Βικέντιος Βιτάλης el18803

Εργαστήριο Μικροϋπολογιστών

7^{Ο} Εξάμηνο, Αναφοράς

3^{ης} εργαστηριακής άσκησης (2^{ης} AVR)

Ομάδα 03

Ζήτημα 3.1

Παρακάτω παρουσιάζεται ο κώδικας του ζητήματος 3.1, ενώ το διάγραμμα ροής βρίσκεται στο τέλος της παρούσας αναφοράς.

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

unsigned char r25, r24, w, x, y, z, r25_new, r24_new, temp_high, temp_low;


//Ρουτίνα καθυστέρησης x μsec
void wait_usec (int x){
	for(int i=0; i<x;  i++) {
		asm("nop"); //1 κύκλος(0.125 μsec)
		asm("nop"); //1 κύκλος(0.125 μsec)
		asm("nop"); //1 κύκλος(0.125 μsec)
		asm("nop"); //1 κύκλος(0.125 μsec)
	}
}
//Ρουτίνα καθυστέρησης y msec
void wait_msec (int y) {
	for(int i=0; i<y; i++) {
		//Καθυστέρηση 1μsec
		wait_usec(990);
	}
}

//Ρουτίνα ελέγχου ολόκληρου του πληκτρολογίου. Ελέγχει το πληκτρολόγιο γραμμή-
//γραμμή
void scan_keypad_sim(void)
{
	unsigned char row=0x10; //Επιλογή γραμμής. Επιλέγω γραμμή θέτοντας σε
	//λογικό 1 το  αντίστοιχο bit από τα 4 MSB της PORTC,
	//δηλαδή τα PC4,PC5,PC6,PC7

	w = x = y = z = 0x00;
```

```
	//Έλεγχος της πρώτης γραμμής 123A, θέτοντας PC4 = 1
	PORTC = row; //θέτω στην έξοδο PORTC τη μεταβλητή row, ώστε να ελέγξω τη
//γραμμή που μου δίνει η row, θέτοντας την αντίστοιχη γραμμή σε λογικό 1
	wait_usec(500); //καθυστέρηση που προστίθεται για τη σωστή λειτουργία
//του προγράμματος απομακρυσμένης πρόσβασης
	asm("nop");
	asm("nop"); //καθυστέρηση για να προλάβει να γίνει η αλλαγή κατάστασης
	w = 0x0F & PINC; //αποθηκεύω την είσοδο PINC στη μεταβλητή w και
//απομονώνω τα 4 LSB της, όπου τα 1 μου δείχνουν ποιοι διακόπτες είναι
//πατημένοι
	w = w << 4; //w = A3210000

	//Έλεγχος της δεύτερης γραμμής 456B
	row = row << 1; //ολίσθηση της μεταβλητής κατά μία θέση αριστερά για να
//επιλέξουμε την επόμενη γραμμή, θέτοντας ένα το επόμενο αριστερότερο bit της
//PORTC
	PORTC = row;
	wait_usec(500);
	asm("nop");
	asm("nop");
	x = 0x0F & PINC; //x = 0000B654

	//Έλεγος της τρίτης γραμμής 789C
	row = row << 1;
	PORTC = row;
	wait_usec(500);
	asm("nop");
	asm("nop");
	y = 0x0F & PINC;
	y = y << 4; //y = C9870000

	//Έλεγχος της τέταρτης γραμμής *0#D
	row = row << 1;
	PORTC = row;
	wait_usec(500);
	asm("nop");
	asm("nop");
	z = 0x0F & PINC; //z = 0000D#0*

	PORTC = 0x00; //προστέθηκε για την απομακρυσμένη πρόσβαση
}

//Ρουτίνα για καταγραφή των πλήκτρων που μόλις πατήθηκαν και αποφυγή του
φαινομένου του σπινθηρισμού
void scan_keypad_rising_edge_sim(void) {
	scan_keypad_sim(); //έλεγχος του πληκτρολογίου για πιεσμένους διακόπτες
	//και αποθήκευση του αποτελέσματος στις μεταβλητές r24, r25
	r25 = w + x; //r25 = A3210000 + 0000B654 = A321B654
	r24 = y + z; //r24 = C987D#0*
	wait_msec(15);
	scan_keypad_sim(); //έλεγχος του πληκτρολογίου ξανά
	r25_new = w + x;
	r24_new = y + z;
	//απόρριψη των πλήκτρων που εμφανίζουν σπινθηρισμό
	r24_new = r24_new & r24;
	r25_new = r25_new & r25;
	r25 = temp_low; //ΦΣόρτωσε την κατάσταση των διακοπτών στην προηγούμενη
κλήση της ρουτίνας
	r24 = temp_high;
	temp_high = r24_new; //Αποθήκευση της νέας κατάστασης των διακοπτών
	temp_low = r25_new;
	r25 = r25^0xFF;
```

```
	r24 = r24^0xFF;
	r24_new = r24_new & r24; //Βρες τους διακόπτες που έχουν μόλις πατηθεί
	r25_new = r25_new & r25;
}

//Ρουτίνα που επιστρέφει τον ascii κωδικό του χαρακτήρα που αντιστοιχεί στον
διακόπτη που έχει πατηθεί. Αν έχουν πατηθεί πολλοί διακόπτες, επιστρέφει τον
ascii του πρώτου διακόπτη που εντοπίζεται με βάση τη σειρά εξερεύνησης, δηλαδή
τον δεξιότερο πατημένο από τους A321B654C987D#0*. Η ρουτίνα καλείται μόνο αν
έχει πατηθεί κάποιο πλήκτρο του πληκτρολογίου
unsigned char keypad_to_ascii_sim(void) {

//Με τον key pointer γίνεται masking στην μεταβλητή r24 = C987D#0* ώστε να
ελέγξουμε ένα bit της σε κάθε επανάληψη. Η μεταβλητή pos θα καθορίσει σε ποια
θέση του πίνακα which_key θα σταματήσουμε
	unsigned char key_pointer = 0x01, which_key[16] = {'*', '0', '#', 'D',
'7', '8', '9', 'C', '4', '5', '6', 'B', '1', '2', '3', 'A'}, pos = 0;
	while(key_pointer != 0) {

	//Εξερευνώντας από τα LSB προς τα MSB, αν βρεις πατημένο διακόπτη,
επέστρεψε το, χαρακτήρα που βρίσκεται στη θέση pos του whick_key
		if ((r24_new & key_pointer) == key_pointer) {
			return which_key[pos];
		}

//Αλλιώς κάνε μία αριστερή ολίσθηση του key_pointer για να ελέγξεις το επόμενο
bit, και αύξησε το pos κατά 1
		else{
			key_pointer = key_pointer << 1;
			pos++;
		}
	}
	//Αν κανένας από τους διακόπτες που περιέχει η r24 δεν είναι πατημένος,
//ελέγχουμε τους διακόπτε της r25
	key_pointer = 0x01;
	while(key_pointer != 0) {
		if ((r25_new & key_pointer) == key_pointer) {
			return which_key[pos];
		}
		else{
			key_pointer = key_pointer << 1;
			pos++;
		}
	}
}

int main(void){
	unsigned char prwto, deytero;
	DDRC = 0xF0; //Τα 4 MSB της PORTC θέτονται ως έξοδοι, για την επιλογή
γραμμής, ενώ τα 4 LSB της θέτονται ως είσοδοι για τον έλεγχο στήλης
	DDRB = 0xFF; //η PORTB τίθεται ως έξοδος, για να ανάψουν τα αντίστοιψα
led
	scan_keypad_rising_edge_sim(); //Κλήση για αρχικοποίηση των μεταβλητών
temp και αποφυγή σφαλμάτων

	//Ανάγνωση του πρώτου ψηφίου
while(true) {
		scan_keypad_rising_edge_sim();
  //Αν τουλάχιστον μία εκ των μεταβλητών r24 και r25 είναι διαφορετική του
//μηδενός, δηλαδή αν τουλάχιστον ένα πλήκτρο του πληκτρολογίου πατήθηκε, βγες
//από το βρόχο. Αλλιώς συνέχισε να διαβάζεις το πληκτρολόγιο μέχρι να πατηθεί
//κάποιο πλήκτρο
```

```
                if(r24_new != 0 || r25_new != 0) {
                        break;
                }
        }
        prwto = keypad_to_ascii_sim(); //o ascii κωδικός του πρώτου πλήκτρου που
//πατάμε αποθηκεύεται στη μεταβλητή prwto
        wait_msec(15);

//Ανάγνωση του δεύτερου ψηφίου
        while(true) {
                scan_keypad_rising_edge_sim();
                if(r24_new != 0 || r25_new != 0) {
                        break;
                }
        }
        deytero = keypad_to_ascii_sim(); //Ο ascii κωδικός του δεύτερου πλήκτρου
//που πατάμε αποθηκεύεται στη μεταβλητή prwto
    scan_keypad_rising_edge_sim(); //προστίθεται για να μπορούμε να διαβάσουμε
//και άλλο πλήκτρο και να αγνοηθεί χωρίς να κολλήσει ο επεξεργαστής


        //Αν πληκτρολογήθηκε ο επιθυμητός αριθμός, δηλαδή το 03 (ομάδα 03)
        if(prwto == '0' && deytero == '1') {
                PORTB = 0xFF; //τότε άναψε όλα τα led της θύρας B
                wait_msec(4000); //για 4 sec
                PORTB = 0x00; //και στη συνέχεια σβήσε τα
        }
        else {
                unsigned char out = 0xFF;
                //Αλλιώς άναβε και σβήνε τα led για 0.5 sec, με 8 συνολικά
//αλλαγές κατάστασης
                for(int i=0; i<8; i++) {
                        PORTB = out;
                        wait_msec(500);
                        out = out^0xFF;
                }
        }
}
```

Ζήτημα 3.2

```
.INCLUDE "m16def.inc"


	; ---- Αρχή τμήματος δεδομένων
	.DSEG
	_tmp_: .byte 2

	; ---- Τέλος τμήματος δεδομένων
	.CSEG

	.org 0
	jmp start

start:


	ldi r24, (1 << PC7) | (1 << PC6) | (1 << PC5) | (1 << PC4) ; θέτει ως
;εξόδους τα 4 MSB
	out DDRC, r24 ; της θύρας PORTC

	ser r16
	out DDRB, r16 ; θέτουμε τη θύρα B ως έξοδο (για το χειρισμό των led)

	ser r24
	out DDRD, r24 ; θέτουμε τη θύρα D ως έξοδο (για το χειρισμό της οθόνης)

	ldi r24, low(RAMEND) ; αρχικοποίηση του stack pointer
	out SPL, r24
	ldi r24, high(RAMEND)
	out SPH, r24

	rjmp main



; καθυστέρηση τόσων μsec όσο το περιεχόμενο του r25:r24
wait_usec:
	sbiw r24 ,1 ; 2 κύκλοι (0.250 μsec)
	nop ; 1 κύκλος (0.125 μsec)
	nop ; 1 κύκλος (0.125 μsec)
	nop ; 1 κύκλος (0.125 μsec)
	nop ; 1 κύκλος (0.125 μsec)
	brne wait_usec ; 1 ή 2 κύκλοι (0.125 ή 0.250 μsec)
	ret ; 4 κύκλοι (0.500 μsec)

wait_msec:
	push r24 ; 2 κύκλοι (0.250 μsec)
	push r25 ; 2 κύκλοι
	ldi r24 , low(998) ; φόρτωσε τον καταχ. r25:r24 με 998 (1 κύκλος - 0.125
μsec)
	ldi r25 , high(998) ; 1 κύκλος (0.125 μsec)
	rcall wait_usec ; 3 κύκλοι (0.375 μsec), προκαλεί συνολικά καθυστέρηση
; 998.375 μsec
	pop r25 ; 2 κύκλοι (0.250 μsec)
	pop r24 ; 2 κύκλοι
	sbiw r24 , 1 ; 2 κύκλοι
	brne wait_msec ; 1 ή 2 κύκλοι (0.125 ή 0.250 μsec)
	ret ; 4 κύκλοι (0.500 μsec)
```

```
; σκανάρει τη γραμμή που υποδεικνύεται από το περιεχόμενο του r24
scan_row_sim:
	out PORTC, r25 ; η αντίστοιχη γραμμή τίθεται στο λογικό ‘1’
	push r24 ; τμήμα κώδικα που προστίθεται για τη σωστή
	push r25 ; λειτουργία του προγραμματος απομακρυσμένης
	ldi r24,low(500) ; πρόσβασης
	ldi r25,high(500)
	rcall wait_usec
	pop r25
	pop r24 ; τέλος τμήμα κώδικα
	nop
	nop ; καθυστέρηση για να προλάβει να γίνει η αλλαγή κατάστασης
	in r24, PINC ; επιστρέφουν οι θέσεις (στήλες) των διακοπτών που είναι
πιεσμένοι
	andi r24 ,0x0f ; απομονώνονται τα 4 LSB όπου τα ‘1’ δείχνουν που είναι
πατημένοι
	ret ; οι διακόπτες.

;Σκανάρει ολόκληρο το πληκτρολόγιο γραμμή-γραμμή. Στους r25:r24 αποθηκεύονται
οι διακόπτες με τη σειρά A321B654C987D#0*
scan_keypad_sim:
	push r26 ; αποθήκευσε τους καταχωρητές r27:r26 γιατι τους
	push r27 ; αλλάζουμε μέσα στην ρουτίνα
	ldi r25 , 0x10 ; έλεγξε την πρώτη γραμμή του πληκτρολογίου (PC4: 1 2 3
;A)
	rcall scan_row_sim
	swap r24 ; Αποθήκευσε το αποτέλεσμα
	mov r27, r24 ; στα 4 msb του r27
	ldi r25 ,0x20 ; έλεγξε τη δεύτερη γραμμή του πληκτρολογίου (PC5: 4 5 6
;B)
	rcall scan_row_sim
	add r27, r24 ; Αποθήκευσε το αποτέλεσμα στα 4 lsb του r27
	ldi r25 , 0x40 ; Έλεγξε την τρίτη γραμμή του πληκτρολογίου (PC6: 7 8 9
;C)
	rcall scan_row_sim
	swap r24 ; αποθήκευσε το αποτέλεσμα
	mov r26, r24 ; στα 4 msb του r26
	ldi r25 ,0x80 ; έλεγξε την τέταρτη γραμμή του πληκτρολογίου (PC7: * 0 #
D)
	rcall scan_row_sim
	add r26, r24 ; αποθήκευσε το αποτέλεσμα στα 4 lsb του r26
	movw r24, r26 ; μετέφερε το αποτέλεσμα στους καταχωρητές r25:r24
	clr r26 ; προστέθηκε για την απομακρυσμένη πρόσβαση
	out PORTC,r26 ; προστέθηκε για την απομακρυσμένη πρόσβαση
	pop r27 ; επανάφερε τους καταχωρητές r27:r26
	pop r26
	ret

;ρουτίνα για καταγραφή των πλήκτρων που μόλις πατήθηκαν και αποφυγή του
φαινομένου του σπινθηρισμού
scan_keypad_rising_edge_sim:
	push r22 ; αποθήκευσε τους καταχωρητές r23:r22 και τους
	push r23 ; r26:r27 γιατι τους αλλάζουμε μέσα στην ρουτίνα
	push r26
	push r27
	rcall scan_keypad_sim ; έλεγξε το πληκτρολόγιο για πιεσμένους διακόπτες
	push r24 ; και αποθήκευσε το αποτέλεσμα
	push r25
	ldi r24 ,15 ; καθυστέρησε 15 ms (τυπικές τιμές 10-20 msec που
;καθορίζεται από τον
```

```
	ldi r25 ,0 ; κατασκευαστή του πληκτρολογίου – χρονοδιάρκεια
;σπινθηρισμών)
	rcall wait_msec
	rcall scan_keypad_sim ; έλεγξε το πληκτρολόγιο ξανά και απόρριψε
	pop r23 ; όσα πλήκτρα εμφανίζουν σπινθηρισμό
	pop r22
	and r24 ,r22
	and r25 ,r23
	ldi r26 ,low(_tmp_) ; φόρτωσε την κατάσταση των διακοπτών στην
	ldi r27 ,high(_tmp_) ; προηγούμενη κλήση της ρουτίνας στους r27:r26
	ld r23 ,X+
	ld r22 ,X
	st X ,r24 ; αποθήκευσε στη RAM τη νέα κατάσταση
	st -X ,r25 ; των διακοπτών
	com r23
	com r22 ; βρες τους διακόπτες που έχουν «μόλις» πατηθεί
	and r24 ,r22
	and r25 ,r23
	pop r27 ; επανάφερε τους καταχωρητές r27:r26
	pop r26 ; και r23:r22
	pop r23
	pop r22
	ret

;Ρουτίνα που επιστρέφει στον r24 τον ascii κωδικό του χαρακτήρα που αντιστοιχεί
στον διακόπτη που έχει πατηθεί. Αν έχουν πατηθεί πολλοί διακόπτες, επιστρέφει
τον ascii του πρώτου διακόπτη που εντοπίζεται με βάση τη σειρά εξερεύνησης,
δηλαδή τον δεξιότερο πατημένο από τους A321B654C987D#0*. Αν δεν έχει πατηθεί
κανένα πλήκτρο, ο r24 παίρνει την τιμή 0x00
keypad_to_ascii_sim:
	push r26 ; αποθήκευσε τους καταχωρητές r27:r26 γιατι τους
	push r27 ; αλλάζουμε μέσα στη ρουτίνα
	movw r26 ,r24 ; λογικό ‘1’ στις θέσεις του καταχωρητή r26 δηλώνουν
	; τα παρακάτω σύμβολα και αριθμούς
	ldi r24 ,'*'
	; r26
	;C 9 8 7 D # 0 *
	sbrc r26 ,0
	rjmp return_ascii
	ldi r24 ,'0'
	sbrc r26 ,1
	rjmp return_ascii
	ldi r24 ,'#'
	sbrc r26 ,2
	rjmp return_ascii
	ldi r24 ,'D'
	sbrc r26 ,3 ; αν δεν είναι ‘1’παρακάμπτει την ret, αλλιώς (αν είναι ‘1’)
	rjmp return_ascii ; επιστρέφει με τον καταχωρητή r24 την ASCII τιμή του
;D.
	ldi r24 ,'7'
	sbrc r26 ,4
	rjmp return_ascii
	ldi r24 ,'8'
	sbrc r26 ,5
	rjmp return_ascii
	ldi r24 ,'9'
	sbrc r26 ,6
	rjmp return_ascii ;
	ldi r24 ,'C'
	sbrc r26 ,7
	rjmp return_ascii
	ldi r24 ,'4' ; Λογικό ‘1’ στις θέσεις του καταχωρητή r27 δηλώνουν
```

```
        sbrc r27 ,0 ; τα παρακάτω σύμβολα και αριθμούς
        rjmp return_ascii
        ldi r24 ,'5'
        ;r27
        ;A 3 2 1 B 6 5 4
        sbrc r27 ,1
        rjmp return_ascii
        ldi r24 ,'6'
        sbrc r27 ,2
        rjmp return_ascii
        ldi r24 ,'B'
        sbrc r27 ,3
        rjmp return_ascii
        ldi r24 ,'1'
        sbrc r27 ,4
        rjmp return_ascii
        ldi r24 ,'2'
        sbrc r27 ,5
        rjmp return_ascii
        ldi r24 ,'3'
        sbrc r27 ,6
        rjmp return_ascii
        ldi r24 ,'A'
        sbrc r27 ,7
        rjmp return_ascii
        clr r24
        rjmp return_ascii
        return_ascii:
        pop r27 ; επανάφερε τους καταχωρητές r27:r26
        pop r26
        ret

;Αποστολή ενός byte, 4 bit τη φορά στον ελεγκτή της οθόνης LCD. Το λογικό
επίπεδο που βρίσκεται ο ακροδέκτης που αντιστοιχεί στο σήμα R/S δεν
επηρεάζεται.
write_2_nibbles_sim:
        push r24 ; τμήμα κώδικα που προστίθεται για τη σωστή
        push r25 ; λειτουργία του προγραμματος απομακρυσμένης
        ldi r24 ,low(6000) ; πρόσβασης
        ldi r25 ,high(6000)
        rcall wait_usec
        pop r25
        pop r24 ; τέλος τμήμα κώδικα
        push r24 ; στέλνει τα 4 MSB
        in r25, PIND ; διαβάζονται τα 4 LSB και τα ξαναστέλνουμε
        andi r25, 0x0f ; για να μην χαλάσουμε την όποια προηγούμενη κατάσταση
        andi r24, 0xf0 ; απομονώνονται τα 4 MSB και
        add r24, r25 ; συνδυάζονται με τα προϋπάρχοντα 4 LSB
        out PORTD, r24 ; και δίνονται στην έξοδο
        sbi PORTD, PD3 ; δημιουργείται παλμός Enable στον ακροδέκτη PD3
        cbi PORTD, PD3 ; PD3=1 και μετά PD3=0
        push r24 ; τμήμα κώδικα που προστίθεται για τη σωστή
        push r25 ; λειτουργία του προγραμματος απομακρυσμένης
        ldi r24 ,low(6000) ; πρόσβασης
        ldi r25 ,high(6000)
        rcall wait_usec
        pop r25
        pop r24 ; τέλος τμήμα κώδικα
        pop r24 ; στέλνει τα 4 LSB. Ανακτάται το byte.
        swap r24 ; εναλλάσσονται τα 4 MSB με τα 4 LSB
        andi r24 ,0xf0 ; που με την σειρά τους αποστέλλονται
        add r24, r25
```

```
        out PORTD, r24
        sbi PORTD, PD3 ; Νέος παλμός Enable
        cbi PORTD, PD3
        ret

;Αποστολή ενός byte δεδομένων στον ελεγκτή της οθόνης lcd. Ο ελεγκτής πρέπει να
βρίσκεται σε 4 bit mode
lcd_data_sim:
        push r24 ; αποθήκευσε τους καταχωρητές r25:r24 γιατί τους
        push r25 ; αλλάζουμε μέσα στη ρουτίνα
        sbi PORTD, PD2 ; επιλογή του καταχωρητή δεδομένων (PD2=1)
        rcall write_2_nibbles_sim ; αποστολή του byte
        ldi r24 ,43 ; αναμονή 43μsec μέχρι να ολοκληρωθεί η λήψη
        ldi r25 ,0 ; των δεδομένων από τον ελεγκτή της lcd
        rcall wait_usec
        pop r25 ;επανάφερε τους καταχωρητές r25:r24
        pop r24
        ret

;Αποστολή μιας εντολής στον ελεγκτή της οθόνης lcd. Ο ελεγκτής πρέπει να
βρίσκεται σε 4 bit mode
lcd_command_sim:
        push r24 ; αποθήκευσε τους καταχωρητές r25:r24 γιατί τους
        push r25 ; αλλάζουμε μέσα στη ρουτίνα
        cbi PORTD, PD2 ; επιλογή του καταχωρητή εντολών (PD2=0)
        rcall write_2_nibbles_sim ; αποστολή της εντολής και αναμονή 39μsec
        ldi r24, 39 ; για την ολοκλήρωση της εκτέλεσης της από τον ελεγκτή της
;lcd.
        ldi r25, 0 ; ΣΗΜ.: υπάρχουν δύο εντολές, οι clear display και return
;home,
        rcall wait_usec ; που απαιτούν σημαντικά μεγαλύτερο χρονικό διάστημα.
        pop r25 ; επανάφερε τους καταχωρητές r25:r24
        pop r24
        ret

; αρχικοποίηση της οθόνης και ρυθμίσεις της οθόνης
lcd_init_sim:
        push r24 ; αποθήκευσε τους καταχωρητές r25:r24 γιατί τους
        push r25 ; αλλάζουμε μέσα στη ρουτίνα

        ldi r24, 40 ; Όταν ο ελεγκτής της lcd τροφοδοτείται με
        ldi r25, 0 ; ρεύμα εκτελεί την δική του αρχικοποίηση.
        rcall wait_msec ; Αναμονή 40 msec μέχρι αυτή να ολοκληρωθεί.
        ldi r24, 0x30 ; εντολή μετάβασης σε 8 bit mode
        out PORTD, r24 ; επειδή δεν μπορούμε να είμαστε βέβαιοι
        sbi PORTD, PD3 ; για τη διαμόρφωση εισόδου του ελεγκτή
        cbi PORTD, PD3 ; της οθόνης, η εντολή αποστέλλεται δύο φορές
        ldi r24, 39
        ldi r25, 0 ; εάν ο ελεγκτής της οθόνης βρίσκεται σε 8-bit mode
        rcall wait_usec ; δεν θα συμβεί τίποτα, αλλά αν ο ελεγκτής έχει
διαμόρφωση
         ; εισόδου 4 bit θα μεταβεί σε διαμόρφωση 8 bit
        push r24 ; τμήμα κώδικα που προστίθεται για τη σωστή
        push r25 ; λειτουργία του προγραμματος απομακρυσμένης
        ldi r24,low(1000) ; πρόσβασης
        ldi r25,high(1000)
        rcall wait_usec
        pop r25
        pop r24 ; τέλος τμήμα κώδικα
        ldi r24, 0x30
        out PORTD, r24
        sbi PORTD, PD3
```

```
        cbi PORTD, PD3
        ldi r24,39
        ldi r25,0
        rcall wait_usec
        push r24 ; τμήμα κώδικα που προστίθεται για τη σωστή
        push r25 ; λειτουργία του προγραμματος απομακρυσμένης
        ldi r24 ,low(1000) ; πρόσβασης
        ldi r25 ,high(1000)
        rcall wait_usec
        pop r25
        pop r24 ; τέλος τμήμα κώδικα
        ldi r24,0x20 ; αλλαγή σε 4-bit mode
        out PORTD, r24
        sbi PORTD, PD3
        cbi PORTD, PD3
        ldi r24,39
        ldi r25,0
        rcall wait_usec
        push r24 ; τμήμα κώδικα που προστίθεται για τη σωστή
        push r25 ; λειτουργία του προγραμματος απομακρυσμένης
        ldi r24 ,low(1000) ; πρόσβασης
        ldi r25 ,high(1000)
        rcall wait_usec
        pop r25
        pop r24 ; τέλος τμήμα κώδικα
        ldi r24,0x28 ; επιλογή χαρακτήρων μεγέθους 5x8 κουκίδων

        rcall lcd_command_sim ; και εμφάνιση δύο γραμμών στην οθόνη
        ldi r24,0x0c ; ενεργοποίηση της οθόνης, απόκρυψη του κέρσορα
        rcall lcd_command_sim
        ldi r24,0x01 ; καθαρισμός της οθόνης
        rcall lcd_command_sim
        ldi r24, low(1530)
        ldi r25, high(1530)
        rcall wait_usec
        ldi r24 ,0x06 ; ενεργοποίηση αυτόματης αύξησης κατά 1 της διεύθυνσης
        rcall lcd_command_sim ; που είναι αποθηκευμένη στον μετρητή διευθύνσεων
;και

         ; απενεργοποίηση της ολίσθησης ολόκληρης της οθόνης
        pop r25 ; επανάφερε τους καταχωρητές r25:r24
        pop r24
        ret


; βασικό πρόγραμμα
main:

        rcall lcd_init_sim ; αρχικοποίηση οθόνης

        rcall scan_keypad_rising_edge_sim ;κλήση για αρχικοποίηση των μεταβλητών
temp και αποφυγή σφαλμάτων

try1:

        rcall scan_keypad_rising_edge_sim ;ανάγνωση του πρώτου ψηφίου
        rcall keypad_to_ascii_sim ;επιστροφή του ascii κωδικού στον r24
        cpi r24, 0x00
        breq try1 ;αν ο r24=0, δηλαδή αν δεν έχει πατηθεί κανένα πλήκτρο, κάνε
;άλμα στο try1 αλλιώς συνέχισε
        mov r28, r24 ;αποθήκευση του ascii του πρώτου πλήκτρου που πατήθηκε στο
;r28
```

```
try2:
	rcall scan_keypad_rising_edge_sim
	rcall keypad_to_ascii_sim
	cpi r24, 0x00
	breq try2
	mov r29, r24 ;αποθήκευση του ascii του δεύτερου πλήκτρου που πατήθηκε
;στο r29
	rcall scan_keypad_rising_edge_sim ;προστίθεται για να μπορούμε να
;διαβάσουμε και άλλο πλήκτρο και να αγνοηθεί χωρίς να κολλήσει ο επεξεργαστής
	cpi r28, '1' ;σύγκρινε τον ascii του πρώτου πλήκτρου με τον ascii του
;χαρακτήρα ‘1’
	brne wrongpsw ;αν δεν είναι ίσοι τότε άλμα στην ετικέτα wrongpsw
	cpi r29, '6' ;σύγκρινε τον ascii του δεύτερου πλήκτρου με τον ascii του
;χαρακτήρα ‘6’
	brne wrongpsw ;αν δεν είναι ίσοι τότε άλμα στην ετικέτα wrongpsw
	rjmp correctpsw ;αλλιώς, άλμα στην ετικέτα correctpsw

wrongpsw:

	clr r24
	ldi r24, 'A'
	rcall lcd_data_sim ;αποστολή του ascii κωδικού του χαρακτήρα ‘A’, ως
;byte δεδομένων
	ldi r24, 'L' ;κάθε χαρακτήρας του μηνήματος «ALARM ON» στέλνεται ως byte
;δεδομένων, ένας τη φορά
	rcall lcd_data_sim
	ldi r24, 'A'
	rcall lcd_data_sim
	ldi r24, 'R'
	rcall lcd_data_sim
	ldi r24, 'M'
	rcall lcd_data_sim
	ldi r24, ' '
	rcall lcd_data_sim
	ldi r24, 'O'
	rcall lcd_data_sim
	ldi r24, 'N'
	rcall lcd_data_sim

	ser r16 ;όλα τα bit του r16 τίθενται σε λογικό 1 ώστε μόλις τον
;εμφανίσομε αρχικά στην B,  να ανάψουν όλα τα leds της B
	ldi r17, 0x07 ;θέλουμε 8 επαναλήψεις

	blinking:
	out PORTB, r16 ;εμφάνισε στην θύρα B το περιεχόμενο του r16
	ldi r24, 0xF4 ;φόρτωσε στο ζεύγος r25:24 τον αριθμό 01F4 = 500
	ldi r25, 0x01
	rcall wait_msec ;καθυστέρηση 500 msec
	com r16 ;αντιστροφή όλων των bit του r16, ώστε στην επόμενη επανάληψη να
;αλλάξουν κατάσταση όλα τα leds της B
	dec r17 ;μείωση του μετρητή επαναληψεων κατά 1
	brne blinking ;όσο ο μετρητής r17 δεν είναι 0, ξανά άλμα στην blinking
	out PORTB, r16 ;σβήσιμο των led μετά το πέρας των 8 αλλαγών κατάστασης
;(4 sec)

	rcall scan_keypad_rising_edge_sim ;προστίθεται για να μπορούμε να
;διαβάσουμε και άλλο πλήκτρο και να αγνοηθεί χωρίς να κολλήσει ο επεξεργαστής
	rjmp main ;άλμα στην αρχή του βασικού προγράμματος


correctpsw:
```

```
	clr r24
	ldi r24, 'W' ;κάθε χαρακτήρας του μηνήματος «WELCOME 03» στέλνεται ως
byte δεδομένων, ένας τη φορά
	rcall lcd_data_sim
	ldi r24, 'E'
	rcall lcd_data_sim
	ldi r24, 'L'
	rcall lcd_data_sim
	ldi r24, 'C'
	rcall lcd_data_sim
	ldi r24, 'O'
	rcall lcd_data_sim
	ldi r24, 'M'
	rcall lcd_data_sim
	ldi r24, 'E'
	rcall lcd_data_sim
	ldi r24, ' '
	rcall lcd_data_sim
	ldi r24, '0'
	rcall lcd_data_sim
	ldi r24, '3'
	rcall lcd_data_sim
	ser r16
	out PORTB, r16 ;άναψε όλα τα led της B
	ldi r24, 0xA0 ;φόρτωσε στο ζεύγος r25:24 τον αριθμό 0FA0 = 4000
	ldi r25, 0x0F
	rcall wait_msec ;καθυστέρηση 4000msec = 4sec
	clr r16
	out PORTB, r16 ;σβήσε όλα τα led της B
	rcall scan_keypad_rising_edge_sim ;προστίθεται για να μπορούμε να
διαβάσουμε και άλλο πλήκτρο και να αγνοηθεί χωρίς να κολλήσει ο επεξεργαστής
	rjmp main ;άλμα στην αρχή του βασικού προγράμματος
```

Διάγραμμα Ροής Προγράμματος

Βικέντιος Βιτάλης el18803