Embedded System of DC Motor Speed Control Based on ARM

Helei Wu, Xueqiang Chen, Lingyan Hu
Information Engineering College, Nanchang University, Nanchang Jiangxi,330031,China
wuhelei2000@163.com, cxqfly@yahoo.com.cn, hulingyan@ncu.edu.cn

Abstract

This paper designed a DC motor speed control system. The controller is ARM S3C2410, and the operating system is μC/OS-II, a real time operating system. Designed a closed loop system of motor speed control, adopted the algorithm *of PWM to control the armature voltage, and motor speed is controlled by regulating, of armature voltage. The system has a good respondence.*

1. Introduction

DC motor has a good speed control respondence, wide speed control range. And it is widely used in speed control systems which need high control requirements, such as rolling mill, double-hulled tanker, high precision digital tools, etc.

When it needs control the speed stepless and smoothness, the mostly used way is to adjust the armature voltage of motor. And there are three methods to attain the aim, they are static controlled rectifier, DC chopper, and Pulse Width Modulation(PWM). Since 1970s, self turn-off apparatus appeared, and PWM developed quickly. Because of its high modulating speed, the system has a good dynamic response, so it is widely used in DC servo systems. And now, microcomputer creates digital and high performance for DC motor control system[1].

2. ARM speed control system

The configuration of DC motor speed control system based on ARM is showed in figure 1, and the hardware circuit is figure 2 and 3. ARM S3C2410 is the control system's heart. It reads the input speed number from keys, display the number and the actual motor speed in the LED simultaneously. The actual motor speed is measured by tachogenerator, and the voltage generated by the tachogenerator is measured by A/D. And then, ARM transforms it to corresponding speed values. And ARM calculates the output PWM duty cycle based on the PID algorithm. Then it can regulate the armature voltage, adjust speed[2].

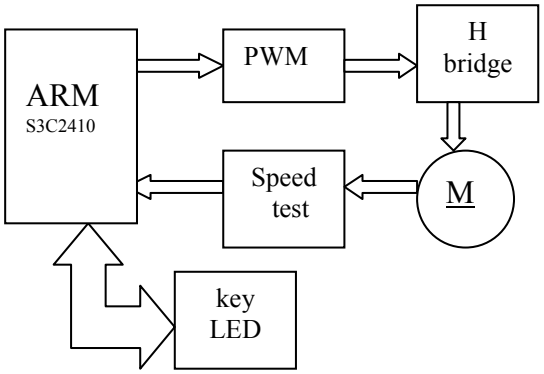

Figure 1. System configuration

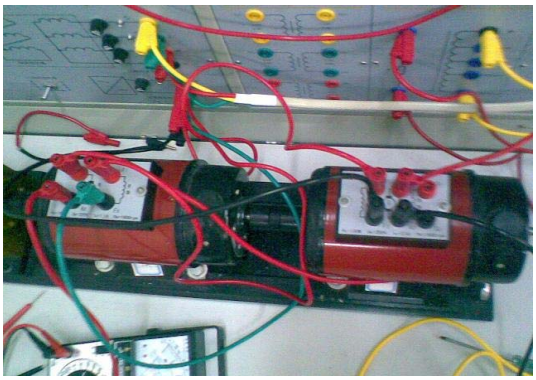

Figure 2. Motor wiring diagram

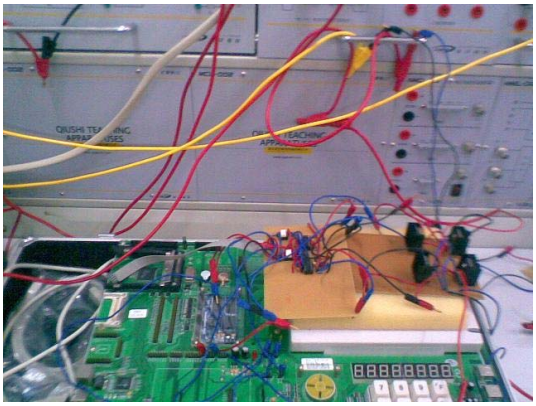

Figure 3. Parts of hardware

978-0-7695-3290-5/08 $25.00 © 2008 IEEE
DOI 10.1109/CCCM.2008.106

2.1 ARM control core S3C2410

ARM embedded system has a small volume, high performance and reliability, and applied to industry[3]. And now, it is used in many fields, such as military affairs, consumer electronics, information appliances, network communications, industrial control, etc. S3C2410 is a 16/32-bit RISC microprocessor of Samsung Electronics Co., Ltd. To reduce total system cost, the S3C2410 includes the following components: separate 16KB instruction and 16KB Data cache, MMU to handle virtual memory management, LCD Controller, NAND Flash Bootloader, System Manager, 3-ch UART, 4-ch DMA, 4-ch Timers with PWM, I/O Ports, RTC, 8-ch 10-bit ADC and Touch Screen Interface, IIC-bus Interface, II-BUS interface, USB Host, USB Device, SD Host & Multi-Media Card Interface, 2-ch SPI and PLL for clock generation[4].

2.2 Hardware of control system

System hardware comprises photoelectricity insulation, keyboard-display, speed measure, H-bridge control, etc. Some parts of system hardware is show as fellows:

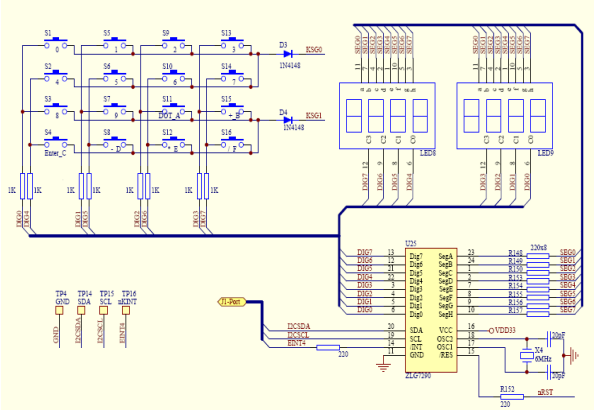

Figure 4. Hardware of keyboard and display

Figure 4 is the configuration of keyboard and LED display. It is controlled by ZLG7290, ZLG7290 has an I^2C interface with CPU, and CPU can read key number when interrupted. 8 bits seven-segment digit display or 64 LED can be controlled by ZLG7290[3].

CPU can read the input speed number when interrupted by key hitting. System stores the number that can be used next.

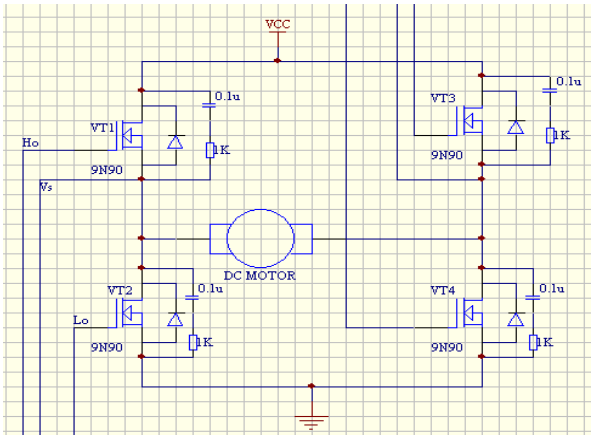

Figure 5. H-bridge PWM control

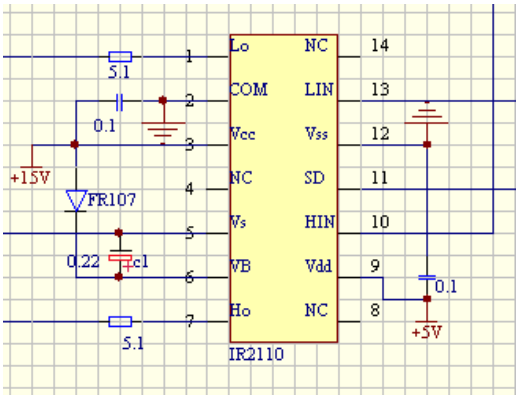

Figure 6. MOSFET driver of H-bridge

Figure 5 and 6 are H-bridge circuit of PWM[5][6]. 4 MOSFET are comprised of it, IR2110 is the driver of MOSFET. The IR2110 is high voltage, high speed power MOSFET and IGBT drivers with independent high and low side referenced output channel[7]. One IR2110 can drive two MOSFET of H-bridge, so system needs two IR2110. Ho and Vs connect to G and S of one upper MOSFET, Lo connects to G of the other lower one. HIN, LIN and SD are the inputs of IR2110, they are compatible with standard CMOS or LSTTL output, down to 3.3V logic. HIN is the logic input for high side gate driver output (HO), LIN is the logic input for low side gate driver output (LO), and SD is the logic input for shutdown. Figure 5 and 6 comprise the bootstrap circuit, C1 is the bootstrap capacitance, C1 is charging from VCC, FR107, C1, the load, and VT2, to make sure that when VT2 is turn-off, VT1 can turn-on by the capacity of C1. So the bootstrap capacitance should be chosen based on the charging and discharging time of the load, and the frequency of HIN and LIN[2].

3. Software of the system

Software of the control system includes A/D sampling, output of PWM, keyboard and display, and the embedded system μC/OS-II, parts of programme listed as below.

3.1 Programme of PWM control

PWM signal is output by the timer, pulse wave as follows.

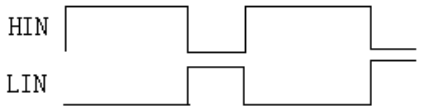

Figure 7. PWM wave

S3C2410 PWM Timers have a double buffering function, enabling the reload value changed for the next timer operation without stopping the current timer operation. So, although the new timer value is set, a current timer operation is completed successfully. In this paper, two channels of PWM are generated by Timer 2 and 3, connected to HIN and LIN.

```
rGPBCON = (rGPBCON & (~(0x03<<4))) | (0x02<<4); // TOUT2 set
void  PWM_ Timer2(uint16 duty)；//Timer2 PWM set
{  // Fclk=200MHz，  Pclk=50MHz. PWM cycle is  1us.
    rTCFG0 =(rTCFG0&~(0xff<<8))|(250<<8);
                                  // prescaler 0 is  250
    rTCFG1 &=~(0xf<<8);
                                 // clock dicider is 1/2
    rTCMPB2 = duty;           // duty cycle
    rTCNTB2 =100;                      // time
    rTCON = (1<<14)|(1<<13);      //update
    rTCON = (1<<12)|(1<<15);      // timer start
}
```

We can control the turn-on or turn-off time by changing the duty.

3.2 Embedded system μC/OS-II

μC/OS-II is a highly portable, ROMable, very scalable, preemptive real-time, deterministic, multitasking kernel, can manage up to 64 tasks, and is simple to use and simple to implement but very effective compared to the price/performance ratio, and supports all type of processors from 8-bit to 64-bit[8]. And now it is widely used in many fields: Engine control, network adapter, industrial robots, cameras, etc.

Figure 8 shows μC/OS-II multitasking management.

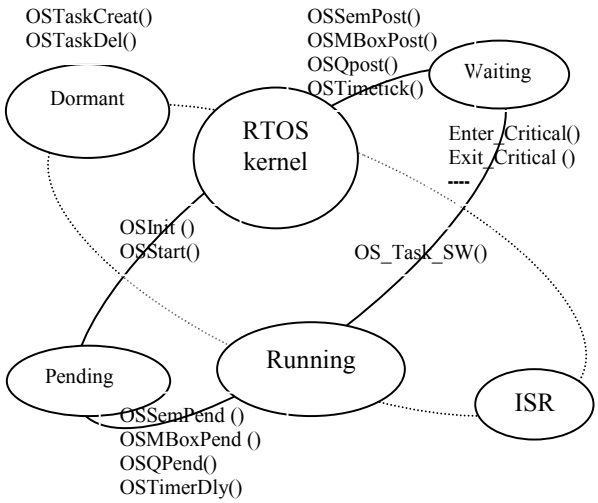

Figure 8. Real time kernel on μC/OS-II

When it is ported to ARM, we should do some revise[8].

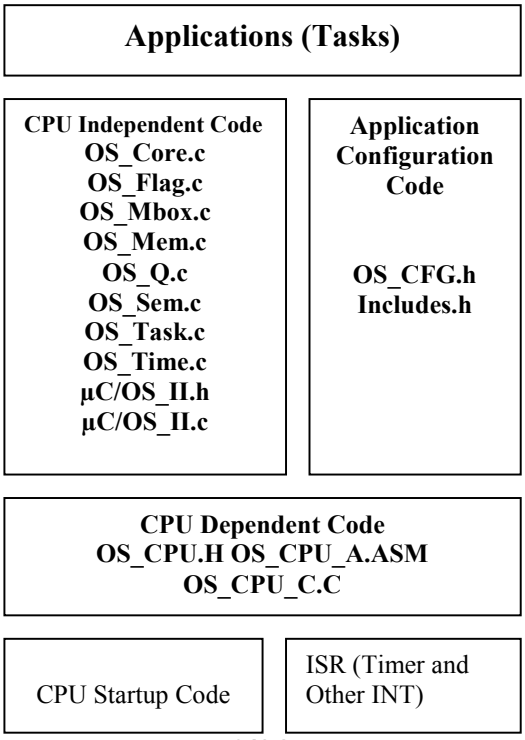

Figure 9. μC/OS-II architecture

From figure 9, we just have to revise almost 3 parts.

(1) Porting of OS_CPU.H

Defining data type which has relationship with CPU in this part.

```
typedef unsigned char  BOOLEAN;
typedef unsigned char  INT8U;
```

```
typedef signed   char  INT8S;
typedef unsigned short INT16U;
typedef signed   short INT16S;
typedef unsigned int   INT32U;
typedef signed   int   INT32S;
typedef  float  FP32;
typedef double  FP64;
typedef  INT32U  OS_STK;
```

µC/OS-II defines two macros to disable and enable interrupts: OS_ENTER_CRITICAL() and OS_EXIT_CRITICAL(), respectively.

```
#define  void OS_ENTER_CRITICAL()
      // disable interrupts
#define OS_EXIT_CRITICAL()
      // enable  interrupts
#define  OS_STK_GROWTH       1
      //  Define stack growth: 1 = Down, 0 = Up
```

⑵ Porting of OS_CPU_C.C

It requires that you write 10 functions:

- OSTaskStkInit();
- OSTaskCreateHook();
- OSTaskDelHook();
- OSTaskSwHook();
- OSTaskIdleHook();
- OSTaskStatHook();
- OSTaskTickHook();
- OSInitHookBegin();
- OSInitHookEnd();
- OSTCBInitHook();

Function OSTaskStkInit() is the only necessary one, the others can just only been declared.

⑶ Porting of OS_CPU_A.S

You should write 4 simple assembly language functions:

- OSStartHighRdy();
- OSCtxSw();
- OSInitCtxSw();
- OSTickISR();

µC/OS-II must have at least one task. The task should been written as a non-returning infinite loop, for example:

```
void Task0(void *pdata)
{
      INT8U err;
      InitTimer();  //  Timer initialization
       ┆
      for(;;){}
}
```

Writing function main(), it consists of user's code, µC/OS-II initialization, task start:

```
void main (void)
{
 OSInit ();
      //  Initialize µC/OS -II
 OSTaskCreate
(Task0,(void*)0, &Task0Stk[Task0StkLengh - 1], 2);
      //  Task0:  system  initialize
 OSTaskCreate
(Task1,(void*)0, &Task1Stk[Task1StkLengh - 1], 3);
      //  Task1: A/D sampling, keyboard, LED display
OSStart ();
      / /  Start Multitasking
return 0;
}
```

OSInit() must be called prior to calling OSStart(). OSStart() should only be called once by your application code.

4. Conclusion

The control system runs well, and has a good system respondence. Because of the high integration of ARM, system has a small volume, low cost, and been easily programmed. The system designed in this paper, can deal with many middle and small power motor control system. ARM control system has a complete set of common peripherals compared with others, gives system a strong task management and real time respondence. When added more external attachment, it really minimizes overall system costs and eliminates the need to configure additional components.

Reference

[1] Chen Boshi. Electric driving automatic control system, Machine Press, Beijing, China, 2006,5

[2] Ma Ruiguo, Liu Weiguo. Special Application of Bootstrap Integrated Drive Circuit IR2110, Power Electronics, 2000,1, pp. 31-33

[3] Zhou Ligong. MagicARM2410 teaching and experiment platform experiment guider, Guangzhou ZhiYuan Electronic Co.,Ltd, 2006,11

[4] Samsung Electronics. S3C2410A – 200MHz & 266MHz 32-Bit RISC Microprocessor Revision 1.0, pp. 1-1.

[5] Mojtaba Mohaddes Khoraddani. Neural network controlled optimal pulse-width modulation for a voltage source converter, University of Manitoba,Canada, 2001,1, pp. 24-29

[6] Tony C.Huang. High performance electric drive systems using fuzzy control. University of Washington, USA, 1995.6, pp. 62-66

[7] International Rectifier. IR2110(S)/IR2113(S)&(PbF) data sheet

[8] Jean J.Labrosse. Micro/OS-II The Real-Time Kernel Second Edition. Buaapress, Beijing, China, 2003,5