Διασύνδεση LCD με τον Z80

Εργασία για το μάθημα
Μικροεπεξεργαστές

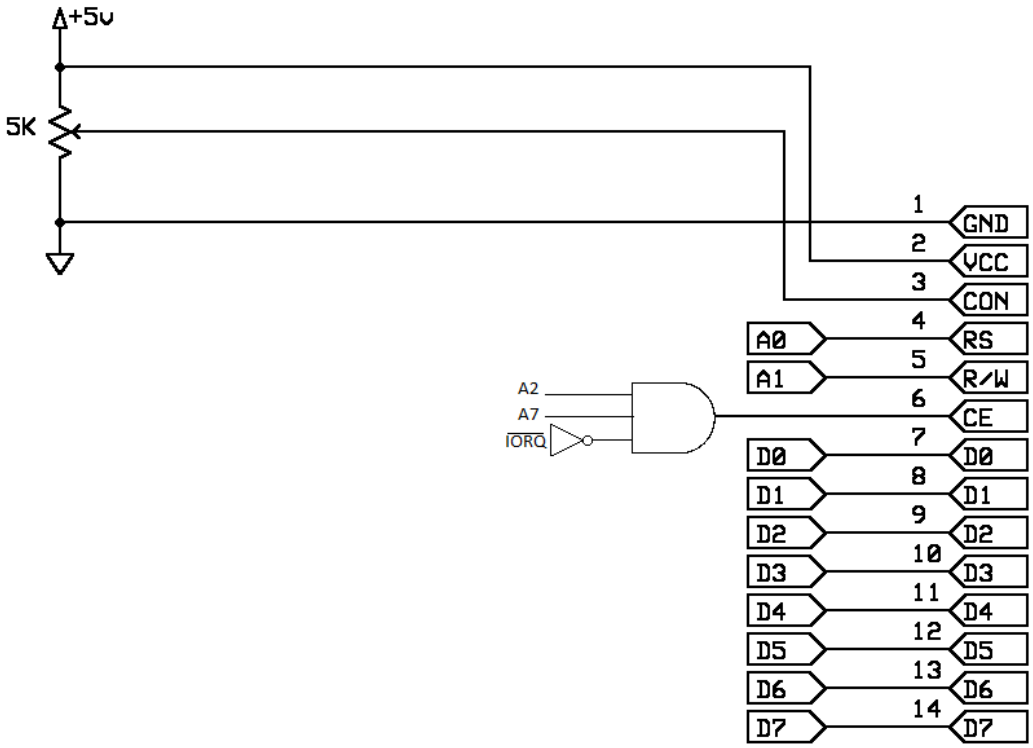
Υπεύθυνος Καθηγητής: Κ. Αδαός

Όνομα: Λαγός Δημήτρης
Α.Μ: 4480
Έτος: ΣΤ’

1. ΕΡΓΑΣΙΑ

Ζητούμενο της εργασίας ήταν η συγγραφή τόσο του κώδικα όσο και ο σχεδιασμός της διασύνδεσης μιας lcd οθόνης με τον μικροεπεξεργαστή Zilog Z80. Πιο συγκεκριμένα οθόνη της εταιρίας Crystalfontz America, Inc , δύο γραμμών επί δεκαέξι χαρακτήρων βασισμένη στον μικροελεγκτή HD44780 της Hitachi.

2. ΔΙΑΣΥΝΔΕΣΗ



Αριστερά φαίνονται τα σήματα που έρχονται από τον Z80 και δεξιά οι ακροδέκτες της οθόνης στους οποίους καταλήγουν. Το σήμα A7, έχει προστεθεί στο σχηματικό για να χρησιμοποιηθεί σαν σήμα επιλογής σε περίπτωση σύνδεσης περισσότερων περιφεριακών στον Z80.

3. ΓΕΝΙΚΗ ΜΟΡΦΗ ΚΩΔΙΚΑ

Για την εκπόνηση της εργασίας επέλεξα να δημιουργήσω μια βιβλιοθήκη επικοινωνίας του Z80 με την οθόνη, η οποία θα χειρίζεται όλα τα σήματα και τις μεθόδους που απαιτούνται ώστε ο τελικός χρήστης να μπορεί να χρησιμοποιήσει την οθόνη με την κλήση απλών συναρτήσεων. Επίσης η βιβλιοθήκη περιέχει έναν buffer ο οποίος τυπώνει στην οθόνη αυτόματα 32 χαρακτήρες από την μνήμη, καθώς επίσης περιέχονται συναρτήσεις οι οποίες εκτελούν όλες τις βασικές λειτουργίες της οθόνης αυτόματα(πχ. δεξιά/αριστερή ολίσθηση του κειμένου , καθαρισμός οθόνης, κέρσορας κτλ.).

→Για να χρησιμοποιήσει ο χρήστης τον buffer αρκεί να φορτώσει στην μεταβλητή *buffer_place* την διεύθυνση της μνήμης που περιέχεται ο πρώτος προς εκτύπωση χαρακτήρας και στην συνέχεια να καλέσει την συνάρτηση *buffer*.

Π.χ.:

```
LD BC, 80FFh; φόρτωσε στον BC την διεύθυνση του
              πρώτου προς εκτύπωση χαρακτήρα

LD (buffer_place), BC; αποθήκευσε την διεύθυνση στην
                       μεταβλητή buffer_place

CALL buffer; κάλεσε τον buffer
```

Σημείωση: Ο buffer μπορεί να κληθεί επαναλαμβανόμενα. Κάθε νέα κλήση του buffer επιστρέφει τον κέρσορα στην αρχική θέση και κάθε νέος χαρακτήρας αντικαθιστά τον παλαιότερο. Μπορεί ο χρήστης ανάμεσα στις κλήσεις των buffer να καλεί την συνάρτηση καθαρισμού της οθόνης (*clear*) ώστε η οθόνη να είναι κενή για κάθε νέα σειρά από 32 χαρακτήρες.

Π.χ.:

```
LD BC, 80FFh; φόρτωσε στον BC την διεύθυνση του
              πρώτου προς εκτύπωση χαρακτήρα

LD (buffer_place), BC; αποθήκευσε την διεύθυνση στην
                       μεταβλητή buffer_place

CALL buffer; κάλεσε τον buffer

CALL clear; καθάρισε την οθόνη

LD BC, 88A1h; φόρτωσε στον BC την διεύθυνση του
              πρώτου προς εκτύπωση χαρακτήρα

LD (buffer_place), BC; αποθήκευσε την διεύθυνση στην

   μεταβλητή buffer_place
```

→Για να στείλει έναν χαρακτήρα ο χρήστης στην οθόνη, αρκεί να αποθηκεύσει την 16δική αναπαράσταση του χαρακτήρα στην μεταβλητή *char* και να καλέσει την συνάρτηση *send_char*.

Π.χ.:

```
LD A, 06Dh; φόρτωσε στον A το γράμμα m

LD (char), A; Αποθήκευσε το στην μεταβλητή char

CALL send_char; κάλεσε την συνάρτηση send_char
```

→Για να στείλει μια εντολή ο χρήστης στην οθόνη, αρκεί να αποθηκεύσει την 16δική αναπαράσταση της εντολής στην μεταβλητή *command* και να καλέσει την συνάρτηση *send_command*.

Π.χ.:

```
LD A, 01h; Καθάρισε την οθόνη

LD (command), A; Αποθήκευσε το στην μεταβλητή
                 command

CALL send_command; κάλεσε την συνάρτηση
                  send_command
```

Σημείωση: Η δεκαεξαδική αναπαράσταση τόσο των χαρακτήρων όσο και των εντολών μπορούν να βρεθούν στο datasheet του controller HD44780 όπως και στο datasheet της υλοποίησης της οθόνης από την Crystalfontz. Στο τελευταίο κεφάλαιο της εργασίας προσφέρονται οι σύνδεσμοι και για τα δύο datasheet.

Λειτουργίες οθόνης

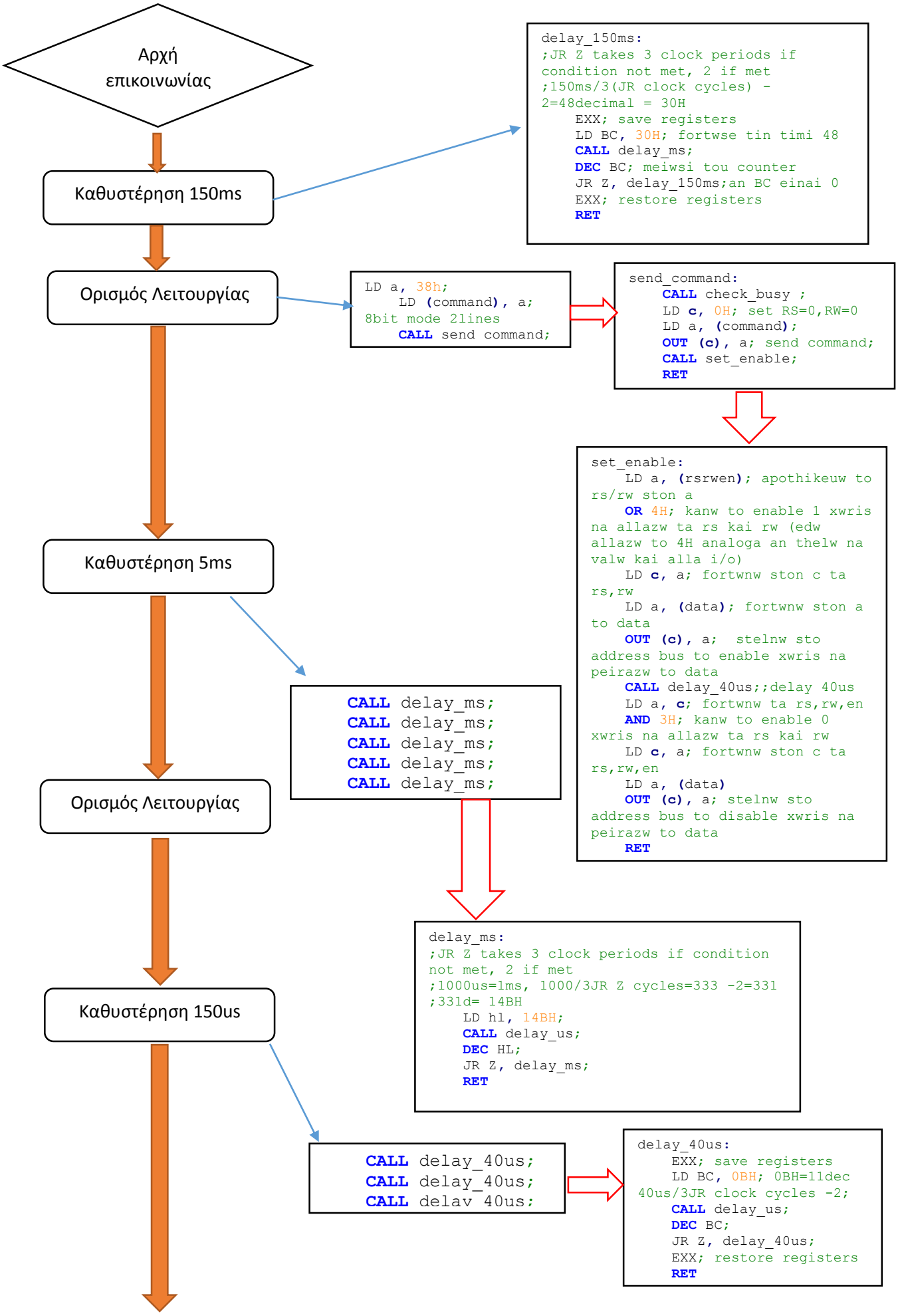
Για να χρησιμοποιήσει ο χρήστης την επιθυμητή λειτουργία αρκεί να εκτελέσει την εντολή CALL ακολουθούμενη από το μνημονικό της λειτουργίας όπως αυτές περιγράφονται παρακάτω.

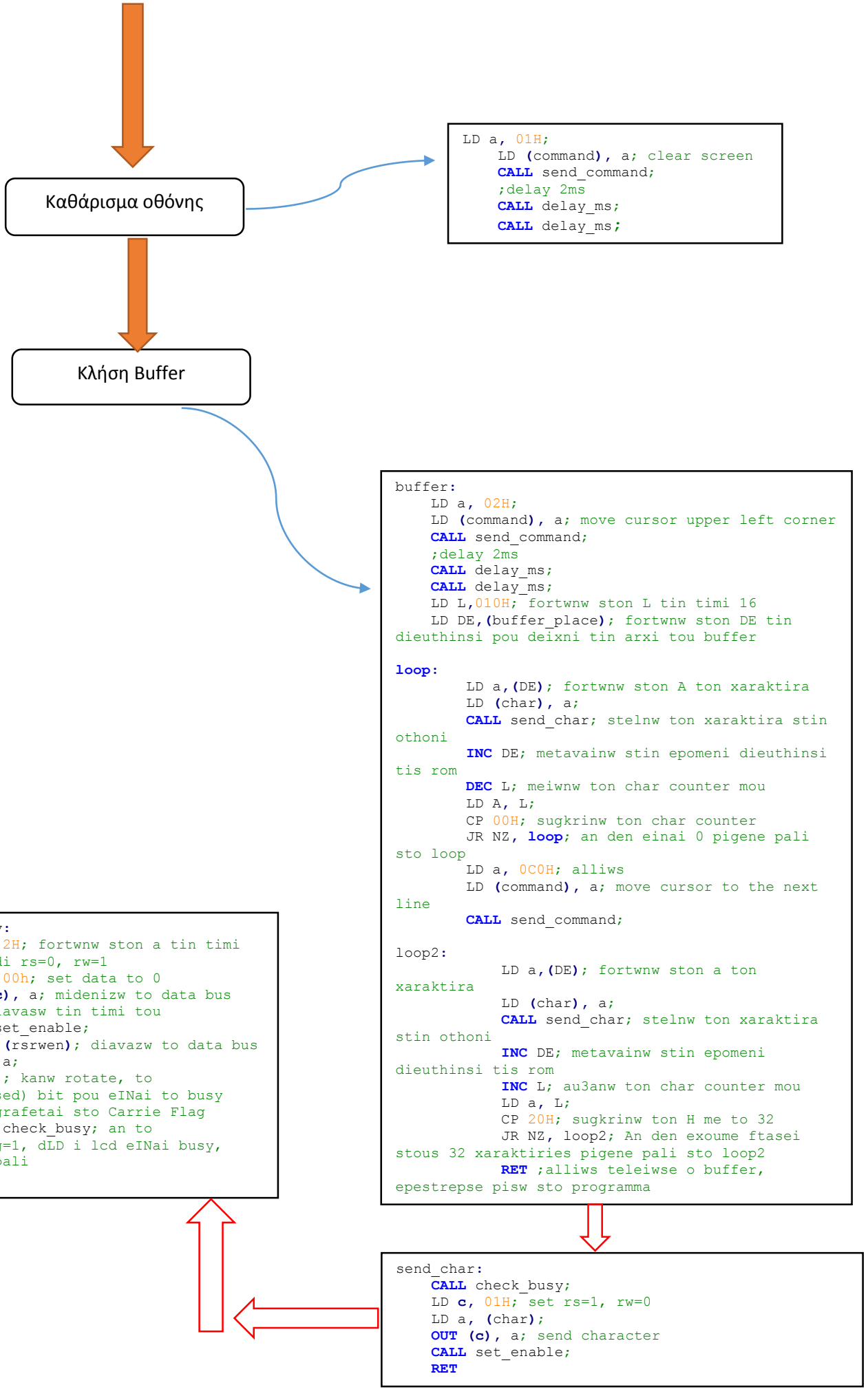
Π.χ.: CALL cursor_off;

- Ολίσθηση ολόκληρης της οθόνης μία θέση δεξιά:
  lcd_scroll_right
- Ολίσθηση ολόκληρης της οθόνης μία θέση αριστερά:
  lcd_scroll_left
- Επιστροφή του κέρσορα στην πάνω αριστερά θέση:
  home
- Ο κέρσορας αναβοσβήνει ολόκληρο το μπλοκ του χαρακτήρα:
  blink_char_cursor
- Ο κέρσορας αναβοσβήνει την υπογράμμιση του χαρακτήρα:
  blink_cursor
- Απενεργοποίηση κέρσορα:
  cursor_off
- Σβήσιμο της οθόνης χωρίς καθαρισμό των χαρακτήρων:
  blank_lcd
- Επαναφορά της οθόνης(με κρυφό κέρσορα):
  restore_lcd
- Μετακίνηση του κέρσορα μία θέση δεξιά:
  cursor_right
- Μετακίνηση του κέρσορα μία θέση αριστερά:
  cursor_left
- Εγγραφή στην οθόνη και παράλληλη ολίσθηση του κειμένου αριστερά:
  left_shift
- Καθαρισμός της οθόνης(διαγραφή όλων των χαρακτήρων:
  clear

ΕΠΕΞΗΓΗΣΗ ΡΟΗΣ ΚΩΔΙΚΑ

Παρακάτω ακολουθεί μια απλή εξήγηση για την δομή του κώδικα, όπως αυτός χρησιμοποιείται για την αρχικοποίηση της οθόνης και την χρήση του buffer που τυπώνει στην οθόνη 32 χαρακτήρες από 32 διαδοχικές θέσεις στην μνήμη.


Αρχή επικοινωνίας
Καθυστέρηση 150ms
Ορισμός Λειτουργίας
Καθυστέρηση 5ms
Ορισμός Λειτουργίας
Καθυστέρηση 150us
delay_150ms:
;JR Z takes 3 clock periods if condition not met, 2 if met
;150ms/3(JR clock cycles) - 2=48decimal = 30H
EXX; save registers
LD BC, 30H; fortwse tin timi 48
CALL delay_ms;
DEC BC; meiwsi tou counter
JR Z, delay_150ms;an BC einai 0
EXX; restore registers
RET
LD a, 38h;
LD (command), a;
8bit mode 2lines
CALL send command;
send_command:
CALL check_busy ;
LD c, 0H; set RS=0,RW=0
LD a, (command);
OUT (c), a; send command;
CALL set_enable;
RET
set_enable:
LD a, (rsrwen); apothikeuw to rs/rw ston a
OR 4H; kanw to enable 1 xwris na allazw ta rs kai rw (edw allazw to 4H analoga an thelw na valw kai alla i/o)
LD c, a; fortwnw ston c ta rs,rw
LD a, (data); fortwnw ston a to data
OUT (c), a; stelnw sto address bus to enable xwris na peirazw to data
CALL delay_40us;;delay 40us
LD a, c; fortwnw ta rs,rw,en
AND 3H; kanw to enable 0 xwris na allazw ta rs kai rw
LD c, a; fortwnw ston c ta rs,rw,en
LD a, (data)
OUT (c), a; stelnw sto address bus to disable xwris na peirazw to data
RET
CALL delay_ms;
CALL delay_ms;
CALL delay_ms;
CALL delay_ms;
CALL delay_ms;
delay_ms:
;JR Z takes 3 clock periods if condition not met, 2 if met
;1000us=1ms, 1000/3JR Z cycles=333 -2=331
;331d= 14BH
LD hl, 14BH;
CALL delay_us;
DEC HL;
JR Z, delay_ms;
RET
CALL delay_40us;
CALL delay_40us;
CALL delay_40us;
delay_40us:
EXX; save registers
LD BC, 0BH; 0BH=11dec 40us/3JR clock cycles -2;
CALL delay_us;
DEC BC;
JR Z, delay_40us;
EXX; restore registers
RET


Καθάρισμα οθόνης
Κλήση Buffer
LD a, 01H;
LD (command), a; clear screen
CALL send_command;
;delay 2ms
CALL delay_ms;
CALL delay_ms;
buffer:
LD a, 02H;
LD (command), a; move cursor upper left corner
CALL send_command;
;delay 2ms
CALL delay_ms;
CALL delay_ms;
LD L,010H; fortwnw ston L tin timi 16
LD DE,(buffer_place); fortwnw ston DE tin dieuthinsi pou deixni tin arxi tou buffer
loop:
LD a,(DE); fortwnw ston A ton xaraktira
LD (char), a;
CALL send_char; stelnw ton xaraktira stin othoni
INC DE; metavainw stin epomeni dieuthinsi tis rom
DEC L; meiwnw ton char counter mou
LD A, L;
CP 00H; sugkrinw ton char counter
JR NZ, loop; an den einai 0 pigene pali sto loop
LD a, 0C0H; alliws
LD (command), a; move cursor to the next line
CALL send_command;
loop2:
LD a,(DE); fortwnw ston a ton xaraktira
LD (char), a;
CALL send_char; stelnw ton xaraktira stin othoni
INC DE; metavainw stin epomeni dieuthinsi tis rom
INC L; au3anw ton char counter mou
LD a, L;
CP 20H; sugkrinw ton H me to 32
JR NZ, loop2; An den exoume ftasei stous 32 xaraktiries pigene pali sto loop2
RET ;alliws teleiwse o buffer, epestrepse pisw sto programma
send_char:
CALL check_busy;
LD c, 01H; set rs=1, rw=0
LD a, (char);
OUT (c), a; send character
CALL set_enable;
RET
check_busy:
LD c, 2H; fortwnw ston a tin timi 0b10 diladi rs=0, rw=1
LD a, 00h; set data to 0
OUT (c), a; midenizw to data bus wste na diavasw tin timi tou
CALL set_enable;
IN a, (rsrwen); diavazw to data bus
LD b, a;
RLC b ; kanw rotate, to 7o(zerobased) bit pou eINai to busy flag antigrafetai sto Carrie Flag
JP c, check_busy; an to CarrieFlag=1, dLD i lcd eINai busy, dokimase pali
RET

4. ΧΡΗΣΙΜΟΙ ΣΥΝΔΕΣΜΟΙ

Datasheet Zilog Z80:

http://www.z80.info/zip/z80cpu_um.pdf

Datasheet Crystafontz 2x16 lcd:

http://www.crystalfontz.com/products/document/906/CFAH1602BNYGJT_v1.0.pdf

Datasheet Hitachi HD44780:

http://lcd-linux.sourceforge.net/pdfdocs/hd44780.pdf

Lcd simulator:

http://www.geocities.com/dinceraydin/djlcdsim/djlcdsim.html

HD44780 controller wikipedia:

http://en.wikipedia.org/wiki/Hitachi_HD44780_LCD_controller

Παρόμοιες εργασίες:

http://www.geocities.com/dinceraydin/lcd/z80example.htm

http://www.zipplet.co.uk/index.php/content/electronics_z80

http://devster.monkeeh.com/z80.html

5.ΚΩΔΙΚΑΣ

Παρατίθεται ο κώδικας της διασύνδεσης του Z80 με την οθόνη:

```
; A0=RS, A1=RW, A2=EN


data          = 8000H
temp          = 8001H
command       = 8002H
rsrwen        = 8003H
char          = 8004H
buffer_place  = 8005H
new_var       = 8007H


    org 00H


set_enable:
    LD a, (rsrwen); apothikeuw to rs/rw ston a
    OR 4H; kanw to enable 1 xwris na allazw ta rs kai rw (edw allazw
to 4H analoga an thelw na valw kai alla i/o)
    LD c, a; fortwnw ston c ta rs,rw
    LD a, (data); fortwnw ston a to data
    OUT (c), a;  stelnw sto address bus to enable xwris na peirazw to
data
    CALL delay_40us;;delay 40us
    LD a, c; fortwnw ta rs,rw,en
    AND 3H; kanw to enable 0 xwris na allazw ta rs kai rw
    LD c, a; fortwnw ston c ta rs,rw,en
    LD a, (data)
    OUT (c), a; stelnw sto address bus to disable xwris na peirazw to
data
    RET

check_busy:
    LD c, 2H; fortwnw ston a tin timi 0b10 diladi rs=0, rw=1
    LD a, 00h; set data to 0
    OUT (c), a; midenizw to data bus wste na diavasw tin timi tou
    CALL set_enable;
    IN a, (rsrwen); diavazw to data bus
    LD b, a;
    RLC b ; kanw rotate, to 7o(zerobased) bit pou eINai to busy flag
antigrafetai sto Carrie Flag
    JP c, check_busy; an to CarrieFlag=1, dLD i lcd eINai busy,
dokimase pali
    RET

send_command:
    CALL check_busy ;
    LD c, 0H; set rs=0,rw=0
    LD a, (command);
    OUT (c), a; send command;
    CALL set_enable;
    RET

send_char:
    CALL check_busy;
    LD c, 01H; set rs1, rw=0
    LD a, (char);
    OUT (c), a; send char
    CALL set_enable;
    RET
```

```
init:
    CALL delay_150ms;delay 150ms
    LD a, 38h;
    LD (command), a; 8bit mode 2lines
    CALL send_command;
    ;delay 5ms
    CALL delay_ms;
    CALL delay_ms;
    CALL delay_ms;
    CALL delay_ms;
    CALL delay_ms;
    LD a, 38h;
    LD (command), a; 8bit mode 2lines
    CALL send_command;
    ;delay 150us
    CALL delay_40us;
    CALL delay_40us;
    CALL delay_40us;
    LD a, 01H;
    LD (command), a; clear screen
    CALL send_command;
    ;delay 2ms
    CALL delay_ms;
    CALL delay_ms;
RET

delay_us:
;NOP takes 1 clock period
;1 NOP @4mhz=0.25*10^-6= 250ns
; 4*250ns=1us
    NOP;
    NOP;
    NOP;
    NOP;
    RET

delay_ms:
;JR Z takes 3 clock periods if condition not met, 2 if met
;1000us=1ms, 1000/3JR Z cycles=333 -2=331
;331d= 14BH
    LD hl, 14BH;
    CALL delay_us;
    DEC HL;
    JR Z, delay_ms;
    RET

delay_150ms:
;JR Z takes 3 clock periods if condition not met, 2 if met
;150ms/3(JR clock cycles) -2=48decimal = 30H
    EXX; save registers
    LD BC, 30H; fortwse tin timi 48
    CALL delay_ms;
    DEC BC; meiwsi tou counter
    JR Z, delay_150ms; an o BC einai 0
    EXX; restore registers
    RET
```

```
delay_40us:
    EXX; save registers
    LD BC, 0BH; 0BH=11dec  40us/3JR clock cycles -2;
    CALL delay_us;
    DEC BC;
    JR Z, delay_40us;
    EXX; restore registers
    RET



;H lcd exei 16 theseis gia xaraktires ana grammi.
;O buffer fortwnei tin axiki dieuthinsi
;pou periexetai o prwtos xaraktiras
;kai stelnei tous xaraktires pou
;periexontai stis 32 sunexomenes theseis
;stin othoni.
;Epeidi oi theseis stin othoni den einai
;sinexomenes otan allazei i grammi
;o buffer eksasfalizei oti oi xaraktires
;tha metaferontai stin swsti thesi.
;Parallila otan gemisei i othoni
;fernei ton cursor stin panw aristera thesi
buffer:
    LD a, 02H;
    LD (command), a; move cursor upper left corner
    CALL send_command;
    ;delay 2ms
    CALL delay_ms;
    CALL delay_ms;
    LD L,010H; fortwnw ston L tin timi 16
    LD DE,(buffer_place); fortwnw ston DE tin dieuthinsi pou deixni
tin arxi tou buffer

loop:
        LD a,(DE); fortwnw ston A ton xaraktira
        LD (char), a;
        CALL send_char; stelnw ton xaraktira stin othoni
        INC DE; metavainw stin epomeni dieuthinsi tis rom
        DEC L; meiwnw ton char counter mou
        LD A, L;
        CP 00H; sugkrinw ton char counter
        JR NZ, loop; an den einai 0 pigene pali sto loop
        LD a, 0C0H; alliws
        LD (command), a; move cursor to the next line
        CALL send_command;

loop2:
            LD a,(DE); fortwnw ston a ton xaraktira
            LD (char), a;
            CALL send_char; stelnw ton xaraktira stin othoni
            INC DE; metavainw stin epomeni dieuthinsi tis rom
            INC L; au3anw ton char counter mou
            LD a, L;
            CP 20H; sugkrinw ton H me to 32
            JR NZ, loop2; An den exoume ftasei stous 32 xaraktiries
pigene pali sto loop2
            RET ;alliws teleiwse o buffer, epestrepse pisw sto
programma
```

```
;Some LCD funtions


lcd_scroll_right: ;scroll screen 1 block right
    LD a, 01EH;
    LD (command),a;
    CALL send_command;
    RET

lcd_scroll_left: ;scroll screen 1 block left
    LD a, 018H;
    LD (command),a;
    CALL send_command;
    RET

home: ;cursor returns to the top left position
    LD a, 02H;
    LD (command), a;
    CALL send_command;
    ;delay 2ms
    CALL delay_ms;
    CALL delay_ms;
    RET

blink_char_cursor: ;turns on blinking char cursor(whole block blinks)
    LD a, 0FH;
    LD (command),a;
    CALL send_command;
    RET

blink_cursor: ;turns on blinking underline only cursor
    LD a, 0EH;
    LD (command),a;
    CALL send_command;
    RET

cursor_off: ;turns cursor off
    LD a, 0CH;
    LD (command),a;
    CALL send_command;
    RET

blank_lcd: ;Blank the display (without clearing)
    LD a, 08H;
    LD (command),a;
    CALL send_command;
    RET

restore_lcd: ; Restore the display (with cursor hidden)
    LD a, 0CH;
    LD (command),a;
    CALL send_command;
    RET

cursor_right: ;moves cursor 1 block right
    LD a, 14H;
    LD (command),a;
    CALL send_command;
    RET

cursor_left: ; moves cursor 1 block left
```

```
    LD a, 10H;
    LD (command),a;
    CALL send_command;
    RET

left_shift: ;writing into lcd shifts text to the left
    LD a, 07H;
    LD (command),a;
    CALL send_command;
    RET

clear: ;clears the lcd
    LD a, 01H;
    LD (command),a;
    CALL send_command;
    ;delay 2ms
    CALL delay_ms;
    CALL delay_ms;
    RET
```