Πολυτεχνική Σχολή
Τμήμα Μηχανικών Η/Υ & Πληροφορικής

Προηγμένοι Μικροεπεξεργαστές

Εργαστηριακή Άσκηση 4

Κατσαρός Ανδρέας 1084522
Ποταμιάνος Άγγελος Νικόλαος 1084537

Πάτρα, 2023-24

Ερωτήματα Εργαστηριακής Άσκησης 4

1 Ενεργοποιείστε την αρχική λειτουργία του συστήματος, δηλαδή τη λειτουργία του ADC με τα δύο κατώφλια υγρασίας. Ανάλογα με την συνθήκη σύγκρισης που ισχύει κάθε φορά, ενεργοποιείται το αντίστοιχο LED (LED0 ή LED1).

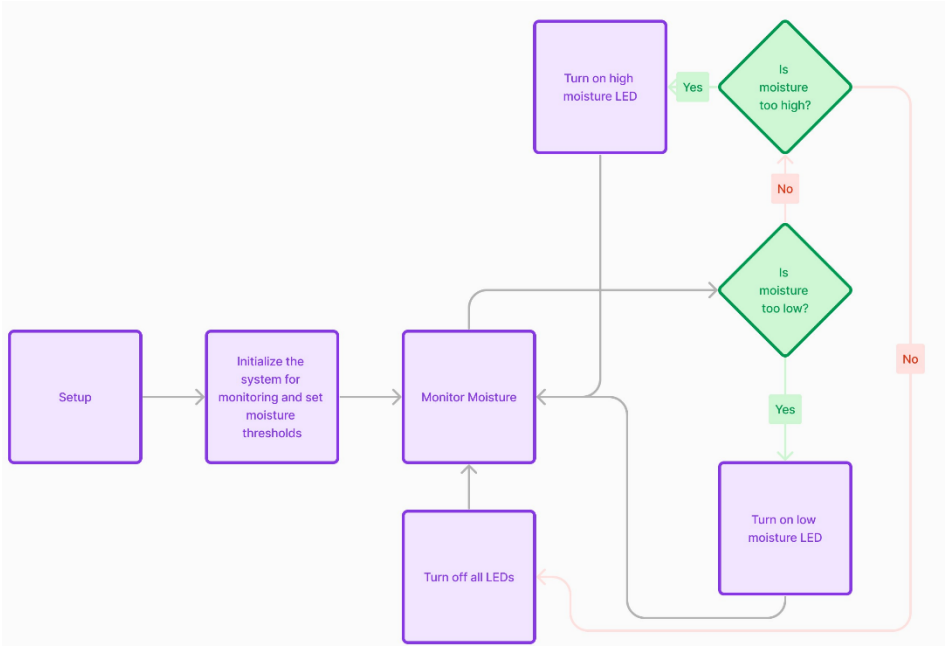
2 Προσθέστε τις δυο λειτουργίες του συστήματος, δηλαδή τη λειτουργία ποτίσματος (ενεργοποίηση χρονιστή – timer/counter) και αερισμού (ενεργοποίηση ενός PWM), που εκτελούνται ανάλογα με τον διακόπτη (SW5 ή SW6) που επιλέχθηκε από τον χρήστη.

3 Υλοποιείστε τη λειτουργία που ειδοποιεί τον χρήστη για λάθος εισαγωγή κουμπιού (δηλαδή ενεργοποίηση όλων των LEDs)

Μπορείτε να έχετε πρόσβαση και σε ολόκληρο τον κώδικα στο τέλος της αναφοράς . Παρακάτω επεξηγούμε αναλυτικά τον κώδικα για την υλοποίηση των ερωτημάτων .

Ερώτημα 1:

Παράθεση Διαγράμματος Ροής :



Επεξήγηση βασικών σημείων του κώδικα:

1. Δήλωση και ορισμός των threshold υγρασίας: Χρησιμοποιούνται δύο threshold (**low_moisture_limit** και **high_moisture_limit**) για να καθορίσουν τις συνθήκες υπό τις οποίες θα ενεργοποιούνται τα LEDs.

```
#define low_moisture_limit 15
#define high_moisture_limit 30
```

2. **Ρύθμιση ADC**: Ο ADC ρυθμίζεται για συνεχή λειτουργία με 10-bit ανάλυση .Επίσης, ορίζονται τα threshold υγρασίας στον ADC.

```
ADC0.CTRLA |= ADC_RESSEL_10BIT_gc; // 10-bit resolution
ADC0.CTRLA |= ADC_FREERUN_bm; // Free-Running mode enabled
ADC0.CTRLA |= ADC_ENABLE_bm; // Ενεργοποίηση ADC
```

3. **Η ISR(ADC0_WCOMP_vect)** : διακοπή που εκτελείται όταν ο ADC ανιχνεύσει μια τιμή που βρίσκεται εκτός των οριοθετημένων threshold (Window Compare Interrupt):

   Έλεγχος Υγρασίας:

   - Χαμηλή Υγρασία: Αν η τιμή του ADC είναι μικρότερη από το ελάχιστο κατώφλι (low_moisture_limit), η μεταβλητή moisture_level_flag ορίζεται σε 1. Αυτό σηματοδοτεί ότι η υγρασία είναι πολύ χαμηλή και τα φυτά χρειάζονται πότισμα.
   - Υψηλή Υγρασία: Αν η τιμή του ADC είναι μεγαλύτερη από το μέγιστο κατώφλι (high_moisture_limit), η moisture_level_flag ορίζεται σε 2. Αυτό υποδεικνύει υπερβολική υγρασία και την ανάγκη για μείωσή της.

```
int moisture_level_flag;
```

```
ISR(ADC0_WCOMP_vect) {
    cli();
    // Ελέγχος αν το αποτέλεσμα είναι μικρότερο από το ελάχιστο όριο
    if (ADC0.RES < low_moisture_limit) {
        moisture_level_flag = 1; // Ορισμός flag για χαμηλή υγρασία
    }
    // Ελέγχος αν το αποτέλεσμα είναι μεγαλύτερο από το μέγιστο όριο
    else if (ADC0.RES > high_moisture_limit) {
        moisture_level_flag = 2; // Ορισμός flag για υψηλή υγρασία
    }
    sei();
}
```

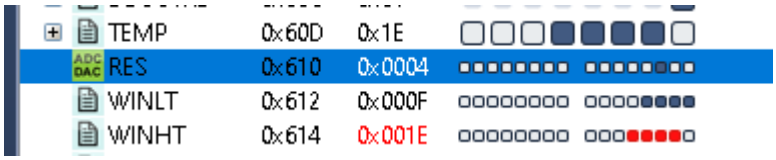
Outputs:

Μέσα στον βρόχο, ελέγχεται η τιμή της **moisture_level_flag** με μια δομή **switch** και εχουμε δυο περιπτώσεις:

Περίπτωση 1:Αν η τιμή είναι **moisture_level_flag** =1, τότε ενεργοποιείτε το LED στο PIN0.
Περίπτωση 2: Αν είναι **moisture_level_flag** = 2, τότε το LED στο PIN1.

1η περίπτωση: Χαμηλή υγρασία

Επιλέγουμε result τέτοιο ώστε ADC0.RES < low_moisture_limit:

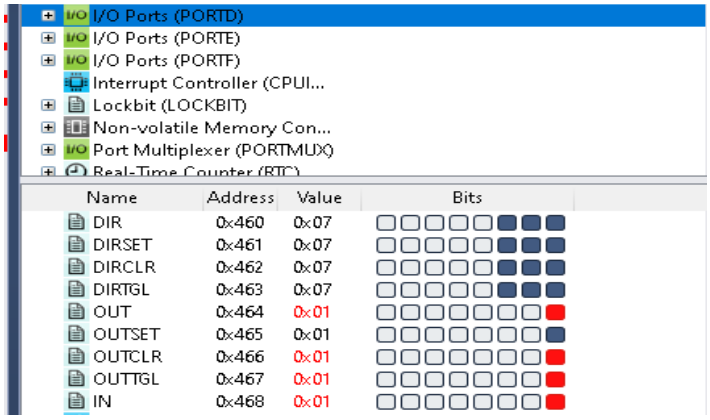


TEMP	0x60D	0x1E	
RES	0x610	0x0004	
WINLT	0x612	0x000F	
WINHT	0x614	0x001E	

Επομένως, μετά την εκτέλεση της ISR: moisture_level_flag = 1;(έχουμε χαμηλή υγρασία).
Στον βρόχο λοιπόν έχουμε : case 1:

```
    // Ενέργειες με βάση την τιμή της moisture_level_flag
    switch(moisture_level_flag) {
        case 1:
        PORTD.OUTSET = PIN0_bm; // Ενεργοποίηση LED0 για χαμηλή υγρασία
        break;
        case 2:
        PORTD.OUTSET = PIN1_bm; // Ενεργοποίηση LED1 για υψηλή υγρασία
        break;
        default:
        // Επαναφορά των LED αν δεν υπάρχει ιδιαίτερη συνθήκη
        PORTD.OUTCLR = PIN0_bm | PIN1_bm;
```

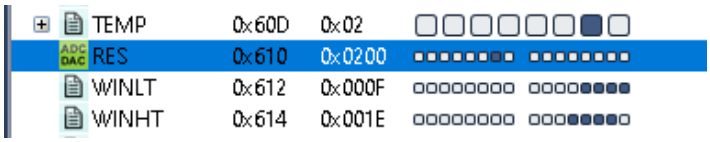
Άρα πρέπει να ανάψει το PIN0 του PortD το οποίο επιτυγχάνεται:



Name	Address	Value	Bits
DIR	0x460	0x07	
DIRSET	0x461	0x07	
DIRCLR	0x462	0x07	
DIRTGL	0x463	0x07	
OUT	0x464	0x01	
OUTSET	0x465	0x01	
OUTCLR	0x466	0x01	
OUTTGL	0x467	0x01	
IN	0x468	0x01	

2η περίπτωση: Υψηλή υγρασία

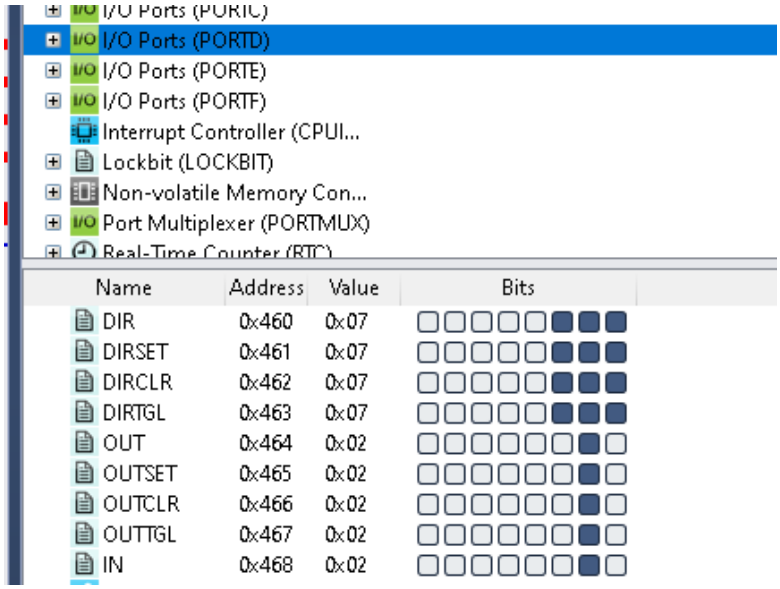
Επιλέγουμε result τέτοιο ώστε ADC0.RES >high_moisture_limit:



Επομένως, μετά την εκτέλεση της ISR: moisture_level_flag = 2;(έχουμε υψηλή υγρασία). Στον βρόχο λοιπόν έχουμε : case 2:

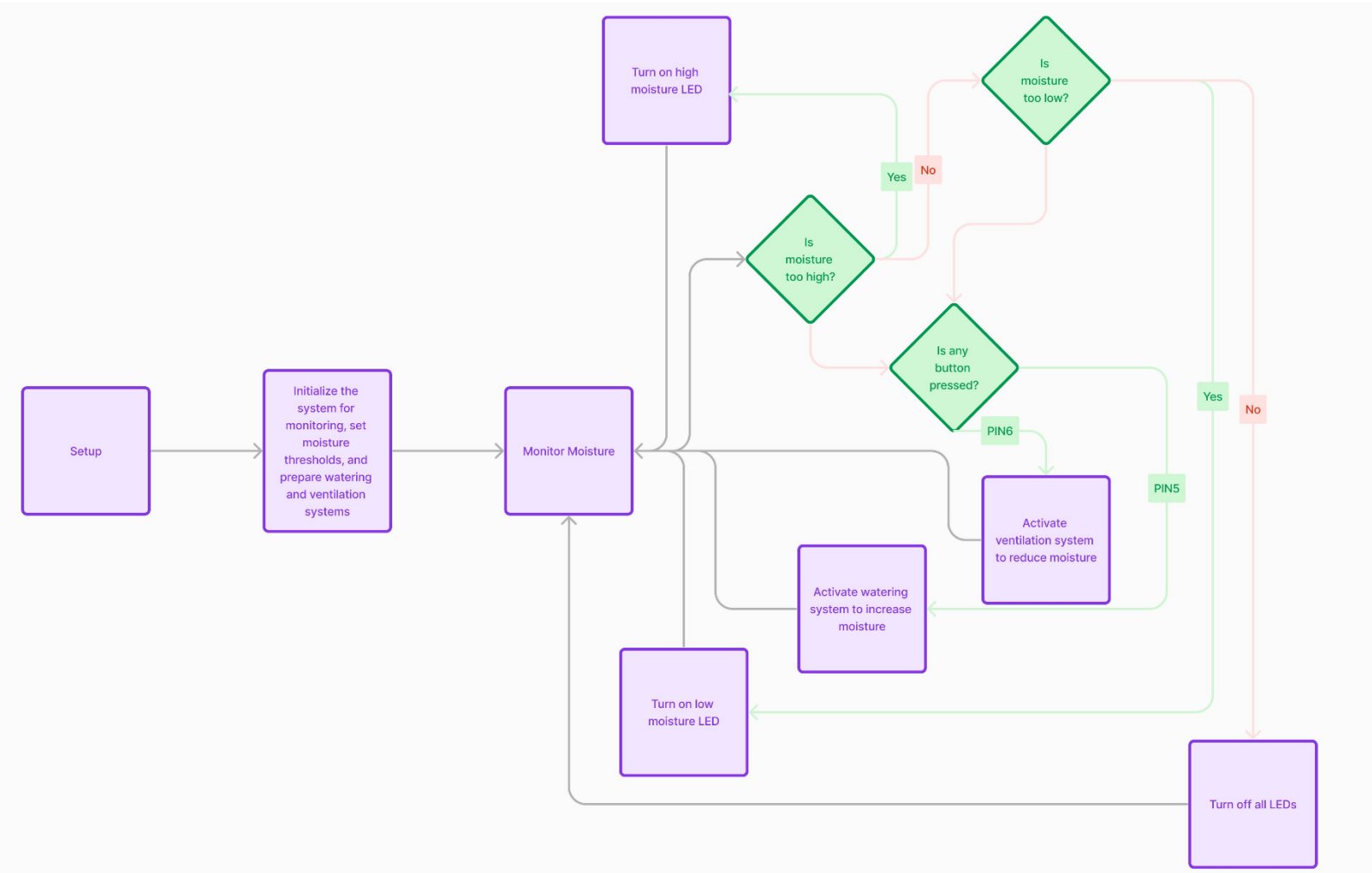
```
ADC0.COMMAND |= ADC_STCONV_bm;

while(1) {
    // Ενέργειες με βάση την τιμή της moisture_level_flag
    switch(moisture_level_flag) {
        case 1:
        PORTD.OUTSET = PIN0_bm; // Ενεργοποίηση LED0 για χαμηλή υγρασία
        break;
        case 2:
        PORTD.OUTSET = PIN1_bm; // Ενεργοποίηση LED1 για υψηλή υγρασία
        break;
        default:
        // Επαναφορά των LED αν δεν υπάρχει ιδιαίτερη συνθήκη
        PORTD.OUTCLR = PIN0_bm | PIN1_bm;
    }
```

Άρα πρέπει να ανάψει το PIN1 του PortD το οποίο επιτυγχάνεται:



Ερώτημα 2:

Παράθεση Διαγράμματος Ροής :



Επιλογή Περιόδου

Η συχνότητα του PWM σήματος (f_{PWM_SS}) εξαρτάται από τη συχνότητα του ρολογιού περιφερειακού f_{CLK_PER} (), τον prescaler (N), και την τιμή του PER (που ορίζει την περίοδο του τάιμερ).

Στην περίπτωση που f_{CLK_PER} είναι 20MHz και ο prescaler N=1024, ο τύπος μας δίνει έναν τρόπο να υπολογίσουμε την τελική συχνότητα του PWM σήματος βάσει της επιθυμητής

περιόδου του τάιμερ (PER). Ουσιαστικά, αυτό μας επιτρέπει να προσαρμόσουμε την ταχύτητα κίνησης του ανεμιστήρα ανάλογα με τις ανάγκες μας.

$$f_{PWM_SS} = \frac{f_{CLK_PER}}{N(PER + 1)}$$

Επεξήγηση βασικών σημείων επεκτάσιμου κώδικα:

1. ISR για Διακόπτες στο PORTF (ISR(PORTF_PORT_vect)):

Αυτή η διακοπή ανταποκρίνεται σε ενεργοποιήσεις των διακοπτών SW5 και SW6 στο PORTF:

- **Έλεγχος Ενεργοποιήσεων Διακοπτών**: Ελέγχει αν τα αντίστοιχα bits των διακοπτών είναι υψηλά (πιεσμένα):

  if (PORTF_IN & PIN5_bm): Ελέγχει αν ο διακόπτης SW5 είναι πιεσμένος και ορίζει την **watering_system_flag** σε 1 για να ενεργοποιήσει το σύστημα ποτίσματος.

  else if (PORTF_IN & PIN6_bm): Ελέγχει αν ο διακόπτης SW6 είναι πιεσμένος και ορίζει την **ventilation_system_flag** σε 1 για να ενεργοποιήσει το σύστημα αερισμού.

```
ISR(PORTF_PORT_vect) {
    cli();
    int portf_intflags = PORTF.INTFLAGS; // Procedure to clear the interrupt flag
    PORTF.INTFLAGS = portf_intflags;

    if (PORTF_IN && PIN5_bm) {
        watering_system_flag = 1; // Set flag to activate watering system
        } else if (PORTF_IN && PIN6_bm) {
        ventilation_system_flag = 1; // Set flag to activate ventilation system
    }
    sei();
```

2. Ενεργοποίηση Χρονιστή (`ISR(TCA0_CMP0_vect)`):

- Τερματισμός Ποτίσματος: Απενεργοποιεί τον χρονιστή μετά την πάροδο του υπολογισμένου χρόνου, επαναφέρει το σύστημα ποτίσματος και σβήνει το αντίστοιχο LED.

```
ISR(TCA0_CMP0_vect) {
    cli();
    TCA0.SINGLE.CTRLA = 0; // Disable timer
    int intflags = TCA0.SINGLE.INTFLAGS; // Procedure to clear interrupt flag
    TCA0.SINGLE.INTFLAGS = intflags;
    watering_system_flag = 0; // Reset watering system flag
    PORTD.OUTCLR = PIN0_bm; // Turn off LED0
```

3. Ενεργοποίηση PWM (`ISR(TCA0_OVF_vect)`):

- Τερματισμός Αερισμού: Παρακολουθεί τον αριθμό των επαναλήψεων του PWM και απενεργοποιεί τον χρονιστή και τα LEDs μετά από τέσσερις κύκλους.

```c
ISR(TCA0_OVF_vect) {
    static uint8_t pwm_count = 0;
    cli();
    pwm_count++;
    if (pwm_count == 4) {
        TCA0.SINGLE.CTRLA = 0; // Disable timer
        PORTD.OUTCLR = PIN1_bm | PIN2_bm; // Turn off LED1 and LED2
        ventilation_system_flag = 0; // Reset ventilation system flag
        pwm_count = 0; // Reset counter
    }
    sei();
```

4) Λειτουργίες `activate_watering_system()` και `activate_ventilation_system()`:

```c
void activate_watering_system(void) {
    // Initialize timer
    TCA0.SINGLE.CNT = 0; // Clear counter
    TCA0.SINGLE.CTRLB = TCA_SINGLE_WGMODE_NORMAL_gc; // Normal Mode
    TCA0.SINGLE.CMP0 = calculated_time; // Set compare value
    TCA0.SINGLE.CTRLA = TCA_SINGLE_CLKSEL_DIV1024_gc | TCA_SINGLE_ENABLE_bm; // Set clock source and enable timer
    TCA0.SINGLE.INTCTRL = TCA_SINGLE_CMP0_bm; // Enable compare interrupt
    PORTD.OUTSET = PIN0_bm; // Turn on LED0
}
```

```c
void activate_ventilation_system(void) {
    // Initialize PWM
    TCA0.SINGLE.CNT = 0; // Clear counter
    TCA0.SINGLE.CTRLB = TCA_SINGLE_WGMODE_DSBOTTOM_gc; // Dual Slope Bottom PWM Mode
    TCA0.SINGLE.PER = 0xFF; // Set period
    TCA0.SINGLE.CMP0 = 0x7F; // Set compare value for 50% duty cycle
    TCA0.SINGLE.CTRLA = TCA_SINGLE_CLKSEL_DIV64_gc | TCA_SINGLE_ENABLE_bm; // Set clock source and enable timer
    TCA0.SINGLE.INTCTRL = TCA_SINGLE_OVF_bm; // Enable overflow interrupt
    PORTD.OUTSET = PIN2_bm; // Turn on LED2
}
```

- Πότισμα: Ρυθμίζει τον χρονιστή με την υπολογισμένη τιμή και ενεργοποιεί το αντίστοιχο LED.

- Αερισμός: Ρυθμίζει τον PWM για τη δημιουργία παλμού με 50% κύκλο εργασίας και ενεργοποιεί το αντίστοιχο LED.

Outputs:

Output διαδικασία για τη λειτουργία ποτίσματος:

A) Ενεργοποίηση SW5 για πότισμα & watering_system_flag=1:

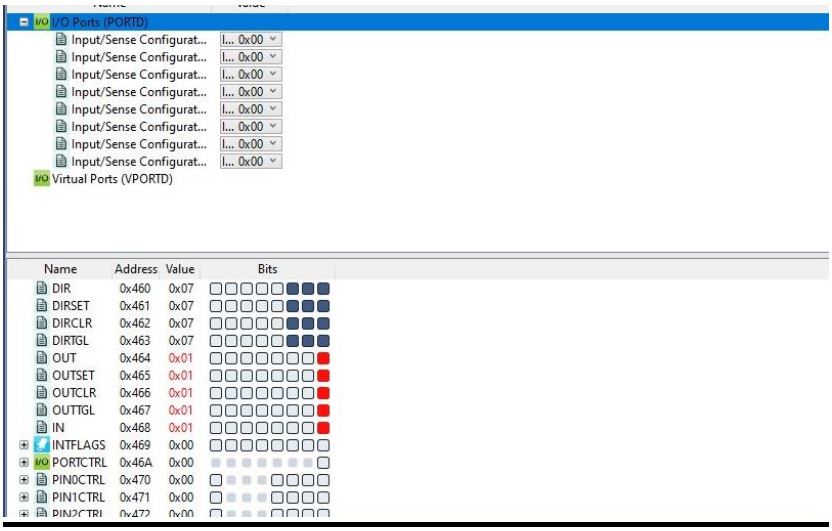
```c
ISR(PORTF_PORT_vect) {
    cli();
    int portf_intflags = PORTF.INTFLAGS; // Procedure to clear the interrupt flag
    PORTF.INTFLAGS = portf_intflags;

    if (PIN5_bm) {
        watering_system_flag = 1; // Set flag to activate watering system
        } else if ( PIN6_bm) {
        ventilation_system_flag = 1; // Set flag to activate ventilation system
    }
    sei();
```

B) Activate_watering_system() function called:

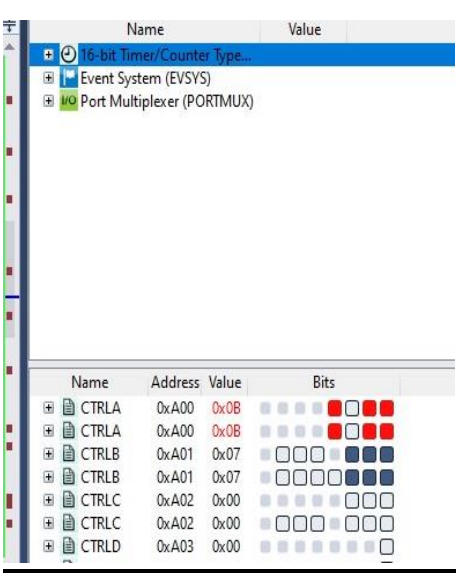
```
// Check if the watering system needs to be activated
if (watering_system_flag) {
    activate_watering_system();
    watering_system_flag = 0; // Reset the flag
}
```

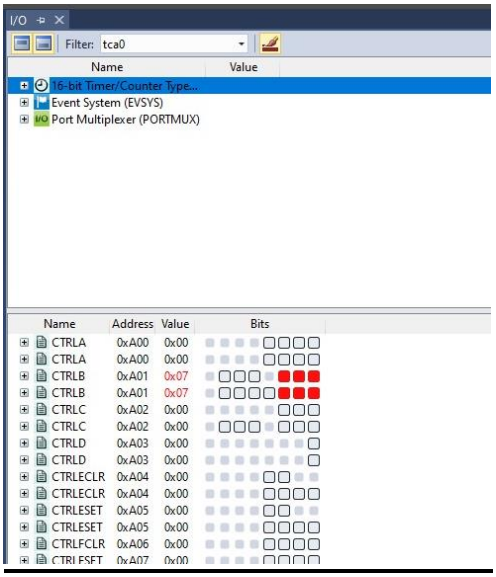
Γ)Συνάρτηση και πότισμα:

```
void activate_watering_system(void) {
    // Initialize timer
    TCA0.SINGLE.CNT = 0; // Clear counter
    TCA0.SINGLE.CTRLB = TCA_SINGLE_WGMODE_NORMAL_gc; // Normal Mode
    TCA0.SINGLE.CMP0 = calculated_time; // Set compare value
    TCA0.SINGLE.CTRLA = TCA_SINGLE_CLKSEL_DIV1024_gc | TCA_SINGLE_ENABLE_bm; // Set clock source and enable timer
    TCA0.SINGLE.INTCTRL = TCA_SINGLE_CMP0_bm; // Enable compare interrupt
    PORTD.OUTSET = PIN0_bm; // Turn on LED0
}
```



Timers :





Output διαδικασία για τη λειτουργία αερισμού:

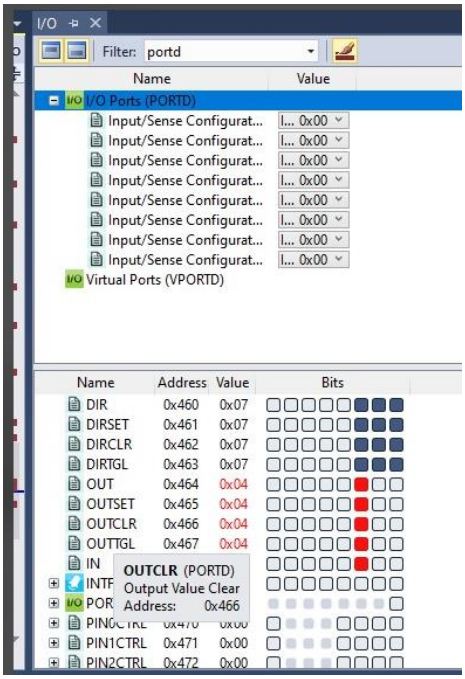
A) Ενεργοποίηση SW6 για πότισμα & ventilation_system_flag=1:

```
ISR(PORTF_PORT_vect) {
    cli();
    int portf_intflags = PORTF.INTFLAGS; // Procedure to clear the interrupt flag
    PORTF.INTFLAGS = portf_intflags;

    if (PIN6_bm) {
        ventilation_system_flag = 1; // Set flag to activate ventilation system
```

B) Activate_ventilation_system() function called:

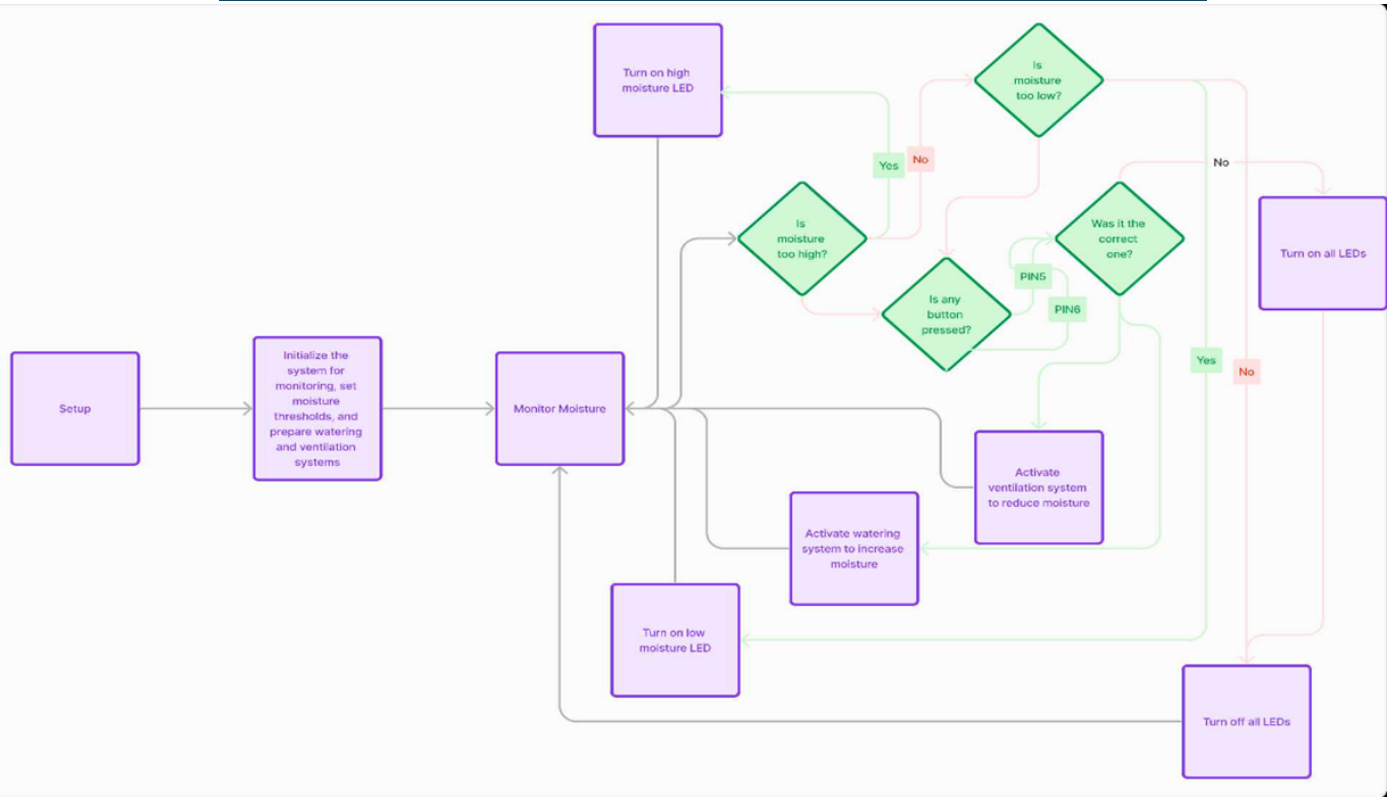
```
    while (1) {
        // Actions based on the moisture_level_flag value
        switch (moisture_level_flag) {
            case 1:
                PORTD.OUTSET = PIN0_bm; // Turn on LED0 for low moisture
                break;
            case 2:
                PORTD.OUTSET = PIN1_bm; // Turn on LED1 for high moisture
                break;
            default:
                // Clear LEDs if no specific condition
                PORTD.OUTCLR = PIN0_bm | PIN1_bm;
                break;
        }

        // Check if the watering system needs to be activated
        if (ventilation_system_flag) {
            activate_ventilation_system();
            watering_system_flag = 0; // Reset the flag
        }
```

Γ) Output αεριαμός:



Ερώτημα 3:

Παράθεση Διαγράμματος Ροής :



Συναρτήσεις activate_all_leds και deactivate_all_leds :

```
void activate_all_leds(void) {
    PORTD.OUTSET = PIN0_bm | PIN1_bm | PIN2_bm; // Turn on all LEDs
}

// Function to deactivate all LEDs
void deactivate_all_leds(void) {
    PORTD.OUTCLR = PIN0_bm | PIN1_bm | PIN2_bm; // Turn off all LEDs
}
```

Οι δύο συναρτήσεις, activate_all_leds και deactivate_all_leds, χρησιμοποιούνται για την ενεργοποίηση και απενεργοποίηση όλων των LEDs σε έναν συγκεκριμένο θύρα (PORTD) του μικροελεγκτή:

Η activate_all_leds: Αυτή η συνάρτηση ουσιαστικά θέτει τα συγκεκριμένα bits του PORTD (που αντιστοιχούν στα pins 0, 1 και 2) σε υψηλό επίπεδο, δηλαδή στην τιμή 1, προκειμένου να ενεργοποιήσει όλα τα LEDs που είναι συνδεδεμένα σε αυτά τα pins.

Η deactivate_all_leds: Αντίστοιχα, αυτή η συνάρτηση θέτει τα συγκεκριμένα bits του PORTD σε χαμηλό επίπεδο, δηλαδή στην τιμή 0, προκειμένου να απενεργοποιήσει

όλα τα LEDs που είναι συνδεδεμένα σε αυτά τα pins.

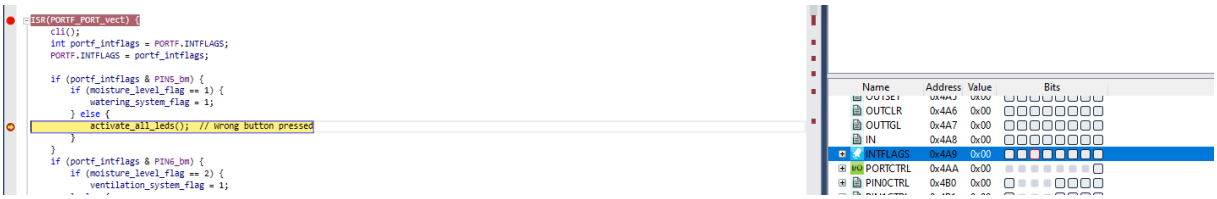
Έλεγχος πατημένου κουμπιου:

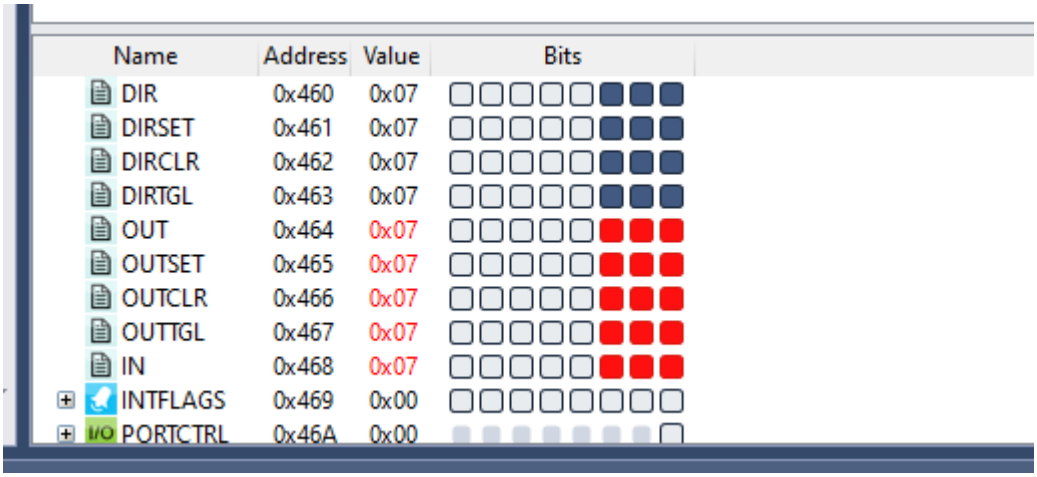
```
ISR(PORTF_PORT_vect) {
    cli();
    int portf_intflags = PORTF.INTFLAGS;
    PORTF.INTFLAGS = portf_intflags;

    if (portf_intflags & PIN5_bm) {
        if (moisture_level_flag == 1) {
            watering_system_flag = 1;
        } else {
            activate_all_leds();  // Wrong button pressed
        }
    }
    if (portf_intflags & PIN6_bm) {
        if (moisture_level_flag == 2) {
            ventilation_system_flag = 1;
        } else {
            activate_all_leds();  // Wrong button pressed
        }
    }
    sei();
}
```

Σε αυτό το κομμάτι, ελέγχεται αν τα κουμπιά SW5 ή SW6 πατήθηκαν. Αν ναι, ελέγχεται η κατάσταση του εδάφους (moisture_level_flag) και εάν δεν είναι σύμφωνη με τη λειτουργία του κουμπιού (π.χ δηλαδή όταν έχουμε υψηλό moisture_level_flag προφανώς θέλουμε ventilation και όχι watering), ενεργοποιούνται όλα τα LEDs με την κλήση της συνάρτησης activate_all_leds(), δηλώνοντας έτσι ένα λάθος εισαγωγής στον χρήστη. Αυτή η προσθήκη εκτελεί τη λειτουργία που ζητείται στο ερώτημα 3.

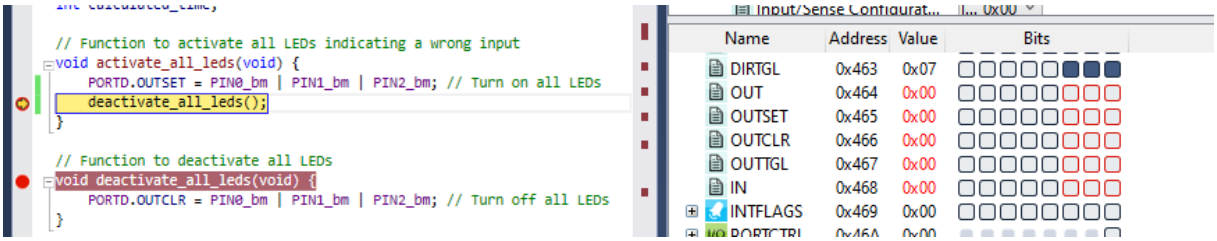
Outputs:

1)Outputs for activation leds:





2)Outputs for deactivation leds:



Επιπλέον σχόλια:

Βάσει των παρατηρήσεων κατά τη δια ζώσης συνάντηση για την τέταρτη εργαστηριακή άσκηση, αναπροσαρμόσαμε και επιδιορθώσαμε τα εξής:

α) Κατάφεραμε σύμφωνα με τις υποδείξεις στον κώδικα να προσθέσαμε το κομμάτι που εκτελεί τη λειτουργία "clear interrupt" στον ADC. Αυτό το κομμάτι κώδικα απενεργοποιεί τις διακοπές προκειμένου να αποφευχθεί η επανάληψη της Interrupt Service Routine (ISR) κατά την εκτέλεσή της. Έτσι, αποφεύγεται η δημιουργία επαναλαμβανόμενων διακοπών που θα μπορούσαν να οδηγήσουν σε μη απρόβλεπτη συμπεριφορά του συστήματος. Παρακάτω παρουσιάζεται το αντίστοιχο κομμάτι κώδικα:

β) Επίσης, προσθέσαμε κώδικα για την απενεργοποίηση (deactivate) των συστημάτων ποτίσματος (watering system) και εξαερισμού (ventilation system) μετά τη λήξη του χρονιστή (timer). Προσπαθήσαμε μεσω του κώδικά μας να εξασφαλίσουμε ότι τα συστήματα απενεργοποιούνται μετά τον προκαθορισμένο χρόνο λειτουργίας τους.

ΠΛΉΡΗΣ ΚΏΔΙΚΑΣ:

ASK1.C:

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

#define low_moisture_limit 15
#define high_moisture_limit 30

int moisture_level_flag;

ISR(ADC0_WCOMP_vect) {
	cli();
	// Ελέγχος αν το αποτέλεσμα είναι μικρότερο από το ελάχιστο όριο
	if (ADC0.RES < low_moisture_limit) {
		moisture_level_flag = 1; // Ορισμός flag για χαμηλή υγρασία
	}
	// Ελέγχος αν το αποτέλεσμα είναι μεγαλύτερο από το μέγιστο όριο
	else if (ADC0.RES > high_moisture_limit) {
		moisture_level_flag = 2; // Ορισμός flag για υψηλή υγρασία
	}
	sei();
}

int main(void) {
	// Αρχικοποίηση LED
	PORTD.DIRSET = PIN2_bm | PIN1_bm | PIN0_bm; // Ρύθμιση εξόδων (2:
Σύστημα Αερισμού, 1: Υψηλή Υγρασία, 0: Χαμηλή Υγρασία)
	PORTF.DIRCLR = PIN5_bm | PIN6_bm;

	ADC0.CTRLA |= ADC_RESSEL_10BIT_gc; // 10-bit resolution
	ADC0.CTRLA |= ADC_FREERUN_bm; // Free-Running mode enabled
	ADC0.CTRLA |= ADC_ENABLE_bm; // Ενεργοποίηση ADC
	ADC0.MUXPOS |= ADC_MUXPOS_AIN7_gc; // Επιλογή Αισθητήρα
	ADC0.DBGCTRL |= ADC_DBGRUN_bm; // Enable Debug Mode

	ADC0.WINLT |= low_moisture_limit; // Ορισμός κατώτερου ορίου
	ADC0.WINHT |= high_moisture_limit; // Ορισμός ανώτερου ορίου
	ADC0.INTCTRL |= ADC_WCMP_bm;
	ADC0.CTRLE |= ADC_WINCM1_bm; // Διακοπή του RES όταν είναι εκτός

	sei();
	ADC0.COMMAND |= ADC_STCONV_bm;

	while(1) {
		// Ενέργειες με βάση την τιμή της moisture_level_flag
		switch(moisture_level_flag) {
			case 1:
			PORTD.OUTSET = PIN0_bm; // Ενεργοποίηση LED0 για χαμηλή
υγρασία
```

```
                    break;
                    case 2:
                    PORTD.OUTSET = PIN1_bm; // Ενεργοποίηση LED1 για υψηλή
υγρασία
                    break;
                    default:
                    // Επαναφορά των LED αν δεν υπάρχει ιδιαίτερη συνθήκη
                    PORTD.OUTCLR = PIN0_bm | PIN1_bm;
            }

            // Επαναφορά της σημαίας moisture_level_flag
            moisture_level_flag = 0;
      }
}
```

ASK2.C

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

#define low_moisture_limit 15
#define high_moisture_limit 30

int moisture_level_flag;
int watering_system_flag = 0;
int ventilation_system_flag = 0;
int calculated_time;

ISR(ADC0_WCOMP_vect) {
    cli();
    // Check if the result is less than the minimum limit
    if (ADC0.RES < low_moisture_limit) {
        moisture_level_flag = 1; // Set flag for low moisture
        calculated_time = low_moisture_limit - ADC0.RES; // Calculate time
for watering
    }
    // Check if the result is greater than the maximum limit
    else if (ADC0.RES > high_moisture_limit) {
        moisture_level_flag = 2; // Set flag for high moisture
    }
    sei();
}

ISR(PORTF_PORT_vect) {
    cli();
    int portf_intflags = PORTF.INTFLAGS; // Procedure to clear the
interrupt flag
    PORTF.INTFLAGS = portf_intflags;

    if (portf_intflags & PIN5_bm) {
        watering_system_flag = 1; // Set flag to activate watering system
    } else if (portf_intflags & PIN6_bm) {
        ventilation_system_flag = 1; // Set flag to activate ventilation
system
    }
    sei();
}
```

```
ISR(TCA0_CMP0_vect) {
    cli();
    TCA0.SINGLE.CTRLA = 0; // Disable timer
    int intflags = TCA0.SINGLE.INTFLAGS; // Procedure to clear interrupt
flag
    TCA0.SINGLE.INTFLAGS = intflags;
    watering_system_flag = 0; // Reset watering system flag
    PORTD.OUTCLR = PIN0_bm; // Turn off LED0
    sei();
}

void activate_watering_system(void) {
    // Initialize timer
    TCA0.SINGLE.CNT = 0; // Clear counter
    TCA0.SINGLE.CTRLB = TCA_SINGLE_WGMODE_NORMAL_gc; // Normal Mode
    TCA0.SINGLE.CMP0 = calculated_time; // Set compare value
    TCA0.SINGLE.CTRLA = TCA_SINGLE_CLKSEL_DIV1024_gc |
TCA_SINGLE_ENABLE_bm; // Set clock source and enable timer
    TCA0.SINGLE.INTCTRL = TCA_SINGLE_CMP0_bm; // Enable compare interrupt
    PORTD.OUTSET = PIN0_bm; // Turn on LED0
}

void activate_ventilation_system(void) {
      // Initialize PWM
      TCA0.SINGLE.CNT = 0; // Clear counter
      TCA0.SINGLE.CTRLB = TCA_SINGLE_WGMODE_DSBOTTOM_gc; // Dual Slope
Bottom PWM Mode
      TCA0.SINGLE.PER = 0xFF; // Set period
      TCA0.SINGLE.CMP0 = 0x7F; // Set compare value for 50% duty cycle
      TCA0.SINGLE.CTRLA = TCA_SINGLE_CLKSEL_DIV64_gc |
TCA_SINGLE_ENABLE_bm; // Set clock source and enable timer
      TCA0.SINGLE.INTCTRL = TCA_SINGLE_OVF_bm; // Enable overflow interrupt
      PORTD.OUTSET = PIN2_bm; // Turn on LED2
}


ISR(TCA0_OVF_vect) {
    static uint8_t pwm_count = 0;
    cli();
    pwm_count++;
    if (pwm_count == 4) {
        TCA0.SINGLE.CTRLA = 0; // Disable timer
        PORTD.OUTCLR = PIN1_bm | PIN2_bm; // Turn off LED1 and LED2
        ventilation_system_flag = 0; // Reset ventilation system flag
        pwm_count = 0; // Reset counter
    }
    sei();
}

int main(void) {
    // Initialize LEDs
    PORTD.DIRSET = PIN2_bm | PIN1_bm | PIN0_bm; // Set as outputs (2:
Ventilation System, 1: High Moisture, 0: Low Moisture)
    PORTF.DIRCLR = PIN5_bm | PIN6_bm; // Set as inputs
    PORTF.PIN5CTRL |= PORT_PULLUPEN_bm | PORT_ISC_BOTHEDGES_gc;
    PORTF.PIN6CTRL |= PORT_PULLUPEN_bm | PORT_ISC_BOTHEDGES_gc;

    // 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; // Select sensor
    ADC0.DBGCTRL |= ADC_DBGRUN_bm; // Enable Debug Mode
    ADC0.WINLT |= low_moisture_limit; // Set lower limit
    ADC0.WINHT |= high_moisture_limit; // Set upper limit
    ADC0.INTCTRL |= ADC_WCMP_bm;
    ADC0.CTRLE |= ADC_WINCM1_bm; // Interrupt when RES is outside the
window

    sei(); // Enable global interrupts
    ADC0.COMMAND |= ADC_STCONV_bm; // Start conversion

    while (1) {
        // Actions based on the moisture_level_flag value
        switch (moisture_level_flag) {
            case 1:
                PORTD.OUTSET = PIN0_bm; // Turn on LED0 for low moisture
                break;
            case 2:
                PORTD.OUTSET = PIN1_bm; // Turn on LED1 for high moisture
                break;
            default:
                // Clear LEDs if no specific condition
                PORTD.OUTCLR = PIN0_bm | PIN1_bm;
                break;
        }

        // Check if the watering system needs to be activated
        if (watering_system_flag) {
            activate_watering_system();
            watering_system_flag = 0; // Reset the flag
        }

        // Check if the ventilation system needs to be activated
        if (ventilation_system_flag) {
            activate_ventilation_system();
            ventilation_system_flag = 0; // Reset the flag
        }

        // Reset the moisture_level_flag
        moisture_level_flag = 0;
    }
}
```

Ask3.c (FINAL)

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

#define low_moisture_limit 15
#define high_moisture_limit 30

int moisture_level_flag;
int watering_system_flag = 0;
```

```c
int ventilation_system_flag = 0;
int calculated_time;

// Function to activate all LEDs indicating a wrong input
void activate_all_leds(void) {
    PORTD.OUTSET = PIN0_bm | PIN1_bm | PIN2_bm; // Turn on all LEDs
    deactivate_all_leds();
}

// Function to deactivate all LEDs
void deactivate_all_leds(void) {
    PORTD.OUTCLR = PIN0_bm | PIN1_bm | PIN2_bm; // Turn off all LEDs
}

ISR(ADC0_WCOMP_vect) {
    cli();  // Disable global interrupts
      int adc_intflags = ADC0.INTFLAGS;
    ADC0.INTFLAGS = adc_intflags;
    // Handling the ADC result
    if (ADC0.RES < low_moisture_limit) {
        moisture_level_flag = 1;
        calculated_time = low_moisture_limit - ADC0.RES;
    } else if (ADC0.RES > high_moisture_limit) {
        moisture_level_flag = 2;
    }
    sei();  // Enable global interrupts
}


ISR(PORTF_PORT_vect) {
    cli();
    int portf_intflags = PORTF.INTFLAGS;
    PORTF.INTFLAGS = portf_intflags;

    if (portf_intflags & PIN5_bm) {
        if (moisture_level_flag == 1) {
            watering_system_flag = 1;
        } else {
            activate_all_leds();  // Wrong button pressed
        }
    }
    if (portf_intflags & PIN6_bm) {
        if (moisture_level_flag == 2) {
            ventilation_system_flag = 1;
        } else {
            activate_all_leds();  // Wrong button pressed
        }
    }
    sei();
}

ISR(TCA0_CMP0_vect) {
    cli();
    TCA0.SINGLE.CTRLA = 0;
    int intflags = TCA0.SINGLE.INTFLAGS;
    TCA0.SINGLE.INTFLAGS = intflags;
    watering_system_flag = 0;
    deactivate_all_leds();  // Reset LEDs to default state after watering
    sei();
```

```
}

void activate_watering_system(void) {
    TCA0.SINGLE.CNT = 0;
    TCA0.SINGLE.CTRLB = TCA_SINGLE_WGMODE_NORMAL_gc;
    TCA0.SINGLE.CMP0 = calculated_time;
    TCA0.SINGLE.CTRLA = TCA_SINGLE_CLKSEL_DIV1024_gc |
TCA_SINGLE_ENABLE_bm;
    TCA0.SINGLE.INTCTRL = TCA_SINGLE_CMP0_bm;
    PORTD.OUTSET = PIN0_bm; // Turn on LED0
}

void activate_ventilation_system(void) {
    TCA0.SINGLE.CNT = 0;
    TCA0.SINGLE.CTRLB = TCA_SINGLE_WGMODE_DSBOTTOM_gc;
    TCA0.SINGLE.PER = 0xFF;
    TCA0.SINGLE.CMP0 = 0x7F;
    TCA0.SINGLE.CTRLA = TCA_SINGLE_CLKSEL_DIV64_gc | TCA_SINGLE_ENABLE_bm;
    TCA0.SINGLE.INTCTRL = TCA_SINGLE_OVF_bm;
    PORTD.OUTSET = PIN2_bm; // Turn on LED2
}

ISR(TCA0_OVF_vect) {
    static uint8_t pwm_count = 0;
    cli();
    pwm_count++;
    if (pwm_count == 4) {
        TCA0.SINGLE.CTRLA = 0;
        deactivate_all_leds();  // Reset LEDs to default state after
ventilation
        ventilation_system_flag = 0;
        pwm_count = 0;
    }
    sei();
}

int main(void) {
    PORTD.DIRSET = PIN2_bm | PIN1_bm | PIN0_bm;
    PORTF.DIRCLR = PIN5_bm | PIN6_bm;
    PORTF.PIN5CTRL |= PORT_PULLUPEN_bm | PORT_ISC_BOTHEDGES_gc;
    PORTF.PIN6CTRL |= PORT_PULLUPEN_bm | PORT_ISC_BOTHEDGES_gc;

    ADC0.CTRLA |= ADC_RESSEL_10BIT_gc | ADC_FREERUN_bm | ADC_ENABLE_bm;
    ADC0.MUXPOS |= ADC_MUXPOS_AIN7_gc;
    ADC0.DBGCTRL |= ADC_DBGRUN_bm;
    ADC0.WINLT |= low_moisture_limit;
    ADC0.WINHT |= high_moisture_limit;
    ADC0.INTCTRL |= ADC_WCMP_bm;
    ADC0.CTRLE |= ADC_WINCM1_bm;

    sei();
    ADC0.COMMAND |= ADC_STCONV_bm;

    while (1) {
        switch (moisture_level_flag) {
            case 1:
                PORTD.OUTSET = PIN0_bm;
                break;
            case 2:
```

```
                PORTD.OUTSET = PIN1_bm;
                break;
            default:
                PORTD.OUTCLR = PIN0_bm | PIN1_bm;
                break;
        }

        if (watering_system_flag) {
            activate_watering_system();
            watering_system_flag = 0;
        }

        if (ventilation_system_flag) {
            activate_ventilation_system();
            ventilation_system_flag = 0;
        }

        moisture_level_flag = 0;
    }
}
```

ΤΕΛΟΣ ΑΝΑΦΟΡΆΣ