Design and Implementation of Vehicle Terminal Graphic Interface Based on QT

Jianwei Zhang1, Xingke Tian1 and Qian Duan1

1North China Vehicle Research Institute, Beijing, 100072, China
*Corresponding author’s e-mail: sxdtzys254@163.com

Abstract. In order to monitor the status of vehicles, the advantages of high real-time, reliability and security in embedded technology are applied. Based on iMX.6 ARM processor, the configuration of cross-environment is completed, and the embedded platform is built. An on-board terminal is developed by using open source graphics library Qt, which realizes the dynamic visualization display of vehicle information.

1. Introduction

Embedded systems are widely used in aerospace, aviation, railway, highway, automobile, home life and other fields. There are a large number of external devices in the system, such as sensors for measuring temperature, pressure, flow and speed, and actuators such as motors. These external devices often need to work in coordination and parallel, that is, the multi-task real-time system, which has higher requirements for the design of the system. When designing a system, we should consider not only the reliability of each external hardware device, but also the resource sharing among multiple devices. Therefore, the design of software and hardware system should take into account the requirements of multi-task and real-time.

Linux operating system is widely used in embedded system design because of its multi-task and real-time characteristics. As a multi-platform GUI system based on C++, QT can provide users with powerful functions to construct graphical interfaces, provide rich multi-threading support and are applied to GUI design of embedded systems. Based on Linux + Qt, this paper realizes the multi-task and multi-thread embedded control system. This paper introduces the application of Qt in the development of embedded graphical interface based on the development of a graphical interface of the vehicle terminal. Based on Linux + Qt, this paper realizes the multi-task and multi-thread embedded control system. This paper introduces the application of Qt in the development of embedded graphical interface based on the development of a graphical interface of the vehicle terminal.

2. Construction of Embedded System Platform

Embedded Linux system usually cannot install a local compiler for local development due to the lack of system resources. It needs to use a host for cross-development. Therefore, a whole embedded system development environment is generally composed of the target board and the host, including the host system environment, cross-compiler environment, system configuration, burning tools and debugging environment. In this paper, we use the connective architecture development mode to connect the host and target board through serial port and Ethernet. We use the embedded development board based on iMX6 processor. The host is a Linux system built on the server, using Ubuntu 14.04. The host is mainly responsible for the establishment of the cross-compiler environment and the
development of embedded applications for ARM sides. The target board is used to debug and run embedded application software.

2.1 Configuration of Shared File System
In embedded Linux development, it is usually necessary to configure the NFS server on the host to share the specific directory of a system to the target system for access and use.

1) Add NFS directory. Modify the /etc/exports file to add an NFS directory and specify the IP and access rights to access the host. The command sudo gedit/etc/exports is executed to open the exports file in the etc directory and add the following contents /home/grosse/nfs*(insecure,rw,sync,no_root_squash). /home/grosse/nfs is a shared directory and files that will be shared by the target board will be placed in this directory.

2) Start the NFS service. The commands sudo/etc/init.d/nfs_kernel-server is executed to start or restart NFS services.

2.2 Construction of Cross-compiler Environment
To make the embedded applications developed on the host run smoothly on the development board across platforms, it is necessary to establish a cross-compiling environment on the host. The cross compiler tool selected in this paper is arm-none-linux-gnueabihf686-pc-linux-gnu.tar.bz2. Following are the steps to install the tool chain:

1) The package file is decompressed and the arm-gcc4.4.3 directory is generated. The directory is placed in a specified directory, such as /usr/local directory. Arm-gcc4.4.3/bin contains executable files of tool chains such as arm-none-linux-gnueabi-gcc, arm-none-linux-gnueabi-g++. The following is the path to the tool chain:

   1) The package file is decompressed and the arm-gcc4.4.3 directory is generated. The directory is placed in a specified directory, such as /usr/local directory. Arm-gcc4.4.3/bin contains executable files of tool chains such as arm-none-linux-gnueabi-gcc, arm-none-linux-gnueabi-g++. The following is the path to the tool chain:

2) Tool link path is set and arm-gcc4.4.3 directory is added to the system PATH environment variable. If installed in /usr/local/arm-gcc4.4.3 directory, cross-compiled directory should be made to /usr/local/arm-gcc4.4.3/bin. Settings: Modify the profile file, add the installation path of the cross-compiler tool chain to the PATH variable, and then run the profile. Execute the following instruction:

   sudo gedit profile to open and edit the document, and add PATH=$PATH:/home/ctools/arm-gcc4.4.3/bin at the end of the document. Run under the user's home directory and execute the following instruction source/etc/profile file to make the settings work.

3. Construction of Embedded GUI Development Environment
The development environment of embedded Qt under Linux generally includes two parts: Qt library and Qt development tool. This paper chooses Qt4.7.3 and Qt Creator 3.5 to design and develop embedded GUI, which only needs to be compiled on a PC. So as long as two Qt libraries are installed, namely, Qt libraries on X11 platform and Qt libraries used by ARM on the target platform. The former is used to develop and debug the GUI application program on the host computer to ensure that the application program can meet the design requirements on the PC, while the latter is used for the cross-compiled embedded application program to run smoothly on the target board.

3.1 Install Qt4/X11
Download the Qt source package: qt-everywhere-opensource-src-4.7.3.tar.bz2. After decompression, enter the qt-everywhere-opensource-src-4.7.3 directory, modify the prefix of built-qt file. According to the real needs, establish a new folder qt, specify qt as the appropriate output directory of compiled results, after modification, run the build-qt file to complete the compilation of Qt.

In order to ensure that the resulting Qt environment can be used, environment variables need to be set, and the /bin directory of QT needs to be added to the system directory. The directory has tools such as qmake needed to compile Qt applications. Add to system directory PATH=$PATH:/home/terminal/qt/bin. After saving and exiting, input qmake -v instruction on the terminal to check whether the path is correct.
3.2 Transplantation of Embedded Qt Library

Compile Qt library of ARM platform into qt-everywhere-opensource-src-4.7.3 directory and configure Qt Library echo yes /configure-opensource-embedded arm -xplatform qws/linux-arm-g++ -no-webkit -qt-libtiff -qt-libmng -qt-mouse-tslib -qtmouse-pc -no-mouse-linuxtp -prefix/opt/qt -I/usr/local/tslib/include -L /usr/local/tslib/lib. In the /configure command, the platform for generating the library is specified by the parameter -embedded arm. Because the target board uses the path compilation and installation make of the tslib library required by the touch screen when configuring the touch screen. The Qt library of the ARM platform compiled by make install is installed in /opt/directory.

Transplant Qt to the target board. When executing Qt programs on the target board, the relevant Qt libraries need to be invoked, so the necessary files such as lib and plugins directories need to be extracted from the /opt/Qt directory and copied to the corresponding directory of the target board /opt/qt. Configuration of environment variables before running Qt application on the target board can be written to script file to execute script file directly.

4. Graphic interface design of vehicle terminal

4.1 Hardware environment

The hardware environment of vehicle terminal graphics interface software is based on a four-core iMX.6 Cortex-A9 ARM SOC processor. The main frequency of ARM core is 1.0GHz. The main frequency of 532MHz GPU is integrated on chip, and the VPU of 1080P@30 frame video processor with hardware coding and decoding core is integrated.

![Hardware Operating Environment](image)

Figure 1. Hardware Operating Environment.

4.2 Interface Graphics Drawing Based on Qt

The Qt drawing system is mainly composed of three parts: QPainter class, QPaintDevice class and QPaintEngine class. Among them, QPainter class is used to perform drawing operations, which is equivalent to the painter, providing drawPixmap, drawImage and other drawing commands. QPaintDevice class is equivalent to canvas, providing QImage, QPixmap, QBitmap, QPicture and other canvas class interfaces. QPaintEngine class provides QPainter class to draw interfaces to different types of devices. It is the concrete realization of basic QPainter class drawing commands.
4.3 Keyboard operation and data acquisition

In the graphical interface software of the vehicle terminal, the operation of interface is accomplished by keyboard information sent by keyboard acquisition software through RS422 serial port. Here, a thread QSerial::QThread uses the thread mechanism of Qt to collect and receive keyboard information. Interface data is collected and received by Flexray bus and placed into the buffer. Here, a thread QBus::QThread is started to receive and send bus data.

Qt threads are mainly composed of thread classes and thread synchronization classes. Among them, QThread is a thread class interface. It provides a platform-independent method to manage threads. Thread synchronization classes include QMutex, QSemaphore, QWaitCondition, QReadWriteLock and so on. It provides thread synchronization interface. In Qt GUI applications, GUI threads are the main threads of GUI and the only threads in Qt that can perform GUI-related operations, but they can have one or more non-GUI threads as working threads at the same time to handle other time-consuming operations in applications.

4.4 Qt signal and slot mechanism

Signal and slot mechanism is the core mechanism of Qt, which is mainly used for communication between objects. It can bind objects that do not understand each other through existing signals and slots or custom signals and slots. The connection mode of signal and slot mainly includes automatic connection, direct connection, queue connection and queue blocking connection. When a signal is connected to a slot, the slot function will be automatically called whenever an object transmits the signal. In particular, the mechanism of signal and slot can support cross-threading connection. Objects in different threads can also use signals to connect with slots, thus realizing the communication between threads.

4.5 Double Buffer Technology

In the process of development, we found that the screen flickered. In response to the drawing message, the form needs complex graphics processing. First, the refresh of the form needs a process of erasing the original image. It fills the form with the background color, and then calls the new drawing code to redraw it, which causes the contrast of the color of the image. When the refresh response is very frequent, this contrast becomes more obvious, and there is a flicker phenomenon.

In order to eliminate flicker, background color filling can be avoided. However, if the original image is not cleaned up at each drawing time, the image will remain. When the form is redrawn, the picture will become messy. Therefore, we use double buffering technology, which is the basic technology in the process of graphics and image processing programming. You can draw in memory first, then copy the completed image to the front desk, and prohibit background refresh, so as to eliminate flicker. When the control draws complex and often refreshed graphs, the control can be permanently saved in an image, ready for the next drawing event. Once a drawing event of control is received, the picture is copied to the control.

5. Conclusion

Based on iMX.6 ARM processor, this paper completes the configuration of the cross-environment, realizes the construction of embedded platform, and designs the graphical interface of the vehicle terminal. This paper expounds the selection of hardware and software, the construction of the platform and the development process of open source graphics library Qt, and puts forward the application method of double buffering technology for screen flicker problems encountered in the development process. Finally, the human-machine graphics interface of the vehicle terminal is realized, which has a friendly interface, stable operation and fast response speed.

References

[1] Peng J, Shi B. (2010) Construction of Embedded GUI Development Platform Based on Qt. Microcomputer Applications, 26:40-42.
[2] Li F. (2008) Qt / Embedded Transplantin and Application under Embedded Linux System. Heilongjiang Science and Technology Information, 27:79-80.

[3] Liu Y. (2011) The Electricity Information Acquisition Terminal Design Based on Linux Technology. Advances in Computer Science, Intelligent System and Environment, 104:89-93.

[4] Dong Y, Han W. (2017) Real-time Research of Linux Operating System Based on Multi-core ARM. Minicomputer System, 38:1262-1266.