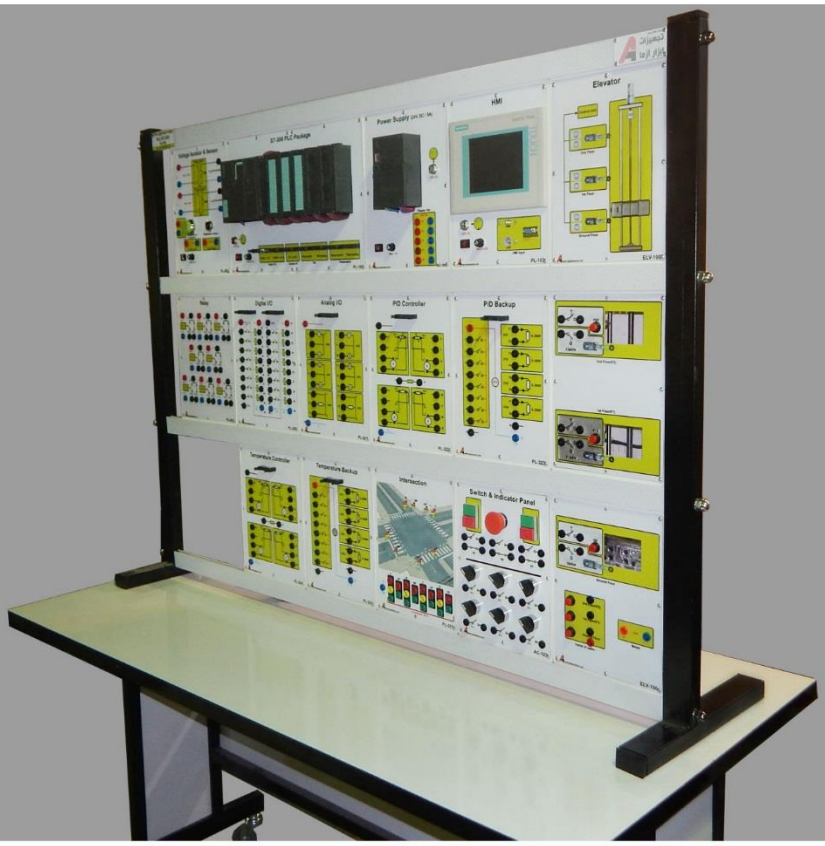
شرکت دانش بنیان

تجهیزات ابزار آزما

نوآوری و فناوری برای توسعه



دستور کار جامع اتوماسیون

تجهیزات آزمایشگ
Laboratory Equipment

نوآوری و فناوری برای توسعه

خراسان رضوی، مشهد، بزرگراه آسیایی، جاده شهرک صنعتی طوس، شهرک فناوری های نوین غذایی، ساختمان پژوهش، طبقه اول، شرکت تجهیزات ابزار آزما

www.abzarazma.ir

info@abzarazma.com

www.aparat.ir/abzarazma

+98 51 35420099 (3line)

شرکت دانش بنیان

تجهیزات ابزار آزما

گاهی مهندسی برق

Electrical Engineering L

ازمایشگاه های اتوماسیون صنعتی و ابزار دقیق

Industrial Automation and Instrumentation Labs

ازمایشگاه ابزار دقیق

- آموزنده الکتروپنوماتیک پایه (EP-100)
- آموزنده الکتروپنوماتیک تکمیلی (EP-101)
- آموزنده الکتروپنوماتیک پیشرفته (EP-102)
- آموزنده ابزار دقیق پایه (AI-113)
- آموزنده ابزار دقیق تکمیلی (AI-114)

ازمایشگاه اتوماسیون صنعتی

- آموزنده PLC LOGO (AI-101)
- آموزنده PLC S7-300 (AI-104)
- آموزنده PLC LG (AI-105)
- آموزنده PLC S7-300 پیشرفته (AI-106)
- آموزنده شبکه صنعتی با PLC S7-300 (AI-108)
- آموزنده مانیتورینگ صنعتی (AI-110)
- آموزنده سیستم های کنترل صنعتی (AI-117)
- آموزنده کنترل کننده منطقی برنامه پذیر (IC-104)

ازمایشگاه های سیستم های کنترل

Control Systems Labs

ازمایشگاه کنترل صنعتی

- آموزنده کنترل دما (IC-100)
- آموزنده کنترل فشار (IC-101)
- آموزنده کنترل سطح و دبی (IC-102)
- آموزنده کنترل سرعت موتور (IC-103)
- آموزنده کنترل کننده منطقی برنامه پذیر (IC-104)
- شبیه ساز آسانسور (AI-91)
- شبیه ساز چراغ راهنمایی (AI-92)
- شبیه ساز کنترل دما (IC-90)
- شبیه ساز کنترل سطح (IC-91)
- آموزنده کنترل کامپیوتری (AI-109)
- آموزنده سیستم های کنترل درایوهای صنعتی (AI-117)
- ماژول مانیتورینگ و کنترل نرم افزاری (DC-65)

ازمایشگاه سیستم های کنترل خطی

- آموزنده کنترل آنالوگ (DC-100)
- آموزنده کنترل آنالوگ و دیجیتال سروو موتور (DC-102)

ازمایشگاه سیستم های کنترل دیجیتال

- آموزنده کنترل دیجیتال (DC-101)
- آموزنده کنترل آنالوگ و دیجیتال سروو موتور (DC-102)

ازمایشگاه کنترل پیشرفته

- آموزنده گوی معلق (SB-100)
- آموزنده پاندول معکوس (IP-101)
- آموزنده شناسایی سیستم (SI-100)
- ربات مسیریاب پیشرفته (RO-100)

ازمایشگاه تجهیزات اندازه گیری

- گشتاورسنج (IM-51)
- کنترل کننده PID (IM-40)
- سرعت سنج (IM-50)
- فرکانس متر (IM-30)
- اندازه گیر فازور (IM-31)
- رله سنکرون چک (IM-21)
- سنکرون ساز اتوماتیک سه فاز (IM-22)
- کسینوس فی متر (IM-12)
- رله حفاظت فرکانسی (IM-20)
- مولتی متر سه فاز (IM-10)
- مولتی فانکشن متر سه فاز (IM-11)

ازمایشگاه ماشین های الکتریکی

- ترانسفورماتور سه فاز (T-12)
- ترانسفورماتور تکفاز (T-11)
- ماشین DC شنت (M-87)
- ماشین DC چندکاره (M-86)
- ماشین AC چندکاره (M-85)
- ماشین القایی روتور سیم پیچی سه فاز (M-82)
- ماشین سنکرون سه فاز (M-80)

کارگاه تاسیسات الکتریکی تکمیلی

- کارگاه سیستم اعلام حریق (ET-116)
- کارگاه سیستم ضد سرقت (ET-115)
- کارگاه دوربین مدار بسته (ET-112)
- کارگاه آیفون صوتی و تصویری (ET-111)
- کارگاه سیستم متمرکز (ET-110)
- کارگاه سیستم تلفن (ET-109)

کارگاه سیم پیچی

- آموزنده ماشین های الکتریکی AC مدل گسترده (MC-112)
- آموزنده ماشین های الکتریکی DC مدل گسترده (MC-111)
- آموزنده کارگاه سیم پیچی (WW-100)

کارگاه برق

- آموزنده مدار فرمان (CO-100)
- آموزنده کارگاه برق خانگی و صنعتی (EW-101)
- آموزنده کارگاه برق خانگی (EW-100)

کارگاه ساختمان هوشمند

- آموزنده خانه هوشمند پیشرفته (SH-101)
- آموزنده خانه هوشمند پایه (SH-100)

کارگاه تاسیسات الکتریکی

- آموزنده تاسیسات الکتریکی (WW-102)
- آموزنده کارگاه سرکابل و مفصل (WW-101)

ازمایشگاه مخابرات دیجیتال

- آموزنده ازمایشگاه مخابرات آنالوگ و دیجیتال (TC-105)
- آموزنده ازمایشگاه مخابرات دیجیتال (TC-103)

ازمایشگاه مدارهای مخابراتی

- آموزنده ازمایشگاه مخابرات آنالوگ و دیجیتال (TC-105)
- آموزنده مدولاسیون دامنه و فرکانس AM/FM (TC-103)
- آموزنده ازمایشگاه مخابرات آنالوگ (TC-101)

تجهیزات صنعتی

Industrial Instrument

ازمایشگاه های تاسیسات الکتریکی

Electrical Installations Labs

دستور کار آزمایشگاه اتوماسیون صنعتی و PLC

پیشگفتار:

پیشنهاد می شود شروع آزمایشگاه با یک یا چند بازدید از مرکز کاربردی مرتبط با مطالب درس شروع شود. در این دستور کار مطالب اساسی در قالب ۱۹ آزمایش ارائه گردیده است. همچنین سعی شده است عمدهٔ مطالب پیش زمینه در متن گنجانده شود و دستورکار از این جهت کمتر نیاز به مراجع بیرونی دارد.

مطالب بیان شده در د ستور کار هر آزمایش شامل مقدمه، شرح آزمایش و تحلیل و جداول مربوطه و در پایان سؤالات مربوط به آزمایش می‌باشد. این دستور کار طوری طرح شده است تا دانشجو حین انجام مراحل مختلف آزمایش بخش‌های مختلف آن را تکمیل نماید.

هر دانشجو قبل از حضور در کلاس می‌بایست یک پیش گزارش راجع به مباحث جلسه جاری و گزارش تکمیل شده جلسه قبل را تحویل نماید. مسلماً گزارش حاصل همراه با نقص و کاستی‌هایی است که با پیشنهادات شما مدرسین و دانشجویان عزیز در نسخه های بعدی برطرف خواهد شد.

نکات مهم:

هشـــدار۱ (خطر شـــوک الکتریکی)

در هنگام انجام سیم بندی دقت کنید که برق دستگاه قطع باشد.

هشـــدار۳ (خطر آســیب به دستگاه و شوک الکتریکی)

برای تعمیر تجهیزات از افراد واجد شـرایط و با هماهنگی شـرکت سـازنده استفاده نمایید.

هشـــدار۴ (خطر آســیب به دستگاه و شوک الکتریکی)

هیچ‌گونه اصلاح و یا تغییری در وضعیت فعلی تجهیزات مجاز نیست.

هشـــدار۶ (خطر آســیب به تجهیزات)

پیش از وصل کردن برق دستگاه، سیم‌بندی با حضور مدرس بررسی گردد.

هشـــدار۸ (خطر آســیب به تجهیزات)

به تحلیل ورودی و خروجی‌های تجهیزات اقدام شـــود و از اعمال ورودی خارج از محدوده مجاز به تجهیز خودداری شود.

فهرست مطالب

جـــــدول راهنما

شماره و عنوان آزمایش	AI-101	AI-104	AI-105	AI-106	AI-108	AI-109	AI-110	AI-117
۱. آشنایی با کنترل کننده های منطقی برنامه پذیر	❋		❋	❋			❋	❋
۲. آشنایی با نرم افزار GMWIN			❋					
۳. مدارهای فرمان و شبیه سازی در GMWIN			❋					
۴. ذخیره سازی اطلاعات و ساختمان داده ها			❋					
۵. عملگرهای مقایسه ای در GMWIN			❋					
۶. فانکشن و فانکشن بلاک ها در GMWIN			❋					
۷. تایمرها در GMWIN			❋					
۸. شمارنده ها(کانترها) در GMWIN			❋					
۹. تبدیل داده های آنالوگ به دیجیتال (A/D) و دیجیتال به آنالوگ (D/A)در GMWIN								
۱۰. الکتروپنوماتیک با PLC و برنامه نویسی در GMWIN								
۱۱. کنترل دور موتور القایی با اینورتر					❋			❋
۱۲. راه اندازی موتور با PLC					❋			❋
۱۳. معرفی ماژول های PLC S7-300		❋		❋	❋			
۱۴. انواع ورودی و خروجی ها در PLC S7-300		❋		❋	❋			
۱۵. مقدمات برنامه نویسی PLC S7-300		❋		❋	❋			
۱۶. نرم افزارSTEP 7 - Micro/WIN32		❋		❋	❋			
۱۷. دیاگرام سیم کش شده استارت کننده یک موتور سه فاز Hard-wired motor starter	❋	❋		❋	❋		❋	
۱۸. آشنایی با ورودی / خروجی های نوع آنالوگ Introduction to Analog I/O	❋	❋		❋	❋		❋	
۱۹. آشنایی با تایمر های پی ال سی ها (Introduction to Timers) PLC S7-300		❋		❋	❋			

۱ آشنایی با کنترل کننده های منطقی برنامه پذیر

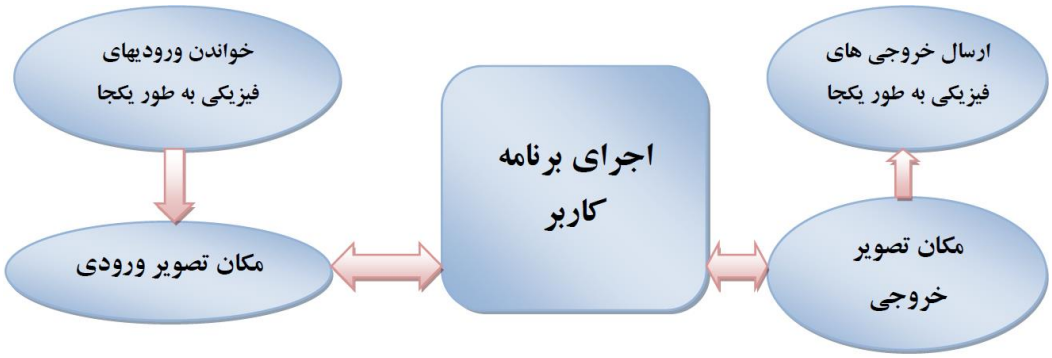
PLC از عبارت *(programmable logic control)* به معنای کنترل کننده منطقی قابل برنامه گرفته شده است. *PLC* کنترل کننده ای است نرم افزاری که در ورودی اطلاعاتی را به صورت باینری یا آنالوگ دریافت و آنها را طبق برنامه ای که در حافظه اش ذخیره شده است پردازش می کند و نتیجه عملیات را از قسمت خروجی به صورت فرمانهایی به گیرنده ها و اجرا کننده های فرمان ارسال می کند .به عبارت دیگر *PLC* عبارت از یک کنترل کننده منطقی است که می تواند منطق کنترل را توسط برنامه برای آن تعریف نمود و در صورت نیاز به راحتی آن را تغییر داد .وظیفه *PLC* قبلاً به عهده مدارات فرمان و رله های کنتاکتوری بود که امروزه استفاده از آنها منسوخ شده است .از اشکالات عمده ای رله ها ای بود که با افزای ای رله ها حج و وزن مدارات فرمان بسیار بزرگ شده و قیمت آنها نیز افزایش می یافت و نهایتاً عیب یابی این گونه مدارات بسیار پیچیده و زمان بر می گردید .برای رفع این معظل مدارات فرمان الکترونیکی ساخته شدند، که آنها به دلیل ای که تک کار بودند و برای استفاده در مدار می بایستی تغییرات عمده ای در آنها ایجاد می شد کارایی کمی داشتند .با استفاده از *PLC* تغییر در روند تولید یا عملکرد ماشین به راحتی صورت می گیردزیرا دیگر لازم نیست سیم کشی هاو سخت افزار سیستم کنترل تغییر کند و تنها کافی است چند سطر برنامه نوشت و به *PLC* ارسال کرد تا کنترل مورد نظر تحقق یابد .از طرف دیگر قدرت *PLC* در انجام عملیات منطقی و محاسباتی و مقایسه ای و نگهداری اطلاعات به مراتب بیشتر از تابلوهای فرمان معمولی است .اکنون بیشتر به تفاوتها و مزایای *PLC* نسبت به مدارات کنتاکتوری می پردازیم.

- استفاده از *PLC* موجب کاهش حجم تابلوی فرمان می گردد.
- استفاده از *PLC* مخصوصاً در فرآیندهای عظیم موجب صرفه جویی قابل توجهی در هزینه لوازم و قطعات می شود.
- *PLC*استهلاک مکانیکی ندارد بنابرای علاوه بر عمر بیشتر نیازی به تعمیرات و سرویس های دوره ای نخواهد داشت.
- *PLC*انرژی کمتری مصرف می کند.
- *PLC*ها برخلاف مدارات رله کنتاکتوری نویز الکتریکی و صوتی ایجاد نمی کند.
- استفاده از یک *PLC* منحصر به پروسه خاصی نیست . با تغییراتی در برنامه می توان به آسانی از آن برای کنترل پروسه های دیگر استفاده کرد.
- طراحی و اجرای مدارات کنترل و فرمان با استفاده از *PLC* بسیار سریع و آسان است.
- برای مدارات کنتاکتوری الگوریت و روش خاصی نداری اما در عیب یابی مدارات *PLC* به راحتی با تغییرات در نرم افزار و *SIMULATION*کردن آن می توان عیب یابی کرد.

۱-۱ نحوه کار*PLC*

در ابتدای راه اندازی مانند هر سیستم مبتنی بر پردازنده در *PLC* نیز برنامه سیستمی اجرا می گردد .پس از اجرای برنامه سیستمی و چک شدن سخت افزار ، در صورتی که شرایط لازم برای ورود به حالت اجرا *(RUN)* فراهم باشد، برنامه کاربر فرا خوانده می شود. برای اجرای برنامه کاربر ابتدا تمام ورودی های *PLC* بطور یک جا فرا خوانده می شود و در وضعیت آنها(صفر یا یک)در مکانی به نام تصویر ورودی (*Imput-Image-Area*) نوشته می شود *PLC* .در خلال اولین *Scan* برنامه از داده های تصویر ورودی استفاده می کند . توجه نمایید در صورتی که در طول اولین *Scan* تغییراتی در ورودی حاصل شود ای تغییرات تا *Scan*بعدی به مکان تصویر ورودیها منتقل نمی شود *PLC* .ضمن *Scan* برنامه کاربر نتایج حاصل را در مکانی بنام تصویر خروجی *(Output-Image-Area)* می نویسد

و بعد از اجرای کامل برنامه و در پایان نتایج را بطور یکجا به خروجی ها ارسال می کند .خواندن یک جای ورودی ها و ارسال یک جای خروجی ها صرفه جویی قابل توجهی در زمان دارد، زیرا خواندن یا نوشت با آدرس دهی یک به یک زمان زیادی را به خود اختصاص می دهد .از جمله مزایای دسترسی به مکانهای تصویر خروجی یا ورودی آن است که امکان *Set* یا *Reset* نمودن هر یک از بیت های ورودی یا خروجی را مستقل از وضعیت فیزیکی آنها فراهم می سازد و این کار مزیت بزرگی به هنگام عیب یابی یا آزمایش یک برنامه نوشته شده محسوب می شود .روش فوق در عین مزایایی که ذکر گردید مسئله ای به نام زمان پاسخ دهی برنامه *(Program Response Time)* را بوجود می آورد.زمان پاسخ دهی مدت زمانی است که طول می کشد تا *PLC* تمام برنامه کاربر را *Scan* کند و در این زمان تغییرات به وجود آمده در ورودیها وارد مکان تصویر ورودی نمی گردد و خروجیها به حالتی که در *Scan* قبلی بوده باقی می ماند این امر در فرآیندهایی با سرعت تغییرات زیاد مشکل ساز است مخصوصاً زمانی که برنامه کاربر طولانی بوده و مدت زیادی صرف *Scan* برنامه می گردد .همچنین گاهی ملاحظات ایمنی لازم می دارد که تغییرات آنی بعضی از ورودیها همواره مورد توجه قرار گیرد .که در اینصورت زمان پاسخ دهی ممک است مانع از ثبت به موقع این تغییرات شود .برای حل این مشکل در زبانهای برنامه نویسی دستورات خاصی گنجانده شده است .با توجه به سرعت بالای *PLC* های امروزی و کندی فرآیندی که توسط آن کنترل می گردند(سیستمهای االکترومکانیکی)زمان پاسخ دهی درشرایط عادی معمولاً مشکلی ایجاد نمی کند.



۲-۱ واحدهای تشکیل دهنده*PLC*

در *PLC* های کوچک پردازنده، حافظه های نیمه هادی، ماژول I/O و منبع تغذیه در یک واحد جای داده شده است .در *PLC*های بزرگتر پردازنده و حافظه در یک واحد، منبع تغذیه در واحد دوم و واسطه های I/O در واحد بعدی قرار دارد.

پردازنده

تمام پردازنده های رایانه ای به گونه ای طراحی شده اند که بتوانند محاسبات منطقی و حسابی را انجام دهند .این عملیات به وسیله ریزپردازنده *(Micriproccessor)* و از طریق به کار گیری دستورالعمل های متفاوت انجام می گیرد.

ماژول ورودی/خروجی

ماژول های ورودی به صورت الکترونیکی چهار کار اصلی انجام می دهند.

۱.حضور یا عدم حضور سیگنال الکتریکی در تمام ورودیها را بررسی می کند .این سیگنالهای ورودی وضعیت قطع یا وصل سوییچها، حسگرها و سایر عناصر در فرآیند کنترل را نمایش می دهند.

۲.این ماژول سیگنال مربوط به وصل بودن را از نظر الکتریکی به سطحی DC که توسط مدارات الکترونیکی ماژول I/O قابل استفاده باشد، تغییر میدهد.برای سیگنال ورودی قطع هیچ تبدیل سیگنالی صورت نمی گیردو نشان دهنده ی حالت قطع است.

۳.این ماژول جدا سازی الکترونیکی را با جدا کردن خروجی ماژول ورودی از ورودی اش به صورت الکترونیکی انجام میدهد

۴.این ماژول سیگنالی را که توسط CPU سیست PLC قابل تشخیص باشد را ایجاد می کند.

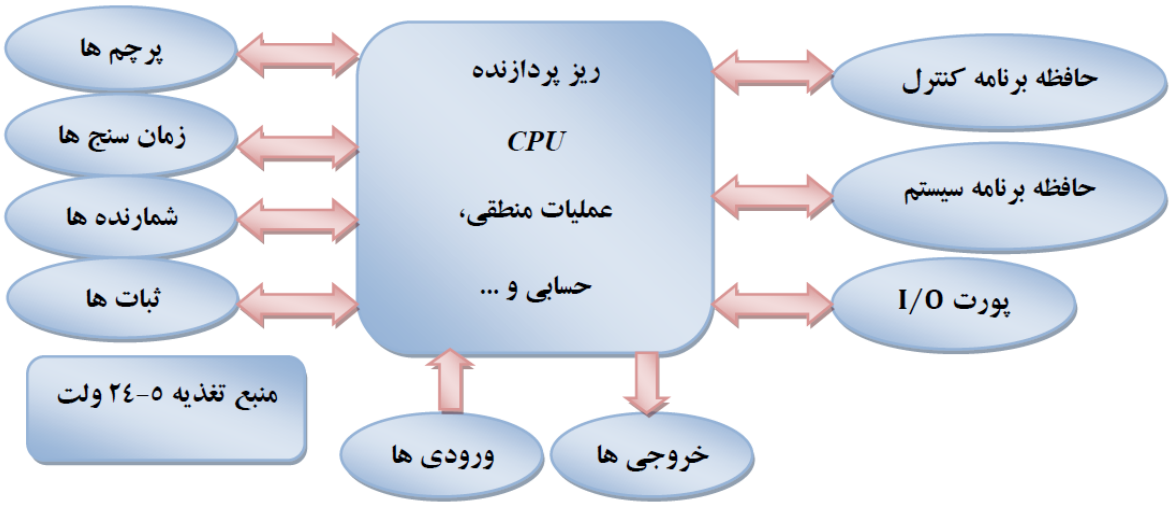
ماژول خروجی به گونه ای عکس ماژول ورودی عمل می کند .یک سیگنال DC که از CPU ارسال می گردد، در هر ماژول خروجی به سیگنال الکتریکی با سطح ولتاژ مناسب به صورت AC یا DC که توسط دستگاهها قابل استفاده باشد تبدیل می گردد.

منابع تغذیه

منبع انرژی که معمولاً استفاده می شود منبع جریان متناوب ۲۲۰ ولت با فرکانس ۵۰ الی ۶۰ هرتز می باشد .از آنجا که اغلب PLC با ولتاژهای ۵+ ، ۵- و ۲۴ ولت کار می کنند، لذا هر PLC باید مجهز به مدارهایی باشد که بتواند این تبدیل ولتاژها را انجام دهد.

۱-۳ ساختمان داخلی PLC

ساختمان داخلی هر PLC کم و بیش مانند ساختمان داخلی هر سیستم ریزپردازنده دیگر است.

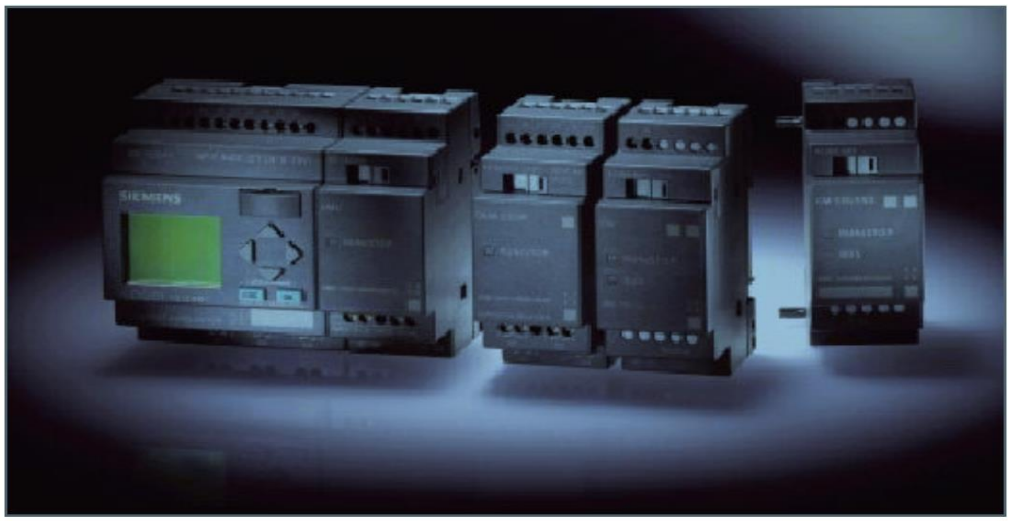


۱-۴ انواع سیستم PLC

به طور کلی از لحاظ ساختمان PLC ها به دو دسته کلی تقسیم می شوند.

1. *PLC* های کامپکت

این نوع ***PLC*** ها برای کنترل سیستمهای با حجم کوچک با تعداد ورودی و خروجی های محدود استفاده می گردند.به علت قابلیت محدودتر این نوع PLC *ها* برای کنترل همزمان تعداد کمتری از پروسه ها و یا کنترل دستگاههای مجزای صنعتی مورد استفاده قرار می گیرد. در این نوع PLC *ها* به همراه CPU تعدادی I/O عرضه می شود . در این نوع PLC *ها* فقط امکان افزایش ورودیها و خروجیها امکان پذیر است. این نوع PLC *ها* معمولاً تا سه ماژول توسعه را پشتیبانی می کنند .در زیر نمونه هایی از این PLC ها نمایش داده می شوند .



میکرو *PLC* ساخت شرکت زیمنس *(LOGO)* - ماژول اصلی به همراه ماژول های اضافی

PLC نوع *GM 7* از شرکت LG - ماژول اصلی به همراه ماژول های اضافی

۲. *PLC*های ماژولار *(Modular)*

در این نوع *PLC ها* هر قسمت از سخت افزار آن به صورت جداگانه ارائه می شود و کاربر باید بسته به نیازهای سخت افزاری پروسه خود آن را تهیه نموده و کنار یکدیگر قرار دهد . این نوع *PLC ها* برای کنترل سایت های کارخانجات بزرگ ا ستفاده میشود . اجزای این نوع *PLC ها* بر روی ریل قرار می گیرند، همچنین در صورت نیاز می توان این پروسه را گسترش داد .در این نوع *PLC ها* تا سه ریل دیگر را می توان به CPU اصلی متصل نمود .در زیر نمونه هایی از آن را مشاهده می کنید.

نمونه ای از *PLC* خانواده *CS1* از شرکت *OMRON*

PLC نوع *GM 7* از شرکتLG

۵-۱ روش و زبان برنامه نویسی PLC

۱. ***IL (Instruction List)*** *:* یک زبان سطح پایین و از زبان های قبلی *PLC* است که به صورت متنی می باشد.این زبان بیشتر شبیه زبان اسمبلرهای میکروپروسسور است.

۲. ***FBD (Function Block Diagram)*** *:* زبان گرافیکی است که قبلاً مورد استفاده قرار می گرفت .در *FBD برنامه* نویسی توسط یک سری بلوکهای پایه که در کنار هم قرار میگیرند انجام میشود.

۳. ***LD (Ladder Diagram)*** *:* روش گرافیکی است که بصورت دیاگرام نردبانی است ولی به صورت پیشرفته تر عرضه شد.در روش جدید LD و FBD میتوانند توام در برنامه بکار روند.

۴. ***ST (Structured Text)*** *:* ***ST*** یک زبان سطح بالا شبیه C یا پاسکال است و کاربردی عالی به ویژه در الگوریتم های پیچیده ریاضی دارد.

۵. ***SFC (Sequential Function Control)*** *:* در این روش برنامه نویسی به مراحلی که ترتیب الگوریتم کنترلی را نشان می دهد تقسیم می گردد و شامل *Step* های مختلف برنامه است.

ماژول های *PLC*

ماژول هایی که معمولا در پیکره بندی *PLC* استفاده می شوند عبارتند از:

۱. منبع تغذیه

۲. *CPU*

۳. ماژول های سیگنال (SM) : این ماژول ها ارتباط بین CPU و محیط خارج را فراهم می کند که دارای انواع زیر می باشد:

- ماژول ورودی دیجیتال (DI) : این ماژول ها به صورت ۲۴ ولت DC یا ۱۲۰ تا ۲۳۰ ولت AC کار می کند
- ماژول خروجی دیجیتال (DO) : این ماژول ها به صورت رله ای ، ترانزیستوری یا ترایاکی است.
- ماژول های ورودی آنالوگ (AI) :این ماژول هابه صورت ولتاژ (۰-۱۰) ولت ، جریان (۰-۲۰)یا (۴-۲۲) میلی آمپر
- ماژول های خروجی آنالوگ (AO) :این ماژول هابه صورت ولتاژ (۰-۱۰) ولت ، جریان (۰-۲۰)یا (۴-۲۲) میلی آمپر

۴.ماژول های واسطه (IM) : این ماژول برای ارتباط اطلاعات از سطوح دیگر در شبکه صنعتی به CPU کاربرد دارند.

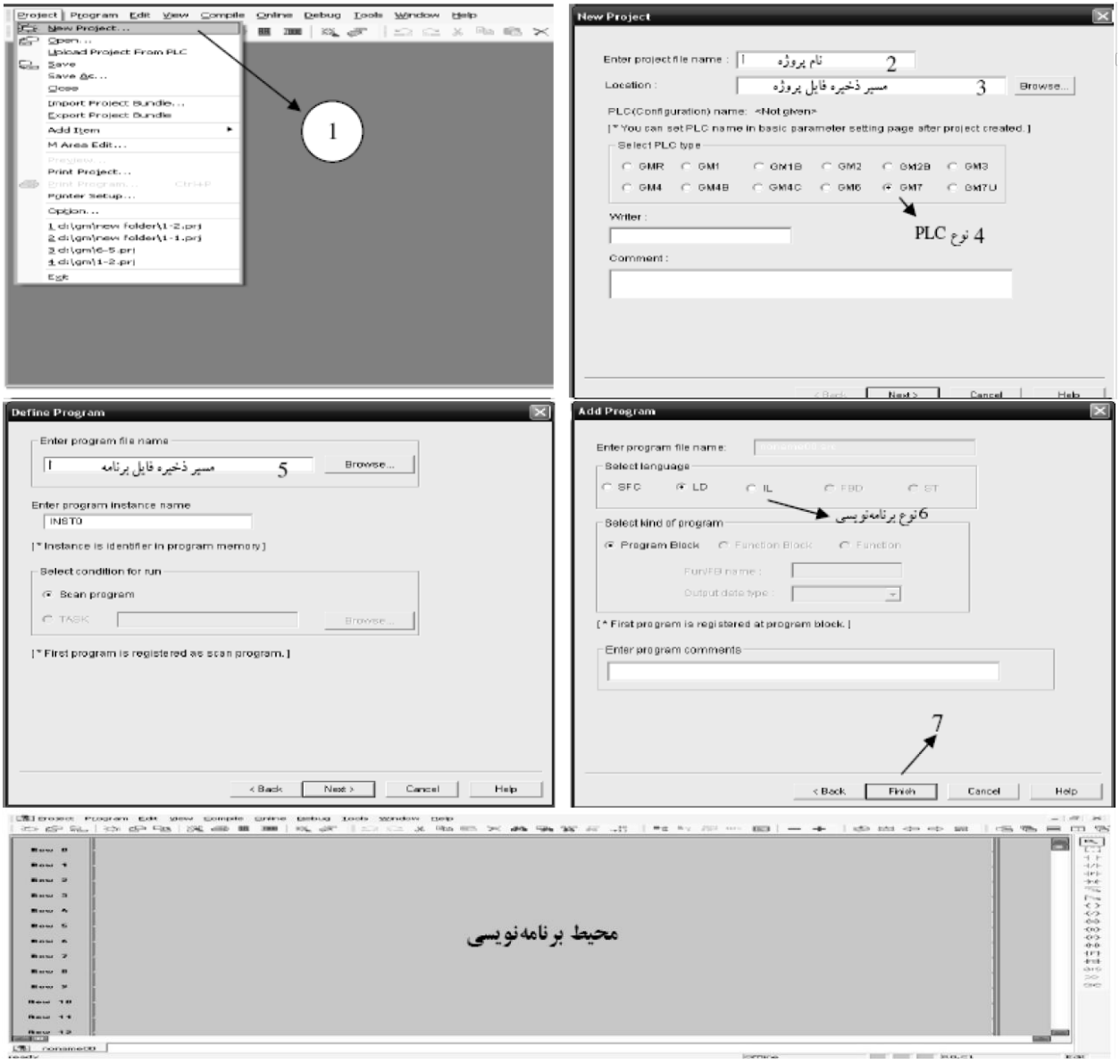
۵. ماژول های تابع (FM) : کارت هایی هستند که عملکرد ویژه ای را انجام می دهند .مانند:

۱. شمارنده سریع ۲. کنترل موقعیت و مکان یابی ۳. کنترل حلقه بسته

۶. پردازنده ارتباطی (CU) : برای ایجاد تسهیل ارتباط در شبکه های صنعتی استفاده می شود.

۲ آشنایی با نرم افزار GMWIN

تمام شرکت های سازنده PLC دارای یک نرم افزار مختص به همان PLC هستند. نرم افزار PLC مذکور **GMWIN است** . زبان مورد استفاده در این آزمایشگاه LD است .در زیر روند ایجاد یک پروژه را مشاهده می کنید.



۲-۱ انواع متغیر در زبان LD

متغیر به معنای چیزی است که اطلاعات آن تغییر می کند مانند یک ورودی، یک خروجی یا یک مکان از حافظه .برای تعریف متغیر به دو صورت عمل می کنیم.

۱.در این روش یک نام به متغیر مورد نظر اختصاص می دهیم . سپس در قسمت تنظیمات آدرس واقعی آن را مشخص می کنیم.

۲.در روش دوم به جای یک نام اختصاصی آدرس همان متغیر را به آن می دهیم . این روش به شرح زیر است.

٪ {موقعیت متغیر} {نوع داده متغیر} X1.X2.X3

٪ به معنای سیگنال ورودی یا خروجی (خارج از PLC) است

I٪ به معنای خواندن بیت های ارتباط با خارج است.

Q٪ به معنای نوشتن در بیت های ارتباط با خارج است.

استاندارد تعداد بیت های ورودی و خروجی برای یک کارت ۸ ،۱۶،۳۲،۶۴ می باشد.

نوع داده متغیر می تواند شامل :

X : به معنی آدرس دهی بیتی

B : به معنی آدرس دهی بایتی

W: به معنی آدرس دهی کلمه ای

M : به معنی دستیابی به ورودی خروجی های مخصوص مونیتورینگ است.

در قسمت X1 یک عدد نوشته می شود که نشانگر شماره PLC در شبکه ای از PLC هاست ، X2 نیز شماره اسلات که بیت یا بایت مورد نظر ما در آن قراردارد نوشته می شود و همچنین X3 شماره ورودی یا خروجی را مشخص می کند .

نکته : در قسمت های *X1* ، *X2* ، *X3* فقط می توان از ۲ تا ۹۹ را نوشت.

مثال:

IX 0.0.5 ٪ : به این معنا است آدرس ورودی که در بیس ۱ از اسلات اول بیت ۵ را بخوان.

QX 1.2.0 ٪ : به این معنا است آدرس خروجی که از بیس ۲ اسلات ۳ از بیت ۰ تا ۷ (یک بایت)را بخوان.

MB 0.3 ٪ : بیت شماره ۴ از بایت شماره ۱ حافظه داخلی.

MW 2.1٪ : بیت شماره ۳ از کلمه(دو بایتی) شماره ۲ حافظه داخلی.

۲-۲ آشنایی با بعضی از المانهای نرم افزار

خط عمودی سمت راست نرم افزار به منزله زمین بوده و دارای منطق صفر است و خط عمودی سمت چپ نرم افزار به منزله خط تغذیه و دارای منطق ۱ است .المانها همیشه بین این دو باس قرار می گیرند و باعث منتقل شدن منطق ۱ به سمت راست و روشن یا خاموش شدن یک وسیله می شود .ارتباط بین المانها به وسیله خطوط انتقال توان که خطوطی عمودی و هم افقی هستند صورت می گیرد .دو المان بسیار پر کاربرد در PLC کنتاکتها و کویل ها هستند .حال انواع این دو المان را شرح می دهیم.

۲-۲-۱ کنتاکت ها

کناکت ها عناصری هستند که هنگامی که فعال می شوند در خروجی آنها مقادیر منطقی فرق می کند .کنتاکت ها بر دو نوع است.

کنتاکت استاتیک (*Static Contact*)

این کنتاکت ها خود دو نوع هستند NO و NC . کنتاکت NO هنگامی که فعال می شود در خروجی مقدار ۱ منطقی را داری در غیر این صورت خروجی صفر منطقی است . اما کنتاکت NC درست برعکسNO است و زمانی که فعال شود در خروجی مقدار صفر منطقی را خواهی داشت در غیر این صورت در خروجی مقدار ۱ را داریم.

کنتاکت تشخیص لبه (*State Transition-Sensing Contact*)

از این کنتاکت ها در مواقعی استفاده می شود که کاری برای یک سیکل زمانی خاصی انجام گیرد .این کنتاکت ها خود دو نوع هستند.

کنتاکت تشخیص دهنده لبه مثبت (P)

اگر ورودی یک کنتاکت در اسک قبلی برنامه صفر بوده و اکنون ۱ است، خروجی ای کنتاکت ۱ می شود.

کنتاکت تشخیص دهنده لبه منفی (N)

اگر ورودی یک کنتاکت در اسک قبلی برنامه ۱ بوده و اکنون صفر است، خروجی ای کنتاکت ۱ می شود.

۲-۲-۲ کویل ها

کویل ها به منزله خروجی های یک برنامه در نظر گرفته می شوند، این کویل ها می توانند بوبین یک رله یا مستقیماً خود خروجی مورد نظر را تحریک کند .کویل دارای انواع متفاوتی هستند که در زیر شرح می دهیم.

کویل لحظه ای

کویل معمولی : اگر مقدار سمت راست ای کویل ۱ شود روشن خواهد شد.

کویل معکوس شده : این کویل برعکس کویل قبلی است یعنی اگر مقدار سمت راست آن ۱ شود خاموش خواهد شد.

کویل های قفل شونده

کویل ست کننده : اگر مقدار سمت راست آن ۱ شود این کویل روشن شده و دیگر صفر یا یک شدن باس سمت راست آن تاثیری در روشن یا خاموش شدن آن ندارد .ای کویل روش می ماند تا زمانی که به وسیله کویل ریست کننده خاموش شود.

کویل ریست کننده : اگر مقدار سمت راست آن ۱ شود این کویل خاموش شده و دیگر صفر یا یک شدن باس سمت راست آن تاثیری در روش یا خاموش شدن آن ندارد .این کویل خاموش می ماند تا زمانی که به وسیله کویل ست کننده روش شود .

کویل های تشخیص دهنده لبه

کویل تشخیص دهنده لبه مثبت : اگر ارزش سمت راست آن در اسکن قبلی برنامه صفر بوده و در اسکن جدید ۱ باشد این کویل برای یک مدت زمان معمولی روشن می شود.

کویل تشخیص دهنده لبه منفی : اگر ارزش سمت راست آن در اسکن قبلی برنامه ۱ بوده و در اسکن جدید صفر باشد این کویل برای یک مدت زمان معمولی روشن می شود.

با استفاده از کنتاکت ها می توان بسیاری از عملیات منطقی را شبیه سازی نمود.

نکته: باید متذکر شد که نباید یک خروجی را از دو نقطه متفاوت فرمان داد.

۲-۲-۳ فانکشن ها و فانکشن بلاک ها

ویژگی فانکشن ها:

- فاقد حافظه هستند.
- در یک فانکشن ها باید نوع داده های ورودی و خروجی یکی باشد.
- هر فانکشن تعدادی ورودی ولی همواره یک خروجی دارند.
- هر فانکشن مجهز به یک پایه EN و یک ENO است، که پایه EN حساس به سطح می باشد و شرط اینکه فانکشن عملیات را اجرا کند این است که نتیجه عمل منطقی سمت چپ آن یک باشد. هرگاه یک خطا در فانکشن رخ دهد پایه ENO صفر می شود.

ویژگی فانکشن بلاک ها:

- دارای حافظه هستند، یعنی نیازی نیست خروجی آن ها را به یک متغیر نسبت دهیم.
- یک فانکشن بلاک می تواند داده های ورودی و خروجی آن با یکدیگر متفاوت باشد.
- یک فانکشن بلاک می تواند تعدادی ورودی و خروجی داشته باشد.
- هر فانکشن بلاک مجهز به یک پایه ورودی است که حساس به لبه می باشد.

بعد از نوشتن برنامه نوبت به ارسال آن به PLC می رسد . برای انجام این عمل دکمه در نوار ابزار استفاده و یا از منوی Online گزینه *Connect+Write+Run+Monitor On* استفاده می کنیم . با زدن این دکمه ابتدا PLC شروع به عیب یابی برنامه می کند و اگر عیبی وجود داشت آن را با خط قرمز نشان می دهد . سپس با زدن دکمه OK ابتدا به PLC متصل می شود، برنامه را در PLC می ریزد، برنامه را اجرا می کند و سپس به صورت *Monitoring* برنامه را روی نرم فزار نشان می دهد .باید دقت شود در هنگام ارسال برنامه به PLC کلید ۳ حالته باید روی در وضعیت وسط قرار گیرد.

برای ساده فهم تر شدن برنامه می توان به هر یک از کنتاکت ها، کویل ها، بیت های حافظه و ... یک اسم اختصاص داد .با دابل کلیک کردن روی کنتاکت مورد نظر پنجره ای به نام *Variables* باز می شود .در این پنجره لیستی از متغیرهایی که نامی به آنها اختصاص داده شده است مشاهده می شود .بقیه مراحل انجام کار در شکل های زیر دیده می شود.

۳-۲ نحوه اتصال سنسور یا شاسی به ورودی PLC

معمولا سنسورهای نوری ، القایی و خازنی دارای سه سیم می باشد که سیم قهوه ای در این سه سنسور مثبت و سیم آبی منفی و مشترک آن است و در نهایت سیم مشکی خروجی است .

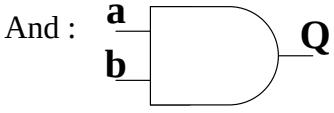
ابتدا پایه ۲۴ G را به COM متصل می کنیم ، سپس مثبت تغذیه را به سیم های قهوه ای سنسورها و صل می کنیم و بعد از آن منفی تغذیه را به سیم آبی و از آنجا به ۲۴+ PLC و صل می کنیم . بعد از مراحل بالا حالا باید خروجی این سنسورها را به PLC وصل می کنیم . برای این کار باید سیم مشکی را به آدرس مورد نظر به عنوان مثال %IX 0.0.0 وصل می کنیم .

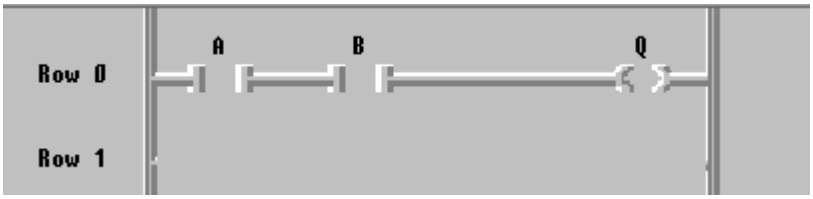
نحوه اتصال بوبین به خروجی PLC

ابتدا پایه ۲۴ G را به COM های خروجی متصل می کنیم ، سپس از ۲۴+ به سر بوبین یا خروجی و از سر دیگر خروجی نیز به آدرس مورد نظر وصل می کنیم .

شروع برنامه نویسی با استفاده از گیت های منطقی و جدول صحت

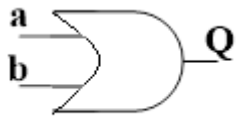
در ابتدا به بررسی گیت های منطقی AND و OR می پردازیم.





A	b	Q
0	0	0
0	1	0
1	0	0
1	1	1

Or :



A	b	Q
0	0	0
0	1	1
1	0	1
1	1	1

همانطور که مشاهده می شود با استفاده از جدول صحت نیز می توان برنامه ای را بصورت ابتکاری زیر نوشت، بدین صورت که :

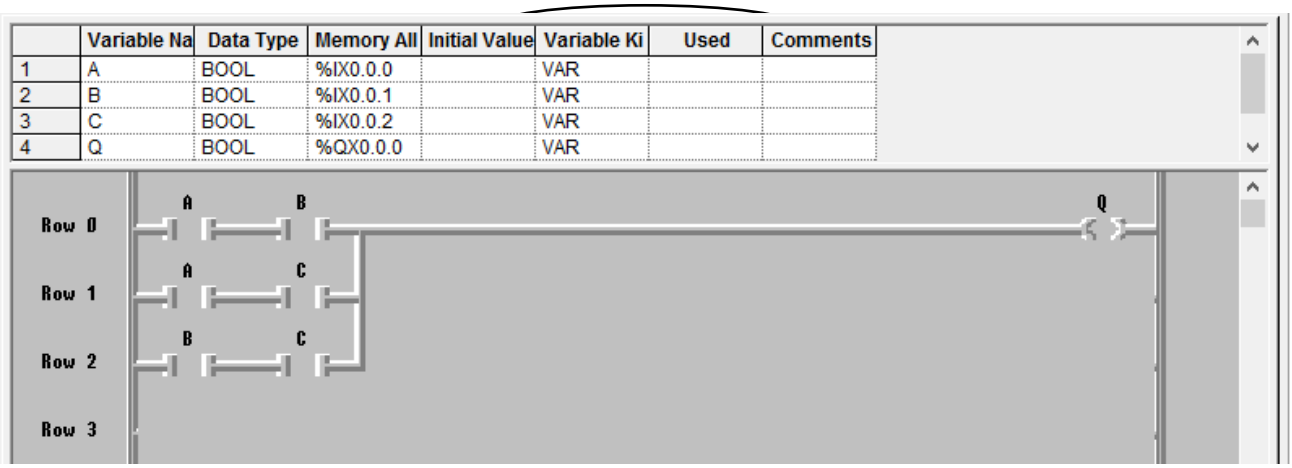
ابتدا در ردیف ها، خروجی هایی که یک شده اند را پیدا کرده، سپس در همان ردیف ها به جای ورودی هایی که صفر هستند تیغه بسته و به جای ورودی هایی که یک هستند، تیغه باز قرار خواهد گرفت و سپس این ورودی ها را به صورت سری به هم متصل کنید. در نهایت برای ردیف های بعدی که یک بوده اند نیز همین کار را نیز انجام دهید و تمامی این حالت ها را با هم موازی کنید.

نکته: بهتر آن است اگر تعداد خط ها زیاد بود ابتدا توسط جدول کارنو ساده سازی انجام شود و سپس با گیت های منطقی برنامه نردبانی آن را نوشت. همانطور که در بالا مشاهده شد، گیت AND یعنی دو کنتاکت با هم سری و گیت OR یعنی دو کنتاکت باهم موازی و تیغه بسته هم عملیات NOT را انجام می دهد.

مانند مثال زیر:

مثال) سه عدد سنسور در گوشه های یک سالن قرار گرفته است. هرگاه نصف بیشتر این سنسورها فعال شدند خروجی نیز فعال شده و فن روشن شود.

a	B	c	Q
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1



1	**0**	**0**	**0**
1	**0**	**1**	**1**
1	**1**	**0**	**1**
1	**1**	**1**	**1**

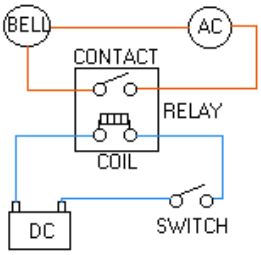
۴-۲ پرسش ها

۱- نحوه شبیه سازی گیت XOR را در نرم افزار مشخص نمائید.

۲- برنامه ای بنویسید که با تحریک دو ورودی از چهار ورودی خروجی فعال و با تحریک ورودی های دیگر خروجی غیر فعال شود؟

۳ مدارهای فرمان وشبیه سازی در GMWIN

هدف اصلی plc جایگزینی بجای رله ها است. میتوان یک رله را بعنوان یک کلید الکترومغناطیسی تصور کرد که ولتاژی را به یک پیچک(coil) اعمال میکند ویک میدان مغناطیسی ایجاد میکند در نهایت کنتاکتهای(contacts) رله را به سمت داخل میکشد و باعث ایجاد یک ارتباط میشود.این کنتاکتها می توانند بعنوان یک کلید درنظر گرفته شوند. در نتیجه بستن مدار، به جریان اجازه میدهند بین دو نقطه شارژ داشته باشد بعنوان مثال زیر میخواهیم یک زنگ را بطور ساده روشن کنیم هر زمان که کلید بسته شود ما سه قطعه را خواهیم داشت کلید، رله و زنگ. هر زمان که کلید بسته شود ما یک جریان به زنگ اعمال میکنیم که باعث به صدا در آمدن آن میشود .



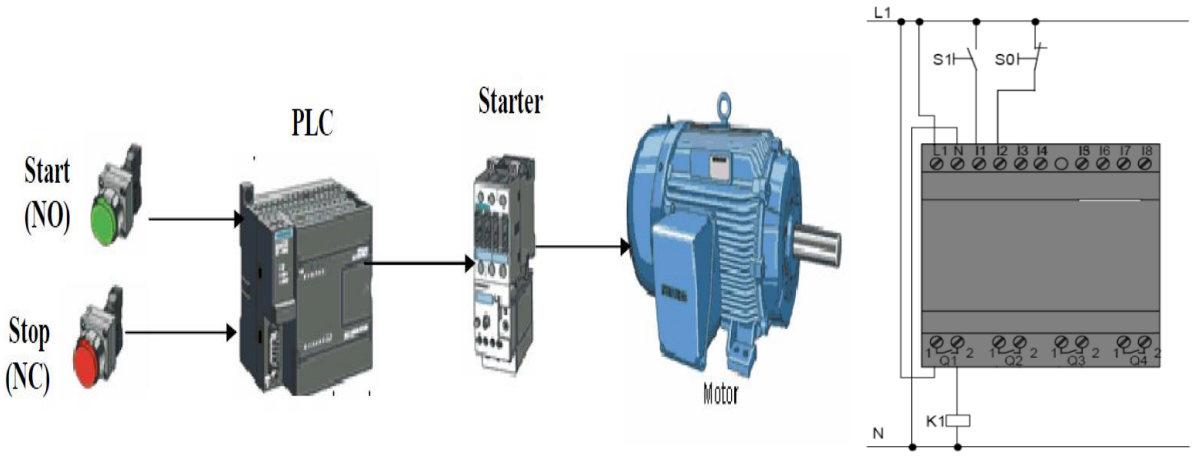
در شکل بالا دو مدار مجزا دیده می شود،مدار پایین قسمت dc و مدار بالا قسمت ac است. همانطور که می بینید یک رلهٔی dc برای کنترل یک مدار ac بکار رفته است هنگامی که کلید باز است جریانی از پیچک رله نمی گذارد به محض اینکه کلید بسته شود جریان از پیچک گذشته و یک میدان مغناطیسی ایجاد می کند که باعث می شود کنتاکت بسته شود حال جریان از زنگ عبورکرده و آژیر روشن خواهد شد.

نکته مهم: با آمدن PLC در صنعت و مدارات برق صنعتی فقط بخش فرمان به عهده این رله ها است و بخش سخت افزار نیز همچنان پابرجا خواهد بود اما دیگر نیازی به کنتاکتورهای کمکی نیست چون در بخش فرمان (برنامه PLC) تمامی کمکی ها قرار گرفته است.

اولین قدم در برنامه نویسی مدارات برق صنعتی، رسم مدار فرمان و تبدیل آن به نمودار نردبانی و در نهایت آدرس دهی سیگنال های خارجی(اعم از ورودی و خروجی ها) شامل شاسی ها و کنتاکتورهای متصل به موتور می باشد. حال به بررسی مدارات فرمان و برنامه نویسی بخش فرمان خواهیم پرداخت:

۳-۱ شرح آزمایش

برنامه ای بنویسید که با زدن شستی *Start* موتور ۱ روش و با زدن شستی *Stop* خاموش شود.



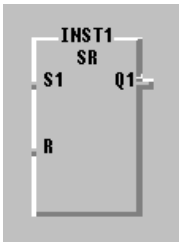
به ساختار برنامه بالا، خود نگه دار (کنتاکتور خروجی موتور موازی با شاسی) گفته می شود. عملکرد این کنتاکت به این صورت است هنگامی که شاسی *Start* زده شد موتور ۱ روشن می شود و هنگامی که شاسی *Start* قطع شد از کنتاکتور موتور ۱ استفاده کرده و از خاموش شدن آن جلوگیری می شود . برنامه فوق را می توان به کمک کویل های *Set* و *Reset* به راحتی مانند زیر نوشت.

با استفاده از کویل های *Set* و *Reset*

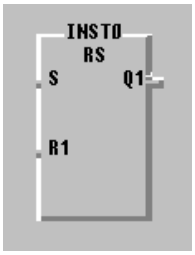
فانکشن بلاک های SR و RS:

عملکرد این فانکشن بلاک ها همانند کویل های set و reset می با شد با این تفاوت که در بلاک SR اولویت با Set ا ست و در بلاک RS اولویت با Reset است.

بلاک SR:



بلاک RS:



همچنین می توان برنامه قبل را با استفاده از فانکشن های SR یا RS نیز می توان نوشت:

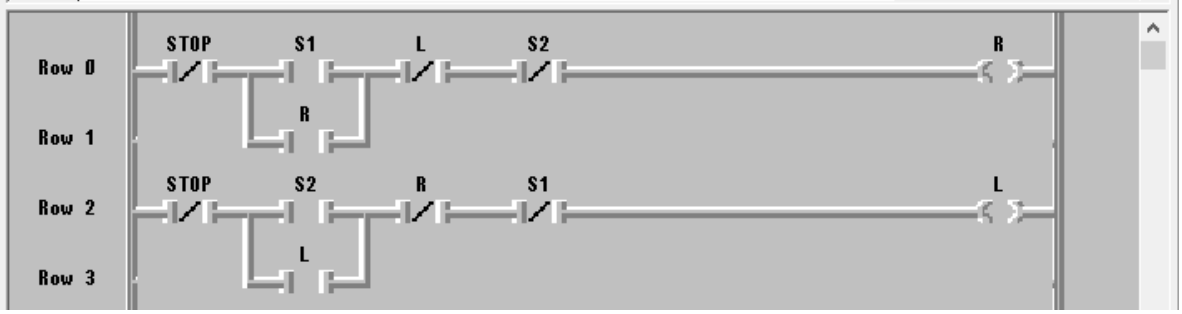
مداری را طراحی کنید که با زدن شستی اول موتور۱ روشن و با زدن شستی دوم موتور۲ روشن شود (اولویت روشن شدن مهم است و باید ابتدا موتور۱ روشن شود) و با زدن Stop هردو موتور خاموش شوند.

در ادامه مداری طرح نمائید که با زدن شستی اول موتور۱ روشن و با زدن شستی دوم موتور اول متوقف و موتور۲ روشن شود (اولویت روشن شدن مهم است و باید ابتدا موتور۱ روشن شود) و با زدن Stop هردو موتور خاموش شوند.
برای راه اندازی موتورها روشهای مختلفی وجود دارد که بنا به نیاز و نوع موتور بایستی مدار فرمان مطابق آن روش طراحی و اجرا گردد. حال به بررسی مدارات کاربردی دیگر در مدارات فرمان می پردازیم:

مثال) مدار **چپگرد-راستگردی** را به گونه ای طراحی کنید که: با زدن شستی S_1 موتور به حالت راستگرد شروع به کار کند و با زدن شستی S_2 موتوربه حالت چپگرد به کار خود ادامه دهد و با زدن شستی Stop موتور در هر حالتی که با شد، خاموش شود. همچنین هرگاه S_1 و S_2 با هم تحریک شدند موتور به حالت قطع برود.(حفاظت الکتریکی)

همانطور که می دانید این مدار کاربردهای بسیار زیادی در مدارات برق صنعتی دارد بعنوان مثال در آسانسورها،بالابرها،پله برقی و...در عمل تغییر جهت گردش موتور با عوض دو فاز امکان پذیر است پس دو کنتاکتور خروجی R وL را جهت تغییر دو فاز در نظر بگیرید که با فرمان دو شاسی دستور می گیرند.اما دقت داشته باشید که در این مسئله، حفاظت الکتریکی ناشی از تحریک همزمان دو شاسی و جلوگیری از اتصال کوتاه را در نظر بگیرید که این امر توسط استفاده از شاسی های دوبل امکان پذیر است. در تمرین بعد حفاظت مکانیکی نیز در نظر گرفته شده است تا شفت رتور در هنگام تغییر جهت گردش آسیب نبیند.

	Variable Na	Data Type	Memory All	Initial Value	Variable Ki	Used	Comments
1	L	BOOL	%QX0.0.1		VAR	*	
2	R	BOOL	%QX0.0.0		VAR	*	
3	S1	BOOL	%IX0.0.1		VAR	*	
4	S2	BOOL	%IX0.0.2		VAR	*	
5	STOP	BOOL	%IX0.0.0		VAR	*	



مدار پرکاربرد دیگر مدار ستاره مثلث است. اصلی ترین هدف مدار این است که جریان راه اندازی در حالت ستاره حدود سه برابر کمتر از حالت مثلث اســت پس در نهایت در حالت مثلث بیشــترین بار را می توان انداخت و به همین خاطر بعد از راه اندازی به حالت مثلث می بریم.

برنامه ای بنویسید که با زدن شاسی s1 موتور با کنتاکتور ستاره و به همراه کنتاکتور اصلی (مشترک) روشن شود با زدن شاسی s2 کنتاکتور ستاره قطع و کنتاکتور مثلث فعال شود(اولویت با ستاره)بازدن شاسی STOP موتور در هر حالتی که بود متوقف گردد.

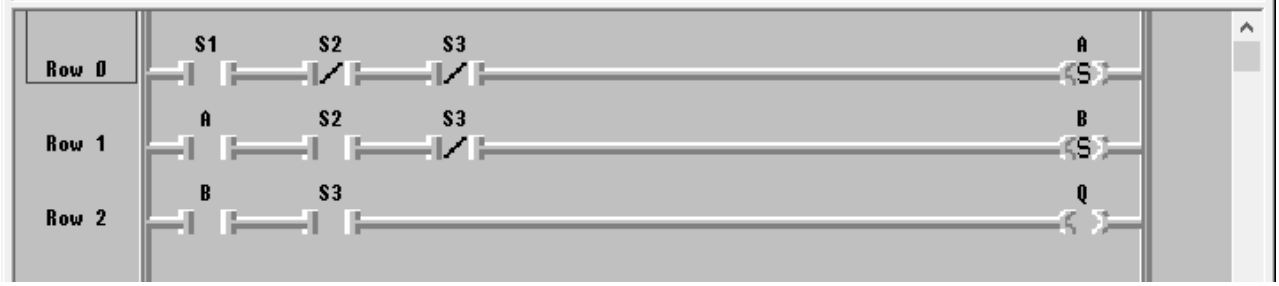
برنامه ای بنویسید که با زدن شاسی s1 موتور با کنتاکتور راستگرد و به همراه کنتاکتور ستاره روشن شود و همینطور با زدن شاسی s2 موتور با کنتاکتور چپگرد و به همراه کنتاکتور ستاره روشن شود با زدن شاسی s3 کنتاکتور ستاره قطع و کنتاکتور مثلث فعال شــود(اولویت با ســتاره)بازدن شــاســی STOP موتور در هر حالتی که بود متوقف گردد.(تغییر جهت گردش با زدن STOP امکان پذیر است.)

۴ ذخیره سازی اطلاعات و ساختمان داده ها

ثبات هایی هستند که برای ذخیره کردن ساده ی اطلاعات تعیین شده اند آنها معمولا برای ذخیره موقتی ریاضیات و اطلاعات در حال دستکاری به کار می روند.بدین منظور بعد از قرار دادن یک نام برای این متغیرها در پنجره دوم باز شده به جای آدرس دهی از گزینه Auto استفاده می کنیم.

مثال) برنامه ای بنویسید که هرگاه سه عدد سنسور به ترتیب اینکه اول سنسور یک، بعد سنسور دو و بعد از آن سنسور سه فعال شد، خروجی فعال شود.(ترتیب فعال شدن خیلی مهم است،اگر رعایت نشد خروجی نیز فعال نشود)

	Variable Na	Data Type	Memory All	Initial Value	Variable Ki	Used	Comments
1	A	BOOL	<Auto>		VAR	*	
2	B	BOOL	<Auto>		VAR	*	
3	Q	BOOL	%QX0.0.0		VAR	*	
4	S1	BOOL	%IX0.0.0		VAR	*	
5	S2	BOOL	%IX0.0.1		VAR	*	
6	S3	BOOL	%IX0.0.2		VAR	*	



در این مثال متغیرهای A و B بصورت کمکی (فلگ) در نظر گرفته شده اند، که نتیجه هر مرحله در هر کدام از آنها ذخیره شده است و Q نیز خروجی اصلی برنامه می باشد.

تمرین کلاسی) یک میز مسابقه چهار نفره طراحی کنید که هر کدام از شرکت کنندگان که زودتر شستی را زد چراغ او روشن شود و چراغ بقیه خاموش بماند. همچنین یک شستی برای مجری به منظور اجازه شروع مسابقه و یک شستی نیز برای ریست کردن هر مرحله از مسابقه در نظر گرفته شود.

ساختمان داده ها در PLC:

داده ها و اعداد در PLC دارای تنوع زیادی هستند .علت آن که همه داده با یک فرمت مشابه تعریف نمی شوند صرفه جویی در حافظه PLC ا ست.به ای معنی که با ای تکنیک می توان متغییرهای متنوع با تعداد بیت های مورد نیاز تعریف کرد .در زبانLD بلوک تبدیل داده ها به ه دیگر برای سهل کردن عملیات ریاضی وجود دارد.

NU	Reserved Word	Data Type	Size(bit)	Range
1	SINT	Sort Integer	8	-128~127
2	INT	Integer	16	-32768~32767
3	DINT	Double Integer	32	-2147483648~2147483647
4	LINT	Long Integer	64	-2^{63}~$2^{63}-1$
5	USINT	Unsigned Short Integer	8	0~255
6	UINT	Unsigned Integer	16	0~65535
7	UDINT	Unsigned Double integer	32	0~4294967295
8	ULINT	Unsigned Long Integer	64	0~$2^{64}-1$
9	REAL	Real Numbers	32	-3.402823E38~-1.401298E-45 1.401298E-45~3.402823E38
10	LREAL	Long Real Integer	64	-1.7976931E308~-4.9406564E-324 4.9406564E-324~1.7976931E308
11	TIME	Duration	32	T#0S~T#49D17H2M47S295MS
12	DATE	Date	16	D#1684-01-01~D#2163-6-6
13	TIME_OF_DAY	Time if Day	32	TOD#00:00:00~TOD#23:59:59:999
14	DATE_AND_TIME	Date and Time	64	DT#1984-01-01-00:00:00 ~ DT#2163-12-31-23:59:59:999
15	STRING	Chatacter String	30*8	Limited within 30 letters
16	BOOL	Boolean	1	0,1
17	BYTE	Bit String of Length 8	8	16#0~16#FF
18	WORD	Bit String of Length 16	16	16#0~16#FFFF
19	DWORD	Bit String of Length 32	32	16#0~16#FFFFFFFF
20	LWORD	Bit String of Length 64	64	16#0~16#FFFFFFFFFFFFFFFF

برای مثال در داده ازنوع Time:

Exp for Time: T#2D3H25M30S25MS

منظور از ...#T اینست که داده از نوع زمان برابراست با : ۲ روز ، سه ساعت ،۲۵ دقیقه، ۳۰ ثانیه و ۲۵ میلی ثانیه

چند نکته مهم:

- اعداد منفی به صورت متم ۲ ذخیره می شوند.
- هر عملی چه منطقی باشد چه محاسباتی فقط بر روی متغیر های هم نوع انجام می شود.
- بیت های داده از نوع بایت دارای ارزش مکانی نیستند.
- برای اینکه بتوان مقدار عددی ذخیره شده در متغیر Byte را خواند، باید آن را به داده ای از نوع usint یا sint تبدیل نمود که با استفاده از بلاک های زیر انجام می پذیرد:

Byte to usintیاByte to sint

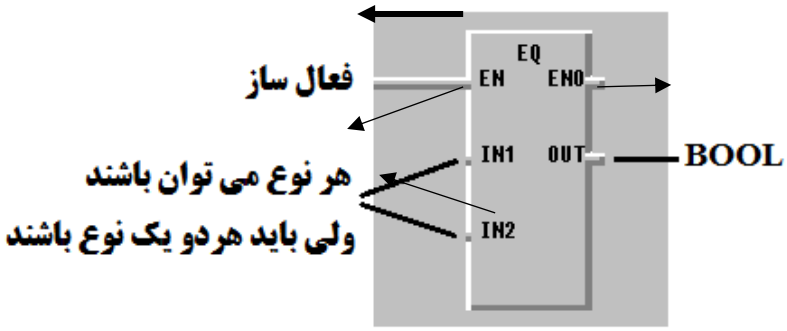
- بالعکس اگر بخواهید داده ای را به صورت بیتی، & یا or کنیم حتماً باید آن را به Byte تبدیل کرد که با استفاده از بلاک های زیر انجام می پذیرد:

sint to Byte یا **usint to Byte**

۵ عملگرهای مقایسه ای در GMWIN

فانکشن های هستند که عملیات مقایسه بین دو داده را انجام می دهند و اگر شرط برقرار بود، خروجی فانکشن یک می شود :

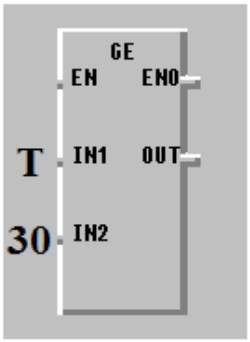
۱) EQ= تساوی



۲) NE= نامساوی

مانند فانکشن EQ است، فقط ورودی ها اگر نامساوی بودن خروجی یک می شود.

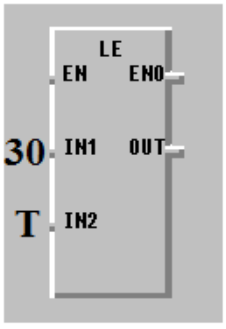
۳) GE= بزرگتر یا مساوی از



اگر $T \geq 30$ بود، خروجی فعال می شود.

۴) GT= بزرگتر از

۵) LE= کوچکتر یا مساوی از



اگر $T \leq 30$ بود، خروجی فعال می شود.

۶) LT= کوچکتر از

نکته: ورودی تمام داده های بالا ازنوع داده های مجاز می باشدو داده های ورودی باید ازیک نوع باشد.

عملگر های ریاضی

۱- **جمع(ADD)**: بین ۲ تا ۸ پایه ورودی می تواند داشته باشد.

۲- **ضرب(MUL)**: بین ۲ تا ۸ پایه ورودی می تواند داشته باشد.

۳- **تقسیم(DIV)**: فقط ۲پایه ورودی دارد و خارج قسمت اعداد را تعیین می کند.

۴- **تفریق(SUB)**: فقط و فقط ۲پایه ورودی دارد.

۵- **باقیمانده(MOD)**: فقط ۲پایه ورودی دارد و باقیمانده اعداد را تعیین می کند.

۶- **قدر مطلق(ABS)**: فقط یک ورودی دارد و قدر مطلق عدد را تعیین می کند.

بعنوان مثال: در عملگر باقیمانده، ورودی اول(IN1) بر ورودی دوم(IN2) تقسیم شده و باقیمانده آن در خروجی ذخیره می گردد. دقت داشته باشید خروجی این فانکشن ها برخلاف عمگرهای مقایسه ای از نوع ورودی ها یعنی عدد می باشد.

عملگرهای منطقی

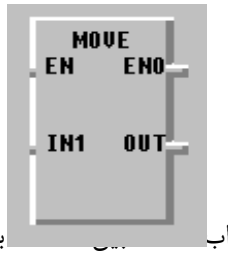
۱- **AND:**بین ۲ تا ۸ پایه ورودی می تواند داشته باشد.

۲- **OR:**بین ۲ تا ۸ پایه ورودی می تواند داشته باشد.

۳- **XOR:**بین ۲ تا ۸ پایه ورودی می تواند داشته باشد.

۴- **NOT:**یک ورودی دارد.

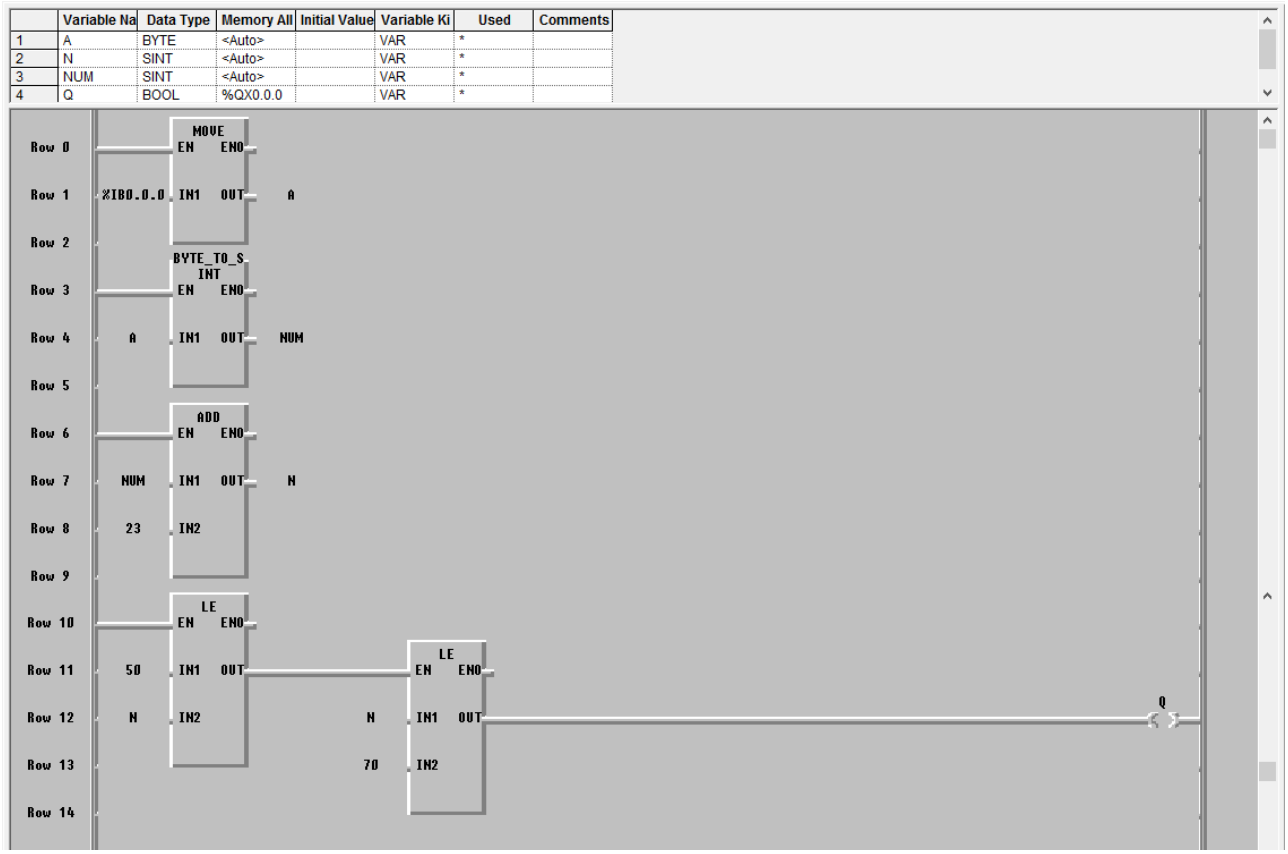
فانکشن MOVE



مقدار ورودی را در خروجی خود کپی می کند.

از این فانکشن در دریافت عدد دودویی از بایت ورودی PLC و همچنین نمایش عدد دودویی در بایت خروجیPLC استفاده می شود.

مثال) برنامه ای بنویسید که عددی را از بایت اول PLCگرفته، با عدد ۲۳ جمع نموده و اگر جواب بصورت بود چراغ سیگنالی روشن شود.



تمرین کلاسی) برنامه ای بنویسید که عددی 4بیتی از بایت اولPLCدریافت کند و چنانچه مضربی از ۵ بود چراغ سیگنالی روشن شود.

تمرین کلاسی) برنامه ای بنویسید که کنترل دمای یک کوره به صورت زیر انجام پذیرد:

If $T \leq 30 \rightarrow FAN_1 = on$

If $30 < T \leq 50 \rightarrow FAN_2 = on$

If $T > 50 \rightarrow FAN_3 = on$

نکته: دما توسط کاربر از Byte دوم PLC داده شود.

دستور SC:

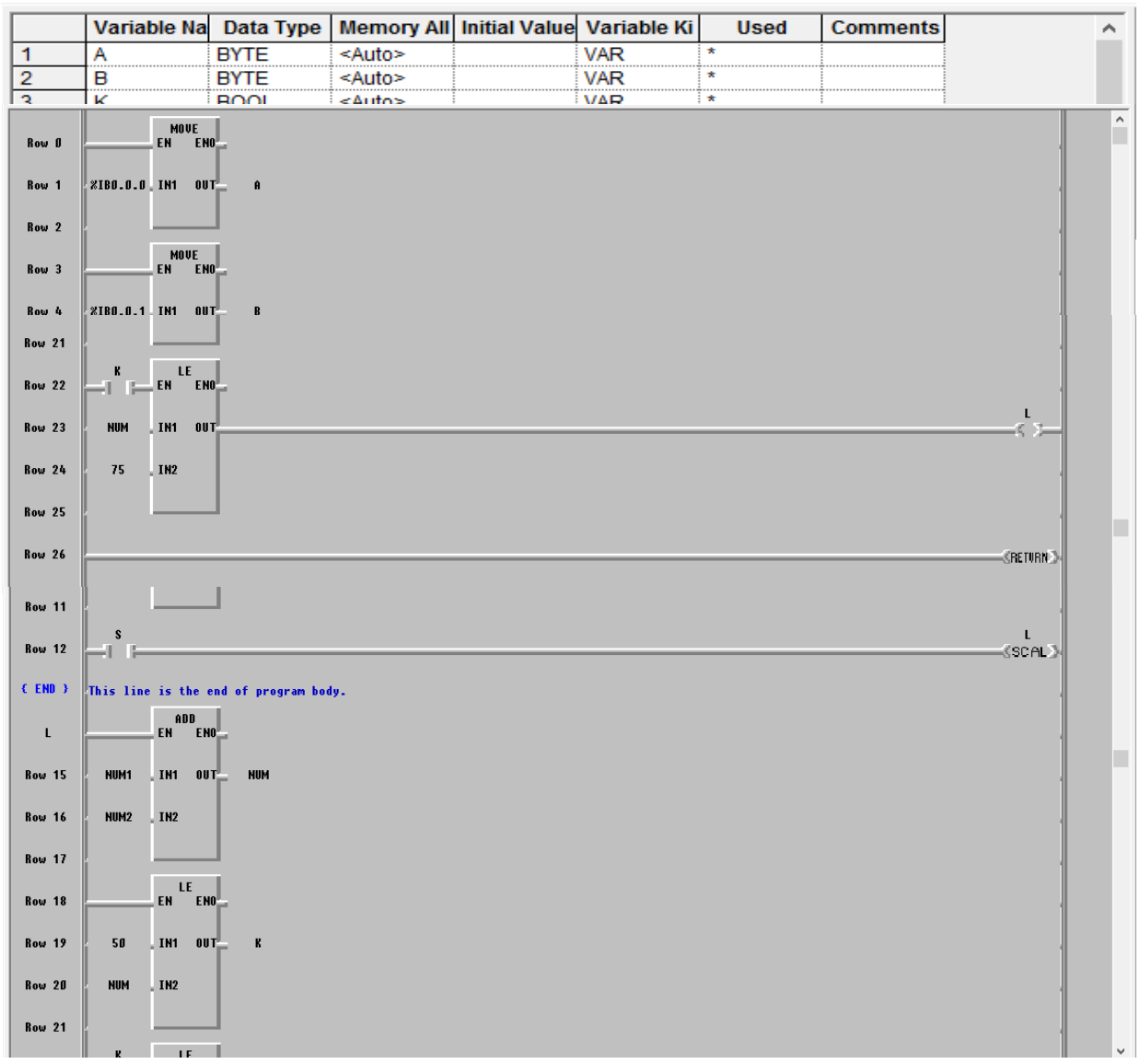
این د ستور برنامه ای را که جزء برنامه ا صلی نی ست که به آن زیر برنامه گفته می شود را مانند یک تابع فراخوانی می کند که برای قرار دادن یک زیر برنامه SC باید مراحل زیر را طی نمود:

۱) تعیین پایان برنامه اصلی با دوبار کلیک کردن برروی سطر آخر و انتخاب گزینه (The end of program body) از پنجره باز شده.

۲) قرار دادن یک برچسب یا Labelبرای شروع زیر برنامه SC.

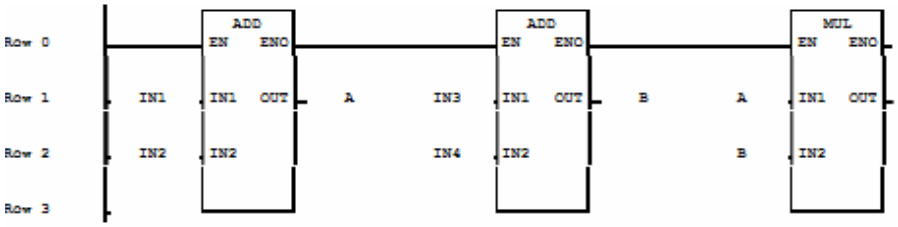
۳) قرار دادن (Return) یا بازگشت در پایان زیر برنامه.

مثال) برنامه ای بنویسید که دو عدد را گرفته و زمانی که یک کلید فشرده شد با فراخوانی یک SC آنها را باهم جمع نموده



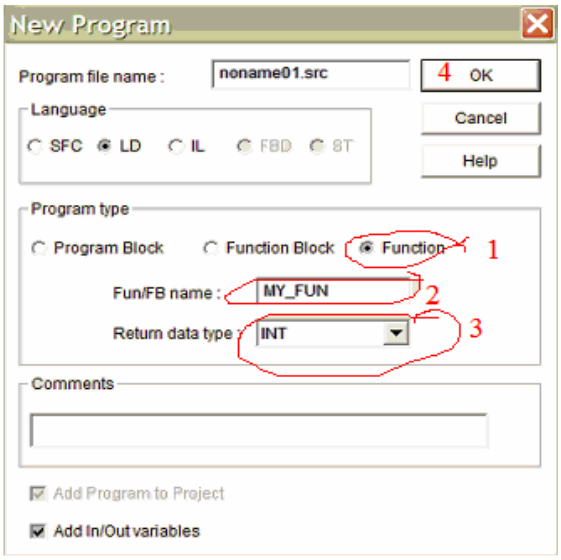
۶ فانکشن و فانکشن بلاک ها در GMWIN

تابع ریاضی مقابل را در نظر بگیرید: Out= (IN1+IN2)*(IN3+IN4)

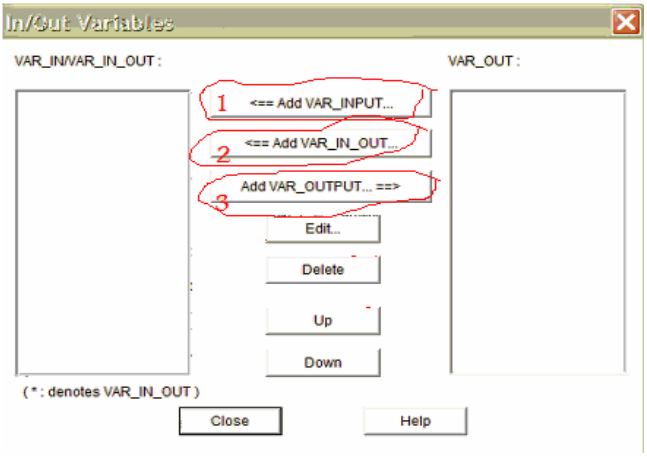


اگر بخواهید آن را بصورت فانکشن بلاک داشته باشید بصورت زیر عمل کنید:

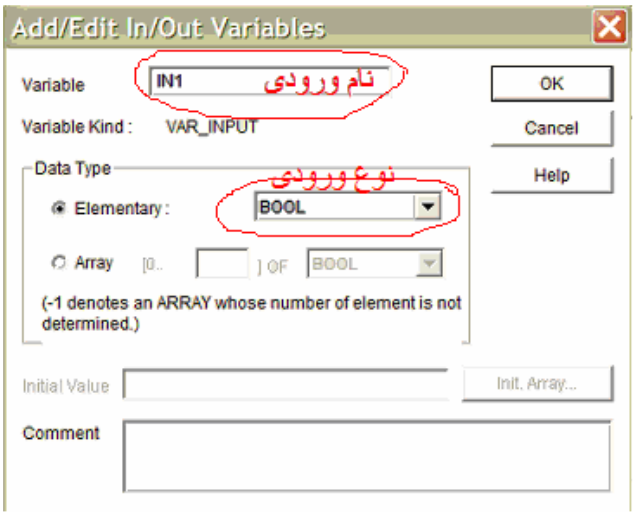
۱- در پنجره باز شده نرم افزار کلید ctrl+N رامیزنیم تاپنجره زیر ظاهرشود.



۲- بعد از انجام مراحل بالا و زدن OK پنجره زیر ظاهر می شود:



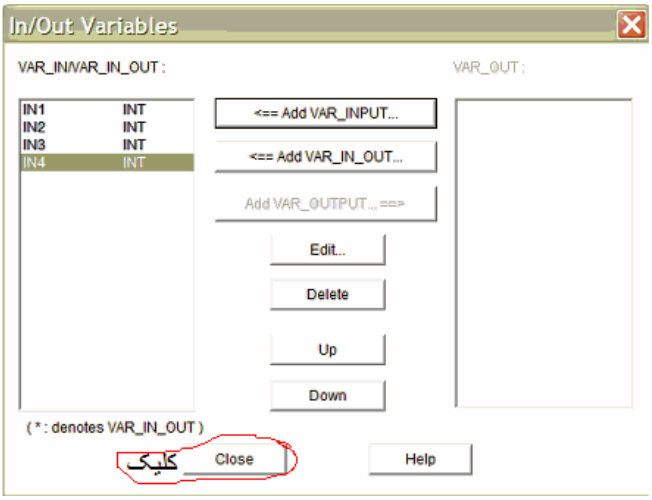
۳- در پنجره بالا ورودی و خروجی ها را به ترتیب وارد می کنیم.بعنوان مثال با کلیک برروی عنوان ورودی ها و انتخاب گزینه اول پنجره زیر باز می شود:



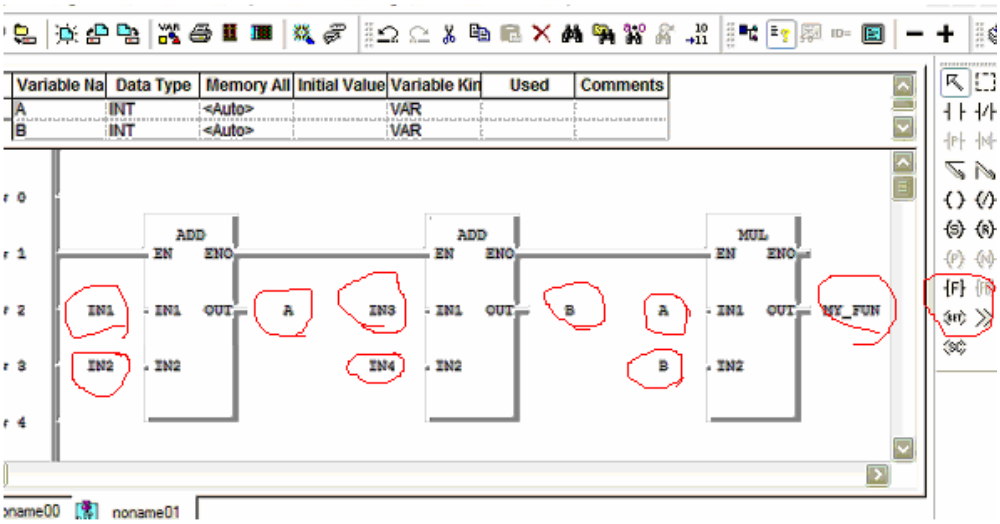
۴- بعداز انتخاب نام و نوع ورودی بر روی OK کلیک کنید تا در پنجره قبل دیده شود.

بعد از انخاب ورودی ها، خروجی ها را نیز بصورت قبل انتخاب کنید،با این تفاوت که ای بار بر روی عنوان سوم پنجره اول کلیک کنید.(اگر خروجی های بیشتری نیاز داشتید برروی عنوان دوم کلیک کنید،که این خروجی ها در قسمت ورودی ها ودر سمت چپ و با رنگ دیگری نشان داده می شود)

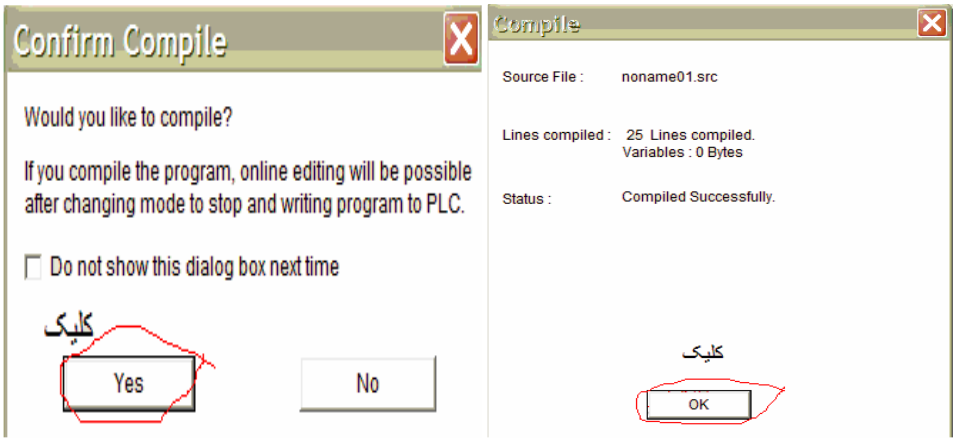
در شکل زیر تمام ورودی و خروجی ها نشان داده شده است:

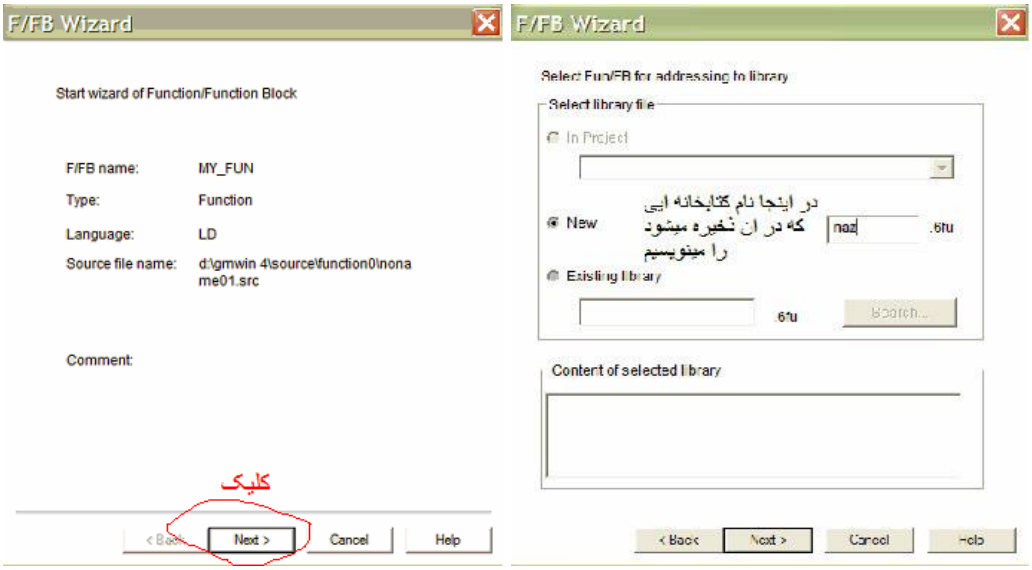


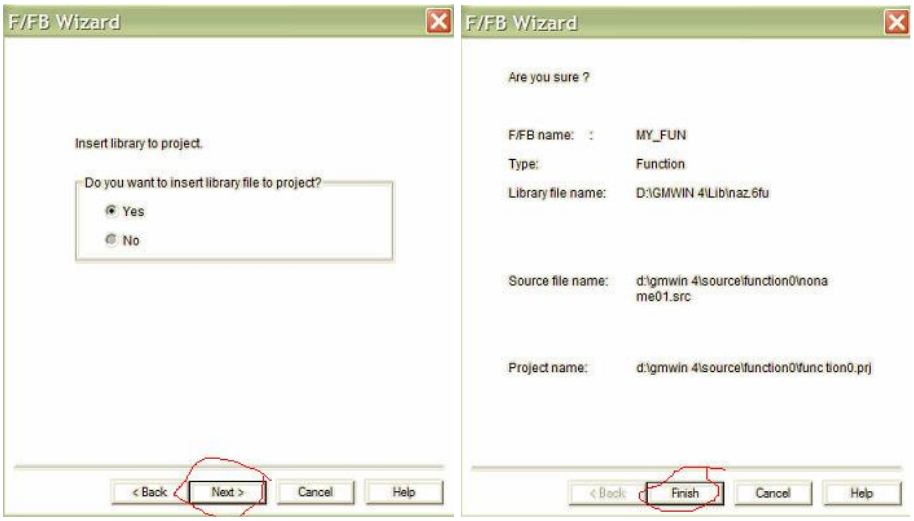
حال به پنجره کار بروید و مدار را طراحی نمایید و ورودی ها را در جای خود قرار دهید و خروجی را در My_fun قرار دهید:



و بعد از انجام مراحل بالا برنامه را compile کنید و بعد از کلیک بر روی عناوین مشخص شده در پنجره های پایین کار به اتمام می رسد:





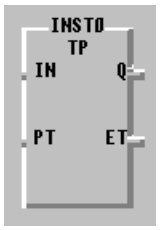


با این روش درطراحی یک برنامه دیگر لازم نیست از کلی فانکشن استفاده نمایید و آن را بصورت یک فانکشن جدا در کتابخانه خواهید داشت.

۷ تایمرها در GMWIN

هرگاه نیاز با شد فا صله زمانی بین دو رویداد را اندازه گرفته و یا عمل خا صی را در مدت زمان م شخص انجام دهیم از تایمرها استفاده می کنیم.

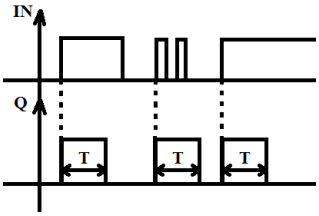
۱) **تایمر ضربه(TP)**:در این تایمر زمانی ک هدر پایه IN تغییر وضعیت ایجاد شود خروجی یک می شود و بعد از گذشت زمان تنظیم شده در PT خروجی صفر می شود .در این تایمر اگر در حین کار تایمر، ورودی قطع و وصل شود خللی در کار تایمر ایجاد نمی کند.



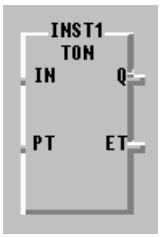
IN: پایه فعال ساز

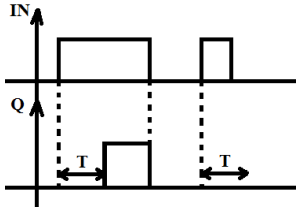
PT: پایه برای زمان تعیین شده توسط کاربر جهت روشن بودن تایمر

ET: پایه شمارنده



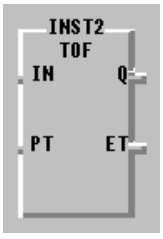
۲) **تایمر تأخیر در وصل(TON)**:این تایمرحساس به لبه بالا رونده است و با تحریک پایه IN ، تایمر شروع به شمارش میکندو بعد از گذشت زمان تعیین شده،خروجی را فعال میکند امابا قطع تحریک،خروجی نیز قطع می شود پس خروجی وابسته به ورودی است. چنانچه تحریک پایه فعال ساز زودتر از

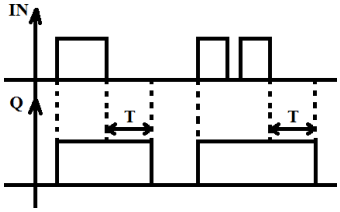




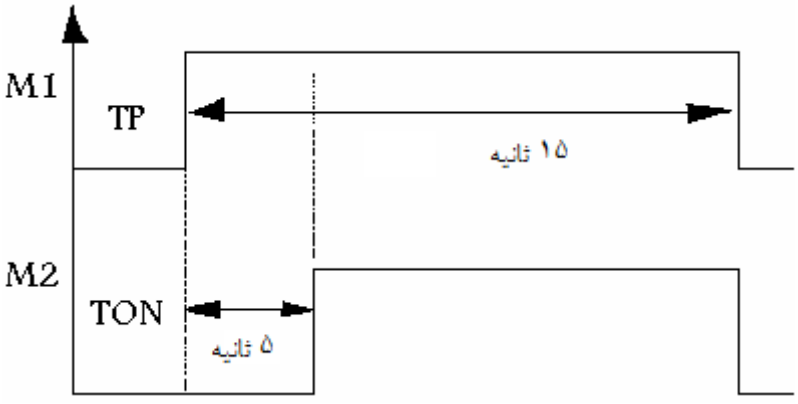
زمان تعیین شده تمام شود، خروجی فعال نمی شود.

۳) **تایمر تأخیر در قطع(TOF)**: در این تایمر با فعال شدن پایه ورودی IN خروجی Q به همزمان فعال می شود . زمانی که ورودی IN غیر فعال شد خروجی همچنان فعال باقی می ماند و ET شروع به شمارش می کند و درست زمانی ET به PT رسید خروجی غیر فعال می شود.

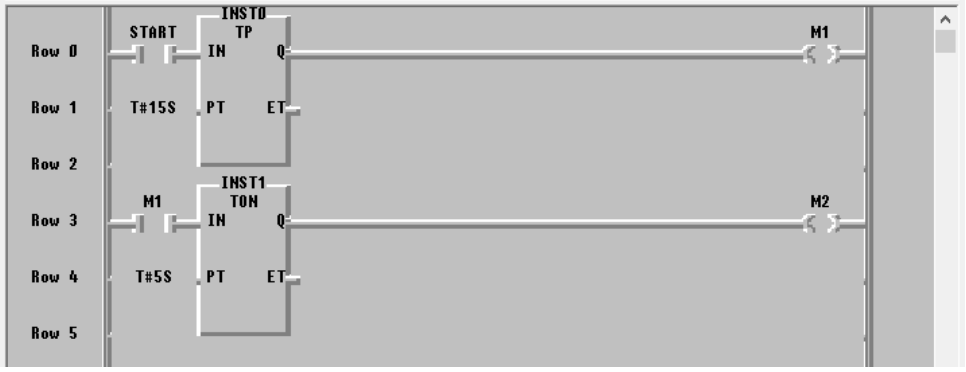




مثال) مداری طراحی کنید که با زدن شستی استارت موتور اول به مدت ۱۵ثانیه روشن بماند و سپس متوقف شود موتور دوم ۵ ثانیه دیرتر از موتور اول روشن شده و همزمان با موتور اول خاموش شود.



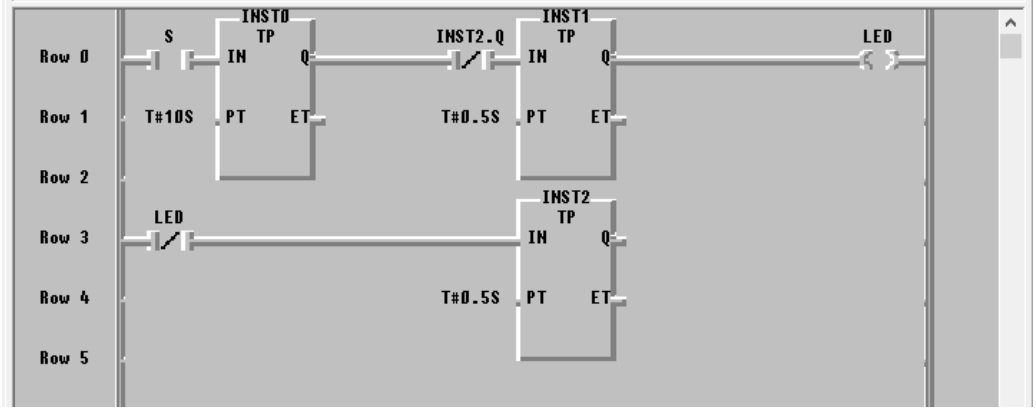
	Variable Na	Data Type	Memory All	Initial Value	Variable Ki	Used	Comments
1	INST0	FB Instanc	<Auto>		VAR	*	
2	INST1	FB Instanc	<Auto>		VAR	*	
3	M1	BOOL	%QX0.0.0		VAR	*	
4	M2	BOOL	%QX0.0.1		VAR	*	
5	START	BOOL	%IX0.0.0		VAR	*	



تمرین کلاسی) مداری طراحی کنید که با زدن کلید S_1بعد از ۱۰ثانیه موتور اول و بعد از ۲۵ ثانیه موتور دوم و سوم روشن شوند و با قطع کلید S_1 موتور اول خاموش و بعد از ۱۰ثانیه موتور دوم خاموش شود و۳۰ثانیه بعد از قطع موتور دوم،موتور سوم نیز خاموش شود.

مثال) برنامه ای بنویســـید که با زدن کلید به مدت ۱۰ثانیه یک LEDبه طور چشـــمک زن با فرکانس ۱هرتز و چرخه کار ۵۰٪ کار کند.

	Variable Na	Data Type	Memory All	Initial Value	Variable Ki	Used	Comments
1	INST0	FB Instanc	<Auto>		VAR	*	
2	INST1	FB Instanc	<Auto>		VAR	*	
3	INST2	FB Instanc	<Auto>		VAR	*	
4	LED	BOOL	%QX0.0.0		VAR	*	
5	S	BOOL	%IX0.0.0		VAR	*	



تمرین کلا سی) بازدن کلید اســتارت مدار چراغ راهنمایی را طوری طراحی کنید که مدت زمان روشــن بودن چراغ قرمز ۱۵ثانیه،چراغ سبز ۱۰ ثانیه و چراغ زرد۵ ثانیه باشد.

حال به بررسی مدارات فرمان بصورت اتوماتیک می پردازیم:

تمرین کلاسی) با زدن شستی Start موتور با کنتاکتور اصلی و صل و موتور به صورت ستاره شروع به کار کند ، بعد از ۵ثانیه کنتاکتور ســـتاره قطع و موتور با کنتاکتور مثلث به کار خود ادامه دهد و با زدن Stop موتور در حالتی که باشـــد، خاموش شود.

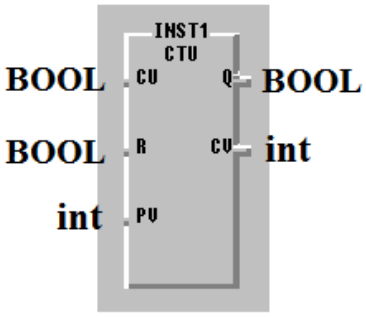
تمرین کلاسی) بازدن شاسی S1 موتور یصورت راستگرد شروع به کار کند، حال بازدن شاسی S2 موتور ۵ثانیه در حالت توقف رفته و سپس تغییر جهت گردش صورت پذیرد و کنتاکتور چپگرد وارد مدار شود.(اولویت فعال شدن کنتاکتورها مهم نیست و فقط توقف ۵ثانیه ای در هنگام تغییر وضعیت مهم می باشد)

تمرین)بازدن شاسی S1 موتور بصورت راستگرد و با کنتاکتور ستاره روشن شود،بعد از ۷ ثانیه کنتاکتور ستاره قطع و کنتاکتور مثلث وارد مدار گردد.حال اگر شاسی S2 زده شد موتور ۵ ثانیه در حالت توقف رفته و سپس تغییرجهت گردش صورت پذیرد. (دقت داشته باشید در هنگام تغییر جهت گردش نیز ستاره مثلث نیز باید مثل راه اندازی ابتدایی مدار باید صورت پذیرد، چون موتور در این مدت متوقف شده است) .با زدن شاسی STOP تمامی کنتاکتورها قطع گردد.

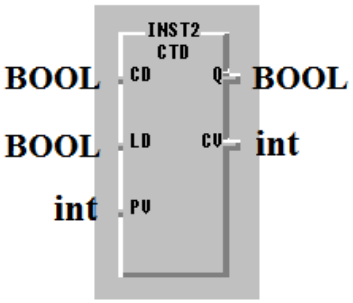
۸ شمارنده ها(کانترها) در GMWIN

عمل شمارنده ها ، شمارش تعداد پالس های ورودی است. کانتر ها از نظر عملکرد به سه قسمت تقسیم می شوند:

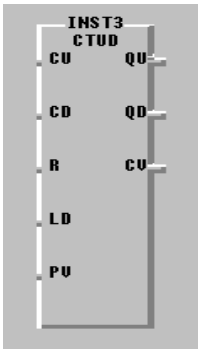
۱) **شمارنده افزایشی:** این کانتر به ازای یک پالس لبه مثبت که در ورودی cu ایجاد شود، یک شماره به بالا می شمارد و یک عدد به cv اضافه می شود.چنانچه عدد شمارش شده(CV) ، از عدد تنظیم شده توسط کاربر (PV) بزرگتر-مساوی شود، خروجی فعال می شود. با تحریک پایه R عدد شمارش شده (CV)، صفر می شود.

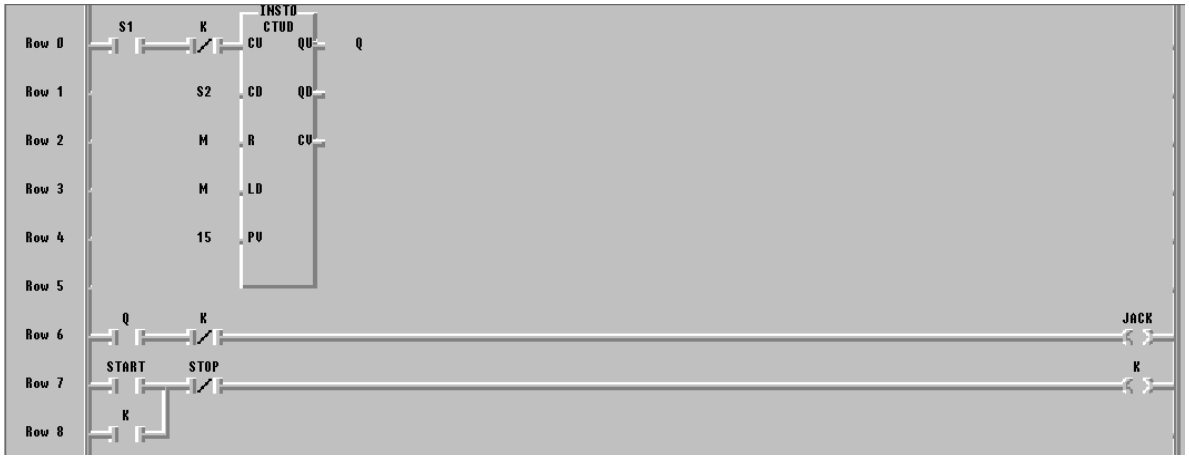


۲) **شمارنده کاهشی:** این کانتر به ازای یک پالس لبه مثبت که در ورودی CD ایجاد شود، یک شماره به پایین می شمارد و یک عدد از cv کم می کند (خاطر نشان شود مقدار اولیه cv=0 است). در این شمارنده چنانچه عدد شمارش شده(CV) از عدد صفر کوچکتر-مساوی شود، خروجی فعال می شود. با تحریک پایه LD عدد شمارش شده با عدد تنظیم شده توسط کاربر برابر می شود.



۳) **شمارنده افزایشی-کاهشی:** این کانتر به ازای یک پالس لبه مثبت که در ورودی cu رو به بالا و با یک پالس لبه مثبت در ورودی CD یک شماره به پایین می شمارد. مابقی پایه ها بصورت کانترهای بالا می باشد. دقت داشته باشید که در این کانتر یا از خروجی افزایشی استفاده می شود یا کاهشی چون کقدار CV اولیه یا باید صفر و یا باید مساوی عدد تنظیم شده توسط کاربر باشد.





مثال) ظرفیت پارکینگی ۱۵اتومبیل است. هرگاه تعداد اتومبیل ها بیشتر از ۱۵ شد، جکی به منظور پر شدن ظرفیت پارکینگ شروع به حرکت کند و هرگاه تعداد اتومبیل ها کمتر از ۱۵ شد، جک برگردد. دو شستی برای جلو آمدن و عقب رفتن جک توسط متصدی نیز در نظر گرفته شود.

	Variable Na	Data Type	Memory All	Initial Value	Variable Ki	Used	Comments
1	INST0	FB Instanc	<Auto>		VAR		
2	JACK	BOOL	%QX0.0.0		VAR		
3	K	BOOL	<Auto>		VAR		
4	M	BOOL	<Auto>		VAR		
5	Q	BOOL	<Auto>		VAR		
6	S1	BOOL	%IX0.0.0		VAR		
7	S2	BOOL	%IX0.0.1		VAR		
8	START	BOOL	%IX0.0.2		VAR		
9	STOP	BOOL	%IX0.0.3		VAR		

تمرین کلا سی) مدار چراغ سیگنالی با فرکانس۰/۵هرتز و چرخه کار۵۰٪ را چنان طراحی کنید که با زدن کلید Start پنج مرتبه چشمک بزند.

تمرین کلاسی) مداری طراحی کنید که با زدن شستی Start موتور روشن شده و روشن بماند و با زدن مجدد همان شستی موتور خاموش شود.

تمرین)مثال بالا را بدون استفاده از کانترها و با استفاده از فلگ ها طراحی کنید.

تمرین کلا سی) مداری طراحی کنید که با زدن شستی S موتور با دور کند شروع به کار نماید و با زدن دوباره همان شستی موتور با دور متوسط به کار خود ادامه دهد و با زدن مجدد همان شستی موتور با دور کند به کار خود ادامه دهد. با زدن شستی Stop هم موتور در هر حالتی متوقف گردد.

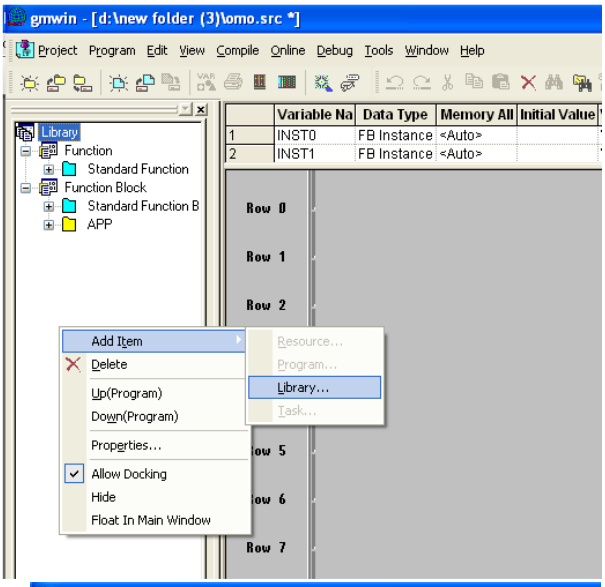
.

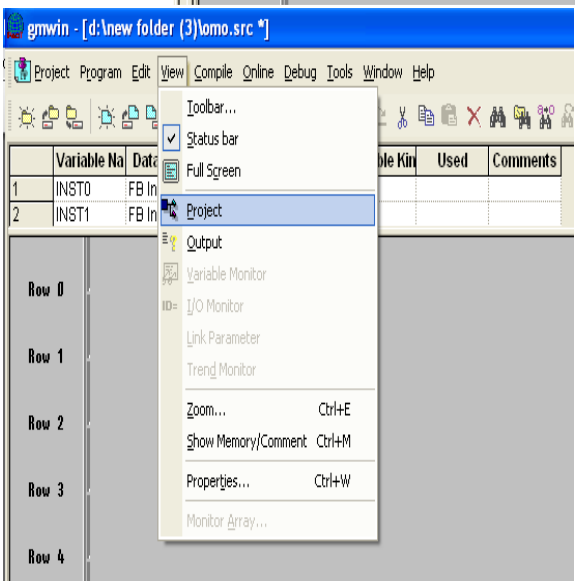
تمرین)شمارنده ای از صفر تا ۹ طوری طراحی نمایید، که بازدن کلید استارت هر یک ثانیه یک عدد به کانتر اضافه شود.

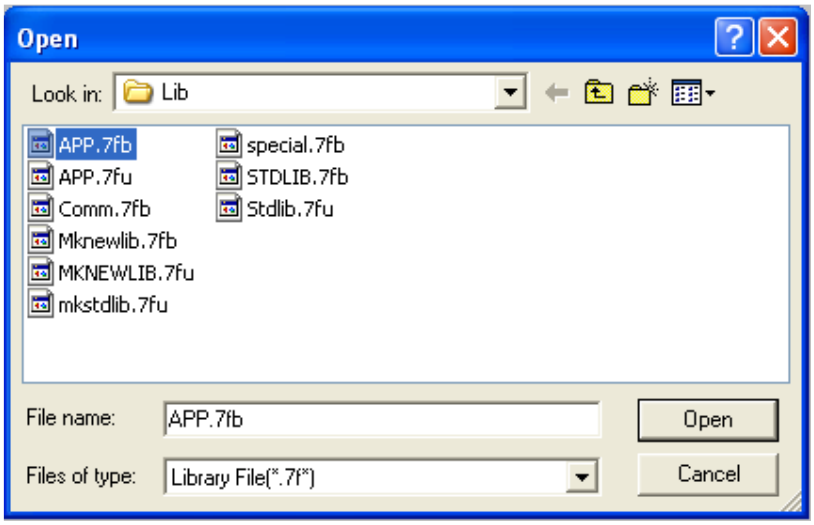
آشنایی با انواع تایمرهای خاص و فلگ های داخلی*PLC*

برای نصب کتابخانه آنها مراحل زیر را دنبال کنید.

ازمنوی *View* گزینه *Project* را انتخاب می کنی . در این حالت پنجره ای در سمت چپ ایجاد می شود .در این گزینه *Library*را انتخاب کرده و بر روی آن را ست کلیک نموده و از منوی باز شده گزینه *Add item* و سپس *Library* را انتخاب می کنیم . با اتخاب آن پنجره کتابخانه ها باز می شود .از لیست کتابخانه ها، کتابخانه *App.7fb* را انتخاب کرده و کلید *open* را می زنی . و از تمام پنجره های فوق خارج می شویم.





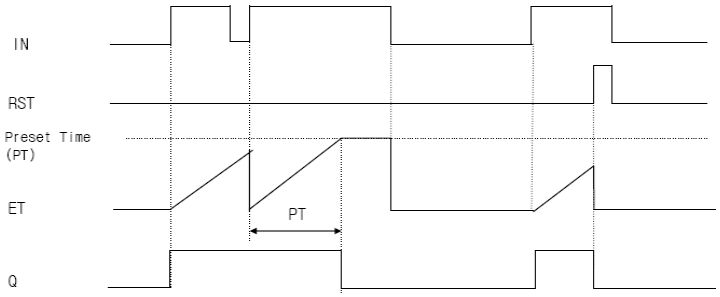


با فراخوانی هرکدام از این کتابخانه ها یک سری تایمر و حتی کانتر و... اضافه خواهندشد.که برای دانستن نحوه عمکرد آن ها help آن بیشترین کمک را خواهد کرد.که به بررسی نحوه عملکرد چند تایمر در زیر پرداخته ایم.

۱. تایمر با قابلیت تحریک دوباره *(Retrigrable Timer , TRTG)*

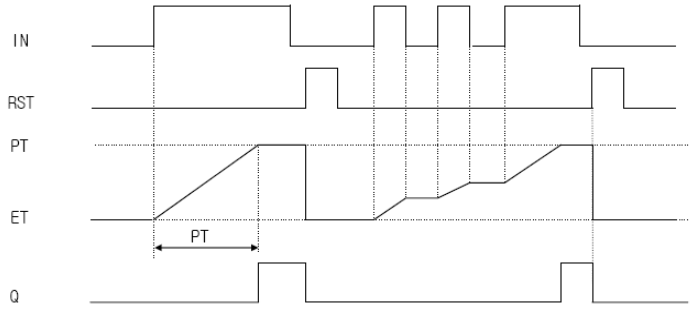
خروجی Q این تایمر همزمان با لبه بالا رونده پالس اعمالی به ورودی IN یک شده و هر گاه مدت زمان سپری شده ET به مقدار اولیه PT برسد خروجی صفر می شود .این تایمر همان تایمر پالس است با دو قابلیت اضافه:

- ✓ اگر تا وقتی که زمان سپری شده به زمان اولیه نرسیده است، دوباره یک لبه بالا رونده در ورودی ایجاد شود زمان ET صفر خواهد شد و دوباره از صفر شروع به افزایش می کند و وقتی به PT رسید خروجی صفر می شود.
- ✓ قابلیت $Reset$ شدن را دارد به ای معنی که هر گاه پایه RST یک شود همزمان Q و ET صفر خواهند شد.



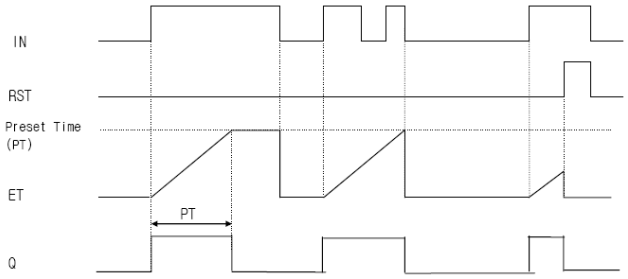
۲. تایمر انتگرالی *(Intrgration , TMR)*

این تایمر مانند تایمر TON عمل می کند .با این تفاوت که PT آن می تواند به صورت گسسته باشد .به این صورت که با یک شدن ورودی زمان سپری شده ET نشان داده می شود، چنانچه قبل از آن که زمان سپری شده به مقدار اولیه برسد ورودی صفرشود مقدار ET متوقف شده و ذخیره می شود .سپس با یک شدن ورودی مقدار سپری شده از جایی که متوقف شده است، شروع به افزایش می نماید (این عمل می تواند چندین بار تکرار شود .) حال اگر مقدار ET به PT برسد خروجی یک خواهد شد .بایک شدن ورودی RST و Q و ET صفر می شود .دیاگرام زمانی آن را می توان مانند شکل زیر نشان داد.



تایمر با قابلیت ریست شدن *(Pulse Timer With Reset , TP_RST)*

عملکرداین تایمرکاملاً مشـابه باتایمرپالس معمولی می باشـد یا این قابلیت که هرگاه ورودی RST یک شـود مقدار Qو ETصفرمیشود.



تایمر انتگرالی چشمک زن *(TMR With Fliker , TMR_FLK)*

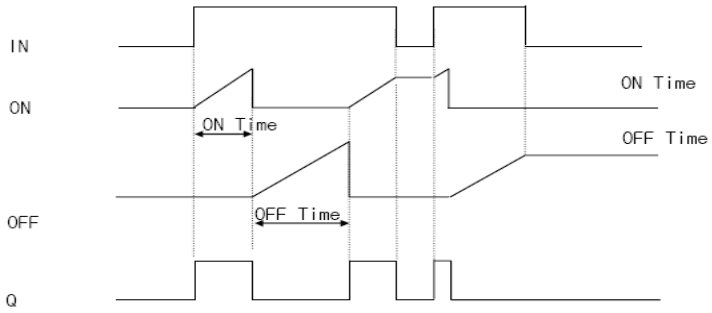
در این تایمر با یک شدن ورودی IN خروجی تایمر تا زمان Ton در وضعیت یک باقی می ماند و پس از گذشت این زمان خروجی صفر می شود و تا زمان $Toff$ خروجی صفر باقی می ماند و مجدداً یک می شود.

۱. هرگاه IN صفر شود زمان سپری شده ذخیره می شود و با یک شدن دوباره IN از همان مقدار قبلی افزایش می یابد.

۲. خروجی در طول مدتی که ورودی صفر است در وضعیت صفر قرار دارد.

۳. اگر مقدار Ton صفر باشد خروجی همواره صفر است.

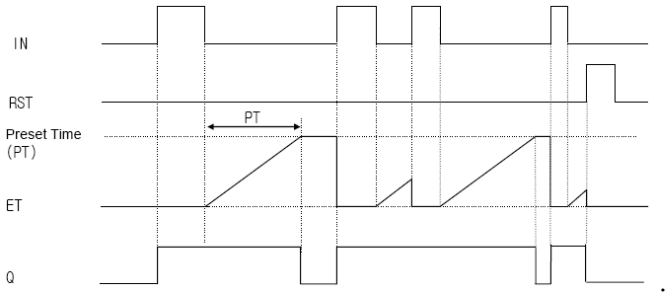
دیاگرام زمانی آن را می توان مانند شکل زیر نشان داد.



تایمر تاخیر در قطع با قابلیت ریست شدن *(Off Delay Timer with Reset)*

در این تایمر با یک شدن ورودی خروجی تایمر نیز یک می شود و زمانی که ورودی صفر شود خروجی پس از گذشت زمان PTیعنی هنگامی که $ET=PT$ می شود صفر می شود.

۱. شروع به کار تایمر درست بعد از صفر شدن ورودی است.
۲. اگر قبل از رسیدن زمان ET به PT ورودی یک شود زمان سپری شده صفر می شود.
۳. در هر زمانی که RST یک شود مقدار ET و خروجی صفر می شود



فلگ ها

فلگ های داخلی PLC را می توان به سه قسمت اصلی تقسیم کرد.

۱. **فلگ های مربوط به خطا:** در صورت بروز برخی از خطاها فلگ هایی به صورت داخلی وجود دارند که این خطاهارا اعلام می کنند .معمولاً به صورت بیتی بوده در صورت بروز خطا یک می شوند.
۲. **فلگ های مربوط به سخت افزار و نرم افزار : *PLC*** ای فلگ ها عموماً جنبه اطلاع رسانی داشته و وضعیت قسمتهای مختلف PLC را نشان می دهند .این فلگ ها اطلاعاتی از قبیل : زمان اسکن برنامه، نوع نرم افزار PLC ،وضعیت کلید سه حالته، وضعیت ارتباط سریال و ... را نمای می دهد و از نوع Bit ، $Byte$ ، $Word$ **و** Int می باشند.
۳. **فلگ های کاربردی:** این فلگ ها توسط کاربر در برنامه کنترلی استفاده می شود .مانند فلگ های یک اسک روشن $(1on\_)$، یک اسک خاموش $(1off\_)$ ، همیشه روش $(on\_)$ ، همیشه خاموش $(off\_)$ و فلگ های تایمر . فلگ های تایمر را به مانند فلگ های یک اسکن روشن که در آزمایش قبل توضیح داده شد می توان به یک کنتاکت

نسبت داد .این فلگ ها در انواع *_T100ms* ، *_T10s* ، *_T1s* ، *_T200ms* ، *_T20ms* ،*_T2s* و *_T60s* می باشند .این فلگ ها با یک شدن ورودی آن ها شروع به قطع و وصل با میزان تناوب نشان داده شده بر روی آنها می نمایند.

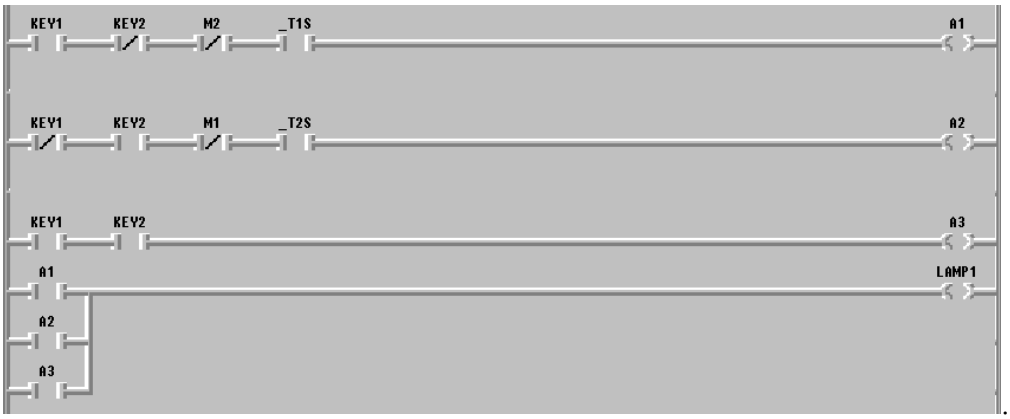
مثال) برنامه ای بنویسید که با فعال شدن *key1* موتور ۱ روشن شودو اولین بارکه *PS1* (سنسور القایی)عمل کردلامپ ۳ روشن شود تا زمانیکه این سنسور عمل میکند لامپ روشن باشد .اگر این سنسور به مدت ۱۲ ثانیه جسمی را حس نکرد لامپ خاموش شود.

مثال) فلاشری به شکل زیر طراحی کنید. (از فلگ های تایمر استفاده کنید)

الف)اگر کلید *key1* وصل بود *Lamp1* با تناوب ۱ ثانیه چشمک بزند.

ب)اگر کلید *key2* وصل بود *Lamp1* با تناوب ۲ ثانیه چشمک بزند.

ج)اگر هر دو وصل بودند *Lamp1* به طور دائم روشن باشد



مثال) فرض کنید موتوری به ازای ساعت کاری مشخصی نیاز به روغن کاری یا گریسکاری دارد .اگر زمان کاری را جمعاً ۲۲ ثانیه فرض کنید و موتور با *Start* وصل و با *Stop* قطع شود در صورتی که جمع ساعت روش بودن موتور به ۲۲ ثانیه رسید موتور خاموش و به مدت ۳ ثانیه روغن کاری شود(.لامپ ۱ روشن شود)

مثال) برنامه ای بنویسید که با زدن *Start* لامپ ۲ به مدت ۱ ثانیه روش شود و ۳۲۲ میلی ثانیه خاموش با شد و با زدن *Stop*لامپ۲ خاموش شود.

تمرین کلاسی) برنامه ای بنویسید که با زدن *Start* لامپ ۱ روش شود .ای لامپ پس از ۵ ثانیه خاموش شود .وقتی لامپ ۱روشن شد پس از ۲/۵ ثانیه لامپ ۲ به مدت ۱ ثانیه روشن شود .در صورتی که پس از روشن شدن لامپ ۱ در هر لحظه کلید *Stop* زده شد تمام لامپ ها خاموش شوند و با زدن *Start* مجدد سیکل تکرار شود.