مدارهای میکرو پروسسوری با سطوح منطقی صفر و یک عمل میکنند.
اما مواقعی پیش می آید که ما با مقادیر پیوسته (آنالوگ) سر و کار داریم , بنابر این اگر بخواهیم از طریق یک سیستم میکرو پروسسوری سیستمی با مقادیر پیوسته (آنالوگ) کنترل کنیم لازم است از یک آی سی واسطه برای تبدیل اعداد باینری به مقدار آنالوگ آن استفاده کنیم.
به این آی سی ها مبدل دیجیتال به آنالوگ و یا
DAC
گفته میشود.
این آیسی ها مقادیر باینری را با توجه به ولتاژی که به پایه ورودی آن داده میشود میسازند.
آیسی ای که ما در این پروژه استفاده کردیم
Dac0800
میباشد که یک مبدل دیجیتال به آنالوگ ۸ بیتی میباشد.
هدف ما در استفاده از این آیسی در این پروژه این است که توسط دو عدد از این آیسی ها دو موج سینوسی تولید کنیم.
برای ایجاد موج سینوسی ما به مقادیر آنالوگ احتیاج داریم که با کمک این آیسی مقادیر میکرو را آنالوگ خواهیم کرد.
پایه های ۵ تا ۱۲ این آیسی ورودی های این ۸ بیت میباشد.
که پایه شماره ۵ با ارزش ترین بیت و پایه ۱۲ کم ارزش ترین بیت میباشد
پایه ۱۴ ورودی منفی و پایه ۱۵ ورودی مثبت برای ایجاد خروجی مناسب میباشند.
سایر اطلاعات مربوط به این آیسی و نقشه راه انداز آن را میتوانید از داخل فایل شماتیک این آیسی مشاهده کنید.

در مورد برنامه :

برای تولید موج سینوسی توسط آیسی های مبدل ما باید مقادیر متناظر با ولتاژ لحظه ای این موج را توسط میکرو کنترلر ایجاد کرده و به آی سی مبدل بدهیم تا در خروجی موج سینوسی داشته باشیم.
البته به دلیل اینکه حد اکثر این آیسی میتواند ۲۵۶ حالت داشته باشد موج ما سینوسی کامل نیست اما بسیار شبیه است.
مقدار لحظه ای یک موج سینوسی در واحد باینری با فرمول زیر محاسبه میشود.
((sin((x/255)*(2*PI))+1)/2)*255
به دلیل اینکه محاسبه این فرمول در هنگام اجرای برنامه میکرو کنترلر فرایندی زمان بر است و ما برای ایجاد یک موج دقیق تر احتیاج به سرعت بالا داریم , به همین منظور این مقادیر را از قبل حساب کرده و داخل میکرو کنترلر قرار میدهیم.
به اینگونه اطلاعات جداول لوک آپ گفته میشود.
به دلیل اینکه اعداد باینری از صفر شروع میشوند لذا ما مجبوریم از عدد ۱۲۸ که نصف ۲۵۶ میباشد برای حالت صفر موج استفاده کنیم.
تا اعداد پایینتر برای سیکل منفی و اعداد بالاتر برای سیکل مثبت این موج مورد استفاده قرار بگیرند.
این فرمولی که در بالا ذکر شد نیز اعداد را طبق خواسته ما بدست می آورد.
در این برنامه چون ما باید دامنه را نیز کنترل کنیم لذا از ۵ جدول لوک آپ با تفاوت ۲۰ درصد استفاده کردیم.

و در هنگام باز خوانی از این جداول با توجه به دامنه تنظیم شده توسط کاربر از جدول مناسب استفاده میشود.
همانطور که گفته شد هدف ما از این پروژه ساخت دو موج سینوسی است با قابلیتهای
تنظیم فرکانس
تعین اختلاف فاز بین دو موج
تعین دامنه موج

تعین فاز در این برنامه بر اساس درجه آن میباشد و از ۰ تا ۳۵۹ درجه را ایجاد میکند.
هر سیکل کامل که در جداول لوک آپ وارد شده است شامل ۲۵۶ حالت است که از صفر تا ۲۵۵ میباشد.
یعنی برای یک سیکل کامل ما ۲۵۶ نمونه را به مبدل دیجیتال به آنالوگ میدهیم.
بنا براین تعین اختلاف فاز نیز یعنی تفاوت در این ۲۵۶ نمونه.
در برنامه ما دو متغیر
(
unsigned char ph1 = 0;
unsigned char ph2 = 0;
)
از نوع ۸ بیتی بدون علامت عهده دار چرخش داخل این جداول لوک آپ هستند و مقدار متناظر با خانه ای که این متغیر ها بدان اشاره میکنند به بدل مربوطه ارسال میشود.
در حالتی که اختلاف فاز صفر درجه است , مقادیر این دو متغیر با هم یکسان هستند .
چون اختلاف فاز ما ۳۶۰ مورد (بر اساس ۳۶۰ درجه) میباشد و جدول لوک آپ ۲۵۶ مورد دارد لذا مقدار اختلاف فاز باید در
(256/360)
ضرب شود تا عدد متناظر با اختلاف فاز بدست آید.
در هر بار که این اختلاف فاز تغیر میکند مقدار متغیر
Ph1
برابر صفر میشود و مقدار متغیر
Ph2
نیز مقدار درجه اختلاف فاز ضرب در عدد ۰,۷۱۱۱۱ که معادل
256/360
میباشد .

برای دقیق بودن زمان فرستادن نمونه ها به مبدل ازوقفه تایمر یک که یک تایمر ۱۶ بیتی است استفاده کردیم.
هنگامی که مقدار وقفه سر ریز میشود روتین وقفه تایمر فراخوانی میشود و دستوراتی که داخل این روتین نوشته شده اجر میشود.
برای تنظیم فرکانس موج مورد نظر باید مقدار تایمر را تغیر دهیم تا وقفه سر ریزی تایمر طبق زمانبندی ما رخ دهد.
کدی که داخل روتین وقفه تایمر نوشته ایم کد ساده ای میباشد که در زیر میبینیم

```
TCNT1=T1;
dac1 = source[Psec][ph1];
dac2 = source[Psec][ph2];
```

ph1+=2;
ph2+=2;

TCNT1
که در خط اول مقدار دهی شده است رجیستر مقدار ۱۶ بیتی تایمر یک میباشد.
با مقدار دهی این رجیستر است که میتوانیم فرکانس موج را تنظیم کنیم.
متغیر
T1
نیز مقدار مناسب برای فرکانس تعین شده را در خود نگه داری میکند.
دو مولفه ای که در خط دوم و سوم این دستورات مقدار دهی شده اند دو پورت از میکرو کنترلر هستند که برای سادگی توسط دو دستور زیر با این اسامی نامگذاری شده اند
#define dac1 PORTA
#define dac2 PORTD
یعنی پایه های دیتای مبدل اول به
PORTA
از میکرو متصل شده و پایه های دیتای مبدل دوم نیز به
PORTD
میکروی ما متصل است.

کار با ال سی دی با کامپایلر کدویژن بسیار ساده است.
برای راحتی کار میتوانیم توسط ابزار کدویزارد به تب ال سی دی رفته و پورت مورد نظر خود را برای کنترل ال سی دی معرفی کنیم.
در برنامه ما از پورت سی برای استفاده از ال سی دی استفاده شده است.
و توسط دستور هایی از قبیل
Lcd_putsf
Lcd_puts
اطلاعات خود را روی صفحه ال سی دی نمایش داده ایم.

در این برنامه از یک صفحه کلید ۴*۴ استفاده شده که طریقه خواندن کلید فشرده شده در آن به اصطلاح روش سرکشی گفته میشود.
ما برای این کار کتابخوانه ای طراحی کردیم تا بتوانیم در برنامه های دیگر نیز براحتی از صفحه کلید استفاده کنیم.
حال به شرح مختصری راجع به این کتابخوانه میپردازیم :

این کتابخوانه کلا دو تابع دارد

```
void keypad_init(void);
unsigned char scan_key(void);
```

تابع اول برای مقدار دهی اولیه به پورت مورد نظر و تعین ورودی و خروجی بودن پینهای پورت مربوطه به کار رفته است.

```
void keypad_init(void)
{

keypad_DDR=0x0F;
keypad_PORT=0xF0;

}
```

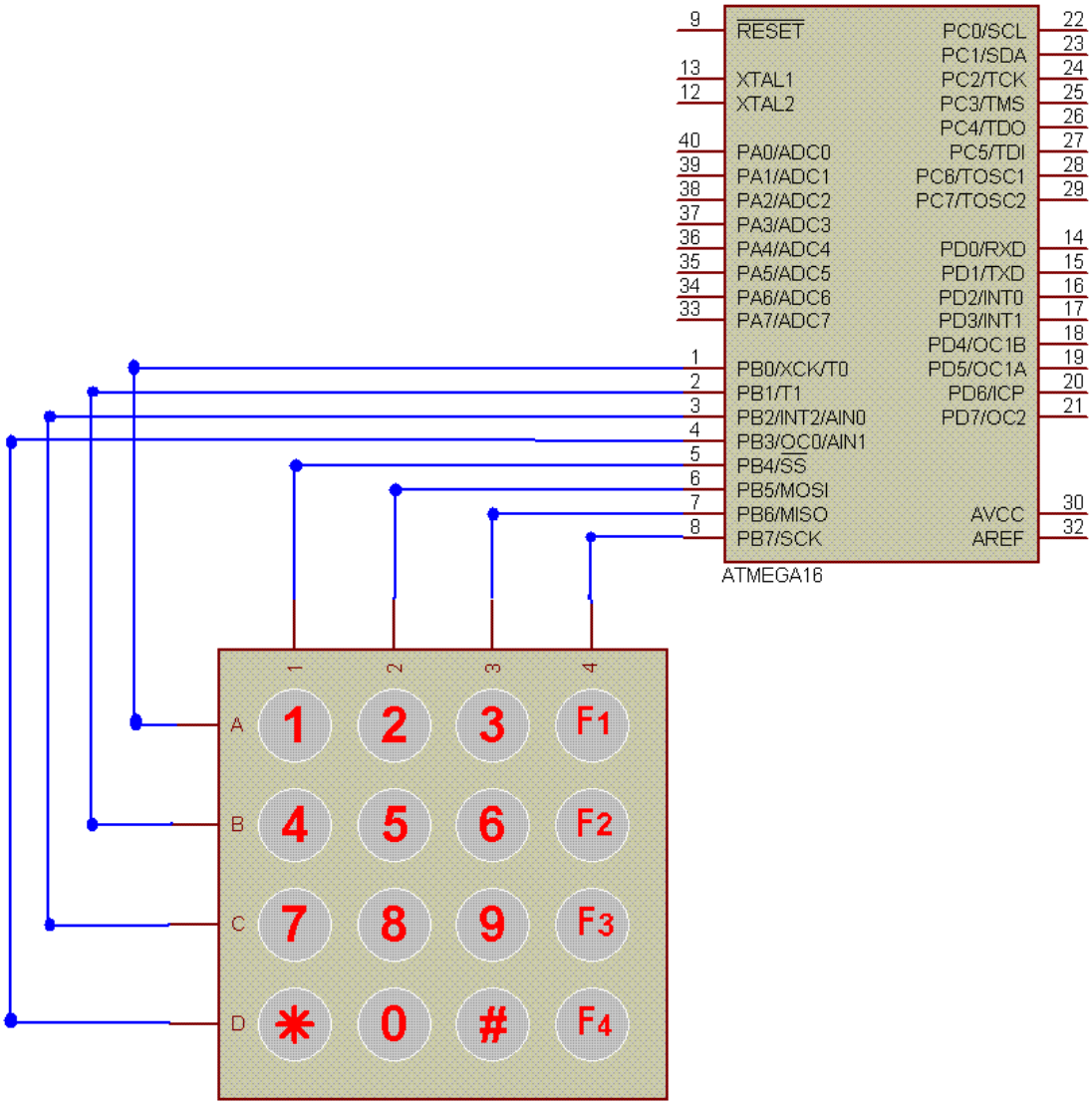
***نکته مهمی که باید به آن توجه کنیم این اس که برای استفاده از این کتابخوانه , باید در برنامه خود سه ماکروی زیر را به این شکل تعریف کنیم.**

```
#define keypad_DDR DDRB
#define keypad_PORT PORTB
#define keypad_PIN PINB
```

این کار برای این است که برنامه نویس هر پورتی را که لازم داشت برای استفاده از کی پد مورد استفاده قرار دهد.

تابع دوم تابعی است که برای دیافت کد کلید فشرده شده از آن استفاده میکنیم.

ابتدا به نحوه اتصال کی پد به پورت میکرو توجه میکنیم.



نحوه کار به این شکل است که ابتدا ۴ پین از میکرو که به سطر های صفحه کلید بصورت خروجی تعریف میشود و ۴ پین دیگر به به سطر ها وصل است به عنوان ورودی .

همینطور مقدار اولیه ورودی ها نیز ست میشود تا هنگام خواندن از پورت مقدار ورودی صفر که نشان دهنده اتصال یا همان فشرده شدن کلیدی از کی پد است مشخص شود.

در صورتی که یکی از پینهای ورودی برابر صفر باشد , یعنی کلیدی فشرده شده است.

ما با خواندن از پینهای ورودی میتوانیم سطری که کلید فشرده شده در آن قرار دارد را مشخص کنیم.

برای مشخص شدن ستون و در نهایت کلید فشرده شده اینبار ۴ پینی که به سطر ها متصل است را ورودی تعریف میکنیم و با مقدار دهی و خواندن از پورت , میتوانیم سطر مورد نظر و در نهایت کلید فشرده شده را پیدا میکنیم.

به این روش , روش سر کشی میگویند.

توابعی که در برنامه اصلی هستند به شرح زیر میباشند

unsigned int get_num(unsigned int min, unsigned int max)

که برای گرفتن اعداد از کی پد است.

برنامه به شکلی است که عدد خوانده شده از صفحه کلید بین مینیمم و ماکسیمم تعین شده در ورودی های این ابع است.

void lcd_panel1(void)

که برای نمایش اطلاعات فرکانس و اختلاف فاز و ... روی صفحه ال سی دی و فعال شدن سرویس وقفه ها به کار میرود.

void panel2(void) // change freq

این تابع را هنگامی فراخوانی میکنیم که کلید فانکشن اول از کی پد فشرده شده باشد , کار این تابع این است که فرکانس موج ما را طبق درخواست کاربر و توسط صفحه کلید تعین میکنیم.

void panel3(void) // change phase

این تابع را هنگامی فراخوانی میکنیم که کلید فانکشن دوم از کی پد فشرده شده باشد , کار این تابع ان است که اختلاف فاز بین دو موج ا طبق درخواست کاربر و توسط صفحه کلید تعین میکنیم.

void panel4(void) // change scale

این تابع را هنگامی فراخوانی میکنیم که کلید فانکشن سوم از کی پد فشرده شده باشد , کار این تابع این است که دامنه فرکانس را توسط صفحه کلید از کاربر دریافت کند.