Platform Management Utility Functions IP Core User's Guide

January 2011
IPUG94_01.1

Table of Contents

Chapter 1. Introduction 5
Quick Facts 5
Features 5
Chapter 2. Functional Description 6
Closed-loop Trim Theory of Operation 7
Closed-loop Trim Engine 7
VID Control 8
Closed-loop Trim Engine Registers 9
Serial Communications Options 12
SPI Protocol 13
Data Format Options 13
System Fault Logging Theory of Operation 14
Design Details 17
CPLD Logic Description 18
Chapter 3. Parameter Settings 20
Configuration 21
Serial Communications Option 21
User Serial Port Type 21
Device Address 21
Clock Polarity 21
Clock Phase 22
Bit Order 22
Closed-loop Trim Option 22
Instantiate Closed-loop Trim Engine 22
Trim Update Rate 22
VID Channel A, B 22
VID Code Lookup Table 22
Fault Logger Option 22
Chapter 4. IP Core Generation 23
Licensing the IP Core 23
Getting Started 23
IPexpress-Created Files and Top Level Directory Structure 24
Simulation Evaluation 25
Chapter 5. Using IP Cores with PAC-Designer 26
Closed-Loop Trim with PAC-Designer 26
Fault Logging with PAC-Designer 27
Chapter 6. Support Resources 28
Lattice Technical Support 28
Online Forums 28
Telephone Support Hotline 28
E-mail Support 28
Local Support 28
Internet 28
References 28
Revision History 28
Ordering Part Number 29
Appendix A. Resource Utilization 29

Introduction

This user's guide describes the Platform Management Utility Functions IP core for the Lattice Platform Manager™ family of devices. The Platform Management Utility Functions IP core implements a configurable, multi-channel, power-supply Closed-loop Trim Engine, and a Fault Logger that can save time-stamped fault records to external SPI Flash memory. Directions for specifying the IP core's configuration, including it in a user's design, and directions for simulation and synthesis are provided in this user's guide.

Quick Facts

Table 1-1 gives quick facts about the Platform Management Utility Functions IP core.

Table 1-1. Platform Management Utility Functions IP Core Quick Facts

		Platform Management Utility Functions IP Configuration		
		Closed Loop Trim with I^2C	**Closed Loop Trim with VID (no Serial I/O)**	**Fault Logger**
Core Requirements	Device Family Supported	Platform Manager		
Resource Utilization	Targeted Devices	LPTM10-12107, LPTM10-12147		
	LUTs	525	306	344
	Slices	263	153	183
	Registers	256	147	316
Design Tool Support	Lattice Implementation	PAC-Designer® 6.0.1[1] or ispLEVER® 8.1 SP1		
	Synthesis	Synopsys® Synplify™ Pro for Lattice D-2009.12L-1		
	Simulation	Aldec® Active-HDL™ 8.2 Lattice Edition		
		Mentor Graphics® ModelSim™ SE 6.3F		

1. PAC-Designer 6.0.1 required for version 1.1 and later of design source code (containing minor enhancements to operation of IP).

Features

- Supports up to eight Closed-loop Trim Channels
- Supports up to two channels of Voltage Identification (VID)
- 16-entry VID table
- I^2C or SPI control of all key Closed-loop Trim functions
- Logging of analog input status to non-volatile memory
- Logging of multiple faults to provide history

Functional Description

The Platform Management Utility Functions IP core facilitates the implementation of three common power supply management functions in Platform Manager devices:

1. Closed-Loop trim of power supplies

2. VID control of power supplies

3. System fault logging to non-volatile memory

Table 2-1 lists the top-level port definitions for this IP core.

Table 2-1. Top-Level Port Definitions

Port	Size	I/O	Description
Common Ports (Used by Both Closed-loop Trim and Fault Logger)			
CLK_8M	1	I	8 MHz clock input (typically derived from analog section)
RESET	1	I	Global reset (active-low)
SYS_SPI_SCLK	1	I	User SPI port SCLK
SYS_SPI_MOSI	1	I	User SPI Port MOSI
SYS_SPI_MISO	1	O	User SPI Port MISO
Ports Used by Fault Logger			
FLT_SPI_SS	1	I	User SPI port slave select for Fault Logger section
CLK_250K	1	I	250 kHz PLD clock from CPLD section
FAULT	1	I	Fault active signal from CPLD section
STAT0	1	I	Status output signal from CPLD section
STAT1	1	I	Status output signal from CPLD section
STAT2	1	I	Status output signal from CPLD section
MEM_SS	1	O	SPI slave select to SPI Flash memory
MEM_SCLK	1	O	SPI clock to SPI Flash memory
MEM_MOSI	1	O	SPI data output to SPI Flash memory
MEM_MISO	1	I	SPI data input to SPI Flash memory
FLT_READY	1	I	System SPI port ready control flag from external microcontroller
FLT_CLEAR	1	I	System SPI port memory clear control flag from external microcontroller
FLT_FULL	1	O	System SPI port memory full control flag to external microcontroller
FLT_BUSY	1	O	System SPI port busy control flag to external microcontroller
Ports used by Closed-Loop Trim			
VIDA0	1	I	VID Channel A select bit 0
VIDA1	1	I	VID Channel A select bit 1
VIDA2	1	I	VID Channel A select bit 2
VIDA3	1	I	VID Channel A select bit 3
VIDA_ENA	1	I	VID Channel A enable (active-low)
VIDB0	1	I	VID Channel B select bit 0
VIDB1	1	I	VID Channel B select bit 1
VIDB2	1	I	VID Channel B select bit 2
VIDB3	1	I	VID Channel B select bit 3
VIDB_ENA	1	I	VID Channel B enable (active-low)
SCP_SPI_SS	1	I	User SPI port Select – Closed-loop Trim

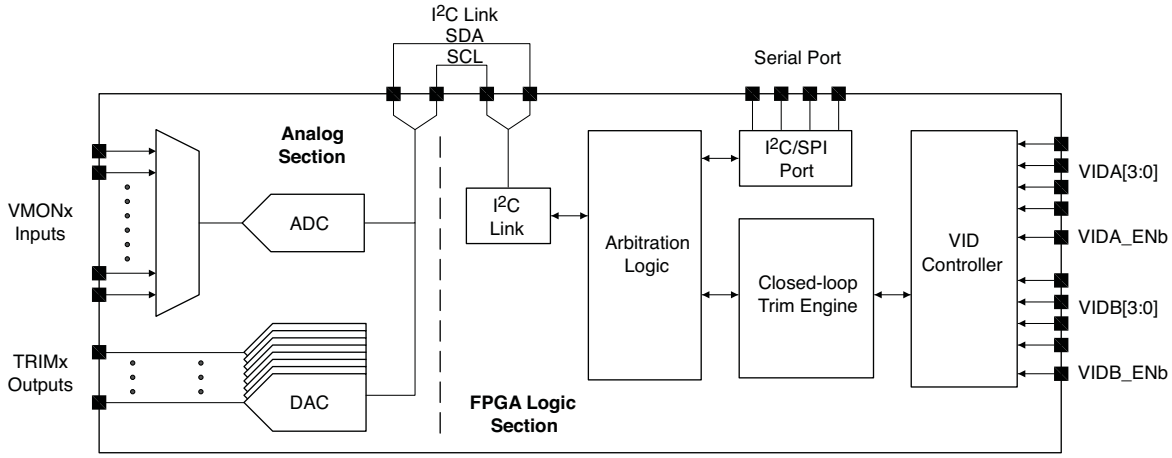
Table 2-1. Top-Level Port Definitions (Continued)

Port	Size	I/O	Description
SCP_SCL	1	I/O	User I²C Port – SCL
SCP_SDA	1	I/O	User I²C Port – SDA
POWR_SCL	1	I/O	I²C Link to analog section (SCL)
POWR_SDA	1	I/O	I²C Link to analog section (SDA)

Closed-loop Trim Theory of Operation

Figure 2-1 shows the top-level organization of the Closed-loop trim (CLT) section of the IP core (FPGA Logic Section) as used in a Platform Manager application. The IP core implements the logic contained in the FPGA section of the device, and communicates to the ADC and DAC hardware functions in the analog section over an I²C link bus optimized for that purpose.

Figure 2-1. Closed-Loop Trim and VID Top-Level Diagram



Some of the key features of the IP core are:

1. Closed-loop Trim Engine

2. Serial Communications Ports

3. VID Control Channels

Each of the above will be discussed in the following subsections.

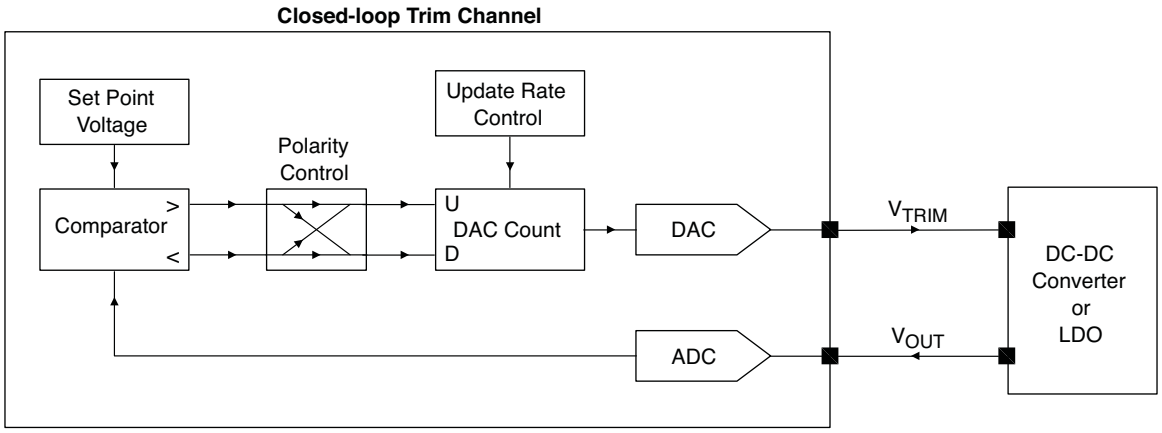
Closed-loop Trim Engine

The purpose of closed-loop trim is to be able to precisely adjust the voltage output of a power supply module to compensate for manufacturing and environmental variations. This function can be extremely useful in high-performance systems where supply voltage drops and ground shift can make open-loop techniques ineffective.

Figure 2-2 shows a functional block diagram of a power supply module being controlled through the closed-loop trim technique. The output voltage (V_{OUT}) of a power supply module, typically either a DC-to-DC converter or linear regulator, is measured by an ADC converter.The resultant value is then compared to a pre-determined 'set point' voltage.If the measured value is different than the set point voltage, a trim voltage supplied to the power supply module (V_{TRIM}) is adjusted. The adjustment is made on the basis of the difference between the set point voltage, measured voltage and polarity of the power supply module's response to a trim signal. For example, assume the power supply module's output voltage increases in response to an increasing trim voltage. In this case, if V_{OUT} is less than the set point voltage, one would increment the DAC count register and increase the V_{TRIM} voltage. This

process is repeated until the V_{OUT} voltage equals the set point voltage. If V_{OUT} is higher than the set point voltage, the DAC count is decremented, and V_{TRIM} reduced, until V_{OUT} equals the set point voltage.

Figure 2-2. Closed-Loop Trim Detail



The IP core provides up to eight independent channels. Each channel is associated with a single, fixed DAC and ADC input (VMON) input. For example, Closed-loop trim channel 1 reads V_{OUT} through VMON1 and adjusts Trim DAC1.

Closed-loop trim operations are configured and controlled through the user serial I/O port (either I^2C or SPI).

VID Control

The Platform Management Utility Functions IP core also provides a voltage identification (VID) function. This allows the user to dynamically select the output the output of a DC-DC converter or linear regulator by specifying a 4-bit identification code.Up to two closed-loop trim channels may be placed under VID control.

The VID function provides a 16-entry look-up table of target voltage settings that are selected by a 4-bit VID code.If two VID channels are instantiated, they will share this look-up table in common.

A VIDx channel is activated when the VIDx_ENA signal is pulled LOW. This causes the VID table entry selected by the associated VIDx[3:0] bits to be loaded into the associated closed-loop trim target register, and the enable bit for that channel to be set HIGH. The VID process then continuously operates to attempt to adjust that channel to the requested VID table voltage. This operation continues even after the VIDx_ENA input is deasserted (brought HIGH). VID operation can only be terminated by a write to the enable register through the serial port. It is therefore not necessary to instantiate the serial port feature if VID is to be used exclusive of other closed-loop trim features. Bringing the VIDx_ENA input low with a different VIDx[3:0] code will cause the VID process to begin trimming to a different target.

The VID function provided by this IP core supports VID operations where a high rate-of-change is not required, as it can take tens to hundreds of milliseconds for the IP core to slew between differing target voltages. Specifically, this VID feature is not intended to support, and will not support, the ‘VID’ features of Intel® processors.

Closed-loop Trim Engine Registers

Table 2-2 lists the user-accessible registers in the closed-loop trim section of the IP core.

Table 2-2. Closed-loop Trim Engine Registers

Name	Address (Hex)	Width	Access	Description
Analog Virtual Registers				
VMON_STATUS0	0x00	8	R	Writes to these register addresses are passed through to the corresponding registers in the analog section of Platform Manager, and reads from these register addresses read from the corresponding analog section registers. For further details on the operation of these registers, refer to the Platform Manager Data Sheet.
VMON_STATUS1	0x01	8	R	
VMON_STATUS2	0x02	8	R	
OUTPUT_STATUS0	0x03	8	R	
OUTPUT_STATUS1	0x04	8	R	
OUTPUT_STATUS2	0x05	8	R	
INPUT_STATUS	0x06	8	R	
ADC_VALUE_LOW	0x07	8	R	
ADC_VALUE_HIGH	0x08	8	R	
ADC_MUX	0x09	8	RW	
UES_BYTE0	0x0A	8	R	
UES_BYTE1	0x0B	8	R	
UES_BYTE2	0x0C	8	R	
UES_BYTE3	0x0D	8	R	
GP_OUTPUT1	0x0E	8	RW	
GP_OUTPUT2	0x0F	8	RW	
GP_OUTPUT3	0x10	8	RW	
INPUT_VALUE	0x11	8	RW	
RESET	0x12	8	W	
TRIM1_TRIM	0x13	8	RW	
TRIM2_TRIM	0x14	8	RW	
TRIM3_TRIM	0x15	8	RW	
TRIM4_TRIM	0x16	8	RW	
TRIM5_TRIM	0x17	8	RW	
TRIM6_TRIM	0x18	8	RW	
TRIM7_TRIM	0x19	8	RW	
TRIM8_TRIM	0x1A	8	RW	

Table 2-2. Closed-loop Trim Engine Registers (Continued)

Name	Address (Hex)	Width	Access	Description
Closed-loop Trim Control Registers				
TRIM_TARGET_HI1	0x20	8	RW	Trim Target and ADC Range settings for each channel
TRIM_TARGET_LO1	0x21	8	RW	
TRIM_TARGET_HI2	0x22	8	RW	
TRIM_TARGET_LO2	0x23	8	RW	
TRIM_TARGET_HI3	0x24	8	RW	
TRIM_TARGET_LO3	0x25	8	RW	
TRIM_TARGET_HI4	0x26	8	RW	
TRIM_TARGET_LO4	0x27	8	RW	
TRIM_TARGET_HI5	0x28	8	RW	
TRIM_TARGET_LO5	0x29	8	RW	
TRIM_TARGET_HI6	0x2A	8	RW	
TRIM_TARGET_LO6	0x2B	8	RW	
TRIM_TARGET_HI7	0x2C	8	RW	
TRIM_TARGET_LO7	0x2D	8	RW	
TRIM_TARGET_HI8	0x2E	8	RW	
TRIM_TARGET_LO8	0x2F	8	RW	
DAC1	0X30	8	RW	Current Closed-loop trim output DAC settings for each channel
DAC2	0X31	8	RW	
DAC3	0X32	8	RW	
DAC4	0X33	8	RW	
DAC5	0X34	8	RW	
DAC6	0X35	8	RW	
DAC7	0X36	8	RW	
DAC8	0X37	8	RW	
CHAN_ENABLE	0x38	8	RW	Enable Closed-loop trim for each channel
LOCK_DETECT	0x39	8	R	Reserved for future use. Reads as 0x00.
LOOP_POLARITY	0x3A	8	RW	Loop Polarity control for each trim channel

Analog Virtual Registers

Read and write operations performed on these registers (addresses 0x00-0x1A) map to identical operations on the corresponding physical registers in Platform Manager's analog section. When directly accessing the analog hardware it is important to be aware of the possibility of contention between user accesses and accesses by the IP core. For example, if one has enabled a closed-loop trim channel and attempts to write to the associated DAC register, the value will be quickly overwritten by the IP core.

TRIM_TARGET_HIx and TRIM_TARGET_LOx Register Descriptions

Figure 2-3. Trim Target Register Pairs

TRIM_TARGET_HIx

ATTEN	X	X	X	D11	D10	D9	D8
b7	b6	b5	b4	b3	b2	b1	b0

TRIM_TARGET_LOx

D7	D6	D5	D4	D3	D2	D1	D0
b7	b6	b5	b4	b3	b2	b1	b0

The trim target registers (Figure 2-3) are used to control the target set point of each trim loop referenced to the ADC measurement code. A 12-bit value is written to these registers indicating the trim target for the channel associated with the register pair. Additionally, the ADC attenuation is set here for each channel, with '0' indicating an ADC range of 0V to 2V and '1' indicating an ADC range of 0V to 6V. Note that regardless of the ATTEN bit setting, all ADC measurements are mapped into the 0-6V range. Setting the ATTEN bit to '0' increases the effective resolution when measuring low voltages.

If a VID channel is instantiated, enabling the VID channel will cause a value from the VID table to be written into the corresponding TRIM_TARGET registers associated with that VID channel.

DACx Register Descriptions
Figure 2-4. DACx Registers

D7	D6	D5	D4	D3	D2	D1	D0
b7	b6	b5	b4	b3	b2	b1	b0

The DAC registers (Figure 2-4) are used primarily for monitoring purposes, as they are updated internally by the trim control loop. They can be set to a preferred initial value prior to initiating a closed-loop trim operation, but should not be set while the trim system is enabled for that channel as doing so may cause abrupt changes in the controlled power supply voltage.

CHAN_ENABLE Register Description
Figure 2-5. Chan_Enable Register

CHEN8	CHEN7	CHEN6	CHEN5	CHEN4	CHEN3	CHEN2	CHEN1
b7	b6	b5	b4	b3	b2	b1	b0

In the Channel enable register (Figure 2-5) setting the bit corresponding to a closed-loop trim channel to '1' initiates the trim operation on that channel. Resetting the bit to '0' stops the stops the trim operation, leaving the trim voltage at the level it was at when the trim operation ceased.

If a VID channel is instantiated, enabling the VID channel will cause a '1' to be written into the CHAN_ENABLE bit corresponding to the trim channel associated with that VID channel. This will initiate trim operations, which will continue until the channel is explicitly disabled by writing a '0' to the appropriate enable bit.

LOCK_DETECT Register Description
Figure 2-6. LOCK_DETECT

LOCK8	LOCK7	LOCK6	LOCK5	LOCK4	LOCK3	LOCK2	LOCK1
b7	b6	b5	b4	b3	b2	b1	b0

The LOCK_DETECT register (Figure 2-6) is reserved for future implementations. It currently reads as 0x00.

LOOP_POLARITY Register Description
Figure 2-7. LOOP_POLARITY

POL8	POL7	PO6	POL5	POL4	POL3	POL2	POL1
b7	b6	b5	b4	b3	b2	b1	b0

The bits of LOOP_POLARITY register (Figure 2-7) control the feedback polarity for the corresponding trim channel. A '0' bit indicates that the trim engine should assume positive gain between the DAC and ADC for that channel, and the DAC count is incremented if the ADC measurement is less than the trim target value, and decremented if the ADC measurement is greater than the trim target value. In contrast, a '1' bit indicates that the trim engine should assume negative gain between the DAC and ADC for that channel, and the DAC count is decremented if the ADC measurement is less than the trim target value, and incremented if the ADC measurement is greater than the trim target value.

Typical Sequence for Setting up a Trim Channel

1. Write the Trim target value and ADC range setting to the channel's TRIM_TARGET_HI and TRIM_TARGET_LO registers.

2. Write loop polarity information to the channel's bit in the LOOP_POLARITY register.

3. Write the initial DAC value to the channel's DAC register (optional).

4. Write a '1' to the channel's associated bit in the CHAN_ENABLE register. This starts the trim process.

Serial Communications Options

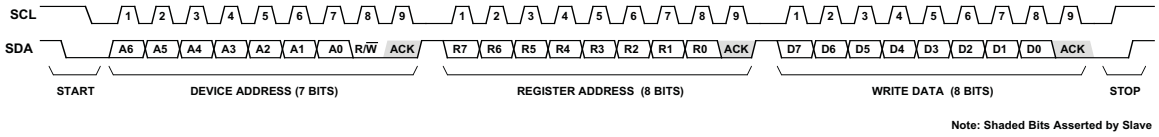
The Platform Management Utility Functions IP core optionally supports serial communication through either I²C or SPI ports. Generally, to use the closed-loop trim functions, either the I²C or SPI options need to be enabled to control the trim process. The exception to this is if one only wants to use the VID channels, which are capable of operating independently of the additional control.

I²C Protocol

The I²C protocol used by this IP core is based on that implemented by the Platform Manager's hardware I²C interface.

To write data to a register, the Closed-loop Trim requires a 3-byte write operation where the first byte specifies the IP core's device address, the second byte specifies the register address to be written to, and the third and final byte specifies the data to be written. Figure 2-8 shows the SDA and SCL waveforms for a register write operation.

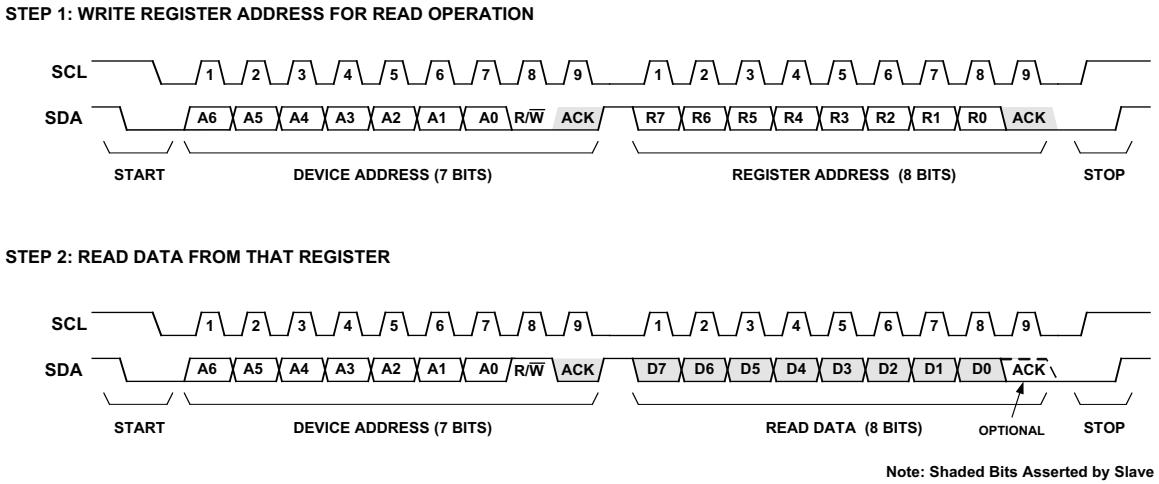
Figure 2-8. I²C Register Write Waveforms



The IP communicates its readiness through the I²C ACK bit. If the user transmits a write command through the I²C port and the IP core is not ready to accept the command, it will not assert the ACK bit. This behavior must be taken into account when designing software to communicate to the IP core (e.g. planning for re-try operations).

Reading a register through the I²C port requires two separate I²C data frames. In the first frame, two bytes are transmitted as a 'write' operation with the device address and the register address. In the second frame, two more bytes are transmitted as a 'read' operation with the device address in the first byte, and the returned data transmitted by the IP core in the second byte. Figure 2-9 shows the details of these two transmission cycles.

Figure 2-9. I²C Register Read Waveforms



In the case of a read operation, the ACK bit is only asserted when the IP core successfully accepts a communication attempt. Because the IP relays certain register read requests to the Platform Manager hardware, there is a finite time elapsed between finishing Step 1 of the read operation and when data becomes available in the IP core to be read in Step 2. If the data is not available, the IP core will not assert the ACK bit in response to the Step 2 read request. The user can repetitively poll the IP core until data becomes available.

A comprehensive overview of I^2C communications is a complex subject that is beyond the scope of this document. For further details, the reader should consult the *I^2C-Bus Specification and User Manual* published by NXP Semiconductors.

SPI Protocol

The Platform Management Utility Functions IP core also supports SPI communications up to a 500kHz bit rate. A 16-bit data frame is employed, supporting compatibility with hardware-based SPI transceivers commonly included with microcontrollers as well as 'bit-banged' implementations in software.

In each communications transaction, a MOSI word is transmitted from the external SPI master to the IP core, and a MISO word is simultaneously transmitted from the IP core to the external SPI master. The meanings of the bits for both the MOSI and MISO words are shown in Figures 2-10 and 2-11.

Figure 2-10. MOSI Word (Received by IP Core)

WR	RD	A5	A4	A3	A2	A1	A0	D7	D6	D5	D4	D3	D2	D1	D0
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Most Significant Byte								Least Significant Byte							

The functions of the bits in the MOSI word are:

- **WR** – '1' indicates that a write of D[7:0] should be performed to register A[5:0]
- **RD** – '1' indicates that the IP core should perform a read of register A[5:0]. The read data will become available on a subsequent transaction.
- **A[5:0]** – Register address for read and write operations
- **D[7:0]** – Data to be written

Figure 2-11. MISO Word (Returned by IP Core)

BF	0	0	0	0	0	0	0	D7	D6	D5	D4	D3	D2	D1	D0
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Most Significant Byte								Least Significant Byte							

The functions of the bits in the MISO word are:

- **BF** – '1' indicates that the IP core is busy and the current transaction (read or write) is invalid
- **D[7:0]** – Data returned from a previous read operation

Data Format Options

The IP core's SPI interface supports a variety of common interface protocols. The three user-selectable options are:

- Clock Polarity (Normal/Inverted)
- Clock Phase (Normal/Delayed)
- Bit Order (MSB/LSB First)

The data frame formats associated with various combinations of these options are shown in the following figures:

Figure 2-12. SPI Data Frame (LSB First, Normal Clock, No Delay)

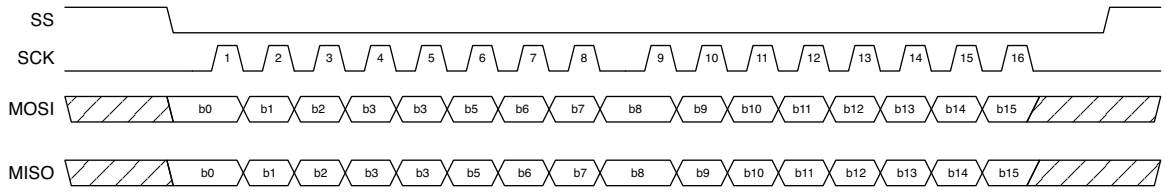


Figure 2-13. SPI Data Frame (LSB First, Inverted Clock, No Delay)

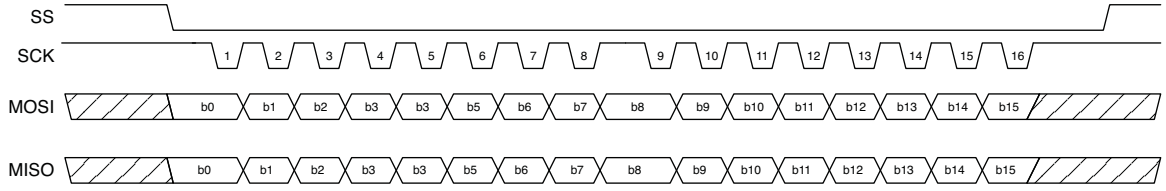


Figure 2-14. SPI Data Frame (MSB First, Normal Clock, No Delay)

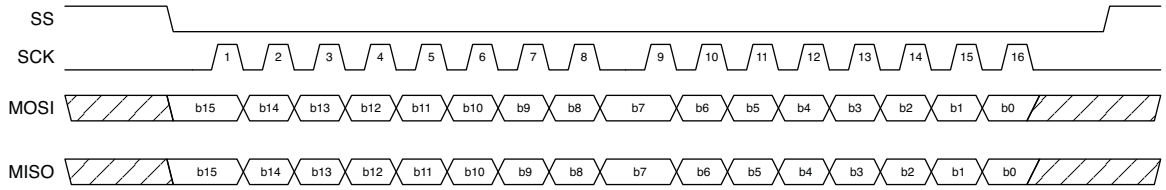
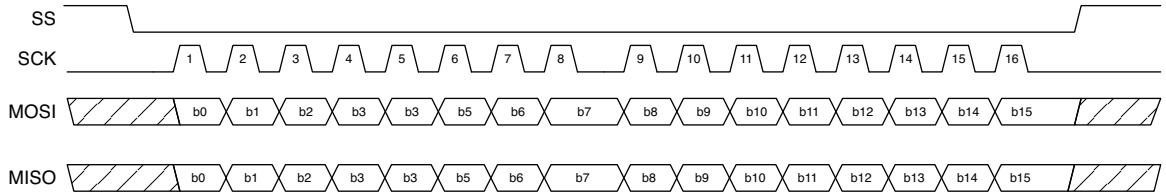


Figure 2-15. SPI Data Frame (LSB First, Normal Clock, Phase Delay)



System Fault Logging Theory of Operation

The Platform Management Utility Functions IP core allows a system to save the state of the different inputs and outputs of the Platform Manager device when a fault condition occurs. The fault record of the inputs and outputs is stored in a SPI Flash memory device to allow the user to read back the fault record later for further analysis if desired.

The Platform Manager device uses the on-board analog-to-digital converter to monitor the voltage levels in the system. The user can set high and low voltage alarm points within the device for each channel and these can then be used to initiate different control actions of the user's choosing. For this IP core a voltage alarm will cause a fault status output to be driven high and the status of all the voltage monitor channels to be output on three data status lines. The user can modify the conditions which will initiate the fault status output as required for their system design.

The fault status output and the three data status lines are connected to the FPGA section of the Platform Manager device which then captures the fault data, formats it, and writes the formatted data to a SPI Flash memory for later retrieval.

The sequence of events can be summarized as follows:

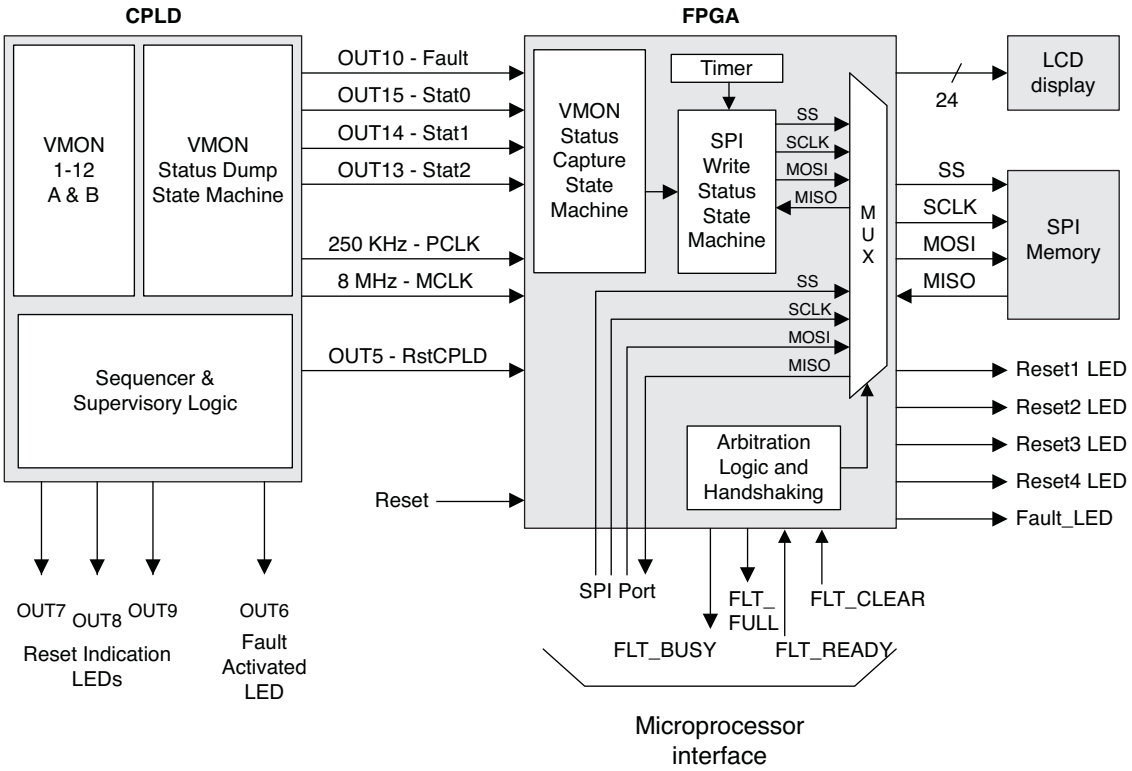
1. The Platform Manager detects a fault on one or more of the VMON inputs and dumps the VMON and I/O status to the FPGA section.
2. The dump of VMON and I/O status is implemented using Supervisory Logic Equations and happens automatically using outputs and clock pins of the Platform Manager CPLD section.
3. The FPGA section implements a receiver state machine to capture the VMON and I/O status.
4. The FPGA section adds a time stamp to the VMON and I/O status information and sends the commands and writes the data to a SPI Flash memory device.

A block diagram of this IP core is shown in Figure 2-16.

To simplify the design and minimize the SPI write time, each page within the SPI Flash is dedicated to a single event log. The first byte of each page is used as a Fault Flag. If the page is erased the Fault Flag will have a value of 0xFF. If the page has any fault data recorded then the Fault Flag will have a value of 0xC3. This allows both this fault logging design or an external fault reader (a microprocessor or microcontroller) to know which pages have valid data. For this IP core, 10 bytes are used to log a fault and the organization of the bytes within a page is shown in Tables 2-3 and 2-4.

A key feature of this IP core is the SPI-MUX and arbitration logic. This allows the Platform Manger to share the SPI Flash Memory with any embedded or stand alone microprocessor or microcontroller. The arbitration logic prevents the two SPI masters from communicating to the SPI Flash memory at the same time.

Figure 2-16. Platform Management Utility Functions IP Core Block Diagram



The arbitration logic uses two outputs, the FLT_BUSY and FLT_FULL signals, to provide a status signal to the outside microprocessor or microcontroller. The FLT_BUSY signal is high when the FPGA section is writing a fault to the SPI memory or waiting to write a fault. The FLT_FULL signal is high when the SPI memory has faults written in

each page and the FPGA section cannot write any additional faults to the memory. For this Fault Logging IP design the fault limit is set at 250 faults.

The arbitration logic also uses two inputs, the FLT_READY and FLT_CLEAR signals, to allow the microprocessor or microcontroller to signal the Platform Manager that it is communicating with the SPI memory directly. When the microprocessor wishes to communicate with the SPI memory directly it should check that the FLT_BUSY signal is low. If FLT_BUSY is high the microprocessor should wait for it to go low. Once the FLT_BUSY signal is low the microprocessor should set the FLT_READY signal low to switch the MUX to allow it to communicate with the SPI memory. When the microprocessor is done communicating with the SPI memory it must set the FLT_READY signal high again to allow the Platform Manager to log faults to the SPI memory. As long as the FLT_READY signal is low the Platform Manager cannot log a fault to the SPI memory. If a single fault condition occurs while the FLT_READY signal is low, the Platform Manager will capture that fault log and wait until the FLT_READY signal is high to write the log to the SPI memory. Additional faults while FLT_READY is low are lost if they occur prior to the first fault being written to the SPI memory.

The microprocessor should pulse the FLT_CLEAR signal low when it is erasing the SPI memory to signal the Platform Manager that it should reset the memory address counter. This will allow the Platform Manager to begin logging faults at the beginning of the SPI memory address again. After the SPI memory has been cleared the FLT_CLEAR signal should be driven high again (or released as there is a internal pull-up on the signal) to wait for the next memory clear command.

Table 2-3. SPI Page Memory Map Byte Arrangement

Byte Address	Byte Description	Comments
0x00	Fault Flag	0xFF = No fault, page empty, 0xC3 = Fault log, page written
0x01	VMON1 - VMON4	See Table 2-4 for details of fault order
0x02	VMON5 – VMON8	See Table 2-4 for details of fault order
0x03	VMON9 – VMON12	See Table 2-4 for details of fault order
0x04	OUT1 – OUT8	See Table 2-4 for details of fault order
0x05	OUT9, OUT11, OUT12, IN1 – IN4	See Table 2-4 for details of fault order
0x06	Seconds_0	Least significant byte – long integer
0x07	Seconds_1	
0x08	Seconds_2	
0x09	Seconds_3	Most significant byte – long integer

Table 2-4. VMON and I/O Fault Memory Map Bit Arrangement

Byte Address	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x01	VMON4_B	VMON4_A	VMON3_B	VMON3_A	VMON2_B	VMON2_A	VMON1_B	VMON1_A
0x02	VMON8_B	VMON8_A	VMON7_B	VMON7_A	VMON6_B	VMON6_A	VMON5_B	VMON5_A
0x03	VMON12_B	VMON12_A	VMON11_B	VMON11_A	VMON10_B	VMON10_A	VMON9_B	VMON9_A
0x04	OUT8	OUT7	OUT6	OUT5	HVOUT4	HVOUT3	HVOUT2	HVOUT1
0x05	IN4	IN3	IN2	IN1	SPARE	OUT12	OUT11	OUT9

In this IP core, the memory map bytes 0x06 through 0x09 contain a 32-bit binary representation of the timer value in seconds. The timer can count up to a value of 2^{32} -1 seconds which is equivalent to over 136 years. The timer will be reset when the device is powered down or a reset is issued.

To implement this IP core, a designer will need to modify the CPLD Main Sequence state machine logic to fit their design requirements.

Design Details

As shown in Figure 2-16, the fault logging process is divided between the CPLD and the FPGA portions of the Platform Manager device. The CPLD detects and dumps the faults to the FPGA that captures the faults. The FPGA then writes them to a non-volatile SPI memory. The FPGA also provides arbitration logic, a timer, and a MUX interface to the SPI memory for microprocessor support.

Figure 2-17 shows the details of the VMON and I/O status dump from the CPLD section of the Platform Manager to the FPGA section and Figure 2-18 shows the transfer of data from the FPGA section to the SPI Flash memory. The VMON and I/O status dump is generated within the CPLD section using an internal 250 KHz clock (fixed) which also drives the PCLK signal. The logic of the CPLD section is clocked from the PCLK signal by default. For this IP core, the PCLK signal is used to drive the VMON Status Capture state machine inside the FPGA section to synchronize with the data output of the CPLD section.

The 24 VMON status inputs and the I/O status are transferred to the FPGA in 13 PLD clocks (52 µs). The FPGA uses the negative edge of the PCLK to latch the VMON and I/O status presented on the three-bit bus (see Figure 2-17).

Figure 2-17. Fault Logging Status Dump Waveforms

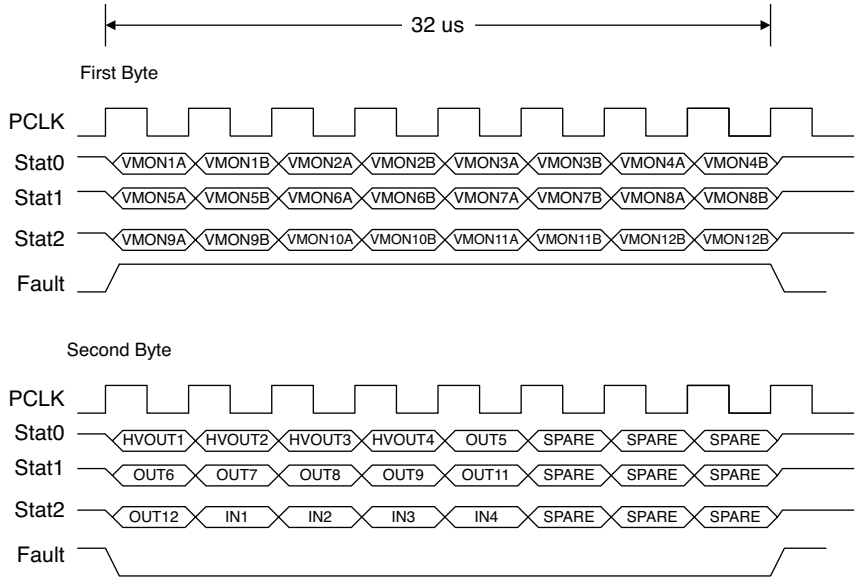
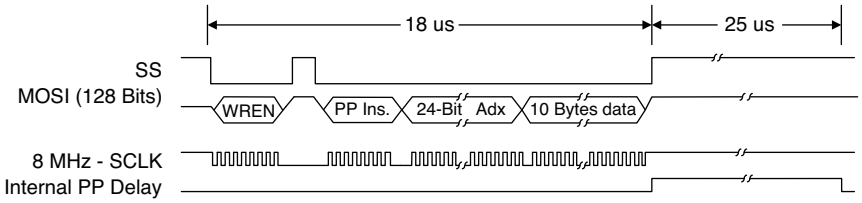


Figure 2-18. Fault Logging SPI Page Program Waveforms



For convenience, this IP core uses the 8 MHz MCLK signal from the CPLD section to drive the SPI memory write operations (the MCLK signal of the Platform manager is a fixed rate clock.) The entire operation takes less than 100 µs to store the fault record to the SPI Flash memory. It takes approximately 52 µs to dump the fault record to the FPGA and approximately 43 µs to store the fault record to the SPI memory as shown in Figures 2-17 and 2-18.

CPLD Logic Description

To implement this IP core, the designer will need to modify the CPLD Main Sequence state machine logic to fit their design requirements. The following section describes the CPLD logic so the user can understand what changes are needed for their design. The Supervisory Equations cause the VMON and I/O status bits to be sent out on the status lines and this section should NOT be modified. The FPGA code will read in the VMON and I/O status bits and send them to the SPI memory and should NOT be modified either. The FPGA code and the Supervisory Equations must be properly matched in order to get the proper order for the fault log record.

The Main Sequence state machine is a very simple monitoring loop. Steps 0 and 1 provide the start-up and initialization of the outputs in a safe state and set the FAULT_ DUMPn flag high. Step 2 waits for the monitored supplies to reach a normal value to prohibit false indications of faults before the system is fully powered up. This step can be modified as required to meet the needs of the design being considered. Step 3 is a NOP which is inserted only to provide an additional comment in the design and can be removed if desired. Step 4 monitors the status of VMON8 and VMON9 and step 5 controls the FAULT_ DUMPn flag. Step 6 waits for both VMON8 and VMON9 to drop below their lower trip points before the design resets itself back to the beginning of the sequence. This step is primarily used to insure that only a single fault is logged for each event when demonstrating this IP core on the Platform Manager Evaluation Board. This step in the sequence should be modified to shut the system down or take other actions as appropriate for the design being considered.

When VMON8 and VMON9 are both between the trip points the system is considered to be running normally. When either of these VMONs moves outside the windowed trip points the FAULT_ DUMPn flag is activated and the system begins logging the status of all the VMON inputs, the digital inputs, the HVOUTS, and the digital outputs. The fault record does not include OUT10, OUT13, OUT14, or OUT15 since these are used by this IP core to log the faults. OUT16 is reserved for use by the long timer function so it is not logged either.

In the Analog Input Settings dialog (from the main schematic window) VMON8 and VMON9 are configured in Window mode with the lower trip point of 1.5V and the upper trip point of 2.5V. On the Platform Manager Evaluation Board VMON8 and VMON9 are connected to the slide potentiometers to simulate either an under voltage or an over voltage situation.

In this IP core, the other analog input settings are all set to predefined limits which are applicable to the Platform Manager Evaluation Board. These settings are used to insure that the faults which are logged while using this IP core with the Platform Manager Evaluation Board produce a known fault record for comparison purposes, based upon the voltages being monitored on the evaluation board. The known fault record is shown in Table 2-5. All these analog input settings can be changed by the designer to values applicable to their design when applying this IP core.

Table 2-5. Standard Fault Record for Operation with the Platform Manager Evaluation Board

VMON or I/O Status	Value[1]
VMON1	A = 0, B = 0
VMON2	A = 0, B = 0
VMON3	A = 1, B = 1
VMON4	A = 1, B = 1
VMON5	A = 0, B = 0
VMON6	A = 0, B = 0
VMON7[2]	A = 1, B = 1
VMON8	Per slider
VMON9	Per slider
VMON10	A = 1, B = 1
VMON11	A = 0, B = 0
VMON12	A = 0, B = 0
HVOUT1	0

VMON or I/O Status	Value[1]
HVOUT2	0
HVOUT3	0
HVOUT4	0
OUT5	0
OUT6	1
OUT7	0
OUT8	0
OUT9	0
OUT11	1
OUT12	1
IN1	1
IN2	1
IN3	1
IN4	1

1. A VMON value of 00 represents and under voltage condition. A VMON value of 11 represents a good reading (using the Windowed mode). A VMON value of 01 represents an over voltage condition. A value of 10 is an invalid reading.
2. VMON7 monitors the Thermistor voltage on the board. This value could change depending upon the ambient temperature of the board.

This design is easily modified to include any number of other VMON or input conditions to trigger the fault dump by adding to the Boolean logic in Step 4. Other sequences and controls could also be added before Step 4 based on the design requirements.

In this design, Step 4 of the main sequence monitors VMON8 and VMON9 (the slide potentiometers on the Platform Manager Evaluation Board). Step 5 sets the FAULT_ DUMPn signal low if the voltage is in alarm. Setting the FAULT_ DUMPn signal low causes the supervisory equations to send out the VMON and I/O status on the three status outputs STAT0, STAT1, and STAT2.

Start-up and Reset Condition

When the design is powered up or a reset is issued, this IP core will begin by reading the SPI Flash memory to determine if any faults are stored in the memory at the current time. The presence of a fault in a page of the SPI Flash is indicated by byte address 0x00 as shown in Table 2-1. If a fault is present in a page the design will increment the page address counter to the next page address and re-check for a fault in the new page. This will continue until an empty SPI page is found and then the next fault will be logged into this location. This allows any existing faults to remain logged into the SPI Flash memory until the microprocessor issues a SPI Flash memory erase command.

When a microprocessor erases to SPI memory, it will also set the FLT_CLEAR status bit low so the IP core will know that a memory clear has occurred. When the FLT_CLEAR status bit is received the IP core will reset the address counter to begin at page 00. Because of this design feature, the faults will be logged into the SPI Flash memory in sequential order.

Pin Assignments

The pin assignments for the CPLD section of the Platform Manager device are contained within the PAC file and can be modified by the designer if necessary. These pin assignments are shown in the Pins window.

The pin assignments for the FPGA section of the Platform Manager device are contained in the “VM1220_LB.lpf” file which is located in the “<project_name>_FPGAFiles” directory. This file is a text file that can be edited with the Lattice Editor built into PAC-Designer or with another ASCII text editor. See the Platform Manager Data Sheet for the list of pin names which are available for use with this device. Note that the existing “VM1220_LB.lpf” file uses the pad names (ball function) but the pin/ball numbers can also be used in the file.

Parameter Settings

The IPexpress tool is used to create IP and architectural modules in the ispLEVER software. Refer to "IP Core Generation" on page 23 for a description on how to generate the IP core. Table 3-1 provides the list of user-configurable parameters for the Platform Management Utility Functions IP core. The parameter settings are specified using the IP core Configuration GUI in IPexpress.

Table 3-1. Parameters Set in IPexpress

Parameters	Range	Default
Serial Communications Option		
User Serial Port Type	None, SPI, I2C	None
Device Address (only applicable to I^2C option)	0x00-0x7F	0x20
Clock Polarity (only applicable to SPI option)	Normal, Inverted	Normal
Clock Phase (only applicable to SPI option)	Normal, Delayed	Normal
Bit Order (only applicable to SPI option)	LSB first, MSB first	LSB first
Closed-loop Trim Option		
Instantiate Closed-loop Trim Engine	No, Yes	No
Trim Update Rate	Fast, Medium, Slow	1
VID Channel A	None, 1, 2, 3, 4, 5, 6, 7, 8	None
VID Channel B	None, 1, 2, 3, 4, 5, 6, 7, 8	None
VID Code Lookup Table [0..15]	0-6, Real	0
Fault Logger Option		
Instantiate Fault Logger Engine	No, Yes	No

In addition to parameters set by the user in IPexpress, a number of 'hidden' configuration parameters are automatically passed to IPexpress from PAC-Designer (Table 3-2) in an LPC file. These parameters describe important configuration data with respect to the analog section of the Platform Manager device. If you invoke IPexpress either from ispLEVER or in stand-alone mode, and attempt to generate the Platform Management Utility Functions IP core, the resultant IP may fail to operate as expected. For this reason, it is absolutely mandatory that IPexpress be invoked only from PAC-Designer to generate this IP core.

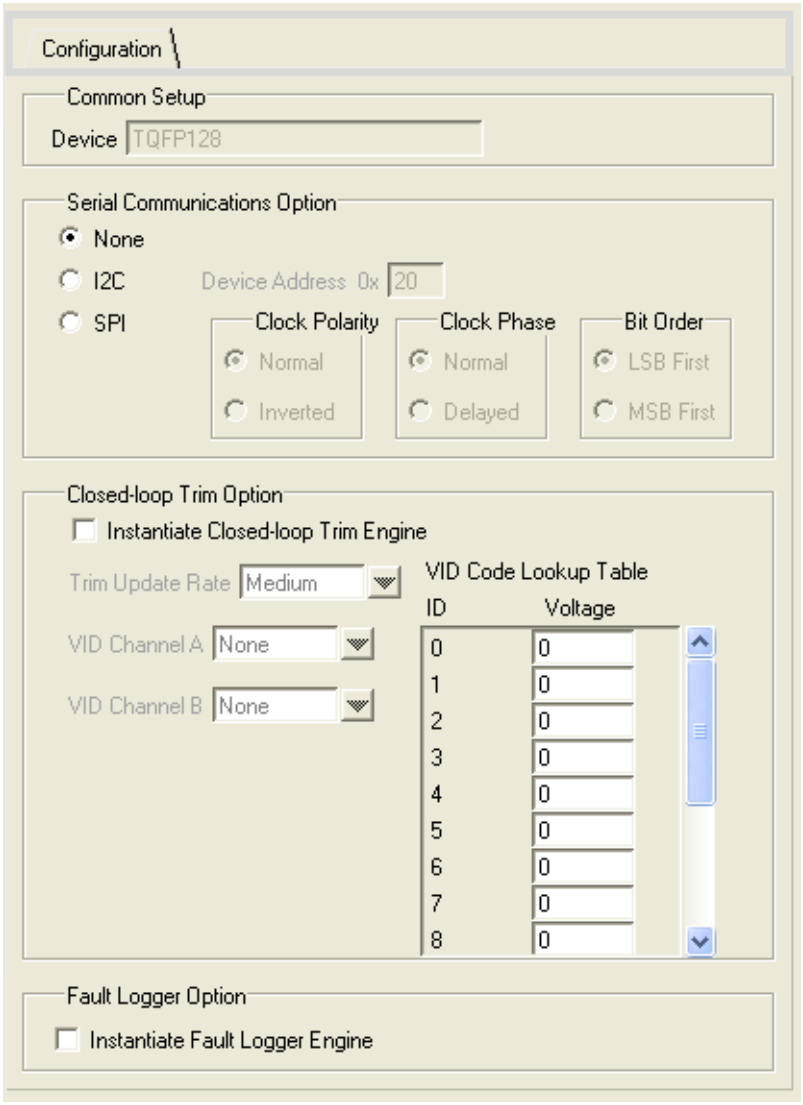
Table 3-2. 'Hidden' Parameters Passed Automatically from PAC-Designer

Parameter	Range	Default
POWRADDR	0-127	0
CLT_POL[1..8]	0,1	0
CLT_DAC[1..8]	0-255	128

Configuration

Figure 3-1 shows the contents of the Configuration tab.

Figure 3-1. Configuration Tab



Serial Communications Option

This section configures a user-accessible serial communications port to the Closed-loop Trim engine.

User Serial Port Type

Selects the type of serial communications port, either I^2C (400 kHz), SPI, or none.

Device Address

If I^2C is selected, this specifies the bus device address.

Clock Polarity

If SPI is selected, selects normal or inverted clock. For a normal clock, data is clocked in on the rising edge and for an inverted clock, data is clocked in on the falling edge.

Clock Phase

If SPI is selected, selects a ‘delayed’ clock protocol.

Bit Order

If SPI is selected, selects the bit order. Note that since SPI data is transmitted in 16-bit packets, this also effectively reverses the order in which the individual bytes are transmitted and received.

Closed-loop Trim Option

This group of options instantiates and configures the Closed-loop Trim and provides for VID control.

Instantiate Closed-loop Trim Engine

Enables the Closed-loop Trim Engine.

Trim Update Rate

Sets the update rate for Closed-loop Trim operations. Fast = 10ms, Medium = 25ms, Slow = 50ms. The loop rate is the time required to read the analog input voltage, compare it to the set point, and increment or decrement the associated DAC for all enabled Closed-loop Trim channels.

VID Channel A, B

Enables and selects a given ADC/DAC channel for VID control.

VID Code Lookup Table

Defines the look-up table of voltages to be used by VID control. ID (0-15) corresponds to VID control codes (0000-1111).

Fault Logger Option

Instantiate Fault Logger Engine. The fault-logger has no user-configurable parameters

IP Core Generation

This chapter provides information on how to generate the Platform Management Utility Functions IP core using the ispLEVER software IPexpress tool, and how to include the core in a top-level design.

Licensing the IP Core

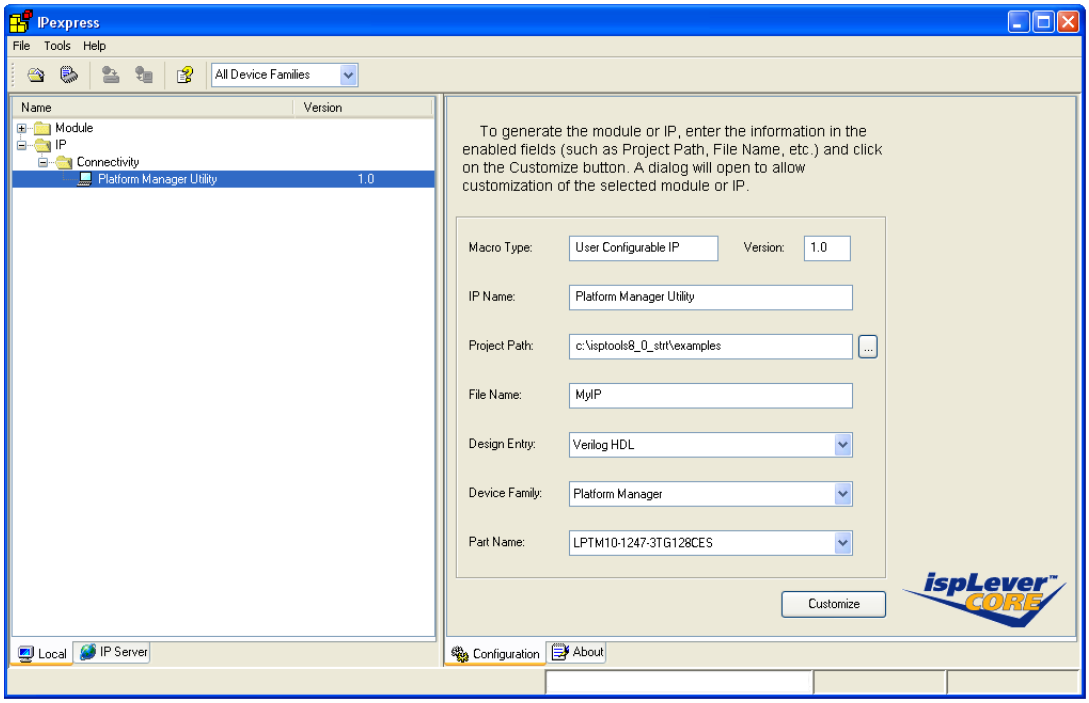
No license is required to use this IP core.

Getting Started

The Platform Management Utility Functions IP core is available for download from the Lattice IP server using the IPexpress tool in ispLEVER. The IP files are automatically installed using ispUPDATE technology in any user-specified directory. After the IP core has been installed, it will be available in the IPexpress GUI dialog box shown in Figure 4-1. The ispLEVER IPexpress tool GUI dialog box for this IP core is shown in Figure 4-1. To generate a specific IP core configuration the user specifies:

- **Project Path** – Path to the directory where the generated IP files will be located.
- **File Name** – "username" designation given to the generated IP core and corresponding folders and files.
- **Design Entry Type** – This IP core generates Verilog output only.
- **Device Family** – This IP only supports the Platform Manager device family
- **Part Name** – Specific targeted part within the selected device family.

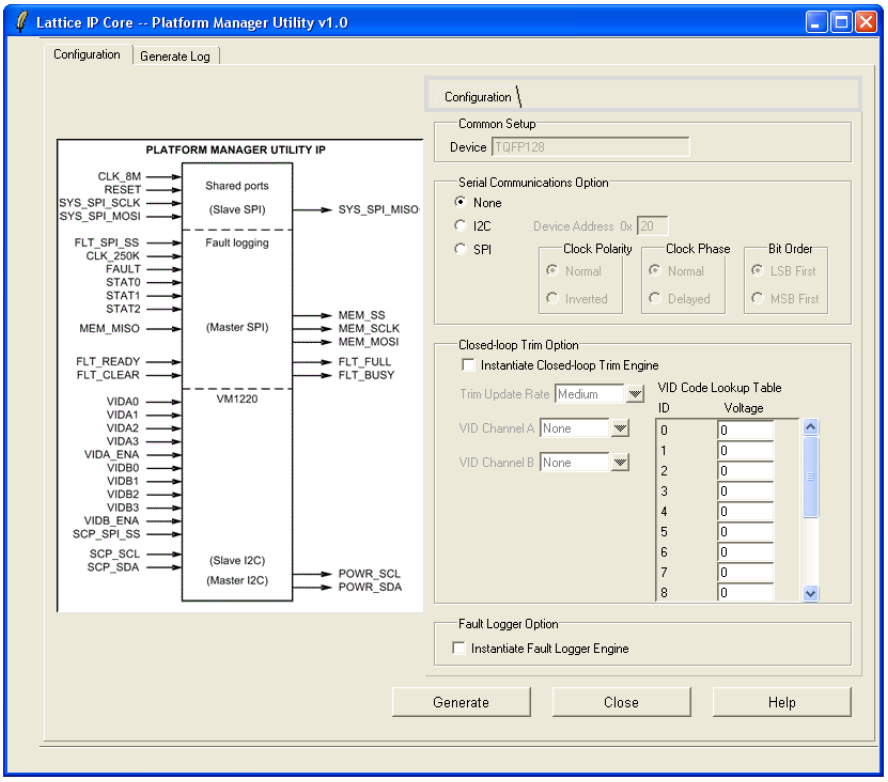
Figure 4-1. The IPexpress Tool Dialog Box



Note that if the IPexpress tool is called from within an existing project, Project Path, Design Entry, Device Family and Part Name default to the specified project parameters. Refer to the IPexpress tool online help for further information. To create a custom configuration, the user clicks the **Customize** button in the IPexpress tool dialog box to display the Platform Management Utility Functions IP core Configuration GUI, as shown in Figure 4-2. From this

dialog box, the user can select the IP parameter options specific to their application. Refer to "Parameter Settings" on page 20 for more information on the Platform Management Utility Functions IP core parameter settings.

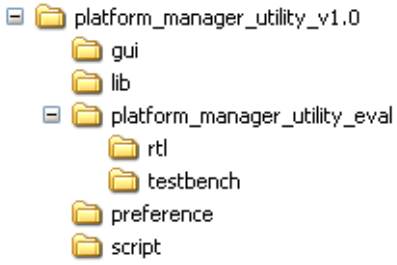
Figure 4-2. The IPexpress Tool Dialog Box - Configuration GUI



IPexpress-Created Files and Top Level Directory Structure

When the user clicks the **Generate** button in the IP Configuration dialog box, the IP core and supporting files are generated in the specified "Project Path" directory. The directory structure of the generated files is shown in Figure 4-3.

Figure 4-3. Platform Management Utility Functions IP Core Directory Structure



The design flow for IP created with the IPexpress tool uses an unprotected Verilog model for both synthesis and simulation. While the resulting model is tailored for automatic integration with the PAC-Designer design environment, it may also be freely modified by the user for use with different design environments and to meet different application requirements.

Table 4-1 provides a list of key files and directories created by the IPexpress tool and how they are used. The IPexpress tool creates several files that are used throughout the design cycle. The names of most of the created files are customized to the user's module name specified in the IPexpress tool.

Table 4-1. File List

File	Description
<username>_bb.v	This file provides the synthesis code for the user's synthesis
<username>_bb_TB.v	This file provides testbench code for the user's simulation
<username>.lpc	This file contains the IPexpress tool options used to recreate or modify the core in the IPexpress tool
generate_core.tcl	Created when the GUI "Generate" button is pushed, invokes generation, may be run from command line
fault_logger_templ.v	Module I/O template for Fault Logger subcomponent
vm1220_templ.v	Module I/O template for Closed-loop Trim/VID subcomponent
<username>_generate.log	IPexpress scripts log file
<username>_gen.log	IPexpress IP generation log file
<username>_filelist.log	IPexpress IP list of generated files

Simulation Evaluation

Simulation support for the IP core is limited to the generation of a skeleton test bench template file (<username>_bb_TB.v), which can be modified by the designer for use Aldec, ModelSim or other supported HDL simulators.

Although the Platform Management Utility Functions IP core can be implemented with any Lattice synthesis flow that supports Platform Manager, it is primarily intended for use with PAC-Designer. This chapter outlines the steps required to incorporate this IP core in a PAC-Designer design.

Note that some parameters for the IP core are passed from PAC-Designer to IPexpress. It is therefore mandatory that IPexpress be launched from PAC-Designer and not from ispLEVER, Diamond, or in stand-alone mode.

Closed-Loop Trim with PAC-Designer

The following procedure can be used to realize the Closed-loop Trim functions of this IP core:

1. Start PAC-Designer 6.0
2. Create a new Platform Manager design or open an existing design.
3. From the main schematic, click on the **I²C block** and set the analog device address. This will be passed to IPexpress for configuring the IP core.
4. From the main schematic, click on the **Margin/Trim block**. Here you will configure the trim DACs for I²C control as well as set trim loop polarity and set initial trim DAC values.
5. For each Trim DAC to be used, click on the entry to open the configuration dialog.
   a. Import suitable DC-DC converter type.
   b. Set 'Profile 0 Mode' to **I²C Controlled Value**.
   c. Set Voltage **Profile 0 Target Voltage**.
   d. Click the **Calculate** button.
   e. Click **OK**.
6. Return to the main schematic.
7. Launch IPexpress (opens in a new window).
   a. Click on the **Utility IP block** in the main schematic.
   b. Check the **Enable** box.
   c. Select the **IP name** in the list.
   d. Click the **Configure** button.
8. Configure the IP core by setting IPexpress options.
   a. Click **Generate**.
   b. Click **Close**.
9. Return to the **Utility IP Setting** window in PAC-Designer and click **OK**.
10. Click on the **FPGA Logic Window**, opening LogiBuilder.
11. Select **File > Import Sub Module > Configure Sub Module** from the main menu.
12. Open the Port Configuration dialog by clicking on the **Port Mapping** button in the Module Definition dialog.
13. Assign signal names to the port names. If an input port is unused you may assign a constant signal value of '0' or '1'. When all ports are assigned, click **OK**.
14. Click **OK** in the Module Definition dialog.

Fault Logging with PAC-Designer

The following procedure can be used to implement the Fault Logging functions in the Platform Manager device. The Platform Management Utility Functions IP core should be loaded into PAC-Designer before beginning other logic implementation because loading the IP will over-write any logic already in the CPLD LogiBuilder window.

1. Start PAC-Designer 6.0
2. Create a new project for the LPTM10-12107 device.
   Note: Do NOT use spaces in the file name or path to the file name.
3. Double-click on the **Utility IP block** in the main schematic window.
4. Select the **Enable** check-box.
5. Select the **IP name** from the list.
6. Click on the **Configure** button.
7. Click on the **Instantiate Fault Logger** check-box.
8. Check that Closed-loop Trim engine check-box is NOT selected for this design.
9. Click the **Generate** button.
10. Click the **Close** button.
11. Click the **OK** button.
12. Click **Yes** to the next dialog box asking if you want to continue.
13. Double-click on the **CPLD Logic block** of the main schematic view to open the LogiBuilder sequence or Supervisory Equations and review the logic for this IP core. Make any required logic additions to the Logi-Builder sequence in this window.
14. Click on the **Compile** icon or use the **Tools > Compile HDL Design** menu item to generate a compiled version of the design.
15. When complete, click on the link for **Click Here to See the Fitter Report**.
16. After reviewing the Fitter report click **OK** to continue.
17. Click on the **JED** button or **File > Export** menu option to generate the JEDEC file.
18. Click **OK** to continue.
19. Check that the combined JEDEC file was created in the Project directory.

Chapter 6:

Support Resources

This chapter contains information about Lattice Technical Support, additional references, and document revision history.

Lattice Technical Support

There are a number of ways to receive technical support.

Online Forums

The first place to look is Lattice Forums (www.latticesemi.com/support/forums.cfm). Lattice Forums contain a wealth of knowledge and are actively monitored by Lattice Applications Engineers.

Telephone Support Hotline

Receive direct technical support for all Lattice products by calling Lattice Applications from 5:30 a.m. to 6 p.m. Pacific Time.

- For USA & Canada: 1-800-LATTICE (528-8423)
- For other locations: +1 503 268 8001

In Asia, call Lattice Applications from 8:30 a.m. to 5:30 p.m. Beijing Time (CST), +0800 UTC. Chinese and English language only.

- For Asia: +86 21 52989090

E-mail Support

- techsupport@latticesemi.com
- techsupport-asia@latticesemi.com

Local Support

Contact your nearest Lattice Sales Office.

Internet

www.latticesemi.com

References

- Platform Manager Data Sheet

Revision History

Date	Document Version	IP Version	Change Summary
October 2010	01.0	1.0	Initial release.
January 2011	01.1	1.1	Minor enhancements to code base.

Appendix A:

Resource Utilization

This appendix gives resource utilization information for Platform Manager devices using the Platform Management Utility Functions IP core. IPexpress is the Lattice IP configuration utility, and is included as a standard feature of the ispLEVER and PAC-Designer design tools. Details regarding the usage of IPexpress can be found in the IPexpress, ispLEVER and PAC-Designer help systems. For more information on the ispLEVER design tools, visit the Lattice web site at: www.latticesemi.com/software.

Table A-1. Closed-Loop Trim/VID Performance and Resource Utilization[1, 2]

Core Configuration	Slices	LUTs	Registers
Closed-loop trim with I^2C port	263	525	256
Closed-loop trim with I^2C port and two instantiated VID channels	262	524	256
Closed-loop trim, no serial port, two VID channels	153	306	147
Closed-loop trim with SPI port	219	438	237

1. Performance and utilization characteristics are generated using LPTM10-12107, with Lattice PAC-Designer 6.0 and ispLEVER 8.1 SP1 software. The performance and utilization may vary when using this design in a different device or software versions.
2. LUT utilization may vary slightly depending on values of numerical constants used for configuration.

Table A-2. Fault Logger Resource Utilization[1, 2, 3]

IP Core Configuration	Device	FPGA LUTs	FPGA Slices	FPGA Registers
Fault Logging	Platform Manager	344	183	316

Device	CPLD Macrocells	CPLD Product Terms	VMONs	I/Os	Timers	HVOUTs
Platform Manager	29	136	2[2]	28[3]	0	0

1. Performance and utilization characteristics are generated using LPTM10-12107, with Lattice PAC-Designer 6.0 and ispLEVER 8.1 SP1 software. The performance and utilization may vary when using this design in a different device or software versions.
2. The VMON inputs used for this design are only needed to demonstrate the design on the Platform Manager Evaluation Board. All 12 VMON inputs are available for the user when implementing this design.
3. The number of I/Os includes 22 I/Os for the FPGA and six CPLD outputs. This is for the Fault Monitoring and Logging IP design with a SPI port interface which is available for a microprocessor or microcontroller. The external port for the microprocessor uses four I/Os for the SPI port plus another four I/Os for the arbitration logic. The external SPI port for the memory uses only four I/Os.
4. LUT utilization may vary slightly depending on values of numerical constants used for configuration.