Τμήμα Μηχανικών Η/Υ και Πληροφορικής
Computer Engineering and Informatics Department (CEID)
www.ceid.upatras.gr

Εργαστήριο Προηγμένοι Μικροεπεξεργαστές

Εργαστηριακές Ασκήσεις

Πάτρα, 2021-22

1. Εισαγωγικά

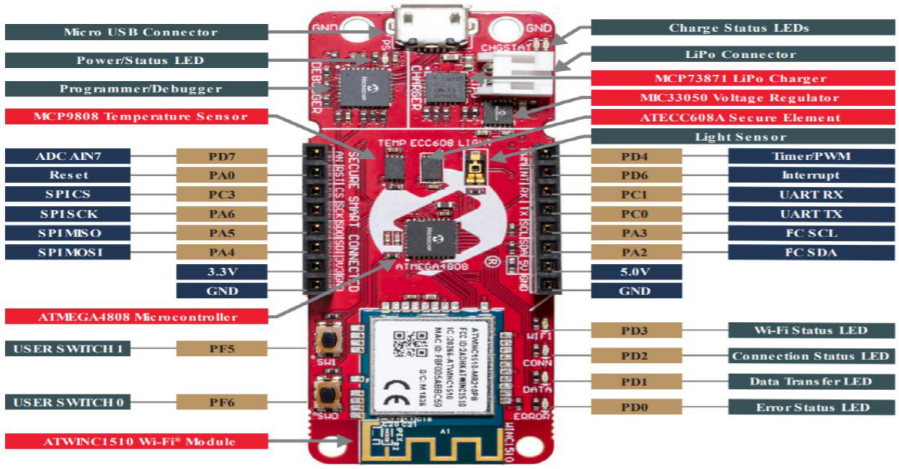
Η AVR είναι μια οικογένεια μικροεπεξεργαστών που αναπτύχθηκε αρχικά από την Atmel και μετέπειτα αγοράστηκε από την Microchip Technology. Βασίζεται σε μια τροποποιημένη Harvard αρχιτεκτονική, 8-bit RISC single-chip μικροεπεξεργαστή. Η AVR είναι μια από τις πρώτες οικογένειες μικροεπεξεργαστών που χρησιμοποίησαν on-chip flash memory, για αποθήκευση προγραμμάτων, σε αντίθεση με μια προγραμματιζόμενη ROM, EPROM ή EEPROM που χρησιμοποιούνταν από άλλους μικροελεγκτές. Οι μικροεπεξεργαστές της οικογένειας αυτής χρησιμοποιούνται σε πολλές εφαρμογές, ως πλήρη ενσωματωμένα συστήματα. Αξιοποιούνται ως εκπαιδευτικές ενσωματωμένες διατάξεις και έγιναν διάσημοι λόγω της ευρείας χρήσης τους, σε πολλά open hardware development boards, της σειράς Arduino.

Το Microchip Studio (ή Atmel® Studio 7) είναι μια ολοκληρωμένη πλατφόρμα ανάπτυξης (Integrated Development Platform - IDP), για την δημιουργία εφαρμογών με AVR και SAM microcontrollers (MCU). Η πλατφόρμα προσφέρει εύχρηστο περιβάλλον για τη σύνταξη, την κατασκευή και τον εντοπισμό σφαλμάτων εφαρμογών που είναι γραμμένα σε C/C ++ ή/και se Assembly. Επίσης, παρέχεται η δυνατότητα εισαγωγής σχεδίων σε Arduino ως C/C++ projects και υποστηρίζονται 500+ AVR και SAM διατάξεις. Τέλος, το Microchip Studio περιλαμβάνει μια τεράστια βιβλιοθήκη πηγαίου κώδικα με 1600+ παραδείγματα εφαρμογών. Για περισσότερες πληροφορίες επισκεφτείτε την σελίδα της Microchip, στην οποία καταγράφονται αναλυτικά όλες οι δυνατότητες του Studio.

2. Το AVR-IoT Wx Development Board

Το AVR-IoT Wx Development Board είναι μια ευέλικτη και εύκολα επεκτάσιμη πλατφόρμα δημιουργίας και ανάπτυξης εφαρμογών. Βασίζεται στην αρχιτεκτονική μικροελεκτή AVR που χρησιμοποιεί τεχνολογία Wi-Fi. Συνοπτικά, μια αντιστοιχη εφαρμογή που αναπτύσσεται στο συγκεκριμένο σύστημα αποτελείται από τρία βασικά blocks:

1. ATmega4808 Microcontroller.
2. ATECC608A Secure Element.
3. ATWINC1510 Wi-Fi Controller Module.



Σχήμα 0.1 : AVR Development Board

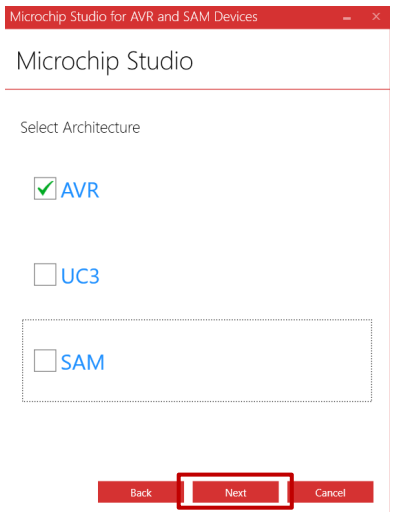
3. Εγκατάσταση Microchip Studio

Το Microchip Studio μπορείτε να το βρείτε διαθέσιμο στον παρακάτω σύνδεσμο:
https://www.microchip.com/mplab/microchip-studio

Η εγκατάσταση του γίνεται εύκολα, ακολουθώντας τα παρακάτω βήματα:

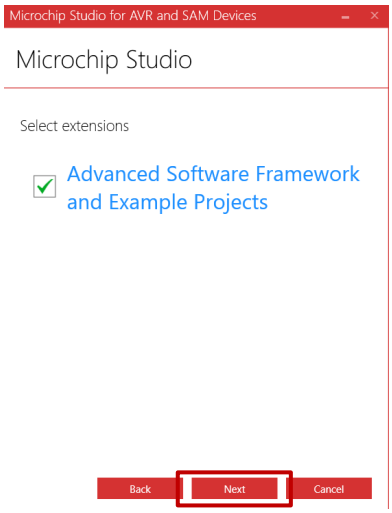
1. Πατήστε Download στην επιλογή Microchip Studio for AVR and SAM Devices 7.0.2542 Web Installer.
2. Αφού ολοκληρωθεί η παραπάνω διαδικασία, ανοίξτε το αρχείο και περιμένετε.
   - Στο παράθυρο που θα εμφανιστεί επιλέξτε το path στο οποίο θα αποθηκευτεί το Studio και επιλέξτε το κουτί “I agree to the Terms and Conditions”.
   - Στη συνέχεια επιλέξτε την επιλογή “Next”.
3. Στο επόμενο παράθυρο, έχετε την επιλογή να εγκαταστήσετε τρεις διαφορετικές οικογένειες που υποστηρίζονται από το Studio:
   - Το πρώτο είναι το AVR, το οποίο θα πρέπει να παραμείνει επιλεγμένο καθώς με αυτό θα ασχοληθούμε.
   - Το UC3 και το SAM αποτελούν άλλα είδη αρχιτεκτονικών και δεν είναι απαραίτητο να τα επιλέξετε για εγκατάσταση.

   Αφού σιγουρευτείτε ότι επιλέξατε το AVR επιλέξτε Next.



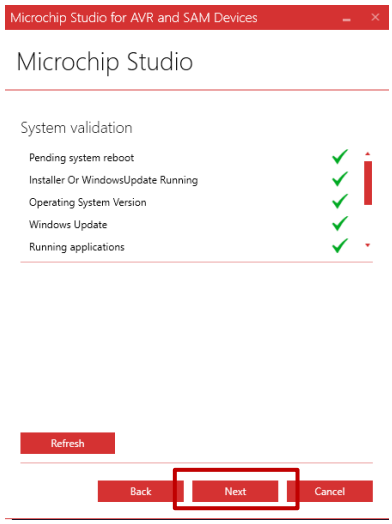
Σχήμα 0.2 : Παράθυρο Επιλογών AVR και SAM Devices

4. Advanced Software Framework (ASF):
   Το ASF παρέχει ένα πλούσιο σύνολο λειτουργικών drivers και code modules που αναπτύχθηκαν από ειδικούς για τη μείωση του χρόνου σχεδιασμού. Το ASF είναι μια δωρεάν βιβλιοθήκη ανοιχτού κώδικα, που έχει σχεδιαστεί για να χρησιμοποιείται στις φάσεις αξιολόγησης, πρωτοτυπίας, σχεδιασμού και παραγωγής. Μπορείτε να το κρατήσετε επιλεγμένο και στον ελεύθερο χρόνο σας να δείτε τα παραδείγματα. Στη συνέχεια, μπορείτε να επιλέξετε “Next”.



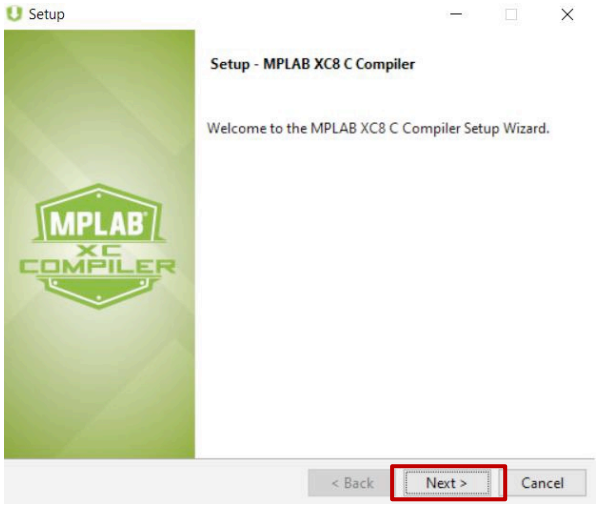
Σχήμα 0.3 : Παράθυρο Επιλογής ASF

5. Εφόσον όλες οι επιλογές του Validation είναι επιλεγμένες, είστε έτοιμοι στη συνέχεια να επιλέξετε “Next”. Στο επόμενο παράθυρο μπορείτε να επιλέξετε το “Install” και να περιμένετε να εγκατασταθεί το Microchip Studio και τα υπόλοιπα εργαλεία που χρειάζονται.

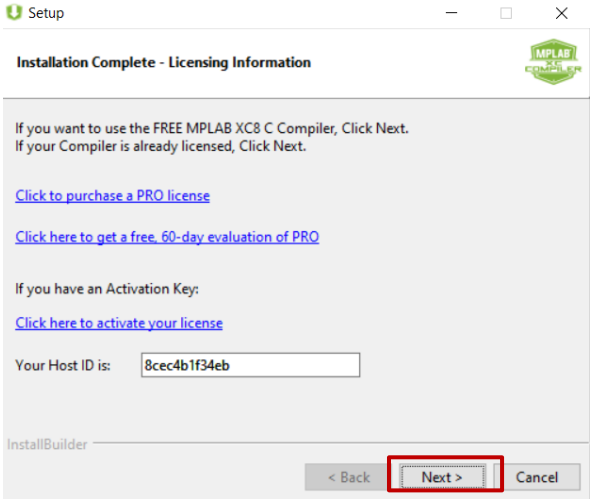


Σχήμα 0.4 : Παράθυρο Επιλογής System Validation

6. Κατά τη διάρκεια της εγκατάστασης, θα εμφανιστεί ένα επιπλέον παράθυρο επιλογών, για την εγκατάσταση του MPLAB XC8 C Compiler, με τον οποίο προσφέρεται η δυνατότητα βελτιστοποίησης του κώδικα.
    - Επιλέξτε το “Next”, όπως φαίνεται στο παρακάτω σχήμα.
    - Έπειτα, επιλέξτε το «I accept the agreement» και στη συνέχεια επιλέξτε το “Next”.
    - Στα επόμενα παράθυρα επιλέξτε με τη σειρά το “Free” και μετά επιλέξτε το “Next”.
    - Μόλις τελειώσει η εγκατάσταση του, θα εμφανιστεί το αντίστοιχο παράθυρο στο οποίο μπορείτε να επιλέξτε το “Next”.
    - Στο επόμενο και τελευταίο για τον compiler παράθυρο μπορείτε να επιλέξτε το “Finish”, με το οποίο ολοκληρώνεται η εγκατάστασή του.



Σχήμα 0.5 : Παράθυρο Compiler Setup Wizard



Σχήμα 0.6 : Παράθυρο Licensing Information

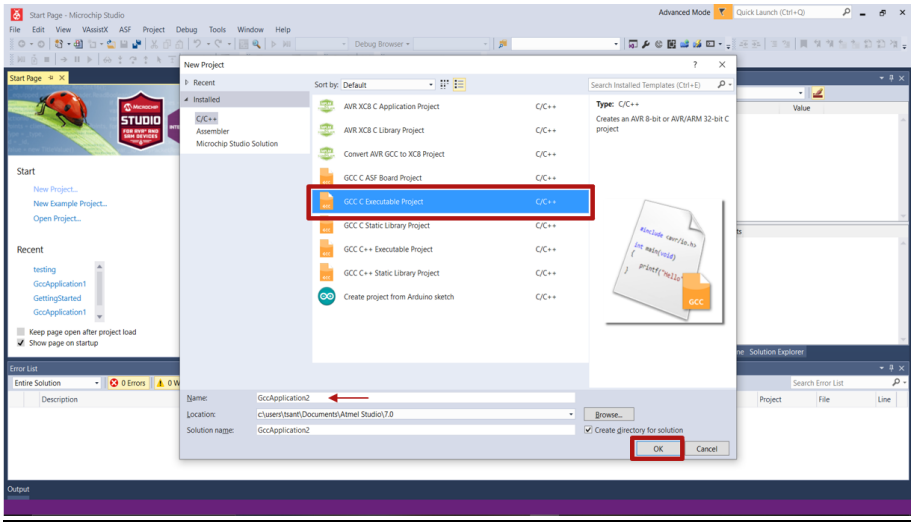
7. Ένα άλλο εργαλείο που θα εγκατασταθεί είναι και το Microsoft Visual, το οποίο γίνεται αυτόματα, χωρίς να απαιτείται κάποια περαιτέρω ενέργεια.

8. Όταν ολοκληρωθεί η εγκατάσταση, θα εμφανιστεί το παρακάτω παράθυρο, στο οποίο μπορείτε να επιλέξετε “Close” και να ανοίξει αυτόματα το Microchip Studio.

Σχήμα 0.7 : Παράθυρο Installation Complete

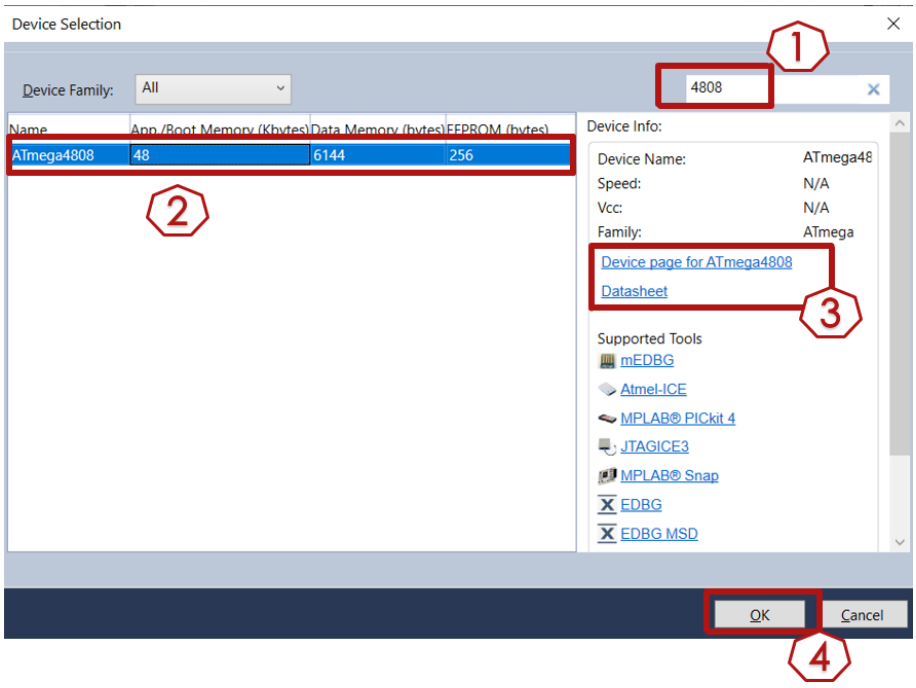
4. Δημιουργία Νέου Project

Στο εγκαταστημένο πλέον Microchip Studio, κατευθυνθείτε στην επιλογή "File → New → Project". Τα παραδείγματα που θα υλοποιηθούν, θα είναι στη γλώσσα C και επομένως θα επιλέξουμε το "GCC C Executable Project". Επιλέγουμε το όνομα της εφαρμογή μας και στη συνέχεια επιλέγουμε "OK".



Σχήμα 0.8 : Παράθυρο Δημιουργίας "New Project"

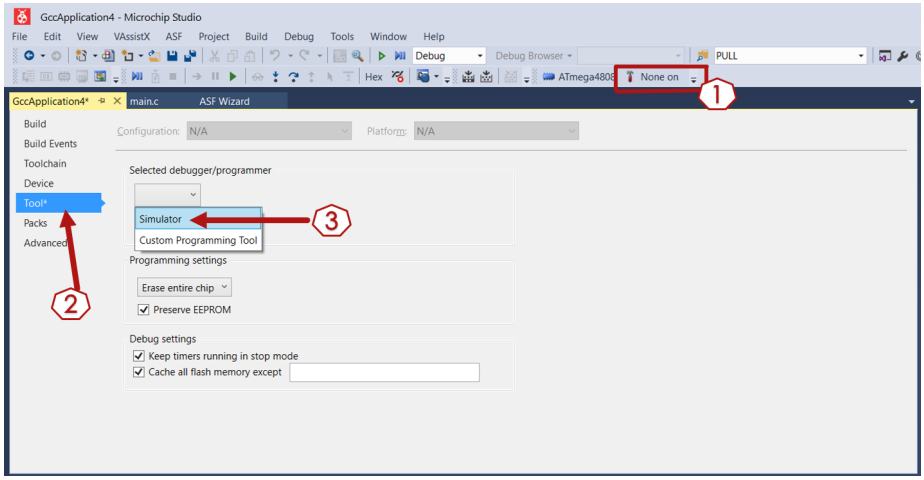
Επιλέγουμε το μικροελεγκτή "ATmega4808", ακολουθώντας τα βήματα, που φαίνονται στην εικόνα του παρακάτω σχήματος. Μπορούμε να δούμε κατευθείαν το Datasheet και το Device Page του συγκεκριμένου μικροελεγκτή, για περισσότερες πληροφορίες σχετικά με τις δυνατότητές του (Βήμα 3). Αφού επιλέξουμε την συσκευή μας, στη συνέχεια επιλέγουμε το "OK" (Βήμα 4) και στη συνέχεια μπορούμε να αρχίσουμε να προγραμματίζουμε.



Σχήμα 0.9 : Παράθυρο Επιλογής “Device”

5. Simulation και Debugging

Είναι σημαντικό όταν κάνουμε πρώτη φορά Debug να έχουμε επιλέξει το “Simulator” ως “Debugger”, ακολουθώντας τα βήματα που φαίνονται στο παρακάτω σχήμα.



Σχήμα 0.9 : Παράθυρο Simulator – Debugger

6. Παράδειγμα 1: Λειτουργία ενός LED

Για να μπορέσουμε να ενεργοποιήσουμε/απενεργοποιήσουμε ένα LED με κάποια συγκεκριμένη τιμή στην περίοδο, πχ 10ms, θα 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 5^{ο} bit του Register DIR (Data Direction) με '1'. Τώρα μπορούμε να δώσουμε την τιμή '0' ή '1' στον Register Out (Output Value) και να "ενεργοποιούμε " ή να "απενεργοποιούμε", το LED αντίστοιχα.

<u>Σημείωση</u>: Στη φάση του Simulation οι χρόνοι του "delay" όπως και των "timers", (που θα δούμε στη συνέχεια) δεν είναι αντιπροσωπευτικοί του χρόνου που αναμένουμε. Μόνο με την εκτέλεση του κώδικα πάνω στην πλακέτα μπορούμε να αντιληφθούμε τους πραγματικούς χρόνους. Για το λόγο αυτό, μπορούμε να επιλέγουμε σχετικά μικρούς χρόνους, για να αποφύγουμε μεγάλες καθυστερήσεις αναμονής.

Ο κώδικας της υλοποίησης του παραδείγματος παρουσιάζεται παρακάτω:

```
#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
	}
}
```

Σχήμα 0.10 : Κώδικας Παραδείγματος 1

Μπορείτε να χρησιμοποιήσετε τον κώδικα του παραδείγματος, για να εκτελέσετε τη διαδικασία του "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 τις τιμές που παίρνουν οι καταχωρητές (Registers) του PORTD.

Σημείωση: Για να δείτε τις τιμές των δηλωμένων σταθερών PIN1_bm και άλλων που θα δούμε στη συνέχεια, μπορείτε να ανατρέξετε στο header file iom4808.h, το οποίο βρίσκεται στον φάκελο που έχετε εγκαταστήσει το Microchip Studio :

πχ Path, "~\Studio\7.0\packs\atmel\ATmega_DFP\1.6.364\include\avr\iom4808.h"

7. Παράδειγμα 2: Interrupt

Τα switchs που μπορούν να χρησιμοποιηθούν βρίσκονται 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.

Όταν γίνει το interrupt πρέπει να 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.

Ο κώδικας της υλοποίησης του παραδείγματος, παρουσιάζεται παρακάτω:

```
#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;
}
```

Σχήμα 0.11 : Κώδικας Παραδείγματος 2

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.

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.

8. Παράδειγμα 3: Timer

Για να λειτουργήσει ο Timer TCA0 σε κανονική λειτουργία (normal mode) και να εκτελεστεί διακοπή (interrupt), όταν φτάσει σε μία προβλεπόμενη τιμή θα 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: Για περισσότερες λεπτομέρειες μπορείτε να ανατρέξτε στη σελ. 243, στο ATmega4808 DataSheet.

Ο μετρητής (counter) θα αρχίσει να μετράει και όταν θα φτάσει την τιμή που θέσαμε στον CMP0 θα ενεργοποιηθεί η ISR Routine με όρισμα το Vector TCA0_CMP0_vect. Αυτό είναι το Interrupt Vector που έχει οριστεί για το συγκεκριμένο Timer.

Ο κώδικας της υλοποίησης του παραδείγματος, παρουσιάζεται παρακάτω:

```
#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;
}
```

Σχήμα 0.12 : Κώδικας Παραδείγματος 3

Ενεργοποιούμε το LED και ενεργοποιούμε το χρονιστή (timer). Όταν ο χρονιστής φτάσει την τιμή που έχουμε θέσει αρχικά, τότε:

- Ενεργοποιείται η ρουτίνα διαχείρισης της διακοπής.
- Αλλάζουμε τη μεταβλητή interr.
- Το πρόγραμμα μας ολοκληρώνεται απενεργοποιώντας το LED.

<u>Σημείωση:</u> Μπορείτε να χρησιμοποιείτε τα breakpoints για να δείτε εάν ένα interrupt ενεργοποιείται.

Η τιμή του καταχωρητή CMP0 (value) ορίζει το 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:

$$f_{timer} = \frac{f_{system}}{\mathrm{N_{prescaler}}}$$

$$value = T * f_{timer}$$

Ο ATmega4808 λειτουργεί σε μέγιστο 20 MHz 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:

$$f_{timer} = \frac{20MHz}{1024} = 19{,}531\ KHz$$

$$T = \frac{value}{f_{timer}} = 1{,}024\ ms$$

Εργαστηριακή Άσκηση 01:

Φανάρια Κυκλοφορίας (Traffic Lights)

1. Περιγραφή

Σκοπός της Εργαστηριακής Άσκησης είναι να προσομοιώσουμε τη λειτουργία μιας διασταύρωσης αυτοκινητόδρομου, η οποία αποτελείται από ένα μεγάλο δρόμο και έναν μικρότερο. Στον μεγάλο δρόμο υπάρχει ένα φανάρι για τα αυτοκίνητα και ένα φανάρι για τους πεζούς, που ανάβει μόνο μετά από πίεση ενός κουμπιού (push button).

Στο μικρό δρόμο υπάρχει ένας αισθητήρας, (που θα υλοποιηθεί με συνάρτηση τυχαιότητας, Random function), ο οποίος όταν εντοπίσει ότι υπάρχει κάποιο αυτοκίνητο που περιμένει, στέλνει εντολή να ανάψει το κόκκινο φανάρι για το μεγάλο δρόμο και το πράσινο για το μικρότερο δρόμο. Ο μικρότερος δρόμος δεν έχει έξυπνο φανάρι για πεζούς.

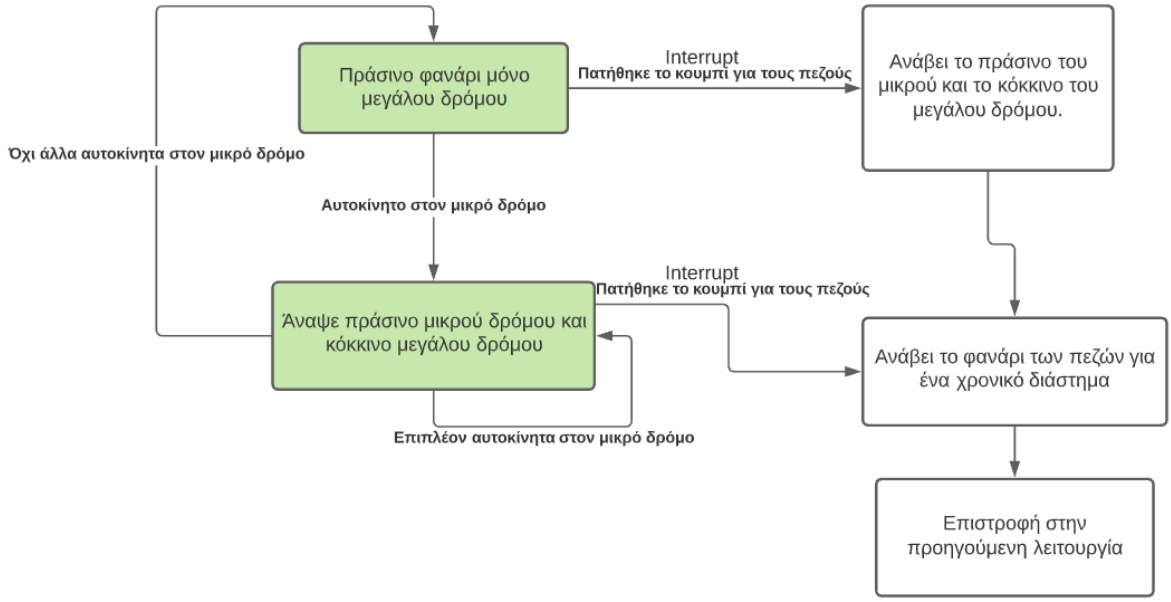
Όταν πατηθεί το κουμπί των πεζών, για το έξυπνο φανάρι του μεγάλου δρόμου, πρώτα ανάβει το κόκκινο φανάρι για τα αυτοκίνητα, για τον μεγάλο δρόμο, μετά ανάβει το πράσινο φανάρι για τα αυτοκίνητα, για το μικρό δρόμο (αν είναι ήδη αναμμένο το αφήνουμε όπως έχει) και τέλος ανάβει το πράσινο φανάρι για τους πεζούς στο μεγάλο δρόμο, για ένα συγκεκριμένο χρονικό διάστημα.

Για την υλοποίηση της Εργαστηριακής Άσκησης 1, θεωρείστε ότι:

- Το φανάρι είναι πράσινο όταν το LED είναι ανοιχτό και κόκκινο όταν είναι σβηστό. Τα τρία PIN του PORTD που χρησιμοποιούνται είναι τα PIN0, PIN1και 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 μπορεί να υλοποιηθεί ο χρόνος στον οποίο είναι αναμμένο το φανάρι των πεζών. Ορίστε εσείς μια μονάδα χρόνου, που σας διευκολύνει, για την προσομοίωση, αλλά εξηγήστε τον τρόπο σκέψης και τη διαδικασία του υπολογισμού της τιμής (value).

2. Διάγραμμα Ροής

Στο παρακάτω Σχήμα 1.1, παρουσιάζεται το αντίστοιχο διάγραμμα ροής, της Εργαστηριακής Άσκησης:



Σχήμα 1.1 : Διάγραμμα Ροής, Εργαστηριακής Άσκησης 01

3. Αναφορά Εργαστηριακής Άσκησης 1

- Αναπτύξτε τον κώδικα σας και βεβαιωθείτε πως όλες οι λειτουργίες επιτελούνται, όπως περιγράφετε στο παραπάνω διάγραμμα ροής.

- Παραδίδετε αναλυτική περιγραφή, με τη λειτουργία του κώδικά σας. Εξηγήστε τον τρόπο με τον οποίο ο κώδικά σας εκτελεί όλες της λειτουργίες, της εφαρμογής που αναλύθηκε παραπάνω, καθώς και τα πιθανά στοιχεία (interrupts, timers, random function, κα), που επιλέξατε και χρησιμοποιήσατε.

  **** Σημείωση: Παραδίδετε τον κώδικας σας ανά ομάδα και τα αναλυτικά σχόλια, για τις εντολές που χρησιμοποιήσατε.*

4. Παράδειγμα 4: Λειτουργία του ADC (Analog to Digital Converter)

Ο ADC είναι ένα σύστημα το οποίο μετατρέπει αναλογικά σήματα που έρχονται ως είσοδοι στα κατάλληλα PINs, όπως ήχος, ρεύμα, φως, κτλ., σε ψηφιακό σήμα. Επομένως, οι τιμές των αναλογικών σημάτων μπορούν να διαβαστούν πλέον από τον μικροελεγκτή και να αξιοποιηθούν κατάλληλα. Ο ADC που εμπεριέχεται στην πλακέτα δέχεται από το PIN7 του PORTD μετρήσεις. Επίσης, τα βασικά modes λειτουργίας του ADC είναι δύο. Το free-running mode στο οποίο ο ADC συνεχώς δέχεται τιμές και τις μετατρέπει σε ψηφιακά σήματα. Η δεύτερη και η προκαθορισμένη λειτουργία του εκτελεί μόνο μία μετατροπή όποτε ζητηθεί στον κώδικα και μετά σταματά.

Στο παράδειγμα, όταν ο ADC θα διαβάζει μετρήσεις που είναι κάτω από μία τιμή (για παράδειγμα 10) θα ενεργοποιείται ένα interrupt. Στο interrupt θα ανάβει ένα LED για 5ms και μετά θα επιστρέφουμε στην αρχική ροή και θα περιμένουμε ξανά την μέτρηση του ADC.

Για την ενεργοποίηση του ADC πρέπει:

1. Πρώτα να επιλέξουμε αν θέλουμε 10-bit ή 8-bit resolution → Resolution Selection bit (RESSEL) in Control A register (ADCn.CTRLA).
2. Έπειτα, επιλέγουμε το Free-Running mode → Γράφουμε ‘1’ στο Free-Running bit (FREERUN) in ADCn.CTRLA.
3. Επιλέγουμε με ποιο bit θα συνδεθεί ο ADC → MUXPOS bit field in MUXPOS register (ADCn.MUXPOS).
4. Κάνουμε enable τον ADC → Γράφουμε ‘1’ στο ENABLE bit in ADCn.CTRLA.
5. Μας δίνεται η επιλογή να ενεργοποιήσουμε το Debug Mode ώστε ο ADC να μην σταματά ποτέ να τρέχει και να λαμβάνει τιμές.

Με τη λειτουργία του Window Comparator Mode ελέγχονται όλες οι τιμές που εισάγονται από τον ADC με μια δοθείσα τιμή, δηλαδή ένα threshold. Για να ενεργοποιηθεί αυτή η λειτουργία πρέπει να ακολουθηθούν τα παρακάτω βήματα:

1. Πρώτα εισάγεται το threshold στην καταχωρητή ADCn.WINLT και/ή ADCn.WINHT.
2. Ενεργοποιείται η λειτουργία δημιουργίας interrupts → Window Comparator Interrupt Enable bit (WCOMP) in Interrupt Control register (ADCn.INTCTRL).
3. Ορίζεται το επιθυμητό Mode (στην περίπτωσή μας θέλουμε interrupt όταν RESULT<THRESHOLD) → WINCM bit field in ADCn.CTRLE.

Εφόσον προγραμματιστεί κατάλληλα ο ADC και οι λειτουργίες του, απομένει να γράψουμε το λογικό ‘1’ στο Start Conversion Bit (STCONV) στον Command Register (ADCn.COMMAND), το οποίο εκκινεί το conversion. Οι τιμές καταγράφονται στον καταχωρητή RES (Result).

<u>Συμβουλή:</u> Ανατρέξτε στο ATmega4808 DataSheet και διαβάστε καλά τις σελίδες 394-421. Αναγράφονται αναλυτικά όλες οι πιθανές λειτουργίες που μπορείτε να χρησιμοποιήσετε για να κάνετε Analog to Digital Conversion και πως να τις προγραμματίσετε.

Ο κώδικας της υλοποίησης του παραδείγματος παρουσιάζεται παρακάτω.

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

int main(){
	PORTD.DIR |= PIN1_bm; //PIN is output
//initialize the ADC for Free-Running mode
	ADC0.CTRLA |= ADC_RESSEL_10BIT_gc; //10-bit resolution
	ADC0.CTRLA |= ADC_FREERUN_bm; //Free-Running mode enabled
	ADC0.CTRLA |= ADC_ENABLE_bm; //Enable ADC
	ADC0.MUXPOS |= ADC_MUXPOS_AIN7_gc; //The bit
//Enable Debug Mode
	ADC0.DBGCTRL |= ADC_DBGRUN_bm;
//Window Comparator Mode
	ADC0.WINLT |= 10; //Set threshold
	ADC0.INTCTRL |= ADC_WCMP_bm; //Enable Interrupts for WCM
	ADC0.CTRLE |= ADC_WINCM0_bm; //Interrupt when RESULT < WINLT
	sei();
	ADC0.COMMAND |= ADC_STCONV_bm; //Start Conversion
while(1){  }
}
```

```c
ISR(ADC0_WCOMP_vect){
	int intflags = ADC0.INTFLAGS;
	ADC0.INTFLAGS = intflags;
	PORTD.OUTCLR= PIN1_bm; //LED is on
	_delay_ms(5);
	PORTD.OUT |= PIN1_bm; //LED is off
}
```

Σχήμα 2.1 : Κώδικας Παραδείγματος 4

Σημείωση: Για την προσομοίωση της λειτουργίας του ADC πρέπει να γράφεται εσείς την τιμή εισόδου στον καταχωρητή RES (Result) χειροκίνητα.

Εργαστηριακή Άσκηση 02:

Έξυπνη Οικιακή Συσκευή, με Κίνηση στο Χώρο

1. Περιγραφή

Σκοπός αυτής της εργαστηριακής άσκησης είναι η προσομοίωση της λειτουργίας μιας έξυπνης οικιακής συσκευής που κινείται στον χώρο ενός άδειου δωματίου. Ξεκινάει από μία γωνία του δωματίου και ο σκοπός της είναι να σχεδιάσει το περίγραμμά του. Καθώς κινείται στον χώρο θα παίρνει τιμές από έναν αισθητήρα που μετράει την απόσταση της συσκευής από ένα εμπόδιο μπροστά της (εδώ οι τιμές του αισθητήρα θα μετρούνται από τον ADC). Αν η τιμή είναι κάτω του επιτρεπτού (ορίστε εσείς μια οποιαδήποτε τιμή μεταξύ 1-254) η συσκευή θα πρέπει να σταματήσει και να κινηθεί αριστερά. Η συσκευή θα επιλέγει να κινηθεί δεξιά αν ο δεξιός αισθητήρας μας δείχνει ότι δεν υπάρχει τοίχος στα δεξιά της συσκευής (επίσης θα προσομοιωθεί με τον ADC), δηλαδή η τιμή είναι πάνω του επιτρεπτού. Όταν φτάσει στην γωνία από την οποία ξεκίνησε θέλουμε να σταματήσει. Επίσης θέλουμε αν πατηθεί ένα switch να μπορεί να γυρίσει πίσω στην θέση της ακολουθώντας την πορεία που έχει κάνει μέχρι τώρα αλλά ανάποδα.

Κάποιες παραδοχές που θα χρησιμοποιηθούν είναι οι ακόλουθες:

- Η κίνηση θα προσομοιωθεί με ένα LED (όταν κινείται ευθεία το LED είναι ανοιχτό αλλιώς κλείνει).
- Υπάρχει μόνο ένας ADC διαθέσιμος στην πλακέτα, επομένως για να τον χρησιμοποιήσετε σε δύο διαφορετικές καταστάσεις (έλεγχος αν πλησιάζει σε τοίχο και έλεγχος αν υπάρχει τοίχος δεξιά) πρέπει να χρησιμοποιήσετε για την μία κατάσταση το free-running mode και για την άλλη να εκτελείται μόνο ένα conversion όταν ζητηθεί (δεν χρειάζεται δηλαδή η δεύτερη κατάσταση να ελέγχεται συνεχώς).
- Η αριστερή και δεξιά κίνηση θα προσομοιωθούν με δύο διαφορετικά LED τα οποία είναι αναμμένα μέχρι ένας timer φτάσει μια συγκεκριμένη τιμή (βάλτε τιμές που σας βολεύουν και εξηγήστε πως βρήκατε τα values).
- Όταν πατηθεί ένα switch θα ενεργοποιείται η ανάποδη πορεία.

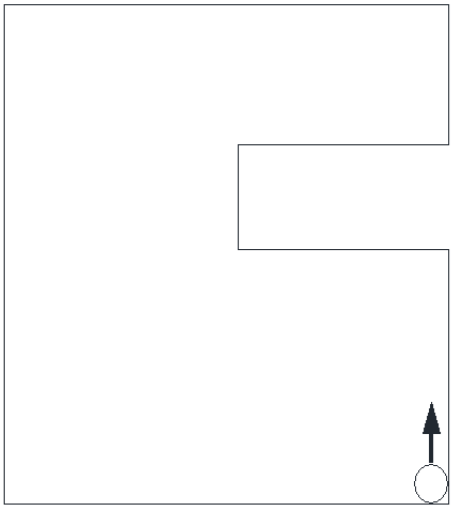
Όσον αφορά την ανάποδη πορεία, η συσκευή θέλουμε να γυρίσει 180 μοίρες. Αυτό θα προσομοιωθεί με τα τρία LEDs να είναι ταυτόχρονα ανοιχτά για ένα χρονικό διάστημα (χρήση timer). Έπειτα, θα εκτελεί την ίδια ακριβώς διαδικασία αλλά ανάποδα. Συγκεκριμένα, καθώς προχωράει μπροστά και βρει τοίχο θέλουμε να στρίψει δεξιά (ανοιχτό το αντίστοιχο LED). Όταν καταλάβει ότι δεν υπάρχει τοίχος αριστερά θέλουμε να κινηθεί αριστερά (ανοιχτό το αντίστοιχο LED). Τέλος, όταν καταλάβει ότι επέστρεψε στην αρχική της θέση, τερματίζει.

2. Ερωτήματα Εργαστηριακής Άσκησης 2

1) Υλοποιήστε τον κώδικα της κίνησης της οικιακής συσκευής, όταν το δωμάτιο είναι τετράγωνο (επομένως δεν χρειάζεται ο δεξιά αισθητήρας και η δεύτερη λειτουργία του ADC).

2) Υλοποιήστε τον κώδικα της κίνησης στο δωμάτιο που παρουσιάζεται στο παρακάτω σχήμα 2.2, εδώ εισάγεται και η δεύτερη λειτουργία του ADC.

3) Υλοποιήστε την ανάποδη λειτουργία.



Σχήμα 2.2 : Σχήμα δωματίου

3. Αναφορά Εργαστηριακής Άσκησης 2

1. Αναπτύξτε κώδικα και βεβαιωθείτε πως όλα τα μονοπάτια λειτουργούν όπως προβλέπεται από το σχήμα ροής.

2. Παραδίδεται αναλυτική αναφορά με τη λειτουργία του κώδικά σας. Εξηγήστε τον τρόπο με τον οποίο ο κώδικά σας εκτελεί όλα τα πιθανά μονοπάτια της εφαρμογής καθώς και τα στοιχεία (interrupts, timers, ADC, κτλ.) που χρησιμοποιήσατε.

**** Σημείωση: Παραδίδετε τον κώδικας σας ανά ομάδα και τα αναλυτικά σχόλια, για τις εντολές που χρησιμοποιήσατε.*

4. Παράδειγμα 5: Λειτουργία του PWM (Pulse-Width Modulation)

Το Pulse-Width Modulation μπορεί να χρησιμοποιηθεί με πολλά περιφερειακά και εφαρμογές, όπως:

- Audio
- Ρύθμιση έντασης LED
- Analog signal generation
- Ρύθμιση servos, DC motors, κτλ.

Μέσω του μικροελεγκτή δημιουργείται ένα waveform το οποίο συνδέεται με το pin ενός περιφερειακού. Συγκεκριμένα, το waveform output είναι διαθέσιμο στο PORT που επιλέγεται και μπορεί να χρησιμοποιηθεί ως έξοδος (αρκεί να οριστεί το PORT ως έξοδος).

Στο παράδειγμα αυτό ενεργοποιείται το PWM ώστε να δημιουργηθεί ένας παλμός που θα λειτουργεί σαν ρολόι. Με αυτό θα αναβοσβήνει ένα LED, το οποία θα ανάβει όταν ο παλμός ανεβαίνει στην ψηλή στάθμη (rising edge) και θα σβήνει όταν πέφτει στη χαμηλή στάθμη (falling edge).

Η χρήση ενός Single-Slope PWM generator θα γίνει με τη βοήθεια του TCA timer:

1. Αρχικά, πρέπει να οριστεί ο prescaler του timer (όπως και στην περίπτωση του απλού timer) →TCA0.SINGLE.CTRLA=TCA_SINGLE_CLKSEL_DIV1024_gc.

2. Έπειτα, πρέπει να δοθεί η μέγιστη τιμή TOP μέχρι την οποία θα μετρήσει ο timer → TCA0.SINGLE.PER = value.

3. Ορίζεται το duty cycle του παλμού μέσω της χρήσης του καταχωρητή CMPx → TCA0.SINGLE.CMP0 = value.

4. Γίνεται η επιλογή του Waveform Generation Mode, στην περίπτωσή μας το Single-Slope → TCA0.SINGLE.CTRLB |= TCA_SINGLE_WGMODE_SINGLESLOPE_gc.

5. Τέλος, ενεργοποιείται ο TCA → TCA0.SINGLE.CTRLA |= TCA_SINGLE_ENABLE_bm.

Με την επιλογή του prescaler και την εκχώρηση τιμής στον καταχωρητή PER, η συχνότητα υπολογίζεται με τον παρακάτω τύπο.

$$f_{PWM\_SS} = \frac{f_{CLK\_PER}}{N(PER + 1)}$$

Ο καταχωρητής PER είναι 16-bit, επομένως το ελάχιστο resolution είναι TCA.PER = 0x0002 και το μέγιστο είναι TCA.PER = MAX-1. Η τιμή του CMPx καθορίζει το duty cycle. Για παράδειγμα, έχει την μισή τιμή του καταχωρητή PER τότε το duty cycle είναι 50%.

<u>Σημείωση:</u> Υπάρχουν και άλλες λειτουργίες του PWM. Για περισσότερες λεπτομέρειες ανατρέξτε στο ATmega4808 DataSheet σελ.191-199.

Όταν ανεβαίνει στην υψηλή στάθμη και όταν κατεβαίνει στην χαμηλή στάθμη, ενεργοποιούνται τα κατάλληλα Interrupts. Η λειτουργία αυτή ορίζεται με τον ακόλουθο τρόπο:

- Ενεργοποιώντας το Overflow Interrupt, δηλαδή εκτελείται διακοπή όταν ο timer είναι ίσος με την BOTTOM τιμή (ανεβαίνει στην ψηλή στάθμη) → TCA0.SINGLE.INTCTRL = TCA_SINGLE_OVF_bm.
- Ενεργοποιώντας και το CMPx Interrupt, δηλαδή εκτελείται διακοπή όταν ο timer είναι ίσος με την τιμή του καταχωρητή CMPx (πηγαίνουμε στην χαμηλή στάθμη) → TCA0.SINGLE.INTCTRL |= TCA_SINGLE_CMP0_bm.

<u>Συμβουλή:</u> Ανατρέξτε στο ATmega4808 DataSheet και διαβάστε καλά τις σελίδες σχετικά με τον TCA0. Αναγράφονται αναλυτικά όλες οι πιθανές λειτουργίες του PWM και υπάρχουν παραδείγματα για τον σχηματισμό παλμών. Επίσης, στην άσκηση μπορείτε να χρησιμοποιήσετε και τους timers TCB που έχουν λειτουργία 8-bit PWM (σελ.239-).

```c
#include <avr/io.h>
#include <avr/interrupt.h>

int main(){
	PORTD.DIR |= PIN1_bm; //PIN is output
	//prescaler=1024
	TCA0.SINGLE.CTRLA=TCA_SINGLE_CLKSEL_DIV1024_gc;
	TCA0.SINGLE.PER = 254; //select the resolution
	TCA0.SINGLE.CMP0 = 127; //select the duty cycle
	//select Single_Slope_PWM
	TCA0.SINGLE.CTRLB |= TCA_SINGLE_WGMODE_SINGLESLOPE_gc;
	//enable interrupt Overflow
	TCA0.SINGLE.INTCTRL = TCA_SINGLE_OVF_bm;
	//enable interrupt COMP0
	TCA0.SINGLE.INTCTRL |= TCA_SINGLE_CMP0_bm;
	TCA0.SINGLE.CTRLA |= TCA_SINGLE_ENABLE_bm; //Enable
	sei();
	while (1){ }
}
```

```c
ISR(TCA0_OVF_vect){
	//clear the interrupt flag
	int intflags = TCA0.SINGLE.INTFLAGS;
	TCA0.SINGLE.INTFLAGS = intflags;
	PORTD.OUT |= PIN1_bm; //PIN is off
}
```

```c
ISR(TCA0_CMP0_vect){
	//clear the interrupt flag
	int intflags = TCA0.SINGLE.INTFLAGS;
	TCA0.SINGLE.INTFLAGS = intflags;
	PORTD.OUTCLR |= PIN1_bm; //PIN is off
}
```

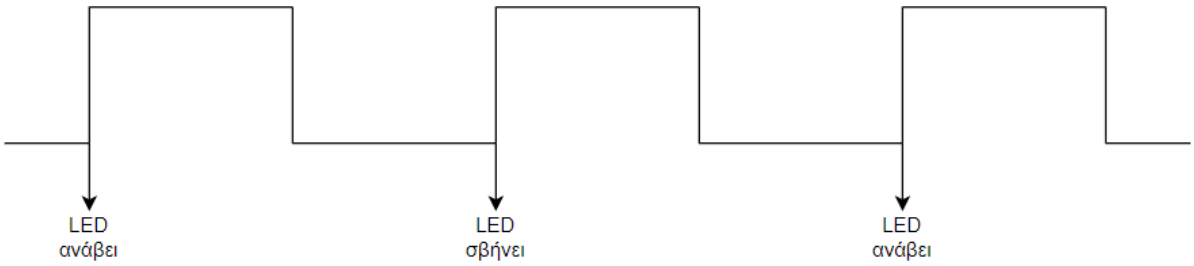
Σχήμα 2.1 : Κώδικας Παραδείγματος 5

Εργαστηριακή Άσκηση 03:

Εξοικείωση με το Pulse-Width Modulation

1. Περιγραφή

Σε αυτή την εργαστηριακή άσκηση, υλοποιείται περαιτέρω η προσομοίωση της οικιακής συσκευής που κινείται στο χώρο. Συγκεκριμένα θα προσομοιωθεί η κίνηση των δύο τροχών της, όταν στρίβει δεξιά και αριστερά, οι οποίοι θα κινούνται σύμφωνα με δύο διαφορετικούς ρυθμούς, που θα καθορίζονται από δύο διαφορετικούς PWM generators (χρησιμοποιήστε όποιους καταχωρητές σας διευκολύνουν). Ο ρυθμός κάθε τροχού θα εμφανίζεται σε ένα LED, το οποίο θα ενεργοποιείται όταν ο παλμός θα βρίσκεται στην ανερχόμενη παρυφή (rising edge) και θα απενεργοποιείται όταν ακολουθεί η ανερχόμενη παρυφή (rising edge) του επόμενου παλμού, (όπως φαίνεται αναλυτικά και στο σχήμα 3.2). Το LED0 (PORTD PIN0) θα αντιστοιχεί στον δεξιό τροχό και το LED1 (PORTD PIN1) στον αριστερό τροχό.



Σχήμα 3.2 : Αναπαράσταση λειτουργίας LED μέσω του PWM.

Στην αρχική κατάσταση τα δύο LEDw (οι δύο παλμοί), αναβοσβήνουν με τον ίδιο ρυθμό καθώς η συσκευή κινείται ευθεία, επομένως οι δύο τροχοί της κινούνται παράλληλα. Με τον ίδιο τρόπο όπως και στην Εργαστηριακή Άσκηση 2, η οικιακή συσκευή έχει έναν αισθητήρα μπροστά της, ο οποίος δίνει τιμές στον ADC. Όταν πλησιάσει σε τοίχο, δηλαδή ο ADC εμφανίσει μια τιμή μικρότερη από μία τυχαία τιμή που έχει ορισθεί, θα πρέπει η συσκευή να σταματήσει να κινείται (να σταματήσουν οι PWMs) και να περιμένει την επόμενη εντολή που θα της δοθεί. Η διαδικασία αυτή θα προσομοιωθεί με ένα τρίτο LED (PORTD PIN2), το οποίο θα είναι μόνο αυτό ανοιχτό μέχρι να δοθεί κάποια άλλη εντολή.

Δύο επιλογές θα υλοποιηθούν όταν η συσκευή θα σταματήσει. Πρώτον, η επιλογή να πάει δεξιά και δεύτερον η επιλογή να πάει αριστερά. Οι δύο επιλογές θα προσομοιωθούν αντίστοιχα με το πάτημα ενός switch (όταν πατηθεί το SW5 (PORTF PIN5) θα πάει δεξιά και όταν πατηθεί το SW6 (PORTF PIN6) θα πάει αριστερά). Ανάλογα με το ποιο κουμπί έχει πατηθεί, πρέπει οι ρόδες της συσκευής να κινηθούν αντίστοιχα:

- Όταν δοθεί εντολή να πάει δεξιά, ο δεξιός τροχός θα κινηθεί με τον διπλάσιο ρυθμό, από ότι ο αριστερός ώστε να στρίψει δεξιά. Επομένως, το LED0 θα αναβοσβήνει με διπλάσιο ρυθμό από ότι το LED1. Με τον ίδιο τρόπο όπως και πριν, θέλουμε τα δύο LEDs να ανάβουν όταν πραγματοποιηθεί η ανερχόμενη παρυφή (rising edge) του αντίστοιχου παλμού και να σβήνουν όταν πραγματοποιηθεί η ανερχόμενη παρυφή (rising edge), του ακόλουθου παλμού.

- Όταν αντίστοιχα δοθεί εντολή να πάει αριστερά, ο αριστερός τροχός θα κινηθεί με τον διπλάσιο ρυθμό από ότι ο δεξιός. Επομένως, αντίστοιχα το LED1 θα αναβοσβήνει με διπλάσιο ρυθμό από ότι το LED0.

- Όταν ξαναπατηθεί ο αντίστοιχος διακόπτης (switch), η διαδικασία της στροφής θα σταματάει και η συσκευή θα επιστρέφει στην αρχική της λειτουργία, (τα δύο LEDs θα αναβοσβήνουν με τον ίδιο ρυθμό).

Παρατήρηση: Καθώς και τα οι δύο διακόπτες (switches) βρίσκονται στο ίδιο PORT (PORTF), θα πρέπει να ελέγξετε μέσω του PORTF.INTFLAGS για να δείτε ποιο κουμπί (flag) ενεργοποιήθηκε. Συγκεκριμένα, θα πρέπει να απομονώσετε το ψηφίο (bit) 5 και ψηφίο (bit)6 του PORTF.INTFLAGS (μέσω κατάλληλου masking) και να ελέγχετε με μια if αν το αντίστοιχο ψηφίο (bit)είναι ΄1΄ (άρα το κουμπί πατήθηκε).

Παρακάτω θα αναφερθούν κάποιες πληροφορίες αναφορικά με τη λειτουργία PWM στον μικροελεγκτή. Αρχικά, πολλοί διαφορετικοί timers/counters εκτελούν λειτουργία PWM. Ο TCA είναι ένας από αυτούς, ο οποίος χρησιμοποιείται και στο παράδειγμα. Αυτός μπορεί να χρησιμοποιηθεί ως έχει, δηλαδή ως ένας 16-bit PWM, ή να ως δύο 8-bit timers/counters, οι οποίοι δημιουργούν δύο διαφορετικά PWMs. Προσοχή: αυτοί οι δύο 8-bit timers/counters δεν περιέχουν όλες τις λειτουργίες interrupts (δείτε σελ. 196 και σελ. 224 του ATmega4808 Datasheet).

Επιπλέον, υπάρχουν τρεις διαφορετικοί TCB timers/counters, ο καθένας από τους οποίους περιέχει ένα 8-bit PWM Mode (σελ. 239-240 του ATmega4808 Datasheet). Όλοι οι καταχωρητές του TCB και πως προγραμματίζονται κατάλληλα για το PWM Mode αναλύονται στις σελίδες 243-253 του ATmega4808 Datasheet. Επίσης, στο pdf “Getting Started with TCB” αναλύεται πως ορίζεται το PWM Mode των TCB (βρίσκεται στο e-class, στο φάκελο Βιβλιογραφία).

**** Σημείωση: Μπορείτε να μελετήσετε με λεπτομέρεια τις αναφερόμενες σελίδες των παραπάνω δύο Datasheets. *****

2. Ερωτήματα Εργαστηριακής Άσκησης 3

1) Υλοποιείστε την αρχική λειτουργία της συσκευής (δηλαδή την κίνηση ευθεία με τα δύο LED να αναβοσβήνουν ταυτόχρονα σύμφωνα με τον κατάλληλο προγραμματισμό των δύο παλμών PWM).

2) Προσθέστε στη λειτουργία σας τον ADC, ο οποίος όταν θα εμφανίζει μια τιμή μικρότερη από ένα threshold θα σταματά τη λειτουργία και θα ανάβει ένα τρίτο LED (LED2).

3) Προσθέστε τη λειτουργία των δύο διακοπτών (switches). Συγκεκριμένα, όταν θα πατηθεί το SW5 (PORTF PIN5) θα ενεργοποιηθεί η δεξιά περιστροφή και όταν πατηθεί το SW6 (PORTF PIN6) θα ενεργοποιηθεί η αριστερή περιστροφή.

4) Υλοποιήστε την δεξιά και αριστερή περιστροφή, προγραμματίζοντας κατάλληλα δύο παλμούς PWM. Ο ένας από αυτούς πρέπει κάθε φορά να είναι διπλάσιος του άλλου και αντίστοιχα με την κίνηση που εκτελείται, να αναβοσβήνουν τα κατάλληλα LEDs.

3. Αναφορά Εργαστηριακής Άσκησης 3

1. Αναπτύξτε τον κώδικα σας και βεβαιωθείτε πως όλα τα μονοπάτια λειτουργούν σωστά, όπως περιγράφεται στην εργαστηριακή άσκηση.

2. Παραδίδετε αναλυτική περιγραφή στην αναφορά σας, με τη λειτουργία του κώδικά σας. Εξηγήστε τον τρόπο με τον οποίο ο κώδικά σας εκτελεί όλα τα πιθανά μονοπάτια της εφαρμογής, καθώς και τα στοιχεία (interrupts, PWM, ADC, κτλ.) που χρησιμοποιήσατε.

3. Τέλος, στην αναφορά σας μπορείτε να σχεδιάσετε τους παλμούς που έχετε δημιουργήσει, να εξηγήσετε πως προέκυψαν και γιατί επιλέξατε αυτόν τον τρόπο υλοποίησης.

**** Σημείωση: Παραδίδετε τον κώδικας σας ανά ομάδα και τα αναλυτικά σχόλια, για τις εντολές που χρησιμοποιήσατε.*

Βιβλιογραφία - Αναφορές

1. Microchip, Επίσημη ιστοσελίδα, https://www.microchip.com/mplab/microchip-studio

2. AVR-IoT WxHardware User Guide, https://ww1.microchip.com/downloads/en/DeviceDoc/AVR-IoT-Wx-Hardware-User-Guide-DS50002805C.pdf

3. ATmega4808/4809 Data Sheet, https://ww1.microchip.com/downloads/en/DeviceDoc/ATmega4808-09-DataSheet-DS40002173B.pdf

4. Microchip, “Advanced Software Framework (ASF)”, https://www.microchip.com/mplab/avr-support/advanced-software-framework

5. Microchip, “MPLAB® XC Compilers”, https://www.microchip.com/en-us/development-tools-tools-and-software/mplab-xc-compilers

6. Getting Started with TCB, http://ww1.microchip.com/downloads/en/Appnotes/TB3214-Getting-Started-with-TCB-90003214A.pdf