MYD-AM335x Linux 4.1.18 Development Guide

MYD-AM335X

MYD-AM335X-Y

MYD-AM335X-J

Table of Contents

Introduction 1.1

1. Software Resources 1.2

2. Deploy development environment 1.3

    2.1 Install tools 1.3.1

    2.2 Setup GCC Toolchain 1.3.2

3. Bulid System 1.4

    3.1 Bulid Bootloader 1.4.1

    3.2 Build Linux Kernel 1.4.2

    3.3 Build Filesystem 1.4.3

    3.4 Build QT 1.4.4

4. Linux Applications Development 1.5

    4.1 LCD 1.5.1

    4.2 Touch Panel 1.5.2

    4.3 RTC 1.5.3

    4.4 RS485 1.5.4

    4.5 CAN Bus 1.5.5

    4.6 Ethernet 1.5.6

    4.7 NAND Flash 1.5.7

    4.8 KeyPad 1.5.8

    4.9 GPIO-LED 1.5.9

    4.10 Audio 1.5.10

    4.11 USB Host 1.5.11

    4.12 USB Device 1.5.12

5. Qt Applications Development 1.6

    5.1 Install QtCreator 1.6.1

    5.2 Config QtCreator 1.6.2

    5.3 Build QT Application 1.6.3

6. Update System 1.7
7. Peripheral Module Use 1.8
7.1 4.3-Inch Resistive Touch Screen 1.8.1
7.2 WIFI Module 1.8.2
7.3 USB Camera Module 1.8.3
7.4 GPS Module 1.8.4
7.5 GPRS Module 1.8.5
Appendix Warranty & Technical Support Services 1.9

MYD-AM335X-Linux-4.1.18 Development Guide

Introduction

This section provides an overview of the areas covered by the documentation,what is contained, the appropriate readers, the documentation version history, and the applicable hardware version.

This chapter describes how to run Linux system and embedded Linux applications and the process of drive development in MYD-AM335X series development board. It includes building the development environment, compiling the source code, examples of Linux application and image download.

This document is suitable for embedded Linux development engineers with some development experience.

Version History

Version Number	Description	Time
V1.0	Initial Version	2017.4.1
V1.1	1. Add kernel and u-boot repository configuration for Buildroot 2. Add MEasy HMI demo to rootfs for development boards	2018.07.01
V1.2	1. Add support of core board with EMMC	2018.07.26
V1.3	1. Add chapters for MYIR peripheral module usage 2. Limit the speed of ETH1 on MYD-AM335X to 100BASE-T.	2018.08.28
V1.4	1. Add support of new EhterNet PHY chip YT8511 in kernel. 2. Capacitive and resistive touch screen are compatible in buildroot. 3. Please fetch sources from	2021.11.18

	https://github.com/MYiR-Dev/myir-ti-buildroot/releases/tag/MYD-AM335X_V20R5_20211118	

Hardware Version

MYD-AM335X, MYD-AM335X-Y,MYD-AM335X-J.

> **Note: This document applies to the above three versions of the hardware, the user can choose the corresponding operation according to the corresponding board.In this paper, the MYD-AM335X series is used to represent the three board.**

1. Software Resources

This section describes the software resources of the MYD-AM335X series development boards.

Table 1-1 Software Resources.

Category	Name	Description	Source	MYD-AM335X	MYD-AM335Y
Bootloader	U-boot	Responsible for system initialization and boot kernel.	YES	√	√
Kernel	Linux 4.1.18	Designed for MYD-AM335X hardware	YES	√	√
Drivers	LCD Controller	LCD driver，for 4.3 inch、7 inch	YES	√	√
Drivers	Touch Panel	Resistive touch screen driver	YES	√	√
Drivers	Touch Panel	Capacitance touch screen driver	YES	√	√
Drivers	RTC	RTC clock driver	YES	√	√
Drivers	UART	Serial driver	YES	√	√
Drivers	SDIO WiFi	WiFi driver	YES	×	√
Drivers	RS485	RS485 driver	YES	√	√
Drivers	CAN	CAN driver	YES	√	√
Drivers	Ethernet	Ethernet driver	YES	√	√
Drivers	MMC/SD	MMC/SD driver	YES	√	√
Drivers	NAND Flash	NAND Flash driver	YES	√	√
Drivers	Audio	Audio driver	YES	√	√
Drivers	GPIO-LED	GPIO-LED driver	YES	√	√
Drivers	I2C	I2C driver	YES	√	√

Drivers	HDMI	HDMI driver	YES	√	×
Drivers	PMU	Power Management Unit driver	YES	√	√
Drivers	USB Host	USB Host driver	YES	√	√
Drivers	USB Device	USB Device driver（Gadget）	YES	√	√
Filesystem	Rootfs	Base on buildroot	Bin	√	√
Filesystem	Rootfs-qt	Qt filesystem	Bin	√	√
Filesystem	UBI	Provide image file	YES	√	√
Filesystem	Ramdisk.gz	ramdisk filesystem	YES	√	√
Demo	LCD	LCD test program	YES	√	√
Demo	RTC	RTC clock test program	YES	√	√
Demo	RS485	RS485 test program	YES	√	√
Demo	CAN	CAN test program	YES	√	√
Demo	Ethernet	Ethernet test program	YES	√	√
Demo	NAND Flash	NAND Flash test program	YES	√	√
Demo	GPIO-LED	GPIO-LED test program	YES	√	√
Demo	Audio	Audio test program	YES	√	√
Demo	Qt	QT environment verification program	YES	√	√

2. Deploy development environment

This section describes the tools and environment, tool chain settings, development environment validation.

PC development environment: Ubuntu12.04/Ubuntu14.04/Ubuntu16.04 64bit Desktop

Cross compiler: gcc5.3(Linaro GCC 2016.02)

Hardware debugging environment structures

- MYD-AM335X Hardware debugging environment structures：

Connect the debug serial port J12 to the PC,set the baudrate of serial port on host PC to 115,200-8-n-1. Specific as follows:

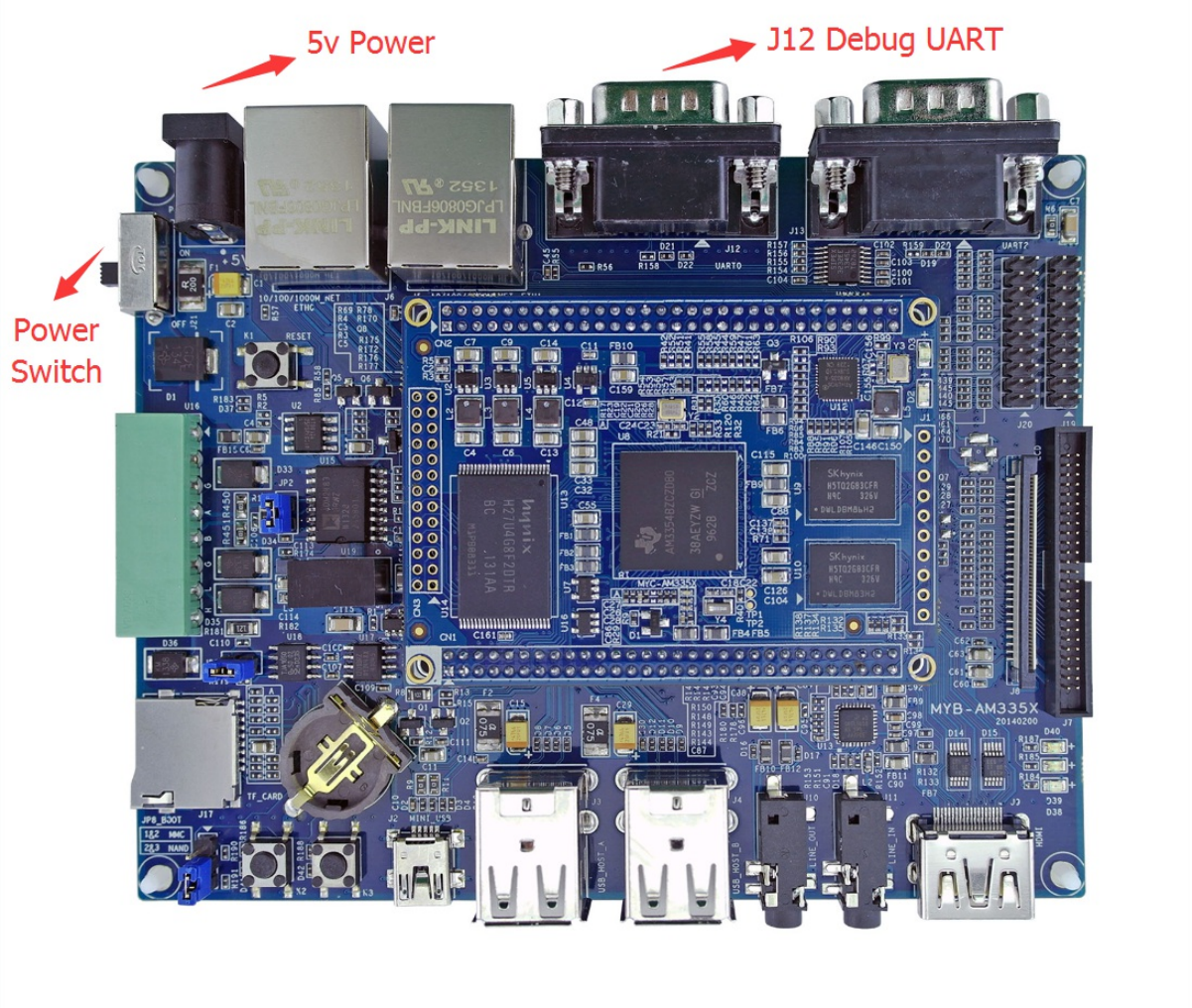

Figure 2-1 MYD-AM335X Hardware Debugging Interface

- MYD-AM335X-Y Hardware debugging environment structures：

Connect the debug serial port J10 to the PC,set the baudrate of serial port on host PC to 115,200-8-n-1. Specific as follows:

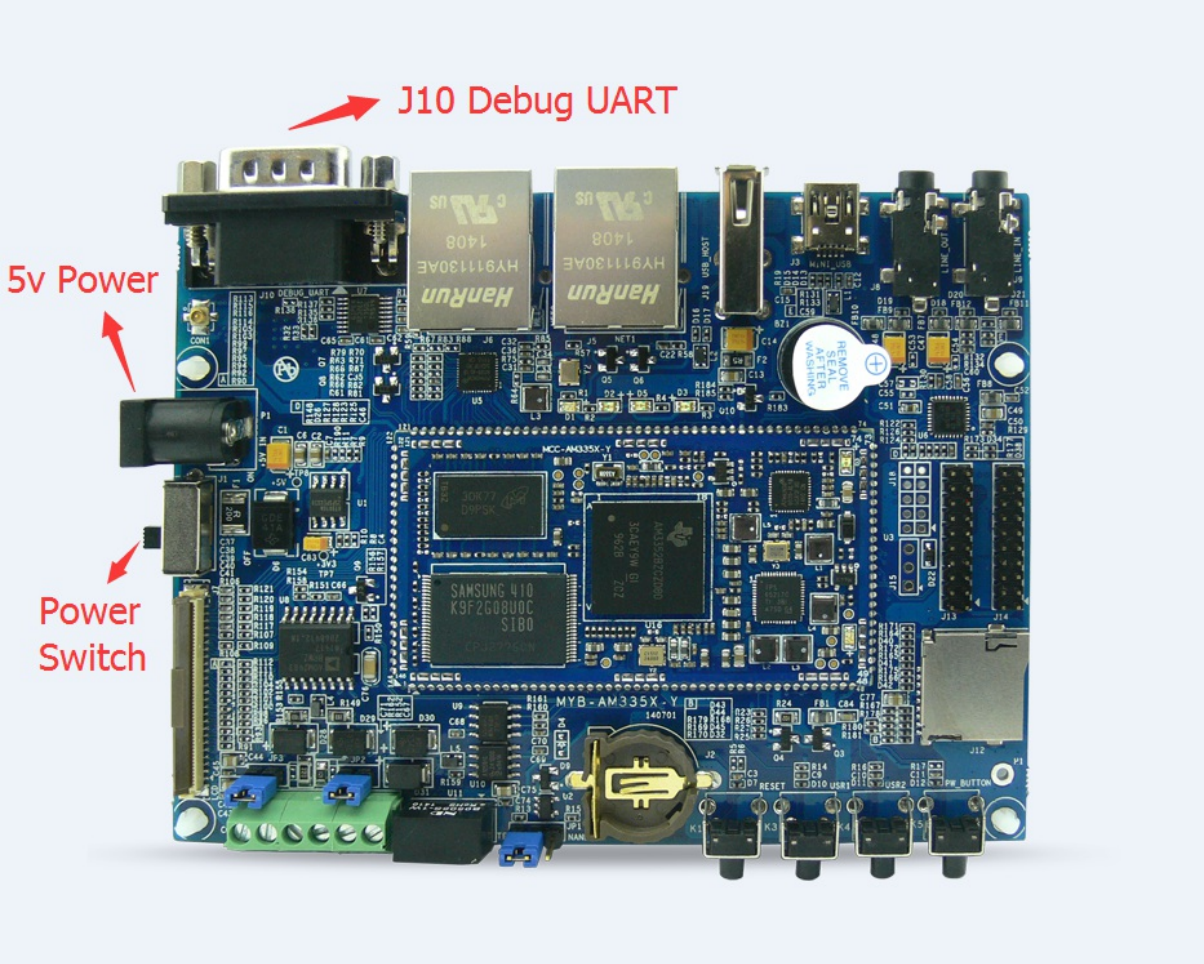

Figure 2-2 MYD-AM335X-Y Hardware Debugging Interface

- MYD-AM335X-J Hardware debugging environment structures：

Connect the debug serial port J3 to the PC,set the baudrate of serial port on host PC to 115,200-8-n-1. Specific as follows:

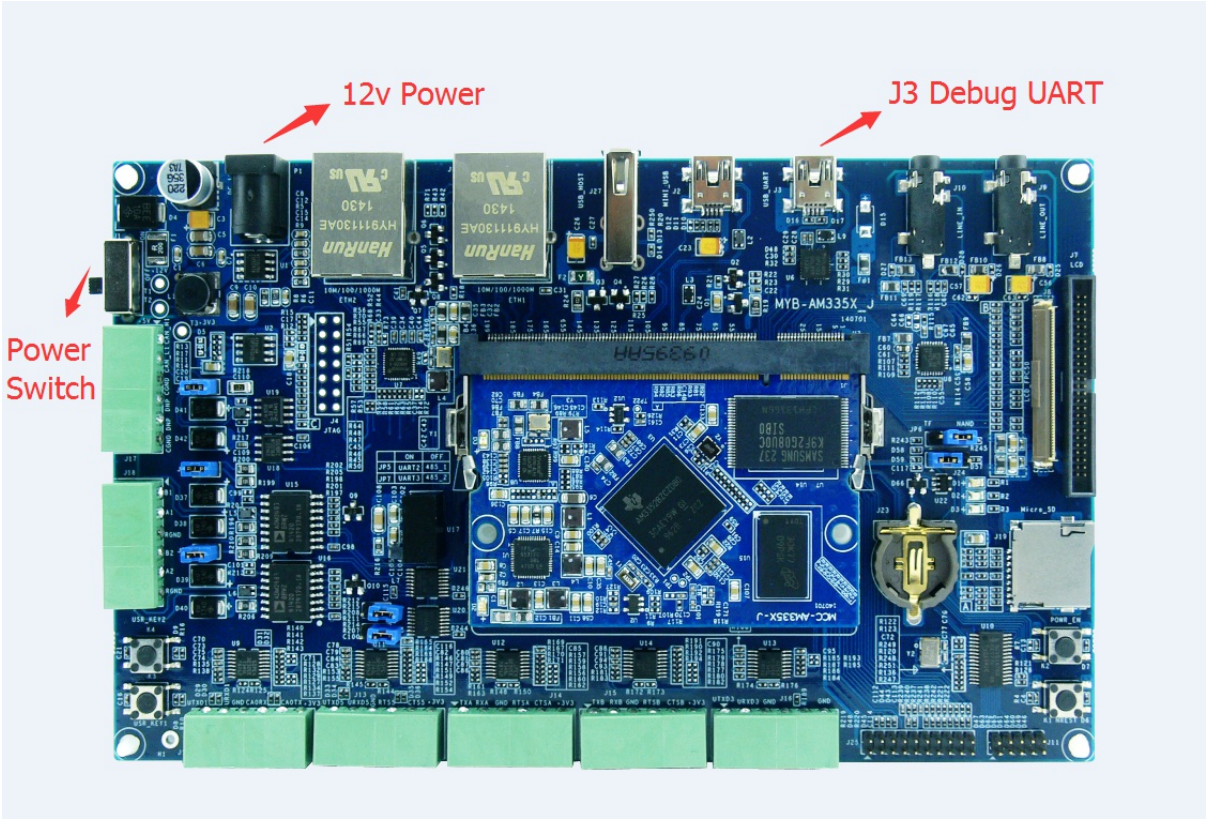

Figure 2-3 MYD-AM335X-J Hardware Debugging Interface

Create a working directory:

Create a work directory and copy the resources from 04-Linux_Source of our release package to the work directory on ubuntu host PC， here is defined by customers according to their development environment.

```
$ mkdir -p <WORKDIR>
$ cp /media/cdrom/04-Linux_Source/* <WORKDIR> -rf
$ ls  <WORKDIR>
Bootloader/ Examples/  Filesystem/  Kernel/  Patches/ ToolChain/
```

2.1 Install tools

Install other development tools

For the sake of convenience we install all the tools and libraries before as shown below:

- Ubuntu 12.04 64Bit Desktop

```
sudo apt-get install build-essential git-core libncurses5-dev
$ sudo apt-get install flex bison texinfo zip unzip zlib1g-dev gettext
$ sudo apt-get install gperf libsdl-dev libesd0-dev libwxgtk2.6-dev
$ sudo apt-get install uboot-mkimage
$ sudo apt-get install g++ xz-utils
```

- Ubuntu 14.04 64Bit Desktop

```
$ sudo apt-get install build-essential git-core libncurses5-dev u-boot-tools
$ sudo apt-get install flex bison texinfo zip unzip zlib1g-dev gettext
$ sudo apt-get insta 64Bit Desktopll gperf libsdl-dev libesd0-dev
$ sudo apt-get install g++ xz-utils
$ sudo apt-get install subversion
```

- Ubuntu 16.04 64Bit Desktop

```
$ sudo apt-get install build-essential git-core libncurses5-dev u-boot-tools
$ sudo apt-get install flex bison texinfo zip unzip zlib1g-dev gettext
$ sudo apt-get install gperf libsdl-dev libesd0-dev
$ sudo apt-get install g++ xz-utils
$ sudo apt-get install subversion
```

  On 64bit Ubuntu OS, some 32bit runtime libraries should be installed as shown below:

```
$sudo apt-get install libc6-i386 lib32stdc++6 lib32z1
```

2.2 Setup GCC Toolchain

Set environment variables:

```
$ cd <WORKDIR>/Toolchain
$ tar xvf gcc-linaro-5.3-2016.02-x86_64_arm-linux-gnueabihf.tar.xz
$ export PATH=$PATH:<WORKDIR>/Toolchain/gcc-linaro-5.3-2016.02-x86_64_arm-linux-gnueab
ihf/bin
$ export ARCH=arm
$ export CROSS_COMPILE=arm-linux-gnueabihf-
```

After executing the "export" command,type "arm" then press TAB to check if it's correctly set. This setting is valid only for the this terminal, for a permanent modification, please modify user profiles.

```
eg:Set user environment variables
vi ~/.profile
```

At the end to add:

```
export ARCH=arm
export CROSS_COMPILE=arm-linux-gnueabihf-
export PATH=$PATH:<WORKDIR>/Toolchain/
gcc-linaro-5.3-2016.02-x86_64_arm-linux-gnueabihf/bin
```

```
source ~/.profile or Reset Ubuntu
```

Test：

```
$echo $ARCH
arm
$echo $CROSS_COMPILE
arm-linux-gnueabihf-
```

Cross compiler verification：

```
arm-linux-gnueabihf-gcc -v
```

```
$ arm-linux-gnueabihf-gcc -v
Using built-in specs.
COLLECT_GCC=arm-linux-gnueabihf-gcc
……
Thread model: posix
gcc version 5.3.1 20160113 (Linaro GCC 5.3-2016.02)
```

3. Bulid System

There are many open source tools for building an embedded Linux system, they are more convenient for embedded software engineers to build bootloader, kernel, filesystem all in one step. Currently, OpenWRT, Buildroot , Yocto are more commonly used.

Buildroot is a simple, efficient and easy-to-use tool to generate embedded Linux systems through cross-compilation,thanks to its kernel-like menuconfig, gconfig and xconfig configuration interfaces, building a basic system with Buildroot is easy, so it's very popular among embedded software engineers.

The following sections will explain U-boot, kernel, filesystem respectively, in the part of filesystem, we build a filesystem with QT5 included, customers can develop QT5 application based on this filesystem easily.

3.1 Bulid Bootloader

Enter the directory of bootloader, uncompress the source code package as shown below：

```
$ cd <WORKDIR>/Bootloader
$ tar -jxvf myir-u-boot.tar.bz2
$ cd myir-u-boot
```

- Compile U-Boot：

The configuration of U-boot for MYD-AM335X series is located at *myir-u-boot/configs/*, the corresponding configuration file name are shown below:

Table 3-1-1 U-boot Config Files for MYIR Development Boards

Board	Configuration File Name
MYD-AM335X(NAND)	myd_c335x_defconfig
MYD-AM335X(EMMC)	myd_c335x_emmc_defconfig
MYD-AM335X-J(NAND)	myd_j335x_defconfig
MYD-AM335X-J(EMMC)	myd_j335x_emmc_defconfig
MYD-AM335X-Y(NAND)	myd_y335x_defconfig
MYD-AM335X-Y(EMMC)	myd_y335x_emmc_defconfig

Compile U-boot for MYD-AM335X development board with NAND：

```
$ make distclean
$ make myd_c335x_defconfig
$ make
```

After compiling is completed, MLO and u-boot.img files will be generated in `myir-u-boot` directory.

3.2 Build Linux Kernel

Enter the work directory and uncompress the Linux Kernel source code package：

```
$ cd <WORKDIR>/
$ tar -jxvf myir-kernel.tar.bz2
$ cd myir-kernel
```

Compile Kernel：

The configuration of Kernel for MYD-AM335X series is located at myir-kernel/arch/arm/configs/,customers can compile Kernel as shown below:

Table 3-2-1 Kernel Config Files for MYIR Development Boards

Board Type	Configuration File Name
MYD-AM335X	myd_c335x_defconfig
MYD-AM335X-Y	myd_y335x_defconfig
MYD-AM335X-J	myd_j335x_defconfig

Compile Kernel for MYD-AM335X development board：

If users want to compile and install kernel modules, they should set `INSTALL_MOD_PATH`, it is useful for debug kernel modules with NFS.

```
$ export  INSTALL_MOD_PATH=$HOME/export/rootfsa/
$ make distclean
$ make myd_c335x_defconfig
$ make zImage dtbs
$ make modules
$ make modules_install
```

The kernel modules have a version magic, it should match the version of zImage. So if users change the version of kernel, they should recompile the zImage and kernel modules together. If the version does not match, the kernel will complain as below:

```
[ 2750.480576] ti_am335x_adc: disagrees about version of symbol dev_warn
[ 2750.487670] ti_am335x_adc: Unknown symbol dev_warn (err -22)
[ 2750.493977] ti_am335x_adc: disagrees about version of symbol dev_err
```

```
[ 2750.502474] ti_am335x_adc: Unknown symbol dev_err (err -22)
```

The configuration for different boards are shown in the table above. After compiling, the kernel image is generated at `arch/arm/boot/zImage`, the DTB files are generated at `arch/arm/boot/dts/myd_c335x.dtb, arch/arm/boot/dts/myd_c335x_emmc.dtb`. The dtb file `arch/arm/boot/dts/myd_c335x.dtb` is for core board with NAND, the dtb file `arch/arm/boot/dts/myd_c335x_emmc.dtb` is for core board with EMMC.

Patch files for different hardware features：

The default configuration is 7-inch screen，256N256D，without SGX. Users can generate a different screen and NAND DDR device tree file from the patch file in `<WORKDIR>/04-Linux_source/Patches`，and then compile the kernel using the device tree file.The following is a list of related patches:

Table 3-2-2 Kernel Patches for MYIR Development Boards

Board Type	MYD-AM335X	MYD-AM335X-Y	MYD-AM335X-J
HDMI patches	myd_c335x_hdmi_display.diff	No expansion	No expansion
4.3 inch screen patch	myd-am335x-lcd4.3.diff	myd-am335x-y-lcd4.3.diff	myd-am335x-j-lcd4.3.diff
SGX patch	myd-am335x-sgx.diff	myd-am335x-y-sgx.diff	myd-am335x-j-sgx.diff

For example: Patch for MYD-AM335X-Y with SGX , 4.3-inch screen:

```
$ patch -p1 < <WORKDIR>/Patches/myd-am335x-y-lcd4.3.diff

$ patch -p1 < <WORKDIR>/Patches/myd-am335x-y-sgx.diff
```

Compile device tree：

```
make dtbs
```

And use the arch/arm/boot/dts directory of the device tree file to replace the system image device tree file.

3.3 Build Filesystem

This section covers the building of filesystem with Buildroot.

3.3.1 Preparation before Building Buildroot

> Note1: After modifying source code of Kernel or U-boot, Buildroot can not update and build it automatically. Customers should commit it to the master branch of their local git repo manually.
>
> Note2: If the source code of Kernel is updated, before building Buildroot again, customers should remove the package "myir-buildroot/dl/linux-master.tar.gz" and the "myir-buildroot/output/build/linux-master" and "myirbuildroot\output\build\linux-headers-master" directories manually. The same to rebuilding of U-boot.
>
> Note3：Before building u-boot and kernel with buildroot, users need to create their own git repository for u-boot and kernel, then replace the git path in the configuration files.

Copy the Buildroot source package customized by MYIR Tech from *04-Linux_Source/Filesystem/myir-buildroot.tar.gz* of our release package to work directory and uncompress it. The content of myir-buildroot.tar.gz is shown below:

```
$ ls -al <WORKDIR>/Filesystem/myir-buildroot
arch   CHANGES           configs     dl    linux            output   support
board  Config.in         COPYING     docs  Makefile         package  system
boot   Config.in.legacy  DEVELOPERS  fs    Makefile.legacy  README   toolchain
```

For more details about the file structure of `Buildroot`, please refer to Buildroot manual https://buildroot.org/downloads/manual/manual.html.The board support files for `MYD AM335X` series development boards are located at `<WORKDIR>/Filesystem/myir-buildroot/board/myir/myd_c335x`, `<WORKDIR>/Filesystem/myir-buildroot/board/myir/myd_y335x`, `<WORKDIR>/Filesystem/myir-buildroot/board/myir/myd_j335x`

3.3.2 Buildroot Configuration

The configuration files for Buildroot are all located at `<WORKDIR>/Filesystem/myir-buildroot/configs/` .

Table 3-3-1 Buildroot Config Files for MYIR Development Boards

Config File	Description
myd_c335x_defconfig	Buildroot configuration without QT5 for MYC-AM335X CPU Module with NAND Flash
myd_c335x_emmc_defconfig	Buildroot configuration without QT5 for MYC-AM335X CPU Module With EMMC
myd_c335x_qt5_defconfig	Buildroot configuration with QT5 for MYD-AM335X development board with NAND
myd_j335x_defconfig	Buildroot configuration without QT5 for MYC-AM335X-J CPU Module with NAND Flash
myd_j335x_emmc_defconfig	Buildroot configuration without QT5 for MYC-AM335X-J CPU Module with EMMC
myd_j335x_qt5_defconfig	Buildroot configuration with QT5 for MYD-AM335X-J development board with NAND Flash
myd_y335x_defconfig	Buildroot configuration without QT5 for MYC-AM335X-Y CPU Module with NAND Flash
myd_y335x_defconfig	Buildroot configuration without QT5 for MYC-AM335X-Y CPU Module with EMMC
myd_y335x_qt5_defconfig	Buildroot configuration with QT5 for MYD-AM335X-Y development board with NAND Flash

```
$ cd myir-buildroot
$ make clean
$ make myd_y335x_defconfig
```

Customers can change the configuration by its kernel-like menuconfig with the command `make menuconfig` . The main configuration for `MYD-AM335X-Y` development board are listed below.

Configuration for Cross Compiler:

Buildroot can use internal cross compile toolchain generated by Buildroot itself, it can also use external cross compile toolchain. In this document, we choose the internal cross compile toochain, it will be generated and stored to `<WORKDIR>/Filesystem/myir-buildroot/output/host/usr/bin/` after compiling.

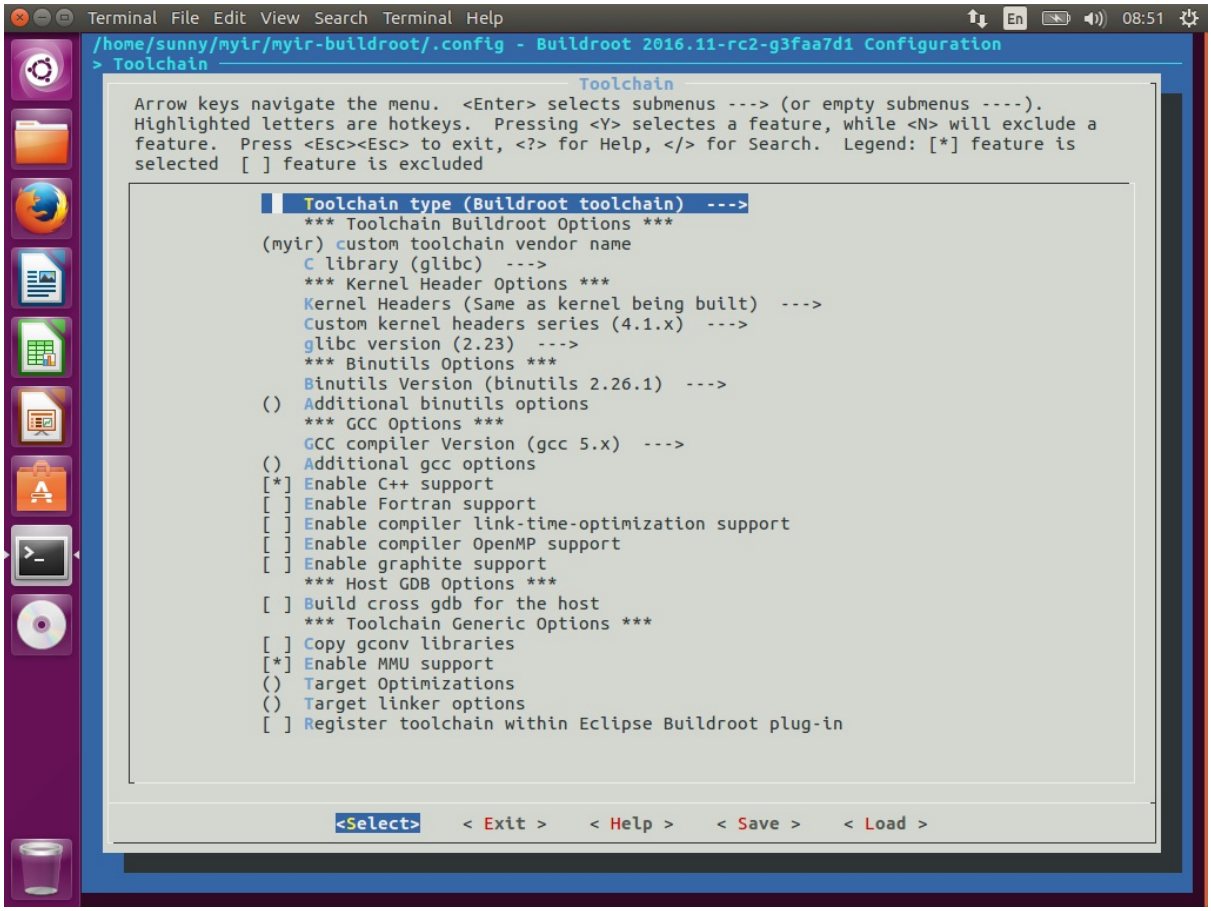

Figure 3-3-1 Configuration for Cross Compiler

Configuration for System:

The configuration for system includes the name of the target system, the welcome message, the init subsystem(busybox/systemV/systemd) and device manage system, customers can also set the password for root user by configuration. For MYD-AM335X-Y development board, the password for root is set to `myirtech` as shown below. If customers do not need to set password, they no need to config the password.

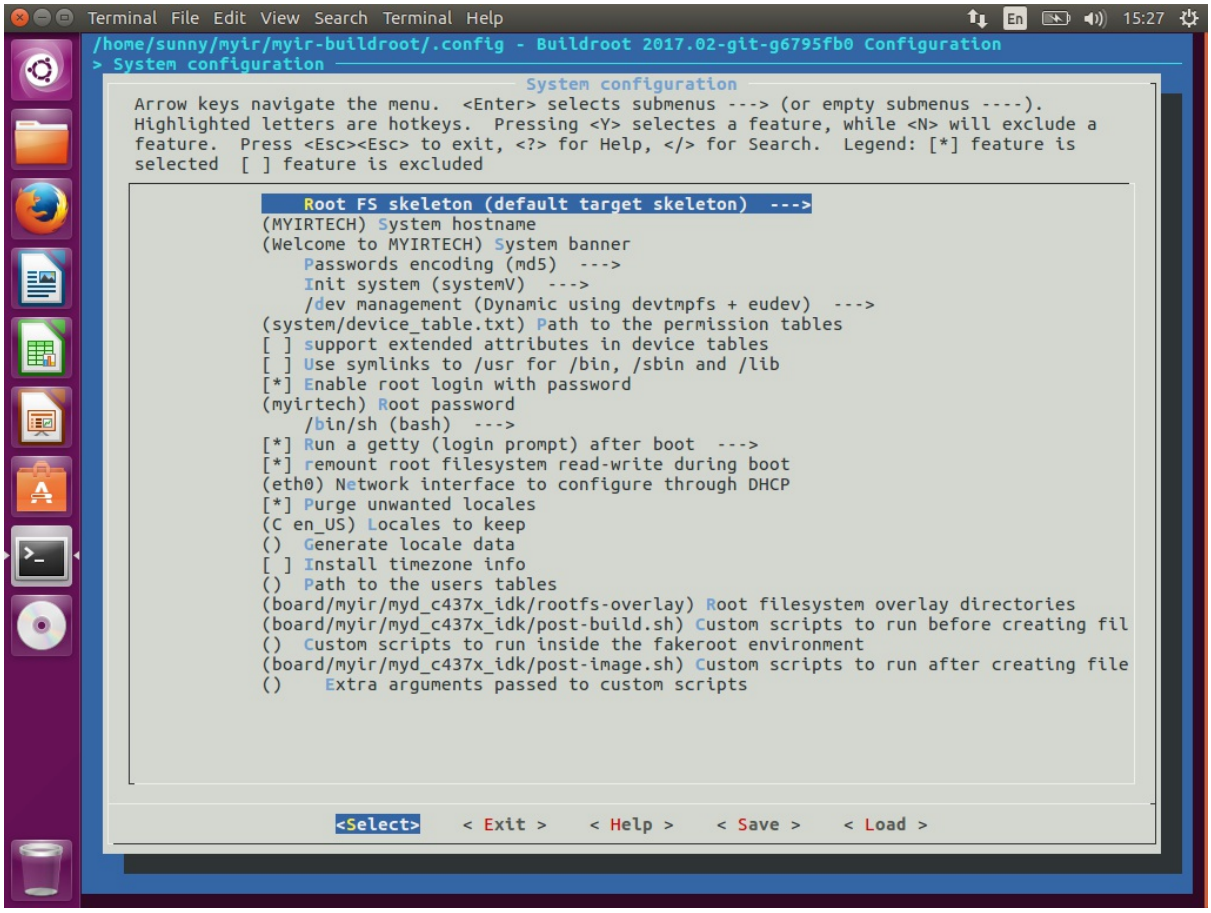

Figure 3-3-2 Configuration for System

Configuration for Bootloader:

The configuration for Bootloader includes the URL of the source code of U-boot, the U-boot configuration file name, the output images of U-boot and so on. They are shown in Figure 3-3-3 below.

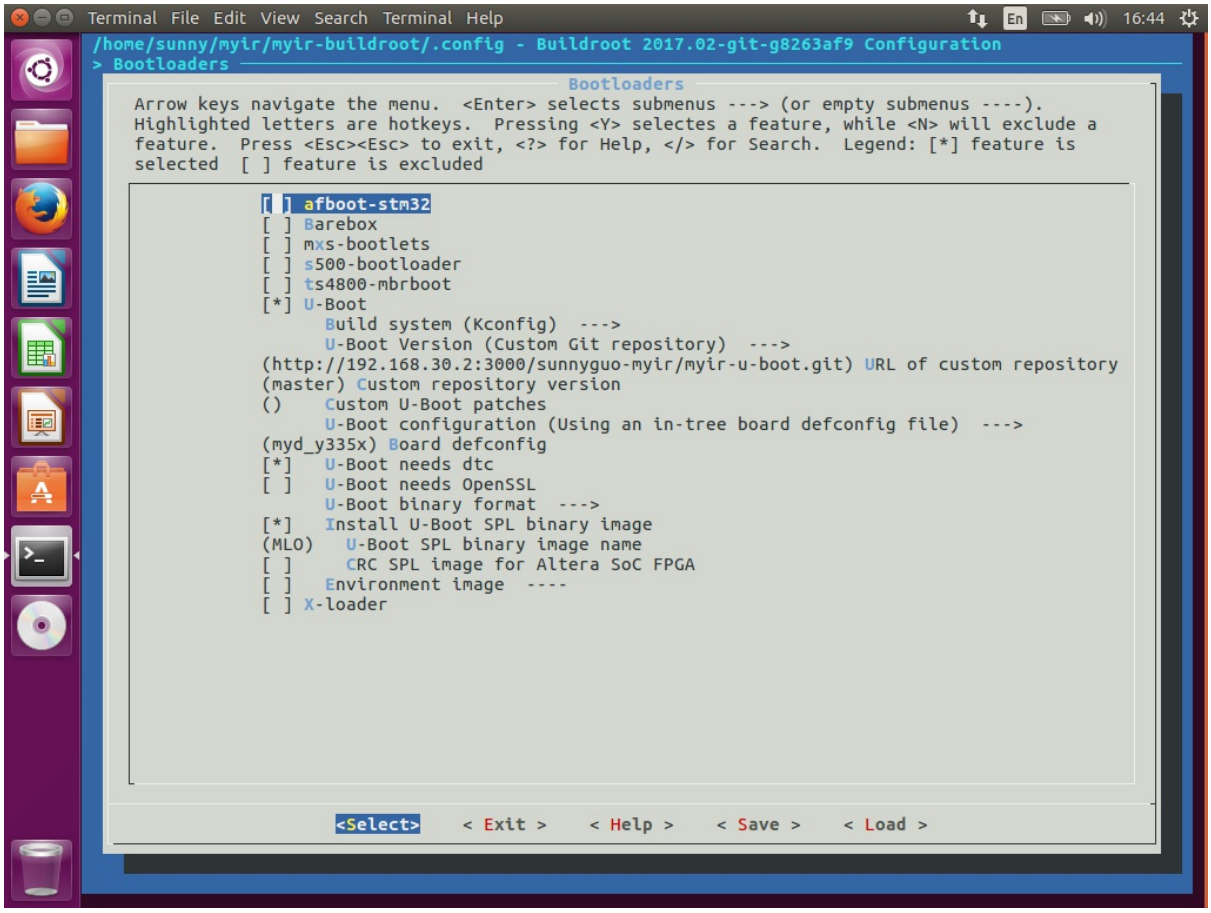

Figure 3-3-3 Configuration for Bootloader

We fetch the source code of U-boot with git from local repository. Users need to build their own git repository, they can also use other protocols or even local directory. For other protocols, please refer to the Buildroot manual. Create a U-boot git repository from `myir-u-boot.tar.gz` :

```
$ cd ~/
$ tar zxvf myir-u-boot.tar.gz
$ cd myir-u-boot
$ git init
$ git add . -f
$ git commit -m "Initial Version" -a
```

And then modify and replace the two items of the configuration files located at `<WORKDIR>/Filesystem/buildroot/configs/myd_y335x_defconfig` and `<WORKDIR>/Filesystem/buildroot/configs/myd_y335x_qt5_defconfig` , as shown below:

```
BR2_TARGET_UBOOT_CUSTOM_REPO_URL="~/myir-u-boot/.git"
BR2_TARGET_UBOOT_CUSTOM_REPO_VERSION="master"
```

Configuration for Kernel:

The configuration of Kernel is similar to that of bootloader.

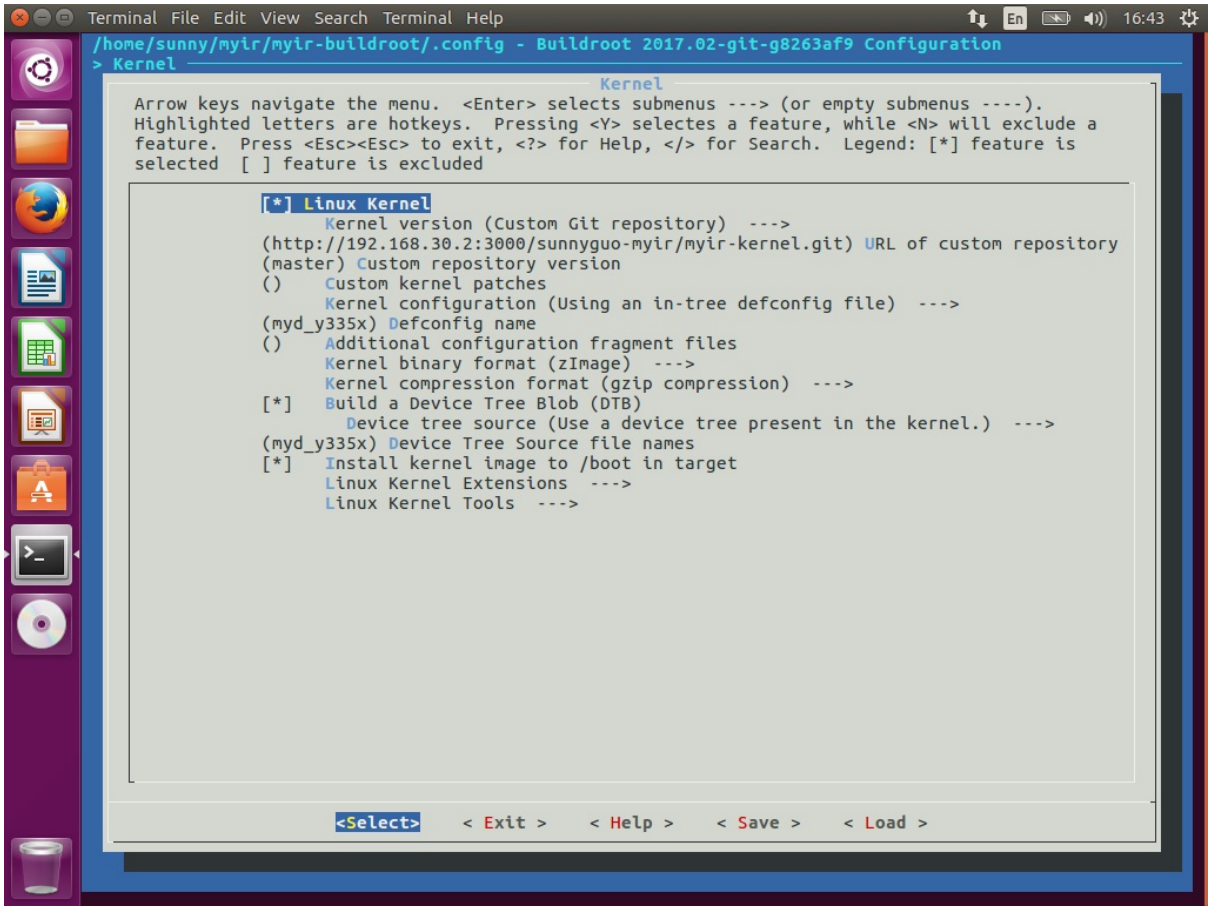

Figure 3-3-4 Configuration for Kernel

We fetch the source code of Kernel with git from local repository. Users need to build their own git repository, they can also use other protocols or even local directory. For other protocols, please refer to the Buildroot manual. Create a kernel git repository from `myir-kernel.tar.gz` :

```
$ cd ~/
$ tar zxvf myir-kernel.tar.gz
$ cd myir-u-boot
$ git init
$ git add . -f
$ git commit -m "Initial Version" -a
```

And then modify and replace the two items of the configuration files located at `<WORKDIR>/Filesystem/buildroot/configs/myd_y335x_defconfig` and `<WORKDIR>/Filesystem/buildroot/configs/myd_y335x_qt5_defconfig`, as shown below:

```
BR2_TARGET_KERNEL_CUSTOM_REPO_URL="~/myir-kernel/.git"
BR2_TARGET_KERNEL_CUSTOM_REPO_VERSION="master"
```

Configuration for Filesystem:

The configuration for filesystem determines what filesystem images are generated in *myir-buildroot/output/images/* directory after compiling, If we choose `ramdisk` in the configuration, we will get a ramdisk filesystem image. EXT2/4, UBIFS, and rootfs tar package can also be create if they are choosed in configuration.

By the way, the rootfs.tar.gz can be uncompressed and used as the nfsroot directory, it can also be made to other formats of filesystem images by host mtd-utils. For example, we can create UBIFS filesystem image without building Buildroot again after doing some modification for rootfs.Firstly, we create a file `ubinize.cfg` as shown below:

```
[ubifs]
mode=ubi
vol_id=0
vol_type=dynamic
vol_name=rootfs
vol_alignment=1
vol_flags=autoresize
image=rootfs.ubifs
```

Then, make a UBIFS image with UBIFS tools by the following processes:

```
$ export PATH=$PATH:<WORKDIR>/Filesystem/myir-buildroot/output/host/usr/sbin
$ tar zxvf rootfs.tar.gz
$ mkfs.ubifs -d rootfs -e 0x1f000 -c 2048 -m 0x800 -x lzo -F -o  rootfs.ubifs
$ ubinize -o rootfs.ubi -m 0x800 -p 0x20000 -s 512 -m 2048 -O 2048 ubinize.cfg
```

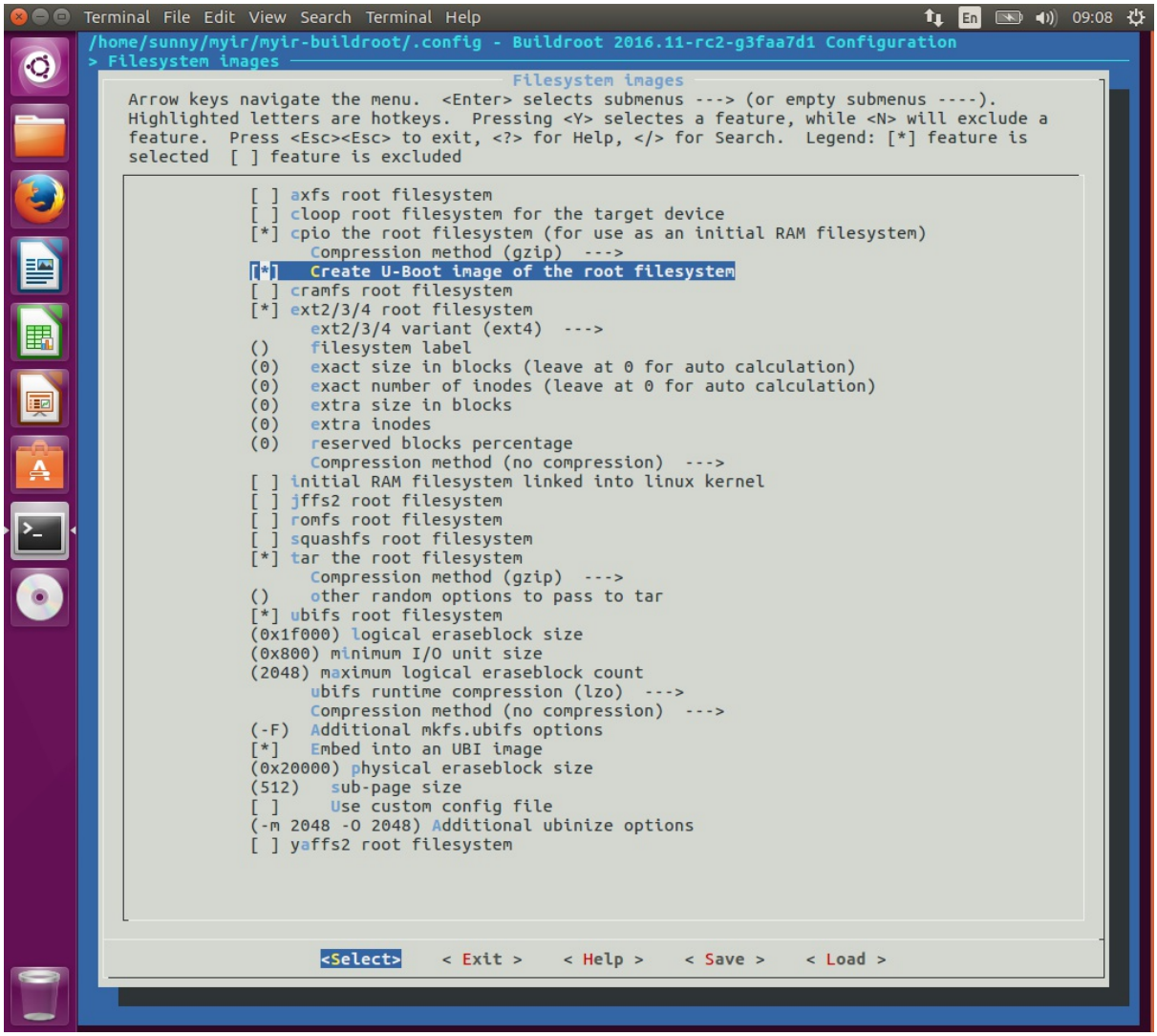

Figure 3-3-5 Configuration for Filesystem

Configuration for Target Packages:

The configuration for target packages is easier, but it is changed more frequently. Customers can choose some hardware tools, such as I2C-tools, spi-tools, can-utils and so on, build them into the filesystem images for debugging. Some network utils, such as DHCP, TFTP, SSH and so on, can aslo be choosed and built into the filesystem images for production. Most commonly used tools are included in the target packages of Buildroot. Customers can also write new target packages and integrate them to Buildroot, please refer to https://buildroot.org/downloads/manual/manual.html#adding-packages for details.

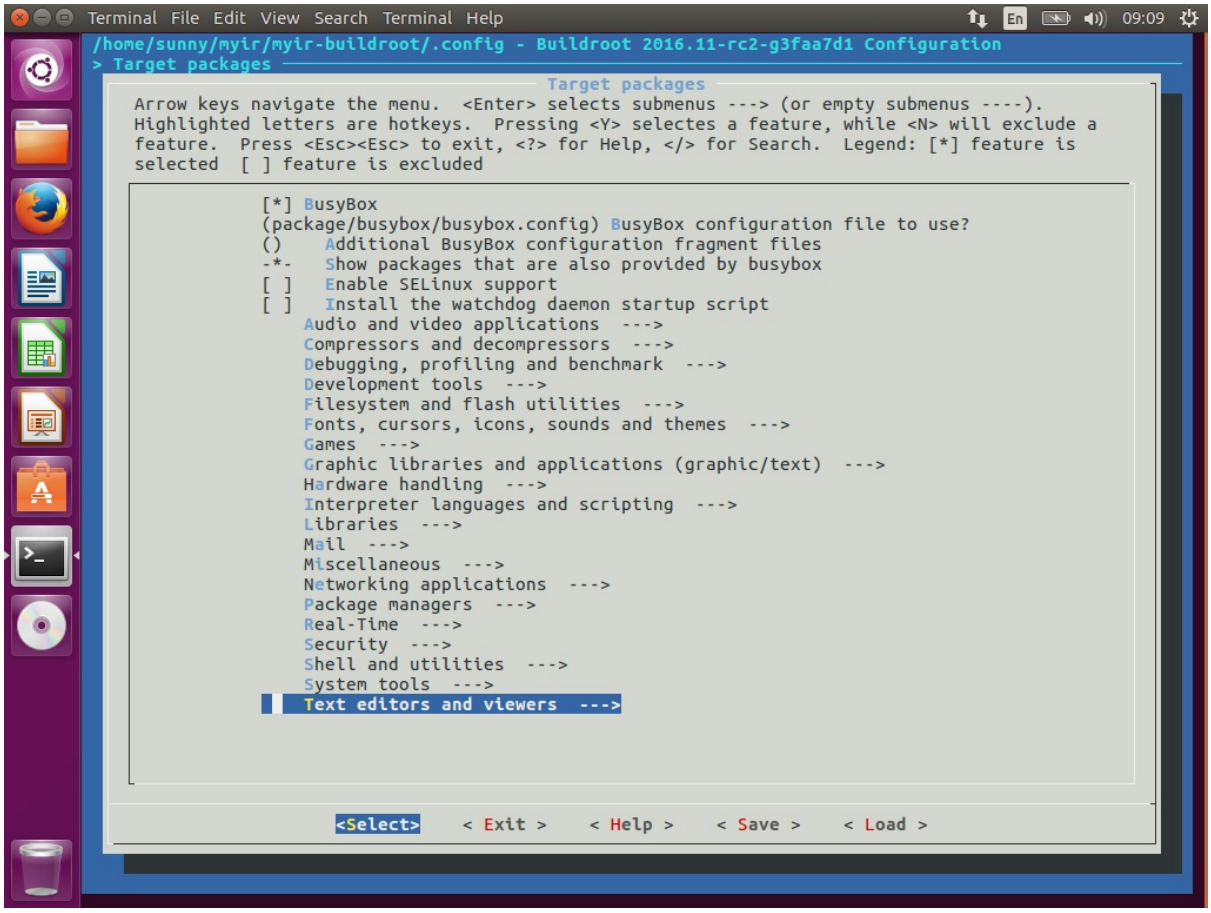

Figure 3-3-6 Configuration for Target Packages

3.3.3 Build Buildroot

Customers can build Buildroot just like building kernel as shown below:

```
$ cd myir-buildroot
$ make
```

An output directory is generated during compilation，The resulte output image files are located at the `<WORKDIR>/Filesystem/myir-buildroot/output/images` directory.

```
$ls -al output/images
boot.vfat  MLO_usbmsc          rootfs.cpio        rootfs.tar     u-boot_emmc.img  u-bo
ot_usbmsc.img    uEnv.txt
MLO        myd_y335x.dtb       rootfs.cpio.gz     rootfs.tar.gz  u-boot.img       uEnv
_mmc.txt         uEnv_usbmsc_ramdisk.txt
MLO_emmc   myd_y335x_emmc.dtb  rootfs.cpio.uboot  rootfs.ubi     u-boot_nand.img  uEnv
_ramdisk.txt     uEnv_usbmsc.txt
```

```
MLO_nand    ramdisk.gz          rootfs.ext2        rootfs.ubifs    u-boot_sd.img    uEnv
_sd_ramdisk.txt  zImage
MLO_sd      readme.txt          rootfs.ext4        sdcard.img      u-boot-spl.bin   uEnv
_sd.txt
```

The bootloader, kernel and all kinds of filesystem images are generated all in one step, they will be introduced in the subsequent section. A cross compile toolchain is also generated at `<WORKDIR>/Filesystem/myir-buildroot/output/` , users can setup this toolchain as follows(suppose myir-buildroot is put into $HOME):

```
export  PATH="$HOME/myir-buildroot/output/host/usr/bin:$PATH"
export  PATH="$HOME/myir-buildroot/output/host/usr/sbin:$PATH"
export  CROSS_COMPILE=arm-myir-linux-gnueabihf-
export  TARGET_CC=arm-myir-linux-gnueabihf-gcc
export  ARCH=arm
```

3.3.4 Filesystem Built by Arago

Customers can also run an demo filesystem image created with Arago on a MYD-AM335X series development board, it was created by TI, please refer to the WIKI page on TI's website.
http://processors.wiki.ti.com/index.php/Processor_SDK_Building_The_SDK.

3.4 Build QT

QT5 is included in Buildroot as a target package, we have provided a config file with QT5 for MYD-AM335X series development board, so it is easy to build filesystem images with QT5 shown as below.

```
$ cd <WORKDIR>/Filesystem/myir-buildroot
```

Build Buildroot:

Before making Buildroot, we should choose a configuration. For details, please refer to the `Table 3-3-1` .

- MYD-AM335X with qt5：

```
$ make myd_c335x_qt5_defconfig
$ make menuconfig  (optional)
$ make
```

After compiling with the config file `myd_c335x_qt5_defconfig` , all the target images are generated at path `myir-buildroot/output/images` . There is a HMI demo named as `MEasy HMI` inclued, which was designed by `MYIR Electornics Limited` , for more details about `MEasy HMI` , please refer to `MEasy HMI Development Guide` .

Beyond the image files, a cross compiler and a qmake tools are generated at path `myir-buildroot/output/host` after building QT5 applications. These will be described in detail in the subsequent sections.

4. Linux Applications Development

This section focuses on application development based on embedded Linux, the following examples provided by MYIR Tech demonstrate how to take the control of some commonly used peripheral devices through Linux applications. The source code of these examples is located at *04-Linux_Source\Examples* of our release package. Please follow the instructions provided in the readme file to set the environment variables, compile the source code and install the binary files into MYD-AM335X series development board.

```
$ cd <WORKDIR>/Examples/
```

Make sure the following environment variables are right.

```
$ export PATH=$PATH:<WORKDIR>/Toolchain/
gcc-linaro-5.3-2016.02-x86_64_arm-linux-gnueabihf/bin
$ export ARCH=arm
$ export CROSS_COMPILE=arm-linux-gnueabihf-
```

After building Buildroot, a cross compile toolchain has been created at *myir-buildroot/output/host/usr/bin*, it can be used here by setting the environment variables instead of the above.

```
$ export ARCH=arm
$ export CROSS_COMPILE=arm-myir-linux-gnueabihf-
$ export PATH=$PATH:<WORKDIR>/Filesystem/myir-buildroot/output/host/usr/bin
$ export PATH=$PATH:<WORKDIR>/Filesystem/myir-buildroot/output/host/usr/sbin
```

Customers could build examples all in one step:

```
$ cd <WORKDIR>/Examples/
$ make OPTION=MYD-AM335X-SERIES
```

Or build the examples respectively:

```
$ cd <WORKDIR>/Examples/<APP_DIR>
```

```
$ make
```

If the binary files have no permission to run, please assign the running permission to them by chmod：

```
# chmod +x *
```

4.1 LCD

This example demonstrates the usage of Linux API for Linux framebuffer, users can use these API to paint points, lines and areas on LCD frame buffer. At the end of this section, demonstrate drawing a picture on LCD framebuffer with `fbv` application built by Buildroot.

Hardware Preparation:

- Hardware debugging environment to see chapter second.

Board Type	MYD-AM335X	MYD-AM335X-Y	MYD-AM335X-J
LCD interface	(MY-TFT070CV2) 7 inch capacitive screen connection J8	(MY-TFT070CV2) 7 inch capacitive screen connection J7	(MY-TFT070CV2) 7 inch capacitive screen connection J8

> **Note: MY-TFT070CV2 information please click LCD Screen.**

Software Preparation:

- Linux Kernel 4.1.18
- framebuffer_test application
- fbv application built by Buildroot

Test Steps:

- Copy cross compiled `<WORKDIR>/Examples/framebuffer/framebuffer_test` to `/usr/bin` directory of the MYD-AM335X series development board, run `framebuffer_test` application as below:

```
# chmod 777 /usr/bin/framebuffer_test
# framebuffer_test -h
Usage: framebuffer_test [options]

Version 1.0
Available options:
-d | --device name   framebuffer device name, default: /dev/fb0
-h | --help          Print this message

# framebuffer_test -d /dev/fb0
```

```
xres:800 >>> yres:480 >>> bpp:32>>>
```

During `framebuffer_test` running, serval colors of background are painted on LCD one by one, and then colorful points, lines, areas are painted.

- Restart the development board,copy a BMP file with 24BPP and resolution of 800*480 to `/media` of the MYD-AM335X series development board, display the picture on LCD by fbv application:

```
# fbv
Usage: fbv [options] image1 image2 image3 ...

Available options:
 --help        | -h : Show this help
 --alpha       | -a : Use the alpha channel (if applicable)
 --dontclear   | -c : Do not clear the screen before and after displaying the image
 --donthide    | -u : Do not hide the cursor before and after displaying the image
 --noinfo      | -i : Supress image information
 --stretch     | -f : Strech (using a simple resizing routine) the image to fit onto
                                          screen if necessary
 --colorstretch| -k : Strech (using a 'color average' resizing routine) the image to
                                            fit onto screen if necessary
 --enlarge     | -e : Enlarge the image to fit the whole screen if necessary
 --ignore-aspect| -r : Ignore the image aspect while resizing
 --delay <d>   | -s <delay> : Slideshow, 'delay' is the slideshow delay in tenths of s
econds.

Keys:
 r            : Redraw the image
 a, d, w, x   : Pan the image
 f            : Toggle resizing on/off
 k            : Toggle resizing quality
 e            : Toggle enlarging on/off
 i            : Toggle respecting the image aspect on/off
 n            : Rotate the image 90 degrees left
 m            : Rotate the image 90 degrees right
 p            : Disable all transformations
Copyright (C) 2000 - 2004 Mateusz Golicz, Tomasz Sterna.
Error: Required argument missing.

# fbv /meida/800-480.bmp
fbv - The Framebuffer Viewer
/media/800-480.bmp
800 x 480
```

After complete, the picture displays just right for the LCD.

4.2 Touch Panel

This example demonstrates how to test touch screen by ts_calibrate application built with Buildroot.

Hardware preparation:

- Hardware debugging environment to see chapter second.

Board Type	MYD-AM335X	MYD-AM335X-Y	MYD-AM335X-J
Touch screen interface	(MY-TFT070RV2) 7 inch resistive screen connection J8	(MY-TFT070CV2) 7 inch capacitive screen/(MY-TFT070RV2) 7 inch resistive screen connection J7	(MY-TFT070CV27) inch capacitive screen/(MY-TFT070RV2) 7 inch resistive screen connection J8

Software Preparation:

- Linux Kernel 4.1.18
- TS_CALIBRATE application

Test Steps:

- Connect MY-TFT070CV2 module to MYD-AM335X series development board, power on the board and view the device node in /dev/input directory.

```
# ls /dev/input
by-path  event1   event3   mice     mouse1
event0   event2   event4   mouse0
# cat /sys/class/input/event0/device/name
beeper
# cat /sys/class/input/event1/device/name
tps65217_pwrbutton
# cat /sys/class/input/event2/device/name
ft5x06_ts
# cat /sys/class/input/event3/device/name
ti-tsc
```

The result above shows the resistive touch screen is corresponding to `/dev/input/event3`; The capacitive touch screen is corresponding to `/dev/input/event2, so test capactive touch screen as below:

```
# export TSLIB_TSDEVICE=/dev/input/event2
# ts_calibrate
xres = 800, yres = 480
Took 4 samples...
Top left : X =   57 Y =   56
Took 2 samples...
Top right : X =  743 Y =   64
Took 4 samples...
Bot right : X =  742 Y =  438
Took 4 samples...
Bot left : X =   61 Y =  443
Took 4 samples...
Center : X =  398 Y =  246
-8.802551 1.024123 -0.004216
-8.078796 -0.002245 0.998305
Calibration constants: -576884 67116 -276 -529452 -147 65424 65536
#
```

- Power off the MYD-AM335X series development board，connect MY-TFT070RV2 module to MYD-AM335X series development board, power on the board and view the device node in /dev/input directory.

```
# ls /dev/input
by-path  event0   event1   event2   event3   mice     mouse0
# cat /sys/class/input/event0/device/name
beeper
# cat /sys/class/input/event1/device/name
ti-tsc
# cat /sys/class/input/event2/device/name
tps65217_pwrbutton
# cat /sys/class/input/event3/device/name
volume_keys@0
#
```

The result above shows the resistive touch screen is corresponding to `/dev/input/event1`, so test resistive touch screen as below:

```
# export TSLIB_TSDEVICE=/dev/input/event1
# ts_calibrate
```

```
xres = 800, yres = 480
Took 32 samples...
Top left : X =  299 Y =  619
Took 21 samples...
Top right : X = 3689 Y =  659
Took 29 samples...
Bot right : X = 3732 Y = 3463
Took 27 samples...
Bot left : X =  280 Y = 3423
Took 18 samples...
Center : X = 2009 Y = 2072
-7.786255 0.204611 -0.000881
-34.248169 -0.001587 0.135514
Calibration constants: -510280 13409 -57 -2244488 -103 8881 65536
```

4.3 RTC

This example demonstrates how to use Linux API to read and write real time on RTC，please refer to the source code for detail. Users can also test the RTC with `date` and `hwclock` command built with Buildroot.

Hardware Preparation:

- Hardware debugging environment to see chapter second.
- CR1220 3V button cell

Board Type	MYD-AM335X	MYD-AM335X-Y	MYD-AM335X-J
CR1220 button cell	J1	J2	J23

Software Preparation:

- Linux Kernel 4.1.18
- date, hwclock command
- rtc_test application

Test Steps:

- Copy cross compiled `<WORKDIR>/Examples/rtc/rtc_test` to `/usr/bin` directory of the MYD-AM335X seriesdevelopment board, run `rtc_test` application as below:

```
# chmod 777 /usr/bin/rtc_test
# rtc_test -h
Usage: rtc_test [options]

Version 1.0
Options:
-d | --device name   rtc device name, default: /dev/rtc0
-w | --write  time     time string with format MMDDhhmm[CCYY][.ss]. such as: 111817582
016.18
-h | --help          Print this message

# rtc_test -d /dev/rtc -w 111817582016.18
date/time is updated to:  18-11-2016, 17:58:18.
```

- Power off the development board, wait for a while, power on again and read the rtc time by `rtc_test` as below:

```
# rtc_test -d /dev/rtc

  Current RTC date/time is 18-11-2016, 18:04:32.
```

- Users can also use date and hwclock command to test RTC as below:

```
# date  081518002016.30                          -- Set the system time to August 15
, 2016 18:00:30
Mon Aug 15 18:00:30 UTC 2016
# date
Mon Aug 15 18:00:38 UTC 2016
# hwclock -w /dev/rtc                           -- Write the system time to rtc
```

- Power off the development board, wait for a while, power on again and read the rtc time by hwclock as below:

```
# hwclock -r /dev/rtc
Mon Aug 15 18:11:08 2016  0.000000 seconds
```

4.4 RS485

This example demonstrates the MYD-AM335X series of the same type of two development board how to use the Linux API configuration development board on the RS485 and send and receive data,please refer to the source code for detail.

Hardware Preparation:

- Hardware debugging environment to see chapter second.

Board Type	MYD-AM335X	MYD-AM335X-Y	MYD-AM335X-J
RS485 interface	U16 4,5 Pin were connected to the same type of development board 4,5 Pin	CON2 5,4 Pin were connected to the same type of development board 5,4 Pin	JP2 and JP3 shorted，JP5 and JP7 shorted，J18 1，2，4，5 Pin were connected to the same type of development board 1，2，4，5 Pin

Software Preparation:

- Linux Kernel 4.1.18
- tty_test application

Board Type	MYD-AM335X	MYD-AM335X-Y	MYD-AM335X-J
Device node	ttyO1	ttyO2	ttyO2 ttyO3

Test Steps:

- Copy cross compiled `<WORKDIR>/Examples/tty/tty_test` to `/usr/bin` directory of the MYD-AM335X series development board, run `tty_test` application as below:

```
# tty_test -h
Usage: tty_test [options]
Version 1.0
Options:
-d | --device name   tty device name, default: /dev/tty0
-m | --mode  mode   operate mode. 0: RS232, 1: RS485  default mode: 0
-f | --flow                    flow control
-b | --baudrate baudrate  set baudrate, default baudrate: 115200
-l | --loop              operate circularly
```

```
-w | --write  frame     frame string. such as: 0123456789
-h | --help          Print this message
```

- One board is used as sender, the other is used as receiver, they communicate with tty_test application as below:

```
# tty_test -d /dev/ttyO1 -b 9600 -w "123456789" -m 1 -l
SEND:123456789
SEND:123456789
SEND:123456789
```

- Execute the following command at other board to receive data as below:

```
# tty_test -d /dev/ttyO2 -b 9600 -m 1  -l
RECV:1, total:1
RECV:2, total:1
RECV:3, total:1
RECV:4, total:1
RECV:5, total:1
RECV:6, total:1
RECV:7, total:1
RECV:8, total:1
RECV:9, total:1
RECV:1, total:1
RECV:2, total:1
RECV:3, total:1
RECV:4, total:1
RECV:5, total:1
RECV:6, total:1
RECV:7, total:1
RECV:8, total:1
RECV:9, total:1
```

- Exchange roles of the two boards, the result is the same.

4.5 CAN Bus

This example demonstrates the MYD-AM335X series of the same type of two development board how to use Linux APIs to send and receive data from CAN bus, please refer to the source code for detail.

Hardware Preparation:

- Hardware debugging environment to see chapter second.

Board Type	MYD-AM335X	MYD-AM335X-Y	MYD-AM335X-J
CAN interface	U16 7,8 Pin were connected to the same type of development board 7,8 Pin	CON2 1,2 Pin were connected to the same type of development board 1,2 Pin	Disconnect jumper cap JP7,Shorted JP4, J17 1,2 Pin were connected to the same type of development board 1,2 Pin

Software Preparation:

- Linux Kernel 4.1.18
- can_test application
- ip link applicatoin

Board Type	MYD-AM335X	MYD-AM335X-Y	MYD-AM335X-J
Operation node	can1	can1	can0

Test Steps:

- Copy cross compiled `<WORKDIR>/Examples/can/can_test` to `/usr/bin` directory of the MYD-AM335x series development board, run `can_test` application as below:

```
# can_test --help
Usage: can_test [options]

Version 1.0
Options:
-d | --device name   can device name: can0
-b | --baudrate baudrate   set baudrate, default baudrate:50000
-l | --loop              operate circularly, default not operate circularly!
-w | --write  frame     frame string with format ID#MESSAGE. such as: 123#112233445566
```

```
-h | --help                Print this message
```

```
# ip link set can1 down
# ip link set can1 type can bitrate 50000 triple-sampling on
# ip link set can1 up
```

- The previous processes are no need to be executed manually. During running `can_test`, it will be set automatically. One board is used as sender, the other is used as receiver, they communicate with can_test application as below:

```
# chmod 777 /usr/bin/can_test
# can_test -d can0 -w 123#112233445566
[ 5783.823623] c_can_platform 481cc000.can can0: setting BTR=1c1d BRPE=0000
[ 5786.888723] can: controller area network core (rev 20120528 abi 9)
[ 5786.895565] NET: Registered protocol family 29
[ 5786.952090] can: raw protocol (rev 20120528)
====== write  frame: ======
 frame_id = 0x123
 frame_len = 6
 frame_data = 0x11 0x22 0x33 0x44 0x55 0x66
==========================
```

- Execute the following command at other board to receive data as below:

```
# chmod 777 /usr/bin/can_test
# can_test -d can1 -l
[ 5888.821956] c_can_platform 481d0000.can can1: setting BTR=1c1d BRPE=0000
[ 5891.884726] can: controller area network core (rev 20120528 abi 9)
[ 5891.898711] NET: Registered protocol family 29
[ 5891.952878] can: raw protocol (rev 20120528)
can1 0x123 [6] 0x11 0x22 0x33 0x44 0x55 0x66
```

- -l option is used for operating circularly.

> Note: In case of the following error, please modify the value of "tx_queue_len" as below:

```
# can_test -d can0 -w 123#112233445566
  can raw socket write: No buffer space available

# echo 1000 > /sys/class/net/can0/tx_queue_len
```

- Exchange roles of the two boards, the result is the same.

4.6 Ethernet

This example demonstrates how to use Linux APIs to send and receive data from Network port, please refer to the source code for detail.

Hardware Preparation:

- Hardware debugging environment to see chapter second.

Board Type	MYD-AM335X	MYD-AM335X-Y	MYD-AM335X-J
Network port interface	Cable connection J5	Cable connection J5	Cable connection J5

Software Preparation:

- Linux Kernel 4.1.18
- server application
- client applicatoin

Test Steps:

- Copy cross compiled `<WORKDIR>/Examples/network/client` to `/usr/bin` directory of the MYD-AM335x series development board,Copy cross compiled `<WORKDIR>/Examples/network/server` to ubuntu 12.04.Assume that the ubuntu12.04 ip is 192.168.30.114.

Configure the development board ip：

```
ifconfig eth0 192.168.30.122 up
```

Configure the network, the development board eth0 connected to the PC with a network cable,to the virtual machine as the server, the development board for the client, first in the virtual machine to execute the following command:

```
$./server 192.168.30.122
```

And then in the development board to execute the following command to see the information sent:

```
# ./client 192.168.30.114
send messages: 1234567890 to 192.168.30.114
send messages: 1234567890 to 192.168.30.114
send messages: 1234567890 to 192.168.30.114
send messages: 1234567890 to 192.168.30.114
send messages: 1234567890 to 192.168.30.114
send messages: 1234567890 to 192.168.30.114
```

At the same time you can see the virtual machine to receive the data sent by the development board:

```
$ ./server 192.168.30.122
received messages: 1234567890@ from 192.168.30.122
received messages: 1234567890@ from 192.168.30.122
received messages: 1234567890@ from 192.168.30.122
received messages: 1234567890@ from 192.168.30.122
received messages: 1234567890@ from 192.168.30.122
received messages: 1234567890@ from 192.168.30.122
```

4.7 NAND Flash

This example demonstrates how to use the Linux API to erase, write, and read NAND Flash on the development board,please refer to the source code for detail.

Hardware Preparation:

- Hardware debugging environment to see chapter second.
- MYD-AM335X series development board.

Software Preparation:

- Linux Kernel 4.1.18
- mtd_test application

Test Steps:

- Copy cross compiled `<WORKDIR>/Examples/mtd/mtd_test` to `/usr/bin` directory of the MYD-AM335x series development board,Executing the following command to nandflash erase, write and read data:

```
# ./mtd_test /dev/mtd7

MTD Type: 4
MTD total size: 131072 bytes
MTD erase size: 131072 bytes
MTD write size: 2048 bytes

erase the last block at 0

erase done!

writing 16 bytes data to flash...
0x00 0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08 0x09 0x0a 0x0b 0x0c 0x0d 0x0e 0x0f
write done!

reading data from flash...
0x00 0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08 0x09 0x0a 0x0b 0x0c 0x0d 0x0e 0x0f
read done!
```

4.8 KeyPad

This example demonstrates how to read the keypad event information by Linux user APIs, please refer to the source code for detail.

Hardware Preparation:

- Hardware debugging environment to see chapter second.
- MYD-AM335x series development board

Software Preparation:

- Linux Kernel 4.1.18
- hexdump command
- keypad_test application

Test Steps:

- Copy cross compiled `<WORKDIR>/Examples/keypad/keypad_test` to `/usr/bin` directory of the MYD-AM335x series development board, run `keypad_test` application as below:

```
$ chmod 777 /usr/bin/keypad_test
$ keypad_test -h
Usage: keypad_test [options]

Version 1.0
Options:
-d | --device name   keypad device name, default: /dev/input/event0
-h | --help          Print this message
```

- View the device nodes of MYD-AM335X keypad, the following information shows `K2` and `K3` keypads are corresponding to `/dev/input/event1`。

```
# ls /dev/input/
by-path/ event0   event1   mice     mouse0

# cat /sys/class/input/event0/device/name
ti-tsc

# cat /sys/class/input/event1/device/name
```

```
volume_keys@0
```

- Test MYD-AM335X `K2` and `K3` keypads as below:

```
# keypad_test -d /dev/input/event1
Event: Code = 115, Type = 1, Value=1                -- press K2
Event: Code = 0, Type = 0, Value=0
Event: Code = 115, Type = 1, Value=0                -- release K2
Event: Code = 0, Type = 0, Value=0
Event: Code = 114, Type = 1, Value=1                -- press K3
Event: Code = 0, Type = 0, Value=0
Event: Code = 114, Type = 1, Value=2
Event: Code = 0, Type = 0, Value=1
Event: Code = 114, Type = 1, Value=0                -- release K3
Event: Code = 0, Type = 0, Value=0
```

- View the device nodes of MYD-AM335X-Y keypad, the following information shows `K3` and `K4` keypads are corresponding to `/dev/input/event3`, `K5` keypad is corresponding to `/dev/input/event2`.

```
# ls /dev/input/
by-path  event0   event1   event2   event3   mice     mouse0

# cat /sys/class/input/event0/device/name
beeper
# cat /sys/class/input/event1/device/name
ti-tsc
# cat /sys/class/input/event2/device/name
tps65217_pwrbutton
# cat /sys/class/input/event3/device/name
volume_keys@0
```

- Test MYD-AM335X-Y `K3` and `K4` keypads as below:

```
# keypad_test -d /dev/input/event3
Event: Code = 115, Type = 1, Value=1                -- press K3
Event: Code = 0, Type = 0, Value=0
Event: Code = 115, Type = 1, Value=0                -- release K3
Event: Code = 0, Type = 0, Value=0
Event: Code = 114, Type = 1, Value=1                -- press K4
Event: Code = 0, Type = 0, Value=0
Event: Code = 114, Type = 1, Value=0                -- release K4
Event: Code = 0, Type = 0, Value=0
```

- Test MYD-AM335X-Y `K5` keypads as below:

```
# keypad_test -d /dev/input/event2
Event: Code = 116, Type = 1, Value=1                -- press K5
Event: Code = 0, Type = 0, Value=0
Event: Code = 116, Type = 1, Value=0                -- release K5
Event: Code = 0, Type = 0, Value=0
```

- View the device nodes of MYD-AM335X-J keypad, the following information shows `K3` and `K4` keypads are corresponding to `/dev/input/event3`, `K2` keypad is corresponding to `/dev/input/event1`.

```
# ls /dev/input/
by-path  event0   event1   event2   event3   mice     mouse0   mouse1

# cat /sys/class/input/event0/device/name
ti-tsc
# cat /sys/class/input/event1/device/name
tps65217_pwrbutton
# cat /sys/class/input/event2/device/name
ft5x06_ts
# cat /sys/class/input/event3/device/name
volume_keys@0
```

- Test MYD-AM335X-J `K3` and `K4` keypads as below:

```
# keypad_test -d /dev/input/event3
Event: Code = 115, Type = 1, Value=1            -- press K3
Event: Code = 0, Type = 0, Value=0
Event: Code = 115, Type = 1, Value=0            -- release K3
Event: Code = 0, Type = 0, Value=0
Event: Code = 114, Type = 1, Value=1            -- press K4
Event: Code = 0, Type = 0, Value=0
Event: Code = 114, Type = 1, Value=0            -- release K4
Event: Code = 0, Type = 0, Value=0
```

- Test MYD-AM335X-J `K2` keypads as below:

```
# keypad_test -d /dev/input/event1
Event: Code = 116, Type = 1, Value=1            -- press K2
Event: Code = 0, Type = 0, Value=0
Event: Code = 116, Type = 1, Value=0            -- release K2
Event: Code = 0, Type = 0, Value=0
```

Each keypad has a event code as show above. the code should be consistent with the value in device tree source.Users can use `hexdump` command to test keypads as below，The following to MYD-AM335X, for example, other types of KeyPad test similar.

```
# hexdump /dev/input/event1
0000000 022b 0000 2a83 0005 0001 0073 0001 0000
0000010 022b 0000 2a83 0005 0000 0000 0000 0000
0000020 022b 0000 78b5 0007 0001 0073 0000 0000
0000030 022b 0000 78b5 0007 0000 0000 0000 0000

0000040 0231 0000 8c69 0001 0001 0072 0001 0000
0000050 0231 0000 8c69 0001 0000 0000 0000 0000
0000060 0231 0000 fc11 0003 0001 0072 0000 0000
0000070 0231 0000 fc11 0003 0000 0000 0000 0000
```

4.9 GPIO-LED

On a embbeded Linux system, the LEDs are commonly controlled by sysfs interface. This example demonstrates how to control the LEDs by sysfs with `echo` command or `led_test` application.

Hardware Preparation:

- Hardware debugging environment to see chapter second.
- MYD-AM335x series development board

Software Preparation:

- Linux Kernel 4.1.18
- echo, led_test application

Test Steps:

- Copy cross compiled `<WORKDIR>/Examples/led/led_test` to `/usr/bin` directory of the MYD-AM335x series development board.
- View the device node of MYD-AM335X LED devices as below:

```
# ls /sys/class/leds
myc:green:user1  myd:green:user2  myd:green:user3
```

- Control the MYD-AM335X LED by `echo` command as below:

```
D3:
#echo "1" > /sys/class/leds/myc\:green\:user1/brightness
#echo "0" > /sys/class/leds/myc\:green\:user1/brightness
D39:
#echo "1" > /sys/class/leds/myd\:green\:user2/brightness
#echo "0" > /sys/class/leds/myd\:green\:user2/brightness
D40:
#echo "1" > /sys/class/leds/myd\:green\:user3/brightness
#echo "0" > /sys/class/leds/myd\:green\:user3/brightness
```

- View the device node of MYD-AM335X-Y LED devices as below:

```
# ls /sys/class/leds
```

```
myc:green:user1  myd:green:user2
```

- Control the MYD-AM335X-Y LED by `echo` command as below:

```
D3:
#echo "1" > /sys/class/leds/myc\:green\:user1/brightness
#echo "0" > /sys/class/leds/myc\:green\:user1/brightness
D2:
#echo "1" > /sys/class/leds/myd\:green\:user2/brightness
#echo "0" > /sys/class/leds/myd\:green\:user2/brightness
```

- View the device node of MYD-AM335X-J LED devices as below:

```
# ls /sys/class/leds
myc:green:user1  myd:green:user2
```

- Control the MYD-AM335X-J LED by `echo` command as below:

```
D3:
# echo "1" > /sys/class/leds/myc\:green\:user1/brightness
# echo "0" > /sys/class/leds/myc\:green\:user1/brightness
D2:
# echo "1" > /sys/class/leds/myd\:green\:user2/brightness
# echo "0" > /sys/class/leds/myd\:green\:user2/brightness
```

- Control the LEDs by 'led_test' application as below:

```
# led_test -h
Usage: led_test [options]

Version 1.0
Options:
-d | --device name   led name myc:blue:cpu0
-l | --light brightness   led brightness. 0~255 0: off.
-h | --help                Print this message

# led_test -d myc:green:user1 -l 0
Set led myc:green:user1 off, brightness = 0
# led_test -d myc:green:user1 -l 1
Set led myc:green:user1 on, brightness = 1
```

4.10 Audio

This example demonstrates how to use the Linux API to control the input and output of audio,please refer to the source code for detail.

Hardware Preparation:

- Hardware debugging environment to see chapter second.

Board Type	MYD-AM335X	MYD-AM335X-Y	MYD-AM335X-J
Audio interface	Audio input J11，Headphones Connect J10	Audio input J9，Headphones Connect J8	Audio input J10，Headphones Connect J9

Software Preparation:

- Linux Kernel 4.1.18
- audio_test application

Test Steps:

- Copy cross compiled `<WORKDIR>/Examples/audio/audio_test` to `/usr/bin` directory of the MYD-AM335x series development board.Plug the headset into the audio output port, enter the audio input to the audio input port, enter the following command at the Linux terminal:

```
# audio_test
rate set to 21999, expected 22000
Init capture successfully, rate: 21999, period_size: 128
rate set to 21998, expected 21999
Period size: 128 frames, buffer size: 256 bytes
```

4.11 USB Host

This example demonstrates how to use USB host to mount mass stroage device and verify the driver of USB host,to achieve the function of reading and writing USB flash disk.

Hardware Preparation:

- Hardware debugging environment to see chapter second.
- USB flash disk

Board Type	MYD-AM335X	MYD-AM335X-Y	MYD-AM335X-J
USB Host interface	J3 or J4	J19	J27

Software Preparation:

- Linux Kernel 4.1.18
- mount and umount commands

Test Steps:

- Plug the USB disk in the USB host interface of development board, use `mount` or `umount` command to load and unload USB disk. When users plug in the USB disk, Linux kernel dumps the message as below:

```
# [  334.568567] usb 2-1.4: new high-speed USB device number 3 using musb-hdrc
[  334.688922] usb 2-1.4: New USB device found, idVendor=1908, idProduct=0226
[  334.696305] usb 2-1.4: New USB device strings: Mfr=0, Product=0, SerialNumber=0
[  334.743502] usb-storage 2-1.4:1.0: USB Mass Storage device detected
[  334.751563] scsi host0: usb-storage 2-1.4:1.0
[  334.762053] usbcore: registered new interface driver usb-storage
[  334.788766] cpu cpu0: clk_voltdm_notifier_handler: Failed to scale voltage(1100000)
: -22
[  334.797531] cpu cpu0: clk_voltdm_notifier_handler: Failed to scale voltage(950000):
 -22
[  334.806827] cpu cpu0: failed to set clock rate: -16
[  334.812093] cpufreq: __target_index: Failed to change cpu frequency: -16
[  335.759638] scsi 0:0:0:0: Direct-Access     Generic  Mass-Storage     1.11 PQ: 0 AN
SI: 2
[  336.488565] sd 0:0:0:0: [sda] 15605760 512-byte logical blocks: (7.99 GB/7.44 GiB)
[  336.497237] sd 0:0:0:0: [sda] Write Protect is off
[  336.503459] sd 0:0:0:0: [sda] Mode Sense: 03 00 00 00
```

```
[  336.509699] sd 0:0:0:0: [sda] No Caching mode page found
[  336.515366] sd 0:0:0:0: [sda] Assuming drive cache: write through
[  336.531208]  sda: sda1
[  336.552003] sd 0:0:0:0: [sda] Attached SCSI removable disk
```

- It shows USB host works well and USB disk is detected, users can mount it to /mnt directory of the embedded Linux system as below:

```
# mount /dev/sda1 /mnt
[  429.892793] FAT-fs (sda1): Volume was not properly unmounted. Some data may be corr
upt. Please run fsck.
# cd /mnt
# ls
1.bmp                mtd                  rootfs.ubi
MLO                  myd_c335x.dtb        spi
audio                myir-linux-examples  u-boot.img
audio1               network              zImage
#
```

- Plug out the USB disk, Linux kernel dumps the message as below:

```
# [  493.899143] usb 2-1.4: USB disconnect, device number 3
```

4.12 USB Device

This example demonstrates how to use USB device interface and verify the driver of USB client. The MYD-AM335X series development board works as a TF card reader, it is connected to the USB host of PC with a USB mini B to USB A cable.

Hardware Preparation：

- Hardware debugging environment to see chapter second.
- One TF card
- One USB mini B to USB A cable.

Board Type	MYD-AM335X	MYD-AM335X-Y	MYD-AM335X-J
MINI USB interface	J2	J3	J3
TF Card	J17	J12	J19

Software Preparation:

- Linux Kernel 4.1.18
- modprobe command

Test Steps:

- After the MYD-AM335X series development is booted, connect it to the USB host interface of PC with a USB mini B to USB A cable, Insert the TF card to MYD-AM335X series development. Load the mass storage gadget driver as below:

```
# modprobe g_mass_storage stall=0 file=/dev/mmcblk0p1 removable=1
[  687.171803] udc musb-hdrc.0.auto: registering UDC driver [g_mass_storage]
[  687.179455] Mass Storage Function, version: 2009/09/11
[  687.184933] LUN: removable file: (no medium)
[  687.192157] lun0: open backing file: /dev/mmcblk0p1
[  687.197379] LUN: removable file: /dev/mmcblk0p1
[  687.202952] Number of LUNs=1
[  687.206057] g_mass_storage gadget: adding config #1 'Linux File-Backed Storage'/bf2
a45cc
[  687.215260] Number of LUNs=1
[  687.219060] g_mass_storage gadget: I/O thread pid: 185
[  687.224576] g_mass_storage gadget: adding 'Mass Storage Function'/dc896b00 to confi
g 'Linux File-Backed Storage'
```

```
/bf2a45cc
[  687.236802] g_mass_storage gadget: cfg 1/bf2a45cc speeds: high full
[  687.243534] g_mass_storage gadget:   interface 0 = Mass Storage Function/dc896b00
[  687.252005] g_mass_storage gadget: Mass Storage Gadget, version: 2009/09/11
[  687.259466] g_mass_storage gadget: userspace failed to provide iSerialNumber
[  687.266948] g_mass_storage gadget: g_mass_storage ready
[  687.273141] g_mass_storage musb-hdrc.0.auto: usb_gadget_udc_start
```

- After g_mass_storage driver is loaded, a removable disk will be detected on PC. The content of this removable disk is just the same with the TF card.

> Note: Beyond that, users can load different gadget modules to achieve different functions. such as g_ether is used to make a RNDIS network interface.

5. Qt Applications Development

In the previous sections, a build tool for QT5 application `qmake` has been created. It is used for generate Makefiles and other project files for QT5 application, after that, users can build QT5 application with cross compile toolchain. For development of larger QT5 application, IDE tools named as `QtCreator` is commonly used. There is a evaluation edition of QtCreator in our release package at the path *03-Tools/qt-creator-opensource-linux-x86_64-4.1.0.run*.

In the following sections, we will introduce the installation, configuration of QtCreator, and demonstrate how to create a simple QT5 application running on MYD-AM335x series development board.

5.1 Install QtCreator

On Ubuntu 64bit OS, customers can install QtCreator as shown below:

```
$ cd <WORKDIR>
$ cp /media/cdrom/03-Tools/Qt/qt-creator-opensource-linux-x86_64-4.1.0.run .
$ chmod a+x qt-creator-opensource-linux-x86_64-4.1.0.run
$ sudo ./qt-creator-opensource-linux-x86_64-4.1.0.run
```

QtCreator will be installed on Ubuntu OS step by step automatically.

5.2 Config QtCreator

- Config Build&Run Environment:

Open QtCreator, choose `Tools` -> `Options`, then the `Build & Run` dialog popups. `Please choose the` Compilers` tab to set compiler for QtCreator as shown below:

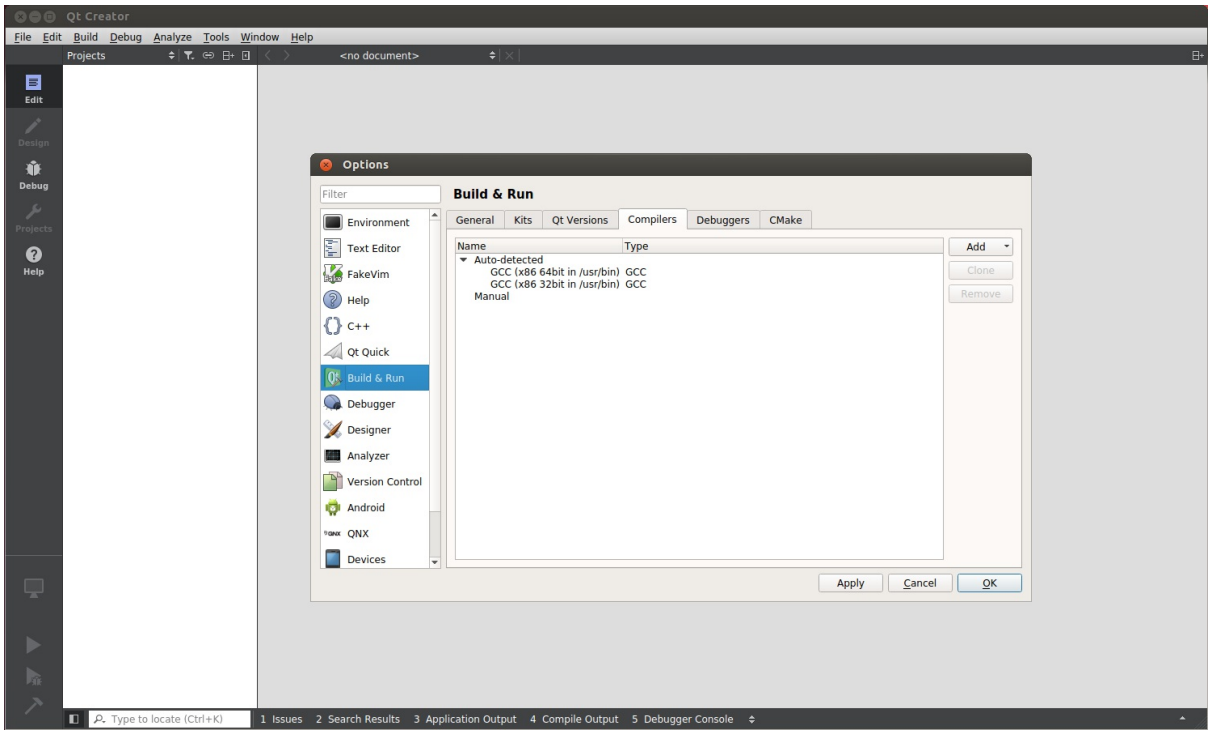

Figure 5-2-1 Settings of Compiler

Press `Add` button at the right side of this dialog, choose `Custom` in the dropdown list, and then set `Name`, `Compiler path`, `Make path` 和 `ABI` as shown below. After complete, press the `Apply` button to save.

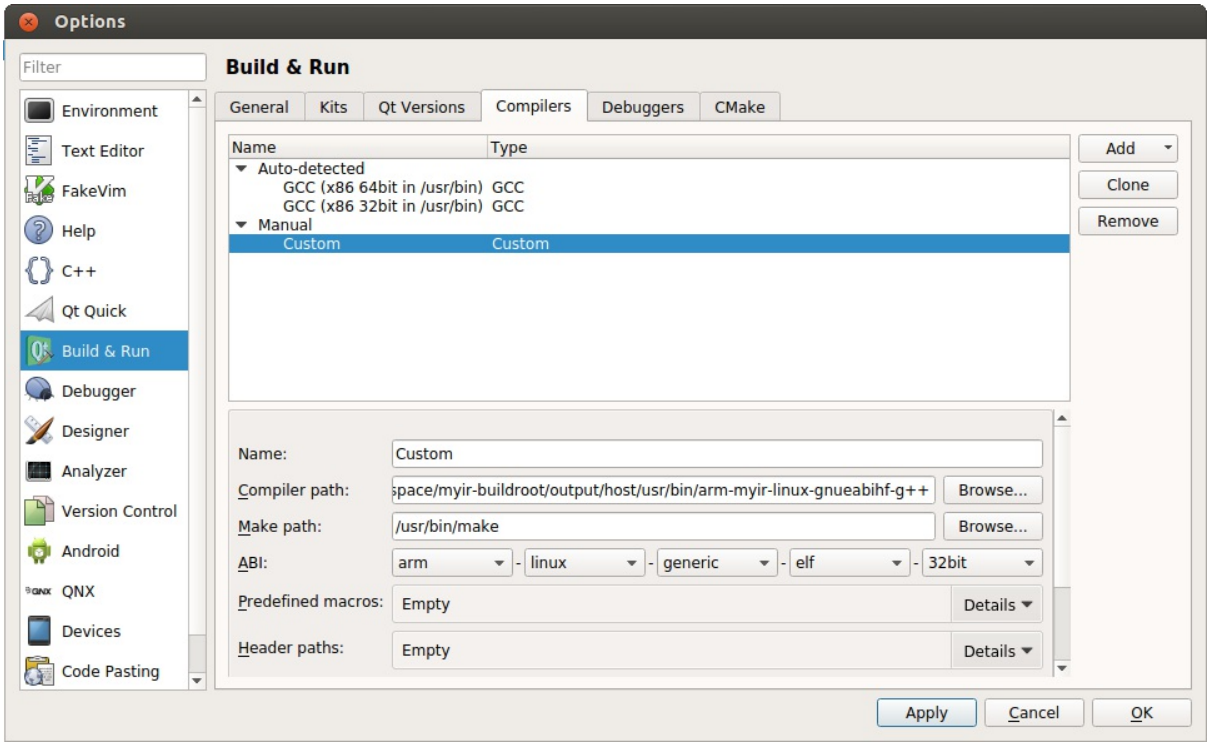

Figure 5-2-2 Add Compiler for QtCreator

At the same dialog, choose `Qt Version` Tab to add qmake, at the right side of this dialog, press `Add` button, then a new dialog popups, please choose the qmake tools described at Chapter 3-4-1. After complete, press `Open` to set qmake, and then `Apply` to save.

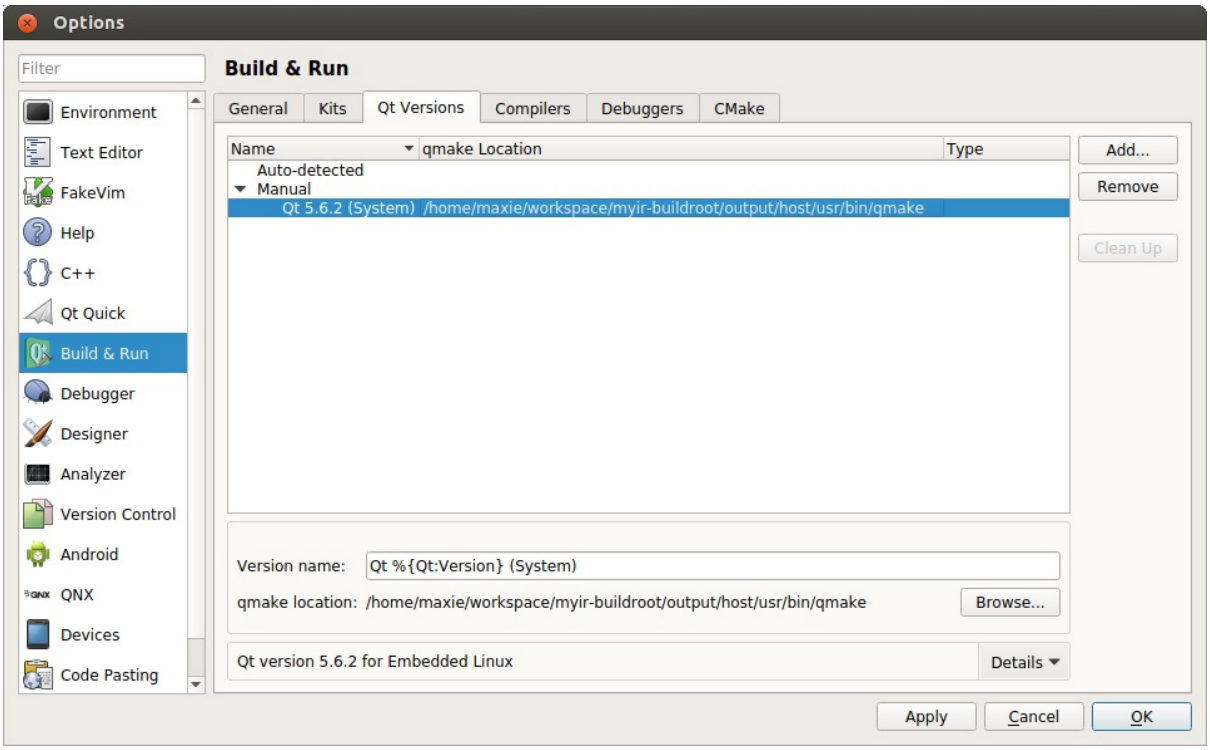

Figure 5-2-3 Choose Qmake

In the `Build&Run` window, continue to choose `Kits` tab, and at the right side of this dialog, press `Add` button, then add settings of running environment for QT5 application. In the `Sysroot` editbox, write the path of cross compile toolchain, in the `Compiler` and `Qt Version` editboxs, write the settings being set before, set `Debugger` to `None`, set `CMake Tools` to default as shown below:

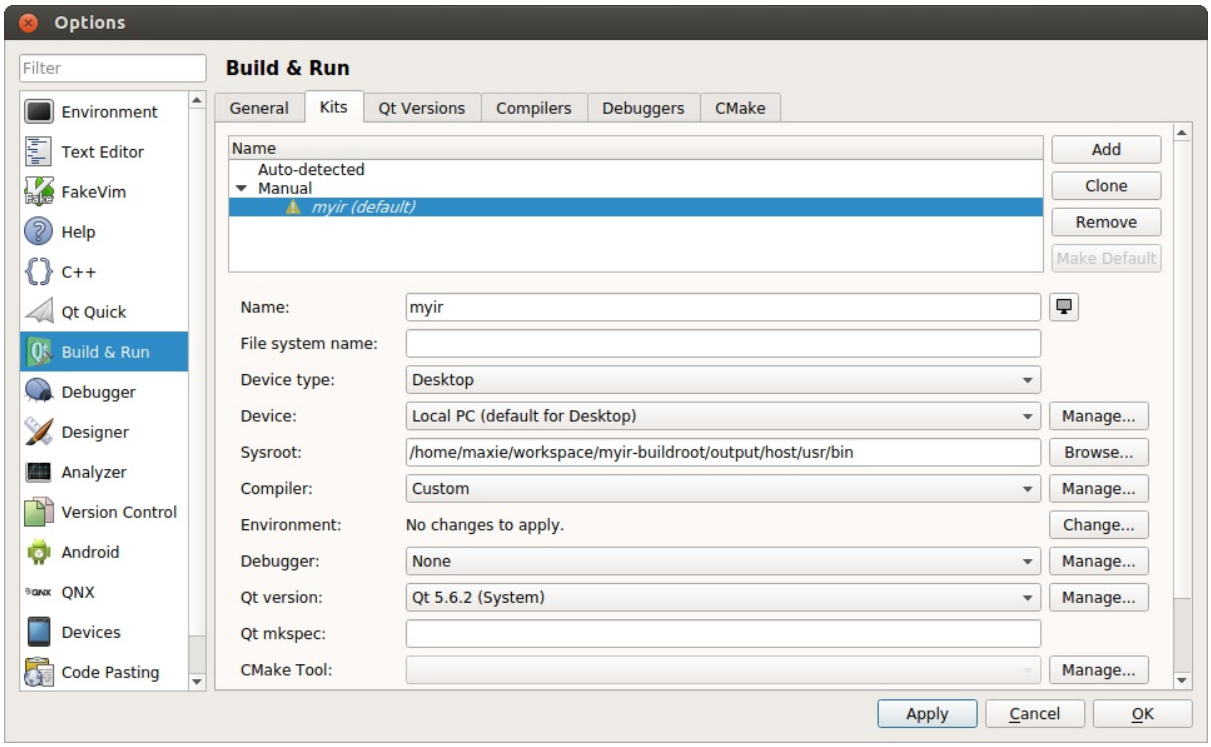

Figure 5-2-4 Add Kits for QtCreator

- Create Helloworld Project:

In the main menu of QtCreate, choose `File` -> `New File or Project`, and in the popup dialog, choose `Application` -> `Qt Widgets Application` as shown below:

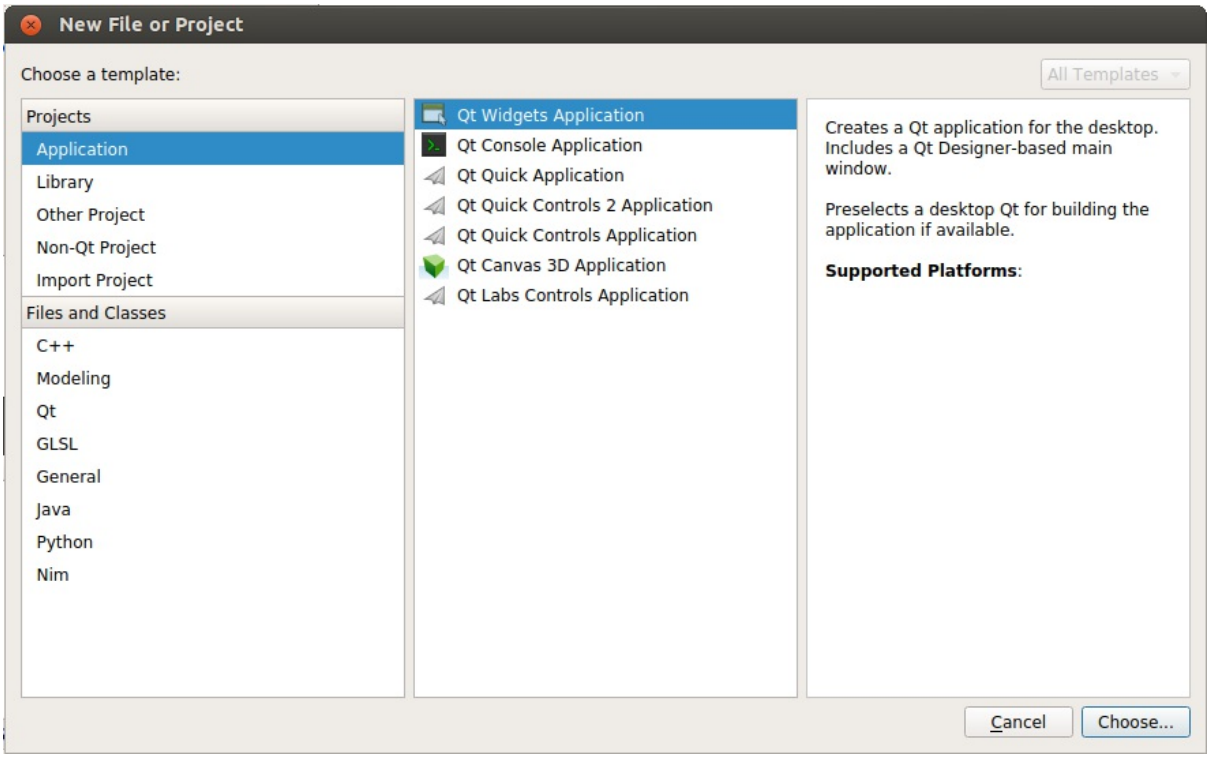

Figure 5-2-5 Create a QT Widgets Application

After pressing `Choose...` button, the `Qt Widgets Application` settings dialog popups, please set the name and path of the project as shown below in `Name` and `Create in` editboxs as shown below:

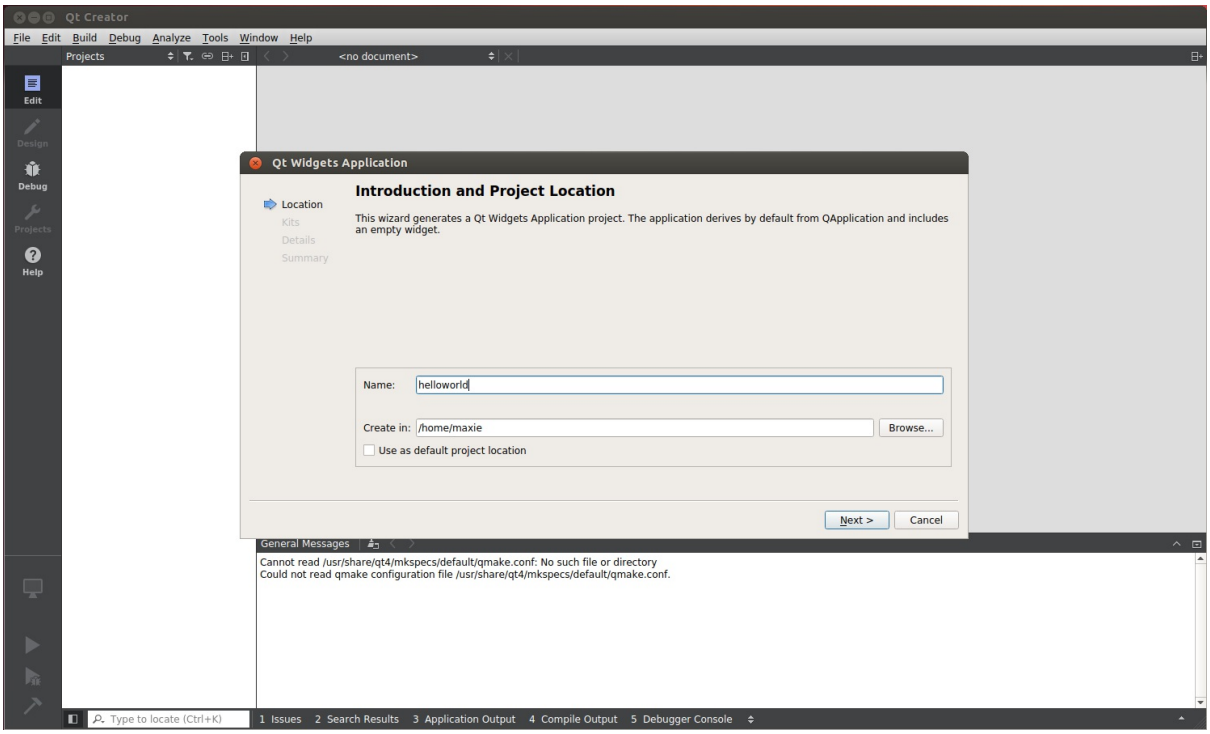

Figure 5-2-6 Set the Name and Path of the Project

Press `Next` button and choose the setting for `Kits` as below:

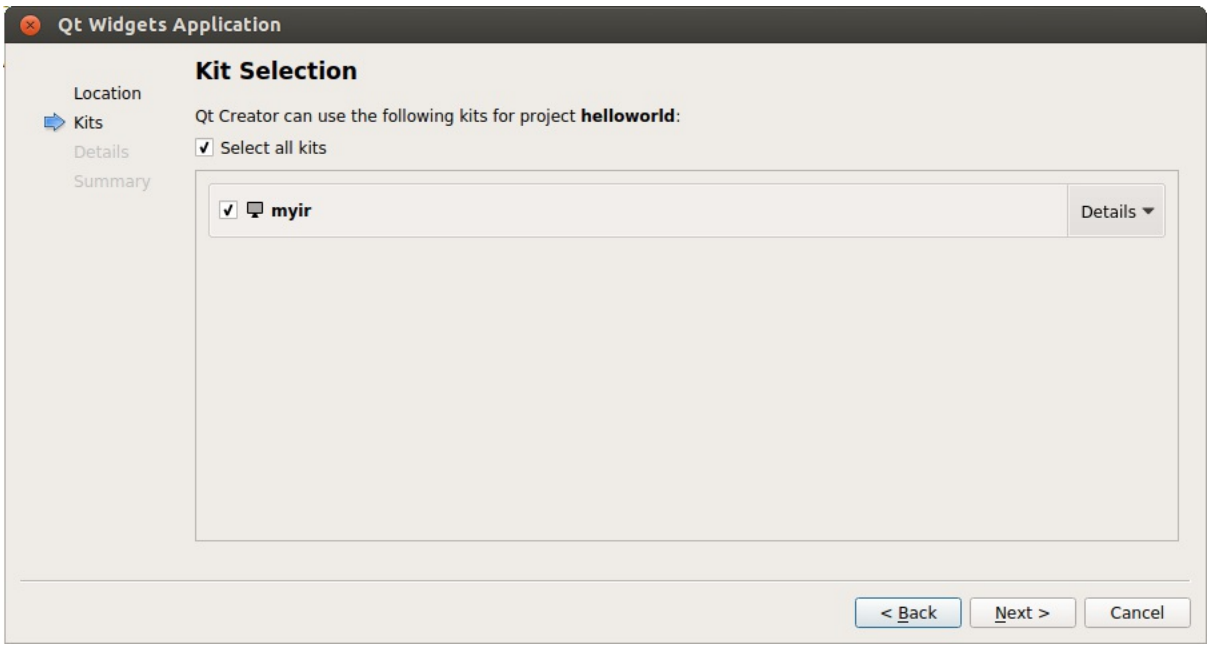

Figure 5-2-7 Set Kits for the Project

Choose the base class of the project as shown below:

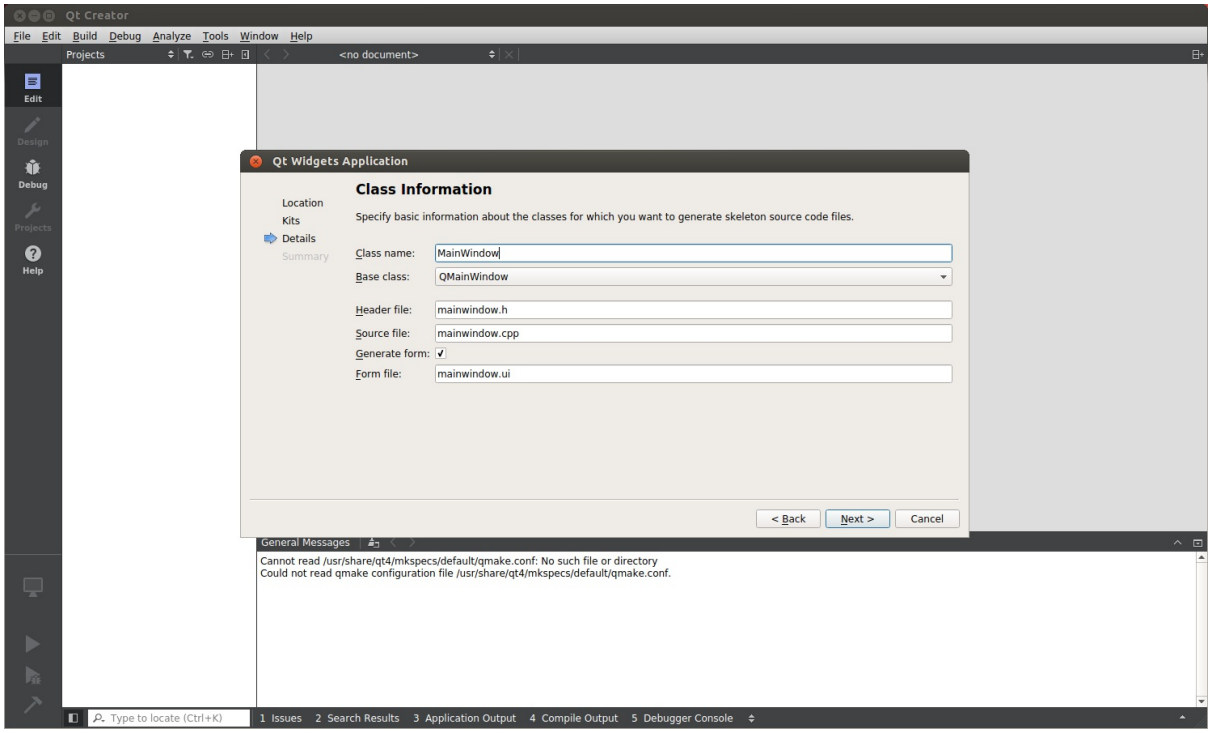

Figure 5-2-8 Choose Base Class

Press `Finish` button to create and save the project.

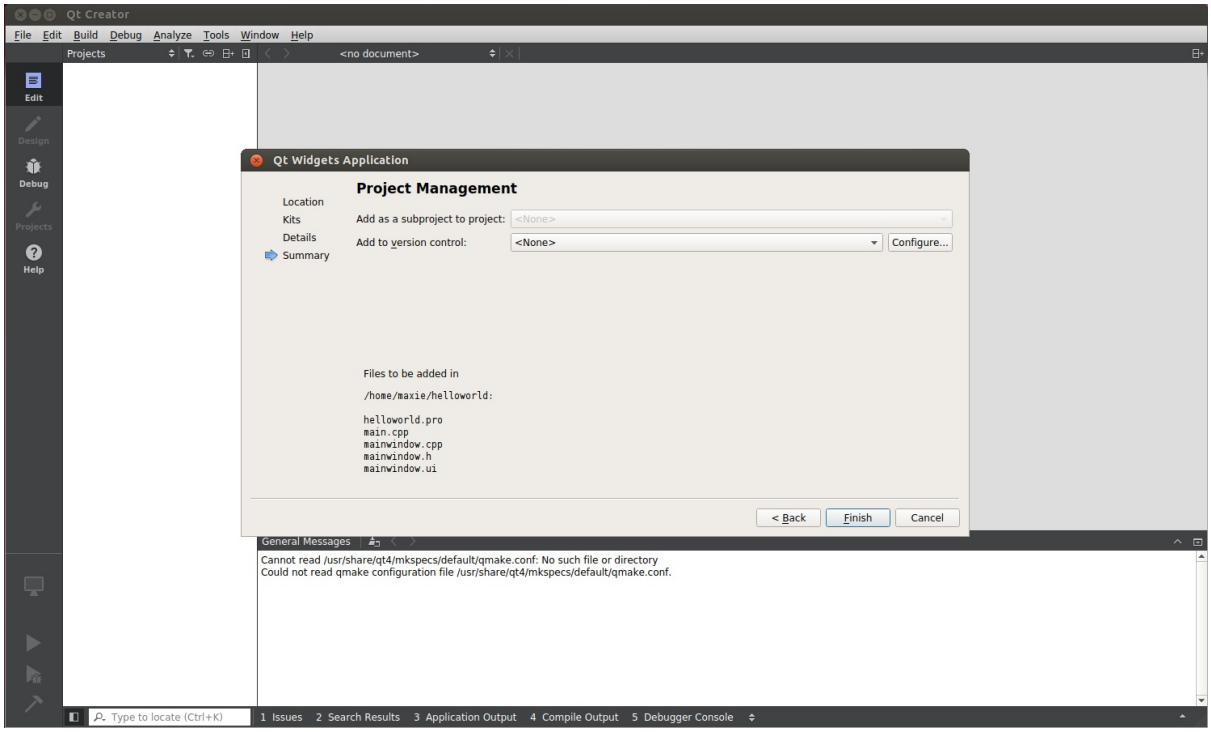

Figure 5-2-9 Finish Creating a New Project

5.3 Build QT Application

In the previous section, a QT5 project named as `helloworld` has been created. It is shown in the project manage of QtCreator as below:

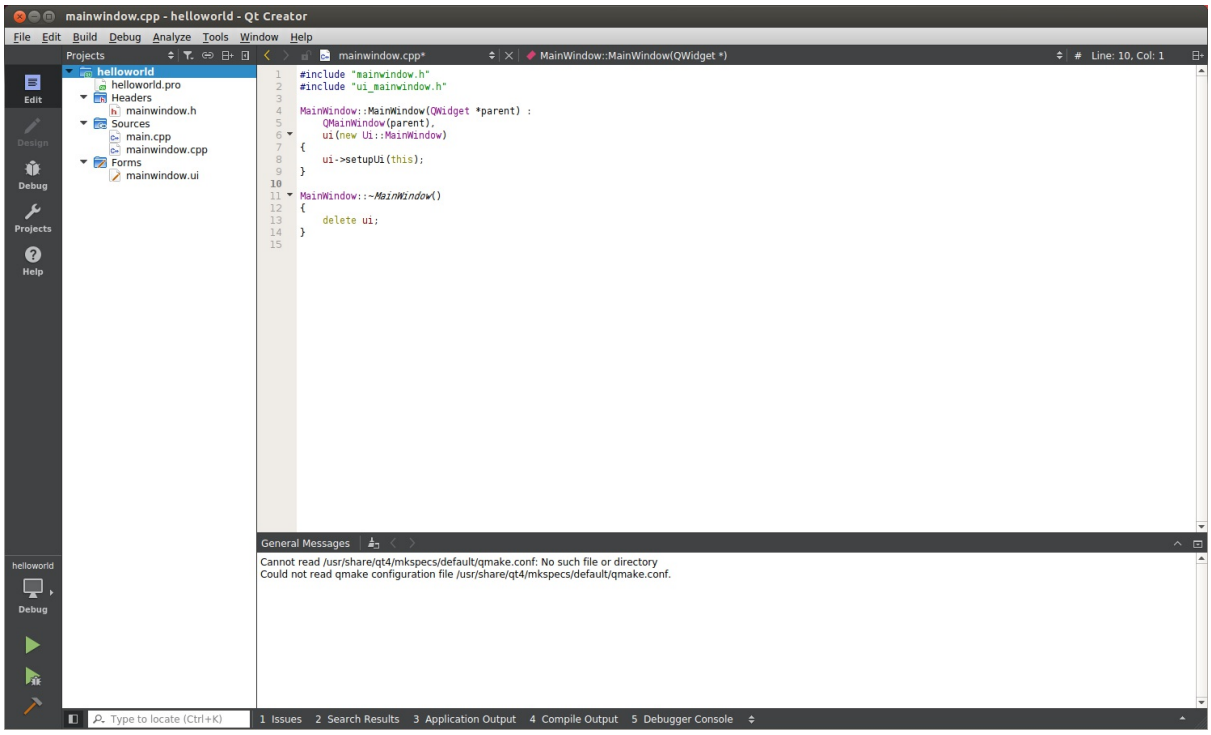

Figure 5-3-1 Project Manager of QtCreator

Double click the `mainwindow.ui` at the left side to open the `Design` view for designing a UI for helloworld project visually. Please drag a `Label` widget to the center of the mainwindow from the widgets list, double click the label and input `Hello, world!".

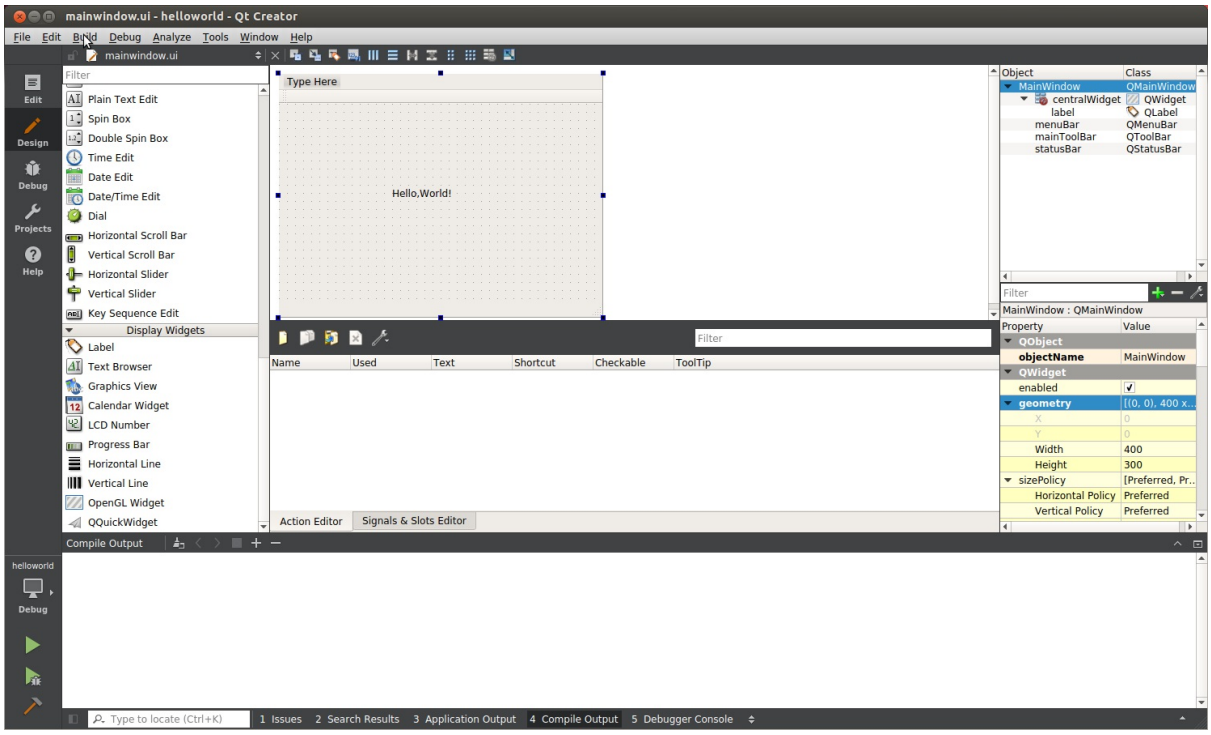

Figure 5-3-2 UI Design View

After complete, choose `Build` -> `Build Project` in the main menu of QtCreator to build the helloworld project. Some log information outputs to the `Compile Output` sub window during compiling, in case of any erros and warning, please fix them and build again.

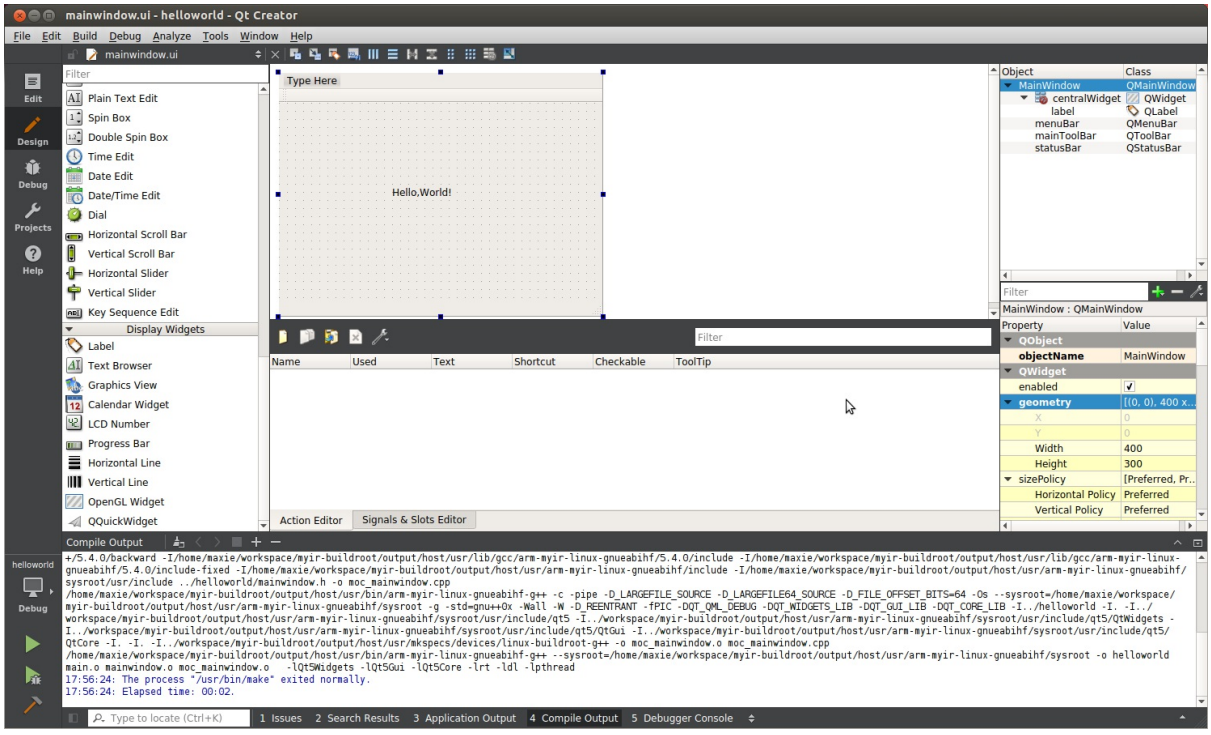

Figure 5-3-3 Build Project

After building, a binary format QT5 application is generated at *~/build-helloworld-myir_dev_kit-Debug/*, please use `file` command to check it and make sure it can work on ARM embedded Linux system as below:

```
file helloworld
helloworld: ELF 32-bit LSB executable, ARM, version 1 (GNU/Linux), dynamically linked
 (uses shared libs), for GNU/Linux 4.1.0, not stripped
```

Finaly, please copy the binary format application `helloworld` to `/usr/bin` directory of MYD-AM335X series development board and execute as below:

```
# helloworld  --platform linuxfb:fb=/dev/fb0
```

A Window with a `Hello,World!` text label displays on the LCD screen as below:

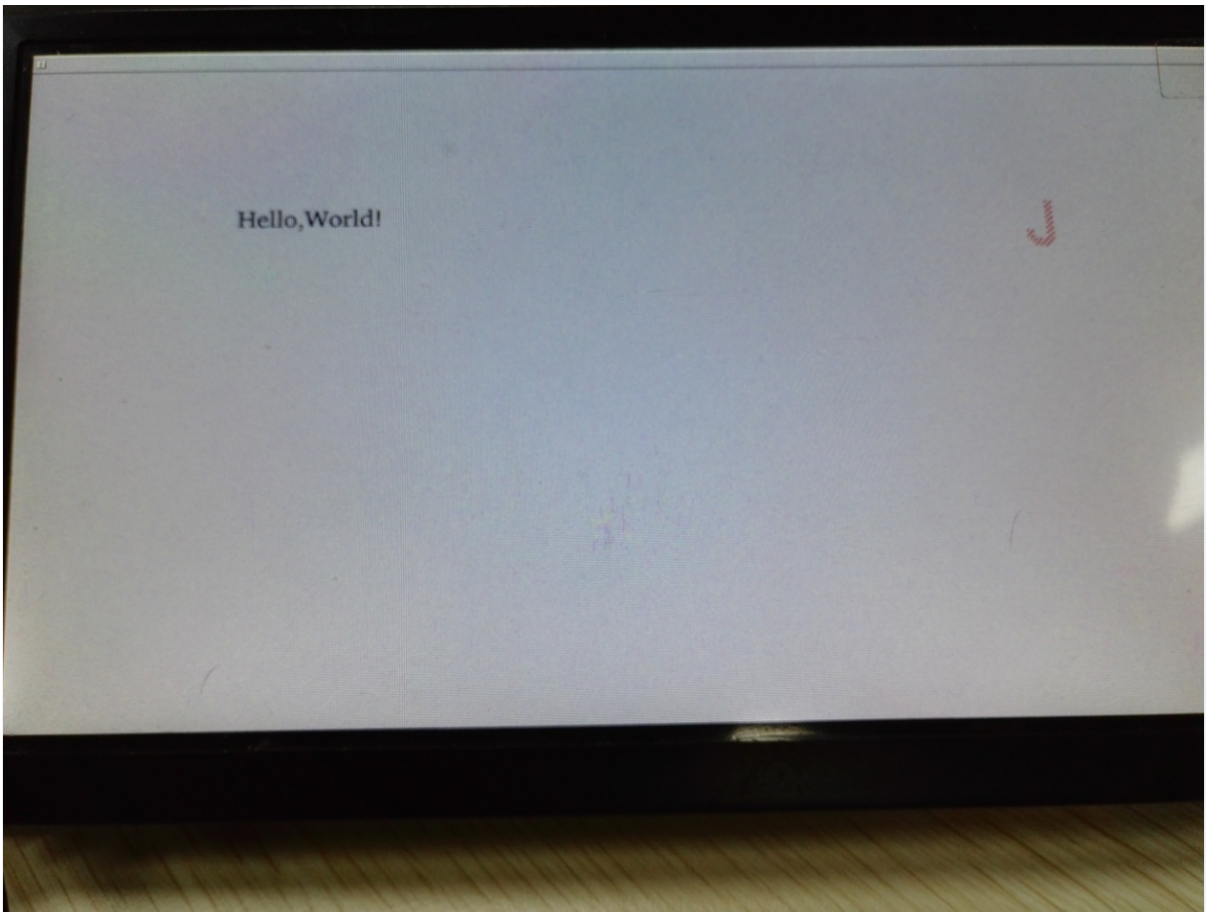

Figure 5-3-4 Execute QT5 Application

6. Update System

This section introduces the Linux system tf card boot, NAND Flash burn,NFS ROOT file system mount. CD image file description：

Table 6-1 Prebuild Image Files Description

File Name	Description
MLO	First stage bootloader（SPL), generated when compiling Buildroot
MLO_nand	First stage bootloader（SPL) for NAND, prebuild by myir from "board/myir/myd_c335x/"
MLO_sd	First stage bootloader（SPL) for TF Card, prebuild by myir from "board/myir/myd_c335x/"
MLO_emmc	First stage bootloader（SPL) for EMMC, prebuild by myir from "board/myir/myd_c335x/"
MLO_usbmsc	First stage bootloader（SPL) for USB Mass Storage, prebuild by myir from "board/myir/myd_c335x/"
u-boot.img	Second stage bootloader, generated when compiling Buildroot
u-boot_nand.img	Second stage bootloader for NAND, prebuild by myir from "board/myir/myd_c335x/"
u-boot_sd.img	Second stage bootloader for TF Card, prebuild by myir from "board/myir/myd_c335x/"
u-boot_emmc.img	Second stage bootloader for EMMC, prebuild by myir from "board/myir/myd_c335x/"
u-boot_usbmsc.img	Second stage bootloader for USB Mass Storage, prebuild by myir from "board/myir/myd_c335x/"
uEnv.txt	Default environment variables for U-boot
uEnv_ramdisk.txt	Environment variables for boot ramdisk images on TF/SD card. Need to be renamed to uEnv.txt
uEnv_sd.txt	Environment variables for boot images on TF/SD card. Need to be renamed to uEnv.txt
uEnv_sd_ramdisk.txt	Environment variables for boot ramdisk images on TF/SD card. Need to be renamed to uEnv.txt

uEnv_usbmsc.txt	Environment variables for boot images on USB Mass Storage. Need to be renamed to uEnv.txt
uEnv_usbmsc_ramdisk.txt	Environment variables for boot ramdisk images on USB Mass Storage. Need to be renamed to uEnv.txt
uEnv_mmc.txt	Environment variables for boot images on EMMC. Need to be renamed to uEnv.txt
zImage	Kernel image
myd_c335x.dtb	Device tree binary for MYD-AM335X with NAND Flash
myd_c335x_emmc.dtb	Device tree binary for MYD-AM335X with EMMC
myd_y335x.dtb	Device tree binary for MYD-AM335X-Y with NAND Flash
myd_y335x_emmc.dtb	Device tree binary for MYD-AM335X-Y with EMMC
myd_j335x.dtb	Device tree binary for MYD-AM335X-J with NAND Flash
myd_j335x_emmc.dtb	Device tree binary for MYD-AM335X-J with EMMC
rootfs.tar.gz	ramdisk filesystem compressed by gzip
rootfs.ubi	UBIFS filesystem image
ramdisk.gz	ramdisk filesystem compressed by gzip
sdcard.img	TF/SD/EMMC disk image

`uEnv*.txt` are text files, they define the environment parameters for U-boot, and then determine the boot process of U-boot. We take the `uEnv_ramdisk.txt` as an example:

`fdtfile` define the device tree name of the board, `devtype` define the boot device type, `devnum` define the device number(0: TF Card, 1: EMMC). `bootdir` define the path of the image files on the boot partition. `bootpart` define the boot partition on the device(0:1 means the first partition on device 0). `uenvcmd` define a script for U-boot, it will load kernel, device tree and bootup.

```
# This uEnv.txt file can contain additional environment settings that you
# want to set in U-Boot at boot time.  This can be simple variables such
# as the serverip or custom variables.  The format of this file is:
#    variable=value
# NOTE: This file will be evaluated after the bootcmd is run and the
#       bootcmd must be set to load this file if it exists (this is the
```

```
#       default on all newer U-Boot images.  This also means that some
#       variables such as bootdelay cannot be changed by this file since
#       it is not evaluated until the bootcmd is run.
#optargs=video=HDMI-A-1:800x600

# Uncomment the following line to enable HDMI display and disable LCD display.
fdtfile=myd_c335x.dtb
devtype=mmc
devnum=0
bootdir=/
bootpart=0:1
uenvcmd=if run loadimage; then run loadfdt; run loadramdisk; echo Booting from mmc${mm
cdev} ...; run ramargs; print bootargs; bootz ${loadaddr} ${rdaddr} ${fdtaddr}; fi;
```

> **Note: Startup of the MYD-AM335X Before starting the development board, note the connection of the JP8 jumper, connect JP8-1 and JP8-2 will boot system from SD Card, JP8-2 and JP8-3 will boot system from NandFlash. To effect JP8'S connection modification, please repower up the board after modification.MYD-AM335X-Y for jp1, MYD-AM335X-J for jp6.**

There are four boot modes for `MYD-AM335X` series development board to run embedded Linux system:

1. Boot from TF/SD card(EXT4 file system).
2. Boot from TF/SD card(Ramdisk file system).
3. Boot from NAND Flash（Ubi file system for core board with NAND Flash).
4. Boot from EMMC(EXT4 file system for core board with EMMC).
5. Boot from Ethernet（NFS root for debug）.

Boot from TF/SD card(EXT4 file system)

> **Note: Writing `sdcard.img` will format the TF card, please backup important files.**

After building Buildroot, a TF/SD card image file named as `sdcard.img` is generated at `<WORKDIR>/Filesystem/myir-buildroot/output/images`. It consists of two partitions, one is FAT partition contains `MLO`, `u-boot.img`, `zImage`, `uEnv.txt` and device tree binary files for MYD-AM335x series development board, the other partition is EXT4 partition, it will be used as the root partition of Linux.

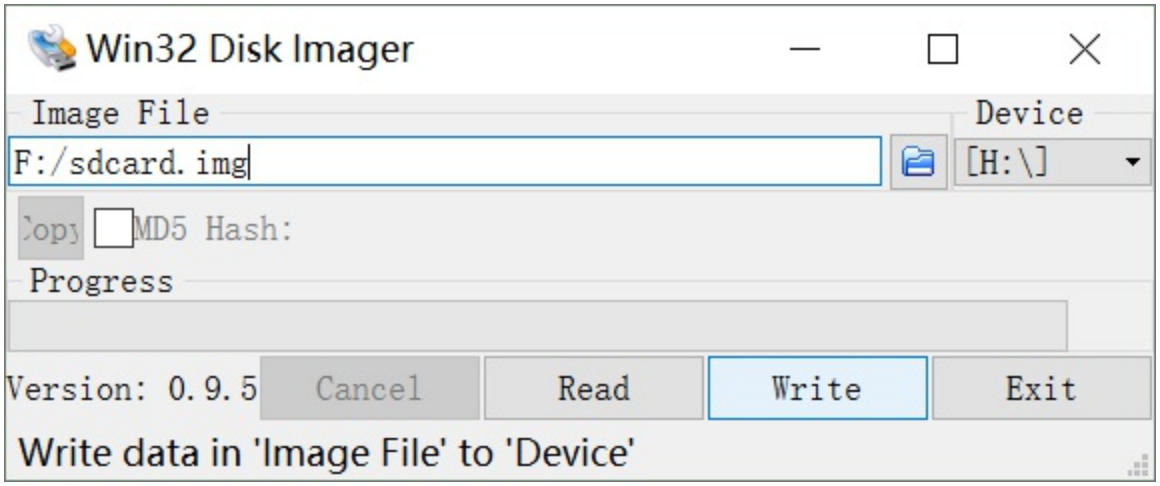

Figure 6-1 Write 'sdcard.img' to TF/SD card with 'win32diskimager'

- Put a TF card into the card reader, and connect the card reader to Windows host PC.
- Write `sdcard.img` to the TF card with win32diskimager.exe as shown in Figure 6-1 above.
- After writing, power off the MYD-AM335x series development board, put the TF card to its TF slot(J19), set the board to boot from TF/SD card by J5.
- Power on the MYD-AM335x series development board, it will boot from TF card and mount the second partition of the TF card as root file system.

Boot from TF/SD card(Ramdisk file system)

> **Note: HP USB Disk Storage Format Tool will clear original partitions of the TF card. To save the partitions, please use the formatting software provided by the computer system. Note: The size of ramdisk.gz is smaller than 32MB.**

- Format TF card

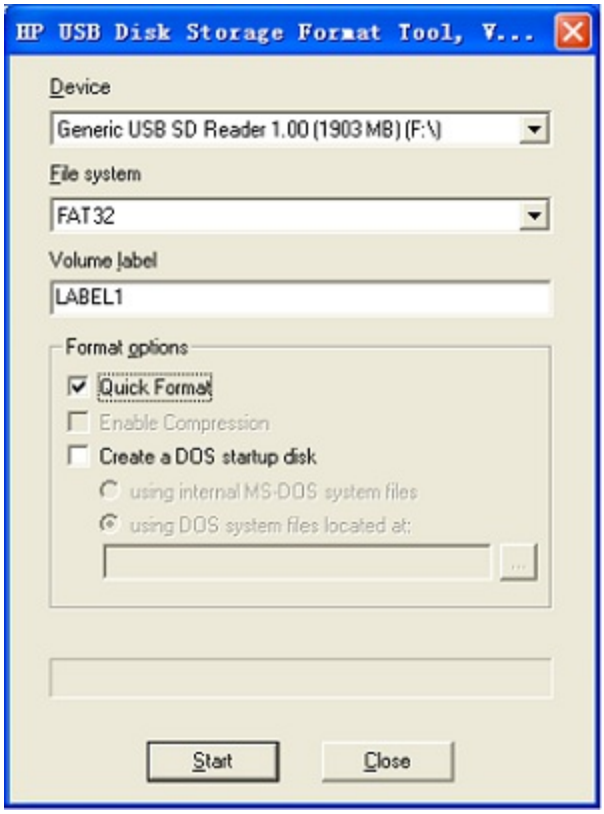

Figure 6-2 Formate TF Card with `HP USB Disk Storage Format Tool 2.0.6`

Please use “HP USB Disk Storage Format Tool 2.0.6” from CD directory “03-Tools” to format TF card.

```
1.Insert MMC/SD card into the card reader ,then connect the reader with the computer.

2.Open the HP USB Disk Storage Format Tool, the following steps will show in detail:

3.Select “FAT32”

4.Click “Start”.

5.When the formatting process is completed, click “OK”.
```

- Update the images.

Copy all files under directory “02-Images\Linux-image” from the CD to the TF card, move `uEnv_ramdisk.txt` to replace `uEnv.txt`. Then insert the TF card to the slot on the development board, Connect the corresponding board to set the start mode of the jumper cap 1-2 Pin, power on the board again,Enter root login.

Boot from NAND Flash（ubi file system for core board with NAND Flash)

Update of NAND boot image needs the aid of u-boot. Whether or not NAND Flash has data, the u-boot booted through TF card can be used to update NAND Flash images.

- Preparation

  1.Format the TF card to FAT or FAT32 file system by “HP USB Disk Storage Format Tool 2.0.6” from directory “03-Tools/” of CD.

  2.Copy "MLO", "u-boot.img", "uEnv.txt", "zImage","myd_*335x.dtb" and "rootfs.ubi" image files under directory “02-Images\Linux-image” to the TF card from the CD.

- Update

  Insert the TF card with the system images into the development board, Connect the corresponding board to set the boot mode of the jumper cap 1-2 pin, power on and boot it. Press any key on the PC keyboard to enter the u-boot according to the following prompts:

```
U-Boot 2016.05 (Jan 09 2017 - 19:37:43 +0800)

       Watchdog enabled
I2C:   ready
DRAM:  512 MiB
NAND:  512 MiB
MMC:   OMAP SD/MMC: 0, OMAP SD/MMC: 1
*** Warning - bad CRC, using default environment

Net:   cpsw
Press SPACE to abort autoboot in 2 seconds
MYIR>#
```

After entering the u-boot command line, input `run updatesys` from the PC keyboard to start automatic updating process. If the partitions were changed, users should erase the whole NAND Flash with command `nand erase.chip`:

```
MYIR># nand erase.chip
```

```
NAND erase.chip: device 0 whole chip
Erasing at 0x1ffe0000 -- 100% complete.
OK

MYIR># run updatesys
switch to partitions #0, OK
mmc0 is current device
reading MLO
55092 bytes read in 10 ms (5.3 MiB/s)

NAND write: device 0 offset 0x0, size 0xd734
 55092 bytes written: OK
reading myd_c335x.dtb
39229 bytes read in 10 ms (3.7 MiB/s)

NAND write: device 0 offset 0x80000, size 0x993d
 39229 bytes written: OK
reading u-boot.img
321300 bytes read in 34 ms (9 MiB/s)

NAND write: device 0 offset 0xc0000, size 0x4e714
 321300 bytes written: OK
reading zImage
4480016 bytes read in 396 ms (10.8 MiB/s)

NAND write: device 0 offset 0x200000, size 0x445c10
 4480016 bytes written: OK
reading rootfs.ubi
24248320 bytes read in 2111 ms (11 MiB/s)

NAND write: device 0 offset 0xa00000, size 0x1720000
 24248320 bytes written: OK
MYIR>#
```

After the system is updated, set the boot mode jumper to 2-3 pins. And then repower the board to boot from the Nand Flash.

Boot from EMMC(EXT4 filesystem for core board with EMMC)

Users who need large size of disk storage will assemble EMMC instead of NAND Flash on MYC-C335X core board. We can program the EMMC by aid of a TF Card with a ramdisk. The steps are shown below:

- Preparation

    1. Format a TF/SD with fat/fat32 format.
    2. Copy MLO, MLO_emmc, u-boot.img, u-boot_emmc.img, uEnv_ramdisk.txt, uEnv_mmc.txt, zImage, myd_c335x_emmc.dtb, ramdisk.gz, rootfs.tar.gz to TF/SD card.
    3. Set "fdtfile=myd_c335x_emmc.dtb" in uEnv_ramdisk.txt and rename uEnv_ramdisk.txt to uEnv.txt.
    4. Boot from TF/SD and login into linux
- Update

    1. Run "/etc/modules-load.myir/updatesys.sh loader2emmc sd" to write the image files from TF/SD to emmc
    2. Change the boot mode to emmc and repower up.

Boot from Ethernet（NFS root for debug）

After building Buildroot, a compressed package named as `rootfs.tar.gz` is generated at `<WORKDIR>/Filesystem/myir-buildroot/output/images`. This package can be used to work as NFS root for MYD-AM335X series development board. In order to boot from ethernet, TFTP and NFS services should be installed and configed as below:

Install TFTP Service

```
$ sudo apt-get install tftp-hpa tftpd-hpa
```

Config TFTP Service

Create a work directory for TFTP, open the configuration file for TFTP as shown below:

```
$ mkdir -p  <WORKDIR>/tftpboot
$ chmod 777 <WORKDIR>/tftpboot
$ sudo vi /etc/default/tftpd-hpa
```

Add or modify the parameters as shown below:

```
TFTP_DIRECTORY="<WORKDIR>/tftpboot"
TFTP_OPTIONS="-l -c -s"
```

Restart TFTP Service:

```
$ sudo service tftpd-hpa restart
```

Copy the `MLO` , `u-boot.img` , `zImage` , `ramdisk.gz` and device tree binary files to the work directory of TFTP service, then users can load these image files to the RAM of MYD-AM335X series development board by TFTP in U-boot console, it is shown below:

```
>#  help tftpboot
tftpboot - boot image via network using TFTP protocol

Usage:
tftpboot [loadAddress] [[hostIPaddr:]bootfilename]
>#  tftpboot ${loadaddr} 192.168.1.111:zImage
```

Install NFS Service.

NFS(Network File System) is a file system can be mounted remotely through network. A directory on NFS server can be used as the root file system of an embedded Linux system. The installation and configuration of NFS service are described below:

```
$ sudo apt-get install nfs-kernel-server
```

Config NFS Service.

Edit the `/etc/exports` file of NFS server, and export a directory at the end of file:

```
$ sudo vi /etc/exports
```

Add or modify the directory to be exported , such as `/home/myir/rootfs` has been added as below:

```
/home/myir/rootfs *(rw,nohide,insecure,no_subtree_check,async,no_root_squash)
```

Restart NFS service:

```
$ cd /home/myir/rootfs
$ sudo tar zxvf  <WORKDIR>/images/rootfs.tar.gz
$ sudo service nfs-kernel-server restart
```

Verify NFS service on NFS server:

```
$ sudo mount -t nfs 127.0.0.1:/home/myir/rootfs /mnt
```

- Mount the NFS ROOT filesystem If NFS service works well, `/home/myir/rootfs` will be mounted at `/mnt` with NFS , and then the NFS server is available for MYD-AM335x series development board.Power Up Modification Startup Mode The jumper cap is started for the nandflash mode，Nandflash must have uboot，The factory has been burned. Copy the zImage and dtb files to the /tftpboot directory, as follows:

  ```
  cp <WORKDIR>/Filesystem/myir-buildroot/output/images/zImage <WORKDIR>/tftpboot
  cp <WORKDIR>/Filesystem/myir-buildroot/output/images/myd_c335x.dtb <WORKDIR>/tftp
  boot
  ```

  Decompression rootfs.tar.gz to the /home/myir/rootfs directory as follows:

  ```
  cd /home/myir/rootfs
  sudo tar -xvf <WORKDIR>/Filesystem/myir-buildroot/output/images/rootfs.tar.gz ./
  ```

  For example, the IP address of the NFS server is set to `192.168.1.111` ,the IP address of the development board U-boot is set to `192.168.1.112` ,as follows:

  ```
  >#  setenv ipaddr 192.168.1.112
  >#  setenv serverip 192.168.1.111
  ```

  Verify the ethernet connection by `ping` command in U-boot console:

  ```
  # ping 192.168.1.111
  ```

Set the development board nfs mount directory and device tree file name, as follows:

```
>#  setenv rootpath /home/myir/rootfs        -- Extract the rootfs.tar.gz to the nfs di
rectory
```

```
>#  setenv fdtfile myd_c335x.dtb
>#  echo $fdtfile                              -- Check if the fdtfile name is correct
```

Save the relevant environment variables,as follows:

```
>#  saveenv
```

Under the U-Boot console, execute the netboot command to start the mounted NFS ROOT file system as follows:

```
>#  run netboot
```

> Note:Ping serverip normal, run runboot after tftp failure, please check the server side tftp service is normal, as follows:
>
> ```
> $ tftp 192.168.1.111
> tftp> get myd_c335x.dtb
> tftp> quit                                    --quit tftp
> ```
>
> If it fails, restart tftp service, as follows:
>
> ```
> $ sudo service tftpd-hpa restart
> ```

7.Peripheral Module Use

This chapter mainly describes the software hardware environment construction and testing process of the MYIR peripheral module on the development board.

The peripheral module support of the MYD-AM335X series development board is shown in the following table:

Module Name	Description	MYD-AM335X	MYD-AM335X-Y	MYD-AM335X-J	Reference Chapter
MY-TFT043RV2	4.3-Inch Resistive Touch Screen	√	√	√	7.1 4.3-Inch Resistive Touch Screen
MY-TFT070CV2	7-Inch Capacitive Touch Screen	√	√	√	4.1 LCD Test
MY-TFT070RV2	7-Inch Resistive Touch Screen	√	√	√	4.1 LCD Test
MY-WF003U	USB WIFI Module	√	√	√	7.2 WIFI Module
MY-GPS008C	GPS Module	√	√	√	7.4 GPS Module
MY-CAM002U	USB Camera Module	√	√	√	7.3 USB Camera Module
MY-GPRS007C	GPRS Module	√	√	√	7.5 GPRS Module
MY-WF004S	SDIO WIFI Module	√	√	√	7.2 WIFI Module
MY-CAM011B	BUS Camera	×	×	×	NONE

7.1 4.3-Inch Resistive Touch Screen

This chapter describes the environment construction and testing process of the 4.3-inch resistive screen on the development board.

Hardware Preparation：

- One MYD-AM335X series development board
- MY-TFT043RV2 connects to LCD interface of MYD-AM335X series development board
- One USB to TTL converter used to connect MYD-AM335X series development board Debug serial port and PC, PC side baud rate setting 115200-8-n-1

Interface	MYD-AM335X	MYD-AM335X-Y	MYD-AM335X-J
Debug Serial	J12 UART0	J10 Debug UART	J3 USB_UART
LCD Interface	J8	J7	J8

Software Preparation:

- Linux Kernel 4.1.18
- framebuffer_test application
- fbv application

Development Board	Device Tree
MYD-AM335X	myd_c335x.dtb
MYD-AM335X-Y	myd_y335x.dtb
MYD-AM335X-J	myd_j335x.dtb

The factory software supports the 7-inch capacitor and the resistance screen directly by default. Now you need to use the 4.3-inch resistive screen. You need to modify the device tree to support it.

The configuration of the 4.3-inch screen in the device tree is commented by default. Search for / *4.3 inch, 480x272 resolution LCD, MYiR* / in the device tree to release the commented code and comment the configuration of the 7-inch screen. After the

modification is completed, use the command make dtbs to compile the device tree, then copy the compiled device to the SD card and update the software of the development board using the methods in Chapter 6.

Test Steps:

- Copy the executable `framebuffer_test` in the directory `<WORKDIR>/Examples/rootfs/usr/bin/` to the development board `/usr/bin/` directory, run `framebuffer_test` application as below:

```
# chmod 777 /usr/bin/framebuffer_test
# framebuffer_test -h
Usage: framebuffer_test [options]

Version 1.0
Available options:
-d | --device name   framebuffer device name, default: /dev/fb0
-h | --help          Print this message

# framebuffer_test -d /dev/fb0
xres:480 >>> yres:272 >>> bpp:32>>>
```

During `framebuffer_test` running, serval colors of background are painted on LCD one by one, and then colorful points, lines, areas are painted.

- Copy a BMP file with 32BPP and resolution of 480*272 to `/media/1.bmp` of the development board, display the picture on LCD by fbv application:

```
# fbv
Usage: fbv [options] image1 image2 image3 ...

Available options:
 --help        | -h : Show this help
 --alpha       | -a : Use the alpha channel (if applicable)
 --dontclear   | -c : Do not clear the screen before and after displaying the image
 --donthide    | -u : Do not hide the cursor before and after displaying the image
 --noinfo      | -i : Supress image information
 --stretch     | -f : Strech (using a simple resizing routine) the image to fit onto
                                          screen if necessary
 --colorstretch| -k : Strech (using a 'color average' resizing routine) the image to
                                            fit onto screen if necessary
 --enlarge     | -e : Enlarge the image to fit the whole screen if necessary
 --ignore-aspect| -r : Ignore the image aspect while resizing
 --delay <d>   | -s <delay> : Slideshow, 'delay' is the slideshow delay in tenths of s
econds.
```

```
Keys:
 r             : Redraw the image
 a, d, w, x    : Pan the image
 f             : Toggle resizing on/off
 k             : Toggle resizing quality
 e             : Toggle enlarging on/off
 i             : Toggle respecting the image aspect on/off
 n             : Rotate the image 90 degrees left
 m             : Rotate the image 90 degrees right
 p             : Disable all transformations
Copyright (C) 2000 - 2004 Mateusz Golicz, Tomasz Sterna.
Error: Required argument missing.

# fbv /meida/1.bmp
fbv - The Framebuffer Viewer
/media/1.bmp
480 x 272
```

After complete, the picture displays just right for the LCD.

7.2 WIFI Module

This chapter describes the environment setup and testing process for USB WIFI and SIDO WIFI on the development board.

Hardware Preparation：

- One MYD-AM335X series development board
- MY-WF003U module、MY-WF004S module
- One USB to TTL converter used to connect MYD-AM335X series development board Debug serial port and PC, PC side baud rate setting 115200-8-n-1

Interface	MYD-AM335X	MYD-AM335X-Y	MYD-AM335X-J
Debug Serial	J12 UART0	J10 Debug UART	J3 USB_UART
USB Interface	J3 J4	J19 USB_HOST	J27 USB_HOST
SDIO Interface	J17 TF_CARD	J12	J19 Micro_SD

Software Preparation:

- Linux Kernel 4.1.18
- wpa_supplicant application
- hostapd application
- iptables application

Development Board	Device Tree
MYD-AM335X	myd_c335x.dtb
MYD-AM335X-Y	myd_y335x.dtb
MYD-AM335X-J	myd_j335x.dtb

Test Steps:

Insert the MY-WF003U module into the development board USB interface or the MY-WF004S module into the development board TF card interface. Use the command `ifconfig -a` to see that the module has loaded the driver and generated the WLAN device.

```
# ifconfig -a


eth0      Link encap:Ethernet  HWaddr 68:9E:19:BC:1C:84
          UP BROADCAST MULTICAST  MTU:1500  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:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
          Interrupt:240

eth1      Link encap:Ethernet  HWaddr 68:9E:19:BC:1C:86
          BROADCAST MULTICAST  MTU:1500  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:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

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)

wlan0     Link encap:Ethernet  HWaddr 00:1D:43:A0:04:11
          BROADCAST MULTICAST  MTU:1500  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:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

WIFI STA Mode

1. Modify the value of ssid and psk in the */etc/wap_supplicant.conf* configuration file, ssid is the name of the WIFI AP in the test environment, and psk is the password of the WIFI AP in the test environment.

```
# cat /etc/wpa_supplicant.conf
ctrl_interface=/var/run/wpa_supplicant
ap_scan=1
p2p_disabled=1

network={
```

```
 ssid="MYIR_TECH"
 scan_ssid=1
 proto=WPA RSN
 pairwise=CCMP TKIP NONE
 key_mgmt=WPA-EAP WPA-PSK IEEE8021X NONE
 group=TKIP CCMP
 psk="myir2016"
 priority=10
}
```

2.Connect WIFI AP.

```
# wpa_supplicant -i wlan0 -c /etc/wpa_supplicant.conf -B
```

3.Get the IP address and DNS.

```
#udhcpc -b -i wlan0
udhcpc: started, v1.25.1
udhcpc: sending discover
udhcpc: sending select for 192.168.30.115
udhcpc: lease of 192.168.30.115 obtained, lease time 3600
deleting routers
adding dns 223.5.5.5
adding dns 201.104.111.114
```

4.Ping Test.

```
# ping www.baidu.com
PING www.baidu.com (61.135.169.125): 56 data bytes
64 bytes from 61.135.169.125: seq=0 ttl=56 time=29.253 ms
64 bytes from 61.135.169.125: seq=1 ttl=56 time=32.218 ms
64 bytes from 61.135.169.125: seq=2 ttl=56 time=24.717 ms
64 bytes from 61.135.169.125: seq=4 ttl=56 time=141.210 ms
64 bytes from 61.135.169.125: seq=5 ttl=56 time=64.064 ms
64 bytes from 61.135.169.125: seq=11 ttl=56 time=133.112 ms
```

WIFI SoftAP Mode

1.Modify the value of ssid and wpa_passphrase in the */etc/hostapd.conf* configuration file. ssid is the name of the ap hotspot generated by the wifi module, and wpa_passphrase is the password of the ap hotspot.

```
# cat /etc/hostapd.conf
interface=wlan0
```

```
ssid=myirAP
hw_mode=g
channel=6
macaddr_acl=0
auth_algs=1
ignore_broadcast_ssid=0
wpa=2
wpa_passphrase=123456789
wpa_key_mgmt=WPA-PSK
wpa_pairwise=TKIP
rsn_pairwise=CCMP
```

2.Modify the contents of the /etc/dhcp/dhcpd.conf configuration file as needed. This configuration file is mainly used to provide DHCP services for devices connected to WIFI APs.

```
# cat /etc/dhcp/dhcpd.conf
ddns-update-style interim;
ignore client-updates;

subnet 192.168.43.0 netmask 255.255.255.0{
range 192.168.43.100 192.168.43.150;
default-lease-time 86400;
max-lease-time 86400;
option routers 192.168.43.1;
option broadcast-address 192.168.43.255;
option subnet-mask 255.255.255.0;
option domain-name "redpinesignals.com";
option domain-name-servers 114.114.114.114;
host VAP_0.redpinesignals.com{
        hardware ethernet 00:23:a7:3a:11:d1;
        fixed-address 192.168.43.1;
        }
}
```

3.Create an ap hotspot.

```
# hostapd /etc/hostapd.conf -B
Configuration file: /etc/hostapd.conf
Using interface wlan0 with hwaddr 00:1d:43:a0:04:11 and ssid "myirAP"
random: Only 15/20 bytes of strong random data available from /dev/random
random: Not enough entropy pool available for secure operations
WPA: Not enough entropy in random pool for secure operations - update keys later when
the first station connects
wlan0: interface state UNINITIALIZED->ENABLED
```

```
wlan0: AP-ENABLED
```

4.Configure the ip address and subnet mask of the ap hotspot. This IP address needs to be set according to the value of the routers in /etc/dhcp/dhcpd.conf.

```
# ifconfig wlan0 192.168.43.1 netmask 255.255.255.0 up
```

5.Start the DHCP service of wlan0.

```
# dhcpd wlan0
Internet Systems Consortium DHCP Server 4.3.5
Copyright 2004-2016 Internet Systems Consortium.
All rights reserved.
For info, please visit https://www.isc.org/software/dhcp/
WARNING: Host declarations are global.  They are not limited to the scope you declared
 them in.
Config file: /etc/dhcp/dhcpd.conf
Database file: /var/lib/dhcp/dhcpd.leases
PID file: /var/run/dhcpd.pid
Wrote 0 deleted host decls to leases file.
Wrote 0 new dynamic host decls to leases file.
Wrote 0 leases to leases file.
Listening on LPF/wlan0/00:1d:43:a0:04:11/192.168.43.0/24
Sending on   LPF/wlan0/00:1d:43:a0:04:11/192.168.43.0/24
Sending on   Socket/fallback/fallback-net
```

After completing the above five steps, you can use other WIFI STA mode devices to connect to the created AP hotspots, but only LAN communication. If you need to perform external network communication, you need to complete the following steps.

1.Connect eth0 to the Internet and obtain an IP address.

```
# udhcpc eth0
udhcpc: started, v1.25.1
udhcpc: sending discover
udhcpc: sending select for 192.168.30.160
udhcpc: lease of 192.168.30.160 obtained, lease time 3600
deleting routers
adding dns 223.5.5.5
adding dns 201.104.111.114
```

2.Enable IP packet forwarding.

```
echo 1 > /proc/sys/net/ipv4/ip_forward
```

3.Configure eth0 as the exit of all packets.

```
iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

After performing the above three steps, the WIFI device can access the Internet through this AP hotspot.

7.3 USB Camera Module

This chapter describes the environment setup and testing process for the USB camera on the development board.

Hardware Preparation：

- One MYD-AM335X series development board
- MY-CAM002U module
- One USB to TTL converter used to connect MYD-AM335X series development board Debug serial port and PC, PC side baud rate setting 115200-8-n-1

Interface	MYD-AM335X	MYD-AM335X-Y	MYD-AM335X-J
Debug Serial	J12 UART0	J10 Debug UART	J3 USB_UART
USB Interface	J3 J4	J19 USB_HOST	J27 USB_HOST

Software Preparation:

- Linux Kernel 4.1.18
- fswebcam application
- fbv application

Development Board	Device Tree
MYD-AM335X	myd_c335x.dtb
MYD-AM335X-Y	myd_y335x.dtb
MYD-AM335X-J	myd_j335x.dtb

Test Steps:

- Insert the camera module into the USB interface of the development board, the system will automatically match the driver and create the device head device /dev/video0.
- Use the test program fswebcam to take a photo.

```
fswebcam -d /dev/video0 --no-banner -r 640x480 image.jpg
```

- Copy image.jpg to the PC to display. If the development board is connected to the LCD screen, you can use the command to output the captured photos to the LCD for display.

```
# fbv image.jpg
fbv - The Framebuffer Viewer
image.jpg
640 x 480
```

7.4 GPS Module

This chapter describes the environment construction and testing process of the GPS module on the development board.

Hardware Preparation：

- One MYD-AM335X series development board
- MY-GPS008C module
- One USB to TTL converter used to connect MYD-AM335X series development board Debug serial port and PC, PC side baud rate setting 115200-8-n-1

Interface	MYD-AM335X	MYD-AM335X-Y	MYD-AM335X-J
Debug Serial	J12 UART0	J10 Debug UART	J3 USB_UART
Expand Interface	J20	J13	J25

Software Preparation:

- Linux Kernel 4.1.18
- microcom application

Development Board	Device Tree
MYD-AM335X	myd_c335x.dtb
MYD-AM335X-Y	myd_y335x.dtb
MYD-AM335X-J	myd_j335x.dtb

Test Steps:

- If the development board has an expand interface, the GPS module is directly connected to the expand interface, and the development board without the expand interface directly uses the idle serial port of the development board to perform docking.
- Use the microcom tool to read the data returned by the GPS module.

```
# microcom -s 9600 /dev/ttyO1
$GPGSA,A,2,01,04,11,17,28,,,,,,,,2.8,2.7,0.9*3C
$GPGSV,3,1,09,01,59,035,53,03,31,127,21,04,30,037,50,06,06,219,33*75
```

```
$GPGSV,3,2,09,11,45,032,42,17,24,285,54,19,09,059,35,28,40,334,51*7C
$GPGSV,3,3,09,32,21,084,27*4B
$GPGGA,061102.0,2233.150278,N,11356.386896,E,1,05,2.7,83.5,M,-1.0,M,,*79
$GPVTG,,T,0.0,M,0.0,N,0.0,K,A*0D
$GPRMC,061102.0,A,2233.150278,N,11356.386896,E,0.0,,230715,0.0,E,A*2E
```

7.5 GPRS Module

This chapter describes the environment construction and testing process of the GPRS module on the development board.

Hardware Preparation：

- One MYD-AM335X series development board
- MY-GPRS007C module
- One USB to TTL converter used to connect MYD-AM335X series development board Debug serial port and PC, PC side baud rate setting 115200-8-n-1

Interface	MYD-AM335X	MYD-AM335X-Y	MYD-AM335X-J
Debug Serial	J12 UART0	J10 Debug UART	J3 USB_UART
Expand Interface	J20	J13	J25

Software Preparation:

- Linux Kernel 4.1.18
- pppd application

Development Board	Device Tree
MYD-AM335X	myd_c335x.dtb
MYD-AM335X-Y	myd_y335x.dtb
MYD-AM335X-J	myd_j335x.dtb

Test Steps:

- If the development board has an expand interface, the GPRS module is directly connected to the expand interface, and the development board without the expand interface directly uses the idle serial port of the development board to perform docking.
- Modify /etc/ppp/peers/gprs in the configuration file /dev/ttyUSB1 to the currently connected serial device, such as /dev/ttyO3.

```
#/etc/ppp/peers/gprs
```

```
# Usage:   root>pppd call gprs
/dev/ttyO3
115200
#crtscts
modem
noauth
debug
nodetach
#hide-password
usepeerdns
noipdefault
defaultroute
user "cmnet"
0.0.0.0:0.0.0.0
#ipcp-accept-local
#ipcp-accept-remote
#lcp-echo-failure  12
#lcp-echo-interval 3
noccp
#novj
#novjccomp
persist
connect '/usr/sbin/chat -s -v -f /etc/ppp/peers/gprs-connect-chat'
#connect '/bin/chat -v -s -f /etc/ppp/gprs-connect-chat'
#disconnect '/bin/chat -v -f /etc/ppp/gprs-disconnect-chat'
```

- Use the command to make a dial-up connection.

```
#pppd call gprs &
```

- ping test。

```
# ping www.baidu.com
PING www.baidu.com (61.135.169.125): 56 data bytes
64 bytes from 61.135.169.125: seq=0 ttl=56 time=29.253 ms
64 bytes from 61.135.169.125: seq=1 ttl=56 time=32.218 ms
64 bytes from 61.135.169.125: seq=2 ttl=56 time=24.717 ms
64 bytes from 61.135.169.125: seq=4 ttl=56 time=141.210 ms
```

Appendix 1 Warranty & Technical Support Services

MYIR Tech Limited is a global provider of ARM hardware and software tools, design solutions for embedded applications. We support our customers in a wide range of services to accelerate your time to market.

MYIR is an ARM Connected Community Member and work closely with ARM and many semiconductor vendors. We sell products ranging from board level products such as development boards, single board computers and CPU modules to help with your evaluation, prototype, and system integration or creating your own applications. Our products are used widely in industrial control, medical devices, consumer electronic, telecommunication systems, Human Machine Interface (HMI) and more other embedded applications. MYIR has an experienced team and provides custom design services based on ARM processors to help customers make your idea a reality.

The contents below introduce to customers the warranty and technical support services provided by MYIR as well as the matters needing attention in using MYIR's products.

Service Guarantee

MYIR regards the product quality as the life of an enterprise. We strictly check and control the core board design, the procurement of components, production control, product testing, packaging, shipping and other aspects and strive to provide products with best quality to customers. We believe that only quality products and excellent services can ensure the long-term cooperation and mutual benefit.

Price

MYIR insists on providing customers with the most valuable products. We do not pursue excess profits which we think only for short-time cooperation. Instead, we hope to establish long-term cooperation and win-win business with customers. So we will offer reasonable prices in the hope of making the business greater with the customers together hand in hand.

Delivery Time

MYIR will always keep a certain stock for its regular products. If your order quantity is less than the amount of inventory, the delivery time would be within three days; if your order quantity is greater than the number of inventory, the delivery time would be always four to six weeks. If for any urgent delivery, we can negotiate with customer and try to supply the goods in advance.

Technical Support

MYIR has a professional technical support team. Customer can contact us by email (support@myirtech.com), we will try to reply you within 48 hours. For mass production and customized products, we will specify person to follow the case and ensure the smooth production.

After-sale Service

MYIR offers one year free technical support and after-sales maintenance service from the purchase date. The service covers:

1. Technical support service

```
 * MYIR offers technical support for the hardware and software materials which ha
ve provided to customers;
 * To help customers compile and run the source code we offer;
 * To help customers solve problems occurred during operations if users follow th
e user manual documents;
 * To judge whether the failure exists;
 * To provide free software upgrading service.
```

However, the following situations are not included in the scope of our free technical support service:

```
    * Hardware or software problems occurred during customers' own development;
    * Problems occurred when customers compile or run the OS which is tailored by them
selves;
    * Problems occurred during customers' own applications development;
    * Problems occurred during the modification of MYIR's software source code.
```

1. After-sales maintenance service

The products except LCD, which are not used properly, will take the twelve months free maintenance service since the purchase date. But following situations are not included in the scope of our free maintenance service:

```
* The warranty period is expired;
* The customer cannot provide proof-of-purchase or the product has no serial number;
* The customer has not followed the instruction of the manual which has caused the dam
age the product;
* Due to the natural disasters (unexpected matters), or natural attrition of the compo
nents,
or unexpected matters leads the defects of appearance/function;
* Due to the power supply, bump, leaking of the roof, pets, moist, impurities into the
 boards,
all those reasons which have caused the damage of the products or defects of appearanc
e;
* Due to unauthorized weld or dismantle parts or repair the products which has caused
the damage of the products or defects of appearance;
* Due to unauthorized installation of the software,
system or incorrect configuration or computer virus which has caused the damage of pro
ducts.
```

Warm tips:

1. MYIR does not supply maintenance service to LCD. We suggest the customer first check the LCD when receiving the goods. In case the LCD cannot run or no display, customer should contact MYIR within 7 business days from the moment get the goods.
2. Please do not use finger nails or hard sharp object to touch the surface of the LCD.
3. MYIR suggests user purchasing a piece of special wiper to wipe the LCD after long time use, please avoid clean the surface with fingers or hands to leave fingerprint.
4. Do not clean the surface of the screen with chemicals.
5. Please read through the product user manual before you using MYIR’s products.
6. For any maintenance service, customers should communicate with MYIR to confirm the issue first. MYIR’s support team will judge the failure to see if the goods need to be returned for repair service, we will issue you RMA number for return maintenance service after confirmation.

1. Maintenance period and charges
    - MYIR will test the products within three days after receipt of the returned goods and inform customer the testing result. Then we will arrange

shipment within one week for the repaired goods to the customer. For any special failure, we will negotiate with customers to confirm the maintenance period.

- For products within warranty period and caused by quality problem, MYIR offers free maintenance service; for products within warranty period but out of free maintenance service scope, MYIR provides maintenance service but shall charge some basic material cost; for products out of warranty period, MYIR provides maintenance service but shall charge some basic material cost and handling fee.

2. Shipping cost

During the warranty period, the shipping cost which delivered to MYIR should be responsible by user; MYIR will pay for the return shipping cost to users when the product is repaired. If the warranty period is expired, all the shipping cost will be responsible by users.

1. Products Life Cycle

MYIR will always select mainstream chips for our design, thus to ensure at least ten years continuous supply; if meeting some main chip stopping production, we will inform customers in time and assist customers with products updating and upgrading.

Value-added Services

1. MYIR provides services of driver development base on MYIR's products, like serial port, USB, Ethernet, LCD, etc.
2. MYIR provides the services of OS porting, BSP drivers' development, API software development, etc.
3. MYIR provides other products supporting services like power adapter, LCD panel, etc.
4. ODM/OEM services.

MYIR Tech Limited

Room 04,6th Floor, Building No.2,Fada Road,

Yunli Intelligent Park, Bantian, Longgang District, Shenzhen, Guangdong, China 518129

Support Email: support@myirtech.com

Sales Email: sales@myirtech.com

Phone: +86-755-22984836

Fax: +86-755-25532724

Website: www.myirtech.com