AJ Rodrigo
821538725
06-18-2021

Lab 5

What does your program do?

My program is basically a combination of lab 3 and lab 4, where we use GPIO and Timers in creating this program that outputs musical notes based on which button the user pushes. This program is capable of outputting the musical tone for however long the user is pushing the button for. I had to do some hand calculations to populate the array with the OCRA values by using the formula: (440 * 2^(1/12))^x where x was the location of a specific button. Also for this lab, instead of using CTC mode, I utilized the Fast PWM mode as per requested. This is similar to the CTC mode one difference being that the fast PWM can generate waveforms that are going to automatically look for in some output pins.

What variables/registers does your program use?

I declared two integers, i and j, that will keep count of which row/column I am scanning through. This for loop traverses through the 2d array, checking what location a button will be pushed.

I also declared an int data that is a 4x4 2d array. This stores the actual number value of the hand calculations of the frequency in the array to which it will be referenced when the computer knows the location of the pressed button.

My program utilizes multiple registers such as: DDRB, PORTB, PINB, TCCR2A, TCCR2B, OCR2A, OCR2B, TIFR2, OCF2A, OCF2B. Most of these registers are mainly used for the algorithm of the timer. Timer/Counter control registers which allows you to set PWM mode + CTC mode + prescale, output compare register A which is the top value of timer, output compare register B which is the duty cycle of the timer, and timer/counter interrupt flag register that resets the overflow flag of the counter.

As far as using the registers, we have to initialize our DDRD and DDRB as outputs and inputs respectively. Since there is a little complexity with this code, we have to enable the pull-up resistor that will treat the input as if the values are ‘1’ by shifting the port bits to 1.

What is the main algorithm/logic of your program?

My main algorithm of this project lies in the infinite while loop where the program is checking each row and column within a nested for loop to see if any button is pushed. If the button is pushed, the program knows the exact location to which it initializes the OCRA and OCRB

registers to that specific value to output the music tone. Since the prompt requires to maintain a 50% duty cycle throughout the program, we can divide the OCRA value by 2 allowing us to achieve this duty cycle. Within this while loop, we also have buffer conditions for OCRA and OCRB to check for overflow, but since we want to allow the sound to keep playing even if the button remains pushed and if it overflows, I included the statement of playing the sound within the while loop. Last but not least, just like lab 3, we must set the rows back to high after traversing through the nested for loop.

What external hardware connections did you make?

Atmel Board using Micro-USB to computer

The 4x4 Keypad is hooked up to the circuit board where eight wires of equal length are put in their respective pin slots. These wires are found connected to the given port bits on the Lab Description (B0, B1, B2, B3, D4, D5, D6, D7). No external cables are connected to this 4x4 keypad, only the wires that are connected from the 4x4 keypad to the Atmel AVR device.

The sound plug which we used two of the three cables attached to it. I utilized the yellow line to connect to ground on the Atmel board and the red line connected into the PD1 pin of the Atmel board in order for there to be sound output. At the end of this sound plug was an AUX input to which the user had to plug in their own personal audio device in order for sound to transmit out. WIthout plugging anything to the female end of the head, the program would run, but to no audio results.

APPENDIX:

```
/*
 * Lab5 forreal.c
 *
 * Created: 6/18/2021 3:37:10 PM
 * Author : ajrodrigo
 */

#include <avr/io.h>


int main(void)
{
    DDRD |= (1 << DDD4) | (1 << DDD5) | (1 << DDD6) | (1 << DDD7);                  //Setting rows as output
    DDRB &= ~(1 << DDB0) & ~(1 << DDB1) & ~(1 << DDB2) & ~(1 << DDB3);           //Setting columns as input
    PORTB |= (1 << PORTB0) | (1<<PORTB1) | (1 << PORTB2) | (1<<PORTB3); //Enables pull-up resistor
    PORTD |= (1 << 4 | 1 << 5 | 1 << 6 | 1 << 7);                                                  //Set all rows high

            DDRD |= (1 << DDD1);
                    //Setting data direction for speaker functionality

            TCCR2A |= (1 << WGM21) | (1 << WGM20) | (1 << COM2B1);                                  //Set the timer
mode to Fast PWM Timer 2 8 bits
```

```
        TCCR2B |= (1 << WGM22) | (1 << CS22) | (1 << CS21);                                    //Set pre-scaler to
256; Timer/Control Register A & B is part of Mode selection

        int data[4][4] = {{141, 133, 125, 118},                                                 //Hand
calculated OCR2A values by using given formula of frequency and period
                                        {111, 105, 99, 93},
                                        {88, 83, 78, 74},
                                        {70, 66, 62, 58}};

  while (1)
                        //Infinite while loop that checks if keys are pressed and if so, play sound
  {
                for(int i = 4; i < 8; i++)
                {
                        PORTD &= ~ (1 << i);
                //Set row to low
                        for(int j = 0; j < 4; j++)
                        {
                                if(!(PINB & (1 << j)))
                //If button is pushed
                                {
                                        while(!(PINB & (1 << j)))
        //
                                        {
                                                OCR2A = data[i-4][j];
                //Setting our OCR2A value to position in the int array
                                                OCR2B = OCR2A / 2;
                        //Duty cycle of 50% remains constant throughout program

                                                while((TIFR2 & (1 << OCF2B)) == 0)
//Wait for OCR2B overflow event
                                                {
                                                        PORTD |= (1 << 1);
                        //If true, keep playing sound while button is being pushed
                                                }
                                                TIFR2 |= (1 << OCF2B);
                //Reset OCR2B overflow flag

                                                while((TIFR2 & (1 << OCF2A)) == 0)
//Wait for OCR2A overflow event
                                                {
                                                        PORTD &= ~(1 << 1);
                        //If false, stop playing sound when button is released
                                                }
                                                TIFR2 |= (1 << OCF2A);
                //Reset OCR2A overflow flag
                                        }
                                }
                        }
                        PORTD |= (1 << i); //Set row to high

                }
  }
}
```