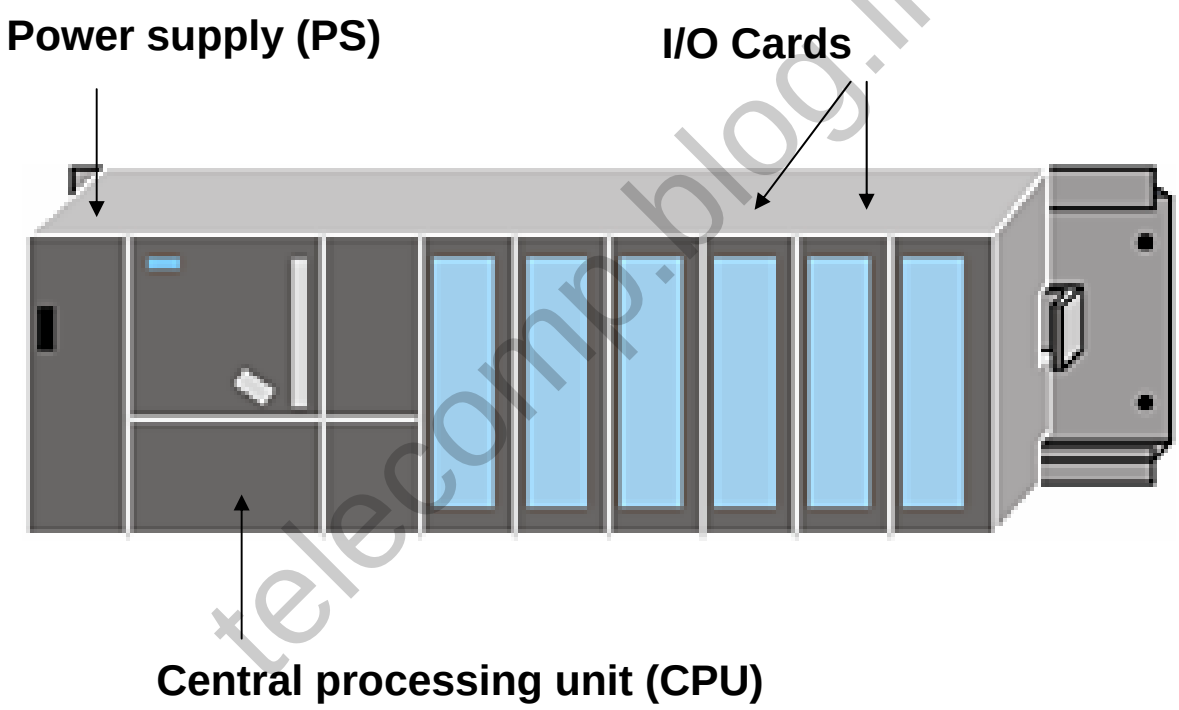
آشنایی با مبانی PLC



مرکز آموزش گروه صنعتی ندا

۱- تاریخچه PLC و تحولات آن ۱-۱

۱-۱) تاریخچه PLC ۱-۱

۱-۲) سازندگان مطرح PLC ۱-۳

۱-۳) کنترل رله- کنتاکتوری ۱-۶

۱-۴) مزایای PLC نسبت به سیستم رله- کنتاکتوری ۱-۸

۲- معرفی PLC و اجزای آن ۲-۱

۲-۱) تعریف PLC ۲-۱

۲-۲) اجزاء تشکیل دهنده یک PLC ۲-۳

۲-۲-۱) منبع تغذیه PS (Power Supply) ۲-۴

۲-۲-۲) واحدپردازش مرکزیCPU (Central Processing Unit) ۲-۵

۲-۲-۳) واحدهای ورودی و خروجی (I/O Units) ۲-۸

۲-۲-۴) واحد حافظه (Memory Unit) ۲-۱۱

۲-۳) انواع PLC ۲-۱۴

۲-۳-۱) PLC یکپارچه ۲-۱۵

۲-۳-۲) PLC ماژولار ۲-۱۶

۳- ورودی ها و خروجی های PLC ۳-۱

۳-۱) تعریف ورودی ها و خروجی هایPLC ۳-۱

۳-۲) تعریف سیگنال دیجیتال ۳-۵

۳-۳) **تعریف** سیگنال آنالوگ ۳-۶

۳-۴) تعریف ورودی و خروجی دیجیتال ۳-۷

۳-۴-۱) ورودی دیجیتال ۳-۷

۳-۴-۲) خروجی دیجیتال ۳-۱۱

۳-۵) تعریف ورودی و خروجی آنالوگ ۳-۱۴

۳-۵-۱) ورودی آنالوگ ۳-۱۴

۳-۵-۲) خروجی آنالوگ ۳-۱۸

۳-۶) کارتهای ورودی و خروجی ۳-۲۰

۳-۷) تبدیلات آنالوگ ها ۳-۲۱

۳-۸) آدرس دهی ورودی خروجی ها ۳-۲۲

۴- برنامه نویسی PLC ۴-۱

۴-۱) سیستم عدد نویسی ۴-۱

۴-۱-۱) سیستم اعشاری (ده دهی) ۴-۲

۴-۱-۲) سیستم باینری (دو دویی) ۴-۳

۴-۱-۲-۱) تبدیل عدد در سیستم باینری به صورت اعشاری ۴-۴

۴-۱-۲-۲) تعریف بیت(Bit) ، بایت(Byte) ،کلمه (word) ۴-۶

۴-۱-۲-۳) منطق صفر و یک ۴-۷

۴-۱-۳) سیستم BCD (Binery Coded Decimal) ۴-۸

۴-۱-۴) سیستم مبنای ۱۶ (HEX) ۴-۹

۴-۱-۴-۱) تبدیل عدد در سیستم اعشاری به صورت HEX ۴-۱۰

۴-۱-۴-۲) تبدیل عدد در سیستم HEX اعشاری به صورت اعشاری ۴-۱۱

۴-۲) اجرای متناوب و چرخشی برنامه ۴-۱۳

۴-۳) ساختار برنامه Program Structure ۴-۱۴

۵- What is IEC 1131-3 ..... ۵-۱

۵-۱) IEC 1131 Ladder Diagram ۵-۲

۵-۲) IEC 1131 Sequential Function Charts ۵-۳

(۵-۳) IEC 1131 Function Block Diagram Overview ۵-۷

(۵-۴) IEC 1131 Structured Text Overview ۵-۸

(۵-۴-۱) Benefits of Structured Text ۵-۹

(۵-۴-۲) Structured Text Examples ۵-۱۰

(۵-۵) IEC 1131 Instruction List Overview ۵-۱۲

(۵-۵-۱) Instruction List example ۵-۱۲

۶- اجزای برنامه نویسی PLC ۶-۱

(۶-۱) دستورالعمل ها ۶-۱

(۶-۱-۱) AND ۶-۲

(۶-۱-۲) OR ۶-۴

(۶-۱-۳) NOT ۶-۶

(۶-۱-۴) SET ۶-۷

(۶-۱-۵) RESET ۶-۸

(۶-۲) مبدل ها (Converter) و مقایسه گرها (Comparators) ۶-۹

(۶-۳) زمان سنج ها (Timers) ۶-۱۰

(۶-۳-۱) Pulse Timer ۶-۱۲

(۶-۳-۲) ON-Delay Timer ۶-۱۳

(۶-۳-۳) OFF-Delay Timer ۶-۱۴

(۶-۴) شمارنده (Counters) ۶-۱۵

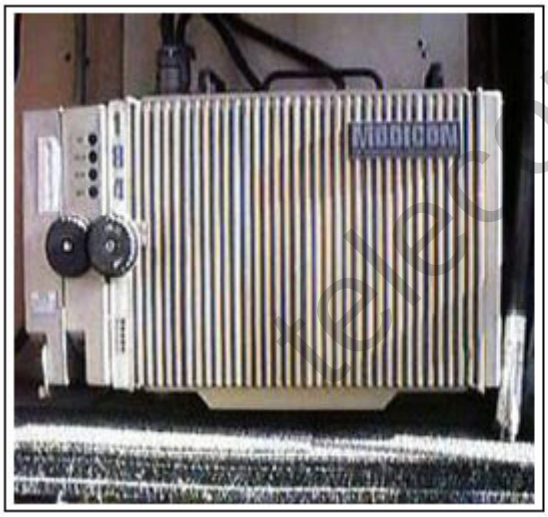
(۶-۴-۱) UP Counter ۶-۱۶

(۶-۴-۲) DOWN Counter ۶-۱۷

۱- تاریخچه PLC و تحولات آن

۱-۱) تاریخچه PLC :

PLC ها تاریخچه کوتاهی دارند و از تولد اولین آنها عمر چندانی نمی گذرد . اولین **PLC** ها در دهه ۷۰ برای استفاده در صنایع اتوموبیل سازی طراحی شدند. نخستین بار کنترلرهای برنامه پذیر توسط شرکت **Modicon** در سال ۱۹۶۸ به در صنعت معرفی شدند که با هدف جایگزینی رله های مکانیکی از آنها استفاده می شد.



اولین PLC، مدل MODICON 084
در سال ۱۹۶۹ اختراع شد.

اولین PLC موفق به طور عملی،
مدل Modicon 184 در سال
۱۹۷۳ به صنعت عرضه شد.

در ابتدا ترغیب کردن صنتگران به استفاده از **PLC** کار چندان ساده ای نبود چون به راحتی قانع نمی شدند که یک مجموعه کوچک از قطعات الکترونیکی به همراه چند خط برنامه بتواند وظایف ۴۰ – ۵۰ تابلوی متشکل از مدارات رله – کنتاکتوری را انجام دهد . اما استفاده از **PLC** با توجه به مزایایی که داشت به تدریج رایج شد و سازندگان متعددی نیز در این رشته پدیدار شدند . با پیشرفت علم الکترونیک **PLC** ها نیز از قابلیت های بهتر و بیشتری برخوردار شدند و در صنایع مختلف به کار گرفته شدند. هم اکنون بیش از میلیونها **PLC** در سراسر دنیا در حال کار هستند و روز به روز نیز به تعداد آنها افزوده می شود.

۱-۲) سازندگان مطرح **PLC** :

در حال حاضر شرکت های زیادی در اکثر کشورهای توسعه یافته تولید کننده **PLC** و قطعات مربوطه هستند . در اینجا به نام چند سازنده که از اعتبار و معروفیت جهانی و معروفیت جهانی برخوردارند اشاره می شود:

- Siemens
- Omron
- Modicon
- GE Fanuc
- Allen-Bradley

شرکت آلمانی Siemens **سازنده** PLC **های سری** S5 **و**S7

SIEMENS

شرکت آمریکایی Allen-Bradely سازنده PLC های سری Control Logix

Rockwell Automation
Allen-Bradley

شرکت ژاپنی OMRON سازنده PLC های سری SYSMAC

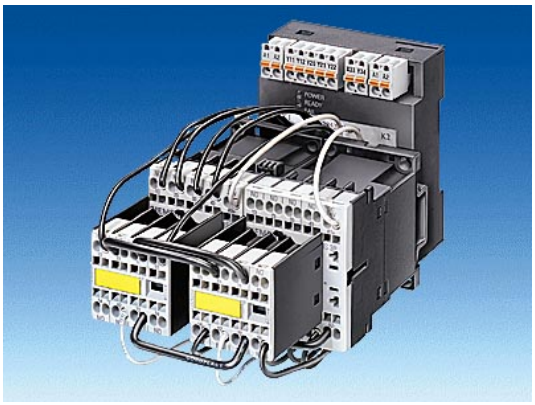
OMRON

شرکت آمریکایی Modicon سازنده PLC های سری Quantum

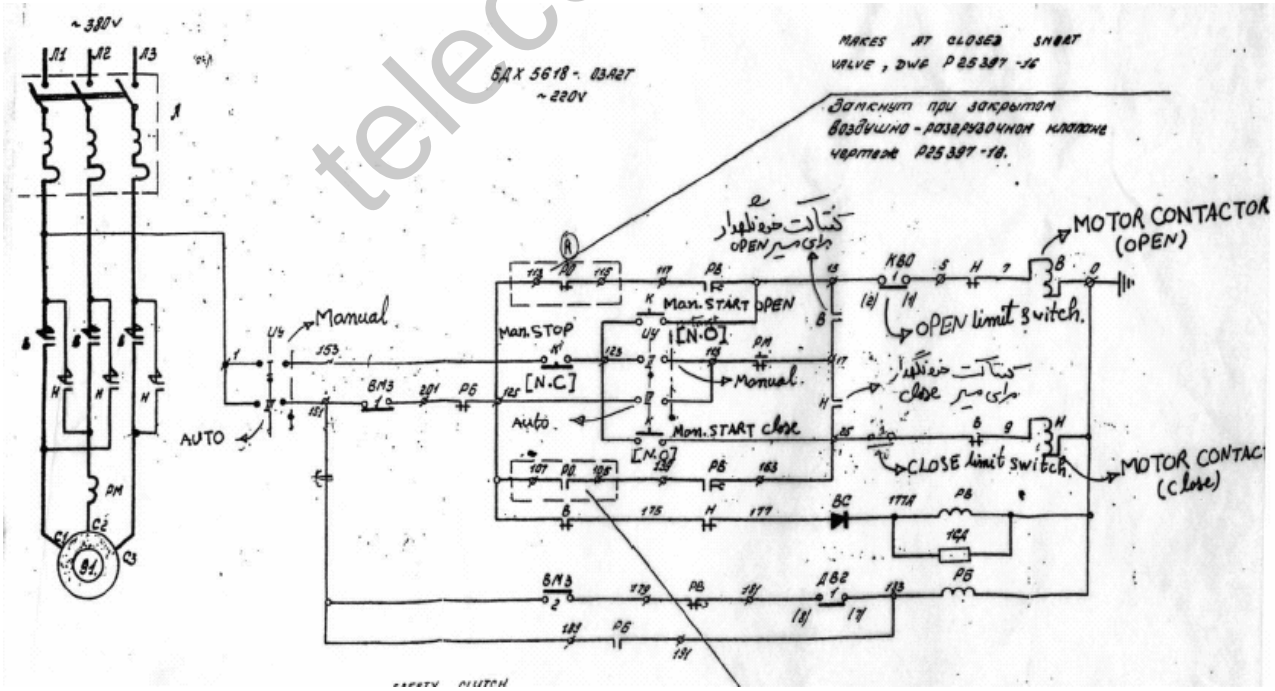
شرکت آمریکایی GE Fanuc سازنده PLC های سری GE Series 90

۳-۱) کنترل رله- کنتاکتوری:

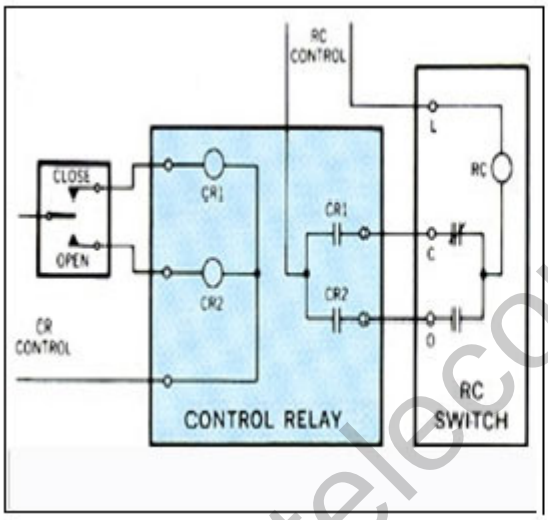
تا قبل از به کار گیری تکنولوژی **PLC** ها سیستم های کنترل با استفاده از ترکیبات رله و کنتاکتوری پیاده سازی می شدند (مدل **Hard-Wired**) .

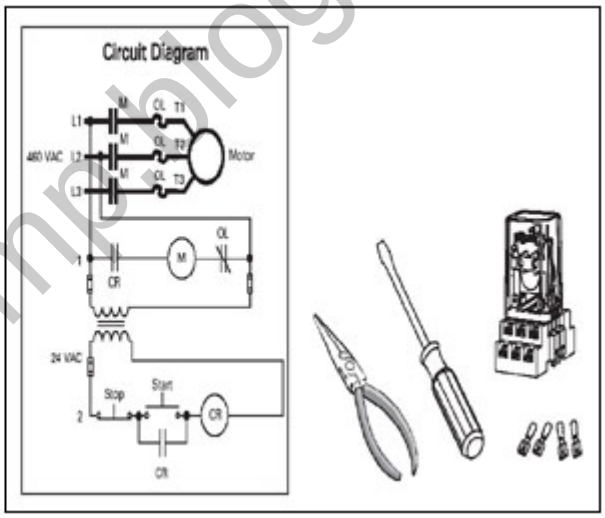


در این مدل ابتدا می بایست نقشه های کلیه مدارات کنترلی طراحی شود.



سپس رله کنتاکتورها و سایر قطعات الکتریکی مورد نیاز انتخاب و نصب شوند و سپس ارتباطات لازم با سیم بین قطعات انجام شود. این روش در مقایسه با استفاده از **PLC** دارای معایب زیادی می باشد که مهمترین آن عدم انعطاف پذیری نسبت به اصلاحات است . یعنی در صورت نیاز به تغییر در منطق کارکرد ، باز و بسته کردن سیم ها و جابجایی قطعات الکتریکی اجتناب ناپذیر است . در صورتی که این عمل در **PLC** تنها با تغییر نرم افزاری در کد برنامه انجام می شود.





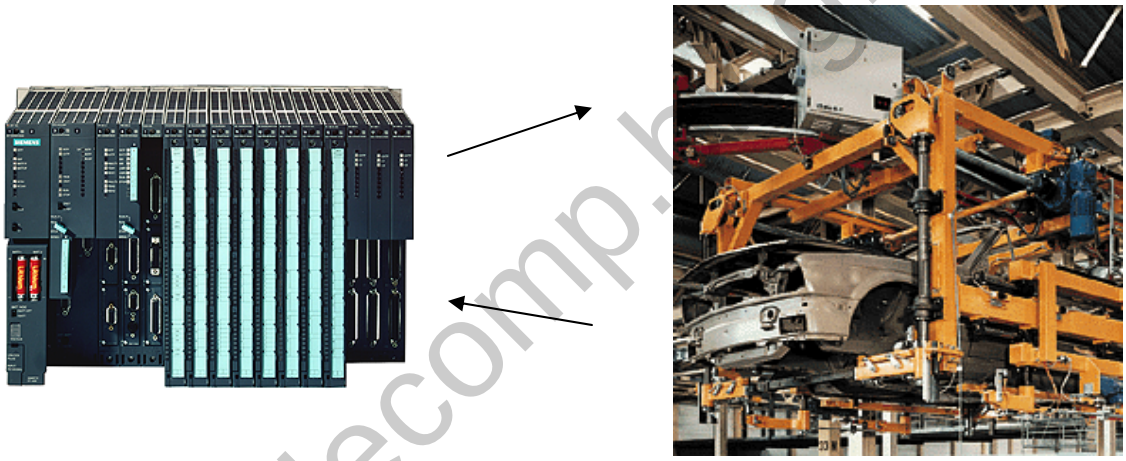
۱-۴) مزایای PLC نسبت به سیستم رله- کنتاکتوری :

- تغییر منطق کارکرد **PLC** بسیار ساده تر و سریعتر (به شکل نرم افزاری) انجام می شود.
- **PLC**ها جای بسیار کمتری اشغال می کند.
- اکثر **PLC** ها دارای واحدهای عیبیابی **(Diagnostic)** هستند و در صورت توقف عملیات علت خطا یا اشکال در برنامه را نشان می دهند.
- برای تکثیر سیستم کنترل مبنی بر **PLC** کافی است تنها برنامه آن کپی شود.
- قابلیت عمر بالاتری دارند (عاری از اشکالات مکانیکی رایج در رله – کنتاکتورها هستند) .

۲- معرفی PLC و اجزای آن

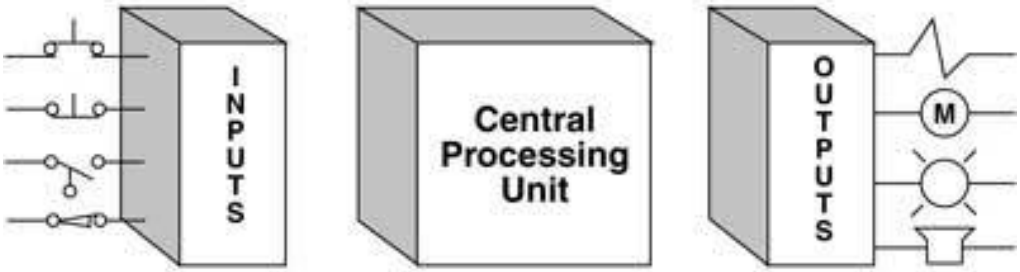
۲-۱) تعریف PLC

PLC ها یک نوع کنترل کننده منطقی (Logical) از خا نواده کامپیوترها هستند که برای کاربردهای صنعتی طراحی و ساخته شده اند. از PLC ها برای انجـام خودکـار عملیـات در کارخا نجات تولیدی استفاده می شود .



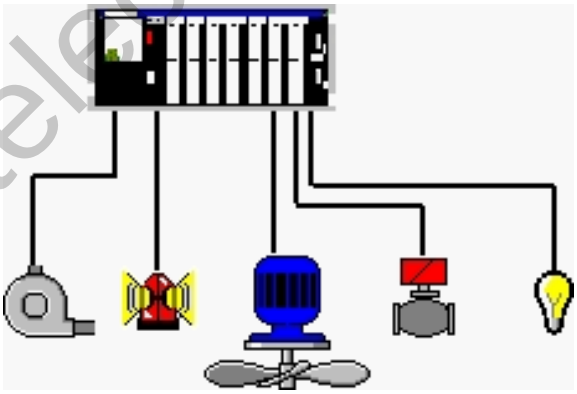
شکل ۱- ۲ : نمونه ای از کاربرد PLC در صنعت

کار عمده و اصلی یک PLC ، گرفتن اطلاعات از واحد تحت کنترل به عنوان ورودی سیستم ، تصمیم گیری با توجه به مقادیر ورودی ها و برنامه ایی که در آن تعبیه شده و درنهایت ایجاد خروجی ها و ارسال آنها به سخت افزارهای میانی جهت هدایت ماشینهای تحت کنترل می باشد.



شکل ۲- ۲ : واحدهای اصلی PLC

برنامه درون PLC ، مجموعه ایی از دستور العمل هایی است که کاربر آنها را متناسب با نحوه عملکرد مکانیسم و فرایند موجود ایجاد کرده و در درون حافظه PLC قرار می دهد. وقتی برنامه اجرا می شود، PLC سیستم را بر اساس مشخصات فرایند مورد نظر، راه می برد.

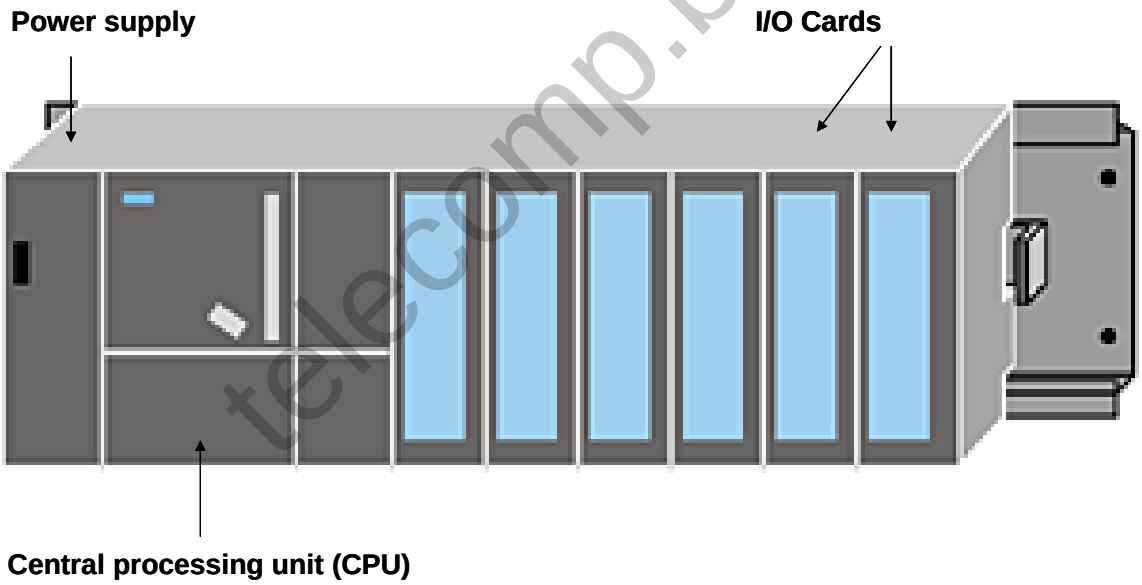


شکل ۳- ۲ : شماتیک PLC همراه با اتصالات آن به تجهیزات درون سایت (ورودی ها و خروجی ها)

۲-۲) اجزاء تشکیل دهنده یک PLC

برای انجام عملیات کنترلی ، تمامی **PLC** ها دارای مجموعه ای از سخت افزارهای گوناگون هستند که هر یک وظیفه مشخصی را انجام می دهند. در این قسمت به معرفی این المان ها پرداخته می شود. این المان ها که به شرح زیر می باشند، در تمامی **PLC** ها وجود دارند.

- منبع تغذیه
- **CPU**
- کارت های ورودی / خروجی



شکل ۴- ۲ : نمایش شماتیک سخت افزار PLC

۱-۲-۲) منبع تغذیه PS (Power Supply)

منبع تغذیه ، برای برق رسانی به تمام قطعات سیستم به کار می رود و مقادیر معمول آن ۲۴ ولت مستقیم ، ۱۱۰ ولت متناوب و ۲۳۰ ولت متناوب است. بنابر این ، کاربر باید پیش از خرید PLC ، امکان تغذیه PLC را بررسی کند.

شکل ۵- ۲ : نمونه ای از یک ماژول منبع تغذیه (PS)

منابع تغذیه مورد استفاده در **PLC** معمولا از نوع **Switching Power Supply** با خروجی بسیار دقیق هستند. قبل از خرید منبع تغذیه بهتر است مصرف جریان کلیه کارت های **PLC** محاسبه و سپس **Power Supply** با جریان دهی بالاتری تهیه شود.

۲-۲-۲) واحدپردازش مرکزی CPU (Central Processing Unit)

CPU ، مغز PLC است ، یعنی دستگاهی که تمام عملکرد ها را با ترتیبی درست کنترل می کند . اما باید توجه داشت که CPU هوشمند نیست وخودش فکر نمی کند بلکه فقط از مجموعه دستور العمل هایی که در حافظه آن قـرار می گیرد ، تبـعیـت می کند . CPU ها بر اساس مقدار حافظه و سرعتی که دارند و همچنین تعداد ورودی و خروجی هایی که می پذیرند ، دسته بندی می شوند.

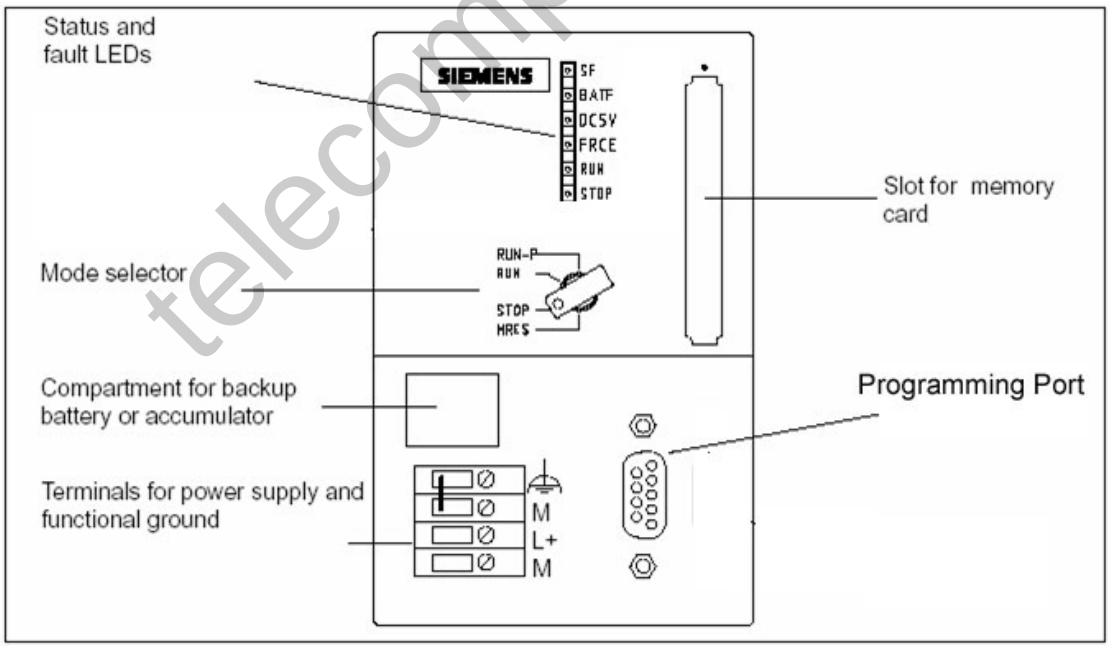
شکل ۶- ۲ : نمونه ای از یک ماژول CPU

معمولا روی بدنه **CPU** تعدادی **LED** وجود دارد که وضعیت جـاری آن را نشان می دهند. تعداد و معنی این **LED** ها در **PLC** های گوناگون متفاوت است.

یک PLC به طور کلی دارای سه وضعیت کاری زیر است:

- **Stop** : برنامه درون **PLC** اجرا نمی شود و خروجی ها غیر فعال هستند.
- **Run** : برنامه دورن **PLC** اجرا می شود و خروجی ها فعال هستند.
- **Fault** : به دلیل بروز اشکالی ، اجرای برنامه متوقف شده و **PLC** به وضعیت **Stop** رفته است.

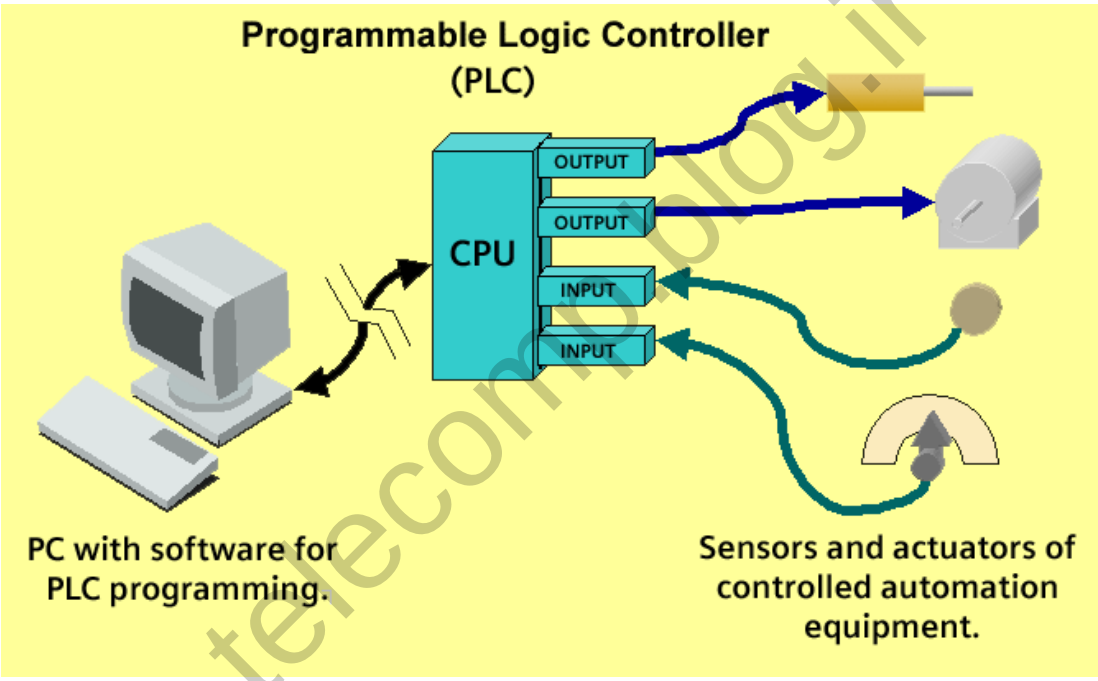
همچنین یک Selector نیز وجود دارد که به وسیله آن می توان وضعیت کاری **PLC** را انتخاب کرد. یعنی به حالت **Stop** یا **Run** تغییر وضعیت داد و یا در صورت لزوم حافظه **PLC** را **reset** کرد.



شکل ۷- ۲ : جزئیات یک ماژول CPU

۳-۲-۲) واحدهای ورودی و خروجی (I/O Units)

سیگنالها و پیغامهایی که در واحد تحت کنترل هستند توسط کارت های ورودی و خروجی با **CPU** در ارتباط هستند. کارت های ورودی و خروجی بر اساس مدارات درون خود با سیگنالها و بوسیله باس داخلی با **CPU** در ارتباط می باشند.

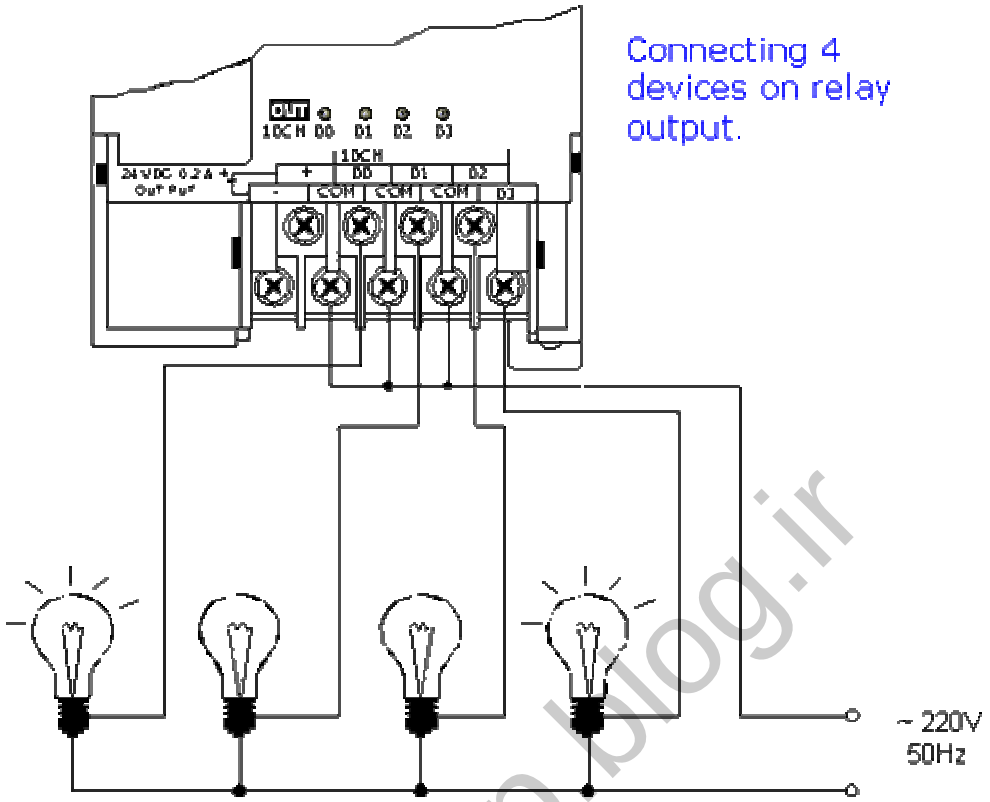


شکل ۸- ۲ : شماتیک تجهیزات ورودی و خروجی همراه با اتصالات آن به PLC

وظیفه اصلی کارت ورودی ، دریافت سیگنالهای مختلف از سایت و تغییر شکل آنها به صورتی است که قابل استفاده برای **CPU** بر اساس این وقایع و برنامه ایی که از پیش در آن وجود دارد ، تصمیم گیری کرده و نتایج خروجی را تولید می کند.

وظیفه خروجی ها مهیا کردن این نتایـج بصورت قا بل ارائـه به سایت می باشد.

شکل ۹- ۲ : نمونه ای از یک ماژول I/O

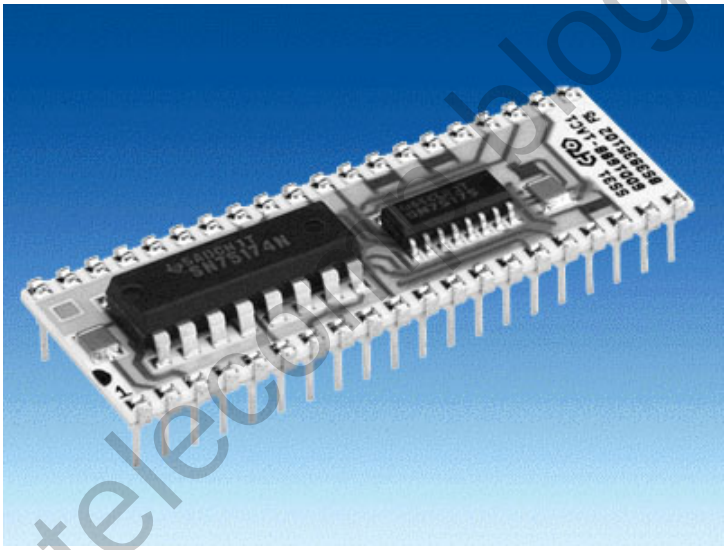


شکل ۱۰- ۲ : نمایش اتصالات یک ماژول خروجی به لامپها

۴-۲-۲) واحد حافظه (Memory Unit)

جهت نگهداری دستورالعملها ، برنامه و تصاویر تعریف شده ، PLC نیاز به حافظه دارد. انواع مختلفی از حافظه در PLC بکار می روند :

- ROM
- RAM
- EPROM



شکل ۱۱- ۲ : : نمونه ای از درون یک ماژول حافظه

حافظه ها سایزها و ظرفیتهای تفـاوتی دارنـد ، مثلا **۴K** ، **۸K**، **M** ۲ ، .... در ذیـل به بررسی چنـد نوع مختلف از حـافظه ها می پردازیم :

الف) حافظه فقط خواندنی ROM

ROM را سازنده برنامه نویسی می کند و حافظه دائمی و غیر فراری است که برنامه و داده های سیستم عامل در آن ذخیره می شود.

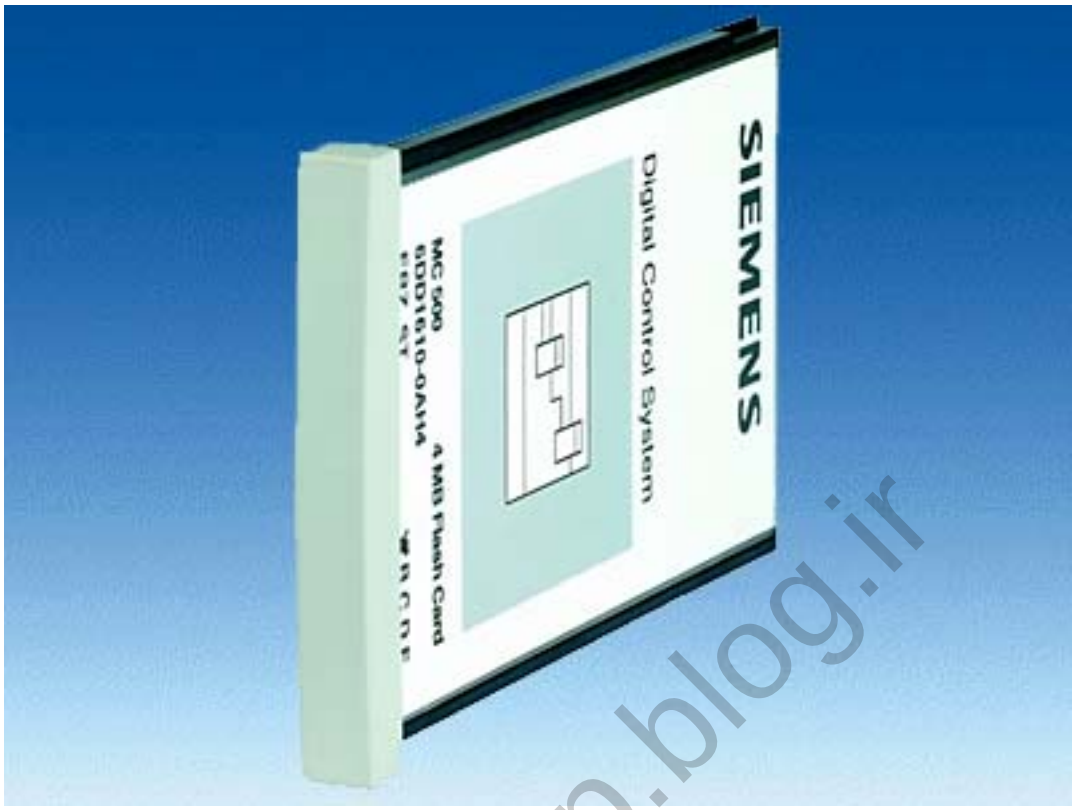
ب) حافظه با دستیابی تصادفی RAM

RAM ها ، حافظه هایی خواندنی و نوشتنی و فرار هستند که کاربر می تواند برنامه را درون آنها بنویسید و از روی آنها اجرا کند اما اطلاعات درون آنها با قطع شدن تغذیه شان از بین رفته و پاک می شود.

داده های درون این حافظه انعطاف پذیر را می توان حین کار با PLC نیز اصلاح ذخیره نمود.

ج) حافظه فقط خواندنی ، برنامه پذیر قابل پاک شدن با برق EEPROM :

EEPROM ها مشابه ROM است . فرمت EEPROM را می توان با پالسهای الکتریکی پاک کـرد ، امـا نوشتن داده در آن ، در مقـایسه با RAM ، وقت بیشتری مـی گیـرد . معـمولا از EEPROM به عنوان حافظه پشتیبان استفاده می کنند.



شکل ۱۲- ۲ : : نمونه ای از یک ماژول حافظه

۲-۳) انواع PLC

PLC ها از لحاظ سخت افزاری به دو کونه کلی در دسترس می باشند **:**

- یکپارچه
- ماژولار

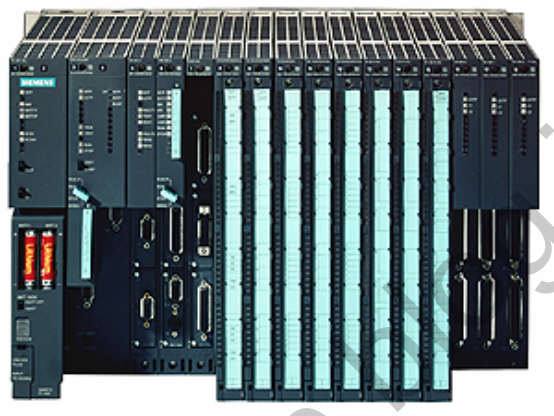
۱-۳-۲) PLC یکپارچه

PLC یکپارچه خود کفاست . یعنی سخت افزار آن ، منبع تغذیه ، **CPU** و تعداد محدودی ورودی وخروجی را بصورت یک بسته یکپارچه شامل می شود. این نوع **PLC** مختصرتر ، ساده تر و ارزانتر و دارای عملکردی محدودتر از گونه دیگر می باشد و از آن برای کنترل کردن نقاله های کوچک ، ماشین های تراش، پرس های ضربه ایی، سیستم های کنترل هیدرولیکی و بادی ، .... می توان استفاده نمود.

شکل ۱۳- ۲ : : چند نمونه از PLC های یکپارچه

۲-۳-۲) PLC ماژولار :

PLC ماژولار از کنـار هم قرار گرفتن ماژولهای مختـلف ، مانند : ماژول منبع تغذیه، ماژول CPU، ماژولهای ورودی، ماژولهای خروجی، کارت های شبکه ساخته می شوند.



شکل ۱۴- ۲ : : نمونه ای از PLC های ماژولار

به علت اینکه در طراحی سیستم کنترل بر اساس این PLC می توان انواع مختلف و تعداد متفاوتی از ماژولها را کنار هم قرار داد ، این سیستم قابلیت انعطاف بیشتری دارد و می توان آن را برای کاربرد های خاص ، مانند سیستم ها کنترل خود کار ماشین ها و کنترل فرایند ، طراحی نمود .

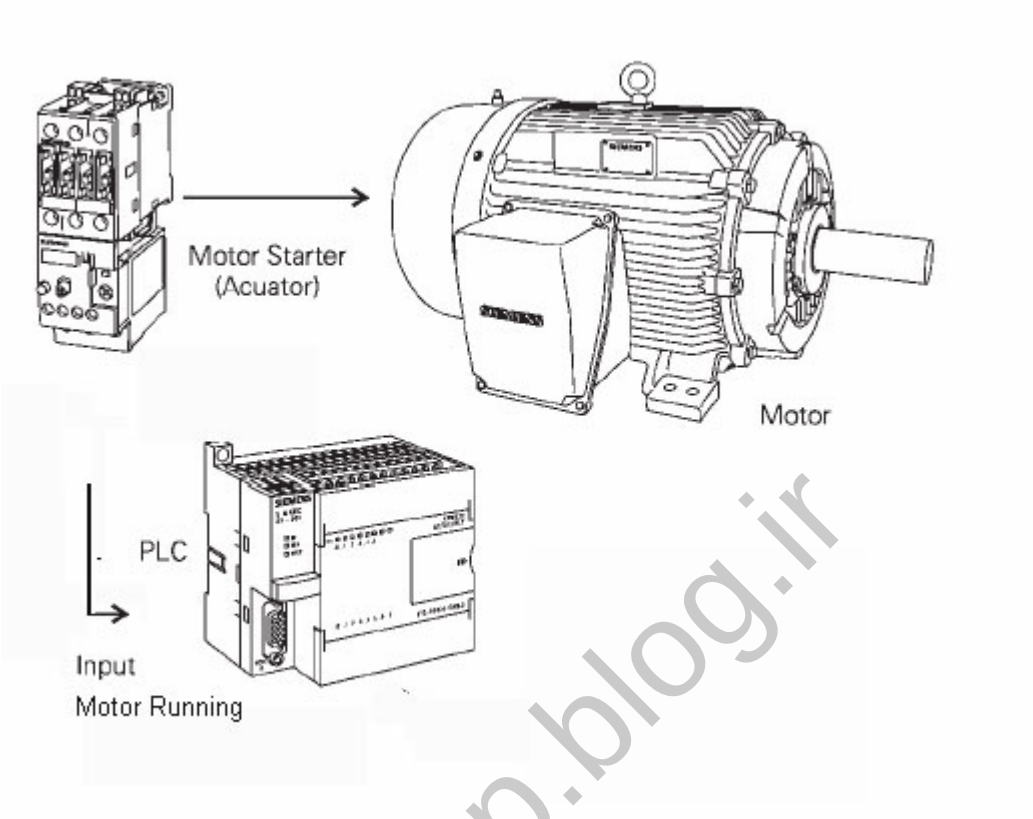
همچنین **PLC** های ماژولار تعداد ورودی ها و خروجی های بیشتری دارند و قابل توسعه دادن **(Expansion)** هستند. در نتیجه می توان آنها را به راحتی برای کار در سیستم های بزرگتر ، با اعمال تغییراتی، استفاده نمود.

۳- ورودی ها و خروجی های PLC

۳-۱) تعریف ورودی ها و خروجی هایPLC :

برای کنترل کردن سیستمی توسط PLC لازم است که اتفاقاتی که در سیستم روی میدهد به اطلاع PLC برسد تا PLC با بررسی آنها و اجـرای منطقـی کـه بـرایش تعریـف شـده ، فرامین لازم جهت کنترل سیستم را صادر کند .

ماشین ها وابزار هایی که در سیستم وجود دارند ، وضعیت های خود و یا اتفاقاتی را که رخ میدهد توسط سیگنالهای الکتریکی بیان می کنند ، کـه ایـن سـیگنالها بنـا بـه نـوع آن ماشین و یا اتفاق ، ماهیت های متفاوتی دارند . مثلا موتوری که در حالت کـار کـردن اسـت یک ولتاژ ۲۴ ولتی را طریق یکی از کنتاکت رله هایش منتقل مـی کنـد و چنانچـه متوقـف باشد، دیگری ولتاژی را منتقل نمی کند . بنابراین دریافت ولتـاژ ۲۴ ولـت از آن کنتاکـت را می توان نشانه کارکردن موتور و عدم دریافت آن را نشانه متوقف بودن آن موتور دانست .

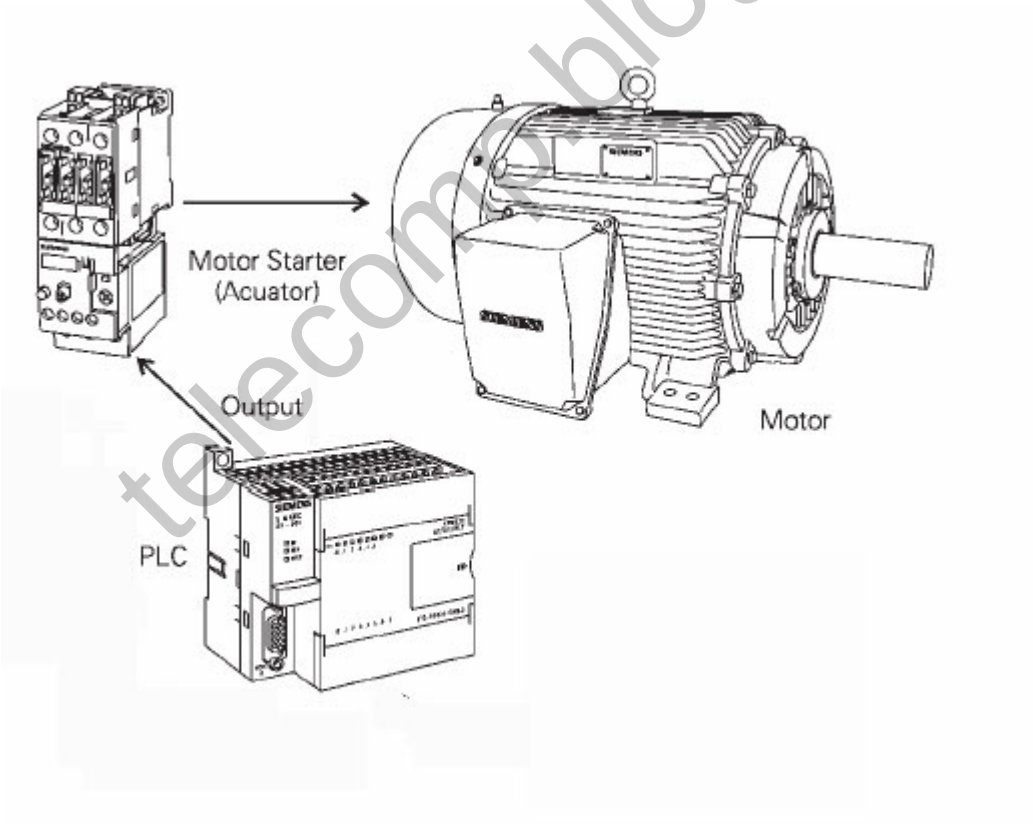


شکل ۱- **۳** : : نمونه ای از ارتباطات ماشینها و ابزارها با ورودی های PLC

برای تصمیم گیری درباره وضعیت این موتور می بایست این سیگنال را که یکی از سیگنالهای موتور است به **PLC** اعمال نمود . برای اینکار احتیاج به یک کانال ورودی می باشد . این ورودی باید قابلیت دریافت ۲۴ و صفر ولت واداشته باشد و از آنجائیکه PLC از مدارات منطقی تشکیل شده اند و با سیگنالهای دیجیتال (منطق صفر و یک) سرو کار دارند، سخت افزار ورودی پس از دریافت این ولتاژ می بایست آن را به منطق صفر و یا یک تبدیل کند .

پس بطورخلاصه می توان گفت کار کردن موتور برای **PLC** ، با یک بودن ورودی مربوطه و متوقف بودن آن با صفر بودن آن ورودی بیان می شود.

اگر ولتاژی روی رله موتور اعمال شود ، موتور را وادار به کار کردن می کند و قطع این ولتاژ باعث توقف کار موتور خواهد شد. بنابراین چنانچه بعد از بررسی وضعیت ها و اجرای برنامه ، مقرر شد تا موتور از طریق PLC فرمان حرکت را بگیرد منطق یک ایجاد شده توسط برنامه ، به یک کانال خروجی که آن را به ولتاژ مورد نظر تبدیل می کند ، داده می شود. اگر آن کانال خروجی را به رله مربوط به موتور متصل کنیم ، موتور فرمان شروع کار کردن را دریافت خواهد کرد، در غیر اینصورت (اعمال منطق صفر به خروجی) موتور فرمان توقف را می گیرد.



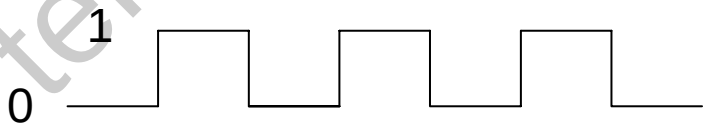
شکل ۲- **۳** : : نمونه ای از ارتباطات ماشینها و ابزارها با خروجی های PLC

نتیجا ، عملکرد خروجی ، تبدیل مقادیر ایجاد شده توسط برنامه به سیگنال الکتریکی مورد نیاز سیستم جهت ارسال فرمان به موتور می باشد. با مثال فوق با نحوه عملکرد ورودی ها ، خروجی ها و نقش سیگنالها آشنا شدید. البته در این مثال سیگنالها همگی دیجیتال بودند. سیگنالهای ورودی و یا خروجی می توانند به صورت دیجیتال و یا آنالوگ باشند.

۲-۳) تعریف سیگنال دیجیتال:

سیگنالی است که تنها دو وضعیت داشته باشد. مثلا اگر ولتاژ باشد تنها دو مقدار ۰ و ۲۴ ولت را شامل شود. که در این صورت یکی از آنها نماینده منطق صفر و دیگری نماینده منطق یک خواهد بود.

تمام وضعیت های دو حالته با سیگنال دیجیتال مشخص می شوند. مثل خاموش / روشن بودن یک موتور یا باز / بسته بودن یک شیر برقی.



شکل ۳- **۳** : : نمونه ای از یک سیگنال دیجیتال

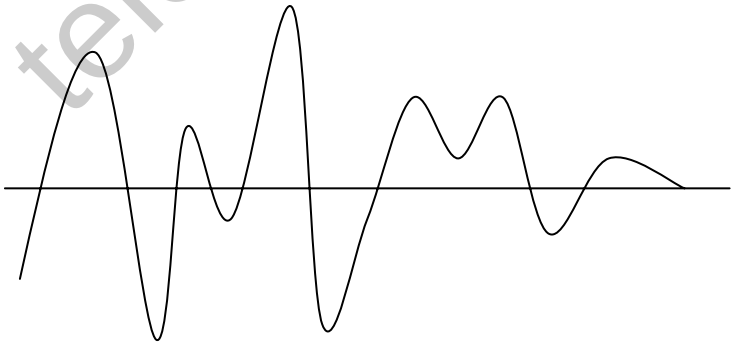
۳-۳) تعریف سیگنال آنالوگ:

سیگنالی است که چندین وضعیت را در یک رنج تغییر سیگنال شامل می شود. مثلا سیگنال آنالوگ در رنج ۴ تا ۲۰ میلی آمپر .

سیگنال های آنالوگ معمول به شرح زیر هستند:

- **4-20 mA ,0-20 mA**
- **1-5 Volt ,0-10 Volt**
- **0-500 mV Thermocouple**
- **RTD**

سیگنال دیجیتال عمدتا در وضعیتهای دو گانه مانند **…., Stop – Start , OFF-ON** کاربرد دارد، در حالیکه سیگنالهای آنالوگ حاوی مقداری هستند مثلا دمای از ۰ تا ۵۰ درجه ، فشار ۰ تا ۱۰۰ بار ، ....



شکل ۴- **۳** : : نمونه ای از یک سیگنال آنالوگ

۳-۴) تعریف ورودی و خروجی دیجیتال:

۳-۴-۱) ورودی دیجیتال :

ورودی های دیجیتال ، سیگنال های دیجیتالی هستند کـه از محـیط بیـرون توسـط سـخت افزاری یه نام کارت ورودی دیجیتال در PLC دریافت می شوند.

سیگنال های دریافتی از المان های زیر ورودی دیجیتال محسوب می شوند:

- **کنتاکت های رله ها**
- **Limit Switch** ها
- **Push Button** ها
- **Proximity Switch** ها
- **Process Switch** ها

الف) کنتاکت های رله ها :

برای اطلاع از وضعیت کاری المان هایی نظیر پمپ ها، موتورها و شیرهای برقی

شکل ۵- ۳ : : نمونه ای از یک کنتاکت رله

ب) Limit Switch ها :

برای آگاهی از رسیدن یک وسیله مکانیکی متحرک به ابتدا / انتهای مسیر حرکت خود

شکل ۶- ۳ : : نمونه هایی از Limit Switch ها

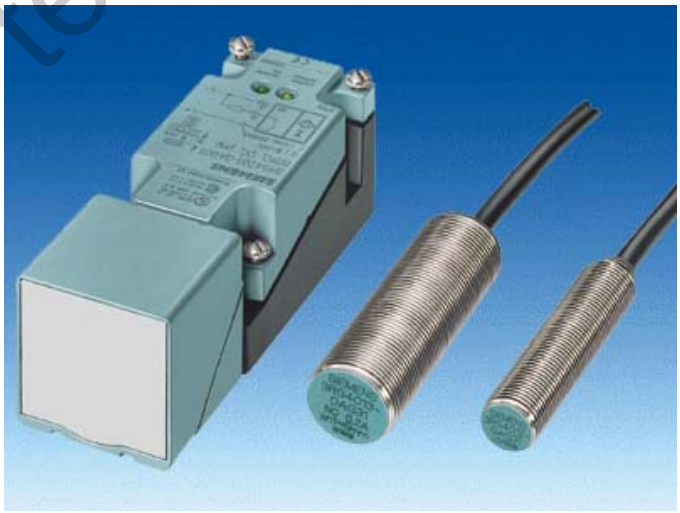
ج) Push Button ها :

برای ارسال فرمان start / stop یا on / off

شکل۷- ۳ : : نمونه هایی از Push Button ها

د) Proximity Switch ها :

برای حس کردن نزدیکی یک جسم فلزی متحرک



شکل۸- ۳ : : نمونه هایی از Proximity Switch ها

ه) Process Switch ها :

برای اطلاع از رسیدن یک کمیت فیزیکی به حد بالا یا پایین خود. نظیر دما، فشار، سطح و فلو

شکل۹- **۳** : : نمونه هایی از Process Switch ها

۲-۴-۳) خروجی دیجیتال :

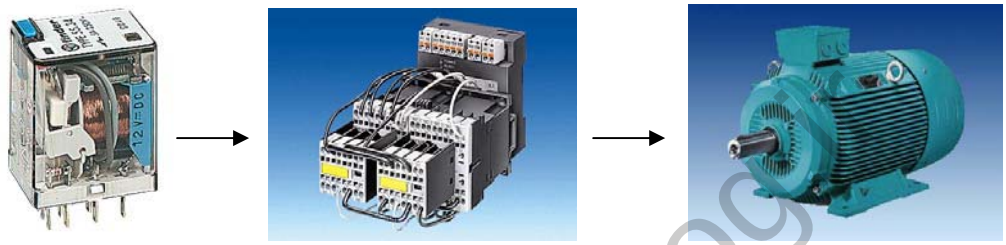
خروجی های دیجیتال سیگنال های دیجیتالی هستند که از PLC توسط سخت افزاری به نام کارت خروجی دیجیتال به محیط بیرون منتقل می شوند.

موارد زیر نمونه هایی از خروجی دیجیتال هستند:

- تحریک کردن بوبین یک رله
- روشن / خاموش کردن چراغ سیگنال ها

الف) تحریک کردن بوبین یک رله :

برای بستن یک کنتاکت و فرمان دادن به یک کنتاکتور جهت خاموش / روشن کردن پمپ ها و موتورها



شکل ۱۰- ۳ : : نمونه ای از تحریک کردن بوبین یک رله

ب) روشن / خاموش کردن چراغ سیگنال ها :

برای نمایش وضعیت on / off پمپ ها و موتورها یا باز و بسته بودن شیرهای برقی و نیز آلارم دادن (از طریق خاموش / روشن شدن)

شکل ۱۱- ۳ : : نمونه ای از چراغهای سیگنال

مثلا وضعیت روشن یا خاموش بودن یک موتور ، خروجی های دیجیتال ، فرامین دو حالته را از **PLC** به سایت منتقل می کنند. از آنجائیکه دو وضعیت را می توان توسط یک تک بیت نمایش داد. آدرس دهی ورودی ها و خروجی های دیجیتال بصورت یک بیتی می باشد.

۳-۵) تعریف ورودی و خروجی آنالوگ :

۳-۵-۱) ورودی آنالوگ :

ورودی های آنالوگ ، سیگنال های آنالوگی هستند که از محیط بیرون توسط سخت افزاری یه نام کارت ورودی آنالوگ در **PLC** دریافت می شوند. در کارت ورودی آنالوگ ، عمل تبدیل آنالوگ به دیجیتال صورت می گیرد

سیگنال های دریافتی از المان های زیر ورودی آنالوگ محسوب می شوند:

- **Temperature Instrument**
- **Pressure Instrument**
- **Level Instrument**
- **Flow Instrument**
- **Load Cell**

توجه داشته باشید که برای هر نوع سیگنال آنالوگ، کارت مخصوص به آن باید استفاده شود. مثلا برای **TC** ها کارتی به کار می رود که ممکن است برای مقادیر ولتاژی قابل استفاده نباشد.

الف) Temperature Instrument :

تجهیز اندازه گیری دما

شکل ۱۲- **۳** : : نمونه های از Temperature Instrument

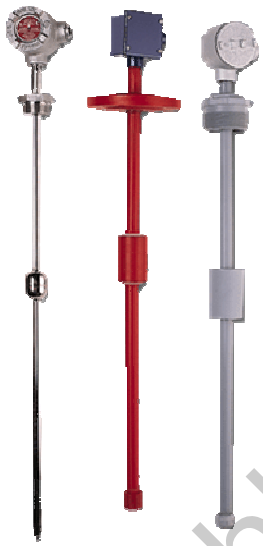
ب) Pressure Instrument :

تجهیز اندازه گیری فشار

شکل ۱۳ - **۳** : : نمونه های از Pressure Instrument

ج) Level Instrument :

تجهیز اندازه گیری سطح



شکل ۱۴ - ۳ : : نمونه های از Level Instrument

د) Flow Instrument :

تجهیز اندازه گیری فلو

شکل ۱۵ - ۳ : : نمونه های از Flow Instrument

ه) Load Cell :

تجهیز اندازه گیری وزن

شکل ۱۶ - ۳ : : نمونه ای از Load Cell

مثلا برای استفاده از درجه حرارت یک مخزن سیگنال آنالوگی که در رنج ۰ تا ۱۰ ولت است و متناظرا دمای ۰ تا ۷۰ درجه را نشان می دهد از سنسور دمای مخزن بصورت ورودی آنالوگ به **PLC** داده می شود. و **PLC** جهت استفاده از آن در برنامه نویسی ، معادل عددی آن را که توسط ورودی آنالوگ ایجاد شده ، را بکار می برد.

۲-۵-۳) ورودی آنالوگ :

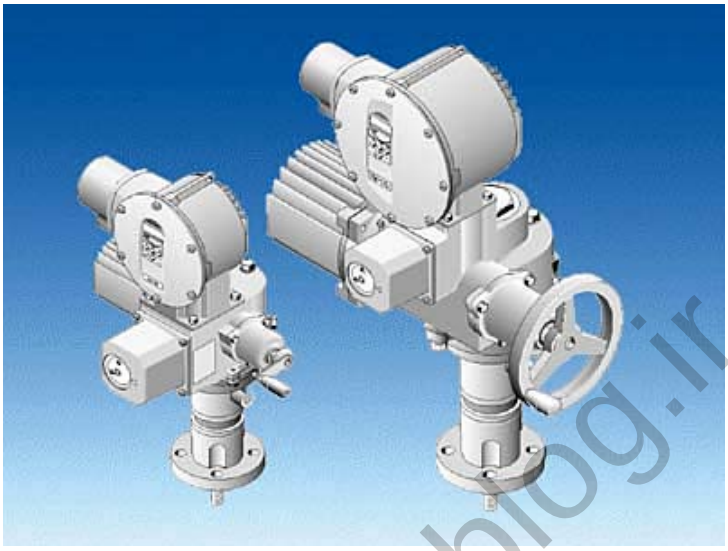
خروجی های آنالوگ سیگنال های آنالوگ ی هستند که از **PLC** توسط سخت افزاری به نام کارت خروجی آنالوگ به محیط بیرون منتقل می شوند. در کارت خروجی آنالوگ ، عمل تبدیل دیجیتال به آنالوگ صورت می گیرد.

موارد زیر نمونه هایی از خروجی آنالوگ هستند:

- **سیگنال ارسالی به کنترل والوها**
- نمایشگرها

الف) سیگنال ارسالی به کنترل والوها :

برای تنظیم آنها روی درصد باز/ بسته بودن مناسب



شکل ۱۷ - **۳** : : نمونه های از کنترل والوها

ب) نمایشگرها :

برای نشان دادن مقدار یک کمیت به صورت دیجیتالی یا عقربه ای

شکل ۱۸ - **۳** : : نمونه های از نمایشگرها

۳-۶) کارتهای ورودی و خروجی :

برای دریافت و ارسال سیگنال های آنالوگ و دیجیتال از سخت افزاری به نام کارت I/O استفاده می شود. شکل و مشخصات کارت ها در اکثر PLC ها شبیه هم است. هر سازنده بسته به نوع سیگنال ممکن است کارت های متنوعی را عرضه کرده باشد.

مثال زیر نحوه نامگذاری کارت ها را در PLC های زیمنس نشان می دهد.

- ✓ **32 DI** (کارت ورودی دیجیتال با ۳۲ کانال)
- ✓ **DO 16** (کارت خروجی دیجیتال با ۱۶ کانال)
- ✓ **AI 8** (کارت ورودی آنالوگ با ۸ کانال)
- ✓ **AO 4** (کارت خروجی آنالوگ با ۴ کانال)

شکل ۱۹ - **۳** : : نمونه های از کارتهای ورودی و خروجیPLC زیمنس

۷- ۳) تبدیلات آنالوگ ها :

هر PLC مشخصه ایی دارد که بر اساس آن تبدیل آنالوگ به دیجتیال و بر عکس را انجام می دهد ، مثلا ممکن است در PLC معادل باینری سیگنالهای آنالوگ را در یک Word (۱۶ بیت) قرار دهند. بدین ترتیب مقدار آنالوگ تبدیل شده صفر تا ۶۵۵۳۶ می تواند باشد. در بعضی PLC ها مقادیر آنالوگ به عددی بین صفر تا ۱۰۰۰۰ تبدیل می شوند.

۸- ۳) آدرس دهی ورودی خروجی ها:

به منظور مشخص شدن سیگنالهای ورودی و یا خروجی PLC به هریک از آنها آدرس منحصر به فردی اختصاص داده می شود و در کاربرد برنامه نویسی از آن آدرسها استفاده می شود.

معمولا در **PLC** ها نحوه نمایش آدرس دهی ورودی و خروجی ها دیجیتال و آنالوگ با هم متفاوتند .

مثلا در **PLC** های ساخت زیمنس سری **S7** از حرف **I** برای نمایش ورودی ها و از حرف **Q** برای نمایش خروجی ها استفاده می شود.

۱-۸-۳) مثال از نحوه آدرس دهی کارت های I/O:

I4.1 : آدرس یکی از کانالهای ورودی دیجیتال

Q5.3 : آدرس یکی از کانالهای خروجی دیجیتال

IW255 : آدرس یکی از کانالهای ورودی آنالوگ

QW267 : آدرس یکی از کانالهای خروجی آنالوگ

۴- برنامه نویسی PLC

برای اینکه سیستمی را بتوان توسط **PLC** کنترل نمود ، می بایست کارهای کنترلی مورد نظر آن را بصورت دستورات برنامه نویسی در **CPU** و حافظه آن ایجاد کنیم تا **PLC** با اجرای کلیه آن دستور العمل ها ، با ترتیب صحیح وظیفه خود را به انجام برساند.

برای نوشتن دستور العمل ها در **PLC** می بایست با روش برنامه نویسی ، نحوه استفاده از ورودی ها ، تبدیل آنها ، ساختمان اصلی برنامه ، روش اجرای آن ، آشنا بود.

۴-۱) سیستم عدد نویسی:

از آنجائیکه PLC ساختاری مانند کامپیوتر دارد ، اطلاعات را بصورت وضعیت روشن و خاموش (یک و صفر) ، متناسب با رقم های باینری (بیت ها) نگهداری می کند. در بعضی موارد اطلاعات ذخیره شده در بیت ها بصورت تکی و گاهی بصورت تعدادی از بیت ها برای نشان دادن یک مقدار عددی بکار می روند . سیستم های عدد نویسی متعددی وجود دارند. در تمامی سیستم های عدد نویسی سه شاخص ارقام ، پایه و اوزان وجود دارد . بنابراین در هر سیستم اعداد با توجه به روشهای آن به فرمی خاص نشان و داده می شوند.

۱-۱-۴) سیستم اعشاری (ده دهی) :

سیستم اعشاری ، سیستم عدد نویسی ایی است که روزانه و بطور معمول ما از آن استفاده میکنیم ، شاخص های نام برده شده ، در عدد نویسی اعشاری عبارتند از :

- ارقام : ۰،۱،۲،۳،۴،۵،۶،۷،۸،۹
- پایه : ۱۰
- اوزان : توانهای ۱۰ (۱،۱۰،۱۰۰،۱۰۰۰ ،...)

همانطور که مشاهده می کنید ، در این سیستم اعداد با ترکیبی از این ۱۰ رقم نوشته می شوند و محل قرار گیری آنها ، وزن آنها (بر اساس توانی از پایه) می باشد.

❖ **مثال** : در عدد ۵۸۶ از ارقام ۶ و ۸ و۵ استفاده شده که در آن ۶ دارای وزن ۱ ($۱۰^{۰}$) ، ۸ دارای وزن ۱۰ ($۱۰^{۱}$) و ۵ دارای ۱۰۰ ($۱۰^{۲}$) می باشند .

$۵۸۶ = ۵ * (۱۰)^{۲} + ۸ * (۱۰)^{۱} + ۶ * (۱۰)^{۰}$

بنابراین در این سیستم وزن هر رقم از سمت راست توانی از پایه می باشد و رقم سمت چپ بیشترین وزن را خواهد داشت .

۲-۱-۴) سیستم باینری (دو دویی) :

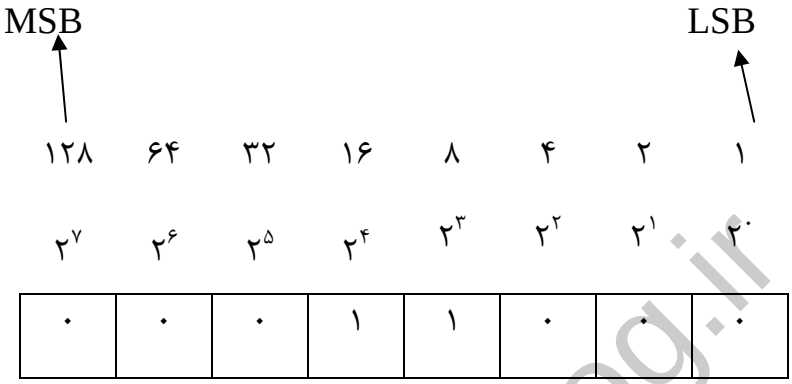
این سیستم عدد نویسی در کنترلر کنترهای قابل برنامه نویسی استفاده می شود و شاخص های آن عبارتند از :

- ارقام : ۰ و ۱
- پایه : ۲
- اوزان : توانهای ۲ (۱،۲،۴،۸،۱۶.....)

در عدد نویسی باینری ارقارم صفر و یک در ستونهایی قرار می گیرند که هریک از آنها دارای وزنی بر اساس توانی از ۲ می باشد. اولین ستون سمت راست دارای وزن ۱ ($۲^{۰}$) است و به آن " کم ارزش ترین بیت " (**LSB**) گفته می شود.

وزن ستونها به سمت چپ دو برابر ستون سمت راستی خود است . مثلا وزن ستون دوم از سمت راست ۲ ($۲^{۱}$) می باشد. به آخرین ستون سمت چپ که دارای بیشترین وزن است " پر ارزش ترین بیت " **(MSB)** گفته می شود .

اگر عددی که به فرم باینری نوشته شده است دارای ۸ بیت باشد تشکیل یک بایت را می دهد ، بنابراین



۱-۲-۱-۴) تبدیل عدد در سیستم باینری به صورت اعشاری :

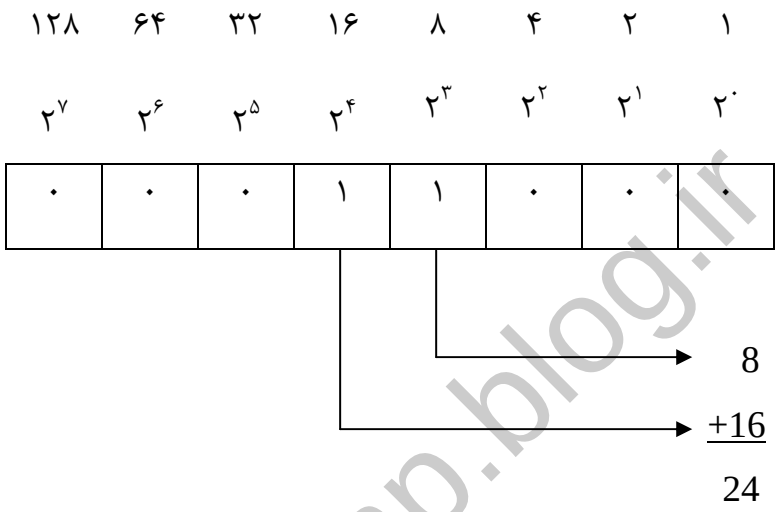
انجام مراحل زیر جهت بدست آوردن معادل اعشاری یک عدد باینری ، می باشد.

۱) از **LSB** به سمت **MSB** ارقام یک را بیابید.

۲) معادل اعشاری آن رقم یک را با توجه به ستونی که در آن وقع است بنویسید.

۳) اعداد بدست آمده را با هم جمع کنید.

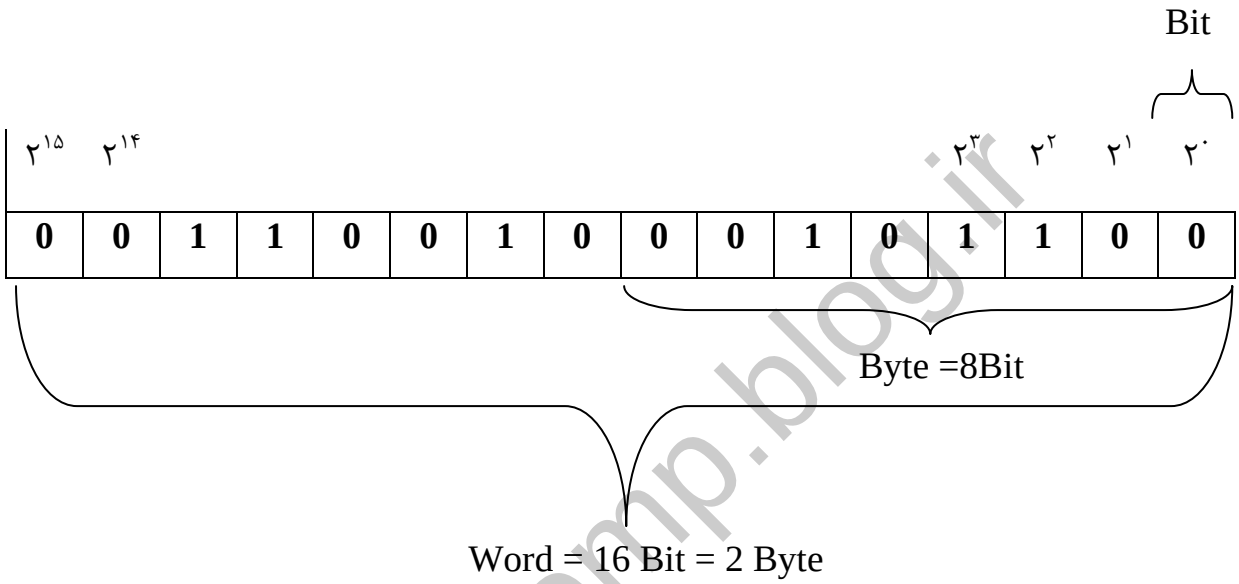
❖ مثال : در این عدد باینری ، چهارمین و پنجمین ستون از سمت راست دارای رقم یک می باشد. معادل اعشاری ستوم چهارم ۸ و ستون پنجم ۱۶ می باشد. جمع کردن این دو عدد ، معادل اعشاری عدد باینری را نشان می دهد.



$(00011000)_2 = 24$

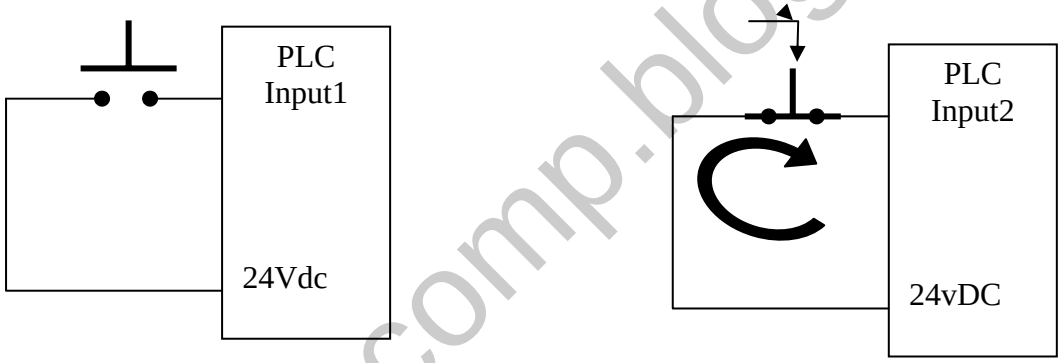
۲-۲-۱-۴) تعریف بیت(Bit) ، بایت(Byte) ،کلمه (word) :

کوچکترین قسمت از اطلاعات باینری را بیت می گویند. ۸ بیت یک بایت و ۱۶ بیت (دو بایت) یک کلمه را می سازند .



۳-۲-۱-۴) منطق صفر و یک:

کنترل کننده برنامه پذیر تنها سیگنالهای روشن و خاموش (بودن و نبودن سیگنال) را می توانند بفهمند و استفاده کنند . در سیستم عدد نویسی باینری هم که تنها دو رقم وجود دارد . عدد ۱ نشان دهنده آمدن سیگنال و یا روشن بودن سوئیچ و عدد صفر نشانده دهنده نیامدن سیگنال و یا خاموش بودن سوئیچ است .



Switch OFF
Logic 0

Switch ON
Logic 1

۳-۱-۴) سیستم BCD (Binery Coded Decimal) :

در این سیستم هریک از ارقام در سیستم اعشاری را بصورت باینری در یک کد چهار بیتی تبدیل می کند . این سیستم اغلب در دستگاههای ورودی و خروجی استفاده می شود. اعداد باینری به گروههای چهار بیتی تقسیم می شوند که هر گروه نماینده یکی از ارقارم عدد اعشاری می باشد.

اعداد در سیستم	
اعشاری	BCD
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

❖ مثال : عدد ۰۲۰۵ در سیستم عدد نویسی **BCD** را به صورت زیر می توان در سیستم باینری نوشت :

(0205) $_{BCD}$ = (0000 0010 0000 0101) $_{Binery}$

۴-۱-۴) سیستم مبنای ۱۶ (HEX):

سیستم عدد نویسی دیگری که در **PLC** کاربرد دارد، **HEX** است . شاخص های آن عبارتند از :

- ارقام : ۰،۱،۲،۳،۴،۵،۶،۷،۸،۹,**A,B,C,D,E,F**
- پایه : ۱۶
- اوزان توانهای ۱۶ (۱،۱۶،۲۵۶،۴۰۹۶)

در این سیستم ده رقم اول همانند ده رقم سیستم اعشاری است . ارقام باقی مانده از شش حرف اول الفبا استفاده می شود :

A=10 , B=11 , C=12

D=13 , E=14 , F= 15

سیستم **HEX** از آن جهت که اعداد با تعداد ارقام زیاد در سیستم باینری را با ارقام کمتری نمایش می دهد ، در **PLC** استفاده م یشود . هر رقم **HEX** می تواند اعداد تا ۴ بیت را نشان دهد.

۱-۴-۱-۴) تبدیل عدد در سیستم اعشاری به صورت HEX :

برای اینکار می بایست عدد رابه ۱۶ تقسیم کرده و آن را بصورت ضرایب ۱۶ بنویسید .

❖ **مثال :** عدد ۲۸ را بر ۱۶ تقسیم کنید ضریب آن یک و باقی مانده ۱۲ می شود:

بنابراین اولین رقم از راست که وزن ۱ ($۱۶^{۰}$) دارد ، ۱۲ است که معادل آن در **HEX**، **C** می شود .

دومین رقم که دارای وزن ۱۶ ($۱۶^{۱}$) است دارای ضریب یک است .

$$۲۸ = ۱ \times ۱۶ + ۱۲ = ۱ \times ۱۶^{۱} + ۱۲ \times ۱۶^{۰} = \mathbf{1C}$$

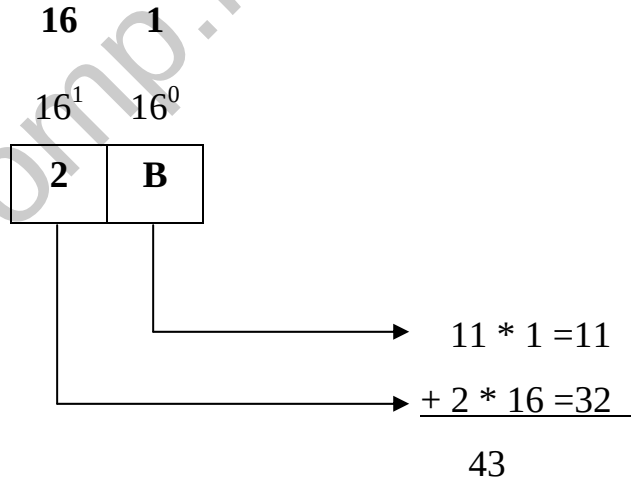
$$(28)_{10} = (1C)_{HEX}$$

2-۴-۱-۴) تبدیل عدد در سیستم **HEX** اعشاری به صورت اعشاری :

برای اینکار می بایست هر رقم را در وزنش ضرب نموده ، سپس حاصل آنها را با هم جمع کنید.

❖ **مثال** : عدد **B۲** در سیستم **HEX** بصورت زیر در اعشاری نمایش داده می شود

(**۲** B) HEX = **۲** * ۱۶^{۱} + B * ۱۶^{۰} = ۳۲+۱۱ = ۴۳



جدول زیر معادل اعداد را در چهار سیستم اعشاری ، باینری ، **BCD** و **HEX** را نشان میدهد.

Decimal	Binary	BCD	Hexadecimal
0	0	0000	0
1	1	0001	1
2	10	0010	2
3	11	0011	3
4	100	0100	4
5	101	0101	5
6	110	0110	6
7	111	0111	7
8	1000	1000	8
9	1001	1001	9
10	1010	0001 0000	A
11	1011	0001 0001	B
12	1100	0001 0010	C
13	1101	0001 0011	D
14	1110	0001 0100	E
15	1111	0001 0101	F
16	1 0000	0001 0110	10
17	1 0001	0001 0111	11
18	1 0010	0001 1000	12
19	1 0011	0001 1001	13
20	1 0100	0010 0000	14
.	.	.	.
.	.	.	.
126	111 1110	0001 0010 0110	7E
127	111 1111	0001 0010 0111	7F
128	1000 0000	0001 0010 1000	80
.	.	.	.
.	.	.	.
510	1 1111 1110	0101 0001 0000	1FE
511	1 1111 1111	0101 0001 0001	1FF
512	10 0000 0000	0101 0001 0010	200

۲-۴) اجرای متناوب و چرخشی برنامه:

مجموعه دستورالعملهایی که کاربر جهت کنترل سیستمی در **PLC** ، به کمک نرم افزار ایجاد می کند را برنامه کاربرمی گویند. سرعت اجرای برنامه و عملیات آن به عملکرد **CPU** ، سرعت آن و همچنین حجم و اندازه برنامه تعداد **I/O** و ارتباطات بستگی دارد .

روش معمول اجرای برنامه در **PLC** بدین ترتیب است که تمامی ورودی ها خوانده می شوند، برنامه کاربر پردازش می شود و خروجی ها را ایجاد می نماید ، سپس تمامی خروجی ها اعمال می شوند . بدین ترتیب به یک چرخه "پیما یش"**(Scan)** و به مدت انجام آن " زمان چرخه" **(Cycle Time)** گفته می شود.

این عمل بصورت مداوم تکرار می شود. یعنی به محض ارسال خروجی ها ، دوباره به اول سیکل رفته ورودی ها را می خواند ، برنامه را پذیرش و خروجی ها را اعمال می کند.

۳-۴) ساختار برنامه Program Structure

نحوه برنامه نویسی و ساختار برنامه نقش مهمی در انجام اصلاحات و تغییرات بعدی بازی می کند . اگر برنامه بصورتی طراحی شود که کم یا اضافه کردن عناصر کنترلی مانند پمپ ، موتور، سوئیچ ، سیگنالهای ، آنالوگ و بسیاری موارد دیگر به راحتی انجام پذیرد ، گفته می شود که برنامه مدولار است . مثلا تنها با اضافه کردن آن بخش از برنامه که مـــربـــوط به پمـــپ است (مدول برنامه پمپ) می توان دستورالعملهای پمپ جدیدی را به برنامه اضافه کرد.

طراحی برنامه به این صورت زمان بیشتری می برد . چون باید تمام حالات و ظرفیت ها در نظر گرفته شود ولی در عوض اجرای تغییرات و کنترل نحوه اجرای برنامه بسیار ساده و آسان خواهد بود.

به هر قسمت از برنامه که مربوط به کنترل بخش خاصی و یا کاربرد مشخصی هستند ، بلوک گفته می شود.

مثلا اگر موتورهای ۱ تا ۲۰ را برنامه نویسی کرده باشید برای اضافه کردن موتور جدید می توان تنها یک Block موتور دیگر به Block های قبلی اضافه نمود.

پس Block ها قسمتی از برنامه کاربردی را شامل می شوند. در هر **PLC** تعدادی از انواع مختلف Block ها وجود دارند. هر Block یی نام و کاربرد خاص خود را دارد.

برخی از انواع Block جهت ذخیره سازی اطلاعات داده ها استفاده می شوند به عنوان مثال مقادیر سیگنالهای ورودی ، خروجی **Set Point** های حلقه های کنترلی ، حد بالا و پائین سیگنالهای آنالوگ و ..... را می توان در آنها نگهداری کرد.

و یا چنانچه عملیاتی مشابه و تکراری در برنامه لازم باشد ، می توان از یکی دیگر از انواع Block ها استفاده نمود . مثلا خواندن یک کانال ورودی آنالوگ و تبدیل آن به عددی

در **Range** مورد نظر ، شامل یک سری از عملیات ریاضی است که برای هر کانال باید بطور یکسان تکرار شود. این عملیات را می توان بصورت تابعی نوشت که متغیرهای آن آدرس کانال مذکور و رنج مورد نظر باشند . بنابراین دیگر نیازی به نوشتن مجدد برنامه برای سایر کانالها نخواهد بود و تنها کافی است تابع مذکور و رنج مورد نظر باشند. بنابراین دیگر نیازی به نوشتن مجدد برنامه برای سایر کانالها نخواهد بود و تنها کافی است تابع مذکور با پارامترهای مورد نظر فراخوانی شوند.

در **PLC** های زیمنس از **(FB) Function Block** ها می توان برای این منظور استفاده نمود.

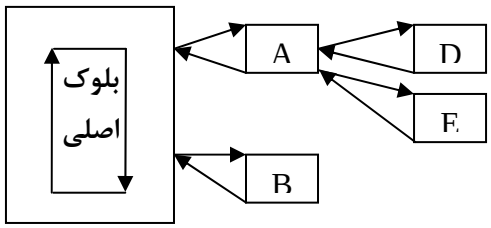
نوع دیگری از Block، دسته از آنها هستند که سازماندهی سایر Block ها را به عهده دارند. مثلا اگر در زمان خاصی قرار باشد عملیاتی اجرا شود و یا برنامه اصلی در اثر اتفاقی متوقف شده و کار دیگری را انجام دهد و یا مواردی دیگر می توان از آنها استفاده نمود .

در **PLC** های زیمنس به این نوع **Block** ها **(OB) Organizution Block** , گفته می شود.

معمولا برنامه دارای یک **Block** اصلی است که اجرای برنامه در **CPU** از آنجا آغاز می شود. و سایر **Black** ها نیز در آن صدا زده می شوند. داخل هر **Block** می توان چند بلوک دیگر را صدا زد و از آنجا استفاده نمود.

در PLC های زیمنس OB بصورت Cyclic اجـرا می شود ، بنابراین برنامه کـاربـر را در آن قـرار می دهند. که ممکن است نوشتن این برنامه از تعدادی از انواع بلوک های دیگر استفاده شده باشد.

برنامه نمونه زیر ترتیب اجرای دستورالعملهای درون **Block** های مختلف یک برنامه را نشان می دهند.



اجرای برنامه از شروع دستورالعملهای **Block** اصلی آغاز می شود، خط به خط برنامه آن را اجرا می کند ، تا جایی که به دستور العمل صدا کردن **Block A** می رسد، پس به اولین خط برنامه **A** رفته و به ترتیب آنها را اجرا می کند ، تا به دستور صدا کردن **Block D** می رسد ،

پس به اولین خط برنامه **D** رفته خطوط آن را به ترتیب اجرا می کند ، پس از اتمام به **A** باز می گردد تا باقی دستور العملهای آن اجرا نماید، سپس به دستور صدا کردن **Block E** می رسد، به اجرای آن می پردازد و پس از اتمام **E** به **A** باز می گردد و بقیه برنامه درون **A** را اجرا می نماید ، با اتمام آن برنامه به بلوک اصلی بازگشته و سایر دستورات آن را اجرا می کند ، تا به دستور صدا کردن **B** می رسد ، **B** را اجرا می کند و مجددا به بلوک اصلی بازگشته و تا انتها آن را انجام می دهد . پس از اتمام دستورات بلوک اصلی خروجی ها را به کانالهای خروجی اعمال کرده و به ابتدای سیکل برنامه باز می گردد.

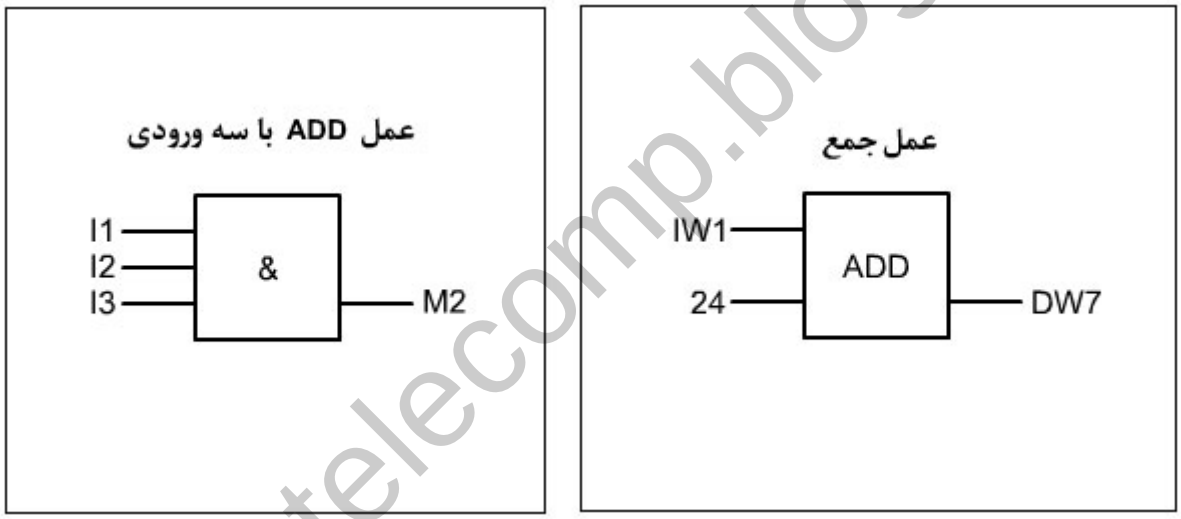
۴-۴) مدل های برنامه نویسی

مجموعه دستورالعملهای یک برنامه بصورت زبان ماشین در اختیار **PLC** جهت اجرا قرار می گیرد ، اما جهت استفاده راحت تر و سریعتر کاربر، برنامه **PLC** را با سه روش نمایش (زبان برنامه نویسی)میتوان ایجاد کرد.

این سه روش عبارتنداز :

- **Function Block Diagram**

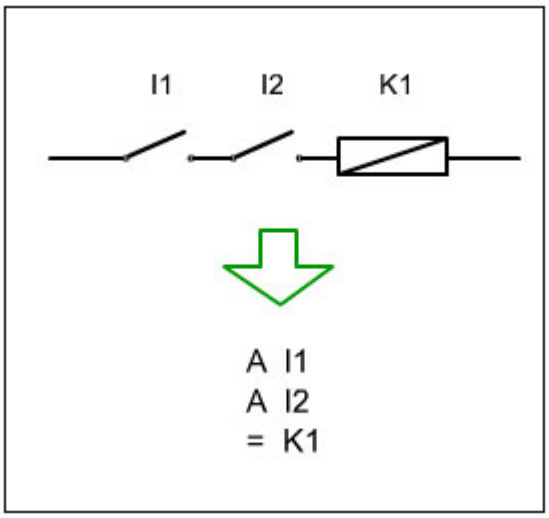
استفاده از بلوک های مستطیل شکل برای نشان دادن یک عمل ریاضی یا منطقی.



ورودی ها در سمت چپ و خروجی ها در سمت راست قرار می گیرند. درک **Logic** نوشته شده با این مدل بسیار آسان و راحت است.

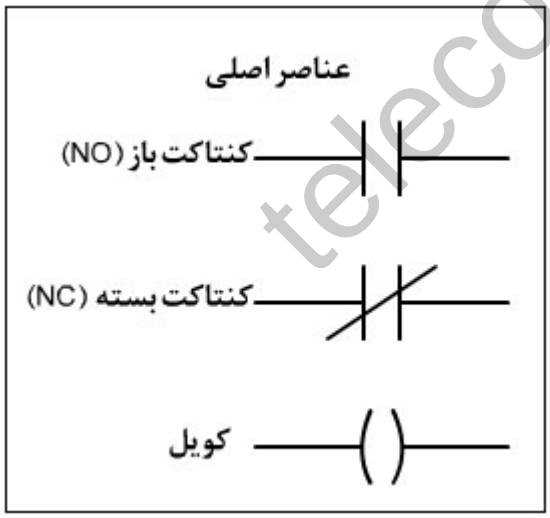
- **Statement List**

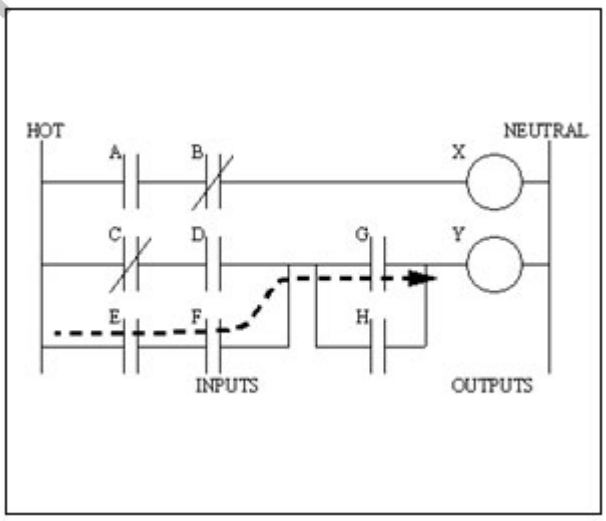
استفاده از زبان ماشین **PLC** . این مدل برنامه نویسی بسیار قدرتمند، ولی دشوار است



- **Ladder Logic**

این مول شباهت زیادی به نقشه مدارات رله-کنتاکتوری دارد.





این سه روش بخشی از استاندارد **IEC 61131-3** هستند که الگویی برای مدل های برنامه نویسی در **PLC** هاست.

5-What is IEC 1131?

Developed with the input of vendors, end-users and academics, IEC 1131 consists of five parts:

1. General information
2. Equipment and test requirements
3. PLC programming languages
4. User guidelines
5. Communications

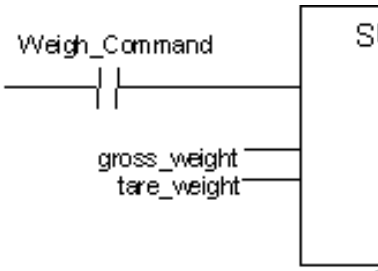
IEC 1131-3 is the international standard for programmable controller programming languages. As such, it specifies the syntax, semantics and display for the following suite of PLC programming languages:

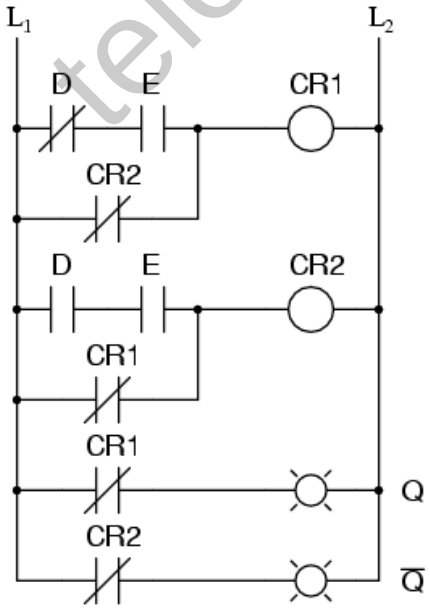
- Ladder diagram (LD)
- Sequential Function Charts (SFC)
- Function Block Diagram (FBD)
- Structured Text (ST)
- Instruction List (IL)

One of the primary benefits of the standard is that it allows multiple languages to be used within the same programmable controller. This allows the program developer to select the language best suited to each particular task. An analogy is that a mechanic wouldn't attempt to repair an automobile using only a screwdriver. The mechanic has a variety of tools available and chooses the best one for each task. Follow the above links for a description of each of the IEC 1131-3 languages and the types of applications they are best suited to.

5-1) IEC 1131 Ladder Diagram

For people who understand relay controls, LD continues to be an advantage in terms of usability. Although it is possible to program all control logic in LD, supplementing LD with other languages allows users access to the language best suited for a particular control task. The standard's implementation of LD appears below.



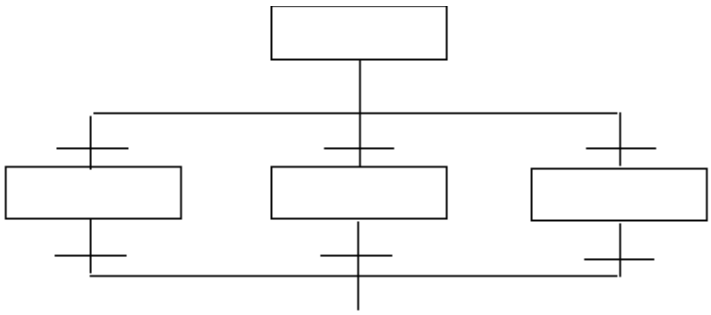


E	D	Q	$\overline{Q}$
0	0	latch	latch
0	1	latch	latch
1	0	0	1
1	1	1	0

5-2) IEC 1131 Sequential Function Charts

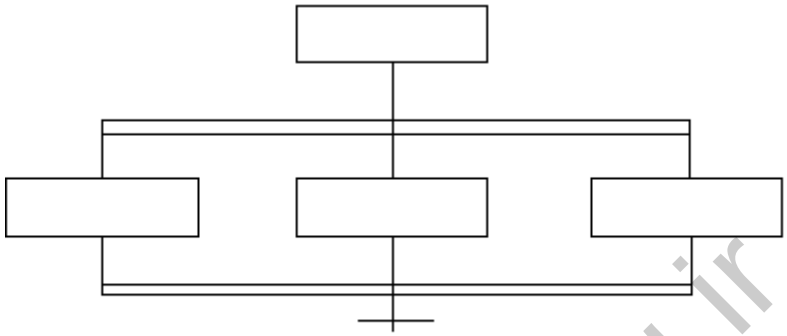
SFC programming offers a graphical method of organizing the program. The three main components of an SFC are steps, actions and transitions. Steps are merely chunks of logic, i.e., a unit of programming logic that accomplishes a particular control task. Actions are the individual aspects of that task. Transitions are the mechanisms used to move from one task to another. Control logic for each Step, Action and Transition is programmed in one of the other languages such as Ladder Diagram or Structured Text.

As a graphical language, SFC programming offers you several choices for executing a program, each depicted in a visually distinct way (Fig. 1). In a sequential configuration, the processor simply executes the actions in step 1 repeatedly, until the transition logic becomes true. The processor then proceeds to step 2. In a selection branch, only one branch is executed depending on which transition is active. In a simultaneous branch, all branches are executed until the transition becomes active. In addition to various types of branches, the operation of individual actions within a step can be varied with the use of action qualifiers.



SFC Selection Branch

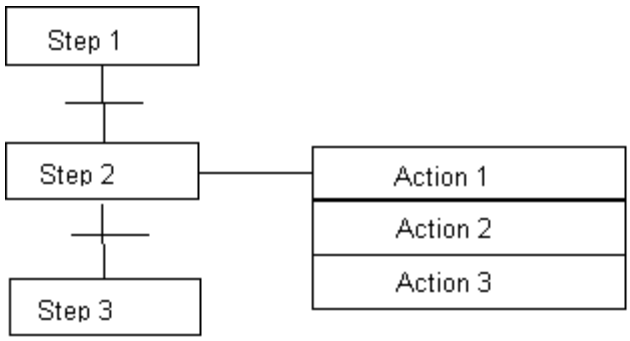
Action qualifiers (Fig. 2) determine how the action is scanned and allow actions to be controlled without additional logic. For example, one could use the L qualifier to limit the time that ingredient valve B is opened.



SFC Simultaneous Branch

In practice, an active step is highlighted to signal to the programmer which part of the program is executing - a useful feature for troubleshooting. This highlighting is an example of the standard's extensibility - the ability of a vendor to add a feature not specified in the standard.

Note that the standard offers SFC programming as an organizing tool. The user chooses whether to use it or not, based on whether the process being controlled is sequential in nature. And even if SFC programming is used, the actions will be written in one of the four programming languages described below. Figure 3 shows a sample net weight calculation as it would be performed in each of these languages. In each example, net weight is calculated by subtracting tare weight from the gross weight.



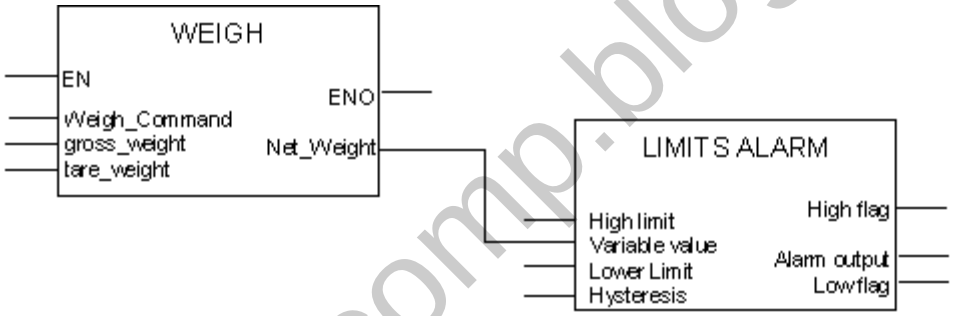
SFC Sequential configuration

SFC Action Qualifiers

SFC Qualifyer	Description
N	Nonstored. Terminate when the step becomes inactive.
S	Set (stored). Continue after the step is deactivated, until the action is reset.
R	Reset. Terminate the execution of an action previously started with the S, SD, SL, or DS qualifier.
L	Time Limited. Start when step becomes active and continue until the step goes inactive or a set time passes.
D	Time Delayed. Start a delay timer when the step becomes active. If the step is still active after the time delay, the action starts and continues until deactivated.
P	Pulse. Start when the step becomes Active/Deactive and execute the action only once.
SD	Stored and time Delayed Action starts after time delay, continues until reset.
DS	Delayed & Stored. If step is still active, action starts after time delay, continues until reset.
SL	Stored & timeLimited. Action starts when step becomes active, continues for a set time or until reset.

5-3) IEC 1131 Function Block Diagram Overview

Like SFC, FBD is a graphical language that allows programming in other languages (ladder, instruction list, or structured text) to be nested within the FBD. In FBD, program elements appear as blocks which are "wired" together in a manner resembling a circuit diagram. FBD is most useful in those applications involving a high degree of information/data flow between control components, such as process control.



5-4) IEC 1131 Structured Text Overview

This high-level language resembles Pascal or Basic, and, in fact, people trained in computer programming languages often find it the easiest language to use for programming control logic. When symbolic addressing is used, ST programs resemble sentences, making it highly intelligible to the novice user as well. ST is ideal for tasks requiring complex math, algorithms or decision-making. Its concise format allows a large algorighm to be displayed on a single page (vs multiple pages of ladder logic).

The IEC 1131-3 standard is extensible. I.E. Vendors may augment their offerings to meet the needs of specific markets. As an example of this extensibility Rockwell Software augments ST with an exclusive feature called "PowerText?". It supplements standard ST with real-time display of discrete status, force status, analog values and floating-point values. This PowerText information is automatically integrated into the source code, and is invaluable for debugging and application commissioning.

5-4-1) Benefits of Structured Text

- People trained in computer languages can easily program control logic
- Symbols make the programs easy to understand
- PowerText facilitates system debugging and application commissioning
- Programs can be created in any text editor
- Runs as fast as ladder logic

5-4-2) Structured Text Constructs

- Bit / Word assignment
- IF-THEN-ELSE
- CASE
- FOR-NEXT
- WHILE
- REPEAT
- Ladder equivalent instructions

5-4-3) Structured Text Examples

Example 1 (Sorting machine)

```
IF (LIMIT_SWITCH_1 AND BOX_PRESENT) THEN
GATE1 := OPEN;
GATE2 := CLOSE;
ELSIF ((LIMIT_SWITCH_2 OR (WEIGHT <> SETPOINT))) THEN
GATE1 := CLOSE;
GATE2 := OPEN;
ELSIF (LIMIT_SWITCH_3 XOR LIMIT_SWITCH_4) THEN
GATE1 := OPEN;
GATE2 := OPEN;
ELSE
GATE1 := CLOSE;
GATE2 := CLOSE;
END_IF;
```

Example 2 (Split-range temperature control, with deadband)

```
CASE (TEMPERATURE) OF
0 .. 120 :
HEAT_VALVE := OPEN;
COOL_VALVE := CLOSE;
150 .. 32767 :
HEAT_VALVE := CLOSE;
COOL_VALVE := CLOSE;
ELSE
HEAT_VALVE := CLOSE;
COOL_VALVE := CLOSE;
END_CASE;
PID (CONTROL_BLOCK, TANK_#27_TEMPERATURE, TIEBACK, COOLANT_VALVE);
```

Example 3 (Computational examples)

```
POWER := (CURRENT ** 2.0) * RESISTANCE;
F8:1 := ((N7:1 * 3.1428571) + (N7:3 / N7:4));
F8:0 := ( SIN (ANGLE)) MOD 6.0;
JSR (3, 0);
```

Example 4 (If-Then-Else example)

```
IF I:000/0 AND !I:001/5 THEN
N7:0 := 1;
ELSIF i:000/2 OR (N7:5 <> (N7:6 * N7:2)) THEN
N7:0 :=2;
```

```
ELSIF !I001/4 THEN
N7:0 := 3;
ELSE
N7:0 :=4;
END_IF;
```

Example 5 (For-Next examples)

```
FOR N7:0 := 10 TO 0 BY -1 DO
N7:4 := N7:4 + 1;
END_FOR;

FOR N7:1 := 0 TO 10 DO
N7:1 := N7:1 + 1;
END_FOR;
```

Example 6 (While example)

```
WHILE I:000/0 AND (I:001/0 OR !I:002/1) DO
N7:0 := N7:0 + 1;
END_WHILE;
```

Example 7 (Case example)

```
CASE N7:12 OF
0:
N7:1 := 1;
1, 9:
N7:1 := 2;
2..4:
N7:1 := 3;
10, 5..8:
N7:1 := 4;
ELSE
N7:1 :=5;
END_CASE;
```

5-4) IEC 1131 Structured Text Overview

This high-level language resembles Pascal or Basic, and, in fact, people

5-5) IEC 1131 Instruction List Overview

This low-level language is similar to Assembly language and is useful in cases where small functions are repeated often. Although it is powerful, it is considered to be difficult to learn.

5-4-1) Instruction List example

(Calculate new weight by subtracting tare weight from net weight)

```
LD weigh_command
JMPC WEIGH_NOW
ST ENO
RET
WEIGH_NOW: LD gross_weight
SUB tare_weight
```

۶- اجزای برنامه نویسی PLC

جهت ایجاد یک برنامه کاربردی می بایست با اجزاء سازنده آن که انواع دستور العمل ها مبدل ها مقایسه گرها بکار گرفتن زمان سنج ها و یا شمارنده ها است آشنا شد.

۶-۱) دستورالعمل ها:

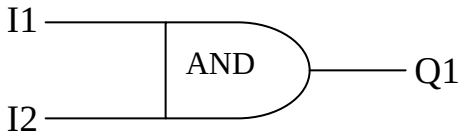
با توجه به سیگنالی که در اختیار است و همچنین منطقی که می خواهیم اجرا شود نیاز به انواع مختلف دستورالعمل ها داریم . گـــروهی از دستورالعمل ها ، برای کار با المانهای بیتی بکـار مـی رونـد (اصطلاحا **Bitwise** هستند) و به آنها دستور العمل های مـنطقی می گویند.

چند نمونه از این دستور العمل ها عبارتند از :

… RS ,SR , RESET , SET , Not , OR , AND

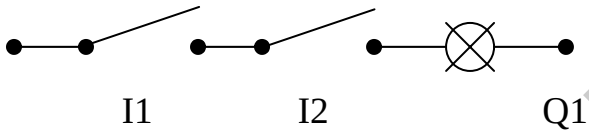
۱-۱-۶) **AND** :

نتیجه **AND** دو یا چند متغیر ، تنها در صورتی که همگی آنها منطق یک داشته باشند ، یک می شود ، در غیر این صورت صفر می باشد. نماد منطقی و جدول صحت آن بصورت زیر می باشد.



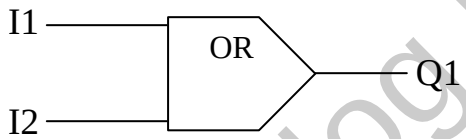
ورودی		خروجی
I_1	I_2	Q_1
0	0	0
0	1	0
1	0	0
1	1	1

پیاده سازی **AND** در فرم **LAD** با سری قرار دادن دو متغیر حاصل می شود. در این حالت اگر هر دو سوئیچ **(I 1, I 2)** بسته باشند ، لامپ **Q1** روشن خواهد شد در غیر اینصورت خاموش می ماند .



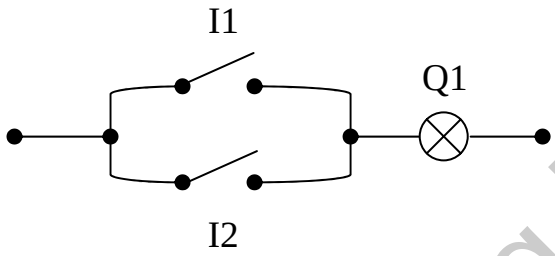
۶-۱-۲) **OR** :

نتیجه دو یا چند متغیر بیتی تنها در صورتی که تمامی آنها صفر است ، و چنانچه تنها یکی از متغیر ها یک باشد ، یک شدن نتیجه تامین می گردد. نماد منطقی و جدول صحت آن بصورت زیر است :



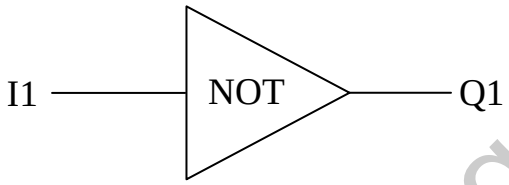
ورودی		خروجی
I_1	I_2	Q_1
0	0	0
0	1	1
1	0	1
1	1	1

پیاده سازی **OR** در فرم **LAD** ، با موازی بستن متغیر ها نشان داده می شود. با بسته شدن هر یک از دو سوئیچ **I 1** یا **I 2** لامپ روشن می شود واگر هر دو باز شوند ، لامپ خاموش می شود.



۳-۱-۶) **NOT** :

NOT نتیجه هر متغیر بیتی را وارونه می کند . چنانچه ورودی آن یک باشد نتیجه را صفر و بالعکس نتیجه ورودی صفر را یک می کند . نماد منطقی و جدول صحت آن بصورت زیر است :



ورودی	خروجی
I 1	**Q 1**
0	1
1	0

۶-۱-۴) **SET** :

با استفاده از **SET** می توان مقدار یک متغیر بیتی را یک کرد. تحریک متغیر تنها با لبه بالا رونده صورت می گیرد. اگر بیتی به یک **SET** شد ، یک باقی می ماند تا اینکه **REST** شود.

ورودی	خروجی
I 1	**Q 1**
0	0
1	1
0	1

۵-۱-۶) **RESET** :

با استفاده از RESET می توان به یک متغیر مقدار " صفر " داد . این عمل با تحریک توسط لبه بالا روند انجام می شود . وقتی بیتی RESET می شود تا هنگامیکه SET نشود ، مقـدار آن صفر باقی می ماند

ورودی	خروجی
I 1	**Q 1**
0	1
1	0
0	0

۲-۶) **مبدل ها** (Converter) **و مقایسه گرها** (Comparators) :

اعداد با فرمت های مختلفی همچون اعداد صحیح **(Integer)** و یا اعداد حقیقی **(Real)** ممکن است در اختیار باشند. عمل مبدل ها تبدیل عدد از فرمتی به فرمت دیگر و یا اعمال تغییراتی بر فرمتهای عددی می باشد مثلا در نرم افزار **Step 7** که مخصوص برنامه نویسی برای **PLC** های **SIEMENS** است .

مبدل **DI-R** یک عدد صحیح که در ۳۲ بیت از حافظه قرار دارد را به عددی با فرمت حقیقی تبدیل می کند و یا مبدل **I-DI** یک عدد صحیح که در ۱۶ بیت از حافظه قرار دارد را به عدد صحیحی که ۳۲ بیت حافظه لازم دارد تبدیل می کند .

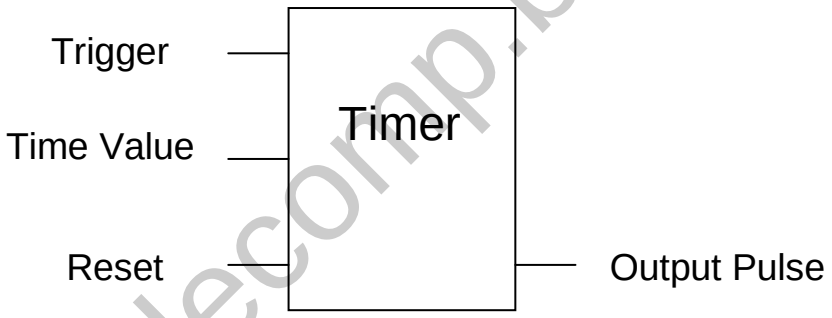
مقایسه گرها اعداد با فرمت های یکسان را با هم مقایسه می کنند. مثلا دو عدد حقیقی را با هم مقایسه می کند و مساوی بودن آنها ، نا برابر بودن آنها ، بزرگتر بودن و یا کوچکتر بودن یکی از آنها را نشان می دهند.

مثلا مقایسه گرهای فوق در **Step7** با دستورات **LT-R , G7 –R , NE-R , EQ-R** معادلند.

۳-۶) زمان سنج ها (Timers) :

تایمرها برای محاسبه زمان خروجی تایمرها ، بر اساس نوع آنها و با توجه به زمانی که برایشان تعریف شده ، ایجاد می شود. به این ترتیب که یک پایه ورودی (پایه **Start**) عملکرد آن را فعال نموده ، زمان مورد نظر محاسبه می شود و بر حسب نوع تایمر ، خروجی متناسبی ایجاد می گردد.

ساختار کلی تایمرها به صورت زیر است:



با اعمال یک "لبه بالا رونده" (**Positive Edge Trigger**) یا انتقال از **"0"** به **"1"** روی پایه تحریک، تایمر فعال شده و شروع به کار می نماید و پالس مشخصی در خروجی آن ظاهر می گردد که شکل و اندازه این پالس بستگی به نوع تایمر دارد. مقدار زمان قابل اندازه گیری نیز از طریق پایه **Time Value** به آن داده می شود.

برای متوقف کردن تایمر نیز از پایه ای به نام **Reset** استفاده می شود. با اعمال یک لبه بالارونده روی این پایه تایمر غیرفعال شده و خروجی آن نیز صفر می گردد.

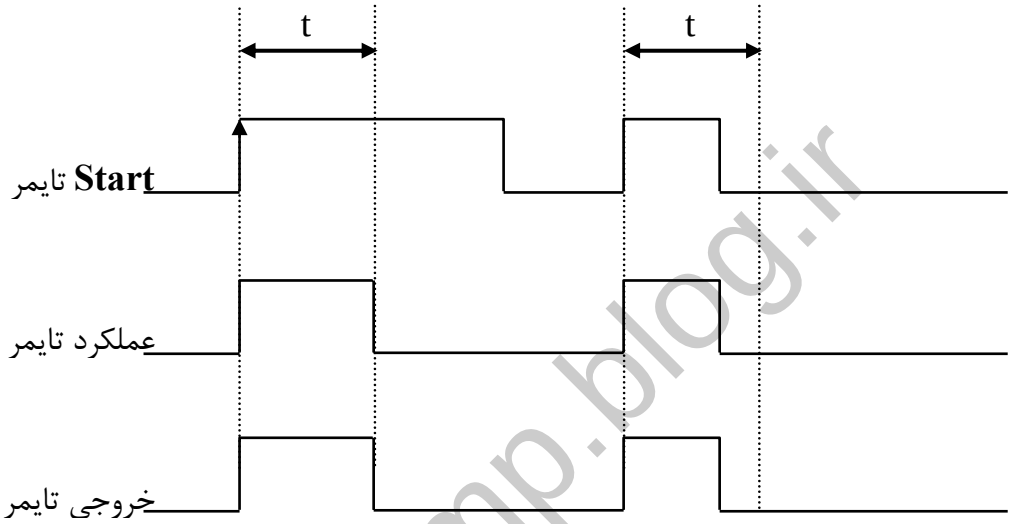
در **PLC** ها جهت برنامه نویسی ممکن است از انواع تایمرها استفاده شود. معمول ترین نوع تایمرها که در اکثر **PLC** ها موجودند و کاربرد دارند ، عبارتند از

- **Pulse Timer**
- **On –delay Timer**
- **OFF-delay Timer**

در توضیحات تایمرها ، به نحوه فعال شدن ، محاسبه زمان و تولید خروجی که تفاوت نوع عملکرد و نحوه استفاده از تایمرها می شوند ، توجه نمائید .

۶-۳-۱) **Pulse Timer** :

در این نوع تایمر با فعال شدن پایه تحریک تایمر شروع به کار کرده و در خروجی پالسی با عرض **t** ایجاد می کند (**t** مدت زمان **Set** شده در تایمراست)

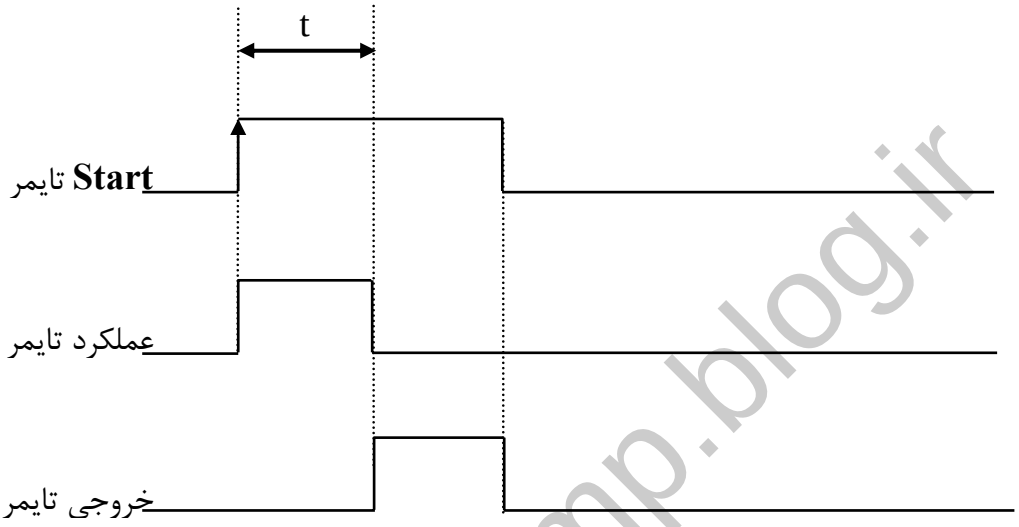


چنانچه ورودی تایمر زودتر از زمان **t** صفر شود، خروجی نیز صفر خواهد شد. اصطلاحا می گوییم ورودی، خروجی را با خودش پایین می کشد.

از این تایمر برای حالت هایی که لازم است یک سیگنال با اعمال تحریک در مدت مشخصی **"1"** و سپس **"0"** شود، استفاده می گردد.

۶-۳-۲) **ON-Delay Timer** :

در این تایمر، خروجی پس از تاخیر زمانی **Set** شده برای تایمر پس از یک شدن ورودی ، یک می شود.

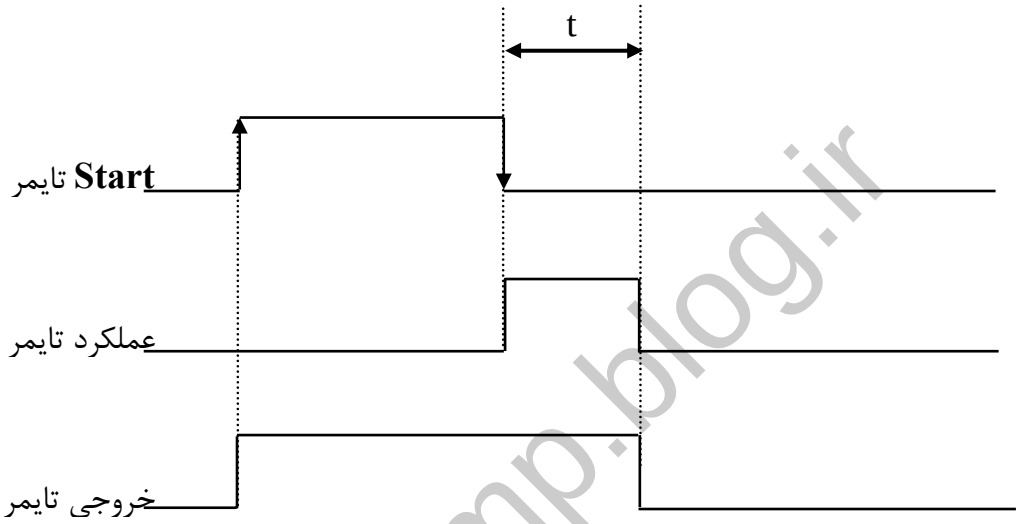


با **"0"** شدن ورودی، خروجی نیز **"0"** صفر می شود.

چنانچه ورودی تایمر زودتر از زمان **t** صفر شود، خروجی نیز صفر خواهد شد. اصطلاحا می گوییم تایمر تحریک را رد می کند.

۶-۳-۳) **OFF-Delay Timer** :

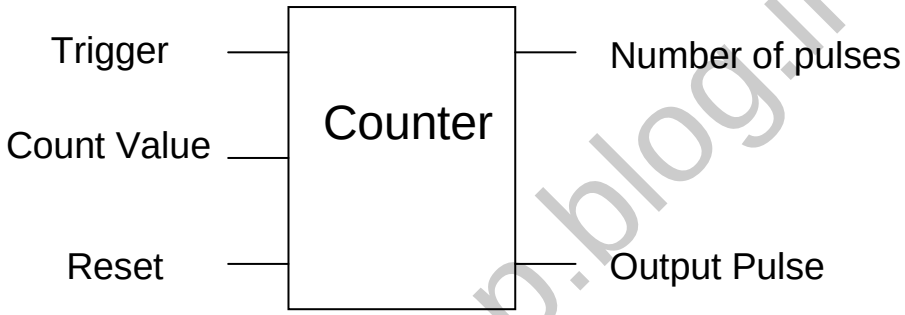
در این تایمر، خروجی با تحریک ورودی فعال شده و هنگامیکه ورودی **"0"** شد، با تاخیر زمانی **t** (**Set** شده برای تایمر) پایین می آید .



۶-۴) شمارنده (**Counters**) :

عمل شمارنده ها ، شمارش تعداد پالس های ورودی است .

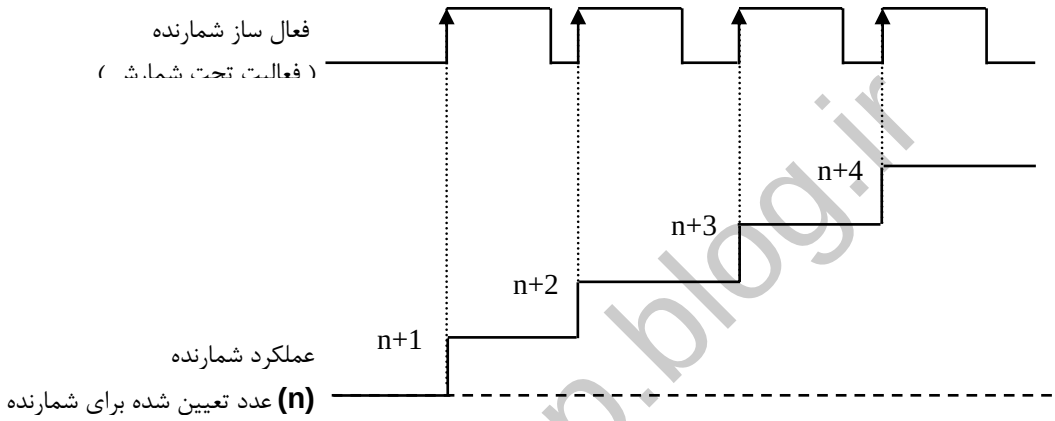
شمارنده ها معمولا یک ورودی جهت تعیین تعداد دفعات شمارش و یک ورودی دیگر برای فعال سازی شمارنده ، دارند.



از شمارنده ها در برنامه نویسی **PLC** استفاده های زیادی می شود. در اکثر **PLC** ها دو نوع شمارنده صعودی **(UP Counter)** و نزولی **(DOWN Counter)** وجود دارند.

۶-۴-۱) **UP Counter** :

در این نوع شمارنده ، با هر بار رخ دادن فرایندی که به پایه فعال ساز شمارنده وصل است ، یک واحد به عدد تعیین شده شمارنده اضافه می شوند.



۶-۴-۲) DOWN Counter :

در این نوع شمارنده با هر بار رخ دادن فریندی که به پایه فعال ساز شمارنده متصل است ، یک واحد از عدد تعیین شده شمارنده کم می شود.