```
#include "main.h"
#include "stm32f3xx_hal_tim.h"
#include "stm32f3xx_hal_adc.h"

UART_HandleTypeDef huart2;

void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_USART2_UART_Init(void);

// TIMインスタンス
TIM_HandleTypeDef htim1;

ADC_HandleTypeDef hadc1;

// ホールセンサの状態を示すフラグ
volatile uint8_t A_Phase_FlagR = 0;
volatile uint8_t A_Phase_FlagF = 0;
volatile uint8_t B_Phase_FlagR = 0;
volatile uint8_t B_Phase_FlagF = 0;
volatile uint8_t C_Phase_FlagR = 0;
volatile uint8_t C_Phase_FlagF = 0;

// GPIOポートとピンの設定
#define A_PHASE_PORT GPIOA
#define A_PHASE_PIN GPIO_PIN_8

#define B_PHASE_PORT GPIOA
#define B_PHASE_PIN GPIO_PIN_9

#define C_PHASE_PORT GPIOA
#define C_PHASE_PIN GPIO_PIN_10

#define ENABLE_A_PORT GPIOC
#define ENABLE_A_PIN GPIO_PIN_10

#define ENABLE_B_PORT GPIOC
#define ENABLE_B_PIN GPIO_PIN_11

#define ENABLE_C_PORT GPIOC
#define ENABLE_C_PIN GPIO_PIN_12

#define HALL_SENSOR_A_PIN GPIO_PIN_15
#define HALL_SENSOR_A_PORT GPIOA
#define HALL_SENSOR_A_EXTI_IRQn EXTI15_10_IRQn

#define HALL_SENSOR_B_PIN GPIO_PIN_3
#define HALL_SENSOR_B_PORT GPIOB
#define HALL_SENSOR_B_EXTI_IRQn EXTI3_IRQn

#define HALL_SENSOR_C_PIN GPIO_PIN_10
#define HALL_SENSOR_C_PORT GPIOB
#define HALL_SENSOR_C_EXTI_IRQn EXTI15_10_IRQn
```

```
void motorInit() {
    // GPIOのクロックを有効化
    __HAL_RCC_GPIOA_CLK_ENABLE();
    __HAL_RCC_GPIOB_CLK_ENABLE();
    __HAL_RCC_GPIOC_CLK_ENABLE();

    // GPIOの設定
    GPIO_InitTypeDef GPIO_InitStruct;
    GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
    GPIO_InitStruct.Pull = GPIO_NOPULL;
    GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_HIGH;
    GPIO_InitStruct.Alternate = GPIO_AF6_TIM1;

    GPIO_InitStruct.Pin = A_PHASE_PIN;
    HAL_GPIO_Init(A_PHASE_PORT, &GPIO_InitStruct);

    GPIO_InitStruct.Pin = B_PHASE_PIN;
    HAL_GPIO_Init(B_PHASE_PORT, &GPIO_InitStruct);

    GPIO_InitStruct.Pin = C_PHASE_PIN;
    HAL_GPIO_Init(C_PHASE_PORT, &GPIO_InitStruct);

    GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
    GPIO_InitStruct.Pull = GPIO_NOPULL;
    GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;

    GPIO_InitStruct.Pin = ENABLE_A_PIN;
    HAL_GPIO_Init(ENABLE_A_PORT, &GPIO_InitStruct);

    GPIO_InitStruct.Pin = ENABLE_B_PIN;
    HAL_GPIO_Init(ENABLE_B_PORT, &GPIO_InitStruct);

    GPIO_InitStruct.Pin = ENABLE_C_PIN;
    HAL_GPIO_Init(ENABLE_C_PORT, &GPIO_InitStruct);

    GPIO_InitStruct.Pin = GPIO_PIN_5;
    HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

    // モータードライバのイネーブルピンをリセット状態に設定
    HAL_GPIO_WritePin(ENABLE_A_PORT, ENABLE_A_PIN, GPIO_PIN_RESET);
    HAL_GPIO_WritePin(ENABLE_B_PORT, ENABLE_B_PIN, GPIO_PIN_RESET);
    HAL_GPIO_WritePin(ENABLE_C_PORT, ENABLE_C_PIN, GPIO_PIN_RESET);
}

void PWM_Init() {
    // タイマーのクロックを有効化
    __HAL_RCC_TIM1_CLK_ENABLE();

    htim1.Instance = TIM1;
    htim1.Init.Prescaler = 0;
    htim1.Init.CounterMode = TIM_COUNTERMODE_UP;
    htim1.Init.Period = 999;  // 50kHzの周期に設定
```

```
    htim1.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
    htim1.Init.RepetitionCounter = 0;
    HAL_TIM_PWM_Init(&htim1);

    TIM_OC_InitTypeDef sConfigOC;
    sConfigOC.OCMode = TIM_OCMODE_PWM1;
    sConfigOC.Pulse = 500;  // 50%のデューティ比に設定
    sConfigOC.OCPolarity = TIM_OCPOLARITY_HIGH;
    sConfigOC.OCFastMode = TIM_OCFAST_DISABLE;

    HAL_TIM_PWM_ConfigChannel(&htim1, &sConfigOC, TIM_CHANNEL_1);
    HAL_TIM_PWM_ConfigChannel(&htim1, &sConfigOC, TIM_CHANNEL_2);
    HAL_TIM_PWM_ConfigChannel(&htim1, &sConfigOC, TIM_CHANNEL_3);

    HAL_TIM_PWM_Start(&htim1, TIM_CHANNEL_1);
    HAL_TIM_PWM_Start(&htim1, TIM_CHANNEL_2);
    HAL_TIM_PWM_Start(&htim1, TIM_CHANNEL_3);
}

void setPhase(uint8_t phase, uint16_t duty) {
    switch(phase) {
        case 0:
                TIM1->CCR1 = 0;
                TIM1->CCR2 = 0;
                TIM1->CCR3 = 0;
                HAL_GPIO_WritePin(ENABLE_A_PORT, ENABLE_A_PIN, GPIO_PIN_RESET);
                HAL_GPIO_WritePin(ENABLE_B_PORT, ENABLE_B_PIN, GPIO_PIN_RESET);
                HAL_GPIO_WritePin(ENABLE_C_PORT, ENABLE_C_PIN, GPIO_PIN_RESET);
                break;
        case 1:
                TIM1->CCR1 = duty;
                TIM1->CCR2 = 0;
                TIM1->CCR3 = 0;
                HAL_GPIO_WritePin(ENABLE_A_PORT, ENABLE_A_PIN, GPIO_PIN_SET);
                HAL_GPIO_WritePin(ENABLE_B_PORT, ENABLE_B_PIN, GPIO_PIN_SET);
                HAL_GPIO_WritePin(ENABLE_C_PORT, ENABLE_C_PIN, GPIO_PIN_RESET);
                break;
        case 2:
                TIM1->CCR1 = duty;
                TIM1->CCR2 = 0;
                TIM1->CCR3 = 0;
                HAL_GPIO_WritePin(ENABLE_A_PORT, ENABLE_A_PIN, GPIO_PIN_SET);
                HAL_GPIO_WritePin(ENABLE_B_PORT, ENABLE_B_PIN, GPIO_PIN_RESET);
                HAL_GPIO_WritePin(ENABLE_C_PORT, ENABLE_C_PIN, GPIO_PIN_SET);
                break;
        case 3:
                TIM1->CCR1 = 0;
                TIM1->CCR2 = duty;
                TIM1->CCR3 = 0;
                HAL_GPIO_WritePin(ENABLE_A_PORT, ENABLE_A_PIN, GPIO_PIN_RESET);
                HAL_GPIO_WritePin(ENABLE_B_PORT, ENABLE_B_PIN, GPIO_PIN_SET);
                HAL_GPIO_WritePin(ENABLE_C_PORT, ENABLE_C_PIN, GPIO_PIN_SET);
                break;
```

```
        case 4:
                TIM1->CCR1 = 0;
                TIM1->CCR2 = duty;
                TIM1->CCR3 = 0;
                HAL_GPIO_WritePin(ENABLE_A_PORT, ENABLE_A_PIN, GPIO_PIN_SET);
                HAL_GPIO_WritePin(ENABLE_B_PORT, ENABLE_B_PIN, GPIO_PIN_SET);
                HAL_GPIO_WritePin(ENABLE_C_PORT, ENABLE_C_PIN, GPIO_PIN_RESET);
                break;
        case 5:
                TIM1->CCR1 = 0;
                TIM1->CCR2 = 0;
                TIM1->CCR3 = duty;
                HAL_GPIO_WritePin(ENABLE_A_PORT, ENABLE_A_PIN, GPIO_PIN_SET);
                HAL_GPIO_WritePin(ENABLE_B_PORT, ENABLE_B_PIN, GPIO_PIN_RESET);
                HAL_GPIO_WritePin(ENABLE_C_PORT, ENABLE_C_PIN, GPIO_PIN_SET);
                break;
        case 6:
                TIM1->CCR1 = 0;
                TIM1->CCR2 = 0;
                TIM1->CCR3 = duty;
                HAL_GPIO_WritePin(ENABLE_A_PORT, ENABLE_A_PIN, GPIO_PIN_RESET);
                HAL_GPIO_WritePin(ENABLE_B_PORT, ENABLE_B_PIN, GPIO_PIN_SET);
                HAL_GPIO_WritePin(ENABLE_C_PORT, ENABLE_C_PIN, GPIO_PIN_SET);
                break;
       default:
                HAL_GPIO_WritePin(ENABLE_A_PORT, ENABLE_A_PIN, GPIO_PIN_RESET);
                HAL_GPIO_WritePin(ENABLE_B_PORT, ENABLE_B_PIN, GPIO_PIN_RESET);
                HAL_GPIO_WritePin(ENABLE_C_PORT, ENABLE_C_PIN, GPIO_PIN_RESET);
                break;
    }
    //for(uint16_t i=0;i<2000;i++);
}

#if 0
void ADC_Init() {
    // ADCのクロックを有効化
    __HAL_RCC_ADC1_CLK_ENABLE();
    // GPIOのクロックを有効化
    __HAL_RCC_GPIOB_CLK_ENABLE();

    // GPIOの設定
    GPIO_InitTypeDef GPIO_InitStruct;
    GPIO_InitStruct.Pin = GPIO_PIN_1;
    GPIO_InitStruct.Mode = GPIO_MODE_ANALOG;
    GPIO_InitStruct.Pull = GPIO_NOPULL;
    HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);

    // ADC設定
    ADC_ChannelConfTypeDef sConfig;
    hadc1.Instance = ADC1;
    hadc1.Init.ClockPrescaler = ADC_CLOCK_SYNC_PCLK_DIV1;
    hadc1.Init.Resolution = ADC_RESOLUTION_12B;
    hadc1.Init.ScanConvMode = DISABLE;
```

```
    hadc1.Init.ContinuousConvMode = ENABLE;
    hadc1.Init.DiscontinuousConvMode = DISABLE;
    hadc1.Init.ExternalTrigConvEdge = ADC_EXTERNALTRIGCONVEDGE_NONE;
    hadc1.Init.ExternalTrigConv = ADC_SOFTWARE_START;
    hadc1.Init.DataAlign = ADC_DATAALIGN_RIGHT;
    hadc1.Init.NbrOfConversion = 1;
    HAL_ADC_Init(&hadc1);

    // ADCチャンネル設定
    sConfig.Channel = ADC_CHANNEL_12;
    sConfig.Rank = 1;
    sConfig.SamplingTime = ADC_SAMPLETIME_1CYCLE_5;
     HAL_ADC_ConfigChannel(&hadc1, &sConfig);
}
#else
void ADC_Init() {
    // ADCのクロックを有効化
    __HAL_RCC_ADC1_CLK_ENABLE();
    // GPIOのクロックを有効化
    __HAL_RCC_GPIOC_CLK_ENABLE();

    // GPIOの設定
    GPIO_InitTypeDef GPIO_InitStruct;
    GPIO_InitStruct.Pin = GPIO_PIN_2;
    GPIO_InitStruct.Mode = GPIO_MODE_ANALOG;
    GPIO_InitStruct.Pull = GPIO_NOPULL;
    HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);

    // ADC設定
    ADC_ChannelConfTypeDef sConfig;
    hadc1.Instance = ADC1;
    hadc1.Init.ClockPrescaler = ADC_CLOCK_SYNC_PCLK_DIV1;
    hadc1.Init.Resolution = ADC_RESOLUTION_12B;
    hadc1.Init.ScanConvMode = DISABLE;
    hadc1.Init.ContinuousConvMode = ENABLE;
    hadc1.Init.DiscontinuousConvMode = DISABLE;
    hadc1.Init.ExternalTrigConvEdge = ADC_EXTERNALTRIGCONVEDGE_NONE;
    hadc1.Init.ExternalTrigConv = ADC_SOFTWARE_START;
    hadc1.Init.DataAlign = ADC_DATAALIGN_RIGHT;
    hadc1.Init.NbrOfConversion = 1;
    HAL_ADC_Init(&hadc1);

    // ADCチャンネル設定
    sConfig.Channel = ADC_CHANNEL_8;// for PC2
    sConfig.Rank = 1;
    sConfig.SamplingTime = ADC_SAMPLETIME_1CYCLE_5;
    HAL_ADC_ConfigChannel(&hadc1, &sConfig);
}
#endif
// ADC値の取得
uint16_t readADC() {
    // ADC変換開始
    HAL_ADC_Start(&hadc1);
```

```
    // 変換完了待ち
    HAL_ADC_PollForConversion(&hadc1, HAL_MAX_DELAY);

    // ADC値の取得
    uint16_t adcValue = HAL_ADC_GetValue(&hadc1);

    return adcValue;
}


int main(void)
{
  HAL_Init();

  SystemClock_Config();

  MX_GPIO_Init();
  MX_USART2_UART_Init();

  // モータの初期化
  motorInit();
  // PWMの初期化
  PWM_Init();
  // ADCの初期化
  ADC_Init();
// ホールセンサーの割り込み設定
    __HAL_RCC_GPIOA_CLK_ENABLE();
    __HAL_RCC_GPIOB_CLK_ENABLE();

    GPIO_InitTypeDef GPIO_InitStruct;
 // 割り込みを立ち上がりエッジと立ち下がりエッジで検出
    GPIO_InitStruct.Mode = GPIO_MODE_IT_RISING_FALLING;
    GPIO_InitStruct.Pull = GPIO_NOPULL; // プルダウンなし
    GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;

    GPIO_InitStruct.Pin = HALL_SENSOR_A_PIN;
    HAL_GPIO_Init(HALL_SENSOR_A_PORT, &GPIO_InitStruct);

    GPIO_InitStruct.Pin = HALL_SENSOR_B_PIN;
    HAL_GPIO_Init(HALL_SENSOR_B_PORT, &GPIO_InitStruct);

    GPIO_InitStruct.Pin = HALL_SENSOR_C_PIN;
    HAL_GPIO_Init(HALL_SENSOR_C_PORT, &GPIO_InitStruct);

   // 割り込み優先度の設定
    HAL_NVIC_SetPriority(HALL_SENSOR_A_EXTI_IRQn, 15, 0);
    HAL_NVIC_SetPriority(HALL_SENSOR_B_EXTI_IRQn, 15, 0);
    HAL_NVIC_SetPriority(HALL_SENSOR_C_EXTI_IRQn, 15, 0);

    // 割り込み処理の有効化
    HAL_NVIC_EnableIRQ(HALL_SENSOR_A_EXTI_IRQn);
    HAL_NVIC_EnableIRQ(HALL_SENSOR_B_EXTI_IRQn);
```

```
    HAL_NVIC_EnableIRQ(HALL_SENSOR_C_EXTI_IRQn);

    for(uint16_t i=0;i<100;i++){
        setPhase(i%7, 500);
        HAL_Delay(10);
    }
  while (1)
  {
            uint16_t adcValue = readADC();
            uint16_t duty = (adcValue * 999) / 4095;  // ADC値をPWM周期に変換
            while(A_Phase_FlagR == 0);
            setPhase(1,duty);
         A_Phase_FlagR = 0;  // フラグをクリア
            while(C_Phase_FlagF == 0);
            setPhase(2,duty);
         C_Phase_FlagF = 0;  // フラグをクリア
            while(B_Phase_FlagR == 0);
            setPhase(3,duty);
         B_Phase_FlagR = 0;  // フラグをクリア
            while(A_Phase_FlagF == 0);
            setPhase(4,duty);
         A_Phase_FlagF = 0;  // フラグをクリア
            while(C_Phase_FlagR == 0);
            setPhase(5,duty);
         C_Phase_FlagR = 0;  // フラグをクリア
            while(B_Phase_FlagF == 0);
            setPhase(6,duty);
         B_Phase_FlagF = 0;  // フラグをクリア
            setPhase(0,duty);
  }
}

void EXTI15_10_IRQHandler(void) {
  if (__HAL_GPIO_EXTI_GET_IT(HALL_SENSOR_A_PIN) != RESET)
  {
    if (HAL_GPIO_ReadPin(HALL_SENSOR_A_PORT, HALL_SENSOR_A_PIN) == GPIO_PIN_SET)
    {
      // 立ち上がりエッジの処理
      //setPhase(1, cduty);
        A_Phase_FlagR=1;
    }
    else
    {
       // 立ち下がりエッジの処理
      //setPhase(4, cduty);
        A_Phase_FlagF=1;
     }
    // 割り込みフラグクリア
    __HAL_GPIO_EXTI_CLEAR_IT(HALL_SENSOR_A_PIN);
  }
  if (__HAL_GPIO_EXTI_GET_IT(HALL_SENSOR_C_PIN) != RESET)
  {
    if (HAL_GPIO_ReadPin(HALL_SENSOR_C_PORT, HALL_SENSOR_C_PIN) == GPIO_PIN_SET)
```

```
    {
      // 立ち上がりエッジの処理
        C_Phase_FlagR=1;
      //setPhase(5, cduty);
    }
    else
    {
     // 立ち下がりエッジの処理
        C_Phase_FlagF=1;
      //setPhase(2, cduty);
    }
    // 割り込みフラグクリア
    __HAL_GPIO_EXTI_CLEAR_IT(HALL_SENSOR_C_PIN);
  }
}


void EXTI3_IRQHandler(void) {
  if (__HAL_GPIO_EXTI_GET_IT(HALL_SENSOR_B_PIN) != RESET)
  {
 if (HAL_GPIO_ReadPin(HALL_SENSOR_B_PORT, HALL_SENSOR_B_PIN) == GPIO_PIN_SET)
    {
      // 立ち上がりエッジの処理
         B_Phase_FlagR=1;
      //setPhase(3,cduty);
    }
    else
    {
      // 立ち下がりエッジの処理
        B_Phase_FlagF=1;
      //setPhase(6, cduty);
    }
    // 割り込みフラグクリア
    __HAL_GPIO_EXTI_CLEAR_IT(HALL_SENSOR_B_PIN);
  }
}
```