AXIOMTEK

IFB125

Linux

Software User's Manual

Disclaimers

This manual has been carefully checked and believed to contain accurate information. Axiomtek Co., Ltd. assumes no responsibility for any infringements of patents or any third party's rights, and any liability arising from such use.

Axiomtek does not warrant or assume any legal liability or responsibility for the accuracy, completeness or usefulness of any information in this document. Axiomtek does not make any commitment to update the information in this manual.

Axiomtek reserves the right to change or revise this document and/or product at any time without notice.

No part of this document may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of Axiomtek Co., Ltd.

Trademarks Acknowledgments

Axiomtek is a trademark of Axiomtek Co., Ltd.

Windows® is a trademark of Microsoft Corporation.

Other brand names and trademarks are the properties and registered brands of their respective owners.

December 2018, Version A2

Printed in Taiwan

Table of Contents

Chapter 1 Introduction

The ultra-compact IFB125 supports the low power RISC-based module (i.MX6UL) processor and is designed to operate at an extended temperature range of -40°C to +70°C in various environments. Featuring multiple built-in serial ports, high-speed LANs, and USB 2.0 ports, the IFB125 offers fast and efficient data computing, communication, and acquisition. Its digital I/O features provide users with convenient connectivity between digital devices and its compact size with Din-rail mounting allows for easy installation and control.

This user's manual is intended for the embedded Linux preinstalled in the IFB125. The embedded Linux is derived from Linux Yocto Board Support Package, which is based on Linux Kernel 3.14.52 and our hardware patches for use with the IFB125.

Software structure

The embedded Linux image is preinstalled on an eMMC Flash memory, which is partitioned and formatted to accommodate boot loader, kernel, and root filesystem. It adopts the standard Linux architecture to allow users to easily develop and deploy application software that follows the Portable Operating System Interface (POSIX).

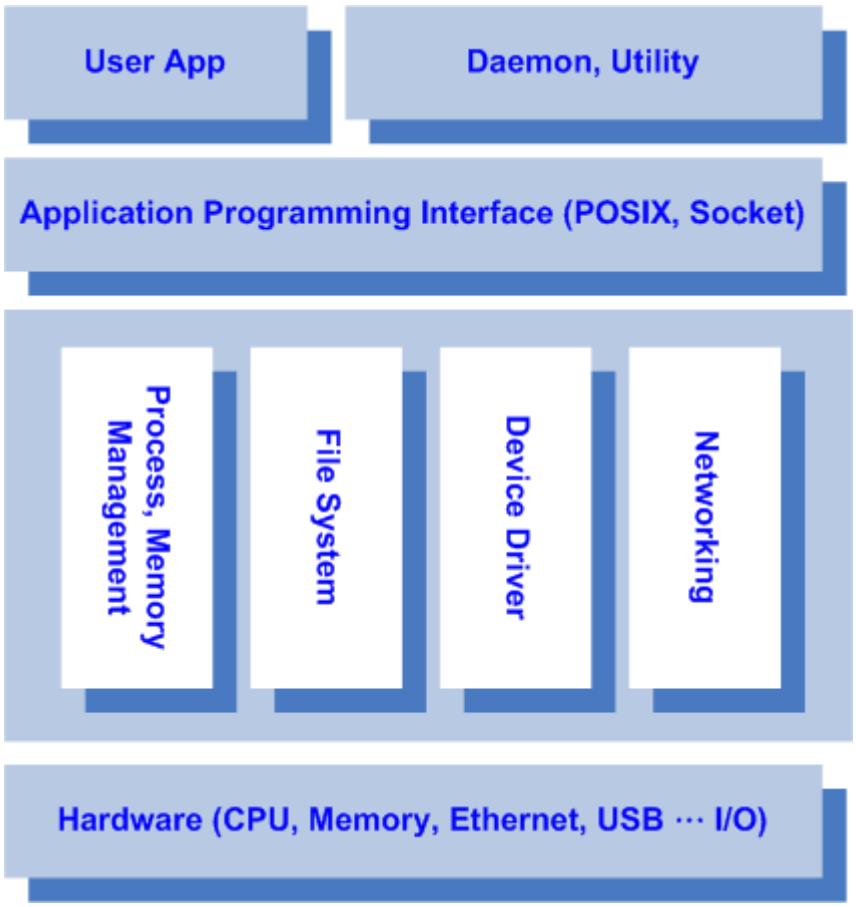
The IFB125 also includes 'librsb10x.so' shared library to facilitate user configuration in monitoring and controlling I/O devices such as DIO, Watchdog Timer, and COM.

In addition to ext3 and ext4 file systems, this embedded Linux kernel is compiled with support for NFS, including server-side, client-side functionality, and 'Root file system on NFS'. Using an NFS root mount provides the advantages including :

- The root file system is not size-restricted by the device's storage like Flash memory.
- Changes made to application files during development are immediately available to the target device.

In order to illustrate the connectivity structure of the device, this image includes the most popular internet protocols, servers and utilities, not only making it easy to download/upload files (Linux kernel, application program, etc) and debug, but also facilitating communication to the outside world via Ethernet, WiFi, and 3G.

For the convenience of operating the embedded Linux, this image includes a number of popular packages such as busybox, udev, etc.



1.1 Specifications

- **OS: Linux**
  - Kernel: 3.14.52 (with NXP and Axiomtek's modified hardware patches)
- **Supported Protocol Types**
  - ICMP.
  - TCP/IP.
  - UDP, DHCP, Telnet, HTTP, HTTPS, SSL, SMTP, NTP, DNS, PPP, PPPoE, FTP, TFTP, NFS.
- **Shell**
  - Bash
- **Supported storage formats**
  - FAT32 /FAT/EXT2/EXT3/EXT4
- **BSP: IFB125-LINUX-bsp**
  - AxTools
  - Image
  - Yocto patches
  - Toolchain
  - mfgtools_for_windows

- **Daemons**
  - Telnetd: Telnet server daemon
  - FTPD: FTP server daemon

- **Utilities**
  - Telnet: Telnet client program
  - FTP: FTP client program
  - TFTP: Trivial File Transfer Protocol client

- **Packages**
  - **Busybox(1.23.1)**: A small collection of standard Linux command-line utilities
  - **udev**: A device manager for Linux kernel
  - **dosfstools** : Utilities for making and checking MS-DOS FAT file system
  - **e2fsprogs**: A set of utilities for maintaining the ext2, ext3 and ext4 file systems
  - **ethtool**: A Linux command for displaying or modifying the Network Interface Controller (NIC) parameters
  - **i2c-tools** : A heterogeneous set of I2C tools for Linux
  - **procps** : Utilities to report the state of the system, including the states of running processes, amount of memory
  - **wireless-tools**: A package of Linux commands (simple text-based utilities/tools) intended to support and facilitate the configuration of wireless devices using the Linux Wireless Extension

- **Development Environment**
  - Host OS/ development OS: Ubuntu 14.04 LTS 32/64bit
    kernel: version: 4.2.0-42
  - machine running Ubuntu, the minimum hard disk space required is about 50 GB for the X11 backend. It is recommended that a minimum of 120 GB is provided in order to have sufficient space to compile all backends together.
  - Toolchain/ cross compiler: ARM, gcc-4.9.2 (Yocto project 1.8.1 Fido)

- **HW's Lib (Hardware's Library)**
  - **Digital I/O**
    - Read digital input
    - Write digital output
  - **COM**
    - RS-232/422/485 mode setting(Default RS232)
  - **SPI**
    - User defined
  - **I2C**
    - Read i2c device
    - Write i2c device
  - **Watch Dog Timer**
    - Enable Watch Dog Timer
    - Set Timer
  - **WiFi (Optional)**
    - Use Wi-Fi module WPEQ-160ACN
  - **3G (Optional)**
    - Use 3G module Quectel UC20

- **4G (Optional)**
  - Use 4G module Sierra MC7304、LARA-R211、LARA-R280

- **Relay**
  - Set relay high or low.

Note

All specifications and images are subject to change without notice..
http://www.axiomtek.com/Default.aspx?MenuId=Products&FunctionId=ProductView&ItemId=24247&upcat=134
.

Command definition:

Command	Definition	Example
=>	**U-Boot**	Ex: **=>** setenv ipaddr 192.168.1.103 Meaning: U-Boot setenv ipaddr 192.168.1.103
~$	**Host PC**	Ex: **~$** sudo apt-get install subversion Meaning: To command sudo apt-get install subverhsion on host PC
~#	**Target (IFB125):**	Ex: ~# /etc/run_rescue Meaning: To command /etc/run_rescue on IFB125

Chapter 2
Getting Started

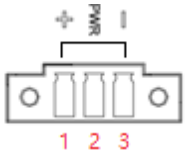
2.1 Connecting the IFB125

The power

Please check power as below:

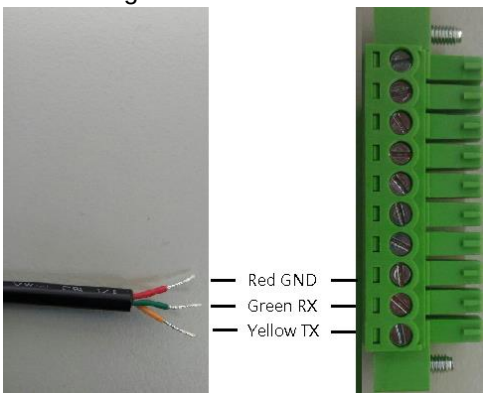
1. DC input range 9~48V
2. DC Terminal Block

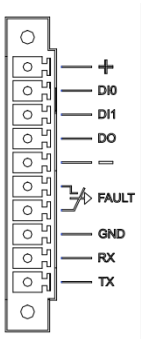
Pin	DC Signal Name
1	Power+
2	N/A
3	Power-



Console Port

- You can use the console port for user debug settings. . Locate the TB10 pins for the console port as illustrated by the table below.
- Connecting to the DIO terminal Block

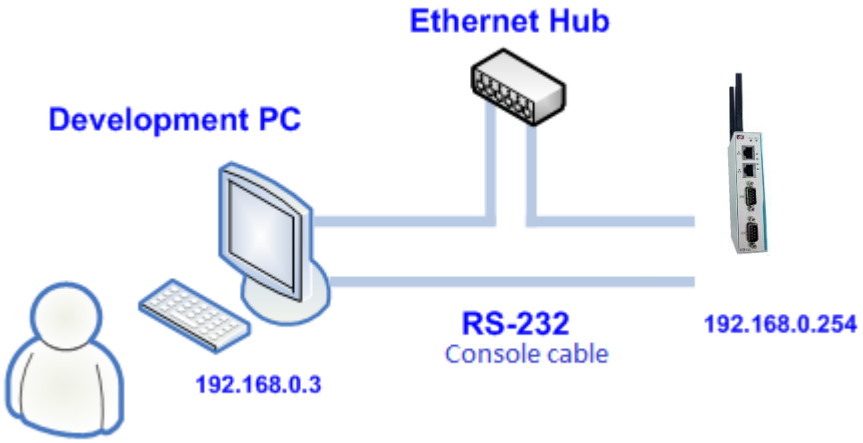




DIO Terminal Block

TB10 Pin No.	Signal name	Meaning
1	COM+	Plus Common for DIO
2	DI0	Digital Input
3	DI1	
4	DO	Digital Output
5	COM-	Minus Common for DIO
6	Relay+	Relay Out
7	Relay-	
8	GND	For Console Port
9	Console RX	
10	Console TX	

You can connect the IFB125 to a personal computer (PC) through either the Serial RS-232 console or SSH over Ethernet:



Note ***If necessary, you can download the BSP support package from Axiomtek's website listed below.***
http://www.axiomtek.com/Default.aspx?MenuId=Products&FunctionId=ProductView&ItemId=24247&upcat=134

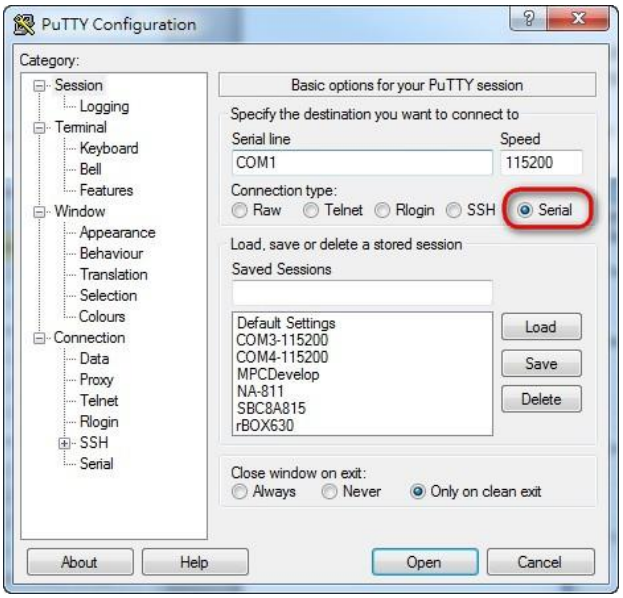
2.1.1 Serial Console

The serial console is a convenient interface for connecting the IFB125 to a desktop PC. Before configuring the IFB125, ensure that your PC has connected to the IFB125 with a console cable.
Please set the system as follows:

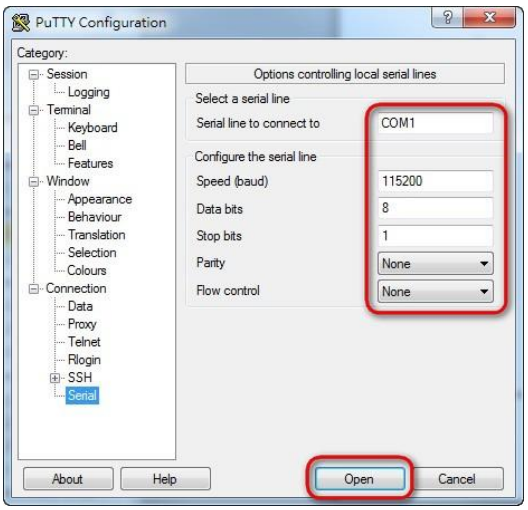
Baudrate: 115200 bps
Parity: None
Data bits: 8
Stop bit: 1
Flow Control: None

You need to configure PuTTY in order to set up and link to the IFB125. Follow the step-by-step instructions below to complete PuTTY configuration.

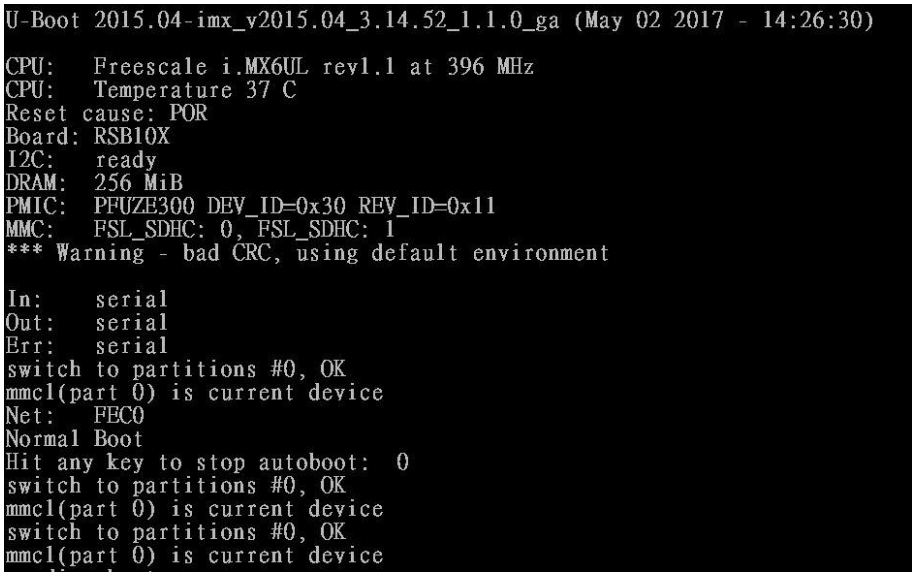
1. Open PuTTY and choose 'Serial' as the connection type.



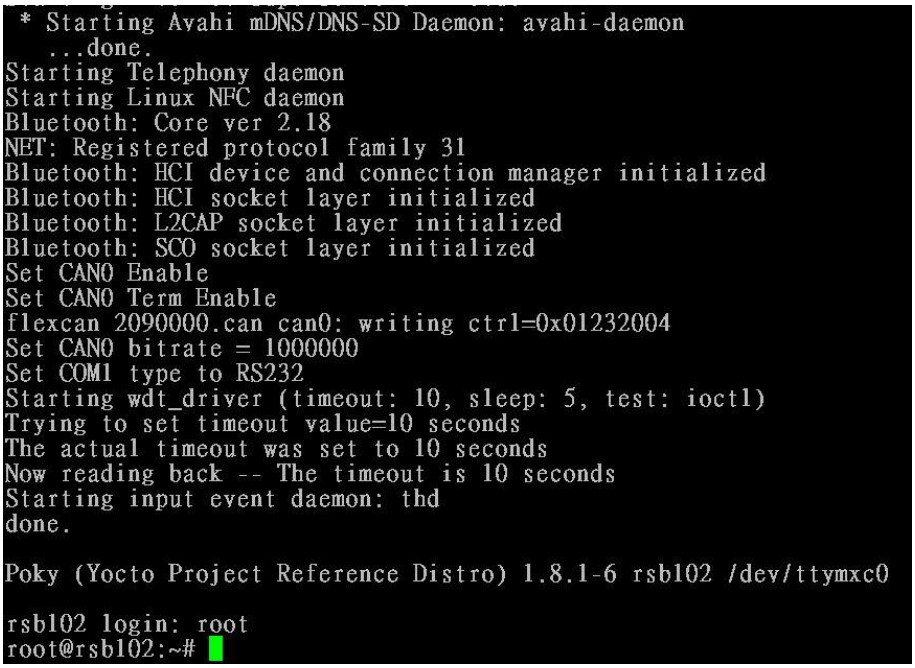
2. Configure the serial port correctly (see image below). Click 'Open' and power on the IFB125.



3. The data of the Bootloader default booting system on eMMC appears.



4. If connection is established successfully, you should see the following image. To log in, enter 'root' (without password).



2.1.2 SSH over Ethernet

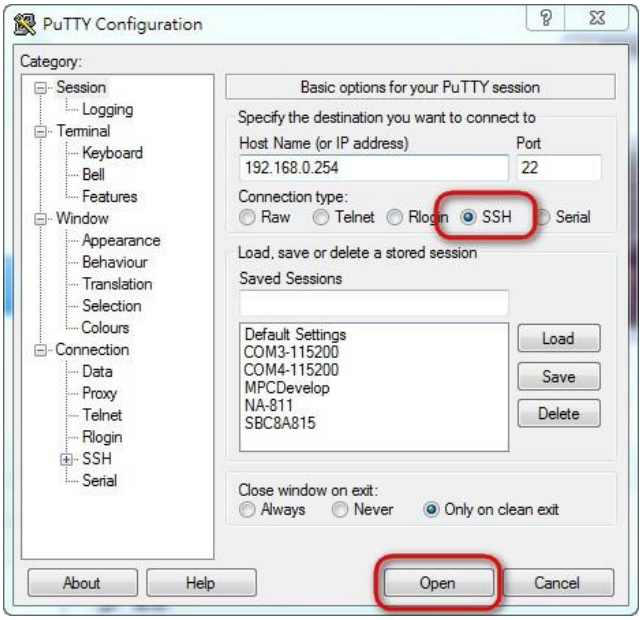
Follow the steps below to connect the IFB125 to a PC over Ethernet under the Windows® and Linux environments respectively.

Note

IFB125 LAN2 default IP address is 192.168.0.254.

For Windows® users:

1. Use PuTTY to set up and link. Open PuTTY and choose 'SSH' as the connection type. Then set the IP address to 192.168.0.254 and click 'Open'.



2. If connection is established successfully, you should see the following image.

3. To log in to the IFB125, enter 'root' (with no password).

For Linux users:

1. Open terminal and enter an 'ssh' command.
   ~$ ssh -l root 192.168.0.254

```
louis@ubuntu:~$ ssh -l root 192.168.0.254
```

2. The following data appears after the connection is established successfully.

```
louis@ubuntu:~$ ssh -l root 192.168.0.254
The authenticity of host '192.168.0.254 (192.168.0.254)' can't be established.
ECDSA key fingerprint is d3:12:94:ec:e4:2a:a4:42:15:90:1b:e1:00:26:48:8a.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.0.254' (ECDSA) to the list of known hosts.
Last login: Tue Sep 16 18:43:52 2014 from louish7697.local
root@axiomtek:~#
```

2.2 How to Develop a Sample Program

In this section, learn how to develop a sample program for the IFB125 with the following step-by-step instructions. The sample program is named 'hello.c'.

1. Create a directory for IFB125 BSP by copying "IFB125-Linux-bsp-x.x.x.tar.gz" to the item shown below:

```
~$ mkdir project
~$ cd project
ryan@axiomtek:~/project$ ls
IFB125 Linux V.1.0.1  IFB125 Linux V.1.0.1.zip
```

2. After extracting the file, you will find a directory IFB125 Linux V.x.x.x

```
ryan@axiomtek:~/project$ cd IFB125\ Linux\ V.1.0.1/
ryan@axiomtek:~/project/IFB125 Linux V.1.0.1$ ls
ChangeLog.txt  IFB125-LINUX-bsp-V.1.0.1
ryan@axiomtek:~/project/IFB125 Linux V.1.0.1$ cd IFB125-LINUX-bsp-V.1.0.1/
ryan@axiomtek:~/project/IFB125 Linux V.1.0.1/IFB125-LINUX-bsp-V.1.0.1$ ls
AxTools  Image  mfgtools_for_windows  README.txt  Toolchain  Yocto patches
```


Note

AxTools: This directory includes a hardware driver and an API library
Image: This directory includes kernel, rootfilesystem
Yocto patches: This directory includes IFB125 hardware patches for Yocto Project 1.8.1.
Toolchain: This directory includes cross compiler toolchain build from Yocto Project 1.8.1.
README.txt: The documentation file of this BSP.

2.2.1 Install Yocto Toolchain

Before you develop and compile a sample program, you should install Yocto toolchain into the development PC. To install Yocto toolchain or refer to Chapter 5 Board Support Package to build the toolchain for IFB series.

1. To check your Ubuntu version on your host PC.

```
~$ uname -m
```

Ubuntu **32-bit** (i686):

```
louis@ubuntu:~$ uname -m
i686
louis@ubuntu:~$
```

Ubuntu **64-bit** (x86_64):

```
louis@ubuntu:~$ uname -m
x86_64
louis@ubuntu:~$
```

2. Copy the toolchain script to the home directory.
   i686 for 32-bit machines or x86_64 for 64-bit machines.

```
ryan@Ubuntu:~/project/IFB112-Linux-bsp-1.0.0/IFB112-Linux-bsp-1.0.0/Toolchain$ ls
32-bit  64-bit
```

3 Execute the toolchain script and press Enter to install to the default directory.

32-bit machines:

~$ bash poky-glibc-i686-meta-toolchain-cortexa7hf-vfp-neon-toolchain-1.8.1.sh

```
ryan@Ubuntu:~/project/IFB112-Linux-bsp-1.0.0/IFB112-Linux-bsp-1.0.0/Toolchain$ cd 32-bit/
ryan@Ubuntu:~/project/IFB112-Linux-bsp-1.0.0/IFB112-Linux-bsp-1.0.0/Toolchain/32-bit$ ls
poky-glibc-i686-meta-toolchain-cortexa7hf-vfp-neon-toolchain-1.8.1.sh
ryan@Ubuntu:~/project/IFB112-Linux-bsp-1.0.0/IFB112-Linux-bsp-1.0.0/Toolchain/32-bit$ bash poky-glibc-i686
-meta-toolchain-cortexa7hf-vfp-neon-toolchain-1.8.1.sh
Enter target directory for SDK (default: /opt/poky/1.8.1):
```

64-bit machines:

~$bash poky-glibc-x86_64-meta-toolchain-cortexa7hf-vfp-neon-toolchain-1.8.1.sh

```
ryan@Ubuntu:~/project/IFB112-Linux-bsp-1.0.0/IFB112-Linux-bsp-1.0.0/Toolchain/64-bit$ ls
poky-glibc-x86_64-meta-toolchain-cortexa7hf-vfp-neon-toolchain-1.8.1.sh
ryan@Ubuntu:~/project/IFB112-Linux-bsp-1.0.0/IFB112-Linux-bsp-1.0.0/Toolchain/64-bit$ bash poky-glibc-x86_
64-meta-toolchain-cortexa7hf-vfp-neon-toolchain-1.8.1.sh
Enter target directory for SDK (default: /opt/poky/1.8.1):
```

4 Check the directory.

```
ryan@Ubuntu:~/project/IFB112-Linux-bsp-1.0.0/IFB112-Linux-bsp-1.0.0/Toolchain/64-bit$ ls
poky-glibc-x86_64-meta-toolchain-cortexa7hf-vfp-neon-toolchain-1.8.1.sh
ryan@Ubuntu:~/project/IFB112-Linux-bsp-1.0.0/IFB112-Linux-bsp-1.0.0/Toolchain/64-bit$ bash poky-glibc-x86_
64-meta-toolchain-cortexa7hf-vfp-neon-toolchain-1.8.1.sh
Enter target directory for SDK (default: /opt/poky/1.8.1):
You are about to install the SDK to "/opt/poky/1.8.1". Proceed[Y/n]?y
```

5 Wait for installation.

```
ryan@Ubuntu:~/project/IFB112-Linux-bsp-1.0.0/IFB112-Linux-bsp-1.0.0/Toolchain/64-bit$ ls
poky-glibc-x86_64-meta-toolchain-cortexa7hf-vfp-neon-toolchain-1.8.1.sh
ryan@Ubuntu:~/project/IFB112-Linux-bsp-1.0.0/IFB112-Linux-bsp-1.0.0/Toolchain/64-bit$ bash poky-glibc-x86_
64-meta-toolchain-cortexa7hf-vfp-neon-toolchain-1.8.1.sh
Enter target directory for SDK (default: /opt/poky/1.8.1):
You are about to install the SDK to "/opt/poky/1.8.1". Proceed[Y/n]?y
Extracting SDK...done
```

6 Installation is completed.

```
ryan@Ubuntu:~/project/IFB112-Linux-bsp-1.0.0/IFB112-Linux-bsp-1.0.0/Toolchain/64-bit$ ls
poky-glibc-x86_64-meta-toolchain-cortexa7hf-vfp-neon-toolchain-1.8.1.sh
ryan@Ubuntu:~/project/IFB112-Linux-bsp-1.0.0/IFB112-Linux-bsp-1.0.0/Toolchain/64-bit$ bash poky-glibc-x86_
64-meta-toolchain-cortexa7hf-vfp-neon-toolchain-1.8.1.sh
Enter target directory for SDK (default: /opt/poky/1.8.1):
You are about to install the SDK to "/opt/poky/1.8.1". Proceed[Y/n]?y
Extracting SDK...done
Setting it up...done
SDK has been successfully set up and is ready to be used.
```

2.2.2 Setting Up the Cross-Development Environment

Before you can develop using the cross-toolchain, you need to set up the cross-development environment, and then you can find this script in the directory you have chosen for installation.

1. To set up the cross-toolchain environment.

~$ source /opt/poky/1.8.1/environment-setup-cortexa7hf-vfp-neon-poky-linux-gnueabi

```
ryan@Ubuntu:~$ source /opt/poky/1.8.1/environment-setup-cortexa7hf-vfp-neon-poky-linux-gnueabi
```

2. Check whether the Cross-Development Environment is successfully set up. You will find the information below if setup is succesful.

~$ echo $CC

```
ryan@Ubuntu:~$ echo $CC
arm-poky-linux-gnueabi-gcc -march=armv7-a -mfloat-abi=hard -mfpu=neon -mtune=cortex-a7 --sysroot=/opt/poky
/1.8.1/sysroots/cortexa7hf-vfp-neon-poky-linux-gnueabi
```

2.2.3 Write and Compile Sample Program

1. Create a directory on your host PC.

~$ mkdir -p example
~$ cd example

```
ryan@Ubuntu:~$ mkdir -p example
ryan@Ubuntu:~$ cd example/
```

2. Use vim to edit hello.c.

~$ vim hello.c

```
#include<stdio.h>
int main()
{
    printf("hello world\n");
    return 0;
}
```

```
#include<stdio.h>
int main()
{
        printf("hello world\n");
        return 0;
}
~
~
~
```

3. To compile the program, enter the commands::

~$ $CC hello.c -o hello

```
ryan@Ubuntu:~/example$ $CC hello.c -o hello
```

4. After compiling, enter the following command and you will see the 'hello' execution file.

~$ ls -l

```
ryan@Ubuntu:~/example$ ls -l
total 16
-rwxrwxr-x 1 ryan ryan 9669  6月  1 15:02 hello
-rw-rw-r-- 1 ryan ryan   71  6月  1 15:02 hello.c
```

Check whether the ARM executable format is created successfully. You will see the information below if the format is successfully created.

~$ file hello

```
ryan@Ubuntu:~/example$ file hello
hello: ELF 32-bit LSB executable, ARM, EABI5 version 1 (SYSV), dynamically linked, interpreter /lib/ld-lin
ux-armhf.so.3, for GNU/Linux 2.6.32, BuildID[sha1]=03ffaa4ff511b9c7c92e32c78e22b5d51f1cea7b, not stripped
ryan@Ubuntu:~/example$
```

2.3 How to Put and Run a Sample Program

This section shows how to put the 'hello' program into the IFB125 and execute the program via FTP, a USB flash drive, and TFTP.

2.3.1 Via FTP

The IFB125 has a built-in FTP server. Users can put the 'hello' program into the IFB125 via FTP by following the steps below.

1. Enable FTPD daemon on the IFB125
   Use vi to create /etc/xinetd.d/ftpd file

   ~# vi /etc/xinetd.d/ftpd

   ```
   service ftp
   {
           port = 21
           disable = no
           socket_type = stream
           protocol = tcp
           wait = no
           user = root
           server = /usr/sbin/ftpd
           server_args = -w /home/root
   }
   ```

   ```
   service ftp
   {
           port = 21
           disable = no
           socket_type = stream
           protocol = tcp
           wait = no
           user = root
           server = /usr/sbin/ftpd
           server_args = -w /home/root
   }
   ~
   ```

2. Restart the FTP server on the IFB125

   ~# /etc/init.d/xinetd reload
   ~# /etc/init.d/xinetd restart

   ```
   root@axiomtek:~# /etc/init.d/xinetd reload
   Reloading internet superserver configuration: xinetd.
   root@axiomtek:~# /etc/init.d/xinetd restart
   Stopping internet superserver: xinetd.
   Starting internet superserver: xinetd.
   root@axiomtek:~#
   ```

3. To connect your host PC to IFB125, enter the command below.
   ~$ ftp 192.168.0.254 (username 'root' without password)

```
louis@ubuntu:~/project/example$ ftp 192.168.0.254
Connected to 192.168.0.254.
220 Operation successful
Name (192.168.0.254:louis): root
331 Please specify password
Password:
230 Operation successful
Remote system type is UNIX.
Using binary mode to transfer files.
```

4. Upload the "hello" program onto the IFB125 from your host PC.

   ftp> put hello

```
ftp> put hello
local: hello remote: hello
200 Operation successful
150 Ok to send data
226 Operation successful
9669 bytes sent in 0.00 secs (165655.8 kB/s)
ftp>
```

5. If the operation is successful on the IFB125, you can see the 'hello' program on IFB125's */home/root* directory.

```
root@axiomtek:~# ls
hello
root@axiomtek:~#
```

6. To change file permission for executable on IFB125, enter the command below.
   ~# chmod a+x hello

```
root@axiomtek:~# ls -l
-rw-r--r--    1 root     root          9669 Sep 16 18:40 hello
root@axiomtek:~# chmod a+x hello
root@axiomtek:~# ls -l
-rwxr-xr-x    1 root     root          9669 Sep 16 18:40 hello
root@axiomtek:~#
```

7. Run the 'hello' program on the IFB125.
   ~# ./hello

```
root@axiomtek:~# ./hello
hello world
root@axiomtek:~#
```

2.3.2 Via a USB Flash Drive

You can put the 'hello' program into the IFB125 via a USB flash drive. Please follow the instructions below.

IFB125 supports storage format FAT32 /FAT/EXT2/EXT3/EXT4

1. From the host PC, copy the 'hello' program to a USB flash drive.

2. Attach the USB flash drive to the IFB125.

3. ~# mkdir /media/sda1

```
root@axiomtek:~# mkdir /media/sda1
root@axiomtek:~#
```

4. ~# mount /dev/sda1 /media/sda1

```
root@axiomtek:~# mount /dev/sda1 /media/sda1/
root@axiomtek:~# ls /media/sda1/
hello
root@axiomtek:~#
```

5. ~# cp /media/sda1/hello /home/root

```
root@axiomtek:~# cp /media/sda1/hello /home/root/
root@axiomtek:~# ls
hello
root@axiomtek:~#
```

6. ~# chmod +x hello

```
root@axiomtek:~# ls -l
-rw-r--r--    1 root     root          9669 Sep 16 18:40 hello
root@axiomtek:~# chmod a+x hello
root@axiomtek:~# ls -l
-rwxr-xr-x    1 root     root          9669 Sep 16 18:40 hello
root@axiomtek:~#
```

7. ~# ./hello

```
root@axiomtek:~# ./hello
hello world
root@axiomtek:~#
```

2.3.3 Via TFTP

The Host Development System Installation already has a TFTP server installed. You can put the 'hello' program into the IFB125 via TFTP. Please follow the instructions below.

1. Refer to section 5.1.1 step 4. "Install and configure the TFTP server" for installation and setup of your TFTP:

2. To copy the "hello" program to the "tftpboot" folder in your host PC, enter the command below:
   ~$ cp hello /tftpboot

```
louis@ubuntu:~/project/example$ ls
hello  hello.c
louis@ubuntu:~/project/example$ cp hello /tftpboot/
louis@ubuntu:~/project/example$ ls /tftpboot/
hello
louis@ubuntu:~/project/example$
```

3. To enter the following command on the IFB125:
   ~# tftp -g -r hello 192.168.0.3 (tftp server IP depend on host PC's IP)

```
root@axiomtek:~# tftp -g -r hello 192.168.0.3
root@axiomtek:~# ls
hello
root@axiomtek:~#
```

4. To enter the following command on the IFB125:
   ~# chmod a+x hello

```
root@axiomtek:~# ls -l
-rw-r--r--    1 root     root          9669 Sep 16 18:40 hello
root@axiomtek:~# chmod a+x hello
root@axiomtek:~# ls -l
-rwxr-xr-x    1 root     root          9669 Sep 16 18:40 hello
root@axiomtek:~#
```

5. Run the 'hello' program on the IFB125:
   ~# ./hello

```
root@axiomtek:~# ./hello
hello world
root@axiomtek:~#
```

2.4 How to Recovery System

This section provides two methods for recovering the IFB125 system to default.

2.4.1 Via run_rescue System Script (under Linux System)

A recovery script is stored inside the /etc folder on the IFB125 Embedded Linux system. If you want to recover your system to factory default settings, follow the instructions below.

1. Run the run_rescue shell script:
   ~# /etc/run_rescue

```
root@axiomtek:~# /etc/run_rescue
Push RESCUE Script to u-boot
Reboot system to RESCUE/UPDATE system

Broadcast message from root@axiomtek (ttymxc0) (Tue Sep 16 20:01:32 2014):
The system is going down for reboot NOW!
INIT: Switching to runlevel: 6
INIT: Sending processes the TERM signal
logout
```

2. When the system reboots, it automatically switches to the rescue mode under u-boot, and starts recovery procedure. During this procedure, four custom LEDs will blink like a marquee.

3. After recovery procedure is completed, the system reboots again automatically, and the system status LED turns from the blinking mode to the always on mode.

2.4.2 Via rescue.scr Script (under u-boot)

Refer to section 5.2.2 for detailed information.

2.5 How to Update System

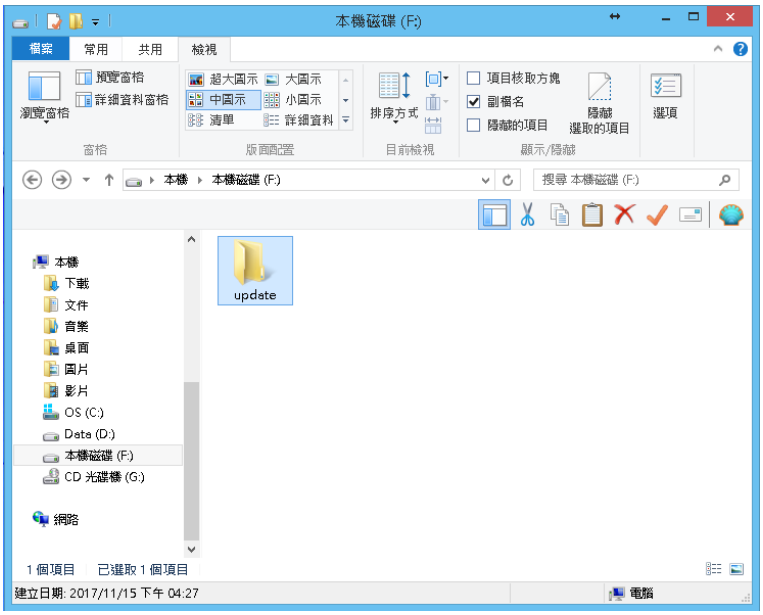
This section shows how to update the IFB125 using the recommended method below.

2.5.1 Via USB Flash Drive

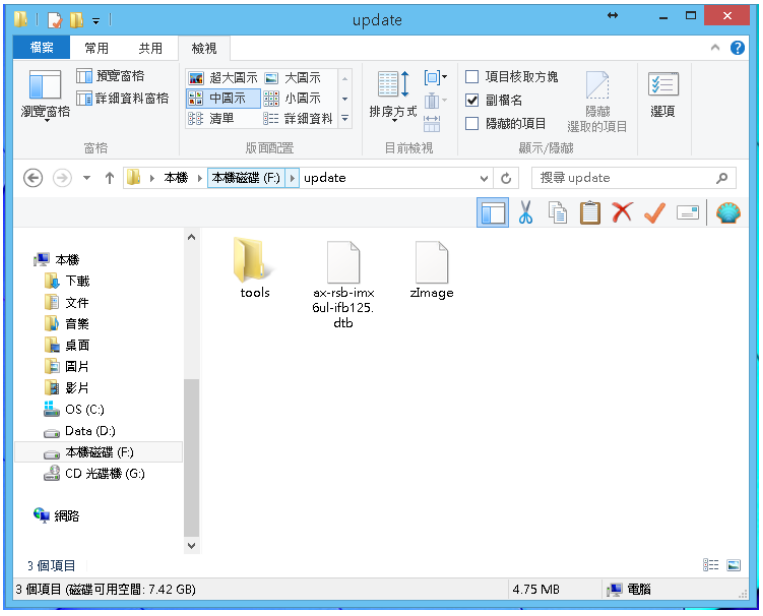
You can use a USB flash drive of DOS FAT32、EXT2、EXT3 or EXT4 formats, but an update folder must be stored on the first partition.

1. From the PC, copy files to a USB flash drive.

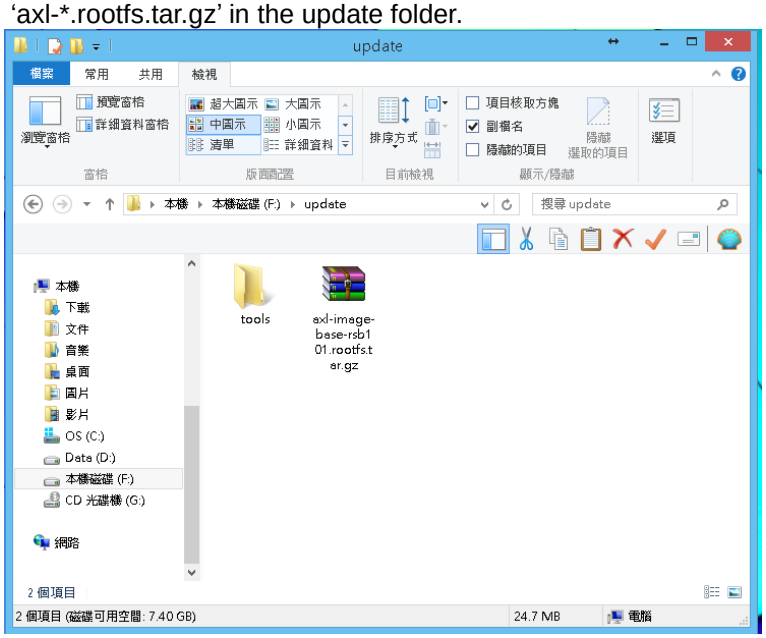
2. Create a folder named "update."
.



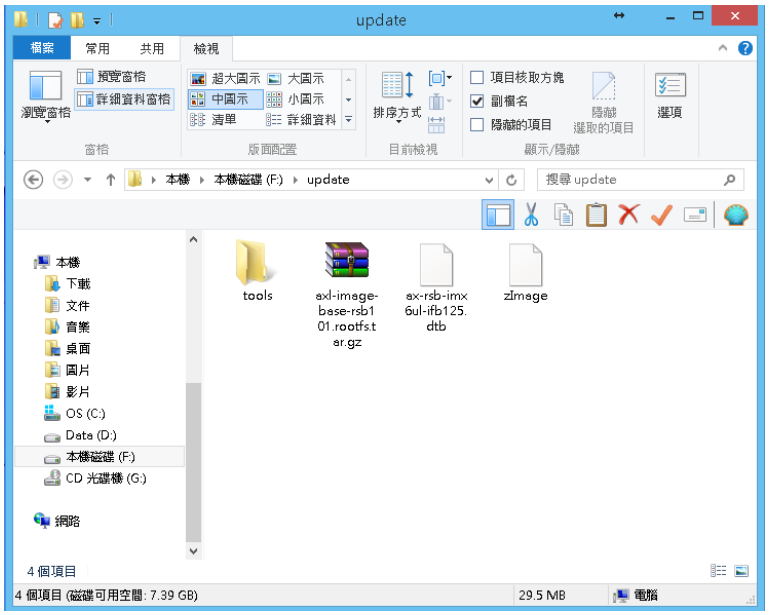
3. If you only want to update the kernel without altering the root filesystem, simply rename the new kernel file to 'zImage' and the dtb file to 'ax-rsb-imx6ul-ifb125.dtb' and then put the files in the update folder.



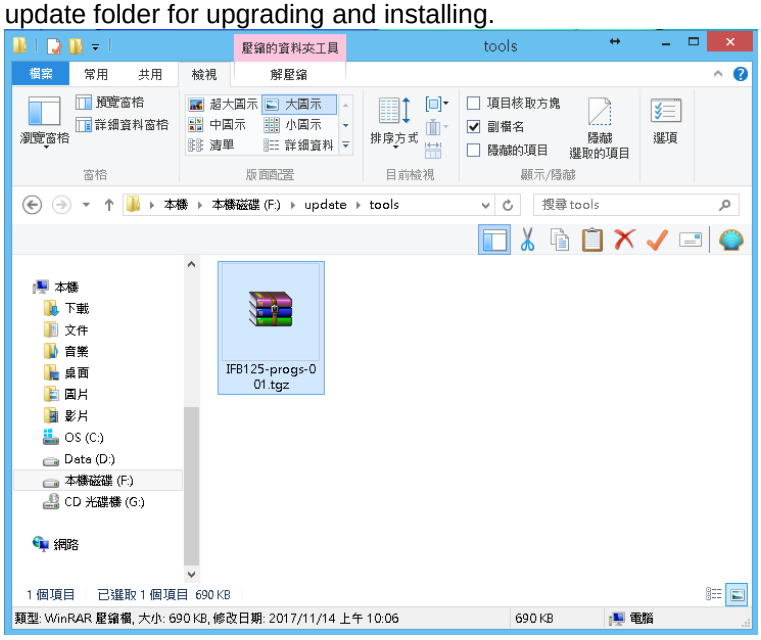
4. If you only want to update root filesystem without altering the kernel, simply put 'axl-*.rootfs.tar.gz' in the update folder.



5. If you want to update both the kernel and root filesystem, put the three files in the update folder.

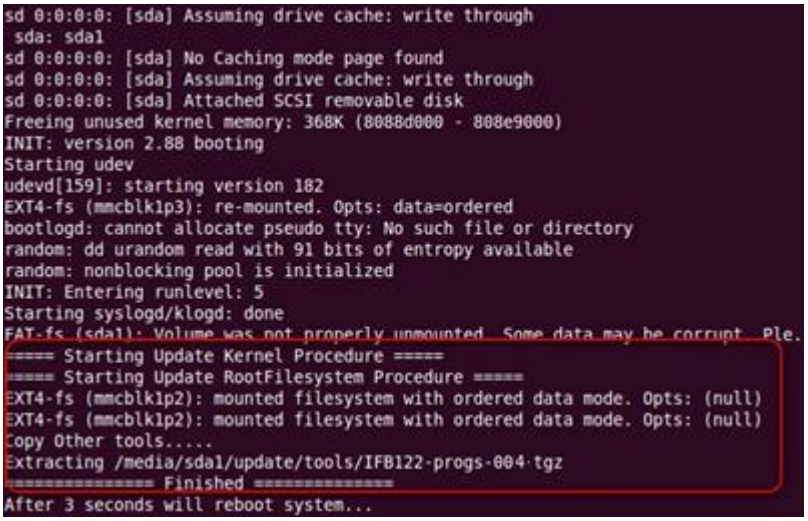


6. If Axiomtek provides other apps or tools to install, create a tools folder under the update folder for upgrading and installing.



7. Attach USB flash drive to IFB125.

8. Run the run_rescue shell script.
~# /etc/run_rescue



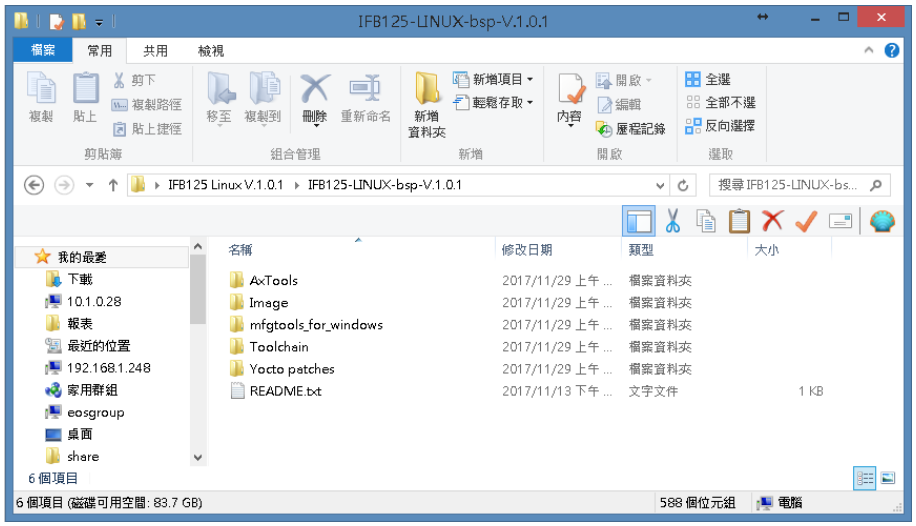
9. During the update procedure, four custom LEDs will blink like a marquee. Until procedure finish, the system will reboot again automatically, and system status LED will turn from the blinking mode to the always on mode.

2.6 How to use MFG tool to download image

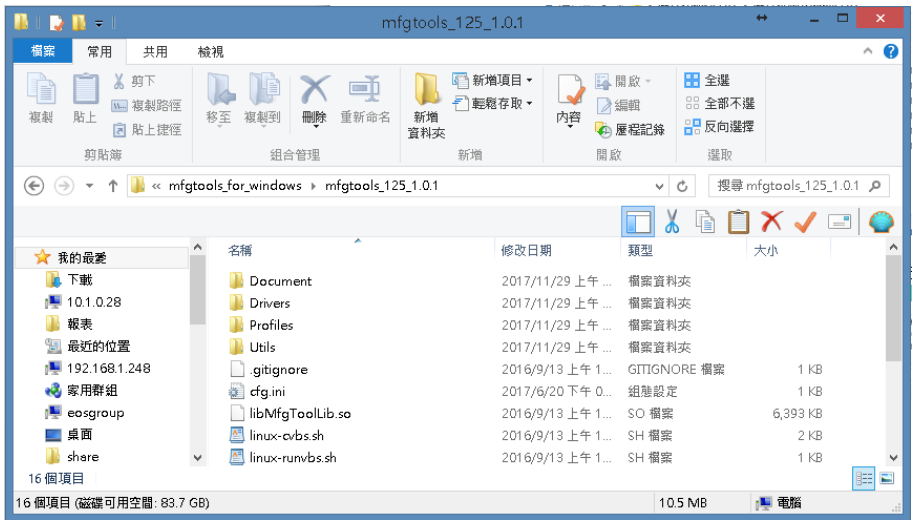
We show you how to use MFG tool to download image to the IFB125 system.

1. Before using the MFG tool, you have to change the IFB125 JP1 boot mode (default emmc boot) to OTG serial downloader mode. Then change the JP3 USB mode (default OTG host mode) to OTC client mode. Connect the IFB125 and PC with a USB cable.

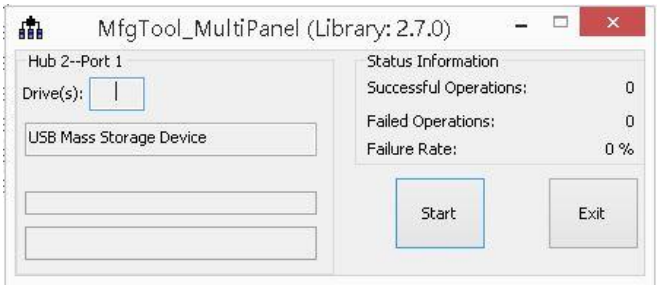
2. Extract Axiomtek's Yocto BSP and you will see mfgtools_125_x.x.x in the mfgtools_for_windows directory



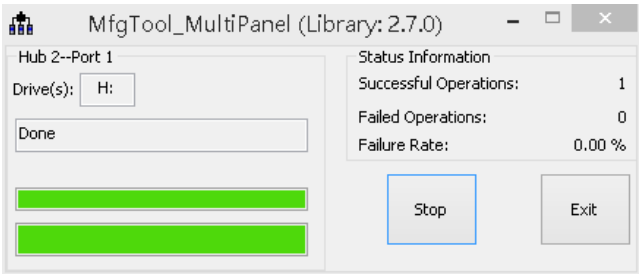
3. Enter mfgtools_for_windows/mfgtools_125_x.x.x directory



4. After double clicking mfgtools-IFB125.vbs, click "Start" to start burning



5. After burning has completed, the status will change to "Done" as below.



6. For detailed information about MFG tool, please refer to "Manufacturing Tool V2 Quick Start Guide.docx" in the "Document\V2" directory.

This page is intentionally left blank.

Chapter 3
The Embedded Linux

3.1 Embedded Linux Image Managing

3.1.1 System Version

This section describes how to determine system version information including kernel and root filesystem versions on the IFB125.

Check kernel version with the following command:
~# uname -r

```
root@axiomtek:~# uname -r
3.14.52-RSB10X-125-003
```

Check root filesystem with the login screen:

```
Poky (Yocto Project Reference Distro) 1.8.1-1 axiomtek /dev/ttymxc0

axiomtek login: root
```

3.1.2 System Time

System time is the time value loaded from RTC each time the system boots up. Read system time with the following command on IFB125:

~# date

```
root@axiomtek:~# date
Tue May  2 07:16:36 UTC 2017
```

3.1.3 Internal RTC Time

The internal RTC time is read from i.MX processor internal RTC.
Note : this time value is not saved when system power is removed.

Read internal RTC time with the following command on IFB125:
~# hwclock -r --rtc=/dev/rtc1

```
root@axiomtek:~# hwclock -r --rtc=/dev/rtc1
Thu Jan  1 00:31:56 1970  0.000000 seconds
```

3.1.4 External RTC Time

The external RTC time is read from RS5C372 external RTC. When system power is removed, this time value is kept as RS5C372 and powered by battery.

Read external RTC time with the following command:
~# hwclock -r

```
root@axiomtek:~# hwclock -r
Tue May  2 07:17:40 2017  0.000000 seconds
```

3.1.5 Watchdog timer

Function: wdt_driver_test.out
Description: When <sleep> parameters is more than <timeout> parameters, watchdog timer will be trigger

Note: The IFB125 has been enabled for default settings, and the default parameters are ***10 5 0***
Commands example: ~# wdt 10 5 0 &

```
root@axiomtek:~# /usr/sbin/wdt
Usage: wdt_driver_test <timeout> <sleep> <test>
    timeout: value in seconds to cause wdt timeout/reset
    sleep: value in seconds to service the wdt
    test: 0 - Service wdt with ioctl(), 1 - with write()
```

3.1.6 Adjusting System Time

1. Manually set up the system time.

Format: YYYYMMDDHHmm.SS
~# date -s 201706061200.00

```
root@axiomtek:~# date -s 201706061200.00
Tue Jun  6 12:00:00 UTC 2017
```

2. Write sync time to internal RTC

~# hwclock -w --rtc=/dev/rtc1
~# hwclock -r --rtc=/dev/rtc1

```
root@axiomtek:~# hwclock -w --rtc=/dev/rtc1
root@axiomtek:~# hwclock -r --rtc=/dev/rtc1
Tue Jun  6 12:05:42 2017  0.000000 seconds
```

3. Write sync time to external RTC

~# hwclock -w
~# hwclock -r

```
root@axiomtek:~# hwclock -w
root@axiomtek:~# hwclock -r
Tue Jun  6 12:08:03 2017  0.000000 seconds
```

3.1.7 LEDs Control

Four custom LEDs are supported by IFB125: LED1, LED2, LED3 and LED4.

Use the sysfs filesystem to control LED on/off state.

1. Turn on LED1

```
~# echo 255 > /sys/class/leds/LED1/brightness
```

```
root@axiomtek:~# echo 255 > /sys/class/leds/LED1/brightness
```

2. Turn on LED2

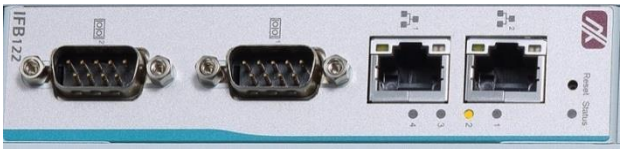
```
~# echo 255 > /sys/class/leds/LED2/brightness
```

```
root@axiomtek:~# echo 255 > /sys/class/leds/LED2/brightness
```

3. Turn off LED1

```
~# echo 0 > /sys/class/leds/LED1/brightness
```

```
root@axiomtek:~# echo 0 > /sys/class/leds/LED1/brightness
```



3.1.8 I2C device

This section describes how to use the I2C device.

1. List all devices from the I2C bus:

```
~# i2cdetect -l
```

```
root@rsb101:~# i2cdetect -l
i2c-0   i2c             21a0000.i2c                             I2C adapter
i2c-1   i2c             21a4000.i2c                             I2C adapter
```

2. Show the device register information in the I2C bus:

```
~# i2cdump -f -y 1 0x50
```

```
root@rsb101:~# i2cdump -f -y 1 0x50
No size specified (using byte-data access)
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f    0123456789abcdef
00: ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff    ................
10: ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff    ................
20: ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff    ................
```

3. Write 0x01 to address 0x00 at the register 0x50 in the I2C-1 device:

```
~# i2cset -f -y 1 0x50 0x00 0x01
```

```
root@rsb101:~# i2cset -f -y 1 0x50 0x00 0x01
```

4. Read address 0x00 at the register 0x50 in the I2C-1 device:

```
~# i2cget -f -y 1 0x50 0x00
```

```
root@rsb101:~# i2cget -f -y 1 0x50 0x00
0x01
```

3.1.9 SPI device

This section describes how to use SPI device.

```
~# ax_spidev_tool
```

```
root@rsb101:~# ax_spidev_tool
spi mode: 0
bits per word: 8
max speed: 1000000 Hz (1000 KHz)
Usage:  [-DsbdlHOLC3] [X]
  -D --device   device to use (default /dev/spidev0.0)
  -s --speed    max speed (Hz)
  -d --delay    delay (usec)
  -b --bpw      bits per word
  -l --loop     loopback
  -H --cpha     clock phase
  -O --cpol     clock polarity
  -L --lsb      least significant bit first
  -C --cs-high  chip select active high
  -3 --3wire    SI/SO signals shared
  -X --xdata    hexadecimal data
ax_spidev_tool -D /dev/spidev0.0 -s 10000000 -b 8 -X 0xbb 0xcc
ax_spidev_tool -H -O -b 16 -X 0xC000

can't send spi message: Success
Aborted
```

Example:

```
~# ax_spidev_tool -D /dev/spidev0.0 -s 10000000 -b 8 -X 0xbb 0xcc
```

```
root@rsb101:~# ax_spidev_tool -D /dev/spidev0.0 -s 10000000 -b 8 -X 0xbb 0xcc
spi mode: 0
bits per word: 8
max speed: 10000000 Hz (10000 KHz)
TX: bb cc        RX: f2f2 00
```

```
~# ax_spidev_tool -H -O -b 16 -X 0xC000
```

```
root@rsb101:~# ax_spidev_tool -H -O -b 16 -X 0xC000
spi mode: 3
bits per word: 16
max speed: 1000000 Hz (1000 KHz)
TX: c000         RX: 00e5
```

```
~# ax_spidev_tool -H -O -b 16 -X 0xEC00
```

```
root@rsb101:~# ax_spidev_tool -H -O -b 16 -X 0xEC00
spi mode: 3
bits per word: 16
max speed: 1000000 Hz (1000 KHz)
TX: ec00         RX: e50a
```

3.2 Networking

3.2.1 FTP – File Transfer Protocol

FTP is a standard network protocol used to transfer files from one host to another host over a TCP-based network.

The IFB125 comes with a built-in FTP server. Section 2.1 shows the steps to put the 'hello' program in the IFB125 via FTP.

3.2.2 TFTP – Trivial File Transfer Protocol

TFTP is a lightweight protocol for transferring files between a TFTP server and a TFTP client over Ethernet. To support TFTP, this embedded Linux image has a built-in TFTP client, and so does its accompanying bootloader U-boot.

Please refer to Chapter 5for descriptions of TFTP server installation and kernel boot up process via TFTP. Section 2.3.3 shows how to transfer files between a server and a client.

3.2.3 How to use a 3G or 4G module (Optional)

1. **3G / 4G module connection to the Internet with PPP**
   This section describes how to use a 3G or 4G module to connect to the Internet with PPP

   1.1 If you are using a Quectel UC20 3G module, follow the instructions below.

   Please execute script for internet connection.
   ~# /etp/ppp/ppp-quectel-on

```
root@axiomtek:~# /etc/ppp/ppp-quectel-on
```

   When you execute script, you may find the information below.

```
PPP generic driver version 2.4.2
pppd options in effect:
dump            # (from command line)
noauth          # (from /etc/ppp/peers/quectel)
user CARD               # (from /etc/ppp/peers/quectel)
password ??????         # (from /etc/ppp/peers/quectel)
/dev/ttyUSB3            # (from /etc/ppp/peers/quectel)
115200          # (from /etc/ppp/peers/quectel)
lock            # (from /etc/ppp/peers/quectel)
connect /usr/sbin/chat -s -v -f /etc/ppp/quectel-chat-connect           # (from)
disconnect /usr/sbin/chat -s -v -f /etc/ppp/quectel-chat-disconnect             )
crtscts         # (from /etc/ppp/peers/quectel)
modem           # (from /etc/ppp/peers/quectel)
hide-password           # (from /etc/ppp/peers/quectel)
ipcp-accept-local               # (from /etc/ppp/peers/quectel)
ipcp-accept-remote              # (from /etc/ppp/peers/quectel)
noipdefault             # (from /etc/ppp/peers/quectel)
defaultroute            # (from /etc/ppp/peers/quectel)
usepeerdns              # (from /etc/ppp/peers/quectel)
nobsdcomp               # (from /etc/ppp/peers/quectel)
root@axiomtek:~#
```

   You can execute command **,ifconfig** to examine PPP0 connection.
   ~# ifconfig

```
root@axiomtek:~# ifconfig
```

PPP0 will be shown after successful connection.

```
ppp0      Link encap:Point-to-Point Protocol
          inet addr:10.116.2.38  P-t-P:10.64.64.64  Mask:255.255.255.255
          UP POINTOPOINT RUNNING NOARP MULTICAST  MTU:1500  Metric:1
          RX packets:6 errors:0 dropped:0 overruns:0 frame:0
          TX packets:5 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:3
          RX bytes:65 (65.0 B)  TX bytes:86 (86.0 B)

root@axiomtek:~#
```

1.2 If you are using a Sierra MC7304 4G module, follow the instructions below.

Please execute script for internet connection.
~# /etp/ppp/ppp-sierra-on

```
root@axiomtek:~# /etc/ppp/ppp-sierra-on
```

When you execute script, you may find the information below.

```
PPP generic driver version 2.4.2
pppd options in effect:
dump            # (from command line)
noauth          # (from /etc/ppp/peers/sierra)
user CARD               # (from /etc/ppp/peers/sierra)
password ??????         # (from /etc/ppp/peers/sierra)
/dev/ttyUSB2            # (from /etc/ppp/peers/sierra)
115200          # (from /etc/ppp/peers/sierra)
lock            # (from /etc/ppp/peers/sierra)
connect /usr/sbin/chat -s -v -f /etc/ppp/sierra-chat-connect            # (from)
disconnect /usr/sbin/chat -s -v -f /etc/ppp/sierra-chat-disconnect              )
crtscts         # (from /etc/ppp/peers/sierra)
modem           # (from /etc/ppp/peers/sierra)
hide-password           # (from /etc/ppp/peers/sierra)
ipcp-accept-local               # (from /etc/ppp/peers/sierra)
ipcp-accept-remote              # (from /etc/ppp/peers/sierra)
noipdefault             # (from /etc/ppp/peers/sierra)
defaultroute            # (from /etc/ppp/peers/sierra)
usepeerdns              # (from /etc/ppp/peers/sierra)
nobsdcomp               # (from /etc/ppp/peers/sierra)
root@axiomtek:~#
```

You can execute command **,ifconfig** to examine PPP0 connection.
~# ifconfig

```
root@axiomtek:~# ifconfig
```

PPP0 will be shown after successful connection.

```
ppp0      Link encap:Point-to-Point Protocol
          inet addr:10.33.122.177  P-t-P:10.64.64.64  Mask:255.255.255.255
          UP POINTOPOINT RUNNING NOARP MULTICAST  MTU:1500  Metric:1
          RX packets:5 errors:0 dropped:0 overruns:0 frame:0
          TX packets:5 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:3
          RX bytes:62 (62.0 B)  TX bytes:86 (86.0 B)

root@axiomtek:~#
```

2. 3G / 4G module connection to the Internet with wvdial Tool

2.1 If you are using a Quectel UC20 3G module, follow the instructions below.

To create a wvdial config

~# vi /etc/wvdial.conf

```
root@axiomtek:~# vi /etc/wvdial.conf
```

Please enter user information as shown below.

```
[Dialer Defaults]
Modem = /dev/ttyUSB3
Baud = 115200
Init 3 =AT+CGDCONT=1,"IP","INTERNET"
Phone = *99#
Password = any
Username = any
Dial Command = ATD
Modem Type = Analog Modem
NEW PPPD = yes
```

Please execute wvdial for internet connection.

~# wvdial &

```
root@axiomtek:~# wvdial &
```

When you execute wvdia, you may find the information below.

```
[1] 426
root@axiomtek:~# --> WvDial: Internet dialer version 1.61
--> Initializing modem.
--> Sending: ATZ
ATZ
OK
--> Modem initialized.
--> Sending: ATD*99#
--> Waiting for carrier.
ATD*99#
CONNECT 14400000
--> Carrier detected.  Waiting for prompt.
--> Don't know what to do!  Starting pppd and hoping for the best.
--> Starting pppd at Mon Aug 15 10:51:15 2016
--> Pid of pppd: 429
PPP generic driver version 2.4.2
--> Using interface ppp0
--> local  IP address 10.112.49.117
--> remote IP address 10.64.64.64
--> primary   DNS address 168.95.1.1
--> secondary DNS address 168.95.192.1

root@axiomtek:~#
```

You can execute command **,ifconfig** to examine PPP0 connection.

~# ifconfig

```
root@axiomtek:~# ifconfig
```

PPP0 will be shown after successful connection.

```
ppp0      Link encap:Point-to-Point Protocol
          inet addr:10.112.49.117  P-t-P:10.64.64.64  Mask:255.255.255.255
          UP POINTOPOINT RUNNING NOARP MULTICAST  MTU:1500  Metric:1
          RX packets:6 errors:0 dropped:0 overruns:0 frame:0
          TX packets:5 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:3
          RX bytes:65 (65.0 B)  TX bytes:86 (86.0 B)

root@axiomtek:~#
```

2.2 If you are using a Sierra MC7304 4G module, follow the instructions below.

To create a wvdial config
~# vi /etc/wvdial.conf

```
root@axiomtek:~# vi /etc/wvdial.conf
```

Please enter user information as below.

[Dialer Defaults]
Modem = /dev/ttyUSB2
Baud = 115200
Init 3 =AT+CGDCONT=1,"IP","INTERNET"
Phone = *99#
Password = any
Username = any
Dial Command = ATD
Modem Type = Analog Modem
NEW PPPD = yes

Please execute wvdial for internet connection.
~# wvdial &

```
root@axiomtek:~# wvdial &
```

When you execute **wvdia,** you may find the information below.

```
[1] 437
root@axiomtek:~# --> WvDial: Internet dialer version 1.61
--> Cannot get information for serial port.
--> Initializing modem.
--> Sending: ATZ
ATZ
OK
--> Modem initialized.
--> Sending: ATD*99#
--> Waiting for carrier.
ATD*99#
CONNECT 100000000
--> Carrier detected.  Waiting for prompt.
--> Don't know what to do!  Starting pppd and hoping for the best.
--> Starting pppd at Mon Aug 15 10:51:09 2016
--> Pid of pppd: 441
PPP generic driver version 2.4.2
--> Using interface ppp0
--> local  IP address 10.33.122.177
--> remote IP address 10.64.64.64
--> primary   DNS address 168.95.1.1
--> secondary DNS address 168.95.192.1

root@axiomtek:~#
```

You can execute command **,ifconfig** to examine PPP0 connection.
~# ifconfig

```
root@axiomtek:~# ifconfig
```

PPP0 will be shown after successful connection.

```
ppp0      Link encap:Point-to-Point Protocol
          inet addr:10.33.122.177  P-t-P:10.64.64.64  Mask:255.255.255.255
          UP POINTOPOINT RUNNING NOARP MULTICAST  MTU:1500  Metric:1
          RX packets:5 errors:0 dropped:0 overruns:0 frame:0
          TX packets:5 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:3
          RX bytes:62 (62.0 B)  TX bytes:86 (86.0 B)

root@axiomtek:~#
```

3. 3G / 4G module connection to the Internet with Ax tool

3.1 If your 3G/4G module use UC20/MC7304 / LARA-R211 / LARA-R280, you can use ax_3g4g_wvdial command.

~# ax_3g4g_wvdial

```
root@rsb201:~# ax_3g4g_wvdial
###axmsg: create wvdil.conf example tool.
###axmsg: set ublox-LARA-R280 wvdil.conf.
```

According to your 3G/4G module,will create a dependency module's configure

Note: LARA-R211 and LARA-R280 use the same driver so you only see LARA-R280.

Please execute wvdial for internet connection.
~# wvdial &

```
root@axiomtek:~# wvdial &
```

When you execute **wvdial,** you may find the information below.

```
root@rsb201:~# wvdial &
[1] 813
root@rsb201:~# --> WvDial: Internet dialer version 1.61
--> Initializing modem.
--> Sending: ATZ
ATZ
OK
--> Sending: ATQ0 V1 E1 S0=0
ATQ0 V1 E1 S0=0
OK
--> Modem initialized.
--> Sending: ATDT*99***4#
--> Waiting for carrier.
ATDT*99***4#
CONNECT
--> Carrier detected.  Starting PPP immediately.
--> Starting pppd at Mon May 21 02:01:33 2018
--> Pid of pppd: 815
PPP generic driver version 2.4.2
--> Using interface ppp0
--> pppd: ◆8T
--> pppd: ◆8T
--> pppd: ◆8T
--> pppd: ◆8T
--> local  IP address 10.203.98.12
--> pppd: ◆8T
--> remote IP address 10.203.98.12
--> pppd: ◆8T
--> primary   DNS address 172.24.9.33
--> pppd: ◆8T
--> secondary DNS address 172.24.9.22
--> pppd: ◆8T
```

PPP0 will be shown after successful connection.

```
ppp0      Link encap:Point-to-Point Protocol
          inet addr:10.203.98.12  P-t-P:10.203.98.12  Mask:255.255.255.255
          UP POINTOPOINT RUNNING NOARP MULTICAST  MTU:1500  Metric:1
          RX packets:10 errors:0 dropped:0 overruns:0 frame:0
          TX packets:9 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:3
          RX bytes:1091 (1.0 KiB)  TX bytes:386 (386.0 B)
```

3.2.4 How to get the 3G/4G module signal strength (Optional)

1. If you are using Quectel UC20, follow the instructions below.

   ~# echo "AT+CSQ" > /dev/ttyUSB3

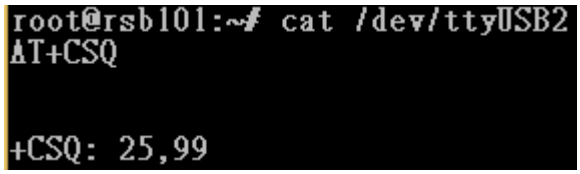
   The "18" is 3G's signal strength. The value is between 0 and 31 and the value "31" implies an excellent signal condition.

2. If you are using MC7304, follow the instructions below.

   ~# echo "AT+CSQ" > /dev/ttyUSB2

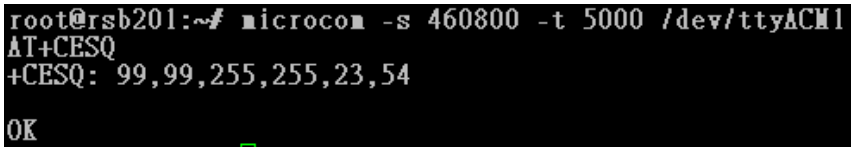
   ~# cat /dev/ttyUSB2

3. If you are using R211/R280, follow the instructions below.

   ~# microcom –s 460800 –t 5000 /dev/ttyACM1

   ~# AT+CESQ

   You will get signal strength as 23(-dBm),54(-dB)

3.2.5 How to use Wi-Fi module (Optional)

If your Wi-Fi module is WPEQ-160ACN, follow the instructions below.

Editor /etc/wpa_supplicant.conf file
~# vi /etc/wpa_supplicant.conf

```
root@axiomtek:~# vi /etc/wpa_supplicant.conf
```

Enter your router's SSID and Password

```
ctrl_interface=/var/run/wpa_supplicant
ctrl_interface_group=0
update_config=1

network={
        ssid="axiomtwek"
        psk="password"
}
```

If the setting is successful, it will automatically connect after reboot.

You can execute command" **ifconfig"** to check connection.
~# ifconfig

```
wlan0     Link encap:Ethernet  HWaddr B0:1F:81:D0:33:EA
          inet addr:192.168.0.41  Bcast:192.168.0.255  Mask:255.255.255.0
          inet6 addr: fe80::b21f:81ff:fed0:33ea/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:22 errors:0 dropped:24 overruns:0 frame:0
          TX packets:26 errors:0 dropped:5 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:5725 (5.5 KiB)  TX bytes:5679 (5.5 KiB)

root@rsb101:~#
```

This page is intentionally left blank.

Chapter 4
Programming Guide

We have released a set of application programming interface (API) functions for users to access/control hardware. With these API functions, users can more easily design their own software. This chapter includes detailed descriptions of each API function and step-by-step code samples showing how it works.

4.1 librsb10x API Functions

The IFB125 BSP includes 'librsb10x.so' shared library for users to access I/O and read back system information. This shared library is kept in BSP, and you can find it in IFB125-rsb_lib-x.x.x.tar.bz2 of AxTools. When you extract the compressed file, you will find not only the shared library, but also a *demo* folder containing an API header file and example programs.

Summary table of available API functions

No.	Function	Description
1	Get_DI0()	Read state on digital input channels.
2	Get_DI1()	Read state on digital input channels.
3	Set_DO()	Set digital output channels state.
4	Get_DI0_not()	Read state on digital input channels.(reverse)
5	Get_DI1_not()	Read state on digital input channels. (reverse)
6	Set_DO0_not()	Set digital output channels state. (reverse)
7	Set_RELAY()	Set relay high or low state.
8	Control_LED()	Enable or disable LED
9	Control_WDT()	Set WDT function

COM sample code:

COM receive

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <errno.h>
#include <termios.h>
#include <fcntl.h>
#include <termios.h>
#include <pthread.h>
#include "serial.h"
#include <asm-generic/ioctls.h>

#define SET_COM_TYPE                    0x542A
#define SET_RS485_TERM                  0x542C

static void help(void) {
        fprintf(stderr,
                "Usage: comRead [MODE]\n"
                "   MODE: 1/2/3 \n"
                "           1=232\n"
                "           2=485\n"
                "           3=422\n");
                exit(1);
}

int main(int argc, char *argv[])
{
        if(argc !=2 || !(atoi(argv[1])==1||atoi(argv[1])==2||atoi(argv[1])==3))
                help();
        int ReadRet,fd,RX_len = 0,OutCount = 0;
        struct termios orig_options,options;
        struct serial_rs485 conf;
        char RecvBuf[128];
        int type = atoi(argv[1]);
        printf("Test for com1 Read(232/422/485) \n");
        //printf("Test for com2 Read(232/422/485) \n");
        printf("example : ./comRead 1 (1=232, 2=485, 3=422)\n");
        fd = open("/dev/ttymxc1", O_RDWR | O_NOCTTY);
        //fd = open("/dev/ttymxc2", O_RDWR | O_NOCTTY);
        if(fd < 0) {
                printf("open error /dev/ttymxc2 error\n");
        }
        //setting com1 as rs485
                switch(type) {
                case 1:
                        printf("Set as RS232\n");
                        break;
                case 2:
                        printf("Set as RS485\n");
                        break;
                case 3:
                        printf("Set as RS422\n");
                        break;
        }
        //init setting
```

```
        fcntl(fd, F_SETFL, 0);
        tcgetattr(fd, &orig_options);
        memset(&options, 0, sizeof(options));
        options.c_cflag &= ~CSTOPB;
        options.c_cflag &= ~CSIZE;
        options.c_cflag |= PARENB;
        options.c_cflag &= ~PARODD;
        options.c_cflag |= CS8;
        options.c_cflag &= ~CRTSCTS;
        options.c_iflag &= ~(IXON | IXOFF | IXANY);
        options.c_lflag &= ~(ICANON | IEXTEN | ISIG | ECHO);
        options.c_oflag &= ~OPOST;
        options.c_iflag &= ~(ICRNL | INPCK | ISTRIP | IXON | BRKINT );
        options.c_cflag |= (CLOCAL | CREAD);
        options.c_cc[VMIN] = 1;
        options.c_cc[VTIME] = 0;

        usleep(100);
        ioctl(fd, SET_COM_TYPE, &type);
        cfsetispeed(&options, B115200);
        cfsetospeed(&options, B115200);
        tcsetattr(fd, TCSANOW, &options);

        while(1)
        {
                //Test Read
                memset(RecvBuf,0x00,sizeof(RecvBuf));
                ReadRet = read(fd, RecvBuf, sizeof(RecvBuf));
                if (ReadRet > 0)
                {
                        printf("Test Read : Len [%d] / Read
[%s]\n",ReadRet,RecvBuf);
                }
                usleep(100000);

        }
        tcsetattr(fd, TCSANOW, &orig_options);
        close(fd); //Close the serial port
        printf("Serial port closed.\n");

        return 0;
}
```

COM send:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <errno.h>
#include <termios.h>
#include <fcntl.h>
#include <termios.h>
#include <pthread.h>
#include "serial.h"
#include <asm-generic/ioctls.h>

#define SET_COM_TYPE                    0x542A
```

```
#define SET_RS485_TERM                  0x542C

static void help(void) {
        fprintf(stderr,
                "Usage: comWrite [MODE]\n"
                "   MODE: 1/2/3 \n"
                "           1=232\n"
                "           2=485\n"
                "           3=422\n");
                exit(1);
}

int main(int argc, char *argv[])
{
        if(argc !=2 || !(atoi(argv[1])==1||atoi(argv[1])==2||atoi(argv[1])==3))
                help();
        int i,WriteRet,fd,TX_len = 0;
        struct termios orig_options,options;
        struct serial_rs485 conf;
        char SendBuf[16];
        int type = atoi(argv[1]);
        printf("Test for com1 Write(232/422/485) \n");
        //printf("Test for com2 Write(232/422/485) \n");
        printf("example : ./comWrite 1 (1=232, 2=485, 3=422)\n");
        fd = open("/dev/ttymxc1", O_RDWR | O_NOCTTY);
        //fd = open("/dev/ttymxc2", O_RDWR | O_NOCTTY);
        if(fd < 0) {
                printf("open error /dev/ttymxc1 error\n");
        }
        //setting com1 as rs485
                switch(type) {
                case 1:
                        printf("Set as RS232\n");
                        break;
                case 2:
                        printf("Set as RS485\n");
                        break;
                case 3:
                        printf("Set as RS422\n");
                        break;
        }
        //init setting
        fcntl(fd, F_SETFL, 0);
        tcgetattr(fd, &orig_options);
    memset(&options, 0, sizeof(options));
        options.c_cflag &= ~CSTOPB;
        options.c_cflag &= ~CSIZE;
        options.c_cflag |= PARENB;
        options.c_cflag &= ~PARODD;
        options.c_cflag |= CS8;
        options.c_cflag &= ~CRTSCTS;
        options.c_iflag &= ~(IXON | IXOFF | IXANY);
        options.c_lflag &= ~(ICANON | IEXTEN | ISIG | ECHO);
        options.c_oflag &= ~OPOST;
        options.c_iflag &= ~(ICRNL | INPCK | ISTRIP | IXON | BRKINT );
        options.c_cflag |= (CLOCAL | CREAD);
        options.c_cc[VMIN] = 1;
        options.c_cc[VTIME] = 0;

        usleep(100);
```

```
        ioctl(fd, SET_COM_TYPE, &type);
        cfsetispeed(&options, B115200);
        cfsetospeed(&options, B115200);
        tcsetattr(fd, TCSANOW, &options);

        printf("start write\n");
        memset(SendBuf,0x00,16);
        sprintf(SendBuf,"hello word");

        for(i=0;i<10;i++)
        {
                //Test Write
                WriteRet = write(fd,SendBuf,strlen(SendBuf));
                if(WriteRet > 0)
                {
                        TX_len = strlen(SendBuf);
                        printf("Test Write :Len [%d] / Send [%s] \n",TX_len,SendBuf);
                }
                else
                {
                        printf("Test Write Fail \n");
                }
                usleep(500000);

        }
        tcsetattr(fd, TCSANOW, &orig_options);
        close(fd); //Close the serial port
        printf("Serial port closed.\n");

        return 0;
}
```

Function: Get_DI0()

Function	**int Get_DI0(int *data);**
Description	Read state on digital input channels.
Arguments	data: This function will store digital input data in this argument.
Return	0: No error. 1: Function fails.
Others	None.

Function: Get_DI1()

Function	**int Get_DO1(int *data);**
Description	Read state on digital input channels.
Arguments	data: This function will store digital output data in this argument.
Return	0: No error. 1: Function fails.
Others	None.

Function: Set_DO()

Function	int Set_DO(int data);
Description	Set digital output channels state.
Arguments	data: Data to be written to digital output channels.
Return	0: No error. 1: Function fails.
Others	None.

Function: Get_DI0_not()

Function	int Get_DI0_not (int *data);
Description	Read state on digital input channels. (reverse)
Arguments	data: This function will store digital input data in this argument.
Return	0: No error. 1: Function fails.
Others	None.

Function: Get_DI1_not ()

Function	int Get_DO1_not (int *data);
Description	Read state on digital input channels. (reverse)
Arguments	data: This function will store digital output data in this argument.
Return	0: No error. 1: Function fails.
Others	None.

Function: Set_DO_not ()

Function	int Set_DO_not (int data);
Description	Set digital output channels state. (reverse)
Arguments	data: Data to be written to digital output channels.
Return	0: No error. 1: Function fails.
Others	None.

DIO sample code:

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <linux/types.h>
#include "librsb10x.h"

int main(int argc, char* argv[])
{
        int ch0,ch1;
        Set_DO0(0);

        Set_DO0(1);
        printf("\nDO0 Oput     = 1\n");
        Get_DI0(&ch0);
        printf("DI0 Input        = %d\n", ch0);
        Get_DI1(&ch1);
        printf("DI1 Input        = %d\n", ch1);
        sleep(1);

        Set_DO0(0);
        printf("\nDO0 Oput     = 0\n");
        Get_DI0(&ch0);
        printf("DI0 Input        = %d\n", ch0);
        Get_DI1(&ch1);
        printf("DI1 Input        = %d\n", ch1);
        sleep(1);

        Set_DO0_not(1);
        printf("\nDO0 Oput not             = 1\n");
        Get_DI0(&ch0);
        printf("DI0 Input        = %d\n", ch0);
        Get_DI1(&ch1);
        printf("DI1 Input        = %d\n", ch1);
        Get_DI0_not(&ch0);
        printf("DI0 Input not    = %d\n", ch0);
        Get_DI1_not(&ch1);
        printf("DI1 Input not    = %d\n", ch1);
        sleep(1);

        Set_DO0_not(0);
        printf("\nDO0 Oput not             = 0\n");
        Get_DI0(&ch0);
        printf("DI0 Input        = %d\n", ch0);
        Get_DI1(&ch1);
        printf("DI1 Input        = %d\n", ch1);
        Get_DI0_not(&ch0);
        printf("DI0 Input not    = %d\n", ch0);
        Get_DI1_not(&ch1);
        printf("DI1 Input not    = %d\n", ch1);
        sleep(1);

        return 0;
}
```

Function: Set_RELAY ()

Function	**int Set_RELAY(int hl);**
Description	Set relay high or low state.
Arguments	hl: relay state. 0: LOW. 1: HIGH.
Return	0: No error. 1: Function fails.
Others	None.

Relay sample code:

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <linux/types.h>
#include "librsb10x.h"

#define HIGH        1
#define LOW         0

int main(int argc, char* argv[])
{
    printf("Turn relay on\n");
    Set_RELAY(HIGH);

    sleep(2);

    printf("Turn relay off\n");
    Set_RELAY(LOW);

    return 0;
}
```

Function: Control_LED ()

Function	**int Control_LED(int num,int enable);**
Description	Enable or disable LED
Arguments	Num : LED number,default as 1 ~ 4 Enable : enable or disable LED 0: disable 1: enable
Return	0: No error. 1: Function fails.
Others	None.

LED sample code:

```
#include <stdio.h>
#include <stdlib.h>
#include "librsb10x.h"

int Control_LED(int num,int enable);

int main()
{
        printf("Function Name : Control_LED(num,enable)\n");
        printf("num: LED number,default 1~4 \n");
        printf("enable: 1 - enable LED , 2 - Disable LED\n");
        printf("turn on LED 1\n");
        Control_LED(1,1);
        sleep(1);
        printf("turn on LED 2\n");
        Control_LED(2,1);
        sleep(1);
        printf("turn on LED 3\n");
        Control_LED(3,1);
        sleep(1);
        printf("turn on LED 4\n");
        Control_LED(4,1);
        sleep(1);
        printf("turn off LED 1\n");
        Control_LED(1,0);
        sleep(1);
        printf("turn off LED 2\n");
        Control_LED(2,0);
        sleep(1);
        printf("turn off LED 3\n");
        Control_LED(3,0);
        sleep(1);
        printf("turn off LED 4\n");
        Control_LED(4,0);
        return 0;
}
```

Function: Control_WDT ()

Function	**int Control_WDT(int timeout,int sleep_time,int test);**
Description	Set WDT Function
Arguments	timeout : value in seconds to cause wdt timeout/reset sleep_time : value in seconds to service the wdt test : 0 – service wdt with ioctl(), 1 – with write()
Return	0: No error. 1: Function fails.

WDT sample code:

```
#include <stdio.h>
#include <stdlib.h>,
#include "librsb10x.h"


int main()
{
        printf("Function Name : Control_WDT(timeout,sleep_time,test)\n");
        printf("timeout: value in seconds to cause wdt timeout/reset \n");
        printf("sleep_time: value in seconds to service the wdt \n");
        printf("test: 0 - Service wdt with ioctl(), 1 - with write()\n");
        printf("\nRun Contrl_WDT(10,5,0)\n");
        Contrl_WDT(10,5,0);
        return 0;
}
```

4.2 Compile Demo Program

4.2.1 Install IFB125 I/O Library

Before you develop and compile a sample program, you should install Yocto toolchain into a development PC. To do so, refer to Chapter 5 "Board Support Package".

1. Set up the cross-development environment on your host PC.

~$ source /opt/poky/1.8.1/environment-setup-cortexa7hf-vfp-neon-poky-linux-gnueabi

```
ryan@axiomtek:~$ source /opt/poky/1.8.1/environment-setup-cortexa7hf-vfp-neon-poky-linux-gnueabi
```

2. To compile and build a demo program for the IFB125, do the following:

Change to *your project* directory.

~$ cd project/IFB125\ Linux\ V.1.0.1/IFB125-LINUX-bsp-V.1.0.1/AxTools/

```
ryan@axiomtek:~$ cd project/IFB125\ Linux\ V.1.0.1/IFB125-LINUX-bsp-V.1.0.1/AxTools/
```

3. Extract driver source to *your project* directory.

~$ tar -xvf IFB125-rsb-lib-1.0.1.tar.bz2

```
ryan@axiomtek:~/project/IFB125 Linux V.1.0.1/IFB125-LINUX-bsp-V.1.0.1/AxTools$ tar -xvf IFB125-rsb-
lib-1.0.1.tar.bz2
rsb_lib/demo/Makefile
rsb_lib/demo/diotool
rsb_lib/demo/ledtest.c
rsb_lib/demo/com_mode.c
rsb_lib/demo/com_port_open.c
rsb_lib/demo/com_mode
rsb_lib/demo/diotool.c
rsb_lib/demo/com_port_open
rsb_lib/librsb10x.h
rsb_lib/librsb10x.so.0
rsb_lib/demo/serial.h
rsb_lib/demo/ledtest
rsb_lib/demo/librsb10x.h
rsb_lib/demo/relay.c
rsb_lib/demo/diotest.c
rsb_lib/demo/wdttest.c
rsb_lib/demo/
rsb_lib/librsb10x.so.1.0.1
rsb_lib/demo/diotest
rsb_lib/
rsb_lib/demo/wdttest
rsb_lib/librsb10x.so
rsb_lib/demo/relay
```

4. Change to *rsb_lib/demo* directory.

~$ cd rsb_lib/demo

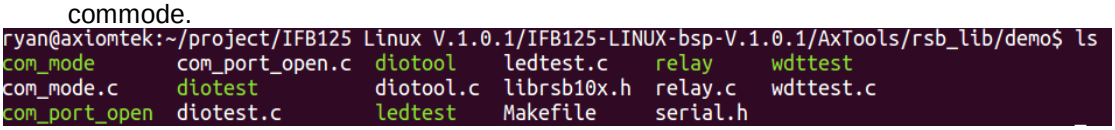
```
ryan@axiomtek:~/project/IFB125 Linux V.1.0.1/IFB125-LINUX-bsp-V.1.0.1/AxTools$ cd rsb_lib/demo/
```

5. Build the demo program.

~$ make

```
ryan@axiomtek:~/project/IFB125 Linux V.1.0.1/IFB125-LINUX-bsp-V.1.0.1/AxTools/rsb_lib/demo$ make
arm-poky-linux-gnueabi-gcc  -march=armv7-a -mfloat-abi=hard -mfpu=neon -mtune=cortex-a7 --sysroot=/
opt/poky/1.8.1/sysroots/cortexa7hf-vfp-neon-poky-linux-gnueabi  -o com_mode com_mode.c -lrsb10x -L.
./
arm-poky-linux-gnueabi-gcc  -march=armv7-a -mfloat-abi=hard -mfpu=neon -mtune=cortex-a7 --sysroot=/
opt/poky/1.8.1/sysroots/cortexa7hf-vfp-neon-poky-linux-gnueabi  -o com_port_open com_port_open.c -l
rsb10x -L../
arm-poky-linux-gnueabi-gcc  -march=armv7-a -mfloat-abi=hard -mfpu=neon -mtune=cortex-a7 --sysroot=/
opt/poky/1.8.1/sysroots/cortexa7hf-vfp-neon-poky-linux-gnueabi  -o diotest diotest.c -lrsb10x -L../
arm-poky-linux-gnueabi-gcc  -march=armv7-a -mfloat-abi=hard -mfpu=neon -mtune=cortex-a7 --sysroot=/
opt/poky/1.8.1/sysroots/cortexa7hf-vfp-neon-poky-linux-gnueabi  -o diotool diotool.c -lrsb10x -L../
arm-poky-linux-gnueabi-gcc  -march=armv7-a -mfloat-abi=hard -mfpu=neon -mtune=cortex-a7 --sysroot=/
opt/poky/1.8.1/sysroots/cortexa7hf-vfp-neon-poky-linux-gnueabi  -o wdttest wdttest.c -lrsb10x -L../
arm-poky-linux-gnueabi-gcc  -march=armv7-a -mfloat-abi=hard -mfpu=neon -mtune=cortex-a7 --sysroot=/
opt/poky/1.8.1/sysroots/cortexa7hf-vfp-neon-poky-linux-gnueabi  -o relay relay.c -lrsb10x -L../
arm-poky-linux-gnueabi-gcc  -march=armv7-a -mfloat-abi=hard -mfpu=neon -mtune=cortex-a7 --sysroot=/
opt/poky/1.8.1/sysroots/cortexa7hf-vfp-neon-poky-linux-gnueabi  -o ledtest ledtest.c -lrsb10x -L../
```

6. Then you should have example programs such as open_comport, diotest, and commode.



4.2.2 Run demo program

Refer to section 2.3 for detailed information.

Chapter 5
Board Support Package (BSP)

5.1 Host Development System Installation

5.1.1 Install Host System

1. Download the Ubuntu 14.04 LTS iso image.

2. Install Ubuntu 14.04.

3. Install host packages required by Yocto development as follows:

```
~$sudo apt-get install wget git-core unzip texinfo libsdl1.2-dev gawk diffstat \
        wget git-core unzip texinfo libsdl1.2-dev gawk diffstat \
        texi2html docbook-utils python-pysqlite2 help2man \
        make gcc g++ desktop-file-utils libgl1-mesa-dev \
        libglu1-mesa-dev mercurial autoconf \
        automake groff curl lzop asciidoc xterm chrpath
```

i.MX layers host packages for a Ubuntu 14.04 host setup only are:

```
~$ sudo apt-get install u-boot-tools
```

4. Install and configure the TFTP server:
After tftpd is installed, configure it by editing /etc/xinetd.d/tftp. Change the default export path (it is either */usr/var/tftpboot* or */var/lib/tftpboot*) to /. Or change the default export path to a new directory you want to download from. Then reboot the hardware.

To install tftpd / tftp/ xineted SOFTWARE

```
~$ sudo apt-get install tftpd tftp xinetd
```

To create a tftp directory

```
~$ sudo mkdir /tftpboot
~$ sudo chmod -R 777 /tftpboot
~$ sudo chown -R nobody /tftpboot
```

To configure the tftp server.

```
~$ sudo vi /etc/xinetd.d/tftp
```

```
service tftp
{
        socket_type     = dgram
        protocol        = udp
        wait            = yes
        user            = root
        server          = /usr/sbin/in.tftpd
        server_args     = -s /tftpboot
        disable         = no
        per_source      = 11
        cps             = 100 2
        flags           = IPv4
}
```

Then restart the TFTP server.
~$ sudo /etc/init.d/xinetd restart

5. Install and configure the NFS server:
~$ sudo aptitude -y install nfs-common nfs-kernel-server portmap

To configure the nfs server, add lines to */etc/exports* as follows:
*/tools/rootfs *(rw,sync,no_root_squash)*
~$ sudo vi /etc/exports

Create a symbolic link to root filesystem which you have built.
~$ sudo mkdir /tools
~$ sudo ln -s ~/project/rootfs /tools/rootfs

Then restart the NFS server.
~$ sudo /etc/init.d/nfs-kernel-server restart

5.1.2 Install Yocto Development

1. Setting up the repo utility. Create a bin folder in the home directory.
~$ mkdir ~/bin (this step may not be required if the bin folder already exists.)
~$ curl http://commondatastorage.googleapis.com/git-repo-downloads/repo > ~/bin/repo
~$ chmod a+x ~/bin/repo

```
ryan@axiomtek:~$ mkdir bin
ryan@axiomtek:~$ curl http://commondatastorage.googleapis.com/git-repo-downloads/repo > ~/bin/repo
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 27759  100 27759    0     0   571k      0 --:--:-- --:--:-- --:--:--  576k
ryan@axiomtek:~$ chmod a+x ~/bin/repo
```

Add the following line to the .bashrc file to ensure that the ~/binfolder is in your PATH variable.

~$ export PATH=~/bin:$PATH

```
ryan@axiomtek:~$ export PATH=~/bin:$PATH
```

2. Setting up the Git environment
~$ git config --global user.name "Your Name"
~$ git config --global user.email "Your Email"

```
ryan@axiomtek:~$ git config --global user.name "axiomtek"
ryan@axiomtek:~$ git config --global user.email "axio@axiomtek.com.tw"
```

3. Download the Freescale's Yocto BSP source
~$ mkdir project
~$ mkdir project/fsl-community-bsp
~$ cd project/fsl-community-bsp
~$ repo init -u
git://git.freescale.com/imx/fsl-arm-yocto-bsp.git -b imx-3.14.52-1.1.0_ga

```
ryan@axiomtek:~$ mkdir project/fsl-community-bsp
ryan@axiomtek:~$ cd project/fsl-community-bsp/
```

```
ryan@axiomtek:~/project/fsl-community-bsp$ repo init -u git://git.freescale.com
imx/fsl-arm-yocto-bsp.git -b imx-3.14.52-1.1.0_ga
Get https://gerrit.googlesource.com/git-repo/clone.bundle
Get https://gerrit.googlesource.com/git-repo
remote: Counting objects: 5, done
remote: Finding sources: 100% (17/17)
remote: Total 17 (delta 1), reused 16 (delta 1)
```

~$ repo sync

```
ryan@axiomtek:~/project/fsl-community-bsp$ repo sync
Fetching project meta-fsl-arm
Fetching project meta-qt5
Fetching projects:  11% (1/9)  Fetching project poky
Fetching projects:  22% (2/9)  Fetching project meta-fsl-demos
Fetching projects:  33% (3/9)  Fetching project meta-browser
Fetching projects:  44% (4/9)  Fetching project meta-fsl-bsp-release
```

Clone Finish

```
 * [new tag]         yocto-2.3.2 -> yocto-2.3.2
 * [new tag]         yocto-2.4   -> yocto-2.4
 * [new tag]         yocto_1.5_M5.rc8 -> yocto_1.5_M5.rc8
Fetching projects: 100% (9/9), done.
Syncing work tree: 100% (9/9), done.
```

4. Extract Axiomtek's Yocto BSP source
   ~$ tar -xvf ../IFB125-LINUX-bsp-1.0.0/Yocto\ patches/meta-axiomtek-2.5.3.tar.gz -C sources

```
ryan@axiomtek:~/project/fsl-community-bsp$ tar -xvf ../../IFB125\ Linux\ V.1.0.1/IFB125-LINUX-bsp-V.1.0.1/
Yocto\ patches/IFB125-meta-axiomtek-2.5.1.tar.gz -C sources/
```

   Check meta-axiomtek

```
ryan@axiomtek:~/project/fsl-community-bsp$ ls sources/
base           meta-browser  meta-fsl-arm-extra    meta-fsl-demos     meta-qt5
meta-axiomtek  meta-fsl-arm  meta-fsl-bsp-release  meta-openembedded  poky
```

5. Update bblayers.conf
   ~$ vim fsl-community-bsp/sources/base/conf/bblayers.conf

```
ryan@axiomtek:~/project/fsl-community-bsp$ vim sources/base/conf/bblayers.conf
```

   And add this line below after *${BSPDIR}/sources/meta-fsl-demos *

   ${BSPDIR}/sources/meta-axiomtek \

```
LCONF_VERSION = "6"

BBPATH = "${TOPDIR}"
BSPDIR := "${@os.path.abspath(os.path.dirname(d.getVar('FILE', True)) + '/../..$

BBFILES ?= ""
BBLAYERS = " \
  ${BSPDIR}/sources/poky/meta \
  ${BSPDIR}/sources/poky/meta-yocto \
  \
  ${BSPDIR}/sources/meta-openembedded/meta-oe \
  ${BSPDIR}/sources/meta-openembedded/meta-multimedia \
  \
  ${BSPDIR}/sources/meta-fsl-arm \
  ${BSPDIR}/sources/meta-fsl-arm-extra \
  ${BSPDIR}/sources/meta-fsl-demos \
  ${BSPDIR}/sources/meta-axiomtek \
"
```

6. First build
   Choose your board
   ~$ DISTRO=poky MACHINE=rsb101 EULA=1 source fsl-setup-release.sh -b build

```
ryan@axiomtek:~/project/fsl-community-bsp$ DISTRO=poky MACHINE=rsb101 EULA=1 source fsl-setup-release.sh -
b build

 Build directory is  build
Configuring for rsb101

Welcome to Freescale Community BSP
```

   Start to build image
   ~$ bitbake axl-image-base

```
ryan@axiomtek:~/project/fsl-community-bsp/build$ bitbake axl-image-base
```

7. After image is built successfully, you can find the file path:
   project/fsl-community-bsp/build/tmp/deploy/images/rsb101

```
ryan@Ubuntu:~/project/fsl-community-bsp/build/tmp/deploy/images/rsb101$ ls
axl-image-base-rsb101-20171116084023.rootfs.manifest
axl-image-base-rsb101-20171116084023.rootfs.tar.gz
axl-image-base-rsb101.manifest
axl-image-base-rsb101.tar.gz
modules--3.14.52-r0-rsb101-20171116084023.tgz
modules-rsb101.tgz
README_-_DO_NOT_DELETE_FILES_IN_THIS_DIRECTORY.txt
zImage
zImage--3.14.52-r0-ax-rsb-imx6ul-ifb125-20171116084023.dtb
zImage--3.14.52-r0-rsb101-20171116084023.bin
zImage-ax-rsb-imx6ul-ifb125.dtb
zImage-rsb101.bin
```

5.1.3 Build and Install the user's Yocto Toolchain

We have provided Yocto Toolchain in IFB125 BSP. However, if you want to build your own toolchain using Yocto development, you can follow the instructions on the host PC:

1. Change to *Yocto development* directory.
   ~$ source setup-environment build

```
ryan@OMG:~/project/fsl-community-bsp$ source setup-environment build

Welcome to Freescale Community BSP

The Yocto Project has extensive documentation about OE including a
reference manual which can be found at:
    http://yoctoproject.org/documentation

For more information about OpenEmbedded see their website:
    http://www.openembedded.org/

You can now run 'bitbake <target>'

Common targets are:
    core-image-minimal
    meta-toolchain
    meta-toolchain-sdk
    adt-installer
    meta-ide-support

Your configuration files at build have not been touched.
```

   ~$ bitbake meta-toolchain

```
louis@ubuntu:~/project/fsl-community-bsp/build$ bitbake meta-toolchain
Parsing recipes:  86% |###################################      | ETA:  00:00:23
```

2. When you have created the toolchain into the Build Directory by following the above steps, you can find the file path: project/fsl-community-bsp/build/tmp/deploy/sdk

   Install the toolchain into your host system /opt directory.
   Note: Installing the toolchain requires root authorization
   ~$bash poky-glibc-x86_64-meta-toolchain-cortexa7hf-vfp-neon-toolchain-1.8.1.sh

```
ryan@Ubuntu:~/project/IFB112-Linux-bsp-1.0.0/IFB112-Linux-bsp-1.0.0/Toolchain/64-bit$ ls
poky-glibc-x86_64-meta-toolchain-cortexa7hf-vfp-neon-toolchain-1.8.1.sh
ryan@Ubuntu:~/project/IFB112-Linux-bsp-1.0.0/IFB112-Linux-bsp-1.0.0/Toolchain/64-bit$ bash poky-glibc-x86_
64-meta-toolchain-cortexa7hf-vfp-neon-toolchain-1.8.1.sh
Enter target directory for SDK (default: /opt/poky/1.8.1):
You are about to install the SDK to "/opt/poky/1.8.1". Proceed[Y/n]?y
Extracting SDK...done
Setting it up...done
SDK has been successfully set up and is ready to be used.
```

5.2 U-Boot for IFB125

5.2.1 Booting the System from eMMC (IFB125 default)

=> run bootcmd

```
Hit any key to stop autoboot:  0
=> run bootcmd
switch to partitions #0, OK
mmc1(part 0) is current device
switch to partitions #0, OK
mmc1(part 0) is current device
reading boot.scr
** Unable to read file boot.scr **
reading zImage
5263808 bytes read in 132 ms (38 MiB/s)
Booting from mmc ...
reading ax-rsb-imx6ul-ifb122.dtb
31768 bytes read in 18 ms (1.7 MiB/s)
Kernel image @ 0x80800000 [ 0x000000 - 0x5051c0 ]
## Flattened Device Tree blob at 83000000
   Booting using the fdt blob at 0x83000000
   Using Device Tree in place at 83000000, end 8300ac17

Starting kernel ...

Booting Linux on physical CPU 0x0
Linux version 3.14.52-RSB10X-003 (jrtiger@test-H97M-D3H) (gcc version 4.9.2 (GC6
CPU: ARMv7 Processor [410fc075] revision 5 (ARMv7), cr=10c53c7d
CPU: PIPT / VIPT nonaliasing data cache, VIPT aliasing instruction cache
```

5.2.2 Booting the Rescue System from eMMC

If the Embedded Linux system is crash and unable to boot, you can recovery the Linux system on u-boot through rescue mode.

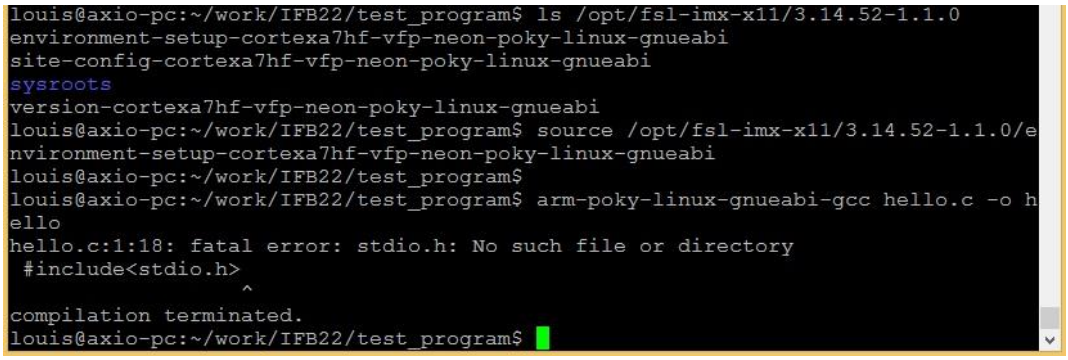
=> setenv script rescue.scr
=> run bootcmd

```
Hit any key to stop autoboot:  0
=> setenv script rescue.scr
=> run bootcmd
switch to partitions #0, OK
mmc1(part 0) is current device
switch to partitions #0, OK
mmc1(part 0) is current device
reading rescue.scr
805 bytes read in 12 ms (65.4 KiB/s)
Running bootscript from mmc ...
## Executing script at 80800000
=== Starting rescue/update system ===
reading rescue.img
5263808 bytes read in 132 ms (38 MiB/s)
reading rescue.dtb
31799 bytes read in 17 ms (1.8 MiB/s)
Kernel image @ 0x80800000 [ 0x000000 - 0x5051c0 ]
## Flattened Device Tree blob at 83000000
   Booting using the fdt blob at 0x83000000
   Using Device Tree in place at 83000000, end 8300ac36

Starting kernel ...

Booting Linux on physical CPU 0x0
```

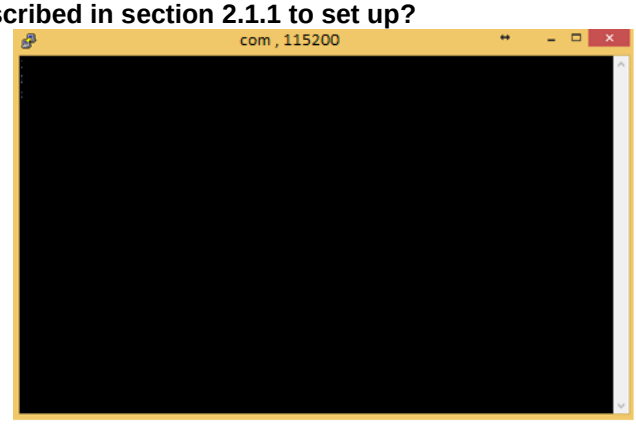
Appendix
Frequently Asked Questions

Q1. When I use toolchain to compile, I can't find the " include" file.

A1: Refer to section 2.3 and 2.2.2 Setting up the Cross-Development Environment for detailed information.
For example: $CC hello.c -o hello

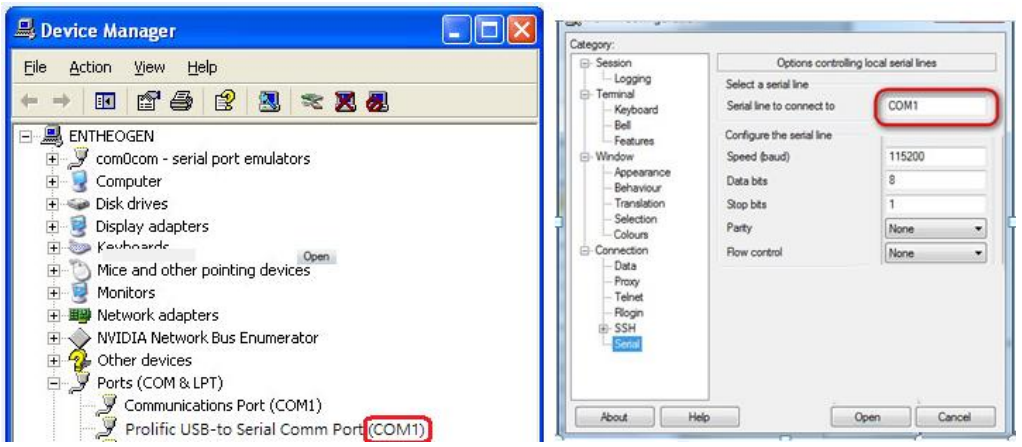


Q2. Why does the screen show nothing as below after I follow the steps described in section 2.1.1 to set up?



A2. Please follow the steps below:

1. Check your power.
2. Check that the name of the serial item "COM port" and the name of the "COM port" in the Device Manager menu are exactly the same as illustrated below.



3. Please check the COM port is RS232 in your PC..

Q3. Why can't I transfer the file to FTP、TFTP、NFS after following the instructions, or disconnection.

A3: Check whether your firewall has been blocked in your host PC or router.