cPCI-6S10 Series

6U CompactPCI Switch Blade
User's Manual

Manual Rev.	1.0
Revision Date:	September 3, 2018
Part No.:	50-15099-1000

Leading **EDGE COMPUTING**

Revision History

Revision	Release Date	Description of Change(s)
1.0	03/09/2018	Initial release

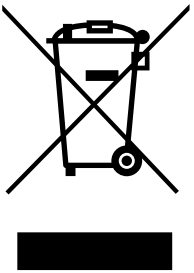
Preface

Disclaimer

The information in this document is subject to change without prior notice in order to improve reliability, design, and function and does not represent a commitment on the part of the manufacturer.

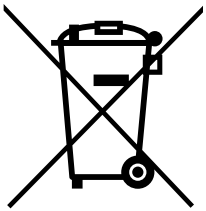
In no event will the manufacturer be liable for direct, indirect, special, incidental, or consequential damages arising out of the use or inability to use the product or documentation, even if advised of the possibility of such damages.

Environmental Responsibility



ADLINK is committed to fulfill its social responsibility to global environmental preservation through compliance with the European Union's Restriction of Hazardous Substances (RoHS) directive and Waste Electrical and Electronic Equipment (WEEE) directive. Environmental protection is a top priority for ADLINK. We have enforced measures to ensure that our products, manufacturing processes, components, and raw materials have as little impact on the environment as possible. When products are at their end of life, our customers are encouraged to dispose of them in accordance with the product disposal and/or recovery programs prescribed by their nation or company.

Battery Labels (for products with battery)



California Proposition 65 Warning

WARNING: This product can expose you to chemicals including acrylamide, arsenic, benzene, cadmium, Tris(1,3-dichloro-2-propyl)phosphate (TDCPP), 1,4-Dioxane, formaldehyde, lead, DEHP, styrene, DINP, BBP, PVC, and vinyl materials, which are known to the State of California to cause cancer, and acrylamide, benzene, cadmium, lead, mercury, phthalates, toluene, DEHP, DIDP, DnHP, DBP, BBP, PVC, and vinyl materials, which are known to the State of California to cause birth defects or other reproductive harm. For more information go to www.P65Warnings.ca.gov**.**

Trademarks

Product names mentioned herein are used for identification purposes only and may be trademarks and/or registered trademarks of their respective companies.

Conventions

Take note of the following conventions used throughout this manual to make sure that users perform certain tasks and instructions properly.

Additional information, aids, and tips that help users perform tasks.

Information to prevent ***minor*** physical injury, component damage, data loss, and/or program corruption when trying to complete a task.

Information to prevent ***serious*** physical injury, component damage, data loss, and/or program corruption when trying to complete a specific task.

Table of Contents

List of Tables

This page intentionally left blank.

List of Figures

This page intentionally left blank.

1 Overview

The ADLINK cPCI-6S10 is a fully managed CompactPCI Gigabit Ethernet switch blade supporting up to twenty four 1GbE ports and two 10GbE SPF+ uplink ports. On the front panel are three GbE ports, two 10GbE SFP+ uplink ports, one COM port and one 10/100 RJ-45 management port. Twenty GbE ports are routed to rear I/O and an IPMI interface is provided to monitor and control system health.

The ADLINK cPCI-6S10 integrates Broadcom BCM56150 switch silicon along with an ARM Cortex-A9 processor and one Broadcom BCM54685 Octal port 1000BASE-T PHY. The cPCI-6S10 is ideal for CompactPCI platform adopters who require high speed and high bandwidth data transport interconnects for packet switching management.

The ADLINK cPCI-6S10 supports ADLINK PacketManager, a software suite with an extensive feature set and integration capabilities that enables powerful networking functionality for the Base Interface. It also provides comprehensive device management capabilities for network administrators.

1.1 Block Diagram

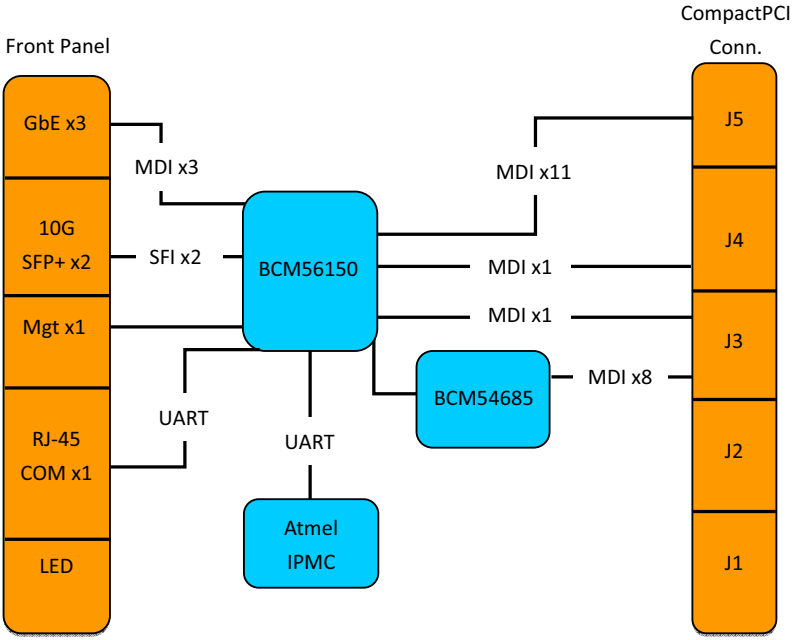


Figure 1-1: cPCI-6S10 Series Block Diagram

1.2 Package Contents

The cPCI-6S10 is packaged with the components listed below (RTMs and adapter kits are optional). If any of the items in the contents list are missing or damaged, retain the shipping carton and packing material and contact the dealer for inspection. Please obtain authorization before returning any product to ADLINK. The packing contents of non-standard configurations may vary depending on customer requests.

- cPCI-6S10 CompactPCI switch blade
- RJ-45 to DB-9 adapter for UART port

The contents of non-standard cPCI-6S10 Series configurations may vary depending on customer requests.

This product must be protected from static discharge and physical shock. Never remove any of the components except at a static-free workstation. Use the anti-static bag shipped with the product when putting the board on a surface. Wear an anti-static wrist strap properly grounded on one of the system's ESD ground jacks when installing or servicing system components.

This page intentionally left blank.

2 Specifications & Board Interfaces

2.1 cPCI-6S10 Specifications

Standards and Interfaces	
CompactPCI Standard	PICMG 2.0 CompactPCI Rev. 3.0 PICMG 2.9 System Management Bus Rev. 1.0 PICMG 2.16 Packet Switching Backplane Rev.1.0
Switch Fabric, PHY	• Broadcom BCM56150 24-port GbE switch with two 10G SFP+ uplink ports • Broadcom BCM54685 Octal-port 10/100/1000BASE-T PHY • Broadcom BCM5221 10/100BASE-TX management port
Networking	• 3x 10/100/1000BASE-T ports to front panel (BCM56150) • 20x 10/100/1000BASE-T ports to rear (BCM56150, 9x to J3, 11x to J5) • 1x inter-switch link 1000BASE-T • 2x 10G SFP+ ports for uplink through BCM56150 • 1x 10/100BASE-TX for management
Front Panel IO	3x 10/100/1000BASE-T RJ-45 ports 2x 10G SFP+ interfaces for uplink interface 1x 10/100 RJ-45 management port 1x UART port via RJ-45
Rear IO	20x 10/100/1000BASE-T ports to J3 and J5
Mechanical & Environmental	
Dimensions	233.35mm x 160mm (L x W), 6U 4HP single slot
Operating Temp.	Standard: 0°C to 60°C ETT version upon request
Storage Temp.	-50°C to 100°C
Humidity	95% non-condensing
Shock	15G peak-to-peak, 11ms duration, non-operating
Vibration	Non-operating: 2Grms, 5 to 500 Hz, each axis
Compliance	CE, FCC Class A
Power Consumption (under Linux)	• Idle (w/o RTM): 12.2W • Full loading (w/o RTM): 15W • Idle (w/ RTM): 15.5W • Full loading (w/ RTM): 20W

Table 2-1: cPCI-6S10 Specifications


NOTE:

Specifications are subject to change without prior notice.

2.2 Board Layout

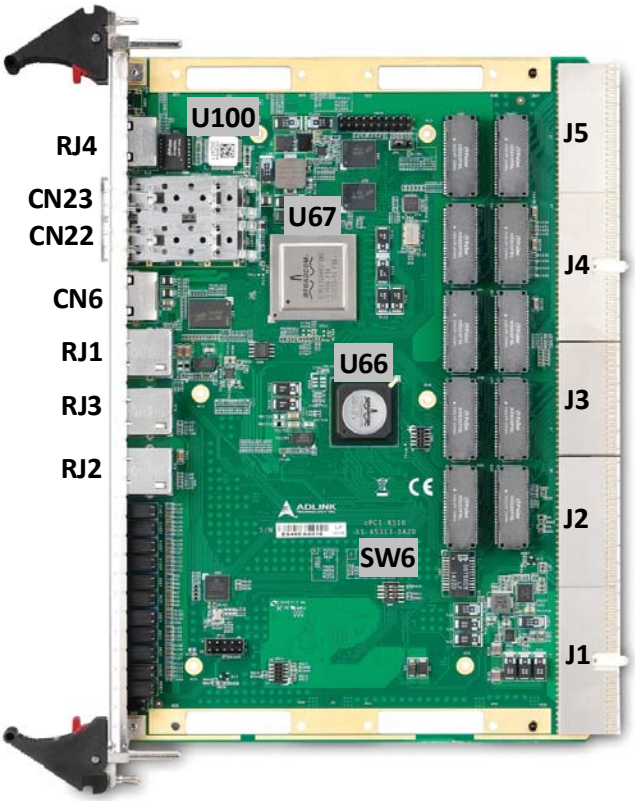
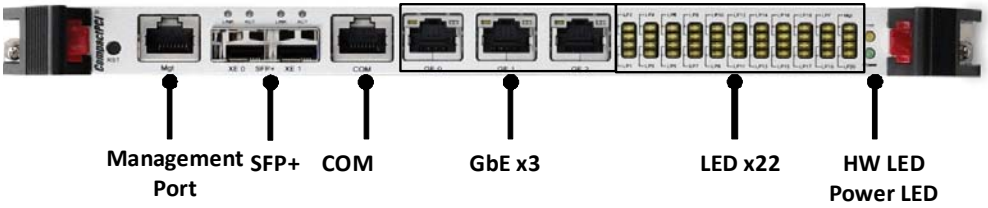


Figure 2-1: cPCI-6S10 Board Layout

U100	BCM5221	**RJ1-3**	10/100/1000 GbE
U67	BCM56150	**RJ4**	10/100 Mgmt GbE
U66	BCM54685	**CN6**	UART Port
J1-J5	CompactPCI connectors	**CN22-23**	SFP+ Uplink ports
SW6	Standalone mode switch		

2.3 Front Panel Layout



The following section describes the behavior of the LEDs on the front panel.

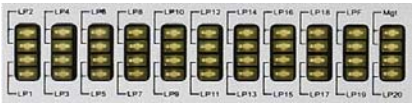
Power LED

Power LED (Green)	Status
On	Power status normal
Off	Power off

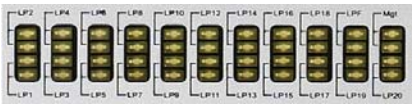
HW LED (Hardware Health)

HW LED (Yellow)	Status
On	POST and sensor status of IPMC is normal
Blinking	TBD
Off	POST and sensor status of IPMC is abnormal

Rear IO GbE Status LEDs (LP1 - LP20)

Upper LED On: Link up Off: Link down **Lower LED** Blink: Packet Activity	

SFP+ Status LEDs (XE 0 - XE 1)

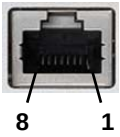
Link LED (Green) ON: Link up OFF: Link down **Act LED (Orange)** Blink: Packet Activity	

2.4 Connector Pinouts

UART COM Connector (CN6)

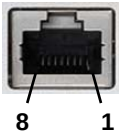
Set SW21 to IPMI or BCM56150 UART MUX debug port

Pin #	10BASE-T/ 100BASE-TX
1	NC
2	NC
3	NC
4	IPM_DBG_TX
5	IPM_DBG_RX
6	GND
7	NC
8	NC



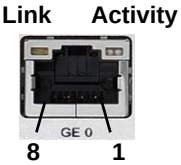
RJ-45 10/100BASE-T Mgmt. Ethernet Connector (RJ4)

Pin #	10BASE-T/ 100BASE-TX
1	NC
2	NC
3	NC
4	IPM_DBG_TX
5	IPM_DBG_RX
6	GND
7	NC
8	NC



RJ-45 Gigabit Ethernet Connectors ((RJ1/2/3)

Pin #	1000BASE-T
1	LAN_TX0+
2	LAN_TX0-
3	LAN_TX1+
4	LAN_TX2+
5	LAN_TX2-
6	LAN_TX1-
7	LAN_TX3+
8	LAN_TX3+

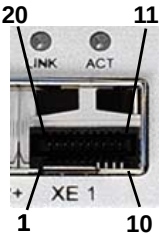


GbE Status LEDs

Left LED
On: Link up
Off: Link down
Right LED:
Blink: Packet Activity

SFP+ 10Gigabit Ethernet Connector (CN22/23)

Pin #	1000BASE-T
1	GND
2	SFP_TX_FAULT
3	SFP_TX_DISABLE
4	SFP_SDA
5	SFP_SCL
6	SFP_MOD_ABS
7	GND
8	SFP_RX_LOS
9	GND
10	GND
11	GND
12	SRDS_RX-N
13	SRDS_RX-P
14	GND
15	SFP_V3P3_R
16	SFP_V3P3_T
17	GND
18	SRDS_TX-P
19	SRDS_TX-N
20	GND



SFP+ 10GbE Status LEDs

Link LED: Yellow (Left)
Act LED: Green (Right)

CompactPCI J1 Connector Pin Assignment

Pin	Z	A	B	C	D	E	F
25	GND	P5V	NC	CPCI_ENUM_L	P3V3	P5V	GND
24	GND	NC	P5V	CPCI_VIO	NC	NC	GND
23	GND	P3V3	NC	NC	P5V	NC	GND
22	GND	NC	GND	P3V3	NC	NC	GND
21	GND	P3V3	NC	NC	GND	NC	GND
20	GND	NC	GND	CPCI_VIO	NC	NC	GND
19	GND	P3V3	NC	NC	GND	NC	GND
18	GND	NC	GND	P3V3	NC	NC	GND
17	GND	P3V3	IPMB_SCL	IPMB_SDA	GND	NC	GND
16	GND	NC	GND	CPCI_VIO	NC	NC	GND
15	GND	P3V3	NC	NC	CPCI_BDSEL_L	NC	GND
12-14	GND	Key Area					Key
11	GND	NC	NC	NC	GND	NC	GND
10	GND	NC	GND	P3V3	NC	NC	GND
9	GND	NC	GND	NC	GND	NC	GND
8	GND	NC	GND	CPCI_VIO	NC	NC	GND
7	GND	NC	NC	NC	GND	NC	GND
6	GND	NC	CPCI_PRESENT_L	P3V3	NC	NC	GND
5	GND	NC	NC	NC	GND	NC	GND
4	GND	P5V_IPMB	CPCI_HEALTHY-L	CPCI_VIO	NC	NC	GND
3	GND	NC	NC	NC	P5V	NC	GND
2	GND	NC	P5V	NC	NC	NC	GND
1	GND	P5V	N12V	NC	P12V	P5V	GND

CompactPCI J2 Connector Pin Assignment

Pin	Z	A	B	C	D	E	F
22	GND	GA4	GA3	GA2	GA1	GA0	GND
21	GND	NC	GND	NC	NC	NC	GND
20	GND	NC	GND	NC	GND	NC	GND
19	GND	GND	GND	IPMB_SDA	IPMB_SCL	NC	GND
18	GND	NC	NC	NC	GND	NC	GND
17	GND	NC	GND	CHASS_RST_L	NC	NC	GND
16	GND	NC	NC	CPCI_DEG_L	GND	NC	GND
15	GND	NC	GND	CPCI_FAL_L	NC	NC	GND
14	GND	NC	NC	NC	GND	NC	GND
13	GND	NC	GND	CPCI_VIO	NC	NC	GND
12	GND	NC	NC	NC	GND	NC	GND
11	GND	NC	GND	CPCI_VIO	NC	NC	GND
10	GND	NC	NC	NC	GND	NC	GND
9	GND	NC	GND	CPCI_VIO	NC	NC	GND
8	GND	NC	NC	NC	GND	NC	GND
7	GND	NC	GND	CPCI_VIO	NC	NC	GND
6	GND	NC	NC	NC	GND	NC	GND
5	GND	NC	GND	CPCI_VIO	NC	NC	GND
4	GND	CPCI_VIO	NC	NC	GND	NC	GND
3	GND	NC	GND	NC	NC	NC	GND
2	GND	NC	NC	SYSEN-L	NC	NC	GND
1	GND	NC	GND	NC	NC	NC	GND

CompactPCI J3 Pin Assignment

Pin	Z	A	B	C	D	E	F
19	GND	SGA4	SGA3	SGA2	SGA1	SGA0	GND
18	GND	LAN_A_LPFp	LAN_A_LPFn	GND	LAN_C_LPFp	LAN_C_LPFn	GND
17	GND	LAN_B_LPFp	LAN_B_LPFn	GND	LAN_D_LPFp	LAN_D_LPFn	GND
16	GND	LAN_A_8P	LAN_A_8N	GND	LAN_C_8P	LAN_C_8N	GND
15	GND	LAN_B_8P	LAN_B_8N	GND	LAN_D_8P	LAN_D_8N	GND
14	GND	LAN_A_7P	LAN_A_7N	GND	LAN_C_7P	LAN_C_7N	GND
13	GND	LAN_B_7P	LAN_B_7N	GND	LAN_D_7P	LAN_D_7N	GND
12	GND	LAN_A_6P	LAN_A_6N	GND	LAN_C_6P	LAN_C_6N	GND
11	GND	LAN_B_6P	LAN_B_6N	GND	LAN_D_6P	LAN_D_6N	GND
10	GND	LAN_A_5P	LAN_A_5N	GND	LAN_C_5P	LAN_C_5N	GND
9	GND	LAN_B_5P	LAN_B_5N	GND	LAN_D_5P	LAN_D_5N	GND
8	GND	LAN_A_4P	LAN_A_4N	GND	LAN_C_4P	LAN_C_4N	GND
7	GND	LAN_B_4P	LAN_B_4N	GND	LAN_D_4P	LAN_D_4N	GND
6	GND	LAN_A_3P	LAN_A_3N	GND	LAN_C_3P	LAN_C_3N	GND
5	GND	LAN_B_3P	LAN_B_3N	GND	LAN_D_3P	LAN_D_3N	GND
4	GND	LAN_A_2P	LAN_A_2N	GND	LAN_C_2P	LAN_C_2N	GND
3	GND	LAN_B_2P	LAN_B_2N	GND	LAN_D_2P	LAN_D_2N	GND
2	GND	LAN_A_1P	LAN_A_1N	GND	LAN_C_1P	LAN_C_1N	GND
1	GND	LAN_B_1P	LAN_B_1N	GND	LAN_D_1P	LAN_D_1N	GND

CompactPCI J4 Connector Pin Assignment

Pin	Z	A	B	C	D	E	F
25	GND	NC	NC	NC	NC	NC	GND
24	GND	NC	NC	NC	NC	NC	GND
23	GND	NC	NC	NC	NC	NC	GND
22	GND	CMM_J4_TX0P	CMM_J4_TX0N	NC	CMM_J4_RX0P	CMM_J4_RX0N	GND
21	GND	NC	NC	NC	NC	NC	GND
20	GND	NC	NC	NC	NC	NC	GND
19	GND	NC	NC	NC	NC	NC	GND
18	GND	NC	NC	NC	IPMB_SCL	IPMB_SDA	GND
17	GND	NC	NC	NC	IPMB_SCL	IPMB_SDA	GND
16	GND	NC	NC	NC	NC	NC	GND
15	GND	NC	NC	NC	NC	NC	GND
12-14	GND	Key Area					Key
11	GND	NC	NC	NC	NC	NC	GND
10	GND	NC	NC	NC	NC	NC	GND
9	GND	NC	NC	NC	NC	NC	GND
8	GND	NC	NC	NC	NC	NC	GND
7	GND	NC	NC	NC	NC	NC	GND
6	GND	NC	NC	NC	NC	NC	GND
5	GND	NC	NC	NC	NC	NC	GND
4	GND	NC	NC	NC	NC	NC	GND
3	GND	NC	NC	NC	NC	NC	GND
2	GND	NC	NC	NC	NC	NC	GND
1	GND	NC	NC	NC	NC	NC	GND

CompactPCI J5 Pin Assignment

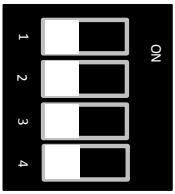
Pin	Z	A	B	C	D	E	F
22	GND	LAN_A_19P	LAN_A_19N	GND	LAN_C_19P	LAN_C_19N	GND
21	GND	LAN_B_19P	LAN_B_19N	GND	LAN_D_19P	LAN_D_19N	GND
20	GND	LAN_A_18P	LAN_A_18N	GND	LAN_C_18P	LAN_C_18N	GND
19	GND	LAN_B_18P	LAN_B_18N	GND	LAN_D_18P	LAN_D_18N	GND
18	GND	LAN_A_17P	LAN_A_17N	GND	LAN_C_17P	LAN_C_17N	GND
17	GND	LAN_B_17P	LAN_B_17N	GND	LAN_D_17P	LAN_D_17N	GND
16	GND	LAN_A_16P	LAN_A_16N	GND	LAN_C_16P	LAN_C_16N	GND
15	GND	LAN_B_16P	LAN_B_16N	GND	LAN_D_16P	LAN_D_16N	GND
14	GND	LAN_A_15P	LAN_A_15N	GND	LAN_C_15P	LAN_C_15N	GND
13	GND	LAN_B_15P	LAN_B_15N	GND	LAN_D_15P	LAN_D_15N	GND
12	GND	LAN_A_14P	LAN_A_14N	GND	LAN_C_14P	LAN_C_14N	GND
11	GND	LAN_B_14P	LAN_B_14N	GND	LAN_D_14P	LAN_D_14N	GND
10	GND	LAN_A_13P	LAN_A_13N	GND	LAN_C_13P	LAN_C_13N	GND
9	GND	LAN_B_13P	LAN_B_13N	GND	LAN_D_13P	LAN_D_13N	GND
8	GND	LAN_A_12P	LAN_A_12N	GND	LAN_C_12P	LAN_C_12N	GND
7	GND	LAN_B_12P	LAN_B_12N	GND	LAN_D_12P	LAN_D_12N	GND
6	GND	LAN_A_11P	LAN_A_11N	GND	LAN_C_11P	LAN_C_11N	GND
5	GND	LAN_B_11P	LAN_B_11N	GND	LAN_D_11P	LAN_D_11N	GND
4	GND	LAN_A_10P	LAN_A_10N	GND	LAN_C_10P	LAN_C_10N	GND
3	GND	LAN_B_10P	LAN_B_10N	GND	LAN_D_10P	LAN_D_10N	GND
2	GND	LAN_A_9P	LAN_A_9N	GND	LAN_C_9P	LAN_C_9N	GND
1	GND	LAN_B_9P	LAN_B_9N	GND	LAN_D_9P	LAN_D_9N	GND

2.5 Switch Settings

Standalone/CMM Mode Switch (SW6)

The cPCI-6S10 comes with SW6 for user to set the blade to stand-alone mode or CMM mode. The cPCI-6S10 can boot without CMM in standalone mode.

Pin	Function
All off	Stand alone mode (default)
2 ON	CMM Disable
3 ON	SYSTEM# ENABLE
4 ON	EJECT Close



Boot Select Switch (SW13)

SW13 switch is to select boot from either SPI flash or NAND flash.

Pin	Function
1 ON	Boot from SPI flash (Default)
2 ON	Boot from nand flash

NAND Flash Boot Mode Switch (SW14)

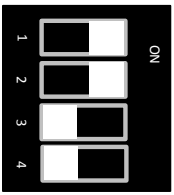
SW14 switch sets either Auto or Manual mode for boot from NAND flash

Pin 1	Pin 2	Function
Off	(no effect)	Auto mode (default)
On	On	Manual mode: Select NAND 0
On	Off	Manual mode: Select NAND 1

Console Port Switch (SW21)

SW21 switch sets the console port to either BCM56150 or to IPMC.

Pin	Function
1/2 On, 3/4 Off	Console port to BCM56150 (Default)
1/2 Off, 3/4 On	Console port to IPMC



3 Hardware Platform Management

3.1 Platform Management Overview

The purpose of the hardware platform management system is to monitor, control, and assure proper operation of CompactPCI blades. The hardware platform management system watches over the basic health of the system, reports anomalies, and provides feedback to the chassis management module (CMM) when needed. The hardware platform management system can retrieve inventory information and sensor readings as well as receive event reports and failure notifications from blades and other Intelligent FRUs. The hardware platform management system can also perform basic recovery operations such as power cycle or reset of managed entities.

The IPMC controller on the cPCI-6S10 supports an intelligent hardware management system, based on the Intelligent Platform Management Interface Specification. The hardware management system provides the ability to manage the power, cooling, and interconnect needs of intelligent devices; monitor events; and log events to a central repository.

3.2 IPMI Commands

3.2.1 Standard Commands

In terminal mode, the command is enclosed by square brackets. Each Hex number is separated by a single blank. This command group ranges from 0x00, 0x02, 0x04, 0x06, 0x08, 0x0A, to 0x0C in Netfn list.

Format

Request: [18 00 02]

- Byte 1: NetFn/rsLUN(00b)
- Byte 2: rqSeq/Bridge
- Byte 3: Command
- Byte 4-n: request data

Response: [1C 00 02 00]

- Byte 1: NetFn
- Byte 2: rqSeq
- Byte 3: Command
- Byte 4: Completion code
- Byte 5-n: response data

NetFn Codes:

Request	Response	Name
00	01	Chassis
02	03	Bridge
04	05	Sensor / Event
06	07	App
08	09	Firmware
0A	0B	Storage
0C	0D	Transport
0E	2B	Reserved
2C	2D	Group Extension
2E	2F	OEM
30	3F	Controller specific OEM/Group

Table 3-1: NetFn codes

Response Codes

Description	Code
IPMI_SUCCESS	0x00
IPMI_NODE_BUSY	0xC0
IPMI_INVALID_COMMAND	0xC1
IPMI_COMMAND_INVALID_FOR_LUN	0xC2
IPMI_TIMEOUT	0xC3
IPMI_OUT_OF_SPACE	0xC4
IPMI_RESERVATION_INVALID_OR_CANCELED	0xC5
IPMI_REQUEST_DATA_TRUNCATED	0xC6
IPMI_REQUEST_DATA_LENGTH_INVALID	0xC7
IPMI_REQUEST_DATA_LENGTH_LIMIT_EXCEEDED	0xC8
IPMI_PARAMETER_OUT_OF_RANGE	0xC9
IPMI_CANNOT_RETURN_NUMBER_OF_REQUESTED_BYTES	0xCA
IPMI_REQUESTED_DATA_NOT_PRESENT	0xCB
IPMI_INVALID_DATA_IN_REQUEST	0xCC
IPMI_COMMAND_ILLEGAL_FOR_SPECIFIED_OBJECT	0xCD
IPMI_CANNOT_PROVIDE_COMMAND_RESPONSE	0xCE
IPMI_CANNOT_EXECUTE_DUPLICATED_REQUEST	0xCF
IPMI_SDR_REPOSITORY_IN_UPDATE_MODE	0xD0
IPMI_DEVICE_IN_FIRMWARE_UPDATE_MODE	0xD1
IPMI_INITIALIZATION_IN_PROGRESS	0xD2
IPMI_DESTINATION_UNAVAILABLE	0xD3
IPMI_INSUFFICIENT_PRIVILEGE_LEVEL	0xD4
IPMI_COMMAND_NOT_SUPPORTED_IN_PRESENT_STATE	0xD5
IPMI_READ_ONLY_PARAMETER	0x82
IPMI_UNSPECIFIED_ERROR	0xFF

Table 3-2: Response Codes

Message Length

The IPMI standard overall message for “no-bridging” messages is specified as 32 bytes, maximum, including slave address. For bridging messages to other interfaces, Master Write-Read and Send Message commands are allowed to exceed 32-bytes on IPMI.

The table below shows the required interface length in IPMI 1.5. Some interfaces have extra recommended values in the IPMI specification.

Interface	Length(bytes)
KCS/SMIC Input	40
KCS/SMIC Output	38
BT Input	42
BT Output	40
IPMB Input	32
IPMB Output	36
SMBus 2.0 Input	36
SMBus 2.0 Output	36
Private Bus Input	34
Private Bus Output	23
LAN/PPP Input	45
LAN/PPP Output	42

Table 3-3: Required Message Length for IPMI 1.5

3.2.2 IPMItool

IPMItool is an open-source software, which supports several message interfaces to communicate with IPMI devices. It includes pre-define commands such like “fru” and “mc”, and can also send raw IPMI commands.

To send IPMI commands to the cPCI-6S10’s IPMC, instead of using IPMItool under Linux login, it is possible to communicate remotely with cPCI-6S10 IPMC through RMCP protocol, with the help of the IPMItool utility.

Message Interfaces

- Access through Linux driver (on LMP locally): ipmitool <command>
- RMCP remote client: ipmitool -I lan -H <hostname> [-p <port>] [-U <username>] [-A <authtype>] <command>
- If bridged command is issued. An extra parameter should be applied.
  - -t: select target slave address (0xB0 for slot1, 0xB2 for slot2, etc, in ADLINK Chassis cPCIS-3300BLS).

Slot No.	IPMB Address (hex)	Type
01	B0	CPU
02	B2	CPU
03	B4	CPU
04	B6	CPU
05	B8	SWH
06	BA	CPU
07	BC	CPU
08	BE	CPU
09	C0	CPU
10	C4	SWH
11	C6	CPU
12	C8	CPU
13	CA	CPU
14	CC	CPU

Table 3-4: cPCIS-3300BLS Chassis Slots

Examples

Access from local LMP

ipmitool mc info

Get device Information from onboard payload with Linux IPMI driver.

ipmitool raw 0x06 0x01

Get device id with IPMI raw command.

Access from RMCP remote client, bridged through the BMC

ipmitool -I lan -H 172.20.5.225 -U admin -P admin -t 0xB8 raw 0x06 0x01

Remote send raw command “Get device Id” to IPMC (**172.20.5.225**) via RMCP protocol with username: **admin**, password: **admin**, and let CMM to bridge it to CPCI-6S10 board plugged in slot 5 (slave address: **0xB8**).

Response

- **Complete:** Completion code is 0x00 and will be skipped, and only response data be printed.
- **Error:** More error message will be printed.

3.2.3 IPMItool Pre-defined Commands

- **ipmitool raw:** Send a raw command request.
- **ipmitool mc:** Print the management controller status and global enabled options.

```
# ipmitool mc
MC Commands:
  reset <warm|cold>
  info
  wdt
  selftest
  getenables
  setenables <option=on|off> ...
    recv_msg_intr    Receive Message Queue Int.
   event_msg_intr    Event Message Buffer Full Int.
    event_msg        Event Message Buffer
    system_event_log System Event Logging
    oem0             OEM 0
    oem1             OEM 1
    oem2             OEM 2
```

- **ipmitool fru:** Print built-in FRU and scan SDR for FRU locators.

```
# ipmitool fru
FRU Device Description: Builtin FRU Device (ID 0)
Board Mfg Date:        Tue Sep 16 20:00:00 2014
Board Mfg:             ADLINK Technology
```

```
Board Product:        cPCI-6S10
Board Serial:         ADLINK-XXXX-XXXX
Board Part Number:    cPCI-6S10
Product Manufacturer: ADLINK Technology
Product Name:         cPCI-6S10
Product Part Number:  cPCI-6S10
Product Version:      A2
Product Serial:       ADLINK-XXXX-XXXX
Product Asset Tag:    N/A
```

- **ipmitool -I lan -H 172.20.5.225 -U admin -P admin -t 0xB4 sdr:** Print Sensor Data Repository entries and readings.

```
BMC_WatchDog    | no reading    | ns
POWER_GOOD      | 0 unspecified | cr
P1V             | 1.01 Volts    | ok
P1V2            | 1.20 Volts    | ok
P1V5            | 1.50 Volts    | ok
P0V75           | 0.74 Volts    | ok
+3.3V           | 3.32 Volts    | ok
+5.0V           | 5.05 Volts    | ok
54685_TEMP      | 47 degrees C  | ok
56150_TEMP      | 67 degrees C  | nc
```

3.2.4 Supported IPMItool Commands

Get Device ID

ipmitool	ipmitool [parameters] mc info
Terminal mode	[18 00 01]: raw 0x06 01
Description	Get device's id from selected MC.

Example

```
root@BDSP-A-0-0-1:~# ipmi mc info
Device ID                 : 18
Device Revision           : 0
Firmware Revision         : 1.2
IPMI Version              : 1.5
Manufacturer ID           : 24339
Manufacturer Name         : Unknown (0x5F13)
Product ID                : 21267 (0x5313)
```

```
Product Name              : Unknown (0x5313)
Device Available          : yes
Provides Device SDRs      : yes
Additional Device Support :
    Sensor Device
    FRU Inventory Device
    IPMB Event Generator
Aux Firmware Rev Info     :
    0xa1
    0x00
    0x00
    0x00
```

Response Data Fields

1	Completion code
2	Device ID. 00 = unspecified.
3	Device Revision
4	Firmware Revision 1
5	Firmware Revision 2
6	IPMI version
7	Addition Device support
8:10	Manufacture ID
11:12	Product ID

Cold Reset

ipmitool	ipmitool [parameters] mc reset cold
Terminal mode	[18 00 02]: raw 0x06 0x02
Description	Reset IPMC

In IPMC console, cold reset message will be printed.

```
<_>: BMR-AVR Firmware (v1.0.2), cPCI edition.
<_>: Pigeon Point Systems (c) Copyright 2004.
<_>: boot_type: 0xA3
<_>: Reset type: COLD, reset cause: Software
<_>: app_status: 0x01
<_>: Operating mode: Normal
```

Response Data Fields

1	Completion code

Reset Watchdog Timer

ipmitool	ipmitool [parameters] mc wdt rst
Terminal mode	[18 00 22]: raw 0x06 0x22
Description	This command is used to reset IPMC watchdog timer. The ipmitool supports this command.

Response Data Fields

1	Completion code

Set Watchdog Timer

ipmitool	ipmitool [parameters] raw 0x06 0x24 0x03 0x01 0x00 0x20 0x00 0x01
Terminal mode	[18 00 24 03 01 00 20 00 01]
Description	This command is used to set IPMC watchdog timer.

Response Data Fields

1	Time use [7] –1b = don't log [6] –1b = do not stop timer on set Watchdog Timer command. [5:3] – reserved [2:0] – timer use 000b = reserved 001b = BIOS FRB2 010b = BIOS/POST 011b = OS Load 100b = SMS/OS 101b = OEM 110b – 111b = reserved
2	Timer actions [7] – reserved [6:4] – pre-timeout interrupt 000b = none 001b = SMI 010b = NMI / Diagnostic Interrupt 011b = Messaging interrupt 110b,111b = reserved [3] – reserved [2:0] = timeout action 000b = no action 001b = Hard reset 010b = Power down 011b = Power cycle 110b,111b = reserved
3	Pre-timeout interval in seconds.
4	Timer use expiration flags clear (0b = leave alone, 1b = clear timer use expiration bit) [7] – reserved [6] – reserved [5] – OEM [4] – SMS/OS [3] – OS/Load [2] – BIOS/POST [1] – BIOS FRB2 [0] – reserved
5	Initial countdown value, lsbyte(100ms/count)
6	Initial countdown value, msbyte

Get Watchdog Timer

ipmitool	ipmitool [parameters] mc watchdog get
Terminal mode	[18 00 25] : raw 0x06 0x25
Description	This command is used to get watchdog timer info. We can use ipmitool command to get this information.

Response:

```
root@iProc /root:~# ipmitool mc watchdog get
Timer Use:              0x42 - BIOS/POST
Timer Actions:          0x01 - Hard Reset
Pre-timeout interval: 0x00
Timer Use Expiration: 0x00
Initial Countdown:    360 ms
Present Countdown:    357 ms
```

Response Data Fields

1	Completion code
2	Timer use [7] –1b = don't log [6] –1b = timer started. 0b = timer stopped. [5:3] – reserved [2:0] – timer use 000b = reserved 001b = BIOS FRB2 010b = BIOS/POST 011b = OS Load 100b = SMS/OS 101b = OEM 110b – 111b = reserved

3	Timer actions [7] – reserved [6:4] – pre-timeout interrupt 000b = none 001b = SMI 010b = NMI / Diagnostic Interrupt 011b = Messaging interrupt 110b,111b = reserved [3] – reserved [2:0] = timeout action 000b = no action 001b = Hard reset 010b = Power down 011b = Power cycle 110b,111b = reserved
4	Pre-timeout interval in seconds.
5	Timer use expiration flags (1b = timer expired while associated 'use' was selected) [7] – reserved [6] – reserved [5] – OEM [4] – SMS/OS [3] – OS/Load [2] – BIOS/POST [1] – BIOS FRB2 [0] – reserved
6	Initial countdown value, lsbyte(100ms/count)
7	Initial countdown value, msbyte
8	Preset countdown value, lsbyte
9	Preset countdown value, msbyte

Master Write-Read

ipmitool	Ipmitool [parameters] raw 0x06 0x52 …
Terminal mode	[18 00 52 ….]
Description	This command is used for low-level I2C write, read, or write-read access to IPMB or private busses behind a management controller.

Request Fields

1	Bus ID: [7:4] channel number [3:1] bus ID, 0-based (always 000b for public bus) [0] bus type: 0 = public (e.g. IPMB or PCI Management Bus) 1 = private bus
2	[7:1] - Slave Address [0] - reserved. Write as 0.
3	Read count. Number of bytes to read, 1 based. 0 = no bytes to read. The maximum read count should be at least 34 bytes. This allows the command to be used for an SMBus Block Read. This is required if the command provides access to an SMBus or IPMB. Otherwise, if FRU SEEPROM devices are accessible, at least 31 bytes must be supported. Note that an implementation can support fewer bytes can be supported if none of the devices to be accessed can handle the recommended minimum.
4:N	Data to write. This command should support at least 35 bytes of write data. This allows the command to be used for an SMBus Block Write with PEC. Otherwise, if FRU SEEPROM devices are accessible, at least 31 bytes must be supported. Note that an implementation is allowed to support fewer bytes if none of the devices to can handle the recommended minimum.

Response Data Fields

1	Completion Code A management controller shall return an error Completion Code if an attempt is made to access an unsupported bus. generic, plus following command specific codes: 81h = Lost Arbitration 82h = Bus Error 83h = NAK on Write 84h = Truncated Read
(2:M)	Bytes read from specified slave address. This field will be absent if the read count is 0. The controller terminates the I 2 C transaction with a STOP condition after reading the requested number of bytes.

Set Event Receiver

ipmitool	ipmitool [parameters] raw 0x04 0x00 0x20 0x00
Terminal mode	[10 00 00 20 00]
Description	Set event receiver address, default event receiver is 0x20.

Request Data Fields

1	Slave address
2	[7:2] = reserved [1:0] = Event receiver LUN

Note: The commands “Set event receiver” and “Get event receiver” are used for event delivery between IPMC and BMC. Other address (except 0x20 to CMM by default) will cause incorrect behavior.

Get Event Receiver

ipmitool	ipmitool [parameters] raw 0x04 0x01
Terminal mode	[10 00 01]
Description	Get event receiver's slave address

Response Data Fields

1	Completion code
2	Slave address
3	[7:2] = reserved [1:0] = Event receiver LUN

Note: The commands "Set event receiver" and "Get event receiver" are used for event delivery between IPMC and CMM. Other address (except 0x20 to CMM by default) will cause incorrect behavior.

Platform Event

ipmitool	ipmitool [parameters] raw 0x04 0x02 0x04 0x01 0x01 0x01 0x07 0xFF 0xFF 0x48
Terminal mode	[10 00 02 04 01 01 01 07 FF FF 48]
Description	Send event to event receiver. We don't need to fire this command directly. This command can be testing when we test sensor event.

Note: The command "Platform event" is only available from Linux driver side. IPMC would only accept this command from Addin card, and drops messages from CMM with bridged format. Because IPMC only send this command to CMM, IPMC don't receive this command coming from outside.

We can test IPMC by sending a event to BMC:

```
root@BCNMB-A:~# ipmitool raw 0x2E 0x88 0x39 0x28
     0x00 0x02
 39 28 00
root@BCNMB-A:~# ipmitool -I lan -H 172.20.225 -U
     admin -P admin  sel list last 1
  2e | 04/28/2010 | 00:49:19 | Add-in Card #0x9d
     | soft reset | Asserted
```

Get Device SDR Info

ipmitool	ipmitool [parameters] raw 0x4 0x20
Terminal mode	[10 00 20]
Description	Get device sdr information

```
root@BCNMB-A:~# ipmitool raw 0x4 0x20
 0a 01 00 00 00 00
```

Response Fields

1	Completion code
2	Number of sensors in device for LUN this command was addressed to.
3	Flags: Dynamic population [7] – 0b = static sensor population 1b = dynamic sensor population [6:4] = reserved Device LUns: [3] – 1b = LUN 3 has sensors [2] – 1b = LUN 2 has sensors [1] – 1b = LUN 1 has sensors [0] – 1b = LUN 0 has sensors
4:7	Sensor Population Change Indicator LS byte first. Four byte timestamp or counter. Updated or incremented each time the sensor population changes. This field is not provided if the flags indicate a static sensor population.

Get Device SDR

ipmitool	ipmitool [parameters] sdr list all
Terminal mode	[10 00 21 00 00 00 00 00 FF]
Description	This command is used to get SDR info. SDR format depends on IPMI spec. Ipmitool support sdr command to read those data. We can use “ipmitool sdr” to get those data without pain. If MC support “SDR repository device”, ipmitool will use “SDR repository” first. When MC only support “Sensor Device”, this command has been applied to retrieve SDR. We fix ipmitool to support list SDR from Device SDR with “list” command.

Example:

List all SDR from IPMC (from Device SDR)

```
root@BCNMB-A:~# ipmitool -I lan -H 172.20.5.225
     -U admin -P admin -t 0xB8 sdr list all
cPCI-6S10        | Dynamic MC @ B4h  | ok
BMC_WatchDog     | 0 unspecified     | nc
POWER_GOOD       | 0 unspecified     | cr
P1V              | 1.01 Volts        | ok
P1V2             | 1.20 Volts        | ok
P1V5             | 1.50 Volts        | ok
P0V75            | 0.74 Volts        | ok
+3.3V            | 3.32 Volts        | ok
+5.0V            | 5.05 Volts        | ok
54685_TEMP       | 48 degrees C      | ok
56150_TEMP       | 66 degrees C      | nc
```

Request Fields

1	Reservation ID. LS Byte. Only required for partial reads with a non-zero ‘Offset into record’ field. Use 0000h for reservation ID otherwise.
2	Reservation ID. MS Byte.
3	Record ID of record to Get LS Byte. 0000h returns the first record
4	Record ID of record to Get, MS Byte.
5	Offset into record

6	Bytes to read. FFh means read entire record

Response Fields

1	Completion code
2	Record ID for next record, LS Byte
3	Record Id for next record. MS Byte
4:3+N	Requested bytes from record

Reserve Device SDR Repository

ipmitool	ipmitool [parameters] raw 0x04 0x22
Terminal mode	[10 00 22]
Description	This command is used to obtain a reservation ID. Reserve ID is used by “Get Device SDR” command.

Response Fields

1	Completion code
2	Reservation ID, LS Byte 0000h reserved
3	Reservation ID, MS Byte

Set Sensor Threshold

ipmitool	ipmitool [parameters] sensor thresh “sensor id” <threshold> <setting>
Terminal mode	[10 00 26 00 00 00 00 00 00 00 00]
Description	This command is used to set sensor threshold. We can use ipmitool to fire this command.

Command Format:

```
sensor thresh <id> <threshold> <setting>
   id        : name of the sensor for which
     threshold is to be set
   threshold : which threshold to set
                  unr = upper non-recoverable
                  ucr = upper critical
                  unc = upper non-critical
                  lnc = lower non-critical
                  lcr = lower critical
                  lnr = lower non-recoverable
   setting   : the value to set the threshold to
```

Example:

```
Set sensor “56150_TEMP”’s UNR threshold to 88
     ipmitool –I lan –H 172.20.5.225 –U admin –P
     admin –t 0xB8 sensor thresh “56150_TEMP” unr
     88
```

Request Fields

1	Sensor number(FFh = reserved)
2	[7:6] – reserved [5] – 1b = set upper non-recoverable threshold [4] – 1b = set upper critical threshold [3] – 1b = set upper non- critical threshold [2] – 1b = set lower non-recoverable threshold [1] – 1b = set lower critical threshold [0] – 1b = set lower non- critical threshold
3	Lower non-critical threshold
4	Lower critical threshold
5	Lower non-recoverable threshold
6	Upper non-critical threshold
7	Upper critical threshold
8	Upper non-recoverable threshold

Response Fields

1	Completion code

Get Sensor Threshold

ipmitool	ipmitool [parameters] sensor get <sensor id>
Terminal mode	[10 00 27 00]
Description	This command is used to get threshold. We can use ipmitool command to get this info.

Example:

```
root@BCNMB-A:~# ipmitool -I lan -H 172.20.5.225
     -U admin -P admin sensor get "56150_TEMP "
Locating sensor record...
Sensor ID              : 56150_TEMP (0x8)
 Entity ID             : 160.96
 Sensor Type (Analog)  : Temperature
 Sensor Reading        : 67 (+/- 0) degrees C
 Status                : Upper Non-Critical
 Lower Non-Recoverable : na
 Lower Critical        : na
 Lower Non-Critical    : na
 Upper Non-Critical    : 60.000
 Upper Critical        : 75.000
 Upper Non-Recoverable : 88.000
 Assertion Events      : unc+
 Deassertion Events    : lnc- lnc+ lcr- lcr+
     lnr- lnr+ unc- ucr- ucr+ unr- unr+
 Assertions Enabled    : unc+ ucr+ unr+
 Deassertions Enabled  : unc+ ucr+ unr+
```

Request Fields

1	Sensor number(FFh = reserved)

Response Fields

1	Completion code
2	[7:6] – reserved [5] – 1b = upper non-recoverable threshold [4] – 1b = upper critical threshold [3] – 1b = upper non- critical threshold [2] – 1b = lower non-recoverable threshold [1] – 1b = lower critical threshold [0] – 1b = lower non- critical threshold
3	Lower non-critical threshold
4	Lower critical threshold
5	Lower non-recoverable threshold
6	Upper non-critical threshold
7	Upper critical threshold
8	Upper non-recoverable threshold

Set Sensor Event Enable

ipmitool	ipmitool [parameters] raw 0x04 0x28 0x01 0xd0 0x30 0x0c 0x00 0x00
Terminal mode	[10 00 28 01 d0 03 0C 00 00]
Description	This command is use to enable sensor events.

Example: Enable sensor 1's assertion for lower non-recoverable going high, and assertion event for upper non-recoverable going high.

```
[10 00 28 01 d0 20 08 00 00]
```

Request Fields

1	Sensor number(FFh = reserved)
2	[7] – 0b = disable all event message from this sensor [6] – 0b = disable scanning on this sensor [5:4] – 00b = do not change individual enables 01h = enable selected event messages 10b = disable selected event messages 11b = reserved
3	For sensors with threshold based events: [7] – 1b = select assertion event for upper non-critical going high [6] – 1b = select assertion event for upper non-critical going low [5] – 1b = select assertion event for lower non-recoverable going high [4] – 1b = select assertion event for lower non-recoverable going low [3] – 1b = select assertion event for lower critical going high [2] – 1b = select assertion event for lower critical going low [1] – 1b = select assertion event for lower non-critical going high [0] – 1b = select assertion event for lower non-critical going low For Sensors with discrete event: [7] – 1b = select assertion event for state bit 7 [6] – 1b = select assertion event for state bit 6 [5 – 1b = select assertion event for state bit 5 [4] – 1b = select assertion event for state bit 4 [3] – 1b = select assertion event for state bit 3 [2] – 1b = select assertion event for state bit 2 [1] – 1b = select assertion event for state bit 1 [0] – 1b = select assertion event for state bit 0

4	For sensors with threshold base events: [7:4] – reserved. [3] – 1b = select assertion event for upper non-recoverable going high [2] – 1b = select assertion event for upper non-recoverable going low [1] – 1b = select assertion event for upper critical going high [0] – 1b = select assertion event for upper critical going low For sensors with discrete events: [7] – reserved [6] – 1b = select assertion event for state bit 14 [5] – 1b = select assertion event for state bit 13 [4] – 1b = select assertion event for state bit 12 [3] – 1b = select assertion event for state bit 11 [2] – 1b = select assertion event for state bit 10 [1] – 1b = select assertion event for state bit 9 [0] – 1b = select assertion event for state bit 8
5	For sensors with threshold based events: [7] – 1b = select deassertion event for upper non-critical going high [6] – 1b = select deassertion event for upper non-critical going low [5] – 1b = select deassertion event for lower non-recoverable going high [4] – 1b = select deassertion event for lower non-recoverable going low [3] – 1b = select deassertion event for lower critical going high [2] – 1b = select deassertion event for lower critical going low [1] – 1b = select deassertion event for lower non-critical going high [0] – 1b = select deassertion event for lower non-critical going low For Sensors with discrete event: [7] – 1b = select deassertion event for state bit 7 [6] – 1b = select deassertion event for state bit 6 [5] – 1b = select deassertion event for state bit 5 [4] – 1b = select deassertion event for state bit 4 [3] – 1b = select deassertion event for state bit 3 [2] – 1b = select deassertion event for state bit 2 [1] – 1b = select deassertion event for state bit 1 [0] – 1b = select deassertion event for state bit 0

6	For sensors with threshold base events: [7:4] – reserved. [3] – 1b = select deassertion event for upper non-recoverable going high [2] – 1b = select deassertion event for upper non-recoverable going low [1] – 1b = select deassertion event for upper critical going high [0] – 1b = select deassertion event for upper critical going low For sensors with discrete events: [7] – reserved [6] – 1b = select deassertion event for state bit 14 [5] – 1b = select deassertion event for state bit 13 [4] – 1b = select deassertion event for state bit 12 [3] – 1b = select deassertion event for state bit 11 [2] – 1b = select deassertion event for state bit 10 [1] – 1b = select deassertion event for state bit 9 [0] – 1b = select deassertion event for state bit 8

Response Fields

1	Completion code

Get Sensor Event Enable

ipmitool	ipmitool [parameters] sensor get <id>
Terminal mode	[10 00 28 01 00 00 00]
Description	This command is used to get which sensor event is enable or no. We can use ipmitool to get this information.

Example:

```
root@BCNMB-A:~# ipmitool -I lan -H 172.20.5.225
     -U admin -P admin sensor get "56150_TEMP "
Locating sensor record...
Sensor ID              : 56150_TEMP (0x8)
 Entity ID             : 160.96
 Sensor Type (Analog)  : Temperature
 Sensor Reading        : 67 (+/- 0) degrees C
 Status                : Upper Non-Critical
 Lower Non-Recoverable : na
 Lower Critical        : na
 Lower Non-Critical    : na
 Upper Non-Critical    : 60.000
 Upper Critical        : 75.000
 Upper Non-Recoverable : 88.000
 Assertion Events      : unc+
 Deassertion Events    : lnc- lnc+ lcr- lcr+
     lnr- lnr+ unc- ucr- ucr+ unr- unr+
 Assertions Enabled    : unc+ ucr+ unr+
 Deassertions Enabled  : unc+ ucr+ unr+
```

Request Fields

1	Sensor number(FFh = reserved)

Response Fields

1	Completion code
2	[7] – 0b = All event messages disabled form this sensor [6] – 0b = Sensor scanning disabled [5:0] - reserved

3	For sensors with threshold based events: [7] – 1b = select assertion event for upper non-critical going high [6] – 1b = select assertion event for upper non-critical going low [5] – 1b = select assertion event for lower non-recoverable going high [4] – 1b = select assertion event for lower non-recoverable going low [3] – 1b = select assertion event for lower critical going high [2] – 1b = select assertion event for lower critical going low [1] – 1b = select assertion event for lower non-critical going high [0] – 1b = select assertion event for lower non-critical going low For Sensors with discrete event: [7] – 1b = select assertion event for state bit 7 [6] – 1b = select assertion event for state bit 6 [5 – 1b = select assertion event for state bit 5 [4] – 1b = select assertion event for state bit 4 [3] – 1b = select assertion event for state bit 3 [2] – 1b = select assertion event for state bit 2 [1] – 1b = select assertion event for state bit 1 [0] – 1b = select assertion event for state bit 0
4	For sensors with threshold base events: [7:4] – reserved. [3] – 1b = select assertion event for upper non-recoverable going high [2] – 1b = select assertion event for upper non-recoverable going low [1] – 1b = select assertion event for upper critical going high [0] – 1b = select assertion event for upper critical going low For sensors with discrete events: [7] – reserved [6] – 1b = select assertion event for state bit 14 [5] – 1b = select assertion event for state bit 13 [4] – 1b = select assertion event for state bit 12 [3] – 1b = select assertion event for state bit 11 [2] – 1b = select assertion event for state bit 10 [1] – 1b = select assertion event for state bit 9 [0] – 1b = select assertion event for state bit 8

5	For sensors with threshold based events: [7] – 1b = select deassertion event for upper non-critical going high [6] – 1b = select deassertion event for upper non-critical going low [5] – 1b = select deassertion event for lower non-recoverable going high [4] – 1b = select deassertion event for lower non-recoverable going low [3] – 1b = select deassertion event for lower critical going high [2] – 1b = select deassertion event for lower critical going low [1] – 1b = select deassertion event for lower non-critical going high [0] – 1b = select deassertion event for lower non-critical going low For Sensors with discrete event: [7] – 1b = select deassertion event for state bit 7 [6] – 1b = select deassertion event for state bit 6 [5] – 1b = select deassertion event for state bit 5 [4] – 1b = select deassertion event for state bit 4 [3] – 1b = select deassertion event for state bit 3 [2] – 1b = select deassertion event for state bit 2 [1] – 1b = select deassertion event for state bit 1 [0] – 1b = select deassertion event for state bit 0
6	For sensors with threshold base events: [7:4] – reserved. [3] – 1b = select deassertion event for upper non-recoverable going high [2] – 1b = select deassertion event for upper non-recoverable going low [1] – 1b = select deassertion event for upper critical going high [0] – 1b = select deassertion event for upper critical going low For sensors with discrete events: [7] – reserved [6] – 1b = select deassertion event for state bit 14 [5] – 1b = select deassertion event for state bit 13 [4] – 1b = select deassertion event for state bit 12 [3] – 1b = select deassertion event for state bit 11 [2] – 1b = select deassertion event for state bit 10 [1] – 1b = select deassertion event for state bit 9 [0] – 1b = select deassertion event for state bit 8

Re-arm Sensor Event

ipmitool	ipmitool [parameters] raw 0x04 0x2a 0x01 0x00
Terminal mode	[10 00 2a 01 00]
Description	This command is used to re-arm a sensor.

Request Fields

1	sensor number (FFh = reserved)
2	[7] - 0b = re-arm all event status from this sensor. If 0, following parameter bytes are ignored, but should still be written as 0, if sent. [6:0] - reserved. Write as 000_0000b.
(3)*	For sensors with threshold based events: [7] - 1b = re-arm assertion event for upper non-critical going high [6] - 1b = re-arm assertion event for upper non-critical going low [5] - 1b = re-arm assertion event for lower non-recoverable going high [4] - 1b = re-arm assertion event for lower non-recoverable going low [3] - 1b = re-arm assertion event for lower critical going high [2] - 1b = re-arm assertion event for lower critical going low [1] - 1b = re-arm assertion event for lower non-critical going high [0] - 1b = re-arm assertion event for lower non-critical going low For sensors with discrete events: [7] - 1b = re-arm assertion event for state bit 7 [6] - 1b = re-arm assertion event for state bit 6 [5] - 1b = re-arm assertion event for state bit 5 [4] - 1b = re-arm assertion event for state bit 4 [3] - 1b = re-arm assertion event for state bit 3 [2] - 1b = re-arm assertion event for state bit 2 [1] - 1b = re-arm assertion event for state bit 1 [0] - 1b = re-arm assertion event for state bit 0

(4)*	For sensors with threshold based events: [7:4] - reserved. Write as 0000b. [3] - 1b = re-arm assertion event for upper non-recoverable going high [2] - 1b = re-arm assertion event for upper non-recoverable going low [1] - 1b = re-arm assertion event for upper critical going high [0] - 1b = re-arm assertion event for upper critical going low For sensors with discrete events: (00h otherwise) [7] - reserved. Ignore on read. [6] - 1b = re-arm assertion event for state bit 14 [5] - 1b = re-arm assertion event for state bit 13 [4] - 1b = re-arm assertion event for state bit 12 [3] - 1b = re-arm assertion event for state bit 11 [2] - 1b = re-arm assertion event for state bit 10 [1] - 1b = re-arm assertion event for state bit 9 [0] - 1b = re-arm assertion event for state bit 8
(5)*	For sensors with threshold based events: [7] - 1b = re-arm deassertion event for upper non-critical going high [6] - 1b = re-arm deassertion event for upper non-critical going low [5] - 1b = re-arm deassertion event for lower non-recoverable going high [4] - 1b = re-arm deassertion event for lower non-recoverable going low [3] - 1b = re-arm deassertion event for lower critical going high [2] - 1b = re-arm deassertion event for lower critical going low [1] - 1b = re-arm deassertion event for lower non-critical going high [0] - 1b = re-arm deassertion event for lower non-critical going low For sensors with discrete events: (00h otherwise) [7] - 1b = re-arm deassertion event for state bit 7 [6] - 1b = re-arm deassertion event for state bit 6 [5] - 1b = re-arm deassertion event for state bit 5 [4] - 1b = re-arm deassertion event for state bit 4 [3] - 1b = re-arm deassertion event for state bit 3 [2] - 1b = re-arm deassertion event for state bit 2 [1] - 1b = re-arm deassertion event for state bit 1 [0] - 1b = re-arm deassertion event for state bit 0

(6)*	For sensors with threshold based events: [7:4] - reserved. Write as 0000b. [3] - 1b = re-arm deassertion event for upper non-recoverable going high [2] - 1b = re-arm deassertion event for upper non-recoverable going low [1] - 1b = re-arm deassertion event for upper critical going high [0] - 1b = re-arm deassertion event for upper critical going low For sensors with discrete events: (00h otherwise) [7] - reserved. Ignore on read. [6] - 1b = re-arm deassertion event for state bit 14 [5] - 1b = re-arm deassertion event for state bit 13 [4] - 1b = re-arm deassertion event for state bit 12 [3] - 1b = re-arm deassertion event for state bit 11 [2] - 1b = re-arm deassertion event for state bit 10 [1] - 1b = re-arm deassertion event for state bit 9 [0] - 1b = re-arm deassertion event for state bit 8

Response Fields

1	Completion code

Get Sensor Reading

ipmitool	ipmitool [parameters] sensor get <id>
Terminal mode	[10 00 2d 01]
Description	This command is used to get sensor reading. We can use ipmitool to read each sensor's reading value.

Example: Get Tmp421 sensor reading

```
root@BCNMB-A:~# ipmitool -I lan -H 172.20.5.225
     -U admin -P admin sensor get "56150_TEMP "
Locating sensor record...
Sensor ID              : 56150_TEMP (0x8)
 Entity ID             : 160.96
 Sensor Type (Analog)  : Temperature
 Sensor Reading        : 67 (+/- 0) degrees C
 Status                : Upper Non-Critical
 Lower Non-Recoverable : na
 Lower Critical        : na
 Lower Non-Critical    : na
 Upper Non-Critical    : 60.000
 Upper Critical        : 75.000
 Upper Non-Recoverable : 88.000
 Assertion Events      : unc+
 Deassertion Events    : lnc- lnc+ lcr- lcr+
     lnr- lnr+ unc- ucr- ucr+ unr- unr+
 Assertions Enabled    : unc+ ucr+ unr+
 Deassertions Enabled  : unc+ ucr+ unr+
```

Request Fields

1	Sensor number(FFh = reserved)

Response Fields

1	Completion code
2	Sensor reading
3	[7] – 0b = all event messages disabled form this sensor [6] – 0b = sensor scaling disabled [5] – 1b = reading/state unavailable [4:0] – reserved

4	For threshold-base sensors Present threshold comparison status [7:6] – reserved [5] – 1b = at or above (>=) upper non-recoverable threshold [4] – 1b = at or above (>=) upper critical threshold [3] – 1b = at or above (>=) upper non- critical threshold [2] – 1b = at or above (<=) lower non- recoverable threshold [1] – 1b = at or above (<=)lower critical threshold [0] – 1b = at or above (<=)lower non- critical threshold For discrete reading sensors [7] – 1b = state 7 asserted [6] – 1b = state 6 asserted [5] – 1b = state 5 asserted [4] – 1b = state 4 asserted [3] – 1b = state 3 asserted [2] – 1b = state 2 asserted [1] – 1b = state 1 asserted [0] – 1b = state 0 asserted
5	For discrete reading sensors only(optional) [7] – reserved [6] – 1b = state 14 asserted [5] – 1b = state 13 asserted [4] – 1b = state 12 asserted [3] – 1b = state 11 asserted [2] – 1b = state 10 asserted [1] – 1b = state 9 asserted [0] – 1b = state 8 asserted

Get Sensor Type

ipmitool	ipmitool [parameters] sensor get <id>
Terminal mode	[10 00 2F 00]
Description	This command is used to get sensor type. We can use ipmitool to get this information

Example:

```
root@BCNMB-A:~# ipmitool -I lan -H 172.20.5.226
     -U admin -P admin sensor get "P1V "
Locating sensor record...
Sensor ID              : P1V (0x2)
 Entity ID             : 3.96
 Sensor Type (Analog)  : Voltage
 Sensor Reading        : 1.005 (+/- 0) Volts
 Status                : ok
 Lower Non-Recoverable : 0.945
 Lower Critical        : 0.965
 Lower Non-Critical    : 0.985
 Upper Non-Critical    : 1.105
 Upper Critical        : 1.125
 Upper Non-Recoverable : 1.155
 Assertion Events      : lnc- lcr-
 Deassertion Events    : lnc+ lcr+ lnr- lnr+
     unc- unc+ ucr- ucr+ unr- unr+
 Assertions Enabled    : lnc- lcr- lnr- unc+
     ucr+ unr+
 Deassertions Enabled  : lnc- lcr- lnr- unc+
     ucr+ unr+
```

Request Fields

1	Sensor number(FFh = reserved)

Response Fields

1	Completion code
2	Sensor type
3	[7] – reserved [6:0] – Event/Reading type code

Get FRU Inventory Area Info

ipmitool	Ipmitool [parameters] raw 0x0A 0x10 0x00
Terminal mode	[28 00 10 00]
Description	Get FRU inventory area info

Request Fields

1	FRU Device ID. (FFh = reserved) 0 – CMM 32h – PSU1 33h – PSU2

Response Fields

1	Completion code
2	FRU inventory area size in bytes, LS byte
3	FRU inventory area size in bytes, MS byte
4	[7:1] – reserved [0] – 0b = device is accessed by bytes 1b = device is accessed by words

Note: The command “Get FRU inventory area info” and “Read FRU data” support customized design on this platform for PSU FRU eeprom re-direction. However, “write FRU data” is not acceptable for PSU.

Read FRU Data

ipmitool	Ipmitool [parameters] raw 0x0A 0x11 0x00 0x00 0x00 0x08
Terminal mode	[28 00 11 00 00 00 08]
Description	Read FRU data

IPMItool Support:

IPMItool supports “read entire fru record” command. We can use below command to get entire fru command:

```
ipmitool fru read 0 fru-data.bin
```

Then fru with id 0 will save to file fru-data.bin

Example:

IPMC

```
root@BCNMB-A:~# ipmitool  fru read 0
     fru-data.bin
Fru Size         : 512 bytes
Done
```

We can print all fru:

```
root@BCNMB-A:~# ipmitool fru
Board Mfg Date        : Tue Sep 16 20:00:00 2014
 Board Mfg            : ADLINK Technology
 Board Product        : cPCI-6S10
 Board Serial         : ADLINK-XXXX-XXXX
 Board Part Number    : cPCI-6S10
 Product Manufacturer: ADLINK Technology
 Product Name         : cPCI-6S10
 Product Part Number : cPCI-6S10
 Product Version      : A2
 Product Serial       : ADLINK-XXXX-XXXX
 Product Asset Tag    : N/A
```

Request Fields

1	FRU Device ID. (FFh = reserved) 0 – CMM 32h – PSU1 33h – PSU2
2	FRU inventory offset to read, LS Byte
3	FRU inventory offset to read, MS Byte
4	Count to read – count is ‘1’ based

Response Fields

1	Completion code
2	Count returned – count is ‘1’ based
3:2+N	Requested data

Note: The command “Get FRU inventory area info” and “Read FRU data” support customized design on this platform for PSU FRU eeprom re-direction. However, “write FRU data” is not acceptable for PSUs.

```
[root@iProc /root]# ipmitool -I lan -H 127.0.0.1
     -U admin -P admin raw 0xa 0x11 0x0 0x0 0x0
     8
 08 01 00 01 0c 13 19 00 c6
```

Write FRU Data

ipmitool	Ipmitool [parameters] raw 0x0A 0x12 0x00 0x00 0x00 0x01
Terminal mode	[28 00 12 00 00 00 01]
Description	This command is used to write raw FRU data to eeprom. We can use ipmitool to write data more easy.

IPMItool Support:

We can use ipmitool to write entire FRU record.

```
ipmitool fru write 0 fru-data.bin
```

Additionally, we can write raw data to eeprom:

```
ipmitool fru 0 field <section> <index> <string>
     <section>: is a string which refers to FRU
          Inventory Information
          Storage Areas and may be referring to:
          c FRU Inventory Chassis Info Area
          b FRU Inventory Board Info Area
          p FRU Inventory Product Info Area
     <index>: specifies the field number. Field
          numbering starts on the first ’English text’
```

```
        field type. For instance in the <board> info
        area field '0' is <Board Manufacturer> and
        field '2' is <Board Serial Number>; see IPMI
        Platform Management FRU Information Storage
        Definition v1.0 R1.1 for field locations.
   <string> must be the same length as the string
        being replaced and must be 8-bit ASCII
        (0xCx).
```

Request Fields

1	FRU Device ID. (FFh = reserved)
2	FRU inventory offset to read, LS Byte
3	FRU inventory offset to read, MS Byte
4:3+N	Data to write

Response Fields

1	Completion code
2	Count written – count is 1 based

Get SDR Repository Info

ipmitool	ipmitool [parameters] sdr info
Terminal mode	[28 00 20]
Description	Get SDR repository information

```
root@BCNMB-A:~# ipmitool -I lan -H 127.0.0.1 -U
     admin -P admin sdr info
SDR Version                       : 0x51
Record Count                      : 117
Free Space                        : 33016 bytes
Most recent Addition              :
Most recent Erase                 :
SDR overflow                      : no
SDR Repository Update Support     : non-modal
Delete SDR supported              : yes
Partial Add SDR supported         : yes
Reserve SDR repository supported  : yes
SDR Repository Alloc info supported : yes
```

Response Fields

1	Completion code
2	SDR Version - version number of the SDR command set for the SDR Device. 51h for this specification. (BCD encoded with bits 7:4 holding the Least Significant digit of the revision and bits 3:0 holding the Most Significant bits.)
3	Record count LS Byte - number of records in the SDR Repository
4	Record count MS Byte - number of records in the SDR Repository
5:6	Free Space in bytes, LS Byte first. 0000h indicates ‘full’, FFFEh indicates 64KB-2 or more available. FFFFh indicates ‘unspecified’.
7:10	Most recent addition timestamp. LS byte first.
11:14	Most recent erase (delete or clear) timestamp. LS byte first.

15	Operation Support [7] - Overflow Flag. 1=SDR could not be written due to lack of space in the SDR Repository. [6:5] - 00b = modal/non-modal SDR Repository Update operation unspecified 01b = non-modal SDR Repository Update operation supported 10b = modal SDR Repository Update operation supported 11b = both modal and non-modal SDR Repository Update supported [4] - reserved. Write as 0b [3] - 1b=Delete SDR command supported [2] - 1b=Partial Add SDR command supported [1] - 1b=Reserve SDR Repository command supported [0] - 1b=Get SDR Repository Allocation Information command supported

Get SDR

ipmitool	Ipmitool [parameters] raw 0xa 0x23
Terminal mode	[28 00 23]
Description	Get a SDR record from repository If get a partial SDR record, user should call “Reserve SDR repository” to get a valid reservation id. A partial get command means offset is not 0 or length is not 0xff. A record id with 0x0000 is first record. If a MC support both “Sensor Device” and “SDR repository device”, “SDR repository device” will has been used first. Ipmitool will check “Get_device_id” command to decide use which SDR source.

Example:

Get SDR use raw command.

```
root@BCNMB-A:~# ipmitool -I lan -H 127.0.0.1 -U
     admin -P admin raw 0xa 0x22
 06 00
root@BCNMB-A:~# ipmitool -I lan -H 127.0.0.1 -U
     admin -P admin raw 0xa 0x23 0x6 0x0 0 0 0
     10
 01 00 00 00 51 12 13 20 00 cc 29 00
```

List all SDR from IPMC (from Device SDR)

```
root@BCNMB-A:~# ipmitool -I lan -H 172.20.5.225
     -U admin -P admin -t 0xB4 sdr list all
BMC_WatchDog     | 0 unspecified     | nc
POWER_GOOD       | 0 unspecified     | cr
P1V              | 1.01 Volts        | ok
P1V2             | 1.20 Volts        | ok
P1V5             | 1.50 Volts        | ok
P0V75            | 0.74 Volts        | ok
+3.3V            | 3.32 Volts        | ok
+5.0V            | 5.05 Volts        | ok
54685_TEMP       | 34 degrees C      | ok
56150_TEMP       | 47 degrees C      | ok
```

Request Fields

1	Reservation ID. LS Byte. Only required for partial reads with a non-zero ‘Offset into record’ field. Use 0000h for reservation ID otherwise.
2	Reservation ID. MS Byte.
3	Record ID of record to Get, LS Byte
4	Record ID of record to Get, MS Byte
5	Offset into record
6	Bytes to read. FFh means read entire record.

Response Fields

1	Completion code
2	Record ID for next record, LS Byte
3	Record ID for next record, MS Byte
4:N	Record Data

3.3 Controller Specific OEM/Group Commands

Show Card Version

ipmitool	ipmitool [parameters] raw 0x30 0x12
Terminal mode	[C0 00 12]
Description	Show 6S10 Card Version.

Request Data Fields

1	None

Response Data Fields

1	Completion Code
2	Predefined – A5h
3	'V'– 56h (Version)
4	APP_FIRMWARE_REV (MSB)
5	APP_FIRMWARE_REV (LSB)
6	'P' –50h (Product)
7	APP_PRODUCT_ID(MSB)
8	APP_PRODUCT_ID(LSB)
9	'S' –53h
10	Reserved – 00h
11	Predefined – 5Ah

Example:

Show Card Version

```
[root@iProc /root]#ipmitool raw 0x30 0x12
 a5 56 01 02 50 53 13 53 00 5a
```

Re-Scan GA Input

ipmitool	ipmitool [parameters] raw 0x30 0x22
Terminal mode	[C0 00 22]
Description	Re-Scan the IPMB address from the HW GA Input

Example:

Use this command to read the Card's IPMB address.

```
[root@iProc /root]#ipmitool raw 0x30 0x22
 b8
```

Request Data Fields

1	None

Response Data Fields

1	Completion Code
2	IPMB Address – xxh

Report Geography Address

ipmitool	ipmitool [parameters] raw 0x30 0xF0
Terminal mode	[C0 00 F0]
description	Get 6S10 Geography Address.

Example:

Use this command to read the Card's IPMB and Geography Address.

```
[root@iProc /root]#ipmitool raw 0x30 0xF0
 b8 03 01
```

Request Data Fields

1	None

Response Data Fields

1	Completion Code
2	IPMB Address – xxh
3	Geography Address – xxh
4	Reserved – 01h

Payload Power Reset

ipmitool	ipmitool [parameters] raw 0x30 0xF5
Terminal mode	[C0 00 F5]
Description	Reset the Payload when it is in PowerOn state.

Request Data Fields

1	None

Response Data Fields

1	Completion Code
2	None

Power Off the Payload

ipmitool	Ipmitool [parameters] raw 0x30 0xF6
Terminal mode	[C0 00 F6]
Description	Power Off the Payload.

Request Data Fields

1	None

Response Data Fields

1	Completion Code

Power On the Payload

ipmitool	Ipmitool [parameters] raw 0x30 0xF7
Terminal mode	[C0 00 F7]
Description	Power On the Payload.

Request Data Fields

1	None

Response Data Fields

1	Completion Code

Set Boot Flash

ipmitool	Ipmitool [parameters] raw 0x30 0x15 0x0
Terminal mode	[C0 00 15 00]
Description	Set the Boot Flash.

Request Data Fields

1	Boot flash number – 0 (Flash #0)/1 (Flash #1)

Response Data Fields

1	Completion Code

Note: This command will not change the physical boot flash CS# signal until a power-off cycle done.

Get Boot Flash

ipmitool	Ipmitool [parameters] raw 0x30 0x16
Terminal mode	[C0 00 16]
Description	Get the Boot Flash No.

Request Data Fields

1	None

Response Data Fields

1	Completion Code
2	Boot flash number – 0/1

This page intentionally left blank.

4 Getting Started

This chapter describes the installation of the cPCI-6S10:

4.1 Heatsink

The cPCI-6S10 comes with on board BGA BCM 56150 processor and heatsink pre-installed. Removal of heatsink/CPU by users is not recommended. Please contact your ADLINK service representative for assistance.

Handle with caution as the heat sink can get very hot. Do not touch the heat sink when installing or removing the board. The board should not be placed on any surface or in any form of storage container until the board and heat sink have cooled down to room temperature.

4.2 Installing the cPCI-6S10

Insert the cPCI-6S10 into a PICMG 2.16 backplane with a fabric slot. The symbol for the switch slot is 

4.3 Configuring the cPCI-6S10

There is a serial port and management port at the front panel of the cPCI-6S10. Both ports can be used to connect to the cPCI-6S10 console. Follow the steps below.

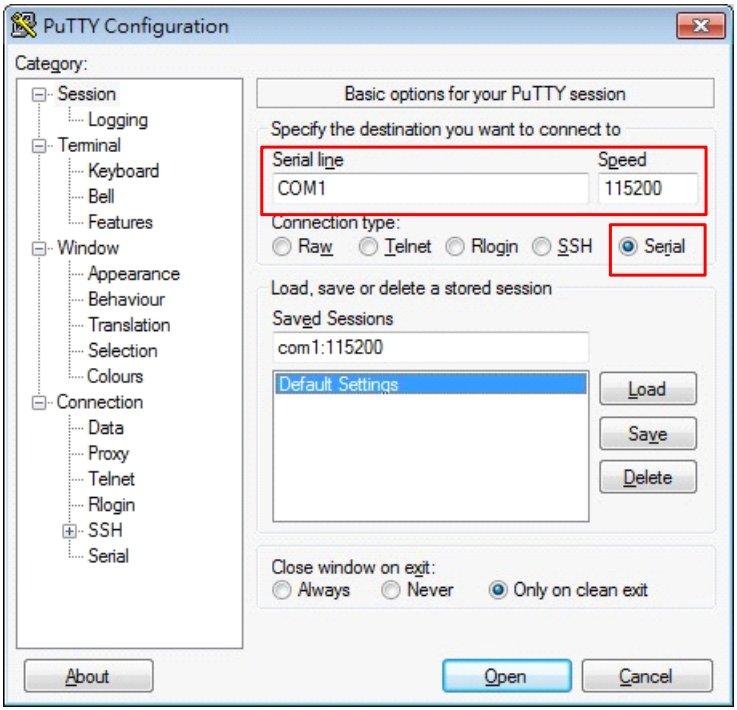
User's can connect to the cPCI-6S10 console using "PuTTY". Download PuTTY from http://www.putty.org. Run the program to open the PuTTY configuration window.

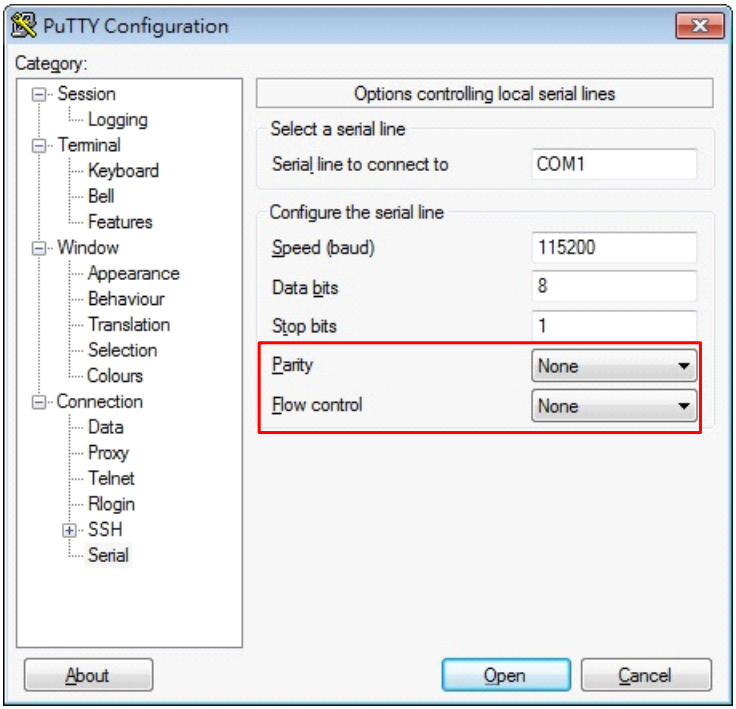
Serial Port Connection

Connect the RJ-45 to DB-9 adapter cable to the COM port on the front panel. Set "Serial line" to the COM port of the client PC.

To connect to the **cPCI-6S10 local management processor** (LMP) console:

- Set SW21 on the cPCI-6S10 to "Console port to BCM56150": 1, 2 On; 3, 4 Off; see “Console Port Switch (SW21)” on page 18
- Enter the following settings into the PuTTY Configuration window:
  - Speed: 115200
  - Parity bit: None
  - Flow Control: None
  - All others settings default
- Click "Open" to open the PuTTY command line interface



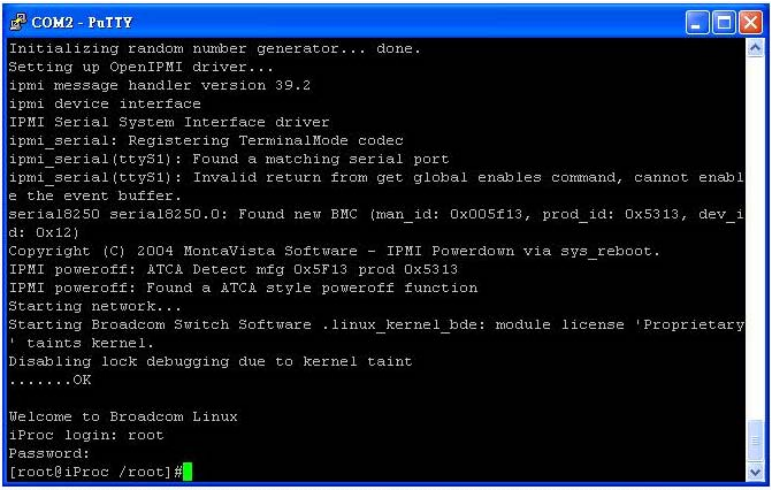


To connect to the **cPCI-6S10 IPMC debug serial port**:

- Set the SW21 the cPCI-6S10 to "Console port to IPMC": 1,2 Off; 3, 4 On; see “Console Port Switch (SW21)” on page 18
- Enter the following settings into the PuTTY Configuration window:
  - Speed: 9600
  - Parity bit: None
  - Flow Control: None
  - All others settings default
- Click "Open" to open the PuTTY command line interface

After powering on the cPCI-6S10, the terminal program PuTTY will connect to the switch blade via the COM port. Login with the following:

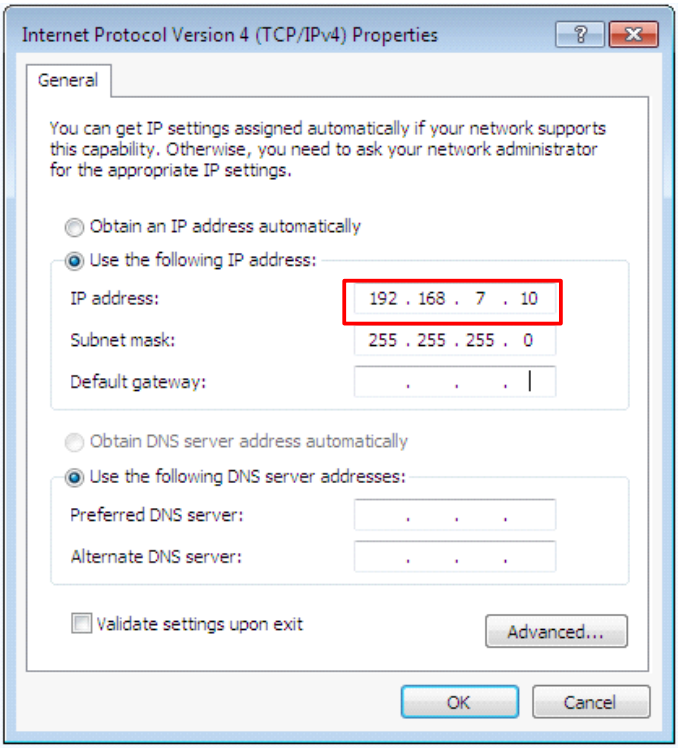
User: root
No Password



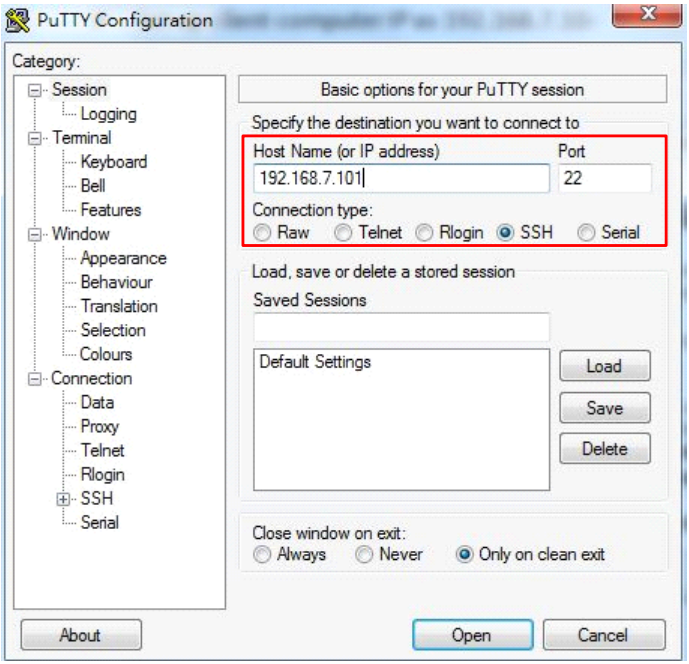
LAN Port Connection

Connect the client computer to the Management port on the front panel with an RJ-45 cable.

Set the client computer IP to 192.168.7.10.



In the PuTTY configuration window, set “Connection type” to “SSH”. Set the host IP to 192.168.7.101, Port 22.



Login as follows:

User: root
No Password

Login to the Broadcom Shell

Users can login to the Broadcom shell by connecting to LAN port or serial port. The application "/usr/local/bcm/bcm.user" is the complete BCM command shell. This includes the BCM shell application as well as the BCM API and all drivers. All BCM SDK related files are stored in the directory "/usr/local/bcm/", and the startup script is "/etc/init.d/S70bcm".

After Linux has started, it runs bcm.user in background mode and listens to port 9895. Users can enter the BCM shell by using the following command:

```
telnet localhost 9895
```

In the CLI, typing "quit" will close the telnet connection, but will not terminate the bcm.user process.

When bcm.user is started, it clears all Ethernet switch settings. Meanwhile, the shell executes the startup script "/usr/local/bcm/rc.soc". The system administrator can customize the startup script and add initialization command sequences as required.

On the cPCI-6S10, another script file "/usr/local/bcm/customer_config.soc" is used to include special configurations for the board. It is executed as the last part of "/usr/local/bcm/rc.soc".

customer extra configurations after SDK init.

/usr/local/bcm/customer_config.soc

```
# ADLINK. Switch LED to Serial2Parallel mode.
setreg top_parallel_led_ctrl 0x3ff00
# ADLINK. Restart LEDuP to load program.
led stop
led load /usr/local/bcm/cpci-6s10_led_0.hex
led auto on
led start
# ADLINK. 54685 LED select & act set.
phy ge16 0x1c 0xb430
phy ge17 0x1c 0xb430
phy ge18 0x1c 0xb430
phy ge19 0x1c 0xb430
```

```
phy ge20 0x1c 0xb430
phy ge21 0x1c 0xb430
phy ge22 0x1c 0xb430
phy ge23 0x1c 0xb430
phy ge16 0x10 0
phy ge17 0x10 0
phy ge18 0x10 0
phy ge19 0x10 0
phy ge20 0x10 0
phy ge21 0x10 0
phy ge22 0x10 0
phy ge23 0x10 0

# Set xe0 and xe2 as SFI
port xe0,xe2 if=sfi an=off speed=10000

# Disable flow control
port ge rpau=off tpau=off
port xe rpau=off tpau=off
```

The example script reads parameters from "/usr/local/bcm/config.bcm.cpci-6s10". This config file contains most of the PHY addresses.

/usr/local/bcm/config.bcm.cpci-6s10

```
obcm5615x_config=0
os=unix
pbmp_xport_xe.0=0x3c000000
phy_port_primary_and_offset_ge8=0x0a00
phy_port_primary_and_offset_ge9=0x0a01
phy_port_primary_and_offset_ge10=0x0a02
phy_port_primary_and_offset_ge11=0x0a03
phy_port_primary_and_offset_ge12=0x0a04
phy_port_primary_and_offset_ge13=0x0a05
phy_port_primary_and_offset_ge14=0x0a06
phy_port_primary_and_offset_ge15=0x0a07
phy_port_primary_and_offset_ge16=0x1200
phy_port_primary_and_offset_ge17=0x1201
phy_port_primary_and_offset_ge18=0x1202
phy_port_primary_and_offset_ge19=0x1203
phy_port_primary_and_offset_ge20=0x1204
phy_port_primary_and_offset_ge21=0x1205
phy_port_primary_and_offset_ge22=0x1206
```

```
phy_port_primary_and_offset_ge23=0x1207
phy_port_primary_and_offset_ge0=0x0207
phy_port_primary_and_offset_ge1=0x0206
phy_port_primary_and_offset_ge2=0x0205
phy_port_primary_and_offset_ge3=0x0204
phy_port_primary_and_offset_ge4=0x0203
phy_port_primary_and_offset_ge5=0x0202
phy_port_primary_and_offset_ge6=0x0201
phy_port_primary_and_offset_ge7=0x0200

# 54685 phy address
port_phy_addr_ge16=0x11
port_phy_addr_ge17=0x12
port_phy_addr_ge18=0x13
port_phy_addr_ge19=0x14
port_phy_addr_ge20=0x15
port_phy_addr_ge21=0x16
port_phy_addr_ge22=0x17
port_phy_addr_ge23=0x18

# 84752 phy address
phy_84752=1
phy_ext_rom_boot=0
port_phy_addr_xe0=0x3e
port_phy_addr_xe2=0x3f
```

The script "/etc/init.d/S70bcm" controls the execution of the BCM shell. "/etc/init.d/S70bcm start" launches the shell in background. This is done automatically every time the cPCI-6S10 boots up. "/etc/init.d/S70bcm stop" stops the BCM shell.

The commands in the BCM shell are classified into two categories: general commands for shell and commands for controlling the Ethernet switch. Refer to the cPCI-6S10 Software User's Manual for information on the commands used on the cPCI-6S10.

This page intentionally left blank.

5 Software Management

5.1 Introduction

cPCI-6S10 board has an Ethernet switch function block integrated in System-on-a-Chip(SOC) BCM56150 Ethernet controller processor as the heart of board. Broadcom provides software package "Broadcom® Network Switching Software SDK". This Broadcom SDK software enables software development for target systems using Broadcom switch devices from the StrataSwitch and StrataXGS families.

The source code provided by Broadcom can be compiled "out-of-the-box" into fully functional images for any of the reference platforms and development kits supported by Broadcom.

The BCM shell is an interactive application running on arm Linux. It allows users to access registers and memories on the switch devices and provides a means of higher-level configuration. The BCM shell has a number of useful capabilities:

- It is a finished, self-contained sample application that serves as a template for new applications.
- It is an efficient diagnostic tool. Customers may include access to BCM shell in their systems if they require extended diagnostics.
- It is a script running application; the SDK includes a number of sample scripts that the BCM shell can execute. The most important and widely used is the rc.soc script, which implements a full device initialization sequence.

5.2 Broadcom Network Switching Software SDK

Broadcom Network Switching Software SDK, hereafter referred to as BCM SDK, stores all related files in the directory "/usr/local/bcm/". The start-up script is "/etc/init.d/S70bcm". The complete BCM command shell is located in "/usr/local/bcm/bcm.user".

bcm.user Application

The bcm.user application is the complete BCM diagnostics shell. This includes the BCM shell application as well as the BCM API and complete driver.


NOTE:

It is not necessary to configure the cPCI-6S10 for general purpose packet switching. Incorrect use of following BCM.USER command may result in malfunction of the blade. Please contact ADLINK for assistance technical if you are not familiar with the bcm.user application.

Building and using this application must always be the first step to advanceyour system, as it allows you to easily validate the hardware platform and the correct operation of the driver software in your system. After the linux_kernel_bde.o and linux_user_bde.o modules have been inserted, and the device file created, you can run the bcm.user application that provides the BCM> prompt.


NOTE:

It is recommended that there should be only one running instance of bcm.user. That means, when bcm.user is running in background mode (system default), you should not launch another instance of it.

bcm.user is launched automatically on the cPCI-6S10 when Linux boots up. All I/O is redirected to a network socket by “netserver”, a useful tool provided by Broadcom. The user can keep the SDK application running in the background and access the diagnostic shell via telnet when necessary.

5.3 Commands

The commands in the BCM shell are classified into two categories. The first category listed below has general commands for the shell. The second category contains commands for controlling the Ethernet switch.

The command syntax is case-insensitive. In the command descriptions below, each command is composed of upper- and lower-case characters. The upper-case characters represent the command abbreviation. For example, the command "ps" is the abbreviation of the command "PortState".

Commands Common to All Modes

Command	Display list of commands
ASSert	Assert
BackGround	Execute a command in the background.
BCM	Set shell mode to BCM.
BCMX	Set shell mode to BCMX.
CASE	Execute command based on string match
CD	Change current working directory
cint	Enter the C interpreter
CONFig	Configure Management interface
CONSole	Control console options
CoPy	Copy a file
DATE	Set or display current date
DeBug	Enable/Disable debug output
DeBugMod	Enable/Disable debug output per module
DELAY	Put CLI task in a busy-wait loop for some amount of time
DEVice	Device add/remove
DISPatch	BCM Dispatch control.
Echo	Echo command line
EDline	Edit file using ancient line editor
EXIT	Exit the current shell (and possibly reset)
EXPR	Evaluate infix expression
FLASHINIT	Initialize on board flash as a file system
FLASHSYNC	Sync up on board flash with file system
FOR	Execute a series of commands in a loop
Help	Print this list OR usage for a specific command
HISTory	List command history
IF	Conditionally execute commands
JOBS	List current background jobs
KILL	Terminate a background job
LOCal	Create/Delete a variable in the local scope
LOG	Enable/Disable logging and set log file
LOOP	Execute a series of commands in a loop

LS	List current directory
MKDIR	Make a directory
MODE	Set shell mode
MORe	Copy a file to the console
MoVe	Rename a file on a file system
NOEcho	Ignore command line
Pause	Pause command processing and wait for input
PRINTENV	Display current variable list
PWD	Print platform dependent working directory
RCCache	Save contents of an rc file in memory
RCLoad	Load commands from a file
REBOOT	Reboot the processor
RM	Remove a file from a file system
RMDIR	Remove a directory
SAVE	Write data to a file
SET	Set various configuration options
SETENV	Create/Delete a variable in the global scope
SHell	Invoke a system dependent shell
SLeep	Suspend the CLI task for specified amount of time
TIME	Time the execution of one or more commands
Version	Print version and build information

5.4 Packet Manager

Users can use the ADLINK proprietary API tool, PacketManager, to configure the cPCI-6S10. For detailed information, please go to the ADLINK portal: **www.adlinktech.com/Products/AdvancedTCA/ARiPSoftware/PacketManager** to download the PacketManager user's manuals:

- ADLINK PacketManager Configuration Guide
- ADLINK PacketManager API Programming Guide

Important Safety Instructions

For user safety, please read and follow all **instructions**, **WARNINGS**, **CAUTIONS,** and **NOTES** marked in this manual and on the associated equipment before handling/operating the equipment.

- Read these safety instructions carefully.
- Keep this user's manual for future reference.
- Read the specifications section of this manual for detailed information on the operating environment of this equipment.
- When installing/mounting or uninstalling/removing equipment:
  - Turn off power and unplug any power cords/cables.
- To avoid electrical shock and/or damage to equipment:
  - Keep equipment away from water or liquid sources;
  - Keep equipment away from high heat or high humidity;
  - Keep equipment properly ventilated (do not block or cover ventilation openings);
  - Make sure to use recommended voltage and power source settings;
  - Always install and operate equipment near an easily accessible electrical socket-outlet;
  - Secure the power cord (do not place any object on/over the power cord);
  - Only install/attach and operate equipment on stable surfaces and/or recommended mountings; and,
  - If the equipment will not be used for long periods of time, turn off and unplug the equipment from its power source.

- Never attempt to fix the equipment. Equipment should only be serviced by qualified personnel.

A Lithium-type battery may be provided for uninterrupted, backup or emergency power.

Risk of explosion if battery is replaced with one of an incorrect type. Dispose of used batteries appropriately.

- Equipment must be serviced by authorized technicians when:
  - The power cord or plug is damaged;
  - Liquid has penetrated the equipment;
  - It has been exposed to high humidity/moisture;
  - It is not functioning or does not function according to the user's manual;
  - It has been dropped and/or damaged; and/or,
  - It has an obvious sign of breakage.

Getting Service

Ask an Expert: http://askanexpert.adlinktech.com

ADLINK Technology, Inc.
9F, No.166 Jian Yi Road, Zhonghe District
New Taipei City 235, Taiwan
Tel: +886-2-8226-5877
Fax: +886-2-8226-5717
Email: service@adlinktech.com

Ampro ADLINK Technology, Inc.
5215 Hellyer Avenue, #110
San Jose, CA 95138, USA
Tel: +1-408-360-0200
Toll Free: +1-800-966-5200 (USA only)
Fax: +1-408-360-0222
Email: info@adlinktech.com

ADLINK Technology (China) Co., Ltd.
300 Fang Chun Rd., Zhangjiang Hi-Tech Park
Pudong New Area, Shanghai, 201203 China
Tel: +86-21-5132-8988
Fax: +86-21-5132-3588
Email: market@adlinktech.com

ADLINK Technology GmbH
Hans-Thoma-Strasse 11
D-68163 Mannheim, Germany
Tel: +49-621-43214-0
Fax: +49-621 43214-30
Email: emea@adlinktech.com

Please visit the Contact page at **www.adlinktech.com** for information on how to contact the ADLINK regional office nearest you.