MYS-6ULX Linux Development Guide

MYiR™ Make Your Idea Real

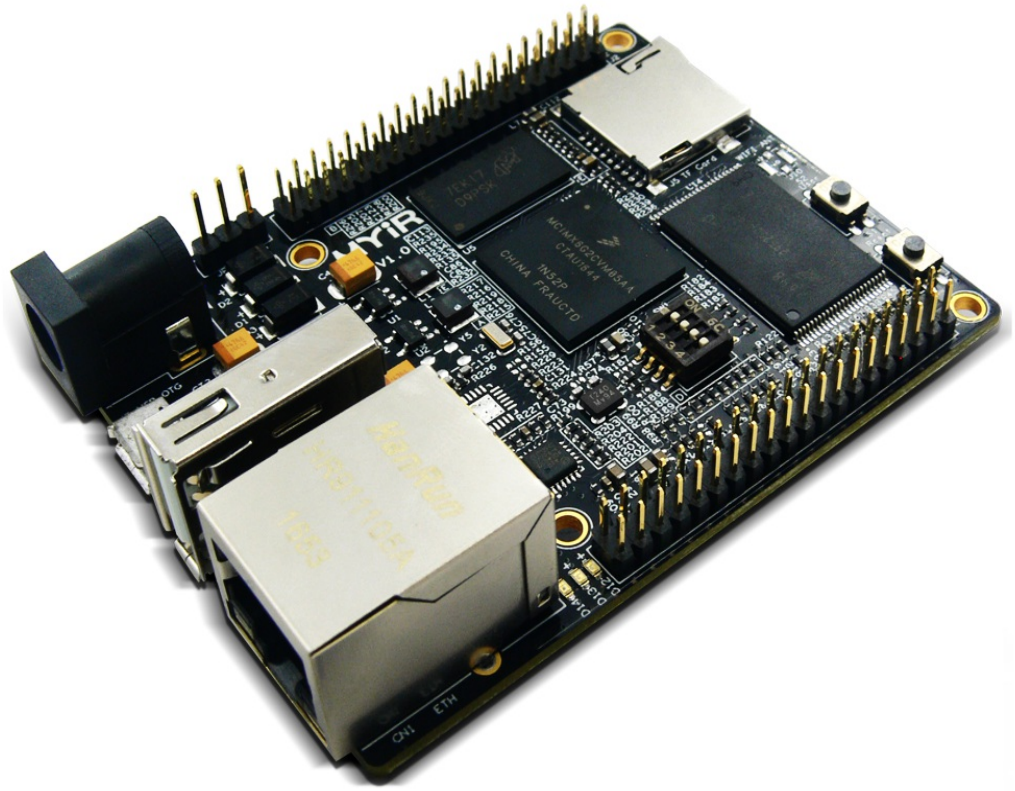


Table of Contents

MYS-6ULX Linux Development Guide

This document introduce the MYS-6ULX and MYB-6ULX development board about Linux compile and deploy, interface usage on baseboard, Qt application development etc.

Version History

Version	Description	Date
V1.0	Init version	2017.04.20
V1.1	add support MYB-6ULX board	2017.07.11
V1.2	add support 7.0 inch LCD screen	2017.08.03

Hardware Version

This document suit for below boards:

- MYS-6ULX-IND
- MYS-6ULX-IoT
- MYB-6ULX

1 Software Resource

MYS-6ULX series boards support the Linux kernel version 4.1.15, and provided with rich hardware resource and software resource. Some functions needs MYB-6ULX board installed on MYS-6ULX board.

Below is MYS-6ULX 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 4.3inch and 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	CANb bus driver	YES
Driver	RS485	RS485 bus driver	YES
Driver	Camera	ov2659 driver	YES
FileSystem	Debian rootfs	Based on Debian build filesystem(include X11 package)	BINARY
FileSystem	Yotcto rootfs	Based Yocto build filesystem(include Qt 5.6 package)	YES
FileSystem	Yotcto rootfs	Based Yocto build filesystem(Full command line package)	YES
Application	GPIO KEY	Reads the GPIO key code value demo	YES
Application	GPIO LED	Operation the GPIO LED demo	YES
Application	NET	Uses TCP/IP Sokect 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	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

PC serial port configure parameters:

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

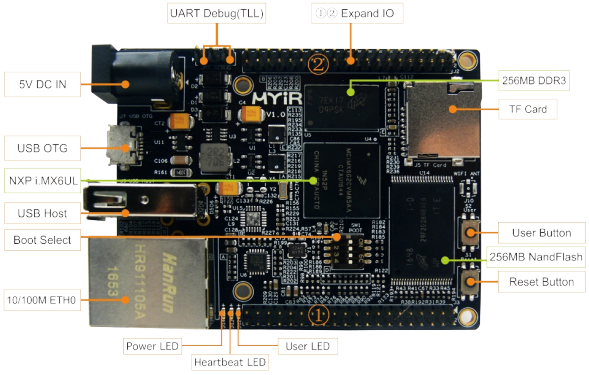


Figure2-1 MYS-6ULX-IND connection diagram

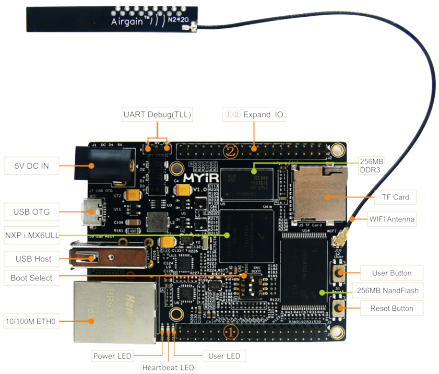


Figure2-2 MYS-6ULX-IoT connection diagram

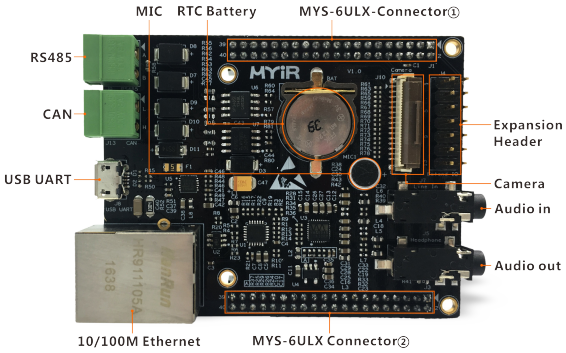


Figure2-3 MYB-6ULX interface diagram

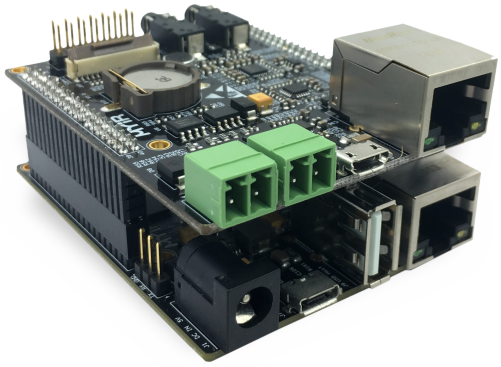


Figure2-4 MYS-6ULX install MYB-6ULX diagram

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 ~/MYS6ULx-devel
export DEV_ROOT=~/MYS6ULx-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 toolchinas, one is support by Linaro.Another built by Yocto.Recommend you use Yocto version 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 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 -v
Using built-in specs.
COLLECT_GCC=arm-linux-gnueabihf-gcc
COLLECT_LTO_WRAPPER=/home/kevinchen/SAMA5D4/gcc-linaro-4.9-2014.
11-x86_64_arm-linux-gnueabihf/bin/../libexec/gcc/arm-linux-gnuea
bihf/4.9.3/lto-wrapper
Target: arm-linux-gnueabihf
Configured with: /home/buildslave/workspace/BinaryRelease/label/
x86_64/target/arm-linux-gnueabihf/snapshots/gcc-linaro-4.9-2014.
11/configure SHELL=/bin/bash --with-bugurl=https://bugs.linaro.o
rg --with-mpc=/home/buildslave/workspace/BinaryRelease/label/x86
_64/target/arm-linux-gnueabihf/_build/builds/destdir/x86_64-unkn
own-linux-gnu --with-mpfr=/home/buildslave/workspace/BinaryRelea
se/label/x86_64/target/arm-linux-gnueabihf/_build/builds/destdir
/x86_64-unknown-linux-gnu
--with-gmp=/home/buildslave/workspace/BinaryRelease/label/x86_64
/target/arm-linux-gnueabihf/_build/builds/destdir/x86_64-unknown
-linux-gnu --with-gnu-as --with-gnu-ld --disable-libstdcxx-pch -
-disable-libmudflap --with-cloog=no --with-ppl=no --with-isl=no
--disable-nls --enable-multiarch --disable-multilib --enable-c99
--with-tune=cortex-a9 --with-arch=armv7-a --with-fpu=vfpv3-d16 -
-with-float=hard --with-mode=thumb --disable-shared --enable-sta
tic
--with-build-sysroot=/home/buildslave/workspace/BinaryRelease/la
bel/x86_64/target/arm-linux-gnueabihf/_build/sysroots/arm-linux-
gnueabihf --enable-lto --enable-linker-build-id --enable-long-lo
ng --enable-shared --with-sysroot=/home/buildslave/workspace/Bin
aryRelease/label/x86_64/target/arm-linux-gnueabihf/_build/builds
/destdir/x86_64-unknown-linux-gnu/libc --enable-languages=c,c++,
fortran,lto -enable-fix-cortex-a53-835769 --enable-checking=rele
ase --with-bugurl=https://bugs.linaro.org
--with-pkgversion='Linaro GCC 2014.11' --build=x86_64-unknown-li
nux-gnu --host=x86_64-unknown-linux-gnu --target=arm-linux-gnuea
bihf --prefix=/home/buildslave/workspace/BinaryRelease/label/x86
_64/target/arm-linux-gnueabihf/_build/builds/destdir/x86_64-unkn
own-linux-gnu
Thread model: posix
gcc version 4.9.3 20141031 (prerelease) (Linaro GCC 2014.11)
```

Yocto Toolchain

Yocto supports 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 libaries and header files.The MYS-6ULX also supports two kinds, those files are list below.

Toolchain file name	Description
myir-imx-fb-glibc-x86_64-fsl-image-qt5-cortexa7hf-neon-toolchain-4.1.15-2.0.1.sh	Include Qt5 libraries
myir-imx-fb-glibc-x86_64-core-image-base-cortexa7hf-neon-toolchain-4.1.15-2.0.1.sh	Not include Qt5 libraries
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-cortexa7
hf-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-corte
xa7hf-neon-poky-linux-gnueabi
arm-poky-linux-gnueabi-gcc -v
Using built-in specs.
COLLECT_GCC=arm-poky-linux-gnueabi-gcc
COLLECT_LTO_WRAPPER=/opt/myir-imx6ulx-fb/4.1.15-2.0.1/sysroots/
x86_64-pokysdk-linux/usr/libexec/arm-poky-linux-gnueabi/gcc/arm-
poky-linux-gnueabi/5.3.0/lto-wrapper
Target: arm-poky-linux-gnueabi
Configured with: ../../../../../../work-shared/gcc-5.3.0-r0/gcc-
5.3.0/configure --build=x86_64-linux --host=x86_64-pokysdk-linux
--target=arm-poky-linux-gnueabi --prefix=/opt/myir-imx6ulx-fb/4.
1.15-2.0.1/sysroots/x86_64-pokysdk-linux/usr --exec_prefix=/opt/
myir-imx6ulx-fb/4.1.15-2.0.1/sysroots/x86_64-pokysdk-linux/usr
--bindir=/opt/myir-imx6ulx-fb/4.1.15-2.0.1/sysroots/x86_64-pokys
dk-linux/usr/bin/arm-poky-linux-gnueabi
--sbindir=/opt/myir-imx6ulx-fb/4.1.15-2.0.1/sysroots/x86_64-poky
sdk-linux/usr/bin/arm-poky-linux-gnueabi --libexecdir=/opt/myir-
imx6ulx-fb/4.1.15-2.0.1/sysroots/x86_64-pokysdk-linux/usr/libexe
c/arm-poky-linux-gnueabi --datadir=/opt/myir-imx6ulx-fb/4.1.15-2
.0.1/sysroots/x86_64-pokysdk-linux/usr/share --sysconfdir=/opt/m
yir-imx6ulx-fb/4.1.15-2.0.1/sysroots/x86_64-pokysdk-linux/etc --
sharedstatedir=/opt/myir-imx6ulx-fb/4.1.15-2.0.1/sysroots/x86_64
-pokysdk-linux/com
--localstatedir=/opt/myir-imx6ulx-fb/4.1.15-2.0.1/sysroots/x86_6
4-pokysdk-linux/var --libdir=/opt/myir-imx6ulx-fb/4.1.15-2.0.1/s
ysroots/x86_64-pokysdk-linux/usr/lib/arm-poky-linux-gnueabi --in
cludedir=/opt/myir-imx6ulx-fb/4.1.15-2.0.1/sysroots/x86_64-pokys
dk-linux/usr/include --oldincludedir=/opt/myir-imx6ulx-fb/4.1.15
-2.0.1/sysroots/x86_64-pokysdk-linux/usr/include --infodir=/opt/
myir-imx6ulx-fb/4.1.15-2.0.1/sysroots/x86_64-pokysdk-linux/usr/s
hare/info
--mandir=/opt/myir-imx6ulx-fb/4.1.15-2.0.1/sysroots/x86_64-pokys
dk-linux/usr/share/man --disable-silent-rules --disable-dependen
cy-tracking --with-libtool-sysroot=/home/blackrose/mys-imx6ul/fs
l-release-yocto/build/tmp/sysroots/x86_64-nativesdk-pokysdk-linu
x --with-gnu-ld --enable-shared --enable-languages=c,c++ --enabl
e-threads=posix --enable-multilib --enable-c99 --enable-long-lon
g --enable-symvers=gnu --enable-libstdcxx-pch --program-prefix=a
rm-poky-linux-gnueabi- --without-local-prefix --enable-lto --ena
ble-libssp --enable-libitm --disable-bootstrap --disable-libmudf
lap --with-system-zlib --with-linker-hash-style=gnu --enable-lin
ker-build-id --with-ppl=no --with-cloog=no --enable-checking=rel
ease --enable-cheaders=c_global --without-isl --with-gxx-include
-dir=/not/exist/usr/include/c++/5.3.0 --with-build-time-tools=/h
ome/blackrose/mys-imx6ul/fsl-release-yocto/build/tmp/sysroots/x8
6_64-linux/usr/arm-poky-linux-gnueabi/bin --with-sysroot=/not/ex
ist --with-build-sysroot=/home/blackrose/mys-imx6ul/fsl-release-
yocto/build/tmp/sysroots/mys6ul14x14 --enable-poison-system-dire
ctories --with-mpfr=/home/blackrose/mys-imx6ul/fsl-release-yocto
/build/tmp/sysroots/x86_64-nativesdk-pokysdk-linux --with-mpc=/h
ome/blackrose/mys-imx6ul/fsl-release-yocto/build/tmp/sysroots/x8
6_64-nativesdk-pokysdk-linux --enable-nls --with-arch=armv7-a
Thread model: posix
gcc version 5.3.0 (GCC)
```

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

3 Build Linux System

This chapter introduces how to build componments of Linux system. The MYS-6ULX includes below parts:

- U-Boot: First level bootloader.
- Linux Kernel: Linux kernel 4.1.15 and drivers suit for MYS-6ULX 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 locate in 04-Source directory.Some packages use ID number in file name.

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

3.1 Compiling U-Boot

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

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

Compiling：

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

The option value is different boot mode. The MYS-6ULX supports four boot modes.

Boot Mode	Config file
MYS-6ULX-IND NAND Flash	mys_imx6ul_14x14_nand_defconfig
MYS-6ULX-IND eMMC Flash	mys_imx6ul_14x14_emmc_defconfig
MYS-6ULX-IoT NAND Flash	mys_imx6ull_14x14_nand_defconfig
MYS-6ULX-IoT eMMC Flash	mys_imx6ull_14x14_emmc_defconfig

U-Boot 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 "mys-imx6ul-boot-sdcard.txt" to generate the "boot.scr" file as example. The mkimage tool source code is locate in "U-Boot/tools" directory. It will be compiled after U-Boot compiled.

```
cat mys-imx6ul-boot-sdcard.txt
setenv mmcroot '/dev/mmcblk0p2 rootwait rw rootdelay=5 mem=256M'
setenv mmcargs 'setenv bootargs console=${console},${baudrate} \
root=${mmcroot} mtdparts=gpmi-nand:5m(boot),10m(kernel),\
1m(dtb),-(rootfs)'
run mmcargs
fatload mmc 0 0x83000000 zImage
fatload mmc 0 0x84000000 mys-imx6ul-14x14-evk-emmc.dtb
bootz 0x83000000 - 0x84000000

./tool/mkimage -A arm -T script -O linux \
-d mys-imx6ul-boot-sdcard.txt boot.scr
```

3.2 Linux Kernel

Enter Kernel directory, extract it:

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

Compiling：

```
make distclean
make mys_imx6_defconfig
make zImage dtbs modules
```

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 File	Description
mys-imx6ul-14x14-evk-emmc.dtb	MYS-6ULX-IND eMMC
mys-imx6ul-14x14-evk-gpmi-weim.dtb	MYS-6ULX-IND NAND
mys-imx6ull-14x14-evk-emmc.dtb	MYS-6ULX-IoT eMMC
mys-imx6ull-14x14-evk-gpmi-weim.dtb	MYS-6ULX-IoT NAND

The MYS-6ULX Micro SD slot is connected to mmc0 controller.So, all dtb files is enabled the mmc0 controller by default.

Those "myb6ulx" tag dtb files are support MYB-6ULX board, configured CAN, RS485, Ethernet, Camera and Audio functions.Please install the MYB-6ULX onto MYS-6ULX-IOT or MYS-6ULX-IND board before use those files.

About SD card or eMMC boot mode, the U-Boot default find the mys-imx6ul-14x14-evk-emmc.dtb or mys-imx6ull-14x14-evk-emmc.dtb file.You need rename those dtb files with "myb6ulx" tag after modify and compile.The NAND boot mode not affect with it.

3.3 Build File System

The Linux platform has many open source tools to build filesystem.These tools has some features to imporve 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 suite 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 a 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 guid 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 difference 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 customize system by NXP.We also support config files to support MYS-6ULX series boards.This will help developer to build Linux system that can be programming to MYS-6ULX series boards.

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

Yocto Project Quick start

Bitback User Manual

Yocto Project Reference Manual

Yocto Project Development Manual

Yocto Project Complete Documentation Set

3.3.1 Yocto build Linux system

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

The Yocto needs download all third software packages from internet.In order to build more speedly, MYS-6ULX also support a full package, you could not download again.

Below has two ways to use Yocto:

- Using MYS-6ULX full package
- Using NXP supports Yocto and config it for MYS-6ULX

If you the first time hear about Yocto, recommend use the first way

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

MYS-6ULX support 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 MYS-6ULX from Yocto.

Attention: The Yocto-downloads.tar.xz file is more large, it does not include in MYS-6ULX iso file.Please visit and download it from http://d.myirtech.com/MYS-6ULX.

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

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

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

NXP support Yocto

Yocto has many sub-porjects, so it use repo tool to manage those projects.You need configure the git and download Yocto from NXP source code repository.

- Setting repo

```
mkdir ~/bin
curl http://commondatastorage.googleapis.com/git-repo-downloads/repo \
> ~/bin/repo
chmod a+x ~/bin/repo
export PATH=~/bin:$PATH
```

- Setting Yocto

```
git config --global user.name "Your Name"
git config --global user.email "Your mail"
```

```
cd $DEV_ROOT
mkdir fsl-release-bsp
cd fsl-replease-bsp
repo init -u git://git.freescale.com/imx/fsl-arm-yocto-bsp.git -b imx-4.1-krogoth
repo sync
```

When the sync complete, you need copy the meta-myir-imx6ulx to fsl-release-bsp/source directory.On next steps, you need add next line to conf/bblayers.conf file.

```
BBLAYERS += " ${BSPDIR}/sources/meta-myir-imx6ulx "
```

Also, needs put the kernel and u-boot source into your home directory in linux.It will be fetch with Yocto.

```
mkdir ~/MYS-6ULX-Linux
mkdir ~/MYS-6ULX-uboot
tar xvf $DEV_ROOT/04-Source/linux-4.1.15.tar.gz -C ~/MYS-6ULX-Linux
tar xvf $DEV_ROOT/04-Source/u-boot-2016.03.tar.gz -C ~/MYS-6ULX-uboot
```

Build system image with Qt5 package

Use script by NXP supported, create a workspace directory, and Yocto will build all under it. MYS-6ULX has two types board, "mys6ull14x14" is MYS-6ULX-IoT and "mys6ul14x14" is MYS-6ULX-IND.

```
DISTRO=myir-imx-fb MACHINE=mys6ul14x14 source fsl-setup-release.sh -b build
bitbake fsl-image-qt5
```

Build system iamge 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 MYS-6ULX to update 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 finish, 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 depend on your PC cpu core number and RAM size. The Yocto recommend use eight core CPU and SSD hardware to impove build speed. Another way, 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/mys6ul14x14/" directory after build complete. Below as example:

```
$ ls -lh tmp/deploy/images/mys6ul14x14/
total 1.8G
-rw-r--r-- 1 kevinchen kevinchen  56M Apr 16 23:05 core-image-minimal-mys6ul14x14-
20170416150516.rootfs.ext4
-rw-r--r-- 1 kevinchen kevinchen 2.1K Apr 16 23:05 core-image-minimal-mys6ul14x14-
20170416150516.rootfs.manifest
-rw-r--r-- 1 kevinchen kevinchen  72M Apr 16 23:05 core-image-minimal-mys6ul14x14-
20170416150516.rootfs.sdcard
-rw-r--r-- 1 kevinchen kevinchen  11M Apr 16 23:05 core-image-minimal-mys6ul14x14-
20170416150516.rootfs.tar.bz2
-rw-r--r-- 1 kevinchen kevinchen 7.3M Apr 16 23:05 core-image-minimal-mys6ul14x14-
20170416150516.rootfs.tar.xz
lrwxrwxrwx 1 kevinchen kevinchen   57 Apr 16 23:05 core-image-minimal-mys6ul14x14.
ext4 -> core-image-minimal-mys6ul14x14-20170416150516.rootfs.ext4
lrwxrwxrwx 1 kevinchen kevinchen   61 Apr 16 23:05 core-image-minimal-mys6ul14x14.
manifest -> core-image-minimal-mys6ul14x14-20170416150516.rootfs.manifest
lrwxrwxrwx 1 kevinchen kevinchen   59 Apr 16 23:05 core-image-minimal-mys6ul14x14.
sdcard -> core-image-minimal-mys6ul14x14-20170416150516.rootfs.sdcard
lrwxrwxrwx 1 kevinchen kevinchen   60 Apr 16 23:05 core-image-minimal-mys6ul14x14.
tar.bz2 -> core-image-minimal-mys6ul14x14-20170416150516.rootfs.tar.bz2
lrwxrwxrwx 1 kevinchen kevinchen   59 Apr 16 23:05 core-image-minimal-mys6ul14x14.
tar.xz -> core-image-minimal-mys6ul14x14-20170416150516.rootfs.tar.xz
-rw-r--r-- 1 kevinchen kevinchen 780M Apr 16 23:08 fsl-image-qt5-mys6ul14x14-
20170416150603.rootfs.ext4
-rw-r--r-- 1 kevinchen kevinchen  35K Apr 16 23:08 fsl-image-qt5-mys6ul14x14-
20170416150603.rootfs.manifest
-rw-r--r-- 1 kevinchen kevinchen 796M Apr 16 23:08 fsl-image-qt5-mys6ul14x14-
20170416150603.rootfs.sdcard
-rw-r--r-- 1 kevinchen kevinchen 166M Apr 16 23:08 fsl-image-qt5-mys6ul14x14-
20170416150603.rootfs.tar.bz2
-rw-r--r-- 1 kevinchen kevinchen 105M Apr 16 23:09 fsl-image-qt5-mys6ul14x14-
20170416150603.rootfs.tar.xz
lrwxrwxrwx 1 kevinchen kevinchen   52 Apr 16 23:08 fsl-image-qt5-mys6ul14x14.
```

```
ext4 -> fsl-image-qt5-mys6ul14x14-20170416150603.rootfs.ext4
lrwxrwxrwx 1 kevinchen kevinchen   56 Apr 16 23:08 fsl-image-qt5-mys6ul14x14.
manifest -> fsl-image-qt5-mys6ul14x14-20170416150603.rootfs.manifest
lrwxrwxrwx 1 kevinchen kevinchen   54 Apr 16 23:09 fsl-image-qt5-mys6ul14x14.
sdcard -> fsl-image-qt5-mys6ul14x14-20170416150603.rootfs.sdcard
lrwxrwxrwx 1 kevinchen kevinchen   55 Apr 16 23:09 fsl-image-qt5-mys6ul14x14.tar.bz2
-> fsl-image-qt5-mys6ul14x14-20170416150603.rootfs.tar.bz2
lrwxrwxrwx 1 kevinchen kevinchen   54 Apr 16 23:09 fsl-image-qt5-mys6ul14x14.tar.xz
-> fsl-image-qt5-mys6ul14x14-20170416150603.rootfs.tar.xz
-rw-r--r-- 2 kevinchen kevinchen 657K Apr 16 23:04 modules--4.1.15-r0-mys6ul14x14-
20170416150349.tgz
lrwxrwxrwx 1 kevinchen kevinchen   49 Apr 16 23:04 modules-mys6ul14x14.tgz ->
modules--4.1.15-r0-mys6ul14x14-20170416150349.tgz
-rw-r--r-- 2 kevinchen kevinchen  294 Apr 16 23:07 README_-_DO_NOT_DELETE_FILES_IN_
THIS_DIRECTORY.txt
-rwxr-xr-x 2 kevinchen kevinchen 375K Apr 16 22:53 u-boot-emmc-2016.03-r0.imx
lrwxrwxrwx 1 kevinchen kevinchen   26 Apr 16 22:53 u-boot.imx -> u-boot-emmc-
2016.03-r0.imx
lrwxrwxrwx 1 kevinchen kevinchen   26 Apr 16 22:53 u-boot.imx-emmc -> u-boot-emmc-
2016.03-r0.imx
lrwxrwxrwx 1 kevinchen kevinchen   26 Apr 16 22:53 u-boot.imx-nand -> u-boot-nand-
2016.03-r0.imx
lrwxrwxrwx 1 kevinchen kevinchen   24 Apr 16 22:53 u-boot.imx-sd -> u-boot-sd-
2016.03-r0.imx
lrwxrwxrwx 1 kevinchen kevinchen   26 Apr 16 22:53 u-boot-mys6ul14x14.imx ->
u-boot-emmc-2016.03-r0.imx
lrwxrwxrwx 1 kevinchen kevinchen   26 Apr 16 22:53 u-boot-mys6ul14x14.imx-emmc ->
u-boot-emmc-2016.03-r0.imx
lrwxrwxrwx 1 kevinchen kevinchen   26 Apr 16 22:53 u-boot-mys6ul14x14.imx-nand ->
u-boot-nand-2016.03-r0.imx
lrwxrwxrwx 1 kevinchen kevinchen   24 Apr 16 22:53 u-boot-mys6ul14x14.imx-sd ->
u-boot-sd-2016.03-r0.imx
-rwxr-xr-x 2 kevinchen kevinchen 427K Apr 16 22:53 u-boot-nand-2016.03-r0.imx
-rwxr-xr-x 2 kevinchen kevinchen 375K Apr 16 22:53 u-boot-sd-2016.03-r0.imx
lrwxrwxrwx 1 kevinchen kevinchen   48 Apr 16 23:04 zImage -> zImage--4.1.15-r0-
mys6ul14x14-20170416150349.bin
-rw-r--r-- 2 kevinchen kevinchen 6.5M Apr 16 23:04 zImage--4.1.15-r0-mys6ul14x14-
20170416150349.bin
-rw-r--r-- 2 kevinchen kevinchen  36K Apr 16 23:04 zImage--4.1.15-r0-mys-imx6ul-14x14-evk-
20170416150349.dtb
-rw-r--r-- 2 kevinchen kevinchen  37K Apr 16 23:04 zImage--4.1.15-r0-mys-imx6ul-14x14-evk-
emmc-20170416150349.dtb
-rw-r--r-- 2 kevinchen kevinchen  37K Apr 16 23:04 zImage--4.1.15-r0-mys-imx6ul-14x14-
evk-gpmi-weim-20170416150349.dtb
lrwxrwxrwx 1 kevinchen kevinchen   48 Apr 16 23:04 zImage-mys6ul14x14.bin -> zImage--
4.1.15-r0-mys6ul14x14-20170416150349.bin
lrwxrwxrwx 1 kevinchen kevinchen   57 Apr 16 23:04 zImage-mys-imx6ul-14x14-evk.dtb ->
zImage--4.1.15-r0-mys-imx6ul-14x14-evk-20170416150349.dtb
lrwxrwxrwx 1 kevinchen kevinchen   62 Apr 16 23:04 zImage-mys-imx6ul-14x14-evk-emmc.
dtb -> zImage--4.1.15-r0-mys-imx6ul-14x14-evk-emmc-20170416150349.dtb
lrwxrwxrwx 1 kevinchen kevinchen   67 Apr 16 23:04 zImage-mys-imx6ul-14x14-evk-gpmi-
weim.dtb -> zImage--4.1.15-r0-mys-imx6ul-14x14-evk-gpmi-weim-20170416150349.dtb
```

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 boot from eMMC
u-boot-nand-2016.03-r0.imx	u-boot used for boot from NAND
u-boot-sd-2016.03-r0.imx	u-boot used for boot 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 Yocto build SDK pakcage

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 differen way, one is suite 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 complete:

```
$ ls tmp/deploy/sdk/ -lh
-rw-r--r-- 1 kevinchen kevinchen 9.5K Apr 17 00:00 myir-imx6ulx-
fb-glibc-x86_64-meta-toolchain-cortexa7hf-neon-toolchain-4.1.15-
2.0.1.host.manifest
-rwxr-xr-x 1 kevinchen kevinchen  76M Apr 17 00:01 myir-imx6ulx-
fb-glibc-x86_64-meta-toolchain-cortexa7hf-neon-toolchain-4.1.15-
2.0.1.sh
-rw-r--r-- 1 kevinchen kevinchen 1.6K Apr 17 00:00 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-iamge-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
-rw-r--r-- 1 kevinchen kevinchen 9.5K Apr 17 07:54 myir-imx6ulx-
fb-glibc-x86_64-fsl-image-qt5-cortexa7hf-neon-toolchain-4.1.15-
2.0.1.host.manifest
-rwxr-xr-x 1 kevinchen kevinchen 587M Apr 17 07:59 myir-imx6ulx-
fb-glibc-x86_64-fsl-image-qt5-cortexa7hf-neon-toolchain-4.1.15-
2.0.1.sh
-rw-r--r-- 1 kevinchen kevinchen  70K Apr 17 07:54 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 distrubuted and installed to other Linux system and compile program to target device.

4 Linux Application Development

The hardware peripherals and application examples of MYS-6ULX development board.

Before use, you need use 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.

```
. /opt/myir-imx6ulx-fb/4.1.15-2.0.1/environment-setup-\
cortexa7hf-neon-poky-linux-gnueabi
arm-poky-linux-gnueabi-gcc -v
Using built-in specs.
COLLECT_GCC=arm-poky-linux-gnueabi-gcc
COLLECT_LTO_WRAPPER=/opt/myir-imx6ulx-fb/4.1.15-2.0.1/sysroots/
x86_64-pokysdk-linux/usr/libexec/arm-poky-linux-gnueabi/gcc/
arm-poky-linux-gnueabi/5.3.0/lto-wrapper
Target: arm-poky-linux-gnueabi
Configured with: ../../../../../../work-shared/gcc-5.3.0-r0/gcc-
5.3.0/configure --build=x86_64-linux --host=x86_64-pokysdk-linux
--target=arm-poky-linux-gnueabi --prefix=/opt/myir-imx6ulx-fb/4.
1.15-2.0.1/sysroots/x86_64-pokysdk-linux/usr --exec_prefix=/opt/
myir-imx6ulx-fb/4.1.15-2.0.1/sysroots/x86_64-pokysdk-linux/usr
--bindir=/opt/myir-imx6ulx-fb/4.1.15-2.0.1/sysroots/x86_64-
pokysdk-linux/usr/bin/arm-poky-linux-gnueabi --sbindir=/opt/
myir-imx6ulx-fb/4.1.15-2.0.1/sysroots/x86_64-pokysdk-linux/usr/
bin/arm-poky-linux-gnueabi --libexecdir=/opt/myir-imx6ulx-fb/
4.1.15-2.0.1/sysroots/x86_64-pokysdk-linux/usr/libexec/
arm-poky-linux-gnueabi --datadir=/opt/myir-imx6ulx-fb/
4.1.15-2.0.1/sysroots/x86_64-pokysdk-linux/usr/share
--sysconfdir=/opt/myir-imx6ulx-fb/4.1.15-2.0.1/sysroots/
x86_64-pokysdk-linux/etc --sharedstatedir=/opt/myir-imx6ulx-fb/
4.1.15-2.0.1/sysroots/x86_64-pokysdk-linux/com --localstatedir=
/opt/myir-imx6ulx-fb/4.1.15-2.0.1/sysroots/x86_64-pokysdk-
linux/var --libdir=/opt/myir-imx6ulx-fb/4.1.15-2.0.1/sysroots/
x86_64-pokysdk-linux/usr/lib/arm-poky-linux-gnueabi --includedir
=/opt/myir-imx6ulx-fb/4.1.15-2.0.1/sysroots/x86_64-pokysdk-linux
/usr/include --oldincludedir=/opt/myir-imx6ulx-fb/4.1.15-2.0.1/
sysroots/x86_64-pokysdk-linux/usr/include --infodir=/opt/
myir-imx6ulx-fb/4.1.15-2.0.1/sysroots/x86_64-pokysdk-linux/usr/
share/info --mandir=/opt/myir-imx6ulx-fb/4.1.15-2.0.1/sysroots/
x86_64-pokysdk-linux/usr/share/man --disable-silent-rules
--disable-dependency-tracking --with-libtool-sysroot=
/home/kevinchen/mys-imx6ul/fsl-release-yocto/build/tmp/sysroots
/x86_64-nativesdk-pokysdk-linux --with-gnu-ld --enable-shared
--enable-languages=c,c++ --enable-threads=posix
--enable-multilib --enable-c99 --enable-long-long
--enable-symvers=gnu --enable-libstdcxx-pch
--program-prefix=arm-poky-linux-gnueabi-
--without-local-prefix --enable-lto --enable-libssp
--enable-libitm --disable-bootstrap --disable-libmudflap
--with-system-zlib --with-linker-hash-style=gnu --enable-linker-
build-id --with-ppl=no --with-cloog=no --enable-checking=release
--enable-cheaders=c_global --without-isl --with-gxx-include-dir=
/not/exist/usr/include/c++/5.3.0 --with-build-time-tools=
/home/kevinchen/mys-imx6ul/fsl-release-yocto/build/tmp/sysroots/
x86_64-linux/usr/arm-poky-linux-gnueabi/bin --with-sysroot=/not/
exist --with-build-sysroot=/home/kevinchen/mys-imx6ul/
fsl-release-yocto/build/tmp/sysroots/mys6ul14x14
--enable-poison-system-directories --with-mpfr=
/home/kevinchen/mys-imx6ul/fsl-release-yocto/build/tmp/sysroots/
x86_64-nativesdk-pokysdk-linux --with-mpc=/home/kevinchen/
mys-imx6ul/fsl-release-yocto/build/tmp/sysroots/x86_64-nativesdk
-pokysdk-linux --enable-nls --with-arch=armv7-a
Thread model: posix
gcc version 5.3.0 (GCC)

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 MYS6ULX board LCD interface(J8).We have two kinds LCD with touch panel, 7-inch capacitive screen is MY-TFT070CV2 and 4.3inch resistive screen is MY-TFT043RV2.

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
```

Configure for MY-TFT070RV2 module

The Linux source of MYS-6ULX series board has alread support display and touch function.The MY-TFT070RV2 touch function through ADC type.You just enable the relative function in dts file.

- MYS-6ULX-IND The first step, edit "arch/arm/boot/dts/mys-imx6ul-14x14-evk.dts" file, 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 = <0xffff>;
     pre-charge-time = <0xffff>;
     status = "okay";
  };
  ```

  The second step, comment the argument of 4.3 inch screen, and enable argument of 7.0 inch screen.Search and modify the display-timings node of lcfif node, change it follow below.

  ```
        display-timings {
            native-mode = <&timing0>;
  /*
             timing0: timing0 {
             clock-frequency = <9200000>
             hsync-len = <41>;
             vback-porch = <2>;
             vfront-porch = <4>;
             vsync-len = <10>;

             hsync-active = <0>;
             vsync-active = <0>;
             de-active = <1>;
             pixelclk-active = <0>;
             };
  */
             timing0: timing0 {
             clock-frequency = <33000000>;
             hactive = <800>;
             vactive = <480>;
             hfront-porch = <210>;
             hback-porch = <46>;
             hsync-len = <1>;
             vback-porch = <22>;
             vfront-porch = <23>;
  ```

```
            vsync-len = <20>;

            hsync-active = <0>;
            vsync-active = <0>;
            de-active = <1>;
            pixelclk-active = <1>;
            };

      };
```

- MYS-6ULX-IoT The MYS-6ULX-IoT use same method with MYS-6ULX-IND, you just edit the "arch/arm/boot/dts/mys-imx6ull-14x14-evk.dts" file follow above step.

Configure for MY-TFT070CV2 module

The touch function of MY-TFT070CV2 module use I2C type.The slave device has already added to i2c2 controller node.You just disable tsc controller and enable argument of 7 inch screen.

- MYS-6ULX-IND The first step, edit "arch/arm/boot/dts/mys-imx6ul-14x14-evk.dts" file, modify the status property of tsc node is disabled value.

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

  The second step, comment the argument of 4.3 inch screen.And enable the argument of 7.0 inch screen.Search and modify display-timings node of lcdif node, change it follow below.

```
      display-timings {
          native-mode = <&timing0>;
/*
            timing0: timing0 {
            clock-frequency = <9200000>
            hsync-len = <41>;
            vback-porch = <2>;
            vfront-porch = <4>;
            vsync-len = <10>;

            hsync-active = <0>;
            vsync-active = <0>;
            de-active = <1>;
            pixelclk-active = <0>;
            };
*/
            timing0: timing0 {
            clock-frequency = <33000000>;
            hactive = <800>;
            vactive = <480>;
            hfront-porch = <210>;
            hback-porch = <46>;
            hsync-len = <1>;
            vback-porch = <22>;
            vfront-porch = <23>;
            vsync-len = <20>;

            hsync-active = <0>;
            vsync-active = <0>;
            de-active = <1>;
            pixelclk-active = <1>;
            };

      };
```

- MYS-6ULX-IoT The MYS-6ULX-IoT use same method with MYS-6ULX-IND.Follow above step to edit "arch/arm/boot/dts/mys-imx6ull-14x14-evk.dts"file.

4.2 Test TouchPanel

This example shows you how to test the touch panel on LCD screen module. The MYS-6ULX supports capacitive and resistive type.We also have two LCD with touch panel module, MY-TFT070CV2 and MY-TFT043RV2.

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 difference device node.

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

# ts_test
```

4.3 Ethernet(CN1)

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.

Configure IP of PC machine and run server program:

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

Run client program on board:

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

4.3 Test GPIO-KEY

This example demonstrates how to read key state and key values in Linux user space. After running the gpio_key program, press or release the S2 key, the debug serial port will output the relevant status information. Press "Ctrl-C" to end the program.

- Run the program on board:

```
# ./gpio_key /dev/input/event2
Hit any key on board ......
key 2 Pressed
key 2 Released
key 2 Pressed
key 2 Released
```

4.5 Test GPIO-LED

This example demonstrates using the Linux system API to operate LED(D12) light on and off. After running the program, D12 flash alternately. Press "Ctrl-C" to end the program.

```
# ./gpio_led /sys/class/leds/user/brightness
```

4.6 Test USB Host

Connect the USB flash disk to USB HOST(J9) interface, will output detection device information. At the same time, use this storage device to mount the Linux system under its read and write.

```
# 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
```

4.11 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
insmod g_mass_storage.ko 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 a 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
```

4.8 Test RS485

Attention: This example needs install the MYB-6ULX on MYS-6ULX-IOT or MYS-6ULX-IND.And uses dtb file to boot Linux that supoorts MYB-6ULX board.

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

MYB-6ULX board have an RS485 interface(J9).You need connect the A，B signal wire to another RS485 device or USB to RS485 converter.

Software

Copy and run the program on MYS-6ULX Linux system.Below is MYS-6ULX as the sender:

```
./rs485_write -d /dev/ttymxc2 -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
```

MYS-6ULX as recevier：

```
./rs485_read -d /dev/ttymxc2 -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
```

4.9 Test CAN Bus

Attention: This example needs install the MYB-6ULX on MYS-6ULX-IOT or MYS-6ULX-IND.And uses dtb file to boot Linux that supoorts MYB-6ULX board.

This example demonstrates the use of the Linux API, which uses the CAN bus interface on the development board to send and receive data. Copy can_send and can_receive to the development board. Perform the following steps

MYD-JA5D2X board has one CAN bus interface(J20), which can communicate with other CAN node device.

Hareware connect

MYB-6ULX board has an CAN bus interface(J11).You need connect the H, L data signal to another CAN device or USB to CAN converter.

Software

Configure the CAN0 bitrate is 50kbps and enable it.

Linux has two commands to configure CAN device, canconfig and ip. The MYS-6ULX use ip as default. canconfig command usage:

```
canconfig can0 bitrate 50000 ctrlmode triple-sampling on
canconfig can0 start
```

ip command usage:

```
ip link set can0 type can bitrate 50000 triple-sampling on
ifconfig can0 up
```

Next, we use cansend/candump command and can_send/can_receive program to test CAN device.The cansend/candump is buildin system rootfs.The can_send/can_receive is our example program.

- MYB-6ULX as sender

Use cansend send data to CAN bus:

```
cansend can0 100#01.02.03.04.05.06.07.08
```

can_send program will sending loop, unitl you use "ctrl + c" to stop it.

```
./can_send -d can0 -i 100 0x11 0x22 0x33 0x44 0x55 0x66 0x77 0x88
```

- MYB-6ULX as receiver

Use candump receive data from CAN bus:

```
candump can0
can0  100   [8]  01 02 03 04 05 06 07 08
```

Use can_receive program receive data from CAN bus: ``` can0 0x100 [8] 0x11 0x22 0x33 0x44 0x55 0x66 0x77 0x88 can0 0x100 [8] 0x11 0x22 0x33 0x44 0x55 0x66 0x77 0x88 can0 0x100 [8] 0x11 0x22 0x33 0x44 0x55 0x66 0x77 0x88 can0 0x100 [8] 0x11 0x22 0x33 0x44 0x55 0x66 0x77 0x88 ...

4.10 Test Audio

Attention: This example needs install the MYB-6ULX on MYS-6ULX-IOT or MYS-6ULX-IND.And uses dtb file to boot Linux that supoorts MYB-6ULX board.

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

Hareware connect

- Connect LINE IN(J7) interface on the MYB-6ULX board, PC Audio-Out via a 3.5mm AUX cable
- HEADPHONE(J5) connecte the your headerphone or speaker

Software operation

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
```

4.11 Test Camera

Attention: This example needs install the MYB-6ULX on MYS-6ULX-IOT or MYS-6ULX-IND.And uses dtb file to boot Linux that supoorts MYB-6ULX board.

MYB-6ULX board has an paraller camera interface(J10).It can connects camera module of MY-CAM011B model.The module and board connects with FPC wire.Inorder the signal wire order, please do not insert other Camera module into interface.This operation will damage your board or camera module.

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

Hardware connect

Use FPC wire connects MYB-CAM011B module and camera interface J10 of MYB-6ULX.

Software operations

The uvc_stream through network show video data.You need setup ethernet IP address of MYS-6ULX, 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 camera only support 800x600.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.

4.12 WiFi Test

Attention: This test only for MYB-6ULX-IOT version

The MYS-6ULX-IOT board has a WiFi module (U13) on back.It's supports Client and AP mode.

Hardware Connective

Use I-PEX interface of wireless antenna connect with J10 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.

```
lsmod
Module                  Size  Used by
8188eu                758318  0

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)
```

Uses wpa_passphrase to generate password and ssid through you inputed arguments. And use wpa_supplicant command to connect WiFi module with AccessPoint.

```
wpa_passphrase "MYiRTech" >> wifi.conf
12345678

cat wifi.conf
# reading passphrase from stdin
network={
    ssid="MYiRTech"
    #psk="12345678"
    psk=b96d9a5de2d9480ad5f987857e20216b47a0c4bf43397825ba909438bc52aaff
}

wpa_supplicant -D wext -B -i wlan0 -c wifi.conf
Successfully initialized wpa_supplicant
rfkill: Cannot open RFKILL control device
R8188EU: Firmware Version 11, SubVersion 1, Signature 0x88e1
MAC Address = a0:2c:36:60:ee:e0
IPv6: ADDRCONF(NETDEV_UP): wlan0: link is not ready
ioctl[SIOCSIWAP]: Operation not permitted
R8188EU: INFO indicate disassoc
```

After the WiFi module connected with AccessPoint, it needs udhcpc service to fetch an AIP address from AccessPoint.

```
udhcpc -b -i wlan0 -R
ifconfig wlan0
wlan0     Link encap:Ethernet  HWaddr a0:2c:36:60:ee:e0
          inet addr:192.168.1.211  Bcast:192.168.1.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:5577 errors:0 dropped:15 overruns:0 frame:0
          TX packets:46 errors:0 dropped:3 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:651690 (636.4 KiB)  TX bytes:7472 (7.2 KiB)
```

AP Mode

AccessPoint mode needs software and hardware to suporrt AP feature. The WiFi module of MYS-6ULX-IOT hs support AP mode.It use Hostapd to support AP function.

Compile Hostapd

MYS-6ULX resource package has contains hostapd source.And prebuilt image has support hostapd package.

The configuration file required by the AP mode is given in the file system and stored in the /etc/wifi-conf/ directory. The contents of udhcpcd.conf are as follows:

```
# the start and end of the IP lease block
start       192.168.10.10
end         192.168.10.254
# the interface that udhcpd will use
interface   wlan0

option  subnet  255.255.255.0
opt router  192.168.10.1
option  domain  local
option  lease   864000
```

The contents of hostapd.conf are as follows:

```
# WPA2-PSK authentication with AES encryption
interface=wlan0
driver=nl80211
ctrl_interface=/var/run/hostapd
ssid=MYIR-WIFI-AP
channel=6
ieee80211n=1
hw_mode=g
ignore_broadcast_ssid=0
wpa=2
wpa_passphrase=MYIR-TECH
wpa_key_mgmt=WPA-PSK
rsn_pairwise=CCMP
```

Then use wlan0 as the route:

```
udhcpd /etc/wifi-conf/udhcpd.conf
hostapd -B /etc/wifi-conf/hostapd.conf
```

Use your mobile phone or other WiFi device to connect the SSID to the "MYIR-WIFI-AP" hotspot, password: "MYIR-TECH", and connect to the correct IP address.

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 MYS-6ULX 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":

```
$ vi /opt/qtcreator-4.1.0/bin/qtcreator.sh
source /opt/myir-imx6ulx-fb/4.1.15-2.0.1/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 "MYS6ULx-GCC", click "Compiler path" beside "Browse.." button to choose "arm-poky-linux-gnueabi-g++" file path. In this case, the path is "/opt/myir-imx6ulx-fb/4.1.15-2.0.1/sysroots/x86_64-pokysdk-linux/usr/bin/arm-poky-linux-gnueabi/arm-poky-linux-gnueabi-g++".When fill are complete, click "Apply".

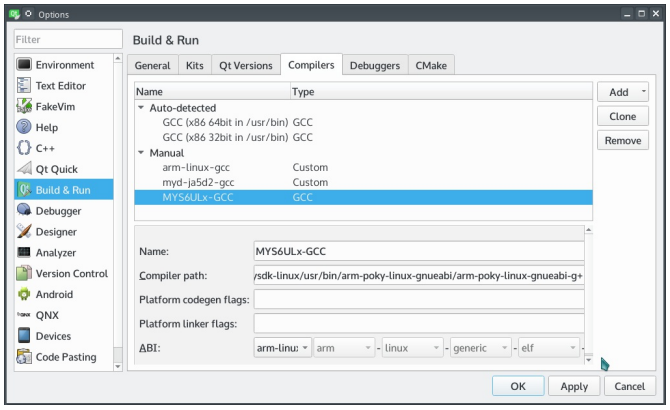


Figure5-1 Config Compiler

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/myir-imx6ulx-fb/4.1.15-2.0.1/sysroots/x86_64-pokysdk-linux/usr/bin/qt5", and choose "qmake" file. click the "Open" button then change "Version name" is "Qt %{Qt:Version} (mys6ulx-qt5)".After that click "Apply" button.

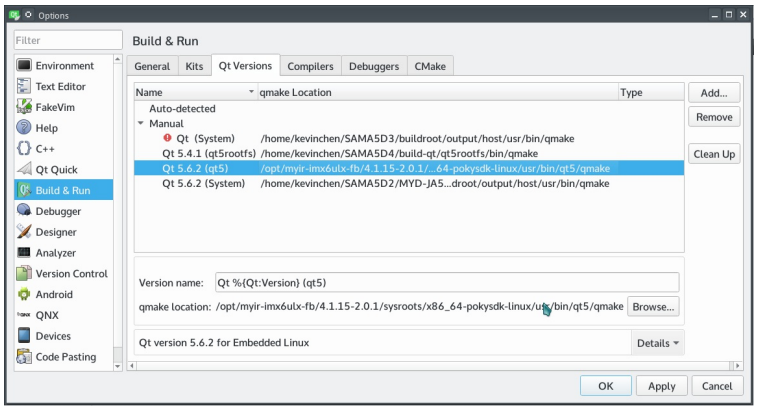


Figure5-2 Configure Qt version

The third step, click the "Device" menu on left panel, and click "Add..." button on right panel.Fill those input box, "Name" is "MYS6ULx Board", "Host name" is IP address of target board(alsa fill any one), "Username" is "root".Then clock "Apply" button.

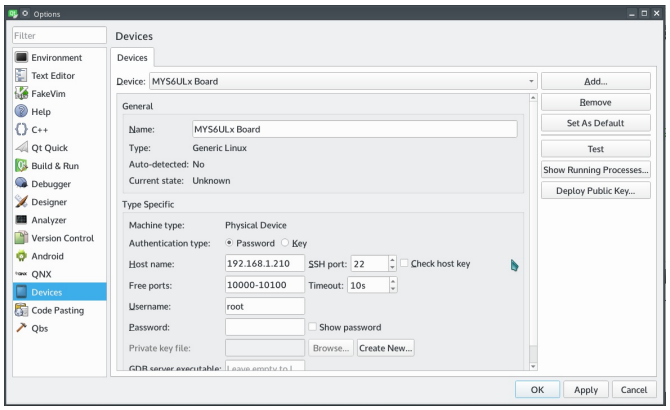


Figure5-3 Configure Qt version

The fourth step, click "Build & Run" menu on left panel will back to "Kit" tab in right panel.The content fill with "Name" is "MYS6ULx-dev-kit", "Device" choose "MYS6ULx 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 "MYS6ULx-GCC" before we configured."Qt version" choose "Qt 5.6.2 (mys6ulx-qt5)" before configured, "Qt mkspec" is "linux-oe-g++".Other use default option, then click"Apply" and "OK" button.

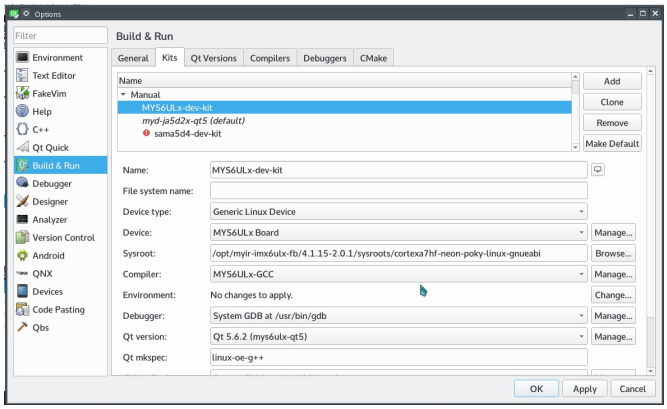


Figure5-4 Configure Kit

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 "helloworld.pro" file, click "Open" button.

The second step, choose "Projects" icon in left panel.The right panel will switch to "Build & Run" tab of "helloworld" project, click "Add kit" pop-down list, and choose "MYS6ULx-dev-kit" option.Then the "helloworld" project will use "MYS6ULx-dev-kit" option to build it.

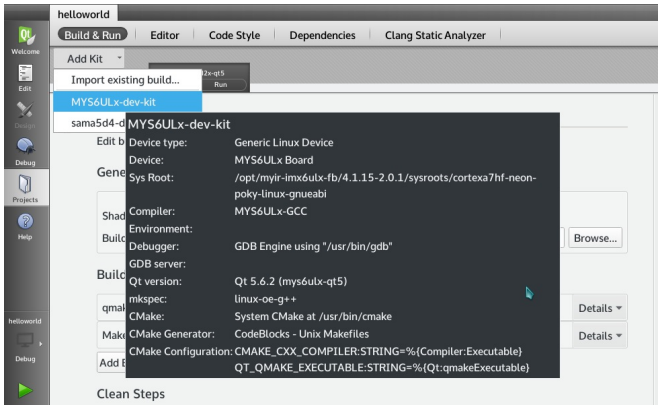


Figure5-3-1 config building option

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

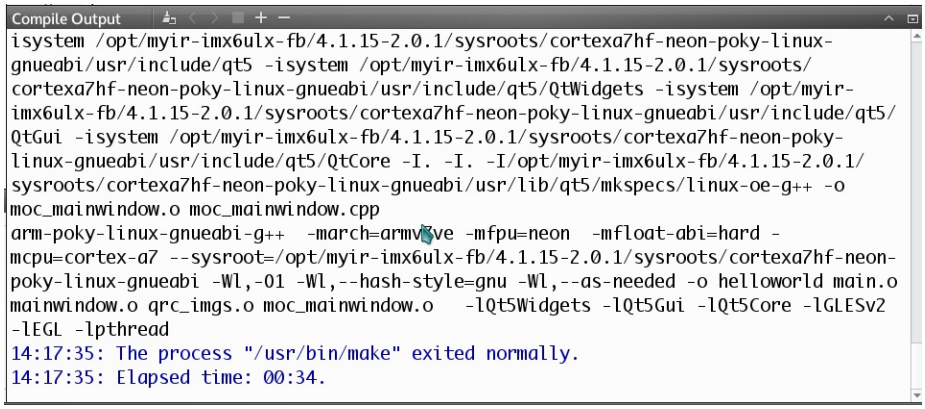


Figure5-3-2 Compling output

QtCreator build helloworld project, compiled binary files stored in the "~/build-helloworld-MYS6ULx_dev_kit-Debug/" directory, you can use the file command to see whether is the compiler for the ARM architecture.

```
$ file ~/build-helloworld-MYS6ULx_dev_kit-Debug/helloworld
/home/kevinchen/build-helloworld-MYS6ULx_dev_kit-Debug/
helloworld: 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 helloworld file to board and run it.

```
# ./helloworld -platform linuxfb
```

The LCD shows Qt windows of "Hello,World!" string。

Figure5-3-3 Run example program

6 System update

MYS-6ULX series board use two methods to update system to NAND flash on board, MfgTool and SD card.

MfgTool update method

Install tool

The NXP supports a manufacture tool called MfgTool, we use MfgTool 2.7.0 version.The MfgTool suppors Windows and Linux system.It is located in directory "03-Tools/ManufactoryTool" of resource package.You can copy and extract it to your work directory.

VBS FileName	Description
core-image-base-mys-6ulx-emmc.vbs	Write core-image-base system into MYS-6ULX board eMMC version
core-image-base-mys-6ulx-nand.vbs	Write core-image-base system into MYS-6ULX board NAND version
fsl-image-qt5-mys-6ulx-emmc.vbs	Write fsl-image-qt5 system into MYS-6ULX board eMMC version
fsl-image-qt5-mys-6ulx-nand.vbs	Write fsl-image-qt5 system MYS-6ULX board NAND version

Update steps(follow the sequence):

Attention: Please do not both plugin DV 5V(J1) and micro usb(J7) at the same time

- Change third bit as ON, four bit as OFF on toggle switch(SW1).
- Use USB cable(Type-A to Micro-B) connect to micro usb port(J7) with PC USB port.
- Double click file "mfgtool2-yocto-mx6ul-evk-nand.vbs" under MfgTool directory, then the MfgTool will show the HID device on reconigz.
- Click the "Start" button on MfgTool GUI, it will auto download system image to storage of board.

The progress bar will show as green when update finish. While it will show as red if failed.In this case you can view "MfgTool.log" file to get more information.Another way is use USB to TTL cable connect to Debug port(JP1), you can view the serial port output to analysis failed reason after update again.

Update files on MfgTool

MfgTool update files has two directories, firmware and files.The files directory store files for burn into flash of MYS-6ULX board. It's locate in "MYS-6ULX-mfgtools/Profiles/Linux/OS Firmware/files/". The firmware directory store files for burn system.It's locate in "MYS-6ULX-mfgtools/Profiles/Linux/OS Firmware/firmware/".You need to update those files when you change flash size or partition offset.

file list of MYS-6ULX-IOT board in files directory:

FileName	Description
core-image-base-mys6ull14x14.rootfs.tar.bz2	core-image-base filesystem
fsl-image-qt5-mys6ull14x14.rootfs.tar.bz2	fsl-image-qt5 filesystem
u-boot-imx6ull14x14evk-ddr256_emmc.imx	uboot support boot from eMMC and DDR 256MB
u-boot-imx6ull14x14evk-ddr256_nand.imx	uboot support boot from NAND and DDR 256MB
zImage-imx6ull	kernel image
zImage-imx6ull-14x14-evk-emmc.dtb	dtb supoort eMMC flash
zImage-imx6ull-14x14-evk-gpmi-weim.dtb	dtb support NAND flash

file list of MYS-6ULX-IND board in files directory:

FileName	Description
core-image-base-mys6ul14x14.rootfs.tar.bz2	core-image-base filesystem
fsl-image-qt5-mys6ul14x14.rootfs.tar.bz2	fsl-image-qt5 filesystem
u-boot-imx6ul14x14evk-ddr256_emmc.imx	uboot support boot from eMMC and DDR 256MB
u-boot-imx6ul14x14evk-ddr256_nand.imx	uboot support boot from NAND and DDR 256MB
zImage-imx6ul	kernel image
zImage-imx6ul-14x14-evk-emmc.dtb	dtb support eMMC flash
zImage-imx6ul-14x14-evk-gpmi-weim.dtb	dtb support NAND flash

Micro SD card update method

Because i.MX6ULL/i.MX6UL chip need kobs-ng to add some header data to bootloader, so it needs read or write under system. The MYS-6ULX series board support sdcard.gz image files.These files can be write into SD card and boot.When sdcard boot up, it will update files into flash of board.

The sdcard.gz image file needs special tool to write Micro SD storage card.The linux user can directly use gzip and dd command.The windows user need "USB Image Tool" to direct write into card.Another way is extract the sdcard.gz and get sdcard file use "Win32ImageWriter" tool write into card.

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 MYS-6ULX board support a tool MYS-6ULX-mkupdate-sdcard 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"
envfile="boot.scr"
kernelfile="zImage"
dtbfile="mys-imx6ull-14x14-evk-emmc.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.

```
sudo ./build-sdcard.sh -p mys6ull -n -d mfgimages-mys-imx6ul-ddr256m-nand256m
```

The tool support four arguments. '-p' stands for a platform, only two option 'mys6ull' and 'mys6ul'. '-n' stands for the storage device of NAND flash. '-e' stands for the storage device of eMMC flash. '-d' stands for target files directory.

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

After builds complete, a sdcard suffix file in current directory, 'mys6ull-update-nand-20170825150819.rootfs.sdcard'.

Build updatable Micro SD

Recommend insert Micro SD card to Card Reader, and plug into PC USB port.MYS-6ULX resource package support some prebuilt sdcard files.You can throught 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 real 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

```
sudo dd if=mys6ull-update-nand-20190919090957.sdcard of=/dev/sdb conv=fsync
```

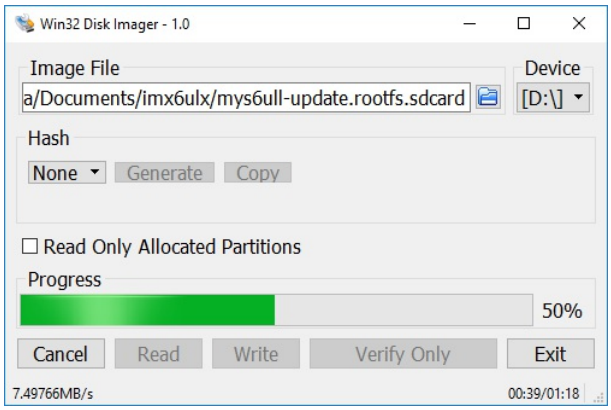
The write speed is relative with USB host version and Micro SD card write speed. We recommend use higher speed class Micro SD storage card.

- Windows system

The Windows user can use USB Image Tool application to write sdcard.gz image file to Micro SD storage card.Another way is extract the sdcard.gz file then 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.



You can plug out Card Reader after progress bard finish.

Take the Micro SD card insert into card slot(J5) on MYS-6ULX series boards.Then change boot toggle switch as SDCARD type:

NAND version boot from SDCard:

Boot Bit	Status
Bit1	OFF
Bit2	ON
Bit3	OFF
Bit4	ON

eMMC version boot from SDCard:

Boot Bit	Status
Bit1	OFF
Bit2	OFF

Bit3	OFF
Bit4	ON

Use USB to TTL cable connect to Debug port(JP1), configure your serial terminal software.Use USB Micro B cable as power plug into USB OTG port(J7) on board(or use DC adapter plug into J1 interface).You can view update progress in serial terminal software.

Also, you can through LED(D12) to view the current update status, the updating is flash, the update success is light on, the update fail is light off.

Boot from Flash of board

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

Boot from NAND flash

Boot Bit	Status
Bit1	ON
Bit2	OFF
Bit3	OFF
Bit4	ON

Boot from eMMC flash

Boot Bit	Status
Bit1	ON
Bit2	ON
Bit3	OFF
Bit4	ON

Reconnect the power adapter, the board will boot into linux on NAND 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