دانشگاه آزاد اسلامی
واحد جنوب

نام پروژه : فانکشن ژنراتور کنترل شونده با میکرو کنترولر

نام استاد :جناب آقای مهندس فرخی

نام دانشجو :نساءبگم اصلانی

رشته تحصیلی :برق و الکترونیک

شماره دانشجویی:8112740494

تاریخ تحویل پروژه: 84/06/10

فهرست مطالب

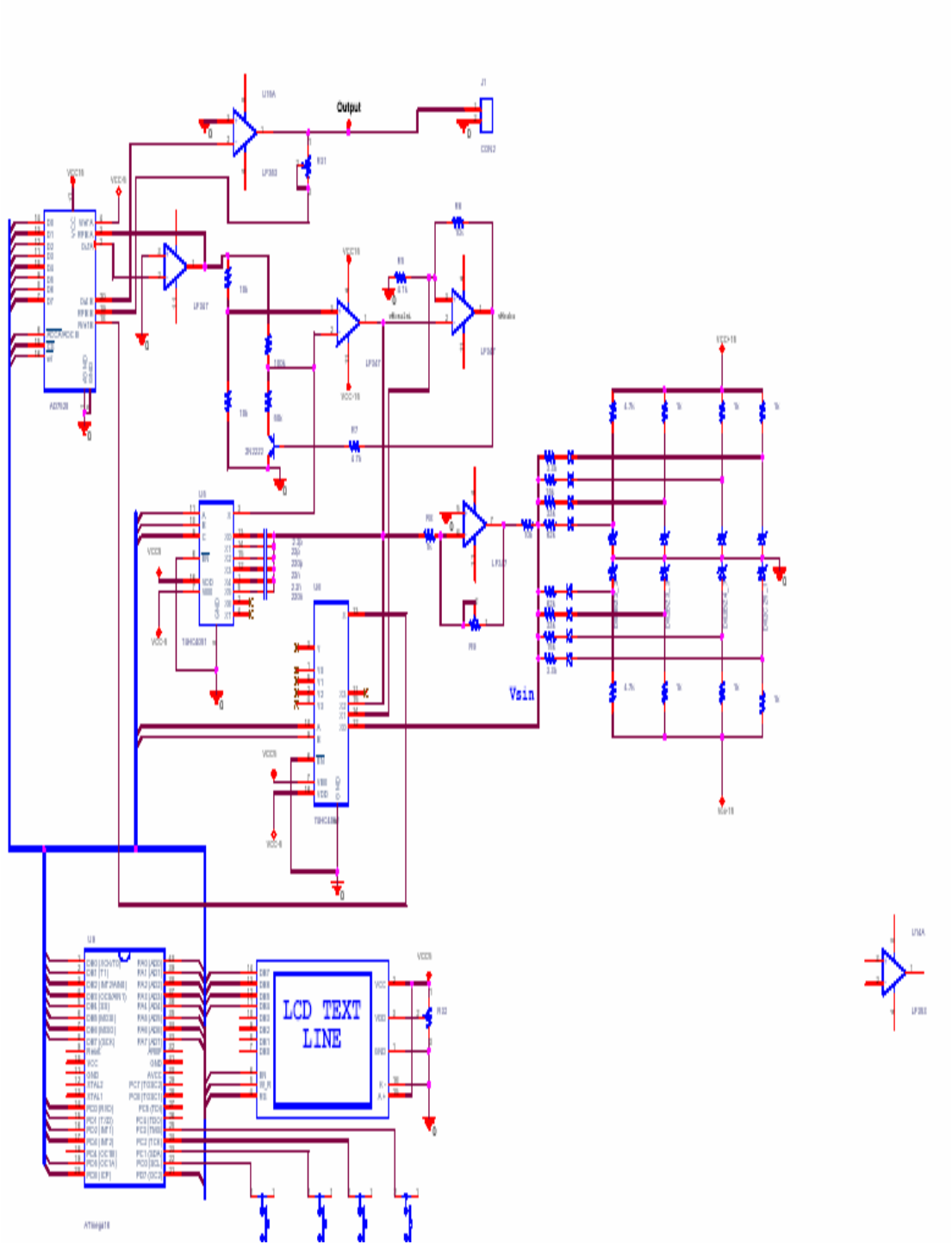
عنوان **صفحه**

مقدمه

سیگنال ژنراتور(مولد پالس) وسیله ای است برای تولید انواع موجهای سینوسی، مربّعی و مثلثی که معمولا در در آزمایشگاههای الکترونیکی به عنوان منبع سیگنال برای مدارهای الکترونیکی ازآن استفاده می کنند .با توجه به عنوان پروژه ،کنترل این مدار به وسیله یک میکروکنترولر که واسط بین کاربر و سیستم می باشد صورت میگیرد.

چکیده مطالب:

در این پروژه از آی سی های مولد این سه پالس استفاده نشده است و میبایست مدار داخلی این آی سی ها شبیه سازی می شد. بدین منظوراز آمپ امپها برای تولید امواج مربعی و مثلثی و از یک مدارشامل مقاومت و دیودها برای تولید موج مثلثی استفاده شده است که کنترل دامنه و فرکانس و نوع موج بوسیله یک میکرو صورت میگیرد. در فصل اول مشخصات و خلاصه ای از مدار و قطعات استفاده شده و نحوه و مدار مولد پالس مربعی ومثلثی و پالس سینوسی و محاسبات مدار و نحوه کنترل مدار بوسیله میکرو مورد نظر آورده شده است و در فصل دوم فلوچارت برنامه و برنامه میکرو که به زبان C نوشته شده و نتیجه پروژه تهیه شده و در آخر پروژه ، DATA SHEET قطعات استفاده شده آورده شده است.


Output
LCD TEXT
LINE
Vsin

فصل اول:

میکرو استفاده شده وسیله ای برای کنترل و تنظیم نوع خروجی ،فرکانس وآفست و ...می باشد.

مشخصات و محدوه این مدار:

انواع موج خروجی : مثلثی ، مربعی، سینوسی

محدوده دامنه : ۱۵ الی ۱۵-

محدوده فرکانس: 50HZ-30KHZ

خلاصه ای از مدار

در این سیستم از قطعات زیر استفاده شده.

میکرو کنترولر سری AVR (ATMega16L) برای کنترل سیستم

یک عدد LCDدو در شانزده متنی برای نمایش خروجی با کاربر

چهار عدد میکرو سویچ برای کنترل سیستم

op ampبرای ایجاد موج مثلثی و مربعی

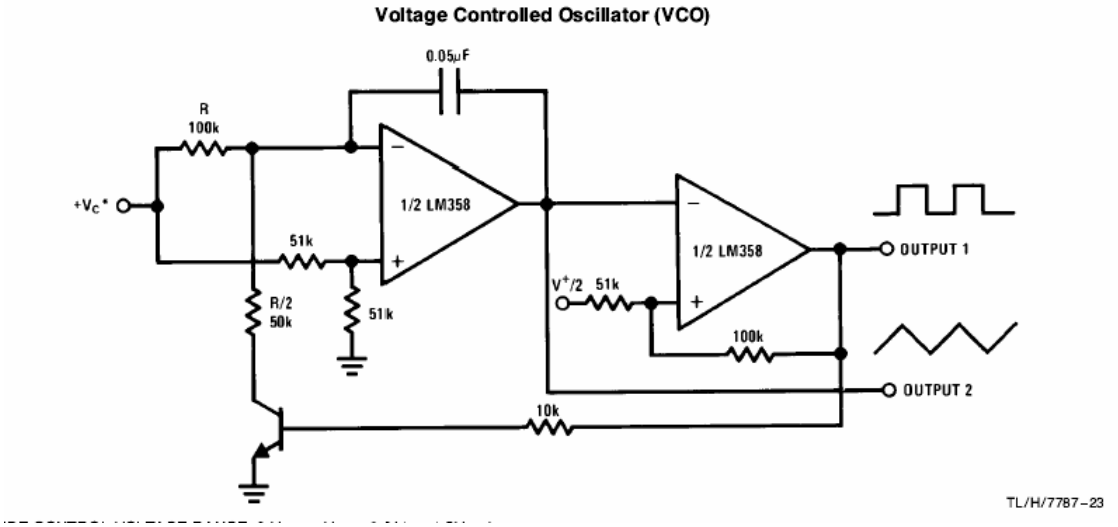
چند عدد دیود زنر و 1N4148و مقاومت برای ایجاد موج سینوسی

آی سی 4051و 4052و AD7523برای کنترل فرکانس ،گین و نوع خروجی

LCDوچند عدد کلید برای نمایش اطلاعات و نیز تغییر امکانات

برای تغذیه از رگولاتور مثبت ومنفی 15و 5ولت استفاده شده است

ایجاد موج مثلثی و مربعی



در اینجا برای ایجاد موج مثلثی از یک انتگرال گیر با آپ امپ استفاده کرده ایم با توجه به شکل فوق

....آپ امپ سمت راست این انتگرال گیر می باشد در طرف چپ مدار از یک مدار اشمیت تریگر

استفاده شده است با بالا رفتن ولتاژ انتگرالگیر این اشمیت تریگر سویچ میکند سپس با فعال کردن ترانزیستور مقاومت R/2فعال شده وباعث میشود که مدار انتگرال گیر به صورت معکوس عمل کرده و ایجاد یک رمپ منفی میکند .به این ترتیب از خروجی آپ امب اول موج مثلثی و از خروجی آپ امپ دوم مربعی گرفته میشود.

محاسبات مدار

در صورتی که ترانزیستور غیر فعال باشد ولتاژ سر پایه منفی برابر Vc/2میباشد و پس جریان عبوری از Rبرابر

$$(Vc - Vc/2) /R = Vc/(2R)$$

و با شارژ شدن خازن و فرمول آن ولتاژ خروجی برابر

$$Vo = 1/C(Vc/2R)t+V0 = (Vc/2RC)t$$

از آن طرف ولتاژ اشمیت ترگر برابر با توجه به مقادیر داخل نقشه و نیز زمین بودن V +برابر 1/3ولتاژ تغذیه می باشد .

زمانی که این ولتاز نیاز دارد تا به ولتاژ ماکزیمم وسپس به حالت اول برسد چهار برابر میباشد در این صورت فرکانس (1/t)به صورت زیر در می آید

$f = 1/t = 4 * (Vc / Vpp) * (1/2RC) = (Vc/Vpp) * (2/RC)$

$Vpp = 1\backslash 3\ Vc$

برای تغییرات در این فرکانس میتوان باکمک تغییر در مقدار Vcویا خازن مدار پرداخت

در اینجا برای کنترل فرکانس در مقادیر کم از Vcاستفاده شده است .برای این منظور از کنترل

دیجیتال یکی از دو کانال مبدل آنالوگ به دیجیتال آسی AD758استفاده میکنیم .

این ای سی دارای دو مبدل آنالوگ به دیجیتال میباشد که یکی از آن برای این منظور ودیگری در

جای دیگر بدان پرداخته میشود .

و برای تغییرات بالا خازن را می بایست تغییر داد که در اینجا از آنالوگ سویچ 4051 که یک

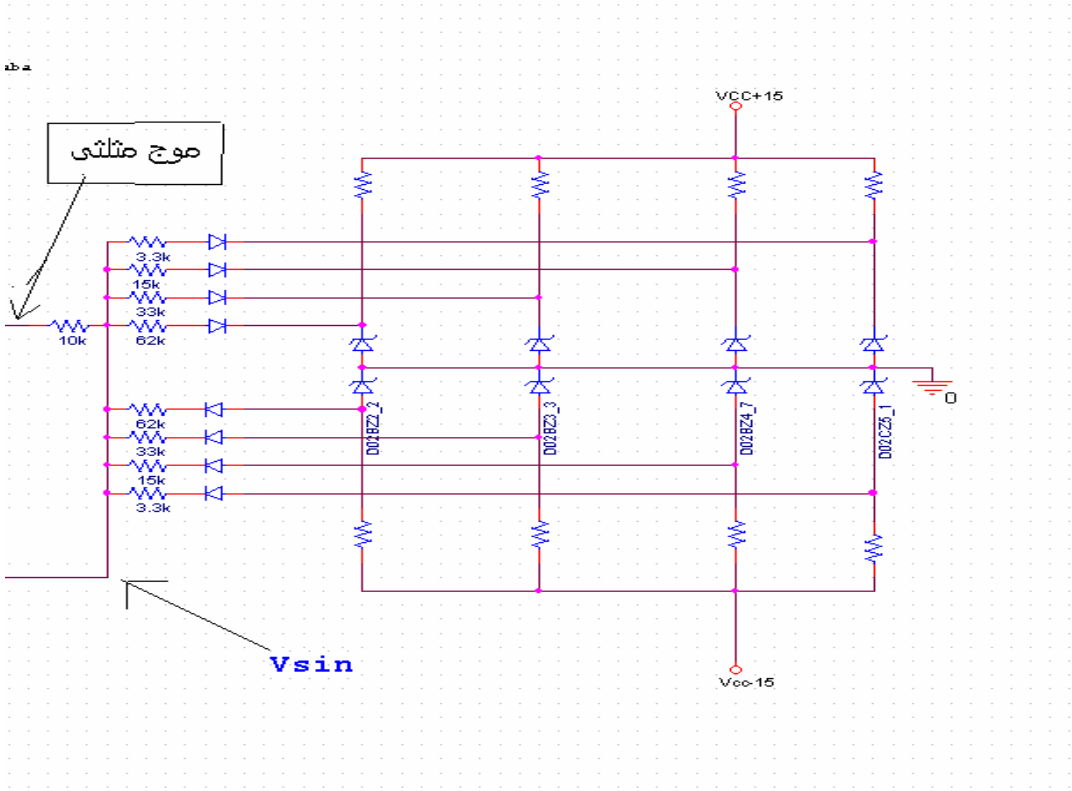
دیکدر یک به هشت میباشد استفاده میکنیم .

مقادیر خازنهای استفاده شده به صورت زیر میباشد.

2.2p, 22p, 220p , 2.2n, 22n ,220n

موج سینوسی

برای ایجاد موج سینوسی از چند منبع ولتاژ که به وسیله دیود زنر ایجاد شده است وچند مقاومت خاص از موج مثلثی تولید میگردد .(مدار شکل زیر)



این مدار موج مثلثی به شکل موج تقریبا سینوسی تبدیل میکند .منابع تغذیه انتخاب شده برابر 2.2ولت 3.3، 4.7، 5.1میباشد در پیک مثبت در بین ولتاژ سفر تا 2.2+0.7 دیود هیچ کدام از دیودها فعال نمیباشند در این صورت ولتاژ ورودی با ولتاژ خروجی برابر می باشد اما بعد از آن تا ولتاژ 4

ولت فقط دیود اول فعال میباشد در این صورت مقاومت 62k و مقاومت 10k باعث کاهش نسبی ولتاژ میگردد برای بقیه دیودها این وضع ادامه میابد تا به ولتاژ سینوسی برسیم.

چگونگی محاسبه

چون شیب رمپ برابر شیب در نقطه صفر در موج سینوسی می باشد داریم

$$Vsin = A \sin(\pi/2\, t);$$

$$Vramp = B\, t$$

$$Vsin'(0) = Vramp'(0);$$

$$A\, \pi/2 \cos(\pi/2\, t) = B$$

$$A = 2B / \pi$$

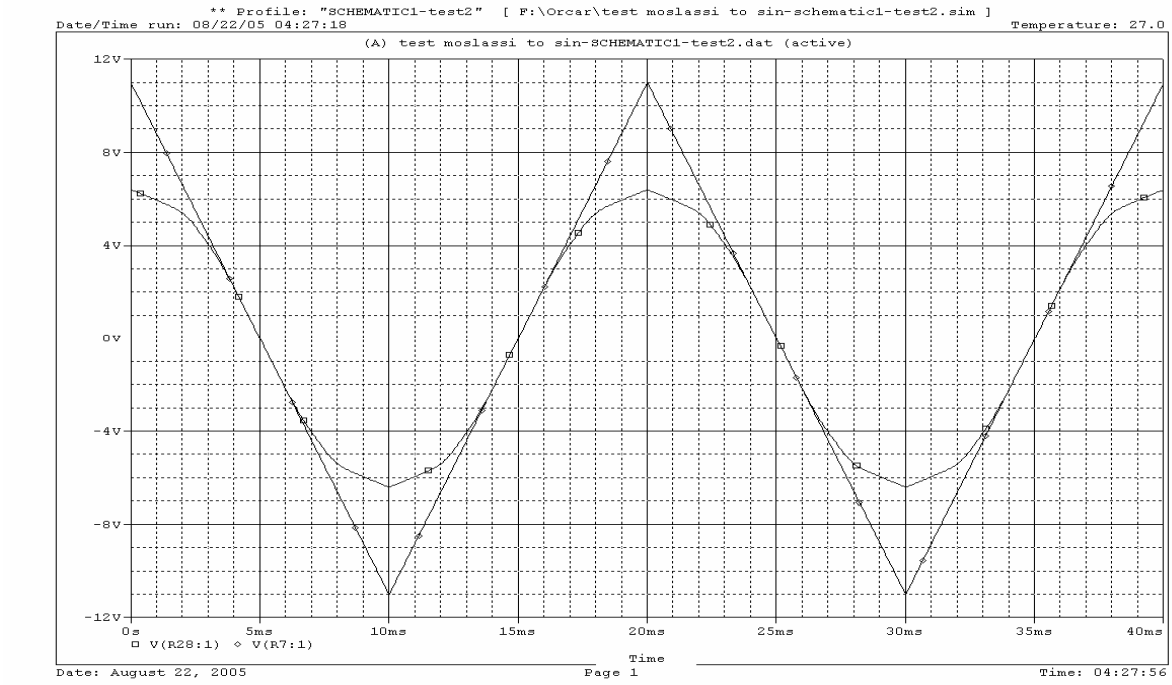
A دامنه موج سینوسی و B دامنه موج رمپ می باشد که با توجه به مقدار گرفته شده 10 ولت برای موج مثلثی مقدار موج سینوس در حدود 6.36 به دست می آید.

فرض کنید n عدد دیود فعال شده باشند در این صورت ولتاژ موج سینوسی برابر با Vsin1 باشد در این صورت ولتاژ موج مثلثی برابر

$$(Vramp - Vsin)/R = (Vsin - V1)/R1 + (Vsin - V2)/R2 + \ldots + (Vsin - Vn)/R$$

برای محاسبه مقاومت اول فرض را بر این گذاشه که مقاومت 10Kرا انتخاب کرده چون دیود اول در ولتاژ 2.9وصل میشود .میتوان به راحتی مقدار 6.2را برای مقاومت اول بدست آورد.وهمچنین با داشتن این مقاومت میتوان مقاومت بعدی را بدست آور تا به مقاومت نهایی رسید.

قابل ذکر است به علت متقارن بودن مدار فقط نیاز به محاسبه نصف مدار میباشد و مقاومت های منفی را ازآن همان مقاومتهای قسمت مثبت میباشد .



کنترل خروجی

برای آنکه خروجی را نیز کنترول کرد از نظر نوع موج و دامنهاز دو آی سی 4052و AD7528 نصف آن استفاده شده است .

آی سی 4052یک آنالگ سویچ یک به چهار میباشد که باآن نوع موج خروجی (مربعی، مثلثی و سینوسی) را انتخاب میکنیم این آی همانند ای سی 4051میباشد .خروجی این ای سی به Vrefآسی AD758رفته که همان آسی که برای کنترل فرکانس هم استفاده میگردد.

ای اسی در این صورت مانند یک ولوم عمل کرده و ولتاژ را کاهش داده که با یک تقویت کنده opampی به مقدار ولتاژ مورد نظر رسید در این صورت کار کنترول دامنه نیز به صورت دیجیتال در آمده است.

میکروکنترولر Atmega16L

این میکرو دارای 32تا I/Oامیباشد

ولتاژ تغذیه 5.5تا 2.7را میتواند تحمل کند

دارای K16بایت حافظه فلش (قابل برنامه ریزی)

و 1024بایت Ramمیباشد

واز فرکانسهای m1وm2وm4و M8را بطور داخلی استفاده میکند

در اینجا این میکرو برای کنترول سیستم استفاده میشود که به وسیله 4کلید که برروی برد تعبیه شده است میتوان کنترول سیستم را به دست گرفت .

برای برنامه نویسی میکرو کنترلر AVRزبان Cکه در قالب codveitonاستفاده میکنیم .

برای کنترل کلیدها که یک سر آنها به زمین وصل شده است و دیگری به میکرو، به طور داخلی به وسیله ریجسترهای کنترول پورت میکرو پولاپ شده که هنگامی که کلیدی فشار داده نشده باشد عدد یک خوانده و در صورت فشار دادن کلید عدد صفر را میکرو بخواند.

برای خواندن کلید از تابع ()GetKeyاستفاده شده است

این تابع چک میکند که آیا کلیدی فشار داده شده است یا خیر .در صورتی که کلید ی فشار داده شود مطابق با آن در خروجی عددی قرار میدهد .

LCD Text

این وسیله برای نمایش خروجی سیستم میباشد.که دارای دو سطر 16کاراکتری است.

فرکانس خروجی ،دامنه خروجی ونوع موج خروجی در این LCDقابل نمایش میباشد.

دستورات مورد نیاز

```
lcd_init(16);
```

این دستور راه انداز LCD Textمیباشد که چگونگی کارکرد آنرا تنظیم میکند .

```
lcd_gotoxy(0,3);
```

این دستور برای بردن خط نشان به نقطه مورد نظر میباشد به عنوان مثال مکان نما را به خط اول کاراکتر چهارم میبرد .

```
lcd_putchar('0');
```

این دستور یک کاراکتر مورد نظر را در جای مکان نما قرار می دهد .

```
lcd_puts (str);
```

این دستور برای نمایش یک سری کاراکتر بر روی LCDمیباشد .این دستور کاراکتر ها را از داخل حافظه RAMبرداشته و بر روی LCDنشان می دهد .

```
lcd_putsf("KHz ");
```

این دستور برای نمایش یک سری کاراکتر بر روی LCDمیباشد .این دستور کاراکتر ها را از داخل حافظه Flashبرداشته و بر روی LCDنشان می دهد.

این دستورات در فایل Lcd.Hذخیره شده است که با دستور _#include <lcd.h>فرا خوانی میگردد .

دستور دیگری که در این برنامه استفاده شده است دستور

void define_char(char flash *pc,char char_code)

میباشد که برای ایجاد کاراکتر جدید میتوان از آن استفاده نمود.برای اطلاع بیشتر به راهنمای این برنامه مراجعه شود.

در اینجا چند دستور دیگر مورد برسی قرار میگیرد.

delay_ms(100);

این دستور برای ایجاد تاخیر در برنامه میشود .

ltoa(256,srt);

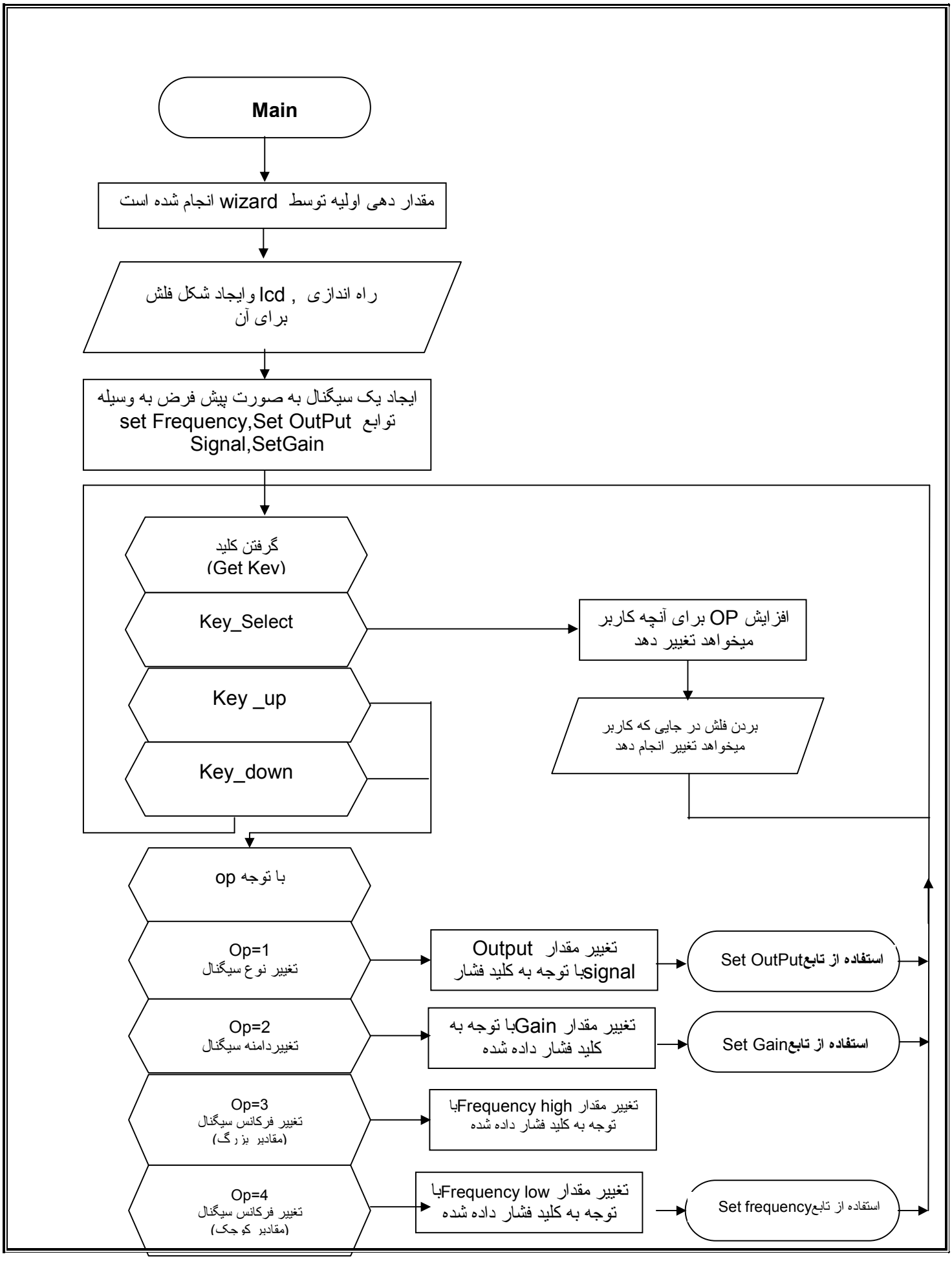
این دستور برای تبدیل عدد به کد ASCIIکه LCDبتواند آن را نمایش بدهد.

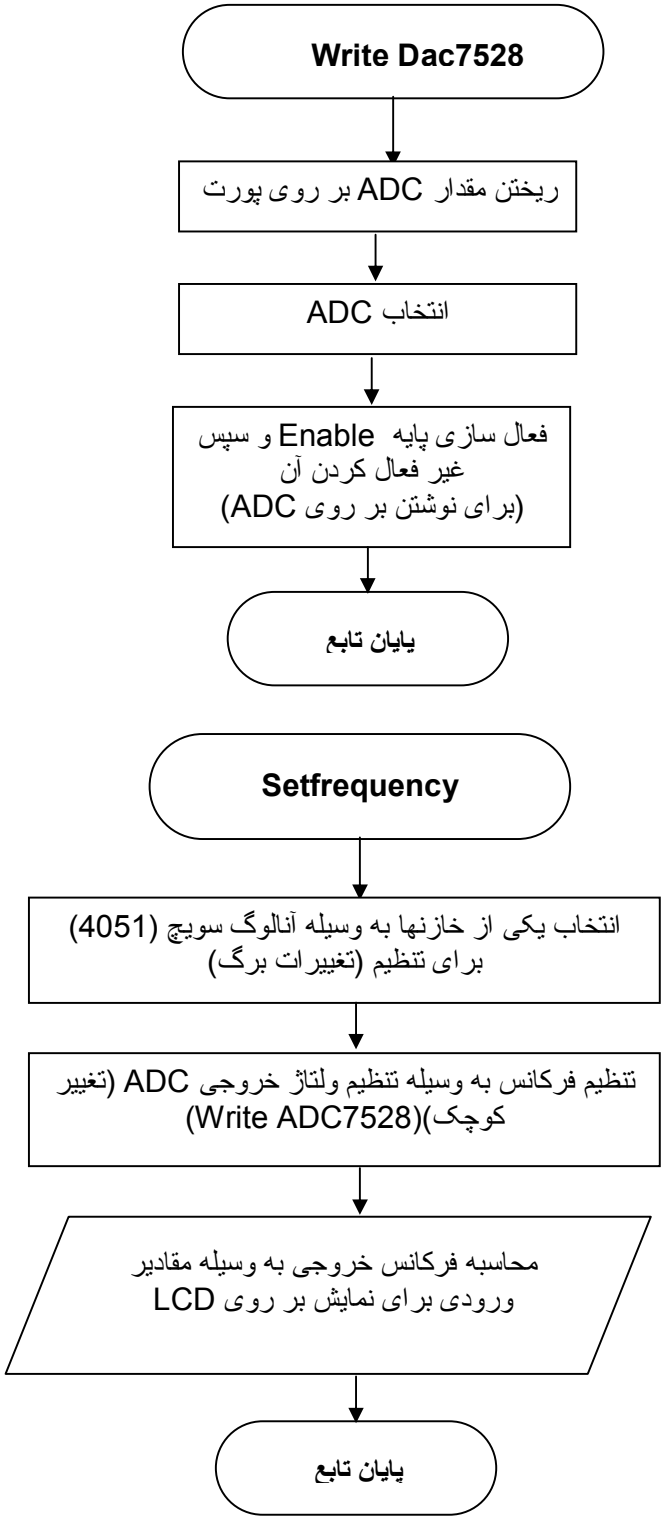
ساختار برنامه :

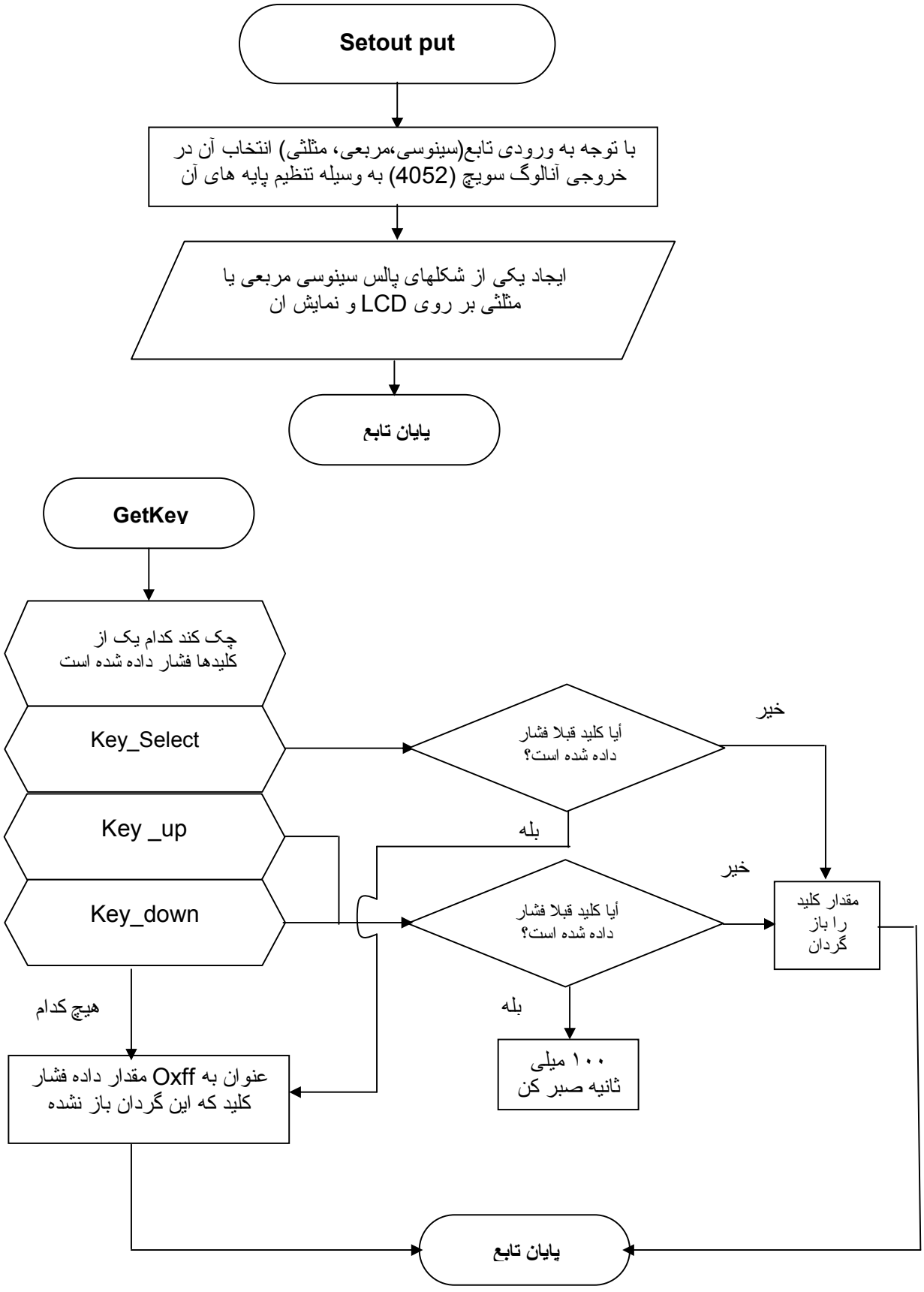
این برنامه ابتدا بعد از تظیم نمودن مقادیر اولیه سیستم به داخل حلقه بی نهایت ()Whileمی افتد که اصل بر نامه در اینجا قرار دارد .سیستم منتظر میماند تا کلیدی فشار داده شود .

در صورتی که کلید Selectباشد باعث تعویض متغیر سیستم می شود که بین متغیر های نوع خروجی ، دامنه خروجی و فرکانس آن در حال گردش می باشد به عنوان مثال در صورتی بروی نوع موج باشد به سراغ دامنه می رود .

در صورتی که کلید upو Downباشد متغیر مورد نظر را تغیر داده و خروجی آنرا در سخت افزار اجرا میکند


Main
مقدار دهی اولیه توسط wizard انجام شده است
راه اندازی , lcd وایجاد شکل فلش برای آن
ایجاد یک سیگنال به صورت پیش فرض به وسیله توابع set Frequency,Set OutPut Signal,SetGain
گرفتن کلید (Get Kev)
Key_Select
Key _up
Key_down
افزایش OP برای آنچه کاربر میخواهد تغییر دهد
بردن فلش در جایی که کاربر میخواهد تغییر انجام دهد
با توجه op
Op=1 تغییر نوع سیگنال
Op=2 تغییردامنه سیگنال
Op=3 تغییر فرکانس سیگنال (مقادیر بزرگ)
Op=4 تغییر فرکانس سیگنال (مقادیر کوچک)
تغییر مقدار Output signalبا توجه به کلید فشار
تغییر مقدار Gainبا توجه به کلید فشار داده شده
تغییر مقدار Frequency highبا توجه به کلید فشار داده شده
تغییر مقدار Frequency lowبا توجه به کلید فشار داده شده
استفاده از تابعSet OutPut
استفاده از تابعSet Gain
استفاده از تابعSet frequency


Write Dac7528
ریختن مقدار ADC بر روی پورت
انتخاب ADC
فعال سازی پایه Enable و سپس
غیر فعال کردن آن
(برای نوشتن بر روی ADC)
پایان تابع
Setfrequency
انتخاب یکی از خازنها به وسیله آنالوگ سویچ (4051)
برای تنظیم (تغییرات برگ)
تنظیم فرکانس به وسیله تنظیم ولتاژ خروجی ADC (تغییر
کوچک)(Write ADC7528)
محاسبه فرکانس خروجی به وسیله مقادیر
ورودی برای نمایش بر روی LCD
پایان تابع


Setout put
با توجه به ورودی تابع(سینوسی،مربعی، مثلثی) انتخاب آن در خروجی آنالوگ سویچ (4052) به وسیله تنظیم پایه های آن
ایجاد یکی از شکلهای پالس سینوسی مربعی یا مثلثی بر روی LCD و نمایش ان
پایان تابع
GetKey
چک کند کدام یک از کلیدها فشار داده شده است
Key_Select
Key _up
Key_down
آیا کلید قبلا فشار داده شده است؟
خیر
بله
آیا کلید قبلا فشار داده شده است؟
خیر
مقدار کلید را باز گردان
بله
۱۰۰ میلی ثانیه صبر کن
هیچ کدام
عنوان به Oxff مقدار داده فشار کلید که این گردان باز نشده
پایان تابع

برنامه نرم افزاری

```
This program was produced by the
CodeWizardAVR V1.23.8c Standard
Automatic Program Generator
©

Project :
Version :
Date    : 1/8/2005
Author  : n.aslani
Company                         :
Comments :


Chip type           : ATmega16L
Program type        : Application
Clock frequency     : 1.000000 MHz
Memory model        : Small
External SRAM size  : 0
Data Stack size     : 256
/*****************************************

#include <mega16.h<
#include <delay.h<
#include <stdlib.h<
 //Alphanumeric LCD Module functions
#asm
.   equ __lcd_port=0x12
#endasm
#include <lcd.h<

 //Declare your global variables here
#define DAC_PORT  PORTA
#define DAC_PIN_WR  PORTB.4
#define DAC_PIN_E   PORTB.3
#define DAC_PIN_AB  PORTB.2

#define F51_PIN0    PORTB.5
#define F51_PIN1    PORTB.6
#define F51_PIN2    PORTB.7

#define F52_PIN0    PORTB.0
#define F52_PIN1    PORTB.1
```

```
#define Key_Select  PINC.0
#define Key_Up      PINC.3
#define Key_Down    PINC.2


void define_char(char flash *pc,char char_code)
{
  char i,a;
  a  = (char_code<<3) | 0x40;
  for (i=0; i<8; i++) lcd_write_byte(a++,*pc;(++
{



void WriteDAC7528(unsigned char DAC,unsigned char AB(
}
 DAC_PORT = DAC;
 if (AB(
}
   DAC_PIN_AB =1;
{
   else
}
   DAC_PIN_AB =0;
  ;{
 DAC_PIN_WR = 0;
 DAC_PIN_E  = 0 ;
#  asm("nop("
 DAC_PIN_WR = 1;
 DAC_PIN_E  = 1;
{


flash char ChMosalasi [] ={ 0x00,0x00,0x00,0x00,0x00,0x11,0x0A,0x04;{
flash char CMosalasi2[] ={ 0x00,0x04,0x0A,0x11,0x00,0x00,0x00,0x00  ;{
flash char ChMoraba   [] ={ 0x00,0x00,0x07,0x04,0x04,0x1C,0x00,0x00;{
flash char ChMoraba2  [] ={ 0x00,0x00,0x1C,0x04,0x04,0x07,0x00,0x00;{
flash char ChSin      [] ={ 0x01,0x02,0x04,0x04,0x04,0x08,0x10,0x00;{
flash char ChSin2     [] ={ 0x10,0x08,0x04,0x04,0x04,0x02,0x01,0x00;{
flash char ChFelesh2  [] ={ 0x00,0x04,0x08,0x1F,0x08,0x04,0x00,0x00;{
flash char ChFelesh   [] ={ 0x00,0x04,0x02,0x1F,0x02,0x04,0x00,0x00;{
#define  LSignalX 11
#define  LSignalY 0
#define  LGainX   1
#define  LGainY  0
#define  LFrequencyX 1
#define  LFrequencyY 1
```

```
#define  LFrequencyX2 15


void SetOutputSignal(unsigned char Mode(
 }
  switch( Mode(
}
  case  0:        // mosalsi
      F52_PIN0 = 0;
      F52_PIN1 = 0;
      define_char(ChMosalasi ,0;(
      define_char(ChMosalasi2,1;(
      break;
  case  1:       //  sin
      F52_PIN0 = 1;
      F52_PIN1 = 0;
      define_char(ChSin  ,0;(
      define_char(ChSin2 ,1;(
      break     ;
  default :       // Moraba
      F52_PIN0 = 0;
      F52_PIN1 = 1;
      define_char(ChMoraba  ,0;(
      define_char(ChMoraba2 ,1;(
{
  lcd_gotoxy(LSignalX,LSignalY;(
  lcd_putsf("\8\9\8\9\8         ;("

{
void SetGain(unsigned char A(
}
  char str[10;[
  WriteDAC7528(A,1;(
  lcd_gotoxy(LGainX,LGainY;(

  lcd_putsf("G ;("=
  itoa((A/20),str;(
  lcd_puts   (str ;(
  lcd_putchar;('.')
  itoa((int)(A%20)*5,str;(
  lcd_puts   (str ;(
  lcd_putsf("v ;("

{
void SetFrequency(unsigned char Ftq ,unsigned char DBFtq(
}
  unsigned char str[10;[
```

```
  unsigned long int  K;
  Ftq +=50;
  switch( DBFtq(
}
  case  0:          //1 Hz
     F51_PIN0 = 1;
     F51_PIN1 = 0;
     F51_PIN2 = 1;
     K = 1;
     break    ;
  case  1:          //10 Hz
     F51_PIN0 = 0;
     F51_PIN1 = 0;
     F51_PIN2 = 1;
     K = 10;
     break    ;
  case  2:         //100 Hz
     F51_PIN0 = 1;
     F51_PIN1 = 1;
     F51_PIN2 = 0;
     K = 100;
     break    ;
  case  3:         //1k Hz
     F51_PIN0 = 0;
     F51_PIN1 = 1;
     F51_PIN2 = 0;
     K = 1;
     break    ;
  case  4:         //10k Hz
     F51_PIN0 = 1;
     F51_PIN1 = 0;
     F51_PIN2 = 0;
     K = 10;
     break;
  case  5:        //100k Hz
     F51_PIN0 = 0;
     F51_PIN1 = 0;
     F51_PIN2 = 0;
     K = 100;
     break    ;
     {

  WriteDAC7528(Ftq,0;(
  lcd_gotoxy(LFrequencyX,LFrequencyY;(
  lcd_putsf("Frq ;("=

  K = K * Ftq;
```

```
  ltoa((K/100),str;(
  lcd_puts   (str ;(

  if ((K%100) != 0(
}
    lcd_putchar;('.')
    ltoa((K%100),str;(
    lcd_puts   (str ;(
{
  if (DBFtq >= 3(
    lcd_putsf("KHz  ;("
  else
    lcd_putsf("Hz  ;("
{

char GetKey()
}
static   char KeySelect_Dwon ;
static   char KeyUp_Dwon ;
static   char KeyDown_Dwon ;

if (Key_Select == 0(
}
   delay_ms(10;(
   if (KeySelect_Dwon==0(
}
       return 0xFF;
{
   KeySelect_Dwon=0;
   return 1;
{
   else
    KeySelect_Dwon=1;


if (Key_Up == 0(
}
   delay_ms(30;(
   if (KeyUp_Dwon!=0(
}
      delay_ms(100;(
      KeyUp_Dwon=0;
{
   return 3;
{
   else
    KeyUp_Dwon=1;
```

```
if (Key_Down == 0(
}
   delay_ms(30;(
   if (KeyDown_Dwon!=0(
}
     delay_ms(100;(
     KeyDown_Dwon=0;
{
   return 2;
{
   else
    KeyDown_Dwon=1;
return 0xFF     ;


{


void main(void(
}
 //Declare your local variables here
  unsigned char OutputSignal = 0,
               Gain = 20,
               FrequencyHigh = 1,
               FrequencyLow  = 50;

 //Input/Output Ports initialization
 //Port A initialization
 //Func0=Out Func1=Out Func2=Out Func3=Out Func4=Out Func5=Out Func6=Out
Func7=Out
 //State0=0 State1=0 State2=0 State3=0 State4=0 State5=0 State6=0 State7=0
PORTA=0x00;
DDRA=0xFF;

 //Port B initialization
 //Func0=Out Func1=Out Func2=Out Func3=Out Func4=Out Func5=Out Func6=Out
Func7=Out
 //State0=0 State1=0 State2=0 State3=0 State4=0 State5=0 State6=0 State7=0
PORTB=0x00;
DDRB=0xFF;

 //Port C initialization
 //Func0=In Func1=In Func2=In Func3=In Func4=In Func5=In Func6=In Func7=In
 //State0=P State1=P State2=P State3=P State4=P State5=P State6=P State7=P
PORTC=0xFF;
DDRC=0x00;
```

```
 //Port D initialization
 //Func0=In Func1=In Func2=In Func3=In Func4=In Func5=In Func6=In Func7=In
 //State0=T State1=T State2=T State3=T State4=T State5=T State6=T State7=T
PORTD=0x00;
DDRD=0x00;

 //Timer/Counter 0 initialization
 //Clock source: System Clock
 //Clock value: Timer 0 Stopped
 //Mode: Normal top=FFh
 //OC0 output: Disconnected
TCCR0=0x00;
TCNT0=0x00;
OCR0=0x00;

 //Timer/Counter 1 initialization
 //Clock source: System Clock
 //Clock value: Timer 1 Stopped
 //Mode: Normal top=FFFFh
 //OC1A output: Discon.
 //OC1B output: Discon.
 //Noise Canceler: Off
 //Input Capture on Falling Edge
TCCR1A=0x00;
TCCR1B=0x00;
TCNT1H=0x00;
TCNT1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;

 //Timer/Counter 2 initialization
 //Clock source: System Clock
 //Clock value: Timer 2 Stopped
 //Mode: Normal top=FFh
 //OC2 output: Disconnected
ASSR=0x00;
TCCR2=0x00;
TCNT2=0x00;
OCR2=0x00;

 //External Interrupt(s) initialization
 //INT0: Off
 //INT1: Off
 //INT2: Off
```

```
GICR|=0x00;
MCUCR=0x00;
MCUCSR=0x00;

 //Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x00;

 //Analog Comparator initialization
 //Analog Comparator: Off
 //Analog Comparator Input Capture by Timer/Counter 1: Off
 //Analog Comparator Output: Off
ACSR=0x80;
SFIOR=0x00;

 //LCD module initialization
lcd_init(16;(

 define_char(ChFelesh ,7;(
 define_char(ChFelesh2 ,6;(
 SetOutputSignal(OutputSignal;(
 SetGain( Gain;(
 SetFrequency(FrequencyLow,FrequencyHigh;(
while (1(
}
     unsigned char  Key;
     unsigned char  op;
     Key = GetKey;()

     lcd_gotoxy(0,3;(
     lcd_putchar(Key+'0;('
     if (Key == 1(
 }
        op;++
        if ( op > 4) op =1;
        switch(op(
}
          case 1: // Signal
             lcd_gotoxy(LFrequencyX2,LFrequencyY;(
             lcd_putchar;(' ')
             lcd_gotoxy(LSignalX-1,LSignalY;(
             lcd_putchar(7;(
             break;
          case 2: // GainX
             lcd_gotoxy(LSignalX-1,LSignalY;(
             lcd_putchar;(' ')
             lcd_gotoxy(LGainX-1,LGainY;(
             lcd_putchar(7;(
```

```
            break;
          case 3: // Frequency
            lcd_gotoxy(LGainX-1,LGainY;(
            lcd_putchar;(' ')
            lcd_gotoxy(LFrequencyX-1,LFrequencyY;(
            lcd_putchar(7;(
            break;
          case 4: // Frequency Low
            lcd_gotoxy(LFrequencyX-1,LFrequencyY;(
            lcd_putchar;(' ')
            lcd_gotoxy(LFrequencyX2,LFrequencyY;(
            lcd_putchar(6;(
            break;
      {
{
      if (Key == 2)// Key up
 }
        switch(op(
}
          case 1: // Signal
            OutputSignal;++
            if (OutputSignal >2 (
                OutputSignal=0  ;
            SetOutputSignal(OutputSignal;(
            break;
          case 2: // Gain
            Gain;++
            SetGain(Gain;(
            break;
          case 3: // Frequency
            FrequencyHigh;++
            if (FrequencyHigh >5 (
               FrequencyHigh =5  ;

            SetFrequency(FrequencyLow,FrequencyHigh;(
            break;
          case 4: // Frequency Low
            FrequencyLow;++
            if (FrequencyLow >200 (
               FrequencyLow =200  ;
            SetFrequency(FrequencyLow,FrequencyHigh;(
            break;
      {
{
      if (Key == 3)// Key Down
 }
        switch(op(
```

```
}
        case 1: // Signal
          OutputSignal;++
          if (OutputSignal >2 (
              OutputSignal=0  ;
          SetOutputSignal(OutputSignal;(
          break;
        case 2: // Gain
          Gain;--
          SetGain(Gain;(
          break;
        case 3: // Frequency
          FrequencyHigh;--
          if (FrequencyHigh >5 (
             FrequencyHigh =0  ;
          SetFrequency(FrequencyLow,FrequencyHigh;(
          break;
        case 4: // Frequency Low
          FrequencyLow;--
          if (FrequencyLow >200 (
             FrequencyLow =0  ;
          SetFrequency(FrequencyLow,FrequencyHigh;(
          break;
    }
}
```

نتیجه گیری :

این مدار برای تولید امواج سینوسی، مثلثی، مربعی در محدوده دامنه ۱۵ الی ۱۵- ولت در محدوده فرکانس ۵۰ هرتز الی ۳۰ کیلو هرتز کار میکند. در اینجا به چند مورد از اشکالات مدار اشاره می شود که این مشکلات ناشی از قطعات استفاده شده در این مدار می باشد.

Opamp : این المان تا فرکانس 1MHZ مي تواند کار کند ولی به دلیل Slowreat آن برای پیک ولتاژ بالاتر کاربرد کمتری دارد.به علت این مشکل در فرکانسهای بالا مشکلات زیادی بوجود می آید.

آنالوگ سویچ ها : این دو IC در هر پایه خازنهای معادل چند پیکو فاراد دارند که هنگام استفاده از فازهای کوچک با آنها موازی میشوند و با مقدار خازن استفاده شده جمع شده و ایجاد اختشاش میکند.

با توجه به نوع چیدما نمدار و استفاده از برد سوراخ دار سلفها و خازنهای ناخواسته در مدار ایجاد می شود که باعث ناپایداری و ایجاد هارمونیک هایی از فرکانس در درون مدار میگردد