MYD-Y6ULX-HMI Linux 开发手册

MYiR™ Make Your Idea Real

目錄

MYD-Y6ULX-HMI Linux开发手册

本文档介绍基于MYD-Y6ULX-HMI系列板子，进行Linux系统的编译和安装，硬件接口的使用，Qt应用开发等。

历史版本

版本号	描述	时间
V1.0	初始版本	2018.10.28

硬件版本

本手册适合于以下型号的板子：

- MYD-Y6ULX-HMI 开发板
- MYD-Y6ULX-CHMI开发套件
- MYD-Y6ULX-HMI-4GEXP扩展板

以下文章使用MYD-Y6ULX-HMI代指,请注意相关的硬件资源说明.

1 软件资源

MYD-Y6ULX-HMI搭载基于Linux 4.1.15内核的操作系统，提供了丰富的系统资源和软件资源。部分资源需要配合相应的扩展模块才能使用。

- 注意: MYD-Y6ULX-HMI出厂时默认烧写了一套DEMO系统，Demo是基于QT5开发的UI，另外带有Tornado Web Server,使用yocto构建出的也是这套系统.
  - DEMO系统代码：
    - 04-Source/HMI-QT5-DEMO.tar.bz2
    - 04-Source/web_server.tar.bz2
  - DEMO系统说明文档：
    - 01-Document/User_Manual/Chinese/MEasy HMI开发手册.pdf

以下是软件资源列表：

类别	名称	描述信息	源码
引导程序	U-boot	u-boot.imx引导启动程序	YES
Linux内核	Linux 4.1.15	基于官方imx_4.1.15_2.0.0_ga版本	YES
设备驱动	USB Host	USB Host驱动	YES
设备驱动	USB OTG	USB OTG驱动	YES
设备驱动	I2C	I2C总线驱动	YES
设备驱动	Ethernet	10/100Mbps以太网驱动	YES
设备驱动	MMC	MMC/eMMC/TF卡存储驱动	YES
设备驱动	LCD	显示驱动，7寸液晶屏	YES
设备驱动	RTC	实时时钟驱动	YES
设备驱动	Touch	电容(FT5316)，电阻触摸	YES
设备驱动	USART	串口驱动	YES
设备驱动	Audio	WM8904驱动	YES
设备驱动	CAN bus	CAN总线驱动	YES
设备驱动	RS485	RS485总线驱动	YES
设备驱动	Camera	OV2659驱动	YES
设备驱动	WiFi & BT	AP6212，SDIO接口WiFi，串口蓝牙	YES
设备驱动	LTE模块	提供移远EC20驱动支持，使用USB驱动，可支持GPS	YES
文件系统	Yocto rootfs	基于Yocto构建带Qt 5.6的文件系统	YES
文件系统	Yocto rootfs	基于Yocto构建终端型的通用文件系统	YES
应用程序	GPIO KEY	GPIO按键指示例程	YES
应用程序	GPIO LED	按键指示灯例程	YES
应用程序	NET	TCP/IP Socket C/S例程	YES
应用程序	RTC	实时时钟例程	YES
应用程序	RS232	RS232例程	YES

应用程序	RS485	RS485例程	YES
应用程序	Audio	Audio例程	YES
应用程序	LCD	显示屏例程	YES
编译工具链	Cross compiler	Linaro GCC 4.9 Hardfloat	BINARY
编译工具链	Cross compiler	Yocto GCC 5.3 Hardfloat	BINARY

表1-1软件资源列表

2 部署开发环境

开发前需要PC安装好Linux操作系统，推荐使用Ubuntu 16.04 64bit发行版，连接网线并配置好网络，后续操作需要连接互联网安装或下载相关软件包。

开发板与计算机连接

1. 计算机使用USB转TTL串口转接线与开发板的DEBUG串口(JP1)连接
2. 运行串口调试应用程序，并选择对应的串口设备
3. 开发板用户名为root,密码为空

计算机端的串口配置参数如下：

- 波特率：115200
- 数据位: 8bit
- 校验方式：None
- 停止位：1bit
- 流控：Disable

产品图片

核心板型号有多种，列表如下：

- MYC-Y6ULY2-256N256D-50-C (标准品)
- MYC-Y6ULY2-256N256D-50-I (标准品)
- MYC-Y6ULY2-512N256D-50-C (非标准品,需提前预定)
- MYC-Y6ULY2-4E512D-50-C (标准品)
- MYC-Y6ULY2-4E512D-50-I (标准品)
- MYC-Y6ULY2-4E256D-50-I (非标准品,需提前预定)

图2-1 MYC-Y6ULX 正面图

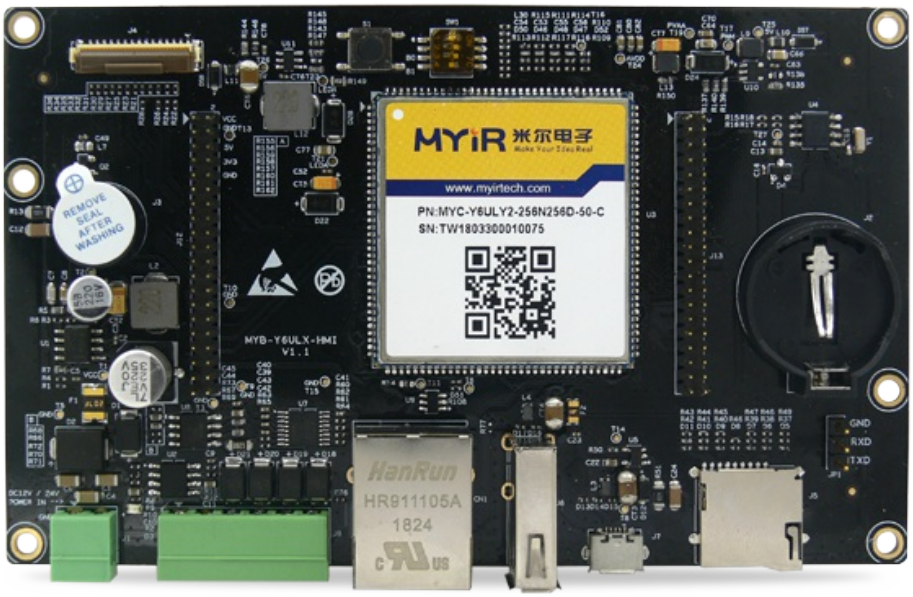

图2-2 MYD-Y6ULX-HMI 正面图

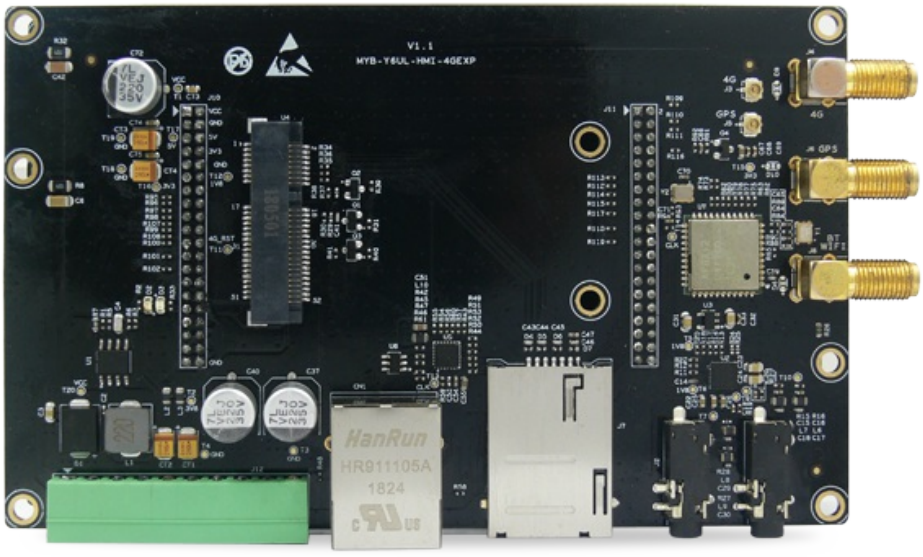

图2-3 MYB-Y6ULX-HMI-4GEXP 正面图

安装必备软件包

```
$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 python git \
make gcc g++ diffstat bzip2 gawk chrpath wget cpio texinfo lzop
```

建立工作目录

建立工作目录，方便设置统一的环境变量路径。拷贝产品光盘中的源码到工作目录下，同时设置DEV_ROOT变量，方便后续步骤的路径访问。

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

配置编译工具

- Linaro交叉编译器: gcc version 4.9.3 20141031 (prerelease) (Linaro GCC 2014.11)
- Yocto交叉编译器: gcc version 5.3.0 (GCC)

这里有两个编译器，一个是Linaro提供，另一个是由Yocto构建的，建议使用Yocto提供的，以便和文件系统统一。

Linaro编译器

```
$cd $DEV_ROOT
$cd 03-Tools/Toolchain
$tar -xvf 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
```

执行完上述命令后输入"arm-linux-gnueabihf-gcc --version"，若有输出版本信息，说明设置成功，以上设置只对当前终端有效。如需永久修改，请修改用户配置文件。

```
$ 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编译工具链

Yocto提供的工具链有两种，一种是底层开发的meta-toolchain，另一种是用于应用开发的工具链。前者和Linaro类似，后者包含应用开发中的相关库，可以直接使用pkg-config工具来解决头文件或库文件的依赖关系。MYD-Y6ULX-HMI的资源包中有两种工具链。

应用层编译工具：

工具链文件名	描述
myir-imx-fb-glibc-x86_64-fsl-image-qt5-cortexa7hf-neon-toolchain-4.1.15-2.0.1.sh	fsl-image-qt5系统的应用工具链
myir-imx-fb-glibc-x86_64-core-image-base-cortexa7hf-neon-toolchain-4.1.15-2.0.1.sh	core-image-base系统的应用工具链

底层编译工具：

工具链文件名	描述
myir-imx-fb-glibc-x86_64-meta-toolchain-cortexa7hf-neon-toolchain-4.1.15-2.0.1.sh	meta-toolchain基础工具链

Yocto编译器是以SDK工具包方式来提供，需要先安装SDK包后，才可以使用。安装方法如下：

以普通用户权限执行shell脚本，运行中会提示安装路径，默认在/opt目录下，同时会提示输入用户密码以便有写入目录的权限。安装完成后，可以使用"source"或"."命令加载工链接环境到当前终端。

本例是把应用开发工具链安装在了/opt/myir-imx6ulx-fb/4.1.15-2.0.1目录下。

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

验证SDK工具链是否安装正确，先使用"source"命令加载Yocto的环境配置文件，然后查看编译器版本。

```
$ 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.
```

同样方法请自行安装底层开发的工具链meta-toolchain。安装两个工具链，请指定不同目录，请勿使用相同目录，出现文件相互覆盖情形。

3 构建系统

本章主要介绍在MYD-Y6ULX-HMI上, Linux操作系统相关部件的编译和使用。MYD-Y6ULX-HMI的Linux系统包含以下部件:

- U-Boot: 引导程序，支持不同方式启动内核。
- Linux Kernel: 适用于MYD-Y6ULX-HMI板的Linux 4.1.15内核，同时包含支持板载外设的驱动。
- Yocto: 一个开源协作项目，提供丰富的模板、工具和方法来支持构建出面向嵌入式产品的自定义Linux系统。

本章中用到的代码存放在资源包04-Source目录下，编译u-boot和Linux内核代码前，请先安装meta-toolchain并加载环境变量到当前shell。

3.1 U-Boot

进入Bootloader目录，解压U-boot源码：

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

开始编译：

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

这里的是配置选项名称，不同的核心板需使用不同的配置选项：

核心板	编译选项
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卡方式启动时默认会先检测"boot.scr"文件，这是u-boot上的脚本镜像文件，用于临时改变启动设备顺序。以下是从TFTP下载zImage和dtb文件并启动的脚本例子。使用mkimage工具"myd-y6ull-boot-mmc0-tftp.txt"文件制做成"boot.scr"文件，mkimage工具是在u-boot的tools目录下，u-boot编译完成后，mkimage也会被编译出来，直接使用即可。

以下是创建的myd-y6ull-boot-mmc0-tftp.txt内容，并生成为启动脚本文件boot.scr。

myd-y6ull-boot-mmc0-tftp.txt的完整内容如下：

```
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
```

生成boot.scr命令如下：

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

3.2 Linux Kernel

进入Kernel目录，解压内核源码：

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

开始编译：

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

编译完成后在"arch/arm/boot"目录会生成内核镜像文件zImage，在"arch/arm/boot/dts"目录会生成DTB文件。

DTB、核心板、底板、扩展板对应关系如下：

DTB	核心板	底板	扩展板
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	无
myd-y6ull-gpmi-weim-without-exp.dtb	MYC-Y6ULY2-256N256D-50-I	MYB-Y6ULX-HMI	无
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	无
myd-y6ull-gpmi-weim-without-exp.dtb	MYC-Y6ULY2-4E512D-50-I	MYB-Y6ULX-HMI	无

MYD-Y6ULX-HMI板上的Micro SD卡槽是连接mmc0控制器，所有的dtb文件都是默认启用mmc0控制器。

更新kernel后，由于版本标识改变，若驱动是以模块方式加载，需要重新编译驱动模块:

```
make modules
```

编译后，可以安装在指定位置：

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

这样就可以把target-kernel目录打包后，解压在MYD-Y6ULX-HMI开发板的/lib目录下使用。

3.3 构建文件系统

Linux系统平台上有许多开源的系统构建框架，这些框架方便了开发者进行嵌入式系统的构建和定制化开发，目前比较常见的有Buildroot, Yocto, OpenEmbedded等等。其中Yocto项目使用更强大和定制化的方法，来构建出适合嵌入式产品的Linux系统。

Yocto不仅仅是一个制做文件系统工具，同时提供整套的基于Linux的开发和维护工作流程，使底层嵌入式开发者和上层应用开发者在统一的框架下开发，解决了传统开发方式下零散和无管理的开发形态。

Yocto是一个开源的“umbrella”项目，意指它下面有很多个子项目，Yocto只是把所有的项目整合在一起，同时提供一个参考构建项目Poky，来指导开发人员如何应用这些项目，构建出嵌入式Linux系统。它包含Bitbake, OpenEmbedded-Core, 板级支持包，各种软件包的配置文件。通过Poky，可以构建出不同类需求的系统，如最小的系统core-image-minimal、全功能命令行系统core-image-base、带Qt5图形库的fsl-image-qt5。

MYD-Y6ULX-HMI提供了符合Yocto的配置文件，帮助开发者构建出可烧写在MYD-Y6ULX-HMI板上的Linux系统镜像。

Yocto还提供了丰富的开发文档资源，让开发者学习并定制自己的系统。由于篇幅有限，不能完整介绍Yocto的使用方法，建议开发者先阅读以下文档后，再开始动手构建。

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 Yocto构建Linux系统

本节适合需要对文件系统进行深度定制的开发者，希望从Yocto构建出符合MYD-Y6ULX-HMI系列开发板的文件系统，同时基于它的定制需求。初次体验使用或无特殊需要的开发者可以直接使用MYD-Y6ULX-HMI已经提供的文件系统。

由于Yocto构建前需要下载文件系统中所有软件包到本地，为了快速构建，MYD-Y6ULX-HMI已经把相关的软件打包好，可以直接解压使用，减少重复下载的时间。

注意：构建Yocto不需要加载工具链环境变量，请创建新shell或打开新的终端窗口。

MYD-Y6ULX-HMI提供的Yocto

解压Yocto源码包，同时解压Yocto-downloads.tar.xz软件包至Yocto目录下。Yocto-downloads.tar.xz是把Yocto构建中用到的第三方软件包打包，免除开发者再次下载花费的时间。

注意：由于Yocto-downloads.tar.xz文件较大，无法与MYD-Y6ULX-HMI打包在同一文件内，请访问网页下载: http://down.myir-tech.com/MYD-Y6ULX-HMI/。文件名为Yocto-downloads.tar.xz。

```
$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
```

还需要将Linux内核和U-Boot代码放在用户家目录下，方便开发和Yocto编译。

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

初始化Yocto构建目录

使用NXP提供的fsl-setup-release.sh脚本，会创建一个工作空间，然后在此空间下构建镜像。执行脚本后会先要求阅读并同意版权声明后才会进入构盡过目录。同时，脚本会默认创建并进入build目录。如果需要特定目录名称，可以使用-b参数，如"-b myir"。 MACHINE可选参数为"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/conf目录下是当前构建的配置文件。上面在初始化后，就可以构建适合"myd-y6ull-hmi"的镜像了。

构建GUI Qt5版的系统

第一次构建时，会需要很长时间，请耐心等待。

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

构建非GUI版的系统

第二次构建时，如果是同设备，不需要修改其它文件，直接编译即可。

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

Image名称	描述	用途
core-image-minimal	minimal版本的文件系统	用于MYD-Y6ULX的最小系统
core-image-base	base版本的终端更多功能的镜像	通用的文件系统
fsl-image-qt5	构建基于Qt5的镜像	带Qt5的通用文件系统

在建文件系统完成后，会在输出目录下有manifest文件，这个文件里包含了对应文件系统中已安装的软件包。

Yocto第一次构建会需要很长时间，取决于计算机的CPU核心数和硬件读写速度。Yocto建议可以使用八核和SSD硬盘可以加速构建速度。第一次构建完成后会生成缓存，后面修改的构建，时间会减少很多。

在构建完成后在会"tmp/deploy/images/myd-y6ull-hmi/"目录下生成不同的文件，以下是构建后的一个例子：

```
$ ls -lh tmp/deploy/images/myd-y6ull-hmi/
total 1.4G
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
```

生成的文件中，有一些是链接文件，下面是不同文件的用途：

文件名	用途
*.rootfs.manifest	文件系统内的软件列表
*.rootfs.ext4	打包成ext4格式的文件系统
*.rootfs.sdcard	可直接写入SD卡，从SD卡启动的镜像
*.rootfs.tar.bz2	打包成tar.bz2格式的文件系统
*.rootfs.tar.xz	打包成tar.xz格式的文件系统
u-boot-sd-2016.03-r0.imx	适合从SD启动的u-boot镜像
u-boot-nand-2016.03-r0.imx	适合从NAND启动的u-boot镜像

Bitbake常用命令

Bitbake 参数	描述
-c fetch	从recipe中定义的地址，拉取软件到本地
-c cleanall	清空整个构建目录
-c deploy	部署镜像或软件包到目标rootfs内
-k	有错误发生时也继续构建
-c compile	重新编译镜像或软件包

更多Yocto使用方法，请参考NXP官方Yocto使用文档《i.MX Yocto Project User's Guide》。

3.3.2 Yocto构建SDK工具

Yocto提供可构建出SDK工具的功能，用于底层或上层应用开发者使用的工具链和相关的头文件或库文件，免去用户手动制做或编译依赖库。SDK工具有两种，一种是适合底层开发的工具链，用于编译u-boot和linux内核代码，另外一种是应用开发工具链，附带目标系统的头文件和库文件，方便应用开发者移植应用在目标设备上。两种SDK工具都是shell自解压文件，执行后，默认安装在/opt目录下。

构建底层工具连

```
bitbake meta-toolchain
```

fsl-release-yocto-hmi构建完成后，在"tmp/deploy/sdk"目录下有三个文件：

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

这里有两个manifest文件，host.manifest是工具链中包含主机端的软件包的列表，target.manifest是包含目标设备端的软件包列表。

构建应用层工具链

应用层工具链是和Image名称是统一的，这里可以使用"fsl-image-qt5"和"core-image-base"两种参数。

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

构建完成后，同样在"tmp/deploy/sdk/"目录下有六个文件：

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

"*.host.manifest*"文件表示工具链中包含主机端的软件包列表，".target.manifest"表示工具链中包含目标设备端的软件包列表。"myir-imx-fb-glibc-x86_64-fsl-image-qt5-cortexa7hf-neon-toolchain-4.1.15-2.0.1.sh"文件是构建出的fsl-image-qt5镜像对应的SDK工具链，"myir-imx-fb-glibc-x86_64-core-image-base-cortexa7hf-neon-toolchain-4.1.15-2.0.1.sh"文件是构建出的core-image-base镜像对应的SDK工具链。可以直接安装在其他Linux系统中，开发和编译目标端设备的二进制程序。

3.3.3 Yocto常见问题

Q1:Yocto构建时出现U-Boot编译问题

编译错误信息如下：

```
WARNING: u-boot-mys6ulx-2016.03-r0 do_fetch: Failed to fetch URL  \
git:////home/KevinChen/MYiR-iMX-uboot;protocol=file;branch=mys-6ulx,  \
attempting MIRRORS if available
ERROR: u-boot-mys6ulx-2016.03-r0 do_fetch: Fetcher failure: \
Unable to find revision 38cc50da55a891ba0ac09346f3a7a6cbb4a20970  \
in branch mys-6ulx even from upstream
ERROR: u-boot-mys6ulx-2016.03-r0 do_fetch: Function failed: \
Fetcher failure for URL: 'git:////home/KevinChen/MYiR-iMX-uboot;i \
protocol=file;branch=mys-6ulx'.  \
Unable to fetch URL from any source.
ERROR: Logfile of failure stored in: /home/KevinChen/MYD-Y6ULX-devel/fsl- \
release-yocto/myd-y6uly2/tmp/work/myd_y6ull14x14-poky-linux-gnueabi/ \
u-boot-mys6ulx/2016.03-r0/temp/log.do_fetch.4991
ERROR: Task 206 (/home/KevinChen/MYD-Y6ULX-devel/fsl-release-yocto/sources \
/meta-myir-imx6ulx/recipes-bsp/u-boot/u-boot-mys6ulx_2016.03.bb, \
do_fetch) failed with exit code '1'
NOTE: Tasks Summary: Attempted 3116 tasks of which  \
3115 didn't need to be rerun and 1 failed.
No currently running tasks (1771 of 3136)

Summary: 1 task failed:
  /home/KevinChen/MYD-Y6ULX-devel/fsl-release-yocto/sources/ \
  meta-myir-imx6ulx/recipes-bsp/ \
  u-boot/u-boot-mys6ulx_2016.03.bb, do_fetch
Summary: There were 3 WARNING messages shown.
Summary: There were 2 ERROR messages shown, returning a non-zero exit code.
```

出现这种情况是由于U-Boot更新后，Yocto没有更新的对应的版本。 修改步骤如下：

1. 进入U-Boot源码目录，使用"git log"查看commit id并复制
2. 修改Yocto源码目录下的"source/meta-myir-imx6ulx/recipes-bsp/u-boot/u-boot-mys6ulx_2016.03.bb"文件中的SRCREV变量为commit id即可。

Q2:Yocto构建时出现Linux kernel编译问题

此问题和上一个问题相似，同样是Linux Kernel更新后，Yocto没有更新版本。修改步骤如下：

1. 进入Linux源码目录，使用"git log"查看commit id并复制
2. 修改Yocto源码目录下的"source/meta-myir-imx6ulx/recipes-kernel/linux/linux-mys6ulx_4.1.15.bb"文件中的SRCREV变量为commit id即可。

Q3:如何重新加载已存在的构建目录

第一次构建时，使用以下命令来创建一个构建目录build，创建完成后会在build目录下。

```
DISTRO=myir-imx-fb MACHINE=myd-y6ull14x14 source fsl-setup-release.sh -b build
```

Yocto中途退出或中断后，可以重新打开新的shell终端，重新加载build构建目录，命令如下：

```
source fsl-setup-release.sh build
```

4 Linux应用开发

本章主要介绍MYD-Y6ULX-HMI板外围硬件设备应用例程的使用。

使用前，需要先安装Yocto提供的SDK工具链，再编译所有例程代码，并拷贝至开发板目录下。

编译应用例程

加载工具链到当前终端后，可以查看gcc的版本信息，确认当前环境已正确加载。

```
$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.
```

编译示例代码：

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

4.1 LCD测试

本例程演示对Linux的FrameBuffer设备操作，实现液晶输出显示RGB颜色和颜色合成测试。例程基于Linux FrameBuffer API接口开发。测试前需要把LCD连接至J9接口上。

米尔科技提供两种LCD模块，7寸电容屏和7寸电阻屏，默认配置使用为七寸电容屏。

执行程序后，LCD液晶屏会出现相应颜色,以下是终端输出信息:

```
# ./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
```

MYD-Y6ULX-HMI板中提供的Linux代码已经支持这两种模块的显示和触摸功能，电阻屏和电容屏的触摸功能有所区别，电阻屏是通过ADC采集，电容屏使用通过I2C通讯读取信息，dts代码中已配置好,只需要启用相应功能即要可。

使用电阻屏时修改tsc的status属性为okay。 编辑相应的设备树文件
i.MX6ULL："arch/arm/boot/dts/myb-y6ull-14x14.dts"
i.MX6UL："arch/arm/boot/dts/myb-y6ul-14x14.dts "

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

4.2 触摸测试

MYD-Y6ULX-HMI支持两种触摸方式，电容触摸和电阻触摸。米尔科技提供两种带触摸的液晶，7寸电容屏和7寸电阻屏,电容屏不需要校准，电容屏需要校准。

电容屏的触摸测试使用ts_calibrate和ts_test命令，ts_calibrate用于触摸校验，ts_test用于测试。其中，命令需要变量"TSLIB_TSDEVICE"来找到触摸设备，不同触摸方式的设备节点不一定相同，请对应设备填写。

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

# ts_calibrate
//测试触摸屏
# ts_test
```

4.3 Ethernet 测试

本例使用Linux sokect API，实现简单的C/S结构的程序，两个程序通过TCP/IP协议栈通信。将可执行程序arm_client拷贝至开发板，pc_server拷贝至PC，将开发板和PC接入网络。

这里以ETH0为例介绍,ETH1使用相同。 在 PC 上配置IP并运行服务程序:

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

在开发板上运行客户程序，将看到所发送的信息:

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

4.4 USB Host 测试

使用USB存储设备插入USB HOST(J6)接口，调试串口会输出检测设备信息。同时，使用将此存储设备挂载至linux系统下对其读写。

```
# 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 USB Device测试

本例程演示使用开发板的Micro USB接口(J7)作为Device模式，可以将指定的文件或设备模拟为设备，连接到其它USB HOST设备。这里把内存作为存储设备提供给HOST设备。

- 开发板上使用modprobe加载驱动:

```
#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
```

- Linux PC机上查看到有USB设备接入，SerialNumber为"1234"，Manufacturer是内核构建版本号:

```
#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, Produ
ct=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+g8d9
8da6 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 RS485 测试

本例程演示如何使用 Linux Serial API编程，使用RS485接口，实现数据发送和接收，详细请参考源码。

硬件连接

MYD-Y6ULX-HMI上有一个RS485接口(J8)，J8的pin.3为RS485的A,J8的pin.5为RS485的B，将RS485的A，B信号线与另外的RS485设备相连。或者是USB转RS485的转换器设备。

软件测试

将编译出来的可执行程序拷贝至MYD-Y6ULX-HMI板系统内。MYD-Y6ULX-HMI作为发送端执行以下命令，另外一端的设备可以接数据。

```
# ./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
```

另一个设备作为接收端：

```
# ./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 RS232 测试

本例程演示如何使用 Linux Serial API编程，使用RS232接口，实现数据发送和接收，详细请参考源码。

硬件连接

MYD-Y6ULX-HMI上有一个RS232接口(J8),J8的pin.5为RS232的TXD,J8的pin.6为RS232的RXD,将TXD，RXD信号线与另外的RS232设备相连接。或者是USB转RS232的转换器设备,此处使用USB转RS232设备连接到PC。

软件测试

将编译出来的可执行程序拷贝至MYD-Y6ULX-HMI开发板系统内。MYD-Y6ULX-HMI作为发送端执行以下命令，PC端接收数据。

```
# ./uart_test -d /dev/ttymxc1 -b 115200
/dev/ttyS1 RECV 10 total
/dev/ttyS1 RECV: 1234567890
/dev/ttyS1 RECV 10 total
/dev/ttyS1 RECV: 1234567890
/dev/ttyS1 RECV 10 total
/dev/ttyS1 RECV: 1234567890
/dev/ttyS1 RECV 10 total
/dev/ttyS1 RECV: 1234567890
```

在PC端可以使用串口工具查看接收的数据。

4.8 Camera 测试

MYD-Y6ULX-HMI上提供一个并行Camera接口(J9)，可以连接MYB-CAM011B型号的Camera模块，模块之间使用FPC线连接。由于信号序列影响，请勿直接将其它型号的Camera的模块插入，否则会引起模块或开发板的损坏。

本例程演示使用一款开源的视频流软件uvc_stream，可以将Camera设备捕捉的数据显示在web页面。

硬件连接

使用FPC数据线将MYB-CAM011B模块和MYD-Y6ULX-HMI板上的J9接口相连接。

软件操作

uvc_stream是通过的网络传输数据，需要先设置好MYD-Y6ULX-HMI板的以太网IP地址，对应系统中的eth0设备。Linux系统中的MY-CAM011B模块的设备，可通通过v4l2-ctl命令来查询到，输出信息的i.MX6S_CSI表示Camera控制器，对应设备是/dev/video1。uvc_stream参数中'-y'是使用yuyv方式，'-P'后面是设置web界面的登录密码，用户名默认为uvc_user。'-r'是指定分辨率。可以用ctrl + c来停止。

```
# 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 -r 800x600
```

uvc_stream提供两种web功能，snapshot和streaming。snapshot的请求URL是snapshot.jpeg，streaming的请求URL是stream.mjpeg。 PC和开发板在同一网络内时，打开流览器，输入地址http://192.168.1.42:8080/stream.mjpeg，可以看到有登录框，输入用户名为uvc_user，密码为123456，就可以看到从MY-CAM011B实时采集到的图像了。

4.9 Audio 测试

硬件连接

- 注意： Audio需要配合MYD-Y6ULX-HMI-4GEXP扩展板使用。

本例程演示使用Linux系统中的arecord/aplay命令对音频接口录音和放音。需要使用两头3.5mm的音频AUX线，将电脑音频输出孔和开发板的LINE IN(J2)接口连接，HEADERPHONE(J1)连接耳机。

软件操作

在电脑中播放音频文件，执行arecord命令会先将LINE IN中的音频录制并保存为test.wav文件。运行一分钟后再按ctrl + c来停止。

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

执行aplay命令来播放上面录制好的音频文件。

```
# aplay test.wav
```

4.10 WiFi 测试

注意：板载的WiFi功能，仅限于NAND版本,且需要配合MYD-Y6ULX-HMI-4GEXP扩展板使用

MYD-Y6ULX-HMI开发板提供一个WiFi模块(U7)，支持Client和AP模式。

硬件连接

将附带SMA接口的天线安装在开发板的J8位置。

Client模式

Client模式是用于将WiFi模块作为客户端设备，主动连接于路由器或其它提供无线热点的设备。

系统中已经加入WiFi模块的驱动，启动后会自动加载相应驱动。驱动加载成功后会出现对应的wlan0网络设备，使用ifconfig命令确认。

```
# 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)
```

下面实现WiFi模块与WiFi热点的链接,设置WiFi热点的用户名和密码， /etc/wifi-conf/WIFI.CONF是给出的例子，根据你的实际情况修改。 修改完成后进行连接，命令如下：

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

连接成功后，会自动获取IP，查看连接和分配的IP

```
~# 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模式

将WiFi模块作为热点，udhcpd.conf和hostapd.conf中配置了AP的相关的信息，用户可自行修改。
开启热点：

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

开启热点后可以在手机上搜索到， SSID为MYIR-WIFI-AP；连接密码为MYIR-TECH。 查看相关信息：

```
# 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 4G 测试

MYD-Y6ULX-HMI开发板提供一个支持4G模块的MINI PCI-E插槽，此插槽使用USB接口与4G模块通讯。目前提供移远EC20的驱动支持。

注意：4G功能需要配合MYD-Y6ULX-HMI-4GEXP扩展板使用

硬件连接

- 安装移远EC20模块到PCI-E插槽(U4)。
- 将两头I-PEX接口的天线安装在移远EC20模块和开发板的J3位置。
- 安装SMA天线到开发板的J4位置。

系统中已经加入4G模块的驱动，启动后会自动加载相应驱动,驱动加载成功后会出现对应的/dev/ttyUSB*设备，查看：

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

系统中已加入ppp软件包，可以直接使用。启用ppp0后会自动拨号，连接成功后即获得IP地址，D3灯常亮。还需要检查/etc/resolve.conf文件中的DNS是否设置正常。

```
# 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
```

然后使用ping命令测试连接4G网络是否正常。

```
# ping myir-tech.com
PING s-26427.gotocdn.com (118.123.18.103) 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
```

如果上面验证步骤有异常，可以查看运行日志。

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

5 QT应用开发

Qt是一个跨平台的图形应用开发框架，被应用在不同尺寸设备和平台上，同时提供不同版权版本供用户选择。MYD-Y6ULX-HMI使用Qt 5.6.2版本进行应用开发。

在Qt应用开发中，推荐使用QtCreator集成开发环境，可以在Linux PC下开发Qt应用，自动化地交叉编译为开发板的ARM架构。

本章使用Yocto构建的SDK工具作为交叉编译系统，配合QtCreator快速开发图形类应用程序。开始本章前，请先完成第三章的Yocto构建过程。或者使用光盘中提供的预编译好的SDK工具包。本章开始前，请安装好应用SDK开发工具。

5.1 安装QtCreator

QtCreator安装包是一个二进制程序，直接执行就可以完成安装。

```
cd $DEV_ROOT
chmod a+x 03-Tools/qt-creator-opensource-linux-x86_64-4.1.0.run
sudo 03-Tools/Qt/qt-creator-opensource-linux-x86_64-4.1.0.run
```

执行安装程序后，一直点击下一步即可完成。默认安装目录在"/opt/qtcreator-4.1.0"。

安装完成后，为了让QtCreator使用Yocto的SDK工具，需要对QtCreator加入环境变量。修改"/opt/qtcreator-4.1.0/bin/qtcreator.sh"文件，在"#! /bin/sh"前加入Yocto的环境配置即可，参考如下：

- 注意：作者安装QT的交叉编译工具链时，目录名作了修改（/opt/qt5-hmi）, 选择工具链时，以你安装的目录名为准。

```
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.
```

使用QtCreator时，请从终端执行"qtcreator.sh"来启动QtCreator，参考如下：

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

5.2 配置QtCreator

第一步，运行 QtCreator 后，依次点击"Tool"->"Options"，出现选项对话框，在左侧点击"Build & Run"，右边选择"Compilers"标签。 点击右侧"Add"按钮，弹出下拉列表后，选择"GCC"，在下面填写"Name"为"MYD-Y6ULX-HMI-GCC"，"Compiler path"点击旁边的"Browse.."按钮选 择到arm-poky-linux-gnueabi-g++的路径，例子中的路径是"/opt/qt5-hmi/sysroots/x86_64-pokysdk-linux/usr/bin/arm-poky-linux-gnueabi/arm-poky-linux-gnueabi-g++"。填写完成后，点击"Apply"。

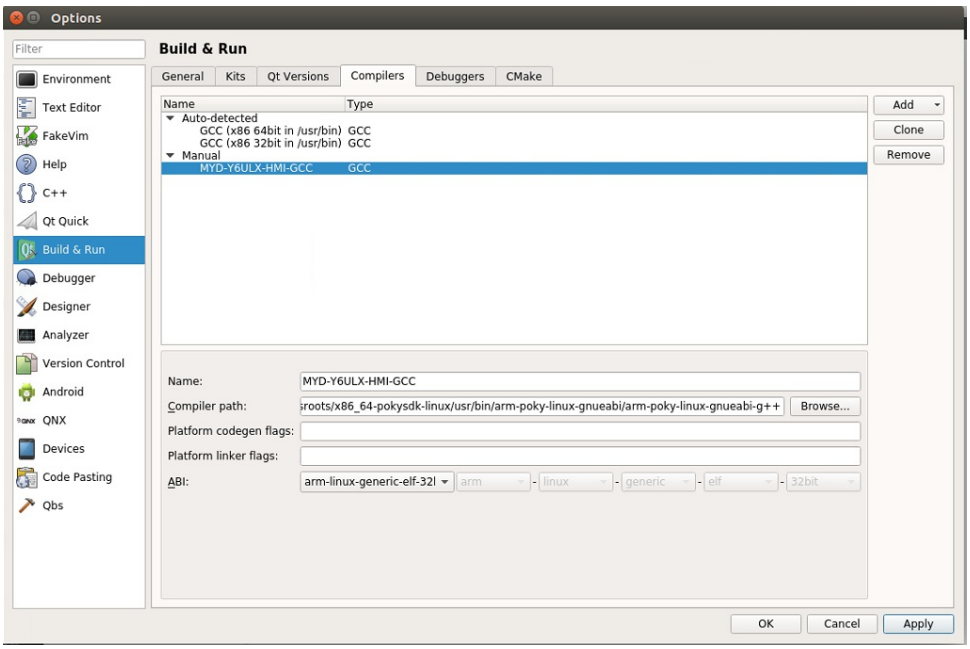

图5-2-1 配置编译器

第二步，选择"Qt Version"标签，在右侧点击"Add..."，会弹出qmake路径选择对话框，这里以"/opt/qt5-hmi/sysroots/x86_64-pokysdk-linux/usr/bin/qt5/qmake"为例子。选择”qmake”文件后，点击"Open"按钮。"Version name"改为"Qt %{Qt:Version} (MYD-Y6ULX-HMI-QT5)"。然后点击"Apply"按钮。

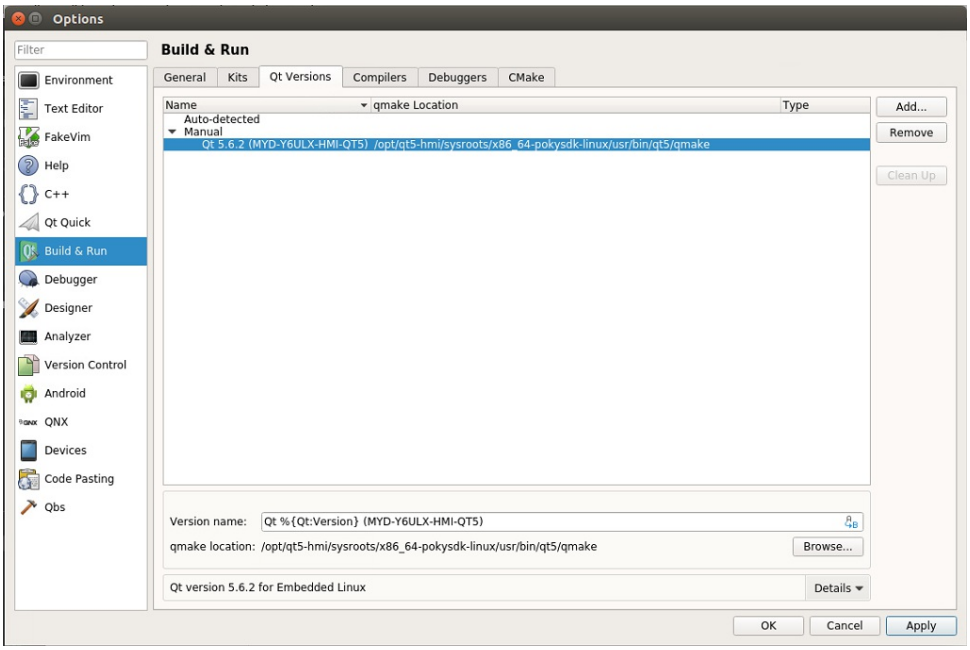

图5-2-2 配置Qt版本

第三步，选择左侧"Device"，点击右边的"Add..."按钮，在弹出的对话框中选择Generic Linux Device,再填写内容"Name"为"MYD-Y6ULX-HMI BOARD "，"Host name"为开发板的IP地址(可以暂时填写任意一个地址)，"Username"为"root"，然后点击"Apply"。

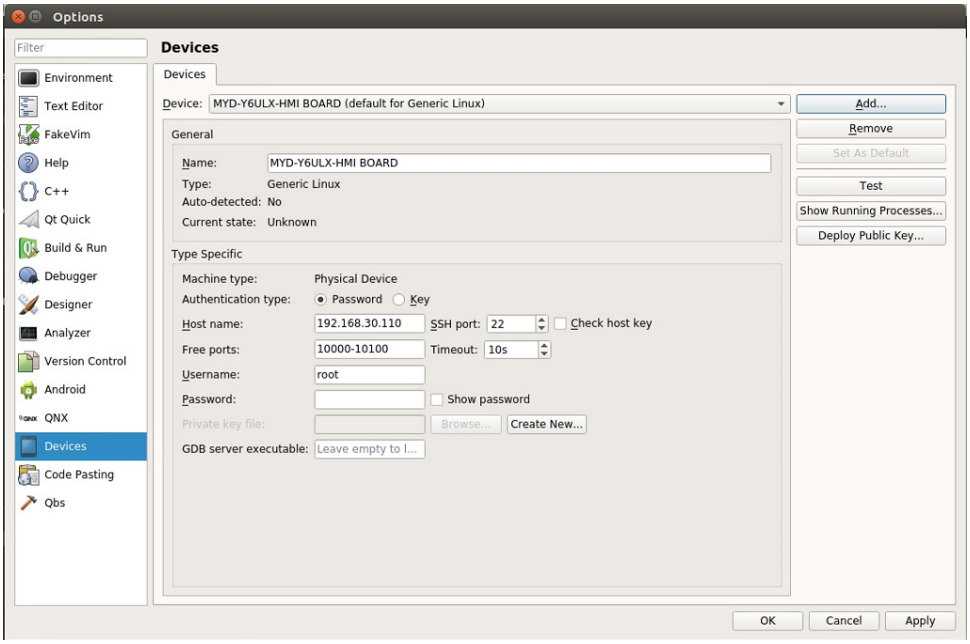

图5-2-3 配置设备

第四步，点击左侧"Build & Run"回到"Kits"标签下，"Name"为"MYD-Y6ULX-HMI-Dev-Kit"，"Device"选择"MYD-Y6ULX-HMI BOARD"选项了。"Sysroot"选择目标设备的系统目录，这里以"/opt/qt5-hmi/sysroots/cortexa7hf-neon-poky-linux-gnueabi"为例。"Compiler"选择之前配置的名称"MYD-Y6ULX-HMI-GCC"，"Qt version"选择之前配置的名称"Qt 5.6.2 (MYD-Y6ULX-HMI-QT5)"，"Qt mkspec"填写为"linux-oe-g++"。其它默认即可，最后点击"Apply"和"OK"按钮。

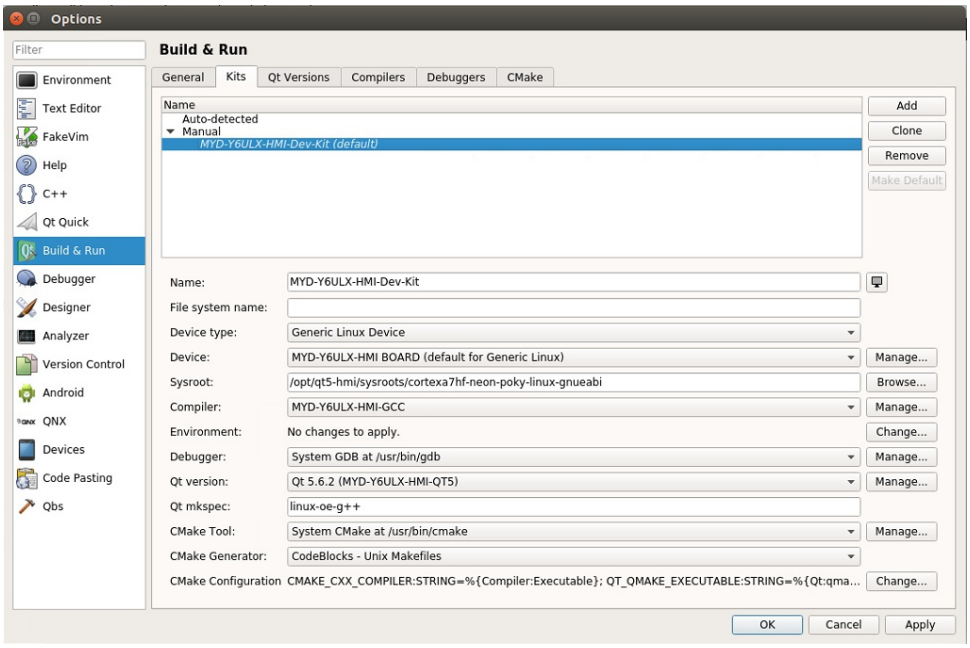

图5-2-4 配置套件

5.3 测试Qt应用

为了方便测试之前的配置是否正确，这里提供了一个Qt例程，打开项目后，配置为相应的编译工具套件，就可以编译此例程。

第一步，在菜单栏选择"File"->"Open File or Project"，在打开的对话框中，浏览到"helloword"例程的目录下，选择"hello_demo_hmi.pro"文件，点击"Open"按钮。

第二步，项目打开后，选择"MYD-Y6ULX-HMI-Dev-Kit"选项，这样"hello_demo_hmi"项目就会使用"MYD-Y6ULX-HMI-Dev-kit"的相关配置构建应用。

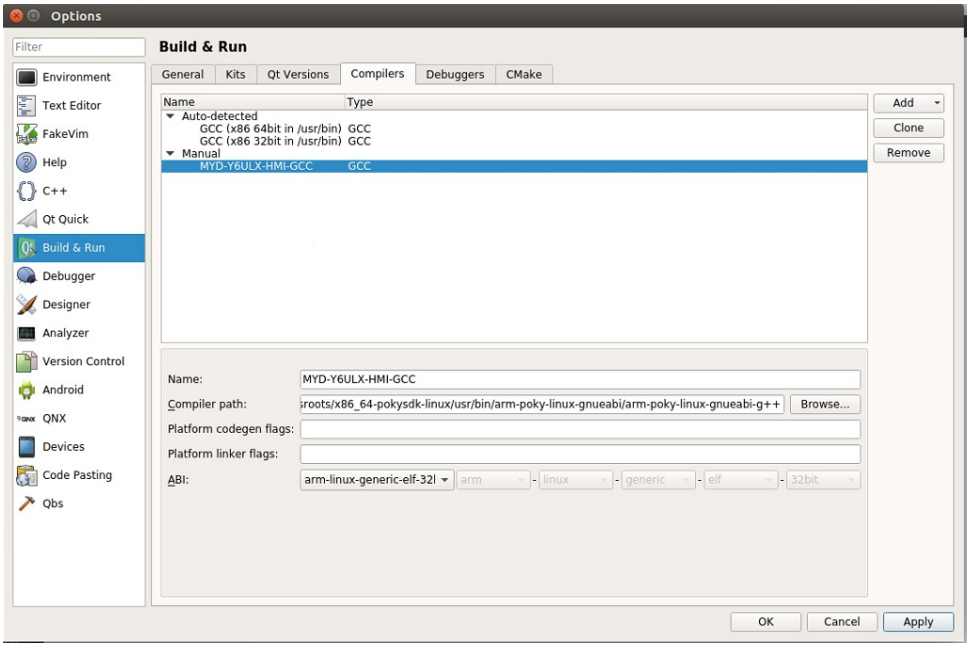

图5-3-1 选择构建配置

第三步，点击菜单栏"Build"->"Build Project hello_demo_hmi"按钮，即可完成项目的编译，同时下侧会有编译过程输出。

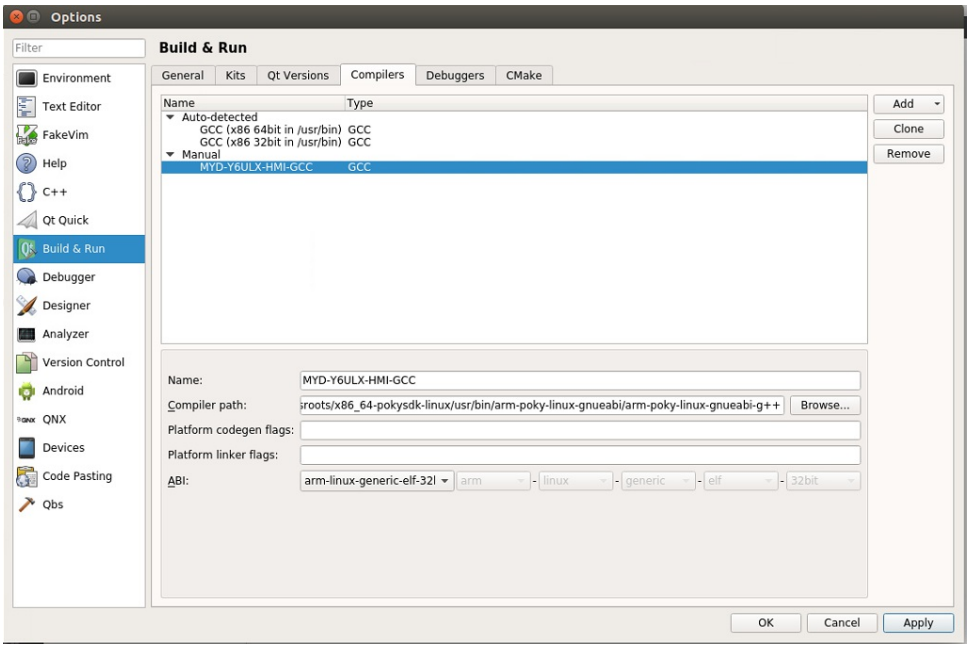

图5-3-2 编译输出结果

QtCreator 构建 hello_demo_hmi 项目后，编译好的二进制文件存放在"~/build-hello_demo_hmi-MYD_Y6ULX_HMI_Dev_Kit-Debug/"目录下，可以使用 file 命令查看，是否编译为 ARM 架构。

```
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
```

然后将 hello_demo_hmi 文件拷贝到开发板下运行即可。

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

将会在 LCD 屏幕上看到Qt 窗口中一个QLabel的文本内容。

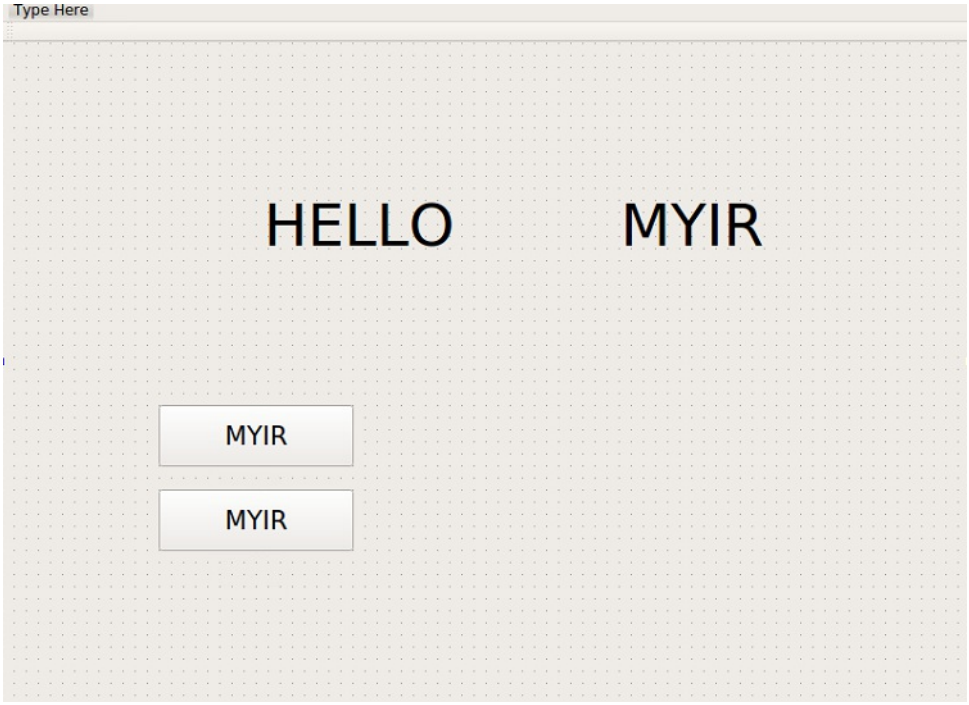

图5-3-3 例程运行结果

6 系统更新

MYD-Y6ULX-HMI系列板子提供了一种更新Linux系统固件的方法，SD卡更新。

- SD卡更新: 使用可启动的SD更新卡，启动后从SD卡烧写文件到板载Flash。

6.1 SD卡更新

MYD-Y6ULX-HMI开发板提供了一个使用SD卡更新系统镜像的文件，sdcard镜像文件。sdcard镜像文件需要使用特殊的磁盘操作工具才可以写入Micro SD卡内，Linux系统用户可以直接使用dd命令，Windows系统用户使用Win32ImageWriter工具。

MYD-Y6ULX-HMI板子的开发资源包内已包含多个sdcard.img.gz文件，位置为02-Images。

文件	核心板	底板	扩展板	文件系统
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

注意：rootfs.sdcard前面的时间为生成文件时的日期时间，请以实际文件为主。

制做SD更新的镜像

如果对Linux kernel，U-Boot或者文件系统有修改，可以自行打包制做sdcard文件，将系统烧写在开发板上。MYD-Y6ULX-HMI开发板提供了一个可以制做SD更新镜像的工具MYiR-iMX-mkupdate-sdcard-HMI，存放在04-Tools/ManufactoryTool目录。

build-sdcard.sh脚本用于制做从SD卡更新系统的镜像，分为两个部分：更新系统和目标文件。 firmware目录下是更新系统，一般情况下不需要修改。 'mfgimages-*'是目标文件，里面的文件最终会烧写进板载的NAND或者eMMC存储芯片内。如果修改u-boot, kernel后，需要把相应的文件替换到目标文件内即可。

'mfgimage-*'目录内的文件名遵循以下方式命名，错误的文件名称在更新系统时不会被识别，会出现升级失败。 这些文件的名称被定义在Manifest文件内，命名规则如下：

```
ubootfile="u-boot.imx"
kernelfile="zImage"
dtbfile="myd-y6ull-gpmi-weim.dtb"
rootfsfile="core-image-base.rootfs.tar.xz"
```

更新程序启动后会根据Manifest文件加载需要的文件，以将它们写入到目标NAND Flash存储芯片。

解压后就可以开始制做镜像了，参考命令如下：

```
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
```

build-sdcard.sh提供了7种参数：

- -p 表示平台，可用参数为"myd-y6ull"代表imx6ull核心板, "myd-y6ull-hmi"代表MYD-Y6ULX-HMI板。
- -n 表示使用NAND Flash。'-e' 表示使用eMMC Flash。这两个不能同时使用。
- -d 表示指定更新目录的位置。
- -t 给示添加一个tag标识到生成的镜像文件名中。
- -s 表示指定内存大小,256或者512
- -f 表示指定文件系统类型,可选值有"core","qt","mini"
- -x 表示有扩张板，没有用则不加此参数

注意：'-n'和'-e'不能同时使用，只能使用一种。

运行结束后会生成一个sdcard.img.gz后缀的文件，如 :
'myd-y6ull-hmi-update-nand-ddr256m-core-20181114191138-exp.rootfs.sdcard.img.gz'。

制做可更新系统的SD卡

MYD-Y6ULX-HMI资源包内提供了用于更新系统的sdcard.gz镜像文件，可以直接使用，也可以使用上一步制做的sdcard.gz文件。MYD-Y6ULX提供好的sdcard.gz文件存放在02-Images目录内。 有了用于更新的SD卡镜像文件，就可以把镜像文件写入到SD卡。为了方便使用，建议把Micro SD插入USB读卡器，再插入电脑USB端口。

注意: 02-Images目录内的文件名的时间标识部分可能与如下示例文件有差异，请以实际为主。

- Linux系统

通常Linux下的存储设备名为"sd[x][n]"形式，x表示第几个存储设备，一般使用字母a~z表示。n表示存储设备的分区，一般使用数字，从1开始。插入后可以使用"dmesg | tail"命令查看新设备的设备名称。这里以"/dev/sdb"设备为例，sdb后面不写任务分区数字。

sdcard文件烧写：

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

sdcard.gz文件烧写：

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

写入的速度与USB和Micro SD卡的速率有关，如果对速度有要求，建议选用更高速度等级的Micro SD存储卡。

- Windows系统

Windows用户可以使用Win32DiskImager工具把sdcard镜像写入Micro SD里。工具在"03-Tools"目录下，解压后，双击"Win32DskImager.exe"应用程序。启动后的界面中，右侧的"Device"是选择要写入的设备盘符。左侧的"Image File"是选择将要写的镜像文件，点击旁边的文件夹图标，选中要写入的文件即可(注意：文件选择对话框中默认是过滤"*.img*"文件，切换成"*.*"，就可以显示到sdcard后缀的文件)。

写入前请再次确认目标磁盘和文件是否正确，避免写入到系统磁盘，损坏Windows系统分区。

注意：由于Win32DiskImage只支持sdcard文件，所以需要先解压sdcard.img.gz文件后才可以使用。

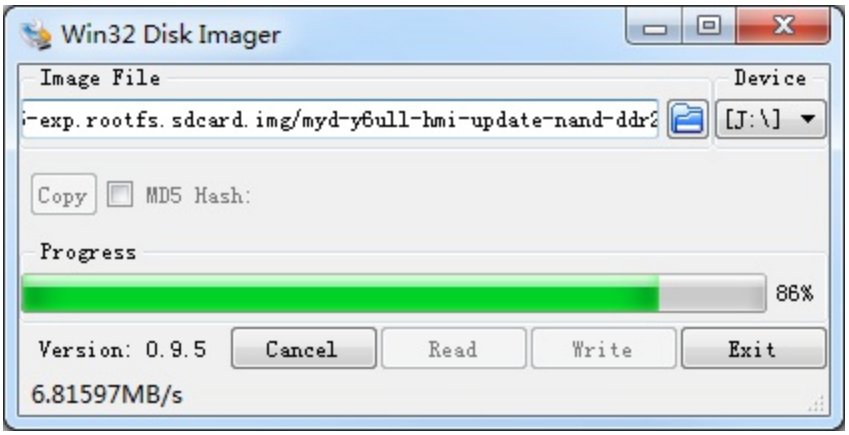

图6-1 Win32DiskImage写入镜像

等待进度条结束后，就可以拔出USB读卡器。

把制做好的Micro SD卡插入开发板的卡槽(J5)，配置启动位拨码开关(SW1)为SDCARD启动方式。

MYD-Y6ULX-HMI的核心板有两种存储方式，NAND和eMMC，启动时对应的拨码开关配置也不同，详情如下。

从SD卡启动的拨码开关配置：

- **MYD-Y6ULX-HMI NAND**存储版本

启动位	状态
Bit1	ON
Bit2	OFF
Bit3	ON
Bit4	OFF

- **MYD-Y6ULX-HMI eMMC**存储版本

启动位	状态
Bit1	ON
Bit2	ON
Bit3	ON
Bit4	OFF

连接USB转TTL串口线至调试串口(JP1)，配置好电脑端的串口终端软件。使用DC 12V电源适配器连接至开发板的电源接口(J1)。通过串口可以看到系统从Micro SD卡启动，并执行更新脚本，把Linux系统镜像文件写入NAND存储芯片内

更新完成后断电，配置启动位拨码开关为板载的NAND或eMMC启动方式，配置详情如下。

从板载flash启动的拨码开关配置：

- **MYD-Y6ULX-HMI NAND**版本

启动位	状态
Bit1	OFF

Bit2	ON
Bit3	ON
Bit4	OFF

重新连接电源，板子、就可以从NAND启动系统了。

- **MYD-Y6ULX-HMI eMMC**版本

启动位	状态
Bit1	OFF
Bit2	OFF
Bit3	ON
Bit4	OFF

重新连接电源，板子就可以从eMMC启动系统了。

附录一 联系方式

销售联系方式

- 网址： www.myir-tech.com
- 邮箱：sales.cn@myirtech.com

深圳总部

- 负责区域：广东 / 四川 / 重庆 / 湖南 / 广西 / 云南 / 贵州 / 海南 / 香港 / 澳门
- 电话：0755-25622735 0755-22929657
- 传真：0755-25532724
- 邮编：518020
- 地址：深圳市龙岗区坂田街道发达路云里智能园2栋6楼04室

上海办事处

- 负责区域：上海 / 湖北 / 江苏 / 浙江 / 安徽 / 福建 / 江西
- 电话：021-60317628 15901764611
- 传真：021-60317630
- 邮编：200062
- 地址：上海市普陀区中江路106号北岸长风I座1402

北京办事处

- 负责区域：北京 / 天津 / 陕西 / 辽宁 / 山东 / 河南 / 河北 / 黑龙江 / 吉林 / 山西 / 甘肃 / 内蒙古 / 宁夏
- 电话：010-84675491 13269791724
- 传真：010-84675491
- 邮编：102218
- 地址：北京市昌平区东小口镇中滩村润枫欣尚2号楼1009

技术支持联系方式

- 电话：0755-25622735
- 邮箱：support@myirtech.com

如果您通过邮件获取帮助时，请使用以下格式书写邮件标题：

```
[公司名称/个人--开发板型号]问题概述
```

这样可以使我们更快速跟进您的问题，以便相应开发组可以处理您的问题。

附录二 售后服务与技术支持

凡是通过米尔科技直接购买或经米尔科技授权的正规代理商处购买的米尔科技全系列产品，均可享受以下权益：

1、6个月免费保修服务周期

2、终身免费技术支持服务

3、终身维修服务

4、免费享有所购买产品配套的软件升级服务

5、免费享有所购买产品配套的软件源代码，以及米尔科技开发的部分软件源代码

6、可直接从米尔科技购买主要芯片样品，简单、方便、快速；免去从代理商处购买时，漫长的等待周期

7、自购买之日起，即成为米尔科技永久客户，享有再次购买米尔科技任何一款软硬件产品的优惠政策

8、OEM/ODM服务

如有以下情况之一，则不享有免费保修服务：

1、超过免费保修服务周期

2、无产品序列号或无产品有效购买单据

3、进液、受潮、发霉或腐蚀

4、受撞击、挤压、摔落、刮伤等非产品本身质量问题引起的故障和损坏

5、擅自改造硬件、错误上电、错误操作造成的故障和损坏

6、由不可抗拒自然因素引起的故障和损坏

产品返修：用户在使用过程中由于产品故障、损坏或其他异常现象，在寄回维修之前，请先致电米尔科技客服部，与工程师进行沟通以确认问题，避免故障判断错误造成不必要的运费损失及周期的耽误。

维修周期：收到返修产品后，我们将即日安排工程师进行检测，我们将在最短的时间内维修或更换并寄回。一般的故障维修周期为3个工作日（自我司收到物品之日起，不计运输过程时间），由于特殊故障导致无法短期内维修的产品，我们会与用户另行沟通并确认维修周期。

维修费用：在免费保修期内的产品，由于产品质量问题引起的故障，不收任何维修费用；不属于免费保修范围内的故障或损坏，在检测确认问题后，我们将与客户沟通并确认维修费用，我们仅收取元器件材料费，不收取维修服务费；超过保修期限的产品，根据实际损坏的程度来确定收取的元器件材料费和维修服务费。

运输费用：产品正常保修时，用户寄回的运费由用户承担，维修后寄回给用户的费用由我司承担。非正常保修产品来回运费均由用户承担。