Sophia Garcia
COMPE 375 Lab
Professor Baris Akslani
January 26th, 2024

Lab 1

Questions:

1. What does your program do?
    a. Transforms a character array (a String) into its Morse code format. The output is expressed as dots and dashes from an LED [200 ms for dots and 600 ms for dashes] on the ATMEL ATmega328P.
2. What variables/registers does your program use?
    a. Registers used in the program relate to the LED port on the ATmega328P; B5.
3. What is the main algorithm/logic of your program?
    a. A string of characters is declared and initialized in the main function before the repeating while loop begins running. Within the while-loop starting at index 0, one character at a time is sent to a separate function to determine what its Morse code equivalent is. Arguments of this function include the character in need of translation. The translation is made through ascii translation to two separate arrays. Each letter and number, initialized in a separate header file, are assigned a value correlating with their pattern Morse code value. The value is presented as an octal value to which each 3-bit value represents either a dot or a dash. A dash is valued at 0b011 [3] while a 0b001 [1] is a dot. After returning to the main function in order to execute the light Morse code. In this function the arguments include the current character Morse code equivalent and the following character. In the function a while loop runs iterating per 3 bits of the current Morse value. Starting at the LSB, the loop will end when a value of 0b000 [0] is found in the value sequence after the MSB. Each octal value before the loop ends is then multiplied by the value [200] which is the determinant of the time the light remains on (200 * 1 goes a 200 millisecond dot; 200 * 3 gives a 600 millisecond dash). In between each will be a 200 millisecond delay. At the end based upon the following character there will either be a 400 millisecond pause, but if the next character is a space {ascii value 32), an if-else statement is proven true at the beginning of the function executing a 1400 millisecond delay and skipping the while loop; ending the function execution. This process continues forever because at the bottom of the while loop there is an if-statement proven true when the following character in the sequence is ‘\0’ indicating the end of the string. The executed statement after the if is proven true is relooping the counter variable back to 0 index restarting the string loop.

4. What external hardware connections did you make?
    a. There was no need for any drastic external hardware connections needing to be placed except for the USB to Type-C cable used to upload the computer code (USB) to the ATmega382P microcontroller.

Appendix:

Main Code:

```
/*
 * Lab1_MorseCode.c
 *
 * Created: 1/26/2024 12:44:32 PM
 * Author : Sophia Garcia
 */
#define F_CPU 16000000UL // 16MHz clock from the debug processor
#include <avr/io.h>
#include <util/delay.h>

#include "Morse_codes.h"

void delay_ms_var(uint8_t delay_t)
{
/*
	function initialized in order for there to be a variable
	parameterized delay; current _delay_ms(int time)
	included library function unable to take in a variable
	time to output
*/
	while(0 < delay_t)
	{
		_delay_ms(10);		//delay in batches of 10 seconds
		delay_t -= 10;  //per round 10 seconds taken from input
	}
}

void display(const uint16_t time)
{
	PORTB |= (1<<PORTB5); //Set port bit B5 to 1 to turn on the LED
	delay_ms_var(time); //delay based on input time; 200 ms if dot; 600 ms if dash
	PORTB &= ~(1<<PORTB5); //Clear port bit B5 to 0 to turn off the LED
	_delay_ms(200); //delay : 200ms
}

int letter_val(const uint16_t val)
{
/*
	match the character with their Morse code value either:
		octal value equivalent ['A'-'Z', '0'-'9']
*/
```

```
	uint16_t ret_val = val;
	volatile uint16_t alphabet[27] = {A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T,
U, V, W, X, Y, Z};
	volatile uint16_t numbers[10] = {zero, one, two, three, four, five, six, seven, eight, nine};
	if(val >= 65)
	{
		ret_val = alphabet[val - 65];
	}
	else if(val <= 57 && val != 32)
	{
		ret_val = numbers[val - 48];
	}
	return ret_val;
}

void space_dot_dash(uint16_t val, const uint16_t next)
{
	// 'fourBit_sect' that is taken from val in the form of an octal value [0O] equal to [0bBBB]
	if(val == 32) //if next char a space then print space of 1400 time
	{
		_delay_ms(1200);
		return;
	}
	volatile uint8_t threeBit_sect;
	while(val != 0) //if val == 00 end loop
	{
		threeBit_sect = val & 07; //isolate LMO of val
		display(200 * threeBit_sect);
		// 03 = dash
		// 01 = dot
		val = val >> 3;
		//go to next hex value until val = 00
	}
	if(next != 32)    //if the next character in string not space print space of 400 ms
		_delay_ms(400);
}

int main(void)
{
   DDRB |= (1<<DDB5);
	char sgn[] = "Sophia Garcia 826433407";      //string to be printed
   while (1)
   {
		static uint8_t cnt = 0;  //forever loop counter
```

```
                uint16_t val = sgn[cnt];        //one char from sgn string of pos [cnt]
                uint8_t nxt = sgn[++cnt];       //the following pos char
                if(val >= 'a')
                        val -= ('a' - 'A');     //if lower case val then change to upper
                val = letter_val(val);//change to header file value
                space_dot_dash(val, nxt);//print the Morse code form of character val
                if(nxt == '\0')
                        cnt = 0;                //if string ended restart
    }
}
```

Header File:

```
#ifndef MORSE_CODES
#define MORSE_CODES
/*
        octal values equivalent to the Morse code form
        each octal value is dot or dash:
        dot = 01
        dash = 03
*/

#define A 031
#define B 01113
#define C 01313
#define D 0113
#define E 01
#define F 01311
#define G 0133
#define H 01111
#define I 011
#define J 03331
#define K 0313
#define L 01131
#define M 033
#define N 013
#define O 0333
#define P 01331
#define Q 03133
#define R 0131
#define S 0111
#define T 03
#define U 0311
#define V 03111
#define W 0331
#define X 03113
#define Y 03313
#define Z 01133

#define one     033331
#define two     033311
#define three   033111
#define four    031111
#define five    011111
#define six     011113
#define seven 011133
```

```
#define eight   011333
#define nine    013333
#define zero    033333

#endif
```