

# User's Manual

## **Computer on Module**

COM Ports

Two USB Hosts

LCD

Ethernet

SD/SDHC

H.263/H.264/MPEG4 Codec

**MXM-6410**



## **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

***Embedian, Inc.***

your systems.

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

# Table of Contents

<b>CHAPTER 1 INTRODUCTION.....</b>	<b>10</b>
1.1 MXM COMPUTER ON MODULE FAMILY .....	10
1.3 BLOCK DIAGRAM.....	11
<b>CHAPTER 2 SPECIFICATIONS .....</b>	<b>13</b>
2.1 FUNCTIONAL SPECIFICATIONS.....	13
2.2 MECHANICAL SPECIFICATION .....	20
2.2.1. <i>Dimensions</i> .....	20
2.2.2. <i>Mechanical Drawing</i> .....	20
2.2.3. <i>Mounting Holes</i> .....	23
2.2.4. <i>Clearances</i> .....	23
2.2.5. <i>Weight</i> .....	23
2.3 ELECTRICAL SPECIFICATION .....	23
2.3.1. <i>Supply Voltages</i> .....	23
2.3.2. <i>Supply Voltage Ripple</i> .....	23
2.3.3. <i>Supply Current (Typical)</i> .....	23
2.3.4. <i>Real-Time Clock (RTC) Battery</i> .....	24
2.3.5. <i>CF</i> .....	24
2.3.6. <i>LCD</i> .....	24
2.4 ENVIRONMENTAL SPECIFICATION .....	24
2.4.1. <i>Temperature</i> .....	24
2.4.2. <i>Humidity</i> .....	24
2.5 MTBF.....	24
2.6 EMI/RFI AND ESD PROTECTION .....	24
<b>CHAPTER 3 QUICK START GUIDE .....</b>	<b>27</b>
3.1 FOR LINUX USERS .....	28
<b>CHAPTER 4 HARDWARE REFERENCES .....</b>	<b>38</b>
4.1 CONNECTOR TYPE .....	38
4.2 CONNECTOR MECHANICAL DRAWING .....	39
4.3 CONNECTOR LOCATION .....	42
4.3.1. <i>Connector Pin Assignments</i> .....	44
<b>CHAPTER 5 USING UBUNTU JAUNTY JACKALOPE.....</b>	<b>62</b>
5.1 BOARD SUPPORT PACKAGE (BSP) .....	62
5.1.1. <i>Drivers</i> .....	62
5.1.2. <i>Default Software Packages</i> .....	63
5.1.3. <i>Booting</i> .....	64
5.2 DEFAULT ROOT PASS AND USER .....	66
5.2.1 <i>Create a User</i> .....	66
5.2.2 <i>Set User Password</i> .....	68
5.2.3 <i>Delete a User</i> .....	69
5.3 NETWORK SETTINGS.....	69
5.3.1 <i>Configure Network Configuration at Boot or Network Restart</i> .....	70
5.4 MANUALLY ADD REPOSITORIES.....	74
5.5 INSTALL SOFTWARE PACKAGES.....	75
5.5.1 <i>List of installed software packages</i> .....	75
5.5.2 <i>Description of installed software packages</i> .....	75
5.5.3 <i>List of available software packages</i> .....	75
5.5.4 <i>Searching a software package: apt-cache search</i> .....	76
5.5.5 <i>Properties and information of a software: apt-cache show</i> .....	77
5.5.6 <i>Installing a software: apt-get install</i> .....	79
5.5.7 <i>Removing a software: apt-get remove</i> .....	79
5.5.8 <i>Updating the software list: apt-get update</i> .....	79
5.5.9 <i>Upgrading the software: apt-get upgrade</i> .....	79
5.5.10 <i>Smart software update: apt-get dist-upgrade</i> .....	80
5.6 COM PORT .....	81

5.6.1 From MXM-6410 .....	81
5.6.2 From external TL16C752B .....	81
5.7 LCD .....	82
5.8 GPIO .....	83
5.9 FTP CLIENT .....	87
5.10 FTP SERVER.....	88
5.11 TIME AND RTC.....	89
5.12 TELNET/SSH SERVER .....	89
5.13 VNC SERVER.....	92
5.14 GDM.....	96
5.15 LXDE.....	96
5.16 CALIBRATION AND TOUCH SCREEN.....	98
5.16.1 Install the Calibration Program.....	98
Note:.....	99
5.17 KIOSK MODE .....	99
5.17.1 Boot Ubuntu 9.04 into text mode instead of graphic mode .....	99
5.17.2 GDM Auto Login.....	100
5.17.3 Auto Start a Program under LXDE .....	101
5.18. MPEG4 DECODER FOR MPLAYER AT DEVICE .....	101
5.18.1. Mplayer running on top of GDM and LXDE .....	102
5.18.2. Mplayer Running on top of frambuffer directly.....	103
5.19 NAND ROOT FILE SYSTEM .....	104
5.19.1 linuxrc .....	104
5.19.2 As a rescue file system.....	105
6.19.3 As a small root file system.....	107
5.20 CROSS TOOLCHAIN .....	108
5.20.1 Installing Toolchain.....	108
5.20.2 Build Uboot.....	109
5.20.3 kernel zImage.....	109
<b>CHAPTER 6 BACKUP AND RESTORE THE ROOT FILE SYSTEM IN SD CARD .....</b>	<b>111</b>
6.1 BACKUP THE ROOT FILE SYSTEM IN SD CARD .....	111
6.2 RESTORE THE ROOT FILE SYSTEM IN SD CARD .....	114
Additional Packages .....	117
<b>CHAPTER 7 USING WINDOWS CE 6.0 .....</b>	<b>120</b>
7.1 GENERAL FEATURES .....	120
7.1.1 Board Support Package (BSP).....	120
7.1.2 Drivers .....	121
7.1.3 Services.....	123
7.1.4 Special Features.....	123
7.2 EBOOT .....	125
7.3. WINDOWS CE 6.0 .....	130
7.3.1. Setting the System Time.....	130
7.3.2. Touch Calibration .....	131
7.3.3. File System and Registry Basic .....	132
7.3.4. Networking Basics.....	135
7.3.5. Telnet Server.....	137
7.3.6. FTP Server.....	142
7.3.7. File Server.....	151
7.3.8. Web Server .....	155
7.3.9. Auto RUN .....	163
7.3.10. COM Ports .....	164
7.3.11. Software Installation .....	167
7.4. CONFIGURE LCD PARAMETERS FOR DIFFERENT KINDS OF LCDS.....	169
7.4.1. View Current LCD Parameters .....	169
7.4.2. Choose Default LCD Parameters .....	170
7.4.3. Set LCD Parameters of Different Types .....	171
7.5. CONFIGURE GPIOs, BACKLIGHT_EN, LCD_PWREN AND VDDLCD_PWREN.....	174
7.5.1. Configure GPIOs Setting at EBOOT.....	174

7.5.2. <i>Configure GPIOs Setting at NK</i> .....	176
7.6. LOGO SPLASH SCREEN CUSTOMIZER .....	177
7.6.1. <i>Upload and save splash screen image into devices</i> .....	177
7.7. WMV9 DECODER FOR WINDOWS MEDIA PLAYER .....	180
7.7.1. <i>Encode Video source as WMV9</i> .....	180
7.7.2. <i>WMV9 Decoder for Windows Media Player</i> .....	181
7.8. OPENGLES USER'S MANUAL .....	182
7.8.1. <i>Software Layers</i> .....	182
7.8.2. <i>Usage</i> .....	183
7.8.3. <i>Shader Compile</i> .....	184
7.8.4. <i>Dll location</i> .....	184
7.8.5. <i>Samples</i> .....	184
<b>CHAPTER 8 USE MXM-6410 HARDWARE MFC MULTIMEDIA FUNCTION .....</b>	<b>186</b>
8.1. MFC DEVICE DRIVER'S API .....	186
8.1.1. <i>CreateFile</i> .....	187
8.1.2. <i>DeviceIoControl</i> .....	188
8.1.3. <i>CloseHandle</i> .....	189
8.1.4. <i>Control Codes for DeviceIoControl()</i> .....	190
8.1.5. <i>Data Structure for Passing the IOCTL Arguments</i> .....	193
<b>CHAPTER 9 GENERAL PCB DESIGN RECOMMENDATIONS .....</b>	<b>195</b>
9.1 NOMINAL BOARD STACK-UP .....	195
9.1.1. <i>Four Layer Board Stackup</i> .....	196
9.1.2. <i>Six Layer Board Stackup</i> .....	197
9.2 DIFFERENTIAL IMPEDANCE TARGETS FOR MICROSTRIP ROUTING .....	199
9.3 ALTERNATIVE STACK UPS .....	199
<b>CHAPTER 10 CARRIER BOARD DESIGN GUIDELINES .....</b>	<b>201</b>
10.1 GENERAL CIRCUIT DESIGN GUIDE .....	202
10.1.1. <i>System-Wise</i> .....	202
10.2 UNIVERSAL SERIAL BUS (USB) .....	206
10.2.1. <i>Universal Serial Bus (USB)</i> .....	206
10.2.2. <i>Signal Description</i> .....	206
10.2.3. <i>Design Guidelines</i> .....	206
10.2.3. <i>Layout Guidelines</i> .....	208
10.3 AC-LINK INTERFACE .....	211
10.3.1. <i>Signal Description</i> .....	211
10.3.2. <i>Design Guidelines</i> .....	211
10.3.3. <i>Layout Guidelines</i> .....	213
10.4 TTL/LVDS LCD .....	218
10.4.1. <i>Signal Description</i> .....	218
10.4.2. <i>Design Guidelines</i> .....	221
10.4.3. <i>Layout Guidelines</i> .....	222
10.5 ETHERNET .....	224
10.5.1. <i>SIGNAL DESCRIPTIONS</i> .....	224
10.6.2. <i>DESIGN GUIDELINES</i> .....	224
10.6.2.1. <i>Differential Pairs</i> .....	224
10.6.2.2. <i>Power Considerations and Ethernet LED</i> .....	225
10.6.3. <i>LAYOUT GUIDELINES</i> .....	226
10.6.3.1. <i>Placement, Signal and Trace Routing</i> .....	226
10.6.3.2. <i>MXM Module 10Base-T/100Base-TX Application</i> .....	228
10.6.3.3. <i>Ground Plane Layout</i> .....	229
10.6.3.4. <i>Power Plane Partitioning</i> .....	230
10.6.3.5. <i>Magnetic Selection Guide</i> .....	231
<b>CHAPTER 11 CARRIER BOARD MECHANICAL DESIGN GUIDELINES .....</b>	<b>233</b>
11.1 MXM MOTHERBOARD FOOTPRINT .....	234
<b>CHAPTER 12 PIN DEFINITION DIFFERENCES BETWEEN EMBEDIAN MXM MODULES</b>	

.....	238
(*)SSP INTERFACE IN MXM-8310 CAN BE CONFIGURED AS SPI INTERFACE BY SOFTWARE. ....	239
(**) THE UART NUMBERING JUST FOLLOWED THE RESPECTIVELY S3C2440 AND PXA320 CPU MANUAL. ....	239
<b>APPENDIX I MXM-6410 LINUX FIRMWARE UPDATE.....</b>	<b>241</b>
A.1. FIRMWARE ARCHITECTURE.....	241
A.2. UPDATE FIRMWARE FROM UBOOT .....	242
A.2.1. Windows Environment .....	242
A.2.1.1. Setup TFTP Server/Client IP Address from Device .....	242
A.2.1.2. Transfer and Write Image by TFTP and “nand write” Command.....	243
A.2.2 Linux Environment.....	246
A.2.2.1. Minicom.....	246
A.2.2.2. TFTP server in Linux PC.....	249
A.2.2.3. Setting up an IP address .....	249
A.2.1.2. Transfer and Write Image by TFTP and “nand write” Command.....	251
A.3 UPDATE FIRMWARE FROM NOR FLASH.....	254
<b>APPENDIX II MXM-6410 WINDOWS CE 6.0 FIRMWARE UPDATE .....</b>	<b>267</b>
A.2.1 FIRMWARE ARCHITECTURE IN NAND FLASH.....	267
A.2.2 RESTORE STEPLDR AND EBOOT FROM NOR FLASH.....	269
A.2.3 RESTORE NK IMAGE FROM EBOOT .....	279
A.2.4 RESTORE FIRMWARE IN NAND FLASH FROM SD BOOT.....	285
A.2.4.1. Using PC Tool to Fusing the SD/SDHC card .....	285
A.2.4.2. Using SD boot fusing NAND flash.....	286

# Chapter

# 1

## Introduction

This Chapter gives background information on the MXM-6410

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 Ubuntu 9.04 and Windows CE 6.0 pre-installed, people could easily develop their programs and make it time to market in very short time.

Based on Samsung ARM1176JZF-S core, the MXM-6410 includes 128MB of NAND Flash, and 128MB of MDDR. Additional interfaces includes TFT LCD interface, two USB host interface, Ethernet interface, four RS232 interface, SD interface, H.263/H.264 multimedia codec, graphic 3D engine that can accelerate OpenGL ES 1.1 & 2.0 rendering, AC97 Audio Interface, IIC, PWM and SPI through a 242-pin MXM golden finger connector as interface. It also includes a sophisticated 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.

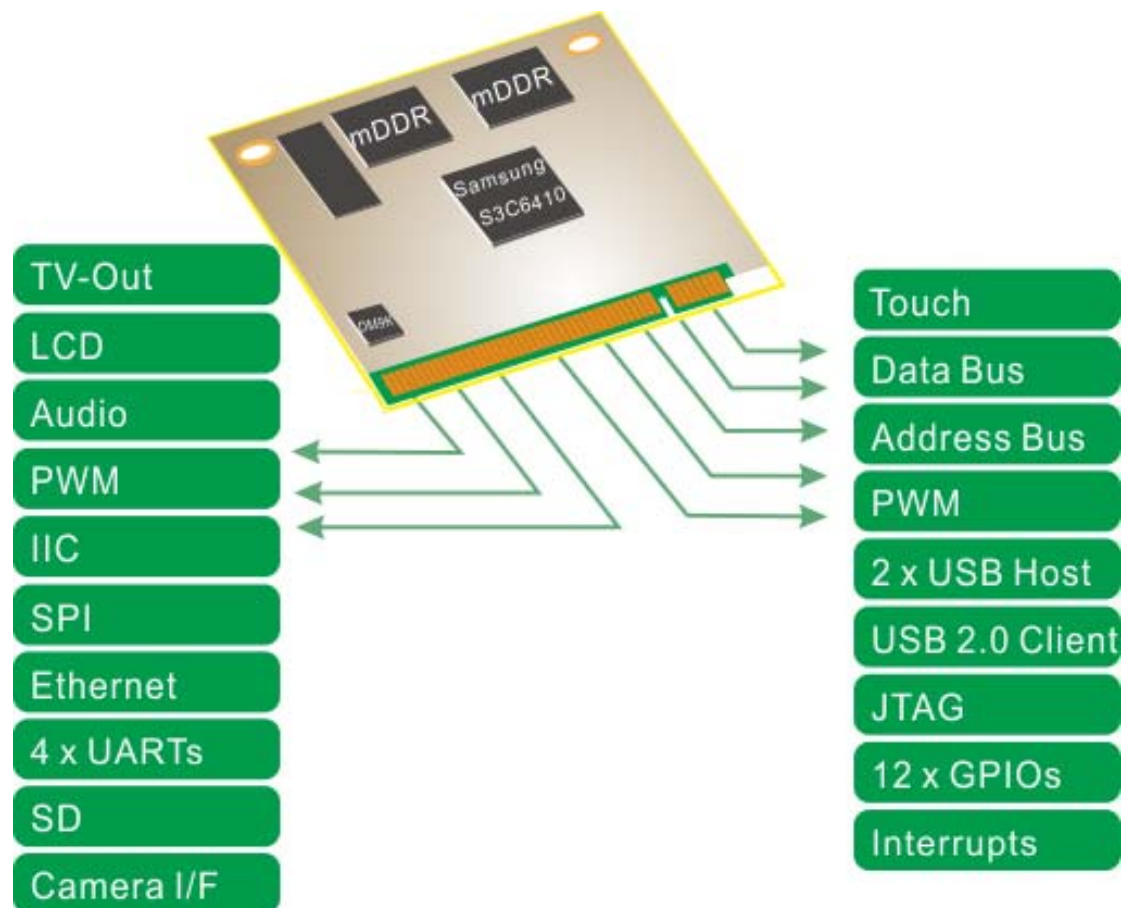
A 242-pin golden finger connector enables the MXM-6410 to interface with the OEM's custom circuitry, and with an evaluation carrier board that is supplied with Embedian's evaluation kit. The evaluation carrier board includes a LCD panel, headers and connectors for all interfaces.

The MXM-6410 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-6410. Arrows indicate direction of control and not necessarily signal flow.

**Figure 1.1 MXM-6410 Block Diagram**



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

# Chapter

# 2

## Specifications

This Chapter contains specifications of MXM-6410.

Section include :

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

## Chapter 2 Specifications

MXM-6410 is a tiny and powerful computer on module in MXM form factor based on the newest Samsung S3C6410 ARM1176JZF-S core processor clocking at 667Mhz that integrated Multi Format Codec (MFC) co-processor supports encoding and decoding of MPEG4/H.263/H.264 and decoding of VC1. This H/W Encoder/Decoder supports real-time video conferencing and TV out for both NTSC and PAL mode. Additional graphic 3D engine is a 3D Graphics Hardware Accelerator which can accelerate OpenGL ES 1.1 & 2.0 rendering. MXM-6410 features state of the art technology, aiming at low power systems that require high CPU performance. They also provide all the interfaces needed in a modern embedded device. It includes TFT LCD interface, two USB host interface, Ethernet interface, four RS232 interface, Camera interface, CF interface, IDE interface, AC97 Audio Interface, IIC, PWM, TV-out and SPI through a 242-pin MXM golden finger connector as interface. MXM-6410 also includes a sophisticated power management mechanism. The small in size makes system integrators and manufacturers flexible in designing their product line with different shapes to make the products fast time to market. It can be applied to the general-purposed embedded application or multimedia-related embedded application by taking advantages of the MFC co-processor. With Linux and Windows CE 6.0 supported, users can easily develop their original application from other platform or develop a new application on this platform. This chapter gives an overall specification for MXM-6410. The specification includes functional, mechanical, electrical, and environmental specifications.

### **2.1 Functional Specifications**

#### **Processor**

- Samsung S3C6410
- ARM1176JZF-S core with Java acceleration engine and 16KB/16KB I/D Cache and 16KB/16KB I/D TCM.
- Clock Rates up to 667Mhz
- 266Mhz 64/32-bit system bus architecture is composed of AXI, AHB and APB buses.
- Multi Format Codec (co-processor) provides encoding and decoding of MPEG-4/H.263/H.264 up to 30fps@SD/D1 and decoding of VC1 video up to 30fps@SD/D1.
- Manufactured using the 65nm process
- 2D graphics acceleration with BitBlit and rotation
- 3D graphics hardware accelerator which can accelerate OpenGL ES 1.1 & 2.0 rendering
- Vector Floating-Point (VFP) coprocessor support allowing efficient implementation of various encryption schemes as well as high quality 3D graphics applications

#### **Power Supply**

- Single input +5V DC power from 242-pin interface
- Power Saving Mode: Nomal, Suspend and Idle

- Real-time clock battery powered

### **Memory**

- Onboard 128MB NAND Flash (Large Block)
- Onboard 128MB mDDR Memory (266Mhz 32-bit Connection, 256MB is available on project based)
- CompactFlash(CF), Type I and Type II, 3.3V, True IDE Mode
- IDE (Shared with CF), ATA Standard support UDMA 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

### **USB Device**

- Chipset: CPU internal
- One USB client 2.0 interface, supporting high speed as Device (480Mbps, on-chip transceiver).
- Compatible with USB specification version 2.0

### **COM Port**

- Chipset: CPU internal
- Four RS232 interface, TTL level (two of them are TX and RX only; and two of them are TX, RX, CTS and RTS)

### **Ethernet**

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

### **CompactFlash(CF) Interface (Shared with IDE)**

- Chipset: CPU CF controller
- Type I and Type II, 3.3V
- Memory mode and True IDE mode
- Compatible with CF+ and CompactFlash Spec (Rev 3.0)

### **SD Card Host Interface**

- Chipset: CPU internal
- SD Memory Card Protocol version 2.0 compatible
- SDIO CARD Protocol version 1.0 compatible
- 128 word FIFO for Tx/Rx
- DMA based or Interrupt based operation
- 3 channel SD/MMC Host Controller

## ***Embedian, Inc.***

- Support CE-ATA Interface

### ***IDE Interface (Shared with CF)***

- Chipset: CPU ATA controller
- compatible with the ATA standard
- Support UDMA mode

### ***CPU Video Graphic Array (VGA)***

- Chipset: CPU LCD controller
- TFT Panel Support
- Up to 800x600 resolutions
- TTL (16-bit/24-bit) interface
- Support 5 Window Layer for PIP or OSD
- Programmable OSC window positioning
- 16-level alpha blending

### ***Video Post Processor***

- Chipset: CPU internal
- Video input format conversion
- Video/Graphic scaling up/down or zooming in/out
- Color space conversion from YCbCr to RGB and from RGB to YCbCr
- Dedicated scaler for TV Encoder

### ***TV Out***

- Chipset: CPU internal
- Video input format conversion
- Video/Graphic scaling up/down or zooming in/out
- Color space conversion from YCbCr to RGB and from RGB to YCbCr
- Dedicated scaler for TV Encoder

### ***AC97 Audio-Codec Interface***

- Chipset: CPU AC-Link/IIS interface
- AC97 version 2.3 compliance interface
- Advanced Linux Sound Architecture (ALSA) API support

### ***Discrete I/O***

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

### ***Multi Format Codec (MFC)***

- Chipset: CPU internal MFC co-processor
- MPEG-4 part-II simple profile encoding/decoding 30fps@SD/D1
- H.264/AVC baseline encoding/decoding 30fps@SD/D1
- H.263 profile3 encoding/decoding 30fps@SD/D1
- VC1 decoding 30fps@SD/D1
- Encoding tools
  - [-16,+16] 1/2 and 1/4 pel accuracy motion estimation using the full-search algorithm
  - Variable block sizes: 16x16, 16x8, 8x16 and 8x8
  - Unrestricted motion vector

- MPEG-4 AC/DC prediction
- H.264/AVC intra-prediction (hardwired mode decision)
- In-loop deblocking filter for both H.264 and H.263 P3
- Error resilience tools
- MPEG-4 resync. Marker and data-partitioning with RVLC
- MPEG-4/AVC FMO
- Bit-rate control (CBR and VBR)
- Decoding tools
  - Support all features of the standards

### **JPEG Codec**

- Chipset: CPU JPEG Codec co-processor
- Compression/decompression up to UXGA size
- Encoding format: YCbCr 4:2:2 / RGB565
- Decoding format: YCbCr 4:4:4/4:2:2/4:2:0 or gray

### **2D Graphic Accelerator**

- Chipset: CPU internal
- Line/Point drawing, BitBLT and Color Expansion /Text Drawing

### **3D Graphic Accelerator**

- Chipset: CPU internal
- 4M triangles/s @133MHz (Transform Only)
- 75.8M pixels/s fill-rates @133MHz (shaded pixels)
- Programmable Shader Model 3.0 support
- 128-bit (32-bit x 4) Floating-point Vertex Shader
- Geometry-texture cache support
- 128-bit (32-bit x 4) Floating-point two Fragment Shaders
- Max. 4K x 4K frame-buffer (16/32-bpp)
- 32-bit depth buffer (8-bit stencil/24-bit Z)
- Texture format: 1/2/4/8/16/32-bpp RGB, YUV 422, S3TC Compressed
- Support max. 8 surfaces (max. 8 user-defined textures)
- API Support: OpenGL ES 1.1 & 2.0, D3D Mobile
- Intelligent Host Interface
  - 15 input data-types, Vertex Buffer & Vertex Cache
- H/W Clipping (Near & Far)
- 8-stage five-threaded Shader architecture
- Primitive assembly & hard-wired triangle setup engine
- One pixels/cycle hard-wired rasterizer
- One texturing engine (one bilinear-filtered texel/cycle each)
- Nearest/bilinear/trilinear filtering
- 8-layered multi-texturing support
- Fragment processing: Alpha/Stencil/Z/Dither/Mask/ROP
- Memory bandwidth optimization through hierarchical caching
  - L1/L2 Texture-caches, Z/Color caches

### **Security Sub-System**

- Chipset: CPU internal
- AES accelerator: ECB, CBC, CTR mode support



## ***Embedian, Inc.***

- DES/3DES accelerator: ECB, CBC mode support
- SHA-1 Hash engine
- H/W HMAC support
- Random Number Generator : PRNG 320-bit generation per 160 cycles
- FIFO-Rx/Tx: (two 32-word) for input and output streaming.
- DMA I/F to SDMA1(Security DMA 1)

### ***Camera Interface***

- Chipset: CPU internal
- ITU-R 601/ITU-R 656 format input support. 8-bit input is supported
- Both progressive and interlaced input are supported
- Camera input resolution up to 4096x4096 in YCbCr 4:2:2 format
  - 4096x4096 input resolution assumes the hardware down-scaling units will be bypass
  - Up to 2048x2048 input resolution can optionally be input to the hardware down-scaling unit
- Resolution down-scaling hardware support for input resolutions up to 2048x2048
- Codec/Preview output image generation (RGB 16/18/24-bit format and YCbCr 4:2:0/4:2:2 format)
- Image windowing and digital zoom-in function
- Image mirror and rotation supports Y-mirror, X-mirror, 90°, 180° and 270° rotation
- H/W Color Space Conversion
- LCD controller direct path supported
- Image effect supported.

### ***Watchdog Timer (WDT)***

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

### ***System Bus***

- Chipset: CPU System Bus
- 16-bit or 8-bit support

### ***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

**Pulse Width Modulation (PWM)**

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

**Touch Panel Interface**

- Chipset: CPU ADC
- 10-bit CMOS ADC

**JTAG**

- Testing and debugging interface

**BIOS**

- Universal Bootloader (u-boot)
- Dual BIOS Support
- Ethernet TFTP download
- Booting from NAND Flash Technology
- Eboot (for Windows CE 6.0)

**Operating System**

- Linux 2.6.24, Ubuntu Linux 9.04 Supports (BIOS, Kernel , rescue root filesystem stored in NAND flash and Rootfs in SD/SDHC card)
- Windows CE 6.0

**Cross Toolchain**

- Based on gcc 4.2.2
- Support EABI
- Support for Cortex-M1 (ARMV6-M) CPUs
- Improved code generation for Cortex-A8 and Cortex-R4 CPUs
- hmb-2 GLIBC binaries

**Dimension**

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

**Packing List**

- 1 x MXM-6410
- Need to purchase the evaluation kit at the very first time

**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-6410 (normal temperature)

***Embedian, Inc.***

- MXM-6410-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-6410 ground plane.

For reliable ground connections, use locking washers (star or split) when securing an MXM-6410 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.

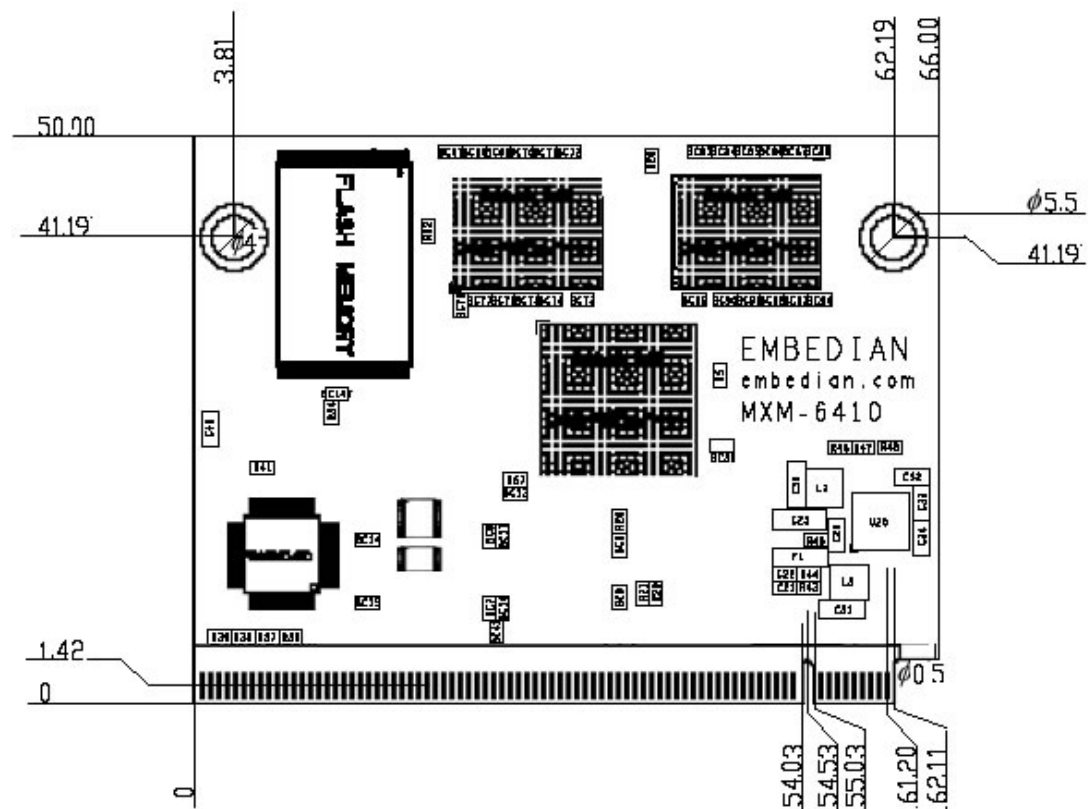
### **2.2.1. Dimensions**

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

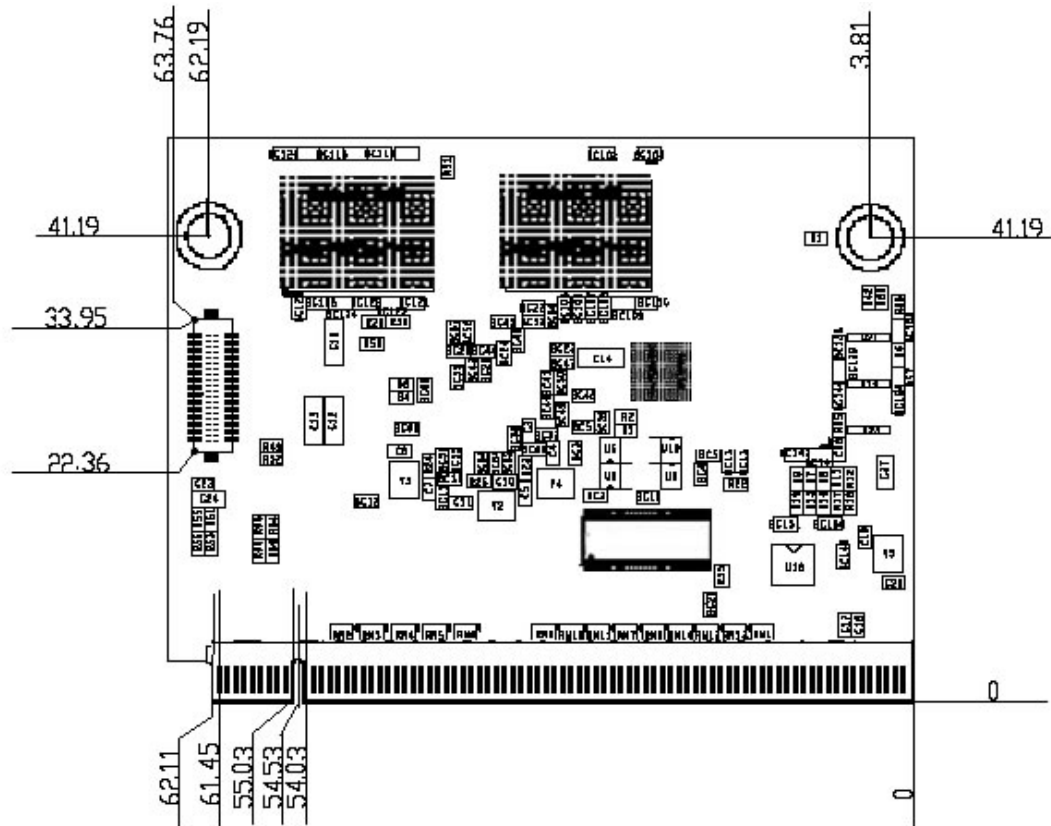
### **2.2.2. Mechanical Drawing**

The following mechanical drawing specifies the dimension of MXM-6410, as well as key components on the board. All dimensions are in mini-meters.

Top View



## Bottom View



### **2.2.3. Mounting Holes**

Two mounting holes are provided for mounting. The diameter of the holes is 4.0 mm. (The diameter of the ring is 5.5mm.) Mounting holes are plated through and connected to the MXM-6410 ground plane.

For reliable ground connections, use locking washers (star or split) when securing an MXM-6410 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-6410 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-6410 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-6410 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 2.1 Power Consumption**

	<b>MXM-6410</b>
<b>Power Consumption</b>	<b>300mA/5V</b>

**Note:**

1. The above data is module only and the tested LCD is 800x480 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-6410 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



***Embedian, Inc.***

suppressors, EMI fences, filters on I/O lines and termination of high-frequency signals are included standard on all systems.

MXM-6410 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.

# Chapter 3

## Quick Start Guide

To save developer's time, this Chapter gives a quick start guide of MXM-6410.

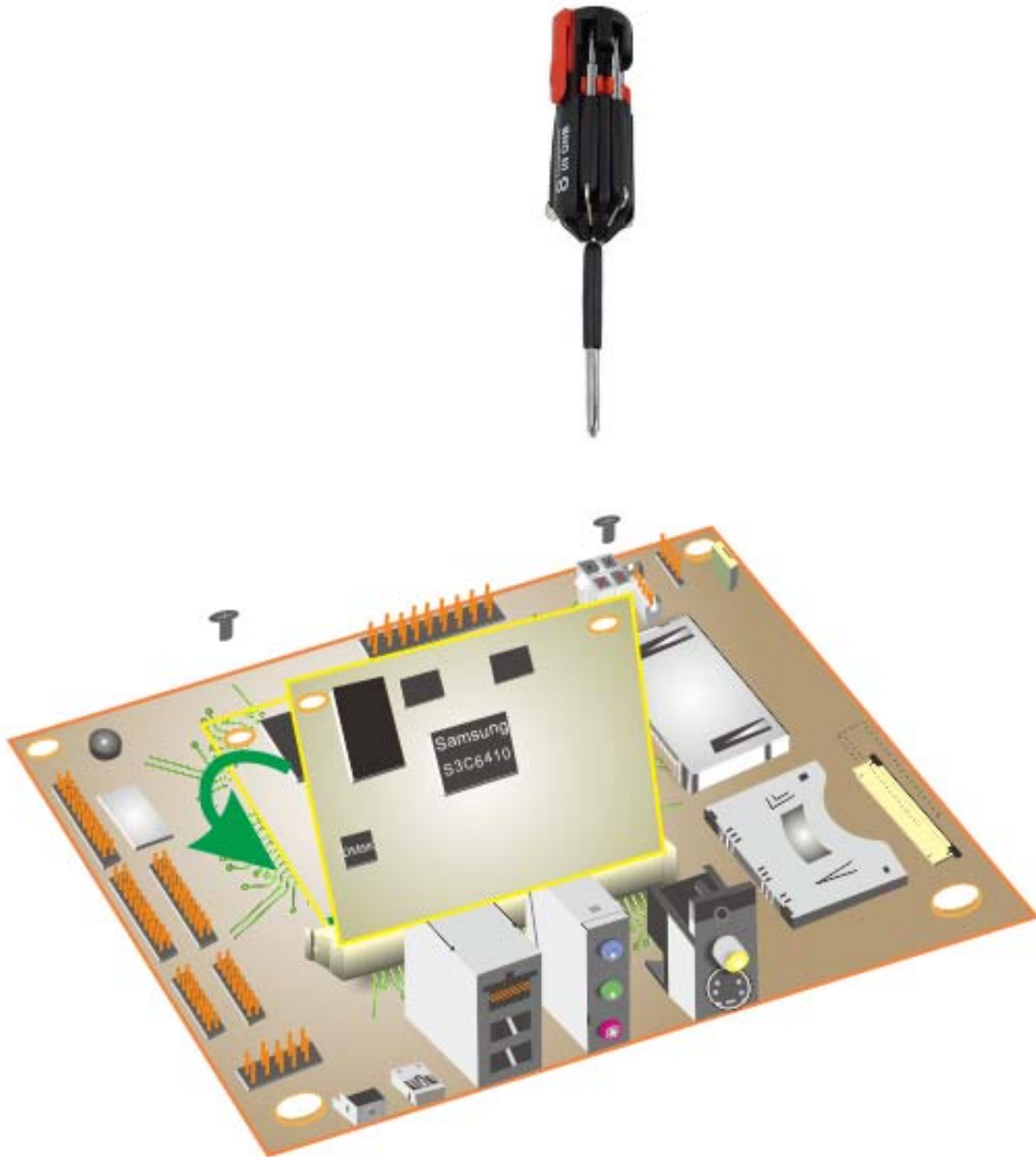
## **Chapter 3 Quick Start Guide**

These quick start guides are intended to provide developers with simple instructions on how to install MXM-6410 from very beginning and have it monitoring your local device inside of 20 minutes. No advanced installation options are discussed here - just the basics that will work for 95% of users who want to get started. This guide will lead you through the process of configuring, installing, and developing MXM-6410. 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-6410. Users need MXM-6410 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. This chapter includes two parts. This first part is Linux users and the second part is for Windows CE 6.0 users.

### 3.1 For Linux Users

**Step 1 : Plug MXM-6410 into the carrier board and tighten it**

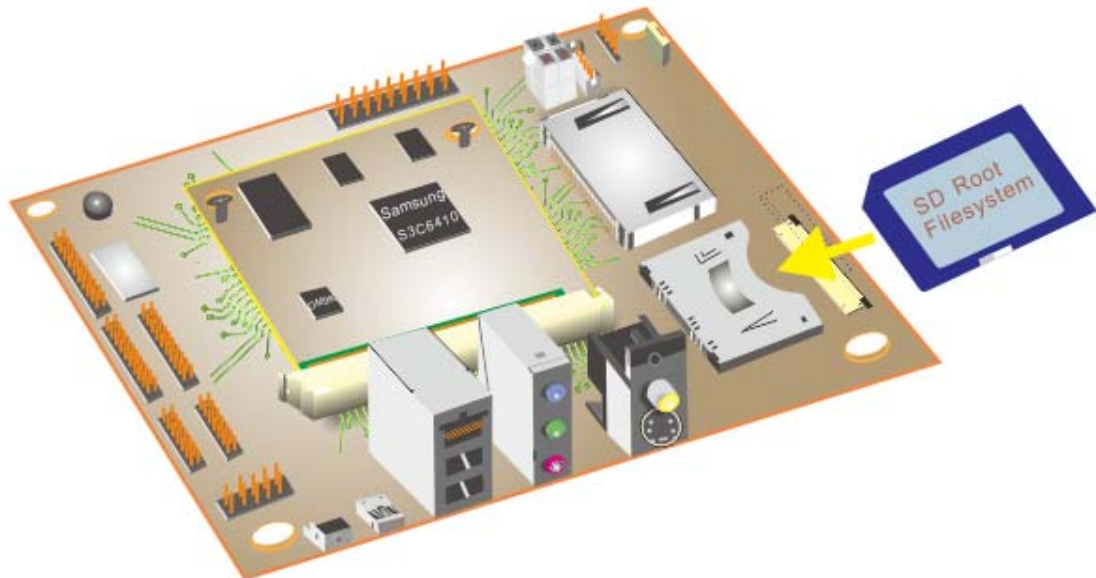
**Figure 3.1 Plug MXM-6410 into the carrier board and tighten it**



Plug MXM-6410 in the carrier board of the evaluation kit at 45 degrees and press down. Use a cross-head screw driver to tighten it. The recommended screws specification is M3 (Metric 3mm head), F-head (Flat head), 5mm in head diameter and 4mm long.

After done, plug the SD/SDHC card with pre-loaded file system into CN24 (SD Socket) connector. The SD/SDHC card with pre-loaded file system is part of the evaluation kit. The root file system for MXM-6410 is Ubuntu 9.04.

**Figure 3.2 Plug the rootfs pre-installed SD/SDHC card into evaluation kit**



Details in regarding to how to make a pre-loaded file system SD/SDHC 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, JP2 needs to be shunt depending on your LCD is 5V or 3.3V. JP5 is to determine if the backlight power is 5V or 12V. If 5V is shunt, the power source is from pin3 of CN8. If 12V is shunt, the power source will be pin2 of CH8.

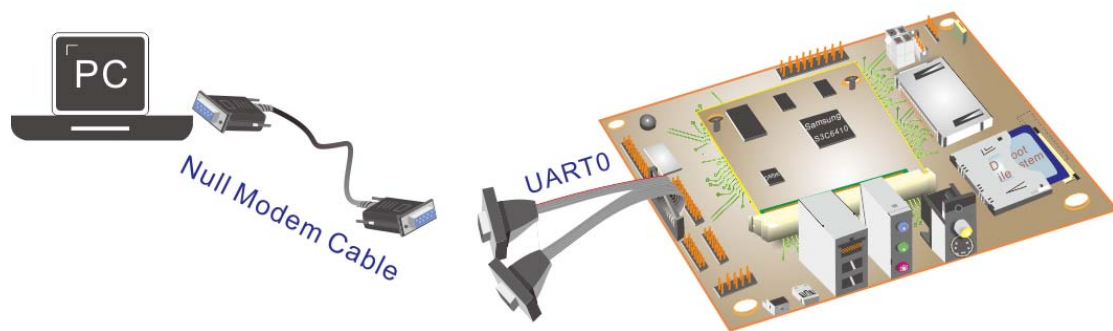
#### ***Note:***

Jumper and connector location can be found at APC-6410's User's Manual.

### ***Step 3: Connect the DUART Console Debug Cable from evaluation kit to a null modem cable and connect that null modem cable to your PC.***

Use Embedian DUART console cable and connect from CN20 of evaluation kit to a female-end null modem cable (TX/RX crossed) and connect that null modem cable to the COM port of your PC. Open the Hyperterminal program of your PC and set the baud rate as **115200, 8N1**.

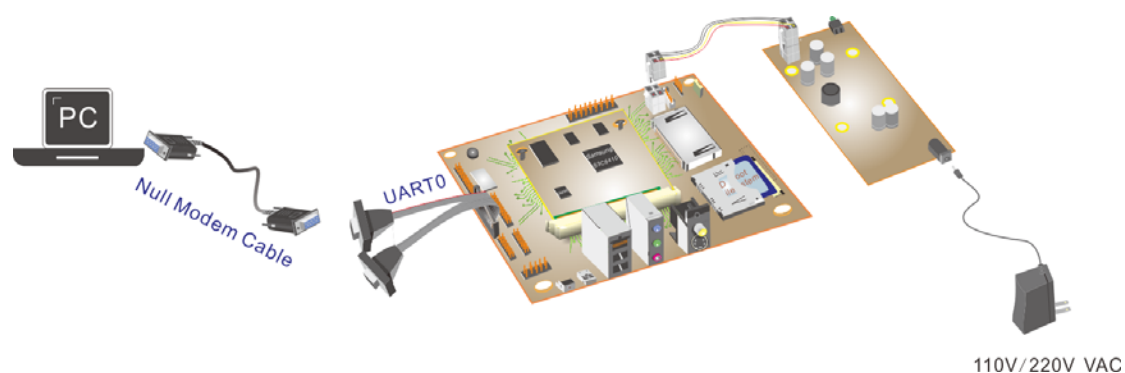
***Figure 3.3 Connect DUART Console Debug Cable to a Null Modem Cable and to your PC***



***Step 4: Apply 5V to the Evaluation Kit***

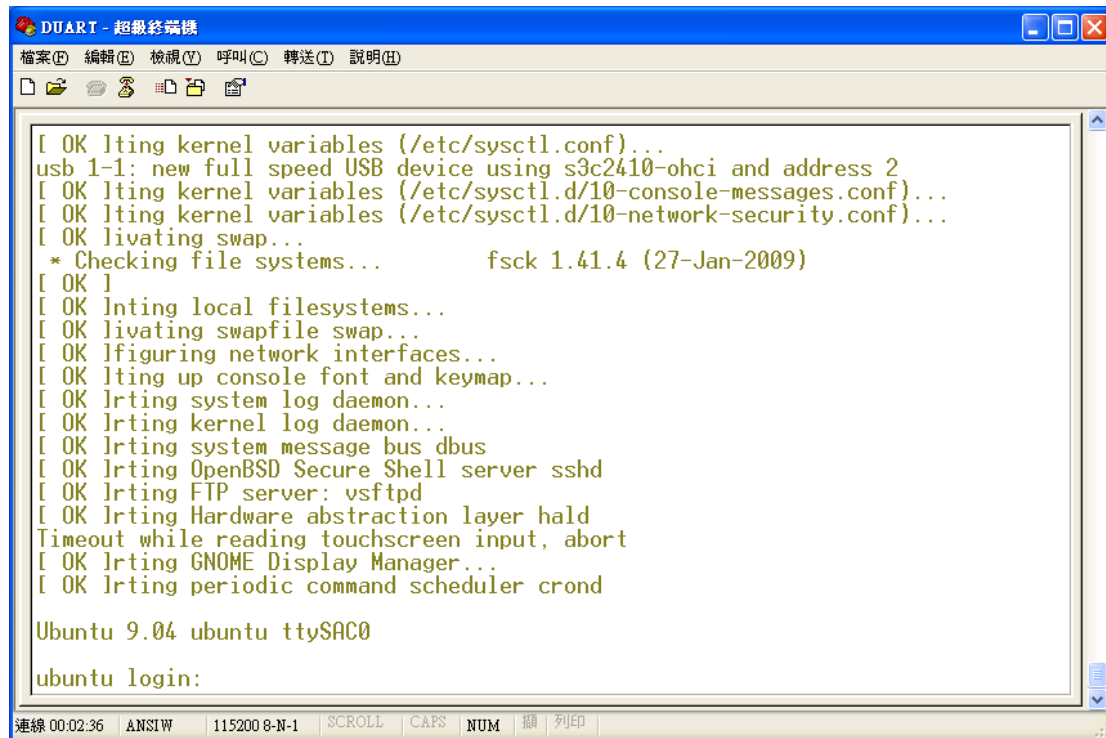
Connect the 12V-2A wall-mount power adapter to the power board and connect the power board to the evaluation board as shown in figure 3.4, the device will be power up. (Note: the power adapter, power board and cables are included in the evaluation kit. Power board mainly converts 12VDC to 5 VDC.)

***Figure 3.4 Apply Power to Evaluation Kit***



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

**Figure 3.5 Boot up messages from Hyperterminal**



```
[ OK lting kernel variables (/etc/sysctl.conf)...
usb 1-1: new full speed USB device using s3c2410-ohci and address 2
[ OK lting kernel variables (/etc/sysctl.d/10-console-messages.conf)...
[ OK lting kernel variables (/etc/sysctl.d/10-network-security.conf)...
[ OK livating swap...
* Checking file systems...          fsck 1.41.4 (27-Jan-2009)
[ OK ]
[ OK lnting local filesystems...
[ OK livating swapfile swap...
[ OK lfiguring network interfaces...
[ OK lting up console font and keymap...
[ OK lrtng system log daemon...
[ OK lrtng kernel log daemon...
[ OK lrtng system message bus dbus
[ OK lrtng OpenBSD Secure Shell server sshd
[ OK lrtng FTP server: vsftpd
[ OK lrtng Hardware abstraction layer hald
Timeout while reading touchscreen input, abort
[ OK lrtng GNOME Display Manager...
[ OK lrtng periodic command scheduler crond

Ubuntu 9.04 ubuntu ttySAC0

ubuntu login:
```

The default root password is “**mxm6410**” (no quot). You can use **passwd** command to change the root password. The default “ubuntu” user password is also “**mxm6410**” (no quot).

### Step 5: Network Configuration

Plug an Ethernet cable to CN17 of your device first.

The default network is set as DHCP. User can use *ifconfig eth0* command to check the device IP address.

```
ubuntu@ubuntu:~$ ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 10:0d:32:10:01:12
          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:95 errors:0 dropped:0 overruns:0 frame:0
          TX packets:3 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
```



## **Embedian, Inc.**

```
RX bytes:12326 (12.3 KB) TX bytes:158 (158.0 B)  
Interrupt:72 Base address:0xe300
```

```
ubuntu@ubuntu:~$
```

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

### **Example:**

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

```
ubuntu@ubuntu:~$ sudo ifconfig eth0 192.168.1.122 netmask 255.255.255.0  
up  
ubuntu@ubuntu:~$
```

### **Note:**

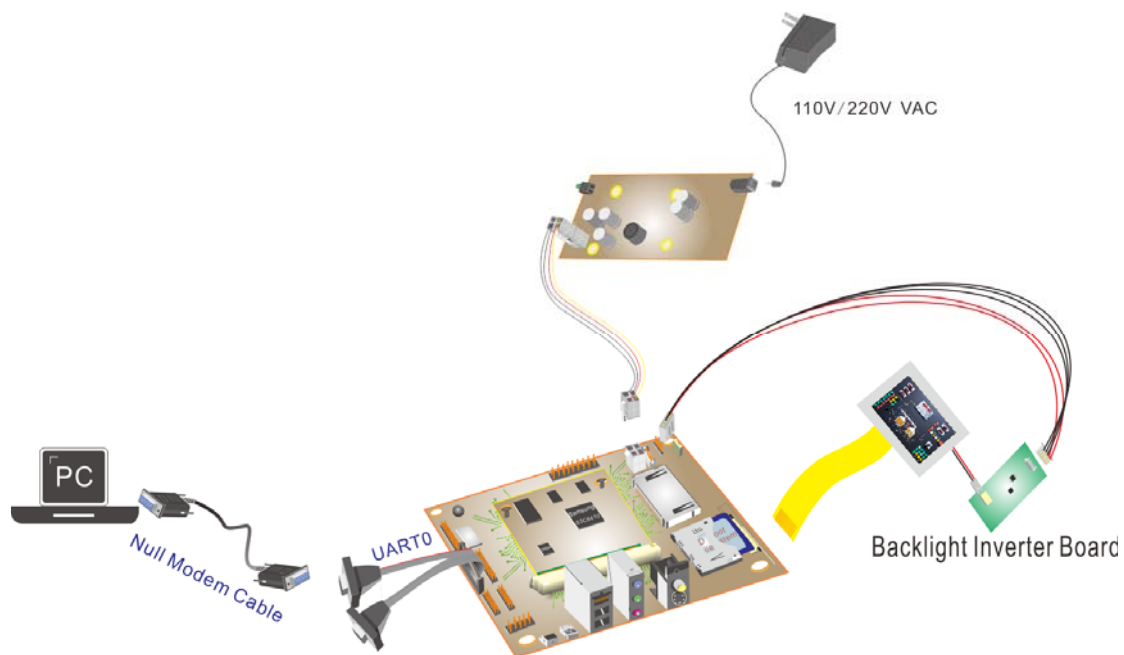
Every MAC address on board will be mapping to Embedian's serial number and is compliant to ISO/IEC 8802 standards. The MAC address is written on EEPROM of the board at factory and Linux kernel driver will read it automatically. There is no way that users can change it from root file systems.

## Step 6: LCD

If you need to connect an LCD, use a FPC cable or LVDS cable (depending on the type of LCD) connect to evaluation kit first. Check if JP2 and JP4 are configured properly. The FPC cable at the board side (CN14) is **top-contacted**. You will need to power the backlight yourself. The LCD backlight jumper block JP4 can be set to determine if the pin 1 of CN9 is bypassed from pin 3 of CN8 (5V) or pin2 of CN8 (12V). MXM-6410 evaluation kit provides users with a 5V or 12V bypass path from pin 2 or pin 3 of CN8 to pin 1 of CN9 for the backlight connection.

Figure 3.6 shows the LCD connection.

**Figure 3.6 LCD Connection**



The device descriptor of the LCD is `/dev/fb0`. The default LCD output will be Ubuntu gdm login screen.

The LCD driver is default built in the kernel instead of driver module.

If your LCD panel is in different resolutions other than 800x480, users also need to use `fbset` command to set up the frame buffer first. You need to `apt-get install fbset` package and `fbset` the framebuffer as follows.

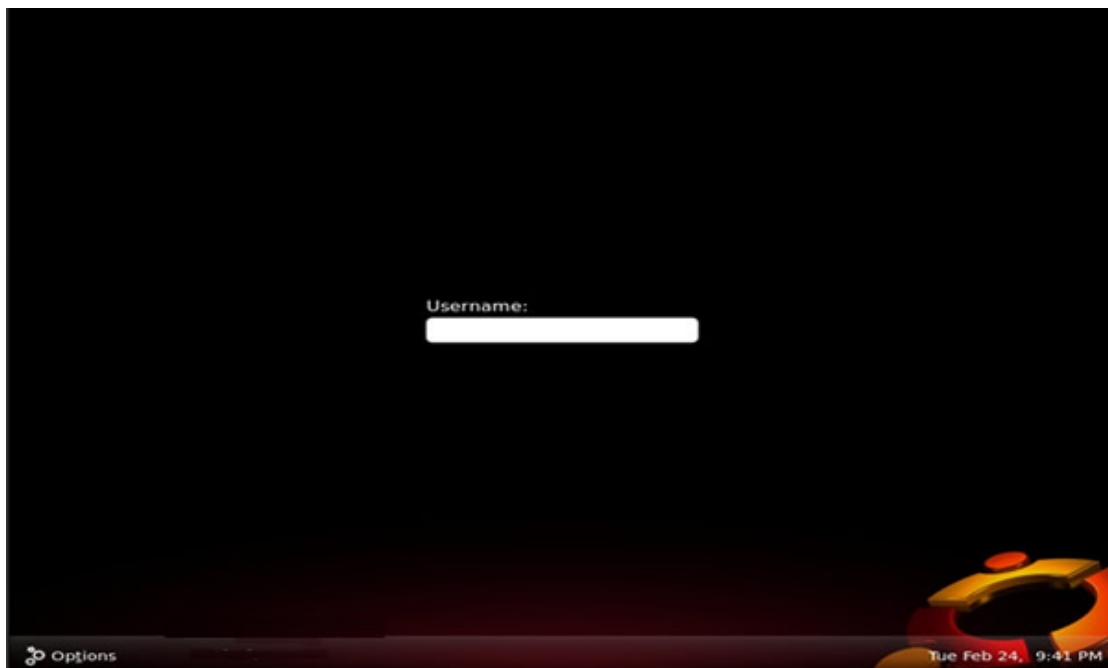
```
root@ubuntu:/etc/apt# sudo apt-get install fbset
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  fbset
0 upgraded, 1 newly installed, 0 to remove and 14 not upgraded.
Need to get 115kB of archives.
```

```
After this operation, 365kB of additional disk space will be used.  
Get:1 http://old-releases.ubuntu.com jaunty/main fbset 2.1-23 [115kB]  
Fetched 115kB in 2s (50.6kB/s)  
Selecting previously deselected package fbset.  
(Reading database ... 34128 files and directories currently installed.)  
Unpacking fbset (from .../fbset_2.1-23_armel.deb) ...  
Setting up fbset (2.1-23) ...  
  
ubuntu@ubuntu:/etc/apt# fbset 800x480-0  
ubuntu@ubuntu:/etc/apt#
```

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.

If your jumper configuration and wiring are correct, you will see the following snapshot on the LCD panel. Use an USB keyboard and mouse to login into Ubuntu system.

**Figure 3.7 Ubuntu 9.04 Boot Up Snapshot**



### **Step 7: Touch and Calibration**

If users would like to use touch screen, connect the 4-wire touch to CN28 connector. Touch panel is calibrated by default. To re-calibrate, users can use the following command line.

```
ubuntu@ubuntu:~# sudo ts_calibrate
```

Or pending at the touch screen for a while when booting.

There are five cross points for users to calibrate. The calibration value will be stored in the NAND flash after calibrated and taking effect at next boot.

# Chapter 4

## Hardware References

This Chapter contains detailed and specialized hardware information.

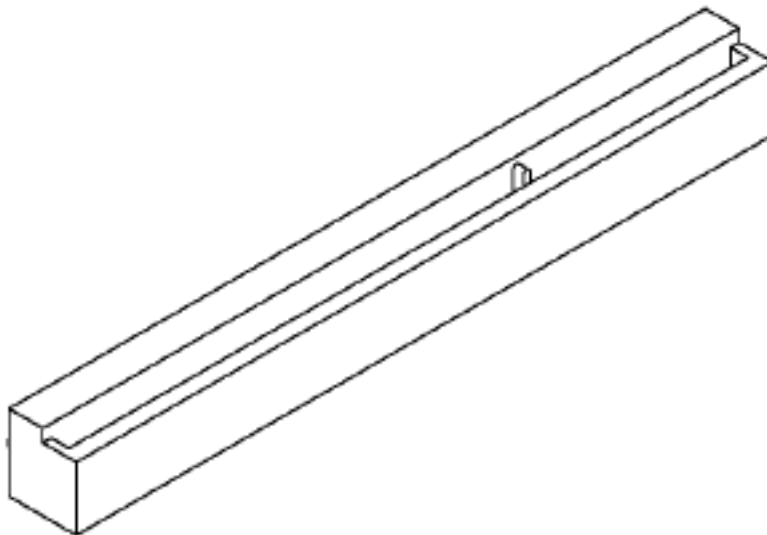
# Chapter 4 Hardware References

This section gives details of the hardware pin out assignment of the MXM-6410.

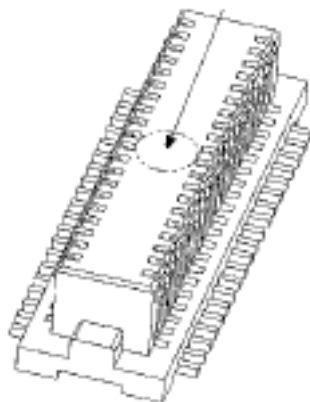
## **4.1 Connector Type**

The MXM-6410 uses MXM 242-pin golden finger as interface. The connector on module is called header and the connector on custom board is called socket.

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



**Figure 4.2 CN2 Header Type (Connector: DF12-40DS-0.5V (\*\*)) (Hirose) or compatible)**



**Figure 4.3 CN2 Socket Type (Mating Connector: DF12(5.0)-40DP-0.5V (\*\*)) (Hirose) or compatible)**



## **4.2 Connector Mechanical Drawing**

The detail connector mechanical drawing is as follows.

**Figure 4.4 CN1 Socket Connector Mechanical Drawing**

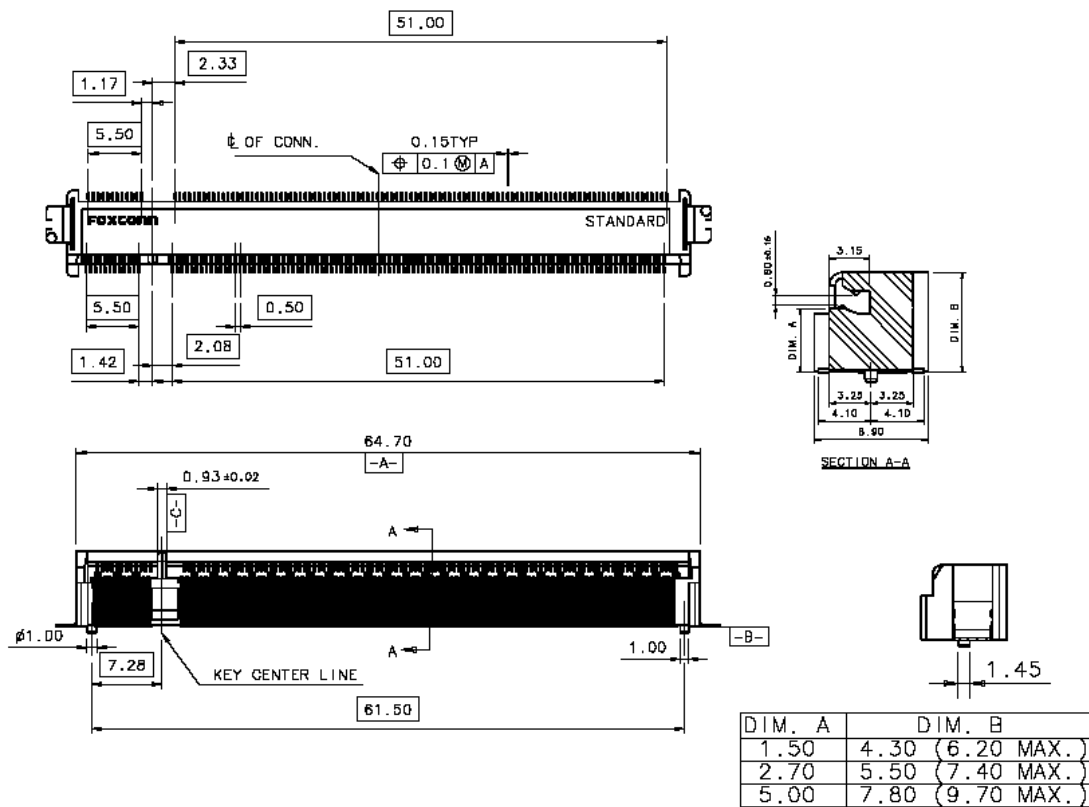
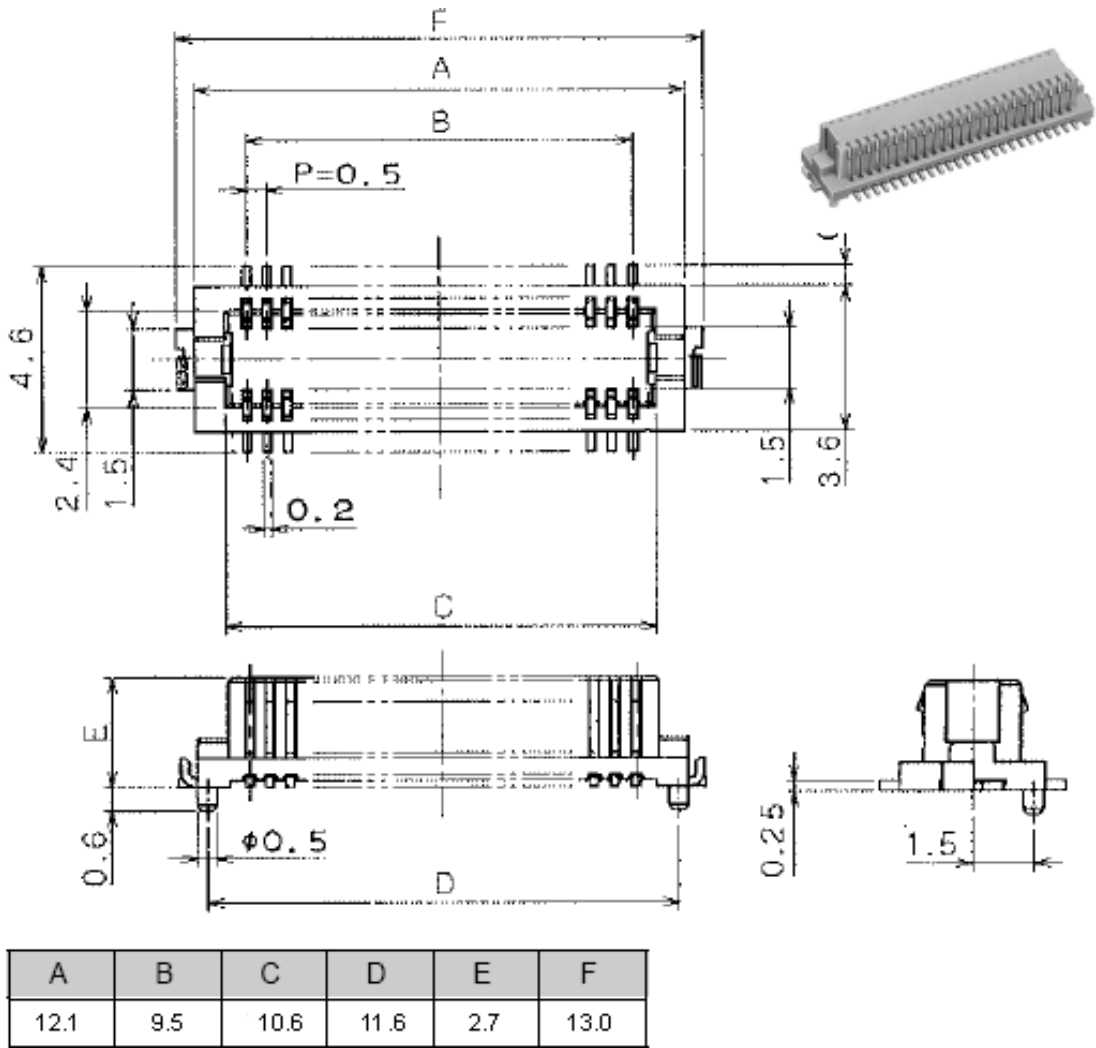
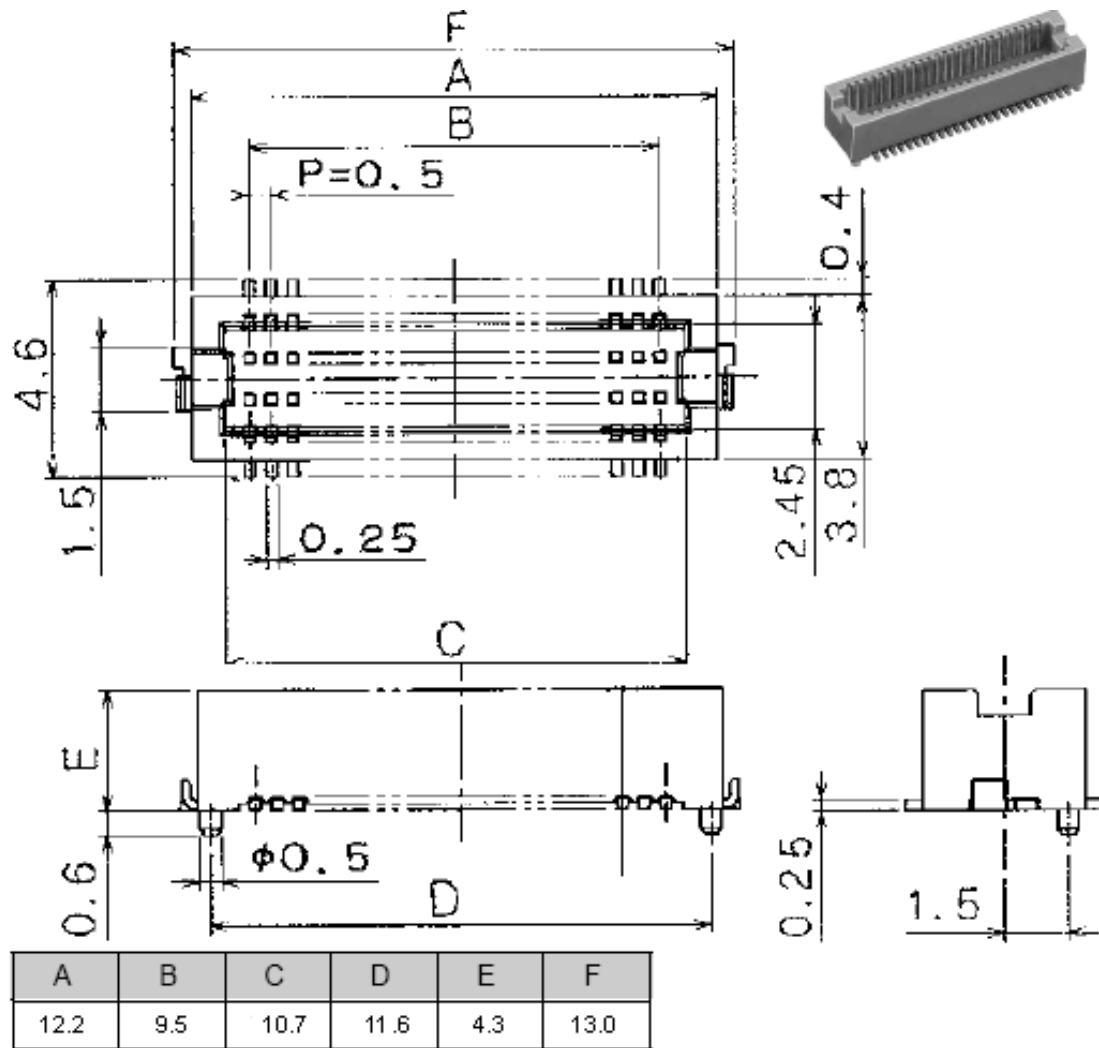




Figure 4.5 CN2 Header Connector Mechanical Drawing



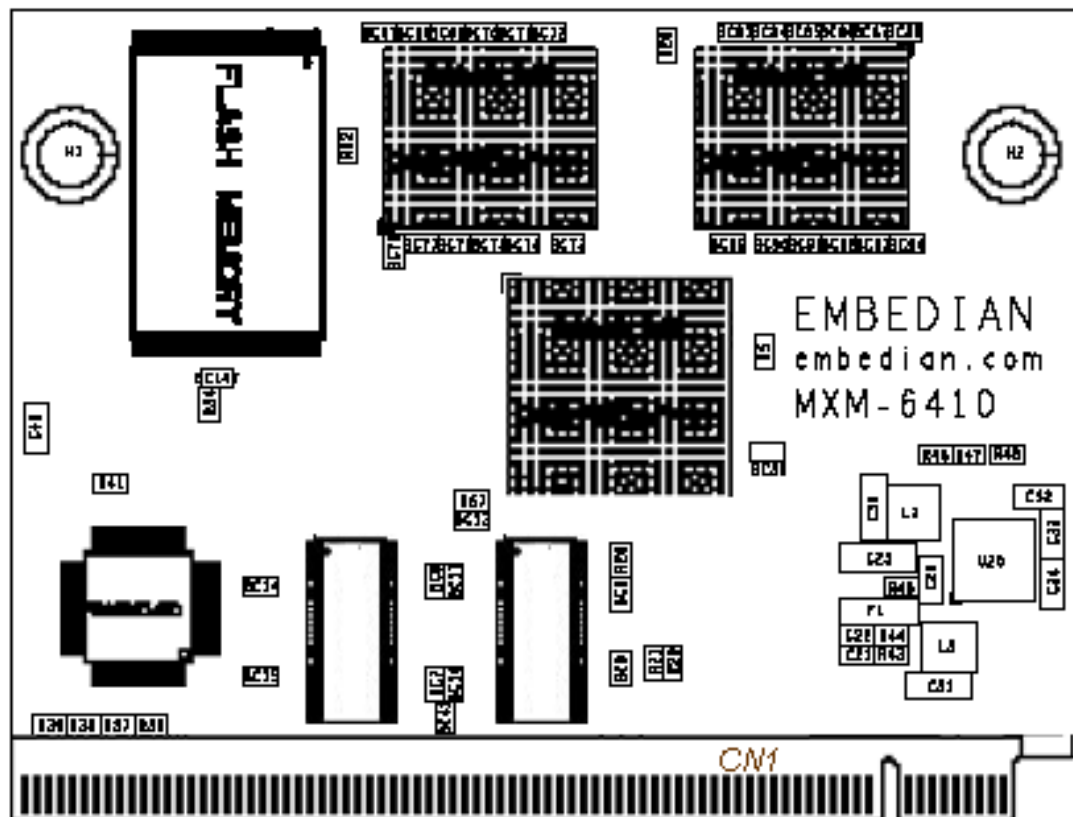
**Figure 4.6 CN2 Socket Connector Mechanical Drawing**



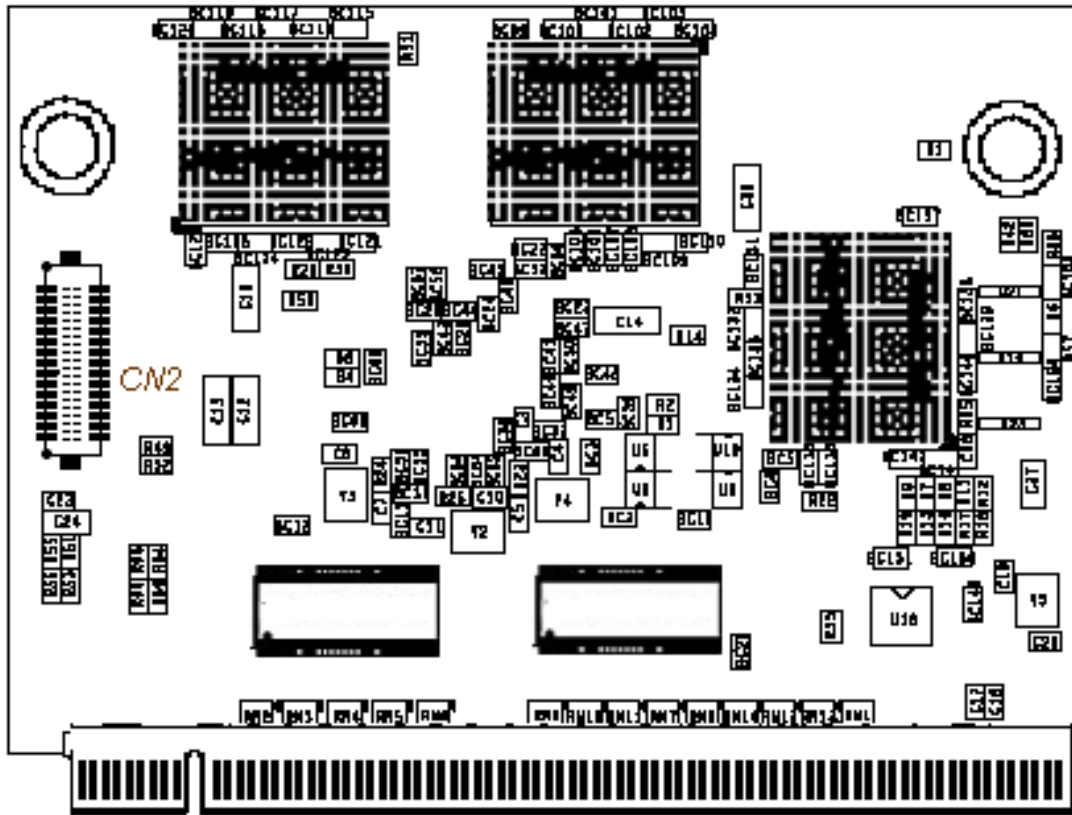
### 4.3 Connector Location

MXM series computer on module use 242-pin MXM form factor golden finger connectors CN1 and a 40-pin DF12(5.0)-40DS-0.5V (\*\*) (Hirose) or compatible connector CN2 as an interface to connect with carrier board. The CN2 is mainly for modem chip related. For other manufactures that are intend to use the MXM form factor, please add your additional pin out to CN2 as well.

Figure 4.3 Connector Location I



### Figure 4.4 Connector Location II



### 4.3.1. Connector Pin Assignments

The following tables describe the electrical signals available on the connectors of the MXM-6410. 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-6410, refer to section 2.2.2. for mechanical drawing.

**Legend :**

<i>NC</i>	Not Connected
<i>RSVD</i>	Reserved for future platform, suggest open at current design
<i>GND</i>	MXM-6410 Ground Plane

### **Signal Types :**


<i>I</i>	signal is an input to the system
<i>O</i>	signal is an output to the system
<i>IO</i>	signal may be input or output
<i>P</i>	power and ground
<i>A</i>	analog signal
<i>ST</i>	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 4.1:		CN1 Connector (Bottom Side)			
Description		Mating Connector : B33P102-0013 (Speed Tech), AS0B326-S78N-7F (Foxconn) or compatible			
Header		Pin	Signal Name	Function	Type
<div>Bottom Side</div> <div><div>1</div><div>3</div></div> <div>241</div>	4-wire Touch Screen				
	1	XP	Plus X-axis on-off control signal	AI	
	3	XM	Minus X-axis on-off control signal	AI	
	5	YP	Plus Y-axis on-off control signal	AI	
	7	YM	Minus Y-axis on-off control signal	AI	
	ADC Input				
	9	AIN3	ADC Input	AI	
	11	AIN2	ADC Input	AI	
	13	AIN1	ADC Input	AI	
	15	AIN0	ADC Input	AI	
	Reserved Pin				
	17	RSVD	Reserved	N.C.	
	19	RSVD	Reserved	N.C.	
	21	RSVD	Reserved	N.C.	
	23	RTCK	TAP Controller Return Clock (for JTAG)	O	
	Key				
	Key				
	Key				
	Key				
	Key				
	Key				
	37	RSVD	Reserved	N.C.	
	39	RSVD	Reserved	N.C.	
	DMA				
	41	DMAACK0	External DMA acknowledge	O	
	43	DMADREQ0	External DMA request	I	
	Address Bus				
	45	ADDR0	Address Bus	O	
	47	ADDR1		O	

49	ADDR2		0
51	ADDR3		0
53	ADDR4		0
55	ADDR5		0
57	ADDR6		0
59	ADDR7		0
61	ADDR8		0
63	ADDR9		0
65	ADDR10		0
67	ADDR11		0
69	ADDR12		0
71	ADDR13		0
73	ADDR14		0
75	ADDR15		0
77	GND	Ground Power	P
<b>Reserved</b>			
79	RSVD	Reserved	0
81	RSVD		0
83	RSVD		0
85	RSVD		0
87	RSVD		0
89	VD16	LCD data bus RED0 (LSB)	0
91	VD17	LCD data bus RED1	0
93	VD8	LCD data bus GREEN0 (LSB)	0
95	VD9	LCD data bus GREEN1	0
97	VD0	LCD data bus BLUE0 (LSB)	0
99	VD1	LCD data bus BLUE1	0
<b>Reserved Pin</b>			
101	CD_ATA	CF Card detection	I
103	INT_ATA	CF Interrupt request from ATA controller	I
105	RESET_ATA	CF CARD Reset	0
107	OE_ATA	CF Output enable strobe	0
109	WE_ATA	CF Write enable strobe	0
111	RSVD	Reserved	N.C.
113	ADRVALID	OneNAND Address Valid	0
115	GND	Ground	P
<b>Chip Select</b>			

	117	nGCS0	Chip Select	O
	119	nGCS1*		O
	121	nGCS2**		O
	123	nGCS3		O
	125	nGCS4		O
	127	nGCS5		O
	129	nWBE0	Write byte enable	O
	131	nWBE1		O
	133	nOE	Output Enable	O
	135	nWE	Write Enable	O
Data Bus				
	137	DATA0	DATA[15:0]  INPUT DATA DURING MEMORY READ AND OUTPUT DATA DURING MEMORY WRITE. BUS WIDTH OF 8/16 BIT IS PROGRAMMABLE	I/O
	139	DATA1		I/O
	141	DATA2		I/O
	143	DATA3		I/O
	145	DATA4		I/O
	147	DATA5		I/O
	149	DATA6		I/O
	151	DATA7		I/O
	153	DATA8		I/O
	155	DATA9		I/O
	157	DATA10		I/O
	159	DATA11		I/O
	161	DATA12		I/O
	163	DATA13		I/O
	165	DATA14		I/O
	167	DATA15		I/O
	169	nWAIT	nWAIT requests	I
	171	CLKOUT	Clock Output	O
	173	WAKEUP	Wakeup requests	I
	175	nRESET_IN	Reset S3C6410A	ST
	177	nRESET_OUT	Reset External Device	O
	179	RSVD	Reserved	N.C.
	181	RSVD		N.C.
	183	RSVD		N.C.
USB Host				
	185	USBH-	USB Host Data -	I/O
	187	USBH+	USB Host Data +	I/O
Reserved				
	189	RSVD	Reserved	N.C.
	191	RSVD		N.C.
USB OTG				

193	USBD-	USB2.0 OTG Data-	I/O
195	USBD+	USB2.0 OTG Data+	I/O
197	GND	GND POWER	P
199	USBVBUS	USB Mini-Receptacle Vbus	P
201	USBID	USB Mini-Receptacle Identifier	I
203	USBDRWBUS	Drive Vbus for Off-Chip Charge Pump	O
205	RSVD		N.C.
207	XSELNAND	Select Flash Memory	I
209	OM1	Operation mode selection	I
211	OM2	Operation mode selection	I
213	OM3	Operation mode selection	I
215	OM4	Operation mode selection	I
<b>Camera Interface</b>			
217	XCICLK	Master Clock to the Camera processor	O
219	XCIHREF	Horizontal Sync, driven by the Camera processor	I
221	XCIPCLK	Pixel Clock, driven by the Camera processor	I
223	XCInRST	Software Reset to the Camera processor	O
225	XCISYNC	Vertical Sync, driven by the Camera processor	I
227	XCIYDATA0	Pixel Data, driven by the Camera processor	I
229	XCIYDATA1	Pixel Data, driven by the Camera processor	I
231	XCIYDATA2	Pixel Data, driven by the Camera processor	I
233	XCIYDATA3	Pixel Data, driven	I




			<i>by the Camera processor</i>	
	235	XCIYDATA4	<i>Pixel Data, driven by the Camera processor</i>	<i>I</i>
	237	XCIYDATA5	<i>Pixel Data, driven by the Camera processor</i>	<i>I</i>
	239	XCIYDATA6	<i>Pixel Data, driven by the Camera processor</i>	<i>I</i>
	241	XCIYDATA7	<i>Pixel Data, driven by the Camera processor</i>	<i>I</i>

**(\*) nGCS1 is reserved for DM9000B**

**(\*\*) nGCS2 is reserved for NAND Flash**

**Table 4.2 CN1 Connector (Top Side)**

<b>Table 4.2:</b>	<b>CN1 Connector (Top Side)</b>			
<b>Description</b>	<b>Mating Connector : B33P102-0013 (Speed Tech), AS0B326-S78N-7F (Foxconn) or compatible</b>			
<b>Header</b>	<b>Pin</b>	<b>Signal Name</b>	<b>Function</b>	<b>Type</b>
 <p>Top Side</p> <p>2 4</p> <p>242</p>	<b>Reserved Pin</b>			
	2	DACOUT0	TV-OUT	A.O.
	4	DACOUT1	TV-OUT	A.O.
	<b>JTAG</b>			
	6	TMS	TAP Controller Mode Select	I
	8	TDO	TAP Controller Data Output	O
	10	TDI	TAP Controller Data Input	I
	12	TCK	TAP Controller Clock	I
	14	nTRST	TAP Controller Reset	I
	<b>AC97</b>			
	16	AC_SYNC	48kHz fixed rate sample sync	O
	18	AC_BIT_CLK	12.288MHz serial data clock	I/O
	20	AC_nRESET	AC'97 Master H/W Reset	O
	22	AC_SDATA_IN	AC'97 input stream	I
	24	AC_SDATA_OUT	AC'97 output stream	O
	Key			
	Key			
	Key			
	Key			
	Key			
	Key			
	<b>Power Input</b>			
	38	EXT5V	DC in 5V	P

	40	EXT5V	DC in 5V	P
	42	EXT5V	DC in 5V	P
	44	EXT5V	DC in 5V	P
	<b>LCD</b>			
	46	VD18	LCD data bus RED2	O
	48	VD19	LCD data bus RED3	O
	50	VD20	LCD data bus RED4	O
	52	VD21	LCD data bus RED5	O
	54	VD22	LCD data bus RED6	O
	56	VD23	LCD data bus RED7 (MSB)	O
	58	VD10	LCD data bus GREEN2	O
	60	VD11	LCD data bus GREEN3	O
	62	VD12	LCD data bus GREEN4	O
	64	VD13	LCD data bus GREEN5	O
	66	VD14	LCD data bus GREEN6	O
	68	VD15	LCD data bus GREEN7 (MSB)	O
	70	VD2	LCD data bus BLUE2	O
	72	VD3	LCD data bus BLUE3	O
	74	VD4	LCD data bus BLUE4	O
	76	VD5	LCD data bus BLUE5	O
	78	VD6	LCD data bus	O

		<i>BLUE6</i>	
80	<i>VD7</i>	<i>LCD data bus BLUE7 (MSB)</i>	<i>O</i>
82	<i>VCLK</i>	<i>LCD clock signal</i>	<i>O</i>
84	<i>HSYNC</i>	<i>Horizontal synchronous signal</i>	<i>O</i>
86	<i>VSYNC</i>	<i>Vertical synchronous signal</i>	<i>O</i>
88	<i>VDEN</i>	<i>Data enable signal</i>	<i>O</i>
90	<i>GND</i>	<i>Ground Power</i>	<i>P</i>
<b><i>PWM</i></b>			
92	<i>PWM0</i>	<i>Pulse Width</i>	<i>O</i>
94	<i>PWM1</i>	<i>Modulation Output</i>	<i>O</i>
<b><i>Reserved</i></b>			
96	<i>RSVD</i>	<i>Reserved</i>	<i>N.C</i>
98	<i>RSVD</i>	<i>Reserved</i>	<i>N.C.</i>
<b><i>IIC</i></b>			
100	<i>IIC_SCL</i>	<i>IIC-bus clock</i>	<i>I/O</i>
102	<i>IIC_SDA</i>	<i>IIC-bus data</i>	<i>I/O</i>
<b><i>SPI</i></b>			
104	<i>SPIMISO0</i>	<i>Master mode: data input; Slave mode: data output</i>	<i>I/O</i>
106	<i>SPIMOSI0</i>	<i>Master mode: data output; Slave mode: data input</i>	<i>I/O</i>
108	<i>SPICLK0</i>	<i>SPI Clock</i>	<i>I/O</i>
110	<i>nSS0</i>	<i>SPI Chip Select</i>	<i>I</i>
112	<i>SPIMISO1</i>	<i>Master mode: data input; Slave mode: data output</i>	<i>I/O</i>
114	<i>SPIMOSI1</i>	<i>Master mode: data output; Slave mode: data</i>	<i>I/O</i>

			input	
116	SPICLK1	SPI Clock	I/O	
118	nSS1	SPI Chip Select	I	
<b>Interrupt</b>				
120	EXT_INT1	External interrupt request	I	
122	EXT_INT2		I	
124	EXT_INT3		I	
126	EXT_INT4		I	
128	EXT_INT5		I	
130	EXT_INT6		I	
132	EXT_INT7		I	
134	EXT_INT8		I	
136	GND	Ground Power	P	
<b>GPIO</b>				
138	GPIO1	General input/output ports	I/O	
140	GPIO2		I/O	
142	GPIO3		I/O	
144	GPIO4		I/O	
146	GPIO5		I/O	
148	GPIO6		I/O	
150	GPIO7		I/O	
152	GPIO8		I/O	
154	GPIO9		I/O	
156	GPIO10		I/O	
158	GPIO11		I/O	
160	GPIO12		I/O	
162	VCCIO_PWREN	External Device Power Control	O	
164	VCCLCD_PWREN	Panel Power Control	O	
166	BACKLIGHT_EN	Panel Backlight Control	O	
168	LCD_PWREN	Panel Signal Control	O	
170	BBAT	RTC Battery Power(DC 3V)	P	

SD Card			
172	SD_nCD	SD Insert Detect	I
174	SD_WP	SD Write Protect	I
176	SDCLK	SD Clock	O
178	SDCMD	SD receive response/ transmit command	O
180	SDDAT0	SD receive/transmit data	I/O
182	SDDAT1	SD receive/transmit data	I/O
184	SDDAT2	SD receive/transmit data	I/O
186	SDDAT3	SD receive/transmit data	I/O
188	GND	Ground Power	P
UART			
190	RXD3	UART receives data input	I
192	TXD3	UART transmits data output	O
Reserved			
194	RSVD	Reserved	N.C.
196	RSVD		N.C.
198	RSVD		N.C.
200	RSVD		N.C.
202	RSVD		N.C.
204	RSVD		N.C.
UART			
206	RXD2	UART receives data input	I
208	TXD2	UART transmits data output	O

	210	<i>nCTS1</i>	<i>UART clear to send input signal</i>	<i>I</i>
	212	<i>nRTS1</i>	<i>UART request to send output signal</i>	<i>O</i>
	214	<i>RXD1</i>	<i>UART receives data input</i>	<i>I</i>
	216	<i>TXD1</i>	<i>UART transmits data output</i>	<i>O</i>
	218	<i>nCTS0</i>	<i>UART clear to send input signal</i>	<i>I</i>
	220	<i>nRTS0</i>	<i>UART request to send output signal</i>	<i>O</i>
	222	<i>RXD0</i>	<i>UART receives data input</i>	<i>I</i>
	224	<i>TXD0</i>	<i>UART transmits data output</i>	<i>O</i>
	<b>Ethernet</b>			
	226	<i>LANLED1</i>	<i>Ethernet Speed LED</i>	<i>O</i>
	228	<i>LANLED2</i>	<i>Ethernet Link LED</i>	<i>O</i>
	230	<i>AVDD18</i>	<i>1.8V For Transformer</i>	<i>P</i>
	232	<i>TX-</i>	<i>Ethernet Transmits data-</i>	<i>O</i>
	234	<i>TX+</i>	<i>Ethernet Transmits data+</i>	<i>O</i>
	236	<i>AGND</i>	<i>Ethernet Ground</i>	<i>P</i>
	238	<i>RX-</i>	<i>Ethernet Receives data-</i>	<i>I</i>
	240	<i>RX+</i>	<i>Ethernet Receives data+</i>	<i>I</i>
	242	<i>AVDD18</i>	<i>1.8V For Transformer</i>	<i>P</i>

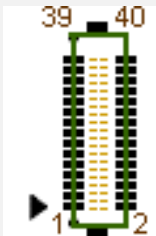
**(\*)At this moment, no Linux driver support.**

#### 4.3.1.2. CN2 Connector

All modem related interface is from CN2.

The following table shows the pin outs of CN2 connector.

**Table 4.3 CN2 Connector**

<b>Table 4.3: CN2 Connector</b>				
<b>Description</b>	<b>Connector: DF12-40DS-0.5V (**) (Hirose) or compatible</b> <b>Mating Connector : DF12(5.0)-40DP-0.5V (**) (Hirose) or compatible</b>			
<b>Header</b>	<b>Pin</b>	<b>Signal Name</b>	<b>Function</b>	<b>Type</b>
	2	XHIDATA0	Data bus, driven by the Modem chip	IO
	4	XHIDATA1	Data bus, driven by the Modem chip	IO
	6	XHIDATA2	Data bus, driven by the Modem chip	IO
	8	XHIDATA3	Data bus, driven by the Modem chip	IO
	10	XHIDATA4	Data bus, driven by the Modem chip	IO
	12	XHIDATA5	Data bus, driven by the Modem chip	IO
	14	XHIDATA6	Data bus, driven by the Modem chip	IO
	16	XHIDATA7	Data bus, driven by the Modem chip	IO
	18	XHIDATA8	Data bus, driven by the Modem chip	IO



	20	XHIDATA9	Data bus, driven by the Modem chip	IO
	22	XHIDATA10	Data bus, driven by the Modem chip	IO
	24	XHIDATA11	Data bus, driven by the Modem chip	IO
	26	XHIDATA12	Data bus, driven by the Modem chip	IO
	28	XHIDATA13	Data bus, driven by the Modem chip	IO
	30	XHIDATA14	Data bus, driven by the Modem chip	IO
	32	XHIDATA15	Data bus, driven by the Modem chip	IO
	34	XHIDATA16	Data bus, driven by the Modem chip	IO
	36	XHIDATA17	Data bus, driven by the Modem chip	IO
	38	RSVD	Reserved	N.C.
	40	GND	Ground	P
	1	XHIADR0	Address bus, driven by the Modem chip	I
	3	XHIADR1	Address bus, driven by the Modem chip	I
	5	XHIADR2	Address bus,	I

			<i>driven by the Modem chip</i>	
	7	XHIADR3	<i>Address bus, driven by the Modem chip</i>	<i>I</i>
	9	XHIADR4	<i>Address bus, driven by the Modem chip</i>	<i>I</i>
	11	XHIADR5	<i>Address bus, driven by the Modem chip</i>	<i>I</i>
	13	XHIADR6	<i>Address bus, driven by the Modem chip</i>	<i>I</i>
	15	XHIADR7	<i>Address bus, driven by the Modem chip</i>	<i>I</i>
	17	XHIADR8	<i>Address bus, driven by the Modem chip</i>	<i>I</i>
	19	XHIADR9	<i>Address bus, driven by the Modem chip</i>	<i>I</i>
	21	XHIADR10	<i>Address bus, driven by the Modem chip</i>	<i>I</i>
	23	XHIADR11	<i>Address bus, driven by the Modem chip</i>	<i>I</i>
	25	XHIADR012	<i>Address bus, driven by the Modem chip</i>	<i>I</i>
	27	XHInCS	<i>Chip select, driven by the Modem chip</i>	<i>I</i>
	29	XHInCS	<i>Chip select for LCD bypass</i>	<i>I</i>

			<i>main, driven by the Modem chip</i>	
	31	<i>XHInCS_SUB</i>	<i>Chip select for LCD bypass sub, driven by the Modem chip</i>	<i>I</i>
	33	<i>XHInWE</i>	<i>Write enable, driven by the Modem chip</i>	<i>I</i>
	35	<i>XHInOE</i>	<i>Read enable, driven by the Modem chip</i>	<i>I</i>
	37	<i>XHInIRQ</i>	<i>Interrupt request to the Modem chip</i>	<i>O</i>
	39	<i>GND</i>	<i>Ground</i>	<i>P</i>

# Chapter

# 5

## Using Ubuntu Jaunty Jackalope (9.04)

This Chapter details how to use the Ubuntu Linux of MXM-6410 computer on module.

Section include :

- Booting
- Default root pass and user
- Network Setting
- Manually add repositories
- Install Software Packages
- COM Port
- LCD
- GPIO
- FTP Client
- FTP Server

***Embedian, Inc.***

- Telnet/SSH Server
- VNC server
- GDM
- LXDE
- Calibration and Touch Screen
- Kiosk Mode
- NAND Root File System
- Cross Toolchain

# Chapter 5 Using Ubuntu Jaunty Jackalope

Ubuntu 9.04 (as known as Jaunty Jackalope) is official released on April 23<sup>rd</sup> of 2009. And Embedian is one of the first few companies to support Ubuntu 9.04 for ARM11 processors. This Chapter gives an overall picture in regarding to the Ubuntu Linux 9.04 features that Embedian provided with for the MXM-6410 computer on module.

This chapter gives an introduction in regarding to use the Ubuntu Linux 9.04 Jaunty Jackalope system on MXM-6410 computer on module. This guide is mainly focus on the topic related to Embedian's products. This guide is not an official Ubuntu documentation. But still, a good reference for those people who are interested in Ubuntu 9.04.

This guide mainly uses MXM-6410 on the evaluation kit as an example.

## **5.1 Board Support Package (BSP)**

The Embedian Board Support Package for Ubuntu Linux 9.04 is one of the most advanced BSPs available in the market. Beside the standard Ubuntu Linux functionality, it also includes a large number of additional drivers as well as optimized versions from standard drivers.

The Embedian Linux kernel is provided with a source code and binary format. This allows customers to customize the external device drivers themselves. With Ubuntu support, this relieves the application-developer from the burden of creating and building the own images. Instead the necessary adaptations can be done by using **apt-get** the software packages from the official Ubuntu 9.04 repository to the on-board root file system.

Customers who follow the built instruction in the BSP will enable you to build exactly the same Linux kernel zImage as Embedian provides by default.

### **5.1.1. Drivers**

The following drivers are integrated in the standard image that comes pre-installed with each MXM-6410 and the evaluation board.

**Table 5.1 Drivers**

<b>Table 5.1</b>	<b>Drivers</b>
<b>Driver</b>	<b>Description</b>
<b>COM1</b>	Support RXD/TXD/CTS/RTS
<b>COM2</b>	Support RXD/TXD
<b>COM3</b>	Support RXD/TXD
<b>COM4</b>	External UART that supports the full RS232 specification with all 9 signals on carrier board
<b>COM5</b>	External UART that supports the full RS232 specification with all 9 signals on carrier board
<b>Ethernet</b>	10/100Mbit driver for Davicom DM9000B
<b>USB Host</b>	4 Ports supporting mass storage devices (USB-stick, hard disk...) and other devices as keyboard, mouse, USB hub...
<b>USB OTG</b>	Client
<b>Audio</b>	16Bit stereo output (up to 48kHz), mic input, line-In supports ALSA standard APIs
<b>Touch Screen</b>	All 4-wire resistive screens supported
<b>Display</b>	All types of displays supported. Use <b>fbset</b> utility to configure different resolutions and panels.
<b>SD/SDHC</b>	Memory cards, Wireless LAN, Bluetooth, GSM, GPS ...
<b>CF</b>	Memory cards, Wireless LAN, modem, Bluetooth, serial card, GPS, GSM, ...
<b>GPIO</b>	12 Configurable GPIO with Sample codes.
<b>Flash File System</b>	Ext3 and FAT32 are default supported
<b>2D Engine</b>	CPU 2D acceleration h/w supported
<b>3D Engine</b>	Support CPU 3D h/w acceleration. Support OpenGL ES 1.1/2.0.
<b>TV-Out</b>	TV output driver for NTSC/PAL format is supported
<b>Camera</b>	Camera drivers for various camera modules
<b>MFC Engine</b>	Support h/w H.263, H.264, VC1
<b>SPI</b>	High speed SPI supported
<b>RTC</b>	Real time clock driver

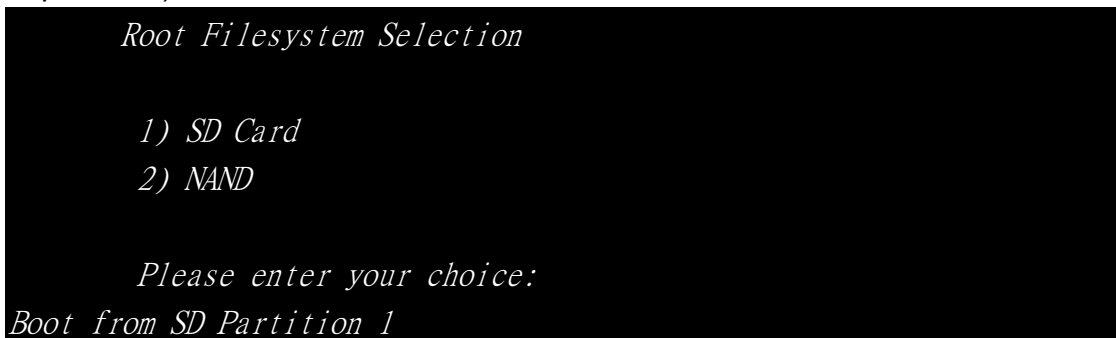
### 5.1.2. Default Software Packages

Users can use **dpkg-query -W** to visualize the list of installed packages or

***dpkg -I*** to obtain the description of the set of installed packages. User can add their application oriented packages from the Ubuntu repository by using ***apt-get install <Package Name>*** as well.

### ***5.1.3. Booting***

When power on, the uboot will initialize the low-level hardware and bring the Linux kernel to DDR RAM. After that, the Linux kernel will take over the system. The linuxrc is a program that is started in the start-up stage of the kernel prior to the actual boot process. This allows you to boot a small modularized kernel and to load only few drivers that are really needed as modules. linuxrc assists in loading relevant drivers manually. The use of linuxrc provides with the choices to boot into a small root file system in NAND or the Ubuntu 9.04 system in SD/SDHC card of carrier board. (The default is set to boot into SD/SDHC card if no key is pressed during boot process.)



```
Root Filesystem Selection

1) SD Card
2) NAND

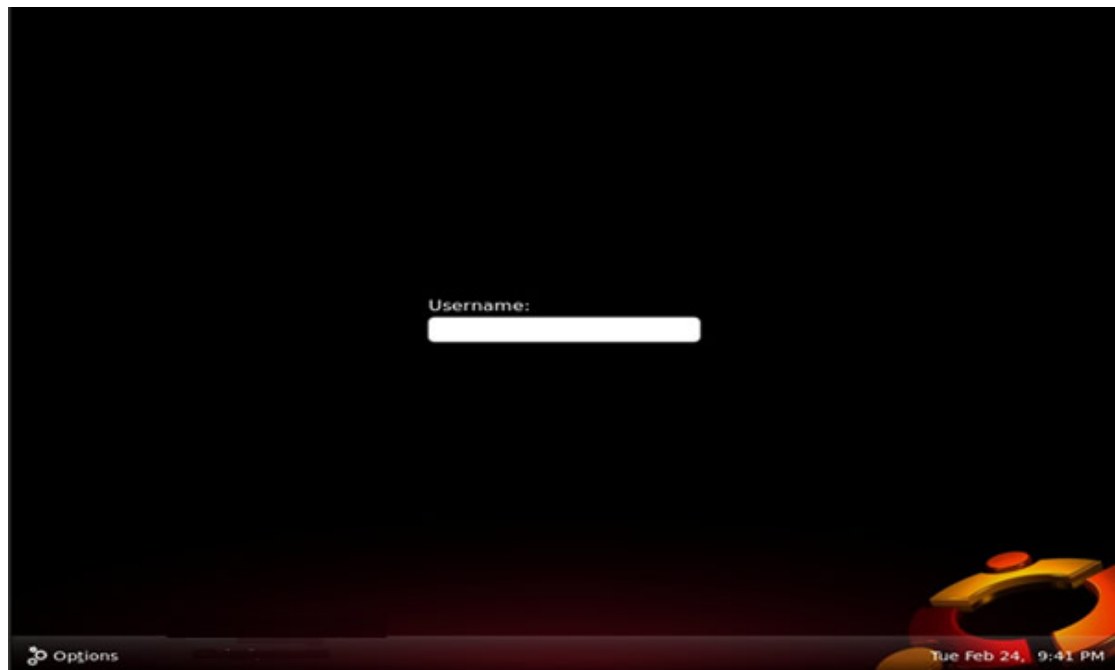
Please enter your choice:
Boot from SD Partition 1
```

The NAND file system is in ext3 format and can be served as a disk-based rescue system or for some simpler applications that don't need a SD root file system at all. For more details in regarding to NAND file system, users can refer to section 5.15.

If you have LCD attached, users should be able to see the following gdm login screen showing on your LCD.



***Embedian, Inc.***



## 5.2 Default root pass and user

The default **root** password is **mxm6410** and default user is **ubuntu** and the ubuntu user default password is also **mxm6410**.

### 5.2.1 Create a User

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

```
ubuntu@ubuntu:~$ useradd --help
Usage: useradd [options] LOGIN

Options:
  -b, --base-dir BASE_DIR      base directory for the new user account
                                home directory
  -c, --comment COMMENT        set the GECOS field for the new user
                                account
  -d, --home-dir HOME_DIR      home directory for the new user account
  -D, --defaults                print or save modified default useradd
                                configuration
  -e, --expiredate EXPIRE_DATE set account expiration date to
                                EXPIRE_DATE
  -f, --inactive INACTIVE      set password inactive after expiration
                                to INACTIVE
  -g, --gid GROUP               force use GROUP for the new user account
  -G, --groups GROUPS           list of supplementary groups for the new
                                user account
  -h, --help                    display this help message and exit
  -k, --skel SKEL_DIR           specify an alternative skel directory
  -K, --key KEY=VALUE           overrides /etc/login.defs defaults
  -l,                           do not add the user to the lastlog and
                                faillog databases
  -m, --create-home             create home directory for the new user
                                account
  -N, --no-user-group           do not create a group with the same name
                                as
                                the user
  -o, --non-unique              allow create user with duplicate
                                (non-unique) UID
```

## **Embedian, Inc.**

```
-p, --password PASSWORD    use encrypted password for the new user
                             account
-r, --system                create a system account
-s, --shell SHELL           the login shell for the new user account
-u, --uid UID               force use the UID for the new user account
-U, --user-group            create a group with the same name as the
user
ubuntu@ubuntu:~$
```

### 5.2.2 Set User Password

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

```
ubuntu@ubuntu:~$ passwd --help
Usage: passwd [options] [LOGIN]

Options:
  -a, --all                report password status on all accounts
  -d, --delete             delete the password for the named
account
  -e, --expire            force expire the password for the named
account
  -h, --help              display this help message and exit
  -k, --keep-tokens       change password only if expired
  -i, --inactive INACTIVE set password inactive after expiration
to INACTIVE
  -l, --lock              lock the password of the named account
  -n, --mindays MIN_DAYS set minimum number of days before
password
                           change to MIN_DAYS
  -q, --quiet             quiet mode
  -r, --repository REPOSITORY change password in REPOSITORY repository
  -S, --status            report password status on the named
account
  -u, --unlock            unlock the password of the named account
  -w, --warndays WARN_DAYS set expiration warning days to WARN_DAYS
  -x, --maxdays MAX_DAYS set maximim number of days before
password
                           change to MAX_DAYS

ubuntu@ubuntu:~$
```

#### Example:

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

## **Embedian, Inc.**

```
ubuntu@ubuntu:~$ sudo useradd -m john
ubuntu@ubuntu:~$ sudo passwd john
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
ubuntu@ubuntu:~$
```

### **5.2.3 Delete a User**

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

```
ubuntu@ubuntu:~$ userdel --help
Usage: userdel [options] LOGIN

Options:
  -f, --force          force removal of files,
                        even if not owned by user
  -h, --help           display this help message and exit
  -r, --remove         remove home directory and mail spool

ubuntu@ubuntu:~$
```

#### **Example:**

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

```
ubuntu@ubuntu:~$ sudo userdel -r john
ubuntu@ubuntu:~$
```

## **5.3 Network Settings**

The default network is set as DHCP. User can use *ifconfig eth0* command to check the device IP address.

```
ubuntu@ubuntu:~$ ifconfig eth0
```

```

eth0      Link encap:Ethernet  HWaddr 10:0d:32:10:01:12
          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:95 errors:0 dropped:0 overruns:0 frame:0
          TX packets:3 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:12326 (12.3 KB)  TX bytes:158 (158.0 B)
          Interrupt:72 Base address:0xe300

ubuntu@ubuntu:~$

```

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

### **Example:**

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

```

ubuntu@ubuntu:~$ sudo ifconfig eth0 192.168.1.122 netmask 255.255.255.0
up
ubuntu@ubuntu:~$

```

### **Note:**

Every MAC address on board will be mapping to Embedian's serial number and is compliant to ISO/IEC 8802 standards. The MAC address is written on EEPROM of the board at factory and Linux kernel driver will read it automatically. There is no way that users can change it from root file systems.

### **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 configuration at boot or network restart, users need to modify the */etc/network/interfaces* file. Network configuration will take effect at next boot or network restart.

#### **5.3.1.1 Static IP**

## **Embedian, Inc.**

To configure the device networking configuration as static IP, user need to modify the `/etc/network/interfaces` file first.

```
ubuntu@ubuntu:~$ vim /etc/network/interfaces
```

Modify the network configurations as follows

After modified this file, you can reboot the device or just use the `/etc/init.d/networking restart` command to restart the networking configuration to get the new network configurations.

```
auto lo
iface lo inet loopback
auto eth0
iface eth0 inet static
    address 192.168.1.121
    netmask 255.255.255.0
    gateway 192.168.1.254
```

### **5.3.1.2 DHCP**

To configure the device networking configuration as DHCP, user need to modify the `/etc/network/interfaces` as follows.

```
auto lo
iface lo inet loopback
auto eth0
#iface eth0 inet static
#    address 192.168.1.121
#    netmask 255.255.255.0
#    gateway 192.168.1.254
iface eth0 inet dhcp
```

After modified this file, you can reboot the device or just use the `/etc/init.d/networking restart` command to restart the networking configuration.

```
ubuntu@ubuntu:~$ sudo /etc/init.d/networking restart
```

```

* Reconfiguring network interfaces...
Internet Systems Consortium DHCP Client V3.1.1
Copyright 2004-2008 Internet Systems Consortium.
All rights reserved.
For info, please visit http://www.isc.org/sw/dhcp/

Listening on LPF/eth0/10:0d:32:10:01:12
Sending on   LPF/eth0/10:0d:32:10:01:12
Sending on   Socket/fallback
Internet Systems Consortium DHCP Client V3.1.1
Copyright 2004-2008 Internet Systems Consortium.
All rights reserved.
For info, please visit http://www.isc.org/sw/dhcp/

Listening on LPF/eth0/10:0d:32:10:01:12
Sending on   LPF/eth0/10:0d:32:10:01:12
Sending on   Socket/fallback
DHCPDISCOVER on eth0 to 255.255.255.255 port 67 interval 4
DHCPDISCOVER on eth0 to 255.255.255.255 port 67 interval 4
DHCPOFFER of 192.168.1.54 from 192.168.1.254
DHCPREQUEST of 192.168.1.54 on eth0 to 255.255.255.255 port 67
DHCPACK of 192.168.1.54 from 192.168.1.254
bound to 192.168.1.54 -- renewal in 40595 seconds.

[ OK ]
ubuntu@ubuntu:~$

```

You can see the device IP has been assigned as 192.168.1.54 after network restart.

**Note:**

At development stage, users might take the MXM-6410 module off from the carrier board and replace by the other new module. If that is the case, Ubuntu will find out the MAC address has been changed and will view the original eth0 network device as eth1. The network configuration settings mentioned above will not take effect. If users want to change the network configuration setting under this situation, you



## ***Embedian, Inc.***

need to modify the `/etc/udev/rules.d/70-persistent-net.rules` file first time when you replace the CPU module.

```
ubuntu@ubuntu:~$ sudo vim /etc/udev/rules.d/70-persistent-net.rules
```

And delete the following lines.

```
# net device ()  
SUBSYSTEM=="net", ACTION=="add", DRIVERS=="?*",  
ATTR{address}=="10:0d:32:10:01:1  
2", ATTR{type}=="1", KERNEL=="eth*", NAME="eth0"
```

After next reboot or network restart, Ubuntu will generate the new settings into this file to fix the network device interface.

## 5.4 Manually Add Repositories

Do this at your own risk. Modify the default Ubuntu sources.list only if you understand what you're doing. Mixing repositories can **break** your system. For more information see the [Ubuntu Command-line Repository guide](#).

Create a backup of your current list of sources.

```
ubuntu@ubuntu:~$ sudo cp -p /etc/apt/sources.list
/etc/apt/sources.list_backup
ubuntu@ubuntu:~$
```

### Note:

1. *sudo* - runs the command with root privileges. *cp* = copy. *-p* = prompt to overwrite if a file already exists.

2. Edit the list of sources:

```
ubuntu@ubuntu:~$ sudo vim /etc/apt/sources.list
```

The default repositories are:

```
deb http://old-releases.ubuntu.com/ubuntu jaunty main restricted
universe multiverse
```

After modified the sources.list, you need to run apt-install update to acquire the update packages.

```
ubuntu@ubuntu:~$ sudo apt-get update
```

3. To use your local mirror you can add "xx." before *archive.ubuntu.com*, where **xx** = your country code.

**Example:** *deb http://gb.archive.ubuntu.com/ubuntu jaunty main restricted universe multiverse* indicates a repository for Great Britain (gb).

## **5.5 Install Software Packages**

### **5.5.1 List of installed software packages**

To visualize the list of installed packages, we use the following command

```
ubuntu@ubuntu:~$ dpkg-query -W
```

To visualize an installed package whose name is **vim** for example, we use the previous command and a redirection (or pipeline) to the *grep* command. Practically, we have:

```
ubuntu@ubuntu:~$ dpkg-query -W |grep -i vim
vim      2:7.2.079-1ubuntu5
vim-common      2:7.2.079-1ubuntu5
vim-runtime     2:7.2.079-1ubuntu5
vim-tiny        2:7.2.079-1ubuntu5
ubuntu@ubuntu:~$
```

### **5.5.2 Description of installed software packages**

The description of the set of installed packages is obtained via the command

```
ubuntu@ubuntu:~$ dpkg -l
```

### **5.5.3 List of available software packages**

The list of available packages is obtained as follows:

```
ubuntu@ubuntu:~$ apt-cache pkgnames
```

We clearly see that the list is not in alphabetical order. To resolve this issue, we redirect and sort the output:

```
ubuntu@ubuntu:~$ apt-cache pkgnames |sort
```

#### 5.5.4 Searching a software package: apt-cache search

To search a software package, we proceed as follows:

```
ubuntu@ubuntu:~$ apt-cache search wget
devscripts - scripts to make the life of a Debian Package maintainer easier
emacs-goodies-el - Miscellaneous add-ons for Emacs
wget - retrieves files from the web
abcde - A Better CD Encoder
apt-mirror - APT sources mirroring tool
apt-zip - Update a non-networked computer using apt and removable media
axel - light download accelerator - console version
axel-dbg - light download accelerator - debugging symbols
axel-kapt - light download accelerator - graphical front-end
epiphany-extension-gwget - Gwget extension for Epiphany web browser
gtm - Multiple files transfer manager
gwget - GNOME front-end for wget
mirror - keeps FTP archives up-to-date
puf - Parallel URL fetcher
snarf - A command-line URL grabber
wget-el - an interface for wget on Emacsen
wput - A tiny wget-like ftp-client for uploading files
ubuntu@ubuntu:~$
```

This command displays software packages containing the expression **wget**.

### **5.5.5 Properties and information of a software: apt-cache show**

To display information related to a package, for instance the **gwget** package, we proceed as follows:

```
ubuntu@ubuntu:~$ apt-cache show gwget
Package: gwget
Priority: optional
Section: universe/gnome
Installed-Size: 1336
Maintainer: Ubuntu MOTU Developers
Original-Maintainer: Arnaud Fontaine
Architecture: armel
Source: gwget2
Version: 1.0.1-0ubuntu1
Replaces: gwget2
Provides: gwget2
Depends: libart-2.0-2 (>= 2.3.18), libatk1.0-0 (>= 1.20.0), libbonobo2-0 (>= 2.15.0), libbonoboui2-0 (>= 2.15.1), libc6 (>= 2.4), libcairo2 (>= 1.2.4), libdbus-1-3 (>= 1.0.2), libdbus-glib-1-2 (>= 0.78), libfontconfig1 (>= 2.4.0), libfreetype6 (>= 2.2.1), libgconf2-4 (>= 2.13.5), libglade2-0 (>= 1:2.6.1), libglib2.0-0 (>= 2.16.0), libgnome2-0 (>= 2.17.3), libgnomecanvas2-0 (>= 2.11.1), libgnomeui-2-0 (>= 2.22.0), libgnomevfs2-0 (>= 1:2.17.90), libgtk2.0-0 (>= 2.16.0), libice6 (>= 1:1.0.0), libnotify1 (>= 0.4.4), libnotify1-gtk2.10, liborbit2 (>= 1:2.14.10), libpango1.0-0 (>= 1.14.0), libpixman-1-0, libpng12-0 (>= 1.2.13-4), libpopt0 (>= 1.14), libsm6, libx11-6, libxcb-render-util0 (>= 0.2.1+git1), libxcb-render0, libxcb1 (>= 1.1.92), libxml2 (>= 2.6.27), libxrender1, zlib1g (>= 1:1.1.4), gconf2 (>= 2.10.1-2), wget (>= 1.10)
Conflicts: epiphany-extension-gwget (<< 0.97-1), gwget2
Filename: pool/universe/g/gwget2/gwget_1.0.1-0ubuntu1_armel.deb
```

```
Size: 230510
MD5sum: 45aad4ccc0f0d9c657ec588d14ec73d4
SHA1: e8f95b0957af34be40c595ee9550508afaf8a5c8
SHA256:
942cfb10fd3d2f6f7517102e461ebd15cf6d495703c8e8a17338d1871d259e44
Description: GNOME front-end for wget
  Gwget offers a GNOME front-end to the popular wget application, with
  enhanced features, such as systray icon, multiple downloads and a
  powerful preferences manager.
Homepage: http://gnome.org/projects/gwget/
Bugs: https://bugs.launchpad.net/ubuntu/+filebug
Origin: Ubuntu

ubuntu@ubuntu:~$
```

Note that the exact name of the package should be entered! For example,

```
ubuntu@ubuntu:~$ apt-cache show flash
W: Unable to locate package flash
E: No packages found
ubuntu@ubuntu:~$
```

This does not yield any result since **flash** is not the name of a package.

### **5.5.6 Installing a software: *apt-get install***

Assume we want to install **firefox**. We type:

```
ubuntu@ubuntu:~$ sudo apt-get install firefox
```

### **5.5.7 Removing a software: *apt-get remove***

To uninstall a package, we can proceed in two ways. Either 1) we uninstall only the software or 2) the software and its configuration files.

In the first case, assuming we want to uninstall the **mplayer** software, we type:

```
ubuntu@ubuntu:~$ sudo apt-get remove mplayer
```

In the second case, if we want to uninstall both **mplayer** and its **configuration files**, we type:

```
ubuntu@ubuntu:~$ sudo apt-get remove --purge mplayer
```

### **5.5.8 Updating the software list: *apt-get update***

Updating a list of software is not the same as updating the software *per se*.

Only the list is actually updated:

```
sudo apt-get update
```

### **5.5.9 Upgrading the software: *apt-get upgrade***

The **upgrade** command installs the most recent versions of all packages on the system.

```
ubuntu@ubuntu:~$ sudo apt-get upgrade
```

To simulate an update installation, i.e. to see which software will be updated, we type:

```
ubuntu@ubuntu:~$ sudo apt-get -s upgrade
```

#### **5.5.10 Smart software update: *apt-get dist-upgrade***

The ***dist-upgrade*** command has the same effect as the 'upgrade' except that a smart management is used in changes of dependencies in new versions: conflict resolution and discarding less important packages for more important ones, for example.

```
ubuntu@ubuntu:~$ sudo apt-get dist-upgrade
```



## **5.6 COM Port**

There are six RS232 ports and one RS422/485 port on MXM-6410 evaluation kit. Four of them (CN20 and CN21) are from MXM-6410 and two of them (CN23) are from external DUART chip TL16C752B via system bus of MXM-6410. This is also a good example to tell users how to add an external chip via system bus. The pin 1-10 of CN20 (UART0) also plays the role of debug port. The RS422/485 port (CN22) share with one of the RS232 (pin 11-20 of CN20) by JP3 jumper selection. That means, RS232 from pin 11-20 of CN20 and RS422/585 (CN22) cannot be used simultaneously. The device descriptor of COM ports is as follows.

### **5.6.1 From MXM-6410**

CN20 → /dev/ttySAC0 (Console, pin 1~9) and /dev/ttySAC1 (UART1, pin 11~19)

CN21 → /dev/ttySAC2 (UART2, pin 1~9) and /dev/ttySAC3 (UART3, pin 11~19)

### **5.6.2 From external TL16C752B**

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

```
# modprobe 8250
```

```
ubuntu@ubuntu:~$ sudo modprobe 8250
ubuntu@ubuntu:~$
```

CN23 --> /dev/ttyS0 (UART4, pin 1~9) and /dev/ttyS1 (UART5, pin 11~19)

#### **Note:**

Users can use *apt-get install minicom* to use *minicom* program to test the COM port device first.

## 5.7 LCD

The device descriptor of the LCD is `/dev/fb0`. The default LCD output will be Ubuntu gdm login screen.

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

If your LCD panel is in different resolutions other than 800x480, users also need to use `fbset` command to set up the frame buffer first. You need to `apt-get install fbset` package and `fbset` the framebuffer as follows.

```
root@ubuntu:/etc/apt# sudo apt-get install fbset
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  fbset
0 upgraded, 1 newly installed, 0 to remove and 14 not upgraded.
Need to get 115kB of archives.
After this operation, 365kB of additional disk space will be used.
Get:1 http://old-releases.ubuntu.com jaunty/main fbset 2.1-23 [115kB]
Fetched 115kB in 2s (50.6kB/s)
Selecting previously deselected package fbset.
(Reading database ... 34128 files and directories currently installed.)
Unpacking fbset (from .../fbset_2.1-23_armel.deb) ...
Setting up fbset (2.1-23) ...

ubuntu@ubuntu:/etc/apt# fbset 800x480-0
ubuntu@ubuntu:/etc/apt#
```

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.

## **5.8 GPIO**

GPIO driver is default built in the kernel instead of driver module. Users need to create a device node first by the following way.

```
ubuntu@ubuntu:~$ sudo mknod /dev/gpioc1 c 253 0
ubuntu@ubuntu:~$
```

Below is the sample code for GPIO.

```
/*
 * This program demonstrates the control of APC-7xxx GPIO ports by using
 * device descriptor.
 *
 * Device Descriptor: /dev/gpioc1
 * 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
```

```

char *dev_desc = "/dev/gpiotl";

struct gpiotl_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 gpiotl_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 gpiotl_desc *gpio)
{
    *byte = ( (gpio->func & 0x3) << 2) |
            ( (gpio->pullup & 0x1) << 1) |
            ( (gpio->dat & 0x1) << 0) ;
}

int read_dev(unsigned char *buf)
{
    struct gpiotl_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 gpiotl_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);
}
```

## **5.9 FTP Client**

The **lftp** is default included in the root file system. You might use *apt-get install* to use a dedicated ftp client. To use the *lftp* FTP client, assuming the remote host IP address is 59.124.115.43 and the user is *eric*,

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

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

You can also use **wget** command to get the file from webserver.

## 5.10 FTP Server

The ftp server is not included in the root file system by default. However, it is very easy to have one. In this section, we would like to take the **vsftpd** server as an example.

First, we need to apt-get install the vsftpd packages.

```
ubuntu@ubuntu:~$ sudo apt-get install vsftpd
```

```
You can use netstat commad to check if the vsftpd has been successfully
installed. ubuntu@ubuntu:/etc/apt# netstat -tul |grep ftp
tcp        0      0  *:ftp                *:*
LISTEN
ubuntu@ubuntu:/etc/apt#
```

If you saw the ftp has been *LISTEN*, then means the vsftpd is running.

The default vsftpd is configured only allow the *anonymous* user to ftp in without password. We now need to configure the */etc/vsftpd.conf* file.

```
ubuntu@ubuntu:~$ sudo vim /etc/vsftpd.conf
```

Below are the common settings for vsftpd. For advanced settings, users can read the remarks from the same file.

*anonymous\_enable=NO* (YES means allow the anonymous user to login to ftp. NO means not allow anonymous user to login to ftp)

*chroot\_local\_user=YES* (Uncomment this to restrict local users to their home directories and cannot change directory to other system directory.)

*local\_umask=022* (Uncomment this will make the privilege of the upload file to 755; If comment out this line, the default privilege of upload file is 700.)

*local\_enable=Yes* (Uncomment this to allow local users to log in.)

*write\_enable=YES* (Uncomment this to enable any form of FTP write command.)

After made the changes, we need to restart the vsftpd server by

```
ubuntu@ubuntu:/etc/apt# sudo /etc/init.d/vsftpd restart
```



```
* Stopping FTP server: vsftpd
[ OK ]
* Starting FTP server: vsftpd
[ OK ]
ubuntu@ubuntu:/etc/apt#
```

And now you can use the common FTP client software like Filezilla to FTP files into the user home directory.

### **5.11 Time and RTC**

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

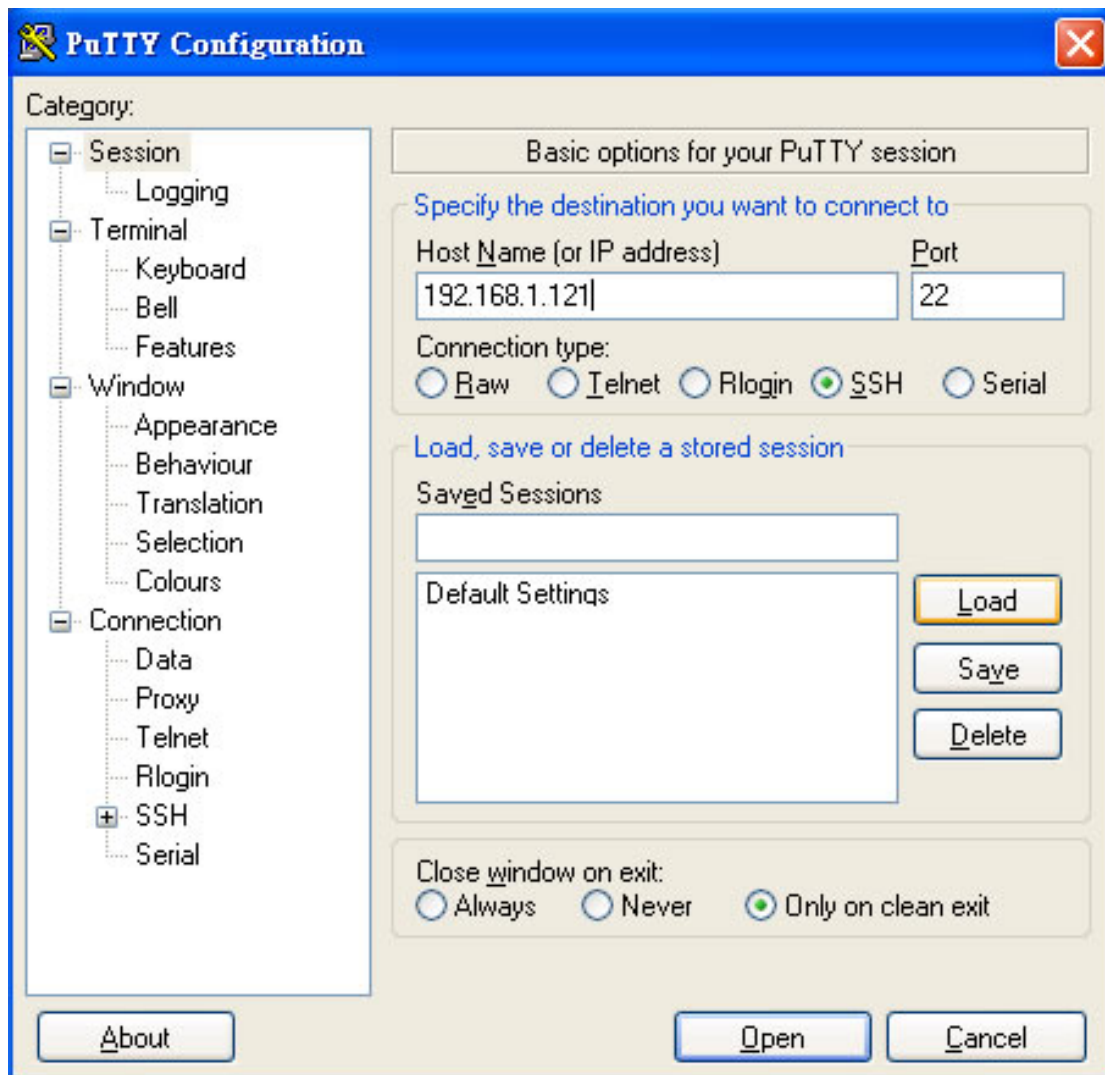
```
# date MMDDhhmmYY
```

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

```
ubuntu@ubuntu:~# date 011315362002
Sun Jan 13 15:36:00 CST 2002
ubuntu@ubuntu:~# hwclock --systohc
root@ubuntu:~#
```

### **5.12 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.



## **Embedian, Inc.**

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

```
login as: root
root@192.168.1.121's password:
Linux ubuntu 2.6.24.2 #191 PREEMPT Fri Oct 23 00:30:52 CDT 2009 armv6l

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 Oct 1 07:20:26 2013
root@ubuntu:~#
```

### 5.13 VNC Server

The VNC server allows users to see the desktop of an MXM-6410 based remote device and control it with your local mouse and keyboard and without attaching an LCD to the device, just like you would do it sitting in the front of that MXM-6410 based device.

First of all you need to install the tightVNC server first.

```
ubuntu@ubuntu:~# sudo apt-get install tightvncserver
```

Now you can start vncserver to have a new display (called :1)

```
ubuntu@ubuntu:~$ sudo vncserver -geometry 800x480
```

*You will require a password to access your desktops.*

*Password:*

*Verify:*

*Would you like to enter a view-only password (y/n)? n*

*xauth: creating new authority file /home/ubuntu/.Xauthority*

*New 'X' desktop is ubuntu:1*

*Creating default startup script /home/ubuntu/.vnc/xstartup*

*Starting applications specified in /home/ubuntu/.vnc/xstartup*

*Log file is /home/ubuntu/.vnc/ubuntu:1.log*

```
ubuntu@ubuntu:~$
```

It will create .vnc directory under /root. It also will ask you a password (insert it twice) and for the viewing password answer NO.

Now it's time to stop it for few more configurations.

```
ubuntu@ubuntu:~$ sudo vncserver -kill :1
```

*Killing Xtightvnc process ID 2527*

```
ubuntu@ubuntu:~$
```

and move to the configuration files

```
ubuntu@ubuntu:~$ sudo vim /home/ubuntu/.vnc/xstartup
```

## ***Embedian, Inc.***

commenting out everything and insert the following two lines:

- *openbox &*
- *lxsession*

```
#xrdb $HOME/.Xresources
#xsetroot -solid grey
#x-terminal-emulator -geometry 80x24+10+10 -ls -title "$VNCDESKTOP
Desktop" &
#x-window-manager &
#/etc/X11/Xsession
openbox &
lxsession
```

And now you can start vncserver again.

```
ubuntu@ubuntu:~$ sudo vncserver -geometry 800x480
```

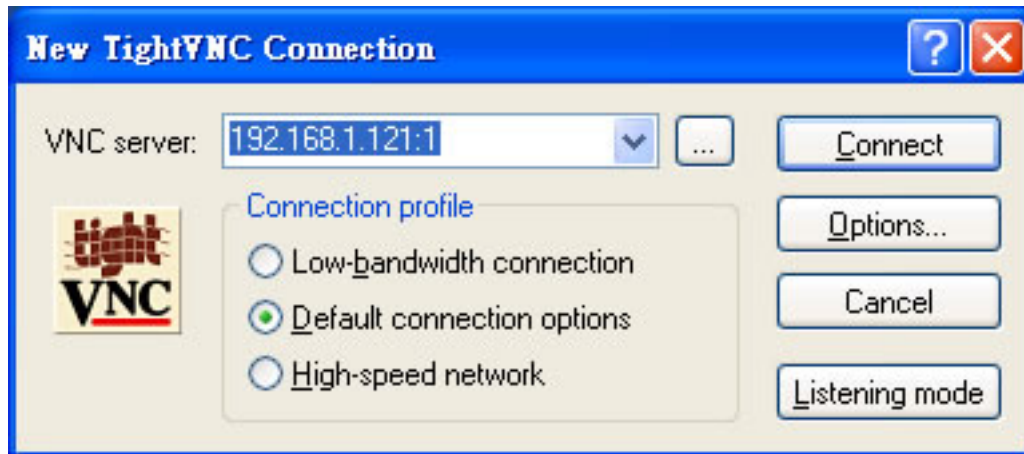
```
New 'X' desktop is ubuntu:1
```

```
Starting applications specified in /home/ubuntu/.vnc/xstartup
Log file is /home/ubuntu/.vnc/ubuntu:1.log
```

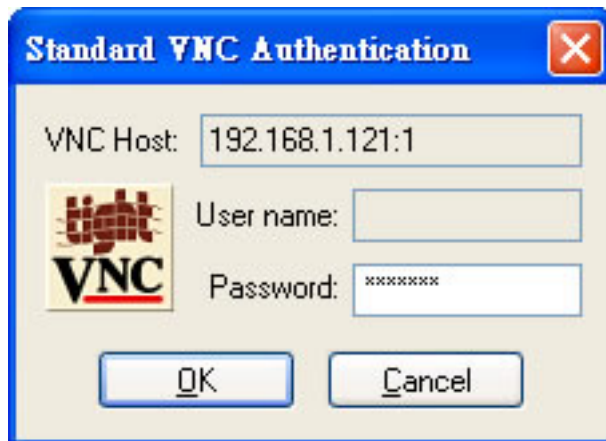
```
ubuntu@ubuntu:~$
```

At the Windows XP client side, you can download the free VNC viewer from <http://www.tightvnc.com/download.php>

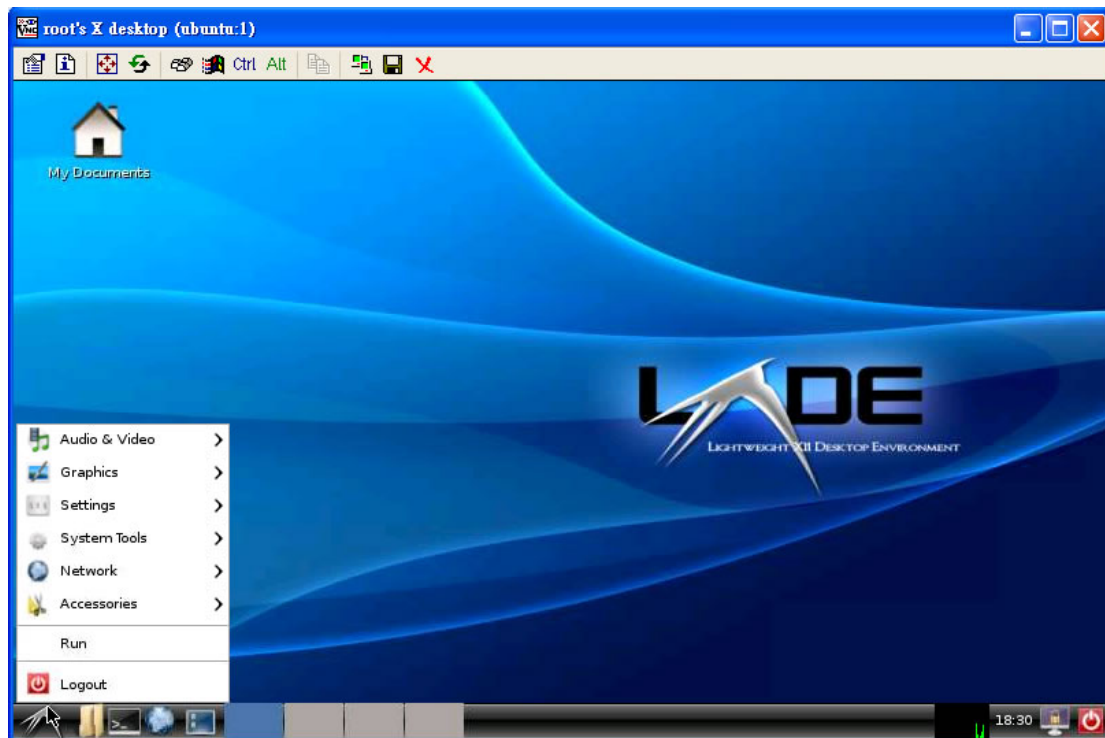
Open up the vncviewer program



Click “Connect” and it will pop out a password authentication window. Enter the password.



Click “OK” and you will see the device desktop on the remote client side.



To kill the VNC service,

```
ubuntu@ubuntu:~$ sudo vncserver -kill :1  
Killing Xtightvnc process ID 4326  
ubuntu@ubuntu:~$
```

## 5.14 GDM

Embedian MXM-6410 uses GDM (Gnome Display Manager) as the graphical login program. GDM is the GNOME Display Manager, a graphical login program that uses XWindow.

GDM provides the equivalent of a "login:" prompt for X displays- it pops up a login window and starts an X session.

It provides all the functionality of xdm, including XDMCP support for managing remote displays.

To stop the gdm service, you can use the command

```
ubuntu@ubuntu:~$ sudo /etc/init.d/gdm stop
```

You can use gdm to load different desktop environments and Window Managers. Because of the memory limitation, the default is using the LXDE system. If users would like to consider GNOME, the 256MB DDR RAM is the minimum requirement.

## 5.15 LXDE

Because of the memory limitation, Embedian chooses LXDE as default desktop environment. After login into the GDM display manager, the GDM will launch LXDE directly by default.

LXDE, *Lightweight X11 Desktop Environment*, is a desktop environment which is lightweight and fast. It is designed to be user friendly and slim, and keep the resource usage low. LXDE uses less RAM and less CPU while being a feature rich operating system. Because of the low usage of resources it also saves energy. We don't tightly integrate every component of LXDE. Instead, we try to make all components independent, so each of them can be used independently with few dependencies. More about LXDE on the [lxde.org](http://lxde.org) website, the [LXDE blogs](#), and in the [LXDE forum](#).

### LXDE Components

- [PCManFM](#), is a fast and lightweight file manager with features like Drag & Drop support, tabbed browsing (Similar to Firefox), Built-in file searching utility, fast load of large directories, File association support (Default application), Thumbnail for image files, Bookmarks support,



## ***Embedian, Inc.***

correct handling of non-UTF-8 encoded filenames and more.

- [LXLauncher](#), easy-mode application launcher
- [LXPanel](#), desktop panel, the panel can generate menu for installed applications automatically from \*.desktop files. It can be configured from GUI preference dialog, and there is no need to edit config files. The component provides a "Run" dialog with autocompletion.
- [LXSession](#), session manager, The LXSession manager is used to automatically start a set of applications and set up a working desktop environment. Moreover, the session manager is able to remember the applications in use when a user logs out and to restart them the next time the user logs in.
- [LXSession Edit](#), window manager used in LXDE can be changed, ability to turn on disabled applications
- [LXAppearance](#), theme switcher. You can change the theme, icons, and fonts used by applications easily.
- [Leafpad](#), text editor
- [Xarchiver](#), archiving
- [GPicView](#), image viewer, GPicView features lightening fast startup and intuitive interface.
- [LXTerminal](#), terminal emulator
- [LXTask](#), task manager / system monitor
- [LXNM](#), lightweight network connection helper daemon for LXDE supporting wireless connections (Linux-only)
- [Openbox](#), window manager and obconf
- [LXRandr](#), screen manager, manages screen resolution and external monitors
- [LXShortCut](#), an easy way to edit application shortcuts
- [LXMusic](#), minimalist xmms2-based music player
- [LXDE Common](#), the default settings configuration file for integrating the different components of LXDE. LXDE Common manages the system behavior and functions to integrate icons and artwork.
- [GtkNetCat](#), Graphic User Interface for netcat. Netcat provides system functions as a computer networking utility for reading from and writing to network connections on either TCP or UDP.

## Embedian Add-on Components

- **GMplayer**, is the GUI of Mplayer. It is integrated by Embedian to support the hardware MFC from CPU. Details can be found at section 5.14 and [Chapter 7](#).
- **Calibrate TouchScreen**, is a tool to calibrate the touch screen.

## 5.16 Calibration and Touch Screen

Some applications use touch panel as input device. If that is the case, users need to install the calibration program that Embedian provided first.

### 5.16.1 Install the Calibration Program

Change directory to /tmp first and copy the "ts\_upgrade.tar.gz" file into the device and extract it as shown below.

```
ubuntu@ubuntu:~# cd /tmp
ubuntu@ubuntu:/tmp# ls
ubuntu@ubuntu:/tmp# lftp eric@59.124.115.45
Password:
lftp eric@59.124.115.45:~> get ts_upgrade.tar.gz
580432 bytes transferred
lftp eric@59.124.115.45:~> bye
ubuntu@ubuntu:/tmp# ls
ts_upgrade.tar.gz
ubuntu@ubuntu:/tmp#
```

Extract this tarball and execute the shell script as follows.

```
ubuntu@ubuntu:/tmp# tar xvfz ts_upgrade.tar.gz
embedian-mxm6410-touchscreen.deb
install.sh
ubuntu@ubuntu:/tmp# ./install.sh
Reading package lists... Done
Building dependency tree
Reading state information... Done
```

Reboot, and you will see the calibration program on the screen. There are five cross points for users to calibrate. The calibration value will be stored in the NAND flash after calibrated. MXM-6410 will check if this calibration value

## **Embedian, Inc.**

exists there or not at next boot. If yes, users will not need to calibrate every time.

### **Note:**

Users need to connect a touch screen to MXM-6410 evaluation kit when install this package, or the boot up process will stop at the following boot up process to wait you input the calibration value.

```
* Starting OpenBSD Secure Shell server sshd          [ OK ]  
* Starting Hardware abstraction layer hald           [ OK ]
```

After calibration, it will bring you to the gdm login screen. To re-calibrate, users can use the following command line.

```
ubuntu@ubuntu:~# sudo ts_calibrate
```

Or pending at the touch screen for a while when booting.

The device will only ask you calibrate at first boot. After the first calibration, the calibration value will be stored. And the next boot will check if this value existing or not. If yes, the device will just use this value.

## **5.17 Kiosk Mode**

### **5.17.1 Boot Ubuntu 9.04 into text mode instead of graphic mode**

Some users' applications are relatively simple and they develop their GUIs directly on top of frame buffer (/dev/fb0). They don't even need X11 running on the device. Or some server applications don't need to have a graphic interface at all. To boot into text mode or to disable a service (such as GDM) from being started in a given runlevel (e.g., 2 which is Ubuntu's default runlevel) is like so:

```
ubuntu@ubuntu:~# sudo /etc/rc2.d/S30gdm /etc/rc2.d/K70gdm  
ubuntu@ubuntu:~# sudo shutdown now -r
```

After reboot, you will see the text mode booting into the LCD screen. You can reverse this file back to boot into graphic mode again.

If you would like launch the, for example, *mplayer* application under the text mode, you can

```
ubuntu@ubuntu:~# mplayer -quiet -fs -vo fbdev test.avi
```

And you can write a simple script to let this program to startup automatically at boot.

If you would like to stop the X11 and X application at runtime, you can

```
ubuntu@ubuntu:~$ sudo /etc/init.d/gdm stop
[sudo] password for ubuntu:
* Stopping GNOME Display Manager...
[ OK ]
ubuntu@ubuntu:~$
```

To make your services startup at boot, Ubuntu Linux use *update-rc.d* command to install and remove System-V style init script links. Let us take the *mysql* service for example.

Turn on or start service called *mysql* on boot

```
ubuntu@ubuntu:/etc$ sudo update-rc.d mysql defaults
```

Remove service called *mysql* on boot:

```
ubuntu@ubuntu:/etc$ sudo update-rc.d mysql remove
```

OR

```
ubuntu@ubuntu:/etc$ sudo update-rc.d -f mysql remove
```

Replace *mysql* name with actual service name.

**Note:**

It is not recommended to boot into text mode by using *update-rc.d remove* and *update-rc.d defaults*. “update-rc.d foo defaults” will not put “foo” back into its previous start-up slot, but puts it at S20 by braindead “default”. It’s the oldest surviving bug in UNIX history. For gdm this can cause real grief because it puts it before processes that should have completed start-up before gdm goes about its merry ways.

### **5.17.2 GDM Auto Login**

To allow Ubuntu GDM Auto Login, you need edit *gdm.conf* and add the user you want to auto login. Let’s assume the user that you would like to auto

## **Embedian, Inc.**

login is *ubuntu*.

```
ubuntu@ubuntu:~$ sudo vim /etc/gdm/gdm.conf
```

At line 48, 49

```
AutomaticLoginEnable=true  
AutomaticLogin=ubuntu
```

Reboot the device will automatically login into the lxde desktop.

### **5.17.3 Auto Start a Program under LXDE**

To auto start a program under LXDE, users can edit */etc/xdg/lxsession/LXDE/autostart* file and add the program that you would like to auto start at the end of this file. Let's take the *firefox* program for example.

```
@lxde-settings  
@xscreensaver -no-splash  
@lxpanel --profile LXDE  
@pcmanfm -d  
@firefox -width 800 -height 480
```

After reboot, you will see the Firefox program auto start at the top of the panel.

#### **Note:**

If users don't even need LXDE to auto start the program, you will need to modify the */etc/alternatives/x-session-manager* to auto start the program and comment out the LXDE auto start script.

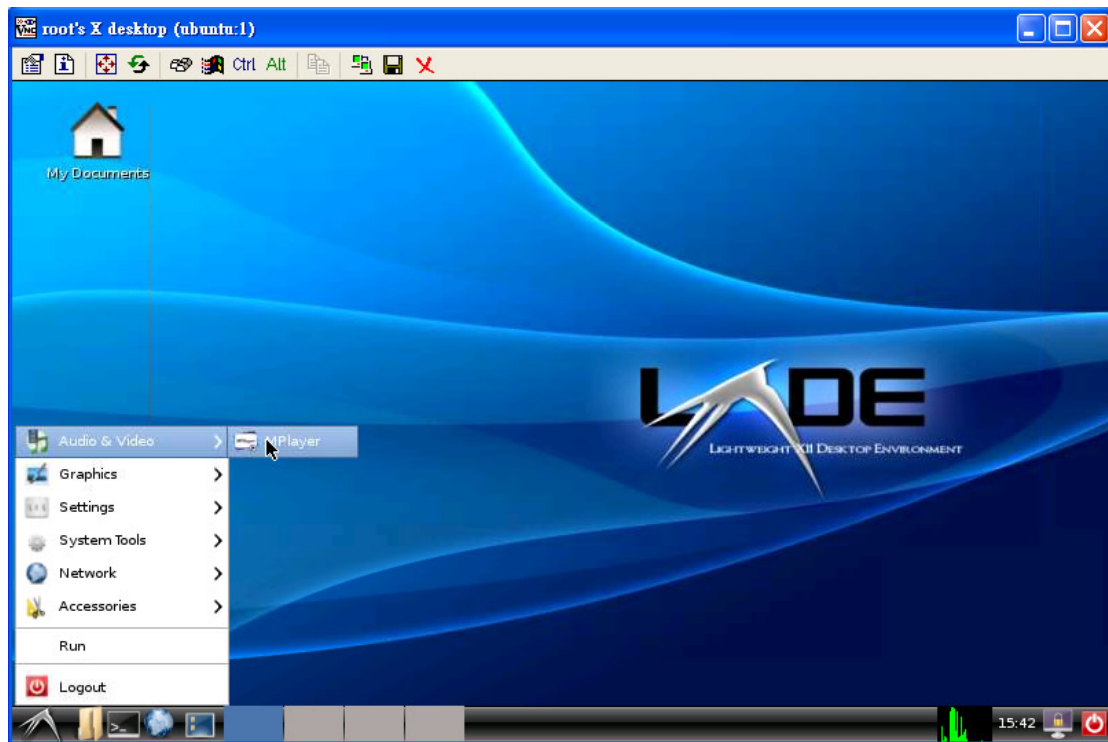
### **5.18. MPEG4 Decoder for Mplayer at Device**

Embedian integrates the CPU MFC hardware codec into Mplayer and natively supports MPEG4 DivX **DX50** and MPEG4 **Xvid** standard. Any video files that generated by a software encoder that supports these standards can be

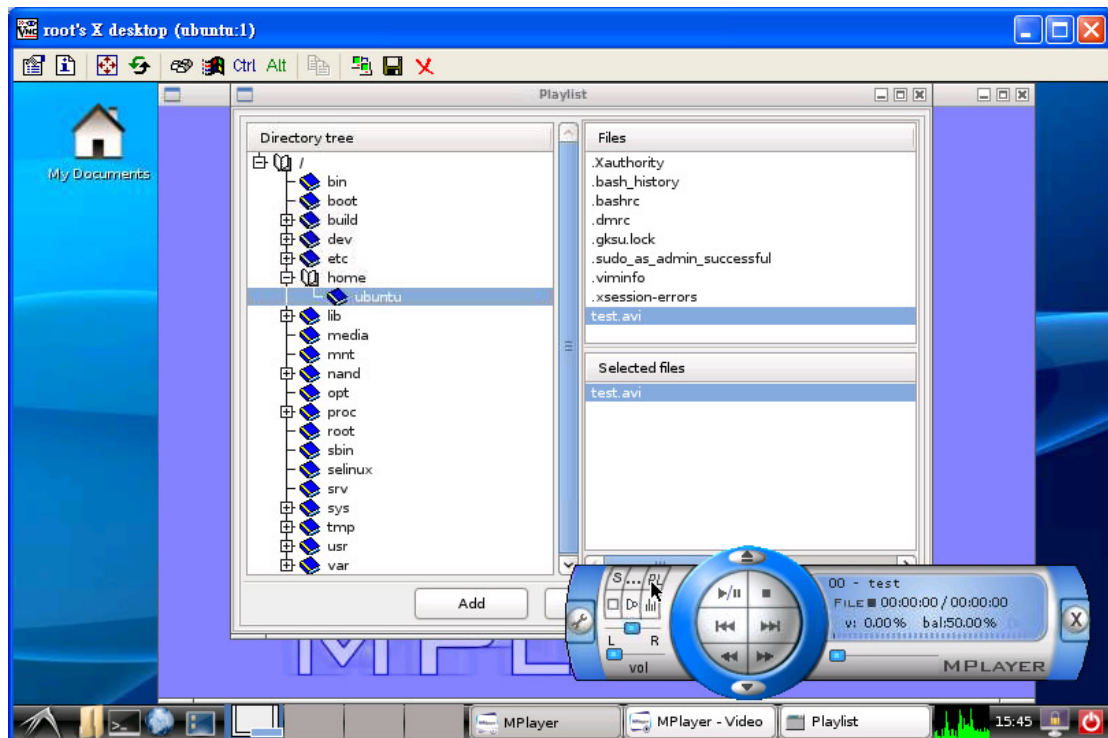
decoded by the hardware decoder supported Mplayer.

#### **5.18.1. Mplayer running on top of GDM and LXDE**

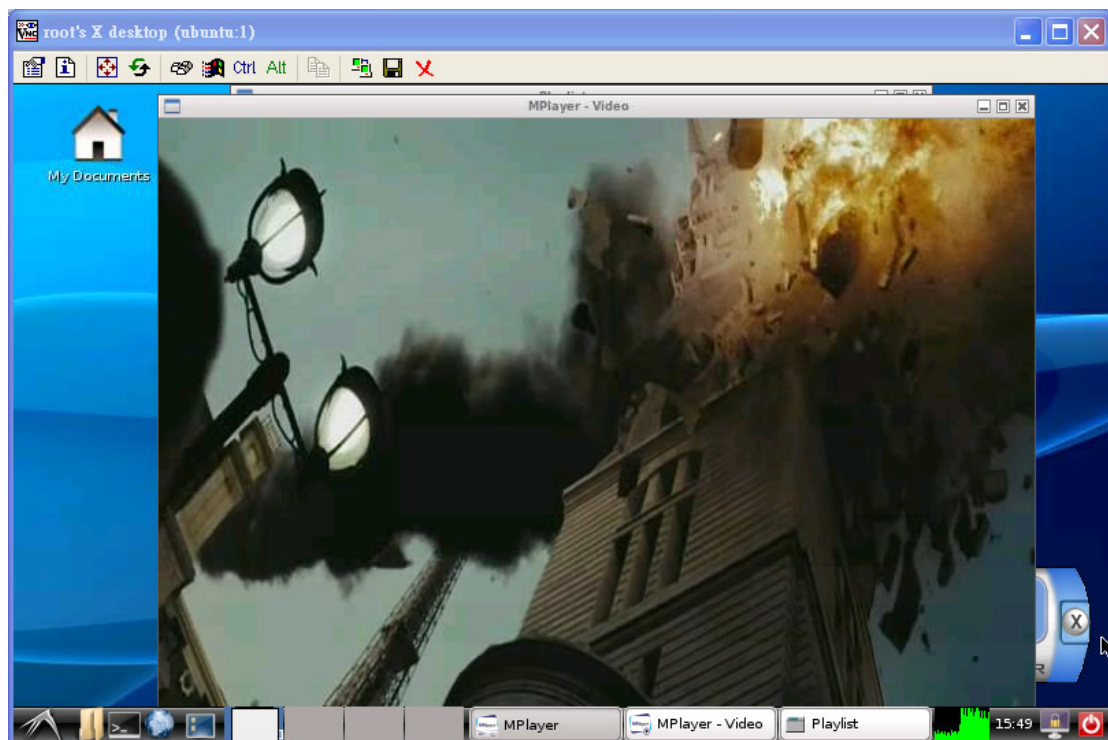
At “Start” → “Audio & Video”, Click “Mplayer”



Add the file to the play list,



Click the triangle sign on panel to play the video.



### **5.18.2. Mplayer Running on top of framebuffer directly**

The previous example of Mplayer is running on top of X. Mplayer can also be played directly on top of frame buffer. If that is the case,

```
ubuntu@ubuntu:~$ sudo /etc/init.d/gdm stop
[sudo] password for ubuntu:
* Stopping GNOME Display Manager...
[ OK ]
ubuntu@ubuntu:~# mplayer -quiet -fs -vo fbdev test.avi
```

**Note:**

-quite parameter is highly recommended.

-fs means full screen

-vo to fbdev means to show video on top of frame buffer directly.

## 5.19 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.

### 5.19.1 linuxrc

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

The use of *linuxrc* provides with the choices to boot into a small root file system in NAND or the Ubuntu 9.04 system in SD card. (If no press anything, the default is set to boot into SD card.)

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

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

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



```
ubuntu@ubuntu:/nand$
```

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

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

### **5.19.2 As a rescue file system**

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

1. Boot into NAND flash first by pressing 2) *NAND* during booting process and login as *root* privilege. (The root pass is **apc7110** by default.)
2. Prepare for a at least 1GB SD/SDHC card. The SDHC card will have better read/write performance, but usually the SDHC card is at least 4GB.
3. The NAND file system will mount partition 1 of SD card by default (The device descriptor of SD device is */dev/mmc0*, and the partition 1 of SD card is */dev/mmc1*). Here would like to format the partition 1 of SD card as EXT3 first.

```
[root@apc7110 /]# mkfs -t ext3 /dev/mmc1
mke2fs 1.37 (21-Mar-2005)
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
488640 inodes, 977263 blocks
```

```

48863 blocks (5.00%) reserved for the super user
First data block=0
30 block groups
32768 blocks per group, 32768 fragments per group
16288 inodes per group
Superblock backups stored on blocks:
    32768, 98304, 163840, 229376, 294912, 819200, 884736

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

This filesystem will be automatically checked every 38 mounts or
180 days, whichever comes first. Use tune2fs -c or -i to override.
[root@apc7110 /]#

```

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

```

[root@apc7110 /]# mount -t ext3 /dev/mmc1 /mnt
kjournald starting. Commit interval 5 seconds
EXT3 FS on hda1, internal journal
EXT3-fs: mounted filesystem with ordered data mode.
[root@apc7110 /]# cd /mnt
[root@apc7110 /mnt]#

```

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

```

[root@apc7110 /mnt]# ftp 192.168.1.10
Connected to 192.168.1.10.
220 (vsFTPD 2.2.2)
Name (192.168.1.10:root): eric
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> get u904.090930.tar.gz
200 PORT command successful. Consider using PASV.

```

## **Embedian, Inc.**

```
150 Opening BINARY mode data connection for u904.090930.tar.gz (219759103
bytes).
226 Transfer complete.
219759103 bytes received in 77.1 seconds (3124731 bytes/s)
ftp>
```

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

```
[root@apc7110 /mnt]# tar xvfz u904.090930.tar.gz
```

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

```
[root@apc7110 /mnt]# rm u904.090930.tar.gz
[root@apc7110 /mnt]# cd ../
```

8. *umount /mnt*

```
[root@apc7110 /]# umount /mnt
[root@apc7110 /]#
```

9. *Reboot*

```
[root@apc7110 /mnt]# cd ../
[root@apc7110 /]# umount /mnt
[root@apc7110 /]# shutdown now -r
Why?
```

You will see the gdm login screen appear on the LCD screen.

### **6.19.3 As a small root file system**

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

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

your program at SD Ubuntu file system.

## **5.20 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. User can get the gcc 4.2 at device from Ubuntu repository by

```
ubuntu@ubuntu:~$ sudo apt-get install gcc
```

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

The cross toolchain version that we are using is 4.2.2 with EABI supported. The file name is cross-4.2.2-eabi.tar.bz2 that can be downloaded from Embedian FTP site.

### **5.20.1 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 cross-4.2.2-eabi.tar.bz2
```

The above command will generate the **4.2.2-eabi** folder under the same directory as you made the commands. Move this folder to **/usr/local/arm** directory.

## **Embedian, Inc.**

```
# mv 4.2.2-eabi /usr/local/arm/  
# export PATH=$PATH:/usr/local/arm/4.2.2-eabi/bin
```

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

```
CROSS_COMPILE = /usr/local/arm/4.2.2-eabi/bin/arm-linux-
```

### **5.20.2 Build Uboot**

1. # Extract *u-boot-6410\_090930.tar.gz* file.
2. # *make clean*
3. # *make smdk6410\_config*
4. # *make*

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

### **5.20.3 kernel zImage**

1. Extract Linux source code *linux-2.6.21-04132009.tar.bz2* that released from Embedian
2. # *tar xvfj linux-2.6.21-04132009.tar.bz2*
3. # *cd linux-2.6.21*
4. # *gzip -d -c ../patch-04032009-0056.gz | patch -p0*
5. # *make mxm6410\_defconfig*

# Chapter

# 6

## **Backup and Restore the Root File System in SD Card**

This Chapter details how to backup and restore the root file systems in SD card of MXM-6410 evaluation kit.

Section include :

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

## Chapter 6 Backup and Restore the Root File System in SD Card

This chapter is mainly for Linux users. For Windows CE 6.0 users, NK is stored in NAND flash and users can neglect this chapter.

This chapter gives an instruction in regarding to how to backup and restore the root file systems in SD card. First, we would like to detail how to backup the root file system in SD card and next, we would like to tell you how to restore the root file system in SD card. This chapter uses MXM-6410 on the evaluation kit as an example.

### 6.1 Backup the root file system in SD card

After developing your program under the Ubuntu Jaunty Jackalope (Ubuntu 9.04), 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 SD/SDHC card off from the MXM-6410 evaluation kit and plug it into a USB SD/SDHC 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 SDHC card storage is 8GB.

Use the `# fdisk -l` command to list your disk information and find the device descriptor of you SD 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: 7948 MB, 7948206080 bytes
81 heads, 10 sectors/track, 19165 cylinders
Units = cylinders of 810 * 512 = 414720 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00000000

```

Device	Boot	Start	End	Blocks	Id	System
/dev/sdb1		1	19165	7761820	83	Linux

```

root@dns3:~#

```

We can see the device descriptor of the USB SD 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 SD/SDHC card to /mnt directory and change directory to the /mnt.

```

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

```

You can ls the file structure.

```

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

```

Next, tar the file system into a file. (The file name in this example is u904.090930\_backup.tar.gz)

```

root@dns3:/mnt# tar cvfz u904.090930_backup.tar.gz
root@dns3:/mnt#

```

You have backup the SD/SDHC root file systems as a file name "320rootfs\_backup.tar.gz"!



***Embedian, Inc.***

## 6.2 Restore the root file system in SD card

Plug a SD/SDHC 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 SDHC card storage is 8GB. (Note: 2GB is minimal requirement for the Embedian official root file system.)

Use the `# fdisk -l` command to list your disk information and find the device descriptor of you SD/SDHC 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: 7948 MB, 7948206080 bytes
81 heads, 10 sectors/track, 19165 cylinders
Units = cylinders of 810 * 512 = 414720 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00000000

   Device Boot      Start         End      Blocks   Id  System
/dev/sdb1             1         19165     7761820    83   Linux

root@dns3:~#
```

We can see the device descriptor of the USB SD/SDHC 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 SD/SDHC card, you have to use `fdisk` to partition

## **Embedian, Inc.**

it first, here we partitioned the SD card as one partition. (New SD/SDHC card should have one partition already by default.)

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

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

Command (m for help): d
Selected partition 1

Command (m for help): n
Command action
    e   extended
    p   primary partition (1-4)
p
Partition number (1-4): 1
First cylinder (1-19165, default 1):
Using default value 1
Last cylinder, +cylinders or +size{K,M,G} (1-19165, default 19165):
Using default value 19165

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 SD/SDHC card as ext3 file system by using `# mkfs -t ext3 /dev/sdb1` command. (In FC, you can also use `# mkfs.ext3 /dev/sdb1` command.)

```
root@dns3:~# mkfs -t ext3 /dev/sdb1
```

```

mke2fs 1.41.14 (22-Dec-2010)
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
Stride=0 blocks, Stripe width=0 blocks
485760 inodes, 1940455 blocks
97022 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=1988100096
60 block groups
32768 blocks per group, 32768 fragments per group
8096 inodes per group
Superblock backups stored on blocks:
    32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632

Writing inode tables: done
Creating journal (32768 blocks): done
Writing superblocks and filesystem accounting information: done

This filesystem will be automatically checked every 25 mounts or
180 days, whichever comes first. Use tune2fs -c or -i to override.
root@dns3:~

```

And next, mount SD/SDHC 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 `u904.090930.tar.gz` into `/mnt` directory and extracting the root file system file into this directory.

```

root@dns3:/mnt# ls
lost+found  u904.090930.tar.gz
root@dns3:/mnt#

```

## **Embedian, Inc.**

You can *ls* the file structure now.

```
root@dns3:/mnt# ls
bin    dev    lib          mnt    proc    selinux  tmp          var
boot   etc    lost+found  nand   root    srv      u904.090930.tar.gz
build  home  media       opt    sbin    sys      usr
root@dns3:/mnt#
```

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

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

Take the SD/SDHC card off from the card reader and put the SD/SDHC card back to SBC and boot. You are done!

### **Additional Packages**

If users attached a touch screen LCD, additional packages for touch needed to be installed. Plug the above SD/SDHC card into MXM-6410 evaluation kit and power on and login to the device.

Change directory to */tmp* first and copy the “*ts\_upgrade.tar.gz*” into the device.

```
root@ubuntu:~# cd /tmp
root@ubuntu:/tmp# ls
root@ubuntu:/tmp# lftp eric@59.124.115.45
Password:
lftp eric@59.124.115.45:~> get ts_upgrade.tar.gz
580432 bytes transferred
lftp eric@59.124.115.45:~> bye
root@ubuntu:/tmp# ls
ts_upgrade.tar.gz
root@ubuntu:/tmp#
```

Extract this tarball and execute the shell script as follows.

```
root@ubuntu:/tmp# tar xvfz ts_upgrade.tar.gz
embedian-mxm6410-touchscreen.deb
install.sh
root@ubuntu:/tmp# ./install.sh
Reading package lists... Done
Building dependency tree
Reading state information... Done
```

Reboot, and you will see the calibration program on the screen. There are five cross points for users to calibrate. The calibration value will be stored in the NAND flash after calibrated. MXM-6410 will check if this calibration value exists there or not at next boot. If yes, users will not need to calibrate every time.

**Note:**

Users need to connect a touch screen to MXM-6410 evaluation kit when install this package, or the boot up process will stop to wait you input the calibration value.

After calibration, it will bring you to the gdm login screen. To re-calibrate, users can use the following command line.

```
# ts_calibrate
```

Or user can be pending at the screen for a few seconds during boot process to re-calibrate. In this way, no command line needed if users would like to do re-calibration.

# Chapter 7

## Using Windows CE 6.0

This Chapter details how to use the Windows CE 6.0 of MXM-6410 computer on module.

Section include :

- EBOOT
- Windows CE 6.0
- Configure LCD parameters for different kinds of LCDs

# Chapter 7 Using Windows CE 6.0

This Chapter gives an overall guide in regarding to the Windows CE 6.0 features and how to use the Windows CE 6.0 system that Embedian provided with for MXM-6410 computer on module. The first is to introduce the general features. Next, we will introduce EBOOT menu configuration and the third part is to introduce the Windows CE 6.0 systems. The last section will teach users how to set the parameters for different kinds of LCDs without re-building from the sources or PB (Platform Builder).

Before going on, users need to install the “wceusbsh” USB drivers that Embedian provided with on your Windows PC first. It is the Windows CE USB RNDIS driver that is required for active sync. (When plug the USB cable to the Windows PC at first time, Windows will ask you to install the driver. Just point to the directory where your USB driver placed and the installation will be done.) Connect the console port (UART0, CN20) to your Windows PC first. Make sure that the jumpers are properly configured.

The examples in this chapter are mainly working together with MXM-6410 evaluation kit.

## 7.1 General Features

### 7.1.1 Board Support Package (BSP)

The Embedian Board Support Package for Windows CE 6.0 is one of the most advanced BSPs available on the market. Beside the standard Windows CE functionality, it includes a large number of additional drivers as well as optimized versions of standard drivers.

The Embedian bootloader and BSP were designed to be very configurable. This relieves the application-developer from the burden of creating an own image. Instead the necessary adaptations can be done by registry settings and adding files to the on-board flash file system.

The BSP is available at Embedian FTP site at <ftp://ftp.embedian.com/pub/MXM-6410/WINCE60/>.

Customers who follow the readme instruction in the BSP folder for Microsoft's Platform Builder will enable you to build exactly the same Windows CE image as Embedian provides by default.



### ***7.1.2 Drivers***

The Embedian Board Support Package for Windows CE 6.0 is one of the most advanced BSPs available on the market. Beside the standard Windows CE function

The following drivers are integrated in the standard image that comes pre-installed with each MXM-6410 board and his evaluation kit.

**Table 7.1.1 Drivers**

<b>Table 7.1.1</b>	<b>Drivers</b>
<b>Driver</b>	<i>Description</i>
<b>COM1</b>	<i>Support RXD/TXD/CTS/RTS</i>
<b>COM2</b>	<i>Support RXD/TXD</i>
<b>COM3</b>	<i>Support RXD/TXD</i>
<b>COM4</b>	<i>External UART that supports the full RS232 specification with all 9 signals.</i>
<b>COM5</b>	<i>External UART that supports the full RS232 specification with all 9 signals.</i>
<b>Ethernet</b>	<i>10/100Mbit driver for Davicom DM9000B</i>
<b>USB Host</b>	<i>4 Ports supporting mass storage devices (USB-stick, hard disc, CD-ROM...) and other devices as keyboard, mouse, USB hub...</i>
<b>USB OTG</b>	<i>Can be used as ActiveSync connection, mass storage or RNDIS</i>
<b>Audio</b>	<i>16Bit stereo output (up to 48kHz), mic input, line-In</i>
<b>Touch Screen</b>	<i>All 4-wire resistive screens supported</i>
<b>Display</b>	<i>All types of displays supported. Easy configuration in EBOOT</i>
<b>SD/SDHC</b>	<i>Memory cards, Wireless LAN, Bluetooth, GSM, GPS ...</i>
<b>CF</b>	<i>Memory cards, Wireless LAN, modem, Bluetooth, serial card, GPS, GSM, ...</i>
<b>GPIO</b>	<i>12 Configurable GPIO with Sample codes. Easy configuration at Eboot and NK</i>
<b>Flash File System</b>	<i>HIVE-based registry. Standard FAT as well as TFAT (transaction safe FAT) are supported</i>
<b>2D Engine</b>	<i>CPU 2D acceleration h/w supported</i>
<b>3D Engine</b>	<i>Support CPU 3D h/w acceleration. Support OpenGL ES 1.1/2.0 and follow the Khronos naming rules</i>
<b>TV-Out</b>	<i>TV output driver for NTSC/PAL format is supported</i>
<b>Camera</b>	<i>Camera drivers for various camera modules</i>
<b>MFC Engine</b>	<i>Support h/w H.263, H.264, VC1</i>
<b>SPI</b>	<i>High speed SPI supported</i>
<b>RTC</b>	<i>Real time clock driver</i>

### **7.1.3 Services**

The following services are integrated in the standard image that comes pre-installed with each MXM-6410 board. User can add their application oriented service from the BSP as well.

**Table 7.1.2 Services**

<b>Table 7.1.2</b>	<b>Services</b>
<b>Service</b>	<b>Description</b>
<b>FTP Server</b>	Access device using the FTP File Transfer Protocol
<b>Telnet Server</b>	Access device using the Telnet protocol
<b>HTTP Server</b>	Access device using the HTTP protocol
<b>File Server</b>	Access device using the Samba protocol
<b>Time Server / Client</b>	Synchronize time with the network using the NTP protocol
<b>Shell</b>	Allow MXM-6410 to run batch file

### **7.1.4 Special Features**

In addition to the standard Windows CE functionality, Embedian has extended the operating system by following features:

**Table 7.1.3 Special Features**

<b>Table 7.1.3</b>	<b>Special Features</b>
<b>Special Features</b>	<b>Description</b>
<b>EBOOT</b>	Configuration through RS232/USB, clear flash registry, download image to RAM, download image to Flash, format flash, set/save configured structures, define initial GPIO configuration and set LCD parameters.  Debug Functionality: erases flash range, write/read 32bit value at address.
<b>Direct SD Boot</b>	Boot Directly from SD card. It is very useful for maintenance purposes.
<b>LCD Customizer</b>	Set all types of LCD resolution parameters in EBOOT and pass to NK.
<b>Splash Screen</b>	Display default start-up screen while Windows CE is booting.

<b>ActiveSync</b>	<i>Support Microsoft Active Sync. via USB 2.0 or serial port</i>
<b>Hive-Based Registry</b>	<i>Support Hive-Based Registry</i>
<b>AutoStart</b>	<i>Automatically execute programs upon system boot or upon removable media insertion</i>
<b>AutoCopy</b>	<i>Automatically copy files to the RAM file system upon system boot or upon removable media insertion. This, for example, can be used to add items to the Windows CE START menu button.</i>
<b>GPIO Customizer</b>	<i>Each GPIO can be configured to act as input, output, high, low or an interrupt source at EBOOT and NK.</i>
<b>Flushing Registry on Changes</b>	<i>Registry will be saved to Flash immediately on changes. Registry will be kept anytime even when power failure.</i>
<b>SplashScreen Customizer</b>	<i>Show your own screen while Windows CE is booting</i>
<b>NAND Flash Writer</b>	<i>To recover the firmware in NAND flash from NOR flash or SD by jumper setting.</i>
<b>Registry Editor</b>	<i>To edit registry locally or remotely</i>
<b>Remote Display</b>	<i>To see device desktop and control it without panel attached</i>

## **7.2 EBOOT**

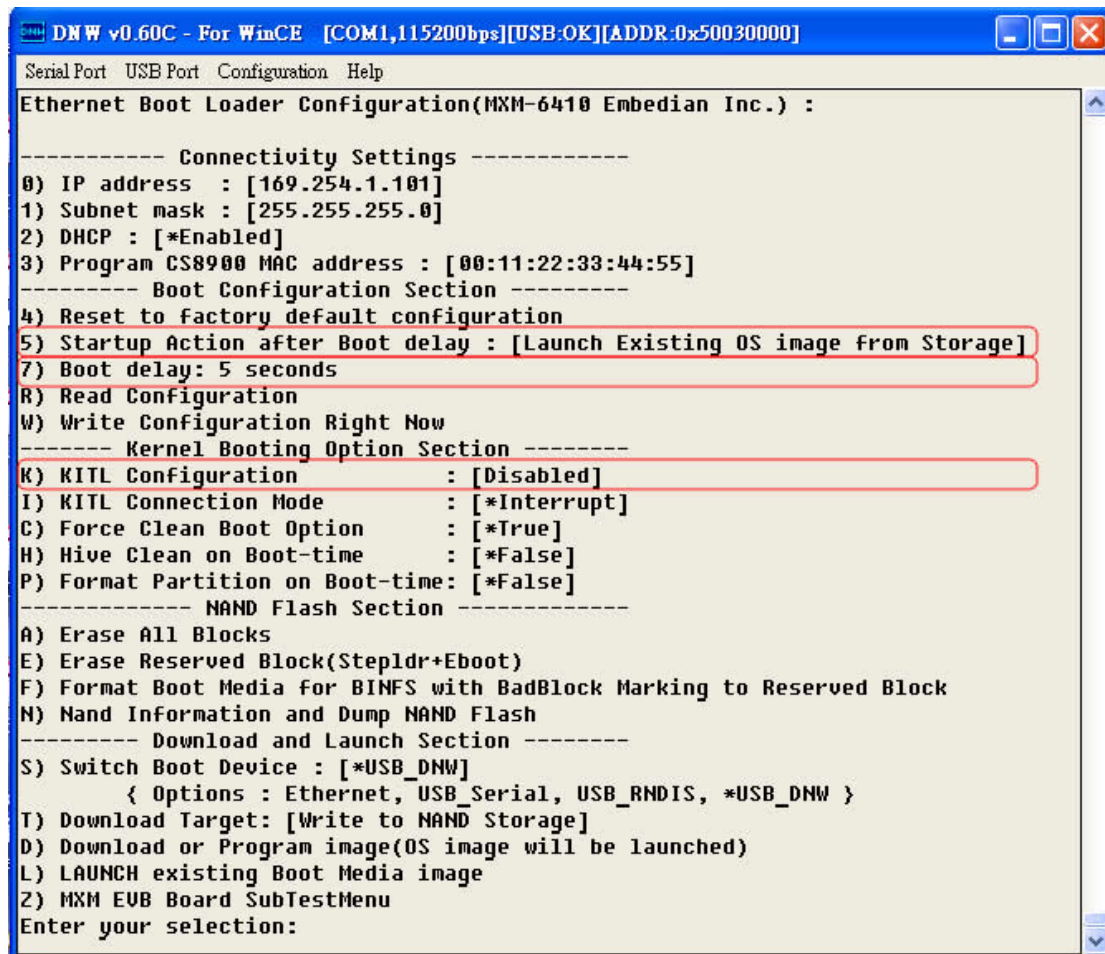
Turn on the power of the MXM-6410 evaluation kit. You should be able to hear a short beep. If you attached an LCD to the evaluation kit, you will be able to see the splash screen as in figure 7.2.1.

***Figure 7.2.1 EBOOT Splash Screen***



To enter EBOOT menu configuration, press [**Space**] bar of your keyboard when booting. And you will see the following screen as shown in figure 7.2.2.

**Figure 7.2.2 EBOOT Menu Configurations**



You need to make sure that the following two items have been configured correctly.

### **First**

Press “5” to change from

“5) Startup Action after Boot delay : [\*Download New image]”

to

“5) Startup Action after Boot delay : [Launch Existing OS image from Storage]”

This is to tell EBOOT to load the Windows CE 6.0 NK.bin image from NAND flash to DDR RAM.

### **Second**

Press “K” to disable the KITL configuration. That is to change from

“K) KITL Configuration : [\*Enabled]”

to

“K) KITL Configuration : [Disabled]”

## ***Embedian, Inc.***

Users can also set up the boot delay interval between EBOOT and NK by pressing “7”.

### **Third**

Before exit EBOOT menu, remember to press “W” to save the new configuration.

The Ethernet connectivity settings in EBOOT menu is for setting the network parameters for Ethernet download images from Platform Builder. User need to choose Ethernet for both download and transport protocols in their PB target connectivity options. And click on the setting button next to download and select the device when it shows up in the “Active Target Devices” window. But Embedian suggests users use USB to download images because of the download speed is faster. The IP address setting here will not be the same as Windows CE 6.0 NK image that is stored as HIVE registry.

After saving the EBOOT configuration by pressing “W”, users can reset the system or press “L” to launch the Windows CE 6.0 NK.bin image from NAND flash.

If you have ActiveSync program (user can download this program for free from Microsoft’s website) installed in your Windows PC and the USB cable is connected to the evaluation kit, you should be able to see the ActiveSync program pop out and ask you to set up a partnership during loading the NK image as shown in figure 7.2.3.

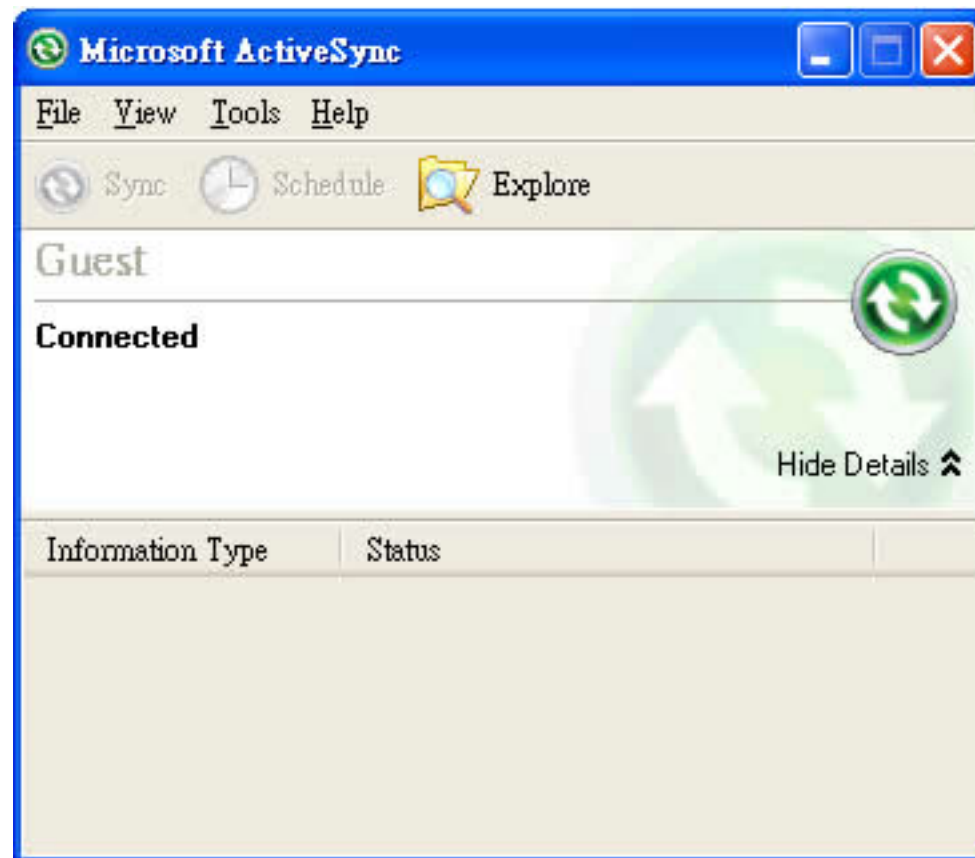
**Figure 7.2.3 ActiveSync Program**



Select “**No**” and click “**Next**” of your ActiveSync program. You will see the screen shown as in figure 7.2.4.

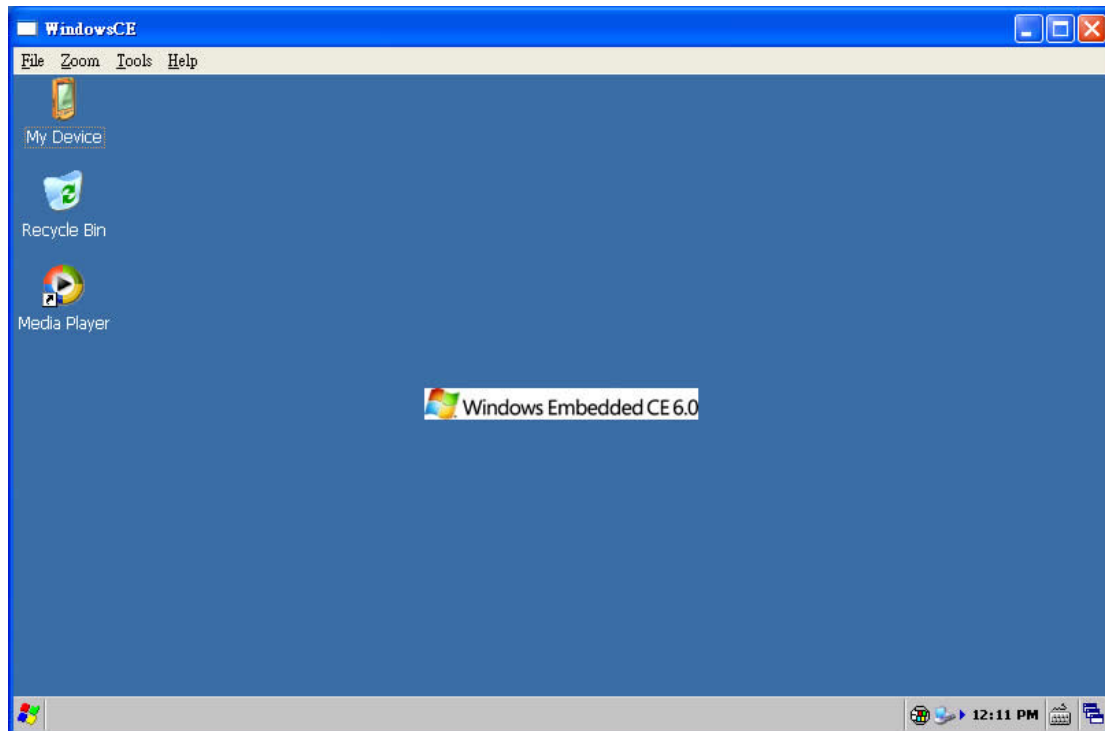


**Figure 7.2.4 ActiveSync Program**



Users should be able to see the Windows CE 6.0 desktop on the LCD screen or remote tools as shown in figure 7.2.5.

**Figure 7.2.5 Windows CE 6.0 Desktop**



Next section, we will introduce the Windows CE 6.0 system.

### **7.3. Windows CE 6.0**

This section will introduce the Windows CE 6.0 system.

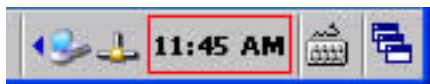
#### **7.3.1. Setting the System Time**

The MXM-6410 evaluation kit includes a battery-backed real-time clock.

To set the System Time:

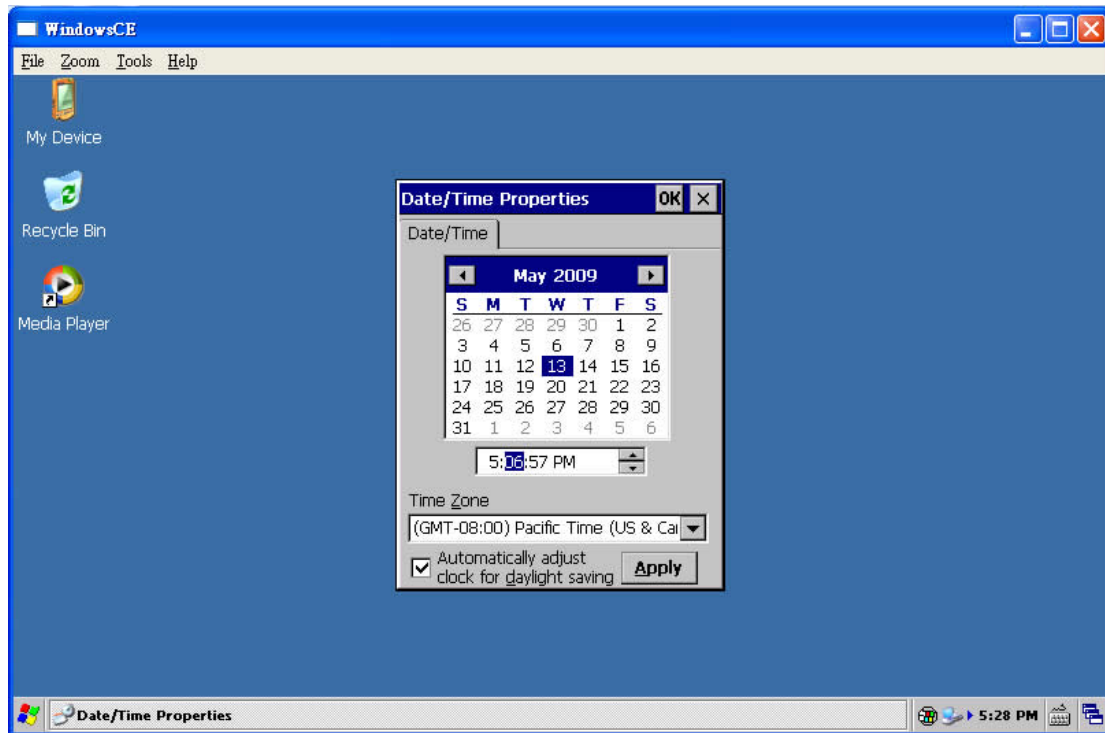
1. Double-tap on the time icon at the bottom right of the taskbar (shown below in figure 7.3.1.).

**Figure 7.3.1 Windows CE Time Icon**



2. Within the Date/Time Properties Dialog (shown below in figure 7.3.1.), set the current date and time then press the OK button to save these settings.

**Figure 7.3.2 Date/Time Properties Dialog**



MXM-6410 evaluation kit comes with a RTC battery pack. When the external DC power is exist, the RTC battery will be charged. And when the external DC power is removed, the RTC battery will supply the RTC to keep the time for few days. It is therefore; users don't have to set up the system time every time when the external DC power is temporarily gone.

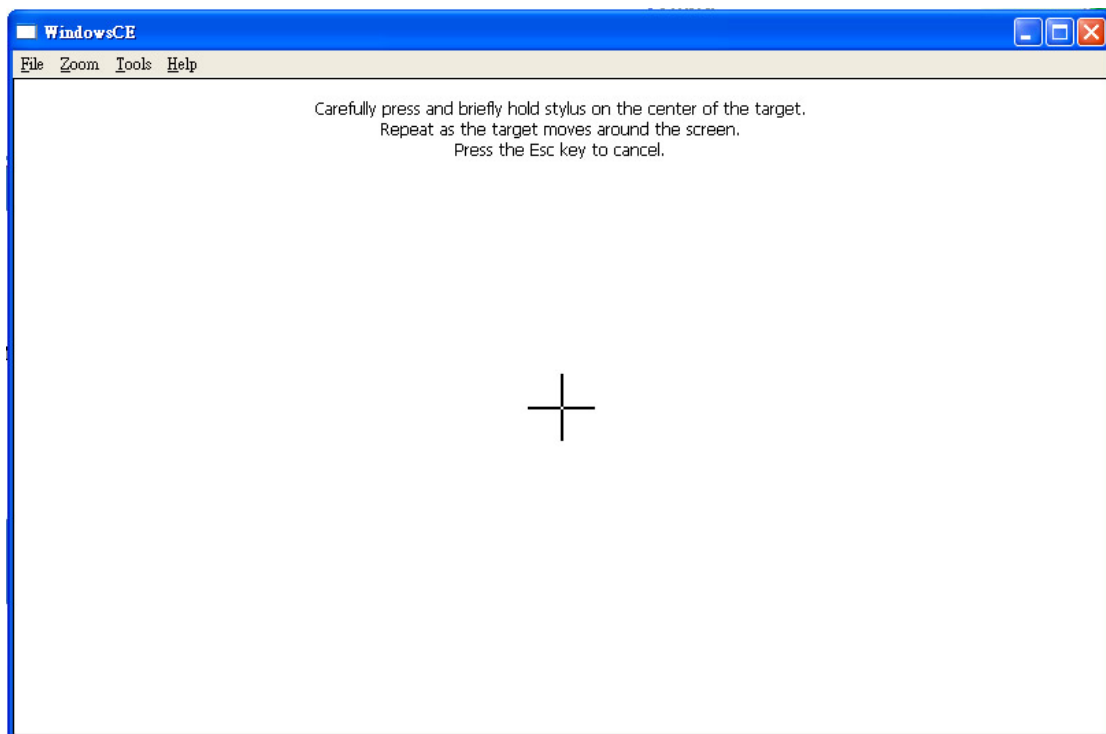
### **7.3.2. Touch Calibration**

If your MXM-6410 evaluation kit is attached to a touch screen, you will need to calibrate your touch screen at first boot. The calibration value will be stored as HIVE. That means you don't need to calibrate again at next boot.

To calibrate your touch screen:

1. Go to **Start->Settings->Control Panel->Stylus**
2. Choose "**Calibration**" Tab and click on "**Recalibrate**". You will see below as shown in figure 7.3.3.

**Figure 7.3.3 Calibrate touch panel**



3. Carefully press and briefly hold stylus on the center of the cross target. Repeat as the cross target move around the screen.

### **7.3.3. File System and Registry Basic**

#### **7.3.3.1. Making Files Persistent**

Files written or copied to **\NandFlash** are persistent from one boot to the next. Files written to **SD** memory cards (files in **\Storage Card**) or USB Compact Flash cards (files in **\Hard Disk**) are also persistent. Files written anywhere else in the system at run time are volatile.

#### **7.3.3.2. Re-formatting the NAND Flash Folder**

**\NandFlash** default has two partitions Part00 and Part01. Part00 is formatted as a BINFS filesystem by default and stores the system firmware including of NK.bin and boot HIVE information. Part01 is formatted as FAT filesystem by default and is reserved for user storage. NAND Flash would be automatically formatted and partitioned and mounted during the boot sequence. It would also create the “Documents and Settings” directory in the **\NandFlash** folder which is required for Hive-based Registry support (system.hv and user.hv) as explained later.

Manually, **\NandFlash** can be reformatted as follows. (Note that in case

the Part00 of the \NandFlash folder is manually formatted, users need to go to eboot menu to recover the NK at next boot.)

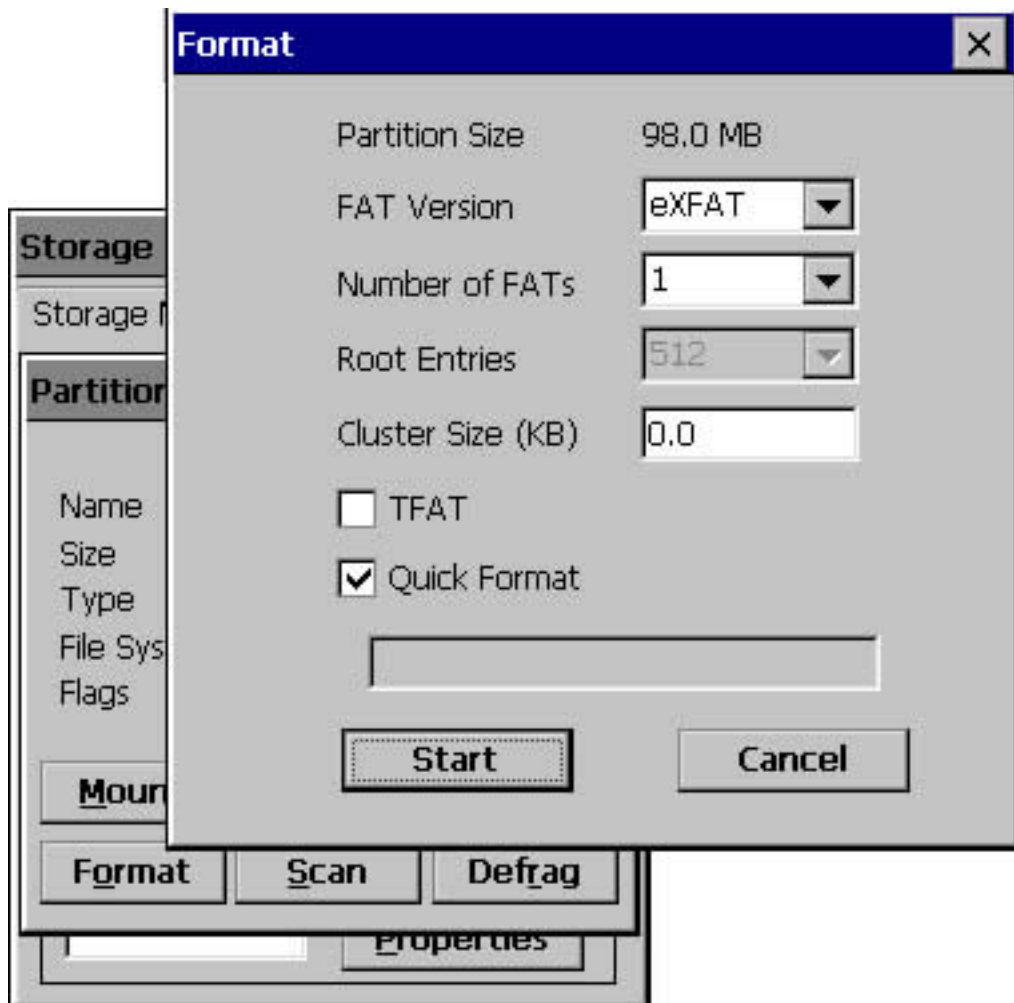
1. Go to **Start->Settings->Control Panel->Storage Manager**
2. Choose the one label with “DSK1: Microsoft Flash DISK” from **Store Info** and choose “Part01 \*” (The asterisk sign represents that this partition has been mounted.) from **Partitions**.

**Figure 7.3.4 Storage Manager Control Panel**



3. Click **Properties**.
4. Click **Dismount**.
5. Click **Format**.

**Figure 7.3.5 Format Part01 of NAND Flash**



6. Click **Start**.
  7. Click **Yes**.
  8. Click **OK**.
  9. Click **Mount**
  10. Click **OK**
  11. Exit out of Storage Properties by Clicking **OK**.
- The user partition of NAND flash has been formatted now.

#### **7.3.3.3. Persisting the Registry**

WinCE Hive-Based Registry has been well implemented. The hive-based registry stores the registry data in files, or hives, which can be kept on any file system. This removes the need to backup and restore on power off. For detailed information, please refer to *Windows CE 6.0 Help* on relating topic. Embedian's CE 6.0 HIVE-based design will be automatically flushing the registry on changes. Users don't need

to flush registry manually.

All the hive-based registries are configured to be stored in **\NandFlash\Documents and Settings**. So if the **\NandFlash** folder is manually formatted, all the registry settings would be gone and it would be restored back to the default registry after the next reboot. Therefore, it is best to reboot the evaluation kit if you have formatted the **\NandFlash** folder before proceeding with any operation.

To restore the default registry (it may interfere with software development), you have to go into the EBOOT, and press “4) Reset to factory default configuration” or just re-flash the NK again.

### **7.3.4. Networking Basics**

#### **7.3.4.1. Getting the IP Address**

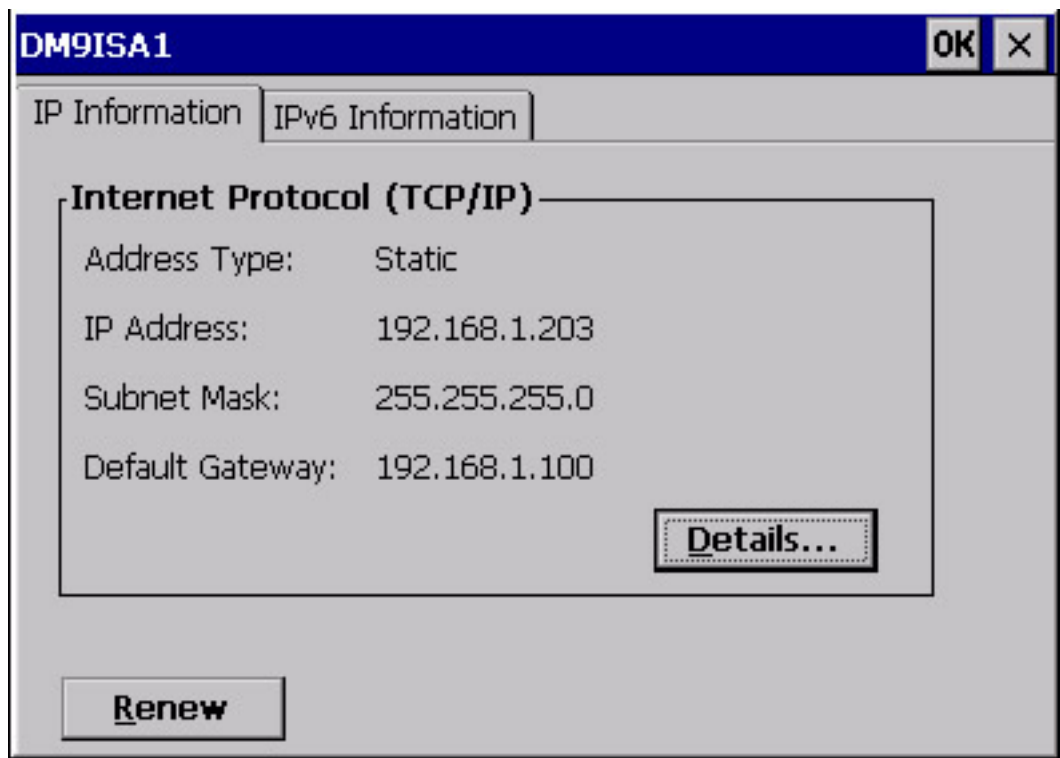
To view your current IP address, simply double click on the network icon as shown in figure 7.3.6.

**Figure 7.3.6 Network Icon**



After double-clicking on this Icon a dialog box will be shown with available IP information as shown in figure 7.3.7.

**Figure 7.3.7 IP Information Dialog**



#### **7.3.4.2. Setting Up IP Address or DHCP**

If networking hardware such as a hub or switch is available, an Ethernet cable can be used to connect the MXM-6410 evaluation kit to this hardware. Because of MXM-6410 Ethernet function supports AUTO-MDIX Ethernet protocol, there is no need of crossover cable even it is direct link. Setting static addresses or DHCP on MXM-6410 is described below. Once the addresses are properly set, the host PC will be able to communicate with the device.

On the MXM-6410 evaluation kit desktop, tap

**Start->Settings->Network and Dial-up Connections.**

Double-tap the network connection icon (named after the DM9ISA1 driver used) to open the **Network Settings** dialog box.

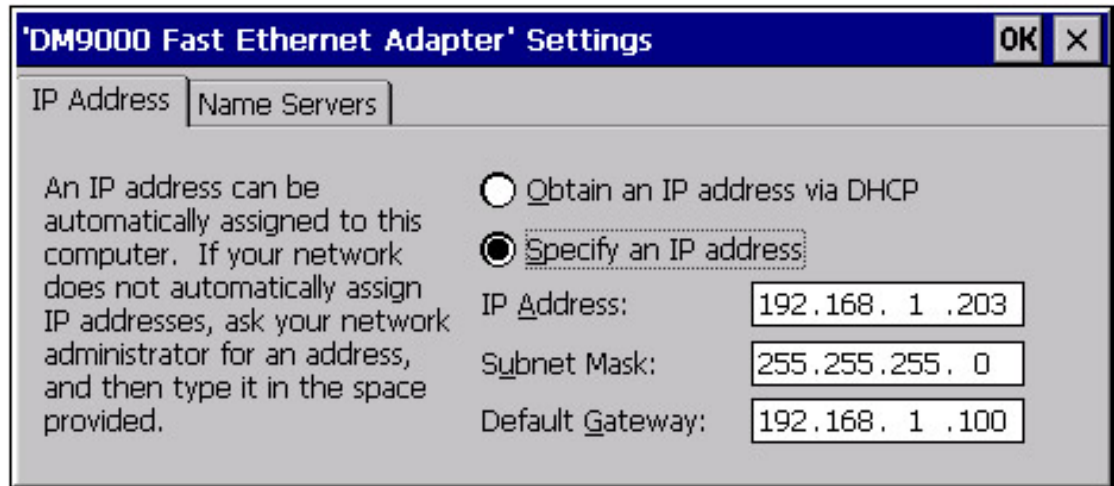
**Figure 7.3.8 Network Connection Icon**





Change the appropriate settings and tap **OK**. Note that the settings will take effect immediately.

**Figure 7.3.9 DHCP or Static IP Address Setting**



Once the IP has been modified, it will be saved to HIVE registry and the IP and Ethernet configuration will be kept at next boot.

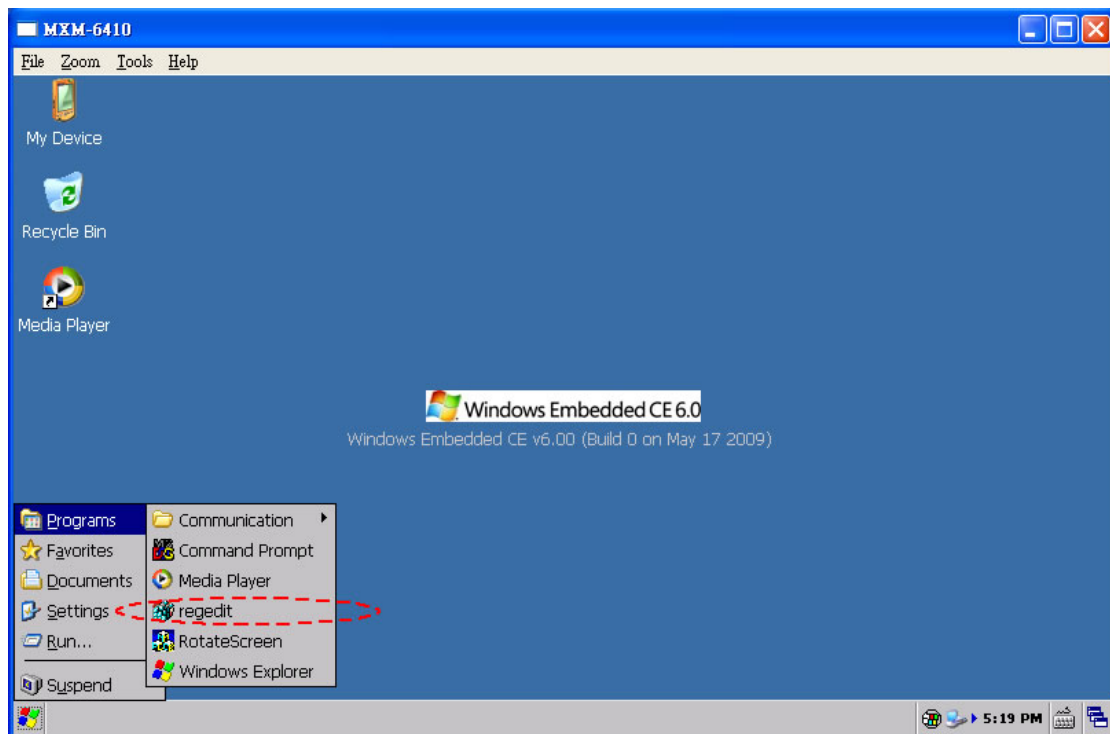
### **7.3.5. Telnet Server**

The Telnet server in the factory-installed image is configured with no access restrictions. You could disable the Telnet server or restrict access to it before deploying the product or connecting to an unsecured network. The Telnet server is not started on MXM-6410 hardware at boot by default.

To turn on the telnet service, users need to enable the "*IsEnabled*" registry key setting under the **HKEY\_LOCAL\_MACHINE\Comm\TELNETD** registry key. Figure 7.3.10 and figure 7.3.11 show how to enable the telnet service.

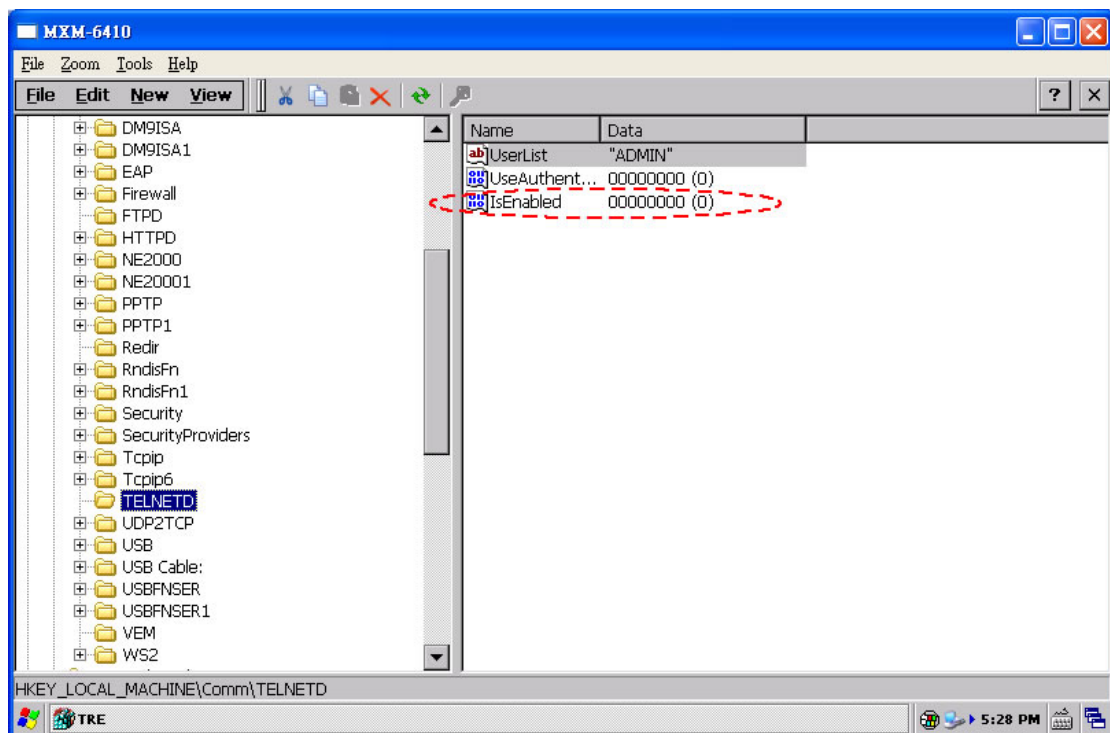
Go to **Start → Programs → regedit**

**Figure 7.3.10 regedit Utility**



The telnet registry settings are under the **HKEY\_LOCAL\_MACHINE\Comm\TELNETD** registry key. Double click the “*IsEnabled*” registry key set it from “0” to “1”.

**Figure 7.3.11 Enable Telnetd Service**

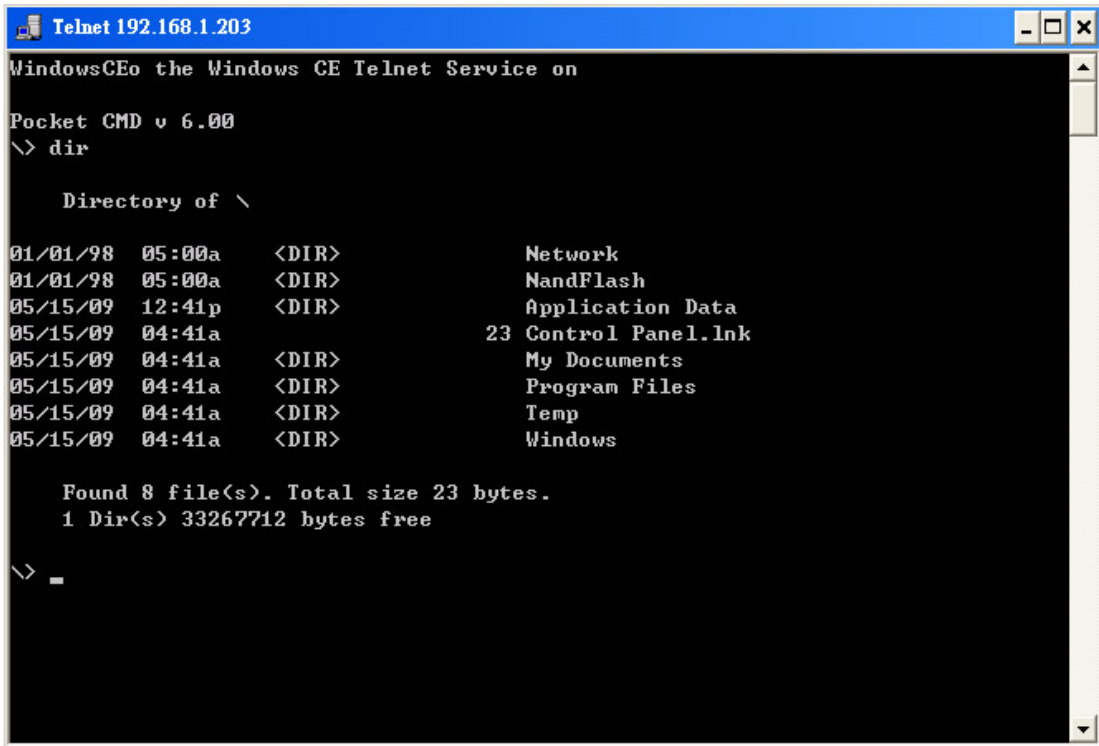


## **Embedian, Inc.**

Note: You will need to reboot the evaluation kit to take effect the registry key enabled.

To access the Telnet server on an MXM-6410 device, simply enter *telnet* <Your Device IP Address>. See the example below.

**Figure 7.3.12 Telnet Prompt**



```
Telnet 192.168.1.203
WindowsCEo the Windows CE Telnet Service on
Pocket CMD v 6.00
\> dir

      Directory of \

01/01/98  05:00a  <DIR>          Network
01/01/98  05:00a  <DIR>          NandFlash
05/15/09  12:41p  <DIR>          Application Data
05/15/09  04:41a          23 Control Panel.lnk
05/15/09  04:41a  <DIR>          My Documents
05/15/09  04:41a  <DIR>          Program Files
05/15/09  04:41a  <DIR>          Temp
05/15/09  04:41a  <DIR>          Windows

      Found 8 file(s). Total size 23 bytes.
      1 Dir(s) 33267712 bytes free

\> _
```

Note: If you used telnet client like putty program to telnet in, the first few commands might give you a “command not found” message. Just a couple of trials will be fine. But Windows telnet command prompt will not appear to see this.

### **7.3.5.1. Default Registry Setting**

It's necessary to be aware of registry settings that impact security. The telnet registry settings are under the

**HKEY\_LOCAL\_MACHINE\Comm\TELNETD** registry key.

Telnet server reads the values in the registry before serving each request. Therefore, changes made to the registry take effect immediately and do not require the telnet server to be restart. The registry change will be kept on next reboot.

**Table 7.3.1. TELNETD Registry Value**

<b>HKEY_LOCAL_MACHINE\Comm\TELNETD</b>	
<b>Value</b>	<b>Description</b>
<b>IsEnabled :</b> <b>REG_DWORD</b>	<i>Indicates if the Telnet server is enabled. Set this value to a non-zero value to enable the server, and to 0 to disable the server. If the sample Telnet server is started and this value has not been set, it defaults to accepting connections.</i>
<b>UseAuthentication:</b> <b>REG_DWORD</b>	<i>To require a password check on the user, set this value to 1; otherwise, set it to 0. By default, the Telnet sample requires authentication.</i>
<b>UserList : String</b>	<i>Provides a comma-separated list of allowed users. Requires UseAuthentication to be enabled.</i>

#### **7.3.5.2. Changing the Telnet Server Access Privileges**

To change the Telnet server access privileges, the Registry Editor tool under the **Windows** folder (or **Start → Programs → regedit**) will be used.

To modify the Telnet server access privileges:

**Step 1** Click the regedit.exe located at \Windows folder.

**Step 2** Browse to the **HKEY\_LOCAL\_MACHINE\Comm\TELNETD**.

**Step 3** Ensure that the **IsEnabled** value is **1**.

**Step 4** Ensure that the **UseAuthentication** value is **0**.

**Step 5** Set the **UserList** value to a semi-colon separated list of users. (For example, **user1;user2**).

Refer to the topic **Telnet Server User Lists** in the Platform Builder documentation for more detailed information on this subject.

#### **Examples:**

[HKEY\_LOCAL\_MACHINE\COMM\TELNETD]

UserList="GladysL;AnnD;-KimY;@SomeGroup;-@villains"

Allows GladysL, AnnD, and the SomeGroup group, but restricts the KimY user and the villains group.

The asterisk or the at sign and the asterisk (\* or @\*) allows all users,

and the combination of the hyphen and the asterisk, or the combination of the hyphen and the at sign and the asterisk (-\* or -@\*) denies all users.

The string is interpreted sequentially. In other words, if GladysL is member of the Finance group, the following strings will allow GladysL.

GladysL

@ Finance

@ Finance;GladysL

GladysL;@ Finance

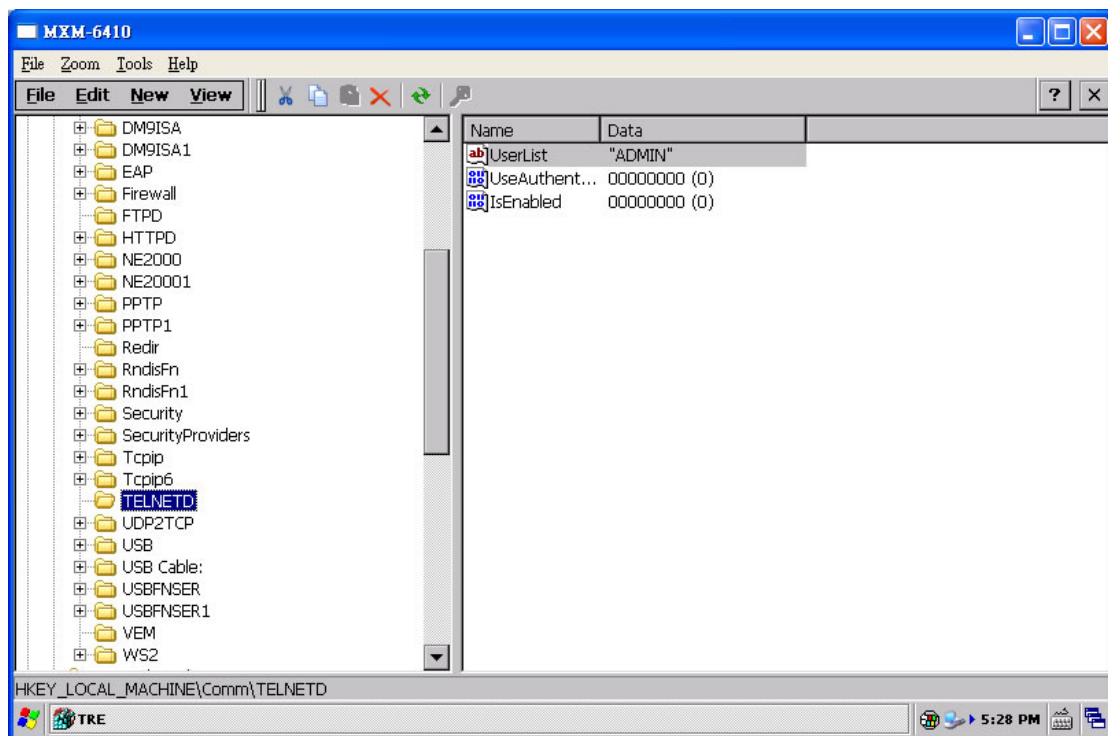
@ Admins;-GladysL

GladysL;-@ Finance

GladysL;-GladysL

\*;-GladysL;-@ Finance

**Figure 7.3.13 Telnet Registry**



### **7.3.5.3. Security Note**

The security on the Telnet server is very light and vulnerable to security attacks. Even if the Telnet server is configured to require password

authentication, the password is sent in plain text across the network and is therefore vulnerable to packet sniffing. A malicious user could obtain the password to MXM-6410 by watching packets sent back and forth between the Telnet server and the client during the authentication stage. If a malicious user could log on to the device, they would have complete privileges over it. This could involve deleting or modifying key system files and the registry.

Because of these serious security risks, it is strongly recommended you only run the Telnet server for development and debugging purposes, on a controlled, private network where you trust the users. It is strongly recommended that you do not deploy this Telnet server on a public network such as the Internet.

#### **7.3.5.4. Recommendations**

**Set the user list and domain variables to prevent hacker attacks on your device.** If Telnet server is used without appropriate values set for the User List and Domain variables, your Telnet server will be vulnerable to hacker attacks.

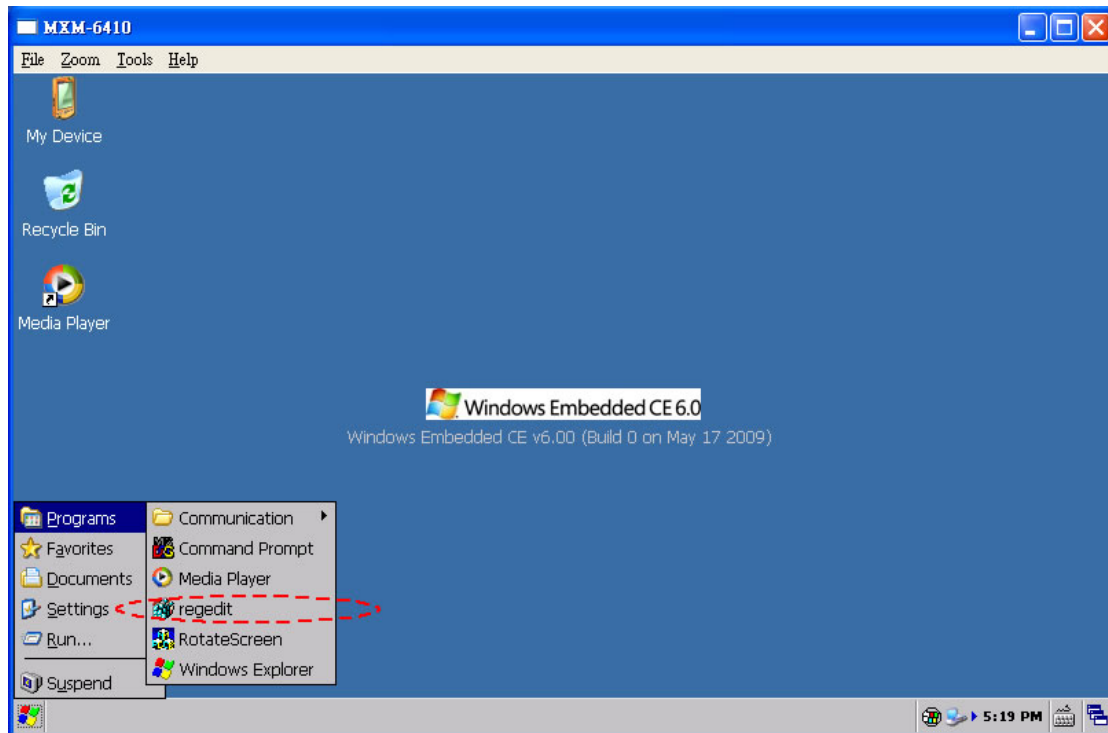
To prevent such attack, the user name is the **UserList** registry value must be set for each of the servers that are currently running. The user will then need to log in with the specified user name and appropriate password to use the server. You can also set the domain variable in the **DefaultDomain** registry value, which is located under **HKEY\_LOCAL\_MACHINE\Comm\Redir** registry key.

#### **7.3.6. FTP Server**

The FTP server in the factory-installed image is configured with no access restrictions. You should disable the FTP server or restrict access to it before deploying the MXM-6410 or connecting to an unsecured network. The FTP server is not started on MXM-6410 hardware at boot by default. To turn on the FTP service, users need to enable the “*IsEnabled*” registry key setting under the **HKEY\_LOCAL\_MACHINE\Comm\FTPD** registry key. Figure 7.3.14 and figure 7.3.15 show how to enable the ftp service.

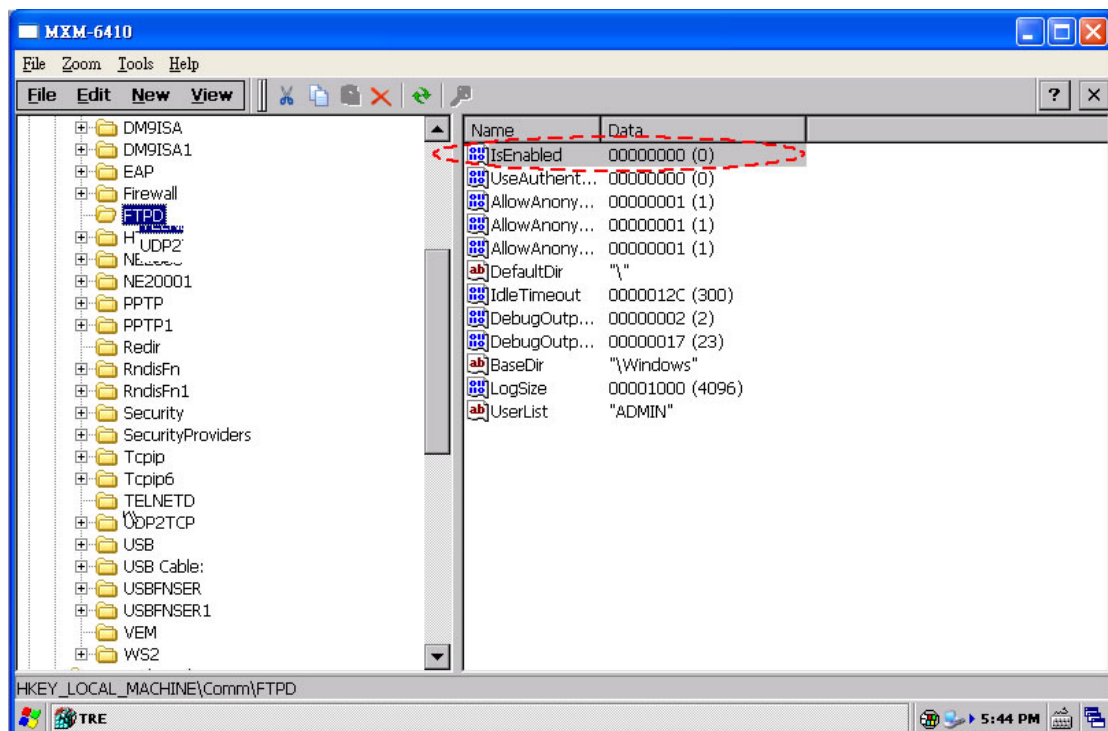
Go to **Start → Programs → regedit**

**Figure 7.3.14 regedit Utility**



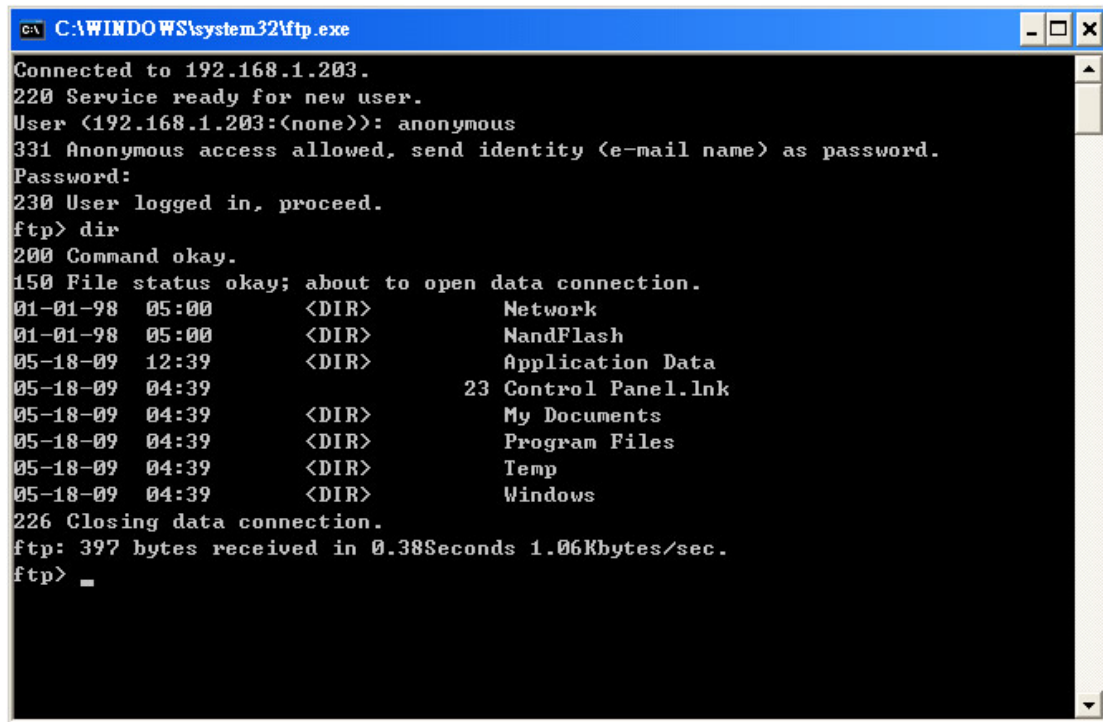
The ftpd registry settings are under the **HKEY\_LOCAL\_MACHINE\Comm\FTPD** registry key. Double click the “IsEnabled” registry key set it from “0” to “1”.

**Figure 7.3.15 Enable Ftpd Service**



To access the FTP server on an MXM-6410 device, simply enter *ftp <Your Device IP address>* at your PC command prompt. See the example below.

**Figure 7.3.16 FTP Prompt**



```
C:\WINDOWS\system32\ftp.exe
Connected to 192.168.1.203.
220 Service ready for new user.
User (192.168.1.203:(none)): anonymous
331 Anonymous access allowed, send identity (e-mail name) as password.
Password:
230 User logged in, proceed.
ftp> dir
200 Command okay.
150 File status okay; about to open data connection.
01-01-98 05:00 <DIR> Network
01-01-98 05:00 <DIR> NandFlash
05-18-09 12:39 <DIR> Application Data
05-18-09 04:39 23 Control Panel.lnk
05-18-09 04:39 <DIR> My Documents
05-18-09 04:39 <DIR> Program Files
05-18-09 04:39 <DIR> Temp
05-18-09 04:39 <DIR> Windows
226 Closing data connection.
ftp: 397 bytes received in 0.38Seconds 1.06Kbytes/sec.
ftp> _
```

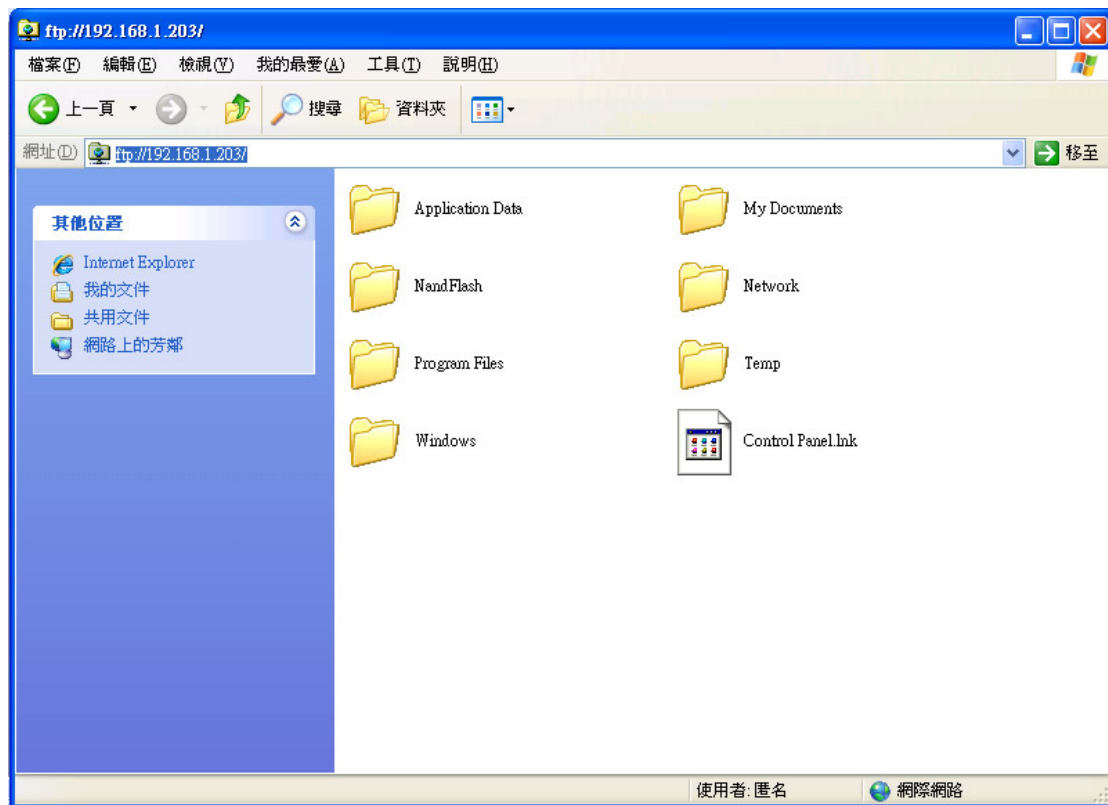
Once this *ftp <Your Device IP address>* command has been entered, the FTP server will respond by asking for a username and password. It is currently configured to allow any user in with any password, including **anonymous**.

### **Use File Browser to access the FTP Server**

A regular file browser in your Windows PC may be used to access the FTP server as well. To use the FTP server this way, use *ftp://the device IP address*.



**Figure 7.3.17 Use File Browser to access FTP Server**



MXM-6410 implementation of FTP server enables you to transfer files from a desktop computer using a TCP/IP connection. The implementation of FTP server in MXM-6410 is based on RFC 959. This included FTP server supports the minimum implementation of the FTP protocol defined in RFC 959. This minimum implementation includes configuration values, transfer parameters, and ASCII and Image data types, and allows FTP to operate with a minimum of error messages.

#### **7.3.6.1. Default Registry Setting**

It's necessary to be aware of registry settings that impact security. The telnet registry settings are under the

**HKEY\_LOCAL\_MACHINE\Comm\FTPD** registry key.

FTP server reads the values in the registry before serving each request. Therefore, changes made to the registry take affect immediately and do not require the ftp server to be restart. The registry change will be kept on next reboot.

**Table 7.3.2. FTPD Registry Value**

<b>HKEY_LOCAL_MACHINE\Comm\FTPD</b>	
<b>Value</b>	<b>Description</b>
<b>AllowAnonymous :</b> <b>REG_DWORD</b>	<p>Default set to 1. Possible values are 0 (false) or 1 (true).</p> <p>Determines whether the server will allow anonymous access.</p>
<b>AllowAnonymousUpload :</b> <b>REG_DWORD</b>	<p>Default set to zero (0). Possible values are 0 (false) or 1 (true).</p> <p>Determines whether authorization is required to upload files to the server, delete files from the server, and rename files.</p>
<b>AllowAnonymousVroots :</b> <b>REG_DWORD</b>	<p>Default set to zero (0). Possible values are 0 (false) or 1 (true).</p> <p>Specifies whether access to virtual roots is granted or denied to anonymous users.</p>
<b>DefaultDir : String</b>	<p>Default root directory. Directory and subdirectories of this key are accessible remotely. If this value is not set in the registry, the default is \Temp.</p>
<b>IsEnabled : REG_DWORD</b>	<p>Default set to 1. Possible values are 0 (false) or 1 (true).</p> <p>Determines whether or not the server will accept incoming connections. This value is typically used to keep the server disabled at boot time.</p>
<b>UserList : REG_MULTI_SZ</b>	<p>Provides a comma-separated list of allowed users.</p>
<b>NoSystemInfo :</b> <b>REG_DWORD</b>	<p>No default set in registry.</p> <p>Uses value of 0 if no value is set. If NoSystemInfo is set to 1, and a remote FTP</p>

	<i>client requests the Operating System name and version from the FTP server (via the "SYST" command), the FTP server will not indicate it.</i>
--	-------------------------------------------------------------------------------------------------------------------------------------------------

### **7.3.6.2. Changing the FTP Server Access Privileges**

To change the FTP server access privileges, the Registry Editor tool under the **Windows** folder (or **Start → Programs → regedit**) will be used.

To modify the FTP server access privileges:

**Step 1** Click the regedit.exe located at **Windows** folder.

**Step 2** Browse to the **HKEY\_LOCAL\_MACHINE\Comm\FTPD**.

**Step 3** Ensure that the *IsEnabled* value is **1**.

**Step 4** Ensure that the *UseAuthentication* value is **0**.

**Step 5** Set the **UserList** value to a semi-colon separated list of users. (For example, **user1;user2**).

Refer to the topic **Telnet Server User Lists** in the Platform Builder documentation for more detailed information on this subject.

#### **Examples:**

[HKEY\_LOCAL\_MACHINE\COMM\TELNETD]

UserList="GladysL;AnnD;-KimY;@SomeGroup;-@villains"

Allows GladysL, AnnD, and the SomeGroup group, but restricts the KimY user and the villains group.

The asterisk or the at sign and the asterisk (\* or @\*) allows all users, and the combination of the hyphen and the asterisk, or the combination of the hyphen and the at sign and the asterisk (-\* or -@\*) denies all users.

The string is interpreted sequentially. In other words, if GladysL is member of the Finance group, the following strings will allow GladysL.

GladysL

@ Finance

@ Finance;GladysL

GladysL;@ Finance

@ Admins;-GladysL

GladysL;-@ Finance  
GladysL;-GladysL  
\*;-GladysL;-@ Finance

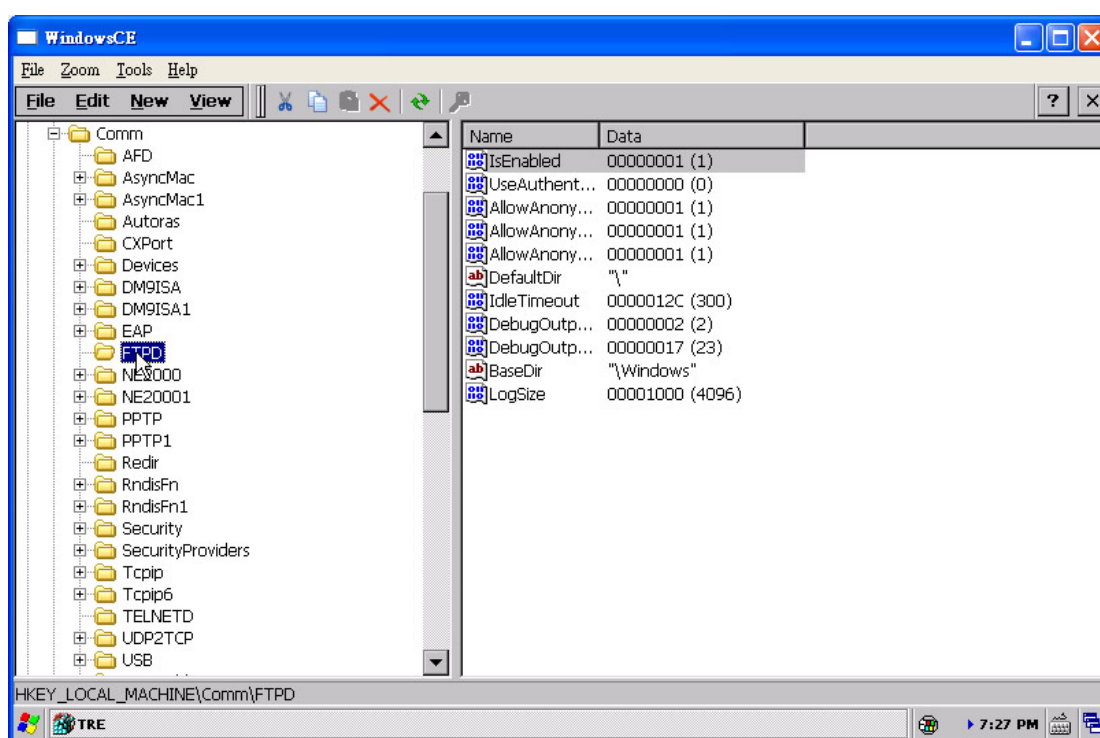
**Step 6** Set the **AllowAnonymous** value to 1 if you wish to enable anonymous FTP logins.

**Step 7** Set the **AllowAnonymousUpload** value to 1 if you wish to enable anonymous users from uploading files to the FTP server.

**Step 8** Set the **DefaultDir** string to whichever directory you wish to use as the FTP root directory. Users will not be able to go outside of this directory.

Further information may be found in the Platform Builder documentation on the FTP server, located in **Application Development > FTP Server**.

**Figure 7.3.18 FTP Registry**



### 7.3.6.3. Security Notes

If **AllowAnonymous** is set to true, it will allow users to connect the server without providing verifiable credentials. Anyone can log in using the username "anonymous" and any password to gain access. It is

recommended that you set this value to false and use the **UserList** registry setting to specify all allowed users.

If **AllowAnonymousUpload** is set to true, unauthenticated users will be able to copy files to, and delete files from, your server. This can be very dangerous because attackers might upload dangerous applications and documents, or they might delete important system files. It is not recommended to allow upload permission for anonymous users.

If **AllowAnonymousVroots** is set to false, anonymous users will only be able to access the main FTP shares. If this value is set to true, unauthenticated users will also be able to access VROOTs as well as the main share. Therefore you should use this setting with caution.

Setting **UseAuthentication** to false enables clients to connect to the server without providing credentials. It is therefore strongly recommended that you do not set this value to false. Change this setting only if you have anonymous clients that must access the server but cannot or will not send USER and PASS credentials.

It is recommended that you set this value to a list of users who should have access to the server and its member VROOTs. Specifying the allowed users in **UserList** and setting **AllowAnonymous** to false will help protect the device from most attacks and keep your files available only to those users who need to see them.

#### **7.3.6.4. Recommendations**

**Set the user list and domain variables to prevent hacker attacks on your device.** If FTP server functionality is used without appropriate values set for the User List and Domain variables, the FTP server will be vulnerable to hacker attacks.

These variables are not set by default. A hacker must only guess the device's password, the way it is set in Control Panel, to obtain access to server.

To prevent such attack, the user name is the **UserList** registry value must be set for each of the servers that are currently running. The user will then need to log in with the specified user name and appropriate password to use the server. You can also set the domain variable in the **DefaultDomain** registry value, which is located under

**HKEY\_LOCAL\_MACHINE\Comm\Redir** registry key. Setting the **DefaultDomain** registry value will require FTP clients to have valid domain credentials to log in.

### **7.3.7. File Server**

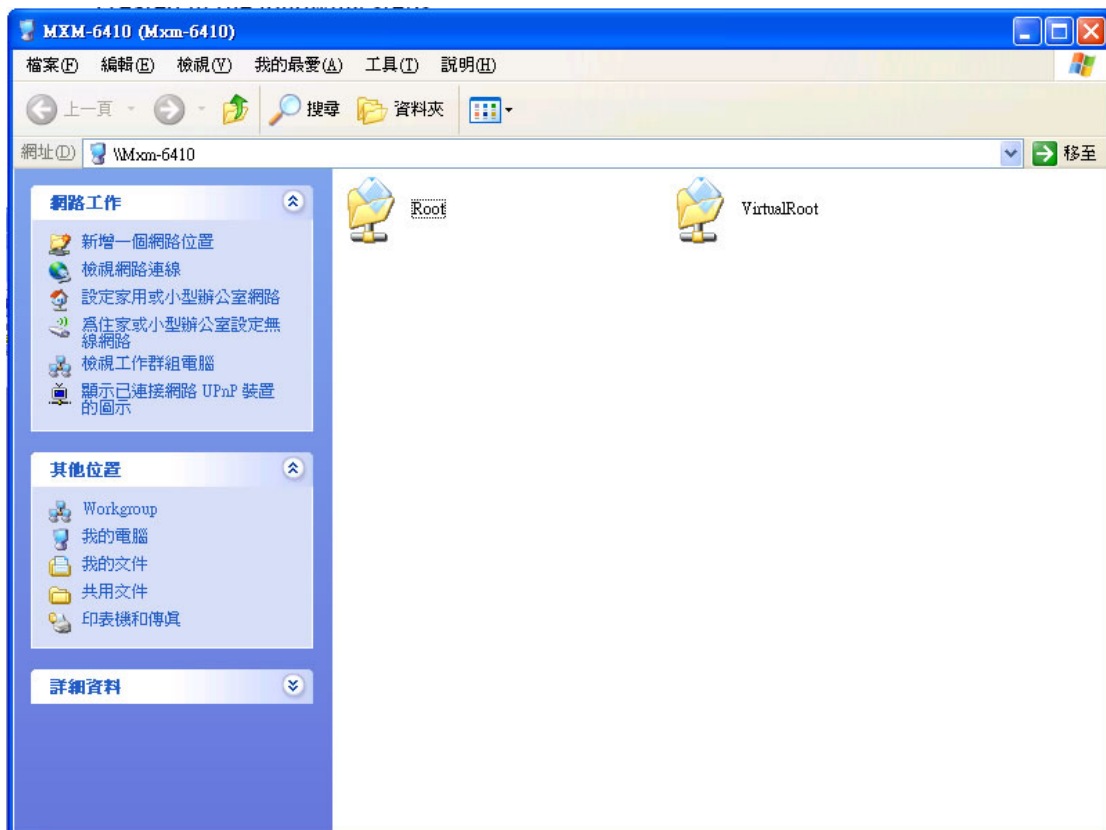
File server in the factory-installed image is configured with no access restrictions. You should disable the file server or restrict access to it before deploying the MXM-6410 or connecting to an unsecured network. The File Server enables clients to access files and other resources, such as printer, from a server over using TCP/IP. File Server uses the Common Internet File System (CIFS). This is an extension to the Server Message Block (SMB) file sharing protocol. CIFS enables a network-enabled application to access and manipulate files and directories on a remote server in the same way that the application accesses and manipulates files and directories on the local system.

To use the file server, users can access and browse the shared folders that created in the default registry in the following steps.

1. On the PC, from the **Start** menu, choose **Run**.
2. In the Open box, type **\\MXM-6410**, and then choose **OK**.

You can now browse the shared files in devices as showed in figure 7.3.19.

**Figure 7.3.19 Access File Server**



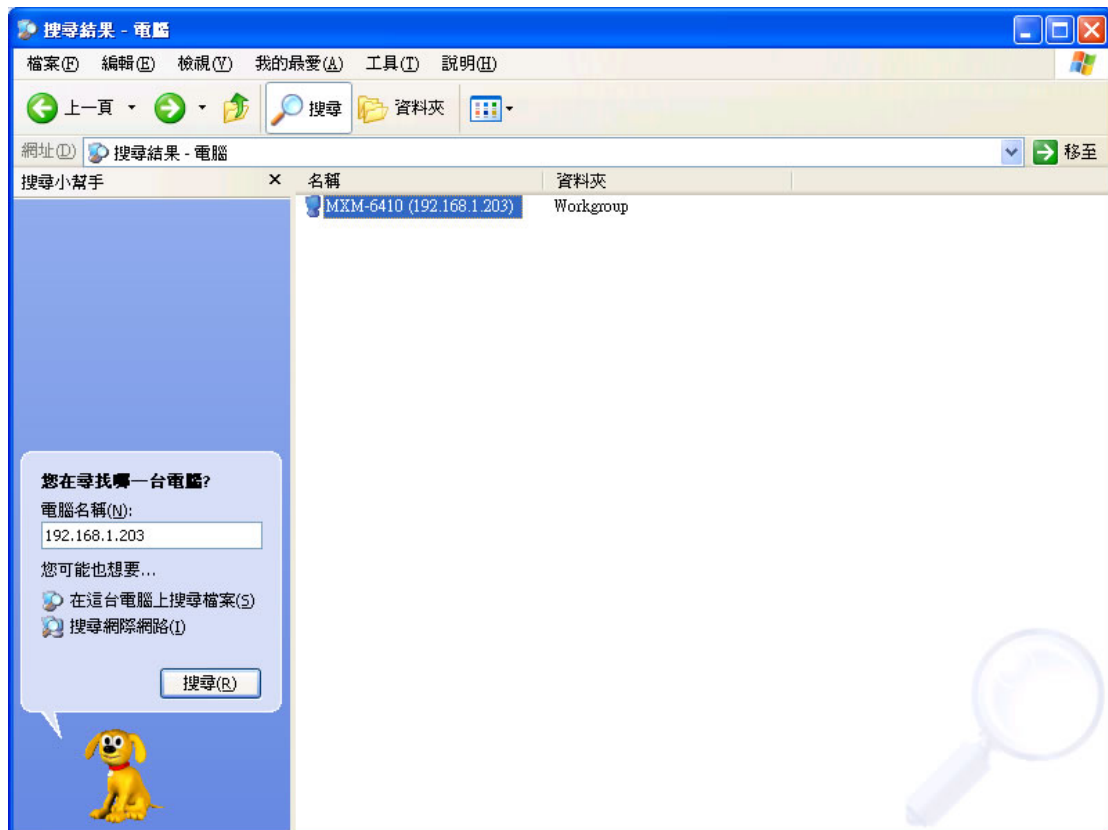
The alternative way to access to shared files in the file server is described as follows.

1. On the PC, from the **Start** menu, choose **Search**.
2. In the searching menu, choose **Computer or People**, and then choose **Computer on the network**.
3. In the Open box, type **MXM-6410** or **<Device IP Address>**, and then click **Search**.

You can now browse the shared files in devices as showed in figure 7.3.20



**Figure 7.3.20 The other way to access file server from PC**



#### **7.3.7.1. Default Registry Settings**

The registry stores information necessary to configure the system for applications and hardware devices. The registry also contains information that the operating system continually references during operation. MXM-6410 enables you to create virtual file server directories. To users who access your file server share, virtual directories appear as subdirectories of the file server share, although these directories may be located in a different folder. You can create a virtual root directory called "VirtualRoot" that points to \Windows directory of the device and a root directory called "Root" that points to \NandFlash directory of the device by specifying the following registry key:

**HKEY\_LOCAL\_MACHINE\Services\SMBServer\Shares\Root** and  
**HKEY\_LOCAL\_MACHINE\Services\SMBServer\Shares\VirtualRoot**

Note the maximum length of the virtual root directory and root directory is 12 characters.

**Table 7.3.3. File Server Registry Value**

<b>HKEY_LOCAL_MACHINE\Services\SMBServer\Shares</b>	
<b>Value</b>	<b>Description</b>
<b>UseAuthentication:</b> <b>REG_DWORD</b>	<i>No default set. Setting this value to 0 will disable the authentication on the file server. The file server will be accessible to all users on the network.</i>

**Table 7.3.4. VirtualRoot/Root of File Server Registry Value**

<b>HKEY_LOCAL_MACHINE\Services\SMBServer\Shares\VirtualRoot</b> <b>HKEY_LOCAL_MACHINE\Services\SMBServer\Shares\Root</b>	
<b>Value</b>	<b>Description</b>
<b>Path : String</b>	<i>Specifies the path to be shared.</i>
<b>Type:</b> <b>REG_DWORD</b>	<i>Setting this value to 1 designates this as a print server share, setting this value to 0 (zero) designates this as a file server share.</i>
<b>UserList : String</b>	<i>Provides a comma-separated list of allowed users. Requires UseAuthentication to be enabled.</i>

#### **7.3.7.2. Security Note**

The default setting of authentication on the file server is disabled. However, it is not recommended that you disable authentication on the file server and you share the \Windows or root directory.

You can specify a list of folders that cannot be shared. You can use any name for each folder you specify in the exclusion list. Setting the

**HKEY\_LOCAL\_MACHINE\Services\SMBServer\Shares\ExcludePaths** registry key prevents the configuration functions from creating the specified shares, so that they cannot be accessed by an un-trusted application.

**Table 7.3.5 An example to exclude the folders to be shared**

<b>HKEY_LOCAL_MACHINE\Services\SMBServer\Shares\ExcludePaths</b>	
<b>Value Name</b>	<b>Value</b>
<b>"Windows"</b>	<b>"\\Windows"</b>
<b>"My Documents"</b>	<b>"Documents and Settings"</b>

### **7.3.8. Web Server**

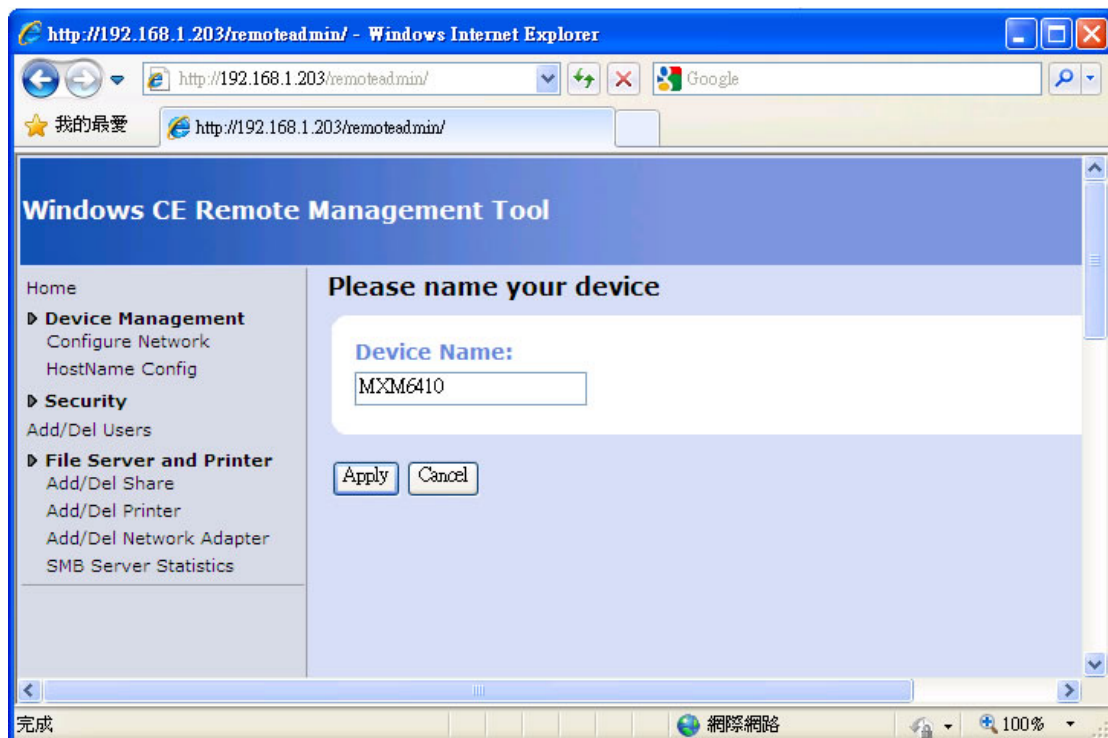
Web server facilitates the use of the Internet for communication between MXM-6410 device and network printers, scanners, and other shared equipment. The Web server applications send Hypertext Markup Language (HTML) pages to a requesting browser. Users only need to have an Internet connection and a browser to be able to make use of the Web server functionality. The Web server supports IPv6 and also supports the use of Active Server Pages (ASP).

The HTTP server is not started on MXM-6410 hardware at boot by default. To turn on the telnet service, users need to enable the *"IsEnabled"* registry key setting under the

**HKEY\_LOCAL\_MACHINE\Comm\HTTPD** registry key.

To access the Web server, just type *<IP Address>* at the Web browser. The default Web service configuration setting is remote administration page. It is very useful to use when the device is without display. At first connection to the Web server, users will be asked to input the password and re-type to confirm. After that, figure 7.3.21 shows the remote administration pages.

**Figure 7.3.21 Remote Administration Page**



**Note:** If users would like to use a own custom webpage, users need to modify the “Default” registry key setting under the **HKEY\_LOCAL\_MACHINE\Comm\HTTPD\ROOTSV** registry key from “\$REDIRECT” to “\windows\www\wwwpubl”

#### **7.3.8.1. Default Registry Settings**

It is necessary to be aware of the registry settings that impact security. The Web server settings are located under the **HKEY\_LOCAL\_MACHINE\Comm\HTTPD** registry key. If you make changes to the Web server registry settings, it is necessary to stop the Web server and restart it to make the changes take effect. The “IsEnabled” registry value is default set to disable the Web service and is checked only when the Web server is initially loaded. If the registry value is set to zero (0), the Web server does not start. Changing this value to zero (0) while the Web server is running has no effect. You also must stop the Web server to make it stop accepting connections.

**Table 7. 3.6. HTTPD Registry Key and Named Values**

<b>HKEY_LOCAL_MACHINE\Comm\HTTPD</b>	
<b>Value</b>	<b>Description</b>
<b>BasicRealm :</b> <b>String</b>	<i>Specifies the string that the Web Server will send to clients as its Basic realm when performing basic authentication. If this registry value is not set, the Web Server will default to using the string "Microsoft-WinCE".</i>
<b>IsEnabled :</b> <b>REG_DWORD</b>	<i>If the value is not set in the registry, the Web Server is enabled. If the value is set to zero (0), the Web Server does not accept connections from the network, even from the local host.</i>
<b>Port :</b> <b>REG_DWORD</b>	<i>Default setting is 80. This port receives HTTP connections. Do not set the port to zero (0). When the Web Server is running from Services.exe, which is the default, this value is ignored and Services.exe becomes the super server.</i>
<b>Basic :</b> <b>REG_DWORD</b>	<i>Default setting is zero (0). If this value is nonzero, the Web Server uses Basic authentication for client connections.</i>  <i>Enables Basic authentication, which enables the client browser to send the user identifier and password to the server.</i>
<b>NTLM :</b> <b>REG_DWORD</b>	<i>Default setting is 1. If this value is set to nonzero, the Web Server uses NTLM authentication for client browser connections. Also, if this value is nonzero, the failure of Basic authentication forces NTLM authentication.</i>  <i>If the value is not set in the registry, NTLM is not used.</i>
<b>DirBrowse :</b> <b>REG_DWORD</b>	<i>Default setting is zero (0). If this value is set to nonzero, directory browsing is allowed. If this value is not set in the registry, directory browsing is turned off.</i>

	<i>Turns on the Web Server's ability to provide local directory browsing.</i>
<b>Filter DLLs :</b> <b>String</b>	<i>Default not set in the registry.</i>  <i>Specifies a list of DLL names, separated by commas, that identifies the filters to use.</i>
<b>DefaultPage :</b> <b>String</b>	<i>Default not set in the registry. If the value is not present in the registry, the Web Server will use "default.htm;index.htm".</i>  <i>Specifies a list of page names, separated by semicolons, that indicates names interpreted by the Web Server to be default pages. When browsing a directory, the Web Server traverses this list searching for a file of the same name in the directory. If the file exists, it is sent to the client. If no matching file exists, the Web Server sends a directory listing or returns an error, depending on whether directory browsing is enabled. If more than one DefaultPage file name is matched, the Web Server uses the first matching file name.</i>
<b>AdminUsers :</b> <b>String</b>	<i>Default not set in the registry.</i>  <i>Specifies a list of user names, separated by semicolons. A user who has gained user access must be listed in this key to gain Administrator access.</i>
<b>LogFileDirectory :</b> <b>String</b>	<i>Default setting is "\windows\www" directory. If the name is not set or if the specified directory is inaccessible, no logging is performed.</i>  <i>Specifies the name of the directory where the logging files are created.</i>
<b>PostReadSize :</b> <b>REG_DWORD</b>	<i>If the value is not set in the registry, PostReadSize will default to 48 KB. The Web Server uses a minimum value of 8192 bytes (8 KB). If the value in the registry is less than 8 KB, the value is ignored and the Web</i>

	<p>Server will use 8 KB.</p> <p>Specifies the maximum number of bytes that the Web Server reads when receiving POST data. To read more data, you must use a raw data filter or call <a href="#">ReadClient</a> in an ISAPI extension.</p>
<p><b>MaxLogSize :</b> <b>REG_DWORD</b></p>	<p>Default setting is 32 KB. If this value is not set in the registry, or if it is set to zero (0), no logging is performed.</p> <p>Specifies the maximum size, in bytes, that a log file can become before it is rolled over.</p>
<p><b>MaxHeaderSize :</b> <b>REG_DWORD</b></p>	<p>Default setting is 48 KB in the registry.</p> <p>Specifies the maximum number of bytes that the Web Server will read of HTTP headers. If the header size exceeds this value, the Web Server will terminate the session and return a message to the client: 400 - Bad Request.</p>
<p><b>MaxConnections :</b> <b>REG_DWORD</b></p>	<p>Default is not set in the registry. If the value is not set in the registry, MaxConnections will default to 10.</p> <p>Specifies the maximum number of simultaneous connections to the Web site. After the maximum number of connections is established, additional client requests will be sent a message: 503 – Server Too Busy.</p>
<p><b>ServerID :</b> <b>String</b></p>	<p>Default is not set in the registry. If the value is not set in the registry, ServerID will default to "Microsoft-WinCE/X.Y", where X is the major version and Y is the minor version of Windows CE-based device.</p> <p>If ServerID is set, the Web Server returns the specified server name in the response header.</p> <p>Identifies the server name that is included when the Web Server generates HTTP response headers. The</p>

	<i>response header includes a field name "Server: ". Optionally, you can include the software version number or any similar information in the string.</i>
--	--------------------------------------------------------------------------------------------------------------------------------------------------------------------



### **7.3.8.2. Security Note**

When using **Basic** authentication, the client browser sends the user identifier and password to the server in clear text. In addition, all data sent between the client and browser is in clear text and therefore vulnerable to packet sniffing. You should consider using SSL to help protect sensitive information.

**NTLM** Although the client browser sends the password to the server in encrypted format, all data sent between the client and the browser is in clear text and therefore vulnerable to packet sniffing. You should using SSL to help protect sensitive information.

**DirBrowse** turns on Web server's ability to provide local directory browsing. This exposes the local file system to remote browser through HTTP. Users can view file lists and download files depending on virtual root and authentication registry settings. Enabling directory browsing increases the potential attack surfaces, therefore you should enable directory browsing only when necessary.

**AdminUsers** User names in this list identify the administrators of the site who have access to all virtual roots hosted on this Web site, including the restricted sites. Choose these users carefully and ensure that they set proper password, otherwise their accounts could be used to gain access to restricted sites.

**MaxConnections** Setting the value too small can block user access to the site. However, if the value is too large, the Web server will consume more system resources. Based on your deployment model, choose this number appropriately.

**ServerID** To avoid revealing the server software information to malicious users, you may want to create a custom server name that obfuscates the Web Server and operating system versions.

### **7.3.8.3. Recommendations**

A typical deployment uses a Web server in a private network to provide a remote user interface to configure a headless device. The registry defines the number of connections and when the **MaxConnections** registry value is not set, the registry limits the number to 10.

A typical deployment uses the Web server to display status information or to host a family or community Web site. You should not use the Web server to perform critical operations, such as machine control or financial processing.

Use NTLM and/or Basic authentication mechanism to limit access to known users only. You can set the option in the

**HKEY\_LOCAL\_MACHINE\COMM\HTTPD** registry key.

SSL protocol helps to protect data from packet sniffing by anyone with physical access to the network.

Carefully choose your virtual roots and limit access to the appropriate files by providing appropriate user access lists. Anonymous users with access to the virtual root may be able to access files and directories within that virtual root.

You can set the options in

**HKEY\_LOCAL\_MACHINE\COMM\HTTPD\ROOTS** registry key.

### **7.3.9. Auto RUN**

When Windows CE begins loading, the kernel starts the file system and examines the **HKEY\_LOCAL\_MACHINE\Init** registry key to identify what applications to run. To control which applications run at system startup, create launch registry values. Launch registry values do not need to be stored in the registry, although you can specify dependencies. You can specify up to 32 applications.

**Table 7.3.7. Named values of HKEY\_LOCAL\_MACHINE\Init Key**

<b>HKEY_LOCAL_MACHINE\Init</b>	
<b>Value</b>	<b>Description</b>
<b>Launch<math>nn</math> : String</b>	<i>Specifies the application to launch in order “nn”</i>
<b>Depend<math>nn</math> : Binary</b>	<i>Launch<math>nn</math> registry values have optional dependencies as denoted by the Depend<math>nn</math> registry value.</i>

**Depend $nn$**  registry values specify applications that Windows CE must be running before the **Launch $nn$**  applications run.

**Depend $nn$**  registry values begin with the keyword **Depend**, followed by the same decimal number as the **Launch $nn$**  registry value.

The **Depend $nn$**  registry values define an order in which Windows CE launches applications. One or more dependent applications can be specified per **Depend $nn$**  value. Dependent applications are specified as a series of Words in hexadecimal notation.

**Table 7.3.8. Typical Init Registry Entry Using Dependencies**

<b>HKEY_LOCAL_MACHINE\Init</b>	
<b>Value Name</b>	<b>Value</b>
<b>Launch20</b>	<i>device.dll</i>
<b>Launch30</b>	<i>gwes.dll</i>
<b>Depend30</b>	<i>2 length binary value</i>
<b>Launch50</b>	<i>explore.exe</i>

<b>Depend50</b>	<i>4 length binary value</i>
<b>Launch60</b>	<i>serviceStart.exe</i>
<b>Depend60</b>	<i>2 length binary value</i>

If your applications have dependencies with other applications, use the registry editor to setup your applications manually. After finishing the registry setup, the system will automatically flush the registry on changes to HIVE.

### **7.3.10. COM Ports**

#### **7.3.10.1. Introduction**

The MXM-6410 and his evaluation kit names the serial ports as “COMx”. Note the **COM\_IOCTL\_SET\_OPERATION\_MODE** call will be returned with error if trying to configure a non-configurable port.

**Table 7.3.9. Related Win32 APIs to Configure COMx**

<b>Device Name: COMx:</b>	
<b>Win32 Functions</b>	<b>Description</b>
<b>CreateFile</b>	<i>Opens COMx devices.</i>
<b>CloseHandle</b>	<i>Closes COMx device.</i>
<b>DeviceIoControl</b>	<i>Calls a customized IOCTL function.</i>

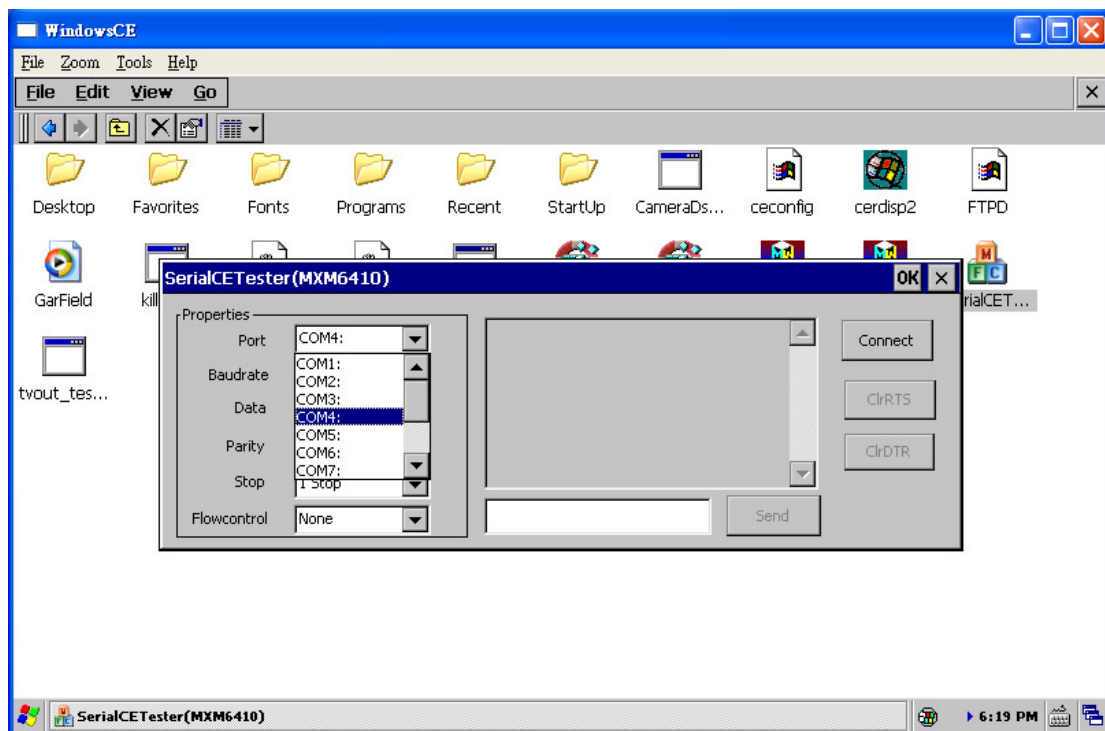
**Table 7.3.10. IO Control Codes for COMx**

<b>Device Name: COMx:</b>	
<b>IO Control Codes</b>	<b>Description</b>
<b>COM_IOCTL_GET_OPERATION_MODE</b>	<i>This IOCTL is used to get the operation mode.</i>
<b>COM_IOCTL_SET_OPREATION_MODE</b>	<i>This IOCTL is used to set the operating mode.</i>

**7.3.10.2. Test COM Ports**

To test COM ports, user ActiveSync program to put the *SerialCETester.exe* that Embedian offered into device \Windows directory and open it. You will see as shown figure 7.3.22.

**Figure 7.3.22 COM Port Test Program**



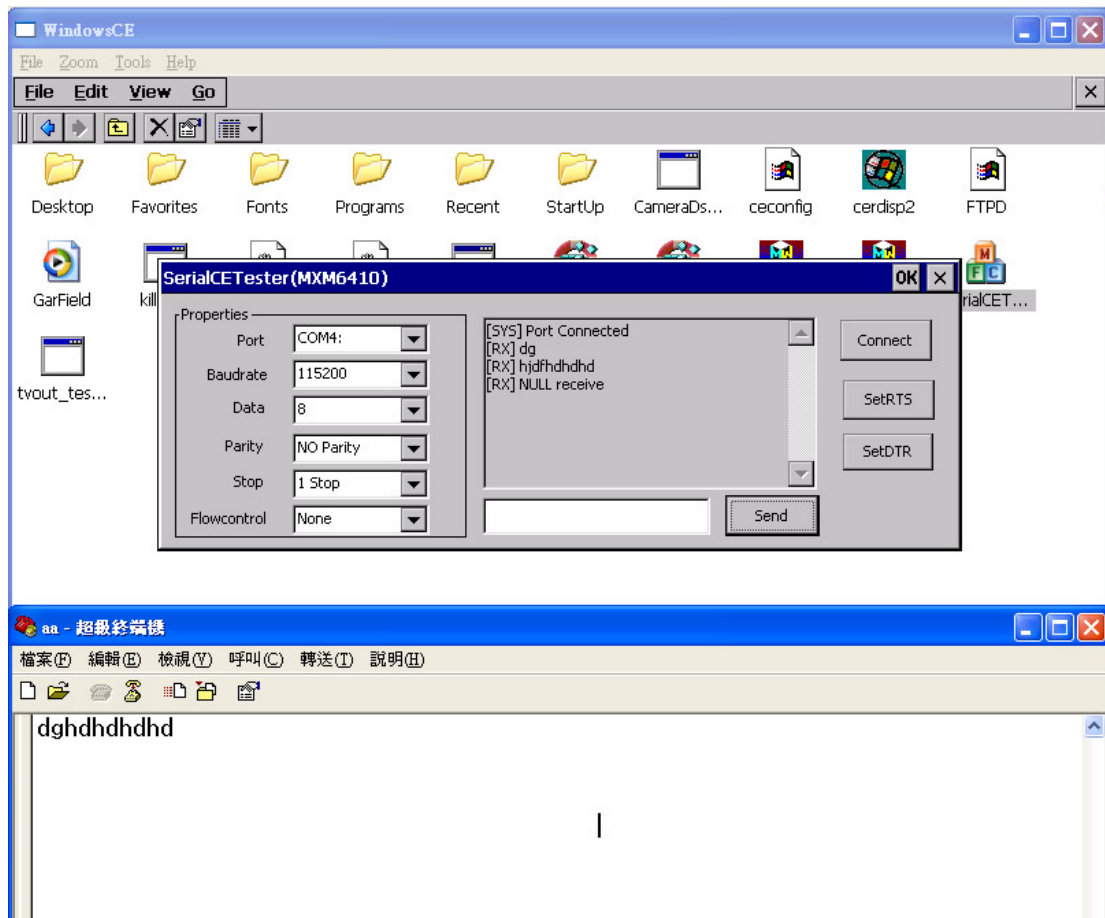
At the PC side, open the hyper terminal program and set the COM port baud rate, configuration and hardware flow control.

At device side, choose which COM port you would like to test and set the COM port baud rate, configuration and hardware flow control exactly the same as that you set in PC Hyperterminal.

Press “Connect” and type and characters from PC. You should be able to receive those characters from the CE device.

Type any characters in the dialog box of the CE test program and then click “Send”. You should be able to receive them in the PC Hyperterminal program. This is shown in figure 7.3.23.

**Figure 7.3.23 COM Port Test Result**



### **7.3.11. Software Installation**

Applications installed on to the Embedian Windows CE products will not remain on the device after rebooting the system. This is due to the applications being installed to the \Windows and \Program Files folders. The \Windows and \Program Files folders are not persistent. For this reason, users can integrate the software from the platform builder and integrate the software into the NK or users can use the batch file to implement that. The following steps must be taken to restore the files to the \Windows and \Program Files folders.

#### **7.3.11.1. Software Installation using a Batch File**

##### **Software Installation**

1. Follow the standard installation of the software application to the Windows CE device.
2. Test the software application.
3. From the Desktop run the application and Save the Registry.
4. Connect the Windows CE device to a PC with ActiveSync.
5. Create a folder on your hard drive of the PC.
6. Copy the files from the Windows CE device to the folder on the hard drive.
  - Directories to check:
    - \Windows
    - \Program Files
    - \Windows\Programs
7. Create a batch file to copy the files to the Windows CE device.
  - Be sure to copy the files to where they were placed during the installation.
  - Folders will need to be created on the Windows CE device to match the ones created during the install.
8. Copy all of the files and the batch file on to a SD/SDHC card.
9. Place the SD/SDHC card in to the Windows CE device.
  - To debug the batch file place a pause as the last line in the file. Then when the batch file runs you will see the results on the Windows CE device.
10. On the Windows CE device run the batch file.
11. Test your software application.
12. Create a Startup folder in the **\NandFlash** folder.
13. Paste a Shortcut to the batch file on the **\StorageCard**.
14. From the Desktop run the application and Save the Registry.
15. Reboot the system.
  - The shortcut should run the batch file from the SD/SDHC card.
16. Test your software application.



**SAMPLE.BAT**

```
md \Windows\Programs
Copy <filename> \Windows\Programs
md \Program Files\<appname>
Copy <filename> \Program Files\<appname>
Pause
<filename> = the file you wish to copy
<appname> = the application name
md = Make Directory command line function (DOS)
```

*Note1: Once the batch file is debugged remove the Pause from the last line and add Echo off as the first line. Echo off will not display the batch file running.*

*Note2: The only way to make the applications permanent on the Windows CE device is to create a new Windows CE Operating Systems Image. User can custom your own Windows CE Images by the BSP that Embedian supplied.*

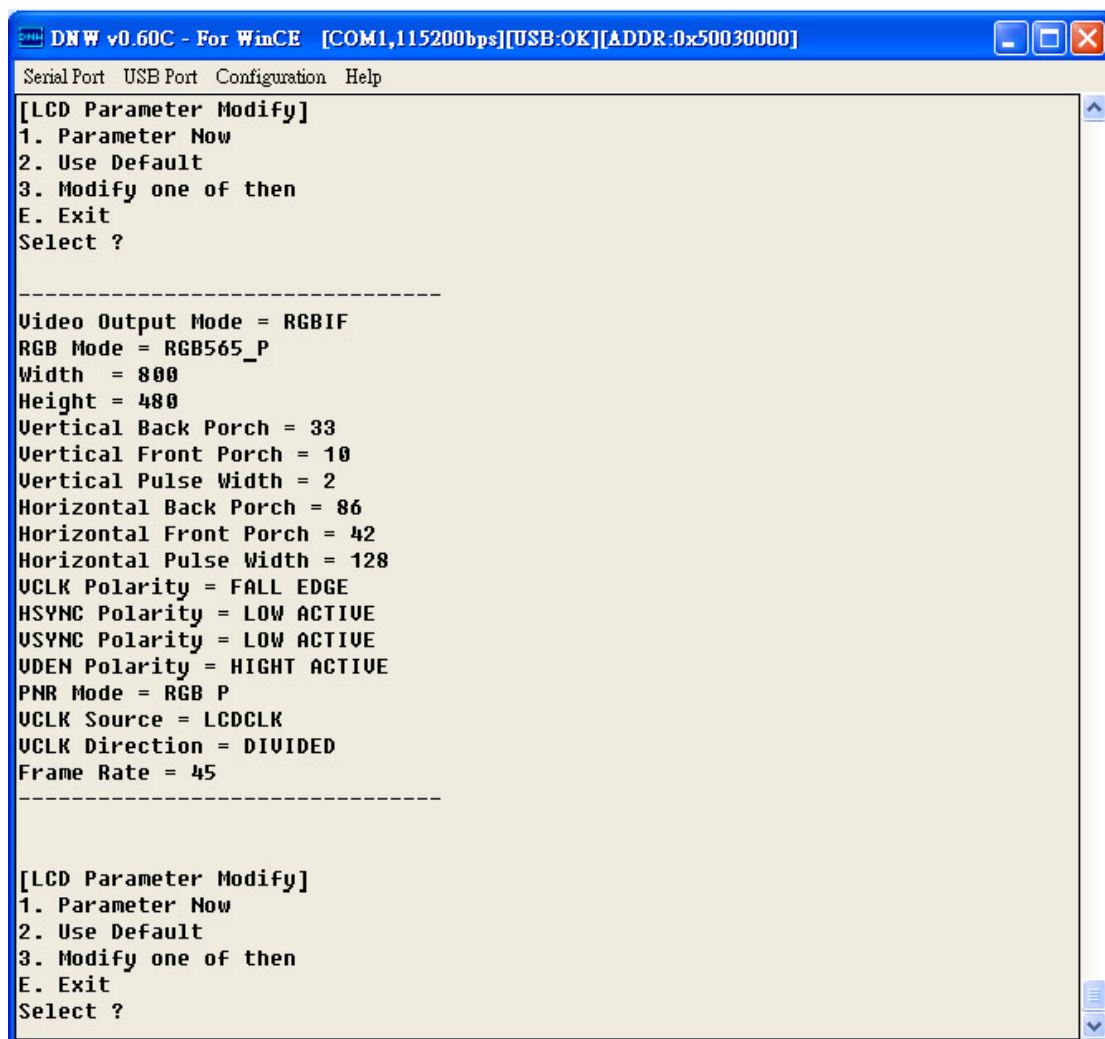
## **7.4. Configure LCD parameters for different kinds of LCDs**

Users don't need to build from BSP to configure LCD parameters of various types. Embedian allow users to configure the LCD parameters at EBOOT menu, after configuring, EBOOT will pass those parameters to NK. Embedian also reserves some very common types LCD parameters as default setting. Users can just choose it instead of setting parameters each by each. Embedian also allow users to view the current LCD parameter settings.

### **7.4.1. View Current LCD Parameters**

To view current LCD parameters, go to EBOOT menu and press "**Z) MXM EVB Board SubTestMenu**" and then press "**2. LCD Panel Parameter Modify**" and then press "**1. Parameter Now**" as shown in figure 7.4.1

**Figure 7.4.1 View Current LCD Parameters**

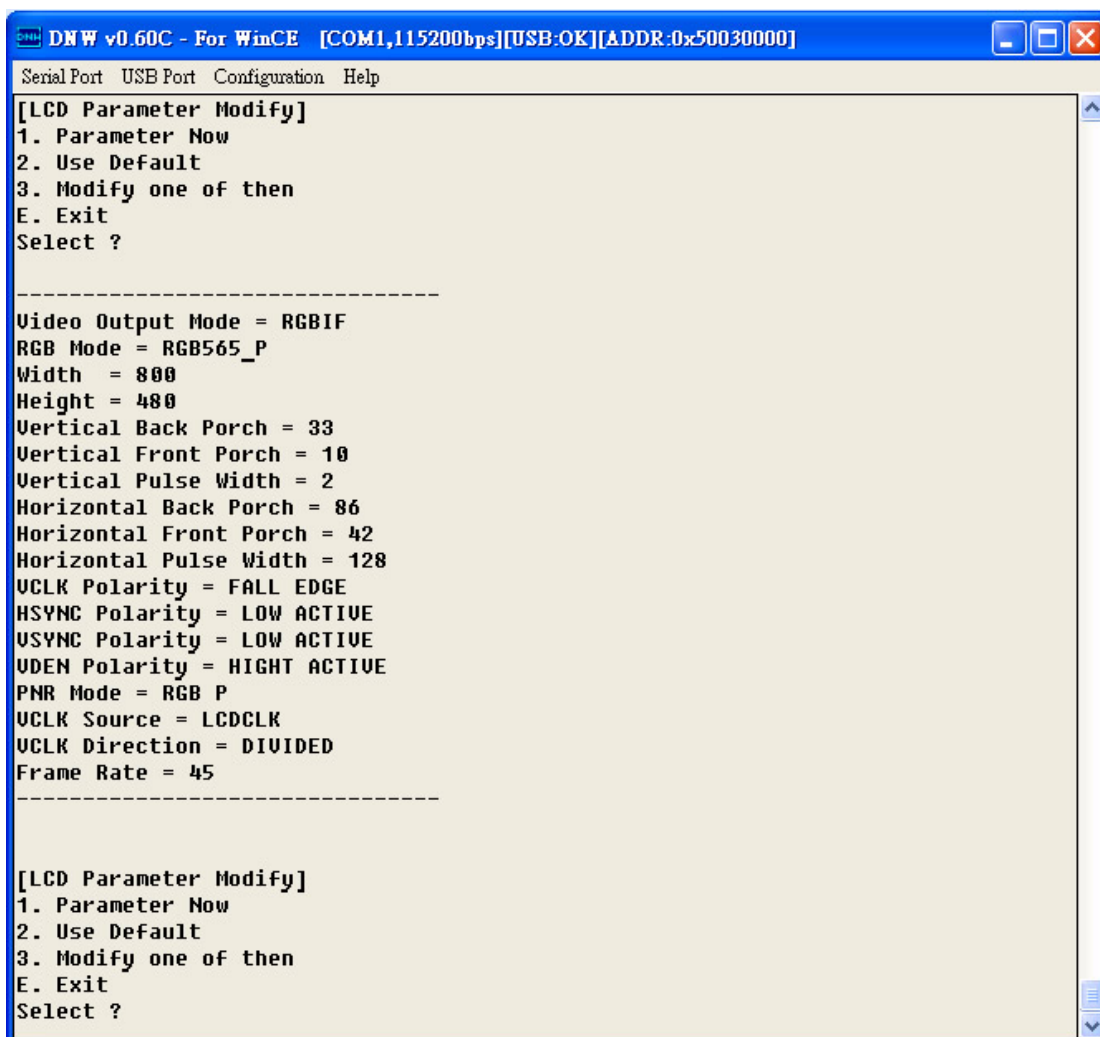


#### **7.4.2. Choose Default LCD Parameters**

Embedian chooses Data Image 5.7-inch FG050605DNSWAG01 (320x234) and PVI 7-inch PM070WT3 (800x480) as default LCD parameters because they are also compatible to many LCDs with same resolutions.

To set the default LCD parameters, go to EBOOT menu and press "**Z) MXM EVB Board SubTestMenu**" and then press "**2. LCD Panel Parameter Modify**" and then press "**2. Use Default**". You will see two types of LCD for your choices. Choose the correct type that you will use. Following figure shows the default LCD parameters.

**Figure 7.4.2 Default LCD Parameters**



To save the configuration, press “E” exit to main menu and press “**W) Write Configuration Right Now**” to save it.

### **7.4.3. Set LCD Parameters of Different Types**

Embedian provides users with an easy way to configure parameters for various types of LCDs.

To configure your LCD parameters, go to EBOOT menu and press “**Z) MXM EVB Board SubTestMenu**” and then press “**2. LCD Panel Parameter Modify**” and then press “**3. Modify one of then**”. You will see the following menu

**[LCD Parameter Modify by each]**

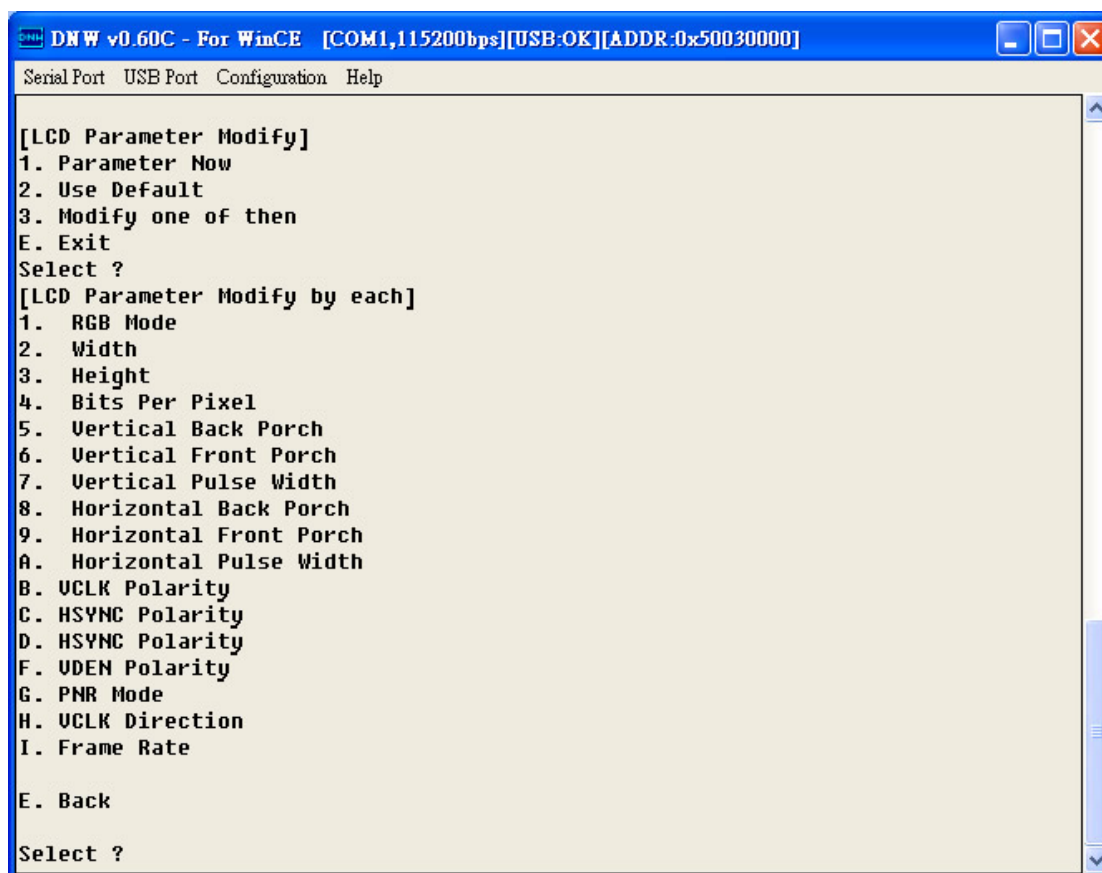
- 1. RGB Mode**
- 2. Width**

- 3. Height**
- 4. Bits Per Pixel**
- 5. Vertical Back Porch**
- 6. Vertical Front Porch**
- 7. Vertical Pulse Width**
- 8. Horizontal Back Porch**
- 9. Horizontal Front Porch**
- A. Horizontal Pulse Width**
- B. VCLK Polarity**
- C. HSYNC Polarity**
- D. HSYNC Polarity**
- F. VDEN Polarity**
- G. PNR Mode**
- H. VCLK Direction**
- I. Frame Rate**

**E. Back**

You can set each of them by just choose them. Following figure shows the setting of each parameter.

**Figure 7.4.3 Configure LCD Parameters**



After done with the setting, press “E” to exit to the main menu and press “W” to save the parameters. The EBOOT will pass the new LCD parameters to NK and you should see the new LCD configuration at next boot.

## 7.5. Configure GPIOs, BACKLIGHT\_EN, LCD\_PWREN and VDDLCD\_PWREN

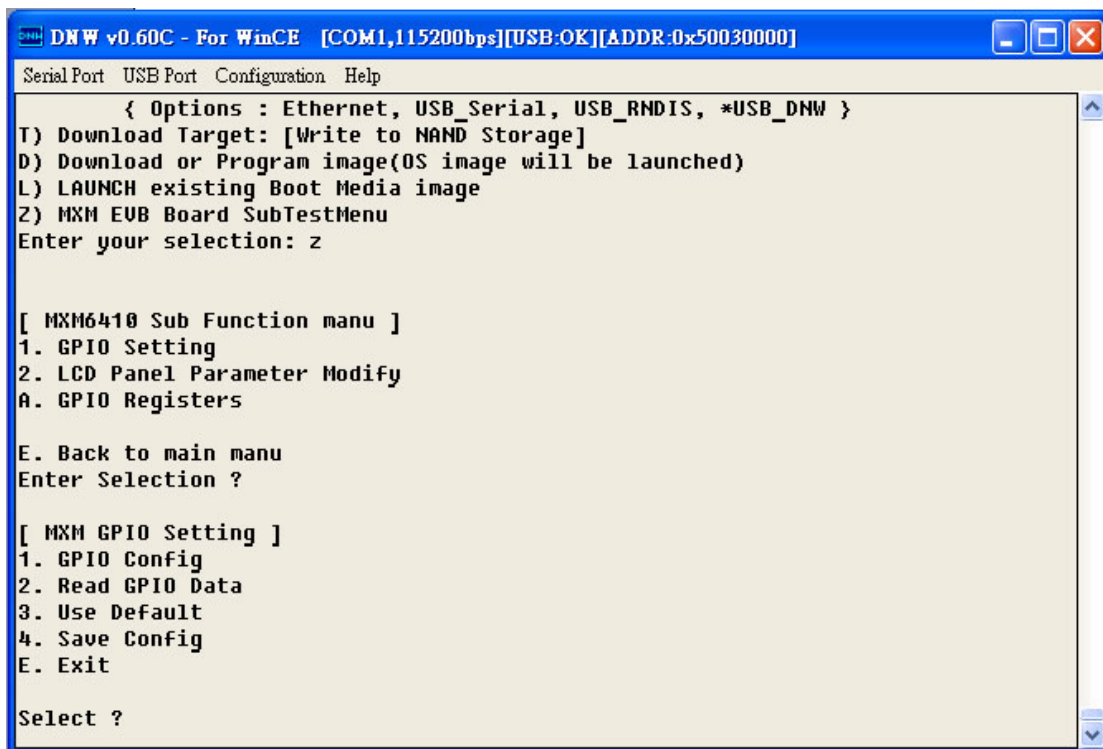
Embedian reserves 12 free available GPIOs as well as BACKLIGHT\_EN signal, LCD\_PWREN signal and VDDLCD\_PWREN for user configuration. Users can set the GPIOs as input or output, high or low, pull-up or pull-down at Eboot or during the NK.

### 7.5.1. Configure GPIOs Setting at EBOOT

Embedian provides users with an easy way to configure GPIO settings for free available GPIOs at EBOOT.

To configure the GPIO settings, go to EBOOT menu and press “**Z) MXM EVB Board SubTestMenu**” and then press “**1. GPIO Setting**”. Users will see GPIO configuration menu as shown in figure 7.5.1.

**Figure 7.5.1 GPIO Setting Menus**



The “**1. GPIO Config**” option can let users configure each GPIO pin as input or output, pull-up or pull-down and high or low.

The “**2. Read GPIO Data**” allows users to read the GPIO data if the GPIO is set to be input and connect to the outer world or if the GPIO is set to be output.

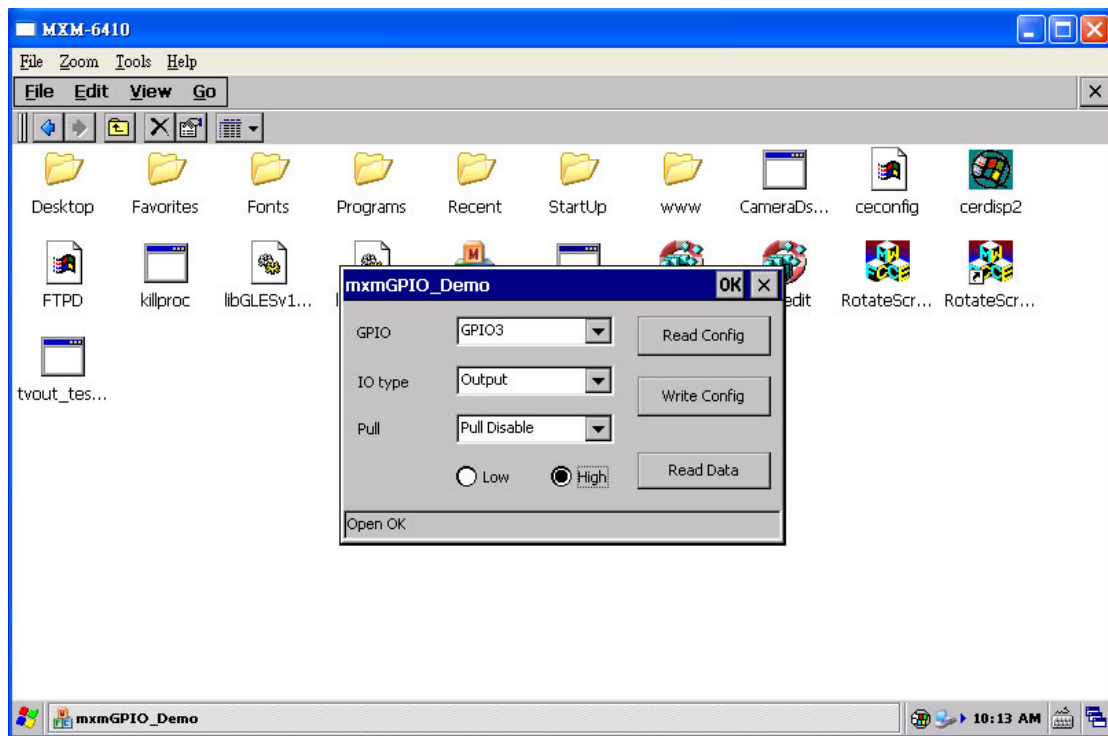
***Embedian, Inc.***

The “**3. Use Default**” will set the GPIO 1 to GPIO 12 input pull-down. BACKLIGHT\_EN, LCD\_PWREN and VDDLCD\_PWREN signals are set to output pull-disable high by default.

### 7.5.2. Configure GPIOs Setting at NK

Embedian provides users with a library and sample demo code to allow user application change the GPIO setting at NK. The control applet named “**mxmGPIO\_demo.exe**” provides a simple user interface to diagnostic the functionality of the digital input and digital output channels.

**Figure 7.5.2 GPIO Demo Program**



**Note:**

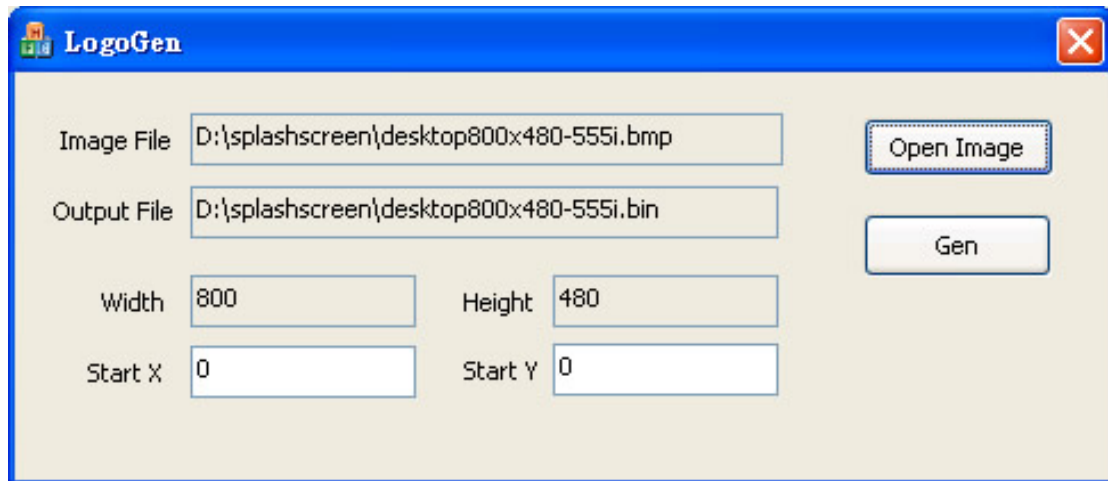
1. The sample code of the GPIO demo program is available at Embedian FTP site.
2. The “Read Config” button at the right hand side of the program will read back the current GPIO setting.
3. If the IO type is “Output”, the “Write Config” button at the right hand side of the program will set the port to “Low” first by default and it will change to high when user click the “High” radio button.
4. When the IO type is set to “Input”, users can know the setting of outer world by clicking “Read Data” button.



## **7.6. Logo Splash Screen Customizer**

Embedian provides a way to allow users to customize their own boot splash screen at EBOOT. To achieve that, users need to prepare for a bmp 555i file format first. And then use the “LogoGen.exe” program than Embedian provided to convert the 555i file format into an EBOOT readable .bin file format.

**Figure 7.6.1 LogoGen Program**

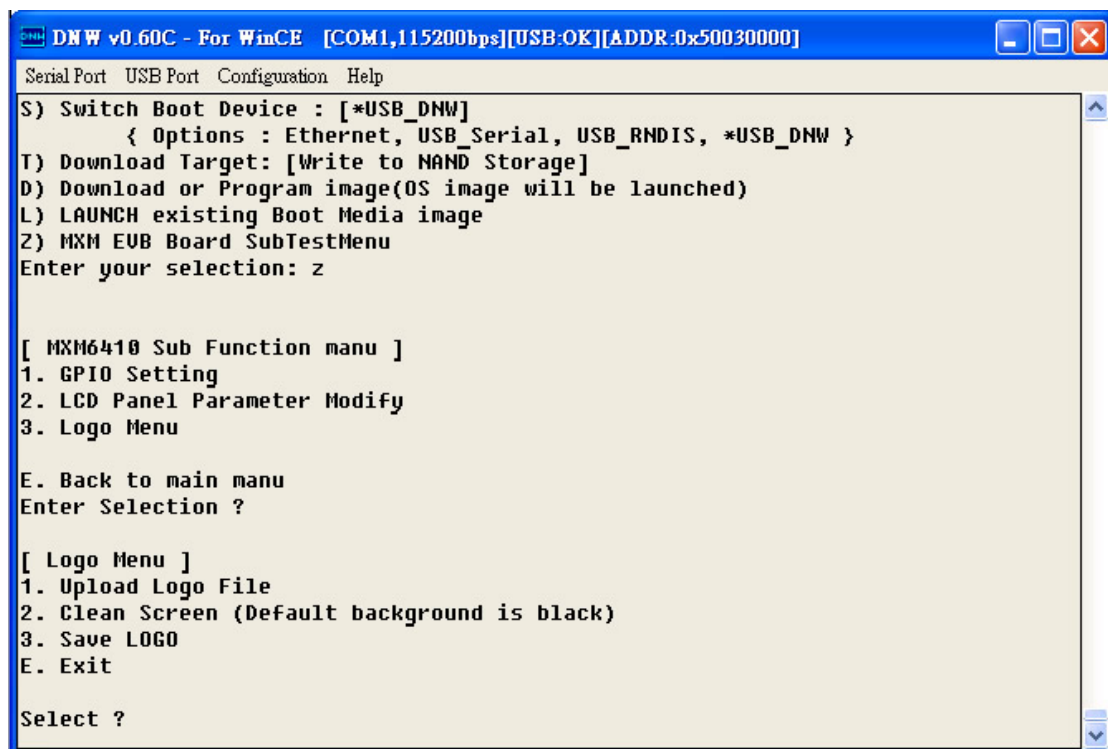


The program allows users set the image width, height, startX and startY parameters and convert to a EBOOT readable .bin file format.

### **7.6.1. Upload and save splash screen image into devices**

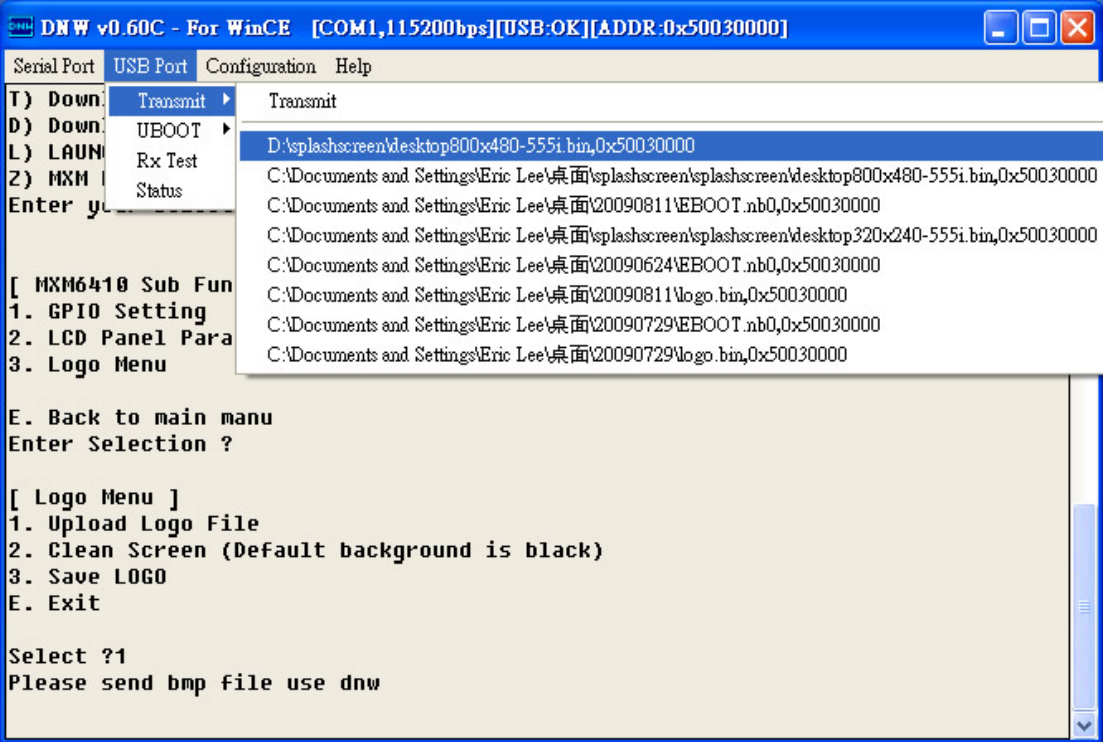
To upload the splash screen image into device, go to EBOOT menu and press “**Z) MXM EVB Board SubTestMenu**” and then press “**3. Logo Menu**”. Users will see sub menu as shown in figure 7.6.2.

**Figure 7.6.2 EBOOT LOGO Menu**



The “**1. Upload Logo File**” option can let users upload the splash screen that just created into the device and shown on the LCD screen immediately. To achieve that, at “**USB Port**” → “**Transmit**” of DNW program, choose the splash screen that just created and you will see the new splash screen shown on the LCD screen.

Figure 7.6.3 Upload Splash Screen



The “**2. Clean Screen (Default background is black)**” allows users to clean the splash screen that you just uploaded.

The “**3. Save LOGO**” will save the splash screen to NAND flash permanently.

**Note:** The maximum size of splash screen now is limited at 800x600x2 = 768000Byte. For file larger than this size will erase the EBOOT image.

Reboot the device and you should be able to see the new splash screen.

**Figure 7.6.4 New Splash Screen**



Next, we would like to give topics related to hardware MFC multimedia function in MXM-6410.

### **7.7. WMV9 Decoder for Windows Media Player**

The Windows media player in Windows CE 6.0 devices supports WMV9 hardware decoder. And Microsoft provides with a free WMV9 PC encoding tool to encode the video sources. First of all, we need to encode the video source as WMV9 format.

#### **7.7.1. Encode Video source as WMV9**

Users can download the free Windows Media Encoding tool that Microsoft provided with from

<http://www.microsoft.com/downloads/details.aspx?FamilyID=5691ba02-e496-465a-bba9-b2f1182cdf24&displaylang=en>

The recommended resolutions for MXM-6410 hardware MFC is 640x480 or 720x480. And the recommended bit rate is around 2000Kbps or less and

frame rate is 30fps. It will take a while to encode the video sources. After encoding, you should be able to see a .wmv output file at the target directory of your PC.

#### ***7.7.2. WMV9 Decoder for Windows Media Player***

Using ActiveSync, upload the wmv file into the device **\NandFlash** or **\Storage Card** directory. At devices, go to **My Device → NandFlash** or **My Device → Storage Card** and double click the file that you just upload. You will see the Media Player software playing the video.

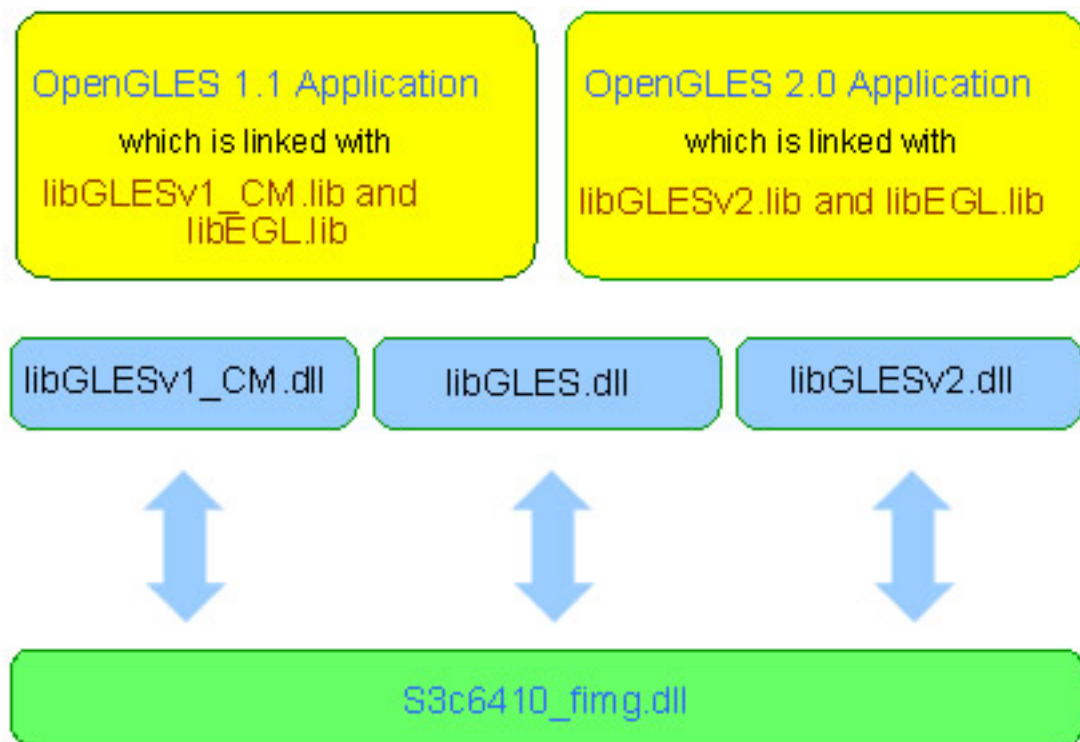
**Note:** Please remember to close remote display tool when playing video because the remote display tool takes a lot of bandwidth.

## 7.8. OpenGL User's Manual

### 7.8.1. Software Layers

OpenGL driver has two layers. One is for mapping H/W and allocating memory. The other one is for OpenGL libraries.

**Figure 7.8.1 MXM-6410 OpenGL Software Layer**



The Figure 7.8.1 shows this driver structure. S3C6410\_fimg.dll is mapping and allocating layer. This driver maps H/W address for library. The library uses this mapped address. And it also allocates physically continuous memory for library and FIMG H/W. If the library requests some texture memory and depth buffer, then this driver dynamically allocates some memory and returns the address.

The libEGL.lib implements EGL 1.3. This will work with OpenGL1.1 and OpenGL2.0

The libGLESv1\_CM.dll implements OpenGL1.1. If you want to know more about OpenGL 1.1, please visit [www.opengl.org](http://www.opengl.org) website.

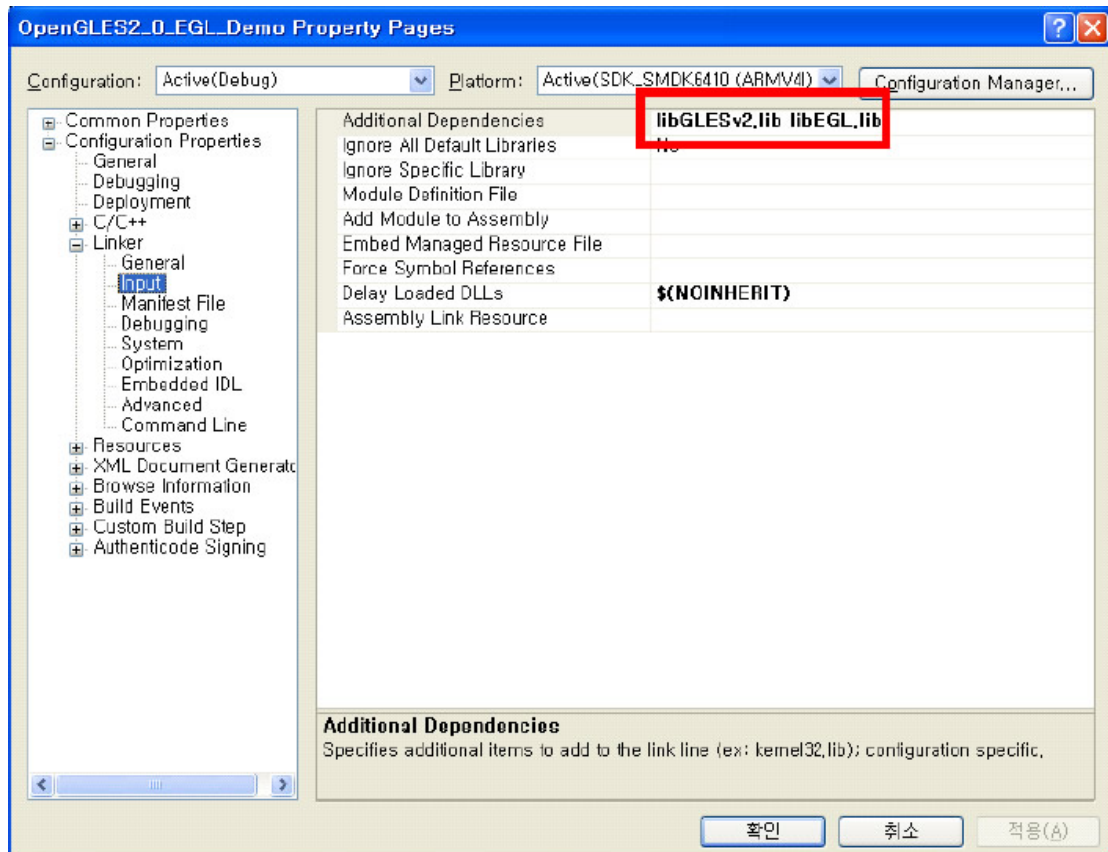
The libGLESv2.dll implements OpenGL2.0. If you want to know more about OpenGL 1.1, please visit [www.opengl.org](http://www.opengl.org) website.

### **7.8.2. Usage**

For using OpenGL ES, You need to link libEGL.lib and libGLESv2.lib libraries.

In Project property of Visual Studio 2005, set Additional Dependency like the following figure 4.2.

**Figure 4.2 Setting Property for OPENGLES2.0 in VS2005**



If you need to use OpenGL ES 1.1, then you need to input libGLESv1\_CM.lib instead of libGLESv2.lib.

### **7.8.3. Shader Compile**

You need to use orion compiler for compiling shader. You can execute attached compiler on MS Windows XP and VISTA.

Ex)

orion -O -f <Fragment Shader File>

orion -O -v <Vertex Shader File>

orion -O -a -f <Fragment Shader ASM File>

orion -O -a -v <Fragment Shader ASM File>

If you type just “orion”, you can see more detail information about this shader compiler.

### **7.8.4. Dll location**

libEGL.dll, libGLv1\_CM.dll and libGLv2.dll should be in \Windows directory or same directory with your application. Second has higher priority.

### **7.8.5. Samples**

#### **7.8.5.1. OpenGL ES 1.1 Demo Application**

The demo program is located at Embedian ftp site and it is a very simple opengles demo. You can get example about OpenGL ES application and compile environment from source code.

You can modify line 80 “int selectedDemo” from 0 to 7.

#### **7.8.5.2. OpenGL ES 2.0 Demo Application**

The demo program is located at Embedian ftp site and it is a very simple opengles demo. You can get example about OpenGL ES application and compile environment from source code.

You can modify line 59 “int selectedDemo” from 0 to 2.



# Chapter

# 8

## **Use APC-6410 Hardware MFC Multimedia Function**

This Chapter gives how to use the hardware MFC multimedia function.

Section include :

- MPEG4 Decoder for Mplayer at Device
- MFC Device Driver's API

# Chapter 8 Use MXM-6410 Hardware MFC Multimedia Function

This Chapter gives topics related to hardware MFC multimedia function in MXM-6410.

## 8.1. MFC Device Driver's API

<b>API Functions</b>	<b>Description</b>
<b>CreateFile</b>	Create the MXM-6410 MFC instance.
<b>DeviceIoControl</b>	IOCTL_MFC_MPEG4_DEC_INIT IOCTL_MFC_MPEG4_ENC_INIT IOCTL_MFC_MPEG4_DEC_EXE IOCTL_MFC_MPEG4_ENC_EXE IOCTL_MFC_H264_DEC_INIT IOCTL_MFC_H264_ENC_INIT IOCTL_MFC_H264_DEC_EXE IOCTL_MFC_H264_ENC_EXE IOCTL_MFC_H263_DEC_INIT IOCTL_MFC_H263_ENC_INIT IOCTL_MFC_H263_DEC_EXE IOCTL_MFC_H263_ENC_EXE IOCTL_MFC_VC1_DEC_INIT IOCTL_MFC_VC1_DEC_EXE IOCTL_MFC_GET_LINE_BUF_ADDR IOCTL_MFC_GET_RING_BUF_ADDR IOCTL_MFC_GET_FRAM_BUF_ADDR
<b>CloseHandle</b>	Close the 6410 MFC instance.

**8.1.1. CreateFile**

<b>CreateFile</b>	
<b>Syntax</b>	<i>HANDLE</i> WINAPI CreateFile( <i>LPCTSTR</i> lpFileName, <i>DWORD</i> dwDesiredAccess, <i>DWORD</i> dwShareMode, <i>LPSECURITY_ATTRIBUTES</i> lpSecurityAttributes, <i>DWORD</i> dwCreationDisposition, <i>DWORD</i> dwFlagsAndAttributes, <i>HANDLE</i> hTemplateFile );
<b>Description</b>	This function creates the MFC instance. Several MFC instance can be made simultaneously. This means that CreateFile function can be called several times in a process (task).
<b>Parameters</b>	<i>lpFileName</i> [IN] : MFC's device driver name. (L"MFC1:") <i>dwDesiredAccess</i> [IN] : GENERIC_READ GENERIC_WRITE <i>dwShareMode</i> [IN] : 0 <i>lpSecurityAttributes</i> [IN] : NULL <i>dwCreationDisposition</i> [IN] : OPEN_EXISTING <i>dwFlagsAndAttributes</i> [IN] : FILE_ATTRIBUTE_NORMAL <i>hTemplateFile</i> [IN] : NULL
<b>Returns</b>	<i>HANDLE</i> of the MFC instance. If it fails, it returns INVALID_HANDLE_VALUE.

### 8.1.2. DeviceIoControl

<b>DeviceIoControl</b>	
<b>Syntax</b>	<pre> BOOL WINAPI DeviceIoControl(     HANDLE hDevice,     DWORD dwIoControlCode,     LPVOID lpInBuffer,     DWORD nInBufferSize,     LPVOID lpOutBuffer,     DWORD nOutBufferSize,     LPDWORD lpBytesReturned,     LPOVERLAPPED lpOverlapped ); </pre>
<b>Description</b>	Most of functions are developed in ioctl. This system call has many functions which is separated by dwIoControlCode
<b>Parameters</b>	<p><i>hDevice [IN] : HANDLE returned by CreateFile() function</i>  <i>dwIoControlCode [IN] : The control code for the operation. Detailed information will explain below.</i>  <i>lpInBuffer [IN] : Structure of the MFC argument</i>  <i>nInBufferSize [IN] : Size of MFC argument structure</i>  <i>lpOutBuffer [OUT] : NULL</i>  <i>nOutBufferSize [OUT] : 0</i>  <i>lpBytesReturned [OUT] : NULL</i>  <i>lpOverlapped [IN] : NULL</i></p>
<b>Returns</b>	<p>If the operation completes successfully, the return value is nonzero.  If the operation fails or is pending, the return value is zero.</p>

### **8.1.3. CloseHandle**

<b>CloseHandle</b>	
<i>Syntax</i>	<i>BOOL WINAPI CloseHandle( HANDLE hDevice );</i>
<i>Description</i>	Closes an open MFC's handle.
<i>Parameters</i>	<i>[IN] hDevice</i> - HANDLE returned by CreateFile() function
<i>Returns</i>	If the function succeeds, the return value is nonzero. If the function fails, the return value is zero

#### 8.1.4. Control Codes for DeviceIoControl()

<b>IOCTL_MFC_MPEG4_DEC_INIT</b> <b>IOCTL_MFC_H263_DEC_INIT</b> <b>IOCTL_MFC_H264_DEC_INIT</b> <b>IOCTL_MFC_VC1_DEC_INIT</b>	
<b>Syntax</b>	BOOL WINAPI DeviceIoControl( HANDLE hDevice, DWORD dwIoControlCode, LPVOID lpInBuffer, DWORD nInBufferSize, LPVOID lpOutBuffer, DWORD nOutBufferSize, LPDWORD lpBytesReturned, LPOVERLAPPED lpOverlapped );
<b>Description</b>	It initializes the MFC's instance with the configure stream.
<b>Parameters</b>	<i>hDevice</i> [IN] : HANDLE returned by CreateFile() function <i>dwIoControlCode</i> [IN] : IOCTL_MFC_MPEG4_DEC_INIT, IOCTL_MFC_H263_DEC_INIT, IOCTL_MFC_H264_DEC_INIT, IOCTL_MFC_VC1_DEC_INIT <i>lpInBuffer</i> [IN] : Pointer to MFC_DEC_INIT_ARG structure. <i>nInBufferSize</i> [IN] : sizeof(MFC_DEC_INIT_ARG) <i>lpOutBuffer</i> [OUT] : NULL <i>nOutBufferSize</i> [OUT] : 0 <i>lpBytesReturned</i> [OUT] : NULL <i>lpOverlapped</i> [IN] : NULL
<b>Returns</b>	If the operation completes successfully, the return value is nonzero. If the operation fails or is pending, the return value is zero.

<b>IOCTL_MFC_MPEG4_DEC_EXE</b> <b>IOCTL_MFC_H263_DEC_EXE</b> <b>IOCTL_MFC_H264_DEC_EXE</b> <b>IOCTL_MFC_VC1_DEC_EXE</b>	
<b>Syntax</b>	BOOL WINAPI DeviceIoControl( HANDLE hDevice, DWORD dwIoControlCode, LPVOID lpInBuffer, DWORD nInBufferSize, LPVOID lpOutBuffer, DWORD nOutBufferSize, LPDWORD lpBytesReturned, LPOVERLAPPED lpOverlapped );
<b>Description</b>	It decodes the stream in the LINE_BUF or RING_BUF.
<b>Parameters</b>	<i>hDevice</i> [IN] : HANDLE returned by CreateFile() function <i>dwIoControlCode</i> [IN] : IOCTL_MFC_MPEG4_DEC_EXE, IOCTL_MFC_H263_DEC_EXE, IOCTL_MFC_H264_DEC_EXE, IOCTL_MFC_VC1_DEC_EXE <i>lpInBuffer</i> [IN] : Pointer to MFC_DEC_EXE_ARG structure. <i>nInBufferSize</i> [IN] : sizeof(MFC_DEC_EXE_ARG) <i>lpOutBuffer</i> [OUT] : NULL <i>nOutBufferSize</i> [OUT] : 0 <i>lpBytesReturned</i> [OUT] : NULL <i>lpOverlapped</i> [IN] : NULL
<b>Returns</b>	If the operation completes successfully, the return value is nonzero. If the operation fails or is pending, the return value is zero.

<b><i>IOCTL_MFC_GET_LINE_BUF_ADDR</i></b> <b><i>IOCTL_MFC_GET_RING_BUF_ADDR</i></b> <b><i>IOCTL_MFC_GET_FRAM_BUF_ADDR</i></b>	
<b><i>Syntax</i></b>	<b><i>BOOL WINAPI DeviceIoControl(</i></b> <b><i>HANDLE hDevice,</i></b> <b><i>DWORD dwIoControlCode,</i></b> <b><i>LPVOID lpInBuffer,</i></b> <b><i>DWORD nInBufferSize,</i></b> <b><i>LPVOID lpOutBuffer,</i></b> <b><i>DWORD nOutBufferSize,</i></b> <b><i>LPDWORD lpBytesReturned,</i></b> <b><i>LPOVERLAPPED lpOverlapped</i></b> <b><i>);</i></b>
<b><i>Description</i></b>	It obtains the address of the LINE_BUF, RING_BUF or FRAM_BUF.
<b><i>Parameters</i></b>	<i>hDevice [IN] : HANDLE returned by CreateFile() function</i> <i>dwIoControlCode [IN] :</i> <b><i>IOCTL_MFC_GET_LINE_BUF_ADDR,</i></b> <b><i>IOCTL_MFC_GET_RING_BUF_ADDR,</i></b> <b><i>IOCTL_MFC_GET_FRAM_BUF_ADDR</i></b> <i>lpInBuffer [IN]: Pointer to MFC_GET_BUF_ADDR_ARG structure.</i> <i>nInBufferSize [IN] : sizeof(MFC_GET_BUF_ADDR_ARG)</i> <i>lpOutBuffer [OUT] : NULL</i> <i>nOutBufferSize [OUT] : 0</i> <i>lpBytesReturned [OUT] : NULL</i> <i>lpOverlapped [IN] : NULL</i>
<b><i>Returns</i></b>	If the operation completes successfully, the return value is nonzero. If the operation fails or is pending, the return value is zero.



### **8.1.5. Data Structure for Passing the IOCTL Arguments**

#### **8.1.5.1. MFC\_ENC\_INIT\_ARG**

<b>MFC_ENC_INIT_ARG</b>	
<i>int ret_code</i>	[OUT] Return code
<i>int in_width</i>	[IN] width of YUV420 frame to be encoded
<i>int in_height</i>	[IN] height of YUV420 frame to be encoded
<i>int in_bitrate</i>	[IN] Encoding parameter: Bitrate (kbps)
<i>int in_gopNum</i>	[IN] Encoding parameter: GOP Number (interval of I-frame)
<i>int in_frameRateRes</i>	[IN] Encoding parameter: Frame rate (Res)
<i>int in_frameRateDiv</i>	[IN] Encoding parameter: Frame rate (Divider)

#### **8.1.5.2. MFC\_ENC\_EXE\_ARG**

<b>MFC_ENC_EXE_ARG</b>	
<i>int ret_code</i>	[OUT] Return code
<i>int out_encoded_size</i>	[OUT] Length of Encoded video stream

#### **8.1.5.3. MFC\_DEC\_INIT\_ARG**

<b>MFC_DEC_INIT_ARG</b>	
<i>int ret_code</i>	[OUT] Return code
<i>int in_strmSize</i>	[IN] Size of video stream filled in STRM_BUF
<i>int out_width</i>	[OUT] width of YUV420 frame
<i>int out_height</i>	[OUT] height of YUV420 frame

#### **8.1.5.4. MFC\_DEC\_EXE\_ARG**

<b>MFC_DEC_EXE_ARG</b>	
<i>int ret_code</i>	[OUT] Return code
<i>int in_strmSize</i>	[IN] Size of video stream filled in STRM_BUF

#### **8.1.5.5. MFC\_GET\_BUF\_ADDR\_ARG**

<b>MFC_DEC_INIT_ARG</b>	
<i>int ret_code</i>	[OUT] Return code
<i>int in_usr_data</i>	[IN] User data for translating Kernel-mode address to Usermode address
<i>int out_buf_addr</i>	[OUT] Buffer address
<i>int out_buf_size</i>	[OUT] Size of buffer address

# Chapter

# 9

## **General PCB Design Recommendations**

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

## Chapter 9 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.

### **9.1 Nominal Board Stack-Up**

The trace impedance typically noted ( $55\ \Omega \pm 10\%$ ) 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- $\Omega$  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.

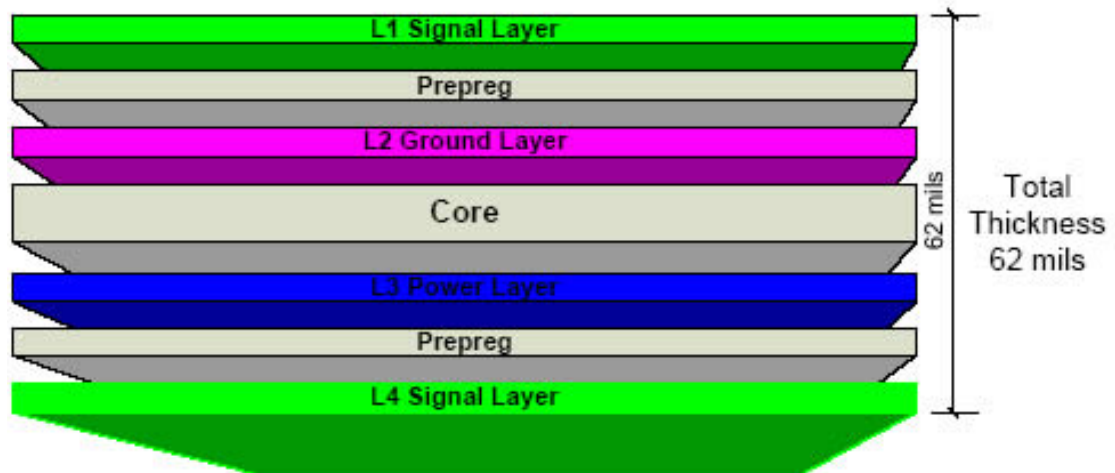
It is important to consider the minimum and maximum impedance of a trace based on the switching of neighboring traces when calculating flight times. Using wider spaces between the traces can minimize this trace-to-trace coupling. In addition, these wider spaces reduce settling time.

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.

### 9.1.1. Four Layer Board Stackup

Figure 9-1 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 9.1 Four-Layer Stack-Up**



**Table 9.1 Recommended Four-Layer Stack-Up Dimensions**

<b>Table 9.1 Recommended Four-Layer Stack-Up Dimensions</b>								
Dielectric Thickness (mil)	Layer	Layer	Signal-End Signals		Differential Signals		USB Differential Signals	
	No	Type	Width (mil)	Impedance (ohm)	Width (mil)	Impedance (ohm)	Width (mil)	Impedance (ohm)
0.7	L1	Signals	6/6	55+/-10%	6/7/6	100+/-10%	6/5/6	90+/-10%
5		Prepreg						
1.4	L2	Ground						
47		Core						
1.4	L3	Power						
5		Prepreg						
0.7	L4	Signals	6/6	55+/-10%	6/7/6	100+/-10%	6/5/6	90+/-10%

**Note:**

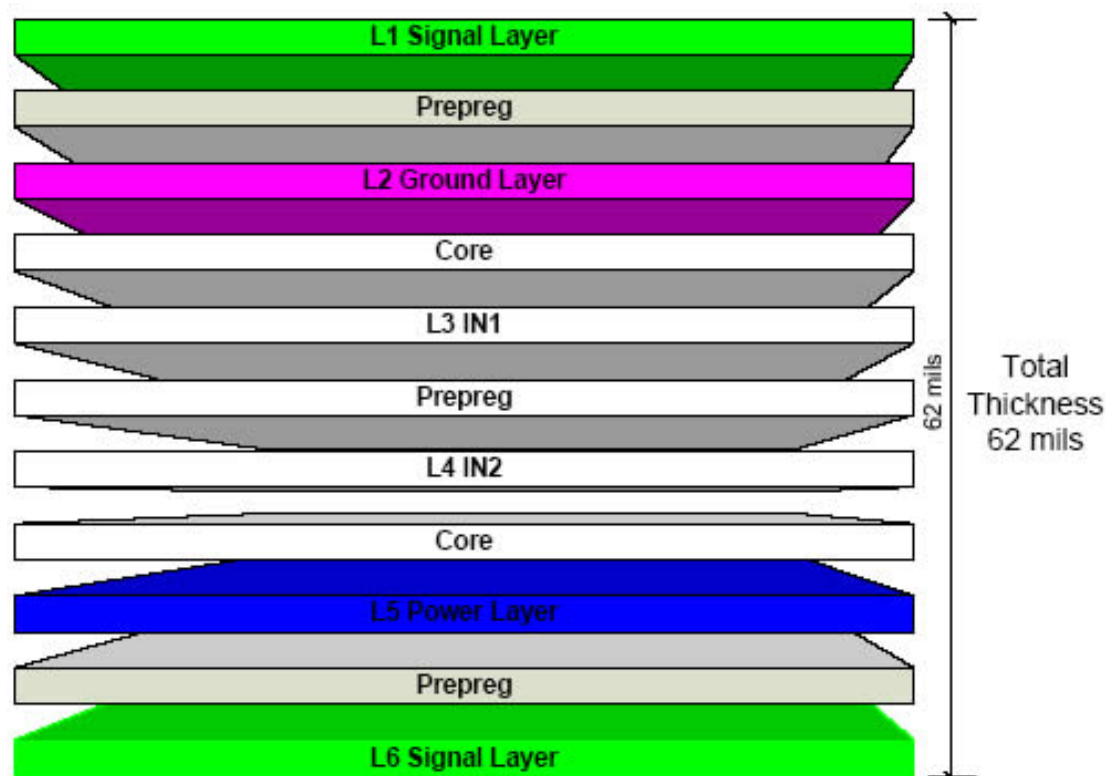
Target PCB Thickness totals 62mil+/-10%

### **9.1.2 Six Layer Board Stackup**

Figure 9-2 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 9.2 Six-Layer Stack-Up**



**Table 9.2 Recommended Six-Layer Stack-Up Dimensions**

<b>Table 9.2 Recommended Six-Layer Stack-Up Dimensions</b>								
Dielectric Thickness (mil)	Layer	Layer	Signal-End Signals		Differential Signals		USB Differential Signals	
	No	Type	Width (mil)	Impedance (ohm)	Width (mil)	Impedance (ohm)	Width (mil)	Impedance (ohm)
1.7	L1	Signals	5/5	55+/-10%	5/6/5	100+/-10%	5/4/5	90+/-10%
4		Prepreg						
1.4	L2	Ground						
5		Core						
1.4	L3	IN1	5/5	55+/-10%	4/8/4	100+/-10%	4/5/4	90+/-10%
35		Prepreg						
1.4	L4	IN2						
5		Core	5/5	55+/-10%	4/8/4	100+/-10%	4/5/4	90+/-10%
1.4	L5	Power						
4		Prepreg						
1.7	L6	Signals	5/5	55+/-10%	5/6/5	100+/-10%	5/4/5	90+/-10%

**Note:**

Target PCB Thickness totals 62mil+/-10%

## **9.2 Differential Impedance Targets for Microstrip Routing**

Table 9.3 shows the target impedance of the differential signals. The carrier board should follow the required impedance in this table.

**Table 9.3 Differential Signals Impedance Requirement**

<b>Table 9.3 Differential Signals Impedance Requirement</b>	
<b>Signal Type</b>	<b>Impedance</b>
<b>USB</b>	<b>90ohm +/- 20%</b>
<b>LAN</b>	<b>100ohm +/- 20%</b>

## **9.3 Alternative Stack Ups**

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

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

# Chapter 10

## Carrier Board Design Guidelines

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



# **Chapter 10 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.

## 10.1 General Circuit Design Guide

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

### 10.1.1. System-Wise

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

**Table 10.1. : System-Wise Circuit Design Guide**

<b>Signal Name</b>	<b>Function</b>	<b>Description</b>
XSELNAND	Select Flash Memory	Internal 10K pull up resistor
OM1	Operation mode selection	Internal 10K pull down resistor
OM2	Operation mode selection	Internal 10K pull up resistor
OM3	Operation mode selection	Internal 10K pull down resistor
OM4	Operation mode selection	Internal 10K pull down resistor
nWAIT	nWAIT Requests	Pull up 100K resistor
WAKEUP	WAKEUP Requests	Internal 4.7K pull down resistor
nRESET_IN	Reset S3C6410	Pull up 100K resistor
nRESET_OUT	Reset External Device	Pull up 100K resistor
nGCS0	Chip Select	Pull up 10K resistor
nGCS1	Chip Select	No Connect. Reversed for DM9000B
nGCS2	Chip Select	No Connect. Reversed for NAND FLASH
nGCS[3..5]	Chip Select	Pull up 10K resistor

**Table 10.2: Device operating mode selection at boot-up**

<b>XSELNAND</b>	<b>OM[4:0]</b>	<b>Boot Device</b>	<b>Function</b>	<b>Clock Source</b>
1	00000	NAND	Small Page, AddrCycle=3	OM[0] is 0  Use external crystal
1	00010		Small Page, AddrCycle=4	
1	00100		Large Page, AddrCycle=4	
1	00110		Large Page, AddrCycle=5	
X	01000	SROM(8bit)	-	
X	01010	SROM(16bit)	-	
0	01100	OneNAND	Don't use NAND Device	
X	01110	MODEM	Don't use Xm0CSn2 for SROMC	
1: NAND 0: OneNAND	11110	Internal ROM	-	

**Table 10.3. : JTAG**

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

**Table 10.4. : IIC**

<b>Signal Name</b>	<b>Function</b>	<b>Description</b>
<i>IIC_SCL</i>	<i>IIC-bus clock</i>	<i>Internal 4.7K pull up resistor</i>
<i>IIC_SDA</i>	<i>IIC-bus data</i>	<i>Internal 4.7K pull up resistor</i>

**Table 10.5. : SD**

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

**Table 10.6. : Power**

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

**Table 10.7. : USB**

<b>Signal Name</b>	<b>Function</b>	<b>Description</b>
<i>USBH- and USBH+</i>	<i>USB Host Data</i>	<i>Differential Pair</i>
<i>USBD- and USBD+</i>	<i>USB Device Data</i>	<i>Differential Pair</i>

**Table 10.8. : Ethernet**

<b>Signal Name</b>	<b>Function</b>	<b>Description</b>
<i>TX- and TX+</i>	<i>Ethernet Transmits data</i>	<i>Differential Pair</i>
<i>RX- and RX+</i>	<i>Ethernet Receives data</i>	<i>Differential Pair</i>

## 10.2 Universal Serial Bus (USB)

MXM computer modules provide two USB 1.1 ports.

### 10.2.1. 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.

### 10.2.2. Signal Description

Table 10.9 shows MXM module USB signals, including pin number, signals, I/O and descriptions.

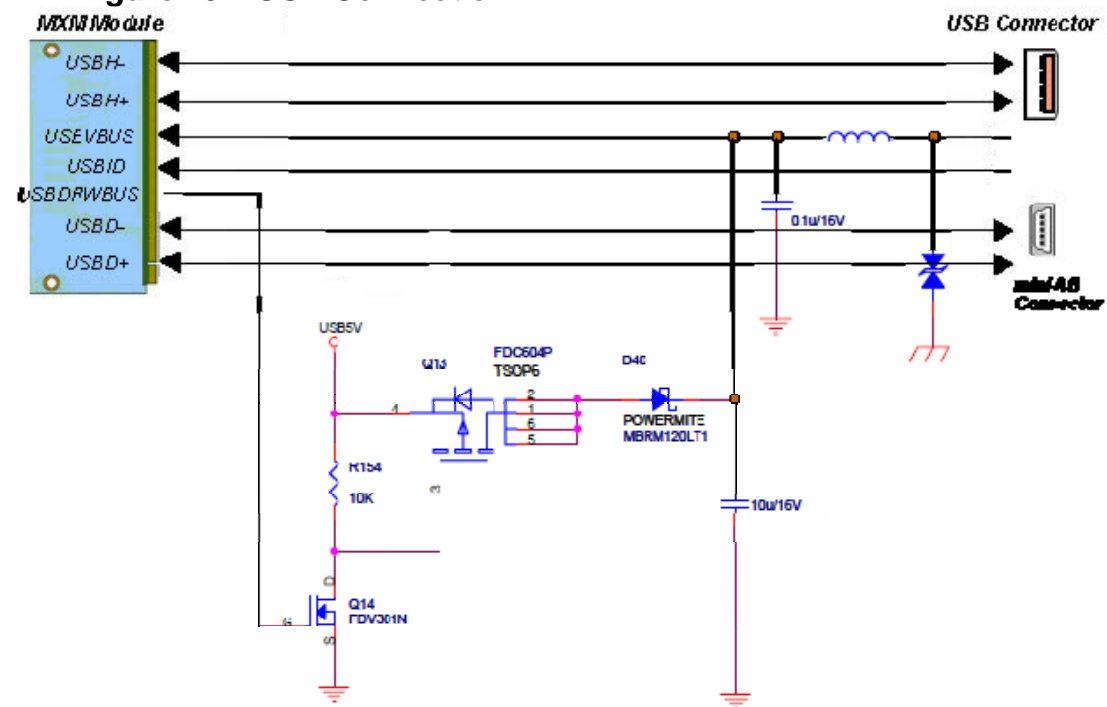
**Table 10.9 Differential Signals Impedance Requirement**

<b>Table 10.9 USB Signal Description</b>			
<b>USB Host</b>			
185	USBH-	USB Host Data -	I/O
187	USBH+	USB Host Data +	I/O
<b>USB OTG</b>			
193	USBD-	USB2.0 OTG Data-	I/O
195	USBD+	USB2.0 OTG Data+	I/O
197	GND	GND POWER	P
199	USBVBUS	USB Mini-Receptacle Vbus	P
201	USBID	USB Mini-Receptacle Identifier	I
203	USBDRWBUS	Drive Vbus for Off-Chip Charge Pump	O

### 10.2.3. Design Guidelines

Figure 10-1 shows USB connections for MXM module USB signals.

**Figure 10.1 USB Connection**



#### 10.2.3.1. 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.

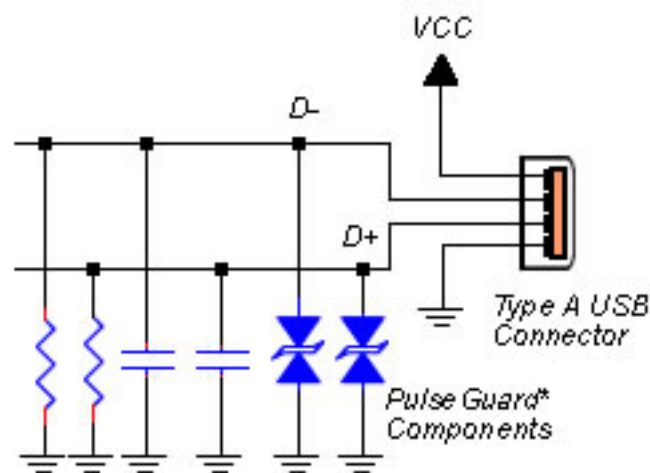
Reducing these values is not recommended. These capacitors should be low ESR, low inductance.

#### 10.2.3.2. 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 equip 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 10.2 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 10.2 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).

### 10.2.3. Layout Guidelines

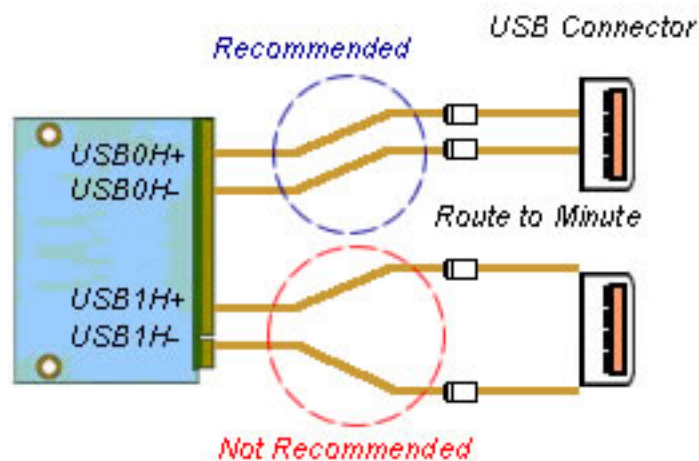


### **10.2.3.1. Differential Pairs**

The USB data pairs (ex. USB0H+ and USB0H-) should be routed on the carrier board as differential pairs, with a differential impedance of 90  $\Omega$ . 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.

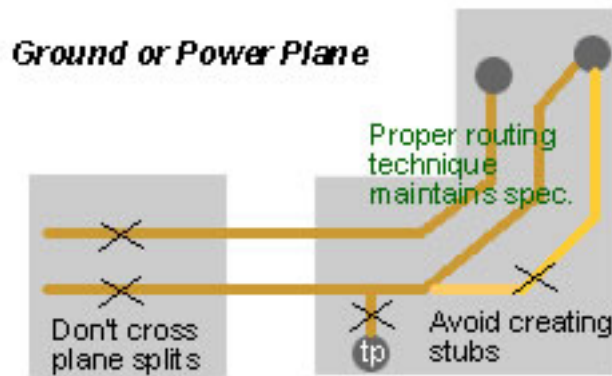
**Figure 10.3 USB Layout Guidelines**



### **10.2.3.2. Cross a plane split**

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

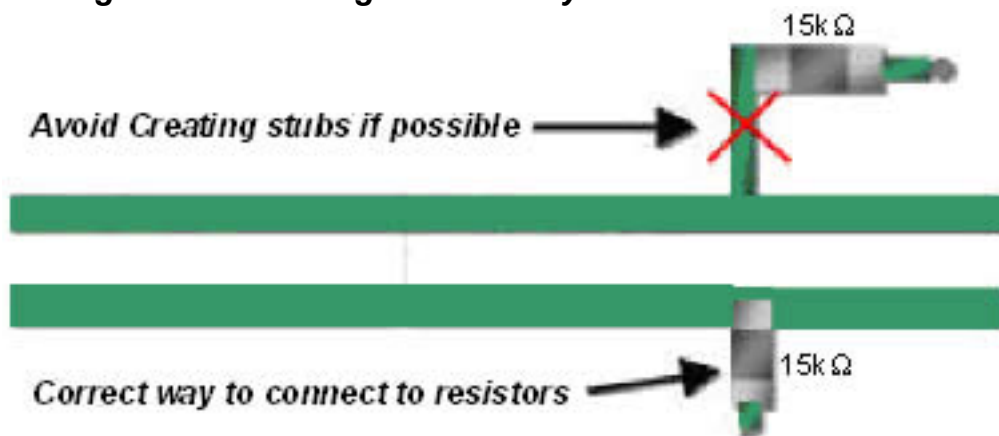
**Figure 10.4 Violations of Proper Routing Techniques**



#### **10.2.3.3. Stubs**

A very common routing mistake is shown in Figure 10.5. Here the designer could have avoided creating unnecessary stubs by proper placement of the pull down resistors over the path of the data traces. Once again, if a stub is unavoidable in the design, no stub should be greater than 200 mils. Here is another example where a stub is created that could have been avoided. Stubs typically cause degradation of signal quality and can also affect EMI.

**Figure 10.5 Creating unnecessary stubs**



### 10.3 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.

#### 10.3.1. Signal Description

Table 10.10 shows MXM module AC-Link signals, including pin number, signals, I/O and descriptions.

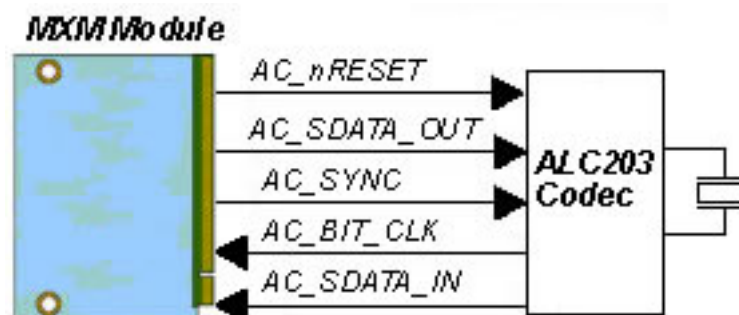
**Table 10.10 Audio Signal Description**

<b>Table 10.10 Audio Signal Description</b>			
<b>AC97</b>			
16	AC_SYNC	48kHz fixed rate sample sync	O
18	AC_BIT_CLK	12.288MHz serial data clock	I/O
20	AC_nRESET	AC'97 Master H/W Reset	O
22	AC_SDATA_IN	AC'97 input stream	I
24	AC_SDATA_OUT	AC'97 output stream	O

#### 10.3.2. Design Guidelines

Figure 10.6 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 10.6 AC-Link Connections**



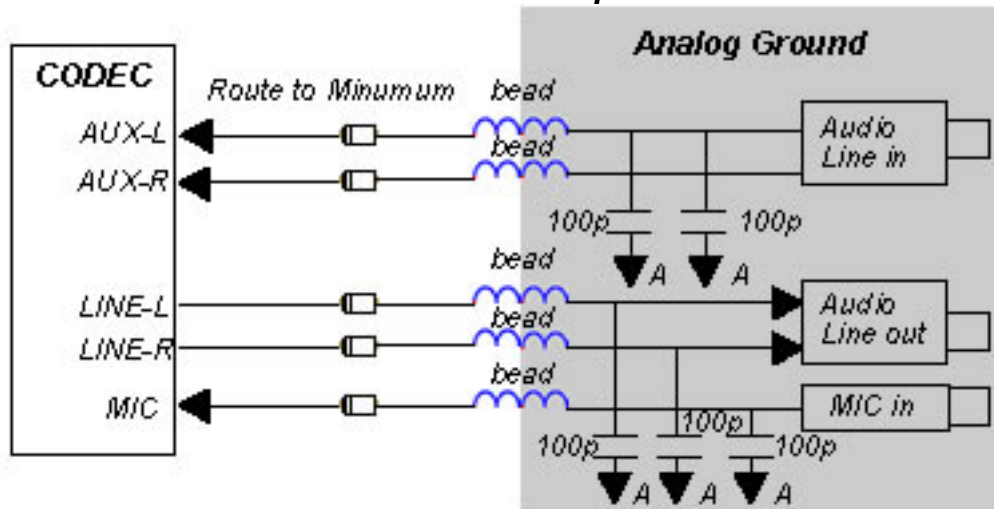
#### 10.3.2.1. Codec Reference and Anti-Aliasing Recommendations

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.

#### 10.3.2.2. 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 10.7 provides an AC'97 example.

**Figure 10.7 AC-Link Audio Ground Technique**



#### 10.3.2.3. 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.

### **10.3.3. 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.

#### **10.3.3.1. Ground and Supply Planes**

Figure 10.8 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.

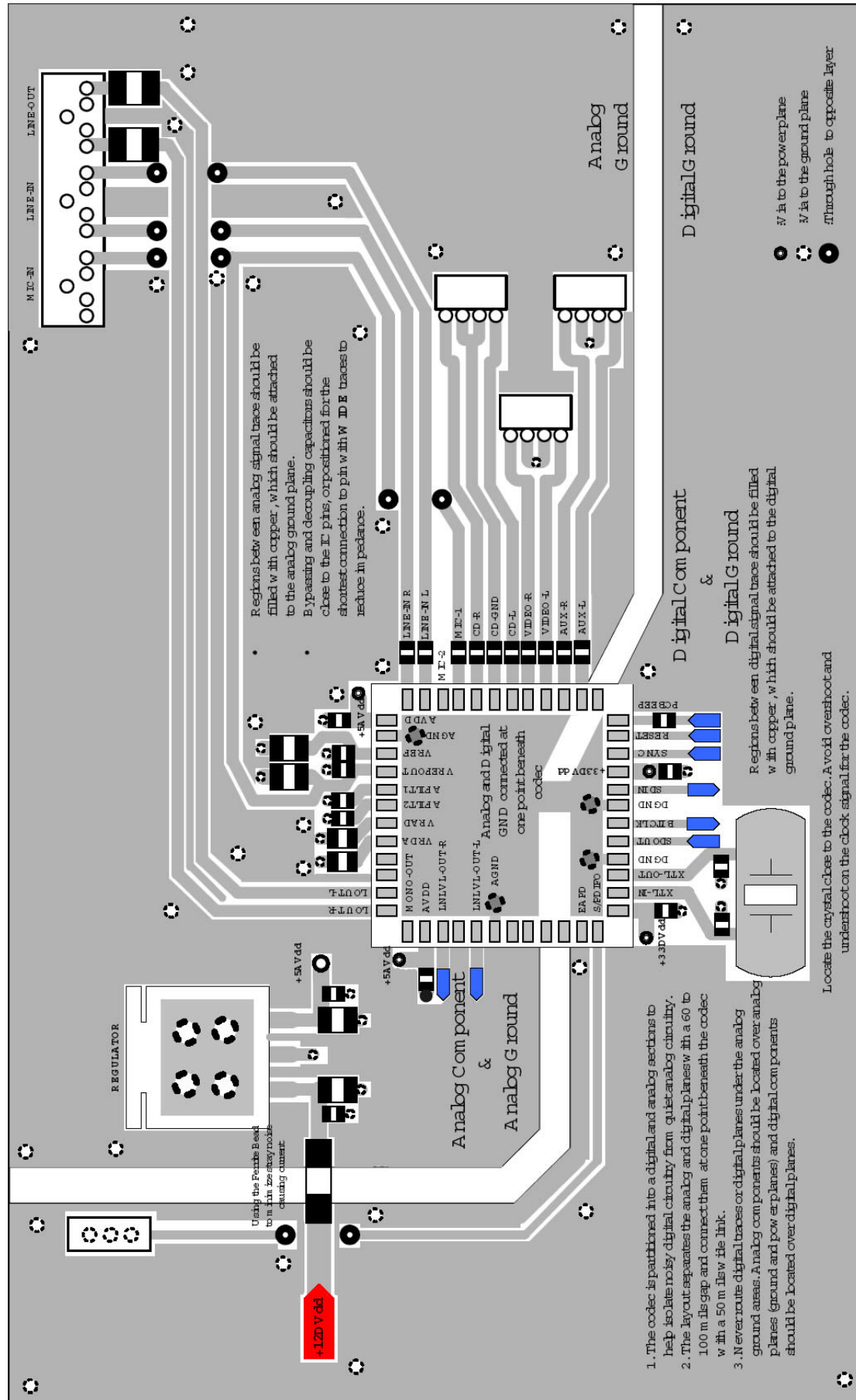
The digital and analog ground planes are tied together by a wide link, about 50mils, at one point, and only one point, beneath the CODEC itself. This acts as the "drawbridge" that goes across the moat. Do not allow any digital or analog signal traces to pass through the drawbridge, or digital noise may 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 10.8 AC-Link Audio Layout Guidelines**



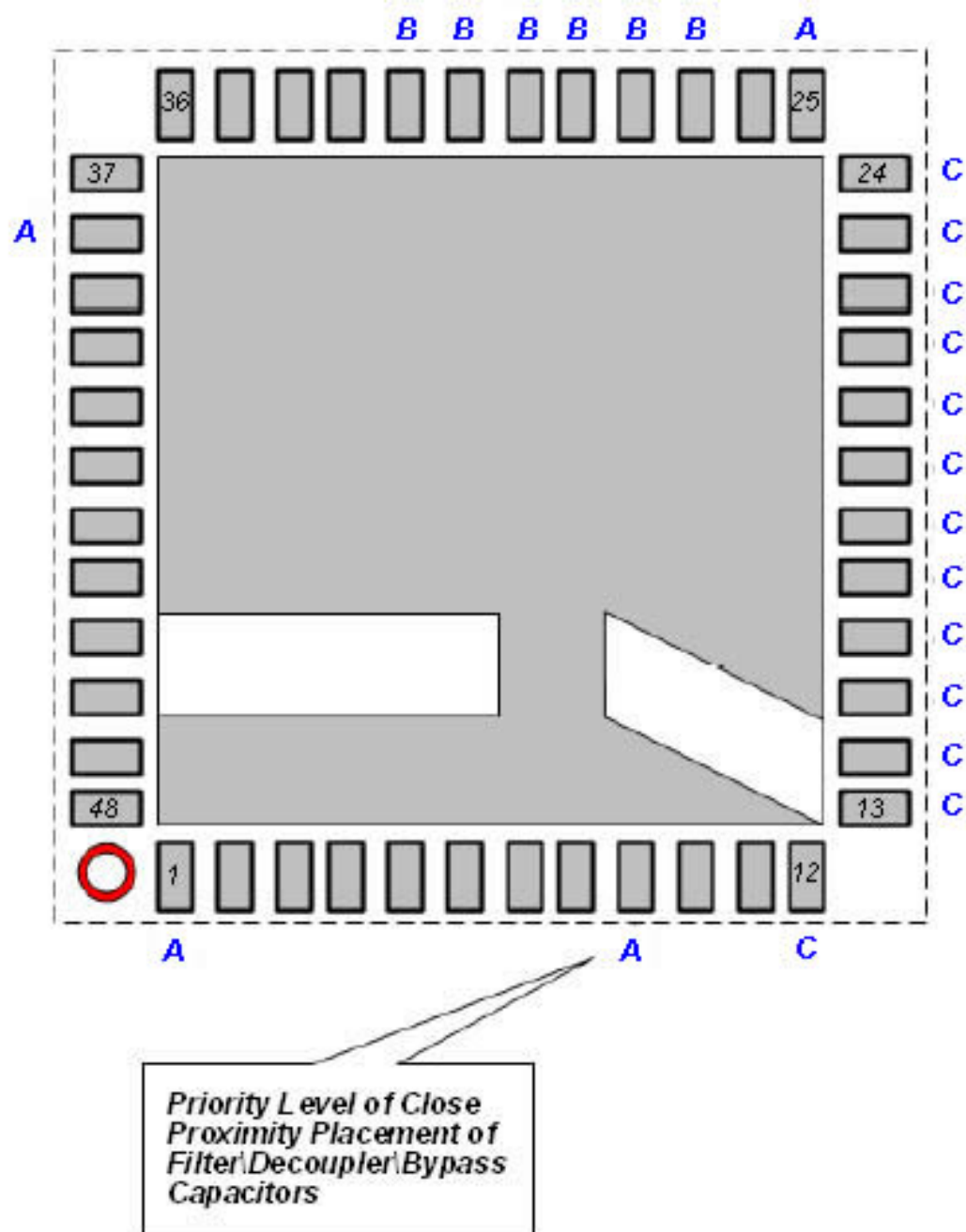
#### **10.3.3.2. Decoupling and Bypassing Capacitors**

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 10.9, 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 10 $\mu$ F surface mount devices are good if they are used in conjunction with 0.1 $\mu$ F 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 10.11 also points out the distribution of CODEC capacitor locations and placement priorities.

**Figure 10.9 CODEC Recommended Capacitor Placement**





**Table 10.11 Series CODEC Capacitor Placement Priorities**

<b>Table 10.11 Realtek Series CODEC Capacitor Placement Priorities</b>		
<b>Signal Description</b>	<b>Package Pins</b>	<b>Priority of Close Proximity to CODEC Pin Placement of Filter and Decoupling Capacitors</b>
Digital Supply Voltage , +5DVdd	1, 9	A
Analog Supply Voltage, +5AVdd	25, 38	A
Voltage Reference Filter(VREF)	27	B
Voltage Reference (VREF_OUT)	28	B
CODEC Filters	29, 30, 31, 32	B
Analog Signal Inputs(Decouple)	12~24	C

## **10.4 TTL/LVDS LCD**

MXM-6410 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.

### **10.4.1. Signal Description**

Table 10.12 shows MXM module TTL level LCD signals, including pin number, signals, I/O and descriptions.

**Table 10.12 CPU LCD Signal Description**

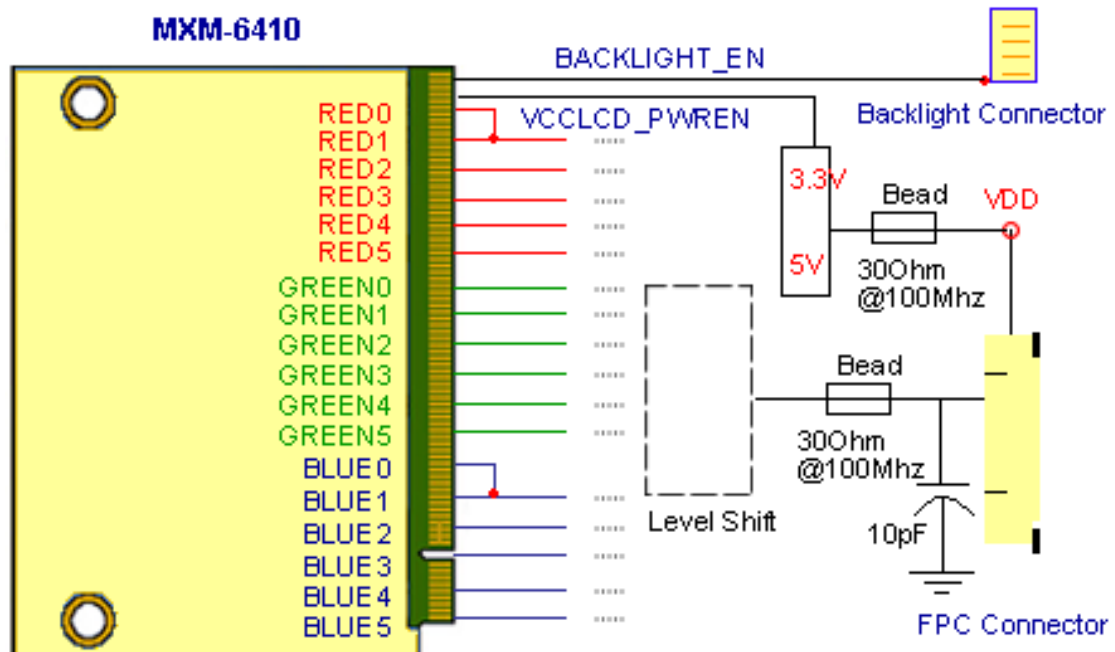
<b>Table 10.12 CPU LCD Signal Description</b>			
<b>CPU LCD</b>			
46	VD18	LCD data bus RED2	0
48	VD19	LCD data bus RED3	0
50	VD20	LCD data bus RED4	0
52	VD21	LCD data bus RED5	0
54	VD22	LCD data bus RED6	0
56	VD23	LCD data bus RED7 (MSB)	0
58	VD10	LCD data bus GREEN2	0
60	VD11	LCD data bus GREEN3	0
62	VD12	LCD data bus GREEN4	0
64	VD13	LCD data bus GREEN5	0
66	VD14	LCD data bus GREEN6	0
68	VD15	LCD data bus GREEN7 (MSB)	0
70	VD2	LCD data bus BLUE2	0
72	VD3	LCD data bus BLUE3	0
74	VD4	LCD data bus BLUE4	0
76	VD5	LCD data bus BLUE5	0
78	VD6	LCD data bus BLUE6	0
80	VD7	LCD data bus BLUE7 (MSB)	0
82	VCLK	LCD clock signal	0
84	HSYNC	Horizontal synchronous signal	0
86	VSYNC	Vertical synchronous signal	0
88	VDEN	Data enable signal	0
89	VD16	LCD data bus RED0 (LSB)	0

91	VD17	LCD data bus RED1	O
93	VD8	LCD data bus GREEN0 (LSB)	O
95	VD9	LCD data bus GREEN1	O
97	VD0	LCD data bus BLUE0 (LSB)	O
99	VD1	LCD data bus BLUE1	O
90	GND	Ground	P

#### **10.4.2. Design Guidelines**

Figure 10.10 shows the TTL LCD connection.

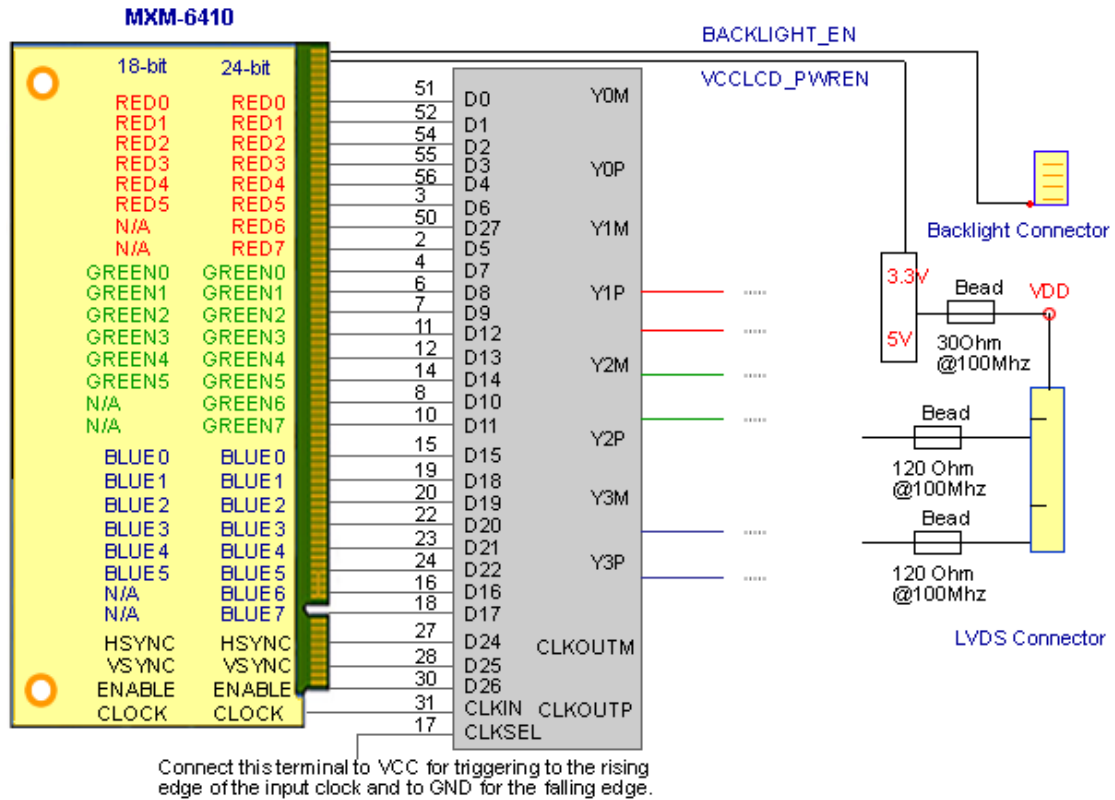
**Figure 10.10 TTL LCD Connection**



All MXM-6410 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 SN75LVDS83 chip. Figure 10.11 shows the LVDS LCD connection.

**Figure 10.11 LVDS LCD Connection**



### 10.4.3. Layout Guidelines

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

Figure 10.12 LVDS LCD Layout Guidelines

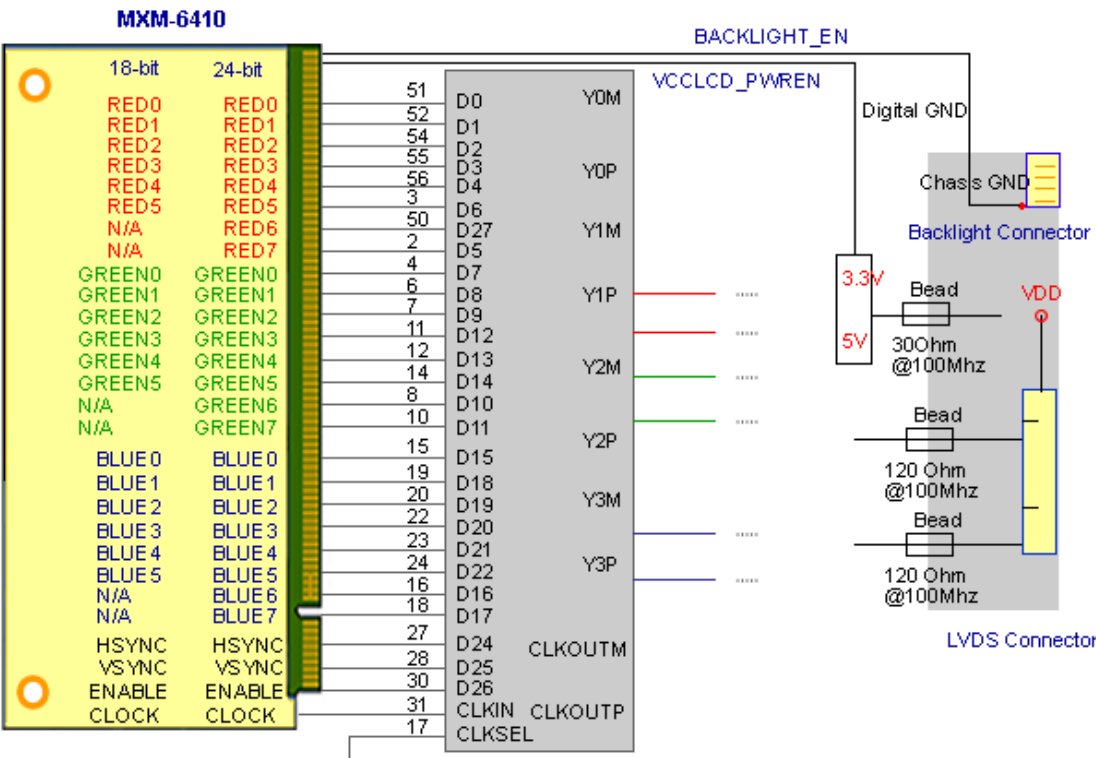
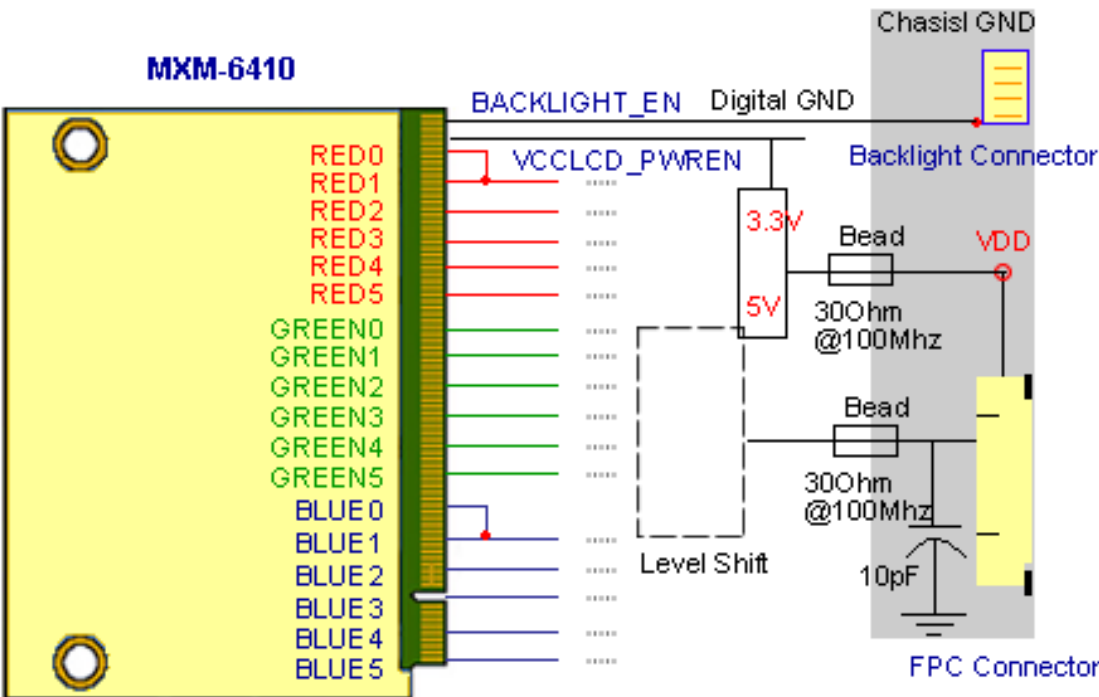


Figure 10.13 TTL LCD Layout Guidelines



## 10.5 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.

### 10.5.1. Signal Descriptions

Table 10.13 shows MXM Module Ethernet signals, including pin number, signals, I/O, power plane and descriptions.

**Table 10.13 Ethernet Signal Description**

<b>Table 10.13 Ethernet Signal Description</b>			
<b>Ethernet</b>			
226	LANLED1	Ethernet Speed LED	O
228	LANLED2	Ethernet Link LED	O
230	AVDD18	1.8V For Transformer	P
232	TX-	Ethernet Transmits data-	O
234	TX+	Ethernet Transmits data+	O
236	AGND	Ethernet Ground	P
238	RX-	Ethernet Receives data-	I
240	RX+	Ethernet Receives data+	I
242	AVDD18	1.8V For Transformer	P

## 10.6.2. Design Guidelines

### 10.6.2.1. 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  $\Omega$ . PCB layout software allows determination of the correct trace width and spacing to achieve this impedance after the PCB stack-up configuration is known.

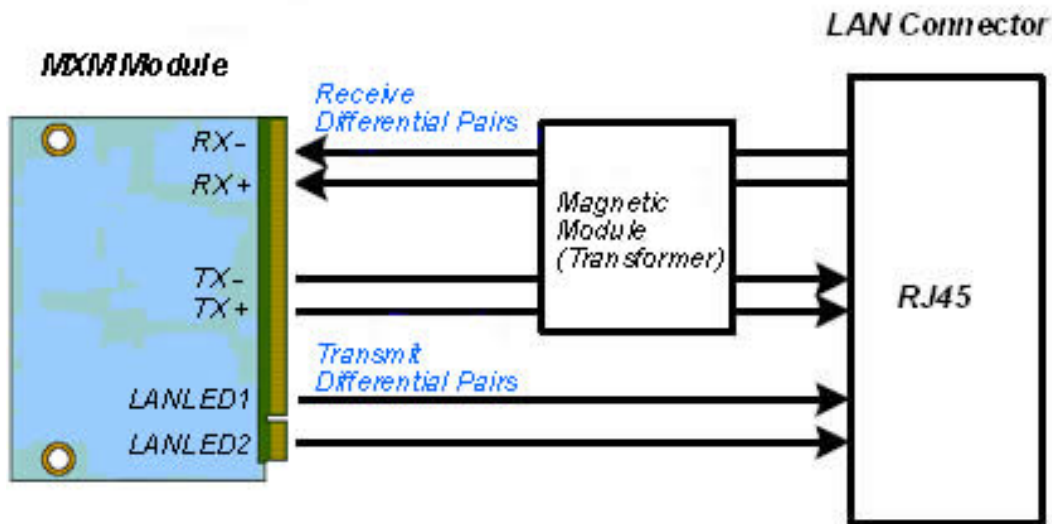
With 10/100M, the TX+, TX- signal pair should be well separated from the RX+, RX- signal pair. Both pairs should be well separated from any other signals on the PCB.

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 10.14 shows the 10/100M Ethernet Connections.



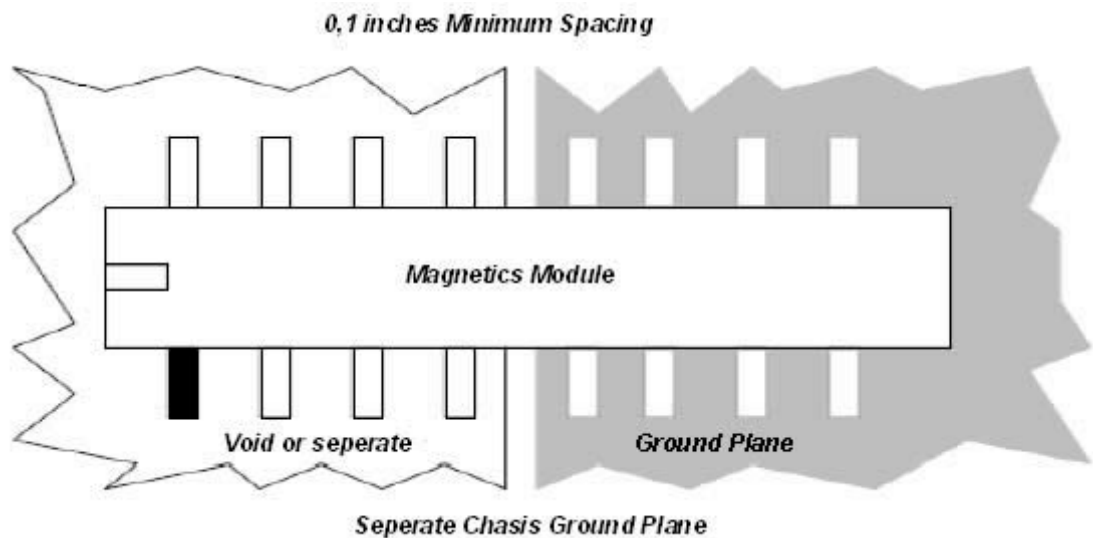
**Figure 10.14 10/100Mbps Ethernet Connection**



#### **10.6.2.2. 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 10.15 shows an example.

**Figure 10.15 Ground Plane Separations**



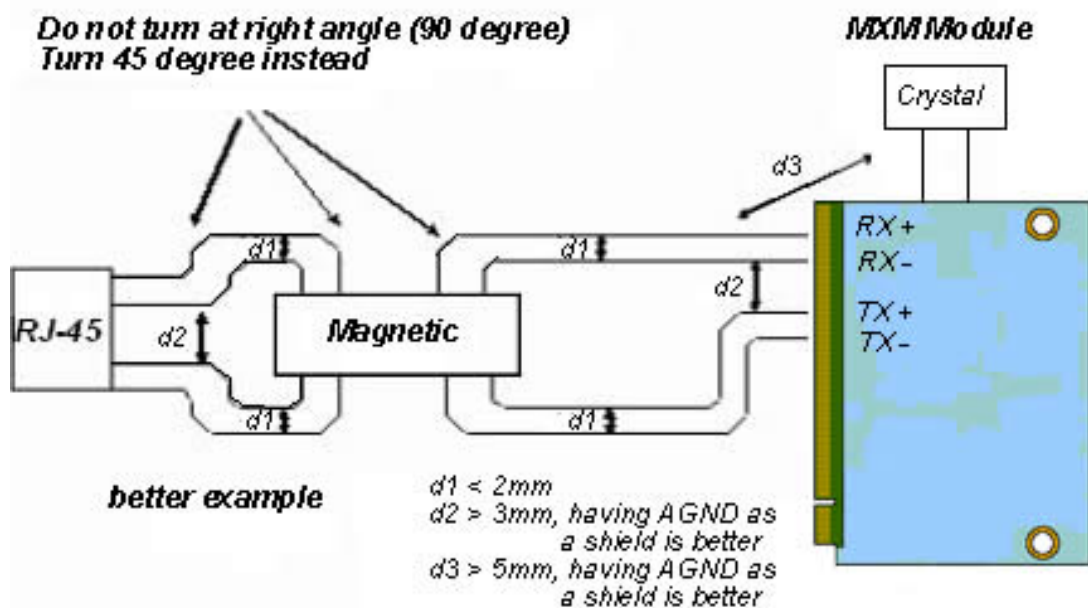
### **10.6.3. Layout Guidelines**

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.

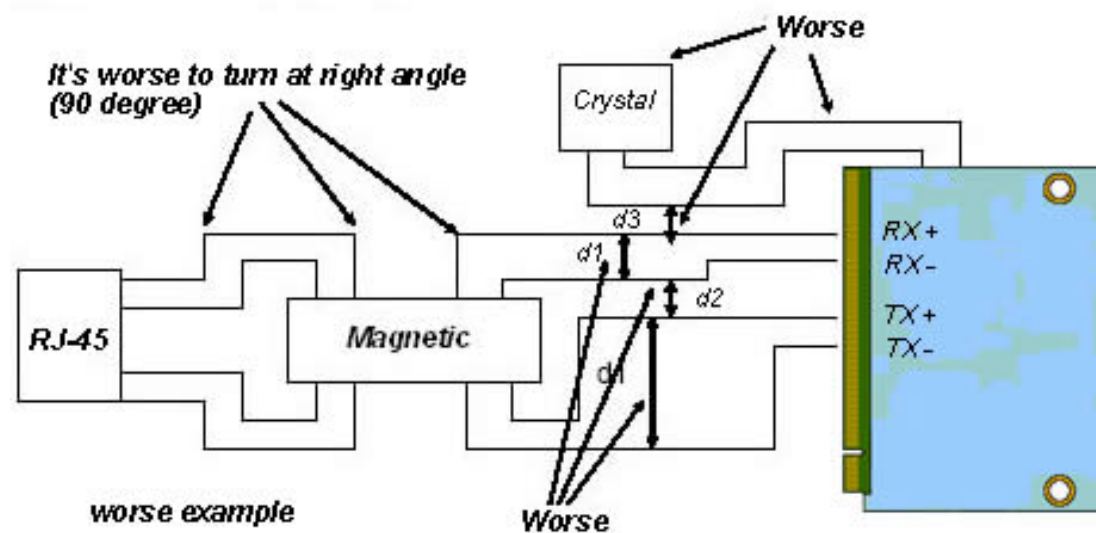
#### **10.6.3.1. Placement, Signal and Trace Routing**

- 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 $\pm$  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 $\pm$  pair.
- It is recommended that RX $\pm$  receive and TX $\pm$  transmit traces turn at 45° angle. Do not turn at right angle.
- Avoid using vias in routing the traces of RX $\pm$  pair and TX $\pm$  pair.
- Do not place the MXM module RX $\pm$  receive pair across the TX $\pm$  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 10.16 and Figure 10.17) does not route any digital signal between the MXM module RX $\pm$  and TX $\pm$  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 10.16 and Figure 10.17) should be tied as closely as possible to the chassis ground through a resistor divider network 75 $\Omega$  resistors (no more than 2mm to the magnetic) and a 0.01 $\mu$ F/2KV bypass capacitor.

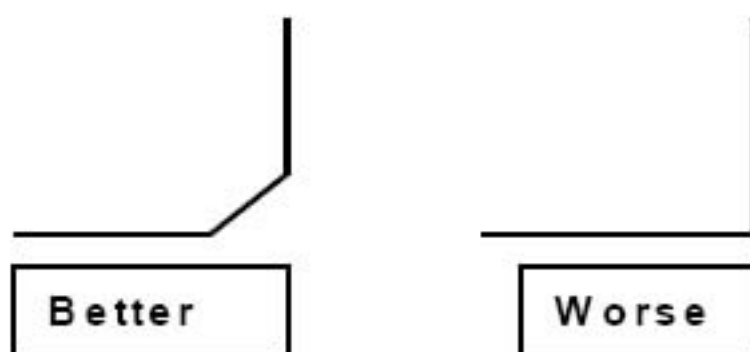
**Figure 10.16 Better examples for signal and trace routing**



**Figure 10.17 Worse examples for signal and trace routing**



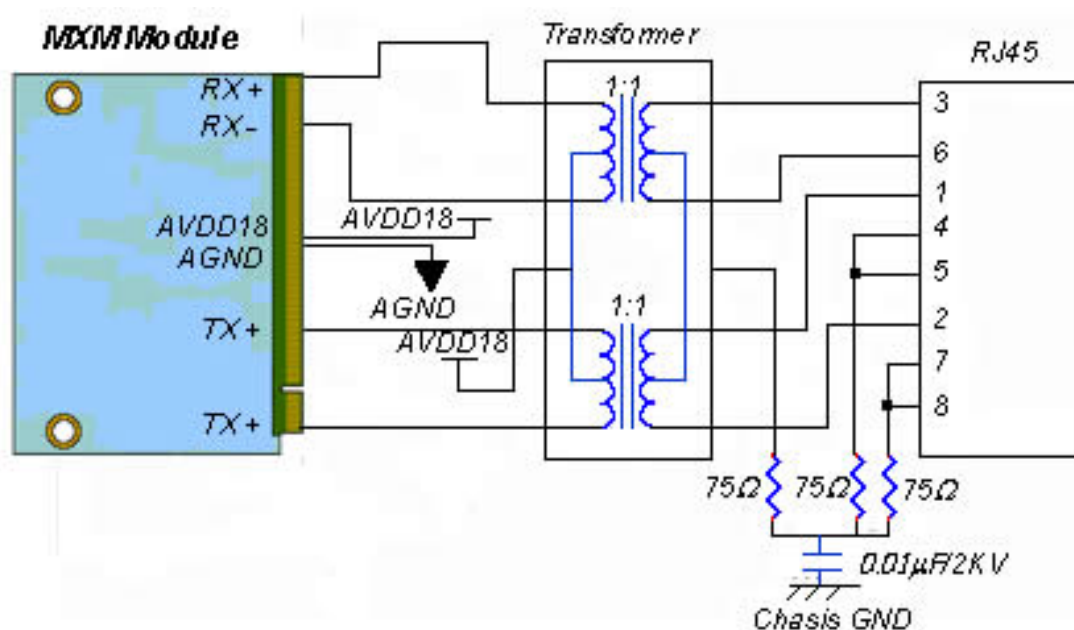
**Figure 10.18 Example for better and worse trace**



### 10.6.3.2. MXM Module 10Base-T/100Base-TX Application

Figure 10.19 illustrates 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.

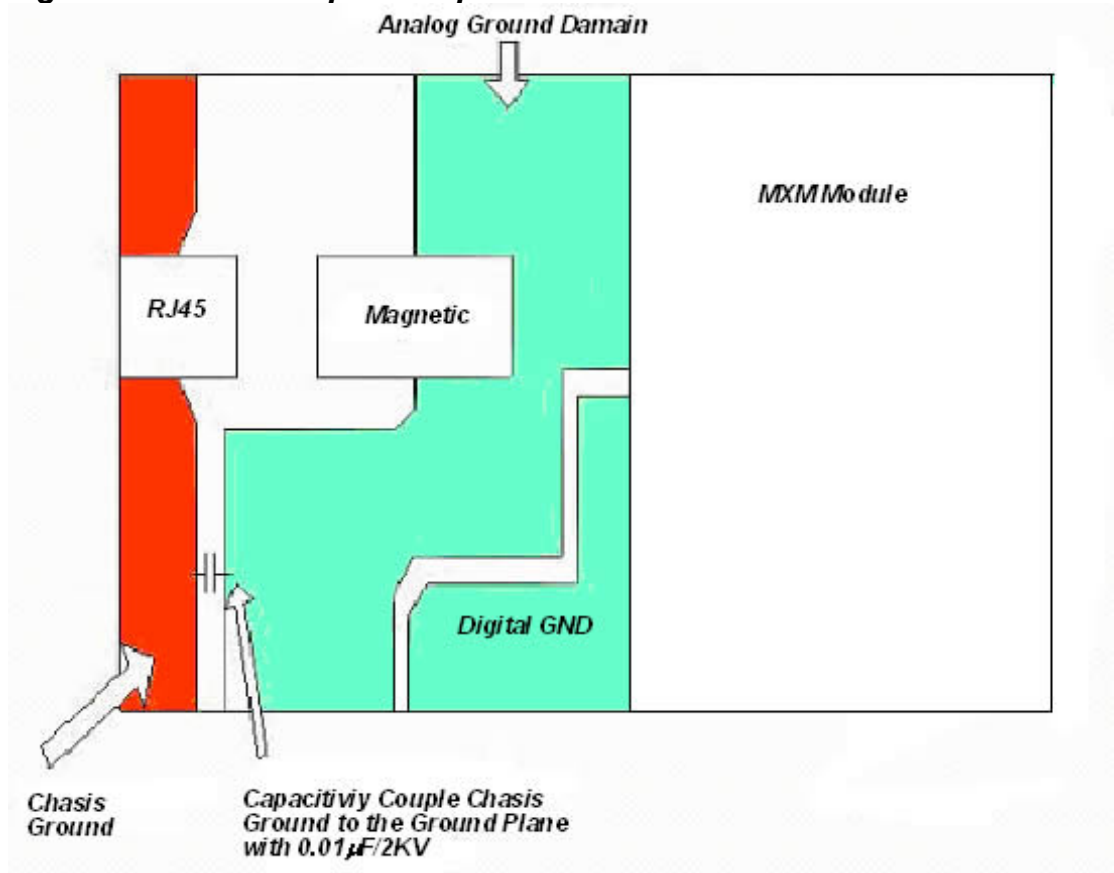
**Figure 10.19 Application with auto\_MDIX transformer (turn ratio 1CT:1CT)**



### **10.6.3.3. 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 10.20.)

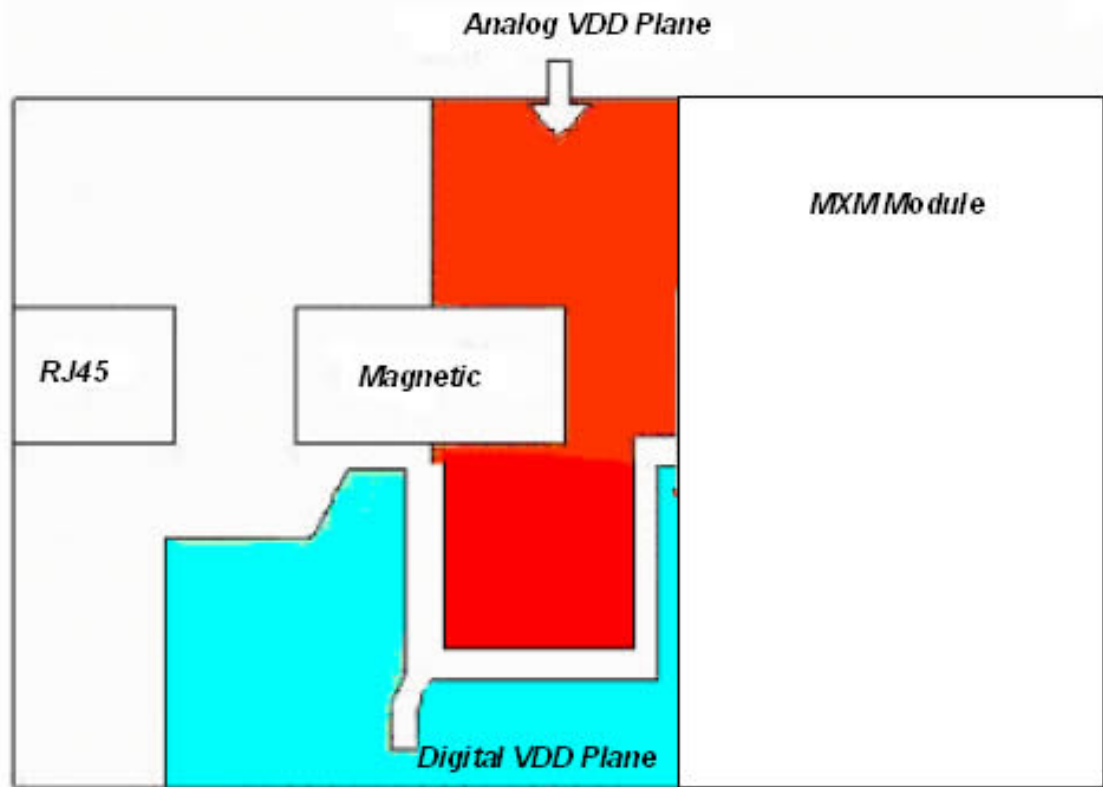
**Figure 10.20 Ground plane separations for MXM module**



#### 10.6.3.4. Power Plane Partitioning

- The power planes should be approximately illustrated in Figure 10.21. No bead is needed to connect two power planes.
- It should separate analog power planes from noisy digital (logic) power planes.

**Figure 10.21 Power planes partitioning for MXM module**



**10.6.3.5. Magnetic Selection Guide**

Refer to the following tables 10.14, 10.15 and 10.16 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 10.14 10/100Mbps RJ45 Jack (Magnetic included)**

<b>Manufacturer</b>	<b>Part Number</b>
Foxconn	JFM24011-0101-4F
YCL	PTC1111-09L1FG

**Table 10.15 10/100Mbps Magnetic Sources**

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

**Table 10.16 Magnetic Specification Requirements**

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

# Chapter 11

## **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.



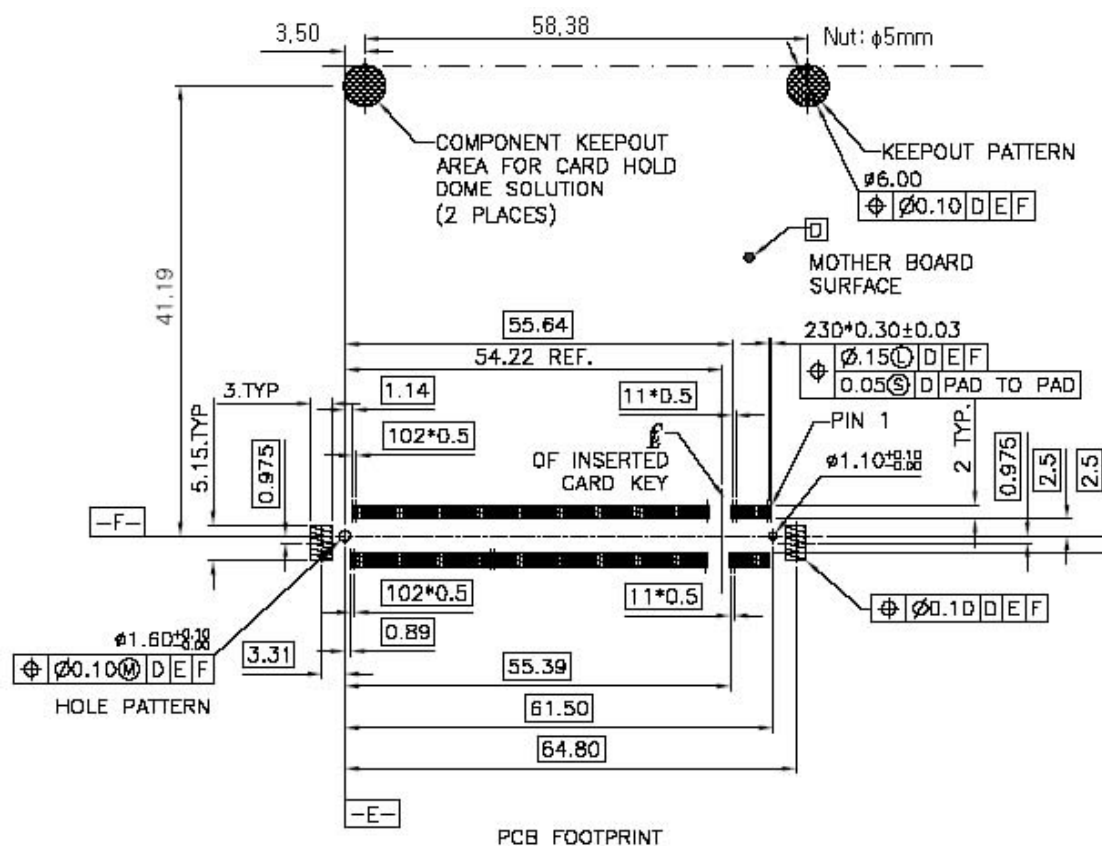
# **Chapter 11 Carrier Board Mechanical Design Guidelines**

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

## 11.1 MXM Motherboard Footprint

Two draw 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 11.1 shows the MXM carrier board footprint.

**Figure 11.1 MXM Motherboard Footprint**



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

Figure 11.2 CN2 Recommended PCB Mounting Pattern

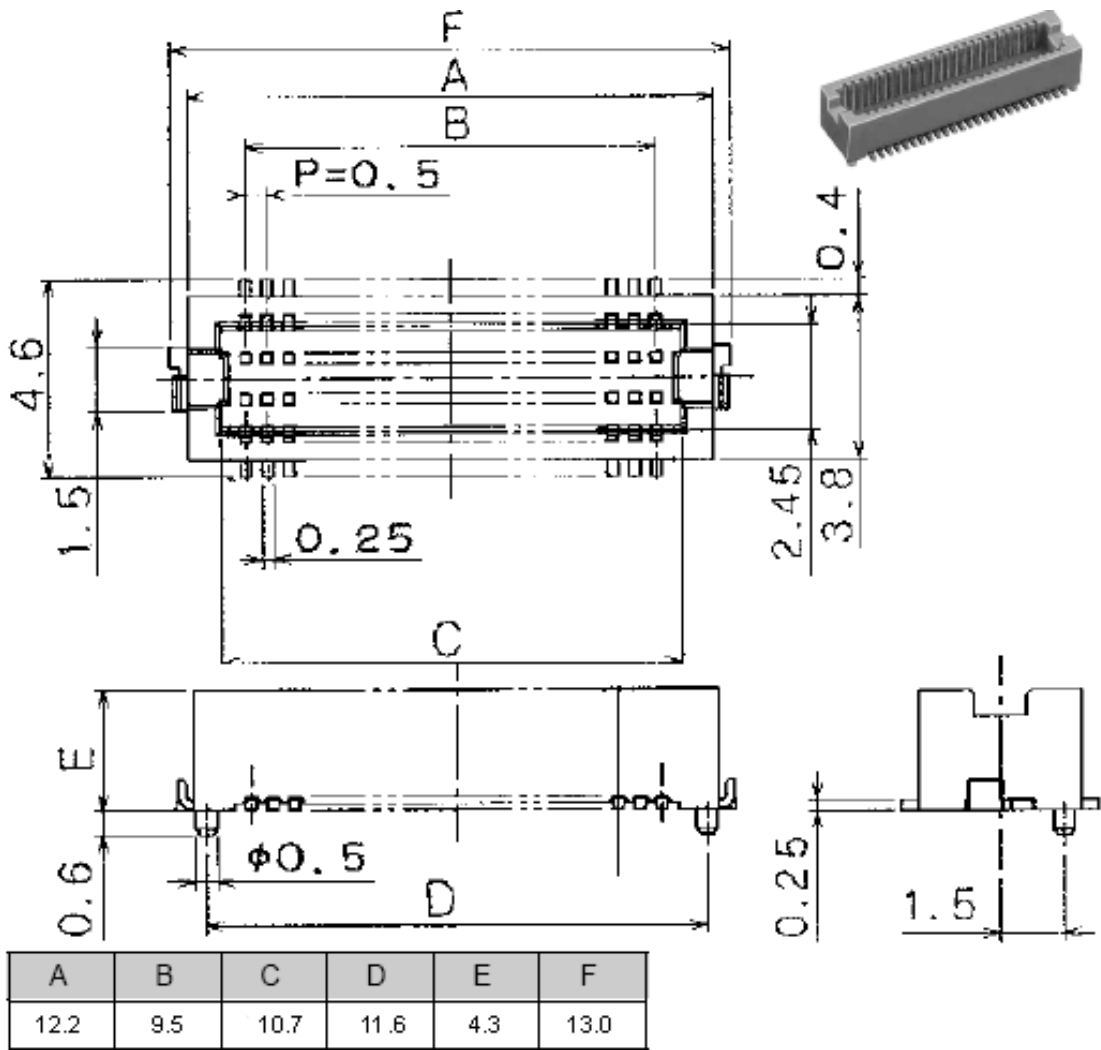
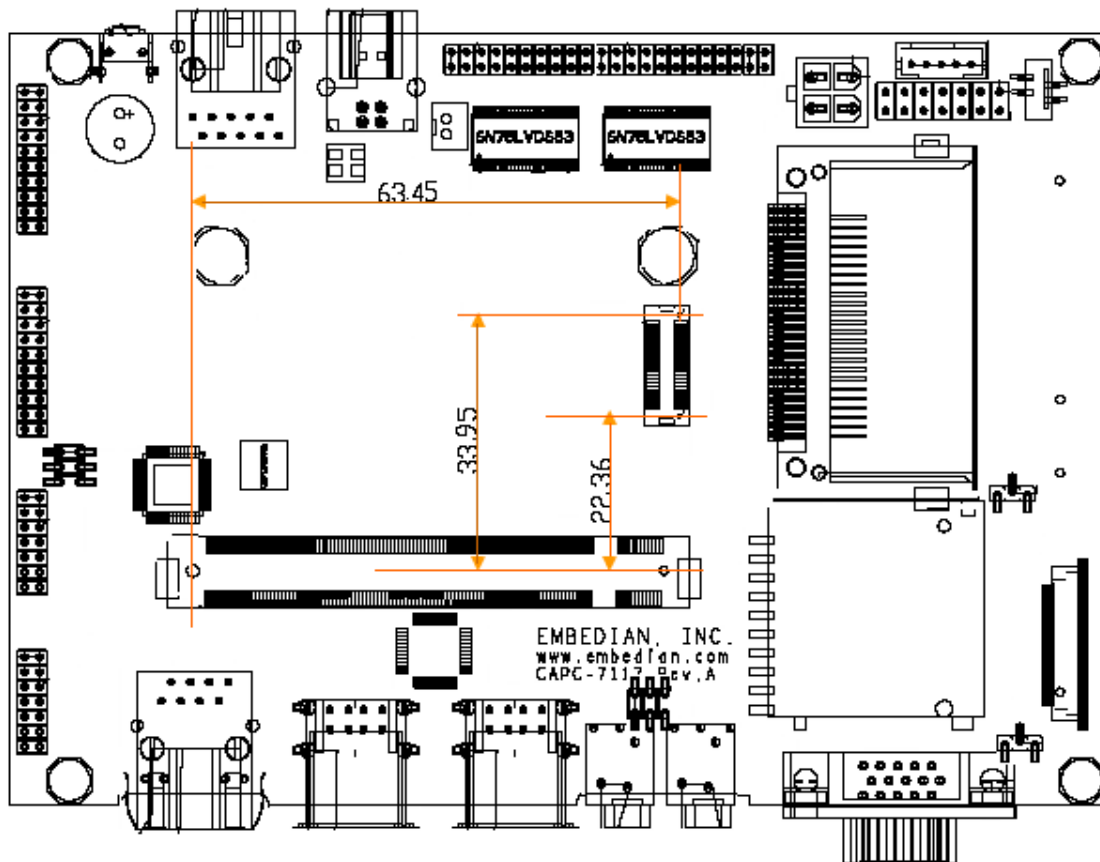


Figure 11.3 shows the MXM motherboard footprint with CN2 connectors

**Figure 11.3 MXM Motherboard Footprint with CN2 Connector**



# Chapter 12

## ***Pin Definition Differences between Embedian MXM modules***

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

## **Chapter 12 *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 12.1 Pin Definition Differences between Embedian MXM modules (Top Side)**

<b>Table 12.1 Pin Definition Differences between Embedian MXM modules (Top Side)</b>			
<b>Pin</b>	<b>MXM-7110/MXM-6410</b>	<b>MXM-8310</b>	<b>MXM-6410</b>
9~15	ADC Input	N.C.	ADC Input
41	DMAACK0	N.C	TBD
99	Address 26	N.C.	TBD
101~113	N.C.	PC Card Bus Related	TBD
179	Boot ROM Select	N.C.	TBD
183	N.C.	One Wire Bus	TBD
199~241	N.C.	USB 2.0 Client UTMI Interface	TBD

**Table 12.2 Pin Definition Differences between Embedian MXM modules (Bottom Side)**

<b>Table 12.2 Pin Definition Differences between Embedian MXM modules (Bottom Side)</b>			
<b>Pin</b>	<b>MXM-7110/MXM-6410</b>	<b>MXM-8310</b>	<b>MXM-6410</b>
104~118	SPI Interface	SSP Interface (*)	TBD
194~224	UART 0,1,2	UART 1,2,3 (**)	TBD

(\*)SSP interface in MXM-8310 can be configured as SPI interface by software.

(\*\*) The UART numbering just followed the respectively S3C2440 and PXA320 CPU manual.

# Appendix



## Linux Firmware Update

This Chapter details how to update Linux firmware in NAND flash.

Section include :

- Firmware Architecture
- Update Firmware from uboot
- Update Firmware from NOR flash



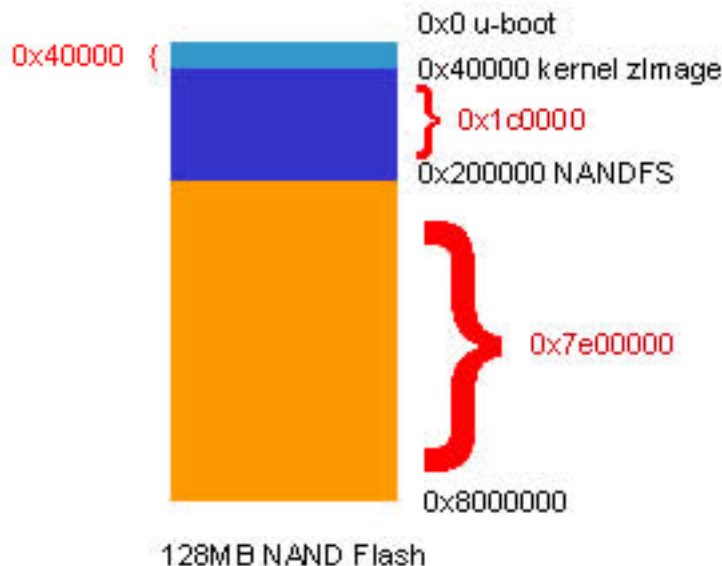
# Appendix I MXM-6410 Linux Firmware Update

This Chapter details firmware upgrade for MXM-6410 when using Linux as the operating systems. The firmware in NAND flash includes uboot, kernel zImage and nandfs image.

This guide mainly uses MXM-6410 on the evaluation kit as an example. Users can follow the same way at your own baseboard.

## A.1. Firmware Architecture

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



**Figure A.1 Firmware Architecture of Linux in NAND Flash**

The *uboot.bin* partition starts from NAND address *0x0000000* (*0x0*). The Linux kernel *zImage* partition starts from NAND address *0x40000*. The NAND filesystem (*nandfs*) is a small ext3 file system for rescue purposed or system that would only need simply application and load the minimum set drivers and this partition starts from the NAND address *0x200000*.

Users need a SD/SDHC card with root file system installed to boot up the complete Ubuntu 9.04 file system. As for how to prepare for an SD/SDHC root file system, this is described at Chapter 6.

Users can update the firmware under uboot or use the onboard NOR flash. The onboard NOR flash is designed for the purpose that when your uboot is erased and cannot boot up anymore. Embedian factory default is firmware pre-installed. Unless necessary, Embedian does not recommend you update firmware (especially uboot) since the system might not boot up anymore if you

did the wrong operation. (If you develop your own uboot and kernel, you will need to do that.) Following tells you howto update firmware from uboot command prompt. The firmware for MXM-6410 includes uboot.bin, zImage, and nand6410.img. Now, you are ready to update firmware from uboot. 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 and NAND root file system. Below we will tell you how to do this under Windows and Linux PC environment. First, you need to set up a tftp server.

### **A.2.1. Windows Environment**

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

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

Users need to install tftp server on Windows. You can download the freeware and install to your Windows PC in the **tftpboot** directory. Copy the **uboot.bin**, **zImage** and **nand6410.img** into this directory. Close your anti-virus software like PC-cillin. (Or close port 69)

First, power on the MXM-6410 evaluation kit with console debug port (UART0, CN20) 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.

```
In:      serial
Out:     serial
Err:     serial
found DM9000 ID:90000a46
DM9000 work in 16 bus width
MAC: 10:d:32:10:1:12:
Hit any key to stop autoboot:  0
SMDK6410 #
```

You can set and add the *tftp* ip address of the MXM-6410 evaluation kit and server by using "**setenv**", command as below.

```
SMDK6410 # setenv ipaddr <device ip>
SMDK6410 # setenv serverip <server ip>
```

After setting up the ip address, use *saveenv* command to save the configuration. Following figure shows the example for setting up the parameters.

Following figure shows the example for setting up the parameters.

**EXAMPLE:**

```
SMDK6410 # setenv ipaddr 192.168.1.202
SMDK6410 # setenv serverip 192.168.1.10
SMDK6410 # saveenv
Saving Environment to NAND...
Erasing Nand...Writing to Nand... done
SMDK6410 #
```

**Note:**

Make sure that the **server** ip for Windows PC and **client ip** for MXM-6410 evaluation kit are in the same network domain.

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

**A.2.1.2. Transfer and Write Image by TFTP and “nand write” Command**

After setting up the tftp server and IP address of MXM-6410 evaluation kit, users can start transfer and write images using uboot *tftp* and *nand write* command. It is necessary to download to DRAM and erase the contents in partition first before writing to NAND.

**uboot.bin**

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

```
SMDK6410 # tftp 50000000 uboot.bin
Found DM9000 ID:90000a46 !
DM9000 work in 16 bus width
[eth_init]MAC:10: d:32:10: 1:12
TFTP from server 192.168.1.10; our IP address is 192.168.1.202
Filename 'uboot.bin' .
Load address: 0x50000000
Loading: T T #####
done
Bytes transferred = 196608 (30000 hex)
SMDK6410 #
```

Now, *uboot.bin* file has been uploaded to DRAM temporary address from your PC. Temporary address is base address of DRAM, Default is set to *0x50000000*. From the log, the file size of *uboot.bin* is 196608 bytes (30000 bytes in HEX).

Write the **uboot.bin** image from DRAM to the NAND by using following *nand write* command. Before writing *uboot.bin* from DRAM to NAND flash, you need to erase the contents of that partition first by *nand erase* command.

```
SMDK6410 # nand erase 0x0 40000

NAND erase: device 0 offset 0x0, size 0x40000
Erasing at 0x20000 -- 100% complete.
OK
SMDK6410 #
```

This is to erase the NAND flash content in the first partition.

```
SMDK6410 # nand write 0x50000000 0x0 30000

NAND write: device 0 offset 0x0, size 0x30000
196608 bytes written: OK
SMDK6410 #
```

Temporary address is base address of DRAM, i.e. 0x50000000. Start NAND address is 0x0. Image size of uboot is 0x30000 (HEX) as we have seen previously when tftping.

**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). Wrong operation will cause the system not booting up anymore.

### **zImage**

Next example shows how to transfer and write Linux kernel. The file name is "**zImage**". Again, we tftp zImage from PC to DRAM of MXM-6410 first by the following command.

```
SMDK6410 # tftp 50000000 zImage
Found DM9000 ID:90000a46 !
DM9000 work in 16 bus width
[eth_init]MAC:10: d:32:10: 1:12
TFTP from server 192.168.1.10; our IP address is 192.168.1.202
Filename 'zImage' .
Load address: 0x50000000
Loading: T #####
#####
```

```
#####  
#####  
#####  
#####  
  
done  
Bytes transferred = 1764420 (1aec44 hex)  
SMDK6410 #
```

Now, zImage file has been uploaded to DRAM temporary address of MXM-6410 from your PC. Temporary address is base address of DRAM, Default is set to *0x50000000*. From the log, we also learned that the file size is *1764420* bytes (*1aec44* bytes in HEX).

Write the zImage image from DRAM to the NAND by using *nand write* command. Again, we need to erase the contents in this partition first.

```
SMDK6410 # nand erase 0x40000 0x200000  
  
NAND erase: device 0 offset 0x40000, size 0x200000  
Erasing at 0x220000 -- 100% complete.  
OK  
SMDK6410 #
```

This is to erase the NAND flash content in the second partition.

```
SMDK6410 # nand write 50000000 0x40000 1af000  
  
NAND write: device 0 offset 0x40000, size 0x1af000  
1765376 bytes written: OK  
SMDK6410 #
```

Temporary address is base address of DRAM, i.e. *0x50000000*. Start NAND address is *0x40000*. Image size of zImage is *1b1800* (HEX).

**Note:**

This image size of *zImage* when *fttping* is *1aec44* (Hex), but the image size we write to NAND flash is *1af000* (Hex). This is because the minimum size to write to this NAND flash is 1 page (2K bytes in decimal, 800 bytes in HEX). It is therefore, the image size that writing to NAND flash has to be a multiple of 800.

### ***nand6410.img***

After writing uboot.bin and kernel zImage images to NAND flash, the last step is to write NAND root file system ***nand6410.img*** image. There are some differences from the above uboot.bin and zImage.

The mobile DDR size is 128MB only. If your NANDFS size is big (like Embedian default nand6410.img is 126MB), you need to use *split* command in Linux to split the NANDFS into two smaller files, or the uboot in DRAM will be overwritten because the execution point of uboot is somewhere in DDR. Once you split two smaller files, it will be a good practice for you to flash nandfs into NAND flash. Just remember that the starting address of nandfs partition is *0x200000*. Below is short description. This example split the nan6410.img as a 64MB image and a 62MB image.

```
SMDK6410 # nand erase 200000 7e000000
SMDK6410 # tftp 50000000 nand6410.img.1
SMDK6410 # nand write 50000000 200000 4000000
SMDK6410 # tftp 50000000 nand6410.img.2
SMDK6410 # nand write 50000000 4200000 3e000000
SMDK6410 #
```

After done, reset MXM-6410 evaluation kit and the firmware will be updated.

### **A.2.2 Linux Environment**

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

#### **A.2.2.1. Minicom**

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

Desktop Linux has Minicom program for serial communication. It is used for command prompt of 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-6410 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**

```
+-----+
| A - Serial Device      : /dev/ttyS0 |
| B - Lockfile Location  : /var/lock  |
| C - Callin Program     :             |
| D - Callout Program    :             |
| E - Bps/Par/Bits       : 115200 8N1 |
| F - Hardware Flow Control : No      |
| G - Software Flow Control : No      |
|                           |
| Change which setting? █          |
+-----+
| Screen and keyboard      |
| Save setup as dfl        |
| Save setup as..          |
| Exit                     |
| Exit from Minicom        |
+-----+

```

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**

```

+-----+-----[Comm Parameters]-----+-----+
| A - Serial De|                                     |         |
| B - Lockfile Loc|      Current: 115200 8N1         |         |
| C - Callin Pro| Speed          Parity      Data |         |
| D - Callout Pro| A: <next>        L: None     S: 5 |         |
| E - Bps/Par/B| B: <prev>         M: Even    T: 6 |         |
| F - Hardware Flo| C: 9600          N: Odd      U: 7 |         |
| G - Software Flo| D: 38400         O: Mark     V: 8 |         |
|               | E: 115200        P: Space   |         |
|               |                                     |         |
| Change which |                                     |         |
+-----+-----+ Stopbits                    +-----+
| Screen a| W: 1          Q: 8-N-1                |         |
| Save set| X: 2          R: 7-E-1                |         |
| Save set|                                     |         |
| Exit    |                                     |         |
| Exit fro| Choice, or <Enter> to exit? █         |         |
+-----+-----+

```

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

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

**Figure A.5 Resetting from Minicom**

```

Welcome to minicom 2.4

OPTIONS: I18n
Compiled on Jan 25 2010, 06:49:09.
Port /dev/ttyS0

Press CTRL-A Z for help on sp+-----+
| Leave without reset? |
|   Yes   No         |
+-----+

```

To use minicom,

```
root@dns2:~# minicom < filename>
```



#### **A.2.2.2. TFTP server in Linux PC**

MXM-6410 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 */tftpboot* directory.

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

Start the tftpd through xinetd and you are done.

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

Now you can download compiled images to the MXM-6410 by using **tftp**. Before downloading the images, connect host PC and MXM-6410 evaluation kit by Ethernet cable.

To download binary image files to MXM-6410, run tftp server service on your computer and put images in **/tftpboot** directory.

#### **A.2.2.3. Setting up an IP address**

Setting up an IP address helps in downloading the compiled images to

MXM-6410.

Connect host PC and MXM-6410 evaluation kit by Ethernet cable.

#### **A.2.2.3.1. Setting IP address for host PC**

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

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

#### **A.2.2.3.2. Setting TFTP Server/Client IP address form MXM-6410 evaluation kit**

Below will be exactly the same as that in Windows environment. Copy the **uboot.bin**, **zimage** and **nand6410.img** into this directory.

First, power on the MXM-6410 evaluation kit with console debug port (UART0, CN20) 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.

```
In:      serial
Out:     serial
Err:     serial
found DM9000 ID:90000a46
DM9000 work in 16 bus width
MAC: 10:d:32:10:1:12:
Hit any key to stop autoboot:  0
SMDK6410 #
```

You can set and add the *tftp* ip address of the MXM-6410 evaluation kit and server by using "**setenv**", command as below.

```
SMDK6410 # setenv ipaddr <device ip>
SMDK6410 # setenv serverip <server ip>
```

After setting up the ip address, use *saveenv* command to save the configuration. Following figure shows the example for setting up the parameters.

Following figure shows the example for setting up the parameters.

#### **EXAMPLE:**

## **Embedian, Inc.**

```
SMDK6410 # setenv ipaddr 192.168.1.202
SMDK6410 # setenv serverip 192.168.1.10
SMDK6410 # saveenv
Saving Environment to NAND...
Erasing Nand...Writing to Nand... done
SMDK6410 #
```

### **Note:**

Make sure that the **server** ip for Windows PC and **client** ip for MXM-6410 evaluation kit are in the same network domain.

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

### **A.2.1.2. Transfer and Write Image by TFTP and “nand write” Command**

After setting up the tftp server and IP address of MXM-6410 evaluation kit, users can start transfer and write images using uboot *tftp* and *nand write* command. It is necessary to download to DRAM and erase the contents in partition first before writing to NAND.

### **uboot.bin**

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

```
SMDK6410 # tftp 50000000 uboot.bin
Found DM9000 ID:90000a46 !
DM9000 work in 16 bus width
[eth_init]MAC:10: d:32:10: 1:12
TFTP from server 192.168.1.10; our IP address is 192.168.1.202
Filename 'uboot.bin' .
Load address: 0x50000000
Loading: T T #####
done
Bytes transferred = 196608 (30000 hex)
SMDK6410 #
```

Now, **uboot.bin** file has been uploaded to DRAM temporary address from your PC. Temporary address is base address of DRAM, Default is set to 0x50000000. From the log, the file size of **uboot.bin** is 196608 bytes (30000 bytes in HEX).

Write the **uboot.bin** image from DRAM to the NAND by using following *nand write* command. Before writing **uboot.bin** from DRAM to NAND flash, you

need to erase the contents of that partition first by *nand erase* command.

```
SMDK6410 # nand erase 0x0 40000

NAND erase: device 0 offset 0x0, size 0x40000
Erasing at 0x20000 -- 100% complete.
OK
SMDK6410 #
```

This is to erase the NAND flash content in the first partition.

```
SMDK6410 # nand write 0x50000000 0x0 30000

NAND write: device 0 offset 0x0, size 0x30000
196608 bytes written: OK
SMDK6410 #
```

Temporary address is base address of DRAM, i.e. 0x50000000. Start NAND address is 0x0. Image size of uboot is 0x30000 (HEX) as we have seen previously when tftping.

**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). Wrong operation will cause the system not booting up anymore.

### **zImage**

Next example shows how to transfer and write Linux kernel. The file name is "**zImage**". Again, we tftp zImage from PC to DRAM of MXM-6410 first by the following command.

```
SMDK6410 # tftp 50000000 zImage
Found DM9000 ID:90000a46 !
DM9000 work in 16 bus width
[eth_init]MAC:10: d:32:10: 1:12
TFTP from server 192.168.1.10; our IP address is 192.168.1.202
Filename 'zImage' .
Load address: 0x50000000
Loading: T #####
#####
#####
```

```
#####  
#####  
#####  
done  
Bytes transferred = 1764420 (1aec44 hex)  
SMDK6410 #
```

Now, zImage file has been uploaded to DRAM temporary address of MXM-6410 from your PC. Temporary address is base address of DRAM, Default is set to *0x50000000*. From the log, we also learned that the file size is *1764420*bytes (*1aec44* bytes in HEX).

Write the zImage image from DRAM to the NAND by using *nand write* command. Again, we need to erase the contents in this partition first.

```
SMDK6410 # nand erase 0x40000 0x200000  
  
NAND erase: device 0 offset 0x40000, size 0x200000  
Erasing at 0x220000 -- 100% complete.  
OK  
SMDK6410 #
```

This is to erase the NAND flash content in the second partition.

```
SMDK6410 # nand write 50000000 0x40000 1af000  
  
NAND write: device 0 offset 0x40000, size 0x1af000  
1765376 bytes written: OK  
SMDK6410 #
```

Temporary address is base address of DRAM, i.e. *0x50000000*. Start NAND address is *0x40000*. Image size of zImage is *1af000* (HEX).

**Note:**

This image size of *zImage* when tftping is *1aec44* (Hex), but the image size we write to NAND flash is *1af000* (Hex). This is because the minimum size to write to this NAND flash is 1 page (2K bytes in decimal, 800 bytes in HEX). It is therefore, the image size that writing to NAND flash has to be a multiple of 800.

***nand6410.img***

After writing uboot.bin and kernel zImage images to NAND flash, the last step is to write NAND root file system **nand6410.img** image. There are some differences from the above uboot.bin and zImage.

The mobile DDR size is 128MB only. If your NANDFS size is big (like Embedian default nand6410.img is 126MB), you need to use *split* command in Linux to split the NANDFS into two smaller files, or the uboot in DRAM will be overwritten because the execution point of uboot is somewhere in DDR. Once you split two smaller files, it will be a good practice for you to flash nandfs into NAND flash. Just remember that the starting address of nandfs partition is 0x200000. Below is short description. This example split the nan6410.img as a 64MB image and a 62MB image.

```
SMDK6410 # nand erase 200000 7e00000
SMDK6410 # tftp 50000000 nand6410.img.1
SMDK6410 # nand write 50000000 200000 4000000
SMDK6410 # tftp 50000000 nand6410.img.2
SMDK6410 # nand write 50000000 4200000 3e00000
SMDK6410 #
```

After done, reset MXM-6410 evaluation kit and the firmware 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 and set the jumper JP1 to NOR flash boot configuration (shunt 1-2). Install the USB driver and DNW programs from Embedian in your host Windows XP PC. Connect the serial console cable from serial console port (UART0, CN20) of device to the COM port of your Windows XP PC. And connect a USB cable from USB device port (CN19) 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 device and you will see the following screen.

```
+-----+
/ S3C6410 USB OTG Downloader v0.2 (2008.07.04) +
/ System ID : Revision [ 0], Pass [ 1]          +
+-----+
ARMCLK: 532.00MHz  HCLKx2: 266.00MHz  HCLK: 133.00MHz  PCLK: 66.50MHz
```

## **Embedian, Inc.**

*VIC mode / Sync Mode*

*USB host is not connected yet.*

*Waiting for USB host connection.*

*!!! USB host is connected !!!*

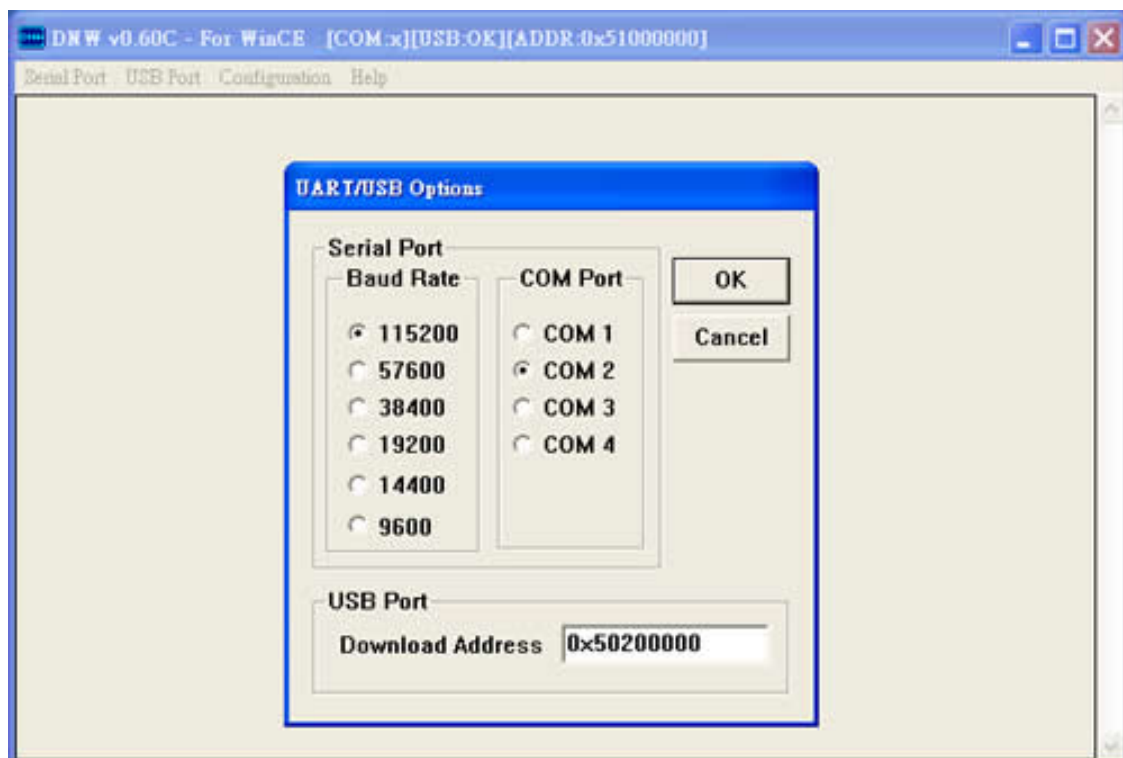
- Bulk In EP : 1*
- Bulk Out EP : 2*
- Speed : High*
- Op Mode : DMA mode*

*Download & Run is selected*

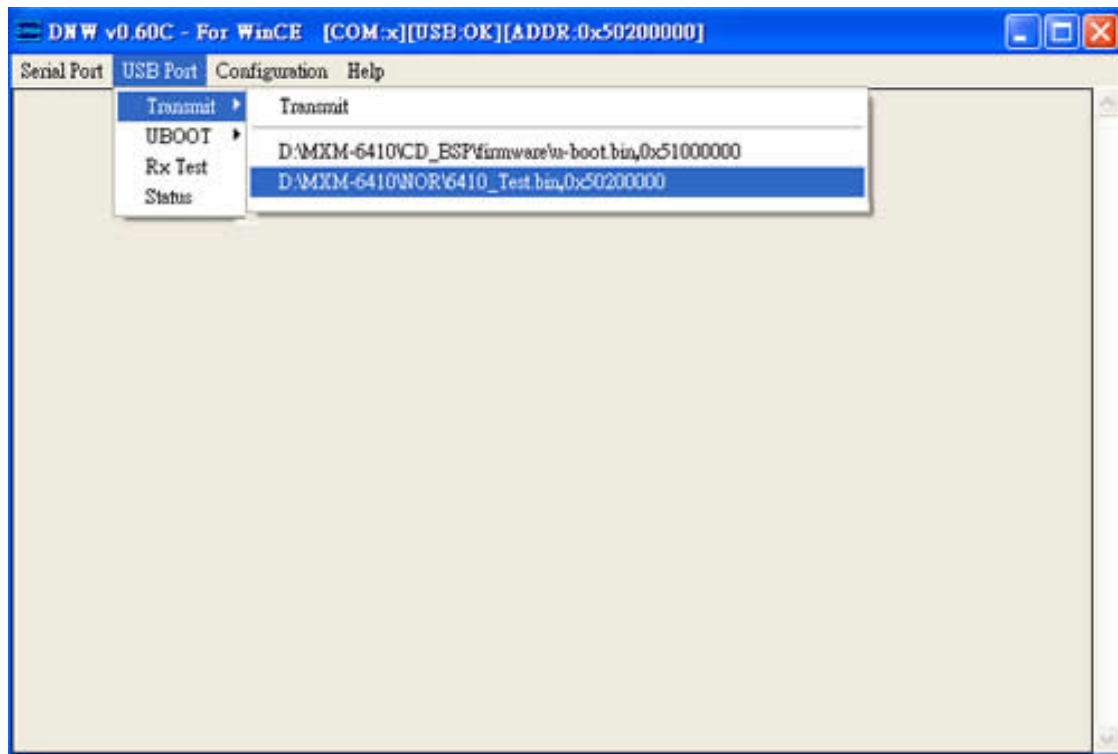
*Select a file to download in DNW*

*If you want to quit, press any key*

Open the DNW program on your Windows XP PC, click on **Configuration** → **Options** and set the USB Port Download Address to 0x50200000 as shown in following figure. After done, click **OK**.



In the DNW program, click on **USB Port** --> **Transmit**, and choose **6410\_Test.bin** file from your PC. This is shown as the following figure.



This is to load the 6410\_Test.bin program to DRAM. This is a program that has many utilities including writing image to NAND flash. After transmit 6410\_Test.bin program to DRAM, you will see the following screen in your Hyperterminal.

```

*****
*          S3C6410 - Test firmware v0.1          *
*****
System ID : Revision [0], Pass [1]
ARMCLK: 532.00MHz  HCLKx2: 266.00MHz  HCLK: 133.00MHz  PCLK: 66.50MHz
VIC mode / Sync Mode

rSPCON: 0xbfcd1500
rGPBCON0: 0x11000000
rGPBCON1: 0x11
rGPHDAT: 0x3c0

0: SFR R/W Test   1: SYSC_Test   2: DMC_Test   3: SROMC_Test
4: ONENAND_Test  5: NAND_Test   6: ATA_Test   7: PCCARD_Test
8: GPIO_Test     9: DMA_Test   10: SBLOCK_Test 11: LCD_Test
12: POST_Test    13: TVENC_Test 14: G2D_Test   15: ROTATOR_Test

```



## **Embedian, Inc.**

```
16: CAMERA_Test   17: MFC_Test      18: JPEG_Test     19: MODEM_Test
20: HOSTIF_Test   21: OtgDev_Test   22: SDMMC_Test     23: HSI_Test
24: SPI_Test      25: IIC_Test      26: UART_Test      27: Timer_Test
28: RTC_Test      29: AC97_Test     30: I2S_Test       31: PCM_Test
32: IRDA_Test     33: ADCTS_Test    34: KEYPAD_Test    35: FIMG3D_Test
36: DM9000_Test   37: 16C752B_Test
```

*Select the function to test : 5*

Enter **5** (5: NAND\_Test), and you will see the following screen.

```
[NAND_Test]
```

```
0: K9F1208(Normal 8bit)
1: K9F2G08(Advanced 8bit)
2: K9HBG08(MLC 8bit)
```

*Select the function to test : 1*

Enter **1** (1: K9F2G08(Advanced 8bit)), and you will see the following screen.

```
[NandT_Advanced8bit]
```

```
0: Read ID
1: Reset command
2: View Invalid Block
3: Page Read&Write
4: Block Erase
5: Binary program
6: Lock&Unlock
7: ECC
8: IROM Code Function
9: Performance
10: Full Function
```

*Select the function to test : 4*

We would like to erase the contents in NAND flash first. Enter **4** (4: Block Erase), and you will see the following screen.

*[NANDT\_Erase]*

*0: Single Block Erase*

*1: Multi Block Erase*

*2: Non\_Check Invalid Data Single block erase*

*3: Non\_Check Invalid Data Multi block erase*

*Select the function to test : 3*

Enter **3** (3: Non\_Check Invalid Data Multi block erase), and you will see the following screen.

*[NANDT\_EraseMultiBlock]*

*Input the Start Block Number to erase the multi-block[multiples of 2 : 0~2047]0*

At the *Input the Start Block Number to erase the multi-block*, enter **0**.

At the *Input the Block Number to erase the multi-block*, enter **2048** as shown in the following screen.

*[NANDT\_EraseMultiBlock]*

*Input the Start Block Number to erase the multi-block[multiples of 2 : 0~2047]0*

*Input the Block Number to erase[multiples of 2 : 2~2048]2048*

*Skip invalid block #780*

*Skip invalid block #1804*

*NAND Erase Block[0 ~ 2047 Block] : Success*

*0: Single Block Erase*

*1: Multi Block Erase*

*2: Non\_Check Invalid Data Single block erase*

*3: Non\_Check Invalid Data Multi block erase*

*Select the function to test :*

## **Embedian, Inc.**

Press **Enter** to exit to the previous screen and to the main screen.

```
*****
*      S3C6410 - Test firmware v0.1      *
*****

System ID : Revision [0], Pass [1]
ARMCLK: 532.00MHz  HCLKx2: 266.00MHz  HCLK: 133.00MHz  PCLK: 66.50MHz
VIC mode / Sync Mode

rSPCON: 0xbfcd1500
rGPHCON0: 0x11000000
rGPHCON1: 0x11
rGPHDAT: 0x3c0

0: SFR_R/W_Test   1: SYSC_Test   2: DMC_Test   3: SROMC_Test
4: ONENAND_Test   5: NAND_Test   6: ATA_Test   7: PCCARD_Test
8: GPIO_Test      9: DMA_Test   10: SBLOCK_Test 11: LCD_Test
12: POST_Test     13: TVENC_Test 14: G2D_Test  15: ROTATOR_Test
16: CAMERA_Test   17: MFC_Test   18: JPEG_Test  19: MODEM_Test
20: HOSTIF_Test   21: OtgDev_Test 22: SDMMC_Test 23: HSI_Test
24: SPI_Test      25: IIC_Test   26: UART_Test  27: Timer_Test
28: RTC_Test      29: AC97_Test 30: I2S_Test   31: PCM_Test
32: IRDA_Test     33: ADCTS_Test 34: KEYPAD_Test 35: FIMG3D_Test
36: DM9000_Test   37: 16C752B_Test

Select the function to test : 21
```

Select **21** (21: OtgDev\_Test) and you will see the following screen.

```
Select the function to test : 21

0: Download Only
1: Upload Only
2: Select Op Mode

Select the function to test : 0
```

You will be asking to enter the temporary download address. Enter 0x51000000 as following screen.

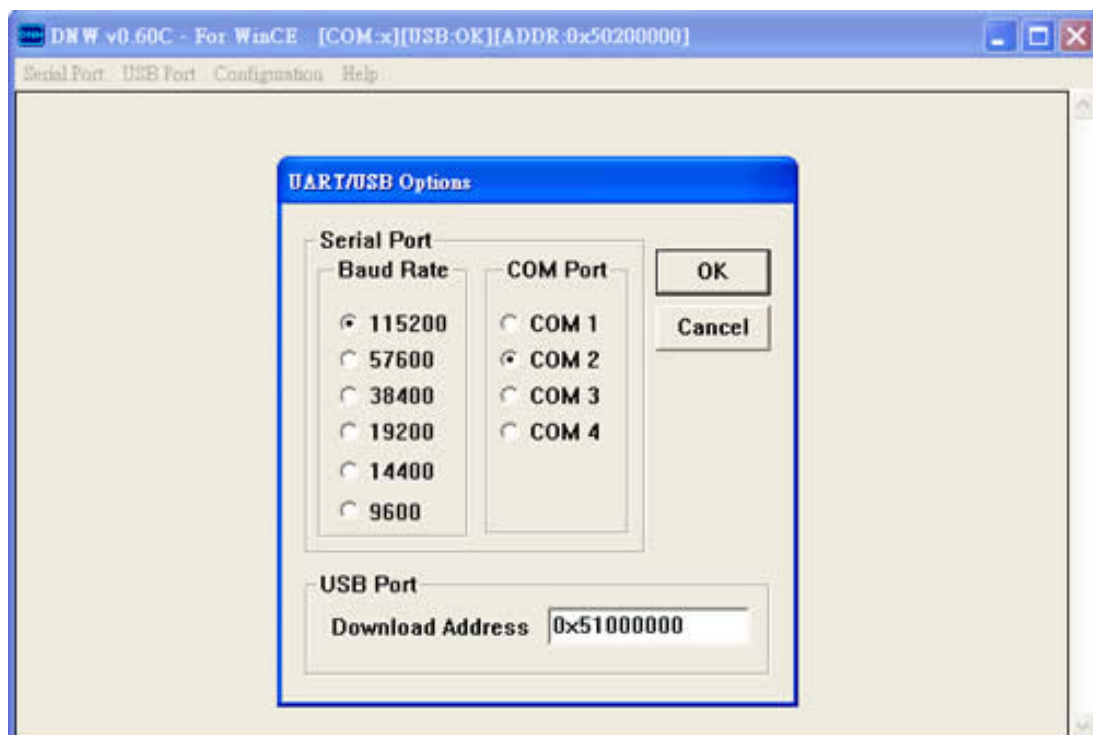
*Enter the download address(0x...):0x51000000  
The temporary download address is 0x51000000.*

*!!! USB host is connected (Speed : High) !!!*

*The temporary download address is 0x51000000.*

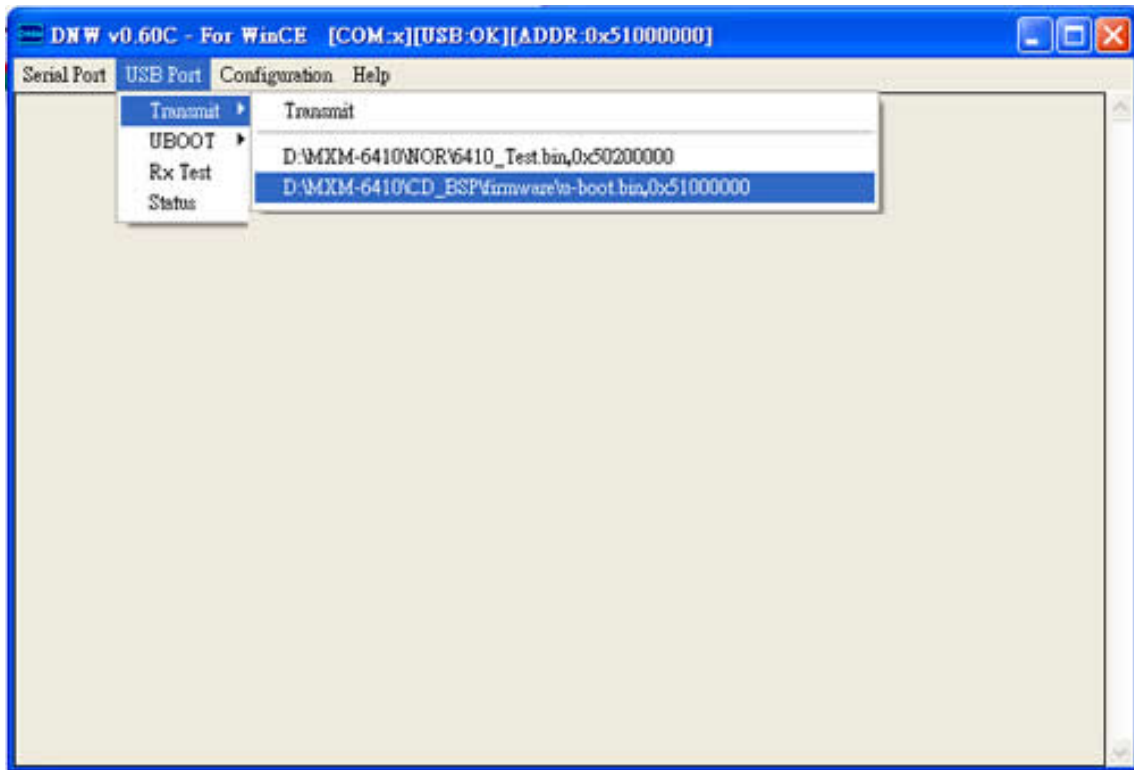
*Select a file to download in DNW  
If you want to quit, press 'x' key*

Next, we need to transmit the *uboot.bin* file to DRAM first. At the DNW program, click on **Configuration** → **Options** again and set the *USB Port Download Address* to 0x51000000 as shown in following figure. After done, click **OK**.



## **Embedian, Inc.**

In the DNW program, click on **USB Port → Transmit**, and choose *uboot.bin* file from your PC. This is shown as the following figure.



You will see the following screen in your Hyperterminal.

```
If you want to quit, press ' x' key

[ADDRESS:51000000h,TOTAL:196618(0x30000)]

Checksum is being calculated....
(If you want to skip, press ' x' key)

Checksum O.K.

0: Download Only
1: Upload Only
2: Select Op Mode

Select the function to test :
```

The uboot now has been uploaded to the DRAM. Next, we would like to fuse the uboot to NAND flash. Press **Enter** to exit to the previous screen and to the main screen.

```
*****
*          S3C6410 - Test firmware v0.1          *
*****

System ID : Revision [0], Pass [1]
ARMCLK: 532.00MHz  HCLKx2: 266.00MHz  HCLK: 133.00MHz  PCLK: 66.50MHz
VIC mode / Sync Mode

rSPCON: 0xbfcd1500
rGPHCON0: 0x11000000
rGPHCON1: 0x11
rGPHDAT: 0x3c0

0: SFR_R/W_Test    1: SYSC_Test      2: DMC_Test        3: SROMC_Test
4: ONENAND_Test    5: NAND_Test      6: ATA_Test        7: PCCARD_Test
8: GPIO_Test       9: DMA_Test      10: SBLOCK_Test    11: LCD_Test
12: POST_Test      13: TVENC_Test     14: G2D_Test       15: ROTATOR_Test
16: CAMERA_Test    17: MFC_Test       18: JPEG_Test      19: MODEM_Test
20: HOSTIF_Test    21: OtgDev_Test    22: SDMMC_Test     23: HSI_Test
24: SPI_Test       25: IIC_Test      26: UART_Test      27: Timer_Test
28: RTC_Test       29: AC97_Test    30: I2S_Test       31: PCM_Test
32: IRDA_Test      33: ADCTS_Test     34: KEYPAD_Test    35: FIMG3D_Test
36: DM9000_Test    37: 16C752B_Test

Select the function to test : 5
```

Select **5** (5: NAND\_Test) and you will see the following screen.

```
Select the function to test : 5

[NAND_Test]

0: K9F1208(Normal 8bit)
1: K9F2G08(Advanced 8bit)
2: K9HBG08(MLC 8bit)

Select the function to test : 1
```

**Embedian, Inc.**

Select **1** (1: K9F2G08(Advanced 8bit)) and you will see the following screen.

*Select the function to test : 1*

*[NandT\_Advanced8bit]*

*0: Read ID*

*1: Reset command*

*2: View Invalid Block*

*3: Page Read&Write*

*4: Block Erase*

*5: Binary program*

*6: Lock&Unlock*

*7: ECC*

*8: IROM Code Function*

*9: Performance*

*10: Full Function*

*Select the function to test : 5*

Select **5** (5: Binary program) and you will see the following screen.

*Select the function to test : 5*

*[NANDT\_ProgramBinary]*

*Caution : You must put BINARY file into 0x51000000 before programming  
Input the Block Number to write[0~2047]*

At Input the Block Number to write[0~2047], enter **0**  
At Input the Page Number to write[0~63], enter **0**  
At Input the Size to program[Bytes] :, enter **196618**  
This is shown as following.

```
Input the Block Number to write[0~2047]0
Input the Page Number to write[0~63]0
Input the Size to program[Bytes] : 196618
```

.

*NAND Program Complete*

- 0: Read ID
- 1: Reset command
- 2: View Invalid Block
- 3: Page Read&Write
- 4: Block Erase
- 5: Binary program
- 6: Lock&Unlock
- 7: ECC
- 8: IROM Code Function
- 9: Performance
- 10: Full Function

*Select the function to test :*

Now, you have successfully write uboot.bin into NAND flash. You can remove the jumper block on JP1 and you will see the uboot booting up.

```
K
U-Boot 1.1.6 (Sep 30 2009 - 16:59:05) for SMDK6410

CPU:      S3C6410@666MHz
          Fclk = 666MHz, Hclk = 133MHz, Pclk = 66MHz, Serial = CLKUART
(ASYNC Mode)
Board:    SMDK6410
DRAM:     128 MB
Flash:    0 kB
NAND:     128 MB
```



## **Embedian, Inc.**

```
*** Warning - bad CRC or NAND, using default environment
```

```
In:      serial
```

```
Out:     serial
```

```
Err:     serial
```

```
found DM9000 ID:90000a46
```

```
DM9000 work in 16 bus width
```

```
MAC: 10:d:32:10:1:12:
```

```
Hit any key to stop autoboot:  0
```

```
SMDK6410 #
```

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 and nand6410.img in this way. Just remember to put the correct input target block number and file size.

# Appendix



## Windows CE 6.0 Firmware Update

This Chapter details how to update Windows CE 6.0 firmware in NAND flash.

Section include :

- Firmware Architecture
- Restore Stepldr and EBOOT from NOR flash
- Restore NK image from EBOOT
- Restore firmware in NAND flash from SD Boot

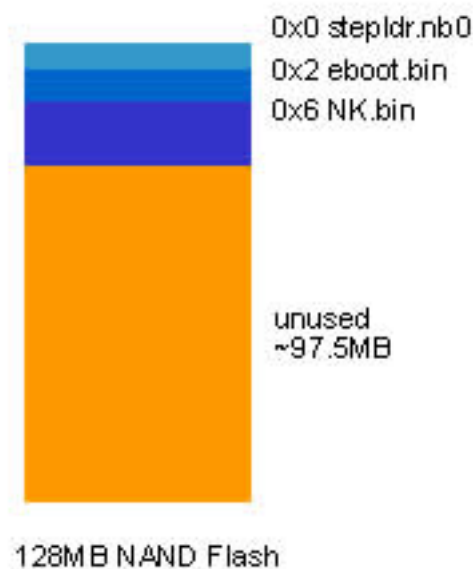
## Appendix II MXM-6410 Windows CE 6.0 Firmware Update

This Chapter details firmware upgrade for MXM-6410 when using Windows CE 6.0 as the operating systems. The firmware in NAND flash includes stepldr.nb0, eoot.bin and NK.bin image.

### A.2.1 Firmware Architecture in NAND flash

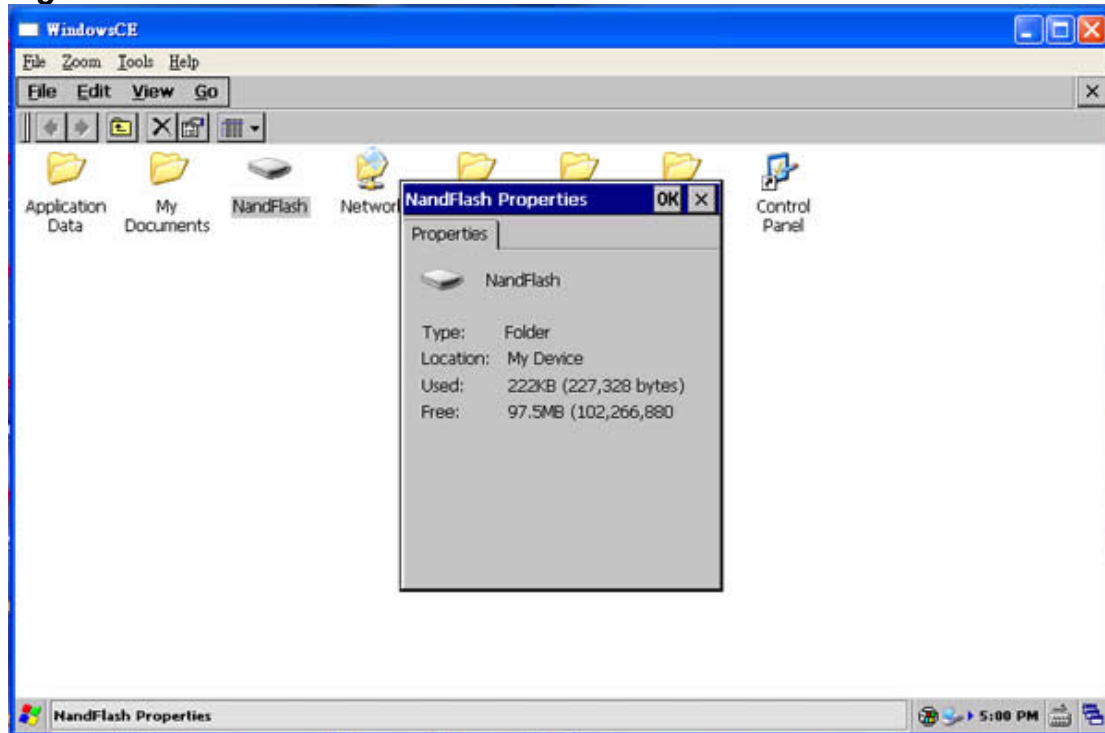
Figure A.2.1 shows the firmware architecture in NAND flash.

**Figure A.2.1** Firmware Architecture in NAND Flash



When power on, the processor will copy the first 4K of BootStrap Loader (stepldr.nb0) to internal RAM. The stepldr will initialize the memory mapping and load the eboot from the 2<sup>nd</sup> block (0x2, 1 block = 128K) of NAND flash. The WinCE 6.0 kernel (NK.bin) starts from the 6th block (0x6). There are about 97.5MB un-used in NAND for users. User can see the flash information at My Device\NandFlash when booting up. Click the mouse or touch on the icon for a few seconds and Tab the “Properties” as Figure A.2.2.

**Figure A.2.2** Flash Information



The purposes of this document are to tell users how to restore or upgrade the firmware in NAND flash. The next chapter will detail that.

### **A.2.2 Restore Stepldr and EBOOT from NOR flash**

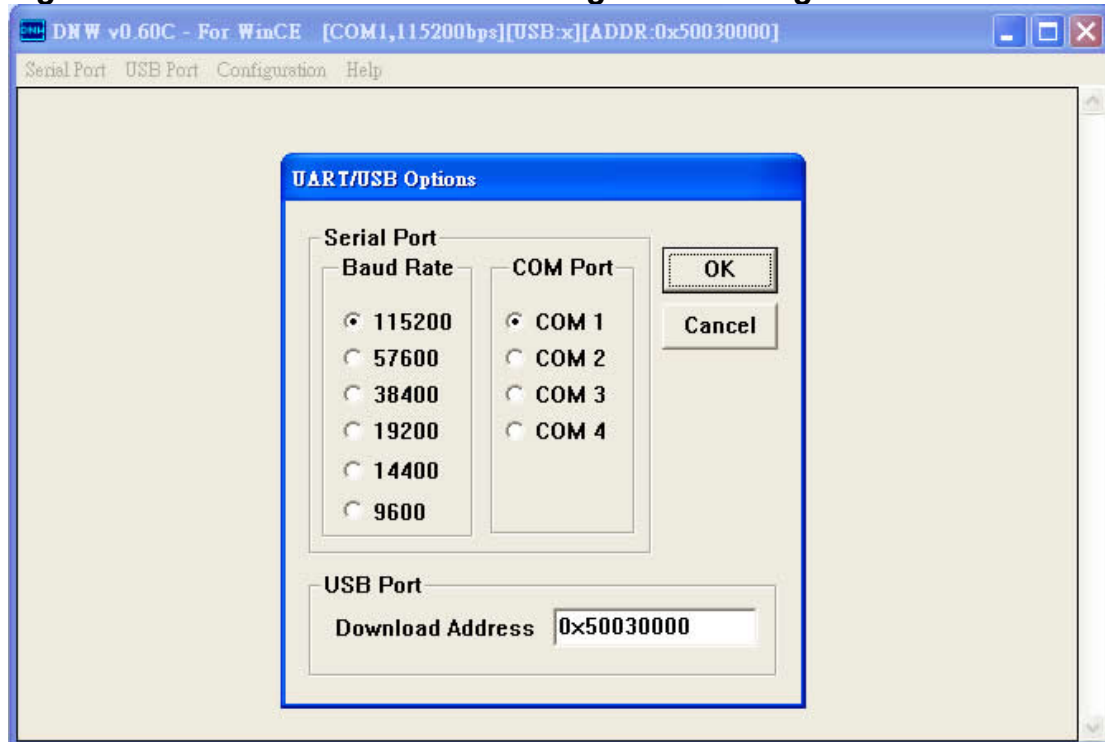
User will need this chapter in case that your EBOOT is erased by accident or gone for somehow. To write Stepldr and EBOOT images to NAND flash using NOR boot, first, power off MXM-6410 evaluation kit and set the jumper JP1 to NOR flash configuration. (Be sure to power off the system before you changing the jumper.)

Install the USB host driver to your Windows system and DNW v0.6C programs that Embedian provided in your host Windows PC. Connect the serial console cable from serial console port (UART 0 in CN20 of MXM-6410 evaluation kit) of device to the COM port of your Windows PC. And connect a USB cable from USB device port (CN19 of Embedian SBC) to USB host port of your Windows PC.

In "**Serial Port**" of your dnw program, click on "**Connect**" and power on MXM-6410 evaluation kit.

In "**Configuration**" of dnw program, at "**Options**" set the COM port as the following setting of figure A.2.3. And the USB Port Download Address is "**0x50030000**", this is the DDR DRAM address where the program will be executed. The baud rate is 115200 and the host COM port is this example is "**COM 1**"

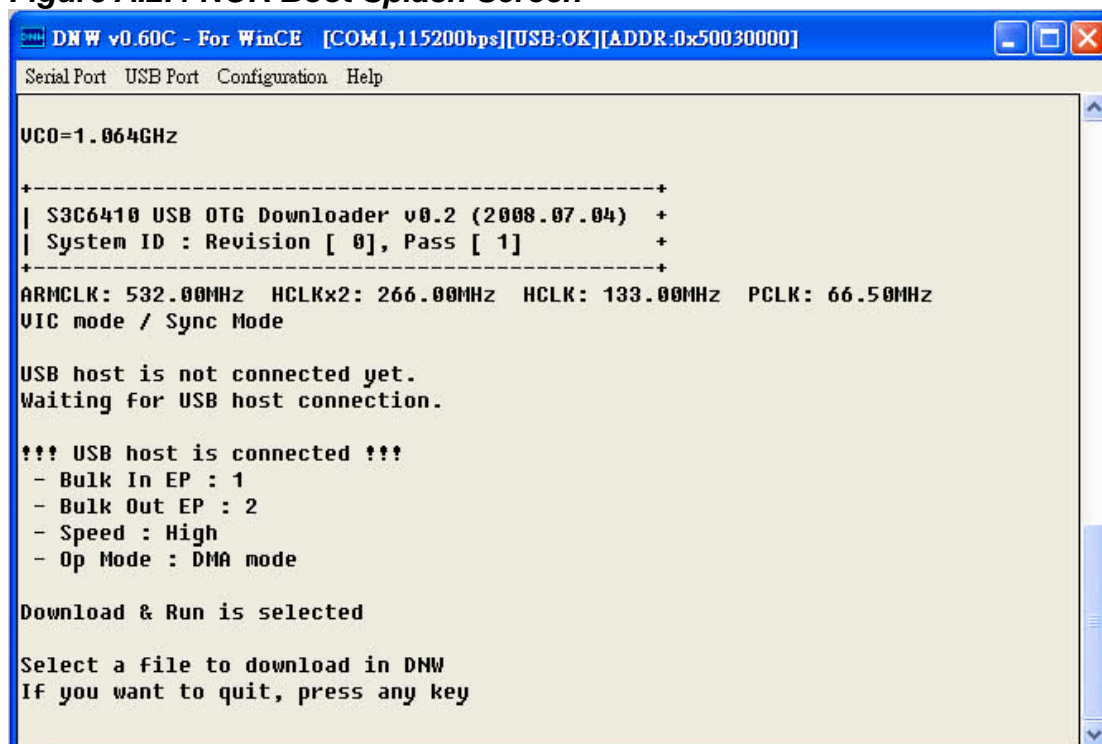
**Figure A.2.3 COM and USB Port Setting of DNW Program**



## **Step 1**

Power on the MXM-6410 evaluation kit and you should see the following screen in your DNW programs as shown in figure A.2.4.

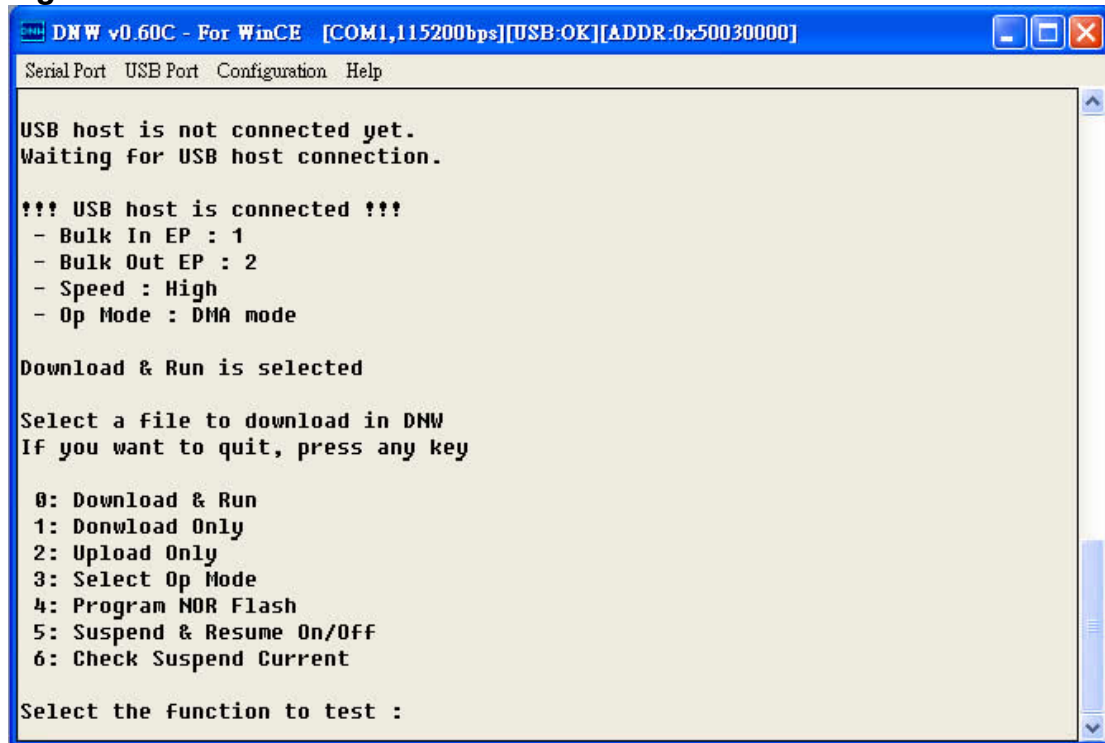
**Figure A.2.4 NOR Boot Splash Screen**



## **Step 2**

Press any key and you will enter the main menu as shown in figure A.2.5.

**Figure A.2.5 NOR Boot Menu**

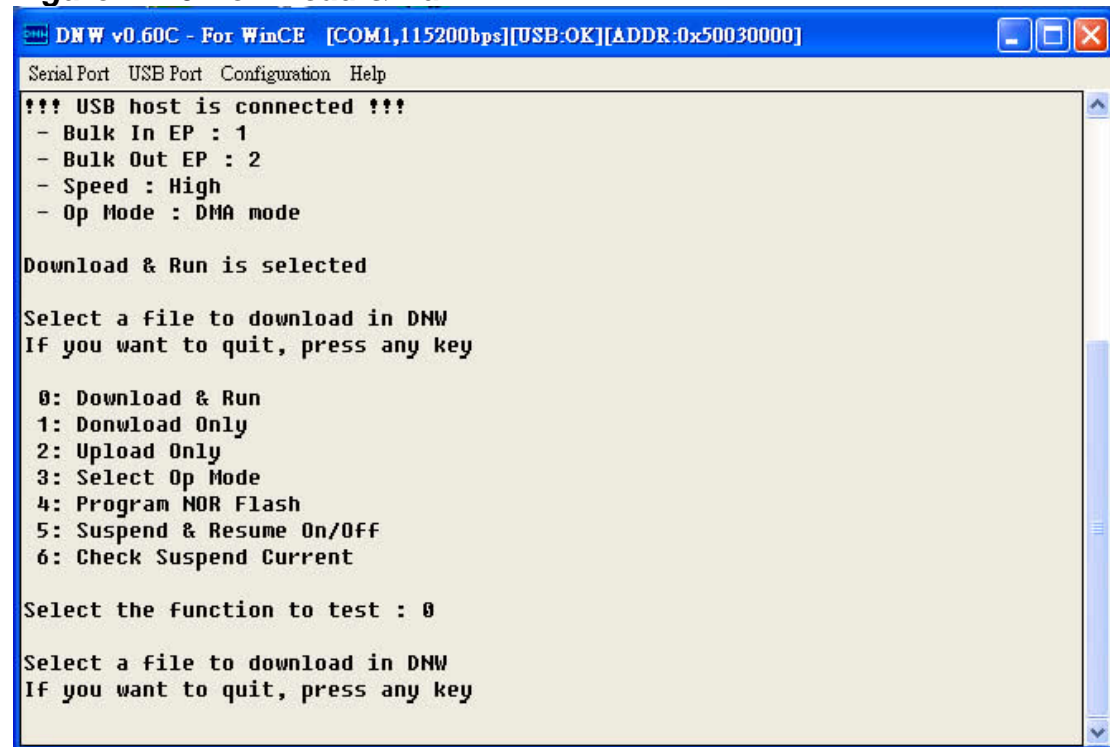




### **Step 3**

Select the function **"0: Download & Run"** and press **"Enter"**. You will see the following screen.

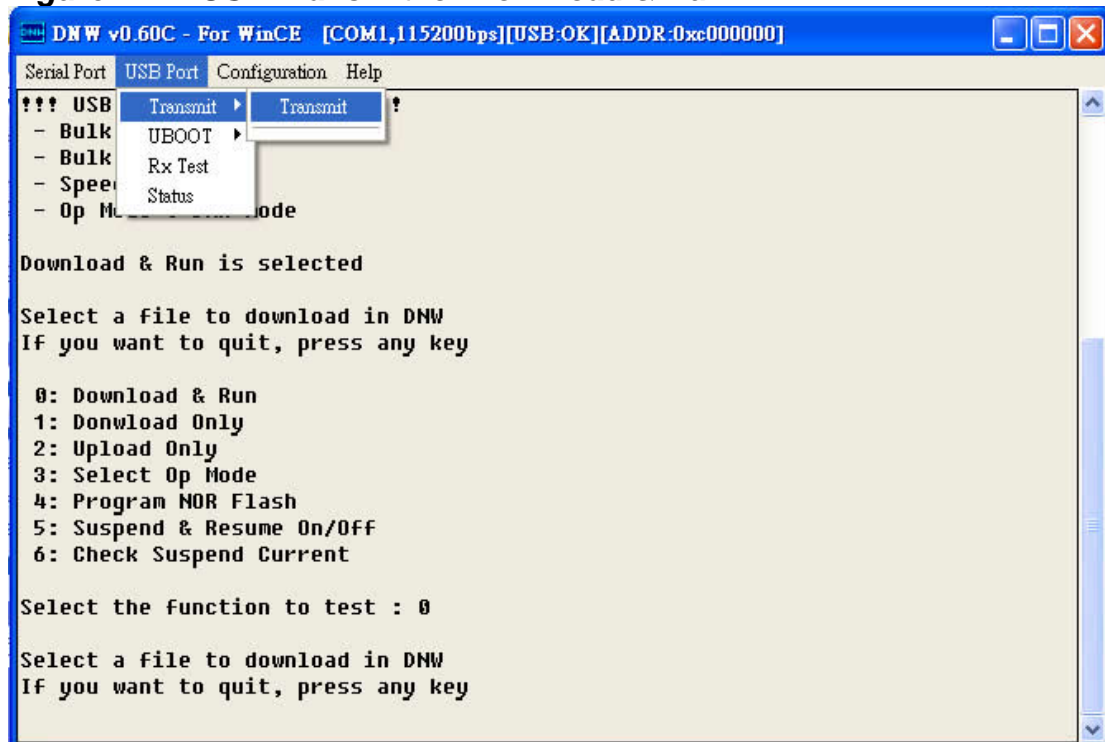
**Figure A.2.6 Download & Run**



#### Step 4

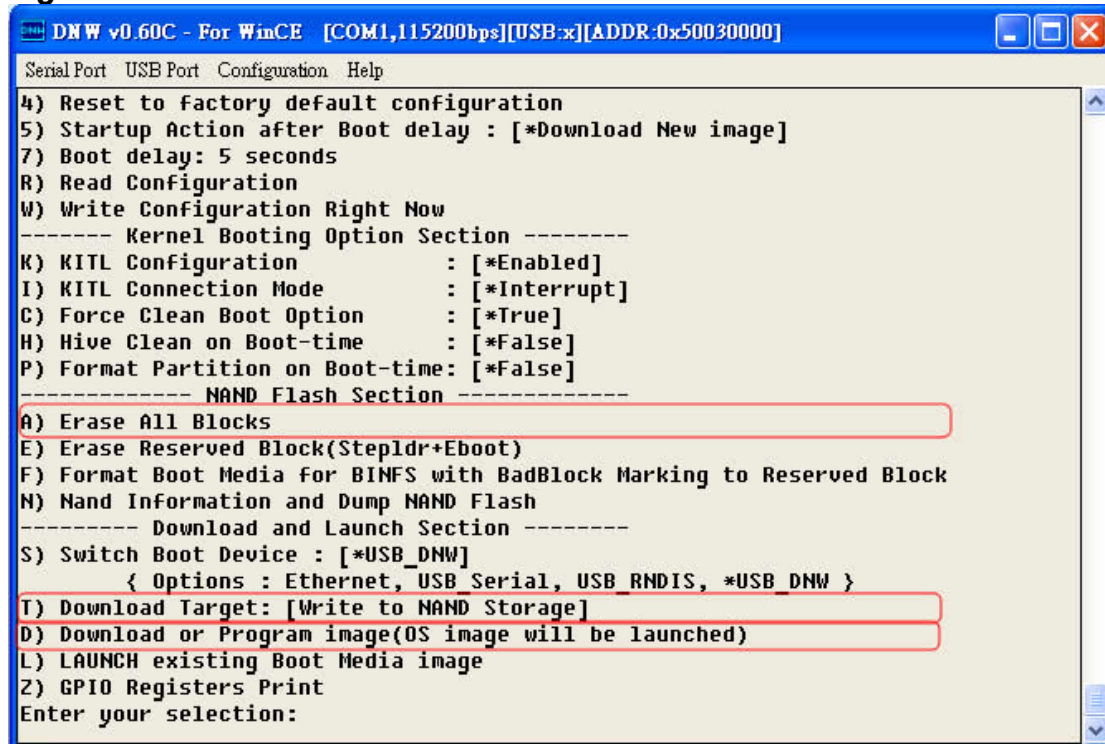
At “USB Port” Tab, Select “Transmit” → “Transmit” as figure A.2.7 shown.

**Figure A.2.7 USB Transmit for Download & Run**



A file browser will pop up and select the folder where you put the **EBOOT.nb0** file. Then the EBOOT will be download the DDR RAM and run. You will see the EBOOT “initiating image download in 5 seconds” messages. Press [Space] bar to enter the EBOOT menu during these 5 seconds as shown in figure A.2.8.

Figure A.2.8 EBOOT Menu



## Step 5

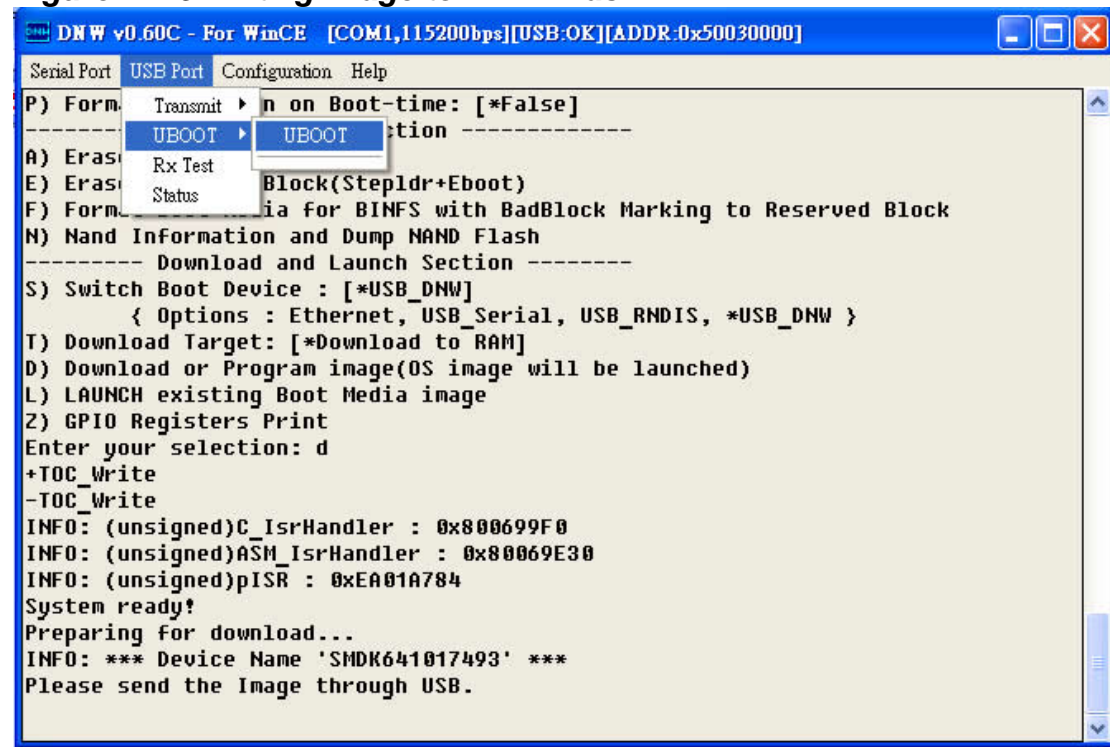
First of all, we recommend you press **A) Erase All Blocks** to erase everything in NAND flash.

Next, press **T) Download Target: [\*Download to RAM]** and change to **T) Download Target: [Write to NAND Storage]**. (By pressing "T" and you can change back and forth.) This is to tell EBOOT that we are going to write image to NAND flash instead of DDR RAM. Press "W" to save this EBOOT configuration.

And press **D) Download or Program image (OS image will be launched)**  
You will see the following screen shown as figure A.2.9.

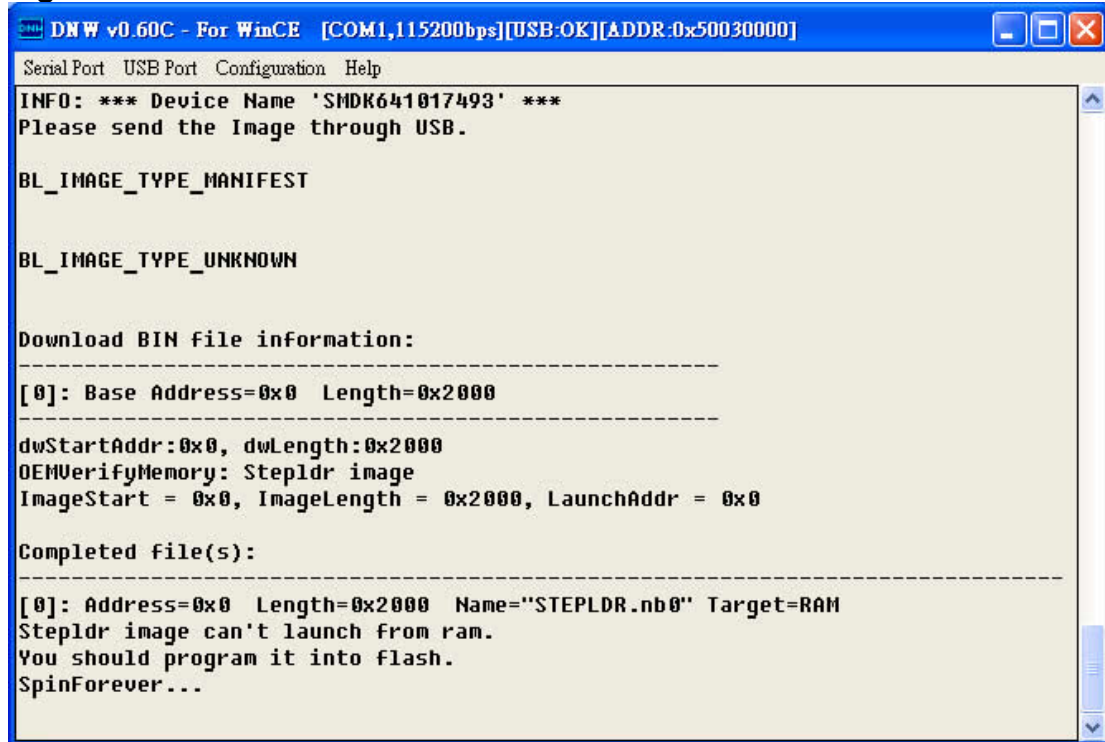
## Step 6

**Figure A.2.9 Writing image to NAND flash**



At “**USB Port**” Tab, select **UBOOT** → **UBOOT** as shown in figure A.2.9. (Note: because we are going to write image to NAND flash, choose UBOOT instead Transmit. Here is different from that in step 4.) A file browser will ask you to select a file that you would like to write to NAND flash. Find the file “**STEPLDR.nb0**” in your PC and click. And you will see the following screen as shown in figure A.2.10.

**Figure A.2.10 Write STEPLDR.nb0 to NAND**



Now you have successfully write the STEPLDR.nb0 file to NAND flash.  
Next, we would like to write **EBOOT.bin** to NAND flash.

### **Step 7**

Repeat steps 1~6, but **DON'T** erase all blocks at step 5 because you have already written the STEPLDR.nb0 to NAND flash. And at step 6 choose the **EBOOT.bin** file. (Note: It is EBOOT.bin instead of EBOOT.nb0.) After writing EBOOT.bin to NAND flash you should be able to see the screen as shown in figure A.2.11.

**Figure A.2.11 Write EBOOT.bin to NAND**

```
DNW v0.60C - For WinCE [COM1,115200bps][USB:OK][ADDR:0x50030000]
Serial Port USB Port Configuration Help
dwTtlSectors: 0x8A
dwLoadAddress: 0x80030000
dwJumpAddress: 0x80060C04
dwStoreOffset: 0x0
sgList[0].dwSector: 0x80
sgList[0].dwLength: 0x8A
}
ID[1] {
  dwVersion: 0x1
  dwSignature: 0x43465348
  String: ''
  dwImageType: 0x2
  dwTtlSectors: 0x0
  dwLoadAddress: 0x0
  dwJumpAddress: 0x0
  dwStoreOffset: 0x0
}
!!!Invalid Image Descriptor: id[2]=0x0
chainInfo.dwLoadAddress: 0X00000000
chainInfo.dwFlashAddress: 0X00000000
chainInfo.dwLength: 0X00000000
}
INFO: Eboot image stored to Smart Media. Please Reboot. Halting...
```

As of now you have successfully written the STEPLDR.nb0 and EBOOT.bin files to NAND flash.

You can turn the power off and set the JP1 jumper to NAND boot and turn the power on again. You should be able to hear a short beep sound and see the splash screen on the LCD as shown in figure A.2.12.

**Figure A.2.12 EBOOT Splash Screen**



Next section, we are going to introduce how to restore/upgrade the NK image from NAND flash by using EBOOT.

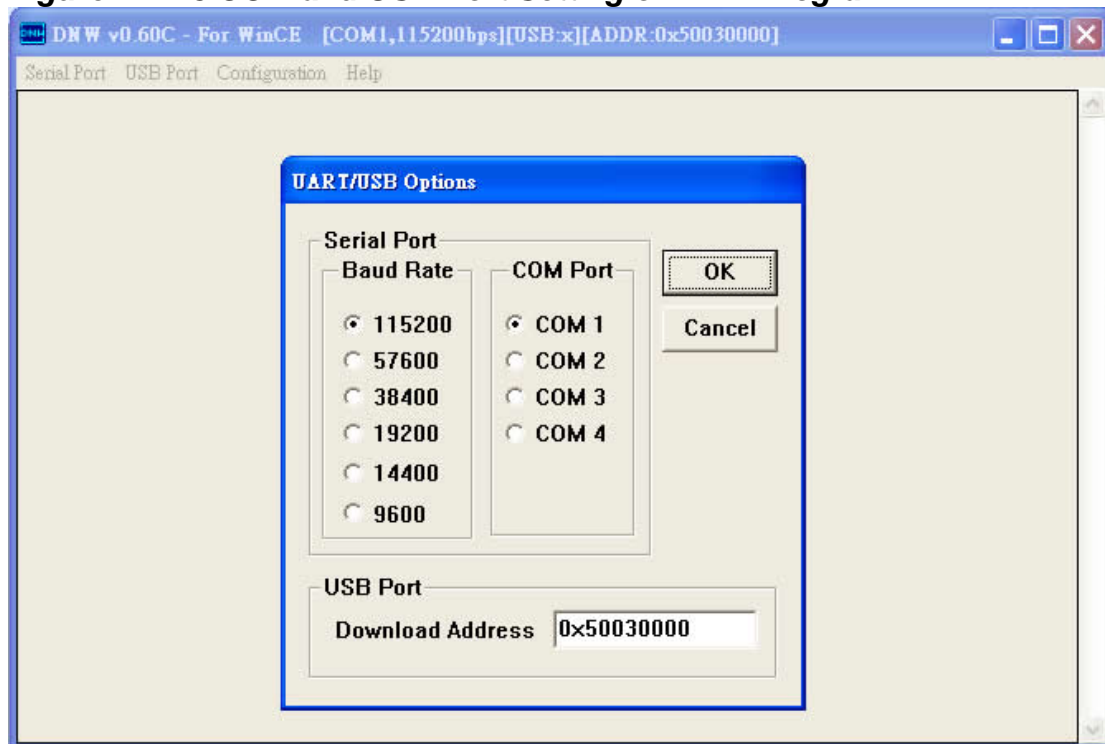
### **A.2.3 Restore NK image from EBOOT**

In previous section, we detailed how to restore/upgrade the steploader and EBOOT from NOR flash utilities. Once you have steploader and eboot up and running in NAND flash. You can use the EBOOT to restore or upgrade the NK image.

Same as section 3.1, first, you need to connect the serial console cable from serial console port (UART 0 in CN20 of MXM-6410 evaluation kit) of device to the COM port of your Windows PC. And connect a USB cable from USB device port (CN19 of Embedian SBC) to USB host port of your Windows PC. In **"Serial Port"** of your dnw program, click on **"Connect"** and power on MXM-6410 evaluation kit.

In **"Configuration"** of dnw program, at **"Options"** set the COM port as the following setting of figure A.2.13. And the USB Port Download Address is **"0x50030000"**, this is the DDR DRAM address where the program will be executed. The baud rate is 115200 and the host COM port is this example is **"COM 1"**

**Figure A.2.13 COM and USB Port Setting of DNW Program**



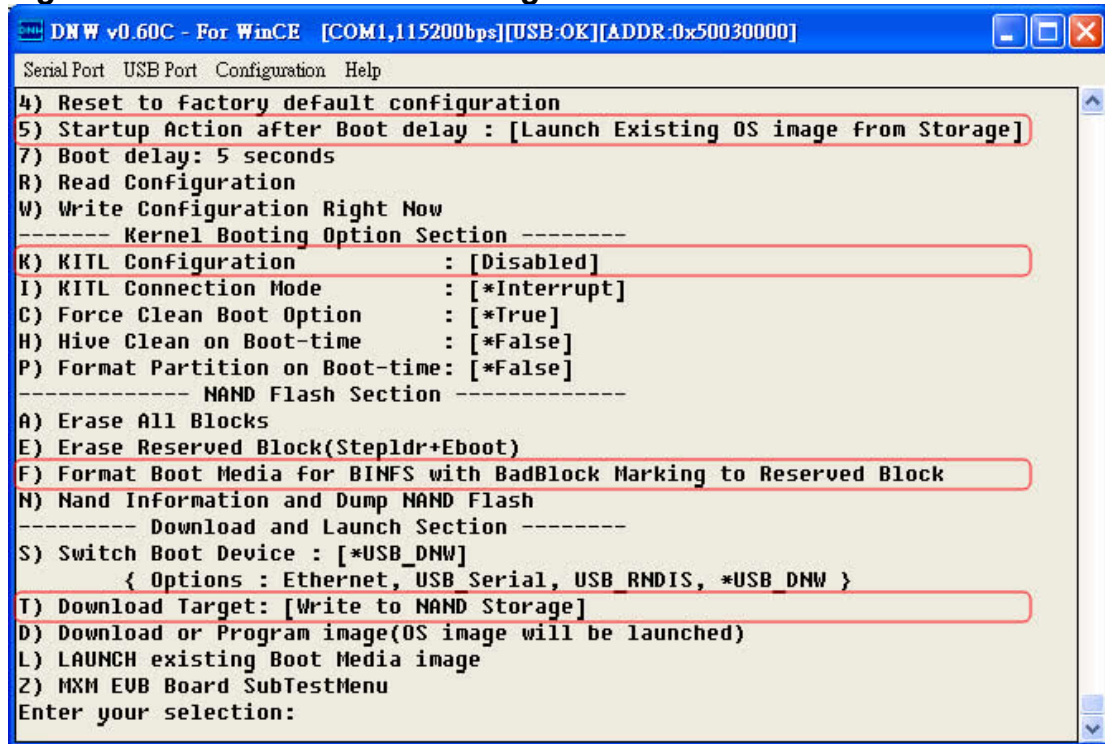


Following is a step-by-step guide.

## Step 1

Set the JP1 to NAND boot (open 1-2 pin of JP1) and power up the MXM-6410 evaluation kit. You should be able to hear a short beep sound and splash screen in the LCD. You will see the EBOOT “initiating image download in 5 seconds” messages. Press **[Space]** bar to enter the EBOOT menu during these 5 seconds as shown in figure A.2.14.

**Figure A.2.14 EBOOT Menu Setting**



## Step 2

First of all, press **5) Startup Action after Boot delay: [Launch Existing OS image from Storage]** to tell EBOOT to load the NK from NAND flash because we are going to write NK to NAND flash later. (The default setting should be 5) Startup Action after Boot delay : [\*Download New image]. Press “5” to change it.)

Next, press **K) KITL Configuration : [Disabled]** to disable the KITL.

Press “W” to save this configuration first.

And next, press **F) Format Boot Media for BINFS with BadBlock Marking to Reserved Block**. This is to format the NK area as BinFS file system and mark the bad blocks in NAND flash.

You will see the following message shortly and then enter into the EBOOT menu again.

Enter your selection: f

Reserving Blocks [0x0 - 0x5] ... Wait for completion



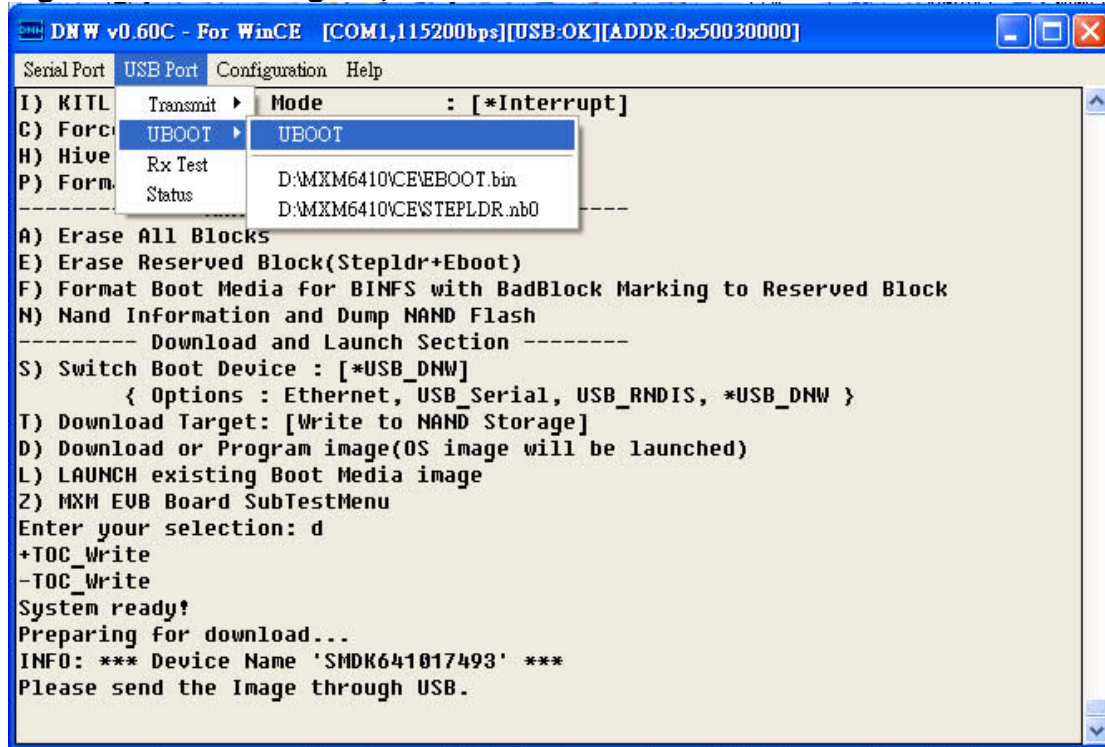
## **Embedian, Inc.**

Reserving is completed.

The last step is to writing **NK.bin** image into NAND flash by pressing **D)** **Download or Program image (OS image will be launched)**

And you will see the following screen as shown in figure A.2.15.

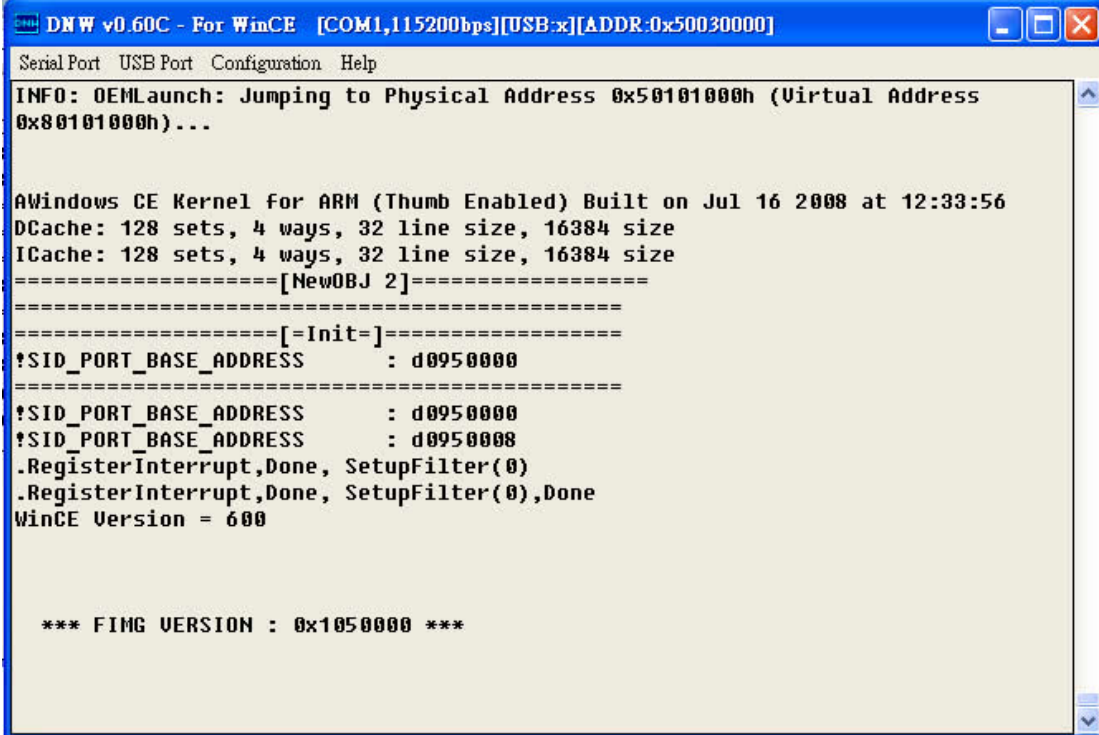
**Figure A.2.15 Writing NK.bin to NAND flash**



### Step 3

At “**USB Port**” Tab, select **UBOOT → UBOOT** as shown in figure A.2.15. (Note: because we are going to write image to NAND flash, choose UBOOT instead Transmit.) A file browser will ask you to select a file that you would like to write to NAND flash. Find the file “**NK.bin**” in your PC and click. Wait for about 30 seconds to let the NK.bin writing into NAND flash. And you will see the following screen as shown in figure A.2.16.

**Figure A.2.16 Done with Writing NK.bin to NAND**



```
DNW v0.60C - For WinCE [COM1,115200bps][USB:x][ADDR:0x50030000]
Serial Port  USB Port  Configuration  Help
INFO: OEMLaunch: Jumping to Physical Address 0x50101000h (Virtual Address
0x80101000h)...

AWindows CE Kernel for ARM (Thumb Enabled) Built on Jul 16 2008 at 12:33:56
DCache: 128 sets, 4 ways, 32 line size, 16384 size
ICache: 128 sets, 4 ways, 32 line size, 16384 size
=====[NewOBJ 2]====
=====[Init]====
!SID_PORT_BASE_ADDRESS : d0950000
====
!SID_PORT_BASE_ADDRESS : d0950000
!SID_PORT_BASE_ADDRESS : d0950000
.RegisterInterrupt,Done, SetupFilter(0)
.RegisterInterrupt,Done, SetupFilter(0),Done
WinCE Version = 600

*** FING VERSION : 0x1050000 ***
```

If you have ActiveSync program (user can download this program for free from Microsoft’s website) installed in your Windows PC and the USB cable is connected to your MXM-6410 evaluation kit, you should be able to see the ActiveSync program pop out and ask you to set up a partnership as shown in figure A.2.17.

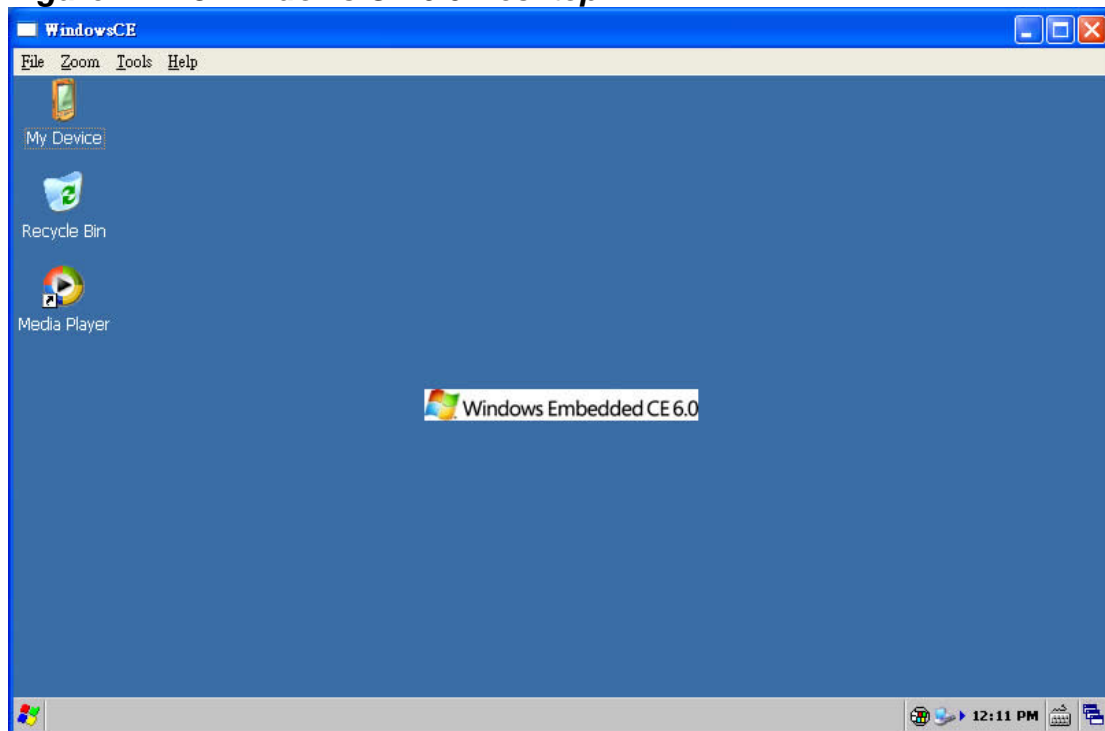
**Figure A.2.17 ActiveSync Program**



Select “**No**” and click “**Next**” of your ActiveSync program.

If you have LCD connected to the MXM-6410 evaluation kit, you should be able to see the Windows CE 6.0 desktop screen as following now. You can also use the remote desktop tool to see the desktop.

**Figure A.2.18 Windows CE 6.0 Desktop**



Now, we have done the firmware restore/upgrade of the Windows CE system.

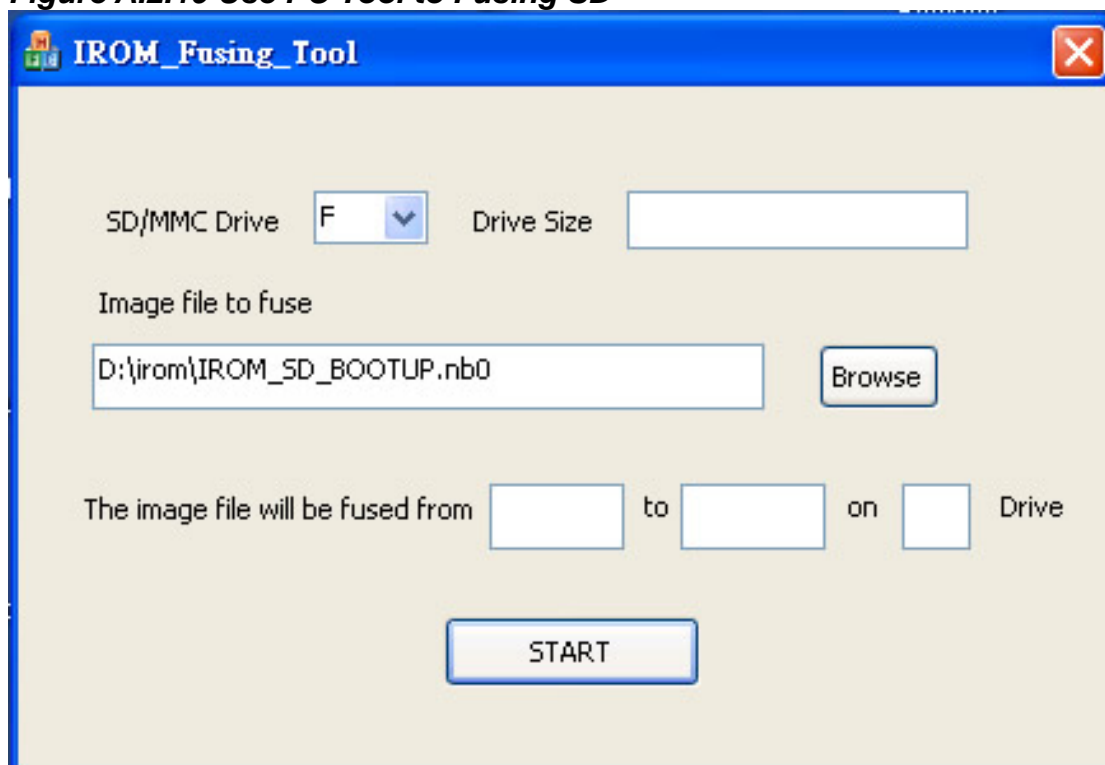
### **A.2.4 Restore firmware in NAND flash from SD Boot**

Embedian also provides a very useful way to recover the firmware in NAND flash from SD card. To brief how this be done, use needs to use the PC tool to write the EBOOT bootable file to a SD card first and then plug that SD card into the MXM-6410 evaluation kit. Set the hardware configuration to SD boot and user will see the EBOOT menu. Details will be described in the following sections.

#### **A.2.4.1. Using PC Tool to Fusing the SD/SDHC card**

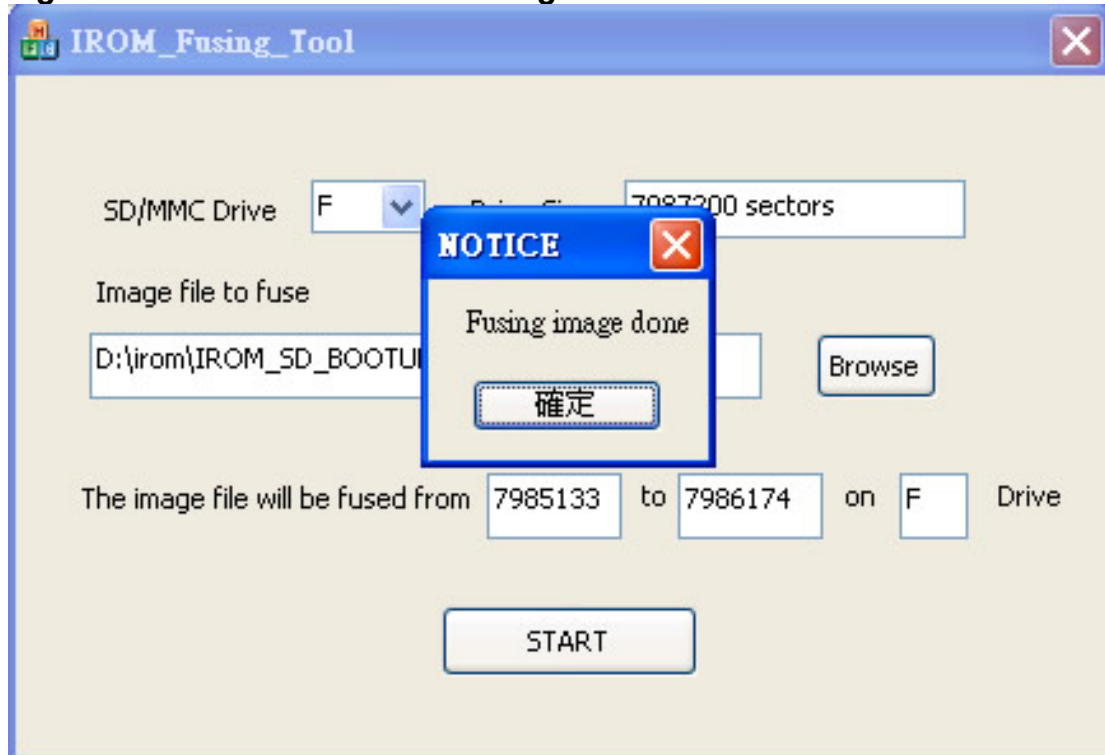
1. Users should format the SD/SDHC card as FAT32 file systems.
2. Run IROM\_Fusing\_Tool.exe (Note: for only SD, MMC, HSMMC card only, not SDHC card. Using IROM\_Fusing\_Tool\_SDHC.exe instead if SDHC cards.)
3. Click the browse button to choose IROM\_SD\_BOOTUP.nb0. (Note: this file already combined the stepldr.nb0, just fusing this file will be fine enough.)
4. Choose the storage drive that SD card is mounted on. **If you choose \*\*NOT\*\* SD card storage drive like HDD drive, It might make a serious problem on your HDD. Please Make sure that storage drive is right.**

**Figure A.2.19 Use PC Tool to Fusing SD**



5. Click "START" button.

**Figure A.2.20 Use PC Tool to Fusing SD Done**



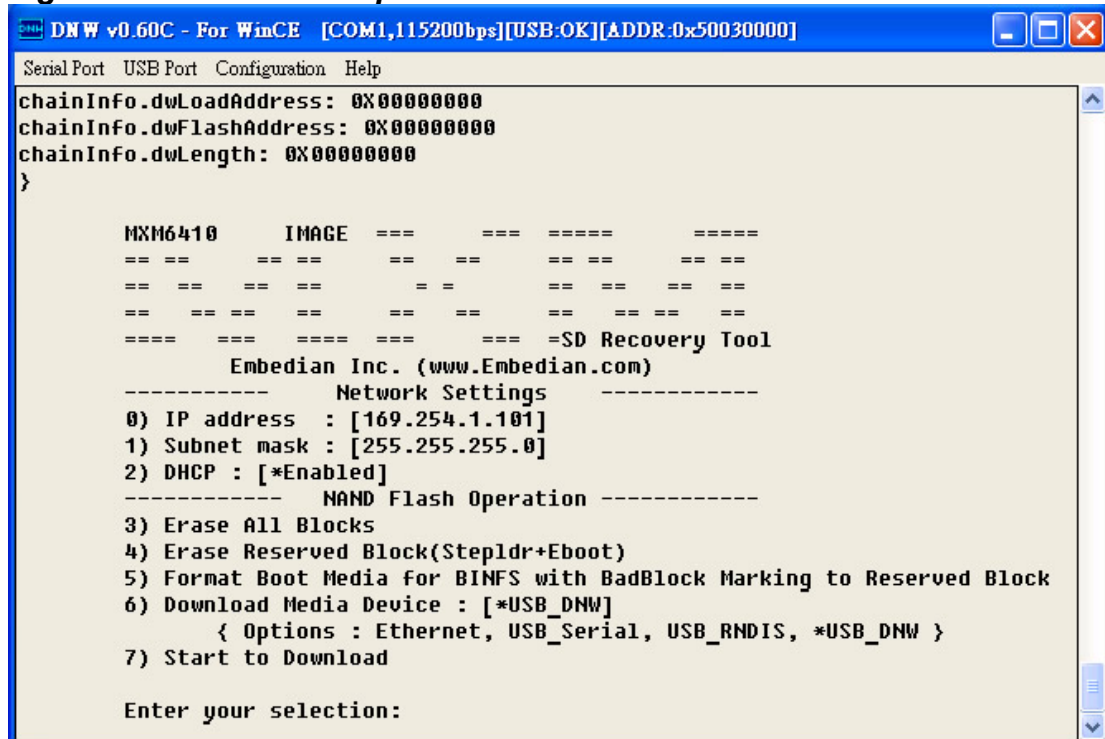
6. If there is the message box like that shown in figure A.2.20, it's done.
7. Remove the SD/MMC card from your PC and put it to the SD card slot of MXM-6410 evaluation kit.
8. Set booting mode as iROM SD booting (shunt jumper block JP5), and turn on the power and you should see the system boot directly from SD now.

Next section, we will detail how to fusing NAND flash using SD boot.

#### **A.2.4.2. Using SD boot fusing NAND flash**

After setting the MXM-6410 evaluation kit to SD boot (shunt jumper block JP5), user should be able to see the following screen after power up.

Figure A.2.21 SD Boot Up Menu



Once you saw the menu shown in figure A.2.21. The fusing procedure will be exactly the same as described in section A.2.2 to restore the NAND flash firmware from NOR flash. The following are the detail step-by-step procedures.

### Step 1

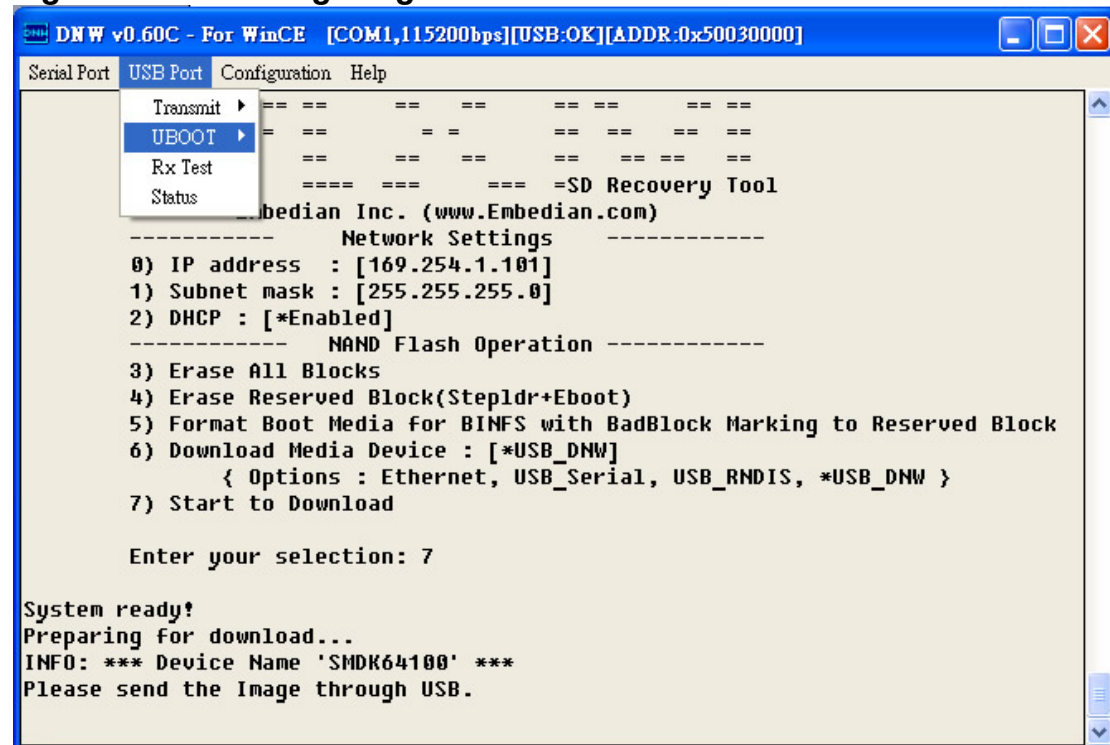
First of all, we recommend you press **3) Erase All Blocks** to erase everything in NAND flash.

Next, press **6) Download Media Device : [\*USB\_DNW]** to choose "USB\_DNW" and press **7) Start to Download**.

You will see the following screen shown as figure A.2.22.

## Step 2

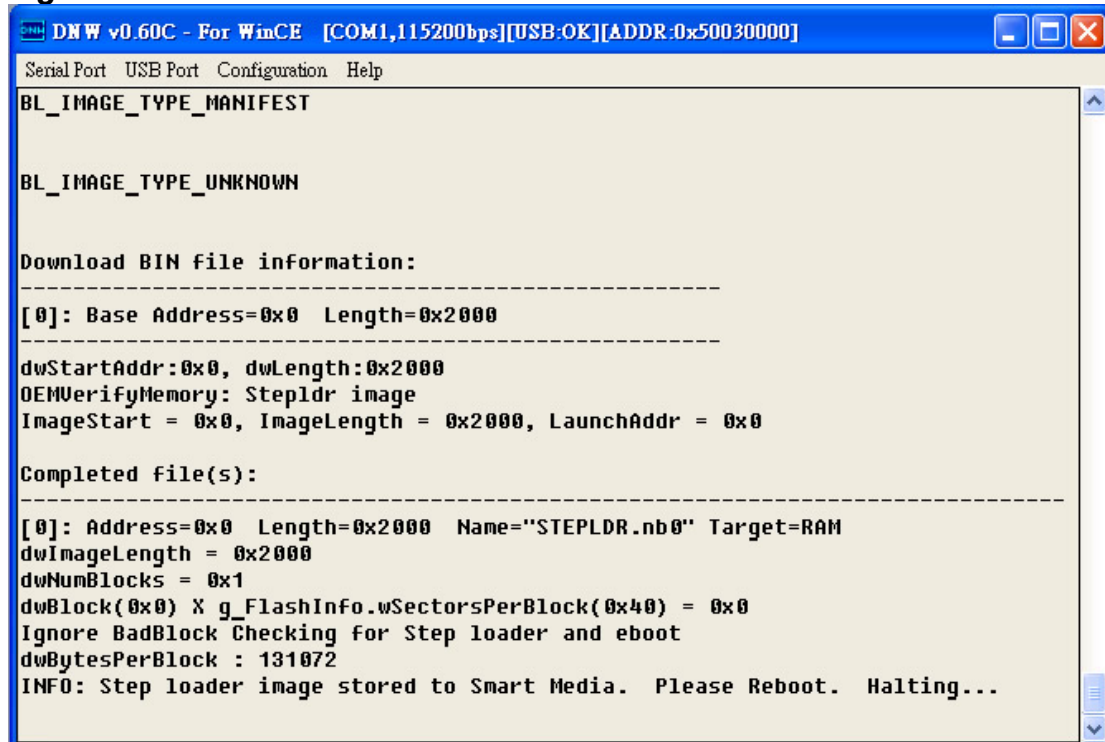
**Figure A.2.22 Writing image to NAND flash at SD Boot**



At “**USB Port**” Tab, select **UBOOT** → **UBOOT** as shown in figure A.2.22. A file browser will ask you to select a file that you would like to write to NAND flash. Find the file “**STEPLDR.nb0**” in your PC and click. And you will see the following screen as shown in figure A.2.23.



**Figure A.2.23 Write STEPLDR.nb0 to NAND at SD Boot**



Now you have successfully write the STEPLDR.nb0 file to NAND flash.  
Next, we would like to write **EBOOT.bin** to NAND flash.

### **Step 3**

Repeat steps 1~2, but **DON'T** erase all blocks at step 1 because you have already written the STEPLDR.nb0 to NAND flash. And at step 2 choose the **EBOOT.bin** file. (Note: It is EBOOT.bin instead of EBOOT.nb0.) After writing EBOOT.bin to NAND flash you should be able to see the screen as shown in figure A.2.24.

**Figure A.2.24 Write EBOOT.bin to NAND at SD boot**

```
DNW v0.60C - For WinCE [COM1,115200bps][USB:OK][ADDR:0x50030000]
Serial Port  USB Port  Configuration  Help
dwTtlSectors: 0x8A
dwLoadAddress: 0x80030000
dwJumpAddress: 0x80060C04
dwStoreOffset: 0x0
sgList[0].dwSector: 0x80
sgList[0].dwLength: 0x8A
}
ID[1] {
  dwVersion: 0x1
  dwSignature: 0x43465348
  String: ''
  dwImageType: 0x2
  dwTtlSectors: 0x0
  dwLoadAddress: 0x0
  dwJumpAddress: 0x0
  dwStoreOffset: 0x0
}
!!!Invalid Image Descriptor: id[2]=0x0
chainInfo.dwLoadAddress: 0X00000000
chainInfo.dwFlashAddress: 0X00000000
chainInfo.dwLength: 0X00000000
}
INFO: Eboot image stored to Smart Media.  Please Reboot.  Halting...
```

As of now you have successfully written the STEPLDR.nb0 and EBOOT.bin files to NAND flash.

You can turn the power off and set the jumper to NAND boot and turn the power on again. You should be able to hear a short beep sound and see the splash screen on the LCD as shown in figure A.2.25.

**Figure A.2.25 EBOOT Splash Screen**



***Embedian, Inc.***

Users now can go back to section A.2.3 to see how to restore/upgrade the NK image from NAND flash by using EBOOT at NAND.