ΑΝΑΦΟΡΑ ΠΡΩΤΟΥ MILESTONE

(HPY411)

Κωνσταντίνος Κασφίκης - (2013030108)

Λουκόπουλος Γεώργιος - (2013030133)

Για την ολοκλήρωση του πρώτου milestone, ήταν απαραίτητη η ενασχόληση μας με το περιβάλλον ανάπτυξης (στην περίπτωση μας Atmel Studio 7). Μέσω αυτού καταφέραμε την επικοινωνια της πλακέτας STK500, με τον υπολογιστή, με σκοπό τον προγραμματισμό της μνήμης flash του μικροελεγκτή ATMEGA16L. Η επικοινωνία επιτεύχθηκε, με τη χρήση ενός αντάπττρα RS232 σε USB, έπειτα απο την εγκατάσταση κατάλληλων drivers για τον αντάπτορα.

Ζητούμενο του πρώτου milestone, ήταν η δημιουργία ενός προγράμματος, το οποίο περιλαμβάνει τη χρήση βασικών εννοιών, όσον αφορά τον προγραμματισμό AVR όπως:

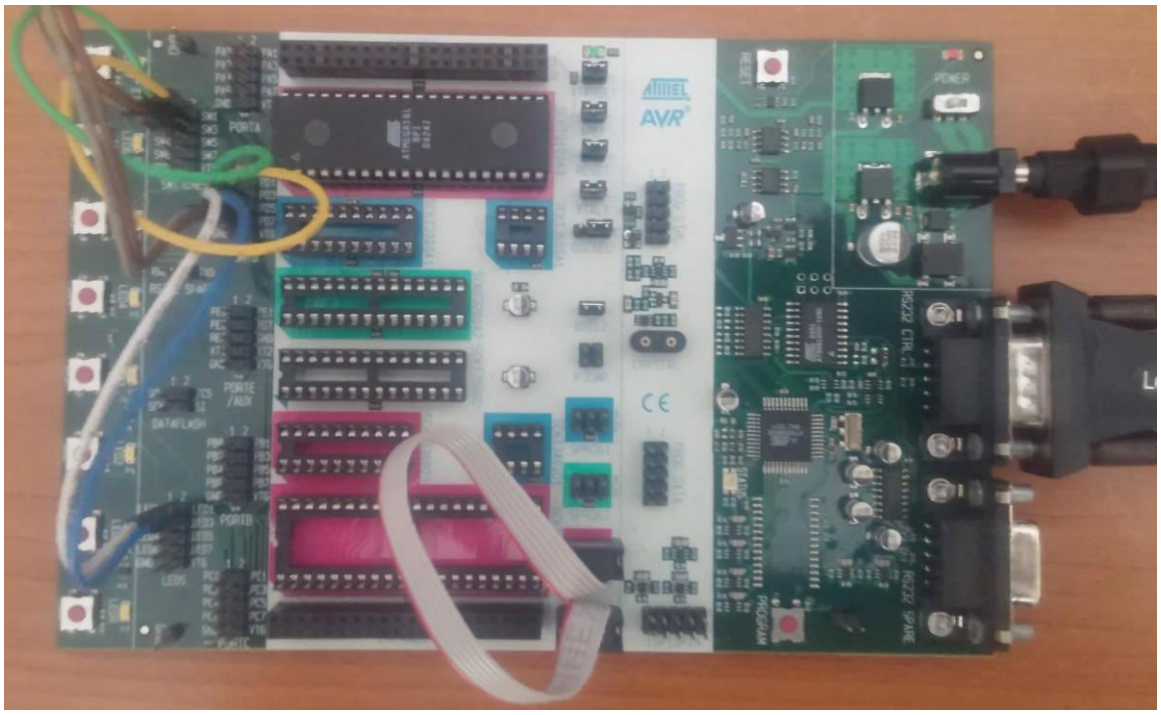
- *Interrupts*
- *I/O*
- *Timers/Counters*
- *Simulation/Download*

Η άσκηση που υλοποιήσαμε είναι, αυτή που παρουσιάσατε στην ανακοίνωση σας.

Περιγραφη Ασκησης

- ### Συνδεσμολογία Πλακέτας:

Συνδέουμε τρία switches της πλακέτας STK500, με τρία pins που αντιστοιχούν σε τρείς εισόδους του μικροελεγκτή ATMEGA16L. Συγκεκριμένα συνδέουμε το switch 0 και 1 με τα pins PORTD2 και PORTD3 αντίστοιχα (Τα pins αυτά αντιστοιχούν επίσης στα interrupts 0 και 1). Επίσης συνδέουμε το switch 3 με την PORTD6. Ακόμα, συνδέουμε τα leds 0 και 1 με τις θύρες PORTD4 και PORTD5. Τόσο τα led, όσο και τα switches είναι ενεργά στο 0.


ATMEL
AVR
POWER
RESET
PROGRAM
RS232 CTRL
RS232 SPARE
PORTA
PORTB
PORTE
/AUX
DATAFLASH
LEDS

- **Κώδικας**

```c
#include <avr/io.h>
#include <avr/interrupt.h>
#define F_CPU 1000000
#define Interrupt_SW1 2
#define Interrupt_SW2 3
#define UserInput_SW3 6
#define LED_1 4
#define LED_2 5

int check;

int main(void)
{
    GICR |= (1<<INT0);
    GICR |= (1<<INT1);
    MCUCR = 0xF;

    DDRD &= ~(1<<Interrupt_SW1);
    DDRD &= ~(1<<Interrupt_SW2);
    DDRD &= ~(1<<UserInput_SW3);
    DDRD |= (1<<LED_1);
    DDRD |= (1<<LED_2);

    sei();
    outputHigh(LED_1);
    outputHigh(LED_2);

    while (1)
    {
    }
}
```

Στο παραπάνω κομμάτι κώδικα, δηλώνουμε ποιές θύρες θα χρησιμοποιήσουμε, από το PORTD του μικροελεγκτή, για εισόδους-εξόδους, ενεργοποιούμε τα Global Interrupts καλώντας της sei() και αρχικοποιούμε την τιμή των δύο εξόδων σε 1 (καθώς τα LED της πλακέτας είναι ενεργά στο 0) καλώντας την συνάρτηση outputHigh(). Έπειτα, καθορίζουμε μία ατέρμονη while ώστε ο controller να παραμένει σε λειτουργία χωρίς όμως να εκπονεί κάποια εργασία. Ουσιαστικά η εκτέλεση, πέραν των αρχικών δηλώσεων, γίνεται με την χρήση των Interrupt Service Routines.

Συγκεκριμένα, οι αρχικές εντολές που θα εκτελεστούν μόνο κατά την έναρξη του μικροελεγκτή, περιλαμβάνουν την ενεργοποίηση των Interrupts 0 και 1 θέτοντας τιμή στα αντίστοιχα πεδία του καταχωρητή GICR και θέτοντας τιμή στα τέσσερα τελευταία πεδία του MCUCR , με σκοπό να ορίσουμε ότι τα Interrupts θα ενεργοποιούνται με Rising Edge. Έπειτα, θέτουμε τιμή στον καταχωρητή DDRD στα αντίστοιχα πεδία με τα pin/ports που χρησιμοποιούμε ως εισόδους και εξόδους.

```c
void outputHigh(int output){
    PORTD |= (1<<output);
}

void outputLow(int output){
    PORTD &= ~(1<<output);
}
```

Δηλώνοντας τις παραπάνω συναρτήσεις, καθορίζουμε την τιμή των εξόδων, που δεικνύεται από τα ορίσματα αυτών, του PORTD σε 0 χρησιμοποιώντας την outputLow() και 1 χρησιμοποιώντας την outputHigh().

```
ISR(INT0_vect){
    if( (PIND&(1<<UserInput_SW3)) == 0){
        check = 1;
        waitForSecs(60634); //5secs

    }
    else{
        check = 0;
        waitForSecs(64555); //1sec

    }
}

ISR(INT1_vect){
    outputHigh(LED_1);
    outputHigh(LED_2);
}

ISR(TIMER1_OVF_vect){
    if(check == 0){
        outputLow(LED_1);
    }
    else{
        outputLow(LED_2);
    }
}
```

Ορίζουμε τρία Interrupt Service Routines τα οποία αφορούν τα INT0,INT1 και TIMER1_OVF.

Συγκεκριμένα:

* INT0 ISR : η ρουτίνα αυτή εκτελείται μόλις προκύψει rising-edge στην είσοδο PORTD2(switch0). Αρχικά, ελέγχει αν η είσοδος του χρήστη είναι 0, η 1 (switch 3) και θέτει αντίστοιχα την τιμή της μεταβλητής check ενώ, στην συνέχεια, καλεί την waitForSecs για 5 δευτερόλεπτα στην περίπτωση που το switch δεν είναι πατημένο αλλιώς για 1 δευτερόλεπτο.

*INT1 ISR : η ρουτίνα αυτή εκτελείται μόλις προκύψει rising-edge στην είσοδο PORTD2(switch1). Κατά την εκτέλεση της θέτει τις εξόδους σε 1, απενεργοποιώντας έτσι τα LED0 και LED1.

*TIMER_OVF ISR :

Ο timer1 του AVR αρχικοποίειται ,με την κλήση της συνάρτησης waitForSecs(). Ο timer ξεκινά ορίζοντας τιμή 101 στα τρία πρώτα πεδία του καταχωρητή TCCR1B, δηλώνοντας έτσι οτι η τιμή του timer θα αυξάνεται κάθε 1024 κύκλος του ρολογιού του μικροελεγκτή.

Το ρολόι του μικροελεγκτή τρέχει στο F_CPU = 1MHz (= 1000000Hz). Διαιρώντας Ftick = F_CPU/1024 = 977Hz ,βρίσκουμε την συχνότητα με την οποία αυξάνεται η τιμή του timer.Επομένως, η τιμή της

περιόδου του timer είναι Ttick = 1/Ftick = 1ms. Ο χρόνος που αποσκοπούμε να μετρήσουμε με τον timer είναι T1=1s και T2=5s.Προκειμένου, να βρούμε πόσα ticks απαιτούνται για την εκάστοτε χρονική τιμή, αρκεί να διαιρέσουμε την Ttick(Nticks(T1) = 1000, Nticks(T2) = 5000). Ο timer1 είναι ένας 16-bit timer,οπότε η μέγιστη τιμή που μπορεί να δεκτεί είναι Max=65535. Αρχικοποιούμε τον timer σε τιμή Max - Nticks (καταχωρητής TCNT1), ώστε να προκύψει overflow μετά απο Nticks, . Μόλις προκύψει overflow επιδιώκουμε να ενεργοποιηθεί ο κατάλληλος ISR. Προκειμένου να ενεργοποιήσουμε το interrupt αυτό θέτουμε καταλλήλως το πεδίο στην θέση TOIE1 του καταχωρητή Timer Interrupt Mask (TIMSK).

```
void waitForSecs(uint32_t ticks){
    TCCR1B = 0x05; //mode
    TCNT1 = ticks;
    TIMSK |= (1<<TOIE1); //Enable Overflow Interrupt
}
```

Όσον αφορά, τον ISR του TIMER1_OVF, αναλόγως με την τιμή που έχει η μεταβλητή check ενεργοποιούμε είτε το LED0 ή το LED1. Η ρουτίνα αυτή εκτελείται μόλις προκύψει overflow στον timer1.

Συνολικά ο κώδικας :

```c
#include <avr/io.h>
#include <avr/interrupt.h>
#define F_CPU 1000000
#define Interrupt_SW1 2
#define Interrupt_SW2 3
#define UserInput_SW3 6
#define LED_1 4
#define LED_2 5

int check;

int main(void)
{
    GICR |= (1<<INT0);
    GICR |= (1<<INT1);
    MCUCR = 0xF;

    DDRD &= ~(1<<Interrupt_SW1);
    DDRD &= ~(1<<Interrupt_SW2);
    DDRD &= ~(1<<UserInput_SW3);
    DDRD |= (1<<LED_1);
    DDRD |= (1<<LED_2);

    sei();
    outputHigh(LED_1);
    outputHigh(LED_2);

    while (1)
    {
    }
}

void outputHigh(int output){
    PORTD |= (1<<output);
}

void outputLow(int output){
    PORTD &= ~(1<<output);
}

void waitForSecs(uint32_t ticks){
    TCCR1B = 0x05; //mode
    TCNT1 = ticks;
    TIMSK |= (1<<TOIE1); //Enable Overflow Interrupt
}
```

```
ISR(INT0_vect){
    if( (PIND&(1<<UserInput_SW3)) == 0){
        check = 1;
        waitForSecs(60634); //5secs

    }
    else{
        check = 0;
        waitForSecs(64555); //1sec

    }
}

ISR(INT1_vect){
    outputHigh(LED_1);
    outputHigh(LED_2);
}

ISR(TIMER1_OVF_vect){
    if(check == 0){
        outputLow(LED_1);
    }
    else{
        outputLow(LED_2);
    }
    TCCR1B = 0x00;
}
```