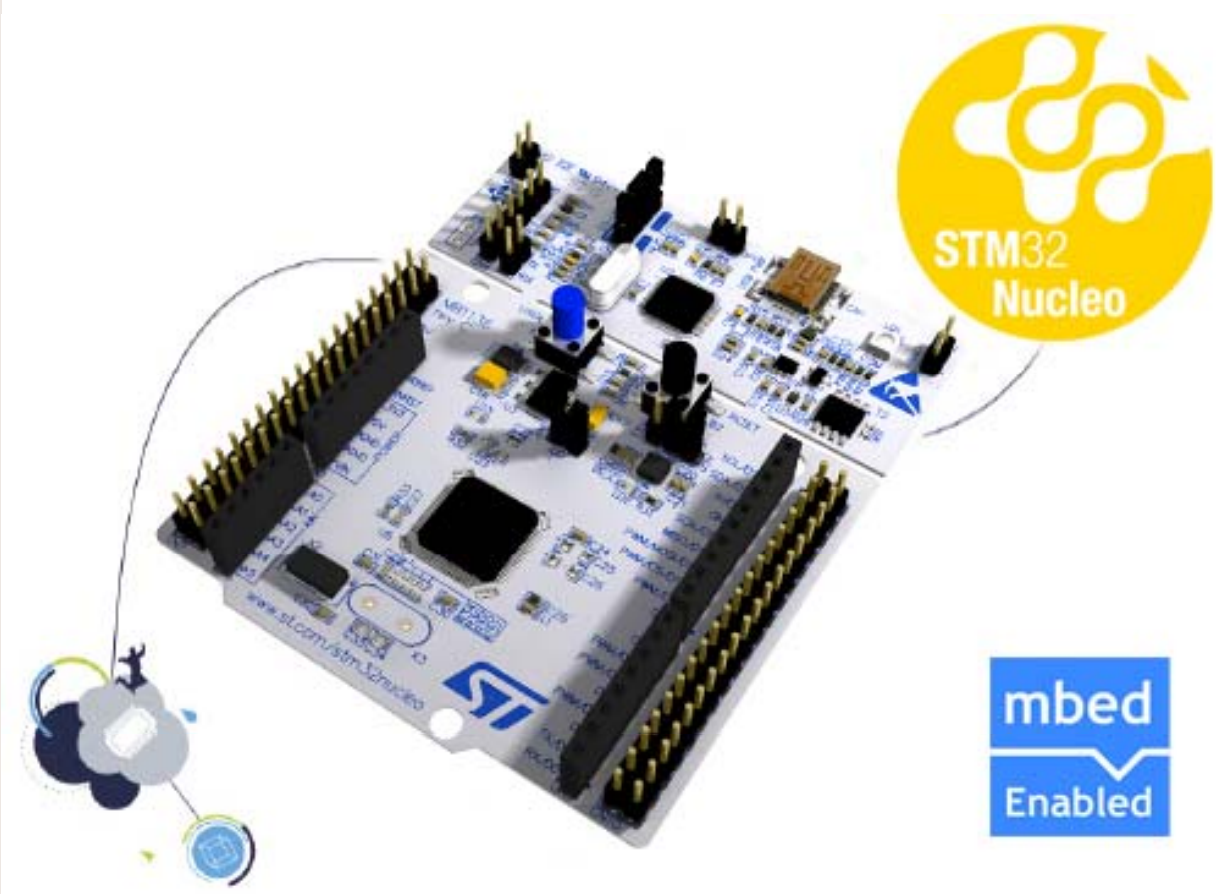
深入浅出STM32-NUCLEO开发平台

--基于STM32-NUCLEO-072RB开发板实例讲解



摩尔吧 MOORE 8

本章节内容

- 中级教程：6路互补PWM输出
- 中级教程：BLDC基础知识
- 中级教程：6步换向法输出PWM驱动BLDC(带HALL)

PWM和STM32-072RB定时器介绍

一、 PWM和STM32-072RB定时器介绍

脉宽调制（PWM:(Pulse Width Modulation）是利用微处理器的数字输出来对模拟电路进行控制的一种非常有效的技术，广泛应用在从测量、通信到功率控制与变换的许多领域中。简易理解，就是对输出脉冲高低电平宽度的调制！

STM32-072RB MCU 有**12**个定时器，其中有高级定时器、通用定时器和基本定时器及系统定时器。在这其中，有一个高级定时器TIMER1可以产生**7路PWM**(CH1/CH2/CH3/CH4/CH1N/CH2N/CH3N), 通用定时器TIMER2/TIMER3可以分别产生**4路PWM** (CH1/CH2/CH3/CH4),通用定时器TIMER15能产生**3路PWM**(CH1/CH2/CH1N),通用定时器TIMER14/TIMER16/TIMER17各能产生**1路PWM**(CH1),这样，总共能产生 **21路PWM。**

高级定时器TIMER1专为电机控制而生，可以产生3对6路互补PWM输出

要利用STM32的定时器来产生PWM，需要用到定时器相关的寄存器。

此外，高级定时器和通用定时器依附于不同的时钟总线

高级定时器 APB2时钟总线　通用定时器　APB1时钟总线

PWM和STM32-072RB定时器介绍

TIMx_CCMRx 寄存器：选择PWM模式

TIMx_ARR 寄存器　：PWM的周期

TIMx_CCRx 寄存器　：PWM占空比

二、STM32 定时器部分库函数介绍

1、定时器结构体成员变量

产生PWM，要用到三个结构体，分别是

TIM_TimeBaseInitTypeDef　基本定时结构体

TIM_OCInitTypeDef　输出比较结构体

TIM_BDTRInitTypeDef　刹车和死区配置结构体

对于三个库函数来实现将结构体值赋值到STM32内部寄存器中

TIM_TimeBaseInit (); TIM_OCInit () TIM_BDTRConfig()

```
typedef struct
{
  uint16_t TIM_Prescaler;         /*!< S                value used to divide the TIM clock.
                                   T                number between 0x0000 and 0xFFFF */

  uint16_t TIM_CounterMode;       /*!< S                ode.
                                   T                value of @ref TIM_Counter_Mode */

  uint32_t TIM_Period;            /*!< S                lue to be loaded into the active
                                   A                the next update event.
                                   This parameter must be a number between 0x0000 and 0xFFFF.  */

  uint16_t TIM_ClockDivision;     /*!< S                ision.
                                       Th               value of @ref TIM_Clock_Division_CKD */

  uint8_t TIM_RepetitionCounter;  /*!< S                n counter value. Each time the RCR downc
                                   R                event is generated and counting restart
                                   from the non-value (N).
                                   This means in PWM mode that (N+1) corresponds to:
                                      - the number of PWM periods in edge-aligned mode
                                      - the number of half PWM period in center-aligned mode
                                   This parameter must be a number between 0x00 and 0xFF.
                                   @note This parameter is valid only for TIM1. */
} TIM_TimeBaseInitTypeDef;
```

时钟预分频

计数模式

定时周期

定时分割

重复计数值

计数模式：向上计数，向下计数和中间对齐模式

```
typedef struct
{
  uint16_t TIM_OCMode;        /*!< mode.
                                   be a value of @ref TIM_Output_Compare_and_PWM_mo

  uint16_t TIM_OutputState;   /*!< state.
                                   @ref TIM_Output_Compare_state */

  uint16_t TIM_OutputNState;  /*!< Output Compare state.
                                   @ref TIM_Output_Compare_N_state
                                   @note This parameter is valid only for TIM1. */

  uint32_t TIM_Pulse;         /*!< loaded into the Capture Compare Reg
                                   between 0x0000 and 0xFFFF ( or 0xFFF
                                   for TIM2) */

  uint16_t TIM_OCPolarity;    /*!<
                                   of @ref TIM_Output_Compare_Polarity *

  uint16_t TIM_OCNPolarity;   /*!< put polarity.
                                   of @ref TIM_Output_Compare_N_Polarity
                                   @note This parameter is valid only for TIM1. */

  uint16_t TIM_OCIdleState;   /*!< state during Idle state.
                                   ef TIM_Output_Compare_Idle_State
                                   @note This parameter is valid only for TIM1. */

  uint16_t TIM_OCNIdleState;  /*!< tate during Idle state.
                                   TIM_Output_Compare_N_Idle_Sta
                                   @note This parameter is valid only for TIM1. */
} TIM_OCInitTypeDef;
```

输出模式

比较端引脚输出状态

互补端引脚输出状态

设置占空比数值

比较端引脚输出极性

互补端引脚输出极性

比较端空闲状态下引脚状态

互补端空闲状态下引脚状态

```
typedef struct
{
  uint16_t TIM_OSSRState;      /*!< ion used in Run mode.
                                    of @ref TIM_OSSR_Off_State_Selecti
  uint16_t TIM_OSSIState;      /*!< n Idle state.
                                    of @ref TIM_OSSI_Off_State_Selecti
  uint16_t TIM_LOCKLevel;      /*!< eters.
                                    This parameter can be a value of @ref TIM_Lock_level */
  uint16_t TIM_DeadTime;       /*!< en the switching-off and the
                                    This parameter can be a number between 0x00 and 0xFF  */
  uint16_t TIM_Break;          /*!< k input is enabled or not.
                                    This parameter can be a value of @ref TIM_Break_Input_enable_dis
  uint16_t TIM_BreakPolarity;  /*!< pin polarity.
                                    of @ref TIM_Break_Polarity */
  uint16_t TIM_AutomaticOutput; /*!< matic Output feature is enabled or
                                    of @ref TIM_AOE_Bit_Set_Reset */
} TIM_BDTRInitTypeDef;
```

运行模式下状态

空闲模式下状态

锁定设置

设置死区时间

刹车功能选择

刹车输入极性配置

自动输出选择

STM32 定时器部分库函数介绍

TIM_OCxPreloadConfig ();
TIM_ARRPreloadConfig ();
TIM_Cmd();
TIM_CtrlPWMOutputs();
TIM _ITConfig ();
TIM_ARRPreloadConfig ();
TIM_ForcedOC1Config ();
TIM_GetITStatus ()
TIM_ClearITPendingBit()
TIM_CCPreloadControl();
TIM_SelectOCxM();
TIM_CCxCmd();
TIM_CCxNCmd();
TIM_GenerateEvent();

STM32-NUCLEO-072RB硬件PWM输出

三、STM32-NUCLEO-072RB开发板6路PWM引脚配置

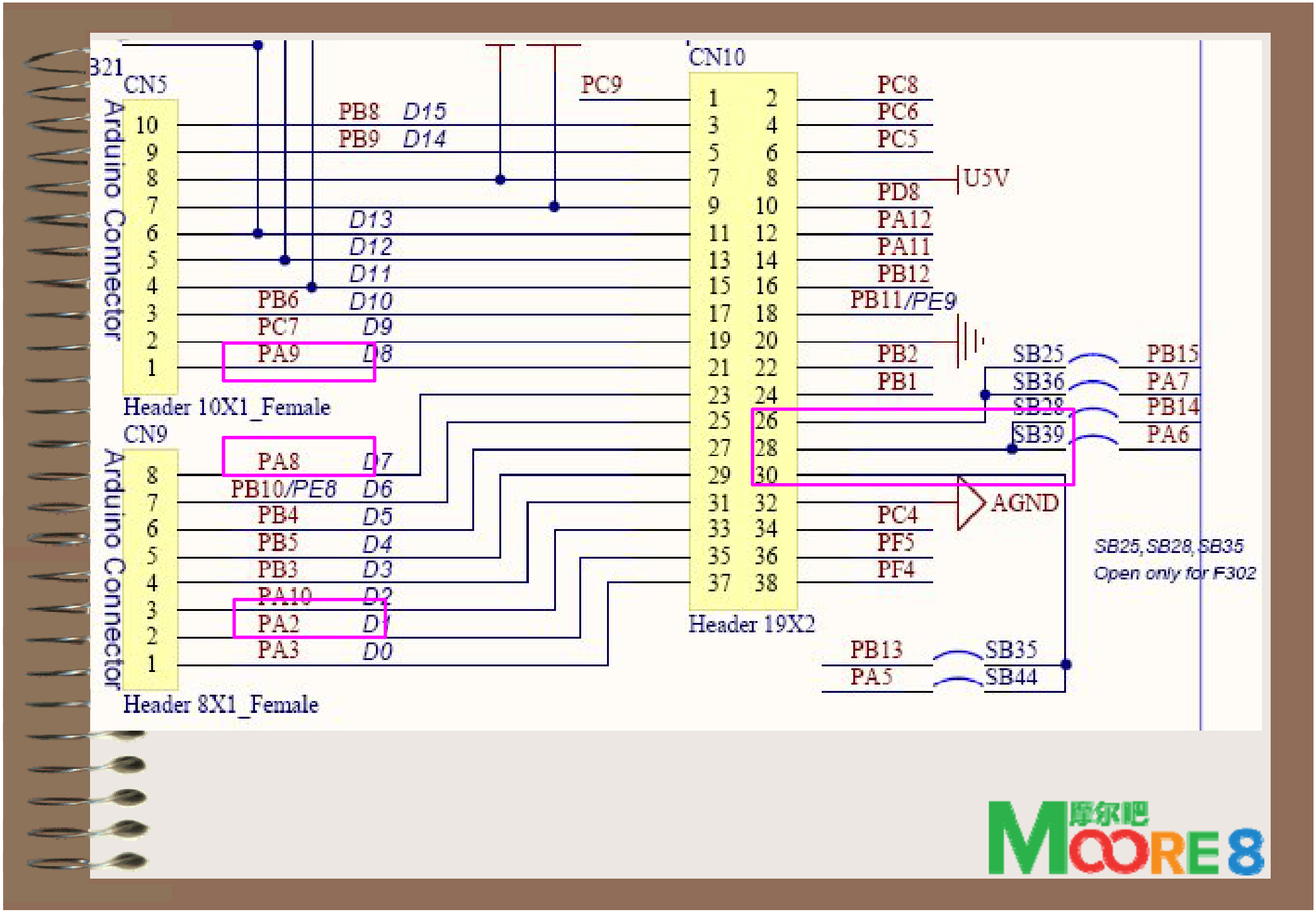
采用了TIMER1高级定时器的3对6路引脚

（**CH1/CH1N,CH2/CH2N,CH3/CH3N**）

PA8/PA9/PA10 /PB13/PB14/PB15

Pin name	AF0	AF1	AF2
PA0		USART2_CTS	TIM2_CH1_ETR
PA1	EVENTOUT	USART2_RTS	TIM2_CH2
PA2	TIM15_CH1	USART2_TX	TIM2_CH3
PA3	TIM15_CH2	USART2_RX	TIM2_CH4
PA4	SPI1_NSS, I2S1_WS	USART2_CK	
PA5	SPI1_SCK, I2S1_CK	CEC	TIM2_CH1_ETR
PA6	SPI1_MISO, I2S1_MCK	TIM3_CH1	TIM1_BKIN
PA7	SPI1_MOSI, I2S1_SD	TIM3_CH2	TIM1_CH1N
PA8	MCO	USART1_CK	TIM1_CH1
PA9	TIM15_BKIN	USART1_TX	TIM1_CH2
PA10	TIM17_BKIN	USART1_RX	TIM1_CH3

Pin name	AF0	AF1	AF2
PB0	EVENTOUT	TIM3_CH3	TIM1_CH2N
PB1	TIM14_CH1	TIM3_CH4	TIM1_CH3N
PB2			
PB3	SPI1_SCK, I2S1_CK	EVENTOUT	TIM2_CH2
PB4	SPI1_MISO, I2S1_MCK	TIM3_CH1	EVENTOUT
PB5	SPI1_MOSI, I2S1_SD	TIM3_CH2	TIM16_BKIN
PB6	USART1_TX	I2C1_SCL	TIM16_CH1N
PB7	USART1_RX	I2C1_SDA	TIM17_CH1N
PB8	CEC	I2C1_SCL	TIM16_CH1
PB9	IR_OUT	I2C1_SDA	TIM17_CH1
PB10	CEC	I2C2_SCL	TIM2_CH3
PB11	EVENTOUT	I2C2_SDA	TIM2_CH4
PB12	SPI2_NSS, I2S2_WS	EVENTOUT	TIM1_BKIN
PB13	SPI2_SCK, I2S2_CK		TIM1_CH1N
PB14	SPI2_MISO, I2S2_MCK	TIM15_CH1	TIM1_CH2N
PB15	SPI2_MOSI, I2S2_SD	TIM15_CH2	TIM1_CH3N


321
CN5
Arduino Connector
10
9
8
7
6
5
4
3
2
1
PB8 D15
PB9 D14
D13
D12
D11
PB6 D10
PC7 D9
PA9 D8
Header 10X1_Female
CN9
Arduino Connector
8
7
6
5
4
3
2
1
PA8 D7
PB10/PE8 D6
PB4 D5
PB5 D4
PB3 D3
PA10 D2
PA2 D1
PA3 D0
Header 8X1_Female
PC9
CN10
1 2
3 4
5 6
7 8
9 10
11 12
13 14
15 16
17 18
19 20
21 22
23 24
25 26
27 28
29 30
31 32
33 34
35 36
37 38
PC8
PC6
PC5
U5V
PD8
PA12
PA11
PB12
PB11/PE9
PB2
PB1
PC4
PF5
PF4
Header 19X2
SB25
SB36
SB28
SB39
PB15
PA7
PB14
PA6
AGND
SB25,SB28,SB35
Open only for F302
PB13
PA5
SB35
SB44

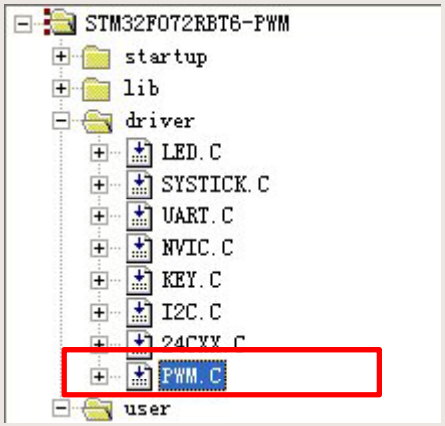
Keil-mdk工程文件及代码实现

四、Keil-mdk工程软件代码实现：

1、Keil-mdk工程模板中加入PWM.C文件，
在includes.h文件中加入PWM.H头文件

2、在PWM.H头文件宏定义

```
#define F072B_TIMER_PWM             TIM1
#define F072B_TIMER_CLK             RCC_APB2Periph_TIM1

#define F072B_PWM_CH_PORT           GPIOA
#define F072B_PWM_CH_CLK            RCC_AHBPeriph_GPIOA

#define F072B_PWM_CHN_PORT          GPIOB
#define F072B_PWM_CHN_CLK           RCC_AHBPeriph_GPIOB

#define F072B_PWM_CH1_PIN           GPIO_Pin_8
#define F072B_PWM_CH1_SOURCE        GPIO_PinSource8
#define F072B_PWM_CH1_AF            GPIO_AF_2

#define F072B_PWM_CH2_PIN           GPIO_Pin_9
#define F072B_PWM_CH2_SOURCE        GPIO_PinSource9
#define F072B_PWM_CH2_AF            GPIO_AF_2

#define F072B_PWM_CH3_PIN           GPIO_Pin_10
#define F072B_PWM_CH3_SOURCE        GPIO_PinSource10
#define F072B_PWM_CH3_AF            GPIO_AF_2

#define F072B_PWM_CH1N_PIN          GPIO_Pin_13
#define F072B_PWM_CH1N_SOURCE       GPIO_PinSource13
#define F072B_PWM_CH1N_AF           GPIO_AF_2

#define F072B_PWM_CH2N_PIN          GPIO_Pin_14
#define F072B_PWM_CH2N_SOURCE       GPIO_PinSource14
#define F072B_PWM_CH2N_AF           GPIO_AF_2

#define F072B_PWM_CH3N_PIN          GPIO_Pin_15
#define F072B_PWM_CH3N_SOURCE       GPIO_PinSource15
#define F072B_PWM_CH3N_AF           GPIO_AF_2
```



Keil-mdk工程文件及代码实现

3、6路PWM引脚端口初始化PWM_GPIO_Init()代码实现

```
void PWM_GPIO_Init(void)
{
    GPIO_InitTypeDef GPIO_InitStructure;
    //时钟配置
    RCC_AHBPeriphClockCmd(FO72B_PWM_CH_CLK|FO72B_PWM_CHN_CLK , ENABLE);    // 使能C

    //GPIOA通道配置 : TIM1 CH1 (PA8)/TIM1 CH2 (PA9) /TIM1 CH3 (PA10)*/
    GPIO_InitStructure.GPIO_Pin =   FO72B_PWM_CH1_PIN|FO72B_PWM_CH2_PIN|FO72B_PWM_CH3_PIN;
    GPIO_InitStructure.GPIO_Mode =  GPIO_Mode_AF;
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
    GPIO_InitStructure.GPIO_PuPd =  GPIO_PuPd_UP ;
    GPIO_Init(FO72B_PWM_CH_PORT , &GPIO_InitStructure);

    //GPIOB通道配置 : TIM1 CH1N (PB13)/TIM1 CH2N (PB14) /TIM1 CH3N (PB15)*/
    GPIO_InitStructure.GPIO_Pin =   FO72B_PWM_CH1N_PIN|FO72B_PWM_CH2N_PIN|FO72B_PWM_CH3N_PIN;
    GPIO_InitStructure.GPIO_Mode =  GPIO_Mode_AF;
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
    GPIO_InitStructure.GPIO_PuPd =  GPIO_PuPd_UP ;
    GPIO_Init(FO72B_PWM_CHN_PORT , &GPIO_InitStructure);

    //端口复用功能配置
    GPIO_PinAFConfig(FO72B_PWM_CH_PORT,    FO72B_PWM_CH1_SOURCE,  FO72B_PWM_CH1_AF);
    GPIO_PinAFConfig(FO72B_PWM_CH_PORT ,   FO72B_PWM_CH2_SOURCE,  FO72B_PWM_CH2_AF);
    GPIO_PinAFConfig(FO72B_PWM_CH_PORT ,   FO72B_PWM_CH3_SOURCE,  FO72B_PWM_CH3_AF);
    GPIO_PinAFConfig(FO72B_PWM_CHN_PORT ,  FO72B_PWM_CH1N_SOURCE, FO72B_PWM_CH1N_AF);
    GPIO_PinAFConfig(FO72B_PWM_CHN_PORT ,  FO72B_PWM_CH2N_SOURCE, FO72B_PWM_CH2N_AF);
    GPIO_PinAFConfig(FO72B_PWM_CHN_PORT ,  FO72B_PWM_CH3N_SOURCE, FO72B_PWM_CH3N_AF);

}
```

端口时钟使能

GPIOA/GPIOB引脚配置

端口引脚映射到PWM功能

4、高级定时器初始化Timer1_Init()代码实现

```
TIM_TimeBaseInitTypeDef TIM_TimeBaseInitStructure;
TIM_OCInitTypeDef  TIM_OCInitStructure;
TIM_BDTRInitTypeDef TIM_BDTRInitStructure;
```

定义三个结构体变量

```
PWM_GPIO_Init(); //端口初始化
RCC_APB2PeriphClockCmd(F072B_TIMER_CLK , ENABLE);   //
//TIM_DeInit(F072B_TIMER_PWM);
```

端口和时钟初始化

```
Pwm_period_num=(SystemCoreClock /8000) - 1;       // 默
```

设定PWM周期值

```
//设定3对PWM占空比
CCR1_Val     = (unsigned short int) (((unsigned long) 300 * (Pwm_period_num   - 1)) / 1000);
CCR2_Val     = (unsigned short int) (((unsigned long) 400 * (Pwm_period_num   - 1)) / 1000);
CCR3_Val     = (unsigned short int) (((unsigned long) 500 * (Pwm_period_num   - 1)) / 1000);
```

设定3对PWM占空比

```
//设置定时器相关参数
TIM_TimeBaseInitStructure.TIM_Prescaler = 0;                        // 48MHZ
TIM_TimeBaseInitStructure.TIM_CounterMode = TIM_CounterMode_Up; // 向上计数模式
TIM_TimeBaseInitStructure.TIM_Period = Pwm_period_num ;             // 自动重装载计数器周期的值
TIM_TimeBaseInitStructure.TIM_ClockDivision = 0;                    // 时钟分频系数--不分频
TIM_TimeBaseInitStructure.TIM_RepetitionCounter = 0;
TIM_TimeBaseInit(F072B_TIMER_PWM, &TIM_TimeBaseInitStructure);
```

```
//PWM模式设置-CH1
TIM_OCInitStructure.TIM_OCMode       = TIM_OCMode_PWM1;          //PWM模式
TIM_OCInitStructure.TIM_OutputState = TIM_OutputState_Enable;  //输出使能
TIM_OCInitStructure.TIM_OutputNState = TIM_OutputNState_Enable;//互补端使能输出
TIM_OCInitStructure.TIM_Pulse = CCR1_Val ; //设置占空比
TIM_OCInitStructure.TIM_OCPolarity    =  TIM_OCPolarity_High;  //设置输出极性
TIM_OCInitStructure.TIM_OCNPolarity  =  TIM_OCPolarity_High;  //设置输出端互补极性
TIM_OCInitStructure.TIM_OCIdleState   =  TIM_OCIdleState_Set;   //死区后
TIM_OCInitStructure.TIM_OCNIdleState =  TIM_OCIdleState_Reset;  //死区
TIM_OC1Init(F072B_TIMER_PWM, &TIM_OCInitStructure);
```

Keil-mdk工程文件及代码实现

```
TIM_OCInitStructure.TIM_Pulse = CCR2_Val ;
TIM_OC2Init(F072B_TIMER_PWM, &TIM_OCInitStructure);

TIM_OCInitStructure.TIM_Pulse = CCR3_Val ;
TIM_OC3Init(F072B_TIMER_PWM, &TIM_OCInitStructure);
```

```
TIM_OC1PreloadConfig(F072B_TIMER_PWM, TIM_OCPreload_Enable);
TIM_OC2PreloadConfig(F072B_TIMER_PWM, TIM_OCPreload_Enable);
TIM_OC3PreloadConfig(F072B_TIMER_PWM, TIM_OCPreload_Enable);
```

```
//死区和刹车功能配置，使用了高级定时器
/**/
TIM_BDTRInitStructure.TIM_OSSRState = TIM_OSSRState_Enable;//运行模式下输出选择
TIM_BDTRInitStructure.TIM_OSSIState = TIM_OSSIState_Enable;//空闲模式下输出选择
TIM_BDTRInitStructure.TIM_LOCKLevel = TIM_LOCKLevel_OFF;    //锁定设置
TIM_BDTRInitStructure.TIM_DeadTime = 0xF2;                  //死区时间--2us
TIM_BDTRInitStructure.TIM_Break = TIM_Break_Disable;         //刹车功能使能
TIM_BDTRInitStructure.TIM_BreakPolarity = TIM_BreakPolarity_Low;//刹车输入极性
TIM_BDTRInitStructure.TIM_AutomaticOutput = TIM_AutomaticOutput_Disable;//自动输出使能
TIM_BDTRConfig(F072B_TIMER_PWM,&TIM_BDTRInitStructure);
```

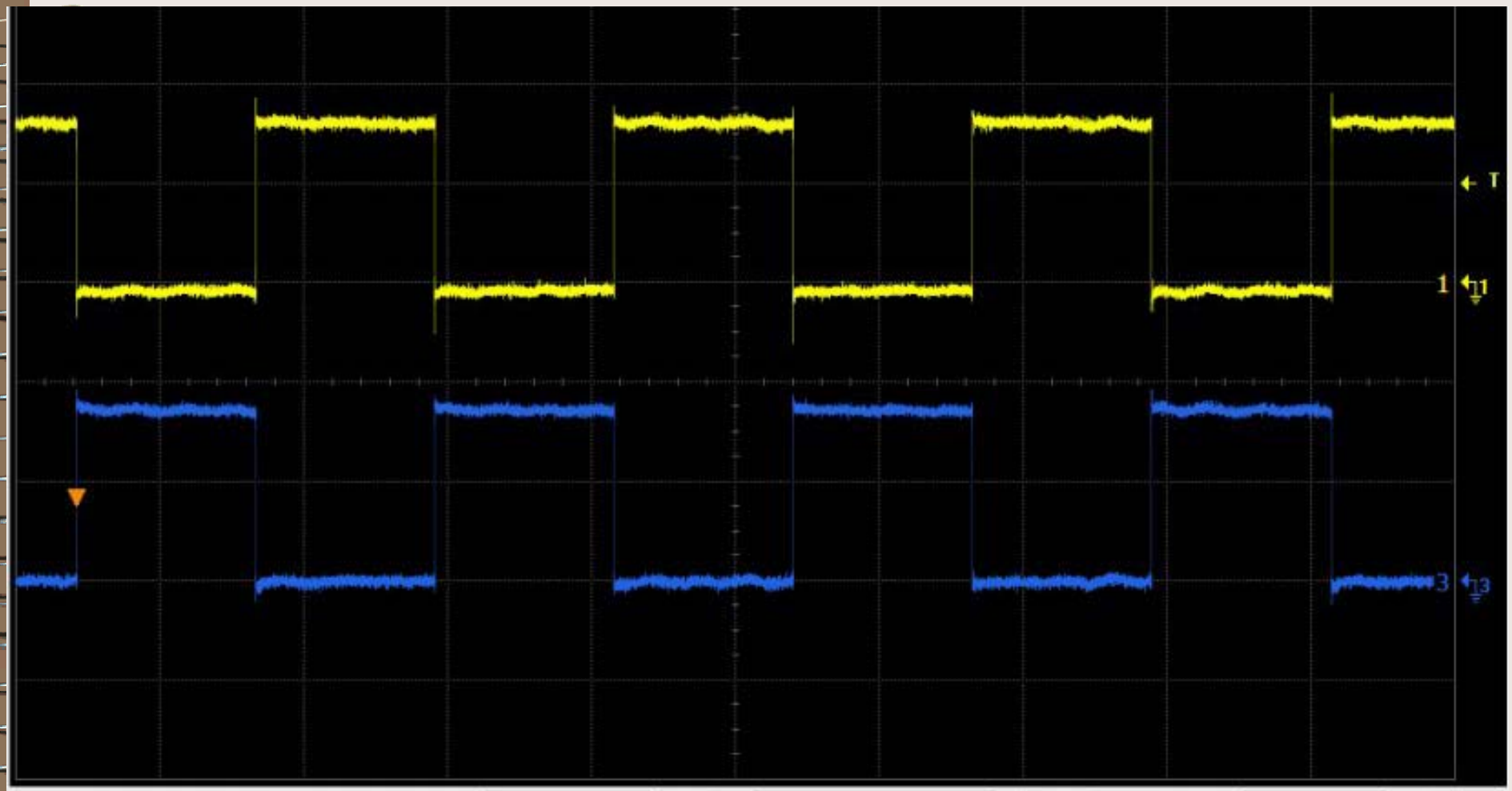
```
//TIM_ARRPreloadConfig(F072B_TIMER_PWM, ENABLE);
TIM_Cmd(F072B_TIMER_PWM, ENABLE);
TIM_CtrlPWMOutputs(F072B_TIMER_PWM, ENABLE);
```

Keil-mdk工程文件及代码实现

5、主函数代码main.c代码实现

```
int main(void)
{
  unsigned short int  i;
  SystemInit(); //系统时钟配置函数，通过不同的时钟定义，来选择不同的主频
  Nvic_Init();
  LED_Init();
  Uart_Init();
  Key_Init();
  I2C_GPIO_Init();
  printf("STM32-NUCLEO-072RB开发板初始化成功!\r\n");
  Timer1_Init();

  while(1)
  {
  }
}
```

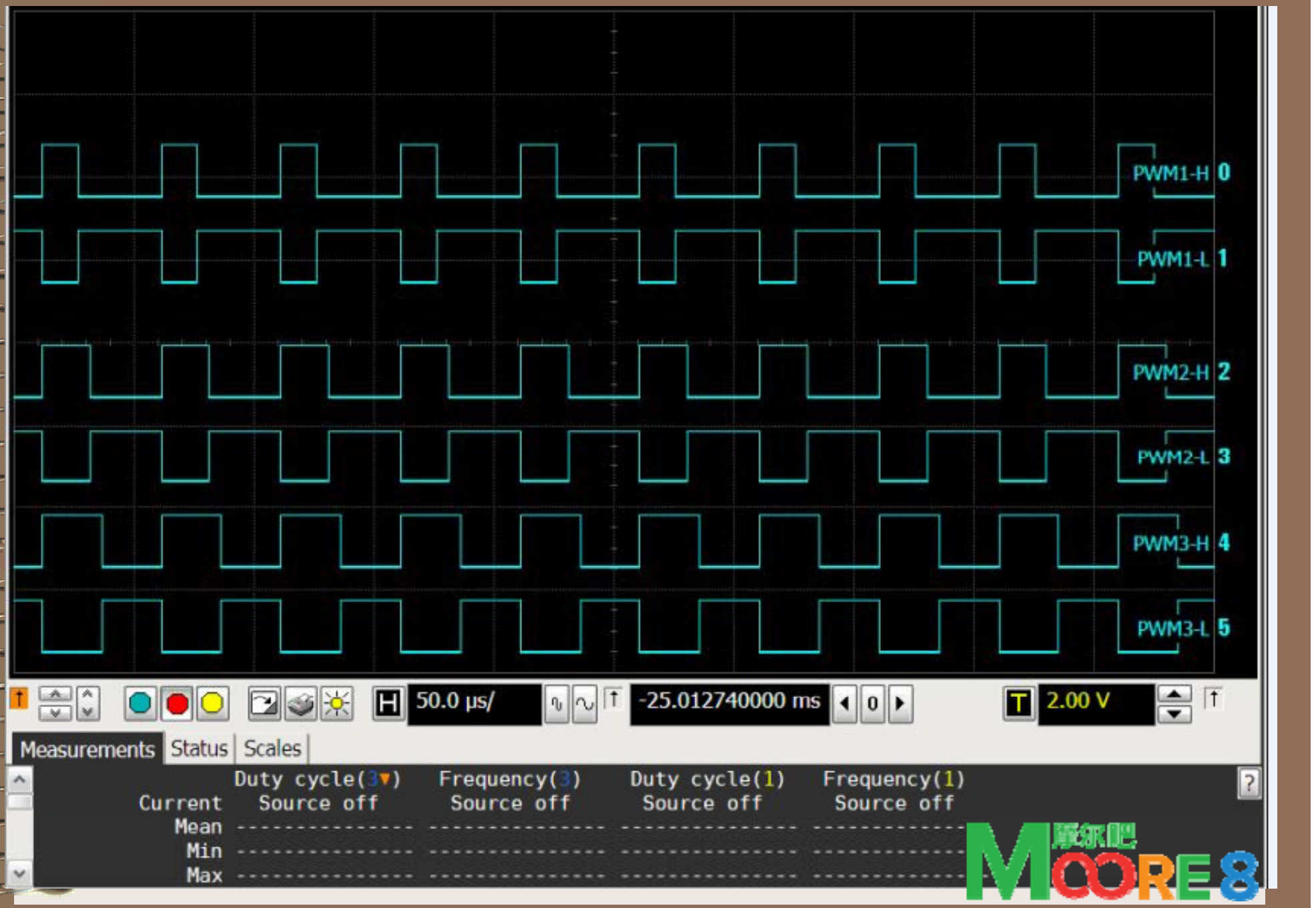
6、实验现象

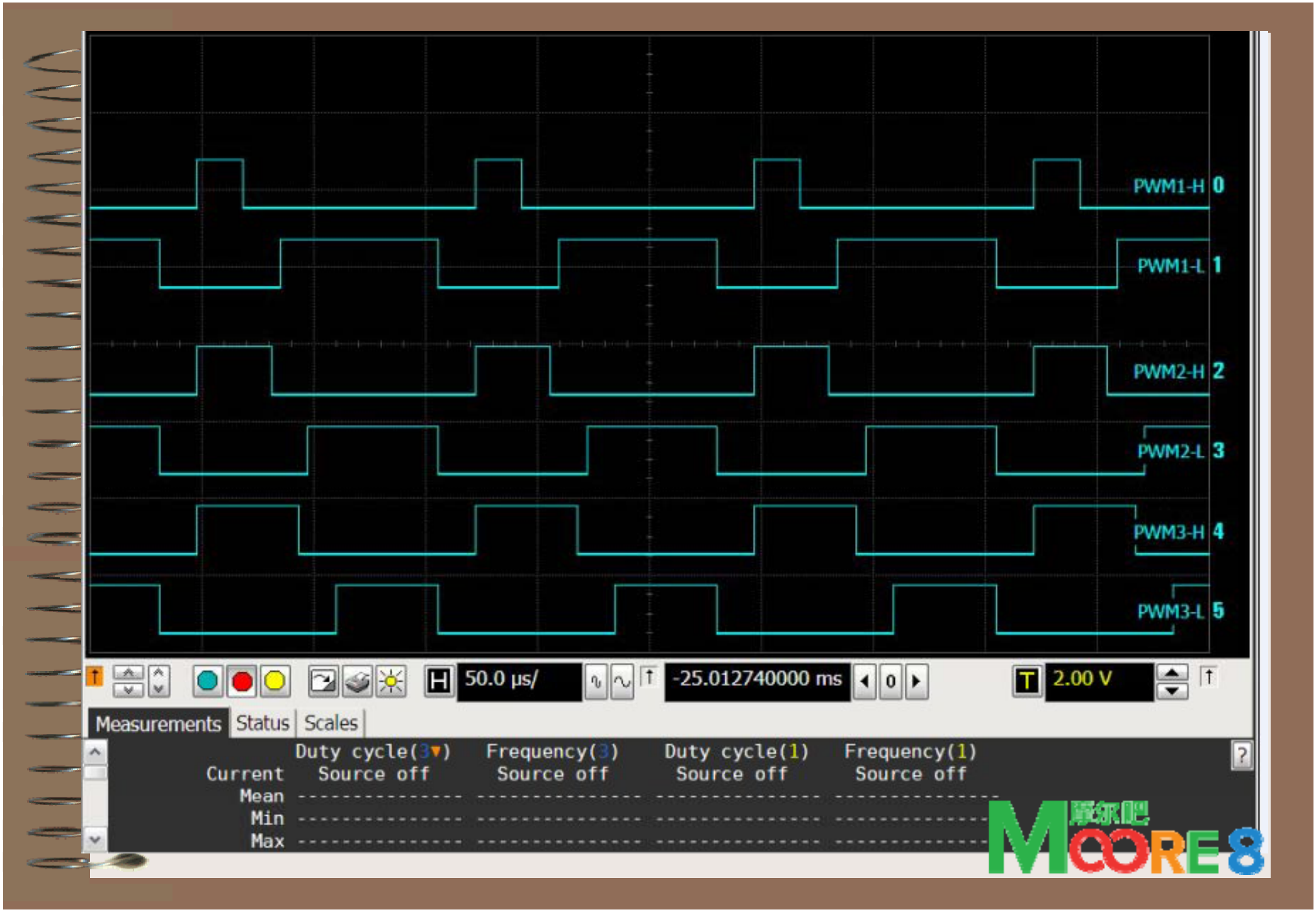


H 20.0 µs/ -1.6995085680 s 0 T 2.00 V

Measurements | Status | Scales

	Duty cycle(3▼)	Frequency(3)	Duty cycle(1)	Frequency(1)
Current	50.1 %	20.076352 kHz	49.9 %	20.076882 kHz
Mean	50.0 %	20.071898 kHz	50.0 %	20.072041 kHz
Min	50.0 %	20.067940 kHz	49.9 %	20.068010 kH
Max	50.1 %	20.076352 kHz	50.0 %	20.076882 kH


PWM1-H 0
PWM1-L 1
PWM2-H 2
PWM2-L 3
PWM3-H 4
PWM3-L 5
H 50.0 µs/
-25.012740000 ms
0
T 2.00 V
Measurements
Status
Scales
Duty cycle(3)
Source off
Frequency(3)
Source off
Duty cycle(1)
Source off
Frequency(1)
Source off
Current
Mean
Min
Max
MOORE 8


PWM1-H 0
PWM1-L 1
PWM2-H 2
PWM2-L 3
PWM3-H 4
PWM3-L 5
H 50.0 µs/
-25.012740000 ms
0
T 2.00 V
Measurements
Status
Scales
Duty cycle(3)
Frequency(3)
Duty cycle(1)
Frequency(1)
Current
Source off
Source off
Source off
Source off
Mean
Min
Max
MOORE8

BLDC(Brushless Direct Current) Motor

BLDC基础知识

一、无刷直流电动机介绍
二、无刷直流电动机结构
三、无刷直流电动机原理
四、六步换向法驱动电机
五、三相H桥逆变电路
六、开关型霍尔传感器

无刷直流电动机结构

BLDC

又称直流永磁电机，或永磁同步电机，采用电机控制器通过电子控制分配方式实现换向!直流无刷电机广泛应用于汽车、工具、工业工控、自动化以及航空航天等行业!

BLDC电机特点：

1、替代直流电机调速、替代变频器+变频电机调速、替代异步电机+减速机调速
2、直流无刷电机具有传统直流电机的所有优点，同时又取消了碳刷、滑环结构
3、直流无刷电机可以低速大功率运行，可以省去减速机直接驱动大的负载；
4、直流无刷电机体积小、重量轻、出力大；
5、转矩特性优异，中、低速转矩性能好，启动转矩大，启动电流小；
6、无级调速，直流无刷电机调速范围广，过载能力强；

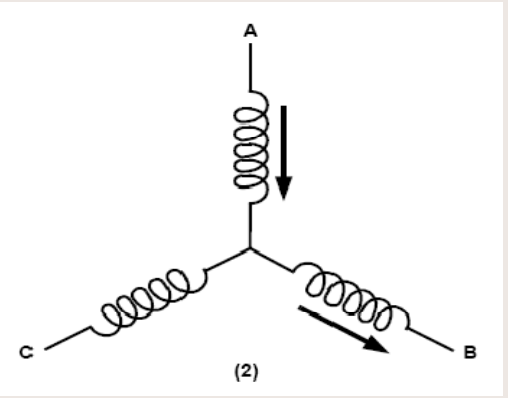
无刷直流电动机结构

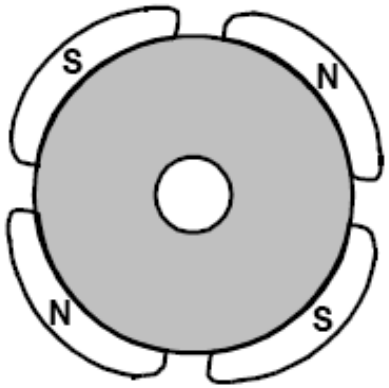
BLDC结构：

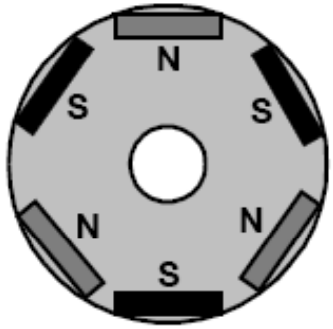
1、由定子、转子、位置传感器组成

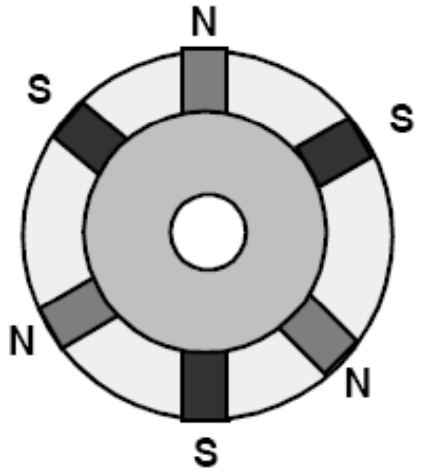
2、定子采用叠片结构并在槽内铺设绕组的方式

3、定子绕组多采用三相并以星形方式连接







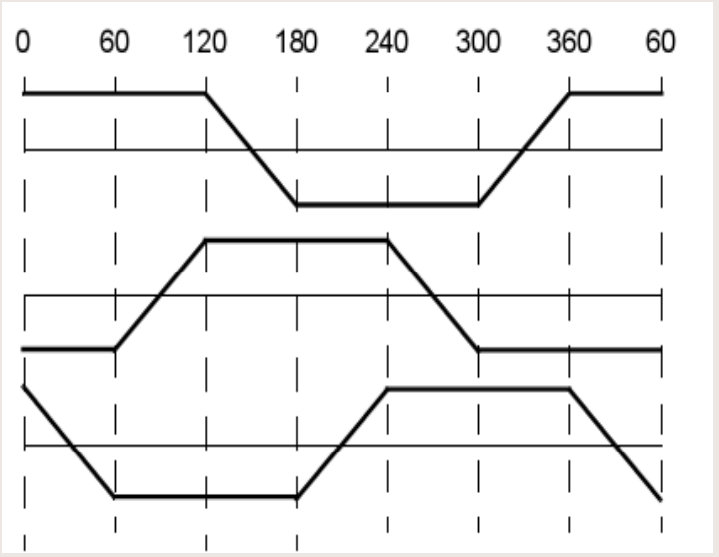


4、将永磁体贴装在非导磁材料表面或镶嵌在其内构成。

5、大部分BLDC采用表面安装方式。

6、多为2到3对极的。

7、磁性材料多采用具有高磁通密度的稀土材料，如鈸铁硼等

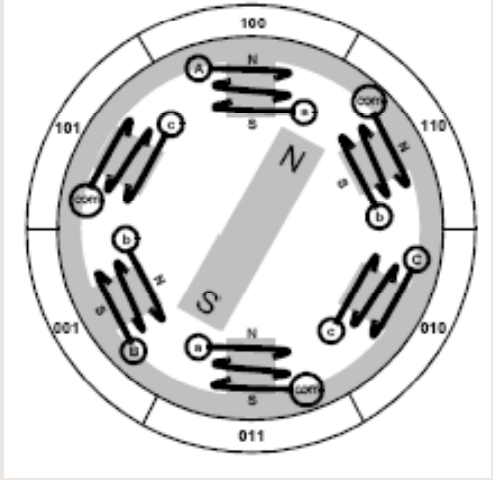


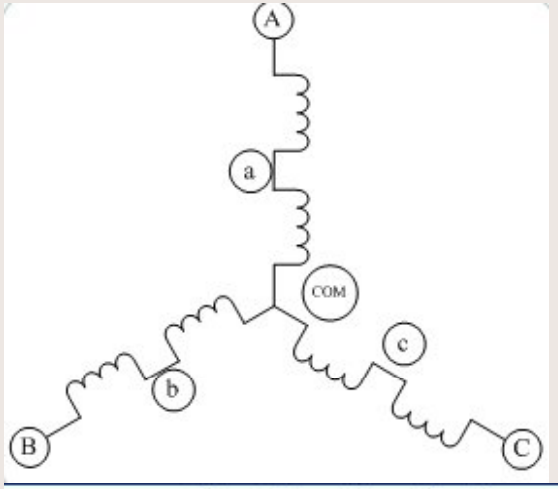
BLDC的定子绕组产生的反电势是梯形波

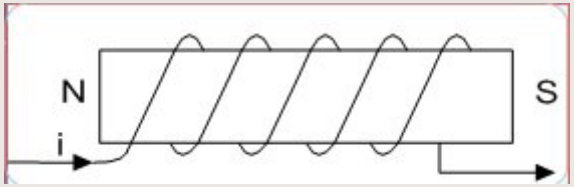
无刷直流电动机原理

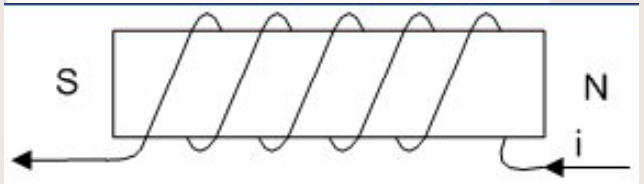
磁场的产生

假定电机定子为3相6极，星型连接。转子为一对极









电流方向不同时，产生的磁场方向不同，比如从A相绕组流入，B相绕组流出，和从B相绕组流入，A相绕组流出产生的磁场方向是不一样的

六步换向法驱动电机

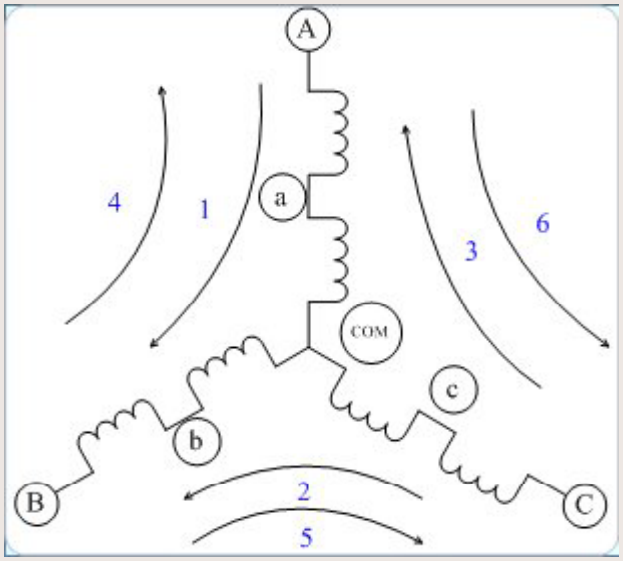
三相绕组通电遵循如下规则：

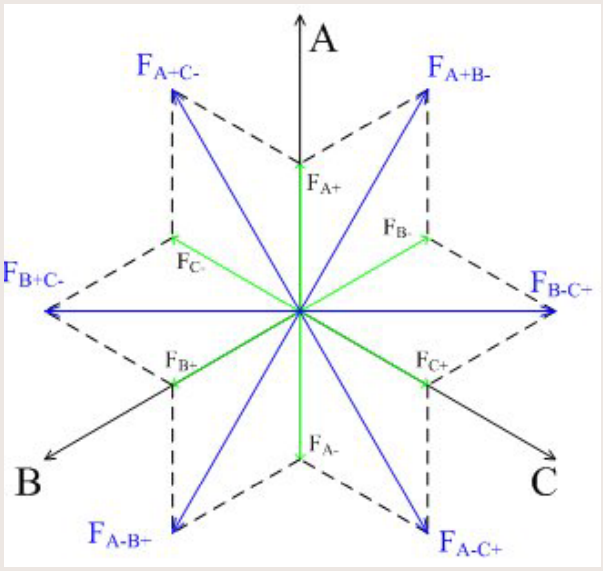
每步三个绕组中一个绕组流入电流，一个绕组流出电流，一个绕组不导通

- 通电顺序如下：

1.A+B-　2.C+B-　3.C+A-　4.B+A-　5. B+C-　6.A+C-

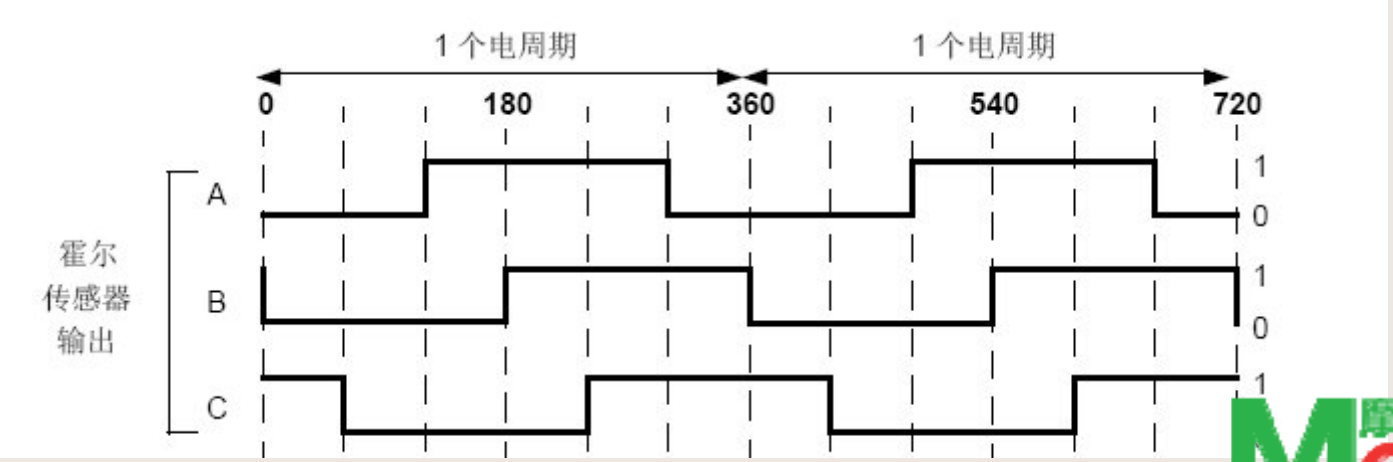
每步磁场旋转60°，每6步磁场旋转一周，每步仅一个绕组被换相



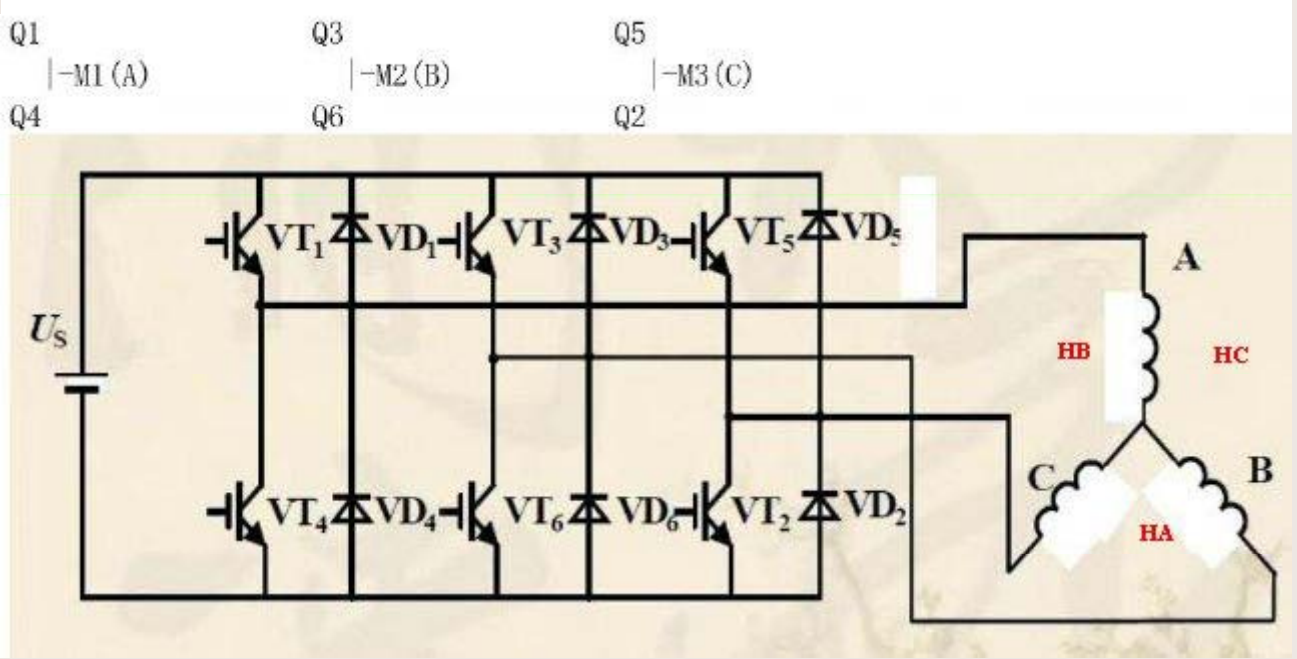


六步换向法驱动电机

- 必须换相才能实现磁场的旋转，如果根据转子磁极的位置换相，并在换相时满足定子磁势和转子磁势相互垂直的条件，就能取得最大转矩
- 要想根据转子磁极的位置换相，换相时就必须知道转子的位置，但并不需要连续的位置信息，只要知道换相点的位置即可
- 在带HALL的BLDC中，一般是由霍尔传感器测量转子的位置，然后由其输出的3位二进制编码去控制逆变器中6个功率管的导通实现换相
- 霍尔传感器在电机中间隔120度或60度按圆周分布



三相H桥逆变电路

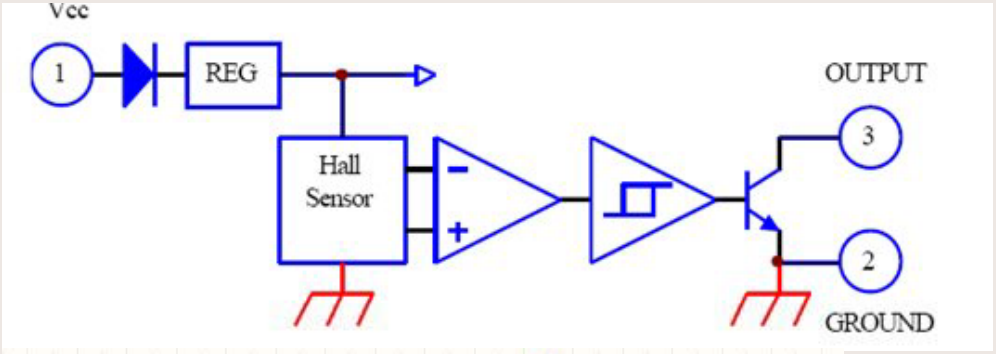


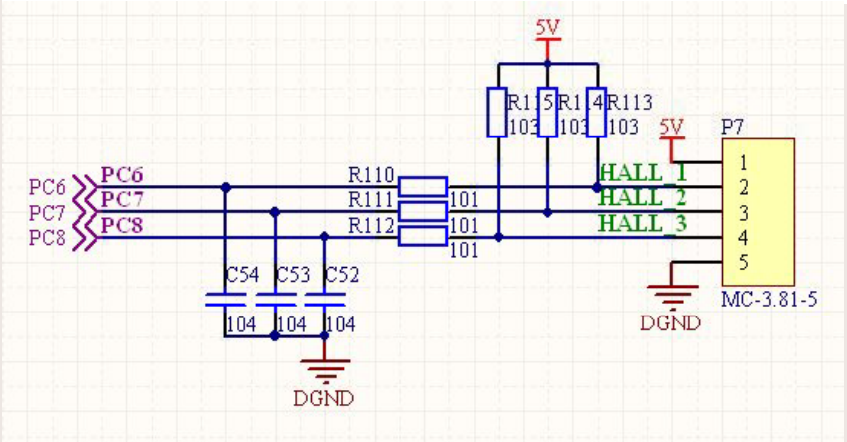
霍尔传感器HC/HB/HA输出二进制编码对于驱动管导通状态

霍尔值	5	4	6	2	3	1
驱动管	T2T3	T3T4	T4T5	T5T6	T6T1	T1T2
导通相	B+C-	B+A-	C+A-	C+B-	A+B-	+C

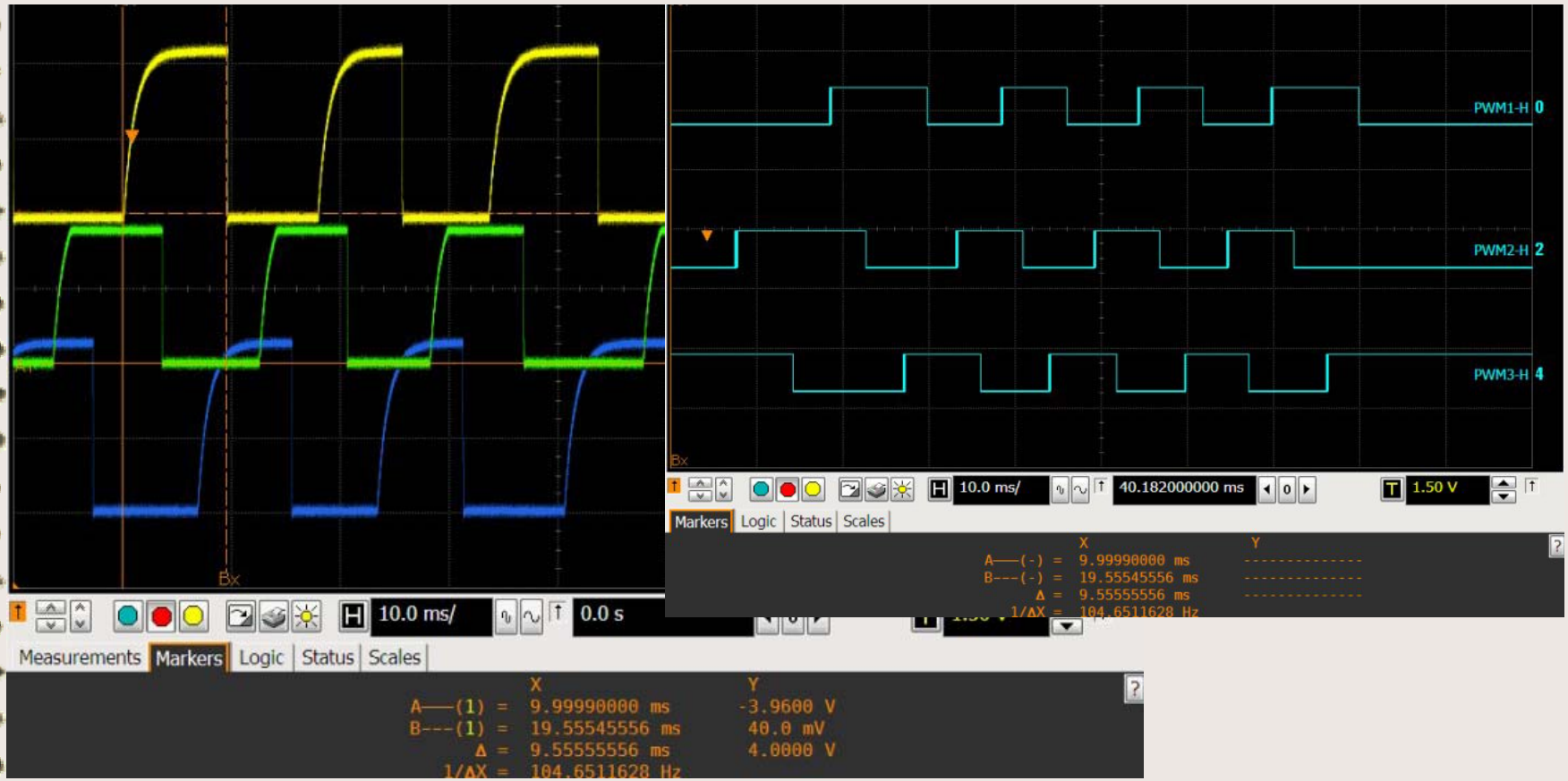
开关型霍尔传感器

- 霍尔元件+信号处理电路=霍尔传感器
- 利用霍尔效应，当施加的磁场达到“动作点”时，OC门输出低电压，称这种状态为“开”；
- 当施加磁场达到“释放点” 使OC门输出高电压，称其为“关”





开关型霍尔传感器



输出值：5-4-6-2-3-1

摩尔吧 MOORE 8

6步换向法输出PWM驱动BLDC

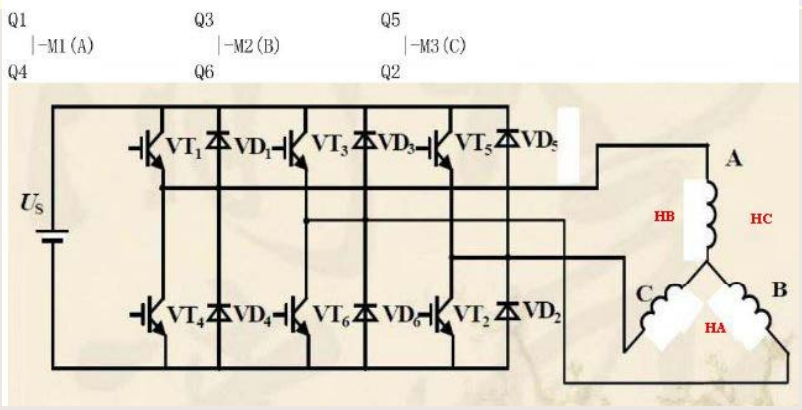
一、换向表分析

BLDC的霍尔传感器不是一上电就处于霍尔值为1的状态，而是处于５４６２３１这6个状态中的任何一个状态，也就是说电机的转子位置不是固定的！所以，在程序中，初始化定时器后，我们要先判定电机的转子位置，也就是判断霍尔值，根据霍尔值来进行换向！

霍尔值	5	4	6	2	3	1
驱动管	T2T3	T3T4	T4T5	T5T6	T6T1	T1T2
导通相	B+C-	B+A-	C+A-	C+B-	A+B-	A+C-

换向表分析

霍尔值	5	4	6	2	3	1
驱动管	T2T3	T3T4	T4T5	T5T6	T6T1	T1T2
导通相	B+C-	B+A-	C+A-	C+B-	A+B-	A+C-
PWM预配置通道	**B+A-**	**C+A-**	**C+B-**	**A+B-**	**A+C-**	**B+C-**
驱动管预导通设置	**T3T4**	**T4T5**	**T5T6**	**T6T1**	**T1T2**	**T2T3**



STM32的COM事件

一、com事件

COM事件是专门为电机控制用的，它只出现在高级定时器中！“当在一个通道上需要互补输出时，预装载位有OCxM、CCxE和CCxNE。在发生COM换相事件时，这些预装载位被传送到影子寄存器。这样你就可以预先设置好下一步骤配置，并在同一个时刻同时修改所有通道的配置。COM可以通过设置TIMx_EGR寄存器的COM位由软件产生，或在TRGI上升沿由硬件产生。”

二、COM事件的触发

1、软件触发：利用库函数实现

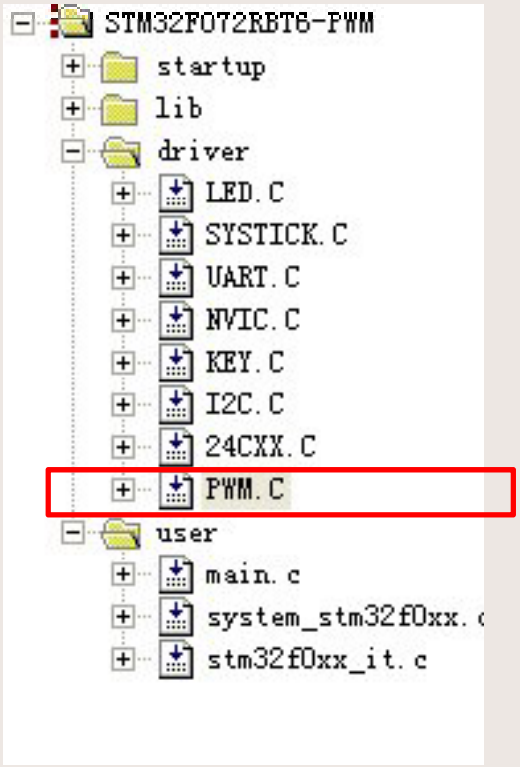
2、硬件触发：利用STM32 定时器霍尔接口实现，由霍尔值改变来产生上升沿触发

Keil-mdk工程文件及代码实现

1、Keil-mdk工程模板中加入PWM.C文件，
在includes.h文件中加入PWM.H头文件

2、在PWM.H头文件宏定义
霍尔传感器引脚PC6/PC7/PC8

```
//霍尔端口初始化
#define FO72B_HALL_PORT                 GPIOC
#define FO72B_HALL_CLK                  RCC_AHBPeriph_GPIOC
#define FO72B_HALL_PIN                  GPIO_Pin_6|GPIO_Pin_7|GPIO_Pin_8
```



Keil-mdk工程文件及代码实现

3、霍尔传感器接口 HALL_GPIO_Init()函数代码实现

```
void HALL_GPIO_Init(void)
{
    GPIO_InitTypeDef GPIO_InitStructure;
    RCC_AHBPeriphClockCmd(FO72B_HALL_CLK , ENABLE);     // 使能GPIOC端口

    GPIO_InitStructure.GPIO_Pin = FO72B_HALL_PIN   ;
    GPIO_InitStructure.GPIO_Mode =  GPIO_Mode_IN;       //GPIO_Mode_AF;//
    GPIO_InitStructure.GPIO_PuPd =  GPIO_PuPd_NOPULL;
    GPIO_Init(FO72B_HALL_PORT, &GPIO_InitStructure);

}
```

4、高级定时器初始化Timer1_Init()代码实现

```
TIM_TimeBaseInitTypeDef TIM_TimeBaseInitStructure;
TIM_OCInitTypeDef  TIM_OCInitStructure;
TIM_BDTRInitTypeDef TIM_BDTRInitStructure;

PWM_GPIO_Init(); //端口初始化
HALL_GPIO_Init();
RCC_APB2PeriphClockCmd(FO72B_TIMER_CLK , ENABLE);  // 打开定时器1的时钟
//TIM_DeInit(FO72B_TIMER_PWM);

Pwm_period_num=(SystemCoreClock /20000) - 1;       // 默认20KHZ
//设定3对PWM占空比
CCR1_Val    = (unsigned short int) (((unsigned long) 500 * (Pwm_period_num   -
CCR2_Val    = (unsigned short int) (((unsigned long) 500 * (Pwm_period_num   -
CCR3_Val    = (unsigned short int) (((unsigned long) 500 * (Pwm_period_num   -

//设置定时器相关参数
TIM_TimeBaseInitStructure.TIM_Prescaler = 0;                    // 48MHZ
TIM_TimeBaseInitStructure.TIM_CounterMode = TIM_CounterMode_Up; // 向上计数模式
TIM_TimeBaseInitStructure.TIM_Period = Pwm_period_num ;         // 自动重装载计
TIM_TimeBaseInitStructure.TIM_ClockDivision = 0;                // 时钟分频系数
TIM_TimeBaseInitStructure.TIM_RepetitionCounter = 0;
```

端口初始化

设置20KHZ，50%占空比

Keil-mdk工程文件及代码实现

```
//PWM模式设置-CH1
TIM_OCInitStructure.TIM_OCMode      = TIM_OCMode_Timing;
TIM_OCInitStructure.TIM_OutputState = TIM_OutputState_Enable;  //
TIM_OCInitStructure.TIM_OutputNState = TIM_OutputNState_Enable;//互补端使能输出
TIM_OCInitStructure.TIM_Pulse = CCR1_Val ; //设置占空比
TIM_OCInitStructure.TIM_OCPolarity   =  TIM_OCPolarity_High;  //设置输出极性
TIM_OCInitStructure.TIM_OCNPolarity  =  TIM_OCNPolarity_High; //设置输出端互补极
TIM_OCInitStructure.TIM_OCIdleState  =  TIM_OCIdleState_Set;  //死区后输出状态
TIM_OCInitStructure.TIM_OCNIdleState =  TIM_OCNIdleState_Set; //死区后互补端输出
TIM_OC1Init(FO72B_TIMER_PWM, &TIM_OCInitStructure);

TIM_OCInitStructure.TIM_Pulse = CCR2_Val ;
TIM_OC2Init(FO72B_TIMER_PWM, &TIM_OCInitStructure);

TIM_OCInitStructure.TIM_Pulse = CCR3_Val ;
TIM_OC3Init(FO72B_TIMER_PWM, &TIM_OCInitStructure);
```

定时模式，不输出**PWM**

```
//死区和刹车功能配置，使用了高级定时器
TIM_BDTRInitStructure.TIM_OSSRState = TIM_OSSRState_Enable;//运行模式下输出
TIM_BDTRInitStructure.TIM_OSSIState = TIM_OSSIState_Enable;//空闲模式下输出
TIM_BDTRInitStructure.TIM_LOCKLevel = TIM_LOCKLevel_OFF;   //锁定设置
TIM_BDTRInitStructure.TIM_DeadTime  = 0x01;                //死区时间--2us
TIM_BDTRInitStructure.TIM_Break = TIM_Break_Disable;        //刹车功能使能
TIM_BDTRInitStructure.TIM_BreakPolarity = TIM_BreakPolarity_High;//刹车输入
TIM_BDTRInitStructure.TIM_AutomaticOutput = TIM_AutomaticOutput_Disable;//
TIM_BDTRConfig(FO72B_TIMER_PWM,&TIM_BDTRInitStructure);

TIM_CCPreloadControl(FO72B_TIMER_PWM, ENABLE);     //使能预装值

TIM_ITConfig(FO72B_TIMER_PWM, TIM_IT_COM, ENABLE); //开启通信事件
TIM_Cmd(FO72B_TIMER_PWM, ENABLE);
TIM_CtrlPWMOutputs(FO72B_TIMER_PWM, ENABLE);
```

开启通信事件中断

Keil-mdk工程文件及代码实现

5、换向函数实现：**BLDC_Hall_Convet()**

```
HALL_Value =(unsigned char)((GPIOC->IDR&0x01C0)>>6);   //
printf("HALL_Value =%d\r\n",HALL_Value);
if(HALL_flag ==1) //正转
{
   switch(HALL_Value)    //根据转子位置，决定CCER输出相位和转子字偏移量
   {
     case 0x05: //配置 T3/T4
```

获取霍尔值

根据霍尔值设置PWM通道—H-PWM—L--ON

```
/*  Channel2 configuration */
TIM_SelectOCxM(TIM1, TIM_Channel_2, TIM_OCMode_PWM1);
TIM_CCxCmd(TIM1, TIM_Channel_2,  TIM_CCx_Enable);
TIM_CCxNCmd(TIM1, TIM_Channel_2, TIM_CCxN_Disable);
```

PWM-2H-PWM

```
/*  Channel1 configuration */
TIM_CCxCmd(TIM1, TIM_Channel_1, TIM_CCx_Disable);
TIM_CCxNCmd(TIM1, TIM_Channel_1, TIM_CCxN_Enable);
TIM_ForcedOC1Config(TIM1,TIM_ForcedAction_Active);
```

PWM-1L-ON

```
/*  Channel3 configuration */
TIM_CCxCmd(TIM1, TIM_Channel_3, TIM_CCx_Disable);
TIM_CCxNCmd(TIM1, TIM_Channel_3,TIM_CCxN_Enable);
TIM_ForcedOC3Config(TIM1,TIM_ForcedAction_InActive);
```

PWM-3H-3L-OFF

```
TIM_GenerateEvent(TIM1, TIM_EventSource_COM);
```

产生霍尔事件

Keil-mdk工程文件及代码实现

```
case 0x04: //T4 T5
      /*  Channel3 configuration */
       TIM_SelectOCxM(TIM1, TIM_Channel_3, TIM_OCMode_PWM1);
       TIM_CCxCmd(TIM1, TIM_Channel_3, TIM_CCx_Enable);
       TIM_CCxNCmd(TIM1, TIM_Channel_3, TIM_CCxN_Disable);

       /*  Channel1 configuration */
       TIM_CCxCmd(TIM1, TIM_Channel_1, TIM_CCx_Disable);
       TIM_CCxNCmd(TIM1, TIM_Channel_1, TIM_CCxN_Enable);
       TIM_ForcedOC1Config(TIM1,TIM_ForcedAction_Active);

       /*  Channel2 configuration */
       TIM_CCxCmd(TIM1, TIM_Channel_2,  TIM_CCx_Disable);
       TIM_CCxNCmd(TIM1, TIM_Channel_2, TIM_CCxN_Enable);
       TIM_ForcedOC2Config(TIM1,TIM_ForcedAction_InActive);

       TIM_GenerateEvent(TIM1, TIM_EventSource_COM);
       break;
```

6、中断复位函数实现：

```
  NVIC_InitStructure.NVIC_IRQChannel = TIM1_BRK_UP_TRG_COM_IRQn;
  NVIC_InitStructure.NVIC_IRQChannelPriority = 0x00;
  NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
  NVIC_Init(&NVIC_InitStructure);
void TIM1_BRK_UP_TRG_COM_IRQHandler(void)
{
  /* Clear TIM1 COM pending bit */
  TIM_ClearITPendingBit(TIM1, TIM_IT_COM);
  BLDC_Hall_Convet();//换向
}
```

Keil-mdk工程文件及代码实现

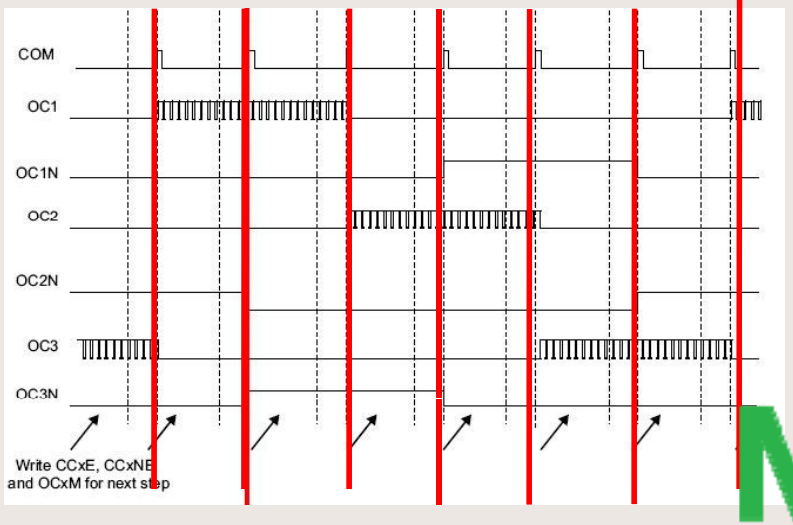
7、主函数代码main.c代码实现

```
int main(void)
{
  unsigned short int  i;
  SystemInit(); //系统时钟配置函数，通过不同的时钟定义，来选择不同的主频
  Nvic_Init();
  LED_Init();
  Uart_Init();
  Key_Init();
  I2C_GPIO_Init();
  //Systick_Init();
  printf("STM32-NUCLEO-072RB开发板初始化成功!\r\n");
  Timer1_Init();
  BLDC_Hall_Convet();

  while(1)
  {



  }



}
```

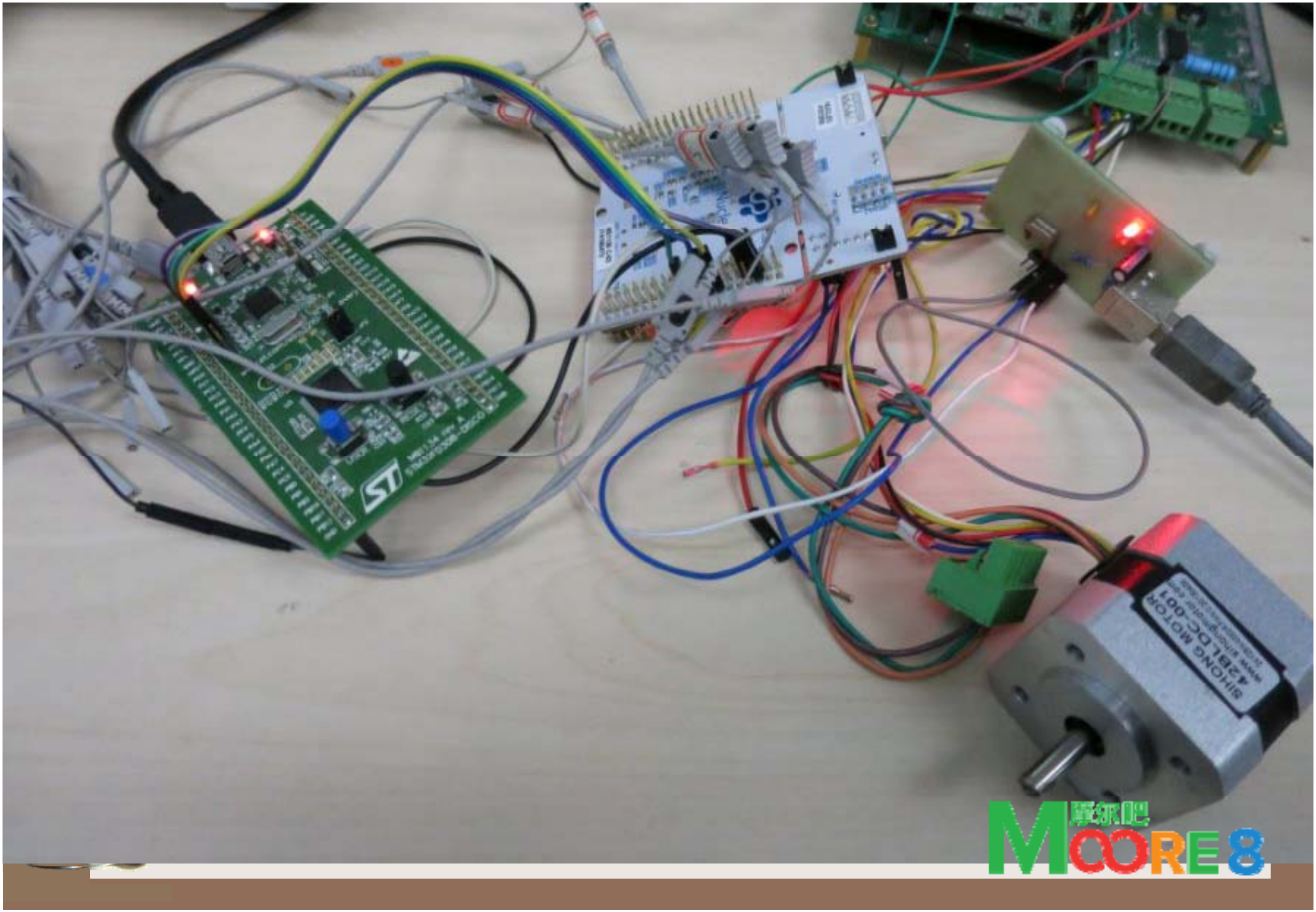
8、实验现象


SSCOM3.2
SSCOM3.2 (作者:聂小猛(丁丁), 主页http://www.mcu51.com,...
HALL_Value =5
HALL_Value =4
HALL_Value =6
HALL_Value =2
HALL_Value =3
HALL_Value =1
打开文件
文件名
发送文件
保存窗口
清除窗口
HEX显示
串口号
COM5
关闭串口
帮助
WWW.MCU51.COM
扩展
波特率
115200
数据位
8
停止位
1
校验位
None
流控制
None
DTR
RTS
定时发送
10
ms/次
HEX发送
发送新行
字符串输入框:
发送
AA
欢迎使用专业串口调试工具SSCOM !
作者: 聂小猛(丁丁)
最新版本下载地址:
http://www.mcu51.com/download/sscom.rar
欢迎提出您的建议!
www.mcu51.com
S:1
R:3179503
COM5已打开 115200bps 8
CTS=0 DSR=0 RLSD=0


PWM1-H 0
PWM1-L 1
PWM2-H 2
PWM2-L 3
PWM3-H 4
PWM3-L 6
5.00 ms/
1.50 V
Markers
Logic
Status
Scales


COM
OC1
OC1N
OC2
OC2N
OC3
OC3N
Write CCxE, CCxNE
and OCxM for next step


摩尔吧
MOORE 8


MOORE 8