PMC 8001 Driver Functional Specification

Version 1.01

Authors: Tim Pearce

+408 421 4506

tpearce@xyratex.com

Date last saved: 12/9/2010 2:58:00 PM

Xyratex Confidential

Document change history

Date	Docn Revision	Change	Initials
20 July 10	0.01	Copied PRD in	TP
26 July 10	0.02	Minor review comments. Filled in WWN setting section plus debug message format. Added parameter summary at end.	TP
26 July 10	0.03	Added corollary to wwn parameters	TP
30 July 10	0.04	WWN update	TP
30 July 10	0.05	Typo in driver uninstall	TP
02 Aug 10	0.06	Updated WWN text	TP
02 Aug 10	0.07	Corrected folder names	TP
05 Aug 10	0.08	Added version reporting	TP
09 Sep 10	0.09	Updated for WWN params etc	TP
16 Sep 10	0.0a	Download now separated from source and requires 4 extra files	TP
14 Oct 10	0.0b	Download modified to allow for files and inbuilt download	TP
05 Nov 10	0.0c	Some updates on debug codes	TP
18 Nov 10	0.0d	Updated OS list	TP
19 Nov 10	1.00	Added review comments - minor typos	TP
09 Dec 10	1.01	Added hotplugging correction	TP

Table of contents

Table of Tables

Table of Figures

Introduction

Notation

This document uses the words MUST and SHOULD to refer to something that is mandatory. The words MAY, CAN, MIGHT and COULD refer to something that is optional.

SCSI is “big-endian” and so all fields should be assumed to be big-endian unless otherwise noted. Big-Endian means that the most significant (highest) byte of a number is presented first in the tables or communication streams.

Numbers that are in hexadecimal will be identified either by a table header saying they are in hex, or by the prefix 0x (e.g. 0x0C) or the suffix h (e.g. 0Ch). Any other numbers should be read as decimal. Binary will be suffixed with a lowercase b e.g. 011b. If it is necessary to put decimal numbers in and it is not clear they are decimal they will be followed by a d e.g. 12d.

Some of the tables have items where the description is followed by a number in brackets.

When this occurs it indicates that the field has a constant value which is the number shown in brackets. It should be noted that even though the field has a fixed value, client programs should still read and interpret it as though it was a variable value because future modifications of the firmware cannot guarantee this will remain constant.

Folders and directories

Note: /l..i/ is a fictitious directory that is used in this document as a shorthand however it is in fact usually of the format

```
/lib/modules/<kernel-version>/kernel/drivers/scsi
```

Where `<kernel-version>` is the kernel version for which the driver was built.

Abbreviations

The following abbreviations may be used in this document.

BMC	Baseboard Motherboard Controller
CDB	Command Data Block – SCSI message that contains a SCSI command sent to target
DFM	Design For Manufacturing
DFT	Design For Testability
EBOD	Expanded Bunch Of Disks
FC	Fibre Channel
FRU	Field Replaceable Unit
GA	General Availability
GEM	Genesis Enclosure Management
HA	High Availability
HBA	Host Bus Adapter
ICL	Inter Controller Link – PCIe based communication between CPUs on different motherboards
IOPs	I/Os Per Second
IPMI	Intelligent Platform Management Interface
iSCSI	Internet SCSI
JBOD	Just a Bunch Of Disks
LED	Light Emitting Diode
ODK	OEM Developer Kit
PCM	Power/Cooling Module
PCI	Peripheral Component Interconnect bus – Used for CPU->IO devices
PCIe	PCI express - High speed PCI using serial communications
PDF	Portable Document Format
PMC	Shorthand for the company name PMC-Sierra
PSMI	Power Supply Management Interface
PSU	Power Supply Unit
RHEL	RedHat Enterprise Linux (RedHat’s Linux operating system distribution)
SAS	Serial Attached SCSI
SATA	Serial ATA
SBB	Storage Bridge Bay
SBOD	Switched Bunch Of Disks
SCSI	Small Computer System Interface
SES	SCSI Enclosure Services
SMP	SAS Management Protocol
SPC	SAS Protocol Controller – SAS HBA chip
SSD	Solid State Drive
STP	SATA Tunneling Protocol – SATA over SAS
VPD	Vital Product Data
XLD	Xyratex Linux Distribution

Overview

The PMC Device Driver is a standard driver designed for Linux systems. As such it fits into a Linux environment in much the same way as any other SCSI based driver.

A phased approach is being taken to development and release of the driver. Not all function in this document is available, or may not be totally available, for each phase. This document assumes nothing about the phases and you should refer to the schedule and/or PRD for more detail on release contents.

The driver fits in as a standard Linux device driver for a SCSI HBA such as in the following. Our implementation is the *pm8001 driver* block:

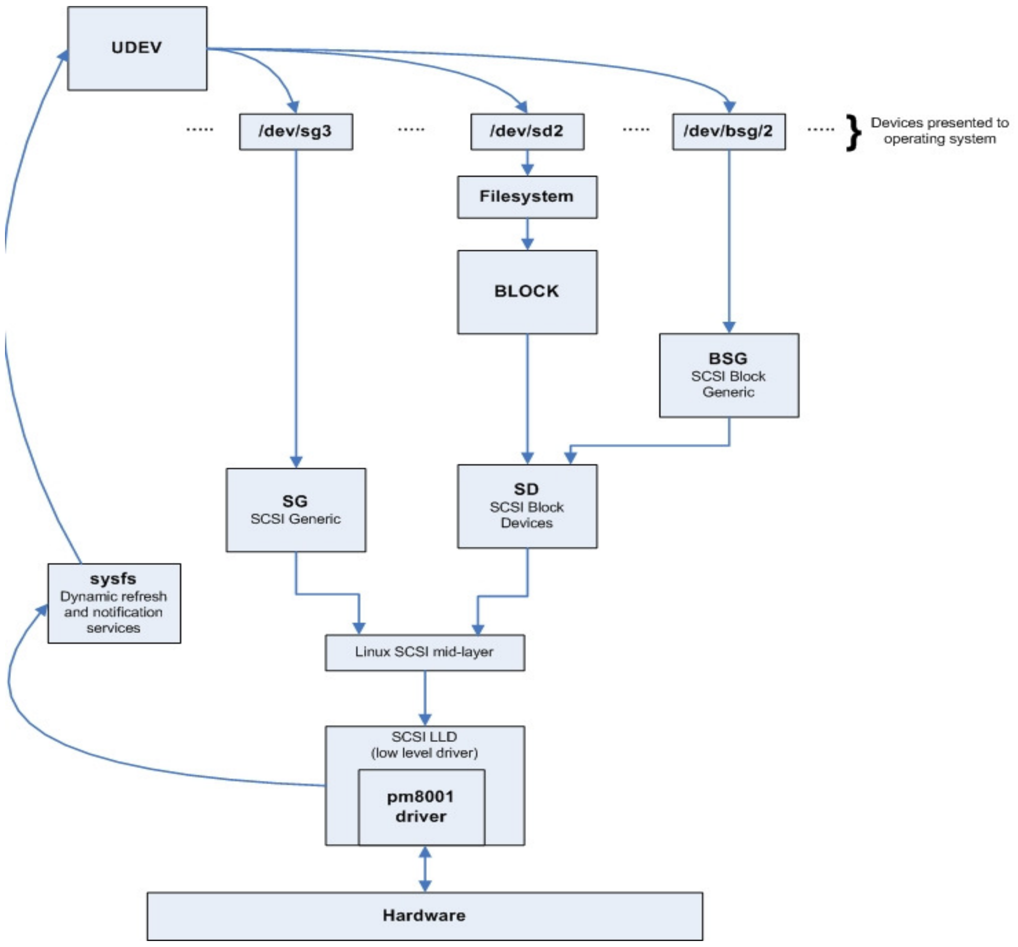
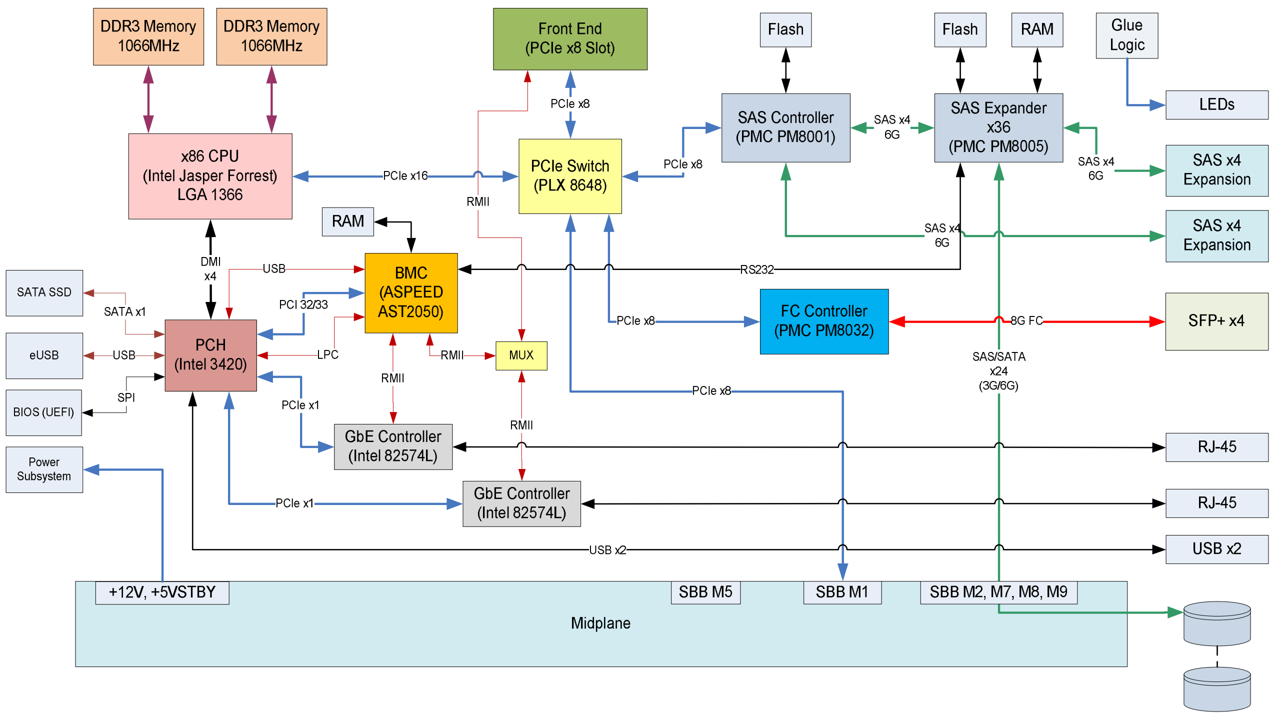


Figure 1 - Basic Linux block diagram

Note that some terms and the exact layout changes from Linux system to system and across revisions of Linux too.

Stingray Hardware Architecture

Stingray is the code name for a Xyratex owned motherboard. It has a basic block layout as shown below.



Design goals

In all designs there are conflicting design considerations. This section outlines the priorities in order for making design decisions

1) Throughput that is as close to line rate as possible. There are some points to note
    a. How busy Linux is with other software will affect this
    b. How many drives are targeted will affect this
    c. The drive types will affect this
    d. The mix of reads and writes and the block sizes will affect this
    e. The use of SAS wide-ports will affect this
    f. Other IO tasks in Linux will affect this
    g. What expansions are included will affect this
    h. Plus more

2) The driver must look and work exactly like any other low level driver under the Linux SCSI driver interface. It should seamless integrate into the Linux system for which it is built and not require bespoke actions or interfaces

3) It needs to be customizable for OEMs e.g. changing driver names etc

4) It must be debuggable when set to the correct mode.

References

[1] PMC Driver User Guide.doc

[2] Stingray PMC Device DriverV3_Rev_g.doc

Support, limitations and caveats

Enclosures

Enclosure is limited to those that support Stingray

- OneStor 2U-12 3.5"
- OneStor 2U-24 2.5"
- OneStor 4U-24 3.5"

SBB (Storage Bridge Bay) support is mandatory

Motherboards

Stingray Generic (non-IBM customers)

- With PMC 8G FC Controller and SFP+ Cages
- Without PMC 8G FC Controller and SFP+ Cages

Stingray tailored for specific customers

CPUs

The driver is specifically targeted to Stingray and any derivative platforms.

- Support x86 Intel processors (EC3539 4-core)
- Support 64 bit only but be portable to 32 bit if needed

SAS Controller

PMC-Sierra PM8001 Rev C SPC (SAS Protocol Controller) using SAS 1.1 and SAS 2.
PMC-Sierra SPC firmware version 01.10.05.00.

Disk Drives

- Seagate "Muskie"/Constellation 3.5" SAS 1TB and 2TB
- Seagate "Eagle"/X15-7 3.5" SAS 300G, 450G and 600G
- Seagate "Dragonfly"/Constellation 2.5" 500G
- Samsung SS805 SSD via Acorn Dongle rev 5+

Operating Systems

All are 64 bit.

- CentOS 5.4 kernel 2.6.18
- CentOS 5.4 kernel 2.6.18-164
- CentOS 5.5 kernel 2.6.18-194

- RedHat Enterprise Linux (RHEL) 5.4 kernel 2.6.18-164
- RedHat Enterprise Linux (RHEL) 6.0 kernel 2.6.32
- RedHat Enterprise Linux (RHEL) 6.0 kernel 2.6.33

Design Limitations and Notes

There is currently no defacto standard support in Linux for SMP pass through. This means an SMP implementation will try to conform to the latest most likely standard for user issued SMP passthru. This will be an unsupported interface in early releases.

This driver will not support pure SATA drives that require STP (SATA Transport Protocol). All SATA drives will have to be used with a dongle.

The driver supports just a single SPC per system/motherboard.

The driver supports both SPCs with attached flash and flash-less SPCs. On startup the driver will detect lack of firmware and will download firmware automatically if needed.

Expansion enclosures support and SAS-2 will not be needed in the first 2 releases.

SCSI Support

All SAS supported SCSI commands are supported as are all block sizes using standard SCSI interfaces to the devices the driver presents. The SPC and the subsequent expanders and targets may limit these however the driver does not. Equally it does not restrict the type and format of standard SCSI responses however other hardware might.

SES is supported just as for any SCSI enclosure. On Stingray SES support is provided by a virtual SES processor target that is in addition to the drives.

The maximum number of addressable drives that this driver supports per SPC is 256.
The maximum number of outstanding/queued commands per drive is 8.

WWN Support and modification

A root WWN is stored by the SPC chip itself. This is a 64 bit WWN. And is used to generate all other WWNs needed by the SPC.The WWN is persisted by the SPC in VPD (Vital product Data). Once the WWN is set it is always persisted.

There are two modes for the WWN. These define whether you want a singled 8x wideport into the SPC or dual 4x ports.

You need to use different parameters depending upon which mode you need

To set the VPD you can either

a) It can be hard coded into SPC firmware (highly non-recommended)
b) Load a whole VPD image when programming the VPD in manufacturing. This is likely supportable by manufacturing only.
c) It can be specified as a parameter when loading the driver. Once set it remains persisted. This is supportable as a method of field change although it is desirable NOT to have all customers changing WWNs so the parameter should not be in any user guides – just available for FAEs and OEMs

To program the root WWN uninstall the driver and then re-install it using one of the following two mutually exclusive parameters based upon the desired wideport mode:

```
 insmod /l..i/pm8001.ko param_wwn_by4=0x009911ff33e21a00
```
or
```
 insmod /l..i/pm8001.ko param_wwn_by8=0x009911ff33e21a00
```

The full 64 bit hex value (16 hex digits) should be used with no spaces, a mandatory 0x as the leading 2 characters and also using hex digits (0-9a-f) only in lower case. Note that if you omit the 0x at the beginning the value will be interpreted as decimal and a translation to hex will be done which means you may not get the values set that you desired.

These parameters cause 2 WWNs to be persisted in both cases for each half of the wideport – however the two WWNs are the same for a 8x so it appears as a single port. The VPD storage for each is as follows:

4x
```
0x009911ff33e21a00
0x009911ff33e21a01
```

8x
```
0x009911ff33e21a00
0x009911ff33e21a00
```

Note that these are root WWNs. All other WWNs used in the SPC are derived from these so an appropriate gap between WWNs assigned during manufacturing to avoid clashes is strongly recommended. A gap of 16 is recommended which implies leaving the last digit as zero in the hex field.

Note the WWN setting parameters (and other methods) should only be used when forcing the WWN to change and not on every driver load in order to cut down on VPD write counts. The programmed value is held in EEPROM and persisted across reboots.

The WWNs are held in EEPROM at offset 0x00 of the SPC EEPROM user space section. They are 64 bits and stored in little-Endian mode.

Pre-Requisites

On all Linux systems libsas (part of Linux) must be installed before installing or using the PMC 8001 driver. If not already installed libsas can be installed as follows:

```
      modprobe libsas
```

In order to ensure that firmware can be downloaded you must ensure that the 4 firmware files supplied by the driver have been copied to /lib/firmware, otherwise the driver will use its default inbuilt firmware for download.

Logging and Debugging

The driver will support a rudimentary debug mode that allows for diagnosis of issues.

To turn on debug mode requires de-loading the driver and re-loading it with the appropriate debug parameter specified.

The Driver has 8 categories of debug messages as follows in order to allow you to decide the quantity of output you require. Note that enabling debug may affect performance.

Category	Bit Field	Hex Format
Error Message Logging	0	0x01
Driver Initialization Logging	1*	0x02
Discovery Layer Logging	2	0x04
I/O Path Logging	3	0x08
libsas EH Function Logging	4	0x10
IOCTL Message Logging	5	0x20
Misc Message Logging group 1	6	0x40
Misc Message Logging group 2	7	0x80

Table 1 - Debugging Categories

* default

The values specified in the table above get used when the driver is installed.

First, if your driver is already installed in non-debugging mode uninstall it by using the following command:

```
rmmod /l..i/pm8001.ko
```

Note: /l..i/ is a fictitious directory that is used in this document as a shorthand however it is in fact usually of the format

```
/lib/modules/<kernel-version>/kernel/drivers/scsi
```

Where `<kernel-version>` is the kernel version for which the driver was built.

Now insure that klogd is running as follows on the Bit Field terminal

```
ps -ef
```

If klogd is not listed in the output from the above then

```
klogd&
```

Once this is done you need to install the driver in debugging mode. Start by ORing the fields in the table above that match what you wish to view. For instance

```
      Bit 0 + Bit 2 + Bit 5
=     0x01  + 0x04  + 0x20
=     0x25
```

Then install the driver in debug mode using the value calculated above in the following command

```
    insmod /dev/pm8001.ko pm8001_logging_level=0x25

(replace 0x25 with whatever your Category combination
 was calculated to be)
```

All output will be sent to the standard Linux kernel log in /var/log/messages.

As implied, unless the parameter is specified at driver load then debug/logging is turned off.

The format of the output is as follows:

```
      function_name: message(s)
```

The function_name is starting with “pm8001_ “ for the device driver for all related driver messages.

.

Handling SBB and Drive Hotplugs

Currently the Stingray hardware has no support for live hotplugs/removals of the Stingray SBB. This means the driver does not need to support Stingray motherboard hot inserts/removals.

Drive hotplug is supported. As shown in "Figure 1 - Basic Linux block diagram" on page 6, Linux has a feedback mechanism for drive hotplug/hot removals that updates the list of devices (e.g. /dev/sg2) that are available to interface with. These can be interfaced with in the standard Linux way with opens and writes/reads.

There is no message passing to any other layer when a drive is inserted/removed other than a refresh of this device list in standard Linux.

SAS does provide some alternatives to this such as transmitting a Broadcast(Change) notification however transmission of these broadcasts to a client application will not be supported in this driver.

It is also possible to force Linux to refresh its device list manually. This changes from Linux to Linux however an example is as follows:

```
1. Determine what is the host number for the HBA:
   ls /sys/class/sas_host/
```

(Values such as host1 or host2 will be listed for you. Refer to them as host# for the purposes of the next step.)

```
2. Ask the HBA to rescan the SCSI devices on that HBA:
   echo - - - > /sys/class/scsi_host/host#/scan
```

(The wildcards "- - -" will force Linux to look at every channel, every target, every lun.)

In general, manual rescan should not be needed although it make take a few seconds for new or removed devices to update the device list in Linux.

When hotplugging drives you must allow time for the whole Linux system stack to settle (i.e. to have recognized and finished handling the drive state change) between individual removals and insertions. This may be several seconds in some systems.

Firmware Download

As previously mentioned the driver supports SPCs without flash. This implies doing a firmware download to the SPC when the device driver is first started up.

There is no mechanism to update the firmware after startup except by reloading drivers with appropriate firmware in them.

The driver will need four files to download which will normally be supplied by the chip vendor.

These files will need to be placed into /lib/firmware in order to be located and downloaded correctly.

If you wish to change or upgrade the firmware you face having to restart the driver to reset its pointers and clean up correctly. With that in mind to upgrade firmware after boot simply replace the firmware files and then unload the driver and reload it again.

This works for flashed and flash-less systems.

If you do not provide the files in /lib/firmware the driver will use its inbuilt firmware to download. It only does this if the first file it looks for to download is inaccessible or missing. If this is the case it switches to downloading the inbuilt code. This implies that providing the firmware in external files is a good way to upgrade firmware after installation, and deleting them is a good way to downgrade back to factory defaults.

Summary of Driver Startup/Installation Parameters

Parameters can be combined and order is not important. At least one space should be used to separate them and the parameter itself contains no spaces, not even on either side of the equals (=) sign. All parameters and their arguments are lowercase.

1) No parameters

   Install with debug turned off and without setting the root WWN

2) pm8001_logging_level=yyyy

   Install with debug turned on. The debug setting specified as yyyy above should be of the format 0xnn when 0 and x are specific characters/letters you must use and the latter two digits – nn in 0xnn – are two hex digits in lower case such as 0x12. These digits represent the debug settings flags shown in “Logging and Debugging” on page 13. The default is 0x01.

3) param_wwn_by4=0x123456789abcdef0
   or
   param_wwn_by8=0x123456789abcdef0

   At installation set (and persist) the SPC root WWNs as 64bit (16 hex digit) numbers specified in the argument. Once set these do not need to be specified again unless the WWN needs to be changed. The WWNs are stored in the SPC VPD EEPROM. The leading 0x is mandatory to be part of the argument as-is.

   There are two modes for the WWN. These define whether you want a singled 8x wideport into the SPC or dual 4x ports.

   These parameters cause 2 WWNs to be persisted in both cases for each half of the wideport – however the two WWNs used are the same for a 8x so it appears as a single port. The VPD storage for each is as follows:

   4x
   ```
   0x009911ff33e21a00
   0x009911ff33e21a01
   ```

   8x
   ```
   0x009911ff33e21a00
   0x009911ff33e21a00
   ```

   Note that these are root WWNs. All other WWNs used in the SPC are derived from these so an appropriate gap between WWNs assigned during manufacturing to avoid clashes is strongly recommended. A gap of 16 is recommended which implies leaving the last digit as zero in the hex field.

This parameter is only to be used when changing the WWNs and not on every driver load as the value set is persisted in EEPROM. When changing WWN ensure you are using one that is correctly assigned to be unique globally.

Do not specify this parameter on every driver startup/load otherwise you will reduce the number of EEPROM writes that remain and possibly exhaust them.

Driver version reporting

It is important to be able to determine the running version of a driver

There are standard ways in Linux to get this information

The first is modinfo e.g.

```
    modinfo pm8001.ko
```

In addition the kernel log contains information about which driver was started such as the entry below

```
Jul 24 11:53:10 CentOS-Stingray kernel: pm8001 0000:04:00.0: pm8001:
driver version 0.1.36
```