M3 memory map and which matches the firmware project's default settings.

The MPM GUI writes MPM configuration register data to the target by reading in the template STAPL file, updating the MPM_Register_Map eNVM data storage client with the values loaded/entered through the GUI and writing a new STAPL file before invoking FlashPro to write this STAPL file to the target. Alternatively, the GUI can connect to the MPM I^2C slave in order to read/write the MPM configuration registers for the purposes if configuration, control, and monitoring.

The MPM register base address must be a 128 byte eNVM page aligned address (that is, lower 7 bits are 0). If the MPM register map base address needs to be changed then both of the above must be changed consistently. If there is a mismatch then MPM will behave unexpectedly (for example, due to reading garbage configuration data).

In general, the MPM register map must be capable of being placed anywhere in otherwise te unused eNVM address space-the amount of eNVM space available differs by die. The MPM register map must not clash with other demands on the eNVM space (for example, firmware usually at 0x60000000, MSS configuration data etc.).

Register Addresses

The MPM registers are identified using 16-bit addresses allowing for up to 64 Kbytes of register space only a portion of which is used for the MPM operation. Configuration registers are real registers insofar as they actually occupy SmartFusion2 eNVM memory in the Cortex-M3 processor address space. Other registers may be pseudo registers insofar as they have specific register addresses but do not occupy any SmartFusion2 eNVM (or other) memory in the Cortex-M3 processor address space. Real registers have their 16-bit MPM register map addresses translated internally into 32-bit Cortex-M3 processor addresses mapping to the relevant underlying eNVM addresses. This involves adding the 16-bit MPM register address to the MPM register map base address described above.

Register Map Overview

Table 4-3 summarizes the SF MPM register map. Each register is described in further detail in subsequent sections of this document.

***Table 4-3* • Register Map Overview**

Configuration Registers - Real Registers that use eNVM Storage	
Per Channel/Rail Configuration Registers - 64 (0x40) bytes for Each Channel/Rail	
Address	**Description**
0x0000	Rail A1 configuration registers
0x0040	Rail A2 configuration registers
.	
.	
0x0F80	Rail A63 configuration registers
0x0FC0	Rail A64 configuration registers
Per-Trigger/Flag Output Configuration Registers - 32 (0x20) Bytes for Each Output	
0x1000	Output 1 configuration registers
0x1020	Output 2 configuration registers
0x17C0	Output 63 configuration registers
0x17E0	Output 64 configuration registers
Miscellaneous configuration Registers	
0x1800	Miscellaneous configuration registers
I^2C Monitoring and Status Registers	
Rail voltage monitoring, RTC registers, I/O states, MPM status, and channel state registers -pseudo registers that do not occupy persistent eNVM memory storage.	
0x1900..0x197F	Rail voltage monitoring registers
0x1980..0x1985	I^2C status, log status and RTC registers
I^2C Command Registers	
0x8000	MPM I^2C command register(s) - also pseudo register(s)

Per-Rail Configuration Data

The per-channel/rail configuration data sub-block base address for channel n (1 <= n <= 64) is (0x0000 + ((n - 1) x 0x40)). Table 4-7 details the per-channel/rail registers and their offsets with respect to this per-channel/rail configuration data sub-block base address.

***Table 4-4* • Pre-Rail Configuration Data**

Name	Offset	Access	Description	
POL_TYPE	0x0000	R/W	0 = APOL 1= DPOL (generic) 2= Intersil ZL6105 3= Lineage PDT012A0X	Specifies whether the channel is an APOL or a DPOL. If it is DPOL, what type of DPOL it is.
VHL	0x0002	R/W	Threshold hysteresis VH[7:0]	Each pair of high/low 8-bit registers combines to make a 16-bit signed mV value. • For positive voltage channels 0mV < OFF < UV2 < UV1 < OV1 < OV2. • For negative channels 0mV > OFF > OV2 > OV1 > UV1 > UV2. Allowable ranges depend on the nature of the underlying ACE channel (for example, direct analog input 0 to +2560 mV or ABPS with a particular prescalar range).
VHH	0x0003	R/W	Threshold hysteresis VH[15:8]	
VOV2L	0x0004	R/W	OV2 threshold VOV2[7:0]	
VOV2H	0x0005	R/W	OV2 threshold VOV2[15:8]	
VOV1L	0x0006	R/W	OV1 threshold VOV1[7:0]	
VOV1H	0x0007	R/W	OV1 threshold VOV1[15:8]	
VUV1L	0x0008	R/W	UV1 threshold VUV1[7:0]	
VUV1H	0x0009	R/W	UV1 threshold VUV1[15:8]	
VUV2L	0x000A	R/W	UV2 threshold VUV2[7:0]	
VUV2H	0x000B	R/W	UV2 threshold VUV2[15:8]	
VOFFL	0x000C	R/W	OFF threshold OFF[7:0]	
VOFFH	0x000D	R/W	OFF threshold OFF[15:8]	
SSLO	0x000E	R/W	Power sequencing slot. Use an extra bit to accommodate 0..64, so Bit[7] reserved, Bits[6:0] power sequence slot number to use. Valid values are: 0: No slot/not sequenced 1-64: Slot in which this channel is sequenced	–
Unused	0x000F	N/A	Unused/reserved	–
SDONL	0x0010	R/W	SDON[7:0]	SDON[15:0] - Per channel power-on sequencing delay with respect to start of relevant sequencing slot. 16-bit unsigned time value in milliseconds.
SDONH	0x0011	R/W	SDON[15:8]	
SDOFFL	0x0012	R/W	SDOFF[7:0]	SDOFF[15:0] - Per channel power-off sequencing delay with respect to start of relevant sequencing slot. 16-bit unsigned time value in milliseconds.
SDOFFH	0x0013	R/W	SDOFF[15:8]	

***Table 4-4* • Pre-Rail Configuration Data (continued)**

Name	Offset	Access	Description	
TRCFG	0x0014	R/W	Trimming configuration Bits[7:3] reserved Bits[1:0] Trimming mode 0: None/disabled (default) 1: Open Loop 2: Closed Loop Bit[2] trimming control DAC type 0: SmartFusion2 MSS OBD (not supported in MPM 6) 1: CorePWM6	Only relevant to trimming enabled positive voltage channels. 0 is default for all channels other than channels 1-4 for which the default is 2. 2: Closed Loop in order to match the demo/reference design setup.
Unused	0x0015	N/A	Unused/reserved	–
VNOML	0x0016	R/W	VNOM[7:0]	16-bit signed mV value where UV1 < VNOM < OV1 (positive voltage channel) or OV1 < VNOM < UV1 (negative voltage channel) and TRILO < VNOM < TRIHI for a trimming enabled channel. Used for closed-loop trimming to requested nominal output voltage.
VNOMH	0x0017	R/W	VNOM[15:8]	
TRSDELL	0x0018	R/W	TRDEL[7:0]	Trimming (Closed-loop) startup delay with respect to the completion of power-on sequencing. Trimming starts once this delay has elapsed. 16-bit unsigned value in milliseconds. Deprecated in v3.0, not configurable through the MPM GUI.
TRSDELH	0x0019	R/W	TRDEL[15:8]	

DACOUT_NOML	0x001A	R/W	DACOUT_NOM[7:0]	For open-loop trimming DACOUT_NOM is a 16-bit value between 0 and 3300 mV specifying the voltage required as input to the regulator's trim pin circuit to achieve the required trimmed nominal output voltage on this channel.
				Internally this mV value is converted to a DAC duty cycle as follows:
				CorePWM implementation:
				`DAC_DUTY_CYCLE = ((DACOUT_NOM / 3300) * 0xFFFF)`
				`SDD Hard Macro implementation;`
				`DAC_DUTY_CYCLE = ((DACOUT_NOM / 3560) * 0xFFFF)`

DACOUT_NOMH	0x001B	R/W	DACOUT_NOM[15:8]	
DACOUT_HIL	0x001C	R/W	DACOUT_HI[7:0]	Similar to DACOUT_NOM, this is the regulator trim pin input voltage required to achieve the required trimmed "high" output voltage on this channel.
DACOUT_HIH	0x001D	R/W	DACOUT_HI[15:8]	

***Table 4-4* • Pre-Rail Configuration Data (continued)**

Name	Offset	Access	Description	
DACOUT_LOL	0x001E	R/W	DACOUT_LO[7:0]	Similar to DACOUT_NOM, this is the regulator trim pin input voltage required to achieve the required trimmed "low" output voltage on this channel.
DACOUT_LOH	0x001F	R/W	DACOUT_LO[15:8]	
TRIHIL	0x0020	R/W	TRIHI[7:0]	Closed-loop trimming high output voltage target. Normally within 10% of VNOM. 16-bit signed mV value.
TRIHIH	0x0021	R/W	TRIHI[15:8]	
TRILOL	0x0022	R/W	TRILO[7:0]	Closed-loop trimming low output voltage target. Normally within 10% of VNOM. 16-bit signed mV value.
TRILOH	0x0023	R/W	TRILO[15:8]	
DPOL_I2C_BUSL	0x0024	R/W	DPOL_I2C_BUS[7:0]	MPM I2C bus identifier - currently ignored.
DPOL_I2C_BUSH	0x0025	R/W	DPOL_I2C_BUS[15:8]	
DPOL_I2C_ADDRL	0x0026	R/W	DPOL_I2C_ADDR[7:0]	DPOL only - PMBus I2C slave address. Only bits[6:0] are significant and store the I2C address in "0x7F" mode which is shifted left one bit internally to make room for I2C R/W/GCA bit.
DPOL_I2C_ADDRH	0x0027	R/W	DPOL_I2C_ADDR[15:8]	
RAIL_NAME	0x0028 - 0x0035	R/W	Rail name as 14 byte array.	Text for MPM GUI to display - 14 bytes, nul padded if less than 14 characters are used.
Unused	0x0036 - 0x003F	N/A	Unused/reserved	

Per-Output Configuration Data, Outputs 1-32

The per-trigger/flag output configuration data sub-block base address for outputs (1 <= n <= 32) is (0x1000 + ((n-1) x 0x20)).

Table 4-8 describes the per-trigger/flag output registers and their offsets with respect to this per-trigger/flag output configuration data sub-block base address.

***Table 4-5* • Pre-Output Configuration Data**

Name	Offset	Access	Description
R1_R2_STATE	0x0000	R/W	Rail A1/A2 states used to generate this flag. Bits[7:4] Rail A2 state Bits[3:0] Rail A1 state 0: Rail does not impact flag 1: OV2 2: OV1 3: UV1 4; UV2 5: Nominal (UV1 < nominal < OV1) 6: OV1 or UV1 7: OV1 or UV2 8: OV2 or OV1 9: OV2 or UV2 10: OFF 11: Not OFF *Note: The following registers use the same encoding.*
R3_R4_STATE	0x0001	R/W	Rail A3/A4 states used to generate this flag.
R5_R6_STATE	0x0002	R/W	Rail A5/A6 states used to generate this flag.
R7_R8_STATE	0x0003	R/W	Rail A7/A8 states used to generate this flag.
R9_R10_STATE	0x0004	R/W	Rail A9/A10 states used to generate this flag.
R11_R12_STATE	0x0005	R/W	Rail A11/A12 states used to generate this flag.
R13_R14_STATE	0x0006	R/W	Rail A13/A14 states used to generate this flag.
R15_R16_STATE	0x0007	R/W	Rail A15/A16 states used to generate this flag.
R17_R18_STATE	0x0008	R/W	Rail A17/A18 states used to generate this flag.
R19_R20_STATE	0x0009	R/W	Rail A19/A20 states used to generate this flag.
R21_R22_STATE	0x000A	R/W	Rail A21/A22 states used to generate this flag.
R23_R24_STATE	0x000B	R/W	Rail A23/A24 states used to generate this flag.
R25_R26_STATE	0x000C	R/W	Rail A25/A26 states used to generate this flag.
R27_R28_STATE	0x000D	R/W	Rail A27/A28 states used to generate this flag.
R29_R30_STATE	0x000E	R/W	Rail A29/A30 states used to generate this flag.
R31_R32_STATE	0x000F	R/W	Rail A31/A32 states used to generate this flag.

***Table 4-5* • Pre-Output Configuration Data (continued)**

Name	Offset	Access	Description
FLAG_CFG	0x0010	R/W	Other flag configuration Bit[7:6] Output logging during the power sequencing 0: None 1: During the power-on sequencing 2: During the power- off sequencing 3: During the power-on and power-off sequencing Bit[5] During the power sequencing 0: Hold/do not update flag 1: Track/update flag Bit[4] combine rail states using: 0: OR 1: AND Bit[3] polarity 0: asserted high 1: asserted low Bits[2:0] combine digital input n using: 0: Ignore 1: OR 2: AND 3: XOR 4: OR (NOT input) 5: AND (NOT input)
FLAG_LOGGING	0x0011	R/W	Bits[2:0] Log 0 to 1 transitions 0: Never 1..7: Always Bits[5:3] Log 1 to 0 transitions 0: Never 1..7: Always
Output name	0x0012-0x001F	R/W	Text for MPM GUI to display - 14 bytes, nul padded if less than 14 characters are used.

Per-Output Configuration Data, Outputs 33-64

The per-trigger/flag output configuration data sub-block base address for output n (33 <= n <= 64) is (0x1400 + ((n - 1) x 0x20)). Table 4-9 details the per-trigger/flag output registers and their offsets with respect to this per-trigger/flag output configuration data sub-block base address.

***Table 4-6* • Pre-Output Configuration Data**

Name	Offset	Access	Description
R33_R34_STATE	0x0000	R/W	Rail A33/A34 states used to generate this flag. Bits[7:4] Rail A2 state Bits[3:0] Rail A1 state 0: Rail does not impact flag 1: OV2 2: OV1 3: UV1 4: UV2 5: Nominal (UV1 < nominal < OV1) 6: OV1 or UV1 7: OV1 or UV2 8: OV2 or OV1 9: OV2 or UV2 10: OFF 11: Not OFF *Note: The following registers use the same encoding.*
R35_R36_STATE	0x0001	R/W	Rail A35/A36 states used to generate this flag.
R37_R38_STATE	0x0002	R/W	Rail A37/A38 states used to generate this flag.
R39_R40_STATE	0x0003	R/W	Rail A39/A40 states used to generate this flag.
R41_R42_STATE	0x0004	R/W	Rail A41/A42 states used to generate this flag.
R43_R44_STATE	0x0005	R/W	Rail A43/A44 states used to generate this flag.
R45_R46_STATE	0x0006	R/W	Rail A45/A46 states used to generate this flag.
R47_R48_STATE	0x0007	R/W	Rail A47/A48 states used to generate this flag.
R49_R50_STATE	0x0008	R/W	Rail A49/A50 states used to generate this flag.
R51_R52_STATE	0x0009	R/W	Rail A51/A52 states used to generate this flag.
R53_R54_STATE	0x000A	R/W	Rail A53/A54 states used to generate this flag.
R55_R56_STATE	0x000B	R/W	Rail A55/A56 states used to generate this flag.
R57_R58_STATE	0x000C	R/W	Rail A57/A58 states used to generate this flag.
R59_R60_STATE	0x000D	R/W	Rail A59/A60 states used to generate this flag.
R61_R62_STATE	0x000E	R/W	Rail A61/A62 states used to generate this flag.
R63_R64_STATE	0x000F	R/W	Rail A63/A64 states used to generate this flag.

***Table 4-6* • Pre-Output Configuration Data (continued)**

Name	Offset	Access	Description
FLAG_CFG	0x0010	R/W	Other flag configuration Bit[7:6] Output logging during the power sequencing 0: None 1: During the power-on sequencing 2: During the power- off sequencing 3: During power-on and power-off sequencing Bit[5] During the power sequencing 0: Hold/do not update flag 1: Track/update flag Bit[4] combine rail states using: 0: OR 1: AND Bit[3] polarity 0: Asserted high 1: Asserted low Bits[2:0] combine digital input n using: 0: Ignore 1: OR 2: AND 3: XOR 4: OR (NOT input) 5: AND (NOT input)
FLAG_LOGGING	0x0011	R/W	Bits[2:0] Log 0 to 1 transitions 0: Never 1..7: Always Bits[5:3] Log 1 to 0 transitions 0: Never 1..7: Always
Output name	0x0012-0x001F	R/W	Text for MPM GUI to display - 14 bytes, nul padded if less than 14 characters are used.

Miscellaneous and Global Configuration Data

The miscellaneous configuration data sub-block base address is 0x1800.

Table 4-7 • **Miscellaneous and Global Configuration Data**

Name	Address	Access	Description	
PWR_ON_CTRL	0x1800	R/W	Power-on sequence control. Bits[7:3] Reserved Bits[2:0] Power-up sequence failure action 0: Hold 1: Shutdown 2: Restart slot 3: Restart sequence 4: Skip slot	–
Unused	0x1801	N/A	Unused/reserved	–
SLOT_TIMEOUT_L	0x1802	R/W	SLOT_TIMEOUT[7:0]	SLOT_TIMEOUT [15:0] - Power-on sequencing slot timeout. 16-bit unsigned time value in milliseconds.
SLOT_TIMEOUT_H	0x1803	R/W	SLOT_TIMEOUT[15:8]	
PWR_OFF_CTRL	0x1804	R/W	Power-off sequence control. Bits[7:3] Reserved Bit[2] Post power off action (currently unused by SF MPM) 0: Stay off 1: Allow restart Bits[1:0] Power off sequence 0: Reverse (slot n, n-1,...,1) 1: Forward (slot 1, 2,...,n) 2: Simultaneous (as if all channels in a single slot but off delays still accounted for)	Bit[2] is deprecated in SF MPM v3.0.
I2C_SLAVE_ADDR	0x1805	R/W	Bits[6:0] I2C slave address	MPM I^2C slave address
LOGGING	0x1806	R/W	Bit[0] MPM engine state changes 0: do not log 1: log Bit[1] Power-on sequencing events 0: do not log 1: log Bit[2] Power-off sequencing events 0: do not log 1: log	Optional logging of the MPM engine state transitions and the power sequencing events.

***Table 4-7* • Miscellaneous and Global Configuration Data (continued)**

Name	Address	Access	Description	
CLEAR_LOG	0x1807	R/W	Bit[0] 0: do not clear log 1: clear log	Clear log next time MPM reinitializes. A 0x00 end of file marker byte is written at the start of the log file so that the next time a log record is written to the log file it is written at the very beginning of the log file. *Note: Only the first byte of the log file is written (to 0x00) when clearing the log. Once the log file has been cleared MPM also clears the CLEAR_LOG register so that the log is not automatically cleared next time when the MPM initializes.*
Unused	0x1808	N/A	Unused/reserved	–

Rail Voltage Monitoring Registers

The rail monitoring pseudo register sub-block base address is 0x1900.

***Table 4-8* • Rail Voltage Monitoring Registers**

Name	Address	Access	Description	
RAIL1_VH	0x1900	R/O	RAIL1_V[15:8]	16-bit signed value representing the instantaneous output voltage for rail A1 in mV.
RAIL1_VL	0x1901	R/O	RAIL1_V[7:0]	
RAIL2_VH	0x1902	R/O	RAIL2_V[15:8]	16-bit signed value representing the instantaneous output voltage for rail A2 in mV.
RAIL2_VL	0x1903	R/O	RAIL2_V[7:0]	
RAIL63_VH	0x197C	R/O	RAIL63_V[15:8]	16-bit signed value representing the instantaneous output voltage for rail A63 in mV.
RAIL63_VL	0x197D	R/O	RAIL63_V[7:0]	
RAIL64_VH	0x197E	R/O	RAIL64_V[15:8]	16-bit signed value representing the instantaneous output voltage for rail A64 in mV.
RAIL64_VH	0x197F	R/O	Bit[0] 0: do not clear log 1: clear log	16-bit signed value representing the instantaneous output voltage for rail A64 in mV.

***Table 4-8* • Rail Voltage Monitoring Registers (continued)**

Name	Address	Access	Description	
I2C_STATUS	0x1980	R/C	Status of the most recent I2C protocol interaction: 0: OK 1: Invalid register address 2: Unrecognized command 3: Invalid data length (0 or > 64) 4: Data length mismatch (length value received did not match the amount of data that followed) 5: NVM write failure (when writing config registers) 6: Bad packet format 7: MPM could not process the command as it was not in the correct state 8 : Trim command failed. 9: Unknown PMBus slave device 10: PMBus write operation failed 11: PMBus read operation failed 12: PEC fail on PMBus data read 13: PMBus read data length too long	Mainly for debugging the I2C interactions. Indicates the status of the most recent I2C protocol interaction.
RTC4	0x1981	R/O	RTC byte 4 - bits[39:32]	40-bit RTC value from the SmartFusion2 target. Read-only access is provided so that an I2C master can read this and synchronize with its own view of real (world) time. Normally all 5 bytes would be read through the I2C in one operation since reading individual bytes separately and recombining them will most likely yield an incorrect time value.
RTC3	0x1982	R/O	RTC byte 4 - that is bits[31:24]	–
RTC2	0x1983	R/O	RTC byte 4 - that is bits[23:16]	–
RTC1	0x1984	R/O	RTC byte 4 - that is bits[15:8]	–
RTC0	0x1985	R/O	RTC byte 4 - that is bits[7:0]	–
Unused	0x1986-0x198F	N/A	Unused/reserved	–
MPM Inputs 1-32	0x1990-0x1993	R/O	Bitmapped value state of the first 32 MPM input pins.	32-bit unsigned value.
MPM Inputs 33-64	0x1994-0x1997	R/O	Bitmapped value state of the last 32 MPM input pins.	32-bit unsigned value.
MPM Outputs 1-32	0x1998-0x199B	R/O	Bitmapped value state of the first 32 MPM output pins.	32-bit unsigned value.

Table 4-8 • **Rail Voltage Monitoring Registers (continued)**

Name	Address	Access	Description	
MPM Outputs 33-64	0x199C-0x199F	R/O	Bitmapped value state of the last 32 MPM output pins.	32-bit unsigned value.
MPM Regulator Enables 1-32	0x19A0-0x19A3	R/O	Bitmapped value state of the first 32 MPM regulator enable pins.	32-bit unsigned value.
MPM Regulator Enables 33-64	0x19A4-0x19A7	R/O	Bitmapped value state of the last 32 MPM regulator enable pins.	32-bit unsigned value.
MPM State	0x19A8-0x19AB	R/O	Current state of the MPM engine: 0: Stopped 1: Starting 2: Started 3: Stopping	32-bit unsigned value.
Unused	0x19AC-0x19AF	N/A	Unused/reserved	–
MPM Channel States	0x19B0-0x1AAF	R/O	64 channel state registers with the contents indicating the state of the corresponding channel as follows: 0: Off 1: Under voltage level 2 2: Under voltage level 1 3: Nominal 4: Over voltage level 1 5: Over voltage level 2	64 x 32-bit unsigned values for channels 1 to 64 in ascending order.

I2C Command Registers

The I2C command pseudo register sub-block base address is 0x8000.

Table 4-9 • **I^{2}C Command Registers**

Name	Address	Access	Description
I2C_CMD_0	0x8000	W/O	I2C command pseudo register. Refer below for more information on how this is used.
Unused	0x8001-0xFFFF	N/A	Unused/reserved

Commands are implemented as writes to an MPM I2C command pseudo register of an 8-bit opcode followed by an optional sequence of up to 63 command data bytes. The reference design implements a single command register at MPM register map address 0x8000 and a set of command opcodes that can be written to this command register. Opcodes that are currently not defined for command register 0x8000 are reserved by Microsemi SoC Products Group for future use. Additional command registers and command opcodes can be implemented by adapting and extending the reference design firmware.

Table 4-10 describes the opcodes defined for command register 0x8000.

Table 4-10 • **Command Registers**

Code	Name	Parameters	Description/Response
0x00	Unused/reserved	N/A	N/A
0x01	START	None	Initiate power-on sequencing. Maps to the MPM driver API mpm_start(). Ignored if the MPM is not in stopped state.

***Table 4-10* • Command Registers (continued)**

Code	Name	Parameters	Description/Response
0x02	STOP	None	Initiate power-off sequencing. Maps to the MPM driver API mpm_stop(). Ignored if the MPM is not in started state.
0x03	MARGIN_LOW	One 8-bit byte containing the channel to be trimmed/margined (1..64).	Set margin level to MARGIN_LOW for the specified channel (1..64). For APOL channels: In open-loop mode, this sets the trim pin voltage to DACOUT_LO. In closed-loop mode, the trim pin voltage is continually adjusted until the output voltage reaches TRLO. For DPOL channels: Uses PMBus OPERATION (0x01) command to program the DPOL to output the margin low voltage. Ignored if the MPM is not in started state.
0x04	MARGIN_NOM	One 8-bit byte containing the channel to be trimmed/margined (1..64)	Set the margin level to NOMINAL (default) for the specified channel. For APOL channels: In open-loop mode, this sets the trim pin voltage to DACOUT_NOM. In closed-loop mode, the trim pin voltage is continually adjusted until the output voltage reaches VNOM. For DPOL channels: Uses PMBus OPERATION (0x01) command to program the DPOL to output the normal nominal voltage. Ignored if the MPM is not in started state.
0x05	MARGIN_HIGH	One 8-bit byte containing the channel to be trimmed/margined (1..64).	Set margin level to MARGIN_HIGH for the channel For APOL channels: In open-loop mode, this sets the trim pin voltage to DACOUT_HI. In closed-loop mode, the trim pin voltage is continually adjusted until the output voltage reaches TRHI. For DPOL channels: Uses PMBus OPERATION (0x01) command to program the DPOL to output the margin high voltage. Ignored if the MPM is not in started state.
0x06	INIT	None	Initiate reinitialization of the MPM. Maps to the MPM driver API mpm_init(). This causes MPM to reload the latest configuration settings. Any MPM configuration register changes made through the I^2C since the last call to mpm_init() take effect at this stage. Ignored if the MPM is not in stopped state.

Table 4-10 • **Command Registers (continued)**

Code	Name	Parameters	Description/Response
0x07	LOG_INFO	None	Returns basic status information about the log as follows: [0..3] Logging API major number. [4..7] Maximum records possible. [8..11] Maximum guaranteed records. In the event of a log rollover the number of records that will be available. [12..15] Current number of entries in the log. [16..19] Current sequence number.
0x08	LOG_RESET	None	Clears the log. All queued messages will be discarded and the log will be erased. The sequence number will be reset to 0 and a logging started entry will be placed in the log with sequence number 0.
0x09	LOG_READ_LAST	None	Reads the last 4 entries from the log (that is, the newest entries). The response read back will contain 64 bytes of data with between 1 and 4 log records returned depending on the number of records in the log (external reading should never see less than 1 record returned as a log started record is inserted into the new log before allowing to read). This command also resets the internal sequential log pointer to the end of the log.
0x0A	LOG_READ_NEXT	None	Read the next 4 entries from the log (these should be the next oldest records if no new entries have been added since the last read operation). The response read back will contain 64 bytes of data with between 0 and 4 log records returned depending on the number of records in the log and our starting position. Reading less than 4 records indicates end of the log.
0x0B	LOG_SYSTEM_INFO	None	Information on the MPM firmware in the device: [0..19] Zero terminated MPM version number string. [20..21] Version major number. [22..23] Version minor number. [24..25] Sub version number. [26..27] Device ID. This is a code to identify the hardware platform: Use 0 for MPM running on the SmartFusion Development Kit board, 1 for MPM running on the SmartFusion Evaluation Kit board, and 2 for MPM running on the SmartFusion2 Development Kit board. Customers can assign their own numbers to allow them ID different products or product revisions etc,.

***Table 4-10* • Command Registers (continued)**

Code	Name	Parameters	Description/Response
0x0C	PMBUS_RAW_WR	3 x single byte fixed parameters followed by a variable length data block. Byte 1: I2C bus number, currently only 1 is allowed here. Byte 2: 7-bit address of PMBus device in low 7 bits. Byte 3: Flag byte, b0 is PEC presence flag and rest are 0. Byte 4: First byte of data block. Overall length is the I2C command length-4.	Writes a block of data to the specified PMBus device. If a PEC byte is supplied and the device does not support it, MPM will remove the PEC byte and likewise, if the device supports PEC and PEC byte is supplied, MPM will append the correct PEC byte before transmission.
0x0D	PMBUS_READ	4 x single byte parameters: Byte 1: I2C bus number, currently only 1 is allowed here. Byte 2: 7-bit address of PMBUS device in low 7 bits. Byte 3: operation - 0 = read byte 1: read word 2-64: read block of n-1 bytes. If b7 is set then read is an extended command. Byte 4 - PMBUS command byte for read operation	This command allows data to be read from the selected PMBus device using byte, word or block reads up to 63 bytes in length. The returned response always has 1 + n bytes of data with the first byte being a status byte with 0 for success and one of the values described for I2C_STATUS on failure.
0x0E-0xFF	Unused/reserved	N/A	N/A
Note: The MPM version 6 register map and tools are not compatible with earlier versions of MPM.			

5 – Data Logging

Data Logging

Events

The following events can be logged (configurable through the MPM GUI or I2C configuration registers).

MPM Engine State Changes

SmartFusion2 MPM consists of four states-stopped, starting, started, and stopping. Transitions between states are conditional on rail statuses or can be initiated through the MPM API functions (or I2C commands). These transitions can be logged. Figure 5-1 illustrates a state diagram of SmartFusion2 MPM.

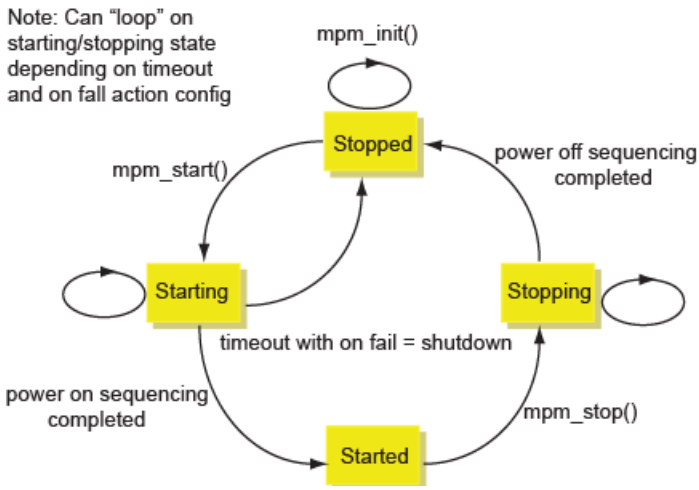

Figure 5-1 • **MPM State Transition Diagram**

Output Trigger or Flag State Change

You may log low-to-high (0 to 1) or high-to-low (1 to 0) transitions on any configured output, which can in turn be configured as a combination of rail flags and corresponding digital input.

Figure 5-2 shows the GUI configuration for output or trigger logging.

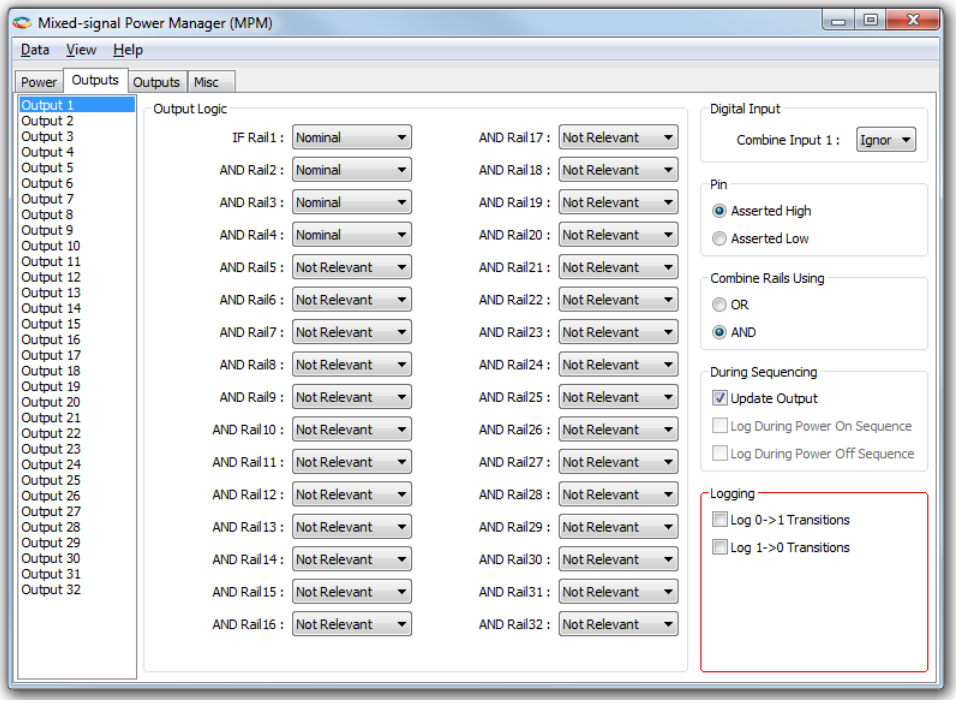

***Figure 5-2* • GUI Configuration of Output or Trigger Data Logging**

Power Sequencing Events

You may log power-on sequencing events or power-off sequencing events. Figure 5-3 shows the GUI (global) configuration for power sequencing event logging.

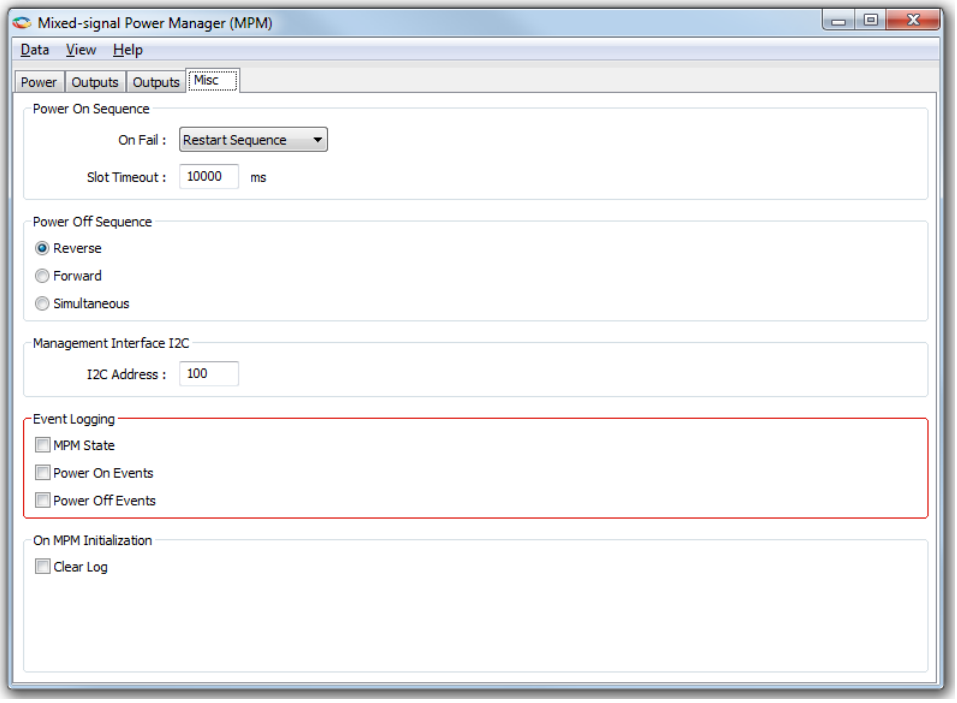

***Figure 5-3* • Power Sequencing Events Logging**

Log Record Format

Each log item consists of a record of 16 bytes as follows (LSB first):

- Byte 0: Log record type
- Byte 1: Log data Byte 1
- Byte 2: Log data Byte 2 for power-on or power-off sequencing events 0x05/0x06 only, otherwise reserved
- Byte 3-4: Reserved
- Byte 5-8: Sequence number, Little-endian order
- Byte 9-13: Log timestamp, Big-endian order
- Byte 14: Checksum of bytes 0-13 inclusive
- Byte 15: Flag Byte

More information on the above bytes is provided in the subsequent sections.

Log Record Type Byte

Table 5-1 describes the record types used for SmartFusion2 MPM data logging.

***Table 5-1* • Data Logging Record Types**

Value	Meaning
0x00	Null record type: Used to indicate the next free slot in the log (empty log item)
0x01	MPM state changed
0x02	Output flag or trigger transition occurred in the lower bank (flag outputs 1-32)
0x03	Output flag or trigger transition occurred in the upper bank (flag outputs 33-64)
0x04	Power-on sequencing event
0x05	Power-off sequencing event
0x06	Log status changed
0xFE	Extension record type
0x07 - 0xFD and 0xFF	Reserved

Log Data

Log data meaning is dependent on the corresponding log record type byte.

***Table 5-2* • Log Data**

Log Record Byte = 0x01 (MPM State Changed)	
Bit(s)	**Meaning**
1:0	0: Stopped
	1: Starting
	2: Started
	3: Stopping
7:2	Unused
Log Record Byte = 0x02 (Output Flag/Trigger Transition Occurred (Outputs 1-32))	
4:0	Output number N, minus 1 (valid values are 0-31, while output numbers range from 1-32)
5	Change to new value of output, 0 or 1
7:6	Unused
Log Record Byte = 0x03 (Output Flag/Trigger Transition Occurred (Outputs 33-64))	
4:0	Output number N, minus 33 (valid values are 0-31, while output numbers range from 33-64)
5	Change to new value of output, 0 or 1
7:6	Unused
Log Record Byte = 0x04/0x05 (Power-On or Power-Off Sequencing Event)	
Byte 1[2:0]	Event Type 0: Start 1: Finish 2: Timeout 3: Hold 4: Shutdown/Terminate; 5: Restart slot 6: Restart Sequence 7: Skip slot]

Table 5-2 • **Log Data**

Byte 1[7:3]	Unused
Byte 2[5:0]	Slot number: For "restart slot events" (configuration 5) and "skip slot events" (configuration 7). This indicates the slot number, minus 1 (valid values are 0-63, while slot numbers range from 1-64)
Byte 2[7:6]	Unused
Log Record Byte = 0x06 (Log Status Changed Event)	
2:0	0: Normal log startup 1: Abnor:mal log startup 2: Log erased
Log Record Byte = 0xFE (Extension Record)	
	Bytes 2..13 in this log record are available for user defined additional data rather than timestamp and sequence. Bytes 14..15 and byte 1 are reserved.

Log Timestamp

The SmartFusion2 MSS RTC 40-bit (5 byte) counter value is used to timestamp log records (the SmartFusion2 RTC is actually 43 bits but the top 3 bits in this application are ignored). By default the RTC is clocked by a 1 MHz clock with a prescaling divider of 3906 so that the resolution of the RTC timer is 1 second / (1MHz/3906) = 1/256th of a second = 3.90625ms and the timer can measure up to 0xFFFFFFFFFF * 0.00390625 seconds =~ 4,294,967,296 seconds =~ 136 years. The RTC timestamp is reset by the power-on reset and system reset. Thus, it cannot be used as a true RTC to allow calendar based time stamping unless an additional battery backed RTC is available in the system.

The timestamp measures the time (in 3.90625ms units) since the most recent reset of the MSS RTC.

Because of the way the MSS RTC operates, there is no guarantee that log records have monotonically incrementing timestamps. The presence of a log entry with timestamp greater than the immediately following log entry indicates logging across a SmartFusion2 device reset.

Due to the fact that the RTC range is 136 years there is no special handling of the situation in which the RTC overflows and wraps around from 0xFFFFFFFFFF to 0x0000000000.

The log file is stored in the SmartFusion2 Development Kit board 8 MB external FLASH memory accessed through the SPI.

Log records are written in blocks of four, and the log wraps around when it is full. Sequence numbers in the log record show the sequence in which the records were written. The timestamps only show times since the power-up as the clock resets to zero when the device is unpowered.

There is an API within the firmware for access to the log records, and they can be accessed using this API through the I2C.

Log Access Through the I2C

The commands supported to access the log through the I2C are described in Table 5-2 on page 70. These are the I2C commands with command codes 0x07 (LOG_INFO) through 0x0B (LOG_SYSTEM_INFO).

The difference between the maximum number of records possible in the log and the maximum number guaranteed is down to a choice in the design to use records in blocks of 4 K or 8 K. Some records in a block might not be used.

The log retrieval scheme assumes that there is a single master reading the logs at a time, as there is a single pointer in the firmware that tracks which record to hand back with each 0x0A (LOG_READ_NEXT).

eNVM Logging

Defining MPM_LOG_TO_SPI configures the MPM to use the external SPI flash and leaving it undefined configures the MPM to use the internal 16 Kb eNVM memory.

When logging to eNVM, there are a maximum of 1022 records and a maximum guaranteed 894 records.

6 – Trimming

Overview

In general, the purpose of trimming is to perform small adjustments on the output voltage of a regulator or power supply (less than 10% of the output) by driving the trim, adjust, or feedback pin of the regulator. There are two main modes for trimming, open-loop, and closed-loop, as described below.

Operation

Open-Loop Trimming

The hardware for open-loop trimming resembles the setup, as shown in Figure 6-1.

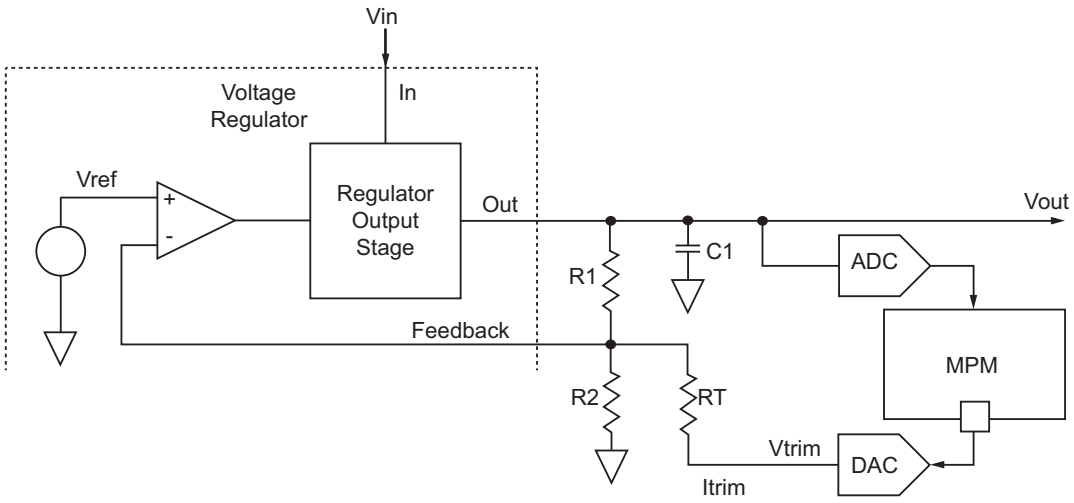

***Figure 6-1* • Open-Loop Trimming Using SmartFusion2 MPM**

The regulator feedback is controlled by SmartFusion2 MPM, which puts out a pulse-width modulated (PWM) signal that acts as a DAC when fed through a low pass filter such as an RC network. With open-loop trimming, you can adjust the regulator output voltage by driving this signal with different voltages of small variation.

Open-loop trimming is a Passive mode. It is not run continually; the feedback pin value is never adjusted. SmartFusion2 MPM sets the feedback pin value at system initialization and leaves it at that fixed value until a reset or power cycle occurs, or if a MARGIN-HIGH or a MARGIN-LOW command is issued.

Closed-Loop Trimming

The hardware for closed-loop trimming is identical to that of open-loop trimming, except that there is feedback from the regulator output to SmartFusion2 MPM, as shown in Figure 6-2.

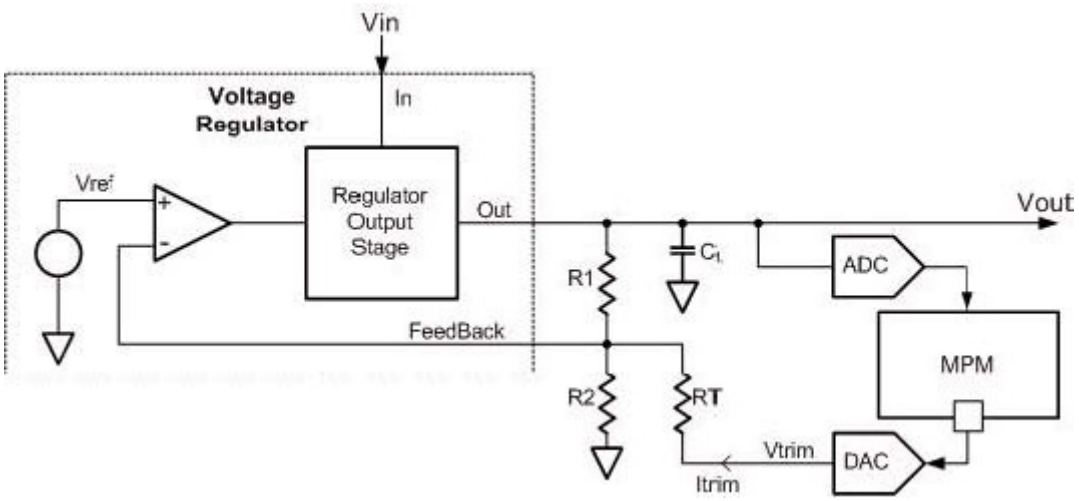

***Figure 6-2* • Closed-Loop Trimming Using SmartFusion2 MPM**

In closed-loop trimming, MPM constantly scans (once per loop) the output voltage of the regulator, and actively adjusts the regulator feedback voltage to drive the regulator to some target output voltage.

Closed-loop trimming is an Active mode; it is continually operating. The algorithm for trimming is linear, and is done as follows:

- Read V_{out} (rail value)
- Compare V_{out} to $V_{outtarget}$
- Set V_{trim} according to the following:
  - If $V_{out} > V_{outtarget}$, $V_{trim} = V_{trim} + 1$
  - If $V_{out} < V_{outtarget}$, $V_{trim} = V_{trim} - 1$

Refer to the "Trim Type" section on page 61 for information on how $V_{outtarget}$ is determined.

SmartFusion2 MPM is responsible for driving the V_{trim} pin of the above systems (both open-loop and closed-loop). However, varying the RT and R2 resistances (R1 is typically internal to regulators) changes the regulator's sensitivity to variations on the R_{trim} pin. It is your responsibility to ensure that RT is large enough so that V_{trim} variations will not put the regulator out of its operating range (typically only small variations on the feedback pin are allowed, on the order of mV's), and not so large as to make V_{trim} voltages changes negligible to the system.

Suggested resistances, as well as trim pin voltage ranges (typically around 0.8 V) can usually be found in regulator datasheets and associated application notes.

GUI Operation

The sections of the GUI highlighted in Figure 6-3 define configuration parameters used for trimming on a per-channel basis.

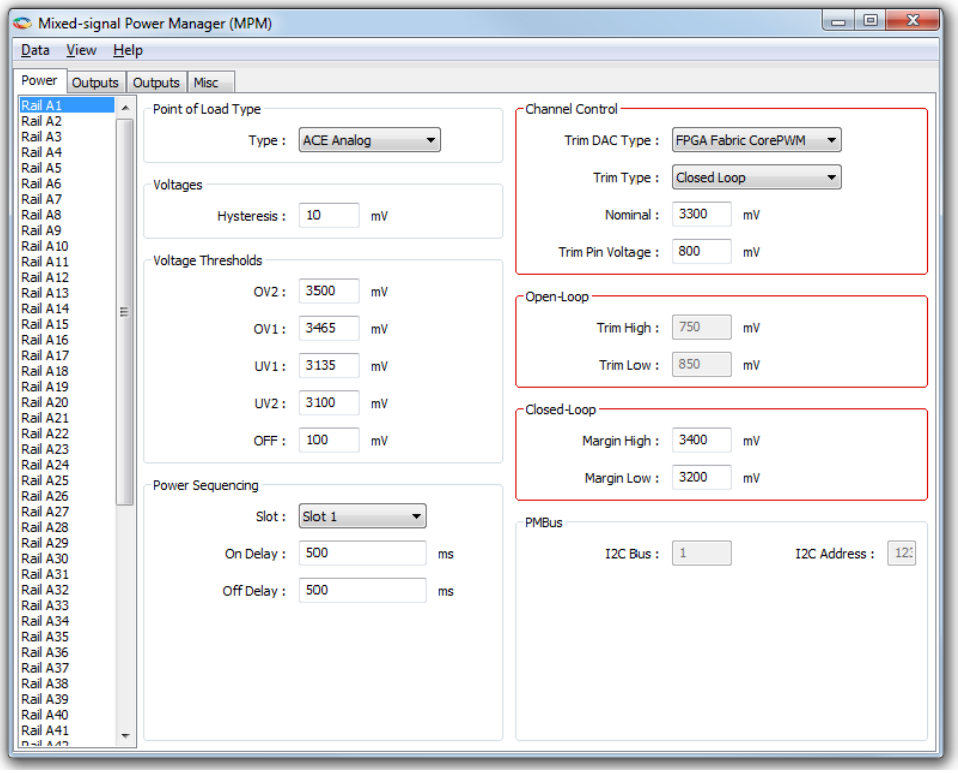

Figure 6-3 • **Trimming GUI Section**

Channel Control

The following section describes the parameters in the Channel Control pane of the Power tab of the GUI. Clicking on the pane displays the online help on the right-side of the GUI.

For DPOL channels only the **Channel Control > Nominal and Closed Loop > Margin High and Closed Loop > Margin Low** settings are relevant. DPOL trimming is done using the PMBus OPERATION (command code 0x01) command. DPOLs do not support dynamic closed loop trimming by way of a trim voltage input signal.

Trim Type

This is the trimming type as described in the "Operation" section on page 59.

Disabled

If this option is selected, trimming is not performed for this rail. The portion of the MPM loop that would normally be executed for trimming is skipped for this particular rail.

Open-Loop

If this option is selected, it resets the trim pin driven with **Trim Pin Voltage**.

After startup, the trim pin voltage is not adjusted, unless a MARGIN_LOW or MARGIN_HIGH command is issued, in which case the trim pin voltage jumps to TRIM_LOW or TRIM_HIGH, respectively.

Closed-Loop

As with open-loop, on reset the trim pin is driven with one of the three values:

However, after all rails have been powered up, the trim pin voltage is adjusted according to the scheme described in the "Closed-Loop Trimming" section on page 60. The target voltage (Vouttarget) is one of the three values:

1. Nominal: By default, or if the MARGIN_NOM command has been received by the MPM.
2. Margin High: If the MARGIN_HIGH command has been received by the MPM.
3. Margin Low: If the MARGIN_LOW command has been received by the MPM.

Note: This is a target voltage, and you must ensure that the voltage is within the operating range of the system. That is, you must ensure that the voltage range on the DAC output, which ranges from 0 V to 3.3 V for soft CorePWM implementation is sufficient for the system (feedback resistor, voltage divider, trim pin operating range, and so on) and for the amount of trimming.

Trim DAC Type

For trimming, this determines the type of DAC used for this rail. SmartFusion2 MPM only supports trimming through FPGA fabric CorePWM.

FPGA Fabric CorePWM

This represents the CorePWM in low-ripple DAC mode. It requires a low-pass filter at the output of the digital I/O.

When using the DMPM-DC the MPM GUI configuration settings for trimming of Channel A1 to Channel A8 must be matched by the appropriate configuration of DMPM-DB trimming jumpers (Jumper fitted to enable trimming, removed to disable trimming):

- Channel A1 - JP3
- Channel A2 - JP26
- Channel A3 - JP27
- Channel A4 - JP28
- Channel A5 - JP2
- Channel A6 - JP29
- Channel A7 - JP30
- Channel A8 - JP31

7 – MPM Firmware Notes

Introduction

This section provides the additional information about the MPM firmware to assist implementers in understanding the code and modifying it for their designs. It provides information on how the MPM is implemented using FreeRTOs and on the compile time defines which can be used to alter facets of MPMs operation.

MPM and FreeRTOS

The MPM 6.1 uses a Real Time Operating System (RTOS) to coordinate the various subsystems that implement its functionality. The MPM 6.1.100 uses FreeRTOS 7.1.1 (www.freertos.org) as its RTOS. The following sections describe the tasks that are used in the MPM 6.1 to implement its functionality.

The MPM reference application consists of a main application task which is responsible for overall system operation and a number of tasks which implement the MPM functionality. The following are the tasks and their relative priorities ranked from lowest to highest:

mpm_logging_task()

mpm_logging_task: This task is currently configured with a priority just above the FreeRTOS idle task and is responsible for initializing the logging system and writing log records into the event log. It has a low priority on the basis that writing to the log is a non-critical operation which should impact the other functionality as little as possible.

It reads the records to log from a queue and writes them to either the internal eNVM or an external SPI connected FLASH memory device. The length of the queue is determined by the MPM_LOG_QUEUE_LEN defined in mpm_logging.h which is currently set to 10.

This task uses two mutual exclusion semaphores to control the queue insertion (guaranteeing insertion and time stamping in the order of arrival) and exclusive access to the SPI interface.

mpm_main_task()

This is an example main application task which is responsible for the user interface through SW1 - SW2, SW10, and the eight LEDs on the SmartFusion2 Development Kit board. This task uses the MPM API to start and stop sequencing and to demonstrate how to monitor and control the MPM engine from an application.

This has a priority which is less than the remaining MPM tasks to ensure that it does not interfere with the operation of MPM. This task sleeps for 20 mS after each pass through the main loop which allows the logging task get an opportunity to run.

mpm_ i2c_slave_task()

This task is responsible for executing commands received over the slave I2C interface. It is linked to the I2C slave write handler ISR call back through a single entry queue which decouples the I2C command processing from the I2C ISR. This task relies on a mutual exclusion semaphore to arbitrate access to the PMBUS interface for some operations.

This task sleeps until an I2C request arrives, processes, and responds to the request and goes back to sleep.

mpm_ threshold_task()

This task is responsible for checking the state of the analog channels based on the voltages read through the ADS7953 ADC. It is also responsible for updating the MPM outputs based on any changes to the channel states.

This task sleeps for 2 ms after each iteration through its checking.

mpm_ timer2_task()

This task is responsible for handling all the sequencing and trimming for MPM. As such, it is the central task for managing the internal operation of MPM. It is also responsible for servicing the MSS_I2C_system_tick() functions for the 2 MSS I2C channels used by MPM to ensure timeout operations work correctly.

This task replaces a periodic interrupt in previous MPM implementations and the timing in the underlying sequencing code has been modified to work correctly even if this task is not called with complete regularity.

This task sleeps for 2 mS after each pass through.

Compile Time Options in the MPM Code

There are a number of build options in the MPM code which are controlled by the macros. The macro definitions can be used to control implementation of optional functionality and/or amounts of resources used by the MPM. Many of these macros are defined in mpm.h.

This section describes the main options and how to configure them.

32 Vs 64 Channel Operation

The MPM 6.1 can support sequencing of a maximum of either 32 or 64 power channels. To ensure that the support of 64 channels does not impact the code and data usage of 32 channel builds, the **MPM_64_CHANNEL_SUPPORT** macro is used to exclude 64 channel specific code and data when it is not required.

MPM is only configured for 64 channel operation, if **MPM_64_CHANNEL_SUPPORT** is defined in the project.

The **MPM_64_CHANNEL_REMAP** macro can be defined to make MPM reconfigure the I/O to allow APOLs 5 to 8 and DPOLs 5 to 8 appear as channels 33 to 40, respectively. This involves remapping the POL enable signals in hardware and remapping the DPOL PG to channel correspondence in the firmware. This allows MPM demonstrate proper 64 channel operation.

Note: The hardware configuration shipped with the MPM 6.1 reference design supports 64 channel operation built-in and the initialization routines can control mapping of the POLs between the lower 32 channels and the upper 32 channels. The additional hardware resources for 64 channel mode can be removed if they are not required and the fabric is reconfigured.

Analog Monitoring of DPOL Voltages

The ADS7953 ADC used on the SF2-DMPM-DB has 16 analog channels. The first eight of these channels are used to monitor the outputs of the APOLs and the remaining eight are connected to the outputs of the DPOLS. Under normal circumstances these eight additional channels are not used but it is possible to set up configurations in MPM which allow them be used to view the output voltage from the DPOLs through the ADC.

Defining the **MPM_SHADOW_APOLS** macro and the **MPM_RELOAD_ENVM** macro creates eight dummy APOL channels numbered 17 to 23 which have identical nominal voltage and thresholds to the corresponding eight DPOLS. For example, DPOL1 and APOL nine (channels 9 and 17) are paired and when channel 17 is read through the MPM GUI it should show the same voltages as channel 9. If **MPM_SHADOW_APOLS** is not defined, the dummy DPOLs can be configured through the MPM GUI to achieve the same result.

Known Good Configuration Setup

When modifying the hardware configuration of the MPM design, switching between 32 and 64 channel build modes or reprogramming the eNVM directly from Flashpro, the configuration in eNVM can become invalid and will usually cause sequencing to fail.

There is a convenient method to ensure that a known good configuration is written to the eNVM on power-up which is controlled through the **MPM_RELOAD_ENVM** macro. Defining this macro in the project causes the MPM to reload the eNVM configuration from a set of values defined in the mpm_init_data.c file if the current contents do not match the values stored there.

This reloading feature is further controlled by SW10.2, as described earlier.

It is important to undefine this macro or set SW10.2 to off when you are finished reloading the eNVM otherwise any configuration changes made through the GUI will be undone at the next reset or power-up.

There are a number of other macros associated with this configuration operation which are documented in the comments in mpm_init_data.h.

Logging Support

The logging of events to nonvolatile memory can be enabled by defining the MPM_ENABLE_LOGGING macro. Defining this macro causes the mpm_logging_task() to be created and scheduled and adds the logging access functionality to the I2C slave interface. This macro is usually defined in the project options and logging is enabled in the default build shipped with the MPM 6.1.

Internal Vs External Logging

When logging is enabled in the MPM (through the **MPM_ENABLE_LOGGING** macro), it is possible to select logging to the internal eNVM memory or to an external SPI connected flash memory device.

This is controlled through the **MPM_LOG_TO_SPI** macro. Defining this macro in mpm_logging.h or in the project configuration, selects logging to an external flash device which is accessed through an SPI interface. On the SmartFusion2 Development Kit board, this flash device is an Atmel AT25DF641, which has up to eight MB of storage available. By default the first 32 KB of this is used for logging.

If this macro is undefined, the 16 KB of eNVM reserved for logging is used by default.

The amount of memory used for logging and its location within the memory device are defined in a series of macros in the mpm_logging.h file. Refer to the associated comments for more information on this. If you increase the size of the eNVM logging memory allocation, you will need to ensure that the details for the associated storage client are also updated.

I2C Slave Interface Support

The I2C slave interface in the MPM 6.1 can be enabled and disabled through the use of the **MPM_ENABLE_I2C** macro. Defining this macro causes the mpm_i2c_slave_task() to be created and scheduled and configures the MSS I2C 0 interface appropriately. This macro is usually defined in the project options and the slave I2C interface is enabled in the default build shipped with the MPM 6.1.

PMBus I^2C Device Selection

The fabric design for the MPM 6.1 has a Core I2C implemented in it which is currently unused. The **USE_MSS_PMBUS** macro must be defined to ensure that the MPM uses MSS I2C 0 for the PMBus interface.

8 – MPM Daughter Board Hardware Guide

Introduction

The SmartFusion2 Digital Mixed Signal Power Manager Daughter Board (SF2-DMPM-DB) enables system designers to evaluate and demonstrate the functionality of Microsemi's power management solutions in hardware, using a 16 regulator benchtop power management development system. The daughter board includes eight analog regulators with adjustable bias voltage POTs, eight digital regulators residing on a connectable PMBus (I^2C) bus and a series of LED's that may be used to demonstrate and explore MPM digital output functionality. Figure 8-1 shows the SF2-DMPM-DB with its major components.

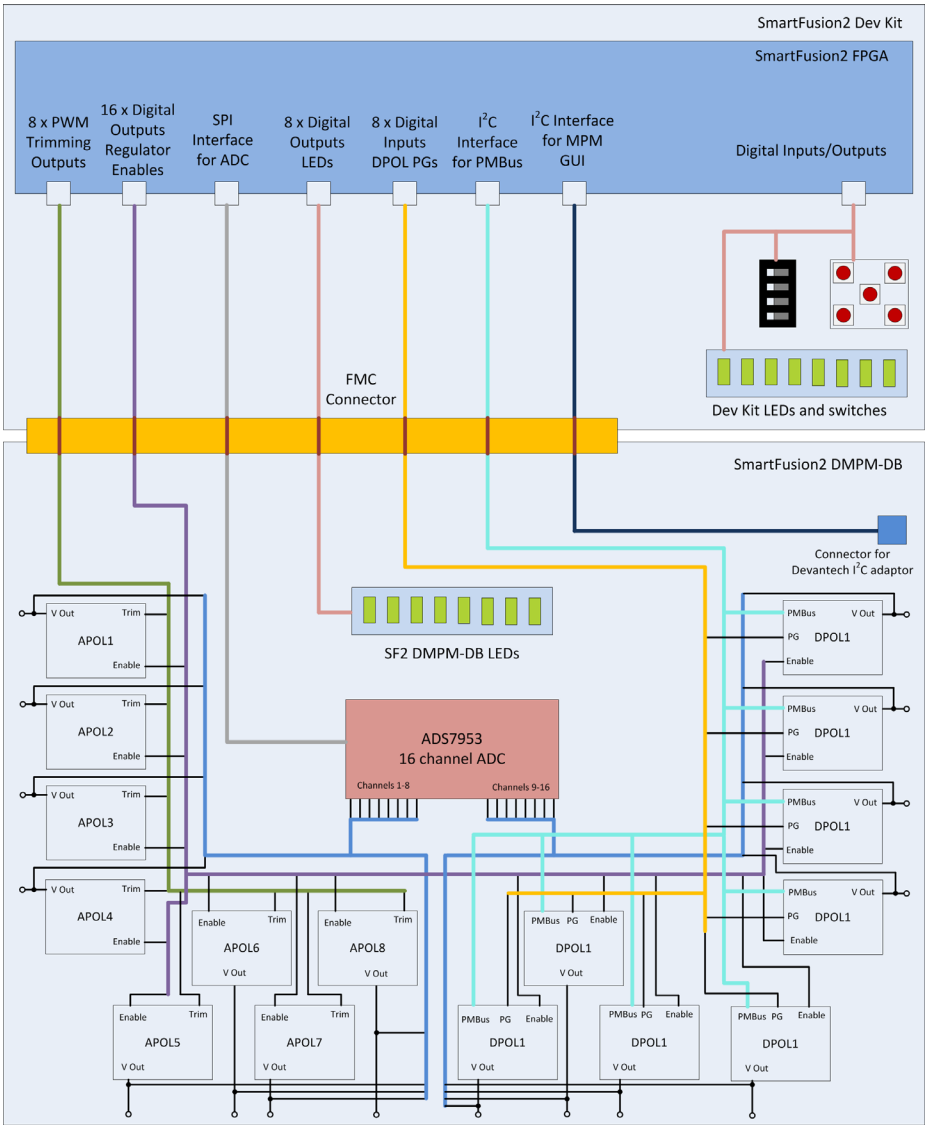

***Figure 8-1* • SF2-DMPM Daughter Board Block Diagram**

Kit Contents

***Table 8-1* • SF2-DMPM Daughter Board Kit Contents**

Quantity	Description
1	SF2-DMPM Daughter Board
1	12V, 5A AC Power Adapter
1	Any plug to Grounded 3 pin UK/Hong Kong Plug Adapter
1	Enhanced USB I2C Module
1	USB Cable A-B 3'
50	Mini Jumpers
1	SF2-DMPM-Kit Quickstart Card

Related Information

- SmartFusion2 Development Kit board
- SmartFusion2 Reference Documents
- Libero IDE Design Software

Board Description

The SF2 DMPM Daughter Board provides a benchtop demonstration and development platform for Microsemi's SF2 MPM reference design, specifically for version 6.1 of the design - which incorporates digital regulators (controlled and monitored via PMBus) in addition to the existing analog regulators from version 3.0 of the reference design, which are monitored through an external ADC.

Figure 8-2 shows the SF2 DMPM Daughter Board connected to the SmartFusion2 Development Kit board through their respective FMC connectors.

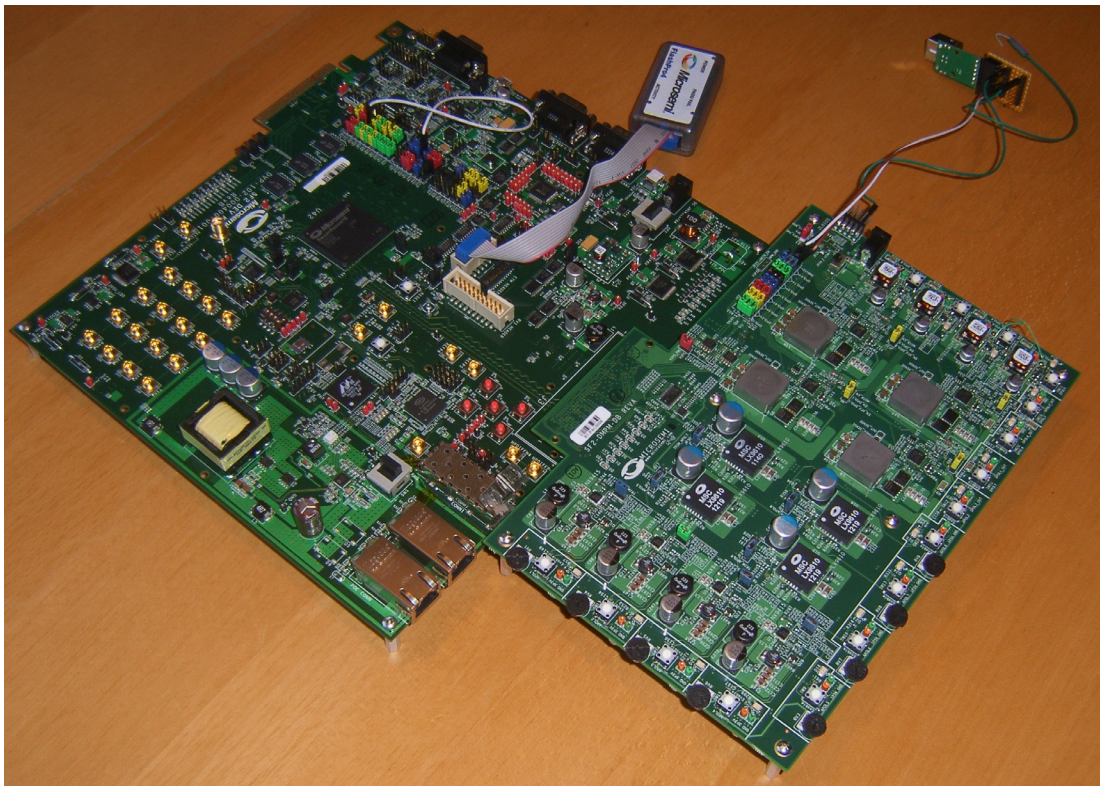

***Figure 8-2* • SF2-DMPM Daughter Board with SmartFusion2 Development Kit Board**

Digital Mixed Signal Power Manager Board Components

***Table 8-2* • DMPM-DB Components**

Serial Number	Name	Description
1	APOL1, 2, 3, 4	3.3 V regulated supply, Microsemi Analog Mixed Signal Group NX9415
2	APOL5, 6, 7, 8	1.5 V regulated supply, Microsemi Analog Mixed Signal Group LX9610
3	DPOL1, 2, 3, 4	3.3 V digital regulated supply, Intersil ZL6105
4	DPOL5, 6, 7, 8	1.5 V digital regulated supply, Lineage Power PDT012A0X
5	SW8, SW13, SW14, SW15, SW9, SW16, SW17, SW18, SW10, SW12, SW19, SW20, SW11, SW21, SW22, SW23	Switch to enable/disable voltage regulators (1 to 16) to demonstrate regulator failure.

Note: DPOL1 to DPOL8 may be programmed to output different voltages.

***Table 8-2* • DMPM-DB Components (continued)**

Serial Number	Name	Description
6	J3	ZL6105 programming header, used in conjunction with the Zilker programmer (ZLUSB EVAL1Z).
7	J23	FMC connector used to connect the SF2-DMPM-DB to the SmartFusion2 Development Kit board.

Note: DPOL1 to DPOL8 may be programmed to output different voltages.

APOL1, 2, 3, 4 - 3.3 V Regulated Supply, Microsemi Analog Mixed Signal Group NX9415

These synchronous buck switching voltage regulators provide a 3.3 V output (as configured) at up to 5 A. Different output voltages can be selected by changing the feedback voltage. For more information, refer to the NX9415 datasheet.

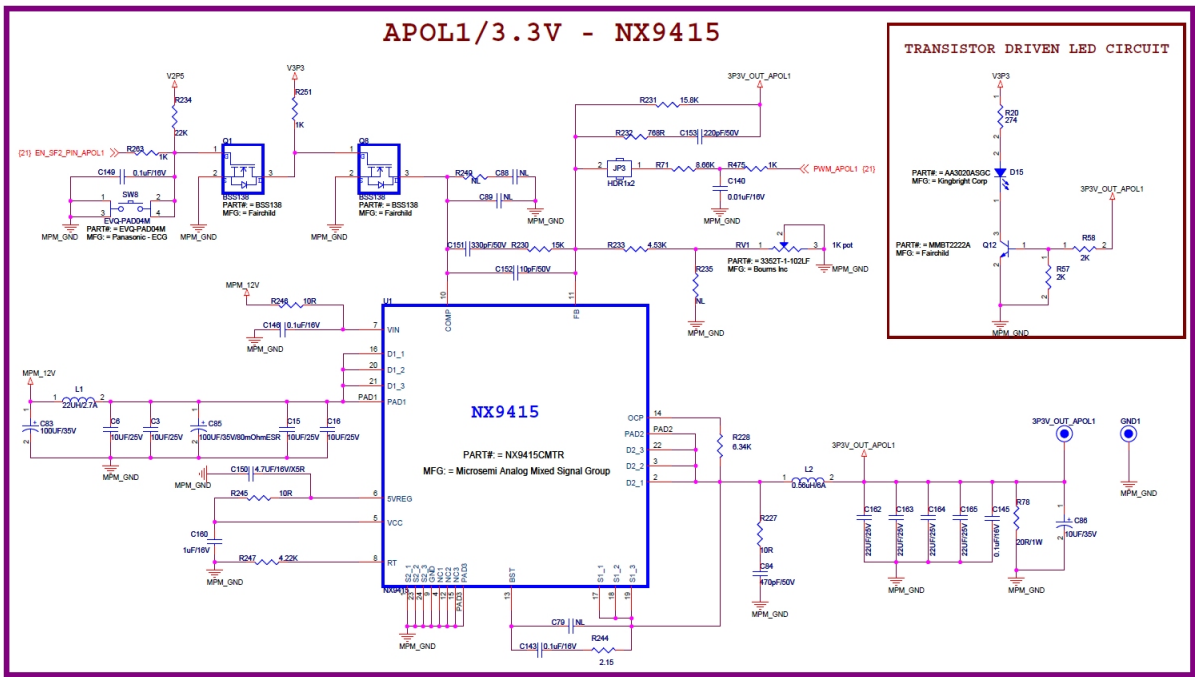

***Figure 8-3* • APOL1 3.3 V Switching Power Supply**

APOL5, 6, 7, 8 - 1.5 V Regulated Supply, Microsemi Analog Mixed Signal Group LX9610

These synchronous buck switching voltage regulators provide a 1.5 V output (as configured). Different output voltages can be selected by changing the feedback voltage. For more information, refer to the LX9610 datasheet.

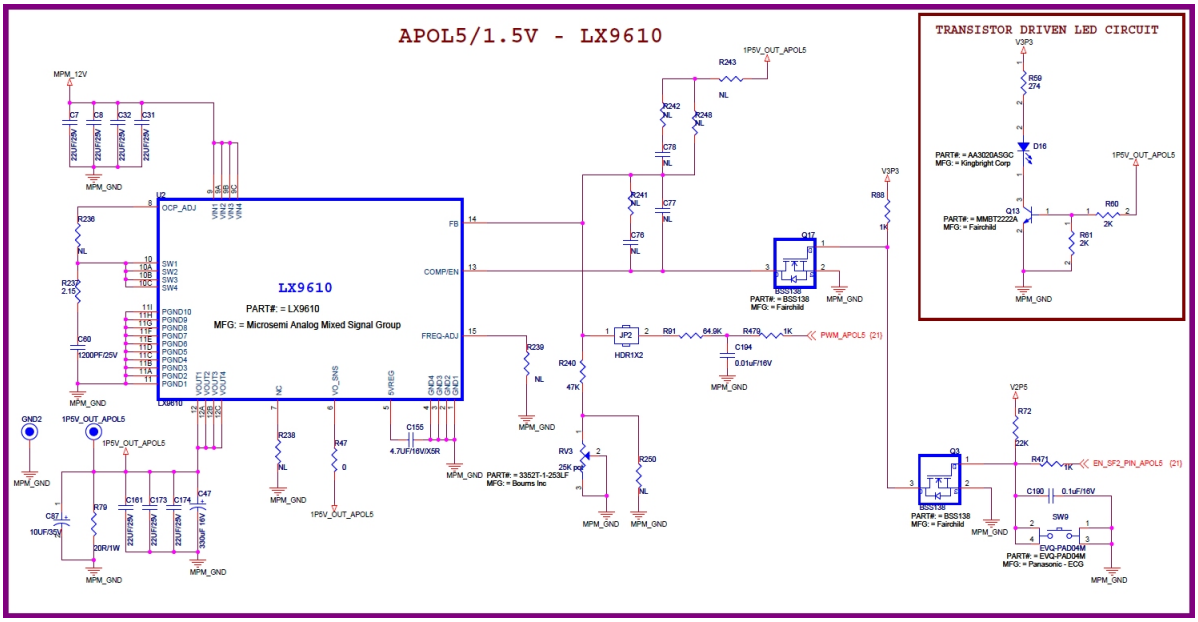

***Figure 8-4* • APOL5 1.5 V Switching Power Supply**

DPOL1, 2, 3, 4 - 3.3 V Intersil ZL6105 Digital Power Supply

These regulators are Intersil ZL6105 digital power controllers. The ZL6105 uses the I2C/PMBus 3-wire communication bus to connect to the host controller (in this case, MPM through the FMC connector on the SF2-DMPM-DB). These devices can be programmed to output voltages of 0.54 V to 5.5 V with currents of up to 3 A. Set the output voltage through the Zilker Labs Power Navigator. For more information, refer to the Intersil website and the ZL6105 datasheet.

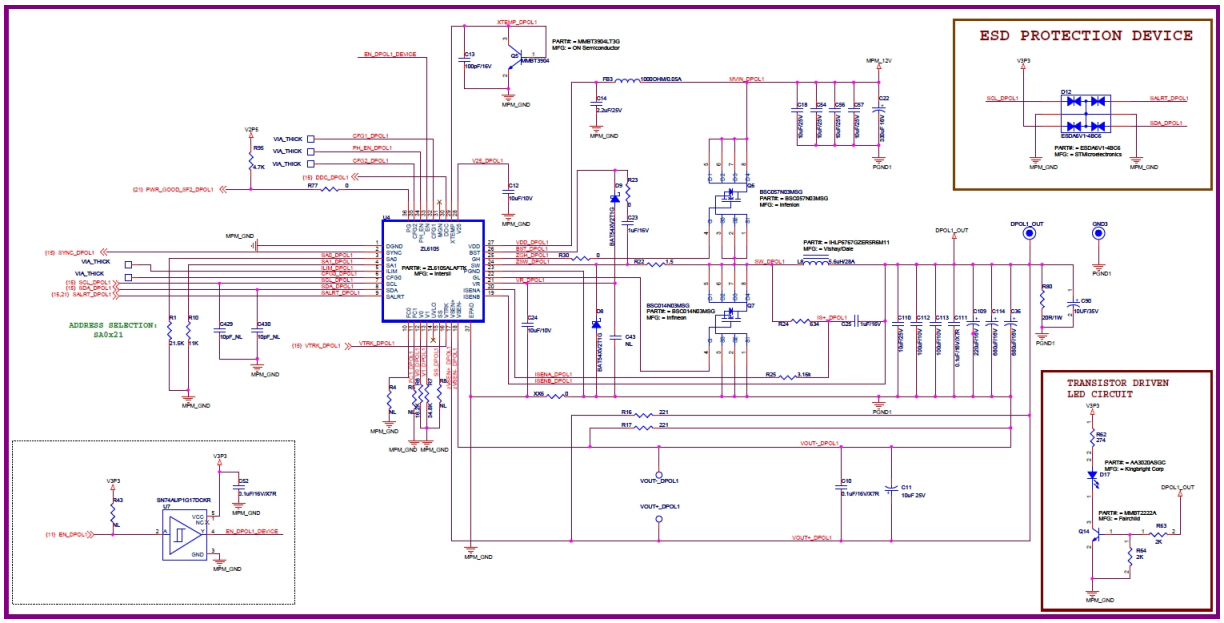

Figure 8-5 • **DPOL1 ZL6015 Regulator**

DPOL5,6,7,8 - 1.5 V Lineage PDT012A0X

The PDT0123A0X power module is a non-isolated DC-DC converter that can deliver up to 12 A of current, and can provide an output voltage ranging from 0.6 Vdc to 5.5 Vdc, programmable through a bootstrap resistor or PMBus configuration.

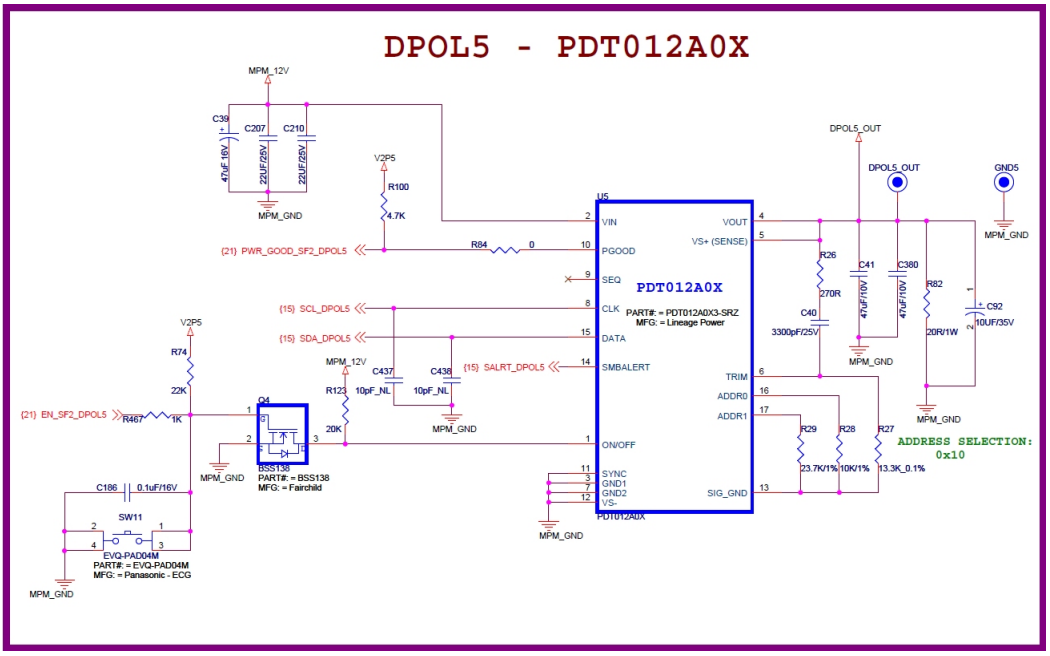

Figure 8-6 • **DPOL5 PDT012A0X**

Power

The SF2-DMPM-DB kit comes with a 12 V power supply, which connects to the supply header (J1).

FMC Connector

This connector is used to interface with the SmartFusion2 Development Kit board.

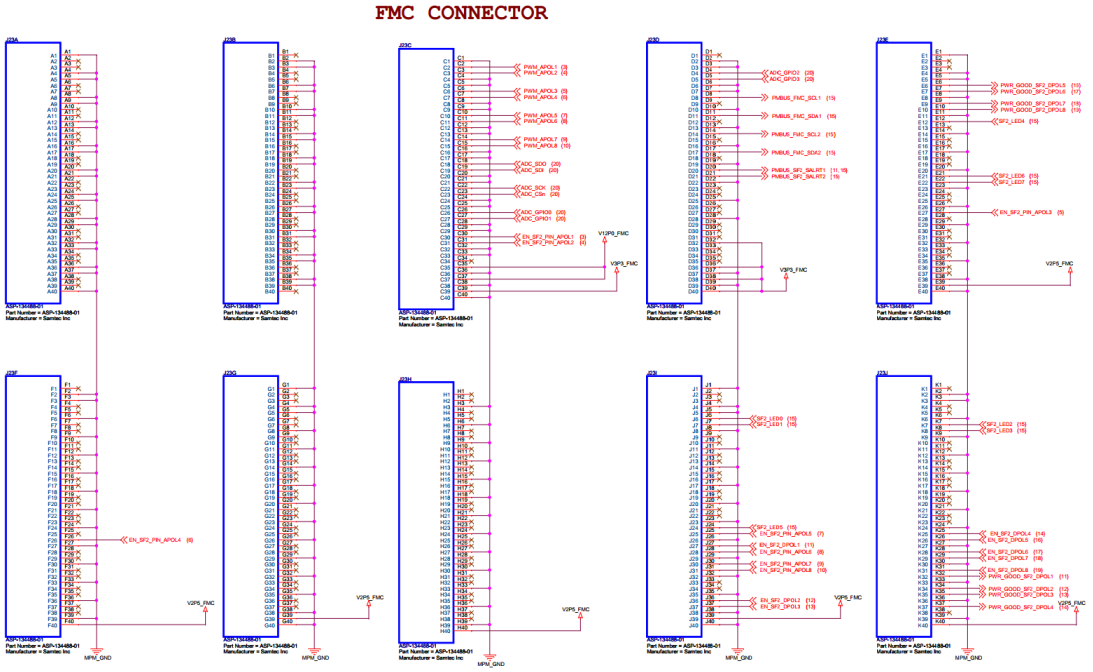

***Figure 8-7* • SF2-DMPM-DB Rev B FMC Connector**

Installation and Switch Settings

Recommended default jumpers settings are shown in Table 8-3. Connect jumpers in the default settings to enable the pre-programmed reference design to function correctly.

Note: Program the DPOL regulators to the default output voltages and settings defined in "Digital Mixed Signal Power Manager Board Components" section, and in the MPM GUI's default settings.

***Table 8-3* • Jumper Settings (DMPM-DB)**

Jumper	Function	Default Setting
J22	Connects DPOL1 to DPOL8 to PMBus (PMBus1 to PMBus2)	All DPOL's connected to PMBus1, pins shorted: A-B, C-D, E-F, G-H, I-J, K-L, M-N
JP3, JP26, JP27, JP28, JP2, JP29, JP30 and JP31	Connects APOL1 to APOL8 trim pin for closed-loop trimming.	Closed
JP21, JP22, JP32 and JP33	Connects DPOL1 to DPOL4 enable pin to FPGA or master enable	Pins 1-2 shorted

***Table 8-3* • Jumper Settings (DMPM-DB) (continued)**

Jumper	Function	Default Setting
JP23	Connects the 3.3 V rail on the SF2-DMPM-DB to either the FMC connector or the Zilker programming header.	Pins 1-2 shorted
JP25	Connects the 2.5 V rail on the SF2-DMPM-DB to the FMC connector	Closed

***Table 8-4* • Switches (DMPM-DB)**

Switch	Description
SW8	Push-button to disable APOL1
SW13	Push-button to disable APOL2
SW14	Push-button to disable APOL3
SW15	Push-button to disable APOL4
SW9	Push-button to disable APOL5
SW16	Push-button to disable APOL6
SW17	Push-button to disable APOL7
SW18	Push-button to disable APOL8
SW10	Push-button to disable DPOL1
SW12	Push-button to disable DPOL2
SW19	Push-button to disable DPOL3
SW20	Push-button to disable DPOL4
SW11	Push-button to disable DPOL5
SW21	Push-button to disable DPOL6
SW22	Push-button to disable DPOL7
SW23	Push-button to disable DPOL8
SW2	Switch ON 12 V DC into MPM

Note: The push-buttons detailed above ground the Enable pin of the corresponding regulator, thereby injecting failure in the power subsystem. The Enable pin is also connected to the pin on the FPGA. The FPGA could be damaged if it is driving High on the enable line and simultaneously shorted to the ground. To avoid this, Microsemi recommends that the FPGA pin driving these enables should be either driving low or tristate. Refer to the Libero design for details on how to do this.

9 – Manufacturing Information

MPM DB Board

The full PCB design layout is provided on the SF2 DMPM-DC Kit webpage. To view the PCB design layout files, use Allegro Free Physical Viewer, which can be downloaded from the Cadence Allegro Download page.

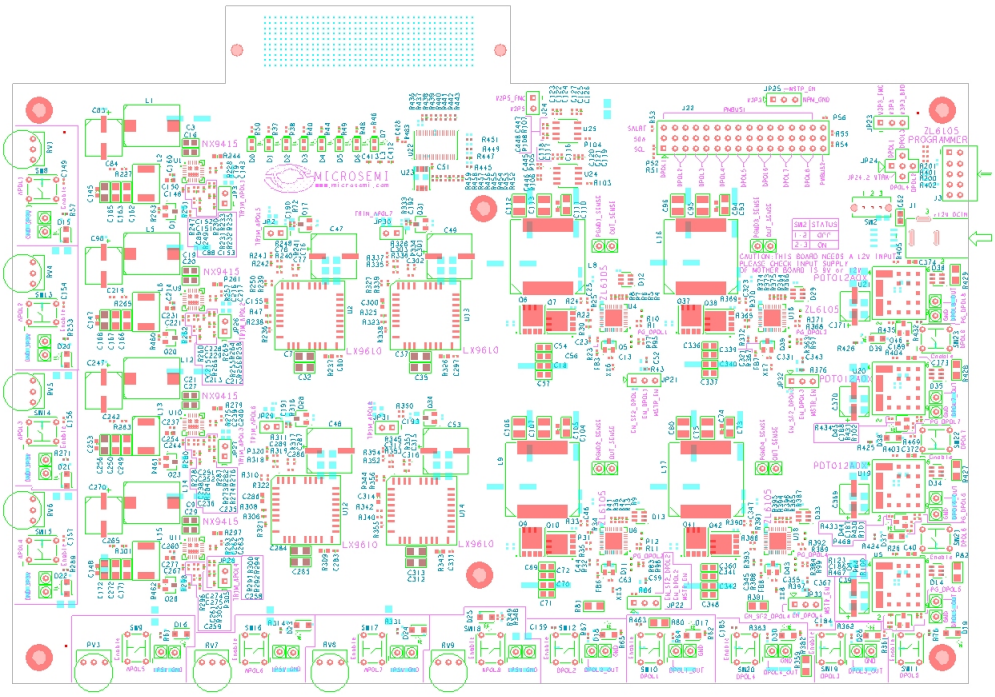

***Figure 9-1* • Top Silk Screen for SF2 DMPM Daughter Board Rev A**

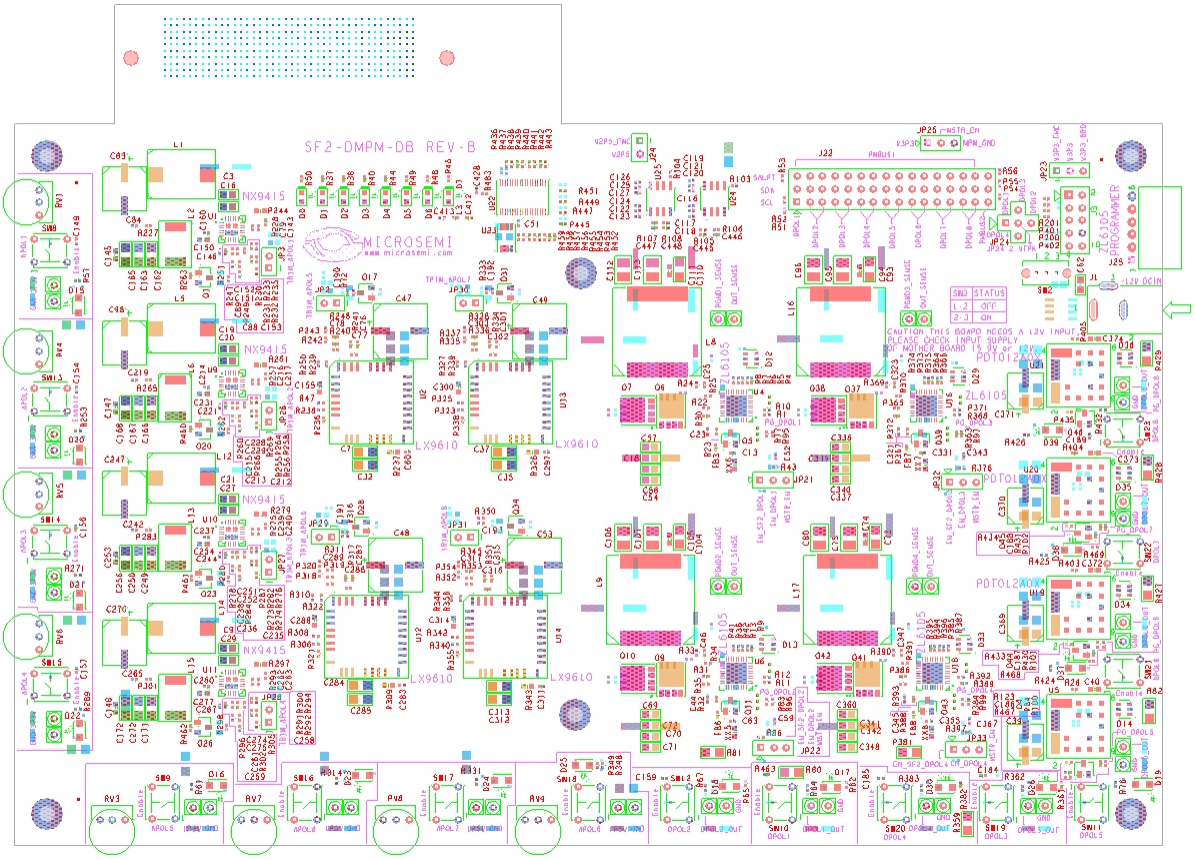

Figure 9-2 • **Top Silk Screen for SF2 DMPM Daughter**

A – Product Support

Microsemi SoC Products Group backs its products with various support services, including Customer Service, Customer Technical Support Center, a website, electronic mail, and worldwide sales offices. This appendix contains information about contacting Microsemi SoC Products Group and using these support services.

Customer Service

Contact Customer Service for non-technical product support, such as product pricing, product upgrades, update information, order status, and authorization.

From North America, call 800.262.1060
From the rest of the world, call 650.318.4460
Fax, from anywhere in the world, 408.643.6913

Customer Technical Support Center

Microsemi SoC Products Group staffs its Customer Technical Support Center with highly skilled engineers who can help answer your hardware, software, and design questions about Microsemi SoC Products. The Customer Technical Support Center spends a great deal of time creating application notes, answers to common design cycle questions, documentation of known issues, and various FAQs. So, before you contact us, please visit our online resources. It is very likely we have already answered your questions.

Technical Support

Visit the Customer Support website (www.microsemi.com/soc/support/search/default.aspx) for more information and support. Many answers available on the searchable web resource include diagrams, illustrations, and links to other resources on the website.

Website

You can browse a variety of technical and non-technical information on the SoC home page, at www.microsemi.com/soc.

Contacting the Customer Technical Support Center

Highly skilled engineers staff the Technical Support Center. The Technical Support Center can be contacted by email or through the Microsemi SoC Products Group website.

Email

You can communicate your technical questions to our email address and receive answers back by email, fax, or phone. Also, if you have design problems, you can email your design files to receive assistance. We constantly monitor the email account throughout the day. When sending your request to us, please be sure to include your full name, company name, and your contact information for efficient processing of your request.

The technical support email address is soc_tech@microsemi.com.

My Cases

Microsemi SoC Products Group customers may submit and track technical cases online by going to My Cases.

Outside the U.S.

Customers needing assistance outside the US time zones can either contact technical support via email (soc_tech@microsemi.com) or contact a local sales office. Sales office listings can be found at www.microsemi.com/soc/company/contact/default.aspx.

ITAR Technical Support

For technical support on RH and RT FPGAs that are regulated by International Traffic in Arms Regulations (ITAR), contact us via soc_tech_itar@microsemi.com. Alternatively, within My Cases, select **Yes** in the ITAR drop-down list. For a complete list of ITAR-regulated Microsemi FPGAs, visit the ITAR web page.

Microsemi Corporate Headquarters
One Enterprise, Aliso Viejo CA 92656 USA
Within the USA: +1 (949) 380-6100
Sales: +1 (949) 380-6136
Fax: +1 (949) 215-4996

Microsemi Corporation (NASDAQ: MSCC) offers a comprehensive portfolio of semiconductor solutions for: aerospace, defense and security; enterprise and communications; and industrial and alternative energy markets. Products include high-performance, high-reliability analog and RF devices, mixed signal and RF integrated circuits, customizable SoCs, FPGAs, and complete subsystems. Microsemi is headquartered in Aliso Viejo, Calif. Learn more at **www.microsemi.com**.

50200457-0/06-13