



AXIOMTEK

IRU131

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.

©Copyright 2018 Axiomtek Co., Ltd.

All Rights Reserved

September 2018, Version A3

Printed in Taiwan

Table of Contents

Disclaimers.....	ii
Chapter 1 Introduction	1
1.1 Specifications.....	2
Chapter 2 Getting Started.....	5
2.1 Connecting the IRU131	5
2.1.1 Serial Console	6
2.1.2 SSH over Ethernet	8
2.2 How to Develop a Sample Program	10
2.2.1 Install Yocto Toolchain	11
2.2.2 Setting Up the Cross-Development Environment	12
2.2.3 Write and Compile Sample Program.....	12
2.3 How to Put and Run a Sample Program.....	13
2.3.1 Via FTP	13
2.3.2 Via USB Flash Drive.....	15
2.3.3 Via TFTP	16
2.4 How to Recovery System	17
2.4.1 Via run_rescue System Script (under Linux System)	17
2.4.2 Via rescue.scr Script (under u-boot)	17
2.5 How to Update System	18
2.5.1 Via USB Flash Drive.....	18
2.6 How to use MFG tool to download image	21
2.7 How to install Axiomtek-provided additional packages.....	22
2.7.1 Install .NET Core package	22
2.7.2 Install OpenJDK package.....	23
Chapter 3 The Embedded Linux.....	25
3.1 Embedded Linux Image Managing	25
3.1.1 System Version	25
3.1.2 System Time.....	25
3.1.3 Internal RTC Time	25
3.1.4 External RTC Time	26
3.1.5 Watchdog timer	26
3.1.6 Adjusting System Time.....	26
3.2 Networking.....	27
3.2.1 FTP – File Transfer Protocol	27
3.2.2 TFTP – Trivial File Transfer Protocol.....	27

3.2.3	NFS – Network File System	27
3.2.4	How to use 3G or 4G module (Optional)	27
3.2.5	How to use Wi-Fi module (Optional)	32
Chapter 4 Programming Guide		35
4.1	librsb20x API Functions	35
4.2	Compile Demo Program	45
4.2.1	Install IRU131 I/O Library	45
4.2.2	Run demo program	46
Chapter 5 Board Support Package (BSP)		47
5.1	Host Development System Installation	47
5.1.1	Install Host System	47
5.1.2	Install Yocto Development	48
5.2	U-Boot for IRU131	52
5.2.1	Booting the System from eMMC (IRU131 default)	52
5.2.2	Booting the Rescue System from eMMC	52
Appendix Frequently Asked Questions		53

Chapter 1

Introduction

The extreme compact IRU131-FL supports the low power RISC-based module (i.MX6UL) processor with extended temperature range of -40°C to +70°C for using in wide range operating environments. Multiple built-in serial ports, high-speed LANs and USB 2.0 ports enable fast and efficient data computation, communication and acquisition. Its digital I/O feature provides users with the convenience of digital devices connection. Besides, Its compact size with Din-rail mounting allows for easy installation into control

This user's manual is for the embedded Linux preinstalled in IRU131. The embedded Linux is derived from Linux Yocto Board Support Package, which is based on Linux Kernel 3.14.52 and our hardware patches to suit IRU131.

Software structure

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

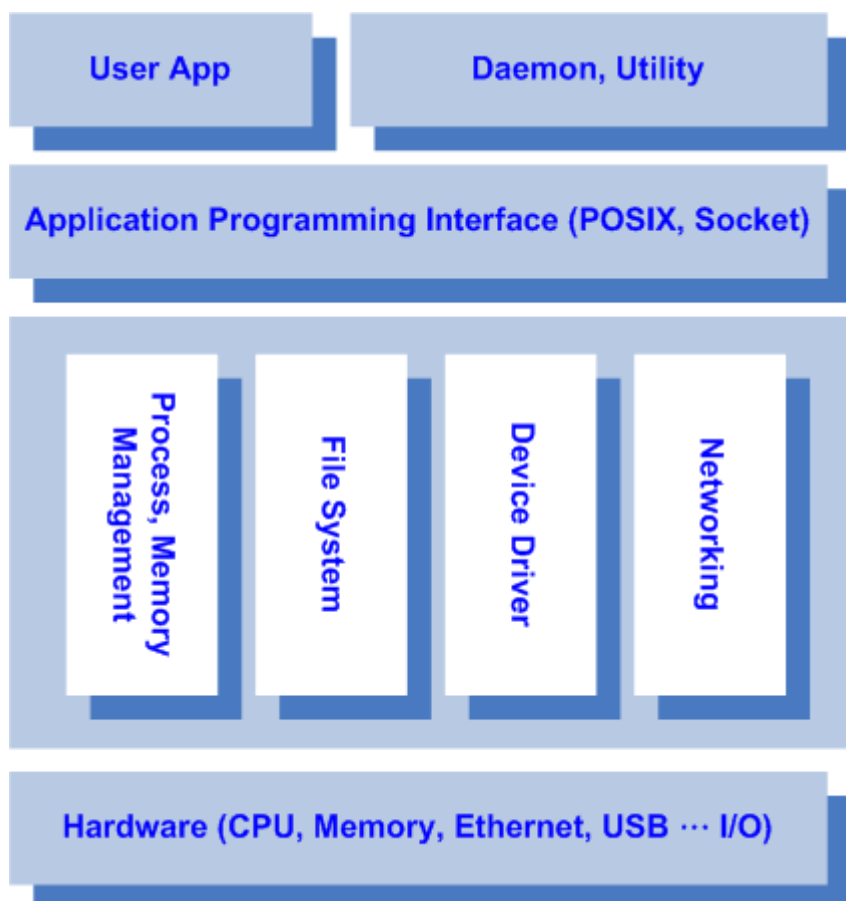
To facilitate user program in monitoring and controlling I/O device such as DIO, Watchdog Timer, DIP switch, USB power, comport , the IRU131 includes 'librsb20x.so' shared library.

In addition to ext3 and ext4 file system, 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 we have several advantages such as:

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

For connectivity, this image includes most popular internet protocols, some servers and utilities not only making it easy for downloading/uploading files (Linux kernel, application program) or for debugging, but also communicating to outside world via Ethernet, WiFi and 3G.

For the convenience of manipulating embedded Linux, this image includes lots 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 <ul style="list-style-type: none"> ● 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	
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 subversion on host PC
~#	Target (IRU-1A):	Ex: ~# /etc/run_rescue Meaning: To command /etc/run_rescue on IRU131

Chapter 2

Getting Started

2.1 Connecting the IRU131

The power

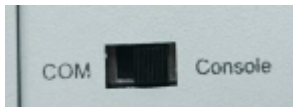
Please check you power as below:

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 IRU131 via a serial cable and change the switch to Console mode (as below).



You can connect the IRU131 to personal computer (PC) in two ways:

- Serial RS-232 console
- SSH over Ethernet



Note

Please download IRU131 support package from Axiomtek's website listed below.

1. Download BSP and User's manual
2. Download OPC UA application

<http://www.axiomtek.com/Default.aspx?MenuId=Products&FunctionId=ProductView&ItemId=24294&upcat=134>

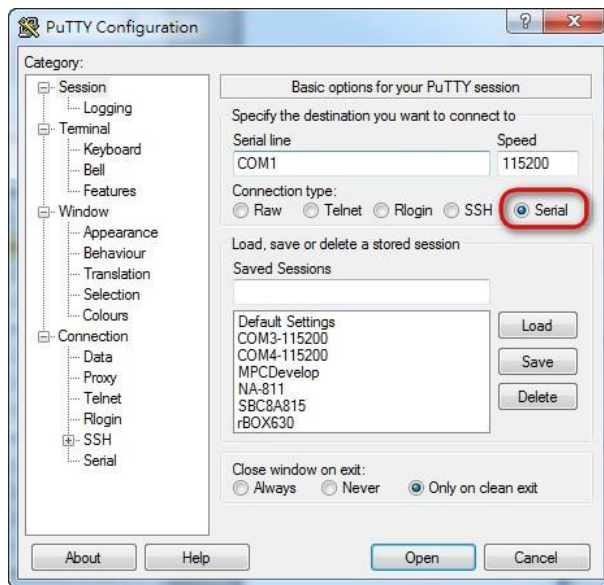
2.1.1 Serial Console

The serial console is a convenient interface for connecting IRU131 to PC. First of all, it is very important to make sure that your desktop connects to IRU131 by console cable. Please set the system as follows:

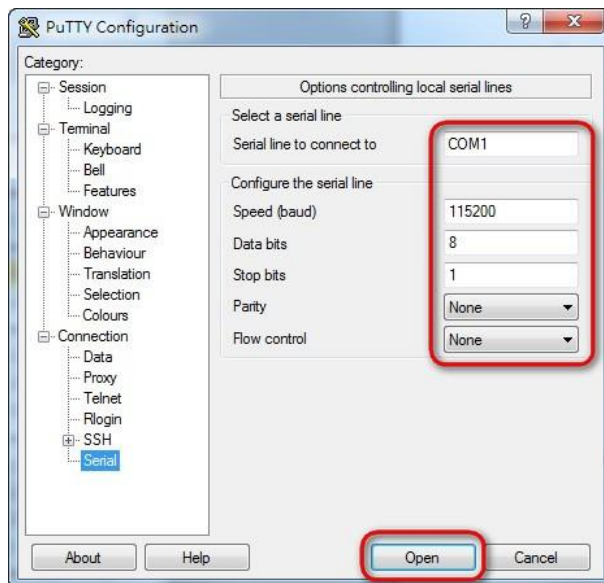
- Baudrate:** 115200 bps
- Parity:** None
- Data bits:** 8
- Stop bit:** 1
- Flow Control:** None

Here we use PuTTY to setup and link to the IRU131. Learn how to do it with these step by step instructions:

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 IRU131.



3. The Bootloader default booting system from eMMC.

```

COM3 - PuTTY
U-Boot 2015.04-imx_v2015.04_3.14.52_1.1.0_ga (Jun 16 2017 - 10:11:18)

CPU:   Freescale i.MX6UL rev1.1 at 396 MHz
CPU:   Temperature 45 C
Reset cause: POR
Board: RSB20X
I2C:   ready
DRAM:  512 MiB
PMIC:  PFUZE300 DEV_ID=0x30 REV_ID=0x11
MMC:   FSL_SDHC: 0, FSL_SDHC: 1
*** Warning - bad CRC, using default environment

In:    serial
Out:   serial
Err:   serial
switch to partitions #0, OK
mmc1(part 0) is current device
Net:   No MAC address found for primary NIC
FEC0
Error: FEC0 address not set.

Normal Boot
Hit any key to stop autoboot: 0
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
4931760 bytes read in 128 ms (36.7 MiB/s)
Booting from mmc ...
reading ax-rsb-imx6ul-irula.dtb
30861 bytes read in 18 ms (1.6 MiB/s)

```

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

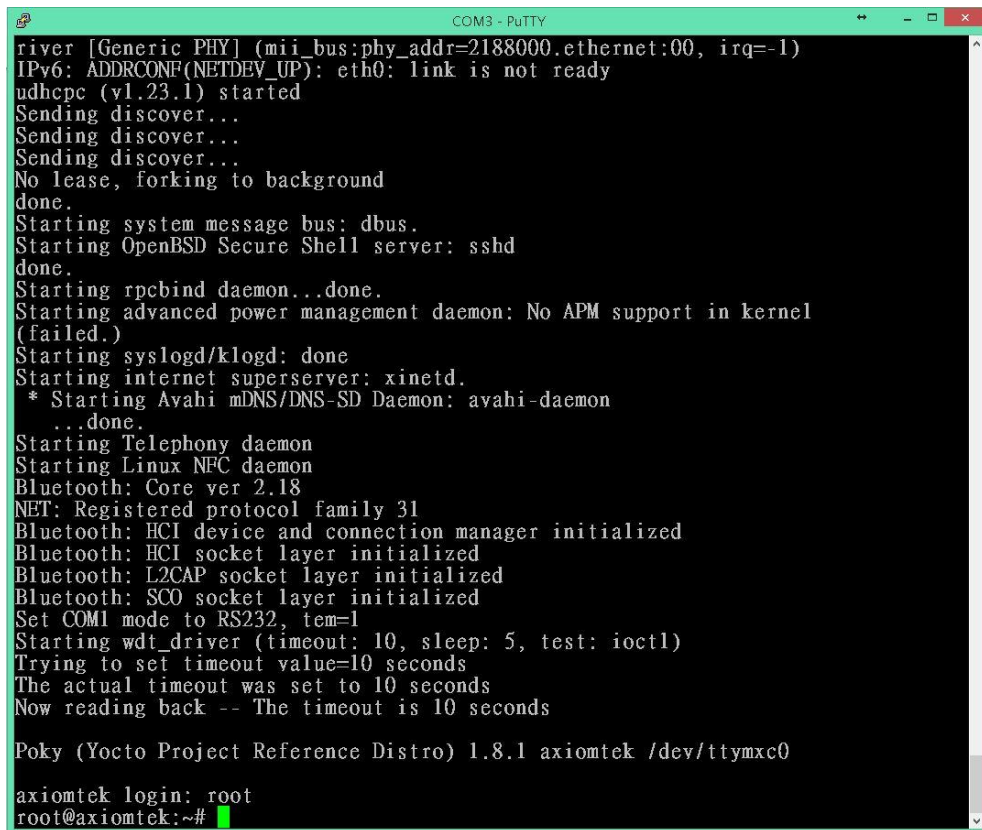
```

COM3 - PuTTY
Configuring network interfaces... fec 2188000.ethernet eth0: Freescale FEC PHY d
i_bus:phy_addr=2188000.ethernet:00, irq=-1)
IPv6: ADDRCONF(NETDEV_UP): eth0: link is not ready
udhcpd (v1.23.1) started
Sending discover...
Sending discover...
Sending discover...
No lease, forking to background
done.
Starting system message bus: dbus.
Starting OpenBSD Secure Shell server: sshd
done.
Starting rpcbind daemon...done.
Starting advanced power management daemon: No APM support in kernel
(failed.)
Starting syslogd/klogd: done
Starting internet superserver: xinetd.
* Starting Avahi mDNS/DNS-SD Daemon: avahi-daemon
...done.
Starting Telephony daemon
Starting Linux NFC daemon
Bluetooth: Core ver 2.18
NET: Registered protocol family 31
Bluetooth: HCI device and connection manager initialized
Bluetooth: HCI socket layer initialized
Bluetooth: L2CAP socket layer initialized
Bluetooth: SCO socket layer initialized
Set COM1 mode to RS232, tem=1
Starting wdt_driver (timeout: 10, sleep: 5, test: ioctl)
Trying to set timeout value=10 seconds
The actual timeout was set to 10 seconds
Now reading back -- The timeout is 10 seconds

Poky (Yocto Project Reference Distro) 1.8.1 axiomtek /dev/ttyxc0
axiomtek login: █

```

- To login, please enter 'root' (without password).



```

river [Generic PHY] (mii_bus:phy_addr=2188000.ethernet:00, irq=-1)
IPv6: ADDRCONF(NETDEV_UP): eth0: link is not ready
udhcpd (v1.23.1) started
Sending discover...
Sending discover...
Sending discover...
No lease, forking to background
done.
Starting system message bus: dbus.
Starting OpenBSD Secure Shell server: sshd
done.
Starting rpcbind daemon...done.
Starting advanced power management daemon: No APM support in kernel
(failed.)
Starting syslogd/klogd: done
Starting internet superserver: xinetd.
* Starting Avahi mDNS/DNS-SD Daemon: avahi-daemon
...done.
Starting Telephony daemon
Starting Linux NFC daemon
Bluetooth: Core ver 2.18
NET: Registered protocol family 31
Bluetooth: HCI device and connection manager initialized
Bluetooth: HCI socket layer initialized
Bluetooth: L2CAP socket layer initialized
Bluetooth: SCO socket layer initialized
Set COM1 mode to RS232, tem=1
Starting wdt_driver (timeout: 10, sleep: 5, test: ioctl)
Trying to set timeout value=10 seconds
The actual timeout was set to 10 seconds
Now reading back -- The timeout is 10 seconds

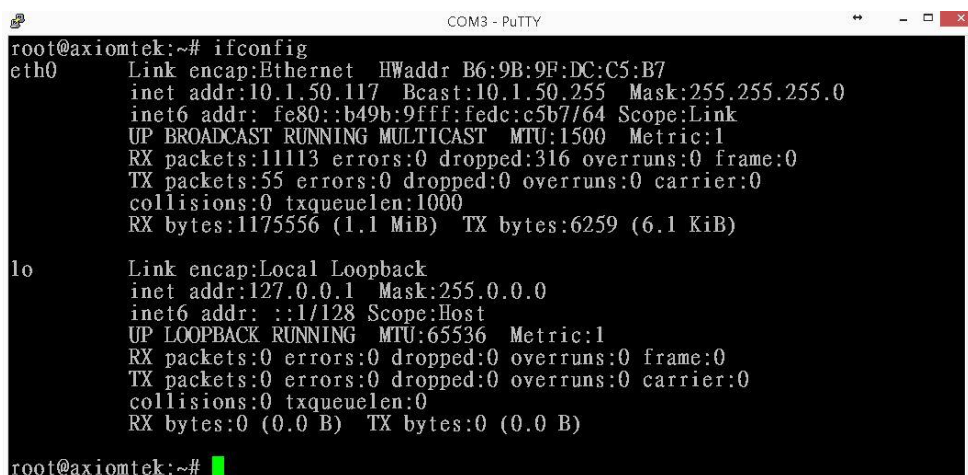
Poky (Yocto Project Reference Distro) 1.8.1 axiomtek /dev/ttymx0
axiomtek login: root
root@axiomtek:~#

```

2.1.2 SSH over Ethernet

Now, we are going to connect the IRU131 to PC over Ethernet. The following illustrations show how to do it under Windows® and Linux environment.

Before starting SSH, you have to check your LAN1 IP address.



```

root@axiomtek:~# ifconfig
eth0      Link encap:Ethernet  HWaddr B6:9B:9F:DC:C5:B7
          inet addr:10.1.50.117  Bcast:10.1.50.255  Mask:255.255.255.0
          inet6 addr: fe80::b49b:9fff:fedc:c5b7/64  Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:11113  errors:0  dropped:316  overruns:0  frame:0
          TX packets:55  errors:0  dropped:0  overruns:0  carrier:0
          collisions:0  txqueuelen:1000
          RX bytes:1175556 (1.1 MiB)  TX bytes:6259 (6.1 KiB)

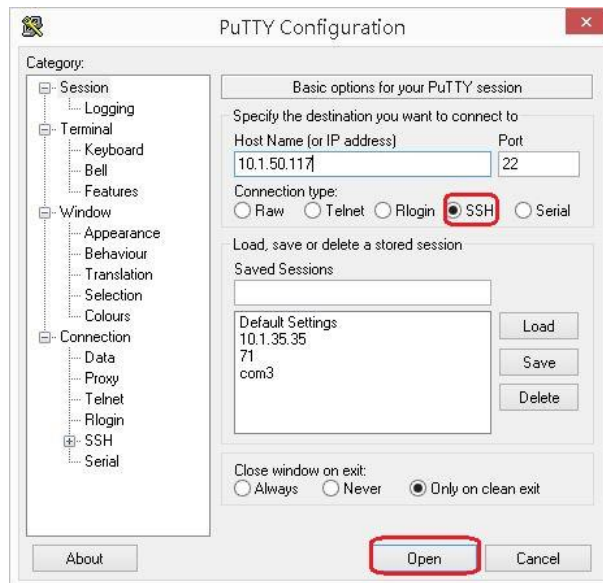
lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128  Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0  errors:0  dropped:0  overruns:0  frame:0
          TX packets:0  errors:0  dropped:0  overruns:0  carrier:0
          collisions:0  txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

root@axiomtek:~#

```

For Windows® users:

1. Here we also 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 login IRU131, please enter 'root' (with no password).



For Linux users:

1. Open terminal and keyin '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 successful.

```
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 IRU131 with the following step by step instructions. The sample program is named 'hello.c'.

1. Create a directory for IRU131 BSP (IRU1A_Linux_x.x.x.zip); related to IRU131 file

```
~$ 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 IRU131-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 include hardware driver and API library**Image** : This directory include kernel, rootfilesystem,dtb**Yocto patches** : This directory include IRU131 hardware patches for Yocto Project 1.8.1**Toolchain** : This directory include cross compiler toolchain build from Yocto Project 1.8.1**README.txt** : This BSP's documentation file

2.2.1 Install Yocto Toolchain

Before you develop and compile sample program, you should install Yocto toolchain into development PC. To install Yocto toolchain or refer to Chapter 5 Board Support Package to build the toolchain for IRU131.

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 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 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 to 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 Install finish.

```
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 the cross-development environment, and then you can find this script in the directory you chose for installation.

1. To set up 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 successful.

```
~$ 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
```

```
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, enter the commands:

```
~$ $CC hello.c -o hello
```

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


- After compiling, enter the following command and you can 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
```

- 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:~/project/example$ file hello
hello: ELF 32-bit LSB executable, ARM, EABI5 version 1 (SYSV), dynamically linked, interpreter /lib/ld-linux-armhf.so.3, for GNU/Linux 2.6.32, BuildID[sha1]=be20b9cf2ff90de9cabcd3db275faf058bd2961b, not stripped
```

2.3 How to Put and Run a Sample Program

This section shows how to put the 'hello' program into the IRU131 and execute the program via FTP, a USB flash drive, and TFTP.

2.3.1 Via FTP

The IRU131 has a built-in FTP server. Users can put 'hello' program to IRU131 via FTP by following the steps below.

- Enable FTPD daemon on IRU131
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
}
```

- Restart FTP server on IRU131

```
~# /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:~#
```

- To connect your host PC to IRU131.
~\$ 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.
```

- Upload "hello" program to IRU131 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)
```

- If the operation is successful on the IRU131, you can see 'hello' program on IRU131's /home/root directory.

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

- To change file permission for executable on IRU131.

```
~# 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
```

- Run the 'hello' program on IRU131.

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

2.3.2 Via USB Flash Drive

Another method of putting 'hello' program into IRU131 is via USB flash drive. Please follow the instructions below.

IRU131 supports storage format FAT32 /FAT/EXT2/EXT3/EXT4

1. From the host PC, copy 'hello' program to USB flash drive.
2. Attach USB flash drive to IRU131.

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

Originally the Host Development System Installation already has TFTP server installed. You can put the 'hello' program into IRU131 via TFTP. Please follow the instructions below.

1. Refer to section 5.1.1 step 4. Install and configure TFTP server for install and setup your TFTP:

2. To copy "hello" program to "tftpboot" folder in 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 IRU131
~# 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 IRU131
~# 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 IRU131.
~# ./hello

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

2.4 How to Recovery System

In this section, we provide 2 methods showing how to recovery IRU131 system to default.

2.4.1 Via run_rescue System Script (under Linux System)

A recovery script in /etc folder on IRU131 Embedded Linux system. If you want to recovery 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.

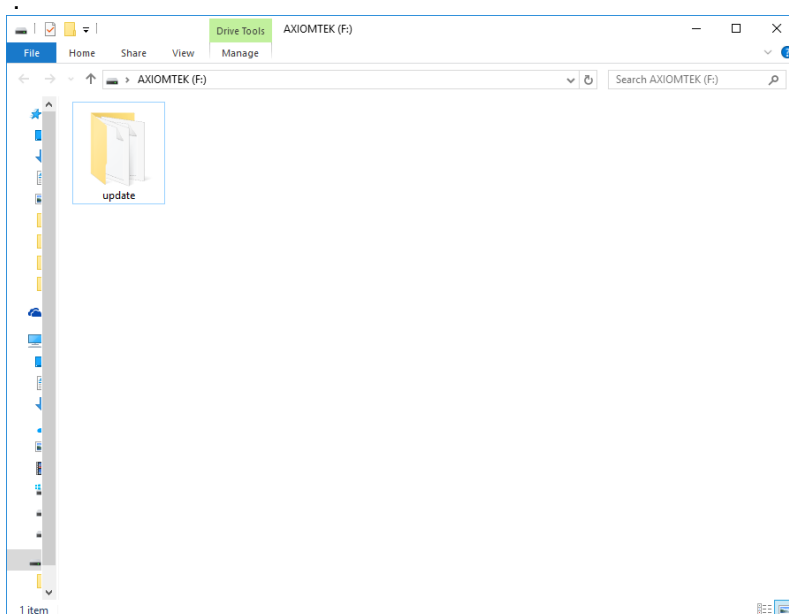
2.5 How to Update System

This section shows how to update the IRU131 using the recommended method below

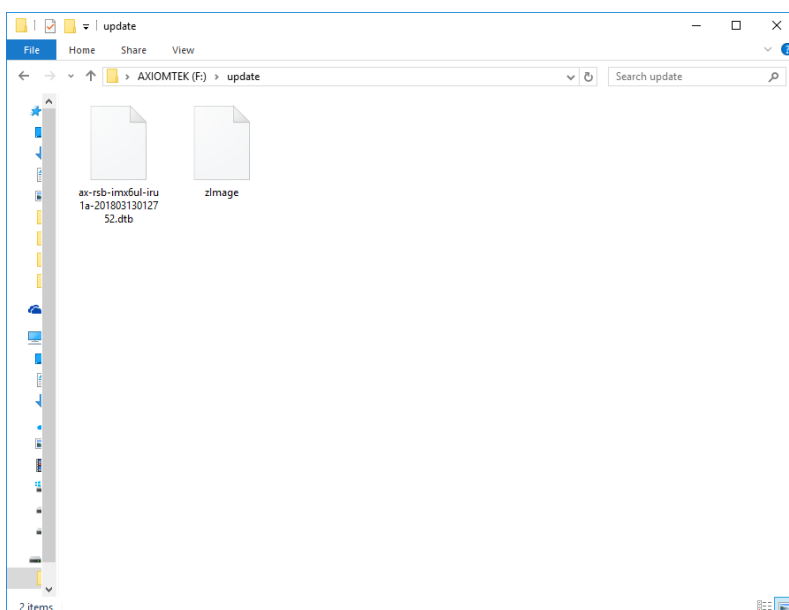
2.5.1 Via USB Flash Drive

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

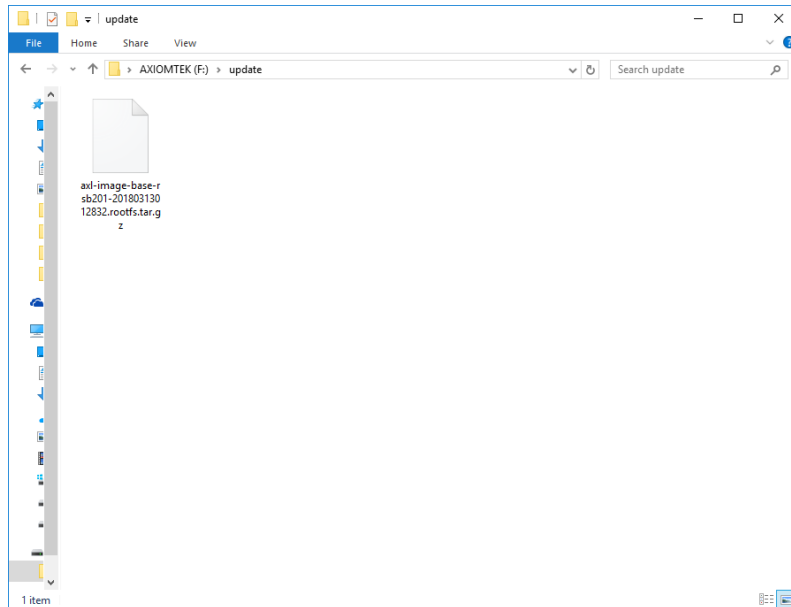
1. From the PC, copy files to USB flash drive.
2. Create a folder name "update"



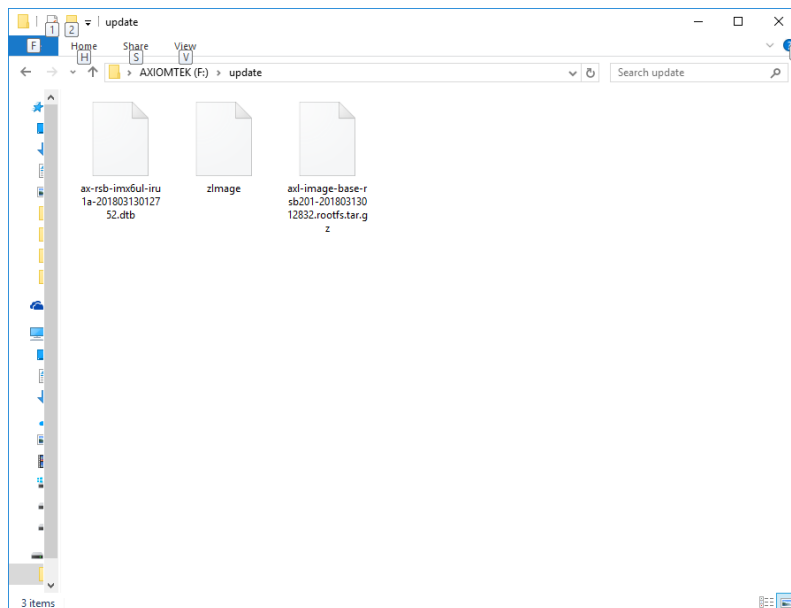
3. If you only want to update the kernel without altering the root filesystem, simply rename the new kernel file to 'zImage' and the dtb file to 'ax-rsb-imx6ul-iru1a.dtb' and then put the files in the update folder.



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



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



- Attach USB flash drive to IRU131.

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

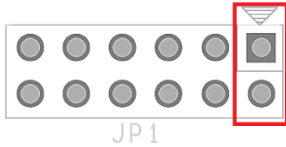
```
sd 0:0:0:0: [sda] Assuming drive cache: write through
sda: sda1
sd 0:0:0:0: [sda] No Caching mode page found
sd 0:0:0:0: [sda] Assuming drive cache: write through
sd 0:0:0:0: [sda] Attached SCSI removable disk
Freeing unused kernel memory: 368K (8088d000 - 808e9000)
INIT: version 2.88 booting
Starting udev
Starting udevd[159]: starting version 182
EXT4-fs (mmcblk1p3): re-mounted. Opts: data=ordered
bootlogd: cannot allocate pseudo tty: No such file or directory
random: dd urandom read with 91 bits of entropy available
random: nonblocking pool is initialized
INIT: Entering runlevel: 5
Starting syslogd/klogd: done
FAT-fs (sda1): Volume was not properly unmounted. Some data may be corrupt. Ple.
===== Starting Update Kernel Procedure =====
===== Starting Update RootFilesystem Procedure =====
EXT4-fs (mmcblk1p2): mounted filesystem with ordered data mode. Opts: (null)
EXT4-fs (mmcblk1p2): mounted filesystem with ordered data mode. Opts: (null)
Copy Other tools.....
Extracting /media/sda1/update/tools/IFB122-progs-004.tgz
===== Finished =====
After 3 seconds will reboot system...
```

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

2.6 How to use MFG tool to download image

We show you how to use MFG tool to download image to the IRU131 system.

1. Before using the MFG tool, you have to change the IRU131 JP1 boot mode (default emmc boot) to OTG serial downloader mode

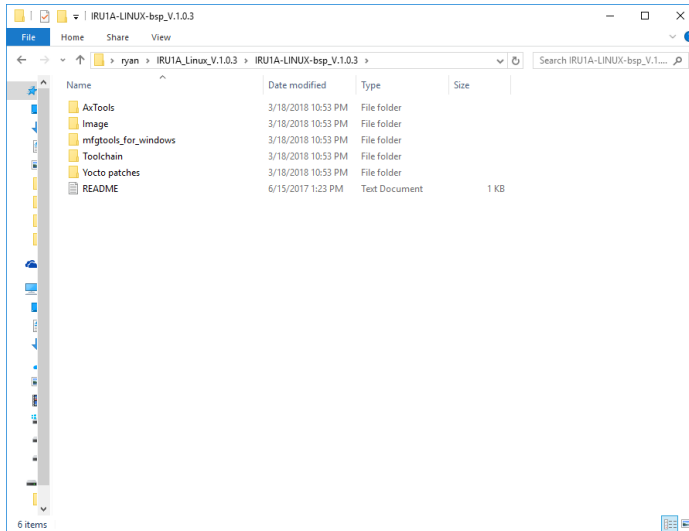


Please change JP3 to OTG Client mode as below .

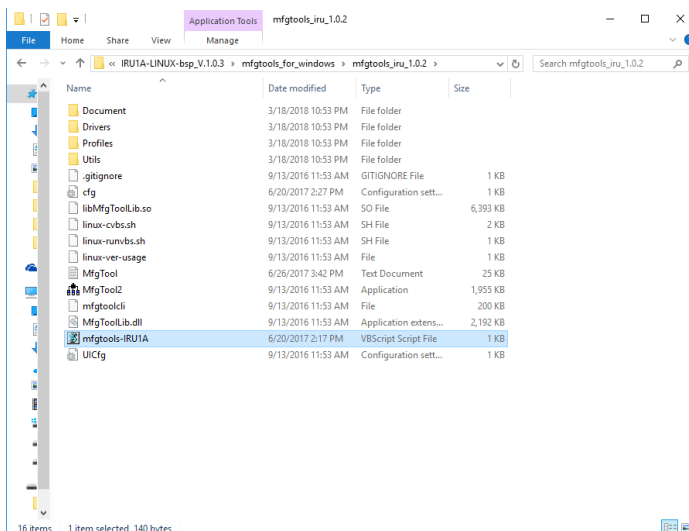


After setting jumper, please connect the IRU131 to PC via USB cable.

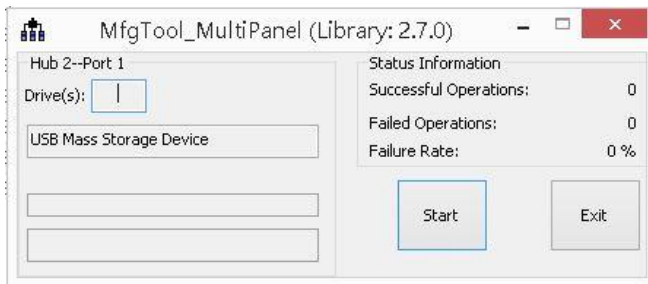
2. Extract Axiomtek's Yocto BSP and you will see mfgtools_iru_x.x.x in the mfgtools_for_windows directory



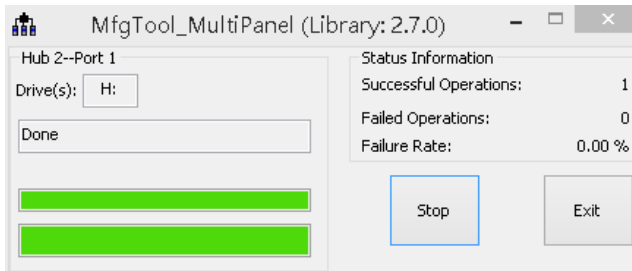
3. Enter mfgtools_for_windows/mfgtools_iru_x.x.x directory



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



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



- 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, Node-red, Node.js 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

- 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-2.1.400-1 45% |*****| 35818k 0:00:19 ETA
```

- 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-2.1.400-1 100% |*****| 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.azure.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.azure.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
```

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 version on IRU131.

Check 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/ttyxc0
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 IRU131:

```
~# 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 IRU131:

```
~# 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 is 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 is more than <timeout> parameters, watchdog timer will be trigger

Note: The IRU131 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 TCP-based network.

The IRU131 comes with a built-in FTP server. Section 2.1 shows the steps to put 'hello' program in the IRU131 via FTP.

3.2.2 TFTP – Trivial File Transfer Protocol

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

In Chapter 5, there are descriptions of TFTP server installation and kernel boot up process via TFTP. Section 2.3.3 shows you how to transfer file between server and client.

3.2.3 NFS – Network File System

NFS enables you to export a directory on an NFS server and mount that directory on remote client machine as if it were a local file system. Using NFS on 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 3G or 4G module (Optional)

1. 3G / 4G module connection to the Internet with PPP

This section describes how to use 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.

```
~# /etc/ppp/ppp-quectel-on
```

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

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

```

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

```

When you execute script, you may find the information as 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 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 `wvdial`, 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 your 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 **wvdial**, 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

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: 08T
--> pppd: 08T
--> pppd: 08T
--> pppd: 08T
--> local IP address 10.203.98.12
--> pppd: 08T
--> remote IP address 10.203.98.12
--> pppd: 08T
--> primary DNS address 172.24.9.33
--> pppd: 08T
--> secondary DNS address 172.24.9.22
--> pppd: 08T

```

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 Wi-Fi module (Optional)

If your Wi-Fi module is WPEQ-160ACN, you can 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 release 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 description of each API function and step-by-step code samples showing how it works.

4.1 librsb20x API Functions

The IRU131 BSP includes 'librsb10x.so' shared library for users to access I/O and read back system information. This shared library is kept in BSP, you can find it in IRU1A-rsb-lib-x.x.x.tar.gz of AxTools. Extract the compressed file, then besides the shared library you can also see a *demo* folder containing API header file and example programs.

Summary table of available API functions

No.	Function	Description
1	Control_WDT	Set WDT functio
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
6	Set_DO0	Set digital output channels 0 state.
7	Set_DO1	Set digital output channels 1 state.
8	Set_DO0_not	Set digital output channels state 0. (reverse)
9	Set_DO1_not	Set digital output channels state 1. (reverse)
10	Get_DI0	Read state on digital input channels 0.
11	Get_DI1	Read state on digital input channels 1.
12	Get_DI0_not	Read state on digital input channels 0.(reverse)
13	Get_DI1_not	Read state on digital input channels 1.(reverse)

Com Port 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/ttymx1", O_RDWR | O_NOCTTY);
    if(fd < 0) {
        printf("open error /dev/ttymx1 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_iflag &= ~(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/ttymx1", O_RDWR | O_NOCTTY);
    if(fd < 0) {
        printf("open error /dev/ttymx1 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_iflag &= ~(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==0)
        printf("Disable USB Power\n");
    return 0;
}
```

Function: Control_EXP_RST ()

Function	int Control_EXP_RST(void)
Description	Set digital output channels to high or low state.
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.

WDT 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;
}

```

Function: Get_DI0()

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

Function: Set_DO0()

Function	int Set_DO0(int data);
Description	Set digital output channels state 0.
Arguments	data: Data to be written to digital output channels.
Return	0: No error. 1: Function fails.
Others	None.

Function: Set_DO1()

Function	int Set_DO1(int data);
Description	Set digital output channels state 1.
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 0. (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 1. (reverse)
Arguments	data: This function will store digital output data in this argument.
Return	0: No error. 1: Function fails.
Others	None.

Function: Set_DO0_not ()

Function	int Set_DO0_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.

Function: Set_DO1_not ()

Function	int Set_DO1_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 "librsb20x.h"

int main(int argc, char* argv[])
{
    printf("Axiomtek Control IRU131 DIO test\n");

    int ch0,ch1;
    Set_DO0(0);
    Set_DO1(0);

    Set_DO0(1);
    Set_DO1(1);
    printf("\nDO0 Oput   = 1\n");
```

```
printf("DO1 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);
Set_DO1(0);
printf("\nDO0 Oput    = 0\n");
printf("DO1 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);
Set_DO1_not(1);
printf("\nDO0 Oput not= 1\n");
printf("DO1 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);
Set_DO1_not(0);
printf("\nDO0 Oput not      = 0\n");
printf("DO1 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;}
```


4.2 Compile Demo Program

4.2.1 Install IRU131 I/O Library

Before you develop and compile sample program, you should install Yocto toolchain into 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 demo program for the IRU131, do the following:
Change to *your project* directory.

```
~$ cd project/IRU1A-Linux_V.1.0.1/IRU1A-LINUX-V.1.0.1/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 rsb_lib/demo/
ryan@Ubuntu:~/project/IRU1A_Linux_V.1.0.1/IRU1A-LINUX-bsp_V.1.0.1/AxTools/rsb_lib/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_l
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 wdtttest wdtttest.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_l
b/demo$ ls
com_mode          diotest.c        exp_rst          Makefile         wdtttest.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    wdtttest

```

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 Ubuntu 14.04 LTS iso image.
2. Install Ubuntu 14.04.
3. Install host packages needed 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 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 whatever directory you want to download from. Then reboot the hardware.

To install tftpd / tftp/ xinetd

```
~$ 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 tftp sever.

```
~$ 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 your build.

```
~$ 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 `~/bin` folder 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-1.9_m1 -> yocto-1.9_m1
* [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.X.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/Yocto\ 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 build image finish, 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 user's Yocto Toolchain

We have provided Yocto Toolchain in IRU131 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
```

```
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 these steps to generate the toolchain into the Build Directory, 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
```

```
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 IRU131

5.2.1 Booting the System from eMMC (IRU131 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 (GCC)
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 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 "include" file.

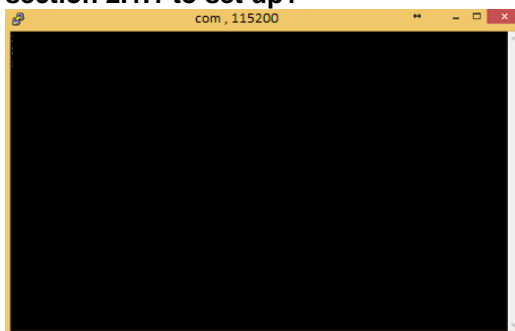
A1: Refer to section 2.3 and 2.2.2 Setting up the Cross-Development Environment for detailed information.

For example: `$CC hello.c -o hello`

```

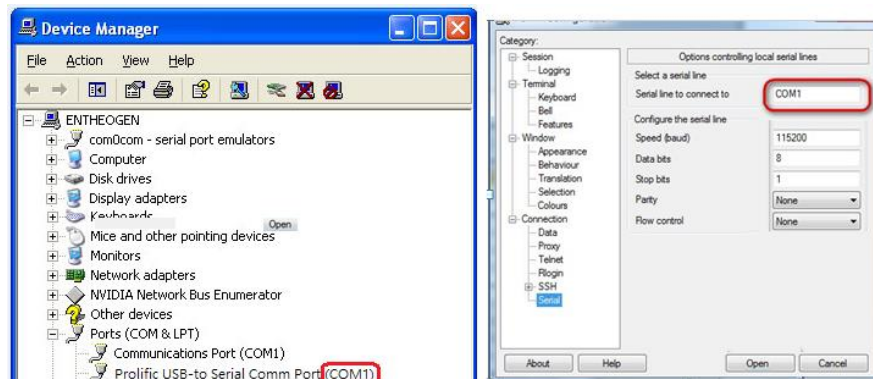
louis@axio-pc:~/work/IFB22/test_program$ ls /opt/fsl-imx-x11/3.14.52-1.1.0
environment-setup-cortexa7hf-vfp-neon-poky-linux-gnueabi
site-config-cortexa7hf-vfp-neon-poky-linux-gnueabi
sysroots
version-cortexa7hf-vfp-neon-poky-linux-gnueabi
louis@axio-pc:~/work/IFB22/test_program$ source /opt/fsl-imx-x11/3.14.52-1.1.0/e
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 as below after I follow the steps described in section 2.1.1 to set up?



A2. Please follow steps as below

1. To check your power.
2. To check serial item "COM port" name and Device Manager "COM port" name are both the same as below.



3. To check if your RS232 port jumper.

Q3. Why can't 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.