COMX-P2020 BSP

User Guide

P/N: 6806800L84C

December 2019

Trademarks

The stylized "S" and "SMART" is a registered trademark of SMART Modular Technologies, Inc. and “SMART Embedded Computing” and the SMART Embedded Computing logo are trademarks of SMART Modular Technologies, Inc. All other names and logos referred to are trade names, trademarks, or registered trademarks of their respective owners. These materials are provided by SMART Embedded Computing as a service to its customers and may be used for informational purposes only.

Disclaimer*

SMART Embedded Computing (SMART EC) assumes no responsibility for errors or omissions in these materials. **These materials are provided "AS IS" without warranty of any kind, either expressed or implied, including but not limited to, the implied warranties of merchantability, fitness for a particular purpose, or non-infringement.** SMART EC further does not warrant the accuracy or completeness of the information, text, graphics, links or other items contained within these materials. SMART EC shall not be liable for any special, indirect, incidental, or consequential damages, including without limitation, lost revenues or lost profits, which may result from the use of these materials. SMART EC may make changes to these materials, or to the products described therein, at any time without notice. SMART EC makes no commitment to update the information contained within these materials.

Electronic versions of this material may be read online, downloaded for personal use, or referenced in another document as a URL to a SMART EC website. The text itself may not be published commercially in print or electronic form, edited, translated, or otherwise altered without the permission of SMART EC.

It is possible that this publication may contain reference to or information about SMART EC products, programming, or services that are not available in your country. Such references or information must not be construed to mean that SMART EC intends to announce such SMART EC products, programming, or services in your country.

Limited and Restricted Rights Legend

If the documentation contained herein is supplied, directly or indirectly, to the U.S. Government, the following notice shall apply unless otherwise agreed to in writing by SMART Embedded Computing.

Use, duplication, or disclosure by the Government is subject to restrictions as set forth in subparagraph (b)(3) of the Rights in Technical Data clause at DFARS 252.227-7013 (Nov. 1995) and of the Rights in Noncommercial Computer Software and Documentation clause at DFARS 252.227-7014 (Jun. 1995).

SMART Embedded Computing, Inc.

2900 S. Diablo Way, Suite 190

Tempe, Arizona 85282

USA

*For full legal terms and conditions, visit *www.smartembedded.com/ec/legal*

Table of Contents

Table of Contents

Table of Contents

List of Figures

List of Figures

List of Tables

List of Tables

About this Manual

Abbreviations

This document uses the following abbreviations:

Abbreviation	Definition
AMC	Alarm Management Controller
ARP	Address Resolution Protocol

Conventions

The following table describes the conventions used throughout this manual.

Notation	Description
0x00000000	Typical notation for hexadecimal numbers (digits are 0 through F), for example used for addresses and offsets
0b0000	Same for binary numbers (digits are 0 and 1)
bold	Used to emphasize a word
`Screen`	Used for on-screen output and code related elements or commands. `Sample of Programming used in a table (9pt)`
`Courier + Bold`	Used to characterize user input and to separate it from system output
Reference	Used for references and for table and figure descriptions
File > Exit	Notation for selecting a submenu
<text>	Notation for variables and keys
[text]	Notation for software buttons to click on the screen and parameter description
...	Repeated item for example node 1, node 2, ..., node 12
. . .	Omission of information from example/command that is not necessary at the time
..	Ranges, for example: 0..4 means one of the integers 0,1,2,3, and 4 (used in registers)
|	Logical OR

Notation	Description
	Indicates a hazardous situation which, if not avoided, could result in death or serious injury
	Indicates a hazardous situation which, if not avoided, may result in minor or moderate injury
	Indicates a property damage message
	Indicates a hot surface that could result in moderate or serious injury
	Indicates an electrical situation that could result in moderate injury or death
	Indicates that when working in an ESD environment care should be taken to use proper ESD practices
	No danger encountered, pay attention to important information

Summary of Changes

This manual has been revised and replaces all prior editions.

Part Number	Publication Date	Description
6806800L84A	December 2010	GA version
6806800L84B	July 2014	Re-branded to Artesyn
6806800L84C	December 2019	Rebrand to SMART Embedded Computing template

Overview

1.1 Overview

The following files will be released with COMX-P2020:

- COMX-P2020.bsp.tar.gz: The tar ball of BSP image (U-boot, Linux Kernel, and root file system) for COMX-P2020, including the following files:
  - u-boot-spi.bin - U-boot Binary file that could be used to upgrade the U-boot on the SPI Flash.
  - u-boot-sd.bin - U-boot Binary file that could be used to upgrade the U-boot on the SD card.
  - uImage - Linux kernel for COMX-P2020.
  - comx.dtb - Linux device tree blob for COMX-P2020.
  - rootfs-min.ext2.img - The minimum root file system in ext2 format, which is helpful in case of deployment on Flash devices which are of smaller sizes. The size of this file is around 4MB.
  - rootfs-usr.ext2.img - The Medium root file system in ext2 format, which is good enough for deployment perspective and supports most of the use cases identified for Residential Gateway, Wireless LAN Access Point, Media Server, NAS box, etc. The size of this file is around 42MB.
  - rootfs-dev.ext2.img - Detailed root file system in ext2 format, which includes the support for debugging, native compilation, profiling etc. The size of this file is around 110MB.
  - rootfs-LRFS.tar.gz - Compressed file that can be used to make Inflated root file system required for HD boot. As default the USB stick will contain inflated ext3FS on partition 1 and the SD card will contain inflated ext3FS on partition2. This can be used to do a hard disk boot from the USB stick or SD card.
  - rootfs-nfs.tar.gz - Compressed root file system for NFS, which can be used for debug and test at LAB.
  - make_sd.sh - The scripts that can be used to make bootable SD card at a Linux host. The bootable SD card can be used to boot COMX-P2020, also it includes the Linux kernel and root file system.
- u-boot.bin: 4M U-boot Binary file for SPI Flash, which is same as u-boot-spi.bin except for the bigger file size. This file is just for factory to program SPI Flash. Although it can also be used to upgrade SPI Flash as u-boot-spi.bin, Not using it is suggested because it will cost much more time than u-boot-spi.bin.
- COMX-P2020.src.tar.gz: The tar ball of the BSP source code, which can be used to compile all of the above BSP targets.

Host Setup

2.1 Setting Up the Host

Linux OS must be installed at host, and RHEL 5.3-32 bit is suggested (Your system may be different and the commands should be adjusted accordingly).

All the operators on the host side should be ordinary users which have "sudo" privilege with no password. To obtain "sudo" privilege, the operator should login as root, and run "visudo" and add the below line at the end, for example:

```
<username>          ALL=(ALL)          NOPASSWD: ALL
```

Use the following steps to setup the host:

1. Turn off firewall for tftp to work. Type iptables -F or type setup at the command line.
2. Install tftp service.
3. Install nfs service.
4. Create the directory for tftp service, let's assume /local/tftpboot
   ```
   sudo mkdir -p /local/tftpboot
   sudo chmod 777 /local/tftpboot
   ```
5. Edit the tftp service configure file /etc/xinetd.d/tftp and enable the service as below:
   ```
   service tftp
   {
   socket_type             = dgram
   protocol                = udp
   wait                    = yes
   user                    = root
   server                  = /usr/sbin/in.tftpd
   server_args             = -s /local/tftpboot
   disable                 = no
   per_source              = 11
   cps                     = 100 2
   flags                   = IPv4
   }
   [percy@localhost /]$ sudo service xinetd restart
   Stopping xinetd: [  OK  ]
   Starting xinetd: [  OK  ]
   ```
6. Create the directory for NFS service, assuming that /local/tftpboot/COMX-P2020/V100R00.
   ```
   mkdir -p /local/tftpboot/COMX-P2020/V100R00
   ```

7. Create a link for the directory of NFS as the following:
   ```
   cd /local/tftpboot/COMX-P2020
   ln -s V100R00 current
   ```
8. Edit /etc/exports and add the following line at the end:
   ```
   /local/tftpboot/COMX-P2020/current *(rw,sync,no_root_squash)
   ```
9. Start the NFS service as below:
   ```
   [percy@localhost /]$ sudo service nfs restart
   Shutting down NFS mountd: [FAILED]
   Shutting down NFS daemon: [FAILED]
   Shutting down NFS quotas: [FAILED]
   Shutting down NFS services:  [FAILED]
   Starting NFS services:  [  OK  ]
   Starting NFS quotas: [  OK  ]
   Starting NFS daemon: [  OK  ]
   Starting NFS mountd: [  OK  ]
   [percy@localhost /]$
   ```

Get and Install the ToolKit and BSP

3.1 Getting and Installing the ToolKit

The ToolKit is not provided. To get the ToolKit, follow the steps below:

1. It is strongly recommended for BSP users to contact NXP to get the LTIB for P2020RDB (P2020-RDB - LTIB BSP ISO (DVD3)) which can be found at NXP's website *https://www.nxp.com/.* The package includes not only the BSP for P2020RDB, but also the toolkit for process of NXP's P2020. (If the file is deleted from the website, please contact NXP).
2. The installation guide of the toolkit can be found at \help\Documents\pdf\P2020RDB_BSP_UserManual.pdf of the iso file. Please refer to Chapter 2: LTIB Basic in the P2020RDB_BSP_UserManual.pdf to install the toolkit. Only after running ltib, can the toolkit be installed at the directory:/opt/freescale
3. After installing the toolkit, create a soft link for /opt/freescale:

   ```
   [percy@localhost ~]$ cd /opt
   [percy@localhost opt]$ sudo ln -s freescale freescale-p2020
   ```

   or change the directory /opt/freescale to /opt/freescale-p2020:

   ```
   [percy@localhost ~]$ sudo mv /opt/freescale /opt/freescale-p2020
   ```

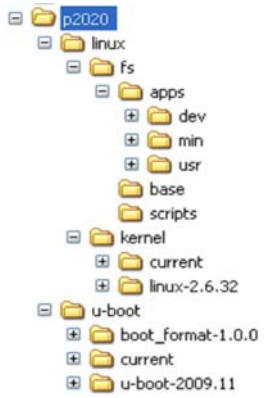
3.2 Getting and Installing the BSP

1. Please contact SMART EC for the tar ball of BSP source file: COMX-2020.src.tar.gz and copy it to the home directory of the user.
2. As a non-root user, uncompress the file COMX-P2020.src.tar.gz.

   ```
   [percy@localhost ~]$ tar zxvf COMX-P2020.src.tar.gz
   ```

3. BSP Source Code Directory Overview
   When the BSP source code has been uncompressed, then the BSP directory tree can be checked as below:

   Figure 3-1BSP Source Code Directory Tree



   - ~/ p2020 is the BSP Directory, which includes all the source code, Makefile and configuration of Linux Kernel, Linux file system and U-boot.
   - ~/p2020/linux/fs is the Linux file system directory, which can be used to compile various file system, such as RAMDISK, Inflated file system tar ball and NFS file system tar ball.
   - ~/p2020/linux/kernel is the Linux kernel directory, which can be used to configure and compile Linux Kernel.
   - ~/u-boot is the U-boot directory, which can be used to compile the U-boot for COMX-P2020.

4. Run "make" at the COMX-P2020 BSP directory to see the help information.

```
 [percy@localhost p2020]$ make
++++++++++++++++++++++++++++++++++++++++++++++++++++++++

 Emerson BlackAdder - P2020 Build

 If you are creating a new release, the appropriate
 version numbers must be updated in the following files:
    versions

++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 Normal Usage Targets:
```

```
  all..................... build all BSP targets
  release................. build release package based on compiled
BSP targets
  clean................... clean all the BSP targets and release
package
++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 Targets: u-boot
  uboot-spi .............. config and build u-boot-spi.bin for SPI
flash
  uboot-sd ............... config and build u-boot-sd.bin for SD
Card
  uboot-clean............. clean the u-boot
++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 Targets: kernel
  config-default......... config the kernel with default
configuration
  kernel-config.......... config the kernel with current
configuration
  kernel................. compile kernel with current
configuration
  kernel-clean............ clean the kernel
++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 Targets: device tree
  dtb..................... compile device tree binary
  dtb-clean............... clean the device tree binary
++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 Targets: rootfs
  rootfs-help............. Help to add specific files and
directories to rootfs
  rootfs-min-ext2......... build min ext2  rootfs
  rootfs-usr-ext2......... build usr ext2  rootfs
  rootfs-dev-ext2......... build dev ext2  rootfs
  rootfs-dev-LRFS......... build dev ininflated ext2FS
  rootfs-nfs.............. build dev rootfs for nfs
  rootfs-clean............ clean the rootfs
++++++++++++++++++++++++++++++++++++++++++++++++++++++++
  current output dir = /home/percy/p2020/.
```

```
  you can set the output dir by modify the variable: OUTPUTDIR
      For example: export OUTPUTDIR=/local/tftpboot/COMX-
P2020/current
```

5. Set the output directory for the compiled BSP targets. If no output directory is set, then the working directory will be regarded as output directory.

```
export OUTPUTDIR=/local/tftpboot/COMX-P2020/current
```

BSP Build Environment

4.1 Build U-boot

4.1.1 About U-boot

The \"U-Boot\" Universal Bootloader project provides firmware with full source code under GPL. Many CPU architectures are supported: PowerPC (MPC5xx, MPC8xx, MPC82xx, MPC7xx, MPC74xx, 4xx), ARM (ARM7, ARM9, StrongARM, Xscale), MIPS (4Kc, 5Kc), x86, etc.

4.1.2 U-boot for COMX-P2020

Since the U-boot has supported a range of CPU architectures including PowerPC (MPC5xx, MPC8xx, MPC82xx, MPC7xx, MPC74xx, 4xx, QorIQ) while P2020 is a base-on-supported-QorIQ processor, it is possible, as well as useful, to provide the U-boot firmware as the COMX-P2020 bootloader.

COMX-P2020 only supports starting-up U-boot from SPI Flash and SD Card.

4.1.3 Memory Map

Table 4-1 Memory Map

Address#	Effective Address	Physical Address	Size	Description
1	0000 0000	0000 0000	2GB	DDR3 Memory
2	8000 0000	8000 0000	512MB	PCIE3 MEM
3	A000 0000	A000 0000	512MB	PCIE2 MEM
4	C000 0000	C000 0000	512MB	PCIE1 MEM
5	FFC0 0000	FFC0 0000	64KB	PCIE3 I/O
6	FFC1 0000	FFC1 0000	64KB	PCIE2 I/O
7	FFC2 0000	FFC2 0000	64KB	PCIE1 I/O
8	FFD0 0000	FFD0 0000	16KB	L1 Data Cache (Only used before memory initialization)
9	FFE0 0000	FFE0 0000	1MB	CSSR

4.1.4 Compile U-boot for SPI Flash

1. Run the command: "make uboot-spi" at the P2020 BSP directory:

```
[percy@localhost p2020]$ make uboot-spi
make -C u-boot -f Makefile uboot-spi
make[1]: Entering directory `/home/percy/p2020/u-boot'
if [ "" != "SPI" ] ; then \
              make config-clean; \
                make -C current -f Makefile
blackadder_SPIFLASH_config; \
              echo "SPI" > uboot.conf ;
……
powerpc-none-linux-gnuspe-objcopy -O srec u-boot u-boot.srec
powerpc-none-linux-gnuspe-objcopy --gap-fill=0xff -O binary u-boot
u-boot.bin
make[2]: Leaving directory `/home/percy/p2020/u-boot/u-boot-
2009.11'
```

```
./boot_format config_sram.dat ./current/u-boot.bin -spi
/local/tftpboot/COMX- p2020/current/u-boot-spi.bin
Write user code to offset 0x400, len = 0x80000, Total reserved space
= 0x80000
Congratulations! It is done successfully.
make[1]: Leaving directory `/home/percy/p2020/u-boot'
[percy@localhost p2020]$
```

2. If the build is successful, the U-boot image for SPI Flash: u-boot-spi.bin can be found at the output directory.

```
 [percy@localhost p2020]$ ls -al /local/tftpboot/COMX-
P2020/current/u-boot-spi.bin
-rw-r--r-- 1 percy percy 525312 Dec 13 11:19 /local/tftpboot/COMX-
P2020/current/u-boot-spi.bin
[percy@localhost p2020]$
```

4.1.5 Compile U-boot for SD Card

1. Run the command: "make uboot-sd" at the P2020 BSP directory:

```
[percy@localhost p2020]$ make uboot-sd
make -C u-boot -f Makefile uboot-sd
make[1]: Entering directory `/home/percy/p2020/u-boot'
if [ "SPI" != "SD" ] ; then \
                make config-clean; \
            make -C current -f Makefile blackadder_SDCARD_config;
\
                echo "SD" > uboot.conf ; \
        fi
make[2]: Entering directory `/home/percy/p2020/
…
powerpc-none-linux-gnuspe-objcopy -O srec u-boot u-boot.srec
powerpc-none-linux-gnuspe-objcopy --gap-fill=0xff -O binary u-boot
u-boot.bin
make[2]: Leaving directory `/home/percy/p2020/u-boot/u-boot-
2009.11'
cp ./current/u-boot.bin /local/tftpboot/COMX-P2020/current/u-
boot-sd.bin
make[1]: Leaving directory `/home/percy/p2020/u-boot'
[percy@localhost p2020]$
```

2. If the build is successful, the U-boot image for SD Card: u-boot-sd.bin can be found at the output directory.

```
[percy@localhost p2020]$ ls -al /local/tftpboot/COMX-P2020/current/u-boot-sd.bin
-rwxr-xr-x 1 percy percy 524288 Dec 13 11:20 /local/tftpboot/COMX-P2020/current/u-boot-sd.bin
[percy@localhost p2020]$
```

4.2 Build Linux Kernel and Device Tree

4.2.1 Configure the Linux kernel with default setting

The COMX-P2020 provides the default configuration file for the linux kernel. To use the default configuration, run the command: "make config-default" at the P2020 BSP directory:

```
 [percy@localhost p2020]$ make config-default
make -C linux/kernel -f Makefile config-default
make[1]: Entering directory `/home/percy/p2020/linux/kernel'
make -C current -f blackadder.mk config-default
make[2]: Entering directory `/home/percy/p2020/linux/kernel/linux-2.6.32'
rm -f /local/tftpboot/COMX-P2020/current/uImage
rm -f include/asm
make -f Makefile mrproper
make[3]: Entering directory `/home/percy/p2020/linux/kernel/linux-2.6.32'
make[3]: Leaving directory `/home/percy/p2020/linux/kernel/linux-2.6.32'
rm -f /local/tftpboot/COMX-P2020/current/comx.dtb
make -f Makefile 85xx/blackadder_defconfig
make[3]: Entering directory `/home/percy/p2020/linux/kernel/linux-2.6.32'
  HOSTCC  scripts/basic/fixdep
  HOSTCC  scripts/basic/docproc
  HOSTCC  scripts/basic/hash
  HOSTCC  scripts/kconfig/conf.o
  HOSTCC  scripts/kconfig/kxgettext.o
  SHIPPED scripts/kconfig/zconf.tab.c
```

```
  SHIPPED scripts/kconfig/lex.zconf.c
  SHIPPED scripts/kconfig/zconf.hash.c
  HOSTCC  scripts/kconfig/zconf.tab.o
  HOSTLD  scripts/kconfig/conf
#
# configuration written to .config
#
make[3]: Leaving directory `/home/percy/p2020/linux/kernel/linux-
2.6.32'
make[2]: Leaving directory `/home/percy/p2020/linux/kernel/linux-
2.6.32'
make[1]: Leaving directory `/home/percy/p2020/linux/kernel'
[percy@localhost p2020]$
```

4.2.2 Configure the Linux Kernel

The COMX-P2020 can work well with default configuration. Also, the user can configure the kernel with the command: "make kernel-config" at the P2020 BSP directory.

This step is not necessary to compile the Linux Kernel; it is just useful for the special requirement on Linux Kernel. Also we strongly suggest to configure the Linux Kernel based on the default COMX-P2020 configuration. (This means run "make config-default" before running "make kernel-config").

```
[percy@localhost p2020]$ make kernel-config
…
.config - Linux Kernel v2.6.32 Configuration
qqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqq
qqqqqqqq
  lqqqqqqqqqqqqqqqqqqqqqqqqq Linux Kernel Configuration
qqqqqqqqqqqqqqqqqqqqqqqqqqk
  x  Arrow keys navigate the menu.  <Enter> selects submenus --->.
Highlighted   x
  x  letters are hotkeys.  Pressing <Y> includes, <N> excludes, <M>
modularizes   x
  x  features.  Press <Esc><Esc> to exit, <?> for Help, </> for Search.
Legend:  x
 x  [*] built-in  [ ] excluded  <M> module  < > module capable
x
```

```
  x
lqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqq
qqqk x
  x x    [ ] 64-bit kernel                                                 x x
  x x        Processor support  --->                                       x x
  x x        General setup  --->                                           x x
  x x    [*] Enable loadable module support  --->                          x x
  x x    [*] Enable the block layer  --->                                  x x
  x x        Platform support  --->                                        x x
  x x        Kernel options  --->                                          x x
  x x        Bus options  --->                                             x x
  x x        Advanced setup  --->                                          x x
  x x    [*] Networking support  --->                                      x x
  x x        Device Drivers  --->                                          x x
  x x        File systems  --->                                            x x
  x x        Library routines  --->                                        x x
  x x        Kernel hacking  --->                                          x x
  x x        Security options  --->                                        x x
  x x    -*- Cryptographic API  --->                                       x x
  x x    [ ] Virtualization  --->                                          x x
  x x    ---                                                               x x
  x x        Load an Alternate Configuration File                          x x
  x x        Save an Alternate Configuration File                          x x
  x x                                                                      x x
  x x                                                                      x x
  x x                                                                      x x
  x
mqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqq
qqqj x

tqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqq
qqqqqqqu
  x                    <Select>    < Exit >    < Help >                      x

mqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqq
qqqqqqqj
```

Once the user changed the configuration of Linux kernel, select YES to save the configuration, or the modification will be discarded.

Figure 4-1 Save Linux Kernel Configuration

4.2.3 Compile the Linux Kernel

1. After configuring the Linux Kernel, run the command: "make kernel" to compile Linux Kernel at the P2020 BSP directory:

```
 [percy@localhost p2020]$ make kernel
make -C linux/kernel -f Makefile kernel
make[1]: Entering directory `/home/percy/p2020/linux/kernel'
make -C current -f blackadder.mk kernel
make[2]: Entering directory `/home/percy/p2020/linux/kernel/linux-
2.6.32'
make -f Makefile uImage
make[3]: Entering directory `/home/percy/p2020/linux/kernel/linux-
2.6.32'
scripts/kconfig/conf -s arch/powerpc/Kconfig
make[3]: Leaving directory `/home/percy/p2020/linux/kernel/linux-
2.6.32'
make[3]: Entering directory `/home/percy/p2020/linux/kernel/linux-
2.6.32'
  CHK     include/linux/version.h
……

Image Name:   Linux-2.6.32
Created:      Mon Dec 13 11:33:16 2010
Image Type:   PowerPC Linux Kernel Image (gzip compressed)
Data Size:    3416064 Bytes = 3336.00 kB = 3.26 MB
Load Address: 0x00000000
Entry Point:  0x00000000
make[3]: Leaving directory `/home/percy/p2020/linux/kernel/linux-
2.6.32'
```

```
cp -f arch/powerpc/boot/uImage /local/tftpboot/COMX-
P2020/current/uImage
make[2]: Leaving directory `/home/percy/p2020/linux/kernel/linux-
2.6.32'
make[1]: Leaving directory `/home/percy/p2020/linux/kernel'
[percy@localhost p2020]$
```

2. If the build is successful, the Linux kernel: uImage can be found at the output directory.

```
[percy@localhost p2020]$ ls -al /local/tftpboot/COMX-
P2020/current/uImage
-rw-rw-r-- 1 percy percy 3416128 Dec 13 11:33 /local/tftpboot/COMX-
P2020/current/uImage
```

4.2.4 Compile Device Tree Bolb

1. To compile the device tree bolb, run the command: "make dtb" at the P2020 BSP directory:

```
[percy@localhost p2020]$ make dtb
make -C linux/kernel -f Makefile dtb
make[1]: Entering directory `/home/percy/p2020/linux/kernel'
make -C current -f blackadder.mk dtb-clean
make[2]: Entering directory `/home/percy/p2020/linux/kernel/linux-
2.6.32'
rm -f /local/tftpboot/COMX-P2020/current/comx.dtb
make[2]: Leaving directory `/home/percy/p2020/linux/kernel/linux-
2.6.32'
make -C current -f blackadder.mk dtb
make[2]: Entering directory `/home/percy/p2020/linux/kernel/linux-
2.6.32'
rm -f /local/tftpboot/COMX-P2020/current/comx.dtb
dtc -f -b 0 -I dts -O dtb -p 1024 -o /local/tftpboot/COMX-
P2020/current/comx.dtb arch/powerpc/boot/dts/blackadder.dts
DTC: dts->dtb  on file "arch/powerpc/boot/dts/blackadder.dts"
make[2]: Leaving directory `/home/percy/p2020/linux/kernel/linux-
2.6.32'
make[1]: Leaving directory `/home/percy/p2020/linux/kernel'
[percy@localhost p2020]$
```

2. If the build is successful, the device tree bolb: comx.dtb can be found at the output directory.

```
[percy@localhost p2020]$ ls -al /local/tftpboot/COMX-P2020/current/comx.dtb
-rw-rw-r-- 1 percy percy 9824 Dec 13 11:35 /local/tftpboot/COMX-P2020/current/comx.dtb
[percy@localhost p2020]$
```

4.3 Build Root File System

There are 5 types of root file systems that can be supported for COMX-P2020.

- rootfs-min.ext2.img - The minimum root file system in ext2 format, which is helpful in case of deployment on Flash devices which are of smaller sizes. The size of this file is around 4MB.
- rootfs-usr.ext2.img - The Medium root file system in ext2 format, which is good enough for deployment perspective and supports most of the use cases identified for Residential Gateway, Wireless LAN Access Point, Media Server, NAS box, etc. The size of this file is around 42MB.
- rootfs-dev.ext2.img - Detailed root file system in ext2 format, which includes the support for debugging, native compilation, profiling etc. The size of this file is around110MB.
- rootfs-LRFS.tar.gz - Compressed file that can be used to make Inflated root file system required for HD boot. As default the USB stick will contain inflated ext3FS on partition 1 and the SD card will contain inflated ext3FS on partition2. This can be used to do a hard disk boot from the USB stick or SD card.
- rootfs-nfs.tar.gz - Compressed root file system for NFS, which can be used for debug and test at LAB.

4.3.1 Build rootfs-min-ext2.img

1. To build rootfs-min-ext2.img, run the command: "make rootfs-min-ext2" at P2020 BSP directory:

```
[percy@localhost p2020]$ make rootfs-min-ext2
make -C linux/fs -f Makefile rootfs-min-ext2
make[1]: Entering directory `/home/percy/p2020/linux/fs'
rm -f /tmp/percy/rootfs_ext2_min.img /local/tftpboot/COMX-P2020/current/rootfs-min.ext2.img
./scripts/gen_rootfs_ext2.sh min
```

```
……

+ cd /tmp/percy
+ echo -n 'Generating rootfs_min.gz in /tmp/percy ... '
Generating rootfs_min.gz in /tmp/percy ... + sudo genext2fs -U -b
14348 -i 1118 -D
/home/percy/p2020/linux/fs/scripts/../base/device_table_min.txt -
d rootfs_min rootfs_ext2_min
+ sudo gzip rootfs_ext2_min
+ echo done.
done.
+ echo 'Generating rootfs_min.img in /tmp/percy ... '
Generating rootfs_min.img in /tmp/percy ...
+ sudo mkimage -n 'Blackadder ext2 ramdisk rootfs' -A ppc -O linux
-T ramdisk -C gzip -d rootfs_ext2_min.gz rootfs_ext2_min.img
Image Name:   Blackadder ext2 ramdisk rootfs
Created:      Mon Dec 13 11:39:01 2010
Image Type:   PowerPC Linux RAMDisk Image (gzip compressed)
Data Size:    3873919 Bytes = 3783.12 kB = 3.69 MB
Load Address: 0x00000000
Entry Point:  0x00000000
+ echo done.
done.
cp   /tmp/percy/rootfs_ext2_min.img /local/tftpboot/COMX-
P2020/current/rootfs-min.ext2.img
make[1]: Leaving directory `/home/percy/p2020/linux/fs'
[percy@localhost p2020]$
```

2. If successful, the rootfs-min-ext2.img can be found at the output directory:

```
 [percy@localhost p2020]$ ls -al /local/tftpboot/COMX-
P2020/current/rootfs-min.ext2.img
-rw-r--r-- 1 percy percy 3873983 Dec 13 11:39 /local/tftpboot/COMX-
P2020/current/rootfs-min.ext2.img
[percy@localhost p2020]$
```

4.3.2 Build rootfs-usr-ext2.img

1. To build rootfs-usr-ext2.img, run the command: "make rootfs-usr-ext2" at P2020 BSP directory:

```
 [percy@localhost P2020]$ make rootfs-usr-ext2
make -C linux/fs -f Makefile rootfs-usr-ext2
make[1]: Entering directory `/home/percy/p2020/linux/fs'
rm -f /tmp/percy/rootfs_ext2_usr.img /local/tftpboot/COMX-
P2020/current/rootfs-usr.ext2.img
./scripts/gen_rootfs_ext2.sh usr
……


+ cd /tmp/percy
+ echo -n 'Generating rootfs_usr.gz in /tmp/percy ... '
Generating rootfs_usr.gz in /tmp/percy ... + sudo genext2fs -U -b
165628 -i 6881 -D
/home/percy/p2020/linux/fs/scripts/../base/device_table_min.txt -
d rootfs_usr rootfs_ext2_usr
+ sudo gzip rootfs_ext2_usr
+ echo done.
done.
+ echo 'Generating rootfs_usr.img in /tmp/percy ... '
Generating rootfs_usr.img in /tmp/percy ...
+ sudo mkimage -n 'Blackadder ext2 ramdisk rootfs' -A ppc -O linux
-T ramdisk -C gzip -d rootfs_ext2_usr.gz rootfs_ext2_usr.img
Image Name:   Blackadder ext2 ramdisk rootfs
Created:      Mon Dec 13 11:41:50 2010
Image Type:   PowerPC Linux RAMDisk Image (gzip compressed)
Data Size:    42143653 Bytes = 41155.91 kB = 40.19 MB
Load Address: 0x00000000
Entry Point:  0x00000000
+ echo done.
done.
cp   /tmp/percy/rootfs_ext2_usr.img /local/tftpboot/COMX-
P2020/current/rootfs-usr.ext2.img
make[1]: Leaving directory `/home/percy/p2020/linux/fs'
[percy@localhost p2020]$
```

2. If successful, the rootfs-usr-ext2.img can be found at the output directory:

```
 [percy@localhost p2020]$ ls -al /local/tftpboot/COMX-
P2020/current/rootfs-usr.ext2.img
-rw-r--r-- 1 percy percy 42143717 Dec 13 11:41
/local/tftpboot/COMX-P2020/current/rootfs-usr.ext2.img
[percy@localhost p2020]$
```

4.3.3 Build rootfs-dev-ext2.img

1. To build rootfs-dev-ext2.img, run the command: "make rootfs-dev-ext2" at P2020 BSP directory:

```
[percy@localhost p2020]$ make rootfs-dev-ext2
make -C linux/fs -f Makefile rootfs-dev-ext2
make[1]: Entering directory `/home/percy/p2020/linux/fs'
rm -f /tmp/percy/rootfs_ext2_dev.img /local/tftpboot/COMX-
P2020/current/rootfs-dev.ext2.img
./scripts/gen_rootfs_ext2.sh dev
+ USAGE='Usage: ./scripts/gen_rootfs_ext2.sh <min/usr/dev>'
……

+ cd /tmp/percy
+ echo -n 'Generating rootfs_dev.gz in /tmp/percy ... '
Generating rootfs_dev.gz in /tmp/percy ... + sudo genext2fs -U -b
392568 -i 16966 -D
/home/percy/p2020/linux/fs/scripts/../base/device_table_min.txt -
d rootfs_dev rootfs_ext2_dev
+ sudo gzip rootfs_ext2_dev
+ echo done.
done.
+ echo 'Generating rootfs_dev.img in /tmp/percy ... '
Generating rootfs_dev.img in /tmp/percy ...
+ sudo mkimage -n 'Blackadder ext2 ramdisk rootfs' -A ppc -O linux
-T ramdisk -C gzip -d rootfs_ext2_dev.gz rootfs_ext2_dev.img
Image Name:   Blackadder ext2 ramdisk rootfs
Created:      Mon Dec 13 11:45:47 2010
Image Type:   PowerPC Linux RAMDisk Image (gzip compressed)
Data Size:    110295687 Bytes = 107710.63 kB = 105.19 MB
```

```
Load Address: 0x00000000
Entry Point:  0x00000000
+ echo done.
done.
cp   /tmp/percy/rootfs_ext2_dev.img /local/tftpboot/COMX-
P2020/current/rootfs-dev.ext2.img
make[1]: Leaving directory `/home/percy/p2020/linux/fs'
[percy@localhost p2020]$
```

2. If successful, the rootfs-dev-ext2.img can be found at the output directory:

```
[percy@localhost p2020]$ ls -al /local/tftpboot/COMX-
P2020/current/rootfs-dev.ext2.img
-rw-r--r-- 1 percy percy 110295751 Dec 13 11:45
/local/tftpboot/COMX-P2020/current/rootfs-dev.ext2.img
[percy@localhost p2020]$
```

4.3.4 Build rootfs-LRFS.tar.gz

1. To build rootfs-LRFS.tar.gz, run the command: "make rootfs-dev-LRFS" at p2020 BSP directory:

```
[percy@localhost p2020]$ make rootfs-dev-LRFS
make -C linux/fs -f Makefile rootfs-dev-LRFS
make[1]: Entering directory `/home/percy/p2020/linux/fs'
rm -f /tmp/percy/rootfs_ext2_dev_LRFS.tar.gz /local/tftpboot/COMX-
P2020/current/rootfs-LRFS.tar.gz
./scripts/gen_rootfs_LRFS.sh dev
+ USAGE='Usage: ./scripts/gen_rootfs_LRFS.sh <min/usr/dev>'
+ '[' 1 -ne 1 ']'
……


+ echo done.
done.
cp   /tmp/percy/rootfs_ext2_dev_LRFS.tar.gz /local/tftpboot/COMX-
P2020/current/rootfs-LRFS.tar.gz
make[1]: Leaving directory `/home/percy/p2020/linux/fs'
[percy@localhost p2020]$
```

2. If successful, the rootfs-LRFS.tar.gz can be found at the output directory:

```
[percy@localhost p2020]$ ls -al /local/tftpboot/COMX-
P2020/current/rootfs-LRFS.tar.gz
-rw-r--r-- 1 percy percy 107445442 Dec 13 11:48
/local/tftpboot/COMX-P2020/current/rootfs-LRFS.tar.gz
```

4.3.5 Build rootfs-nfs.tar.gz

1. To build rootfs-nfs.tar.gz, run the command: "make rootfs-nfs" at P2020 BSP directory:

```
[percy@localhost p2020]$ make rootfs-nfs
make -C linux/fs -f Makefile rootfs-dev-nfs
make[1]: Entering directory `/home/percy/p2020/linux/fs'
rm -f /tmp/percy/rootfs_nfs.tar.gz /local/tftpboot/COMX-
P2020/current/rootfs-nfs.tar.gz
./scripts/gen_rootfs_nfs.sh dev
……
+ sudo cp -a
/home/percy/p2020/linux/fs/scripts/../apps/dev/nfs/../all/etc
/home/percy/p2020/linux/fs/scripts/../apps/dev/nfs/../all/lib
/home/percy/p2020/linux/fs/scripts/../apps/dev/nfs/../all/sbin
/home/percy/p2020/linux/fs/scripts/../apps/dev/nfs/../all/usr
rootfs_dev/
+ rm -rf /home/percy/p2020/linux/fs/scripts/../apps/dev/nfs/../all
+ sudo cp -a
/home/percy/p2020/linux/fs/scripts/../apps/dev/nfs/etc
/home/percy/p2020/linux/fs/scripts/../apps/dev/nfs/usr
rootfs_dev/
+ '[' -V100R00 '!=' '' ']'
+ rm -rf
/home/percy/p2020/linux/fs/scripts/../apps/dev/nfs/etc/.version
+ echo -n 'Generating rootfs_dev.tar.gz in /tmp/percy ... '
Generating rootfs_dev.tar.gz in /tmp/percy ... + sudo mv rootfs_dev
rootfs_nfs
+ sudo tar zcvf rootfs_nfs.tar.gz rootfs_nfs
+ echo done.
done.
cp   /tmp/percy/rootfs_nfs.tar.gz /local/tftpboot/COMX-
P2020/current/rootfs-nfs.tar.gz
make[1]: Leaving directory `/home/percy/p2020/linux/fs'
```

```
[percy@localhost p2020]$
```

2. If successful, the rootfs-nfs.tar.gz can be found at the output directory:

```
[percy@localhost p2020]$ ls -al /local/tftpboot/COMX-
P2020/current/rootfs-nfs.tar.gz
-rw-r--r-- 1 percy percy 200279316 Dec 13 11:50
/local/tftpboot/COMX-P2020/current/rootfs-nfs.tar.gz
[percy@localhost p2020]$
```

4.3.6 Build release package for COMX-P2020

1. The release package of COMX-P2020 includes not only COMX-P2020.bsp.tar.gz, but also the 4M U-boot Image for SPI flash which will be used at factory to program the SPI flash. Before building the release package, make sure that all the above BSP targets should be build in advance.

```
[percy@localhost p2020]$ make release
./make_spi_image.sh /local/tftpboot/COMX-P2020/current
4+0 records in
4+0 records out
4194304 bytes (4.2 MB) copied, 0.0185468 seconds, 226 MB/s
1026+0 records in
1026+0 records out
525312 bytes (525 kB) copied, 0.0113922 seconds, 46.1 MB/s
/local/tftpboot/COMX-P2020/current/u-boot.bin create OK.
./make_bsp_package.sh /local/tftpboot/COMX-P2020/current
cp: cannot create regular file `/local/tftpboot/COMX-
P2020/current/./make_sd.sh': Permission denied
make_sd.sh
u-boot-spi.bin
u-boot-sd.bin
comx.dtb
uImage
rootfs-LRFS.tar.gz
rootfs-nfs.tar.gz
rootfs-dev.ext2.img
make: *** [release] Interrupt

[percy@localhost p2020]$ make release
```

```
./make_spi_image.sh /local/tftpboot/COMX-P2020/current
4+0 records in
4+0 records out
4194304 bytes (4.2 MB) copied, 0.018337 seconds, 229 MB/s
1026+0 records in
1026+0 records out
525312 bytes (525 kB) copied, 0.00351805 seconds, 149 MB/s
/local/tftpboot/COMX-P2020/current/u-boot.bin create OK.
./make_bsp_package.sh /local/tftpboot/COMX-P2020/current
make_sd.sh
u-boot-spi.bin
u-boot-sd.bin
comx.dtb
uImage
rootfs-LRFS.tar.gz
rootfs-nfs.tar.gz
rootfs-dev.ext2.img
rootfs-usr.ext2.img
rootfs-min.ext2.img
The following files should be created at: /local/tftpboot/COMX-
P2020/current
ls -al /local/tftpboot/COMX-P2020/current/u-boot.bin
/local/tftpboot/COMX-P2020/current/COMX-P2020.bsp.tar.gz
-rw-rw-r-- 1 percy percy 462424461 Dec 13 12:12
/local/tftpboot/COMX-P2020/current/COMX-P2020.bsp.tar.gz
-rw-rw-r-- 1 percy percy   4194304 Dec 13 12:12
/local/tftpboot/COMX-P2020/current/u-boot.bin
[percy@localhost p2020]$
```

2. If successful, the COMX-P2020.bsp.tar.gz and u-boot.bin can be found at the output directory:

```
 [percy@localhost p2020]$ ls -al  /local/tftpboot/COMX-
P2020/current/*
-rw-rw-r-- 1 percy percy 462424461 Dec 13 12:12
/local/tftpboot/COMX-P2020/current/COMX-P2020.bsp.tar.gz
-rw-rw-r-- 1 percy percy   4194304 Dec 13 12:12
/local/tftpboot/COMX-P2020/current/u-boot.bin
[percy@localhost p2020]$
```

U-boot Deployment

5.1 Switch Setting on COMX-P2020

Check and set the switch as below before powering on the board:
S3_14 = OFF, S3_23 = OFF, S7_14 = OFF, S7_23 = OFF,
S2_23 = ON, S2_14 = OFF, S1_23 = OFF, S1_14 = OFF.
The booting configuration is listed below. By default, the board will boot from the SD card.

Table 5-1 Content for ID EEPROM

S2_23	S2_14	S1_23	S1_14	Description
ON	OFF	OFF	ON	Boot from SPI
ON	OFF	OFF	OFF	Boot form SDHC (Default)

5.2 Host Setup

1. Connect the Host and the first network port of COMX-P2020 to the network switch, and set the IP Address on the Host. Here we set the Host IP Address: 192.168.0.197 with network mask 255.255.255.0.
2. Connect the UART#1 of COMX-P2020 to the host via a cross cable serial connection.
3. Start a terminal such as minicom and set it up to talk to the COMX-P2020 board.
4. Serial Setup:
   Baud rate= 115200; Data bits = 8; Parity = None; Stop bits = 1; Flow Control = None
5. Power on the board and check the console prompt.
6. Extract the BSP targets from the compressed BSP tar ball as below:

```
[percy@localhost current]$ pwd/local/tftpboot/COMX-P2020/current
[percy@localhost current]$ tar zxvf COMX-P2020.bsp.tar.gz
make_sd.sh
u-boot-spi.bin
u-boot-sd.bin
comx.dtb
uImage
rootfs-LRFS.tar.gz
rootfs-nfs.tar.gz
rootfs-dev.ext2.img
rootfs-usr.ext2.img
rootfs-min.ext2.img
[percy@localhost current]$
```

7. Extract the NFS root file system from the compressed NFS tar ball as below:

```
[percy@localhost current]$ sudo tar zxvf rootfs-nfs.tar.gz
[percy@localhost current]$ ls -al rootfs_nfs
total 80
drwxrwxr-x 20 percy percy 4096 Dec  9 14:55 .
drwxr-xr-x  3 percy percy 4096 Dec 14 11:23 ..
drwxr-xr-x  2  5117  5000 4096 Nov 23 17:30 bin
drwxr-xr-x  2  5117  5000 4096 Nov 23 10:57 boot_format
drwxr-xr-x  2 root  root  4096 Apr 29  2010 dev
drwxrwxrwx 13  5101  5000 4096 Dec  9 14:54 etc
drwxr-xr-x  3 root  root  4096 Nov 23 11:53 home
drwxr-xr-x 10 root  root  4096 Nov 23 11:54 include
drwxr-xr-x  5  5117  5000 4096 Oct 30 07:09 lib
lrwxrwxrwx  1 root  root    11 Dec 14 11:23 linuxrc -> bin/busybox
drwxr-xr-x  6 root  root  4096 Apr 29  2010 man
drwxr-xr-x  7 root  root  4096 Nov 23 11:53 mnt
drwxr-xr-x  3 root  root  4096 Nov 23 11:53 opt
drwxr-xr-x  2 root  root  4096 Apr 29  2010 proc
drwxr-xr-x  3 root  root  4096 Nov 23 11:54 root
drwxr-xr-x  2  5117  5000 4096 Nov 23 14:44 sbin
drwxr-xr-x  3 root  root  4096 Nov 23 11:54 share
drwxr-xr-x  2 root  root  4096 Apr 29  2010 sys
drwxrwxrwt  3 root  root  4096 Apr 29  2010 tmp
drwxrwxrwx 15  5101  5000 4096 Jun 25 11:51 usr
drwxr-xr-x 11 root  root  4096 Apr 29  2010 var
[percy@localhost current]$
```

5.3 Start-up U-boot

5.3.1 Program SD Card

By default, the SD card includes the U-boot, Linux Kernel and root file system which can be used to start-up the COMX-P2020. If the SD card is damaged, we can program it at the Linux host with a USB reader as below:

- Create the /local/tftpboot/COMX-P2020/current directory at host, and copy the release file: COMX-P2020.bsp.tar.gz to the directory: /local/tftpboot/COMX-P2020/current

```
[percy@localhost current]$ sudo chmod 777 /local/tftpboot/COMX-
P2020/current
[percy@localhost current]$ ls -al COMX-P2020.bsp.tar.gz
-rwxr--r-- 1 percy percy 462501957 Dec  9 14:57 COMX-
P2020.bsp.tar.gz
```

- Extract all the BSP targets from the COMX-P2020.bsp.tar.gz: `[percy@localhost current]$ tar zxvf COMX-P2020.bsp.tar.gz`

```
[percy@localhost current]$ ls -al
total 910224
drwxr-xr-x 2 percy percy      4096 Dec 10 09:37 .
drwxrwxr-x 8 percy percy      4096 Dec  9 18:07 ..
-rw-r--r-- 1 percy percy      9824 Dec  9 14:51 comx.dtb
-rwxr--r-- 1 percy percy 462501957 Dec  9 14:57 COMX-
P2020.bsp.tar.gz
-r-xr-xr-x 1 percy percy      5771 Dec  9 12:57 make_sd.sh
-rw-r--r-- 1 percy percy 110443332 Dec  9 14:53 rootfs-dev.ext2.img
-rw-r--r-- 1 percy percy 107400408 Dec  9 14:54 rootfs-LRFS.tar.gz
-rw-r--r-- 1 percy percy   3872536 Dec  9 14:51 rootfs-min.ext2.img
-rw-r--r-- 1 percy percy 200253398 Dec  9 14:55 rootfs-nfs.tar.gz
-rw-r--r-- 1 percy percy  42137827 Dec  9 14:51 rootfs-usr.ext2.img
-rwxr-xr-x 1 percy percy    524288 Dec  9 14:40 u-boot-sd.bin
-rw-r--r-- 1 percy percy    525312 Dec  9 14:39 u-boot-spi.bin
-rw-r--r-- 1 percy percy   3416506 Dec  9 14:51 uImage
```

- Insert the MicroSD to a USB reader, and connect the USB reader to the host.
- Check the device name for the USB reader. The following command shows that the device name for USB reader is /dev/sdb.

```
[percy@localhost current]$ ls -al /dev/sd*
brw-r----- 1 root disk 8,  0 Dec  2 17:16 /dev/sda
brw-r----- 1 root disk 8,  1 Dec  2 17:17 /dev/sda1
brw-r----- 1 root disk 8,  2 Dec  2 17:16 /dev/sda2
brw-r----- 1 root disk 8, 16 Dec  2 18:43 /dev/sdb
[percy@localhost current]$ sudo fdisk -l /dev/sdb

Disk /dev/sdb: 2002 MB, 2002780160 bytes
62 heads, 62 sectors/track, 1017 cylinders
Units = cylinders of 3844 * 512 = 1968128 bytes

Device Boot      Start         End      Blocks   Id  System
```

- Run the script make_sd.sh to program the SD card for COMX-P2020 as below:
  `[percy@localhost COMX-P2020]$ sudo ./make_sd.sh /dev/sdb`

  Normally the program process will last for 3-5 minutes. If the SD card is upgraded successfully, the following information can be shown as below:

  ```
  Program SD successfully, first partition size = 300 MByte,
  cost time: 198 seconds
  ```

  Otherwise, the SD card program has failed.

- The second parameter of the script: make_sd.sh can be used to change the size of first partition. To change the size of the first partition to 250M, run the script as below:
- The second parameter of the script: make_sd.sh can be used to change the size of first partition. To change the size of the first partition to 250M, you can run the script as below:

  The second parameter of the script: make_sd.sh can be used to change the size of first partition. To change the size of the first partition to 250M, you can run the script as below:

  ```
  [percy@localhost COMX-P2020]$ sudo ./make_sd.sh /dev/sdb 250
  ……
  Program SD successfully, first partition size = 250 MByte,
  cost time: 171 seconds
  ```

  The size of the first partition should be greater than 160M, and less than 2000M, or the script will show input parameter error.

The P2020 Version1.x cannot support booting from the SD Card, only P2020 Version2.0 support booting from SD Card.

The CPU version can be found at the output of UART0 when booting from U-Boot:

```
U-Boot 2009.11-V100B09 (Sep 24 2010 - 17:19:44)
CPU0:  P2020E, Version: 2.0, (0x80ea0020)
Core:  E500, Version: 5.0, (0x80211050)
Clock Configuration:
```

To understand the structure of SD card, please refer the chapter: LINK deploy filesystem to SD Card.

5.3.2 Start-up U-boot from SD Card

Set the switch as: SW1[1] = OFF, SW1[2] = OFF, SW2[1] = OFF, SW2[2] = ON, and power on the board.

Normally, you can see the following information from the serial terminal as below:

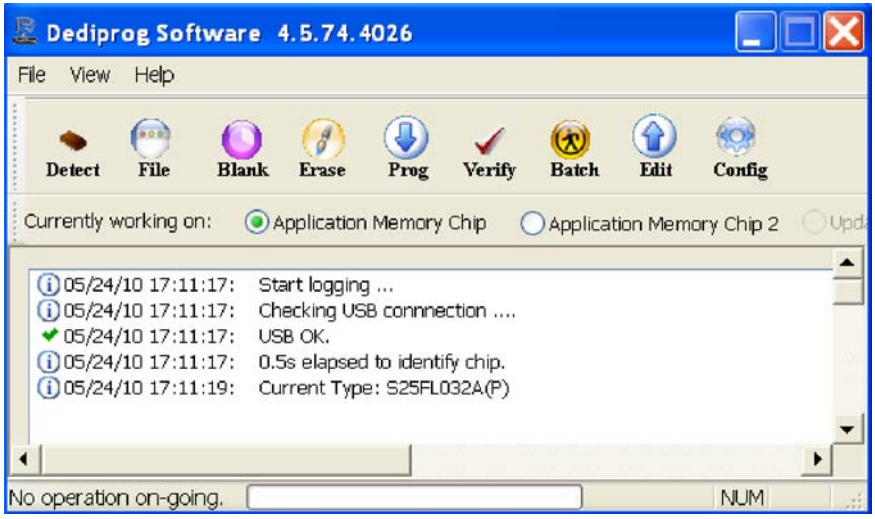
```
U-Boot 2009.11-V100R00 (Dec 09 2010 - 14:40:51)


CPU0:  P2020E, Version: 2.0, (0x80ea0020)
Core:  E500, Version: 5.0, (0x80211050)
Clock Configuration:
       CPU0:1200 MHz, CPU1:1200 MHz,
       CCB:600  MHz,
      DDR:333.333 MHz (666.667 MT/s data rate) (Asynchronous), LBC:37.500
MHz
L1:    D-cache 32 kB enabled
       I-cache 32 kB enabled
I2C:   ready
SPI:   ready
DRAM:   2 GB
L2:    512 KB enabled
MMC:  FSL_ESDHC: 0
EEPROM: NXID v0
EEPROM: COMX
    PCIE3 connected to Slot0 as Root Complex (base addr ffe08000)
    PCIE3 on bus 00 - 00
    PCIE2 connected to Slot 1 as Root Complex (base addr ffe09000)
    PCIE2 on bus 01 - 01
    PCIE1 connected to Slot 2 as Root Complex (base addr ffe0a000)
    Current Status: LSR-11, LTSSM-16, PEX width-x1, Clock-2.5GT/s
              Scanning PCI bus 03
        03  00  18ca  0027  0300  00
    PCIE1 on bus 02 - 03
In:    serial
Out:   serial
Err:   serial
Net:   eTSEC1, eTSEC2, eTSEC3
Hit any key to stop autoboot:  0
=>
```

5.3.3 Program SPI Flash with SF100

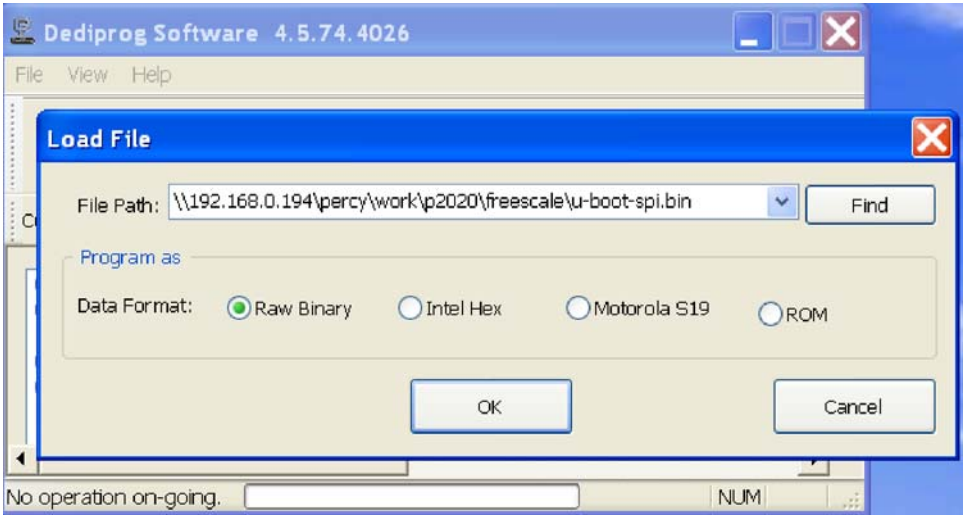
1. Install the driver and application of the SF100 at the Windows OS.
2. Connect the SF100 to the USB port and insert the SPI Flash into the socket of SF100. The COMX-P2020 support the SPI Flash: S25FL032A(P)

3. Run DediPro.exe:



If the driver and application for SF100 installs successfully, the Dediprog can find the SPI Flash: S25FL032A(P) as above.

4. Load file by clicking the File and select the u-boot-spi.bin.Select "Raw Binary".



5. Program the SPI Flash by click the toolbar: Batch, Then the Dediprog will erase, program and verify the SPI Flash automatically.

5.3.4 Set the Switch

Set the switch as: SW1[1] = OFF, SW1[2] = ON, SW2[1] = ON, SW2[2] = OFF, and power on the board.

Normally, you can see the following information from the serial terminal as below:

```
U-Boot 2009.11-V100R00 (Dec 09 2010 - 14:39:42)

CPU0:  P2020E, Version: 2.0, (0x80ea0020)
Core:  E500, Version: 5.0, (0x80211050)
Clock Configuration:
       CPU0:1200 MHz, CPU1:1200 MHz,
       CCB:600  MHz,
     DDR:333.333 MHz (666.667 MT/s data rate) (Asynchronous), LBC:37.500
MHz
L1:    D-cache 32 kB enabled
       I-cache 32 kB enabled
I2C:   ready
SPI:   ready
DRAM:   2 GB
```

```
L2:    512 KB enabled
MMC:  FSL_ESDHC: 0
EEPROM: NXID v0
EEPROM: COMX

    PCIE3 connected to Slot0 as Root Complex (base addr ffe08000)
    PCIE3 on bus 00 - 00

    PCIE2 connected to Slot 1 as Root Complex (base addr ffe09000)
    PCIE2 on bus 01 - 01

    PCIE1 connected to Slot 2 as Root Complex (base addr ffe0a000)
    Current Status: LSR-11, LTSSM-16, PEX width-x1, Clock-2.5GT/s
              Scanning PCI bus 03
        03  00  18ca  0027  0300  00
    PCIE1 on bus 02 - 03

In:    serial
Out:   serial
Err:   serial
Net:   eTSEC1, eTSEC2, eTSEC3
Hit any key to stop autoboot:  0
=>
```

5.4 U-boot Command

To see a list of the available U-Boot commands, type "help" or "?". This will display a list of all commands that are available in your current configuration. Proceed as follows:

```
=> help
?       - alias for 'help'
at24c02 - read/write at24c02
base    - print or set address offset
bdinfo  - print Board Info structure
boot    - boot default, i.e., run 'bootcmd'
bootd   - boot default, i.e., run 'bootcmd'
bootelf - Boot from an ELF image in memory
```

```
bootm   - boot application image from memory
bootp   - boot image via network using BOOTP/TFTP protocol
bootvx  - Boot vxWorks from an ELF image
cmp     - memory compare
coninfo - print console devices and information
cp      - memory copy
cpu     - Multiprocessor CPU boot manipulation and release
crc32   - checksum calculation
date    - get/set/reset date & time
echo    - echo args to console
editenv - edit environment variable
exit    - exit script
ext2load- load binary file from a Ext2 filesystem
ext2ls  - list files in a directory (default /)
fatinfo - print information about filesystem
fatload - load binary file from a dos filesystem
fatls   - list files in a directory (default /)
fdt     - flattened device tree utility commands
func    - execute func
go      - start application at address 'addr'
gpio    - gpio operation
help    - print online help
i2c     - I2C sub-system
i2cswitch- connect i2c bus to i2c switch
iic     - iic read and write
iminfo  - print header information for application image
imxtract- extract a part of a multi-image
interrupts- enable or disable interrupts
irqinfo - print information about IRQs
itest   - return true/false on integer compare
loadb   - load binary file over serial line (kermit mode)
loads   - load S-Record file over serial line
loady   - load binary file over serial line (ymodem mode)
loop    - infinite loop on address range
mac     - display and program the system ID and MAC addresses in EEPROM
md      - memory display
```

```
mii     - MII utility commands
mm      - memory modify (auto-incrementing address)
mmc     - MMC sub system
mmcinfo - mmcinfo <dev num>-- display MMC info

mtest   - simple RAM read/write test
mw      - memory write (fill)
nfs     - boot image via network using NFS protocol
nm      - memory modify (constant address)
pci     - list and access PCI Configuration Space
ping    - send ICMP ECHO_REQUEST to network host
printenv- print environment variables
rarpboot- boot image via network using RARP/TFTP protocol
reset   - Perform RESET of the CPU
run     - run commands in an environment variable
saveenv - save environment variables to persistent storage
sensor  - LM75 Sensor
setenv  - set environment variables
setexpr - set environment variable as the result of eval expression
sf      - SPI flash sub-system
showvar - print local hushshell variables
sleep   - delay execution for some time
source  - run script from memory
test    - minimal test like /bin/sh
tftpboot- boot image via network using TFTP protocol
usb     - USB sub-system
usbboot - boot from USB device
version - print monitor version
wdg     - setting watchdog
=>
```

Most of the displayed commands were inherited from U-boot.

5.4.1 UART

The P2020 consists of two universal asynchronous receiver/transmitters (UARTs). The UARTs act independently; The UART#1 acts as the standard input and output device at U-boot; the UART#2 is not used at U-boot, but the command: func testuart1 can be used to test UART#2 at U-boot:

1. Connect the UART#2 to another serial port of PC;
2. Connect another terminal to the board's serial console port UART#2 with a baudrate of 115200 8n1n.
3. Run the command at the terminal to UART#1:

   ```
   => func testuart1
   Please test at UART1, press 'q' to quit:
   ```

4. Enter any character at the terminal to UART#2. You can see the character you entered at the terminal, which means that the input to UART#2 can be outputted to UART#2.
5. Press 'q' at the terminal to UART#2, the UART#2 test will end successfully.

5.4.2 GPIO

There are a total of 16 GPIO used at COMX-P2020.

Table 5-2 GPIO Description

Name	Intput/ Output	Reset Value	Description
GPIO 0	Input		Connected to the COM-E Carrier Board
GPIO 1	Input		Connected to the COM-E Carrier Board
GPIO 2	Input		Connected to the COM-E Carrier Board
GPIO 3	Input		Connected to the COM-E Carrier Board
GPIO 4	Output	0	Connected to the COM-E Carrier Board
GPIO 5	Output	0	Connected to the COM-E Carrier Board
GPIO 6	Output	0	Connected to the COM-E Carrier Board
GPIO 7	Output	0	Connected to the COM-E Carrier Board
GPIO 8			Multiplex as SDHC_CD: which is used to check if the SD insert or not.
GPIO 9			Multiplex as SDHC_WP: which is used to check the SD is write protect or not.

Table 5-2 GPIO Description (continued)

Name	Intput/ Output	Reset Value	Description
GPIO 10	Output	0	It can be used to clear WDT timer. If the pin is set to 1, the WDT timer will be cleared.
GPIO 11	Input		Connect to COM-E Carry board
GPIO 12	Output	1	If this pin is set to 1, and s3[14] is set to OFF, then Serdes#2 is switch to COMe PCI #2; If this pin is set to 0, or s3[14] is set to ON, then Serdes#2 is switch to GEPHY2. For Blackadder-P2020, GPIO[12] must be set to 1 and s3[14] must be set to OFF.
GPIO 13	Input		Connect to COM-E Carry board.
GPIO 14	Output		Connect to COM-E Carry board
GPIO 15	Input		Connect to COM-E Carry board

- Show GPIO Direction:
  The command: "gpio showdirection" can be used to show the setting of all GPIO's direction. For example:

```
=> gpio showdirection
GPIO0 = Input(00)
GPIO1 = Input(00)
GPIO2 = Input(00)
GPIO3 = Input(00)
GPIO4 = Output(10)
GPIO5 = Output(10)
GPIO6 = Output(10)
GPIO7 = Output(10)
GPIO8 = Input(00)
GPIO9 = Input(00)
GPIO10 = Output(10)
GPIO11 = Input(00)
GPIO12 = Output(10)
GPIO13 = Input(00)
```

```
GPIO14 = Input(00)
GPIO15 = Input(00)
=>
```

- Show GPIO input value:
  The command : "gpio showinput" can be used to show the input value of GPIO. For example:

```
=> gpio showinput f
Input of GPIO15 is: 1
```

- Set GPIO output value:
  The command: "gpio setoutput" can be used to set the output value of GPIO. For example: we can set the output of GPIO10 to 1 as below:

```
=> gpio setoutput a 1
=> gpio showinput a
Input of GPIO10 is: 1
=>
```

5.4.3 ID EEPROM (AT24C02)

There are two AT24C02 used at COMX-P2020, one is ID EEPROM, and the other is Board EEPROM.

The ID EEPROM is used to store serial number, MAC address etc. The I2C address for ID EEPROM is 0xA0(8 bit). The content of ID EEPROM is list as below:

Table 5-3 Content for ID EEPROM

Address	Name	Length	Value/ Example	Description	Static/ Variable
0x00-0x03	Tag ID	4 Byte	NXID	EEPROM Tag, Format Always the four ISO-8859 characters "NXID" with no null termination.	Static
0x04-0x0F	Serial Number	12 Byte	"12345678901\0	"From 1 to 11 ISO-8859 characters describing the module family, null terminated.	Variable

Table 5-3 Content for ID EEPROM (continued)

Address	Name	Length	Value/ Example	Description	Static/ Variable
0x10-0x14	Errata Level	5 Byte	"1234	"From 1 to 4 ISO-8859 characters describing the module family, null terminated.	Variable
0x15-0x1A	Build Data	6 Byte	10h 10h 18h 09h 02h 55h Means: 2010/10/18 09:02:55	BCD date values, as YYMMDDhhmmss	Variable
0x1B	Reserved	1 Byte		Reserved	Static
0x1C-0x1F	NXID Version	4 Byte	00h 00h 00h 00h	32bit NXID version	Static
0x20-0x3F	Reserved	20 Byte		Reserved	Static
0x40	MAC Number	1 Byte	03h	MAC number	Static
0x41	Reserved	1 Byte		Reserved	Static
0x42-0x47	MAC Address 1	6 Byte	00h 21h 79h b5h 1eh d8h Means: 00:21:79:b5:1e:d8	MAC Address for eTSEC1	Variable
0x48-0x4D	MAC Address 2	6 Byte	00h 21h 79h b5h 1eh d9h Means: 00:21:79:b5:1e:d9	MAC Address for eTSEC2	Variable
0x4E-0x53	MAC Address 3	6 Byte	00h 21h 79h b5h 1eh dah Means: 00:21:79:b5:1e:da	MAC Address for eTSEC3	Variable
0x54-0x71	Reserved	30 Byte		Reserved	Static
0x72-0x75	CRC32	4 Byte	00h a0h 14h 10h Means the CRC is: 0x00A01410	32 bit CRC of values 0x00 through 0x71, inclusive	Variable
0x76-0xFF	Reserved	138 Byte		Reserved	Static

- Chip probe:
  ```
  i2c dev 0
  Setting bus to 0
  i2c probe
  Valid chip addresses: 50 70
  Excluded chip addresses: 29
  =>
  ```

The following command can be used to program the ID EEPROM:

- Set Tag value:
  ```
  mac id
  ```
- Program system serial number, for example: 12345678901
  ```
  mac n 12345678901
  ```
- Program errata data, for example: 1234
  ```
  mac e 1234
  ```
- Set program date, for example: 2010-4-30 16:50:50
  ```
  mac d 100430165000
  ```
- Program the number of ports:
  ```
  mac p 3
  ```
- Program the MAC address for port X [X=0...2], for example:
  ```
  mac 0 00:05:9f:ff:91:01
  mac 1 00:05:9f:ff:91:02
  mac 2 00:05:9f:ff:91:03
  ```
- Save the seeting to the ID EEPROM:
  ```
  mac save
  Programming passed.
  ```
- Show the setting of the ID EEPROM:
  ```
  mac
  ID: NXID v0
  SN: 12345678901
  Errata: 1234
  Build date: 2010/04/30 16:50:00
  Eth0: 00:05:9f:ff:a1:01
  Eth1: 00:05:9f:ff:a1:02
  Eth2: 00:05:9f:ff:a1:03
  CRC: 5a314c3d
  ```

5.4.4 Board EEPROM (AT24C02)

The I2C address for Board EEPROM is 0xA8(8 bit), and it is used to store the information of processor family, module family and board configuration etc. The content of Board EEPROM is list as below:

Table 5-4 Content for Board EEPROM

Address	Name	Length	Value/ Example	Description	Static/ Variable
0x00-0x03	Tag ID	4 Byte	"COMX	"EEPROM Tag, Format Always the four ISO-8859 characters "COMX" with no null termination.	Static
0x04-0x0F	Reserved	12 Bytes		Reserved.	Static
0x10-0x1F	Processor Family	16 Bytes	"P2020\0	"From 1 to 15 ISO-8859 characters describing the processor family, null terminated.	Static
0x20-0x27	Processor Version	8 Bytes	"B\0	"From 1 to 7 ISO-8859 characters describing the processor version, null terminated	Variable
0x27-0x2F	Processor Errata	8 Bytes	"00\0	"From 1 to 7 ISO-8859 characters describing the processor errata level, null terminated.	Variable
0x30-0x3F	Module Family	16 Bytes	See MRD sec 3.2: "COME1\0" for P4080 derivatives "COME2\0" for P2020 derivitaves "COME3\0" for P1022 derivatives	From 1 to 15 ISO-8859 characters describing the module family, null terminated	Static
0x40-0x47	Module Version	8 Bytes	"GA\0	"From 1 to 7 ISO-8859 characters describing the module version, null terminated	Variable

Table 5-4 Content for Board EEPROM (continued)

Address	Name	Length	Value/ Example	Description	Static/ Variable
0x48-0x4F	Module Errata	8 Bytes	"00\0	"From 1 to 7 ISO-8859 characters describing the module errata level, null terminated	Variable
0x50-0x57	Memory Size	8 Bytes	"2048\0	"From 1 to 7 ISO-8859 characters describing the memory size in Megabytes null terminated.	Variable
0x58-0x7F	Module Description String	40 Bytes	Example: "ComExpress module on P2020(1.2GHz)\0	"Module Description String, null terminated	Static
0x80-0x9F	Manufacturer String	32 Bytes	"Emerson Network Power\0	"Module Manufacturer String, null terminated	Static
0xA0-0xFB	Reserved	92 Bytes		Reserved for future use.	Static
0xFC-0xFF	CRC32	4 Bytes	For example: 00h a0h 14h 10h Means the CRC is: 0x00A01410	Checksum (version >=1) 32 bit CRC of values 0x00 through 0xFB, inclusive	Variable

- Chip probe:

```
=> i2c dev 0
Setting bus to 0
=> i2cswitch connect 0
=> i2c probe
Valid chip addresses: 38 50 54 70
Excluded chip addresses: 29
=>
```

The following command can be used to program the Board EEPROM:

- Set Tag value:

```
brd id
```

- Set the process family:

```
brd pf P2020
```

- Set the process version:
  `brd pv B`
- Set the process errata:
  `brd pe 00`
- Set the module family:
  `brd mf COME2`
- Set the module version:
  `brd mv GA`
- Set the module errata:
  `brd me 00`
- Set the memory size:
  `brd ms 2048`
- Set the module description string:
  `brd md 'ComExpress module on P2020(1.2GHz)'`
- Set the manufacturer string:
  `brd ma 'Emerson Network Power'`
- Save the setting to Board EEPROM:
  `brd save`
- To verify the Board EEPROM (including CRC), run the command "brd" at U-Boot:

  Show the setting of the Board EEPROM:

brd

```
ID: COMX
Processor Family: P2020
Processor Version: B
Processor Errata: 00
Module Family: COME2
Module Version: GA
Module Errata: 00
Memory Size(MB): 2048
Module Description: ComExpress module on P2020(1.2GHz)
Manufacturer: Emerson Network Power
CRC: b5d7e834
=>
```

5.4.5 I2C Switch (PCA9545)

The PCA9545 is a quad bi-directional translating switch controlled via the I2C bus#1. The device's I2C address is: 0xE0(8-bit).

- Chip Probe:

```
i2c dev 0
Setting bus to 0
i2c probe
Valid chip addresses: 50 70
```

- The command "i2cswitch" can be used to connect and disconnect the I2C bus:

```
i2cswitch
Usage
i2cswitch connect I2CBusNo - connect I2C Bus to I2C Switch
i2cswitch disconnect I2CBusNo - disconnect I2C Bus to I2C
Switch
i2cswitch show  - Show I2C Bus that connected to I2C Switch
```

- Connect I2C Bus through PCA9545
  The command: "i2cswitch connect" can be used to connect the channel 0 - 3 to I2C Bus. For example, we can connect the channel 0 to I2C Bus:

```
i2c dev 0
Setting bus to 0
i2c probe
Valid chip addresses: 50 70
Excluded chip addresses: 29
=> i2cswitch connect 0
=> i2c probe
Valid chip addresses: 38 50 54 70
Excluded chip addresses: 29
=>
```

Once the channel 0 is connected to the I2C Bus, the other I2C chips (38, 54) can be detected at the testing bed which connect the I2C#1 through channel 0:

- Show the channel connected to the I2C Bus:
  The command: "i2cswitch show" can be used to show all the channels that connected to the I2C bus.

```
i2cswitch connect 0
i2cswitch connect 1
i2cswitch show
Connected I2C Bus: 0 1
```

- Disconnect channel:

```
i2cswitch disconnect 0
i2cswitch show
Connected I2C Bus: 1
=> i2c probe
Valid chip addresses: 50 70
Excluded chip addresses: 29
```

Once the channel 0 has been disconnected from the I2C bus, then the all the I2C chips at the testing bed (38, 54) will disappear from the I2C bus.

5.4.6 Temperature Sensor (LM75CIM-3)

The LM75 is a temperature sensor.

1. Chip Probe:
   LM75's I2C address is 90h(8-bit). It can be accessed at I2C#2.

```
=> i2c dev 1
Setting bus to 1
=> i2c probe
Valid chip addresses: 1B 33 48 53 68
Excluded chip addresses:
```

2. Get current temperature:
   The command: "sensor get" can be used to show current temperature:

```
=> i2c dev 1
Setting bus to 1
=> i2c probe
Valid chip addresses: 1B 33 48 53 68
Excluded chip addresses:
```

5.4.7 DDR3 SPD

1. Chip Probe:
   The DDR3 SPD address is A6h(8-bit), and it can be accessed at I2C#2.
   Chip Probe:
   The DDR3 SPD address is A6h(8-bit), and it can be accessed at I2C#2.
2. Read SPD
   The SPD information can be read out with the following commands:

```
=> i2c md 53 0.1 100
0000: 92 10 0b 08 02 11 00 09 0b 52 01 08 0c 00 34 00   .........R....4.
0010: 6c 78 6c 30 6c 11 20 8c 70 03 3c 3c 00 f0 83 05   lxl0l. .p.<<....
0020: 80 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0030: 00 00 00 00 00 00 00 00 00 00 00 00 0f 11 1f 00   ................
0040: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0050: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0060: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0070: 00 00 00 00 00 01 94 01 09 50 06 a8 11 6d fd 64   .........P...m.d
0080: 53 47 35 37 32 35 36 38 45 4d 52 30 36 39 53 32   SG572568EMR069S2
0090: 53 45 00 00 80 ce 53 4d 41 52 54 4d 6f 64 75 6c   SE....SMARTModul
00a0: 61 72 54 65 63 68 6e 6f 6c 6f 67 69 65 73 00 00   arTechnologies..
00b0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00c0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00d0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00e0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00f0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
=>
```

5.4.8 M41ST85WMX6TR

The M41ST85W can be used as RTC and Watchdog. For RTC, we can use the standard U-boot command "date" to set and show the date and time. For the watchdog, we can use the command: "wdg" to enable, disable and reset the watchdog.

```
=> wdg
Usage
wdg enable second - Enable Watchdog at second
wdg disable  - Disable Watchdog
wdg reset  - Reset Watchdog
```

1. Chip Probe:
   The M41ST85W address is D0h(8-bit), and it can be accessed at I2C#2.

   ```
   => i2c dev 1
   Setting bus to 1
   => i2c probe
   Valid chip addresses: 1B 33 48 53 68
   Excluded chip addresses:
   ```

2. Set and show date and time:
   The standard U-boot command "date" can be used to set and show the date for M41ST85W. For example, we can set the time to 2010-05-25 11:11:50 by the following command:

```
=> date 052511112010.50
Date: 2010-05-25 (Tuesday)    Time: 11:11:50
=> date
Date: 2010-05-25 (Tuesday)    Time: 11:11:51
=>
```

3. Enable Watchdog
   We can Enable the watchdog by the command: "wdg enable"; For example:

```
=> wdg enable 20
Watchdog Enable at 32 second!
```

   The above command means that once we enable the watchdog and don't reset the watchdog or disable the watchdog in 32 seconds, the system will reboot automatically. The time of the watchdog can be set from 1 -124 seconds.

4. Reset Watchdog
   There are two ways to reset the watchdog:
   - A transition (high-to-low or low-to-high) can be applied to the Watchdog Input pin (WDI) which connects with the GPIO10.
   - The processor can perform a WRITE of the Watchdog Register. The time-out period then starts over.

   Here we use the first way to reset watchdog. The command: "wdg reset" can be used to reset the watchdog:

```
=> wdg enable 20
=> wdg reset
=> wdg reset
```

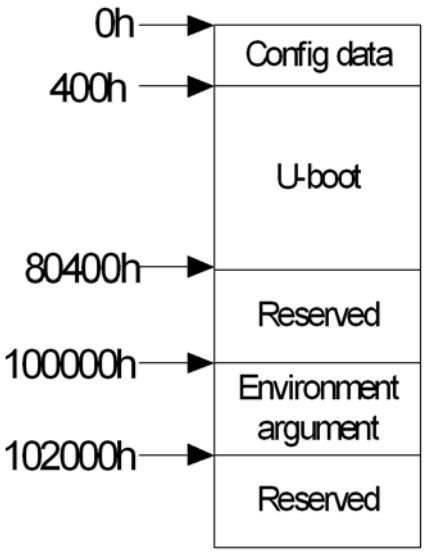
5. Disable Watchdog
   We can disable the watchdog by the command: "wdg disable":

```
=> wdg enable 20
=> wdg disable
```

5.4.9 SPI Flash

COMX-P2020 has an on-module SPI socket with 4MB SPI FLASH M25P32-VMW6 installed. It can be used to store U-boot and environment arguments. The content of SPI Flash is list as below:

Figure 5-1 SPI Flash Content



The U-boot provides the standard command "sf" to check, read, erase and write SPI Flash. This part is not included in this document.

```
=> help sf
sf - SPI flash sub-system
Usage:
sf probe [bus:]cs [hz] [mode]   - init flash device on given SPI bus
                                  and chip select
sf read addr offset len         - read `len' bytes starting at
                                  `offset' to memory at `addr'
sf write addr offset len        - write `len' bytes from memory
                                  at `addr' to flash at `offset'
sf erase offset len             - erase `len' bytes from `offset'
```

5.4.10 SD Card

The U-boot provides the standard command "mmc" to check, read, erase and write SD Card. This part is not included in this document.

```
=> help mmc
mmc - MMC sub system
Usage:
mmc read <device num> addr blk# cnt
mmc write <device num> addr blk# cnt
mmc rescan <device num>
mmc list - lists available devices
```

5.4.11 USB

COMX-P2020 implements a dual-role (DR) USB module. This module is connected to USB2514i which can provide 4 downstream ports. U-boot can scan the USB devices which connected with USB2514i. If the device is a storage device, then U-boot can read and write the block of the device.

The U-boot provides standard command: usb to find and read USB storage devices. This part is not included in this document.

```
=> help usb
usb - USB sub-system

Usage:
usb reset - reset (rescan) USB controller
usb stop [f]  - stop USB [f]=force stop
usb tree  - show USB device tree
usb info [dev] - show available USB devices
usb storage  - show details of USB storage devices
usb dev [dev] - show or set current USB storage device
usb part [dev] - print partition table of one or all USB storage devices
usb read addr blk# cnt - read `cnt' blocks starting at block `blk#'
to memory address `addr'
```

5.4.12 PCI Express

There are three PCI Express controllers at COMX-P2020: the PCI Express 1 is connected to GPU Z11M, the other two are connected to the carrier board. The U-boot will scan and show the PCI devices when booting as follows:

```
PCIE3 connected to Slot0 as Root Complex (base addr ffe08000)
Current Status: LSR-11, LTSSM-16, PEX width-x1, Clock-2.5GT/s
          Scanning PCI bus 01
    01  00  8086  107d  0200  00
PCIE3 on bus 00 - 01

PCIE2 connected to Slot 1 as Root Complex (base addr ffe09000)
PCIE2 on bus 02 - 02

PCIE1 connected to Slot 2 as Root Complex (base addr ffe0a000)
Current Status: LSR-11, LTSSM-16, PEX width-x1, Clock-2.5GT/s
          Scanning PCI bus 04
    04  00  18ca  0027  0300  00
PCIE1 on bus 03 - 04
```

The U-boot provides standard command "pci" to probe and configure PCI Express devices. This part is not included in this document.

```
=> pci
Scanning PCI devices on bus 0
BusDevFun  VendorId   DeviceId   Device Class       Sub-Class
_____________________________________________________________
00.00.00   0x1957     0x0070     Processor              0x20
=> help pci
pci - list and access PCI Configuration Space

Usage:
pci [bus] [long]
    - short or long list of PCI devices on bus 'bus'
pci header b.d.f
    - show header of PCI device 'bus.device.function'
pci display[.b, .w, .l] b.d.f [address] [# of objects]
    - display PCI configuration space (CFG)
pci next[.b, .w, .l] b.d.f address
    - modify, read and keep CFG address
pci modify[.b, .w, .l] b.d.f address
    -  modify, auto increment CFG address
pci write[.b, .w, .l] b.d.f address value
    - write to CFG address
```

5.4.13 eTSEC & BCM5482

COMX-P2020 provides three enhanced three-speed Ethernet controllers (eTSECs) which interface to 10Mbps, 100Mbps, and 1Gbps Ethernet/IEEE 802.3 networks. Two BCM5482 was used to connect with eTSECs through RGMII. Each BCM5482 can provide two Ethernet transceivers designed for 1000Mbps, 100Mbps and 10Mbps application. The Phy Address for the eTSECs are list below:

eTSEC0 : phyID = 0

eTSEC1 : phyID = 2

eTSEC2 : phyID = 1

The U-boot provides standard command "mii" to probe, read and write BCM5482 register. This part is not included in this document.

```
=> help mii
mii - MII utility commands

Usage:
mii device                     - list available devices
mii device <devname>           - set current device
mii info   <addr>              - display MII PHY info
mii read   <addr> <reg>        - read  MII PHY <addr> register <reg>
mii write  <addr> <reg> <data> - write MII PHY <addr> register <reg>
mii dump   <addr> <reg>        - pretty-print <addr> <reg> (0-5 only)
Addr and/or reg may be ranges, e.g. 2-7.
=>
```

5.5 Configure U-boot Environment

5.5.1 Default U-boot environment

When the COMX-P2020 U-boot comes up, it creates the following environment by default:

```
=> printenv
ramboot=setenv bootargs root=/dev/ram rw DVO_OUTPUT=$DVO_OUTPUT
console=$consoledev,$baudrate $othbootargs;tftp $ramdiskaddr
$tftppath/$ramdiskfile;tftp $loadaddr $tftppath/$bootfile;tftp $fdtaddr
$tftppath/$fdtfile;bootm $loadaddr $ramdiskaddr $fdtaddr
```

```
nfsboot=setenv bootargs root=/dev/nfs rw nfsroot=$serverip:$rootpath
ip=$ipaddr:$serverip:$gatewayip:$netmask:$hostname:$netdev:off
DVO_OUTPUT=$DVO_OUTPUT console=$consoledev,$baudrate $othbootargs;tftp
$loadaddr $tftppath/$bootfile;tftp $fdtaddr $tftppath/$fdtfile;bootm
$loadaddr - $fdtaddr
bootdelay=10
baudrate=115200
loads_echo=1
ipaddr=192.168.0.250
serverip=192.168.0.197
rootpath=/local/tftpboot/COMX-P2020/current/rootfs_nfs
gatewayip=192.168.0.1
netmask=255.255.255.0
hostname=COMX-P2020
bootfile=uImage
loadaddr=1000000
bootcmd=run sdboot
sdboot=setenv bootargs root=/dev/mmcblk0p2 rw rootdelay=$rootdelaysecond
DVO_OUTPUT=$DVO_OUTPUT console=$consoledev,$baudrate $othbootargs;
mmcinfo;ext2load mmc 0:2 $loadaddr /boot/$bootfile;ext2load mmc 0:2
$fdtaddr /boot/$fdtfile;bootm $loadaddr - $fdtaddr
sdfatboot=setenv bootargs root=/dev/ram rw rootdelay=$rootdelaysecond
DVO_OUTPUT=$DVO_OUTPUT console=$consoledev,$baudrate $othbootargs;
mmcinfo;fatload mmc 0:1 $loadaddr $bootfile;fatload mmc 0:1 $fdtaddr
$fdtfile;fatload mmc 0:1 $ramdiskaddr $ramdiskfile;bootm $loadaddr
$ramdiskaddr $fdtaddr
usbboot=setenv bootargs root=/dev/sda1 rw rootdelay=$rootdelaysecond
DVO_OUTPUT=$DVO_OUTPUT console=$consoledev,$baudrate $othbootargs;usb
start;ext2load usb 0:1 $loadaddr /boot/$bootfile;ext2load usb 0:1
$fdtaddr /boot/$fdtfile;bootm $loadaddr - $fdtaddr
usbfatboot=setenv bootargs root=/dev/ram rw DVO_OUTPUT=$DVO_OUTPUT
console=$consoledev,$baudrate $othbootargs; usb start; fatload usb 0:2
$loadaddr $bootfile; fatload usb 0:2 $fdtaddr $fdtfile; fatload usb 0:2
$ramdiskaddr $ramdiskfile; bootm $loadaddr $ramdiskaddr $fdtaddr
usbext2boot=setenv bootargs root=/dev/ram rw DVO_OUTPUT=$DVO_OUTPUT
console=$consoledev,$baudrate $othbootargs; usb start; ext2load usb 0:4
$loadaddr $bootfile; ext2load usb 0:4 $fdtaddr $fdtfile; ext2load usb 0:4
$ramdiskaddr $ramdiskfile; bootm $loadaddr $ramdiskaddr $fdtaddr
```

```
upgradespi=sf probe 0;setenv startaddr 0;setenv erasesize a0000;tftp
1000000 $tftppath/$uboot_spi;sf erase $startaddr $erasesize;sf write
1000000 $startaddr $filesize;sf erase 100000 120000
clearspienv=sf probe 0;sf erase 100000 20000
othbootargs=ramdisk_size=700000 cache-sram-size=0x10000
netdev=eth0
rootdelaysecond=15
uboot_nor=u-boot-nor.bin
uboot_spi=u-boot-spi.bin
uboot_sd=u-boot-sd.bin
consoledev=ttyS0
ramdiskaddr=2000000
ramdiskfile=rootfs-dev.ext2.img
fdtaddr=c00000
fdtfile=comx.dtb
tftppath=COMX-P2020/current
DVO_OUTPUT=enable
ethaddr=00:80:42:05:44:02
eth1addr=00:80:42:05:44:03
eth2addr=00:80:42:05:44:04
ethact=eTSEC1

Environment size: 2804/8188 bytes
=>
```

5.5.2 Configure U-boot for tftp

The following environment settings are required to configure the u-boot for tftp:

```
=> setenv ethaddr <MAC Address 1>
=> setenv eth1addr <MAC Address 2>
=> setenv eth2addr <MAC Address 3>
=>setenv ethact eTSEC1
=>setenv ipaddr <board_ipaddress>
=>setenv serverip <tftp_serverip>
=>setenv gatewayip <your_gatewayip>
=>setenv netmask <net_mask>
```

```
=> ping <tftp_serverip>
Enet starting in 100BT/FD
Speed: 100, full duplex
Using eTSEC1 device
host 192.168.0.197 is alive
=> tftp COMX-P2020/current/u-boot-spi.bin
Enet starting in 100BT/FD
Speed: 100, full duplex
Using eTSEC1 device
TFTP from server 192.168.0.197; our IP address is 192.168.0.250
Filename 'COMX-P2020/current/u-boot-spi.bin'.
Load address: 0x1000000
Loading: #################################
done
Bytes transferred = 525312 (80400 hex)
=>
=>saveenv
```

5.5.3 Configuring U-Boot for Ramdisk Deployment using TFTP

The following environment settings are required to configure the u-boot for ramdisk deployment using tftp.

At U-boot prompt, set u-boot environment like:

```
=>setenv ipaddr <board_ipaddress>
=>setenv serverip <tftp_serverip>
=>setenv gatewayip <your_gatewayip>
=>setenv netmask <net_mask>
=>setenv ramdiskfile <root_image>
=>setenv tftppath COMX-P2020/current
=>saveenv
```

Now the board is ready to start Linux with RASDISK.

To load the RAMDISK of developer rootfs image, please set the following environment variable:

```
=>setenv ramdiskfile rootfs-dev.ext2.img
```

To load the RAMDISK of user rootfs image, please set the following environment variable:

```
=>setenv ramdiskfile rootfs-usr.ext2.img
```

To load the RAMDISK of min rootfs image, please set the following environment variable:

```
=>setenv ramdiskfile rootfs-min.ext2.img
```

5.6 Upgrade U-boot for SPI Flash Using TFTP

1. Copy the U-boot binary file: u-boot-spi.bin to the directory: /local/tftpboot/COMX-P2020/current at tftp server.
2. Configure U-boot for tftp:

```
=> setenv ethaddr <MAC Address 1>
=> setenv eth1addr <MAC Address 2>
=> setenv eth2addr <MAC Address 3>
=>setenv ethact eTSEC1
=>setenv ipaddr <board_ipaddress>
=>setenv serverip <tftp_serverip>
=>setenv gatewayip <your_gatewayip>
=>setenv netmask <net_mask>
=>setenv tftppath COMX-P2020/current
=> ping <tftp_serverip>
Enet starting in 100BT/FD
Speed: 100, full duplex
Using eTSEC1 device
host 192.168.0.197 is alive
=> tftp COMX-P2020/current/u-boot-spi.bin
Enet starting in 100BT/FD
Speed: 100, full duplex
Using eTSEC1 device
TFTP from server 192.168.0.197; our IP address is 192.168.0.250
Filename 'COMX-P2020/current/u-boot-spi.bin'.
```

```
Load address: 0x1000000
Loading: #####################################
done
Bytes transferred = 525312 (80400 hex)
=>
=>saveenv
```

3. Run command "run upgradespi" to upgrade U-boot for SPI Flash.

```
run upgradespi
4096 KiB S25FL032A(P) at 0:0 is now current device
Enet starting in 100BT/FD
Speed: 100, full duplex
Using eTSEC1 device
TFTP from server 192.168.0.197; our IP address is 192.168.0.250
Filename 'u-boot-spi.bin'.
Load address: 0x1000000
Loading: #####################################
done
Bytes transferred = 525312 (80400 hex)
```

4. Reset the board to boot up from new U-boot image.

Start Linux

6.1 RAMDISK Deployment Using tftp

1. Configure U-Boot for RAMDISK Deployment using TFTP.

```
=> setenv ethaddr <MAC Address 1>
=> setenv eth1addr <MAC Address 2>
=> setenv eth2addr <MAC Address 3>
=>setenv ethact eTSEC1
=>setenv ipaddr <board_ipaddress>
=>setenv serverip <tftp_serverip>
=>setenv gatewayip <your_gatewayip>
=>setenv netmask <net_mask>
=> setenv ramdiskfile rootfs-min.ext2.img
```

2. Run command: "run ramboot" to start Linux, for example:

```
=> run ramboot
Enet starting in 100BT/FD
Speed: 100, full duplex
Using eTSEC1 device
TFTP from server 192.168.0.197; our IP address is 192.168.0.250
Filename 'COMX-P2020/rootfs-min.ext2.img'.
Load address: 0x2000000
Loading:
#################################################################

#################################################################

#################################################################
         ##########################################
done
Bytes transferred = 3473422 (35000e hex)
Enet starting in 100BT/FD
Speed: 100, full duplex
Using eTSEC1 device
TFTP from server 192.168.0.197; our IP address is 192.168.0.250
Filename 'COMX-P2020/COMX.uImage'.
Load address: 0x1000000
```

```
Loading:
#################################################################

#################################################################

#################################################################
         ######################################
done
Bytes transferred = 3416476 (34219c hex)
Enet starting in 100BT/FD
Speed: 100, full duplex
Using eTSEC1 device
TFTP from server 192.168.0.197; our IP address is 192.168.0.250
Filename 'COMX-P2020/COMX.dtb'.
Load address: 0xc00000
Loading: #
Done
……
……
……

        Welcome to the LTIB Embedded Linux Environment

!!!!! WARNING !!!!!!!

The default password for the root account is: root
please change this password using the 'passwd' command
and then edit this message (/etc/issue) to remove this message

COMX-P2020 login: root
Password:
[root@COMX-P2020 /root]#
```

6.2 NFS Deployment

1. Configure U-Boot for NFS Deployment.

```
=> setenv ethaddr <MAC Address 1>
=> setenv eth1addr <MAC Address 2>
=> setenv eth2addr <MAC Address 3>
=>setenv ethact eTSEC1
=>setenv ipaddr <board_ipaddress>
=>setenv serverip <tftp_serverip>
=>setenv gatewayip <your_gatewayip>
=>setenv netmask <net_mask>
=> setenv tftppath COMX-P2020/current/rootfs_nfs
=> setenv rootpath /local/tftpboot/COMX-P2020/current/rootfs_nfs
```

2. Run Command: "run nfsboot" to start Linux, for example:

```
=> run nfsboot
Enet starting in 100BT/FD
Speed: 100, full duplex
Using eTSEC1 device
TFTP from server 192.168.0.197; our IP address is 192.168.0.250
Filename 'COMX-P2020/current/uImage'.
Load address: 0x1000000
Loading:
#################################################################

#################################################################

#################################################################
         ########################################
done
Bytes transferred = 3416476 (34219c hex)
Enet starting in 100BT/FD
Speed: 100, full duplex
Using eTSEC1 device
TFTP from server 192.168.0.197; our IP address is 192.168.0.250
Filename 'COMX-P2020/current/comx.dtb'.
Load address: 0xc00000
```

```
Loading: #
done
Bytes transferred = 9824 (2660 hex)
WARNING: adjusting available memory to 30000000
## Booting kernel from Legacy Image at 01000000 ...
   Image Name:   Linux-2.6.32
   Created:      2010-09-28   2:19:17 UTC
   Image Type:   PowerPC Linux Kernel Image (gzip compressed)
   Data Size:    3416412 Bytes =  3.3 MB
   Load Address: 00000000
   Entry Point:  00000000
   Verifying Checksum ... OK
…
…


        Welcome to the LTIB Embedded Linux Environment

!!!!! WARNING !!!!!!!

The default password for the root account is: root
please change this password using the 'passwd' command
and then edit this message (/etc/issue) to remove this message

COMX-P2020 login: root
Password:
[root@COMX-P2020 root]#
```

6.3 Starting Linux from USB Disk

The Linux kernel and filesystem can be put to the USB disk for production deployment. As such the USB sticks included in the COMX-P2020 kit will contain the following images as per the partitions mentioned.

Table 6-1 Partitions on USB Disk

Partition Number	Size	Type	Contents
1	1G	Ext2 (System ID = 83)	Inflated rootfs required for HD boot
2	1G	FAT16 (System ID = 6)	uImage, comx.dtb, rootfs-dev.ext2.img, rootfs-usr.ext2.img, rootfs-min.ext2.img
3	1G	Linux swap (System ID = 82)	
4	1G	Ext2 (System ID = 83)	uImage, comx.dtb, rootfs-dev.ext2.img, rootfs-usr.ext2.img, rootfs-min.ext2.img

6.3.1 Deploy Filesystem to the USB Disk

The following method can deploy filesystem to an empty 4G Byte USB Disk:

1. Start COMX-P2020 to Linux with NFS file system.
2. Insert a 4G Byte USB sticks to COMX-P2020 and fdisk the USB disk as the above partition table.

```
[root@COMX-P2020 /root]#fdisk /dev/sda
...
   Device Boot      Start         End      Blocks   Id  System
/dev/sda1               1         142     1140583+  83  Linux
/dev/sda2             143         265      987997+   6  FAT16
/dev/sda3             266         396     1052257+  82  Linux swap
/ Solaris
/dev/sda4             397         518      979965   83  Linux
```

3. Format the partition with the following command:

```
[root@COMX-P2020 root]# umount /dev/sda1
[root@COMX-P2020 root]# umount /dev/sda2
[root@COMX-P2020 root]# umount /dev/sda3
[root@COMX-P2020 root]# umount /dev/sda4
```

```
[root@COMX-P2020 /root]#mkfs.ext2 /dev/sda1
[root@COMX-P2020 /root]#mkfs.vfat /dev/sda2
[root@COMX-P2020 /root]#mkfs.ext2 /dev/sda4
```

4. Mount the partitions and copy the desired files from tftp server to the partitions with the following command:

```
[root@COMX-P2020 /root]#mkdir /mnt/usb1
[root@COMX-P2020 /root]#mkdir /mnt/usb2
[root@COMX-P2020 /root]#mkdir /mnt/usb4

[root@COMX-P2020 /root]#mount /dev/sda1 /mnt/usb1
[root@COMX-P2020 /root]#mount /dev/sda2 /mnt/usb2
[root@COMX-P2020 /root]#mount /dev/sda4 /mnt/usb4
```

5. Copy the release image to partition2 of USB.

```
[root@COMX-P2020 /root]#scp
<user>@<tftp_serverip>:/local/tftpboot/COMX-P2020/current/u-boot-
*.bin /mnt/usb2
[root@COMX-P2020 /root]#scp
<user>@<tftp_serverip>:/local/tftpboot/COMX-
P2020/current/comx.dtb /mnt/usb2
[root@COMX-P2020 /root]#scp
<user>@<tftp_serverip>:/local/tftpboot/COMX-P2020/current/uImage
/mnt/usb2
[root@COMX-P2020 /root]#scp
<user>@<tftp_serverip>:/local/tftpboot/COMX-P2020/current/rootfs-
*.* /mnt/usb2
[root@COMX-P2020 /root]# ls -al /mnt/usb2
total 613944
drwxr-xr-x  2 root root      4096 Sep 28 14:38 .
drwxr-xr-x 26 root root      4096 Sep 28 06:25 ..
-rwxr-xr-x  1 root root      9824 Sep 28 14:34 comx.dtb
-rwxr-xr-x  1 root root   3416476 Sep 28 14:34 uImage
-rwxr-xr-x  1 root root 195078114 Sep 28 14:37 rootfs-LRFS.tar.gz
-rwxr-xr-x  1 root root 198560701 Sep 28 14:36 rootfs-dev.ext2.img
-rwxr-xr-x  1 root root   3473422 Sep 28 14:37 rootfs-min.ext2.img
-rwxr-xr-x  1 root root 195107868 Sep 28 14:38 rootfs-nfs.tar.gz
-rwxr-xr-x  1 root root  31955486 Sep 28 14:38 rootfs-usr.ext2.img
-rwxr-xr-x  1 root root    524288 Sep 28 14:33 u-boot-sd.bin
```

```
-rwxr-xr-x  1 root root    525312 Sep 28 14:33 u-boot-spi.bin
[root@COMX-P2020 user]#
```

6. Copy the release image to partition4 of USB.

```
[root@COMX-P2020 root]# cp /mnt/usb2/* /mnt/usb4/.
[root@COMX-P2020 root]# ls -al /mnt/usb4
total 614592
drwxr-xr-x  3 root root      4096 Sep 28 14:46 .
drwxr-xr-x 26 root root      4096 Sep 28 06:25 ..
-rwxr-xr-x  1 root root      9824 Sep 28 14:42 comx.dtb
-rwxr-xr-x  1 root root   3416476 Sep 28 14:42 uImage
drwx------  2 root root     16384 Sep 28 14:22 lost+found
-rwxr-xr-x  1 root root 195078114 Sep 28 14:43 rootfs-LRFS.tar.gz
-rwxr-xr-x  1 root root 198560701 Sep 28 14:44 rootfs-dev.ext2.img
-rwxr-xr-x  1 root root   3473422 Sep 28 14:44 rootfs-min.ext2.img
-rwxr-xr-x  1 root root 195107868 Sep 28 14:46 rootfs-nfs.tar.gz
-rwxr-xr-x  1 root root  31955486 Sep 28 14:46 rootfs-usr.ext2.img
-rwxr-xr-x  1 root root    524288 Sep 28 14:46 u-boot-sd.bin
-rwxr-xr-x  1 root root    525312 Sep 28 14:46 u-boot-spi.bin
[root@COMX-P2020 root]#
```

7. Decompress the inflated file system to partition1 of USB.

```
[root@COMX-P2020 root]# tar zxvf /mnt/usb2/rootfs-LRFS.tar.gz -C
/mnt/usb1
[root@COMX-P2020 root]# cp /mnt/usb2/uImage /mnt/usb1/boot/.
[root@COMX-P2020 root]# cp /mnt/usb2/comx.dtb /mnt/usb1/boot/.
 [root@COMX-P2020 root]# ls -al /mnt/usb1
total 100
drwxrwxr-x 22 user user  4096 Sep 28 14:53 .
drwxr-xr-x 26 root root  4096 Sep 28 06:25 ..
drwxr-xr-x  2 user user  4096 Sep 28 14:53 bin
drwxr-xr-x  2 user user  4096 Sep 28 14:48 boot
drwxr-xr-x  2 user user  4096 Sep 28 14:53 boot_format
drwxr-xr-x  2 user user  4096 Sep 28 14:53 dev
drwxrwxr-x 11 user user  4096 Sep 28 14:53 etc
drwxr-xr-x  3 user user  4096 Sep 28 14:48 home
drwxr-xr-x 10 user user  4096 Sep 28 14:48 include
drwxr-xr-x  5 user user  4096 Sep 28 14:48 lib
```

```
lrwxrwxrwx  1 user user    11 Sep 28 14:53 linuxrc -> bin/busybox
drwx------  2 root root 16384 Sep 28 14:22 lost+found
drwxr-xr-x  6 user user  4096 Sep 28 14:53 man
drwxr-xr-x 18 user user  4096 Sep 28 14:48 mnt
drwxr-xr-x  3 user user  4096 Sep 28 14:53 opt
drwxr-xr-x  2 user user  4096 Jan 14  2010 proc
drwxrwxr-x  3 user user  4096 Sep 28 14:53 root
drwxr-xr-x  2 user user  4096 Sep 28 14:53 sbin
drwxr-xr-x  3 user user  4096 Sep 28 14:53 share
drwxr-xr-x  2 user user  4096 Jan 14  2010 sys
drwxrwxr-x  2 user user  4096 Sep 17 01:39 tmp
drwxrwxr-x 15 user user  4096 Sep 28 14:53 usr
drwxr-xr-x 11 user user  4096 Sep 28 14:53 var
[root@COMX-P2020 root]#
[root@COMX-P2020 root]# ls -al /mnt/usb1/boot/*
-rwxr-xr-x 1 root root    9824 Sep 28 14:59 /mnt/usb1/boot/comx.dtb
-rwxr-xr-x 1 root root 3416476 Sep 28 14:59 /mnt/usb1/boot/uImage
[root@COMX-P2020 root]#
```

8. Umount the USB device to ensure all the date has been written to USB.

```
[root@COMX-P2020 root]#umount /mnt/usb1

[root@COMX-P2020 root]#umount /mnt/usb2
[root@COMX-P2020 root]#umount /mnt/usb4
```

6.3.2 Starting kernel with Inflated ext2FS

By default the USB stick will contain inflated ext2FS on partition 1. This can be used to do a hard disk boot from the USB stick. USB disk may also be used here.

To starting kernel with inflated ext2FS, please run the command: run usbboot at U-boot prompt:

```
=> run usbboot
(Re)start USB...
USB:   Register 10011 NbrPorts 1
USB EHCI 1.00
scanning bus for devices... 3 USB Device(s) found
       scanning bus for storage devices... 1 Storage Device(s) found
```

```
Loading file "/boot/uImage" from usb device 0:1 (usbda1)
.............................................................
.....................................3416476 bytes read
.Loading file "/boot/comx.dtb" from usb device 0:1 (usbda1)
9824 bytes read
WARNING: adjusting available memory to 30000000
## Booting kernel from Legacy Image at 01000000 ...
   Image Name:   Linux-2.6.32
   Created:      2010-09-28   2:19:17 UTC
   Image Type:   PowerPC Linux Kernel Image (gzip compressed)
   Data Size:    3416412 Bytes =  3.3 MB
   Load Address: 00000000
   Entry Point:  00000000
   Verifying Checksum ... OK
## Flattened Device Tree blob at 00c00000
   Booting using the fdt blob at 0xc00000
   Uncompressing Kernel Image ... OK
WARNING: could not find compatible node fsl,p1020-immr: FDT_ERR_NOTFOUND.
Using P2020 RDB machine description
Memory CAM mapping: 256/256/256 Mb, residual: 1280Mb
Linux version 2.6.32 (percy@localhost.localdomain) (gcc version 4.3.2
(GCC) ) #1 SMP Tue Sep 28 10:19:08 HKT 2010
CPU maps initialized for 1 thread per core
bootconsole [udbg0] enabled
setup_arch: bootmem
mpc85xx_rdb_setup_arch()
……



        Welcome to the LTIB Embedded Linux Environment

!!!!! WARNING !!!!!!!

The default password for the root account is: root
please change this password using the 'passwd' command
```

```
and then edit this message (/etc/issue) to remove this message

COMX-P2020 login: root
Password:
[root@COMX-P2020 root]#
```

6.3.3 Starting kernel with Ramdisk using FAT16 partition

By default the USB stick will contain FAT16 partition on partition 2, as shown in the above table.

To boot from this FAT16 partition, execute the following command:

```
=> run usbfatboot
(Re)start USB...
USB:   Register 10011 NbrPorts 1
USB EHCI 1.00
scanning bus for devices... 3 USB Device(s) found
       scanning bus for storage devices... 1 Storage Device(s) found
reading uImage
.....................................................................
.....................................

3416476 bytes read
reading comx.dtb
.
9824 bytes read
reading rootfs-min.ext2.img
.......................
.....................................................................
...................

3473422 bytes read
WARNING: adjusting available memory to 30000000
## Booting kernel from Legacy Image at 01000000 ...
   Image Name:   Linux-2.6.32
   Created:      2010-09-28   2:19:17 UTC
   Image Type:   PowerPC Linux Kernel Image (gzip compressed)
```

```
   Data Size:    3416412 Bytes =  3.3 MB
   Load Address: 00000000
   Entry Point:  00000000
   Verifying Checksum ... OK
## Loading init Ramdisk from Legacy Image at 02000000 ...
   Image Name:   COMX ext2 ramdisk rootfs
   Created:      2010-09-28   2:22:47 UTC
   Image Type:   PowerPC Linux RAMDisk Image (gzip compressed)
   Data Size:    3473358 Bytes =  3.3 MB
   Load Address: 00000000
   Entry Point:  00000000
   Verifying Checksum ... OK
## Flattened Device Tree blob at 00c00000
   Booting using the fdt blob at 0xc00000
   Uncompressing Kernel Image ... OK
   Loading Ramdisk to 2fcb0000, end 2fffffce ... OK
……


        Welcome to the LTIB Embedded Linux Environment

!!!!! WARNING !!!!!!!

The default password for the root account is: root
please change this password using the 'passwd' command
and then edit this message (/etc/issue) to remove this message

COMX-P2020 login: root
Password:
[root@COMX-P2020 /root]#
```

6.3.4 Starting the Kernel with Ramdisk using ext2 Partition

By default the USB stick will contain ext2 partition on partition 4, as shown in the above table.

To boot from the partition 4, execute the following command:

```
=> run usbext2boot
(Re)start USB...
USB:   Register 10011 NbrPorts 1
USB EHCI 1.00
scanning bus for devices... 3 USB Device(s) found
       scanning bus for storage devices... 1 Storage Device(s) found
Loading file "uImage" from usb device 0:4 (usbda4)
.................................................................
.....................................3416476 bytes read
Loading file "comx.dtb" from usb device 0:4 (usbda4)
.9824 bytes read
Loading file "rootfs-dev.ext2.img" from usb device 0:4 (usbda4)
......................
.................................................................
.................................................................
.................................................................
.................................................................
.................................................................

Fingerprint: md5 03:02:17:fb:90:97:05:48:23:ec:2b:7a:07:af:9e:27
Starting the dropbear ssh server:
Starting the boa webserver:


        Welcome to the LTIB Embedded Linux Environment

!!!!! WARNING !!!!!!!

The default password for the root account is: root
please change this password using the 'passwd' command
and then edit this message (/etc/issue) to remove this message

COMX-P2020 login: PHY: mdio@ffe24520:00 - Link is Up - 100/Full
ADDRCONF(NETDEV_CHANGE): eth0: link becomes ready
```

```
        Welcome to the LTIB Embedded Linux Environment

!!!!! WARNING !!!!!!!

The default password for the root account is: root
please change this password using the 'passwd' command
and then edit this message (/etc/issue) to remove this message

COMX-P2020 login: root
Password:
[root@COMX-P2020 root]#
```

6.4 Starting Linux from SD Card

The Linux kernel and filesystem can be put to the SD card for production deployment. As such the SD card included in the COMX-P2020 kit will contain the following images as per the partitions mentioned.

Table 6-2 Partitions on the SD card

Partition Number	Size	Type	Contents
1	300M	FAT16 (System ID = 6)	FAT16 Partition to store BSP targets
	516K or more	RAW Data	U-boot (512K Byte) + 4K U-boot Environment Parameter. (This is for booting from SD Card)
2	1.7G	Ext3 (System ID = 83)	Inflated Root File system included: uImage, comx.dtb and root file system.

6.4.1 Deploy Filesystem to the SD Card

The following method can deploy filesystem to an empty 2GB SD Card:

1. Start COMX-P2020 to Linux with NFS file system.
2. Insert a 2GB SD to COMX-P2020 and fdisk the SD as the above partition table:
   `[root@COMX-P2020 root]# umount /dev/mmcblk0p1`

```
[root@COMX-P2020 root]# umount /dev/mmcblk0p2
[root@COMX-P2020 root]# fdisk /dev/mmcblk0
......
Command (m for help): p

Disk /dev/mmcblk0: 2002 MB, 2002780160 bytes
62 heads, 62 sectors/track, 1017 cylinders
Units = cylinders of 3844 * 512 = 1968128 bytes
Disk identifier: 0x00000000

        Device Boot      Start         End      Blocks   Id  System
/dev/mmcblk0p1               1         152      292113    6  FAT16
/dev/mmcblk0p2             154        1017     1660608   83  Linux

……
```

By default, we set the size of the first partition to 300 MB, which is big enough to install the BSP targets for the SD card.

3. Format the partition with the following command:

```
[root@COMX-P2020 root]# umount /dev/mmcblk0p1
[root@COMX-P2020 root]# umount /dev/mmcblk0p2
[root@COMX-P2020 /root]#mkfs.vfat /dev/mmcblk0p1
[root@COMX-P2020 /root]#mkfs.ext3 /dev/mmcblk0p2
```

4. Mount the partitions and copy the desired files from tftp server to the partitions with the following command:

```
[root@COMX-P2020 /root]#mkdir /mnt/sd1
[root@COMX-P2020 /root]#mkdir /mnt/sd2

[root@COMX-P2020 /root]#mount /dev/mmcblk0p1 /mnt/sd1
[root@COMX-P2020 /root]#mount /dev/mmcblk0p2 /mnt/sd2
```

5. Copy the release image to partition1 of SD.

```
[root@COMX-P2020 /root]#scp
<user>@<tftp_serverip>:/local/tftpboot/COMX-P2020/current/u-boot-
*.bin /mnt/sd1
[root@COMX-P2020 /root]#scp
<user>@<tftp_serverip>:/local/tftpboot/COMX-
P2020/current/comx.dtb /mnt/sd1
```

```
[root@COMX-P2020 /root]#scp
<user>@<tftp_serverip>:/local/tftpboot/COMX-P2020/current/uImage
/mnt/sd1
 [root@COMX-P2020 /root]#scp
<user>@<tftp_serverip>:/local/tftpboot/COMX-P2020/current/rootfs-
*.img /mnt/sd1
[root@COMX-P2020 /root]#scp
<user>@<tftp_serverip>:/local/tftpboot/COMX-P2020/current/
rootfs-LRFS.tar.gz /mnt/sd1
 [root@COMX-P2020 root]# ls -al /mnt/sd1
total 262064
drwxr-xr-x  2 root root      4096 Nov 26 16:34 .
drwxr-xr-x 14 root root      4096 Dec 14  2010 ..
-rwxr-xr-x  1 root root      9824 Nov 26 16:30 comx.dtb
-rwxr-xr-x  1 root root 107400408 Nov 26 16:35 rootfs-LRFS.tar.gz
-rwxr-xr-x  1 root root 110443332 Nov 26 16:32 rootfs-dev.ext2.img
-rwxr-xr-x  1 root root   3872536 Nov 26 16:32 rootfs-min.ext2.img
-rwxr-xr-x  1 root root  42137827 Nov 26 16:33 rootfs-usr.ext2.img
-rwxr-xr-x  1 root root    524288 Nov 26 16:30 u-boot-sd.bin
-rwxr-xr-x  1 root root    525312 Nov 26 16:30 u-boot-spi.bin
-rwxr-xr-x  1 root root   3416506 Nov 26 16:31 uImage [root@COMX-
P2020 root]#
```

6. Copy the release image to partition2 of SD.

```
[root@COMX-P2020 root]# cd /mnt/sd2
[root@COMX-P2020 sd2]# tar zxvf /mnt/sd1/rootfs-LRFS.tar.gz
[root@COMX-P2020 sd2]# cd /root
[root@COMX-P2020 root]# cp /mnt/sd1/comx.* /mnt/sd2/boot/.
[root@COMX-P2020 root]# cp /mnt/sd1/uImage /mnt/sd2/boot/.
 [root@COMX-P2020 root]# ls -al /mnt/sd2
total 100
drwxrwxr-x 22 user user  4096 Sep 28 16:41 .
drwxr-xr-x 25 root root  4096 Sep 28 08:20 ..
drwxr-xr-x  2 user user  4096 Sep 28 16:41 bin
drwxr-xr-x  2 user user  4096 Sep 28 16:42 boot
drwxr-xr-x  2 user user  4096 Sep 28 16:41 boot_format
drwxr-xr-x  2 user user  4096 Sep 28 16:41 dev
drwxrwxr-x 11 user user  4096 Sep 28 16:40 etc
```

```
drwxr-xr-x  3 user user  4096 Sep 28 16:36 home
drwxr-xr-x 10 user user  4096 Sep 28 16:36 include
drwxr-xr-x  5 user user  4096 Sep 28 16:36 lib
lrwxrwxrwx  1 user user    11 Sep 28 16:41 linuxrc -> bin/busybox
drwx------  2 root root 16384 Sep 28 16:20 lost+found
drwxr-xr-x  6 user user  4096 Sep 28 16:40 man
drwxr-xr-x 18 user user  4096 Sep 28 16:36 mnt
drwxr-xr-x  3 user user  4096 Sep 28 16:40 opt
drwxr-xr-x  2 user user  4096 Jan 14  2010 proc
drwxrwxr-x  5 user user  4096 Sep 28 16:41 root
drwxr-xr-x  2 user user  4096 Sep 28 16:40 sbin
drwxr-xr-x  3 user user  4096 Sep 28 16:41 share
drwxr-xr-x  2 user user  4096 Jan 14  2010 sys
drwxrwxr-x  2 user user  4096 Sep 17 01:39 tmp
drwxrwxr-x 15 user user  4096 Sep 28 16:40 usr
drwxr-xr-x 11 user user  4096 Sep 28 16:40 var
 [root@COMX-P2020 root]# ls -al /mnt/sd2/boot/COMX.*
-rwxr-xr-x 1 root root    9824 Sep 28 16:42 /mnt/sd2/boot/COMX.dtb
-rwxr-xr-x 1 root root 3416476 Sep 28 16:42
/mnt/sd2/boot/COMX.uImage
[root@COMX-P2020 root]#
```

7. Umount the SD to ensure all the data has been saved to SD Card.

```
[root@COMX-P2020 root]# umount /dev/mmcblk0p1
[root@COMX-P2020 root]# umount /dev/mmcblk0p2
```

8. Add U-boot code to the SD Card.

   - This step is not necessary if the user doesn't want to boot from the SD Card.
   - To boot from the SD Card, the user must make sure that P2020 must be version 2.0 or above or it can not boot from the SD Card.
   - To boot from the SD Card, the user must fdisk and format the partition at SD Card as above steps at first. The first partition of SD must be FAT16, and its size must less than 2GB.
   - Please run following command:

```
[root@COMX-P2020 root]# umount /dev/mmcblk0p1
[root@COMX-P2020 root]# umount /dev/mmcblk0p2
[root@COMX-P2020 root]# cd /boot_format/
```

```
[root@COMX-P2020 boot_format]# scp
<user>@<tftp_serverip>:/local/tftpboot/COMX-P2020/current/u-boot-
sd.bin .
[root@COMX-P2020 boot_format]# make
gcc -Wall -I. -O2  -c boot_format.c -o boot_format.o
gcc   boot_format.o -o boot_format
[root@COMX-P2020 boot_format]# ./boot_format config_sram.dat u-
boot-sd.bin -sd /dev/mmcblk0
Congratulations! It is done successfully.
```

6.4.2 Starting Kernel with Inflated ext2FS

By default the SD Card will contain inflated ext2FS on partition 2. This can be used to do a hard disk boot from the SD Card.

To start kernel with inflated ext2FS, please run the command at U-boot prompt:

```
=> run sdboot
Device: FSL_ESDHC
Manufacturer ID: 1b
OEM: 534d
Name: 00000
Tran Speed: 25000000
Rd Block Len: 512
SD version 2.0
High Capacity: No
Capacity: 2002780160
Bus Width: 4-bit
Loading file "/boot/uImage" from mmc device 0:2 (xxa2)
3416506 bytes read
Loading file "/boot/comx.dtb" from mmc device 0:2 (xxa2)
9824 bytes read
WARNING: adjusting available memory to 30000000
## Booting kernel from Legacy Image at 01000000 ...
   Image Name:   Linux-2.6.32
   Created:      2010-12-09   6:51:04 UTC
   Image Type:   PowerPC Linux Kernel Image (gzip compressed)
   Data Size:    3416442 Bytes =  3.3 MB
```

```
   Load Address: 00000000
   Entry Point:  00000000
   Verifying Checksum ... OK
## Flattened Device Tree blob at 00c00000
   Booting using the fdt blob at 0xc00000
   Uncompressing Kernel Image ... OK
WARNING: could not find compatible node fsl,p1020-immr: FDT_ERR_NOTFOUND.
Using P2020 RDB machine description
Memory CAM mapping: 256/256/256 Mb, residual: 1280Mb
Linux version 2.6.32 (ec7987@cncdebaobs04.emrsn.org) (gcc version 4.3.2
(GCC) ) #1 SMP Thu Dec 9 14:51:54 CST 2010
CPU maps initialized for 1 thread per core
bootconsole [udbg0] enabled
setup_arch: bootmem
mpc85xx_rdb_setup_arch()
Found FSL PCI host bridge at 0x00000000ffe08000. Firmware bus number: 0-
>255
PCI host bridge /pcie@ffe08000  ranges:
 MEM 0x0000000080000000..0x000000009fffffff -> 0x0000000080000000
  IO 0x00000000ffc10000..0x00000000ffc1ffff -> 0x0000000000000000
/pcie@ffe08000: PCICSRBAR @ 0xfff00000
Found FSL PCI host bridge at 0x00000000ffe09000. Firmware bus number: 0->1
PCI host bridge /pcie@ffe09000  ranges:
 MEM 0x00000000a0000000..0x00000000bfffffff -> 0x00000000a0000000
  IO 0x00000000ffc20000..0x00000000ffc2ffff -> 0x0000000000000000
/pcie@ffe09000: PCICSRBAR @ 0xfff00000
Found FSL PCI host bridge at 0x00000000ffe0a000. Firmware bus number: 0-
>255
PCI host bridge /pcie@ffe0a000  ranges:
 MEM 0x00000000c0000000..0x00000000dfffffff -> 0x00000000c0000000
  IO 0x00000000ffc30000..0x00000000ffc3ffff -> 0x0000000000000000
/pcie@ffe0a000: PCICSRBAR @ 0xfff00000
MPC85xx RDB board from Freescale Semiconducto
……
```

```
        Welcome to the LTIB Embedded Linux Environment

!!!!! WARNING !!!!!!!

The default password for the root account is: root
please change this password using the 'passwd' command
and then edit this message (/etc/issue) to remove this message

COMX-P2020 login: root
Password:
[root@COMX-P2020 root]#
```

6.4.3 Starting Kernel with Ramdisk Using FAT16 Partition

By default, the SD Card will contain FAT16 partition on partition 1, as shown in the above table.

To boot from this FAT16 partition, execute the following command:

```
=> run sdfatboot
Device: FSL_ESDHC
Manufacturer ID: 1b
OEM: 534d
Name: 00000
Tran Speed: 25000000
Rd Block Len: 512
SD version 2.0
High Capacity: No
Capacity: 2002780160
Bus Width: 4-bit
reading uImage

3416506 bytes read
reading comx.dtb

9824 bytes read
reading rootfs-dev.ext2.img
```

```
110443332 bytes read
WARNING: adjusting available memory to 30000000
## Booting kernel from Legacy Image at 01000000 ...
   Image Name:   Linux-2.6.32
   Created:      2010-12-09   6:51:04 UTC
   Image Type:   PowerPC Linux Kernel Image (gzip compressed)
   Data Size:    3416442 Bytes =  3.3 MB
   Load Address: 00000000
   Entry Point:  00000000
   Verifying Checksum ... OK
## Loading init Ramdisk from Legacy Image at 02000000 ...
   Image Name:   Blackadder ext2 ramdisk rootfs
   Created:      2010-12-09   6:54:03 UTC
   Image Type:   PowerPC Linux RAMDisk Image (gzip compressed)
   Data Size:    110443268 Bytes = 105.3 MB
   Load Address: 00000000
   Entry Point:  00000000
   Verifying Checksum ... OK
## Flattened Device Tree blob at 00c00000
   Booting using the fdt blob at 0xc00000
   Uncompressing Kernel Image ... OK
   Loading Ramdisk to 296ac000, end 2ffffb04 ... OK
WARNING: could not find compatible node fsl,p1020-immr: FDT_ERR_NOTFOUND.
Using P2020 RDB machine description
Memory CAM mapping: 256/256/256 Mb, residual: 1280Mb
Linux version 2.6.32 (ec7987@cncdebaobs04.emrsn.org) (gcc version 4.3.2
(GCC) ) #1 SMP Thu Dec 9 14:51:54 CST 2010
Found initrd at 0xe96ac000:0xeffffb04
CPU maps initialized for 1 thread per core
bootconsole [udbg0] enabled
setup_arch: bootmem
mpc85xx_rdb_setup_arch()
.........................
```

```
        Welcome to the LTIB Embedded Linux Environment

!!!!! WARNING !!!!!!!

The default password for the root account is: root
please change this password using the 'passwd' command
and then edit this message (/etc/issue) to remove this message

COMX-P2020 login: root
Password:
[root@COMX-P2020 root]#
```

Linux Deployment

7.1 I2C Driver

The Linux has already included the driver of I2C bus. The device can be accessed through the file: /dev/i2c-0 /dev/i2c-1

```
[root@COMX-P2020 /]# ls /dev/i2c*
/dev/i2c-0  /dev/i2c-1
```

7.2 Detect I2C Device

The application "i2cdetect" can be used to detect the devices on the i2c bus.

- To Detect the devices on I2C bus 0:

```
[root@COMX-P2020 /]# /usr/sbin/i2cdetect 0
WARNING! This program can confuse your I2C bus, cause data
loss and worse!
I will probe file /dev/i2c-0.
I will probe address range 0x03-0x77.
Continue? [Y/n] y
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: 50 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: 70 -- -- -- -- -- -- --
[root@COMX-P2020 /]#
```

- To Detect the devices on I2C bus 1:

```
[root@COMX-P2020 /]# /usr/sbin/i2cdetect 1
WARNING! This program can confuse your I2C bus, cause data
loss and worse!
I will probe file /dev/i2c-1.
I will probe address range 0x03-0x77.
Continue? [Y/n] y
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
```

```
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- 1b -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- 33 -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- 48 -- -- -- -- -- -- --
50: -- -- -- 53 -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- UU -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
[root@COMX-P2020 /]#
```

7.3 Dump I2C Device

- To dump the registers on the I2C Device, please use the application: i2cdump

```
[root@COMX-P2020 /]# /usr/sbin/i2cdump --help
Error: Unsupported option "--help"!
Usage: i2cdump [-f] [-y] [-r first-last] I2CBUS ADDRESS
[MODE [BANK [BANKREG]]]
  I2CBUS is an integer or an I2C bus name
  ADDRESS is an integer (0x03 - 0x77)
  MODE is one of:
    b (byte, default)
    w (word)
    W (word on even register addresses)
    s (SMBus block)
    i (I2C block)
    c (consecutive byte)
    Append p for SMBus PEC
[root@COMX-P2020 /]#
```

- To dump the content of at24c02, which i2c bus is 0, and i2c address is 0x50, please run the application as below:

```
[root@COMX-P2020 /]# i2cdump 0 0x50 b
WARNING! This program can confuse your I2C bus, cause data
loss and worse!
I will probe file /dev/i2c-0, address 0x50, mode byte
Continue? [Y/n] y
```

```
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
0123456789abcdef
00: 4e 58 49 44 31 32 33 34 35 36 37 38 39 30 31 00
NXID12345678901.
10: 31 32 33 34 00 10 04 30 16 50 00 ff 00 00 00 00
1234.??0?P......
20: ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
................
30: ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
................
40: 03 ff 00 05 9f ff a1 01 00 05 9f ff a1 02 00 05
?..??.??.??.??.?
50: 9f ff a1 03 ff ff ff ff ff ff ff ff ff ff ff ff
?.??............
60: ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
................
70: ff ff 5a 31 4c 3d ff ff ff ff ff ff ff ff ff ff
..Z1L=..........
80: ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
................
90: ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
................
a0: ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
................
b0: ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
................
c0: ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
................
d0: ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
................
e0: ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
................
f0: 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a
ZZZZZZZZZZZZZZZZ
```

7.4 Get Registers of I2C Device

- The application "i2cget" can be used to get registers of i2c device:

```
[root@COMX-P2020 /]# ./usr/sbin/i2cget --help
Error: Unsupported option "--help"!
Usage: i2cget [-f] [-y] I2CBUS CHIP-ADDRESS [DATA-ADDRESS
[MODE]]
  I2CBUS is an integer or an I2C bus name
  ADDRESS is an integer (0x03 - 0x77)
  MODE is one of:
    b (read byte data, default)
    w (read word data)
    c (write byte/read byte)
    Append p for SMBus PEC
[root@COMX-P2020 /]#
```

- To Read the first byte of AT24C02, run the command:

```
[root@COMX-P2020 /]# ./usr/sbin/i2cget 0 0x50 0
WARNING! This program can confuse your I2C bus, cause data
loss and worse!
I will read from device file /dev/i2c-0, chip address 0x50,
data address
0x00, using read byte data.
Continue? [Y/n] y
0x4e
[root@COMX-P2020 /]#
```

7.5 Get Temperature of LM75

```
[root@P2020RDB lib]# /usr/sbin/i2cget 1 0x48 0
WARNING! This program can confuse your I2C bus, cause data loss and worse!
I will read from device file /dev/i2c-1, chip address 0x48, data address
0x00, using read byte data.
Continue? [Y/n] y
0x2d
[root@P2020RDB lib]#
```

The 0x2d means that current temperature is 45 degree Celsius.

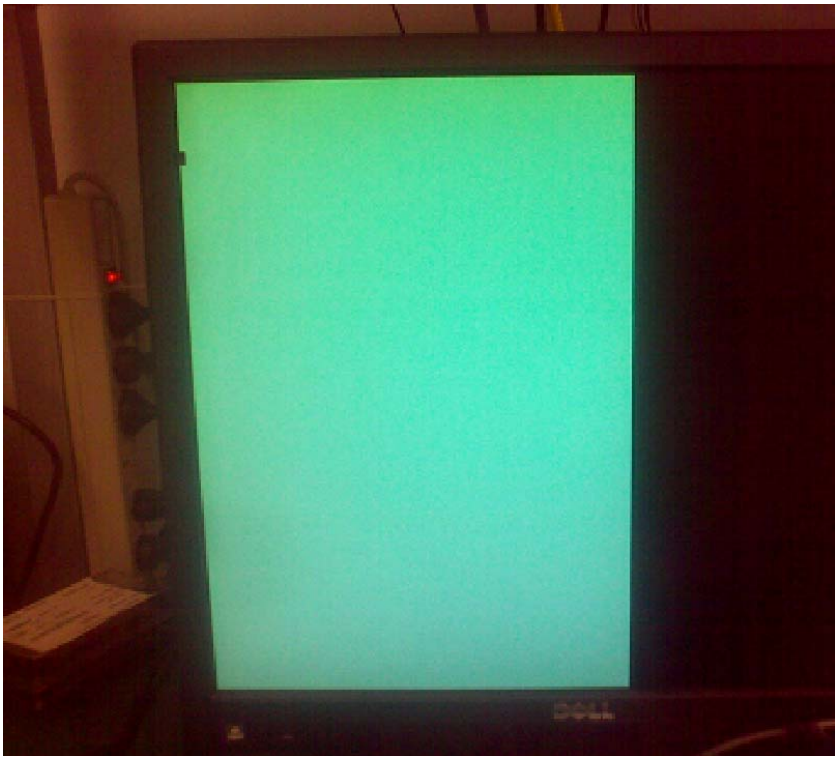
7.6 Frame buffer of XGI

The file system of Linux has provided Frame buffer test application to verify that the XGI can work well.

```
[root@COMX-P2020 root]# /usr/private/framebuffer/fbtest 50
res = 1024 * 768 * 16 bit
line_length = 2048
```

Then we can check the output of VGA that the pixel will be changed to another color one by one. The process will take 50 seconds.

Figure 7-1 Output for Framebuffer Test Program



7.7 MiniGUI Example

The file system of Linux has provided Minigui test application to verify that the XGI, USB mouse and USB keyboard can work well at COMX-P2020.

1. Insert USB mouse and USB keyboard to the COMX-P2020, this is a must for this example.
2. Run the Minigui sample and check the output of VGA to verify that all devices can work well at the board.

```
[root@COMX-P2020 root]# cd /usr/private/mg-samples-1.6.10/src
[root@COMX-P2020 src]# ./mywins
```

Figure 7-2 Output for MiniGUI Example