
بسم الله الرحمن الرحیم

کنترل اکولایزری و استخراج مؤلفه های فرکانسی

محمد قامت

بهار ۱۳۹۱

فهـــرسـت :

پیشگفتار

درس پروژه ساخت سه واحد عملی است که دانشجویان درآخرین ترم دوره تحصیلی خود آن را می گذرانند تا توانایی پیاده سازی عملی هرآنچه که تاکنون آموخته اند پیدا کنند. در طول دوره کارشناسی ما واحدهای مختلفی را اعم از تئوری و عملی گذراندیم. درس پروژه ساخت به ما کمک می کند تا از دروس گذشته بتوانیم یک پیاده سازی عملی و در عین حال کاربردی داشته باشیم و از مدارهای مختلی که فراگرفته ایم در راستای یک هدف خاص استفاده کنیم. با توجه به این نکته که ما در دروس عملی و آزمایشگاهی گذشته معمولا فقط به بررسی مدارها به صورت جداگانه پرداخته و هیچ گاه از آن ها برای یک کاربرد استفاده نکرده ایم، درس پروژه ساخت می تواند کمک بزرگی در این راستا به دانشجویان داشته باشد.

به نظر من درس پروژه ساخت به تقویت دانشجو در تمام ابعاد در زمینه تحصیل علم کمک می کند؛ هم در مسائل تئوری هم در فراگیری روش های بهتر تحقیق و پژوهش و هم در بعد عملی و مهم تر از همه در برقراری پیوند بین همه این ابعاد بسیار مفید است.

مطالبی که در این پایان نامه به آن پرداخته شده است، شامل معرفی پروژه در فصل یک، شرح مدار و بخش سخت افزاری پروژه در فصل دوم، توضیح در مورد نرم افزارهای استفاده شده در این پروژه مثل Filter Solution و CodeVision که در فصل سوم آورده شده است،خلاصه ای مطالب بیان شده در فصل چهارم و بالاخره پیوست ها که شامل دیتاشیت ها، نقشه ها و... است، می باشد.

در پایان بر خود فرض می دانم تا در آخرین گزارشم در دوره کارشناسی از تمام استادانی که در طی این دوره برای من و سایر دوستانم زحمات زیادی کشیده اند سپاسگزاری نمایم و تشکر ویژه ای از جنا دکتر ملک محمد و همچنین جنا مهندس شهیدی که در طول این دوره این حقیر را تحمل نموده اند، داشته باشم.

در پناه حق سبز باشید و پایدار

فصل اول: معرفی پروژه

۱-۱- عنوان پروژه و اهداف آن:

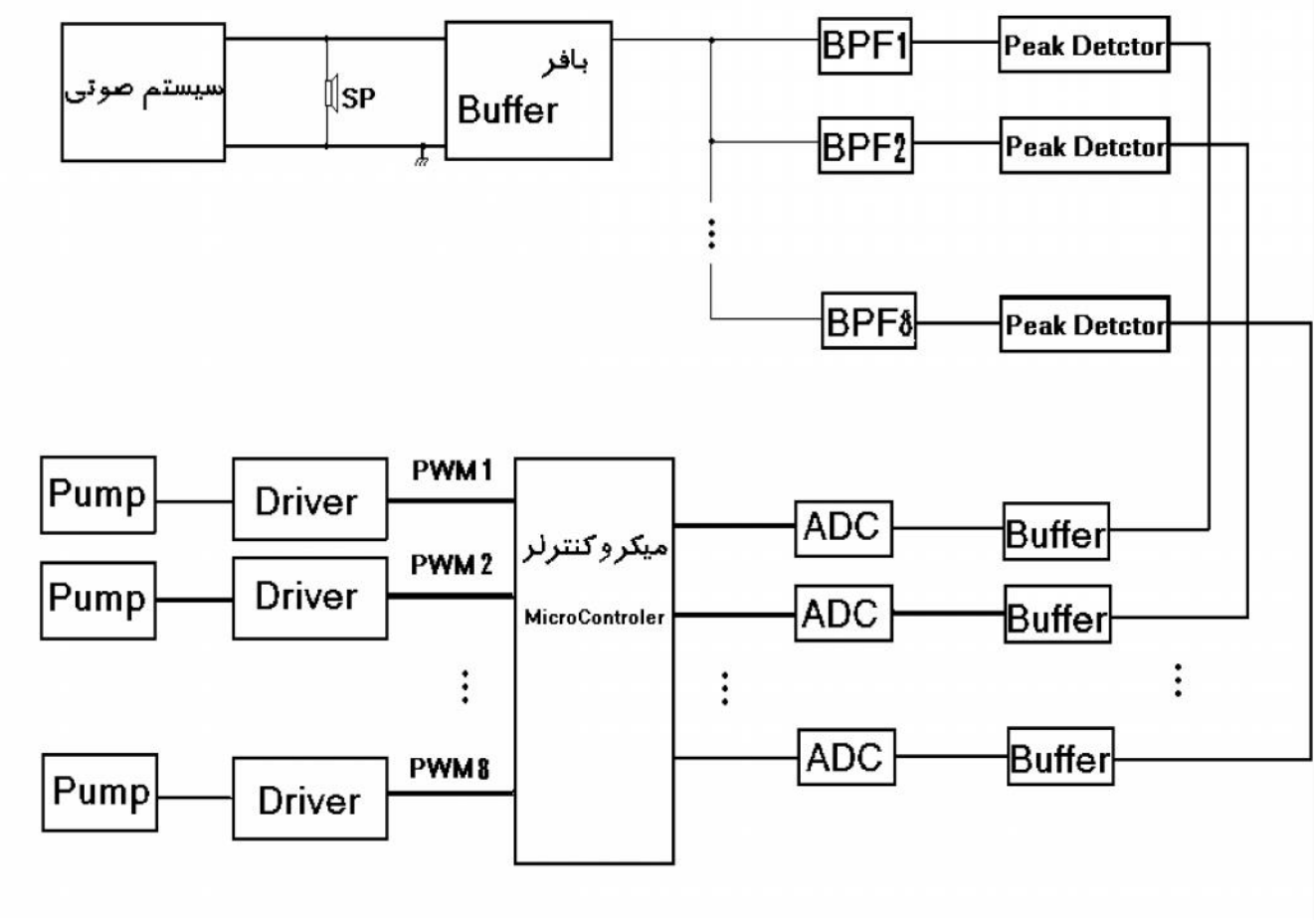
پروژه ای که در این پایان نامه به آن پرداخته شده است، "**کنترل اکولایزری و استخراج مؤلفه های فرکانسی**" نام دارد.

هدف کلی در این پروژه این است که مدار و سیستمی طراحی کنیم، که با پخش موسیقی، مؤلفه های مختلف فرکانسی آن تفکیک شده و با انجام یک سری پردازش ها نهایتاً حرکاتی موزون از فواره های آ که توسط تعدادی پمپ آ ایجاد می شود را به وجود آورد.

برای این کار باید مؤلفه های فرکانسی صوت توسط تعدادی فیلتر میان گذر استخراج و تفکیک شده و سپس به ولتاژهایی dc که متناسب با دامنه هر مؤلفه است، تبدیل شود. این ولتاژها سپس باید توسط مدار مبدل آنالوگ به دیجیتال به مقادیری دیجیتال تبدیل گردیده و پس از آن توسط مداری که در این جا میکروکنترلر این نقش را بازی می کند به تعداد مؤلفه های جدا شده از صوت، موج های PWM برای مدار راه انداز پمپ های فواره ها تولید شود. در نهایت هم مدار راه انداز با توجه به موج های PWM ورودی اش، پمپ های آ را راه اندازی و سطح آ خروجی از آن ها را کنترل می نماید.

۱-۲- بلاک دیاگرام پروژه:

بلاک دیاگرام پروژه با توجه به مطالب گفته شده در بالا به صورت شکل ۱-۱ می باشد.



شکل ۱-۱: بلاک دیاگرام پروژه

همان طور که در شکل مشاهده می شود؛ بلاک دیاگرام شامل بخش های زیر می باشد:

۱- مدار بافر صوت برای نمونه برداری از صوت بدون ایجاد اختلال در سیستم صوتی

۲- فیلترهای میان گذر جهت تفکیک مؤلفه های فرکانسی صوت

۳- مدار آشکارساز پیک جهت تبدیل هر مؤلفه به مقدار dc

۴- بافر برای جلوگیری از تغییرات خروجی مدار آشکارساز پیک در اثر بارگذاری طبقه بعد

۵- مدار مبدل آنالوگ به دیجیتال ADC برای تبدیل مقادیر dc به مقادیر دیجیتال بخش پردازنده سیستم که میکروکنترلر AVR می باشد که بخش ADC را نیز در درون خود جای داده است.

۶- مدار راه انداز پمپ های آ خروجی

۷- پمپ های آ خروجی جهت ایجاد فواره های آ

۱-۳- خلاصه ای از مطالب فصول بعد

در فصل دوم بخش های مختلف بلاک دیاگرام و مدارهای آن ها و ملاحظات عملی مربوط به آن ها شرح داده شده است که به صورت تیتروار شامل بخش های زیر است:

۲-۱- نمونه برداری از صوت و بافر کردن آن

۲-۲- تفکیک مؤلفه های فرکانسی

۲-۳- انتخا نوع فیلترها

۲-۴- انتخا فرکانس عبور

۲-۵- طراحی فیلترها

۲-۶- تبدیل سیگنال تفکیک شده هر مولفه به موج dc

۲-۷- بخش میکروکنترلر و پردازش

۲-۸- طبقه راه انداز پمپ های آ

۲-۹- پمپ های آ خروجی

در فصل سوم به چگونگی برنامه نویسی و نرم افزارهایی که در این پروژه مورد استفاده قرار گرفته است، پرداخته شده است و نحوه استفاده از نرم افزار Filter Solution جهت طراحی فیلتر و چگونگی برنامه نویسی به زبان C در محیط نرم افزار CodeVision برای میکروکنترلر AVR که در این پروژه استفاده شده است و همچنین کار با این نرم افزار توضیح داده شده است. لازم به ذکر است که در فصل سوم سعی شده است که از پرداختن به مسائل غیرضروری و توضیحات اضافه در کار با نرم افزارها و همچنین برنامه نویسی اجتنا گردد.

در فصل چهارم خلاصه ای از پروژه و پیشنهاداتی جهت کامل تر کردن آن ارائه شده است.

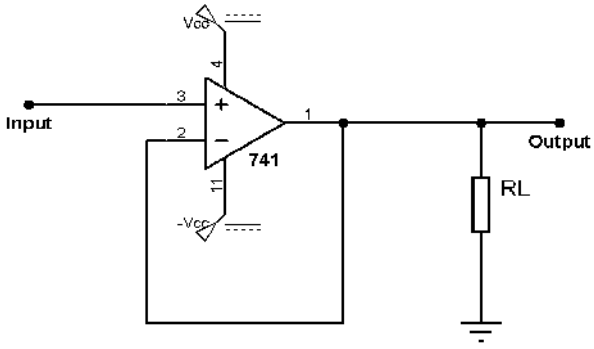
بخش پیوست ها شامل دیتاشیت های قطعات به کار رفته برای این پروژه دیتاشیت ترانزیستورها BC337 و 2N3055 و آی سی ها LM324، 741، MEGA16-AVR می باشد.

فصل دوم: تشریح بلاک دیاگرام و مدار

۲-۱- نمونه برداری از صوت و بافر کردن آن

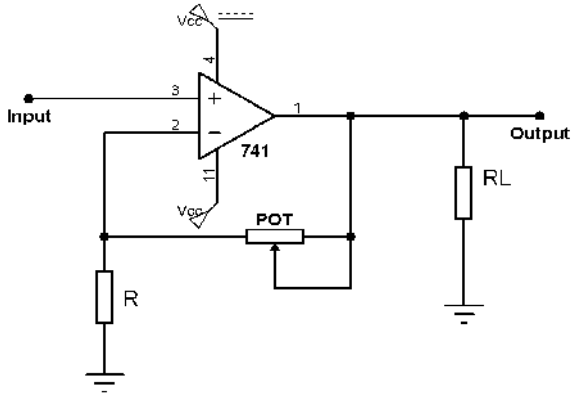
در ابتدا پیش از هرعمل یا پردازشی روی صوت باید از صوت در حال پخش نمونه برداری کرد.این نمونه برداری باید از ولتاژ بوده و می تواند از ولتاژ دوسر بلندگو باشد. نکته ای که در این جا باید در نظر گرفته شود، این است که نمونه برداری نباید درکار سیستم صوتی اختلالی ایجاد کند. بنابراین برای این که جریانی هنگام نمونه برداری از سیستم صوتی کشیده نشود، از یک طبقه بافر استفاده می کنیم. بافر مداری است که ولتاژ را انتقال داده ولی جریان بسیار کمی در حد صفر می کشد و در عین حال قابلیت جریان دهی زیاد در خروجی را داراست. به عبارت دیگر بافر نمونه بردار ولتاژی است که از بلوکی که از ان نمونه برداری می کند، جریان نمی کشد یا جریان بسیار کمی می کشد اما می تواند در خروجی با دادن همان ولتاژ نمونه برداری شده جریان بالایی را تامین کند.

پس برای این که خللی در کار سیستم صوتی ایجاد نشود، برای نمونه برداری از ولتاژ بلندگو از یک طبقه بافر استفاده می کنیم. شکل ۲-۱ یک نمونه مدار بافر با استفاده از تقویت کننده عملیاتی مورد استفاده در این پروژه را نشان می دهد.



شکل۲ـ ۱: مدار بافر

برای این که علاوه بر بافر کردن در صورت نیاز بتوان سیگنال ورودی را نیز تقویت نمود به جای مدار فوق از مدار شکل ۲ـ۲ استفاده می کنیم.



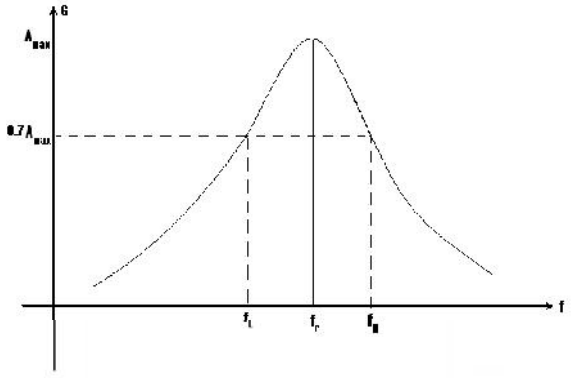
شکل۲ـ۲: مدار بافر با بهره متغیر

۲ـ۲ تفکیک مؤلفه های فرکانسی

همان طور که می دانید هر صوت از هارمونیک ها و مولفه های فرکانسی بی شماری تشکیل شده است.برای این که بتوانیم یک آبنمای اکولایزری یا رقص نور اکولایزری یا چیزی شبیه به این ها را بسازیم، ابتدا باید تعدادی از این مولفه های فرکانسی را جدا کرد و با انجام یک سری پردازش ها هر یک از این مولفه ها را به یکی از فواره ها یا ستون های رقص نور مرتبط ساخت. برای تفکیک مولفه های

مختلف فرکانسی باید از تعدادی فیلتر برای جداسازی این مولفه ها استفاده کرد. این فیلترها باید از نوع میان گذر باشند تا بتوان یک فرکانس خاص یا به عبارت بهتر یک باند فرکانسی خاص را استخراج نمود. عبارت باند فرکانسی از این نظر به کار برده شده است، که هیچ فیلتر ایده آلی در دنیا وجود ندارد و همه فیلترها دارای یک پهنای باندی می باشند که علاوه بر فرکانس مورد نظر دسته ای از فرکانس های مجاور آن را نیز عبور می دهند.

پاسخ فرکانسی یک فیلتر میان گذر به طور کلی به صورت شکل ۲-۳ می باشد.



شکل۲-۳:پاسخ فرکانسی فیلتر میان گذر

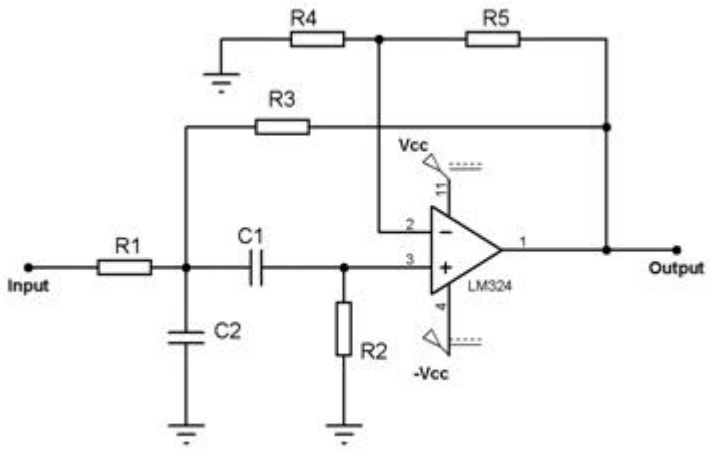
$$BW = F_H - F_L \qquad Q = \frac{F_r}{BW}$$

رابطه ۲-۱: پهنای باند و ضریب کیفیت فیلتر

۲-۳- انتخا نوع فیلترها

فیلترهای میان گذری که برای این پروژه در نظر گرفته شده است، نوع RC-Active خازنی مقاومتی فعال می باشد. علت انتخا این نوع فیلتر، تقویت کنندگی علاوه بر فیلترینگ، پاسخ مناسب تر، عدم استفاده از سلف و مزایای دیگر این نوع فیلترها می باشد.

در شکل ۲-۴ یک نمونه مدار فیلتر میان گذر RC-Active که در این پروژه نیز مورد استفاده قرار گرفته است، مشاهده می شود.



شکل۲-۴: فیلتر میان گذر فعال

عنصر فعال این فیلتر، تقویت کننده عملیاتی Op-Amp می باشد که به صورت مدار مجتمع IC در بازار موجود می باشد و معروف ترین آن آی سی 741 می باشد که یک آی سی هشت پایه است. همچنین آی سی LM324 نیز یک نمونه دیگر از تقویت کننده های عملیاتی می باشد که درون چهار آن عدد تقویت کننده عملیاتی وجود دارد و یک آی سی چهارده پایه می باشد. از نمونه های دیگر تقویت کننده های عملیاتی می توان به آی سی های 747، OP27 ، 741S، LM301 و بسیاری دیگر از آی سی های موجود در بازار اشاره کرد.

در این پروژه از آی سی های ۷۴۱ و LM324 به عنوان تقویت کننده عملیاتی در مدارها استفاده شده است. دیتاشیت آن ها در بخش پیوست ها موجود است .

بنابراین پس از نمونه برداری از صوت و بافر کردن آن به وسیله تعدادی فیلتر میان گذر که مدار آن ها در شکل بالا قابل مشاهده می باشد، هر مولفه فرکانسی یا به عبارت بهتر هر باند فرکانسی صوت را تفکیک می کنیم.

۲-۴- انتخا فرکانس عبور فیلترها

پس از انتخا نوع فیلترها، باید فرکانس عبور آن ها را انتخا نماییم. از آن جایی که مدار نهایی ما در این پروژه متناسب با صوت پخش شده که موسیقی می باشد عمل می کند، لذا بهتر است که فرکانس عبور فیلترها از فرکانس نت های موسیقی انتخا شود. اکنون به طور مختصر به بررسی نت های موسیقی و فرکانس آن ها می پردازیم تا بهتر بتوانیم فرکانس عبور فیلترها را انتخا کنیم.

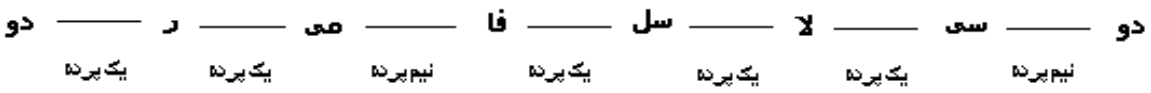
به طور کلی صداها به دو دسته صداهای غیرموسیقیایی مثل بسته شدن در،خش خش برگ ها و ... و صداهای موسیقیایی که آن ها را آوا می نامیم، تقسیم می شوند.آواهای مختلف در موسیقی دارای فرکانس های مختلف هستند که از قواعد خاصی پیروی می کنند. هر آوا در موسیقی دارای نشانه و صدای خاصی است که به آن "نت موسیقی" می گویند.نت های موسیقی در واقع نشانه هایی هستند که زیر و بمی و همچنین مدت زمان صدا را مشخص می کنند.از آن جایی که نحوه نگارش و زبان نوشتاری موسیقی و یا شکل و نشانه های موسیقی در این تحقیق و این پروژه مورد نیاز نیاز نیست، از بحث در مورد آن خودداری می شود و فقط درباره خود نت ها و فرکانس آن ها بحث می شود.

در موسیقی هفت نت اصلی وجود دارد که این هفت نت نماینده صداها از صدای بم تر تا صدای زیرتر می باشند. این نت ها به ترتیب " دو، ر، می، فا، سل، لا، سی " می باشند که نت "ر" از نت "دو" زیرترمی

باشد و همین طور نت "می" از نت "ر" زیرتر و الی آخر. نکته مهم دیگر این است که پس از نت "سی" دوباره نت "دو" است که از نت "سی" زیرتر است و این هفت نت همین طور تکرار می شوند. درواقع اگر این نت ها را مثل شکل ۲-۵ روی یک دایره در نظر بگیریم، با حرکت در جهت عقربه های ساعت هر نت از نت قبل از خود زیرتر و با حرکت در جهت خلاف عقربه های ساعت هر نت از نت قبل از خود بم تر است.

شکل ۲-۵: دایره نت ها

به فاصله هر هشت نت در موسیقی یک اکتاو گفته می شود. در موسیقی فاصله نت ها از لحاظ زیر و بمی را با اصطلاح پرده بیان می کنند. شکل ۲-۶ فاصله نت ها را از هم در یک اکتاو نشان می دهد.



شکل۲-۶: فاصله نت ها از هم

بنابراین هر اکتاو از شش پرده یا دوازده نیم پرده تشکیل شده است.همان طور که گفته شد، فاصله نت ها و فرکانس آن ها از یک قاعده خاصی پیروی می کند. در این قاعده فرکانس نت ها از نت بم تر تا نت زیرتر یک تصاعد هندسی را تشکیل می دهند که قدر نسبت آن $q = 2^{1/12} = \sqrt[12]{2}$ می باشد، یعنی اگر یک نت را نت پایه در نظر بگیریم و فرکانسآن را بدانیم به ازای هر نیم پرده زیر شدن فرکانس پایه باید در قدر نسبت ضر شود.

بنابراین فرکانس هر نت طبق فاصله ای که از نتی فرکانس آن را می دانیم، از رابطه زیر به دست می آید:

$$f = f_{base} * 2^{n/12}$$ رابطه۲-۲:

n = 1,2,3,...,12 فاصله نت موردنظر از نت پایه برحسب نیم پرده

فرمول محاسبه فرکانس یک نت

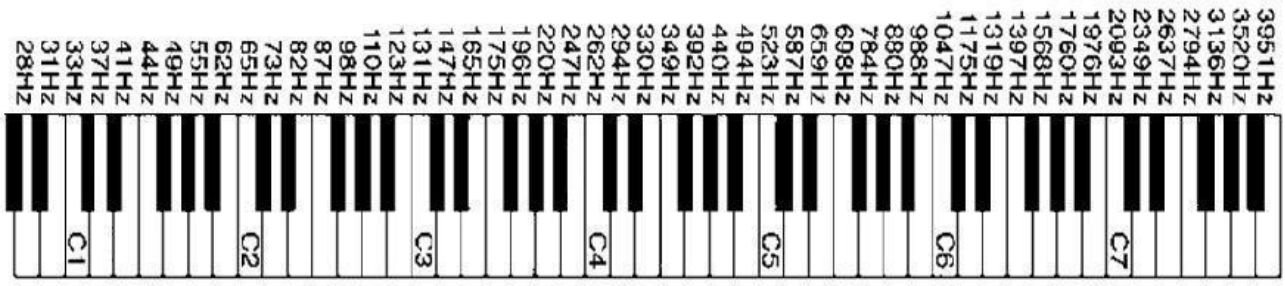
به عنوان مثال با توجه به این که فرکانس نت "لا" مرکزی پیانو *440Hz* است، فرکانس نت "سی" بعد از آن به صورت زیر محاسبه می شود:

$$f_{si} = f_{la} * 2^{2/12} = 440 * 1.122 \approx 494Hz$$

رابطه۲-۳: فرکانس نت سی

رابطه بالا نشان می دهد که نسبت فرکانس یک نت با فرکانس همان نت در اکتاو پایین تر که شش پرده یا دوازده نیم پرده فاصله دارند ، ۲ می باشد. مثلا فرکانس نت "دو" دو برابر فرکانس همان نت در اکتاو پایین تر است.

با توجه به مطالب بیان شده، فرکانس تمام نت های موسیقی قابل محاسبه بوده و در شکل ۲-۷ مشاهده می گردد.



شکل ۲-۷: فرکانس نت ها

با بررسی رقص نور های اکولایزری متوجه می شویم که در همه آن ها فرکانس های انتخا شده برای تفکیک، فرکانس نت های موسیقی یا تقریب بسیار نزدیکی از آن ها و یا هارمونیک های مرتبه بالاتر آن ها می باشد. بنابراین بهتر است فرکانس های عبور برای این پروژه هم از فرکانس نت های موسیقی انتخا شود.

فرکانس های انتخا شده برای فیلترهای این پروژه، در زیر مشاهده می شوند.

200Hz - 440Hz - 660Hz - 990Hz - 1.2KHz - 1.6KHz - 3.9KHz

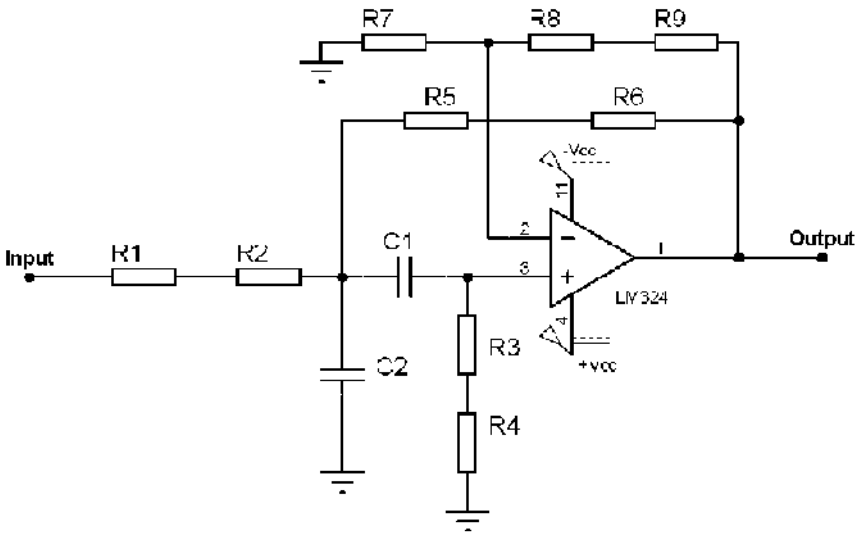
۲-۵- طراحی فیلترها

اکنون که نوع و فرکانس عبور فیلترها مشخص گردید، نحوه طراحی فیلترها مورد بررسی قرار می گیرد.

طراحی فیلتر های موجود در این پروژه به صورت نرم افزاری و به کمک نرم افزار **Filter Solution** انجام گرفته است.

نرم افزار **Filter Solution** قابلیت طراحی انواع فیلترها را دارا است؛ که قابلیت ها و چگونگی استفاده از این نرم افزار در فصل سوم به طور کامل بررسی خواهد شد.

همان طور که قبلاً گفته شد، مدار فیلتر فعال طراحی شده به کمک این نرم افزار به صورت شکل ۲-۴ می باشد.از آن جایی که در طراحی فیلترها تمامی مقادیر طراحی شده استاندارد نیستند، مدار شکل ۲-۸ به عنوان مدار نهایی فیلترهای میان گذر در نظر گرفته شده است و مقادیر المان ها به صورتی که در زیر شکل مشاهده می شود، خواهد بود. با توجه به این که همه مقدار خازن ها و مقاومت های مدار مقادیر استاندارد می باشد، برای این که هم مقادیر استاندارد باشد و هم مقدار طراحی شده برای آن ها تغییر زیادی نکند، از مقاومت های سری در این مدار زیاد استفاده شده است.



شکل ۲-۸: مدار نهایی فیلتر میان گذر

200Hz:

R1=10k - R2=470 - R3=3.9k - R4=220 - R5=2.2k - R6=330 - R7=10k - R8=12k - R9=2.2k C1=270nf - C2=270nf

440Hz:

R1=10k - R2=560 - R3=3.9k - R4=330 - R5=2.2k - R6= - R7=10k - R8=12k - R9=2.2k - C1=120nf - C2=120nf

660Hz:

R1=10k - R2=390 - R3=3.9k - R4=220 - R5=2.2k - R6=330 - R7=10k - R8=12k - R9=2.2k - C1=82nf - C2=82nf

990Hz:

R1=10k - R2=150 - R3=3.9k - R4=150 - R5=2.2k - R6=330 - R7=10k - R8=12k - R9=2.2k C1=56nf - C2=56nf

1.2KHz:

R1=10k - R2=0 - R3=3.9k - R4=100 - R5=2.2k - R6=330 - R7=10k - R8=12k - R9=2.2k - C1=47nf - C2=47nf

1.6KHz:

R1=10k - R2=680 - R3=3.9k - R4=330 - R5=2.2k - R6=470 - R7=10k - R8=12k - R9=2.2k C1=33nf - C2=33nf

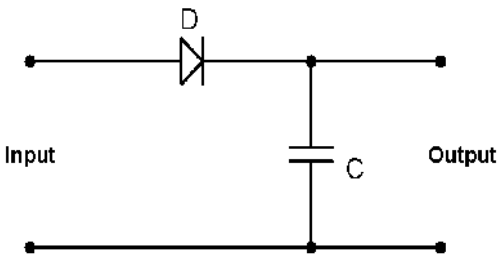
3.9KHz:

R1=8.2k - R2=1.2k - R3=3.9k - R4=0 - R5=2.2k - R6=330 - R7=10k - R8=12k - R9=2.2k - C1=15nf - C2=15nf

۲-۶- تبدیل سیگنال تفکیک شده هر مولفه به موج DC

پس از این که هفت مولفه فرکانسی صوت توسط هفت فیلتر میان گذر طراحی شده استخراج گردید، این مولفه ها باید پیش از اعمال به بخش مبدل آنالوگ به دیجیتال (ADC میکروکنترلر، توسط مداری به یک موج dc متناسب با دامنه هر مولفه تبدیل شود، یعنی هفت موج استخراج شده از صوت به هفت ولتاژ dc متغیر با دامنه این موج ها تبدیل شود. مداری که برای این کار در نظر گرفته شده است، یک مدار آشکارساز پیک موج (Peak Detector می باشد.

شکل ۲-۹ یک مدار ساده آشکارساز پیک موج را نشان می دهد.



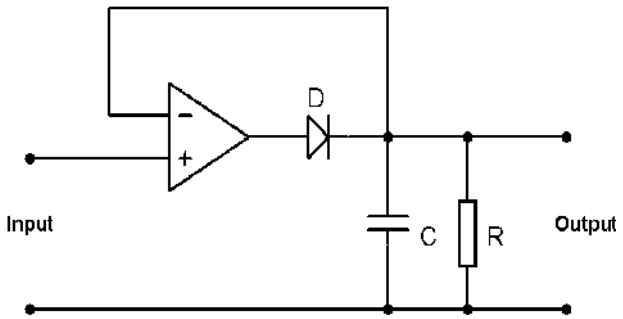
شکل ۲-۹: مدار آشکارساز پیک ساده

همان طور که در شکل ۲-۹ مشاهده می شود، دیود قسمت های منفی موج را حذف کرده و پس از آن خازن به اندازه ولتاژ پیک موج ورودی شارژ می شود و آن را در خود ذخیره می کند. اگر دامنه ورودی زیاد شود، خازن دوباره شروع به شارژ کرده ومقدار پیک موج ورودی را در خود ذخیره می کند.

عیب این مدار این است که دامنه ورودی کم شود دیگر خازن نمی تواند ولتاژ پیک ورودی را دنبال کرده و همچنان ولتاژ پیک قبلی که مقدار بیشتری دارد را در خود نگه می دارد. برای رفع این عیب باید یک مقاومت نسبتاً زیاد با خازن موازی کرد که مسیری برای دشارژ خازن ایجاد نموده و در صورت کاهش دامنه ورودی باز هم خازن بتواند مقدار پیک آن را دنبال کند.

عیب دیگر مدار فوق این است که خروجی به میزان ولتاژ دیود از مقدار پیک ورودی کم تر است.

همچنین پس از افزودن مقاومت، با اتصال به طبقه بعد و یا به طور کلی پس از کشیدن هر جریان از مدار، خروجی مدار دیگر dc ثابت نخواهد بود و مقداری رایپل متناسب با جریان کشیده شده خواهد داشت. برای رفع این اشکالات نیز از یک مدار آشکارساز پیک فعال که در آن از یک OpAmp استفاده شده است، بهره می گیریم. بنابراین مدار نهایی آشکارساز پیک مورد استفاده، به صورت شکل ۲-۱۰ خواهد بود.



شکل ۲-10: مدار آشکارساز پیک

پس از طبقه آشکارساز پیک که سیگنال ac هر مولفه را به یک موج dc متناسب با دامنه آن مولفه تبدیل می کند، یک طبقه بافر دیگر قرار داده تا در کشیده شدن جریان از این طبقه توسط طبقه بعد، هرچند ناچیز، اختلالی در عملکرد مدار آشکارساز پیک ایجاد نشود.

شکل مدار بافر موردنظر که به آن اشاره شد، همان بافری است که در شکل ۲-۲ آمده است.

۲-۷- بخش میکروکنترلر و پردازش

اکنون که صوت ورودی را به هفت موج dc متناسب با مولفه های آن تبدیل کردیم، زمان آن رسیده که این موج ها یا درواقع ولتاژها را به هفت موج PWM برای طبقه راه انداز پمپ های خروجی تبدیل نماییم. برای این کار از یک میکرو کنترلر AVR AVR-ATMEGA16 استفاده می کنیم. دیتاشیت آی سی AVR-ATMEGA16 در بخش پیوست ها موجود می باشد .

نرم افزار مورد استفاده برای برنامه نویسی برای میکروکنترلر AVR در این پروژه" CodeVision " بوده که از زبان " C " برای برنامه نویسی در آن استفاده می شود.

نحوه استفاده از میکروکنترلر AVR بدین صورت می باشد که ابتدا هفت ولتاژ مربوط به هفت مولفه را به ورودی های ADC مبدل آنالوگ به دیجیتال میکروکنترلر داده و سپس با استفاده از این مقادیر خوانده شده در برنامه هفت موج PWM مدنظر را ایجاد می نماییم. نکته ای که در اینجا باید در نظر گرفته شود، این است که از خروجی های PWM خود میکروکنترلر نمی توان استفاده کرد زیرا در اکثر میکروکنترلرها یک یا دو خروجی PWM بیشتر وجود نداشته در صورتی در این پروژه حداقل به هفت موج PWM نیاز است. بنابراین این هفت موج PWM را با استفاده از برنامه نویسی باید ایجاد نمود.

۲-۷-۱- توضیح مختصری درباره میکروکنترلرهای AVR

میکروکنترلرهای AVR میکروکنترلرهای هشت بیتی ساخت شرکت ATMEL می باشد.

خانواده میکروکنترلرهای AVR دارای محدوده نسبتا وسیعی می باشند و دارای سه خانواده کلی زیر است:

۱- ATTINY AVR

۲- AT90S سری کلاسیک

۳- ATMEGA AVR

این میکروکنترلرها بر مبنای معماری RISC بهبود یافته طراحی شده است. Reduced RISC Instruction Set Computer- کامپیوتر با مجموعه دستورات کاهش یافته را می توان در مقابل CISC کامپیوتر با مجموعه دستورات پیچیده قرار داد. RISC معماری از CPU می باشد که مجموعه دستورات آن سریع الاجرا و ساده است ولی نوشتن برنامه را مشکل تر اما سریع تر می کند.

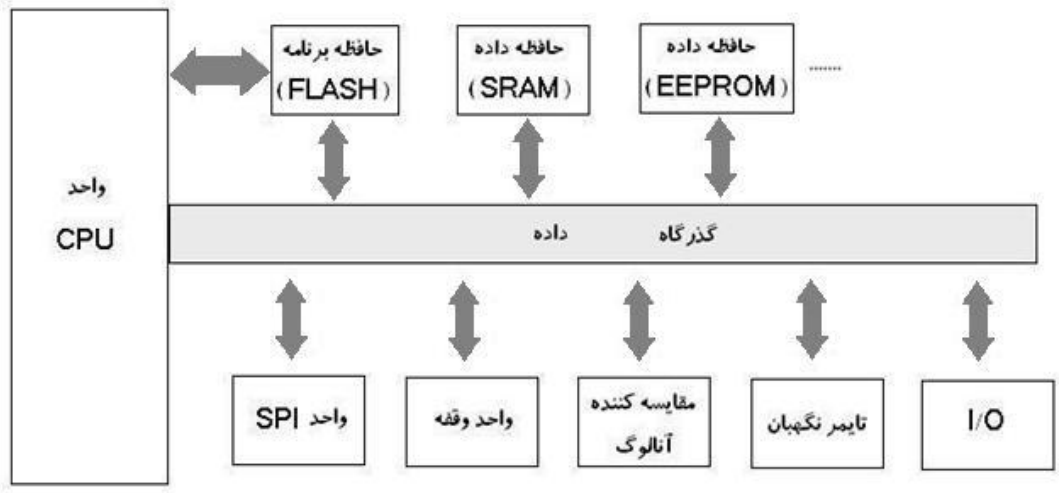
از امکانات جانبی میکروکنترلرهای AVR که تقریبا در همه آن ها مشترک است، می توان به موارد زیر اشاره کرد:

- مبدل آنالوگ به دیجیتال ۱۰ بیتی ADC
- مقایسه کننده آنالوگ
- تایمر/ کانتر با خروجی PWM
- واسط ارتباط سریال 12C 2wire
- واسط ارتباط سریال USART
- تایمر نگهبان Watchdog

۲-۷-۲- ساختار پردازنده AVR

در میکروکنترلرهای AVR یک واحد مرکزی وجود دارد که تمام فعالیت های میکروکنترلر را مدیریت و تمام عملیات لازم را بر روی داده ها انجام می دهد. همچنین وظیفه ارتباط با حافظه ها و کنترل تجهیزات جانبی را به عهده دارد. به این واحد MCU می گویند.

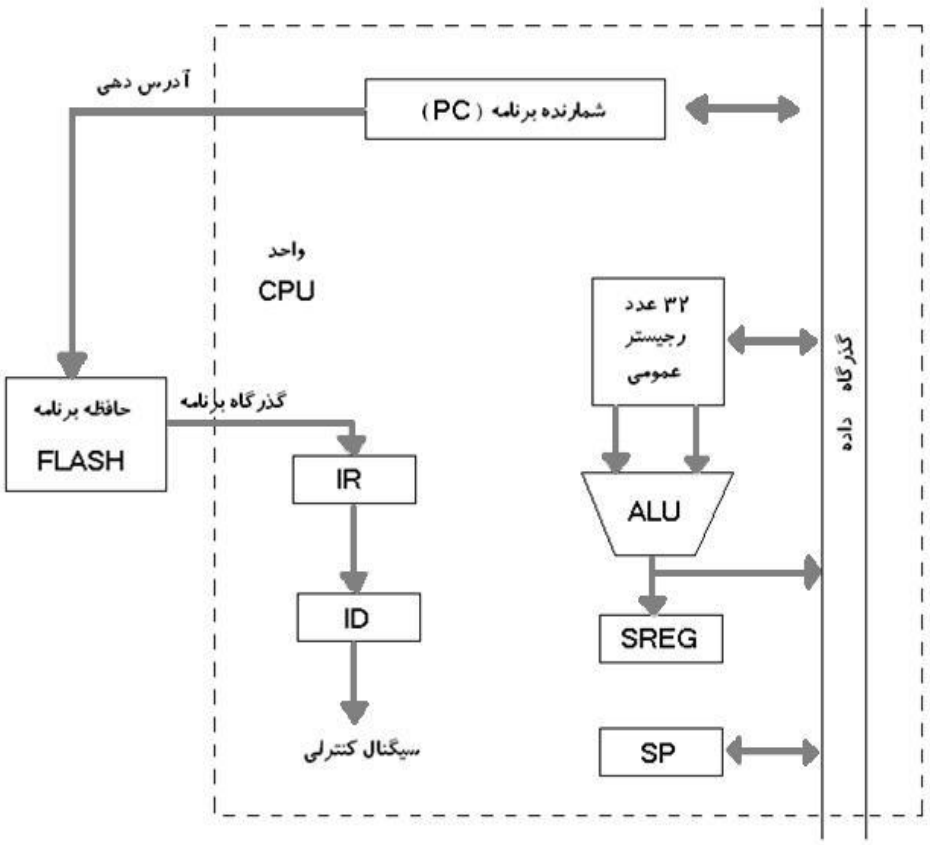
شکل ۲-۱۱ ساختار کلی MCU Master Control Unit را نشان می دهد.



شکل۲-۱۱:

ساختارMCU

شکل ۲-۱۲ هم ساختار واحد CPU را نشان می دهد.



شکل ۲ـ۱۲: واحدCPU

همان طور که مشاهده می شود، شامل بخش هایی زیر می باشد:

- رجیسترهای عمومی که ۳۲ عدد بوده و از R0 تا R31 که هرکدام هشت بیتی بوده، می باشند.

- واحد ALU که وظیفه انجام کلیه عملیات ریاضی و منطقی را به عهده دارد و با رجیسترهای R0 تا R32 به طور مستقیم در ارتباط است.

- IR رجیستر دستور کد دستورالعملی که از حافظه برنامه FLASH خوانده شده و باید اجرا شود را در خود داراست.

- ID واحد رمزگشایی دستور که تشخیص می دهد که کد واقع در IR مربوط به کدام دستورالعمل است و سیگنال های کنترلی برای اجرای آن را صادر می نماید.

- PC شمارنده برنامه آدرس دستورالعمل بعدی که باید اجرا شود را در خود جای داده و با اجرای هر دستورالعمل محتوای آن افزایش یافته و به دستورالعمل بعدی اشاره می کند.

۲-۷-۳- نوشتن برنامه و برنامه ریزی میکروکنترلر

همان طور که قبلا اشاره شد، برای نوشتن برنامه برای میکروکنترلر از زبان C و نرم افزار CodeVision استفاده می کنیم.

چگونگی نوشتن برنامه به زبان C و کار با نرم افزار CodeVision در فصل بعد مورد بحث و بررسی قرار خواهد گرفت.

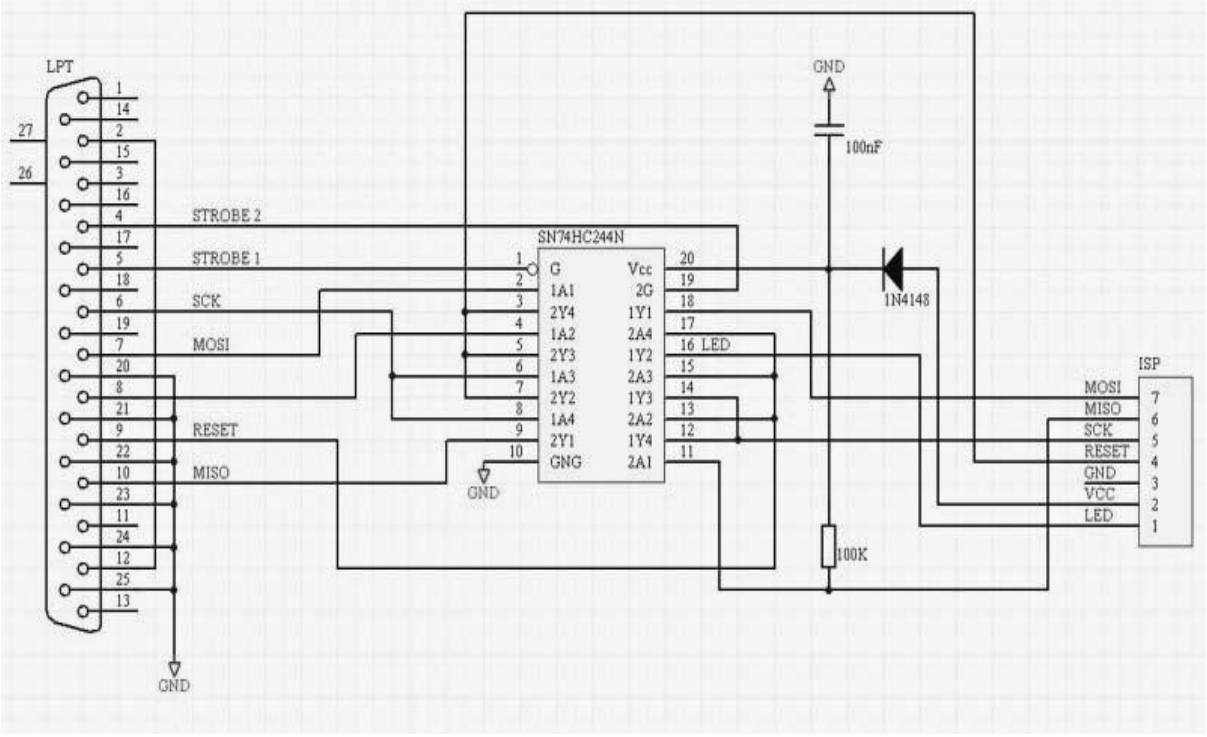
کاری که قرار است میکروکنترلر انجام دهد و چگونگی آن در فصل بعد توضیح داده می شود؛ این است که پس از جداسازی مؤلفه های فرکانسی صوت و تبدیل آن ها به ولتاژهای dc، میکروکنترلر این ولتاژهای آنالوگ را به مقادیر دیجیتال تبدیل کرده و سپس با توجه به مقدار دیجیتال شده هر مؤلفه، به تعداد مؤلفه های جدا شده که هفت مؤلفه می باشد، موج های PWM برای راه اندازی پمپ های آ ایجاد نماید.

۲-۷-۴- پروگرامر و پروگرام کردن میکروکنترلر

برای برنامه ریزی یا پروگرام کردن قبل از هر چیز به یک مدار واسط بین میکروکنترلر و کامپیوتر یا "پروگرامر" نیاز است.

برای پروگرام کردن میکروکنترلرهای AVR از پروگرامرهای مختلفی می توان استفاده کرد. لیست مدل های مختلف پروگرامرها که با استفاده از نرم افزار CodeVision قابل پروگرام کردن می باشند را می توان در پنجره ای که با انتخا گزینه Programmer از منوی Setting در محیط CodeVision ظاهر می شود، مشاهده نمود.

پروگرامری که از آن برای پروگرام کردن میکروکنترلر دراین پروژه از آن استفاده شده است، STK200+/300 می باشد. مدار این پروگرامر در شکل ۲-۱۳ قابل مشاهده می باشد.



شکل ۲-۱۳: پروگرامر STK200+/300

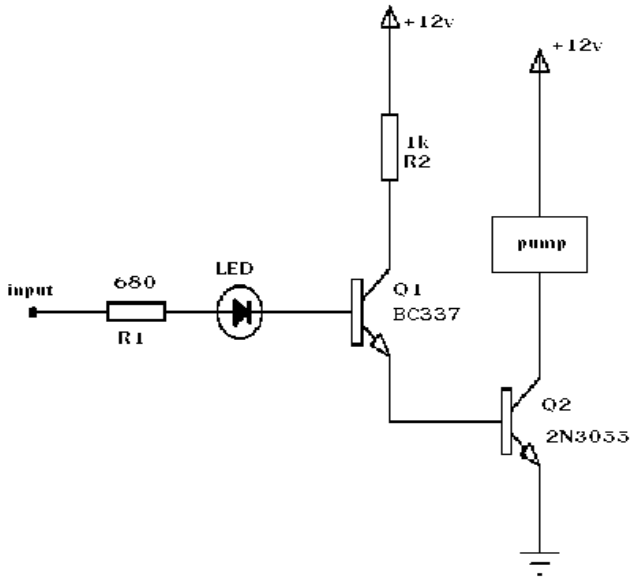
پس از این که با استفاده از پروگرامری که اشاره شد، توسط پورت پارالل کامپیوتر با میکروکنترلر ارتباط برقرار شد، می توان توسط نرم افزار CodeVision میکروکنترلر را برنامه ریزی نمود.

همانند سایر امور نرم افزاری چگونگی پروگرام کردن میکروکنترلر نیز در فصل بعد توضیح داده شده است.

۲-۸- طبقه راه انداز پمپ های آ

از آنجایی که پمپ های آ فواره ها دارای ولتاژ نامی ۱۲ ولت و جریانی بیش از ۲ آمپر است و خروجی های میکروکنترلر نمی تواند تامین کننده این جریان و ولتاژ باشد، بنابراین از یک طبقه راه انداز قبل از پمپ ها استفاده می کنیم تا براساس ورودی اش موج های PWM خروجی میکروکنترلر در خروجی اش پمپ های آ را راه اندازی نماید.

شکل ۲-۱۴ مدار این طبقه راه انداز را نشان می دهد.



شکل ۲-۱۴: مدار درایور

همان طور که در شکل مشاهده می شود، از دو طبقه ترانزیستور پشت سر هم به صورت زوج دارلینگتون برای راه اندازی پمپ ها استفاده شده است. ترانزیستور اول جریان بیس ترانزیستور دوم را تامین کرده و ترانزیستور دوم جریان خروجی را تامین می کند. ورودی مدار هم همان خروجی های طبقه قبل موج های PWM ایجاد شده توسط میکروکنترلر می باشد.

وظیفه المان های مدار به طور خلاصه به شرح زیر است:

- Q1 : ترازیستور NPN معمولی جهت تامین جریان بیس Q2 مدل:BC337
- Q2 : ترانزیستور NPN قدرت جهت تامین جریان خروجی مدل:2N3055
- R1 : مقاومت بیس Q1 جهت محدود کردن جریان بیس Q1 و LED که از میکروکنترلر کشیده می شود.
- R2 : مقاومت کلکتور Q1 برای محدود کردن جریان کلکتور Q1 و بیس Q2
- LED نشان دهنده سطح dc موج PWM که کم و زیاد شدن نور آن تغییرات سطح dc را رؤیت پذیر می نماید.

۲-۹- پمپ های آ خروجی

همان طور که قبلا اشاره شد،هفت پمپ آ جهت ایجاد هفت فواره مورد نیاز است. پمپ های استفاده شده در این پروژه، پمپ های معمولی با ولتاژ نامی ۱۲ ولت و جریان حدود ۲.۵ آمپر می باشند که توسط مدار درایور،راه اندازی شده و ارتفاع های مختلفی از فواره های آ را می توانند ایجاد کنند.

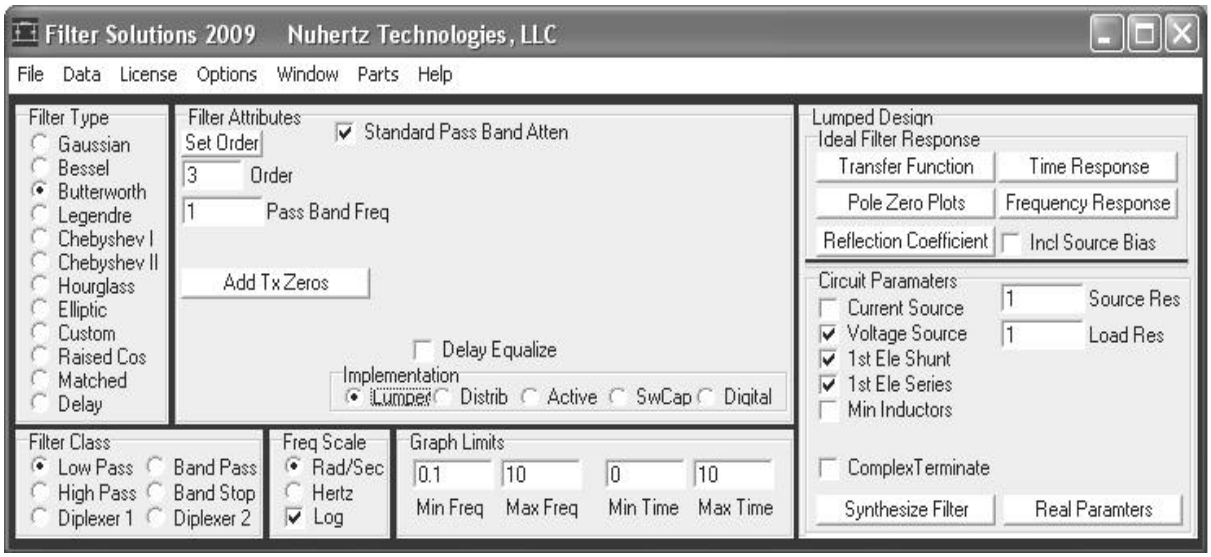
فصل سوم: تشریح نرم افزارها و برنامه ها

۳-۱- طراحی فیلتر به کمک نرم افزار " Filter Solution "

همان طور که در قبلاً اشاره شد، طراحی فیلتر های موجود در این پروژه به صورت نرم افزاری و به کمک نرم افزار **Filter Solution** انجام گرفته است.

با استفاده از نرم افزار **Filter Solution** می توان هر نوع فیلتری را طراحی نمود. این نرم افزار قابلیت طراحی کلاس های مختلف فیلتر (میان گذر، پایین گذر و...) و انواع فیلترها (باترورث، چپی چف، بسل، تاخیری و...) را دارا می باشد. با استفاده از آن می توان فیلترهای اکتیو یا پسیو را طراحی کرد. همچنین این نرم افزار قابلیت طراحی فیلترهای مرتبه بالا (تا مرتبه ۲۱) را داراست. به دلیل قابلیت های خوب این نرم افزار در طراحی فیلترها، از آن برای طراحی فیلترهای مورد نظر در این پروژه استفاده گردیده است.

شکل ۳-۱ پنل اصلی نرم افزار **Filter Solution** نشان می دهد.



شکل ۳ـ ۱: پنل اصلی Filter Solution

همان طور که مشاهده می شود، در قسمت چپ صفحه می توان نوع فیلتر را مشخص نمود. در پایین آن کلاس فیلتر تعیین می شود. قسمت مرکزی پنل مربوط انتخاب مرتبه فیلتر و فرکانس عبور یا قطع و همچنین نوع پیاده سازی فیلتر می باشد و در نهایت قسمت راست مربوط به پاسخ فیلتر و پارامترهای مداری آن می باشد.

پس از وارد کردن تمامی ورودی ها، با زدن دکمه Synthesize Filter (ترکیب فیلتر) مدار نهایی فیلتر مورد نظر را مشاهده خواهیم کرد.

در این جا تمام بخش های موجود در صفحه اصلی این نرم افزار توضیح داده می شود.

Filter Type:

این بخش برای تعیین نوع فیلتر می باشد. در این قسمت دوازده نوع فیلتر نظیر باترورث، چپی چف، بسل، تاخیری و... قابل انتخاب می باشد.

Filter Class:

کلاس فیلتر (پایین گذر، میان گذر، بالاگذر و...) در این بخش انتخاب می شود.

Filter Attributes:

این بخش برای تعیین مشخصات فیلتر می باشد. مرتبه فیلتر، فرکانس مرکزی و پهنای باند یا فرکانس های قطع فیلتر را در این بخش می توان تعیین نمود.

Implementation:

نوع پیاده سازی فیلتر در این قسمت تعیین می شود. مثلاً در این پروژه از فیلترهای فعال (Active) استفاده شده است.

Freq. Scale:

مقیاس یا شاخص فرکانس را نشان می دهد که می تواند بر حسب هرتز(Hz) یا رادیان بر ثانیه(Rad/sec) باشد. به عبارت دیگر نشان دهنده این است که پارامترهای مربوط به فرکانس نظیر فرکانس قطع بر فرکانس دلالت دارد یا سرعت زاویه ای را نشان می دهد.

Graph Limits:

حدود گرافیکی یا محدوده های نمودارها را نشان می دهد. (حداکثر یا حداقل مقادیر روی محور زمان یا فرکانس در نمودارها)

Ideal Filter Response:

پاسخ ها و نمودارهای فیلتر طراحی شده(البته نمودارهای فیلتر در حالت ایده آل)را در این بخش می توان مشاهده نمود. نمودارهای تابع انتقال(Transfer Function)، پاسخ زمانی(Time Response)، نمودار صفر و قطب(Pole Zero Plots)، پاسخ فرکانسی(Frequency Response) و ضریب انعکاس(Reflection coefficient) را در این قسمت می توان انتخاب و مشاهده کرد.

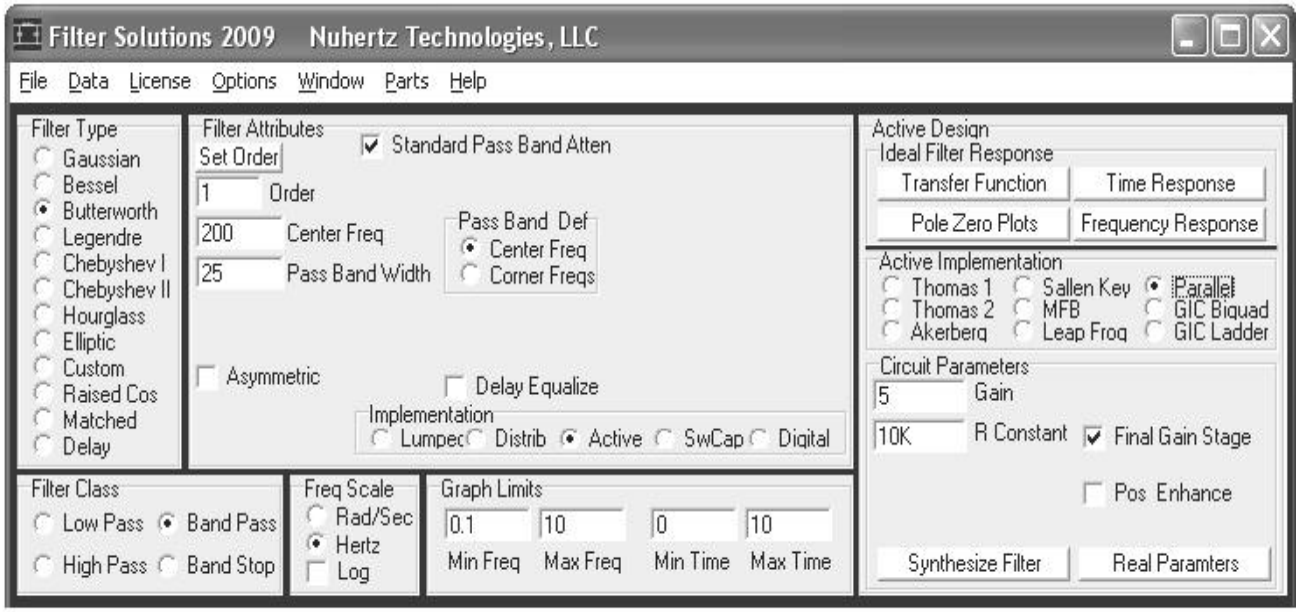
Circuit Parameter:

پارامترهای مداری نظیر مقدار مقاومت یا بهره مدار در این بخش قابل تعیین است.

Synthesize Filter:

پس از تعیین تمامی مقادیر و پارامترهای طراحی فیلتر، با انتخاب این گزینه مدار طراحی شده فیلتر در یک پنجره جداگانه روی صفحه ظاهر می شود.

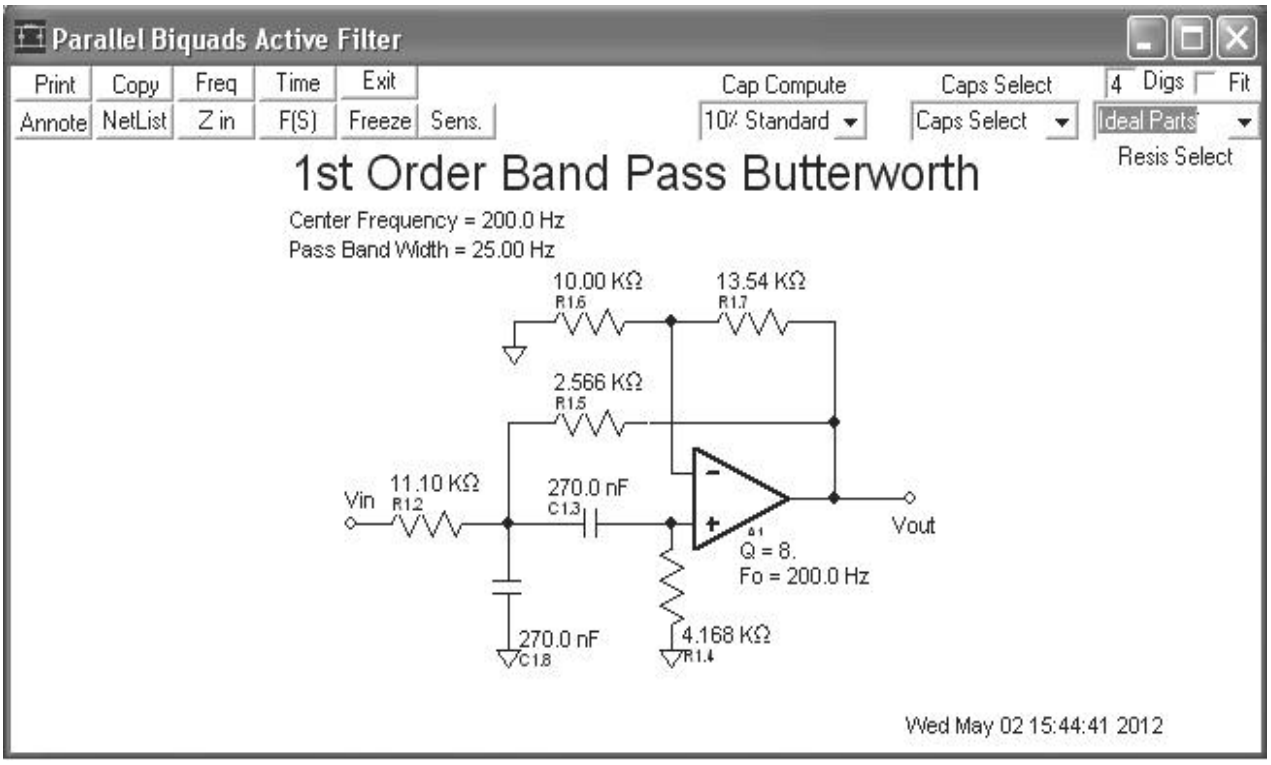
به عنوان مثال شکل ۳-۲ صفحه اصلی نرم افزار Filter Solution در حالی که تنظیمات ورودی برای فیلتر 200Hz استفاده شده در این پروژه به آن داده شده است، نشان می دهد.



شکل ۳-۲: تنظیمات فیلتر 200Hz

همان طور که گفته شد، با زدن دکمه ترکیب مدار، پنجره ای همانند شکل ۳-۳ که نشان دهنده مدار فیلتر و یک سری تنظیمات است در صفحه ظاهر می گردد. در قسمت بالای آن بخشی به نام Cap Compute می باشد که تعیین کننده مقدار خازن هایی که به ازای آن طراحی بقیه مقادیر صورت می گیرد، می باشد. در این جا مقدار خازن بر حسب استاندارد ۱۰٪ انتخاب گردیده است تا بر اساس خازن

های استاندارد موجود در بازار طراحی انجام شود. قسمت های Resis Select و Caps Select مربوط به این است که در صورتی مقادیر مقاومت ها و خازن های طراحی شده استاندارد نبود، مقادیر آن ها را به نزدیک ترین مقدار استاندارد تغییر دهد که البته در این صورت فرکانس عبور و پهنای باند کمی تغییر خواهد کرد و در زیر مدار نشان داده خواهد شد.



شکل ۳-۳: مدار طراحی شده

از آن جایی که همه مقادیر خازن ها و مقاومت های طراحی شده، مقادیر استاندارد نیستند؛ همان طور که در فصل قبل نیز به آن اشاره شد، از مدار شکل ۲-۸ به عنوان مدار نهایی فیلتر ها استفاده شده است تا هم مقادیر استاندارد باشند و هم مقدار طراحی شده برای آن ها زیاد تغییر نکند.

برای شش فیلتر دیگر هم همین تنظیمات اعمال شده و فقط فرکانس و پهنای باند آن ها متفاوت می باشد.در فصل ۲ تمامی مقادیر طراحی شده برای هر هفت فیلتر مورد نیاز در این پروژه آورده شد.

ضریب کیفیت در نظر گرفته شده برای طراحی تمامی فیلترها Q = 8 می باشد.

۳-۲- برنامه نویسی به زبان C

همان طور که اشاره شد، برای نوشتن برنامه برای میکروکنترلر از زبان C و نرم افزار CodeVision استفاده می کنیم.

شکل کلی هر برنامه در زبان C به صورت زیر می باشد:

```
# include < فایل سرتیتری .h >

# include < فایل سرتیتری .h >

…

معرفی و تعریف توابع

تعریف متغیرها و ثابت ها

…

Void  main (void)

{

…

برنامه اصلی

…
```

}

در اینجا تعدادی از دستورات زبان C در محیط CodeVision که در این پروژه استفاده شده است،به طور خلاصه توضیح داده می شود:

تعریف یک متغیر: (char,int,…) نام متغیر [=مقدار] ;

نوع متغیر

تعیین جهت یک پورت(مثلا PORTA)به عنوان خروجی: DDRA = 0xFF ;

تعیین جهت یک پورت(مثلا PORTA)به عنوان ورودی: DDRA = 0x00 ;

OR کردن دو متغیر: C = A | B ;

AND کردن دو متغیر: C = A & B ;

دستور شرطی if :

if (شرط اجرای دستور)

{

دستورات

}

else

{

دستورات

}

OR در حالت شرط(مثلا در دستور if): (A || B)

AND در حالت شرط: (A && B)

دستور حلقه for :

for (ابتدا=متغیر ;انتها=متغیر ;گام)

{

دستورات

}

ایجاد تاخیر(میکروثانیه): delay_us(مقدار) ;

ایجاد تاخیر(میلی ثانیه): delay_ms(مقدار) ;

خواندن ADC کانال nاُم: نام متغیر دلخواه = read_adc(n) ;

۲-۳-۱- برنامه نهایی برای میکروکنترلر

با توجه به توضیحات اشاره شده، برنامه نهایی برای میکروکنترلر به صورت زیر می باشد:

```
#include <mega16.h>
#include <delay.h>
#define ADC_VREF_TYPE 0x60
unsigned char adc[8];
int i,j,k;
ADCSRA|=0x40;
while ((ADCSRA & 0x10)==0);
ADCSRA|=0x10;
return ADCH;
}
#include <spi.h>
void ADC_MODE (void)
{
    if (adc!=0)PORTC = 255;
     else PORTC=0;
      for (i=0;i<8;i++)
      {
       adc[i] = read_adc(i);
      }
      for (i=0;i<=255;i++)
      {
        if (adc[0]==i & adc[0] !=255)
PORTC.0=0;
        if (adc[1]==i & adc[1] !=255)
PORTC.1=0;
unsigned char read_adc(unsigned
char adc_input)
{
ADMUX=adc_input |
(ADC_VREF_TYPE & 0xff);
delay_us(10);
        if (adc[2]==i & adc[2] !=255)
PORTC.2=0;
        if (adc[3]==i & adc[3] !=255)
PORTC.3=0;
        if (adc[4]==i & adc[4] !=255)
PORTC.4=0;
        if (adc[5]==i & adc[5] !=255)
PORTC.5=0;
        if (adc[6]==i & adc[6] !=255)
PORTC.6=0;
        if (adc[7]==i & adc[7] !=255)
PORTC.7=0;

        delay_us(1);
      }
}
void MODE1 (void)
{
   for (j=0;j<250;j++)
    {
    PORTC = 255;
    for (i=0;i<=255;i++)
```

```
    {
      delay_us(1);
      if (i==150)PORTC=0;
    }
    }
    PORTC = 255;
    delay_ms(1200);
    PORTC= 0 ;
    delay_ms(650);
}
void MODE2 (void)
{
        PORTC=1;
        delay_ms(500);
        PORTC=2;
        delay_ms(500);
        PORTC=4;
        delay_ms(500);
        PORTC=8;
        delay_ms(500);
        PORTC=16;
        delay_ms(500);
        PORTC=32;
        delay_ms(500);
        PORTC=64;
        delay_ms(500);
        PORTC=128;
        delay_ms(500);
}
void MODE3 (void)
{
        PORTC=~1;
        delay_ms(500);
        PORTC=~2;
        delay_ms(500);
        PORTC=~4;
        delay_ms(500);
        PORTC=~8;
        delay_ms(500);
        PORTC=~16;
        delay_ms(500);
        PORTC=~32;
        delay_ms(500);
        PORTC=~64;
        delay_ms(500);
        PORTC=~128;
        delay_ms(500);
}
void MODE4 (void)
{
```

```
        PORTC=1|2;
        delay_ms(500);
        PORTC=2|4;
        delay_ms(500);
        PORTC=4|8;
        delay_ms(500);
        PORTC=8|16;
        delay_ms(500);
        PORTC=16|32;
        delay_ms(500);
        PORTC=32|64;
        delay_ms(500);
        PORTC=64|128;
        delay_ms(500);
        PORTC=128|1;
        delay_ms(500);
}
void MODE5 (void)
{
        PORTC=1|4;
        delay_ms(500);
        PORTC=2|8;
        delay_ms(500);
        PORTC=4|16;
        delay_ms(500);
        PORTC=8|32;
        delay_ms(500);
        PORTC=16|64;
        delay_ms(500);
        PORTC=32|128;
        delay_ms(500);
}
void MODE6 (void)
{
        PORTC=1|128;
        delay_ms(500);
        PORTC=2|64;
        delay_ms(500);
        PORTC=4|32;
        delay_ms(500);
        PORTC=8|16;
        delay_ms(500);
        PORTC=32|4;
        delay_ms(500);
        PORTC=64|2;
        delay_ms(500);
}
void MODE7 (void)
{
        PORTC=1;
```

```
delay_ms(500);
PORTC=1|2;
delay_ms(500);
PORTC=1|2|4;
delay_ms(500);
PORTC=1|2|4|8;
delay_ms(500);
PORTC=1|2|4|8|16;
delay_ms(500);
PORTC=1|2|4|8|16|32;
delay_ms(500);
PORTC=1|2|4|8|16|32|64;
delay_ms(500);
PORTC=1|2|4|8|16|32|64|128;
delay_ms(500);
PORTC=~1;
delay_ms(500);
PORTC=~(1|2);
delay_ms(500);
PORTC=~(1|2|4);
delay_ms(500);
PORTC=~(1|2|4|8);
delay_ms(500);
PORTC=~(1|2|4|8|16);
delay_ms(500);
PORTC=~(1|2|4|8|16|32);
delay_ms(500);
PORTC=~(1|2|4|8|16|32|64);
delay_ms(500);

PORTC=~(1|2|4|8|16|32|64|128);
delay_ms(500);
}
void MODE8 (void)
{
PORTC=1;
delay_ms(500);
PORTC=1|2;
delay_ms(500);
PORTC=1|2|4;
delay_ms(500);
PORTC=1|2|4|8;
delay_ms(500);
PORTC=1|2|4|8|16;
delay_ms(500);
PORTC=1|2|4|8|16|32;
delay_ms(500);
PORTC=1|2|4|8|16|32|64;
delay_ms(500);
PORTC=1|2|4|8|16|32|64|128;
delay_ms(500);
```

```
        PORTC=2|4|8|16|32|64|128;
        delay_ms(500);
        PORTC=4|8|16|32|64|128;
        delay_ms(500);
        PORTC=8|16|32|64|128;
        delay_ms(500);
        PORTC=16|32|64|128;
        delay_ms(500);
        PORTC=16|32|64|128;
        delay_ms(500);
        PORTC=32|64|128;
        delay_ms(500);
        PORTC=64|128;
        delay_ms(500);
        PORTC=128;
        delay_ms(500);
}
void MODE9 (void)
{
   for (j=0;j<200;j++)
    {
    PORTC = 255;
    for (i=0;i<=255;i++)
    {
      delay_us(1);
      if (i==80)PORTC.0=0;
      if (i==80)PORTC.7=0;
      if (i==130)PORTC.1=0;
      if (i==130)PORTC.6=0;
      if (i==190)PORTC.2=0;
      if (i==190)PORTC.5=0;
    }
    }
    for (j=0;j<200;j++)
    {
    PORTC = 255;
    for (i=0;i<=255;i++)
    {
      delay_us(1);
      if (i==80)PORTC.3=0;
      if (i==80)PORTC.4=0;
      if (i==130)PORTC.2=0;
      if (i==130)PORTC.5=0;
      if (i==190)PORTC.1=0;
      if (i==190)PORTC.6=0;
    }
    }

}
void MODE10 (void)
```

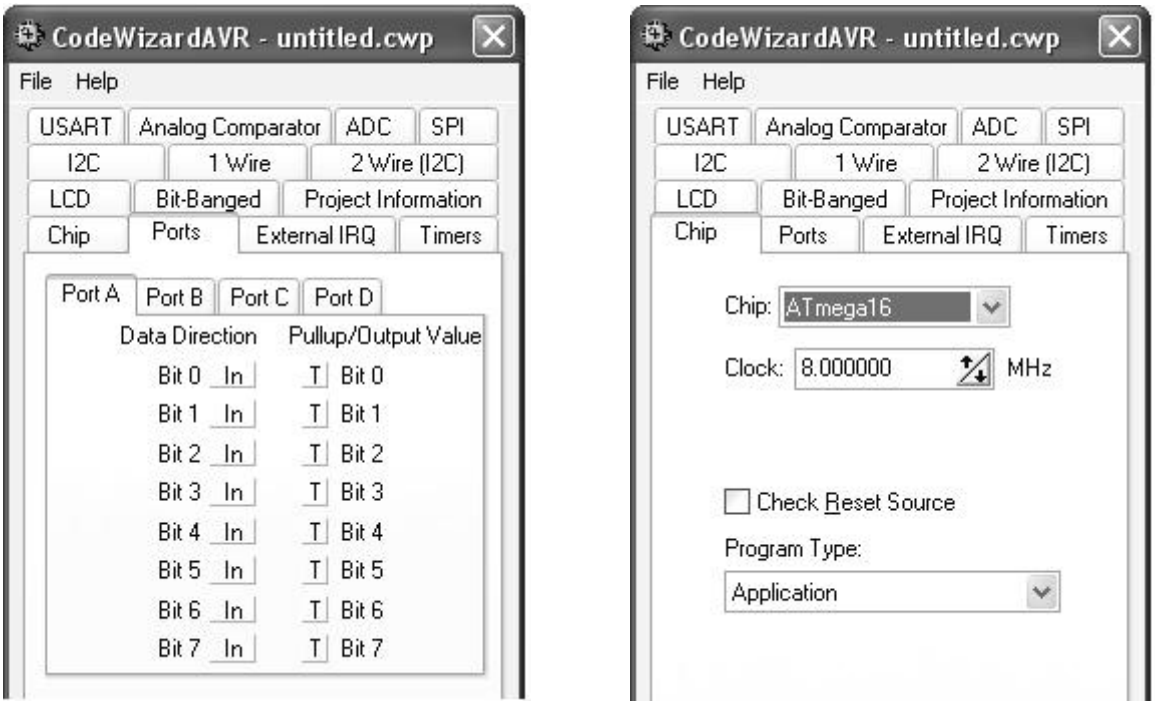
```
{
        PORTC=1|4|16|64;
        delay_ms(500);
        PORTC=2|8|32|128;
        delay_ms(500);
}
void MODE_SELECT (int m)
{
    if(m>=0  && m< 10) MODE1();
    if(m>=10 && m< 20) MODE2();
    if(m>=20 && m< 30) MODE3();
    if(m>=30 && m< 40) MODE4();
    if(m>=40 && m< 50) MODE5();
    if(m>=50 && m< 60) MODE6();
    if(m>=60 && m< 70) MODE7();
    if(m>=70 && m< 80) MODE8();
    if(m>=80 && m< 90) MODE9();
    if(m>=90 && m<100) MODE10();
}
void IDLE_MODE (void)
{
    for (k=0;k<100;k++)
        MODE_SELECT(k);
}
void main(void)
{
PORTA=0x00;
DDRA=0x00;
PORTB=0x00;
DDRB=0x40;
PORTC=0x00;
DDRC=0xFF;
PORTD=0x00;
DDRD=0x00;
PORTC= 0xff;
while (1)
    {
      if (PINB.0==1)
          ADC_MODE();
      if (PINB.0==0)
          IDLE_MODE();
     };
}
```

۳-۳- کار با نرم افزار CodeVision

پس از باز کردن نرم افزار CodeVision،ایجاد یک پروژه جدید و ذخیره آن؛ می توان شروع به برنامه نویسی کرد.چگونگی نوشتن برنامه به زبان C در محیط CodeVision در قسمت قبل به طور خلاصه شرح داده شد.اکنون برخی دیگر از نکات مربوط به این نرم افزار شرح داده می شود.

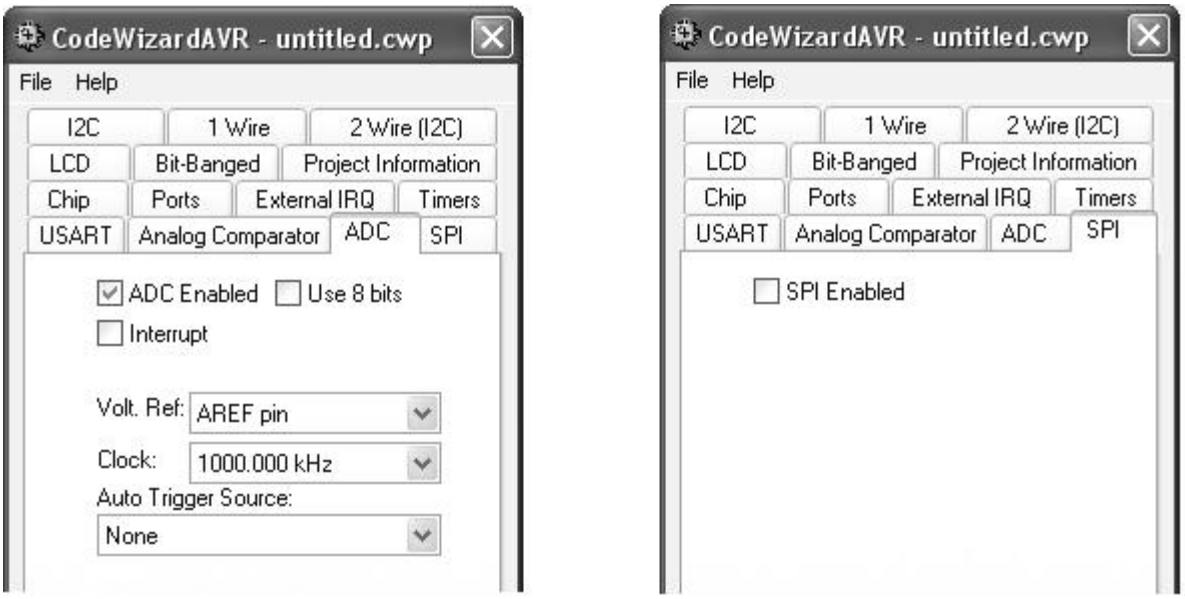
۳-۳-۱- پنجره CodeVision Wizard

برخی دستورات را می توان بدون نوشتن برنامه و از طریق CodeVision Wizard که در منوی Tools می باشد، نیز انجام داد.



شکل ۳-۴: پنجره CodeVision Wizard

همان طور که در شکل ۳-۴ مشاهده می شود، در زبانه Chip از پنجره CodeVision Wizard می توان نوع و فرکانس چیپ را تعیین کرد.



شکل ۳-۵: پنجره CodeVision Wizard

همچنین جهت پورت ها را در زبانه Ports از پنجره CodeVision Wizard می توان تعیین کرد.

با توجه به شکل ۳-۵ فعال سازی مبدل آنالوگ به دیجیتال (ADC) و انجام تنظیمات آن نظیر ولتاژ مرجع، کلاک و ... در زبانه ADC از این پنجره امکان پذیر است.

در زبانه SPI نیز می توان بیت SPI را فعال کرد. در صورت استفاده برخی پروگرمرها برای برخی چیپ ها نیاز است که این بیت فعال شود.

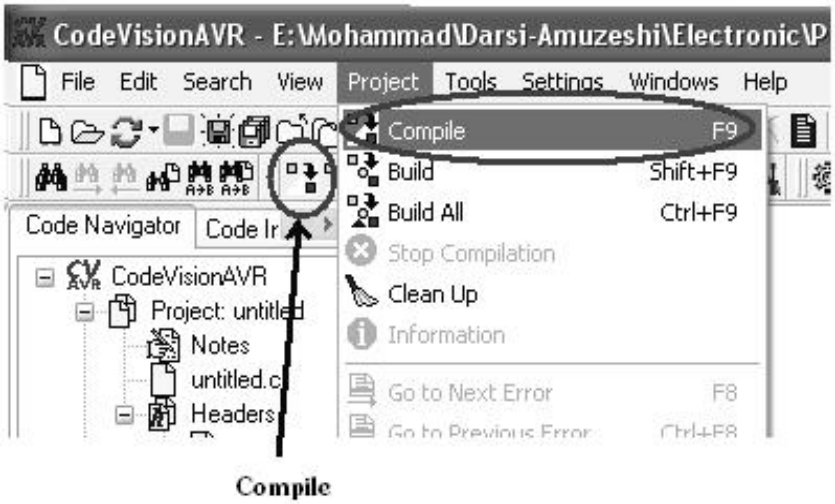
پس از انجام تنظیمات دلخواه از پنجره CodeVision Wizard با انتخاب گزینه Generate, Save and Exit از منوی File از این پنجره، فایل مورد نظر ذخیره شده و در صفحه اصلی نرم افزار

CodeVision کدهای نوشته شده ای به زبان C، متناسب با تظیمات انجام شده ظاهر می گردد که در داخل حلقه بی نهایت آن می توانیم برنامه دلخواه خود را بنویسیم.

۳-۳-۲- کامپایل کردن برنامه

پس از نوشتن کدها و برنامه موردنظر، برای این که فایلی به زبان ماشین جهت برنامه ریزی میکروکنترلر ایجاد شود؛باید ابتدا برنامه را کامپایل کرد تا در صورت عدم بروز خطا بتوان آن را روی میکروکنترلر پروگرام کرد.

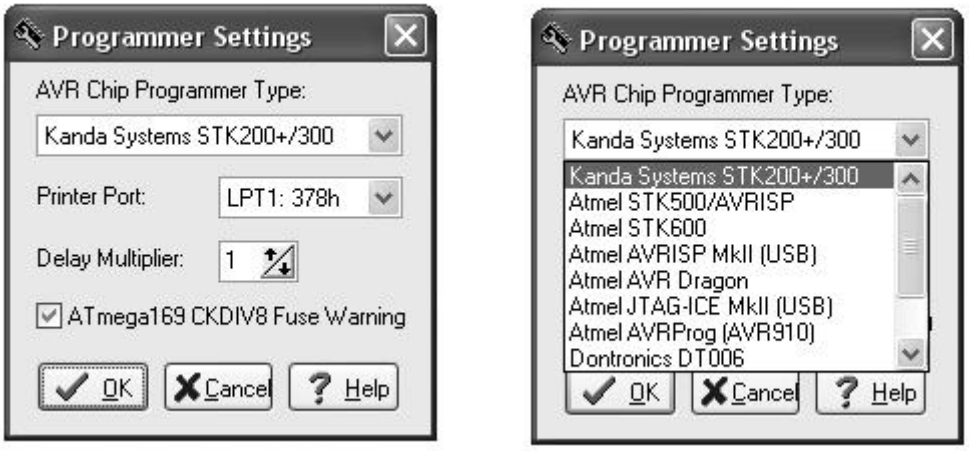
برای کامپایل کردن برنامه نوشته شده در CodeVision از ابزار کامپایل که در شکل ۳-5 مشاهده می شود یا کلید F9 و یا گزینه Compile از منوی Project استفاده می کنیم.



شکل ۳-۶: کامپایل کردن برنامه

۳-۳-۳- پروگرام کردن میکروکنترلر

پس از نوشتن برنامه مورد نظر را در محیط CodeVision و کامپایل کردن آن جهت پروگرام کردن میکروکنترلر، ابتدا به منوی Setting رفته و گزینه Programmer را انتخاب کرده تا پنجره مربوط به انتخاب نوع پروگرامر باز شود.



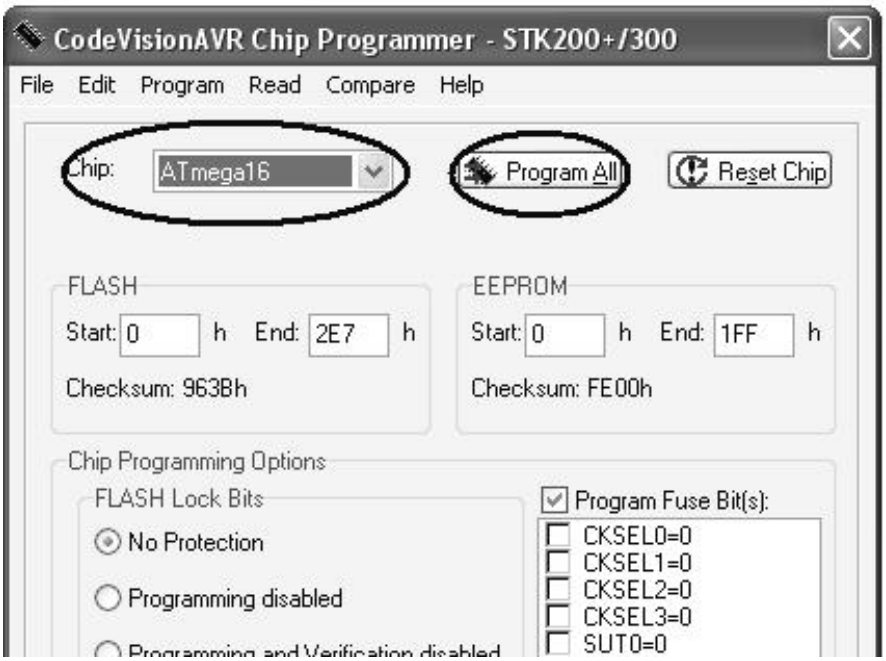
شکل ۳-۷: پنجره Programmer Setting

از این پنجره نوع پروگرامر (STK200+/300) را انتخاب می کنیم.

پس از تعیین نوع پروگرامر گزینه Chip Programmer از منوی Tools انتخاب می کنیم تا پنجره مربوط به پروگرام کردن میکروکنترلر باز شود.

شکل ۳-۸ پنجره Chip Programmer را نشان می دهد. در قسمت Chip باید چیپ مورد استفاده را وارد نموده که در این پروژه از میکروکنترلر ATMEGA16 استفاده شده است.

اکنون که برنامه مورد نظر برای میکروکنترلر نوشته شده، نوع پروگرامرو نوع چیپ مشخص شده است، همچنین ارتباط سخت افزاری از طریق پروگرامر با میکروکنترلر برقرار است، با زدن کلید Program All روی صفحه پنجره Chip Programmer برنامه نوشته شده روی میکروکنترلر برنامه ریزی می گردد.



شکل ۳-۸: پنجره Chip Programmer

فصل چهارم: خلاصه پروژه و پیشنهادات

در این فصل به مرور آنچه که تاکنون بیان شده و ارائه پیشنهاداتی برای افرادی که تمایل به استفاده یا بهبود این پروژه را دارند؛ پرداخته شده است.

۴-۱- خلاصه پروژه

هدف کلی این پروژه این است که مدار و سیستمی طراحی کنیم، که با پخش موسیقی، مؤلفه های مختلف فرکانسی آن تفکیک شده و با انجام یک سری پردازش ها نهایتاً حرکاتی موزون از فواره های آب که توسط تعدادی پمپ آب ایجاد می شود را به وجود آورد.

برای تحقق این امر بلاک دیاگرام شکل ۱-۱ در نظر گرفته شد. به عنوان ورودی سیستم از ولتاژ دوسر بلندگوی در حال پخش استفاده شده است. برای جلوگیری از اثر بارگذاری روی سیستم صوتی پس از نمونه برداری از ولتاژ دوسر بلندگو آن بافر شده است (شکل ۲-۲). پس از آن، از هفت فیلتر میان گذر (شکل ۲-۴) روی فرکانس های مختلف صوتی، برای جدا کردن مؤلفه های مختلف فرکانسی صوت استفاده شده است. اکنون که مؤلفه های فرکانسی صوت جدا شده است؛ باید این سیگنال ها را به ولتاژهای dc بین صفر تا پنج ولت تبدیل کرده تا قابل اعمال به میکروکنترلر جهت پردازش آن باشد. این کار توسط مدار آشکارساز پیک (شکل ۲-۱۰) انجام می گیرد. سپس هرکدام از مؤلفه های صوت که به ولتاژ dc تبدیل شده، را بافر کرده تا اثر بارگذاری روی مدار آشکارساز پیک از بین رود. پس از آن این ولتاژها را به ورودی ADC میکروکنترلر داده تا به مقادیر دیجیتال تبدیل شود. سپس با توجه به این مقادیر در برنامه نوشته شده برای میکروکنترلر، هفت موج PWM متناسب با مقدار هریک از کانال های

ADC (که نشان دهنده مقدار هریک از مؤلفه های صوت است) ایجاد شده است. این هفت موج PWM توسط مدارهای راه انداز (شکل ۲-۱۴) از لحاظ ولتاژ و جریان تقویت شده و به پمپ های آب تولید کننده فواره ها اعمال می گردد تا حرکات موزون این فواره ها متناسب با موسیقی پخش شده از بلندگو ایجاد شود.

۴-۲- پیشنهادات

در این بخش برای افرادی که تمایل به پیاده سازی و بهبود این پروژه را دارند، پیشنهاداتی جهت کامل تر کردن این پروژه و ایجاد چالش های جدید برای تحقیق و انجام کار عملی بیشتر ارائه شده است.

یکی از چیزهایی که می توان به این پروژه اضافه کرد، نورپردازی می باشد که خود می تواند یک پروژه جداگانه باشد. می توان رقص نورهای مختلفی اطراف فواره ها یا نورهای رنگی داخل مخزن آب فواره ها قرار داد تا حرکات آب و فواره ها زیباتر شود.

یکی دیگر از کارهایی که می توان برای تغییر یا به نوعی بهبود این پروژه انجام داد؛ این است که به جای فواره های مستقل از هم که هر یک نشان دهنده یکی از مؤلفه های فرکانسی است، ابتدا تمام خروجی های قسمت جدا کننده مؤلفه های فرکانسی که به ورودی های ADC میکروکنترلر می رود؛ مورد پردازش قرار گرفته و پارامترهایی نظیر سرعت تغییرات مؤلفه ها، مقدار دامنه مؤلفه های زیرتر (فرکانس بالاتر) یا مؤلفه های بم تر(فرکانس پایین تر) و ... مورد محاسبه و تجزیه و تحلیل قرار گرفته و پس از آن بر اساس این پارامترها همه پمپ های آب با هم فرمان گرفته و آرایشی خاص از فواره ها

مثل کم و زیاد شدن ارتفاع همه فواره ها با هم همراه با موسیقی و یا به ترتیب زیاد شدن ارتفاع آب در فواره ها و یا ... را ایجاد نمایند. این امر مستلزم این است که تغییرات اساسی در برنامه نویسی برای میکروکنترلر ایجاد شده و برنامه پیچیده تری برای آن نوشته شود.

منابع و مراجع :

۱- "**مرجع کامل میکروکنترلرهای AVR**" (تألیف محمد مهدی پرتوی فر - فرزاد مظاهریان - یوسف بینانلو_ انتشارات نص_چاپ ۸۹)

۲- "**نظری به موسیقی**" (تألیف روح الله خالقی_انتشارات صفی علیشاه_چاپ ۸۶)

۳- وبسایت **www. fairchildsemicoductor.com**

۴- وبسایت **www. datasheetcatalog.com**

پیوست ها :

بلوک دیاگرام :