به نام خدا

پایان نامه پروژه فانکشن ژنراتور

مقدمه :

اساس کار این فانکشن ژنراتور بر یک میکرو کنترلر می باشد که تقریباً تمامی وظایف از جمله گرفتن نوع سیگنال و میزان فرکانس و نیز آنالیز آنها و انجام محاسبات لازم و در نهایت ایجاد هر کدام از شکل موج ها در ماژولی مشخص را به عهده دارد. باقی اجزاء مداری وظیفه کمک رسانی و تکمیل کار های میکروکنترلر را دارا هستند. به عنوان مثال دک ساخته شده از مقاومت ها اعداد دودویی را که میکرو تولید می کند گرفته و آنها را به سیگنالهای الکتریکی تبدیل می کند.

فصل اول: شرح و کاربرد پروژه

این پروژه یکی از انواع فانکشن ژنراتورها می باشد که برای مقاصد خاصی از آن استفاده می شود. فانکشن ژنراتورهای دیگری هم موجود می باشند که نسبت به این پروژه ارزان تر می باشند و در رنج و سیعتری فعالیّت می کنند. ولی مضیتی که موجب استفاده این فانکشن ژنراتور می شود انعطاف پذیری بیشتر و شخصی سازی های آن و نیز طرح های توسعه آن می باشد که هر شخص میتواند مطابق سلیقه خود انجام دهد.

موارد مصرف و کاربرد این پروژه تقریبا می توان گفت که در همه جا است و در هر کجا که سیگنال الکتریکی به کار آید از این وسیله می توان استفاده کرد به طور مثال اگر یک تعمیر کار تلویزیون این وسیله را داشته باشد و از طرح توسعه این محصول به گونه ای استفاده کرده باشد که برایش یک پترن بسازد، به راحتی میتواند این وسیله را جایگزین پترن ژنراتور خود کند ، چون این دستگاه قادر است چندین (تا چند هزار) شکل

موج خاص را ذخیره کند و در هر لحظه یکی از آنها را ایجاد کند .
مثال دیگری که از کاربرد های این پروژه می توان بیان کرد تولید دقیق سیگنال های خاص می باشد به عنوان مثال اگر از این وسیله در مطب ها، بیمارستان ها و اتاق های جراحی استفاده شود این دستگاه قادر خواهد بود که سیگنال مغز یا قلب بیمار را ذخیره کرده و آن را به سرعت تقلید کند و به صورت مداوم آن را تولید کند.(این عمل می تواند فواید خاص خود را داشته باشد از جمله اعمال سیگنال های شوک الکتریکی ، خون رسانی طبیعی با استفاده از دستگاه پمپاژ خونی که سیگنالش را از این پروژه دریافت می کند و یا انجام پروژه های تحقیقی در دانشگاهها و مراکز پزشکی به وسیله این پروژه). این پروژه همچنین قابلیت استفاده عمومی در مراکز آموزشی الکترونیکی به عنوان یک فانکشن ژنراتور معمولی را نیز دارد.
همان طور که می دانید فانکشن ژنراتور ها دستگاه هایی می باشند که در سطوح مختلف فرکانسی (و یا ولتاژی) سیگنال های مختلفی تولید می کنند. فانکشن ژنراتور ها جایگاه خاصی در مدارات ایفا می کنند که از این کاربردها میتوان به استفاده در مداراتی که به هر نحو به مخابرات و ارتباط های مخابراتی سر و کار دارند(مانند تلویزیون ها ، رادیوها، موبایل ها و سیستم های تلفنی) اشاره کرد. این فانکشن ژنراتورها بر حسب مدل ساخته شدنشان به انواع مختلفی تقسیم می شوند.
فانکشن ژنراتورهایی که به وسیله آی سی های از پیش طراحی شده ساخته می شوند (و اغلب در آنها از آی سی 565 استفاده شده است) دارای کیفیت خوبی (از نظر پایداری فرکانسی- ولتاژ ثابت در اغلب فرکانس ها و از نظر ایجاد رنج وسیعی از فرکانسها و سیگنال ها) می باشند. قیمت آنها نیز در مقایسه با این کیفیت کار بسیار مناسب می باشد.
ولی اگر هزینه کمتری مدّنظر باشد می توان یک فانکشن ژنراتور دست ساز را طبق نقشه های موجود درست کرد. این فانکشن ژنراتور ها بر اصل شارژ و دشارژینگ خازن ها و سوئچینگ (قطع و وصل شدن) ترانزیستور ها عمل می کنند و کیفیت بسیار پایینی را ازخود نشان می دهند.

اما فانکشن ژنراتوری که در این پروژه استفاده شده است از یک میکروکنترلر به عنوان هسته مرکزی خود سود برده است که قابلیت مانور در نوع سیگنال های خواسته شده را میسر می کند.
هر چند این فانکشن ژنراتورها در فرکانس های بالا و موج های معمولی که توسط دیگر دستگاهها به راحتی ساخته می شوند، ضعیف عمل می کند و خروجی اش را به صورت پلّه پلّه نشان می دهد ولی در سیگنال های غیر معمولی (مثل سیگنال قلب در انسان) و در فرکانس های پایین بسیار مناسب است.
البته چون خروجی میکروکنترلر دودویی بوده و برای انسان قابل فهم نیست در این پروژه از یک عدد مبدل دیجیتال به آنالوگ یا همان ((دی تو اِی)) استفاده شده است که خروجی را به صورت آنالوگ در می آورد و می توان از آن در کارهای عملی استفاده کرد و دیتا شیت آن نیز در صفحات بعدی آورده شده است.
مضیتی که این پروژه نسبت به فانکشن ژنراتور های دیگر دارد این است که در این پروژه از یک کیبورد استفاده شده است که این قابلیت را به وجود می آورد که کاربر عدد دقیق فرکانسی که مدنظر دارد را مستقیماً به صفحه کلید وارد کرده و خروجی را به طور دقیق مشاهده کند.(مثلاً عدد 12365 هرتز).
در این پروژه از یک ال سی دی نیز استفاده شده است که مدل سیگنال درخواستی و فرکانس وارد شده به صفحه کلید را به منظور پیش گیری از خطا نمایش می دهد.
هر چند این فانکشن ژنراتور در فرکانس پایینی فعالیت می کند و در ماکزیمم فرکانس خود شکل پله پله ای را ایجاد می کند ولی در کل برای موارد خاص ساخته شده است و در آن **کاربردهای** غیر عمومی عملکرد خوبی را دارا است.
((کلیات عملکرد فانکشن ژنراتور))
عملکرد این پروژه به این صورت است که میکروکنترلر به عنوان هسته مرکزی دستگاه، خروجی یکی(در قسمت توسعه ممکن است دو یا سه عدد) از پورت های خود را طبق مدل سیگنال مربعی ، مثلثی، دندان ارّه ای یا سینوسی ردیابی می کند و آن را کاهش و یا افزایش می دهد البته این عملکرد برای ایجاد یک فرکانس از هر مدل سیگنال به کار می رود.
برای داشتن رنجی از فرکانس احتیاج به سلکتور یا صفحه کلید می باشد که در اینجا صفحه کلید ترجیح داده شده است.

میکروکنترلر وظیفه دارد صفحه کلید را اسکن و رفرش کرده و هر بار که کلید جدیدی زده شد. فرکانس خواسته شده را محاسبه و یادداشت کند و هنگامی که کلید اینتر زده شد منتظر گرفتن نوع سیگنال شود تا با گرفتن آن به سراغ ماژول های نوشته شده در برنامه که هر کدام مختص یک نوع سیگنال می باشد برود و با داشتن میزان فرکانس و محاسبه میزان پریود و میزان و دفعات پر شدن تایمرها مشغول به تولید سیگنال ها شود.

(((توسعه)))

در این قسمت قصد داریم طرح ها و ایده هایی که می توان بر این پروژه اضافه و پیاده کرد را ارائه کنیم.

اولین ایده ای که برای توسعه در نظر گرفته شده است اضافه کردن مدل های دیگری از سیگنال های تولیدی می باشد.

شاید یک استفاده کنند از این دستگاه بخواهد سیگنال قلب را برای مصارف پزشکی بازسازی کند لذا در این طرح این قابلیت ایجاد شده است.

راهکار ایجاد موج دلخواه ازاین دستگاه:

ابتدا آن سیگنالی را که میخواهید برایتان تولید شود را روی یک کاغذ میلی متری (تقسیم شده) کشیده و آن را طبق زمان (در محور افقی) رسم کنید و سپس زمان ها را بر روی محور افقی به قسمت های یک میکروثانیه ای تقسیم کنید و در نهایت میزان ولتاژهای مشابه هر کدام از این قسمت ها را مشاهده و یادداشت نمایید.

جدولی تهیه کنید که دو ردیف سطر و به تعداد قسمتهای زمانی سیگنالتان ستون داشته باشد. در سطر بالایی اعداد 1 تا عدد آخرین قسمت زمانی و در سطر پایینی ولتاژ های مربوط به هرکدام را یادداشت کنید.

البته اگر سیگنال شما شامل مقادیر منفی نیز می شود شما باید آن را در برنامه به بالا برده (شیفت بدهید) و سپس در دنیای فیزیکی آن را به وسیله ی یک شیفت دهنده ولتاژ به پایین (به میزانی که بالا برده اید) شیفت دهید. مثلاً اگر سیگنال شما شامل ولتاژ های منفی سه تا مثبت سه می شود شما به کلیه مقادیر سه عدد اضافه کنید تا مقادیر از صفر تا 6 بشوند و سپس در مدارتان ولتاژ ها را سه ولت به پایین شیفت دهید.

نکته دیگر اینکه، میکروکنترلر8051 فقط قابلیت تقسیم مقادیر زمانی در 1 میکرو ثانیه را دارا می باشد، اما با ای وی آرمی توانید تقسیم بندی ها را افزایش دهید و اگر باز هم دقت بیشتری مدَنظر بود باید از کامپیوتر استفاده کنید. تقسیم بندی کمتر باعث می شود که شما شکل خروجی خود را بسیار کم دقت تر از آن چیز که می خواستید داشته باشید.

در بعضی از موارد می توان هنگامی که خروجی کم دقت داریم (بمنظور حالت پله پله شدن خروجی ان هم در فرکانس های بالا است) از یک عدد خازن جهت صاف کردن شکل موج خروجی استفاده کرد که البته این عمل مضرات خاص خود را دارا می باشد و در اکثر موارد جوابگو نیست.

توسعه دیگری که برای این دستگاه می توان در نظر گرفت این است که آنرا چند منظوره (و حتی همه منظوره) کرد. با کمی دستکاری در برنامه این پروژه آن را اسیلوسکوپ، اهم متر ،فارادمتر، آمپرمتر، ولت متر، هانری متر و فرکانس متر تغییر داد که البته در بعضی از موارد کمی اضافه کردن قطعه لازم است که آن را به یک دستگاه همه منظوره تبدیل می کند. آنچه به عنوان هسته مرکزی یک فانکشن ژنراتور لازم است همه در این پروژه رعایت شده است و شما کمی باید آن را مطابق سلیقه خود شخصی سازی کنید. مثلاً اگر شما دوست دارید که یک اسیلوسکوپ داشته باشید باید یک ال سی دی گرافیکی به آن اضافه کنید و برنامه را مطابق این ایده پیش ببرید که هر سیگنالی که در ورودی دریافت می شود به داده تبدیل شده و مطابق یک رنج ثابت زمانی (بالاتر از یک میکرو) نسبت به پایین ترین خط ال سی دی (که همان محور زمان در نظر گرفته شده است) با ارتفاعی به اندازه داده ای که از سیگنال بدست آمده است نمایش داده شود.

توسعه دیگر این است که از دکمه *2 که روی صفحه کیبورد تعبیه شده است به گونه ای دیگر استفاده شود. در حال حاضر این دکمه کاملاً به صورت نرم افزاری عمل کرده و عدد وارد شده به کیبورد (توسط کاربر) را در دو ضرب می کند. شما می توانید این دکمه را به مدارات ضرب کننده در فرکانس وصل کنید و با هر بار فشردن دکمه *2 هر بار یکی از این مدارات ضرب کننده در دو را در خروجی روشن کنید.

راه دیگر این است که به آی سی ضرب کننده فرکانس یک سلکتور وصل کنید و دو عدد از آنها را با هم ترکیب کنید تا بتوانید سیگنال ورودی آن را 100 بار از لحاظ فرکانس بزرگ کنید، در مرحله بعد هم این سلکتور(دیجیتالی را که به وسیله چند ترانزیستور ساخته شده است) را به میکرو وصل کنید تا میکرو بتواند با یک فرستادن به هر کدام از این ترانزیستورها عمل ضرب را کنترل کند. شما باید نام دکمه ضرب در دو را به دکمه ضرب تغییر دهید و برای فشرده شدن آن نیز برنامه کوچک مجزایی بنویسید به این صورت که بعد از زدن این دکمه میکرو دو عدد رادریافت کند، به ازای عدد(رقم 0 تا 9) اول یکی از ده سلکتور های وصل شده به آی سی ضرب کننده فرکانس را یک کند و به ازای عدد دوم یکی از ده سلکتوری که به آی سی دوم وصل شده است را .

توسعه دیگر و یا بهتری که میتوان انجام داد این است که نقش میکرو را از هسته اصلی به ناظر و هادی تغییر داد و آن نقش را به آی سی 565 داد.

همان ایده قبلی را دنبال کنید و فقط به آن یک عدد از این آی سی ها اضافه کنید،(این آی سی ها سه ورودی دارد که با یک کردن از هر کدام از این ورودی ها یکی از سه موج سینوسی – مربعی و مثلثی تولید خواهد شد)، و به میکرو وظیفه یک کردن یکی از این ورودی ها را بدهید(و البته وظیفه قبلی که استفاده از آی سی ضرب کننده فرکانس بود هم بر آن نگه دارید)، در این آی سی 565 یک قابلیتی طراحی شده است که مطابق ولتاژ ورودی فرکانس را کم و زیاد می کند، شما می توانید رابطه ولتاژ و فرکانس آن را از شرکت سازنده (توسط اینترنت)گرفته و رابطه معکوس آن را در برنامه میکروکنترلر خود قرار دهید و از سوی دیگر یکی از پورت های خروجی میکرو را به مبدل دیجیتال به آنالوگ وصل کنید و خروجی آن را به ورودی ولتاژ آی سی 565 وصل کنید.(اگر این پروژه برایتان سخت است به ورودی 565 یک ولتاژ ثابت متصل کنید و آن را ضرب کنید). البته یکی از مضراتی که این طرح دارد این است که دیگر شما نمی توانید هر نوع سیگنالی را تولید کنید ولی در عوض رنج کار فرکانسی شما بسیار با این آی سی افزایش پیدا می کند.

اگر شما می خواهید یک اهم متر از این پروژه تولید کنید شما باید یک بار جریان و بار دیگر ولتاژ دو سر مقاومت واقع بر ورودی را کنترل کنید و در انتها آنها را بر هم تقسیم کنید و میزان مقاومت را در خروجی به گونه ای نمایش دهید.
توسعه را می توانید در موارد دیگر از جمله آمپر متر و ولت متر تکرار کنید . اگر فارادمتر می خواهید ولتاژ و جریان آن را در فرکانس های مختلف یادداشت کنید(شما حتی می توانید این منحنی را نمایش دهید) و در نهایت با مقایسه آن با یک منحنی ثابت به میزان فاراد آن خازن پی برده نمایش دهید. توسعه های بسیار زیاد دیگری نیز می توان بر روی این پروژه انجام داد ولی مهمترین و ارزان ترین های آن فقط ارائه گردید.

فصل چهارم :راههای رسیدن به برنامه نهایی پروژه.

برای دستیابی به برنامه ی نهایی چهار برنامه نوشته شده است که در زیرهر یک را توضیح می دهیم.

توضیحاتی در مورد برنامه ی اول:

در اولین برنامه که تولید دو شکل موج توسط دو DACهشت بیتی است . ابتدا نحوه ی تولید شکل موج سینوسی را توضیح می دهیم.

برای تولید موج سینوسی توسط آی سی های مبدل ما باید مقادیرمتناظر با ولتاژ لحظه ای این موج را توسط میکروکنترولر ایجاد کرده وبه آی سی مبدل بدهیم تا در خروجی موج سینوسی داشته باشیم

البته بدلیل اینکه این آی سی می تواند 256حالت داشته باشد موج ما سینوسی کامل نیست اما بسیار شبیه است.

مقدارلحظه ای یک شکل موج سینوسی در واحد باینری با فرمول زیر محا سبه می شود:

((sin((x/255)*(2*PI))+1)/2)*255

بدلیل اینکه محا سبه این فرمول در هنگام اجرای برنامه میکروکنترولر فرایندی زمان بر است وما برای ایجاد یک موج

دقیق تر احتیاج به سرعت بالا داریم .به همین منظور این مقادیر از قبل حساب کرده وداخل میکروکنترولر قرار می دهیم.
به این گونه اطلاعات جداول لوک آپ گفته می شود.
بدلیل اینکه اعداد باینری از صفر شروع می شود لذا ما مجبوریم از عدد 128که نصف 256می باشد برای حالت صفر موج استفاده کنیم.
تا اعداد پایینتر برای سیکل منفی واعداد بالا تر برای سیکل مثبت این موج مورد استفاده قرار بگیرد.
فرمول فوق نیز اعداد را طبق خواسته ی ما بدست می آورد.
در زیر نمونه ای از جدول را آورده ایم.

{
128 , 129 , 129 , 130 , 131 , 131 , 132 , 133 , 133 , 134 , 134 , 135 , 136 , 136 , 137 ,
137 , 138 , 139 , 139 , 140 , 140 , 141 , 141 , 142 , 142 , 143 , 143 , 144 , 144 , 145 ,
145 , 146 , 146 , 146 , 147 , 147 , 148 , 148 , 148 , 149 , 149 , 149 , 150 , 150 , 150 ,
151 , 151 , 151 , 151 , 152 , 152 , 152 , 152 , 152 , 152 , 153 , 153 , 153 , 153 , 153 ,
153 , 153 , 153 , 153 , 153 , 153 , 153 , 153 , 153 , 153 , 153 , 153 , 152 , 152 , 152 ,
152 , 152 , 152 , 151 , 151 , 151 , 151 , 150 , 150 , 150 , 149 , 149 , 149 , 148 , 148 ,
148 , 147 , 147 , 146 , 146 , 146 , 145 , 145 , 144 , 144 , 143 , 143 , 142 , 142 , 141 ,
141 , 140 , 140 , 139 , 139 , 138 , 137 , 137 , 136 , 136 , 135 , 134 , 134 , 133 , 133 ,
132 , 131 , 131 , 130 , 129 , 129 , 128 , 128 , 127 , 126 , 126 , 125 , 124 , 124 , 123 ,
123 , 122 , 121 , 121 , 120 , 120 , 119 , 118 , 118 , 117 , 117 , 116 , 116 , 115 , 114 ,
114 , 113 , 113 , 112 , 112 , 111 , 111 , 110 , 110 , 110 , 109 , 109 , 108 , 108 , 107 ,
107 , 107 , 106 , 106 , 106 , 105 , 105 , 105 , 105 , 104 , 104 , 104 , 104 , 103 , 103 ,
103 , 103 , 103 , 103 , 102 , 102 , 102 , 102 , 102 , 102 , 102 , 102 , 102 , 102 , 102 ,
102 , 102 , 102 , 102 , 103 , 103 , 103 , 103 , 103 , 103 , 104 , 104 , 104 , 104 , 105 ,
105 , 105 , 105 , 106 , 106 , 106 , 107 , 107 , 107 , 108 , 108 , 109 , 109 , 110 , 110 ,
110 , 111 , 111 , 112 , 112 , 113 , 113 , 114 , 114 , 115 , 116 , 116 , 117 , 117 , 118 ,
118 , 119 , 120 , 120 , 121 , 121 , 122 , 123 , 123 , 124 , 124 , 125 , 126 , 126 , 127 ,
128
}

در این برنامه چون ما باید دامنه رانیز کنترل کنیم لذا از پنج جدول لوک آپ با تفاوت 20 درصدی استفاده کردیم.

ودر هنگام باز خوانی از جدول با توجه به دامنه ی تنظیم شده توسط کاربر از جدول منا سب استفاده می شود.
همانطور که گفته شد هدف ما از این برنامه ساخت دو موج سینوسی با قابلیت تنظیم فرکانس ، فازو دامنه موج است.
تعین فاز که در این برنامه بر اساس درجه ی آن می باشد و از 0تا359درجه را ایجاد می کند.
هر سیکل کامل که در جداول لوک آپ وارد شده است شامل 256 حالت است که از0تا 256 می باشد.
یعنی برای یک سیکل کامل ما 256 نمونه را به مبدل انالوگ به دیجیتال می دهیم.
بنابراین برای تعین اختلاف فاز نیز یعنی تفاوت در این 256 نمونه
در برنامه ما دو متغیر
(
unsigned char ph1 = 0;
unsigned char ph2 = 0;
)
از نوع 8بیتی بدون علامت عهده دار چرخش داخل این جداول لوک آپ هستند و مقدار متناظر با خانه ای که این متغیر ها بدان اشاره می کنند به بدل مربوط، ارسال می شود.
در حالتی که اختلاف فاز صفر درجه باشد مقادیر این دو متغیر باهم یکسان هستند.
چون اختلاف فاز ما 360 درجه است وجدول لوک آپ256 مورد دارد بنابرین مقادیر اختلاف فاز در (256/360) ضرب شود تا عدد متناظر با اختلاف فاز بدست آید.
در هر بار که مقدار این اختلاف فاز تغییر می کند مقدار Ph1برابر صفر می شود وPh2 نیز مقدار درجه اختلاف فاز در 0.77777که معادل 256/360 است ضرب می کنیم
برای دقیق بودن زمان فرستادن نمونه ها به مبدل از وقفه ی تایمر یک استفاده می کنیم.
هنگامی که مقدار وقفه سرریز می شود روتین وقفه تایمر فراخوانی می شود ودستوراتی که داخل این روتین نوشته شده اجرا می شود

برای تنظیم فرکانس موج مورد نظر باید مقدارتایمر را تغییر دهیم تا در وقفه سرریزی تایمر طبق زمانبندی ما رخ دهد
کدی که در داخل روتین وقفه تایمر نوشته ایم کد ساده ای می باشد که در زیر می بینیم

```
TCNT1=T1;
dac1 = source[Psec][ph1];
dac2 = source[Psec][ph2];
ph1+=2;
 ph2+=2;
```

TCNT1که در خط اول مقدار دهی شده است رجیستر مقدار 16 بیتی تایمر یک می باشد.
با مقدار دهی این رجیستر است که می توانیم فرکانس موج را تنظیم کنیم.
متغیر T1 نیز مقدارمناسب برای فرکانس تعین شده را در خود نگه داری کند.
دو مولفه ای که که در خط دوم وسوم این دستورات مقدار دهی شده اند دو پورت از میکروکنترولر 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 PORTAاز میکرو متصل شده وپایه های دیتای مبدل دوم نیز بهPORTD میکروی ما متصل است

کار با ال سی دی با کامپایلر کدویژن بسیار ساده است.
برای راحتی کار می توانیم توسط ابزار کدویزارد به تب ال سی دی استفاده شده است.
و توسط دستوراتی از قبیل Lcd_putsf و Lcd_puts اطلاعات خود را روی صفحه
ال سی دی نمایش داده ایم.

در این برنامه از یک صفحه کلید 4*4 استفاده شده که طریقه ی خواندن کلید فشرده شده در ان به اصطلاع روش سر کشی گفته می شود.

ما برای این کار کتابخوانه ای طراحی کردیم تا بتوانیم در برنامه های دیگر نیز براحتی از صفحه کلید استفده کنیم.

حال به شرح مختصری راجع به این کتابخوانه می پردازیم:

این کتابخوانه کلادو تابع دارد

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

تابع اول برای مقدار دهی اولیه به پورت مورد نظر وتعین ورودی وخروجی بودن پین های پورت مربوطه به کار رفته است.

```
void keypad_init(void)
{
keypad_DDR=0x0F;
 keypad_PORT=0xF0;
}
```

*نکته مهمی که که باید به آن توجه کنیم این است که برای استفاده از این کتابخوانه باید در برنامه خود سه ماکروی زیر را بهاین 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 کدکلید فشرده شده از ان استفاده میکنیم.

ابتدا به نحوه اتصال کی پد به پورت میکرو توجه می کنیم.

نحوه کار به این شکل است که ابتدا 4پین از میکرو که به سطر های صفحه کلید به صورت خروجی تعریف می شود و4 پین دیگر به سطر ها وصل است به عنوان ورودی .

همینطور مقدار اولیه ورودی ها نیز ست می شود تا 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 پین ها ی ورودی برابر صفر باشد. یعنی کلیدی فشرده شده است.
ما با خواندن از پین های ورودی می توانیم سطری که کلیدفشرده شده در ان قرار دارد را مشخص کنیم.
برای مشخص شدن ستون ودر نهایت کلید فشرده شده اینبار 4 پینی که به سطر ها متصل است را ورودی تعریف می کنیم وبا مقدار دهی وخواندن از پورت می توانیم سطر مورد نظر ودر نهایت کلید فشرده شده را پیدا می کنیم.
به این روش سر کشی می گویند.

توابعی که در برنامه اصلی هستند به شرح زیر می باشد.

1.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 وماکزیمم تعین شده در ورودی های این تابع است.

2. void lcd_panel1(void)

که برای نمایش اطلاعات فرکانس و اختلاف فاز و... روی صفحه ی ال سی دی و فعال شدن سرویس وقفه ها به کار
این زیر برنامه نما یش صفحه ی اولیه ال سی دی است و همچنین مقدار هر یک را طبق جدول های پایین انتخاب می کند.

3. void panel2(void) // change freq

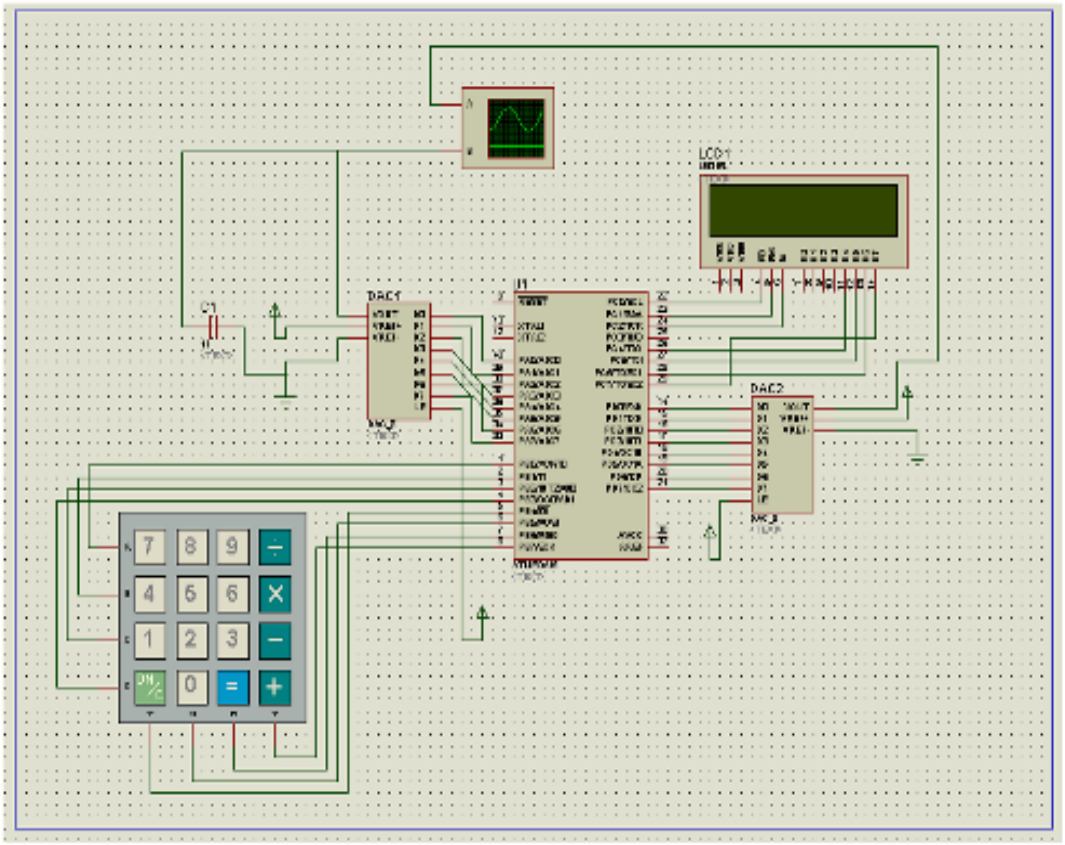
این تابع را هنگامی فراخوانی می کنیم که کلید فانکشن اول از کی پد فشرده شده باشد کار این تابع این است که فرکانس موج ما را طبق درخواست کاربر وتوسط صفحه کلید تعین می کنیم.

4. void panel3(void) // change phase

این تابع را هنگامی فراخوانی می کنیم که کلید فانکشن دوم از کی پد فشرده شده باشد کار این تابع این است که اختلاف فاز بین دو موج ما را طبق درخواست کاربر وتوسط صفحه کلید تعین می کنیم.

5. void panel3(void) // change amp

این تابع را هنگامی فراخوانی می کنیم که کلید فانکشن سوم از کی پد فشرده شده باشد کار این تابع این است که دامنه ی موج ما را طبق درخواست کاربر وتوسط صفحه کلید تعین می کنیم.
در زیر نمای کامل این قسمت از پروژه را آورده ایم:



برنامه ی اول:

```
#include <mega16.h>
#include <stdlib.h>
```

```
#include <delay.h>

#define keypad_DDR DDRB
#define keypad_PORT PORTB
#define keypad_PIN PINB
#define key_ctr 5
#define key_delay 1



unsigned char const source[5][256] = {
{
128 , 129 , 129 , 130 , 131 , 131 , 132 , 133 , 133 , 134 , 134 , 135 , 136 , 136 , 137 , 137 , 138
, 139 , 139 , 140 , 140 , 141 , 141 , 142 , 142 , 143 , 143 , 144 , 144 , 145 , 145 , 146 , 146 ,
146 , 147 , 147 , 148 , 148 , 148 , 149 , 149 , 149 , 150 , 150 , 150 , 151 , 151 , 151 , 151 , 152
, 152 , 152 , 152 , 152 , 152 , 153 , 153 , 153 , 153 , 153 , 153 , 153 , 153 , 153 , 153 , 153 ,
153 , 153 , 153 , 153 , 153 , 153 , 152 , 152 , 152 , 152 , 152 , 152 , 151 , 151 , 151 , 151 , 150
, 150 , 150 , 149 , 149 , 149 , 148 , 148 , 148 , 147 , 147 , 146 , 146 , 146 , 145 , 145 , 144 ,
144 , 143 , 143 , 142 , 142 , 141 , 141 , 140 , 140 , 139 , 139 , 138 , 137 , 137 , 136 , 136 , 135
, 134 , 134 , 133 , 133 , 132 , 131 , 131 , 130 , 129 , 129 , 128 , 128 , 127 , 126 , 126 , 125 ,
124 , 124 , 123 , 123 , 122 , 121 , 121 , 120 , 120 , 119 , 118 , 118 , 117 , 117 , 116 , 116 , 115
, 114 , 114 , 113 , 113 , 112 , 112 , 111 , 111 , 110 , 110 , 110 , 109 , 109 , 108 , 108 , 107 ,
107 , 107 , 106 , 106 , 106 , 105 , 105 , 105 , 105 , 104 , 104 ,
104 , 104 , 103 , 103 , 103 , 103 , 103 , 103 , 102 , 102 , 102 , 102 , 102 , 102 , 102 , 102 , 102
, 102 , 102 , 102 , 102 , 102 , 103 , 103 , 103 , 103 , 103 , 103 , 104 , 104 , 104 , 104 ,
105 , 105 , 105 , 105 , 106 , 106 , 106 , 107 , 107 , 107 , 108 , 108 , 109 , 109 , 110 , 110 , 110
, 111 , 111 , 112 , 112 , 113 , 113 , 114 , 114 , 115 , 116 , 116 , 117 , 117 , 118 , 118 , 119 ,
120 , 120 , 121 , 121 , 122 , 123 , 123 , 124 , 124 , 125 , 126 , 126 , 127 , 128 }, //-------%20---
----
{
129 , 130 , 131 , 132 , 134 , 135 , 136 , 138 , 139 , 140 , 141 , 142 , 144 , 145 , 146 , 147 , 148
, 150 , 150 , 152 , 153 , 154 , 155 , 156 , 157 , 158 , 159 , 160 , 161 , 162 , 163 , 164 , 164 ,
165 , 166 , 167 , 168 , 168 , 169 , 170 , 171 , 171 , 172 , 172 , 173 , 174 , 174 , 175 , 175 , 176
, 176 , 176 , 177 , 177 , 177 , 178 , 178 , 178 , 178 , 178 , 178 , 178 , 178 , 178 , 178 , 178 ,
178 , 178 , 178 , 178 , 178 , 178 , 177 , 177 , 177 , 176 , 176 , 176 , 175 , 175 , 174 , 174 , 173
, 172 , 172 , 171 , 171 , 170 , 169 , 168 , 168 , 167 , 166 , 165 , 164 , 164 , 163 , 162 , 161 ,
160 , 159 , 158 , 157 , 156 , 155 , 154 , 153 , 152 , 150 , 150 , 148 , 147 , 146 , 145 , 144 , 142
```

, 141 , 140 , 139 , 138 , 136 , 135 , 134 , 132 , 131 , 130 , 129 , 128 , 126 , 125 , 124 , 123 ,
121 , 120 , 119 , 118 , 116 , 115 , 114 , 113 , 112 , 110 , 109 , 108 , 107 , 106 , 105 , 104 , 102
, 101 , 100 , 99 , 98 , 97 , 96 , 95 , 94 , 93 , 92 , 92 , 91 , 90 , 89 , 88 , 87 , 87 , 86 ,
85 , 84 , 84 , 83 , 83 , 82 , 82 , 81 , 80 ,
80 , 80 , 79 , 79 , 78 , 78 , 78 , 78 , 77 , 77 , 77 , 77 , 77 , 77 , 77 , 77 , 77 , 77 ,
77 , 77 , 77 , 77 , 78 , 78 , 78 , 78 , 79 , 79 , 80 , 80 , 80 , 81 , 82 , 82 , 83 , 83 , 84 ,
84 , 85 , 86 , 87 , 87 , 88 , 89 , 90 , 91 , 92 , 92 , 93 , 94 , 95 , 96 , 97 , 98 , 99 , 100
, 101 , 102 , 104 , 105 , 106 , 107 , 108 , 109 , 110 , 112 , 113 , 114 , 115 , 116 , 118 , 119 ,
120 , 121 , 123 , 124 , 125 , 126 , 128 }, //-------%40-------

{
130 , 131 , 133 , 135 , 137 , 139 , 141 , 143 , 144 , 146 , 148 , 150 , 152 , 154 , 155 , 157 , 159
, 161 , 162 , 164 , 166 , 167 , 168 , 170 , 172 , 173 , 174 , 176 , 178 , 179 , 180 , 182 ,
183 , 184 , 186 , 187 , 188 , 189 , 190 , 191 , 192 , 193 , 194 , 195 , 196 , 197 , 198 , 198 , 199
, 200 , 200 , 201 , 202 , 202 , 202 , 203 , 203 , 203 , 204 , 204 , 204 , 204 , 204 , 204 ,
204 , 204 , 204 , 204 , 204 , 203 , 203 , 203 , 202 , 202 , 202 , 201 , 200 , 200 , 199 , 198 , 198
, 197 , 196 , 195 , 194 , 193 , 192 , 191 , 190 , 189 , 188 , 187 , 186 , 184 , 183 , 182 ,
180 , 179 , 178 , 176 , 174 , 173 , 172 , 170 , 168 , 167 , 166 , 164 , 162 , 161 , 159 , 157 , 155
, 154 , 152 , 150 , 148 , 146 , 144 , 143 , 141 , 139 , 137 , 135 , 133 , 131 , 130 , 128 ,
126 , 124 , 122 , 121 , 118 , 116 , 114 , 113 , 111 , 109 , 107 , 106 , 104 , 102 , 100 , 98 , 96 ,
95 , 94 , 92 , 90 , 88 , 87 , 85 , 84 , 82 , 81 , 79 , 78 , 76 , 75 , 74 ,
72 , 71 , 70 , 68 , 67 , 66 , 65 , 64 , 63 , 62 , 61 , 60 , 59 , 59 , 58 , 57 , 56 , 56 , 55 ,
54 , 54 , 53 , 53 , 53 , 52 , 52 , 52 , 52 , 52 , 52 , 52 , 52 ,
52 , 52 , 52 , 52 , 52 , 52 , 52 , 53 , 53 , 53 , 54 , 54 , 55 , 56 , 56 , 57 , 58 , 59 , 59 ,
60 , 61 , 62 , 63 , 64 , 65 , 66 , 67 , 68 , 70 , 71 , 72 , 74 ,
75 , 76 , 78 , 79 , 81 , 82 , 84 , 85 , 87 , 88 , 90 , 92 , 94 , 95 , 96 , 98 , 100 , 102 ,
104 , 106 , 107 , 109 , 111 , 113 , 114 , 116 , 118 , 121 , 122 , 124 , 126 , 128
}, //-------%60-------

{
130 , 133 , 135 , 138 , 141 , 143 , 146 , 148 , 150 , 153 , 155 , 158 , 160 , 162 , 165 , 167 , 170
, 172 , 174 , 176 , 178 , 181 , 182 , 185 , 186 , 189 , 190 , 193 , 194 , 197 , 198 , 200 ,
202 , 203 , 206 , 207 , 209 , 210 , 211 , 213 , 214 , 215 , 217 , 218 , 219 , 220 , 222 , 222 , 223
, 224 , 225 , 226 , 226 , 227 , 227 , 228 , 229 , 229 , 230 , 230 , 230 , 230 , 230 , 230 ,
230 , 230 , 230 , 230 , 230 , 229 , 229 , 228 , 227 , 227 , 226 , 226 , 225 , 224 , 223 , 222 , 222
, 220 , 219 , 218 , 217 , 215 , 214 , 213 , 211 , 210 , 209 , 207 , 206 , 203 , 202 , 200 ,

198 , 197 , 194 , 193 , 190 , 189 , 186 , 185 , 182 , 181 , 178 , 176 , 174 , 172 , 170 , 167 , 165
, 162 , 160 , 158 , 155 , 153 , 150 , 148 , 146 , 143 , 141 , 138 , 135 , 133 , 130 , 128 ,
126 , 123 , 121 , 118 , 115 , 113 , 110 , 108 , 106 , 103 , 101 , 98 , 96 , 94 , 91 , 89 , 86 , 84
, 82 , 80 , 78 , 75 , 74 , 71 , 70 , 67 , 66 , 63 , 62 , 59 , 58 , 56 ,
54 , 53 , 50 , 49 , 47 , 46 , 45 , 43 , 42 , 41 , 39 , 38 , 37 , 36 , 34 , 34 , 33 , 32 , 31 ,
30 , 30 , 29 , 29 , 28 , 27 , 27 , 26 , 26 , 26 , 26 , 26 , 26 ,
26 , 26 , 26 , 26 , 26 , 27 , 27 , 28 , 29 , 29 , 30 , 30 , 31 , 32 , 33 , 34 , 34 , 36 , 37 ,
38 , 39 , 41 , 42 , 43 , 45 , 46 , 47 , 49 , 50 , 53 , 54 , 56 ,
58 , 59 , 62 , 63 , 66 , 67 , 70 , 71 , 74 , 75 , 78 , 80 , 82 , 84 , 86 , 89 , 91 , 94 , 96 ,
98 , 101 , 103 , 106 , 108 , 110 , 113 , 115 , 118 , 121 , 123 , 126 , 128
}, //-------%80-------

{
131 , 134 , 137 , 140 , 144 , 147 , 150 , 153 , 156 , 159 , 162 , 165 , 168 , 171 , 174 , 177 , 180
, 183 , 185 , 188 , 191 , 194 , 196 , 199 , 201 , 204 , 206 , 209 , 211 , 214 , 216 , 218 ,
220 , 222 , 225 , 227 , 229 , 230 , 232 , 234 , 236 , 237 , 239 , 240 , 242 , 243 , 245 , 246 , 247
, 248 , 249 , 250 , 251 , 252 , 252 , 253 , 254 , 254 , 255 , 255 , 255 , 255 , 255 , 255 ,
255 , 255 , 255 , 255 , 255 , 254 , 254 , 253 , 252 , 252 , 251 , 250 , 249 , 248 , 247 , 246 , 245
, 243 , 242 , 240 , 239 , 237 , 236 , 234 , 232 , 230 , 229 , 227 , 225 , 222 , 220 , 218 ,
216 , 214 , 211 , 209 , 206 , 204 , 201 , 199 , 196 , 194 , 191 , 188 , 185 , 183 , 180 , 177 , 174
, 171 , 168 , 165 , 162 , 159 , 156 , 153 , 150 , 147 , 144 , 140 , 137 , 134 , 131 , 128 ,
125 , 122 , 119 , 116 , 112 , 109 , 106 , 103 , 100 , 97 , 94 , 91 , 88 , 85 , 82 , 79 , 76 , 73
, 71 , 68 , 65 , 62 , 60 , 57 , 55 , 52 , 50 , 47 , 45 , 42 , 40 , 38 ,
36 , 34 , 31 , 29 , 27 , 26 , 24 , 22 , 20 , 19 , 17 , 16 , 14 , 13 , 11 , 10 , 9 , 8 , 7 , 6
, 5 , 4 , 4 , 3 , 2 , 2 , 1 , 1 , 1 , 1 , 1 , 1 ,
1 , 1 , 1 , 1 , 1 , 2 , 2 , 3 , 4 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 13 , 14 , 16 , 17
, 19 , 20 , 22 , 24 , 26 , 27 , 29 , 31 , 34 , 36 , 38 ,
40 , 42 , 45 , 47 , 50 , 52 , 55 , 57 , 60 , 62 , 65 , 68 , 71 , 73 , 76 , 79 , 82 , 85 , 88 ,
91 , 94 , 97 , 100 , 103 , 106 , 109 , 112 , 116 , 119 , 122 , 125 , 128
}//------%100-------
};

#define D20 0
#define D40 1
#define D60 2
#define D80 3
#define D100 4

```
#define F1hz 125000
#define dac1 PORTA
#define dac2 PORTD
#define F1 14
#define F2 24
#define F3 34
#define _FF_ 0.71112

unsigned char const table[4][4]={{0,1,2,3},{4,5,6,7},{8,9,10,11},{12,13,14,15}};
// Alphanumeric LCD Module functions
#asm
   .equ __lcd_port=0x15 ;PORTC
#endasm
#include <lcd.h>
// Declare your global variables here
unsigned char Psec = D100;  //perSec Damane Freq
unsigned char ph1 = 0;      //zavie phase 1
unsigned char ph2 = 0;      //zavie phase 2
unsigned long T1;           //Timer Numeric
unsigned int frequnce = 100;
unsigned int radu = 0;
unsigned char scale = 100;
// Timer 1 overflow interrupt service routine

interrupt [TIM1_OVF] void timer1_ovf_isr(void)
{
// Reinitialize Timer 1 value
TCNT1=T1;
dac1 = source[Psec][ph1];
dac2 = source[Psec][ph2];
ph1+=2;
ph2+=2;

//TCNT1L=0xAA;
// Place your code here

}
```

```
// Declare your global variables here
unsigned char scan_key(void)
{
  unsigned char
i=0,j=0,key_ctr_temp=0,key_temp=0,scan_code[4]={0xFE,0xFD,0xFB,0xF7};

  for (i=0;i<4;i++)
  {
      keypad_PORT|=0x0F;
      keypad_PORT&=scan_code[i];
      delay_ms(5);
      //key_temp=keypad_PIN;
      if (keypad_PIN<0xF0)
      {
        key_temp=keypad_PIN;
        for(j=0;j<key_ctr;j++)
        {
          delay_ms(key_delay);
          if(keypad_PIN==key_temp)
            key_ctr_temp++;
        }
//          if(key_ctr_temp==key_ctr)
        {   key_ctr_temp = keypad_PIN;
            switch(key_ctr_temp)
            {
                  case 238:
                        return 1;
                        break;
                  case 237:
                        return 4;
                        break;
                  case 235:
                        return 7;
                        break;
                  case 231:
                        return 41;
```

```
        break;
    case 222:
        return 2;
        break;
    case 221:
        return 5;
        break;
    case 219:
        return 8;
        break;
    case 215:
        return 0;
        break;
    case 190:
        return 3;
        break;
    case 189:
        return 6;
        break;
    case 187:
        return 9;
        break;
    case 183:
        return 43;
        break;
    case 126:
        return 14;
        break;
    case 125:
        return 24;
        break;
    case 123:
        return 34;
        break;
    case 119:
        return 44;
        break;
```

```
                default:
                    return 255;
                    break;
            }
        }
      }
  }
  return 255 ;
}
unsigned int get_num(unsigned int min, unsigned int max)
{
unsigned int numb = 0;
unsigned char _num, _x = 0, _key;
_num = 0;
do
{
 lcd_gotoxy(_x, 1);
_key = scan_key();
if (_key != 255)
{
  delay_ms(5);
  _key = scan_key();
  if (_key != 255)
  {
    switch (_key)
    {
     case 1  : if (_x < 4) {_num = 1; numb = numb *10; numb += _num; lcd_putsf("1"); _x++;
delay_ms(150);} break;
     case 2  : if (_x < 4) {_num = 2; numb = numb *10; numb += _num; lcd_putsf("2"); _x++;
delay_ms(150);} break;
     case 3  : if (_x < 4) {_num = 3; numb = numb *10; numb += _num; lcd_putsf("3"); _x++;
delay_ms(150);} break;
     case 4  : if (_x < 4) {_num = 4; numb = numb *10; numb += _num; lcd_putsf("4"); _x++;
delay_ms(150);} break;
     case 5  : if (_x < 4) {_num = 5; numb = numb *10; numb += _num; lcd_putsf("5"); _x++;
delay_ms(150);} break;
```

```
    case 6  : if (_x < 4) {_num = 6; numb = numb *10; numb += _num; lcd_putsf("6"); _x++;
delay_ms(150);} break;
    case 7  : if (_x < 4) {_num = 7; numb = numb *10; numb += _num; lcd_putsf("7"); _x++;
delay_ms(150);} break;
    case 8  : if (_x < 4) {_num = 8; numb = numb *10; numb += _num; lcd_putsf("8"); _x++;
delay_ms(150);} break;
    case 9  : if (_x < 4) {_num = 9; numb = numb *10; numb += _num; lcd_putsf("9"); _x++;
delay_ms(150);} break;
    case 0  : if (_x < 4) {_num = 0; numb = numb *10; numb += _num; lcd_putsf("0"); _x++;
delay_ms(150);} break;
    case 43 : if (_x > 0) {_x--; lcd_gotoxy(_x, 1); lcd_putsf(" "); numb = numb / 10;
delay_ms(150);}; break;
    case 44 : if (numb < min) {numb = min; }; if (numb > max) {numb = max; }; break;
   }
   lcd_gotoxy(_x, 1);
  }
}
} while(_key != 44);
//numb = max-min;
return numb;
}
void lcd_panel1(void)
{
char s_s[5];
#asm("cli")
lcd_clear();
lcd_gotoxy(0,0);
lcd_putsf("Freq :      hz. ");
lcd_gotoxy(0, 1);
lcd_putsf("R :   . Y-G:%   ");
itoa(frequnce, s_s);
lcd_gotoxy(7, 0);
lcd_puts(s_s);
itoa(radu, s_s);
lcd_gotoxy(3, 1);
lcd_puts(s_s);
itoa(scale, s_s);
```

```
lcd_gotoxy(13, 1);
lcd_puts(s_s);
#asm("sei")
}
void panel2(void)          // change freq
{
#asm("cli")
lcd_clear();
lcd_gotoxy(0, 0);
lcd_putsf("2000 > Freq > 9");
frequnce = get_num(10, 1300);
ph1 = 0;
ph2 = radu * _FF_;
T1 = ((F1hz / frequnce) - 34)*-1;
TCNT1 = T1;
lcd_panel1();
}
void panel3(void)          // change phase
{
#asm("cli")
lcd_clear();
lcd_gotoxy(0, 0);
lcd_putsf("Phase R  (0~359)");
ph1 = 0;
radu = get_num(0, 359);
ph2 = radu * _FF_;
T1 = ((F1hz / frequnce) )*-1;
TCNT1 = T1;
lcd_panel1();
}
void panel4(void)          // change scale
{
#asm("cli")
lcd_clear();
lcd_gotoxy(0, 0);
lcd_putsf("Scale 1~5 * %20");
Psec = get_num(1, 5) - 1;
```

```
switch(Psec)
{
case 0 : scale = 20; break;
case 1 : scale = 40; break;
case 2 : scale = 60; break;
case 3 : scale = 80; break;
case 4 : scale = 100; break;
}
ph1 = 0;
ph2 = radu * _FF_;
T1 = ((F1hz / frequnce) - 34)*-1;
TCNT1 = T1;
lcd_panel1();
}
void main(void)
{
unsigned char keybd;
// Declare your local variables here
// Input/Output Ports initialization
// Port A initialization
// Func7=Out Func6=Out Func5=Out Func4=Out Func3=Out Func2=Out Func1=Out
Func0=Out
// State7=0 State6=0 State5=0 State4=0 State3=0 State2=0 State1=0 State0=0
PORTA=0x00;
DDRA=0xFF;
// Port B initialization
// Func7=Out Func6=Out Func5=Out Func4=Out Func3=Out Func2=Out Func1=Out
Func0=Out
// State7=0 State6=0 State5=0 State4=0 State3=0 State2=0 State1=0 State0=0
PORTB=0xF0;
DDRB=0x0F;
// Port C initialization
// Func7=Out Func6=Out Func5=Out Func4=Out Func3=Out Func2=Out Func1=Out
Func0=Out
// State7=0 State6=0 State5=0 State4=0 State3=0 State2=0 State1=0 State0=0
PORTC=0x00;
DDRC=0xFF;
```

```
// Port D initialization
// Func7=In Func6=In Func5=In Func4=In Func3=Out Func2=Out Func1=Out Func0=Out
// State7=T State6=T State5=T State4=T State3=0 State2=0 State1=0 State0=0
PORTD=0x00;
DDRD=0xFF;
// 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: 16000.000 kHz
// Mode: Normal top=FFFFh
// OC1A output: Discon.
// OC1B output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer 1 Overflow Interrupt: On
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
TCCR1A=0x00;
TCCR1B=0x01;
TCNT1H=0xAA;
TCNT1L=0xAA;
ICR1H=0x00;
ICR1L=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
MCUCR=0x00;
MCUCSR=0x00;
// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x04;
// Analog Comparator initialization
// Analog Comparator: Off
// Analog Comparator Input Capture by Timer/Counter 1: Off
ACSR=0x80;
SFIOR=0x00;
// LCD module initialization
lcd_init(16);
// Global enable interrupts
frequnce = 100;
T1 = ((F1hz / frequnce) -33)*-1;
lcd_panel1();
while (1)
    {
    keybd = scan_key();
    if (keybd != 255)
    {
      switch (keybd)
      {
         case F1 : panel2(); break;
         case F2 : panel3(); break;
         case F3 : panel4(); break;
```

```
    }
   }
//  }
   };
}
```

توضیحاتی در مورد برنامه ی دوم:

برای اینکه به برنامه ی نهایی برسیم باید چند شکل موج را روی خروجی ظاهر کنیم دراین جا با راهنمایی استاد راهنما، پروژه خود را تکمیل کردیم .

با تکمیل کردن پروژه یکی از خروجی ها را حذف، و به جای scale بندی از چندین فرمول برای هر شکل موج استفاده کردیم و برنامه همان برنامه ی قبلی می باشد.در برنامه ی موجود فقط قسمت هایی که تغییر کرده است توضیح داده می شود.

همان طور که مشاهده می کنید در برنا مه Scale ها حذف شده است و به جای آن از فرمولی برای تولید شکل موج سینوسی استفاده کرده ایم. در key pad یک کلید مخصوص تغییر شکل موج قرار داده ایم که نیازمند نوشتن زیر برنامه ی // void panel5(void) change WAVE می باشد.

در این زیر برنامه تنها چیزی که نیاز به توضیح دارد خط ; m=get_num(1, 5) است که بوسیله ی تابع get_num اعداد یک تا پنج را در m ریخته تا ازآن جهت انتخاب شکل موج استفاده کنیم و تابع void GenerateShape(void) که برنامه هر شکل موج را در آن نوشته شده است.

1- برنامه ی موج سینوسی :

```
for(teta=0;teta<=360;teta++)
{
if(amp>6.0)amp=1.0;
out[teta]=((0.2)*amp*120*sin(teta*2*3.141592/360.)+128);
}
```

خط اول فاز شکل موج می باشد که 360حالت برای این شکل موج دارد. خط دوم کنترل دامنه است که نمی گذارد دامنه ما بیشتر

از 5 ولت شود. خط سوم فرمول رابطه ی سینوسی است،که به ازای هر ورودی لحظه ای از شکل موج مان را برای ما مشخص می کند.

2- برنامه موج دندان اره ای :

```
for(teta=0;teta<=360;teta++)
{
if(amp>6.0)amp=1.0;
out[teta]= ((0.2)*amp*(teta*255/360.));
}
```

دررابطه فوق به ازای افزایش teta خروجی نیز به صورت صعودی افزایش پیدا می کند البته باید این را در نظر داشت که خروجی میکرو بیش از256 حالت نمی باشد و به همین خاطر در مقدار(255/360) ضرب می کنیم. و به همین خاطر وقتی teta=360 است خروجی میکرو ماکزیمم مقدار را دارد(طبق جدول تناسب).

3- برنامه موج مثلثی :

```
for(teta=0;teta<180;teta++)
{
if(amp>6.0)amp=1.0;
out[teta]=(0.2)*amp*(teta*255/180.);
}
for(teta=180;teta<=360;teta++)
{
if(amp>6.0)amp=1.0;
out[teta]=(0.2)*amp*((360-teta)*255/180.);
}
```

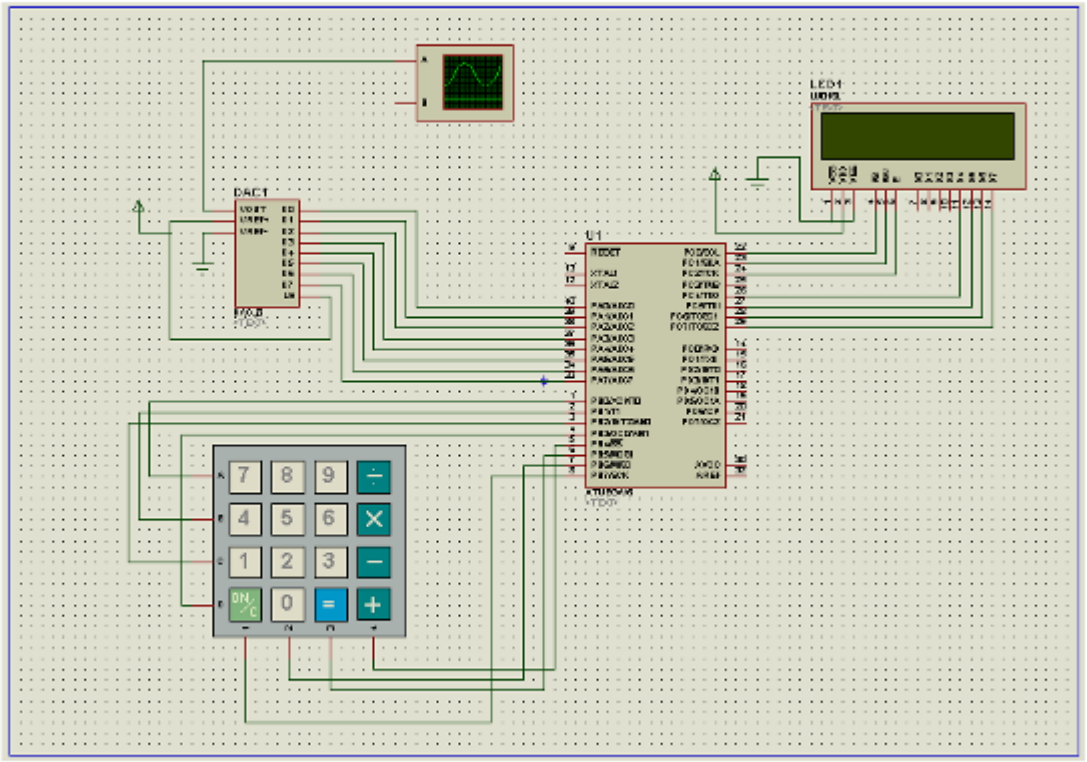
در رابطه ی بالا چون شکل موج مثلثی از دو قسمت صعودی و نزولی تشکیل شده است از دو حلقه ی for برای خروجی استفاده کرده ایم، که در قسمت اول بصورت خطی با افزایش teta خروجی نیز افزایش پیدا می کند تا به مقدار teta=180 خروجی ماکزیمم حالت(255) خواهد بود ودر قسمت دوم خروجی به صورت نزولی کاهش پیدا می کند تا به مقدار teta=360 برسد در این حالت خروجی مینیمم مقدار را خواهد داشت.

4- برنامه موج مربعی :

```
for(teta=181;teta<=360;teta++)
{

                                                    if(amp>6.0)amp=1.0;
out[teta]=(0.2)*amp*((2*(360-teta)*1/360.)+128);
}
```

در زیر نمای کامل این قسمت از پروژه را آورده ایم:



برنامه ی دوم :

```
#include <mega16.h>
#include <stdlib.h>
#include <delay.h>
#include <math.h>
#define keypad_DDR DDRB
```

```
#define keypad_PORT PORTB
#define keypad_PIN PINB
#define key_ctr 5
#define key_delay 1
#define F1hz 125000
#define dac1 PORTD
#define F1 14
#define F2 24
#define F3 34
#define F4 41
#define _FF_ 0.71112
unsigned char out[360];
// Alphanumeric LCD Module functions
#asm
  .equ __lcd_port=0x1B ;PORTA
#endasm
#include <lcd.h>
// Declare your global variables here
unsigned long T1;          //Timer Numeric
unsigned int frequnce = 100;
unsigned int radu = 0;
unsigned char scale = 100;
unsigned int teta2=0;
unsigned char m =1;
unsigned int teta=0;
float amp=0.5;
// Timer 1 overflow interrupt service routine
interrupt [TIM1_OVF] void timer1_ovf_isr(void)
{
// Reinitialize Timer 1 value
TCNT1=T1;
dac1 = out[teta2];
teta2++;
if(teta2>360)teta2=0;
}
// Declare your global variables here
unsigned char scan_key(void)
```

```
{
  unsigned char
i=0,j=0,key_ctr_temp=0,key_temp=0,scan_code[4]={0xFE,0xFD,0xFB,0xF7};
  for (i=0;i<4;i++)
  {
      keypad_PORT|=0x0F;
      keypad_PORT&=scan_code[i];
      delay_ms(5);
      //key_temp=keypad_PIN;
      if (keypad_PIN<0xF0)
      {
        key_temp=keypad_PIN;
        for(j=0;j<key_ctr;j++)
        {
          delay_ms(key_delay);
          if(keypad_PIN==key_temp)
            key_ctr_temp++;
        }
//        if(key_ctr_temp==key_ctr)
       {   key_ctr_temp = keypad_PIN;
           switch(key_ctr_temp)
           {
                case 238:
                     return 1;
                     break;
                case 237:
                     return 4;
                     break;
                case 235:
                     return 7;
                     break;
                case 231:
                     return 41;
                     break;
                case 222:
                     return 2;
                     break;
```

```
        case 221:
            return 5;
            break;
        case 219:
            return 8;
            break;
        case 215:
            return 0;
            break;
        case 190:
            return 3;
            break;
        case 189:
            return 6;
            break;
        case 187:
            return 9;
            break;
        case 183:
            return 43;
            break;
        case 126:
            return 14;
            break;
        case 125:
            return 24;
            break;
        case 123:
            return 34;
            break;
        case 119:
            return 44;
            break;
        default:
            return 255;
            break;
    }
```

```
        }
       }
  }
  return 255 ;
   }
unsigned int get_num(unsigned int min, unsigned int max)
{
unsigned int numb = 0;
unsigned char num, x = 0, key;
num = 0;
do
{
 lcd_gotoxy(x, 1);
 key = scan_key();
if (key != 255)
{
 delay_ms(5);
 key = scan_key();
 if (key != 255)
 {
  switch (key)
  {
   case 1  : if (x < 4) {num = 1; numb = numb *10; numb += num; lcd_putsf("1"); x++;
delay_ms(500);} break;
   case 2  : if (x < 4) {num = 2; numb = numb *10; numb += num; lcd_putsf("2"); x++;
delay_ms(500);} break;
   case 3  : if (x < 4) {num = 3; numb = numb *10; numb += num; lcd_putsf("3"); x++;
delay_ms(500);} break;
   case 4  : if (x < 4) {num = 4; numb = numb *10; numb += num; lcd_putsf("4"); x++;
delay_ms(500);} break;
   case 5  : if (x < 4) {num = 5; numb = numb *10; numb += num; lcd_putsf("5"); x++;
delay_ms(500);} break;
   case 6  : if (x < 4) {num = 6; numb = numb *10; numb += num; lcd_putsf("6"); x++;
delay_ms(500);} break;
   case 7  : if (x < 4) {num = 7; numb = numb *10; numb += num; lcd_putsf("7"); x++;
delay_ms(500);} break;
```

```
    case 8  : if (x < 4) {num = 8; numb = numb *10; numb += num; lcd_putsf("8"); x++;
delay_ms(500);} break;
    case 9  : if (x < 4) {num = 9; numb = numb *10; numb += num; lcd_putsf("9"); x++;
delay_ms(500);} break;
    case 0  : if (x < 4) {num = 0; numb = numb *10; numb += num; lcd_putsf("0"); x++;
delay_ms(500);} break;
    case 43 : if (x > 0) {x--; lcd_gotoxy(x, 1); lcd_putsf(" "); numb = numb / 10;
delay_ms(500);}; break;
    case 44 : if (numb < min) {numb = min; }; if (numb > max) {numb = max; }; break;
   }

   lcd_gotoxy(x, 1);
  }
}
} while(key != 44);
//numb = max-min;
return numb;
}
void lcd_panel1(void)
{
char s_s[5];
#asm("cli")
lcd_clear();
lcd_gotoxy(0,0);
lcd_putsf("F:      hz. ");
lcd_gotoxy(0, 1);
lcd_putsf("R :   . Y-G:%   ");
itoa(frequnce, s_s);
lcd_gotoxy(2, 0);
lcd_puts(s_s);
itoa(radu, s_s);
lcd_gotoxy(3, 1);
lcd_puts(s_s);
itoa(scale, s_s);
lcd_gotoxy(13, 1);
lcd_puts(s_s);
lcd_gotoxy(0, 12);
```

```
lcd_puts(s_s);
#asm("sei")
}
void panel2(void)          // change freq
{
#asm("cli")
lcd_clear();
lcd_gotoxy(0, 0);
lcd_putsf("2000 > Freq > 9");
frequnce = get_num(10, 1300);
teta2 = 0;
T1 = ((F1hz / frequnce) - 34)*-1;
TCNT1 = T1;
TCNT1=0XFFF0;
lcd_panel1();
}
void panel3(void)          // change phase
{
#asm("cli")
lcd_clear();
lcd_gotoxy(0, 0);
lcd_putsf("Phase R  (0~359)");
teta2= 0;
radu = get_num(0, 359);
lcd_panel1();
}
void panel4(void)          // change volt
{
   #asm("cli")
lcd_clear();
lcd_gotoxy(0, 0);
lcd_putsf("Amplitude 1~5");
lcd_gotoxy(0, 1);
lcd_putsf("  =v");
lcd_gotoxy(3, 1);
teta2 = 0;
amp= get_num(1, 5) ;
```

```
TCNT1 = T1;
lcd_panel1();
}
void panel5(void)          // change WAVE
{
#asm("cli")
lcd_clear();
lcd_gotoxy(0, 0);
lcd_putsf("1)Sin,  2)Ramp");
lcd_gotoxy(0, 1);
lcd_putsf("3)Train,4)Square");
 m=get_num(1, 5) ;
teta2  = 0;
lcd_panel1();
 }
void GenerateShape(void)
{
//sin
if(m==1)
      {
for(teta=0;teta<=360;teta++)
{
if(amp>6.0)amp=1.0;
out[teta]=((0.2)*amp*120*sin(teta*2*3.141592/360.)+128);
}
}
// Ramp
      if(m==2)
      {
for(teta=0;teta<=360;teta++)
{
if(amp>6.0)amp=1.0;
out[teta]=((0.2)*amp*(teta*255/360.));
}
}
// Traingle
if(m==3)
```

```
        {
for(teta=0;teta<180;teta++)
{
if(amp>6.0)amp=1.0;
out[teta]=(0.2)*amp*(teta*255/180.);
}
for(teta=180;teta<=360;teta++)
{
if(amp>6.0)amp=1.0;
out[teta]=(0.2)*amp*((360-teta)*255/180.);
}
}
//square
if(m==4)
        {
for(teta=181;teta<=360;teta++)
{
if(amp>6.0)amp=1.0;
out[teta]=(0.2)*amp*((2*(360-teta)*1/360.)+128);
}
}
}
void main(void)
{
unsigned char keybd;
// Declare your local variables here
// Input/Output Ports initialization
// Port A initialization
// Func7=Out Func6=Out Func5=Out Func4=Out Func3=Out Func2=Out Func1=Out
Func0=Out
// State7=0 State6=0 State5=0 State4=0 State3=0 State2=0 State1=0 State0=0
PORTA=0x00;
DDRA=0xFF;
// Port B initialization
// Func7=Out Func6=Out Func5=Out Func4=Out Func3=Out Func2=Out Func1=Out
Func0=Out
// State7=0 State6=0 State5=0 State4=0 State3=0 State2=0 State1=0 State0=0
```

```
PORTB=0xF0;
DDRB=0x0F;
// Port C initialization
// Func7=Out Func6=Out Func5=Out Func4=Out Func3=Out Func2=Out Func1=Out
Func0=Out
// State7=0 State6=0 State5=0 State4=0 State3=0 State2=0 State1=0 State0=0
PORTC=0x00;
DDRC=0xFF;
// Port D initialization
// Func7=In Func6=In Func5=In Func4=In Func3=Out Func2=Out Func1=Out Func0=Out
// State7=T State6=T State5=T State4=T State3=0 State2=0 State1=0 State0=0
PORTD=0x00;
DDRD=0xFF;
// 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: 16000.000 kHz
// Mode: Normal top=FFFFh
// OC1A output: Discon.
// OC1B output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer 1 Overflow Interrupt: On
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
TCCR1A=0x00;
TCCR1B=0x01;
TCNT1H=0xAA;
TCNT1L=0xAA;
```

```
ICR1H=0x00;
ICR1L=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
MCUCR=0x00;
MCUCSR=0x00;
// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x04;
// Analog Comparator initialization
// Analog Comparator: Off
// Analog Comparator Input Capture by Timer/Counter 1: Off
ACSR=0x80;
SFIOR=0x00;
// LCD module initialization
lcd_init(16);
// Global enable interrupts
frequnce = 100;
T1 = ((F1hz / frequnce) -33)*-1;
lcd_panel1();
while (1)
    {
    GenerateShape();
```

```
    keybd = scan_key();
    if (keybd != 255)
    {
     switch (keybd)
     {
       case F1 : panel2(); break;
       case F2 : panel3(); break;
       case F3 : panel4(); break;
       case F4 : panel5(); break;
     }
    }
//   }
    };
}
```

توضیحاتی در مورد برنامه ی سوم:

در برنامه قبل دو اشکال وجود داشت که یکی از ایراد های آن وجود فرکانس پایین حدود300 هرتز بود که با افزایش آن دامنه ی شکل موج خیلی کم می شد و دیگری وجود اعوجاج در شکل موج ها بود.که با یک ایده ی جدید از دو برنامه ی قبل یک برنامه ی جدید نوشته شد. که بطور کامل در زیر توضیح داده شده است.

این برنامه دارای پنج scale است که ما به ترتیب نام هریک را آورده ایم:

1- شکل موج مربعی:
2- شکل موج دندان اره ای:
3- شکل موج مثلثی :
4- شکل موج سینوسی:
5- شکل موج پالس:

1- موج مربعی:

برای بدست آوردن یک موج مربعی از آنجایی که 256 حالت داریم باید نصف این حالات را در مینیمم مقدار قرار بدهیم و نصف مقدار دیگر را در ماکزیمم مقدار قرار دهیم.

در زیر scale مربوط به شکل موج مربعی را آورده ایم:

{
255, 255, 255, 255, 255,255, 255, 255, 255, 255, 255, 255,255, 255, 255,
255, 255,255, 255, 255,255, 255, 255,255, 255, 255, 255,255, 255, 255,
255, 255, 255, 255, 255,255, 255, 255, 255, 255, 255, 255,255, 255, 255,
255, 255,255, 255, 255,255, 255, 255,255, 255, 255, 255,255, 255, 255,
255, 255, 255, 255, 255,255, 255, 255, 255, 255, 255, 255,255, 255, 255,
255, 255,255, 255, 255,255, 255, 255,255, 255, 255, 255,255, 255, 255,
255, 255, 255, 255, 255,255, 255, 255, 255, 255, 255, 255,255, 255, 255,
255, 255,255, 255, 255,255, 255, 255,255, 255, 255, 255,255, 255, 255,
255,255, 255, 255, 255,255, 255, 255,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1
}

2- موج دندان اره ای :

همانطور که از شکل، این موج بر می آید این موج از یک مقدار معین شروع وبه صورت یک واحد یک واحد افزایش پیدا می کند تا به مقدار ماکزیمم برسد و بعد از آن به مقدار اولیه میرسد و این روند تکرار پیدا می کند پس برای ساختن موجramp ابتدا باید بوسیله ی این حالات یک پالس مورب بسازیم تا با قرار دادن آن در یک حلقه موج مان ساخته شود برای این کار در scaleمان از یک شروع می کنیم تا به 255 برسیم

در زیر scaleشکل موج دندان اره ای را مشاهده می کنید.

{
1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,2
9,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52,53,
54,55,56,57,58,59,60,61,62,63,64,65,66,67,68,69,70,71,72,73,74,75,76,77,78
,79,80,81,82,83,84,85,86,87,88,89,90,91,92,93,94,95,96,97,98,99,100,101,10
2,103,104,105,106,107,108,109,110,111,112,113,114,115,116,117,118,119,1
20,121,122,123,124,125,126,127,128,129,130,131,132,134,135,136,137,138,
139,140,141,142,143,144,145,146,147,148,149,150,151,152,153,154,155,15

6,157,158,158,159,160,161,162,163,164,165,166,167,168,169,170,171,172,1
73,174,175,176,177,178,179,180,181,182,183,184,185,186,187,188,189,190,
191,192,193,194,195,196,197,198,199,200,201,202,203,204,205,206,207,20
8,209,210,211,212,213,214,215,216,217,218,219,220,221,222,223,224,225,2
26,227,228,229230,231,232,233,234,235,236,237,238,239,240,241,242,243,
244,245,246,247,248,249,250,251,252,253,254,255

}

3- موج مثلثی :

برای بدست اودن این موج باید ابتدا از مینیمم مقداربا افزایش واحد به واحد به ماکزیمم رسید البته باید این وضعیت در نصف حالات صورت گیرد چون این مقدار دوباره به صورت کاهش واحد به واحد به مینیمم برسد که یک شکل مثلث مانند ایجاد شود که با تکرار ان یک موج مثلثی ایجاد می شود

بنابراین 128 تای اول حالت صعودی شکل موج ما را تشکیل می دهند و128 تای بعد حالت نزولی شکل موج مان است. وهمچنین باید این را در نظر داشت. که برای اینکه ماکسیم ولتاژ ما برای این شکل موج 5 ولت باشد باید در ماکزیمم مقدار به 255 برسیم پس حالت افزایش یا کاهش ما به صورت یک در میان صورت میگیرد که هر واحد ما دوعدد در نظر گرفته شده است.

در زیر scaleشکل موج دندان اره ای را مشاهده می کنید.

{

1,3,5,7,9,11,13,15,17,19,21,23,25,27,29,31,33,35,37,39,41,43,45,47,49,51,53
,55,57,59,61,63,65,67,69,71,73,75,77,79,81,83,85,87,89,91,93,95,97,99,101,
103,105,107,109,111,113,115,117,119,121,123,125,127,129,131,135,137,13
9,141,143,145,147,149,151,153,157,159,161,163,165,167,169,171,173,175,1
77,179,181,183,185,187,189,191,193,195,197,199,201,203,205,207,209,211,
213,215,217,219,221,223,225,227,229,231,233,235,237,239,241,243,245,24
7,249,251,253,255,253,251,249,247,245,243,241,239,237,235,233,231,229,2
27,225,223,221,219,217,215,213,211,209,207,205,203,201,199,197,195,193,

191,189,187,185,183,181,179,177,175,173,171,169,167,165,163,161,159,15
7,155,153,151,149,147,145,143,141,139,137,135,133,131,129,127,125,123,1
21,119,117,115,113,111,109,107,105,103,101,99,97,95,93,91,89,87,85,83,81
,79,77,75,73,71,69,67,65,63,61,59,57,55,53,51,49,47,45,43,41,39,37,35,33,3
1,29,27,25,23,21,19,17,15,13,11,9,7,5,3,1
}

4- موج سینوسی :

توضیحات این شکل موج را در برنامه ی اول آورده ایم.و فقط scale مورد استفاده را آورده ایم.

{
131 , 134 , 137 , 140 , 144 , 147 , 150 , 153 , 156 , 159 , 162 , 165 , 168 , 171 , 174 ,
177 , 180 , 183 , 185 , 188 , 191 , 194 , 196 , 199 , 201 , 204 , 206 , 209 , 211 , 214 ,
216 , 218 , 220 , 222 , 225 , 227 , 229 , 230 , 232 , 234 , 236 , 237 , 239 , 240 , 242 ,
243 , 245 , 246 , 247 , 248 , 249 , 250 , 251 , 252 , 252 , 253 , 254 , 254 , 255 , 255 ,
255 , 255 , 255 , 255 , 255 , 255 , 255 , 255 , 255 , 254 , 254 , 253 , 252 , 252 , 251 ,
250 , 249 , 248 , 247 , 246 , 245 , 243 , 242 , 240 , 239 , 237 , 236 , 234 , 232 , 230 ,
229 , 227 , 225 , 222 , 220 , 218 , 216 , 214 , 211 , 209 , 206 , 204 , 201 , 199 , 196 ,
194 , 191 , 188 , 185 , 183 , 180 , 177 , 174 , 171 , 168 , 165 , 162 , 159 , 156 , 153 ,
150 , 147 , 144 , 140 , 137 , 134 , 131 , 128 , 125 , 122 , 119 , 116 , 112 , 109 , 106 ,
103 , 100 , 97 , 94 , 91 , 88 , 85 , 82 , 79 , 76 , 73 , 71 , 68 , 65 , 62 , 60 , 57
, 55 , 52 , 50 , 47 , 45 , 42 , 40 , 38 ,
36 , 34 , 31 , 29 , 27 , 26 , 24 , 22 , 20 , 19 , 17 , 16 , 14 , 13 , 11 , 10 , 9 ,
8 , 7 , 6 , 5 , 4 , 4 , 3 , 2 , 2 , 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 ,
, 2 , 2 , 3 , 4 , 4 , 5 , 6 , 7 , 8 , 9 , 10 ,
11 , 13 , 14 , 16 , 17 , 19 , 20 , 22 , 24 , 26 , 27 , 29 , 31 , 34 , 36 , 38 , 40 ,
42 , 45 , 47 , 50 , 52 , 55 , 57 , 60 , 62 , 65 , 68 , 71 , 73 , 76 , 79 ,
82 , 85 , 88 , 91 , 94 , 97 , 100 , 103 , 106 , 109 , 112 , 116 , 119 , 122 , 125 , 128
}

5-پالس :

برای بدست اوردن چنین شکل موجی باید لحظه ای را ماکزیمم مقدار قرار دهیم وبعد از ان همه را مینیمم کنیم scaleاین برنامه نیز در پایین قرار دارد.

```
{
255, 255, 255, 255, 255,255, 255, 255, 255, 255, 255, 255,255, 255, 255,
255, 255,255, 255, 255,1,1,1,1,1,1,1,1,
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
}
```

_زیر برنامه key_pad :

برنامه ی key_pad ی که برای دو برنامه دیگر نوشته بودیم سرعت اسکن پایینی داشت. بنابراین با روش دیگری که توضیح خواهیم داد سرعت خواندن و همچنین حجم برنامه که در دو برنامه ی قبل زیاد بود در این برنامه مقداری کمتر شد.

از آنجایی که اصول کلیش را توضیح داده ایم فقط قسمتی که تغییر داده ایم یعنی فقط زیر برنامه اش را توضیح می دهیم .

ابتدا دو متغیر آرایه ای زیر راتعریف می کنیم :

```
flash unsigned char init[4][4]={{15,3,2,1},{20,6,5,4},{30,9,8,7},{40,25,10,11}};

flash unsigned char read[4]={0xEF,0xDF,0xBF,0x7F};
```

و از هشت پین موجود در کیبورد چهار تای اول را به عنوان خروجی و چهار پین بعدی را به عنوان ورودی قرار دادیم بعد به وسیله یک حلقه for متغیر read را به ترتیب برابر پین خروجی قرار دادیم .

بنابراین با این کار 4 پین اول را به ترتیب صفر وچهار تای بعد ،set خواهد شد.پس از یک تاخیر ورودی را برابر متغیرkey قرار می دهیم وان را باـ0x0F اند بیتی می کنیم تا پین خروجیreset شود. حال میکرو با دستور switch_case کلیدی که زده شده است را پیدا می کند .

با یک مثال بحث را روشن ترمی کنیم اگر پین اول ورودی زده شده باشد یعنی (0x0E)

متغیرinit را بصورت زیر در key میریزد.

```
key=init[0][j];
```

با فرض اینکه j=2 باشد از انجایی که j در متغیر، شماره ی بسته را مشخص می کند پس شماره ی30 یا همان دکمه یf3 رادر کی پد وارد کرده ایم.

در زیر برنامه کا مل این قسمت را آورده ایم

```
void key_read(void){
unsigned int j;
for(j=0;j<4;j++){
kbd_col=read[j];
delay_us(20);
key=kbr_row;
key=key&0x0F;
switch(key)
{
case 0x0E:
[key=init[0][j];
temp=key;
Break;
case 0x0D:
key=init[1][j];
temp=key;
Break;
case 0x0B:
key=init[2][j];
temp=key;
Break;
case 0x07:
key=init[3][j];
temp=key;
Break;
```

```
Default:
Break;
}
}
}
```

برنامه کلید های کنترلی که در داخل وقفه ی خارجی INT0 که به صورت falling edge می باشد ، را نوشته ایم.حال نحوه ی کار هر یک از کلید های کنترلی راتوضیح می دهیم

کلید f3:

اگر کلید f3 که معادل رقم 30 است زده شده باشد فرکانس مدار دو برابر می شود. برنامه این قسمت به این صورت است که با هر بار زدن کلید ، به k یک واحد اضافه می شود مقدار اولیه این تابع برابر یک است که اگر k=2 شود متغیرx را مساوی 10 میکند واگر k=3 شد k را برابر یک می کند

در این برنامه از تایمر یک نیز استفاده کرده ایم که میکرو وقفه ی تایمر یک را به صورت CTC با رجیستر OCR1A مقایسه می کند.

آرایه ی مربوط در وقفه ی تایمر، موج را روی یک پورت که به dacمی رود می فرستد در این ارایه که در حالت عادی پنج تا پنج تا مقدار لحظه ای این موج ها را از scale های مر بوطه بر می دارد به محض زدن کلید f3این کار ده تا ده تا انجام می شود که این کار باعث دو برابر شدن فرکانس می شود.

کلید f2:

کلید f2که معادل عدد 20 می باشد جهت کنترل دامنه مورد استفاده قرار می گیرد. محدوده ی دامنه ی این پروژه تا5 ولت است که با دو رقم اعشار می توان مقدار آن را وارد کرد به محض زدن کلیدf2 روی صفحه ی lcd توضیح های زیر ظاهر می شود :

```
" < Limit Of Redu:"
"---V 5>AMP*10>1 "
```

علاوه بر این میکرو en=1 قرار می دهد که جهت فعال کردن مقدار پذیری برای وارد کردن دامنه است.به محض زدن عددی اگر en=1 باشد فرمول num1=(num1*10)+temp; اجرا می شود متغیر temp مقدار عددی است که ما می دهیم. مقدار اولیه num1 نیز صفر است حا ل با فرض اینکه می خواهیم 4.7 ولت را وارد کنیم نحوه ی اجرای فرمول را توضیح می دهیم.

ابتدا رقم 4 را وارد می کنیم و بعد از آن رقم 7 را . بنا براین در مرحله ی اول num1 برابر 4می شود وبعد از آن طبق num1=(4*10)+7 مقدار آن برابر47 می شود که این مقدار را روی lcd نمایش می دهد.

و بعد از وارد کردن enter چنین عمل می کند:

اگر عدد ما دو رقمی باشد. تقسیم بر 10 می شود واگر سه رقمی باشد ان را تقسیم بر 100 می کند و روی lcd نمایش می دهد البته این مقدار را برابر OCR2 قرار داده است .که برای تعین دامنه مرجع DAC ما است.

کلید f1:

این کلید که معادل عدد 15 است جهت کنترل فر کانس می باشد . محدوده فر کانسی ما از 1 تا 20 کیلو هرتز است. به محض زدن این کلید توضیحات زیر روی lcd نمایش داده می شود وهمچنین میکرو en=2 قرار می دهد.

"<Limit Of Ferq:"

"----Hz 20k>Fr>1"

اگر en=2 باشد فرمول num=(num*10)+temp; اجرا می شود متغیر temp مقدار عددی است که ما می دهیم .البته مقدار اولیه num نیز صفر است.

وهمچنین فرمول fr=(1000000/(num+180)); را اجرا می کند .fr میزان فرکانس را مشخص می کند .

بعد زدن enter مقدار fr را با OCR1A برابر قرار می دهد.ومقدار فرکانس را روی lcd نمایش می دهد.

کلید PROG:

این کلید که معادل عدد 11 است جهت انتخاب شکل موج مورد نظرمان می باشد.که با زدن این کلید و فشردن کلید اینتر شکل موج ما تغییر می کند .

نحوه ی ان به این شکل است که در هر بار زدن این کلید ، به متغییر a یک واحد اضافه می شود که a نیز نقش انتخاب scaleها را در آرایه out=source[a][i]; بر عهده دارد. بعد از پنج بار زدن این کلید که a=5 میشود آن را برابر صفر قرار می دهد .

برای درک بیشتر این موضوع به ازای هر مقدار a که شکل موج مربوط به ان انتخاب می شود را در زیر آورده ایم.

موج مربعی

```
if(a==0)lcd_putsf("SQR")
```

موج ramp

```
if(a==1)lcd_putsf("RAP");
```

موج سینوسی

```
if(a==2)lcd_putsf("SIN");
```

موج مثلثی

```
if(a==3)lcd_putsf("TRG");
```

پالس

```
if(a==4)lcd_putsf("PUL");
```

توضیحاتی در مورد برنامه ی چهارم:

در برنامه نهایی نسبت به برنامه قبل برای دستورات کلید های کنترلی key_pad ، بجای استفاده از دستور if ازدستور switch_case استفاده کرده ایم .

و همچنین کنترل فاز که در بر نامه قبل استفاده نکرده بودیم در این برنامه از ان استفاده کردیم.

کلید M:

این کلید معادل رقم 40 است .که با زدن هر بار کلید لحظه شروع شکل موج مان از نیمسیکل مثبت یا منفی تغیر پیدا می کند.در حالت عادی شکل موج مان از سیکل مثبت شروع می شود.

ماژول کلیدM:

```
case 40:
Ph++;
delay_ms(100);
if(ph==3)ph=1;

if(ph==1) {       //phase++++++++++
i=0;
lcd_gotoxy(15,1);
lcd_putsf("+");
{
if(ph==2){       //phase----------
i=125;
lcd_gotoxy(15,1);
lcd_putsf ("-");
{
temp=0;
```

فصل پنجم : برنامه ی نهایی پروژه

در زیر برنامه ی نهایی پروژه را آورده ایم:

```
#include <mega16.h>
#include <delay.h>
#include <stdio.h>
#include <stdlib.h>

#asm
```

```
  .equ __lcd_port=0x15 ;PORTC
#endasm
#include <lcd.h>

#define out PORTA
#define kbd_col PORTB
#define kbr_row PINB
unsigned char en=0;
unsigned char i,a;
unsigned char *str;
char str1[8];
unsigned char key,k=1,x=5,ph;
float amp,ampt,num1;
unsigned int temp;
unsigned long int num=0,fr=0;
flash unsigned char init[4][4]={{15,3,2,1},{20,6,5,4},{30,9,8,7},{40,25,10,11}};
flash unsigned char read[4]={0xEF,0xDF,0xBF,0x7F};
flash unsigned int source[5][256] = {
{255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,2
55,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255
,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,2
55,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255
,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,2
55,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,1,1,
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1
},
{
1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34
,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52,53,54,55,56,57,58,59,60,61,62,63,64,
65,66,67,68,69,70,71,72,73,74,75,76,77,78,79,80,81,82,83,84,85,86,87,88,89,90,91,92,93,94,
95,96,97,98,99,100,101,102,103,104,105,106,107,108,109,110,111,112,113,114,115,116,117,
118,119,120,121,122,123,124,125,126,127,128,129,130,131,132,133,134,135,136,137,138,13
9,140,141,142,143,144,145,146,147,148,149,150,151,152,153,154,155,156,157,158,159,160,
161,162,163,164,165,166,167,168,169,170,171,172,173,174,175,176,177,178,179,180,181,18
2,183,184,185,186,187,188,189,190,191,192,193,194,195,196,197,198,199,200,201,202,203,
```

204,205,206,207,208,209,210,211,212,213,214,215,216,217,218,219,220,221,222,223,224,22
5,226,227,228,229,230,231,232,233,234,235,236,237,238,239,240,241,242,243,244,245,246,
247,248,249,250,251,252,253,254,255
},
{
131 , 134 , 137 , 140 , 144 , 147 , 150 , 153 , 156 , 159 , 162 , 165 , 168 , 171 , 174 , 177 , 180
, 183 , 185 , 188 , 191 , 194 , 196 , 199 , 201 , 204 , 206 , 209 , 211 , 214 , 216 , 218 , 220 ,
222 , 225 , 227 , 229 , 230 , 232 , 234 , 236 , 237 , 239 , 240 , 242 , 243 , 245 , 246 , 247 , 248
, 249 , 250 , 251 , 252 , 252 , 253 , 254 , 254 , 255 , 255 , 255 , 255 , 255 , 255 , 255 , 255 ,
255 , 255 , 255 , 254 , 254 , 253 , 252 , 252 , 251 , 250 , 249 , 248 , 247 , 246 , 245 , 243 , 242
, 240 , 239 , 237 , 236 , 234 , 232 , 230 , 229 , 227 , 225 , 222 , 220 , 218 , 216 , 214 , 211 ,
209 , 206 , 204 , 201 , 199 , 196 , 194 , 191 , 188 , 185 , 183 , 180 , 177 , 174 , 171 , 168 , 165
, 162 , 159 , 156 , 153 , 150 , 147 , 144 , 140 , 137 , 134 , 131 , 128 , 125 , 122 , 119 , 116 ,
112 , 109 , 106 , 103 , 100 , 97 , 94 , 91 , 88 , 85 , 82 , 79 , 76 , 73 , 71 , 68 , 65 , 62 ,
60 , 57 , 55 , 52 , 50 , 47 , 45 , 42 , 40 , 38 ,
36 , 34 , 31 , 29 , 27 , 26 , 24 , 22 , 20 , 19 , 17 , 16 , 14 , 13 , 11 , 10 ,
9 , 8 , 7 , 6 , 5 , 4 , 4 , 3 , 2 , 2 , 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 , 2
, 2 , 3 , 4 , 4 , 5 , 6 , 7 , 8 , 9 , 10 ,
11 , 13 , 14 , 16 , 17 , 19 , 20 , 22 , 24 , 26 , 27 , 29 , 31 , 34 , 36 , 38 , 40 , 42 , 45 ,
47 , 50 , 52 , 55 , 57 , 60 , 62 , 65 , 68 , 71 , 73 , 76 , 79 ,
82 , 85 , 88 , 91 , 94 , 97 , 100 , 103 , 106 , 109 , 112 , 116 , 119 , 122 , 125 , 128
},
{
1,3,5,7,9,11,13,15,17,19,21,23,25,27,29,
31,33,35,37,39,41,43,45,47,49,51,53,55,
57,59,61,63,65,67,69,71,73,75,77,79,81,
83,85,87,89,91,93,95,97,99,101,103,105,
107,109,111,113,115,117,119,121,123,125,
127,129,131,133,135,137,139,141,143,145,
147,149,151,153,155,157,159,161,163,165,
167,169,171,173,175,177,179,181,183,185,
187,189,191,193,195,197,199,201,203,205,
207,209,211,213,215,217,219,221,223,225,
227,229,231,233,235,237,239,241,243,245,
247,249,251,253,255,253,251,249,247,245,243,241,239,237,235,233,231,229,227,225,223,22
1,219,217,215,213,211,209,207,205,203,201,199,197,195,193,191,189,187,185,183,181,179,
175,173,171,169,167,165,163,161,159,157,155,153,151,149,147,145,143,141,139,137,135,13

```
3,131,129,127,125,123,121,119,117,115,113,111,109,107,105,103,101,99,97,95,93,91,89,87,
85,83,81,79,77,75,73,71,69,67,65,63,61,59,57,55,53,51,49,47,45,43,41,39,37,35,33,31,29,27,
25,23,21,19,17,15,13,11,9,7,5,3,1
},
{
255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,1,1,1,1
,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
1,1,
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1
}
};
//-----------------------read key pad****
void key_read(void){
unsigned int j;
for(j=0;j<4;++j){
kbd_col=read[j];
delay_us(20);
key=kbr_row;
key=key&0x0F;
switch(key)
{
case 0x0E:
key=init[0][j];
temp=key;
break;
case 0x0D:
key=init[1][j];
temp=key;
break;
case 0x0B:
key=init[2][j];
temp=key;
break;
case 0x07:
```

```
key=init[3][j];
temp=key;
break;
default:
break;
}
}
}
// External Interrupt 0 service routine
interrupt [EXT_INT0] void ext_int0_isr(void)
{

////////////////////////////////////////////////
switch (temp){
case 30:   // f3 f3 f3 f3 f3 f3 f3 f3 f3  {fr*2}
k++;
delay_ms(50);
if(k==3)k=1;
if(k==1)x=5;      //frequence *1
if(k==2)x=10;     //frequence *2

temp=0;
break;

//////////////////phase phase M M M M M M M
case 40:
ph++;
delay_ms(100);
if(ph==3)ph=1;

if(ph==1){        //phase+++++++++
i=0;
 lcd_gotoxy(15,1);
 lcd_putsf("+");
        }
if(ph==2){        //phase----------
i=125;
```

```
 lcd_gotoxy(15,1);
 lcd_putsf("-");
        }
   temp=0;
break;
//**************************F1 F1 F1 F1 F1 F1--FREQUENC
case 15:
   lcd_clear();
   lcd_gotoxy(0,0);
   lcd_putsf(">Limit Of Ferq: ");
   lcd_gotoxy(0,1);
   lcd_putsf("----Hz  20k>Fr>1");

   en=3;

   num=0;
   delay_ms(1);
   fr=0;
   temp=0;
break;
//*****************************F2 F2 F2 F2 F2 --REDU
case 20:
   lcd_clear();
   lcd_gotoxy(0,0);
   lcd_putsf(">Limit Of Redu: ");
   lcd_gotoxy(0,1);
   lcd_putsf("--- V 5>AMP*10>1");

   en=1;

   num1=0;
   amp=0;
   ampt=0;
   temp=0;
break;

//*****************************PROG PROG PROG PROG --MOJ
```

```
case 11:
   ++a;
   if(a==5)a=0;
   lcd_gotoxy(13,0);
   if(a==0)lcd_putsf("SQR");
   if(a==1)lcd_putsf("RAP");
   if(a==2)lcd_putsf("SIN");
   if(a==3)lcd_putsf("TRG");
   if(a==4)lcd_putsf("PUL");

   temp=0;
break;
//*********************************ENTER ENTER ENTER ENTER
case 25:

   OCR1A=fr;        //SHOW FREQUENCE
   lcd_clear();
   lcd_gotoxy(0,0);
   lcd_putsf("Freq:     Hz-   ");
   lcd_gotoxy(0,1);
   lcd_putsf("Redu:     Volt  ");
   itoa(num,str);
   lcd_gotoxy(5,0);
   lcd_puts(str);

   if(num1>=10){     //SHOW AMPT
   amp=(num1/10)*51;
   OCR2=amp;
   ampt=num1/10;
   }
   if(num1>=100){
   amp=(num1/100)*51;
   OCR2=amp;
   ampt=num1/100;
   }
   lcd_gotoxy(5,1);
   ftoa(ampt,2,str1);
```

```
    lcd_puts(str1);

    lcd_gotoxy(13,0);
    if(a==0)lcd_putsf("SQR");
    if(a==1)lcd_putsf("RAP");
    if(a==2)lcd_putsf("SIN");
    if(a==3)lcd_putsf("TRG");
    if(a==4)lcd_putsf("PUL");

    temp=0;
    ampt=0;
    amp=0;
    en=10;
break;
}
////////////////////////////////////////////////
if(temp!=0 && temp<=10){   //ENTER NUMBER 0-9
    if(temp==10)temp=0;

    if(en==1){                  //FOR DAMANE
        num1=(num1*10)+temp;
        lcd_gotoxy(0,1);
        itoa(num1,str);
        lcd_puts(str);
    }

    if(en==3){
        num=(num*10)+temp;      //FOR FREQUENCE
        fr=(1000000/(num+180));
        lcd_gotoxy(0,1);
        itoa(num,str);
        lcd_puts(str);
    }

    temp=0;

}
```

```
    delay_ms(88);
    temp=0;
}
// Timer 1 output compare A interrupt service routine
interrupt [TIM1_COMPA] void timer1_compa_isr(void)
{
  i=i+x;
  if(i>=255)
  i=0;
  out=source[a][i];
}

// Declare your global variables here

void main(void)
{
// Declare your local variables here

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

// Port B initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTB=0x0F;
DDRB=0xF0;

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

```
// Port D initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTD=0x00;
DDRD=0x80;

// 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: 16000.000 kHz
// Mode: CTC top=OCR1A
// OC1A output: Discon.
// OC1B output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer 1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: On
// Compare B Match Interrupt: Off
TCCR1A=0x00;
TCCR1B=0x09;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
```

```
OCR1BL=0x00;


// Timer/Counter 2 initialization
// Clock source: System Clock
// Clock value: 16000.000 kHz
// Mode: Fast PWM top=FFh
// OC2 output: Non-Inverted PWM
ASSR=0x00;
TCCR2=0x69;
TCNT2=0x00;
OCR2=0x00;

// External Interrupt(s) initialization
// INT0: On
// INT0 Mode: Falling Edge
// INT1: Off
// INT2: Off
GICR|=0x40;
MCUCR=0x02;
MCUCSR=0x00;
GIFR=0x40;

// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x10;

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

// LCD module initialization
lcd_init(16);

// Global enable interrupts
#asm("sei")
```

```
lcd_gotoxy(0,0);
lcd_putsf("   *Arta.s*   ");
delay_ms(20);

lcd_clear();
lcd_putsf("press any key...");
while (1)
    {
     key_read();
    };
}
```

طراحی مدار :

مدار ما نسبت به برنامه ای که نوشته ایم سه قطعه ی جانبی نیاز دارد ، که عبارتند از : lcd ، dac و key_pad ما باید نحوه وصل این سه قطعه به میکرو را پیدا می کردیم تا شماتیک مدار خود را بکشیم.

ابتدا برای راحتی کار تصمیم گرفتیم مدار خود را بخش به بخش جلو ببریم ابتدا قسمت lcd را روی برد بورد بستیم قبل از اینکه وصل کنیم فکر می کردیم براحتی از این قسمت جواب می گیریم اما این جور نبود وبر روی lcd فقط عدد 100 ظاهر می شد و ما بعد از دیدن جواب این شکلی سراغ تعویض پورت که به lcd وصل می شد رفتیم . که با این کار نیز مشکل ما حل نشد پس از پرسش از دوستان متوجه شدیم که باید فیوز بیت های میکرو را تغییر می دادیم.پس از وارد کردن فیوز بیت صحیح و پروگرم کردن دوباره میکرو و وصل آن به مدار این بار روی صفحه ی lcd منوی اولیه ی شروع به کاری که برای میکرو تعریف کرده بودیم ، روی lcd نمایش داده شد.

بعد از این قسمت نوبت به key_padمی رسید پس از وصل اولیه که نسبت به مدار نهایی هیچ چیزی وصل نبود بطور قطع جواب نگرفتیم. با کمی فکر متوجه شدیم که از آنجایی که ما از اینتراپت صفر استفاده می کنیم .ما باید چهار پین ورودی کی پد را توسط چهار دیود که آند دیود ها به هم وصل اند (بعدا توضیح خواهیم داد که چرا از دیود استفاده می کنیم) را به INT0 وصل می کنیم .

چون این وفقه به صورت پایین رونده فعال می شود پس باید یک حالت سطح مثبت دائم درست کنیم که تنها با زدن کلیدی از ورودی، این سطح صفر شود تا اینتراپت فعال شود.برای اینکه این پین را مثبت نگه داریم از یک مقاومت و خازن استفاده کردیم اما به محض اینکه پینی از ورودی زده شود آن پین صفر شده وبقیه پین ها یک هستند با قرار دادن دیود ها موجب این شدیم که تنها، پینی که زده شده است دیودش هدایت کند و بقیه قطع باشند با این کار سطح صفر مان درست شده و یک پالس پایین رونده ساخته ایم. اگر دیود ها نباشند سطح مثبت نیز به این پین وارد می شود وهیچ وقت این پالس پایین رونده درست نمی شود.

در این قسمت نیز بعد از زدن روی برد بورد از مدار جواب گرفتیم

بعد از این قسمت نوبت به مبدل دیجیتال به آنالوگ (dac) می رسد که به نظر می رسد سخترین قسمت مدار باشد اما با مراجعه به دیتا شیت این قطعه مدار خروجی خود را از آنجا پیدا کردیم حال همه مدار را روی برد برد بستیم وخوشبختانه مدارمان جواب داد.