

# MICROCHIP STUDIO

AVR PROGRAMMING

Κυριακή Τσαντικήδου  
e-mail: k DOT tsantikidou AT upatras DOT gr

# Διάρθρωση Παρουσίασης

1. Εισαγωγή
2. Εξοικείωση με το εργαλείο
3. Μηχανισμοί Λειτουργίας και Παραδείγματα
4. Πρώτη Εργαστηριακή Άσκηση
5. Βιβλιογραφία

# Εισαγωγή

Ενότητα 1


# Εισαγωγή (1/2)

- ▶ Ροή Εργαστηριακών Ομάδων
  - ▶ Το Εργαστήριο ξεκινάει την εβδομάδα στις **20.03** με την ομάδα Α και την Εργαστηριακή Άσκηση 1.
  - ▶ Την εβδομάδα από **27.03** συνεχίζει η ομάδα Β με την Εργαστηριακή Άσκηση 1.
  - ▶ Συνεχίζει η ίδια λογική ανά βδομάδα.
- ▶ Θα παραδίδεται αναφορά ανά ομάδα εργασίας (**2 άτομα**).
- ▶ Δίνεται δυνατότητα για **ΜΙΑ** απουσία κατά το μέγιστο, η οποία μπορεί να αναπληρωθεί στο τέλος.
- ▶ Σε περίπτωση απουσιών το Εργαστήριο θα πρέπει να επαναληφθεί στο σύνολό του.
- ▶ Η πρώτη εργαστηριακή άσκηση είναι βασική και **ΔΕΝ** θα αναπληρωθεί.

# Εισαγωγή (2/2)

- ▶ Παράδοση εργαστηριακών ασκήσεων ηλεκτρονικά σε αντίστοιχες ομάδες στο e-class με χρονικό περιθώριο **ΜΙΑΣ** ημέρας μετά την ολοκλήρωση της εργαστηριακής ομάδας (ώρας).
  - ▶ π.χ. Η Εργαστηριακή Ομάδα Α1 -> Τρίτη, ώρα 18:00-20:00 παραδίδει την εργαστηριακή της άσκηση μέχρι την επόμενη μέρα, Τετάρτη, ώρα 18:00.
- ▶ Η παρουσίαση της εργαστηριακής σας άσκησης κατά την διάρκεια του εργαστηρίου προσμετράται στην τελική σας βαθμολογία.
- ▶ Ο προβιβάσιμος βαθμός υπολογίζεται τόσο με τον μέσο όρο των αναφορών των εργαστηριακών ασκήσεων, όσο και με την τελική εξέταση του Εργαστηρίου.

# ΕΠΙΚΟΙΝΩΝΙΑ




- Ενεργά εργαλεία
  - Ανακοινώσεις
  - Εγγραφα
  - Ομάδες Χρηστών
- Ανενεργά εργαλεία
- Διαχείριση μαθήματος

## Εργαστήριο Προηγμένοι Μικροεπεξεργαστές

Δρ. Βησσαρίων Φερεντίνος

Περιγραφή



8ο Εξάμηνο, υποχρεωτικό μάθημα, 2 ώρες θεωρία και 2 ώρες εργαστήριο.

Βασικές έννοιες, θεμελιώδεις ορισμοί, συσχεδίαση υλικού και λογισμικού, παράγοντες συσχεδίασης, μοντελοποίηση, ροή δεδομένων, υλοποίηση σε υλικό, υλοποίηση σε λογισμικό, ροή ελέγχου: σχεδιασμός & υλοποίηση, μηχανές πεπερασμένων καταστάσεων, μικροπρογραμματισμός, ενσωματωμένοι πυρήνες γενικού σκοπού, κατηγορίες μικροεπεξεργαστών, οργάνωση προγράμματος, ποιότητα κώδικα-μεταγλώττισης, συστήματα σε υλικό, αρχές σχεδίασης, αρχιτεκτονικές σχεδίασης, δίαυλοι σε υλικό, δίκτυα σε υλικό, περιβάλλοντα διεπαφής υλικού και λογισμικού, σχεδιασμός μικροεπεξεργαστών, σχεδιασμός συν-επεξεργαστών, σύγχρονες τεχνολογίες, σύγχρονοι & μελλοντικοί μικροεπεξεργαστές/συν-επεξεργαστές, προχωρημένα θέματα και εφαρμογές.

Πληροφορίες

**Επικοινωνία**

Διδάσκοντες 2022-23

Δρ. Βησσαρίων Φερεντίνος,  
Εντεταλμένος Διδάσκων  
e-mail: ferentinosv@upatras.gr

Κυριακή Τσαντικίδου,  
e-mail: k.tsantikidou@upatras.gr

Μάριος Τσάβος,  
e-mail: tsavos@ceid.upatras.gr

# Εξοικείωση με το Εργαλείο

Ενότητα 2

# Εισαγωγή στο AVR-IoT Wx Development Board

- ▶ Το AVR-IoT Wx Development Board είναι μια μικρή και εύκολα επεκτάσιμη πλατφόρμα επίδειξης και ανάπτυξης IoT εφαρμογών.
- ▶ Βασισμένη στην αρχιτεκτονική μικροελεκτή AVR που χρησιμοποιεί τεχνολογία Wi-Fi.
- ▶ Μια IoT εφαρμογή μπορεί να απλοποιηθεί χωρίζοντας το πρόβλημα σε τρία blocks:
  - ▶ Έξυπνο → ATmega4808 microcontroller
  - ▶ Ασφαλές → ATECC608A secure element
  - ▶ Διασύνδεση → ATWINC1510 Wi-Fi controller module

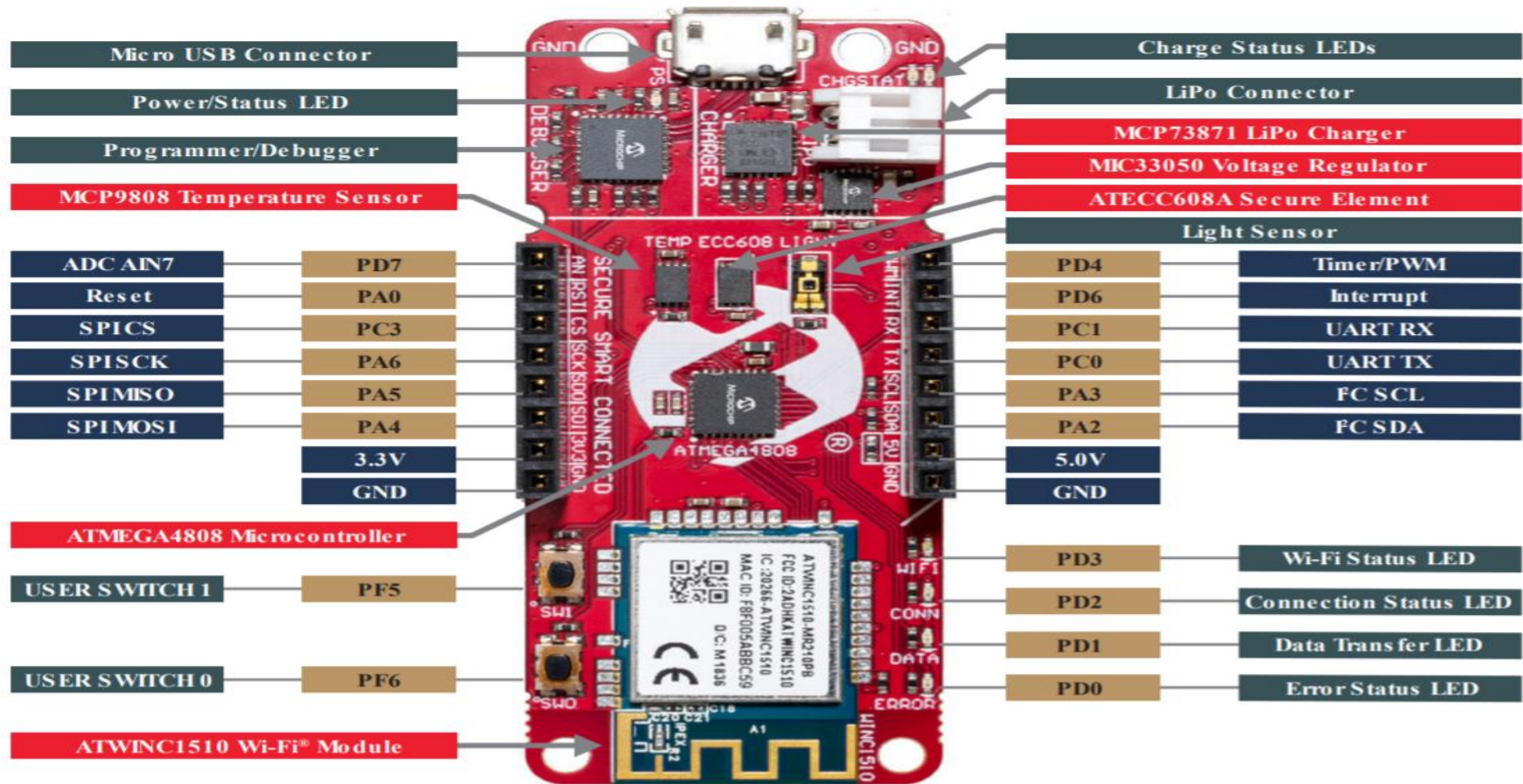
Περισσότερες πληροφορίες ανατρέξτε στα παρακάτω pdf:

<https://ww1.microchip.com/downloads/en/DeviceDoc/AVR-IoT-Wx-Hardware-User-Guide-DS50002805C.pdf>

<https://ww1.microchip.com/downloads/en/DeviceDoc/ATmega4808-09-DataSheet-DS40002173B.pdf>



# Board Overview



# Εισαγωγή στο Microchip Studio

- ▶ Το Microchip Studio (ή Atmel® Studio 7) είναι μια ολοκληρωμένη πλατφόρμα (Integrated Development Platform - IDP) για την ανάπτυξη και το debugging εφαρμογών με AVR® και SAM microcontroller (MCU).
- ▶ Εύχρηστο περιβάλλον για την σύνταξη, την κατασκευή και τον εντοπισμό σφαλμάτων εφαρμογών που είναι γραμμένα σε C/C++ ή σε assembly.
- ▶ Δυνατότητα εισαγωγής σχεδίων σε Arduino ως C/C++ projects και υποστήριξη 500+ AVR και SAM devices.
- ▶ Τεράστια βιβλιοθήκη πηγαίου κώδικα με 1600+ παραδείγματα εφαρμογών.

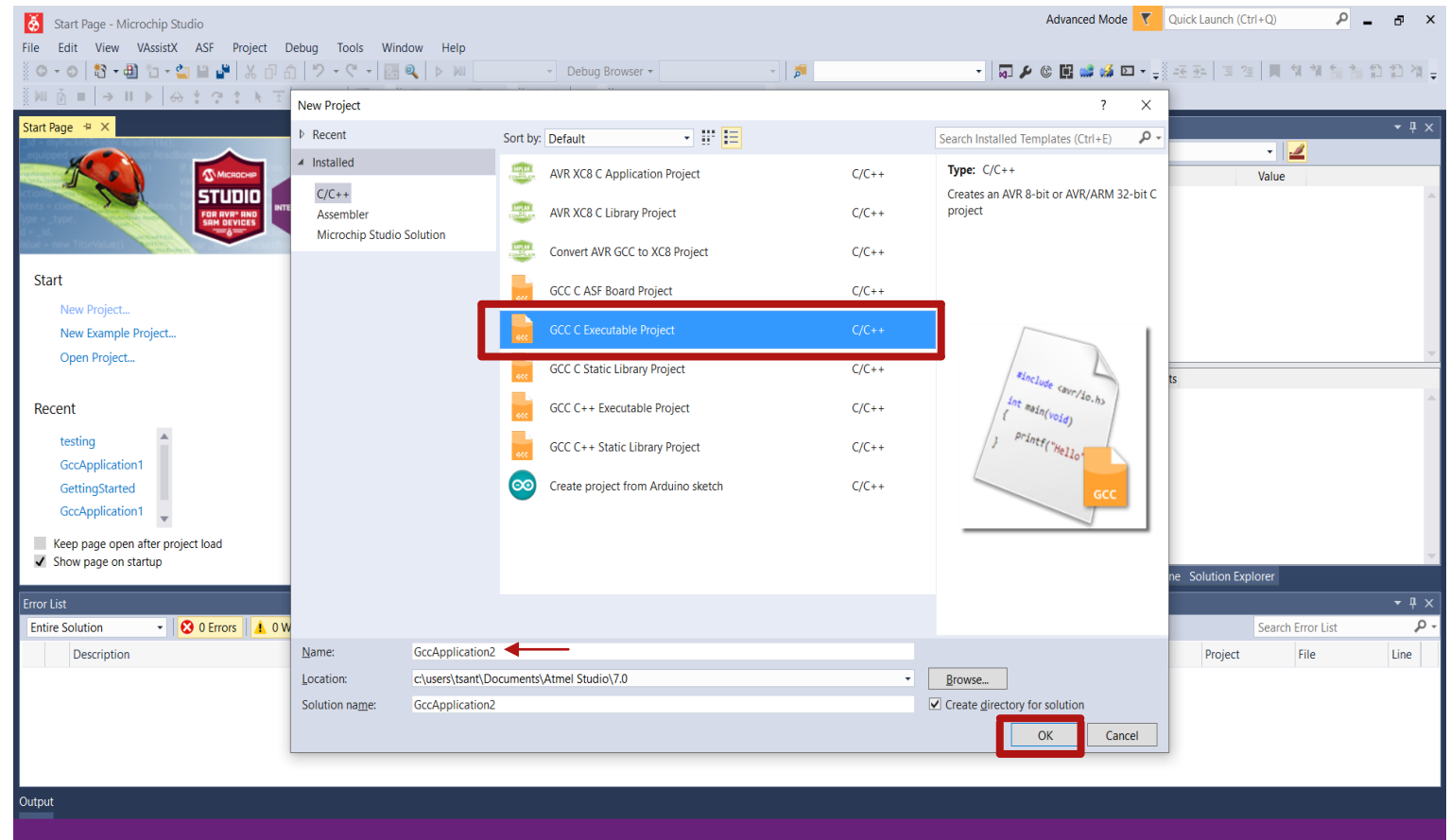
Περισσότερες πληροφορίες στην παρακάτω ιστοσελίδα: <https://www.microchip.com/mplab/microchip-studio>

# Εγκατάσταση Εργαλείων Simulation

- ▶ Θα βρείτε το Microchip Studio στην παρακάτω σελίδα:  
<https://www.microchip.com/mplab/microchip-studio>
- ▶ Έπειτα, πατήστε Download στην επιλογή [Microchip Studio for AVR and SAM Devices 7.0.2542 Web Installer](#).
- ▶ Αφού κατέβει, ανοίξτε το αρχείο και ακολουθήστε τις οδηγίες εγκατάστασης του φυλλαδίου εργαστηριακών ασκήσεων.

# Δημιουργία Νέου Project

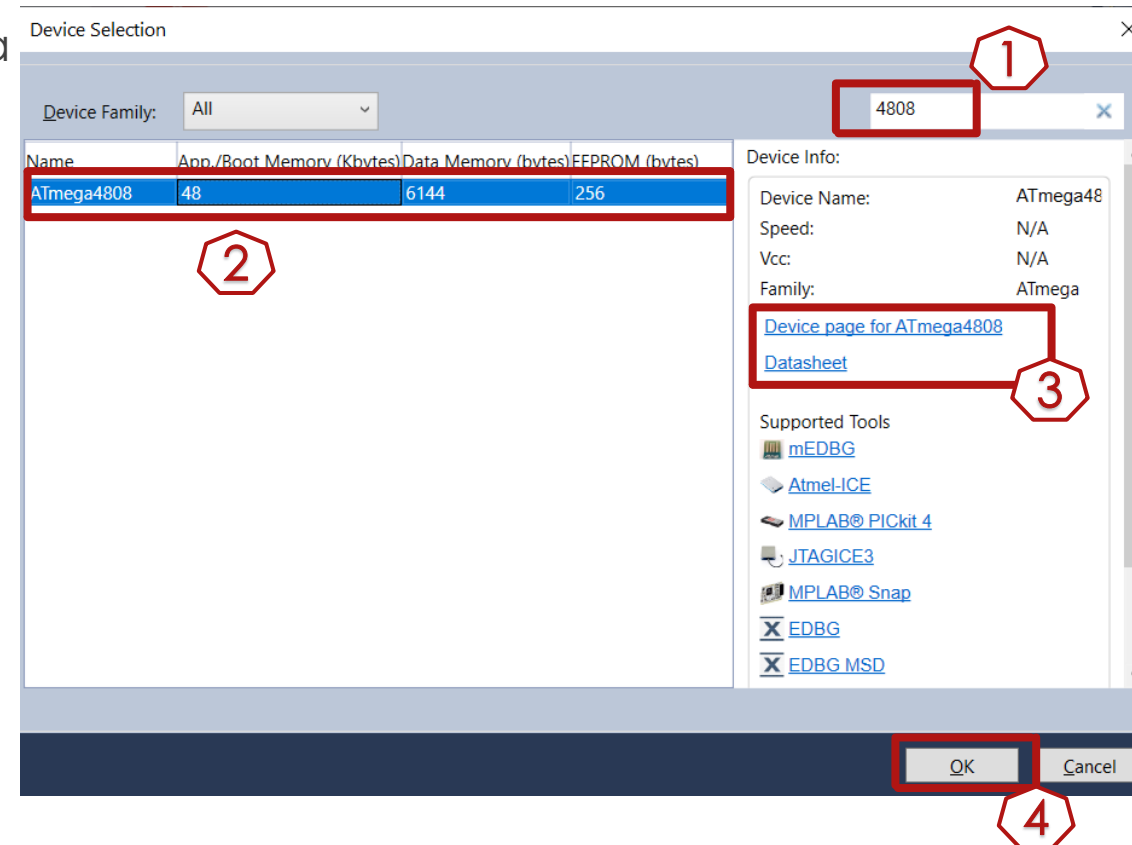
- ▶ Στο εγκαταστημένο πλέον Microchip Studio, κατευθυνθείτε στο File → New → Project.
- ▶ Οι ασκήσεις που θα υλοποιηθούν στα πλαίσια αυτού του μαθήματος θα είναι στη γλώσσα C και επομένως θα επιλέξουμε τον GCC C Executable Project.
- ▶ Επιλέγουμε το όνομα της εφαρμογής μας και πατάμε OK.



# Επιλογή Συσκευής

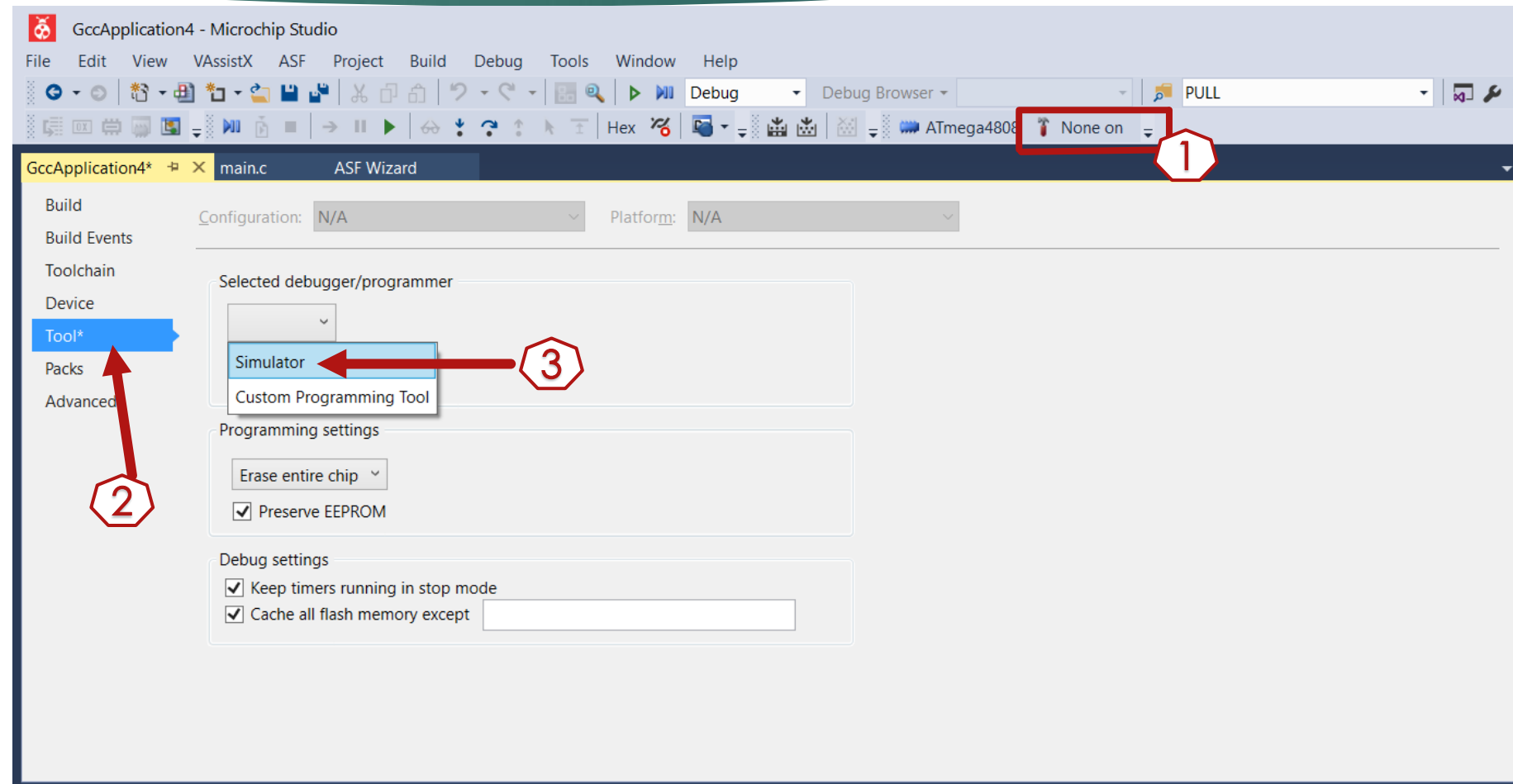
- ▶ Επιλέγουμε τον ATmega4808 ακολουθώντας τα βήματα όπως στην εικόνα.
- ▶ Μπορούμε να δούμε κατευθείαν το datasheet και το device page του συγκεκριμένου microcontroller για περισσότερες πληροφορίες σχετικά με τις δυνατότητές του (βήμα 3).
- ▶ Αφού επιλέξουμε την συσκευή μας, πατάμε OK (βήμα 4) και είμαστε πλέον έτοιμοι να αρχίσουμε να προγραμματίζουμε.

Για περισσότερα tutorials σχετικά με το Microchip Studio μπορείτε να βρείτε στο παρακάτω link:  
<https://www.youtube.com/watch?v=8HNG8EnAjfw&list=PL9B4edd-p2ajbkbFpi47P9PfMtQuV0OwQ&index=1>



# Επιλογή Προσομοίωσης

- Είναι σημαντικό όταν κάνουμε πρώτη φορά Debug να έχουμε επιλέξει το Simulator ως Debugger, ακολουθώντας τα βήματα όπως φαίνεται στο σχήμα.





# Περιβάλλον Εργασίας

The screenshot displays the Microchip Studio IDE interface. The main window shows a C program for an AVR microcontroller. The program includes headers for AVR and delay functions, defines a delay constant, and contains a main function that configures PORTD and enters a loop with delays.

Red arrows point to the following buttons in the toolbar:

- Run/Continue**: Points to the green play button.
- Step Over**: Points to the blue button with a right-pointing arrow.
- Step Into**: Points to the blue button with a downward-pointing arrow.

The **I/O View** on the right shows the hardware registers for PORTD. The table below lists the registers and their values:

Name	Address	Value	Bits
DIR	0x460	0x00	
DIRSET	0x461	0x00	
DIRCLR	0x462	0x00	
DIRTGL	0x463	0x00	
OUT	0x464	0x00	
OUTSET	0x465	0x00	
OUTCLR	0x466	0x00	
OUTTGL	0x467	0x00	
IN	0x468	0x00	
INTFL...	0x469	0x00	
PORT...	0x46A	0x00	
PIN0C...	0x470	0x00	
PIN1C...	0x471	0x00	
PIN2C...	0x472	0x00	
PIN3C...	0x473	0x00	
PIN4C...	0x474	0x00	

# Καταχωρητές

16

I/O		Filter:	
Name	Value		
16-bit Timer Type B (TCB0)			
16-bit Timer Type B (TCB1)			
16-bit Timer Type B (TCB2)			
16-bit Timer/Counter Type...			
TCA_SINGLE[0]			
TCA_SPLIT[0]			
Analog Comparator (AC0)			
Analog to Digital Converte...			
Bod interface (BOD)			
Clock controller (CLKCTRL)			
Configurable Custom Logi...			
CPU (CPU)			
CRCSCAN (CRCSCAN)			
Event System (EVSYS)			
Fuses (FUSE)			
General Purpose IO (GPIO)			
I/O Ports (PORTA)			
I/O Ports (PORTB)			
I/O Ports (PORTC)			
I/O Ports (PORTD)			
I/O Ports (PORTE)			
Name	Address	Value	Bits
DIR	0x460	0x02	00000001
DIRSET	0x461	0x02	00000001
DIRCLR	0x462	0x02	00000001
DIRTGL	0x463	0x02	00000001
OUT	0x464	0x40	00010000
OUTSET	0x465	0x40	00010000
OUTCLR	0x466	0x40	00010000
OUTTGL	0x467	0x40	00010000
IN	0x468	0x00	00000000
INTFL...	0x469	0x00	00000000
PORT...	0x46A	0x00	00000000
PIN0C...	0x470	0x00	00000000
PIN1C...	0x471	0x00	00000000
PIN2C...	0x472	0x00	00000000
PIN3C...	0x473	0x00	00000000
PIN4C...	0x474	0x00	00000000
PIN5C...	0x475	0x00	00000000
PIN6C...	0x476	0x00	00000000
PIN7C...	0x477	0x00	00000000



# Ροή Λειτουργίας Προγράμματος

Break All (Pause)

Stop Debugging

Εντολή προς  
εκτέλεση

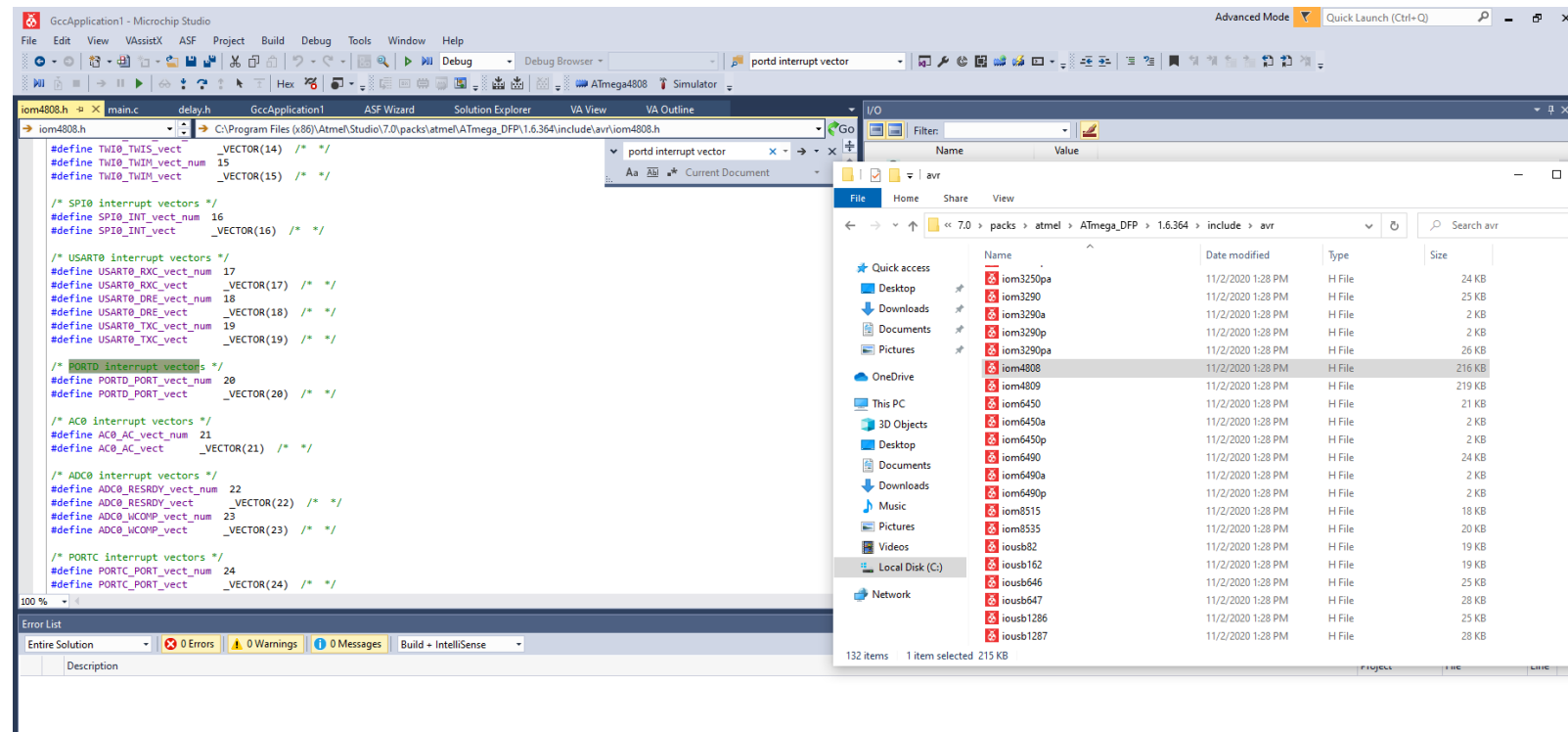
Breakpoint

The screenshot shows the Microchip Studio IDE with the following components:

- Code Editor:** Displays a C program for an ATmega4808. The code includes `<avr/io.h>` and `<util/delay.h>`. It defines `del 10` and has a `main` function that configures `PORTD` and enters a `while` loop. A breakpoint is set on the first line of the `while` loop.
- Debugger:** The status bar at the bottom indicates the program is "Stopped".
- I/O View:** On the right, a list of I/O registers is shown. The `OUT` register (address 0x464) is selected, showing its value as 0x82.
- Autos Window:** At the bottom left, the "Autos" window shows the current value of the `OUTCLR` variable as 0.

# Αρχείο “iom4808.h”

- ▶ Το header file “iom4808.h” περιέχει χρήσιμες πληροφορίες για την υλοποίηση κώδικα όπως:
  - ▶ Τιμές δηλωμένων σταθερών (PIN1\_bm)
  - ▶ Interrupt Vectors
  - ▶ Και άλλα που θα δούμε στη συνέχεια
- ▶ Θα το βρείτε στον φάκελο που έχετε εγκαταστήσει το Microchip:
   
“~\Studio\7.0\packs\atmel\ATmega\_DFP\1.6.364\include\avr\iom4808.h”



# Μηχανισμοί Λειτουργίας και Παραδείγματα

Ενότητα 3

# ΣΚΟΠΟΣ

Ο σκοπός των παρακάτω παραδειγμάτων είναι η εξοικείωσή σας με:

- ▶ 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)

# Διεύθυνση Δεδομένων (Data Direction)

- ▶ Για την ενεργοποίηση και την απενεργοποίηση ενός LED πρέπει πρώτα να οριστεί το Direction του συγκεκριμένου Pin ως Output.
- ▶ Αρχικά, το PORT που θα χρησιμοποιηθεί είναι το PORTD καθώς τα τέσσερα πρώτα PINs του συνδέονται με LEDs πάνω στο Board (όπως φαίνεται και στο Board Overview).
- ▶ Για να οριστεί ένα PIN ως Output πρέπει να τεθεί το αντίστοιχο bit του register DIR (Data Direction) σε λογικό "1".
- ▶ Με την τιμή **0** ή **1** στον register OUT (Output Value), το LED **ενεργοποιείται** ή **απενεργοποιείται** αντίστοιχα (το σήμα είναι αρνητικής λογικής).

Σημείωση: Στην Προσομοίωση οι χρόνοι του delay όπως και των timers δεν είναι αντιπροσωπευτικοί του πραγματικού χρόνου. Μέσω της υλοποίησης στην πλακέτα σχετίζονται πραγματικά οι τιμές με πραγματικό χρόνο. Για αυτό προς το παρόν ορίζονται μικροί χρόνοι για να μην υπάρχει μεγάλη καθυστέρηση.

# Υλοποίηση Πρώτου Παραδείγματος: Διαχείριση LED

```
#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
    }
}
```

- ▶ Το σύστημα ανάβει και σβήνει ένα LED με συχνότητα 10 ms.
- ▶ 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"

# Ενεργοποίηση Διακοπής μέσω Διακόπτη (Switch)

- ▶ Οι διακόπτες (switches) 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 οριστεί σε ποιο σημείο του παλμού θα ενεργοποιηθεί η μονάδα διαχείρισης του interrupt. Επιλέγεται η 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.

# Συνάρτηση Διαχείρισης Ρουτίνας (ISR Routine)

- ▶ Όταν γίνει η διακοπή, το σύστημα οδηγείται σε μία συνάρτηση η οποία θα τη διαχειριστεί.
- ▶ Αυτή η συνάρτηση είναι μια Ρουτίνα Διαχείρισης Διακοπής (ISR Routine), η οποία παίρνει ως όρισμα το ένα interrupt vector το οποίο της αντιστοιχεί.
- ▶ Για παράδειγμα το interrupt vector για τη διακοπή του PINF είναι το `PORTF_PORT_vect`.

Σημείωση: Στο αρχείο `iom4808.h` υπάρχουν όλα τα interrupt vectors που μπορούν να χρησιμοποιηθούν από το board.



# Υλοποίηση Δεύτερου Παραδείγματος: Διακοπές

```
#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(); //disable interrupts
}
```

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

    x=1;
}
```

- ▶ Το σύστημα ενεργοποιεί την ρουτίνα διαχείρισης interrupt με το πάτημα ενός switch.

# Λειτουργία Κώδικα (2<sup>ο</sup> Παράδειγμα)

- ▶ Στο σύστημά μας έχουμε ένα LED και ένα Switch.
- ▶ Η κανονική λειτουργία ορίζει το LED να παραμένει ενεργοποιημένο.
- ▶ Όταν πατηθεί το switch ενεργοποιείται το interrupt και αλλάζει η τιμή της μεταβλητής **x** ώστε να απενεργοποιηθεί το LED και να σταματήσει η ροή του προγράμματος.

# Προσομοίωση (Simulation)

- ▶ Για να ενεργοποιηθεί το interrupt πρέπει να πατηθεί το 5<sup>ο</sup> bit του register INTFLAGS στο PORTF, εφόσον το πρόγραμμα βρίσκεται σε κάποιο breakpoint.
- ▶ Μόλις το bit αλλάξει από '0' σε '1' και το πρόγραμμα συνεχίσει με την επόμενη εντολή (STEP OVER), θα ενεργοποιηθεί η συνάρτηση διαχείρισης διακοπών (ISR).

# Ενεργοποίηση Χρονιστή (Timer)

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

- ▶ Να δοθεί η τιμή '0' στον CTRLB register (Normal Mode),
- ▶ Να δοθεί η τιμή '0' στον CNT register (ο timer μηδενίζεται),
- ▶ Να δοθεί η προβλεπόμενη τιμή στον CMP0 register,
- ▶ Να ενεργοποιήσουμε τα interrupts μέσω του INTCTRL register,
- ▶ Να τεθεί το clock frequency,
- ▶ Να ενεργοποιηθεί μέσω του CTRLA register.

Σημείωση: Ανατρέξτε στη σελ. 243 στο ATmega4808 DataSheet.

# Συνάρτηση Διαχείρισης Ρουτίνας Timer

- ▶ 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 τον συγκεκριμένο timer.

# 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:

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

# Υλοποίηση Τρίτου Παραδείγματος: Χρονιστής (Timer)

```
#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
    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 = 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;
}
```

► Timer που οδηγεί σε interrupt.

# Λειτουργία Κώδικα (3ο Παράδειγμα)

- ▶ Ενεργοποιείται το LED και ο timer.
- Όταν ο timer φτάσει την τιμή που έχει οριστεί:
- ▶ Ενεργοποιείται η ρουτίνα διαχείρισης της διακοπής.
- ▶ Αλλάζει η μεταβλητή **x**.
- ▶ Το πρόγραμμα τελιώνει σβήνοντας το LED

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



# Πρώτη Εργαστηριακή Άσκηση

Ενότητα 4

# Σκοπός Εργαστηριακής Άσκησης

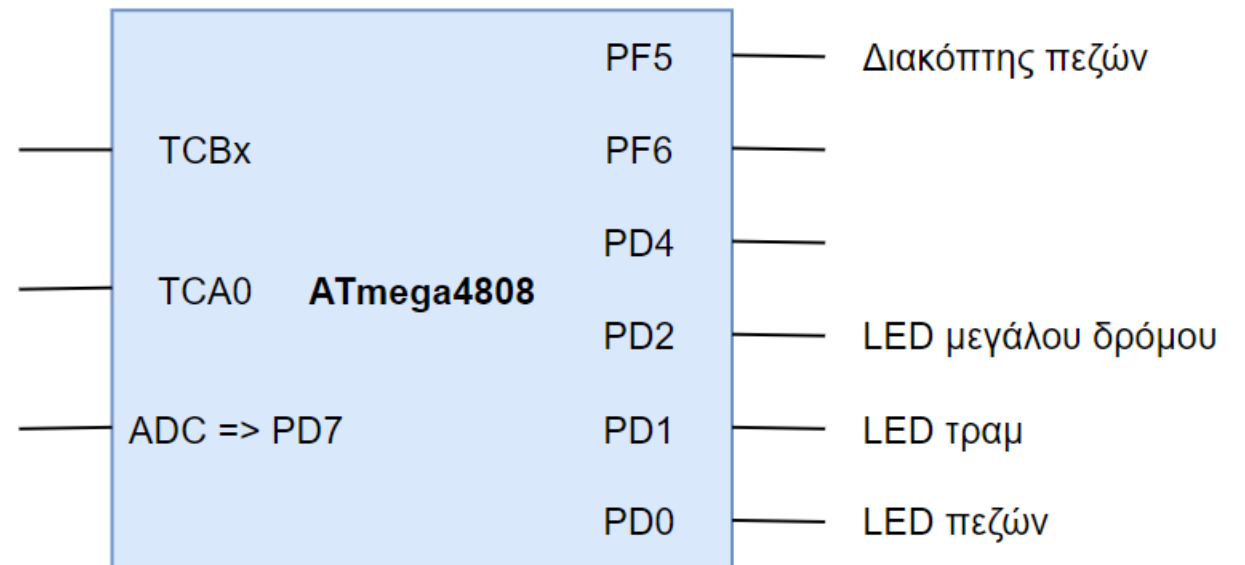
- ▶ Προσομοίωση της λειτουργία μιας διασταύρωσης αυτοκινητόδρομου με ένα μεγάλο δρόμο και έναν κάθετο σιδηρόδρομο.
- ▶ Στον μεγάλο δρόμο υπάρχει ένα φανάρι για τα αυτοκίνητα και ένα φανάρι για τους πεζούς, που ανάβει μόνο μετά από πίεση ενός κουμπιού για χρονικό διάστημα  $t = T_2$ .
- ▶ Τραμ περνάει από τον σιδηρόδρομο ανά τακτά χρονικά διαστήματα  $t = T_1$  και διακόπτει την κανονική κυκλοφορία (δηλαδή το πράσινο για τους πεζούς ενεργοποιείται για ένα χρονικό διάστημα  $t = T_2$ ).
- ▶ Η επόμενη ενεργοποίηση του πράσινου φαναριού για τους πεζούς μπορεί να πραγματοποιηθεί μόνο μετά το πέρας ενός χρονικού διαστήματος  $t = T_3$ .
- ▶ Στην περίπτωση που πατηθεί το κουμπί του φαναριού των πεζών ενώ είναι ήδη ενεργό, η λειτουργία που εκτελείται δεν πρέπει να διακοπεί.

# Υλοποίηση Άσκησης

- ▶ 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 (διάβαση πεζών), PIN1 (τραμ) και PIN2 (αυτοκινητόδρομος).
- ▶ 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 (PIN5 του PORTF).
- ▶ Δύο χρονιστές -> ένας για τα δύο χρονικά διαστήματα  $t = T2$  και  $t = T3$  και ο δεύτερος για το χρονικό διάστημα  $t = T1$  (κάθε πότε περνάει τραμ).

# Απεικόνιση ATmega4808

Χρόνος για παρακολούθηση της διέλευσης του τραμ  $t = T1$   
Χρόνος για διέλευση πεζών και τραμ  $t = T2$   
Χρόνος αδράνειας της λειτουργίας του κουμπιού των πεζών  $t = T3$



# Ερωτήματα

- ▶ Πρώτο Ερώτημα:
  - ▶ Υλοποιήστε την λειτουργία της διαχείρισης των φαναριών του μεγάλου δρόμου και των πεζών, χρησιμοποιώντας όλες τις διαδικασίες που εξηγήθηκαν.
- ▶ Δεύτερο Ερώτημα:
  - ▶ Προσθέστε την λειτουργία διαχείρισης του τραμ και του χρονικού διαστήματος ανά το οποίο περνάει από το συγκεκριμένο σημείο του σιδηρόδρομου.

# Βιβλιογραφία

# Βιβλιογραφία

- ▶ Microchip, Επίσημη ιστοσελίδα: <https://www.microchip.com/mplab/microchip-studio>
- ▶ Microchip, “AVR-IoT Wx Hardware User Guide”:  
<https://ww1.microchip.com/downloads/en/DeviceDoc/AVR-IoT-Wx-Hardware-User-Guide-DS50002805C.pdf>
- ▶ Microchip, “ATmega4808/4809 Data Sheet”:  
<https://ww1.microchip.com/downloads/en/DeviceDoc/ATmega4808-09-DataSheet-DS40002173B.pdf>
- ▶ Microchip, “Advanced Software Framework (ASF)”:  
<https://www.microchip.com/mplab/avr-support/advanced-software-framework>
- ▶ Microchip, “MPLAB® XC Compilers”: <https://www.microchip.com/en-us/development-tools-tools-and-software/mplab-xc-compilers>