目录

[汇编重点20% 1](#_Toc133254609)

[作业3 1](#_Toc133254610)

[startup.s 1](#_Toc133254611)

[作业4 2](#_Toc133254612)

[中断机制20% 6](#_Toc133254613)

[cs\_led.c 6](#_Toc133254614)

[cs\_key.c 8](#_Toc133254615)

[cs\_key.h 11](#_Toc133254616)

[cs\_beep.c 11](#_Toc133254617)

[cs\_beep.h 11](#_Toc133254618)

[时钟定时功能10-15% 12](#_Toc133254619)

[systick.c 12](#_Toc133254620)

[systick.h 13](#_Toc133254621)

[cs\_beep.c 13](#_Toc133254622)

[cs\_key.c 14](#_Toc133254623)

[main.c 16](#_Toc133254624)

[定时器复用功能20-25% 17](#_Toc133254625)

[cs\_key.c 17](#_Toc133254626)

[cs\_beep.c 20](#_Toc133254627)

[main.c 23](#_Toc133254628)

[串口通信20-25% 24](#_Toc133254629)

[usart.c 24](#_Toc133254630)

[track.c 27](#_Toc133254631)

[motor.c 27](#_Toc133254632)

[一些其他的补充知识 30](#_Toc133254633)

[作业1 30](#_Toc133254634)

[作业2 31](#_Toc133254635)

# 汇编重点20%

作业3

startup.s

stack\_size EQU 0x200 ;512B

vectors\_size EQU 0x400 ;1KB

;define stack

AREA m\_stack, NOINIT, READWRITE

\_stack

SPACE stack\_size

\_stack\_top

;define INT-VEC

AREA RESET, DATA, READONLY

\_vectors

DCD \_stack\_top;first is stack top

DCD test\_code ;second is begin code

SPACE vectors\_size

\_vectors\_end

;define code

AREA m\_code, CODE, READONLY, ALIGN=3

test\_code

;my test code

B . ;while(1)

END

作业4

**用汇编语言实现一个函数，判断a是否为b的倍数。**

**C:**

**/\***

**return 1: a是b的倍数**

**return 0: a不是b的倍数**

**\*/**

**int Is\_Multi(int a, int b) 3,2**

**{**

**int i = b; 2**

**while (b <= a)**

**{**

**if (a == b)**

**{**

**return 1;**

**}**

**b = b + i;**

**}**

**return 0;**

**}**

stack\_size EQU 0x200 ;512B

vectors\_size EQU 0x400 ;1KB

;define stack

AREA m\_stack, NOINIT, READWRITE

\_stack

SPACE stack\_size

\_stack\_top

;define data

AREA m\_data, DATA, READWRITE

\_data\_a

SPACE 4

;define INT-VEC

AREA RESET, DATA, READONLY

\_vectors

DCD \_stack\_top ;first is stack top

DCD test\_code ;second is begin code

SPACE vectors\_size

\_vectors\_end

;define code

AREA m\_code, CODE, READONLY, ALIGN=3

test\_code PROC

;my test code

MOV R0,#8;R0 a

MOV R1,#2;b

BL Is\_Multi

LDR R1,=\_data\_a

STR R0,[R1];R0-->data\_a

B .

ENDP

Is\_Multi PROC

PUSH {R2-R3,LR}

MOV R2,R0;R2-a-R0

MOV R0,#0;0 is not multi

MOV R3,R1;i(R3) = b(R1)

\_loop

CMP R2,R1;a--b

BLT \_loop\_end;< LT-end

MOVEQ R0,#1;1 is multi--R0

BEQ \_loop\_end;RETURN

ADD R1,R1,R3;b = b + i

B \_loop

\_loop\_end

POP {R2-R3,PC}

ENDP

END

**用汇编语言实现一个函数，判断一个数m是否为质数。**

**C:**

**int Is\_Prime(int x)**

**{**

**int i ;**

**for (i = 2; i < x; i++)**

**{**

**if (Is\_Multi(x, i))**

**{**

**return 0;**

**}**

**}**

**return 1;**

**}**

stack\_size EQU 0x200 ;512B

vectors\_size EQU 0x400 ;1KB

;define stack

AREA m\_stack, NOINIT, READWRITE

\_stack

SPACE stack\_size

\_stack\_top

;define data

AREA m\_data, DATA, READWRITE

\_data\_a

SPACE 4

;define INT-VEC

AREA RESET, DATA, READONLY

\_vectors

DCD \_stack\_top ;first is stack top

DCD test\_code ;second is begin code

SPACE vectors\_size

\_vectors\_end

;define code

AREA m\_code, CODE, READONLY, ALIGN=3

test\_code PROC

;my test code

MOV R0,#4;R0 a

BL Is\_Prime

LDR R4,=\_data\_a

STR R0,[R4];R0-->data\_a

B .

ENDP

Is\_Prime PROC

PUSH {R1-R3,R5,LR}

MOV R2,R0;R2-m(R0)

MOV R5,#1;1 is prime --R5

MOV R3,#2;i=2--R3

\_loop\_m

CMP R3,R2;i--m

BGE \_loop\_m\_end;i>=m GE--end

MOV R0,R2;(R0)a-m(R2)

MOV R1,R3;(R1)b-i(R3)

BL Is\_Multi

CMP R0,#1;if(1 is multi)

MOVEQ R5,#0;R0==1--return 0 is not prime

BEQ \_loop\_m\_end

ADD R3,R3,#1;i++

B \_loop\_m

\_loop\_m\_end

POP {R1-R3,R5,PC}

ENDP

Is\_Multi PROC

PUSH {R2-R3,LR}

MOV R2,R0;R2-a-R0

MOV R0,#0;0 is not multi

MOV R3,R1;i(R3) = b(R1)

\_loop

CMP R2,R1;a--b

BLT \_loop\_end;< LT-end

MOVEQ R0,#1;1 is multi--R0

BEQ \_loop\_end;RETURN

ADD R1,R1,R3;b = b + i

B \_loop

\_loop\_end

POP {R2-R3,PC}

ENDP

END

# 中断机制20%

cs\_led.c

#include "cs\_led.h"

//D1-PF9 D2-PF10 D3-PE13 D4-PE14

//OUT PP 50M NOUPDN LED-ON/OFF

void led\_init(void)

{

//clk\_enable PF PE

RCC\_AHB1PeriphClockCmd(RCC\_AHB1Periph\_GPIOE|RCC\_AHB1Periph\_GPIOF, ENABLE);

//

//PF9 PF10

GPIO\_InitTypeDef GPIO\_InitStruct;

GPIO\_InitStruct.GPIO\_Mode =GPIO\_Mode\_OUT;

GPIO\_InitStruct.GPIO\_OType =GPIO\_OType\_PP;

GPIO\_InitStruct.GPIO\_Pin =GPIO\_Pin\_9|GPIO\_Pin\_10;

GPIO\_InitStruct.GPIO\_PuPd =GPIO\_PuPd\_NOPULL;

GPIO\_InitStruct.GPIO\_Speed =GPIO\_Speed\_50MHz;

GPIO\_Init(GPIOF, &GPIO\_InitStruct);

//PE13 PE14

GPIO\_InitStruct.GPIO\_Pin =GPIO\_Pin\_13|GPIO\_Pin\_14;

GPIO\_Init(GPIOE, &GPIO\_InitStruct);

//ALL OFF

GPIO\_SetBits(GPIOF, GPIO\_Pin\_9|GPIO\_Pin\_10);

GPIO\_SetBits(GPIOE, GPIO\_Pin\_13|GPIO\_Pin\_14);

}

void led\_ctrl( int led\_num)

{

switch (led\_num)

{

case LED1:

{

GPIO\_ResetBits(GPIOF, GPIO\_Pin\_9);

GPIO\_SetBits(GPIOF, GPIO\_Pin\_10);

GPIO\_SetBits(GPIOE, GPIO\_Pin\_13|GPIO\_Pin\_14);

break;

}

case LED2:

{

GPIO\_SetBits(GPIOF, GPIO\_Pin\_9);

GPIO\_ResetBits(GPIOF, GPIO\_Pin\_10);

GPIO\_SetBits(GPIOE, GPIO\_Pin\_13|GPIO\_Pin\_14);

break;

}

case LED3:

{

GPIO\_SetBits(GPIOF, GPIO\_Pin\_9|GPIO\_Pin\_10);

GPIO\_ResetBits(GPIOE, GPIO\_Pin\_13);

GPIO\_SetBits(GPIOE, GPIO\_Pin\_14);

break;

}

case LED4:

{

GPIO\_SetBits(GPIOF, GPIO\_Pin\_9|GPIO\_Pin\_10);

GPIO\_SetBits(GPIOE, GPIO\_Pin\_13);

GPIO\_ResetBits(GPIOE, GPIO\_Pin\_14);

break;

}

default:

break;

}

}

void led\_loop(void)

{

for(int i = LED1;i<=LED4;i++)

{

led\_ctrl(i);

delay(0x111111);

}

}

void delay(int num)

{

while(num--);

}

cs\_key.c

#include "cs\_key.h"

//S1-PA0 S2-PE2 S3-PE3 S4-PE4

void key\_init(void)

{

//0.clk\_enable GPIO-AHB1 SYSCFG-APB2 EXIT-APB2

RCC\_AHB1PeriphClockCmd(RCC\_AHB1Periph\_GPIOA|RCC\_AHB1Periph\_GPIOE, ENABLE);

RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_SYSCFG|RCC\_APB2Periph\_EXTIT, ENABLE);

//1.GPIO-IN NOUPDN 50M

GPIO\_InitTypeDef GPIO\_InitStruct;

GPIO\_InitStruct.GPIO\_Mode =GPIO\_Mode\_IN;

GPIO\_InitStruct.GPIO\_Pin =GPIO\_Pin\_0;

GPIO\_InitStruct.GPIO\_PuPd =GPIO\_PuPd\_NOPULL;

GPIO\_InitStruct.GPIO\_Speed =GPIO\_Speed\_50MHz;

GPIO\_Init(GPIOA, &GPIO\_InitStruct);

GPIO\_InitStruct.GPIO\_Pin =GPIO\_Pin\_2|GPIO\_Pin\_3|GPIO\_Pin\_4;

GPIO\_Init(GPIOE, &GPIO\_InitStruct);

//2.SYSCFG-GPIOx-EXIT Linex

SYSCFG\_EXTILineConfig(EXTI\_PortSourceGPIOA, EXTI\_PinSource0);

SYSCFG\_EXTILineConfig(EXTI\_PortSourceGPIOE, EXTI\_PinSource2);

SYSCFG\_EXTILineConfig(EXTI\_PortSourceGPIOE, EXTI\_PinSource3);

SYSCFG\_EXTILineConfig(EXTI\_PortSourceGPIOE, EXTI\_PinSource4);

//3.EXIT

EXTI\_InitTypeDef EXTI\_InitStruct;

EXTI\_InitStruct.EXTI\_Line =EXTI\_Line0|EXTI\_Line2|EXTI\_Line3|EXTI\_Line4;

EXTI\_InitStruct.EXTI\_LineCmd =ENABLE;

EXTI\_InitStruct.EXTI\_Mode =EXTI\_Mode\_Interrupt;

EXTI\_InitStruct.EXTI\_Trigger =EXTI\_Trigger\_Falling;

EXTI\_Init(&EXTI\_InitStruct);

//4.NVIC

NVIC\_PriorityGroupConfig(NVIC\_PriorityGroup\_2);

NVIC\_InitTypeDef NVIC\_InitStruct;

NVIC\_InitStruct.NVIC\_IRQChannel =EXTI0\_IRQn;//PA0-EXTI0

NVIC\_InitStruct.NVIC\_IRQChannelCmd =ENABLE;

NVIC\_InitStruct.NVIC\_IRQChannelPreemptionPriority =2;//0-3

NVIC\_InitStruct.NVIC\_IRQChannelSubPriority =2;//0-3

NVIC\_Init(&NVIC\_InitStruct);

NVIC\_InitStruct.NVIC\_IRQChannel =EXTI2\_IRQn;//PE2-EXTI2

NVIC\_Init(&NVIC\_InitStruct);

NVIC\_InitStruct.NVIC\_IRQChannel =EXTI3\_IRQn;//PE3-EXTI3

NVIC\_Init(&NVIC\_InitStruct);

NVIC\_InitStruct.NVIC\_IRQChannel =EXTI4\_IRQn;//PE4-EXTI4

NVIC\_Init(&NVIC\_InitStruct);

}

void EXTI0\_IRQHandler (void)//s1-pa0-exti0---pf9-d1

{

if(SET==EXTI\_GetITStatus(EXTI\_Line0))

{

delay(10000);

if(Bit\_RESET == GPIO\_ReadInputDataBit(GPIOA, GPIO\_Pin\_0))

{

GPIO\_ToggleBits(GPIOF, GPIO\_Pin\_9);

}

EXTI\_ClearITPendingBit(EXTI\_Line0);

}

}

void EXTI2\_IRQHandler (void)//s2-pe2-exti2---buzzer(pf8-1)-d2

{

if(SET==EXTI\_GetITStatus(EXTI\_Line2))

{

delay(10000);

//DO

if(Bit\_RESET == GPIO\_ReadInputDataBit(GPIOE, GPIO\_Pin\_2))

{

GPIO\_SetBits(GPIOF, GPIO\_Pin\_8);

}

//CLEAR

EXTI\_ClearITPendingBit(EXTI\_Line2);

}

}

void EXTI3\_IRQHandler (void)//s3-pa2-exti3---buzzer(pf8-0)

{

if(SET==EXTI\_GetITStatus(EXTI\_Line3))

{

delay(10000);

if(Bit\_RESET == GPIO\_ReadInputDataBit(GPIOE, GPIO\_Pin\_3))

{

GPIO\_ResetBits(GPIOF, GPIO\_Pin\_8);

}

EXTI\_ClearITPendingBit(EXTI\_Line3);

}

}

void EXTI4\_IRQHandler (void)//s4-pa3-exti3---pf12-d4

{

if(SET==EXTI\_GetITStatus(EXTI\_Line4))

{

delay(10000);

if(Bit\_RESET == GPIO\_ReadInputDataBit(GPIOE, GPIO\_Pin\_4))

{

GPIO\_ToggleBits(GPIOE, GPIO\_Pin\_14);

}

EXTI\_ClearITPendingBit(EXTI\_Line4);

}

}

cs\_key.h

#ifndef \_\_KEY\_H\_\_

#define \_\_KEY\_H\_\_

#include "stm32f4xx.h"

void key\_init(void);

#endif

cs\_beep.c

#include "cs\_beep.h"

//S1-PA0 S2-PE2 S3-PE3 S4-PE4

void beep\_init(void)

{

//clk\_enable GPIOF---AHB1

RCC\_AHB1PeriphClockCmd(RCC\_AHB1Periph\_GPIOF, ENABLE);

//PF8

GPIO\_InitTypeDef GPIO\_InitStruct;

GPIO\_InitStruct.GPIO\_Mode =GPIO\_Mode\_OUT;

GPIO\_InitStruct.GPIO\_OType =GPIO\_OType\_PP;

GPIO\_InitStruct.GPIO\_Pin =GPIO\_Pin\_8;

GPIO\_InitStruct.GPIO\_PuPd =GPIO\_PuPd\_NOPULL;

GPIO\_InitStruct.GPIO\_Speed =GPIO\_Speed\_50MHz;

GPIO\_Init(GPIOF, &GPIO\_InitStruct);

}

cs\_beep.h

#ifndef \_\_BEEP\_H\_\_

#define \_\_BEEP\_H\_\_

#include "stm32f4xx.h"

void beep\_init(void);

#endif

# 时钟定时功能10-15%

systick.c

#include "systick.h"

//\*

void systick\_cfg(uint32\_t clk\_cnt)

{

//ctrl:bit0-1 enable bit1-1 int bit2-1 168MHz

rSysTickCtrl =0;

rSysTickCtrl |=(1<<0)|(1<<1)|(1<<2);

//reload val

rSysTickLoad =clk\_cnt;//N=167999 1ms

//cur val 0

rSysTickVal =0;

}

static uint32\_t systick\_num;

void SysTick\_Handler(void)

{

if (systick\_num != 0x00)

{

systick\_num--;

}

}

void Delay\_ms(uint32\_t num)//5000

{

systick\_cfg(167999);

systick\_num=num;

while(systick\_num != 0x00);

rSysTickCtrl &=~(1<<0);//bit0-0 disable

}

systick.h

#ifndef \_SYSTICK\_H\_

#define \_SYSTICK\_H\_

#include "stm32f4xx.h"

#define rSysTickCtrl  \*((volatile unsigned long\*)0xE000E010)

#define rSysTickLoad  \*((volatile unsigned long\*)0xE000E014)

#define rSysTickVal  \*((volatile unsigned long\*)0xE000E018)

void systick\_cfg(uint32\_t);

void Delay\_ms(uint32\_t);

void Delay\_us(uint32\_t);

#endif

cs\_beep.c

#include "cs\_beep.h"

//S1-PA0 S2-PE2 S3-PE3 S4-PE4

void beep\_init(void)

{

//clk\_enable GPIOF---AHB1

RCC\_AHB1PeriphClockCmd(RCC\_AHB1Periph\_GPIOF, ENABLE);

//PF8

GPIO\_InitTypeDef GPIO\_InitStruct;

GPIO\_InitStruct.GPIO\_Mode =GPIO\_Mode\_OUT;

GPIO\_InitStruct.GPIO\_OType =GPIO\_OType\_PP;

GPIO\_InitStruct.GPIO\_Pin =GPIO\_Pin\_8;

GPIO\_InitStruct.GPIO\_PuPd =GPIO\_PuPd\_NOPULL;

GPIO\_InitStruct.GPIO\_Speed =GPIO\_Speed\_50MHz;

GPIO\_Init(GPIOF, &GPIO\_InitStruct);

}

void beep\_test(void)

{

GPIO\_SetBits(GPIOF, GPIO\_Pin\_8);

Delay\_ms(5000);

GPIO\_ResetBits(GPIOF, GPIO\_Pin\_8);

}

cs\_key.c

#include "cs\_key.h"

uint8\_t status;

//S1-PA0 S2-PE2 S3-PE3 S4-PE4

void key\_init(void)

{

//0.clk\_enable GPIO-AHB1 SYSCFG-APB2 EXIT-APB2

RCC\_AHB1PeriphClockCmd(RCC\_AHB1Periph\_GPIOA|RCC\_AHB1Periph\_GPIOE, ENABLE);

RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_SYSCFG|RCC\_APB2Periph\_EXTIT, ENABLE);

//1.GPIO-IN NOUPDN 50M

GPIO\_InitTypeDef GPIO\_InitStruct;

GPIO\_InitStruct.GPIO\_Mode =GPIO\_Mode\_IN;

GPIO\_InitStruct.GPIO\_Pin =GPIO\_Pin\_0;

GPIO\_InitStruct.GPIO\_PuPd =GPIO\_PuPd\_NOPULL;

GPIO\_InitStruct.GPIO\_Speed =GPIO\_Speed\_50MHz;

GPIO\_Init(GPIOA, &GPIO\_InitStruct);

GPIO\_InitStruct.GPIO\_Pin =GPIO\_Pin\_2|GPIO\_Pin\_3|GPIO\_Pin\_4;

GPIO\_Init(GPIOE, &GPIO\_InitStruct);

//2.SYSCFG-GPIOx-EXIT Linex

SYSCFG\_EXTILineConfig(EXTI\_PortSourceGPIOA, EXTI\_PinSource0);

SYSCFG\_EXTILineConfig(EXTI\_PortSourceGPIOE, EXTI\_PinSource2);

SYSCFG\_EXTILineConfig(EXTI\_PortSourceGPIOE, EXTI\_PinSource3);

SYSCFG\_EXTILineConfig(EXTI\_PortSourceGPIOE, EXTI\_PinSource4);

//3.EXIT

EXTI\_InitTypeDef EXTI\_InitStruct;

EXTI\_InitStruct.EXTI\_Line =EXTI\_Line0|EXTI\_Line2|EXTI\_Line3|EXTI\_Line4;

EXTI\_InitStruct.EXTI\_LineCmd =ENABLE;

EXTI\_InitStruct.EXTI\_Mode =EXTI\_Mode\_Interrupt;

EXTI\_InitStruct.EXTI\_Trigger =EXTI\_Trigger\_Falling;

EXTI\_Init(&EXTI\_InitStruct);

//4.NVIC

NVIC\_PriorityGroupConfig(NVIC\_PriorityGroup\_2);

NVIC\_InitTypeDef NVIC\_InitStruct;

NVIC\_InitStruct.NVIC\_IRQChannel =EXTI0\_IRQn;//PA0-EXTI0

NVIC\_InitStruct.NVIC\_IRQChannelCmd =ENABLE;

NVIC\_InitStruct.NVIC\_IRQChannelPreemptionPriority =2;//0-3

NVIC\_InitStruct.NVIC\_IRQChannelSubPriority =2;//0-3

NVIC\_Init(&NVIC\_InitStruct);

NVIC\_InitStruct.NVIC\_IRQChannel =EXTI2\_IRQn;//PE2-EXTI2

NVIC\_Init(&NVIC\_InitStruct);

NVIC\_InitStruct.NVIC\_IRQChannel =EXTI3\_IRQn;//PE3-EXTI3

NVIC\_Init(&NVIC\_InitStruct);

NVIC\_InitStruct.NVIC\_IRQChannel =EXTI4\_IRQn;//PE4-EXTI4

NVIC\_Init(&NVIC\_InitStruct);

}

void EXTI0\_IRQHandler (void)//s1-pa0-exti0---pf9-d1

{

if(SET==EXTI\_GetITStatus(EXTI\_Line0))

{

delay(10000);

if(Bit\_RESET == GPIO\_ReadInputDataBit(GPIOA, GPIO\_Pin\_0))

{

GPIO\_ToggleBits(GPIOF, GPIO\_Pin\_9);

}

EXTI\_ClearITPendingBit(EXTI\_Line0);

}

}

void EXTI2\_IRQHandler (void)//s2-pe2-exti2---buzzer(pf8-1)-d2

{

if(SET==EXTI\_GetITStatus(EXTI\_Line2))

{

delay(10000);

//DO

if(Bit\_RESET == GPIO\_ReadInputDataBit(GPIOE, GPIO\_Pin\_2))

{

status=1;

}

//CLEAR

EXTI\_ClearITPendingBit(EXTI\_Line2);

}

}

void EXTI3\_IRQHandler (void)//s3-pa2-exti3---buzzer(pf8-0)

{

if(SET==EXTI\_GetITStatus(EXTI\_Line3))

{

delay(10000);

if(Bit\_RESET == GPIO\_ReadInputDataBit(GPIOE, GPIO\_Pin\_3))

{

GPIO\_ResetBits(GPIOF, GPIO\_Pin\_8);

}

EXTI\_ClearITPendingBit(EXTI\_Line3);

}

}

void EXTI4\_IRQHandler (void)//s4-pa3-exti3---pf12-d4

{

if(SET==EXTI\_GetITStatus(EXTI\_Line4))

{

delay(10000);

if(Bit\_RESET == GPIO\_ReadInputDataBit(GPIOE, GPIO\_Pin\_4))

{

GPIO\_ToggleBits(GPIOE, GPIO\_Pin\_14);

}

EXTI\_ClearITPendingBit(EXTI\_Line4);

}

}

main.c

#include "main.h"

extern uint8\_t status;

/\*\*

  \* @brief  Main program

  \* @param  None

  \* @retval None

  \*/

int main(void)

{

led\_init();

key\_init();

beep\_init();

  /\* Infinite loop \*/

  while (1)

  {

   if(1==status)

   {

   beep\_test();

status=0;

   }

  }

}

# 定时器复用功能20-25%

cs\_key.c

#include "cs\_key.h"

uint8\_t status;

//S1-PA0 S2-PE2 S3-PE3 S4-PE4

void key\_init(void)

{

//0.clk\_enable GPIO-AHB1 SYSCFG-APB2 EXIT-APB2

RCC\_AHB1PeriphClockCmd(RCC\_AHB1Periph\_GPIOA|RCC\_AHB1Periph\_GPIOE, ENABLE);

RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_SYSCFG|RCC\_APB2Periph\_EXTIT, ENABLE);

//1.GPIO-IN NOUPDN 50M

GPIO\_InitTypeDef GPIO\_InitStruct;

GPIO\_InitStruct.GPIO\_Mode =GPIO\_Mode\_IN;

GPIO\_InitStruct.GPIO\_Pin =GPIO\_Pin\_0;

GPIO\_InitStruct.GPIO\_PuPd =GPIO\_PuPd\_NOPULL;

GPIO\_InitStruct.GPIO\_Speed =GPIO\_Speed\_50MHz;

GPIO\_Init(GPIOA, &GPIO\_InitStruct);

GPIO\_InitStruct.GPIO\_Pin =GPIO\_Pin\_2|GPIO\_Pin\_3|GPIO\_Pin\_4;

GPIO\_Init(GPIOE, &GPIO\_InitStruct);

//2.SYSCFG-GPIOx-EXIT Linex

SYSCFG\_EXTILineConfig(EXTI\_PortSourceGPIOA, EXTI\_PinSource0);

SYSCFG\_EXTILineConfig(EXTI\_PortSourceGPIOE, EXTI\_PinSource2);

SYSCFG\_EXTILineConfig(EXTI\_PortSourceGPIOE, EXTI\_PinSource3);

SYSCFG\_EXTILineConfig(EXTI\_PortSourceGPIOE, EXTI\_PinSource4);

//3.EXIT

EXTI\_InitTypeDef EXTI\_InitStruct;

EXTI\_InitStruct.EXTI\_Line =EXTI\_Line0|EXTI\_Line2|EXTI\_Line3|EXTI\_Line4;

EXTI\_InitStruct.EXTI\_LineCmd =ENABLE;

EXTI\_InitStruct.EXTI\_Mode =EXTI\_Mode\_Interrupt;

EXTI\_InitStruct.EXTI\_Trigger =EXTI\_Trigger\_Falling;

EXTI\_Init(&EXTI\_InitStruct);

//4.NVIC

NVIC\_PriorityGroupConfig(NVIC\_PriorityGroup\_2);

NVIC\_InitTypeDef NVIC\_InitStruct;

NVIC\_InitStruct.NVIC\_IRQChannel =EXTI0\_IRQn;//PA0-EXTI0

NVIC\_InitStruct.NVIC\_IRQChannelCmd =ENABLE;

NVIC\_InitStruct.NVIC\_IRQChannelPreemptionPriority =2;//0-3

NVIC\_InitStruct.NVIC\_IRQChannelSubPriority =2;//0-3

NVIC\_Init(&NVIC\_InitStruct);

NVIC\_InitStruct.NVIC\_IRQChannel =EXTI2\_IRQn;//PE2-EXTI2

NVIC\_Init(&NVIC\_InitStruct);

NVIC\_InitStruct.NVIC\_IRQChannel =EXTI3\_IRQn;//PE3-EXTI3

NVIC\_Init(&NVIC\_InitStruct);

NVIC\_InitStruct.NVIC\_IRQChannel =EXTI4\_IRQn;//PE4-EXTI4

NVIC\_Init(&NVIC\_InitStruct);

}

void EXTI0\_IRQHandler (void)//s1-pa0-exti0---pf9-d1

{

if(SET==EXTI\_GetITStatus(EXTI\_Line0))

{

delay(10000);

if(Bit\_RESET == GPIO\_ReadInputDataBit(GPIOA, GPIO\_Pin\_0))

{

status=1;

}

EXTI\_ClearITPendingBit(EXTI\_Line0);

}

}

void EXTI2\_IRQHandler (void)//s2-pe2-exti2---buzzer(pf8-1)-d2

{

if(SET==EXTI\_GetITStatus(EXTI\_Line2))

{

delay(10000);

//DO

if(Bit\_RESET == GPIO\_ReadInputDataBit(GPIOE, GPIO\_Pin\_2))

{

status=2;

}

//CLEAR

EXTI\_ClearITPendingBit(EXTI\_Line2);

}

}

void EXTI3\_IRQHandler (void)//s3-pa2-exti3---buzzer(pf8-0)

{

if(SET==EXTI\_GetITStatus(EXTI\_Line3))

{

delay(10000);

if(Bit\_RESET == GPIO\_ReadInputDataBit(GPIOE, GPIO\_Pin\_3))

{

status=0;

}

EXTI\_ClearITPendingBit(EXTI\_Line3);

}

}

void EXTI4\_IRQHandler (void)//s4-pa3-exti3---pf12-d4

{

if(SET==EXTI\_GetITStatus(EXTI\_Line4))

{

delay(10000);

if(Bit\_RESET == GPIO\_ReadInputDataBit(GPIOE, GPIO\_Pin\_4))

{

GPIO\_ToggleBits(GPIOE, GPIO\_Pin\_14);

}

EXTI\_ClearITPendingBit(EXTI\_Line4);

}

}

cs\_beep.c

#include "cs\_beep.h"

//S1-PA0 S2-PE2 S3-PE3 S4-PE4

void beep\_init(int n)

{

//clk\_enable GPIOF---AHB1 TIM13-APB1

RCC\_AHB1PeriphClockCmd(RCC\_AHB1Periph\_GPIOF, ENABLE);

RCC\_APB1PeriphClockCmd(RCC\_APB1Periph\_TIM13, ENABLE);

//PF8-TIM13 MODE:AF OSPEED:50M NO UPDN

GPIO\_InitTypeDef GPIO\_InitStruct;

GPIO\_InitStruct.GPIO\_Mode =GPIO\_Mode\_AF;

GPIO\_InitStruct.GPIO\_Pin =GPIO\_Pin\_8;

GPIO\_InitStruct.GPIO\_PuPd =GPIO\_PuPd\_NOPULL;

GPIO\_InitStruct.GPIO\_Speed =GPIO\_Speed\_50MHz;

GPIO\_Init(GPIOF, &GPIO\_InitStruct);

GPIO\_PinAFConfig(GPIOF, GPIO\_PinSource8,  GPIO\_AF\_TIM13);

//2.TIM13-TIMEBASE

TIM\_TimeBaseInitTypeDef TIM\_TimeBaseInitStruct;

TIM\_TimeBaseInitStruct.TIM\_CounterMode =TIM\_CounterMode\_Up;

TIM\_TimeBaseInitStruct.TIM\_Period =n;//-CNT

TIM\_TimeBaseInitStruct.TIM\_Prescaler =8399;//84M--10K

TIM\_TimeBaseInit(TIM13, &TIM\_TimeBaseInitStruct);

//3.TIM13-OC1

TIM\_OCInitTypeDef TIM\_OCInitStruct;

TIM\_OCInitStruct.TIM\_OCMode =TIM\_OCMode\_PWM1;

TIM\_OCInitStruct.TIM\_OCPolarity =TIM\_OCPolarity\_High;

TIM\_OCInitStruct.TIM\_OutputState =TIM\_OutputState\_Enable;

TIM\_OCInitStruct.TIM\_Pulse =0.2\*n;//-CCR

TIM\_OC1Init(TIM13, &TIM\_OCInitStruct);

//TIM13-OC1-PRE-LOAD

TIM\_OC1PreloadConfig(TIM13, TIM\_OCPreload\_Enable);

//4.TIM13\_ENABLE

TIM\_ARRPreloadConfig(TIM13, ENABLE);

TIM\_Cmd(TIM13, ENABLE);

}

int note[] =

{

M6,M6,M6,M5,M6,M7,M6,M3,M5,M6,M7,M7,M7,M3,M6,

M7,M7,M7,M6,M7,M2,M1,M2,M3,M3,M3,M7,M3,

M3,M5,M6,M7,M6,M6,M3,M5,M6,M7,M2,M2,M1,M2,M3,M3,M3,M6,M3,M2,M1,

M5,M5,M5,M5,M3,M5,M5,M6,M1,M1,M2,L7,L6,

H1,H1,H1,H2,M7,M6,M7,M6,M3,M3,M3,H2,H2,H1,M7,M6,M7,M6,M3

};

float rhythm[] =

{

2,1,1,0.5,0.5,0.5,0.5,1,1,1,1,1,1,1,1,4,

2,1,1,0.5,0.5,1,1,1,1,1,1,1,2,4,

1,0.5,0.5,0.5,1,1,1,0.5,0.5,0.5,1,1,1,1,1,1,1,2,1,0.5,1,2,

1.25,1,1,1,1,1,0.5,0.5,1,1,0.5,0.5,2,

2,1,1,0.5,0.5,1,0.5,0.5,2,1,1,1,1,0.5,0.5,1,0.5,0.5,2

};

int note2[]=

{

M1,M1,M5,M5,M6,M6,M5,M4,M4,M3,M3,M2,M2,M1,

    M5,M5,M4,M4,M3,M3,M2,M5,M5,M4,M4,M3,M3,M2,

    M1,M1,M5,M5,M6,M6,M5,M4,M4,M3,M3,M2,M2,M1

};

int rhythm2[]=

{

1,1,1,1,1,1,2,1,1,1,1,1,1,2,

1,1,1,1,1,1,2,1,1,1,1,1,1,2,

1,1,1,1,1,1,2,1,1,1,1,1,1,2,

};

uint32\_t length = sizeof(note)/sizeof(note[0]);

extern uint8\_t status;

void music\_play(void)

{

for(int i=0;i<length;i++)

{

if(status==0)

{

break;

}

if(note[i]==ZERO)

{

RCC\_AHB1PeriphClockCmd(RCC\_AHB1Periph\_GPIOF, DISABLE);

RCC\_APB1PeriphClockCmd(RCC\_APB1Periph\_TIM13, DISABLE);

Delay\_ms((uint32\_t)(1000\*rhythm[i]));

}

else

{

beep\_init(note[i]);

Delay\_ms((uint32\_t)(400\*rhythm[i]));

}

}

RCC\_APB1PeriphClockCmd(RCC\_APB1Periph\_TIM13, DISABLE);

RCC\_AHB1PeriphClockCmd(RCC\_AHB1Periph\_GPIOF, DISABLE);

}

void music\_play2(void)

{

for(int i=0;i<length;i++)

{

if(status==0)

{

break;

}

if(note2[i]==ZERO)

{

RCC\_AHB1PeriphClockCmd(RCC\_AHB1Periph\_GPIOF, DISABLE);

RCC\_APB1PeriphClockCmd(RCC\_APB1Periph\_TIM13, DISABLE);

Delay\_ms((uint32\_t)(1000\*rhythm2[i]));

}

else

{

beep\_init(note2[i]);

Delay\_ms((uint32\_t)(400\*rhythm2[i]));

}

}

RCC\_AHB1PeriphClockCmd(RCC\_AHB1Periph\_GPIOF, DISABLE);

RCC\_APB1PeriphClockCmd(RCC\_APB1Periph\_TIM13, DISABLE);

}

void beep\_test(void)

{

GPIO\_SetBits(GPIOF, GPIO\_Pin\_8);

Delay\_ms(3000);

GPIO\_ResetBits(GPIOF, GPIO\_Pin\_8);

}

main.c

#include "main.h"

extern uint8\_t status;

/\*\*

  \* @brief  Main program

  \* @param  None

  \* @retval None

  \*/

int main(void)

{

key\_init();

  /\* Infinite loop \*/

  while (1)

  {

   if(1==status)

{

   if(status == 0) break;

music\_play();

status=0;

}

if(2==status)

{

   if(status == 0) break;

music\_play2();

status=0;

}

  }

}

# 串口通信20-25%

usart.c

#include "usart.h"

//usart1-tx

void usart\_init(void)

{

//0.CLK-ENABLE PA-AHB1 USART1-APB2

//USART1--APB2 USART3--APB1

RCC\_AHB1PeriphClockCmd(RCC\_AHB1Periph\_GPIOA | RCC\_AHB1Periph\_GPIOB, ENABLE);

RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_USART1, ENABLE);

RCC\_APB1PeriphClockCmd(RCC\_APB1Periph\_USART3, ENABLE);

//1.GPIO PA9-AF-USART1\_TX PA10-AF-USART1\_RX

//GPIO PB10-AF-USART3\_TX PB11-AF-USART3\_RX

GPIO\_InitTypeDef GPIO\_InitStruct;

GPIO\_InitStruct.GPIO\_Mode =GPIO\_Mode\_AF;

GPIO\_InitStruct.GPIO\_Pin =GPIO\_Pin\_9 | GPIO\_Pin\_10;

GPIO\_InitStruct.GPIO\_PuPd =GPIO\_PuPd\_NOPULL;

GPIO\_InitStruct.GPIO\_Speed =GPIO\_Speed\_50MHz;

GPIO\_Init(GPIOA, &GPIO\_InitStruct);

GPIO\_InitStruct.GPIO\_Pin =GPIO\_Pin\_10 | GPIO\_Pin\_11;

GPIO\_Init(GPIOB, &GPIO\_InitStruct);

//PA9,10-AF-USART1

GPIO\_PinAFConfig(GPIOA, GPIO\_PinSource9, GPIO\_AF\_USART1);

GPIO\_PinAFConfig(GPIOA, GPIO\_PinSource10, GPIO\_AF\_USART1);

GPIO\_PinAFConfig(GPIOB, GPIO\_PinSource10, GPIO\_AF\_USART3);

GPIO\_PinAFConfig(GPIOB, GPIO\_PinSource11, GPIO\_AF\_USART3);

//2.USART USART3

USART\_InitTypeDef USART\_InitStruct;

USART\_InitStruct.USART\_BaudRate =9600;

USART\_InitStruct.USART\_HardwareFlowControl =USART\_HardwareFlowControl\_None;

USART\_InitStruct.USART\_Mode =USART\_Mode\_Rx|USART\_Mode\_Tx;

USART\_InitStruct.USART\_Parity =USART\_Parity\_No;

USART\_InitStruct.USART\_StopBits =USART\_StopBits\_1;

USART\_InitStruct.USART\_WordLength =USART\_WordLength\_8b;

USART\_Init(USART1,&USART\_InitStruct);

USART\_Init(USART3,&USART\_InitStruct);

//3.USART-IT

USART\_ITConfig(USART1, USART\_IT\_RXNE, ENABLE);

USART\_ITConfig(USART3, USART\_IT\_RXNE, ENABLE);

//4.NVIC

NVIC\_PriorityGroupConfig(NVIC\_PriorityGroup\_2);//2 bits for pre-emption priority 0-3

//2 bits for subpriority 0-3

NVIC\_InitTypeDef NVIC\_InitStruct;

NVIC\_InitStruct.NVIC\_IRQChannel =USART1\_IRQn;

NVIC\_InitStruct.NVIC\_IRQChannel  =USART3\_IRQn;

NVIC\_InitStruct.NVIC\_IRQChannelCmd =ENABLE;

NVIC\_InitStruct.NVIC\_IRQChannelPreemptionPriority =2;//0-3

NVIC\_InitStruct.NVIC\_IRQChannelSubPriority =2;//0-3

NVIC\_Init(&NVIC\_InitStruct);

//5.USART-CMD

USART\_Cmd(USART1, ENABLE);

USART\_Cmd(USART3, ENABLE);

}

int fputc(int c, FILE \*stream)

{

 USART1->SR;

 USART\_SendData(USART1, c & 0xff);

 while( USART\_GetFlagStatus(USART1, USART\_FLAG\_TC) == RESET);

 return(c);

}

uint8\_t revc\_byte=0;

void USART1\_IRQHandler(void)

{

//

if(SET==USART\_GetITStatus(USART1, USART\_IT\_RXNE))

{

revc\_byte=USART\_ReceiveData(USART1)&(uint8\_t)0xff;

USART\_ClearITPendingBit(USART1, USART\_IT\_RXNE);

}

}

void USART3\_IRQHandler(void)

{

//

if(SET==USART\_GetITStatus(USART3, USART\_IT\_RXNE))

{

revc\_byte=USART\_ReceiveData(USART3)&(uint8\_t)0xff;

USART\_ClearITPendingBit(USART3, USART\_IT\_RXNE);

}

}

track.c

#include "track.h"

void Track\_Init(void)

{

//0.clk enable

RCC\_AHB1PeriphClockCmd(RCC\_AHB1Periph\_GPIOB, ENABLE);

//1.GPIO

GPIO\_InitTypeDef GPIO\_InitStruct;

GPIO\_InitStruct.GPIO\_Mode =GPIO\_Mode\_IN;

GPIO\_InitStruct.GPIO\_Pin =GPIO\_Pin\_1| GPIO\_Pin\_2;

GPIO\_InitStruct.GPIO\_PuPd =GPIO\_PuPd\_NOPULL;

GPIO\_InitStruct.GPIO\_Speed =GPIO\_Speed\_50MHz;

GPIO\_Init(GPIOB, &GPIO\_InitStruct);

}

void Track\_Auto(void)

{

if(left == WHITE && right == WHITE)

{Go\_Up();}

if(left == BLACK && right == WHITE)

{Go\_Left();}

if(left == WHITE && right == BLACK)

{Go\_Right();}

if(left == BLACK && right == BLACK)

{Stop();}

}

motor.c

#include "motor.h"

uint8\_t value;

void Motor\_Init(void)

{

//0.clk enable

RCC\_AHB1PeriphClockCmd(RCC\_AHB1Periph\_GPIOC, ENABLE);

GPIO\_InitTypeDef GPIO\_InitStruct;

GPIO\_InitStruct.GPIO\_Mode =GPIO\_Mode\_OUT;

GPIO\_InitStruct.GPIO\_OType =GPIO\_OType\_PP;

GPIO\_InitStruct.GPIO\_Pin =GPIO\_Pin\_6;

GPIO\_InitStruct.GPIO\_PuPd =GPIO\_PuPd\_NOPULL;

GPIO\_InitStruct.GPIO\_Speed =GPIO\_Speed\_50MHz;

GPIO\_Init(GPIOC, &GPIO\_InitStruct);

GPIO\_InitStruct.GPIO\_Pin =GPIO\_Pin\_7;

GPIO\_Init(GPIOC, &GPIO\_InitStruct);

GPIO\_InitStruct.GPIO\_Pin =GPIO\_Pin\_8;

GPIO\_Init(GPIOC, &GPIO\_InitStruct);

GPIO\_InitStruct.GPIO\_Pin =GPIO\_Pin\_9;

GPIO\_Init(GPIOC, &GPIO\_InitStruct);

}

static void Set\_La(uint8\_t value)

{

if(value==1)

{

GPIO\_SetBits(GPIOC,GPIO\_Pin\_6);

}

else

{

GPIO\_ResetBits(GPIOC,GPIO\_Pin\_6);

}

}

static void Set\_Lb(uint8\_t value)

{

if(value==1)

{

GPIO\_SetBits(GPIOC,GPIO\_Pin\_7);

}

else

{

GPIO\_ResetBits(GPIOC,GPIO\_Pin\_7);

}

}

static void Set\_Ra(uint8\_t value)

{

if(value==1)

{

GPIO\_SetBits(GPIOC,GPIO\_Pin\_8);

}

else

{

GPIO\_ResetBits(GPIOC,GPIO\_Pin\_8);

}

}

static void Set\_Rb(uint8\_t value)

{

if(value==1)

{

GPIO\_SetBits(GPIOC,GPIO\_Pin\_9);

}

else

{

GPIO\_ResetBits(GPIOC,GPIO\_Pin\_9);

}

}

void Go\_Up(void)//前进

{

Set\_La(1);//左轮正向转动

Set\_Lb(0);

Set\_Ra(1);//右轮正向转动

Set\_Rb(0);

}

void Go\_Back(void)//后退

{

Set\_La(0);//左轮反向转动

Set\_Lb(1);

Set\_Ra(0);//右轮反向转动

Set\_Rb(1);

}

void Stop(void)

{

Set\_La(0);

Set\_Lb(0);

Set\_Ra(0);

Set\_Rb(0);

}

int m = 80;

void Go\_Right(void)

{

Set\_La(1);//左轮正向转动

Set\_Lb(0);

Set\_Ra(0);//右轮以0.8倍速反向转动

Set\_Rb(1);

Delay\_ms(m);

Set\_Rb(0);

Delay\_ms(100-m);

}

void Go\_Left(void)

{

Set\_Ra(1);//右轮正向转动

Set\_Rb(0);

Set\_La(0);//左轮以0.8倍速反向转动

Set\_Lb(1);

Delay\_ms(m);

Set\_Lb(0);

Delay\_ms(100-m);

}

# 一些其他的补充知识

作业1

STM32F407ZETx

CPU工作原理

CPU = ALU + Control Unit

Register(寄存器)：CPU内部的存储区域，暂时存放参与运算的数据和运算结果。

寄存器:锁存器/触发器实现，只包含存储电路的时序逻辑电路。

为什么需要“堆栈”？为了支持过程调用(函数)。

“现场保护”

函数的具体功能的代码

“现场恢复”

Cortex M4有两个堆栈，双堆栈

MSP 主堆栈指针

PSP 进程堆栈指针

为什么需要双堆栈呢？

为了支持操作系统。把操作系统用的堆栈和用户进程用的堆栈分开。

R14(LR): Linked Register 链接寄存器

在执行过程调用的指令的时候，我们需要保存该指令的下一条指针的地址，

因为这个地址，就是需要返回的地址。

有一个专门的寄存器，用来保存过程调用调用的返回地址。->LR(R14)

MOV R0, #3

MOV R1, #4

BL sum ;

// BL:把下一条指令的地址(如下的: (A))存放在LR中

// 跳转是通过把：要跳到的那个地址，直接赋值给PC

// sum -> PC

(A) ADD R0, R1,

sum:

ADD R0,R0,R1

MOV PC, LR ; -> return 函数返回，过程返回。

R15(PC): Program Counter 程序计数器。 保存下一条要执行的指令的地址。

PC会在取指后，会自动增加指令所占的bits位数。

在ARM Cortex M4, PC + 4

在有“指令流水线”情况下，PC的值会有所不同

作业2

应用状态寄存器 APSR: 计算结果的标志

N Z C V Q

N Z C V Q

我们每一条指令的执行都可以影响这些状态标志位。

N: 负数标志。

Z : Zero 。零标志。结果所有bit位都为0,则xPSR.Z == 1

是0不是0

C: Carry 借位或进位标志。bit29

进位： 在做加法运算时，产生了进位。则C == 1,否则 C == 0

借位： 在做减法运算时，没产生借位。则C == 1,否则 C == 0

ADC, ADD, CMN 加法。如果产生了进位，则C == 1,否则 C == 0

SBC, SUB, CMP 减法。如果生生了借位，则C == 0,否则 C == 1

V: oVerflow 溢出标志。bit28

反映有符号数做加减运算所得结果是否溢出，如果运算结果超过当前

运算位数所能表示的范围，则溢出 xPSR.V = 1, 否则为0. 在有符号的运算中，进位(借位,C)与溢出是两个完全不同的概念。

Q: 饱和标志

饱和计算： 通过将数据强制置为最大(或最小)允许值，减小了

数据畸变，当然畸变仍然存在，不过若数据没有超过最大

允许范围太多，就不会有太大的问题。

ICI : Interruptible-Continuable Instrument 可中断－可继续指令位

Cortex M4工作模式

"模式"： 不同环境，不同的角色

ARM cortex M4有两种工作模式:

Thread Mode: 线程模式

Handler Mode: 处理模式(异常中断模式)

异常/中断 是什么？ 打断CPU指令执行顺序的事件，称为中断。

为什么要支持两种模式呢？ 为什么不只用一种模式呢？ Thread Mode

如果只用一种模式，thread mode,为了响应一些外部事件(比如说，用户是否

按下某个按键？):

轮询：轮流询问。 通过轮询，CPU也可能 响应外部事件，但是

轮询天生就有缺陷:

(1) 浪费CPU

(2) 占用总线， Bus is always busy.

(3) 轮询有一个时间差，轮询的时间间隔。不及时！！！

在CPU内部设计一个 “中断模式”：

为了提高效率和响应速度。

两种模式之间是怎么切换的呢？ 重要。如图thread\_Mode与Handler\_Mode之间的切换

Handler Mode :

中断模式，当一些比较重要的事件，产生时，CPU中止正在做的

事情，切换到Handler Mode下去执行，此时 “特权等级”

中断处理完成后，再返回到断点处，继续Thread Mode运行。

Thread Mode:

线程模式。

特权等级 : 可以跑一些如OS的代码

非特权等级: 可以跑一些如 "用户态"的代码

特权等级 -> 非特权等级

但是:

非特权等级 不可以 切换到 特权事件，除非产生“中断”