User's Manual

Computer on Module
COM Ports
Two USB Hosts
LCD
Ethernet
CompactFlash
Audio

MXM-8310

USER INFORMATION

About This Manual

This document provides information about products from EMBEDIAN, INC. No warranty of suitability, purpose, or fitness is implied. While every attempt has been made to ensure that the information in this document is accurate, the information contained within is supplied “as-is” and is subject to change without notice.

For the circuits, descriptions and tables indicated, EMBEDIAN assumes no responsibility as far as patents or other rights of third parties are concerned.

Copyright Notice

Trademarks

The following lists the trademarks of components used in this board.

- ARM is a registered trademark of ARM Limited.
- Linux is a registered trademark of Linus Torvalds.
- WinCE is a registered trademark of Microsoft
- Samsung is a registered trademark of Samsung Electronics.
- All other products and trademarks mentioned in this manual are trademarks of their respective owners.

Standards

EMBEDIAN is under the guidance of ISO 9001 standards.

Warranty

This EMBEDIAN product is warranted against defects in material and workmanship for the warranty period from the date of shipment. During the warranty period, EMBEDIAN will at its discretion, decide to repair or replace defective products.

Within the warranty period, the repair of products is free of charge as long as warranty conditions are observed.

The warranty does not apply to defects resulting from improper or inadequate maintenance or handling by the buyer, unauthorized modification or misuse, operation outside of the product’s environmental specifications or improper installation or maintenance.

EMBEDIAN will not be responsible for any defects or damages to other products not supplied by EMBEDIAN that are caused by a faulty EMBEDIAN product.

Technical Support

Technicians and engineers from EMBEDIAN and/or its subsidiaries and official distributors are available for technical support. We are committed to making our product easy to use and will help you use our products in

your systems.
Before contacting EMBEDIAN technical support, please consult our Web site for the latest product documentation, utilities, and drivers. If the information does not help solve the problem, contact us by e-mail or telephone.

Table of Contents

Chapter

1

Introduction

This Chapter gives background information on the MXM-8310
Section include：

- MXM computer on Module Family
- Block diagram

Chapter 1 Introduction

1.1 MXM Computer on Module Family

MXM embedded ARM computer on modules are a small size, new concept, reliable, low power and powerful embedded ARM computers. MXM modules are widely used in Notebook graphic card. And Embedian is the world first to leverage this form factor into standard industrial design. Most importantly, all RISC-based modules will be pin-to-pin compatible from Embedian to save customers design efforts and extend their product lifetime.

MXM-8310 is based on the new Marvell XScale PXA320 (Monahans-P) processor and runs at 806MHz. The computer on module power consumption is optimized using the new Wireless Intel Speedstep Technology. With small size (66mmx50mm) and very power saving, MXM-8310 features state of the art technology, aiming at low power systems that require high CPU performance and space critical application.

It is designed to meet the needs for embedded networking and graphical systems, especially for high performance but requires low power consumption applications. It is aiming for automatic data collection field such as RFID terminals, batch/ wireless data collection terminals, wireless barcode scanner, POS terminals, thin clients, server PC, biometric access control terminals, automation, transportation, transaction terminals, portable test instrument, advanced remote controller, and GPS systems for retail, light industrial and medical/pharmaceutical applications. With all drivers pre-installed for Linux 2.6.26, people could easily develop their programs and make it time to market in very short time.

Based on Marvell/Intel XSCALE Core SoC, the MXM-8310 includes 128MB of NAND Flash, and 128MB of Mobile DDR. Additional interfaces include three RS-232 ports, Ethernet interface, two USB host ports, a Compact Flash controller that supports true IDE modes, 12 GPIOs, a real-time clock, an LCD controller supporting up to 800x600 displays and an audio AC97 interface.

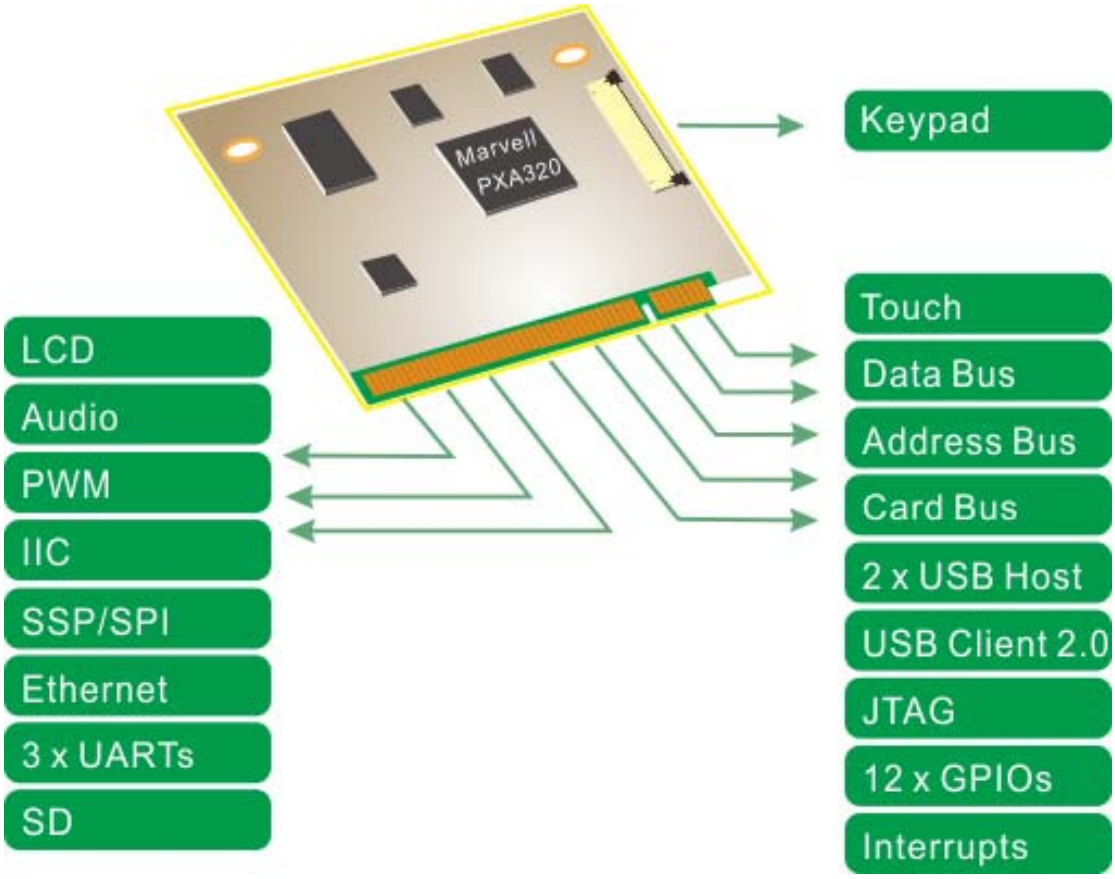
A 242-pin golden finger connector enables the MXM -8310 to interface with the OEM's custom circuitry, and with an evaluation carrier board that is supplied with Embedian's evaluation kit. The evaluation carrier board includes a LCD panel, headers and connectors for all interfaces.

The MXM-8310 is a member of MXM series computer on module line families and is designed in a 66mm x 50mm factor.

1.2 Block Diagram

The following diagram illustrates the system organization of the MXM-8310. Arrows indicate direction of control and not necessarily signal flow.

Figure 1.1 MXM-8310 Block Diagram



Details for this diagram will be explained in the following chapters.

Chapter

2

Specifications

This Chapter contains specifications of MXM-8310.
Section include：

- Functional specifications
- Mechanical specifications
- Electrical specifications
- Environmental specifications
- MTBF
- EMI/RFI and ESD protection

Chapter 2 Specifications

This chapter gives an overall specification for MXM-8310. The specification includes functional, mechanical, electrical, and environmental specifications.

2.1 Functional Specifications

Processor
- Marvell PXA320 (Monahans-P)
- 32-bit XSCALE Core complies with ARM Architecture V5TE instruction set
- Clock Rates up to 806Mhz
- 260Mhz System BUS
- 256KB L2 Cache
- Booting from NAND Flash
- Intel Wireless MMX 2 and SSE integer instruction capability
- 2D Graphic Acceleration
- Ultra-Low Power Consumption

Power Supply
- Single input +5V DC power from 242-pin interface
- Real-time clock battery powered

Memory
- Onboard 128MB NAND Flash (SLC)
- Onboard 128MB Mobile DDR memory (32-bit, 256MB Mobile DDR available on project based)
- CompactFlash(CF), Type I and Type II, 3.3V, True IDE Mode

Universal Serial Bus (USB)
- Chipset：CPU internal
- Two USB 1.1 host ports (12Mbit/s speed)
- OHCI Rev. 1.0 Compliance
- USB legacy keyboard, mouse and hard disk support

USB 2.0 Device
- Chipset: CPU internal
- USB2.0 High Speed
- USB2.0 UTMI interface (need a UTMI transceiver on carrier board)
- Compatible with USB specification version 2.0 and UTMI (Universal Transceiver Macrocell Interface)

COM Port
- Chipset：CPU internal
- Three Serial ports
- Two with TX, RX, CTS and RTS and One with Full RS232.

CompactFlash(CF) Interface

- Chipset：CPU Card Bus
- Type I and Type II, 3.3V
- True IDE mode and I/O mode

Ethernet
- Chipset：Davicom DM9000B
- One 10/100Mbps Ethernet (MAC integrated)
- Compliance with IEEE 802.3u 100Base-TX and 802.3 10Base –T
- Compliance with IEEE 802.3u auto-negotiation protocol for automatic link-type selection
- Full-duplex/half -duplex capability
- Supports IEEE 802.3x full duplex flow control
- Auto-MDIX support

IDE Interface
- Chipset：CPU Card Bus
- ATA PIO Mode

AC97 Audio-Codec Interface
- Chipset：CPU AC-link interface
- Support 16-bit/32-bit sample size receive and transmit FIFO
- AC97 version 2.3 compliance interface
- 1-ch stereo PCM inputs/ 1-ch stereo PCM outputs1-ch MIC input
- Advanced Linux Sound Architecture (ALSA) API support

Discrete I/O
- 12 general-purpose digital I/Os
- 8 External interrupt to eliminate performance hogging polling

IIC Interface
- Chipset：CPU internal
- 1-ch Multi-Master IIC-Bus
- Serial, 8-bit oriented and bi-directional data transfers can be made at up to 100 Kbit/s in Standard mode or up to 400 Kbit/s in Fast mode.

SSP/SPI Interface
- Chipset：CPU internal
- Compatible with 2-ch Serial Peripheral Interface Protocol version 2.11
- 2x8 bits Shift register for Tx/Rx
- DMA-based or interrupt-based operation

Real-Time Clock (RTC)
- Chipset：MAXIM DS1337
- Two Time-of-Day Alarms

Watchdog Timer (WDT)
- Chipset：CPU internal
- 16-bit Watchdog Timer
- Interrupt Request or System Reset at Timeout

CPU Video Graphic Array (VGA)
- Chipset：CPU internal
- TFT Panel Support
- Up to 800x600 resolutions
- 16-bit 65,000 color support
- TTL and 18-bit interface

Pulse Width Modulation (PWM)
- Chipset：CPU Internal
- 4-ch 16-bit Timer with PWM / 1-ch 16-bit internal timer with DMA-based or interrupt-based operation
- Programmable duty cycle, frequency, and polarity

Touch Panel/ADC Interface
- Chipset：CPU Internal ADC
- 10-bit CMOS ADC

System Bus (ISA-like Interface)
- Chipset：CPU internal DFI Bus
- For add-on companion chip
- 8/16-bit connection

JTAG
- Testing and debugging interface

BIOS
- Blob
- Serial or Ethernet TFTP download
- Booting from NAND Flash Technology

Operating System
- Linux 2.6.26, ARM Linux Supports (BIOS, Kernel Stored in NAND and Rootfs in CF)

Cross Toolchain
- Based on gcc 4.1.1 with iwmmx support
- Support EABI
- humb-2 GLIBC binaries

Dimension
- Width x Length (W x L): 66mmx50mm

Packing List
- 1 x MXM-8310

Document Deliverable
- blob (source and binary)
- kernel (source and binary)
- root file systems
- design guide

- user's manual
- carrier reference schematics (pdf and Orcad format)
- cross toolchain

Ordering Information

- MXM-8310 (normal temperature)
- MXM-8310-I (industrial temperature -40^{o}C~85^{o}C)

2.2 Mechanical Specification

The MXM series embedded ARM computer boards is very tiny (66mm x 50mm) in form factor. This section describes the component dimensions and mounting of the board. Detailed drawings are available from Embedian for production customers.

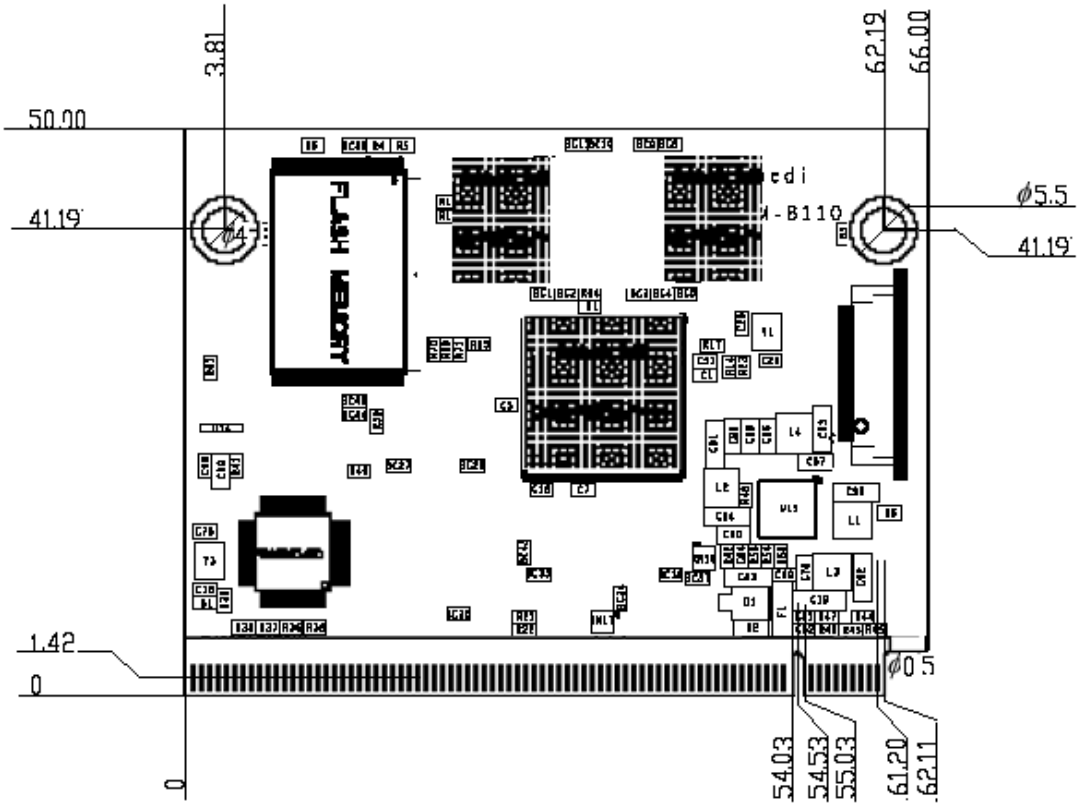
2.2.1. Dimensions

Length x Width: 66mm x 50mm (2.60" x 1.97")

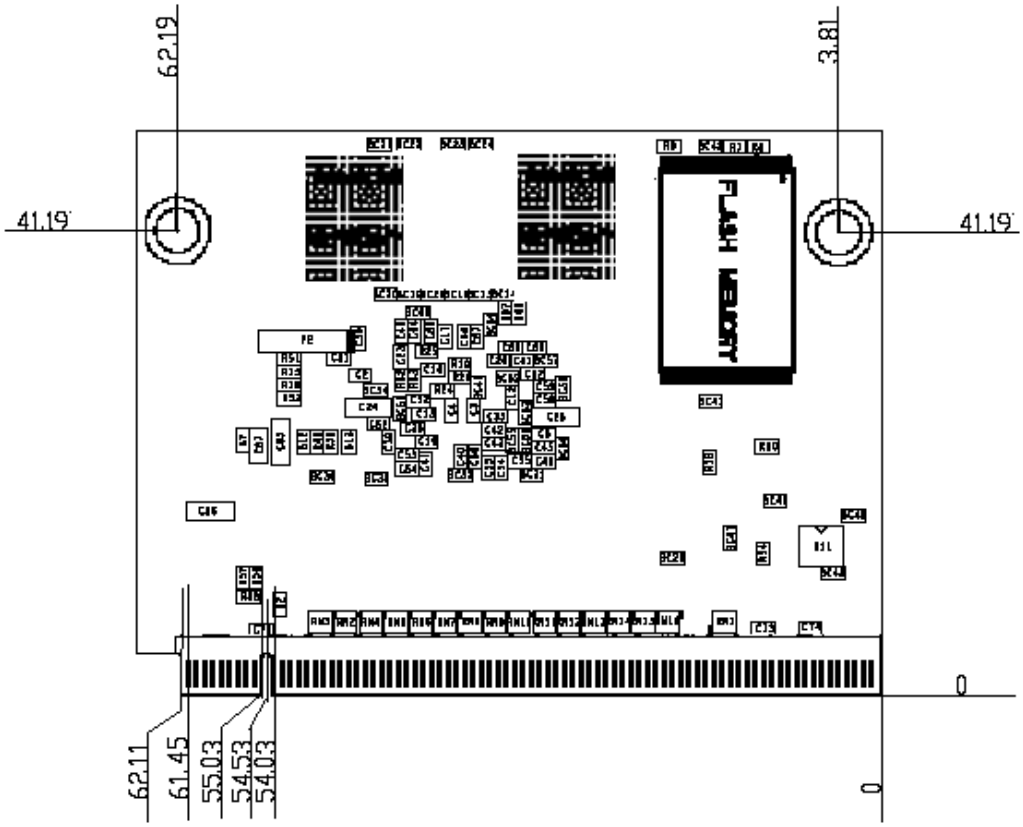
2.2.2. Mechanical Drawing

The following mechanical drawing specifies the dimension of MXM-8310, as well as key components on the board. All dimensions are in mini-meters.

Top View


3.81
62.19
66.00
50.00
41.19
φ4
φ5.5
41.19
FLASH MEMORY
1.42
0
φ0.5
0
54.03
54.53
55.03
61.20
62.11

Bottom View



2.2.3. Mounting Holes

Two mounting holes are provided for mounting. The diameter of the holes is 4.0 mm. (The diameter of the ring is 5.5mm.) Mounting holes are plated through and connected to the MXM-8310 ground plane.
For reliable ground connections, use locking washers (star or split) when securing an MXM-8310 in a carrier board. Make sure that the washers do not extend beyond the limits of the pads provided (5.5mm). A M3 (Metric 3mm), F (Flat) head, 4mm long, 5mm in head diameter, and 1mm head thick screw is recommended.

2.2.4. Clearances

The MXM-8310 has a low profile. Key clearances are as follows:

Height on Top
Max 2.8 mm (110.24 mil)

Height on Bottom
Maximum 2.4 mm (94.49 mil)

Board Thickness
1.2 mm

Clearance over Top and Bottom
6.4 mm

2.2.5. Weight

About 20g (full featured version)

2.3 Electrical Specification

2.3.1. Supply Voltages

- +5V DC power (+/- 5%)

MXM-8310 computer on module require a +5V power supply from custom carrier board.

2.3.2. Supply Voltage Ripple

100mV peak to peak 0 - 20MHz

2.3.3. Supply Current (Typical)

MXM-8310 is a low power consumption computer on module. The power-consumption tests were executed to give an overview of the electrical conditions for several operational states.
The typical power consumption of MXM-8310 is 400mA@5V.

Note:

1. The above data is measure purely on module and the tested LCD resolutions are 640x480 TFT panel.

2.3.4. Real-Time Clock (RTC) Battery

- Voltage range: 1.8V – 3.6V (Typical@3.0V)
- Quiescent current: max. 3uA@3.0 V

2.3.5. CF

- 3.3V only

2.3.6. LCD

The LCD signal control voltage specification is as follows.

- +3.3/5V for TTL level LCD Panel

2.4 Environmental Specification

2.4.1. Temperature

- Operating: -5° C to +75 ° C(*) (with appropriate airflow)
- Non-operating: -10 to +85 ° C (non-condensing)

Note:

(*) The maximum operating temperature is the maximum measurable temperature on any spot on the module's surface. You must maintain the temperature according to the above specification.

2.4.2. Humidity

- Operating: 0 to 95% (non-condensing)
- Non-operating: 0 to 95% (non-condensing)

2.5 MTBF

- System MTBF (hours) : >100,000 hours

The above MTBF (Mean Time Between Failure) values were calculated using a combination of manufacturer's test data, if the data was available, and a Bellcore calculation for the remaining parts. The Bellcore calculation used is "Method 1 Case 1". In that particular method the components are assumed to be operating at a 50 % stress level in a 40° C ambient environment and the system is assumed to have not been burned in. Manufacturer's data has been used wherever possible. The manufacturer's data, when used, is specified at 50°C, so in that sense the following results are slightly conservative. The MTBF values shown below are for a 40°C in an office or telecommunications environment. Higher temperatures and other environmental stresses (extreme altitude, vibration, salt water exposure, etc.) lower MTBF values.

2.6 EMI/RFI and ESD Protection

The MXM-8310 series computer on module incorporates a number of standard features that protect it from electrostatic discharge (ESD) and suppress electromagnetic and radio-frequency interference (EMI/RFI). Transient voltage suppressors, EMI fences, filters on I/O lines and termination of high-frequency signals are included standard on all systems.

MXM-8310 provides surge protection on the input power lines of itself. This is especially important if the power supply wires will be subject to EMI/RFI or ESD. If the system incorporates other external boards, it is the responsibility of

the designer or integrator to provide surge protection on the system input power lines.

Quick Start Guide

To save developer's time, this chapter gives a quick start guide of MXM-8310 evaluation kit.
Section include：

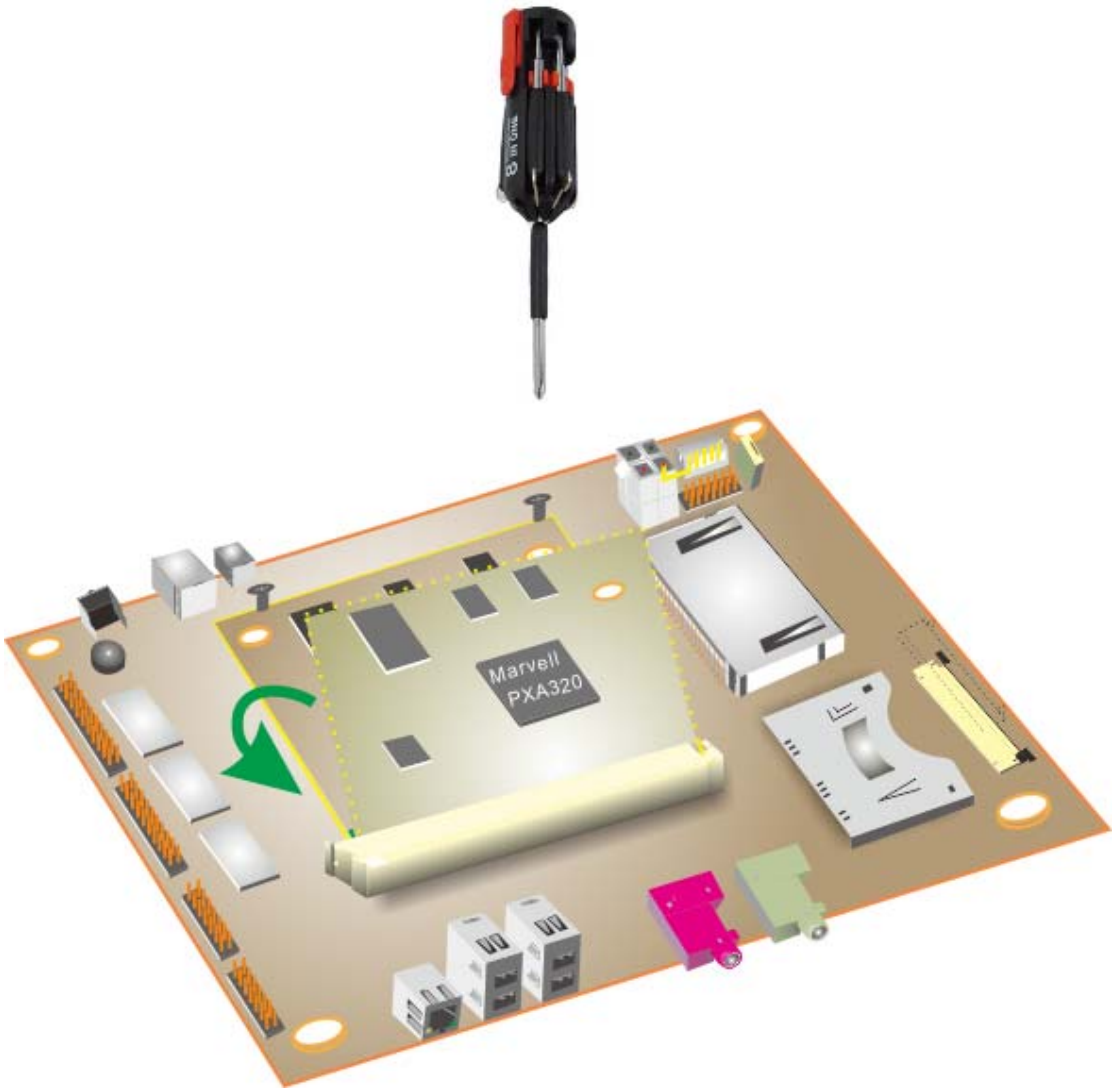
- Plug MXM-8310 into the carrier board and tighten it
- Check Jumper Location
- Connect the Console Debug Cable from evaluation board to your PC.
- Apply Power to the Evaluation Kit
- Network Configuration
- LCD

Chapter 3 Quick Start Guide

These quick start guides are intended to provide developers with simple instructions on how to install MXM-8310 from very beginning and have it monitoring your local device inside of 20 minutes. No advanced installation options are discussed here - just the basics that will work for 95% of users who want to get started. This guide will lead you through the process of configuring, installing, and developing MXM-8310. This guide was written to be as clear as possible and to provide only the details necessary to get you up and running with MXM-8310. Users need MXM-8310 evaluation kit at the development stage. This guide mainly works with the evaluation kit. For more in-depth information, links to other chapters will be located where appropriate.

Step 1：Plug MXM-8310 into the carrier board and tighten it

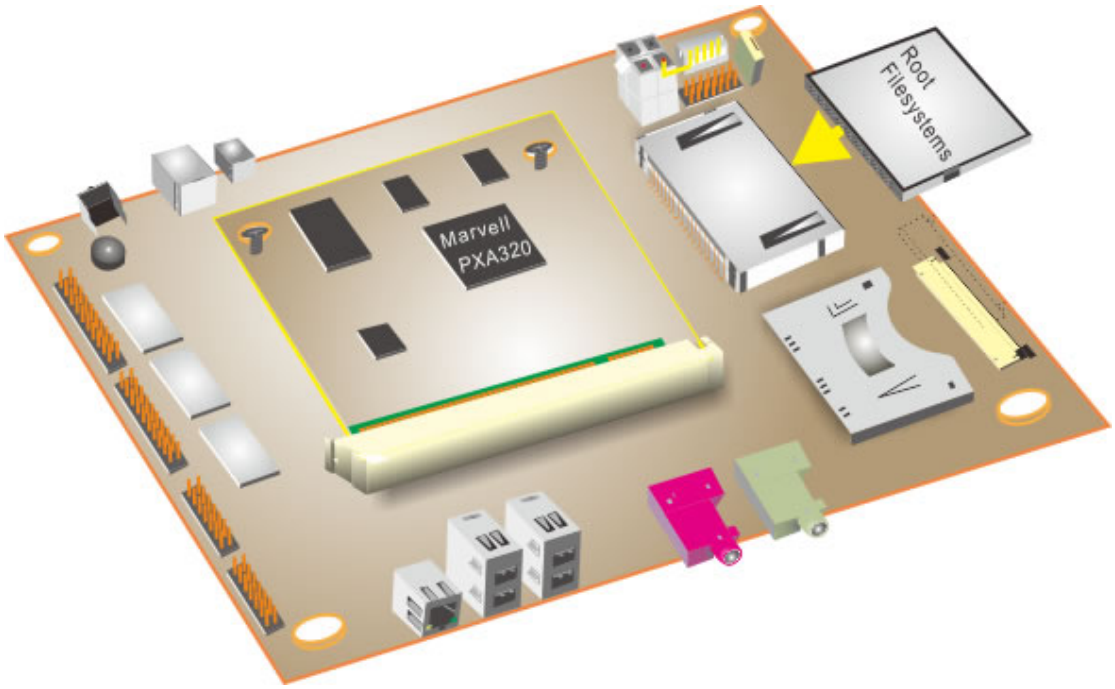
Figure 3.1 Plug MXM-8310 into the carrier board and tighten it



Plug MXM-8310 in the carrier board of the evaluation kit at 45 degrees and press down. Use a cross-head screw driver to tighten it. The recommended screws specification is M3 (Metric 3mm head), F-head (Flat head), 5mm in head diameter and 4mm long.

After done, plug the CF card with pre-loaded file system into CN25 (CF socket) connector. The CF card with pre-loaded file system is part of the evaluation kit.

Figure 3.2 Plug the rootfs pre-installed CF card into evaluation kit



Details in regarding to how to make a pre-loaded file system CF card can be found at section 6.2.

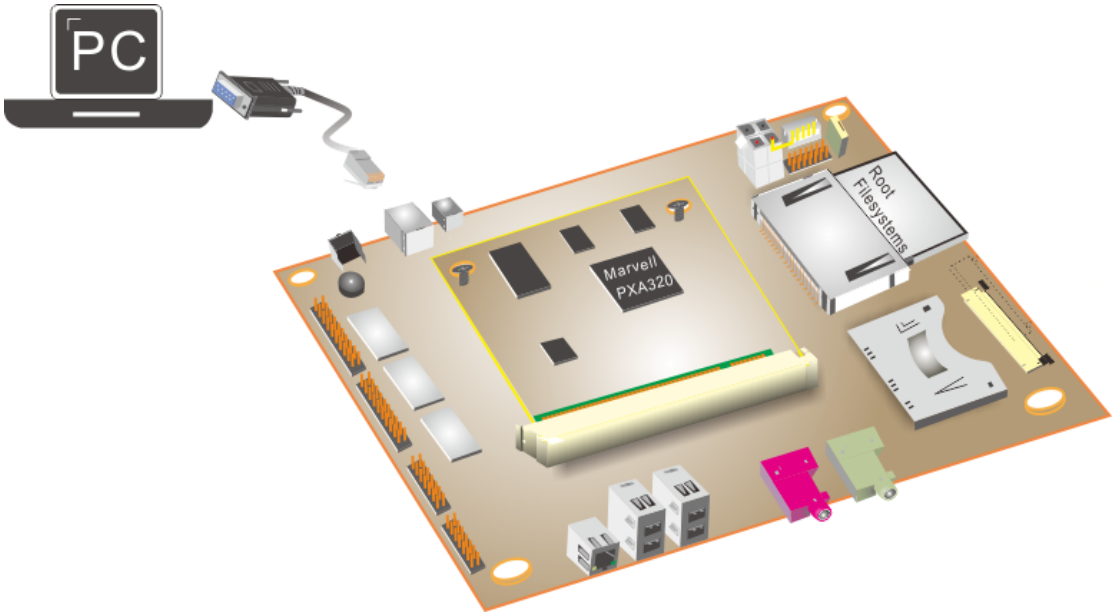
Step 2: Check Jumper Location

Different configurations can be set by several jumper blocks on board. For example, if you attached an LCD, JP4 needs to be shunt depending on your LCD is 5V or 3.3V.

Step 3: Connect the Console Debug Cable from evaluation board to your PC.

Use Embedian console cable and connect from CN6 of the evaluation kit to the COM port of your PC. Open the Hyperterminal program of your PC and set the baud rate as ***115200, 8N1***.

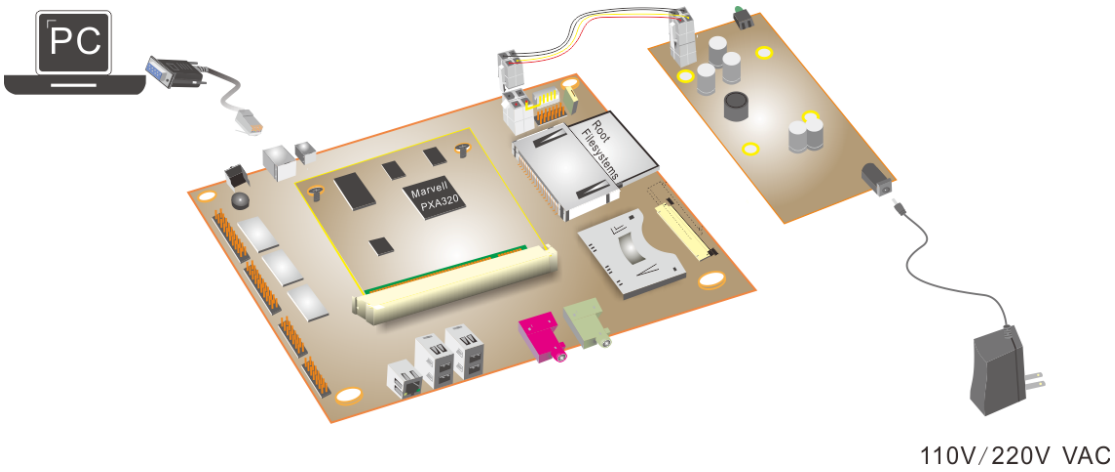
Figure 3.3 Connect Console Debug Cable



Step 4: Apply Power to the Evaluation Kit

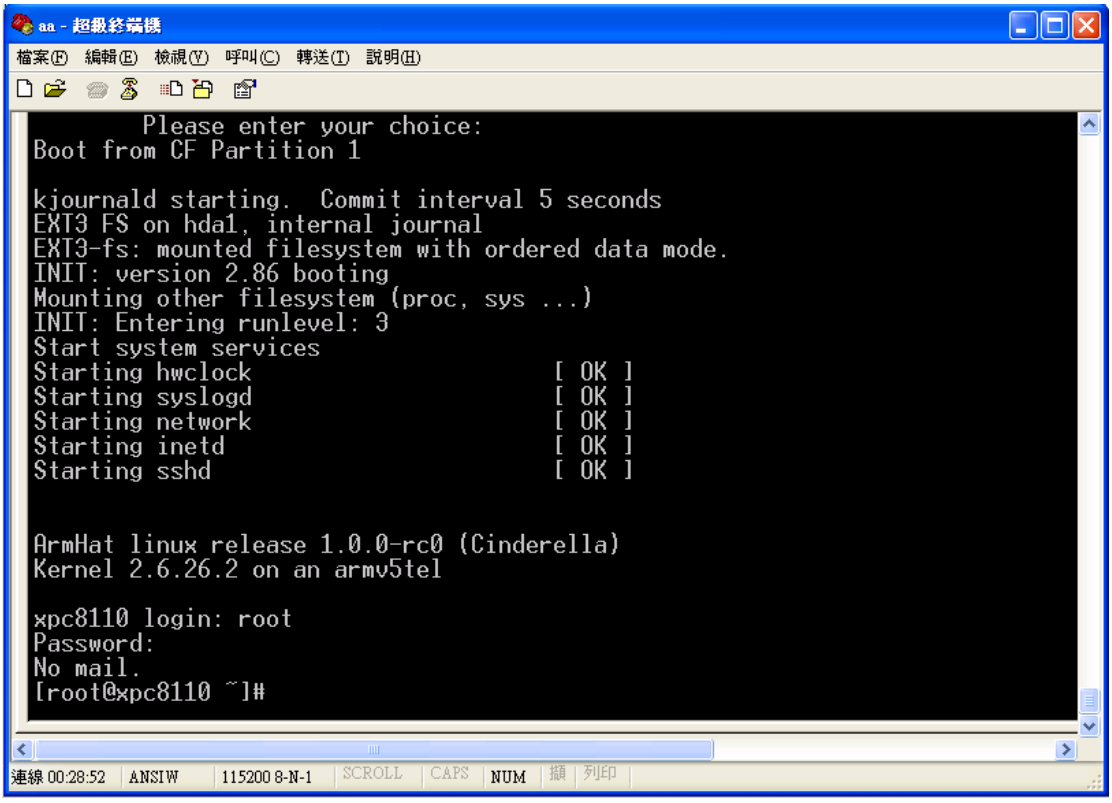
Connect the 12V-2A wall-mount power adapter to the power board and connect the power board to the evaluation board as shown in figure 3.4, the device will be power up. (Note: the power adapter, power board and cables are included in the evaluation kit.)

Figure 3.4 Apply Power to Evaluation Kit



You will see the boot messages from the Hyperterminal as shown in figure 3.5.

Figure 3.5 Boot up messages from Hyperterminal



The default root password is "***xpc8110***" (no quot). You can use ***passwd*** command to change the root password.

Step 5: Network Configuration

Plug an Ethernet cable to CN12 of your device first.

The default IP is set static and network configuration is as follows.

IP address 192.168.1.122
netmask 255.255.255.0
gateway 192.168.1.254

```
[root@xpc8110 ~]# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 10:0D:FF:FF:00:0D
          inet addr:192.168.1.122  Bcast:192.168.1.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:932 errors:0 dropped:0 overruns:0 frame:0
          TX packets:794 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:86083 (84.0 Kb)  TX bytes:146446 (143.0 Kb)
          Interrupt:73 Base address:0x2300
```

Users can use ***ifconfig*** to change the IP address at runtime.

Example:
Below is an example to change the IP address to 192.168.1.123 and netmask to 255.255.255.0 at runtime.

```
[root@xpc8110 ~]# ifconfig eth0 192.168.1.123 netmask 255.255.255.0 up
[root@xpc8110 ~]#
```

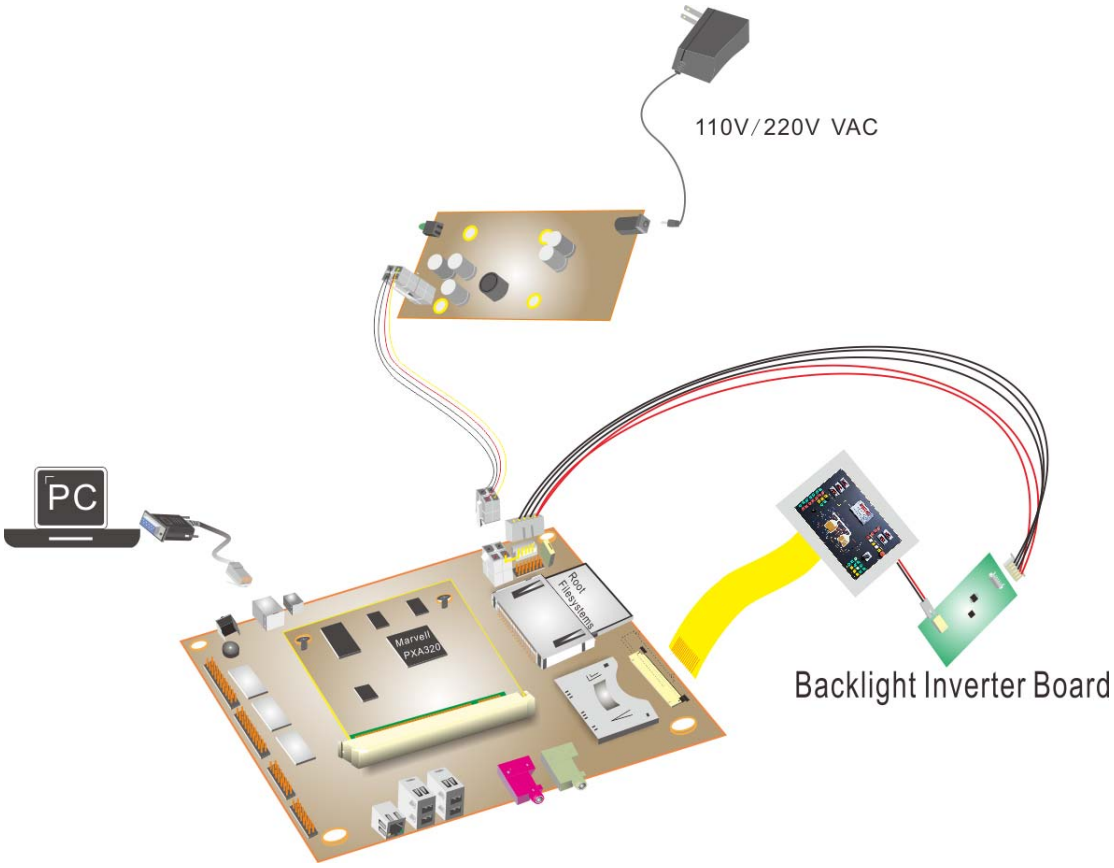
Change Network Configurations Permanently:
The *ifconfig* command only changes the network setting at runtime. After reboot or network restart, the network configuration will be restored to default values. To configure the network configurations at boot or network restart, users need to modify the ***/etc/sysconfig/network-scripts/ifcfg-eth0*** file. Network configuration will take effect at next boot or network restart.

Step 6: LCD

If you need to connect an LCD, use FPC cable or LVDS cable (depending on the type of LCD) connect to the evaluation kit first. Check if JP4 is configured properly. The FPC cable at the board side (CN22) is **top-contacted**. Connect CN2 to the backlight inverter board as shown in figure 3.6. And connect the LCD to the other connector of the backlight inverter board as well. All connectors are in fool proof design and you will not be wrong.

Figure 3.6 shows the LCD connection.

Figure 3.6 LCD Connection



The device descriptor of the LCD is registered as */dev/fb0*. For Embedian default root file systems, there will be no graphic user interface (GUI) outputs to LCD.
To better protect your LCD, the panel power (JP4) and backlight power (pin1 of CN2) is controlled by two switches via two GPIOs and default is set to *off*. You will need to turn it on first by the following command.

```
[root@xpc8110 ~]# modprobe backlight
LCD backlight & panel power control interface for XPC-81xx.
```

This is to load the driver that controls the switches.

```
[root@xpc8110 ~]# echo "1" >> /proc/panel_power
GPIO-88 autorequested
[root@xpc8110 ~]# echo "1" >> /proc/backlight
GPIO-97 autorequested
[root@xpc8110 ~]#
```

This is to turn the switch on.
The LCD driver is a kernel module and you need to load the lcd module first by the following command.

```
[root@xpc8110 ~]# modprobe pxafb
```

Users also need to use *fbset* command to set up the frame buffer first. The settings are located in the file */etc/fb.modes*. After done the above steps, users can *cat* a simple pattern to LCD to see if your LCD is wired correctly.

Chapter

4

Hardware References

This Chapter contains detailed and specialized hardware information.

Chapter 4 Hardware References

This section gives details of the hardware pin out assignment of the MXM-8310.

4.1 Connector Type

The MXM-8310 uses MXM 242-pin golden finder as interface. The connector on module is called header and the connector on custom board is called socket.

Figure 4.1 CN1 Socket connector Type (Mating Connector: B33P102-0013 (Speed Tech), AS0B326-S78N-7F (Foxconn) or compatible)

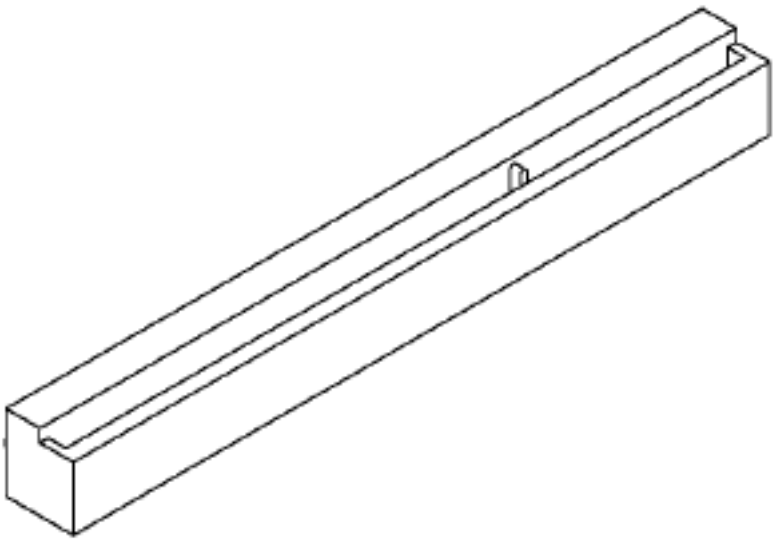
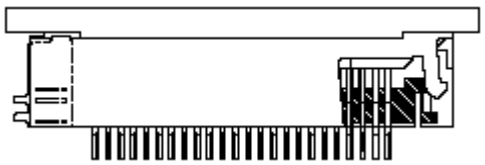


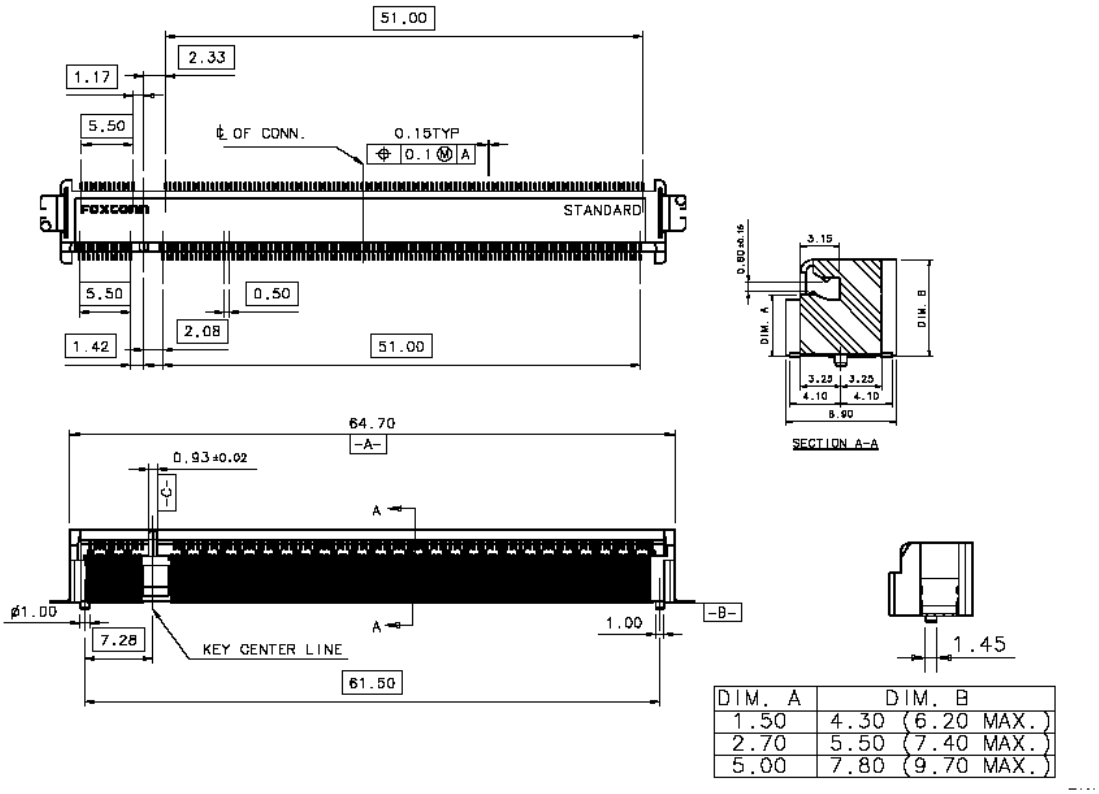
Figure 4.2 CN2 Socket Type (Connector : FPC connector Pitch 0.5mm)



4.2 Connector Mechanical Drawing

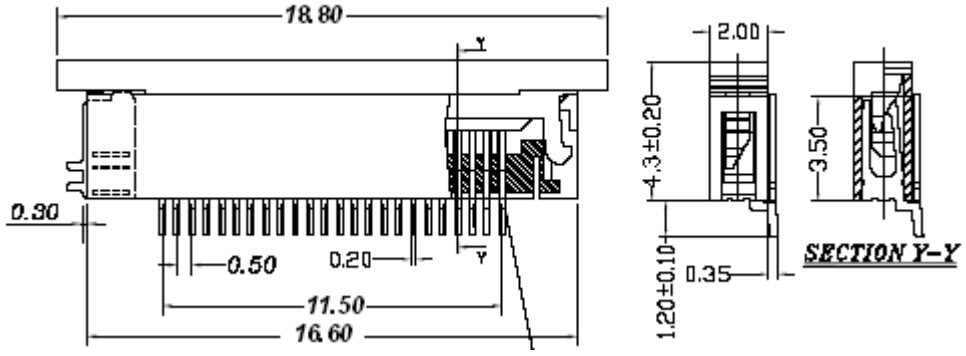
The detail connector mechanical drawing is as follows.

Figure 4.3 CN1 Socket Connector Mechanical Drawing



DIM. A	DIM. B
1.50	4.30 (6.20 MAX.)
2.70	5.50 (7.40 MAX.)
5.00	7.80 (9.70 MAX.)

Figure 4.4 CN2 Socket Connector Mechanical Drawing



4.3 Connector Location

MXM series computer on module uses 242-pin MXM form factor golden finger connectors CN1 as an interface to connect with carrier board. The CN2 in MXM-8310 is mainly for keypad related interface. If your application doesn't need keypad, please skip this connector.

Figure 4.5 Connector Location I

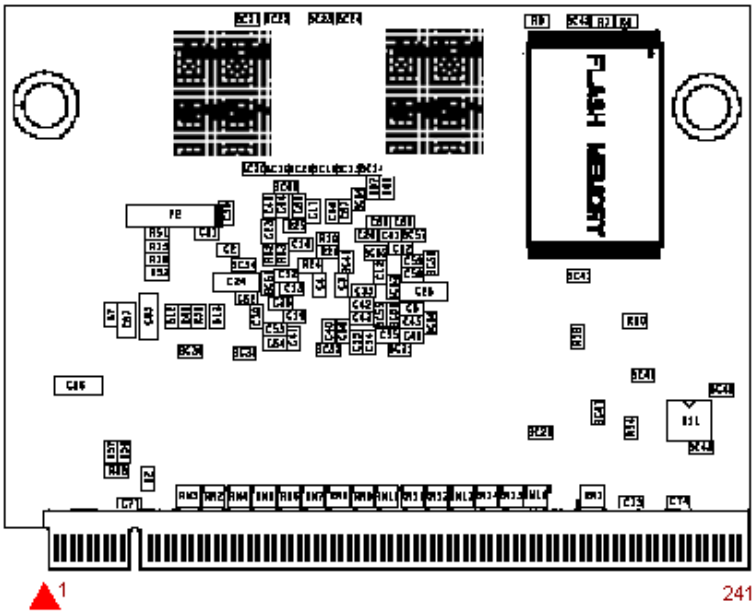
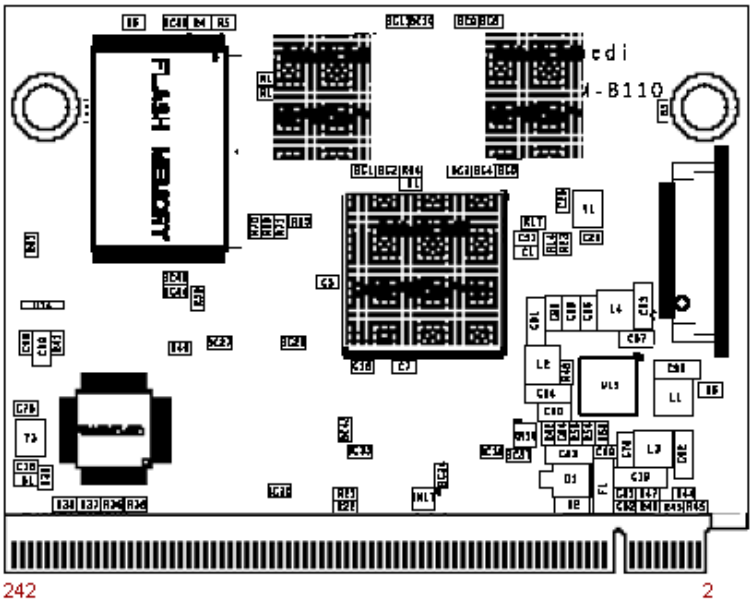


Figure 4.6 Connector Location II



4.4.1. Connector Pin Assignments

The following tables describe the electrical signals available on the connectors of the MXM-8310. Each section provides relevant details about the connector including part numbers, mating connectors, signal descriptions and references to related chapters. For precision measurements of the location of the connectors on the MXM-8310, refer to section 2.2.2. for mechanical drawing.

Legend：

NC	Not Connected
RSVD	Reserved for future platform, suggest open at current design
GND	MXM-8310 Ground Plane

Signal Types：

I	signal is an input to the system
O	signal is an output to the system
IO	signal may be input or output
P	power and ground
A	analog signal
ST	schmitt-trigger

4.4.1.1. CN1 Connector (Golden Finger)

Address bus, data bus, CompactFlash, IDE, JTAG, Ethernet, chip select signal, external interrupt signals and all other CPU related are from CN1.

The following table shows the pin outs of CN1 connector.

Table 4.1 CN1 Connector (Bottom Side)

Table 4.1:	***CN1 Connector (Bottom Side)***			
Description	***Mating Connector : B33P102-0013 (Speed Tech), AS0B326-S78N-7F (Foxconn) or compatible***			
Header	***Pin***	***Signal Name***	***Function***	***Type***
Bottom Side 1 3 241	*4-wire Touch Screen*			
	1	*XP*	*Plus X-axis on-off control signal*	*AI*
	3	*XM*	*Minus X-axis on-off control signal*	*AI*
	5	*YP*	*Plus Y-axis on-off control signal*	*AI*
	7	*YM*	*Minus Y-axis on-off control signal*	*AI*
	Reserved Pin			
	9	*RSVD*	*Reserved*	*N.C.*
	11	*RSVD*	*Reserved*	*N.C.*
	13	*RSVD*	*Reserved*	*N.C.*
	15	*RSVD*	*Reserved*	*N.C.*
	Reserved Pin			
	17	*RSVD*	*Reserved*	*N.C.*
	19	*RSVD*	*Reserved*	*N.C.*
	21	*RSVD*	*Reserved*	*N.C.*
	23	*RSVD*	*Reserved*	*N.C.*
	Key			
	Key			
	Key			
	Key			
	Key			
	Key			
	37	*RSVD*	*Reserved*	*N.C.*
	39	*RSVD*	*Reserved*	*N.C.*
	41	*RSVD*	*Reserved*	*N.C.*
	DMA			
	43	*DMADREQ0*	*External DMA request*	*I*
	Address Bus			
	45	*ADDR0*	*Address Bus*	*O*
	47	*ADDR1*		*O*
	49	*ADDR2*		*O*
	51	*ADDR3*		*O*

	53	ADDR4		O
	55	ADDR5		O
	57	ADDR6		O
	59	ADDR7		O
	61	ADDR8		O
	63	ADDR9		O
	65	ADDR10		O
	67	ADDR11		O
	69	ADDR12		O
	71	ADDR13		O
	73	ADDR14		O
	75	ADDR15		O
	77	GND	Ground Power	P
	Address Bus			
	79	ADDR16		O
	81	ADDR17		O
	83	ADDR18		O
	85	ADDR19		O
	87	ADDR20		O
	89	ADDR21	Address Bus	O
	91	ADDR22		O
	93	ADDR23		O
	95	ADDR24		O
	97	ADDR25		O
	Card Bus Related			
	99	RSVD	Reserved	N.C.
	101	nPCE1	Lower Byte enable for the card interface	O
	103	nPCE2	Higher Byte enable for the card interface	O
	105	nPREG	Attribute Memory Select for the card interface	O
	107	nPIOR	Card interface I/O space output enable	O
	109	nPIOW	Card interface I/O space write enable	O
	111	nIOIS16	Card interface input from I/O space telling size of data bus	I
	113	nPWAIT	Card interface input for inserting wait states	I

	115	GND	Ground	P
	Chip Select			
	117	nGCS0	Chip Select	O
	119	nGCS1		O
	121	nGCS2		O
	123	nGCS3		O
	125	nGCS4		O
	127	nGCS5		O
	129	nWBE0	Write byte enable	O
	131	nWBE1		O
	133	nOE	Output Enable	O
	135	nWE	Write Enable	O
	Data Bus			
	137	DATA0	DATA[15:0] INPUT DATA DURING MEMORY READ AND OUTPUT DATA DURING MEMORY WRITE. BUS WIDTH OF 8/16 BIT IS PROGRAMMABLE	I/O
	139	DATA1		I/O
	141	DATA2		I/O
	143	DATA3		I/O
	145	DATA4		I/O
	147	DATA5		I/O
	149	DATA6		I/O
	151	DATA7		I/O
	153	DATA8		I/O
	155	DATA9		I/O
	157	DATA10		I/O
	159	DATA11		I/O
	161	DATA12		I/O
	163	DATA13		I/O
	165	DATA14		I/O
	167	DATA15		I/O
	169	nWAIT	nWAIT requests	I
	171	CLKOUT	Clock Output	O
	173	WAKEUP	Wakeup requests	I
	175	nRESET_IN	Reset PXA320	ST
	177	nRESET_OUT	Reset External Device	O
	Reserved			
	179	RSVD	Reserved	N.C.
	181	RSVD		N.C.
	183	One Wire	1-Wire bidirectional data bus	I/O
	USB Host 1			
	185	USBH-	USB Host Data -	I/O
	187	USBH+	USB Host Data +	I/O
	Reserved			

	189	RSVD	Reserved	N.C.
	191	RSVD		N.C.
	USB Host 2			
	193	USBH-	USB Host Data -	I/O
	195	USBH+	USB Host Data +	I/O
	197	GND	GND POWER	P
	USB Client 2.0 UTMI(*)			
	199	U2D_XCVR_SELECT	UTMI Transceiver Select	O
	201	U2D_TERM_SELECT	UTMI Termination Select	O
	203	U2D_SUSPENDM_X	UTMI Suspend	O
	205	UTM_LINESTATE1	UTMI Line State	I
	207	UTM_LINESTATE0	UTMI Line State	I
	209	U2D_OPMODE1	UTMI Operating Mode	O
	211	U2D_OPMODE0	UTMI Operating Mode	O
	213	UTM_CLK	UTMI Clock	I
	215	U2D_RESET	UTMI Reset	O
	217	U2D_TXVALID	UTMI Transmit Valid	O
	219	UTM_TXREADY	UTMI Transmit Data Ready	I
	221	UTM_RXVALID	UTMI Receive Data Valid	I
	223	UTM_RXACTIVE	UTMI Receive Active	I
	225	U2D_RXERROR	UTMI Receive Error	O
	227	U2D_DATA0	UTMI Data Bus	I/O
	229	U2D_DATA1	UTMI Data Bus	I/O
	231	U2D_DATA2	UTMI Data Bus	I/O
	233	U2D_DATA3	UTMI Data Bus	I/O
	235	U2D_DATA4	UTMI Data Bus	I/O
	237	U2D_DATA5	UTMI Data Bus	I/O
	239	U2D_DATA6	UTMI Data Bus	I/O
	241	U2D_DATA7	UTMI Data Bus	I/O

Table 4.2 CN1 Connector (Top Side)

Table 4.2:	***CN1 Connector (Top Side)***			
Description	***Mating Connector : B33P102-0013 (Speed Tech), AS0B326-S78N-7F (Foxconn) or compatible***			
Header	***Pin***	***Signal Name***	***Function***	***Type***
	*Reserved Pin*			
	2	*RSVD*	*Reserved*	*N.C.*
	4	*RSVD*		*N.C.*
	JTAG			
	6	*TMS*	*TAP Controller Mode Select*	*I*
	8	*TDO*	*TAP Controller Data Output*	*O*
	10	*TDI*	*TAP Controller Data Input*	*I*
	12	*TCK*	*TAP Controller Clock*	*I*
	14	*nTRST*	*TAP Controller Reset*	*I*
	AC97			
	16	*AC_SYNC*	*48kHz fixed rate sample sync*	*O*
	18	*AC_BIT_CLK*	*12.288MHz serial data clock*	*I/O*
	20	*AC_nRESET*	*AC'97 Master H/W Reset*	*O*
	22	*AC_SDATA_IN*	*AC'97 input stream*	*I*
	24	*AC_SDATA_OUT*	*AC'97 output stream*	*O*
	Key			
	Key			
	Key			
	Key			
	Key			
	Key			
	Power Input			
	38	*EXT5V*	*DC in 5V*	*P*
	40	*EXT5V*	*DC in 5V*	*P*
	42	*EXT5V*	*DC in 5V*	*P*
	44	*EXT5V*	*DC in 5V*	*P*
	Reserved			
	46	*RSVD*	*Reserved*	*N.C.*
	CPU LCD			
	48	*LDD11*	*LCD data bus RED0 (LSB)*	*O*
	50	*LDD12*	*LCD data bus RED1*	*O*
	52	*LDD13*	*LCD data bus RED2*	*O*
	54	*LDD14*	*LCD data bus RED3*	*O*

	56	LDD15	LCD data bus RED4 (MSB)	O
	58	LDD5	LCD data bus GREEN0 (LSB)	O
	60	LDD6	LCD data bus GREEN1	O
	62	LDD7	LCD data bus GREEN2	O
	64	LDD8	LCD data bus GREEN3	O
	66	LDD9	LCD data bus GREEN4	O
	68	LDD10	LCD data bus GREEN5 (MSB)	O
	Reserved			
	70	RSVD	Reserved	N.C.
	72	LDD0	LCD data bus BLUE0 (LSB)	O
	74	LDD1	LCD data bus BLUE1	O
	76	LDD2	LCD data bus BLUE2	O
	78	LDD3	LCD data bus BLUE3	O
	80	LDD4	LCD data bus BLUE4 (MSB)	O
	82	VCLK	LCD clock signal	O
	84	HSYNC	Horizontal synchronous signal	O
	86	VSYNC	Vertical synchronous signal	O
	88	VDEN	Data enable signal	O
	90	GND	Ground Power	P
	PWM			
	92	PWM0	Pulse Width Modulation Output	O
	94	PWM1		O
	96	PWM2		O
	98	PWM3		O
	IIC			
	100	IICSCL	IIC-bus clock	I/O
	102	IICSDA	IIC-bus data	I/O
	SSP(**)			
	104	SSP3_RXD	Synchronous Serial Protocol Receive Data	I
	106	SSP3_TXD	Synchronous Serial Protocol Transmit Data	O

	108	*SSP3_SCLK*	*Synchronous Serial Protocol Serial Clock*	*I/O*
	110	*SSP3_SFRM*	*Synchronous Serial Protocol Serial Frame Indicator*	*I*
	112	*SSP4_RXD*	*Synchronous Serial Protocol Receive Data*	*I*
	114	*SSP4_TXD*	*Synchronous Serial Protocol Transmit Data*	*O*
	116	*SSP4_SCLK*	*Synchronous Serial Protocol Serial Clock*	*I/O*
	118	*SSP4_SFRM*	*Synchronous Serial Protocol Serial Frame Indicator*	*I*
	Interrupt			
	120	*EXT_INT1*	*External interrupt request*	*I*
	122	*EXT_INT2*		*I*
	124	*EXT_INT3*		*I*
	126	*EXT_INT4*		*I*
	128	*EXT_INT5*		*I*
	130	*EXT_INT6*		*I*
	132	*EXT_INT7*		*I*
	134	*EXT_INT8*		*I*
	136	*GND*	*Ground Power*	*P*
	GPIO			
	138	*GPIO1*	*General input/output ports*	*I/O*
	140	*GPIO2*		*I/O*
	142	*GPIO3*		*I/O*
	144	*GPIO4*		*I/O*
	146	*GPIO5*		*I/O*
	148	*GPIO6*		*I/O*
	150	*GPIO7*		*I/O*
	152	*GPIO8*		*I/O*
	154	*GPIO9*		*I/O*
	156	*GPIO10*		*I/O*
	158	*GPIO11*		*I/O*
	160	*GPIO12*		*I/O*
	162	*VCCIO_PWREN*	*External Device Power Control*	*O*
	164	*VCCLCD_PWREN*	*Panel Power Control*	*O*
	166	*BACKLIGHT_EN*	*Panel Backlight Control*	*O*
	168	*LCD_PWREN*	*Panel Signal Control*	*O*

	170	BBAT	RTC Battery Power(DC 3V)	P
	SD Card			
	172	SD_nCD	SD Insert Detect	I
	174	SD_WP	SD Write Protect	I
	176	SDCLK	SD Clock	O
	178	SDCMD	SD receive response/ transmit command	O
	180	SDDAT0	SD receive/transmit data	I/O
	182	SDDAT1	SD receive/transmit data	I/O
	184	SDDAT2	SD receive/transmit data	I/O
	186	SDDAT3	SD receive/transmit data	I/O
	188	GND	Ground Power	P
	Reserved			
	190	RSVD	Reserved	N.C.
	192	RSVD	Reserved	N.C.
	UART			
	194	nRI1	UART ring indicator input signal	I
	196	nDCD1	UART data carrier detect input signal	I
	198	nDSR1	UART data set ready input signal	I
	200	nDTR1	UART data terminal ready output signal	O
	202	nCTS1	UART clear to send input signal	I
	204	nRTS1	UART request to send output signal	O
	206	RXD1	UART receives data input	I
	208	TXD1	UART transmits data output	O
	210	nCTS3	UART clear to send input signal	I
	212	nRTS3	UART request to send output signal	O
	214	RXD3	UART receives data input	I
	216	TXD3	UART transmits data output	O
	218	nCTS2	UART clear to send input signal	I

	220	*nRTS2*	*UART request to send output signal*	*O*
	222	*RXD2*	*UART receives data input*	*I*
	224	*TXD2*	*UART transmits data output*	*O*
	Ethernet			
	226	*LANLED1*	*Ethernet Speed LED*	*O*
	228	*LANLED2*	*Ethernet Link LED*	*O*
	230	*AVDD18*	*1.8V For Transformer*	*P*
	232	*TX-*	*Ethernet Transmits data-*	*O*
	234	*TX+*	*Ethernet Transmits data+*	*O*
	236	*AGND*	*Ethernet Ground*	*P*
	238	*RX-*	*Ethernet Receives data-*	*I*
	240	*RX+*	*Ethernet Receives data+*	*I*
	242	*AVDD18*	*1.8V For Transformer*	*P*

(**) SSP can also be configured as SPI by software.

Table 4.3 CN2 Connector

Table 4.3:	***CN2 Connector***			
Description	***FPC connector (Pitch 0.5mm)***			
Header	***Pin***	***Signal Name***	***Function***	***Type***
	*1*	*VCC33*	*DC 3.3V Output*	*P*
	2	*VCC33*	*DC 3.3V Output*	*P*
	3	*GND*	*Ground*	*P*
	4	*GND*	*Ground*	*P*
	5	*KP_DKIN0*	*Direct Key Inputs*	*I*
	6	*KP_DKIN1*	*Direct Key Inputs*	*I*
	7	*KP_DKIN2*	*Direct Key Inputs*	*I*
	8	*KP_DKIN3*	*Direct Key Inputs*	*I*
	9	*KP_MKIN0*	*Matrix Key Returns*	*I*
	10	*KP_MKIN1*	*Matrix Key Returns*	*I*
	11	*KP_MKIN2*	*Matrix Key Returns*	*I*
	12	*KP_MKIN3*	*Matrix Key Returns*	*I*
	13	*KP_MKIN4*	*Matrix Key Returns*	*I*
	14	*KP_MKIN5*	*Matrix Key Returns*	*I*
	15	*KP_MKIN6*	*Matrix Key Returns*	*I*
	16	*KP_MKIN7*	*Matrix Key Returns*	*I*
	17	*KP_MKOUT0*	*Matrix Key Outputs*	*O*
	18	*KP_MKOUT1*	*Matrix Key Outputs*	*O*
	19	*KP_MKOUT2*	*Matrix Key Outputs*	*O*
	20	*KP_MKOUT3*	*Matrix Key Outputs*	*O*
	21	*KP_MKOUT4*	*Matrix Key Outputs*	*O*
	22	*KP_MKOUT5*	*Matrix Key Outputs*	*O*
	23	*KP_MKOUT6*	*Matrix Key Outputs*	*O.*
	24	*KP_MKOUT7*	*Matrix Key Outputs*	*O*

Software References

This Chapter details how to use the Embedian Linux of MXM-8310 computer on module and its evaluation kit.

Section include：

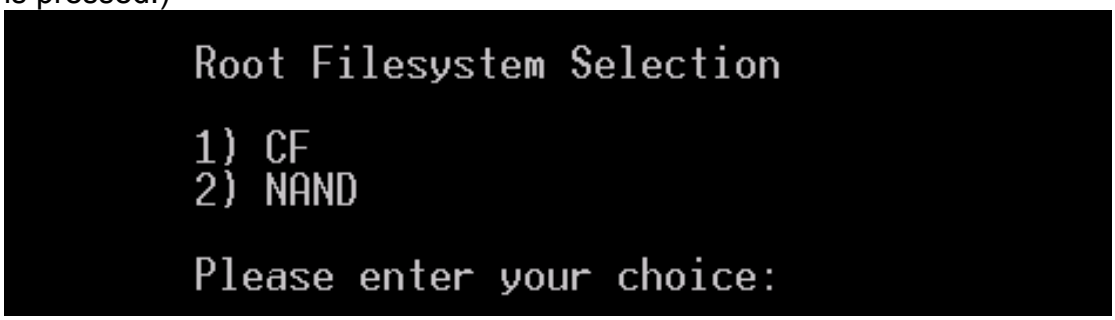
- Booting
- Default root pass and user
- Network Setting
- COM Port
- LCD
- GPIO
- Install Software Packages
- FTP Client
- Tine and RTC
- Telnet/SSH Server
- NAND Root File System
- Cross Toolchain

Chapter 5 Software References

This section gives an introduction in regarding to use Linux on MXM-8310 computer on modules. This guide is mainly focus on the topic related to Embedian' products and not intends to provide with a Linux guide. The Linux on MXM-8310 is pretty much the same as that in Desktop. This guide mainly uses MXM-8310 on the evaluation kit as an example.

5.1 Booting

When power on, the blob will initialize the low-level hardware and bring the Linux kernel to DDR RAM. After that, the Linux kernel will take over the system. The linuxrc is a program that is started in the start-up stage of the kernel prior to the actual boot process. This allows you to boot a small modularized kernel and to load only few drivers that are really needed as modules. linuxrc assists in loading relevant drivers manually. The use of linuxrc provides with the choices to boot into a small root file system in NAND or the complete Linux system in CF card. (The default is set to boot into CF root file systems if no key is pressed.)



The NAND file system is in ext3 format and can be served as a disk-based rescue system or for some simpler applications. For more details in regarding to NAND file system, users can refer to section 5.11.

5.2 Default root pass and user

The default ***root*** password is ***xpc8110*** .

5.2.1 Create a User

To add a user, you can use ***useradd*** command.

5.2.2 Set User Password

After create a user, you can use ***passwd*** command to set the password.

Example:

Below is an example to create a user *john* with home directory and set his password.

```
[root@xpc8110 ~]# useradd -m john
[root@xpc8110 ~]# passwd john
Changing password for john
Enter the new password (minimum of 5, maximum of 8 characters)
Please use a combination of upper and lower case letters and numbers.
New password:
Re-enter new password:
Password changed.
[root@xpc8110 ~]#
```

5.2.3 Delete a User

To delete a user, you can use ***userdel*** command.

Example:

Below is an example to delete a user *john* with removal of home directory and mail spool.

```
[root@xpc8110 ~]# userdel -r john
```

5.3 Network Settings

The default IP is set static and network configuration is as follows.

IP address 192.168.1.122
netmask 255.255.255.0
gateway 192.168.1.254

```
[root@xpc8110 ~]# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 10:0D:FF:FF:00:0D
          inet addr:192.168.1.122  Bcast:192.168.1.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:932 errors:0 dropped:0 overruns:0 frame:0
          TX packets:794 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:86083 (84.0 Kb)  TX bytes:146446 (143.0 Kb)
          Interrupt:73 Base address:0x2300
```

Users can use ***ifconfig*** to change the IP address at runtime.

Example:

Below is an example to change the IP address to *192.168.1.122* and netmask to *255.255.255.0* at runtime.

```
[root@xpc8110 ~]# ifconfig eth0 192.168.1.123 netmask 255.255.255.0 up
[root@xpc8110 ~]#
```

Note:

Every MAC address on board will be mapping to Embedian's serial number and is compliant to ISO/IEC 8802 standards.

5.3.1 Configure Network Configuration at Boot or Network Restart

The *ifconfig* command only changes the network setting at runtime. After reboot or network restart, the network configuration will be restored to default values. To configure the network configurations at boot or network restart, users need to modify the ***/etc/sysconfig/network-scripts/ifcfg-eth0*** file. Network configuration will take effect at next boot or network restart.

5.4 COM Port

There are three COM ports in MXM-8310. Two DUART chips (TL16c752b) are added via system bus on carrier board. This is also a good example to tell users how to add an external chip via system bus. There are 6 RS232s in the evaluation kit. Two of them (console ports and CN10) are from MXM-8310 serial port signals (CPU internal) and four of them (CN8 and CN9) are from external DUART chip TL16C752B.

The device descriptor of COM ports is as follows.

CPU

Console → */dev/ttyS1*
CN10 → */dev/ttyS2*

The RS232s from CPU is built in the Linux kernel and ready to use.

TL16c752B

To use these COM ports, you need to load the drivers first by the following command.

 # modprobe 8250

```
[root@xpc8110 /dev]# modprobe 8250
Serial: 8250/16550 driver $Revision: 1.90 $ 4 ports, IRQ sharing disabled
serial8250.0: ttySA0 at MMIO 0x14000000 (irq = 142) is a 8250
serial8250.0: ttySA1 at MMIO 0x14000040 (irq = 143) is a 8250
serial8250.0: ttySA2 at MMIO 0x14000080 (irq = 144) is a 8250
serial8250.0: ttySA3 at MMIO 0x140000c0 (irq = 145) is a 8250
[root@xpc8110 /dev]# _
```

CN9 → */dev/ttySA0* (pin 1~9) and */dev/ttySA1* (pin 11~19)
CN8 → */dev/ttySA2* (pin 1~9) and */dev/ttySA3* (pin 11~19)

Note:

1. Users can add init script in */erc/rc.d/rc3* file. For example, if you add *modprobe 8250* into that file, COM ports drivers will be loaded at boot.

2. *minicom* program is pre-installed in the rootfs, users can use this program to test the COM port device first.

5.5 LCD

The device descriptor of the LCD is */dev/fb0*. For Embedian default root file systems, there will be no outputs to LCD.
To better protect your LCD, the panel power (JP4) and backlight power (pin1 of CN2) is controlled by two switches via two GPIOs and default is set to *off*. You will need to turn it *on* first by the following command.

 # modprobe backlight

This is to load the GPIO driver module for the control of panel power and backlight.

 # echo "1" >> /proc/panel_power
 # echo "1" >> /proc/backlight

This is to turn on the switches that control the panel power and backlight. You can echo them to 0 to set it back to off.

The LCD driver is a kernel module and you need to load the module first by the following command.

 # modprobe pxafb

Users also need to use *fbset* command to set up the frame buffer first. For example,

 # fbset VGA16

The settings are located in the file */etc/fb.modes*. Different LCDs have different settings. You can add your own LCD settings into this file and fbset it. After done the above steps, users can *cat* a simple pattern to LCD to see if your LCD is wired correctly.

5.6 GPIO

GPIO driver is default built in the kernel instead of driver module.

Below is the sample code for GPIO.

```
/*
 * This program demostrates the contorl of XPC-8xxx GPIO ports by using
device descriptor.
 *
 * Device Descriptor:    /dev/gpioctl
 * Operations:
 *       Read:
 *            Returns "GPIO Port Descriptor" representing current GPIO
settings.
```

```
 *       Write:
 *            Setup the GPIO ports by using "GPIO Port Descriptor".
 *
 * GPIO Port Descriptor:
 *       The GPIO Port Descriptor contains 12 bytes each for one GPIO port
from J0 to J11.
 *       Each byte has following format:
 *            Bit[3:2]   Function 0 = Input,      1 = Output,    2 = Special,   3     =
Reserved
 *            Bit[1]          Pullup          0 = Enable,    1 = Disable
 *            Bit[0]          Data            0 = Low, 1 = High
 */


#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <stdio.h>
#include <errno.h>

char *dev_desc = "/dev/gpioctl";

struct gpioctl_desc {
   unsigned int dat:1;                  // bit 0
   unsigned int pullup:1;               // bit 1
   unsigned int func:2;                 // bit 3:2
} __attribute__ ((packed));

void inline byte_to_desc(unsigned char *byte, struct gpioctl_desc *gpio)
{
   gpio->dat      = (*byte >> 0) & 0x1;
   gpio->pullup  = (*byte >> 1) & 0x1;
   gpio->func     = (*byte >> 2) & 0x3;
}

void inline desc_to_byte(unsigned char *byte, struct gpioctl_desc *gpio)
{
   *byte =   ( (gpio->func & 0x3) << 2) |
         ( (gpio->pullup      & 0x1) << 1) |
         ( (gpio->dat   & 0x1) << 0) ;
}

int read_dev(unsigned char *buf)
{
   struct gpioctl_desc *gpio = malloc(sizeof(*gpio));
   char *str;
   int fd, i;

               fd = open(dev_desc, O_RDONLY);
```

```
    if (fd == -1)
        return -ENODEV;

    i = read(fd, buf, 12);
    close(fd);

    for (i=0; i<12; i++) {
        byte_to_desc(&buf[i], gpio);
        switch (gpio->func) {
            case 0:     str = "Input";      break;
            case 1:     str = "Output";         break;
            case 2:     str = "System";         break;
            default:  str = "Reserve";   break;
        }

        printf("GPJ[%d]: function = %s, pullup = %s, data = %d\n",
                i, str, gpio->pullup ? "Disable" : "Enable", gpio->dat);
    }

    return 0;
}

int write_dev(unsigned char *buf)
{
    struct gpioctl_desc *gpio = malloc(sizeof(*gpio));
    char *str;
    int fd, i;

    for (i=0; i<12; i++) {
        gpio->func = 1;
        gpio->pullup = 1;
        gpio->dat = 1;
        desc_to_byte(&buf[i], gpio);
    }

            fd = open(dev_desc, O_WRONLY);
    if (fd == -1)
        return -ENODEV;

    write(fd, buf, 12);
    close(fd);

    return 0;
}

int main()
{
    unsigned char buf[12];
    int i;
    int err;
```

```
    err = read_dev(buf);
    if (err)
        return err;

    return write_dev(buf);
}
```

5.7 Install Software Packages

Unlike FC or Ubuntu systems, users cannot use *yum install* or *apt-get install* to install a software package on MXM-8310 evaluation kit. However, GCC 3.4.6 is pre-installed in Embedian Linux system. Users can install a software package from a source tarball. This will be exactly the same as that in a PC system.
Users can also compile their source tarballs from cross toolchain at their host PC. Details can be referred to section 5.12.

5.8 FTP Client

The ***lftp*** is default included in the root file system. You might use other dedicated ftp client by compiling from the source tarball. To use the *ltfp* FTP client, assuming the remote host IP address is *59.124.115.43* and the user is *eric*,

```
ubuntu@ubuntu:~$ lftp -u eric 59.124.115.43
Password:
lftp eric@59.124.115.43:~> bye
ubuntu@ubuntu:~$
```

You can use ***put <filename>*** to put transmit a file from local device to remote server and ***get <filename>*** to get a file from remote server to local device, and use ***bye*** to exit the lftp command mode.
You can also use ***wget*** command to get the file from webserver.

5.9 Time and RTC

Users can use *date* command to set the system runtime clock.

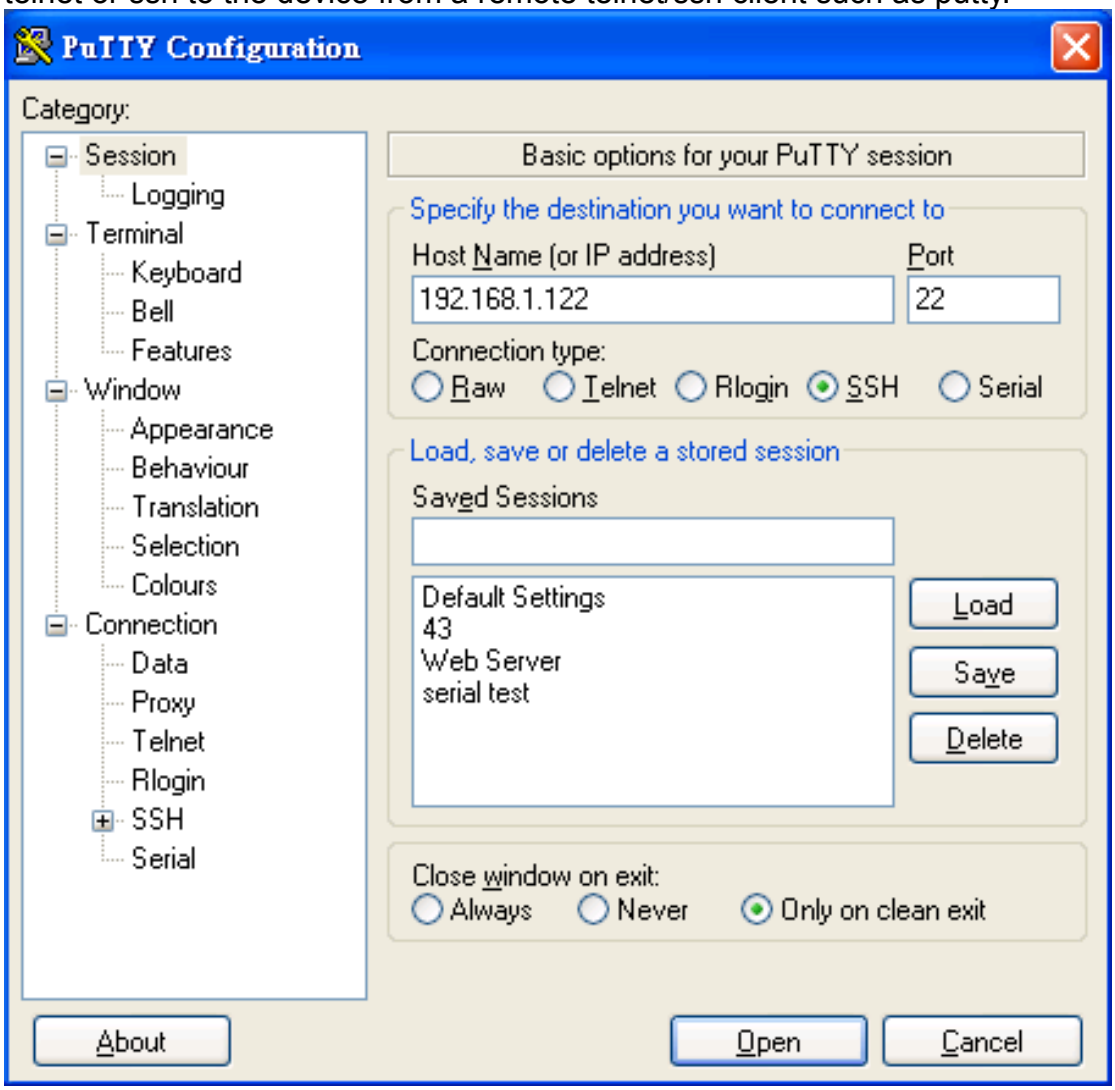
 # date MMDDhhmmYY

The system clock will be restored to default at next reboot. To save the system into hardware, use the following command.

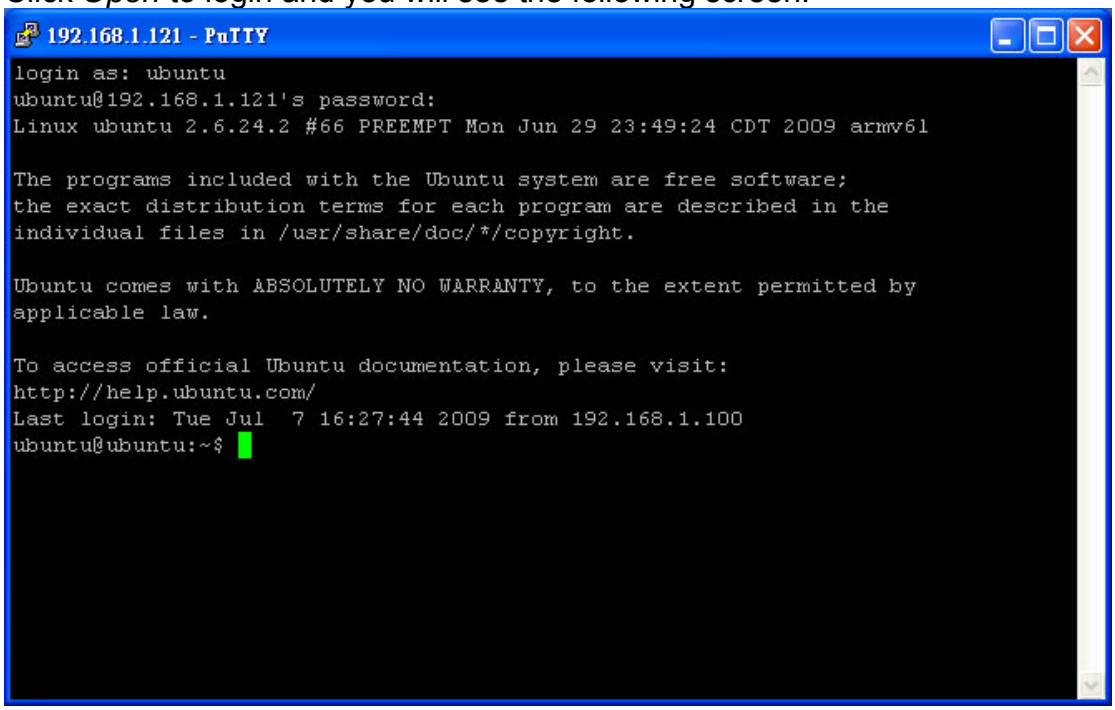
 # hwclock --systohc

5.10 Telnet/SSH Server

The telnet and ssh server are default included in the root file system. You can telnet or ssh to the device from a remote telnet/ssh client such as putty.



Click *Open* to login and you will see the following screen.



5.11 NAND Root File System

The *linuxrc* file in the NAND flash determines where the root file system should boot into. This section mainly introduces the NAND file system.

5.11.1 linuxrc

The *linuxrc* is a program that is started in the start-up stage of the kernel prior to the actual boot process. This allows you to boot a small modularized kernel and to load the few drivers that are really needed as modules. *linuxrc* assists in loading relevant drivers manually.

The use of *linuxrc* provides with the choices to boot into a small root file system in NAND or the complete Linux system in CF card. (If no press anything, the default is set to boot into CF card.)

The *linuxrc* file is located in the NAND flash. User can edit it if they purely want to use NAND flash as their main root file system. There are two ways to access *linuxrc*.

First, if user boot into CF Linux file systems, the NAND flash will be mounted automatically. And user can just access the file that is located at */nand* directory.

```
ubuntu@ubuntu:/$ cd /nand
ubuntu@ubuntu:/nand$ ls
bin  etc  linuxrc     mnt   root  selinux  tmp  var
dev  lib  lost+found  proc  sbin  sys      usr
ubuntu@ubuntu:/nand$
```

Second, users can boot in NAND flash first by pressing *2) NAND* during the booting process. (The root pass is ***apc7110*** by default.) The *linuxrc* file is located at / directory. The NAND file system is also an EXT3 file system. Users can edit the file just you do in any Linux PC.

```
[root@apc7110 /]# ls
bin  etc  linuxrc     mnt   root  selinux  tmp  var
dev  lib  lost+found  proc  sbin  sys      usr
[root@apc7110 /]#
```

5.11.2 As a rescue file system

The NAND file system can play a role of rescue file system, especially when the main root file system in CF card is corrupted or cannot boot into for some reason. Here we would like to give you a guide to restore the CF root file system from NAND file system.

1. Boot into NAND flash first by pressing *2) NAND* during booting process and login as *root* privilege. (The root pass is ***apc7110*** by default.)
2. Prepare for a at least 1GB CF card.
3. The NAND file system will mount partition 1 of CF card by default (The device descriptor of CF device is */dev/hda,* and the partition 1 of CF card is */dev/hda1*). Here would like to format the partition 1 of CF card as EXT3 first.

```
[root@xpc8110 /]# mkfs -t ext3 /dev/hda1
mke2fs 1.37 (21-Mar-2005)
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
489600 inodes, 978290 blocks
48914 blocks (5.00%) reserved for the super user
First data block=0
30 block groups
32768 blocks per group, 32768 fragments per group
16320 inodes per group
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912, 819200, 884736

Writing inode tables: done
Creating journal (8192 blocks): done
Writing superblocks and filesystem accounting information: done

This filesystem will be automatically checked every 32 mounts or
180 days, whichever comes first.  Use tune2fs -c or -i to override.
[root@xpc8110 /]#
```

4. Mount the partition 1 of the CF card as */mnt* and *cd* to */mnt* directory

```
[root@xpc8110 /]# mount -t ext3 /dev/hda1 /mnt
[root@xpc8110 /]# cd /mnt
[root@xpc8110 /mnt]#
```

5. Ftp the rootfs tarball into this directory. Let's assume that the root file system is located at 192.168.1.10 ftp server.

```
[root@xpc8110 /mnt]# ftp 192.168.1.10
Connected to 192.168.1.10.
220 (vsFTPd 2.2.2)
Name (192.168.1.10:root): eric
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> get 320rootfs_20090918.tar.gz
200 PORT command successful. Consider using PASV.
150 Opening BINARY mode data connection for 320rootfs_20090918.tar.gz (231933804
 bytes).
226 Transfer complete.
231933804 bytes received in 164 seconds (1413417 bytes/s)
ftp>
```

6. Extract this tarball (You might need to set the system time by *date* command first.)

```
[root@xpc8110 /mnt]# tar xvfz 320rootfs_20090918.tar.gz
```

7. remove the tarball and exit */mnt* directory

```
[root@xpc8110 /mnt]# rm 320rootfs_20090918.tar.gz
[root@xpc8110 /mnt]# cd ../
[root@xpc8110 /]#
```

8. *umount /mnt*
9. *Reboot*

Your CF root file system is restored.

5.11.3 As a small root file system

At development stage, it is recommended that user develop their program under CF root file system. Users can use *gcc* to do natively make first. After development work done, you can copy the new binary files to NAND flash and do the test again. And then modify the *linuxrc* to boot into NAND flash only.

The other alternative is to use the cross compiler to develop your application at PC. After you done the development, you can ftp the program into the NAND flash and make a test. You can also do this way when developing your program at CF root file system.

5.12 Cross Toolchain

For kernel compile, since it doesn't rely on any libraries and is totally independent, we do suggest use this cross-compile tool that could save lots of time, and no problem at all for applications.

For applications, we do suggest you switch to native compile mode since the host pc which used to make the s/w doesn't know the s/w environment of target platform. There is a gcc compiler in CF root file systems.

IF YOU ARE USING ROOTFS IN CF CARD, WE STRONGLY SUGGEST USE NATIVE COMPILE MODE, at least, at the final stage of test.

The crosss toolchain version that we are using is 4.1.1 with iwmmxt and EABI supported. The file name is arm-iwmmxt-linux-gnueabi-4.1.1-gpl.tgz that can be downloaded from Embedian FTP site.

In this section, we introduce how to install the cross toolchain first. Last, we will lead you how to build blob and kernel zImage.

5.12.1 Installing Toolchain

Building the tool chain is not a trivial exercise and for most common situations pre-built tool chains already exists. Unless you need to build your own, or you want to do it anyway to gain a deeper understanding, then simply installing and using a suitable ready-made tool chain is strongly recommended.

Please follow the commands below and install the toolchain in the directory mentioned below:

 # mkdir –p /usr/local/arm
 # tar xvfz arm-iwmmxt-linux-gnueabi-4.1.1-gpl.tgz

The above command will generate the ***arm-linux-4.1.1*** folder under the same directory as you made the commands. Move this folder to ***/usr/local/arm*** directory.

 # mv arm-linux-4.1.1 /usr/local/arm/
 # export PATH=$PATH:/usr/local/arm/arm-linux-4.1.1/bin

As of now, you have installed the cross toolchain into your Linux PC.
At your application that you would like to cross complied, you need to modify the *Makefile* and point the CROSS_COMPILE to

CROSS_COMPILE = /usr/local/arm/arm-linux-4.1.1/bin/arm-linux-

5.12.2 Build Blob

1. Extract *blob_20091002.tar.gz* file.
2. *# ./config.sh MonahansP*

See Appendix I for firmware update. Unless necessary, we do not recommend you flash bootloader. It might cause to boot failure.

5.12.3 Build kernel zImage

1. Download official Linux mainline 2.6.26.2 from
ftp://ftp.kernel.org/pub/linux/kernel/v2.6/
2. *# tar xvfj linux-2.6.26.2.tar.bz2*
3. *# cd linux-2.6.26.2*
4. *# gzip -d -c ../patch.10162009.gz | patch -p0*
5. *# make xpc8100_defconfig*

Backup and Restore the Root File System in CF Card

This Chapter details how to backup and restore the root file systems in CF card of MXM-8310/XPC-8310.

Section include：

- Backup the root file systems in CF card
- Restore the root file systems in CF card

Chapter 6 Backup and Restore the Root File System in CF Card

This chapter gives an instruction in regarding to how to backup and restore the root file systems in CF card. First, we would like to detail how to backup the root file system in CF card and next, we would like to tell you how to restore the root file system in CF card.

6.1 Backup the root file system in CF card

After developing your program under the Embedian default root filesystem, users might want to backup the whole file system. In this section, we will tell users how to backup the whole root file system.

Take the CF card off from the device and plug it into a USB CF card reader and plug the card reader into the USB port of your Linux PC. The operating system of the Linux PC in this example is Ubuntu 11.04 and the CF card storage is 4GB.

Use the # *fdisk -l* command to list your disk information and find the device descriptor of you CF USB reader.

```
root@dns3:~# fdisk -l

Disk /dev/sda: 1000.2 GB, 1000203804160 bytes
255 heads, 63 sectors/track, 121601 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x000d8811

   Device Boot      Start         End      Blocks   Id  System
/dev/sda1   *           1       24316   195311616   83  Linux
/dev/sda2           24316       24565     1999872   82  Linux swap /
Solaris
/dev/sda3           24565      121602   779448320   83  Linux

Disk /dev/sdb: 4009 MB, 4009549824 bytes
124 heads, 62 sectors/track, 1018 cylinders
Units = cylinders of 7688 * 512 = 3936256 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0xf578ad7b
```

```
   Device Boot      Start         End      Blocks   Id  System
/dev/sdb1               1        1018     3913161   83  Linux
root@dns3:~#
```

We can see the device descriptor of the USB CF card reader is in disk */dev/sdb* and there is one partition */dev/sdb1*. (Note: The device descriptor might be different in your Linux PC.)

Next, mount CF card to */mnt* directory and change directory to the */mnt*.

```
root@dns3:~# mount -t ext3 /dev/sdb1 /mnt
root@dns3:~# cd /mnt
root@dns3:/mnt#
```

You can *ls* the file structure.

```
root@dns3:/mnt# ls
bin  etc   lib  opt   root  selinux  tmp  var
dev  home  mnt  proc  sbin  sys      usr
root@dns3:/mnt#
```

Next, tar the file system into a file. (The file name in this example is 320rootfs_backup.tar.gz)

```
root@dns3:/mnt# tar cvfz 320rootfs_backup.tar.gz
root@dns3:/mnt#
```

You have backup the CF root file systems as file name "*320rootfs_backup.tar.gz*"!

6.2 Restore the root file system in CF card

You need a Linux PC first. Plug a CF card into a USB card reader and plug the card reader into the USB port of your Linux PC. The operating system of the Linux PC in this example is Ubuntu 11.04 and the CF card storage is 4GB. (Note: 1GB is minimal requirement for the Embedian official root file system.)
Use the
 # fdisk -l
command to list your disk information and find the device descriptor of you CF USB reader.

```
root@dns3:~# fdisk -l

Disk /dev/sda: 1000.2 GB, 1000203804160 bytes
255 heads, 63 sectors/track, 121601 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x000d8811

   Device Boot      Start         End      Blocks   Id  System
/dev/sda1   *           1       24316   195311616   83  Linux
/dev/sda2           24316       24565     1999872   82  Linux swap /
Solaris
/dev/sda3           24565      121602   779448320   83  Linux

Disk /dev/sdb: 4009 MB, 4009549824 bytes
124 heads, 62 sectors/track, 1018 cylinders
Units = cylinders of 7688 * 512 = 3936256 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0xf578ad7b

   Device Boot      Start         End      Blocks   Id  System
/dev/sdb1               1        1018     3913161   83  Linux
root@dns3:~#
```

We can see the device descriptor of the USB CF card reader is in disk */dev/sdb* and there is one partition */dev/sdb1*. (Note: The device descriptor might be different in your Linux PC.)
If there is no partition in your CF card, you have to use *fdisk* to partition it first, here we partitioned the CF card as one partition. (New CF card should have one partition already by default.)

```
root@dns3:/# fdisk /dev/sdb

WARNING: DOS-compatible mode is deprecated. It's strongly recommended to
         switch off the mode (command `c`) and change display units to
         sectors (command `u`).

Command (m for help): d
Selected partition 1

Command (m for help): n
Command action
   e   extended
   p   primary partition (1-4)
p
Partition number (1-4): 1
First cylinder (1-1018, default 1):
Using default value 1
Last cylinder, +cylinders or +size{K,M,G} (1-1018, default 1018):
Using default value 1018

Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.
Syncing disks.
root@dns3:/#
```

Next, we need to format the CF card as ext3 file system by using
 # mkfs -t ext3 /dev/sdb1
command. (In Ubuntu or FC, you can also use *# mkfs.ext3 /dev/sdb1* command.)

```
root@dns3:/# mkfs -t ext3 /dev/sdb1
mke2fs 1.41.14 (22-Dec-2010)
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
Stride=0 blocks, Stripe width=0 blocks
```

```
244800 inodes, 978290 blocks
48914 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=1002438656
30 block groups
32768 blocks per group, 32768 fragments per group
8160 inodes per group
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912, 819200, 884736

Writing inode tables: done
Creating journal (16384 blocks): done
Writing superblocks and filesystem accounting information: done

This filesystem will be automatically checked every 30 mounts or
180 days, whichever comes first.  Use tune2fs -c or -i to override.
root@dns3:/#
```

And next, mount CF card to */mnt* directory and change directory to the */mnt*.

```
root@dns3:/# mount -t ext3 /dev/sdb1 /mnt
root@dns3:/# cd /mnt
root@dns3:/mnt#
```

Next, *cp* the rootfs file into /mnt directory and extracting the root file system file into this directory.

```
root@dns3:/mnt# ls
320rootfs_20090918.tar.gz  lost+found
root@dns3:/mnt#
```

Extract the root filesystem into */mnt* directory.

```
root@dns3:/mnt# tar xvfz 320rootfs_20090918.tar.gz
```

You can *ls* the file structure now.

```
root@dns3:/mnt# ls
320rootfs_20090918.tar.gz dev home lost+found nand proc sbin  sys  usr
bin                       etc lib  mnt        opt root selinux tmp var
root@dns3:/mnt#
```

Last, remove the tarball and exit the */mnt* directory and *umount* the device.

```
root@dns3:/mnt# rm -f 320rootfs_20090918.tar.gz
```

```
root@dns3:/mnt# cd ../
root@dns3:/# umount /mnt
root@dns3:/#
```

Take the CF card off from the card reader and put the CF card back to MXM-8310 evaluation kit and boot. You are done!

Chapter 7

General PCB Design Recommendations

A general description of the Printed Circuit Board (PCB) for MXM computer on module carrier boards is provided in this section.

Section include：

- Nominal Board Stack-Up
- Differential Impedance Targets for Microstrip Routing
- Alternative Stack Ups

Chapter 7 General PCB Design Recommendations

This section gives general description of the design recommendation of the Printed Circuit Board (PCB) for MXM computer on module carrier boards. From a cost- effectiveness point of view, a four-layer board is the target platform for the carrier board design. For better quality, a six-layer or eight-layer board is preferred.

7.1 Nominal Board Stack-Up

The trace impedance typically noted (55 Ω ± 10%) is the “nominal” trace impedance for a 5-mil wide external trace and a 4-mil wide internal trace. However, some stackups may lead to narrower or wider traces on internal or external layers in order to meet the 55-Ω impedance target, that is, the impedance of the trace when not subjected to the fields created by changing current in neighboring traces. Note the trace impedance target assumes that the trace is not subjected to the EMI fields created by changing current in neighboring traces.

It is important to consider the minimum and maximum impedance of a trace based on the switching of neighboring traces when calculating flight times. Using wider spaces between the traces can minimize this trace-to-trace coupling. In addition, these wider spaces reduce settling time.

Coupling between two traces is a function of the coupled length, the distance separating the traces, the signal edge rate, and the degree of mutual capacitance and inductance. In order to minimize the effects of trace-to-trace coupling, the routing guidelines documented in this Section should be followed. Also, all high speed, impedance controlled signals should have continuous GND referenced planes and cannot be routed over or under power/GND plane splits.

7.1.1. Four Layer Board Stackup

Figure 7.1 illustrates an example of a four-layer stack-up with 2 signal layers and 2 power planes. The two power planes are the power layer and the ground layer. The layer sequence of component-ground-power-solder is the most common stack-up arrangement from top to bottom.

Figure 7.1 Four-Layer Stack-Up

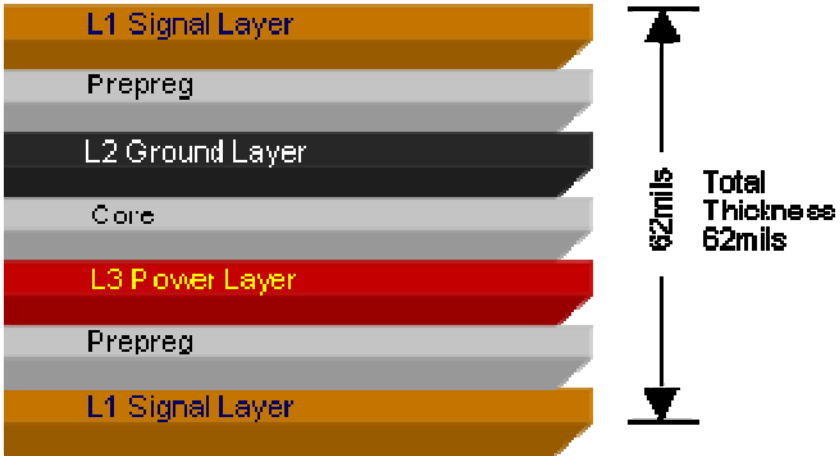


Table 7.1 Recommended Four-Layer Stack-Up Dimensions

Table 7.1 Recommended Four-Layer Stack-Up Dimensions								
Dielectric Thickness (mil)	*Layer*	*Layer*	*Signal-End Signals*		*Differential Signals*		*USB Differential Signals*	
	No	*Type*	*Width (mil)*	*Impe-dance (ohm)*	*Width (mil)*	*Impe-dance (ohm)*	*Width (mil)*	*Impe-Dance (ohm)*
0.7	*L1*	*Signals*	*6/6*	*55+/-10%*	*6/7/6*	*100+/-10%*	*6/5/6*	*90+/-10%*
5		*Prepreg*						
1.4	*L2*	*Ground*						
47		*Core*						
1.4	*L3*	*Power*						
5		*Prepreg*						
0.7	*L4*	*Signals*	*6/6*	*55+/-10%*	*6/7/6*	*100+/-10%*	*6/5/6*	*90+/-10%*

Note:
Target PCB Thickness totals 62mil+/-10%

7.1.2 Six Layer Board Stackup

Figure 7.2 illustrates an example of a six-layer stack-up with 4 signal layers and 2 power planes.

The two power planes are the power layer and the ground layer. The layer sequence of component-ground-IN1-IN2-power-solder is the most common stack-up arrangement from top to bottom.

Figure 7.2 Six-Layer Stack-Up

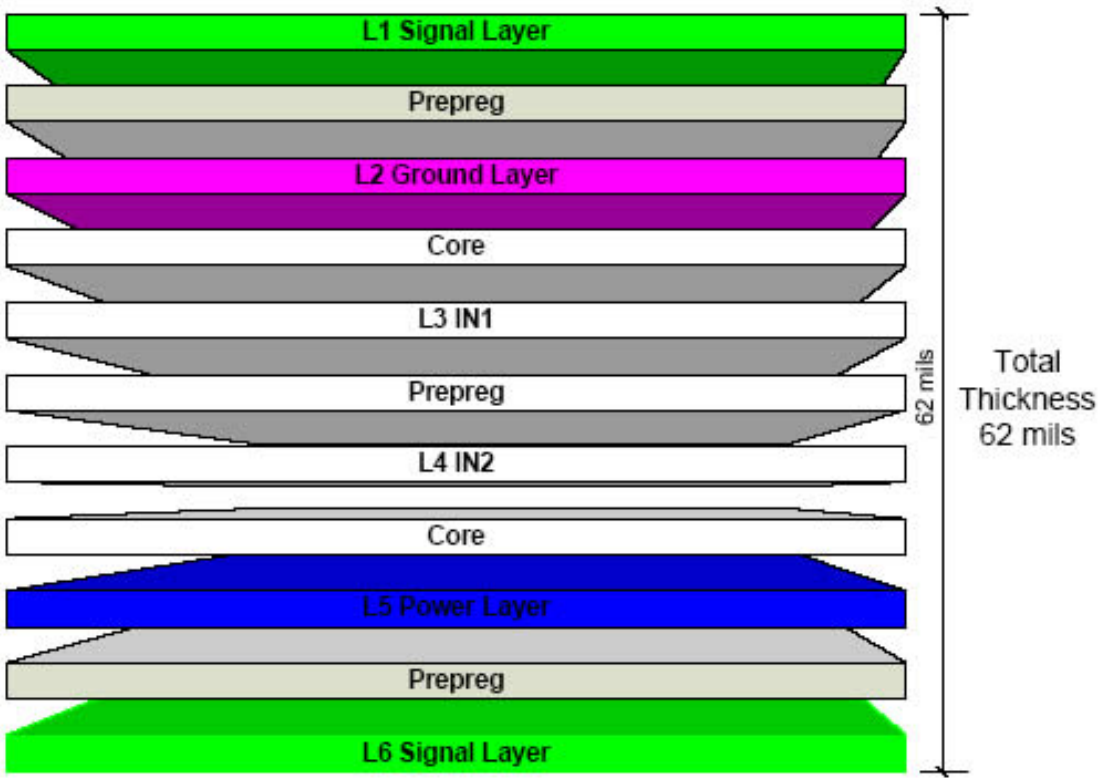


Table 7.2 Recommended Six-Layer Stack-Up Dimensions

Table 7.2 Recommended Six-Layer Stack-Up Dimensions								
Dielectric Thickness (mil)	*Layer*	*Layer*	*Signal-End Signals*		*Differential Signals*		*USB Differential Signals*	
	No	*Type*	*Width (mil)*	*Impe-dance (ohm)*	*Width (mil)*	*Impe-dance (ohm)*	*Width (mil)*	*Impe-Dance (ohm)*
1.7	*L1*	*Signals*	*5/5*	*55+/-10%*	*5/6/5*	*100+/-10%*	*5/4/5*	*90+/-10%*
4		*Prepreg*						
1.4	*L2*	*Ground*						
5		*Core*						
1.4	*L3*	*IN1*	*5/5*	*55+/-10%*	*4/8/4*	*100+/-10%*	*4/5/4*	*90+/-10%*
35		*Prepreg*						
1.4	*L4*	*IN2*						
5		*Core*	*5/5*	*55+/-10%*	*4/8/4*	*100+/-10%*	*4/5/4*	*90+/-10%*
1.4	*L5*	*Power*						
4		*Prepreg*						
1.7	*L6*	*Signals*	*5/5*	*55+/-10%*	*5/6/5*	*100+/-10%*	*5/4/5*	*90+/-10%*

Note:
Target PCB Thickness totals 62mil+/-10%

7.2 Differential Impedance Targets for Microstrip Routing

Table 7.3 shows the target impedance of the differential signals. The carrier board should follow the required impedance in this table.

Table 7.3 Differential Signals Impedance Requirement

Table 7.3 Differential Signals Impedance Requirement	
Signal Type	***Impedance***
USB	*90ohm +/- 20%*
LAN	*100ohm +/- 20%*

7.3 Alternative Stack Ups

When customers choose to use different stack-ups (number of layers, thickness, trace width, etc.), the following key elements should be observed:

1. Final post lamination, post etching, and post plating dimensions should be used for electrical model extractions.
2. All high-speed signals should reference solid ground planes through the length of their routing and should not cross plane splits. To guarantee this, both planes surrounding strip-lines should be GND.
3. Recommends that high-speed signal routing be done on internal, strip-line layers. High-speed routing on external layers should be minimized in order to avoid EMI. Routing on external layers also introduces different delays compared to internal layers. This makes it extremely difficult to do length matching if routing is done on both internal and external layers.

Chapter 8

Carrier Board Design Guidelines

A detail description of design guidelines for the MXM computer on module carrier boards is provided in this section.

Section include：

- General Circuit Design Guide
- Universal Serial Bus (USB)
- AC-Link Interface
- D/A
- Ethernet

Chapter 8 Carrier Board Design Guidelines

This section gives detail description of the design recommendation of the MXM computer on module carrier boards. It points out the rules that need to be carefully followed in circuit design and layout.

8.1 General Circuit Design Guide

This section states the circuit design guide. Please follow carefully or the system might not able to boot.

8.1.1. System-Wise

The following tables describe the system-wise circuit design guide that need to be carefully followed. System might not boot if didn't followed correctly.

Table 8.1. : System-Wise Circuit Design Guide

Signal Name	Function	Description
nWAIT	nWAIT Requests	Pull up 100K resistor
WAKEUP	WAKEUP Requests	Internal 10K pull up resistor (*)
nRESET_IN	Reset PXA320	Pull up 100K resistor
nRESET_OUT	Reset External Device	Pull up 100K resistor
nGCS[0..5]	Chip Select	Pull up 10K resistor

(*) In MXM-7110/MXM-7114, WAKEUP Requests is Internal 4.7K pull down resistor.

Table 8.2. : JTAG

Signal Name	Function	Description
TMS	TAP Controller Mode Select	Pull up 10K resistor
TDO	TAP Controller Data Output	- -
TDI	TAP Controller Data Input	Pull up 10K resistor
TCK	TAP Controller Clock	Pull up 10K resistor
nTRST	TAP Controller Reset	Pull up 10K resistor

Table 8.3. : IIC

Signal Name	Function	Description
IICSCL	*IIC-bus clock*	*Internal 1.2K pull up*
IICSDA	*IICI-bus data*	*Internal 1.2K pull up*

Table 8.4. : SD

Signal Name	Function	Description
SD_nCD	*SD Insert Detect*	*Pull up 49.9K resistor*
SD_WP	*SD Write Protect*	*Pull up 49.9K resistor*
SDCLK	*SD Clock*	*- -*
SDCMD	*SD receive response/ transmit command*	*Pull up 49.9K resistor*
SDDAT0	*BootRom Select*	*Pull up 49.9K resistor*
SDDAT1	*SD receive/transmit data*	*Pull up 49.9K resistor*
SDDAT2	*SD receive/transmit data*	*Pull up 49.9K resistor*
SDDAT3	*SD receive/transmit data*	*Pull up 49.9K resistor*

Table 8.5. : One Wire

Signal Name	Function	Description
One Wire	*1-Wire bidirectional data bus*	*Internal 5K pull up*

Table 8.6. : Power

Signal Name	Function	Description
EXT5V	*DC5V Input*	*DC5V +-5%*
BBAT	*RTC Battery Power(DC 3V)*	*DC3V*
GND	*Ground Power*	*All Ground should be tied together, except for AGND*
AVDD18	*1.8V For Transformer*	*DC1.8V output to transformer*
AGND	*Analog Ground*	*Analog ground to transformer*

Table 8.7. : USB

Signal Name	Function	Description

USBH- and USBH+	*USB Host Data*	*Differential Pair*
USBD- and USBD+	*USB Device Data*	*Differential Pair*

Table 8.8. : USB

Signal Name	***Function***	***Description***
TX- and TX+	*Ethernet Transmits data*	*Differential Pair*
RX- and RX+	*Ethernet Receives data*	*Differential Pair*

8.2 Universal Serial Bus (USB)

MXM computer modules provide two USB 1.1 ports.

8.2.1. Universal Serial Bus (USB)

The Universal Serial Bus (USB) provides a bi-directional, isochronous, hot-attachable Plug and Play serial interface for adding external peripheral devices such as game controllers, communication devices and input devices on a single bus.

USB stands for Universal Serial Bus, an industry-standard specification for attaching peripherals to a computer. It delivers high performance, the ability to plug in and unplug devices while the computer is running, great expandability, and a wide variety of solutions.

8.2.2. Signal Description

Table 8.9 shows MXM module USB signals, including pin number, signals, I/O and descriptions.

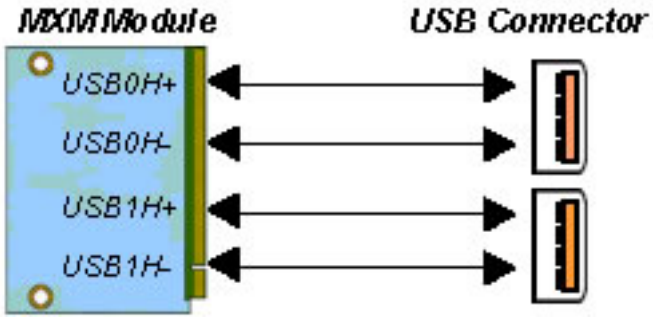
Table 8.9 Differential Signals Impedance Requirement

Table 8.9 USB Signal Description			
USB Host 0			
185	*USB0H-*	*USB0 Host Data -*	*I/O*
187	*USB0H+*	*USB0 Host Data +*	*I/O*
USB Host 1			
193	*USB1H-*	*USB1 Host Data -*	*I/O*
195	*USB1H+*	*USB1 Host Data +*	*I/O*

8.2.3. Design Guidelines

Figure 8.1 shows USB connections for MXM module USB signals.

Figure 8.1 USB Connection



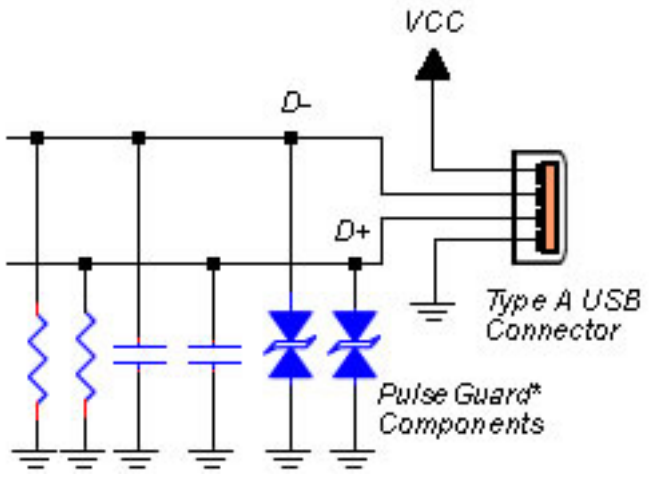
8.2.3.1. Low ESR Capacitor

You can hot plug USB devices. In fact, this is one of the virtues of USB relative to most other legacy interfaces. The design of the USB power-decoupling network must absorb the momentary current surge from hot plugging an unpowered device.
Reducing these values is not recommended. These capacitors should be low ESR, low inductance.

8.2.3.2. ESD or EMI suppression components

The following guidelines apply to the selection and placement of common mode chokes and ESD protection devices. Some USB designs will need additional ESD or EMI suppression components on the USB data lines. These are most effective when they are placed near the external USB connector and grounded to a low-impedance ground plane. MXM modules equips with two USB ports. Some people implement three or four ports. If the application needs more than two USB ports, a low cost USB hub IC can be integrated onto the carrier board and connected to the USB0 or USB1 ports on the MXM module. This provides a larger number of USB ports.
A design may include a RC filter to provide a stuffing option in the event the filter is needed to pass EMI testing. Figure 8.2 shows the schematic of a typical RC filter and ESD suppression components. The RC filter should be placed as close as possible to the USB connector signal pins.

Figure 8.2 RC Filter



Note:

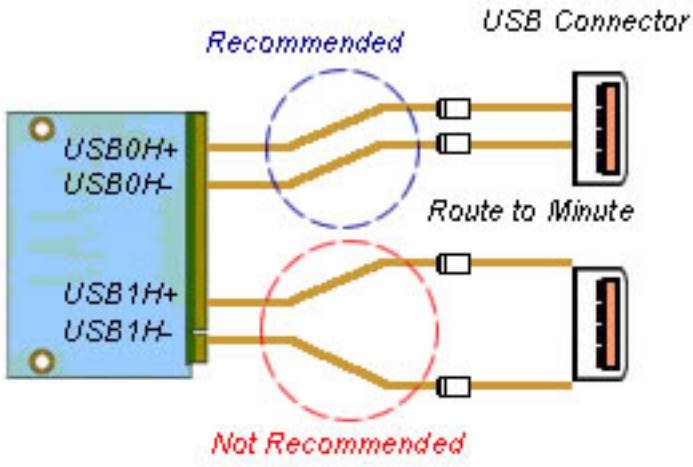
ESD protection and RC filter are only needed if the design does not pass EMI or ESD testing. Basically, it is recommended to add them in the USB 1.1 interface. Footprints for ESD suppression components should be included in the event that a problem occurs (General routing and placement guidelines should be followed).

8.2.3. Layout Guidelines

8.2.3.1. Differential Pairs

The USB data pairs (ex. USB0H+ and USB0H-) should be routed on the carrier board as differential pairs, with a differential impedance of 90 Ω. PCB layout software usually allows determining the correct trace width and spacing to achieve this impedance, after the PCB stack-up configuration is known.
As per usual differential pair routing practices, the two traces of each USB pair should be matched in length and kept at uniform spacing. Sharp corners should be avoided. At the MXM module and connector ends of the routes, loop areas should be minimized. USB data pairs should be routed as far from other signals as possible.

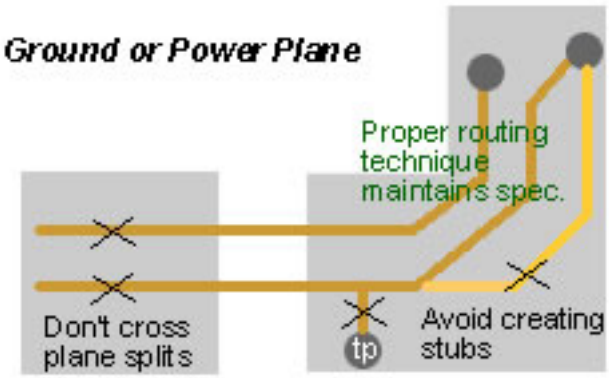
Figure 8.3 USB Layout Guidelines



8.2.3.2. Cross a plane split

The mistake shown here is where the data lines cross a plane split. This causes unpredictable return path currents and would likely cause a signal quality failure as well as creating EMI problems.

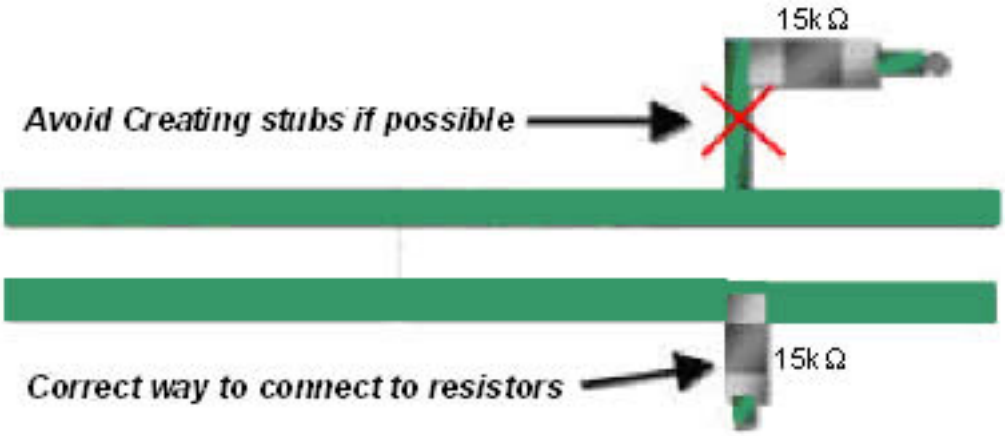
Figure 8.4 Violations of Proper Routing Techniques



8.2.3.3. Stubs

A very common routing mistake is shown in Figure 8.5. Here the designer could have avoided creating unnecessary stubs by proper placement of the pull down resistors over the path of the data traces. Once again, if a stub is unavoidable in the design, no stub should be greater than 200 mils. Here is another example where a stub is created that could have been avoided. Stubs typically cause degradation of signal quality and can also affect EMI.

Figure 8.5 Creating unnecessary stubs



8.3 AC-Link Interface

MXM module provides an AC Link interface which is compliant to AC.97 Rev. 2.3 Specification. Please establish the AC.97 CODEC on the carrier board for your application.

8.3.1. Signal Description

Table 8.10 shows MXM module AC-Link signals, including pin number, signals, I/O and descriptions.

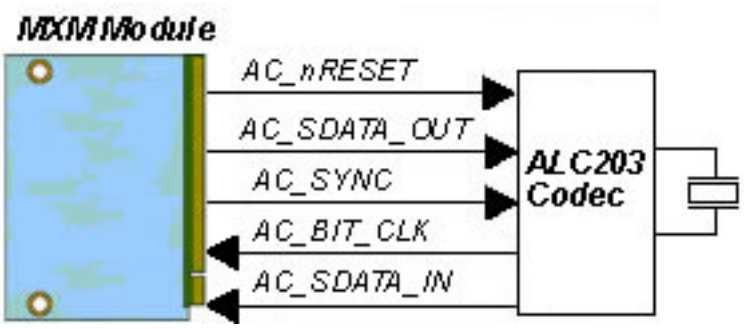
Table 8.10 Audio Signal Description

Table 8.10 Audio Signal Description			
AC97			
16	*AC_SYNC*	*48kHz fixed rate sample sync*	*O*
18	*AC_BIT_CLK*	*12.288MHz serial data clock*	*I/O*
20	*AC_nRESET*	*AC'97 Master H/W Reset*	*O*
22	*AC_SDATA_IN*	*AC'97 input stream*	*I*
24	*AC_SDATA_OUT*	*AC'97 output stream*	*O*

8.3.2. Design Guidelines

Figure 8.6 shows the connections for MXM module AC link signals. AC_BITC_LK is a 12.288 MHz clock driven by a crystal to the MXM module digital controller and to the codec.

Figure 8.6 AC-Link Connections



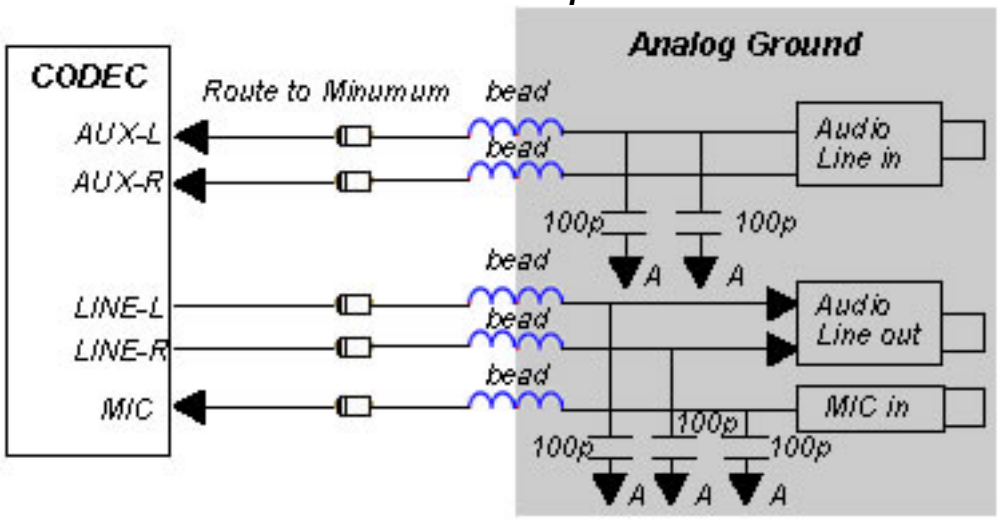
8.3.2.1. Codec Reference and Anti-Aliasing Recommendations

Place all ADC/DAC anti-aliasing filters and reference capacitors within 0.5 inches of their respective codec pins. All filter capacitors' ground connections should attach to ground trace from the codec to the capacitors without allowing vias to the digital ground plane. The audio codec should be placed in the quietest part (away from significant current paths and ground bounce) of the carrier board.

8.3.2.2. Grounding Techniques

Take care when grounding back panel audio jacks, especially the line in and microphone jacks. Avoid grounding the audio jacks to the ground plane directly under the connectors. Doing so raises the potential for audio noise to be induced on the inputs due to the difference in ground potential between the audio jacks and the codec's ground point. Figure 8.7 provides an AC'97 example.

Figure 8.7 AC-Link Audio Ground Technique



8.3.2.3. AC link Stereo Microphone & Line In / Auxiliary In consideration

Back panel microphone input signal should be independent routed, and the ground return paths should be isolated from the carrier board ground plane. Use a capacitor to filter noise from the microphone bias net feeding all microphone jacks. Route microphone traces as far away as possible from non-microphone trace and digital traces. Audio designs that support up to 2 V RMS line input signals are recommended, but not required. To support audio inputs up to 2 V RMS, designs should implement a voltage divider network to effectively reduce the input level 6 dB prior to reaching the codec.

8.3.3. Layout Guidelines

Proper component placement and routing are crucial to ensure maximum performance from the AC'97 device. This document discusses methods to provide a proper design execution, including properly isolating the digital and analog circuitry, the effects of ground and supply plane geometry, decoupling/bypassing/filtering capacitor placement priorities, AC-LINK signals, analog power supplies, and analog ground planes.

8.3.3.1. Ground and Supply Planes

Figure 8.8 shows a ground plane layout for an onboard AC'97 CODEC. This layout separates the analog and digital ground planes with a 60 to 100 mil gap. The moat helps to isolate noisy digital circuitry from the quieter analog audio circuitry.

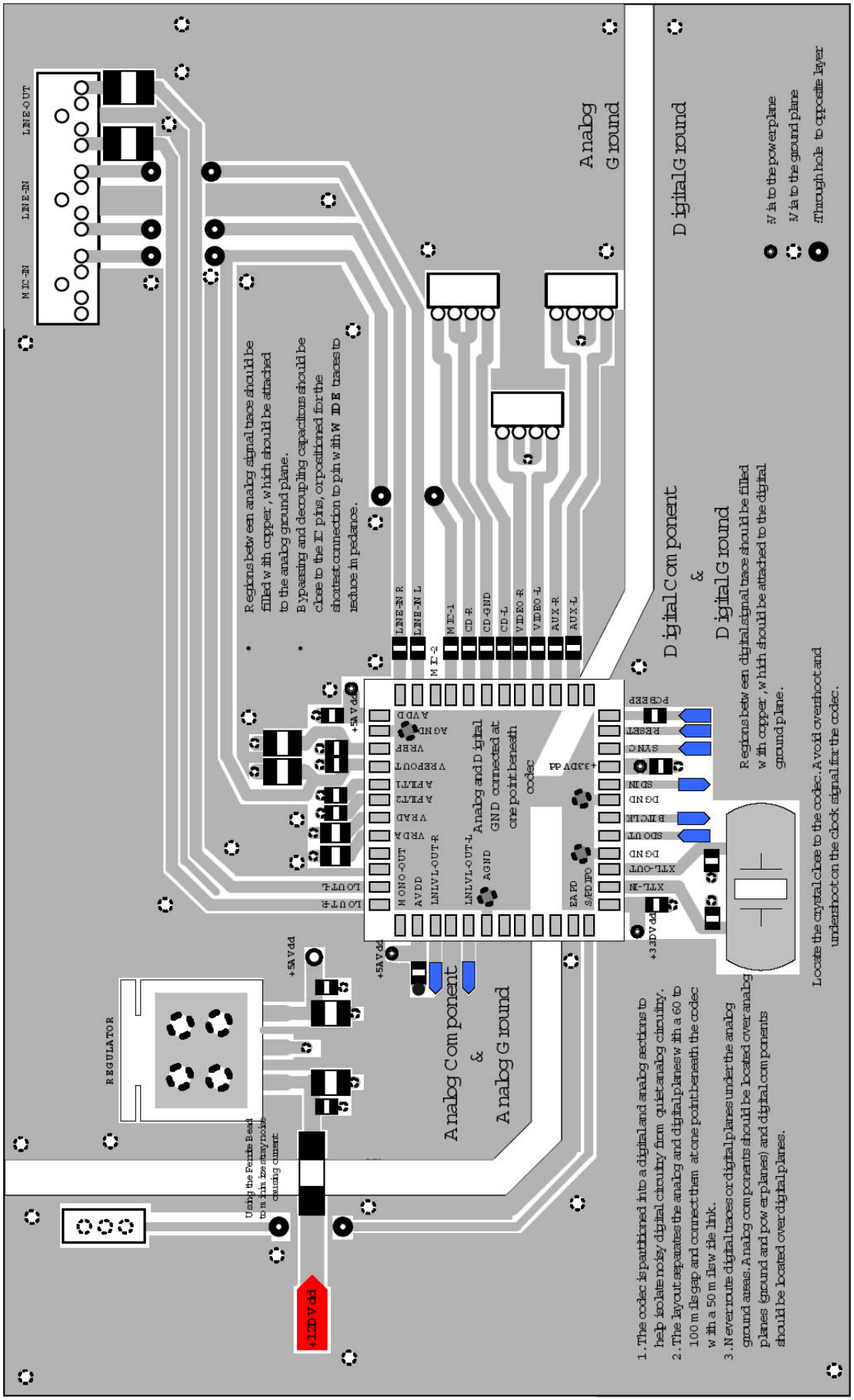
The digital and analog ground planes are tied together by a wide link, about 50mils, at one point, and only one point, beneath the CODEC itself. This acts as the "drawbridge" that goes across the moat. Do not allow any digital or analog signal traces to pass through the drawbridge, or digital noise may be induced into the analog signals, resulting in deteriorating audio performance. In addition, NO SIGNALS WHATEVER is permitted to cross the moat. To do so creates a "slot antenna" radiator which will beat the PCB layout with crosstalk, creating large amounts of EMI, resulting in a poor system.

For a layout that helps to reduce noise, separate analog and digital ground planes should be provided, with the digital components located over the digital ground plane, and the analog components, including the analog power regulators, located over the analog ground plane. In addition to ground planes scheme, digital and analog power supply planes should be partitioned directly over their respective ground planes. Be careful in using the split ground plane and match non-overlapping +5Avdd supply planes. The power and ground planes should be separated by approximately 40mils for a four layer PCB design. Use power and ground planes to form a natural, high capacitive, bypass capacitor to reduce overall PCB noise.

The general rules are:

1. The codec is partitioned into a digital and analog section to help isolate noisy digital circuitry from quiet analog circuitry.
2. The layout separates the analog and digital planes with a 60 to 100 mils gap and connects them at one point beneath the codec with a 50 mils wide link.
3. Never route digital traces or digital planes under the analog ground areas. Analog components should be located over analog planes (ground and power planes) and digital components should be located over digital planes.

Figure 8.8 AC-Link Audio Layout Guidelines



8.3.3.2. Decoupling and Bypassing Capacitors

Bypass capacitors on the PCB are used to short digital noise to ground. Commonly, the CODEC generates noise when its internal digital circuitry turns current on and off. These current changes arise in the power and ground pins for the related section of the CODEC. The goal is to force AC currents to flow in the shortest possible loop from the supply pin through the bypass cap and back into the CODEC through the nearby ground pin. A bypassing circuit is supposed to be a low lead inductance between the CODEC and the bypass capacitors when in the operating frequency of the CODEC. The longer the trace – the greater the inductance. To avoid long-trace inductance effects, use the shortest possible traces for bypass capacitors, with wide traces to reduce impedance. For best performance, use supply bypass leads of less than one-half inch.

In Figure 8.9, pins labeled “A” priority require bypass caps placed around the CODEC, which should be located as close as possible to the supply pins. The capacitors must have low inductance and low equivalent series resistance (ESR). Tantalum 10μF surface mount devices are good if they are used in conjunction with 0.1μF ceramics. The filter capacitors with “B” priority (Pins 27&28) and analog ground stabilize the reference voltage for internal Ops should be placed close to the CODEC.

A good reference voltage is relative to good analog performance. The decoupling capacitors (“C” priority) should be close to the specified CODEC pins (pins 12 to 24), or positioned for the shortest connections to those pins, with wide traces to reduce impedance. Table 8.11 also points out the distribution of CODEC capacitor locations and placement priorities.

Figure 8.9 CODEC Recommended Capacitor Placement

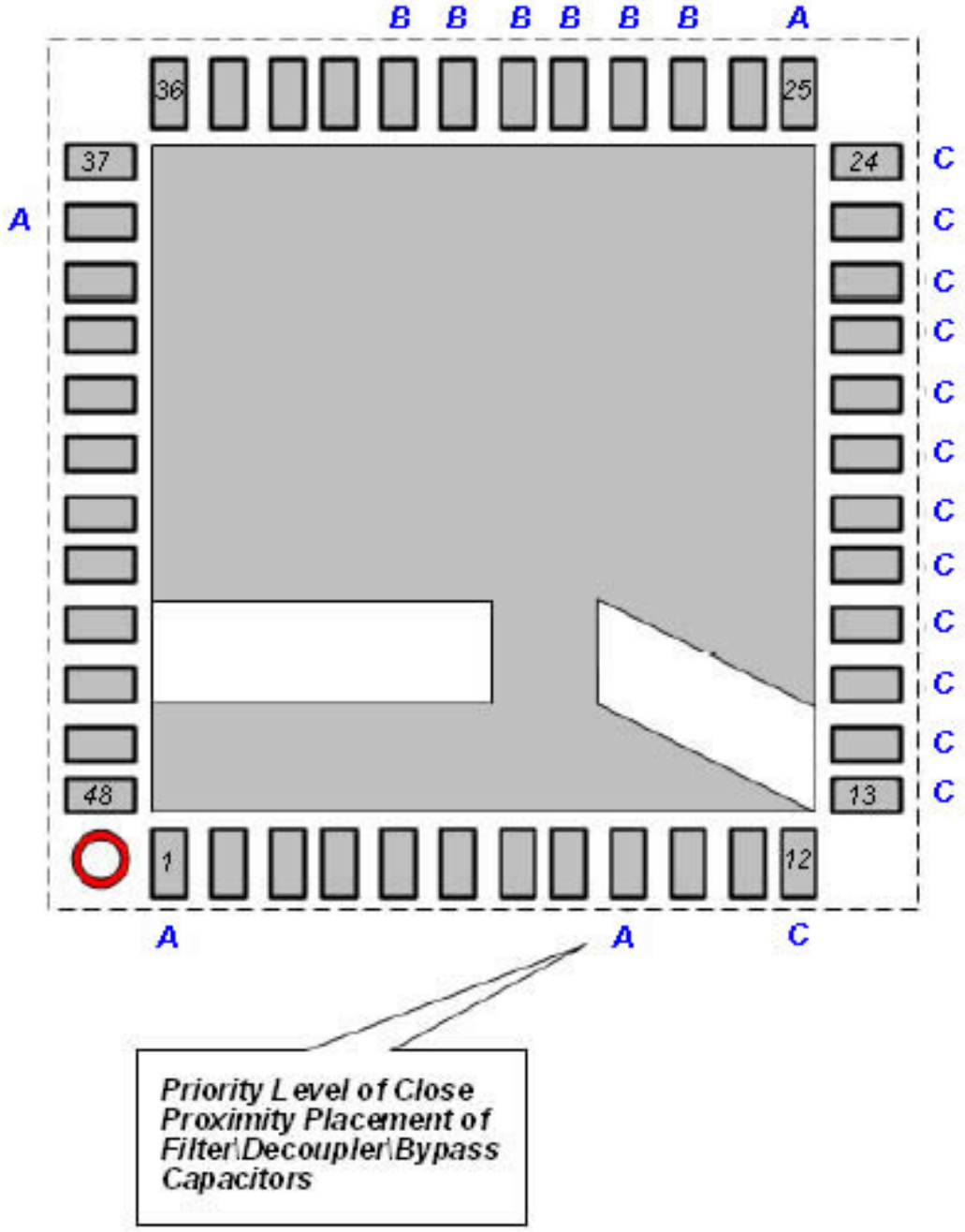


Table 8.11 Series CODEC Capacitor Placement Priorities

Table 8.11 Realtek Series CODEC Capacitor Placement Prioroties		
Signal Description	***Package Pins***	***Priority of Close Proximity to CODEC Pin Placement of Filter and Decoupling Capacitors***
Digital Supply Voltage , +5DVdd	*1, 9*	*A*
Analog Supply Voltage, +5AVdd	*25, 38*	*A*
Voltage Reference Filter(VREF)	*27*	*B*
Voltage Reference (VREF_OUT)	*28*	*B*
CODEC Filters	*29, 30, 31, 32*	*B*
Analog Signal Inputs(Decouple)	*12~24*	*C*

8.4 TTL/LVDS LCD

MXM-8310 equips 18-bit TTL-level LCD signals. Additional transceiver will be needed if users would like to use LVDS panel.

8.4.1. Signal Description

Table 8.12 shows MXM module TTL level LCD signals, including pin number, signals, I/O and descriptions.

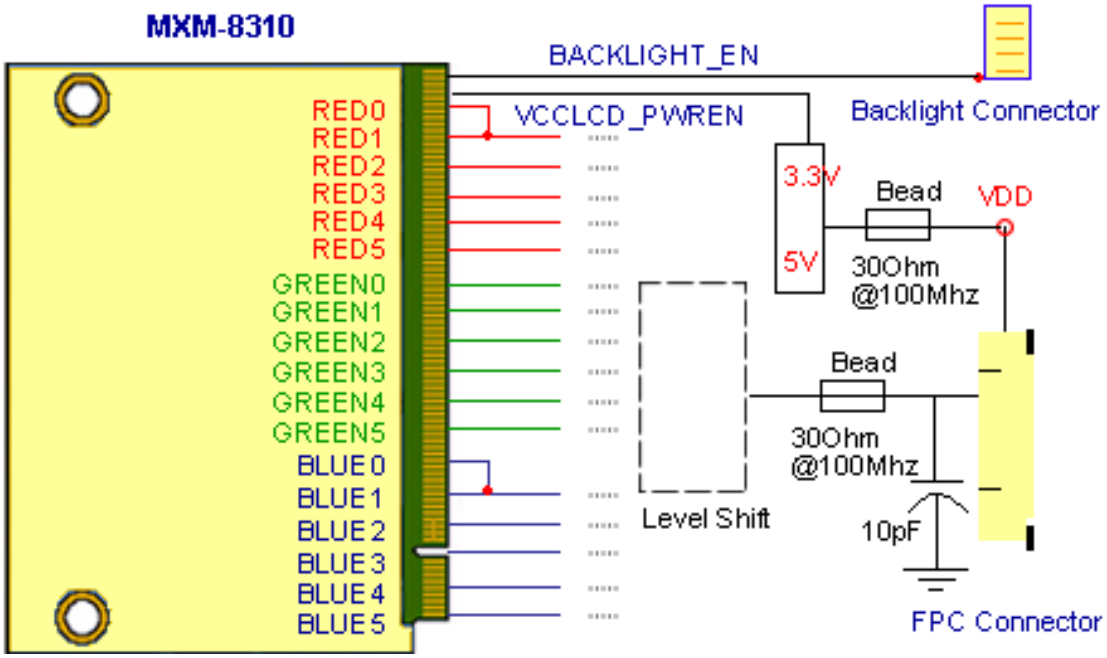
Table 8.12 CPU LCD Signal Description

Table 8.12 CPU LCD Signal Description			
CPU LCD			
48	*LDD11*	*LCD data bus RED0 (LSB)*	*O*
50	*LDD12*	*LCD data bus RED1*	*O*
52	*LDD13*	*LCD data bus RED2*	*O*
54	*LDD14*	*LCD data bus RED3*	*O*
56	*LDD15*	*LCD data bus RED4 (MSB)*	*O*
58	*LDD5*	*LCD data bus GREEN0 (LSB)*	*O*
60	*LDD6*	*LCD data bus GREEN1*	*O*
62	*LDD7*	*LCD data bus GREEN2*	*O*
64	*LDD8*	*LCD data bus GREEN3*	*O*
66	*LDD9*	*LCD data bus GREEN4*	*O*
68	*LDD10*	*LCD data bus GREEN5 (MSB)*	*O*
72	*LDD0*	*LCD data bus BLUE0 (LSB)*	*O*
74	*LDD1*	*LCD data bus BLUE1*	*O*
76	*LDD2*	*LCD data bus BLUE2*	*O*
78	*LDD3*	*LCD data bus BLUE3*	*O*
80	*LDD4*	*LCD data bus BLUE4 (MSB)*	*O*
82	*VCLK*	*LCD clock signal*	*O*
84	*HSYNC*	*Horizontal synchronous signal*	*O*
86	*VSYNC*	*Vertical synchronous signal*	*O*
88	*VDEN*	*Data enable signal*	*O*

8.4.2. Design Guidelines

Figure 8-10 shows the TTL LCD connection.

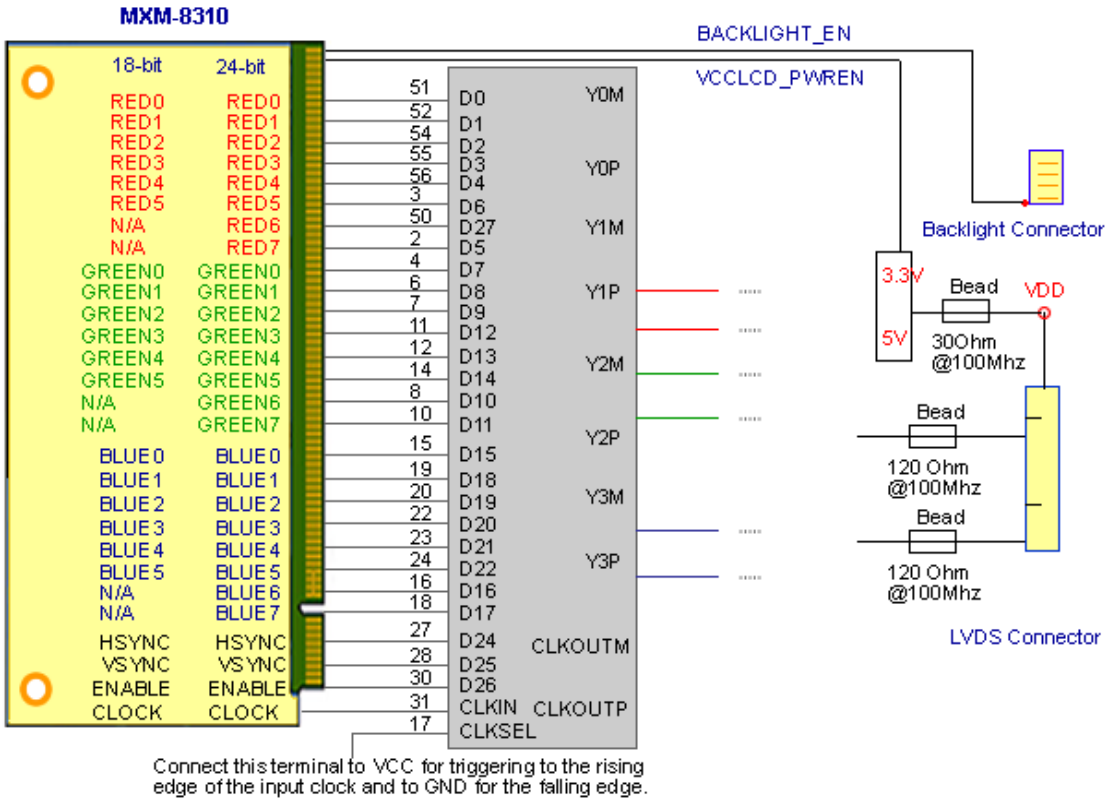
Figure 8.10 TTL LCD Connection



All MXM-8310 TTL LCD signal level is 3.3V. For 5V or 1.8V signal level TTL LCD, an additional level shift is required. If you need to support 3.3 and 5V level LCD, level shift and 0 Ohm resistor co-layout is suggested.

If user wants to connect a LVDS panel, a TTL to LVDS transmitter is required. Embedian recommend SN75LVDSS83 chip. Figure 8.11 shows the LVDS LCD connection.

Figure 8.11 LVDS LCD Connection



8.4.3. Layout Guidelines

Each LVDS channel is required to be length matched to within +/- 20 mils of each other.

Figure 8.12 LVDS LCD Layout Guidelines

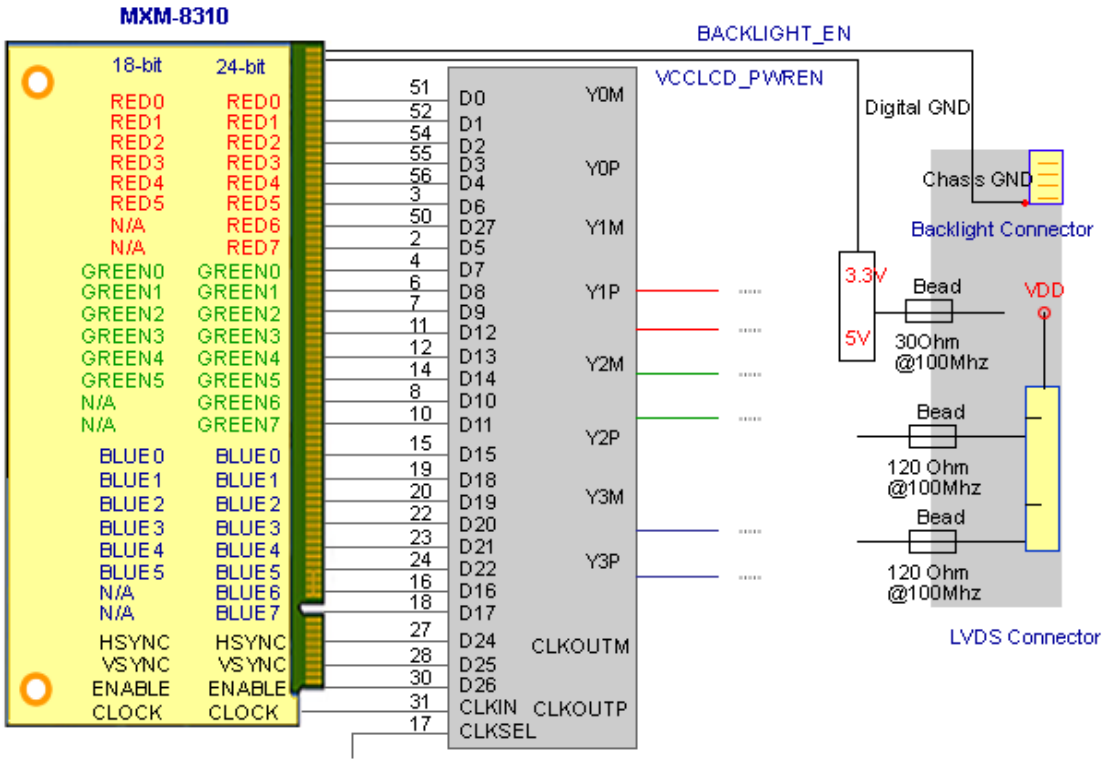
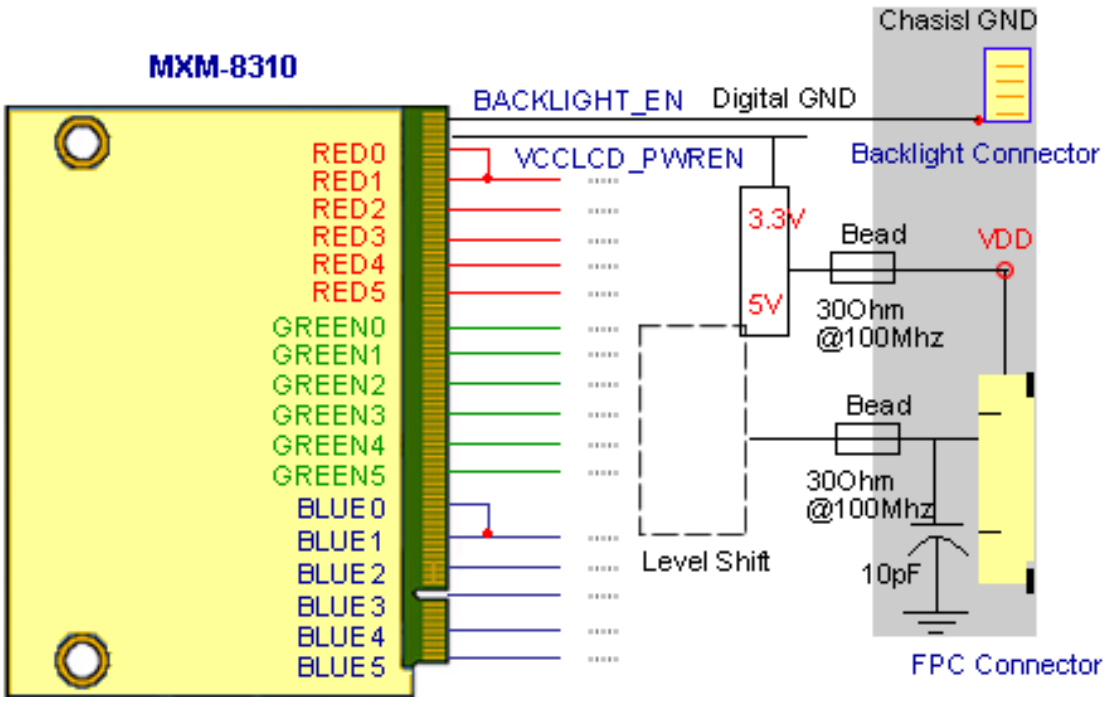


Figure 8.13 TTL LCD Layout Guidelines



8.5 DSUB15 VGA

Additional D/A is required and is to convert the LCD TTL signal to analog display signals if user want to connect to a DSUB15 VGA connector. This section is for MXM-8310 developers and would like to have a D-Sub 15 VGA interface so they can connect a legacy monitor directly. Please note that the MXM-8310 can only drive up to 800x600 resolutions.

8.5.1. Signal Description

Table 8.12 shows MXM module TTL level LCD signals, including pin number, signals, I/O and descriptions. Here we just attached again.

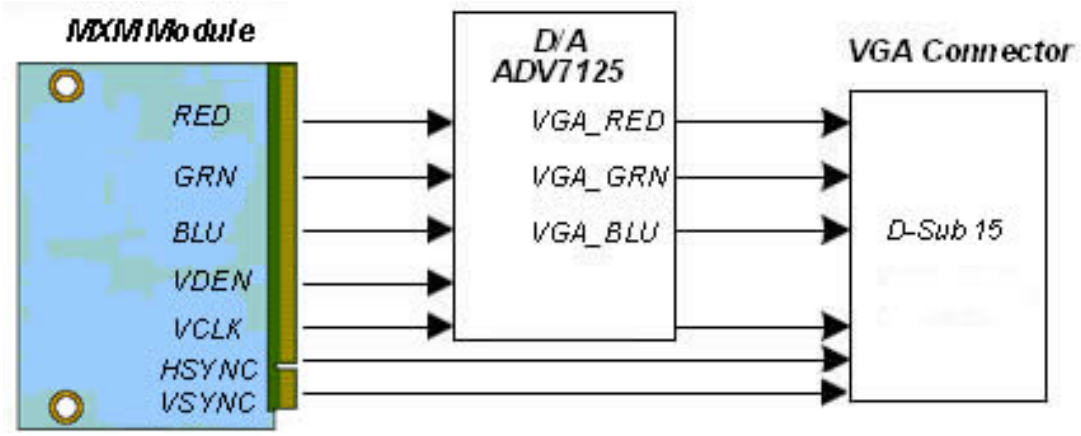
Table 8.12 CPU LCD Signal Description

Table 8.12 CPU LCD Signal Description			
CPU LCD			
48	*LDD11*	*LCD data bus RED0 (LSB)*	*O*
50	*LDD12*	*LCD data bus RED1*	*O*
52	*LDD13*	*LCD data bus RED2*	*O*
54	*LDD14*	*LCD data bus RED3*	*O*
56	*LDD15*	*LCD data bus RED4 (MSB)*	*O*
58	*LDD5*	*LCD data bus GREEN0 (LSB)*	*O*
60	*LDD6*	*LCD data bus GREEN1*	*O*
62	*LDD7*	*LCD data bus GREEN2*	*O*
64	*LDD8*	*LCD data bus GREEN3*	*O*
66	*LDD9*	*LCD data bus GREEN4*	*O*
68	*LDD10*	*LCD data bus GREEN5 (MSB)*	*O*
72	*LDD0*	*LCD data bus BLUE0 (LSB)*	*O*
74	*LDD1*	*LCD data bus BLUE1*	*O*
76	*LDD2*	*LCD data bus BLUE2*	*O*
78	*LDD3*	*LCD data bus BLUE3*	*O*
80	*LDD4*	*LCD data bus BLUE4 (MSB)*	*O*
82	*VCLK*	*LCD clock signal*	*O*
84	*HSYNC*	*Horizontal synchronous signal*	*O*
86	*VSYNC*	*Vertical synchronous signal*	*O*
88	*VDEN*	*Data enable signal*	*O*

8.5.2. Design Guidelines

Figure 8.14 shows the connections for MXM module LCD to D/A signals.

Figure 8.14 LCD to D/A Connection



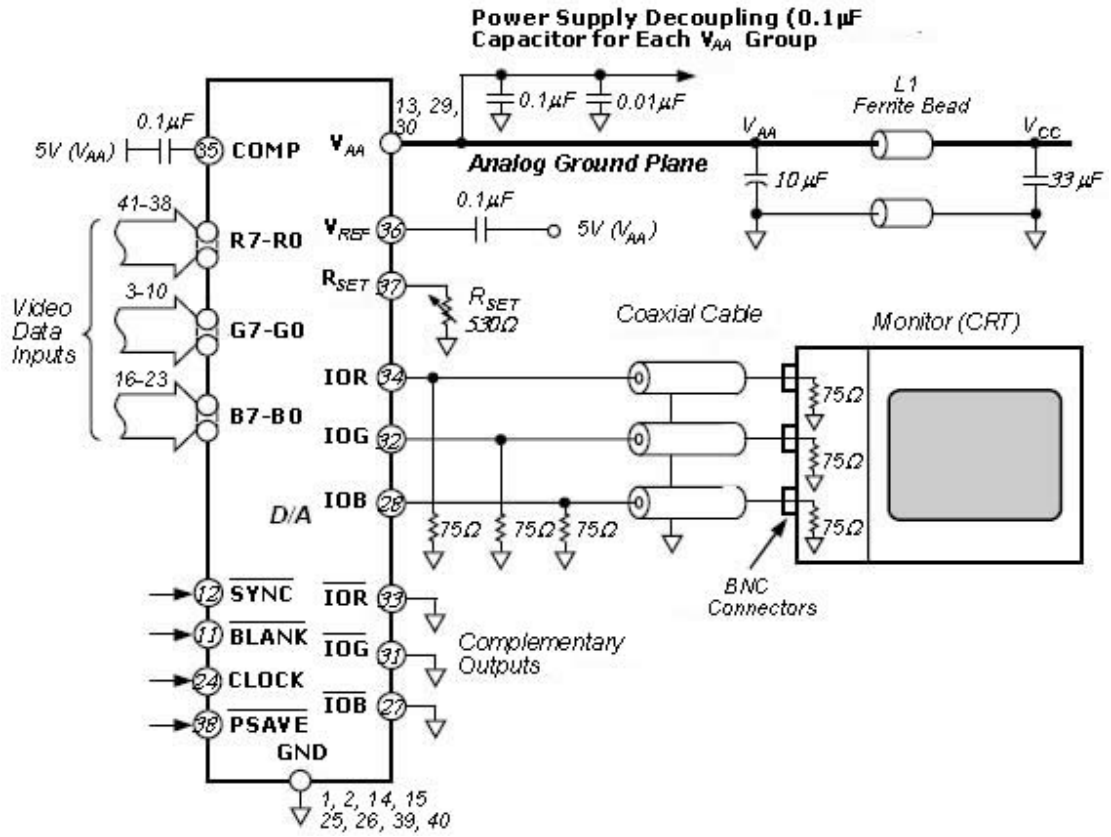
For signals connecting D/A to D-Sub15, users can refer to the next “VGA” section.

8.5.3. Layout Guidelines

To complement the excellent noise performance of the D/A, it is imperative that great care be given to the PC board layout. Figure 8.15 shows a recommended connection diagram for the ADV7125.

The layout should be optimized for lowest noise on the D/A power and ground lines. This can be achieved by shielding the digital inputs and providing good decoupling. The lead length between groups of VAA and GND pins should by minimized to minimize inductive ringing.

Figure 8.15 Typical Connection Diagram



8.5.3.1. Ground Planes

The D/A and associated analog circuitry should have a separate ground plane referred to as the analog ground plane. This ground plane should connect to the regular PCB ground plane at a single point through a ferrite bead, as illustrated in Figure 8.15. This bead should be located as close as possible (within three inches) to the D/A.

The analog ground plane should encompass all D/A ground pins, voltage reference circuitry, power supply bypass circuitry, the analog output traces, and any output amplifiers.

The regular PCB ground plane area should encompass all the digital signal traces, excluding the ground pins, leading up to the D/A.

8.5.3.2. Power Planes

The PC board layout should have two distinct power planes, one for analog circuitry and one for digital circuitry. The analog power plane should encompass the D/A (VAA) and all associated analog circuitry. This power plane should be connected to the regular PCB power plane (VCC) at a single point through a ferrite bead. This bead should be located within three inches of the D/A.

The PCB power plane should provide power to all digital logic on the PC board, and the analog power plane should provide power to all D/A power pins, voltage reference circuitry, and any output amplifiers.

The PCB power and ground planes should not overlay portions of the analog power plane. Keeping the PCB power and ground planes from overlaying the analog power plane will contribute to a reduction in plane-to-plane noise coupling.

8.5.3.3. Supply Decoupling

Noise on the analog power plane can be further reduced by the use of multiple decoupling capacitors (see Figure 8.15).

Optimum performance is achieved by the use of 0.1 μF ceramic capacitors. Each of the two groups of VAA should be individually decoupled to ground. This should be done by placing the capacitors as close as possible to the device with the capacitor leads as short as possible, thus minimizing lead inductance.

It is important to note that while the D/A contains circuitry to reject power supply noise, this rejection decreases with frequency. If a high frequency switching power supply is used, the designer should pay close attention to reducing power supply noise. A dc power supply filter (Murata BNX002) will provide EMI suppression between the switching power supply and the main PCB. Alternatively, consideration could be given to using a three-terminal voltage regulator.

8.5.3.4. Digital Signal Interconnect

The digital signal lines to the D/A should be isolated as much as possible from the analog outputs and other analog circuitry. Digital signal lines should not overlay the analog power plane.

Due to the high clock rates used, long clock lines to the D/A should be avoided to minimize noise pickup.

Any active pull-up termination resistors for the digital inputs should be connected to the regular PCB power plane (VCC) and not the analog power plane.

8.5.3.5. Analog Signal Interconnect

The D/A should be located as close as possible to the output connectors, thus minimizing noise pickup and reflections due to impedance mismatch.

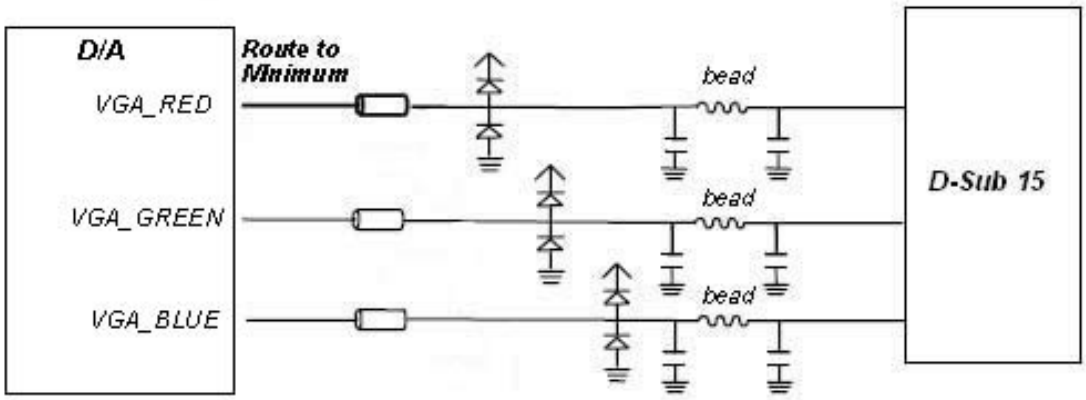
The video output signals should overlay the ground plane and not the analog power plane, thereby maximizing the high frequency power supply rejection.

For optimum performance, the analog outputs should each have a source termination resistance to ground of 75 Ω (doubly terminated 75 Ω configuration). This termination resistance should be as close as possible to the ADV7125 to minimize reflections.

8.5.3.6. RLC Components

Serial ferrite beads for the RGB lines should have high frequency characteristics to eliminate relative noise.

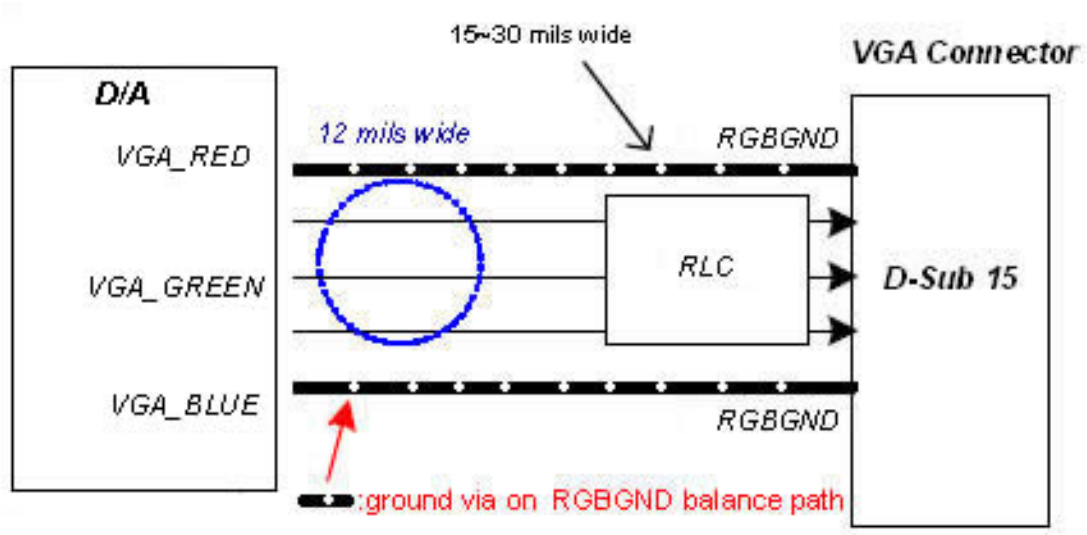
Figure 8.16 VGA layout guidelines



8.5.3.7. RGB Output Current Balance Path

Analog R, G and B (red, green and blue) traces should be designed to be as short as possible. Careful design, however, will allow considerable trace lengths with no visible artifacts. GNDRGB is an "analog current balance path" for the RGB lines. In terms of layout, GNDRGB should follow 2 traces that encapsulate the RGB traces all the way to the D-shell connector (VGA Port) and should not be tied to ground until connected to the Right Angle D-type connector.

Figure 8.17 RGB Output Layout Guidelines



8.6 Ethernet

MXM module supports the Davicom DM9000B IEEE802.3 network interface and flexible dynamically loadable EEPROM algorithm. The network interface complies with the IEEE standard for 10BASE-T and 100BASE-T Ethernet interfaces.

8.6.1. Signal Descriptions

Table 8.13 shows MXM Module Ethernet signals, including pin number, signals, I/O, power plane and descriptions.

Table 8.13 Ethernet Signal Description

Table 8.13 Ethernet Signal Description			
Ethernet			
226	*LANLED1*	*Ethernet Speed LED*	*O*
228	*LANLED2*	*Ethernet Link LED*	*O*
230	*AVDD18*	*1.8V For Transformer*	*P*
232	*TX-*	*Ethernet Transmits data-*	*O*
234	*TX+*	*Ethernet Transmits data+*	*O*
236	*AGND*	*Ethernet Ground*	*P*
238	*RX-*	*Ethernet Receives data-*	*I*
240	*RX+*	*Ethernet Receives data+*	*I*
242	*AVDD18*	*1.8V For Transformer*	*P*

8.6.2. Design Guidelines

8.6.2.1. Differential Pairs

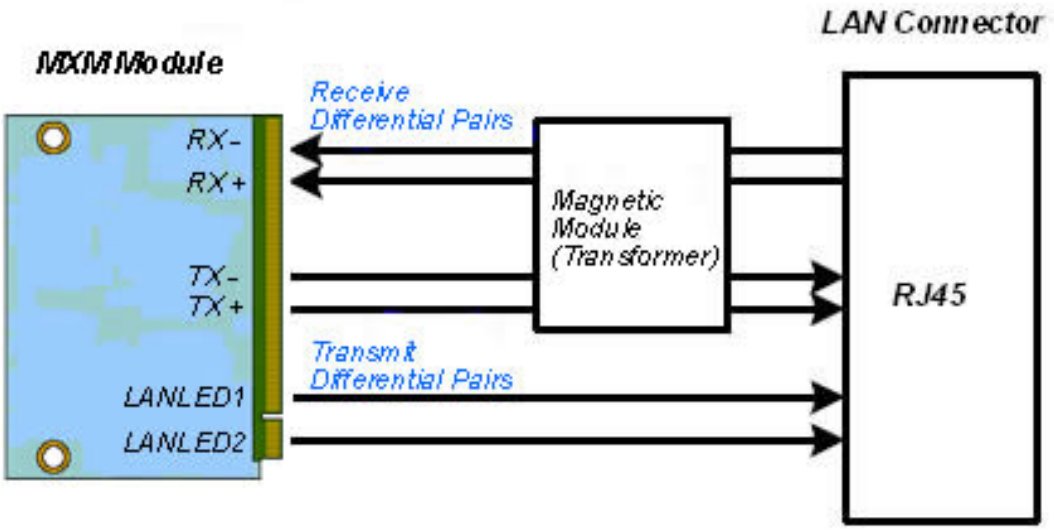
Route the transmit and receive lines on the input (MXM module) side of the coupling transformer on the carrier board PCB as differential pairs, with a differential impedance of 100 Ω. PCB layout software allows determination of the correct trace width and spacing to achieve this impedance after the PCB stack-up configuration is known.

With 10/100M, the TX+, TX- signal pair should be well separated from the RX+, RX- signal pair. Both pairs should be well separated from any other signals on the PCB.

The total routing length of these pairs from the MXM module to the Ethernet jack should be made as short as practical. If the carrier board layout doesn't specify where the Ethernet jack is located, it should be placed close to the MXM module pins.

Figure 8.18 shows the 10/100M Ethernet Connections.

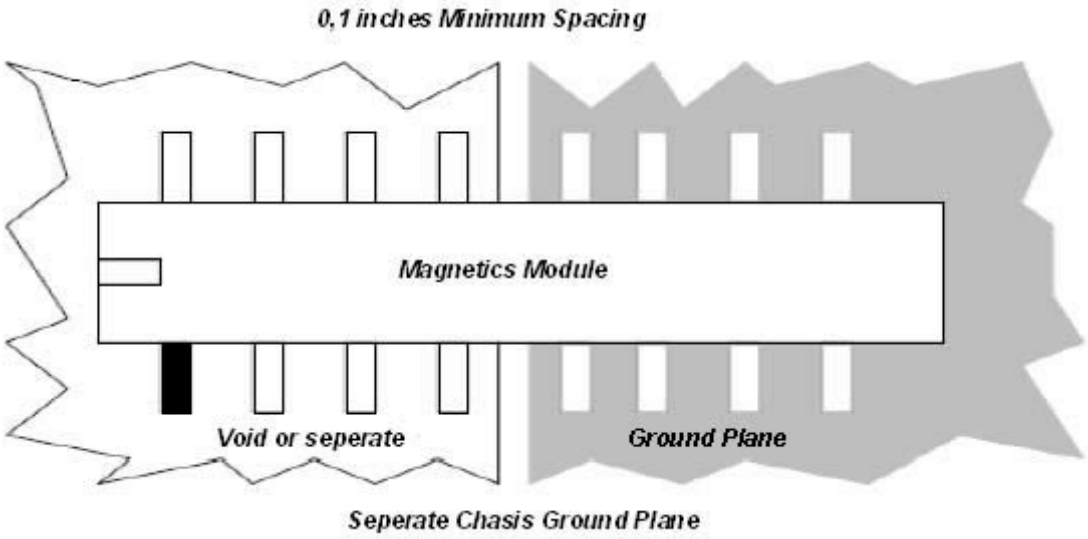
Figure 8.18 10/100Mbps Ethernet Connection



8.6.2.2. Power Considerations and Ethernet LED

In general, any section of traces that are intended for use with high-speed signals should observe proper termination practices. Many board layouts remove the ground plane underneath the transformer and the RJ-45 jack to minimize capacitive coupling of noise between the plane and the external Ethernet cable. Figure 8.19 shows an example.

Figure 8.19 Ground Plane Separations



8.6.3. Layout Guidelines

Critical signal traces should be kept as short as possible to decrease the likelihood of being affected by high frequency noise from other signals; including noise carried on power and ground planes. Keeping the traces as short as possible can also reduce capacitive loading.

8.6.3.1. Placement, Signal and Trace Routing

- Place the 10/100M magnetic as closely as possible to the MXM module (no more than 10mm) and to the RJ-45 connector.
- Traces routed from the MXM module RX± pair to the 10/100M magnetic and the RJ45 connector should run symmetrically, directly, identically, and closely (no more than 2mm). The same rule is applied to traces routed from the MXM module TX± pair.
- It is recommended that RX± receive and TX± transmit traces turn at 45° angle. Do not turn at right angle.
- Avoid using vias in routing the traces of RX± pair and TX± pair.
- Do not place the MXM module RX± receive pair across the TX± transmit pair. Keep the receive pair away from the transmit pair (no less than 3mm). It's better to place ground plane between these two pairs of traces.
- The network interface (see Figure 8.20 and Figure 8.21) does not route any digital signal between the MXM module RX± and TX± pairs to the RJ-45. Keep the two pairs away from all the other active signals and the chassis ground.
- It should be no power or ground plane in the area under the network side of the 10/100M magnetic and the area under the RJ-45 connector.
- Any terminated pins of the RJ-45 connector and the magnetic (see Figure 8.20 and Figure 8.21) should be tied as closely as possible to the chassis ground through a resistor divider network 75Ω resistors (no more than 2mm to the magnetic) and a 0.01μF/2KV bypass capacitor.

Figure 8.20 Better examples for signal and trace routing

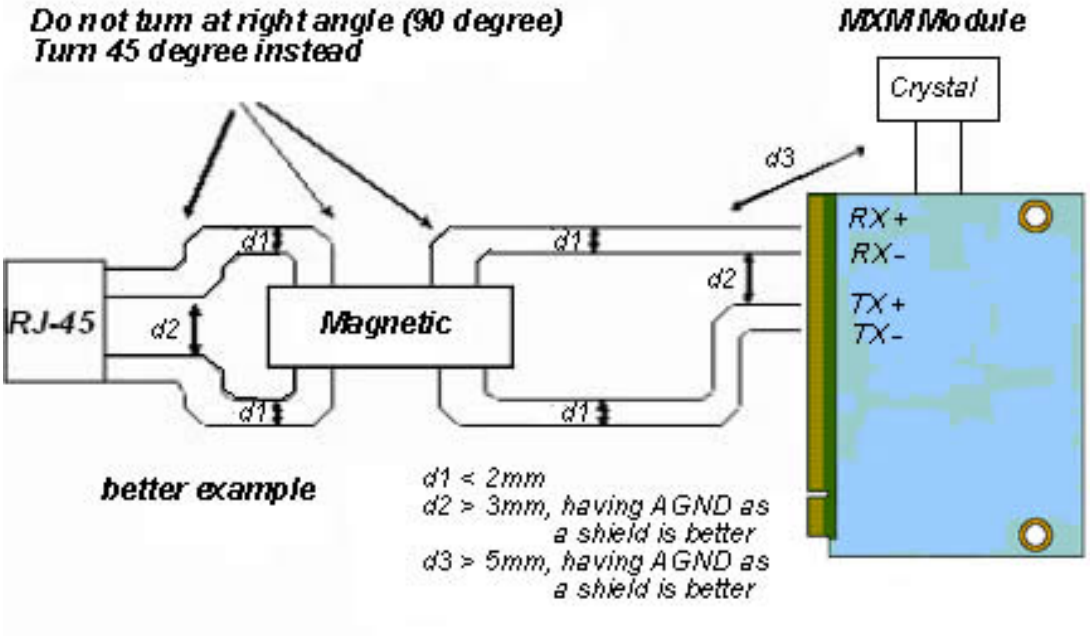


Figure 8.21 Worse examples for signal and trace routing

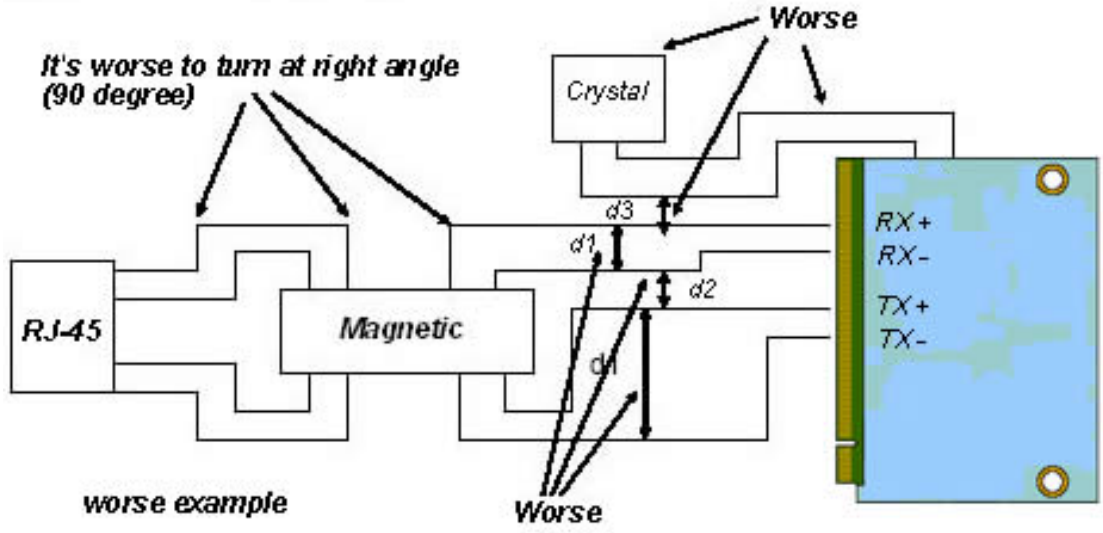
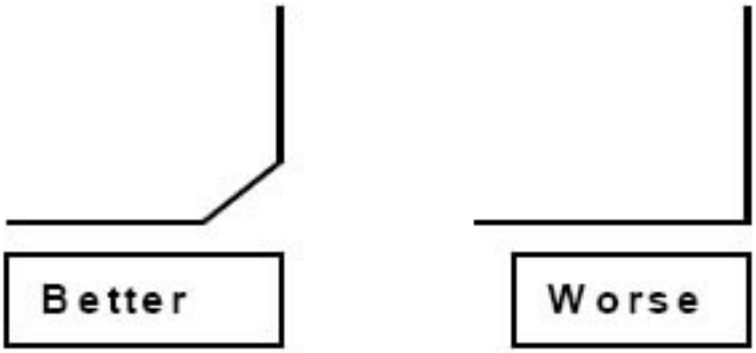


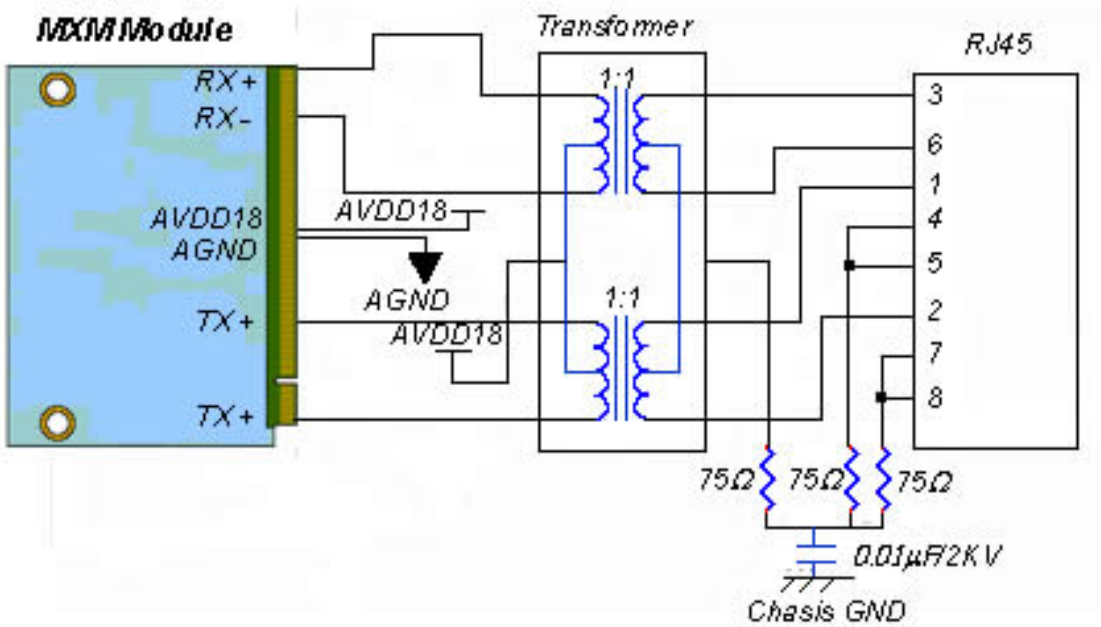
Figure 8.22 Example for better and worse trace



8.6.3.2. MXM Module 10Base-T/100Base-TX Application

Fig. 8.23 illustrate the two types of the specific magnetic interconnect and how to connect with MXM module. These magnetics are not pin-to-pin compatible. It must be considered when using the MXM-module in auto-MDIX mode.

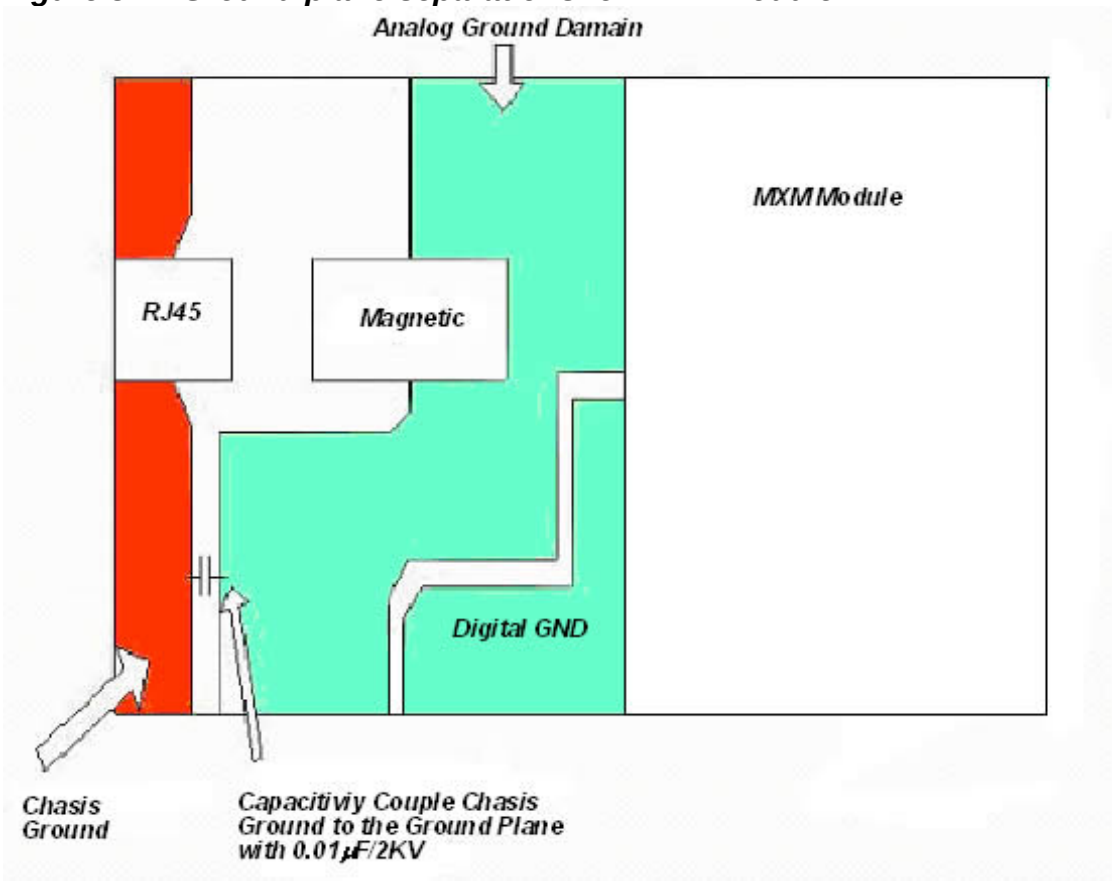
Figure 8.23 Application with auto_MDIX transformer (turn ratio 1CT:1CT)



8.6.3.3. Ground Plane Layout

- Place a single ground plane approach to minimize EMI. Ground plane partitioning can cause increased EMI emissions that could make the network interface circuit not comply with specific FCC part 15 and CE regulations.
- Ground plane need separate analog ground domain and digital ground domain, the analog ground domain and digital ground domain connected line is far away the AGND pins of MXM module (see Figure 8.24.)

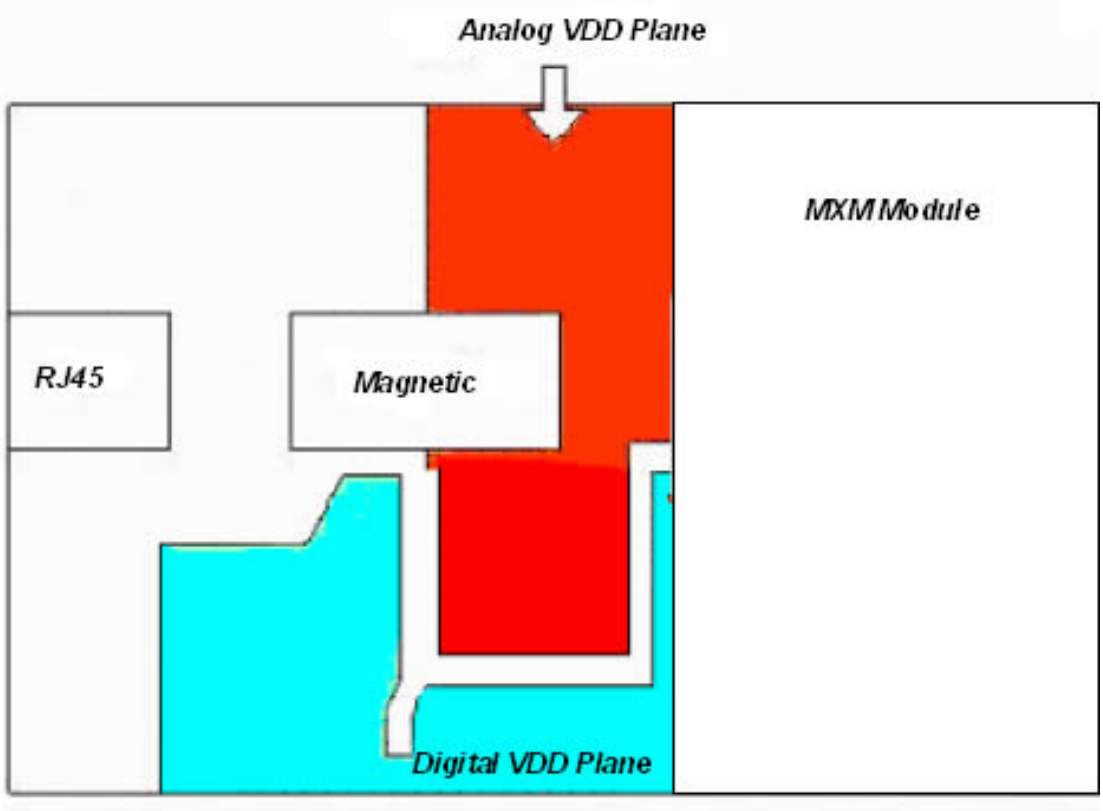
Figure 8.24 Ground plane separations for MXM module



8.6.3.4. Power Plane Partitioning

- The power planes should be approximately illustrated in Figure 8.25. No bead is needed to connect two power planes.
- It should separate analog power planes from noisy digital (logic) power planes.

Figure 8.25 Power planes partitioning for MXM module



8.6.3.5. Magnetic Selection Guide

Refer to the following **t**ables 8.14, 8.15 and 8.16 for 10/100M magnetic sources and specification requirements. The magnetic which meet these requirements are available from a variety of magnetic manufacturers. Designers should test and qualify all magnetic specifications before using them in an application. The magnetic listed in the following table are electrical equivalents, but may not be pin-to-pin equivalents.

Table 8.14 10/100Mbps RJ45 Jack (Magnetic included)

Manufacturer	***Part Number***
Foxconn	*JFM24011-0101-4F*
YCL	*PTC1111-09L1FG*

Table 8.15 10/100Mbps Magnetic Sources

Manufacturer	***Part Number***
Pulse Engineering	*PE-68515, H1102*
YCL	*PH163112, PH163539*
Halo	*TG110-S050N2, TG110-LC50N2*
Bel Fuse	*S558-5999-W2*
GTS	*FC-618SM*
MACOM	*HS9016, HS9024*

Table 8.16 Magnetic Specification Requirements

Parameter	***Values***	***Units***	***Test Condition***
Tx/Rx turns ratio	*1:1 CT/1:1*	*-*	*-*
Inductance	*350*	*μH (Min)*	*-*
Insertion loss	*1.1*	*DB (Max)*	*1-100Mhz*
Return loss	*-18*	*DB (Min)*	*1-30 Mhz*
	-14	*DB (Min)*	*30-60 Mhz*
	-12	*DB (Min)*	*60-80 Mhz*
Differential to common mode rejection	*-40*	*DM (Min)*	*1-60 Mhz*
	-30	*DB (Min)*	*60-100 Mhz*
Transformer isolation	*1500*	*V*	*-*

Carrier Board Mechanical Design Guidelines

A detail description of mechanical design guidelines for the MXM computer on module carrier boards is provided in this chapter.
Section include：

- MXM Motherboard Footprint

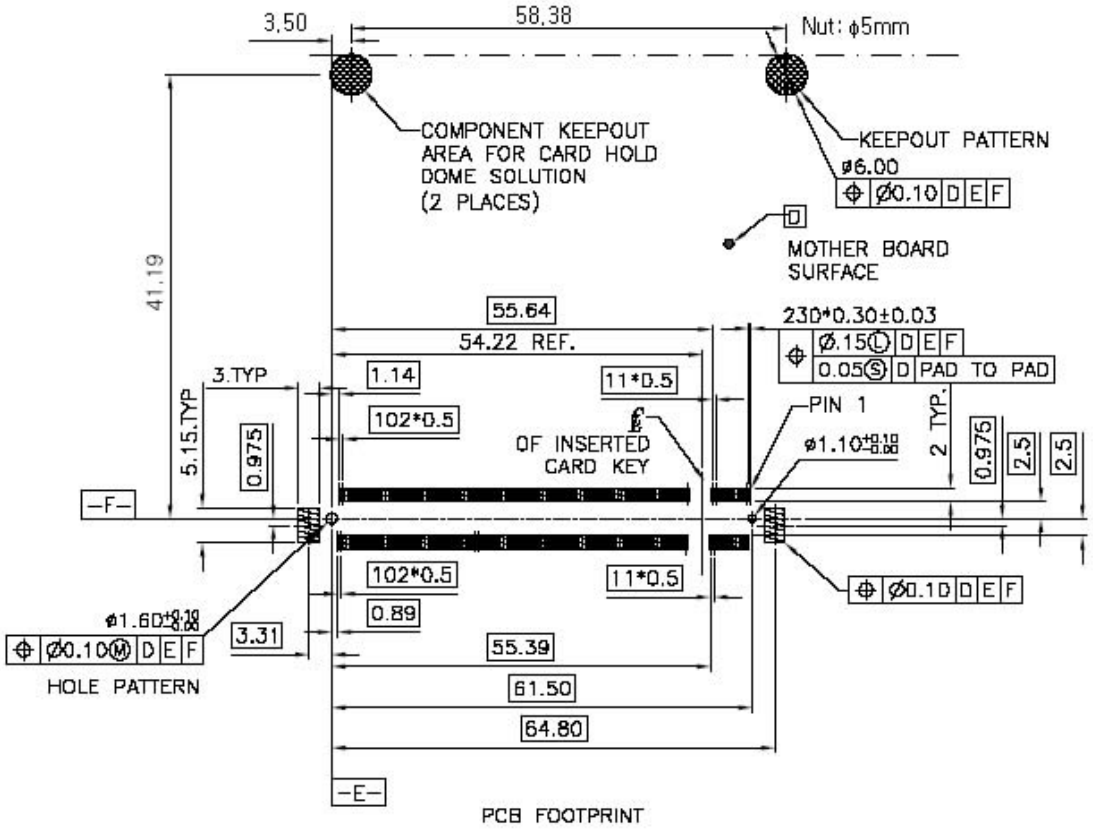
Chapter 9 Carrier Board Mechanical Design Guidelines

This section gives detail description of the mechanical design recommendation of the MXM computer on module carrier boards.

9.1 MXM Motherboard Footprint

Two drew holes in carrier boards to fix the MXM modules. The height of support pillars depends on the MXM connector height option that customers choose. In the evaluation kit that Embedian offers, the height of MXM connector is 5mm and the height of support pillars is also 5mm. The screws that Embedian use is M3 (Metric 3mm), F (Flat) head, 4mm long, 5mm in head diameter, and 1mm head in thick. Figure 9.1 shows the MXM carrier board footprint.

Figure 9.1 MXM Motherboard Footprint



For detail connector mechanical drawing, pin numbering and manufacturers, please also refer to Chapter 4.

Chapter 10

Pin Definition Differences between Embedian MXM modules

This chapter describes the pin definition differences of the 242-pin golden fingers between Embedian MXM modules.

Chapter 10 *Pin Definition Differences between Embedian MXM modules*

This chapter describes the pin definition differences of the 242-pin golden fingers between Embedian MXM modules. It is therefore; user can easily make a comparison when upgrade their modules to share the same basebaord.

Table 10.1 Pin Definition Differences between Embedian MXM modules (Top Side)

Table 10.1 Pin Definition Differences between Embedian MXM modules (Top Side)			
Pin	*MXM-7110/MXM-7114*	*MXM-8310*	*MXM-6410*
9~15	*ADC Input*	*N.C.*	*ADC Input*
41	*DMAACK0*	*N.C*	*TBD*
99	*Address 26*	*N.C.*	*TBD*
101~113	*N.C.*	*PC Card Bus Related*	*TBD*
179	*Boot ROM Select*	*N.C.*	*TBD*
183	*N.C.*	*One Wire Bus*	*TBD*
199~241	*N.C.*	*USB 2.0 Client UTMI Interface*	*TBD*

Table 10.2 Pin Definition Differences between Embedian MXM modules (Bottom Side)

Table 10.2 Pin Definition Differences between Embedian MXM modules (Bottom Side)			
Pin	*MXM-7110/MXM-7114*	*MXM-8310*	*MXM-6410*
104~118	*SPI Interface*	*SSP Interface (*)*	*TBD*
194~224	*UART 0,1,2*	*UART 1,2,3 (**)*	*TBD*

(*)SSP interface in MXM-8310 can be configured as SPI interface by software.
(**) The UART numbering just followed the respectively S3C2440 and PXA320 CPU manual.

Firmware Upgrade

This Chapter details how to update firmware in NAND flash.
Section include：

- Firmware Architecture
- Update Firmware from blob

Appendix I MXM-8310 Firmware Update

This Chapter details firmware upgrade for MXM-8310. The firmware in NAND flash includes blob, kernel zImage and nandfs image. This guide mainly uses MXM-8310 on the evaluation kit as an example.

A.1. Firmware Architecture

Figure A.1 shows the firmware architecture of Linux in NAND Flash.

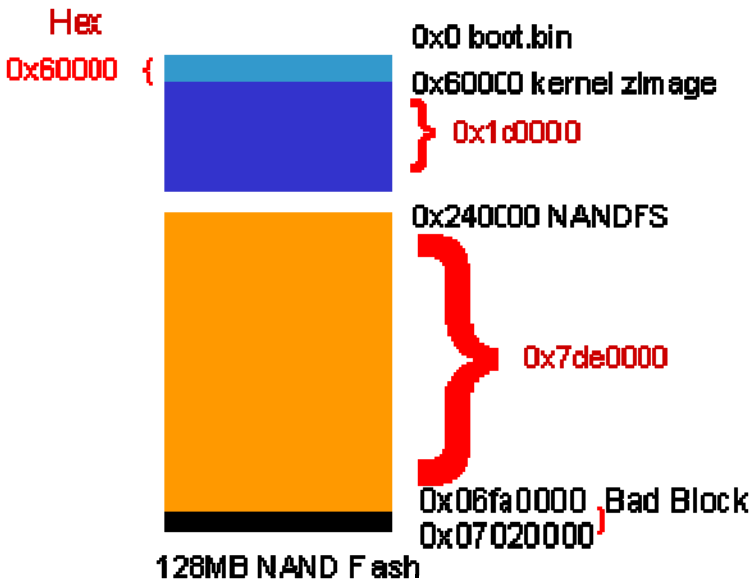


Figure A.1 Firmware Architecture of Linux in NAND Flash

The boot.bin starts from NAND address 0x0. The Linux kernel zImage starts from NAND address 0x60000. The NAND filesystem is a small file system for rescue purposed or system that would only need simply application and load the minimum set drivers and starts from the NAND address 0x240000. The last partition is reserved for NAND bad block tables.

Users need a CF card with root file system installed to boot up the complete root file system. The will be described at Chapter 6.

Users can update the firmware under blob or use In-Circuit Emulator (ICE) debug tool. The Embedian factory default is fimware pre-installed. Unless necessary, Embedian doesn't recommend you update firmware (especially blob) since the system might not boot anymore if you did the wrong operation. (If you develop your own blob and kernel, you will need to do that. You can send your image to Embedian to let us test for you first.) Following tells you howto update firmware from blob command prompt.

First, you need to prepare for proper header for processor to load from NAND to mobile DDR. To do that, you will need to combine three files (*MHP_linux_NTOBM.bin*, *OBM_NTIM.bin* and *blob*) into one file called "*boot.bin*" by the following instruction.

In a Linux PC, put the following five files into same directory. They are:

MHP_linux_NTOBM.bin
OBM_NTIM.bin
blob
link_image
descr_mxm-8310.txt

Execute the following command.

```
root@dns3:~# ./link_image -d descr_mxm-8310
```

This will create a file called "*boot.bin*". This file is the bootloader of MXM-8310.

Now, you are ready to update firmware from blob.

A.2. Update Firmware from blob

You could use blob tftp command to download blob, Linux kernel and NAND root file system. Below we will tell you how to do this under Windows and Linux PC environment. First, you need to set up a tftp server.

A.2.1. Windows Environment

Open up Windows Hyperterminal and set up the serial port (*115200, 8N1*).

A.2.1.1. Setup TFTP Server/Client IP Address from Device

Users need to install tftp server on Windows. You can download the freeware and install to your Windows PC in the **tftproot** directory. Copy the **boot.bin**, **zImage** and **nand.img** into this directory. Close your anti-virus software like PC-cillin. (Or close port 69)

First, power on the device with console debug port connected to your PC and Ethernet cable connected to a local network and go to blob command prompt by pressing any key at boot. You will see blob command prompt like this.

```
blob >
```

You can set and add the *tftp* ip address of the device and server by using "***setip***", command as below.

```
blob > setip server <ip>
blob > setip client <ip>
```

Following figure shows the example for setting up the parameters.

EXAMPLE:

```
blob> setip server 192.168.1.10
Set IP to 192.168.1.10
blob> setip client 192.168.1.202
Set IP to 192.168.1.202
blob>
```

Note:
Make sure that the ***server*** ip for Windows PC and ***client*** *ip* for MXM-8310 evaluation kit are in the same network domain.

After setting up the IP address and wire everything right, you could start the tftp download.

A.2.1.2. Transfer and Write Image by TFTP and "nandwrite" Command

After setting up the tftp server and IP address of devices, users can start transfer and write images using blob *tftp* and *nandwrite* command. It is necessary to download to DRAM first before writing to NAND.

Boot.bin
The following command shows how to transfer ***boot.bin*** images to DRAM. To update bootloader:

```
blob> unlock_bootarea
NAND boot area unlocked.
blob> tftp boot.bin
TFTPing 1*^^^^^^^^^^^^^^^^^^^^^^^^^^^^ OK.
received 770 blocks (393216 bytes)
tftp_cmd: file `boot.bin` loaded via tftp to address 0x80800000.
blob>
```

Now, boot.bin file has been upload to DRAM temporary address from your PC. Temporary address is base address of DRAM, Default is set to *0x80800000*. From the log, the file size of *boot.bin* is *393216* bytes (*60000* bytes in HEX).

Write the **boot.bin** image from DRAM to the NAND by using following *nandwrite* command.

```
blob> nandwrite -j 0x80800000 0x0 0x60000
Write 0x60000 length data from RAM: 0x80800000 to flash: 0x0
```

```
Write flash from 0x0, length 0x60000
Erase flash from 0x0, length 0x60000
...Done
...Done
blob>
```

Temporary address is base address of DRAM, i.e. 0x80800000. Start NAND address is 0x0. Image size of bootloader is *0x60000* (HEX) as we have seen previously when tftping.

Note: blob contains specific hardware information and is well configured by Embedian. It is usually no need to modify. Unless necessary or you are an experienced engineer, it is not recommended to update blob (boot.bin). Wrong operation will cause the system not booting anymore.

zImage

Next example shows how to transfer and write Linux kernel. The file name is "**zImage**". Again, we tftp zImage from PC to DRAM first by the following command.

```
blob> tftp zImage
dm9000: rx fifo error
dm9000: rx crc error
dm9000: rx length too big
TFTPing
zImage*^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
^^^^^^^^^^^^^^^^^^^^^^^^^^^^ OK.
received 2824 blocks (1444980 bytes)
tftp_cmd: file `zImage` loaded via tftp to address 0x80800000.
blob>
```

Now, zImage file has been upload to DRAM temporary address from your PC. Temporary address is base address of DRAM, Default is set to 0x80800000. From the log, we also learned that the file size is *1444980* bytes (*160C74* bytes in HEX).

Write the **zImage** image from DRAM to the NAND by using *nandwrite* command again.

```
blob> nandwrite -j 0x80800000 0x60000 0x160c74
Write 0x160c74 length data from RAM: 0x80800000 to flash: 0x60000
Write flash from 0x60000, length 0x160c74
```

```
Erase flash from 0x60000, length 0x161000
............Done
............Done
blob>
```

Temporary address is base address of DRAM, i.e. *0x80800000*. Start NAND address is *0x60000*. Image size of zImage is *160c74* (HEX) as we have seen previously when tftping.

nand_fs.ext3.img

After writing boot.bin and kernel images to NAND flash, the last step is to write NAND root file system ***nand_fs.ext3.img*** image. There are some differences from the above boot.bin and zImage.
The mobile DDR size is 128MB only. If your NANDFS size is big (like Embedian default nand_fs.ext3.img is 114.6M), you need to use *split* command in Linux to split the NANDFS into two smaller files, or the blob will be overwritten in DDR because the execution point of blob is somewhere in DDR. Once you split two smaller files, it will be a good practice for you to flash nandfs into NAND flash. Just remember that the starting address of nandfs partition is *0x240000*.
After done, reset MXM-8310 evaluation kit and the firmware will be updated.

A.2.2 Linux Environment

In this section, we will detail how to transfer and write firmware under Linux PC. First, we need to set up minicom so that we could see the message from the console port.

A.2.2.1. Minicom

Before transferring images using tftp, you should know how to use Minicom so that you could see the messages from console port. In this section will explain how to setup Minicom.

Desktop Linux has Minicom program for serial communication. It is used for command prompt of blob or shell prompt of embedded Linux.

Set up the values before using Minicom program. To execute minicom on setting mode:

```
root@dns2:~# minicom -s
```

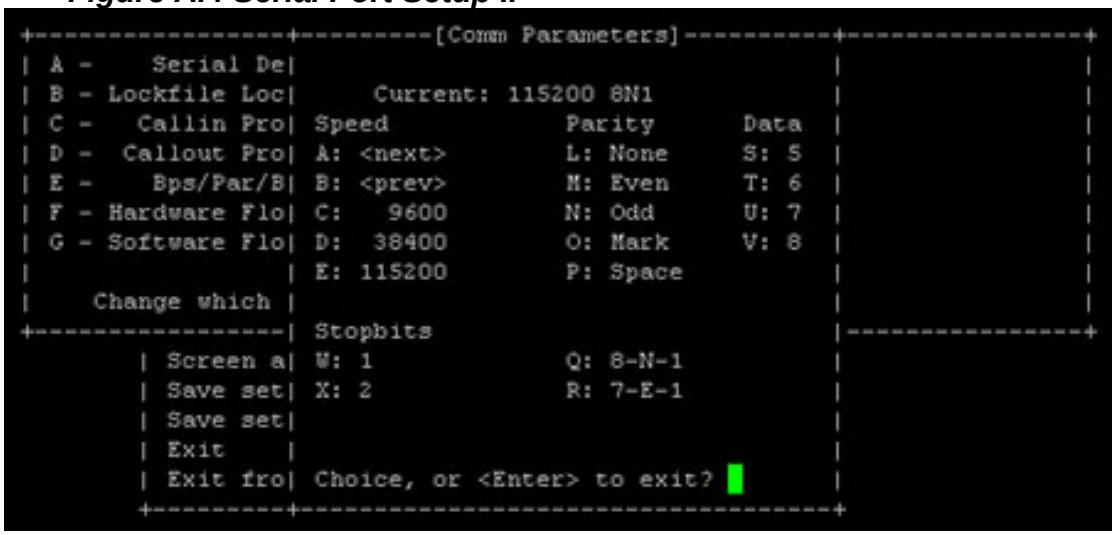
Figure A.2 Minicom Setup

Please select '***Serial port setup***'. Select ***'A***' for setting '***Serial Device***', then type the device descriptor of serial port in your PC which is connected to the MXM-8310 evaluation kit. (You need to figure out the device descriptor of COM port in your Linux PC. In our example, it is ***/dev/ttyS0***)

Figure A.3 Serial Port Setup I

Select '***E***' for setting up '***Bps/Par/Bits***' and enter the next screen. Select '***I***' to set up '***bps***' to **115200**. Select '***V***' to set up '***Data bits***' to **8**. Select '***W***' to set up '***Stop bits***' to '***1***', and '***L***' to set up '***parity***' to '***NONE***'. After done, press, '**Enter**' to save and exit this screen.

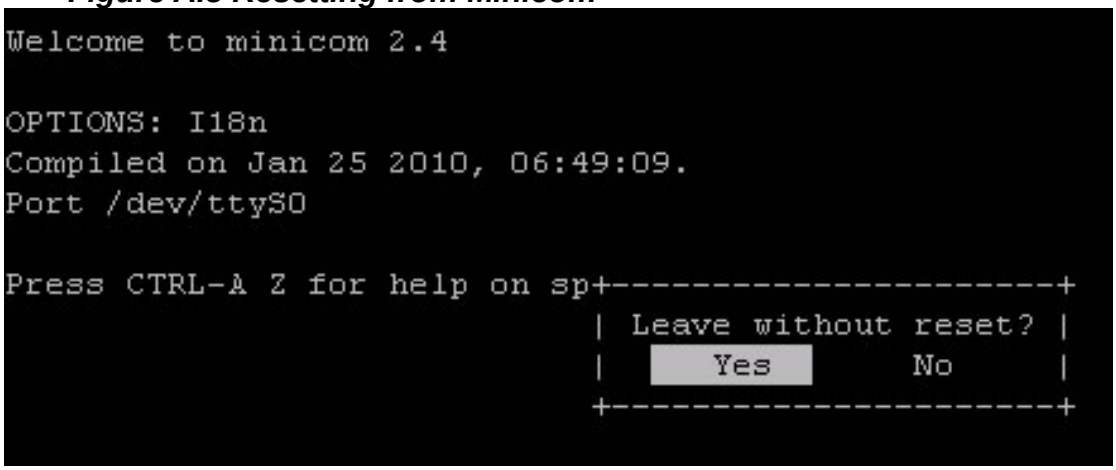
Figure A.4 Serial Port Setup II



Push '***F***' key for setting up '***Hardware Flow Control***' to '***NO***'.
Push '***G***' key for setting up '***Software Flow Control***' to '***NO***'. The default value is '***NO***'. Please refer to figure 3.
Once setting is done, please press '***Enter'*** key. And select '***Save setuo as ..'***. Save the configuration as a *<filename>* then press '***Exit'*** to exit the minicom setup program.

To quit from ***Minicom***, please press **'*Ctrl + A*'** and then **'*Z*'**, at last push **'*Q*'** key. Then Selecting **'*Yes*'**, ***Minicom*** is quitted.

Figure A.5 Resetting from Minicom



To use minicom,

root@dns2:~# minicom < filename>

A.2.2.2. TFTP server in Linux PC

MXM-8310 evaluation kit communicates firmware to PC via tftp protocol. It is therefore; you need to install a tftp server first in your Linux PC. This section uses Ubuntu as an example and tells you how to install and set up a tftp server.

First of all, since tftp server is not s stand-alone package, you need to install *tftpd (server)*, *tftp (client)*and *xinetd* packages.

```
root@dns2:~# sudo apt-get install xinetd tftpd tftp
```

Next, create a file called *tftp* under */etc/xinetd.d/* directory.

```
root@dns2:~# sudo vim /etc/xinetd.d/tftp
```

And put this entry:

```
service tftp
{
protocol        = udp
port            = 69
socket_type     = dgram
wait            = yes
user            = nobody
server          = /usr/sbin/in.tftpd
server_args     = /tftpboot
disable         = no
}
```

The last is to make a */tftproot* directory.

```
root@dns2:~# sudo mkdir /tftproot
root@dns2:~# sudo chmod -R 777 /tftproot
```

Start the tftpd through xinetd and you are done.

```
root@dns2:~# sudo /etc/init.d/xinetd start
```

Now you can download compiled images to the MXM-8310 evaluation kit by using ***tftp***. Before downloading the images, connect host PC and MXM-8310 evaluation kit by Ethernet cable.

To download binary image files to MXM-8310, run tftp server service on your computer and put images in ***/tftproot*** directory.

A.2.2.3. Setting up an IP address

Setting up an IP address helps in downloading the compiled images to MXM-8310.
Connect host PC and MXM-8310 evaluation kit by Ethernet cable.

A.2.2.3.1. Setting IP address for host PC

On Your Linux Host PC, run the terminal and execute following commands to set up and IP address.

```
[root@localhostt]# ifconfig eth0 down
[root@localhost]# ifconfig eth0 192.168.1.10 netmask 255.255.255.0 up
[root@localhost]# ifconfig
```

A.2.2.3.2. Setting TFTP Server/Client IP address form MXM-8310/XPC-8310

Below will be exactly the same as that in Windows environment. Copy the ***boot.bin***, ***zImage*** and***nand.img*** into /tftproot directory in your host PC. Run the ***Minicom*** first in your host PC and power on MXM-8310 evaluation kit. Power on the device with console debug port connected to your PC and Ethernet cable connected to local network and go to blob command prompt by pressing any key at boot.

You will see blob command prompt like this.

```
blob >
```

You can set and add the *tftp* ip address of the device and server by using "***setip***", command as below.

```
blob > setip server <ip>
blob > setip client <ip>
```

Following figure shows the example for setting up the parameters.

EXAMPLE:

```
blob> setip server 192.168.1.10
Set IP to 192.168.1.10
blob> setip client 192.168.1.202
Set IP to 192.168.1.202
blob>
```

Note:
Make sure that the ***server*** ip for Windows PC and ***client*** *ip* for MXM-8310 evaluation kit are in the same network domain.

After setting up the IP address and wire everything right, you could start the tftp download.

A.2.2.4. Transfer and Write Image by TFTP and "nandwrite" Command

After setting up the tftp server and IP address of devices, users can start transfer and write images using blob *tftp* and *nandwrite* command. It is necessary to download to DRAM first before writing to NAND.

Boot.bin

The following command shows how to transfer ***boot.bin*** images to DRAM. To update bootloader:

```
blob> unlock_bootarea
NAND boot area unlocked.
blob> tftp boot.bin
TFTPing 1*^^^^^^^^^^^^^^^^^^^^^^^^^^^^ OK.
received 770 blocks (393216 bytes)
tftp_cmd: file `boot.bin` loaded via tftp to address 0x80800000.
blob>
```

Now, boot.bin file has been upload to DRAM temporary address from your PC. Temporary address is base address of DRAM, Default is set to *0x80800000*. From the log, the file size of *boot.bin* is *393216* bytes (*60000* bytes in HEX).

Write the **boot.bin** image from DRAM to the NAND by using following *nandwrite* command.

```
blob> nandwrite -j 0x80800000 0x0 0x60000
Write 0x60000 length data from RAM: 0x80800000 to flash: 0x0
Write flash from 0x0, length 0x60000
Erase flash from 0x0, length 0x60000
...Done
...Done
blob>
```

Temporary address is base address of DRAM, i.e. 0x80800000. Start NAND address is 0x0. Image size of bootloader is *0x60000* (HEX) as we have seen previously when tftping.

Note: blob contains specific hardware information and is well configured by Embedian. It is usually no need to modify. Unless necessary or you are an

experienced engineer, it is not recommended to update blob (boot.bin). Wrong operation will cause the system not booting anymore.

zImage

Next example shows how to transfer and write Linux kernel. The file name is "**zImage**". Again, we tftp zImage from PC to DRAM first by the following command.

```
blob> tftp zImage
dm9000: rx fifo error
dm9000: rx crc error
dm9000: rx length too big
TFTPing
zImage*^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
^^^^^^^^^^^^^^^^^^^^^^^^^^^^ OK.
received 2824 blocks (1444980 bytes)
tftp_cmd: file `zImage` loaded via tftp to address 0x80800000.
blob>
```

Now, zImage file has been upload to DRAM temporary address from your PC. Temporary address is base address of DRAM, Default is set to 0x80800000. From the log, we also learned that the file size is *1444980* bytes (*160C74* bytes in HEX).

Write the **zImage** image from DRAM to the NAND by using *nandwrite* command again.

```
blob> nandwrite -j 0x80800000 0x60000 0x160c74
Write 0x160c74 length data from RAM: 0x80800000 to flash: 0x60000
Write flash from 0x60000, length 0x160c74
Erase flash from 0x60000, length 0x161000
............Done
............Done
blob>
```

Temporary address is base address of DRAM, i.e. *0x80800000*. Start NAND address is *0x60000*. Image size of zImage is *160c74* (HEX) as we have seen previously when tftping.

nand_fs.ext3.img

After writing boot.bin and kernel images to NAND flash, the last step is to

write NAND root file system ***nand_fs.ext3.img*** image. There are some differences from the above boot.bin and zImage.
The mobile DDR size is 128MB only. If your NANDFS size is big (like Embedian default nand_fs.ext3.img is 114.6M), you need to use *split* command in Linux to split the NANDFS into two smaller files, or the blob will be overwritten in DDR because the execution point of blob is somewhere in DDR. Once you split two smaller files, it will be a good practice for you to flash nandfs into NAND flash. Just remember that the starting address of nandfs partition is *0x240000*.
After done, reset MXM-8310 evaluation kit and the firmware will be updated.