Design of Portability Program on Embedded System

Zhenglai Ma[1], Hong He[2], Xiong Zhou[1]
[1]*School of automation, Wuhan university of technology, Wuhan, 430070, China*
[2]*Hebei Engineering and Technology College, Cangzhou 061001, China*

ABSTACT: Aim to Cortex-M embedded chip's driver program can't transplant to different hardware platform smoothly. According to the traditional development model of embedded software, interface abstraction layer are taken from different hardware platforms. The unity of the standard interface provided is provided to hardware driver layer by the middle layer. Through the experiment, driver program can transplant to different Cortex-M kernel chip hardware platforms smoothly, according to the new development model. This interface make program reusable, greatly improve the development efficiency of the program.
KEYWORDS: Cortex-M; Embedded system; Standard interface

1 INTRODUCTION

ARM Cortex-M is extremely popular with a large number of embedded developers[1]. The standard software interface of Cortex-M is unified, but chip peripherals interface is not[2]. The semiconductor manufacturers to develop distinctive chip, they provide board support package (BSP) interface of each is different. Therefore the code reusability is low, in the face of the complex multitasking application functions, it is difficult to a parallel development and debugging, this limits the efficiency of embedded application software development an important factor. This paper mainly studies how to build a unified general peripherals interface and use this interface to achieve diver program transplants smoothly in different Cortex-M hardware platform.

2 BOARD SUPPORT PACKAGE AND EMBEDDED SYSTEM BASIC DRIVE MODEL

BSP is located between hardware and operating system, BSP development in the early stages of the whole system development. Embedded operating systems usually provide BSP reference design, including the driver and startup code. Due to the diversity and complexity of the embedded hardware, developers often need to make big changes to your reference design[3]. Board support package (BSP) is a middle tier between the platform hardware and operating system driver tier program. Generally think, BSP belongs to a part of the operating system, mainly to support of the operating system, and provides API interface to driver program. It can make for operating system run on the hardware platform.

In the field of embedded, MCU communicate with peripheral module via a particular protocol. The protocol includes two layers： Physical layer and Data layer [4-6]. The physical tier agree on communication port, the data tier agree on the way of communication and coding decoding rules. This is the contract programming model, we can communicate with peripheral through the conventions of interfaces and protocols and don't need to worry about internal implementation method of modules. General embedded software system hierarchical model is as figure 1.

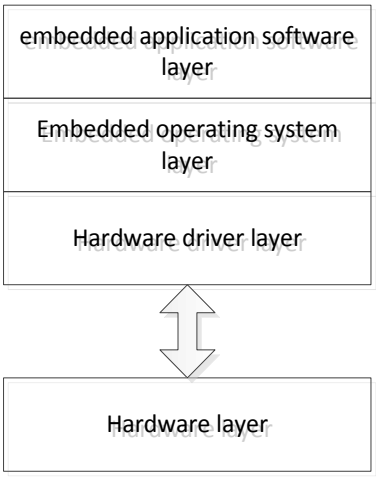

Figure1.Basic hierarchical model of embedded software

Hardware layer contains the basic communication port, communication mode, such as I2C, SPI, UART, CAN, GPIO and so on. Hardware driver layer will package complex protocol instructions into simple interface and provide for the use of the upper layer, at the same time, hardware driver layer translate the command from upper layer into specific signal which peripherals can identify.

3 THE REALIZATION AND VALIDATION OF THE MIDDLE LAYER OF SOFTWARE DEVELOPMENT MODEL

3.1 *The realization of the middle layer of software development model*

Peripheral has different characteristics of different semiconductor manufacturers, but overall, different chip peripherals types tend to be the same. Cortex-M kernel chip function block diagram is as follows:

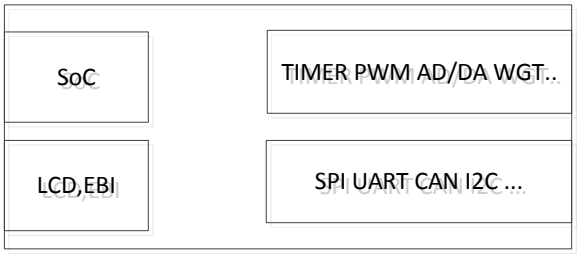

Figure2.Cortex-M kernel chip function diagram\

Cortex-M kernel chip of ST, NUC, TI, Atmel, LPC, contains SPI, I2C, UART, PWM, AD/DA, TIMER, WDT, RTC peripheral. The advantage of different manufacturers of chip point is different. Same manufacturers focus on the low power consumption, same focus on data transmission[7].

According to the basic of the embedded system development model, developers only need to understand adjacent layer interface, don't need to care contact of different layer [8]. Driver code based on BSP can't transplant to another chip smoothly. Driver must be regained, when we change the chip. In order to solve this problem, we improve the system of driver model. As shown in the figure 3.

A middle layer is added between the hardware layer and hardware driver layer. The realization of the middle layer is to extract the upward abstraction layer upward, according to the manufacturer's underlying hardware interface. The middle layer is similar to manufacturer's BSP, but the interface which middle layer provided to the hardware driver layer, is unified. We only need achieve this kind of interface in different chip. The realization of the middle layer can be finished by a third party. Developers as long as to master the standard interfaces, don't have to find different chip interface and cost a lot of time and energy, when developing embedded system in different chip.

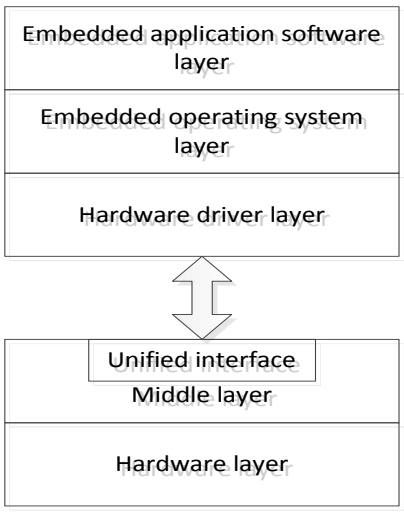

Figure3. Increase the middle layer of embedded software development model

3.2 *The validation of the middle layer of software development model*

In order to verify the new development model is correct, we can use NUC123 hardware platform of Nuvoton. Now we take an example of GPIO.

The function of GPIO configuration:

(1) Input/output functions
(2) The high/low level output
(3) Pull down/on

Define the unified standard GPIO API interface:

xGPIO_Mode() //GPIO mode set
xGPIO_Write() // write a value to a pin output register
xGPIO_Read() //read a value from a GPIO port

First of all, define the GPIO peripheral address:

#define xGPIO_PORTC_BASE 0x50004080

Then define the address of pins:

#define xGPIO_PIN_0 0x00000001
#define xGPIO_PIN_1 0x00000002
#define xGPIO_PIN_2 0x00000004
#define xGPIO_PIN_3 0x00000008
#define xGPIO_PIN_4 0x00000010
#define xGPIO_PIN_5 0x00000020
#define xGPIO_PIN_6 0x00000040
#define xGPIO_PIN_7 0x00000080

Third, achieve the function of register reading and writing. For example, how to write GPIOC output register.

```
void xGPIO_Write(unsigned long ulPort,
                 unsigned long ulPins,
                 unsigned char ucVal)
  {
  xASSERT(GPIOBaseValid(ulPort));
  xHWREG(ulPort + GPIO_DOUT) = ((ucVal&
1)

(xHWREG(ulPort + GPIO_DOUT) | ulPins) :
```

```
(xHWREG(ulPort + GPIO_DOUT)&~(ulPins)));
  }
```

Through the above three steps, we can realize simple operation of GPIO. Developers can use standard interface to operate GPIO and don't worry about the hardware differences.

According to the improvement of the embedded software development model, we program a function. This function show me how to control LED through pin 0 of GPIO port C.

```
  void LED_Toggle(void)
  {
   unsignedlongi;
   xGPIO_Mode
( xGPIO_PORTC_BASE,xGPIO_PIN_0,GPIO_DIR_MO
DE_OUT );
   while (1)
   {
     for(i = 0; i< 0x1FFFF; i++ );
     xGPIO_Write( xGPIO_PORTC_BASE,
xGPIO_PIN_0, 1 );
     for(i = 0; i< 0x1FFFF; i++ );
     xGPIO_Write( xGPIO_PORTC_BASE,
xGPIO_PIN_0, 0 );
   }
  }
```

Compiled, the function achieve on NUC123 development platform of Nuvoton and STM32F107 development platform of ST.

The experimental phenomena as shown in the figure 4:

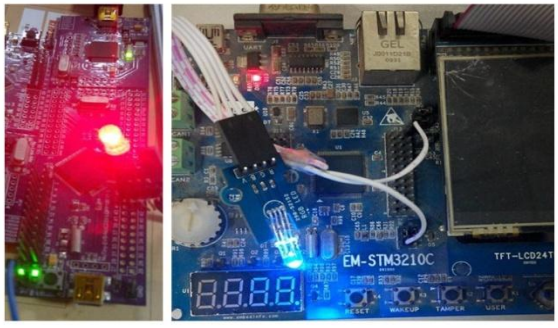

Figure 4.The results on two kinds of development platform

NUC123 development platform is on the left, STM32F107 development platform is on the right. According to the results, the same program achieve function and transplants smoothly on two different kinds of hardware platform.

4 CONCLUSION

A middle layer is added between software development model hardware layer and hardware driver layer. Middle layer achieve the function of reading and writing hardware platform and provide a unified standard interface for the upper. Through the experiment, this kind of development mode is been proved. Thus, program can transplant from one Cortex-M hardware platform to another Cortex-M smoothly. Developers can implement program development and code reuse, only need to master unified standard interface. Program development efficiency and quality has been greatly improved. There are still some problems in the process of the realization of the middle tier. For example, same function does not achieve; the code size increase. Later, the research mainly is to achieve other function and increase code quality and decrease code size.

REFERENCES

[1] 2012-2013 Embedded development practitioners report in China, EEWORLD，2013

[2] Cortex-M Technical Reference Manual, ST, 2012

[3] Xiaowei Xu, Zhixian Wang, Xiaoye Cao. A Kind of Software Architecture for Embedded System Development. Computer Science, 2007(34)

[4] Xiaochuan Pu, Feifei Lu. Investigation and realization of refactoriable BSP, Journal of Computer Applications, 2009(29)

[5] Shaohua Lu. Embedded development platform for the design and implementation of a hardware abstraction layer. Wuhan University of Technology, 2009

[6] Jie Zhou, Chongyu Guo. Research and building of BSP in embedded system. Computer Engineering a nd Design, 2012(10)

[7] NUC200 / 220 Technical Reference Manual V1.00, Taiwan：Nuvoton，2013

[8] Bin Hou, Lichen Zhang. Software architecture model based on middle layer. Electronic Design Enginerring, 2011(10)