Custom Embedded Solutions

Network Appliance

PL-80720

User Manual

Dec 2014

Manual Edition 1.0, Aug., 2014

Version 1.0

The content of this document is intended to be accurate and reliable; the original manufacturer assumes no responsibility for any inaccuracies that may be contained in this manual. The original manufacturer reserves the right to make improvements to the products described in this manual at any time without prior notice.

Trademarks

RTL is a trademark of Realtek

Microsoft, Windows, Windows NT are either trademarks or registered trademarks of Microsoft Corporation

All other product names mentioned herein are used for identification purpose only and may be trademarks and/or registered trademarks of their respective companies

Limitation of Liability

While reasonable efforts have been made to ensure the accuracy of this document, the manufacturer and distributor assume no liability resulting from errors or omissions in this document, or from the use of the information contained herein.

For more information on PL-80720 or other WIN products, please visit our website http://www.win-ent.com.

For technical support send your inquiries to consultants@win-ent.com.

Contents

Chapter 1. General Information

1.1 Introduction

The PL-80720 is a desktop hardware platform designed for network service applications. It is suitable for SOHO (Small Office, Home Office), SMB (Small Medium Business), and ROBO (Remote Office, Branch Office) segments. The PL-80720 supports the Cavium® OCTEON III CN7010 family of processors and supports onboard DDR3L memory up to 4GB.

In order to provide the best network performance and utilization, the standard device comes equipped with onboard eMMC flash for boot device and has a place reserved for an optional slim-type one 2.5" SATA HDD. PL-80720 supports 5 Copper GbE ports on rear-panel. For easy access, the front panel also has two USB 2.0 port, one RJ-45 console port and LED indicators that monitor power and storage device activities for local system management, maintenance and diagnostics. In addition, the PL-80720 supports one mini-PCIe slot. It is RoHS, FCC and CE compliant.

1.2 Specifications

Processor System	CPU	Cavium OCTEON III CN7010 family processor, up to 1.0GHz.
	Chipset	Cavium OCTEON III CN7010 SOC
	BIOS	U-Boot
Memory	Technology	Onboard 36bit DDR3L 1066MHz Memory. ECC Support
	Capacity	Up to 4 GB onboard
Expansion	Expansion Slots	One mini-PCIe socket.
Ethernet	GbE Ethernet	Five RJ45 GbE ports, RTL8211E and RTL8214B via RGMII and QSGMII
Storage	SATA HDD	One internal 2.5" SATA HDD bay (Option)
	eMMC	On-board 4GB eMMC Flash, up to 8GB
I/O	USB	Two external USB2.0 one internal 5x2 pin header
	Serial	one RJ45 Console port (COM1) one internal 5x2 pin header (COM2)
	GPIO	4 x GPO pin header
Power Supply	Watt	40W AT power adaptor supply
Mechanical and Environmental	Form Factor	Desktop
	LED	1 x Power LED, (Green) 1 x System LED, (Green, Yellow) 1 x HDD LED, (Green) 1 x USB LED, (Green)
	Dimensions（W x D x H）	232mm (W) x 152mm (D) x 44mm (H)
	Operating Temperature	Operating: 0 ~ 40°C (32 ~ 104°F)
	Storage Temperature	-20 ~ 75°C (-4 ~ 167°F)
	Humidity	10 ~ 85% relative humidity, non-operating, non-condensing
	Certifications	CE/FCC

1.3 Ordering Information

We provide some accessories for PL-80720 appliance.

PL-8072A	Desktop Cavium OCTEON III CN7010 Network System, 5 Copper GbE, eMMC, SATA, Mini-PCIe
DK001	Cable development kit CB-CO5204-00 Cross over cable CB-EC5200-00 Ethernet cable CB-RJDB91-00 RJ45 Console cable

1.4 Packaging

Make sure that the following items have been included in the package before installation:

1. PL-80720 Appliance
2. Quick Installation Guide (Optional)
3. Cables (Optional)

If any item of above is missing or damaged, please contact your dealer or retailer from whom you purchased the PL-80720. Keep the box and carton for possible near-term shipment or storage of the PL-80720.
After you unpack the goods, inspect and make sure the packaging is intact. Do not connect the power adapter to the PL-80720 if it appears damaged.

Note: Keep the PL-80720 in the original packaging until you start installation.

1.5 Precautions

Please make sure you properly ground yourself before handling the PL-80720 appliance or other system components. Electrostatic discharge can be easily damage the PL-80720 appliance.

Do not remove the anti-static packing until you are ready to install the PL-80720 appliance.

Ground yourself before removing any system component from its protective anti-static packaging. To ground yourself grasp the expansion slot covers or other unpainted parts of the computer chassis.

Handle the PL-80720 appliance by its edges and avoid touching the components on it.

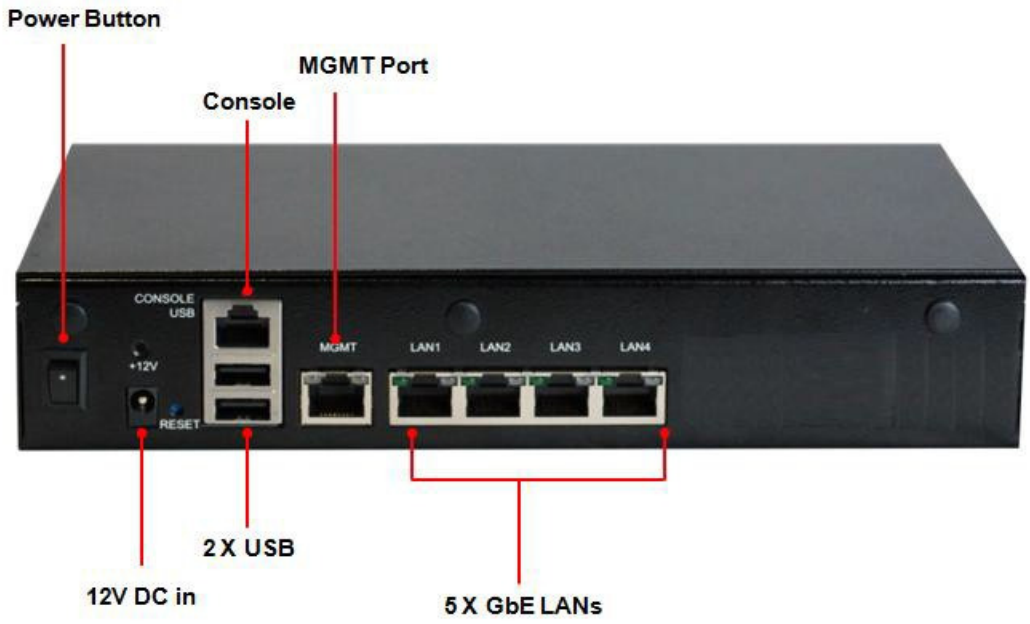
CAUTION

There is a risk of explosion if battery is replaced by incorrect type.
Dispose of used batteries according to the instructions.

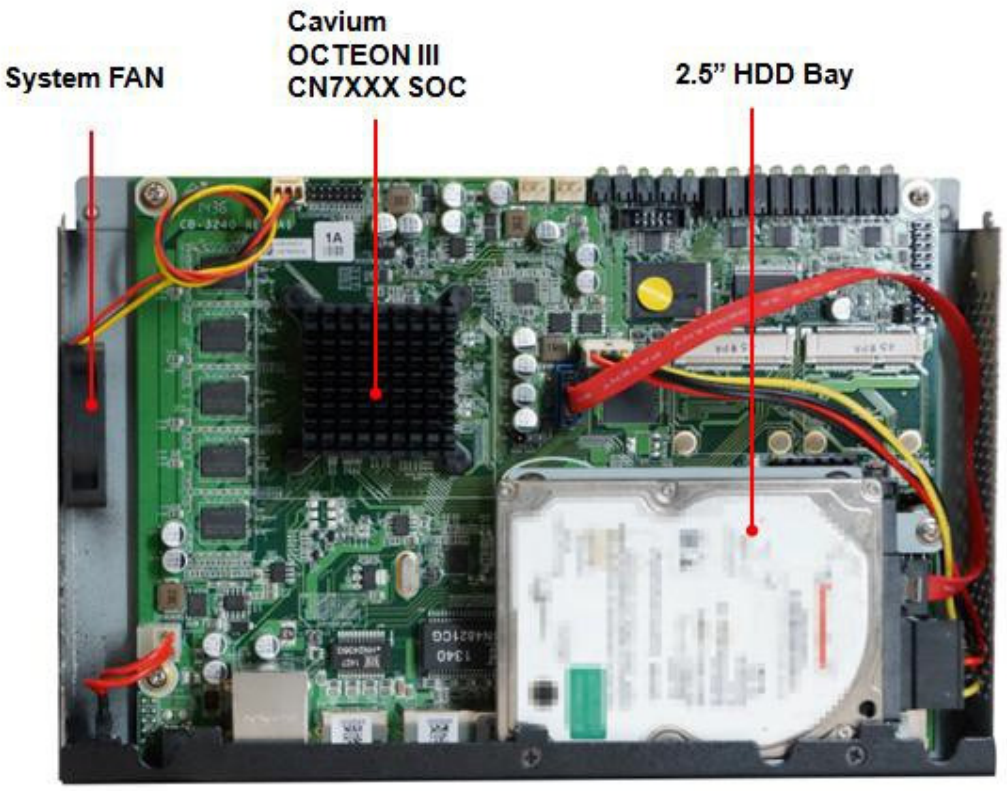
1.6 System Layout

PL-80720 Front Side Layout

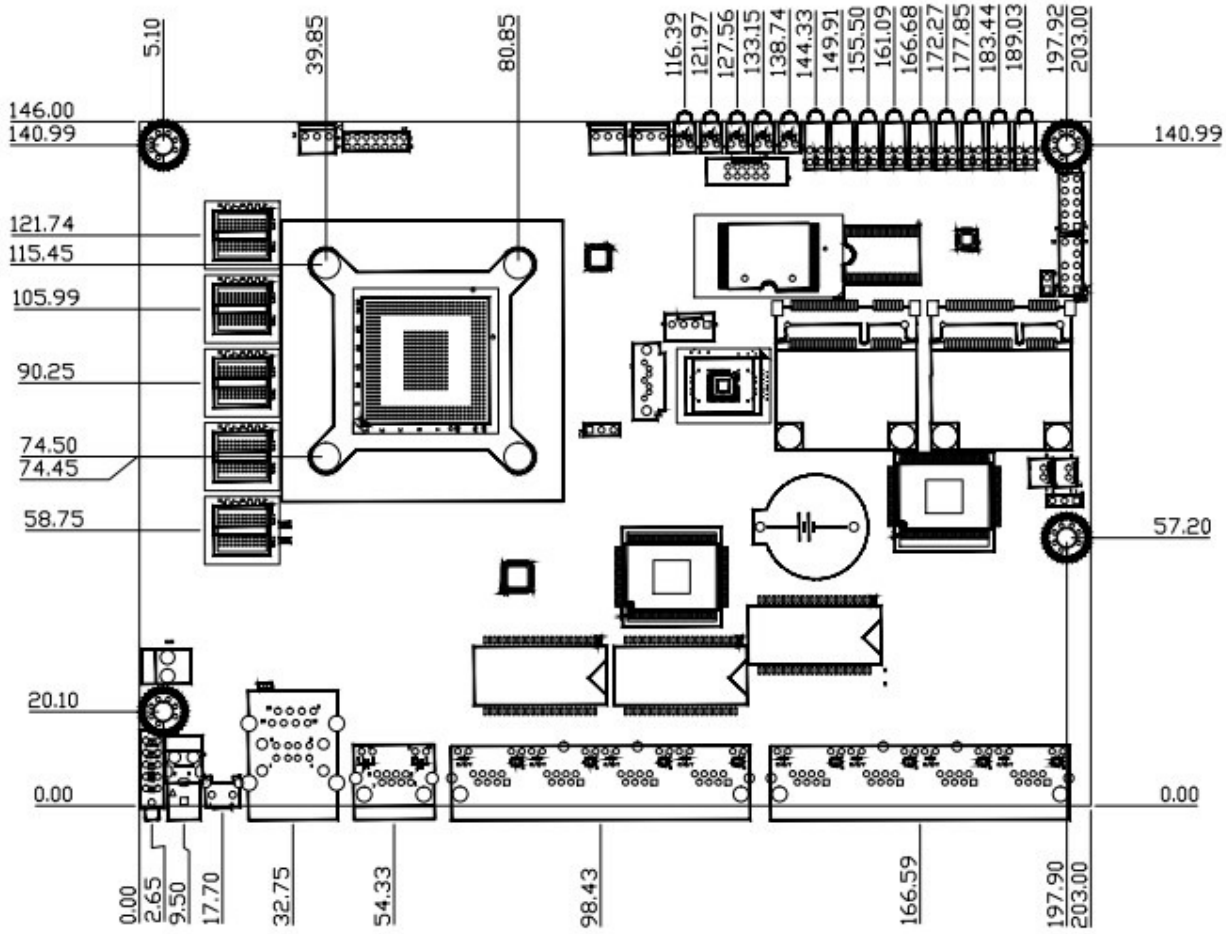
PL-80720 Rear Side Layout



PL-80720 Internal Layout

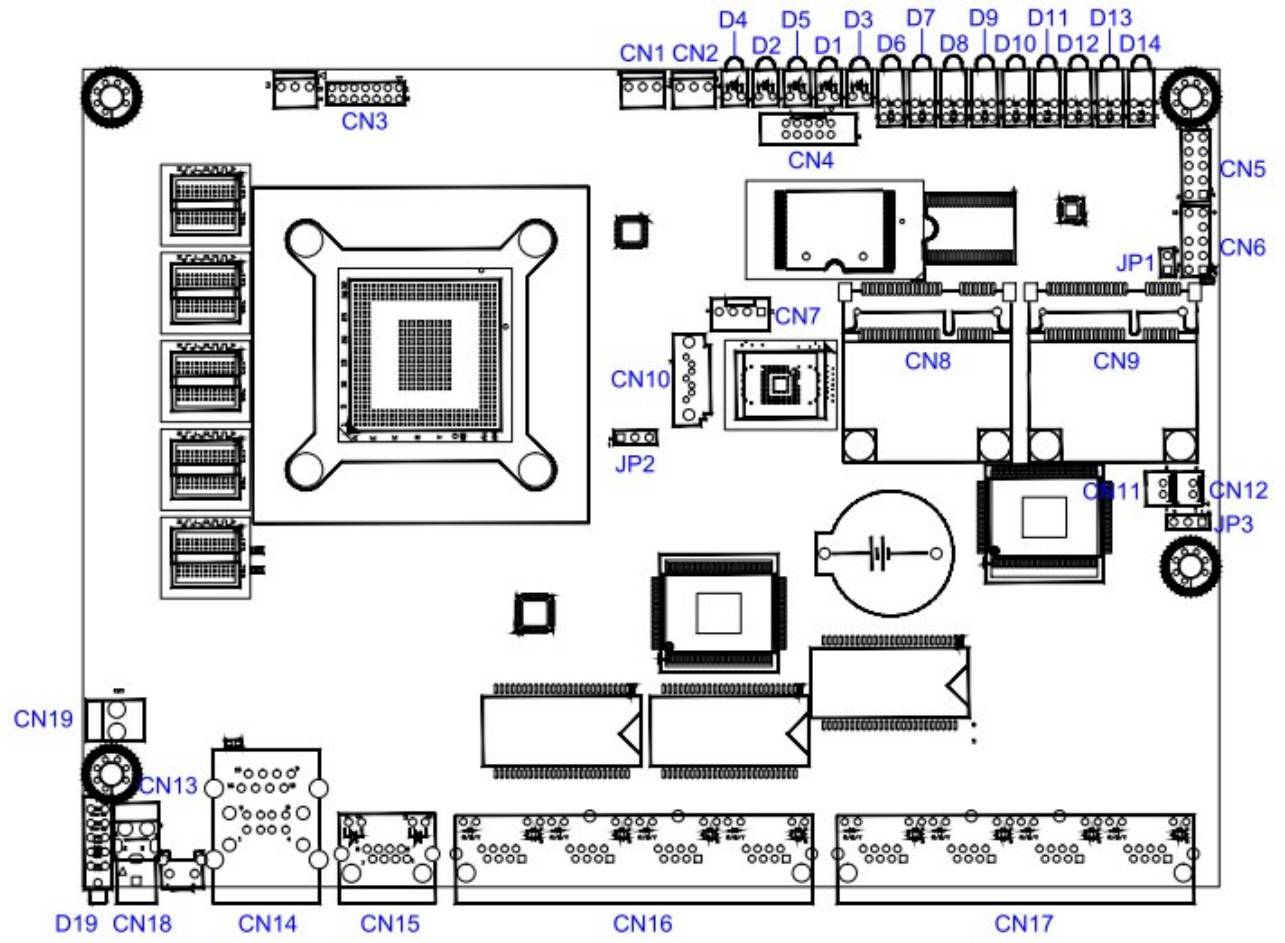


1.7 Board Dimensions



Chapter 2. Connector/Jumper Configuration

2.1 Connector/Jumper Locations and Definitions



MB-80720 Connectors & Jumpers

Connector List			
Connector	**Description**	**Connector**	**Description**
CN1	System FAN Connector	CN18	POWER JACK (+12V Only)
CN2	System FAN Connector	CN19	POWER SWITCH
CN3	JTAG Pin Header (Debug only)	JP1	WATCH DOG Select
CN4	COM2 BOX Header	JP2	PCIE GEN1/GEN2 Select
CN5	GPIO Pin Header	JP3	BYPASS Select
CN6	MCU Programming Pin Header (Debug only)	D1	USB LED
CN7	SATA Power Connector	D2	SYSTEM LED
CN8	MINIPCIE Connector	D3	BYPASS LED
CN9	MINIPCIE Connector	D4	POWER LED
CN10	SATA Connector	D5	HDD LED
CN11	WLAN LED 1x2 Wafer (from CN9)	D6~D14	LAN LED
CN12	WLAN LED 1x2 Wafer (from CN8)	LED8	Front panel LED : (From top to bottom) (POWER LED , HDD LED , SYSTEM LED , USB LED , BYPASS LED)
CN13	3.96mm 1x2 POWER (+12V Only)		
CN14	Console & USB Connector		
CN15	RJ45 LAN		
CN16	RJ45 LAN PORT (1X4)		
CN17	RJ45 LAN PORT (1X4)		

2.2 Connector and Jumper Settings

CN1: System FAN connector

Pin	Define
1	Ground
2	+12V
3	N/A

CN2: System FAN connector

Pin	Define
1	Ground
2	+12V
3	N/A

CN4: COM2 BOX Header

Pin	Define	Pin	Define
1	N/A	2	SIN
3	SOUT	4	N/A
5	GND	6	N/A
7	RTS	8	CTS
9	N/A	10	N/A

CN5: GPIO Pin Header

Pin	Define	Pin	Define
1	GPIO16	2	GP16+
3	GPIO17	4	GP17+
5	GPIO18	6	GP18+
7	GPIO19	8	GP19+
9	GND	10	+3.3V

CN7: SATA PWR

Pin	Define
1	+12V
2	GND
3	GND
4	+5V

CN8/9: MINI-PCIE

Pin	Define	Pin	Define
1	WAKE#	2	3.3V
3	Reserved	4	GND
5	Reserved	6	1.5V
7	CLKREQ#	8	Reserved
9	GND	10	Reserved
11	REFCLK-	12	Reserved
13	REFCLK+	14	Reserved
15	GND	16	Reserved
17	Reserved	18	GND
19	Reserved	20	Reserved
21	GND	22	PERST#
23	PERN0	24	+3.3VAUX
25	PERP0	26	GND
27	GND	28	+1.5V
29	GND	30	SMB_CLK
31	PETN0	32	SMB_DATA
33	PETP0	34	GND
35	GND	36	USB_D-
37	Reserved	38	USB_D+
39	Reserved	40	GND
41	Reserved	42	LED_WWAN#
43	Reserved	44	LED_WLAN#
45	Reserved	46	LED_WPAN#
47	Reserved	48	+1.5V
49	Reserved	50	GND
51	Reserved	52	+3.3V

CN10: SATA Connector

Pin	Signal
1	Ground
2	TXP
3	TXN
4	Ground
5	RXN
6	RXP
7	Ground

CN11: WLAN LED 1x2 Wafer

Pin	Signal
1	VDD3P3V
2	LED1 WLAN

CN12: WLAN LED 1x2 Wafer

Pin	Signal
1	VDD3P3V
2	LED2 WLAN

CN13: 3.96mm 1x2 POWER (+12V Only)

Pin	Define	Pin	Define
1	GND	2	+12V

CN14 : Console & USB Connector

USB	
Pin	**Define**
1	+5V
2	DATA0-
3	DATA0+
4	GND
5	+5V
6	DATA1-
7	DATA1+
8	GND
RJ45	
Pin	**Define**
9	RTS#
10	N/A
11	SOUT
12	GND
13	GND
14	SIN
15	N/A
16	CTS#

CN15: LAN RJ-45 Connector

Pin	Define
1	TX+
2	TX-
3	RX+
4	Chassis Ground
5	Chassis Ground
6	RX-
7	Chassis Ground
8	Chassis Ground
D1: Speed indicated LED	
1 Gbps	GREEN
100 Mbps	YELLOW
D2 :Link/Activity LED	
Link	GREEN
Activity	BLINKING

CN18: DC +12V Power Jack (2Pin)

Pin	Define	Pin	Define
1	+12V	2	GND

CN19: POWER SWITCH

Pin	Signal
1	GND
2	+12V

Jumper Setting

JP1: WATCH DOG Select

Pin	Setting	
	Open	NON WDT
	Close	WDT

JP2: PCIE GEN1/GEN2 Select

Pin	Setting	
	1-2	GEN2
	2-3	GEN1

JP3: BYPASS Select

Pin	Setting	
	1-2	BYPASS Enabled
	2-3	BYPASS Disabled

Chapter 3. System Setup

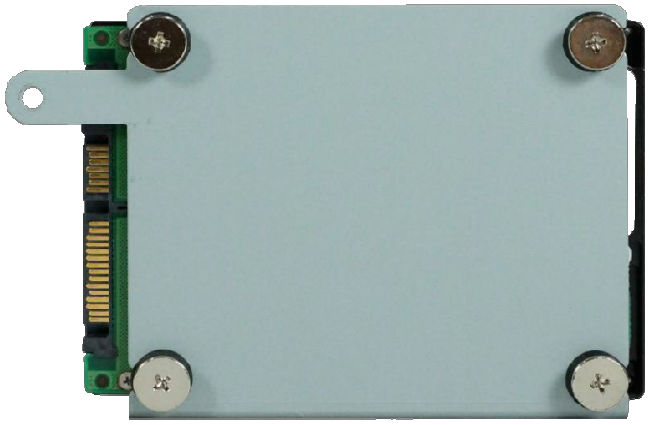
3-1 Case Open

- 3-1-1 Remove screws per fig.1

- 3-1-2 Slide open the case to expose the MB

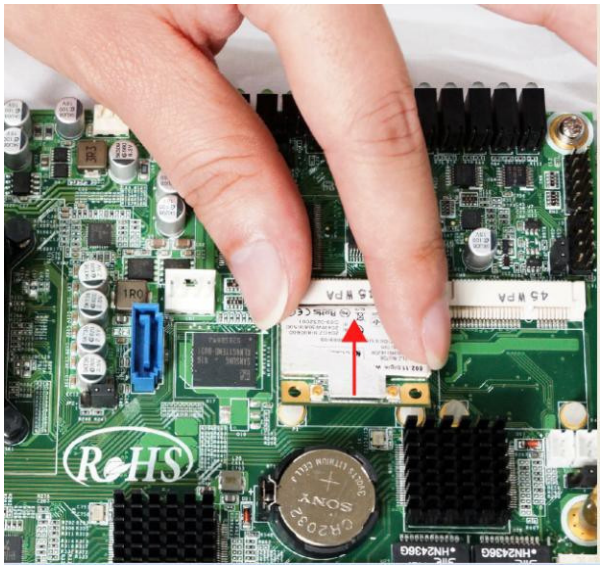
3-2 Install Hard Disk Drive

- **3-2-1 Fix on 3.5” HDD to HDD bracket**



3-3 Install MiniPCIe module

- **3-3-1 Insert your mini PCIe card into the slot**



Chapter 4. U-Boot Reference Manual

The ROM chip of your PL-80720 board is configured with a customized Basic Input/Output System (U-Boot). The U-Boot is a set of permanently recorded program routines that give the system its fundamental operational characteristics. It also tests the computer and determines how the computer reacts to instructions that are part of the programs.

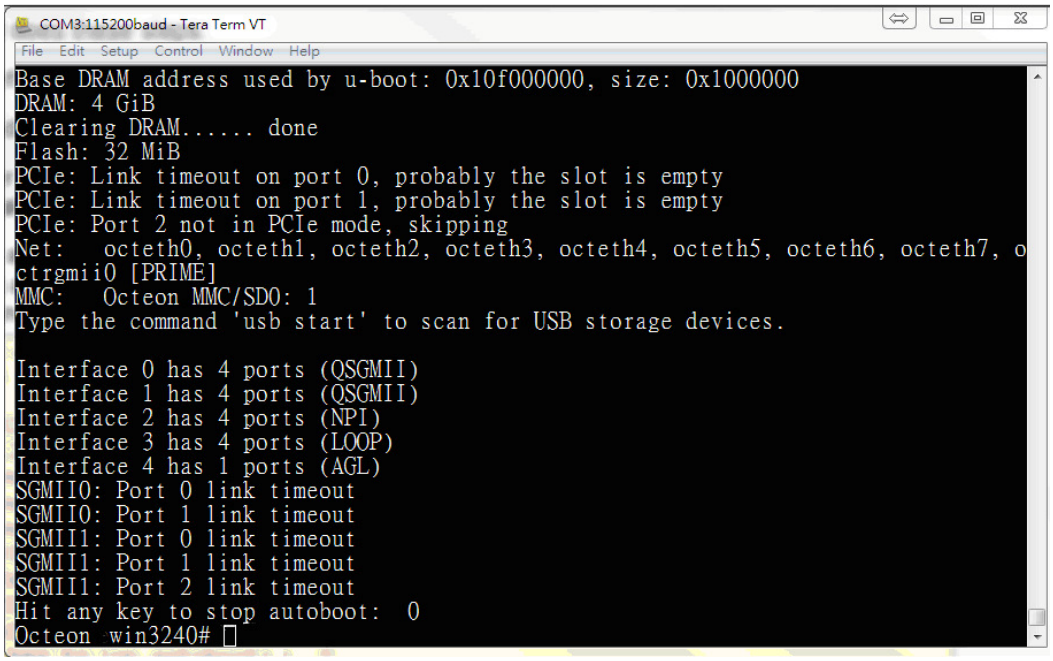
4.1 The boot process

After power-up or reset, the processor loads the U-Boot boot loader in several steps.

- The processor does these steps:
  - ◇ Executes a primary bootstrap that configures the interrupt and exception vectors, clocks, and SDRAM
  - ◇ Decompresses the U-Boot code from flash to RAM
  - ◇ Passes execution control to the U-Boot

- U-Boot does these steps:
  - ◇ Configures the Ethernet MAC address, flash, and, serial console
  - ◇ Loads the settings stored as environment variables in flash
  - ◇ After a few seconds (a length of time you can program), automatically boots the pre-installed kernel

To stop the automatic booting (autoboot) of the pre-installed kernel, send a character to the serial port by pressing a key from the serial console connected to the target. If U-Boot is stopped, it displays a command line console (also called monitor).



4.2 Built-in commands

For a complete list and brief descriptions of the built-in commands, at the U-Boot monitor prompt, enter either of these commands:

- help
- ?

You see a list like this one

The available commands can vary according to the capabilities of your

hardware platform.
For more information about a command, enter:
help (command name)
Example: # help run

Note: As you enter the first letters of a command, U-Boot searches its list of built-in commands until it finds a match. For example, if you enter save or sav or even sa, U-Boot executes the saveenv command.
You need to enter enough letters for U-Boot to determine the command to execute. For example, if you enter loa U-Boot cannot tell whether to execute loadb, loads or loady, and you get an 'Unknown command' message.

4.3 Information on commands

Command	Description
bdinfo	Prints board info structure.
coninfo	Prints console devices and information.
date [MMDDhhmm[[CC]YY][.ss]]	Gets / sets / resets system date/time.
fatinfo <interface> <dev[:part]>	Prints information about the file system from *'dev'* on *'interface.'*
flinfo [bank]	Prints information about the flash memory banks.
fsinfo	Prints information about file systems.
meminfo	Display memory information
mii info <addr>	Prints MII PHY info
mmcinfo	display MMC info
octreginfo	print register information
version	Displays U-Boot version and timestamp.

4.4 MII commands

⬇ To access the Ethernet PHY use this commands:

Command	Description
mii device	Lists available devices.
mii device <device name>	Set current device.
mii read <addr > <reg>	Reads register *'reg'* from MII PHY *'addr'*.
mii write <addr > <reg> <data>	Writes *'data'* to register *'reg'* at MII PHY *'addr'*.
mii dump <addr > <reg>	Displays data of register *'reg'* from MII PHY *'addr'*.

4.5 Network commands

⬇ This table shows the network-related commands:

Command	Description
bootp [loadAddress] [bootFilename]	Boots the image over the network using the BootP/TFTP protocol. If no argument is given, bootp takes the values from the *'loadaddr'* and *'bootfile'* environment variables.
dhcp	Requests an IP address from a DHCP server. If the 'autoload' variable is set to 'yes', also transfers the file to which the 'bootfile' environment variable points to the 'loadaddr" RAM memory address by TFTP.
ping <pingAddress>	Pings the IP address passed as parameter. If the other end responds, you see this message: "host <*pingAddress*> is alive".
tftpboot [loadAddress] [bootfilename]	Using FTP, transfers image *'bootfilename'* into the RAM address *'loadAddress'*.
nfs [loadAddress] [host ip addr:bootfilename]	Using NFS, transfers image *'bootfilename'* into the RAM address *'loadAddress'*.

4.6 USB commands

⬇ To access the USB subsystem, use the usb command, followed by its operations:

Command	Description
usb start	start (scan) USB controller
usb reset	Resets (rescans) USB controller
usb stop [f]	Stops USB [f]=force stop
usb tree	Shows USB device tree
usb info [dev]	Shows available USB devices
usb storage	Shows details of USB storage devices
usb dev [dev]	Shows or set current USB storage device
usb part [dev]	Prints the partition table of one or all USB storage devices
usb read addr blk# cnt	Reads *'cnt'* blocks starting at block *'blk#'* to RAM address *'addr'*
fatload usb <dev[:part]> <addr> <filename>	Reads *'filename'* image from partition *'part'* of USB device *'dev'* into the RAM memory address *'addr'*. If *part* is not specified, partition 1 is assumed.
usbboot	Boots from usb device
usb write addr blk# cnt	write `cnt' blocks starting at block `blk#' from memory address `addr'

4.7 Memory commands

⬇ These commands manage RAM memory:

Command	Description
cmp[.b, .w, .l] addr1 addr2 count	Compares memory contents from address *'addr1'* to *'addr2'* for as many *'count'* bytes, words, or long words.
cp[.b, .w, .l] source target count	Copies memory contents from address *'source'* to *'target'* for as many *'count'* bytes, words, or long words.
go addr [arg ...]	Starts the application at address *'addr'* passing 'arg' as arguments.
md[.b, .w, .l] <address> [# of objects]	Displays the contents of the memory at address 'addr' for as many '[#of objects]' bytes, words, or long words.
mm[.b, .w, .l] <address>	Lets you modify locations of memory, beginning at `address,' which gets auto-incremented.
mw[.b, .w, .l] <address> <value > [count]	Writes 'value' into 'address' for as many 'count' bytes, words, or long words.
nm[.b, .w, .l] address	Lets you modify a fixed location of memory.
protect [on|off] ...	Protects/unprotects NOR sector(s).

4.8 Serial port commands

⬇ Use these commands to work with the serial line:

Command	Description
loadb [off] [baud]	Loads binary file over serial line with offset *'off'* and baud rate *'baud'* (Kermit mode)
loads [off]	Loads S-Record file over the serial line with offset *'off'*
loady [off] [baud]	Loads binary file over the serial line with offset *'off'* and baud rate *'baud'* (Ymodem mode)

4.9 Environment variables commands

⬇ To read, write, and save environment variables, use these commands:

Command	Description
printenv [name ...]	If no variable is given as argument, prints all U-Boot environment variables. If a list of variable names is passed, prints only those variables.
envreset	erase the environment in flash
saveenv	Writes the current variable values to non-volatile memory (NVRAM).
setenv name [value]	If no value is given, the variable is deleted. If the variable is dynamic, it is reset to the default value. If a value is given, sets variable 'name´ to value 'value'.

4.10 I2C commands

Command	Description
i2c bus [muxtype:muxaddr:muxchannel]	show I2C bus info
crc32 chip address[.0, .1, .2] count	compute CRC32 checksum
i2c dev [dev]	show or set current I2C bus
i2c loop chip address[.0, .1, .2] [# of objects]	looping read of device
i2c md chip address[.0, .1, .2] [# of objects]	read from I2C device
i2c mm chip address[.0, .1, .2]	write to I2C device (auto-incrementing)
i2c mw chip address[.0, .1, .2] value [count]	write to I2C device (fill)
i2c nm chip address[.0, .1, .2]	write to I2C device (constant address)
i2c probe [address]	test for and show device(s) on the I2C bus
i2c read chip address[.0, .1, .2] length memaddress	read to memory
i2c write memaddress chip address[.0, .1, .2] length	write memory to i2c
i2c reset	re-init the I2C Controller
i2c speed [speed]	show or set I2C bus speed

4.11 Environment variables

U-Boot uses environment variables to tailor its operation. The environment variables configure settings such as the baud rate of the serial connection, the seconds to wait before auto boot, the default boot command, and so on.

These variables must be stored in either non-volatile memory (NVRAM) such as an EEPROM or a protected flash partition.
The factory default variables and their values also are stored in the U-Boot binary image itself. In this way, you can recover the variables and their values at any time with the envreset command.

Environment variables are stored as strings (case sensitive). Custom variables can be created as long as there is enough space in the NVRAM
.

4.11.1 Simple and recursive variables

Simple variables have a name and a value (given as a string):

```
#   setenv myNumber 123456
#   printenv myNumber
myNumber=123456
```

To expand simple variables, enclose them in braces and prefix a dollar sign:

```
#   setenv myNumber 123456
#   setenv var This is my number: ${myNumber}
#   printenv var
var=This is my number: 123456
```

Recursive variables (or scripts) contain one or more variables within their own value. The inner variables are not expanded in the new variable. Instead, they are expanded when the recursive variable is run as a command, as shown here:

```
#   setenv dumpaddr md.b \${addr} \${bytes}
#   printenv dumpaddr
dumpaddr=md.b ${addr} ${bytes}
#   setenv addr 2C000
#   setenv bytes 5
#   run dumpaddr
0002c000: 00 00 00 00 00    .....
```

You must use the back slash '\' before '$' to prevent variables from being expanded into other variables' values.

4.11.2 Scripts

In U-Boot, a script is made up of variables that contain a set of commands; the commands are executed one after another.
Consider this variable:

```
#   printenv cmd1
setenv var val;printenv var;saveenv
```

If you were to run this script, with run cmd1 the var variable would be created with val value, the value would be printed to the console, and the variables would be saved to either the EEPROM or flash partition dedicated to variables.

```
#   run cmd1
var=val
Saving Environment to Flash...
```

```
Un-Protected 1 sectors
Erasing Flash...
. done
Erased 1 sectors
Writing to Flash... done
Protected 1 sectors
```

Separate the commands in a script with semi-colons. (;). As with recursive variables, this sign must be preceded by a back-slash sign or it is considered the termination of the first command itself.
This is how you would save cmd1:

```
#   setenv cmd1 setenv var val\;printenv var\;saveenv
```

For running commands stored in variables, use the run command and its variables separated by spaces:

```
#   setenv cmd1 setenv var val
#   setenv cmd2 printenv var
#   setenv cmd3 saveenv
#   run cmd1 cmd2 cmd3
```

4.11.3 System variables

U-Boot uses several built-in variables:

4.11.3.1 Common system variables

Variable	Description
autoload	If set to "no" (or any string beginning with 'n'), the **bootp**, or **dhcp** command performs a configuration lookup from the BOOTP / DHCP server but does not try to load any image using TFTP.
autostart	If set to "yes", an image loaded using the **bootp**, **dhcp** or **tftpboot** commands is automatically started (by internally calling the **bootm** command).
baudrate	The baud rate of the serial connection.
bootcmd	Defines a command string that is automatically executed when the initial countdown is not interrupted. Executed only when the **bootdelay** variable is also defined.
bootdelay	Seconds to wait before running the automatic boot process in **bootcmd**.
bootfile	Name of the default image to load with TFTP.
filesize	Contains the size of the last file transferred by TFTP or USB.
fileaddr	The RAM address where the last file transferred by TFTP was placed.
stdin	Standard input system.
stdout	Standard output system.
stderr	Standard error output system.
verify	If set to 'n' or 'no,' disables the checksum calculation over the complete image in the **bootm** command to trade speed for safety in the boot process. Note that the header checksum is still verified.
ipaddr	IP address of the target's Ethernet interface.
netmask	Subnet mask of Ethernet interface.
gatewayip	IP address used as network gateway.
serverip	IP address of the host PC (for remote connections like TFTP transfers).

4.11.3.2 Common system variables

Protected variables

Several variables are of great relevance for the system and are stored in a protected section of NVRAM.

Some of these protected variables are, for example, the serial number of the module and the MAC addresses of the network interfaces, which are programmed during production and normally should not be changed.

4.12 Boot commands

U-Boot runs code placed in RAM, although it also can read from other media. The boot process normally takes place in two steps:

- Reading the OS image from media (Ethernet, flash, USB) into RAM
- Jumping to the first instruction of the image in RAM

4.12.1 From Ethernet

The most common way to boot an image during development is by transferring it using TFTP over the Ethernet interface. You do this with the tftpboot command, passing:

- The address of RAM in which to place the image
- The image file name

```
#   tftpboot <loadAddress> <bootfilename>
```

The TFTP transfer takes place between the serverip address (host) and the ipaddr address (target). The host must be running a TFTP server and have bootfilename archive placed in the TFTP-exposed directory.

For Linux kernel images, if the autostart variable is set to yes, this command directly boots the kernel after downloading it.

4.12.2 From USB

Another way to boot an image is by reading it from a USB flash storage device. The USB disk must be formatted in FAT file system.

To read an image from a USB flash disk, enter:

```
# usb start
# fatload usb <dev>[:partition] <loadAddress> <bootfilename>
```

This command reads file bootfilename from device dev, partition partition of the USB flash disk into the RAM address loadAddress. Device and partition are given as a number (0, 1, 2...).
If no partition is specified, partition 1 is assumed.

4.12.3 Booting images in RAM

After the image is transferred to RAM, you can boot it in following command.

- For Linux images:

```
#   bootm <loadAddress>
```

where loadAddress is the address in RAM at which the image resides.

4.13 Automatic booting

If U-Boot is not interrupted after the delay established in **bootdelay**, the automatic boot process takes place. Automatic booting consists of running what is specified in the **bootcmd** environment variable.
In other words, automatic booting has the same effect as doing the following examples:

```
#   run bootcmd
```

If, for example, if you want to automatically boot a Linux kernel image from eMMC flash, set **bootcmd** like this:

```
# setenv 'bootcmd fatload mmc 1 0x100000 vmlinux.64;bootoctlinux 0x100000 coremask=0x1 mem=0'
```

Note: If bootdelay is set to 0, the autoboot happens immediately after U-Boot starts. To stop the process and enter the monitor, press a key as soon as the first U-Boot output lines appear.

4.14 Firmware update commands

The boot loader, kernel, and other data stored in flash form the firmware of the device. Because U-Boot can write any part of flash, its flash commands can be used to reprogram (update) any part of the firmware. This includes the boot loader itself.

The update process normally takes place in three steps:

- Reading image from media (Ethernet, USB) into RAM memory
- Erasing the flash that is to be updated
- Copying the image from RAM into flash

4.14.1 Updating flash with images in RAM

Flash memory must be updated with images located in RAM memory. You can move images to RAM using either Ethernet or USB (see section 4.2 for more information).

To erase flash and copy the images from RAM to flash, use these commands:

- For eMMC flash memory:

```
# mmc erase blk# cnt
#mmc write addr blk# cnt
```

blk# is the beginning block. **cnt** is the number of block. The block size is 512 bytes. addr is the spurce address.

◆ NOR flash bootloader image update example:

usb start;fatload usb 0 0x100000
u-boot-octeon_win3241nor.bin;bootloaderupdate

◆ eMMC flash bootloader image update example:

mmc dev 1
usb start
fatload usb 0 0x100000 emm-boot.bin
mmc write 0x100000 0 10
fatload usb 0 0x100000 u-boot-octeon_generic_emmc_stage2.bin
mmc write 0x100000 10 3E0
mw.b 0x100000 0 8192
mmc write 0x100000 3F0 10

4.15 other important commands

Following list PL-80720series other important commands:

Command	Description
bootoct	Boot from an Octeon Executive ELF image in memory
bootoctlinux	Boot from a linux ELF image in memory
saveenv	Writes the current variable values to non-volatile memory (NVRAM).
dtt	Read temperature from Digital Thermometer and Thermostat
fatload	load binary file from a dos filesystem
fatls	list files in a directory (default /)
gpio	input/set/clear/toggle gpio pins
i2c	I2C sub-system
lanbypass	LAN bypass control function
md	memory display
mii	MII utility commands
mmc	MMC sub system
mmcinfo	display MMC info
mw	memory write (fill)
reset	Perform RESET of the CPU
sata	SATA sub system
sysled	set SYSTEM LED
usb	USB sub-system

Appendix A: Cable Development Kit

The PL-80720 offers some cables for development use.

DK001

Item & Description	Part No.	Qty
Ethernet Cat.5 Cable 2M/ RoHS	CB-EC5200-01	1
Cross Over 2M Color/ RoHS	CB-CO5204-01	1
RJ45 to DB9 2M Cable/ RoHS	CB-RJDB91-01	1

CB-EC5200-01

CB-CO5202/4-01

CB-RJDB91-01