AXIOMTEK

IRU151

Linux

Software User's Manual

RoHS

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.

September 2018, Version A3

Printed in Taiwan

Table of Contents

Chapter 1 Introduction

The ultra-compact IRU151-FL 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 IRU151-FL enables fast and efficient data computation, communication and acquisition. Besides, its compact size with Din-rail mounting allows for easy installation and control.

This user's manual is written for the embedded Linux preinstalled in the IRU151. The embedded Linux is derived from the Linux Yocto Board Support Package, which is based on Linux Kernel 3.14.52 and our hardware patches for use with the IRU151.

Software structure

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

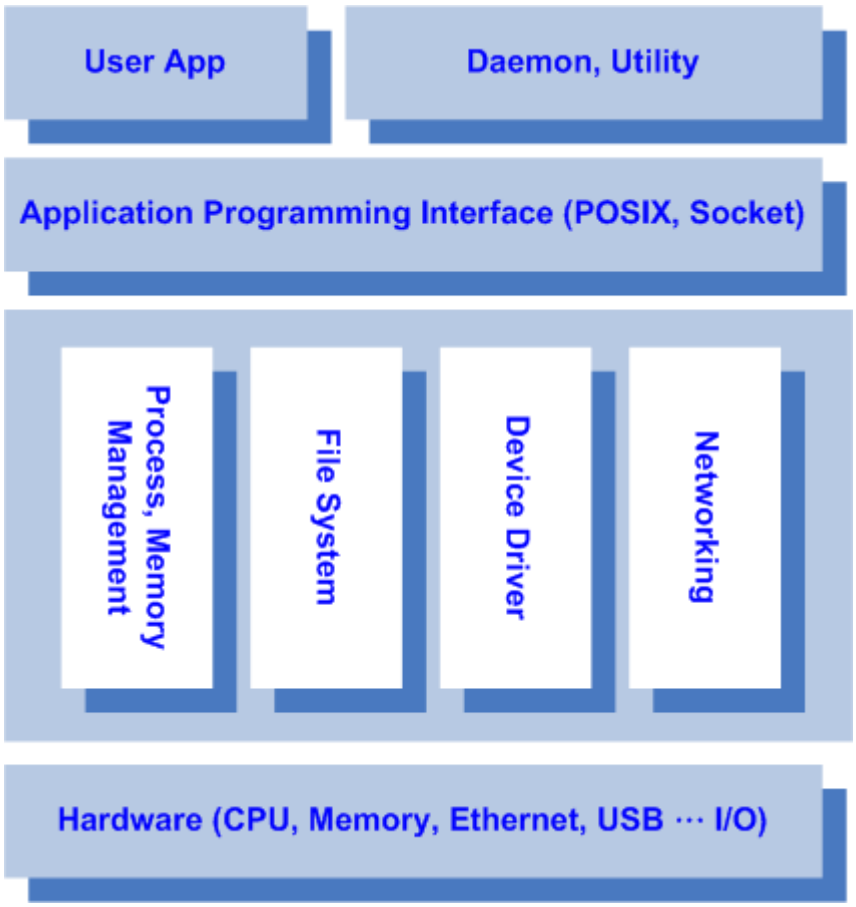
The IRU151 includes a 'librsb20x.so' shared library to facilitate user programs in monitoring and controlling I/O devices such as Watchdog Timer, DIP switch, USB power, and COM port Furthermore, the IRU151 provides software drivers for IRU151 USB data acquisition modules, such as the general configuration function group, the DIO function group and the analog input function group.

In addition to ext3 and ext4 file systems, this embedded Linux kernel is compiled with support for NFS, including the server-side and 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.

For connectivity, the Linux image includes the most popular internet protocols, some servers and utilities, not only making it easy to download/upload files (Linux kernel, application programs, etc.) and debug, but also facilitating communication to the outside world via Ethernet, WiFi and 3G.

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



1.1 Specifications

OS	Yocto Project 1.8.1 Fido
Kernel	Version : 3.14.52 (with NXP and Axiomtek's modified hardware patch)
Busybox	Version:1.23.1, a collection of standard Linux command-line utilities
Storage formats	Support FAT32/FAT/EXT2/EXT3/EXT4
Shell	Bash
BSP	IRU1A-Linux-bsp ● AxTools ● Image ● Yocto patches ● Toolchain ● Mfgtool
Protocol type	ICMP, TCP/IP, UDP, DHCP, Telnet, HTTP, HTTPS, SSL, SMTP, NTP, DNS, PPP, PPPoE, FTP, TFTP, NFS, OPC-UA, MQTT
Daemons	
Telnetd	Telnet server daemon
Ftpd	FTP server daemon
Sshd	Secure shell server
Pppd	Point-to-point protocol

Utilities	
Telnet	Telnet client program
FTP	FTP client program
TFTP	Trivial File Transfer Protocol client
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 ex2,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 on the state of the system, including the states of running processes and 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
Iperf	Network performance measurement tool
Xinetd	Manages internet-based connectivity
Openssh	Based on SSH protocol for remotely controlling or transferring files
Openssh-sftp	Secure File Transfer Protocol
Ntp	Network Time Protocol, used to synchronize time
Wvdial	Point-to-Point Protocol dialer
Curl	Transfer data tool
Mosquitto	Version : 1.5, provide MQTT broker and tools
Python	Version : 2.7, Python development environment
Nodejs	Version : 8.11.2, cross-platform JavaScript run-time environment
Gcc	Version : 4.9.2, cross compiler
G++	Version : 4.9.2, cross compiler
Development Environment	
Host OS/development	Ubuntu 14.04 LTS 32/64bit
Kernel	Version : 4.2.0-42
Toolchain	ARM, gcc-4.9.2 (Yocto project 1.8.1 Fido)
Machine running Ubuntu: the minimum hard disk space required is about 50 GB for the X11 backend. It is recommended that at least 120 GB is provided in order to have sufficient space to compile all backends together.	
Hardware's Library	
Data acquisition	- General configuration function - DIO function - Analog input function
Comport	- RS-232/422/485 mode setting (Default RS232)

Watchdog timer	- Enable Watch Dog Timer - Set Timer
DIP switch	- Get DIP switch status
USB Power	- Disable/Enable USB Power
Wi-Fi (Optional)	- Use a Wi-Fi module WPEQ-160ACN
3G (Optional)	- Use a 3G module Quectel UC20
4G (Optional)	- Use a 4G module MC7304, LARA-R211 or LARA-R280

Note

1. All specifications and images are subject to change without notice.

2. 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 (IRU-1A):**	Ex: **~#** /etc/run_rescue Meaning: To command /etc/run_rescue on IRU151

Chapter 2
Getting Started

2.1 Connecting the IRU151

The power

Please check the system power as below:

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

Pin	DC Signal Name
1	Power+
2	Power-
3	DI
4	DI_G

The console

Connect your computer to the IRU151 using a serial cable and change the switch to the Console mode (as shown below).

You can connect the IRU151 to a personal computer (PC) using one of the following connection types:

- Serial RS-232 console
- SSH over Ethernet

Note

Please download the IRU151 support package from Axiomtek's website listed below.

1. ***BSP and User's manual***
2. ***OPC UA application***
3. ***LabVIEW package for your host PC***

http://www.axiomtek.com/Default.aspx?MenuId=Products&FunctionId=ProductView&ItemId=24524&upcat=134&C=IRU151

-

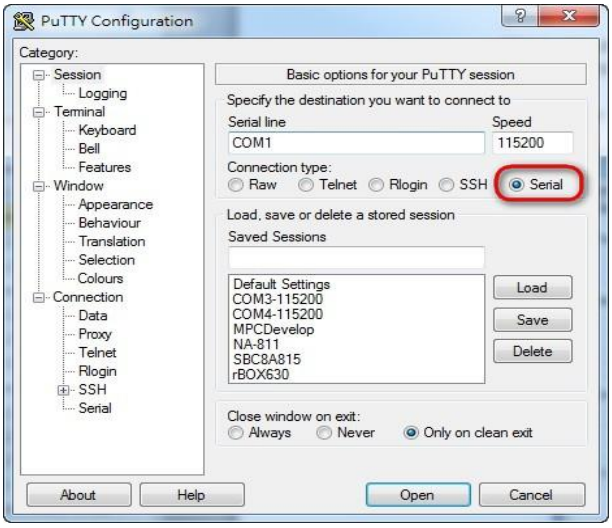
2.1.1 Serial Console

The serial console is a convenient interface for connecting the IRU151 to a PC. First of all, it is very important to make sure that your desktop connects to the IRU151 with a serial cable. Please set the system as follows:

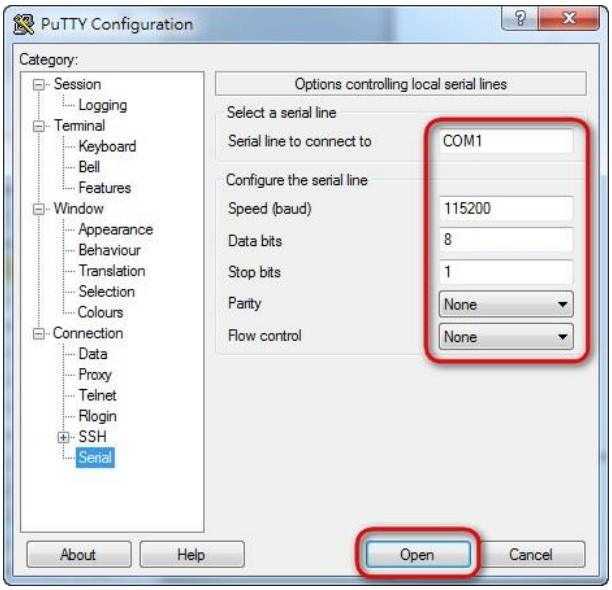
Baudrate: 115200 bps
Parity: None
Data bits: 8
Stop bits: 1
Flow Control: None

Use PuTTY to set up and link to the IRU151. Follow these step-by-step instructions:

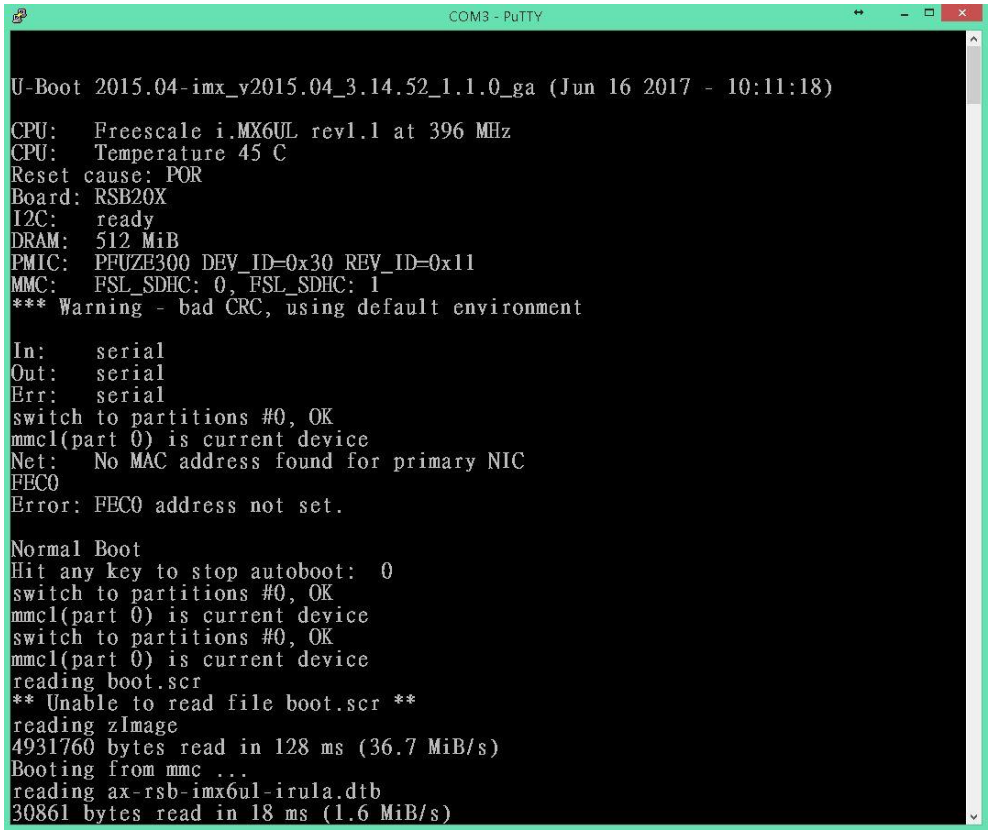
1. Open PuTTY and choose 'Serial' as the connection type.



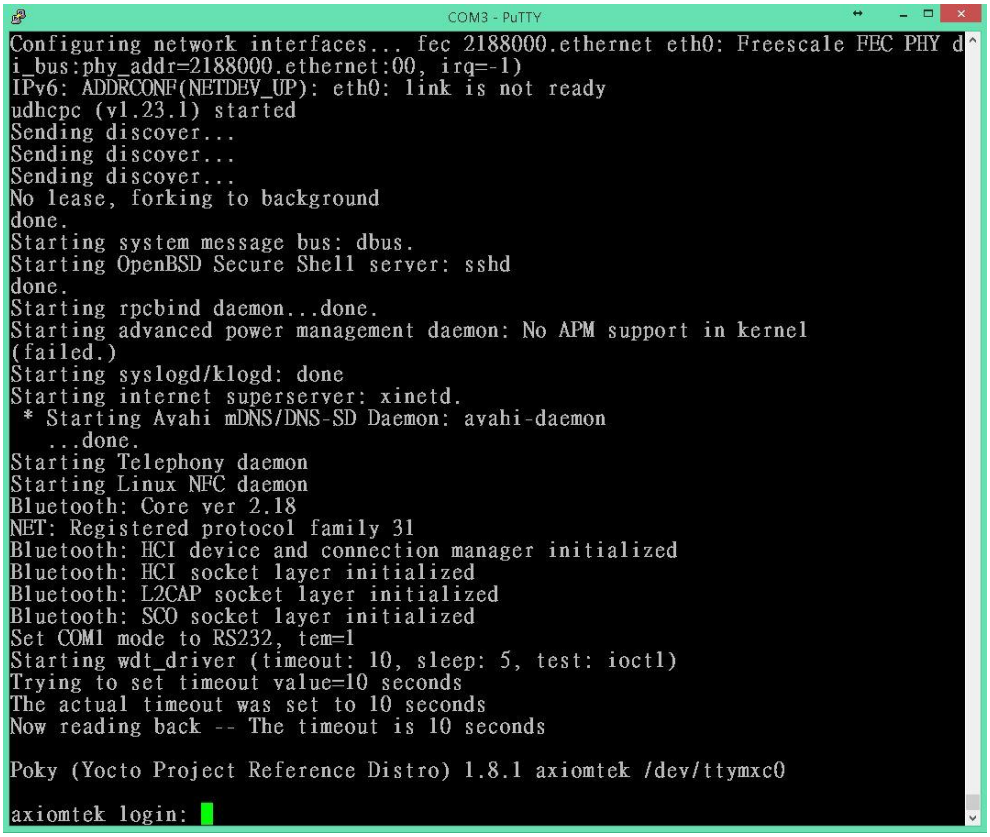
2. Configure the serial port correctly (see the image below). Click 'Open' and power on the IRU151.



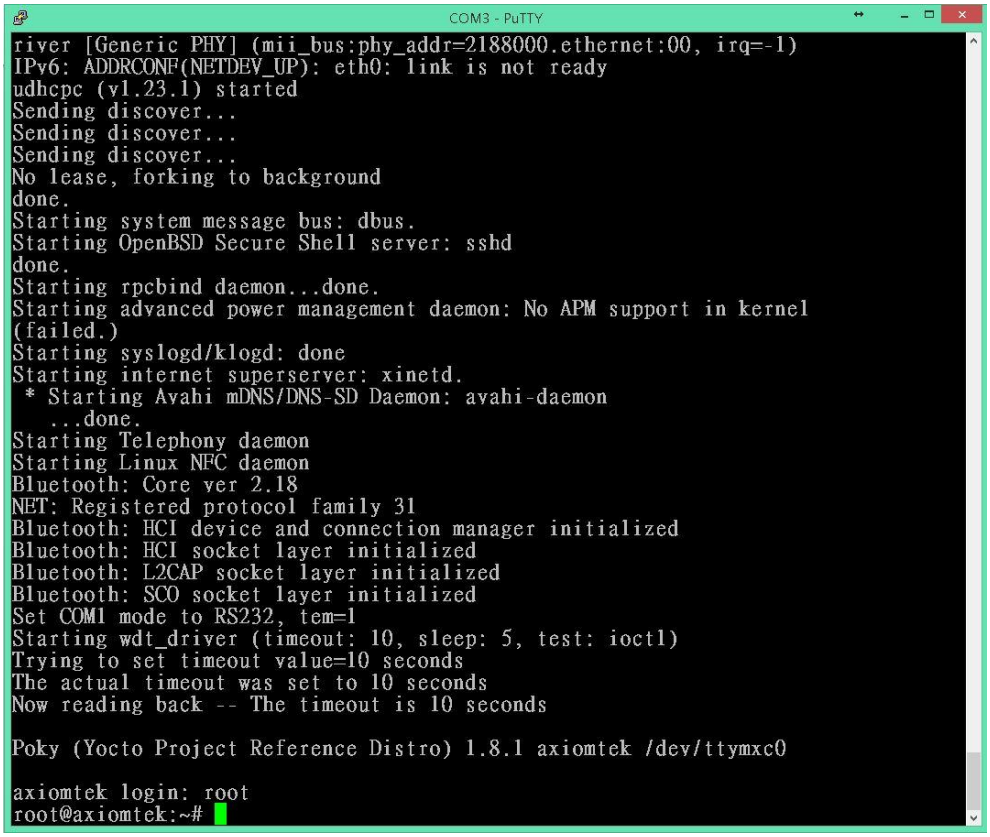
3. The data of the default Bootloader booting system from eMMC appears.



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



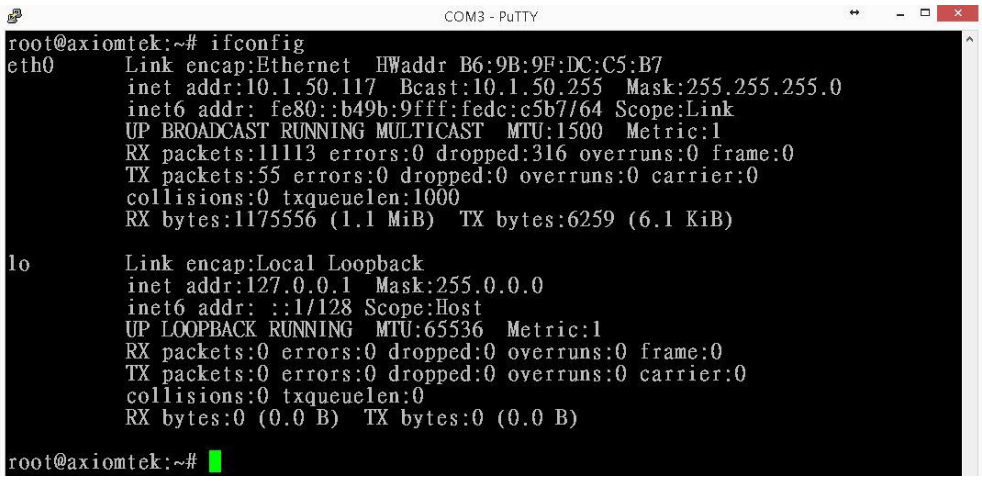
5. To log in, please enter 'root' (without a password).



2.1.2 SSH over Ethernet

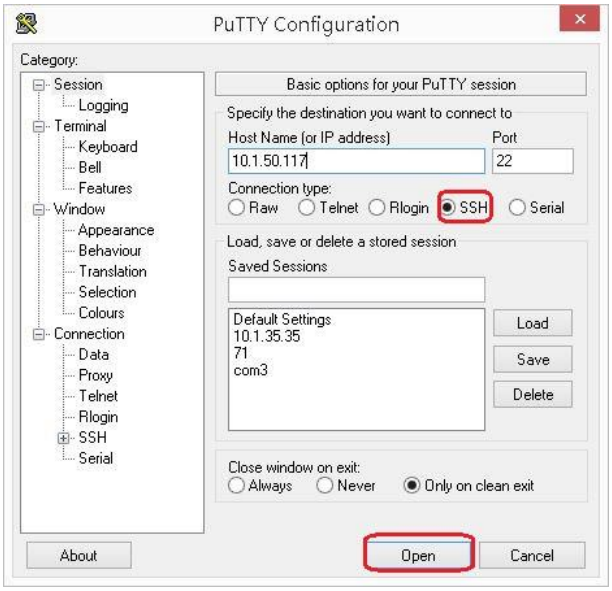
Follow the steps below to connect the IRU151 to a PC over Ethernet under the Windows® and Linux environments respectively.

Before starting SSH you have to check your LAN1 IP address, if you don't already know it.



For Windows® users:

1. Use PuTTY to setup and link. Open PuTTY and choose 'SSH' as the connection type. Then set the IP address to 10.1.50.117 and click 'Open'.



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

3. To log in to the IRU151, please enter 'root' (with no password).

For Linux users:

1. Open terminal and enter an 'ssh' command.
~$ ssh -l root 10.1.50.117

```
ryan@OMG:~$ ssh -l root 10.1.50.117
```

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

```
ryan@OMG:~$ ssh -l root 10.1.50.117
The authenticity of host '10.1.50.117 (10.1.50.117)' can't be established.
ECDSA key fingerprint is 19:44:21:77:ae:b2:36:c5:e6:25:f5:9e:25:af:93:ae.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '10.1.50.117' (ECDSA) to the list of known hosts.
Last login: Wed Jun 28 06:53:41 2017 from 10.1.35.67
root@axiomtek:~#
```

2.2 How to Develop a Sample Program

In this section, learn how to develop a sample program for the IRU151 with the following step-by-step instructions. The sample program is named 'hello.c'.

1. Create a directory for IRU151 BSP (IRU1A_Linux_x.x.x.zip).
~$ mkdir project
~$ cd project

```
ryan@Ubuntu:~$ mkdir project
ryan@Ubuntu:~$ cd project/
ryan@Ubuntu:~/project$ ls
IRU1A_Linux_V.1.0.1.zip
```

2. After extracting the file, you will find a directory IRU151-LINUX-bsp-x.x.x

```
ryan@Ubuntu:~/project$ cd IRU1A_Linux_V.1.0.1/IRU1A-LINUX-bsp_V.1.0.1/
ryan@Ubuntu:~/project/IRU1A_Linux_V.1.0.1/IRU1A-LINUX-bsp_V.1.0.1$ ls
AxTools  Image  README.txt  Toolchain  Yocto patches
```

Note

AxTools : This directory includes a hardware driver and an API library
Image : This directory includes kernel, rootfilesystem,dtb
Yocto patches : This directory includes IRU151 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. You can follow the steps below to install Yocto toolchain or refer to Chapter 5 "Board Support Package" to build the toolchain for the IRU151.

1. To check your Ubuntu version on your host PC.
   ~$ uname -m

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

```
ryan@Ubuntu:~$ uname -m
x86_64
```

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

```
ryan@Ubuntu:~/project/IRU1A_Linux_V.1.0.1/IRU1A-LINUX-bsp_V.1.0.1$ cd Toolchain/
ryan@Ubuntu:~/project/IRU1A_Linux_V.1.0.1/IRU1A-LINUX-bsp_V.1.0.1/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/IRU1A_Linux_V.1.0.1/IRU1A-LINUX-bsp_V.1.0.1/Toolchain$ cd 32-bit/
ryan@Ubuntu:~/project/IRU1A_Linux_V.1.0.1/IRU1A-LINUX-bsp_V.1.0.1/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/IRU1A_Linux_V.1.0.1/IRU1A-LINUX-bsp_V.1.0.1/Toolchain$ cd 64-bit/
ryan@Ubuntu:~/project/IRU1A_Linux_V.1.0.1/IRU1A-LINUX-bsp_V.1.0.1/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/IRU1A_Linux_V.1.0.1/IRU1A-LINUX-bsp_V.1.0.1/Toolchain$ cd 64-bit/
ryan@Ubuntu:~/project/IRU1A_Linux_V.1.0.1/IRU1A-LINUX-bsp_V.1.0.1/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):
```

5. Wait for installation.

```
ryan@Ubuntu:~/project/IRU1A_Linux_V.1.0.1/IRU1A-LINUX-bsp_V.1.0.1/Toolchain$ cd 64-bit/
ryan@Ubuntu:~/project/IRU1A_Linux_V.1.0.1/IRU1A-LINUX-bsp_V.1.0.1/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/IRU1A_Linux_V.1.0.1/IRU1A-LINUX-bsp_V.1.0.1/Toolchain$ cd 64-bit/
ryan@Ubuntu:~/project/IRU1A_Linux_V.1.0.1/IRU1A-LINUX-bsp_V.1.0.1/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 a cross-development environment, and then you can find this script in the directory you have chosen for installation.

1. To set up a 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-gnueab
i
```

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 --sysr
oot=/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

```
louis@ubuntu:~/project$ mkdir -p example
louis@ubuntu:~/project$ cd example/
louis@ubuntu:~/project/example$
```

2. Use vi to edit hello.c.

~$ vi 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, please do the following:

~$ $CC hello.c -o hello

```
ryan@Ubuntu:~/project/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:~/project/example$ ls -l
total 16
-rwxrwxr-x 1 ryan ryan 9669  6月 29 16:32 hello
-rw-rw-r-- 1 ryan ryan   72  6月 29 16:32 hello.c
```

5. Check whether the file ARM executable format is successful or not.
If it is successful, you will see the information below.

~$ file hello

```
ryan@Ubuntu:~/project/example$ file hello
hello: ELF 32-bit LSB executable, ARM, EABI5 version 1 (SYSV), dynamically linked, interprete
r /lib/ld-linux-armhf.so.3, for GNU/Linux 2.6.32, BuildID[sha1]=be20b9cf2ff90de9cabcd3db275fa
f058bd2961b, not stripped
```

2.3 How to Put and Run a Sample Program

This section shows how to put the 'hello' program into the IRU151 and execute it via FTP, a USB flash drive, and TFTP.

2.3.1 Via FTP

The IRU151 has a built-in FTP server. Users can put the 'hello' program into the IRU151 via FTP by following the steps below.

1. Enable FTPD daemon on the IRU151.
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 IRU151.
~# /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 the IRU151.
   ~$ ftp 10.1.50.117 (username 'root' without password)

```
ryan@Ubuntu:~/project/example$ ftp 10.1.50.117
Connected to 10.1.50.117.
220 Operation successful
Name (10.1.50.117:ryan): 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 to the IRU151 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 (214599.6 kB/s)
```

5. If the operation is successful on the IRU151, you will see the 'hello' program on the IRU151's */home/root* directory.

```
root@axiomtek:~# ls -l
-rw-r--r--    1 root     root          9669 Jun 28 08:58 hello
```

6. To change file permission for executable on the IRU151.
   ~# chmod a+x hello

```
root@axiomtek:~# chmod a+x hello
root@axiomtek:~# ls -l
-rwxr-xr-x    1 root     root          9669 Jun 28 08:58 hello
```

7. Run the 'hello' program on the IRU151.
   ~# ./hello

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

2.3.2 Via a USB Flash Drive

Users can put the 'hello' program into the IRU151 via a USB flash drive. Please follow the instructions below.

The IRU151 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 IRU151.

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 IRU151 via TFTP. Please follow the instructions below.

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

2. To copy the "hello" program to the "tftpboot" folder in your host PC
   ~$ 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 IRU151
   ~# tftp -g -r hello 192.168.0.3 (tftp server IP depends 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 IRU151
   ~# 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 IRU151.
   ~# ./hello

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

2.4 How to Recover the System

This section provides two methods for recovering the IRU151 system to default.

2.4.1 Via run_rescue System Script (under Linux System)

A recovery script is stored in the /etc folder on the IRU151 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

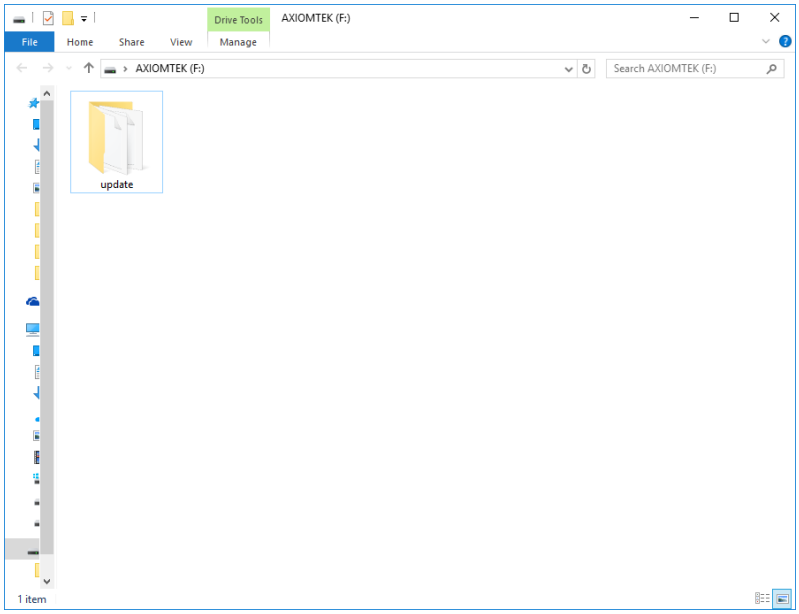
This section shows how to update the IRU151 using the recommended method below.

2.5.1 Via a USB Flash Drive

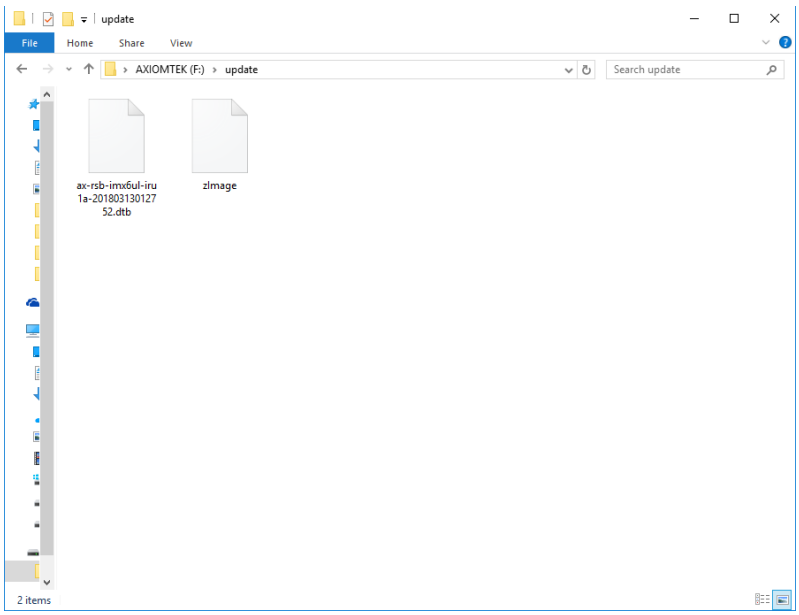
You can use a USB flash drive of DOS FAT32、EXT2、EXT3 or EXT4 formats, but an update folder must be stored on the first partition.

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

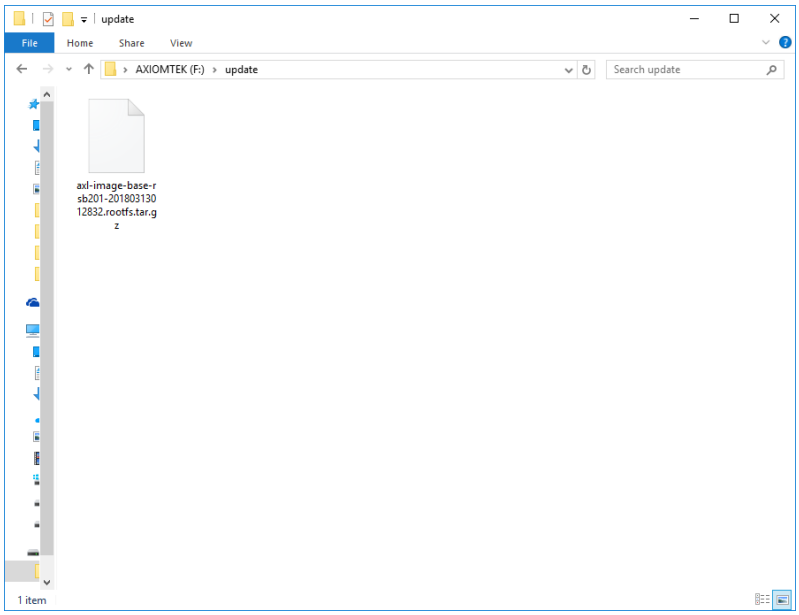
2. Create a folder named "update".



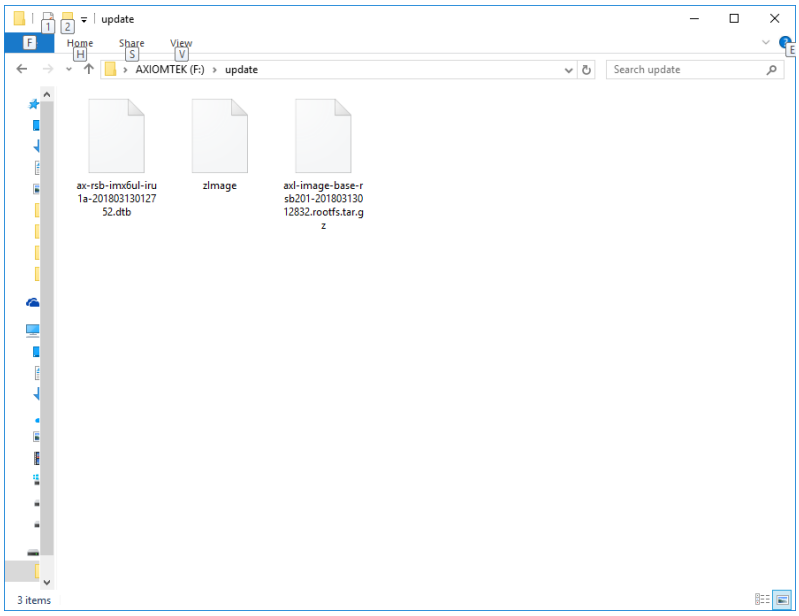
3. If you only want to update the kernel without altering the root filesystem, simply rename the new kernel file to 'zImage' and the dtb file to 'ax-rsb-imx6ul-iru1a.dtb' and then put the files in the update folder.



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

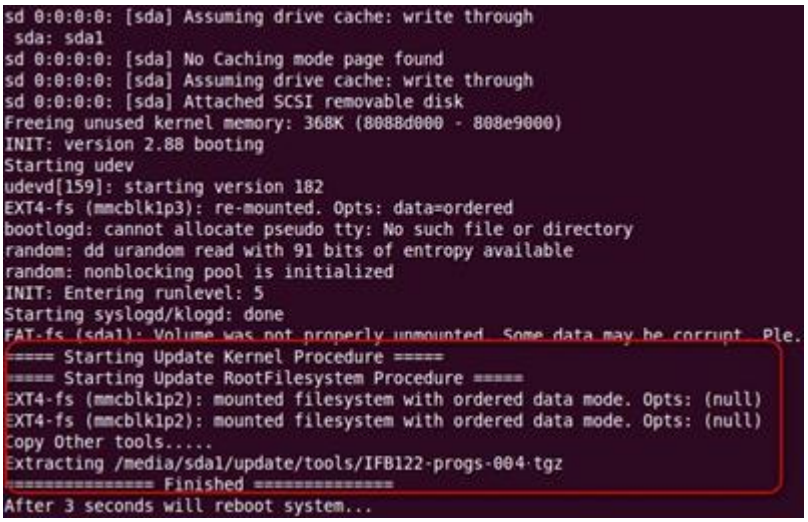


5. If you want to update both kernel and root filesystem, put the three files in the update folder.



6. Attach the USB flash drive to IRU151.

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

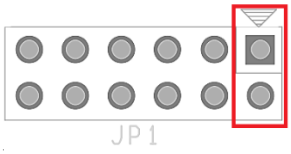


8. During this update procedure, four custom LEDs will blink like marquee. Until the procedure is finished, the system will reboot again automatically, and system status LED will change from blinking to always on.

2.6 How to use MFGtool to download image

We show you how to use MFG tool to download image to the IRU151 system.

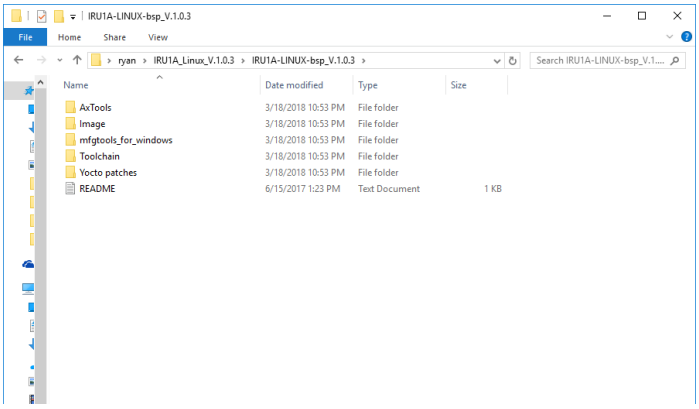
1. Before using the MFG tool, you have to change the IRU151 JP1 boot mode (default emmc boot) to OTG serial downloader mode



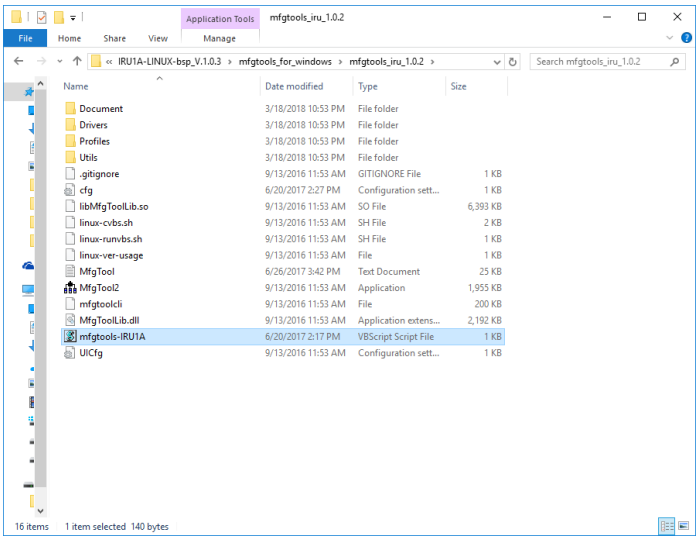
Please change JP3 to OTG Client mode as below.

After setting jumper, please connect the IRU151 to PC via USB cable.

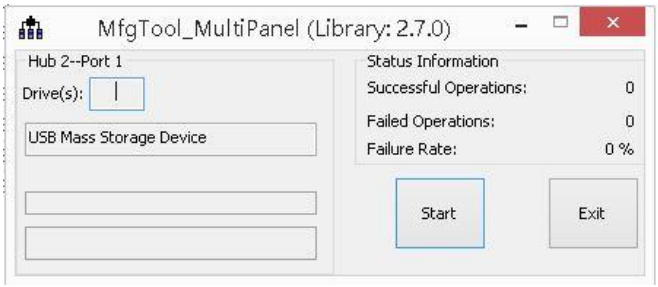
2. Extract Axiomtek's Yocto BSP and you will see mfgtools_iru_x.x.x in the mfgtools_for_windows directory



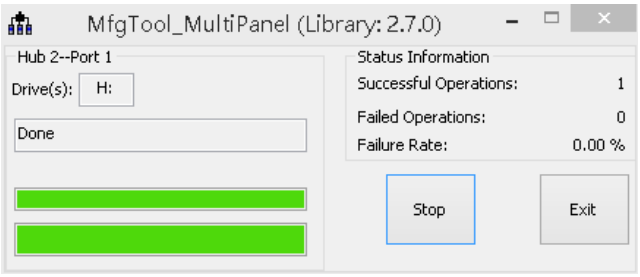
3. Enter mfgtools_for_windows/mfgtools_iru_x.x.x directory



4. After double clicking mfgtools-IRU1A.vbs, click "Start" to start burning



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



6. If your burning is successful, please set your jumper JP1 and JP3 to default setting. For detailed information about MFG tool, please refer to "Manufacturing Tool V2 Quick Start Guide.docx" in the "Document\V2" directory.

2.7 How to install Axiomtek-provided additional packages

This section shows how to install Axiomtek-provided additional packages, such as dotnet core, and OpenJDK. An installed script is stored in the /opt folder on the IRU152 Embedded Linux system. If you want to install these packages, please check your network and follow the instructions below.

2.7.1 Install .NET Core package

1. Run the setup_dotnet_core shell script.
   ~# /opt/setup_dotnet_core install

```
root@rsb201:~# /opt/setup_dotnet_core install
###axmsg: install.
###axmsg: Download dotnet-sdk-2.1.400.
Connecting to download.microsoft.com (104.124.16.61:443)
dotnet-sdk-2.1.400-l  45% |***************                 | 35818k  0:00:19 ETA
```

2. If installation is successful, you need to reboot system. Please press 'y' to reboot.

```
root@rsb201:~# /opt/setup_dotnet_core install
###axmsg: install.
###axmsg: Download dotnet-sdk-2.1.400.
Connecting to download.microsoft.com (104.124.16.61:443)
dotnet-sdk-2.1.400-l 100% |********************************| 78781k  0:00:00 ETA
axmsg: Download dotnet-sdk-2.1.400 success.
###axmsg: Create dotnet-sdk_env file.
###axmsg: Add /opt/dotnet/dotnet_env.sh to /etc/profile.
###axmsg: install dotnet-sdk-2.1.400.
###axmsg: install dotnet-sdk-2.1.400 finish.
###axmsg: Please reboot!!! to apply dotnet-sdk-2.1.400 environment variables.
###axmsg: reboot Now??? (y/n):y
```

3. Check dotnet version.
   ~# dotnet --version

```
root@rsb201:~# dotnet --version
2.1.400
```

4. If you want to uninstall, just follow the instructions below and reboot.
   ~# /opt/setup_dotnet_core uninstall

```
root@rsb201:~# /opt/setup_dotnet_core uninstall
###axmsg: uninstall dotnet-sdk-2.1.400.
###axmsg: uninstall dotnet-sdk-2.1.400 finish.
###axmsg: Please reboot!!! to apply dotnet-sdk-2.1.400 environment variables.
###axmsg: reboot Now??? (y/n):y
```

2.7.2 Install OpenJDK package

1. Run the setup_dotnet_core shell script.
   ~# /opt/setup_openjdk install

```
rsb201 login: root
root@rsb201:~# /opt/setup_openjdk install
###axmsg: install.
###axmsg: Download Open-JDK-1.8.0_162.
Connecting to cdn.azul.com (54.230.147.95:80)
ezdk-1.8.0_162-8.27.  19% |*****                          | 26021k  0:02:09 ETA
```

2. If installation is successful, you need to reboot system. Please press 'y' to reboot.

```
root@rsb201:~# /opt/setup_openjdk install
###axmsg: install.
###axmsg: Download Open-JDK-1.8.0_162.
Connecting to cdn.azul.com (54.230.147.95:80)
ezdk-1.8.0_162-8.27. 100% |********************************|   131M  0:00:00 ETA
axmsg: Download Open-JDK-1.8.0_162 success.
###axmsg: Create setting open-jdk_env file.
###axmsg: Add /usr/lib/jvm/openjdk_env.sh to /etc/profile.
###axmsg: install Open-JDK-1.8.0_162.
###axmsg: install Open-JDK-1.8.0_162 finish.
###axmsg: Please reboot!!! to apply Open-JDK-1.8.0_162 environment variables.
###axmsg: reboot Now??? (y/n):y
```

3. Check OpenJDK version.
   ~# java -version

```
root@rsb201:~# java -version
openjdk version "1.8.0_162"
OpenJDK Runtime Environment (Zulu Embedded 8.27.0.91-linux-aarch32hf) (build 1.8.0_162-b91)
OpenJDK Client VM (Zulu Embedded 8.27.0.91-linux-aarch32hf) (build 25.162-b91, mixed mode, Evaluation)
```

4. If you want to uninstall, just follow the instructions below and reboot.
   ~# /opt/setup_openjdk uninstall

```
root@rsb201:~# /opt/setup_openjdk uninstall
###axmsg: uninstall Open-JDK-1.8.0_162.
###axmsg: uninstall Open-JDK-1.8.0_162 finish.
###axmsg: Please reboot!!! to apply Open-JDK-1.8.0_162 environment variables.
###axmsg: reboot Now??? (y/n):y
```

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 IRU151.

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

```
root@axiomtek:~# uname -r
3.14.52-RSB20X-001
root@axiomtek:~#
```

Check root filesystem with the login screen:

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

axiomtek login:
```

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 the IRU151:

~# date

```
root@axiomtek:~# date
Wed Jun 28 09:28:10 UTC 2017
```

3.1.3 Internal RTC Time

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

Read internal RTC time with the following command on the IRU151:
~# hwclock -r --rtc=/dev/rtc1

```
root@axiomtek:~# hwclock -r --rtc=/dev/rtc1
Thu Jan  1 03:04:20 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
Wed Jun 28 09:29:27 2017  0.000000 seconds
```

3.1.5 Watchdog timer

Function: wdt_driver_test.out
Description: When <sleep> parameters exceed <timeout> parameters, watchdog timer will be triggered

Note: The IRU151 has been enabled for default settings, and the default parameters are ***10 5 0***
Commands example: ~# wdt 10 5 0 &

```
root@axiomtek:~# 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 date 201509161714.05

```
root@axiomtek:~# date -s 201706291200.05
Thu Jun 29 12:00:05 UTC 2017
```

2. Write sync time to internal RTC

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

```
root@axiomtek:~# hwclock -w --rtc=/dev/rtc1
root@axiomtek:~# hwclock -r --rtc=/dev/rtc1
Thu Jun 29 12:01:01 2017  0.000000 seconds
```

3. Write sync time to external RTC

~# hwclock -w

```
root@axiomtek:~# hwclock -w
root@axiomtek:~# hwclock -r
Thu Jun 29 12:02:05 2017  0.000000 seconds
```

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 IRU151 comes with a built-in FTP server. Section 2.1 shows the steps to put the 'hello' program in the IRU151 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, so does its accompanying bootloader U-boot.

Please refer to Chapter 5 for 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 NFS – Network File System

NFS enables you to export a directory on an NFS server and mount that directory on a remote client machine as if it were a local file system. Using NFS on a target machine, we can have access to a huge number of files, libraries, and utilities during development and debugging, as well as booting up kernel.

This embedded Linux kernel is compiled with support for NFS, including server-side, client-side functionality and 'Root file system on NFS'.

3.2.4 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 your 3G module is Quectel UC20, follow the instructions below.

To create a wvdial config

~# vi /etc/wvdial.conf

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

Please enter your information as 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, please 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/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)
```

3.2.5 How to use a 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 librsb20x API Functions

The IRU151 BSP includes a 'librsb10x.so' shared library for users to access I/O and read back system information. This shared library is kept in BSP, which you can find in IRU1A-rsb-lib-x.x.x.tar.gz of AxTools. When extracting the compressed file, besides the shared library you will also see a *demo* folder containing an API header file and example programs.

Summary table of available API functions

No.	Function	Description
1	Control_WDT()	Set WDT function
2	Get_DIP_Switch	Get DIP Switch value
3	Control_USB_PWR_EN	Control USB OTG Power enable
4.	Control_EXP_RST	Control Expansion Board reset
5	Control_WIFI_LINK_LED	Set MINICARD WIFI link led

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


int main(int argc, char *argv[])
{
        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 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);
	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);


	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


int main(int argc, char *argv[])
{
        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("example : ./comWrite 1 (1=232, 2=485, 3=422)\n");
        fd = open("/dev/ttymxc1", 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_DIP_Switch()

Function	**int Get_DIP_Switch(int *data);**
Description	Read DIP Switch value
Arguments	data: DIP Switch value
Return	0: No error. 1: Function fails.
Others	None.

SAMPLE CODE:

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <linux/types.h>
#include "librsb20x.h"

int main(int argc, char* argv[]) {
      int xch;
      Get_DIP_Switch(&xch);
      printf("%d\n", xch);
      return 0;
}
```

Function: Control_USB_PWR_EN()

Function	**int Control_USB_PWR_EN (int data);**
Description	Control USB OTG Power
Arguments	data: 0 : Disable 1 : Enable
Return	0: No error. 1: Function fails.
Others	None.

SAMPLE CODE:

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <linux/types.h>
#include "librsb20x.h"

static void help(void) {
        fprintf(stderr,
                "Usage: command [EN] \n"
                "    EN : 0/1 \n");
                exit(1);
}

int main(int argc, char* argv[])
{
        if( argc !=2 || !(atoi(argv[1])==0 || atoi(argv[1])==1) )
                help();
        int value=atoi(argv[1]);
        Contrl_USB_PWR_EN(value);
        if(value==1)
                printf("Enable USB Power\n");
        else if(value==1)
                printf("Disable USB Power\n");
    return 0;
}
```

Function: Control_EXP_RST ()

Function	**int Control_EXP_RST(void)**
Description	Contrl Expansion Board Reset.
Arguments	none
Return	0: No error. 1: Function fails.
Others	None.

SAMPLE CODE:

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <linux/types.h>
#include "librsb20x.h"

int main(int argc, char* argv[])
{
        Contrl_EXP_RST();
    return 0;
}
```

Function: Control_WIFI_LINK_LED ()

Function	**Int Control_WIFI_LINK_LED(int num, int data);**
Description	Set Minicard wifi link led
Arguments	Num: 1 : Minicard 1 2 : Minicard 2 data: 0 : Disable 1 : Enable
Return	0: No error. 1: Function fails.
Others	None.

SAMPLE CODE:

```
#include <errno.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <linux/types.h>
#include "librsb20x.h"

static void help(void) __attribute__ ((noreturn));

static void help(void) {
        fprintf(stderr,
                "Usage: command [NUM] [ENABLE]\n"
                "   NUM     : 1/2 \n"
                "   ENABLE: 0/1 \n");
                exit(1);
}

int main(int argc, char *argv[]) {
        if(argc  !=3  ||  !(atoi(argv[1])==1  ||  atoi(argv[1])==2)  ||  !(atoi(argv[2])==0  ||
atoi(argv[2])==1) )
                help();
        int num      = atoi(argv[1]);
        int enable = atoi(argv[2]);
        printf("Set WIFI_%d ,link led=%d\n", num, enable);
        Contrl_WIFI_LINK_LED(num, enable);
        exit(0);
}
```

Function: Control_WDT ()

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

SAMPLE CODE:

```
#include <stdio.h>
#include <stdlib.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 IRU151 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@Ubuntu:~$ source /opt/poky/1.8.1/environment-setup-cortexa7hf-vfp-neon-poky
-linux-gnueabi
```

2. To compile and build a demo program for the IRU151, please do the following:
Change to *your project* directory.
~$ cd project/IRU1A-Linux_V.X.X.X/IRU1A-LINUX-V.X.X.X/AxTools

```
ryan@Ubuntu:~$ cd project/IRU1A_Linux_V.1.0.1/IRU1A-LINUX-bsp_V.1.0.1/AxTools/
ryan@Ubuntu:~/project/IRU1A_Linux_V.1.0.1/IRU1A-LINUX-bsp_V.1.0.1/AxTools$ ls
IRU1A-progs-001.tgz  IRU1A-rsb-lib-1.0.1.tar.gz
```

3. Extract driver source to *your project* directory.
~$ tar -zxv -f IRU1A-rsb-lib-1.0.x.tar.gz

```
ryan@Ubuntu:~/project/IRU1A_Linux_V.1.0.1/IRU1A-LINUX-bsp_V.1.0.1/AxTools$ tar -
zxv -f IRU1A-rsb-lib-1.0.1.tar.gz -C .
rsb_lib/demo/usb_pwr_en
rsb_lib/librsb20x.so.1.0.1
rsb_lib/demo/diotool
rsb_lib/demo/com_mode.c
rsb_lib/demo/dipswitch
rsb_lib/demo/com_mode
rsb_lib/demo/Makefile
rsb_lib/demo/com_port_open
rsb_lib/demo/led_wifi_link.c
rsb_lib/demo/diotest.c
rsb_lib/demo/com_port_open.c
rsb_lib/demo/diotool.c
rsb_lib/demo/led_wifi_link
rsb_lib/demo/serial.h
rsb_lib/librsb20x.so.0
rsb_lib/demo/exp_rst
rsb_lib/demo/wdttest.c
rsb_lib/demo/
rsb_lib/demo/exp_rst.c
rsb_lib/librsb20x.h
rsb_lib/demo/diotest
rsb_lib/
rsb_lib/demo/usb_pwr_en.c
rsb_lib/demo/librsb20x.h
rsb_lib/librsb20x.so
rsb_lib/demo/wdttest
rsb_lib/demo/dipswitch.c
```

4. Change to *rsb_lib/demo* directory.
~$ cd ~/project/rsb_lib/demo

```
ryan@Ubuntu:~/project/IRU1A_Linux_V.1.0.1/IRU1A-LINUX-bsp_V.1.0.1/AxTools$ cd rs
b_lib/demo/
ryan@Ubuntu:~/project/IRU1A_Linux_V.1.0.1/IRU1A-LINUX-bsp_V.1.0.1/AxTools/rsb_li
b/demo$ ls
```

5. Build the demo program.

~$ make

```
ryan@Ubuntu:~/project/IRU1A_Linux_V.1.0.1/IRU1A-LINUX-bsp_V.1.0.1/AxTools/rsb_li
b/demo$ make
arm-poky-linux-gnueabi-gcc  -march=armv7-a -mfloat-abi=hard -mfpu=neon -mtune=co
rtex-a7 --sysroot=/opt/poky/1.8.1/sysroots/cortexa7hf-vfp-neon-poky-linux-gnueab
i  -o com_mode com_mode.c -lrsb20x -L../
arm-poky-linux-gnueabi-gcc  -march=armv7-a -mfloat-abi=hard -mfpu=neon -mtune=co
rtex-a7 --sysroot=/opt/poky/1.8.1/sysroots/cortexa7hf-vfp-neon-poky-linux-gnueab
i  -o com_port_open com_port_open.c -lrsb20x -L../
arm-poky-linux-gnueabi-gcc  -march=armv7-a -mfloat-abi=hard -mfpu=neon -mtune=co
rtex-a7 --sysroot=/opt/poky/1.8.1/sysroots/cortexa7hf-vfp-neon-poky-linux-gnueab
i  -o dipswitch dipswitch.c -lrsb20x -L../
arm-poky-linux-gnueabi-gcc  -march=armv7-a -mfloat-abi=hard -mfpu=neon -mtune=co
rtex-a7 --sysroot=/opt/poky/1.8.1/sysroots/cortexa7hf-vfp-neon-poky-linux-gnueab
i  -o diotest diotest.c -lrsb20x -L../
arm-poky-linux-gnueabi-gcc  -march=armv7-a -mfloat-abi=hard -mfpu=neon -mtune=co
rtex-a7 --sysroot=/opt/poky/1.8.1/sysroots/cortexa7hf-vfp-neon-poky-linux-gnueab
i  -o diotool diotool.c -lrsb20x -L../
arm-poky-linux-gnueabi-gcc  -march=armv7-a -mfloat-abi=hard -mfpu=neon -mtune=co
rtex-a7 --sysroot=/opt/poky/1.8.1/sysroots/cortexa7hf-vfp-neon-poky-linux-gnueab
i  -o exp_rst exp_rst.c -lrsb20x -L../
arm-poky-linux-gnueabi-gcc  -march=armv7-a -mfloat-abi=hard -mfpu=neon -mtune=co
rtex-a7 --sysroot=/opt/poky/1.8.1/sysroots/cortexa7hf-vfp-neon-poky-linux-gnueab
i  -o usb_pwr_en usb_pwr_en.c -lrsb20x -L../
arm-poky-linux-gnueabi-gcc  -march=armv7-a -mfloat-abi=hard -mfpu=neon -mtune=co
rtex-a7 --sysroot=/opt/poky/1.8.1/sysroots/cortexa7hf-vfp-neon-poky-linux-gnueab
i  -o wdttest wdttest.c -lrsb20x -L../
arm-poky-linux-gnueabi-gcc  -march=armv7-a -mfloat-abi=hard -mfpu=neon -mtune=co
rtex-a7 --sysroot=/opt/poky/1.8.1/sysroots/cortexa7hf-vfp-neon-poky-linux-gnueab
i  -o led_wifi_link led_wifi_link.c -lrsb20x -L../
```

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

```
ryan@Ubuntu:~/project/IRU1A_Linux_V.1.0.1/IRU1A-LINUX-bsp_V.1.0.1/AxTools/rsb_li
b/demo$ ls
com_mode         diotest.c    exp_rst          Makefile      wdttest.c
com_mode.c       diotool      exp_rst.c        serial.h
com_port_open    diotool.c    led_wifi_link    usb_pwr_en
com_port_open.c  dipswitch    led_wifi_link.c  usb_pwr_en.c
diotest          dipswitch.c  librsb20x.h      wdttest
```

4.2.2 Run a demo program

Refer to section 2.3 for detailed information.

This page is intentionally left blank.

Chapter 5
Board Support Package (BSP)

5.1 Host Development System Installation

5.1.1 Install Host System

1. Download 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 \
        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 \
        gcc-multilib g++-multilib
```

   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

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

   To create 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 NFS server:
~$ sudo aptitude -y install nfs-common nfs-kernel-server portmap

To configure 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 youhave 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 needed if the bin folder already exists)
~$ curl http://commondatastorage.googleapis.com/git-repo-downloads/repo > ~/bin/repo
~$ chmod a+x ~/bin/repo

```
louis@ubuntu:~$ mkdir ~/bin
louis@ubuntu:~$ 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 26223  100 26223    0     0   3345      0  0:00:07  0:00:07 --:--:--  5815
louis@ubuntu:~$ 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

```
louis@ubuntu:~$ export PATH=~/bin:$PATH
```

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

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

3. Download the Freescale's Yocto BSP source
~$ mkdir fsl-community-bsp
~$ cd 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

```
louis@ubuntu:~$ mkdir project/fsl-community-bsp
louis@ubuntu:~$ cd project/fsl-community-bsp/
louis@ubuntu:~/project/fsl-community-bsp$ repo init -u git://git.freescale.com/i
mx/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
```

~$ repo sync

```
louis@ubuntu:~/project/fsl-community-bsp$ repo sync
Fetching project fsl-community-bsp-base
Fetching project meta-openembedded
remote: Counting objects: 215, done.
remote: Total 215 (delta 0), reused 0 (delta 0), pack-reused 215
Receiving objects: 100% (215/215), 46.16 KiB | 0 bytes/s, done.
Resolving deltas: 100% (114/114), done.
From git://github.com/Freescale/fsl-community-bsp-base
 * [new branch]      daisy      -> freescale/daisy
 * [new branch]      danny      -> freescale/danny
```

Clone Finish

```
 * [new tag]         yocto-2.0  -> yocto-2.0
 * [new tag]         yocto-2.0.1 -> yocto-2.0.1
 * [new tag]         yocto-2.0.2 -> yocto-2.0.2
 * [new tag]         yocto-2.1  -> yocto-2.1
 * [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.

louis@ubuntu:~/project/fsl-community-bsp$
```

4. Extract Axiomtek's Yocto BSP source

~$ tar -xvf ../IRU1A-LINUX-bsp_1.0.X/Yocto\ patches/IRU1A-meta-axiomtek-x.x.x.tar.gz -C sources

```
louis@ubuntu:~/project/fsl-community-bsp$ tar -xvf ../IFB122-LINUX-bsp-1.0.0/Yoc
to\ patches/meta-axiomtek-2.5.0.tar.gz -C sources
```

Check meta-axiomtek

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

5. Update bblayers.conf

~$ vi fsl-community-bsp/sources/base/conf/bblayers.conf

And add this line 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=rsb201 source fsl-setup-release.sh -b build

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

Your build environment has been configured with:

    MACHINE=rsb201
    SDKMACHINE=i686
    DISTRO=poky
    EULA=1
BSPDIR=
BUILD_DIR=.
louis@ubuntu:~/project/fsl-community-bsp/build$
```

Start to build image

~$ bitbake axl-image-base

```
louis@ubuntu:~/project/fsl-community-bsp/build$ bitbake axl-image-base
Parsing recipes:   7% |###                                      | ETA:  00:02:32
```

7. After image is built successfully, you can find the file path.

The file path: project/fsl-community-bsp/build/tmp/deploy/images/rsb201

```
ryan@Ubuntu:~/fsl-community-bsp/build/tmp/deploy/images/rsb201$ ls
axl-image-base-rsb201-20170616062346.rootfs.manifest
axl-image-base-rsb201-20170616062346.rootfs.tar.gz
axl-image-base-rsb201.manifest
axl-image-base-rsb201.tar.gz
modules--3.14.52-r0-rsb201-20170616062330.tgz
modules-rsb201.tgz
README_-_DO_NOT_DELETE_FILES_IN_THIS_DIRECTORY.txt
u-boot.imx
u-boot-rsb201.imx
u-boot-sd-2015.04-r0.imx
zImage
zImage--3.14.52-r0-ax-rsb-imx6ul-iru1a-20170616062330.dtb
zImage--3.14.52-r0-rsb201-20170616062330.bin
zImage-ax-rsb-imx6ul-iru1a.dtb
zImage-rsb201.bin
```

5.1.3 Build and install the user's Yocto Toolchain

We have provided Yocto Toolchain in IRU151 BSP. However, if you want to build your toolchain through Yocto development, you can follow the instructions on your host PC:

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

```
louis@ubuntu:~/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
```

   ~$ bitbake meta-toolchain

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

2. After following these steps to generate the toolchain into the Build Directory, you will 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

```
louis@ubuntu:~/project/fsl-community-bsp/build/tmp/deploy/sdk$ bash poky-glibc-x
86_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
[sudo] password for louis:
Extracting SDK...done
Setting it up...done
SDK has been successfully set up and is ready to be used.
louis@ubuntu:~/project/fsl-community-bsp/build/tmp/deploy/sdk$
```

5.2 U-Boot for the IRU151

5.2.1 Booting the system from eMMC (IRU151 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 damaged and unable to boot, you can recover the Linux system on u-boot through the 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
```

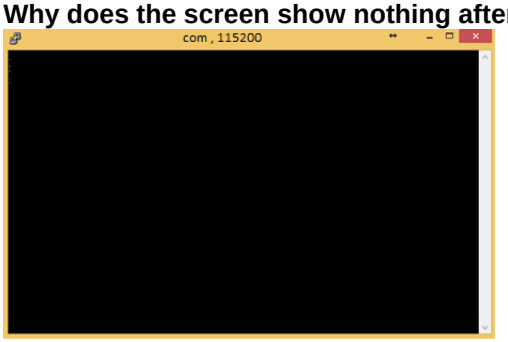
Appendix
Frequently Asked Questions

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

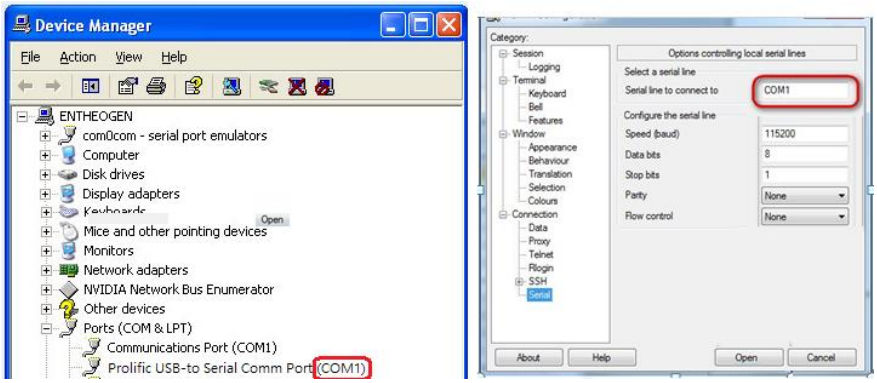
```
louis@axio-pc:~/work/IFB22/test_program$ ls /opt/fsl-imx-x11/3.14.52-1.1.0
environment-setup-cortexa7hf-vfp-neon-poky-linux-gnueabi
site-config-cortexa7hf-vfp-neon-poky-linux-gnueabi
sysroots
version-cortexa7hf-vfp-neon-poky-linux-gnueabi
louis@axio-pc:~/work/IFB22/test_program$ source /opt/fsl-imx-x11/3.14.52-1.1.0/e
nvironment-setup-cortexa7hf-vfp-neon-poky-linux-gnueabi
louis@axio-pc:~/work/IFB22/test_program$
louis@axio-pc:~/work/IFB22/test_program$ arm-poky-linux-gnueabi-gcc hello.c -o h
ello
hello.c:1:18: fatal error: stdio.h: No such file or directory
 #include<stdio.h>
                  ^
compilation terminated.
louis@axio-pc:~/work/IFB22/test_program$
```

Q2. Why does the screen show nothing after I follow section 2.1.1 to set up?



A2. Please follow the steps below.

1. Check your power.
2. Make sure that the serial item "COM port" and Device Manager "COM port" are showing the same name, as illustrated below.



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

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

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