UEFI Development Exploration 47 – Porting GUILite to UEFI

原创 luobing4365 Posted on 2020-03-14 16:14:49 Read 3.7k Collection 9 Likes 1

Category Column: UEFI Development Article Tags: UEFI Programming Low-level programming GUI Library UEFI BIOS

(Please keep it-> Author: Luo Bing https://blog.csdn.net/luobing4365)

In the previous blog, a C++ programming framework has been built. Although not all C++ features are supported, such as new and delete, and virtual destructors , etc., it is sufficient for porting GuiLite. Now let's start this work.

1 Code selection

GuiLite has a lot of examples, from controls to animations and even 3D implementations. After browsing around, I chose to start with HelloTimer.

Here is the code running on the STM32F103:

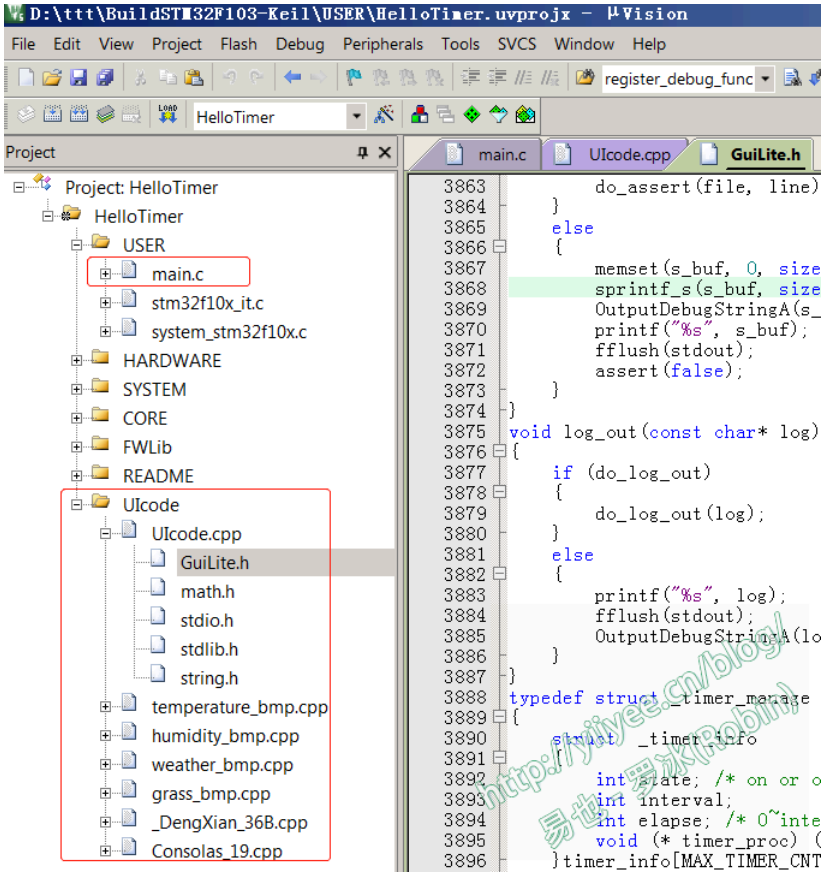

Figure 1 STM32F103 example: HelloTimer

Of course, GuiLite also provides Windows MFC and Linux versions of the code for this routine. I am familiar with MFC, and I started learning GuiLite code by reading MFC code.

The reason for choosing to use the microcontroller code for transplantation is mainly based on two considerations:

1) UEFI is closer to an embedded environment than an operating system, and many operating system mechanisms are not provided. For example, there is no ready-made architecture for the process mechanism, and it must be implemented by yourself;

2) In the STM32F103 code, the C function entry is used to call the C++ code, and the same is true for UEFI. Compared with MFC and Linux , which can directly use the original C++ code, the MCU code is more referenceable.

As shown in Figure 1, the main code that needs to be transplanted is circled in red. Except for the core code GuiLite.h, which needs to be modified, and the code in the main function, which needs to be rewritten, other codes including Uicode.cpp and grass_bmp.cpp basically do not need to be modified too much.

2. Migration and code modification

I transplanted by gradually adding code, mainly using the following steps:

Step 1: Build the framework code

On the previous UEFI code, GuiLite.h of GuiLite was added to solve several problems in the compilation process. The UEFI code I wrote includes graphics programming, keyboard processing, file and peripheral access, etc. In order to avoid too many branches, only the graphics processing code is used to build the framework.

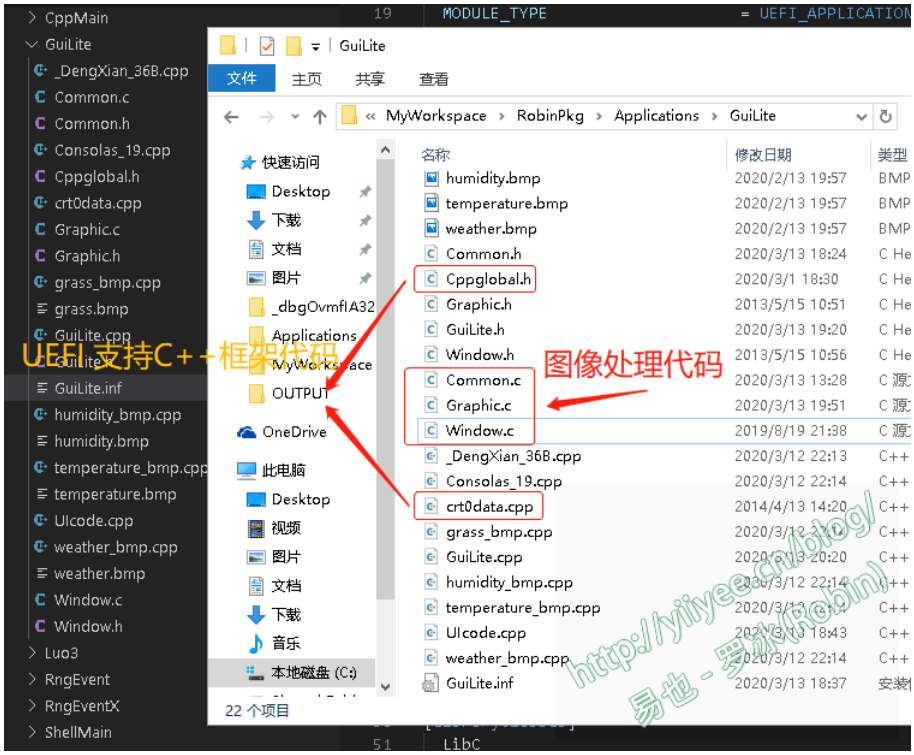

Figure 2 Code classification

Copy the code from HelloTimer (which can be downloaded from the GuiLite repository). Figure 2 shows all the code files after successful transplantation, which can be divided into three categories:

1) Interface functions under UEFI

Including Common.c, Common.h, Graphic.c, Graphic.h, Window.c and Window.h. These codes have been explained in previous blogs, general functions and graphics processing functions;

2) C++ framework code

Includes crt0data.cpp, Cppglobal.h, and there is a calling method in the main function of GuiLite.cpp;

3) GuiLite library related code

Including GuiLite.h, Uicode.cpp and all other cpp files except GuiLite.cpp. The bmp files in the folder are not involved in the compilation, the data has been extracted into the source file, but it is put together for easy viewing.

Step 2: Compile GuiLite.h

Add #include "GuiLite.h" in GuiLite.cpp, and add GuiLite.h in the [Sources] section of GuiLite.inf.

Open the command line compiler and enter the command:

build -p RobinPkg\Robinpkg.dsc -m RobinPkg\Application\GuiLite\GuiLite.inf

You will find many warnings and errors.

The main warning is the problem of data conversion, such as converting int to short, which will cause data truncation. This can be solved by modifying the compilation options or forcing the conversion. Here I use the latter method.

The error prompt is that the virtual destructor will call the delete function. This function cannot be located. Just change it to a normal destructor.

As for the problem of library functions in GuiLite having the same name as UEFI, such as macro MAX and macro MIN, just change the name.

Step 3 : Compile other GuiLite library functions

Add functions such as UIcode.cpp to the compilation, and add these source files to the [Sources] section of GuiLite.inf. The main function actually only needs to call the function startHelloTimer, so remove GuiLite.h from GuiLite.cpp and include it in UIcode.cpp.

After compiling, there are few errors, mainly because mathematical functions such as _cos and _sin cannot be found. In the [LibraryClassess] section of GuiLite.inf, add LibMath and compile successfully.

Also add the source files of the UEFI interface functions. When including the header files in GuiLite.cpp, you need to add the keyword extern "C".

At this point, the framework code is complete.

Step 4: Add working code

There is not much working code that needs to be added, mainly to implement two interface functions: draw_pixel and fill_rect, functions for drawing points and rectangular blocks.

The color in GuiLite library is represented by a UINT type, which can be converted to the color under UEFI. The code for drawing points and rectangular blocks has been implemented in the previous graphics library. The interface is called directly. The specific implementation can be viewed in GuiLite.cpp.

Add other running codes, including setting display mode, setting background, etc., and the porting work is completed.

3 Operation Effect

Compile the code and the running effect is as follows:

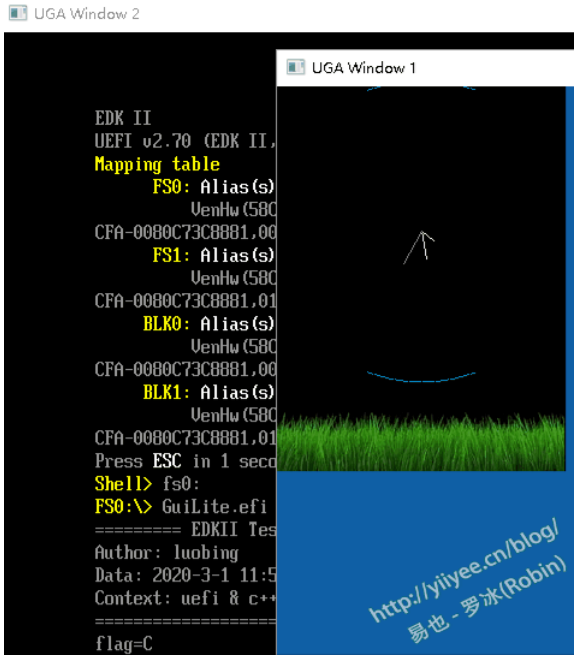

Figure 3 Operational effect after transplantation

Among the routines provided in the GuiLite library, if the code is for STM32 , it can be easily ported according to the above method.

As for porting Windows/Linux routines, there are still two tasks to be done:

One is to realize multi-process. This can be achieved through the event mechanism of UEFI;

The second is the processing of the human-computer interface, including the mouse and keyboard. These codes have been discussed in detail in previous blogs, and all that needs to be done is to integrate them into the current code.

Gitee address: https://gitee.com/luobing4365/uefi-explorer
Project code is located at: / FF RobinPkg/RobinPkg/Applications/GuiLite