IFB122

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.

Apr. 2023, Version A6

Printed in Taiwan

Table of Contents

Chapter 1
Introduction

The extreme compact IFB122 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 IFB122. The embedded Linux is derived from Linux Yocto Board Support Package, which is based on Linux Kernel 5.10.72 and our hardware patches to suit IFB122.

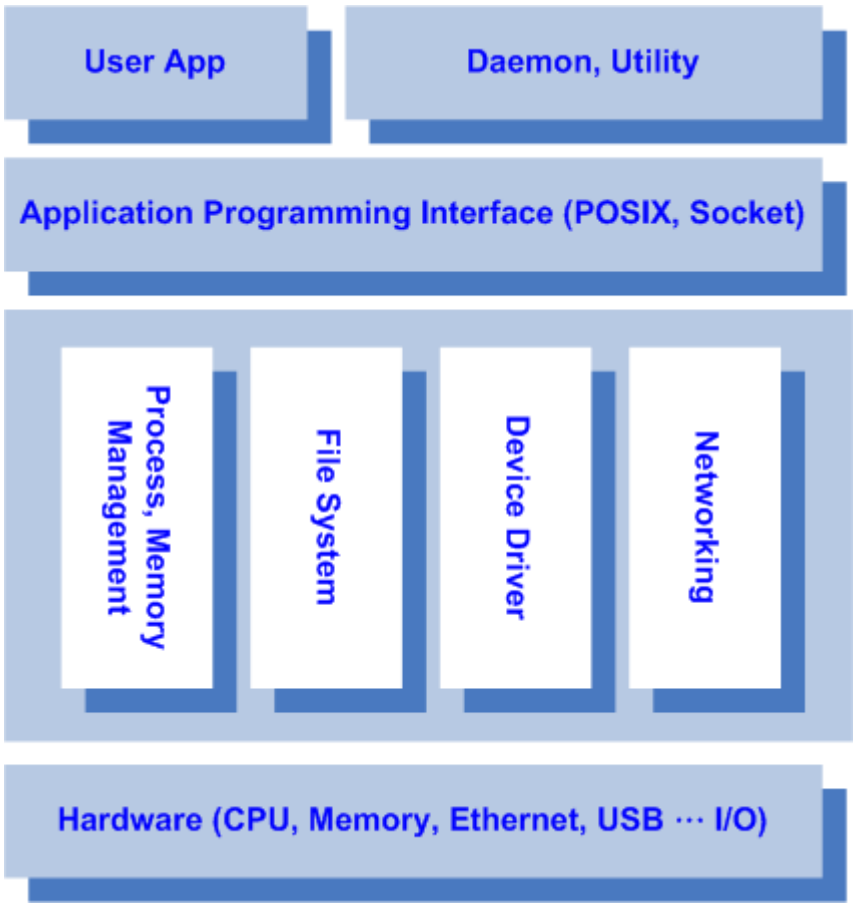
Software structure

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, the IFB122 includes 'libEApi.so' shared library.

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 4G.

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



1.1 Specifications

- **OS: Linux**
  - Kernel: 5.10.72 (with NXP and Axiomtek hardware modified patch)
- **Shell**
  - Bash
- **Support storage format**
  - FAT32 /FAT/EXT2/EXT3/EXT4
- **Daemons**
  - Telnetd: Telnet server daemon
  - FTPD: FTP server daemon
- **Utilities**
  - Telnet: Telnet client program
  - FTP: FTP client program
  - TFTP: Trivial File Transfer Protocol client
- **Packages**
  - **busybox**: Small collection of standard Linux command-line utilities

- **udev**: A device manager for Linux kernel
- **dosfstools** : Utilities for making and checking MS-DOS FAT file system
- **e2fsprogs**: A set of utilities for maintaining the ext2, ext3 and ext4 file systems
- **ethtool**: A Linux command for displaying or modifying the Network Interface Controller (NIC) parameters
- **i2c-tools** : A heterogeneous set of I2C tools for Linux
- **procps** : Utilities to report on the state of the system, including the states of running processes, amount of memory

- **Development Environment**
  - Host OS/ development OS: The recommended minimum Ubuntu version is 18.04 or later
  - 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, which is enough to compile all backends together.
  - Toolchain/ cross compiler: toolchain-x.x.x-hardknott (Yocto project 3.3 Hardknott)

- **HW's Lib (Hardware's Library)**
  - **Digital I/O**
    - Read digital input
    - Write digital output
  - **COM**
    - RS-232/422/485 mode setting(Default RS232)
  - **Watch Dog Timer**
    - Enable Watch Dog Timer
    - Set Timer
  - **Relay**

    Set relay high or low.

Note

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

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 (IFB122):**	Ex: **~#** /etc/run_rescue Meaning: To command /etc/run_rescue on IFB122

Chapter 2
Getting Started

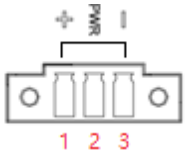
2.1 Connecting the IFB122

The power

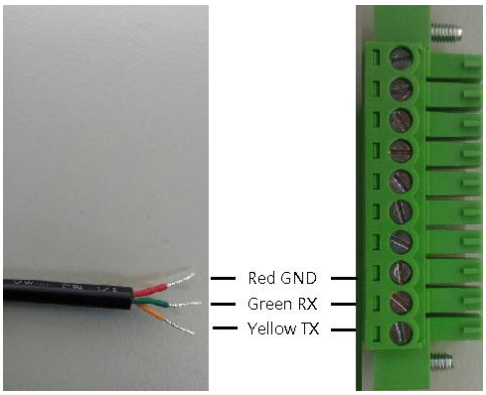
Please check you power as below:

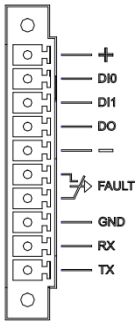
1. DC input range 9~48V
2. DC Terminal Block

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



Console Port

- For user setting with debug. You can find TB10 pins for console port as below table.
- Connected to DIO terminal Block
- 

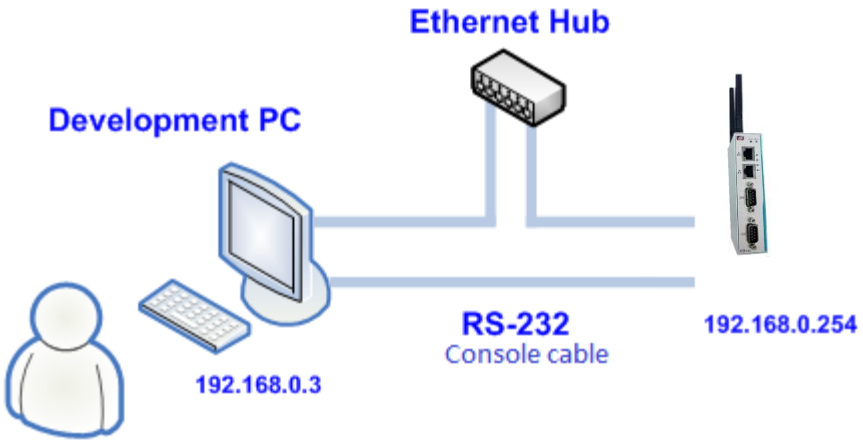


http://www.axiomtek.com/Default.aspx?MenuId=Products&FunctionId=gSearch&keyword=IFB122FB122 DIO Terminal Block

TB18 Pin No.	Signal name	Meaning
1	COM+	Plus Common for DIO
2	DI0	Digital Input
3	DI1	
4	DO	Digital Output
5	COM-	Minus Common for DIO
6	Relay+	Relay Out
7	Relay-	
8	GND	For Console Port
9	Console RX	
10	Console TX	

You can connect the IFB122 to personal computer (PC) in two ways:

- Serial RS-232 console
- SSH over Ethernet



Note

Please download below data from Axiomtek's website as below list if you have the demand.

- ***BSP support package.***
- **http://www.axiomtek.com/Default.aspx?MenuId=Products&FunctionId=gSearch&keyword=IFB122**
-

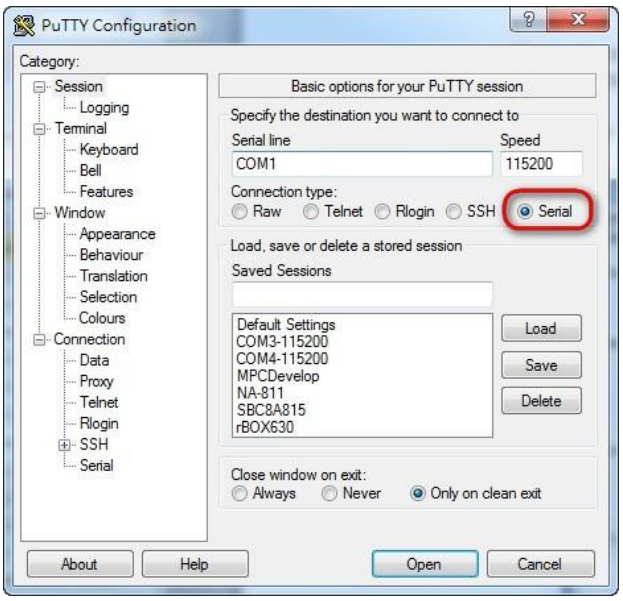
2.1.1 Serial Console

The serial console is a convenient interface for connecting IFB122 to PC. First of all, it is very important to make sure that your desktop connects to IFB122 by console cable. Please set the system as follows:

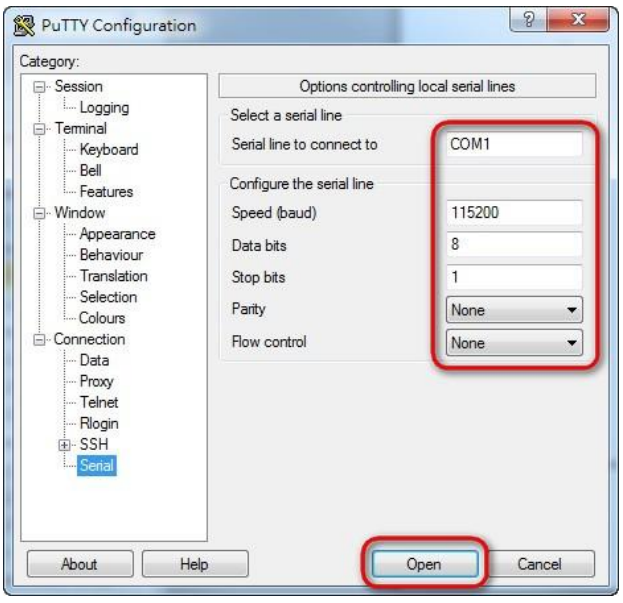
Baudrate: 115200 bps
Parity: None
Data bits: 8
Stop bit: 1
Flow Control: None

Here we use PuTTY to setup and link to the IFB122. Learn how to do it with these step by step instructions:

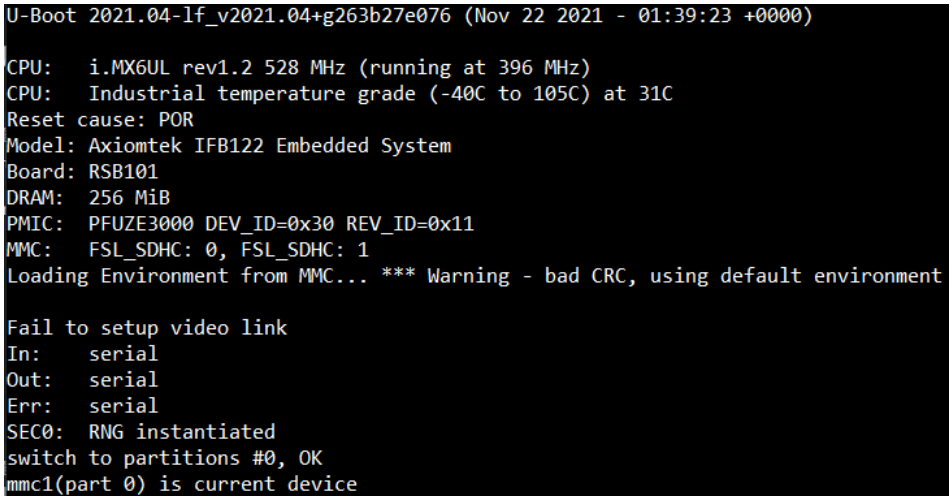
1. Open PuTTY and choose 'Serial' as the connection type.



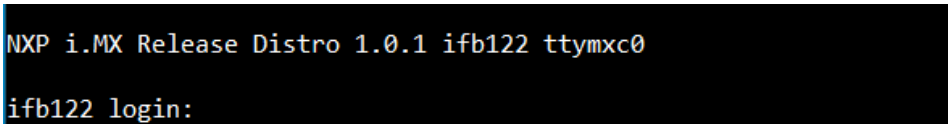
2. Configure the serial port correctly (see image below). Click Open and power on the IFB122.



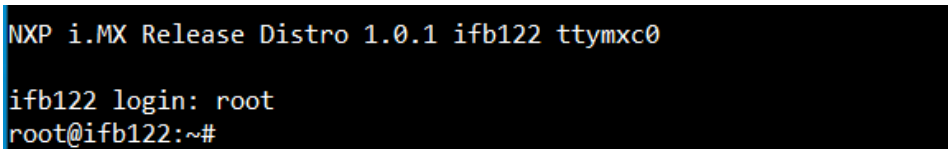
3. The Bootloader default booting system from eMMC.



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



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

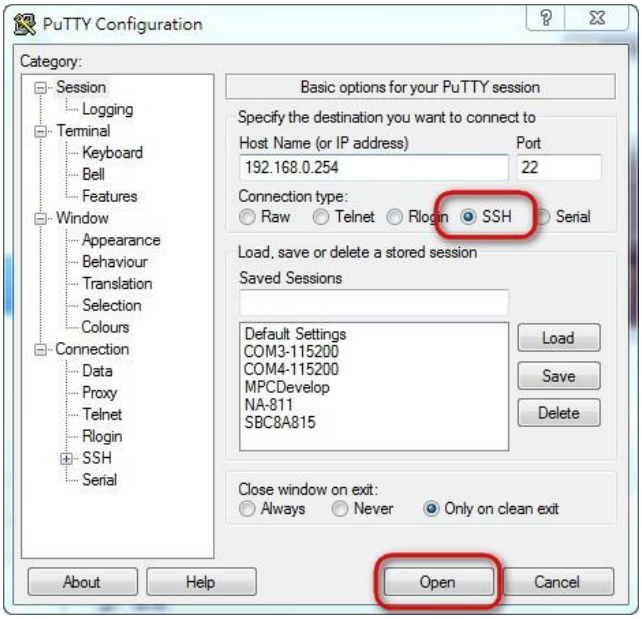


2.1.2 SSH over Ethernet

Now, we are going to connect the IFB122 to PC over Ethernet. The following illustrations show how to do it under Windows® and Linux environment.

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 and click Open.



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

3. To login IFB122, please enter 'root' (with no password).

For Linux users:

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

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

2. After the connection is established successfully.

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

2.2 How to Develop a Sample Program

In this section, learn how to develop a sample program for IFB122 with the following step by step instructions. The sample program is named 'hello.c'.

1. To Create a directory for IFB122 BSP (IFB122_L510_vx.x.x.zip); related to IFB122 file

~$ mkdir project
~$ cd project

```
axiomtek@axiomtek-PC:~$ mkdir project
axiomtek@axiomtek-PC:~$ cd project
axiomtek@axiomtek-PC:~/project$ ls
IFB122_L510_v1.0.1.zip
```

2. After extracted the file, you will find a directory IFB122_L510_vx.x.x

```
axiomtek@axiomtek-PC:~/project$ cd IFB122_L510_v1.0.1/
axiomtek@axiomtek-PC:~/project/IFB122_L510_v1.0.1$ ls
Image  Toolchain  Yocto_patches
```

Image : This directory include kernel, rootfilesystem
Yocto_patches : This directory include IFB122 hardware patches for Yocto Project 3.3
Toolchain : This directory include cross compiler toolchain build from Yocto Project 3.3

2.2.1 Install Yocto Toolchain

Before you develop and compile sample program, you should install Yocto toolchain into development PC. You can follow below step to install Yocto toolchain or refer to Chapter 5 Board Support Package to build the toolchain for IFB122.

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

```
axiomtek@axiomtek-PC:~$ uname -a
Linux axiomtek-PC 5.4.0-42-generic #46~18.04.1-Ubuntu SMP Fri Jul 10 07
:21:24 UTC 2020 x86_64 x86_64 x86_64 GNU/Linux
```

2. Copy the toolchain script to home directory.

```
axiomtek@axiomtek-PC:~/project/IFB122_L510_v1.0.1$ cd Toolchain/
axiomtek@axiomtek-PC:~/project/IFB122_L510_v1.0.1/Toolchain$ cp fsl-imx-xwayla
nd-glibc-x86_64-imx-image-core-cortexa7t2hf-neon-ifb122-toolchain-1.0.1.sh ~/
```

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

$./fsl-imx-xwayland-glibc-x86_64-imx-image-core-cortexa7t2hf-neon-ifb122-toolchain-1.0.1.sh

```
axiomtek@axiomtek-PC:~$ ./fsl-imx-xwayland-glibc-x86_64-imx-image-core-cortexa
7t2hf-neon-ifb122-toolchain-1.0.1.sh
NXP i.MX Release Distro SDK installer version 1.0.1
=================================================
Enter target directory for SDK (default: /opt/fsl-imx-xwayland/1.0.1):
```

4 Check the directory.

```
axiomtek@axiomtek-PC:~$ ./fsl-imx-xwayland-glibc-x86_64-imx-image-core-cortexa
7t2hf-neon-ifb122-toolchain-1.0.1.sh
NXP i.MX Release Distro SDK installer version 1.0.1
=================================================
Enter target directory for SDK (default: /opt/fsl-imx-xwayland/1.0.1):
You are about to install the SDK to "/opt/fsl-imx-xwayland/1.0.1". Proceed [Y/
n]? Y
```

5 Wait to installation.

```
axiomtek@axiomtek-PC:~$ ./fsl-imx-xwayland-glibc-x86_64-imx-image-core-cortexa
7t2hf-neon-ifb122-toolchain-1.0.1.sh
NXP i.MX Release Distro SDK installer version 1.0.1
=================================================
Enter target directory for SDK (default: /opt/fsl-imx-xwayland/1.0.1):
You are about to install the SDK to "/opt/fsl-imx-xwayland/1.0.1". Proceed [Y/
n]? Y
[sudo] password for axiomtek:
Extracting SDK.........................
```

6 Install finish.

```
axiomtek@axiomtek-PC:~$ ./fsl-imx-xwayland-glibc-x86_64-imx-image-core-cortexa
7t2hf-neon-ifb122-toolchain-1.0.1.sh
NXP i.MX Release Distro SDK installer version 1.0.1
=================================================
Enter target directory for SDK (default: /opt/fsl-imx-xwayland/1.0.1):
You are about to install the SDK to "/opt/fsl-imx-xwayland/1.0.1". Proceed [Y/
n]? Y
[sudo] password for axiomtek:
Extracting SDK.....................................................................
.....................................................................
.....................................................................
.............done
Setting it up...done
SDK has been successfully set up and is ready to be used.
Each time you wish to use the SDK in a new shell session, you need to source t
he environment setup script e.g.
 $ . /opt/fsl-imx-xwayland/1.0.1/environment-setup-cortexa7t2hf-neon-poky-linu
x-gnueabi
```

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.

$. /opt/fsl-imx-xwayland/1.0.1/environment-setup-cortexa7t2hf-neon-poky-linux-gnueabi

```
axiomtek@axiomtek-PC:~$ . /opt/fsl-imx-xwayland/1.0.1/environment-setup-cortex
a7t2hf-neon-poky-linux-gnueabi
```

2. To check Cross-Development Environment whether successful or not
   It is successful, if you can find the information as below.

$ echo $CC

```
axiomtek@axiomtek-PC:~$ echo $CC
arm-poky-linux-gnueabi-gcc -mthumb -mfpu=neon -mfloat-abi=hard -mcpu=cortex-a7
 -fstack-protector-strong -O2 -D_FORTIFY_SOURCE=2 -Wformat -Wformat-security -
Werror=format-security --sysroot=/opt/fsl-imx-xwayland/1.0.1/sysroots/cortexa7
t2hf-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:

~$ $CC hello.c -o hello

```
louis@ubuntu:~/project/example$ $CC hello.c -o hello
```

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

~$ ls -l

```
louis@ubuntu:~/project/example$ ls -l
total 16
-rwxrwxr-x 1 louis louis 9669  7月 20 16:59 hello
-rw-rw-r-- 1 louis louis   70  7月 20 16:54 hello.c
louis@ubuntu:~/project/example$
```

2.3 How to Put and Run a Sample Program

In this section, we provide 3 methods showing how to put the 'hello' program into IFB122 and execute it.

2.3.1 Via FTP

The **IFB122** system has a built-in FTP server. Users can put 'hello' program to IFB122 via FTP by following the steps below.

1. Enable FTPD daemon on IFB122

   # vi /etc/xinetd.d/ftpd

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

2. Restart FTP server on IFB122

   # systemctl restart xinetd

```
root@ifb122:~# systemctl restart xinetd
```

3. To connect your host PC to IFB122.
   ~$ ftp [ip] (username 'root' without password)

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

4. Upload "hello" program to IFB122 from your host PC

   ftp> put hello

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

5. If the operation is successful on IFB122, you can see 'hello' program on IFB122's */home/root* directory.

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

6. To change file permission for executable on IFB122.
   ~# chmod a+x hello

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

7. Run the 'hello' program on IFB122.
   ~# ./hello

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

2.3.2 Via USB Flash Drive

Another method of putting 'hello' program into IFB122 is via USB flash drive. Please follow the instructions below.

IFB122 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 IFB122.

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

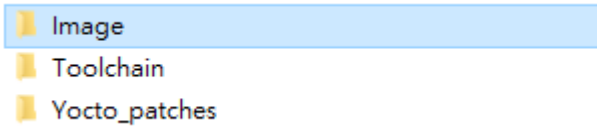
```
root@axiomtek:~# ./hello
hello world
root@axiomtek:~#
```

2.5 How to use MFG tool to download image

We show you how to use MFG tool to download image to the IFB122 system.

1. Before using the MFG tool, you have to change the IFB122 JP1 boot mode (default emmc boot) to OTG serial downloader mode. Then change the JP3 USB mode (default OTG host mode) to OTC client mode. Connect the IFB122 and PC with a USB cable.

2. Extract Axiomtek's Yocto BSP and you will see Image in the IFB122_L510_vx.x.x directory



For Windows® users:

1. Open Windows PowerShell or CMD and switch to tool path

```
PS C:\WINDOWS\system32> d:
PS D:\> cd .\IFB122_L510_v1.0.1\Image\ax_uuu_v1.1.0\
```

2. Run flash command: > .\uuu.exe .\ifb122_emmc.uuu

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

```
PS D:\IFB122_L510_v1.0.1\Image\ax_uuu_v1.1.0> .\uuu.exe .\ifb122_emmc.uuu
uuu (Universal Update Utility) for nxp imx chips -- libuuu_1.4.193-0-ge56424c

Success 1    Failure 0


1:3     27/27 [Done                                ] FBK: Done
```

For Linux users:

1. Open Terminal and switch to tool path

```
axiomtek@axiomtek-PC:~$
axiomtek@axiomtek-PC:~$ cd project/IFB122_L510_v1.0.1/Image/ax_uuu_v1.1.0/
```

2. Run flash command: $ sudo ./uuu ifb122_emmc.uuu

```
axiomtek@axiomtek-PC:~$ cd project/IFB122_L510_v1.0.1/Image/ax_uuu_v1.1.0/
axiomtek@axiomtek-PC:~/project/IFB122_L510_v1.0.1/Image/ax_uuu_v1.1.0$ sudo ./uuu ifb1
22_emmc.uuu
[sudo] password for axiomtek:
uuu (Universal Update Utility) for nxp imx chips -- libuuu_1.4.193-0-ge56424c

Success 1    Failure 0


1:8     27/27 [Done                                ] FBK: Done
```

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

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 IFB122.

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

```
root@ifb122:~# uname -r
5.10.72-lts-5.10.y+g00471544c239
```

Check root filesystem with the login screen:

```
NXP i.MX Release Distro 1.0.1 ifb122 ttymxc0

ifb122 login: root
```

3.1.2 System Time

System time is the time value loaded from RTC each time the system boots up. Read system time with the following command on IFB122:

~# date

```
root@axiomtek:~# date
Fri Jan 29 17:30:07 UTC 2016
```

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 IFB122:
~# hwclock -r --rtc=/dev/rtc1

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

3.1.4 External RTC Time

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

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

```
root@axiomtek:~# hwclock -r
Thu Jul 14 13:32:20 2016  0.000000 seconds
```

1.1.5 Watchdog timer

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

Note: IFB122 has been enabled for default setting, and the default parameters is ***10 5 0***
Commands example: ~# wdt 10 5 0 &

```
root@ifb122:/unit_tests/Watchdog# ./wdt_driver_test.out 10 5 0

---- Running < ./wdt_driver_test.out > test ----

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
```

Adjusting System Time

1. Manually set up the system time.

Format: YYYYMMDDHHmm.SS
~# date -s date -s 201509161714.05

```
root@axiomtek:~# date -s 201509161714.05
Wed Sep 16 17:14:05 UTC 2015
```

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
Wed Sep 16 17:15:35 2015  0.000000 seconds
root@axiomtek:~#
```

3. Write sync time to external RTC

~# hwclock -w

```
root@axiomtek:~# hwclock -w
root@axiomtek:~# hwclock -r
Wed Sep 16 17:16:31 2015  0.000000 seconds
root@axiomtek:~#
```

3.1.7 LEDs Control

Four custom LEDs are supported by IFB122: LED1, LED2, LED3 and LED4.

Use sysfs filesystem to control LED on/off state.

1. Turn on LED1

```
~# echo 255 > /sys/class/leds/led1/brightness
```

```
root@ifb122:~# echo 255 > /sys/class/leds/led1/brightness
```

2. Turn on LED2

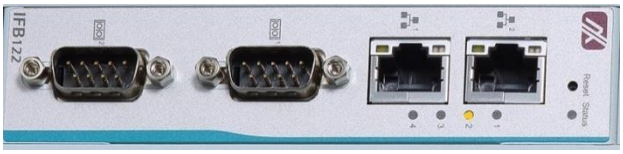
```
~# echo 255 > /sys/class/leds/led2/brightness
```

```
root@ifb122:~# echo 255 > /sys/class/leds/led2/brightness
```

3. Turn off LED1

```
~# echo 0 > /sys/class/leds/led1/brightness
```

```
root@ifb122:~# echo 0 > /sys/class/leds/led1/brightness
```



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 IFB122 comes with a built-in FTP server. Section 2.3 shows the steps to put 'hello' program to IFB122 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.

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 EApi API Functions

The IFB122 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 RSB10X-rsb_lib-x.x.x.tar.bz2 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	EApiGPIOGetLevel()	Read high or low state on digital input/ output channels.
2	EApiGPIOSetLevel()	Write high or low state on digital input/ output channels.
3	EApiWDogStart()	Enable watchdog timer
4	EApiWDogTrigger()	Reset WDT counter.
5	EApiWDogStop()	Disable watchdog timer
6	EApiComGetType()	Get COM port communication mode type.
7	EApiComSetType()	Set COM port communication mode type.
8	EApiComGetTermination()	Get termination of specified COM port.
9	EApiComSetTermination()	Set termination of specified COM port.

4.1.1 EApiGPIOGetLevel

```
EApiGPIOGetLevel(
        __IN    ax_eapi_arm_id_t Id
        __IN    uint32_t Bitmask
        __OUT uint32_t* pLevel
 )
```

Description

Read high or low state on digital input/ output channels

In/Out	Parameter Name	Description
__IN	id	AX_EAPI_DI_0 AX_EAPI_DI_1 AX_EAPI_DO_0 AX_EAPI_DI AX_EAPI_DO
__IN	Bitmask	currently not supported by EApi, set it as 0
__OUT	pLevel	AX_EAPI_DI_* AX_EAPI_DO_* 0 : low 1: high AX_EAPI_DI AX_EAPI_DO 0~255 all DI/DO

4.1.2 EApiGPIOSetLevel

```
EApiGPIOSetLevel(
        __IN    ax_eapi_arm_id_t Id
        __IN    uint32_t Bitmask
        __IN uint32_t Level
)
```

Description

Write high or low state on digital input/ output channels.

In/Out	Parameter Name	Description
__IN	id	AX_EAPI_DO_0 AX_EAPI_DO

__IN	Bitmask	currently not supported by EApi, set it as 0
__IN	Level	AX_EAPI_DO_* 0 : low 1: high AX_EAPI_DO 0~255 set all DO high/low

4.1.3 EApiWDogStart

```
EApiWDogStart(
    __IN   uint32 Delay
    __IN   uint32_t EventTimeout
    __IN uint32_t ResetTimeout
)
```

Description

Start the watchdog timer and set the parameters.

In/Out	Parameter Name	Description
__IN	Delay	currently not supported by EApi, set it as 0
__IN	EventTimeout	currently not supported by EApi, set it as 0
__IN	ResetTimeout	Watchdog timeout interval in milliseconds to trigger a reset.

4.1.4 EApiWDogTrigger

```
EApiWDogTrigger(void)
```

Description

Trigger the watchdog timer

4.1.5 EApiWDogStop

```
EApiWDogStop(void)
```

Description

Stop the operation of the watchdog timer.

4.1.6 EApiComGetType

```
EApiComGetType(
    __IN    ax_eapi_arm_id_t Id
```

```
    __OUT   uint32_t* type
)
```

Description

Get COM port communication mode type.

In/Out	Parameter Name	Description
__IN	id	AX_EAPI_COM_1 AX_EAPI_COM_2
__OUT	type	1: RS232 Enable 2: RS485 2W Enable 3: RS422 /RS485 4W Enable

4.1.7 EApiComSetType

```
EApiComSetType(
    __IN   ax_eapi_arm_id_t Id,
    __IN   uint32_t type
)
```

Description

Set COM port communication mode type.

In/Out	Parameter Name	Description
__IN	id	AX_EAPI_COM_1 AX_EAPI_COM_2
__IN	type	1: RS232 Enable 2: RS485 2W Enable 3: RS422 /RS485 4W Enable

4.1.8 EApiComGetTermination

```
EApiComGetTermination(
``      __IN    ax_eapi_arm_id_t Id
      __OUT   uint32_t* enable
)
```

Description

Get termination of specified COM port.

In/Out	Parameter Name	Description
__IN	id	AX_EAPI_COM_1_TERMINATION

		AX_EAPI_COM_2_TERMINATION
__OUT	enable	0: disable 1: enable

4.1.9 EApiComSetTermination

```
EApiComSetTermination(
        __IN   ax_eapi_arm_id_t Id,
        __IN   uint32_t enable
)
```

Description

Set termination of specified COM port.

In/Out	Parameter Name	Description
__IN	id	AX_EAPI_COM_1_TERMINATION AX_EAPI_COM_2_TERMINATION
__IN	enable	0: disable 1: enable

4.2 Compile Demo Program

4.2.1 Build demo program

To compile and build demo program for IFB122, please do:

Change to *demo* directory.
cd /usr/src/eapi_demo

Build the demo program.

$ make

4.2.2 Run demo program

cd /usr/src/eapi_demo

#./testeapi

Chapter 5
Board Support Package (BSP)

5.1 Host Development System Installation

5.1.1 Install Host System

1. Download Ubuntu 18.04 or later LTS iso image.

2. Install Ubuntu 18.04 or later.

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

   ```
   $ sudo apt-get install gawk wget git-core diffstat unzip texinfo gcc-multilib build-essential chrpath socat cpio python python3 python3-pip python3-pexpect xz-utils debianutils iputils-ping python3-git python3-jinja2 libegl1-mesa libsdl1.2-dev pylint3 xterm rsync curl
   ```

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 https://storage.googleapis.com/git-repo-downloads/repo > ~/bin/repo
   $ 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
   ```

2. Setting up the Git environment

   ```
   ~$ git config --global user.name "Your Name"
   ~$ git config --global user.email "Your Email"
   ```

3. Download the Freescale's Yocto BSP source

   ```
   $ mkdir imx-yocto-bsp
   $ cd imx-yocto-bsp
   $ repo init -u https://source.codeaurora.org/external/imx/imx-manifest
   -b imx-linux-hardknott -m imx-5.10.72-2.2.0.xml
   $ repo sync
   ```

4. Extract Axiomtek's Yocto BSP source

   ```
   $ unzip IFB122_L510_v1.0.1.zip
   $ cp IFB122_L510_v1.0.1/Yocto_patches/ifb122_yocto_v1.0.1/meta-axiomtek imx-yocto-bsp/sources
   ```

   Check meta-axiomtek

5. Update bblayers.conf
   $ vi imx-yocto-bsp/sources/base/conf/bblayers.conf
   And add this line after *${BSPDIR}/sources meta-freescale-distro* \

   ${BSPDIR}/sources/meta-axiomtek \

6. First build
   Change to imx-yocto-bsp directory

   $ cd imx-yocto-bsp

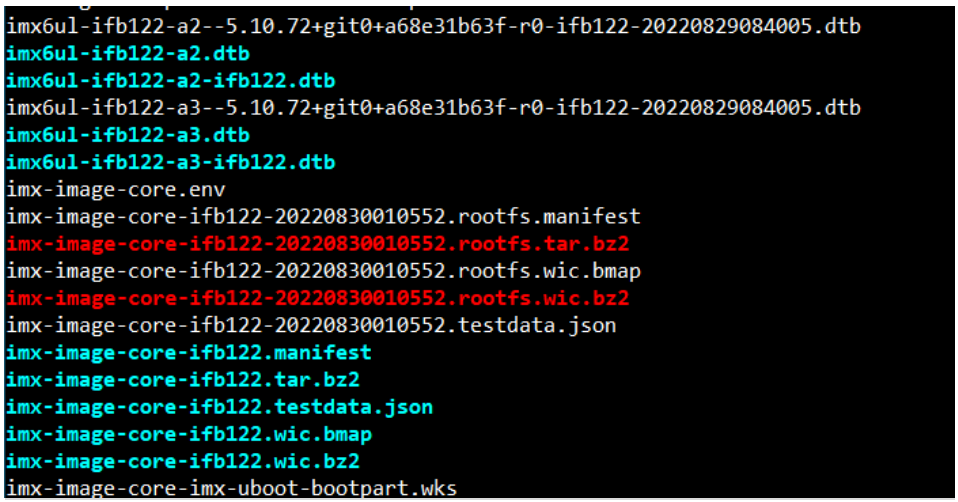
   Choose your board

   $ DISTRO=imx-xwayland-ifb122 MACHINE=ifb122 source imx-setup-release.sh -b build

   Start to build image

   $ bitbake imx-image-core

7. After build image finish, you can find the file path.
   The file path: imx-yocto-bsp/build/tmp/deploy/images/ifb122



5.1.3 Build and Install user's Yocto Toolchain

We have provided Yocto Toolchain in IFB122 BSP. However, if you want to build your toolchain by Yocto development, you can follow the instructions on 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 imx-image-core -c populate_sdk

2. After these steps to generate the toolchain into the Build Directory, you can find the file path: imx-yocto-bsp/build/tmp/deploy/sdk

   Install the toolchain into your host system /opt directory.
   Note: It needs root authorization
   $./fsl-imx-xwayland-glibc-x86_64-imx-image-core-cortexa7t2hf-neon-ifb122-toolchain-1.0.1.sh

```
axiomtek@axiomtek-PC:~$ ./fsl-imx-xwayland-glibc-x86_64-imx-image-core-cortexa
7t2hf-neon-ifb122-toolchain-1.0.1.sh
NXP i.MX Release Distro SDK installer version 1.0.1
==================================================
Enter target directory for SDK (default: /opt/fsl-imx-xwayland/1.0.1):
You are about to install the SDK to "/opt/fsl-imx-xwayland/1.0.1". Proceed [Y/
n]? Y
[sudo] password for axiomtek:
Extracting SDK.....
```

5.2 U-Boot for IFB122

5.2.1 Booting the System from eMMC (IFB122 default)

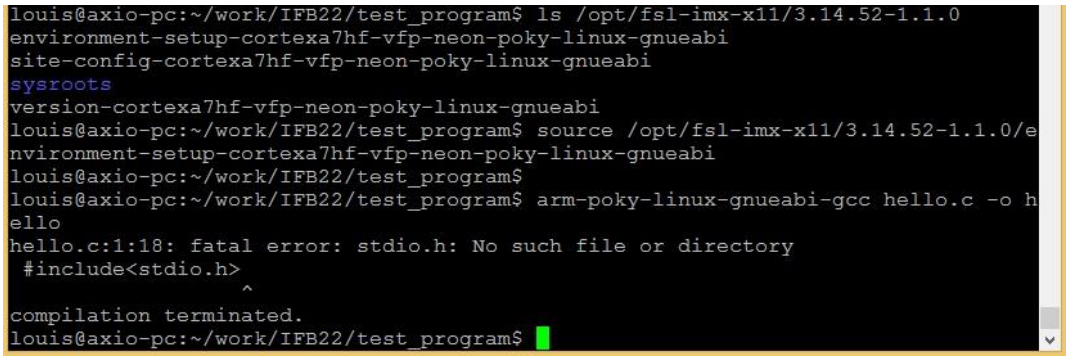
=> run bootcmd

```
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
```

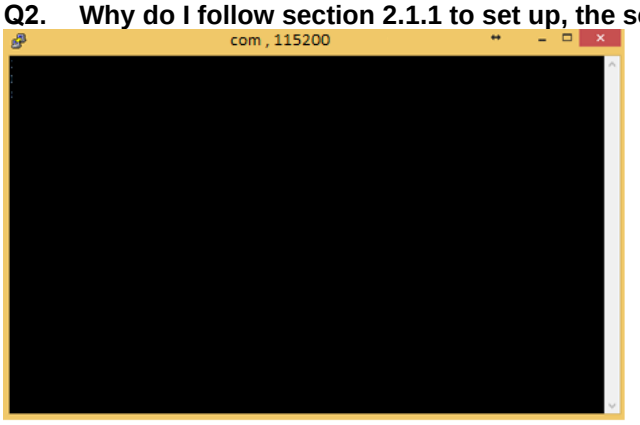
Appendix
Frequently Asked Questions

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

A1: Refer to section 2.2 for detailed information 2.2.1 Setting up the Cross-Development Environment
For example: $CC hello.c -o hello

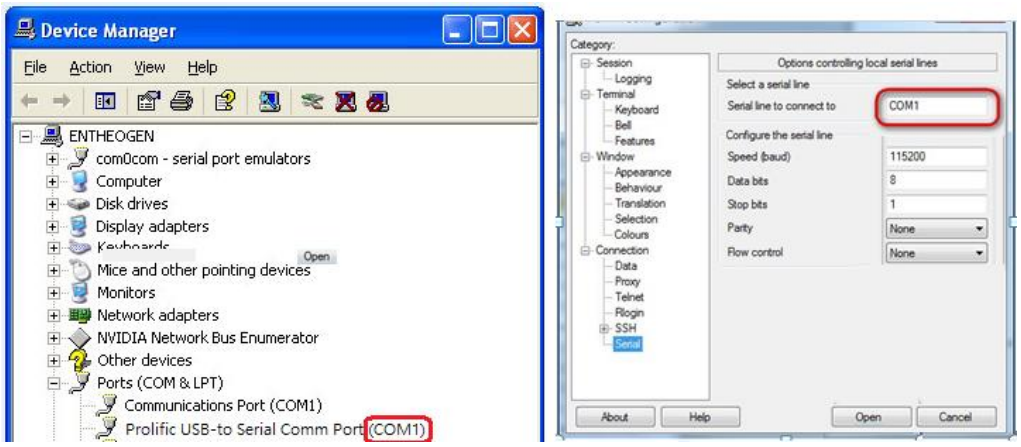


Q2. Why do I follow section 2.1.1 to set up, the screen is shown as below?



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 after following the instructions, or disconnect.

A3: To check your firewall been blocked in your host PC or router.