
Mixed Signal Power Manager for SmartFusion Reference Design

User's Guide



Table of Contents

Introduction	5
Using MPM	6
Installation	6
Hardware Setup	6
MPM Reference Design Demo	12
MPM Design Files	17
MPM GUI Overview	22
MPM for SmartFusion Reference Design Flow	30
Purpose	30
Tools/Resources Used	30
Building the Softconsole Firmware Image	31
MSS ENVN Data Storage Clients	32
Libero SoC Design Flow	36
STAPL File Generation and GUI Integration	36
MPM for SmartFusion Reference Design I²C	37
I ² C Operation	37
I ² C Register Map	40
SmartFusion MPM Data Logging	56
Data Logging	56
MPM for SmartFusion Reference Design Trimming	61
Operation	61
GUI Operation	62
MPM Daughter Card Hardware Guide	65
Introduction	65
Kit Contents	66
Related Information	66
Board Description	66
Installation and Switch Settings	72
Manufacturing Information	75
Product Support	77
Customer Service	77
Customer Technical Support Center	77
Technical Support	77
Website	77
Contacting the Customer Technical Support Center	77

ITAR Technical Support.....	78
-----------------------------	----

List of Changes	79
------------------------------	-----------

List of Changes	79
-----------------------	----

Introduction

This document explains how to use the mixed signal power manager (MPM) reference design for the SmartFusion® customizable system-on-chip (cSoC) device, using the Digital MPM Daughter Card (DMPM-DC) connected to either the A2F-EVAL-KIT or the A2F500-DEV-KIT.

Note: Board revisions prior to A2F-EVAL-KIT Rev. 2 do not have I²C interfacing capabilities.

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 DMPM-DC.

MPM is a reference design that is programmed into the SmartFusion cSoC 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 (I²C) interface.

Based on Microsemi's SmartFusion cSoC, MPM delivers superior power monitoring, power sequencing, closed-loop trimming, and power-up and power-down control of up to 64 external power supplies which can be a mix of Analog Points of Load (APOLs) and Digital Points of Load (DPOLs). 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 Microsemi Libero System-on-Chip (SoC) tools.

This adds more flexibility, reduces total parts count on the board level, and increase system reliability by eliminating single points of failure.

With MPM, use of the Libero SoC tools is optional. Using the MPM GUI tool, you can configure the SmartFusion device that is programmed with the MPM reference design to revise set points and change sequencing without opening Libero SoC or reprogramming the SmartFusion cSoC FPGA fabric. The MPM GUI tool writes register values to on-chip embedded flash memory, which control power sequencing and monitoring functionality of the MPM reference design. The MPM GUI tool programs the board and configuration settings by launching FlashPro software which communicates with the target over a USB connected on board or separate FlashPro programmer device. In addition the MPM GUI can configure, control and monitor the MPM target over I²C using an appropriate USB to I²C Dongle.

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

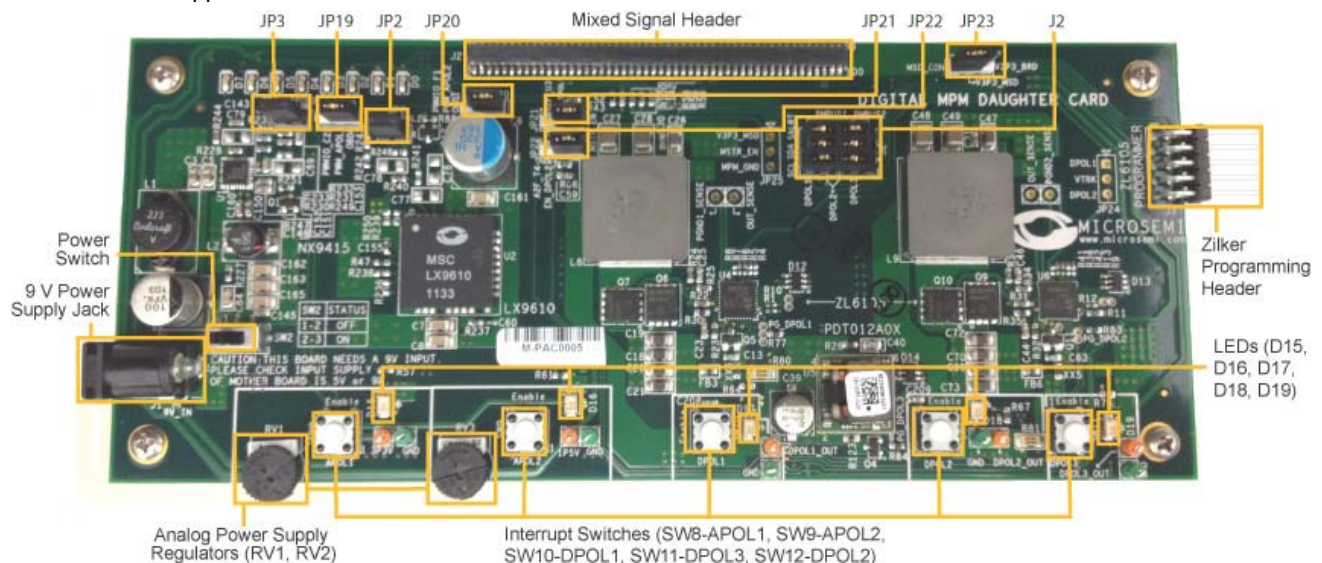


Figure 1 · MPM Daughter Card

Using MPM

Installation

Run the installer and follow the installation wizard instructions. By default MPM installs into the *C:\Microsemi\SF_MPM_RefDesign_v5.0* folder and this is the recommended location for it. In particular avoid installing it in a folder than is very deeply nested or has a very long path name otherwise some tools may encounter problems accessing files with names longer than 259 characters¹.

Once installed MPM adds the following options to the Windows start menu:

- ✓ Start
 - ✓ All Programs
 - Microsemi SmartFusion MPM Reference Design v5.0
 - ❖ Browse Design Files
 - Opens in Windows Explorer the folder containing the MPM Libero SoC hardware and SoftConsole firmware projects
 - ❖ MPM GUI
 - Runs the MPM GUI
 - ❖ Uninstall
 - Uninstalls MPM

Hardware Setup

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

You will need the following:

- SmartFusion Evaluation Kit Board – A2F[200]-EVAL-KIT Rev 2 or later²
(http://www.microsemi.com/soc/products/hardware/devkits_boards/smartfusion_eval.aspx)

or

- SmartFusion Development Kit Board – A2F500-DEV-KIT Rev A or later
(http://www.microsemi.com/soc/products/hardware/devkits_boards/smartfusion_dev.aspx)

and

- MPM Daughter Card (DMPM-DC)
(www.microsemi.com/soc/products/hardware/devkits_boards/mpm_dc.aspx)

and

- Devantech/Robot Electronics USB-ISS communications module

(http://robot-electronics.co.uk/acatalog/USB_I2C.html)

1. See this Microsoft MSDN article for more on path length issues: <http://msdn.microsoft.com/en-us/library/aa365247.aspx>.

2. Pre-Rev 2 boards will also work but do not expose the SmartFusion MSS I2C_1 signals which are required by default for communication between an I2C master such as the MPM GUI and the MPM I2C slave.

SmartFusion Evaluation Kit Board Setup

Ensure that the following jumpers are installed:

- J6 pins 1-2
- J7 pins 1-2

- JP6 pins 2-3
- JP7 pins 1-2
- JP9
 - ✓ optionally connect the Devantech/Robot Electronics USB-ISS to JP9 as explained below
- JP10 pins 1-2
- JP11 pins 1-2
- JP12 pins 1-2
- JP13 pins 1-2
- JP14 pins 1-2

- Turn the 50K_POT potentiometer wheel to the half-way point

SmartFusion Development Kit Board Setup

Ensure that the following jumpers are installed:

- J7
 - ✓ pins 2-3
 - ✓ pins 14-15
 - ✓ optionally connect the Devantech/Robot Electronics USB-ISS to J7 as explained below

- JP1 pins 1-2
- JP2 pins 1-2
- JP4
 - ✓ pins 1-3
 - ✓ pins 7-9
- JP5 pins 1-3
- JP6 pins 2-3
- JP7 pins 1-2
- JP8
 - ✓ pins 3-4
 - ✓ pins 7-8
 - ✓ pins 11-12
 - ✓ pins 15-16
- JP11 pins 1-2
- JP12 pins 1-2
- JP13 pins 1-2
- JP14 pins 1-2
- JP15 pins 1-2
- JP16 pins 2-3
- JP17 pins 1-2
- JP18 pins 1-2
- JP19 pins 1-2

- JP20 pins 1-2
- JP21 pins 1-2
- JP22 pins 2-3
- JP23 pins 1-2
- JP24 pins 1-2
- JP27 pins 1-2
- JP28 pins 1-2
- J32
 - ✓ pins 1-2
 - ✓ pins 3-4
 - ✓ pins 5-6
- SW9 middle position
- Turn the RV1 potentiometer thumbwheel to the half-way point

MPM Daughter Card Board Setup

Ensure that the following jumpers are installed:

- J2
 - ✓ J2A pin 1-J2B pin 2
 - ✓ J2C pin 3-J2D pin 4
 - ✓ J2A pin 6-J2B pin 7
 - ✓ J2C pin 8-J2D pin 9
 - ✓ J2A pin 11-J2B pin 12
 - ✓ J2C pin 13-J2D pin 14
- JP2 pins 1-2
- JP3 pins 1-2
- JP19 pins 1-2
- JP20 pins 1-2
- JP21 pins 1-2
- JP22 pins 1-2
- Turn the APOL1 RV1 potentiometer thumbwheel to the half-way point
- Turn the APOL2 RV3 potentiometer thumbwheel to the half-way point

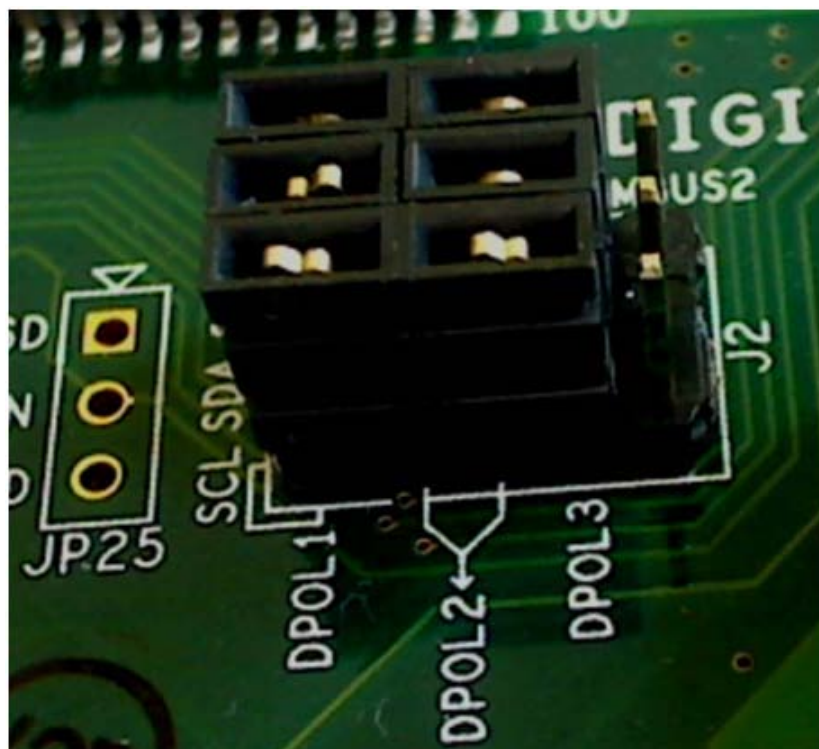


Figure 2 · DMPM-DC J2 jumper configuration

Programming MPM

The first time you use MPM you need to use the MPM GUI to program the MPM design to the target hardware.

- For the SmartFusion Evaluation Kit Board
 - ✓ Connect a mini USB cable between your PC and the J13 USB PROG FlashPro programmer connector
 - ✓ Connect a mini USB cable between your PC and the J14 USB/UART and power connector
- For the SmartFusion Development Kit Board
 - ✓ Connect the 5V power supply to the J1 5V INPUT SUPPLY connector
 - ✓ Connect a LCPS (Low Cost Programmer Stick) FlashPro programmer to J15
 - ✓ Connect a mini USB cable between your PC and the LCPS
 - ✓ Power the board on using SW6 VIN_5V

- Run the MPM GUI.
 - ✓ Select **Data > FlashPro > Choose STAPL** template and select the appropriate STAPL file for your target hardware (A2F200 or A2F500) from the C:\Microsemi\SF_MPM_RefDesign_v5.0\bin\template folder.¹
 - ✓ Select **Data > File > Load Values** to load the reference design demo settings from C:\Microsemi\SF_MPM_RefDesign_v5.0\bin\SF_MPM_Reference_Design.txt.
 - ✓ Select **Data > FlashPro > FlashPro Setup** and browse to and select the FlashPro software executable in your Libero SoC v10.1 or later installation.
 - ✓ Select **Data > FlashPro > Write NVM & Fabric** and the MPM GUI will launch FlashPro and program the full design (MSS configuration, MPM firmware, MPM configuration data in ENVN along with the FPGA fabric logic) to the target hardware. You will see a Command Shell “DOS Box” appear reporting progress. When prompted you can close this. At this stage the MPM target should be programmed with the MPM reference design.
- Power off the SmartFusion board by disconnecting the USB cable and/or power supply cables.
- Connect the MPM Daughter Card to the SmartFusion Evaluation or Development Kit Board by connecting their respective J21 mixed signal connectors/headers.
- Connect the 9V power supply to the MPM Daughter Card’s J1 9V_IN connector
- For the SmartFusion Evaluation Kit Board
 - ✓ Connect a mini USB cable between your PC and the J13 USB PROG FlashPro programmer connector
 - ✓ Connect a mini USB cable between your PC and the J14 USB/UART/power connector
- For the SmartFusion Development Kit Board
 - ✓ Connect the 5V power supply to the J1 5V INPUT SUPPLY connector
 - ✓ Connect a LCPS (Low Cost Programmer Stick) FlashPro programmer to J15
 - ✓ Connect a mini USB cable between your PC and the LCPS
 - ✓ Power the board on using SW6 VIN_5V
- Power the MPM Daughter Card on using SW2
- Reset the SmartFusion board by pressing SW3 RESET on the A2F-EVAL-KIT board or SW8 RESET on the A2F500-DEV-KIT board.
- Once the initial programming of the MPM design to the target has been performed then MPM configuration setting changes alone can be programmed to the target’s MSS ENVN either using **Data > FlashPro > Write NVM** or over an I²C connection between the MPM GUI and the MPM target hardware using **Data > I²C > Read Config Via I²C** and **Write Config Via I²C**.

I²C Setup

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

¹ The latest SmartFusion boards have an OLED display which sits on a small green interposer PCB board which is connected to the main board PCB. For such boards use the [MPM_A2F200_template.stp](#)/[MPM_A2F500_template.stp](#) files. Older boards have an OLED display which is connected directly to the main board PCB. For these older boards use the [MPM_A2F200_old_OLED_template.stp](#)/[MPM_A2F500_old_OLED_template.stp](#) files. If after programming MPM to your board, connecting the DMPM-DC and power cycling the hardware the OLED is blank then please make sure that you are using the appropriate STAPL file.

Use the included standard USB A/B cable to connect the USB-ISS to your PC. Install the drivers that are bundled with MPM GUI install in the C:\Microsemi\SF_MPM_RefDesign_v5.0\Devantech_USB-ISS_drivers folder.

After installing the drivers, and plugging the USB-ISS module into a spare USB port, you need to know to which COM port it has been assigned to. This will vary from system to system depending on how many COM ports you currently have installed. To find out where it is, right-click on your **My Computer** desktop icon and select **Properties > Hardware > Device Manager**. Now scroll down and open the **Ports (COM & LPT)** tab. You should see the USB serial port listed - COM5 in the example below. If you want to change the COM port number - just right-click on it, select **properties > advanced > COM port number** from the available list. The COM port default settings are sufficient.

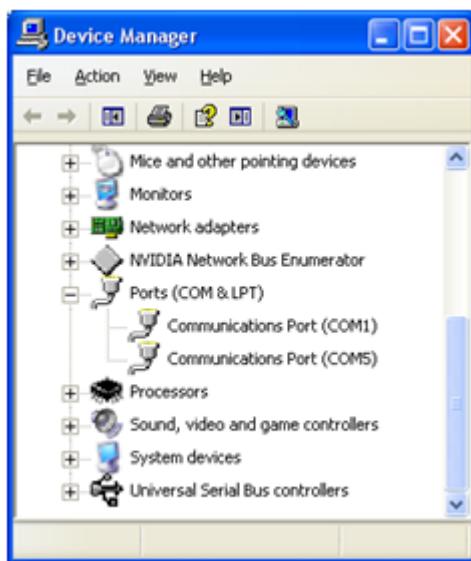


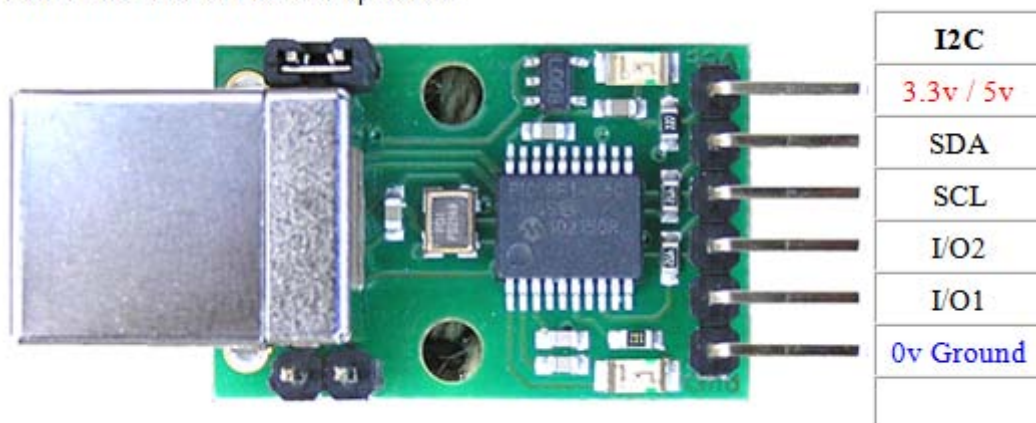
Figure 3 · Device Manager

To allow the MPM GUI to communicate with MPM, connect the Devantech USB-ISS to the SmartFusion board's MSS I2C_1 by installing female to female jumper cables between the following pins:

Table 1 · Devantech/Robot Electronics USB-ISS Connections

I ² C Signal Board	SCL	SDA	GND
Devantech/Robot Electronics USB-ISS http://www.robot-electronics.co.uk/htm/usb_iss_tech.htm	SCL (I/O 3)	SDA (I/O 4)	0V/Ground
A2F500-DEV-KIT Rev A www.microsemi.com/soc/products/hardware/devkits_boards/smartfusion_dev.aspx	J7 pin 6	J7 pin 10	J13 pin 2 or 10
A2F[200]-EVAL-KIT Rev 2 www.microsemi.com/soc/products/hardware/devkits_boards/smartfusion_eval.aspx	JP9 pin 2	JP9 pin 1	JP9 pin 3

Power Link - Remove for 3.3v operation.



Bootloader Link - Fit link for Bootloader mode

Figure 4 · Devantech/Robot Electronics USB-ISS

As explained earlier note the COM port assigned to the USB-ISS communications module. Run the MPM GUI, select **Data > I²C > Test I²C Dongle**. Select the USB-ISS COM port from the **Communications > Port**: drop-down list. Select **Test Dongle** from the **Activity > Action** drop-down list and click the **Go** button. If the test is successful then you should see the following in the **Status** text box.

```
Starting 'Test of Communications with Dongle'
```

```
I2C Speed set to 1 MHz
```

```
Communicated with dongle
```

```
Firmware Version: 07 02 80
```

```
Serial No: [00001425]
```

```
Test of Communications with Dongle Completed
```

If the test fails then you will see this and you need to review and correct the setup:

```
Starting 'Test of Communications with Dongle'
```

```
Failed to open I2C port
```

MPM Reference Design Demo

Without MPM GUI

Once the MPM reference design has been programmed to the target hardware the demo can be viewed even without the MPM GUI by using the SmartFusion Evaluation/Development Kit OLED and SW1/SW2 switches.

The OLED will display the following prompts:

```
SW1: power up/down
```

```
SW2: next OLED page
```

Use push-button switch SW1 on the SmartFusion 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 2 · 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 remains 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], Outputs [31:16], Outputs [47:32] and Outputs [63:48] 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 . . o o . 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], Inputs [31:16], Inputs [47:32] and Inputs [63:48] 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]/[46:32]/[63:47].
- Regulator Enables [15:0], [31:16],[47:32] and [63:48] 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 APOL1 regulator enable
 - ✓ Regulator Enable 2: MPM-DC APOL2 regulator enable
 - ✓ Regulator Enable 3: MPM-DC DPOL1 regulator enable
 - ✓ Regulator Enable 4: MPM-DC DPOL2 regulator enable
 - ✓ Regulator Enable 5: MPM-DC DPOL3 regulator enable

Using SW2, cycle the OLED display back to the Status page and select SW 1 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 and, if necessary, adjust the potentiometer thumbwheel to ensure that it reaches nominal value. If open-loop trimming is enabled, the open-loop trim pin voltage will only achieve nominal value if the potentiometer is suitably adjusted.

When power sequencing has completed and all regulators have reached nominal voltage, the status changes to Started and APOL closed-loop trimming is also enabled if applicable. Closed-loop trimming keeps the APOL channel output voltage at the nominal value specified in the GUI, even when the potentiometer is adjusted. You can disable trimming by removing the Trim jumper for a given regulator. Removing MPM-DC JP3 for MPM Channel 1/APOL1 and MPM-DC JP2 for MPM Channel 2/APOL2 disables closed-loop trimming and you see that the output voltage can be varied using the potentiometer for that channel. Reinstalling the jumper reactivates closed-loop trimming and brings it back to nominal.

Channel A6 is connected to the A2F-EVAL-KIT board 50K_POT/A2F500-DEV-KIT board RV1 potentiometer controlled 3.3V circuit and the voltage can be freely adjusted between 0V and 2.56V 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 APOL1
 - ✓ Microsemi NX9415 3.3 V nominal regulator
 - ✓ POT range when switched on approximately 3040 – 3553 mV
 - ✓ ACE ABPS2 \pm 5.12 V channel
 - ✓ Optionally open and closed loop trimmed using MSS SDD0 or CorePWM output 1 depending on MPM GUI configuration and MPM-DC jumper settings. The default demo setup is closed loop trimming using CorePWM output 1.
- Channel A2
 - ✓ MPM-DC APOL2
 - ✓ Microsemi LX9610 1.5 V nominal regulator
 - ✓ POT range when switched on approximately 1341 – 1640 mV
 - ✓ ACE ABPS6 \pm 2.56V channel
 - ✓ Optionally open and closed loop trimmed using MSS SDD1 or CorePWM output 2 depending on MPM GUI configuration and MPM-DC jumper settings. The default demo setup is closed loop trimming using CorePWM output 2.
- Channel A3
 - ✓ MPM-DC DPOL1
 - ✓ Intersil ZL6105 3.3V nominal DPOL
 - ✓ PMBus address 0x21
- Channel A4
 - ✓ MPM-DC DPOL2
 - ✓ Intersil ZL6105 1.5V nominal DPOL
 - ✓ PMBus address 0x20
- Channel A5
 - ✓ MPM-DC DPOL3
 - ✓ Lineage PDT012A0X 1.5V nominal DPOL
 - ✓ PMBus address 0x12
- Channel A6
 - ✓ A2F-EVAL-KIT 50K_POT/A2F500-DEV-KIT RV1
 - ✓ 3.3V always on (not switchable) power supply
 - ✓ Potentiometer range 0-2560mV
 - ✓ ACE CM0 0-2560mV ADC Direct Input

Note: If you use the MPM GUI to configure any of the voltage settings (for example, thresholds \pm hysteresis) out of range of the relevant underlying APOL ACE channel, the MPM firmware will ignore these as invalid and not display them. For example, if you set the OV2 on Channel 6 to 3300mV, this channel will disappear.

- Note:** The MPM DPOL channels are programmed over PMBus switched on/off using the DPOL discrete enable digital input and monitored using the DPOL Power Good/PG (or equivalent) digital output.
- Note:** It is the MPM demo program (main.c) which implements SW1/2 control, OLED prompts and the functionality underlying presses of SW1/2. The demo program is provided as a simple illustration of how MPM and the MPM API can be used but it is not part of the MPM reference design "core engine" per se. Refer to the demo program code to see how MPM can be used and deployed and feel free to adapt it to your own specific needs.

With MPM GUI

The MPM reference design demo can also be exercised using the MPM GUI communicating with the MPM target I2C slave via the Devantech/Robot Electronics USB-ISS I²C communications module. Ensure that the USB-ISS hardware and drivers are installed, configured and working as described earlier.

The default I2C slave address for MPM is 100 (decimal). All menu options under **Data > I²C** launch the **MPM I²C Communications** dialog with different default settings. For example **Data > I²C > Test I²C Dongle** gives:

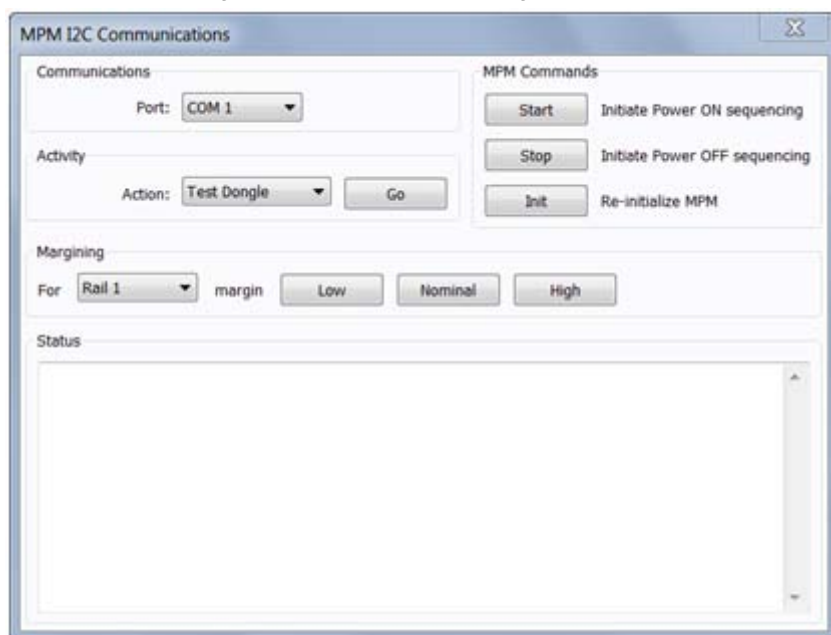


Figure 5 · MPM I²C Communications Dialog

As explained earlier select the appropriate COM port for the USB-ISS device from the **Communications > Port:** drop-down and then click the **Activity > Go** button to check that MPM GUI communication with the USB-ISS works. If not ensure that all hardware, driver and software configurations are correct.

Once MPM GUI communications with the USB-ISS device are working select **Find I²C Address** from the **Activity > Action** drop-down and click the **Activity > Go** button. For the default configuration with MPM I2C slave address 100, you should see:

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

If the **Find I²C Address** option is not used to dynamically scan for the MPM slave address then the GUI will by default try to use the I2C address specified under **Misc > Management Interface I²C > I²C Address**.

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

- Activity
 - ✓ **Test Dongle:** tests communication with the USB-ISS dongle only. The USB-ISS does not need to be connected to the MPM target.
 - ✓ **Find I2C Address:** dynamically scans to search for the MPM I2C slave address.
 - ✓ **Read:** reads the configuration settings from the target via I2C and populates the MPM GUI settings using them.
 - ✓ **Write:** writes the configuration settings in the MPM GUI to the MPM target via I²C. These settings will take effect the next time MPM is reinitialized (in stopped mode) and then (re)started. Note that it is now possible to Read, reconfigure the MPM configuration settings and Write in order to change the configuration of the MPM target without using the **Data > FlashPro > Write NVM** and a call out to the FlashPro software to write an update STAPL file.
 - ✓ **Monitor:** enters monitoring mode so that the GUI can display live updates of the target state using the Meters and Memory Map views.
 - ✓ **Go/Stop:** click the Go button to run the chosen activity. While running the Go button changes to a Stop button and can be pressed to prematurely terminate the active activity.
- MPM Commands
 - ✓ **Start:** initiate power on sequencing which is the same as pressing the A2F-EVAL-KIT/A2F-DEV-KIT SW1 when MPM is in stopped mode. No effect if MPM is already started.
 - ✓ **Stop:** initiate power off sequencing which is the same as pressing the A2F-EVAL-KIT/A2F-DEV-KIT board SW1 when MPM is in started mode. No effect if MPM is already stopped. Note that you can start MPM using the GUI and stop it using SW1 and vice versa if you want.
 - ✓ **Init:** re-initialize MPM – i.e. reload the latest MPM configuration data from ENVN. Only works when MPM is in stopped mode – no effect otherwise.
- Margining
 - ✓ **For:** select the channel/rail to which the following margining command will be applied.
 - ✓ **Margin:** the Low, Nominal and High buttons cause the selected channel/rail to margin to its low, nominal or high set voltage. No effect if MPM is not started or the relevant channel is not available.
- Status
 - ✓ Displays information about the progress and status of the most recent activity, command or margining operation.

MPM Design Files

SoftConsole Firmware Project

A SoftConsole workspace containing the reference design firmware and demonstration program application code is included in

C:\Microsemi\SF_MPM_RefDesign_v5.0\design_files\SoftConsole_workspace\SF_MPM_RefDesign.

To access this, do the following:

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

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

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

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

Review *main.c* to 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 I2C 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* file describes the public interface to the MPM driver.

The *mpm/mpm.c* file implements the actual MPM driver functionality that interfaces with the underlying MPM hardware design.

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

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

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

void mpm_task_init(void);

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

/* void mpm_start()
 *
 * Starts the MPM engine.
 *
 * If MPM is not in state mpm_is_stopped then this method does nothing and
 * just returns immediately otherwise if MPM is in state mpm_is_stopped
 * then this method:
 *
 * - puts MPM into state mpm_is_starting
 * - starts channel threshold monitoring
 * - starts channel open loop trimming where applicable
 * - initiates power on sequencing
 * - waits until power on sequencing has successfully completed
 * - starts channel closed loop trimming where applicable
 * - puts MPM into state mpm_is_running
 * - channel threshold monitoring remains active
 */
void mpm_start();

/* void mpm_stop()
```

```
*
* Stops the MPM engine
*
* If MPM is not in state mpm_is_running then this method does nothing and
* just returns immediately otherwise if MPM is in state mpm_is_running
* then this method:
*
* - puts MPM into state mpm_is_stopping
* - 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 passed in is
* a valid channel recognized by the MPM engine and false otherwise.
*
* For a channel to be valid it must meet the following criteria:
*
* - Signal name in ACE configuration must be "MPM_Channel_<n>..." where
*   <n> is a unique identification number between 1 and min(64,
```

```

* MPM_MAX_NUMBER_OF_CHANNELS) and "..." can be any other text.
* - ACE configuration for this channel must include one "UNDER" threshold
*   flag named "DOWN" and one "OVER" threshold flag named "UP" with any
*   valid voltage level (the threshold voltage levels are dynamically
*   adjusted at runtime by the MPM engine)
* - None of the threshold/hysteresis/nominal voltage values configured
*   through the MPM GUI is out of range of the underlying ACE (ABPS or
*   direct analog input) voltage channel.
*/
bool mpm_is_valid_channel(mpm_channel_number_t);

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

```

Libero SoC Hardware Project

A Libero SoC v10.0 project implementing the MPM hardware is included in
 C:\Microsemi\SF_MPM_RefDesign_v5.0\design_files\Libero_project\SF_MPM_RefDesign.

To use this:

1. Make a backup or work copy of the original Libero SoC project bundled with the package.
 2. Run Libero SoC v10.1 SP1.
 3. Browse to the Libero IDE project folder.
 C:\Microsemi\SF_MPM_RefDesign_v5.0\design_files\Libero_project\SF_MPM_RefDesign
 4. Select SF_MPM_RefDesign.prjx 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 SoC Catalog.
 6. The design comprises a top-level SmartDesign that instantiates the SmartFusion MSS and several fabric-based peripherals
- SmartFusion MSS resources used
 - ✓ MPM ACE channel and related configuration for APOL channels
 - ✓ MSS GPIOs used for interfacing to SW1/2 push-buttons, MPM-DC LEDs and DPOL PG (Power Good) inputs.
 - ✓ MSS eNVM data storage client placeholders for MPM firmware, MPM configuration data and MPM logging
 - ✓ MSS SSD for hard-macro driven trimming of APOL channels. The MSS SSD operates between 0 V and 2.56 V, while Core PWM (below) operates between 0 V and 3.3 V.
 - ✓ MSS I2Cs for OLED output (I2C_0) and MPM I2C slave interfacing (I2C_1)
 - ✓ Clock, reset and MSS/FPGA interrupt and AMBA configuration.
 - ✓ Timer control (TIM2) for scheduling of MPM firmware operations.
 - ✓ RTC for log record time stamping.

- Fabric-based logic

- ✓ CoreAPB3 for interfacing MSS to fabric DirectCore peripherals.
- ✓ 2 x CoreGPIO (MPM_GPIO_Digital_IOs + MPM_GPIO_Digital_IOs_II) implementing up to 64 general digital inputs and 64 flag digital outputs.
- ✓ 2 x CoreGPIO (MPM_GPIO_Regulator_Enables + MPM_GPIO_Regulator_Enables_II) implementing up to 64 APOL/DPOL regulators enable digital outputs.
- ✓ CorePWM (MPM_PWM_Trimming_Outputs) implementing up to 16 APOL channel trimming PWM DAC outputs.
- ✓ CoreI2C for MPM to DPOL PMBus connectivity.
- ✓ CoreGPIO (MPM_GPIO_Mode) for mode selection to allow mapping APOL 2 and DPOL 3 to upper 32 regulator enables.
- ✓ CoreInterrupt (MPM_CoreInterrupt) for routing I²C interrupts to fabric int.
- ✓ Multiplexor (MPM_Regulator_Mux_0) to switch regulator enables between low and high 32 bit blocks.
- ✓ BIBUFs for PMBus bidirectional signal support.

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.

MSS ACE Configuration for APOL Channels

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(64, 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 \pm 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.

DPOL Channels

MPM DPOL channels require the following:

1. The DPOL for the channel must be connected to the MPM PMBus
2. Each DPOL on the MPM PMBus must obviously have a unique I²C/PMBus slave address
3. As with APOL channels the enable/switching input to the DPOL must be connected to the relevant MPM_GPIO_Regulator_Enable output (Channel <n> enable must be connected to MPM_GPIO_Regulator_Enable:GPIO_OUTS [n-1] where $1 \leq n \leq \min(64, \text{MPM_MAX_NUMBER_OF_CHANNELS})$). Note that in this release DPOLs are switched only using the discrete enable input and not also using PMBus OPERATION (0x01) command.
4. The Power Good (PG) DPOL output signal must be connected to a suitable MPM input and the MPM firmware modified to handle this. Refer to how the MPM-DC Channels 3/4/5 (DPOL1/2/3) are connected in the reference design Libero project.

MPM GUI Overview

This section briefly summarizes the layout of the MPM GUI.

Property Page Tabs

Built-in Help

The main display of the Mixed Signal Power Manager (MPM) has four property page tabs:

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

Numerous configuration settings can be set from these tabs. A built-in help panel can be shown using the **Help > Help** menu option.

When the title of any of the groups of configuration parameters is clicked, the background to that group of parameters changes to yellow and the text in the help panel updates to describe those options.

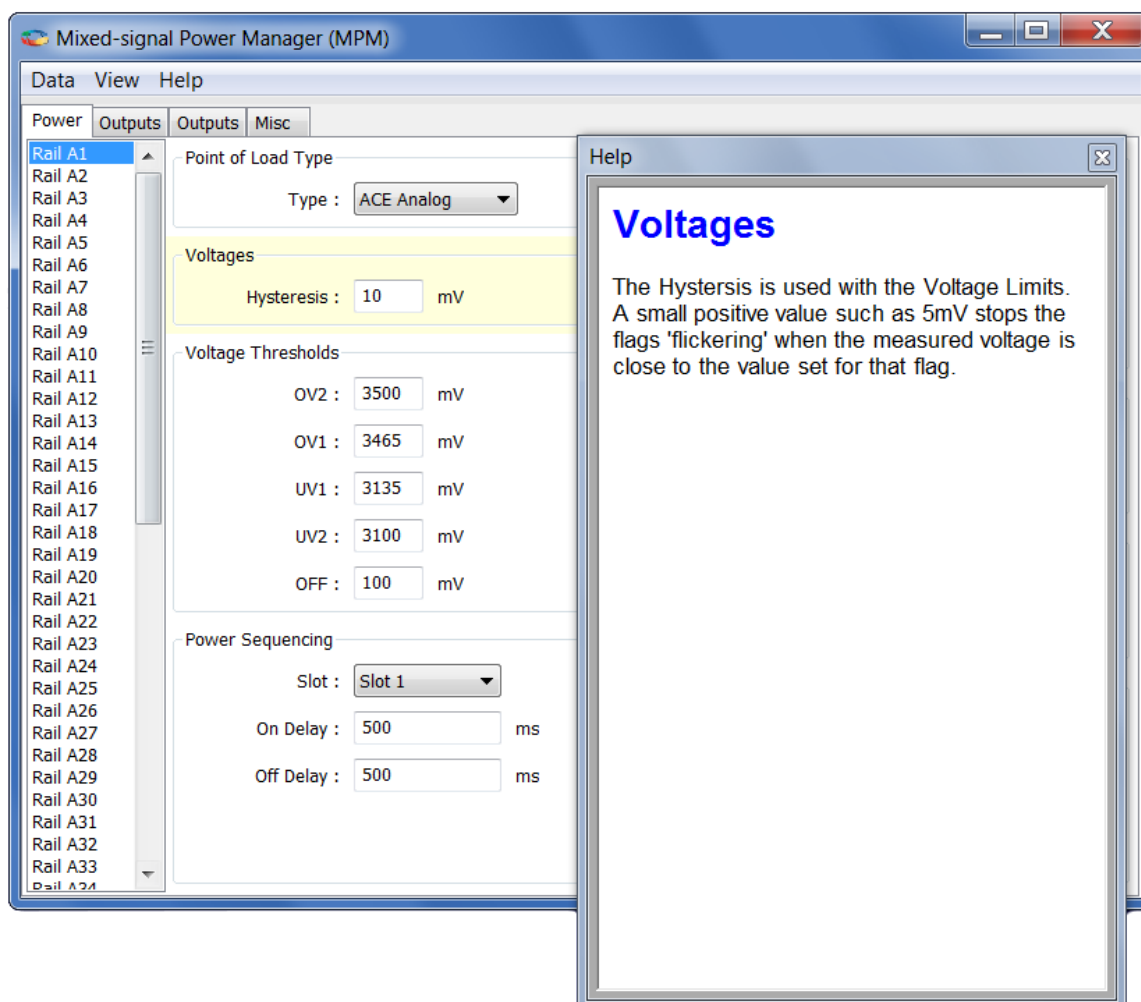


Figure 6 · Built-In Help Window

Power Property Page Tab

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

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

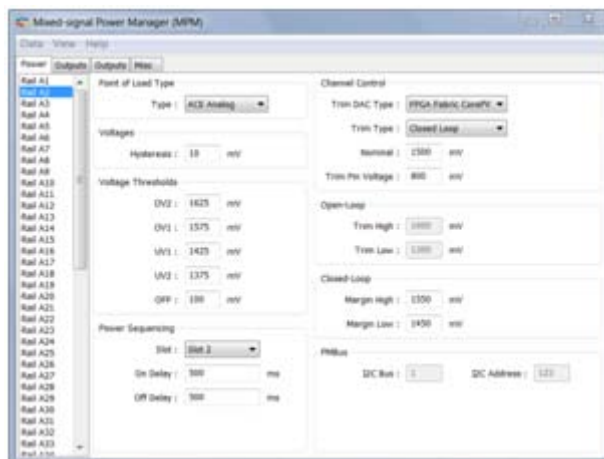


Figure 7 · Power Property Page Tab Window

Outputs Property Page Tabs

The Outputs tabs are used to define the conditions for and polarity of up to 64 digital outputs based on the states of various MPM channels. There are two such tabs, one for outputs 1 to 32 and one for outputs 33 to 64. The tabs support configuration of if/how each digital output is combined with a corresponding digital input, when the output is changed, and whether and how the output is logged.

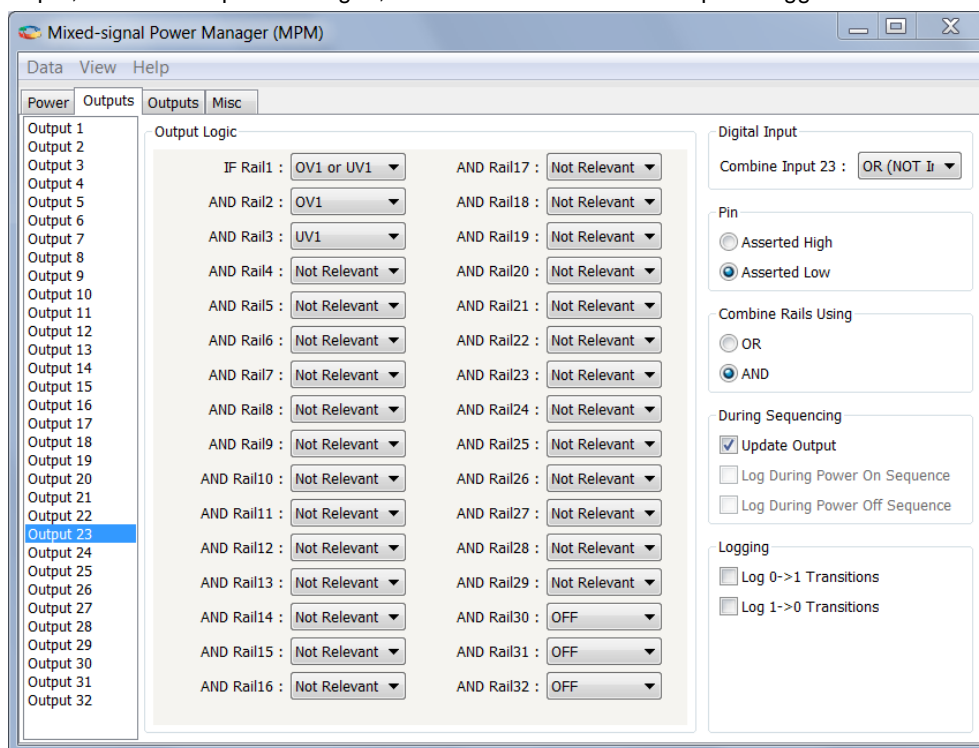


Figure 8 · Outputs Property Page Tab Window

Miscellaneous Property Page Tab

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

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

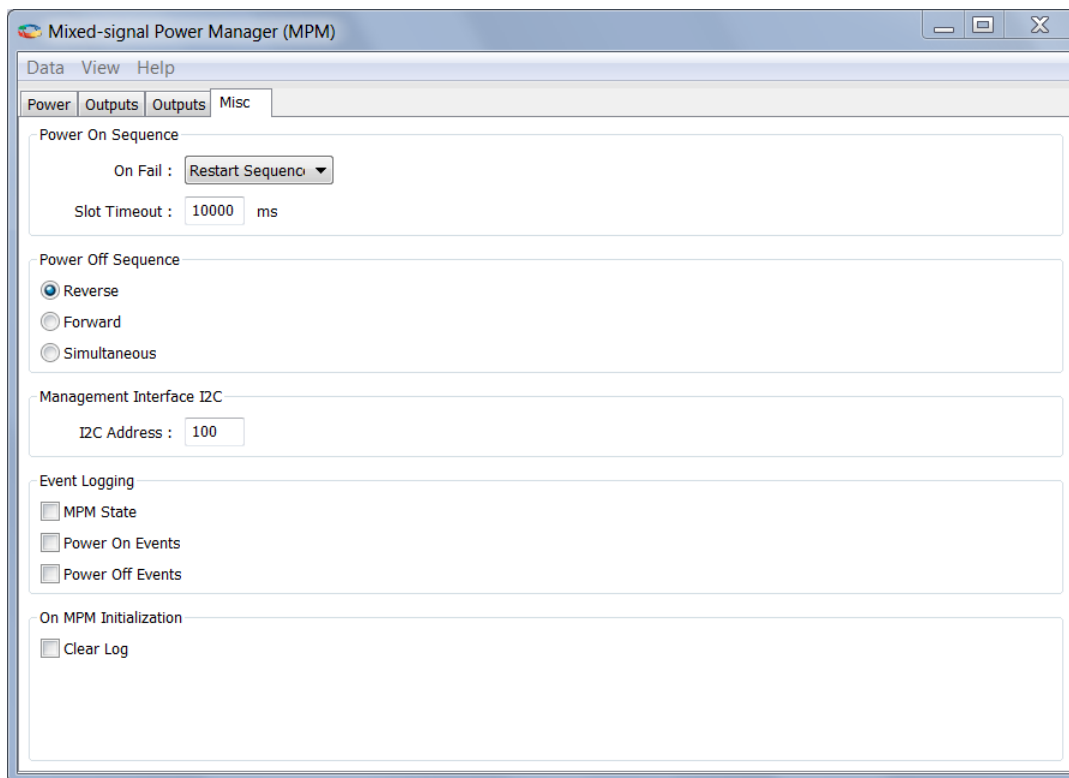


Figure 9 · Miscellaneous Property Page Tab Window

Menu Options

Figure 10 shows what activities can be initiated from the MPM GUI menu bar

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

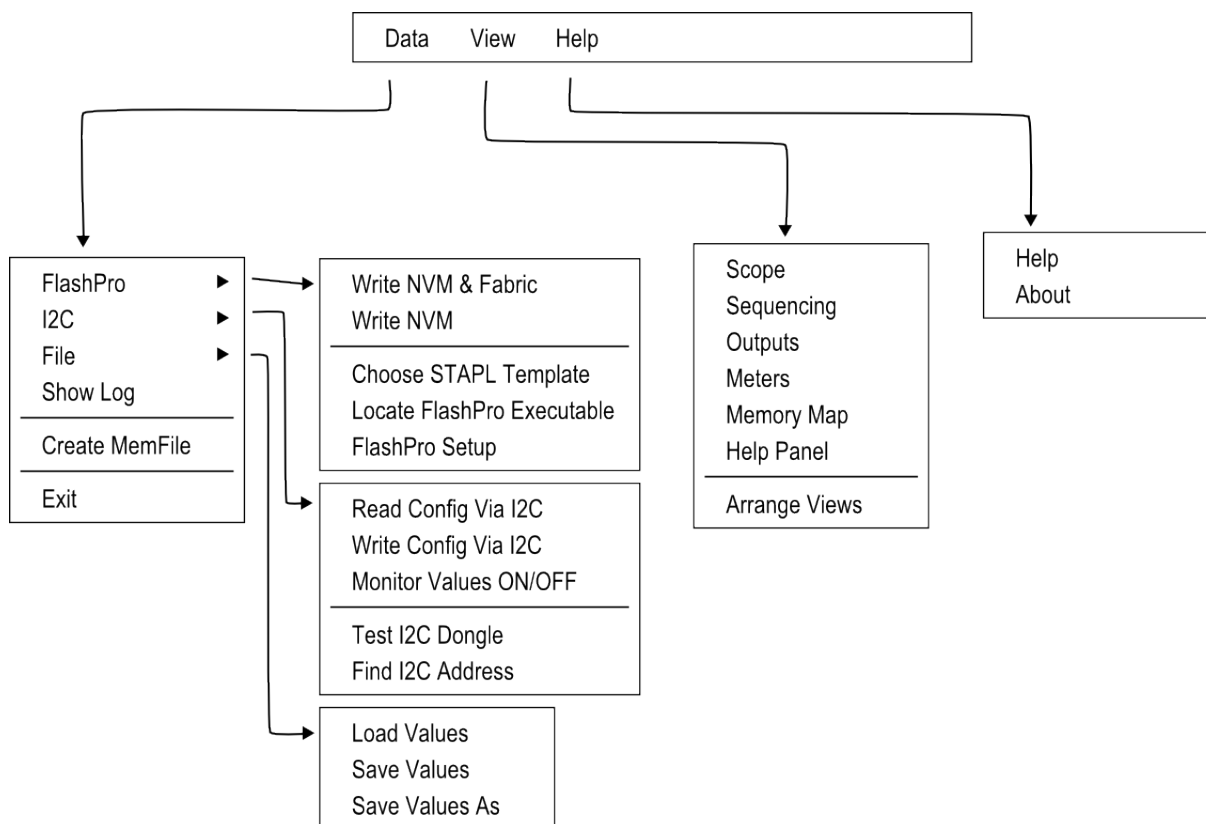


Figure 10 • Menu Options

I²C Communications

The MPM I²C Communications dialog is for communication from the GUI to MPM using I²C via the Devantech/Robot Electronics USB-ISS communications module/dongle. In the **Activity** box choosing an activity and pressing **Go** will start that activity. A log of what happens will be displayed in the Status view

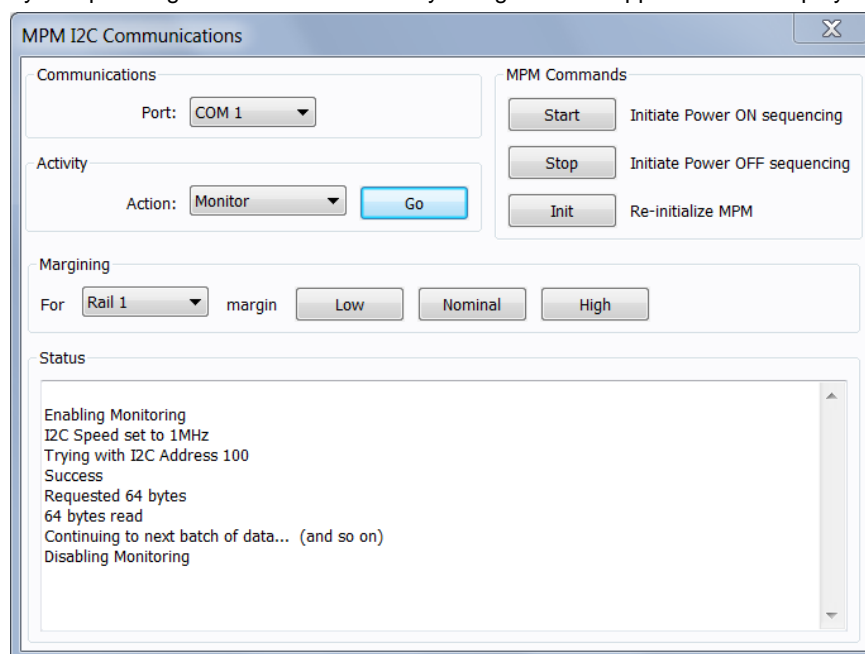


Figure 11 • I²C Communication

Table 3 · Status

Activity	Description
Test Dongle	Tests communications between the PC and the dongle only. The dongle does not need to be connected to the Mixed Signal Power Manager.
Read	Reads the entire MPM configuration. The configuration values in the main tabs are updated, as are the values in the GUIs copy of the memory map.
Write	Writes the entire MPM configuration using values in the main tab.
Monitor	Repeatedly read the live analog values at address 0x1900 to 0x197F. This activity continues until “Stop” is pressed.
Find I²C Address	This scans the I ² C bus looking for an MPM device. The I ² C address in the Misc tab is the first address the GUI tries.

The text on the **Go** button changes to **Stop** when there is I²C activity. The activity can be stopped by clicking the **Stop** button. The dialog also allows for single actions – to start power ON or power OFF sequencing, to reinitialize MPM and to perform margining on a particular rail.

Log Window

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

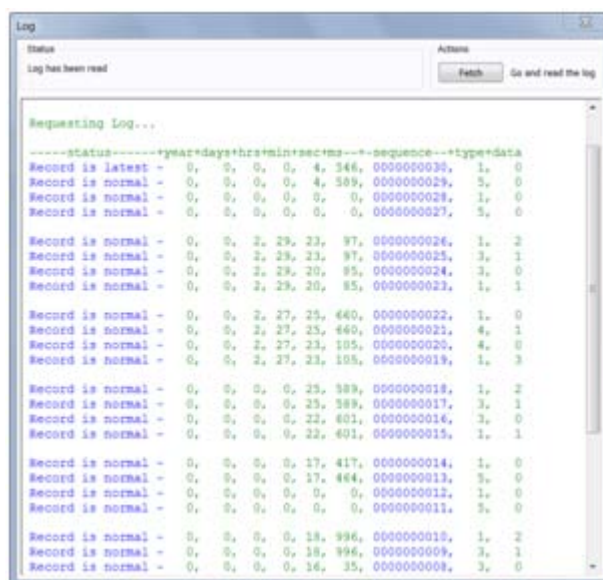


Figure 12 · Log Window

Views

Sequencing View

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

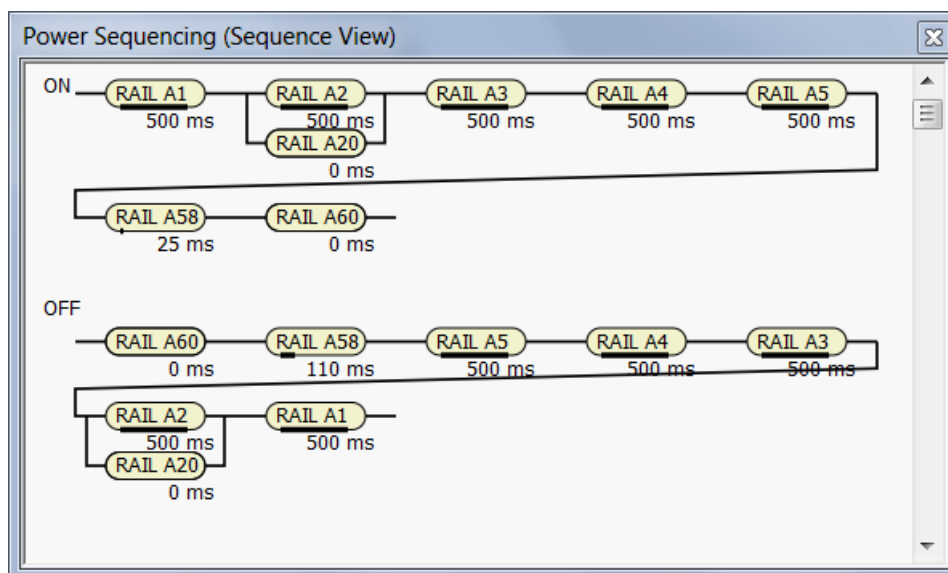


Figure 13 · Sequencing View

Scope View

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

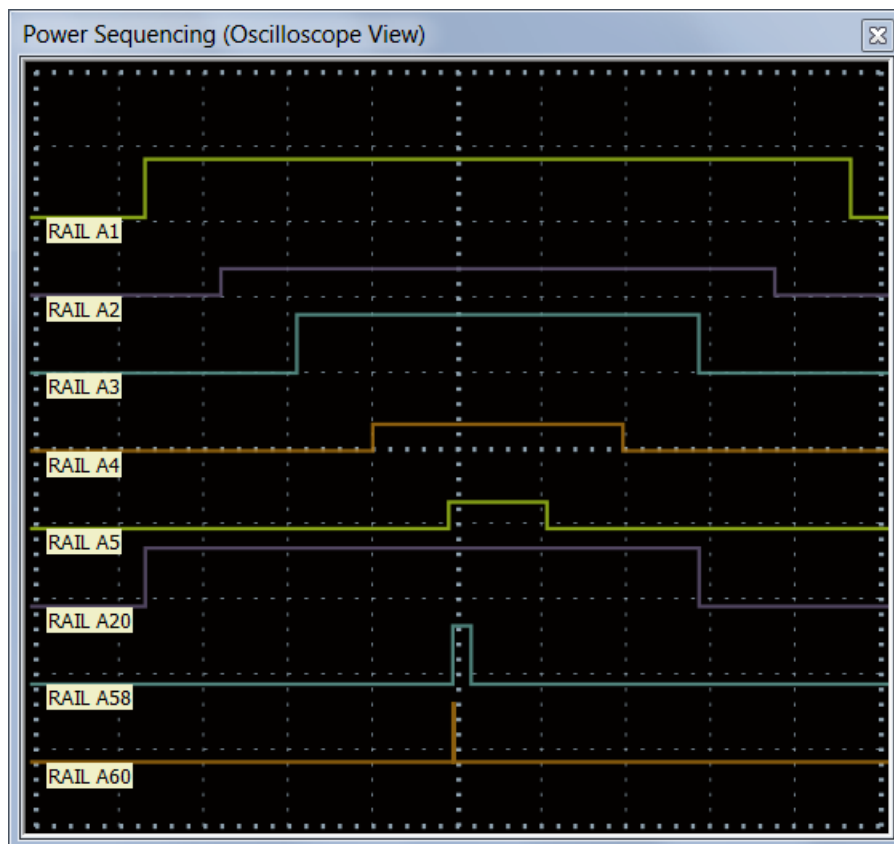


Figure 14 · Scope View

Output View

The Outputs view gives a schematic view of the output logic selected on the current outputs tab.

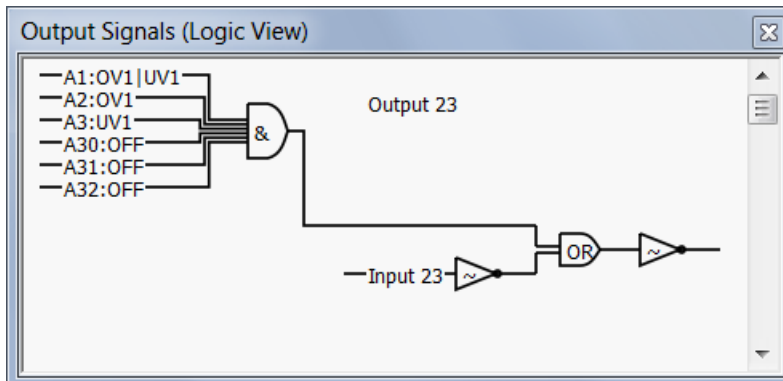


Figure 15 · Output View

Meters View

The meters view shows the values of the first twelve analog values read back by the MPM firmware at address 0x1900. For the meters to reflect live values the USB-ISS module must be connected to the SmartFusion board and monitoring must be active on the MPM I²C Communications Dialog. The meters ranges automatically adjust to your voltage conditions. Values up to 50 volts are shown. The actual value is shown numerically in Figure 16.

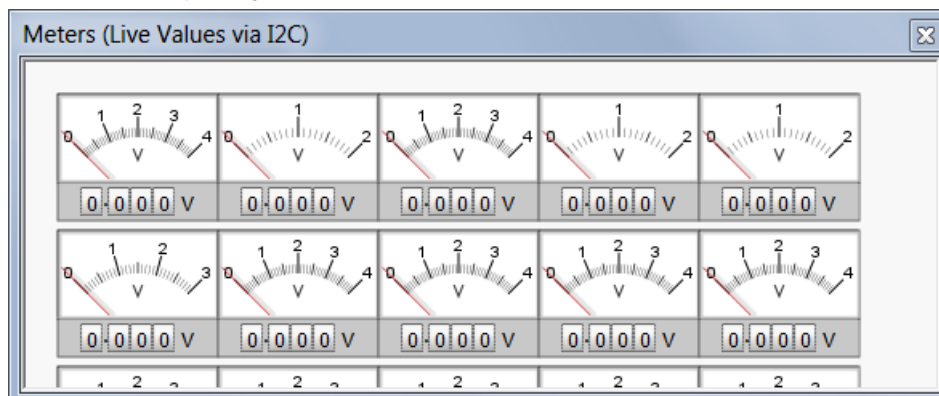


Figure 16 · Meters View

Memory Map

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

Memory Map

R0000:	00	00	0A	00	AC	0D	89	0D	3F	0C	1C	0C	64	00	01	00
R0010:	F4	01	F4	01	05	00	E4	0C	00	00	20	03	EE	02	52	03
R0020:	48	0D	80	0C	01	00	7B	00	00	00	00	00	00	00	00	00
R0030:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
R0040:	00	00	0A	00	59	06	27	06	91	05	5F	05	64	00	02	00
R0050:	F4	01	F4	01	05	00	DC	05	00	00	20	03	E8	03	14	05
R0060:	0E	06	AA	05	01	00	7B	00	00	00	00	00	00	00	00	00
R0070:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
R0080:	01	00	00	00	D3	0E	2E	0E	9A	0B	F5	0A	0A	00	03	00
R0090:	F4	01	F4	01	04	00	E4	0C	00	00	00	00	00	00	00	00
R00A0:	89	0D	3F	0C	01	00	21	00	00	00	00	00	00	00	00	00
R00B0:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
R00C0:	01	00	00	00	BD	06	72	06	46	05	FB	04	0A	00	04	00
R00D0:	F4	01	F4	01	04	00	DC	05	00	00	00	00	00	00	00	00
R00E0:	27	06	91	05	01	00	20	00	00	00	00	00	00	00	00	00
R00F0:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
R0100:	01	00	00	00	BD	06	72	06	46	05	FB	04	0A	00	05	00
R0110:	F4	01	F4	01	04	00	DC	05	00	00	00	00	00	00	00	00
R0120:	27	06	91	05	01	00	12	00	00	00	00	00	00	00	00	00
R0130:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
R0140:	00	00	0A	00	FC	08	98	08	08	07	A4	06	64	00	00	00
R0150:	00	00	00	00	00	00	D0	07	00	00	00	00	00	00	00	00
R0160:	34	08	6C	07	01	00	7B	00	00	00	00	00	00	00	00	00
R0170:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
R0180:	00	00	0A	00	74	0E	AC	0D	1C	0C	54	0B	64	00	00	00
R0190:	00	00	00	00	00	00	E4	0C	00	00	20	03	BC	02	84	03
R01A0:	48	0D	80	0C	01	00	7B	00	00	00	00	00	00	00	00	00
R01B0:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
R01C0:	00	00	0A	00	74	0E	AC	0D	1C	0C	54	0B	64	00	00	00
R01D0:	00	00	00	00	00	00	E4	0C	00	00	20	03	BC	02	84	03
R01E0:	48	0D	80	0C	01	00	7B	00	00	00	00	00	00	00	00	00
R01F0:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
R0200:	00	00	0A	00	74	0E	AC	0D	1C	0C	54	0B	64	00	00	00

Figure 17 · Memory Map View

MPM for SmartFusion Reference Design Flow

Purpose

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

Tools/Resources Used

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

- Libero SoC v10.1 SP1 (v10.1.1.6)
- SoftConsole v3.3 (v3.3.0.14)
- IP cores – MSS and DirectCores
 - ✓ SmartFusion MSS v2.5.106
 - ✓ CoreAPB3 v3.0.103
 - ✓ CoreGPIO v3.0.120
 - ✓ CoreI2C v7.0.102
 - ✓ CoreInterrupt v1.1.101
 - ✓ CorePWM v4.1.106
- Firmware drivers
 - ✓ SmartFusion MSS drivers
 - ✓ SmartFusion CMSIS-PAL v2.3.103
 - ✓ SmartFusion MSS ACE driver v2.3.105
 - ✓ SmartFusion MSS GPIO driver v2.0.105
 - ✓ SmartFusion MSS I2C driver v3.0.102 - This has been modified from the stock driver to integrate with Free RTOS when using a slave write handler
 - ✓ SmartFusion MSS eNVM driver v2.2.102
 - ✓ SmartFusion MSS PDMA driver 2.0.102
 - ✓ SmartFusion MSS RTC driver 2.0.100
 - ✓ SmartFusion MSS SPI driver 2.1.100
 - ✓ SmartFusion MSS Timer driver v2.1.101
 - ✓ SmartFusion MSS Watchdog driver v2.0.107
- DirectCore drivers
 - ✓ Hardware Abstraction Layer (HAL) v2.2.102
 - ✓ CoreGPIO driver v3.0.101
 - ✓ CoreI2C driver v3.0.102
 - ✓ CorePWM driver v2.1.107
- Other drivers
 - ✓ Unified I2C driver abstraction layer (DAL) v1.0.101
 - ✓ PMBus driver v1.1.102 - This has been modified to fix a bug in the linked list handling
 - ✓ SPI Flash driver - This has been modified from the version supplied with the SPI sample code to enhance reliability with the A2F200 and provide detection of DEV Kit vs EVAL Kit.

- Non Microsemi software
 - ✓ FreeRTOS version 7.1.1

Building the Softconsole Firmware Image

The SmartFusion MPM reference design release contains a SoftConsole v3.3 project, which may be modified by the user, located in the Softconsole workspace at

C:\Microsemi\SF_MPM_RefDesign_v5.0\design_files\SoftConsole_workspace\SF_MPM_RefDesign

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

Once the project has been compiled and tested, the user can build a final version for deployment in NVM by **Project > Build Configurations > Set Active > Release**. The Release target is configured to link using the mpm/mpm-production-run-from-envm.ld linker script. This linker script creates a firmware image suitable for storage in eNVM which has been mirrored to 0x00000000. Once stored in eNVM mirrored to 0x00000000 the program will start executing directly from eNVM. The Figure 18 shows a screenshot of this option

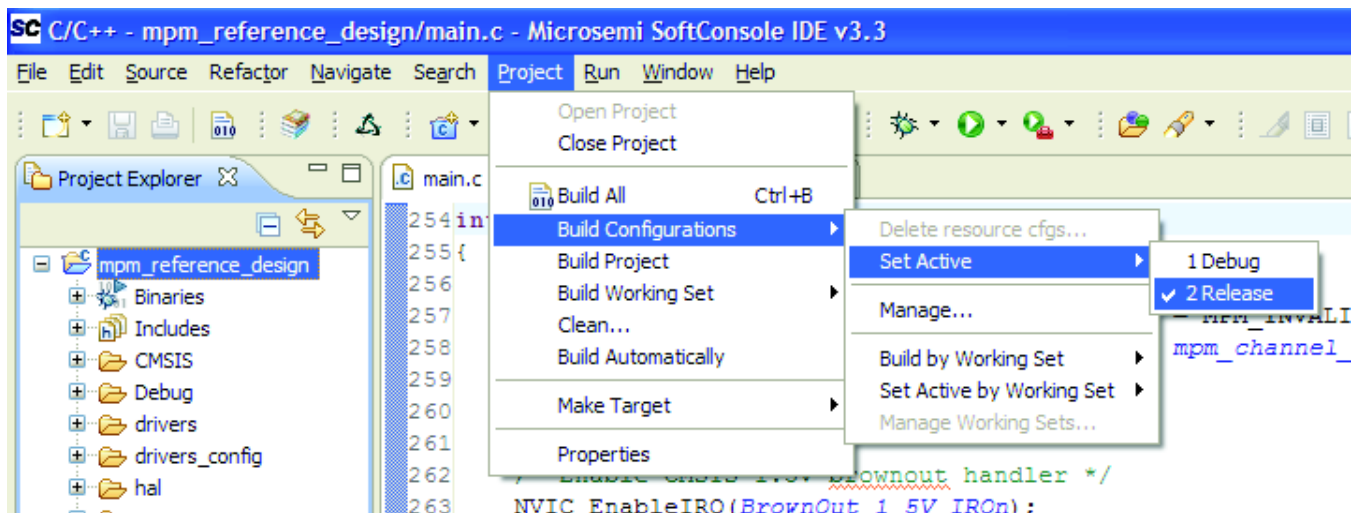


Figure 18 · Setting Active Build to Release in SoftConsole

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 "The SmartFusion MPM Reference design defines three MPM specific eNVM data storage clients as illustrated in Figure 19 for this step.

MSS ENVM Data Storage Clients

The SmartFusion MPM Reference design defines three MPM specific eNVM data storage clients as illustrated in Figure 19.

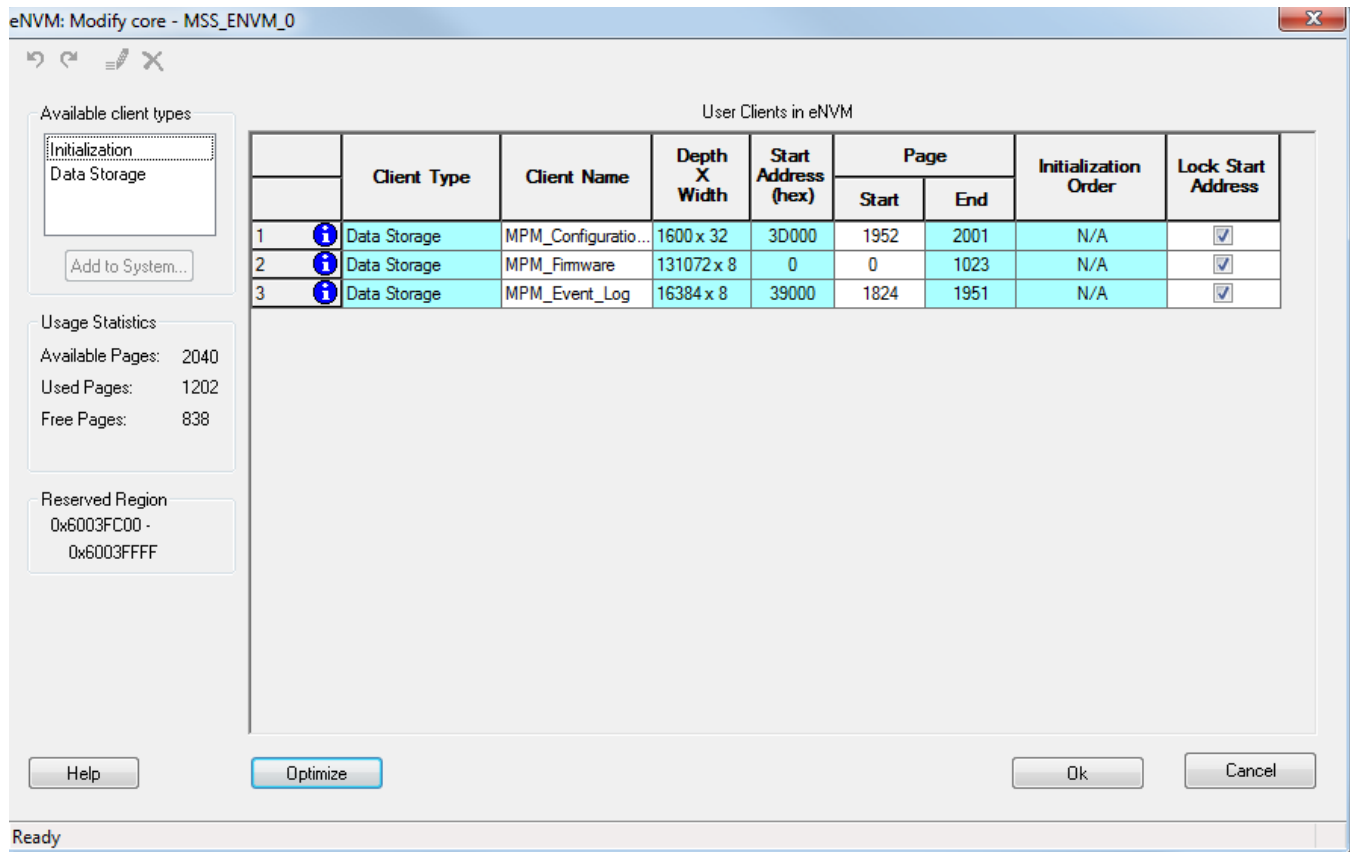


Figure 19 · MSS eNVM Configuration GUI

1. MPM_Configuration_Data

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

The MPM_Configuration_Data ENVM data storage client configuration is as follows in the reference design:

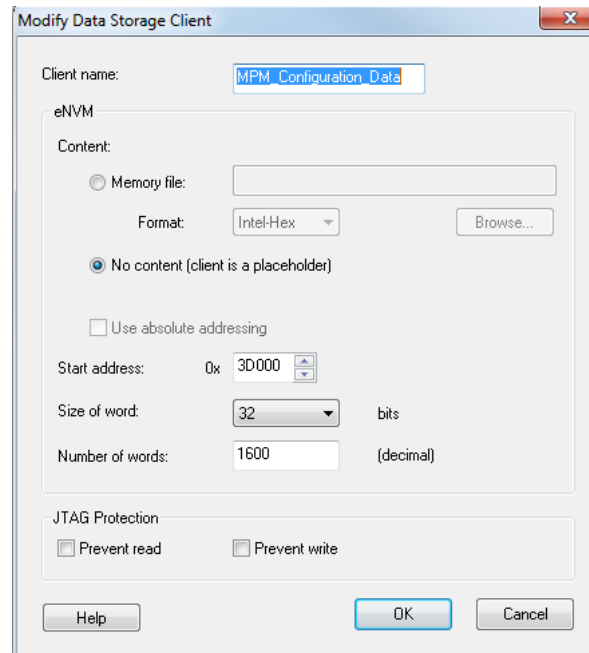


Figure 20 · Data Storage Client for MPM Configuration Data

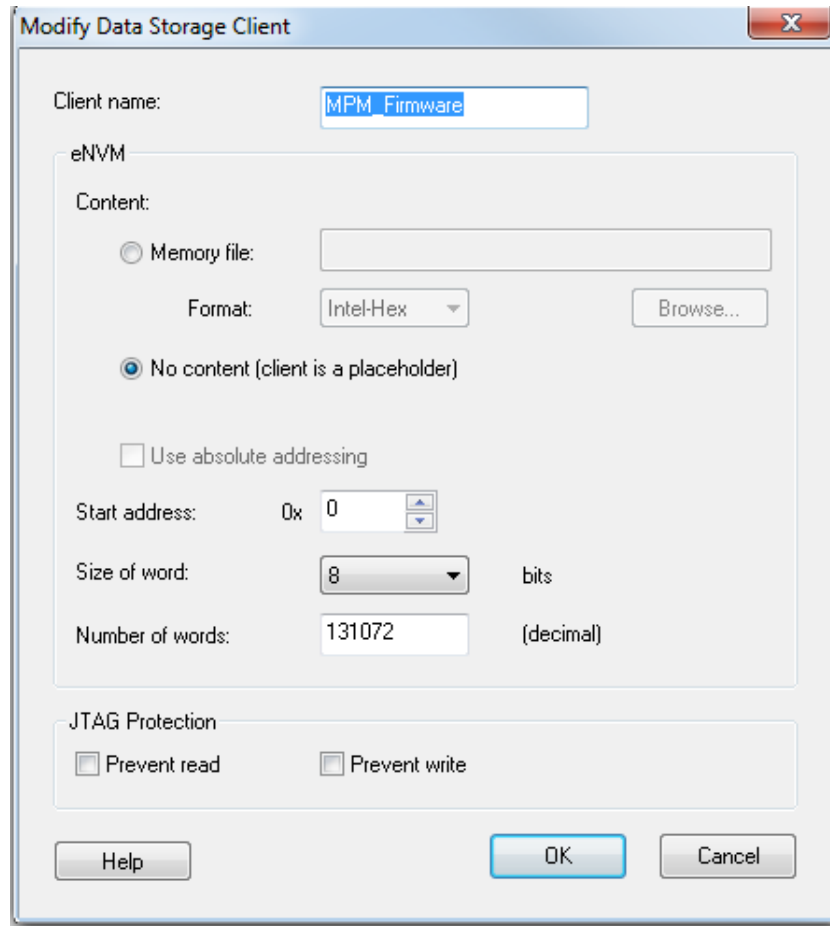
Note that if the base address of the MPM_Configuration_Data eNVM data storage block needs to change (e.g. because the SmartFusion MSS/ACE/etc. configuration data also stored in eNVM extends into this region) then the address must be changed consistently in both the eNVM data storage client configurator and by changing this manifest constant in mpm.c:

```

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

2. MPM_Firmware

This is the region in NVM that stores the MPM firmware for execution out of reset. The MPM_Firmware eNVM data storage client configuration is as shown in Figure 21:



The screenshot shows a 'Modify Data Storage Client' dialog box. The 'Client name' field contains 'MPM_Firmware'. Under the 'eNVM' section, the 'Content' radio button is selected for 'No content (client is a placeholder)'. The 'Format' dropdown is set to 'Intel-Hex'. The 'Start address' is '0x 0', 'Size of word' is '8 bits', and 'Number of words' is '131072 (decimal)'. The 'JTAG Protection' section has 'Prevent read' and 'Prevent write' both unchecked. At the bottom are 'Help', 'OK', and 'Cancel' buttons.

Figure 21 · Data Storage Client for MPM Firmware

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

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

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

3. MPM_Event_Logging

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

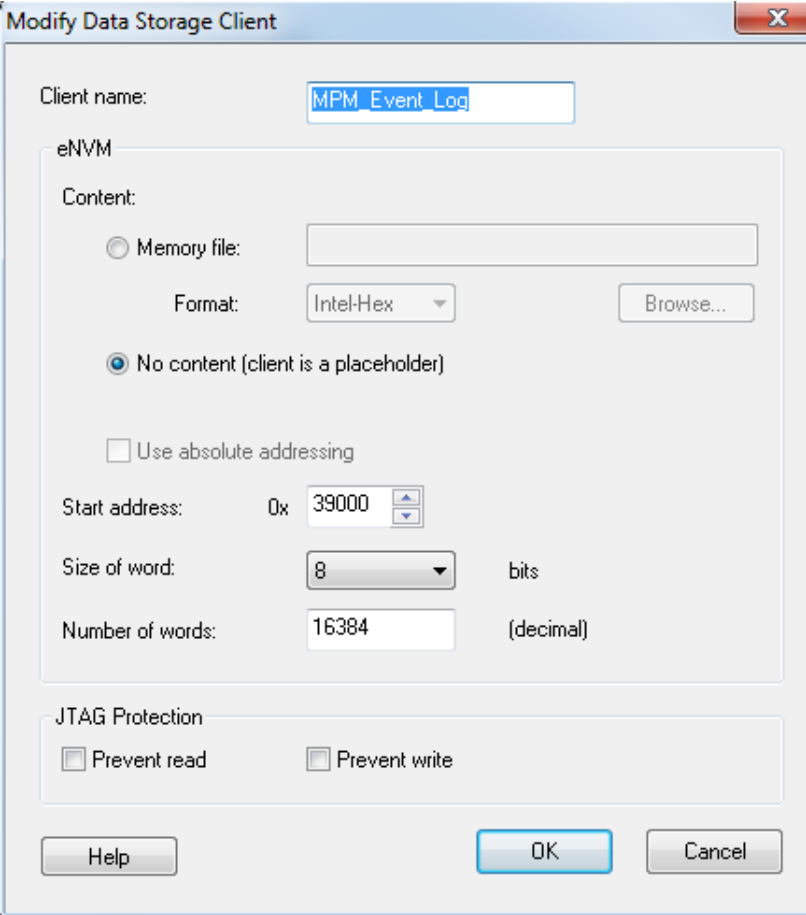


Figure 22 · Data Storage Client for MPM_Event_Logging

Note that, as with MPM_Configuration_Data, the base address and size of this memory region must be consistent in both the eNVM data storage client configurator and the mpm.c file:

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

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

Libero SoC Design Flow

If any changes have been made to the MPM Reference Design Libero project (e.g. changing die from A2F200 to A2F500, modification of MSS configuration settings, addition of FPGA fabric logic etc.) the project can be run through the normal Libero SoC design flow.

STAPL File Generation and GUI Integration

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

To do this run FlashPro interactively from Libero SoC and export the STAPL file for the PDB file. Once generated this STAPL file can be copied to the MPM GUI

C:\Microsemi\SF_MPM_RefDesign_v5.0\template folder where it can be selected from the GUI using the **Data > FlashPro > Choose STAPL** Template menu option.

Note that separate template STAPL files are required when targeting A2F200 or A2F500 devices.

MPM for SmartFusion Reference Design I²C

I²C Operation

Overview

SmartFusion MPM Reference Design uses a standard 2-wire I²C slave interface, which is by default implemented using the SmartFusion microcontroller subsystem (MSS) I2C_1 interface for compatibility with the SmartFusion Development Kit Board and SmartFusion Evaluation Kit Boards¹. 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 utilizing 7-bit addressing format at effective 100 kbps and 400 kbps data rates². For more information on I2C, please consult the [Philips/NXP I2C specification document](#).

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

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

Board

I²C access from an I²C master to the MPM I²C slave is provided via the MSS I2C_1 SCL and SDA pins on the SmartFusion Development Kit Board and SmartFusion Evaluation Kit Boards via the pins in the Table 4. The MPM GUI can act as an I²C master communicating with the MPM I²C slave via the Devantech/Robot Electronics USB-ISS USB to I²C dongle and the corresponding pins on this dongle are also listed.

Table 4 · Devantech/Robot Electronics USB-ISS Connections

I ² C Signal Board	SCL	SDA	GND
Devantech/Robot Electronics USB-ISS http://www.robot-electronics.co.uk/htm/usb_iss_tech.htm	SCL (I/O 3)	SDA (I/O 4)	0V/Ground
A2F500-DEV-KIT Rev A www.microsemi.com/soc/products/hardware/devkits_boards/smartfusion_dev.aspx	J7 pin 6	J7 pin 10	J13 pin 2 or 10
A2F[200]-EVAL-KIT Rev 2 www.microsemi.com/soc/products/hardware/devkits_boards/smartfusion_eval.aspx	JP9 pin 2	JP9 pin 1	JP9 pin 3

1. Specifically the SmartFusion Development Kit Board A2F500-DEV-KIT Rev A and the SmartFusion Evaluation Kit Board A2F[200]-EVAL-KIT Rev 2. Note A2F[200]-EVAL-KIT boards prior to Rev 2 do not expose the MSS I2C_1 signals
2. Note that effective data rates may be lower due to clock stretching. Also note that the I²C clock is configurable to other clock rates as well and is dependent on the I²C input clock(s).



The SmartFusion cSoC boards' MSS I²C SCL and SDA signals have 10 K Ohms pull-up resistors and have an operating voltage of 3.3 V³. For more detailed board specifications, consult the SmartFusion Development/Evaluation Kit Board and Devantech/Robot Electronics USB-ISS documentation.

GUI

As in previous releases of MPM the GUI can program the MPM design or just MPM configuration data to the target by calling out to the FlashPro software and passing the relevant "template" STAPL file which it fills in with the user specified MPM configuration settings.

In addition, once MPM has initially been programmed to the target, the GUI can optionally connect via the Devantech/Robot Electronics USB-ISS USB to I2C dongle to the MPM I²C slave on MSS I2C_1 in order to reprogram, control, and monitor MPM.

I²C Protocol

This section describes the application-level protocol; that is, the meaning of specific transmitted and received bytes from / to the SmartFusion 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 in Figure 23.

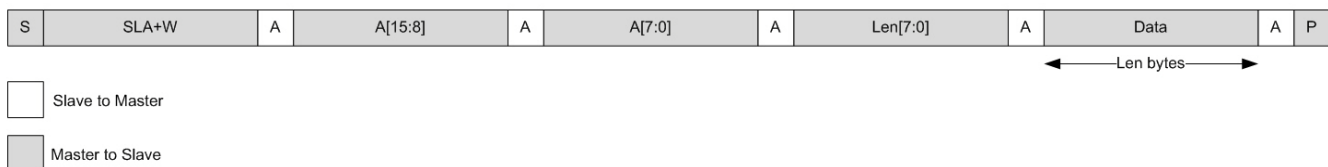


Figure 23 · I²C Block Write

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 38 for specific addresses). The first data byte is written to the above address, and all subsequent data is written to incremental addresses.
- Len [7:0] is the number of data bytes that follow; 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 in Figure 24.

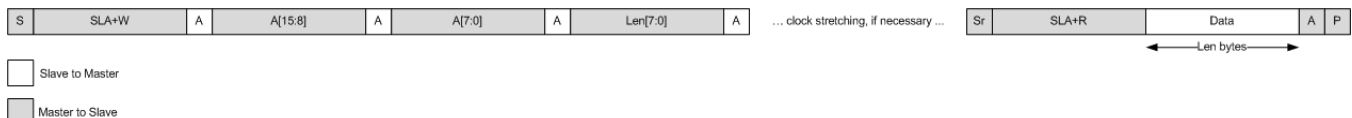


Figure 24 · I²C Block Read

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

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 38 for specific addresses). The first data byte is read from the above address, and all subsequent data is read from incremental addresses.
- Len [7:0] is the number of data bytes that follow; valid range is 1-64.
- Clock stretching⁴ 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 38 for specific addresses) are used to issue real-time commands to MPM, as seen in Figure 25.

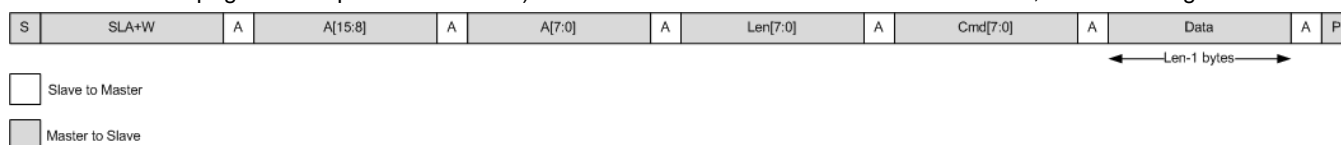


Figure 25 · I²C Command

Note:

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

I²C Status

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

Types of Access

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. Configuration registers changes do not take effect immediately but only when MPM is next reinitialized/restarted.

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. (for example attempting to stop MPM when it has not been started). Command register writes map to runtime calls inside the MPM engine to code that implements the relevant command functionality.

4. For more information on I2C clock stretching, refer to Section 3.9 of the I2C Rev 03 specification.

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 0x1900 reads the 16 bit signed mV values for 64 of the MPM rails/channels and returns them in order of MPM rail/channel number (1..64)⁵. 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 and will take effect next time MPM is reinitialized/restarted.

I²C Register Map

MPM Memory Map Overview

Table 5 · Memory Map Overview

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

5. Contiguous reads may not be possible across all register sub-blocks.

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

The default can be overridden by adding the following to the SoftConsole MPM firmware project properties:

`-DMPM_CONFIGURATION_DATA_BASE_ADDRESS=<some-eNVM-address>`

2. In the MPM Libero project's MSS configuration where the "MPM_Register_Map" eNVM data storage client's eNVM offset address is specified. By default this is set to 0x0003D000 which is implicitly with respect to the eNVM base address of 0x60000000 in the Cortex-M3 processor 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 the user through GUI and writing a new STAPL file before invoking FlashPro to write this STAPL file to the target. Alternatively the GUI can connect to the MPM I2C slave in order to read/write MPM configuration registers for the purposes of configuration, control and monitoring.

The MPM register base address must be a 128 byte eNVM page aligned address (i.e. 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 – e.g. due to reading garbage configuration data.

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

Register Addresses

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

Register Map Overview

The Table 6 summarizes the SF MPM register map. Each register is described in further detail in subsequent sections of this document.

Table 6 · Register Map Overview

Configuration Registers - 'real' registers that use eNVM storage	
Per channel/rail configuration registers - 64 (0x40) bytes for each channel/rail	
Address	Description
0x0000	Rail A1 configuration registers
0x0040	Rail A2 configuration registers
.	

.	
0x0F80	Rail A63 configuration registers
0x0FC0	Rail A64 configuration registers
Per-trigger/flag output configuration registers – 32 (0x20) bytes for each output	
0x1000	Output 1 configuration registers
0x1020	Output 2 configuration registers
.	
.	
0x17C0	Output 63 configuration registers
0x17E0	Output 64 configuration registers
Miscellaneous configuration registers	
Address	Description
0x1800	Miscellaneous configuration registers
I²C monitoring and status registers	
Rail voltage monitoring and RTC registers – “pseudo” registers that do not occupy persistent eNVM memory storage.	
Address	Description
0x1900..0x197F	Rail voltage monitoring registers
0x1980..0x1985	I ² C status, log status and RTC registers
I²C command registers	
Address	Description
0x8000	MPM I ² C command register(s) – also “pseudo” register(s)

Per-rail Configuration Data

The per-channel/rail configuration data ‘sub-block’ base address for channel n ($1 \leq n \leq 64$) is $(0x0000 + ((n - 1) \times 0x40))$. The Table 7 details the per-channel/rail registers and their offsets with respect to this per-channel/rail configuration data ‘sub-block’ base address.

Table 7 · Pre-rail Configuration Data

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

SDONH	0x0011	R/W	SDON[15:8]	relevant sequencing slot. 16 bit unsigned time value in milliseconds.
SDOFFL	0x0012	R/W	SDOFF[7:0]	SDOFF[15:0] - per channel power off sequencing delay with respect to start of relevant sequencing slot. 16 bit unsigned time value in milliseconds.
SDOFFH	0x0013	R/W	SDOFF[15:8]	
TRCFG	0x0014	R/W	Trimming configuration Bits [7:3] reserved Bits [1:0] trimming mode 0 = None/disabled (default) 1 = Open Loop 2 = Closed Loop Bit [2] trimming control DAC type 0 = SmartFusion MSS OBD 1 = CorePWM ⁶	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	0x0015	N/A	Unused/reserved	-
VNOML	0x0016	R/W	VNOM[7:0]	16 bit signed mV value where UV1 < VNOM < OV1 (positive voltage channel) or OV1 < VNOM < UV1 (negative voltage channel) and TRILO < VNOM < TRIHI for a trimming enabled channel. Used for closed loop trimming to requested nominal output voltage.
VNOMH	0x0017	R/W	VNOM[15:8]	
TRSDLL	0x0018	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.
TRSDHL	0x0019	R/W	TRDEL[15:8]	
DACOUT_NOML	0x001A	R/W	DACOUT_NOM[7:0]	For open loop trimming DACOUT_NOM is

DACOUT_NOMH	0x001B	R/W	DACOUT_NOM[15:8]	a 16 bit value between 0 and 3300 mV 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_HIL	0x001C	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	0x001D	R/W	DACOUT_HI[15:8]	
DACOUT_LOL	0x001E	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	0x001F	R/W	DACOUT_LO[15:8]	
TRIHIL	0x0020	R/W	TRIH[7:0]	Closed loop trimming high output voltage target. Normally within 10% of VNOM. 16 bit signed mV value.
TRIHIL	0x0021	R/W	TRIH[15:8]	
TRILOL	0x0022	R/W	TRILO[7:0]	Closed loop trimming low output voltage target. Normally within 10% of VNOM. 16 bit signed mV value.
TRILOH	0x0023	R/W	TRILO[15:8]	
DPOL_I2C_BUSL	0x0024	R/W	DPOL_I2C_BUS[7:0]	MPM I ² C bus identifier – currently ignored.
DPOL_I2C_BUSH	0x0025	R/W	DPOL_I2C_BUS[15:8]	
DPOL_I2C_ADDRL	0x0026	R/W	DPOL_I2C_ADDR[7:0]	DPOL only – PMBus I ² C slave address. Only bits [6:0] are significant and store the I ² C address in "0x7F" mode which is shifted left one bit internally to make room for I ² C R/W/GCA bit.
DPOL_I2C_ADDRH	0x0027	R/W	DPOL_I2C_ADDR[15:8]	
Unused	0x0028 - 0x003F	N/A	Unused/reserved	-

Per-output Configuration Data, Outputs 1-32

The per-trigger/flag output configuration data 'sub-block' base address for outputs ($1 \leq n \leq 32$) is $(0x1000 + ((n-1) \times 0x20))$.

The Table 8 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 8 · Pre-output Configuration Data

Name	Offset	Access	Description
R1_R2_STATE	0x0000	R/W	<p>Rail A1/A2 states used to generate this flag.</p> <p>Bits[7:4] Rail A2 state</p> <p>Bits[3:0] Rail A1 state</p> <p>0 = rail does not impact flag</p> <p>1 = OV2</p> <p>2 = OV1</p> <p>3 = UV1</p> <p>4 = UV2</p> <p>5 = Nominal ($UV1 < \text{nominal} < OV1$)</p> <p>6 = OV1 or UV1</p> <p>7 = OV1 or UV2</p> <p>8 = OV2 or OV1</p> <p>9 = OV2 or UV2</p> <p>10 = OFF</p> <p>11 = Not OFF</p> <p><i>Note: The following registers use the same encoding.</i></p>
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.
FLAG_CFG	0x0010	R/W	<p>Other flag configuration</p> <p>Bit[7:6] Output logging during power sequencing</p> <p>0 = None</p> <p>1 = During power on sequencing</p> <p>2 = During power down off sequencing</p> <p>3 = During power on and power off sequencing</p> <p>Bit[5] During power sequencing</p> <p>0 = Hold/don't update flag</p> <p>1 = Track/update flag</p> <p>Bit[4] combine rail states using:</p> <p>0 = OR</p> <p>1 = AND</p> <p>Bit[3] polarity</p> <p>0 = asserted high</p> <p>1 = asserted low</p> <p>Bits[2:0] combine digital input n using:</p> <p>0 = Ignore</p> <p>1 = OR</p> <p>2 = AND</p> <p>3 = XOR</p> <p>4 = OR (NOT input)</p> <p>5 = AND (NOT input)</p>
FLAG_LOGGING	0x0011	R/W	<p>Bits[2:0] Log 0 to 1 transitions</p> <p>0 = Never</p> <p>1..7 = Always</p> <p>Bits[5:3] Log 1 to 0 transitions</p> <p>0 = Never</p> <p>1..7 = Always</p>
Unused	0x0012-0x001F	N/A	Unused/reserved

Per-output Configuration Data, Outputs 33-64

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

Table 9 · Pre-output Configuration Data

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

R59_R60_STATE	0x000D	R/W	Rail A59/A60 states used to generate this flag.
R61_R62_STATE	0x000E	R/W	Rail A61/A62 states used to generate this flag.
R63_R64_STATE	0x000F	R/W	Rail A63/A64 states used to generate this flag.
FLAG_CFG	0x0010	R/W	<p>Other flag configuration</p> <p>Bit[7:6] Output logging during power sequencing</p> <p>0 = None</p> <p>1 = During power on sequencing</p> <p>2 = During power down off sequencing</p> <p>3 = During power on and power off sequencing</p> <p>Bit[5] During power sequencing</p> <p>0 = Hold/don't update flag</p> <p>1 = Track/update flag</p> <p>Bit[4] combine rail states using:</p> <p>0 = OR</p> <p>1 = AND</p> <p>Bit[3] polarity</p> <p>0 = asserted high</p> <p>1 = asserted low</p> <p>Bits[2:0] combine digital input n using:</p> <p>0 = Ignore</p> <p>1 = OR</p> <p>2 = AND</p> <p>3 = XOR</p> <p>4 = OR (NOT input)</p> <p>5 = AND (NOT input)</p>
FLAG_LOGGING	0x0011	R/W	<p>Bits[2:0] Log 0 to 1 transitions</p> <p>0 = Never</p> <p>1..7 = Always</p> <p>Bits[5:3] Log 1 to 0 transitions</p> <p>0 = Never</p> <p>1..7 = Always</p>
Unused	0x0012-0x001F	N/A	Unused/reserved

Miscellaneous/Global Configuration Data

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

Table 10 - Miscellaneous/Global Configuration Data

Name	Address	Access	Description	
PWR_ON_CTRL	0x1800	R/W	Power on sequence control. Bits[7:3] reserved Bits[2:0] power up sequence failure action 0 = hold 1 = shutdown 2 = restart slot 3 = restart sequence 4 = skip slot	-
Unused	0x1801	N/A	Unused/reserved	-
SLOT_TIMEOUT_L	0x1802	R/W	SLOT_TIMEOUT[7:0]	SLOT_TIMEOUT [15:0] - power on sequencing slot timeout. 16 bit unsigned time value in milliseconds.
SLOT_TIMEOUT_H	0x1803	R/W	SLOT_TIMEOUT[15:8]	
PWR_OFF_CTRL	0x1804	R/W	Power off sequence control. Bits[7:3] reserved Bit[2] post power off action (currently unused by SF MPM) 0 = stay off 1 = allow restart Bits[1:0] power off sequence 0 = reverse (slot n, n-1,...,1) 1 = forward (slot 1, 2,...,n) 2 = simultaneous (as if all channels in a single slot but off delays still accounted for)	Bit[2] is deprecated in SF MPM v3.0
I2C_SLAVE_ADDR	0x1805	R/W	Bits[6:0] I ² C slave address	MPM I ² C slave address

LOGGING	0x1806	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.
CLEAR_LOG	0x1807	R/W	Bit[0] 0 = don't clear log 1 = clear log	Clear log next time MPM reinitializes. A 0x00 "end of file" marker byte is written at the start of the log file so that the next time a log record is written to the log file it is written at the very beginning of the log file. Note that only the first byte of the log file is written (to 0x00) when clearing the log. Once the log file has been cleared MPM also clears the CLEAR_LOG register so that the log is not automatically cleared again next time MPM initializes.
Unused	0x1808	N/A	Unused/reserved	-

Rail Voltage Monitoring Registers

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

Table 11 · Rail Voltage Monitoring Registers

Name	Address	Access	Description	
RAIL1_VH	0x1900	R/O	RAIL1_V[15:8]	16 bit signed value representing the instantaneous output voltage for rail A1 in mV.
RAIL1_VL	0x1901	R/O	RAIL1_V[7:0]	
RAIL2_VH	0x1902	R/O	RAIL2_V[15:8]	16 bit signed value representing the instantaneous output voltage for rail A2 in mV.
RAIL2_VL	0x1903	R/O	RAIL2_V[7:0]	
.				
.				

RAIL63_VH	0x197C	R/O	RAIL63_V[15:8]	16 bit signed value representing the instantaneous output voltage for rail A63 in mV.
RAIL63_VL	0x197D	R/O	RAIL63_V[7:0]	
RAIL64_VH	0x197E	R/O	RAIL64_V[15:8]	16 bit signed value representing the instantaneous output voltage for rail A64 in mV.
RAIL64_VL	0x197F	R/O	RAIL64_V[7:0]	
I2C_STATUS	0x1980	R/C	Status of most recent I ² C 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 I ² C interactions. Indicates the status of the most recent I ² C protocol interaction.
RTC4	0x1981	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 I ² C in one operation since reading individual bytes separately and recombining them will most likely yield an incorrect time value.
RTC3	0x1982	R/O	RTC byte 4 – i.e. bits[31:24]	-
RTC2	0x1983	R/O	RTC byte 4 – i.e. bits[23:16]	-
RTC1	0x1984	R/O	RTC byte 4 – i.e. bits[15:8]	-
RTC0	0x1985	R/O	RTC byte 4 – i.e. bits[7:0]	-

Unused	0x1986-0x19FF	N/A	Unused/reserved	-
--------	---------------	-----	-----------------	---

I²C Command Registers

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

Table 12 · I²C Command Registers

Name	Address	Access	Description
I2C_CMD_0	0x8000	W/O	I ² C command pseudo register. See below for details about how this is used.
Unused	0x8001-0xFFFF	N/A	Unused/reserved

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

Table 13 describes the opcodes defined for command register 0x8000.

Table 13 · Command Register

Name	Address	Access	Description
0x00	Unused/reserved	N/A	N/A
0x01	START	None	Initiate power on sequencing. Maps to the MPM driver API <code>mpm_start()</code> . Ignored if MPM is not in stopped state.
0x02	STOP	None	Initiate power off sequencing. Maps to the MPM driver API <code>mpm_stop()</code> . Ignored if MPM is not in started state.
0x03	MARGIN_LOW	One eight bit byte containing the channel to be trimmed/margined (1..64).	Set margin level to MARGIN_LOW for the specified channel (1..64). For Analog Point of Load (APOL) channels: In open-loop mode, this sets the trim pin voltage to DACOUT_LO. In closed-loop mode, the trim pin voltage is continually adjusted until the output voltage reaches TRLO. For Digital Point of Load (DPOL) channels: Uses PMBus OPERATION (0x01) command to program the DPOL to output the margin low voltage. Ignored if MPM is not in started state.

0x04	MARGIN_NOM	One eight bit byte containing the channel to be trimmed/margined (1..64)	<p>Set margin level to NOMINAL (default) for the specified channel (1..64).</p> <p>For Analog Point of Load (APOL) channels:</p> <p>In open-loop mode, this sets the trim pin voltage to DACOUT_NOM.</p> <p>In closed-loop mode, the trim pin voltage is continually adjusted until the output voltage reaches VNOM.</p> <p>For Digital Point of Load (DPOL) channels:</p> <p>Uses PMBus OPERATION (0x01) command to program the DPOL to output the normal nominal voltage.</p> <p>Ignored if MPM is not in started state.</p>
0x05	MARGIN_HIGH	One eight bit byte containing the channel to be trimmed/margined (1..64).	<p>Set margin level to MARGIN_HIGH for the specified channel (1..64).</p> <p>For Analog Point of Load (APOL) channels:</p> <p>In open-loop mode, this sets the trim pin voltage to DACOUT_HI.</p> <p>In closed-loop mode, the trim pin voltage is continually adjusted until the output voltage reaches TRHI.</p> <p>For Digital Point of Load (DPOL) channels:</p> <p>Uses PMBus OPERATION (0x01) command to program the DPOL to output the margin high voltage.</p> <p>Ignored if MPM is not in started state.</p>
0x06	INIT	None	<p>Initiate reinitialization of MPM. Maps to the MPM driver API mpm_init(). This causes MPM to (re)load the latest configuration settings. Any MPM configuration register changes made via I²C since the last call to mpm_init() take effect at this stage.</p> <p>Ignored if MPM is not in stopped state.</p>
0x07	LOG_INFO	None	<p>Returns basic status information about the log as follows:</p> <p>[0..3] Logging API Major number.</p> <p>[4..7] Max Records possible.</p> <p>[8..11] Max Guaranteed Records. In the event of a log roll-over the number of records that will be available.</p> <p>[12..15] Current number of entries in the log.</p> <p>[16..19] Current Sequence No.</p>
0x08	LOG_RESET	None	<p>Force a clear down of the log. All queued messages will be discarded, the log will be erased, the sequence number will be reset to 0 and a logging started entry will be placed in the log with sequence number 0.</p>

0x09	LOG_READ_LAST	None	Reads the last 4 entries from the log (i.e. the newest entries). The response read back will contain 64 bytes of data with between 1 and 4 log records returned depending on the number of records in the log (external reading should never see less than 1 record returned as we insert a log started record into the new log before allowing read access). This command also resets the internal sequential log pointer to the end of the log.
0x0A	LOG_READ_NEXT	None	Read the next 4 entries from the log (these should be the next oldest records if no new entries have been added since the last read operation). The response read back will contain 64 bytes of data with between 0 and 4 log records returned depending on the number of records in the log and our starting position. Reading less than 4 records indicates the end of the log has been reached.
0x0B	LOG_SYSTEM_INFO	None	Information about the MPM firmware running on the device: [0..19] Zero terminated MPM version number string. [20..21] Version major number. [22..23] Version minor number. [24..25] Sub version number. [26..27] Device Id. This is a code to identify the hardware platform: we use 0 for MPM running on the Smart Fusion development kit board and 1 for MPM running on the Smart Fusion evaluation kit board. Customers can assign their own numbers to allow them id different products or product revisions etc
0x0C- 0xFF	Unused/reserved	N/A	N/A

Notes

The MPM version 5 register map and tools are not backward compatible with previous versions of MPM.

SmartFusion MPM Data Logging

Data Logging

Events

The following events can be logged (configurable via the MPM GUI or I²C 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 I²C commands). These transitions can be logged. Figure 26 illustrates a state diagram of SF MPM.

Note: Can "loop" on starting/stopping state depending on timeout and on fall action config

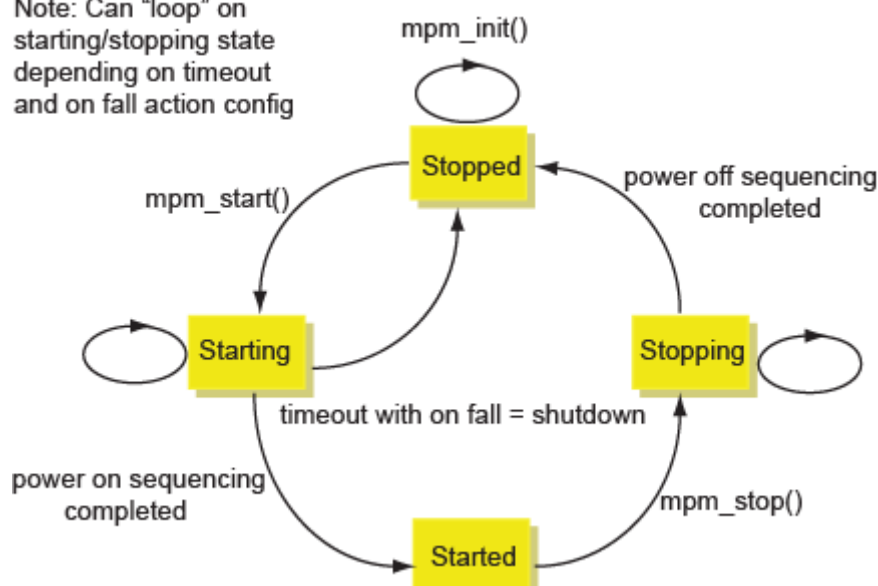


Figure 26 · MPM State Transition Diagram

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.

Figure 27 shows the GUI configuration for output/trigger logging.

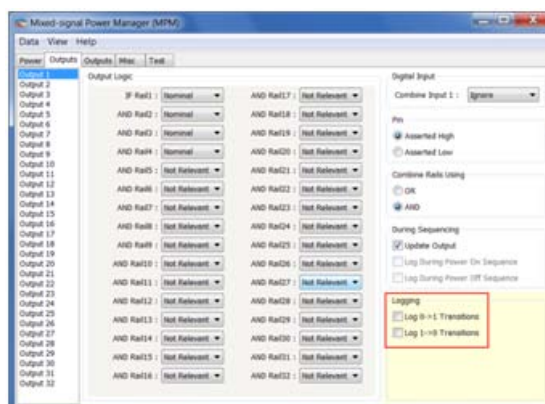


Figure 27 · GUI Configuration of Output/Trigger Data Logging

Power Sequencing Events

The user may log power-on sequencing events or power-off sequencing events. Figure 28 shows the GUI (global) configuration for power sequencing event logging.

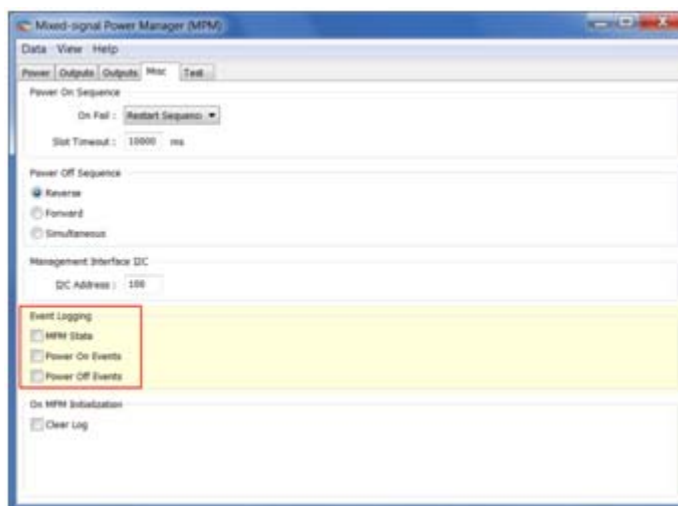


Figure 28 · Power Sequencing Events Logging

Log Record Format

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

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

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

Log Record Type Byte

Table 14 describes record types used for SF MPM data logging.

Table 14 · 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 occurred in the lower bank (flag outputs 1-32)
0x03	Output flag/trigger transition occurred in the upper bank (flag outputs 33-64)
0x04	Power on sequencing event
0x05	Power off sequencing event
0x06	Log status changed
0xFE	Extension record type
0x07 - 0xFD and 0xFF	Reserved

Log Data

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

Table 15 · Log Data

Log record byte = 0x01 (MPM state changed)	
Bit(s)	Meaning
1:0	0 = stopped
	1 = starting
	2 = started
	3 = stopping
7:2	Unused
Log record byte = 0x02 (Output flag/trigger transition occurred (outputs 1-32))	
4:0	Output number N, minus 1 (valid values are 0-31, while output numbers range from 1-32)
5	Change to new value of output - 0 or 1
7:6	Unused

Log record byte = 0x03 (Output flag/trigger transition occurred (outputs 33-64))	
4:0	Output number N, minus 33 (valid values are 0-31, while output numbers range from 33-64)
5	Change to new value of output - 0 or 1
7:6	Unused
Log record byte = 0x04/0x05 (Power on/off sequencing event)	
Byte 1[2:0]	Event Type [0 = start; 1 = finish; 2 = timeout; 3 = hold; 4 = shutdown/terminate; 5 = restart slot; 6 = restart sequence; 7 = skip slot]
Byte 1[7:3]	Unused
Byte 2[5:0]	Slot number: For "restart slot events" (configuration 5) and "skip slot events" (configuration 7), this indicates the slot number, minus 1 (valid values are 0-63, while slot numbers range from 1-64)
Byte 2[7:6]	Unused
Log record byte = 0x06 (Log status changed event)	
2:0	0 = Normal Log Startup 1 = Abnormal Log Startup 2 = Log Erased
Log record byte = 0xFE (Extension record)	
	Bytes 2..13 in this log record are available for user defined additional data rather than timestamp and sequence. Bytes 14..15 and byte 1 are reserved.

Log Timestamp

The 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 128¹ 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 0xFFFFFFFF * 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 cSoC 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 0xFFFFFFFF to 0x00000000.

The log file is stored in the Smart Fusion Evaluation kit 8 MB external FLASH memory accessed via SPI.

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

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

Log Access Via I²C

The commands supported to access the log via I²C are described in Table 14. These are the I²C commands with command codes 0x07 (LOG_INFO) through to 0x0B (LOG_SYSTEM_INFO).

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

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

eNVM Logging

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

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

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 29.

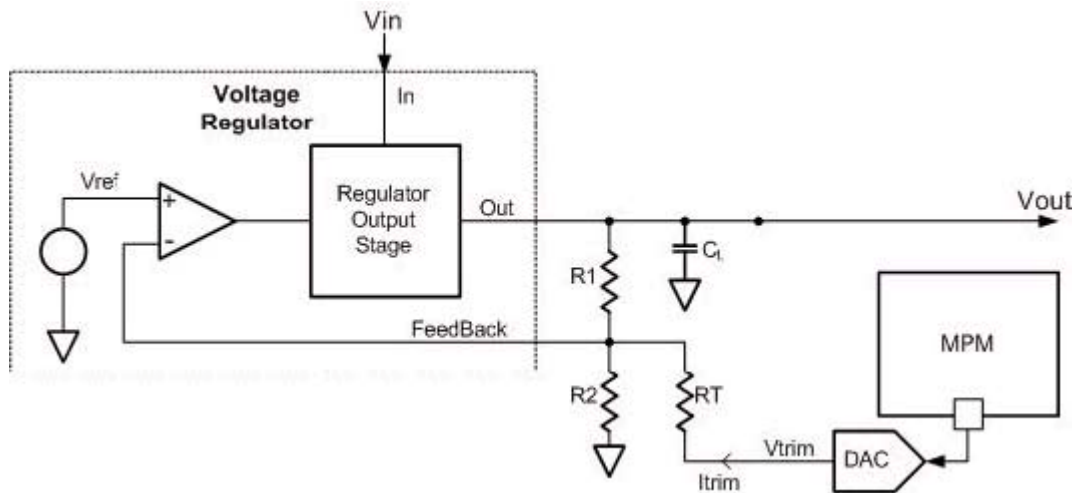


Figure 29 · Open-Loop Trimming Using SmartFusion MPM

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 30.

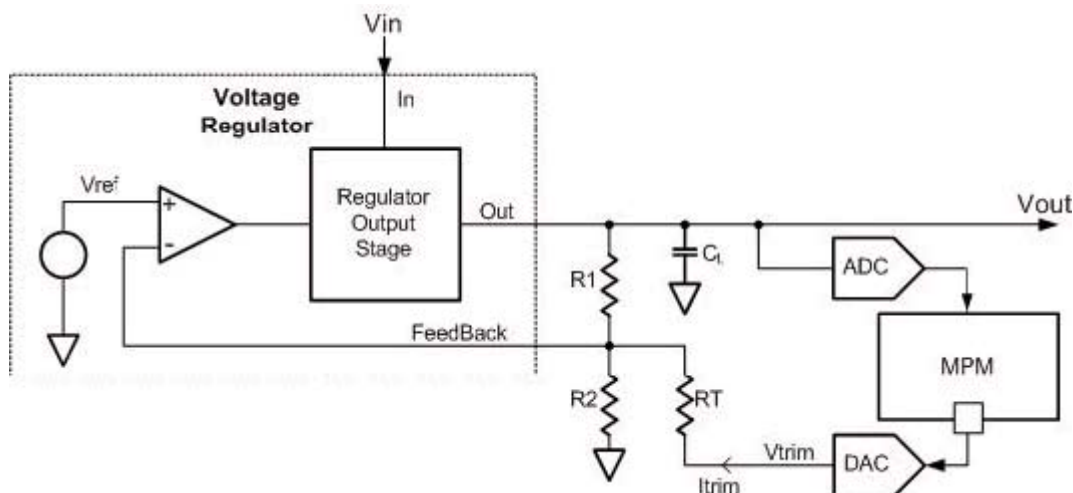


Figure 30 · Closed-Loop Trimming Using SmartFusion MPM

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 $V_{outtarget}$
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 "Trim Type" section on page 61 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 R_T and R_2 resistances (R_1 is typically internal to regulators) changes the regulator's sensitivity to variations on the R_{trim} pin. It is your responsibility to ensure that R_T is large enough so that V_{trim} variations will not put the regulator out of its operating range (typically only small variations on the feedback pin are allowed, on the order of mV's), and not so large as to make V_{trim} voltages changes negligible to the system.

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

GUI Operation

The sections of the GUI highlighted in Figure 31 define configuration parameters used for trimming on a per-channel basis.

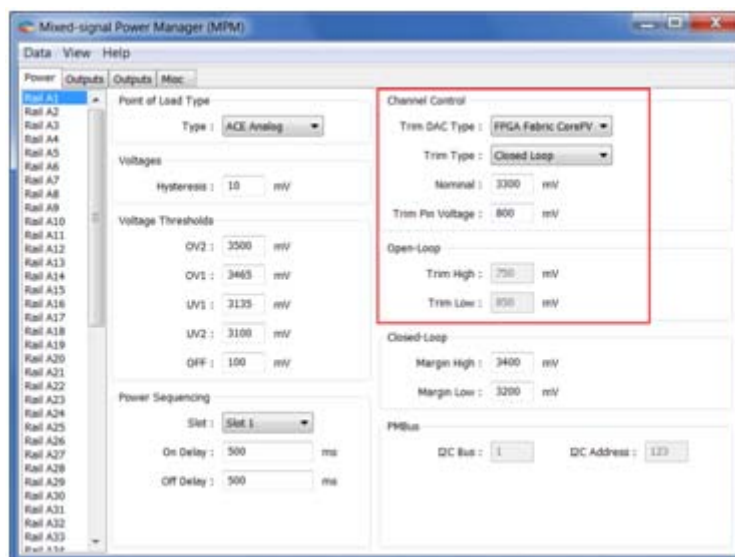


Figure 31 · Trimming GUI Section

Channel Control

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

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

Trim Type

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

Disabled

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

Open-Loop

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

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

Closed-Loop

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

However, after all rails have been powered up, the trim pin voltage is adjusted according to the scheme described in the "Closed-Loop Trimming" section on page 60. The target voltage ($V_{outtarget}$) 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 by 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.

Trim DAC Type

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

MSS ACE SDD

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 CorePWM

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

When using the DMPM-DC the MPM GUI configuration settings for trimming of Channel A1 and Channel A2 must be matched by the appropriate configuration of DMPM-DC trimming jumpers.

Channel A1

- Trim Type = Disabled
 - ✓ No jumper on JP3
- Trim Type = Open Loop or Closed Loop
 - ✓ Jumper on JP3 pins 1-2
- Trim DAC Type
 - ✓ MSS ACE SDD
 - Jumper on JP19 pins 2-3
 - ✓ FPGA Fabric CorePWM
 - Jumper on JP19 pins 1-2

Channel A2

- Trim Type = Disabled
 - ✓ No jumper on JP2
- Trim Type = Open Loop or Closed Loop
 - ✓ Jumper on JP2 pins 1-2
- Trim DAC Type
 - ✓ MSS ACE SDD
 - Jumper on JP20 pins 2-3
 - ✓ FPGA Fabric CorePWM
 - Jumper on JP20 pins 1-2

Note: If Channel A2 is configured for MSS ACE SDD trimming but Channel A1 is not then it is necessary to remove the jumper from JP20 2-3 and install a jumper patch cable from JP19 pin 3 to JP20 pin 2. This is because MSS ACE SDDs for trimming are allocated by MPM on a first come first served basis by MPM Channel number and the DMPM-DC is designed to expect SDD0 is used to trim Channel 1 and SDD1 is used to trim Channel 2 so using SDD0 to trim Channel requires the additional jumper cable.

MPM Daughter Card Hardware Guide

Introduction

The Digital Mixed Signal Power Manager (DMPM) Daughter Card (DC) enables system designers to evaluate and demonstrate the functionality of Microsemi's power management solutions in hardware, using a five regulator benchtop power management development system. The daughter card includes two analog regulators with adjustable bias voltage POTs, three digital regulators residing on a connectable PMBus (I²C) bus, five push buttons, and a series of LED's that may be used to demonstrate and explore MPM digital output functionality. The Figure 32 shows the MPM daughter card with its major components.

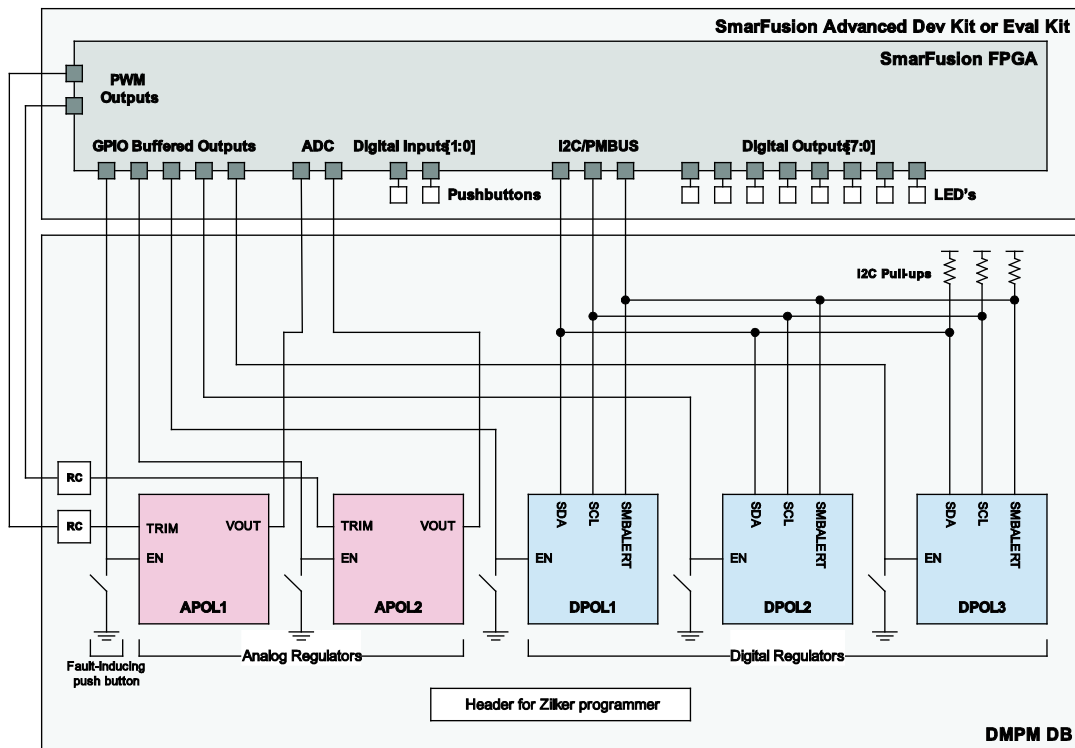


Figure 32 · MPM Daughter Card

- Note:** It is possible to leave DPOL2 and DPOL3 unconnected from PMBus / I2C bus using jumper settings.
- Note:** "RC" above, denotes an RC network (low pass filter), which acts as a DAC when connected to the PWM output from MPM.
- Note:** GPIO Buffered outputs only output "tristate" or "low". This is to prevent short-circuiting the FPGA I/O when a fault-inducing push-button is pressed. All enable pins are pulled "high", so that "tristate" corresponds to active.
- Note:** Not shown are additional LED's present on the DMPM-DB itself.
- Note:** The Header can only be used to program DPOL1 and DPOL2 (using the Zilker ZLUSB USB programmer), which are Intersil regulators. DPOL3 must be programmed via the PMBus.

Kit Contents

Table 16 · MPM Daughter Card Kit Contents

Quantity	Description
1	DMPM Daughter Card
1	9 V power supply
1	Enhanced USB I ² C Module
1	USB Cable A-B 3'
3	Jumper wire female-female 4"
1	DMPM-DC-Kit Quickstart Card

Related Information

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

Board Description

The DMPM Daughter Card provides a benchtop demonstration and development platform for Microsemi's MPM reference design, specifically for version 4.0 of the design – which incorporates digital regulators (controlled and monitored via PMBus) in addition to the existing analog regulators from version 3.0 of the reference design.

Figure 33 is the DMPM Daughter Card connected to the A2F-EVAL-KIT (SmartFusion Evaluation Kit Board) via their respective Mixed Signal Headers. The DMPM Daughter Card may also be connected the A2F-DEV-KIT (SmartFusion Development Kit Board).

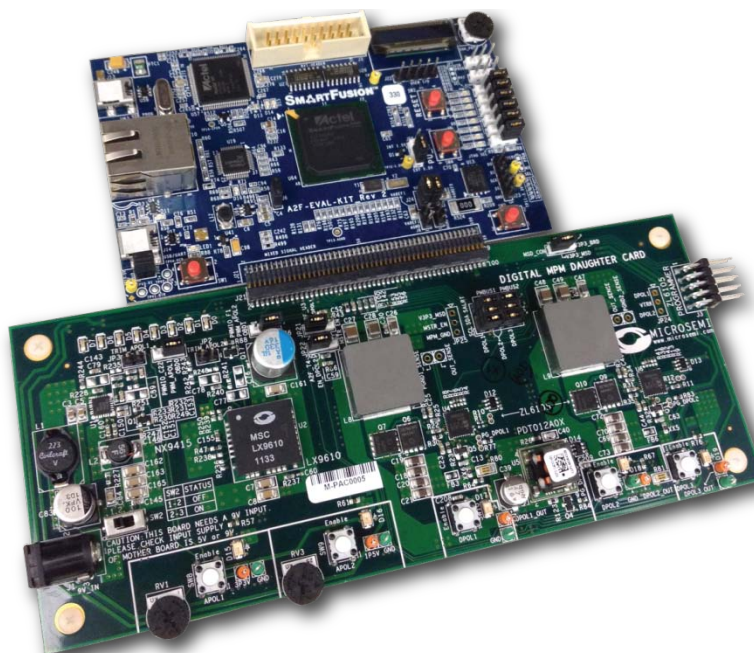


Figure 33 · MPM Daughter Card with SmartFusion Evaluation Kit Board

Digital Mixed Signal Power Manager Board Components

Table 17 · DMPM-DB Components

Serial Number	Name	Description
1	APOL1	3.3 V regulated supply, Microsemi Analog Mixed Signal Group NX9415
2	APOL2	1.5 V regulated supply, Microsemi Analog Mixed Signal Group LX9610
3	DPOL1	3.3 V digital regulated supply, Intersil ZL6105
4	DPOL2	1.5 V digital regulated supply, Intersil ZL6105
5	DPOL3	1.5 V digital regulated supply, Lineage Power PDT012A0X
6	SW8, SW9, SW10, SW11, SW12	Switch to enable/disable voltage regulators (1 to 5) to demonstrate regulator failure.
7	J3	ZL6105 programming header, used in conjunction with the Zilker programmer (ZLUSB EVAL1Z).
8	J21	Mixed Signal Header, used to connect the DMPM-DB to the A2F-EVAL-KIT or the A2F-DEV-KIT

Note: DPOL1, DPOL2, and DPOL3 maybe be programmed to output different voltages.

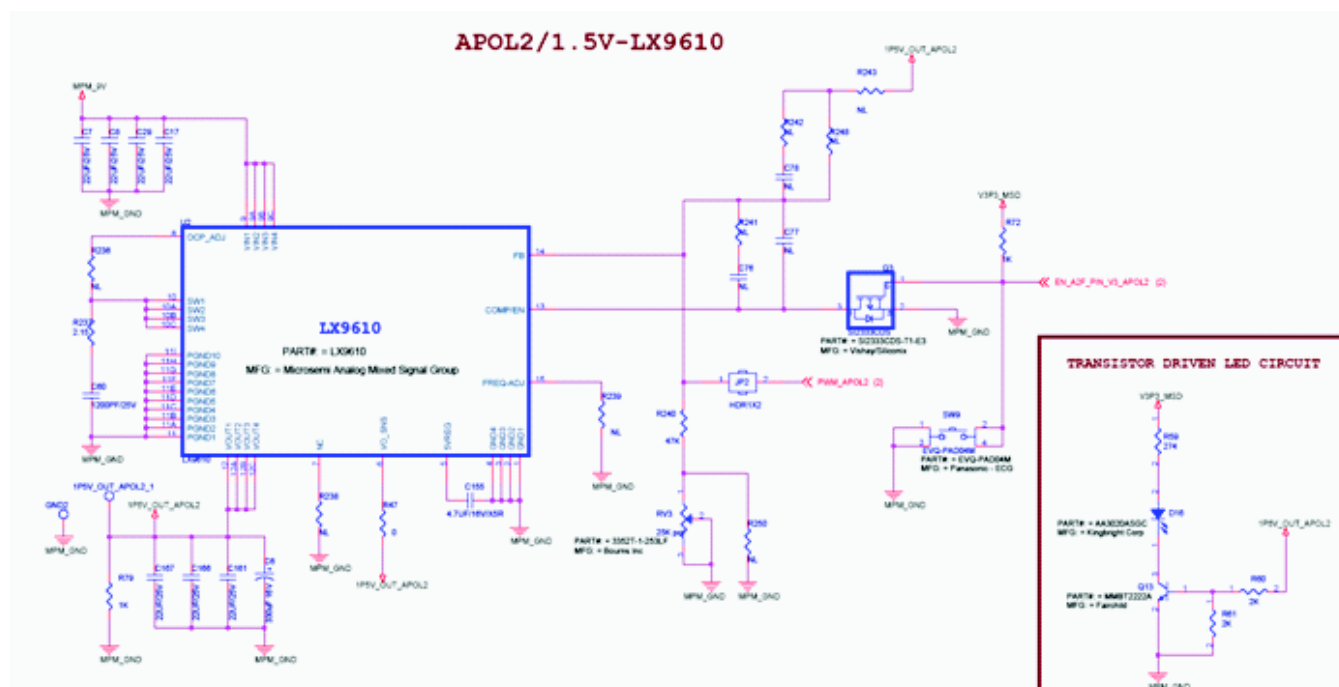


Figure 35 · APOL2 1.5 V Switching Power Supply

DPOL1/DPOL2, 3.3 V/1.5 V Intersil ZL6105 Digital Power Supply

Both DPOL1 and DPOL2 are Intersil ZL6105 digital power controllers. The ZL6105 uses the I2C/PMBus 3-wire communication bus to connect to the host controller (in this case, MPM via the Mixed Signal Header on the DMPM-DB). Both devices can be programmed to output voltages of 0.54 V to 5.5 V with currents of up to 3A. The user may set the output voltage via the Zilker Labs Power Navigator. For more information, please refer to the Intersil website and the ZL6105 datasheet.

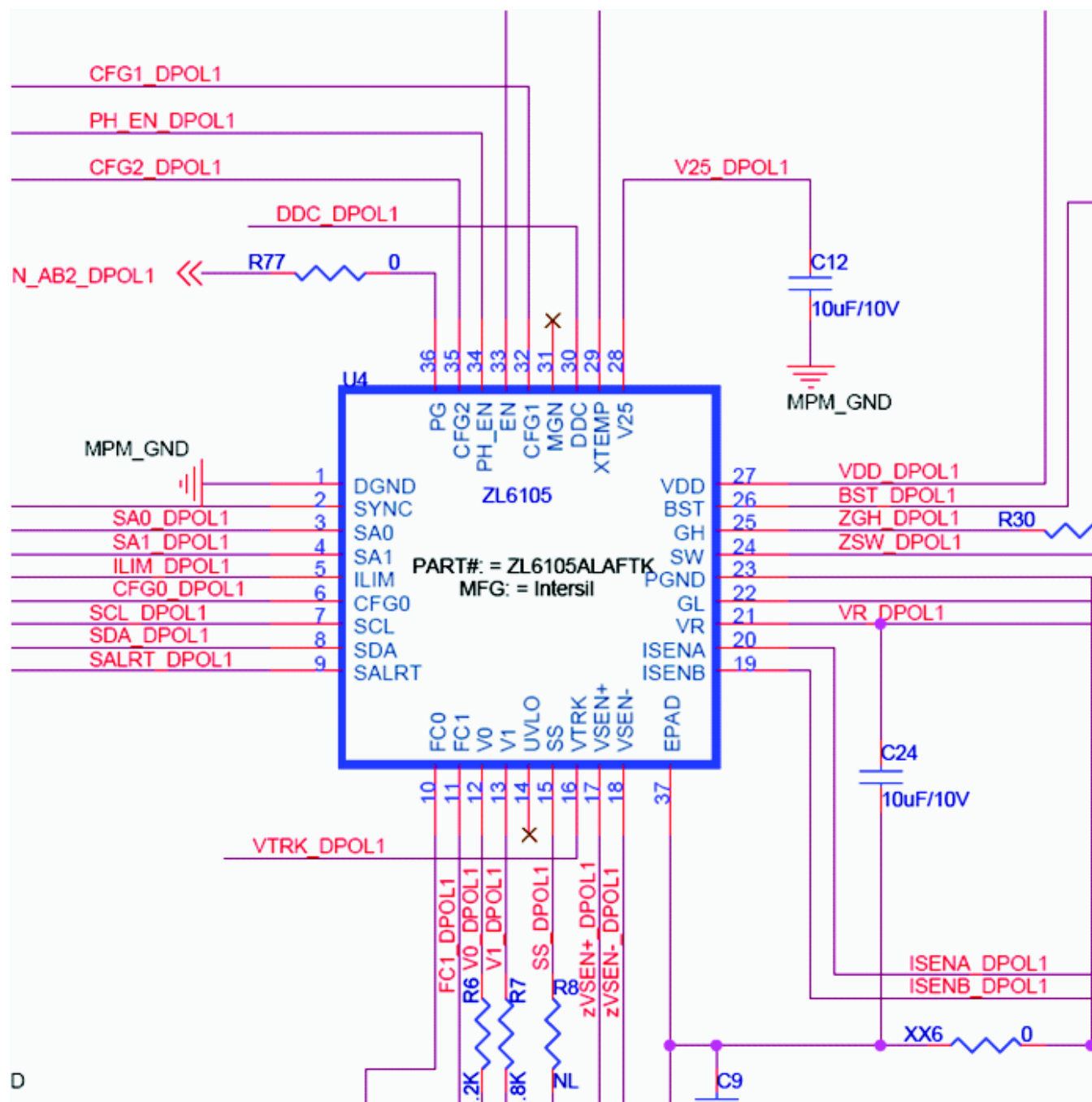


Figure 36 · DPOL1/2 ZL6015 Regulator

DPOL3 2.5 V Lineage PDT012A0X

The PDT0123A0X power module is a non-isolated DC-DC converter that can deliver up to 12 A of current, and can provide an output voltage ranging from 0.6 Vdc to 5.5 Vdc, programmable via either a bootstrap resistor or via PMBus configuration. DPOL3 also resides on PMBus2, and can be accessed by MPM via the Mixed Signal Header if PMBus2 is connected PMBus1 (see default jumper settings below).

DPOL3/2.5V-PDT012A0X

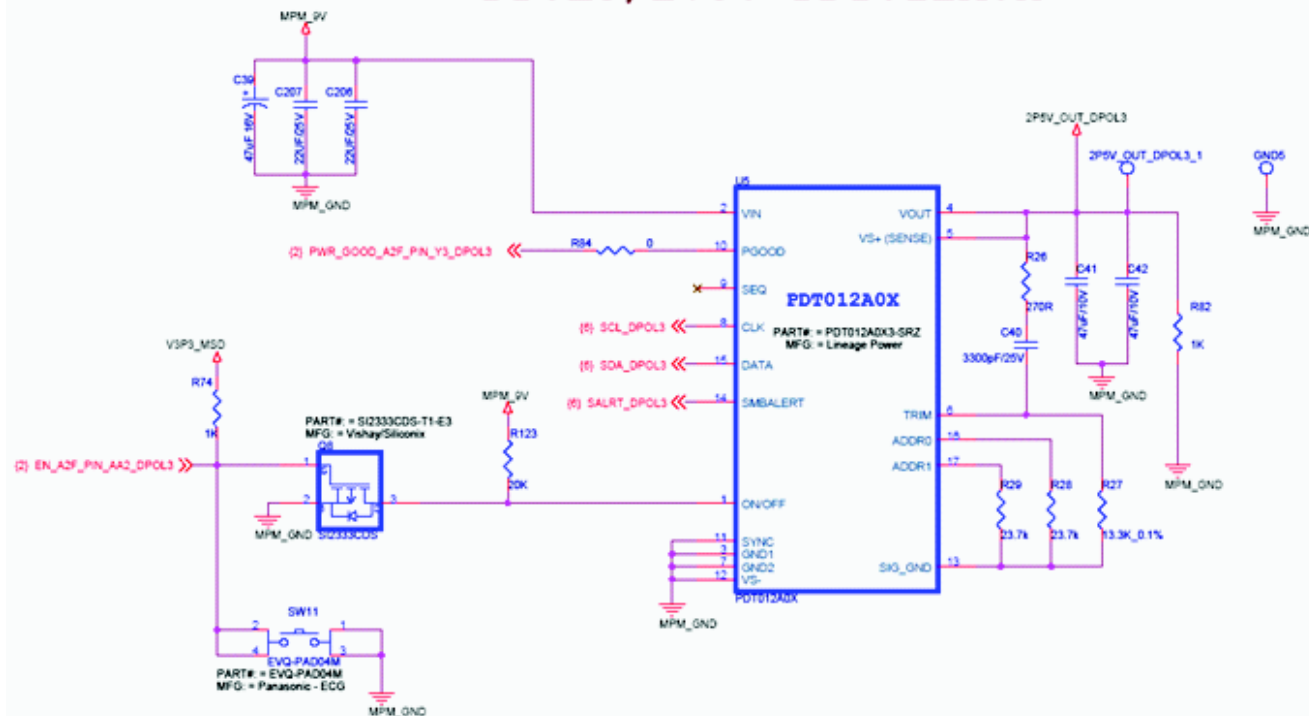


Figure 37 · DPOL3 PDT012A0X

Power

The DMPM-DB kit comes with a 9 V power supply, which connects to the supply header (J1) on the left side of the board. Make sure to use the 9 V supply for the DMPM-DB and not the A2F-DEV-KIT, which requires a 5 V power supply.

Mixed Signal Header

This connector is used to interface with the SmartFusion cSoC board (A2F-EVAL-KIT or A2F-DEV-KIT).



Recommended default jumpers settings are show in Table 18. Connect jumpers in the default settings to enable the pre-programmed reference design to function correctly. Note that the user must program the Digital Points of Load (DPOL) regulators to the default output voltages and settings defined in "Digital Mixed Signal Power Manager Board Components" section, and in the MPM GUI's default settings.

Table 18 · Jumper Settings (DMPM-DB)

Jumper	Function	Default Setting
J2	Connects DPOL2 and DPOL3 to PMBus (PMBus1 to PMBus2)	All DPOL's connected to bus, pins shorted: 1-2, 3-4, 6-7, 8-9, 11-12, 13-14
JP2	Connects APOL1 trim pin for closed-loop trimming	Closed
JP3	Connects APOL2 trim pin for closed-loop trimming	Closed
JP22	Connects DPOL1 enable pin to FPGA or to master enable	Pins 1-2 shorted
JP21	Connects DPOL2 enable pin to FPGA or to master enable	Pins 1-2 shorted
JP24	Connects VTRK pin of DPOL's to J3 header	Open
JP25	Sets master enable for DPOL's high or low	Open

Table 19 · Switches (DMPM-DB)

Switch	Description
SW8	Push-button to disable APOL1
SW9	Push-button to disable APOL2
SW10	Push-button to disable DPOL1
SW11	Push-button to disable DPOL2
SW12	Push-button to disable DPOL3
SW2	Switch ON 9 V DC into MPM

Note: The push-buttons SW8, SW9, SW10, and SW11, SW12 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. The FPGA could be damaged if it is driving HIGH on the enable line and simultaneously shorted to ground. To avoid this, Microsemi recommends that the FPGA pin driving these enables should be either driving Low or Tristate. Please refer to the Libero design for details on how to do this.

Manufacturing Information

MPM DB Board

The full PCB design layout is provided on the MPM-DC 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.

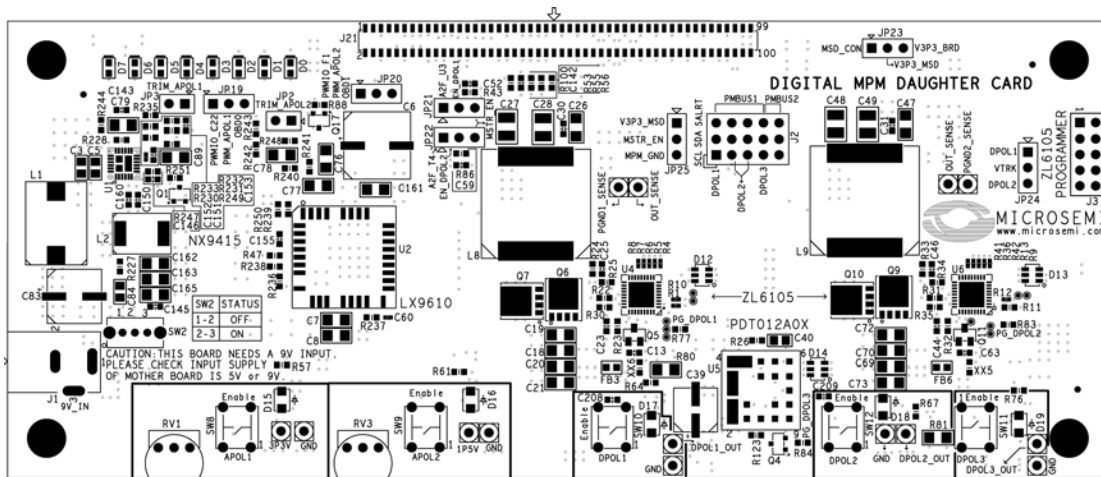


Figure 39 · Top Silk Screen for MPM Daughter Card

Product Support

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

Customer Service

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

From North America, call **800.262.1060**

From the rest of the world, call **650.318.4460**

Fax, from anywhere in the world **408.643.6913**

Customer Technical Support Center

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

Technical Support

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

Website

You can browse a variety of technical and non-technical information on the Microsemi SoC Products Group home page, at <http://www.microsemi.com/soc/>.

Contacting the Customer Technical Support Center

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

Email

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

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

My Cases

Microsemi SoC Products Group customers may submit and track technical cases online by going to [My Cases](#).

Outside the U.S.

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

ITAR Technical Support

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

List of Changes

List of Changes

The following table lists critical changes that were made in each revision of the chapter.

Date	Changes	Page
50200275-4/12.12	Modified MPM GUI Overview (SAR 41581).	22
	Modified MPM Design Files (SAR 41587 and SAR 42263).	17
	Modified MSS ENV M Data Storage Clients (SAR 41587).	32
	Modified I2C Operation (SAR 41587).	37
	Modified I2C Register Map (SAR 41587).	40
	Modified Installation and Switch Settings (SAR 41587).	72
	Modified Figure 6 , Figure 27 , Figure 28 , Figure 31 , Table 14, and Table 15 (SAR 41689 and SAR 42263).	22, 57, 57, 63
50200275-3/05.12	Modified "MPM Daughter Card Board Setup" section (SAR 38865).	7
	Modified "MPM Reference Design Demo" section (SAR 38865).	10
	Modified "Tools/Resources used" section (SAR 38865).	27
	Modified Table 3-7 (SAR SAR 38865).	47
	Modified Table 4-2 (SAR SAR 38865).	55
	Modified "Channel Control" section (SAR 38865).	61
50200275-1/04.12	Modified "Introduction" section and added Figure 1 (SAR 37497).	3
	Modified "Hardware Setup" section (SAR 37497).	5
	Modified "MPM Daughter Card Board Setup", "Programming MPM", "I2C Setup" sections (SAR 37497).	7,8,9
	Modified "Meters View" section (SAR 37497).	25
	Modified Table 6-1 and added Figure 6-2 (SAR 37497).	64
	Modified Figure A-1 (SAR 37497).	71

*The part number is located on the last page of the document. The digits following the slash indicate the month and year of publication.



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

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

© 2012 Microsemi Corporation. All rights reserved. Microsemi and the Microsemi logo are trademarks of Microsemi Corporation. All other trademarks and service marks are the property of their respective owners.