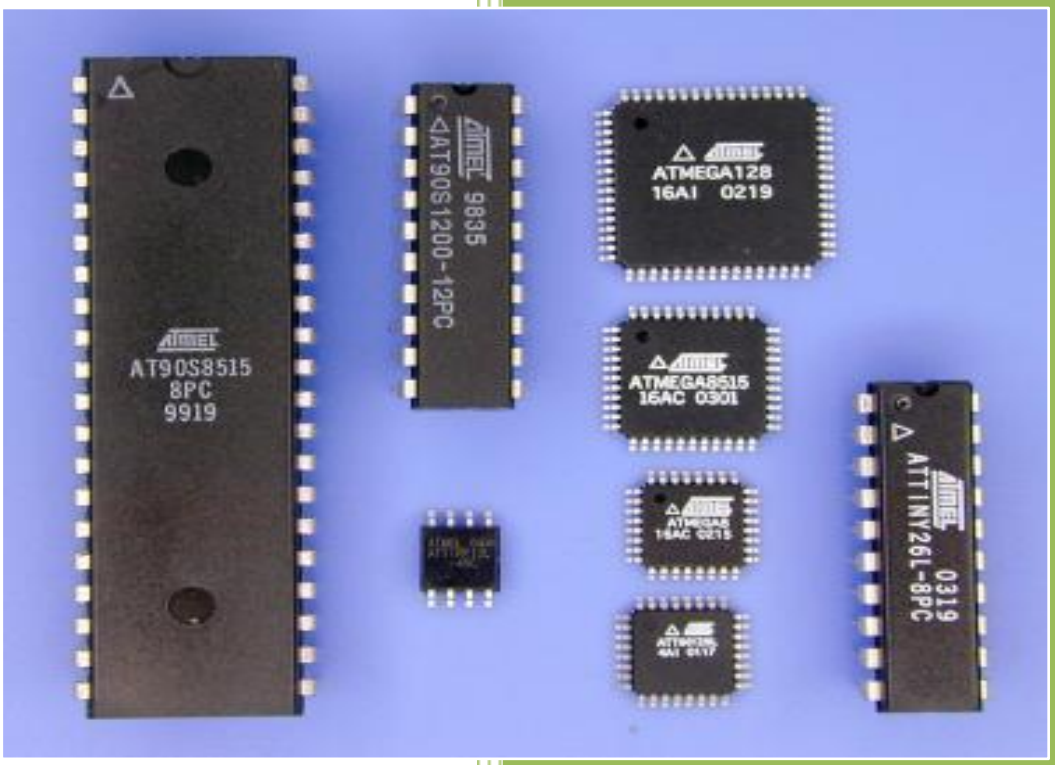
لطفا برای صرفه‌جویی در مصرف کاغذ، این دستور کار را به صورت دو رو چاپ کنید.

دانشگاه صنعتی خواجه نصیرالدین طوسی
گروه کنترل و سیستم
آزمایشگاه ابزار دقیق

مقدمه‌ای بر میکروکنترلرهای AVR



میکروکنترلر در اصطلاح به ریز پردازنده‌هایی گفته می‌شود که بجز CPU حداقل شامل سیستم‌های ورودی و خروجی حافظه و مدارات ارتباط با حافظه در داخل تراشه اصلی هستند و نیازی به مدارات واسطه بیرونی جهت ارتباط با سیستم‌های جانبی را ندارد. البته امکانات همه میکروکنترلرها مشابه و یکسان نمی‌باشد و برخی از میکروکنترلرها علاوه بر امکانات فوق شامل مبدل‌های دیجیتال به آنالوگ و آنالوگ به دیجیتال و یا حتی امکانات بیشتر و اختصاصی‌تر می‌باشند. میکروکنترلری که در آزمایشگاه ابزار دقیق مورد استفاده قرار می‌گیرد از نوع AVR و با نام ATmega32 ساخت شرکت Atmel می‌باشد. برای آشنایی نسبی با عملکرد این میکروکنترلر مطالعه فایل آماده شده توصیه می‌شود.

آخرین به روز رسانی: ۱۵ بهمن ۱۳۹۱

K. N. Toosi University of Technology
Instrumentation Lab

http://saba.kntu.ac.ir/eecd/instlab

فهرست مطالب

مقدمه

امروزه به علت فراگیر شدن سیستم‌های دیجیتالی و کامپیوتری، استفاده از ریز پردازنده‌ها و پردازنده‌های سیگنال‌های دیجیتال گسترش یافته است. میکروکنترلرها یکی از انواع ریزپردازنده‌ها می‌باشند که برای کاربردهای کنترلی و صنعتی طراحی شده‌اند و بیشتر برای پردازش اطلاعات یک سیستم صنعتی و کنترل یک فرآیند استفاده می‌شوند. از میکروکنترلرهای پر استفاده در صنعت می‌توان به سری میکروکنترلرهای AVR، PIC و ARM اشاره کرد.

میکروکنترلرهای AVR یکی از انواع میکروکنترلرها هستند که به علت امکانات مناسب و سادگی کار با آنها و همچنین قیمت مناسب، کاربرد فراوانی پیدا کرده‌اند. میکروکنترلرهای خانواده AVR ساخت شرکت Atmel می‌باشند که به طور وسیعی در ایران مورد استفاده قرار می‌گیرند. در آزمایشگاه ابزار دقیق از میکروکنترلرهای AVR برای انجام عملیات محاسباتی مورد نظر و یا نمایش اطلاعات دریافتی از سنسورها و مدارات مرتبط با آنها استفاده می‌شود. در این جزوه به معرفی مختصری از میکروکنترلرهای مورد استفاده در آزمایشگاه و روش کار با آنها پرداخته می‌شود.

معرفی

در آزمایشگاه ابزار دقیق از میکروکنترلرهای AVR سری Atmega16 و Atmega32 استفاده می‌شود. این دو میکروکنترلر از نظر ساختاری و ترتیب پایه‌ها مشابه هم بوده و تفاوت در میزان حافظه Flash و EEPROM آنها می‌باشد. برای موارد استفاده در آزمایشهای این آزمایشگاه نیاز به حافظه چندانی نمی‌باشد و از میکروکنترلرهای Atmega16 استفاده می‌شود. برای اطلاع از ترتیب پایه‌های میکروکنترلر باید به برگه راهنما (datasheet) آن مراجعه نمود. ولتاژ تغذیه میکروکنترلر باید در محدوده ۲.۷ تا ۵.۵ ولت باشد. یک میکروکنترلر برای عملکرد درست و اجرای هر دستور سطح پایین، نیاز به اعمال پالس ساعت (clock) دارد که در میکروکنترلرهای AVR می‌توان از کلاک داخلی، پالس کلاک خارجی و یا کریستال خارجی استفاده نمود.

از مهم‌ترین امکانات میکروکنترلر Atmega16 می‌توان به موارد زیر اشاره نمود:

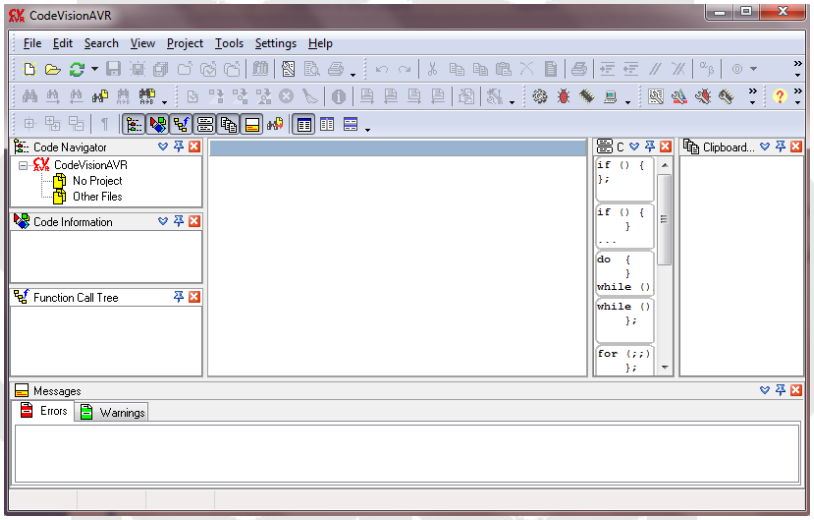
- ۳۲ رجیستر ۸ بیتی همه منظوره؛
- سه نوع حافظه Flash، EEPROM و SRAM؛
- دو تایمر/ شمارنده ۸ بیتی و یک تایمر/ شمارنده ۱۶ بیتی با مُدهای کاری مختلف؛
- ۸ کانال مبدل آنالوگ به دیجیتال (ADC) ۱۰بیتی؛
- قابلیت ارتباط سریال با رایانه یا وسایل جانبی دیگر (USART)؛
- منابع وقفه داخلی و خارجی.

برای اطلاع از جزئیات هر بخش یک میکروکنترلر و روش کار با آن باید به دیتاشیت آن نوع میکروکنترلر مراجعه نمود. یک بخش دیگر در میکروکنترلرهای AVR فیوز بیت (fuse bit) های آنها می‌باشد. فیوز بیتها قسمتی از حافظه AVR هستند که امکاناتی را در اختیار کاربر قرار داده و با پاک کردن برنامه میکروکنترلر، مقدار این بیت‌ها تغییر نمی‌کند. صفر به معنی فعال بودن و یک به معنی غیرفعال بودن یک فیور بیت می‌باشد. از عملکردهایی که توسط فیوز بیت‌ها قابل تنظیم هستند می‌توان به نحوه برنامه‌ریزی میکروکنترلر، انتخاب منبع کلاک و نحوه boot شدن قطعه اشاره کرد. این بیت‌ها را در هنگام برنامه ریزی و توسط نرم‌افزار می‌توان تنظیم نمود.

برنامه‌ریزی میکروکنترلرهای AVR

برنامه ریزی میکروکنترلرها به روشهای مختلفی مانند ISP و JTAG انجام می‌گیرد که هر کدام نیاز به نرم‌افزار واسط و سخت‌افزار (programmer) مخصوص به خود دارند. در این آزمایشگاه از روش ISP برای برنامه‌ریزی میکروکنترلرها استفاده می‌شود.

همچنین برنامه نویسی و پیاده‌سازی الگوریتم کنترلی یا محاسباتی مورد نظر نیز به روشهای مختلفی قابل انجام است. معمولاً برنامه میکروکنترلرهای AVR به زبان اَسِمبلی یا C نوشته می‌شود. در آزمایشگاه ابزار دقیق از نرم‌افزار Codevision برای برنامه‌نویسی، کامپایل کردن برنامه و پروگرم کردن میکروکنترلرها استفاده می‌شود. محیط این نرم افزار مشابه شکل ۱ می‌باشد. در ادامه روش کار با این نرم‌افزار بیان می‌شود.

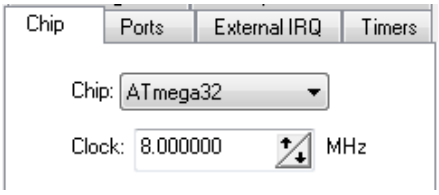


شکل ۱: محیط نرم افزار CodeVision

برای ایجاد یک پروژه جدید از منوی File گزینه New را انتخاب کرده، در پنجره باز شده نوع فایل را Project انتخاب کرده و در پاسخ به پرسش مطرح شده برای استفاده از CodeWizard گزینه Yes را انتخاب نمایید. در ادامه در قبال نوع AVR انتخابی گزینه اول را که شامل Atmega (و نه Atxmega) است را برگزینید تا صفحه مربوط به تنظیمات اولیه آن ظاهر شود. در این صفحه که شامل سربرگ‌های متعددی است، می‌بایست کاربردهای مورد نیاز را مشخص نمایید تا نرم افزار به طور خودکار Register های مورد نظر را تنظیم و پیش‌برنامه را برای ادامه کار برنامه‌نویسی تدوین نماید. در این بخش تنها سربرگ‌هایی توضیح داده می‌شوند که برای انجام آزمایشات نیاز به تنظیم آنها می‌باشد.

سربرگ Chip

در سربرگ Chip باید نوع تراشه مورد استفاده و فرکانس کاری مورد نظر را تنظیم کرد.



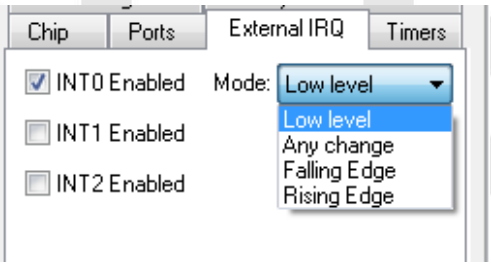
شکل ۲: سربرگ Chip

سربرگ Ports

در سربرگ Ports می‌توان نوع عملکرد پایه‌های میکروکنترلر (ورودی یا خروجی بودن) را تنظیم کرد و در صورتی که یک پایه به عنوان خروجی انتخاب شود، می‌توان به آن مقدار اولیه داد. همچنین برای پایه‌های ورودی نیز می‌توان وضعیت Pull-up یا Tri-state را انتخاب نمود. البته باید توجه نمود که پایه‌ها برای منظور دیگری (مثلاً اتصال به LCD) استفاده نشده باشند.

سربرگ External IRQ

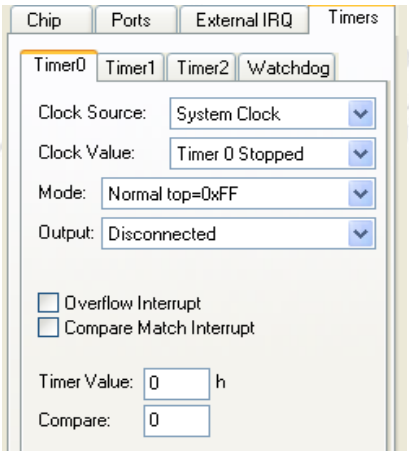
در این قسمت می‌توان پایه وقفه خارجی و نوع آن (فعال با سطح پایین، هردو لبه بالا و پایین رونده، لبه پایین رونده و یا لبه بالا رونده) را مشخص کرد. گزینه INT0 مربوط به وقفه خارجی صفر است که مستقیماً پایه INT0 را می‌خواند و به همین ترتیب برای دیگر وقفه‌ها.



شکل ۳: سربرگ وقفه خارجی

سربرگ Timers

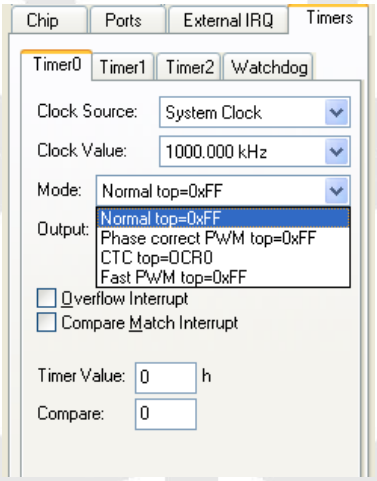
در سربرگ Timers می‌توان تایمر و یا شمارنده مورد نظر را فعال و تنظیمات آن را انجام داد. میکروکنترلر Atmega16 دارای ۳ تایمر/ شمارنده می‌باشد که Timer0 و Timer2 هشت بیتی بوده و Timer1 شانزده بیتی می‌باشد. هر تایمر/ شمارنده i دارای ۳ رجیستر اصلی TCCRi، TCNTi و OCRi می‌باشد که به ترتیب جهت پیکربندی تایمر، مقدار شمارنده و مقدار مقایسه استفاده می‌شوند.



شکل ۴: سربرگ تایمر

در میکروکنترلرها یک تایمر در واقع یک شمارنده می‌باشد که با فرکانس مشخصی عمل شمارش را انجام می‌دهد. در قسمت Clock Source می‌توان منبع تولید پالس برای شمارنده‌ها را انتخاب نمود. در صورتی که گزینه System Clock انتخاب شود، شمارنده با کلاک سیستم یا کسری از آن ($1/8$، $1/64$، $1/256$ و $1/1024$) عمل شمارش را انجام داده و یک واحد به متغیر TCNTi اضافه می‌کند و در واقع به عنوان یک تایمر عمل می‌کند. در صورتی که هر رجیستر TCNT به حداکثر مقدار خود (۲۵۵ برای شمارنده ۸ بیتی یا ۶۵۵۳۵ برای شمارنده ۱۶ بیتی) برسد، Overflow کرده و مقدار آن صفر می‌شود. در قسمت Mode، نوع عملکرد تایمر/ شمارنده را می‌توان تنظیم نمود که در ادامه به طور خلاصه معرفی می‌شوند:

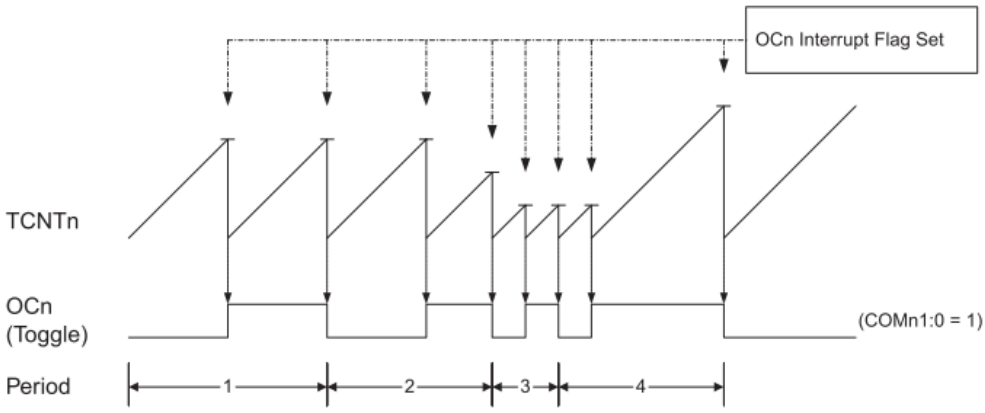
- حالت Normal top: با انتخاب این حالت، شمارنده به صورت بالاشمار رفتار می‌کند و تا مقدار سرریز به شمارش ادامه می‌دهد. در صورتی که گزینه Overflow Interrupt فعال شده باشد، پس از اینکه سرریز شمارنده اتفاق افتاد، وقفه فعال شده و برنامه وارد روتین وقفه مربوط به شمارنده می‌شود.



شکل ۵: مُدهای کاری مربوط به Timer0

- مد CTC (Clear Timer on Compare match)

در این مد زمانی که محتوای تایمر (TCNTn) با رجیستر OCRn برابرشود صفر می گردد. در حقیقت دراین مد رجیستر OCRn مقدار حداکثر تایمر و رزولوشن آن را تعیین می کند. نمودار زمانی این مود به صورت شکل زیر است:



در این حالت پس از هر بار رسیدن محتوای تایمر به حداکثر مقدار خود که توسط رجیستر OCRn مشخص می شود، پرچم OCFn (Output Compare) یک می‌گردد و در صورتی که از قبل تطابق مقایسه (Compare Match) فعال شده باشد روتین وقفه اجرا می گردد. اصولا از این وقفه برای تغییر محتوای OCRn استفاده می شود. در این مد می توان پایه OCn (Output Compare) را طوری تنظیم کرد تا در زمان تطابق مقایسه (Compare Match) تغییر وضعیت دهد. در این صورت یک موج مربعی با Duty Cycle ۵۰ درصد بر روی پایه OCn تولید خواهد شد که فرکانس آن با تغییر محتوای رجیستر OCRn قابل تغییر است. در این مد فرکانس موج مربعی از رابطه ی زیر بدست می آید.

$$f_{OCn} = \frac{f_{clk}}{2.N.(1+OCRn)}$$

که در آن متغیر N میزان تقسیم فرکانسی را نشان می دهد. این متغیر در تایمرهای مختلف می تواند مقادیر زیر را به خود بگیرد:

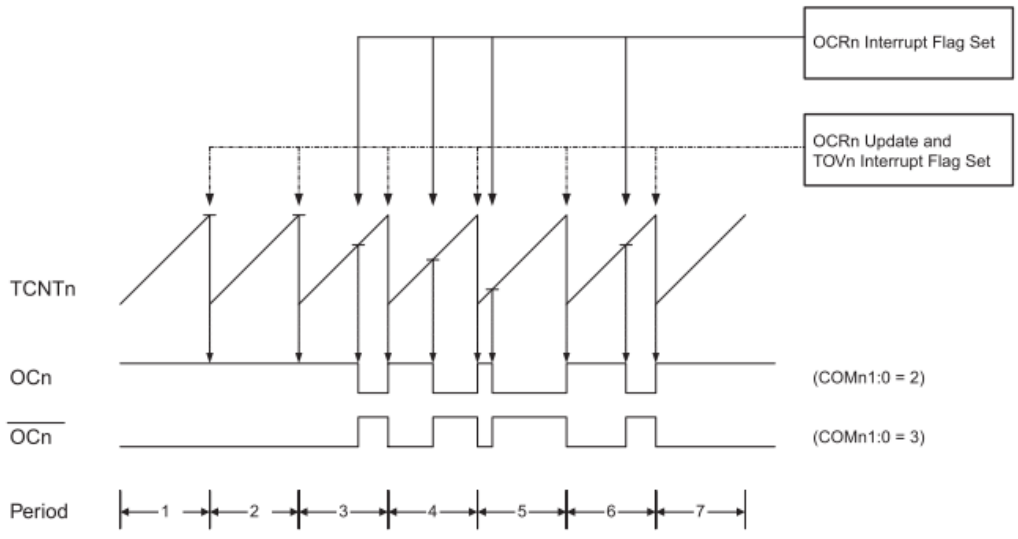
$$for\ Timer0\ and\ Timer1\ N = 1,8,64,256,1024$$

$$for\ Timer2\ N = 1,8,64,128,256,1024$$

از کاربرد های این مد می توان به مدولاسیون فرکانسی (FM) اشاره کرد.

- مد Fast PWM (Fast Pulse Width Modulation)

این مد امکان ایجاد موج PWM با فرکانس بالا را فراهم می کند. موج PWM خروجی در پایه ی OCn (در تایمر ۱ دو پایه ی OC1A و OC1B) تولید می شود. برای این منظور لازم است تا این پایه قبلا به صورت خروجی تنظیم شده باشد. علاوه بر این همانطور که در شکل زیر مشاهده می کنید. شکل موج PWM خروجی می تواند به صورت Inverted و یا Non-Inverted تنظیم گردد. در این مد محتوای تایمر (TCNTn) از مقدار صفر شروع به افزایش کرده و پس از رسیدن به مقدار حداکثر خود دوباره صفر می شود. در این حین محتوای تایمر به صورت مداوم با محتوای رجیستر OCRn مقایسه می شود و در صورتی که این دو مقدار برابر شوند، پایه ی خروجی OCn مطابق شکل تغییر علامت می دهد.



در این مد فرکانس PWM تولید شده در پایه خروجی از رابطه زیر بدست می آید:

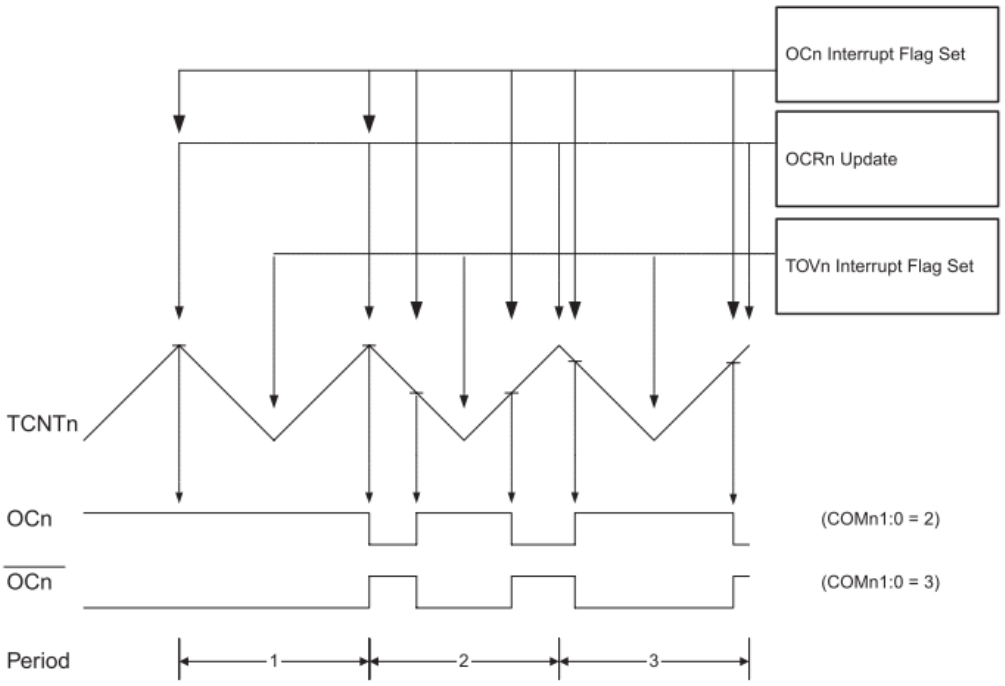
$$f_{OCnxPWM}=\frac{f_{clk}}{N.(1+TOP)}$$

که در این رابطه TOP حداکثر مقدار رجیستر TCNTn می باشد.

از کاربرد های موج PWM تولید شده در این مد می توان به سوئیچینگ،یکسوسازی،تبدیل دیجیتال به آنالوگ و راه اندازی موتورها و عملگرهای DC نام برد.

- مد Phase Correct PWM

این مود بر خلاف روش قبل که موج PWM را به صورت تک شیبه (Single Slope) تولید می کرد، موج PWM را به صورت دو شیبه (Dual Slope) تولید می کند. در این مد محتوای تایمر (TCNTn) از مقدار صفر شروع به افزایش می کند و پس از رسیدن به مقدار حداکثر خود شروع به کاهش می کند.در این حین محتوای تایمر به صورت مداوم با رجیستر OCRn نیز مقایسه می شود و در صورتی که مقدار TCNTn در هر یک از حالات صعودی و نزولی با مقدار OCRn برابر شود، پایه ی خروجی OCn مطابق شکل زیر تغییر حالت می دهد.در اینجا هم شبیه حالت قبل پایه OCn می تواند به صورت Inverted و یا Non-Inverted تنظیم گردد. علاوه بر این لازم است تا این پایه قبل از این حتما به عنوان خروجی مشخص شده باشد. ئر این مد به دلیل وجود ساختار دوشیبه حداکثر فرکانس موج ایجاد شده نصف فرکانس ایجاد شده در مد Fast PWM خواهد بود.

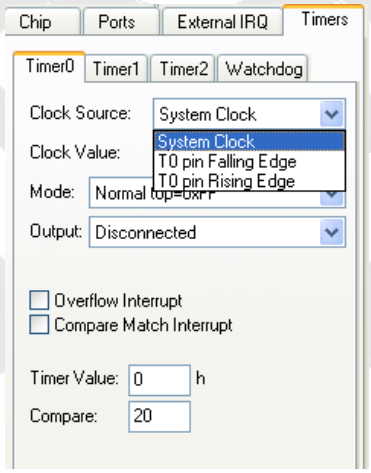


برای محاسبه ی فرکانس PWM می توان از رابطه ی زیر استفاده کرد:

$$f_{OCnxPWM}=\frac{f_{clk}}{2.N.(TOP)}$$

توجه شود که تا وقتی تایمر/ شمارنده غیرفعال نشود، با فرکانس تعیین شده به عمل شمارش ادامه می‌دهد و رجیستر TCNT مرتباً افزایش یافته و سرریز می‌کند. بنابراین در صورتی که مایل به ایجاد یک پالس با دوره زمانی مشخص باشیم، باید پس از رسیدن مقدار TCNT به مقدار مورد نظر (OCR)، مقدار آن را صفر کرد تا دور بعدی شمارش از صفر شروع شود.

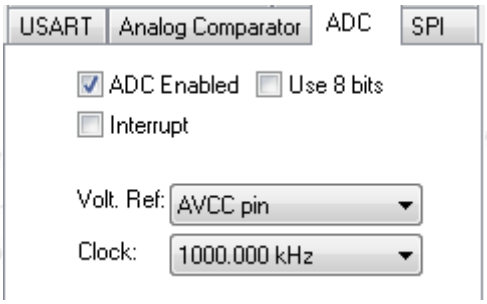
پالس ورودی برای شمارنده را می‌توان از یک منبع خارجی نیز دریافت کرد. این حالت معمولاً برای شمارنده و زمانی که مایل به شمارش یک پالس خارجی باشیم، استفاده می‌شود. در این حالت پالس خارجی را باید به پایه T0، T1 یا T2 قطعه (متناسب با تایمر انتخابی) متصل کرد. همچنین می‌توان شمارنده را نسبت به لبه بالا رونده یا پایین رونده پالس ورودی حساس کرد.



شکل ۶: انتخاب منبع پالس برای شمارنده

سربرگ ADC

این قسمت مربوط به مبدل آنالوگ به دیجیتال میکروکنترلر می باشد. با انتخاب گزینه ADC Enabled مبدل آنالوگ به دیجیتال داخلی میکروکنترلر فعال می‌گردد. در این حالت باید توجه نمود که پایه‌های مربوط به ADC که در Atmega16 پایه‌های پورت A هستند، قبلاً به عنوان ورودی تعریف شده باشند.



شکل ۷: سربرگ مبدل آنالوگ به دیجیتال

در حالت عادی نوع مبدل ADC، ۱۰ بیتی می‌باشد که با انتخاب گزینه Use 8 bits می‌توان از دقت ۸ بیت استفاده کرد. در واقع در حالت ۱۰ بیتی، تفکیک‌پذیری[1] مبدل ADC به ازای ولتاژ مرجع ۵ ولت، برابر با $\frac{5}{1024} \cong 5\ mV$ می‌باشد. با فعال کردن ADC تابعی با نام read_adc() در برنامه ایجاد می‌شود:

```
unsigned int read_adc(unsigned char adc_input)
```

این تابع می‌تواند مقدار ولتاژ اعمال شده به پایه‌های ADC0 تا ADC7 را بخواند و آن به عددی بین ۰ تا ۱۰۲۳ نگاشت کند (اگر ولتاژ ورودی برابر با ولتاژ مرجع باشد، مقدار ۱۰۲۳ و اگر صفر ولت باشد مقدار صفر را برمی‌گرداند). به عنوان مثال دستور زیر ولتاژ پورت ADC3 را در متغیر a می‌ریزد:

```
unsigned int a;
a=read_adc(3);
```

برای مبدل ADC نیاز به یک ولتاژ مرجع می‌باشد که مقدار ولتاژ ورودی نسبت به آن سنجیده می‌شود. ولتاژ مرجع را می‌توان به ۳ روش تعیین کرد:

- متصل کردن ولتاژ مرجع خارجی به پایه AREF میکروکنترلر؛
- استفاده از ولتاژ AVCC که همان ۵ ولت بوده و به پایه AVCC میکروکنترلر متصل است؛
- استفاده از ولتاژ ۲.۵۶ ولت که به صورت داخلی ایجاد می‌شود و نیازی به اتصال ولتاژ خارجی به پایه AREF ندارد.

همچنین فرکانس نمونه‌برداری مبدل ADC را نیز می‌توان در قسمت Clock انتخاب کرد. در این مورد باید به ماهیت سیگنال ورودی و تغییرات آن توجه نمود. برای سیگنالهای DC خروجی سنسورها که تغییرات آنها کمتر از ۱۰۰ هرتز می‌باشد، می‌توان فرکانس نمونه‌برداری را حداقل مقدار ممکن (با توجه به معیار نایکوئیست) انتخاب کرد.

سربرگ Alphanumeric LCD

در این قسمت می‌توان تنظیمات لازم برای اتصال یک نمایشگر کاراکتری به میکروکنترلر را فراهم آورد. گزینه‌های موجود در این قسمت به طور خلاصه در شکل زیر معرفی شده‌اند.

شکل ۸: سربرگ LCD و نحوه اتصال LCD

[1] Resolution

در صورتی فعال کردن این قسمت و انتخاب LCD و تعیین پورت مربوطه، توابعی برای استفاده از LCD فعال خواهند شد. برخی از مهم‌ترین توابع این قسمت بدین شرح هستند:

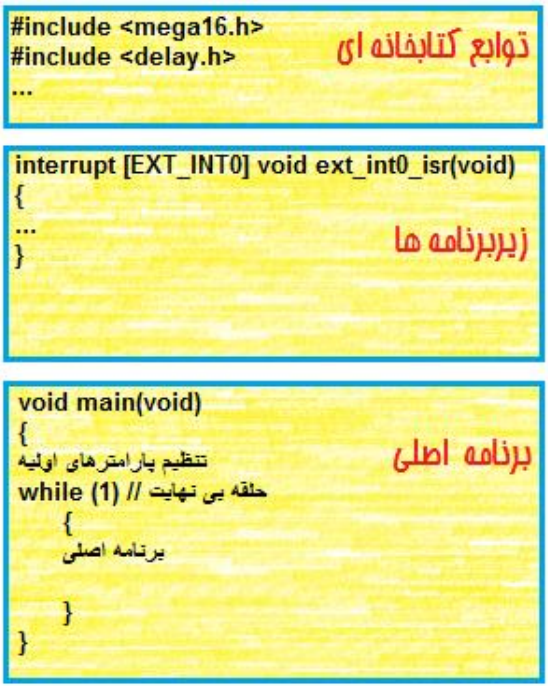
```
lcd_clear();          //    پاک کردن صفحه
lcd_gotoxy(0,0);      //    رفتن به نقطه ۰ و ۰ (عدد اول موقعیت افقی و عدد دوم شماره سطر)
lcd_putsf("Hello");   //    نمایش یک رشته ثابت
lcd_puts(str);        //    نمایش یک رشته متغیری
lcd_putchar('a');     //    نمایش یک کاراکتر
```

چون تنها رشته‌های کاراکتری را می‌توان روی LCD نمایش داد، برای نمایش اعداد ابتدا باید آنها را به کاراکتر عددی معادل تبدیل کرده (مثلاً کاراکتر 5 برای عدد 5) و سپس برای LCD ارسال کرد. برای این کار از دستور ftoa و itoa که زیر مجموعه کتابخانه stdlib.h می‌باشند، استفاده می‌شود. دستور ftoa برای تبدیل یک عدد اعشاری (float) به معادل کاراکتری آن و دستور itoa برای تبدیل یک عدد صحیح (integer) به معادل کاراکتری می‌باشد. نحوه استفاده از این دستورات به صورت زیر می‌باشد.

```
float  a=90.12;
int b=1390;
char str1[5]   ,  str2[5] ;

ftoa( a , 1 , str1 );          //    عدد ۱ نشان دهنده تعداد رقم بعد از اعشار می‌باشد
itoa( b , str2 )
```

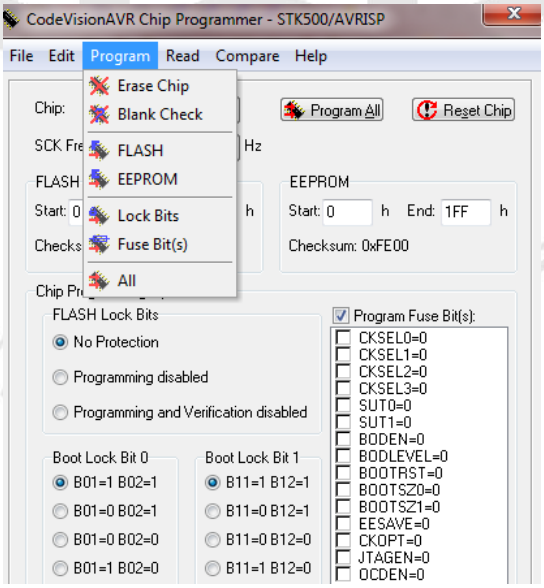
پس از انجام تنظیمات اولیه در Code Wizard، با کلیک بر گزینه Generate, Save and Exit فایل نهایی آماده شده و محیط برنامه‌نویسی به زبان C که شامل کدهای مربوط به تنظیمات اولیه است، ظاهر می‌شود و می‌توان شروع به برنامه‌نویسی کرد. ساختار کلی یک برنامه در نرم افزار CodeVision به صورت زیر است:



شکل ۹: ساختار برنامه نویسی در CodeVision

برنامه ریزی میکروکنترلر

پس از تکمیل برنامه در نهایت با فشردن کلید F9 برنامه Compile شده و در صورت عدم وجود خطا، با استفاده از Ctrl+F9 می‌توان فایل‌های obj. و hex. متناظر با برنامه را تولید کرد. در صورتی که پروگرمر مورد استفاده توسط برنامه Codevision شناخته شده باشد، می‌توان با انتخاب Chip Programmer از منوی Tools به بارگزاری برنامه ایجاد شده در میکروکنترلر اقدام کرد. همچنین در صورتی که پروگرمر مورد استفاده در برنامه شناسایی نشده باشد، باید فایل hex. ایجاد شده را در نرم‌افزار مربوط به پروگرمر باز کرده و توسط آن نرم‌افزار برنامه (فایل hex) را در میکروکنترلر بارگزاری کرد.



شکل ۱۰: Chip Programmer

یادآوری دستورات زبان برنامه نویسی C

در این بخش به معرفی بعضی از دستورات مورد نیاز در زبان c یادآوری می‌شوند.

انواع داده

در بسیاری از موارد نیاز است تا متغیرهایی را تعریف و از آن ها در برنامه استفاده شود. قالبهای اصلی برای تعریف متغیرها در Codevision به صورت زیر هستند:

Type	Size (Bits)	Range
bit	1	0 , 1
bool, _Bool	8	0 , 1
char	8	-128 to 127
unsigned char	8	0 to 255
signed char	8	-128 to 127
int	16	-32768 to 32767
short int	16	-32768 to 32767
unsigned int	16	0 to 65535
signed int	16	-32768 to 32767
long int	32	-2147483648 to 2147483647
unsigned long int	32	0 to 4294967295
signed long int	32	-2147483648 to 2147483647
float	32	±1.175e-38 to ±3.402e38
double	32	±1.175e-38 to ±3.402e38

عملگرهای مقایسه‌ای

عملگر	عملکرد	مثال	نتیجه
>	بزرگتر	2>3	False
<	کوچکتر	'm'>'e'	True
>=	بزرگتر یا مساوی	5>=5	True
<=	کوچکتر یا مساوی	2.5<=4	True
==	تساوی	'A' =='B'	False
!=	نامساوی	2!=3	True

نمونه ای از دستورات شرطی

```
if (شرط)
        {دستور۱}
```

```
else
        {دستور۲};
```

اگر شرط برقرار بود «دستور ۱» وگرنه «دستور ۲» اجرا می‌شود.

```
while (شرط)
    {دستور};
```

تا زمانی که شرط برقرار است «دستور» اجرا شود.

```
for (i==0شروع ; i==100پایان ; i++گام حرکت)
    {دستور};
```

به ازای i=0 تا زمانی که i به ۱۰۰ برسد «دستور» اجرا و با هر بار اجرا یک واحد به i افزوده می‌شود.

مهمترین توابع کتابخانه‌ای که با آنها سروکار دارید، موارد زیر هستند:

```
#include <mega16.h>  // برای مشخص کردن نوع میکرو
#include <lcd.h>     // برای استفاده از دستورات نمایشگر
#include <delay.h>   // استفاده از تاخیر
#include <stdio.h>   // بعضی دستورات ورودی خروجی
#include <stdlib.h>  // دستورات تغییر نوع داده و ...
```

تمرین‌های کاربردی

۱- نوشتن یک متن بر روی LCD کاراکتری:

ابتدا تنظیمات مربوط به میکروکنترلر (atmega16) و LCD کاراکتری ۲×۱۶ را در برنامه Codevision انجام دهید. با تعریف متغیرهای لازم و اضافه کردن توابع کتابخانه‌ای مورد نیاز، یک متن دلخواه (نام خانوادگی اعضای گروه) را در سطر اول نمایشگر و حاصلضرب دو عدد اعشاری دلخواه را در سطر دوم، نمایش دهید.

۲- استفاده از ADC:

ابتدا با استفاده از منبع ولتاژ ثابت و یک پتانسیومتر، یک ولتاژ dc بین ۰ تا ۵ ولت ایجاد کرده و آن را به پایه ADC0 میکروکنترلر متصل نمایید. سپس با انجام تنظیمات مربوط به ADC در برنامه Codevision، برنامه‌ای بنویسید که مقدار ولتاژ متصل به پایه‌ی ADC0 را روی سطر اول به صورت زیر نمایش دهد.

V = x.xxx volts

۳- استفاده از وقفه خارجی:

در این قسمت نیز با کمک مربیان آزمایشگاه، دو کلید فشاری (push botton) به پایه‌های INT0 و INT1 طوری متصل کنید که در حالت عادی ولتاژ ۵ ولت به پایه‌های وقفه خارجی متصل بوده و با فشردن هر کلید ولتاژ پایه وقفه مربوط به آن صفر شود. سپس برنامه‌ای بنویسید که با فشردن کلید متصل به INT0، مقدار یک متغیر یک واحد اضافه شده و با فشردن کلید دیگر که به INT1 متصل است، یک واحد از مقدار آن متغیر کم شود. مقدار آن متغیر را همواره بر روی LCD نمایش دهید.

۴- استفاده از تایمر / شمارنده:

ابتدا یک عدد LED را به یکی از پایه‌های خروجی میکروکنترلر متصل کنید (مقاومت ۳۳۰ اهم بین LED و زمین را فراموش نکنید). با انجام تنظیمات مناسب در Codewizard، برنامه‌ای بنویسید که یک چراغ چشمک‌زن داشته باشیم.

۵- استفاده از نرم‌افزار Proteus:

در این قسمت می‌خواهیم از نرم افزار Proteus که یک شبیه‌ساز مناسب و کاربردی برای مدارات میکروکنترلری می‌باشد، استفاده نماییم. پس از ایجاد یک فایل جدید در این نرم‌افزار، یک عدد میکروکنترلر Atmega16، یک عدد LCD کاراکتری ۲×۱۶، دو عدد کلید فشاری، یک مقاومت متغیر برای ایجاد ولتاژ آنالوگ و یک عدد LED انتخاب کرده و آنها را به طور مناسب به هم متصل نمایید. سپس با باز کردن منوی مشخصات میکروکنترلر، فایل‌های coff یا hex مربوط به برنامه‌های نوشته شده در قسمتهای قبل را به عنوان برنامه میکروکنترلر انتخاب کنید. در این حالت باید نتایجی مشابه نتایج عملی مشاهده شود. توجه کنید که در مورد برنامه مربوط به تایمر یا ADC اتصالات در شبیه‌ساز، مشابه حالت عملی انجام شوند.