Εργαστηριακή Άσκηση 2

MICROCHIP STUDIO

Διάρθρωση Παρουσίασης

- Σκοπός
- Περιγραφή τριών παραδειγμάτων
- Παράδειγμα 1 και Υλοποίηση
- Παράδειγμα 2 και Υλοποίηση
- Παράδειγμα 3 και Υλοποίηση
- Επεξήγηση εργαστηριακής άσκησης

Σκοπός

Ο σκοπός της άσκησης αυτής είναι η εξοικείωσή σας με:

- το Simulation και Debugging στο Microchip Studio
- τη διαχείριση των pins εισόδων/εξόδων του AVR-IoT Development Board
- τη μονάδα διαχείρισης διακοπών (interrupts)
- τη χρήση των μετρητών του συστήματος (Timer/Counter)

Περιγραφή παραδειγμάτων

- Στο πρώτο παράδειγμα θα υλοποιηθεί ένα σύστημα το οποίο ανάβει και σβήνει ένα LED με συχνότητα 10 ms.
- Στο δεύτερο παράδειγμα θα υλοποιηθεί ένα σύστημα το οποίο ενεργοποιεί τη μονάδα διαχείρισης interrupt με το πάτημα ενός switch.
- Στο τρίτο παράδειγμα θα υλοποιηθεί ένας timer που οδηγεί σε interrupt.

Παράδειγμα 1

ΛΕΙΤΟΥΡΓΙΑ ΕΝΟΣ LED

Data Direction

- Για να μπορέσουμε να αναβοσβήνουμε το LED με συχνότητα 10ms πρέπει πρώτα να ορίσουμε το Direction του συγκεκριμένου Pin ως Output.
- Αρχικά, το PORT που θα χρησιμοποιήσουμε είναι το PORTD καθώς τα τέσσερα πρώτα PINs του συνδέονται με LEDs πάνω στο Board μας (όπως φαίνεται και στο Board Overview).
- Για να ορίσουμε ένα PIN ως Output πρέπει να θέσουμε το 5ο bit του register DIR (Data Direction) με 1.
- Τώρα μπορούμε να δώσουμε την τιμή 0 ή 1 στον register Out (Output Value) και να ανάβουμε ή να σβήνουμε το LED αντίστοιχα.

Σημείωση: Στο Simulation οι χρόνοι του delay όπως και των timers (που θα δούμε στη συνέχεια) δεν είναι αντιπροσωπευτικοί του χρόνου που θα περιμένετε στο σπίτι. Χρειάζεται την υλοποίηση στην πλακέτα για να δούμε τους πραγματικούς χρόνους. Για αυτό προς το παρόν βάζουμε μικρούς χρόνους για να μην περιμένουμε πολύ.

Υλοποίηση

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

#define del 10

int main(void){
	PORTD.DIR |= PIN1_bm; //PIN is output
	PORTD.OUT |= PIN1_bm; //LED is off
	while (1) {
		PORTD.OUTCLR= PIN1_bm; //on
		_delay_ms(del); //wait for 10ms
		PORTD.OUT |= PIN1_bm; //off
		_delay_ms(del); //wait for 10ms
	}
}
```

- Πάρτε τον κώδικα και κάντε Simulation στο Microchip Studio.
- Ανοίξτε το I/O παράθυρο (Debug → Windows → I/O) και ξεκινήστε το Debugging.
- Εκτελέστε βήμα βήμα τις εντολές (η συνάρτηση delay δεν εκτελείται βηματικά) και παρατηρήστε τι τιμές έχουν τα registers του PORTD.
- *Σημείωση: Για να δείτε τις τιμές των δηλωμένων σταθερών PIN1_bm και άλλων που θα δούμε στη συνέχεια ανατρέξτε στο header file iom4808.h το οποίο βρίσκεται στον φάκελο που έχετε εγκαταστήσει το Microchip*
- "~\Studio\7.0\packs\atmel\ATmega_DFP\1.6.364\include\avr\iom4808.h"

Παράδειγμα 2

INTERRUPT

Ενεργοποίηση Interrupt σε switch

- Τα switchs που μπορούν να χρησιμοποιηθούν είναι στο PORTF και είναι τα PIN5 και PIN6 (ανατρέξτε στο Board Overview).
- Για τη χρήση ενός switch πρέπει να είναι ενεργοποιημένο το Pullup enable bit που του αντιστοιχεί (ανατρέξτε σελ. 158 στο ATmega4808 DataSheet).
- Επίσης, πρέπει να επιλέξουμε σε ποιο σημείο του παλμού θα ενεργοποιηθεί η μονάδα διαχείρισης του interrupt. Εδώ επιλέγουμε την ενεργοποίησή της και στις δύο άκρες του παλμού (ανατρέξτε σελ. 158 στο ATmega4808 DataSheet).
- Με την ενεργοποίηση των συγκεκριμένων bits του PIN5 μπορεί το σύστημα να δεχτεί interrupt όταν πατηθεί το switch.

ISR Routine

- Όταν γίνει το interrupt πρέπει οδηγηθούμε σε μία συνάρτηση η οποία θα το διαχειριστεί.
- Αυτή η συνάρτηση είναι μια ISR Routine η οποία παίρνει ως όρισμα το ένα interrupt vector το οποίο αντιστοιχεί σε μία διακοπή.
- Για παράδειγμα το interrupt vector για τη διακοπή του PINF είναι το `PORTF_PORT_vect`. Στο αρχείο iom4808.h υπάρχουν όλα τα interrupt vectors που μπορούν να χρησιμοποιηθούν από το board μας.

Υλοποίηση

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

#define del 10
int interr=0; //logic flag

int main() {
     PORTD.DIR |= PIN1_bm; //PIN is output
     PORTD.OUT |= PIN1_bm; //LED is off
//pullup enable and Interrupt enabled with sense on both edges
     PORTF.PIN5CTRL |= PORT_PULLUPEN_bm | PORT_ISC_BOTHEDGES_gc;
     sei(); //enable interrupts
     while (interr==0) {
          PORTD.OUTCLR= PIN1_bm; //on
          _delay_ms(del);
          PORTD.OUT |= PIN1_bm; //off
          _delay_ms(del);
     }
cli(); //disenable interrupts
}
```

```
ISR(PORTF_PORT_vect){

//clear the interrupt flag

int intflags = PORTF.INTFLAGS;

PORTF.INTFLAGS=intflags;

interr=1;

}
```

Λειτουργία κώδικα

- Στο σύστημά μας έχουμε ένα LED και ένα Switch.
- Η κανονική λειτουργία ορίζει το LED να αναβοσβήνει με συχνότητα 10ms.
- Όταν πατηθεί το switch ενεργοποιείται το interrupt και αλλάζω την τιμή της μεταβλητής **interr** ώστε να σταματήσει να αναβοσβήνει το LED και να σταματήσει η λειτουργία.

Simulation

- Για να ενεργοποιήσετε το interrupt πρέπει να πατήσετε το 5ο bit του register INTFLAGS στο PORTF, εφόσον το πρόγραμμα είναι σε ένα breakpoint της επιλογής σας.
- Μόλις αλλάξετε το bit από ‘0’ σε ‘1’ και πατήστε το πρόγραμμα να πάει στην επόμενη εντολή (STEP OVER), θα ενεργοποιηθεί το interrupt routine και θα δείτε ότι βρισκόμαστε πλέον στη συνάρτηση αυτή.

Παράδειγμα 3

TIMER

Ενεργοποίηση Timer

Για να λειτουργήσει ο timer TCA0 σε κανονική λειτουργία και να κάνει interrupt όταν φτάσει μία προβλεπόμενη τιμή πρέπει:

- Να δώσουμε την τιμή '0' στον CTRLB register (Normal Mode)
- Να δώσουμε την τιμή '0' στον CNT register (θέτουμε τον timer στο μηδέν)
- Να δώσουμε την προβλεπόμενη τιμή στον CMP0 register
- Να θέσουμε το clock frequency και να τον ενεργοποιήσουμε μέσω του CTRLA register
- Τέλος, να ενεργοποιήσουμε τα interrupts μέσω του INTCTRL register

Σημείωση: Ανατρέξτε στη σελ. 243 στο ATmega4808 DataSheet

ISR Routine

- Ο counter θα αρχίσει να μετράει και όταν θα φτάσει την τιμή που θέσαμε θα ενεργοποιηθεί η ISR routine με όρισμα το vector `TCA0_CMP0_vect` .
- Αυτό είναι το interrupt vector που έχει οριστεί για το συγκεκριμένο timer.

Υλοποίηση

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

#define ped 20

int interr=0;

int main() {
  PORTD.DIR |= PIN1_bm; //PIN is output
  PORTD.OUTCLR= PIN1_bm; //LED is on

//(σελ 219, 224, 205) 16-bit counter high and low
  TCA0.SINGLE.CNT = 0; //clear counter
  TCA0.SINGLE.CTRLB = 0; //Normal Mode (TCA_SINGLE_WGMODE_NORMAL_gc  σελ 207)
  TCA0.SINGLE.CMP0 = ped; //When reaches this value -> interrupt CLOCK FREQUENCY/1024
  TCA0.SINGLE.CTRLA = TCA_SINGLE_CLKSEL_DIV1024_gc; //(= 0x7<<1 σελ 224 )
  TCA0.SINGLE.CTRLA |=1;//Enable
  TCA0.SINGLE.INTCTRL = TCA_SINGLE_CMP0_bm; //Interrupt Enable (=0x10)
  sei(); //begin accepting interrupt signals
  while (interr==0) {
  }
  PORTD.OUT |= PIN1_bm; //LED is off
  cli();
}
```

```
ISR(TCA0_CMP0_vect){

    TCA0.SINGLE.CTRLA = 0; //Disable
    //clear flag
    int intflags = TCA0.SINGLE.INTFLAGS;
    TCA0.SINGLE.INTFLAGS=intflags;
    interr=1;
}
```

Λειτουργία κώδικα

- Ανάβω το LED και ενεργοποιώ τον timer.

Όταν ο timer φτάσει την τιμή που έχω θέσει:

- Ενεργοποιείται η ρουτίνα διαχείρισης της διακοπής
- Αλλάξω τη μεταβλητή **interr**
- Το πρόγραμμα τελειώνει σβήνοντας το LED

Σημείωση. Μπορείτε να χρησιμοποιείτε τα breakpoints για να δείτε αν ένα interrupt ενεργοποιείται.

Πως θέτω την τιμή του CMP0;

- Η τιμή του καταχωρητή CMP0 (value) ορίζει τον χρονικό διάστημα μέχρι να κάνει interrupt ο timer (γενικά να τελειώσει και να αρχίσει από την αρχή).
- Ο τύπος για τον υπολογισμό είναι ο παρακάτω:

$$f_{timer} = \frac{f_{system}}{N_{prescaler}} \qquad value = T * f_{timer}$$

- Ο ATmega4808 λειτουργεί σε μέγιστο 20MHz και το $N_{prescaler}$ παίρνει την τιμή που έχουμε ορίσει εμείς για clock frequency.
- Για παράδειγμα στον κώδικά μας δώσαμε τιμή 20 άρα ο χρόνος του timer είναι:

$$f_{timer} = \frac{20MHz}{1024} = 19.531{,}25\ Hz \qquad T = \frac{value}{f_{timer}} = 1{,}024\ ms$$

Εργαστηριακή Άσκηση

TRAFFIC LIGHTS

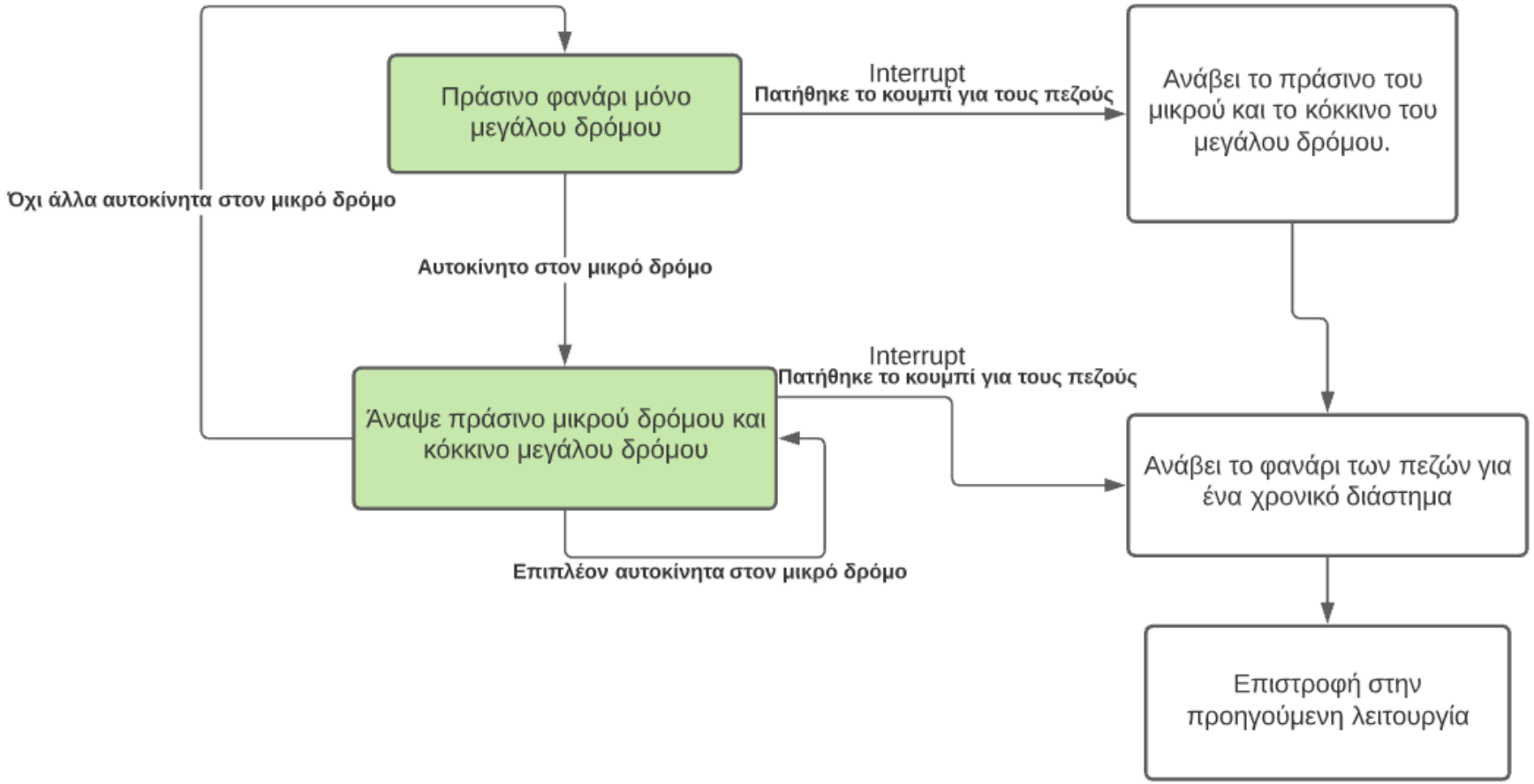
Περιγραφή

- Σκοπός είναι να προσομοιώσουμε τη λειτουργία μίας διασταύρωσης, η οποία αποτελείται από ένα μεγάλο δρόμο και έναν μικρότερο.
- Στον μεγάλο δρόμο υπάρχει το φανάρι για τα αυτοκίνητα και το φανάρι για τους πεζούς που ανάβει μόνο μετά από πίεση κουμπιού.
- Στον μικρό δρόμο υπάρχει ένας αισθητήρας (που εδώ θα υλοποιηθεί με τη συνάρτηση random) ο οποίος όταν εντοπίσει ότι υπάρχει κάποιο αυτοκίνητο που περιμένει, στέλνει εντολή να ανάψει το κόκκινο για τον μεγάλο δρόμο και το πράσινο για αυτόν. Αυτός ο δρόμος δεν έχει έξυπνο φανάρι για πεζούς.
- Όταν πατηθεί το κουμπί των πεζών, ανάβει το κόκκινο φανάρι για τον μεγάλο δρόμο, ανάβει το πράσινο για τον μικρό δρόμο (αν είναι ήδη αναμμένο το αφήνουμε όπως έχει) και ανάβει το πράσινο φανάρι για ένα συγκεκριμένο χρονικό διάστημα.

Παραδοχές

- Το φανάρι είναι πράσινο όταν το LED είναι ανοιχτό και κόκκινο όταν είναι σβηστό. Τα τρία PIN του PORTD που χρησιμοποιείται είναι τα PIN0, PIN1 και PIN2.
- Ο αισθητήρας να προσομοιωθεί με τη συνάρτηση random, δηλαδή όταν η random δώσει έναν τυχαίο αριθμό που τελειώνει σε 0, 5 ή 8 να θεωρείται πως υπάρχει αυτοκίνητο στον μικρό δρόμο. Δεν χρειάζεται να γίνει κάποιο interrupt, με μια απλή if αρκεί.
- Με interrupt πρέπει να υλοποιηθεί το πάτημα του κουμπιού των πεζών. Η χρήση του PIN5 του PORTF θα χρησιμοποιηθεί.
- Με timer και interrupt πρέπει να υλοποιηθεί ο χρόνος στον οποίο είναι αναμμένο το φανάρι των πεζών. Ορίστε εσείς έναν εικονικό χρόνο που σας βολεύει στην προσομοίωση αλλά εξηγήστε τον τρόπο σκέψης και την διαδικασία του υπολογισμού του value.

Διάγραμμα ροής



Παραδοτέα

- Παραδίδεται αναλυτική αναφορά με την λειτουργία του κώδικά σας.
- Επίσης, παραδίδεται τον κώδικά σας με αναλυτικά σχόλια.