میکروکنترلر

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 الی 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

PORT : در هر سیستم تعدادی PORT وجود دارد که وظیفه آنها ورود و خروج اطلاعات است.

Bus : در هر سیستم سه نوع Bus وجود دارد: Data, Address, Control

- Address مشخص می کند با کدام حافظه یا پورت کار داریم .
- Control مشخص می کندچه کاری داریم.(خواندن یا نوشتن...)
- Data توسط این خطوط اطلاعات بین cpu و بقیه مدار منتقل می شود.

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.

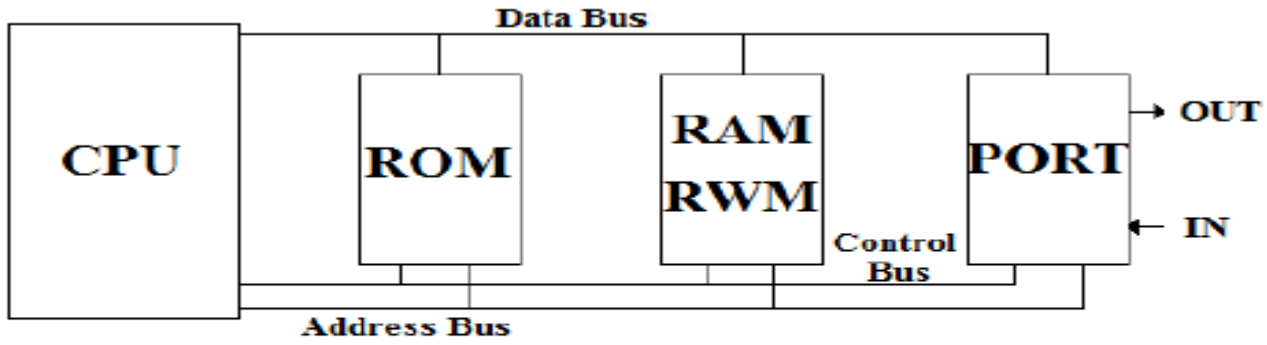
مبدل دیجیتال به آنالوگ (DAC):

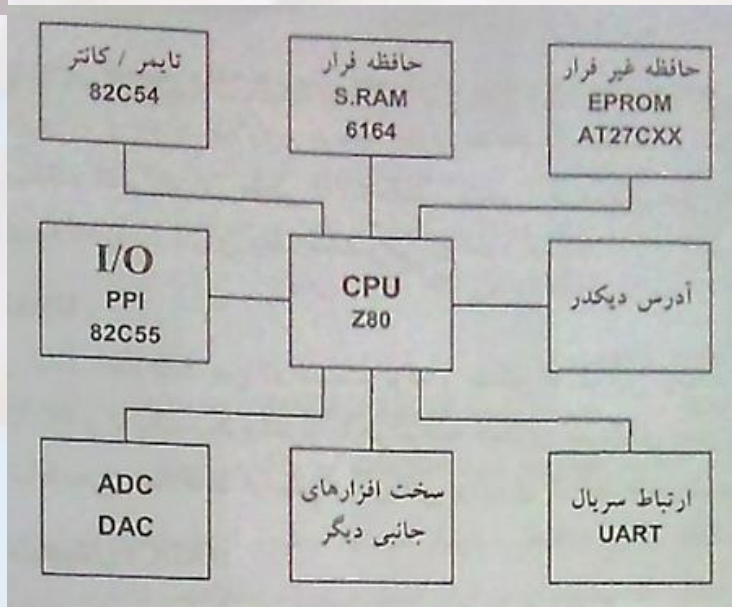
برای اینکه بتوانیم ولتاژ متغیر در خروجی، توسط میکروکنترلر ایجاد کنیم باید از مبدل دیجیتال به آنالوگ استفاده کنیم. مانند: (تولید موج سینوسی) لازم به ذکر است که میکروکنترلرهای AVR فاقد مبدل DAC هستند اما از ویژگی PWM تایمرهای آن می‌توان DAC را شبیه‌سازی کرد.

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.

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.





CPU : Central Processing Unit یا Microprocessor

در هر سیستم یک CPU وجود دارد که کار کنترل و پردازش Data را انجام می دهد.

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.

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،

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.

RISC: دستورهای ساده تر، برنامه نویسی مشکل، سرعت بالا (AVR)

CISC: دستورهای پیچیده، برنامه نویسی ساده تر، سرعت پایین (8051)

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.

۳-۱ 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

۴-۱ 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 امولاتور (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 IMHZ 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 مد 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 IMHZ 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.

۵-۱ 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
```

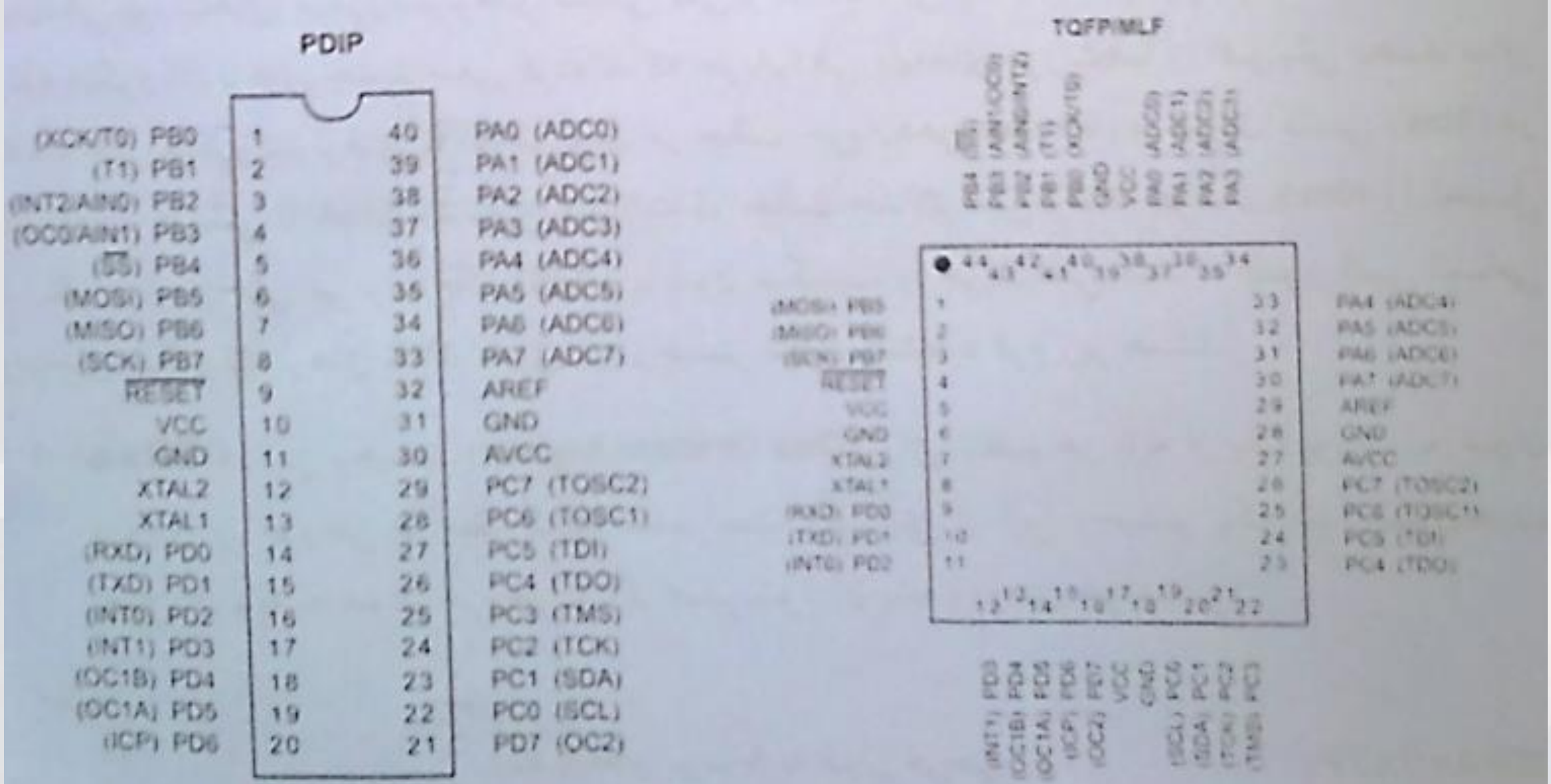
۳. **PINx.n** : این رجیستر (Port Input Pin Address) برای دریافت دیتا از ورودی است هرگاه میکروکنترلر بخواهد داده‌ای را از ورودی بخواند باید ابتدا رجیستر DDRx.n در حالت ورودی تنظیم شده باشد و سپس داده مورد نظر از رجیستر PINx.n به صورت بیتی توسط دستورهای شرطی خوانده می‌شود. همچنین خواندن به صورت بایتی،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

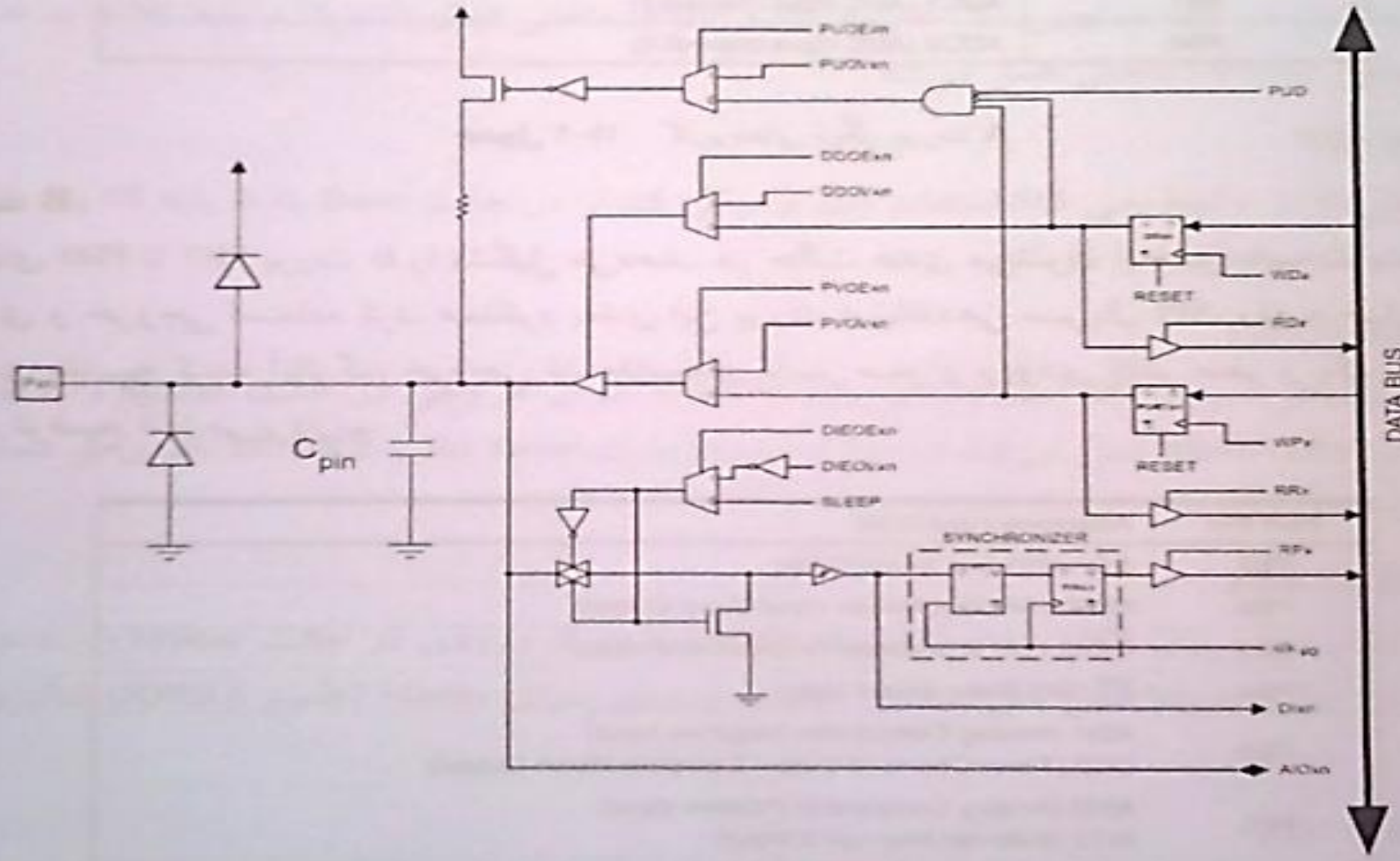


شکل ۱-۲ فرم بسته‌بندی ATmega16 با ترکیب PDIP و TQFP/MLF

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.