# 定时器捕获电机控制与检测系统设计

## 硬件电路连接

【定时器】

定时器捕获输入PA8[CN\_9\_8]

【五线四相步进电机驱动】

右下角接5V电压 GND接地

P1.7→GPIOD0[CN\_7\_9] P1.6→GPIOD1[CN\_7\_10]

P2.2→GPIOD2[CN\_7\_4] P2.1→GPIOD3[CN\_7\_11]

【外部中断按键13控制转速与转动方向】

外部中断按键->PC13

【霍尔编码器反馈信号】

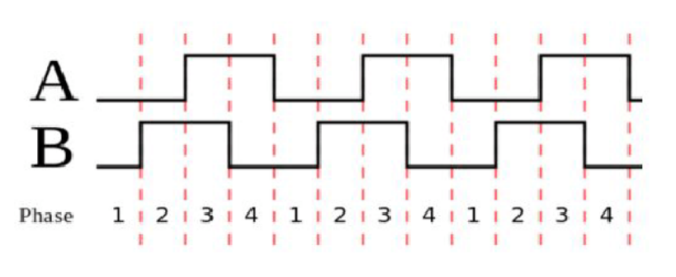
反馈信号A相→PC11[CN\_7\_2]作为通用输入

反馈信号B相→PA8[CN\_9\_8]定时器捕获输入

电源控制线悬空

## 四相编码器

四相编码器AB相信号，如下图：



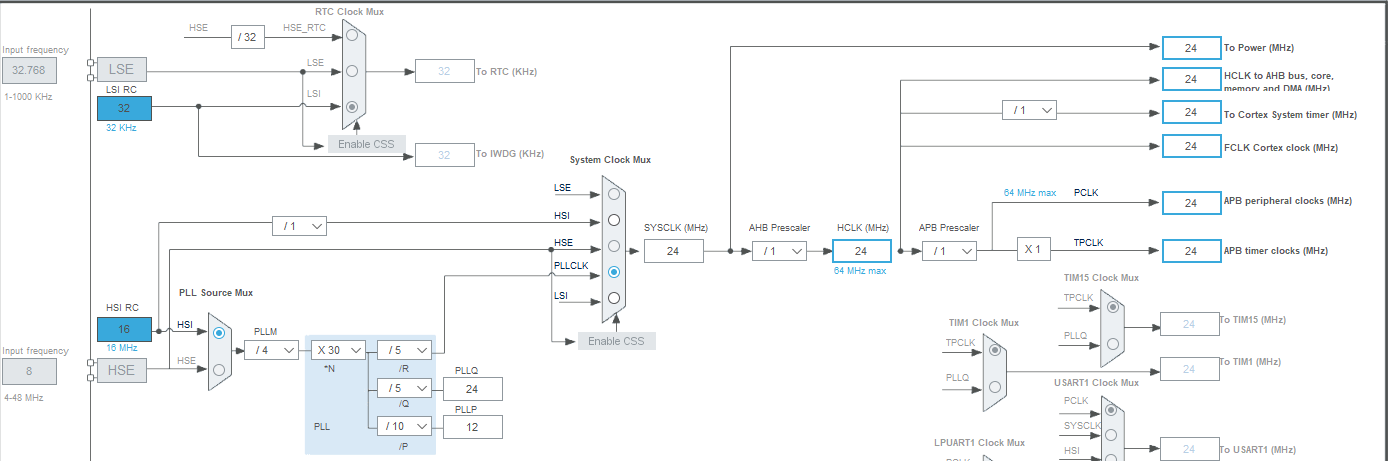
外部中断12连接在B相信号线，且其为下降沿触发。PC11连接在A相信号线，且为通用输入模式。①当步进电机向前旋转（上图中信号方向“→”），定时器捕获通道触发即B相信号线的下降沿时，A相信号线为高电平，此时检测到向前转。②当步进电机向后旋转（上图中信号方向“⬅”），定时器捕获通道触发即B相信号线的下降沿时，A相信号线为低电平，此时检测到向后转。

## 创建工程

将RCC如下选项设置为“FALSE”

C:\Users\李天凌\AppData\Local\Temp\WeChat Files\8dc9339ed9318ecae6f39c23a562fa7.png

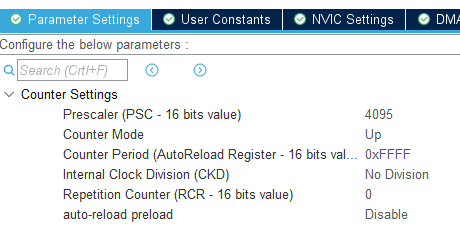
按照下图所示设置系统时钟



定时器通道1工作模式设置为“直接输入捕获模式”

C:\Users\李天凌\AppData\Local\Temp\WeChat Files\d39c7e7b52dcd4b3437542b8a796542.png

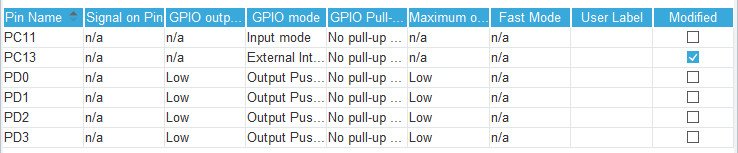
定时器1预分频值4095，计数周期0xFFFF



将PD0到PD3设置为通用输出模式用以驱动步进电机

将PC11设置为通用输入模式，用以获取思想编码器另一根线的状态

将PC13设置为下降沿触发的外部中断，用以改变电机转动方向和转速



在NVIC使能定时器1捕获中断4-15外部中断



## 添加设计代码

**变量定义与函数声明**

#include "stdio.h"

unsigned char Step\_table[]={0x08,0x04,0x02,0x01};

int time=4;

int mode=0;

char fangxiang[5];

uint32\_t uwIC2Value1 = 0; //触发捕获的第一个计数值

uint32\_t uwIC2Value2 = 0; //触发捕获的第二个计数值

uint32\_t uwDiffCapture = 0; //触发捕获的时间差

uint16\_t uhCaptureIndex = 0; //捕获指标：标明第一次捕获还是第二次捕获

double uwFrequency = 0; //频率值(转速)

**中断标志位处理函数**

ITStatus EXTI\_GetITStatus(uint32\_t EXTI\_Line)

{

ITStatus bitstatus = RESET; //初始位状态 0

uint32\_t enablestatus = 0; //初始使能状态 0

/\* 检查参数 \*/

assert\_param(IS\_GET\_EXTI\_LINE(EXTI\_Line));

enablestatus = EXTI->IMR1 & EXTI\_Line;

if (((EXTI->FPR1 & EXTI\_Line) != (uint32\_t)RESET) && (enablestatus != (uint32\_t)RESET))

bitstatus = SET;

else

bitstatus = RESET;

return bitstatus;

}

void EXTI\_ClearITPendingBit(uint32\_t EXTI\_Line)

{

/\* 检查参数 \*/

assert\_param(IS\_EXTI\_LINE(EXTI\_Line));

EXTI->FPR1 = EXTI\_Line;

}

**主函数非循环部分打开定时器1通道1**

if(HAL\_TIM\_IC\_Start\_IT(&htim1, TIM\_CHANNEL\_1) != HAL\_OK)

Error\_Handler(); //打开失败

**主函数循环部分依据电机工作模式与转速驱动电机**

while (1)

{

if(mode==0)

for(int i=0;i<4;i++)

{

GPIOD->ODR=Step\_table[i];

HAL\_Delay(time);

}

else

for(int i=0;i<4;i++)

{

GPIOD->ODR=Step\_table[3-i];

HAL\_Delay(time);

}

}

**编写13号中断服务程序与定时器1捕获中断回调函数**

void EXTI4\_15\_IRQHandler(void)

{

if (EXTI\_GetITStatus(0x2000) != 0x00) // 0x2000==EXTI\_PIN\_13 换方向换速度

{

EXTI\_ClearITPendingBit(0x2000); //Clear interrupt flag bit

if((mode==0)&&(time>=14))

mode++;

else if((mode==1)&&(time<=6))

mode=0;

else if(mode==1)

time=time-2;

else if(mode==0)

time=time+2;

}

}

void HAL\_TIM\_IC\_CaptureCallback(TIM\_HandleTypeDef \*htim)

{

if (htim->Channel == HAL\_TIM\_ACTIVE\_CHANNEL\_1)

{

if(uhCaptureIndex == 0) /\* 第一次捕获 \*/

{

uwIC2Value1 = HAL\_TIM\_ReadCapturedValue(htim, TIM\_CHANNEL\_1); /\* 获取计数值 \*/

uhCaptureIndex = 1;

}

else if(uhCaptureIndex == 1) /\* 第二次捕获 \*/

{

uwIC2Value2 = HAL\_TIM\_ReadCapturedValue(htim, TIM\_CHANNEL\_1); /\* 获取计数值 \*/

if (uwIC2Value2 > uwIC2Value1) /\* 计算两次捕获计数器值之差 \*/

{

uwDiffCapture = (uwIC2Value2 - uwIC2Value1);

}

else if (uwIC2Value2 < uwIC2Value1)

{

/\*0xFFFF 是最大的TIM1\_CCRx寄存器值，转了一圈回来了Value2+max+1 - Value1 \*/

uwDiffCapture = ((0xFFFF - uwIC2Value1) + uwIC2Value2) + 1;

}

else /\*如果捕获值相等,就代表超过了最高的频率测量值（计数器还没反应过来已经捕获两次了）\*/

Error\_Handler();

/\* 频率计算: 在这个例子中，定时器1时钟源是APB1Clk \*/

uwFrequency = HAL\_RCC\_GetPCLK1Freq() \* 1.000 / uwDiffCapture / 4096; //4096是定时器1分频值

uhCaptureIndex = 0;

}

if(HAL\_GPIO\_ReadPin(GPIOC,GPIO\_PIN\_11)==1)

strcpy(fangxiang, "Qian");

else

strcpy(fangxiang, "Hou ");

}

}

**在stm32g0xx\_it.c文件下注释原有13号中断入口**

/\*void EXTI4\_15\_IRQHandler(void)

{

HAL\_GPIO\_EXTI\_IRQHandler(GPIO\_PIN\_13);

} \*/

## 检测性能分析

系统时钟为24MHz，定时器分频系数4096，故最大检测频率：24MHz/4096=5859.375Hz

定时器1最大计数值为0xFFFF，计数周期为2^16=65536，故最小检测频率为24MHz/4096/65536=0.0894Hz

## 结果展示

进入Debug界面将需要观测的值添加到“watch”窗口。利用外部中断按键“USER”切换步进电机转动方向与转速。

