2023

# Αναφορά 6ης Εργαστηριακής Άσκησης

***Εργαστήριο Μικροϋπολογιστών***

ΠΑΠΑΔΟΠΟΥΛΟΣ ΣΠΥΡΙΔΩΝ (ΑΜ): 03120033
ΕΜΜΑΝΟΥΗΛ ΞΕΝΟΣ (ΑΜ):03120850

## *Ζήτημα πρώτο*

Σε αυτό το ζήτημα θα υλοποιήσουμε τις συναρτήσεις που ζητούνται από την εκφώνηση. Αυτές οι συναρτήσεις θα μας βοηθήσουν στα επόμενα ζητήματα να πάρουμε την είσοδο από το 4x4 πληκτρολόγιο.

α) Η συνάρτηση scan_row ελέγχει την γραμμή του ορίσματός για πατημένους διακόπτες. Το αποτέλεσμα αυτής της συνάρτησης είναι ένας 8bit αριθμός. Αρχικά βρίσκουμε ποια από τις 4 γραμμές πρέπει να ελέγξουμε και στέλνουμε σε αυτή 0. Στην συνέχεια διαβάζουμε την θύρα εισόδου/εξόδου IO1 του PCA9555 και κρατάμε το αποτέλεσμα των τεσσάρων MSB. Αυτό γίνεται γιατί για μία δεδομένη γραμμή τα κουμπιά που πατήθηκαν φαίνονται από τις στήλες που συμβολίζονται από τα EXT_PORT[7:4]. Ο κώδικας είναι ο παρακάτω:

```
//function to get the number of row to check if a button is pressed.
Returns an 8 bit int with the buttons that were pressed.
uint8_t scan_row(uint8_t row){
    uint8_t check=1;
    check = check << row;
    PCA9555_0_write(REG_OUTPUT_1, ~check);
    uint8_t result = PCA9555_0_read(REG_INPUT_1)>>4;
    return result;
}
```

β) Η συνάρτηση scan_keypad ελέγχει τις 4 γραμμές του πληκτρολογίου για πατημένα κουμπιά. Η συνάρτηση επιστρέφει ένα 16bit αριθμό που δείχνει ποια από τα 16 κουμπιά του πληκτρολογίου έχουν πατηθεί. Τα κουμπιά του πληκτρολογίου αντιστοιχούν στα bit από το LSB προς το MSB ξεκινώντας από αριστερά προς τα δεξιά και από πάνω προς το κάτω. Δηλαδή το LSB αντιστοιχεί στο ‘1’, το δεύτερο LSB στο ‘2’, το πέμπτο LSB στο ‘4’ κτλ. Αυτή την αντιστοίχιση την επιτυγχάνουμε κάνοντας κάνοντας 12 αριστερά shift στο αποτέλεσμα του σκαναρίσματος της γραμμής 0, κάνοντας 8 αριστέρα shift στο σκανάρισμα της γραμμής 1, κάνοντας 4 αριστερά shift στο αποτέλεσμα του σκαναρίσματος της γραμμής 2 και όλα τα παραπάνω προστίθενται με το σκανάρισμα της γραμμής 3. Το αποτέλεσμα της παραπάνω άθροισης είναι το αποτέλεσμα της συνάρτησης scan_keypad. Ο κώδικας είναι ο παρακάτω:

```
//function to check the entire 4x4 keypad. Returns an 16 bit int with
the buttons that were pressed.
//The formation of the result is r0_r1_r2_r3 where its r0 is 4 bits
IO1_7, IO1_6, IO_5, IO_4
uint16_t scan_keypad(){
    uint16_t result;
    result= (scan_row(0)<<12) + (scan_row(1)<<8) + (scan_row(2)<<4)
+scan_row(3);
    return result;
}
```

γ) Η συνάρτηση scan_keyboard_rising_edge() επιστρέφει το αποτέλεσμα των πατημένων κουμπιών ελέγχοντας όμως το ενδεχόμενο να υπάρχει σπινθηρισμός. Αυτό το επιτυγχάνει ελέγχοντας δύο φορές τα πατημένα κουμπιά του πληκτρολογίου με 15ms διαφορά μεταξύ των δύο ελέγχων και συγκρίνοντας του αποτέλεσμα των δύο ελέγχων. Σε περίπτωση που ένα κουμπί ήταν πατημένο μόνο σε έναν από τους δύο ελέγχους τότε υπήρξε σπινθηρισμός και

έτσι αυτό το κουμπί δεν πρέπει να φαίνεται πατημένο στο αποτέλεσμα της συνάρτησης. Σε περίπτωση που ένα κουμπί ήταν πατημένο και στις δύο μετρήσεις σημαίνει ότι αυτό το κουμπί πρέπει να περάσει στο αποτέλεσμα. Ο έλεγχος αυτός γίνεται στη τέταρτη γραμμή του κώδικα που φαίνεται παρακάτω. Στην συνάρτηση αυτή αποθηκεύεται επίσης κάθε φορά και το αποτέλεσμα το αποτέλεσμα των κουμπιών που είναι πατημένα. Προφανώς η πρώτη τιμή αυτού του αποτέλεσματος θα είναι 0xFF γιατί στην αρχή του προγράμματος κανένα κουμπί δεν έχει προλάβει να πατηθεί. Παρακάτω φαίνεται ο κώδικας που υλοποιεί τα παραπάνω.

```
uint16_t previous_result=0xFF;

uint16_t scan_keyboard_rising_edge(){
    uint16_t result0 = scan_keypad();
    _delay_ms(15);
    uint16_t result1 = scan_keypad();
    uint16_t same = (~result0 & ~result1);
    previous_call=~same;
    return same;
}
```

δ) Η συνάρτηση keypad_to_ascii() επιστρέφει των ascii κωδικό του κουμπιού που πατήθηκε. Σε αυτή την συνάρτηση παίρνουμε ως παραδοχή από την εκφώνηση ότι κάθε φορά ένα κουμπί είναι πατημένο. Σε αυτή την συνάρτηση μέσω ενός switch που παίρνει ως όρισμα το αποτέλεσμα της scan_keyboard_rising_edge() επιστρέφουμε των ascii κωδικό του κουμπιού που είναι πατημένο σύμφωνα με την απεικόνιση που περιγράφηκε στην συνάρτηση scan_keypad() παραπάνω. Σε περίπτωση που δεν έχει πατηθεί κανένα κουμπί επιστρέφουμε 0 που είναι ο κωδικός του null character. Το return 0 στο τέλος της συνάρτησης είναι προκειμένου να μην εμφανίζει ο compiler το warning ότι σε μια non void function δεν επιστρέφουμε τίποτα, αλλά στην πραγματικότητα αυτό δεν θα γίνει ποτέ. Ο κώδικας της keypad_to_ascii() είναι ο ακόλουθος:

```
uint8_t keypad_to_ascii(){

        switch(scan_keyboard_rising_edge()) {
        case 0: return 0;
        case 1: return '1';
        case 2: return '2';
        case 4: return '3';
        case 8: return 'A';
        case 16: return '4';
        case 32: return '5';
        case 64: return '6';
        case 128: return 'B';
        case 256: return '7';
        case 512: return '8';
        case 1024: return '9';
        case 2048: return 'C';
        case 4096: return '*';
        case 8192: return '0';
        case 16384: return '#';
        case 32768: return 'D';
```

```
    }
        return 0;
}
```

ε) Το πρόγραμμα που κάνει την αντιστοίχιση των LED της PORTB με τα κουμπιά που πατήθηκαν στο 4x4 πληκτρολόγιο που περιγράφεται στην εκφώνηση είναι ο παρακάτω:

```
int main(){

    twi_init();

    DDRB=0xFF;

    PCA9555_0_write(REG_CONFIGURATION_1, 0xF0); //Set EXT_PORT1[7:4] as
input and EXT_PORT[3:0] as output

    char input;

    while(1){
        input=keypad_to_ascii();
        if(input=='1') PORTB=0x01;
        if(input=='5') PORTB=0x02;
        if(input=='9') PORTB=0x04;
        if(input=='D') PORTB=0x08;\
        else PORTB=0;

    }
}
```

Πρώτα αρχικοποιούμε τον δίαυλο, θέτουμε το PORTB ως έξοδο και θέτουμε τα EXT_PORT[0:3] ως εξόδους (τα bits που αντιστοιχούν στις γραμμές του πληκτρολογίου) και τα EXT_PORT[4:7] ως εισόδους (τα bit που αντιστοιχούν στις στήλες του 4x4 πληκτρολογίου). Στην συνέχεια ελέγχοντας συνεχώς στο πληκτρολόγιο και βάζοντας στην μεταβλητή input τον ASCII χαρακτήρα του κουμπιού που πατήθηκε στο πληκτρολόγιο ελέγχουμε αν ο χαρακτήρας είναι το ‘1’, το ‘5’, το ‘9’ ή το D και ανάβουμε το κατάλληλο LED της PORTB. Σε περίπτωση που δεν πατήθηκε κανένα από τα κουμπιά που αντιστοιχούν στους παραπάνω ASCII χαρακτήρες τα LED του PORTB είναι σβηστά.

---

## *Ζήτημα δεύτερο*

---

Με βάση τις συναρτήσεις που κατασκευάσαμε στο μέρος πρώτο θα υλοποιήσουμε και το ζητούμενο αυτής της άσκησης. Στην άσκηση αυτή μας ζητείται να απεικονίζουμε στο lcd display το πλήκτρο που πατήθηκε τελευταίο. Για αυτό θα χρησιμοποιήσουμε τις συναρτήσεις για τον χειρισμό του LCD display μέσω του PORT_EXT και IO0 όπως κάναμε στην προηγούμενη εργαστηριακή άσκηση. Σε αυτό το ζητούμενο για ευκολία θα παραθέσουμε το κομμάτι κώδικα που είναι κρίσιμο για την υλοποίηση μας. Ακολούθως παρατίθεται ο κώδικας για την άσκηση αυτή( εξαιρουμένων των γνωστών από προηγούμενες ασκήσεις συναρτήσεων):

```
```

```
uint8_t my_data;
void write_lcd_display(){
    uint8_t ascii;
    PCA9555_0_write(REG_CONFIGURATION_0, 0x00); //Set EXT_PORT0 as
output
    ascii=keypad_to_ascii();
    if(my_data!=ascii && ascii!=0){
        lcd_clear_display();
        lcd_data(ascii);
        my_data=ascii;
    }

}

int main(){
    twi_init();
    lcd_init();
    my_data=0;
    while(1){
        write_lcd_display();
    }
}
```

Για την υλοποίηση μας ορίσαμε μία συνάρτηση write_lcd_display η οποία θα κάνει ακριβώς το ζητούμενο της άσκησης. Στην αρχή αρχικοποιούμε τον δίαυλο επικοινωνίας και το LCD display και θέτουμε την global μεταβλητή my_data( δηλαδή μία θέση μνήμης στην μνήμη του μικροεπεξεργαστή) στην τιμή 0 . Η μεταβλητή my_data θα αποθηκεύει τον κώδικα ascii της τρέχουσας τιμής που φαίνεται στο display, ώστε να γνωρίζουμε πότε πρέπει να αλλάξουμε το display. Αφού πραγματοποιήσουμε αυτές τις αρχικοποιήσεις εκτελούμε ένα ατέρμονο while loop το οποίο καλεί σε κάθε επανάληψη την wirte_lcd_display . Στην write_lcd_display αρχικά θέτουμε το IO0 και κατ' επέκταση το EXT_PORT0 ως έξοδο και στην συνέχεια διαβάζουμε από το πληκτρολόγιο καλώντας την keypad_to_ascii() που είδαμε στο πρώτο ζήτημα. Αυτή μας επιστρέφει τον χαρακτήρα ascii που πατιέται την στιγμή αυτή στο πληκτρολόγιο . Στην συνέχεια ελέγχουμε αν η τιμή που επέστρεψε η keypad_to_ascii είναι ίση με 0 ή με την τρέχουσα έξοδο του display( δηλαδή ελέγχουμε αν στο πληκτρολόγιο δεν πατιέται κανένας χαρακτήρας ή αν αυτός που πατιέται είναι ίδιος με αυτόν που είχε πατηθεί στην προηγούμενη επανάληψη). Αν ο χαρακτήρας που πατιέται δεν άλλαξε σε σχέση με πριν ή αν δεν πατιέται κανένας χαρακτήρας τότε η ρουτίνα επιστρέφει χωρίς να κάνει τίποτα( αφού δεν θέλουμε να αλλάξει η έξοδος στο display αν δεν πατάμε κάποιον χαρακτήρα ή αν στο display έχουμε ήδη τον χαρακτήρα που πατιέται από προηγούμενη κλήση της write_lcd_display). Αν δεν ισχύει κάτι από τα παραπάνω τότε καθαρίζουμε το display και γράφουμε σε αυτό τον χαρακτήρα που μόλις πατήθηκε. Τέλος ενημερώνουμε την τιμή my_data η οποία περιέχει τον χαρακτήρα που φαίνεται τώρα στο display και η ρουτίνα επιστρέφει τον έλεγχο στο κύριο πρόγραμμα.

## *Ζήτημα τρίτο*

Στο ζήτημα αυτό μας ζητήθηκε να υλοποιήσουμε μία ηλεκτρονική κλειδαριά, στην οποία όταν ο χρήστης θα πατάει τον αριθμό της ομάδας μας (45) μέσω του πληκτρολογίου και θα ανάβουν για 4 sec τα LED του PORTB , ενώ αν πατηθεί οποιοσδήποτε άλλος συνδυασμός πλήκτρων θα αναβοσβήνουν τα LED του PORTB( με περίοδο 500msec) για 5sec. Σε κάθε περίπτωση θα πρέπει για 5sec αφού πατηθούν τα 2 πλήκτρα του keypad το πρόγραμμα να μην δέχεται άλλη είσοδο. Εμείς σε αυτό πραγματοποιήσαμε τις εξής προσθήκες: Όταν ο χρήστης πατάει ένα πλήκτρο στο lcd display εμφανίζεται το σύμβολο “*” που υποδηλώνει ότι ο χρήστης έχει εισάγει έναν αριθμό. Αν ο χρήστης βάλει τον σωστό κωδικό στο lcd_display γράφεται “Access permitted” ειδάλλως γράφεται “Access denied”. Ομοίως με το ζήτημα 2 και εδώ παραθέτουμε το κομμάτι του κώδικα που είναι χρήσιμο για αυτή την άσκηση καθώς η συναρτήσεις για τον χειρισμό της επικοινωνίας μέσω του διαύλου, αλλά και για το lcd display έχουν δοθεί σε άλλες ασκήσεις και στο ζήτημα 1. Ακολούθως παραθέτουμε τον κώδικα μας και στην συνέχεια τον εξηγούμε:

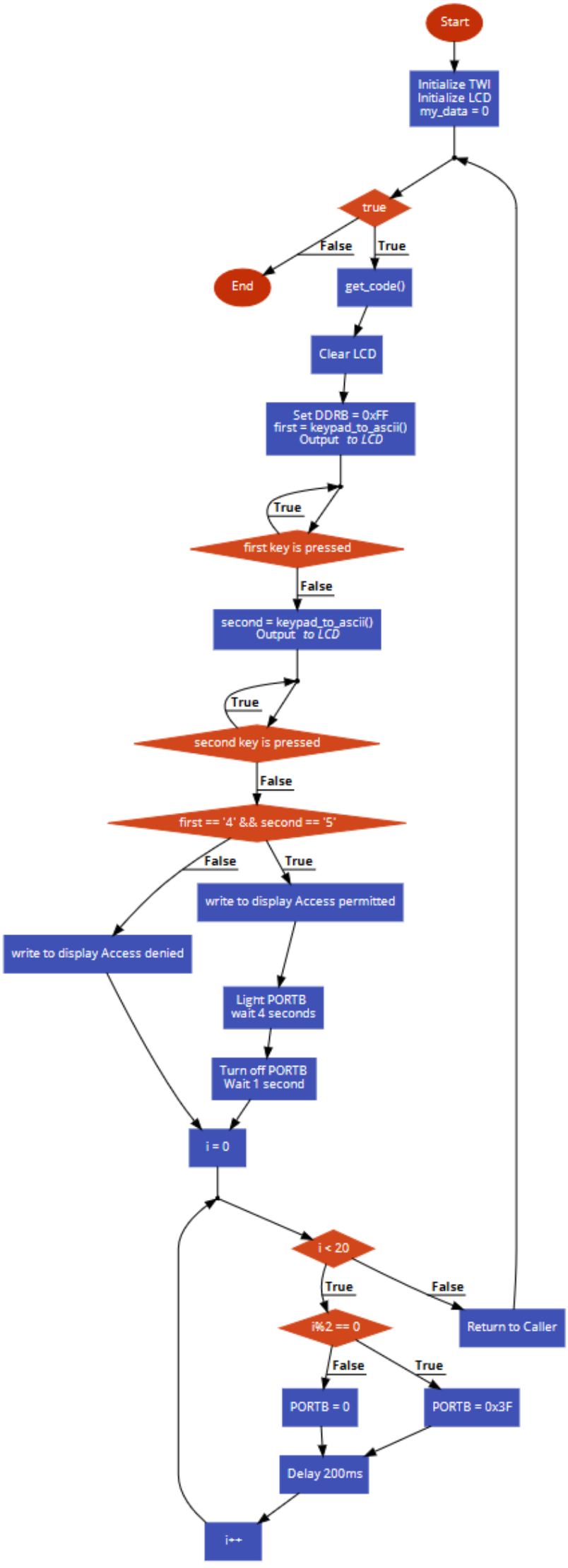
```
void lcd_string(const char *c){
    while(*c!='\0'){
        lcd_data(*c);
        c++;
    }
}

void correct_code(){
    lcd_clear_display();
    lcd_string("Access permitted");
}

void false_code(){
    lcd_clear_display();
    lcd_string("Access denied");
}

void get_code(){
    uint8_t first,second;
    lcd_clear_display();
    DDRB=0xFF;
    while((first=keypad_to_ascii())==0);
    lcd_data('*');
    while(keypad_to_ascii()!=0);
    while((second=keypad_to_ascii())==0 );
    lcd_data('*');
    while(keypad_to_ascii()!=0);
    if(first=='4' && second=='5'){
        correct_code();
        PORTB=0x3F;
```

```
        _delay_ms(4000);
        PORTB=0;
        _delay_ms(1000);
    }
    else{
        false_code();
        uint8_t i;
        for(i=0;i<20;i++){
            if(i%2==0)
                PORTB=0x3F;
            else
                PORTB=0;
            _delay_ms(250);
        }

    }
}

int main(){
    twi_init();
    lcd_init();
    while(1){
        get_code();
    }
}
```

Αρχικά, όπως και στο ζήτημα 2, αρχικοποιούμε την επικοινωνία μέσω του διαύλου καθώς και το lcd display. Στην συνέχεια εκτελούμε ένα ατέρμονο Loop που καλεί την συνάρτηση get_code που υλοποιεί το ζητούμενο της άσκησης. Πριν εξηγήσουμε όμως την get_code ας εξηγήσουμε κάποια σχετικά περιττά κομμάτια πρώτα. Η συνάρτηση lcd_string παίρνει σαν είσοδο ένα string και καλεί για κάθε character του string την lcd_data ώστε τα περιεχόμενα του string να απεικονιστούν στο display. Η correct_code() απλώς καθαρίζει το display και γράφει access permitted σε αυτό, ενώ η false_code() καθαρίζει το display και γράφει access denied σε αυτό . Εφόσον εξηγήσαμε αυτές τις βοηθητικές συναρτήσεις θα εξηγήσουμε το κύριο ζητούμενο της άσκησης την get_code(). Αρχικά, καθαρίζουμε το lcd display και θέτουμε το DDRB ως έξοδο. Στην συνέχεια εκτελούμε ένα while loop όπου αναθέτουμε το αποτέλεσμα της keyboard_to_ascii (που διαβάζει από το πληκτρολόγιο) σε μία μεταβλητή first , η οποία αποτελεί το πρώτο όρισμα του κωδικού. Το while loop αυτό σταματάει όταν το first πάρει κάποια τιμή και σταματήσει να είναι 0 ( που είναι η τιμή του null character). Στην συνέχεια τυπώνουμε στο display το * για να δείξουμε ότι το πρώτο πλήκτρο πατήθηκε και εκτελούμε ένα while loop που σταματάει όταν ο χρήστης αφήσει το πλήκτρο που πατάει. Επαναλαμβάνουμε την ίδια διαδικασία και για το δεύτερο ψηφίο. Στην συνέχεια ελέγχουμε εάν πατήθηκε ο σωστός κωδικός. Εάν ναι τότε τυπώνουμε στο display access permitted, ανάβουμε τα led του PORTB για 4sec και στην συνέχεια περιμένουμε 1sec ακόμα για να πετύχουμε την προδιαγραφή ότι πρέπει για 5sec το πρόγραμμα μας να μην δέχεται άλλη είσοδο , αφού λάβει μία είσοδο. Αντίθετα αν ο κωδικός είναι λάθος, το πρόγραμμα τυπώνει

στο display access denied και εκτελεί ένα for loop 20 φορές. Κάθε φορά που εκτελείται πραγματοποιείται μία καθυστέρηση 250msec(για συνολική καθυστέρηση 5 sec). Όταν η μεταβλητή που αλλάζει εντός του loop ( το i) παίρνει άρτιες τιμές τα LED του PORTB ανάβουν , ενώ όταν παίρνει περιττές τιμές τα LED του PORTB σβήνουν επιτυγχάνοντας το ζητούμενο.

Αυτή ήταν η υλοποίηση μας για το ζητούμενο αυτό. Στην συνέχεια παραθέτουμε το διάγραμμα ροής για το ζήτημα 3:


Start
Initialize TWI
Initialize LCD
my_data = 0
true
False
True
End
get_code()
Clear LCD
Set DDRB = 0xFF
first = keypad_to_ascii()
Output to LCD
True
first key is pressed
False
second = keypad_to_ascii()
Output to LCD
True
second key is pressed
False
first == '4' && second == '5'
False
True
write to display Access permitted
write to display Access denied
Light PORTB
wait 4 seconds
Turn off PORTB
Wait 1 second
i = 0
i < 20
True
False
Return to Caller
i%2 == 0
False
True
PORTB = 0
PORTB = 0x3F
Delay 200ms
i++