Fault Logging Using Platform Manager 2

March 2015 Technical Note TN1277

Introduction

For complex systems, often there is a requirement to monitor system power supplies, currents, temperatures, I/Os, or other critical signals and report or log any faults if the system is operating out of its specifications. In addition, for maintenance and troubleshooting purposes it is very desirable to know the cause of a system failure, the type of failure condition (over or under specified limit), the time of failure, and the status of critical signals and I/Os at the time of failure.

The Lattice Platform Manager 2 features easy to implement Fault logging functions built in to the system. These include:

- Capture and record fault conditions within short span of fault occurrence.
- Capture and record Timestamp of the fault.
- Other system functions such as shutdown sequences continue to operate in parallel during fault recording.
- Record critical signals using built-in or external memory.
- No Need to power down the board to read the fault log.

The Lattice Diamond software provides the Platform Designer tool which is used to configure Platform Manager 2 device for system operation. The Fault log component of this tool allows easy configuration of the fault logging task with the desired features.

Platform Manager 2 provides following options to store Fault Log data:

1. Store Data in the ASC section
2. Store Data in the FPGA section
3. Store Data in External SPI Flash

If the data is stored in the FPGA section or an External SPI Flash, more flexibility is allowed with regards to how much information will be stored in the log. Through the Platform Designer tool, the designer can choose to select one of the above options to store Fault Log data and how much user data will be stored.

This document describes the built-in hardware features and the software tools used to configure the storing of Fault Log Data to the ASC, FPGA section or External SPI Flash memory.

Fault Logger Component

Figure 1 shows the Fault Log component from the Platform Designer tool. There are three choices at the top of the display to choose the type of fault logging desired. In addition, when the Full Featured Fault Log is selected there is an option to select where the fault log will be stored. The description of each choice is shown in Table 1.

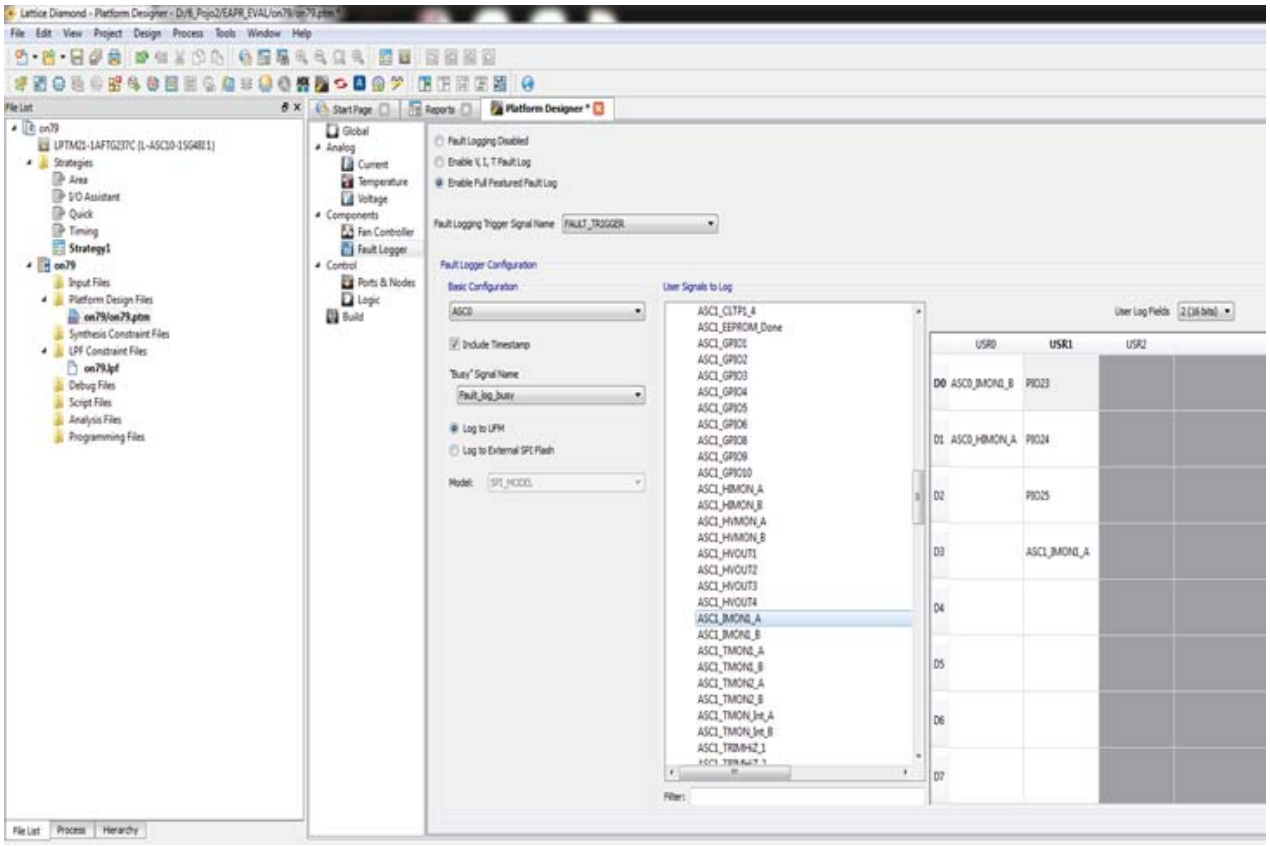
Table 1. Fault Logging Selections in the Platform Designer Tool

Primary Selection	Option	Description
Fault Logging Disabled	none	Fault Logging not active
Enable V, I, T Fault Log	Log to EEPROM Log to SRAM	Store Data in the ASC section
Enable Full Featured Fault Log/ User Tag Operation	Log to UFM	Store Data in the FPGA section
	Log to External SPI Flash	Store Data in External SPI Flash

The Full Featured Fault Log uses either an external SPI flash memory or the User Flash memory (UFM) within the Platform Manager 2 device or MachXO2 FPGA to store data. In Figure 1, the selection shown is “Enable Full Featured Fault Log/User Tag Operation Enabled” with the “Log to UFM” option.

When the Full Featured Fault Log is selected, it automatically instantiates LatticeMico8 IP (soft microcontroller) and Fault Logger IP in the FPGA portion of LPTM21 or the MachXO2 FPGA, to perform the operation of logging data to the UFM or External SPI flash.

Figure 1. Fault Logger Component of the Platform Designer Tool; Full Featured Interface



The designer can configure the following settings using the Fault Logger GUI interface within Platform Designer.

Fault Logging Trigger Signal Name
This is the signal which triggers the fault logging event. This signal can be an I/O or an internal node. The signal name can be selected from the LOGIC SIGNAL POOL list which is managed using the Ports and Nodes tab. You can add Nodes and rename Ports and Nodes at any time and then return to the fault Logging tab to change the selection. Renamed signals do not need to be re-selected the fault logger interface automatically synchronizes the name change.

The Trigger Signal is active high and must remain high for a minimum of 64us to reliably capture the fault condition. Typically the Trigger Signal is driven from the Logic section of Platform Designer in an Output or from a Boolean expression based on the design requirements.

Hardware Log Configuration

When the "Enable V, I, T Fault Log / User Tag Operation Disabled" is selected only the storage method and User Signals need to be configured.

Basic Configuration

Two choices are provided; Log to EEPROM and Log to SRAM. Logging to SRAM is useful in debugging during system development and Logging to EEPROM is useful for troubleshooting systems deployed in the field.

User Signals to Log

The Designer can specify in the Fault Logging GUI interface eight additional signals per ASC to log along with the ASC data. The available signals can be FPGA I/O or internal nodes listed in the Ports and Nodes tabs of the Platform Builder tool. Simply click and drag the signal from the LOGIC SIGNAL POOL list on the left to the bit location on the right.

Full Featured Fault Log Configuration

When the "Enable Full Featured Fault Log / User Tag Operation Enabled" is selected the interface changes to support configuration of the additional features.

Basic Configuration
Directly under “Basic Configuration”, a drop-down list is provided that is used to define the number of ASCs used to gather the data for the fault log. The default selection is "ASC0" and the number of selections depends on how many ASCs are used in the platform design. The available selections for a system with two ASCs are ASC0 and ‘Log ASC0 through ASC1’. The latter entry means both ASC0 and ASC1 data are logged. For a system with four ASCs, the available selections include ASC0, ASC0 through ASC1, ASC0 through ASC2, and ASC0 through ASC3.

For each ASC included, the system logs the respective VMON, IMON, and TMON comparator values, in addition to the GPIO and HVOUT values.

Include Timestamp
When selected, the fault log data includes a 32-bit timestamp, which provides a relative time for the event. The timestamp counts seconds from the Platform Manager 2 power up and its upper limit is 10 years, at which time the timer stops and holds the final value.

“Busy” Signal Name
This is a signal from the Fault Logger that is logic high during a fault log writing process. The user logic should not activate the trigger signal when the fault logger is busy. The signal name can be selected from the LOGIC SIGNAL POOL list which is managed using the Ports and Nodes tab.

Log to UFM

When selected, directs the IP to configure the fault log for the on-chip UFM memory of the FPGA section of the LPTM21 or the MachXO2 FPGA (when the MachXO2-ASC combination is used).

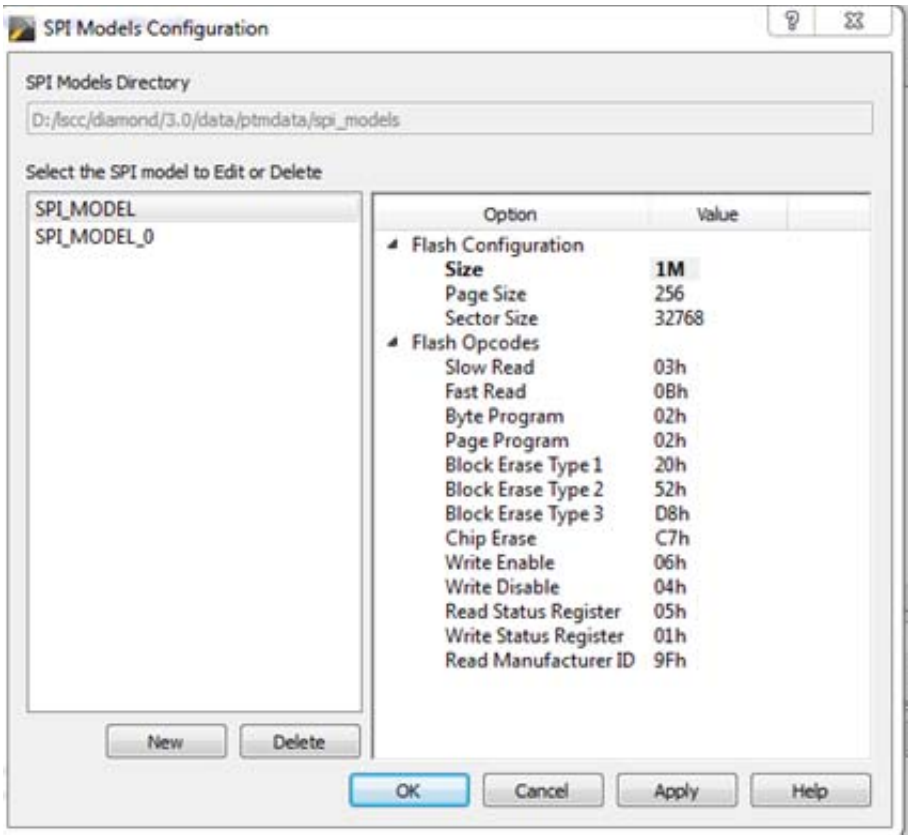
Log to External SPI Flash
When selected, directs the IP to configure the fault log for an external SPI flash using the SPI interface of the FPGA section of the LPTM21 or MachXO2.

The designer can create a SPI Model depending upon the Flash configuration and Op-Codes through Platform Designer tool as shown in Figure 2. This allows the use of different SPI Flash devices which may use different Op-Codes than what is included in the Platform Designer software.

User Signals to Log
The Designer can specify in the Fault Logging GUI interface additional signals to log with the ASC data. The available signals can be FPGA I/O or internal nodes listed in the Ports and Nodes tabs of the Platform Builder tool. Simply click and drag the signal from the LOGIC SIGNAL POOL list on the left to the bit location on the right.

User Log Fields
The Designer can specify how many bytes of data can be logged with the ASC data in the Fault Logging GUI interface. The default value is 1 byte but the designer can choose to store up to 4 bytes of data or zero bytes if desired when using the full featured fault logging.

Figure 2. Creating SPI Models



Fault Log Record Memory Map

The Fault Log data can be up to 66 Bytes of information depending upon number of external ASCs and User signal bytes selected as indicated in Table 2.

Table 2. Detailed Description of Fault Log Frame

Byte 0	Fault Flag (0x03C)
Byte 1	Fault Log length
Byte 2 to 8	ASC0 data
Byte 9 to 1	ASC1 (optional)
Byte 16 to 22	ASC2 (optional)
Byte 23 to 29	ASC3 (optional)
Byte 30 to 36	ASC4 (optional)
Byte 37 to 43	ASC5 (optional)
Byte 44 to 50	ASC6 (optional)
Byte 51 to 57	ASC7 (optional)
Byte 58 to 61	USER Signal from the “User Log Fields” in the GUI (optional)
Byte 62 to 65	Timer (if enabled)
Byte 66	Limiter 0x2A indicating end of page

Each ASC that is included in the fault log will require 7 bytes of data for the log. The User Signal data will be as long as the designer specifies in the GUI, from 0 to 4 bytes. The timer always requires 4 bytes of data when selected. The Fault Log length is used to indicate the number of bytes required for the selected configuration depending upon number of ASCs, User Signal bytes, and whether the Timer is enabled.

A Fault Flag, 0x03C, is written into the first byte of a new fault log to indicate start of a Fault logging Page. (The Flash memory will have a value of 0xFF in each byte when it is erased.) When the Platform Manager 2 is powered up, it will read the Flash memory contents looking for the flag value of 0x03C to see if any Fault log records are present in order to prevent over-writing previous fault records. The Platform Manager 2 will then begin writing new fault log records to the next available empty page of the memory. The system will always write the fault logs in sequential pages of memory.

When erasing memory, it is important to not leave erased pages in between valid records otherwise the Platform Manager 2 will attempt to over-write previous records resulting in corruption of both the new and old data records.

An example in Table 3 shows the fault log map generated for an LPTM21 with no external ASCs, four bytes of user data, and the Timer Enabled.

Table 3. Example of Fault Logging Map

BYTE	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
0	0	0	1	1	1	1	0	0
1	0	0	0	1	0	0	0	1
2	FL_FULL	FL_ACT	0	0	1	EE_DONE	1	AGOOD
3	AGOOD	RGPIO10	RGPIO9	RGPIO8	RGPIO7	RGPIO6	RGPIO5	RGPIO4
4	RGPIO3	RGPIO2	RGPIO1	RHVOUT4	RHVOUT3	RHVOUT2	RHVOUT1	IMON_1B
5	IMON_1A	HIMONB	HIMONA	HVMONB	HVMONA	VMON_9B	VMON_9A	VMON_8B
6	VMON_8A	VMON_7B	VMON_7A	VMON_6B	VMON_6A	VMON_5B	VMON_5A	VMON_4B
7	VMON_4A	VMON_3B	VMON_3A	VMON_2B	VMON_2A	VMON_1B	VMON_1A	TMON_2B
8	TMON_2A	TMON_1B	TMON_1A	TMonInt_B	TMonInt_A	1	0	1
9	USR0[7]	USR0[6]	USR0[5]	USR0[4]	USR0[3]	USR0[2]	USR0[1]	USR0[0]
10	USR1[7]	USR1[6]	USR1[5]	USR1[4]	USR1[3]	USR1[2]	USR1[1]	USR1[0]
11	USR2[7]	USR2[6]	USR2[5]	USR2[4]	USR2[3]	USR2[2]	USR2[1]	USR2[0]
12	USR3[7]	USR3[6]	USR3[5]	USR3[4]	USR3[3]	USR3[2]	USR3[1]	USR3[0]
13	Timer[31]	Timer[30]	Timer[29]	Timer[28]	Timer[27]	Timer[26]	Timer[25]	Timer[24]
14	Timer[23]	Timer[22]	Timer[21]	Timer[20]	Timer[19]	Timer[18]	Timer[17]	Timer[16]
15	Timer[15]	Timer[14]	Timer[13]	Timer[12]	Timer[11]	Timer[10]	Timer[9]	Timer[8]
16	Timer[7]	Timer[6]	Timer[5]	Timer[4]	Timer[3]	Timer[2]	Timer[1]	Timer[0]
17	0	0	1	0	1	0	1	0

Reading data from UFM or External Flash

The fault log data stored in the UFM can be read via the external slave SPI port or I2C primary port. For more details on accessing UFM via I2C refer to TN1246, Using User Flash Memory and Hardened Control Functions in MachXO2 Devices Reference Guide.

The fault log data stored in the external flash can be read via the SPI interface.

Summary

Using the Lattice Platform Manager 2 devices or the MachXO2 with companion ASC device(s) allows the designer to configure the fault log operation to meet their specific needs. The Fault Logger GUI component of the Platform Designer tool provides the designer an easy and flexible tool to configure the Fault Logging feature of Platform Manager 2 designs. It also supports storing large fault log data in the internal UFM or in an external SPI flash.

Related Literature

- DS1042, L-ASC10 Data Sheet
- DS1043, Platform Manager 2 Family Data Sheet
- TN1246, Using User Flash Memory and Hardened Control Functions in MachXO2 Devices Reference Guide
- Platform Designer Software Tutorial

Technical Support Assistance

e-mail: techsupport@latticesemi.com

Internet: www.latticesemi.com

Revision History

Date	Version	Change Summary
March 2015	1.1	Remove LPTM20 references.
December 2013	01.0	Initial release.