ΕΡΓΑΣΤΗΡΙΟ ΜΙΚΡΟΕΠΕΞΕΡΓΑΣΤΩΝ

4η ΑΣΚΗΣΗ

28/05/2022

ΠΑΠΑΔΟΠΟΥΛΟΥ ΑΙΚΑΤΕΡΙΝΗ 1067535

ΠΡΑΠΠΑΣ ΤΡΑΝΤΑΦΥΛΛΟΣ 1067504

Πείραμα

Ο κώδικας της άσκησης είναι ο ακόλουθος:

```
#include <avr/io.h>
#include <avr/interrupt.h>
#include <avr/sleep.h>
#include <util/delay.h>
#include <stdlib.h>

#define ped 10

int errorFlag = 0;
int correctDigitCounter = 0;
int digitCounter = 0;
int clockInterrupt = 0;
int adcStart = 1;
int adcInterrupt = 0;
int incorrectPINCounter = 0;
int risingEdgeCounter = 0;
int waiting = 0;
int pwmActive = 0;

void enterPIN(void)
{

	// If the Switch 6 is pressed...
	if(digitCounter == 1 && PORTF.INTFLAGS == 0x40 )
	{
		int intflags = PORTF.INTFLAGS;
		PORTF.INTFLAGS = intflags;
		// Increments the counter by one
		correctDigitCounter++;
	}

	// If the Switch 5 is pressed...
	else if(digitCounter == 2 && PORTF.INTFLAGS == 0x20)
	{
		int intflags = PORTF.INTFLAGS;
		PORTF.INTFLAGS = intflags;

		// Increments the counter by one
		correctDigitCounter++;
	}

	// If the Switch 5 is pressed...
	else if(digitCounter == 3 && PORTF.INTFLAGS == 0x20)
	{
		int intflags = PORTF.INTFLAGS;
		PORTF.INTFLAGS = intflags;

		// Increments the counter by one
		correctDigitCounter++;
	}
	// If the Switch 6 is pressed...
	else if(digitCounter == 4 &&  PORTF.INTFLAGS == 0x40 )
	{
		int intflags = PORTF.INTFLAGS;
		PORTF.INTFLAGS = intflags;

		// Increments the counter by one
		correctDigitCounter++;
```

```
        }

        // If four buttons are pressed...
        if (digitCounter == 4)
        {
                // If the correct password is pressed...
                if(correctDigitCounter == 4)
                {
                        incorrectPINCounter = 0;
                        // Sets the flag to 1
                        errorFlag = 1;
                }
                // Else...
                else
                {
                        // Increments the incorrect pin counter by one
                        incorrectPINCounter++;
                }
        }
}

int main(void)
{
        // LED 0 for PWM and ADC
        PORTD.DIR |= PIN0_bm;

        // Pull-up enable and interrupt enabled with sense on both edges
        PORTF.PIN0CTRL |= PORT_PULLUPEN_bm | PORT_ISC_BOTHEDGES_gc;
        PORTF.PIN1CTRL |= PORT_PULLUPEN_bm | PORT_ISC_BOTHEDGES_gc;
        PORTF.PIN2CTRL |= PORT_PULLUPEN_bm | PORT_ISC_BOTHEDGES_gc;
        PORTF.PIN3CTRL |= PORT_PULLUPEN_bm | PORT_ISC_BOTHEDGES_gc;
        PORTF.PIN4CTRL |= PORT_PULLUPEN_bm | PORT_ISC_BOTHEDGES_gc;
        PORTF.PIN5CTRL |= PORT_PULLUPEN_bm | PORT_ISC_BOTHEDGES_gc;
        PORTF.PIN6CTRL |= PORT_PULLUPEN_bm | PORT_ISC_BOTHEDGES_gc;
        PORTF.PIN7CTRL |= PORT_PULLUPEN_bm | PORT_ISC_BOTHEDGES_gc;

        // Enable interrupts
        sei();

        // Running the program non stop
        while(1)
        {
                // While the password is incorrect....
                while(errorFlag == 0)
                {

                }
                errorFlag = 0;

                _delay_ms(1);

                // Clear counter
                TCA0.SINGLE.CNT = 0;

                // Normal Mode
                TCA0.SINGLE.CTRLB = 0;

                // When reaches this value -> interrupt CLOCK FRENQUENCY/1024
                TCA0.SINGLE.CMP0 = ped;
                TCA0.SINGLE.CTRLA = TCA_SINGLE_CLKSEL_DIV1024_gc;

                // Enable
```

```
		TCA0.SINGLE.CTRLA |=1;

		// Interrupt Enable (=0x10)
		TCA0.SINGLE.INTCTRL = TCA_SINGLE_CMP0_bm;

		// Begin accepting interrupt signals
		sei();

		// While the clockInterrupt interrupt flag is 1...
		while (clockInterrupt == 0)
		{
			// Wait...
		}

		while(adcStart == 0)
		{
			// Single-Running mode enabled
			ADC0.CTRLA = 0;
			ADC0.CTRLA |= ADC_ENABLE_bm;
			ADC0.CTRLA |= ADC_RESSEL_10BIT_gc;

			// Enable Interrupts for WCM
			ADC0.INTCTRL |= ADC_WCMP_bm;
			ADC0.WINLT = 10;

			// Interrupt when RESULT < WINHLT
			ADC0.CTRLE = 0x01;

			// Start Conversion
			ADC0_COMMAND |= ADC_STCONV_bm;
			_delay_ms(5);
		}

		if(adcInterrupt == 1)
		{
			clockInterrupt = 0;

			// Clear counter
			TCA0.SINGLE.CNT = 0;

			// Normal Mode
			TCA0.SINGLE.CTRLB = 0;

			// When reaches this value -> interrupt CLOCK
FRENQUENCY/1024
			TCA0.SINGLE.CMP0 = ped;
			TCA0.SINGLE.CTRLA = TCA_SINGLE_CLKSEL_DIV1024_gc;

			// Enable
			TCA0.SINGLE.CTRLA |=1;

			// Interrupt Enable (=0x10)
			TCA0.SINGLE.INTCTRL = TCA_SINGLE_CMP0_bm;

			// Begin accepting interrupt signals
			sei();

			// While the clockInterrupt interrupt flag is 1...
			while ((clockInterrupt == 0 && incorrectPINCounter < 3) &&
errorFlag == 0)
			{
```

```
                        }

                        errorFlag = 0;

                        if(clockInterrupt == 1 || incorrectPINCounter == 3)
                        {
                                // Sets the PWM to active
                                pwmActive = 1;
                                TCA0.SINGLE.CNT =0; //CLEAR COUNTER
                                TCA0.SINGLE.CTRLB=0; //NORMAL MODE
                                TCA0.SINGLE.CMP0= 10; //
                                TCA0.SINGLE.CTRLA = TCA_SINGLE_CLKSEL_DIV1024_gc;
                                TCA0.SINGLE.CTRLA |=1;
                                TCA0.SINGLE.INTCTRL = TCA_SINGLE_CMP0_bm;

                                while (errorFlag == 0)
                                {
                                }
                                // Clears the error flag
                                errorFlag = 0;
                                // Sets the PWM to inactive
                                pwmActive = 0;
                        }
                }
        }
}


ISR(TCA0_OVF_vect){
        int intflags = TCA0.SINGLE.INTFLAGS;
        TCA0.SINGLE.INTFLAGS = intflags;
        // Opens the LED 0
        PORTD.OUTCLR |= PIN0_bm;
}

ISR(TCA0_CMP0_vect)
{
        // Disable
        TCA0.SINGLE.CTRLA = 0;
        // Clear flags
        int intflags = TCA0.SINGLE.INTFLAGS;
        TCA0.SINGLE.INTFLAGS = intflags;

        if(pwmActive == 1)
        {
                // Closes the LED 0
                PORTD.OUT |= PIN0_bm;
        }
        // Set the pedestrian's interrupt flag to 0
        clockInterrupt = 1;
        adcStart = 0;
}

// ADC..
ISR(ADC0_WCOMP_vect)
{
        int intflags = ADC0.INTFLAGS;
        ADC0.INTFLAGS = intflags;

        // LED forward is on
        PORTD.OUTCLR = PIN0_bm;
```

```
        _delay_ms(5);

        adcStart = 1;
        adcInterrupt = 1;
}



ISR(PORTF_PORT_vect)
{
        digitCounter++;

        enterPIN();

        // Clear the interrupt flag
        int intflags = PORTF.INTFLAGS;
        PORTF.INTFLAGS = intflags;

        if(digitCounter == 4)
        {
                digitCounter = 0;
                correctDigitCounter = 0;
        }

        // Set the waiting interrupt flag to 1
        waiting = 1;
}
```

Για την υλοποίηση του πρώτου ερωτήματος υλοποιήσαμε μια μέθοδο enterPIN() η οποία ελέγχει την θέση και την εγκυρότητα του κουμπιού που πατήθηκε(ο έλεγχος για το αν είναι σωστό το PIN γίνεται αφού πατηθούν και τα 4 πλήκτρα), η οποία καλείται όταν ενεργοποιείται η ISR του PORTF (η ISR κάνει fire λόγω των

PORTF.PIN5CTRL |= PORT_PULLUPEN_bm | PORT_ISC_BOTHEDGES_gc;)

Στο αρχικό στάδιο όσες φορές και να βάλει λάθος κωδικό δεν συμβαίνει κάτι, με το που εισάγει τον σωστό κωδικό βγαίνουμε από την while(errorFlag == 0) και τρέχει ο κώδικας για την ενεργοποίηση του Timer μόλις φτάσουμε στην while(clockInterrupt == 0) τότε κάνει fire η ISR του Timer. Στην συνέχεια αφού ενεργοποιήσουμε τον ADC σε single mode (διότι δεν θέλουμε να τρέχει καθ' όλη την διάρκεια του προγράμματος και για αυτό τον έχουμε μέσα σε while() μέχρι να βάλουμε ένα RES το οποίο να είναι μικρότερο του Threshold που έχουμε ορίσει. Όταν πέσουμε κάτω από το Threshold τότε πυροδοτείται η ISR του ADC. Ύστερα αφού ξανά ορίσουμε τον Timer μέσα στην while ((clockInterrupt == 0 && incorrectPINCounter < 3) && errorFlag == 0) έχουμε 3 περιπτώσεις. Πρώτον μπορούμε να πληκτρολογήσουμε τον σωστό κωδικό όπως στην πρώτη περίπτωση και συνεπώς το errorFlag να γίνει 1 και να αρχίσει το πρόγραμμα από την αρχή. Δεύτερον να εισάγουμε 3 φορές λάθος τον κωδικό με αποτέλεσμα να έχουμε θέμα στο incorrectPINCounter να βγούμε από την while και να πάμε στην if(clockInterrupt == 1 || incorrectPINCounter == 3) στην οποία ενεργοποιείται το PWM το οποίο κάνει fire στις ISR(TCA0_CMP0_vect), ISR(TCA0_OVF_vect) όπου και αναβοσβήνει το LED0, ενώ παράλληλα μας δίνεται η δυνατότητα να πληκτρολογήσουμε τον σωστό κωδικό και να επιστρέψουμε στην αρχική κατάσταση. Τέλος μπορεί να τελειώσει ο Timer που έχουμε θέσει με αποτέλεσμα να καταλήξουμε πάλι στην if(clockInterrupt == 1 || incorrectPINCounter == 3) και να αρχίσει να τρέχει το PWM μέχρι να βάλουμε τον σωστό κωδικό και να γυρίσουμε στην αρχή του προγράμματος.