MEasy HMI Development Guide

MYiR™ Make Your Idea Real

Table of Contents

MEasy HMI V1.0 Development Guide

Introduction

This article mainly describes how to build and run MEasy HMI V1.0 applications on the development boards of `MYiR`, including the development environment setup, source code compilation, instance analysis of MEasy HMI applications, D-Bus library API introduction.

This document is suitable for embedded linux development engineers with QT, Python web-backend and front-end development engineers with certain development experience.

Version History:

Version	Description	Time
V1.0	Initial Version	2018.5.1

Hardware Version: This document applies to AM335X, AM437X and i.MX6UL series development boards of MYIR currently.

> Note: The default password for root of the embedded Linux system is not set.

1. Introduction of MEasy HMI Framework

MEasy HMI is a set of human-machine interfaces which contains a local HMI based on QT5 and a Web HMI based on Python2 back-end and HTML5 front-end. It runs on development boards with LCD, touch panel, ethernet and so on. The dependency software includes dbus, connman and QT5 applications, python, tornado and other components.

MEasy HMI block diagram is shown as below：

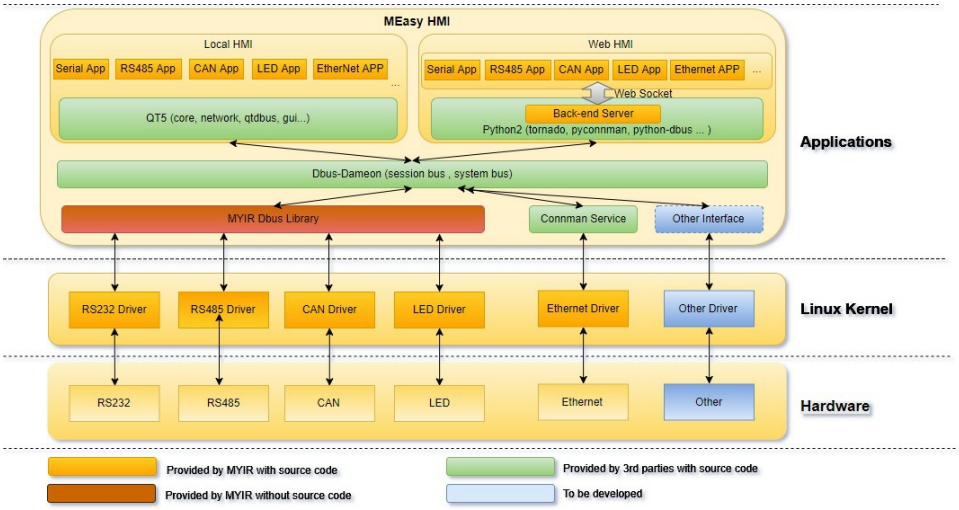

Figure 1-1 MEasy HMI Framework

D-Bus is an advanced inter-process communication mechanism that is provided by the freedesktop.org project and is distributed under the GPL license. The main purpose of D-Bus is to provide communication for processes in the Linux desktop environment, and to pass Linux desktop environment and Linux kernel events as messages to the process.

More details about D-Bus can be found here https://www.freedesktop.org/wiki/Software/dbus/

The MEasy HMI uses D-Bus as the access interface for the QT application and the underlying hardware. The MYIR provides a complete set of control and communication interfaces for RS232, RS485, CAN, and LED hardware and encapsulates the interface into a library for external use based on D-BUS Method and Signal (Chapter 7 describes these methods and Signal).

Connman is software for managing network devices running on embedded linux devices. Connman is a fully modular system that can be expanded by plugins to support the management of EtherNet, WIFI, 3G/4G, Bluetooth and other network devices. .

For more details on Connman, please refer to https://01.org/en/node/2207

The MEasy HMI uses Connman as the EtherNet access interface to manage EtherNet by calling the D-bus based Method and Signal provided by the connman service (Chapter 7 introduces these methods and signals).

The directory structure of MEasy HMI is shown on target boards as below. They will be introduced in the following sections in detail.

```
/
├── 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. How to use local HMI

This section mainly introduces how to use the applications in MEasy HMI.

Software Environment：

- u-boot
- linux-4.1.x
- File system with QT5 operating environment
- MEasy HMI V1.0 application

The above software has been programmed into the corresponding development board.

Hardware Environment：

- MY-TFT070CV2 capacitive screen/HDMI display
- AM437X, AM335X i.MX6UL series development board

> The default factory program only supports the MY-TFT070CV2 capacitive screen. Users using the HDMI display need to compile their own HDMI-compatible programs according to the development board manual.

Hardware connection method:

Table 2-1 LCD Interface of Development Boards

Board	**LCD Interface**
MYD-C437X	J8 LCD_16bit
MYD-C437X-PRU	J20 LCD_16B
Rico Board	J9 LCD
MYD-AM335X	J8 LCD
MYD-AM335X-Y	J7 LCD
MYD-AM335X-J	J8 LCD
MYD-Y6ULX	J3 LCD
MYS-6ULX	J8 LCD

2.1 LED

This example shows how to use the LED application in the MEasy HMI to control the LED on the development board. For details, refer to the source code of mxled.

Software Environment：

- LED Application

Hardware Environment：

- LED on development board

Table 2-1-1 Development board LED list

Board	LED1	LED2	LED3	LED4
MYD-C437X	D36 core board	D34 base board	D35 base board	D36 base board
MYD-C437X-PRU	D36 core board	D20 base board	D19 base board	D18 base board
Rico Board	D23	D24	D25	D26
MYD-AM335X	D3 core board	D40 base board	D39 base board	-
MYD-AM335X-Y	D3 core board	D2 base board	-	-
MYD-AM335X-J	D3 core board	D2 base board	-	-
MYD-Y6ULX	D30	-	-	-
MYS-6ULX	D13	D12	-	-

UI Description：

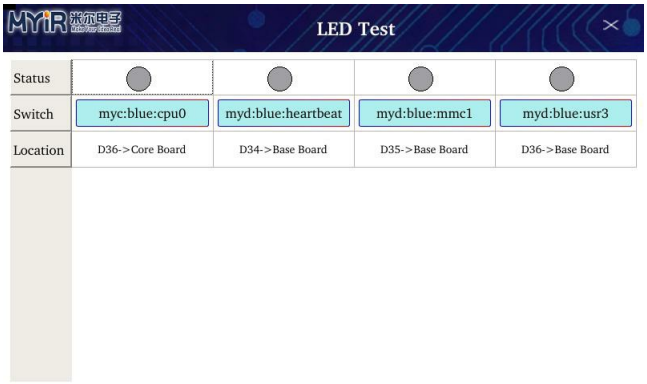

Figure 2-1-1 Local LED test UI

Status bar: indicates the status of current LED.

Switch bar: button to control the switch of current LED.

Position bar: Description of the current silk screen location on the LED redevelopment board.

Test steps:

- The corresponding LED light is controlled by operating the button corresponding to the switch bar.

2.2 Serial

This example shows how to use the serial port application in MEasy HMI to configure the serial port device of the development board and serial port to send and receive data test. For details, please refer to the source code mxserial.

Software Environment：

- Serial Application

Hardware Environment：

- One MYIR development board with serial port
- PC with serial assistant software

Table 2-2-1 Development board serial list

Board	Interface	Data Cable
MYD-C437X	J17 UART3	DB9 To USB
MYD-C437X-PRU	J12 RXD TXD GND	TTL To USB
Rico Board	-	-
MYD-AM335X	J13 UART2	DB9 to USB
MYD-AM335X-Y	J13 UART1	TTL to USB
MYD-AM335X-J	J12 UART1	TTL to USB
MYD-Y6ULX	J10 PIN9、PIN10	TTL to USB
MYS-6ULX	-	-

UI Description：

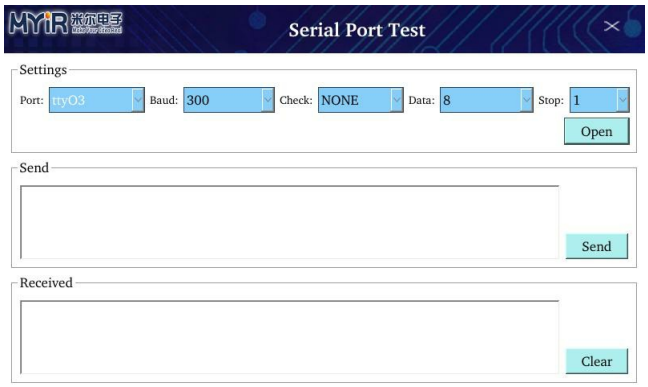

Figure 2-2-1 Local serial port test UI

Test steps:

- Use a cable to connect the USB port on the PC and the serial port on the development board.
- Open the PC serial port assistant software, set the serial port parameters and open the serial port.
- Open the serial port application in MEasy HMI, set the same serial port parameters as the PC and open the serial port.
- Send data on the PC and the development board respectively, and then see if the data can be received on both sides.

2.3 RS485

This routine demonstrates how to use the RS485 application in MEasy HMI to configure the RS485, transmit-and-receive data with RS485. For details, refer to the source code of mxrs485.

Software Environment：

- RS485 Application

Hardware Environment：

- Two development boards with RS485 interface
- Data cable connects the RS485 interface of two boards，485A<->485A，485B<->485B，GND<->GND.

Table 2-3-1 Development board RS485 interface list

Board	Interface	Data Cable
MYD-C437X	J15 485_A 485_B GND	DuPont Line
MYD-C437X-PRU	J10 485A 485B GND	DuPont Line
Rico Board	-	-
MYD-AM335X	U16 UART1	DuPont Line
MYD-AM335X-Y	CON2 UART2	DuPont Line
MYD-AM335X-J	J18 UART2	DuPont Line
MYD-Y6ULX	J10 PIN3、PIN4	DuPont Line
MYS-6ULX	J9	DuPont Line

UI Description：

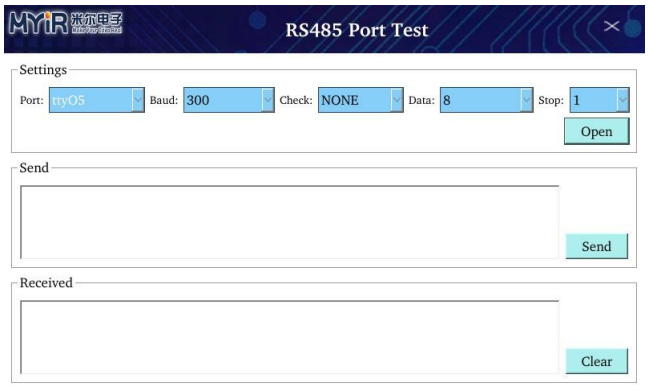

Figure 2-3-1 Development board RS485 configuration

Test steps:

- Connecting RS485 Interface of Two Development Boards with DuPont Cable
- Start the RS485 applications in the MEasy HMI in their respective development boards
- Configure the RS485 configuration group box parameters of the development board, the port may be different, to ensure that the two development board baud rate, parity, data bits and stop bits are consistent.
- Click the Open button, and then send and receive data on the two development boards.

2.4 CAN BUS

This routine demonstrates how to use CAN bus application to configure the CAN bus, transmit-and-receive data with the CAN interface. For details, refer to the source code of mxcan.

Software Environment：

- CAN BUS Application

Hardware Environment：

- Two development boards with CAN interface
- Data cable connects the two boards' CAN interface，CANH0<->CANH0，CANL0<->CANL0，GND<->GND

Table 2-4-1 Development board CAN interface list

Board	Interface	Date Cable
MYD-C437X	J15 CANH0 CANL0 CANH1 CANL1GND	DuPont Line
MYD-C437X-PRU	J10 CANH CANL GND	DuPont Line
Rico Board	-	-
MYD-AM335X	U16 DCAN1	DuPont Line
MYD-AM335X-Y	CON2 DCAN1	DuPont Line
MYD-AM335X-J	J17 DCAN1	DuPont Line
MYD-Y6ULX	J10 PIN3、PIN4	DuPont Line
MYS-6ULX	J13	DuPont Line

UI Description：

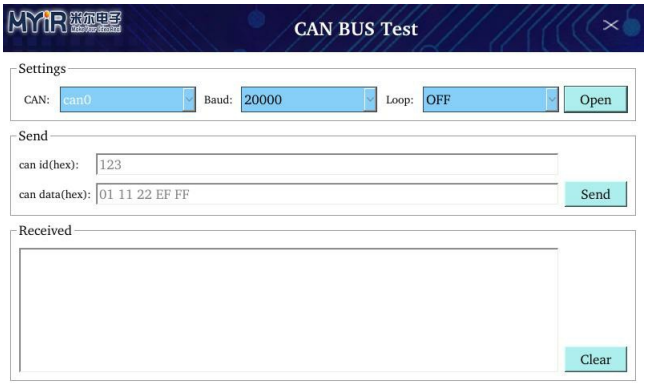

Figure 2-4-1 Development board CAN interface test

Setting group box: Set CAN parameters and turn on CAN

Send group box: Send CAN data, including CAN ID and CAN data

Receive group box: introduce CAN data

> Note:
>
> 1. Click the edit box in the sending group box to pop up the soft keyboard. After entering the data, you need to click the blue Close button on the soft keyboard to close the soft keyboard. Then click the send button to send the data out. After editing, the edit box is sent. The data in the data is automatically cleared.
> 2. The sending data consists of CAN ID and CAN data: The CAN ID is three digits, such as 001, 011, 111; the CAN data is an array of two digits separated by spaces, for example 01 02 03 04 05 06 07 08.
> 3. Some type of boards do not support loopback mode.

Test steps:

- Using the DuPont cable to connect the two development board's CAN interface.
- Launch CAN bus applications in MEasy HMI.
- Configure the parameters of the CAN of each development board. The ports may not be the same. The baud rate of the two development boards must be the same.
- After the configuration is complete, click the `Open` button, and then transmit and receive data on the two development boards.

2.5 Ethernet

This example shows how to use the Ethernet application in the MEasy HMI to configure the development board's network port and test network port connectivity. For details, refer to the source code of mxnet.

Software Environment：

- Ethernet Application

Hardware Environment：

- One router that can provide DHCP service
- One board with Ethernet interface

Table 2-5-1 Development board Ethernet port list

Board	Interface
MYD-C437X	J11 EHT1 J10 ETH2
MYD-C437X-PRU	J6 ETH1 J26 PRU-ETH0 J27 PRU-ETH1
Rico Board	J5 ETHERNET
MYD-AM335X	J5 J6
MYD-AM335X-Y	J5 J6
MYD-AM335X-J	J5 J6
MYD-Y6ULX	CN2 ETH2 CN1 ETH1
MYS-6ULX	CN1 ETH CN1 ETH2

UI Description：

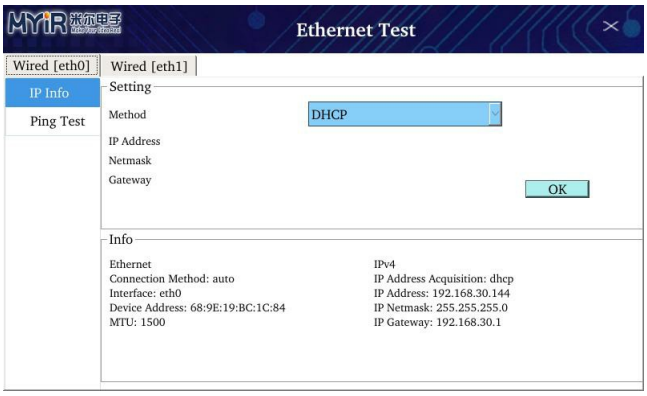

Figure 2-5-1 Development board network port configuration

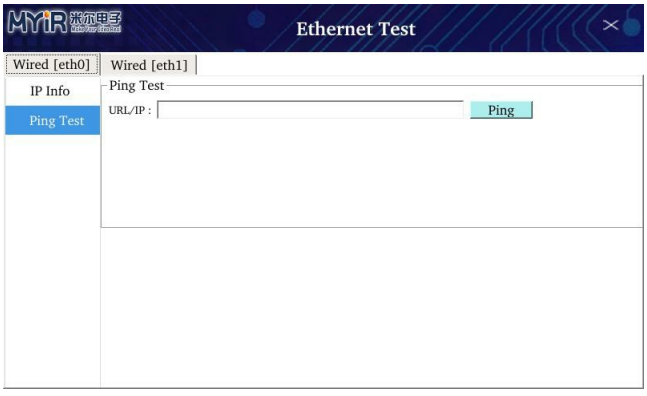

Figure 2-5-2 Development board port test

Tab page: Function page corresponding to network card

IP Information Page: Contains settings group box and information group box

Ping Test: Test Network Connectivity Pages

> Note:
>
> 1. The application will detect if the Ethernet cable is connected or not. Only when the Ethernet cabled is connected to the Ethernet Jack, the configuration and testing page will show up. The number of Ethernet configurations depends connected Ethernet ports.
> 2. When the IP acquisition mode is switched to `Manual` mode, the IP address, subnet mask, and gateway input boxes will pop up, which can be used to configure IP manually.
>
> Test steps:

- Insert the network cable into the Ethernet Jack of the development board.
- Open the Ethernet test application in the MEasy HMI and check whether the information group box has successfully obtained the IP information.
- Switch to `Ping test` page to test network connectivity.

2.6 Task Manager

This routine demonstrates how to use the task manager application in the MEasy HMI to view system resource status and process information. For details, please refer to the source code of mxtaskmanager.

Software Environment：

- Task Manager Application

Hardware Environment：

- Any development boards support MEasy HMI

UI Description：

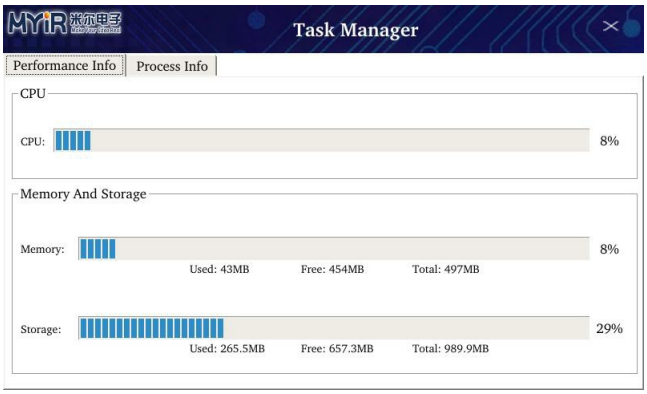

Figure 2-6-1 Development board performance information

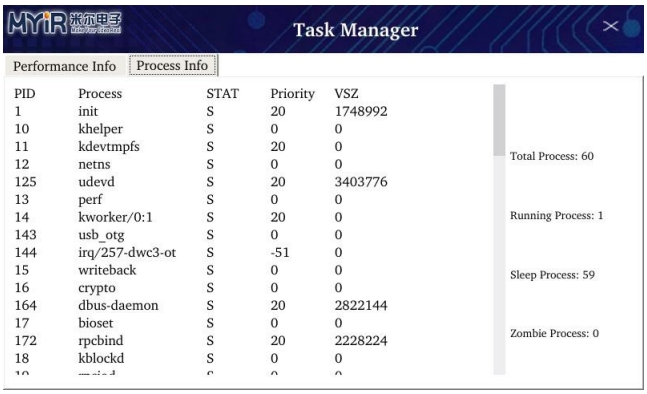

Figure 2-6-2 Development board process information

Performance Information Page: Contains current processor usage, current memory usage, and current storage space usage.

Process information page: Displays all processes and process status information running on the current development board.

> Note: The storage space only shows the size of the root partition and does not represent the size of the entire storage device.

Test steps:

- Open the Task Manager application in the MEasy HMI to view related performance information and process information.

2.7 MYIR Support

This routine demonstrates how to use the MYIR Support application in MEasy HMI to obtain contact information with us. For details, please refer to the source code of mxsupport.

Software Environment：

- MYIR Support Application

Hardware Environment：

- Any development boards supporting MEasy HMI

UI Description：

Figure 2-7-1 Technical Support Information

Use steps:

- Open the MYIR Support application in MEasy HMI.

2.8 System Info

This routine demonstrates how to use the system information application in the MEasy HMI to view the hardware and software information of the development board. For details, refer to the source code of mxinfo.

Software Environment：

- System Info Application

Hardware Environment：

- Any development boards supporting MEasy HMI

UI Description：

Figure 2-8-1 System Information

Use steps:

- Open the System Info application in MEasy HMI.

3. HMI Application Development

This chapter focuses on how to build a development and compilation environment for MEasy HMI, designed to help users develop their products better and faster through QT5. Including the establishment of embedded QT5 runtime environment, the construction of qmake compiler environment, the installation and configuration of QT Creator and the compilation and operation of MEasy HMI applications.

3.1 Setting up the environment

The environment here means the QT5 runtime environment on the development board and the qmake and cross-compiler on the ubuntu host side. The qmake and cross-compiler for AM335X and AM437X on the ubuntu host are compiled by buildroot. The QT5 environment of the i.MX6UL series is compiled using yocto. For details, please refer to the following tables.

Table 3-1-1 Compile QT5 for AM335X, AM437X, i.MX6UL Series Development Board

Board	Document Section
MYD-C437X	MYD-AM437X Series Linux 4.1.18 Development Guide.pdf 3.4 Build QT
MYD-C437X-PRU	MYD-AM437X Series Linux 4.1.18 Development Guide.pdf 3.4 Build QT
Rico Board	Rico Board Linux 4.1.18 Development Guide.pdf 3.4 Build QT
MYD-AM335X	MYD-AM335X Linux 4.1.18 Development Guide.pdf 3.4 Build QT
MYD-AM335X-Y	MYD-AM335X-Y Linux 4.1.18 Development Guide.pdf 3.4 Build QT
MYD-AM335X-J	MYD-AM335X-J Linux 4.1.18 Development Guide.pdf 3.4 Build QT
MYD-Y6ULX	MYD-Y6ULX-LinuxDevelopmentGuide_zh 3.3 Build File System-Build system image with Qt5 package
MYS-6ULX	MYS-6ULX-LinuxDevelopmentGuide_zh.pdf 3.3 Build File System-Build system image with Qt5 package

Table 3-1-2 Install QT Creator

Board	Document Section
MYD-C437X	MYD-AM437X Series Linux 4.1.18 Development Guide.pdf 5.1 Install QT Creator
MYD-C437X-PRU	MYD-AM437X Series Linux 4.1.18 Development Guide.pdf 5.1 Install QT Creator
Rico Board	Rico Board Linux 4.1.18 Development Guide.pdf 5.1 Install QT Creator
MYD-AM335X	MYD-AM335X Linux 4.1.18 Development Guide.pdf 5.1 Install QT Creator
MYD-AM335X-Y	MYD-AM335X-Y Linux 4.1.18 Development Guide.pdf 5.1 Install QT Creator
MYD-AM335X-J	MYD-AM335X-J Linux 4.1.18 Development Guide.pdf 5.1 Install QT Creator
MYD-Y6ULX	MYD-Y6ULX-LinuxDevelopmentGuide_zh.pdf 5.1 Install QT Creator
MYS-6ULX	MYS-6ULX-LinuxDevelopmentGuide_zh.pdf 5.1 Install QT Creator

Table 3-1-3 AM335X, AM437X, i.MX6UL series development board configuration QT Creator

Board	Document Section
MYD-C437X	MYD-AM437X Series Linux 4.1.18 Development Guide.pdf 5.2 Config QT Creator
MYD-C437X-PRU	MYD-AM437X Series Linux 4.1.18 Development Guide.pdf 5.2 Config QT Creator
Rico Board	Rico Board Linux 4.1.18 Development Guide.pdf 5.2 Config QT Creator
MYD-AM335X	MYD-AM335X Linux 4.1.18 Development Guide.pdf 5.2 Config QT Creator
MYD-AM335X-Y	MYD-AM335X-Y 4.1.18 Development Guide.pdf 5.2 Config QT Creator
MYD-AM335X-J	MYD-AM335X-J Linux 4.1.18 Development Guide.pdf 5.2 Config QT Creator
MYD-Y6ULX	MYD-Y6ULX-LinuxDevelopmentGuide_zh.pdf 5.2 Config QT Creator
MYS-6ULX	MYS-6ULX-LinuxDevelopmentGuide_zh.pdf 5.2 Config QT Creator

3.2 Compiling HMI Applications

This chapter mainly describes the compilation process of MEasy HMI.

We provide MEasy HMI source code located in the /04-Linux_Source/MEasy_HMI/mxde.tar.gz directory on the CD-ROM. Copy mxde.tar.gz to the ubuntu directory working directory and extract it.

The following describes how to import mxde project into Qt Creator, open QT Creator, click `File` -> `Open File or Project` in the menu bar and then pop up the box as shown in Figure 3-2-1, enter mxde project directory, click `mxde.pro` and click the `Open` button to open the mxde project.

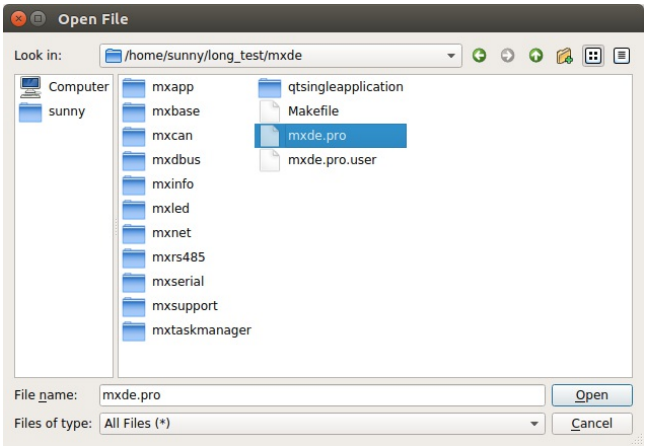

Figure 3-2-1 Project selection box

After opening the project, enter the configuration page, select the compilation tool chain, directly select the kits configured in Chapter 3.1, click the `Configure Project` button, and then enter the project directory.

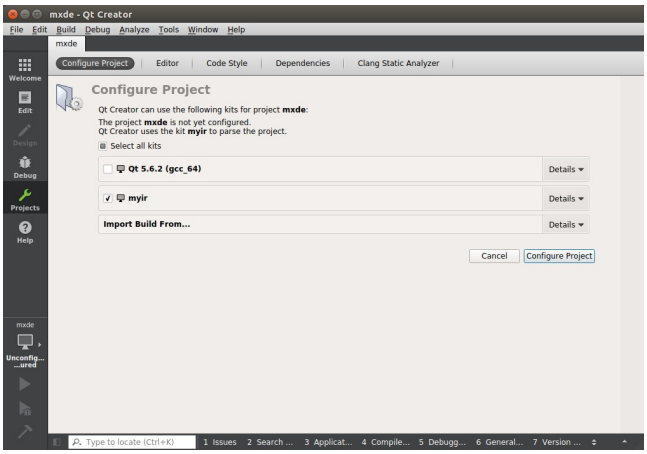

Figure 3-2-2 Config kits

After entering the mxde project, you can see the directory structure of the entire project, as shown in Figure 3-2-3. Then you can compile the project, before compiling can choose to compile the output mode, here we choose Release mode, and then you can select the right lower corner of the small hammer icon to compile the entire project, or by clicking on the menu bar `Build-` > `Build Project "mxde"` to compile the entire project.

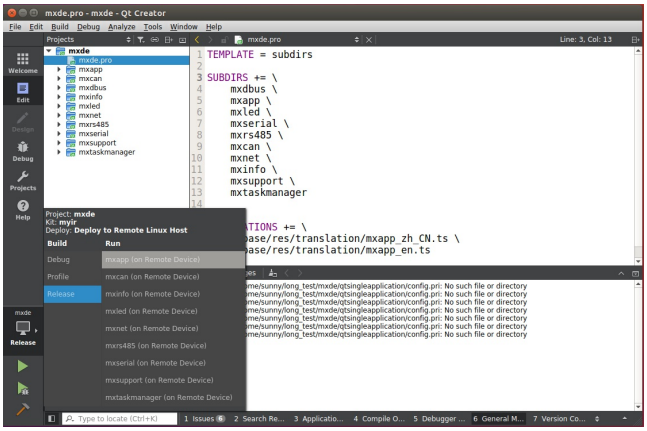

Figure 3-2-3 Project Directory

The compilation process can be seen from the bottom of the Table column `4 Compile Output`, as shown in Figure 3-2-4.

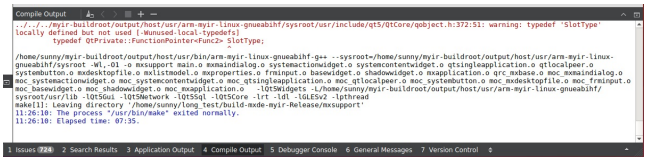

Figure 3-2-4 Compile Output

The errors and warnings that appear in the compilation can be seen from the bottom of the Table column `1 Issues`, as shown in Figure 3-2-5. If the compile error can be output from here for analysis problems.

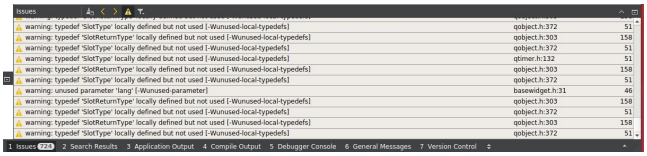

Figure 3-2-5 Issues Output

3.3 Running HMI Applications

This chapter mainly describes the running process of MEasy HMI.

After the compilation is complete, the compiled program can be uploaded to the development board for running. There are two methods for uploading the program to the development board.

Method One: Direct Upload Operation by Configuring Qt Creator

- Configuring Qt Creator remote devices

By selecting the menu bar `Tools->Options->Devices` , select `myir (default for Generic Linux)` in Device, enter the development board IP in the *Type Specific* field (you need to log in to the development board through the serial port to view), user name, and do not need to fill in the password. As shown in Figure 3-3-1.

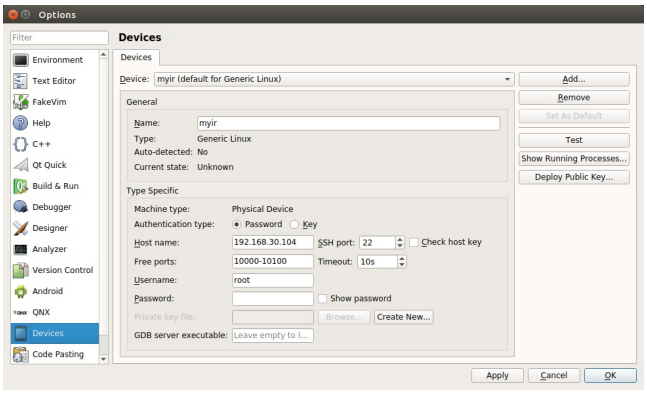

Figure 3-3-1 Device Configuration

- Test remote device connectivity

After the input is complete, click the `Apply` button, and then click the `Test` button on the right side will automatically pop up the test connection window when *Device test finished successfully*. The word means the test connection is successful. As shown in Figure 3-3-2.

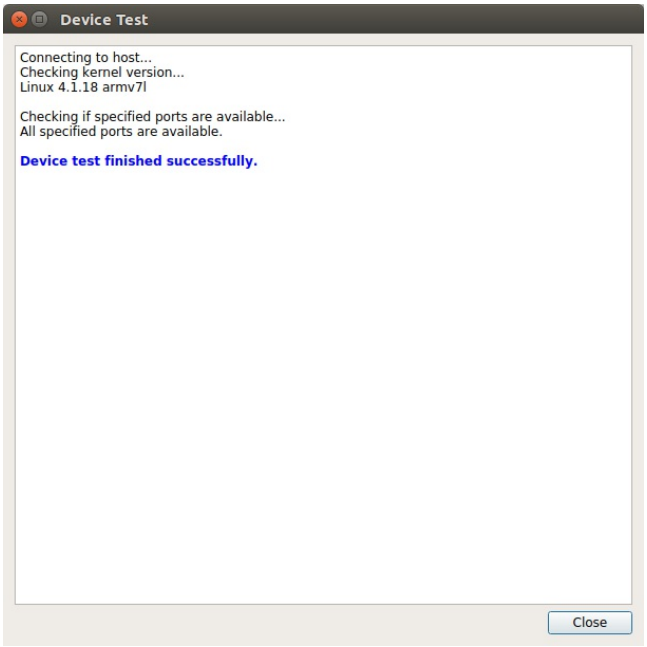

Figure 3-3-2 Equipment testing

- Specify the program to run

After the test connection is successful, return to the main interface of Qt Creator. To specify the program you want to run, select the program that needs to run as *mxapp*. As shown in Figure 3-3-3.

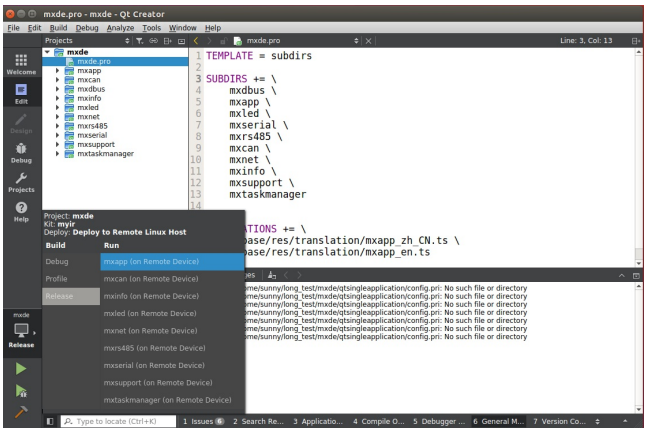

Figure 3-3-3 Select the program to run

- Specify the parameters for the program to run

After specifying the program to be run, you also need to specify the program's operating parameters. Click `Projects->mxde->Build & Run ->myir->Run` to pull down to the Run configuration column and write in the Arguments input box *--platform linuxfb*, which completes the specification of the program's operating parameters. As shown in Figure 3-3-4.

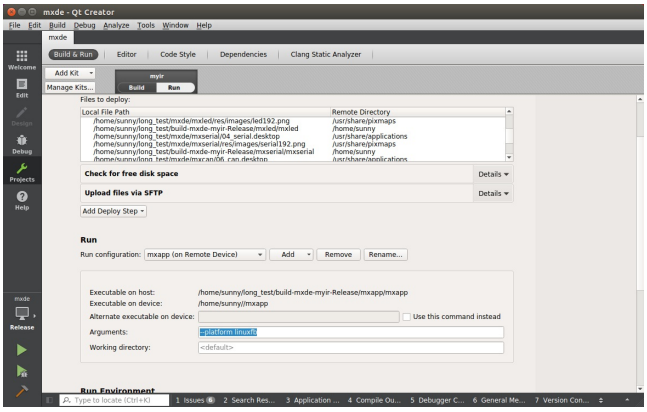

Figure 3-3-4 Program operating parameters

- Kill the running MEasy HMI related program on the development board

After specifying the running parameters, you need to log in to the development board and kill the currently running MEasy HMI related program. The operation is as follows:

```
# killall mxbackend
```

```
# killall mxapp
```

- Upload the program to the development board and run

Click the Run button in the lower left corner, or click the menu bar `Build->Run` to upload mxapp to the development board and run it. On the 7-inch screen, you can see the MEasy HMI interface. The running debugging information can be seen in *3 Application Output*, as shown in Figure 3-3-5.

Figure 3-3-5 Application Output

> Note: If you need to run mxserial mxrs485 mxcan mxled these applications, you need to run mxbackend first, but also need to ensure that these applications and mxbackend connection dbus bus to the same address. The method of operation is as follows:

1.Sets the DBUS_SESSION_BUS_ADDRESS environment variable currently running on the serial terminal.

```
# 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.Run the daemon.

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

3.Configure the dbus return bus DBUS_SESSION_BUS_ADDRESS environment variable to run the program in Qt Creator.

Take mxled as an example to illustrate the configuration in Qt Creator. Click on the left `Projects-> mxde-> Build & Run-> myir -> Run -> Run configureation` Select mxled here, in the *Arguments* input box --platform linuxfb, *in the Run Enviroment,* click the Details button, and then click the `Add` button, The *Variable* column fills in the DBUS_SESSION_BUS_ADDRESS, *Value _column fills in the value created in the first step dbus-launch above _unix:abstract=/tmp/dbus-qb40GAMAnL*. As shown in Figure 3-3-6.

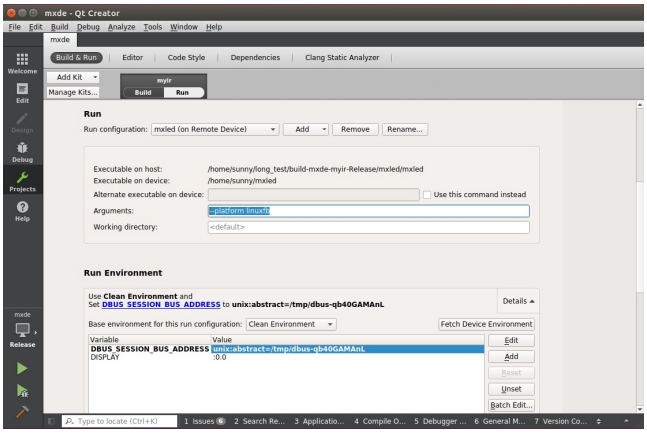

Figure 3-3-6 Application dbus environment variable configuration

4.Click on the `Run` button in the lower left corner, or click on the menu bar `Build->Run` to upload mxled to the development board and run.

Method two: directly copy the compiled program to the development board

Click the `Projects` button on the left, you can see the project's compilation and configuration. The *Build directory* in the General column shows the path of the mxde project's compiled output . You can copy the program directly to the development board from here. As shown in Figure 3-3-7.

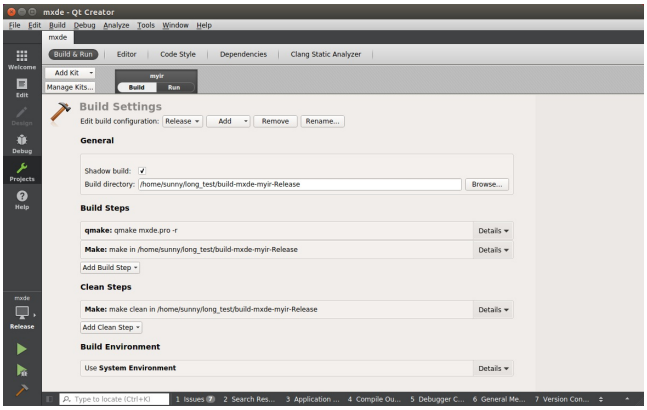

Figure 3-3-7 Project compilation output

Open the compile output directory, enter the mxapp directory, and copy the mxapp application to the development board. The method of operation is as follows:

```
# ./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
QLayout::addChildLayout: layout "" already has a parent
QWidget::setLayout: Attempting to set QLayout "" on HomeContentWidget "", which already has a layout
QLayout: Attempting to add QLayout "" to HomeContentWidget "", which already has a layout
QWidget::setLayout: Attempting to set QLayout "" on HomeContentWidget "", which already 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 Add an Application to HMI

This chapter focuses on the application of the user added to the MEasy HMI.

If users want to display the user's application in the MEasy HMI, it only takes a few steps to complete the operation.

- Create a user's Qt Widgets Application type application named user_app, use the above compiler environment compiled and copied to the development board */home/myir* directory below
- Create a 192*192 resolution icon user_app.png for this application, copy it to the development board_ /usr/share/pixmaps _directory
- Create a desktop configuration file belonging to the user in the development board */usr/share/applications* directory. Start with a number such as 09_user_app.desktop. The contents of the configuration file are as follows:

```
[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;
```

- After completing the above steps, restart the development board and you will see that the user's application appears on the MEasy HMI interface.

4 How to use Web HMI

This chapter mainly introduces how to use Web HMI to control the peripherals of the development boards.

Software Environment：

- u-boot
- linux-4.1.x
- File system with Python2, tornado, python-dbus and other operating environments
- MEasy Web HMI V1.0 Application

Hardwate Environment：

- Prepare one of the AM437X, AM335X, and i.MX6UL series development boards

Note：

- Preparation:

Before starting the system, set up any Ethernet interface on the development board to the same subnet with the remote host.

- Web login

After the development board is powered on and the network is connected, the serial port will print the IP address and port number bound to the Web HMI backend service. The log is as follows:

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

Open `http://192.168.1.100:8090/login` (The IP that is filled in here is based on the actual IP address of the development board) in a browser to login, the user name and password are set to `admin` as default.

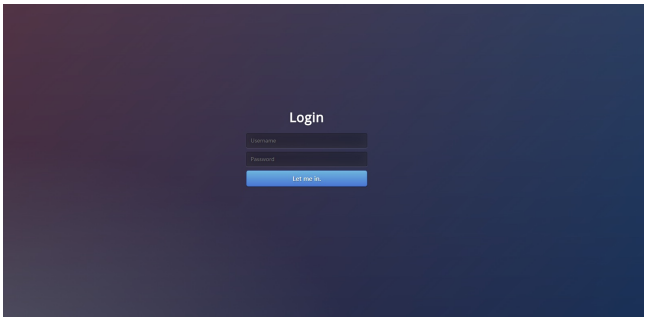

Figure4-1 Web HMI Login

- Web language version

Web HMI provides Chinese and English versions, the default is English version (switching the language will automatically close the previously opened module device).

- Synchronization

Local HMI and Web HMI can open the same device at the same time, they operate on the same device handle with the same configuration. If the local HMI opens the RS232 device first, the Web HMI will read the configuration set by the local HMI and vice versa. If the RS232, RS485, CAN receive data from other devices, the data can be accepted and displayed both on the local HMI and Web HMI.

4.1 LED

Web HMI will automatically read the number and status of the LED on the development board and display it. Click the LED_ON or LED_OFF button to control the light on and off.

Hardware environment：

Please refer to the chapter2.1 for hardware environment.

UI：

Figure4-1-1 Web HMI Test LED

4.2 Serial

This example shows how to use Web HMI to configure the RS232 on the development board, and then test the data send and receive. For details,refer to the source code.

Hardware Environment：

Please refer to the chapter2.2 for hardware environment.

- Select the configured parameters first, then click the Open button
- Modify the configuration parameters will automatically turn off the device, so it need to be opened again.

Note：

Port options in the interface can be modified or added in the board_cfg.json configuration file. **UI：**

Figure4-2-1 Web HMI Test RS232

4.3 Test RS485

This example shows how to use Web HMI to configure the RS485 on the development board, and then test the data send and receive. For details,refer to the source code.

Hardware Environment：

Please refer to the chapter2.3 for hardware environment.

- Select the configured parameters first, then click the Open button
- Modify the configuration parameters will automatically turn off the device, need to open again

Note：
Port options in the interface can be modified or added in the board_cfg.json configuration file. **UI：**

Figure4-3-1 Web HMI Test RS485

4.4 Test CAN Bus

This example shows how to use Web HMI to configure the CAN bus on the development board, and then test the data send and receive. For details,refer to the source code.

Hardware Environment：

Please refer to the chapter2.4 for hardware environment.

- Select the configured parameters first, then click the Open button
- Modify the configuration parameters will automatically turn off the device, need to open again

Note：

Port options in the interface can be modified or added in the board_cfg.json configuration file.

Figure4-4-1 Web HMI Test Can bus

4.5 EtherNet

This example shows how to use Web HMI to manage the network on the development board. For details,refer to the source code,using the pyconnman component.

Hardware Environment：

Please refer to the chapter2.5 for hardware environment.

- The web page can display the network status in real time. You can also set the network information.
- When you modify the IP, you need to pay attention to it. If you modify the network card used by the web server, you will be prompted to modify the development board IP after clicking Confirm. At the same time, the web service is disconnected and the development board restarts.

Note：

The `EtherNet` tab page in the interface is displayed only when the network cable is connected.

Figure4-5-1 Web HMI EtherNet

4.6 Support

This page provides information such as our address and contact information.

Figure4-6-1 Support

5 Web HMI Application Development

The Web HMI back-end service uses python2 as a development language and was developed based on tornado4.x. i.MX6UL series development board uses yocto to build the file system; AM437X, AM335X series development board uses buildroot to build the file system. They are building a Web HMI runtime environment,There will be some differences in the methods, which will be explained separately below.

Web HMI Application Directory

```
├── application.py
├── handler
├── README.md
├── server.py
├── statics
└── template
```

Web HMI Application

Start the Web HMI backend service by adding the following startup script during system startup. You need to start the DBUS and CONNMAN services before starting the Web HMI.

```
#!/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 &
```

For yocto, add the above script to the yocto source file fsl-release-yocto/sources/meta-myir-imx6ulx/recipes-myir/myir-rc-local/myir-rc-local.

When the application starts, it will read the development board configuration file /usr/share/myir/board_cfg.json. The configuration file defines RS232, RS485, CAN corresponding device file nodes, dbus parameters, system information, led information, etc. , modify this configuration file, the Web will correspond to the changes (need to restart the Web service).

```
{
        "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 Tech 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 Add runtime library on i.MX6UL series development board

The Web HMI backend service provided by our company is developed using python. The following Python library needs to be added on the i.MX6UL platform：

```
- 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
- pyconnman-0.1.0.tar.bz2
- tornado-4.5.2.tar.bz2
```

The i.MX6UL platform uses yocto to easily add related libraries and build systems. You can view official Python support files at the following URL.

```
http://cgit.openembedded.org/meta-openembedded/tree/meta-python
```

Go to the above URL select recipes-devtools/python, you can see a lot of .bb file, as follows:

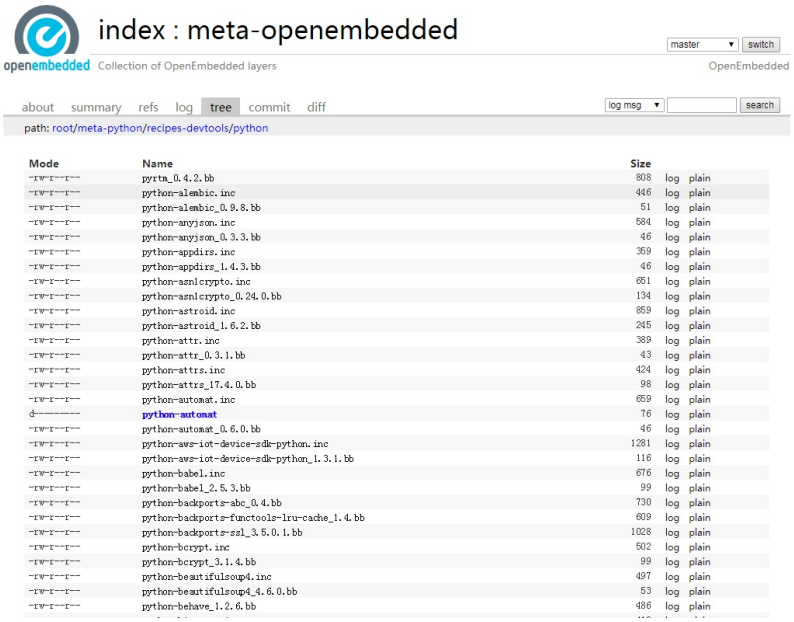

Figure5-1-1 BB files of meta-python

Here we can find the documents we need, the official resources are constantly updated, the names and versions may not be completely consistent, according to the actual situation to choose,The following list is the software version used by our company:

Table 5-1-1 Python Libraries

Python Library	bb	path
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 and 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-	python-pyconnman_0.1.0.bb	fsl-release-yocto/sources/meta-openembedded/meta-

We uploaded these configuration files to github for download.

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

Some BB configuration files already exist in the yocto source code. If not, you need to download them, after adding the supported bb files, the necessary source code libraries will be downloaded from the official network when the system is built and cross-compiled. To enable these libraries to be cross-compiled and integrated into the file system, modify the corresponding file system bbappend file, execute this event when executing the build system, and add the following in the bb file.

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

Our company provides three kinds of file system construction, corresponding bb file is as follows:

Table 5-1-2 i.MX6UL Yocto System Configuration

Filesystem	BB Files
core-image-minimal	core-image-minimal.bbappend
core-image-base	core-image-base.bbappend
fsl-image-qt5	fsl-image-qt5.bbappend

5.2 Add runtime library on AM335X and AM437X Series Development Boards

MEasy Web HMI back-end server is based on python tornado and D-bus, most of the runtime libraries are built by `Buildroot` except for `girepository-1.0` and `gobject-introspection`, which are stored at `myir-buildroot/board/myir/common/HMI/usr/lib`.

The main configuration of Buildroot for MEasy Web HMI is shown as below(Including the QT5 for local HMI):

```
BR2_PACKAGE_DBUS=y
BR2_PACKAGE_DBUS_CPP=y
BR2_PACKAGE_DBUS_GLIB=y
BR2_PACKAGE_DBUS_PYTHON=y
BR2_PACKAGE_DBUS_TRIGGERD=y
# For Web HMI
BR2_PACKAGE_PYTHON=y
BR2_PACKAGE_PYTHON_SSL=y
BR2_PACKAGE_PYTHON_PYCONNMAN=y
BR2_PACKAGE_PYTHON_SETUPTOOLS=y
BR2_PACKAGE_PYTHON_SIMPLEJSON=y
BR2_PACKAGE_PYTHON_TORNADO=y
# For Local HMI
BR2_PACKAGE_QT5=y
BR2_PACKAGE_QT5BASE=y
BR2_PACKAGE_QT5BASE_LICENSE_APPROVED=y
BR2_PACKAGE_QT5BASE_XML=y
BR2_PACKAGE_QT5BASE_GUI=y
BR2_PACKAGE_QT5BASE_WIDGETS=y
BR2_PACKAGE_QT5BASE_OPENGL=y
BR2_PACKAGE_QT5BASE_OPENGL_ES2=y
BR2_PACKAGE_QT5BASE_LINUXFB=y
# BR2_PACKAGE_QT5BASE_FONTCONFIG is not set
BR2_PACKAGE_QT5BASE_GIF=y
BR2_PACKAGE_QT5BASE_JPEG=y
BR2_PACKAGE_QT5BASE_PNG=y
BR2_PACKAGE_QT5BASE_DBUS=y
BR2_PACKAGE_QT5BASE_TSLIB=y
```

6 MEasy HMI Applications Integration

In the previous chapter we introduced the directory structure of the MEasy HMI on the target board, as well as the runtime environment and development process of the local HMI and Web HMI. This chapter will focus on how to integrate the MEasy HMI application into the target board system. Then the MEasy HMI starts up on the development board.

6.1 Integrate MEasy HMI Application on i.MX6UL Series Development Boards

A package in Yocto is placed in a bb file, then a very large number of bb files integrate a recipe, and then many recipes form a meta layer. Users can add a soft Package by adding a bb (bitbake configuration file) file to the recipe. So, we can add web HMI to our system by adding the following web-demo.bb to fsl-release-yocto/sources/meta-myir-imx6ulx/recipes-myir:

```
└── web-demo
    └── web-demo.bb
```

The source code of web-demo.bb is shown as below:

```
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}"
```

The following describes how to integrate the local HMI into our system by adding the qt-demo.bb file to fsl-release-yocto/sources/meta-myir-imx6ulx/recipes-myir:

```
├── qt-demo
   └── qt-demo.bb
```

The source code of qt-demo.bb is shown as below:

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

LIC_FILES_CHKSUM = "file://${COMMON_LICENSE_DIR}/MIT;md5=0835ade698e0bcf8506ecda2f7b4f302"

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 : Specify the source file
- LIC_FILES_CHKSUM : File and corresponding md5 values
- do_compile、do_install : Perform bitbake method, compile source code and install program to file system

- FILES_${PN} : Add a supported directory
- SRCREV : Specifies the version of the software to use. It can be modified according to the actual situation

Then you need to add the web-demo.bb task before building the file system. Refer to Table 5-1-2 to modify the bbappend file of the corresponding file system and add the following content:

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

Finally start building the system, such as building a file system with qt, then execute the command: `bitbake fsl-image-qt5`

For the construction of the file system, refer to the Chapter3.3 of the document MYD-Y6ULX-LinuxDevelopmentGuide_en.pdf published by MYD-Y6ULX.

6.2 Integrate MEasy HMI Application on AM335X,AM437X Series Development Boards

MEasy HMI contains local QT5 applications, Web back-end and Web front-end. After we have finished developing the applications seperately, we should integrate them into the rootfs system. For buildroot, we put the common files of local HMI applications and Web HMI applications into `myir-buildroot/board/myir/common/HMI` , and put the different board configuration file into `myir-buildroot/board/myir/myd_xxx/rootfs-overlay/usr/share/myir/board_cfg.json` .

```
#!/bin/sh -e
# File: board/myir/myd_xxx/post-build.sh

    cp -a board/myir/common/HMI/*  output/target

    ...
```

When builing the rootfs system in `Buildroot` , all the file will be copied to `myir-buildroot/output/target` by a shell scripts `myir-buildroot/board/myir/myd_xxx/post-build.sh` .

7. Introduction of D-Bus API

This chapter focuses on the interface in the MYIR D-Bus library and the D-Bus interface provided by Connman, which is a network management service.

The interface of the MYIR D-Bus library is also created based on D-Bus. The D-Bus Methods and Signals are defined in the `mxde.xml` file in the `mxdbus` directory, it will be explained in detail in the following sections.

The QT applications with MYIR D-Bus library should add `Qt-Dbus` component and put the `mxde.xml' into the project. During the compiling, the corresponding QT signals and slots will be created.

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>
```

Method：

getLedList Method of get the name and status of all lights on the board

Return：

Name	ASCII type-code	Description	Example
leds	s	get the name and status of all lights	"led1 0 \n led2 0 \n"

Method：

setLedBrightress Method of set the state of the LED

Input：

Name	ASCII type-code	Description	Example
led	s	led name	"led1"
brightness	i	led status 0 is off 1 is on	1

Return：

Name	ASCII type-code	Description	Example
result	i	Successful execution returns 0	0

Signal：

sigLedBrightnessChanged Signal of led status has changed

Return：

Name	ASCII type-code	Description	Example
message	s	The status and name of the light.	"led1 1"

7.2 Serial

```
<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>
```

Method：

openSerialPort Method of open the serial port

Input：

Name	ASCII type-code	Description	Example
dev_name	s	Serial device name	“/dev/ttyO5”

Return：

Name	ASCII type-code	Description	Example
uart_fd	i	Serial device open handle. If the serial device has been opened to return 0, then tty_configure is assigned to resolve.	4
tty_configure	s	A string consisting of a device name, open handle, baud rate, data bits, serial port mode, flow control, check bits, and stop bits separated by spaces	“/dev/ttyO3 4 300 8 0 0 NONE 1”

Method：

closeSerialPort Method of close the serial port method

Input：

Name	ASCII type-code	Description	Example
uart_fd	i	Open the handle of the serial port	4

Return：

Name	ASCII type-code	Description	Example
result	i	Successful execution returns 0	0

Method：

setSerialPort Method of configuring the configuration of the serial port

Input：

Name	ASCII type-code	Description	Example
parameter	s	The serial port is configured with a string consisting of baud rate, data bits, serial port mode, flow control, parity, and stop bits separated by spaces. Serial Mode 0 means RS232 1 means RS485	“4 115200 8 0 0 78 1”

Return：

Name	ASCII type-code	Description	Example
result	i	Successful execution returns 0	0

Method：

getSerialList Method of obtaining serial device on development board

Return：

Name	ASCII type-code	Description	Example
serial_liast	s	Return a list of serial devices on the device, separated by spaces.	“/dev/ttyO3 /dev/ttyO4”

Method：

SerialWrite Method serial device write data

Input：

Name	ASCII type-code	Description	Example
uart_fd	i	Open the handle of the serial port	4
data	s	Data string	"123456789"
size	i	Data length	9

Return：

Name	ASCII type-code	Description	Example
result	i	Successful execution returns 0	0

Signal：

sigSerialRecv Signal of serial device receives data

Return：

Name	ASCII type-code	Description	Example
uart_fd	i	Serial device handle	4
data	s	Serial device data received	"123456789"
size	i	Data length	9

7.3 RS485

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

Method：

getRs485List Method of get the list of the development board RS485 devices

Return：

Name	ASCII type-code	Description	Example
rs485_list	s	Return the list of RS485 devices on the device, separated by a space.	“/dev/ttyO5 /dev/ttyO6”

The RS485 configuration interface is the same as the read-write interface and the serial port, but when the setSerialPort method is called, the serial port mode in the passed parameter should be 1 RS485 mode.

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>
```

Method：

getCanList Method of get a list of CAN devices on the development board.

Return：

Name	ASCII type-code	Description	Example
can_list	s	Return the list of CAN devices on the device, separated by a space.	“can0 can1”

Method：

openCanPort Method of open the CAN device

Input：

Name	ASCII type-code	Description	Example
can_name	s	The name of the CAN device.	“can0”

Return：

Name	ASCII type-code	Description	Example
can_fd	i	CAN device opened handle.	4

Method：

closeCanPort Method of close CAN device

Input：

Name	ASCII type-code	Description	Example
can_fd	i	CAN device opened handle.	4
can_name	s	The name of the CAN device.	"can0"

Return：

Name	ASCII type-code	Description	Example
result	i	Successful execution returns 0.	0

Method：

closeCanLoop Method of closing the loop mode of CAN device

Input：

Name	ASCII type-code	Description	Example
can_fd	i	CAN device opened handle.	4
can_name	s	The name of the CAN device.	"can0"

Return：

Name	ASCII type-code	Description	Example
result	i	Successful execution returns 0.	0

Method：

setCanPort Method of set up a CAN device

Input：

Name	ASCII type-code	Description	Example
can_name	s	The name of the CAN device.	"can0"
bitrate	i	Baud rate.	115200
status	i	CAN device switch state Open 1 Close 0.	1
loop	s	Set whether to open loopback ON OFF.	"OFF"

Return：

Name	ASCII type-code	Description	Example
result	i	Execution returns 0 if the CAN device has been opened and returns 100. Can_configure is now assigned and parsed.	0
can_configure	s	A string consisting of space, separated by a device name, opened handle, baud rate, and loopback mode.	“can0 4 20000 OFF”

Method：

CanWrite Method of write data to CAN device

Input：

Name	ASCII type-code	Description	Example
can_fd	i	CAN device opened handle.	4
data	s	Data string.	"123456789"
size	i	Data length.	9

Return：：

Name	ASCII type-code	Description	Example
result	i	Successful execution returns 0.	0

Signal：

sigCanRecv Signal of CAN device receive data

Return：

Name	ASCII type-code	Description	Example
can_fd	i	CAN device opened handle.	4
can_id	i	The ID of the CAN data frame.	0x123
can_dlc	i	The length of the CAN data.	4
can_data	s	CAN data.	"0x11 0x22 0x33 0x44"

7.5 Connman

Connman's network management service provides more dbus methods and signals. Here we only describe the methods and signals used by our MEasy HMI.

```
                    <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>
```

Method：

GetServices Method for obtaining network port service available in current development board

Return：

Name	ASCII type-code	Description	Example
services	a(oa{sv})	QDBusArgument class	Examples are as follows

```
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"
                     )
                  ]
            )
      }
   ]
```

Method:

SetProperty Method of set network port information

Input：

Name	ASCII type-code	Description	Example
name	s	Network port setting item name	Examples are as follows

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

Method：

PropertyChanged Network port change signal

Return：

Name	Type	Description	Example
name	s	Network port setting item name	"IPv4"
value	v	Set the value of the item	Examples are as follows

```
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"
      )
   ]
```

Signal：

ServicesChanged Network port signal

Return：

Name	ASCII type-code	Description	Example
remove	ao	QDBusArgument class	Examples are as follows
changed	a(oa{sv})	QDBusArgument class	Examples are as follows

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

Appendix A Warranty & Technical Support Services

MYIR Electronics 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 Electronics Limited

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

Support Email: support@myirtech.com

Sales Email: sales@myirtech.com

Phone: +86-755-22984836

Fax: +86-755-25532724

Website: www.myirtech.com