**LAB NOTE**

**Subject: Hardware/Software Interfacing**

**Lab 5: Stepper Motor**

**Student: Minh Quan Tran**

**Oct 10th, 2024**

Table of Contents

[1. Objectives 4](#_Toc179417403)

[2. Problems and Solutions 5](#_Toc179417404)

[2.1 Problems 5](#_Toc179417405)

[2.2 Solutions 5](#_Toc179417406)

[3. Software Design 6](#_Toc179417407)

[3.1 List of function 6](#_Toc179417408)

[3.2 While loop 8](#_Toc179417409)

**TABLE OF FIGURES**

[Figure 1‑1: Basic outcome 4](#_Toc179417398)

[Figure 1‑2: Intermediatet outcome 4](#_Toc179417399)

[Figure 1‑3: Advance outcome 4](#_Toc179417400)

[Figure 1‑4: Maximum outcome 5](#_Toc179417401)

# Objectives

* Using STM32F411 board and STM32CubeIDE in Windows, create code to

• Write code for the STM32F411 to configure and use Timers.

Basic:

A close up of a text

Description automatically generated

Figure 1‑1: Basic outcome

* Intermediate:

A close up of a computer

Description automatically generated

Figure 1‑2: Intermediatet outcome

* Advance:

A close-up of a computer chip

Description automatically generated

Figure 1‑3: Advance outcome

A computer chip with text and a diagram

Description automatically generated

Figure 1‑4: Maximum outcome

# Problems and Solutions

## Problems

* No problem

## Solutions

# Software Design

## List of function

* This function is to create a delay in micro second by counting the number of ticks has passed using CNT registers.

**void microDelay(uint32\_t usDelay)**

**{**

**// Get the current timer counter value**

**uint32\_t startTime = \_\_HAL\_TIM\_GET\_COUNTER(&htim1);**

**uint32\_t ticks = usDelay - 1; // 1 ticks = 1 us**

**// Poll the CNT register until the specified number of ticks has passed**

**while ((\_\_HAL\_TIM\_GET\_COUNTER(&htim1) - startTime) < ticks)**

**{**

**// Wait until the timer reaches the required ticks**

**}**

**}**

* This function is used to toggle LED whenever the timer overflow

**void HAL\_TIM\_PeriodElapsedCallback(TIM\_HandleTypeDef \*htim)**

**{**

**if (htim->Instance == TIM11)**

**{**

**HAL\_GPIO\_TogglePin(GPIOB, Step\_Pin);**

**static uint32\_t stepCount = 0;**

**stepCount++;**

**#ifdef ADVANCE**

**if (stepCount > 2)**

**{**

**HAL\_TIM\_Base\_Stop\_IT(&htim11);**

**stepCount = 0;**

**runFlag = *DONE*;**

**}**

**#else**

**if (stepCount > abs(numberOfSteps) \* 2)**

**{**

**HAL\_TIM\_Base\_Stop\_IT(&htim11);**

**stepCount = 0;**

**runFlag = DONE;**

**}**

**#endif**

**}**

**}**

* This function is used to initialize the motor

**void** **stepInit**(**void**)

{

HAL\_GPIO\_WritePin(GPIOC,Reset\_Pin,*GPIO\_PIN\_SET*); // High - Normal operation, Low - Reset

HAL\_GPIO\_WritePin(GPIOC,PS\_Pin,*GPIO\_PIN\_SET*); // High - Operation mode, Low - Standby mode

HAL\_GPIO\_WritePin(GPIOA,VRef\_Pin,*GPIO\_PIN\_SET*); // High - 100% current, Low - 0 current

**printf**("Inital Motor \r\n");

}

* This function is used to enable/disable the motor

**void** **stepEnable**(bool enable)

{

**if**(!enable)

{

HAL\_GPIO\_WritePin(GPIOC,OE\_Pin,*GPIO\_PIN\_SET*);

**printf**("Disable Motor\r\n");

}

**else**

{

HAL\_GPIO\_WritePin(GPIOC,OE\_Pin,*GPIO\_PIN\_RESET*);

**printf**("Enable Motor\r\n");

}

**return**;

}

* This function is used to control the number of steps and how long will a step will run in the motor

**void** **stepSteps**(int32\_t numberOfSteps, uint32\_t usDelay)

{

**#ifdef** USE\_INTERRUPT

runFlag = *IN\_PROCESS*;

htim11.Init.Period = usDelay / 2 - 1;

**if** (HAL\_TIM\_Base\_Init(&htim11) != *HAL\_OK*)

{

Error\_Handler();

}

HAL\_TIM\_Base\_Start\_IT(&htim11);

//Wait until finish step

**while**(runFlag == *IN\_PROCESS*);

**#else** // Basic (No interrupt needed)

// Tell which direction the motor is running

**if** (numberOfSteps > 0)

{

HAL\_GPIO\_WritePin(GPIOC,FR\_Pin,GPIO\_PIN\_SET);

}

**else**

{

HAL\_GPIO\_WritePin(GPIOC,FR\_Pin,GPIO\_PIN\_RESET);

}

printf("Stepping Motor\r\n");

**for** (**int** i = 0; i < abs(numberOfSteps); i++)

{

HAL\_GPIO\_WritePin(GPIOB,Step\_Pin,GPIO\_PIN\_SET);

microDelay(usDelay / 2);

HAL\_GPIO\_WritePin(GPIOB,Step\_Pin,GPIO\_PIN\_RESET);

microDelay(usDelay / 2);

}

printf("Finish Stepping\r\n");

**#endif**

}

* This function is used to run the motor using the trapezoidal motion profile

**#ifdef** ADVANCE

**void** **runTrapezoidalMotion**(int32\_t numberOfSteps, uint32\_t usDelay)

{

// Tell which direction the motor is running

**if** (numberOfSteps > 0)

{

HAL\_GPIO\_WritePin(GPIOC,FR\_Pin,*GPIO\_PIN\_SET*);

}

**else**

{

HAL\_GPIO\_WritePin(GPIOC,FR\_Pin,*GPIO\_PIN\_RESET*);

}

numberOfSteps = **abs**(numberOfSteps);

// Ensure number of rampSteps is not greater than half of the total number of steps

int32\_t rampSteps = MIN(RAMP\_STEP, numberOfSteps / 2);

// Middle steps at max speed

int32\_t middleSteps = numberOfSteps - 2 \* rampSteps;

**printf**("Stepping Motor\r\n");

// Acceleration Phase

**for** (int32\_t i = 0; i < rampSteps; i++)

{

// Linearly interpolate delay between max speed and min speed

uint32\_t currentDelay = MAX\_DELAY - (MAX\_DELAY - usDelay) \* i / rampSteps;

stepSteps(1, currentDelay);

}

// Constant Speed Phase

**for** (int32\_t i = 0; i < middleSteps; i++)

{

stepSteps(1, usDelay); //Intended speed

}

// Deceleration Phase

**for** (int32\_t i = 0; i < rampSteps; i++)

{

// Linearly interpolate delay between minUsDelay and maxUsDelay

uint32\_t currentDelay = usDelay + (MAX\_DELAY - usDelay) \* i / rampSteps;

stepSteps(1, currentDelay);

}

**printf**("Finish Stepping\r\n");

}

**#endif**

## While loop

**while** (1)

{

**if**(doPrompt != 0)

{

doPrompt = 0;

**printf**(PROMPT);

}

**switch**(GetCharFromUART2())

{

**case** '0':

enableFlag = !enableFlag;

stepEnable(enableFlag);

**break**;

**case** '1':

numberOfSteps += 400; // Rotate an addition 90o

**printf**("Set steps = %ld, Set delay = %u \r\n",numberOfSteps, delay );

**break**;

**case** '2':

numberOfSteps -= 400; // Reduce 90 degree

**printf**("Set steps = %ld, Set delay = %u \r\n",numberOfSteps, delay );

**break**;

**case** '3':

**if** (delay < MAX\_DELAY)

{

delay += 50;

**printf**("Set steps = %ld, Set delay = %u \r\n",numberOfSteps, delay );

}

**else**

{

**printf**("Delay can't be higher than %duS\r\n", MAX\_DELAY);

}

**break**;

**case** '4':

**if** (delay > MIN\_DELAY)

{

delay -= 50;

**printf**("Set steps = %ld, Set delay = %u \r\n",numberOfSteps, delay );

}

**else**

{

**printf**("Delay can't be lower than %duS\r\n", MIN\_DELAY);

}

**break**;

**case** '5':

**#ifdef** ADVANCE

runTrapezoidalMotion(numberOfSteps,delay);

**#else**

stepSteps(numberOfSteps,delay);

printf("RUNNING\r\n");

**#endif**

**break**;

**case** TIMEOUT\_ERROR:

**printf**("Prompt timeout\r\n");

**break**;

**default**:

**break**;

}

/\* USER CODE END WHILE \*/

/\* USER CODE BEGIN 3 \*/

}

**REFERENCES**