

MEasy HMI开发手册



LED



RS232



RS485



CAN



EtherNet



Contact Us

Contact Us

E-mail: sales@myirtech.com or myirtech@yahoo.com

Phone: +86-755-22984836

Fax: +86-755-25532724

Address : Room 04, 6th Floor, Building No.2, Fada Road, Yunli Smart Park, Bantian, Longgang District, Shenzhen, Guangdong, China 518129



发光二极管



串口



RS485



CAN总线



以太网



任务管理器



米尔支持



系统信息

目录

前言	0
1. MEasy HMI框架介绍	1
2. 本地HMI使用介绍	2
2.1 串口	2.1
2.2 RS485	2.2
2.3 以太网	2.3
2.4 任务管理器	2.4
2.5 米尔支持	2.5
2.6 系统信息	2.6
2.7 数据库	2.7
2.8 摄像头	2.8
3. 本地HMI应用开发	3
3.1 环境搭建	3.1
3.2 编译本地HMI应用	3.2
3.3 运行本地HMI应用	3.3
3.4 添加应用到本地HMI	3.4
4. Web HMI使用介绍	4
4.1 串口	4.1
4.2 RS485	4.2
4.3 以太网	4.3
4.4 米尔支持	4.4
5. Web HMI应用开发	5
5.1 i.MX6UL系列开发板上添加运行时库	5.1
6. MEasy HMI应用集成	6
6.1 i.MX6UL系列开发板上集成MEasy HMI应用	6.1
7. DBUS API介绍	7
7.1 LED	7.1
7.2 串口	7.2
7.3 RS485	7.3

7.4 CAN	7.4
7.5 Connman	7.5
附录A	8
附录B	00

MEasy HMI V1.0开发手册

前言

本文档主要讲述MEasy HMI的基本框架，并演示MEasy HMI在深圳市米尔电子有限公司(下文简称“米尔”)开发板上的运行，此外还进一步说明了MEasy HMI开发环境的搭建和源码的编译以及应用集成，以实例的形式讲述了如何在MEasy HMI框架的基础上开发更多的应用。

本文档适合有一定开发经验的嵌入式linux开发工程师，QT开发工程师和Web前端和后端开发工程师。

版本历史:

版本号	描述	时间
V1.0	初始版本	2018.5.1
V1.1	针对MYD-Y6ULX-HMI板子适配相关的功能	2018.12.20

硬件版本:

文档v1.0适用于米尔AM437X, AM335X, i.MX6UL系列开发板，具体信息以相应产品的发布包为准。

文档v1.1适用于米尔MYD-Y6ULX-HMI系列开发板，具体信息以相应产品的发布包为准。

注意: 开发板Linux系统默认的root账户密码为空。

1. MEasy HMI框架介绍

MEasy HMI是深圳市米尔科技有限公司开发的一套人机界面框架，它包含基于QT5的本地HMI和远程的Web HMI。本地HMI需要硬件平台具备显示单元、输入单元、通讯接口、数据存贮单元等；软件部分需要包含dbus、connman和QT5运行时环境等。Web HMI是B/S架构的应用，需要网络接口支持，软件部分包含Python2.x以及tornado, javascript, css, HTML, websocket等运行环境。本地HMI和Web HMI的结构框图如下所示：

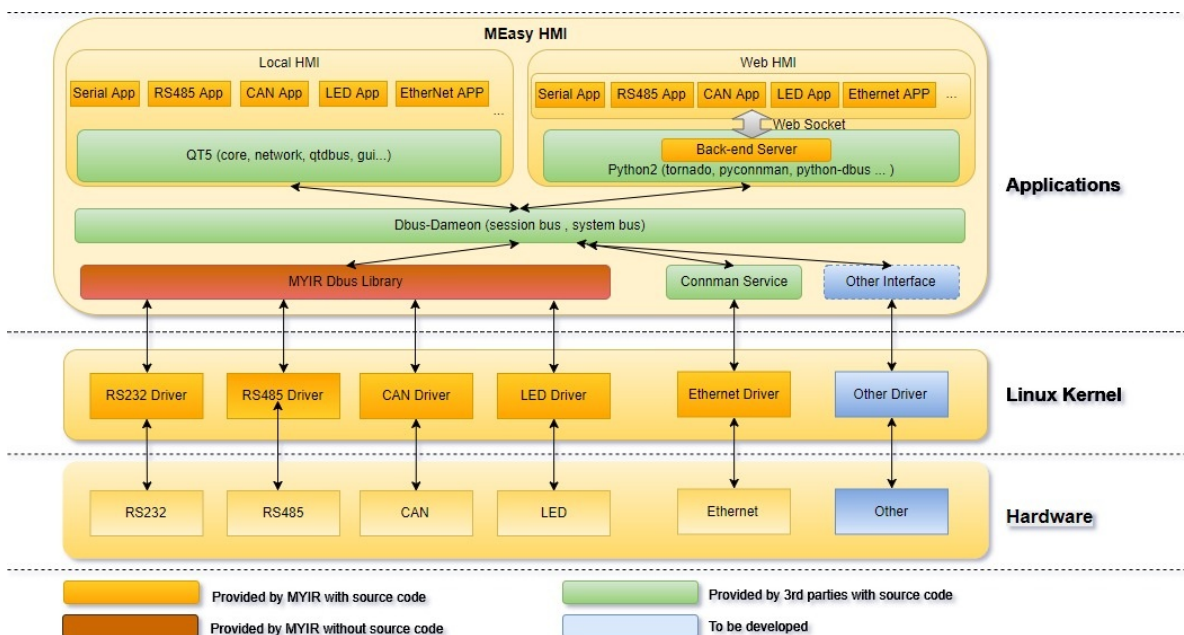


图1-1 MEasy HMI结构框图

MEasy HMI使用D-Bus作为应用程序和底层硬件的访问接口。RS232、RS485、CAN、LED这些硬件使用米尔提供的一套完整的控制和通信接口，对外提供基于D-BUS的Method和Signal，在最后一章会介绍这些Method和Signal。用户可以根据需要对我们提供的接口进行扩展以实现更强大的功能，关于D-bus的更多细节请参考<http://dbus.freedesktop.org>。

MEasy HMI中的网络管理应用则使用开源的Connman作为中间层来实现对网络设备的控制，Connman也是一个基于D-Bus的完全模块化的系统，可以通过插件化进行扩展，以支持EtherNet、WIFI、3G/4G、Bluetooth等网络设备的管理。关于Connman的更多细节请参考这<https://01.org/zh/node/2207>

MEasy HMI在目标板上目录结构如下，在后面的章节中将会详细介绍MEasy HMI应用如何集成到目标板系统中。

```
/
├─ home
│   └─ myir
│       ├── mxapp
│       ├── mxbackend
│       ├── mxcan
│       ├── mxinfo
│       ├── mxled
│       ├── mxnet
│       ├── mxrs485
│       ├── mxserial
│       ├── mxsupport
│       └─ mxtaskmanager
└─ usr
    ├── bin
    │   ├── psplash
    │   └─ psplash-write
    ├── lib
    │   ├── fonts
    │   │   └─ msyh.ttc
    │   ├── girepository-1.0
    │   ├── gobject-introspection
    │   ├── libgirepository-1.0.la
    │   ├── libgirepository-1.0.so
    │   ├── libgirepository-1.0.so.1
    │   ├── libgirepository-1.0.so.1.0.0
    │   └─ python2.7
    └─ share
        ├── applications
        ├── myir
        │   ├── mxde.xml
        │   ├── settings.ini
        │   ├── board_cfg.json
        │   └─ www
        │       ├── application.py
        │       ├── handler
        │       ├── README.md
        │       ├── server.py
        │       ├── statics
        │       └─ template
        └─ pixmaps
```

2. 本地HMI使用介绍

本节主要介绍MEasy 本地HMI中每个APP的使用及使用过程中注意的细节。

软件环境：

- u-boot
- linux-4.1.x
- 带QT5运行环境的文件系统
- MEasy 本地HMI V1.1应用程序

以上软件已经在出厂的时候烧写到对应的开发板里面了。

硬件环境：

- MY-TFT070CV2电容屏
- MYD-Y6ULX-HMI

默认出厂程序只支持MY-TFT070CV2电容屏。

硬件连接方式：

表2-1 开发板显示屏接口

开发板	LCD接口
MYD-Y6ULX-HMI	J9 LCD_16bit

2.1 串口

本例程演示如何使用MEasy HMI中的串口应用来配置开发板的串口设备及串口收发数据测试，详情请参考源码mxserial。

软件环境:

- 串口应用程序

硬件环境：

- 带UART接口的开发板一块
- 装有串口助手软件的PC

表2-1-1 开发板串口列表

开发板	接口	数据线
MYD-Y6ULX-HMI	J8的PIN5、PIN6	杜邦线

界面介绍：



图2-1-1 本地串口测试界面

注意：点击发送组框中的编辑框会弹出软键盘，输入完数据后需要点击软键盘中蓝色Close按钮才能关闭软键盘，然后点击发送按钮才能将数据发送出去，发送完毕后编辑框中的数据自动被清空。

测试步骤：

- 用连接线连接PC端USB和开发板串口。

- 打开PC端串口助手软件，设置串口参数并打开串口。
- 打开MEasy HMI中串口应用，设置和PC端同样的串口参数并打开串口。
- 分别在PC端和开发板端发送数据，然后看两端能否收到数据。

2.2 RS485

本例程演示如何使用MEasy HMI中的RS485应用来配置开发板的RS485设备及RS485收发数据测试，详情请参考源码mxrs485。

软件环境:

- RS485应用程序

硬件环境：

- 带RS485接口的开发板两块
- 数据线连接两块板的RS485接口，485A<->485A，485B<->485B，GND<->GND

表2-2-1 开发板RS485接口列表

开发板	接口	数据线
MYD-Y6ULX	J8的PIN3、PIN4	杜邦线

界面介绍：



图2-2-1 开发板RS485配置

注意：点击发送组框中的编辑框会弹出软键盘，输入完数据后需要点击软键盘中蓝色Close按钮才能关闭软键盘，然后点击发送按钮才能将数据发送出去，发送完毕后编辑框中的数据自动被清空。

测试步骤：

- 使用杜邦线连接两块开发板的RS485接口

- 分别启动各自开发板中的MEasy HMI中的RS485应用程序
- 配置各开发板RS485设置组框的参数，端口可能会不一样，要保证两块开发板的波特率、校验位、数据位和停止位一致。
- 配置完成后点击打开按钮，然后在两块开发板进行收发数据测试。

2.3 以太网

本例程演示如何使用MEasy HMI中的以太网应用来配置开发板的网口及测试网口连通性，详情请参考源码mxnet。

软件环境:

- 以太网应用程序

硬件环境：

- 可提供DHCP服务的路由器一个
- MYD-Y6ULX-HMI的开发板一块
- MYB-Y6ULX-HMI的开发板一块

表2-3-1 开发板以太网口列表

开发板	接口
MYD-Y6ULX-HMI	CN1
MYB-Y6ULX-HMI-4GEXP	CN1

界面介绍：



图2-3-1 开发板网口配置



图2-3-2 开发板网口测试

标签页：网卡对应的功能页面

IP信息页面：包含设置组框和信息组框

Ping测试：测试网络连通性的页面

注意：

1. 标签页是动态创建的，在没插网线的情况下看不到任何界面的，插入几个网线到网口，就会创建几个标签页，同理拔掉网线会删除对应的标签页。
2. IP获取方式切换到Manual模式下，会弹出IP地址、子网掩码、网关的输入框，可用于手动配置IP。
3. 在IP获取方式为Manual模式下，点击IP地址、子网掩码、网关的输入框编辑框会弹出软键盘，输入完数据后需要点击软键盘中蓝色Close按钮才能关闭软键盘，然后点击确定按钮才能将IP地址、子网掩码、网关这些信息配置下去，配置完毕后编辑框中的数据自动被清空。

测试步骤：

1. 将网线插入开发板的网口。
2. 打开MEasy HMI中的以太网测试应用程序，查看信息组框是否成功获取到IP信息。
3. 切换到Ping测试页面测试网络联通性。

2.4 任务管理器

本例程演示如何使用MEasy HMI中的任务管理器应用程序查看系统资源状态和进程信息，详情请参考源码mxtaskmanager。

软件环境:

- 任务管理器应用程序

硬件环境:

- MYD-Y6ULX-HMI的开发板一块

界面介绍:



图2-4-1 开发板性能信息

注意：存储空间只是展示了root分区大小，并不是代表整个存储设备的空间。

测试步骤:

打开MEasy HMI中任务管理器应用程序即可查看相关的性能信息和进程信息。

2.5 米尔支持

本例程演示如何使用MEasy HMI中的米尔支持应用获得与我司的联系方式，详情请参考源码mxsupport。

软件环境:

- 米尔支持应用程序

硬件环境：

- MYD-Y6ULX-HMI的开发板一块

界面介绍：



图2-5-1 开发板技术支持信息

使用方法：

打开MEasy HMI中米尔支持应用程序即可。

2.6 系统信息

本例程演示如何使用MEasy HMI中的系统信息应用来查看开发板的软硬件信息，详情请参考源码mxinfo。

软件环境:

- 系统信息应用程序

硬件环境：

- MYD-Y6ULX-HMI的开发板一块

界面介绍：



图2-6-1 开发板系统信息

使用方法：

打开MEasy HMI中的系统信息应用程序即可。

2.7 数据库

本例程演示如何使用MEasy HMI中的SQLite应用读写数据库，以系统时间为数据源，详情请参考源码sqlite。

软件环境:

- 数据库应用程序

硬件环境：

- MYD-Y6ULX-HMI的开发板一块

界面介绍：

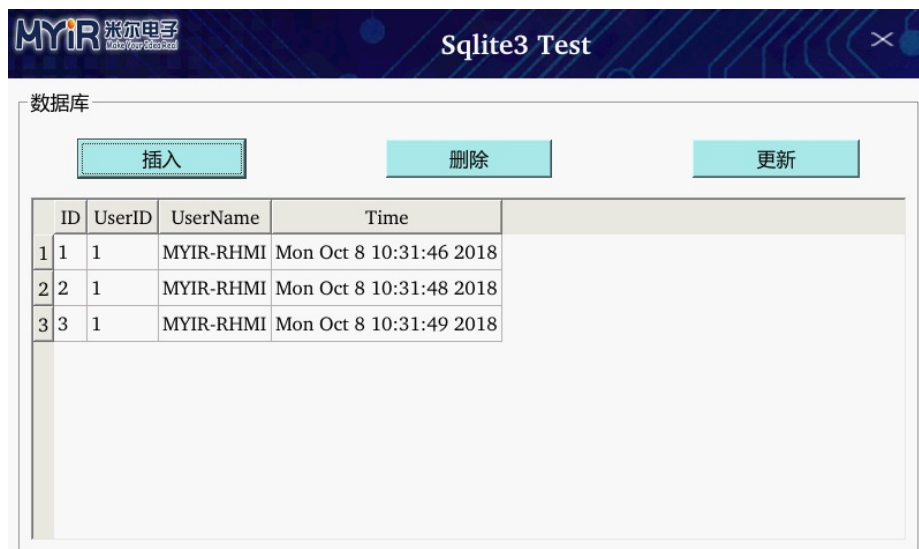


图2-7-1 开发板Sqlite3

使用方法：

打开MEasy HMI中数据库应用.

- 点击插入按钮会向数据库中写入当前时间和用户名等参数,然后再从数据中读取出来数据显示到界面。
- 选中要删除的一条数据,点击删除按钮,会删除数据库中的此条数据,然后反馈到界面

2.8 摄像头

本例程演示如何使用MEasy HMI中的摄像头应用实现预览拍照和保存等功能,详情请参考源码mxcamera。建议使用USB摄像头。

软件环境:

- 摄像头应用程序

硬件环境:

- MYD-Y6ULX-HMI的开发板一块

界面介绍:



图2-8-1 Camera界面

使用方法:

打开MEasy HMI中摄像头应用

- 设置端口,选中摄像头对应的设备,如/dev/video2
- 设置摄像头的分辨率
- 点击预览,按钮下方显示预览的画面
- 点击拍照,按钮下方显示当前的一帧画面
- 点击保存,将当前帧画面保存为jpg格式的文件,保存在/home/root目录下

3. 本地HMI应用开发

本章重点介绍如何构建MEasy本地HMI的开发编译环境，旨在帮助用户更好更快捷的通过QT5来开发自己的产品。其中包括嵌入式QT5运行环境搭建，qmake编译环境的搭建，QT Creator安装和配置及MEasy本地HMI应用的编译与运行。

3.1 环境搭建

本章讲述的MEasy本地HMI的环境包含开发板上的QT5运行环境和ubuntu主机端的的qmake及交叉编译工具链。i.MX6UL系列的QT5运行环境使用yocto编译生成，具体操作参见对应产品的软件开发手册。

表3-1-1 i.MX6UL系列开发板编译QT

开发板	文档章节
MYD-Y6ULX-HMI	MYD-Y6ULX-HMI Linux开发手册.pdf 3.3编译QT
MYD-Y6ULX	MYD-Y6ULX-LinuxDevelopmentGuide_zh 3.3构建文件系统-构建GUI Qt5版的系统
MYS-6ULX	MYS-6ULX-LinuxDevelopmentGuide_zh.pdf 3.3构建文件系统-构建GUI Qt5版的系统

QT5开发工具QT Creator的安装过程直接参考其文档的章节具体如下表：

表3-1-2 安装QT Creator

开发板	文档章节
MYD-Y6ULX-HMI	MYD-Y6ULX-HMI Linux开发手册.pdf 5.1编译QT
MYD-Y6ULX	MYD-Y6ULX-LinuxDevelopmentGuide_zh 5.1安装QT Creator
MYS-6ULX	MYS-6ULX-LinuxDevelopmentGuide_zh.pdf 5.1安装QT Creator

QT5开发工具QT Creator的配置过程直接参考其文档的章节具体如下表：

表3-1-3 i.MX6UL系列开发板配置QT Creator

开发板	文档章节
MYD-Y6ULX-HMI	MYD-Y6ULX-HMI Linux开发手册.pdf 5.2配置QT Creator
MYD-Y6ULX	MYD-Y6ULX-LinuxDevelopmentGuide_zh 5.2配置QT Creator
MYS-6ULX	MYS-6ULX-LinuxDevelopmentGuide_zh.pdf 5.2配置QT Creator

3.2 编译本地HMI应用

本章节主要讲述MEasy本地HMI的编译过程。

我们提供MEasy本地HMI源码位于光盘文件的_/04-Source/HMI-QT5-DEMO.tar.bz2目录，将mxde.tar.gz拷贝到ubuntu目录工作目录，并解压出来。

下面讲述如何将mxde这个工程导入到QT Creator中，打开QT Creator，在菜单栏中依次点击 File -> Open File or Project 然后弹出如图3-2-1 选择框，进入mxde工程目录，点击 mxde.pro 并点击 Open 按钮即可打开mxde这个工程。

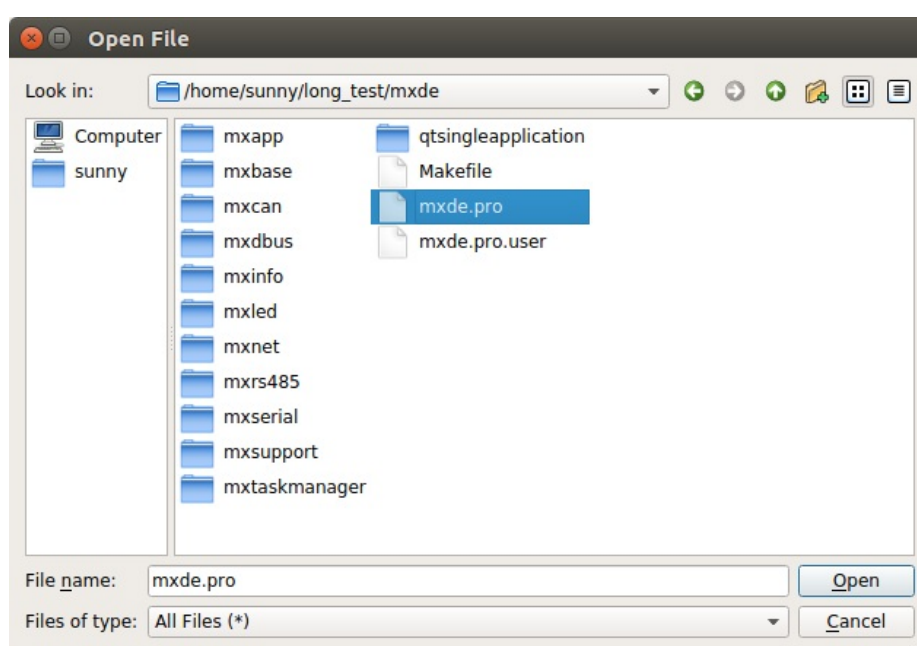


图3-2-1 工程选择框

打开工程后进入配置页面，选择编译工具链，这里直接选择3.1章配置好的kits，点击 Configure Project 按钮然后进入工程目录。

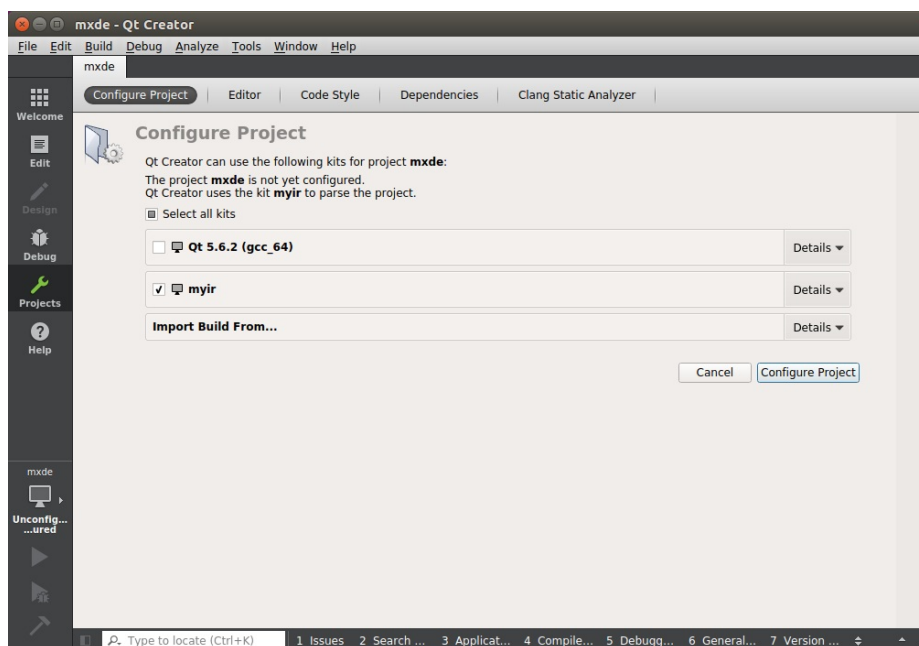


图3-2-2 配置kits

进入mxde工程后即可看到整个工程的目录结构，如图3-2-3所示。然后可以对工程进行编译了，编译之前可以选择下编译输出的模式，这里我们选择Release模式，然后可以选择右下角小锤子图标进行编译整个工程，或者通过点击菜单栏 `Build -> Build Project "mxde"` ,进行编译整个工程。

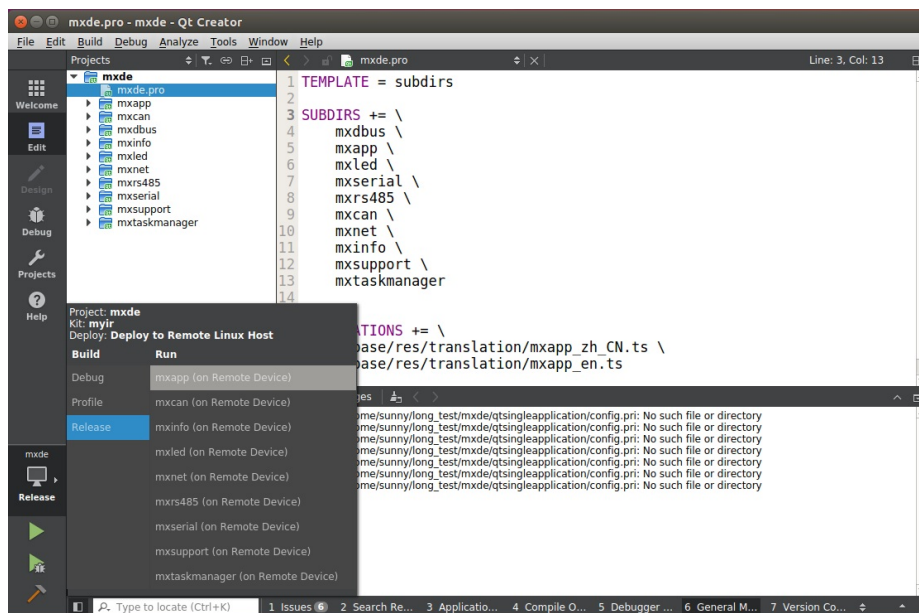


图3-2-3 工程目录

编译过程可以从底部Table栏 `4 Compile Output` 中看到，如图3-2-4所示。

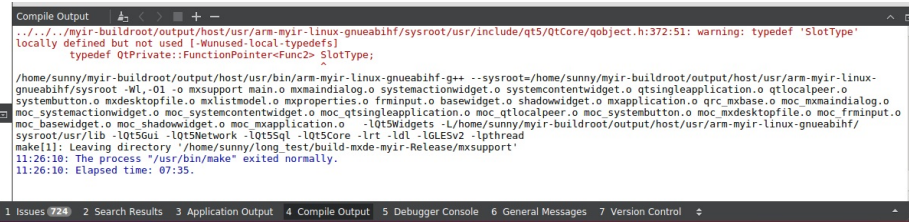


图3-2-4 编译输出

编译中出现的错误和警告信息可以从底部Table栏 1 Issues 中看到，如图3-2-5所示。如果编译出错可以从这里输出来进行分析问题。

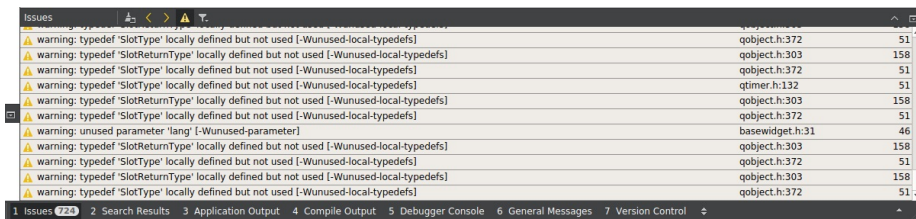


图3-2-5 问题输出

3.3 运行本地HMI应用

本章节主要讲述MEasy本地HMI的运行过程。编译完成后可以将编译完成的程序上传至开发板进行运行，这里提供两个方法将程序上传至开发板。

方法一：通过配置Qt Creator直接上传运行

- 配置Qt Creator的远程设备

通过选择菜单栏 `Tools -> Options -> Devices` 在Device中选择 `myir (default for Generic Linux)`，在Type Specific栏输入开发板IP（需要通过串口登录到开发板查看）、用户名，不需要填写密码。如图3-3-1所示。

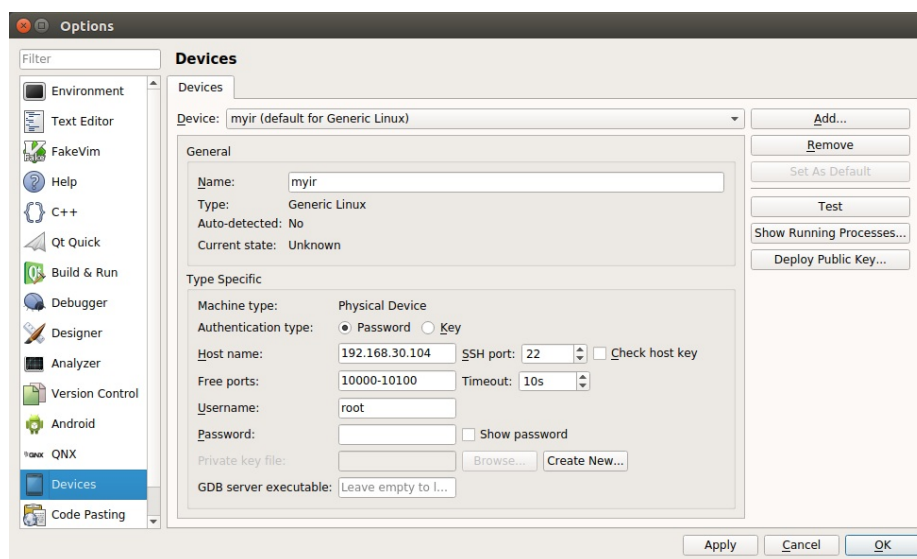


图3-3-1 设备配置

- 测试远端设备连通性

输入完毕后点击 `Apply` 按钮，然后点击右侧 `Test` 按钮会自动弹出测试连接的窗口当出现 `Device test finished successfully.`字样意味着测试连接成功。如图3-3-2所示。

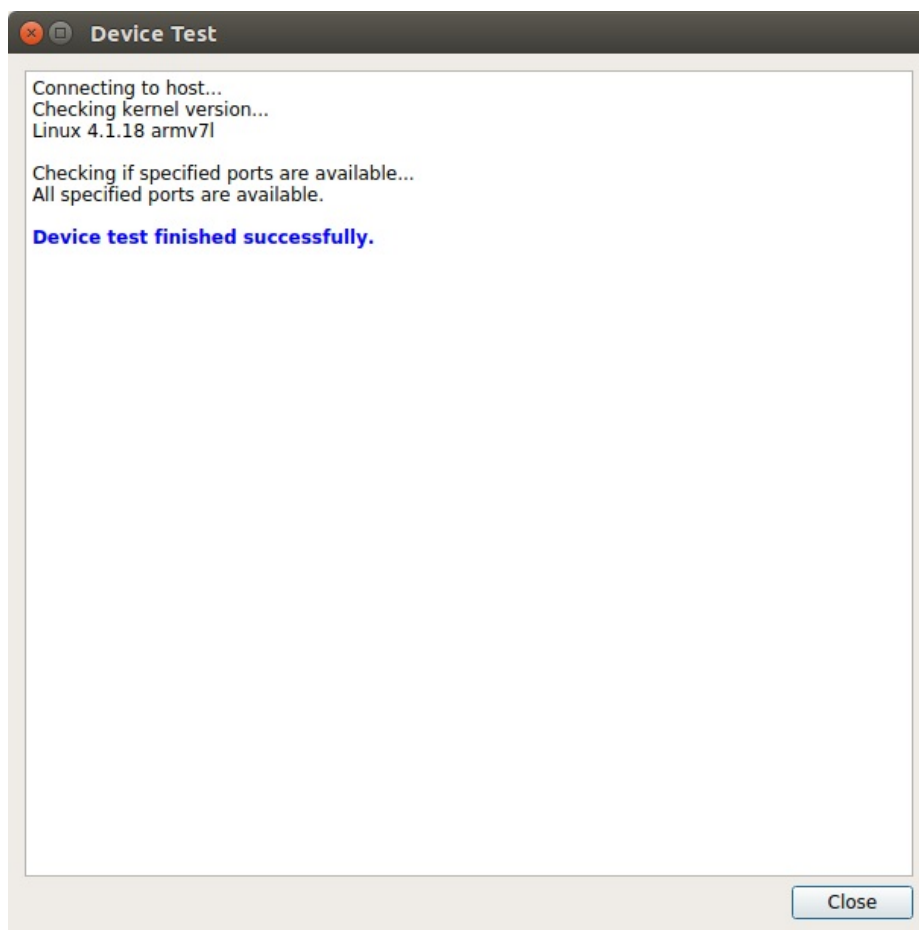


图3-3-2 设备测试

- 指定运行的程序

测试连接成功后，返回到Qt Creator的主界面，要指定你运行的程序，选择需要Run的程序为mxapp。如图3-3-3所示。

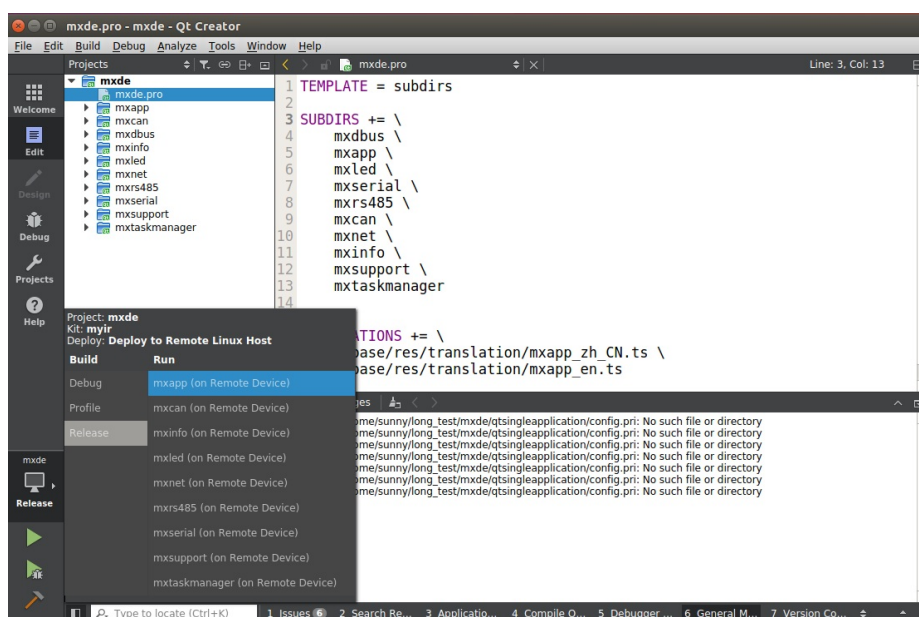


图3-3-3 选择需要运行的程序

- 指定程序运行的参数

指定完需要运行的程序以后，还需要指定程序的运行参数，依次点击左侧 `Projects` -> `mxde` -> `Build & Run` -> `myir` -> `Run` 下拉到Run配置栏，在Arguments输入框写入--`platform linuxfb`，即完成了程序运行参数的指定。如图3-3-4所示。

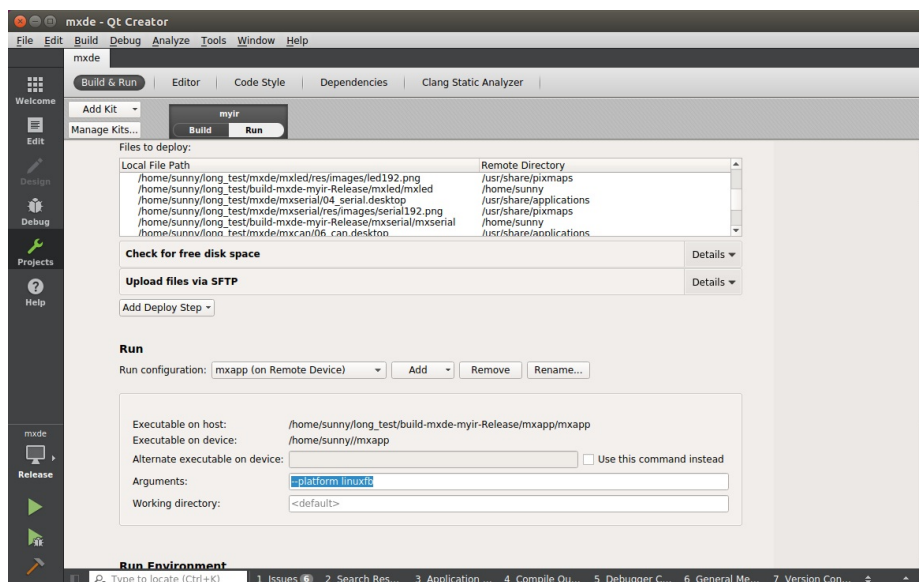


图3-3-4 程序运行参数

- 杀掉开发板上正在运行的MEasy HMI相关程序

指定完运行的参数以后，还需要登录到开发板，杀掉当前运行的MEasy HMI相关程序，操作如下：

```
# killall mxbackend
```

```
# killall mxapp
```

- 上传程序到开发板并运行

点击左下角的运行按钮，或者点击菜单栏 `Build` -> `Run` 就可以将mxapp上传至开发板并运行。7寸屏幕上可以看到MEasy HMI的界面，运行调试信息可以在 `3 Application output` 中看到，如图3-3-5所示。



图3-3-5 程序输出

注意：如果需要运行mxserial mxrs485 mxcan mxled这些应用程序，需要先运行mxbackend，还需要保证这些应用程序和mxbackend所连接dbus总线为同一个地址。操作方法如下：

1.设置串口终端当前运行的DBUS_SESSION_BUS_ADDRESS环境变量。

```
# dbus-launch
DBUS_SESSION_BUS_ADDRESS=unix:abstract=/tmp/dbus-qb40GAMAnL,guid=e3ab6092d0c14d9b138e64435ae0b6b0
DBUS_SESSION_BUS_PID=655
# export DBUS_SESSION_BUS_ADDRESS=unix:abstract=/tmp/dbus-qb40GAMAnL
```

2.运行后台程序。

```
# cd /home/myir/
# ./mxbackend
```

3.配置Qt Creator中运行程序的dbus会话总线DBUS_SESSION_BUS_ADDRESS环境变量。

以mxled为例说明Qt Creator中的配置。点击左侧 **Projects** -> **mxde** -> **Build & Run** -> **myir** -> **Run** -> **Run configuration** 这里选择mxled，在Arguments输入框写入--platform linuxfb，Run Environment中点击Details按钮，然后点击Add按钮，Variable栏填写DBUS_SESSION_BUS_ADDRESS，Value栏填写上面第一步dbus-launch创建的值unix:abstract=/tmp/dbus-qb40GAMAnL。如图3-3-6所示。

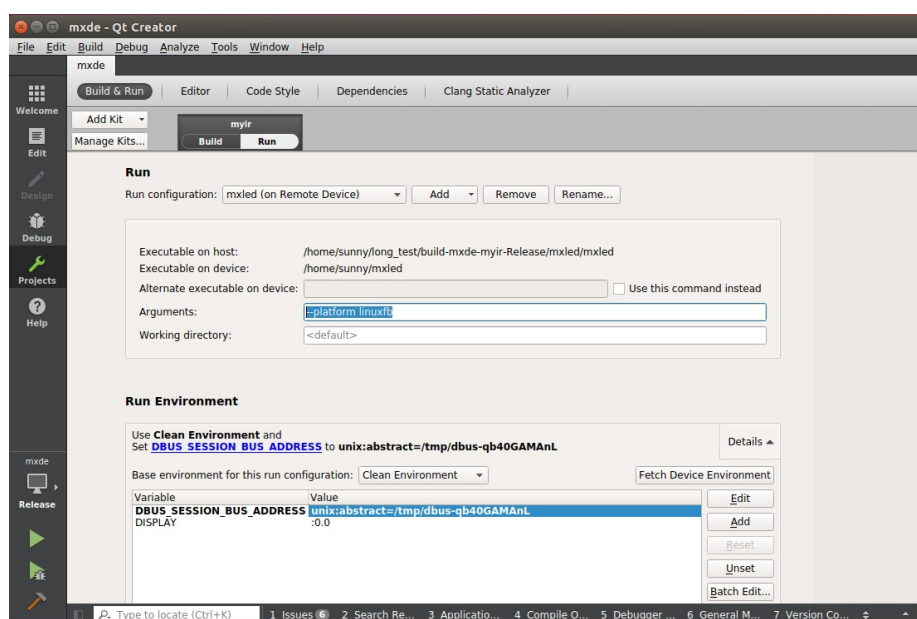


图3-3-6 应用程序dbus环境变量配置

4.点击左下角的运行按钮，或者点击菜单栏 **Build** -> **Run** 就可以将mxled上传至开发板并运行

方法二：直接拷贝编译好的程序到开发板

点击左侧 **Projects** 按钮,可以看到工程的编译配置，**General**栏中Build directory显示的就是mxde工程编译输出的路径，可以从这里直接把程序拷贝到开发板中。如图3-3-7所示。

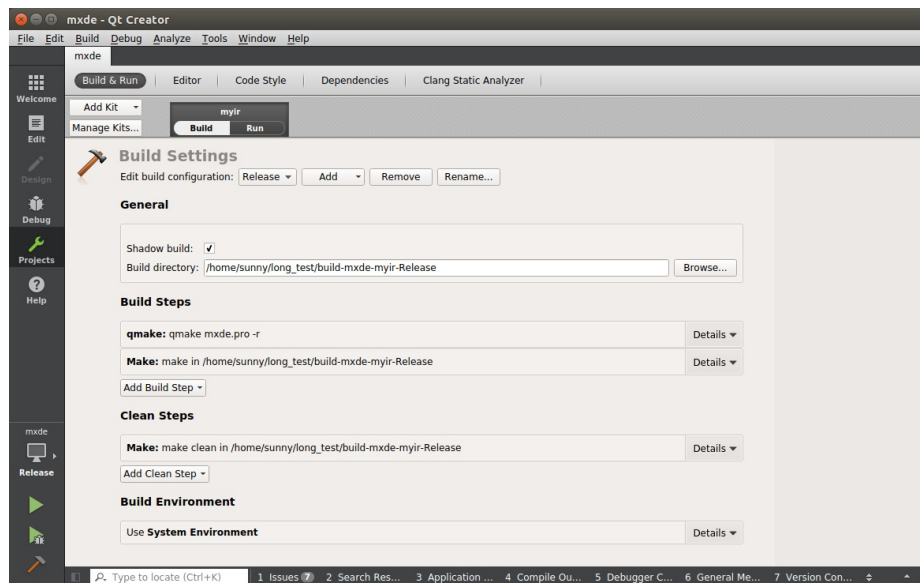


图3-3-7 工程编译输出

打开编译输出目录，进入mxapp目录，拷贝mxapp这个应用程序到开发板。运行方法如下：

```
# ./mxapp --platform linuxfb
=== w= 800 h=480
800 300 m_default_action_height
800 60 m_other_action_height
800 180 m_default_content_height
800 420 m_other_content_height
800 480
Could not parse application stylesheet
800 300 of HomeActionWidget
libpng warning: iCCP: known incorrect sRGB profile
800 180 of HomeContentWidget
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
QWidget::addChildLayout: layout "" already has a parent
QWidget::setLayout: Attempting to set QWidget "" on HomeContentWidget "", which already has a layout
QWidget::addChildLayout: layout "" already has a parent
QWidget::setLayout: Attempting to set QWidget "" on HomeContentWidget "", which already has a layout
QWidget::addChildLayout: layout "" already has a parent
QWidget::setLayout: Attempting to set QWidget "" on HomeContentWidget "", which already has a layout
```

```
QWidget::setLayout: Attempting to set QLayout "" on HomeContentWidget "", which al
ready has a layout
800 60 of BOXA
libpng warning: iCCP: known incorrect sRGB profile
800 420 of BoxContentWidget
loadApplicationWidgets
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
```

3.4 添加应用到本地HMI

本章节主要讲述向MEasy本地HMI中添加的用户的应用。用户如果需要在MEasy本地HMI中显示用户的应用，只需要一下几步就可以完成操作。

- 创建用户的Qt Widgets Application类型的应用命名为user_app，使用上述编译环境编译完成后拷贝至开发板/home/myir目录下
- 为这个应用创建一个192*192分辨率的图标user_app.png，拷贝至开发板/usr/share/pixmaps目录下
- 在开发板/usr/share/applications目录下创建一个属于用户的桌面配置文件，命名以数字开头例如09_user_app.desktop，配置文件里面内容如下：

```
[Desktop Entry]
Name=user_app
Name[zh_TW]=用户应用
Name[zh_CN]=用户应用

Type=Application
Icon=/usr/share/pixmaps/user_app.png
Exec=/home/myir/user_app --platform linuxfb
Terminal=false
MimeType=application/x-directory;inode/directory;
Categories=System;FileTools;Utility;Qt;FileManager;
```

- 完成上述步骤以后，重新启动开发板，就可以看到用户的应用出现在MEasy本地HMI的界面中。

4 Web HMI使用介绍

本章主要介绍使用Web HMI控制开发板的外围设备，以及使用过程中的注意事项。

软件环境：

- u-boot
- linux-4.1.x
- 带Python2, tornado, python-dbus等运行环境的文件系统
- MEasy Web HMI V1.1应用程序

以上软件已经在出厂的时候烧写到对应的开发板里面了。

硬件环境：

- i.MX6UL系列开发板一块

注意事项：

- 使用前的准备

开机之前先搭建好网络环境，开发板任一以太网接口和远端主机位于同一子网。

- web登录方式

开发板上电，网络连接成功之后串口会打印Web HMI后端服务绑定的IP地址及端口号，log如下：

```
Development server is running at http://192.168.1.100:8090/login
```

在远端主机浏览器中输入：<http://192.168.1.100:8090/login> (此处输入的IP以开发板实际的IP地址为准)进入登录界面。登录用户名和密码默认都是admin。

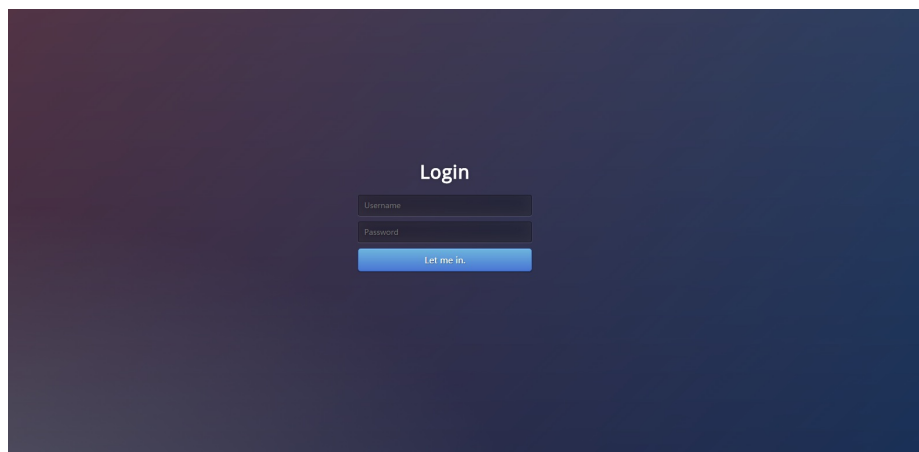


图4 Web HMI 登录界面

- web 语言版本

Web HMI提供了中文和英文两种版本（切换语言时会自动关闭以前打开的模块设备）。

- 同步处理

本地HMI和Web HMI可以同时打开同一个设备，但是他们操作的是同一个设备句柄，设备的参数一样。如本地HMI先打开RS232设备，Web HMI再打开时会读取本地HMI设置的参数来使用，反之亦然。开发板RS232、RS485、CAN接到其它设备发来的数据，在本地HMI和Web HMI上可以同时接受数据并显示。

4.1 串口

本例程演示如何使用Web HMI配置开发板上的 RS232 并使其发送和接收数据。

硬件环境：

硬件连接参见2.1章节。

- 先下拉配置好参数，再点击打开按钮
- 修改配置参数后会先关闭设备，需要再次打开

注意：

界面中的Port的选项可以在board_cfg.json配置文件修改或添加，界面显示如下：



图4-1-1 Web端串口配置界面

4.2 RS485 测试

本例程演示如何使用 web 配置开发板上的 RS485 并使其发送和接收数据。

硬件环境：

硬件连接参见2.2章节。

- 先下拉配置好参数，再点击打开按钮
- 修改配置参数后会先关闭设备，需要再次打开

注意：

界面中的Port的选项可以在board_cfg.json配置文件修改或添加，界面显示如下：



图4-2-1 Web端RS485测试界面

4.3 以太网

本例程演示使用pyconnman对网卡进行管理。

硬件环境：

硬件连接参见2.3章节。

- 在页面可以实时的显示网卡状态，也可以修改设置网卡。
- 修改IP时需要注意，如果修改的是web server使用的网卡，会有提示窗口，点击确认后修改开发板IP，web 服务断开，开发板重启。

注意：

界面中的网卡标签页在有网线连通的时候才会显示，界面显示如下：



图4-3-1 Web端以太网测试界面

4.4 米尔支持

本页面中提供了我司的地址和联系方式等信息。



图4-4-1 米尔支持

5 Web HMI应用开发

Web HMI后端服务是使用python2作为开发语言，基于tornado4.x开发的。i.MX6UL系列开发板使用yocto构建文件系统; 下面详细介绍：

Web HMI应用目录

```
|— application.py
|— handler
|— README.md
|— server.py
|— statics
└— template
```

启动Web HMI应用

在系统启动过程中加入下面的启动脚本启动Web HMI后端服务，启动Web HMI之前需要先启动DBUS和CONNMAN服务。

```
#!/bin/sh
python /usr/share/myir/init_boardcfg.py &

if test -z "$DBUS_SESSION_BUS_ADDRESS" ; then
    eval `dbus-launch --sh-syntax`
    echo "D-Bus per-session daemon address is: $DBUS_SESSION_BUS_ADDRESS"
fi
export DBUS_SESSION_BUS_ADDRESS="$DBUS_SESSION_BUS_ADDRESS"

/home/myir/mxbackend &
python /usr/share/myir/www/server.py &

TS_CALIBRATION_FILE=/etc/pointercal
if [ ! -f $TS_CALIBRATION_FILE ];then
    export TSLIB_TSDEVICE=/dev/input/touchscreen0
    ts_calibrate
fi
/home/myir/mxapp --platform linuxfb &
```

对于yocto，可以将上面的脚本加入到yocto源码文件 fsl-release-yocto/sources/meta-myir-imx6ulx/recipes-myir/myir-rc-local/myir-rc-local 中。

程序启动时会读取开发板的配置文件/usr/share/myir/board_cfg.json，配置文件中定义了RS232、RS485、CAN对应的设备文件节点，dbus的参数，系统性信息、led的信息等，修改此配置文件，Web会对应的改变（需重启Web服务）。

```
{
  "board_info": {
    "rs232": [
      "ttymxc1"
    ],
    "rs485": [
      "ttymxc3"
    ],
    "can": [
      "can0"
    ],
    "led": [
      "myc:blue":"cpu0 - D30 - Core Board"
    ],
    "system": {
      "HMI_version": "MEasy HMI V1.0",
      "linux_version": "linux-4.1.15",
      "uboot_version": "u-boot-2016.03",
      "gcc_version": "arm-linux-gcc 5.3.0",
      "manufacturer": "MYIR Electronics Limited",
      "board": "MYD-Y6ULX",
      "CPU": "i.MX6ULL",
      "memory": "256MB",
      "storage": "256MB"
    }
  },
  "dbus_info": {
    "dbus_name": "com.myirtech.mxde",
    "dbus_path": "/com/myirtech/mxde",
    "dbus_interface": "com.myirtech.mxde.MxdeInterface"
  }
}
```


5.1 i.MX6UL系列开发板上添加运行时库

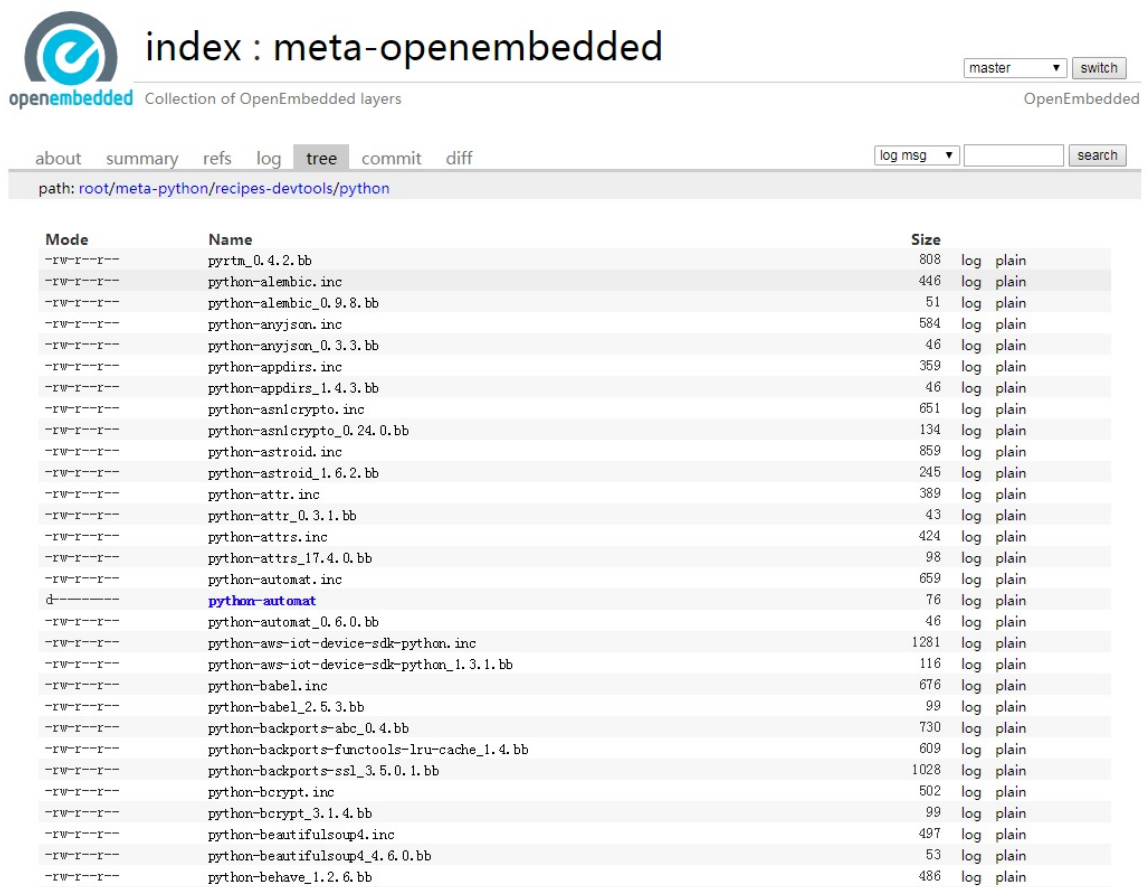
我司提供的Web HMI后端服务是以python开发，在i.MX6UL平台需要添加以下Python库：

- backports_abc-0.5.tar.bz2
- certifi-2017.11.5.tar.bz2
- simplejson-3.8.2.tar.bz2
- singledispatch-3.4.0.3.tar.bz2
- pyconman-0.1.0.tar.bz2
- tornado-4.5.2.tar.bz2

i.MX6UL平台使用yocto可以很方便的添加相关的库并构建系统，在下面的网址，可以查看到官方提供的Python支持文件。

<http://cgit.openembedded.org/meta-openembedded/tree/meta-python>

进入上面的网址中选择recipes-devtools/python，可以看到很多的.bb文件，如下。



The screenshot shows the 'tree' view of the meta-python directory in the OpenEmbedded repository. The page title is 'index : meta-openembedded' and the path is 'root/meta-python/recipes-devtools/python'. The table below lists the files and their sizes.

Mode	Name	Size		
-rw-r--r--	pyrtm_0.4.2.bb	808	log	plain
-rw-r--r--	python-alembic.inc	446	log	plain
-rw-r--r--	python-alembic_0.9.8.bb	51	log	plain
-rw-r--r--	python-anyjson.inc	584	log	plain
-rw-r--r--	python-anyjson_0.3.3.bb	46	log	plain
-rw-r--r--	python-appdirs.inc	359	log	plain
-rw-r--r--	python-appdirs_1.4.3.bb	46	log	plain
-rw-r--r--	python-asn1crypto.inc	651	log	plain
-rw-r--r--	python-asn1crypto_0.24.0.bb	134	log	plain
-rw-r--r--	python-astroid.inc	859	log	plain
-rw-r--r--	python-astroid_1.6.2.bb	245	log	plain
-rw-r--r--	python-attr.inc	389	log	plain
-rw-r--r--	python-attr_0.3.1.bb	43	log	plain
-rw-r--r--	python-atrs.inc	424	log	plain
-rw-r--r--	python-atrs_17.4.0.bb	98	log	plain
-rw-r--r--	python-automat.inc	659	log	plain
d-----	python-automat	76	log	plain
-rw-r--r--	python-automat_0.6.0.bb	46	log	plain
-rw-r--r--	python-aws-iot-device-sdk-python.inc	1281	log	plain
-rw-r--r--	python-aws-iot-device-sdk-python_1.3.1.bb	116	log	plain
-rw-r--r--	python-babel.inc	876	log	plain
-rw-r--r--	python-babel_2.5.3.bb	99	log	plain
-rw-r--r--	python-backports-abc_0.4.bb	730	log	plain
-rw-r--r--	python-backports-functools-lru-cache_1.4.bb	609	log	plain
-rw-r--r--	python-backports-ssl_3.5.0.1.bb	1028	log	plain
-rw-r--r--	python-bcrypt.inc	502	log	plain
-rw-r--r--	python-bcrypt_3.1.4.bb	99	log	plain
-rw-r--r--	python-beautifulsoup4.inc	497	log	plain
-rw-r--r--	python-beautifulsoup4_4.6.0.bb	53	log	plain
-rw-r--r--	python-behave_1.2.6.bb	486	log	plain

图5-1-1 meta python内容

在此可以找到我们需要的文件，官方的资源在不断更新的，名称和版本可能不是完全一致的，根据实际情况选择,如下列表为我司使用的软件版本:

表5-1-1 python库列表

Python库	bb文件	存放路径
backports_abc-0.5.tar.bz2	python-backports-abc_0.4.bb	fsl-release-yocto/sources/meta-openembedded/meta-python/recipes-devtools/python
certifi-2017.11.5.tar.bz2	python-certifi_2018.1.18.bb和python-certifi.inc	fsl-release-yocto/sources/meta-openembedded/meta-python/recipes-devtools/python
simplejson-3.8.2.tar.bz2	python-simplejson_3.8.2.bb	fsl-release-yocto/sources/meta-openembedded/meta-python/recipes-devtools/python
singledispatch-3.4.0.3.tar.bz2	python-singledispatch_3.4.0.3.bb	fsl-release-yocto/sources/meta-openembedded/meta-python/recipes-devtools/python
tornado-4.5.2.tar.bz2	python-tornado_4.3.bb	fsl-release-yocto/sources/meta-openembedded/meta-python/recipes-devtools/python
pyconnman-0.1.0.tar.bz2	python-pyconnman_0.1.0.bb	fsl-release-yocto/sources/meta-openembedded/meta-python/recipes-connectivity/python-pyconnman

这些配置文件上传到了github,用户可以自行下载。

<https://github.com/hufan/yocto-config-bb/tree/master>

有的bb配置文件在yocto源码中已经存在，没有的则需要下载，添加支持的bb文件后，在构建系统时就会从官方网络上下载这些需要的库源码，进行交叉编译。要让这些库交叉编译后整合到文件系统，需修改对应的文件系统bbappend文件,在执行构建系统时执行此事件，在bb文件中加入如下内容。

```
...
libxml2 \
python-lxml \
python-certifi \
python-simplejson \
python-singledispatch \
python-backports-abc \
python-pyconnman \
python-tornado \
...
```

我司提供了三种文件系统的构建，对应bb文件如下：

表5-1-2 i.MX6UL yocto系统配置表

文件系统	bb文件
core-image-minimal	core-image-minimal.bbappend
core-image-base	core-image-base.bbappend
fsl-image-qt5	fsl-image-qt5.bbappend

6 MEasy HMI应用集成

在前面的章节中我们介绍了MEasy HMI在目标板上的目录结构，以及本地HMI和Web HMI的运行环境，开发过程。本章将重点介绍如何将MEasy HMI应用集成到目标板系统当中，使之开机就能启动。详情参考04-Source/fsl-release-yocto-hmi.tar.bz2的yocto代码，使用yocto编译系统时默认构建的就是此系统。

6.1 i.MX6UL系列开发板上集成MEasy HMI应用

Yocto中一个软件包是放在bb文件里的，然后非常多的bb文件集成一个recipe，然后很多的recipe又组成一个meta layer。因此，要加入一个软件包可以在recipe以下加入一个bb（bitbake配置文件）。下面介绍在系统中加入web-demo, 在目录fsl-release-yocto/sources/meta-myrir-imx6ulx/recipes-myrir下建立如下目录结构：

```
|
├─ web-demo
│   └─ web-demo.bb
```

web-demo.bb是对应执行的任务，主要工作是编译代码然后整合软件到rootfs，以shell为开发语言。

web-demo.bb的内容如下：

```
DESCRIPTION = "web demo"
DEPENDS = "zlib glibc ncurses "
SECTION = "libs"
LICENSE = "MIT"
PV = "3"
PR = "r0"

PACKAGES = "${PN}-dbg ${PN} ${PN}-doc ${PN}-dev ${PN}-staticdev ${PN}-locale"
PACKAGES_DYNAMIC = "${PN}-locale-*"

SRCREV = "9b0038497d884db1e11046a8fbc8b219bcd6699c"
SRC_URI = "git://github.com/hufan/web-demo-bb;protocol=https;branch=web_server"
LIC_FILES_CHKSUM = "file://${COMMON_LICENSE_DIR}/MIT;md5=0835ade698e0bcf8506ecda2f7b4f302"
S = "${WORKDIR}/git"
do_compile () {
    tar xvf cJSON.tar.bz2
    make
}

do_install () {
    install -d ${D}/usr/share/myir/
    install -d ${D}/usr/share/myir/www/
    install -d ${D}/lib/
    install -d ${D}/usr/bin/

    cp -s ${S}/*.so* ${D}/lib/
    cp -r ${S}/web_server/* ${D}/usr/share/myir/www/
```

```

if [ ${MACHINE} = "myd-y6ul14x14" ]
then
install -m 0755 ${S}/board_cfg_mydy6ul.json ${D}/usr/share/myir/board_cfg.json
elif [ ${MACHINE} = "myd-y6ull14x14" ]
then
install -m 0755 ${S}/board_cfg_mydy6ull.json ${D}/usr/share/myir/board_cfg.json
elif [ ${MACHINE} = "mys6ul14x14" ]
then
install -m 0755 ${S}/board_cfg_mysy6ul.json ${D}/usr/share/myir/board_cfg.json
elif [ ${MACHINE} = "mys6ull14x14" ]
then
install -m 0755 ${S}/board_cfg_mysy6ull.json ${D}/usr/share/myir/board_cfg.json
fi

install -m 0755 ${S}/mxde.xml ${D}/usr/share/myir/
install -m 0755 ${S}/settings.ini ${D}/usr/share/myir/
install -m 0755 ${S}/psplash ${D}/usr/bin/
install -m 755 ${S}/init_boardcfg.py ${D}/usr/share/myir/
}

FILES_${PN} = "/home/myir/ \
              /usr/share/myir/ \
              /usr/share/myir/www/ \
              /usr/share/myir/www/* \
              /usr/share/myir/*/* \
              /lib/ \
              /usr/bin/ \
              "

TARGET_CC_ARCH += "${LDFLAGS}"
INSANE_SKIP_${PN}-dev = "ldflags"
INSANE_SKIP_${PN} = "${ERROR_QA} ${WARN_QA}"

```

下面介绍在系统中加入qt-demo,在目录fsl-release-yocto/sources/meta-myir-
imx6ulx/recipes-myir下建立如下目录结构：

```

├─ qt-demo
└─ qt-demo.bb

```

qt-demo.bb的内容如下：

```

DESCRIPTION = "qt app"
DEPENDS = "zlib glibc ncurses "
SECTION = "libs"
LICENSE = "MIT"
PV = "3"
PR = "r0"

```

```

LIC_FILES_CHKSUM = "file://${COMMON_LICENSE_DIR}/MIT;md5=0835ade698e0bcf8506ecda2f
7b4f302"

SRCREV = "ba71eadf84c2b57a2a751aae89ac453c7d05bef2"
SRC_URI = " \
    git://github.com/hufan/web-demo-bb;protocol=https;branch=qt-app \
    "
S_G = "${WORKDIR}/git"

do_install () {
    install -d ${D}/usr/share/myir/
    install -d ${D}/usr/share/applications/
    install -d ${D}/usr/share/pixmaps/
    install -d ${D}/usr/lib/fonts/
    install -d ${D}/lib/
    install -d ${D}/home/myir/

    cp -r ${S_G}/applications/* ${D}/usr/share/applications/
    cp -r ${S_G}/pixmaps/* ${D}/usr/share/pixmaps/
    cp -r ${S_G}/msyh.ttc ${D}/usr/lib/fonts/
    cp -rfav ${S_G}/so/*.so* ${D}/lib/
    cp ${S_G}/qt-app/* ${D}/home/myir/

}

FILES_${PN} = "/home/myir/ \
    /usr/share/myir/ \
    /usr/lib/fonts/ \
    /lib/ \
    /usr/share/applications/ \
    /usr/share/pixmaps/ \
    "

#For dev packages only
INSANE_SKIP_${PN}-dev = "ldflags"
INSANE_SKIP_${PN} = "${ERROR_QA} ${WARN_QA}"

```

- SRC_URI : 指定源文件
- LIC_FILES_CHKSUM : 文件和对应的md5值
- do_compile、do_install : 执行bitbake的方法，编译源码和安装程序到文件系统
- FILES_\${PN} : 添加支持的目录
- SRCREV : 指定使用软件的版本，可以根据实际情况修改

然后还需在构建文件系统前加入web-demo.bb任务，参考 表5-1-2 修改对应文件系统的bbappend文件，添加如下内容：

```
...
```

```
web-demo \  
qt-demo \  
...
```

最后开始构建系统，如构建带qt的文件系统，则执行命令：`bitbake fsl-image-qt5`。

关于文件系统的构建，可以参考MYD-Y6ULX发布的文档MYD-Y6ULX-LinuxDevelopmentGuide_zh.pdf的3.3章节。

7. DBUS API介绍

本章节将会列出MYIR Dbus Library库中的接口和网络管理服务Connman提供的dbus接口，并对其中的DBUS接口函数以及信号进行说明。关于MYIR Dbus Library的详细内容用户可以自行参考MxDbus源码。MxDbus中所用到的dbus Method和Signal都在mxdbus目录下的mxde.xml文件中可以看到，在编译本地HMI应用的过程中会生成相应的QT信号和槽，这个过程可以参考源码。

7.1 LED

```

<method name="getLedList">
    <arg name="leds" type="s" direction="out"/>
</method>
<method name="setLedBrightress">
    <arg name="led" type="s" direction="in"/>
    <arg name="brightness" type="i" direction="in"/>
    <arg name="result" type="i" direction="out"/>
</method>
<signal name="sigLedBrightnessChanged">
    <arg name="message" type="s" direction="out"/>
</signal>

```

方法：

`getLedList` 获取开发板所有灯的名称和状态的方法

返回值：

名称	类型	说明	示例
leds	QString	返回所有灯的名字和状态。	"led1 0 \n led2 0 \n"

方法：

`setLedBrightress` 设置LED的状态的方法

输入：

参数	类型	说明	示例
led	QString	led名称。	"led1"
brightness	int	led状态 0表示关 1表示开。	1

返回值：

名称	类型	说明	示例
result	int	执行成功返回0。	0

信号：

`sigLedBrightnessChanged` led状态改变发送的信号

返回值：

名称	类型	说明	示例
message	QString	灯的状态和名称。	"led1 1"

7.2 串口

```

<method name="openSerialPort">
    <arg name="dev_name" type="s" direction="in"/>
    <arg name="uart_fd" type="i" direction="out"/>
    <arg name="tty_configure" type="s" direction="out"/>
<method name="closeSerialPort">
    <arg name="uart_fd" type="i" direction="in"/>
    <arg name="result" type="i" direction="out"/>
</method>
<method name="setSerialPort">
    <arg name="parameter" type="s" direction="in"/>
    <arg name="result" type="i" direction="out"/>
</method>
<method name="getSerialList">
    <arg name="serial_list" type="s" direction="out"/>
</method>
<method name="SerialWrite">
    <arg name="uart_fd" type="i" direction="in"/>
    <arg name="data" type="s" direction="in"/>
    <arg name="size" type="i" direction="in"/>
    <arg name="result" type="i" direction="out"/>
</method>
<signal name="sigSerialRecv">
    <arg name="uart_fd" type="i" direction="out"/>
    <arg name="data" type="s" direction="out"/>
    <arg name="size" type="i" direction="out"/>
</signal>

```

方法：

openSerialPort 打开串口的方法

输入：

参数	类型	说明	示例
dev_name	QString	串口设备的名称	“/dev/ttyO5”

返回值：

名称	类型	说明	示例
uart_fd	int	串口设备打开的句柄。如果串口设备已被打开返回0，此时tty_configure被赋值了可以解析。	4
		串口设备名称 打开句柄 波特率 数据	“/dev/ttyO5”

tty_configure	QString	由设备名称、打开句柄、波特率、数据位、串口模式、流控、校验位、停止位以空格相隔组成的字符串	"/dev/ttyO3 4 300 8 0 0 NONE 1"
---------------	---------	---	---------------------------------------

方法：

closeSerialPort 关闭串口方法

输入：

参数	类型	说明	示例
uart_fd	int	打开串口的句柄	4

返回值：

名称	类型	说明	示例
result	int	执行成功返回0	0

方法：

setSerialPort 设置串口的配置的方法

输入：

参数	类型	说明	示例
parameter	QString	串口配置由波特率、数据位、串口模式、流控、校验位、停止位以空格相隔组成的字符串。串口模式0表示RS232 1表示RS485	"4 115200 8 0 0 78 1"

返回值：

名称	类型	说明	示例
result	int	执行成功返回0	0

方法：

getSerialList 获取开发板上的串口设备的方法

返回值：

名称	类型	说明	示例
serial_list	QString	返回设备上的串口设备列表，以空格隔开	"/dev/ttyO3 /dev/ttyO4"

方法：

SerialWrite 串口设备写数据的方法

输入：

参数	类型	说明	示例
uart_fd	int	打开串口的句柄	4
data	QString	数据字符串	"123456789"
size	int	数据长度	9

返回值：

名称	类型	说明	示例
result	int	执行成功返回0	0

信号：

sigSerialRecv 串口设备收到数据的信号

返回值：

名称	类型	说明	示例
uart_fd	int	串口设备的句柄	4
data	QString	串口设备收到的数据	"123456789"
size	int	数据长度	9

7.3 RS485

```
<method name="getRs485List">
  <arg name="rs485_list" type="s" direction="out"/>
</method>
```

方法：

getRs485List 获取开发板RS485设备列表的方法

返回值：

名称	类型	说明	示例
rs485_list	QString	返回设备上的RS485设备列表，以空格隔开。	“/dev/ttyO5 /dev/ttyO6”

RS485配置接口和读写接口和串口的一致，只是在调用setSerialPort这个方法的时候，传递的参数里面的串口模式应为1 RS485模式。

7.4 CAN

```
<method name="getCanList">
  <arg name="can_list" type="s" direction="out"/>
</method>
<method name="openCanPort">
  <arg name="can_name" type="s" direction="in"/>
  <arg name="can_fd" type="i" direction="out"/>
</method>
<method name="closeCanPort">
  <arg name="can_name" type="s" direction="in"/>
  <arg name="can_fd" type="i" direction="in"/>
  <arg name="result" type="i" direction="out"/>
</method>
<method name="closeCanLoop">
  <arg name="can_name" type="s" direction="in"/>
  <arg name="can_fd" type="i" direction="in"/>
  <arg name="result" type="i" direction="out"/>
</method>
<method name="setCanPort">
  <arg name="can_name" type="s" direction="in"/>
  <arg name="bitrate" type="i" direction="in"/>
  <arg name="status" type="i" direction="in"/>
  <arg name="loop" type="s" direction="in"/>
  <arg name="ret" type="i" direction="out"/>
  <arg name="can_configure" type="s" direction="out"/>
</method>
<method name="CanWrite">
  <arg name="can_fd" type="i" direction="in"/>
  <arg name="data" type="s" direction="in"/>
  <arg name="size" type="i" direction="in"/>
  <arg name="result" type="i" direction="out"/>
</method>
<signal name="sigCanRecv">
  <arg name="can_fd" type="i" direction="out"/>
  <arg name="can_id" type="i" direction="out"/>
  <arg name="can_dlc" type="i" direction="out"/>
  <arg name="can_data" type="s" direction="out"/>
</signal>
```

方法：

getCanList 获取开发板上的CAN设备列表的方法

返回值：

--	--	--	--

名称	类型	说明	示例
can_list	QString	返回设备上的CAN设备列表，以空格隔开。	“can0 can1”

方法：

openCanPort 打开CAN设备的方法

输入：

参数	类型	说明	示例
can_name	QString	CAN设备的名称。	“can0”

返回值：

名称	类型	说明	示例
can_fd	int	CAN设备打开的句柄。	4

方法：

closeCanPort 关闭CAN设备的方法

输入：

参数	类型	说明	示例
can_fd	int	CAN设备打开的句柄。	4
can_name	QString	CAN设备的名称。	"can0"

返回值：

名称	类型	说明	示例
result	int	执行成功返回0。	0

方法：

closeCanLoop 关闭CAN设备回环模式的方法

输入：

参数	类型	说明	示例
can_fd	int	CAN设备打开的句柄。	4
can_name	QString	CAN设备的名称。	"can0"

返回值：

名称	类型	说明	示例
result	int	执行成功返回0。	0

方法：

setCanPort 设置CAN设备的方法

输入：

参数	类型	说明	示例
can_name	QString	CAN设备的名称。	"can0"
bitrate	int	波特率。	115200
status	int	CAN设备开关状态 打开1 关闭0。	1
loop	QString	设置是否打开回环 打开ON 关闭OFF。	"OFF"

返回值：

名称	类型	说明	示例
result	int	执行成功返回0，如果CAN设备已被打开返回100，此时can_configure被赋值了可以解析。	0
can_configure	QString	由设备名、打开的句柄、波特率、回环模式以空格隔开组成的字符串。	"can0 4 20000 OFF"

方法：

CanWrite CAN设备写数据的方法

输入：

参数	类型	说明	示例
can_fd	int	CAN设备打开的句柄。	4
data	QString	数据字符串。	"123456789"
size	int	数据长度。	9

返回值：

名称	类型	说明	示例
result	int	执行成功返回0。	0

信号：

sigCanRecv CAN设备收到数据的信号

返回值：

名称	类型	说明	示例
can_fd	int	CAN设备打开的句柄。	4
can_id	int	CAN数据帧的ID。	0x123
can_dlc	int	CAN数据的长度。	4
can_data	QString	CAN数据。	"0x11 0x22 0x33 0x44"

7.5 Connman

Connman这个网络管理的服务提供的dbus方法和信号比较多，我们这里只讲述我们MEasy HMI用使用到的方法和信号。

```
>
    <method name="GetServices">
        <arg name="services" type="a(oa{sv})" direction="out"/>
    </method>
    <method name="SetProperty">
        <arg name="name" type="s" direction="in"/>
    </method>
    <signal name="PropertyChanged">
        <arg name="name" type="s"/>
        <arg name="value" type="v"/>
    </signal>
    <signal name="ServicesChanged">
        <arg name="changed" type="a(oa{sv})"/>
        <arg name="removed" type="ao"/>
    </signal>
```

方法：

GetServices 获取当前开发板可用的网口服务的方法

返回值：

名称	类型	说明	示例
services	“a(oa{sv})”	QDBusArgument类	示例如下

```
array [
  struct {
    object path "/net/connman/service/ethernet_689e19bc1c84_cable"
    array [
      dict entry(
        string "Type"
        variant
          string "ethernet"
      )
      dict entry(
        string "IPv4"
        variant
          array [
            dict entry(
              string "Method"
              variant
                string "dhcp"
            )
          ]
      )
    ]
  }
]
```

```

        dict entry(
            string "Address"
            variant                                string "192.168.30.120"
        )
        dict entry(
            string "Netmask"
            variant                                string "255.255.255.0"
        )
        dict entry(
            string "Gateway"
            variant                                string "192.168.30.1"
        )
    ]
)
}
]

```

方法：

SetProperty 设置网口信息

输入：

参数	类型	说明	示例
name	QString	网口设置项名称	示例如下

```

string "IPv4.Configuration"
variant    array [
    dict entry(
        string "Method"
        variant                                string "dhcp"
    )
]

```

信号：

PropertyChanged 网口信息改变的信号

返回值：

名称	类型	说明	示例
name	QString	网口设置项名称	"IPv4"
value	QVariant	设置项的值	示例如下

```

variant    array [

```

```

dict entry(
  string "Method"
  variant                                string "dhcp"
)
dict entry(
  string "Address"
  variant                                string "192.168.30.149"
)
dict entry(
  string "Netmask"
  variant                                string "255.255.255.0"
)
]

```

信号：

ServicesChanged 网口变动的信号

返回值：

名称	类型	说明	示例
remove	ao	QDBusArgument类	示例如下
changed	a(oa{sv})	QDBusArgument类	示例如下

```

array [
  struct {
    object path "/net/connman/service/ethernet_689e19bc1c84_cable"
    array [
    ]
  }
]
array [
  object path "/net/connman/service/ethernet_689e19bc1c86_cable" //remove
]

```

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：027-59621648
- 邮箱：support.cn@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。