User's Manual

Computer on Module
COM Ports
Two USB Hosts
LCD
Ethernet
CompactFlash
Dual Displays

MXM-7114
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
Copyright © 2006 EMBEDIAN, INC..
All rights reserved. No part of this manual may be reproduced, transmitted, transcribed, stored in a retrieval system, or translated into any language or computer language, in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), without the express written permission of EMBEDIAN.

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.
CHAPTER 6 BACKUP AND RESTORE THE ROOT FILE SYSTEM IN CF CARD ............74
  6.1 BACKUP THE ROOT FILE SYSTEM IN CF CARD......................................................74
  6.2 RESTORE THE ROOT FILE SYSTEM IN CF CARD ................................................76
  6.3 DEBIAN ETCH ...........................................................................................................79

CHAPTER 7 GENERAL PCB DESIGN RECOMMENDATIONS .........................81
  7.1 NOMINAL BOARD STACK-UP ......................................................................................81
  7.1.1. Four Layer Board Stackup ....................................................................................82
  7.1.2 Six Layer Board Stackup .........................................................................................83
  7.2 DIFFERENTIAL IMPEDANCE TARGETS FOR MICROSTRIP ROUTING .............85
  7.3 ALTERNATIVE STACK UPS .......................................................................................85

CHAPTER 8 CARRIER BOARD DESIGN GUIDELINES ..................................87
  8.1 GENERAL CIRCUIT DESIGN GUIDE .................................................................88
    8.1.1. System-Wise ........................................................................................................88
  8.2 UNIVERSAL SERIAL BUS (USB) ............................................................91
    8.2.1. Universal Serial Bus (USB) .................................................................................91
    8.2.2. Signal Description ...............................................................................................91
    8.2.3. Design Guidelines ..............................................................................................91
    8.2.3. Layout Guidelines ..............................................................................................92
  8.3 AC-LINK INTERFACE .........................................................................................95
    8.3.1. Signal Description ..............................................................................................95
    8.3.2. Design Guidelines ............................................................................................95
    8.3.3. Layout Guidelines .............................................................................................97
  8.4 TTL/LVDS LCD .................................................................................................102
    8.4.1. Signal Description .............................................................................................102
    8.4.2. Design Guidelines ............................................................................................104
    8.4.3. Layout Guidelines .............................................................................................105
  8.5 VGA .......................................................................................................................107
    8.5.1. Signal Description .............................................................................................107
    8.5.2. Design Guidelines ............................................................................................108
    8.5.3. Layout Guidelines .............................................................................................108
  8.6 ETHERNET .............................................................................................................110
    8.6.1. SIGNAL DESCRIPTIONS ..................................................................................110
    8.6.2. DESIGN GUIDELINES ......................................................................................110
      8.6.2.1. Differential Pairs .........................................................................................110
      8.6.2.2. Power Considerations and Ethernet LED ....................................................111
    8.6.3. LAYOUT GUIDELINES ....................................................................................112
      8.6.3.1. Placement, Signal and Trace Routing ..............................................................112
      8.6.3.2. MXM Module 10Base-T/100Base-TX Application ..........................................114
      8.6.3.3. Ground Plane Layout ..................................................................................115
      8.6.3.4. Power Plane Partitioning .............................................................................116
      8.6.3.5. Magnetic Selection Guide ............................................................................117

CHAPTER 9 CARRIER BOARD MECHANICAL DESIGN GUIDELINES ...............119
  9.1 MXM MOTHERBOARD FOOTPRINT ............................................................120

CHAPTER 10 PIN DEFINITION DIFFERENCES BETWEEN EMBEDIAN MXM MODULES .................................................................124
  A.1. FIRMWARE ARCHITECTURE ..............................................................................127
    A.2. UPDATE FIRMWARE FROM UBOOT .................................................................128
      A.2.1. Windows Environment ....................................................................................128
      A.2.1.1. Setup TFTP Server/Client IP Address from Device ....................................128
      A.2.1.2. Transfer and Write Image to NAND by TFTP and “nandw” Command .......130
      A.2.2 Linux Environment .........................................................................................135
      A.2.2.1. Minicom ......................................................................................................135
      A.2.2.2. TFTP server in Linux PC ............................................................................140
      A.2.2.3. Setup TFTP Server/Client IP Address from Device ....................................140
Introduction

This Chapter gives background information on the MXM-7114
Section include:
- MXM computer on Module Family
- Comparison of MXM series 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.

It is designed to meet the needs for embedded networking and graphic enhanced systems, especially for mobile or stationary computer in automatic data collection field such as RFID terminals, batch/ wireless data collection terminals, M2M, medical, POS terminals, fiscal printer, fiscal printer, biometric access control terminals, transportation, transaction terminals, portable test instrument, advanced remote controller, and GPS systems for retail, light industrial and medical/pharmaceutical applications. With Debian Etch pre-installed, people could easily develop their programs and make it time to market in very short time.

Based on Samsung ARM920T SoC plus Silicon Motion SM502 graphic enhanced, the MXM-7114 includes 64MB of NAND Flash, and 64MB of SDRAM. Additional interfaces includes TFT LCD interface, two USB host interface, Ethernet interface, five RS232 interface, CF interface, IDE interface, AC97 Audio Interface, IIC, PWM and SPI through a 242-pin MXM golden finger connector as interface. It also includes a sophisticate power management mechanism. The small in size makes system integrators and manufacturers flexible in designing their product line with different shapes and, with firmware pre-installed, to make the products fast to market.

A 242-pin golden finger connector enables the MXM-7114 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.

1.2 Comparison of MXM Series Computer on Module Family

MXM series computer on module families host a Samsung S3C2440A SOC (ARM920T core) clocking at 400Mhz with MMU and AMBA BUS. The differences between each model are listed in the following table. Users could decide the best features and choose the right model number.
Table 1.1

<table>
<thead>
<tr>
<th></th>
<th>MXM-7110</th>
<th>MXM-7114</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processor</td>
<td>S3C2440A</td>
<td>S3C2440A</td>
</tr>
<tr>
<td>Clocking Rate</td>
<td>400Mhz</td>
<td>400Mhz</td>
</tr>
<tr>
<td>NAND Flash</td>
<td>64MB</td>
<td>64MB</td>
</tr>
<tr>
<td>SDRAM</td>
<td>64MB</td>
<td>64MB</td>
</tr>
<tr>
<td>RS232</td>
<td>3 (from CPU)</td>
<td>5 (three from CPU and two from SM502)</td>
</tr>
<tr>
<td>USB Host 1.1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Connector</td>
<td>242-pin MXM golden finger</td>
<td>242-pin MXM golden finger + 60-pin 0.5mm pitch SMT board-to-board connector (**)</td>
</tr>
<tr>
<td>CompactFlash</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>10/100 Mbps Ethernet</td>
<td>1 x DM9000B</td>
<td>1 x DM9000B</td>
</tr>
<tr>
<td>LCD</td>
<td>from CPU (up to 640x480)</td>
<td>from CPU and SM 502 (up to 1280x1024)</td>
</tr>
<tr>
<td>Touch Panel</td>
<td>4-wire FPC</td>
<td>4-wire FPC</td>
</tr>
<tr>
<td>GPIO</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>PWM</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>RTC</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>WatchDog Timer</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>IIC</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>AC97</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>SPI</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>IDE</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>System BUS</td>
<td>16-bit</td>
<td>16-bit</td>
</tr>
</tbody>
</table>

**Note:**
(*) MXM-7114 = MXM+7110 + SM502 related functionality.
(**) The 60-pin 0.5mm pitch SMT board-to-board connector is all SM502 related signal. For MXM-7110, user doesn’t need this connector. And For other manufacturers that would like to adopt this form factor, add this type of connector if you have companion chip other than CPUs.

The MXM-7114 is a member of MXM series computer on module line families and is designed in a 66mm x 50mm factor.
1.3 Block Diagram

The following diagram illustrates the system organization of the MXM-7114. Arrows indicate direction of control and not necessarily signal flow.

Figure 1.1 MXM-7114 Block Diagram

Details for this diagram will be explained in the following chapters.
This Chapter contains specifications of MXM-7114.
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-7114. The specification includes functional, mechanical, electrical, and environmental specifications.

2.1 Functional Specifications

Processor
- On-board Samsung S3C2440A
- 32-bit ARM920T Core
- Clock Rates up to 400Mhz
- 133Mhz System BUS
- Voltage and Frequency Scaling
- Booting from NAND Flash

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

Memory
- Onboard 64MB NAND Flash
- Onboard 64MB SDRAM (32-bit)
- 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, one can be configured as a slave port)
- OHCI Rev. 1.0 Compliance
- USB legacy keyboard, mouse and hard disk support

COM Port
- Chipset : CPU internal and SM502
- Five RS232 ports, (three from CPU and two from SM502)
- Four with TX, RX, CTS and RTS. One with TX, RX only.

CompactFlash(CF) Interface
- Chipset : CPU Memory Bus
- Type I and Type II, 3.3V
- True IDE 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
Embedian, Inc.

- Full-duplex/half-duplex capability
- Supports IEEE 802.3x full duplex flow control
- Auto-MDIX support

**IDE Interface**
- Chipset: CPU Memory Bus
- ATA PIO Mode

**AC97 Audio-Codec Interface**
- Chipset: CPU AC-link interface
- Support 16-bit Sample
- AC97 version 2.3 compliance interface
- 1-ch stereo PCM inputs/1-ch stereo PCM outputs/1-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.

**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

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

**SM502 Video Graphic Array (VGA)**
- Chipset: Silicon Motion SM502
- TFT Panel Support
- 1280 x 1024 resolution
- 260,000 color support
- External 8MB SDRAM onboard
- 24-bit interface

**CPU Video Graphic Array (VGA)**
- Chipset: CPU internal
- TFT/STN Panel Support
- 640 x 480 resolution for best performance
- 65,000 color support
- TTL interface
8/16-bit color depth

**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

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

**JTAG**
- Testing and debugging interface

**BIOS**
- Universal Bootloader (u-boot)
- Serial or Ethernet TFTP download
- Booting from NAND Flash Technology

**Operating System**
- Linux 2.6.18 (Debian ARM Linux, Rootfs stored in CF, Uboot and Kernel Stored in NAND)

**Cross Toolchain**
- Based on gcc 3.4.1

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

**Packing List**
- 1 x MXM-6410

**Document Deliverable**
- uboot (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-7114 (normal temperature)
- MXM-7114-E (extended temperature -25°C~80°C)
2.2 Mechanical Specification

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-7114 ground plane. For reliable ground connections, use locking washers (star or split) when securing an MXM-7114 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.

2.2.1. 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-7114, as well as key components on the board. All dimensions are in mini-meters.
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-7114 ground plane.
For reliable ground connections, use locking washers (star or split) when securing an MXM-7114 in a carrier board. Make sure that the washers do not extend beyond the limits of the pads provided (5.5mm). A M3, F head, 4mm long, 5mm in diameter, and 1mm head thick screw is recommended.

2.2.4. Clearances
The MXM-7114 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-7114 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-7114 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.
Following table lists the typical power consumption of each MXM series computer on module. All I/Os are up under the testing environment.

<table>
<thead>
<tr>
<th></th>
<th>MXM-7110</th>
<th>MXM-7114*</th>
</tr>
</thead>
<tbody>
<tr>
<td>Power Consumption</td>
<td>250mA/5V</td>
<td>300mA/5V</td>
</tr>
</tbody>
</table>

MXM-7114 User’s Manual
**Embedian, Inc.**

**Note:**
1. The above data is module only and the tested LCD is 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-7114 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-7114 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-7114.
Chapter 3 Quick Start Guide

These quick start guides are intended to provide developers with simple instructions on how to install MXM-7114 from the 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-7114. 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-7114. Users need MXM-7114 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-7114 into the carrier board and tighten it

Figure 3.1 Plug MXM-7114 into the carrier board and tighten it

Plug MXM-7114 in the carrier board of the evaluation kit at 45 degrees and press down. Use a cross-head screwdriver 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. The root file system for MXM-7114 includes Embedian’s own or Debian Etch.
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 of the evaluation kit

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 5V 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 5V 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](image)

The default root password is “apc7110” (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**: 192.168.1.121
- **netmask**: 255.255.255.0
- **gateway**: 192.168.1.254

```
[root@apc7110 ~]# ifconfig eth0
[root@apc7110 ~]# ifconfig eth0
eth0  Link encap:Ethernet  HWaddr 10:0D:32:1F:19:39
inet addr:192.168.1.121  Bcast:192.168.1.255  Mask:255.255.255.0
          Metric:1
```

```
UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
```
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@apc7110 ~]# ifconfig eth0 192.168.1.123 netmask 255.255.255.0 up
[root@apc7110 ~]#
```

**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

MXM-7114 supports dual LCD displays. One is output from CPU LCD (TTL) and the other is output from external graphic chip SM502 (LVDS and DSUB-15). If you need to connect an LCD, use a FPC cable or LVDS cable (depending on the type of LCD) connect to SBC first. Check if JP4 is configured properly. The FPC cable at the board side (CN22) is top-contacted and is from CPU controller. LVDS connector (CN20) is located on the back side of the board and is from SM502 graphic chip. You will need to power the backlight of LCD yourself. MXM-7114 evaluation kit provides users with a 12V bypass path from pin 2 of CN3 to pin 1 of CN2 on carrier board for the backlight connection. Users can also use a standard monitor is attached a DSUB15 cable to CN18 directly.

Figure 3.6 shows the LCD connection.

**Figure 3.6 LCD Connection**

The device descriptor of the LCD is registered as /dev/fb/0 and /dev/fb/1 depending on which driver load first. 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 JP5) 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@apc7110 ~]# modprobe backlight
LCD backlight & panel power control interface for APC-71xx.
```

This is to load the driver that controls the switches.

```
[root@apc7110 ~]# echo "1" >> /proc/panel_power
[root@apc7110 ~]# echo "1" >> /proc/backlight
[root@apc7110 ~]#
```

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 two ways.
You can edit /etc/sysconfig/devices

```
#!/bin/sh

LOADDRV_APM=YES
LOADDRV_APCI2E=YES
LOADDRV_USB_HOST=YES
LOADDRV_USB_DISK=YES
# LOADDRV_SOC_LCD=YES
# LOADDRV_SM501FB=YES
LOADDRV_SM502FB=YES
# SM501FB_OUTPUT=VGA
# SM501FB_OUTPUT=LCD
LOADDRV_AC97=YES
LOADDRV_TOUCHSCREEN=YES
```

By setting the LOADDRV_SM502FB or LOADDRV_SOC_LCD equals to YES depending which drivers that you would like to load.
Or simply load the drive by the following command.

```
[root@apc7110 ~]# modprobe sm502fb
Found SM502 chip revision c0
DRAM_CTRL = 0cb021c0
[root@apc7110 ~]#
```

Or you can also load the SOC LCD driver.

```
[root@apc7110 ~]# modprobe s3c2410fb
fb0: s3c2410fb frame buffer device
```
Users also need to use `fbset` command to set up the frame buffer first. For example,

```
[root@apc7110 ~]# fbset 1024x768-60
```

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.

User can also attach a DSUB-15 monitor to CN18 directly.

Figure 3.7 shows the DSUB-15 monitor connection.

*Figure 3.7 DSUB-15 VGA Connection*

The DSB15 VGA frame buffer output is the same as LVDS output (CN20). To use DSUB-15 VGA output, users can follow exactly the same as using LCD except users do not need to modprobe the backlight module and turn the GPIO switch on.
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-7114. CN1 and CN2 mentioned on this chapter means the connector on MXM-7114 module.

4.1 Connector Type
The MXM-7114 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)*

---

*Embedian, Inc.*

33
**Figure 4.2 CN2 Header Type** (Connector: *DF12(5.0)-60DP-0.5V (**) (Hirose) or compatible*)

![Figure 4.2 CN2 Header Type](image1)

**Figure 4.3 CN2 Socket Type** (Mating Connector : *DF12(5.0)-60DP-0.5V.***)

![Figure 4.3 CN2 Socket Type](image2)

### 4.2 Connector Mechanical Drawing

The detail connector mechanical drawing is as follows.
Figure 4.4 CN1 Socket Connector Mechanical Drawing
**Figure 4.5 CN2 Header Connector Mechanical Drawing**

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>17.2</td>
<td>14.5</td>
<td>15.7</td>
<td>16.6</td>
<td>2.3</td>
<td>18.0</td>
</tr>
</tbody>
</table>
**Figure 4.6 CN2 Socket Connector Mechanical Drawing**

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>17.1</td>
<td>14.5</td>
<td>15.6</td>
<td>16.6</td>
<td>4.2</td>
<td>18.0</td>
</tr>
</tbody>
</table>

**4.3 Connector Location**

MXM series computer on module use 242-pin MXM form factor golden finger connectors CN1 and a 60-pin DF12(5.0)-60DP-0.5V (**) (Hirose) or compatible connector CN2 as an interface to connect with carrier board. The CN2 is mainly for SM502 related. For other manufactures that are intend to use the MXM form factor, please add your companion chip pin out to CN2 as well.
Figure 4.3 Connector Location I
4.3.1. Connector Pin Assignments

The following tables describe the electrical signals available on the connectors of the MXM-7114. 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-7114, refer to section 2.2.2. for mechanical drawing.

**Legend:**

<table>
<thead>
<tr>
<th>NC</th>
<th>Not Connected</th>
</tr>
</thead>
<tbody>
<tr>
<td>RSVD</td>
<td>Reserved for future platform, suggest open at current design</td>
</tr>
<tr>
<td>GND</td>
<td>MXM-7114 Ground Plane</td>
</tr>
</tbody>
</table>

**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
### 4.3.1.1. 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>
<thead>
<tr>
<th>Description</th>
<th>Mating Connector : B33P102-0013 (Speed Tech), AS0B326-S78N-7F (Foxconn) or compatible</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Header</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Pin</strong></td>
<td><strong>Signal Name</strong></td>
</tr>
<tr>
<td>4-wire Touch Screen</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>7</td>
</tr>
<tr>
<td>ADC Input</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>13</td>
</tr>
<tr>
<td></td>
<td>15</td>
</tr>
<tr>
<td>Reserved Pin</td>
<td>17</td>
</tr>
<tr>
<td></td>
<td>19</td>
</tr>
<tr>
<td></td>
<td>21</td>
</tr>
<tr>
<td></td>
<td>23</td>
</tr>
<tr>
<td>Key</td>
<td>37</td>
</tr>
<tr>
<td>Key</td>
<td>39</td>
</tr>
<tr>
<td>DMA</td>
<td>41</td>
</tr>
<tr>
<td></td>
<td>43</td>
</tr>
<tr>
<td>Address Bus</td>
<td>45</td>
</tr>
<tr>
<td></td>
<td>47</td>
</tr>
<tr>
<td></td>
<td>49</td>
</tr>
<tr>
<td></td>
<td>51</td>
</tr>
<tr>
<td>Address Bus</td>
<td>Address Bus</td>
</tr>
<tr>
<td>-------------</td>
<td>-------------</td>
</tr>
<tr>
<td>53</td>
<td>ADADDR4</td>
</tr>
<tr>
<td>55</td>
<td>ADADDR5</td>
</tr>
<tr>
<td>57</td>
<td>ADADDR6</td>
</tr>
<tr>
<td>59</td>
<td>ADADDR7</td>
</tr>
<tr>
<td>61</td>
<td>ADADDR8</td>
</tr>
<tr>
<td>63</td>
<td>ADADDR9</td>
</tr>
<tr>
<td>65</td>
<td>ADADDR10</td>
</tr>
<tr>
<td>67</td>
<td>ADADDR11</td>
</tr>
<tr>
<td>69</td>
<td>ADADDR12</td>
</tr>
<tr>
<td>71</td>
<td>ADADDR13</td>
</tr>
<tr>
<td>73</td>
<td>ADADDR14</td>
</tr>
<tr>
<td>75</td>
<td>ADADDR15</td>
</tr>
<tr>
<td>77</td>
<td>GND</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Address Bus</th>
</tr>
</thead>
<tbody>
<tr>
<td>79</td>
</tr>
<tr>
<td>81</td>
</tr>
<tr>
<td>83</td>
</tr>
<tr>
<td>85</td>
</tr>
<tr>
<td>87</td>
</tr>
<tr>
<td>89</td>
</tr>
<tr>
<td>91</td>
</tr>
<tr>
<td>93</td>
</tr>
<tr>
<td>95</td>
</tr>
<tr>
<td>97</td>
</tr>
<tr>
<td>99</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Reserved Pin</th>
</tr>
</thead>
<tbody>
<tr>
<td>101</td>
</tr>
<tr>
<td>103</td>
</tr>
<tr>
<td>105</td>
</tr>
<tr>
<td>107</td>
</tr>
<tr>
<td>109</td>
</tr>
<tr>
<td>111</td>
</tr>
<tr>
<td>113</td>
</tr>
<tr>
<td>115</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chip Select</th>
</tr>
</thead>
<tbody>
<tr>
<td>117</td>
</tr>
<tr>
<td>119</td>
</tr>
<tr>
<td>121</td>
</tr>
<tr>
<td>123</td>
</tr>
<tr>
<td>125</td>
</tr>
<tr>
<td>127</td>
</tr>
<tr>
<td>129</td>
</tr>
<tr>
<td>131</td>
</tr>
<tr>
<td>133</td>
</tr>
<tr>
<td>Port</td>
</tr>
<tr>
<td>------</td>
</tr>
<tr>
<td>135</td>
</tr>
<tr>
<td>137</td>
</tr>
<tr>
<td>139</td>
</tr>
<tr>
<td>141</td>
</tr>
<tr>
<td>143</td>
</tr>
<tr>
<td>145</td>
</tr>
<tr>
<td>147</td>
</tr>
<tr>
<td>149</td>
</tr>
<tr>
<td>151</td>
</tr>
<tr>
<td>153</td>
</tr>
<tr>
<td>155</td>
</tr>
<tr>
<td>157</td>
</tr>
<tr>
<td>159</td>
</tr>
<tr>
<td>161</td>
</tr>
<tr>
<td>163</td>
</tr>
<tr>
<td>165</td>
</tr>
<tr>
<td>167</td>
</tr>
<tr>
<td>169</td>
</tr>
<tr>
<td>171</td>
</tr>
<tr>
<td>173</td>
</tr>
<tr>
<td>175</td>
</tr>
<tr>
<td>177</td>
</tr>
<tr>
<td>179</td>
</tr>
<tr>
<td>181</td>
</tr>
<tr>
<td>183</td>
</tr>
<tr>
<td>185</td>
</tr>
<tr>
<td>187</td>
</tr>
<tr>
<td>189</td>
</tr>
<tr>
<td>191</td>
</tr>
<tr>
<td>193</td>
</tr>
<tr>
<td>195</td>
</tr>
<tr>
<td>197</td>
</tr>
<tr>
<td>199</td>
</tr>
<tr>
<td>201</td>
</tr>
<tr>
<td>203</td>
</tr>
<tr>
<td>205</td>
</tr>
<tr>
<td>207</td>
</tr>
<tr>
<td>209</td>
</tr>
</tbody>
</table>
## Embedian, Inc.

<table>
<thead>
<tr>
<th></th>
<th>RSVD</th>
<th>N.C.</th>
</tr>
</thead>
<tbody>
<tr>
<td>211</td>
<td>RSVD</td>
<td>N.C.</td>
</tr>
<tr>
<td>213</td>
<td>RSVD</td>
<td>N.C.</td>
</tr>
<tr>
<td>215</td>
<td>RSVD</td>
<td>N.C.</td>
</tr>
<tr>
<td>217</td>
<td>RSVD</td>
<td>N.C.</td>
</tr>
<tr>
<td>219</td>
<td>RSVD</td>
<td>N.C.</td>
</tr>
<tr>
<td>221</td>
<td>RSVD</td>
<td>N.C.</td>
</tr>
<tr>
<td>223</td>
<td>RSVD</td>
<td>N.C.</td>
</tr>
<tr>
<td>225</td>
<td>RSVD</td>
<td>N.C.</td>
</tr>
<tr>
<td>227</td>
<td>RSVD</td>
<td>N.C.</td>
</tr>
<tr>
<td>229</td>
<td>RSVD</td>
<td>N.C.</td>
</tr>
<tr>
<td>231</td>
<td>RSVD</td>
<td>N.C.</td>
</tr>
<tr>
<td>233</td>
<td>RSVD</td>
<td>N.C.</td>
</tr>
<tr>
<td>235</td>
<td>RSVD</td>
<td>N.C.</td>
</tr>
<tr>
<td>237</td>
<td>RSVD</td>
<td>N.C.</td>
</tr>
<tr>
<td>239</td>
<td>RSVD</td>
<td>N.C.</td>
</tr>
<tr>
<td>241</td>
<td>RSVD</td>
<td>N.C.</td>
</tr>
</tbody>
</table>
### Table 4.2: CN1 Connector (Top Side)

#### Description

Mating Connector: B33P102-0013 (Speed Tech), A5OB326-S78N-7F (Foxconn) or compatible

<table>
<thead>
<tr>
<th>Header</th>
<th>Pin</th>
<th>Signal Name</th>
<th>Function</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Reserved Pin</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>RSVD</td>
<td>Reserved</td>
<td>N.C.</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>RSVD</td>
<td></td>
<td>N.C.</td>
<td></td>
</tr>
<tr>
<td><strong>JTAG</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>TMS</td>
<td>TAP Controller Mode Select</td>
<td>I</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>TDO</td>
<td>TAP Controller Data Output</td>
<td>O</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>TDI</td>
<td>TAP Controller Data Input</td>
<td>I</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>TCK</td>
<td>TAP Controller Clock</td>
<td>I</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>nTRST</td>
<td>TAP Controller Reset</td>
<td>I</td>
<td></td>
</tr>
<tr>
<td><strong>AC97</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>AC_SYNC</td>
<td>48kHz fixed rate sample sync</td>
<td>O</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>AC_BIT_CLK</td>
<td>12.288MHz serial data clock</td>
<td>I/O</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>AC_nRESET</td>
<td>AC’97 Master H/W Reset</td>
<td>O</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>AC_SDATA_IN</td>
<td>AC’97 input stream</td>
<td>I</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>AC_SDATA_OUT</td>
<td>AC’97 output stream</td>
<td>O</td>
<td></td>
</tr>
<tr>
<td><strong>Power Input</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>38</td>
<td>EXT5V</td>
<td>DC in 5V</td>
<td>P</td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>EXT5V</td>
<td>DC in 5V</td>
<td>P</td>
<td></td>
</tr>
<tr>
<td>42</td>
<td>EXT5V</td>
<td>DC in 5V</td>
<td>P</td>
<td></td>
</tr>
<tr>
<td>44</td>
<td>EXT5V</td>
<td>DC in 5V</td>
<td>P</td>
<td></td>
</tr>
<tr>
<td><strong>Reserved</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>46</td>
<td>RSVD</td>
<td>Reserved</td>
<td>N.C.</td>
<td></td>
</tr>
<tr>
<td><strong>CPU LCD</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>48</td>
<td>VD19</td>
<td>LCD data bus RED0 (LSB)</td>
<td>O</td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>VD20</td>
<td>LCD data bus RED1</td>
<td>O</td>
<td></td>
</tr>
<tr>
<td><strong>52</strong></td>
<td><strong>VD21</strong></td>
<td>LCD data bus</td>
<td>RED2</td>
<td>O</td>
</tr>
<tr>
<td>-------</td>
<td>---------</td>
<td>--------------</td>
<td>------</td>
<td>---</td>
</tr>
<tr>
<td><strong>54</strong></td>
<td><strong>VD22</strong></td>
<td>LCD data bus</td>
<td>RED3</td>
<td>O</td>
</tr>
<tr>
<td><strong>56</strong></td>
<td><strong>VD23</strong></td>
<td>LCD data bus</td>
<td>RED4 (MSB)</td>
<td>O</td>
</tr>
<tr>
<td><strong>58</strong></td>
<td><strong>VD10</strong></td>
<td>LCD data bus</td>
<td>GREEN0 (LSB)</td>
<td>O</td>
</tr>
<tr>
<td><strong>60</strong></td>
<td><strong>VD11</strong></td>
<td>LCD data bus</td>
<td>GREEN1</td>
<td>O</td>
</tr>
<tr>
<td><strong>62</strong></td>
<td><strong>VD12</strong></td>
<td>LCD data bus</td>
<td>GREEN2</td>
<td>O</td>
</tr>
<tr>
<td><strong>64</strong></td>
<td><strong>VD13</strong></td>
<td>LCD data bus</td>
<td>GREEN3</td>
<td>O</td>
</tr>
<tr>
<td><strong>66</strong></td>
<td><strong>VD14</strong></td>
<td>LCD data bus</td>
<td>GREEN4</td>
<td>O</td>
</tr>
<tr>
<td><strong>68</strong></td>
<td><strong>VD15</strong></td>
<td>LCD data bus</td>
<td>GREEN5 (MSB)</td>
<td>O</td>
</tr>
</tbody>
</table>

Reserved
<table>
<thead>
<tr>
<th><strong>70</strong></th>
<th><strong>RSVD</strong></th>
<th>Reserved</th>
<th>N.C.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>72</strong></td>
<td><strong>VD3</strong></td>
<td>LCD data bus</td>
<td>BLUE0 (LSB)</td>
</tr>
<tr>
<td><strong>74</strong></td>
<td><strong>VD4</strong></td>
<td>LCD data bus</td>
<td>BLUE1</td>
</tr>
<tr>
<td><strong>76</strong></td>
<td><strong>VD5</strong></td>
<td>LCD data bus</td>
<td>BLUE2</td>
</tr>
<tr>
<td><strong>78</strong></td>
<td><strong>VD6</strong></td>
<td>LCD data bus</td>
<td>BLUE3</td>
</tr>
<tr>
<td><strong>80</strong></td>
<td><strong>VD7</strong></td>
<td>LCD data bus</td>
<td>BLUE4 (MSB)</td>
</tr>
<tr>
<td><strong>82</strong></td>
<td><strong>VCLK</strong></td>
<td>LCD clock signal</td>
<td>O</td>
</tr>
<tr>
<td><strong>84</strong></td>
<td><strong>HSYNC</strong></td>
<td>Horizontal synchronous signal</td>
<td>O</td>
</tr>
<tr>
<td><strong>86</strong></td>
<td><strong>VSYNC</strong></td>
<td>Vertical synchronous signal</td>
<td>O</td>
</tr>
<tr>
<td><strong>88</strong></td>
<td><strong>VDEN</strong></td>
<td>Data enable signal</td>
<td>O</td>
</tr>
<tr>
<td><strong>90</strong></td>
<td><strong>GND</strong></td>
<td>Ground Power</td>
<td>P</td>
</tr>
</tbody>
</table>

**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 |

**SPI**
<p>| <strong>104</strong> | <strong>SPIMISO0</strong> | Master mode: data input; | I/O |</p>
<table>
<thead>
<tr>
<th>I/O Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>106</td>
<td>Slave mode: data output; Master mode: data output; Slave mode: data input</td>
</tr>
<tr>
<td>108</td>
<td>SPI Clock</td>
</tr>
<tr>
<td>110</td>
<td>SPI Chip Select</td>
</tr>
<tr>
<td>112</td>
<td>Slave mode: data input; Master mode: data output</td>
</tr>
<tr>
<td>114</td>
<td>Master mode: data output; Slave mode: data input</td>
</tr>
<tr>
<td>116</td>
<td>SPI Clock</td>
</tr>
<tr>
<td>118</td>
<td>SPI Chip Select</td>
</tr>
<tr>
<td><strong>Interrupt</strong></td>
<td></td>
</tr>
<tr>
<td>120</td>
<td>EXT_INT1</td>
</tr>
<tr>
<td>122</td>
<td>EXT_INT2</td>
</tr>
<tr>
<td>124</td>
<td>EXT_INT3</td>
</tr>
<tr>
<td>126</td>
<td>EXT_INT4</td>
</tr>
<tr>
<td>128</td>
<td>EXT_INT5</td>
</tr>
<tr>
<td>130</td>
<td>EXT_INT6</td>
</tr>
<tr>
<td>132</td>
<td>EXT_INT7</td>
</tr>
<tr>
<td>134</td>
<td>EXT_INT8</td>
</tr>
<tr>
<td>136</td>
<td>GND</td>
</tr>
<tr>
<td><strong>GPIO</strong></td>
<td></td>
</tr>
<tr>
<td>138</td>
<td>GPIO1</td>
</tr>
<tr>
<td>140</td>
<td>GPIO2</td>
</tr>
<tr>
<td>142</td>
<td>GPIO3</td>
</tr>
<tr>
<td>144</td>
<td>GPIO4</td>
</tr>
<tr>
<td>146</td>
<td>GPIO5</td>
</tr>
<tr>
<td>148</td>
<td>GPIO6</td>
</tr>
<tr>
<td>150</td>
<td>GPIO7</td>
</tr>
<tr>
<td>152</td>
<td>GPIO8</td>
</tr>
<tr>
<td>154</td>
<td>GPIO9</td>
</tr>
<tr>
<td>156</td>
<td>GPIO10</td>
</tr>
<tr>
<td>158</td>
<td>GPIO11</td>
</tr>
<tr>
<td>160</td>
<td>GPIO12</td>
</tr>
<tr>
<td>162</td>
<td>VCCIO_PWREN</td>
</tr>
<tr>
<td>164</td>
<td>VCCLCD_PWREN</td>
</tr>
<tr>
<td>166</td>
<td>BACKLIGHT_EN</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>168</td>
<td>LCD_PWREN</td>
</tr>
<tr>
<td>170</td>
<td>BBAT</td>
</tr>
<tr>
<td><strong>SD Card (*)</strong></td>
<td><strong>SD Card (*)</strong></td>
</tr>
<tr>
<td>172</td>
<td>SD_nCD</td>
</tr>
<tr>
<td>174</td>
<td>SD_WP</td>
</tr>
<tr>
<td>176</td>
<td>SDCLK</td>
</tr>
<tr>
<td>178</td>
<td>SDCMD</td>
</tr>
<tr>
<td>180</td>
<td>SDDAT0</td>
</tr>
<tr>
<td>182</td>
<td>SDDAT1</td>
</tr>
<tr>
<td>184</td>
<td>SDDAT2</td>
</tr>
<tr>
<td>186</td>
<td>SDDAT3</td>
</tr>
<tr>
<td>188</td>
<td>GND</td>
</tr>
<tr>
<td><strong>Reserved</strong></td>
<td><strong>Reserved</strong></td>
</tr>
<tr>
<td>190</td>
<td>RSVD</td>
</tr>
<tr>
<td>192</td>
<td>RSVD</td>
</tr>
<tr>
<td>194</td>
<td>RSVD</td>
</tr>
<tr>
<td>196</td>
<td>RSVD</td>
</tr>
<tr>
<td>198</td>
<td>RSVD</td>
</tr>
<tr>
<td>200</td>
<td>RSVD</td>
</tr>
<tr>
<td>202</td>
<td>RSVD</td>
</tr>
<tr>
<td>204</td>
<td>RSVD</td>
</tr>
<tr>
<td><strong>UART</strong></td>
<td><strong>UART</strong></td>
</tr>
<tr>
<td>206</td>
<td>RXD2</td>
</tr>
<tr>
<td>208</td>
<td>TXD2</td>
</tr>
<tr>
<td>210</td>
<td>nCTS1</td>
</tr>
<tr>
<td>212</td>
<td>nRTS1</td>
</tr>
<tr>
<td>214</td>
<td>RXD1</td>
</tr>
<tr>
<td>216</td>
<td>TXD1</td>
</tr>
<tr>
<td>218</td>
<td>nCTS0</td>
</tr>
<tr>
<td>Port</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>------------------------------</td>
</tr>
<tr>
<td>220</td>
<td>nRTS0 UART request to send output signal</td>
</tr>
<tr>
<td>222</td>
<td>RXD0 UART receives data input</td>
</tr>
<tr>
<td>224</td>
<td>TXD0 UART transmits data output</td>
</tr>
<tr>
<td></td>
<td><strong>Ethernet</strong></td>
</tr>
<tr>
<td>226</td>
<td>LANLED1 Ethernet Speed LED</td>
</tr>
<tr>
<td>228</td>
<td>LANLED2 Ethernet Link LED</td>
</tr>
<tr>
<td>230</td>
<td>AVDD18 1.8V For Transformer</td>
</tr>
<tr>
<td>232</td>
<td>TX- Ethernet Transmits data-</td>
</tr>
<tr>
<td>234</td>
<td>TX+ Ethernet Transmits data+</td>
</tr>
<tr>
<td>236</td>
<td>AGND Ethernet Ground</td>
</tr>
<tr>
<td>238</td>
<td>RX- Ethernet Receives data-</td>
</tr>
<tr>
<td>240</td>
<td>RX+ Ethernet Receives data+</td>
</tr>
<tr>
<td>242</td>
<td>AVDD18 1.8V For Transformer</td>
</tr>
</tbody>
</table>

(*)At this moment, no Linux driver support.
4.3.1.2. CN2 Connector
All SM502 related interface is from CN2.
The following table shows the pin outs of CN2 connector.

**Table 4.3 CN2 Connector**

<table>
<thead>
<tr>
<th>Description</th>
<th>Connector: DF12(5.0)-60DP-0.5V (**)(Hirose) or compatible Mating Connector: DF12(5.0)-60DP-0.5V</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Header</strong></td>
<td><strong>Pin</strong></td>
</tr>
<tr>
<td>2</td>
<td>SM502_VCLK</td>
</tr>
<tr>
<td>4</td>
<td>SM502_HSYNC</td>
</tr>
<tr>
<td>6</td>
<td>SM502_VSYNC</td>
</tr>
<tr>
<td>8</td>
<td>SM502_VDEN</td>
</tr>
<tr>
<td>10</td>
<td>SM502_LCDPWREN</td>
</tr>
<tr>
<td>12</td>
<td>GND</td>
</tr>
<tr>
<td>14</td>
<td>SM502_VGA_B</td>
</tr>
<tr>
<td>16</td>
<td>SM502_VGA_G</td>
</tr>
<tr>
<td>18</td>
<td>SM502_VGA_R</td>
</tr>
<tr>
<td>20</td>
<td>SM502_VGA_HSYNC</td>
</tr>
<tr>
<td>22</td>
<td>SM502_VGA_VSYNC</td>
</tr>
<tr>
<td>24</td>
<td>GND</td>
</tr>
<tr>
<td>26</td>
<td>GND</td>
</tr>
<tr>
<td><strong>ZV Port (</strong>)**</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>VP_VSYNC</td>
</tr>
<tr>
<td>30</td>
<td>VP_HREF</td>
</tr>
<tr>
<td>32</td>
<td>VPCLK</td>
</tr>
</tbody>
</table>
### Video Data Input

<table>
<thead>
<tr>
<th>Pin</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>34</td>
<td>ZV0</td>
<td></td>
</tr>
<tr>
<td>36</td>
<td>ZV1</td>
<td></td>
</tr>
<tr>
<td>38</td>
<td>ZV2</td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>ZV3</td>
<td></td>
</tr>
<tr>
<td>42</td>
<td>ZV4</td>
<td></td>
</tr>
<tr>
<td>44</td>
<td>ZV5</td>
<td></td>
</tr>
<tr>
<td>46</td>
<td>ZV6</td>
<td></td>
</tr>
<tr>
<td>48</td>
<td>ZV7</td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>RSVD</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

### SM502 UART 0

<table>
<thead>
<tr>
<th>Pin</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>52</td>
<td>SM502_TXD0</td>
<td>UART transmits data output</td>
</tr>
<tr>
<td>54</td>
<td>SM502_RXD0</td>
<td>UART receives data input</td>
</tr>
<tr>
<td>56</td>
<td>SM502_nCTS0</td>
<td>UART clear to send input signal</td>
</tr>
<tr>
<td>58</td>
<td>SM502_nRTS0</td>
<td>UART request to send output signal</td>
</tr>
</tbody>
</table>

### Power Input

<table>
<thead>
<tr>
<th>Pin</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>60</td>
<td>EXT5V</td>
<td>DC5V Input</td>
</tr>
</tbody>
</table>

### SM502 LCD Data Bus

<table>
<thead>
<tr>
<th>Pin</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SM502_LCDDATA0</td>
<td>LCD data bus BLUE0 (LSB)</td>
</tr>
<tr>
<td>3</td>
<td>SM502_LCDDATA1</td>
<td>LCD data bus BLUE1</td>
</tr>
<tr>
<td>5</td>
<td>SM502_LCDDATA2</td>
<td>LCD data bus BLUE2</td>
</tr>
<tr>
<td>7</td>
<td>SM502_LCDDATA3</td>
<td>LCD data bus BLUE3</td>
</tr>
<tr>
<td>9</td>
<td>SM502_LCDDATA4</td>
<td>LCD data bus BLUE4</td>
</tr>
<tr>
<td>11</td>
<td>SM502_LCDDATA5</td>
<td>LCD data bus BLUE5</td>
</tr>
<tr>
<td>13</td>
<td>SM502_LCDDATA6</td>
<td>LCD data bus BLUE6</td>
</tr>
<tr>
<td>15</td>
<td>SM502_LCDDATA7</td>
<td>LCD data bus BLUE7 (MSB)</td>
</tr>
<tr>
<td>17</td>
<td>SM502_LCDDATA8</td>
<td>LCD data bus GREEN0 (LSB)</td>
</tr>
<tr>
<td>19</td>
<td>SM502_LCDDATA9</td>
<td>LCD data bus GREEN1</td>
</tr>
<tr>
<td>21</td>
<td>SM502_LCDDATA10</td>
<td>LCD data bus GREEN2</td>
</tr>
<tr>
<td>23</td>
<td>SM502_LCDDATA11</td>
<td>LCD data bus GREEN3</td>
</tr>
<tr>
<td>25</td>
<td>SM502_LCDDATA12</td>
<td>LCD data bus</td>
</tr>
</tbody>
</table>

*O* indicates input, *I* indicates output, *P* indicates power input.
<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>27</td>
<td>SM502_LCDDATA13</td>
<td>LCD data bus</td>
<td>GREEN4</td>
</tr>
<tr>
<td>29</td>
<td>SM502_LCDDATA14</td>
<td>LCD data bus</td>
<td>GREEN5</td>
</tr>
<tr>
<td>31</td>
<td>SM502_LCDDATA15</td>
<td>LCD data bus</td>
<td>GREEN6</td>
</tr>
<tr>
<td>33</td>
<td>SM502_LCDDATA16</td>
<td>LCD data bus</td>
<td>GREEN7 (MSB)</td>
</tr>
<tr>
<td>35</td>
<td>SM502_LCDDATA17</td>
<td>LCD data bus</td>
<td>RED0 (LSB)</td>
</tr>
<tr>
<td>37</td>
<td>SM502_LCDDATA18</td>
<td>LCD data bus</td>
<td>RED1</td>
</tr>
<tr>
<td>39</td>
<td>SM502_LCDDATA19</td>
<td>LCD data bus</td>
<td>RED2</td>
</tr>
<tr>
<td>41</td>
<td>SM502_LCDDATA20</td>
<td>LCD data bus</td>
<td>RED3</td>
</tr>
<tr>
<td>43</td>
<td>SM502_LCDDATA21</td>
<td>LCD data bus</td>
<td>RED4</td>
</tr>
<tr>
<td>45</td>
<td>SM502_LCDDATA22</td>
<td>LCD data bus</td>
<td>RED5</td>
</tr>
<tr>
<td>47</td>
<td>SM502_LCDDATA23</td>
<td>LCD data bus</td>
<td>RED6</td>
</tr>
<tr>
<td>49</td>
<td>RSVD</td>
<td>Reserved</td>
<td>N.C.</td>
</tr>
</tbody>
</table>

**SM502 UART 1**

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>51</td>
<td>SM502_TXD1</td>
<td>UART transmits data output</td>
<td>O</td>
</tr>
<tr>
<td>53</td>
<td>SM502_RXD1</td>
<td>UART receives data input</td>
<td>I</td>
</tr>
<tr>
<td>55</td>
<td>SM502_nCTS1</td>
<td>UART clear to send input signal</td>
<td>I</td>
</tr>
<tr>
<td>57</td>
<td>SM502_nRTS1</td>
<td>UART request to send output signal</td>
<td>O</td>
</tr>
<tr>
<td>59</td>
<td>EXT5V</td>
<td>DC5V Input</td>
<td>P</td>
</tr>
</tbody>
</table>

(**) At this moment, no driver support yet.
Software References

This Chapter details how to use the Embedian Linux of MXM-7114 on 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
- Debian Etch
Chapter 5 Software References

MXM-7114 can use Embedian’s default root filesystem or Debian Etch as the root file system. This chapter gives an introduction in regarding to use Linux on MXM-7114 computer on modules. This guide is mainly focus on the topic related to Embedian’s products and not intends to provide with a Linux guide. The Linux on MXM-7114 is pretty much the same as that in Desktop. For Debian Etch users, there are so many references material available and we only give brief guide in the last part of this chapter. This guide mainly uses MXM-7114 on the evaluation kit as an example.

5.1 Booting

When power on, the uboot will initialize the low-level hardware and bring the Linux kernel to SDRAM. 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.)

<table>
<thead>
<tr>
<th>Root Filesystem Selection</th>
</tr>
</thead>
<tbody>
<tr>
<td>1) CF</td>
</tr>
<tr>
<td>2) HD</td>
</tr>
<tr>
<td>3) NAND</td>
</tr>
</tbody>
</table>

Please enter your choice:
Load root filesystem from CF Disk Partition 1

The NAND file system is in cramfs 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 \texttt{apc7110}.

5.2.1 Create a User
To add a user, you can use \texttt{useradd} command.

5.2.2 Set User Password
After create a user, you can use \texttt{passwd} command to set the password.

\textbf{Example:}
Below is an example to create a user \textit{john} with home directory and set his password.

\begin{verbatim}
[root@apc7110 ~]# useradd -m john
[root@apc7110 ~]# 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:
Bad password: too simple.
Warning: weak password (enter it again to use it anyway).
New password:
Re-enter new password:
Password changed.
[root@apc7110 ~]#
\end{verbatim}
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@apc7110 ~]# userdel -r john
[root@apc7110 ~]#
```

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

**IP address 192.168.1.121**

**netmask 255.255.255.0**

**gateway 192.168.1.254**

```
[root@apc7110 ~]# ifconfig eth0
eth0 Link encap:Ethernet HWaddr 10:0D:32:1F:19:39
  inet addr:192.168.1.121 Bcast:192.168.1.255
  Mask:255.255.255.0
  UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
  RX packets:1546 errors:0 dropped:0 overruns:0 frame:0
  TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
  collisions:0 txqueuelen:1000
  RX bytes:140271 (136.9 Kb) TX bytes:0 (0.0 b)
  Interrupt:17 Base address:0x2300
[root@apc7110 ~]#
```

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@apc7110 ~]# ifconfig eth0 192.168.1.123 netmask 255.255.255.0 up
[root@apc7110 ~]#
```

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 five COM ports in MXM-7114. Three of them are from CPU internal and two of them are from SM502 graphic chip. Additional 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-7114 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/tts/1
- CN10 → /dev/tts/0

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@apc7110 ~]# modprobe 8250
Serial: 8250/16550 driver $Revision$ 4 ports, IRQ sharing disabled
TL16C752B serial driver for APC7110
serial8250: ttyS0 at MMIO 0x18000000 (irq = 50) is a ST16654
serial8250: ttyS1 at MMIO 0x18000040 (irq = 51) is a ST16654
serial8250: ttyS2 at MMIO 0x18000080 (irq = 52) is a ST16654
serial8250: ttyS3 at MMIO 0x180000c0 (irq = 53) is a ST16654
[root@apc7110 ~]#
```

- CN9 → /dev/ttS/0 (pin 1~9) and /dev/ttS/1 (pin 11~19)
- CN8 → /dev/ttS/2 (pin 1~9) and /dev/ttS/3 (pin 11~19)

**Note:**
1. Users can add init script in /etc/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 registered as /dev/fb/0 and /dev/fb/1 depending on which driver load first. For Embedded 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 JP5) and backlight power (pin1 of CN2 on carrier board) 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@apc7110 ~]# modprobe backlight
```

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

```
[root@apc7110 ~]# echo "1" >> /proc/panel_power
[root@apc7110 ~]# echo "1" >> /proc/backlight
[root@apc7110 ~]#
```

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 lcd module first by the following two ways.

You can edit `/etc/sysconfig/devices`.

```
#!/bin/sh

LOADDRV_APM=Yes
LOADDRV_APCIDE=Yes
LOADDRV_USB_HOST=Yes
LOADDRV_USB_DISK=Yes
# LOADDRV_SOC_LCD=Yes
# LOADDRV_SM501FB=Yes
LOADDRV_SM502FB=Yes
# SM501FB_OUTPUT=VGA
# SM501FB_OUTPUT=LCD
LOADDRV_AC97=Yes
LOADDRV_TOUCHSCREEN=Yes
```

By setting the `LOADDRV_SM502FB` or `LOADDRV_SOC_LCD` equals to `YES` depending which drivers that you would like to load.

Or simply load the drive by the following command.

```
[root@apc7110 ~]# modprobe sm502fb
```
Found SM502 chip revision c0
DRAM_CTRL = 0cb021c0

[root@apc7110 ~]#

Or you can also load the SOC LCD driver.

[root@apc7110 ~]# modprobe s3c2410fb
fb0: s3c2410fb frame buffer device

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

[root@apc7110 ~]# fbset 1024x768-60

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. Both LVDS and VGA will be output if users uses SM502.

5.6 GPIO
GPIO driver is default a kernel driver module and you need to modprobe it first.

[root@apc7110 ~]# modprobe gpioctl
GPIO (J0-J11) control interface for APC-71xx.
[root@apc7110 ~]#

Below is the sample code for GPIO.

/
* This program demostrates the control of APC-7xxx 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
#include
#include
#include
#include
#include
#include

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-7114 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.

Note: If user uses Debian Etch, you can use apt-get install to install a new
Embedian, Inc.

software package.
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 *lftp* FTP client, assuming the remote host IP address is 59.124.115.43 and the user is *eric*.

```
[root@apc7110 /]# lftp -u eric 59.124.115.43
Password:
lftp eric@59.124.115.43:~> bye
[root@apc7110 /]#
```

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.

```bash  
# date MMDDhhmmYY  
```

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

```bash  
# 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.

```
login as: ubuntu
ubuntu@192.168.1.121's password:
Linux ubuntu 2.6.24-2 #66 PREEMPT Mon Jun 29 23:49:24 CDT 2009 armv61

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To access official Ubuntu documentation, please visit:
http://help.ubuntu.com/
Last login: Tue Jul  7 16:27:44 2009 from 192.168.1.100
ubuntu@ubuntu:~$  
```
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 (cramfs) 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. To access `linuxrc`, press "3" (3 NAND) during boot up process and enter the NAND flash as follows.

```
You can see the `linuxrc` file and edit it.
```

The NAND file system is a cramfs and is a read-only file system. User can also the way in section 5.11.2 to modify `linuxrc`.
5.11.2 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 and integrate them into a cramfs NAND image and update your new firmware into the device 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 integrate your program into a cramfs NAND image and update your new firmware into the device and make a test. Below tells you how to make a cramfs filesystem in a Linux PC.

5.11.1.1 How to make a cramfs NAND filesystem
This guide has to be done in a Linux PC.
The file name of the nandfs is called initrd.img. The first step is to mount initrd.img and make a tarball and un-mount it.

```
root@dns3:/# mkdir tmpdir
root@dns3:/# mount initrd.img tmpdir -o loop
root@dns3:/# cd tmpdir
root@dns3:/# tar -cvf ../nandfs.tar ./
root@dns3:/# cd ../
root@dns3:/# umount tmpdir
```

The second step is to extract the tarball and modify your codes.

```
root@dns3:/# mkdir rootfs
root@dns3:/# tar -xvf rootfs.tar -C rootfs
```

The last step is to make a new nandfs again. The filename here is initrd_new.img.

```
root@dns3:/# mkcramfs nandfs initrd_new.img
Directory data: 59728 bytes
Everything: 13216 kilobytes
Super block: 76 bytes
CRC: d8abf360
root@dns3:/#
```

You will see the new nandfs image initrd_new.img and update your new firmware into your device.

5.11.3 Use CF card as a mass storage
This section will tell you how to use CF card as a mass storage in NAND filesystems.
1. Boot into NAND flash first by pressing 3) NAND during booting process.
2. Prepare for a CF card.
3. Load the CF driver first.

```
[root@APC7100 /]# modprobe apc-ide
hda: TRANSCEND, ATA DISK drive
ide0 at 0xc4c00020-0xc4c00027,0xc4c0001c on irq 48
hda: max request size: 128 KiB
hda: 7831152 sectors (4009 MB) w/1 KiB Cache, CHS=7769/16/63
hda: hda
APC-7100 IDE driver. Found IDE device at 08000000 irq 48
[root@APC7100 /]
```

4. 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. The default is in EXT2 file system.

```
[root@APC7100 /]# mkfs /dev/hda1
mke2fs 1.37 (21-Mar-2005)
ext2fs_check_if_mount: No such file or directory while determining whether /dev/hda1 is mounted.
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
Writing superblocks and filesystem accounting information: done
This filesystem will be automatically checked every 34 mounts or 180 days, whichever comes first. Use tune2fs -c or -i to override.
[root@APC7100 /]
```

5. Mount the partition 1 of the CF card as /mnt and cd to /mnt directory
6. Now, you can use CF card as your mass storage.

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 arm-linux-gcc-3.4.1. The file name is arm-linux-gcc-3.4.1.tar.bz2 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 jxvf arm-linux-gcc-3.4.1.tar.bz2
```

The above command will generate the 3.4.1 folder under the same directory as you made the commands. Move this folder to /usr/local/arm directory.

```
# mv 3.4.1 /usr/local/arm/
# export PATH=$PATH:/usr/local/arm/3.4.1/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/3.4.1/bin/arm-linux-
```
5.12.2 Build Uboot
1. Extract uboot.tar.gz file.
2. # .make clean
3. # make smdk2440_config
4. # make
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
2. # tar xvfj linux-2.6.18.1.tar.bz2
3. # cd linux-2.6.18.1
4. # gzip -d -c ../patch-apc7100-090521-1047.gz | patch -p0
5. # make apc7100_defconfig

5.13 Debian Etch
In addition to Embedian’ default root filesystem, Embedian also provides with Debian Etch as the root file system. To use Debian Etch as the root filesystem, users can extract the Debian tarball into the CF card as that describes in Chapter 6 “Back and Restore Root File Systems”. The file name is “debian_20110622.tar.gz”: After done, plug the CF card into the device and you can see the following boot log.

Starting system log daemon: syslogd.
Starting kernel log daemon: klogd.
Starting portmap daemon...Already running.
Starting MTA: exim4.
Starting internet superserver: inetd.
Starting OpenBSD Secure Shell server: sshd.
Starting NFS common utilities: statd.
Starting deferred execution scheduler: atd
Starting periodic command scheduler: crond.

Debian GNU/Linux 4.0 debian console

debian login:

The default root password is also apc7110.
Users might need to modify the Debian source lists at /etc/apc/source.list as following screen.
# deb http://ftp.uwsg.indiana.edu/linux/debian/ etch main
# deb http://ftp.uwsg.indiana.edu/linux/debian/ etch main
# deb-src http://ftp.uwsg.indiana.edu/linux/debian/ etch main
# deb http://security.debian.org/ etch/updates main
# deb-src http://security.debian.org/ etch/updates main

deb http://archive.debian.org/debian/ etch contrib main non-free

After done, users can use `apt-get install` to install a new package. It is a Debian
and users can just follow the standard way to develop your program. The
driver descriptors are the same and users need to `modprobe` it if the driver is
written as a kernel module.
Chapter 6

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-7114 evaluation kit. Section include:

- Backup the root file systems in CF card
- Restore the root file systems in CF card
- Debian Etch
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. This chapter uses MXM-7114 on the evaluation kit as an example.

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
```

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.

You can ls the file structure.

Next, tar the file system into a file. (The file name in this example is 7110rootfs_backup.tar.gz)

You have backup the CF root file systems as file name “7110rootfs_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 command to list your disk information and find the device descriptor of your CF USB reader.

```bash
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                  1018  3913161   83  Linux
```

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

MXM-7114 User’s Manual
one partition already by default.)

```bash
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.)
```

```bash
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.

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
rootfs_2.0.0.tar.gz  lost+found
root@dns3:/mnt#
```

```
Extract the root filesystem into /mnt directory.

root@dns3:/mnt# tar xvfz rootfs_2.0.0.tar.gz

You can ls the file structure now.

```
root@dns3:/mnt# ls
rootfs_2.0.0.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.

```
MXM-7114 User’s Manual
```
root@dns3:/mnt# rm -f rootfs_2.0.0.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-7114 evaluation kit and boot. You are done!

6.3 Debian Etch

In addition to Embedian’s default rootfs, Debian Etch is also provided from Embedian. To use Debian Etch, users can follow exactly the same steps as mentioned above and use the file “debian_20110622.tar.gz” from Embedian.
A general description of the Printed Circuit Board (PCB) for MXM computer on module carrier boards is provided in this section.
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>
<thead>
<tr>
<th>Dielectric Thickness (mil)</th>
<th>Layer Type</th>
<th>Layer</th>
<th>Signal-End Signals Width (mil)</th>
<th>Impedance (ohm)</th>
<th>Differential Signals Width (mil)</th>
<th>Impedance (ohm)</th>
<th>USB Differential Signals Width (mil)</th>
<th>Impedance (ohm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.7</td>
<td>Prepreg</td>
<td>L1</td>
<td>Signals 6/6</td>
<td>55+/-.10%</td>
<td>6/7/6</td>
<td>100+/-.10%</td>
<td>6/5/6</td>
<td>90+/-.10%</td>
</tr>
<tr>
<td>5</td>
<td>Prepreg</td>
<td>L2</td>
<td>Ground</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.4</td>
<td>Core</td>
<td>L3</td>
<td>Power</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Prepreg</td>
<td>L4</td>
<td>Signals 6/6</td>
<td>55+/-.10%</td>
<td>6/7/6</td>
<td>100+/-.10%</td>
<td>6/5/6</td>
<td>90+/-.10%</td>
</tr>
</tbody>
</table>

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>
<thead>
<tr>
<th>Dielectric Thickness (mil)</th>
<th>Layer</th>
<th>Layer Type</th>
<th>Signal-End (mil)</th>
<th>Impedance (ohm)</th>
<th>Differential Width (mil)</th>
<th>Impedance (ohm)</th>
<th>USB Differential Width (mil)</th>
<th>Impedance (ohm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.7</td>
<td>L1</td>
<td>Signals</td>
<td>5/5</td>
<td>55+/-10%</td>
<td>5/6/5</td>
<td>100+/-10%</td>
<td>5/4/5</td>
<td>90+/-10%</td>
</tr>
<tr>
<td>4</td>
<td>Prepreg</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.4</td>
<td>L2</td>
<td>Ground</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Core</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.4</td>
<td>L3</td>
<td>IN1</td>
<td>5/5</td>
<td>55+/-10%</td>
<td>4/8/4</td>
<td>100+/-10%</td>
<td>4/5/4</td>
<td>90+/-10%</td>
</tr>
<tr>
<td>35</td>
<td>Prepreg</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.4</td>
<td>L4</td>
<td>IN2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Core</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.4</td>
<td>L5</td>
<td>Power</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Prepreg</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.7</td>
<td>L6</td>
<td>Signals</td>
<td>5/5</td>
<td>55+/-10%</td>
<td>5/6/5</td>
<td>100+/-10%</td>
<td>5/4/5</td>
<td>90+/-10%</td>
</tr>
</tbody>
</table>

**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>
<thead>
<tr>
<th>Signal Type</th>
<th>Impedance</th>
</tr>
</thead>
<tbody>
<tr>
<td>USB</td>
<td>90ohm +/- 20%</td>
</tr>
<tr>
<td>LAN</td>
<td>100ohm +/- 20%</td>
</tr>
</tbody>
</table>

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.
Carrier Board Design Guidelines

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

<table>
<thead>
<tr>
<th>Signal Name</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
</table>
| OM0          | BootRom Select          | NAND boot: OPEN (internal pull down 10K resistor)  
                | NOR boot: pull up 1K resistor (only support 16bit NOR boot) |
| nWAIT        | nWAIT Requests          | Pull up 100K resistor                             |
| WAKEUP       | WAKEUP Requests         | Internal 4.7K pull down resistor                 |
| nRESET_IN    | Reset S3C2440A          | Pull up 100K resistor                             |
| nRESET_OUT   | Reset External Device   | Pull up 100K resistor                             |
| nGCS[0..5]   | Chip Select             | Pull up 10K resistor                              |

Table 8.2. : JTAG

<table>
<thead>
<tr>
<th>Signal Name</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TMS</td>
<td>TAP Controller Mode Select</td>
<td>Pull up 10K resistor</td>
</tr>
<tr>
<td>TDO</td>
<td>TAP Controller Data Output</td>
<td>- -</td>
</tr>
<tr>
<td>TDI</td>
<td>TAP Controller Data Input</td>
<td>Pull up 10K resistor</td>
</tr>
<tr>
<td>TCK</td>
<td>TAP Controller Clock</td>
<td>Pull up 10K resistor</td>
</tr>
<tr>
<td>nTRST</td>
<td>TAP Controller Reset</td>
<td>Pull up 10K resistor</td>
</tr>
</tbody>
</table>
Table 8.3. : IIC

<table>
<thead>
<tr>
<th>Signal Name</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IICSCL</td>
<td>IIC-bus clock</td>
<td>Pull up 1K resistor</td>
</tr>
<tr>
<td>IICSDA</td>
<td>IIC-bus data</td>
<td>Pull up 1K resistor</td>
</tr>
</tbody>
</table>

Table 8.4. : SD

<table>
<thead>
<tr>
<th>Signal Name</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SD_nCD</td>
<td>SD Insert Detect</td>
<td>Pull up 49.9K resistor</td>
</tr>
<tr>
<td>SD_WP</td>
<td>SD Write Protect</td>
<td>Pull up 49.9K resistor</td>
</tr>
<tr>
<td>SDCLK</td>
<td>SD Clock</td>
<td>- -</td>
</tr>
<tr>
<td>SDCMD</td>
<td>SD receive response/</td>
<td>Pull up 49.9K resistor</td>
</tr>
<tr>
<td></td>
<td>transmit command</td>
<td></td>
</tr>
<tr>
<td>SDDAT0</td>
<td>BootRom Select</td>
<td>Pull up 49.9K resistor</td>
</tr>
<tr>
<td>SDDAT1</td>
<td>SD receive/transmit data</td>
<td>Pull up 49.9K resistor</td>
</tr>
<tr>
<td>SDDAT2</td>
<td>SD receive/transmit data</td>
<td>Pull up 49.9K resistor</td>
</tr>
<tr>
<td>SDDAT3</td>
<td>SD receive/transmit data</td>
<td>Pull up 49.9K resistor</td>
</tr>
</tbody>
</table>

Table 8.5. : Power

<table>
<thead>
<tr>
<th>Signal Name</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXT5V</td>
<td>DC5V Input</td>
<td>DC5V +5%</td>
</tr>
<tr>
<td>BBAT</td>
<td>RTC Battery Power(DC 3V)</td>
<td>DC3V</td>
</tr>
<tr>
<td>GND</td>
<td>Ground Power</td>
<td>All Ground should be tied together, except for AGND</td>
</tr>
<tr>
<td>AVDD18</td>
<td>1.8V For Transformer</td>
<td>DC1.8V output to transformer</td>
</tr>
<tr>
<td>AGND</td>
<td>Analog Ground</td>
<td>Analog ground to transformer</td>
</tr>
</tbody>
</table>

Table 8.6. : USB

<table>
<thead>
<tr>
<th>Signal Name</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>USBH- and</td>
<td>USB Host Data</td>
<td>Differential Pair</td>
</tr>
<tr>
<td>USBH+</td>
<td></td>
<td></td>
</tr>
<tr>
<td>USBD- and</td>
<td>USB Device Data</td>
<td>Differential Pair</td>
</tr>
<tr>
<td>USBD+</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Signal Name</td>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>-----------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>TX- and TX+</td>
<td>Ethernet</td>
<td>Differential Pair</td>
</tr>
<tr>
<td></td>
<td>Transmits data</td>
<td></td>
</tr>
<tr>
<td>RX- and RX+</td>
<td>Ethernet</td>
<td>Differential Pair</td>
</tr>
<tr>
<td></td>
<td>Receives data</td>
<td></td>
</tr>
</tbody>
</table>
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 8.8 shows MXM module USB signals, including pin number, signals, I/O and descriptions.

Table 8.8 USB Signal Description

<table>
<thead>
<tr>
<th>USB Host 0</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>185</td>
<td>USB0H-</td>
<td>USB0 Host Data - I/O</td>
</tr>
<tr>
<td>187</td>
<td>USB0H+</td>
<td>USB0 Host Data + I/O</td>
</tr>
<tr>
<td>USB Host 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>193</td>
<td>USB1H-</td>
<td>USB1 Host Data - I/O</td>
</tr>
<tr>
<td>195</td>
<td>USB1H+</td>
<td>USB1 Host Data + I/O</td>
</tr>
</tbody>
</table>

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

MXM-7114 User’s Manual
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.
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.
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 8.9 shows MXM module AC-Link signals, including pin number, signals, I/O and descriptions.

<table>
<thead>
<tr>
<th>Table 8.9 Audio Signal Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AC</strong></td>
</tr>
<tr>
<td>16</td>
</tr>
<tr>
<td>18</td>
</tr>
<tr>
<td>20</td>
</tr>
<tr>
<td>22</td>
</tr>
<tr>
<td>24</td>
</tr>
</tbody>
</table>

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 8.10 also points out the distribution of CODEC capacitor locations and placement priorities.
Figure 8.9 CODEC Recommended Capacitor Placement

Priority Level of Close Proximity Placement of Filter, Decoupler, Bypass Capacitors
### Table 8.10 Realtek Series CODEC Capacitor Placement Priorities

<table>
<thead>
<tr>
<th>Signal Description</th>
<th>Package Pins</th>
<th>Priority of Close Proximity to CODEC Pin Placement of Filter and Decoupling Capacitors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Digital Supply Voltage, +5DVdd</td>
<td>1, 9</td>
<td>A</td>
</tr>
<tr>
<td>Analog Supply Voltage, +5AVdd</td>
<td>25, 38</td>
<td>A</td>
</tr>
<tr>
<td>Voltage Reference Filter(VREF)</td>
<td>27</td>
<td>B</td>
</tr>
<tr>
<td>Voltage Reference (VREF_OUT)</td>
<td>28</td>
<td>B</td>
</tr>
<tr>
<td>CODEC Filters</td>
<td>29, 30, 31, 32</td>
<td>B</td>
</tr>
<tr>
<td>Analog Signal Inputs(Decouple)</td>
<td>12–24</td>
<td>C</td>
</tr>
</tbody>
</table>
8.4 TTL/LVDS LCD
MXM-7114 equips 16-bit TTL-level LCD signals from CPU internal and 24-bit TTL-level LCD and VGS signals from SM502. Additional transceiver will be needed if users would like to use LVDS panel.

8.4.1. Signal Description
Table 8.11 shows MXM module TTL level LCD signals, including pin number, signals, I/O and descriptions.
Table 8.11 CPU LCD Signal Description

<table>
<thead>
<tr>
<th>CPU LCD</th>
<th>Description</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>48</td>
<td>VD19 LCD data bus RED0 (LSB)</td>
<td>O</td>
</tr>
<tr>
<td>50</td>
<td>VD20 LCD data bus RED1</td>
<td>O</td>
</tr>
<tr>
<td>52</td>
<td>VD21 LCD data bus RED2</td>
<td>O</td>
</tr>
<tr>
<td>54</td>
<td>VD22 LCD data bus RED3</td>
<td>O</td>
</tr>
<tr>
<td>56</td>
<td>VD23 LCD data bus RED4 (MSB)</td>
<td>O</td>
</tr>
<tr>
<td>58</td>
<td>VD10 LCD data bus GREEN0 (LSB)</td>
<td>O</td>
</tr>
<tr>
<td>60</td>
<td>VD11 LCD data bus GREEN1</td>
<td>O</td>
</tr>
<tr>
<td>62</td>
<td>VD12 LCD data bus GREEN2</td>
<td>O</td>
</tr>
<tr>
<td>64</td>
<td>VD13 LCD data bus GREEN3</td>
<td>O</td>
</tr>
<tr>
<td>66</td>
<td>VD14 LCD data bus GREEN4</td>
<td>O</td>
</tr>
<tr>
<td>68</td>
<td>VD15 LCD data bus GREEN5 (MSB)</td>
<td>O</td>
</tr>
<tr>
<td>72</td>
<td>VD3 LCD data bus BLUE0 (LSB)</td>
<td>O</td>
</tr>
<tr>
<td>74</td>
<td>VD4 LCD data bus BLUE1</td>
<td>O</td>
</tr>
<tr>
<td>76</td>
<td>VD5 LCD data bus BLUE2</td>
<td>O</td>
</tr>
<tr>
<td>78</td>
<td>VD6 LCD data bus BLUE3</td>
<td>O</td>
</tr>
<tr>
<td>80</td>
<td>VD7 LCD data bus BLUE4 (MSB)</td>
<td>O</td>
</tr>
<tr>
<td>82</td>
<td>VCLK LCD clock signal</td>
<td>O</td>
</tr>
<tr>
<td>84</td>
<td>HSYNC Horizontal synchronous signal</td>
<td>O</td>
</tr>
<tr>
<td>86</td>
<td>VSYNC Vertical synchronous signal</td>
<td>O</td>
</tr>
<tr>
<td>88</td>
<td>VDEN Data enable signal</td>
<td>O</td>
</tr>
</tbody>
</table>
8.4.2. Design Guidelines
Figure 8-10 shows the TTL LCD connection.

**Figure 8.10 TTL LCD Connection**

All MXM-7114 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.
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 VGA
MXM-7114 module with SM502 enhanced provides analog display signals. There are three kinds of signals -- red, green, and blue -- that send color information to a VGA monitor. These three signals each drive an electron gun that emits electrons which paint one primary color at a point on the monitor screen. Analog levels between 0 (completely dark) and 0.7 V (maximum brightness) on these control lines tell the monitor what intensities of these three primary colors to combine to make the color of a dot (or pixel) on the monitor’ screen.

8.5.1. Signal Description
Table 8.12 shows MXM module VGA signals, including pin number, signals, I/O and descriptions.

Table 8.12 VGA Signal Description

<table>
<thead>
<tr>
<th>Pin</th>
<th>Signal Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>SM502_VGA_B</td>
</tr>
<tr>
<td>16</td>
<td>SM502_VGA_G</td>
</tr>
<tr>
<td>18</td>
<td>SM502_VGA_R</td>
</tr>
<tr>
<td>20</td>
<td>SM502_VGA_HSYNC</td>
</tr>
<tr>
<td>22</td>
<td>SM502_VGA_VSYNC</td>
</tr>
</tbody>
</table>

Note:
The pin numbering here is the CN2 connector on MXM-7114 module instead of golden finger (CN1) of the MXM-7114.
8.5.2. Design Guidelines
Figure 8.14 shows the connections for MXM module VGA signals.

*Figure 8.14 VGA Connection*

8.5.3. Layout Guidelines

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

*Figure 8.15 VGA layout guidelines*
8.5.3.2. 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 8.16 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>
<thead>
<tr>
<th>Ethernet</th>
<th>Signal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>226</td>
<td>LANLED1</td>
<td>Ethernet Speed LED</td>
</tr>
<tr>
<td>228</td>
<td>LANLED2</td>
<td>Ethernet Link LED</td>
</tr>
<tr>
<td>230</td>
<td>AVDD18</td>
<td>1.8V For Transformer</td>
</tr>
<tr>
<td>232</td>
<td>TX-</td>
<td>Ethernet Transmits data-</td>
</tr>
<tr>
<td>234</td>
<td>TX+</td>
<td>Ethernet Transmits data+</td>
</tr>
<tr>
<td>236</td>
<td>AGND</td>
<td>Ethernet Ground</td>
</tr>
<tr>
<td>238</td>
<td>RX-</td>
<td>Ethernet Receives data-</td>
</tr>
<tr>
<td>240</td>
<td>RX+</td>
<td>Ethernet Receives data+</td>
</tr>
<tr>
<td>242</td>
<td>AVDD18</td>
<td>1.8V For Transformer</td>
</tr>
</tbody>
</table>

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 8.17 shows the 10/100M Ethernet Connections.
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 8.18 shows an example.

Figure 8.18 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 8.19 and Figure 8.20) 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 8.19 and Figure 8.20) 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 8.19 Better examples for signal and trace routing

*Do not turn at right angle (90 degree)*
*Turn 45 degree instead*

*better example*
- $d_1 < 2\text{mm}$
- $d_2 > 3\text{mm}$, having AGND as a shield is better
- $d_3 > 5\text{mm}$, having AGND as a shield is better

Figure 8.20 Worse examples for signal and trace routing

*It's worse to turn at right angle (90 degree)*

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

Figure 8.22 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 8.22 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 8.23.)

Figure 8.23 Ground plane separations for MXM module
8.6.3.4. **Power Plane Partitioning**

- The power planes should be approximately illustrated in Figure 8.24. No bead is needed to connect two power planes.
- It should separate analog power planes from noisy digital (logic) power planes.

*Figure 8.24 Power planes partitioning for MXM module*
8.6.3.5. Magnetic Selection Guide
Refer to the following tables 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)

<table>
<thead>
<tr>
<th>Manufacturer</th>
<th>Part Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Foxconn</td>
<td>JFM24011-0101-4F</td>
</tr>
<tr>
<td>YCL</td>
<td>PTC1111-09L1FG</td>
</tr>
</tbody>
</table>

Table 8.15 10/100Mbps Magnetic Sources

<table>
<thead>
<tr>
<th>Manufacturer</th>
<th>Part Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pulse Engineering</td>
<td>PE-68515, H1102</td>
</tr>
<tr>
<td>YCL</td>
<td>PH163112, PH163539</td>
</tr>
<tr>
<td>Halo</td>
<td>TG110-S050N2, TG110-LC50N2</td>
</tr>
<tr>
<td>Bel Fuse</td>
<td>S558-5999-W2</td>
</tr>
<tr>
<td>GTS</td>
<td>FC-618SM</td>
</tr>
<tr>
<td>MACOM</td>
<td>HS9016, HS9024</td>
</tr>
</tbody>
</table>

Table 8.16 Magnetic Specification Requirements

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Values</th>
<th>Units</th>
<th>Test Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tx/Rx turns ratio</td>
<td>1:1</td>
<td>CT/1:1</td>
<td>-</td>
</tr>
<tr>
<td>Inductance</td>
<td>350</td>
<td>μH (Min)</td>
<td>-</td>
</tr>
<tr>
<td>Insertion loss</td>
<td>1.1</td>
<td>DB (Max)</td>
<td>1-100Mhz</td>
</tr>
<tr>
<td>Return loss</td>
<td>-18</td>
<td>DB (Min)</td>
<td>1-30 Mhz</td>
</tr>
<tr>
<td></td>
<td>-14</td>
<td>DB (Min)</td>
<td>30-60 Mhz</td>
</tr>
<tr>
<td></td>
<td>-12</td>
<td>DB (Min)</td>
<td>60-80 Mhz</td>
</tr>
<tr>
<td>Differential to common mode rejection</td>
<td>-40</td>
<td>DM (Min)</td>
<td>1-60 Mhz</td>
</tr>
<tr>
<td></td>
<td>-30</td>
<td>DB (Min)</td>
<td>60-100 Mhz</td>
</tr>
<tr>
<td>Transformer isolation</td>
<td>1500</td>
<td>V</td>
<td>-</td>
</tr>
</tbody>
</table>
Carrier Board
Mechanical Design Guidelines

A detail description of mechanical design guidelines for the MXM computer on module carrier boards is provided in this section.
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 high option that customers choose. In the evaluation kit that Embedian offers, the MXM connector height option is 5mm and the height of support pillars is also 5mm. The screws that Embedian use is M3, F head, 4mm long, 5mm in 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 refer to Chapter 4.

Figure 9.2 CN2 Recommended PCB Mounting Pattern
Figure 9.3 shows the MXM motherboard footprint with CN2 connectors.
Figure 9.3 MXM Motherboard Footprint with CN2 Connector
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>
<thead>
<tr>
<th>Pin</th>
<th>MXM-7110/MXM-7114</th>
<th>MXM-8310</th>
<th>MXM-6410</th>
</tr>
</thead>
<tbody>
<tr>
<td>9~15</td>
<td>ADC Input</td>
<td>N.C.</td>
<td>ADC Input</td>
</tr>
<tr>
<td>41</td>
<td>DMAACK0</td>
<td>N.C.</td>
<td>TBD</td>
</tr>
<tr>
<td>99</td>
<td>Address 26</td>
<td>N.C.</td>
<td>TBD</td>
</tr>
<tr>
<td>101~113</td>
<td>N.C.</td>
<td>PC Card Bus Related</td>
<td>TBD</td>
</tr>
<tr>
<td>179</td>
<td>Boot ROM Select</td>
<td>N.C.</td>
<td>TBD</td>
</tr>
<tr>
<td>183</td>
<td>N.C.</td>
<td>One Wire Bus</td>
<td>TBD</td>
</tr>
<tr>
<td>199~241</td>
<td>N.C.</td>
<td>USB 2.0 Client UTMI Interface</td>
<td>TBD</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Pin</th>
<th>MXM-7110/MXM-7114</th>
<th>MXM-8310</th>
<th>MXM-6410</th>
</tr>
</thead>
<tbody>
<tr>
<td>104~118</td>
<td>SPI Interface</td>
<td>SSP Interface (*)</td>
<td>TBD</td>
</tr>
<tr>
<td>194~224</td>
<td>UART 0,1,2</td>
<td>UART 1,2,3 (**)</td>
<td>TBD</td>
</tr>
</tbody>
</table>

(*) 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 uboot
- Update Firmware from NOR flash
Appendix I MXM-7114 Firmware Update

This Chapter details firmware upgrade for MXM-7114. The firmware in NAND flash includes uboot, kernel zImage, sysconfig and nandfs (initrd) image. First, we will introduce the firmware architecture. Next, we will detail how to update firmware from uboot. In the last section, we will introduce how to do firmware update from NOR flash in case that your uboot has been erased for somehow. This guide mainly uses MXM-7114 on the evaluation kit as an example. Users can use your own carrier board to do the same thing.

A.1. Firmware Architecture

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

![Firmware Architecture of Linux in NAND Flash](image)

The `uboot.bin` starts from NAND address `0x00000000`. The Linux kernel `zImage` starts from NAND address `0x00030000`. `Sysconfig` is a partition that stores system configuration like IP/MAC address and starts from NAND address `0x00200000`. The NAND filesystem (initrd) is a small cramfs 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 **0x00400000**. The NAND address after **0x01100000** is unused and reserved for your own uses. There are 3 partitions by default for this unused area.

Users need a CF card with root file system installed to boot up the complete root file system. The root filesystem in CF card can use Embedian's rootfs or Debian Etch. The will be described at Chapter 6 – Backup and Restore Root Filesystems.

Users can update the firmware under **uboot** or use onboard NOR flash. The onboard NOR flash is designed for the purpose that when your uboot is erased and cannot boot up anymore. The Embedian factory default is firmware pre-installed. Unless necessary, Embedian doesn’t recommend you update firmware (especially uboot) since the system might not boot up anymore if you did the wrong operation. Following tells you how to update firmware from uboot command prompt. The firmwares for MXM-7114 include **uboot.bin**, **zImage**, **sysconfig.img** and **initrd.img**.

Now, you are ready to update firmware from uboot.

### A.2. Update Firmware from uboot

You could use uboot tftp command to download uboot, Linux kernel zImage, sysconfig, and nandfs (initrd). 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 **uboot.bin**, **zImage**, **sysconfig.img** and **initrd.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 uboot command prompt by pressing any key at boot. You will see uboot command prompt like this.
U-Boot 1.0.0 (Dec 4 2008 - 22:56:33)

U-Boot code: 33F80000 -> 33F9A438  BSS: -> 33F9DA8
IRQ Stack: 33fbeda4
FIQ Stack: 33fbfda4
DRAM Configuration:
Bank #0: 30000000 64 MB
NAND: 64 MB
In: serial
Out: serial
Err: serial
Found DM9000 ID:90000a46 !
DM9000 work in 16 bus width
Hit any key to stop autoboot: 0
APC-7100 #

You can set and add the tftp ip address of the device and server by using "setenv" command as below. Following shows the example for setting up the parameters.

APC-7100 # setenv serverip 192.168.1.10
APC-7100 # setenv ipaddr 192.168.1.105
APC-7100 #

The example here uses 192.168.1.10 as the tftp server ip address and 192.168.1.105 as the device ip address.
After done, use saveenv command to save to settings.

APC-7100 # saveenv
Saving Environment to Flash...
NAND Flash writing
Source base address =0x33fbf0
Target start block number=8
Target size (0x4000*n) =0x4000
.........................

APC-7100 #

Make sure that the serverip for Windows PC and ipaddr for client ip of MXM-7114 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 to NAND by TFTP and “nandw” Command**

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

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

```
APC-7100 # tftp 30000000 uboot.bin
Found DM9000 ID:90000a46 !
DM9000 work in 16 bus width
TFTP from server 192.168.1.10; our IP address is 192.168.1.105
Filename `uboot.bin`.
Load address: 0x30000000
Loading: T ################################
done
Bytes transferred = 107576 (1a438 hex)
APC-7100 #
```

Now, **uboot.bin** file has been uploaded to DRAM temporary address from your PC (tftp server). Temporary address is base address of DRAM and default is set to 0x300000000. We can see the file size is 107576 (1a438 hex) bytes from the log. Write the **uboot.bin** image from DRAM to the NAND by using following nandw command.

```
APC-7100 # nandw 0 1a438 30000000
NAND Flash writing
Source base address =0x30000000
Target start block number=0
Target size (0x4000*n) =0x1a438
```

MXM-7114 User’s Manual
The first argument (0) in `nandw` command is the starting block of NAND flash. The second argument (1a438) is the file size in hex. The third argument (30000000) is the temporary address of DRAM. Start NAND address is 0x00000000 (0th block). Image size of uboot has to be below 30000 (HEX) because it cannot exceed the partition size.

**Note:** uboot 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 uboot (uboot.bin).

**zImage**

Next, we would like to show how to transfer and write Linux kernel zImage. The file name is "zImage". Again, we tftp zImage from PC (tftp server) to DRAM first by the following command.

```
APC-7100 # tftp 30000000 zImage
Found DM9000 ID:90000a46 !
DM9000 work in 16 bus width
TFTP from server 192.168.1.10; our IP address is 192.168.1.105
Filename `zImage`.
Load address: 0x30000000
Loading: T#############################################
#################################################
#################################################
#################################################
#################################################
###########################
done
Bytes transferred = 1636484 (18f884 hex)
APC-7100 #
```

Now, zImage file has been uploaded to DRAM temporary address from your PC (tftp server). Temporary address is base address of DRAM and default is set to 0x30000000. We can see the file size is 1636484 (18f884 hex) bytes from the log.

Write the zImage image from DRAM to NAND Flash by using `nandw` command again.

```
APC-7100 # nandw 0xc 18f884 30000000
NAND Flash writing
```
The first argument (0xc) in \textit{nandw} command is the starting block (hex) of NAND flash. The second argument (18f884) is the file size in hex. The third argument (30000000) is the temporary address of DRAM. Starting NAND address in hex is 0x00030000 (0xc\textsuperscript{th} block). Image size of \textit{zImage} has to be below 1D0000 (HEX) because it cannot exceed the partition size.

\textbf{sysconfig}

Third, we would like to show how to transfer and write \textit{sysconfig.img}. The file name is "\textit{sysconfig.img}". Again, we \texttt{tftp} \textit{sysconfig.img} from PC(tftp server) to DRAM first by the following command.

\begin{verbatim}
APC-7100 # tftp 30000000 sysconfig.img
Found DM9000 ID:90000a46 !
DM9000 work in 16 bus width
TFTP from server 192.168.1.10; our IP address is 192.168.1.105
Filename `sysconfig.img`.
Load address: 0x30000000
Loading: T
#################################################################

APC-7100 #
\end{verbatim}
Now, sysconfig.img file has been uploaded to DRAM temporary address from your PC (tftp server). Temporary address is base address of DRAM and default is set to 0x30000000. We can see the file size is 1048576 (100000 hex) bytes from the log.

Write the sysconfig.img image from DRAM to NAND Flash by using nandw command again.

```
APC-7100 # nandw 0x80 100000 30000000
NAND Flash writing
Source base address      =0x30000000
Target start block number=128
Target size  (0x4000*n)  =0x100000
....................................................................
....................................................................
....................................................................
....................................................................
....................................................................
....................................................................
....................................................................
....................................................................
....................................................................
....................................................................
....................................................................
....................................................................
....................................................................
....................................................................
....................................................................
....................................................................
....................................................................
....................................................................
....................................................................
...................
APC-7100 #
```
The first argument (0x80) in nandw command is the starting block (hex) of NAND flash. The second argument (100000) is the file size in hex. The third argument (30000000) is the temporary address of DRAM. Starting NAND address in hex is 0x00200000 (0x80th block). Image size of sysconfig.img has to be below 200000 (HEX) because it cannot exceed the partition size.

**initrd.img**

Last, we would like to show how to transfer and write initrd.img. initrd is a small cramfs filesystem in NAND flash. Again, we tftp zImage from PC(tftp server) to DRAM first by the following command.

```
APC-7100 # tftp 30000000 initrd.img
Found DM9000 ID:90000a46 !
DM9000 work in 16 bus width
TFTP from server 192.168.1.10; our IP address is 192.168.1.105
Filename `initrd.img`
Load address: 0x30000000
Loading: T
#############################################################################
#############################################################################
#############################################################################
#############################################################################
#############################################################################
#############################################################################
#############################################################################
#############################################################################
#############################################################################
#############################################################################
#############################################################################
#############################################################################
#############################################################################
#############################################################################
#############################################################################
#############################################################################
#############################################################################
#############################################################################
#############################################################################
done
Bytes transferred = 13533184 (ce8000 hex)
APC-7100 #
```
Now, initrd.img file has been uploaded to DRAM temporary address from your PC (tftp server). Temporary address is base address of DRAM and default is set to 0x30000000. We can see the file size is 13533184 (ce8000 hex) bytes from the log.

Write the initrd.img image from DRAM to NAND Flash by using nandw command again.

```
APC-7100 # nandw 0x100 ce8000 30000000
NAND Flash writing
Source base address =0x30000000
Target start block number=256
Target size (0x4000*n) =0xce8000
.......................................................................
.......................................................................
.......................................................................
.......................................................................
.......................................................................
.......................................................................
.......................................................................
...............
APC-7100 #
```

The first argument (0x100) in nandw command is the starting block (hex) of NAND flash. The second argument (ce8000) is the file size in hex. The third argument (30000000) is the temporary address of DRAM. Starting NAND address in hex is 0x00400000 (0x100th block). Image size of initrd has to be below D00000 (HEX) because it cannot exceed the partition size.

After done, reset MXM-7114 evaluation kit and the firmware in MXM-7114 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 uboot 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

+-----[configuration]-----+
| Filenames and paths   |
| File transfer protocols|
| Serial port setup     |
| Modem and dialing     |
| Screen and keyboard   |
| Save setup as dfl     |
| Save setup as..       |
| Exit                  |
| Exit from Minicom     |
+------------------------+
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 MXM-7114 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**

![Serial Port Setup](image)
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 setup 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**
A.2.2.2. TFTP server in Linux PC

MXM-7114 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 a 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
```

A.2.2.3. Setup TFTP Server/Client IP Address from Device

Below will be exactly the same as that in Windows environment. Copy the `uboot.bin`, `zImage`, `sysconfig.img` and `initrd.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 local network and go to uboot command prompt by pressing any key at boot. You will see uboot command prompt like this.
You can set and add the `tftp` ip address of the device and server by using the `setenv` command as below. Following shows the example for setting up the parameters.

```
APC-7100 # setenv serverip 192.168.1.10
APC-7100 # setenv ipaddr 192.168.1.105
APC-7100 #
```

The example here uses 192.168.1.10 as the `tftp` server ip address and 192.168.1.105 as the device ip address.

After done, use `saveenv` command to save to settings.

```
APC-7100 # saveenv
Saving Environment to Flash...
NAND Flash writing
Source base address =0x33fbfdba0
Target start block number=8
Target size (0x4000*n) =0x4000

.........................
```

Make sure that the `serverip` for Windows PC and `ipaddr` for client ip of MXM-7114 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 to NAND by TFTP and “nandw” Command

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

#### uboot.bin

The following command shows how to transfer uboot.bin images to DRAM. To update uboot.bin to DRAM:

```plaintext
APC-7100 # tftp 30000000 uboot.bin
Found DM9000 ID:90000a46!
DM9000 work in 16 bus width
TFTP from server 192.168.1.10; our IP address is 192.168.1.105
Filename `uboot.bin`.
Load address: 0x30000000
Loading: T ######################
done
Bytes transferred = 107576 (1a438 hex)
APC-7100 #
```

Now, uboot.bin file has been uploaded to DRAM temporary address from your PC (tftp server). Temporary address is base address of DRAM and default is set to 0x300000000. We can see the file size is 107576 (1a438 hex) bytes from the log.

Write the uboot.bin image from DRAM to the NAND by using following nandw command.

```plaintext
APC-7100 # nandw 0 1a438 30000000
NAND Flash writing
Source base address =0x30000000
Target start block number=0
Target size  (0x4000*n) =0x1a438
.........................
........................
........................
APC-7100 #
```
The first argument (0) in `nandw` command is the _starting block_ of NAND flash. The second argument (1a438) is the _file size in hex_. The third argument (30000000) is the temporary address of DRAM. Start NAND address is 0x00000000 (0th block). Image size of uboot has to be below 20000 (HEX) because it cannot exceed the partition size.

**Note:** uboot 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 uboot (uboot.bin).

**zImage**

Next, we would like to show how to transfer and write Linux kernel zImage. The file name is _"zImage"_. Again, we tftp zImage from PC (tftp server) to DRAM first by the following command.

```
APC-7100 # tftp 30000000 zImage
Found DM9000 ID:90000a46 !
DM9000 work in 16 bus width
TFTP from server 192.168.1.10; our IP address is 192.168.1.105
Filename `zImage`.
Load address: 0x30000000
Loading: T###############################################
#################################################
#################################################
#################################################
#################################################
###########################
done
Bytes transferred = 1636484 (18f884 hex)
APC-7100 #
```

Now, zImage file has been uploaded to DRAM temporary address from your PC (tftp server). Temporary address is base address of DRAM and default is set to 0x30000000. We can see the file size is 1636484 (_18f884_ hex) bytes from the log.

Write the zImage image from DRAM to NAND Flash by using `nandw` command again.

```
APC-7100 # nandw 0xc 18f884 30000000
NAND Flash writing
```
The first argument (0xc) in `nandw` command is the starting block (hex) of NAND flash. The second argument (18f884) is the file size in hex. The third argument (30000000) is the temporary address of DRAM. Starting NAND address in hex is 0x00020000 (0xc\textsuperscript{th} block). Image size of zImage has to be below 1E0000 (HEX) because it cannot exceed the partition size.

**sysconfig**

Third, we would like to show how to transfer and write sysconfig.img. The file name is "sysconfig.img". Again, we tftp sysconfig.img from PC(tftp server) to DRAM first by the following command.

```
APC-7100 # tftp 30000000 sysconfig.img
Found DM9000 ID:90000a46 !
DM9000 work in 16 bus width
TFTP from server 192.168.1.10; our IP address is 192.168.1.105
Filename `sysconfig.img`.
Load address: 0x30000000
Loading: T
#################################################################
```

**Source base address** = 0x30000000  
**Target start block number** = 12  
**Target size** (0x4000*n) = 0x18f884
Now, sysconfig.img file has been uploaded to DRAM temporary address from your PC (tftp server). Temporary address is base address of DRAM and default is set to 0x30000000. We can see the file size is 1048576 (100000 hex) bytes from the log.
Write the sysconfig.img image from DRAM to NAND Flash by using `nandw` command again.

```
APC-7100 # nandw 0x80 100000 30000000
NAND Flash writing
Source base address      =0x30000000
Target start block number=128
Target size  (0x4000*n)  =0x100000
....................................................................
....................................................................
....................................................................
....................................................................
....................................................................
....................................................................
....................................................................
....................................................................
....................................................................
 .................
APC-7100 #
```
The first argument (0x80) in \textit{nandw} command is the starting block (hex) of NAND flash. The second argument (100000) is the file size in hex. The third argument (30000000) is the temporary address of DRAM. Starting NAND address in hex is 0x00200000 (0x80^{th} block). Image size of sysconfig.img has to be below 200000 (HEX) because it cannot exceed the partition size.

\textit{initrd.img}

Last, we would like to show how to transfer and write initrd.img. initrd is a small cramfs filesystem in NAND flash. Again, we tftp zImage from PC(tftp server) to DRAM first by the following command.

\texttt{APC-7100 \# tftp 30000000 initrd.img}
\texttt{Found DM9000 ID:90000a46 !}
\texttt{DM9000 work in 16 bus width}
\texttt{[eth_init]MAC:10: d:32:1f:19:39}
\texttt{TFTP from server 192.168.1.10; our IP address is 192.168.1.105}
\texttt{Filename `initrd.img`.
Load address: 0x30000000}
\texttt{Loading: T}
\begin{verbatim}
#################################################################
#################################################################
#################################################################
#################################################################
#################################################################
#################################################################
#################################################################
#################################################################
#################################################################
#################################################################
#################################################################

done
Bytes transferred = 13533184 (ce8000 hex)
\end{verbatim}

APC-7100 \#
Now, initrd.img file has been uploaded to DRAM temporary address from your PC (tftp server). Temporary address is base address of DRAM and default is set to 0x30000000. We can see the file size is 13533184 (ce8000 hex) bytes from the log.
Write the initrd.img image from DRAM to NAND Flash by using nandw command again.

```
APC-7100 # nandw 0x100 ce8000 30000000
NAND Flash writing
Source base address =0x30000000
Target start block number=256
Target size (0x4000*n) =0xce8000
....................................................................
....................................................................
....................................................................
....................................................................
....................................................................
....................................................................
....................................................................
....................................................................
....................................................................
....................................................................
....................
APC-7100 #
```

The first argument (0x100) in nandw command is the starting block (hex) of NAND flash. The second argument (ce8000) is the file size in hex. The third argument (30000000) is the temporary address of DRAM. Starting NAND address in hex is 0x004000000 (0x100th block). Image size of initrd has to be below D000000 (HEX) because it cannot exceed the partition size.

After done, reset MXM-7114 evaluation kit and the firmware in MXM-7114 will be updated.
A.3 Update Firmware from NOR flash

The previous section mainly tells you how to update firmware from uboot. Embedian designs a way to recover/update firmware using onboard NOR flash in case that your uboot has been erased for somehow.

First, power off the device (MXM-7114 evaluation kit) and set the jumper JP2 to NOR flash boot configuration (shunt 1-2) on the evaluation kit. Install the USB driver and DNW programs from Embedian in your host Windows XP PC. Connect the serial console cable from serial console port (CN6) of MXM-7114 evaluation kit to the COM port of your Windows XP PC. And connect a USB cable from USB device port (CN5) to USB host port of your Windows XP PC.

Open the Hyperterminal program of your Windows PC and set the baud rate as **115200, 8N1**. Power on the evaluation kit and you will see the following screen.

```
========= Embedian, Inc. =========
========== APC-7110 ===========
DIVN_UPLL0
MPLLVal [M:6eh,P:3h,S:1h]
CLKDIVN:7h
FCLK=399.7MHz, DMA mode
USB: IN_ENDPOINT:1 OUT_ENDPOINT:3
FORMAT: +++)
NOTE: 1. Power off/on or press the reset button for 1 sec in order to get a valid USB device address.
2. For additional menu, Press any key.

###### Select Menu ######
[0] Download & Run
[1] Download Only
[2] Test SDRAM
[4] Clear unused area in SDRAM
```
Enter "5" and you will see the following screen.

+---------------------------------------------+
| S3C2440A Firmware-Test ver 0.03 Jan 2004.   |
+---------------------------------------------+
[CPU ID=32440001h]
[Core voltage: 1.30V]
[XTAL = 16.9344MHz]
[FCLK = 399.65MHz]
[HCLK = 133.22MHz]
[PCLK =  66.61MHz]
[UCLK =  48.00MHz]
[rSTATUS2=0x1]
[rSTATUS3=0x0]
[rSTATUS4=0x0]
[rSRCPND=0x7]
[rINTPND=0x0]

0:RTC Test         1:AC97 Test        2:Uart Test        3:DM9000 Test
4:TL16C752B Test   5:NOR Program      6:NAND Program

Select the function to test :
Enter “6” (6:NAND Program) and you will be brought to the following screen.

Select the function to test : 6

Nand test

Nand flash test start.

NFSTAT: 0x85

NAND device: Manufacturer ID: 0xec, Chip ID: 0x76
(Samsung SLC NAND 64MB 3.3V 8-bit)
Small Block NAND. 4096 blocks.
1 block = 32 pages = 16K Bytes

0:Read ID          1:Nand reset       2:Block erase      3:Page read
4:Page write       5:Nand R/W test    6:Check Badblock   7:Nand Block lock
8:Soft Unlock      9:NAND Program

Select(-1 to exit):

It is better to erase first before writing to NAND. The unit here is in decimal. The uboot partition is 12 blocks and here we erase the first partition as an example.

Enter “2” (2: Block erase) and you will see the following screen.

Select(-1 to exit): 2

NAND Block erase
Block # to erase: 12
12-block erased.

0:Read ID          1:Nand reset       2:Block erase      3:Page read
4:Page write       5:Nand R/W test    6:Check Badblock   7:Nand Block lock
8:Soft Unlock      9:NAND Program

Select(-1 to exit):
Enter "9" (9: Nand Program) and you will see the following screen.

Select (-1 to exit): 9

[ NAND Flash writing program]

[ Using USB Download]
USB: IN_ENDPOINT:1 OUT_ENDPOINT:3
FORMAT: +

The download buffer is from 0x30100000~
Data package is discarded first 8 bytes
USB host is not connected yet.
USB host is connected, Waiting a download.

Here the program will detect if the USB cable is connected, if not, the program will tell you to connect the USB cable. Plug the USB cable and USB is waiting for download the u-boot binary.

Now we would like to write uboot binary into NAND flash. First, we need to use DNW program to transmit uboot.bin from your PC to the SDRAM of MXM-7114 through USB. Open the DNW program, click “Configuration” → “Option” tab and you set the Download Address to 0x30100000 as follows and then click “OK”.

Figure A.2 Setup Download Address of DNW program
In “**USB Port**” of DNW program, click “**Transmit**” as following figure.

**Figure A.3 Transmit through USB Port**

A file browser will ask you to transmit the file. Find the file “**uboot.bin**” in your PC and click. Hyperterminal will ask you to enter the input target block number as follows.

```
Now, Downloading [ADDRESS:3010000h,TOTAL:107586]
RECEIVED FILE SIZE: 107586
(716.8KB/S,0.2S)
Now, USBcheckSum calculation
Download O.K.
ID=0xec76
NFCONF = 0x3776

Source size:0h~1a438h

Available target block number: 0~4096
Input target block number:
```

The unit here is in decimal. Enter 0 and you will see the uboot has been write to NAND flash as following screen.
Uboot now has been updated. You can switch JP2 to NAND boot and you will see the uboot booting.

Once the uboot is up, you can use uboot `tftp` to update other firmware as mentioned in previous section.
You can also keep updating zImage, sysconfig.img and initrd.img in this way. Just remember to put the correct input target block number.