MYD-Y6ULX-HMI Linux Development Guide

MYIR™ Make Your Idea Real

Table of Contents

MYD-Y6ULX-HMI Linux Development Guide

This document introduce the MYD-Y6ULX-HMI development board about Linux compile and deploy, interface usage on baseboard, Qt application development etc.

Version History

Version	Description	Date
V1.0	Initialize version	2018.10.28

Hardware Version

This document suit for below boards:

- MYD-Y6ULX-HMI
- MYD-Y6ULX-HMI-4GEXP
- MYD-Y6ULX-CHMI

1 Software Resource

MYD-Y6ULX-HMI series boards support the Linux kernel version 4.1.15, and provided with rich hardware resource and software resource.

- Attention: MYD-Y6ULX-HMI is factory-programmed with a DEMO system by default. Demo is a UI developed based on QT5. It also has a Tornado Web Server. Use yocto to build out this system.0
- DEMO system code:
  - 04-Source/HMI-QT5-DEMO.tar.bz2
  - 04-Source/web_server.tar.bz2
- DEMO system documentation:
  - 01-Document/User_Manual/Chinese/MEasy HMI Development Manual.pdf

Below is MYD-Y6ULX-HMI software resource table:

Categary	Name	Description	Source
Bootloader	U-boot	u-boot.imx will boot chip to work	YES
Linux kernel	Linux 4.1.15	Based on official version imx_4.1.15_2.0.0_ga	YES
Driver	USB Host	USB Host	YES
Driver	USB OTG	USB OTG driver	YES
Driver	I2C	I2C bus driver	YES
Driver	Ethernet	10/100Mbps ethernet driver	YES
Driver	MMC	MMC/SDIO/TF Card	YES
Driver	LCD	Supports 7.0 inch	YES
Driver	RTC	Read/write real date time	YES
Driver	Touch Panel	Supports Capacity and Resistive touch panel	YES
Driver	USART	Serial port driver	YES
Driver	LED	GPIO LED	YES
Driver	KEY	GPIO KEY	YES
Driver	Audio	WM8904 codec driver	YES
Driver	CAN bus	CAN bus driver	YES
Driver	RS485	RS485 bus driver	YES
Driver	Camera	OV2659 driver	YES
Driver	WiFi & BT	AP6212	YES
Driver	LTE module(Optional)	4G module(Quectl EC20) use USB driver,Support GPS	YES
FileSystem	Yocto rootfs	Based Yocto build filesystem(include Qt 5.6 package)	YES
FileSystem	Yocto rootfs	Based Yocto build filesystem(Full command line package)	YES
Application	GPIO KEY	Reads the GPIO key code value demo	YES

Application	GPIO LED	Operate the GPIO LED demo	YES
Application	NET	Uses TCP/IP Socket API, support Client and Server demo	YES
Application	RTC	Read/write real date time demo	YES
Application	RS232	Read/write RS232 port demo	YES
Application	RS485	Read/write RS485 port demo	YES
Application	Audio	Audio play/capture demo	YES
Application	Framebuffer	LCD test program	YES
Toolchain	Cross compiler	Linaro GCC 4.9 Hardfloat	BINARY
Toolchain	Cross compiler	Yocto GCC 5.3 Hardfloat	BINARY

Table1-1 Software Resource

2 Deploy Development Environment

You need to install the Linux Operation System on your host PC.Recommend to use the Ubuntu 16.04 64bit distribution, and connect the network.Next steps we will install some packages from internet.

Connect development board with PC

1. PC use USB to TLL cable with DEBUG port(JP1) on board.
2. Open serial program with exist serial device.
3. The development board user name is root, no password.

PC serial port configure parameters:

- Baudrate: 115200
- Data bit: 8bit
- Parity: None
- Stop bit: 1bit
- Flow control: Disable

Product Pictures

There are the following core boards:

- MYC-Y6ULY2-256N256D-50-C
- MYC-Y6ULY2-256N256D-50-I
- MYC-Y6ULY2-512N256D-50-C (Non-standard items, reservation is required)
- MYC-Y6ULY2-4E512D-50-C
- MYC-Y6ULY2-4E512D-50-I
- MYC-Y6ULY2-4E256D-50-I (Non-standard items, reservation is required)

Figure2-1 MYC-Y6ULY2-256N256D-50-C top view

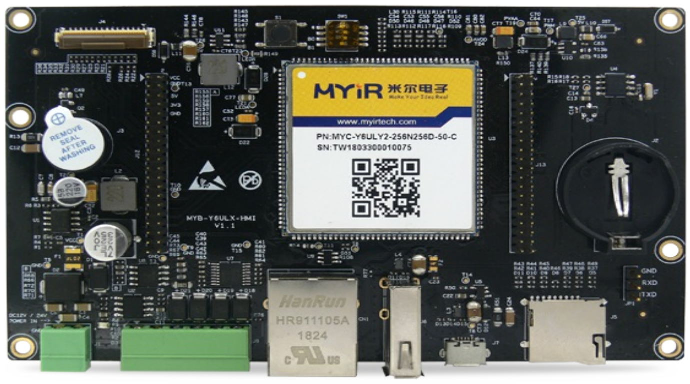


Figure2-2 MYD-Y6ULX-HMI top view

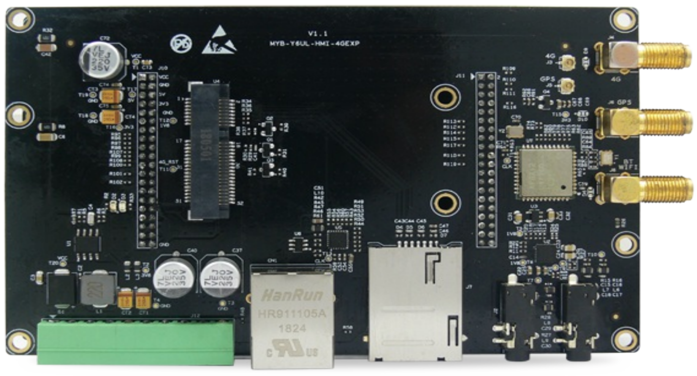


Figure2-3 MYB-Y6ULX-HMI-4GEXP top view

Install necessary software packages

```
$sudo apt-get install build-essential git-core libncurses5-dev flex bison \
texinfo zip unzip zlib1g-dev gettext u-boot-tools g++ xz-utils mtd-utils \
gawk diffstat gcc-multilib lzop
```

Build work directory

Create a working directory to facilitate the creation of an unified environment variable path. Copy the product CD-ROM source code to the working directory, while setting the DEV_ROOT variable to enable the follow-up step path accessed.

```
$mkdir -p ~/MYD-Y6ULX-devel
$export DEV_ROOT=~/MYD-Y6ULX-devel
$cp -r <DVDROM>/02-Images $DEV_ROOT
$cp -r <DVDROM>/03-Tools $DEV_ROOT
$cp -r <DVDROM>/04-Source $DEV_ROOT
```

Configure toolchain

- Linaro toolchain : gcc version 4.9.3 20141031 (prerelease) (Linaro GCC 2014.11)
- Yocto toolchain: gcc version 5.3.0 (GCC)

There are two cross compile toolchains, one is support by Linaro.Another built by Yocto.Recommend you use Yocto toolchain to build all source code.

Linaro Toolchain

```
$cd $DEV_ROOT/
$tar -xvf 03-Tools/Toolchain/gcc-linaro-4.9-2014.11-x86_64_arm-linux-gnueabihf.tar.xz
```

```
$export PATH=$PATH:$DEV_ROOT/gcc-linaro-4.9-2014.11-x86_64_arm-linux-gnueabihf/bin
$export CROSS_COMPILE=arm-linux-gnueabihf-
$export ARCH=arm
```

Check if the toolchain is correct using below command.You have setup correct environment on current SHELL when you get version infomation.If you want it always available, you need to modify your shell config file.

```
$ arm-linux-gnueabihf-gcc --version

arm-linux-gnueabihf-gcc (Linaro GCC 2014.11) 4.9.3 20141031 (prerelease)
Copyright (C) 2014 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

Yocto Toolchain

Yocto provided two kinds toolchain, one is low-level development toolchain meta-toolchain, another is application development toolchain.The low-level toolchain likes Linaro.The another used for application development, include more third-party libaries and header files.The MYD-Y6ULX-HMI also supports both two kinds, those files are listed below.

Toolchain for Application:

Toolchain file name	Description
myir-imx-fb-glibc-x86_64-fsl-image-qt5-cortexa7hf-neon-toolchain-4.1.15-2.0.1.sh	toolchain for fsl-image-qt5 system image, icludes Qt5 libraries
myir-imx-fb-glibc-x86_64-core-image-base-cortexa7hf-neon-toolchain-4.1.15-2.0.1.sh	toolchain for core-image-base system image, without any graphics libraries

Toolchain for driver:

Toolchain file name	Description
myir-imx-fb-glibc-x86_64-meta-toolchain-cortexa7hf-neon-toolchain-4.1.15-2.0.1.sh	meta-toolchain

Yocto toolchain distribute SDK package type. You need install the toolchain SDK package, then use it. Below is install method:

Run shell script as normal user.It will request you to input install path, default is under "/opt" directory.Then you will reuquest to set permission to directory. You can use "source" or "." to load toolchain environment to current shell when your installation finish.

Below example intall the toolchain into '/opt/myir-imx6ulx-fb/4.1.15-2.0.1' directory.

```
$ ./myir-imx-fb-glibc-x86_64-fsl-image-qt5-cortexa7hf-neon-toolchain-4.1.15-2.0.1.sh
Freescale i.MX Release Distro SDK installer version 4.1.15-2.0.1
================================================================
Enter target directory for SDK (default: /opt/myir-imx-fb/4.1.15-2.0.1): /opt/myir-imx6ulx-fb/4.1.15-2.0.1
Do You are about to install the SDK to "/opt/myir-imx6ulx-fb/4.1.15-2.0.1". Proceed[Y/n]? Y
[sudo] password for kevinchen:
Extracting SDK................................................
................................................................
...............done
Setting it up...done
SDK has been successfully set up and is ready to be used.
Each time you wish to use the SDK in a new shell session, you ne
ed to source the environment setup script e.g.
$ . /opt/myir-imx6ulx-fb/4.1.15-2.0.1/environment-setup-cortexa7hf-neon-poky-linux-gnueabi
```

Check the toolchain SDK is correct after installation.Using the "source" command to load environment file to shell and check the compiler version.

```
source /opt/myir-imx6ulx-fb/4.1.15-2.0.1/environment-setup-cortexa7hf-neon-poky-linux-gnueabi
arm-poky-linux-gnueabi-gcc --version

arm-poky-linux-gnueabi-gcc (GCC) 5.3.0
Copyright (C) 2015 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
```

```
  warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

According the steps, you can install the low-level toolchain meta-toolcahin.Please input different path to store the toolchain, otherwize it will cover existing files in same directory.

3 Build Linux System

This chapter introduces how to build components of Linux system. The MYD-Y6ULX-HMI includes below parts:

- U-Boot: First level bootloader.
- Linux Kernel: Linux kernel 4.1.15 and drivers suit for MYD-Y6ULX-HMI board.
- Yocto: An open source collaboration project that provides templates, tools and methods to help you create custom Linux-based system for embedded products regardless of the hardware architecture.

These software are location in 04-Source directory.Some packages use ID number in file name.

Before compiling u-boot and Linux kernel source code, you need install meta-toolchain and load the environment variables into current shell.

3.1 U-Boot

Enter Bootloader directory, extract U-boot source tar source package:

```
cd $DEV_ROOT/04-Source
tar -xvf MYiR-iMX-uboot.tar.gz
cd MYiR-iMX-uboot
```

Compiling：

```
make distclean
make <config>
make
```

Choose the right config file according to your CPU part number.

Core board	Config file
MYC-Y6ULY2-256N256D-50-C	myd_y6ull_14x14_nand_defconfig
MYC-Y6ULY2-256N256D-50-I	myd_y6ull_14x14_nand_defconfig
MYC-Y6ULY2-4E512D-50-C	myd_y6ull_14x14_emmc_defconfig
MYC-Y6ULY2-4E512D-50-I	myd_y6ull_14x14_emmc_defconfig

U-Boot SD boot mode will search and execute a script file "boot.scr" when U-Boot booting up. It used to change boot type in temporary.Next is use TFTP to download zImage and dtb to boot system as example. Using mkimage tool through "myd-y6ull-boot-mmc0-tftp.txt" to generate the "boot.scr" file as example. The mkimage tool source code is locating in "U-Boot/tools" directory. It will be compiled after U-Boot compiled.

The text myd-y6ull-boot-mmc0-tftp.txx reads as follows:

```
setenv mmcroot '/dev/mmcblk0p2 rootwait rw rootdelay=5 mem=256M'
run mmcargs
tftpboot 0x83000000 zImage
tftpboot 0x84000000 myd-y6ull-gpmi-weim.dtb
bootz 0x83000000 - 0x84000000
```

The command to generate boot.scr is as follows:

```
mkimage -A arm -T script -O linux -d myd-y6ull-boot-mmc0-tftp.txt boot.scr
```

3.2 Linux Kernel

Enter Kernel directory, extract it:

```
cd $DEV_ROOT/04-Source
tar -xvf MYiR-iMX-Linux.tar.bz2
cd MYiR-iMX-Linux
```

Compiling:

```
make distclean
make myd_y6ulx_hmi_defconfig
make zImage dtbs
```

When the compilation is done, the kernel image file zImage is generated in the 'arch/arm/boot' directory, and the DTB file is generated in the 'arch/arm/boot/dts' directory.

DTB, core board, Base board and extension board shall be paired as follows:

DTB file	Core board	Base board	Extension board
myd-y6ull-gpmi-weim.dtb	MYC-Y6ULY2-256N256D-50-C	MYB-Y6ULX-HMI	MYB-Y6ULX-HMI-4GEXP
myd-y6ull-gpmi-weim.dtb	MYC-Y6ULY2-256N256D-50-I	MYB-Y6ULX-HMI	MYB-Y6ULX-HMI-4GEXP
myd-y6ull-gpmi-weim-without-exp.dtb	MYC-Y6ULY2-256N256D-50-C	MYB-Y6ULX-HMI	-
myd-y6ull-gpmi-weim-without-exp.dtb	MYC-Y6ULY2-256N256D-50-I	MYB-Y6ULX-HMI	-
myd-y6ull-emmc.dtb	MYC-Y6ULY2-4E512D-50-C	MYB-Y6ULX-HMI	MYB-Y6ULX-HMI-4GEXP
myd-y6ull-emmc.dtb	MYC-Y6ULY2-4E512D-50-I	MYB-Y6ULX-HMI	MYB-Y6ULX-HMI-4GEXP
myd-y6ull-gpmi-weim-without-exp.dtb	MYC-Y6ULY2-4E512D-50-C	MYB-Y6ULX-HMI	-
myd-y6ull-gpmi-weim-without-exp.dtb	MYC-Y6ULY2-4E512D-50-I	MYB-Y6ULX-HMI	-

The MYD-Y6ULX-HMI Micro SD slot is connected to mmc0 controller.So, all dtb files enabled the mmc0 controller by default.

When you build kernel complete, the version tag will changed automatically.If you driver load with module type, you should recompile driver module.

```
make modules
```

After compile completation, it will be installed to specify path:

```
mkdir ../target-kernel
make INSTALL_MOD_PATH=../target-kernel modules_install
```

Then you can package the target-kernel directory and extract it into /lib directory on file system of MYD-Y6ULX-HMI board.

3.3 Build File System

The Linux platform has many open source tools to build filesystem.These tools has some features to help developer build filesystem more easier in system build or customize it.Recently, some are more populator Buildroot, Yocto, OpenEmbedded etc.The Yocto project support more powerful and system method to build a linux system to suit your product.

Yocto not only a build tool for file system, it also has full workflow to build and maintain under Linux. It makes platform developer and application developer working together under same framework. And resolve non-united and non-manage on the legacy develop way.

Yocto is an open source "umbrella" project.It means has more sub-projects.Yocto just containe all other projects and support an reference build system "Poky". The Poky project will guide developer how to use, build, embedded Linux system.It has Bitbake, OpenEmbedded-Core, BSP package and more kinds of software packages and config files.Through Poky to build different requirment system, eg: the minimal system core-image-minimal, include GUI system fsl-image-gui, include Qt5 graphics system fsl-image-qt5.

NXP i.MX6UL/i.MX6ULL support build file to apply on Yocto project.These files will build a customization system by NXP.We also provide config files to support MYD-Y6ULX-HMI series boards.This will help developer to build Linux system that can be programming to MYD-Y6ULX-HMI series boards.

Yocto has more rich development resource, help engineers to learn and customization the system. This document can't cover full usage on Yocto.We recommend developer to build system after reading these documents.

Yocto Project Quick start

Bitback User Manual

Yocto Project Reference Manual

Yocto Project Development Manual

Yocto Project Complete Documentation Set

i.MX Yocto Project User's Guide

3.3.1 Using Yocto build Linux system

This section suit for developer on customization file system.If you just want to use it, please use the prebuilt file system.

The Yocto needs download all third-party software packages from internet.In order to build more speedly, MYD-Y6ULX-HMI also support a full package, you desn't need download again.

Attention: Building Yocto does not use previous toolchain, please open new tab for shell or new terminal window.

MYD-Y6ULX-HMI supports full Yocto package

Extract Yocto source package, and extract the Yocto-downloads.tar.xz into Yocto source direcotry.The Yocto-downloads.tar.xz includes all packages when building MYD-Y6ULX-HMI from Yocto.

Attention: The Yocto-downloads.tar.xz file is more large, it is not included in MYD-Y6ULX-HMI iso file.Please visit and download it from http://d.myirtech.com/MYD-Y6ULX-HMI.

```
cd $DEV_ROOT
tar xvf 04-Source/fsl-release-yocto-hmi.tar.bz2
tar xvf 04-Source/Yocto-downloads.tar.xz -C fsl-release-yocto-hmi
```

Last, also needs put the kernel and u-boot source into your home directory in linux.It will be fetched by Yocto.

```
$tar xvf 04-Source/MYiR-iMX-Linux.tar.bz2 -C ~/
$tar xvf 04-Source/MYiR-iMX-uboot.tar.bz2 -C ~/
```

Init Yocto

Using script fsl-setup-release.sh,create a build directory, and Yocto will build all under it. The MACHINE parameter is "myd-y6ull-hmi"

```
$cd fsl-release-yocto-hmi
$DISTRO=myir-imx-fb MACHINE=myd-y6ull-hmi source fsl-setup-release.sh -b build
$tree conf/
conf/
├── bblayers.conf
├── bblayers.conf.org
├── local.conf
├── local.conf.org
├── local.conf.sample
├── sanity_info
└── templateconf.cfg
```

Build system image with Qt5 packages

The first build will cost more time, please take a coffee and wait it completion.

```
$bitbake fsl-image-qt5
```

Build system image with full command line packages

You doesn't need to modify any file, just let Yocto to build it.

```
$bitbake core-image-base
```

Image Name	Description	Used for
core-image-minimal	minimal file system	used for MYD-Y6ULX-HMI to minimal system
core-image-base	base file system has more command line feature	full commnand line system, no GUI
fsl-image-qt5	system use Qt5 as GUI	used for graphics requirment

After build process finished, it will output manifest file.This file has each package name and version be installed to target file system.

The first build process of Yocto will take more time, this depends on your PC cpu core number and RAM size. The Yocto recommend use eight core CPU and SSD hardware to impove build speed. Additionally, the Yocto will generate cache after first build, the next build process also save more time for you.

All output files are in "tmp/deploy/images/myd-y6ull-hmi/" directory after build complete. Below as example:

```
$ ls -lh tmp/deploy/images/myd-y6ull-hmi
总用量 1.1G
core-image-base-myd-y6ull-hmi-20181112072545.rootfs.ext4
core-image-base-myd-y6ull-hmi-20181112072545.rootfs.manifest
core-image-base-myd-y6ull-hmi-20181112072545.rootfs.sdcard
core-image-base-myd-y6ull-hmi-20181112072545.rootfs.tar.bz2
core-image-base-myd-y6ull-hmi-20181112072545.rootfs.tar.xz
core-image-base-myd-y6ull-hmi.ext4 -> core-image-base-myd-y6ull-hmi-20181112072545.rootfs.ext4
core-image-base-myd-y6ull-hmi.manifest -> core-image-base-myd-y6ull-hmi-20181112072545.rootfs.manifest
core-image-base-myd-y6ull-hmi.sdcard -> core-image-base-myd-y6ull-hmi-20181112072545.rootfs.sdcard
core-image-base-myd-y6ull-hmi.tar.bz2 -> core-image-base-myd-y6ull-hmi-20181112072545.rootfs.tar.bz2
core-image-base-myd-y6ull-hmi.tar.xz -> core-image-base-myd-y6ull-hmi-20181112072545.rootfs.tar.xz
core-image-minimal-myd-y6ull-hmi-20181112101358.rootfs.ext4
core-image-minimal-myd-y6ull-hmi-20181112101358.rootfs.manifest
core-image-minimal-myd-y6ull-hmi-20181112101358.rootfs.sdcard
core-image-minimal-myd-y6ull-hmi-20181112101358.rootfs.tar.bz2
core-image-minimal-myd-y6ull-hmi-20181112101358.rootfs.tar.xz
core-image-minimal-myd-y6ull-hmi.ext4 -> core-image-minimal-myd-y6ull-hmi-20181112101358.rootfs.ext4
core-image-minimal-myd-y6ull-hmi.manifest -> core-image-minimal-myd-y6ull-hmi-20181112101358.rootfs.manifest
core-image-minimal-myd-y6ull-hmi.sdcard -> core-image-minimal-myd-y6ull-hmi-20181112101358.rootfs.sdcard
core-image-minimal-myd-y6ull-hmi.tar.bz2 -> core-image-minimal-myd-y6ull-hmi-20181112101358.rootfs.tar.bz2
core-image-minimal-myd-y6ull-hmi.tar.xz -> core-image-minimal-myd-y6ull-hmi-20181112101358.rootfs.tar.xz
fsl-image-qt5-myd-y6ull-hmi-20181112071320.rootfs.ext4
fsl-image-qt5-myd-y6ull-hmi-20181112071320.rootfs.manifest
fsl-image-qt5-myd-y6ull-hmi-20181112071320.rootfs.sdcard
fsl-image-qt5-myd-y6ull-hmi-20181112071320.rootfs.tar.bz2
fsl-image-qt5-myd-y6ull-hmi-20181112071320.rootfs.tar.xz
fsl-image-qt5-myd-y6ull-hmi.ext4 -> fsl-image-qt5-myd-y6ull-hmi-20181112071320.rootfs.ext4
fsl-image-qt5-myd-y6ull-hmi.manifest -> fsl-image-qt5-myd-y6ull-hmi-20181112071320.rootfs.manifest
fsl-image-qt5-myd-y6ull-hmi.sdcard -> fsl-image-qt5-myd-y6ull-hmi-20181112071320.rootfs.sdcard
fsl-image-qt5-myd-y6ull-hmi.tar.bz2 -> fsl-image-qt5-myd-y6ull-hmi-20181112071320.rootfs.tar.bz2
fsl-image-qt5-myd-y6ull-hmi.tar.xz -> fsl-image-qt5-myd-y6ull-hmi-20181112071320.rootfs.tar.xz
u-boot-emmc-2016.03-r0.imx
u-boot.imx -> u-boot-sd-2016.03-r0.imx
u-boot.imx-emmc -> u-boot-emmc-2016.03-r0.imx
u-boot.imx-nand -> u-boot-nand-2016.03-r0.imx
u-boot.imx-sd -> u-boot-sd-2016.03-r0.imx
u-boot-myd-y6ull-hmi.imx -> u-boot-sd-2016.03-r0.imx
u-boot-myd-y6ull-hmi.imx-emmc -> u-boot-emmc-2016.03-r0.imx
u-boot-myd-y6ull-hmi.imx-nand -> u-boot-nand-2016.03-r0.imx
u-boot-myd-y6ull-hmi.imx-sd -> u-boot-sd-2016.03-r0.imx
u-boot-nand-2016.03-r0.imx
u-boot-sd-2016.03-r0.imx
```

Some files are link file in output files.Below is description:

File Name	Usage
*.rootfs.manifest	The list of system include packages
*.rootfs.ext4	File system package to EXT4 format file
*.rootfs.sdcard	Image can be write to SD card and boot from SD card
*.rootfs.tar.bz2	File system package to tar.bz2
*.rootfs.tar.xz	File system package to tar.xz

u-boot-emmc-2016.03-r0.imx	u-boot used for booting from eMMC
u-boot-nand-2016.03-r0.imx	u-boot used for booting from NAND
u-boot-sd-2016.03-r0.imx	u-boot used for booting SD card

Bitbake common usage

Bitbake arguments	Description
-c fetch	Download package from predefined of recipe
-c cleanall	Clean all build directory
-c deploy	Deploy image or package to target rootfs
-k	Continue when error occure
-c compile	Recompile image or package

3.3.2 Using Yocto build SDK package

Yocto supports SDK generating function. It used for low-level or application level to compile source code.You doesn't need to manually handle the dependency softwares or libraries.The SDK package has two different way, one is suit low-level deveop toolchain, used for compile u-boot and linux.Another used for application development, it contains header files and libraries on target sysroot. The developer will be more convenient to development program to target device. The two kinds SDK package use shell self-extra file, it will be installed under "/opt" directory.

Build low-level toolchain

```
bitbake meta-toolchain
```

The directory "tmp/deploy/sdk" has three files after build finish:

```
$ ls tmp/deploy/sdk/ -lh
myir-imx6ulx-fb-glibc-x86_64-meta-toolchain-cortexa7hf-neon-toolchain-4.1.15-2.0.1.host.manifest
myir-imx6ulx-fb-glibc-x86_64-meta-toolchain-cortexa7hf-neon-toolchain-4.1.15-2.0.1.sh
myir-imx6ulx-fb-glibc-x86_64-meta-toolchain-cortexa7hf-neon-toolchain-4.1.15-2.0.1.target.manifest
```

Here has two kinds manifest file, "host.manifest" is a list of host software packages, "target.manifest" is a list of target device packages.

Build application-level toolchain

The application-level toolchain use same name with Image.This case you can use "fsl-image-qt5" or "core-image-base" as image name argument.

```
bitbake -c populate_sdk <image name>
```

The directory "tmp/deploy/sdk/" has three files after build finish:

```
$ ls tmp/deploy/sdk/ -lh
myir-imx6ulx-fb-glibc-x86_64-fsl-image-qt5-cortexa7hf-neon-toolchain-4.1.15-2.0.1.host.manifest
myir-imx6ulx-fb-glibc-x86_64-fsl-image-qt5-cortexa7hf-neon-toolchain-4.1.15-2.0.1.sh
myir-imx6ulx-fb-glibc-x86_64-fsl-image-qt5-cortexa7hf-neon-toolchain-4.1.15-2.0.1.target.manifest
```

The "*.host.manifest" is a list of host install packages. The "*.target.manifest" is a list of target device installed packages.The file "myir-imx6ulx-fb-glibc-x86_64-fsl-image-qt5-cortexa7hf-neon-toolchain-4.1.15-2.0.1.sh" is SDK toolchain. It can be distributed and installed to other Linux system and compile program to target device.

4 Linux Application Development

The hardware peripherals and application examples of MYD-Y6ULX-HMI development board.

Before use, you need to Yocto SDK toolchain to compile all the example code, and copy to the development board directory.

Compile example program

Load the toolchain environment to current shell, and check the gcc version to verify environment correct.

```
$source /opt/myir-imx6ulx-fb/4.1.15-2.0.1/environment-setup-\
cortexa7hf-neon-poky-linux-gnueabi

$arm-poky-linux-gnueabi-gcc --version
$arm-poky-linux-gnueabi-gcc --version
arm-poky-linux-gnueabi-gcc (GCC) 5.3.0
Copyright (C) 2015 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

Compile the sample code:

```
$cd $DEV_ROOT/04-Sources
$tar xvf example.tar.bz2
$cd example
$make
```

4.1 Test LCD

This example demonstrates the operation of the FrameBuffer of Linux, enabling color and color grid testing.You need connect the LCD to MYD-Y6ULX-HMI board LCD interface(J9).

We have two kinds LCD with touch panel. The 7-inch capacitive screen is configured by default.

The LCD screen will display the corresponding color, the following is the terminal output information:

```
./framebuffer_test
The framebuffer device was opened successfully.
vinfo.xres=480
vinfo.yres=272
vinfo.bits_per_bits=16
vinfo.xoffset=0
vinfo.yoffset=0
red.offset=11
green.offset=5
blue.offset=0
transp.offset=0
finfo.line_length=960
finfo.type = PACKED_PIXELS
The framebuffer device was mapped to memory successfully.
color: red   rgb_val: 0000F800
color: green   rgb_val: 000007E0
color: blue   rgb_val: 0000001F
color: r & g   rgb_val: 0000FFE0
color: g & b   rgb_val: 000007FF
color: r & b   rgb_val: 0000F81F
color: white   rgb_val: 0000FFFF
color: black   rgb_val: 00000000
```

The Linux source of MYD-Y6ULX-HMI series board has already supports the display and touch functions of the two modules. The touch function of the resistive screen and the capacitive screen is different. The resistive screen is collected by the ADC, and the capacitive screen is read by I2C communication. Information, dts code has been configured, you only need to enable the corresponding function.

When using the resistive screen, modify the status attribute of tsc to okay. Edit the corresponding devicetree : i.MX6ULL："arch/arm/boot/dts/myb-y6ull-14x14.dts" i.MX6UL："arch/arm/boot/dts/myb-y6ul-14x14.dts " modify the status property of tsc node to okay value.

```
&tsc {
     pinctrl-names = "default";
     pinctrl-0 = <&pinctrl_tsc>;
     xnur-gpio = <&gpio1 3 GPIO_ACTIVE_LOW>;
     measure-delay-time = <0xfffff>;
     pre-charge-time = <0xffff>;
     status = "okay";
};
```

4.2 Test TouchPanel

This example shows you how to test the touch panel on LCD screen module. The MYD-Y6ULX-HMI supports 7-inch capacitive screen and 7-inch resistive screen, the capacitive screen does not need to be calibrated, and the capacitive screen needs to be calibrated.

You can use ts_calibrate and ts_test command to test your LCD and touch panel are working.The "TSLIB_TSDEVICE" point the touch device node, capacitive and resistive has different device node.

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

# ts_test
```

4.3 Ethernet

This example uses the TCP/IP socket API to implement a simple C/S structure of the program. Copy the executable program arm_client to the development board, pc_server copy to the PC, the development board and PC access network.

The MYD-Y6ULX-HMI has two ethernet interfaces, CN1 and CN2.

Configure IP of PC machine and run server program:

```
$ sudo ifconfig eth0 192.168.1.111
$ ./pc_server
REC FROM: 192.168.1.222
```

Configure IP of CN2 and run client program on board:

```
# ifconfig eth0 192.168.1.222
# ./arm_client 192.168.1.111
from server: Make Your idea Real!
```

4.4 Test USB Host

Connect the USB flash disk to USB HOST(J6) interface, will output detection device information. At the same time,the system will mount the sotrage device ,then you are able to write and read.

```
# usb 1-2: USB disconnect, device number 6
usb 1-2: new high-speed USB device number 7 using atmel-ehci
usb 1-2: New USB device found, idVendor=0bda, idProduct=0316
usb 1-2: New USB device strings: Mfr=1, Product=2,
SerialNumber=3
usb 1-2: Product: USB3.0-CRW
usb 1-2: Manufacturer: Generic
usb 1-2: SerialNumber: 20120501030900000
usb-storage 1-2:1.0: USB Mass Storage device detected
scsi host5: usb-storage 1-2:1.0
scsi 5:0:0:0: Direct-Access     Generic- SD/MMC
1.00 PQ: 0 ANSI: 4
sd 5:0:0:0: [sda] 31116288 512-byte logical blocks: (15.9 GB/
14.8 GiB)
sd 5:0:0:0: [sda] Write Protect is off
sd 5:0:0:0: [sda] Write cache: disabled, read cache: enabled,
doesn't support DPO or FUA
 sda: sda1 sda2
 sd 5:0:0:0: [sda] Attached SCSI removable disk

# mount /dev/sda1 /mnt/
# echo "hello" > /mnt/hello.txt
# cat /mnt/hello.txt
hello
```

4.5 Test USB Device

This example shows how to use USB device mode through the Micro USB interface(J7) on board.It will attach a specified file or device as a Gadget device. It works as a storage device connected to the HOST device.

- Operation steps on board:

```
#mkfs.vfat /dev/ram1
#modprobe g_mass_storage file=/dev/ram1 removable=1 \
iSerialNumber="1234"

[ 3048.950498] Mass Storage Function, version: 2009/09/11
[ 3048.982245] LUN: removable file: (no medium)
[ 3048.997849] LUN: removable file: /dev/ram1
[ 3049.000674] Number of LUNs=1
[ 3049.002272] Number of LUNs=1
[ 3049.023990] g_mass_storage gadget: Mass Storage Gadget,
version: 2009/09/11
[ 3049.029682] g_mass_storage gadget: g_mass_storage ready
[ 3094.766373] g_mass_storage gadget: high-speed config #1:
Linux File-Backed Storage
```

- The host PC display an USB device connected and SerialNumber is "1234":

```
#dmesg | tail -n 20
[2872436.778616] usb 1-1: USB disconnect, device number 102
[2872436.779156] sd 3:0:0:0: [sdb] Synchronizing SCSI cache
[2872436.779201] sd 3:0:0:0: [sdb] Synchronize Cache(10) failed:
Result: hostbyte=DID_NO_CONNECT driverbyte=DRIVER_OK
[2872442.508567] usb 1-1: new high-speed USB device number 103
using xhci_hcd
[2872442.650549] usb 1-1: New USB device found, idVendor=0525,
idProduct=a4a5
[2872442.650551] usb 1-1: New USB device strings: Mfr=3,
Product=4, SerialNumber=5
[2872442.650552] usb 1-1: Product: Mass Storage Gadget
[2872442.650553] usb 1-1: Manufacturer: Linux 4.1.15-1.2.0+g8d98
da6 with 2184000.usb
[2872442.650554] usb 1-1: SerialNumber: 1234
[2872442.657827] usb-storage 1-1:1.0: USB Mass Storage device
detected
[2872442.657895] usb-storage 1-1:1.0: Quirks match for vid 0525
pid a4a5: 10000
[2872442.657923] scsi host3: usb-storage 1-1:1.0
[2872443.669426] scsi 3:0:0:0: Direct-Access     Linux    File-
Stor Gadget 0401 PQ: 0 ANSI: 2
[2872443.669886] sd 3:0:0:0: Attached scsi generic sg1 type 0
[2872443.670820] sd 3:0:0:0: [sdb] 131072 512-byte logical
blocks: (67.1 MB/64.0 MiB)
[2872443.779976] sd 3:0:0:0: [sdb] Write Protect is off
[2872443.779979] sd 3:0:0:0: [sdb] Mode Sense: 0f 00 00 00
[2872443.890093] sd 3:0:0:0: [sdb] Write cache: enabled,
read cache: enabled, doesn't support DPO or FUA
[2872444.110372]  sdb:
[2872444.330074] sd 3:0:0:0: [sdb] Attached SCSI removable disk
```

4.6 Test RS485

This example demonstrates how to use the Linux serial API to configure the RS485 on the development board to send and receive data. For details, refer to the source code.

Hardware

MYD-Y6ULX-HMI board is equipped with a RS485 interface(J8.3 for RS485-A, J8.4 for RS485-B).You need to the A，B signal wire to another RS485 device or USB to RS485 converter.

Software

Copy and run the program on MYD-Y6ULX-HMI Linux system.Below is MYD-Y6ULX-HMI as the sender:

```
# ./rs485_write -d /dev/ttymxc3 -b 4800 -e 1
SEND[20]: 0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08 0x09 0x0a 0x0b 0x0c 0x0d 0x0e 0x0f 0x10 0x11 0x12 0x13 0x14
SEND[20]: 0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08 0x09 0x0a 0x0b 0x0c 0x0d 0x0e 0x0f 0x10 0x11 0x12 0x13 0x14
SEND[20]: 0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08 0x09 0x0a 0x0b 0x0c 0x0d 0x0e 0x0f 0x10 0x11 0x12 0x13 0x14
SEND[20]: 0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08 0x09 0x0a 0x0b 0x0c 0x0d 0x0e 0x0f 0x10 0x11 0x12 0x13 0x14
```

Other device as recevier：

```
# ./rs485_read -d /dev/ttymxc3 -b 4800 -e 1
RECV[20]: 0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08 0x09 0x0a 0x0b 0x0c 0x0d 0x0e 0x0f 0x10 0x11 0x12 0x13 0x14
RECV[20]: 0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08 0x09 0x0a 0x0b 0x0c 0x0d 0x0e 0x0f 0x10 0x11 0x12 0x13 0x14
RECV[20]: 0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08 0x09 0x0a 0x0b 0x0c 0x0d 0x0e 0x0f 0x10 0x11 0x12 0x13 0x14
RECV[20]: 0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08 0x09 0x0a 0x0b 0x0c 0x0d 0x0e 0x0f 0x10 0x11 0x12 0x13 0x14
```

4.7 Test RS232

This example demonstrates how to use the Linux serial API to configure the RS232 on the development board to send and receive data. For details, refer to the source code.

Hardware

MYD-Y6ULX-HMI board is equipped with a RS232 interface(J8.5 for RS232-TX, J8.6 for RS232-RX).You need to the TXD,RXD signal wire to another RS232 device or USB to RS232 converter.

Software

Copy and run the program on MYD-Y6ULX-HMI Linux system.Below is MYD-Y6ULX-HMI as the sender:

```
# ./uart_test -d /dev/ttymxc1 -b 115200
/dev/ttymxc1 RECV 10 total
/dev/ttymxc1  RECV: 1234567890
/dev/ttymxc1  RECV 10 total
/dev/ttymxc1  RECV: 1234567890
/dev/ttymxc1  RECV 10 total
/dev/ttymxc1  RECV: 1234567890
/dev/ttymxc1  RECV 10 total
/dev/ttymxc1  RECV: 1234567890
```

4.8 Test Camera

MYD-Y6ULX-HMI board has an 8-bit parallel camera interface(J4).It can connects camera module of MY-CAM011B model.The module and board connects with FPC wire.

- Attention: Please do not insert other camera model, this operation maybe damage the board or camera module.

This example program use an open source software uvc_stream.It supports show video in web from camera capture.

Hardware

Use FPC wire connects MYB-CAM011B module and camera interface J4 of MYD-Y6ULX-HMI.

Software

The uvc_stream uses network show video data.You need setup ethernet IP address of MYD-Y6ULX-HMI, the correspond device is eth1. Uses "v4l2-ctl" command to query the device node of MY-CAM011B on Linux system.It outputs information about video device.The "i.MX6S_CSI" string is camera controller and correspond string "/dev/video1" is device node of MY-CAM011B module. The uvc_stream parameter '-y' means use yuyv type, the '-P' means setting password of web page. The '-r' means define resolution.The uvc_stream default username is uvc_user.

```
# ifconfig eth1 192.168.1.42
# v4l2-ctl --list-devices

    i.MX6S_CSI (platform:21c4000.csi):
    /dev/video1

    pxp (pxp_v4l2):
        /dev/video0

# ./uvc_stream -d /dev/video1 -y -P 123456
```

The uvc_stream program supports two kinds web functions, snapshot and streaming. The snapshot function request URL is snapshot.jpeg, and streaming function request URL is stream.mjpeg.

Let PC and board has same network, open your browser, visit http://192.168.1.42:8080/stream.mjpeg.After enter, you can see the login window, login with uvc_user, 123456.Now, you can see video stream from web on MY-CAM011B captured.

4.9 Test Audio

- You will need the MYB-Y6ULX-HMI-4GEXP IO Board to complete the testing of Audio function

This example demonstrates the development onboard audio interface using the arecord/aplay program in Linux systems.

Hareware

- Connect LINE IN(J2) interface on the MYB-Y6ULX-HMI board, PC Audio-Out via a 3.5mm AUX cable
- HEADPHONE(J1) connect your headerphone or speaker

Software

The PC plays the audio file and execute arecord command on board.It will recored data and save to test.wav file.You can use "ctrl + c" stop it after one minute.

```
# arecord -f cd test.wav
```

Use aplay command to play file that previous step recoreded.

```
# aplay test.wav
```

4.10 Test WiFi

- You will need the MYB-Y6ULX-HMI-4GEXP IO Board to complete the testing of WiFi function.
- The Wifi funciton are only available in NandFlash CPU moudle,beacuse the re-use of the SDIO signal. For EMMC version CPU module,WiFi function can not be used.

The MYD-Y6ULX-HMI board has a WiFi module (U7).It's supports Client and AP mode.

Hardware Connective

Use SMA interface of wireless antenna connect with J8 position of board.

Client Mode

The Client mode means WiFi module as client device connect to your route or other AccessPoint device.

Our Linux prebuilt system has added driver of WiFi module.It will be auto loaded when system startup. And also use lsmod to confirm it.The wlan0 network device has exist when driver loaded success.The ifconfig command can be used confirm it.

```
# ifconfig wlan0
wlan0     Link encap:Ethernet  HWaddr a0:2c:36:60:ee:e0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:3388 errors:0 dropped:10 overruns:0 frame:0
          TX packets:37 errors:0 dropped:3 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:395459 (386.1 KiB)  TX bytes:6074 (5.9 KiB)
```

The following is a link between the WiFi module and the WiFi hotspot, setting the username and password of the WiFi hotspot, /etc/wifi-conf/WIFI.CONF is an example given, modified according to your actual situation. After the modification is completed, Connection command:

```
# /etc/wifi-conf/ifup_wifi_sta
```

After the connection is successful, the IP will be obtained automatically, and the connected and assigned IP will be viewed.

```
~# iwconfig
wlan0     IEEE 802.11  ESSID:"MYIR_TECH"
          Mode:Managed  Frequency:2.437 GHz  Access Point: 30:FC:68:9A:E8:99
          Bit Rate=6.5 Mb/s   Tx-Power:32 dBm
          Retry min limit:10   RTS thr:off   Fragment thr:off
          Power Managementmode:All packets received
          Link Quality=2/5  Signal level=-74 dBm  Noise level=-91 dBm
          Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
          Tx excessive retries:0  Invalid misc:0   Missed beacon:0

sit0      no wireless extensions.

lo        no wireless extensions.

eth0      no wireless extensions.

eth1      no wireless extensions.

# ifconfig wlan0
wlan0     Link encap:Ethernet  HWaddr 28:ed:e0:7b:99:01
          inet addr:192.168.40.103  Bcast:192.168.40.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:58 errors:0 dropped:0 overruns:0 frame:0
          TX packets:49 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:2989 (2.9 KiB)  TX bytes:10804 (10.5 KiB)
```

AP Mode

AccessPoint mode needs software and hardware to suporrt AP feature. It use hostapd to support AP function.

The information about the AP is configured in udhcpd.conf and hostapd.conf, and can be modified by the user. Turn on hotspots:

```
# /etc/wifi-conf/ifup_wifi_ap
```

After turning on the hotspot, you can search on the phone, the SSID is MYIR-WIFI-AP; the connection password is MYIR-TECH. View related information:

```
# iwconfig
wlan0     IEEE 802.11  ESSID:"MYIR-WIFI-AP"
          Mode:Master  Frequency:2.437 GHz  Access Point: 28:ED:E0:7B:99:01
          Bit Rate=96 Mb/s   Tx-Power:32 dBm
          Retry min limit:10   RTS thr:off   Fragment thr:off
          Encryption key:off
          Power Management:off

# ifconfig wlan0
wlan0     Link encap:Ethernet  HWaddr 28:ed:e0:7b:99:01
          inet addr:192.168.10.1  Bcast:192.168.10.255  Mask:255.255.255.0
          inet6 addr: fe80::2aed:e0ff:fe7b:9901/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:40 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:7476 (7.3 KiB)
```

4.11 Test 4G LTE

- You will need the MYB-Y6ULX-HMI-4GEXP IO Board to complete the testing of 4G function
- You will need to make some change on USB driver if you need to use other 4G modules,but the current EC20 driver will be a good referen

The MYD-Y6ULX-HMI board support LTE module through MINI PCI-E slot with USB interface.Currently, the MYD-Y6ULX-HMI board provides support for Quectl's EC20 model.

Hardware

- Install Quectl EC20 module into PCI-E slot(U4).
- Use I-PEX interface wire connect LTE module and J3 position of board.
- Install SMA wireless antenna to J4 position.

Software

Our Linux prebuilt system has added driver of 4G module.It will be auto loaded when system startup. And also use ls to confirm it.

```
# ls /dev/ttyUSB*
/dev/ttyUSB0 /dev/ttyUSB1 /dev/ttyUSB2 /dev/ttyUSB3 /dev/ttyUSB4
```

The Linux system of MYD-Y6ULX-HMI series board has provided ppp package.You can just enable ppp0 device, it will auto dial-up.

```
# ifup ppp0
# ifconfig ppp0
ppp0      Link encap:Point-to-Point Protocol
          inet addr:10.163.130.65  P-t-P:10.64.64.64  Mask:255.255.255.255
          UP POINTOPOINT RUNNING NOARP MULTICAST  MTU:1500  Metric:1
          RX packets:5 errors:0 dropped:0 overruns:0 frame:0
          TX packets:5 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:3
          RX bytes:62 (62.0 B)  TX bytes:86 (86.0 B)

# cat /etc/resolv.conf
nameserver 202.96.128.86
nameserver 202.96.134.133
```

Using the ping command to test whether the module is connected to the 4G network.

```
# ping myirtech.com
PING myirtech.com (50.6.151.71) 56(84) bytes of data.
64 bytes from 118.123.18.103: icmp_seq=1 ttl=117 time=80.5 ms
64 bytes from 118.123.18.103: icmp_seq=2 ttl=117 time=179 ms
64 bytes from 118.123.18.103: icmp_seq=3 ttl=117 time=378 ms
64 bytes from 118.123.18.103: icmp_seq=4 ttl=117 time=118 ms
64 bytes from 118.123.18.103: icmp_seq=5 ttl=117 time=122 ms
64 bytes from 118.123.18.103: icmp_seq=6 ttl=117 time=177 ms
```

If you encounter any issue on above steps, please check the log to find the reason.

```
# cat /var/log/quectel-dial.log
```

5 QT application development

Qt is the faster, smarter way to create innovative devices, modern UIs & applications for multiple screens. Cross-platform software development at its best. The MYD-Y6ULX-HMI uses Qt 5.6.2 version. In Qt application development, it is recommended to use QtCreator IDE.It can be developed Qt application more easier, automated cross-compiler for the development board of the ARM architecture.

This chapter uses Yocto SDK as cross compile tool to work with QtCreator to quickly develop graphical applications. Before starting this chapter, please complete the Chapter3 to build Qt, get an available ARM version of the Qt graphics library.

Please install the Yocto application-level SDK before you start it.

5.1 Install QtCreator

QtCreator installation package is a binary program, can be directly installed to your host PC.

```
$ cd $DEV_ROOT/04-Sources
$ 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
```

When the installation process is done, click on the next step to complete. The default installation directory is in the "/opt/qtcreator-4.1.0".

In order to QtCreator use Yocto SDK, we need add environment to QtCreator, modify the file "/opt/qtcreator-4.1.0/bin/qtcreator.sh". Add below command before the line "#! /bin/sh":

- Attention: When the author installs QT's cross-compilation toolchain, the directory name is modified (/opt/qt5-hmi). When selecting the toolchain, the directory name you installed is correct.

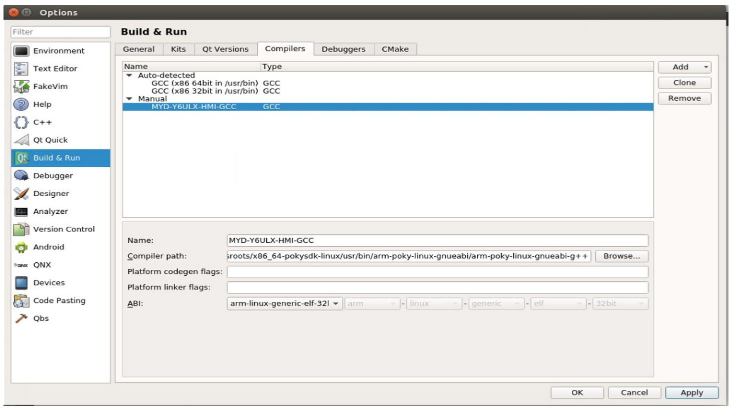
```
$ vi /opt/qtcreator-4.1.0/bin/qtcreator.sh
source /opt/qt5-hmi/environment-setup-cortexa7hf-neon-poky-linux-gnueabi
#! /bin/sh

# Use this script if you add paths to LD_LIBRARY_PATH
# that contain libraries that conflict with the
# libraries that Qt Creator depends on.
```

When you use QtCreator, you need start it from terminal to execute "qtcreator.sh".

```
/opt/qtcreator-4.1.0/bin/qtcreator.sh &
```

5.2 Configure QtCreator

The first step, run QtCreator, followed by "Tool" -> "Options", the Options dialog box appears, click "Build & Run" on the left, right select "Compilers" label. Click on the right "Add" button, pop-up drop-down list, select "GCC", fill the following input boxes, "Name" is "MYD-Y6ULX-HMI-GCC", click "Compiler path" beside "Browse.." button to choose "arm-poky-linux-gnueabi-g++" file path. In this case, the path is "/opt/qt5-hmi/sysroots/x86_64-pokysdk-linux/usr/bin/arm-poky-linux-gnueabi/arm-poky-linux-gnueabi-g++".When fill are complete, click "Apply".

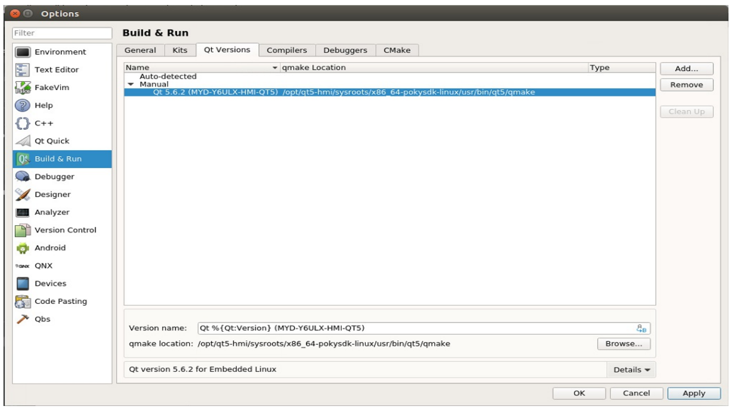


Name	Type
Auto-detected	
GCC (x86 64bit in /usr/bin)	GCC
GCC (x86 32bit in /usr/bin)	GCC
Manual	
MYD-Y6ULX-HMI-GCC	GCC

Name: MYD-Y6ULX-HMI-GCC
Compiler path: sroots/x86_64-pokysdk-linux/usr/bin/arm-poky-linux-gnueabi/arm-poky-linux-gnueabi-g++
Platform codegen flags:
Platform linker flags:
ABI: arm-linux-generic-elf-32l arm - linux - generic - elf - 32bit

Figure5-2-1 Config Compiler

The second step, and then select the "Qt Version" tab, click the right side of the "Add ...", will pop up qmake path selection dialog box.In this case, qmake file path is "/opt/qt5-hmi/sysroots/x86_64-pokysdk-linux/usr/bin/qt5", and choose "qmake" file. click the "Open" button then change "Version name" is "Qt %{Qt:Version} (MYDY6ULX-HMI-QT5)".After that click "Apply" button.

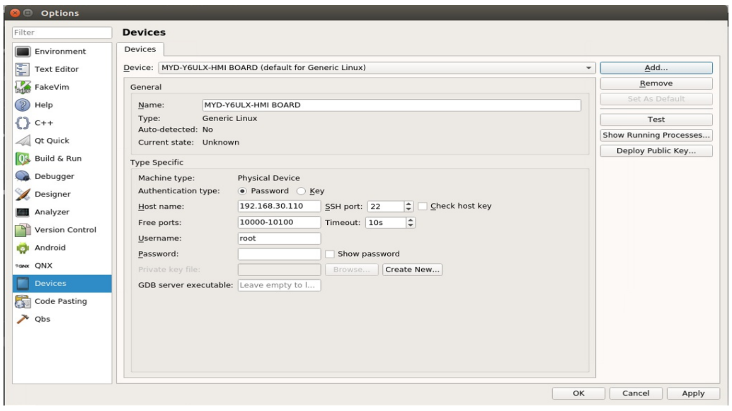


Name	qmake Location	Type
Auto-detected		
Manual		
Qt 5.6.2 (MYD-Y6ULX-HMI-QT5)	/opt/qt5-hmi/sysroots/x86_64-pokysdk-linux/usr/bin/qt5/qmake	

Version name: Qt %{Qt:Version} (MYD-Y6ULX-HMI-QT5)
qmake location: /opt/qt5-hmi/sysroots/x86_64-pokysdk-linux/usr/bin/qt5/qmake
Qt version 5.6.2 for Embedded Linux

Figure5-2-2 Configure Qt version

The third step, click the "Device" menu on left panel, and click "Add..." button on right panel.Fill those input box, "Name" is "MYD-Y6ULX-HMI BOARD", "Host name" is IP address of target board(alsa fill any one), "Username" is "root".Then clock "Apply" button.



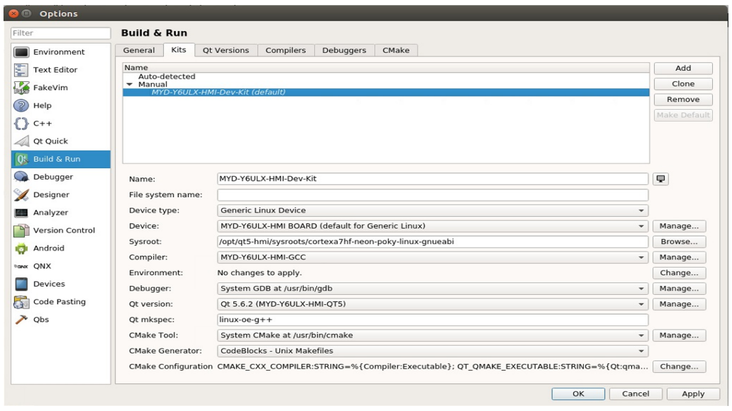
Device: MYD-Y6ULX-HMI BOARD (default for Generic Linux)

General
Name: MYD-Y6ULX-HMI BOARD
Type: Generic Linux
Auto-detected: No
Current state: Unknown

Type Specific
Machine type: Physical Device
Authentication type: [✓ Password] Key
Host name: 192.168.30.110 SSH port: 22 Check host key
Free ports: 10000-10100 Timeout: 10s
Username: root
Password: Show password
Private key file:
GDB server executable: Leave empty to l...

Figure5-2-3 Configure Qt version

The fourth step, click "Build & Run" menu on left panel will back to "Kit" tab in right panel.The content fill with "Name" is "MYD-Y6UKX-HMI-Dev-kit", "Device" choose "MYD-Y6ULX-HMI BOARD" option."Sysroot" choose the sysroot of target board.In this case , use "/opt/myir-imx6ulx-fb/4.1.15-2.0.1/sysroots/cortexa7hf-neon-poky-linux-gnueabi"."Compiler" choose "MYD-Y6ULX-HMI-GCC" before we configured."Qt version" choose "Qt 5.6.2 (MYD-Y6ULX-HMI-QT5)" before configured, "Qt mkspec" is "linux-oe-g++".Other use default option, then click"Apply" and "OK" button.



Build & Run

Name
Auto-detected
Manual
MYD-Y6ULX-HMI-Dev-Kit (default)

Name: MYD-Y6ULX-HMI-Dev-Kit
File system name:
Device type: Generic Linux Device
Device: MYD-Y6ULX-HMI BOARD (default for Generic Linux)
Sysroot: /opt/qt5-hmi/sysroots/cortexa7hf-neon-poky-linux-gnueabi
Compiler: MYD-Y6ULX-HMI-GCC
Environment: No changes to apply.
Debugger: System GDB at /usr/bin/gdb
Qt version: Qt 5.6.2 (MYD-Y6ULX-HMI-QT5)
Qt mkspec: linux-oe-g++
CMake Tool: System CMake at /usr/bin/cmake
CMake Generator: CodeBlocks - Unix Makefiles
CMake Configuration CMAKE_CXX_COMPILER:STRING=%{Compiler:Executable}; QT_QMAKE_EXECUTABLE:STRING=%{Qt:qma...

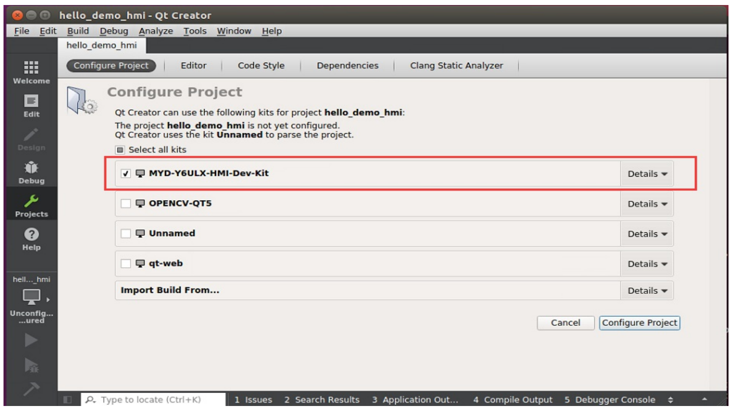
Figure5-2-4 Configure Kit

5.3 Test Qt application

In order to test previous configure is correct, we support a Qt example.You just open, config and compile it.

The first step, in the menu bar, select "File" -> "Open File or Project", in the open dialog box, browse and select "hellowrold" example project, choose "hello_demo_hmi.pro" file, click "Open" button.

The second step,choose "MYD-Y6ULX-HMI-Dev-Kit" option.Then the "hello_demo_hmi" project will use "MYD-Y6ULX-HMI-Dev-Kit" option to build it.


hello_demo_hmi - Qt Creator
File Edit Build Debug Analyze Tools Window Help
hello_demo_hmi
Configure Project | Editor | Code Style | Dependencies | Clang Static Analyzer
Welcome
Edit
Design
Debug
Projects
Help
hell..._hmi
Unconfig...ured
Details
Cancel
Configure Project
Type to locate (Ctrl+K)
1 Issues 2 Search Results 3 Application Out... 4 Compile Output 5 Debugger Console

Configure Project

Qt Creator can use the following kits for project **hello_demo_hmi**:

The project **hello_demo_hmi** is not yet configured.
Qt Creator uses the kit **Unnamed** to parse the project.

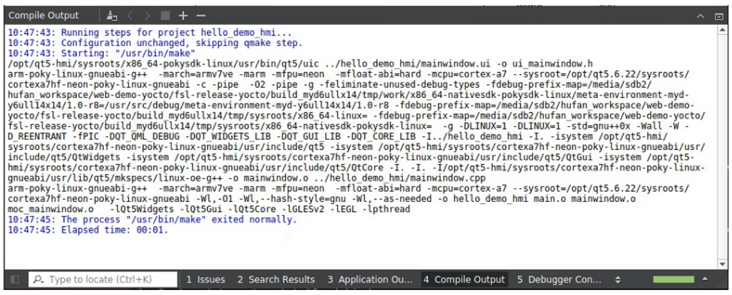
Select all kits

- [✓] MYD-Y6ULX-HMI-Dev-Kit
- [ ] OPENCV-QT5
- [ ] Unnamed
- [ ] qt-web

Import Build From...

Figure5-3-1 Config building option

Step 3, click the menu bar "Build" -> "Build Project hello_demo_hmi" button, you can complete the project compilation, while the bottom window will output compile process.


Compile Output
Type to locate (Ctrl+K)
1 Issues 2 Search Results 3 Application Ou... 4 Compile Output 5 Debugger Con...

```
10:47:43: Running steps for project hello_demo_hmi...
10:47:43: Configuration unchanged, skipping qmake step.
10:47:43: Starting: "/usr/bin/make"
/opt/qt5-hmi/sysroots/x86_64-pokysdk-linux/usr/bin/qt5/uic ../hello_demo_hmi/mainwindow.ui -o ui_mainwindow.h
arm-poky-linux-gnueabi-g++  -march=armv7ve -marm -mfpu=neon  -mfloat-abi=hard -mcpu=cortex-a7 --sysroot=/opt/qt5.6.22/sysroots/
cortexa7hf-neon-poky-linux-gnueabi -c -pipe  -O2 -pipe -g -feliminate-unused-debug-types -fdebug-prefix-map=/media/sdb2/
hufan_workspace/web-demo-yocto/fsl-release-yocto/build_myd6ullx14/tmp/work/x86_64-nativesdk-pokysdk-linux/meta-environment-myd-
y6ull14x14/1.0-r8=/usr/src/debug/meta-environment-myd-y6ull14x14/1.0-r8 -fdebug-prefix-map=/media/sdb2/hufan_workspace/web-demo-
yocto/fsl-release-yocto/build_myd6ullx14/tmp/sysroots/x86_64-linux= -fdebug-prefix-map=/media/sdb2/hufan_workspace/web-demo-yocto/
fsl-release-yocto/build_myd6ullx14/tmp/sysroots/x86_64-nativesdk-pokysdk-linux=  -g -DLINUX=1 -DLINUX=1 -std=gnu++0x -Wall -W -
D_REENTRANT -fPIC -DQT_QML_DEBUG -DQT_WIDGETS_LIB -DQT_GUI_LIB -DQT_CORE_LIB -I../hello_demo_hmi -I. -isystem /opt/qt5-hmi/
sysroots/cortexa7hf-neon-poky-linux-gnueabi/usr/include/qt5 -isystem /opt/qt5-hmi/sysroots/cortexa7hf-neon-poky-linux-gnueabi/usr/
include/qt5/QtWidgets -isystem /opt/qt5-hmi/sysroots/cortexa7hf-neon-poky-linux-gnueabi/usr/include/qt5/QtGui -isystem /opt/qt5-
hmi/sysroots/cortexa7hf-neon-poky-linux-gnueabi/usr/include/qt5/QtCore -I. -I. -I/opt/qt5-hmi/sysroots/cortexa7hf-neon-poky-linux-
gnueabi/usr/lib/qt5/mkspecs/linux-oe-g++ -o mainwindow.o ../hello_demo_hmi/mainwindow.cpp
arm-poky-linux-gnueabi-g++  -march=armv7ve -marm -mfpu=neon  -mfloat-abi=hard -mcpu=cortex-a7 --sysroot=/opt/qt5.6.22/sysroots/
cortexa7hf-neon-poky-linux-gnueabi -Wl,-O1 -Wl,--hash-style=gnu -Wl,--as-needed -o hello_demo_hmi main.o mainwindow.o
moc_mainwindow.o   -lQt5Widgets -lQt5Gui -lQt5Core -lGLESv2 -lEGL -lpthread
10:47:45: The process "/usr/bin/make" exited normally.
10:47:45: Elapsed time: 00:01.
```

Figure5-3-2 Compling output option

QtCreator build hello_demo_hmi project, compiled binary files stored in the "~/build-hello_demo_hmi-MYD_Y6ULX_HMI_Dev_Kit-Debug/" directory, you can use the file command to see whether is the compiler for the ARM architecture.

```
$ file ~/build-hello_demo_hmi-MYD_Y6ULX_HMI_Dev_Kit-Debug/hello_demo_hmi
/home/kevinchen/build-hello_demo_hmi-MYD_Y6ULX_HMI_Dev_Kit-Debug/
hello_demo_hmi: ELF 32-bit LSB executable, ARM, EABI5 version 1
(GNU/Linux), dynamically linked, interpreter /lib/ld-linux-
armhf.so.3, for GNU/Linux 2.6.32, BuildID[sha1]=
9c5f22deb1d8272c2a81528c171d215896112784, not stripped
```

Copy the hello_demo_hmi file to board and run it.

```
# ./hello_demo_hmi -platform linuxfb
```

The LCD shows Qt windows of "HELLO MYIR" string。

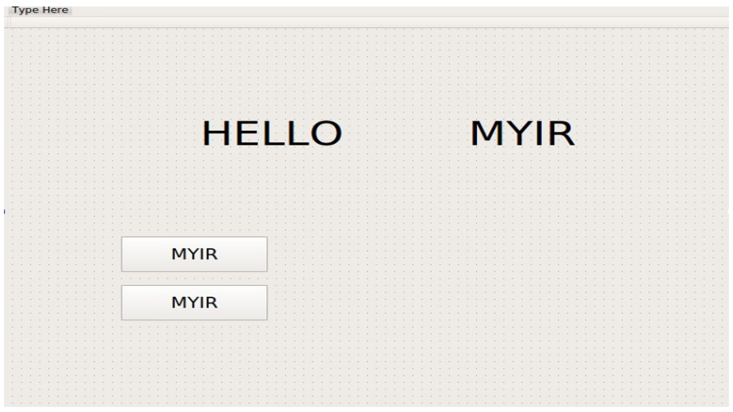


Figure5-3-3 Run example program

6 System update

MYD-Y6ULX-HMI series provides a way to update the system to the board and update the NAND flash to the SD card.

- SD Card method: Using updatable SD image to write files into flash.

6.1 SD Card Update

The sdcard image file needs special tool to write Micro SD storage card.The linux user can directly use dd command.The windows user need "Win32ImageWriter" tool.

MYD-Y6ULX-HMI series boards provide two sdcard file.It's path in directory 02-Images.

File name	Core board	Base board	Extension board	File system
myd-y6ull-hmi-update-emmc-ddr512m-core-20181225221233-exp.rootfs.sdcard.img.gz	MYC-Y6ULY2-256N256D-50-C 或 MYC-Y6ULY2-256N256D-50-I	MYB-Y6ULX-HMI	MYB-Y6ULX-HMI-4GEXP	core-image-base
myd-y6ull-hmi-update-emmc-ddr512m-core-20181225221241-without_exp.rootfs.sdcard.img.gz	MYC-Y6ULY2-256N256D-50-C 或 MYC-Y6ULY2-256N256D-50-I	MYB-Y6ULX-HMI	无	core-image-base
myd-y6ull-hmi-update-emmc-ddr512m-qt-20181225221216-exp.rootfs.sdcard.img.gz	MYC-Y6ULY2-256N256D-50-C 或 MYC-Y6ULY2-256N256D-50-I	MYB-Y6ULX-HMI	MYB-Y6ULX-HMI-4GEXP	fsl-image-qt5
myd-y6ull-hmi-update-emmc-ddr512m-qt-20181225221225-without_exp.rootfs.sdcard.img.gz	MYC-Y6ULY2-256N256D-50-C 或 MYC-Y6ULY2-256N256D-50-I	MYB-Y6ULX-HMI	无	fsl-image-qt5
myd-y6ull-hmi-update-nand-ddr256m-core-20181225221201-exp.rootfs.sdcard.img.gz	MYC-Y6ULY2-256N256D-50-C 或 MYC-Y6ULY2-256N256D-50-I	MYB-Y6ULX-HMI	MYB-Y6ULX-HMI-4GEXP	core-image-base
myd-y6ull-hmi-update-nand-ddr256m-core-20181225221209-without_exp.rootfs.sdcard.img.gz	MYC-Y6ULY2-256N256D-50-C 或 MYC-Y6ULY2-256N256D-50-I	MYB-Y6ULX-HMI	MYB-Y6ULX-HMI-4GEXP	core-image-base
myd-y6ull-hmi-update-nand-ddr256m-qt-20181225221245-exp.rootfs.sdcard.img.gz	MYC-Y6ULY2-256N256D-50-C 或 MYC-Y6ULY2-256N256D-50-I	MYB-Y6ULX-HMI	MYB-Y6ULX-HMI-4GEXP	core-image-base
myd-y6ull-hmi-update-nand-ddr256m-qt-20181225221253-without_exp.rootfs.sdcard.img.gz	MYC-Y6ULY2-256N256D-50-C 或 MYC-Y6ULY2-256N256D-50-I	MYB-Y6ULX-HMI	MYB-Y6ULX-HMI-4GEXP	core-image-base

Attention: The date string tag is generated by the tool on file name *.rootfs.sdcard.Please be based on actual.

Build updatable SD Card system image

If you modify the Linux kernel, U-Boot or Yocto, then you need a tool for update those files into the board. The MYD-Y6ULX-HMI board support a tool MYiR-iMX-mkupdate-sdcard-HMI that builds updatable SD Card image.It locates in '04-Tools/ManufactoryTool' directory.

The build-sdcard.sh script used to generate a system image that contains update system and update target files. The firmware directory used for the system of the update.Generally, you do not modify it otherwise your NAND flash or other BSP code changed.

The "mfgimages-*" directory store need update files.Those name of files are defined in 'Manifest' file, please follow below rules:

```
ubootfile="u-boot.imx"
```

```
kernelfile="zImage"
dtbfile="myd-y6ull-14x14-gpmi-weim.dtb"
rootfsfile="core-image-base.rootfs.tar.xz"
```

The 'envfile' variable only used for eMMC flash type. The update program will read the Manifest file and load those files be written into flash.

The reference command is as follows:

```
sudo  ./build-sdcard.sh -s 256 -n  -x -f qt  -p myd-y6ull-hmi -d mfgimages-myd-y6ull-nand
sudo  ./build-sdcard.sh -s 256 -n   -f qt  -p myd-y6ull-hmi -d mfgimages-myd-y6ull-nand

sudo  ./build-sdcard.sh -s 256 -n  -x -f core  -p myd-y6ull-hmi -d mfgimages-myd-y6ull-nand
sudo  ./build-sdcard.sh -s 256 -n   -f core  -p myd-y6ull-hmi -d mfgimages-myd-y6ull-nand

sudo  ./build-sdcard.sh -s 512 -e  -x -f qt  -p myd-y6ull-hmi -d mfgimages-myd-y6ull-emmc
sudo  ./build-sdcard.sh -s 512 -e   -f qt  -p myd-y6ull-hmi -d mfgimages-myd-y6ull-emmc

sudo  ./build-sdcard.sh -s 512 -e  -x -f core  -p myd-y6ull-hmi -d mfgimages-myd-y6ull-emmc
sudo  ./build-sdcard.sh -s 512 -e   -f core  -p myd-y6ull-hmi -d mfgimages-myd-y6ull-emmc
```

The tool support four arguments. '-p' stands for a platform, the value is 'myd-y6ull-hmi'. '-n' stands for the storage device of NAND flash. '-e' stands for the storage device of eMMC flash. '-s' stands for stands for DDR size,256 or 512. '-d' stands for target files directory. '-f' stands for .File system type,the value is 'qt','core','mini' '-x' stands for Expansion board,if not, this parameter is not needed.

Attention: the '-n' and '-e' do not both exist in the argument.

After builds complete, a sdcard.img.gz suffix file in current directory, 'myd-y6ull-hmi-update-nand-ddr256m-core-20181114191138-exp.rootfs.sdcard.img.gz'.

Making updatable Micro SD

Insert Micro SD card to Card Reader, and plug into PC USB port.MYD-Y6ULX-HMI resource package support some prebuilt sdcard.gz files.You can use the tool to write it into your SD card.Those files locate in 02-Images direcory of resource package.

Attention: The date tag of file name is always changed, please follow the actual in 02-Images directory.

- Linux system

Generally, linux use "sd[x][n]" format to naming a storage device.The x means which storage device, represent use a ~ z character.The n means partition that storage device, use digit start from 1. You can use "dmesg | tail" command to view device name when you plugin Card Reader.In this case, we use "/dev/sdb" as example.

Attention: the "/dev/sdb" do not append any digit

Write sdcard file into USB storage:

```
sudo dd if= myd-y6ull-hmi-update-nand-ddr256m-core-20181114191138-exp.rootfs.sdcard.img \
of=/dev/sdb conv=fsync
```

Write sdcard.gz file into USB storage:

```
gzip -dc myd-y6ull-hmi-update-nand-ddr256m-core-20181114191138-exp.rootfs.sdcard. \
img.gz | sudo dd of=/dev/sdb conv=fsync
```

The write speed is relative with USB host version and Micro SD card write speed. We recommend use higher speed class Micro SD storage card.

- Windows system

The Windows user can use Win32DiskImager tool to write sdcard image file to Micro SD storage card.The tool is located in "03-Tools" directory.Extract it and double click "Win32DiskImager.exe" program.After Win32DiskImager window shows up, the right "Device" list is to choose which device needs to operation.The left "Image File" input box is to show which file needs to be operation through the folder icon to browse and choose file.(Attention: the file choose dialog default use "*.img" to filter files, you need change it to "*.*" type)

You need confirm the device and file before write operation.The wrong device will damage your system partition or other storage device.

In this case, "D:" is the Card Reader device.

Attention: You need extract the sdcard.img.gz file when you use Win32DiskImage write into USB storage.

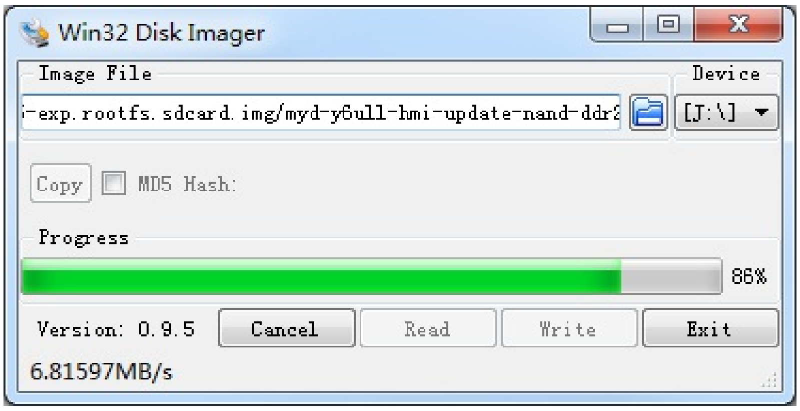


Figure6-1 Win32DiskImage write sdcard image file

You can plug out Card Reader after progress bard finish.

Take the Micro SD card insert into card slot(J5) on MYD-Y6ULX-HMI series boards.Then change boot switch as SDCARD type.

MYD-Y6ULX-HMI core board has two storage modes, NAND and eMMC, and the corresponding dial-code switch configuration at startup is also different, details are as follows.

BOOT From Sdcard:

- **MYD-Y6ULX-HMI NAND flash**

Boot bit	Status
Bit1	ON
Bit2	OFF
Bit3	ON
Bit4	OFF

- **MYD-Y6ULX-HMI eMMC flash**

Boot bit	Status
Bit1	OFF
Bit2	ON
Bit3	ON
Bit4	OFF

Use USB to TTL cable connect to Debug port(JP1), configure your serial terminal software.Use DC adapter plug into J1 interface.You can view update progress in serial terminal software.

BOOT From onboard flash:

- **MYD-Y6ULX-HMI NAND flash**

  You need power down and change the boot switch(SW1) to NAND boot type when you follow each way from two ways.

Boot bit	Status
Bit1	OFF
Bit2	ON
Bit3	ON
Bit4	OFF

Reconnect the power adapter, the board will boot into linux from NAND flash。

You need power down and change the boot switch(SW1) to eMMC boot type when you follow each way from two ways.

- **MYD-Y6ULX-HMI eMMC flash**

Boot bit	Status
Bit1	OFF
Bit2	OFF
Bit3	ON
Bit4	OFF

Reconnect the power adapter, the board will boot into linux from eMMC flash。

Appendix A 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:

Technical support service

- MYIR offers technical support for the hardware and software materials which have 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 the 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 themselves;
- Problems occurred during customers' own applications development;
- Problems occurred during the modification of MYIR's software source code.

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 damage the product;
- Due to the natural disasters (unexpected matters), or natural attrition of the components, 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 appearance;
- 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 products.

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.

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.

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.

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 Inteiligent Park, Bantian, Longgang District.

Support Email: support@myirtech.com

Sales Email: sales@myirtech.com

Phone: +86-755-22984836

Fax: +86-755-25532724

Website: www.myirtech.com