Hi
I am writing a code to drive a stepper motor half step as well as micro step. Before going towards the code first the driver what I am using is H bridge.The fig shown below

A Winding1 B Winding 2
H Bridge H Bridge
A0 | /A0 B0 | /B0

A1 | /A1 B1 | /B1

I am really a new guy in this and i have never drive a motor I have written code for half step but I am not sure it will work V supply is 5 volt and the motor is 2 volt . Delay or you can say acceleration for that I have used DMA. but I think I have to use pwm too for (got to know from internet source) to control the heat of the motor how to use that in my code along with the present code . How to achieve for microstepping too .Please help me . I don't know how to do it . Any help will be really appreciated

Half step table (higher Pchannel, Lower N channel)

A0	A1	A0/	A1/	B0	B1	B0/	B1/
0	0	1	1	0	0	1	1
0	0	1	1	1	0	1	0
0	0	1	1	1	1	0	0
1	0	1	0	1	1	0	0
1	1	0	0	1	1	0	0
1	1	0	0	1	0	1	0
1	1	0	0	0	0	1	1
1	0	1	0	0	0	1	1

I have paste the code below which is done so far Please help

```
#include <stm32f0xx.h>
#include "stm32f0xx_adc.h"
#include "stm32f0xx_gpio.h"
#include "stm32f0xx_rcc.h"
#include "stm32f0xx_usart.h"
#include <stm32f0xx_syscfg.h>
#include "stm32f0xx_rcc.h"
#include "stm32f0xx_misc.h"
#include "stm32f0xx_tim.h"
#include "stm32f0xx_dma.h"



#include "stdio.h"

/*****************************************************************************************/
/**
 * global variables
 *
 */
int index = 0;

#define SAMPLES 4

uint32_t SampleVector[SAMPLES] = {60000 -1 , 60000 -1 , 60000 -1, 60000 -1}; // TIM2 32-bit   // 5 milli sec
```

```
/*******************************************************************************/

void RCC_Configuration(void)
{
 /* DMA1 (TIM2 on APB1) clock enable */
 RCC_AHBPeriphClockCmd(RCC_AHBPeriph_DMA1, ENABLE);

 /* TIM2 clock enable */
 RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM3, ENABLE);

 /* GPIOA clock enable */
 RCC_AHBPeriphClockCmd(RCC_AHBPeriph_GPIOA, ENABLE);

 RCC_AHBPeriphClockCmd(RCC_AHBPeriph_GPIOC, ENABLE);

 RCC_AHBPeriphClockCmd(RCC_AHBPeriph_GPIOB, ENABLE);
}

/*******************************************************************************/

void GPIO_Configuration(void)
{
 GPIO_InitTypeDef GPIO_InitStructure;



      GPIO_InitStructure.GPIO_Pin = GPIO_Pin_8 | GPIO_Pin_9;
      GPIO_InitStructure.GPIO_Mode = GPIO_Mode_OUT;
      GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
      GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
      GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL;
      GPIO_Init(GPIOC, &GPIO_InitStructure);

 GPIO_InitStructure.GPIO_Pin = GPIO_Pin_6; // PA1 TIM2_CH2
 GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF;
 GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
 GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL;
 GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
 GPIO_Init(GPIOA, &GPIO_InitStructure);

 /* Connect TIM2 pin  */
 GPIO_PinAFConfig(GPIOA, GPIO_PinSource6, GPIO_AF_1); // PA1 TIM2_CH2


 /*
    * PA8 A0, PA9 A1, PA10 A0/, PA11 /A1 FET 4 pins of left driver controlling the motor
  */

 GPIO_InitStructure.GPIO_Mode = GPIO_Mode_OUT;
  GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_8|GPIO_Pin_9|GPIO_Pin_10|GPIO_Pin_11;
  GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL;
  GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
  GPIO_Init(GPIOA,&GPIO_InitStructure);

  /*
      *  PB4 B0, PB5 B1, PB0 B0/, PB1 /B1 FET 4 pins of right driver controlling the motor
    */
```

```
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_OUT;
      GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
      GPIO_InitStructure.GPIO_Pin = GPIO_Pin_4|GPIO_Pin_5|GPIO_Pin_0|GPIO_Pin_1;
      GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL;
      GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
      GPIO_Init(GPIOB,&GPIO_InitStructure);

     /*
      * Configure ADC3 Channel12 pin as analog input
      */
         GPIO_InitStructure.GPIO_Pin = GPIO_Pin_2;
         GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AN;
         GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL ;
         GPIO_Init(GPIOC, &GPIO_InitStructure);

}

/*****************************************************************************/

void DMA_Configuration(void)
{
 DMA_InitTypeDef DMA_InitStructure;

 DMA_DeInit(DMA1_Channel3);


 DMA_InitStructure.DMA_PeripheralBaseAddr = (uint32_t)&TIM2->ARR;
 DMA_InitStructure.DMA_MemoryBaseAddr = (uint32_t)&SampleVector[0];
 DMA_InitStructure.DMA_DIR =  DMA_DIR_PeripheralSRC;
 DMA_InitStructure.DMA_BufferSize = SAMPLES;
 DMA_InitStructure.DMA_PeripheralInc = DMA_PeripheralInc_Disable;
 DMA_InitStructure.DMA_MemoryInc = DMA_MemoryInc_Enable;
 DMA_InitStructure.DMA_PeripheralDataSize = DMA_PeripheralDataSize_Word; // TIM2 is 32-bit
 DMA_InitStructure.DMA_MemoryDataSize = DMA_MemoryDataSize_Word;
 DMA_InitStructure.DMA_Mode = DMA_Mode_Circular;
//DMA_InitStructure.DMA_Mode = DMA_Mode_Normal;
 DMA_InitStructure.DMA_Priority = DMA_Priority_High;
 DMA_InitStructure.DMA_M2M = DMA_M2M_Disable;
//  DMA_InitStructure.DMA_FIFOMode = DMA_FIFOMode_Enable;
//  DMA_InitStructure.DMA_FIFOThreshold = DMA_FIFOThreshold_Full;
 // DMA_InitStructure.DMA_MemoryBurst = DMA_MemoryBurst_Single;
//  DMA_InitStructure.DMA_PeripheralBurst = DMA_PeripheralBurst_Single;

 DMA_Init( DMA1_Channel3, &DMA_InitStructure);

 DMA_Cmd(DMA1_Channel3, ENABLE);
}

/*****************************************************************************/

void TIM3_Configuration(void)
{
 TIM_TimeBaseInitTypeDef TIM_InitStruct;
 TIM_OCInitTypeDef TIM_OCInitStructure;

 /* Time base configuration */
 TIM_InitStruct.TIM_Prescaler = 4 - 1;              // This will configure the clock to 32 MHz
 TIM_InitStruct.TIM_CounterMode = TIM_CounterMode_Up; // Count-up timer mode
 TIM_InitStruct.TIM_Period = 60000 - 1;              //  5ms initially
```

```
  TIM_InitStruct.TIM_ClockDivision = TIM_CKD_DIV1;     // Divide clock by 1
  TIM_InitStruct.TIM_RepetitionCounter = 0;            // Set to 0, not used
  TIM_TimeBaseInit(TIM3, &TIM_InitStruct);

  TIM_ARRPreloadConfig(TIM3, ENABLE);

  /* Output Compare Toggle Mode configuration: Channel2 - so we can actually measure widths */
  TIM_OCInitStructure.TIM_OCMode = TIM_OCMode_Toggle; // 80 KHz
  TIM_OCInitStructure.TIM_OutputState = TIM_OutputState_Enable;
  TIM_OCInitStructure.TIM_OCPolarity = TIM_OCPolarity_High;
  TIM_OCInitStructure.TIM_Pulse = 1;
  TIM_OC2Init(TIM3, &TIM_OCInitStructure);

  /* TIM2 Update Interrupt enable - for whatever purpose, likely to saturate */
  TIM_ITConfig(TIM3, TIM_IT_Update, ENABLE);

  /* TIM2 Update DMA Request enable */
  TIM_DMACmd(TIM3, TIM_DMA_Update, ENABLE);

  /* TIM2 enable counter */
  TIM_Cmd(TIM3, ENABLE);
}

#if 0
void ADC_init()
{
   ADC_InitTypeDef ADC_InitStructure;
   ADC_CommonInitTypeDef  ADC_CommonInitStructure;
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_ADC3,ENABLE);
   /* ADC Common Init **********************************************************/
   ADC_CommonInitStructure.ADC_Mode = ADC_Mode_Independent;
   ADC_CommonInitStructure.ADC_Prescaler = ADC_Prescaler_Div2;
   ADC_CommonInitStructure.ADC_DMAAccessMode = ADC_DMAAccessMode_Disabled;
   ADC_CommonInitStructure.ADC_TwoSamplingDelay = ADC_TwoSamplingDelay_5Cycles;
   ADC_CommonInit(&ADC_CommonInitStructure);


   /* ADC3 Init ****************************************************************/

   ADC_InitStructure.ADC_Resolution = ADC_Resolution_12b;
   ADC_InitStructure.ADC_ScanConvMode = DISABLE;
   ADC_InitStructure.ADC_ContinuousConvMode = ENABLE;
   ADC_InitStructure.ADC_ExternalTrigConvEdge = ADC_ExternalTrigConvEdge_None;
   ADC_InitStructure.ADC_DataAlign = ADC_DataAlign_Right;
   ADC_InitStructure.ADC_NbrOfConversion = 1;
   ADC_Init(ADC3, &ADC_InitStructure);
    ADC_Cmd(ADC3,ENABLE);

   /* ADC3 regular channel12 configuration *************************************/
   ADC_RegularChannelConfig(ADC3, ADC_Channel_12, 1, ADC_SampleTime_3Cycles);


   /* Enable DMA request after last transfer (Single-ADC mode) */
    ADC_DMARequestAfterLastTransferCmd(ADC3, ENABLE);




   ADC_AnalogWatchdogThresholdsConfig(ADC3, 2200, 0);
```

```
    ADC_AnalogWatchdogSingleChannelConfig(ADC3, ADC_Channel_12);
    ADC_ClearFlag(ADC3, ADC_FLAG_AWD);
    ADC_AnalogWatchdogCmd(ADC3, ADC_AnalogWatchdog_SingleRegEnable);
    ADC_ITConfig(ADC3, ADC_IT_AWD, ENABLE); // Enable ADC1 AWD interrupt




    NVIC_InitTypeDef NVIC_InitStructure;


    /* Configure and enable ADC interrupt */
    NVIC_InitStructure.NVIC_IRQChannel = ADC_IRQn;
    NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0;
    NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0;
    NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
    NVIC_Init(&NVIC_InitStructure);
     ADC_SoftwareStartConv(ADC3);
}
#endif
/*******************************************************************************/

void NVIC_Configuration(void)
{
  NVIC_InitTypeDef NVIC_InitStructure;

  /* Enable TIM2 Interrupt */
  NVIC_InitStructure.NVIC_IRQChannel = TIM3_IRQn;

  NVIC_InitStructure.NVIC_IRQChannelPriority = 0;
  NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
  NVIC_Init(&NVIC_InitStructure);

}

/*******************************************************************************/

void TIM3_IRQHandler(void)
{
#if 0
    static int i = 0;
  if (TIM_GetITStatus(TIM3, TIM_IT_Update) != RESET)
  {

      if (i == 0)
      {
         GPIO_SetBits(GPIOC, (GPIO_Pin_8 | GPIO_Pin_9));
         i = 1;
      }

      else
      {

         GPIO_ResetBits(GPIOC, (GPIO_Pin_8 | GPIO_Pin_9));
        i = 0;
    }
    TIM_ClearITPendingBit(TIM3, TIM_IT_Update);

  }
```

```
#endif
#if 1
 if (TIM_GetITStatus(TIM3, TIM_IT_Update) != RESET)
 {
    switch(index)
      {
       case 0:
       // GPIO_Write(GPIOA,GPIO_Pin_8|GPIO_Pin_11);
          GPIOA->ODR &= 0xF0FF;
          GPIOB->ODR &= 0xFFCC;
          GPIOA->ODR |= 0x0C00;
          GPIOB->ODR |= 0x0003;
          break;

       case 1:

       // GPIO_ResetBits(GPIOB,GPIO_Pin_4|GPIO_Pin_1 );


        GPIOB->ODR &= 0xFFCC;
        GPIOB->ODR |= 0x0011;
         //index = -1;
      // TIM_ITConfig(TIM3, TIM_IT_Update, DISABLE);
       break;

       case 2 :
          //GPIO_Write(GPIOB,GPIO_Pin_5|GPIO_Pin_0);

         GPIOB->ODR &= 0xFFCC;
          GPIOB->ODR |= 0x0030;
           TIM_ITConfig(TIM3, TIM_IT_Update, DISABLE);
       break;

       case 3:
          // GPIO_ResetBits(GPIOA,GPIO_Pin_8|GPIO_Pin_11);
          GPIOA->ODR &= 0xF0FF;
          GPIOA->ODR |= 0x0500;

       break;

       case 4:
          //GPIO_Write(GPIOA,GPIO_Pin_9|GPIO_Pin_10);
          GPIOA->ODR &= 0xF0FF;
          GPIOA->ODR |= 0x0300;

       break;

       case 5:
       //    GPIO_ResetBits(GPIOB,GPIO_Pin_5|GPIO_Pin_0);
          GPIOB->ODR &= 0xFFCC;
          GPIOB->ODR |= 0x0011;
       break;
       case 6:
          // GPIO_Write(GPIOB,GPIO_Pin_4|GPIO_Pin_1);
          GPIOB->ODR &=  0xFFCC;;
          GPIOB->ODR |= 0x0003;
       break;
       case 7:
          //GPIO_ResetBits(GPIOA,GPIO_Pin_9|GPIO_Pin_10);
```

```
            GPIOA->ODR &= 0xF0FF;
             GPIOA->ODR |= 0x0500;
             index = -1;


          break;


          }
          index ++ ;
       // index = 0;
          TIM_ClearITPendingBit(TIM3, TIM_IT_Update);

    }
  #endif
  }


  /*******************************************************************************/

  int main(void)
  {
   RCC_Configuration();

   NVIC_Configuration();

   GPIO_Configuration();

   //STM_EVAL_LEDInit(LED3);

   DMA_Configuration();

   TIM3_Configuration();

   while(1); // Don't want to exit
  }

  /*******************************************************************************/

  #ifdef  USE_FULL_ASSERT
  /**
    * @brief  Reports the name of the source file and the source line number
    *   where the assert_param error has occurred.
    * @param  file: pointer to the source file name
    * @param  line: assert_param error line source number
    * @retval None
    */
  void assert_failed(uint8_t* file, uint32_t line)
  {
    /* User can add his own implementation to report the file name and line number,
       ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */

    /* Infinite loop */
    while (1)
    {
    }
  }
  #endif
```

```
/*****************************************************************************************/
```