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

Εργαστήριο Προηγμένοι Μικροεπεξεργαστές

Εργαστηριακές Ασκήσεις

Πάτρα, 2022-23

1 Εισαγωγικά

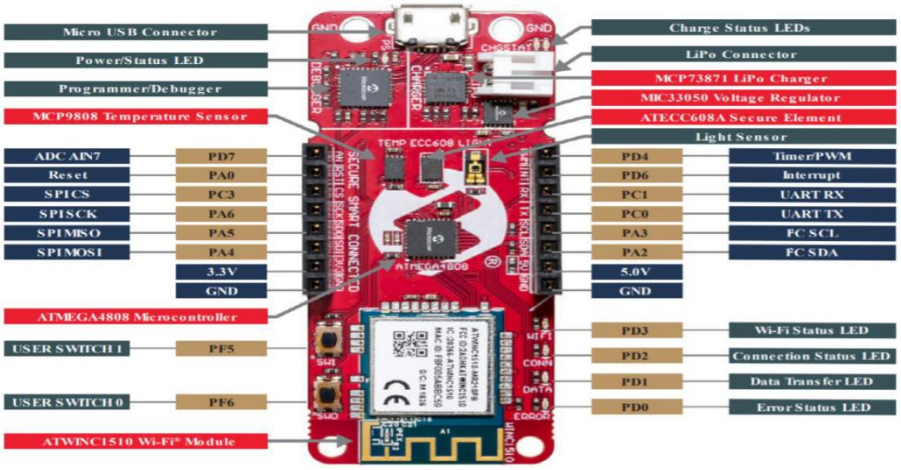
Η AVR είναι μια οικογένεια μικροεπεξεργαστών που αναπτύχθηκε αρχικά από την Atmel και μετέπειτα αγοράστηκε από την Microchip Technology. Βασίζεται σε μια τροποποιημένη Harvard αρχιτεκτονική, 8-bit RISC single-chip μικροεπεξεργαστή. Η AVR είναι μια από τις πρώτες οικογένειες μικροεπεξεργαστών που χρησιμοποίησαν 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 σε 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 αντίστοιχη 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

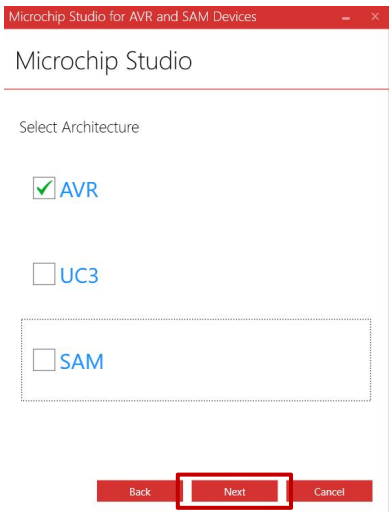
3 Εγκατάσταση Microchip Studio

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:

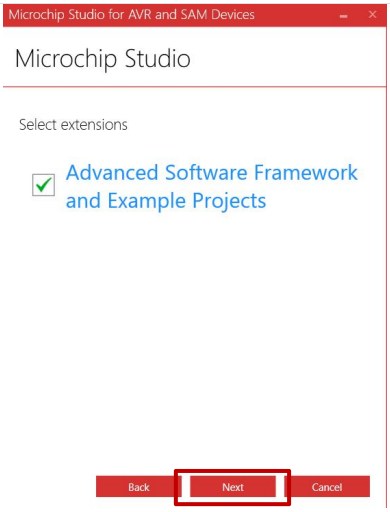
1 Πατήστε Download στην επιλογή Microchip Studio for AVR and SAM Devices 7.0.2542 Web Installer.
2 Αφού ολοκληρωθεί η παραπάνω διαδικασία, ανοίξτε το αρχείο και περιμένετε.
   - Στο παράθυρο που θα εμφανιστεί επιλέξτε το path στο οποίο θα αποθηκευτεί το Studio και επιλέξτε το κουτί “I agree to the Terms 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”.



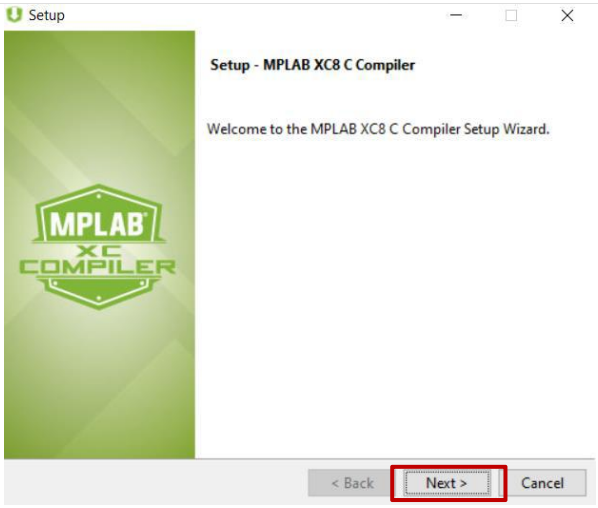
Σχήμα 0.3 : Παράθυρο Επιλογής ASF

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.

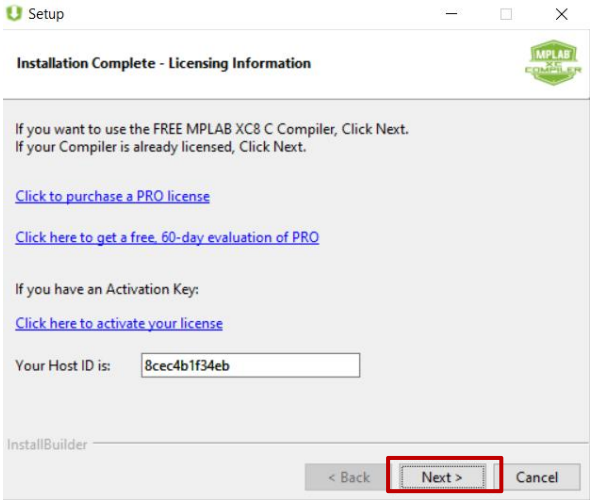
Σχήμα 0.4 : Παράθυρο Επιλογής System Validation

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

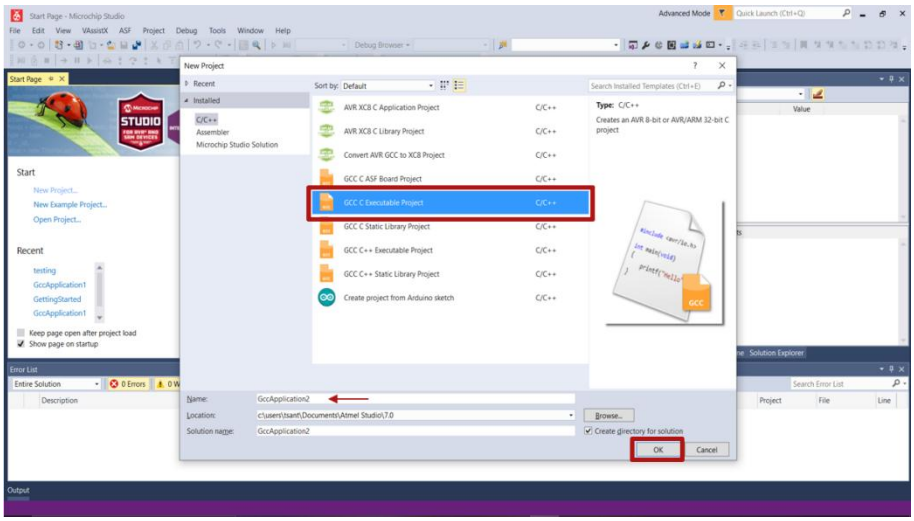
7 Ένα άλλο εργαλείο που θα εγκατασταθεί είναι και το Microsoft Visual, το οποίο γίνεται αυτόματα, χωρίς να απαιτείται κάποια περαιτέρω ενέργεια.

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

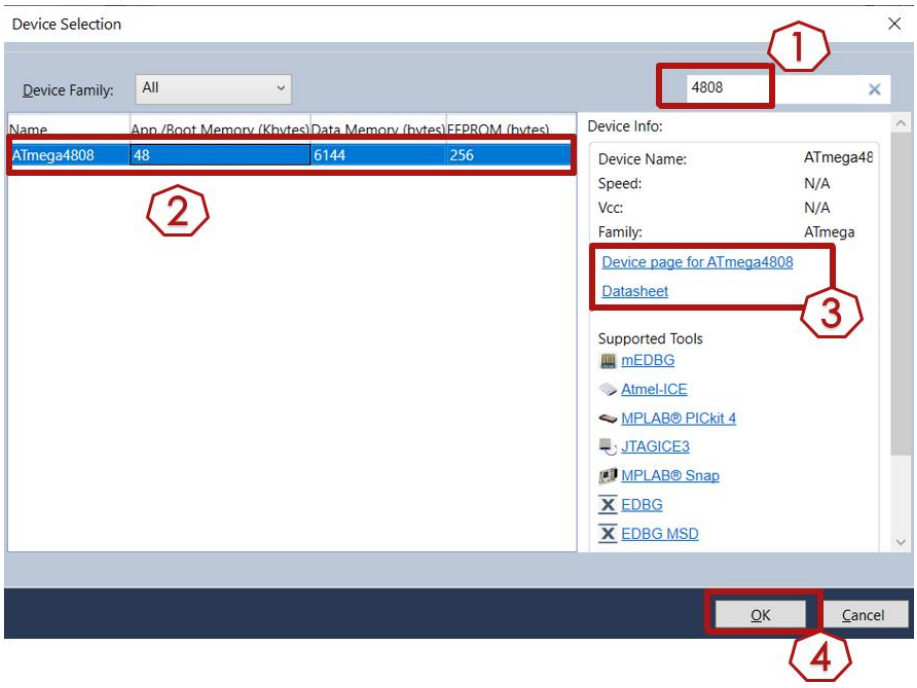
4 Δημιουργία Νέου Project

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 εφαρμογής μας και στη συνέχεια επιλέγουμε “OK”.



Σχήμα 0.8 : Παράθυρο Δημιουργίας “New Project”

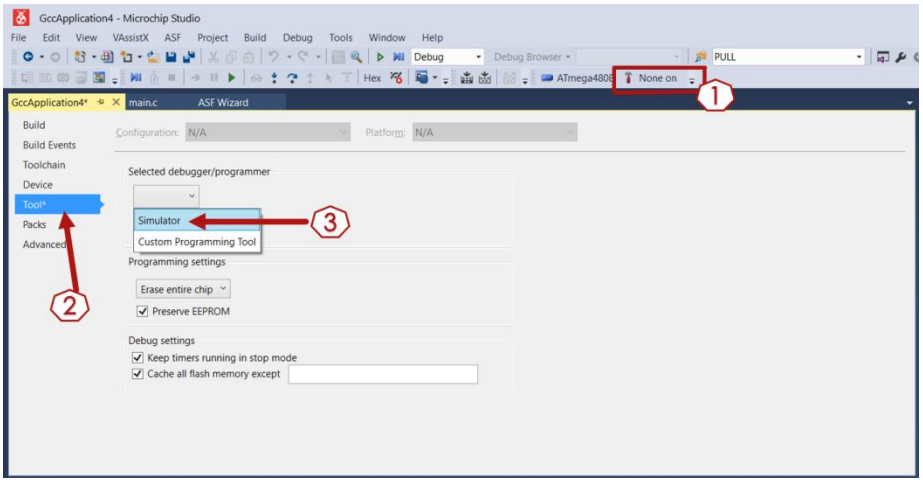
Επιλέγουμε τον μικροελεγκτή “ATmega4808”, ακολουθώντας τα βήματα, που φαίνονται στην εικόνα του παρακάτω σχήματος. Μπορούμε να δούμε κατευθείαν το Datasheet και το Device Page του συγκεκριμένου μικροελεγκτή, για περισσότερες πληροφορίες σχετικά με τις δυνατότητές του (Βήμα 3). Αφού επιλέξουμε την συσκευή 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 0.10 : Παράθυρο Simulator – Debugger

6 Παράδειγμα 1: Λειτουργία ενός LED

Για να ενεργοποιηθεί/απενεργοποιηθεί ένα LED με κάποια συγκεκριμένη τιμή στην περίοδο, π.χ. T=10ms, θα πρέπει πρώτα να οριστεί το "Direction" του συγκεκριμένου Pin ως "Output". Αρχικά, το PORT που θα χρησιμοποιηθεί είναι το PORTD καθώς οι τέσσερες πρώτοι ακροδέκτες (PINs) του συνδέονται με LEDs, πάνω στην πλακέτα (όπως φαίνεται και στο Board Overview). Για να οριστεί ένα PIN ως "Output" πρέπει να τεθεί το ψηφίο (bit) 5 του Register DIR (Data Direction) με '1'. Τώρα μπορεί να δοθεί η τιμή '0' ή '1' στον Register Out (Output Value) και να "ενεργοποιείται" ή να "απενεργοποιείται", το LED αντίστοιχα.

Σημείωση: 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){
	//PIN is output
	PORTD.DIR |= 0b00000010; //PIN1_bm
	//LED is off
	PORTD.OUT |= 0b00000010; //PIN1_bm
	while (1) {
		//on
		PORTD.OUTCLR= 0b00000010; //PIN1_bm
		_delay_ms(del); //wait for 10ms
		//off
		PORTD.OUT |= 0b00000010; //PIN1_bm
		_delay_ms(del); //wait for 10ms
	}
}
```

Σχήμα 0.11 : 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 :

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

7 Παράδειγμα 2: Διακοπή (Interrupt)

Οι διακόπτες (switches) που μπορούν να χρησιμοποιηθούν βρίσκονται στο PORTF και είναι τα PIN5 και PIN6 (ανατρέξτε στο Board Overview). Για τη χρήση ενός διακόπτη (switch), το pullup enable bit που του αντιστοιχεί πρέπει να είναι ενεργοποιημένο (ανατρέξτε σελ. 158 στο ATmega4808 DataSheet). Επίσης, το σημείο του παλμού στο οποίο θα ενεργοποιηθεί η μονάδα διαχείρισης της διακοπής (interrupt) πρέπει να οριστεί. Εδώ η ενεργοποίησή της και στις δύο άκρες του παλμού επιλέγεται (ανατρέξτε σελ. 158 στο ATmega4808 DataSheet). Με την ενεργοποίηση των συγκεκριμένων ψηφίων (bits) του PIN5, το σύστημα μπορεί να δεχτεί διακοπές (interrupts) όταν πατηθεί ο διακόπτης (switch).

Όταν γίνει η διακοπή (interrupt), συγκεκριμένη συνάρτηση η οποία θα το διαχειριστεί, ενεργοποιείται. Αυτή η συνάρτηση είναι μια ρουτίνα διαχείρισης διακοπών (ISR routine) η οποία παίρνει ως όρισμα μια διεύθυνση (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 συγκεκριμένο board.

Ο κώδικας της υλοποίησης του παραδείγματος, παρουσιάζεται παρακάτω:

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

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

int main() {

	PORTD.DIR |= 0b00000010; //PIN is output
	PORTD.OUT |= 0b00000010; //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 (x==0) {

		PORTD.OUTCLR= 0b00000010; //on

	}
	PORTD.OUT |= 0b00000010; //off
	cli(); //disenable interrupts

}

ISR(PORTF_PORT_vect){
	//clear the interrupt flag
	int y = PORTF.INTFLAGS;
	PORTF.INTFLAGS=y;
	x=1;
}
```

Σχήμα 0.12 : Κώδικας Παραδείγματος 2

Στο σύστημα υπάρχει ένα LED και ένας διακόπτης (switch). Σύμφωνα με την κανονική λειτουργία του προγράμματος, το LED ενεργοποιείται και απενεργοποιείται με περίοδο T=10ms. Όταν πατηθεί ο διακόπτης ενεργοποιείται η διακοπή (interrupt) και αλλάζει η τιμή μιας τυχαίας μεταβλητής (int x) ώστε να σταματήσει η λειτουργία του LED και του προγράμματος.

Για να ενεργοποιηθεί η διακοπή (interrupt) το ψηφίο (bit) 5 του register INTFLAGS στο PORTF πρέπει να πατηθεί, εφόσον το πρόγραμμα είναι σε κάποιο breakpoint. Μόλις το ψηφίο (bit) αλλάξει από ‘0’ σε ‘1’ και το πρόγραμμα συνεχίσει με την επόμενη εντολή (STEP OVER), θα ενεργοποιηθεί η ρουτίνα διαχείρισης διακοπών (interrupt routine).

8 Παράδειγμα 3: Χρονιστής (Timer)

Για να λειτουργήσει ο χρονιστής TCA0 timer σε κανονική λειτουργία (normal mode) και να ενεργοποιηθεί διακοπή (interrupt), όταν φτάσει σε μία προβλεπόμενη τιμή θα πρέπει:

- Να δοθεί η τιμή ‘0’ στον CTRLB register (Normal Mode).
- Να δοθεί η τιμή ‘0’ στον CNT register (θέτουμε τον χρονιστή – timer στο μηδέν).
- Να δοθεί η προβλεπόμενη τιμή στον CMP0 register.
- Να ενεργοποιηθούν οι διακοπές (interrupts) μέσω του INTCTRL register.
- Να τεθεί η συχνότητα ρολογιού (clock frequency).
- Τέλος, να ενεργοποιηθεί ο timer μέσω του CTRLA register.

Σημείωση: Για περισσότερες λεπτομέρειες μπορείτε να ανατρέξτε στη σελ. 243, στο ATmega4808 DataSheet.

Ο TCA0 θα αρχίσει να μετράει και όταν θα φτάσει την τιμή που τέθηκε στον CMP0 θα ενεργοποιηθεί η ISR routine με όρισμα το vector TCA0_CMP0_vect. Αυτό είναι το interrupt vector που έχει οριστεί για τον συγκεκριμένο χρονιστή/μετρητή (timer/counter).

Ο κώδικας της υλοποίησης του παραδείγματος, παρουσιάζεται παρακάτω:

```
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>
#define ped 20

int x=0;

int main() {

	PORTD.DIR |= 0b00000010; //PIN is output
	PORTD.OUTCLR= 0b00000010; //LED is on
	//(σελ 219, 224, 205) 16-bit counter high and low
	TCA0.SINGLE.CNT = 0; //clear counter
	//Normal Mode (TCA_SINGLE_WGMODE_NORMAL_gc  σελ 207)
	TCA0.SINGLE.CTRLB = 0;
	//When CMP0 reaches this value -> interrupt //CLOCK FREQUENCY/1024
	TCA0.SINGLE.CMP0 = ped;
	TCA0.SINGLE.CTRLA = 0x7<<1; //TCA_SINGLE_CLKSEL_DIV1024_gc σελ 224
	TCA0.SINGLE.CTRLA |=1;//Enable
	TCA0.SINGLE.INTCTRL = TCA_SINGLE_CMP0_bm; //Interrupt Enable (=0x10)
	sei(); //begin accepting interrupt signals
	while (x==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;
	x=1;
}
```

Σχήμα 0.13 : Κώδικας Παραδείγματος 3

Ενεργοποιείται το LED και ο χρονιστής (timer). Όταν ο χρονιστής φτάσει την τιμή που έχει τεθεί αρχικά, τότε:

- Ενεργοποιείται η ρουτίνα διαχείρισης της διακοπής.
- Αλλάζει μια τυχαία μεταβλητή (x), για να μπορεί να ελεγχθεί μέσα στην συνάρτηση main αν εκτελέστηκε η ρουτίνα διαχείρισης της διακοπής ώστε το πρόγραμμα να συνεχίσει με την επόμενη λειτουργία.
- Το πρόγραμμα 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 διακοπή (interrupt) ο χρονιστής (timer), (γενικά να τελειώσει και να αρχίσει από την αρχή). Ο τύπος για τον υπολογισμό είναι ο παρακάτω:

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

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

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

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

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

9 Παράδειγμα 4: Λειτουργία του Μετατροπέα Αναλογικού σε Ψηφιακό (Analog to Digital Converter – ADC)

Ο Μετατροπέας Αναλογικού σε Ψηφιακό (Analog to Digital Converter – ADC) είναι ένα σύστημα το οποίο μετατρέπει αναλογικά σήματα που έρχονται ως είσοδοι στα κατάλληλα PINs, όπως ήχος, ρεύμα, φως, κτλ., σε ψηφιακά σήματα. 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 επίπεδα (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 (single-conversion mode) 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 RES=10) θα ενεργοποιείται μια διακοπή (interrupt). Στη διακοπή (interrupt) θα ανάβει ένα LED για T=5ms και μετά θα επιστρέφουμε στην αρχική ροή και θα περιμένουμε ξανά την μέτρηση του ADC.

Για την ενεργοποίηση του ADC:

1. Πρώτα το resolution ορίζεται σε 10-bit ή 8-bit → Resolution Selection bit (RESSEL) στον Control A register (ADCn.CTRLA).
2. Έπειτα, το Free-Running Mode ενεργοποιείται→ ‘1’ στο Free-Running bit (FREERUN) στο ADCn.CTRLA. Στην περίπτωση που ζητείται να εκτελεστεί η προκαθορισμένη λειτουργία του ADC (single-conversion mode), δηλαδή να εκτελεστεί μία μόνο μετατροπή όποτε ζητηθεί, αυτό το ψηφίο (bit) δεν ενεργοποιείται και παραμένει ‘0’.
3. Ορίζεται το ψηφίο (bit) με το οποίο θα συνδεθεί ο ADC → MUXPOS bit field στο MUXPOS register (ADCn.MUXPOS).
4. Ο ADC ενεργοποιείται→ Γράφουμε ‘1’ στο ENABLE bit στο ADCn.CTRLA.
5. Δίνεται η επιλογή της ενεργοποίησης του 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 κατώφλι (threshold). Για να ενεργοποιηθεί αυτή η λειτουργία πρέπει να ακολουθηθούν τα παρακάτω βήματα:

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

Εφόσον προγραμματιστεί κατάλληλα ο ADC και οι λειτουργίες του, απομένει να γραφτεί 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). Αυτή η εντολή πρέπει να εκτελεστεί μόνο εφόσον προγραμματιστεί εξ’ ολοκλήρου ο ADC με τους παραπάνω τρόπους που εξηγήθηκαν. Στο Free-Running Mode αρκεί να ενεργοποιηθεί μία φορά αυτή η εντολή ώστε συνεχώς να γίνονται μετατροπές νέων τιμών από αναλογικό σε ψηφιακό. Στην προκαθορισμένη λειτουργία (single-conversion mode) κάθε φορά που απαιτείται μία μετατροπή νέων τιμών, πρέπει να εκτελείται η εντολή Start Conversion ώστε ο ADC να μετατρέψει το καινούριο σήμα εισόδου από αναλογικό σε ψηφιακό.

<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:

```
#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){

		;

	}

}

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 0.14 : Κώδικας Παραδείγματος 4

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

10. Παράδειγμα 5: Λειτουργία του Παλμοευρικού Διαμορφωτή (Pulse-Width Modulator – PWM)

Ο Παλμοευρικός Διαμορφωτής (Pulse-Width Modulator – PWM) 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 μια κυματομορφή (waveform) η οποία συνδέεται με το PIN ενός περιφερειακού. Συγκεκριμένα, η έξοδος της κυματομορφής (waveform output) είναι διαθέσιμη στο PORT που επιλέγεται και μπορεί να χρησιμοποιηθεί ως έξοδος (αρκεί να οριστεί το PORT ως έξοδος).

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

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

1 Αρχικά, πρέπει να οριστεί ο prescaler του χρονιστή (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 χρονιστής (timer) → TCA0.SINGLE.PER = value.

3 Ορίζεται ο κύκλος λειτουργίας (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 TCA0.SINGLE.PER = 0x0002 και το μέγιστο είναι TCA0.SINGLE.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 οι κατάλληλες διακοπές (interrupts). Η λειτουργία αυτή ορίζεται με τον ακόλουθο τρόπο:

- Ενεργοποιώντας το Overflow Interrupt, δηλαδή εκτελείται διακοπή όταν ο χρονιστής (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 χρονιστής (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 χρονιστές (timers) TCB που έχουν λειτουργία 8-bit PWM (σελ.239-242).

Ο κώδικας της υλοποίησης του παραδείγματος παρουσιάζεται παρακάτω:

```
#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){
		;
	}
}


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

```
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 0.15 : Κώδικας Παραδείγματος 5

Εργαστηριακή Άσκηση 01:

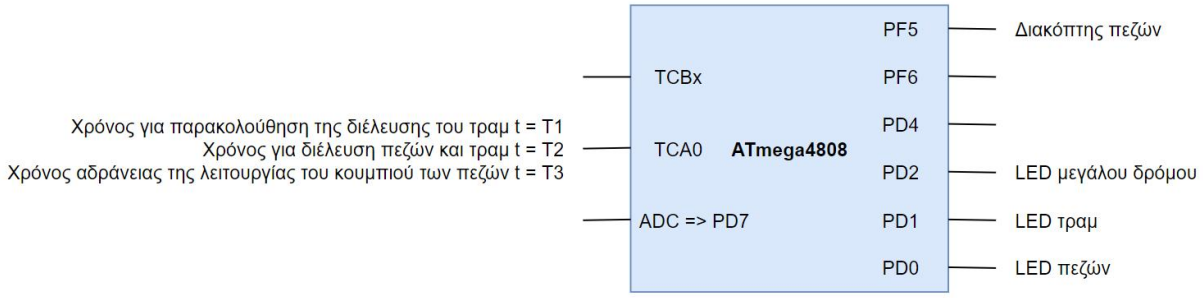
Φανάρια Κυκλοφορίας (Traffic Lights)

1 Περιγραφή

Σκοπός της Εργαστηριακής Άσκησης είναι η προσομοίωση της λειτουργία μιας διασταύρωσης αυτοκινητόδρομου, η οποία αποτελείται από ένα μεγάλο δρόμο και έναν κάθετο σιδηρόδρομο. 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). Τραμ περνάει από τον σιδηρόδρομο ανά τακτά χρονικά διαστήματα t = T1 και διακόπτει την κανονική κυκλοφορία, δηλαδή το κόκκινο φανάρι για το μεγάλο δρόμο και το πράσινο για τους πεζούς ενεργοποιείται για ένα χρονικό διάστημα t = T2. Επίσης, όταν πατηθεί το κουμπί των πεζών, ανάβει το κόκκινο φανάρι για τα αυτοκίνητα, για τον μεγάλο δρόμο μαζί με το πράσινο φανάρι για τους πεζούς, για το ίδιο χρονικό διάστημα t = T2. Μετά το πέρας του διαστήματος t = T2, το κόκκινο φανάρι των πεζών και το πράσινο φανάρι του μεγάλου δρόμου ενεργοποιούνται. Για να μπορέσει να πατηθεί ξανά το κουμπί των πεζών (δηλαδή να ενεργοποιηθεί το πράσινο φανάρι για τους πεζούς) πρέπει να έχει περάσει ένα χρονικό διάστημα t = T3. Στην περίπτωση που πατηθεί το κουμπί του φαναριού των πεζών και είναι ήδη ενεργό (είτε έχει πατηθεί ήδη μια φορά είτε περνάει το τραμ στο σιδηρόδρομο) η λειτουργία που εκτελείται δεν πρέπει να διακοπεί και τα ενεργά χρονικά διαστήματα πρέπει να διατηρηθούν μέχρι την ολοκλήρωσή τους.

Για την υλοποίηση της Εργαστηριακής Άσκησης 1, θεωρείστε ότι:

- Το φανάρι είναι πράσινο όταν το LED είναι ανοιχτό (λογικό ‘0’) και κόκκινο όταν είναι σβηστό (λογικό ‘1’). Τα τρία PIN του PORTD που χρησιμοποιούνται είναι τα PIN0, PIN1 και PIN2. Για την προσομοίωση του τραμ, το αντίστοιχο LED (PIN1) θα είναι ενεργοποιημένο όταν το τραμ περνάει από το σημείο αυτό του σιδηρόδρομου και απενεργοποιημένο σε διαφορετική περίπτωση.
- Με διακοπή (interrupt) πρέπει να υλοποιηθεί το πάτημα του κουμπιού των πεζών. Η χρήση του PIN5 του PORTF θα χρησιμοποιηθεί.
- Ο χρονιστής/μετρητής TCA0 μπορεί να χωριστεί σε δύο μικρότερους χρονιστές/μετρητές των 8-bit ο καθένας, με το σετ των εντολών TCA0.SPLIT. Ο ένας από αυτούς θα χρησιμοποιηθεί για τα δύο χρονικά διαστήματα t = T2 και t = T3. Ο δεύτερος για το χρονικό διάστημα t = T1 το οποίο ορίζει κάθε πότε περνάει τραμ, με την αντίστοιχη λειτουργία να ενεργοποιείται. Ορίστε εσείς τις μονάδες χρόνου, που σας διευκολύνουν, για την προσομοίωση, αλλά εξηγήστε τον τρόπο σκέψης και τη διαδικασία του υπολογισμού των τιμών (values).



Σχήμα 1.1 : Απεικόνιση ATmega4808, Εργαστηριακής Άσκησης 01

Χρήσιμη οδηγία: Δείτε τα παραπάνω παραδείγματα 1, 2, και 3.

2 Ερωτήματα Εργαστηριακής Άσκησης 1

1 Υλοποιήστε την λειτουργία της διαχείρισης των φαναριών του μεγάλου δρόμου και των πεζών, χρησιμοποιώντας όλες τις διαδικασίας που εξηγήθηκαν.

2 Προσθέστε την λειτουργία διαχείρισης του τραμ και του χρονικού διαστήματος ανά το οποίο περνάει από το συγκεκριμένο σημείο του σιδηρόδρομου.

3 Αναφορά Εργαστηριακής Άσκησης 1

1 Αναπτύξτε τον κώδικα σας και βεβαιωθείτε πως όλες οι λειτουργίες επιτελούνται, όπως προβλέπεται.

2 Παραδίδετε αναλυτική περιγραφή, με τη λειτουργία του κώδικά σας, μαζί με διάγραμμα ροής. Εξηγήστε τον τρόπο με τον οποίο ο κώδικάς σας εκτελεί όλες τις διαδικασίες, της εφαρμογής που αναλύθηκε παραπάνω, καθώς και τα πιθανά στοιχεία (interrupts, timers, κ.α.), που επιλέξατε και χρησιμοποιήσατε.

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

Εργαστηριακή Άσκηση 02:

Έξυπνη Οικιακή Συσκευή, με Κίνηση στο Χώρο

1 Περιγραφή

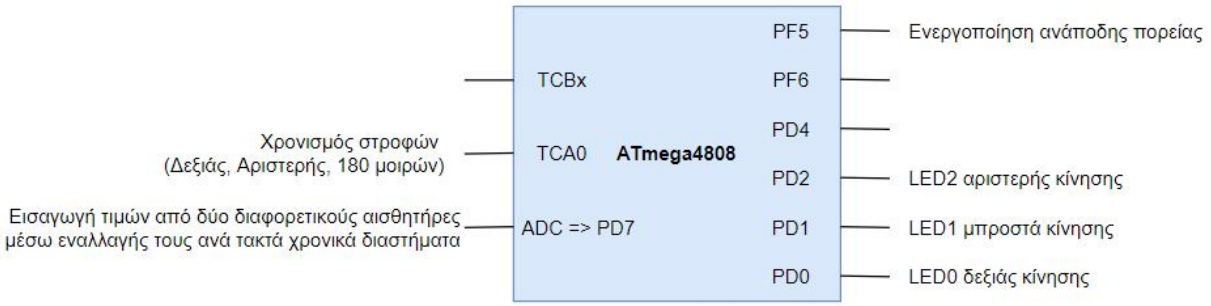
Σκοπός αυτής της εργαστηριακής άσκησης είναι η προσομοίωση της λειτουργίας μιας έξυπνης οικιακής συσκευής που κινείται στον χώρο ενός άδειου δωματίου. Ξεκινάει από μία γωνία του δωματίου και ο σκοπός της είναι να σχεδιάσει το περίγραμμά του. Καθώς κινείται στον χώρο θα παίρνει τιμές από έναν αισθητήρα που μετράει την απόσταση της συσκευής από ένα εμπόδιο μπροστά της (εδώ οι τιμές του αισθητήρα θα μετρούνται από τον Μετατροπέα Αναλογικού σε Ψηφιακό, Analog to Digital Converter – 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 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, η συσκευή πρέπει να σταματήσει. Επίσης, αν πατηθεί ένας διακόπτης (switch), η συσκευή πρέπει να εκτελέσει στροφή 180 μοιρών και να γυρίσει πίσω στην θέση της ακολουθώντας την πορεία που έχει κάνει μέχρι τώρα.

Κάποιες παραδοχές που θα χρησιμοποιηθούν είναι οι ακόλουθες:

- Η κίνηση θα προσομοιωθεί με ένα LED (όταν κινείται ευθεία το LED1 είναι ανοιχτό αλλιώς κλείνει).
- Υπάρχει μόνο ένας ADC διαθέσιμος στην πλακέτα, επομένως για να χρησιμοποιηθεί σε δύο διαφορετικές λειτουργίες (έλεγχος αν πλησιάζει σε τοίχο και έλεγχος αν υπάρχει τοίχος δεξιά στην κανονική πορεία ή αριστερά στην περίπτωση της ανάποδης πορείας) θα πρέπει να δέχεται τιμές από κάθε αισθητήρα εναλλάξ ανά τακτά χρονικά διαστήματα. Συγκεκριμένα, αρχικά θα δέχεται συνεχόμενα τιμές από τον πλαϊνό αισθητήρα (Free-Running Mode) και μόλις περάσουν T1=3s τότε θα δεχτεί μία και μόνο τιμή από τον μπροστά αισθητήρα (Single Conversion). Μόλις περάσουν T2= 2s, ο ADC θα επιστρέψει στην πρώτη του λειτουργία, δηλαδή να δέχεται τιμές από τον πλαϊνό αισθητήρα (Free-Running Mode). Αυτές οι διαδικασίες θα πρέπει συνεχώς να εκτελούνται μέχρι την ολοκλήρωση του περιγράμματος του δωματίου.
- Η δεξιά και αριστερή στροφή θα προσομοιωθούν με δύο διαφορετικά LED (LED0 και LED2 αντίστοιχα) τα οποία είναι αναμμένα μέχρι ένας χρονιστής/μετρητής (TCA0 timer/counter) φτάσει μια συγκεκριμένη τιμή (σας παροτρύνουμε να βάλετε τιμές που σας βολεύουν και να εξηγήστε πως βρήκατε τα values).
- Όταν πατηθεί ένας διακόπτης (Switch 5) θα ενεργοποιείται η ανάποδη πορεία.

Όσον αφορά την ανάποδη πορεία, η συσκευή πρέπει να σταματήσει και να γυρίσει 180 μοίρες σε οποιοδήποτε σημείο του δωματίου πατηθεί ο διακόπτης (Switch 5). Αυτό θα προσομοιωθεί με τα τρία LEDs (LED 0, 1, 2) ταυτόχρονα ενεργοποιημένα για ένα χρονικό διάστημα (χρήση χρονιστή – timer). Έπειτα, θα εκτελεί την ίδια ακριβώς διαδικασία αλλά ανάποδα.

Συγκεκριμένα, καθώς προχωράει μπροστά και ο ADC δεχθεί τιμή μικρότερη του κατωφλίου, η συσκευή θα στρίψει δεξιά (LED0). Ο ADC συνεχίζει να δέχεται τιμές από τον μπροστά αισθητήρα με λειτουργία Single Conversion. Όταν καταλάβει ότι δεν υπάρχει τοίχος αριστερά, δηλαδή ο ADC δεχτεί τιμή μεγαλύτερου του κατωφλίου που έχει οριστεί από τον αριστερό αισθητήρα, η συσκευή κινείται αριστερά (LED2). Ο αριστερός αισθητήρας λειτουργεί με τον ίδιο τρόπο όπως και ο δεξιός αισθητήρας. Επομένως, για την αριστερή στροφή αυτή τη φορά ο ADC θα βρίσκεται σε λειτουργία Free-Running Mode. Τέλος, όταν καταλάβει ότι επέστρεψε στην αρχική της θέση, τερματίζει.



Σχήμα 2.1 : Απεικόνιση ATmega4808, Εργαστηριακής Άσκησης 02

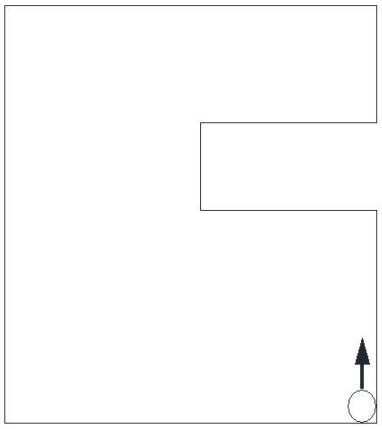
Χρήσιμη οδηγία: Δείτε το παραπάνω παράδειγμα 4.

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). Ωστόσο, οι λειτουργίες με τους χρονισμούς θα χρειαστούν για την προσομοίωση των στροφών.

2 Υλοποιήστε τον κώδικα της κίνησης για ένα τυχαίο δωμάτιο που περιέχει και δύο αμβλείες γωνίες, εδώ εισάγεται και η δεύτερη λειτουργία του ADC. Στο παρακάτω σχήμα 2.2 παρουσιάζεται ένα παραδειγματικό τέτοιο δωμάτιο για την σωστή κατανόηση του προβλήματος. Ωστόσο, η συσκευή θα πρέπει να μπορεί να ανταποκριθεί σωστά και σε ένα άγνωστο δωμάτιο.

3 Υλοποιήστε την ανάποδη λειτουργία.



Σχήμα 2.2 : Σχήμα δωματίου

3 Αναφορά Εργαστηριακής Άσκησης 2

1 Αναπτύξτε κώδικα και βεβαιωθείτε πως όλες οι λειτουργίες εκτελούνται όπως προβλέπεται.

2 Παραδίδεται αναλυτική αναφορά με τη λειτουργία του κώδικά σας, μαζί με διάγραμμα ροής. Εξηγήστε τον τρόπο με τον οποίο ο κώδικάς σας εκτελεί όλες τις διαδικασίες, της εφαρμογής που αναλύθηκε παραπάνω, καθώς και τα πιθανά στοιχεία (interrupts, timers, ADC, κτλ.) που επιλέξατε και χρησιμοποιήσατε.

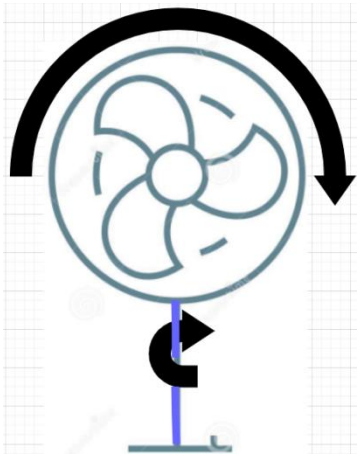
**** Σημείωση: Παραδίδετε τον κώδικα ανά ομάδα και τα αναλυτικά σχόλια, για τις εντολές που χρησιμοποιήσατε.*

Εργαστηριακή Άσκηση 03:

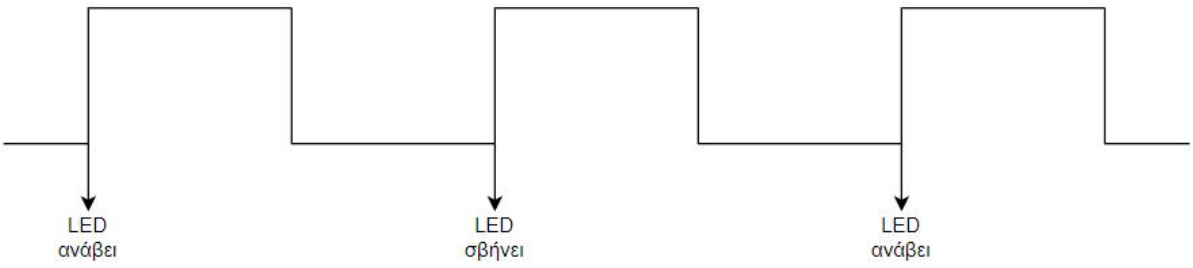
Εξοικείωση με την Παλμοευρική Διαμόρφωση (Pulse-Width Modulation – PWM)

1 Περιγραφή

Σε αυτή την εργαστηριακή άσκηση, θα προσομοιωθεί η κίνηση ενός ανεμιστήρα. Ένας ανεμιστήρας αποτελείται από δύο περιστροφικές κινήσεις, μία κυκλική κίνηση των λεπίδων και μία κυκλική κίνηση της βάσης, ώστε ο ανεμιστήρας να μπορεί να περιστρέφεται και να καλύπτει περισσότερο χώρο, όπως απεικονίζεται και στο σχήμα 3.1. Οι δύο αυτές κυκλικές κινήσεις θα προσομοιωθούν με δύο διαφορετικούς ρυθμούς, που θα καθορίζονται από δύο διαφορετικούς Παλμοευρικούς Διαμορφωτές (Pulse-Width Modulators – PWMs), (μπορείτε να χρησιμοποιήσετε όποιον καταχωρητή σας διευκολύνει). Ο ρυθμός κάθε κίνησης θα εμφανίζεται σε ένα LED (LED0 και LED1), 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 στην κίνηση των λεπίδων και το LED1 (PORTD PIN1) στην κίνηση της βάσης.



Σχήμα 3.1 : Δύο κινήσεις ανεμιστήρα.



Σχήμα 3.2 : Αναπαράσταση λειτουργίας LED μέσω του PWM.

Όταν ενεργοποιηθεί ένας διακόπτης (switch 5 του PORTF), ο ανεμιστήρας ενεργοποιείται, δηλαδή οι ακόλουθες ενέργειες εκτελούνται:

- Η κυκλική κίνηση της βάσης ενεργοποιείται, η οποία προσομοιώνεται με ένα παλμό περιόδου Tb = 2 ms και κύκλο λειτουργίας (duty cycle) Db = 40%. Ο παλμός αυτός οδηγεί το LED1 του PORTD μέσω των ανερχόμενων παρυφών (rising edges).

- Η κυκλική κίνηση των λεπίδων ενεργοποιείται, η οποία προσομοιώνεται με έναν παλμό περιόδου Tl = 1 ms και κύκλο λειτουργίας (duty cycle) Dl = 50%. Ο παλμός αυτός οδηγεί το LED0 του PORTD μέσω των ανερχόμενων παρυφών (rising edges).

- Ο Μετατροπέας Αναλογικού σε Ψηφιακό (Analog to Digital Converter – ADC) ενεργοποιείται και όταν εντοπίσει κάποια τιμή μικρότερη ενός προκαθορισμένου κατωφλίου (threshold), δηλαδή όταν κάποιο αντικείμενο εντοπιστεί πολύ κοντά στις λεπίδες που περιστρέφονται, ο ανεμιστήρας απενεργοποιείται άμεσα και ένα τρίτο LED (LED2 του PORTD) ενεργοποιείται. Για να λειτουργήσει ξανά ο ανεμιστήρας πρέπει να ενεργοποιηθεί ο διακόπτης (switch 5 του PORTF) και να απενεργοποιηθεί το LED2 του PORTD.

Όταν ενεργοποιηθεί δεύτερη φορά ο ίδιος διακόπτης (switch 5 του PORTF) μετά την εκκίνηση της λειτουργίας του ανεμιστήρα, η περίοδος του παλμού της κυκλικής κίνησης των λεπίδων διπλασιάζεται με κύκλο λειτουργίας (duty cycle) Tl = 50%. Ο ADC και ο παλμός της κυκλικής κίνησης της βάσης συνεχίζουν τη λειτουργία τους όπως περιγράφηκε παραπάνω. Αν ο διακόπτης αυτός πατηθεί τρίτη φορά, τότε ο ανεμιστήρας απενεργοποιείται και οι δύο παλμοί μαζί με τον ADC σταματούν τη λειτουργίας τους.

Σχήμα 3.3 : Απεικόνιση ATmega4808, Εργαστηριακής Άσκησης 03

<u>Παρατήρηση:</u> Καθώς υπάρχουν διαφορετικές ρουτίνες διακοπών (ISR) για τον καθένα παλμό, πρέπει να δοθεί προσοχή πότε οι παλμοί έρχονται σε ανερχόμενη παρυφή (rising edge). Δύο ρουτίνες διακοπών δεν μπορούν να εκτελούνται ταυτόχρονα, επομένως για να μην χαθεί η συχνότητα των LEDS, οι δύο παλμοί είτε δεν πρέπει να έρχονται ταυτόχρονα σε ανερχόμενη παρυφή, είτε θα χρησιμοποιηθεί μία ISR μέσα στην οποία θα ελέγχεται η ενεργοποίηση της δεύτερης ISR μέσω των αντίστοιχων flags των καταχωρητών INTFLAGS.

Παρακάτω θα αναφερθούν κάποιες πληροφορίες αναφορικά με τη λειτουργία PWM στον μικροελεγκτή. Αρχικά, πολλοί διαφορετικοί χρονιστές/μετρητές (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 ως δύο 8-bit

timers/counters, οι οποίοι δημιουργούν δύο διαφορετικά PWMs (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. *****

Χρήσιμη οδηγία: Δείτε το παραπάνω παράδειγμα 5.

2 Ερωτήματα Εργαστηριακής Άσκησης 3

1 Υλοποιείστε τη λειτουργία ενεργοποίησης του ανεμιστήρα μετά την ενεργοποίηση του διακόπτη (switch 5), δηλαδή οι δύο παλμοί PWM αρχικοποιούνται και οδηγούν τα δύο LEDs.

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

3 Προσθέστε τη δεύτερη λειτουργία του παλμού της κυκλικής κίνησης των λεπίδων μέσω της ενεργοποίησης του διακόπτη (switch 5). Επίσης, προσθέστε τη λειτουργία απενεργοποίησης του ανεμιστήρα.

3 Αναφορά Εργαστηριακής Άσκησης 3

1 Αναπτύξτε τον κώδικα σας και βεβαιωθείτε πως όλες οι λειτουργίες εκτελούνται σωστά, όπως περιγράφεται στην εργαστηριακή άσκηση.

2 Παραδίδετε αναλυτική περιγραφή στην αναφορά σας, με τη λειτουργία του κώδικά σας, μαζί με διάγραμμα ροής. Εξηγήστε τον τρόπο με τον οποίο ο κώδικά σας εκτελεί όλες τις διαδικασίες, της εφαρμογής που αναλύθηκε παραπάνω, καθώς και τα πιθανά στοιχεία (interrupts, PWM, ADC, κτλ.) που επιλέξατε και 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