کتاب PDF

میکرو کنترلرهای AVR با پروژه های ۱۰۰٪ عملی

تألیف جابر الوندی

منبع درس ریزپردازنده۱ دانشگاه پیام نور

فایل PDF موجود نسخه RF، در بین سایر فایل های PDF ریزپردازنده موجود در سایت های دانلود کتاب، کامل ترین فایل PDF محسوب میشود. چرا که در سایر کتاب های PDF مشابه، علاوه بر ناقص بودن صفحات کتاب، نظم و ترتیب خاصی بین صفحات وجود ندارد که این نقص، باعث ایجاد مشکل و سردرگمی برای دانشجویان در هنگام مطالعه کتاب PDF می شود.

از همین رو ویرایش RF کتاب پی دی اف ریزپردازنده۱ را که نواقص فایل های قبلی را برطرف کرده، برای استفاده دانشجویان عزیزی که به کتاب اصلی این درس دسترسی ندارند، آماده کرده ایم.

Edited by Aref.b *4123ph@gmail.com*

فصل ۱

ساختار داخلی میکروکنترلرهای AVR

اهداف

۱. آشنایی با میکروکنترلر ATmega16
۲. معرفی فیوز بیت‌های میکروکنترلر
۳. آشنایی با پورت‌های ورودی و خروجی
۴. ارائه منبع تغذیه میکروکنترلر با ورودی AC و DC
۵. آشنایی با ساختار داخلی میکروکنترلر ATmega16
۶. معرفی کلاک سیستم و انواع منابع پالس ساعت در میکروکنترلر
۷. آشنایی با مُدهای Sleep و معرفی تایمر نگهبان (Watchdog)

۱-۱ تعاریف اولیه از ساختار میکروکنترلر

در این قسمت می‌خواهیم شما را با چند تعریف ساده از اجزای یک میکروکنترلر آشنا کنیم تا ذهن شما آماده پذیرش مطالب در ادامه آموزش کتاب باشد.

انواع حافظه ماندگار (دائمی) :

حافظه‌های ماندگار به حافظه‌هایی گفته می‌شود که بتوانند اطلاعات داده شده را نگه دارند حتی اگر تغذیه آنها قطع شود نباید این اطلاعات پاک شود.

حافظه PROM : این نوع حافظه فقط خواندنی، تنها می‌تواند یکبار برنامه‌ریزی شود.

حافظه EPROM : حافظه فقط خواندنی است که اطلاعات توسط مدار واسط، با ولتاژ الکتریکی نوشته و با نور ماورای بنفش مثل نور خورشید توسط پنجره شیشه‌ای که معمولاً با یک بر چسب پوشیده می‌شود پاک می‌گردد.

حافظه EEPROM : حافظه فقط خواندنی است که اطلاعات توسط مدار واسط، با ولتاژ الکتریکی

نوشته و پاک می‌شود این نوع حافظه دو مدل است:

الف- موازی (Parallel) : آدرس‌دهی، نوشتن و خواندن حافظه به صورت موازی انجام می‌گیرد.

ب- سریال (Serial) : این حافظه اغلب در بسته‌بندی‌های کوچک ۸ پایه قرار می‌گیرند و آدرس‌دهی حافظه و آدرس‌دهی سخت‌افزاری و همچنین عمل نوشتن و خواندن به صورت سریال و فقط توسط ۲ پایه انجام می‌گیرد به این نوع ارتباط‌دهی I2C گفته می‌شود.

انواع حافظه فرار (غیر دائمی) :

حافظه‌های فرار به حافظه‌هایی گفته می‌شود که بتوانند اطلاعات داده شده را نگه دارند اما بر خلاف حافظه‌های ماندگار، اگر تغذیه آنها قطع شود اطلاعات نیز پاک می‌شود.

SRAM : این نوع حافظه از نوع Static یا پایدار است. منظور از پایداری این است که نیازی به تازه‌سازی ندارد و تا وقتی که تغذیه آن بر قرار است می‌تواند اطلاعات داده شده را حفظ نماید.

DRAM : این نوع حافظه از نوع Dynamic یا غیر پایدار است. یعنی CPU باید مدام با یک فاصله زمانی مشخص، دیتای ذخیره شده در این نوع حافظه را بازسازی کند این قبیل از حافظه‌ها با ظرفیت‌های بالا ساخته می‌شوند و بیشتر در کنار ریز پردازنده‌هایی با سرعت بالا قرار می‌گیرند.

پورت‌های ورودی و خروجی :

منظور از Port یا درگاه این است که یک سری پایه برای ارتباط با دنیای بیرون تراشه فراهم شده است که هم می‌توانند ورودی و هم خروجی باشند. هر میکروکنترلر با توجه به نوع بسته‌بندی دارای یک إلی چندین پورت است و الزاماً یک پورت دارای ۸ پایه نیست.

Buffer (تقویت کننده جریان) :

تراشه‌هایی بنام بافر وجود دارند که ما در راه‌اندازی جریان بیشتر از ۲۰ میلی‌آمپر از آنها استفاده می‌کنیم مانند : 74HC245 و 74HC244 و ULN2003

سیکل ماشین :

CPU و سایر قسمت‌های میکروکنترلر برای انجام هر دستور به میزان خاصی زمان نیاز دارند که به آن سیکل ماشین (پالس ساعت) گفته می‌شود.

(فرکانس اسیلاتور ÷ 1)= سیکل ماشین

تایمر یا کانتر :

برای زمان‌سنجی دقیق از تایمر و برای شمارش پالس‌های بیرونی از کانتر استفاده می‌کنیم. تایمر، پالس ساعت خود را از کلاک سیستم دریافت می‌کند و شروع به شمارش می‌کند اما کانتر، پالس ساعت خود را از پایه Tn دریافت و شمارش می‌کند به طور مثال برای ساختن یک ساعت دیجیتال به تایمر و برای شمارش شیشه‌های نوشابه عبوری از جلوی سنسور به کانتر نیاز داریم.

وقفه‌ها :
منظور از وقفه تأخیر زمانی نیست. وقفه به معنی قطع کردن برنامه جاری و سرویس دادن به تابع وقفه است. برای اینکه میکروکنترلر بتواند علاوه بر برنامه جاری به سایر قسمت‌ها یا المان دیگری سرویس بدهد باید از وقفه استفاده کنیم اجزای جانبی CPU مانند : تایمر و کانتر، مبدل ADC، مقایسه کننده آنالوگ، ارتباط سریال، TWI و ... دارای وقفه مخصوص به خود هستند.

ارتباط سریال USART :
این ارتباط‌دهی برای تبادل اطلاعات بین دو سیستم بوده و ممکن است این ارتباط بین دو میکروکنترلر یا یک میکروکنترلر با PC و یا یک میکروکنترلر با هر تراشه دیگری که دارای این پروتکل باشد، برقرار شود. در این ارتباط‌دهی، اطلاعات ارسالی و دریافتی بر روی ۲ خط صورت می‌گیرد.

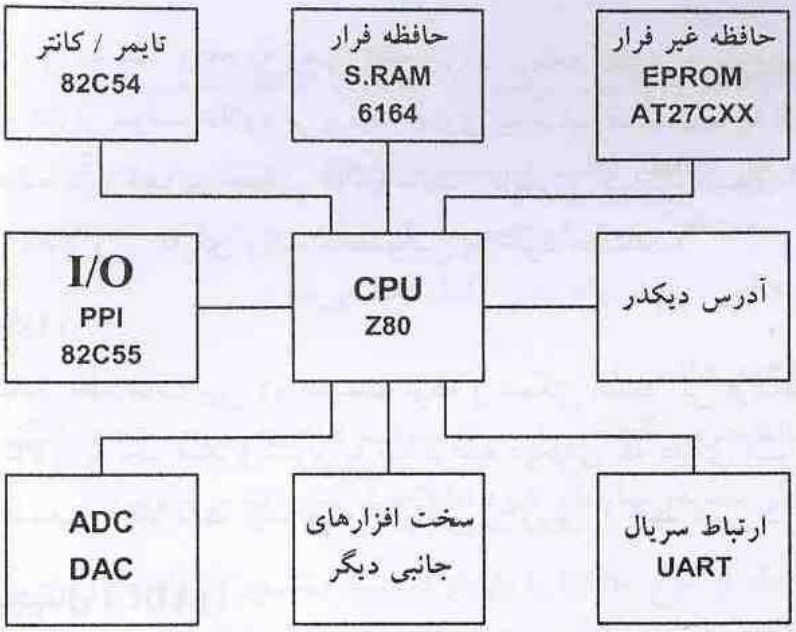
مبدل آنالوگ به دیجیتال (ADC) :
برای تبدیل ولتاژ خوانده شده از یک المان یا یک سنسور، از مبدل آنالوگ به دیجیتال استفاده می‌کنیم. برخی از میکروکنترلرهای AVR دارای مبدل ADC با دقت حداکثر ۱۰ بیت هستند.

مبدل دیجیتال به آنالوگ (DAC) :
برای اینکه بتوانیم ولتاژ متغیر در خروجی، توسط میکروکنترلر ایجاد کنیم باید از مبدل دیجیتال به آنالوگ استفاده کنیم. مانند: (تولید موج سینوسی) لازم به ذکر است که میکروکنترلرهای AVR فاقد مبدل DAC هستند اما از ویژگی PWM تایمرهای آن می‌توان DAC را شبیه‌سازی کرد.

سنسور (Sensor) :
المان یا وسیله‌ای که یک پارامتر فیزیکی مانند : دما، فشار، رطوبت، گاز، مادون قرمز، میدان مغناطیسی و ... را به پارامتر الکتریکی (ولتاژ یا جریان) تبدیل می‌کند.

۱-۲ مقدمه‌ای بر میکروکنترلرها

با پیشرفت علم و تکنولوژی در الکترونیک تراشه‌هایی به عنوان میکروپروسسورها طراحی و تولید شدند تا قبل از سال ۱۹۷۱ میلادی اگر شخص طراح، سیستمی را می‌خواست طراحی کند باید سیستم مورد نظر خود را به شرکت‌های سازنده میکروپروسسور ارائه می‌داد تا طراحی و ساخته شود و یا اینکه مجبور بود با استفاده از آی‌سی‌های دیجیتالی، سیستم مورد نظر خود را طراحی کند. از این پس شرکت‌های سازنده میکروپروسسورها، از جمله شرکت Zilog تصمیم به‌ساخت میکروپروسسوری نمود که بتوان آن را در اختیار کاربر قرار داد و به هر صورت ممکن که می‌خواهد سیستم مورد نظر خود را طراحی کند و به همین دلیل میکروپروسسور Z80 را به بازار عرضه کرد و نرم‌افزار کامپایلر به زبان اسمبلی و پروگرامر آن را نیز ارائه داد. به طور کلی اگر یک شخص از میکروپروسسور ۸ بیتی Z80 برای سیستمی استفاده کند، باید المان‌های جانبی CPU را نیز علاوه بر سخت‌افزار سیستم مورد نظر، در کنار میکروپروسسور Z80 قرار دهد.



شکل ۱-۱ بلوک دیاگرام یک CPU به همراه اجزای جانبی آن

همان طور که مشاهده می‌نمائید برای اینکه از یک میکروپروسسور حتی برای ساده‌ترین سیستم بخواهیم استفاده کنیم باید از المان‌های جانبی دیگری نیز بهره بگیریم. این عمل سبب افزایش قیمت و پیچیده شدن سخت‌افزار پروژه مورد نظر می‌گردد از این پس شرکت‌های سازنده قطعه به فکر تراشه‌ای بودند که تمام امکانات جانبی میکروپروسسور را به همراه خود CPU داشته باشد تا شخص طراح با قیمت مناسب و سخت‌افزاری کمتر بتواند سیستم مورد نظر خود را بسازد.

در سال ۱۹۸۱ میلادی شرکت Intel تراشه‌ای را به عنوان میکروکنترلر خانواده 8051 به بازار عرضه کرد. این میکروکنترلر دارای CPU ۸ بیتی، تایمر و کانتر، وقفه، تبادل سریال، حافظه SRAM و حافظه غیر فرار (FLASH) داخلی می‌باشد. این میکروکنترلر در اوایل، از نوع حافظه PROM یعنی نوع OTP (One Time Program) بهره می‌برد این نوع میکروکنترلر فقط یک بار قابل برنامه‌ریزی بود. بعداً این میکروکنترلر توسعه یافت و از حافظه‌های EPROM استفاده کردند. این نوع میکروکنترلر با شماره 87Cxx آغاز می‌شد و حُسن این حافظه در این بود که می‌توانست توسط پنجره‌ی شیشه‌ای که بالای تراشه قرار داشت در مجاورت نور ماورای بنفش مثل نور خورشید قرار گرفته و بعد از چند دقیقه پاک شود. با پیدایش نسل جدیدی از حافظه‌های ماندگار در میکروکنترلر 8051 از حافظه Flash استفاده کردند این نوع حافظه با ولتاژ الکتریکی نوشته و پاک می‌شود. این سری با شماره 89Cxx آغاز می‌شود که امروزه نیز همچنان مورد استفاده قرار می‌گیرد. شرکت‌های سازنده قطعه دیگری تحت لیسانس شرکت Intel از میکروکنترلر MCS-51 تولید کردند از جمله این شرکت‌ها می‌توان به شرکت Atmel اشاره کرد. قطعاتی که این شرکت از این میکروکنترلر تولید می‌کند با نام AT89Cxx شروع می‌شوند. این شرکت بعداً نوع توسعه یافته 8051 را با سری شماره AT89Sxx ارائه کرد. فرقی که نسخه S با نسخه C دارد در نحوه‌ی برنامه‌ریزی تراشه است زیرا نوع S می‌تواند داخل مدار پروگرام شود.

برای پروگرام کردن نسخه نوع S نیازی به خارج کردن میکروکنترلر از مدار نداریم و می‌توانیم در داخل سیستم، توسط کابلی با یک بافر، عمل پروگرام کردن را از طریق پورت LPT انجام دهیم.

سرانجام شرکت Atmel میکروکنترلرهای جدیدی را مانند دیگر شرکت‌های سازنده قطعه از جمله شرکت Microchip که میکروکنترلرهای PIC را تولید کرده است، شرکت Atmel نیز خانواده AVR را در سال ۱۹۹۷ میلادی به بازار عرضه نموده است. در یک تالار گفتگو به زبان لاتین، برخی بر این باور بودند که واژه AVR مخفف نام سازندگان اصلی شرکت ATMEL با نام‌های Vegard و Alf می‌باشد که حرف R نیز به نوع معماری RISC بکار رفته اشاره دارد. البته دقیقاً مشخص نیست که AVR مخفف چه کلمه‌ای می‌باشد اما ممکن است فقط یک نماد تجاری شرکت ATMEL باشد. خانواده AVR به سه سری تقسیم می‌شوند :

۱. سری AT90S ۲. سری ATtiny ۳. سری ATmega

میکروکنترلرهای AVR قابلیت‌های بیشتری نسبت به میکروکنترلر خانواده 8051 دارند و از نسل جدیدی از حافظه‌های Flash و معماری RISC که در ادامه توضیح می‌دهیم بهره می‌گیرند.

۱. سری AT90S : این سری، اعضای کلاسیک خانواده AVR را تشکیل می‌دهند. این سری قابلیت‌های کمتری نسبت به دو سری بعدی دارند و چون کمتر استفاده می‌شوند با اینکه امکانات کمتری هم دارند قیمت مناسبی ندارند که در مقایسه با سری سوم خیلی مقرون به صرفه نیستند.

AT90S4434	AT90S8535	AT90S4433	AT90S2323	AT90S1200
AT90S8534	AT90S4414	AT90S8515	AT90S2343	AT90S2313

جدول ۱-۱ میکروکنترلرهای سری AT90S

۲. سری ATtiny : این میکروکنترلرها در ابعاد کوچک 8، 20 و 28 پایه هستند و قابلیت‌های خوبی نسبت به سری اول دارند و بیشتر در سیستم‌هایی که نیاز به پورت بالا نیست استفاده می‌شوند. یکی از اعضای ۸ پایه این سری ATtiny85 است که دارای امکانات خوبی از جمله مبدل ADC است.

ATtiny10	ATtiny12	ATtiny15	ATtiny25	ATtiny28	ATtiny85
ATtiny11	ATtiny13	ATtiny22	ATtiny26	ATtiny45	ATtiny2313

جدول ۱-۲ میکروکنترلرهای سری ATtiny

۳. سری ATmega : این سری از میکروکنترلرهای AVR امکانات بیشتری نسبت به دو سری قبلی دارند و توجه مخاطبان را به خود جلب نموده‌اند. در کشور ما نیز این سری زیاد استفاده می‌شود و در اکثر قطعات فروشی‌ها، حتی شهرستانها نیز با قیمت مناسب یافت می‌شود. با توجه به اینکه این سری قابلیت‌های بیشتر و قیمت مناسب‌تری نسبت به دو سری قبلی دارد ما برای این کتاب میکروکنترلر ATmega16 را انتخاب کرده‌ایم. این میکروکنترلر ۴۰ پایه است و از امکانات خوبی بر خوردار است. شما با یادگیری این میکروکنترلر به راحتی می‌توانید با هر یک اعضای خانواده AVR کار کنید. در پروژه‌های کتاب از میکروکنترلر ATmega16 استفاده شده است اما شما می‌توانید با کمی تغییر جزئی،

از برنامه و سخت‌افزار پروژه‌ها، برای میکروکنترلر دیگری از خانواده AVR استفاده نمائید.

ATmega48	ATmega603	ATmega165	ATmega324
ATmega8	ATmega649	ATmega168	ATmega325
ATmega88	ATmega6490	ATmega169	ATmega329
ATmega64	ATmega16	ATmega128	ATmega3290
ATmega640	ATmega161	ATmega1280	ATmega3250
ATmega644	ATmega162	ATmega1281	ATmega256
ATmega6450	ATmega163	ATmega32	ATmega8535
ATmega670	ATmega164	ATmega323	ATmega8515

جدول ۱-۳ میکروکنترلرهای سری ATmega

از آنجا که اصل سازگاری از طرف شرکت‌های سازنده قطعه رعایت می‌شود میکروکنترلرهای AVR تمام قابلیت‌های میکروکنترلر 8051 را دارند و از امکانات جدیدی از جمله ADC، TWI، SPI، EEPROM و ... بر خوردار هستند. اما شرکت Atmel میکروکنترلری از سری AVR را طراحی کرده است که بتواند جایگزین مداراتی شود که در آنها میکروکنترلر 8051 بکار رفته است به طور مثال میکروکنترلر ATmega8515 پایه‌هایی شبیه به AT89C52 دارد تا به توان به سادگی آن را در برد میکروکنترلر قدیمی جایگزین کرد.

معماری میکروکنترلرهای AVR (RISC):

به طور کلی دو نوع معماری برای ساخت میکروکنترلرها وجود دارد:

۱. **معماری CISC** (Complex Instruction Set Computer)

تاریخچه این نوع معماری به قبل از سال ۱۹۸۰ میلادی بر می‌گردد. اکثر میکروپروسسورها و میکروکنترلرهای قدیمی از این نوع معماری، در آنها استفاده شده است. در این معماری تعداد دستورات بیشتر و پیچیده‌تر است اما برنامه‌نویسی آن به خصوص اسمبلی ساده‌تر شده است و از طرفی سرعت اجرایی دستورات پایین‌تر است.

۲. **معماری RISC** (Reduced Instruction Set Computer)

بعد از سال ۱۹۸۰ میلادی، معماری جدیدی طراحی شد. در این نوع معماری تعداد دستورات کاهش پیدا کرد و از طرفی سرعت اجرایی دستورات ۱۰ برابر نسبت به معماری قبلی افزایش یافت و برنامه‌نویسی به زبان اسمبلی را قدری پیچیده و سخت کرد اما با وجود ساختار بهینه شده میکروکنترلرهای AVR با حافظه‌های ظرفیت بالا و همچنین استفاده از معماری RISC، امکان برنامه‌نویسی به زبان‌های سطح بالاتر مانند C و بیسیک فراهم گردید.

تفاوت معماری RISC با معماری CISC:

۱. تعداد و اندازه دستورات در RISC کمتر از CISC است در نتیجه سرعت RISC بیشتر است.
۲. در معماری RISC انتقال داده از رجیستر به رجیستر و یا حافظه به حافظه صورت نمی‌گیرد.
۳. تعداد رجیسترها در معماری RISC بیشتر است.

۴. برنامه‌نویسی به زبان اسمبلی در معماری RISC پیچیده‌تر از CISC است.
۵. اکثر دستورات در معماری RISC در یک کلاک سیکل اجرا می‌شوند.
۶. مصرف توان معماری CISC بیشتر از RISC است.
۷. برنامه‌های MSDOS در معماری RISC قابل اجرا نیست همین دلیل باعث شده است تا اکثر میکروپروسسورها (نظیر 80X86 ساخت شرکت Intel) از معماری CISC استفاده کنند.

تفاوت میکروپروسسورها با میکروکنترلرها :

میکروپروسسورها همان طور که قبلاً ذکر کردیم فقط یک پردازنده کوچک هستند و باید المان‌های جانبی نظیر RAM، ROM، تایمر یا کانتر، وقفه و سایر المان‌های جانبی در کنار میکروپروسسور قرار داده شود تا بتوان یک سیستم چند منظوره را طراحی کرد به طور مثال CPU کامپیوتر یک میکروپروسسور است و PC یک سیستم چند منظوره است که می‌تواند چندین عمل را اجرا کند اما میکروکنترلرها دارای المان‌های جانبی محدود در یک بسته‌بندی نظیر FLASH SRAM تایمر یا کانتر، وقفه و غیره هستند و فقط می‌توانند در سیستم‌های تک منظوره استفاده شوند. میکروکنترلر به معنی کنترل کننده کوچک است به طور مثال کنترل دمای یک دستگاه صنعتی توسط میکروکنترلر یک سیستم تک منظوره است. میکروپروسسورها از معماری CISC استفاده می‌کنند اما میکروکنترلرهای پیشرفته از جمله AVR و PIC از معماری RISC استفاده می‌کنند البته میکروکنترلرهای قدیمی‌تر از جمله 8051 از معماری CISC استفاده می‌کند و اساسی‌ترین تفاوت میکروپروسسورها انعطاف‌پذیری آنهاست که می‌توان RAM و ROM آنها را افزایش داد اما در میکروکنترلرها میزان ظرفیت حافظه‌ها ثابت است.

تفاوت میکروکنترلرهای AVR با دیگر میکروکنترلرها :

اینکه بگوییم چه میکروکنترلری قویتر و بهتر است بسیار سخت است و نمی‌توان در حالت کلی میکروکنترلرها را با هم مقایسه کرد. اما به طور مثال میکروکنترلر PIC به شماره PIC18F452 را می‌توان با میکروکنترلر AVR به شماره ATmega32 مقایسه کرد قابلیت‌های این دو میکروکنترلر بسیار شبیه هم است. برای اینکه شما قضاوت صحیح‌تری داشته باشید پیشنهاد می‌کنم که ویژگی‌های این دو میکروکنترلر را از برگه اطلاعاتی آنها مطالعه کنید.

۱-۳ خصوصیات میکروکنترلر ATmega16

- قابلیت اجرایی بالا و توان مصرفی پایین
- **معماری پیشرفته RISC**
  - ۱۳۱ دستورالعمل قدرتمند که تنها در یک کلاک سیکل اجرا می‌شوند
  - دارای ۳۲ رجیستر ۸ بیتی همه منظوره
  - سرعتی حداکثر تا ۱۶ مگاهرتز برای نوع ATmega16
- **حافظه غیرفرار برنامه و دیتا**
  - 16k بایت حافظه Flash با قابلیت خود برنامه‌ریزی

تحمل حافظه Flash : قابلیت 10,000 بار نوشتن و پاک کردن
- 512 بایت حافظه EEPROM داخلی قابل برنامه‌ریزی
تحمل حافظه EEPROM : قابلیت 100,000 بار نوشتن و پاک کردن
- 1k بایت حافظه SRAM داخلی
- قفل برنامه Flash و EEPROM جهت حفاظت از برنامه نوشته شده

• **قابلیت ارتباط‌دهی JTAG (IEEE Std.)**
- حمایت از اشکال‌زدایی تراشه داخل سیستم
- قابلیت برنامه‌ریزی FLASH، EEPROM، Fuse Bits و Lock bits از طریق JTAG

• **ویژگی‌های جانبی**
- دو تایمر یا کانتر ۸ بیتی با مقسم فرکانسی مجزا و مُد مقایسه‌ای
- یک تایمر یا کانتر ۱۶ بیتی با مقسم فرکانسی مجزا و دارای مُد مقایسه‌ای و مُد Capture
- RTC با اسیلاتور مجزا
- دارای 4 کانال PWM
- 8 کانال مبدل آنالوگ به دیجیتال 10 بیتی
8 کانال Single-ended (سیگنالی که نسبت به زمین سنجیده می‌شود)
دارای 7 کانال تفاضلی فقط برای بسته‌بندی نوع TQFP
دارای 2 کانال تفاضلی با قابلیت برنامه‌ریزی گین 1x، 10x و 200x
- ارتباط سریال دو سیمه (Tow Wire)
- USART سریال قابل برنامه‌ریزی
- ارتباط سریال SPI به صورت Master / Slave
- دارای تایمر Watchdog قابل برنامه‌ریزی با اسیلاتور مجزا داخلی
- مقایسه کننده آنالوگ داخلی

• **ویژگی‌های خاص میکروکنترلر**
- Reset فعال در موقع وصل تغذیه با قابلیت برنامه‌ریزی آشکارساز Brown-Out
- دارای اسیلاتور RC کالیبره شده داخلی
- دارای منابع وقفه داخلی و خارجی
- دارای شش مُد Sleep :
Idle, ADC Noise Reduction, Power-save, Power-down, Standby Extended Standby

• **پایه‌های ورودی و خروجی و نوع بسته‌بندی**
- 32 خط ورودی و خروجی قابل برنامه‌ریزی
- 40 پایه PDIP، 44 پایه TQFP و 44 پایه MLF

• **ولتاژهای عملیاتی**
- 2.7v تا 5.5v برای ATmega16L

- 4.5v تا 5.5v برای ATmega16

- **فرکانس‌های کاری**
  - 0 تا 8MHZ برای ATmega16L
  - 0 تا 16MHZ برای ATmega16
- **مصرف توان با شرایط 1MHZ, 3V, 25°C برای ATmega16L**
  - در حالت فعال 1.1mA
  - در مُد توان Idle : 0.35mA
  - در مُد توان Power-down : کمتر از 1µA

۱-۴ فیوز بیت‌های میکروکنترلر ATmega16

فضای اختصاص داده شده به فیوز بیت‌ها، از نوع حافظه ماندگار است. فیوز بیت‌ها برای تنظیمات خاصی استفاده می‌شوند و با پاک کردن میکروکنترلر از بین نمی‌روند و تغییر آنها فقط از طریق پروگرامر امکان پذیر است و برای تنظیم آنها نیاز به برنامه‌نویسی خاصی نداریم و موقع پروگرام کردن توسط ابزار نرم‌افزار CodeVisionAVR آنها را تنظیم و برنامه‌ریزی می‌کنیم. فیوز بیت‌ها با 0 برنامه‌ریزی و با 1 غیر فعال می‌شوند. توجه کنید که برنامه‌ریزی فیوز بیت‌ها باید قبل از قفل کردن تراشه صورت گیرد. میکروکنترلرهای AVR بسته به نوع قابلیتی که دارند، دارای فیوز بیت‌های متفاوتی هستند ما در این قسمت به توضیح فیوز بیت‌های ATmega16 می‌پردازیم برای این که بدانید میکروکنترلری که با آن کار می‌کنید دارای چه ویژگی و چه فیوز بیت‌هایی می‌باشد، به برگه اطلاعاتی آن در CD همراه کتاب مراجعه نمائید. میکروکنترلر ATmega16 دارای ۲ بایت فیوز بیت طبق جدول‌های ۱-۴ و ۱-۵ می‌باشد.

- **فیوز بیت OCDEN (On Chip Debug Enable)**

موقعیکه فیوز بیت ارتباط دهی JTAG فعال شده باشد و برنامه میکروکنترلر را قفل نکرده باشیم می‌توانیم با فعال کردن فیوز بیت OCDEN برنامه میکروکنترلر را به طور آنلاین در حین اجرا توسط مدار واسط که از ارتباط سریال JTAG استفاده می‌کند و توسط نرم‌افزار AVR Studio مشاهده کنیم. به این نوع آنالیز اِمولاتور (Emulator) یا شبیه‌ساز سخت‌افزاری گفته می‌شود. لازم به ذکر است که فعال کردن این فیوز بیت مصرف توان میکروکنترلر را افزایش می‌دهد.(این فیوز بیت به طور پیش فرض غیر فعال است).

- **فیوز بیت JTAGEN**

با فعال کردن این فیوز بیت می‌توان میکروکنترلر را از طریق ارتباط‌دهی استاندارد JTAG برنامه‌ریزی کرد. (این فیوز بیت به طور پیش فرض فعال است)

نکته مهم : چون پایه‌های ارتباط‌دهی JTAG در میکروکنترلر ATmega16 بر روی PC2 تا PC5

قرار دارد باید در زمانی که ما از این ارتباط‌دهی استفاده نمی‌کنیم آن را غیر فعال کنیم، در غیر اینصورت نمی‌توانیم از پایه‌های PC2 تا PC5 استفاده کنیم.

Fuse High Byte	Bit No.	Description	Default Value
OCDEN(4)	7	Enable OCD	1 (unprogrammed, OCD disabled)
JTAGEN	6	Enable JTAG	0 (programmed, JTAG enabled)
SPIEN(1)	5	Enable SPI Serial Program and Data Downloading	0 (programmed, SPI prog. enabled)
CKOPT(2)	4	Oscillator options	1 (unprogrammed)
EESAVE	3	EEPROM memory is preserved through the Chip Erase	1 (unprogrammed, EEPROM not preserved)
BOOTSZ1	2	Select Boot Size (see Table 100 for details)	0 (programmed)(3)
BOOTSZ0	1	Select Boot Size (see Table 100 for details)	0 (programmed)(3)
BOOTRST	0	Select reset vector	1 (unprogrammed)

جدول ۱-۴ بایت بالایی فیوز بیت‌های ATmega16

Fuse Low Byte	Bit No.	Description	Default Value
BODLEVEL	7	Brown-out Detector trigger level	1 (unprogrammed)
BODEN	6	Brown-out Detector enable	1 (unprogrammed, BOD disabled)
SUT1	5	Select start-up time	1 (unprogrammed)(1)
SUT0	4	Select start-up time	0 (programmed)(1)
CKSEL3	3	Select Clock source	0 (programmed)(2)
CKSEL2	2	Select Clock source	0 (programmed)(2)
CKSEL1	1	Select Clock source	0 (programmed)(2)
CKSEL0	0	Select Clock source	1 (unprogrammed)(2)

جدول ۱-۵ بایت پایینی فیوز بیت‌های ATmega16

- **فیوز بیت SPIEN**

اگر این فیوز بیت فعال باشد می‌توان میکروکنترلر را از طریق ارتباط‌دهی SPI برنامه‌ریزی کرد این فیوز بیت در نرم‌افزار CodeVisionAVR قابل دسترس نیست. (این فیوز بیت به طور پیش فرض فعال است)

- **فیوز بیت CKOPT**

با فعال کردن این فیوز بیت می‌توانیم از حداکثر دامنه نوسان اسیلاتور خارجی استفاده کنیم زمانی که این فیوز بیت فعال باشد، خروجی اسیلاتور به صورت Rail-to-Rail کار می‌کند. یعنی دامنه

طرفی باعث افزایش توان مصرفی در میکروکنترلر می‌شود. (این فیوز بیت به طور پیش فرض غیرفعال است)

- **فیوز بیت EESAVE**

در موقعیکه ما میکروکنترلر را پاک می‌کنیم EEPROM نیز پاک می‌شود. اگر بخواهیم در موقع پاک شدن حافظه Flash از محتوای حافظه EEPROM محافظت کنیم باید این فیوز بیت را فعال کنیم. (این فیوز بیت به طور پیش فرض غیر فعال است)

- **فیوز بیت‌های BOOTSZ0 و BOOTSZ1**

این دو فیوز بیت میزان حافظه اختصاص داده شده BOOT را تعیین می‌کنند و برنامه‌ریزی آنها طبق جدول ۱-۶ می‌باشد. (به طور پیش فرض هر دو فیوز بیت فعال هستند)

BOOTSZ1	BOOTSZ0	Boot Size	Pages	Application Flash Section	Boot Loader Flash Section	End Application section	Boot Reset Address (start Boot Loader Section)
1	1	128 words	2	$0000 - $1F7F	$1F80 - $1FFF	$1F7F	$1F80
1	0	256 words	4	$0000 - $1EFF	$1F00 - $1FFF	$1EFF	$1F00
0	1	512 words	8	$0000 - $1DFF	$1E00 - $1FFF	$1DFF	$1E00
0	0	1024 words	16	$0000 - $1BFF	$1C00 - $1FFF	$1BFF	$1C00

جدول ۱-۶ تعیین ظرفیت حافظه Boot

- **فیوز بیت BOOTRST**

این فیوز بیت برای انتخاب بردار Reset است اگر غیر فعال باشد بردار Reset از آدرس 0X0000 حافظه خواهد بود اما اگر این فیوز بیت فعال شود به ابتدای آدرسی که فیوز بیت‌های BOOTSZ1 و BOOTSZ0 طبق جدول ۱-۶ تعیین کرده‌اند پرش می‌کند.(این فیوز بیت به طور پیش فرض غیر فعال است)

- **فیوز بیت BODEN**

برای فعال کردن مدار Brown-out باید این فیوز بیت فعال شود. مدار Brown-out آشکارساز ولتاژ تغذیه است که اگر از 2.7 یا 4 ولت کمتر شود میکروکنترلر را Reset می‌کند.(این فیوز بیت به طور پیش فرض غیر فعال است)

- **فیوز بیت BODLEVEL**

اگر فیوز بیت BODEN فعال شده باشد و فیوز بیت BODLEVEL برنامه‌ریزی نشده باشد آنگاه با کاهش ولتاژ VCC کمتر از 2.7v میکروکنترلر Reset می‌شود و اگر فیوز بیت BODLEVEL فعال شود آنگاه با کاهش ولتاژ VCC کمتر از 4v میکروکنترلر Reset می‌شود.(این فیوز بیت به طور پیش فرض غیر فعال است)

- **فیوز بیت‌های SUT0 و SUT1**

این دو فیوز بیت، زمان شروع (Start-up) را در موقع وصل تغذیه طبق جدول ۱-۷ تعیین می‌کنند. (به طور پیش فرض SUT0 فعال و SUT1 غیر فعال است)

CKSEL0	SUT1..0	Start-up Time from Power-down and Power-save	Additional Delay from Reset (V_{CC} = 5.0V)	Recommended Usage
0	00	258 CK(1)	4.1 ms	Ceramic resonator, fast rising power
0	01	258 CK(1)	65 ms	Ceramic resonator, slowly rising power
0	10	1K CK(2)	–	Ceramic resonator, BOD enabled
0	11	1K CK(2)	4.1 ms	Ceramic resonator, fast rising power
1	00	1K CK(2)	65 ms	Ceramic resonator, slowly rising power
1	01	16K CK	–	Crystal Oscillator, BOD enabled
1	10	16K CK	4.1 ms	Crystal Oscillator, fast rising power
1	11	16K CK	65 ms	Crystal Oscillator, slowly rising power

جدول ۱-۷ تنظیم فیوز بیت‌های Start-up

- **فیوز بیت‌های CKSEL3 و CKSEL2 , CKSEL1, CKSEL0**

توسط این فیوز بیت‌ها نوع و مقدار فرکانس اسیلاتور را تعیین می‌کنیم. به طور پیش فرض فیوز بیت CKSEL0 غیر فعال و بقیه فعال هستند یعنی فرکانس 1MHZ داخلی انتخاب شده است. اگر بخواهیم فرکانس کاری اسیلاتور داخلی را تنظیم کنیم این فیوز بیت‌ها را طبق جدول ۱-۸ تنظیم می‌کنیم و اگر بخواهیم از کریستال خارجی استفاده کنیم باید این فیوز بیت‌ها را طبق جدول ۱-۹ در حالت یک یعنی غیر فعال قرار دهیم. در تنظیم این فیوز بیت‌ها دقت نمایید به طور مثال اگر اشتباهی تمام این فیوز بیت‌ها را فعال کنیم طبق جدول ۱-۹ مُد کلاک خارجی انتخاب می‌شود که در این حالت میکروکنترلر نه با نوسان‌ساز داخلی و نه با کریستال خارجی کار می‌کند بلکه توسط کلاک خارجی که به پایه XTAL1 اعمال می‌شود کار می‌کند. همچنین توجه کنید اگر شما از کریستال خارجی برای میکروکنترلر خود استفاده می‌نمایید باید حتماً موقع پروگرامر کردن نیز کریستال به میکروکنترلر وصل باشد ولی در حالت استفاده از نوسان‌ساز داخلی نیازی به قرار دادن کریستال بیرونی ندارید.

CKSEL3..0	Nominal Frequency (MHz)
0001(1)	1.0
0010	2.0
0011	4.0
0100	8.0

۱- میکروکنترلر با مقدار پیش فرض 1MHZ تنظیم شده است.

جدول ۱-۸ تنظیم نوسان‌ساز کالیبره شده داخلی

Device Clocking Option	CKSEL3..0
External Crystal/Ceramic Resonator	1111 - 1010
External Low-frequency Crystal	1001
External RC Oscillator	1000 - 0101
Calibrated Internal RC Oscillator	0100 - 0001
External Clock	0000

جدول ۱-۹ تعیین منبع کلاک سیستم

نکته : اگر به طور تصادفی فیوز بیت‌ها را اشتباه تنظیم کردید و با قرار دادن کریستال خارجی، میکروکنترلر توسط پروگرامر شناسایی نشد، یک فرکانس 1MHZ توسط یک میکروکنترلر دیگری به پایه XTAL1 میکروکنترلر مذکور اعمال کنید و توسط پروگرامر، فیوز بیت‌ها را صحیح تنظیم نمائید.

۱-۵ پورت‌های ورودی و خروجی میکروکنترلر ATmega16

هر میکروکنترلر AVR بسته به نوع قابلیت و بسته‌بندی که دارد دارای یک سری پایه‌های ورودی و خروجی است. ممکن است عموماً یک پورت دارای ۸ پایه نباشد به طور مثال پورت C میکروکنترلر ATmega8 دارای ۵ پایه می‌باشد. پورت‌های تمام میکروکنترلرهای AVR می‌توانند به صورت ورودی و خروجی عمل کنند در حالت اولیه Reset تمام ورودی و خروجی‌ها در حالت Tri-state قرار می‌گیرند. Tri-state یعنی حالتی که پایه پورت امپدانس بالا می‌باشد. همچنین تمامی پایه‌های پورت‌ها مجهز به مقاومت بالاکش (Pull-up) داخلی هستند که می‌توانند در حالت ورودی فعال شوند. از آنجایی که جریان‌دهی پورت‌های میکروکنترلرهای قدیمی نمی‌توانستند حتی یک LED را روشن کنند شرکت‌های سازنده میکروکنترلرهای جدید سعی کرده‌اند که جریان‌دهی پایه‌های پورت‌ها را افزایش دهند بافر Latch داخلی میکروکنترلرهای AVR می‌تواند در حالت جریان‌دهی (Source) و جریان کشی (Sink) در حالت فعال، جریانی تا 20mA را تأمین کند البته در حالت حداکثر می‌تواند جریان 40mA را تحمل کند. بنابراین به راحتی می‌تواند یک LED و یا سون سگمنت را جریان‌دهی کند. به طور کلی تمامی پورت‌های میکروکنترلرهای AVR دارای سه رجیستر تنظیم کننده به فرم زیر هستند:

۱. **DDRx.n** : این رجیستر (Data Direction Register) برای تنظیم هر پایه از یک پورت به عنوان ورودی و خروجی در نظر گرفته شده است. اگر بیتی از این رجیستر یک شود نشان‌دهنده تعیین آن پایه به عنوان خروجی و اگر صفر شود آن پایه ورودی خواهد بود.

مثال :

```
DDRA=0xFF;      // تمام پایه‌های پورت A به عنوان خروجی
DDRA=0x00;      // تمام پایه‌های پورت A به عنوان ورودی
DDRC.0=1;       // پایه PC.0 از پورت C به عنوان خروجی
DDRC.0=0;       // پایه PC.0 از پورت C به عنوان ورودی
```

۲. **PORTx.n** : این رجیستر (Port Data Register) برای ارسال دیتا به خروجی می‌باشد هر موقع

میکروکنترلر بخواهد داده‌ای را به خروجی بفرستد باید ابتدا رجیستر DDRx.n در حالت خروجی تنظیم شده باشد و سپس داده مورد نظر در رجیستر PORTx.n قرار می‌گیرد.

مثال :

```
DDRB=0xFF;          //تمام پایه‌های پورت B به عنوان خروجی
PORTB=46;           //عدد ۴۶ دسیمال به خروجی پورت B ارسال می‌گردد
```

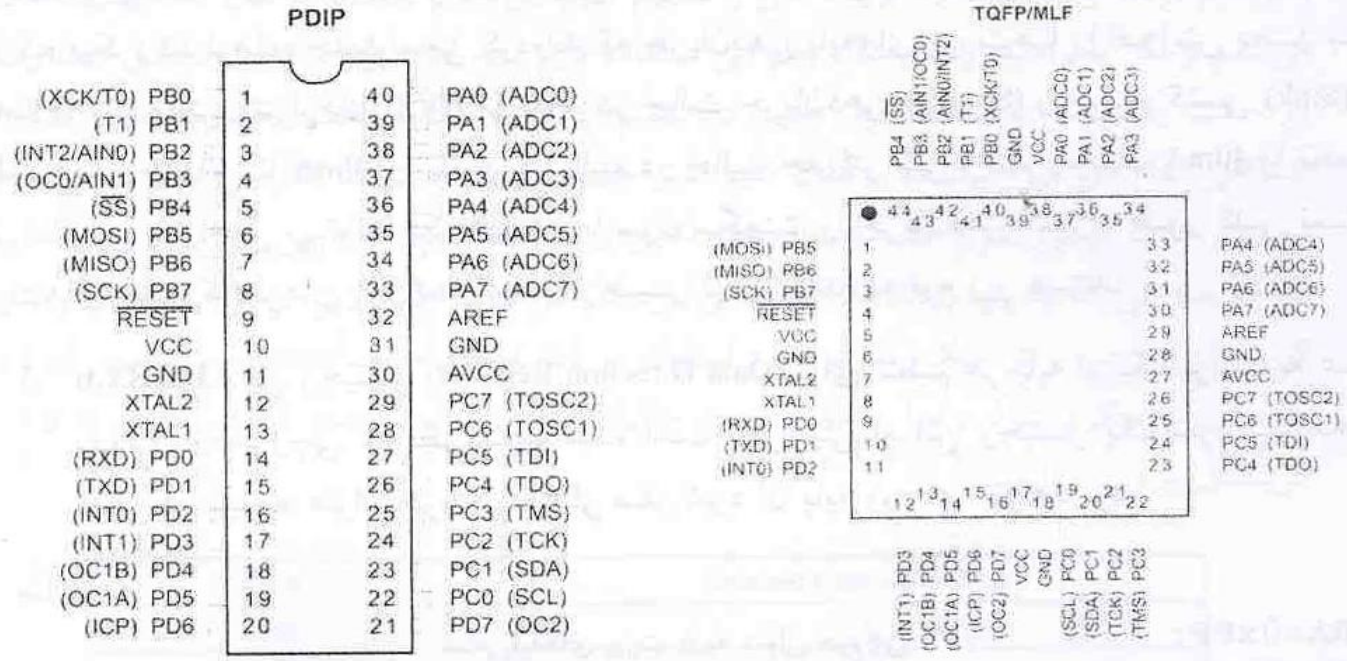
۳. **PINx.n** : این رجیستر (Port Input Pin Address) برای دریافت دیتا از ورودی است هرگاه میکروکنترلر بخواهد داده‌ای را از ورودی بخواند باید ابتدا رجیستر DDRx.n در حالت ورودی تنظیم شده باشد و سپس داده مورد نظر از رجیستر PINx.n به صورت بیتی توسط دستورهای شرطی خوانده می‌شود. همچنین خواندن به صورت بایتی، با یک متغیر ۸ بیتی انجام می‌شود.

مثال :

```
PORTC.5=1;          //فعال کردن مقاومت Pull-up داخلی پایه PC5
DDRC.5=0;           //تعیین پایه PC5 به عنوان ورودی
if(PINC.5 == 0){    //اگر پایه PC5 برابر صفر شد دستور العمل‌ها اجرا شوند
دستور العمل‌ها;
}
```

مثال :

```
DDRC=0x00;          //تعیین تمام پایه‌های پورت C به عنوان ورودی
Data=PINC;          //خواندن دیتا از پورت C و قرار دادن آن در متغیر Data
```



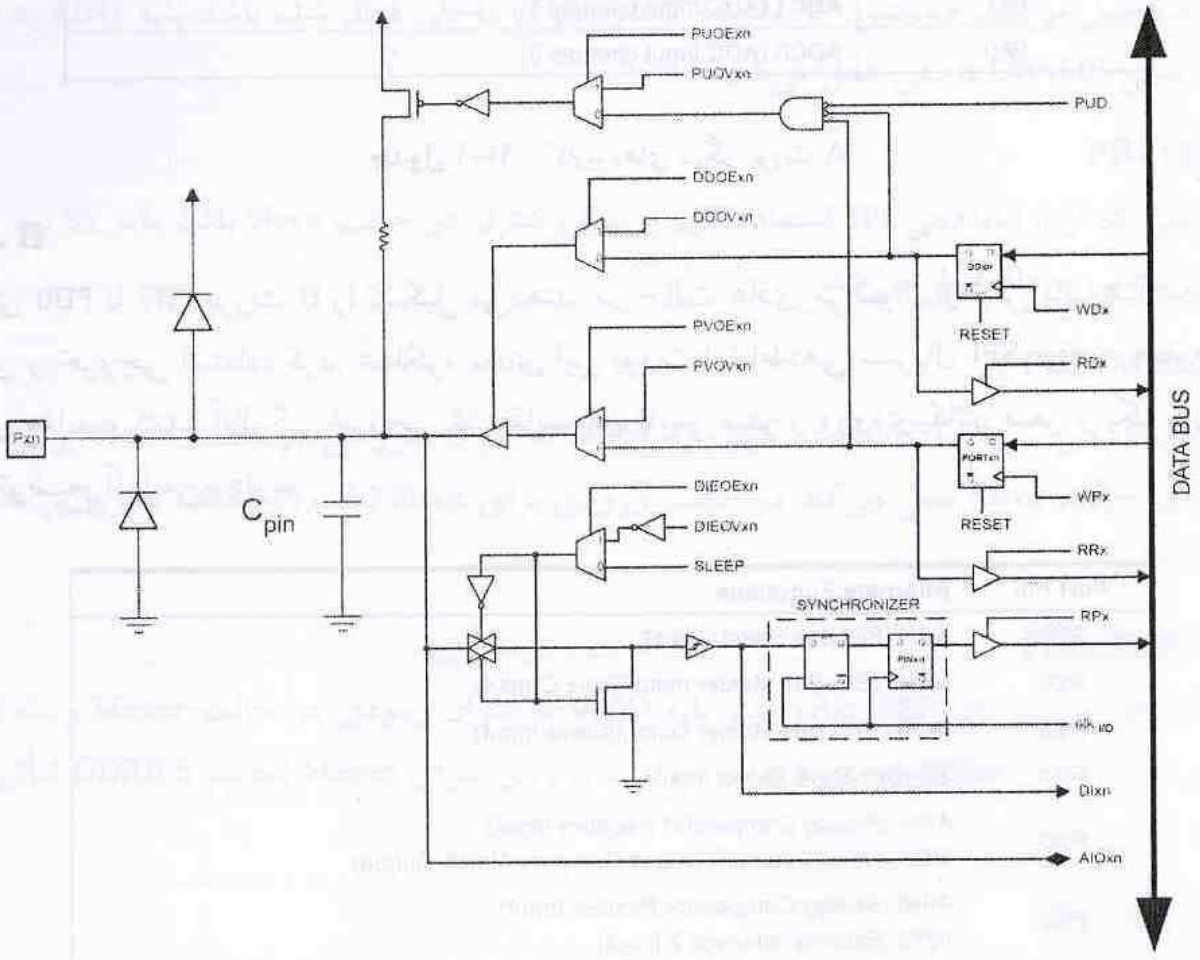
شکل ۱-۲ فرم بسته‌بندی ATmega16 با ترکیب PDIP و TQFP/MLF

در شکل ۱-۳ بلوک دیاگرام، یک پایه از پورت میکروکنترلر سری ATmega را مشاهده می‌فرمائید.

DDxn	PORTxn	PUD (in SFIOR)	I/O	Pull-up	Comment
0	0	X	Input	No	Tri-state (Hi-Z)
0	1	0	Input	Yes	Pxn will source current if ext. pulled low.
0	1	1	Input	No	Tri-state (Hi-Z)
1	0	X	Output	No	Output Low (Sink)
1	1	X	Output	No	Output High (Source)

جدول ۱-۱۰ نحوه‌ی پیکربندی پورت‌ها

همان طور که در شکل ۱-۳ پیداست، هر پایه از پورت میکروکنترلر، توسط دو دیود که به VCC و GND وصل شده‌اند در برابر الکتریسیته ساکن محافظت می‌شود و توسط یک خازن داخلی فرکانس‌های بالا و مخرب که باعث اثرات ناخواسته می‌شوند خنثی می‌شود.



شکل ۱-۳ بلوک دیاگرام یک پایه میکروکنترلر ATmega16

کاربردهای دیگر پورت‌های میکروکنترلر ATmega16

پورت A

پایه‌های PA0 تا PA7 پورت A را تشکیل می‌دهند. در حالت عادی می‌توان از این پایه‌ها به عنوان ورودی و خروجی استفاده کرد. کاربرد بعدی پایه‌های پورت A، به عنوان ورودی مالتی پلکسر مبدل آنالوگ به دیجیتال می‌باشد. توجه کنید در صورتی که شما به طور مثال از کانال ADC0 استفاده

می‌کنید می‌توانید از دیگر پایه‌های پورت A برای کاربردهای دیگر به عنوان ورودی و خروجی استفاده کنید. اما بهتر است در هنگام عمل تبدیل مبدل آنالوگ به دیجیتال داده‌ای به خروجی پورت A ارسال نشود برای آشنایی بیشتر با عملکرد دوم این پورت می‌توانید به فصل مبدل آنالوگ به دیجیتال مراجعه نمائید.

Port Pin	Alternate Function
PA7	ADC7 (ADC input channel 7)
PA6	ADC6 (ADC input channel 6)
PA5	ADC5 (ADC input channel 5)
PA4	ADC4 (ADC input channel 4)
PA3	ADC3 (ADC input channel 3)
PA2	ADC2 (ADC input channel 2)
PA1	ADC1 (ADC input channel 1)
PA0	ADC0 (ADC input channel 0)

جدول ۱-۱۱ کاربردهای دیگر پورت A

پورت B

پایه‌های PB0 تا PB7 پورت B را تشکیل می‌دهند. در حالت عادی می‌توان از این پایه‌ها به عنوان ورودی و خروجی استفاده کرد. عملکرد بعدی این پورت ارتباط‌دهی سریال SPI، وقفه خارجی دو، ورودی مقایسه کننده آنالوگ، خروجی مُد مقایسه‌ای تایمر صفر و ورودی کانتر صفر و یک می‌باشد که به توضیح آنها می‌پردازیم.

Port Pin	Alternate Functions
PB7	SCK (SPI Bus Serial Clock)
PB6	MISO (SPI Bus Master Input/Slave Output)
PB5	MOSI (SPI Bus Master Output/Slave Input)
PB4	$\overline{SS}$ (SPI Slave Select Input)
PB3	AIN1 (Analog Comparator Negative Input) OC0 (Timer/Counter0 Output Compare Match Output)
PB2	AIN0 (Analog Comparator Positive Input) INT2 (External Interrupt 2 Input)
PB1	T1 (Timer/Counter1 External Counter Input)
PB0	T0 (Timer/Counter0 External Counter Input) XCK (USART External Clock Input/Output)

جدول ۱-۱۲ کاربردهای دیگر پورت B

پایه PB0 (XCK/T0)

اگر کانتر صفر استفاده شود ورودی کانتر صفر پایه T0 خواهد بود. همچنین اگر USART در مُد سنکرون کار کند، پایه XCK به عنوان خروجی کلاک همزمان کننده USART عمل می‌کند.

پایه PB1 (T1)

اگر کانتر یک استفاده شود ورودی کانتر یک پایه T1 خواهد بود.

پایه PB2 (INT2/AIN0)

اگر وقفه خارجی دو توسط رجیسترهای مربوطه فعال شود آنگاه پایه INT2 به عنوان ورودی وقفه خارجی دو عمل می‌کند. همچنین اگر مقایسه کننده آنالوگ داخلی فعال شده باشد پایه AIN0 به عنوان ورودی مثبت OPAMP داخلی عمل می‌کند.

پایه PB3 (OC0/AIN1)

در صورتی که از مُد مقایسه‌ای تایمر صفر استفاده کنیم و خروجی مقایسه‌ای فعال شده باشد پایه OC0 به عنوان خروجی مُد مقایسه‌ای تایمر صفر و در مُد PWM تایمر صفر به عنوان خروجی سیگنال PWM تولید شده عمل می‌کند. همچنین اگر مقایسه کننده آنالوگ داخلی فعال شده باشد پایه AIN1 به عنوان ورودی منفی OPAMP داخلی عمل می‌کند.

پایه PB4 (SS)

در شرایطی که از ارتباط‌دهی SPI استفاده کنیم و میکروکنترلر در حالت Slave باشد پایه SS به عنوان ورودی انتخاب Slave عمل می‌کند.

پایه PB5 (MOSI)

اگر از ارتباط‌دهی سریال SPI استفاده کنیم پایه MOSI به عنوان خروجی در حالت Master و به عنوان ورودی در حالت Slave عمل می‌کند در حالت ورودی برای Slave تنظیم DDRB.5 تأثیری بر عملکرد این پایه ندارد.

پایه PB6 (MISO)

اگر از ارتباط‌دهی سریال SPI استفاده کنیم پایه MISO به عنوان ورودی در حالت Master و به عنوان خروجی در حالت Slave عمل می‌کند در حالت ورودی برای Master تنظیم DDRB.6 تأثیری بر عملکرد این پایه ندارد.

پایه PB7 (SCK)

در ارتباط‌دهی SPI پایه SCK به عنوان خروجی Master و به عنوان ورودی کلاک Slave عمل می‌کند زمانی که Slave انتخاب شود این پایه ورودی خواهد بود و اگر Master انتخاب شود باید توسط DDRB.7 جهت داده تنظیم گردد.

پورت C

پایه‌های PC0 تا PC7 پورت C را تشکیل می‌دهند. در حالت عادی می‌توان از این پایه‌ها به عنوان ورودی و خروجی استفاده کرد. عملکرد بعدی این پورت ارتباط‌دهی سریال TWI، ارتباط‌دهی استاندارد JTAG و کریستال پالس ساعت واقعی RTC تایمر دو است.

Port Pin	Alternate Function
PC7	TOSC2 (Timer Oscillator Pin 2)
PC6	TOSC1 (Timer Oscillator Pin 1)
PC5	TDI (JTAG Test Data In)
PC4	TDO (JTAG Test Data Out)
PC3	TMS (JTAG Test Mode Select)
PC2	TCK (JTAG Test Clock)
PC1	SDA (Two-wire Serial Bus Data Input/Output Line)
PC0	SCL (Two-wire Serial Bus Clock Line)

جدول ۱-۱۳ کاربردهای دیگر پورت C

پایه PC0 (SCL)

زمانی که از ارتباط‌دهی سریال دو سیمه TWI استفاده شود، پایه SCL به عنوان کلاک عمل می‌کند. این پایه، کلاک استاندارد I2C است.

پایه PC1 (SDA)

زمانی که از ارتباط‌دهی سریال دو سیمه TWI استفاده شود، پایه SDA به عنوان خط دیتا عمل می‌کند. این پایه، دیتا استاندارد I2C است.

پایه PC2 (TCK)

در ارتباط‌دهی استاندارد JTAG که برای برنامه‌ریزی میکروکنترلر نیز استفاده می‌شود پایه TCK به عنوان کلاک تست به صورت سنکرون عمل می‌کند. توجه کنید در صورت فعال بودن ارتباط‌دهی JTAG نمی‌توان از این پایه به عنوان ورودی و خروجی استفاده کرد. برای غیر فعال کردن این ارتباط‌دهی به بخش توضیح فیوز بیت‌ها مراجعه کنید.

پایه PC3 (TMS)

در ارتباط‌دهی JTAG پایه TMS برای انتخاب مُد تست می‌باشد در صورت فعال بودن JTAG دیگر نمی‌توان از این پایه به عنوان ورودی و خروجی استفاده کرد.

پایه PC4 (TDO)

در ارتباط‌دهی JTAG پایه TDO به عنوان خروجی داده سریال عمل می‌کند و در صورت فعال بودن JTAG دیگر نمی‌توان از این پایه به عنوان ورودی و خروجی استفاده کرد.

پایه PC5 (TDI)

در ارتباط‌دهی JTAG پایه TDI به عنوان ورودی داده سریال عمل می‌کند و در صورت فعال بودن JTAG دیگر نمی‌توان از این پایه به عنوان ورودی و خروجی استفاده کرد.

پایه PC6 (TOSC1)

در صورتی که بخواهیم از RTC تایمر دو استفاده کنیم باید از کریستال 32.768KHZ پالس ساعت

استفاده کنیم پایه TOSC1 به عنوان پایه اول اسیلاتور پالس زمان واقعی می‌باشد. در صورت فعال شدن بیت AC2 در رجیستر ASSR، تایمر دو، پالس خود را از کریستال پالس ساعت تأمین می‌کند و دیگر نمی‌توان از این پایه در این حالت، به عنوان ورودی و خروجی استفاده کرد.

پایه PC7 (TOSC2)

در صورتی که بخواهیم از RTC تایمر دو استفاده کنیم باید از کریستال 32.768KHZ پالس ساعت استفاده کنیم پایه TOSC2 به عنوان پایه دوم اسیلاتور پالس زمان واقعی می‌باشد. در صورت فعال شدن بیت AC2 در رجیستر ASSR، تایمر دو، پالس خود را از کریستال پالس ساعت تأمین می‌کند و دیگر نمی‌توان از این پایه در این حالت، به عنوان ورودی و خروجی استفاده کرد.

پورت D

پایه‌های PD0 تا PD7 پورت D را تشکیل می‌دهند در حالت عادی می‌توان از این پایه‌ها به عنوان ورودی و خروجی استفاده کرد. عملکردهای بعدی این پورت ارتباط‌دهی سریال USART، وقفه‌های خارجی صفر و یک، خروجی‌های مُد مقایسه‌ای تایمر یک و خروجی مُد مقایسه‌ای تایمر ۲ و ورودی Capture تایمر یک می‌باشد.

Port Pin	Alternate Function
PD7	OC2 (Timer/Counter2 Output Compare Match Output)
PD6	ICP (Timer/Counter1 Input Capture Pin)
PD5	OC1A (Timer/Counter1 Output Compare A Match Output)
PD4	OC1B (Timer/Counter1 Output Compare B Match Output)
PD3	INT1 (External Interrupt 1 Input)
PD2	INT0 (External Interrupt 0 Input)
PD1	TXD (USART Output Pin)
PD0	RXD (USART Input Pin)

جدول ۱-۱۴ کاربردهای دیگر پورت D

پایه PD0 (RXD)

در ارتباط‌دهی سریال USART پایه RXD بدون در نظر گرفتن DDRD.0 به عنوان ورودی داده سریال پیکربندی می‌شود.

پایه PD1 (TXD)

در ارتباط‌دهی سریال USART پایه TXD بدون در نظر گرفتن DDRD.1 به عنوان خروجی داده سریال پیکربندی می‌شود.

پایه PD2 (INT0)

اگر وقفه خارجی صفر فعال شده باشد پایه INT0 به عنوان منبع ورودی وقفه صفر عمل می‌کند.

پایه PD3 (INT1)

اگر وقفه خارجی یک فعال شده باشد پایه INT1 به عنوان منبع ورودی وقفه یک عمل می‌کند.

پایه PD4 (OC1B)

در صورت استفاده از مُد مقایسه‌ای تایمر یک و فعال کردن خروجی مقایسه‌ای، پایه OC1B به عنوان خروجی دوم مقایسه‌ای تایمر یک عمل می‌کند همچنین در مُد PWM تایمر یک، این پایه می‌تواند خروجی سیگنال PWM تولید شده باشد.

پایه PD5 (OC1A)

در صورت استفاده از مُد مقایسه‌ای تایمر یک و فعال کردن خروجی مقایسه‌ای، پایه OC1A به عنوان خروجی اول مقایسه‌ای تایمر یک عمل می‌کند همچنین در مُد PWM تایمر یک، این پایه می‌تواند خروجی سیگنال PWM تولید شده باشد.

پایه PD6 (ICP)

اگر واحد تسخیر کننده یعنی مُد Capture تایمر یک فعال شده باشد پایه ICP به عنوان ورودی Capture تایمر یک عمل می‌کند.

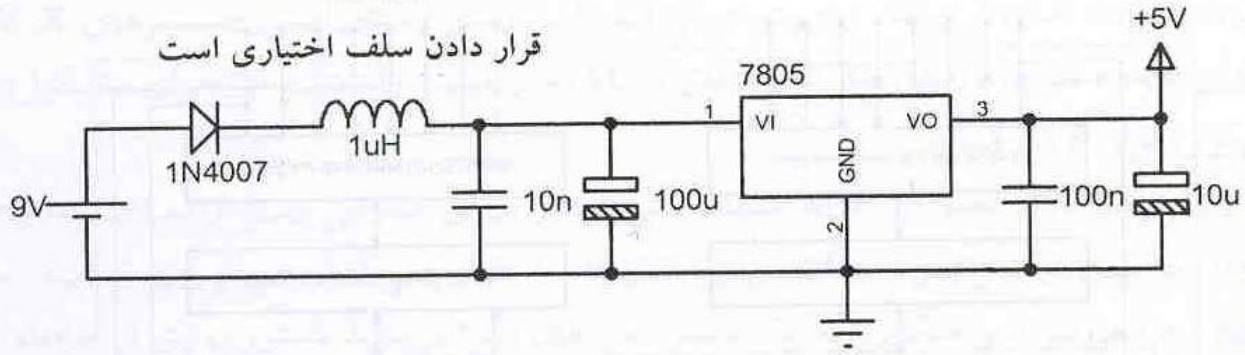
پایه PD7 (OC2)

در صورت استفاده از مُد مقایسه‌ای تایمر ۲ و فعال کردن خروجی مقایسه‌ای، پایه OC2 به عنوان خروجی مقایسه‌ای تایمر دو عمل می‌کند. همچنین در مُد PWM تایمر دو، این پایه می‌تواند خروجی سیگنال PWM تولید شده باشد.

۱-۶ تغذیه مناسب جهت بایاس میکروکنترلر

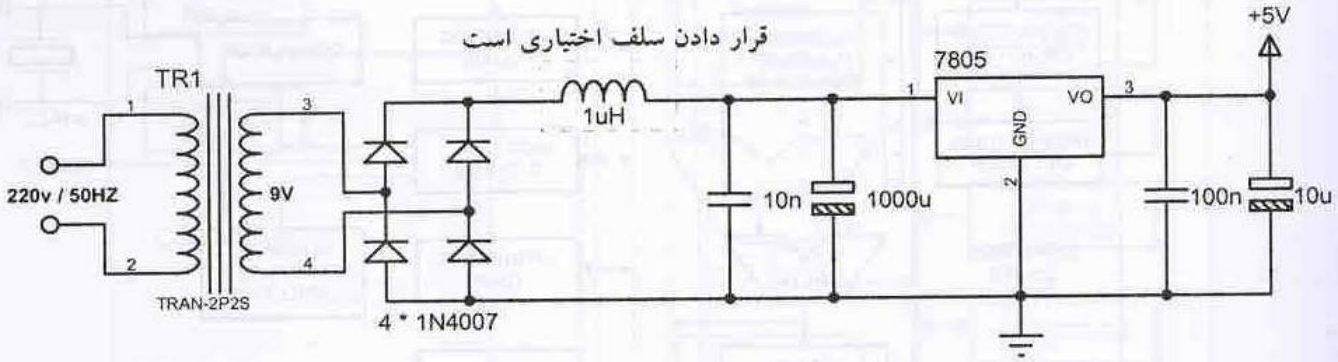
پایه‌های VCC و GND تغذیه میکروکنترلر را تأمین می‌کنند. میکروکنترلرهای AVR همان طور که در ویژگی‌های ATmega16 بیان کردیم می‌توانند با ولتاژ 2.7v تا 5.5v برای نوع L و ولتاژ 4v تا 5.5v برای نوع بدون L کار کنند. اما از آنجایی که اکثر تراشه‌ها و المان‌هایی نظیر LCD و آی‌سی‌های دیجیتال TTL و ... از ولتاژ 5v استفاده می‌کنند بنابراین به طور استاندارد تغذیه میکروکنترلر را 5 ولت انتخاب می‌کنند و چون ولتاژ بیشتر از 5 ولت، باعث سوختن میکروکنترلر می‌شود برای تغذیه آن از رگولاتور 5 ولتی 7805 استفاده می‌شود. البته باید نویز محیط را نیز در نظر بگیریم.

از منبع تغذیه DC ورودی 9 ولتی استفاده کرده‌ایم زیرا ولتاژ ورودی رگولاتور بهتر است 3 ولت بیشتر از ولتاژ نامی باشد. همچنین ورودی را می‌توان 12 ولت انتخاب کرد اما این امر باعث افزایش تلفات و گرما در رگولاتور می‌شود. استفاده از دیود برای محافظت رگولاتور در برابر پلاریته معکوس می‌باشد. سلف 1µH به همراه خازن عدسی 10nF به عنوان فیلتر حذف نویز ورودی رگولاتور عمل می‌کنند و خازن عدسی 100nF نویزهای تغذیه ۵ ولت خروجی رگولاتور را حذف می‌کند و خازن الکترولیتی 100µF در ورودی رگولاتور به صاف کردن تغذیه ورودی کمک می‌کند و همچنین خازن الکترولیتی 10µF خروجی رگولاتور باعث جلوگیری از اُفت ولتاژ تغذیه ۵ ولت می‌شود.



شکل ۱-۴ تغذیه میکروکنترلر با ورودی DC

در صورتی که بخواهیم از تغذیه با ورودی AC برای میکروکنترلر خود استفاده نمائیم باید طبق مدار شکل ۱-۵ عمل کنیم تمام توضیحات داده شده در مدار قبلی، در این مدار نیز صدق می‌کند. تنها تفاوت این مدار وجود ترانس کاهنده ۲۲۰ ولت به ۹ ولت می‌باشد که توسط پل دیود به صورت تمام موج یکسو می‌شود و توسط خازن شیمیایی 1000µF صاف می‌گردد.



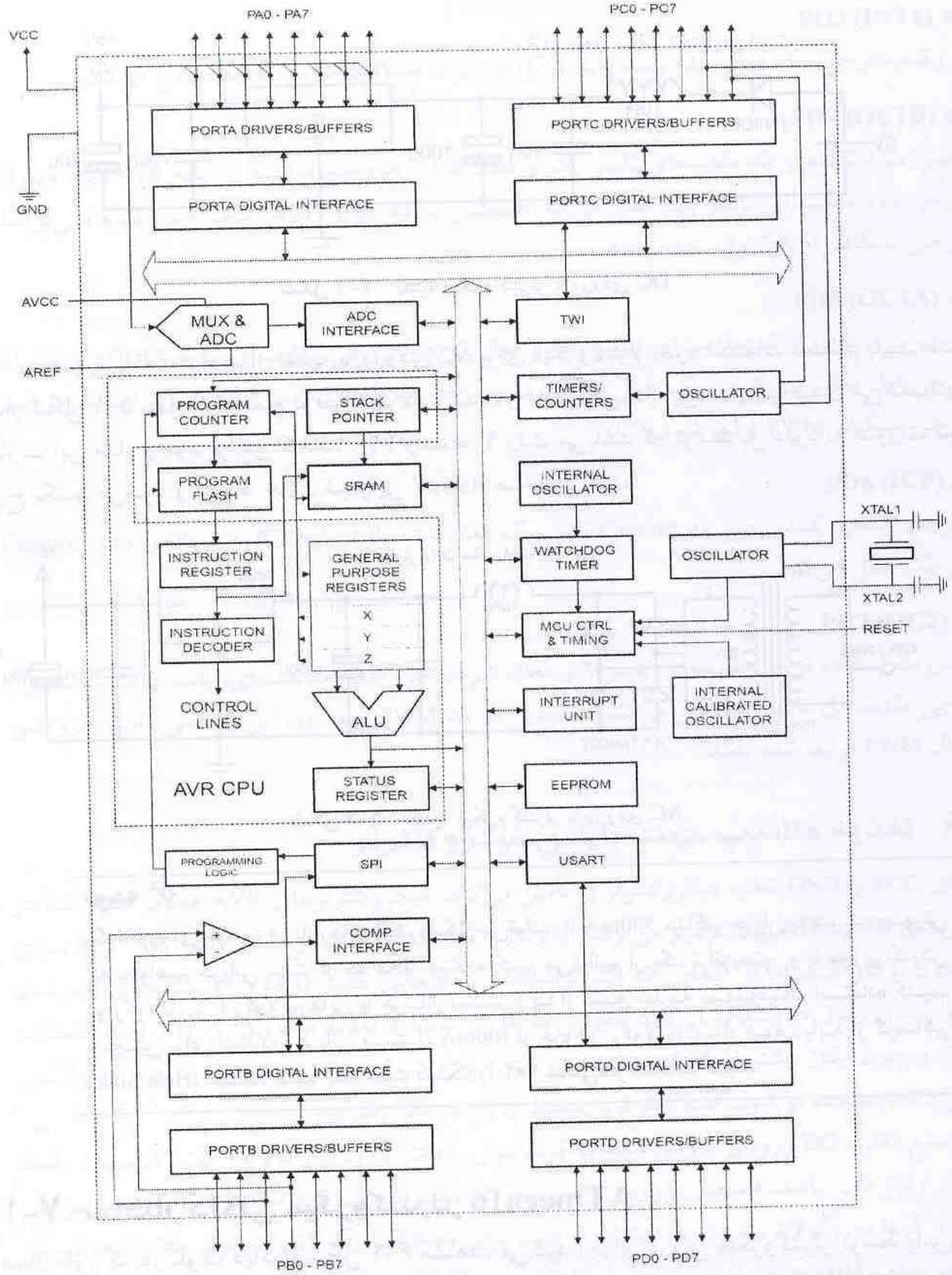
شکل ۱-۵ تغذیه میکروکنترلر با ورودی AC

توجه

رگولاتورهای موجود در بازارهای الکترونیکی می‌توانند تا 500mA حداکثر جریان بدهند در صورتی که بخواهیم جریانی بیشتر از حد مجاز استفاده کنیم می‌توانیم از یک ترانزیستور و یا موازی کردن دو رگولاتور و یا رگولاتورهایی با جریان بیشتر و یا از منبع تغذیه سوئیچینگ استفاده کنیم. همچنین برای استفاده، جریانی بیشتر از 100mA از خروجی رگولاتور مدار فوق، باید از گرماگیر (Heat Sink) استفاده کنیم ابعاد هیت سینک را ۲×۲ سانتی‌متر انتخاب کنید.

۱-۷ ساختار داخلی میکروکنترلر ATmega16

همان طور که در بلوک دیاگرام شکل ۱-۶ مشاهده می‌کنید اجزای درونی میکروکنترلر توسط باس‌های داخلی به هم متصل شده‌اند و یک بخش از این بلوک دیاگرام با خط چین با نام AVR CPU مشخص شده است که در واقع پردازنده اصلی AVR می‌باشد. در این قسمت قصد داریم اجزای تشکیل‌دهنده CPU میکروکنترلر AVR را توضیح دهیم.

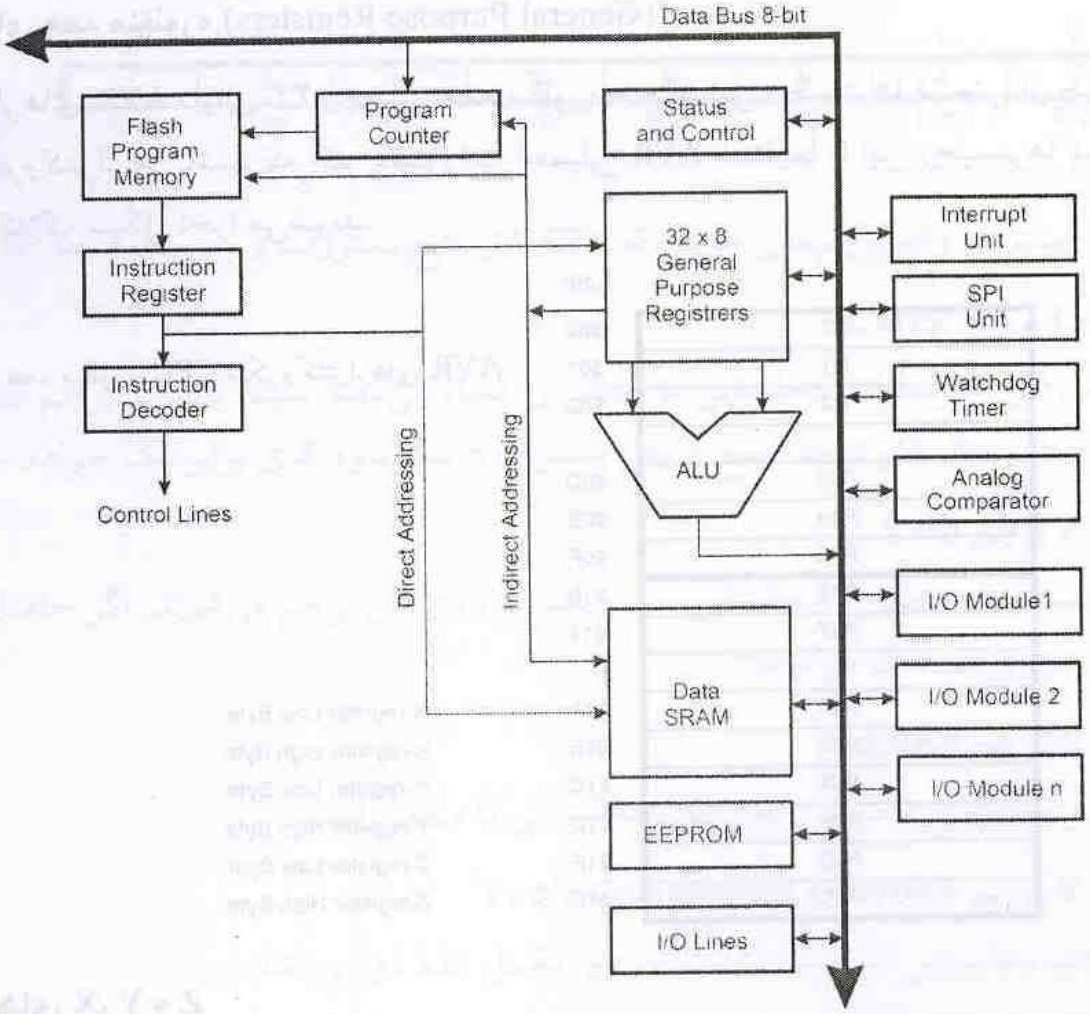


شکل ۱-۶ بلوک دیاگرام داخلی میکروکنترلر ATmega16

کار اصلی یک CPU دسترسی به حافظه‌ها، محاسبات ریاضی و منطقی، کنترل وسایل جانبی و بررسی وقفه‌های هر یک از قسمت‌ها می‌باشد.

شمارنده برنامه، اشاره‌گر پشته، رجیستر دستورات، آشکارساز دستورات، رجیسترهای X، Y و Z، رجیسترهای همه منظوره، واحد محاسبه و منطق (ALU) و رجیستر وضعیت، از اجزای تشکیل‌دهنده CPU میکروکنترلر AVR هستند.

میکروکنترلر AVR از معماری RISC استفاده می‌کند و برای کارایی بهتر از ساختار Harvard و همچنین از حافظه‌ها و باس‌های جداگانه برای انتقال داده استفاده می‌کند. دستورات با یک سطح pipelining اجرا می‌شوند و هنگامی که یک دستور در حال اجرا می‌باشد دستور بعدی از حافظه برنامه pre-fetched می‌شود با این روش هر دستور تنها در یک کلاک سیکل اجرا می‌شود.



شکل ۱-۷ بلوک دیاگرام ساختار میکروکنترلرهای AVR

شمارنده برنامه PC (Program Counter)

CPU میکروکنترلر برای اینکه دستورات را از اولین آدرس حافظه برنامه خط به خط بخواند نیاز به یک شمارنده برنامه می‌باشد. افزایش PC آدرس خط بعدی را برای اجرای دستورات فراهم می‌کند.

اشاره‌گر پشته SP (Stack Pointer)

در میکروکنترلر AVR اشاره‌گر پشته از دو رجیستر ۸ بیتی استفاده می‌کند. اشاره‌گر پشته برای ذخیره موقت اطلاعات در دستورهای فراخوانی CALL ، PUSH و POP و متغیرهای محلی، روتین‌های وقفه و توابع استفاده می‌شود.

رجیستر دستورات (Instruction Register)

در اصل CPU میکروکنترلر برای فهمیدن انجام یک دستورالعمل از کد ماشین استفاده می‌کند و به هر یک از این کدها یک سمبل در زبان اسمبلی اختصاص داده می‌شود. رجیستر دستورات، تمامی دستورهای اسمبلی در نظر گرفته شده از طرف شرکت سازنده Atmel را شامل می‌شود.

آشکارساز دستورات (Instruction Decoder)

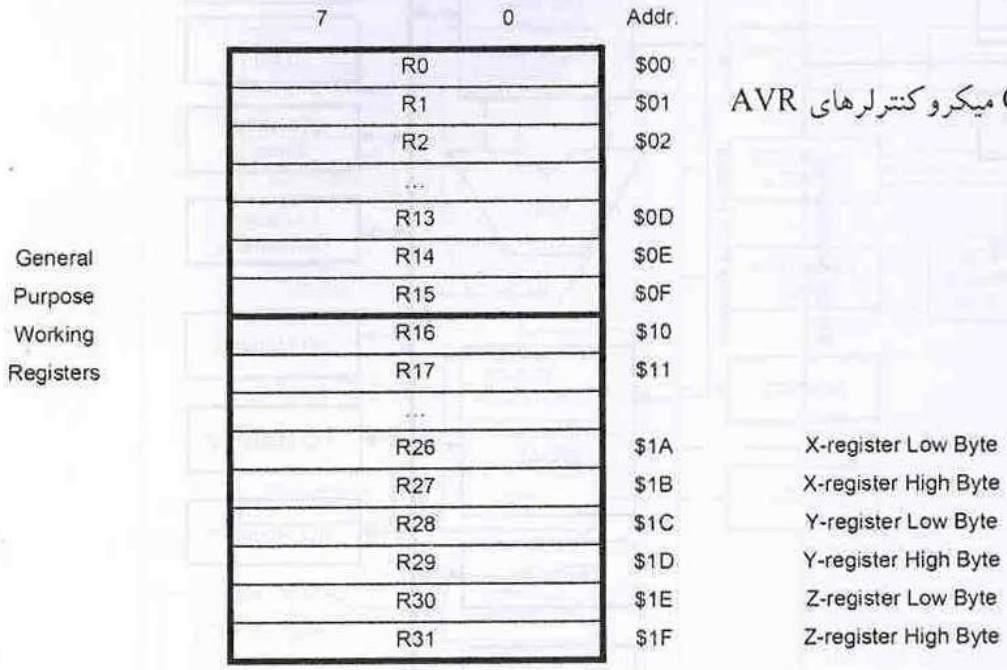
شمارنده برنامه افزایش می‌یابد و کد هر دستور توسط CPU خوانده می‌شود و با توجه به رجیستر دستورات، کد خوانده شده آشکار می‌گردد و CPU متوجه می‌شود که کد دستور به چه معنی و مفهومی است.

رجیسترهای همه منظوره (General Purpose Registers)

میکروکنترلرهای AVR دارای ۳۲ رجیستر همه منظوره هستند این رجیسترها قسمتی از حافظه SRAM داخلی میکروکنترلر می‌باشند که اکثر دستورات اسمبلی AVR مستقیماً با این رجیسترها دسترسی دارند و در یک کلاک سیکل اجرا می‌شوند.

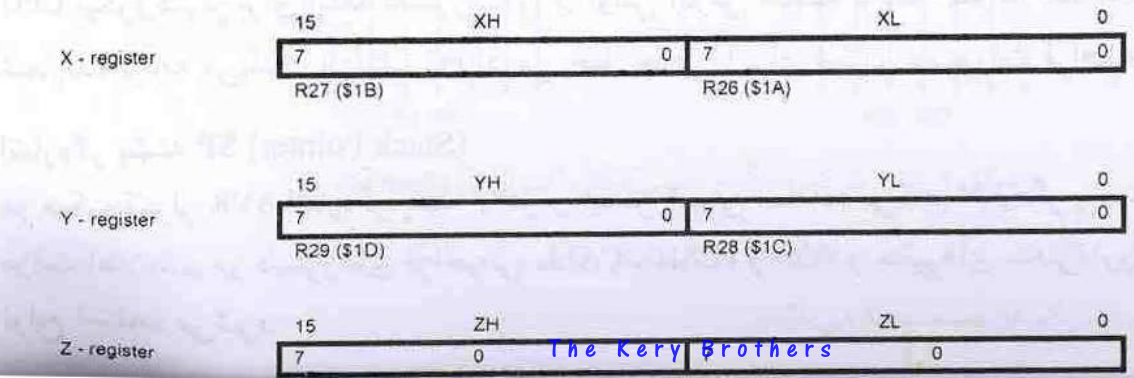
شکل ۱-۸

رجیسترهای همه منظوره CPU میکروکنترلرهای AVR



رجیسترهای X، Y و Z

وظیفه این سه رجیستر که البته از ترکیب رجیسترهای R26 تا R31 بوجود می‌آیند اشاره‌گر ۱۶ بیتی جهت آدرس‌دهی غیر مستقیم فضای داده هستند و بسیاری از دستورات اسمبلی AVR با این سه رجیستر عمل می‌کنند.



واحد محاسبه و منطق ALU (Arithmetic Logic Unit)

ALU در میکروکنترلر AVR به صورت مستقیم با تمام ۳۲ رجیستر همه منظوره ارتباط دارد. عملیاتهای محاسباتی با رجیسترهای همه منظوره در یک کلاک سیکل اجرا می‌شوند به طور کلی عملکرد ALU را می‌توان به سه قسمت اصلی ریاضیاتی، منطقی و توابع بیتی تقسیم‌بندی کرد. در برخی از ALUهای توسعه یافته در معماری میکروکنترلرهای AVR از یک ضرب کننده با قابلیت ضرب اعداد بدون علامت و علامتدار و نیز اعداد اعشاری استفاده شده است.

رجیستر وضعیت SREG (Status Register)

Bit	7	6	5	4	3	2	1	0	
	I	T	H	S	V	N	Z	C	SREG
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

بیت‌های این رجیستر در واقع پرچم‌هایی هستند که CPU را از نتایج دستورات و وضعیت برنامه آگاه می‌کنند.

- **بیت 0 – C پرچم Carry**

انجام دستوراتی که عملیات محاسباتی یا منطقی را انجام می‌دهند سبب تأثیر بر پرچم نقلی(Carry) می‌شود. به طور مثال اگر نتیجه جمع ۸ بیتی بیشتر از ۸ بیت شود کری برابر یک خواهد شد.

- **بیت 1 – Z پرچم Zero**

برخی از دستورات منطقی و محاسباتی سبب تأثیر بر روی این پرچم می‌شوند. اگر حاصل عملیات صفر شود این پرچم فعال می‌گردد.

- **بیت 2 – N پرچم Negative**

این بیت نشانگر نتایج منفی در عملیاتهای محاسباتی یا منطقی است.

- **بیت 3 – V پرچم Two's Complement Overflow**

در دستورات محاسباتی این بیت نشانگر سرریز مکمل عدد دو می‌باشد.

- **بیت 4 – S بیت علامت Sign Bit**

این بیت حاصل XOR دو پرچم N و V می‌باشد. S = N xor V (S=0 علامت + و S=1 علامت –)

- **بیت 5 – H پرچم Half Carry**

در انجام دستورات محاسباتی BCD پرچم نقلی کمکی تأثیر می‌بیند.

- **بیت 6 – T پرچم Bit Copy Storage**

می‌توانیم از بیت T به عنوان مبدأ و یا مقصد یک عملیات بیتی استفاده کنیم.

- **بیت 7 – I پرچم Global Interrupt Enable**

این بیت برای فعال‌سازی وقفه همگانی است اگر این بیت یک شود به دیگر وقفه‌ها در صورت فعال

شدن پاسخ داده می‌شود. در موقع رخ دادن یکی از وقفه‌ها برنامه وارد تابع وقفه مربوطه می‌شود در این حالت این پرچم توسط سخت‌افزار صفر می‌شود و به وقفه دیگری پاسخ داده نمی‌شود و در برگشت از تابع وقفه مربوطه این بیت به طور اتوماتیک توسط سخت‌افزار یک می‌شود.

در زبان برنامه‌نویسی C این بیت به صورت زیر فعال و غیر فعال می‌شود.

```
#asm("sei")        // فعال کردن وقفه همگانی
#asm("cli")        // غیر فعال کردن وقفه همگانی
```

۱-۸ انواع حافظه در میکروکنترلرهای AVR

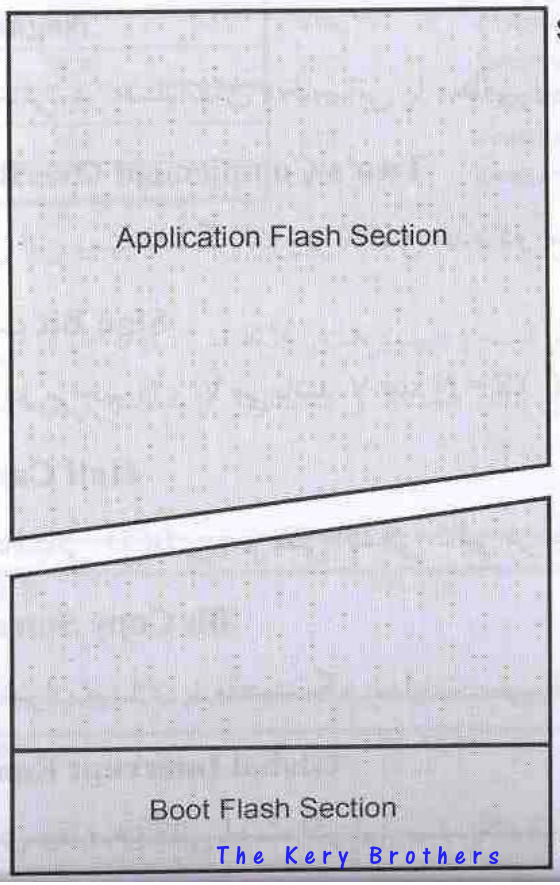
میکروکنترلرهای AVR دارای سه نوع حافظه هستند:

حافظه Flash

در این حافظه کدهای برنامه یعنی همان فایل *.hex که توسط پروگرامر بر روی تراشه Load می‌شود قرار می‌گیرد و CPU میکروکنترلر برنامه را که اجرا می‌کند کد دستورالعمل را از این حافظه برداشت می‌کند. حافظه ثابت (Flash) میکروکنترلرهای AVR از نسل جدید این حافظه می‌باشد و دارای دو قسمت Application و Boot Loader هستند. در قسمت Application کدهای برنامه قرار می‌گیرد اما ناحیه Boot این امکان را فراهم می‌کند که میکروکنترلر بدون استفاده از ابزار پروگرامر، برنامه حافظه Flash را تغییر دهد. به طور مثال نحوه‌ی تعیین یک آرایه در حافظه ثابت بصورت زیر است:

```
flash char row[]={0xfe,0xfd,0xfb,0xf7};    // ۴ عدد ۸ بیتی ذخیره شده در flash
```

شکل ۱-۹ آدرس و تقسیم‌بندی حافظه Flash را به دو قسمت نشان می‌دهد.



شکل ۱-۹

نقشه حافظه برنامه (Flash)

حافظه EEPROM

این حافظه جزء حافظه‌های ماندگار می‌باشد که میکروکنترلر می‌تواند اطلاعاتی را در این حافظه داخلی بنویسد و یا اطلاعاتی را از آن بخواند. همچنین لازم به یادآوری است که این حافظه در صورت قطع تغذیه میکروکنترلر پاک نمی‌گردد. از این حافظه زمانی استفاده می‌شود که میکروکنترلر باید دیتایی را در خود ثبت کند و بعداً آن دیتا را به کاربر اعلام کند و در صورتی که میکروکنترلر Reset شد یا تغذیه آن قطع گردید داده ذخیره شده از بین نرود. باید توجه کنید که برای خواندن و نوشتن در حافظه eeprom باید یک زمان تأخیری در نظر بگیرید. در جدول ۱-۱۵ مدت زمان مناسب با کلاک 1MHZ، 8.5 میلی ثانیه بیان شده است.

تعریف متغیر در حافظه eeprom بصورت زیر است:

```
eeprom unsigned int X=0xff;        // متغیر X در حافظه eeprom ذخیره شده است
```

Symbol	Number of Calibrated RC Oscillator Cycles(1)	Typ Programming Time
EEPROM write (from CPU)	8448	8.5 ms

Note: 1. Uses 1 MHz clock, independent of CKSEL Fuse setting.

جدول ۱-۱۵ زمان برنامه‌ریزی حافظه eeprom

حافظه SRAM

همان طور که ذکر کردیم کدهای برنامه در حافظه Flash قرار می‌گیرند و CPU میکروکنترلر کدهای دستورات را آشکار می‌کند حال باید حاصل دستورات انجام شده در یک حافظه موقت ذخیره گردد این حافظه در میکروکنترلر AVR از نوع Static RAM می‌باشد. رجیسترهای همه منظوره و رجیسترهای ورودی و خروجی نیز جزء این حافظه می‌باشند. محتوای این حافظه با قطع تغذیه پاک می‌گردد و در صورتی که میکروکنترلر را Reset کنیم محتوای رجیسترها صفر می‌شود اما محتوای حافظه SRAM صفر نمی‌شود.

به طور مثال متغیر M را در حافظه SRAM ذخیره کنید.

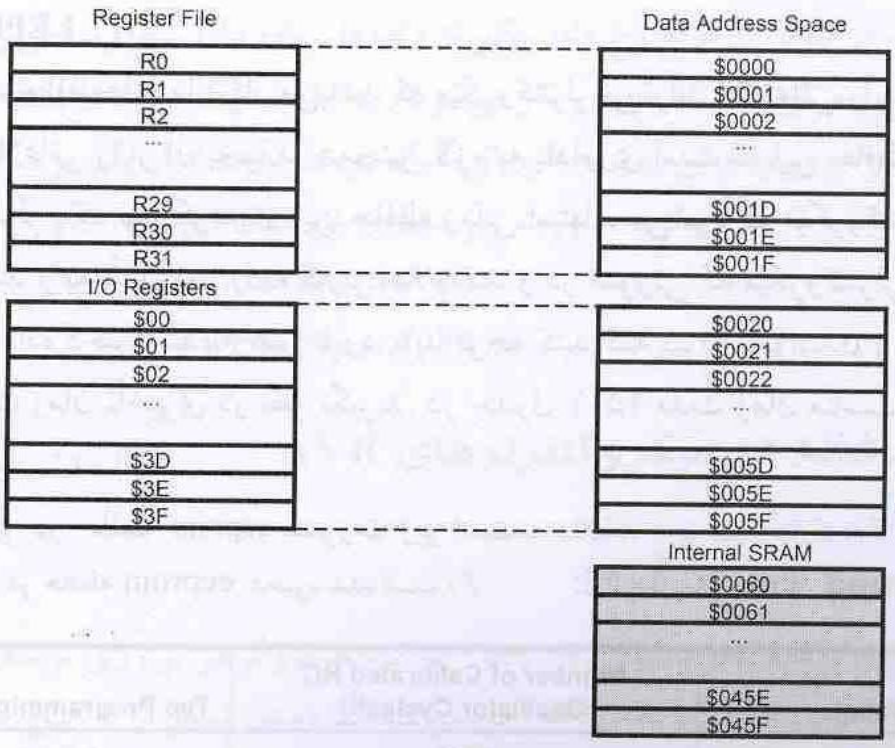
```
unsigned char M=0x12;
```

بنابراین مشخص می‌شود در تعریف هر متغیری که قبل از آن از کلمه کلیدی flash و eeprom استفاده نشود به مفهوم ذخیره متغیر در حافظه SRAM می‌باشد.

شکل ۱-۱۰ تقسیم بندی حافظه SRAM داخلی میکروکنترلر ATmega16 را نشان می‌دهد.

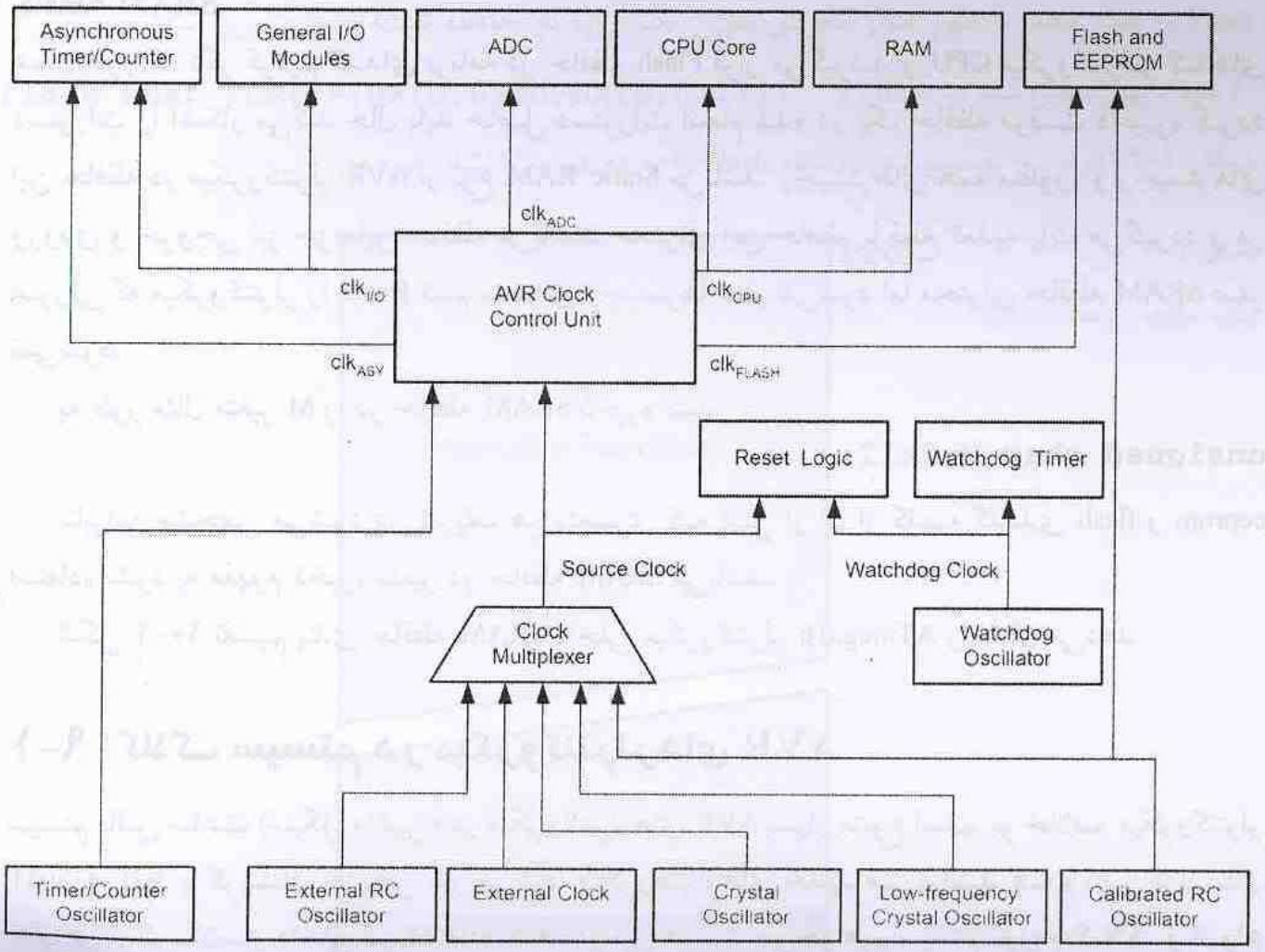
۱-۹ کلاک سیستم در میکروکنترلرهای AVR

سیستم پالس ساعت (سیکل ماشین) در میکروکنترلرهای AVR بسیار متنوع است. بر خلاف میکروکنترلر 8051 که فقط با کریستال خارجی کار می‌کرد، میکروکنترلرهای جدید می‌توانند علاوه بر کریستال خارجی از نوسان‌ساز داخلی نیز استفاده کنند. در این قسمت می‌خواهیم سیستم توزیع کلاک و انواع نوسان‌سازها را توضیح دهیم.



شکل ۱-۱۰ نقشه حافظه دیتا (RAM)

شکل ۱-۱۱ سیستم پالس ساعت را برای قسمت‌های مختلف درونی نشان می‌دهد.



شکل ۱-۱۱ بلوک دیاگرام کلاک سیستم

همان طور که در شکل ۱-۱۱ مشاهده می‌فرمائید منبع کلاک توسط یک مالتی پلکسر، پالس لازم را به واحد کنترل کننده کلاک اعمال می‌کند. در واقع کلاک لازم جهت راه‌اندازی می‌تواند یکی از نوسان‌سازها باشد و امکان استفاده همزمان از آنها وجود ندارد.

پالس CLKCPU به هسته (اجزای درونی) CPU و SRAM داخلی اعمال می‌شود، پالس CLKADC، کلاک لازم را جهت مبدل آنالوگ به دیجیتال فراهم می‌کند که توسط قسمت مبدل ADC می‌تواند به طور مجزا تقسیم گردد، پالس CLKFLASH کلاک لازم را برای حافظه Flash و eeprom داخلی فراهم می‌سازد، پالس CLKI/O برای تولید پالس ماژول‌های ورودی و خروجی نظیر USART، SPI، شمارنده و وقفه‌ها بکار برده می‌شود، پالس CLKASY برای راه‌اندازی آسنکرون تایمر یا کانتر دو برای استفاده از فرکانس 32.768KHZ اسیلاتور RTC است. همچنین تایمر Watchdog از یک اسیلاتور مجزا داخلی استفاده می‌کند.

منابع پالس ساعت میکروکنترلرهای AVR

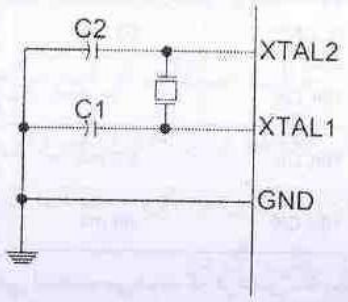
همان طور که ذکر کردیم میکروکنترلرهای AVR می‌توانند از نوسان‌سازهای داخلی و یا خارجی استفاده نمایند. برای تعیین نوسان‌ساز سیستم باید از فیوز بیت‌های میکروکنترلر ATmega16 طبق جدول ۱-۱۶ استفاده کنیم که در قسمت توضیح فیوز بیت‌ها نیز توضیح داده شد.

Device Clocking Option	CKSEL3..0
External Crystal/Ceramic Resonator	1111 - 1010
External Low-frequency Crystal	1001
External RC Oscillator	1000 - 0101
Calibrated Internal RC Oscillator	0100 - 0001
External Clock	0000

جدول ۱-۱۶ تعیین منبع کلاک سیستم

نوسان‌ساز با کریستال خارجی

برای استفاده از کریستال خارجی، باید فیوز بیت‌های CKSEL3..0 را طبق جدول ۱-۱۶ به صورت "1111" برنامه‌ریزی کنیم. پایه‌های خارجی XTAL1 و XTAL2 طبق شکل ۱-۱۲ توسط دو خازن عدسی (خازن‌های بالانس) با مقادیر یکسان به یک کریستال متصل می‌گردند.



شکل ۱-۱۲ نحوهٔ اتصال کریستال خارجی

خازن‌های C1 و C2 را معمولاً 22PF انتخاب می‌کنیم. وظیفه این دو خازن حذف نویز الکترومغناطیس اطراف کریستال می‌باشد که طبق جدول ۱-۱۷ با توجه به کریستال استفاده شده تعیین می‌شوند. در PCB برای حذف نویز، بدنه کریستال خارجی را به زمین وصل می‌کنند اما نباید بدنه کریستال حرارت ببیند زیرا ممکن است به آن آسیب برسد.

CKOPT	CKSEL3..1	Frequency Range (MHz)	Recommended Range for Capacitors C1 and C2 for Use with Crystals (pF)
1	101[1]	0.4 - 0.9	–
1	110	0.9 - 3.0	12 - 22
1	111	3.0 - 8.0	12 - 22
0	101, 110, 111	1.0 ≤	12 - 22

۱- این گزینه در صورت استفاده از رزوناتور سرامیکی، انتخاب می‌شود.

جدول ۱-۱۷ انتخاب خازن‌ها در فرکانس‌های مختلف

هنگام استفاده از کریستال خارجی می‌توان با فعال کردن فیوز بیت CKOPT دامنه نوسان اسیلاتور را حداکثر کرد. در این حالت اسیلاتور به صورت Rail-to-Rail عمل می‌کند یعنی اگر تغذیه میکروکنترلر ۵ ولت باشد حداکثر دامنه پالس ساعت نیز ۵ ولت خواهد بود. این حالت برای محیط‌های پر نویز مانند کارخانه‌های صنعتی بسیار مناسب است. البته فعال کردن این فیوز بیت به اندازه چند میلی آمپر جریان مصرفی میکروکنترلر را افزایش می‌دهد. در میکروکنترلرهای بدون پسوند L اگر بخواهیم از کریستال 16MHZ استفاده کنیم باید فیوز بیت CKOPT فعال گردد در غیر اینصورت حداکثر کریستال خارجی 8MHZ خواهد بود. زمان Start-up برای استفاده از کریستال خارجی توسط فیوز بیت‌های SUT0 و SUT1 طبق جدول ۱-۱۸ تعیین می‌گردد. منظور از زمان Start-up مدت زمانی است که تغذیه به میکروکنترلر وصل می‌شود و بعد از مدتی که نوسانات اسیلاتور پایدار شد میکروکنترلر Reset شده و برنامه را اجرا می‌کند این مدت زمان در حدود چند میلی ثانیه می‌باشد.

CKSEL0	SUT1..0	Start-up Time from Power-down and Power-save	Additional Delay from Reset (V_{CC} = 5.0V)	Recommended Usage
0	00	258 CK[1]	4.1 ms	Ceramic resonator, fast rising power
0	01	258 CK[1]	65 ms	Ceramic resonator, slowly rising power
0	10	1K CK[2]	–	Ceramic resonator, BOD enabled
0	11	1K CK[2]	4.1 ms	Ceramic resonator, fast rising power
1	00	1K CK[2]	65 ms	Ceramic resonator, slowly rising power
1	01	16K CK	–	Crystal Oscillator, BOD enabled
1	10	16K CK	4.1 ms	Crystal Oscillator, fast rising power
1	11	16K CK	65 ms	Crystal Oscillator, slowly rising power

۱- این گزینه‌ها فقط برای زمانی استفاده می‌شوند که فرکانس بالا نباشد و برای کریستال‌ها مناسب نیستند.
۲- این گزینه‌ها فقط برای نوسان‌سازهای رزوناتور سرامیکی مفید هستند.

جدول ۱-۱۸ انتخاب زمان شروع (Start-up) در حالت نوسان‌ساز خارجی

نوسان‌ساز با کریستال فرکانس پایین

منظور از کریستال فرکانس پایین، کریستال 32.768KHZ می‌باشد در صورتی که فیوز بیت‌های CKSEL3..0 به صورت "1001" برنامه‌ریزی شوند پالس ساعت سیستم از کریستال خارجی فرکانس پایین استفاده می‌کند. در این حالت اگر فیوز بیت CKOPT فعال شود خازن داخلی بین دو پایه XTAL1 و XTAL2 فعال می‌گردد و مقدار این خازن 36pF است. نحوه‌ی استفاده از این نوسان‌ساز طبق شکل ۱-۱۲ می‌باشد و همچنین زمان Start-up در این حالت طبق جدول ۱-۱۹ تعیین می‌شود.

SUT1..0	Start-up Time from Power-down and Power-save	Additional Delay from Reset (V_{CC} = 5.0V)	Recommended Usage
00	1K CK(1)	4.1 ms	Fast rising power or BOD enabled
01	1K CK(1)	65 ms	Slowly rising power
10	32K CK	65 ms	Stable frequency at start-up
11	Reserved		

۱- این گزینه‌ها برای کاربردهایی است که پایداری فرکانسی در زمان شروع مهم نباشد.

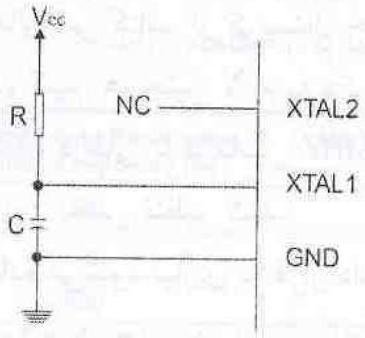
جدول ۱-۱۹ انتخاب زمان شروع (Start-up) در حالت نوسان‌ساز خارجی فرکانس پایین

نوسان‌ساز با RC خارجی

از اسیلاتور با RC خارجی در کاربردهایی که به تغییرات زمان و فرکانس حساسیت نداشته باشیم استفاده می‌کنیم. فرکانس این اسیلاتور از رابطه زیر بدست می‌آید که در آن مقدار خازن حداقل باید 22PF انتخاب شود و مقدار مقاومت بین 3kΩ تا 100kΩ انتخاب می‌شود.

$$f = \frac{1}{3RC}$$

شکل ۱-۱۳ نحوه‌ی استفاده از اسیلاتور خارجی RC را نشان می‌دهد.



شکل ۱-۱۳ نحوهٔ اتصال نوسان‌ساز RC خارجی

با برنامه‌ریزی فیوز بیت CKOPT در حالت استفاده از این نوع نوسان‌ساز، می‌توان خازن 36pF داخلی بین پایه XTAL1 و GND را فعال نموده و از خازن بیرونی استفاده نکنیم.

در این نوع نوسان‌ساز چهار مُد برای محدوده‌های مختلف فرکانسی طبق جدول ۱-۲۰ وجود دارد که می‌توان با فیوزبیت‌های CKSEL3..0 تنظیم گردد.

همچنین در این نوع نوسان‌ساز زمان Start-up توسط فیوز بیت‌های SUT0 و SUT1 طبق جدول ۱-۲۱ تعیین می‌شود.

CKSEL3..0	Frequency Range (MHz)
0101	≤ 0.9
0110	0.9 - 3.0
0111	3.0 - 8.0
1000	8.0 - 12.0

جدول ۱-۲۰ انتخاب محدوده نوسان‌ساز RC خارجی

SUT1..0	Start-up Time from Power-down and Power-save	Additional Delay from Reset (V_{CC} = 5.0V)	Recommended Usage
00	18 CK	–	BOD enabled
01	18 CK	4.1 ms	Fast rising power
10	18 CK	65 ms	Slowly rising power
11	6 CK[1]	4.1 ms	Fast rising power or BOD enabled

۱- زمانی که فرکانس میکروکنترلر بالا باشد، نباید از این گزینه استفاده کنیم.

جدول ۱-۲۱ انتخاب زمان شروع (Start-up) در حالت نوسان‌ساز RC خارجی

نوسان‌ساز با اسیلاتور RC کالیبره شده داخلی

اسیلاتور RC کالیبره شده داخلی می‌تواند فرکانس‌های ثابت 1MHZ، 2MHZ، 4MHZ و 8MHZ را در شرایط تغذیه 5v+ و در دمای 25°C ایجاد نماید. فرکانس کاری این اسیلاتور به شدت به ولتاژ تغذیه، درجه حرارت محیط و مقدار بایت رجیستر OSCCAL وابسته می‌باشد.

از آنجایی که این نوع نوسان‌ساز به دما و ولتاژ وابسته است پیشنهاد می‌کنیم در موقع استفاده از تبادل سریال USART و دیگر پروتکل‌ها و برنامه‌هایی که به زمان بسیار وابسته هستند از کریستال خارجی استفاده کنید. در تمام پروژه‌های این کتاب از کریستال خارجی استفاده شده است که در برخی می‌توانستیم از اسیلاتور داخلی استفاده کنیم. همچنین لازم به ذکر است که میکروکنترلر ATmega16 به طور پیش فرض از اسیلاتور کالیبره شده داخلی با فرکانس 1MHZ استفاده می‌کند که شما باید فیوز بیت‌های CKSEL3..0 را طبق مباحث این بخش تنظیم کنید.

زمانی که از اسیلاتور داخلی استفاده می‌شود نیازی به قرار دادن کریستال خارجی نیست و پایه‌های XTAL1 و XTAL2 آزاد گذاشته می‌شود و همچنین در این نوسان‌ساز، نباید فیوز بیت CKOPT فعال باشد. مقدار فرکانس این اسیلاتور توسط فیوز بیت‌های CKSEL3..0 طبق جدول ۱-۲۲ تعیین می‌شود.

CKSEL3..0	Nominal Frequency (MHz)
0001[1]	1.0
0010	2.0
0011	4.0
0100	8.0

۱- میکرو به طور پیش فرض بر روی 1MHZ داخلی انتخاب شده است.

جدول ۱-۲۲ انتخاب فرکانس نوسان‌ساز کالیبره شده داخلی

زمان Start-up نیز توسط فیوز بیت‌های SUT0 و SUT1 طبق جدول ۱-۲۳ تنظیم می‌گردد.

SUT1..0	Start-up Time from Power-down and Power-save	Additional Delay from Reset (V_{CC} = 5.0V)	Recommended Usage
00	6 CK	–	BOD enabled
01	6 CK	4.1 ms	Fast rising power
10(1)	6 CK	65 ms	Slowly rising power
11	Reserved		

۱- زمان شروع به طور پیش فرض بر روی 65ms انتخاب شده است.

جدول ۱-۲۳ انتخاب زمان شروع (Start-up) در حالت نوسان‌ساز کالیبره شده داخلی

رجیستر کالیبراسیون OSCCAL

Bit	7	6	5	4	3	2	1	0	
	CAL7	CAL6	CAL5	CAL4	CAL3	CAL2	CAL1	CAL0	OSCCAL
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	Device Specific Calibration Value								

در صورت استفاده از اسیلاتور RC کالیبره شده داخلی در هر بار که میکروکنترلر Reset می‌شود مقدار رجیستر OSCCAL، Load می‌شود و اسیلاتور به طور خودکار تنظیم می‌گردد. در حالت عادی استفاده از این نوع اسیلاتور ±3% خطا دارد اما اگر از شرایط تغذیه 5v+ و یا دمای 25°C خارج گردد ممکن است این خطا به 10% افزایش یابد که با توجه به اینکه برای دسترسی به حافظه Flash و حافظه eeprom از این اسیلاتور می‌خواهد استفاده شود ممکن است نتایج نامشخصی داشته باشد. با نوشتن مقدار 0x00 کمترین و با نوشتن مقدار 0xFF در این رجیستر بیشترین فرکانس ممکن انتخاب می‌گردد. تنظیم دقیق این کالیبراسیون خیلی تضمینی نیست. اما مقدار بایت کالیبراسیون را می‌توانیم در هر شرایط دمایی طوری تنظیم کنیم که خطا به ±1 کاهش یابد.

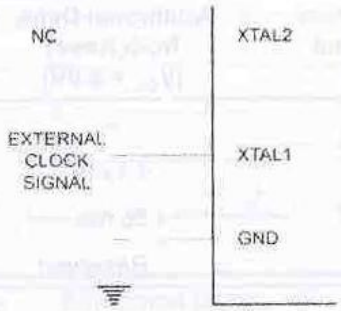
OSCCAL Value	Min Frequency in Percentage of Nominal Frequency (%)	Max Frequency in Percentage of Nominal Frequency (%)
$00	50	100
$7F	75	150
$FF	100	200

جدول ۱-۲۴ محدوده فرکانسی نوسان‌ساز RC کالیبره شده داخلی

نوسان‌ساز با کلاک خارجی

در صورت تنظیم فیوز بیت‌های CKSEL3..0 به صورت "0000" میکروکنترلر AVR پالس ساعت خود را از یک منبع خارجی که به پایه ورودی تقویت کننده نوسان‌ساز یعنی XTAL1 اعمال می‌شود، دریافت می‌کند. با برنامه‌ریزی فیوز بیت CKOPT می‌توان خازن 36pF داخلی بین پایه XTAL1 و GND را فعال کرد. تغییرات فرکانس در این حالت نباید بیشتر از ±2 باشد در غیر اینصورت میکروکنترلر ممکن است رفتار نامشخصی داشته باشد.

شکل ۱-۱۴ نحوه‌ی اعمال کلاک خارجی را نشان می‌دهد و پایه XTAL2 آزاد رها می‌شود.



شکل ۱-۱۴ نحوهٔ اعمال کلاک خارجی به میکروکنترلر

توجه

ممکن است فیوز بیت‌های CKSEL3..0 را به طور ناخواسته به صورت "0000" برنامه‌ریزی کنید اما قصد استفاده از کریستال خارجی را داشته باشید می‌بینید میکروکنترلر شما کار نمی‌کند و توسط پروگرامر شناسایی نمی‌شود برای تغییر فیوز بیت با پروگرامر به شکل صحیح، باید یک فرکانس 1MHZ با دامنه ۵ ولت به پایه XTAL1 اعمال کنید.

زمان Start-up در موقع استفاده از کلاک خارجی به عنوان پالس ساعت سیستم، می‌تواند توسط فیوزبیت‌های SUT0 و SUT1 طبق جدول ۱-۲۵ تنظیم گردد.

SUT1..0	Start-up Time from Power-down and Power-save	Additional Delay from Reset (V_{CC} = 5.0V)	Recommended Usage
00	6 CK	–	BOD enabled
01	6 CK	4.1 ms	Fast rising power
10	6 CK	65 ms	Slowly rising power
11	Reserved		

جدول ۱-۲۵ انتخاب زمان شروع (Start-up) در حالت نوسان‌ساز کلاک خارجی

نوسان‌ساز مجزا تایمر یا کانتر دو

برخی از میکروکنترلرهای AVR از جمله ATmega16 دارای پایه‌های TOSC1 و TOSC2 می‌باشند. زمانی که تایمر دو به عنوان RTC عمل می‌کند از کریستال پالس ساعت (32.768KHZ) که به این دو پایه متصل می‌گردد کلاک دریافت می‌کند. اسیلاتور برای استفاده از این نوع کریستال بهینه شده و نیازی به قرار دادن خازن‌های حذف نویز بیرونی نمی‌باشد. زیرا اسیلاتور فرکانس پایین است و ممکن است، سبب توقف نوسانات اسیلاتور گردد.

۱-۱۰ مُدهای مختلف Sleep

همان طور که در ویژگی‌های میکروکنترلر ATmega16 مطرح کردیم این میکرو دارای شش مُد Sleep

می‌باشد هر یک از این مُدها در عملکردهای متفاوتی کاربرد دارند. هدف از مُدهای توان، کاهش توان مصرفی میکروکنترلر می‌باشد. این مُدها باعث می‌شود تا در پروژه‌هایی که از باتری برای تغذیه استفاده می‌شود مصرف جریان میکروکنترلر کاهش یابد. هنگامی که از مُدهای Sleep استفاده می‌کنیم، کلاک قسمتی از اجزای درونی متوقف می‌شود و با رخ دادن یک وقفه، CPU و دیگر قسمت‌های میکروکنترلر از مُد خواب (Sleep) بیدار می‌شوند.

رجیستر MCUCR (MCU Control Register)

Bit	7	6	5	4	3	2	1	0	
	SM2	SE	SM1	SM0	ISC11	ISC10	ISC01	ISC00	MCUCR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

نیبل بالایی این رجیستر برای تنظیم مُدهای توان می‌باشد.

- **بیت‌های SM2..0 – 7 , 5 , 4**

این بیت‌ها طبق جدول ۱-۲۶ یکی از مُدهای Sleep را تعیین می‌کنند.

- **بیت SE - 6**

اگر این بیت یک شود مُد Sleep فعال می‌شود.

SM2	SM1	SM0	Sleep Mode
0	0	0	Idle
0	0	1	ADC Noise Reduction
0	1	0	Power-down
0	1	1	Power-save
1	0	0	Reserved
1	0	1	Reserved
1	1	0	Standby(1)
1	1	1	Extended Standby(1)

۱- مدهای Standby و Extended Standby فقط برای حالت کریستال یا رزوناتور خارجی می‌باشند.

جدول ۱-۲۶ انواع مُدهای Sleep

کتابخانه sleep.h

برای فعال کردن مُدهای توان در CodeVisionAVR می‌توانید از فایل سرآمد sleep.h استفاده کنید.

```
void sleep_enable(void);        //فعال کردن مُد sleep
void sleep_disable(void);       //غیر فعال کردن مُد sleep
```

مُد Idle

در این حالت CPU میکروکنترلر متوقف می‌شود اما ارتباط‌دهی سریال SPI، TWI و USART،

مقایسه‌کننده آنالوگ، مبدل ADC، تایمر یا کانترها، Watchdog و وقفه‌ها می‌توانند بکار خود ادامه دهند. در این مُد مصرف تست شده با ولتاژ 3v و دمای 25°C و نوسان‌ساز 1MHZ برای ATmega16L، 0.35mA بوده است.

```
void idle(void);      //میکروکنترلر به مُد Idle می‌رود
```

مُد ADC Noise Reduction

در این حالت نیز CPU میکروکنترلر متوقف می‌شود اما ارتباط‌دهی سریال TWI، مبدل ADC، تایمر یا کانتر ۲، Watchdog و وقفه‌های خارجی می‌توانند بکار خود ادامه دهند. کاربرد این مُد برای کاهش نویز محیط، برای ADC می‌باشد. وقتی میکروکنترلر وارد این مُد شده باشد اعمال یک Reset خارجی، Reset توسط تایمر Watchdog، Reset از طریق مدار Brown-out، وقفه کامل شدن تبدیل ADC، وقفه ارتباط‌دهی TWI، وقفه سرریز تایمر یا کانتر۲، وقفه آماده بودن eeprom و هر یک از وقفه‌های خارجی می‌توانند میکروکنترلر را از مُد Sleep بیدار کنند.

مُد Power-down

در این حالت نوسان‌ساز خارجی متوقف می‌گردد. در این مُد وقفه‌های خارجی، ارتباط سریال TWI و تایمر Watchdog می‌توانند بکار خود ادامه دهند. در این مُد مصرف تست شده با ولتاژ 3v و دمای 25°C و نوسان‌ساز 1MHZ برای ATmega16L، کمتر از 1μA بوده است. اعمال یک Reset خارجی، Reset توسط تایمر Watchdog، Reset از طریق مدار Brown-out، وقفه کامل شدن تبدیل ADC، وقفه ارتباط‌دهی TWI و هر یک از وقفه‌های خارجی می‌توانند میکروکنترلر را از این مُد Sleep بیدار کنند.

```
void powerdown(void);      //میکروکنترلر به مُد powerdown می‌رود
```

مُد Power-save

این حالت بسیار شبیه مُد Power-down می‌باشد. تنها تفاوت بین آنها این است که تایمر یا کانتر دو، می‌تواند به صورت غیر همزمان (یعنی بیت AS2 در رجیستر ASSR یک شده باشد) در این مُد بکار خود ادامه دهد. در این حالت میکروکنترلر با رخ دادن وقفه سرریز تایمر یا کانتر۲ و همچنین رخ دادن وقفه تطبیق مقایسه تایمر یا کانتر دو، از این مُد خارج می‌شود.

```
void powersave(void);      //میکروکنترلر به مُد powersave می‌رود
```

مُد Standby

زمانی که از کریستال یا رزوناتور خارجی به عنوان منبع پالس ساعت استفاده می‌شود، می‌توان از این مُد که بسیار عملکردی مشابه مُد Power-down دارد استفاده کرد با این تفاوت که اسیلاتور می‌تواند بکار خود ادامه دهد.

```
void Standby(void);      //میکروکنترلر به مُد Standby می‌رود
```

مُد Extended Standby

زمانی که از کریستال یا رزوناتور خارجی به عنوان منبع پالس ساعت استفاده می‌شود، می‌توان از این

مُد که بسیار عملکردی مشابه مُد Power-save دارد استفاده کرد با این تفاوت که اسیلاتور می‌تواند بکار خود ادامه دهد.

```
void extended_Standby (void);    // میکروکنترلر به مُد extended Standby می‌رود
```

۱-۱۱ منابع Reset میکروکنترلر ATmega16

Reset در واقع به نوعی یک وقفه است که می‌تواند توسط هر یک از منابع تحریک کننده آن رخ دهد. هنگامی که Reset رخ می‌دهد تمامی رجیسترهای ورودی و خروجی و همچنین دیگر رجیسترهای کنترلی با توجه به مقادیر پیش فرض در نظر گرفته شده، تنظیم می‌شوند. بعد از Reset یک مدت زمان کوتاه به اندازه زمان تعیین شده توسط Start-up طول می‌کشد، سپس بعد از پایداری نوسان‌ساز، برنامه از بردار Reset آدرس 0x0000 شروع می‌شود. در صورتی که بردار Reset را توسط فیوزبیت‌های BOOTSZ1..0 تعیین کرده باشیم آنگاه بردار Reset از قسمت Boot حافظه Flash داخلی آغاز می‌شود. سیستم Reset میکروکنترلر ATmega16، از طریق رجیستر MCUCSR منبع Reset را تشخیص می‌دهد و از ۵ طریق ممکن Reset می‌گردد.

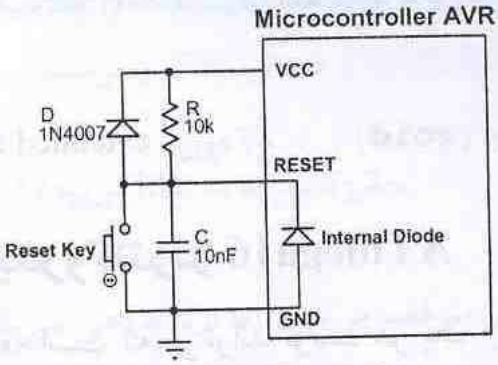
رجیستر MCUCSR

Bit	7	6	5	4	3	2	1	0	
	JTD	ISC2	–	JTRF	WDRF	BORF	EXTRF	PORF	MCUCSR
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	See Bit Description					

۱. **Power-on Reset** : هنگامی که ولتاژ تغذیه از ولتاژ آستانه (VPOT) پایین‌تر شود، بیت PORF در رجیستر MCUCSR یک شده و میکروکنترلر Reset می‌شود.

۲. **External Reset** : اگر یک پالس با سطح صفر منطقی به مدت طولانی‌تر از سیکل پالس ساعت میکروکنترلر، به پایه خارجی RESET اعمال شود، بیت EXTRF در رجیستر MCUCSR یک شده و میکروکنترلر Reset می‌شود. در حالت اجرای برنامه باید این پایه توسط یک مقاومت 10k به VCC وصل گردد. شکل ۱-۱۵ نحوه‌ی وصل کردن مقاومت R را به پایه خارجی Reset نشان می‌دهد. همچنین بدلیل اینکه این پایه، فقط دارای یک دیود داخلی متصل به زمین است و فاقد دیود متصل به VCC است لذا می‌توان با قرار دادن یک دیود خارجی به صورت معکوس از اثر ناخواسته الکتریسته ساکن جلوگیری کرد و نویزهای احتمالی محیط را نیز، می‌توان توسط یک خازن عدسی خنثی نمود و توسط یک کلید فشاری می‌توان میکروکنترلر را به صورت دستی Reset کرد. لازم به ذکر است که قرار دادن دیود، خازن و کلید فشاری اختیاری است.

۳. **Watchdog Reset** : اگر تایمر نگهبان زمان (Watchdog Timer) فعال شود و از مقدار نهایی خود سرریز کند، بیت WDRF در رجیستر MCUCSR یک شده و میکروکنترلر را Reset می‌کند.



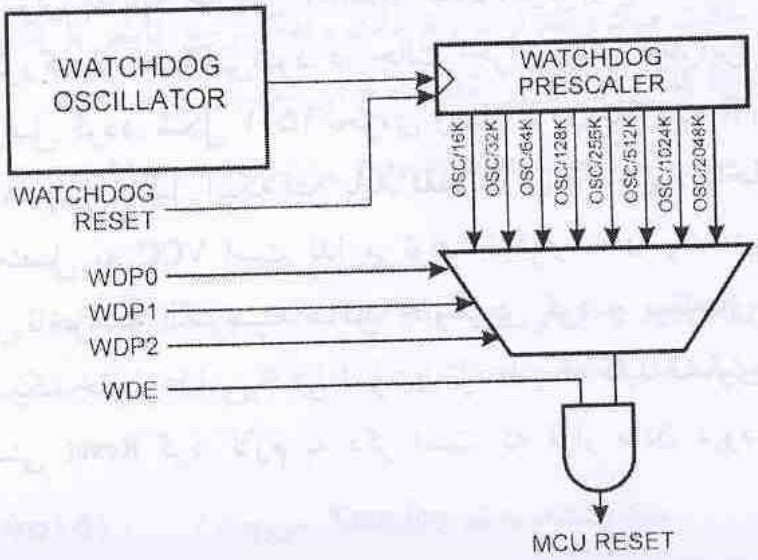
شکل ۱-۱۵ نحوه‌ی استفاده از پایه RESET

۴. **Brown-out Reset** : در صورت فعال بودن مدار Brown-out توسط فیوز بیت BODEN، اگر مقدار ولتاژ تغذیه از حد ولتاژ 2.7v یا 4v که توسط فیوز بیت BODLEVEL تعیین می‌گردد پایین‌تر بیاید، بیت BORF در رجیستر MCUCSR یک شده و میکروکنترلر را Reset می‌کند.

۵. **JTAG AVR Reset** : تا وقتی که در حلقه اسکن سیستم JTAG، رجیستر Reset یک منطقی باشد، بیت JTRF در رجیستر MCUCSR یک شده و میکروکنترلر Reset خواهد شد.

۱-۱۲ تایمر Watchdog

میکروکنترلرهای AVR دارای یک تایمر نگهبان زمان هستند. وظیفه این تایمر در صورت فعال شدن توسط برنامه‌نویسی، این است که از عملکرد صحیح برنامه میکروکنترلر می‌توان اطمینان حاصل کرد. این تایمر زمانی که فعال می‌شود توسط کلاک داخلی مجزا با فرکانس 1MHZ (تحت شرایط تغذیه ۵ ولت) شروع به شمارش می‌کند و بعد از یک مدت زمانی طبق تقسیم فرکانسی تنظیم شده توسط کاربر، این تایمر به مقدار نهایی خود می‌رسد و میکروکنترلر را Reset می‌کند.



شکل ۱-۱۶ بلوک دیاگرام تایمر Watchdog

WDTCR - رجیستر کنترل تایمر Watchdog

Bit	7	6	5	4	3	2	1	0	
	-	-	-	WDTOE	WDE	WDP2	WDP1	WDP0	WDTCR
Read/Write	R	R	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

➢ **بیت‌های WDP2 , WDP1 , WDP0 – 2:0** (Watchdog Timer Prescaler)

توسط این سه بیت طبق جدول ۱-۲۷ می‌توان تقسیم فرکانسی را برای مدت زمان لازم جهت Reset میکروکنترلر، توسط تایمر Watchdog تنظیم نمود.

WDP2	WDP1	WDP0	Number of WDT Oscillator Cycles	Typical Time-out at V_{CC} = 3.0V	Typical Time-out at V_{CC} = 5.0V
0	0	0	16K (16,384)	17.1 ms	16.3 ms
0	0	1	32K (32,768)	34.3 ms	32.5 ms
0	1	0	64K (65,536)	68.5 ms	65 ms
0	1	1	128K (131,072)	0.14 s	0.13 s
1	0	0	256K (262,144)	0.27 s	0.26 s
1	0	1	512K (524,288)	0.55 s	0.52 s
1	1	0	1,024K (1,048,576)	1.1 s	1.0 s
1	1	1	2,048K (2,097,152)	2.2 s	2.1 s

جدول ۱-۲۷ انتخاب تقسیم کنندهٔ فرکانسی کلاک تایمر نگهبان زمان (Watchdog)

➢ **بیت WDE - 3** (Watchdog Enable)

با یک کردن این بیت تایمر نگهبان زمان فعال می‌شود و برای غیر فعال کردن تایمر Watchdog باید علاوه بر صفر کردن بیت WDE، می‌بایست بیت WDTOE را نیز یک کرد.

روش غیر فعال کردن تایمر نگهبان زمان :

۱. نوشتن یک منطقی در بیتهای WDTOE و WDE

۲. نوشتن صفر منطقی پس از چهار سیکل در بیت WDE

➢ **بیت WDTOE - 4** (Watcdog Turn-off Enable)

زمانی که در بیت WDE صفر منطقی نوشته می‌شود باید این بیت یک شود در غیر اینصورت تایمر نگهبان زمان نمی‌تواند غیر فعال شود. با نوشتن یک منطقی در این بیت، پس از چهار سیکل توسط سخت‌افزار به طور اتوماتیک صفر می‌شود.

استفاده از تایمر Watchdog در برنامه‌نویسی C

شاید سؤال شما این باشد که چه لزومی به استفاده از تایمر Watchdog وجود دارد؟ اصلاً چرا باید میکروکنترلر Reset شود؟ جواب این است که میکروکنترلر ممکن است در حلقه‌ای از برنامه منتظر دیتایی شده باشد ولی آن دیتا را به علت نامشخصی دریافت نمی‌کند و یا اینکه نویز سبب شده است

که برنامه میکروکنترلر به بردار آدرسی نامشخصی از حافظه پرش کرده باشد. پس در هر صورت میکروکنترلر هنگ کرده است. با فعال کردن تایمر Watchdog می‌توان محتوای این تایمر را قبل از رسیدن به مقدار نهایی خود، توسط دستور اسمبلی زیر Reset کرد :

```
#asm ("WDR")
```

بنابراین تا زمانی که برنامه به درستی کار می‌کند محتوای این تایمر توسط برنامه Reset می‌شود و اجازه Reset کردن میکروکنترلر را نمی‌دهد. اما اگر میکروکنترلر به قسمتی از برنامه رسید و نتوانست به موقع تایمر نگهبان را Reset کند آنگاه تایمر Watchdog این فرصت را پیدا می‌کند که به مقدار نهایی خود برسد و میکروکنترلر را Reset کند و برنامه را از سرگردانی نجات دهد.

جهت فعال کردن تایمر Watchdog، به طور مثال با زمان سرریز یک ثانیه، باید رجیستر WDTCR به صورت WDTCR=0x0E; مقداردهی شود و برای غیر فعال کردن این تایمر از تابع زیر استفاده می‌کنیم.

```
void WDT_off(void){      //تابع غیر فعال کردن تایمر نگهبان
WDTCR=0x18;              //نوشتن یک در بیت‌های WDE و WDTOE
WDTCR=0x00;              //خاموش شدن تایمر نگهبان زمان
}
```

تمرین‌های فصل ۱

۱-۱ انواع حافظه‌های دائمی و غیر دائمی را نام برده و به اختصار توضیح دهید؟

۱-۲ تفاوت معماری CISC با معماری پیشرفته RISC چیست؟ (۵ مورد را نام ببرید)

۱-۳ تفاوت میکروکنترلر با میکروپروسسور را به اختصار توضیح دهید؟ آیا در هر سیستم، می‌توان این دو را جایگزین یکدیگر کرد؟چرا؟

۱-۴ تفاوت میکروکنترلرهایی که پسوند L دارند با آنهایی که بدون پسوند L هستند در چیست؟

۱-۵ نقش فیوز بیت‌ها در میکروکنترلرهای AVR چیست؟

۱-۶ در صورتی که بخواهیم از کریستال خارجی استفاده کنیم فیوز بیت‌ها را چگونه تنظیم نماییم؟ نقش فیوز بیت CKOPT در این حالت را توضیح دهید؟ با وجود نوسان‌ساز داخلی چه نیازی به کریستال خارجی داریم؟

۱-۷ کاربرد ارتباط‌دهی JTAG را توضیح دهید؟ فیوز بیت JTAGEN چه نقشی دارد؟

۱-۸ یک مدار تغذیه ۵ ولتی که بتواند جریان خروجی حداکثر 1A را تأمین کند، طرح نمائید؟

۱-۹ بلوک دیاگرام داخلی میکروکنترلر ATmega16 را به اختصار ترسیم نمائید و در مورد هر قسمت به صورت خلاصه توضیح دهید؟

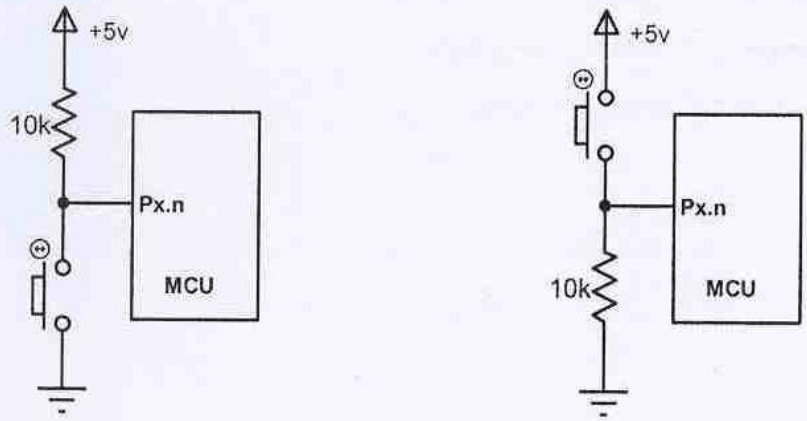
۱-۱۰ انواع منابع پالس ساعت را همراه با ترسیم هر قسمت به طور کامل توضیح دهید؟

۱-۱۱ مُدهای توان در چه نوع پروژه‌هایی کاربرد دارند؟ دو مورد از مُدهای توان را به دلخواه انتخاب کرده و با هم مقایسه کنید؟

۱-۱۲ انواع منابع Reset را در میکروکنترلر ATmega16 نام برده و یکی را به دلخواه توضیح دهید؟

۱-۱۳ تایمر Watchdog در میکروکنترلرهای AVR چه کاربردی دارد؟ نحوه‌ی فعال‌سازی و غیرفعال کردن تایمر نگهبان را در زبان C توضیح دهید؟

۱-۱۴ طبق شکل‌های زیر می‌توان به میکروکنترلر کلید فشاری متصل نمود. کدام شکل ممکن است به پایه پورت آسیب برساند؟ چرا؟



(ب) (الف)

۱-۱۵ در صورتی که به طور اشتباه فیوز بیت‌ها را تنظیم کنیم و پروگرامر ISP قادر به شناسایی میکرو نباشد، مشکل کار کجاست؟ چگونه می‌توان این مشکل را بر طرف نمود؟

۱-۱۶ اگر از کریستال خارجی استفاده نماییم و خازن‌های بالانس C1 و C2 را قرار ندهیم چه مشکلی پیش می‌آید؟ چگونه می‌توانیم کلاک میکروکنترلر را در برابر نویز مصون نماییم؟