نمایش موج سینوسیِ متصل به میکرو بر روی LCDی گرافیکی به زبانِ اسمبلی

تهیه کننده:**منصور اثنی‌عشری اصفهانی**
Mansoor_e1986@yahoo.com

چکیده مقاله: پروژه‌ای که ملاحظه می‌فرمایید نمایش موج ورودی به میکروکنترلر AVR بر روی LCDی گرافیکی می‌باشد. این موج می‌تواند هم برق DC و هم ac با هر شکل موجی باشد. همچنین برنامهٔ میکرو به زبان اسمبلی است. می‌توان از چنین برنامه‌هایی برای ساخت اسیلوسکوپ‌های دیجیتالی استفاده کرد.

مقدمه

پروژه‌ای که ملاحظه می‌فرمایید به این صورت است که، موج ورودی به میکرو را بر روی LCDی گرافیکی نمایش می‌دهد. این موج می‌تواند هم برق DC و هم ac با هر شکل موجی(سینوسی، مربعی، مثلثی و ...) باشد ولی محدودیت‌هایی نیز دارد. در زیر به خصوصیاتی که باید این موج داشته باشد اشاره می‌شود.

چون میکروی این پروژه از خانوادهٔ میکروکنترلرهایِ AVR و از نوع Atmega16 است، نمی‌توان به پایه‌های آن ولتاژی بیشتر از 5(v) وصل کرد پس یا باید از ولتاژ به کمک یک تقسیم ولتاژ نمونه گیری کرد و آن نمونه را به میکرو وصل کرد و یا اینکه ماکسیمم ولتاژ از 5(v) بیشتر نشود، لذا در عمل آفست ولتاژ سینوسی برابر 2.5(v) قرار داده شده است تا دامنه ولتاژ بتواند تا 2.49(v) افزایش یابد و آسیبی به میکرو نرسد.

محدودیتِ دیگری که در این موج ورودی باید اعمال کرد در مورد فرکانس آن است. چون این پروژه مخصوص فرکانسی در حدود 50(Hz) طراحی شده است، فرکانس‌های در محدودهٔ $0<f<200$ را به خوبی و تا کمی بیشتر از آن را تا حدی نمایش می‌دهد.

تذکر: چون در این میکرو فقط پرتِ A دارای مبدل ADC است، باید موج ورودی را به PA0 یعنی پایهٔ 40 وصل کنیم.

LCDی گرافیکی و نحوهٔ کارکرد آن

LCD ایی که در این پروژه قرار دارد به این صورت است که، دارای ۲چیپ است که یکی برای کنترلِ 64*64 پیکسلِ سمت چپ و دیگری برای 64*64 پیکسلِ راست است. هر 64*64 پیکسل نیز به این صورت است که، دارای 64 ستون(یعنی محور x) و در هر ستون 8 تکهٔ 8 بیتی(یعنی محور y) می‌باشد.

ما به کمک برنامه نویسی می‌توانیم 8 بیت 8 بیت رویِ LCD پیکسل‌ها را روشن یا خاموش کنیم. پس برای روشن شدن یک نقطه باید موقعیت در محور x و سپس ابتدای آن 8بیتِ مورد نظر را انتخاب کنیم و دیتای خود را (که 8بیتی است را) بر روی LCDبفرستیم. پس برای محور x، 64 حالت و برای محور y، 8 حالت داریم به علاوهٔ آن 8بیتِ دیتا. در ضمن برای ارسال اطلاعات نمایشی و یا اطلاعات پیکربندی باید نوع اطلاعات را به کمک پایه Rs مشخص کنیم. باید به این نکته نیز توجه داشت که هر کدام از این مراحل تاخیرِ مربوط به خود را دارد و باید در برنامه لحاظ کرد.

تذکر: باید در ابتدایِ کار LCD را روشن کرد.

برای انجام هر کدام از این کارها باید کد خاصی را آن هم بصورت اطلاعات پیکر بندی به LCD ارسال کرد، مثلاً در کدِ 0b0011111a، اگر a، 1 باشد LCD روشن و اگر 0 باشد خاموش می‌شود، یا در کدِ 0b01abcdef، به کمکِ آن 6بیت موقعیت روی محور x مشخص می‌شود.

به عنوانِ مثال برنامهٔ زیر موقعیت روی محور x را مشخص می‌کند:

```
ldi r20,0b01000000      ;baraye
mogheiyate x
out portd,r20
nop     ;yek takhir kootah
cbi portb,0             ;E=0
call delay ; yek takhir toolani
sbi portb,0             ;E=1
```

پایهٔ PB0 از میکرو که به پایهٔ E از LCD متصل است به عنوانِ کلاکِ LCD عمل می‌کند، که با لبهٔ پایین رونده کار می‌کند. در ضمن باید جایی قبل از نوشتنِ این برنامه چیپِ چپ یا راست را انتخاب کرده باشیم، همینطور Rs را صفر، تا مشخص شود که اطلاعات پیکربندی است و نیز W/R را صفر کنیم تا اطلاعات نوشتنی باشد نه خواندنی.

الگوریتم کلی برنامه

در این برنامه که به زبان اسمبلی نوشته شده است، بعد از پیکربندی‌های لازم و انتخابِ فرکانس مبدلِ ADC، همچنین کانال صفر به عنوانِ ورودی، مقدارِ ولتاژ را (به صورتِ 10بیتی) می‌خواند و پس از تبدیلاتِ لازم آن را بر رویِ LCD، در مکانِ مناسبش نشان می‌دهد، سپس دوباره ولتاژ را می‌خواند و مثلِ حالتِ قبل آن را در محلِ مناسبش نمایش می‌دهد و این کار را 128 بار تکرار می‌کند.

ولی مشکلی که در اینجا وجود دارد این است که نوشتن بر رویِ LCD تاخیرِ زیادی دارد و در خواندنِ موجِ ورودی خطایِ زیادی بوجود می‌آورد.(یعنی فرکانس‌هایِ بالا را نمی‌تواند بخواند، اگرچه فرکانس‌هایِ پایین را نیز درست نمی‌خواند.) برای رفعِ این مشکل بعد از هر بار خواندنِ ولتاژ، آن را سریعاً بر روی خانه‌ای از حافظهٔ RAM نوشته‌ایم و در پایانِ خواندنِ هر 128بار، مقادیر را از حافظهٔ RAM خوانده و بعد از تبدیلاتِ‌لازم آن را بر روی LCD نمایش داده‌ایم.

چون از اول به ترتیب مقادیر را خوانده‌ایم پس در محورِ x نیز به ترتیب پیش می‌رویم و پس از مشخص کردنِ موقعیتِ y (که در قسمتِ " الگوریتم تبدیل کد ۱۰بیتی متناظر با ولتاژ، به کدی برای LCDی

گرافیکی" توضیح داده شده است) نقطهٔ مورد نظر را روشن می‌کنیم.
تذکر: باید توجه داشت که رویِ محورِ x پس از 64خانه پیشروی باید چیپِ بعدی را انتخاب کرد و دوباره رویِ آن از 0 تا 63خانه پیش رفت.
مشخص کردنِ موقعیتِ y در قسمتِ بعدی توضیح داده شده است.

الگوریتم تبدیل کد ۱۰بیتی متناظر با ولتاژ، به کدی برای LCDی گرافیکی:

منظور از این قسمت آن تبدیلاتی است که موقعیتِ محورِ y را تعیین می‌کند.
ما رویِ محورِ y ، 8قسمت داریم، پس ابتدا مشخص می‌کنیم که ولتاژِ موردِ نظرِ ما در در کدام یک از این 8قسمت باید قرار گیرد، که این کار به سادگی با چند مقایسه طبقِ برنامهٔ زیر انجام می‌گیرد:

```
ldi r18,0xbf
ldi r25,x+
ldi r24,x+
dobare_tn1:
cpi r25,0x00
brne k1
cpi r24,0x7f
brlo door_tn
k1:
sbiw r25:r24,0x32
sbiw r25:r24,0x32
sbiw r25:r24,0x1b
dec r18
jmp dobare_tn1
door_tn:
```

عددِ متناظر با ولتاژ را از حافظهٔ RAM می‌خواند و با 127 مقایسه می‌کند، اگر بزرگتر بود، به موقعیتِ x را که به پایین‌ترین قسمت(از آن 8 قسمت) اشاره می‌کند یکی اضافه می‌کند و از عددِ 10بیتی، 127 را کم می‌کند. آنقدر اینکار را انجام می‌دهد تا عدد کمتر از 127 شود.
عددِ 127 از آنجایی آمده است که بزرگترین عددی که می‌خواند 1024 است که بر 8 تقسیم شده است.($1024/8=128$ پس یعنی 0 تا 127)
به این دلیل در برنامه برایِ افزایشِ موقعیتِ x از Dec استفاده شده که عددِ 111 به پایین‌ترین خانه اشاره دارد و هر بار یکی از آن کم کنیم ، به یک خانهٔ بالاتر اشاره می‌کند.
در پایانِ این بلوکِ برنامه باید این موقعیت را به LCD ارسال کرد.
بعد از تعیینِ این موقعیت باید مشخص شود که این عدد (،که کوچک‌تر از 127 است) کدام یکی از خانه‌هایِ آن دیتایِ 8 بیتی را باید روشن کند، که این کار نیز مثل قبل به سادگی با چند مقایسه قابل انجام است.

توابع برنامه

۱- $pekarbandi\_adc$ ، در این تابع مبدلِ ADC پیکربندی شده است. خصوصیاتی از قبیلِ انتخاب کانالِ صفر برای موجِ ورودی، تقسیم فرکانسِ 32 برای مبدل، انتخاب $Aref$ به عنوانِ ولتاژِ مرجع و انتخابِ مدِ $Single\ \ Ended$ ، به میکرو نسبت داده شده است.
۲- $tabdil\_adc\_RAM$ ، در این تابع پس از خواندنِ ولتاژ به صورتِ 10بیتی ، اعداد را

از خانهٔ *100* تا *256*ِ حافظهٔ *RAM* ذخیره می‌کند.

۳- *pekarbandi_rooshan_lcd* ، در این تابع پیکربندی‌های *LCD* و همچنین روشن شدن *LCD* انجام شده است. توضیحاتِ مربوط به این تابع در قسمتِ «***LCD*ی گرافیکی و نحوهٔ کارکرد آن**» داده شده است.

۴- *mogheiyat_y* ، در این قسمتِ برنامه اشاره‌گر (*Cursor*) را در اولین خانهٔ محورِ *x* انتقال می‌دهد، تا بعداً به ترتیب جلو برود و اعدادی که آن‌ها نیز به ترتیب ذخیره شده‌اند را در موقعیتِ خودشان نمایش دهد.

۵- *testi* ، این تابع که از برنامه حذف شده است برای تستِ سه تابعِ بعدی نوشته شده بوده است و به این صورت عمل می‌کرده که خانه‌های موردِ نظر از حافظهٔ *RAM* را با اعدادی منظم پر می‌کند تا در سه تابعِ بعدی بازخوانی شوند و یک خطِ مورب رویِ صفحه نمایش دهد.

۶- *tabdilat_namayesh* و *mogheiyat_y_new* و *portnamayeshi*، این سه تابع مهم‌ترین بخشِ این پروژه هستند و همان‌طور که در قسمت « الگوریتم تبدیل کد ۱۰بیتی متناظر با ولتاژ، به کدی برای *LCD*ی گرافیکی» به اندازهٔ کافی توضیح داده شد، وظیفهٔ تبدیلِ کدِ *10*بیتی متناظر با ولتاژ را به نقاطی بر روی صفحهٔ *LCD* بر عهده دارند. همانطور که وضیح داده شده است، الگوریتمِ این قسمت تا حدودی ساده و بسیار کاربردی است.

۷- *delay* ، این تابع تأخیرِ لازم برای ارسالِ اطلاعات به *LCD* را فراهم می‌کند که در برنامه بیشتر از صد بار فراخوانی می‌شود.

متنِ برنامه

```
;        In The Name of GOD
; Design By Mansoor Esnaashari
;mobile:     09357914208
; e-mail: mansoor_e1986@yahoo.com
;---------------------------------------------
.include "m16def.inc"
.org 0
 jmp Reset
;******** Reset **********(r22)
Reset:
 ldi  R22,low(0x45F) ;taine mahale poshte
 out  SPL,R22
 ldi  R22,high(0x45F)
 out  SPH,R22
 ldi r22,(1<<ivce)   ;gharar dadan
Interrupt_ha dar ghesmate Application az
hafezeye flash
 out gicr,r22
 ldi r22,(0<<ivsel)|(0<<ivce)
 out gicr,r22

 ser r22 ;port D,B ra khorooji tarif kardam
 out ddrd,r22
 out ddrb,r22
 clr r22    ;port A ra voroodi tarif kardam
 out ddra,r22
 sbi portb,5       ;payeye Reset=1
 clr r21
call pekarbandi_adc
call tabdil_adc_RAM
call pekarbandi_rooshan_lcd
call mogheiyat_y
;call testi
 ldi r26,0x00 ;x=0x100
 ldi r27,0x01
a:
 call tabdilat_namayesh
 call mogheiyat_y_new
 cpi r21,0x02
brne a
nothing: jmp nothing

;******* pekarbandi_adc *******(r22)
pekarbandi_adc:
```

```
 cbi portb,3;cs1=0
 sbi portb,0;e=1     ;chon ba labe pain
ravandeye E kar mikonad
 ldi r22,0b00111111  ;roshan kardan
safheye lcd
 out portd,r22
 nop                 ;yek takhir kootah
 cbi portb,0;e=0
 call delay
 sbi portb,0;e=1
 ldi r22,0b11000000
 out portd,r22
 nop                 ;yek takhir kootah
 cbi portb,0;e=0
 call delay
 sbi portb,0;e=1
ret
;------------------------------
;**** mogheiyat_y ***
mogheiyat_y:
 ldi r20,0b01000000  ;baraye mogheiyate
y
 out portd,r20
 nop                 ;yek takhir kootah
 cbi portb,0;e=0
 call delay
 sbi portb,0;e=1
ret
;---------------------------------
;***** tabdilat_namayesh ******
tabdilat_namayesh:
ldi r18,0xbf
 ld r25,x+
 ld r24,x+
dobare_tn1:
 cpi r25,0x00
 brne k1
 cpi r24,0x7f
 brlo door_tn
k1:
 sbiw r25:r24,0x32
 sbiw r25:r24,0x32
 sbiw r25:r24,0x1b
 dec r18
jmp dobare_tn1
door_tn:
 out portd,r18       ;taeene mogheiyate x be
komake display start line
 nop          ;yek takhir kootah
 cbi portb,0;e=0
 call delay
 sbi portb,0;e=1
call portnamayeshi
ret
;------------------------------
r22,(1<<aden)|(0<<adate)|(1<<adps2)|(
0<<adps1)|(1<<adps0)
 out adcsra,r22
ret
;------------------------------
;*** tabdil_adc_RAM ***
***(r22,r24,r25,r26,r27)
tabdil_adc_RAM:
 ldi r26,0x00 ;x=0x100
 ldi r27,0x01
 ldi r22,0x80
 dobare_tar:
 dec r22

 sbi adcsra,adsc
  bala: sbic adcsra,adsc ;sabr mikone ta
amaliate tabdil(ADC) tamam beshe
 jmp bala

 in r24,adcl
 in r25,adch
 st x+,r25
 st x+,r24
 cpi r22,0x00
 breq bad_tar
 jmp dobare_tar
 bad_tar:
ret
;---------------------------------
;*** pekarbandi_rooshan_lcd ***(r22)
pekarbandi_rooshan_lcd:
 cbi portb,2;Rs=0    ;etelaate peykar bandi
 cbi portb,1;R/W=0   ;baraye neveshtan
bar rooye LCD
sbi portb,3;cs2=0   ;entekhabe chip.
baraye neveshtane samte rast ya chap
 cbi portb,4;cs1=1
 sbi portb,0;e=1     ;chon ba labe pain
ravandeye E kar mikonad
 ldi r22,0b00111111  ;roshan kardan
safheye lcd
 out portd,r22
 nop             ;yek takhir kootah
 cbi portb,0;e=0
 call delay
 sbi portb,0;e=1
 ldi r22,0b11000000
 out portd,r22
 nop                 ;yek takhir kootah
 cbi portb,0;e=0
 call delay
 sbi portb,0;e=1
sbi portb,4;cs2=1   ;entekhabe chip.
baraye neveshtane samte rast ya chap
```

```
;**** portnamayeshi ****(r18,r24,r25)
portnamayeshi:
 ldi r18,0b01111111
 cpi r24,0x0f
 brlo door2

 ldi r18,0b10111111
 cpi r24,0x1f
 brlo door2
 ldi r18,0b11011111
 cpi r24,0x2f
 brlo door2
 ldi r18,0b11101111
 cpi r24,0x3f
 brlo door2
 ldi r18,0b11110111
 cpi r24,0x4f
 brlo door2
 ldi r18,0b11111011
 cpi r24,0x5f
 brlo door2
 ldi r18,0b11111101
 cpi r24,0x6f
 brlo door2
 ldi r18,0b11111110
door2:
 sbi portb,2;Rs=1    ;etelaati ke gharar ast
namayesh dade shavad
 out portd,r18
 nop
 cbi portb,0;e=0
 call delay
 sbi portb,0;e=1
 call delay
 cbi portb,2;Rs=0    ;etelaate peykar bandi
 nop
ret
;------------------------------
;*** mogheiyat_y_new **(r20,dar tabe
delay r22)
mogheiyat_y_new:
 inc r20
 out portd,r20
 nop                ;yek takhir kootah
 cbi portb,0;e=0
 call delay
 sbi portb,0;e=1

 cpi r20,0x80
 brne payan_myn
  ldi r20,0b01000000;baraye mogheiyate y
  sbi portb,3;cs2=0   ;entekhabe chip.
baraye neveshtane samte rast ya chap
  cbi portb,4;cs1=1
  inc r21
payan_myn:
ret
;--------------------------------
;********* testi ********
testi:
 ldi r26,0x00 ;x=0x100
 ldi r27,0x01

ldi r22,0x80
clr r24
clr r25
dobare_tar11:
 dec r22
adiw r25:r24,0x08

 st x+,r25
 st x+,r24

 cpi r22,0x00
 breq bad_tar11
 jmp dobare_tar11
 bad_tar11:
ret
;-------------------------------

;********** delay *******(r22)
delay:
 ldi r22,0xa0
dec1: dec r22
 cpi r22,0
 brne dec1
ret
;-----------------------------------
;Design By Mansoor Esnaashari
;e-mail: mansoor_e1986@yahoo.com
```

تهیه کننده:

منصور اثنی‌عشری

e-mail: ***mansoor_e1986@yahoo.com***

با تشکر فراوان از آقای علی فلاح که من را در ساخت این پروژه یاری رسانید.