پاییز ۱۳۹۴

پرهام الوانی

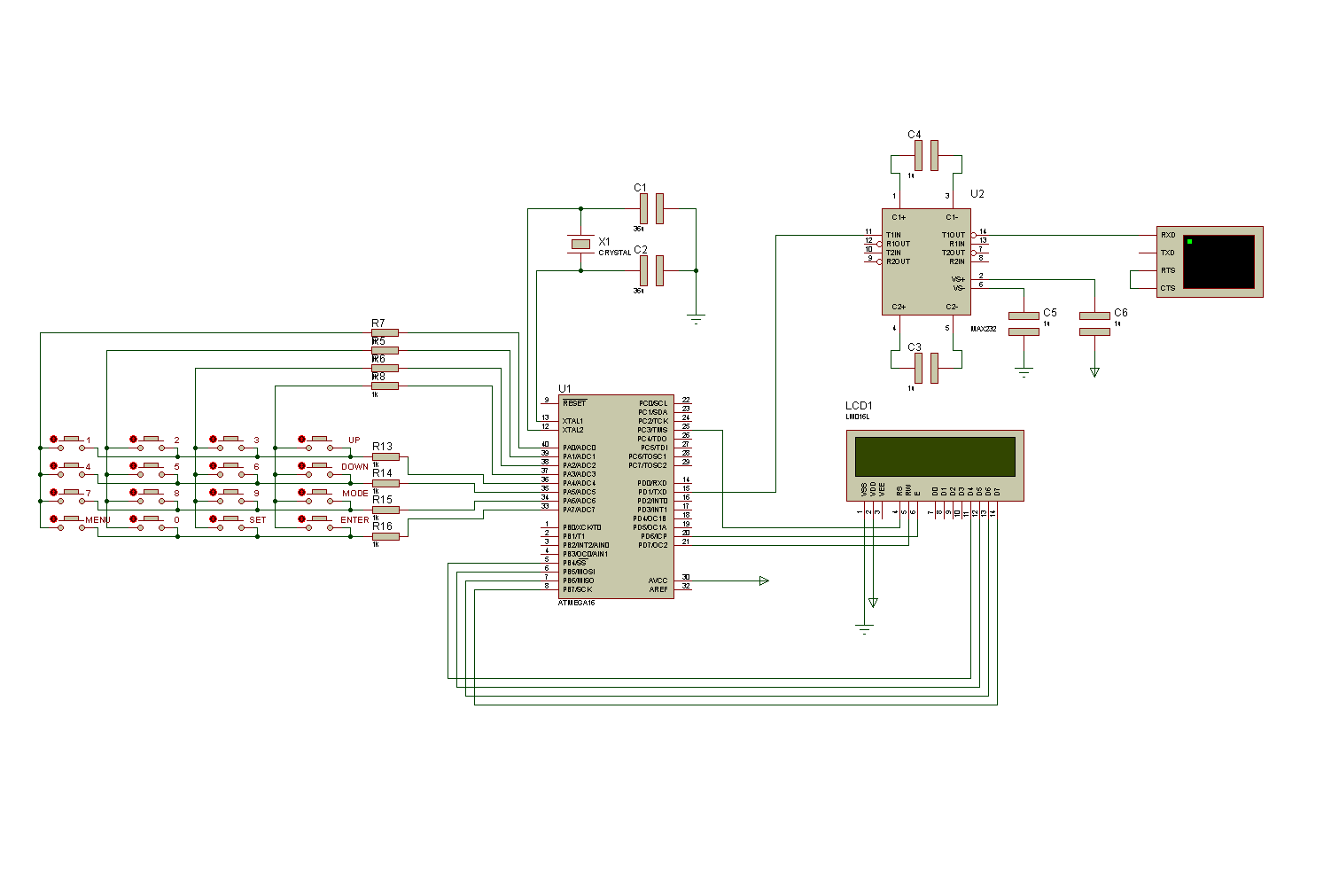
۹۲۳۱۰۵۸

پروژه پایانی

ریزپردازنده

# شمای کلی پروژه

هدف این پروژه طراحی یک کیبورد متصل به میکروکنترلر و نشان دادن ورودی گرفته شده از آن بر روی LED و فرستادن آن از طریق USART به کامپیوتر است.



Reset

در ابتدای برنامه با برنامه ریزی وقفه RESET تنظیمات اولیه مربوط به میکروکنترلر را انجام می‌دهیم.

.org $000

reset\_label:

jmp reset\_isr

reset\_isr:

cli

ldi r16 , LOW(RAMEND)

out SPL , r16

ldi r16 , HIGH(RAMEND)

out SPH , r16

; Set USART buffer pointers

ldi YL, LOW(buffer)

ldi YH, HIGH(buffer)

; PA0 - PA3 --> input + pullup

; PA4 - PA7 --> output

ldi r16, $F0

out DDRA, r16

in r16, SFIOR

andi r16, $FB

out SFIOR, r16

ldi r16, $0F

out PORTA, r16

; PB0 - PB6 --> output

; PB7 --> Input

ldi r16, $7F

out DDRB, r16

; PD6 - PD7 --> Output

ldi r16, (1 << PD6) | (1 << PD7)

out DDRD, r16

ldi r16, (1 << PD6)

out PORTD, r16

; Set USART baud rate to 2400kbps with 1Mhz clock

ldi r16, HIGH(25)

out UBRRH, r16

ldi r16, LOW(25)

out UBRRL,r16

; Set USART startup settings

; Stop bit = 2

; Parity = ODD

; Data bit = 8

ldi r24, (0 << UMSEL) | (1 << UCSZ1) | (1 << URSEL) | (1 << UPM1) | (0 << UPM0) | (0 << UCPOL) | (1 << UCSZ0) | (1 << USBS)

out UCSRC, r24

ldi r24, (0 << UCSZ2) | (1 << TXEN) | (0 << RXEN)

out UCSRB, r24

ldi r24, (0 << U2X) | (0 << MPCM)

out UCSRA, r24

; PC0 - PC3 --> Output

ldi r16, $0F

out DDRC, r16

call lcd\_function\_set

call lcd\_init

sei

jmp start

# Keypad

با طراحی صفحه کلید ماتریسی روی PORTA قابلیت استفاده از کلیدهای روی کیت آموزشی را ایجاد کردیم. پایه های PORTA به این صورت تنظیم شده‌اندکه 4 بیت پرارزش خروجی برای مشخص کردن سطر و 4 بیت کم ارزش ورودی برای مشخص کردن ستون کلید فشرده شده استفاده می‌شوند.

در تابعkey\_poll به روش polling چک می‌شود که آیا کلیدی فشار داده شده است یا خیر. با فشار داده شدن یک کلید تابع key\_find صدا زده میشود. سپس تابع delay صدا زده میشود که این delay برای جلوگیری از لغزش هنگام فشار دادن کلیدها قرار داده شده است. و بعد از پیدا شدن کلید تابع lcd\_write برای نوشتن روی LCD فراخوانی میشود.

; Keypad interrupt rutine, providing delay

; to keep key bauncing away :)

key\_poll:

wdr

call delay

in r16, PINA

ori r16, $F0

cpi r16, $FF

breq key\_poll

wdr

call key\_find

; Show in LCD

call lcd\_write

ldi r16, $0F

out PORTA, r16

st Y+, r0

mov r21, r0

cpi r21, '\n'

brne key\_poll\_ret

call usart\_send

call lcd\_clear

key\_poll\_ret:

ret

; Find key pressed on row keypad

; and put in r0;

key\_find:

ldi r17, $00

ldi r18, $00

ldi r19, $7F

key\_find\_loop1:

mov r16, r19

out PORTA, r16

nop

in r16, PINA

ldi r20, $04

key\_find\_loop2:

ror r16

brcc key\_find\_ret

dec r20

cpi r20, $00

brne key\_find\_loop2

inc r18

inc r18

inc r18

inc r18

sec

ror r19

inc r17

cpi r17, $04

brne key\_find\_loop1

key\_find\_ret:

mov r0, r20

add r0, r18

dec r0

mov r16, r0

cpi r16, $00

breq key\_find\_0

cpi r16, $01

breq key\_find\_1

cpi r16, $02

breq key\_find\_2

cpi r16, $03

breq key\_find\_3

cpi r16, $04

breq key\_find\_4

cpi r16, $05

breq key\_find\_5

cpi r16, $06

breq key\_find\_6

cpi r16, $07

breq key\_find\_7

cpi r16, $08

breq key\_find\_8

cpi r16, $09

breq key\_find\_9

cpi r16, $0A

breq key\_find\_10

cpi r16, $0B

breq key\_find\_11

cpi r16, $0C

breq key\_find\_12

cpi r16, $0D

breq key\_find\_13

cpi r16, $0E

breq key\_find\_14

cpi r16, $0F

breq key\_find\_15

key\_find\_0:

; ENTER

ldi r16, '\n'

mov r0, r16

ret

key\_find\_1:

; SET

ldi r16, 'S'

mov r0, r16

ret

key\_find\_2:

ldi r16, '0'

mov r0, r16

ret

key\_find\_3:

; MENU

ldi r16, 'M'

mov r0, r16

ret

key\_find\_4:

; MODE

ldi r16, 'm'

mov r0, r16

ret

key\_find\_5:

ldi r16, '9'

mov r0, r16

ret

key\_find\_6:

ldi r16, '8'

mov r0, r16

ret

key\_find\_7:

ldi r16, '7'

mov r0, r16

ret

key\_find\_8:

; DOWN

ldi r16, 'D'

mov r0, r16

ret

key\_find\_9:

ldi r16, '6'

mov r0, r16

ret

key\_find\_10:

ldi r16, '5'

mov r0, r16

ret

key\_find\_11:

ldi r16, '4'

mov r0, r16

ret

key\_find\_12:

; UP

ldi r16, 'U'

mov r0, r16

ret

key\_find\_13:

ldi r16, '3'

mov r0, r16

ret

key\_find\_14:

ldi r16, '2'

mov r0, r16

ret

key\_find\_15:

ldi r16, '1'

mov r0, r16

ret

; Create delay with repeating nop

delay:

ldi r18, $01

delay\_loop\_3:

ldi r17, $E0

delay\_loop\_2:

ldi r16, $FF

delay\_loop\_1:

wdr

dec r16

cpi r16, $00

brne delay\_loop\_1

dec r17

cpi r17, $00

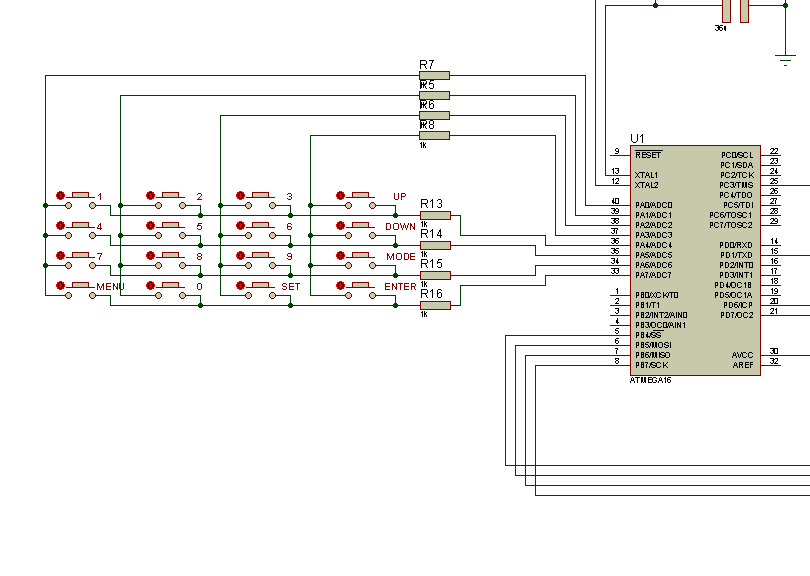
brne delay\_loop\_2

dec r18

cpi r18, $00

brne delay\_loop\_3

ret

****

# LCD

برای کار با LCD به چهار تابع اصلی که در ادامه آمده است نیاز داریم، این توابع همانطور که از اسمشان بر‌می‌آید برای انجام تنظیمات اولیه، نوشتن و پاک کردن صفحه نمایش می‌باشند.

; Function set of led

lcd\_function\_set:

; E = PD6 = 1

ldi r16, (1 << PD6)

out PORTD, r16

; PB0 - PB7 --> Output

ldi r16, $FF

out DDRB, r16

; Out HIGH($20)

ldi r16, $20

out PORTB, r16

; RS = PC3 = 0

; RW = PD7 = 0

; E = PD6 = 0

ldi r16, (0 << PC3)

out PORTC, r16

ldi r16, (0 << PD7) | (0 << PD6)

out PORTD, r16

; PB0 - PB6 --> output

; PB7 --> Input

ldi r16, $7F

out DDRB, r16

lcd\_function\_set\_busy:

; E = PD6 = 1

ldi r16, (1 << PD6)

out PORTD, r16

nop

; RS = PC3 = 0

; RW = PD7 = 1

; E = PD6 = 0

ldi r16, (0 << PC3)

out PORTC, r16

ldi r16, (1 << PD7) | (0 << PD6)

out PORTD, r16

; Get busy flag

in r16, PINB

mov r17, r16

; E = PD6 = 1

ldi r16, (1 << PD6)

out PORTD, r16

nop

; RS = PC3 = 0

; RW = PD7 = 1

; E = PD6 = 0

ldi r16, (0 << PC3)

out PORTC, r16

ldi r16, (1 << PD7) | (0 << PD6)

out PORTD, r16

; destroy chert flag :D

in r16, PINB

; Restore busy flag

mov r16, r17

ori r16, $7F

cpi r16, $FF

breq lcd\_function\_set\_busy

; E = PD6 = 1

ldi r16, (1 << PD6)

out PORTD, r16

; PB0 - PB7 --> Output

ldi r16, $FF

out DDRB, r16

; Out HIGH($20)

ldi r16, $20

out PORTB, r16

; RS = PC3 = 0

; RW = PD7 = 0

; E = PD6 = 0

ldi r16, (0 << PC3)

out PORTC, r16

ldi r16, (0 << PD7) | (0 << PD6)

out PORTD, r16

; E = PD6 = 1

ldi r16, (1 << PD6)

out PORTD, r16

; Out LOW($20)

ldi r16, $00

out PORTB, r16

; RS = PC3 = 0

; RW = PD7 = 0

; E = PD6 = 0

ldi r16, (0 << PC3)

out PORTC, r16

ldi r16, (0 << PD7) | (0 << PD6)

out PORTD, r16

; E = PD6 = 1

ldi r16, (1 << PD6)

out PORTD, r16

; PB0 - PB6 --> output

; PB7 --> Input

ldi r16, $7F

out DDRB, r16

ret

; Set dispaly, cursor on .. let's rock :)

lcd\_init:

lcd\_init\_busy:

; E = PD6 = 1

ldi r16, (1 << PD6)

out PORTD, r16

nop

; RS = PC3 = 0

; RW = PD7 = 1

; E = PD6 = 0

ldi r16, (0 << PC3)

out PORTC, r16

ldi r16, (1 << PD7) | (0 << PD6)

out PORTD, r16

; Get busy flag

in r16, PINB

mov r17, r16

; E = PD6 = 1

ldi r16, (1 << PD6)

out PORTD, r16

nop

; RS = PC3 = 0

; RW = PD7 = 1

; E = PD6 = 0

ldi r16, (0 << PC3)

out PORTC, r16

ldi r16, (1 << PD7) | (0 << PD6)

out PORTD, r16

; destroy chert flag :D

in r16, PINB

; Restore busy flag

mov r16, r17

ori r16, $7F

cpi r16, $FF

breq lcd\_init\_busy

; E = PD6 = 1

ldi r16, (1 << PD6)

out PORTD, r16

; PB0 - PB7 --> Output

ldi r16, $FF

out DDRB, r16

; Out HIGH(0x0F)

ldi r16, $00

out PORTB, r16

; RS = PC3 = 0

; RW = PD7 = 0

; E = PD6 = 0

ldi r16, (0 << PC3)

out PORTC, r16

ldi r16, (0 << PD7) | (0 << PD6)

out PORTD, r16

; E = PD6 = 1

ldi r16, (1 << PD6)

out PORTD, r16

nop

; Out LOW(0x0F)

ldi r16, $F0

out PORTB, r16

; RS = PC3 = 0

; RW = PD7 = 0

; E = PD6 = 0

ldi r16, (0 << PC3)

out PORTC, r16

ldi r16, (0 << PD7) | (0 << PD6)

out PORTD, r16

; E = PD6 = 1

ldi r16, (1 << PD6)

out PORTD, r16

; PB0 - PB6 --> output

; PB7 --> Input

ldi r16, $7F

out DDRB, r16

ret

; Clear display

lcd\_clear:

call min\_delay

; E = PD6 = 1

ldi r16, (1 << PD6)

out PORTD, r16

; PB0 - PB7 --> Output

ldi r16, $FF

out DDRB, r16

; Out HIGH(0x01)

ldi r16, $00

out PORTB, r16

; RS = PC3 = 0

; RW = PD7 = 0

; E = PD6 = 0

ldi r16, (0 << PC3)

out PORTC, r16

ldi r16, (0 << PD7) | (0 << PD6)

out PORTD, r16

; E = PD6 = 1

ldi r16, (1 << PD6)

out PORTD, r16

nop

; Out LOW(0x01)

ldi r16, $10

out PORTB, r16

; RS = PC3 = 0

; RW = PD7 = 0

; E = PD6 = 0

ldi r16, (0 << PC3)

out PORTC, r16

ldi r16, (0 << PD7) | (0 << PD6)

out PORTD, r16

; E = PD6 = 1

ldi r16, (1 << PD6)

out PORTD, r16

; PB0 - PB6 --> output

; PB7 --> Input

ldi r16, $7F

out DDRB, r16

ret

; Send one byte stored in r0 into LCD

lcd\_write:

call min\_delay

; E = PD6 = 1

ldi r16, (1 << PD6)

out PORTD, r16

; PB0 - PB7 --> Output

ldi r16, $FF

out DDRB, r16

; Out HIGH(r0)

mov r16, r0

out PORTB, r16

; RS = PC3 = 1

; RW = PD7 = 0

; E = PD6 = 0

ldi r16, (1 << PC3)

out PORTC, r16

ldi r16, (0 << PD7) | (0 << PD6)

out PORTD, r16

; E = PD6 = 1

ldi r16, (1 << PD6)

out PORTD, r16

nop

nop

; Out LOW(r0)

mov r16, r0

swap r16

out PORTB, r16

; RS = PC3 = 1

; RW = PD7 = 0

; E = PD6 = 0

ldi r16, (1 << PC3)

out PORTC, r16

ldi r16, (0 << PD7) | (0 << PD6)

out PORTD, r16

; E = PD6 = 1

ldi r16, (1 << PD6)

out PORTD, r16

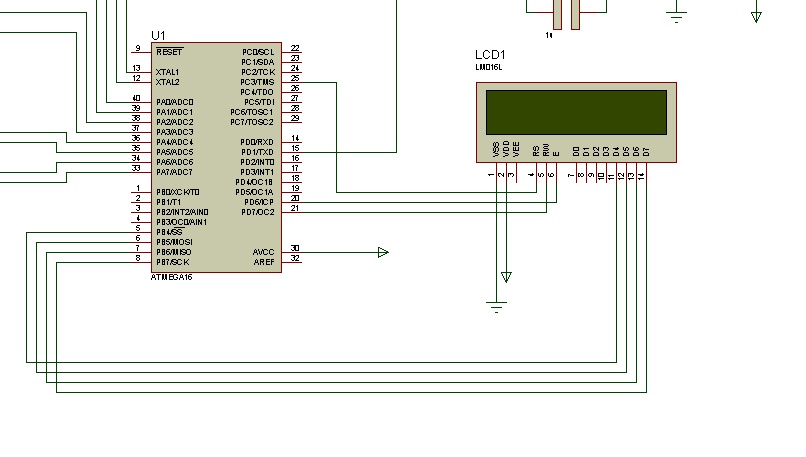
; PB0 - PB6 --> output

; PB7 --> Input

ldi r16, $7F

out DDRB, r16

ret

****

USART

**تنظمات اولیه:**

در این قسمت baud rate USART روی 2400bps با کلاک 1MHZ تنظیم شده است. در ادامه نیز تنظیمات مربوط به نحوه ارسال اعمال شده است.

; Set USART baud rate to 2400kbps with 1Mhz clock

ldi r16, HIGH(25)

out UBRRH, r16

ldi r16, LOW(25)

out UBRRL,r16

; Set USART startup settings

; Stop bit = 2

; Parity = ODD

; Data bit = 8

ldi r24, (0 << UMSEL) | (1 << UCSZ1) | (1 << URSEL) | (1 << UPM1) | (0 << UPM0) | (0 << UCPOL) | (1 << UCSZ0) | (1 << USBS)

out UCSRC, r24

ldi r24, (0 << UCSZ2) | (1 << TXEN) | (0 << RXEN)

out UCSRB, r24

ldi r24, (0 << U2X) | (0 << MPCM)

out UCSRA, r24

**ارسال:**

در این قسمت تابع usart\_send آورده شده است که با فشار دادن کلید Enter فراخوانی میشود و اطلاعات موجود در بافر را ارسال می‌کند.

; begin to Y

usart\_send:

wdr

ldi ZL, LOW(buffer)

ldi ZH, HIGH(buffer)

usart\_send\_try:

sbis UCSRA, UDRE

rjmp usart\_send\_try

ld r16, Z+

out UDR, r16

cp ZH, YH

brne usart\_send\_try

cp ZL, YL

brne usart\_send\_try

ldi YL, LOW(buffer)

ldi YH, HIGH(buffer)

ret

