ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Εργαστήριο Μικροϋπολογιστών

$3^{η}$ εργαστηριακή άσκηση

EasyAvr6

Αναστάσιος Λαγός - 03113531

Άσκηση 1

Το διάγραμμα ροής βρίσκεται στο τέλος του pdf.

Κώδικας

```c
#undef F_CPU
#define F_CPU 8000000UL //8MHz

#ifndef __DELAY_BACKWARD_COMPATIBLE__
#define __DELAY_BACKWARD_COMPATIBLE__   //Allows function _delay_us/_delay_ms to accept variable arguments, not just constant int.
#endif

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

#define NOP(){__asm__ __volatile__("nop");} //assembly nop

/*Keypad Functions*/
unsigned int scan_row_sim(unsigned int row);
void scan_keypad_sim();
unsigned int scan_keypad_rising_edge_sim(unsigned int flick_time);
unsigned char keypad_to_ascii_sim();

unsigned int swap(unsigned int val);

/*Delay functions*/
void wait_usec(unsigned int delay);
void wait_msec(unsigned int delay);

unsigned int buttons[2], ram[2];

/*function that delays for delay time*/
void wait_usec(unsigned int delay){
    unsigned int i;
    for(i = 0; i < (delay/10); i++) {    //10 usec delay for delay/10 times
        _delay_us(10);
    }
    if (delay % 10) {    //delay for the remainder accordingly
        _delay_us(delay % 10);
    }
}
/*same function as wait_usec but for milliseconds*/
void wait_msec(unsigned int delay) {
    unsigned int i ;
    for(i = 0; i < (delay / 10); i++){
        _delay_ms(10);
```

```
    }
    if(delay % 10) {
        _delay_ms(delay % 10);
    }
}
/*Function that does __asm__("swap")*/
unsigned int swap(unsigned int val) {
    return ((val & 0x0F) << 4 | (val & 0xF0) >> 4);
}

/*Function that returns which columns are pressed for a given row*/
unsigned int scan_row_sim(unsigned int row) {
    PORTC = row;//Search in line row.
    wait_usec(500); //Delay required for a successful remote operation

    NOP();
    NOP();//Delay to allow for a change of state

    return PINC & 0x0F; //Return 4 LSB
}
/*Function that scans the whole keypad*/
void scan_keypad_sim() {
    buttons[1] = swap(scan_row_sim(0x10));// A 3 2 1
    buttons[1] += scan_row_sim(0x20);// A 3 2 1 B 6 5 4

    buttons[0] = swap(scan_row_sim(0x40));// C 9 8 7
    buttons[0] += (scan_row_sim(0x80)); // C 9 8 7 D # 0 *

    PORTC = 0x00; // added only for the remote operation
    return;
}
/*Function that checks which buttons where pressed since its last call*/
unsigned int scan_keypad_rising_edge_sim(unsigned int flick_time) {
    scan_keypad_sim(); // do the first scan
    unsigned int temp[2]; // store first scan
    temp[0] = buttons[0];
    temp[1] = buttons[1];
    wait_msec(flick_time); //wait for flick time

    scan_keypad_sim();  //scan second time
    buttons[0] &= temp[0]; //remove flick values
    buttons[1] &= temp[1];

    temp[0] = ram[0];   //get the last state from previous call to rising_edge
    ↪  from "RAM"
    temp[1] = ram[1];
    ram[0] = buttons[0];    //update the new previous state
    ram[1] = buttons[1];
```

```
    buttons[0] &= ~temp[0]; //Keep values that change from 1 to 0
    buttons[1] &= ~temp[1];

    return (buttons[0] || buttons[1]);
}
/*Function that returns the ASCII code of button pressed*/
unsigned char keypad_to_ascii_sim() {
    unsigned int select;
    for(select = 0x01; select <= 0x80; select <<= 1) {
        switch(buttons[0] & select) {
            case 0x01:
                return '*';
            case 0x02:
                return '0';
            case 0x04:
                return '#';
            case 0x08:
                return 'D';
            case 0x10:
                return '7';
            case 0x20:
                return '8';
            case 0x40:
                return '9';
            case 0x80:
                return 'C';
        }
    }
    for(select = 0x01; select <= 0x80; select <<= 1) {
        switch(buttons[1] & select) {
            case 0x01:
                return '4';
            case 0x02:
                return '5';
            case 0x04:
                return '6';
            case 0x08:
                return 'B';
            case 0x10:
                return '1';
            case 0x20:
                return '2';
            case 0x40:
                return '3';
            case 0x80:
                return 'A';
        }
    }
```

```
    return 0;
}

int main(void)
{
    DDRC = 0xF0; //Initialize PORTC and LEDs. Internal resistor pull up must be
    ↪  deactivated
    DDRB = 0xFF;

    while(1) {
        unsigned char button1, button2;
        ram[0] = 0, ram[1] = 0;
        PORTB = 0x00;
        while(1) {
            if(scan_keypad_rising_edge_sim(15)) {
                button1 = keypad_to_ascii_sim();
                break;
            }
        }
        while(1) {
            if(scan_keypad_rising_edge_sim(15)) {
                button2 = keypad_to_ascii_sim();
                break;
            }
        }
        if ((button1 == '7') & (button2 == '1')) {
            PORTB = 0xFF;
            wait_msec(4000);
        }
        else {
            for (int i = 0; i < 4; i++) {
                PORTB = 0xFF;
                wait_msec(500);
                PORTB = 0x00;
                wait_msec(500);
            }
        }
    }
}
```

Άσκηση 2

Κώδικας

```
.DSEG
_tmp_:.byte 2
.CSEG

.include "m16def.inc"

.org 0x00
rjmp main

main:
    ldi r24, low(RAMEND) ; Initialize stack pointer
    out SPL, r24
    ldi r24, high(RAMEND)
    out SPH, r24

    clr r20
    clr r21

    ldi r24, (1 << PC7)|(1 << PC6)|(1 << PC5)|(1 << PC4)    ;Initialize 4 MSB of
    ↪ PORTC as outputs
    out DDRC, r24

    ser r24
    out DDRB, r24   ;Initialize LEDs
    out DDRD, r24   ;Initialize PORTD(LCD) as output


    first_button:
        clr r24
        out PORTB, r24  ;LEDs off
        ldi r24, 0x0F   ;flicker time 15us
        rcall scan_keypad_rising_edge_sim
        mov r22, r24    ;get all button
        or r22, r25     ;states on r22 (only one bit will be set)
        cpi r22, 0
        breq first_button   ; if no button is pressed then loop till a button is
        ↪ pressed
    first_ascii:
        rcall keypad_to_ascii_sim
        mov r20, r24    ;first button

    second_button:
        ldi r24, 0x0F
        rcall scan_keypad_rising_edge_sim
```

```
        mov r22, r24
        or r22, r25
        cpi r22, 0
        breq second_button
    second_ascii:
        rcall keypad_to_ascii_sim
        mov r21, r24    ;second button

    check:
        cpi r20, '7'
        brne wrong
        cpi r21, '1'
        brne wrong  ;if this condition is wrong then we have a correct psw

    correct:    ;display message, LEDs one -> 4 secs -> LEDs off -> clr LCD
        clr r24
        rcall lcd_init_sim
        ldi r24, 'W'
        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, '7'
        rcall lcd_data_sim
        ldi r24, '1'
        rcall lcd_data_sim

        ldi r24, 0x0F
        rcall scan_keypad_rising_edge_sim   ;for successful remote operation
        ser r24
        out PORTB, r24
        ldi r24, low(4000)
        ldi r25, high(4000)
        rcall wait_msec

        clr r24
        rcall lcd_init_sim
```

```
91
92          jmp first_button
93
94      wrong:  ;display alarm -> LEDs = 4x(on x 0.5sec -> off x 0.5sec) -> clr LCD
95          clr r24
96          rcall lcd_init_sim
97          ldi r24, 'A'
98          rcall lcd_data_sim
99          ldi r24, 'L'
100         rcall lcd_data_sim
101         ldi r24, 'A'
102         rcall lcd_data_sim
103         ldi r24, 'R'
104         rcall lcd_data_sim
105         ldi r24, 'M'
106         rcall lcd_data_sim
107         ldi r24, ' '
108         rcall lcd_data_sim
109         ldi r24, 'O'
110         rcall lcd_data_sim
111         ldi r24, 'N'
112         rcall lcd_data_sim
113
114         ldi r24, 0x0F
115         rcall scan_keypad_rising_edge_sim   ;for successful remote operation
116         ldi r24,4
117         loop:
118         push r24
119         ser r24
120         out PORTB, r24
121         ldi r24, low(500)
122         ldi r25, high(500)
123         rcall wait_msec
124         clr r24
125         out PORTB, r24
126         ldi r24, low(500)
127         ldi r25, high(500)
128         rcall wait_msec
129         pop r24
130         subi r24, 1
131         brne loop
132
133         clr r24
134         rcall lcd_init_sim
135
136     jmp first_button
137
138     wait_msec:  ;Routine that waits msec equal to r25:r24
```

```
139         push r24
140         push r25
141         ldi r24, low(1000)
142         ldi r25, high(1000)
143         rcall wait_usec
144         pop r25
145         pop r24
146         sbiw r24, 1
147         brne wait_msec
148
149         ret
150
151     wait_usec:  ;Routine that waits usec equal to r25:r24.Only for this routine,
        ↪  instruction cycles are taken into account
152         sbiw r24, 1 ;2 cycles
153         nop
154         nop
155         nop
156         nop
157         brne wait_usec ;1 cycle the majority of the time
158
159         ret
160
161     scan_row_sim:
162         out PORTC, r25 ; η αντίστοιχη γραμμή τίθεται στο λογικό '1'
163         push r24 ; τμήμα κώδικα που προστίθεται για τη σωστή
164         push r25 ; λειτουργία του προγραμματος απομακρυσμένης
165         ldi r24,low(500) ; πρόσβασης
166         ldi r25,high(500)
167         rcall wait_usec
168         pop r25
169         pop r24 ; τέλος τμήμα κώδικα
170         nop
171         nop ; καθυστέρηση για να προλάβει να γίνει η αλλαγή κατάστασης
172         in r24, PINC ; επιστρέφουν οι θέσεις (στήλες) των διακοπτών που είναι
            ↪  πιεσμένοι
173         andi r24 ,0x0f ; απομονώνονται τα 4 LSB όπου τα '1' δείχνουν που είναι
            ↪  πατημένοι
174         ret ; οι διακόπτες
175
176     scan_keypad_sim:
177         push r26 ; αποθήκευσε τους καταχωρητές r27:r26 γιατι τους
178         push r27 ; αλλάζουμε μέσα στην ρουτίνα
179         ldi r25 , 0x10 ; έλεγξε την πρώτη γραμμή του πληκτρολογίου (PC4: 1 2 3
            ↪  A)
180         rcall scan_row_sim
181         swap r24 ; αποθήκευσε το αποτέλεσμα
182         mov r27, r24 ; στα 4 msb του r27
```

```
183         ldi r25 ,0x20 ; έλεγξε τη δεύτερη γραμμή του πληκτρολογίου (PC5: 4 5 6
            ↪ B)
184         rcall scan_row_sim
185         add r27, r24 ; αποθήκευσε το αποτέλεσμα στα 4 lsb του r27
186         ldi r25 , 0x40 ; έλεγξε την τρίτη γραμμή του πληκτρολογίου (PC6: 7 8 9
            ↪ C)
187         rcall scan_row_sim
188         swap r24 ; αποθήκευσε το αποτέλεσμα
189         mov r26, r24 ; στα 4 msb του r26
190         ldi r25 ,0x80 ; έλεγξε την τέταρτη γραμμή του πληκτρολογίου (PC7: * 0 #
            ↪ D)
191         rcall scan_row_sim
192         add r26, r24 ; αποθήκευσε το αποτέλεσμα στα 4 lsb του r26
193         movw r24, r26 ; μετέφερε το αποτέλεσμα στους καταχωρητές r25:r24
194         clr r26 ; προστέθηκε για την απομακρυσμένη πρόσβαση
195         out PORTC,r26 ; προστέθηκε για την απομακρυσμένη πρόσβαση
196         pop r27 ; επανάφερε τους καταχωρητές r27:r26
197         pop r26
198         ret
199
200     scan_keypad_rising_edge_sim:
201         push r22 ; αποθήκευσε τους καταχωρητές r23:r22 και τους
202         push r23 ; r26:r27 γιατι τους αλλάζουμε μέσα στην ρουτίνα
203         push r26
204         push r27
205         rcall scan_keypad_sim ; έλεγξε το πληκτρολόγιο για πιεσμένους διακόπτες
206         push r24 ; και αποθήκευσε το αποτέλεσμα
207         push r25
208         ldi r24 ,15 ; καθυστέρησε 15 ms (τυπικές τιμές 10-20 msec που
            ↪ καθορίζεται από τον
209         ldi r25 ,0 ; κατασκευαστή του πληκτρολογίου ( χρονοδιάρκεια
            ↪ σπινθηρισμών)
210         rcall wait_msec
211         rcall scan_keypad_sim ; έλεγξε το πληκτρολόγιο ξανά και απόρριψε
212         pop r23 ; όσα πλήκτρα εμφανίζουν σπινθηρισμό
213         pop r22
214         and r24 ,r22
215         and r25 ,r23
216         ldi r26 ,low(_tmp_) ; φόρτωσε την κατάσταση των διακοπτών στην
217         ldi r27 ,high(_tmp_) ; προηγούμενη κλήση της ρουτίνας στους r27:r26
218         ld r23 ,X+
219         ld r22 ,X
220         st X ,r24 ; αποθήκευσε στη RAM τη νέα κατάσταση
221         st -X ,r25 ; των διακοπτών
222         com r23
223         com r22 ; βρες τους διακόπτες που έχουν «μόλις» πατηθεί
224         and r24 ,r22
225         and r25 ,r23
```

```
226         pop r27 ; επανάφερε τους καταχωρητές r27:r26
227         pop r26 ; και r23:r22
228         pop r23
229         pop r22
230         ret
231
232     keypad_to_ascii_sim:
233         push r26 ; αποθήκευσε τους καταχωρητές r27:r26 γιατι τους
234         push r27 ; αλλάζουμε μέσα στη ρουτίνα
235         movw r26 ,r24 ; λογικό '1' στις θέσεις του καταχωρητή r26 δηλώνουν
236         ; τα παρακάτω σύμβολα και αριθμούς
237         ldi r24 ,'*'
238         ; r26
239         ;C 9 8 7 D # 0 *
240         sbrc r26 ,0
241         rjmp return_ascii
242         ldi r24 ,'0'
243         sbrc r26 ,1
244         rjmp return_ascii
245         ldi r24 ,'#'
246         sbrc r26 ,2
247         rjmp return_ascii
248         ldi r24 ,'D'
249         sbrc r26 ,3 ; αν δεν είναι '1'παρακάμπτει την ret, αλλιώς (αν είναι '1')
250         rjmp return_ascii ; επιστρέφει με τον καταχωρητή r24 την ASCII τιμή του
            ↪ D.
251         ldi r24 ,'7'
252         sbrc r26 ,4
253         rjmp return_ascii
254         ldi r24 ,'8'
255         sbrc r26 ,5
256         rjmp return_ascii
257         ldi r24 ,'9'
258         sbrc r26 ,6
259         rjmp return_ascii ;
260         ldi r24 ,'C'
261         sbrc r26 ,7
262         rjmp return_ascii
263         ldi r24 ,'4' ; λογικό '1' στις θέσεις του καταχωρητή r27 δηλώνουν
264         sbrc r27 ,0 ; τα παρακάτω σύμβολα και αριθμούς
265         rjmp return_ascii
266         ldi r24 ,'5'
267         ;r27
268         ;A 3 2 1 B 6 5 4
269         sbrc r27 ,1
270         rjmp return_ascii
271         ldi r24 ,'6'
272         sbrc r27 ,2
```

```
273         rjmp return_ascii
274         ldi r24 ,'B'
275         sbrc r27 ,3
276         rjmp return_ascii
277         ldi r24 ,'1'
278         sbrc r27 ,4
279         rjmp return_ascii ;
280         ldi r24 ,'2'
281         sbrc r27 ,5
282         rjmp return_ascii
283         ldi r24 ,'3'
284         sbrc r27 ,6
285         rjmp return_ascii
286         ldi r24 ,'A'
287         sbrc r27 ,7
288         rjmp return_ascii
289         clr r24
290         rjmp return_ascii
291         return_ascii:
292         pop r27 ; επανάφερε τους καταχωρητές r27:r26
293         pop r26
294         ret
295
296     write_2_nibbles_sim:
297         push r24 ; τμήμα κώδικα που προστίθεται για τη σωστή
298         push r25 ; λειτουργία του προγραμματος απομακρυσμένης
299         ldi r24 ,low(6000) ; πρόσβασης
300         ldi r25 ,high(6000)
301         rcall wait_usec
302         pop r25
303         pop r24 ; τέλος τμήμα κώδικα
304         push r24 ; στέλνει τα 4 MSB
305         in r25, PIND ; διαβάζονται τα 4 LSB και τα ξαναστέλνουμε
306         andi r25, 0x0f ; για να μην χαλάσουμε την όποια προηγούμενη κατάσταση
307         andi r24, 0xf0 ; απομονώνονται τα 4 MSB και
308         add r24, r25 ; συνδυάζονται με τα προϋπάρχοντα 4 LSB
309         out PORTD, r24 ; και δίνονται στην έξοδο
310         sbi PORTD, PD3 ; δημιουργείται παλμός Enable στον ακροδέκτη PD3
311         cbi PORTD, PD3 ; PD3=1 και μετά PD3=0
312         push r24 ; τμήμα κώδικα που προστίθεται για τη σωστή
313         push r25 ; λειτουργία του προγραμματος απομακρυσμένης
314         ldi r24 ,low(6000) ; πρόσβασης
315         ldi r25 ,high(6000)
316         rcall wait_usec
317         pop r25
318         pop r24 ; τέλος τμήμα κώδικα
319         pop r24 ; στέλνει τα 4 LSB. Ανακτάται το byte.
320         swap r24 ; εναλλάσσονται τα 4 MSB με τα 4 LSB
```

```
321         andi r24 ,0xf0 ; που με την σειρά τους αποστέλλονται
322         add r24, r25
323         out PORTD, r24
324         sbi PORTD, PD3 ; Νέος παλμός Enable
325         cbi PORTD, PD3
326         ret
327
328     lcd_data_sim:
329         push r24
330         push r25
331         sbi PORTD,PD2
332         rcall write_2_nibbles_sim
333         ldi r24,43
334         ldi r25,0
335         rcall wait_usec
336         pop r25
337         pop r24
338         ret
339
340     lcd_command_sim:
341         push r24 ; αποθήκευσε τους καταχωρητές r25:r24 γιατί τους
342         push r25 ; αλλάζουμε μέσα στη ρουτίνα
343         cbi PORTD, PD2 ; επιλογή του καταχωρητή εντολών (PD2=0)
344         rcall write_2_nibbles_sim ; αποστολή της εντολής και αναμονή 39μsec
345         ldi r24, 39 ; για την ολοκλήρωση της εκτέλεσης της από τον ελεγκτή της
            ↪ lcd.
346         ldi r25, 0 ; ΣΗΜ.: υπάρχουν δύο εντολές, οι clear display και return
            ↪ home,
347         rcall wait_usec ; που απαιτούν σημαντικά μεγαλύτερο χρονικό διάστημα.
348         pop r25 ; επανάφερε τους καταχωρητές r25:r24
349         pop r24
350         ret
351
352     lcd_init_sim:
353         push r24 ; αποθήκευσε τους καταχωρητές r25:r24 γιατί τους
354         push r25 ; αλλάζουμε μέσα στη ρουτίνα
355
356         ldi r24, 40 ; Όταν ο ελεγκτής της lcd τροφοδοτείται με
357         ldi r25, 0 ; ρεύμα εκτελεί την δική του αρχικοποίηση.
358         rcall wait_msec ; Αναμονή 40 msec μέχρι αυτή να ολοκληρωθεί.
359         ldi r24, 0x30 ; εντολή μετάβασης σε 8 bit mode
360         out PORTD, r24 ; επειδή δεν μπορούμε να είμαστε βέβαιοι
361         sbi PORTD, PD3 ; για τη διαμόρφωση εισόδου του ελεγκτή
362         cbi PORTD, PD3 ; της οθόνης, η εντολή αποστέλλεται δύο φορές
363         ldi r24, 39
364         ldi r25, 0 ; εάν ο ελεγκτής της οθόνης βρίσκεται σε 8-bit mode
365         rcall wait_usec ; δεν θα συμβεί τίποτα, αλλά αν ο ελεγκτής έχει
            ↪ διαμόρφωση
```

```
366         ; εισόδου 4 bit θα μεταβεί σε διαμόρφωση 8 bit
367         push r24 ; τμήμα κώδικα που προστίθεται για τη σωστή
368         push r25 ; λειτουργία του προγραμματος απομακρυσμένης
369         ldi r24,low(1000) ; πρόσβασης
370         ldi r25,high(1000)
371         rcall wait_usec
372         pop r25
373         pop r24 ; τέλος τμήμα κώδικα
374         ldi r24, 0x30
375         out PORTD, r24
376         sbi PORTD, PD3
377         cbi PORTD, PD3
378         ldi r24,39
379         ldi r25,0
380         rcall wait_usec
381         push r24 ; τμήμα κώδικα που προστίθεται για τη σωστή
382         push r25 ; λειτουργία του προγραμματος απομακρυσμένης
383         ldi r24 ,low(1000) ; πρόσβασης
384         ldi r25 ,high(1000)
385         rcall wait_usec
386         pop r25
387         pop r24 ; τέλος τμήμα κώδικα
388         ldi r24,0x20 ; αλλαγή σε 4-bit mode
389         out PORTD, r24
390         sbi PORTD, PD3
391         cbi PORTD, PD3
392         ldi r24,39
393         ldi r25,0
394         rcall wait_usec
395         push r24 ; τμήμα κώδικα που προστίθεται για τη σωστή
396         push r25 ; λειτουργία του προγραμματος απομακρυσμένης
397         ldi r24 ,low(1000) ; πρόσβασης
398         ldi r25 ,high(1000)
399         rcall wait_usec
400         pop r25
401         pop r24 ; τέλος τμήμα κώδικα
402         ldi r24,0x28 ; επιλογή χαρακτήρων μεγέθους 5x8 κουκίδων
403         rcall lcd_command_sim ; και εμφάνιση δύο γραμμών στην οθόνη
404         ldi r24,0x0c ; ενεργοποίηση της οθόνης, απόκρυψη του κέρσορα
405         rcall lcd_command_sim
406         ldi r24,0x01 ; καθαρισμός της οθόνης
407         rcall lcd_command_sim
408         ldi r24, low(1530)
409         ldi r25, high(1530)
410         rcall wait_usec
411         ldi r24 ,0x06 ; ενεργοποίηση αυτόματης αύξησης κατά 1 της διεύθυνσης
412         rcall lcd_command_sim ; που είναι αποθηκευμένη στον μετρητή διευθύνσεων
            ↪ και
```

```
413         ; απενεργοποίηση της ολίσθησης ολόκληρης της οθόνης
414         pop r25 ; επανάφερε τους καταχωρητές r25:r24
415         pop r24
416         ret
417
```

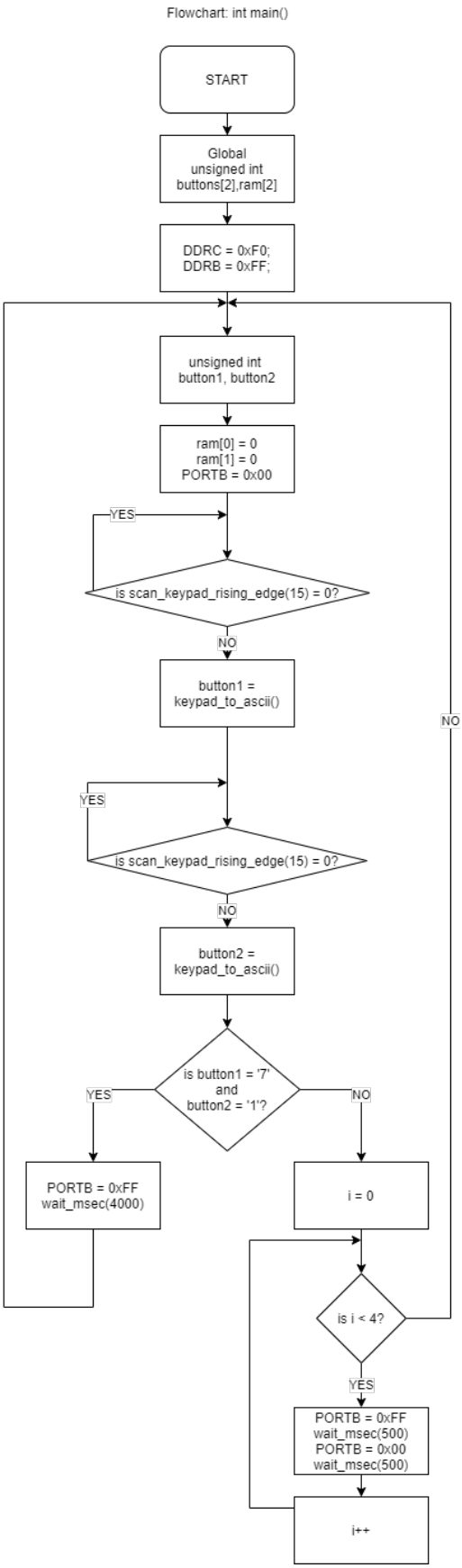

Figure 1: main

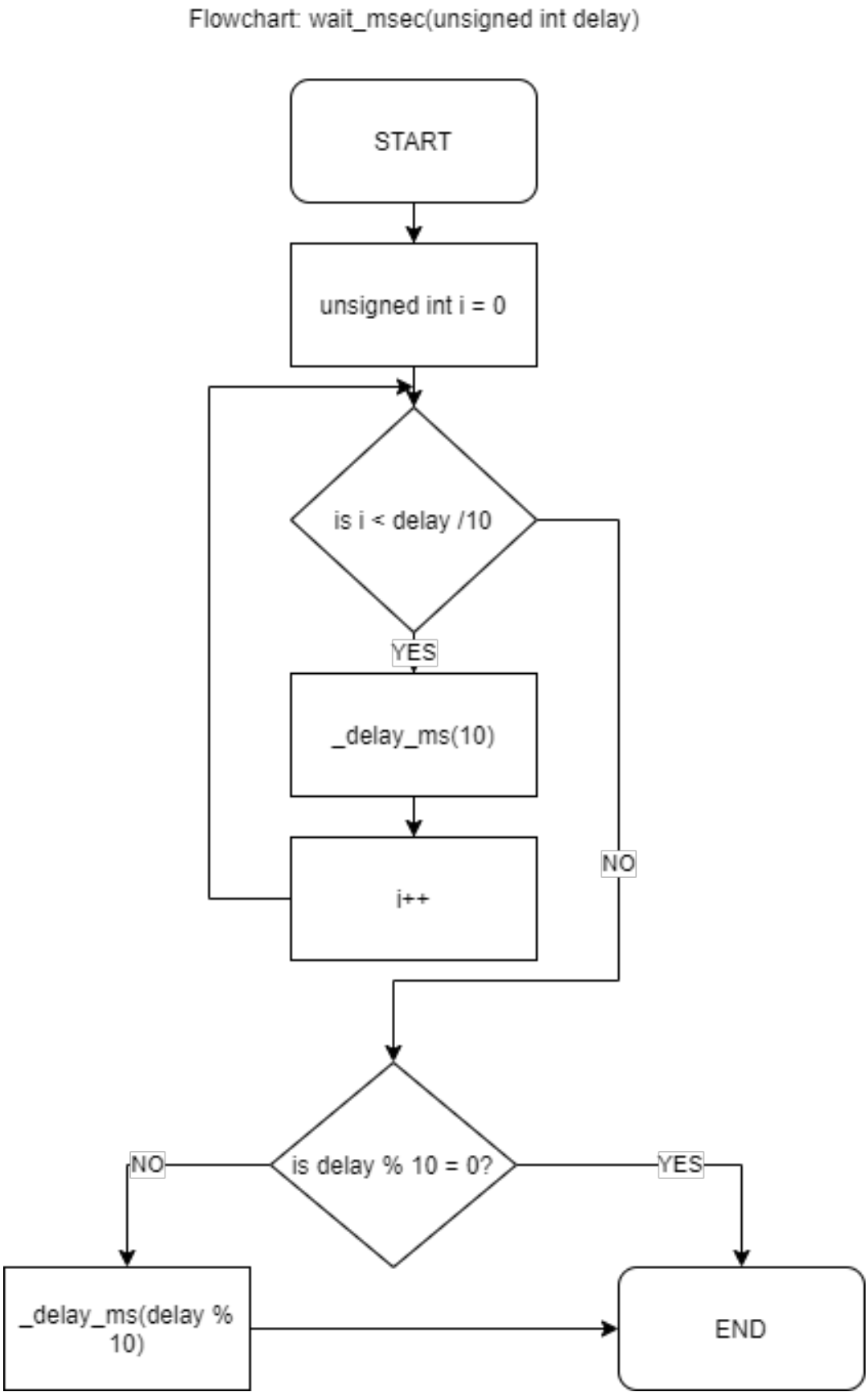

Figure 2: wait_msec

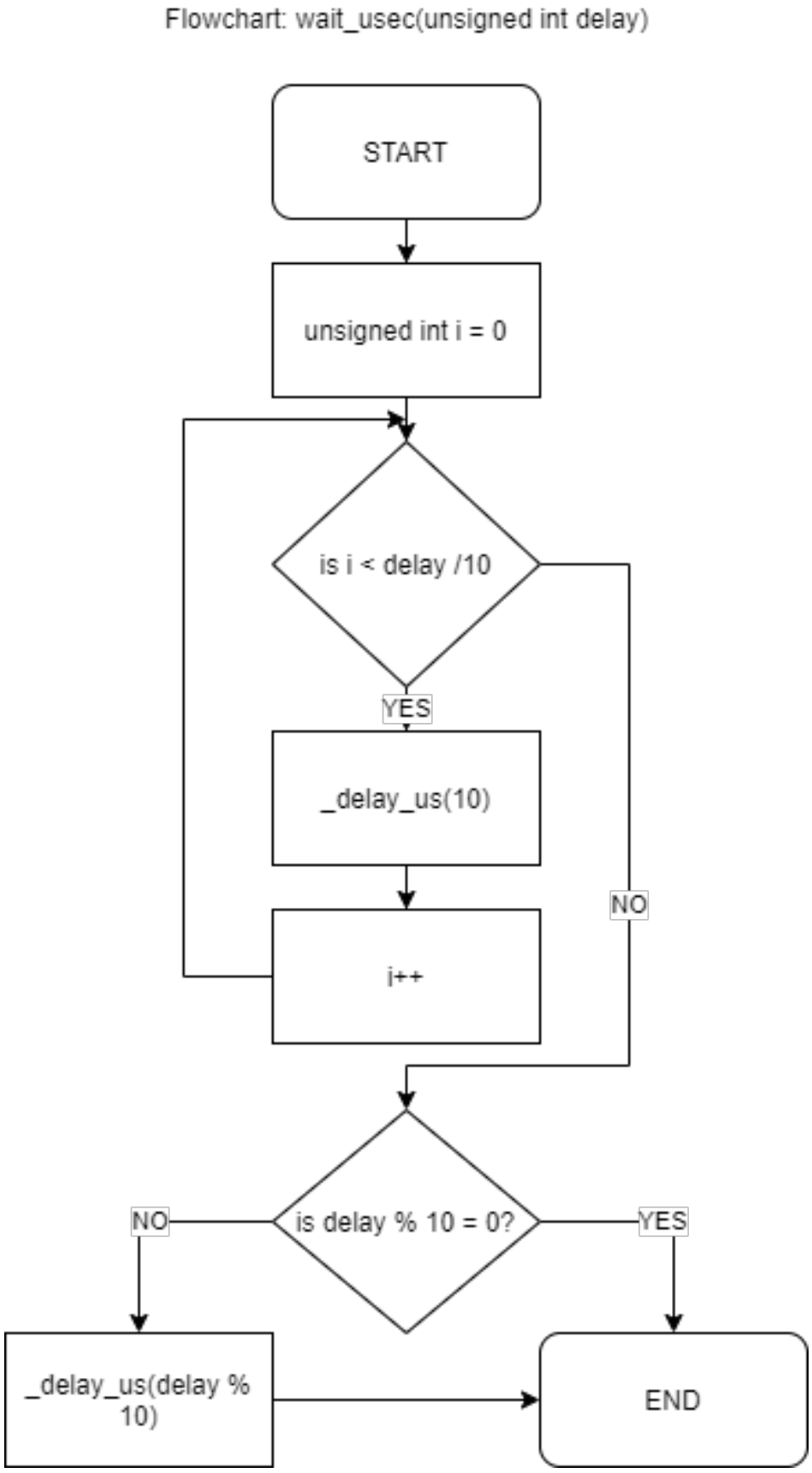

Figure 3: wait_usec

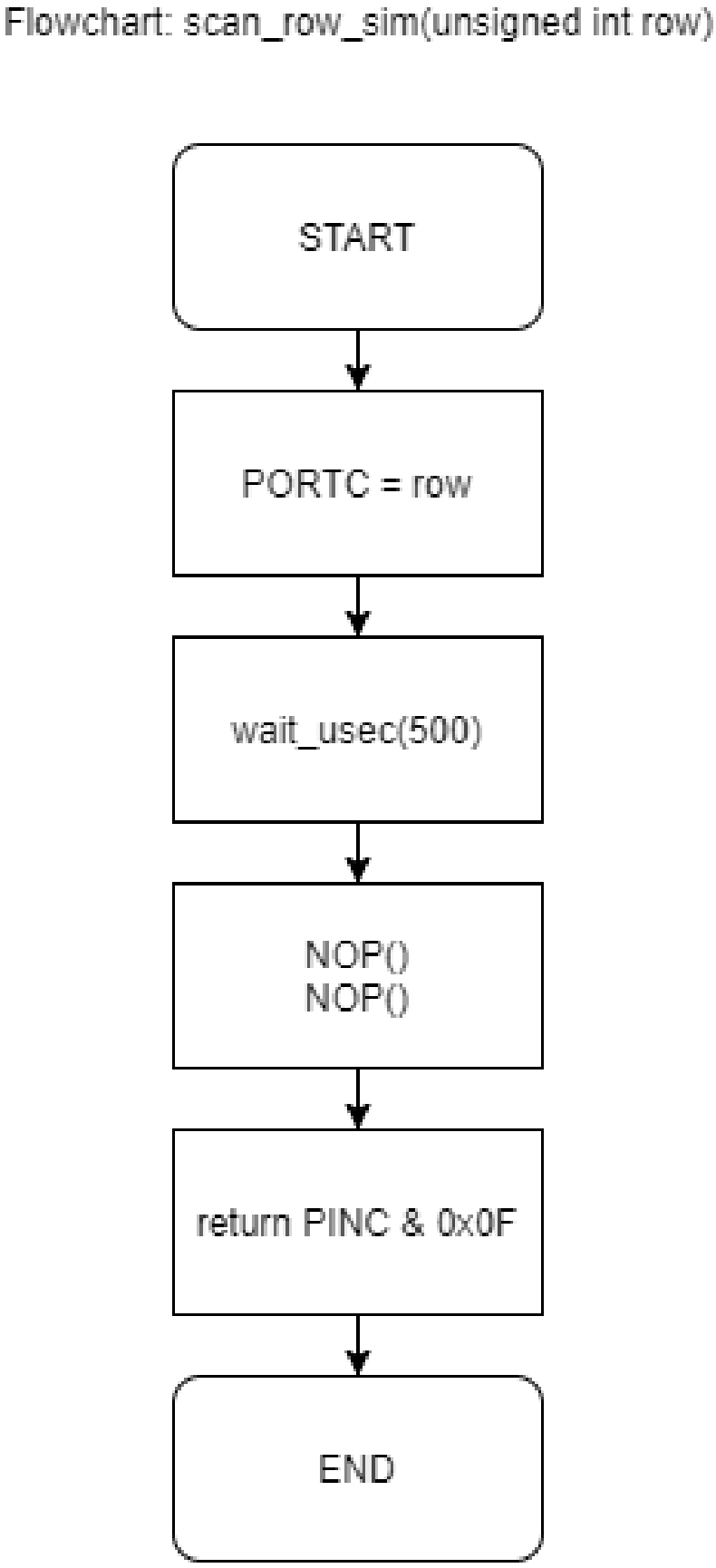

Figure 4: swap

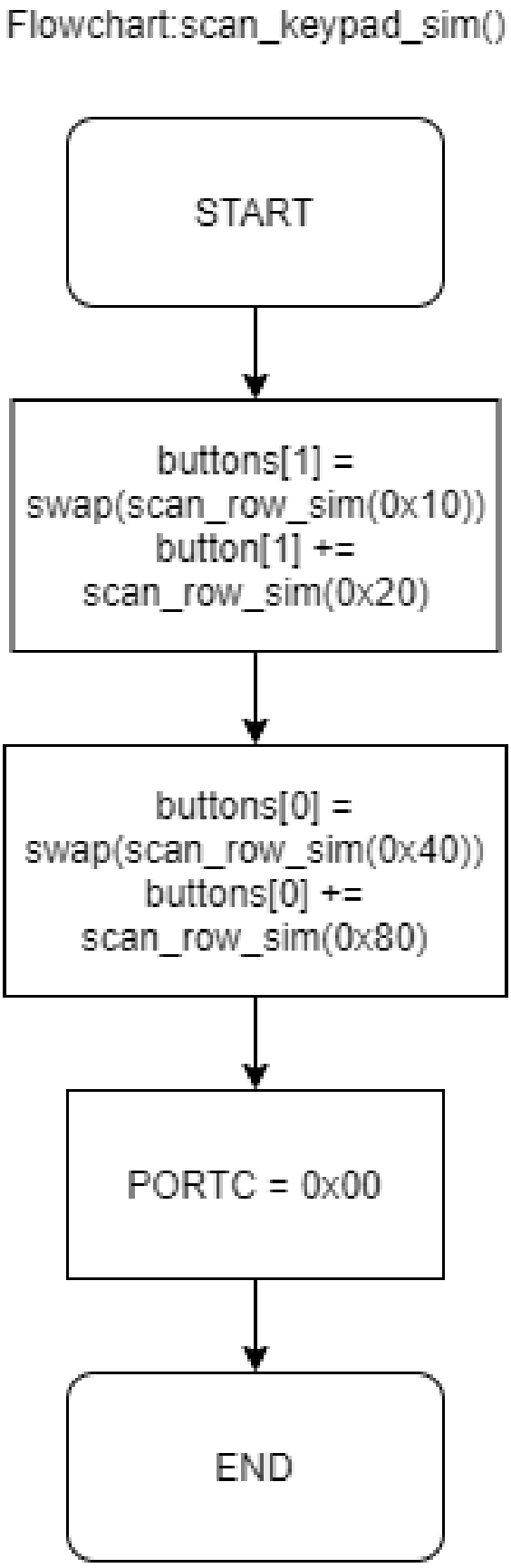

Figure 5: scan_keypad

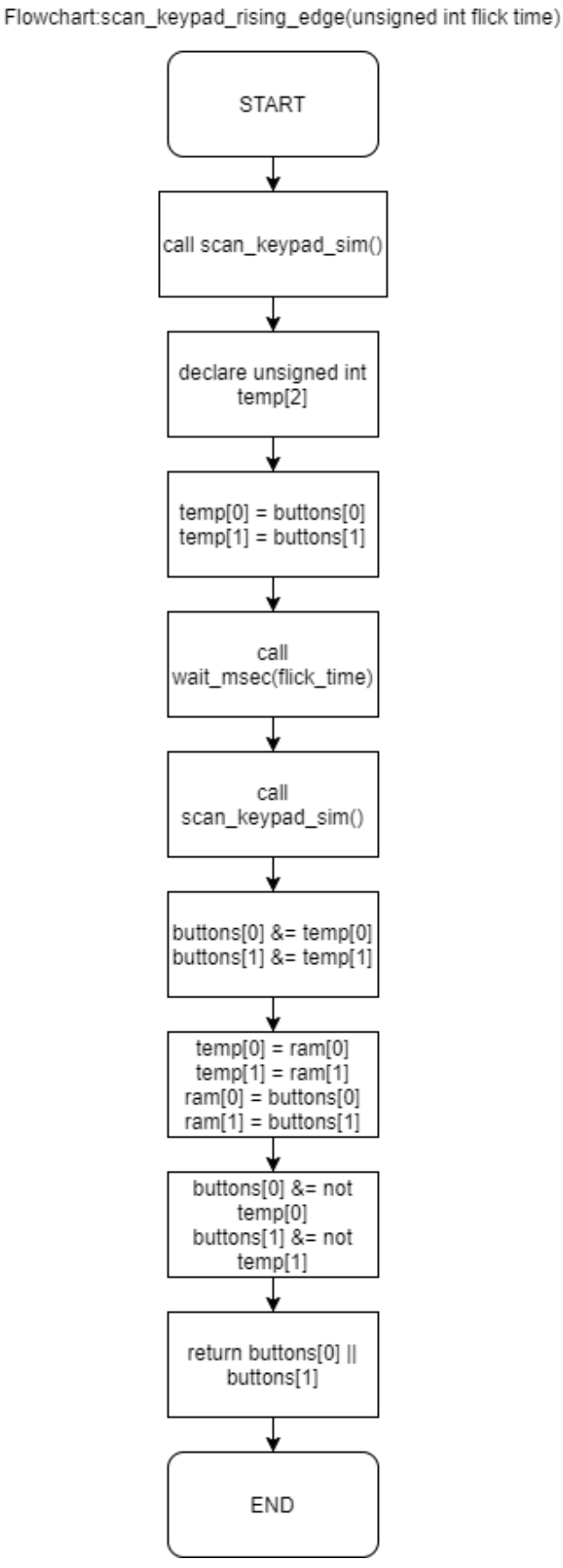

Figure 6: scan_keypad_rising_edge

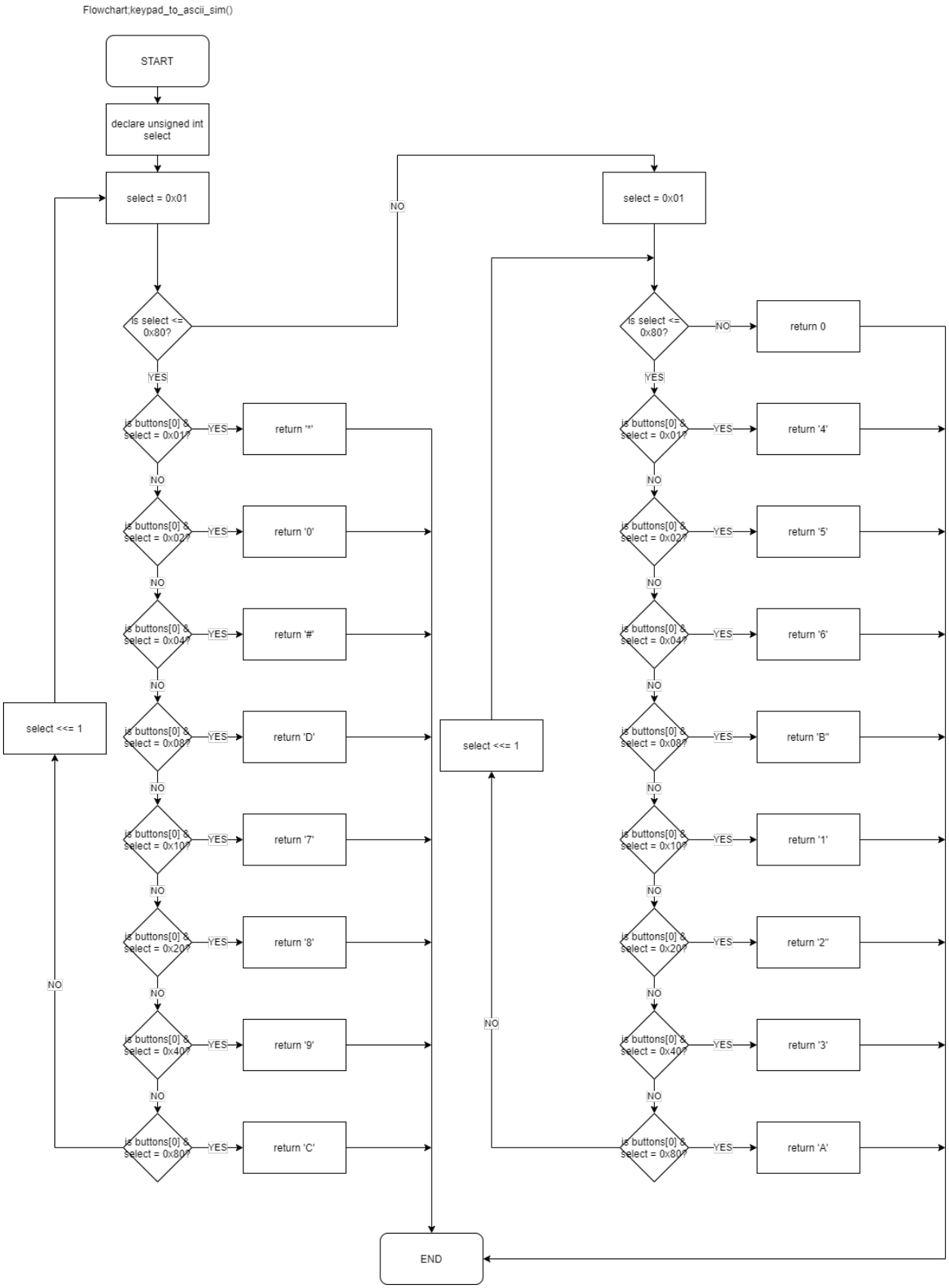

Figure 7: keypad_to_ascii