Mixed Signal Power Manager for SmartFusion Reference Design

User's Guide

Actel
POWER MATTERS

Actel Corporation, Mountain View, CA 94043

Printed in the United States of America

Part Number: 50200275-2

Release: September 2010

Trademarks

Table of Contents

Introduction

This document explains how to use the Mixed Signal Power Manager (MPM) reference design for SmartFusion™, using the MPM Daughter Card (MPM-DC) connected to either the SmartFusion Evaluation Kit or the SmartFusion Development Kit. You should read the SmartFusion Evaluation Kit User's Guide or the SmartFusion Development Kit User's Guide as appropriate for the configuration you are using with the MPM-DC.

MPM is a reference design that is programmed into a SmartFusion device and can be controlled and modified by the MPM graphical user interface (GUI), as well as via a standard 2-wire Inter-Integrated Circuit (I^2C) interface.

Based on Actel's mixed signal flash FPGA technology, MPM delivers superior power monitoring, power sequencing, closed-loop trimming, and power-up and power-down control of up to 22 external power supplies (external regulators). You do not need 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 you to configure power management and drive output signals as the monitored voltages meet or deviate from the user-programmed operating limits, all without opening the Actel Libero® Integrated Design Environment (IDE). This adds more flexibility, reduces total parts count on the board level, and increases system reliability by eliminating single points of failure.

With MPM, use of the Libero IDE tools is optional. Using the MPM GUI tool, you can configure a SmartFusion device that is programmed with the MPM reference design to revise setpoints and change sequencing without opening Libero IDE or reprogramming the FPGA fabric. The MPM GUI tool writes register values to on-chip embedded flash memory, which controls power sequencing and monitoring functionality of the MPM reference design.

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

System Summary

MPM for SmartFusion

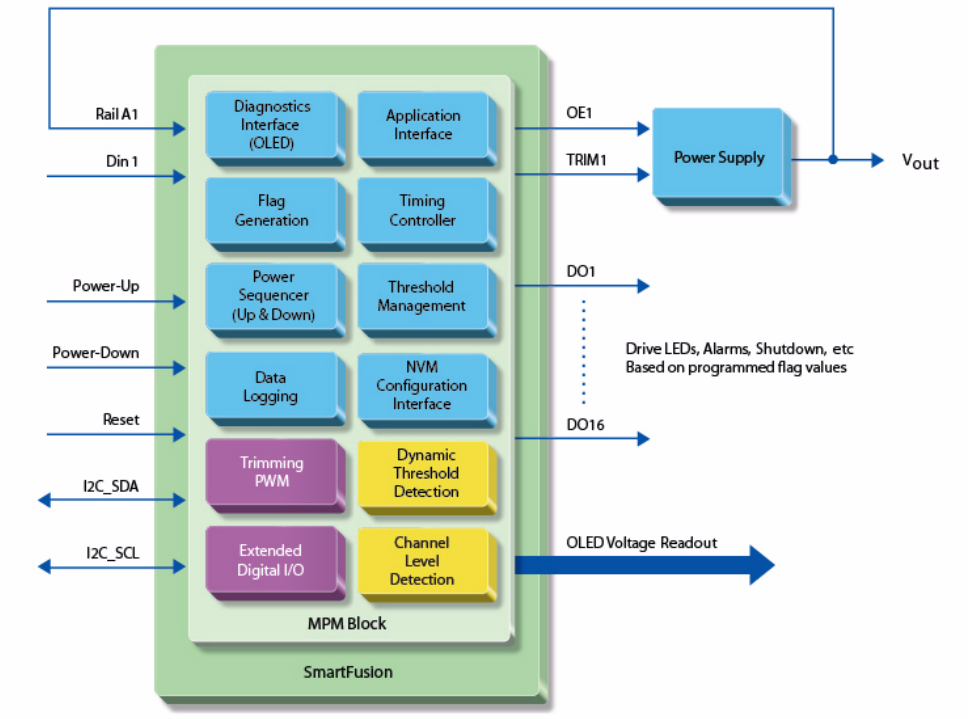

Figure I-1 **• Typical System-Level Diagram**

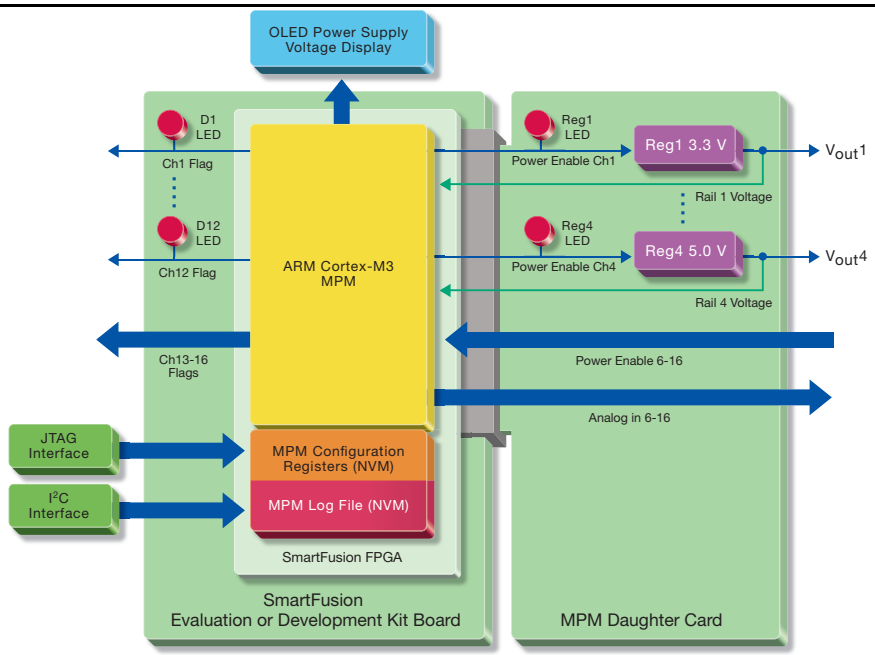

Figure I-2 **• MPM on the MPM Daughter Card (MPM-DC)**

1 – MPM Graphical User Interface (GUI)

Overview

MPM includes a Windows®-based executable standalone GUI (the MPM GUI) that enables you to access the register data stored in the FPGA NVM. MPM configuration data, including voltage limits, digital input and outputs settings, and power sequencing settings, is stored in NVM.

The following sections provide detailed procedures for using the GUI to install, configure, and run the MPM for SmartFusion designs.

MPM for SmartFusion

Installation

Run the installer and follow the installation wizard instructions. The installer adds Windows Start Menu entries, allowing you to run the MPM GUI, browse the design files, and uninstall the package. The MPM GUI also requires that the FlashPro software be installed. You can download FlashPro using the download link at www.actel.com/download/program_debug/flashpro/default.aspx.

Using the SmartFusion MPM Demonstration

Having run the installer, you can immediately try out the bundled demonstration program that illustrates some of the features of SmartFusion MPM.

Start by connecting your A2F-EVAL-KIT/A2F-DEV-KIT with your MPM-DC using the mixed signal headers (J21). To demonstrate open-loop or closed-loop trimming, the following jumpers must be installed on the MPM-DC:

- Regulator trimming circuits: JP12, JP13, JP15, JP14
- Trimming DAC option jumpers. For each of the following, install a jumper on pin 1-2 (on Brd PWM-center): JP19, JP20, JP17, JP18

Connect the 9 V power supply to the MPM Daughter Card and the USB cables to the parent board.

Note: Two USB connections are required for the A2F-EVAL-KIT board: one for communications and the second to power the board. There is no power supply connection to the A2F-EVAL-KIT. The A2F-DEV-KIT uses a single USB connection and a separate power supply.

Switch on power for both boards. Now start the MPM GUI by double-clicking on the MPM GUI icon or selecting **Start > Actel SmartFusion MPM Reference Design > MPM GUI**.

Load the demonstration configuration settings by selecting **File > Load Values** and browse to <SF_MPM_RefDesign_Root>\bin\ SF_MPM_Reference_Design.txt.

The first time you select **File > Write NVM** or **File > Write NVM & Fabric**, the MPM GUI will prompt you to locate the FlashPro software executable. If the board has not been programmed with the MPM design previously, you must load the MPM design into the SmartFusion FPGA.

To write the MPM design and all current configuration settings to the SmartFusion device, select **File > Write NVM & Fabric**.

The following design elements are written to the SmartFusion device on the parent board:

- Fabric logic FDB file
- SmartFusion MSS configuration EFC file
- MPM firmware (stored in an MSS eNVM data storage client by default at eNVM address offset 0x00000000)
- MPM configuration data (stored in an MSS eNVM data storage client by default at eNVM address offset 0x0003F000)

After the MPM design is loaded, you can use **File > Write NVM** to change/update the MPM configuration settings. After you have programmed the demonstration design and MPM configuration settings, the A2F-EVAL-KIT/A2F-DEV-KIT board OLED displays a prompt indicating that the MPM demonstration program is running.

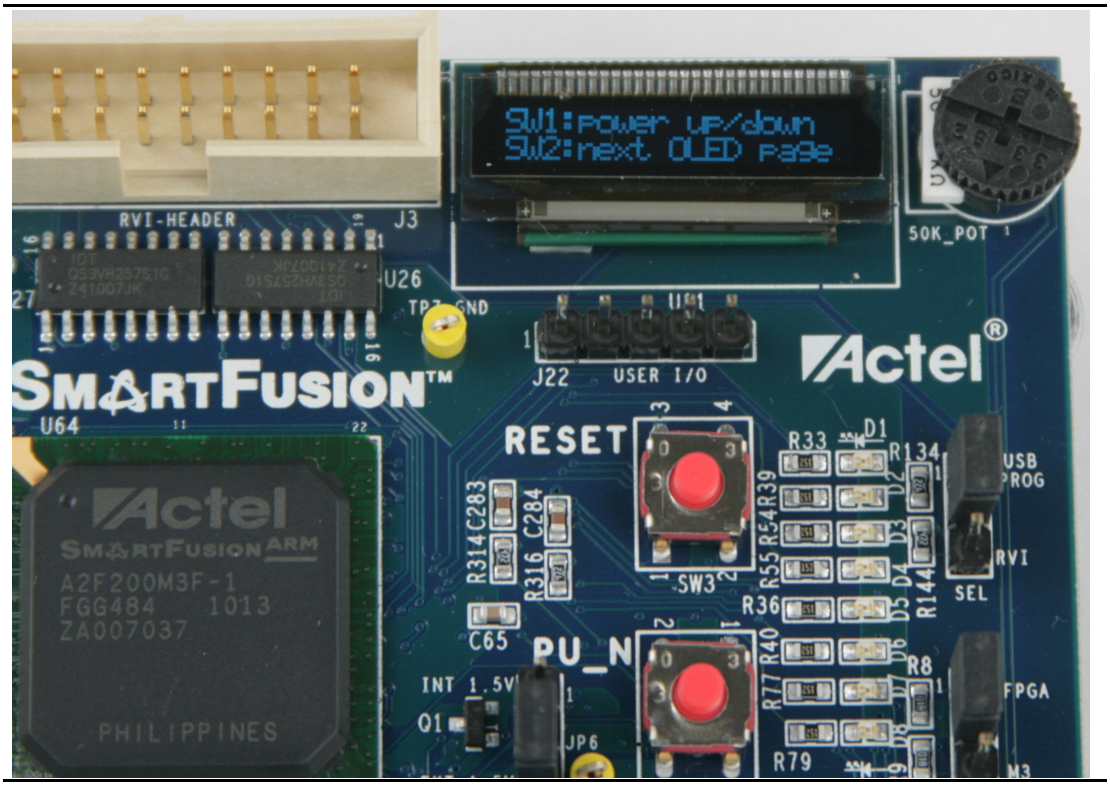

***Figure 1-1* • OLED Message After Successful Programming**

Use push-button switch SW1 on the parent board to toggle power-on and power-off sequencing and SW2 to cycle between the various OLED display pages:

- Help (only displayed once)
- Version (only displayed once)
- MPM status

***Table 1-1* • MPM Status Description**

Status	Description
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 remain operational.
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.

- Channel 1 to Channel n voltage and threshold based state (OFF, UV2, UV1, NOMINAL, OV1, OV2)
- Outputs [15:0] and Outputs [31:16] reflect the current state of the output flags digital output lines where a 1 is represented by a ‘o’ character and a 0 is represented by a ‘.’ For example, 0x1234 = “...o..o...oo.o..”). In the demonstration, the following output flags are connected to LEDs:
  - Output 1: MPM-DC LED D0
  - Output 2: MPM-DC LED D1
  - Output 3: MPM-DC LED D2
  - Output 4: MPM-DC LED D3
  - Output 5: MPM-DC LED D4
  - Output 6: MPM-DC LED D5
  - Output 7: MPM-DC LED D6
  - Output 8: MPM-DC LED D7
  - Output 9: A2F-EVAL-KIT/A2F-DEV-KIT LED D1
  - Output 10: A2F-EVAL-KIT/A2F-DEV-KIT LED D2
  - Output 11: A2F-EVAL-KIT/A2F-DEV-KIT LED D3
  - Output 12: A2F-EVAL-KIT/A2F-DEV-KIT LED D4
- Inputs [15:0] and Inputs [31:16] reflect the current state of the general purpose digital inputs that can be combined into the digital output flag generation logic. The same representation of 1 and 0 bits is used as for Outputs [15:0]/[31:16].
- Regulator Enables [15:0] and [31:16] reflect the current state of the regulator enable digital outputs using the same representation as before. In the demonstration design, the following regulator enables are connected:
  - Regulator Enable 1: MPM-DC REG1
  - Regulator Enable 2: MPM-DC REG2
  - Regulator Enable 3: MPM-DC REG3
  - Regulator Enable 4: MPM-DC REG4

Using SW2, cycle the OLED display back to the Status page and select SW1 to initiate power-up sequencing. The state changes to Starting and you can see the various regulator enabled LEDs on the MPM Daughter Card turning on. If the status does not change to Started and the power-on sequence restarts, cycle through the individual channels to see which one is not reaching nominal when switched on. Adjust the POT to ensure that it reaches nominal value. If open-loop trimming is enabled, open-loop trim pin voltage will only achieve nominal value if the POT is suitably adjusted.

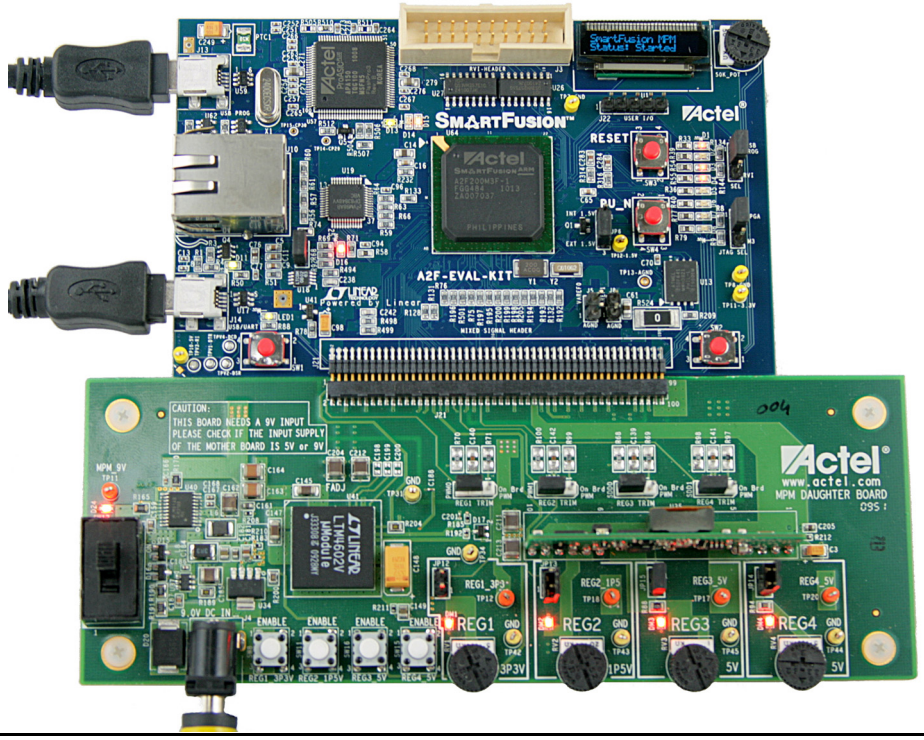

***Figure 1-2* • MPM-DC-KIT MPM Daughter Card Attached to SmartFusion Evaluation Kit**

When power sequencing has completed and all regulators have reached nominal voltage, the status changes to Started and closed-loop trimming is also enabled. Closed-loop trimming keeps the channel output voltage at the nominal value specified in the GUI, even when the POT is adjusted. You can disable trimming by removing the Trim jumper for a given regulator. Removing JP12 for regulator 1, JP13 for regulator 2, JP14 for regulator 3, or JP15 for regulator 4 disables closed-loop trimming and you see that the output voltage can be varied using the POT for that channel. Reinstalling the jumper reactivates closed-loop trimming and brings it back to nominal.

Channel A5 is connected to the A2F-EVAL-KIT/A2F-DEV-KIT board RV1 POT-controlled 3.3 V circuit and the voltage can be freely adjusted between 0 V and 2.56 V using the POT. This voltage channel is hardwired to a SmartFusion ACE direct analog input channel which is restricted to a range of 0-2.56 V.

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

- Channel A1
  - MPM-DC REG1
  - National Semiconductor LM3100MH 3.3 V nominal regulator
  - POT range approximately 2592 – 3552 mV
  - ACE ABPS2 ± 5.12 V channel
- Channel A2
  - MPM-DC REG2
  - National Semiconductor LP38693MP-ADJ 1.5 V nominal regulator
  - POT range approximately 1350 – 1672 mV
  - ACE ABPS6 ± 2.56 V channel
- Channel A3
  - MPM-DC REG3
  - Artesyn ATA010A0X3Z 5 V nominal regulator
  - POT range approximately 3960 – 5464 mV
  - ACE APBS3 ± 10.24 V channel
- Channel A4
  - MPM-DC REG4
  - Linear Technologies LTM4602EV 5 V nominal regulator
  - POT range c. 3832 – 4928 mV
  - ACE ABPS7 ± 10.24 V channel
- Channel A5
  - A2F-DEV-KIT/A2F-EVAL-KIT board RV1 circuit
  - 0 – 3.3 V range
  - POT range 0 – 2.56 V
  - ACE ADC Direct Input 0 – 2.56 V channel

Note: If you use the MPM GUI to configure any of the voltage settings (for example, thresholds ± hysteresis) out of range of the relevant underlying ACE channel, the MPM firmware will ignore these as invalid and not display them. For example, if you set the OV2 on Channel 5 to 3300 mV, this channel will disappear. Even though the RV1 POT on the A2F-EVAL-KIT/A2F-DEV-KIT boards has a range of 0 – 3.3 V, it is hardwired to an ACE direct analog input channel with a range of only 0 – 2.56 V.

Reference Design Firmware

A SoftConsole workspace containing the reference design firmware and demonstration program application code is included. To access this, do the following:

1. Make a backup or work copy of the original SoftConsole workspace.
2. Run SoftConsole v3.1.
3. **File > Switch Workspace > Other…**
4. Browse to the reference design SoftConsole workspace folder. For example, C:\Actel\SF_MPM_RefDesign_v2.0\design_files\SoftConsole_workspace\SF_MPM_RefDesign.
5. Click **OK**.

SoftConsole reopens using the reference design firmware workspace, which contains a single project implementing the reference design demonstration 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 see how the MPM driver is used by the demonstration program. Note that the demonstration program also includes other firmware cores used directly by the application code or MPM driver. For example, OLED sample application code built on top of the MSS I^2C driver, MSS GPIO driver, MSS Watchdog driver, and the MPM driver includes the MSS ACE driver, MSS Timer driver (TIM2), fabric-based CoreGPIO/CorePWM drivers, and others.

The mpm/mpm.h files describe the public interface to the MPM driver.

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

MPM Driver API

The MPM driver API is quite simple and described in the mpm/mpm.h.

```
/* 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();

/* 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 pased 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(32,
 *   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()
 * uint32_t mpm_get_digital_outputs()
 * uint32_t mpm_get_regulator_enable_outputs()
 *
 * Returns a 32 bit bitmask [31:0] representing the current state of the
 * relevant digital I/Os.
 */
uint32_t mpm_get_digital_inputs();
uint32_t mpm_get_digital_outputs();
uint32_t mpm_get_regulator_enable_outputs();
```

Reference Design Hardware

A Libero IDE project implementing the MPM hardware is included. To access this MPM Libero IDE project:

1. Make a backup or work copy of the original Libero IDE project bundled with the package.
2. Run Libero IDE v9.0 SP1.
3. Browse to the Libero IDE project C:\Actel\SF_MPM_RefDesign_v2.0\design_files\Libero_project\SmartFusion_MPM_Reference_Design.
4. Select SmartFusion_MPM_Reference_Design.prj and **Open**.
5. If you receive any warnings about missing IP cores, make sure to download them from the repository to the vault using the Libero IDE catalog.
6. The design comprises a top-level SmartDesign that instantiates the SmartFusion MSS and several fabric-based peripherals.
   - SmartFusion MSS
     - MPM ACE channel and related configuration
     - MSS GPIOs used for interfacing to SW1/2 push-buttons and MPM-DC LEDs
     - MSS eNVM data storage client placeholders for MPM configuration data and MPM firmware
     - MSS SSD for hard-macro driven trimming. The MSS SDD operates between 0V and 2.56V, while Core PWM (below) opearates between 0V and 3.3V.
   - Fabric-based logic
     - CoreAPB3 for interfacing MSS to fabric DirectCore peripherals
     - CoreGPIO (MPM_GPIO_Digital_IOs) implementing up to 32 general digital inputs and 32 flag digital outputs
     - CoreGPIO (MPM_GPIO_Regulator_Enables) implementing up to 32 regulator enable digital outputs
     - CorePWM (MPM_PWM_Trimming_Outputs) implementing up to 16 trimming PWM channels
     - Other ancillary logic; For example, TRIBUFFs for driving the MPM-DC regulator enable digital outputs

Note: In the reference design, some but not all output flag digital outputs are connected to LEDs, and some but not all regulator enable digital outputs are connected to regulator enables.

MPM ACE Configuration

MPM channels are configured in the ACE as follows:

1. Signal name must be prefixed with ‘MPM_Channel_<n>…’, where ‘…’ represents any other text needed to name the ACE channel signal and <n> is a unique MPM channel number between 1 and min (32, MPM_MAX_NUMBER_OF_CHANNELS).
2. One OVER threshold named UP and one UNDER threshold named DOWN must be configured. The initial voltage values for these thresholds are not important and can be configured to any valid value. The MPM driver dynamically reprograms these on the fly when managing the channels. For hysteresis-based thresholds, the MPM GUI specified hysteresis value will be used. State filtered (assert/deassert samples) based thresholds are used as is in the ACE configuration.

Channels meeting these criteria that are not given ‘out of range’ threshold ± hysteresis configurations via the MPM GUI will be recognized and managed by the MPM firmware.

Note: While modifying/adapting the reference design if ACE configuration is changed, remember to copy the MSS configurator generated drivers_config/mss_ace folder into your SoftConsole project to ensure that the firmware and hardware are always kept in synchronization.

Running the MPM for SmartFusion Reference Design Demo

1. Connect your A2F-EVAL-KIT/A2F-DEV-KIT parent board to your MPM Daughter Card.
2. Ensure that the following MPM DC jumpers are installed:
   – JP17-JP20: Pins 1-2 (on Brd PWM-centre), – in order to ensure that the regulator trimming circuits are connected to the fabric-based CorePWM outputs from the A2F200 (Refer to the "MPM for SmartFusion Reference Design Trimming" on page 49).
   – JP12-JP15: Jumpers on all of these in order to enable (where configured) open-loop trimming during power-on sequencing and closed-loop trimming thereafter.
3. Plug in the power supply as required, being careful to connect the 9 V power supply to the MPM-DC only. Switch on power to both boards.
4. Run the SF MPM reference design installer and click through the wizard.
5. Run the MPM GUI (from the installer or from the Start menu)
6. Load the demo configuration settings (SF_MPM_Reference_Design.txt), which should be listed at the bottom of the File menu.
7. The first time, choose **File > Write NVM & Fabric** to program the whole design (fabric logic FDB, MSS configuration EFC, MPM firmware, and configuration data). You will be prompted to locate the FlashPro executable.
8. Thereafter, assuming you don't change the Libero IDE project or SoftConsole project) you select **File > Write NVM**.
9. If all goes well, you should see the SF MPM reference design come to life on the OLED. The first OLED page displayed should tell you that you use SW1 to alternately select power on or off and SW2 to cycle between the different OLED pages.
10. The first two OLED pages (Help and Version Information) are only displayed once and not again after you have cycled through all OLED pages.
11. You can open the bundled SoftConsole project to see what the demo application code main.c looks like. You can also look at the hardware design and the implementation of the MPM engine driver in the SoftConsole project.
12. There are a few GUI items that are basically ignored by the current firmware:
    – DAC Type (only fabric-based CorePWM supported at the moment – no ACE SDD/OBD)
    – Trimming Control > Startup Delay

GUI Configuration Tabs

Power

The **Power** tab (Figure 1-3) is used to assign requirements for each voltage rail to be controlled, such as the nominal voltage, threshold/flag voltages, and power-up/-down sequences. For more information, refer to the side panel Help by clicking on the associated boxes.

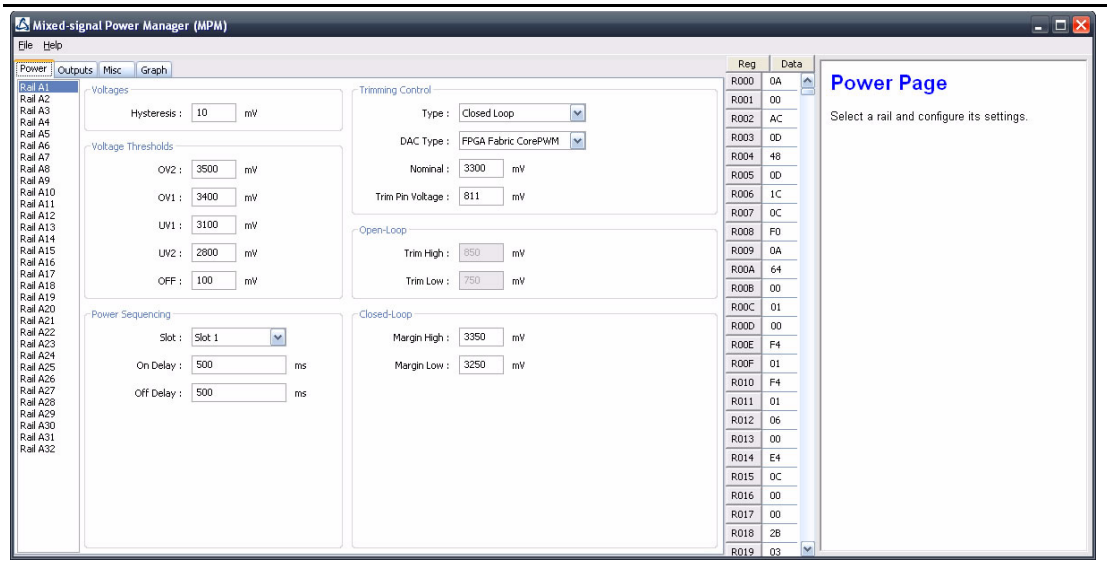

***Figure 1-3* • Power Tab**

Outputs

The **Outputs** tab (Figure 1-4) is used to define the condition for and polarity of the assertion of 16 digital outputs. This tab also enables you to utilize the 16 digital inputs to control the output logic via the digital input panel. For more information, refer to the side panel Help by clicking on the related boxes.

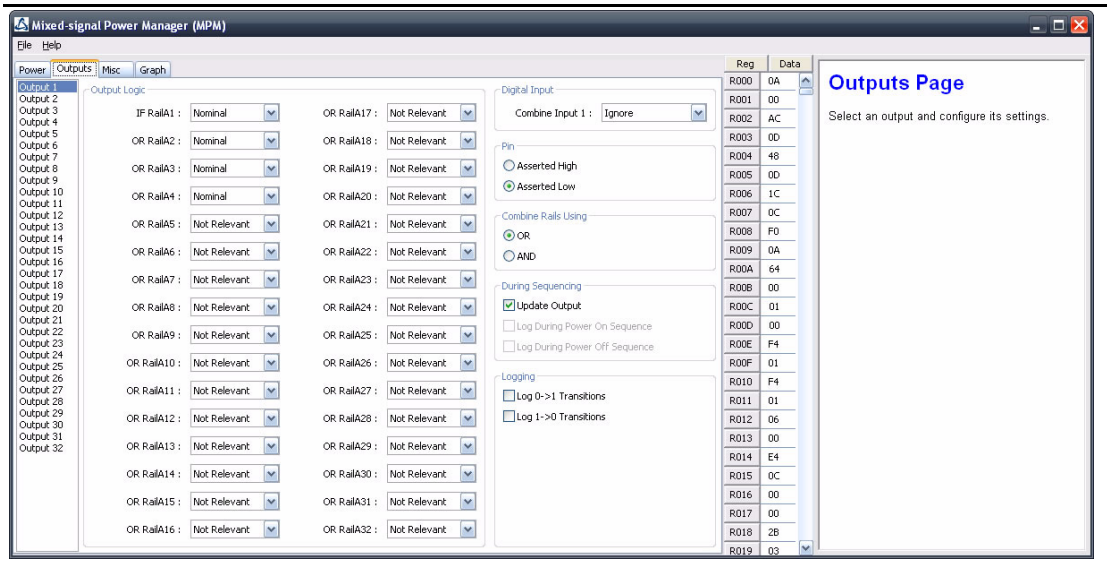

***Figure 1-4* • Outputs Tab**

Misc

The **Misc** tab (Figure 1-5) is used to configure calibration settings and actions on power-up and power-off. For more information, refer to the side panel Help by clicking on the related boxes.

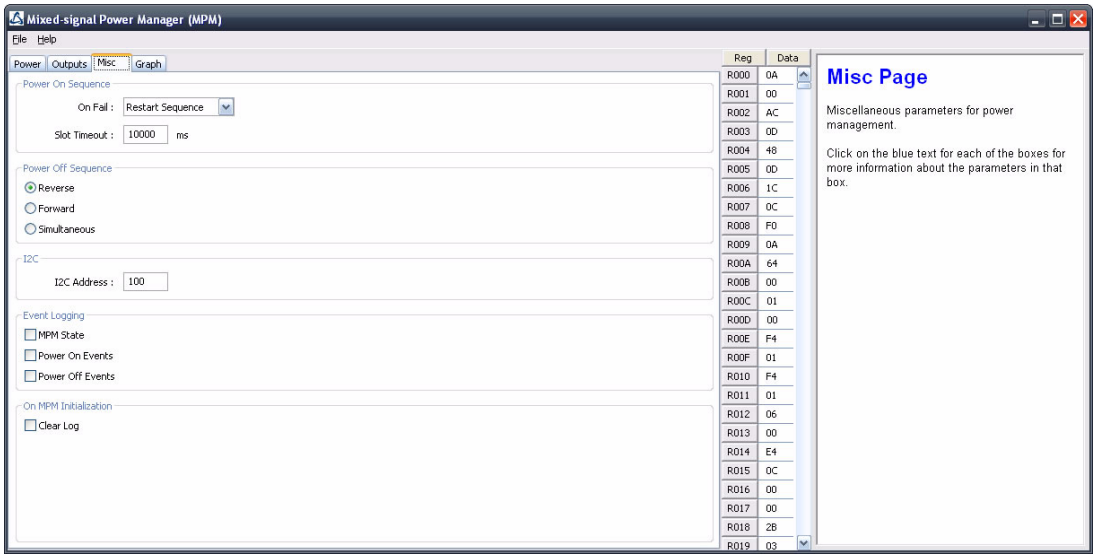

***Figure 1-5* • Misc Tab**

Graph

The **Graph** tab (Figure 1-6) is a visual representation of the power-up and power-down sequencing. 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 on the **Power** tab. Rise and fall times are not accurate and voltage levels are not taken into account. For more information, refer to the side panel Help by clicking on the related boxes.

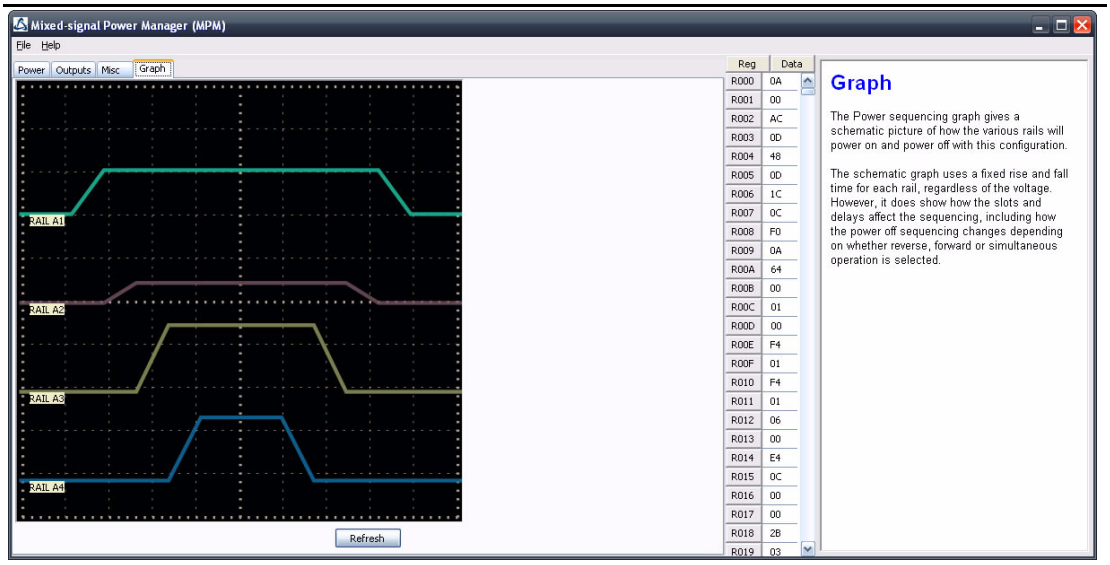

***Figure 1-6* • Graph Tab**

Programming the Device

In the MPM GUI, the column of buttons marked R and followed by a number (Figure 1-7) displays a register map for the configuration data. This relates to how the parameters are stored in the NVM of the FPGA.

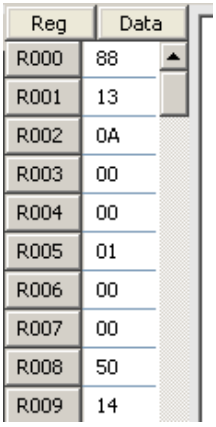

***Figure 1-7* • Register Data**

The File menu contains the “Write NVM & Fabric”, “Write NVM”, “Choose STAPL File”, “Locate Flashpro Executable”, “Create Memfile” and “Check Config” menu options.

Write NVM & Fabric programs the NVM register values and the FPGA device with the MPM circuit design. Write Device is required when the FPGA device is either blank or programmed with a different design. You can perform updates to flags and settings using the Write NVM menu option.

Write NVM writes the configuration created using the various tabs in the MPM GUI and programs the

register values shown in the column of buttons marked R, followed by a number.

Choose STAPL File allows the user to choose a different STAPL file than the default so that, for example, the user can select between the A2F200 and A2F500 implementations of MPM on the fly.

Locate Flashpro Executable allows the user to select the location of the Flashpro binary, whether it be a standalone version of Flashpro or the version bundled with an Actel Libero installation.

Create Memfile creates a memory content file, which is used to program NVM. This is automatically done when either Write NVM or Write NVM & Fabric is invoked.

Check Config prints the current MPM GUI configuration to the window pane on the right side of the GUI.

Connections, Jumper Switches, and Settings

Connect jumpers in the default settings to enable the MPM for SmartFusion design to function correctly before powering up the boards.

- Jumpers from FB (feedback voltage) of each regulator for voltage trimming
  - JP12, JP13, JP14, JP15 – Populated with jumper
- Jumpers for selection of PWM or SDD
  - JP17, JP18, JP19, JP20 – Connect pins 1-2
- Push-button switches to disable each regulator and simulate a power failure
  - SW8, SW11, SW15, SW16

Push-button fault injection switches SW8, SW11, SW15, and SW16 ground the enable pin of the corresponding regulator, thereby injecting failure in the power subsystem. However, this enable pin is also connected to the pin on the SmartFusion FPGA (MPM_REG1_EN, MPM_REG2_EN, MPM_REG3_EN, MPM_REG4_EN).The FPGA could be damaged if configured to drive High on the enable line when the fault injection switch is pressed. To avoid this, Actel recommends that the FPGA pin driving this enable should be driving Low or tristated.

MPM Reference Design Description

The MPM demonstration design takes advantage of the processing power and programmable flexibility of the SmartFusion intelligent mixed signal FPGA on the SmartFusion Evaluation Kit or Development Kit. Connect either kit to the Daughter Card and download the software as described to run the demo.

The Daughter Card consists of four regulated power supplies running from a 9 V supply.

- Switching regulator 1.5 V
- Switching regulator 3.3 V
- DC-DC regulator 5 V
- DC-DC regulator 5 V

Using the MPM design one can do the following:

- Monitor voltage for all rails
- Sequence different power rails for power-up and power-down
- Trim and margin a voltage rails in a closed-loop manner
- Sweep the output voltage (POT circuit to change resistor on feedback voltage)
- Induce failures by disabling the enable input of regulator (push-button to GND enable)

Power sequencing is done by sequentially asserting or deasserting channel enable pins for power-up and power-down, respectively, and monitoring the associated channel voltage. All enable pins to the regulator are active high.

Software for MPM Reference Design

Download the MPM executable that contains the reference design from the Actel website: www.actel.com/products/hardware/devkits_boards/mpm_dc.aspx.

Run the reference design executable. This installs the MPM GUI. From the MPM GUI you can program the demo design into the SmartFusion device and configure your power sequence or trimming. You can update the sequence by uploading only the NVM register locations after the first time you program the main design to the board.

Running the MPM for SmartFusion Reference Design

With the SmartFusion Evaluation Kit

Connect the MPM Daughter Card to the A2F-EVAL-KIT board. Connect both USB cables from your PC to the evaluation kit. One USB provides power and UART connection; the other provides the programming connection. If you are using the Daughter Card with the evaluation kit, you can remove the plastic legs and add the rubber feet supplied for this purpose.

With the SmartFusion Development Kit

Connect the MPM Daughter Card to the A2F-DEV-KIT board. Connect the 9 V power supply to the MPM Daughter Card and the 5 V power supply to the development kit. It is a good idea to double check the voltages before connecting the supplies. Connect the low-cost programming stick to the development kit and connect with a USB cable to the PC for programming. Also connect the other USB cable to the USB connection on the board and to your PC. From the MPM GUI, select **File > Write Device** to load the MPM design to the device for the first time. Use the switches on the evaluation or development kit to activate the power-up sequence. You can then create interrupts and change POT settings to review the performance of the board. The OLED shows the voltage of each of the first four power rails.

To change the power sequence, use the GUI to change the settings and then do **File > Write NVM** to update the register settings.

2 – MPM for SmartFusion Reference Design Flow

Building the Firmware Image

The SmartFusion MPM Reference Design release contains a SoftConsole project, that may be modified by you, located at <SFMPM_directory>/design_files. For information on the MPM driver and how to use/modify it, refer to the "MPM Driver API" section on page 12.

Once the source has been compiled and tested, you can build a final version for deployment in NVM by selecting **Project > Build Configurations > Set Active > Release** in the SoftConsole workspace. Setting the build type to Release will cause the production-execute-in-place.ld linker script to be used, which will create a firmware for storage in and execution from NVM, mirrored to address 0x00000000 on reset. The figure below shows a screenshot of this option.

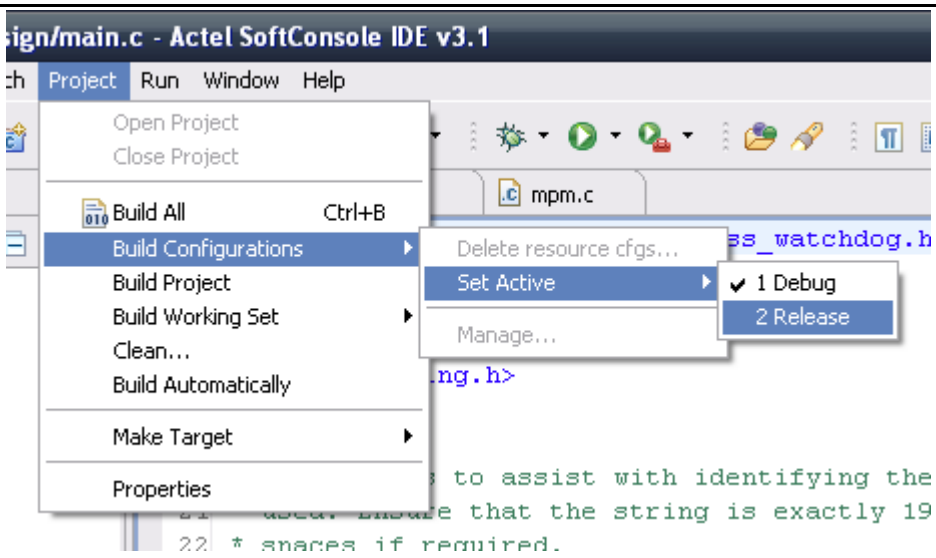

***Figure 2-1* • Setting Active Build to Release in SoftConsole**

At this point you may build the project. One of the steps of the MPM project build 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 "NVM Space/Data Client Management" section on page 22 for this step.

NVM Space/Data Client Management

In the SmartFusion MPM Reference designs, there are four distinct NVM spaces, three of which are defined in the microcontoller subsystem (MSS) configurator's NVM configuration GUI as data storage clients in the provided Libero project, picture below (MPM_Configuration, MPM_Event_Log, and MPM_Firmware). The third (MSS Configuration Data) does not require a Data Storage Client, as it is generated by default and included in the programming flow (more on programming in "Programming File Generation and GUI Integration" section on page 27).

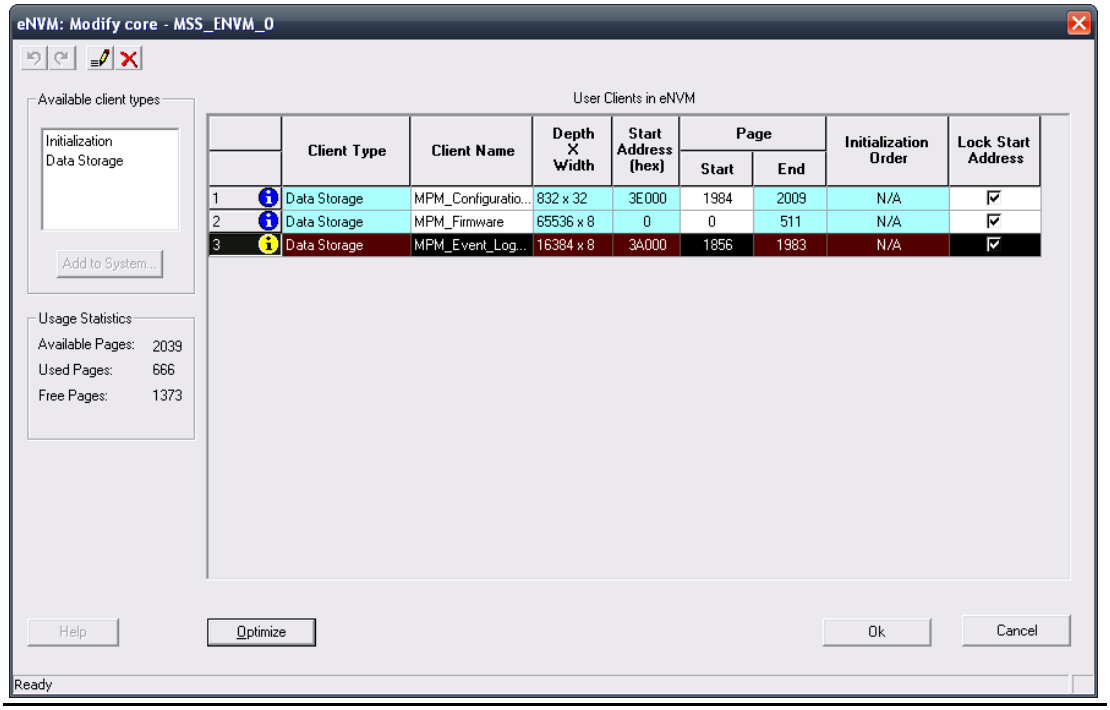

***Figure 2-2* • MSS NVM Configuration GUI**

MSS Configuration

The SmartFusion MSS generates NVM configuration data that is stored in the NVM starting at address 0x000000. The size, however, is variable, and depends on the complexity of the configuration as set in the SmartDesign MSS Configurator.

To generate, you should select the Generate option in the MSS configurator, as shown below.

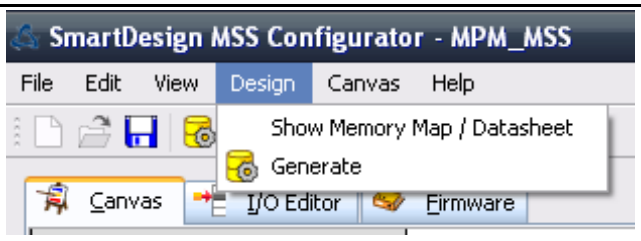

***Figure 2-3* • Generating MSS**

The following message should appear when the MSS is generated:

```
Info: Design "MPM_MSS" was successfully generated.
Netlist generated to <path\to>\MPM_MSS\MPM_MSS.vhd
Firmware files are generated to <path\to>\firmware
NVM programming file generated to <path\to>\MSS_NVM_0\MSS_NVM_0.efc
```

Note: If there are any updated firmware files (if a new version of a firmware core has been selected in the MSS configurator), it is your responsibility to copy the firmware files to the appropriate location in the MPM SoftConsole project, and to rebuild the firmware hex file.

As mentioned, this region of NVM does not have its own Data Storage Client, since an embedded Flash memory content file (EFC) is automatically generated by the MSS configurator in SmartDesign and automatically included during programming file generation.

MPM Configuration Data

The MPM configuration data is stored in NVM. By default, with the MPM driver unchanged, it has a base address in the NVM of 0x3EF00 and consists of 832 32-bit words.

Note: This is the default MPM configuration data size. If you add NVM configuration registers in the firmware, this may grow larger and would need to be adjusted as such.

The included Libero project contains a Data Storage Client (visible under the MSS configurator, see above) that is pre-configured with the above parameters. Size increases from firmware changes need to be reflected in this Data Storage Client, the GUI of which is shown below.

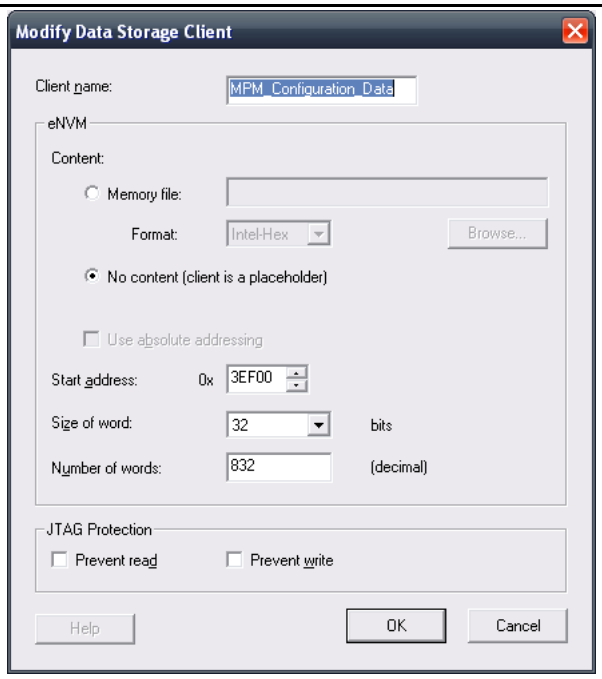

Figure 2-4 • **Data Storage Client for MPM Configuration Data**

The base address of 0x3EF00 may need to be increased if the MSS configuration is complex (since it is variable-sized) and extends beyond 0x3EF00.

To do so, the Data Storage Client base address would need to be increased such that there is no overlap between MPM Configuration Data and MSS Configuration Data. If any of the above parameters are adjusted, a corresponding change needs to be made to the firmware file mpm.c, in the following lines:

```
/* Base address of MPM configuration data */
#ifndef MPM_CONFIGURATION_DATA_BASE_ADDRESS
#define MPM_CONFIGURATION_DATA_BASE_ADDRESS (0x6003F000)
#endif
```

You may calculate how much space the MSS configuration is taking up in NVM by looking at ‘Available Pages’ statistic in Figure 2-2 on page 22. In Figure 2-4 on page 23, there are 2040 pages made up of 128 bytes each, available for general use. Since there are 2048 pages of data available for an A2F200 device (1024 pages for A2F060 and 4092 for A2F500), this means that 8 pages of data are used for MSS configuration.

In practice, you do not need to recalculate this difference whenever a change is made, since the MSS NVM configuration GUI will notify the user if there is an overlap in NVM regions.

MPM Firmware

This is the region in NVM that stores and runs the MPM firmware, from which the ARM Cortex-M3 device executes in place (running from NVM). Its Data Storage Client is shown below.

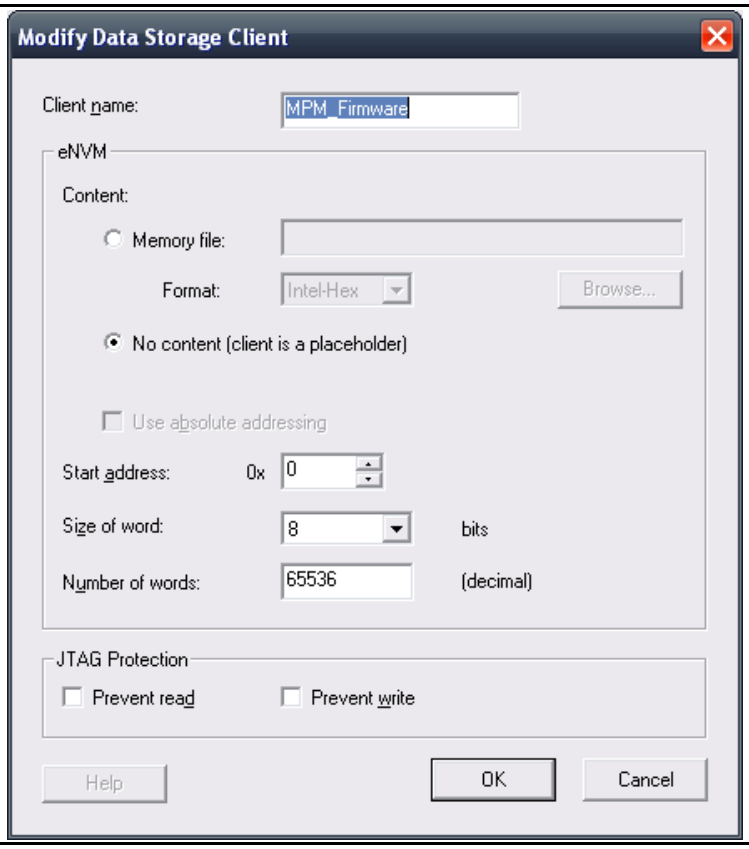

***Figure 2-5* • Data Store Client for MPM Firmware**

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

```
<SoftConsole-Workspace>\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 Log File

This region in NVM is reserved (via an NVM placeholder) for MPM data logging, if event logging is enabled via the MPM configuration registers, to ensure that MSS Configuration / boot code does not place data in this region of NVM. The Data Storage Client configuration window for the MPM Event Log is shown below.

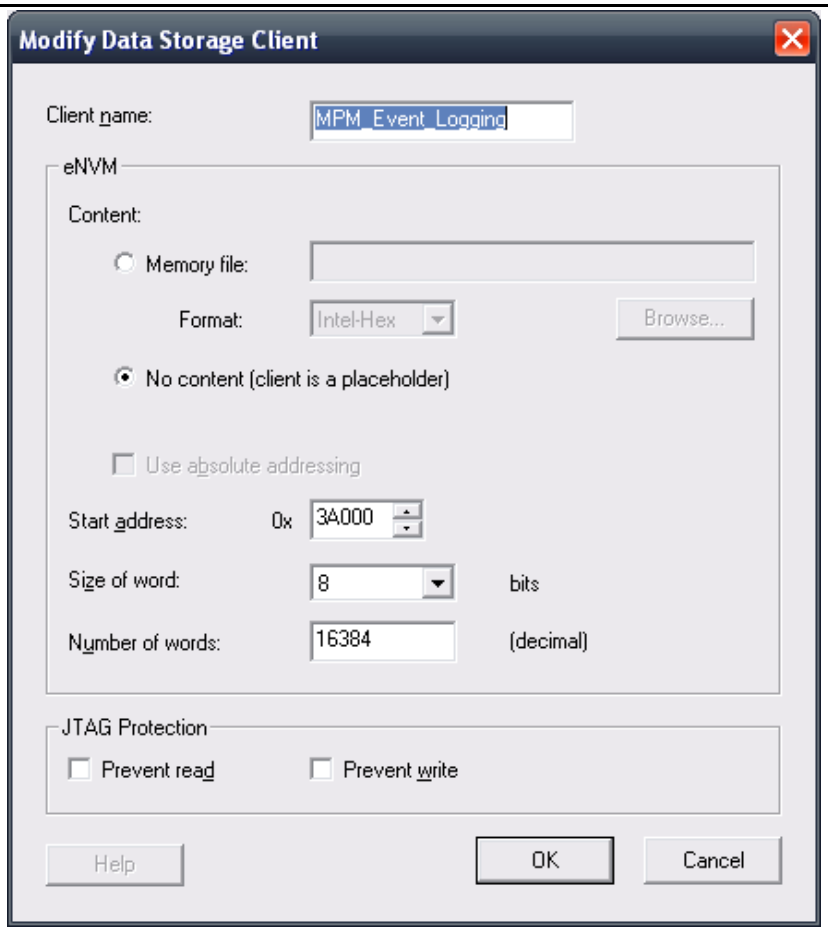

***Figure 2-6* • MPM Event Logging Data Storage Client**

Note that, as with MPM configuration data, the base address and size of this region of NVM must correspond between the MSS configuration (above) and the firmware configuration as defined in mpm.h.

Synthesis and Layout

Once the MSS has been configured and generated, and (optional) any custom logic has been added to the SmartFusion fabric in SmartDesign, you should go through the normal Synthesis and Place and Route flow. For more details on this, please refer to the Actel Libero IDE User's Guide or online help.

The result of place and route should be a top-level FDB file, which is the fabric portion of the FPGA programming file.

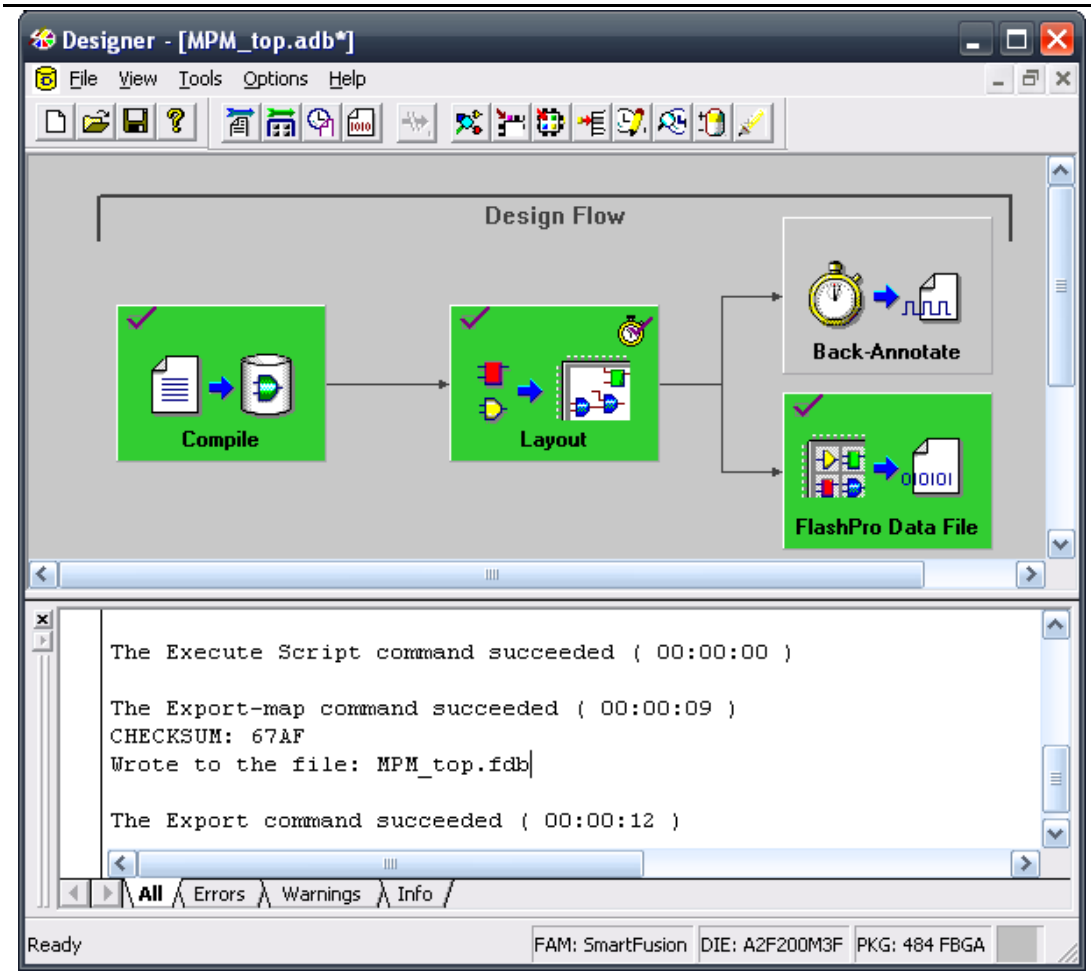

***Figure 2-7* • FDB File Generation**

Programming File Generation and GUI Integration

Once all FDB files are generated, the next step is to create a PDB file, which combines fabric programming data (FDB files) with NVM programming data (EFC files). To do this, you should invoke FlashPro from the Libero Project Flow view, select **Configure Device**, and then **Modify** (programming file) to invoke FlashPoint. A configuration GUI, as shown below, is displayed.

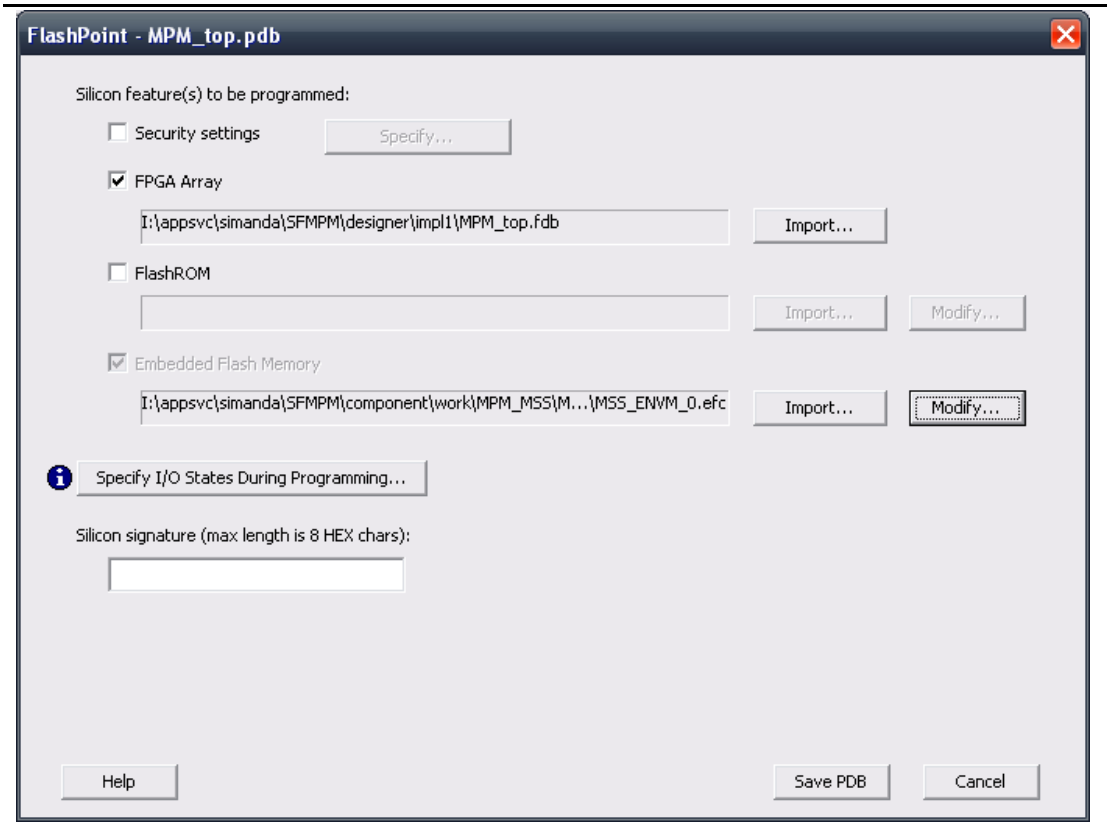

***Figure 2-8* • FlashPoint Programming File Modification**

There may be warnings present if the current PDB file is not up to date with generated FDB or EFC files, in which case you should import the newer, modified PDB/EFC source, and select **Save PDB** at the bottom of the dialog.

The SF MPM GUI requires a STAPL file (not PDB) so at this point you must generate a STAPL file by selecting **File > Export > Export Single Programming File**, at which point the following GUI is displayed.

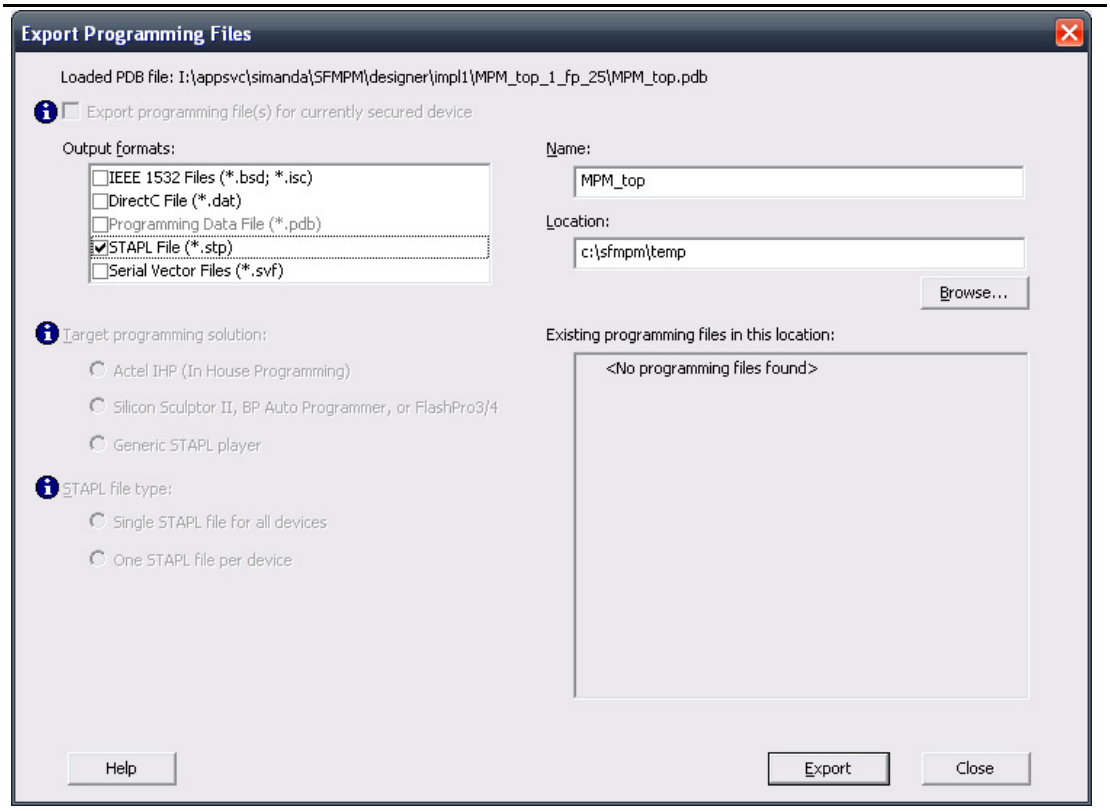

***Figure 2-9* • Programming File Export GUI**

You should select STAPL file (*.stp), as shown above, and select a custom name and location for the generated STAPL file, then select **Export**.

The next step is to rename the generated STAPL file to Template.stp and to copy it to the SF MPM Reference Design GUI template folder (<SFMPM>\template\Template.stp) to replace the original STAPL file used by the GUI when writing MPM configuration data (and all other NVM contents including firmware) to the target.

Note: The original Template.stp file should be backed up in case it is ever needed. Alternatively, the user can select a unique name, and point to the new STAPL file via the GUI “Choose STAPL Temple” menu option.

At this point, the SF MPM GUI is ready for programming/writing fabric with the user-updated design in the future.

3 – MPM for SmartFusion Reference Design I²C

I²C Operation

Overview

SmartFusion MPM Reference Design uses a standard 2-wire I²C slave interface, which is implemented using the SmartFusion MSS I2C hard macro - specifically, MSS_I2C_1 for compatibility with the SmartFusion Advanced Development Kit (A2F-DEV-KIT). The host can perform block reads and writes of up to 64 bytes at a time. The SmartFusion MPM Reference Design I2C protocol conforms to the Philips Inter-Integrated Circuit (I²C) v2.1 specification (7-bit addressing format at effective 100 kbps and 400 kbps data rates). For more information on I²C, refer to the Philips/NXP I2C specification document.

Note: The following I²C registers are configured via the GUI or I²C host writes:

- Slave Address (7-bit)
- I²C Clock Divider

For more information on how this pertains to the MSS I²C or CoreI2C, refer to the appropriate Datasheet/Handbook. Refer to "I2C Register Map" section on page 32 for specific configuration register information.

Board

I²C access to MPM is provided via the I2C_1_SCL and I2C_1_SDA pins on the SmartFusion Advanced Development Kit. These pins correspond to pins 6 and 10 of J7. I²C pull-up resistors (10K) are provided on-board. I²C has an operating voltage of 3.3 V on the SmartFusion Advanced Development Kit. For more detailed board specifications, consult the A2F-DEV-KIT board schematics.

I²C Protocol

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

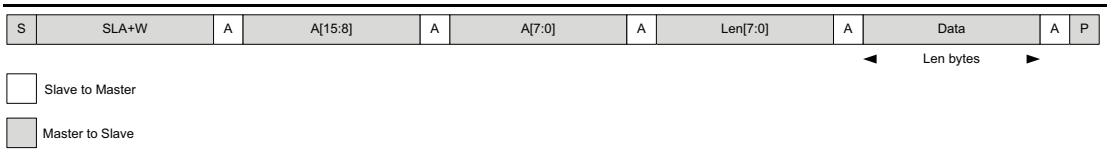

***Figure 3-1* • I²C Block Write**

Note:

- '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 (see section "I2C Register Map" on page 32 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, valid range is 1-64.

Block Read

A host can read up to 64 bytes at a time from SmartFusion MPM registers by issuing a block read, as shown below.

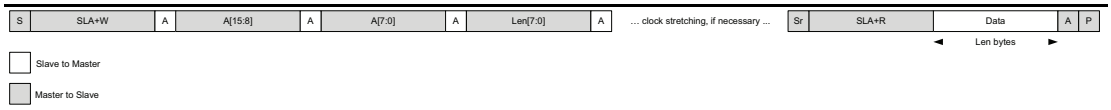

Figure 3-2 • I2C Block Read

Note:

- '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 (see section "I2C Register Map" on page 32 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, valid range is 1-64.
- Clock stretching 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 "I2C Register Map" section on page 32 for specific addresses) are used to issue real-time commands to MPM, as seen below.

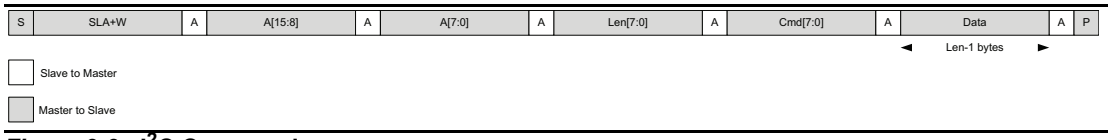

Figure 3-3 • I2C Command

Note:

- Cmd[7:0] is a command op-code.
- Len[7:0] is the number of data bytes that follow the command op-code, valid range is 0-63.
- 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" on page 32 reflects the result of the most recent I2C access.

Types of Access

All I2C access is read and/or write access to the real or pseudo registers defined in the MPM register map. Three types of access are possible:

Three specific types of I2C access are supported:

- Read/write access to configuration registers - write access to configuration registers is only allowed when MPM is in the stopped state and each individual write operation is immediately committed to eNVM. Attempts to write configuration registers while MPM is not stopped are ignored.
- Read only access to monitoring pseudo registers - monitoring pseudo registers can be read any time no matter what state MPM is in. 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 MPM is not in the appropriate state.

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 MPM register address 0x0D00 reads the 16 bit signed mV values for all 32 possible MPM rails/channels and returns them in order of MPM rail/channel number.

Individual 8 or 16 bit register reads or writes are also possible. Bear in mind that each individual configuration register write will be committed immediately to eNVM.

Commands

Commands are implemented as writes of an 8 bit opcode and an optional sequence of commands data bytes (up to 64) to an MPM I^2C command pseudo register. The reference design defines 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 Actel for future use. Additional command registers and command opcodes can be implemented by end users by adapting and extending the reference design firmware.

The following table describes the opcodes defined for command register 0x8000.

***Table 3-1* • Command Register Opcodes**

Command opcode	Name	Data	Description
0x00	Unused/reserved	N/A	N/A
0x01	START	None	Initiate power on sequencing. Maps to the MPM driver API mpm_start(). Ignored if MPM is not in stopped state.
0x02	STOP	None	Initiate power off sequencing. Maps to the MPM driver API mpm_stop(). Ignored if MPM is not in stopped state.
0x03	MARGIN_LOW	One eight bit byte containing the channel to be trimmed/margined.	Set margin level to MARGIN_LOW. 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. Also contains channel number to be set to MARGIN_LOW. Ignored if MPM is not in started state.
0x04	MARGIN_NOM	One eight bit byte containing the channel to be trimmed/margined.	Set margin level to NOMINAL (default) 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. Also contains channel number to be set to NOMINAL. Ignored if MPM is not in started state.

Table 3-1 • **Command Register Opcodes**

Command opcode	Name	Data	Description
0x05	MARGIN_HIGH	One eight bit byte containing the channel to be trimmed/margined.	Set margin level to MARGIN_HIGH. 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. Also contains channel number to be set to MARGIN_HIGH. Ignored if MPM is not in started state.
0x06	INIT	None	Re-initialize MPM by reloading all configuration date and reinitializing the MPM engine and internal data structures. Ignored unless MPM is in stopped state. Normally used to reinitialize MPM via I2C to take account of previously applied MPM configuration register changes.
0x06-0xFF	Unused/reserved	N/A	N/A

I2C Register Map

The I2C register map used in SmartFusion MPM Reference Design v3.0 is backwards compatible with v2.0, with certain exceptions: some registers are deprecated and not implemented, and there is an addition of logging-specific registers. Deprecated registers are italicized in the tables below.

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 via MPM_CONFIGURATION_DATA_BASE_ADDRESS manifest constant in mpm.h that specifies the Cortex-M1 eNVM base address of the MPM register map. By default this is arbitrarily set to 0x6003F000 which is offset 0x0003F000 from the start of eNVM at Cortex-M3 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's MSS configuration where the ‘MPM_Register_Map’ eNVM data storage client's eNVM offset address is specified. By default this is arbitrarily set to 0x0003F000 which is implicitly with respect to the eNVM base address of 0x60000000 in the Cortex-M3 memory map and which obviously matches the firmware project's default setting above. 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 by you through GUI and writing a new STAPL file before invoking FlashPro to write this STAPL file to the target.

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 the general case the MPM register map must be capable of being placed anywhere in the otherwise unused eNVM address space. The amount of eNVM space available differs by die - 128KB on A2F060 (subject to confirmation), 256KB on A2F200, and 512KB on A2F500. Obviously the MPM register map must not clash with other demands on eNVM space (for example, firmware usually at 0x60000000, MSS configuration data etc).

Register Addresses

MPM registers are identified using 16 bit addresses allowing for up to 64Kbytes of register space only a fraction of which is used for MPM operation. Configuration registers are 'real' registers insofar as they actually occupy SmartFusion eNVM memory in the Cortex-M3 address space. Other registers may be 'pseudo' registers insofar as they have specific register addresses but do not occupy any SmartFusion eNVM (or other) memory in the Cortex-M3 address space. 'Real' registers have their 16 bit MPM register map addresses translated internally into 32 bit Cortex-M3 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

The table below summarizes the SF MPM register map. Each register is described in further detail in subsequent sections of this document.

***Table 3-2* • Register Map Overview**

Address	Description
Configuration Registers - 'real' registers that use eNVM storage	
Per channel/rail configuration registers - 64 (0x40) bytes for each channel/rail	
0x0000	Rail A1 configuration registers
0x0040	Rail A2 configuration registers
0x0080	Rail A3 configuration registers
0x00C0	Rail A4 configuration registers
0x0100	Rail A5 configuration registers
0x0140	Rail A6 configuration registers
0x0180	Rail A7 configuration registers
0x01C0	Rail A8 configuration registers
0x0200	Rail A9 configuration registers
0x0240	Rail A10 configuration registers
0x0280	Rail A11 configuration registers
0x02C0	Rail A12 configuration registers
0x0300	Rail A13 configuration registers
0x0340	Rail A14 configuration registers
0x0380	Rail A15 configuration registers
0x03C0	Rail A16 configuration registers
0x0400	Rail A17 configuration registers
0x0440	Rail A18 configuration registers
0x0480	Rail A19 configuration registers
0x04C0	Rail A20 configuration registers
0x0500	Rail A21 configuration registers
0x0540	Rail A22 configuration registers
0x0580	Rail A23 configuration registers
0x05C0	Rail A24 configuration registers
0x0600	Rail A25 configuration registers

Table 3-2 • **Register Map Overview**

Address	Description
0x0640	Rail A26 configuration registers
0x0680	Rail A27 configuration registers
0x06C0	Rail A28 configuration registers
0x0700	Rail A29 configuration registers
0x0740	Rail A30 configuration registers
0x0780	Rail A31 configuration registers
0x07C0	Rail A32 configuration registers
Per-trigger/flag output configuration registers – 32 (0x20) bytes for each output	
0x0800	Output 1 configuration registers
0x0820	Output 2 configuration registers
0x0840	Output 3 configuration registers
0x0860	Output 4 configuration registers
0x0880	Output 5 configuration registers
0x08A0	Output 6 configuration registers
0x08C0	Output 7 configuration registers
0x08E0	Output 8 configuration registers
0x0900	Output 9 configuration registers
0x0920	Output 10 configuration registers
0x0940	Output 11 configuration registers
0x0960	Output 12 configuration registers
0x0980	Output 13 configuration registers
0x09A0	Output 14 configuration registers
0x09C0	Output 15 configuration registers
0x09E0	Output 16 configuration registers
0x0A00	Output 17 configuration registers
0x0A20	Output 18 configuration registers
0x0A40	Output 19 configuration registers
0x0A60	Output 20 configuration registers
0x0A80	Output 21 configuration registers
0x0AA0	Output 22 configuration registers
0x0AC0	Output 23 configuration registers
0x0AE0	Output 24 configuration registers
0x0B00	Output 25 configuration registers
0x0B20	Output 26 configuration registers
0x0B40	Output 27 configuration registers
0x0B60	Output 28 configuration registers

***Table 3-2* • Register Map Overview**

Address	Description
0x0B80	Output 29 configuration registers
0x0BA0	Output 30 configuration registers
0x0BC0	Output 31 configuration registers
0x0BE0	Output 32 configuration registers
Miscellaneous configuration registers	
0x0C00	Miscellaneous configuration registers
I²C monitoring and command registers - 'pseudo' registers that do not use eNVM storage.	
Rail voltage monitoring registers	
0x0D00	Rail voltage monitoring registers
I²C command registers	
0x8000	I²C command registers

Per-rail Configuration Data

The per-channel/rail configuration data 'sub-block' base address for channel n (1 <= n <= 32) is (0x0000 + ((n - 1) x 0x40)). The following table details the per-channel/rail registers and their offsets with respect to this per-channel/rail configuration data 'sub-block' base address.

***Table 3-3* • Per-rail Configuration Data**

Name	Offset	Access	Description	
VHL	0x0000	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 nature of underlying ACE channel (For example, direct analog input 0 to +2560mV or ABPS with a particular prescalar range).
VHH	0x0001	R/W	Threshold hysteresis VH[15:8]	
VOV2L	0x0002	R/W	OV2 threshold VOV2[7:0]	
VOV2H	0x0003	R/W	OV2 threshold VOV2[15:8]	
VOV1L	0x0004	R/W	OV1 threshold VOV1[7:0]	
VOV1H	0x0005	R/W	OV1 threshold VOV1[15:8]	
VUV1L	0x0006	R/W	UV1 threshold VUV1[7:0]	
VUV1H	0x0007	R/W	UV1 threshold VUV1[15:8]	
VUV2L	0x0008	R/W	UV2 threshold VUV2[7:0]	
VUV2H	0x0009	R/W	UV2 threshold VUV2[15:8]	
VOFFL	0x000A	R/W	OFF threshold OFF[7:0]	
VOFFH	0x000B	R/W	OFF threshold OFF[15:8]	
SSLO	0x000C	R/W	Power sequencing slot. Bits [7:6] reserved Bits [5:0] power sequencing slot number to use. Valid values are: • 0: No slot/not sequenced • 1-32: Slot in which this channel is sequenced	-
Unused	0x000D	N/A	Unused/reserved	-

***Table 3-3* • Per-rail Configuration Data**

Name	Offset	Access	Description	
SDONL	0x000E	R/W	SDON[7:0]	SDON[15:0] ñ per channel power on sequencing delay with respect to start of relevant sequencing slot. 16 bit unsigned time value in milliseconds.
SDONH	0x000F	R/W	SDON[15:8]	
SDOFFL	0x0010	R/W	SDOFF[7:0]	SDOFF[15:0] ñ per channel power off sequencing delay with respect to start of relevant sequencing slot. 16 bit unsigned time value in milliseconds.
SDOFFH	0x0011	R/W	SDOFF[15:8]	
TRCFG	0x0012	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 = SmartFusion MSS OBD • 1 = CorePWM	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=Closed Loop in order to match the demo/reference design setup.
Unused	0x0013	N/A	Unused/reserved	-
VNOML	0x0014	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	0x0015	R/W	VNOM[15:8]	
TRSDELL	*0x0016*	*R/W*	*TRDEL[7:0]*	*(Closed loop) trimming startup delay with respect to completion of power on sequencing. Trimming starts after this delay has elapsed. 16 bit unsigned value in ms. Deprecated in v3.0, not configurable via MPM GUI.*
TRSDELH	*0x0017*	*R/W*	*TRDEL[15:8]*	

***Table 3-3* • Per-rail Configuration Data**

Name	Offset	Access	Description	
DACOUT_NOML	0x0018	R/W	DACOUT_NOM[7:0]	For open loop trimming DACOUT_NOM is a 16 bit value between 0 and 3300mV specifying the voltage required as input to the regulator's trim pin circuit in order 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	0x0019	R/W	DACOUT_NOM[15:8]	
DACOUT_HIL	0x001A	R/W	DACOUT_HI[7:0]	Similar to DACOUT_NOM except that this is the regulator trim pin input voltage required to achieve the required trimmed high output voltage on this channel.
DACOUT_HIH	0x001B	R/W	DACOUT_HI[15:8]	
DACOUT_LOL	0x001C	R/W	DACOUT_LO[7:0]	Similar to DACOUT_NOM except that this is the regulator trim pin input voltage required to achieve the required trimmed low output voltage on this channel.
DACOUT_LOH	0x001D	R/W	DACOUT_LO[15:8]	
TRIHIH	0x001E	R/W	TRIHI[7:0]	Closed loop trimming high output voltage target. Normally within 10% of VNOM. 16 bit signed mV value.
TRIHIL	0x001F	R/W	TRIHI[15:8]	
TRILOH	0x0020	R/W	TRILO[7:0]	Closed loop trimming low output voltage target. Normally within 10% of VNOM. 16 bit signed mV value.
TRILOL	0x0021	R/W	TRILO[15:8]	
Unused	0x0022 - 0x003F	N/A	Unused/reserved	-

Per-output Configuration Data

The per-trigger/flag output configuration data 'sub-block' base address for output n (1 <= n <= 32) is (0x0800 + ((n - 1) x 0x20)). The following table 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 3-4* • Per-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 3-4* • Per-output Configuration Data (continued)**

Name	Offset	Access	Description
FLAG_CFG	0x0010	R/W	Other flag configuration Bit[7:6] Output logging during power sequencing: • 0 = None • 1 = During power on sequencing • 2 = During power down off sequencing • 3 = During power on and power off sequencing Bit[5] During power sequencing: • 0 = Hold/don't 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
Unused	0x0012-0x001F	N/A	Unused/reserved

Miscellaneous/Global Configuration Data

The miscellaneous configuration data 'sub-block' base address is 0x0C00.

***Table 3-5* • Miscellaneous/Global Configuration Data**

Name	Address	Access	Description	
PWR_ON_CTRL	0x0C00	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	0x0C01	N/A	Unused/reserved	-
SLOT_TIMEOUT_L	0x0C02	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	0x0C03	R/W	SLOT_TIMEOUT[15:8]	
PWR_OFF_CTRL	0x0C04	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	0x0C05	R/W	Bits[6:0] I2C slave address	-
LOGGING	0x0C06	R/W	Bit[0] MPM engine state changes • 0 = don't log • 1 = log Bit[1] Power on sequencing events • 0 = don't log • 1 = log Bit[2] Power off sequencing events • 0 = don't log • 1 = log	Optional logging of MPM engine state transitions and power sequencing events.

***Table 3-5* • Miscellaneous/Global Configuration Data**

Name	Address	Access	Description	
CLEAR_LOG	0x0C07	R/W	Bit[0] • 0 = don't clear log • 1 = clear log	Clear log next time MPM (re)initializes
Unused	0x0C08-0x0CFF	N/A	Unused/reserved	-

Rail Voltage Monitoring Registers

The rail monitoring 'pseudo' register 'sub-block' base address is 0x0D00.

***Table 3-6* • Rail Voltage Monitoring Registers**

Name	Address	Access	Description	
RAIL1_VH	0x0D00	R/O	RAIL1_V[15:8]	16 bit signed value representing the instantaneous output voltage for rail A1 in mV.
RAIL1_VL	0x0D01	R/O	RAIL1_V[7:0]	
RAIL2_VH	0x0D02	R/O	RAIL2_V[15:8]	16 bit signed value representing the instantaneous output voltage for rail A2 in mV.
RAIL2_VL	0x0D03	R/O	RAIL2_V[7:0]	
RAIL3_VH	0x0D04	R/O	RAIL3_V[15:8]	16 bit signed value representing the instantaneous output voltage for rail A3 in mV.
RAIL3_VL	0x0D05	R/O	RAIL3_V[7:0]	
RAIL4_VH	0x0D06	R/O	RAIL4_V[15:8]	16 bit signed value representing the instantaneous output voltage for rail A4 in mV.
RAIL4_VL	0x0D07	R/O	RAIL4_V[7:0]	
RAIL5_VH	0x0D08	R/O	RAIL5_V[15:8]	16 bit signed value representing the instantaneous output voltage for rail A5 in mV.
RAIL5_VL	0x0D09	R/O	RAIL5_V[7:0]	
RAIL6_VH	0x0D0A	R/O	RAIL6_V[15:8]	16 bit signed value representing the instantaneous output voltage for rail A6 in mV.
RAIL6_VL	0x0D0B	R/O	RAIL6_V[7:0]	
RAIL7_VH	0x0D0C	R/O	RAIL7_V[15:8]	16 bit signed value representing the instantaneous output voltage for rail A7 in mV.
RAIL7_VL	0x0D0D	R/O	RAIL7_V[7:0]	
RAIL8_VH	0x0D0E	R/O	RAIL8_V[15:8]	16 bit signed value representing the instantaneous output voltage for rail A8 in mV.
RAIL8_VL	0x0D0F	R/O	RAIL8_V[7:0]	
RAIL9_VH	0x0D10	R/O	RAIL9_V[15:8]	16 bit signed value representing the instantaneous output voltage for rail A9 in mV.
RAIL9_VL	0x0D11	R/O	RAIL9_V[7:0]	
RAIL10_VH	0x0D12	R/O	RAIL10_V[15:8]	16 bit signed value representing the instantaneous output voltage for rail A10 in mV.
RAIL10_VL	0x0D13	R/O	RAIL10_V[7:0]	
RAIL11_VH	0x0D12	R/O	RAIL11_V[15:8]	16 bit signed value representing the instantaneous output voltage for rail A11 in mV.
RAIL11_VL	0x0D15	R/O	RAIL11_V[7:0]	
RAIL12_VH	0x0D16	R/O	RAIL12_V[15:8]	16 bit signed value representing the instantaneous output voltage for rail A12 in mV.
RAIL12_VL	0x0D17	R/O	RAIL12_V[7:0]	

***Table 3-6* • Rail Voltage Monitoring Registers**

Name	Address	Access	Description	
RAIL13_VH	0x0D18	R/O	RAIL13_V[15:8]	16 bit signed value representing the instantaneous output voltage for rail A13 in mV.
RAIL13_VH	0x0D19	R/O	RAIL13_V[7:0]	
RAIL14_VH	0x0D1A	R/O	RAIL14_V[15:8]	16 bit signed value representing the instantaneous output voltage for rail A14 in mV.
RAIL14_VL	0x0D1B	R/O	RAIL14_V[7:0]	
RAIL15_VH	0x0D1C	R/O	RAIL15_V[15:8]	16 bit signed value representing the instantaneous output voltage for rail A15 in mV.
RAIL15_VL	0x0D1D	R/O	RAIL15_V[7:0]	
RAIL16_VH	0x0D1E	R/O	RAIL16_V[15:8]	16 bit signed value representing the instantaneous output voltage for rail A16 in mV.
RAIL16_VL	0x0D1F	R/O	RAIL16_V[7:0]	
RAIL17_VH	0x0D20	R/O	RAIL17_V[15:8]	16 bit signed value representing the instantaneous output voltage for rail A17 in mV.
RAIL17_VL	0x0D21	R/O	RAIL17_V[7:0]	
RAIL18_VH	0x0D22	R/O	RAIL18_V[15:8]	16 bit signed value representing the instantaneous output voltage for rail A18 in mV.
RAIL18_VL	0x0D23	R/O	RAIL18_V[7:0]	
RAIL19_VH	0x0D24	R/O	RAIL19_V[15:8]	16 bit signed value representing the instantaneous output voltage for rail A19 in mV.
RAIL19_VL	0x0D25	R/O	RAIL19_V[7:0]	
RAIL20_VH	0x0D26	R/O	RAIL20_V[15:8]	16 bit signed value representing the instantaneous output voltage for rail A20 in mV.
RAIL20_VL	0x0D27	R/O	RAIL20_V[7:0]	
RAIL21_VH	0x0D28	R/O	RAIL21_V[15:8]	16 bit signed value representing the instantaneous output voltage for rail A21 in mV.
RAIL21_VL	0x0D29	R/O	RAIL21_V[7:0]	
RAIL22_VH	0x0D2A	R/O	RAIL22_V[15:8]	16 bit signed value representing the instantaneous output voltage for rail A22 in mV.
RAIL22_VL	0x0D2B	R/O	RAIL22_V[7:0]	
RAIL23_VH	0x0D2C	R/O	RAIL23_V[15:8]	16 bit signed value representing the instantaneous output voltage for rail A23 in mV.
RAIL23_VL	0x0D2D	R/O	RAIL23_V[7:0]	
RAIL24_VH	0x0D2E	R/O	RAIL24_V[15:8]	16 bit signed value representing the instantaneous output voltage for rail A24 in mV.
RAIL24_VL	0x0D2F	R/O	RAIL24_V[7:0]	
RAIL25_VH	0x0D30	R/O	RAIL25_V[15:8]	16 bit signed value representing the instantaneous output voltage for rail A25 in mV.
RAIL25_VL	0x0D31	R/O	RAIL25_V[7:0]	
RAIL26_VH	0x0D32	R/O	RAIL26_V[15:8]	16 bit signed value representing the instantaneous output voltage for rail A26 in mV.
RAIL26_VL	0x0D33	R/O	RAIL26_V[7:0]	
RAIL27_VH	0x0D32	R/O	RAIL27_V[15:8]	16 bit signed value representing the instantaneous output voltage for rail A27 in mV.
RAIL27_VL	0x0D35	R/O	RAIL27_V[7:0]	

***Table 3-6* • Rail Voltage Monitoring Registers**

Name	Address	Access	Description	
RAIL28_VH	0x0D36	R/O	RAIL28_V[15:8]	16 bit signed value representing the instantaneous output voltage for rail A28 in mV.
RAIL28_VL	0x0D37	R/O	RAIL28_V[7:0]	
RAIL29_VH	0x0D38	R/O	RAIL29_V[15:8]	16 bit signed value representing the instantaneous output voltage for rail A29 in mV.
RAIL29_VH	0x0D39	R/O	RAIL29_V[7:0]	
RAIL30_VH	0x0D3A	R/O	RAIL30_V[15:8]	16 bit signed value representing the instantaneous output voltage for rail A30 in mV.
RAIL30_VL	0x0D3B	R/O	RAIL30_V[7:0]	
RAIL31_VH	0x0D3C	R/O	RAIL31_V[15:8]	16 bit signed value representing the instantaneous output voltage for rail A31 in mV.
RAIL31_VL	0x0D3D	R/O	RAIL31_V[7:0]	
RAIL32_VH	0x0D3E	R/O	RAIL32_V[15:8]	16 bit signed value representing the instantaneous output voltage for rail A32 in mV.
RAIL32_VL	0x0D3F	R/O	RAIL32_V[7:0]	
I2C_STATUS	0x0D40	R/C	Status of 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 amount of data that followed) • 5 = NVM write failure (when writing config registers) • 6 = Invalid write (attempt to write to read only register)	Mainly for debugging I²C interactions. Indicates the status of the most recent I²C protocol interaction.
LOG_STATUS	0x0D41	R/O	Bit[0] – log full • 0 = not full • 1 = full	Read this to check if the log is full. If it is then the CLEAR_LOG configuration register can be written to reset/clear the log next time MPM initializes (either hard reset or soft I2C_INIT command)

Table 3-6 • **Rail Voltage Monitoring Registers**

Name	Address	Access	Description	
RTC4	0x0D42	R/O	RTC byte 4 – i.e. bits[39:32]	40 bit RTC (Real Time Counter) value from the SmartFusion 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 via I2C in one operation since reading individual bytes separately and recombining them will most likely yield an incorrect time value.
RTC3	0x0D43	R/O	RTC byte 4 – i.e. bits[31:24]	-
RTC2	0x0D44	R/O	RTC byte 4 – i.e. bits[23:16]	-
RTC1	0x0D45	R/O	RTC byte 4 – i.e. bits[15:8]	-
RTC0	0x0D46	R/O	RTC byte 4 – i.e. bits[7:0]	-
Unused	0x0D46-0x0FFF	N/A	Unused/reserved	-
Log file	0x1000-0x7FFF	R/O	Log file	28KB of address space allowing access to the log file/buffer. The actual size of the log file depends on the MPM implementation. Log file records start at 7 byte aligned addresses. A log record whose first byte (record type) is 0x00 indicates the end of the log. Thus a sequential scan of the log involves reading from 0x1000 until a record has a record type of 0x00 or the address 0x7FFF is reached.

I2C Command Registers

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

Table 3-7 • **I2C Command Registers**

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

Notes

It should be noted that, because there have been significant changes to the SF MPM Register Map since the release of SmartFusion MPM Reference Design v1.0, the configuration GUI from v2.0 of the design is not backwards compatible, nor is the v2.0 GUI compatible with v1.0 of the design.

Version 3.0 is, however, backward compatible with version 2.0 of SF MPM, with some deprecated/unused registers as noted above.

4 – SmartFusion MPM Data Logging

Data Logging

Events

The following events can be logged (configurable via the MPM GUI or I^2C configuration registers).

MPM Engine State Changes

SF MPM consists of four states - stopped, starting, started, and stopping. Transitions between states are conditional on Rail statuses or can be initiated via MPM API functions (or I^2C commands). These transitions can be logged. Figure 4-1 illustrates a state diagram of SF MPM.

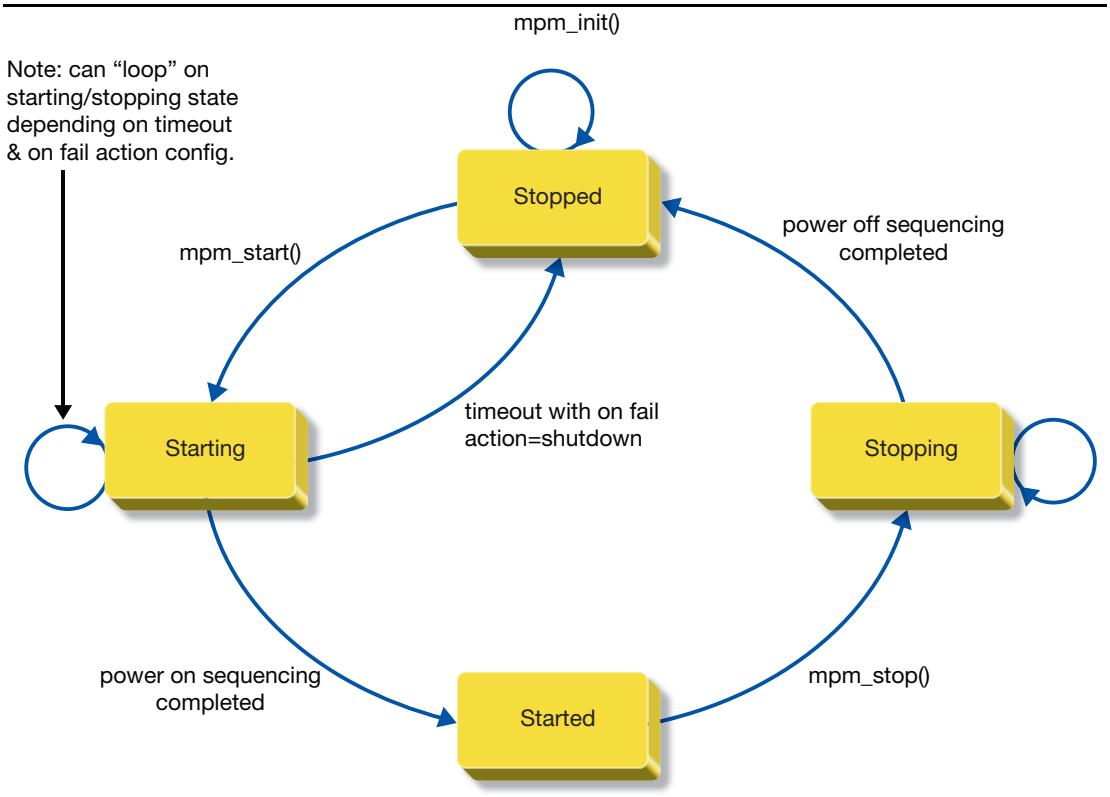

***Figure 4-1* • MPM State Transition Diagram**

Output Trigger/Flag State Change

Users 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.

The figure below shows the GUI configuration for output/trigger logging.

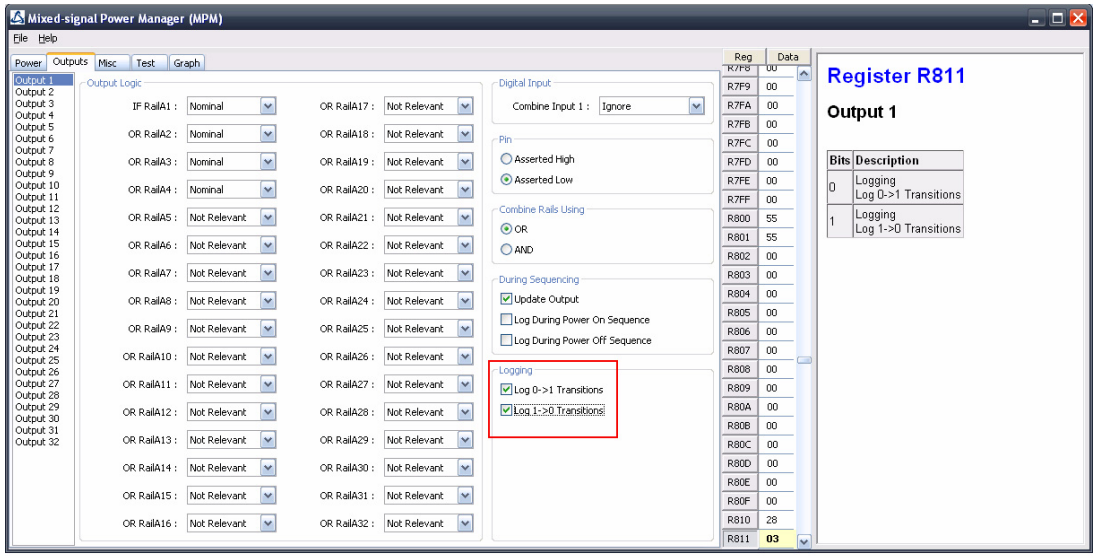

***Figure 4-2* • GUI Configuration of Output/Trigger Data Logging**

Power Sequencing Events

The user may log power-on sequencing events or power-off sequencing events. The figure below shows the GUI (global) configuration for Power sequencing event logging.

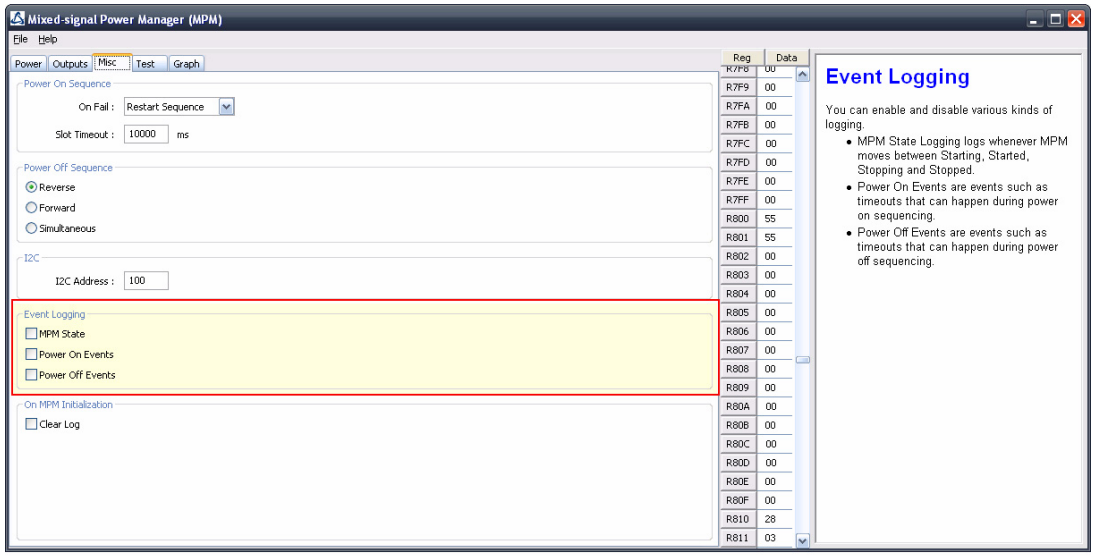

***Figure 4-3* • Power Sequencing Event Logging**

Log Record Format

Each log item consists of a record of 7 bytes as follows (LSB first):

- Byte 0: log record type
- Byte 1: log data
- Byte 2-6: log timestamp

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

Log Record Type Byte

The table below describes record types used for SF MPM data logging.

Table 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/trigger transition occured
0x03	Power on sequencing event
0x04	Power off sequencing event
0x05 - 0xFF	Reserved

Log Data

Log data meaning is dependent on the corresponding log record type byte. The table below describes the meaning of the log data byte given the log record type byte.

Table 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)	
4:0	Output number N, minus 1 (valid values are 0-31, while output numbers range from 1-32)
5	New value of output, high or low (0 or 1)
7:6	Unused
Log record byte = 0x03/0x04 (Power on/off sequencing event)	
2:0	Event Type [0 = start; 1 = finish; 2 = timeout; 3 = hold; 4 = shutdown/terminate; 5 = restart slot; 6 = restart sequence; 7 = skip slot]
7:3	Slot number: For “restart slot events” (configuration 5) and “skip slot events” (configuration 7), this indicates the slot number, minus 1 (valid values are 0-31, while slot numbers range from 1-32)

Log Timestamp

The SmartFusion MSS Real Time Counter (RTC) 40 bit (5 byte) counter value is used to timestamp log records. By default the RTC is clocked by a 32.768KHz clock with a prescaling divider of 128[1] so that the

resolution of the RTC timer is 1 second / (32.768KHz / 128) = 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 timestamp measures the time (in 3.90625ms units) since the most recent reset of the MSS RTC. Battery backup of the RTC can be used on SmartFusion to maintain the RTC counter across SmartFusion device resets. If this is not used, then the RTC will reset to 0 each time the device is reset. Note that the RTC does not automatically start counting out of reset and only does so when explicitly programmed to do so – for example, via the MSS RTC driver's MSS_RTC_init() API.

MPM's mpm_init() API initializes and starts the RTC which will start it counting from the last stored value (0 after a reset where no battery backup is used otherwise the last stored count). Note that there is an undocumented delay of perhaps 8-10 seconds between starting the RTC and it actually (re)starting to count.

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 SmartFusion device reset in the absence of battery backup of the RTC.

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.

Log file

The log file is stored in non volatile memory – by default SmartFusion ENVM. The base address and size of the log file is defined in the Libero hardware and SoftConsole firmware projects. In the former a new MSS ENVM placeholder data storage client named "MPM_Event_Log" is defined in order to reserve a portion of the ENVM address space for logging. The ENVM offset address and length is defined here. In SoftConsole the Cortex-M3 ENVM address and length of the log file is also defined. The address and length defined in both projects must match in the same way that these must match for the MPM_Configuration_Data ENVM data storage client. Any change to the base address or length must be made in both places.

The log file is written until full. The configuration register CLEAR_LOG is provided which, when set to 1, causes MPM to clear the log file next time it is initialized (via a driver API or I^2C invocation of mpm_init()). Where the log file is not already full, the first record with a type of 0x00[2] indicates the next free slot in the file. When writing log records 8 bytes will be written (subject to reaching the end of the log) – the 7 bytes of the record itself plus a single 0x00 byte to ensure that the free slot marker always exists.

Read only access to the log file is via SmartFusion MPM I^2C slave addresses 0x1000 to 0x7FFF which means that up to 28KB of logging space can be made available.[3]

The default reference design has a <= 28KB MSS ENVM data storage client defined in the Libero hardware project and corresponding Cortex-M3 ENVM base address and extent values defined in the SoftConsole firmware project to implement the default log file. Logging can be redirected to any suitable memory region (for example, external Flash etc.) through adaptation of the MPM engine's internal method for writing/committing log records.

1. *If the RTC clock source (32.768KHz) and/or divider (128) can be changed then any concomitant changes to the RTC tick time (3.906265ms) and total lifetime (136 years) would need to be accounted for.*
2. *The first 0x00 byte on a 7 byte boundary in the log.*
3. *To make more logging space available the command register currently at 0x8000 could be moved towards the top of the MPM I2C slave address space but this would impact existing deployments and documentation etc. so will not be done in v3.0.*

5 – MPM for SmartFusion Reference Design Trimming

Operation

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.

Open-Loop Trimming

The hardware for open-loop trimming resembles the setup shown in Figure 5-1.

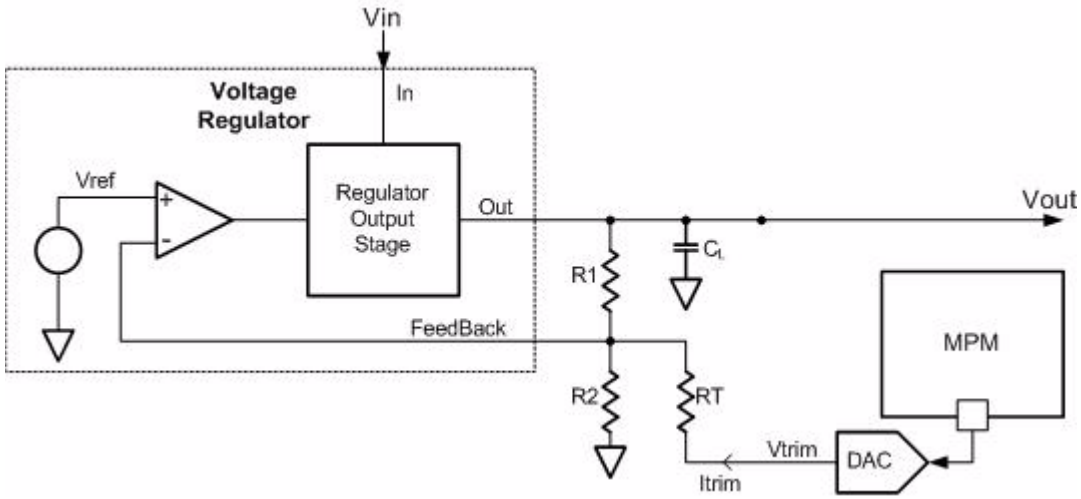

***Figure 5-1* • Open-Loop Trimming Using SmartFusion MPM**

The regulator feedback is controlled by SmartFusion 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. SmartFusion 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 SmartFusion MPM, as seen in Figure 5-2.

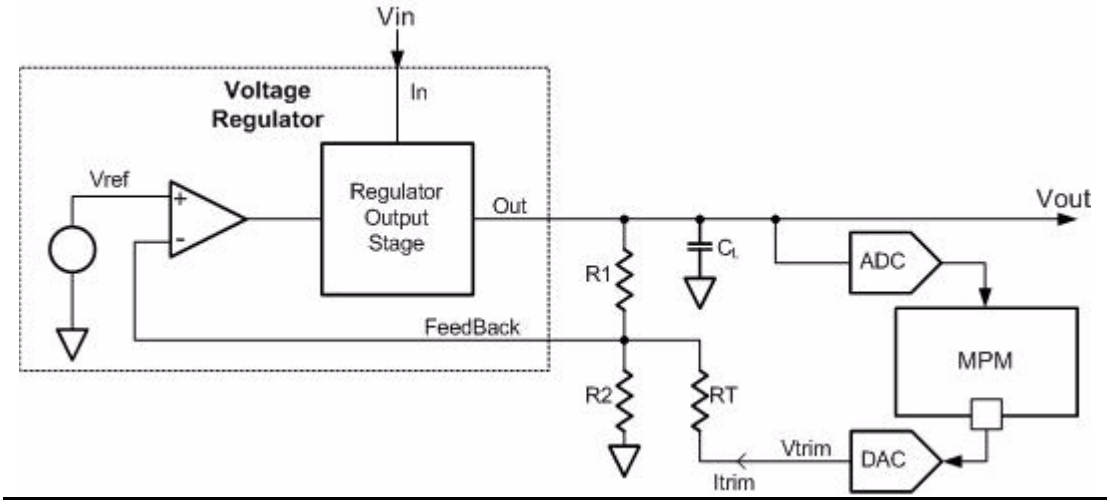

***Figure 5-2* • Closed-Loop Trimming Using SmartFusion MPM**

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:

1. Read V_{out} (rail value)
2. Compare V_{out} to Vouttarget
3. 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 "Type" section on page 51 for information on how $V_{outtarget}$ is determined.

Note: For SmartFusion MPM Reference Design v3.0, you can select (by setting NVM configuration registers via the MPM GUI or I2C) either a DAC implementation of CorePWM combined with an RC network (or other low pass filter) or the hard-macro SmartFusion Sigma Delta DAC, which requires no external circuitry.

SmartFusion 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 Rtrim 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), and not so large as to make Vtrim 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 5-3 below define configuration parameters used for trimming on a per-channel basis.

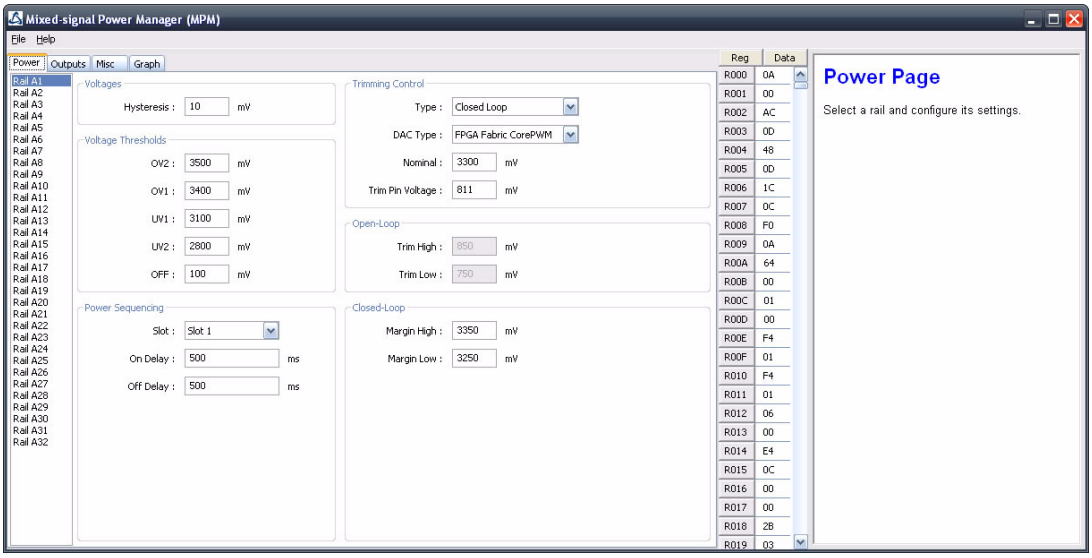

***Figure 5-3* • Trimming GUI Section**

Trimming Control

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

Type

This is the trimming type as described in the "Operation" section on page 49.

None

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 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 50. The target voltage ($V_{outtarget}$) is one of three values:

1. Nominal: By default, or if the MARGIN_NOM command has been received by MPM
2. Margin High: If the MARGIN_HIGH command has been received by MPM
3. Margin Low: If the MARGIN_LOW command has been received my 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 and 0 V to 2.56 V for hard Sigma Delta DAC implementation, is sufficient for the system (feedback resistor, voltage divider, trim pin operating range, etc.) and for the amount of trimming you expect to be performed.

DAC Type

For trimming, this determines the type of DAC used for this rail.

OBD Hard-Macro

The SmartFusion on-chip sigma delta DAC. As mentioned above, the voltage range for the SmartFusion SDD is 0-2.56 V. This still covers typical regulator trim pin ranges, which tend to be around 1 V.

FPGA Fabric

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

6 – MPM Daughter Card Hardware Guide

Introduction

The Mixed Signal Power Manager (MPM) Daughter Card enables system designers to evaluate the functionality of Actel's power management solutions in hardware with a four-regulator benchtop power management development system. The MPM Daughter Card includes four regulators, four voltage bias POTs and four fault introduction push-button switches, implementing four fully independent power supplies that can be varied and interrupted to demonstrate the management capabilities of Actel's mixed signal power management solution, MPM.

MPM takes advantage of the processing power and programmable flexibility of the SmartFusion intelligent mixed signal FPGAs included on the SmartFusion Evaluation Kit or SmartFusion Development Kit when these kits are connected to the MPM Daughter Card and programmed with the MPM reference design. This section explores the MPM Daughter Card board hardware.

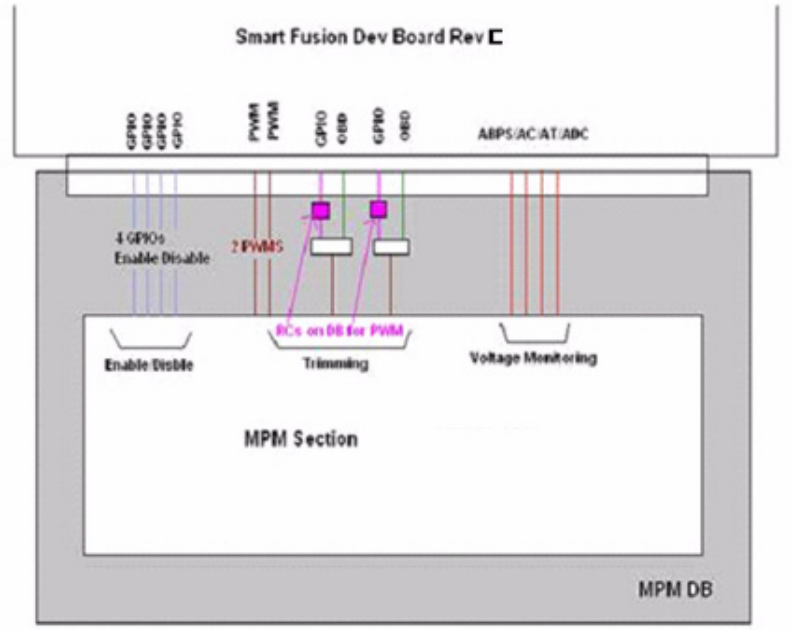

***Figure 6-1* • MPM Daughter Card**

Kit Contents

***Table 1* • MPM Daughter Card Kit Contents**

Quantity	Description
1	MPM Daughter Card
1	9 V power supply
5	Jumpers in small packet
1	Quickstart Card

Related Information

- SmartFusion Development Kit
- SmartFusion Reference Documents
- Libero IDE Design Software

Board Description

The MPM Daughter Card provides a benchtop demonstration and development platform for Actel's MPM reference design. MPM delivers power monitoring, power sequencing, and threshold monitoring of up to 22 power regulators using an easy-to-use standalone GUI tool.

Designers using MPM can now replace discrete power management devices, add more flexibility by leveraging SmartFusion's reprogrammable flash FPGA technology, and reduce total parts count at the board level. MPM delivers highly configurable, integrated power management using one high-reliability, low-power, flash-based SmartFusion mixed signal FPGA.

The MPM daughter board, shown in Figure 6-2, connects to the SmartFusion Evaluation Kit (A2F-EVAL-KIT) or the SmartFusion Development Kit (A2F-DEV-KIT) via the mixed signal header.

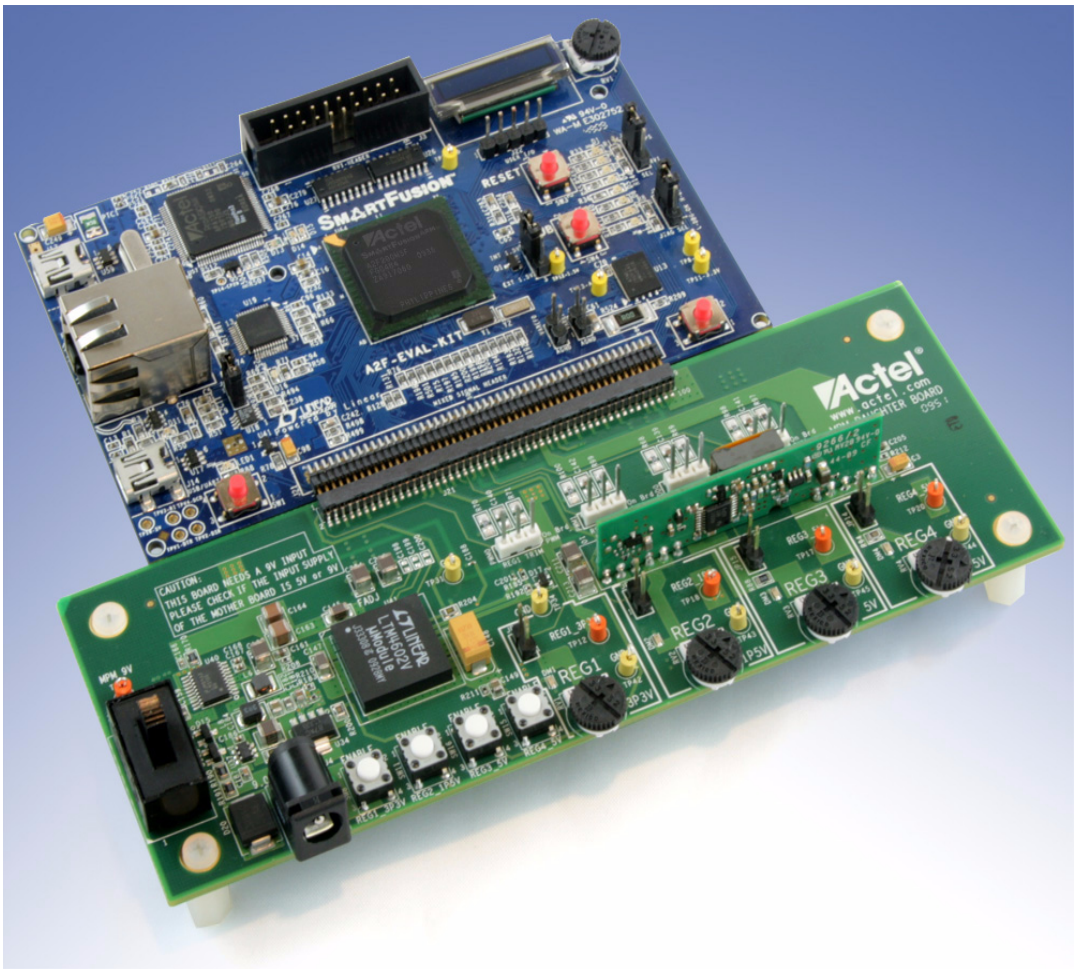

***Figure 6-2* • MPM Daughter Board with SmartFusion Eval Kit**

Mixed Signal Power Manager Board Components

Table 2 • **Mixed Signal Power Manager Board Components**

Serial Number	Name	Description
1	MPM_REG1_5V	5 V regulated supply1, Lineage Power ATA010A0X43
2	MPM_REG2_5V	5 V regulated supply2, Linear Technology
3	MPM_REG1_3P3	3.3 V regulated supply, National LM3100MH
4	MPM_REG2_1P5	1.5 V regulated supply, National LP38693-ADJ
5	RV1, RV2, RV3, RV4	POT to vary the voltage O/P of regulator
6	SW8, SW11, SW15, SW16	Switch to enable/diable MPM voltage regulators

Installation and Settings

Jumper and Switch Settings

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

Table 3 • **Jumper Settings (MPM)**

Jumper	Development Kit Function	Default Setting
JP12	Jumper from pin FB (Feedback Voltage) of 3.3 V regulator for voltage trimming	Closed
JP13	Jumper from pin ADJ (Adjust Voltage) of 1.5 V regulator for voltage trimming	Closed
JP14	Jumper from pin Voset (Voltage) of 5 V DC-DC regulator for voltage trimming	Closed
JP15	Jumper from pin trim (Voltage) of 5 V DC-DC Convertor output for Voltage Trimming	Closed
JP19	Trimming DAC option Reg1	PWM0
JP20	Trimming DAC option Reg2	PWM1
JP17	Trimming DAC option Reg3	SDD0
JP18	Trimming DAC option Reg4	SDD1

Note: For JP17 and JP18, connecting pins 2-3 connect SDD0 and SDD1 to Reg3 and Reg4 trimming circuitry respectively. Since there are only two available SDD's on board for JP19 and JP20, connecting pins 2-3 connect fabric outputs with added RC networks to Reg1 and Reg2 trimming circuitry respectively.

Table 4 • **Table Switches (MPM)**

Switch	Comments
SW17	Switch ON 9 V DC into MPM
SW8	Push-button to disable 3.3 V supply regulator
SW11	Push-button to disable 1.5 V supply regulator
SW16	Push-button to disable 5 V supply regulator (ARTESYN)
SW15	Push-button to disable 5 V supply regulator (Linear Tech)

Note: The push-buttons SW8, SW11, SW15, and SW16 ground the Enable pin of the corresponding regulator, thereby injecting failure in power subsystem. The Enable pin is also connected to the pin on the FPGA (MPM_REG1_EN, MPM_REG2_EN, MPM_REG3_EN, and MPM_REG4_EN). The FPGA could be damaged if it is driving HIGH on the enable line and simultaneously shorted to ground. To avoid this, Actel recommends that the FPGA pin driving these enables should be either driving Low or tristate.

Hardware Components

Mixed Signal Power Management Description

The MPM Daughter Card includes 4 sets of regulated power supply running from a 9 V supply.

- Switching regulator 1.5 V
- Switching regulator 3.3 V
- DC-DC regulators 5 V
- DC-DC regulators 5 V

Using the MPM GUI you can do the following:

- Monitor voltage for all rails
- Sequence different power rails for power-up and power-down
- Trim and margin voltage rails in a closed-loop
- Sweep the output voltage (POT circuit to change resistor on feedback voltage)
- Induce failures by disabling the enable input of the regulator (push-button to GND enable)

MPM manages power sequencing by sequentially asserting or deasserting channel enable pins for power-up and power-down as per the configuration defined in the MPM GUI and written to on-chip nonvolatile memory registers, and monitoring the associated channel voltage. All Enable pins to the regulators are active high.

Board Components

3.3 V Supply, MPM_REG1_3P3, National LM3100MH

This regulated switching mode power supply can deliver up to 1.5 A of output current with regulated output voltage from 0.8 VDC to 5.0 VDC over a wide range of input voltage (VIN = 4.5 – 36 VDC). Different output voltages can be set by changing the value of feedback voltage on feedback pin (FB) using a resistor feedback divider.

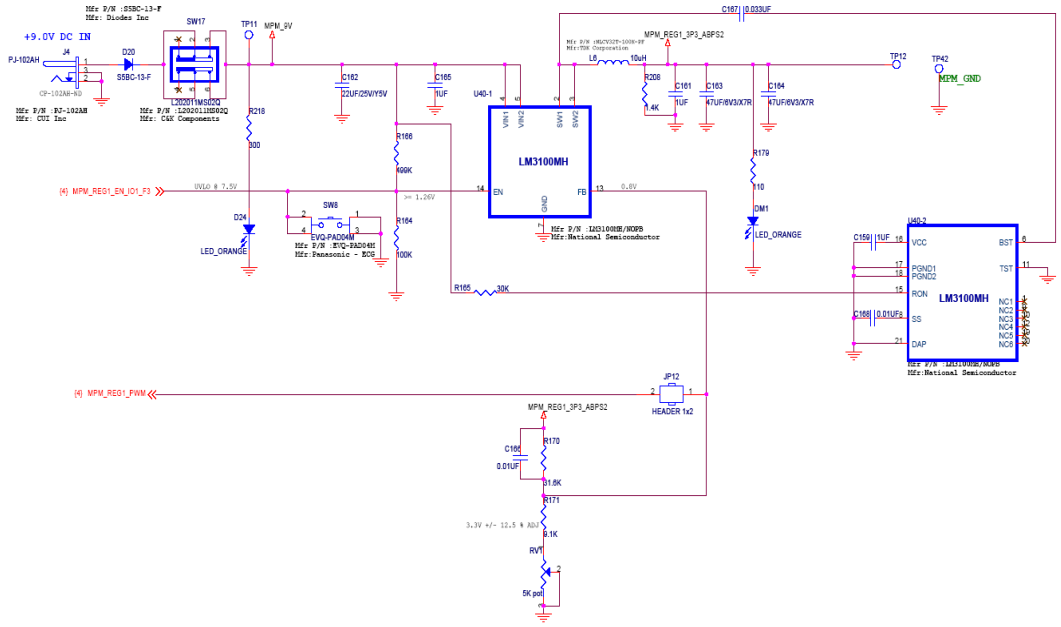

***Figure 6-3* • MPM Switching Power Supply 3.3 V**

1.5 V Power Supply, MPM_REG2_1P5, National LP38693-ADJ

This is a low drop-out voltage regulator (250 mV at 500 mA with 5 V O/P) with output voltage range from 1.25 V to 9 V. The input has a wide input range from 2.7 V to 10 V. The output voltage can be set by controlling the voltage feedback input to ADJ. Pulling the enable pin down to a logic Low turns off the part.

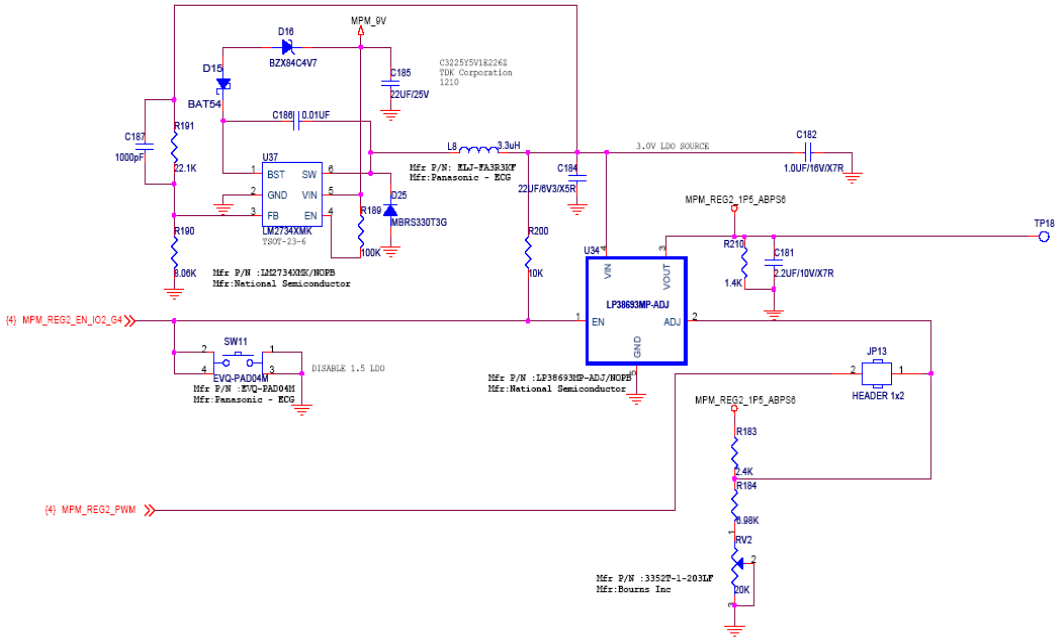

***Figure 6-4* • MPM Switching Power Supply 1.5 V**

5 V Power Supply 1, MPM_REG3_5V, Lineage Power ATA010A0X43

The 12 V singe in-line package (SIP) power module can deliver up to 10 A of output current with regulated output voltage from 0.75 VDC to 5.0 VDC over a wide range of input voltage (VIN = 8.3 – 14 VDC). The output voltage can be controlled using an external resistor between the TRIM and the GROUND pins of the module.

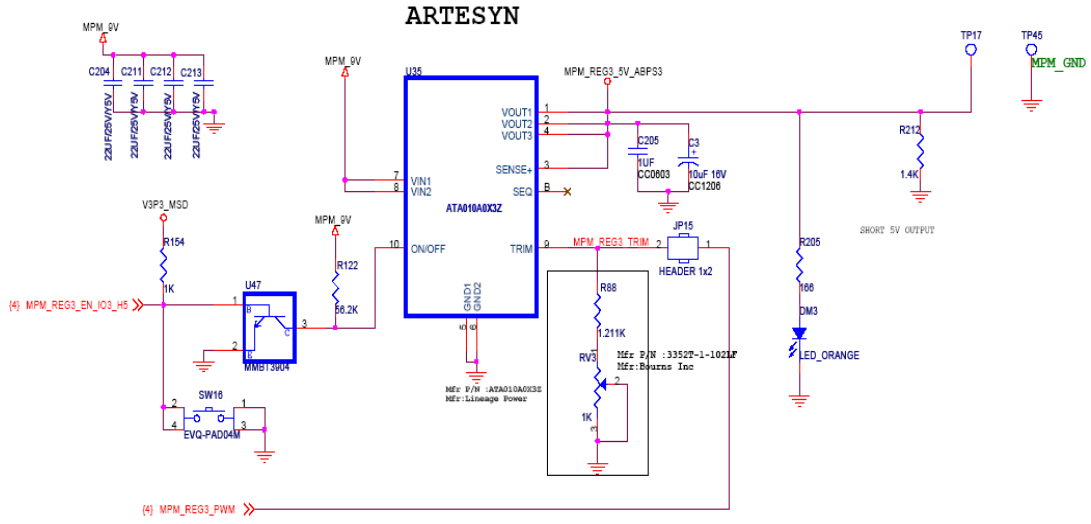

***Figure 6-5* • MPM DC-DC Power Supply 5 V**

5 Volt Supply 2, MPM_REG4_5V, Linear Technology LTM 4602

This DC/DC converter can deliver up to 6 A of output current with regulated output voltage from 0.6 VDC to 5.0 VDC over a wide range of input voltage (VIN = 4.5 – 20 VDC). Different output voltages can be programmed using resistors between the VOSET and SGND pins.

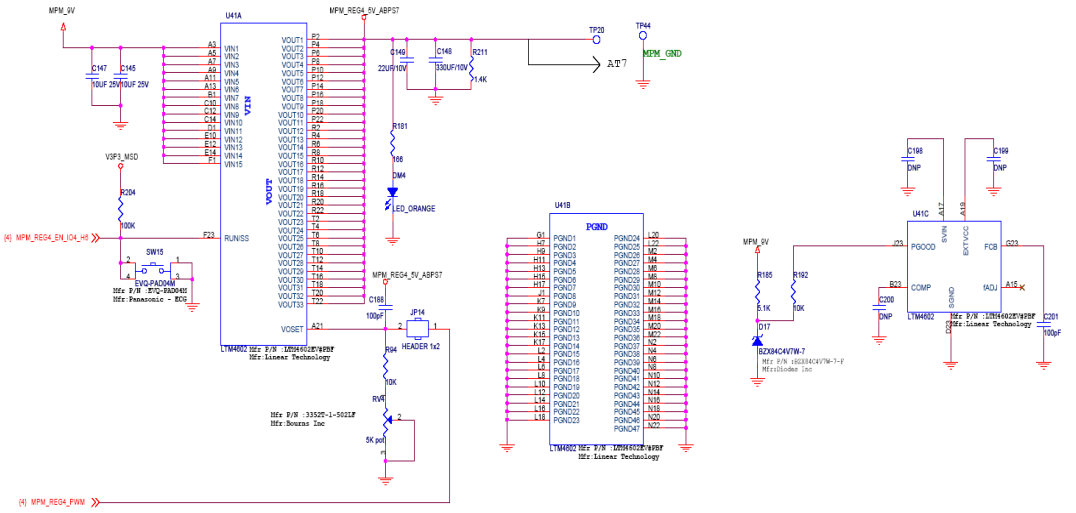

Figure 6-6 • **MPM DC-DC PowerSupply 5 V**

Power

The MPM-DC kit includes an external 9 V supply.

Connector

MPM Daughter Board Mixed Signal Header for SmartFusion Board

This connector is for interfacing MPM board to the SmartFusion board mixed signal header.

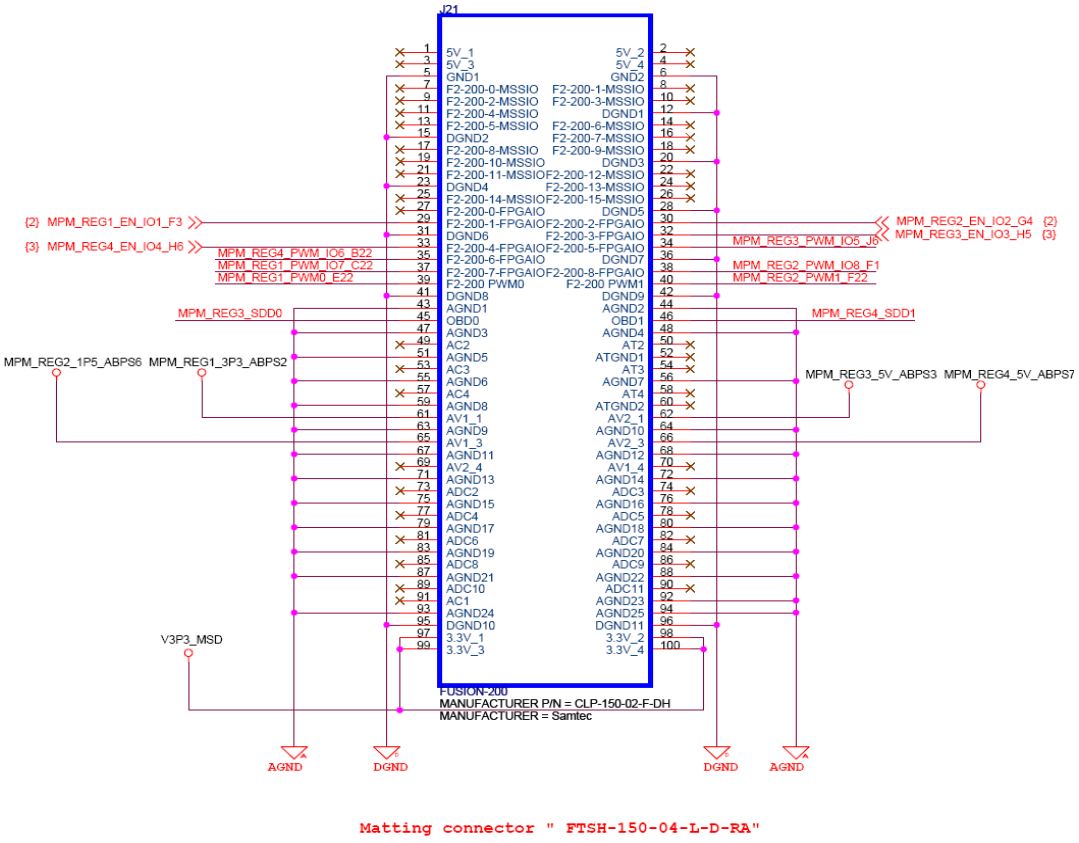

***Figure 6-7* • Mixed Signal Connector on MPM DB for Interface to SmartFusion Board**

Related Documents

- SmartFusion Documents
- Libero IDE Design Software

A – Manufacturing Information

MPM DB Board

The full PCB design layout is provided on the MPM Daughter Card Kit webpage. To view the PCB design layout files, you can use Allegro Free Physical Viewer, which can be downloaded from the Cadence Allegro Download page.

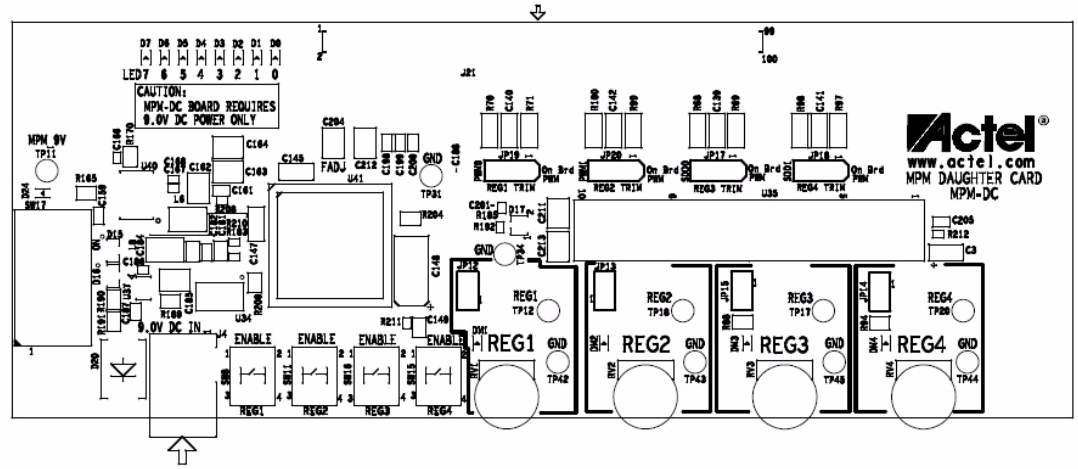

***Figure A-1* • Top Silk Screen for MPM Daughter Board**

B – Product Support

Actel backs its products with various support services including Customer Service, a Customer Technical Support Center, a web site, an FTP site, electronic mail, and worldwide sales offices. This appendix contains information about contacting Actel 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 Northeast and North Central U.S.A., call **650.318.4480**
From Southeast and Southwest U.S.A., call **650. 318.4480**
From South Central U.S.A., call **650.318.4434**
From Northwest U.S.A., call **650.318.4434**
From Canada, call **650.318.4480**
From Europe, call **650.318.4252** or **+44 (0) 1276 401 500**
From Japan, call **650.318.4743**
From the rest of the world, call **650.318.4743**
Fax, from anywhere in the world **650.318.8044**

Actel Customer Technical Support Center

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

Actel Technical Support

Visit the Actel Customer Support website (www.actel.com/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 Actel web site.

Website

You can browse a variety of technical and non-technical information on Actel's home page, at www.actel.com.

Contacting the Customer Technical Support Center

Highly skilled engineers staff the Technical Support Center from 7:00 a.m. to 6:00 p.m., Pacific Time, Monday through Friday. Several ways of contacting the Center follow:

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 tech@actel.com.

Phone

Our Technical Support Center answers all calls. The center retrieves information, such as your name, company name, phone number and your question, and then issues a case number. The Center then forwards the information to a queue where the first available application engineer receives the data and returns your call. The phone hours are from 7:00 a.m. to 6:00 p.m., Pacific Time, Monday through Friday. The Technical Support numbers are:

650.318.4460
800.262.1060

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

Index

Actel is the leader in low power FPGAs and mixed signal FPGAs and offers the most comprehensive portfolio of system and power management solutions. Power Matters. Learn more at www.actel.com.

Actel Corporation
2061 Stierlin Court
Mountain View, CA
94043-4655 USA
Phone 650.318.4200
Fax 650.318.4600

Actel Europe Ltd.
River Court,Meadows Business Park
Station Approach, Blackwater
Camberley Surrey GU17 9AB
United Kingdom
Phone +44 (0) 1276 609 300
Fax +44 (0) 1276 607 540

Actel Japan
EXOS Ebisu Buillding 4F
1-24-14 Ebisu Shibuya-ku
Tokyo 150 Japan
Phone +81.03.3445.7671
Fax +81.03.3445.7668
http://jp.actel.com

Actel Hong Kong
Room 2107, China Resources Building
26 Harbour Road
Wanchai, Hong Kong
Phone +852 2185 6460
Fax +852 2185 6488
www.actel.com.cn

50200275-2/9.10