فصل اول

اصول ومبانی کنترل کننده های قابل برنامه ریزی (PLC)

اهداف آموزشی

۱. آشنایی با انواع مختلف PLC ها
۲. آشنایی با سخت افزار PLC
۳. عملکرد درونی وپردازش سیگنال PLC ها

۱-۱- مقدمه

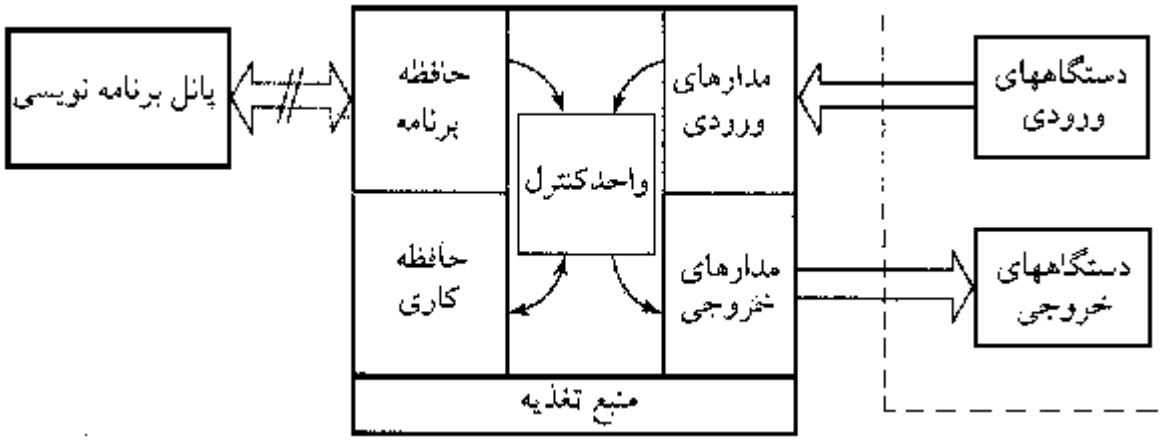
نیاز به کنترل کننده هایی با هزینه کمتر ، کاربرد متنوع تر و سهولت استفاده بیشتر ، منجر به توسعه کنترل کننده های قابل برنامه بر مبنای میکرو پروسسورها شد و از آنها بطور گسترده ای در کنترل فرآیندهای و ماشین آلات استفاده گردید .

PLC ها در آغاز بعنوان جانشینی برای سیستم های منطقی رله ای و تایمری غیر قابل تغییر توسط اپراتور طراحی می شوند تا به جای تابلوهای کنترل متداول و قدیمی استفاده شوند. PLC ها توانستند سهولت و استفاده و قابلیت انعطاف پذیری زیادی را به سیستم های کنترل ارزانی دارند .این کار بوسیله برنامه ریزی آنها و اجرای دستورالعمل های منطقی ساده که اغلب به شکل دیاگرام نردبانی است صورت میگیرد . PLC ها دارای یک سری توابع درونی از قبیل تایمرها ، شمارنده ها و شیفت رجسترها می باشند که امکان کنترل مناسب را ، حتی با استفاده از کوچکترین PLC نیز فراهم می آورند .

یک PLC با خواندن سیگنال های ورودی ، دریافتی از پروسه مورد نظر ، کار خود را شروع کرده و سپس دستور العمل های منطقی (که قبلاً برنامه ریزی شده و در حافظه جا گرفته است) را بر روی این سیگنال های ورودی اعمال می کند و در پایان سیگنال های خروجی مطلوب را برای راه اندازی تجهیزات و ماشین آلات پروسه تولید می نماید . تجهیزات استانداردی درون PLC ها تعبیه شده اند که به آنها اجازه می دهد مستقیماً و بدون نیاز به واسطه های مداری یا رله ای ، به المانهای خروجی یا محرک و مبدل های ورودی (مانند پمپ ها و سوپاپ ها) متصل شوند .

با استفاده از PLC ها ، تغییر یک سیستم کنترل بدون نیاز به تغییر محل اتصالات سیم ها ممکن شده است و برای هر گونه ، تغییر کافی است که برنامه کنترل تغییر یابد .

PLC از نظر ساختمان داخلی شبیه کامپیوترهای معمولی هستند (شکل ۱-۱)، ولی برخی ویژگیهای خاص ، آنها را ابزاری مناسب جهت انجام عملیات کنترل صنعتی نموده است .



شکل ۱-۱ ساختار داخلی یک کنترل کننده قابل برنامه نویسی

برخی از این ویژگیها عبارتند از :

۱- تجهیزات حفاظت کننده PLC ها از نویز و شرایط نامساعد محیطی

۲- ساختار ماژولار PLC ها که به سادگی امکان تعویض یا افزودن واحد یا واحدهایی را به PLC میدهد (مثلاً واحد ورودی و خروجی)

۳- اتصالات استاندارد ورودر /خروجی و نیز سطوح سیگنال استاندارد

٤- زبان برنامه نویسی قابل درک و آسان (مانند دیاگرام ،نردبانی و یا نمودار وظایف)

۵- سهولت در برنامه ریزی و برنامه نویسی مجدد در حین کارکرد فرآیند.

۱-۲- نگاهی گذرا بر تاریخچه PLC

اندیشه ساخت PLC در آغاز سال ۱۹۶۸ توسط یک گروه از مهندسین شرکت General motors آمریکا مطرح شد . در این طرح کنترل کننده می بایست دارای خصوصیات اولیه زیر می بود :

۱- به سادگی قابل برنامه ریزی و همچنین برنامه ریزی مجدد بوده (ترجیحاً در کارخانه) ونیز ، قابلیت تغییر ترتیب و توالی عملیات کنترل را داشته باشد .

۲- نگهداری و تعمیرات آن آسان باشد ، ترجیحاً با استفاده از ماژول های افزودنی

۳- الف) دارای قابلیت اطمینان بیشتر در محیط های صنعتی می باشد .

ب) کوچکتر از رله معادلش باشد

٤- در عمل هزینه قابل رقابت با تابلوهای رله ای و نیمه هادی داشته باشد .

این امر موجب شعله ور شدن شوق شدیدی در بین مهندسین همه شاخه های علوم در مورد اینکه چگونه از PLC می توان در کنترل های صنعتی استفاده کرد گردید. این بذل توجه شدید به قابلیت و تسهیلات برتر PLC ها بود که سبب شد آن ها را به سرعت به فن آوری روز و در دسترس تبدیل کند . دستور العمل ها نیز سیر تکاملی خود را به سرعت از فرمان های منطقی ساده به دستورالعمل های شامل اجرای عملیات مربوط به شمارنده ها ، تایمرها ، شیفت رجیسترها و سپس توابع ریاضی پیشرفته در PLC های بزرگتر طی کردند . به موازات آن ، در سخت افزار PLC نیز پیشرفت ها با حافظه های بزرگتر و تعداد بیشتر ورودی ها و خروجی های تعبیه شده بر روی ماژول های جدیدتر ، دنبال شد در سال ۱۹۷۶ دیگر امکان کنترل ماژول های ورودی /خروجی را دور فراهم آمده بود. در این گونه کاربردها تعداد متعددی از این ورودی / خروجی ها که چند صد متر با PLC ، در فاصله داشتند می بایست از طریق یک خط ارتباطی بطور مداوم Monitoring شوند و یا دستورات لازم به آنها اعمال شود در سال ۱۹۷۷ یک PLC اساس میکرو پروسسوری (PLC مبتی بر ریز پردازنده) به وسیله شرکت آمریکایی آلن برادلی معرفی شد . این PLC بر مبنای ریز پردازنده 8080 نباشد بوده اما از یک پردازشگر دیگر به منظور اداره دستور العمل های منطق بیت در سرعت بالا ، سود می جست .

آهنگ رشد کاربرد PLC ها در صنایع ، تولید کنندگان را تشویق به گسترش و توسعه خانواده سیستم های اساس_میکرو پروسسوری با سطوح عملیاتی مختلف کرده امروزه محدوده PLC های در دسترس از PLC های جامع و کامل کوچک با ۲۰ ورودی /خروجی و ۵۰۰ مرحله مرحله ۳ یا گام برنامه نویسی تا سیستم های ماژولار با ماژول های قابل افزایش را در بر گرفته است . این ماژول ها برای انجام وظایفی نظیر :

۱- ورودی / خروجی آنالوگ

۲- کنترل PID (سه جمله ای : تناسبی ، انتگرال گیر و مشتق گیر)

۳- ارتباطات

٤- نمایش گرافیکی

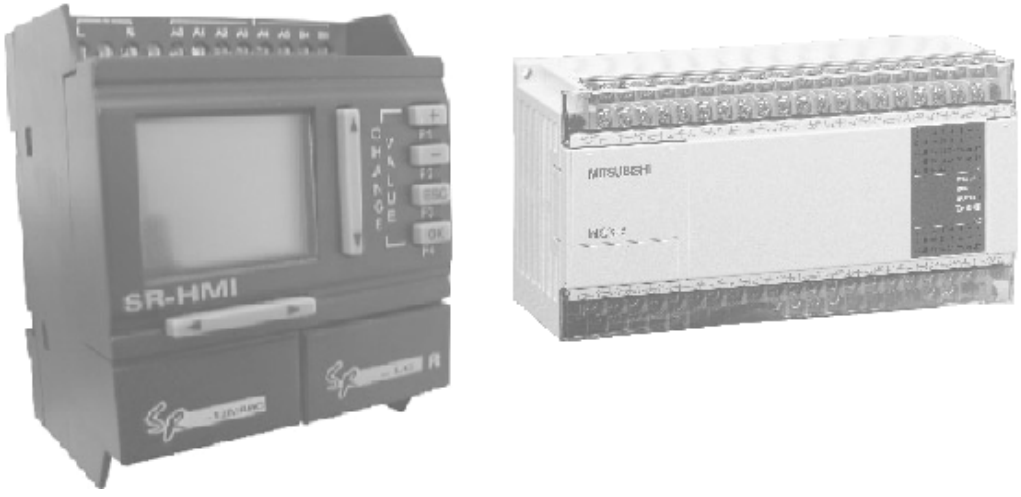
۵- ورودی /خروجی اضافی

۶- حافظه های اضافی

سال	جنبه پیشرفت
۱۹۶۸	اندیشه و مفهوم کنترل‌کننده‌های قابل برنامه‌ریزی زاده شد.
۱۹۶۹	سخت افزار CPUی کنترل‌کننده با دستورات منطقی و 1K حافظه و ۱۲۸ ترمینال ورودی/ خروجی
۱۹۷۴	استفاده از چند پردازشگر با یک PLC ـ تایمرها و شمارنده‌ها ؛ عملیات محاسباتی، K ۱۲ حافظه و ۱۰۲۴ ترمینال ورودی/ خروجی
۱۹۷۶	سیستم‌های ورودی/ خروجی با فاصله دور معرفی شد
۱۹۷۷	PLCهای اساس‌میکروپروسسوری معرفی شدند
۱۹۸۰	توسعه مدول‌های ورودی/ خروجی هوشمند
	بهبود وسایل و تسهیلات ارتباطاتی
	بهبود خصوصیات نرم‌افزارها (مثلا قابلیت مستندسازی)
۱۹۸۳	PLCهای کوچک و کم هزینه معرفی شدند
۱۹۸۵ به بعد	شبکه‌بندی همه سطوح PLC، کامپیوترها و ماشین‌های تحت استاندارد GM MAP، کنترل توزیع شده و سیستم مراتبی کارخانه‌های صنعتی

جدول ۱-۱ ظهور و توسعه کنترل کننده های قابل برنامه ریزی طی سالیان گذشته

راهکارهای ماژولار سازی PLC ها ، امکان گسترش یا بهبود یک سیستم کنترل را با حداقل هزینه و اشکالات فراهم می سازد امروزه PLC ها تقریباً با همان سرعت پیشرفت میکروکامپیوترها مراحل پیشرفت و توسعه را پشت سر میگذارند ، با این تفاوت که تاکید ویژه PLC ها بر روی کنترل کننده های کوچک ، کنترل عددی /وضعیتی و شبکه های ارتباطی می باشد . از نظر بازار نیز ، بازار کنترل کننده های کوچک از اوایل سال های دهه ۸۰ رشد سریعی را شاهد بوده است چرا که در خلال این سالها ، تعدادی از کمپانی ها ژاپنی ، PLC های بسیار کوچک و کم هزینه ای را معرفی کردند که از سایر محصولات آن زمان بسیار ارزانتر بودند به این دلیل مشتریان بالقوه ای در صنعت ، توانایی خرید کنترل کننده های قابل برنامه ریزی را یافتند . این روند با عرضه PLC های کارآمدتر تا حد ممکن ارزانتر ، ادامه یافت . در شکل (۱-۲ الف) نمونه هایی از PLC هایی کوچک نشان داده شده است .



شکل (۱-۲) PLC های کوچک

۱-۳- مقایسه PLC با سایر سیستم های کنترل

جدول ۱-۲ مقایسه ای بین انواع متفاوت ابزارهای کنترلی را به تصویر می کشد. این جدول تنها اشاره ای کلی بر برخی قابلیت های آنها همراه با اطلاعات تکنیکی است که می تواند از برگه اطلاعات سازندگان تهیه شود .

پارامترها	سیستم‌های رله‌ای	سیستم‌های منطقی دیجیتالی	سیستم‌های کامپیوتری	PLCها
هزینه اجرای هر عمل	نسبتا کم	کم	زیاد	کم
ابعاد فیزیکی	بزرگ	بسیار فشرده	نسبتا فشرده	بسیار فشرده
سرعت اجرا	کم	بسیار سریع	نسبتا سریع	سریع
مصونیت در قبال نویز الکتریکی	عالی	خوب	بسیار خوب	خوب
نصب	طراحی و نصب وقت‌گیر	طراحی وقت‌گیر	برنامه نویسی بسیار وقت‌گیر	برنامه نویسی و نصب آسان
قابلیت اجرایی توابع پیچیده	ندارد	دارد	دارد	دارد
سهولت تغییر عملیات	بسیار مشکل	مشکل	کاملا ساده	بسیار ساده
سهولت نگهداری	بد-به دلیل تعداد متعددی کنتاکت	بد اگر آی‌سی‌ها لحیم شده باشند	بد-تعداد زیادی بورد مخصوص	خوب-تعداد اندکی کارت استاندارد

جدول ۱-۲ مقایسه سیستم های کنترل

با یک مقایسه ساده PLC ها خود را بعنوان بهترین انتخاب برای سیستم های کنترل نشان می دهند. مگر اینکه سرعت عملکرد بالاتر از PLC با حفاظت و مقاومت در برابر نویزهای الکتریکی مد نظر باشد که در این حالتها به ترتیب سیستم های دیجیتالی غیر قابل تغییر توسط اپراتور و سیستم های رله ای انتخاب شایسته ای برای سیستم کنترل می باشند . در این امر به کارگیری توابع پیچیده نیز ، یک کامپیوتر معمولی تا اندازه ای بهتر از یک PLC بزرگ مجهز به کارت توابع مربوطه می باشد. اما فقط از نظر ایجاد توابع نه اجرای آنها ، زیرا PLC ها بدلیل محول کردن بخشی از پردازش به پردازشگرهای اختصاصی مربوط به ماژول های اجرا کننده وظایف ویژه (نظیر ماژول های PLC) کارآمد تر خواهند بود. بنابراین PLC ها محاسبات مربوط به این توابع کنترلی را مستقل از پردازشگر اصلی انجام می دهند و در واقع مانند یک سیستم چند پردازشگر (با چند پردازشگر) عمل می کنند.

PLC ها دارای یک سری مشخصات سخت افزاری و نرم افزاری می باشند که آنها را برای کنترل محدوده وسیعی از تجهیزات صنعتی ، بسیار ایده آل ساخته است. اکنون به ذکر جزئیات این ویژگی ها خواهیم پرداخت.

۱-۴- سخت افزار PLC

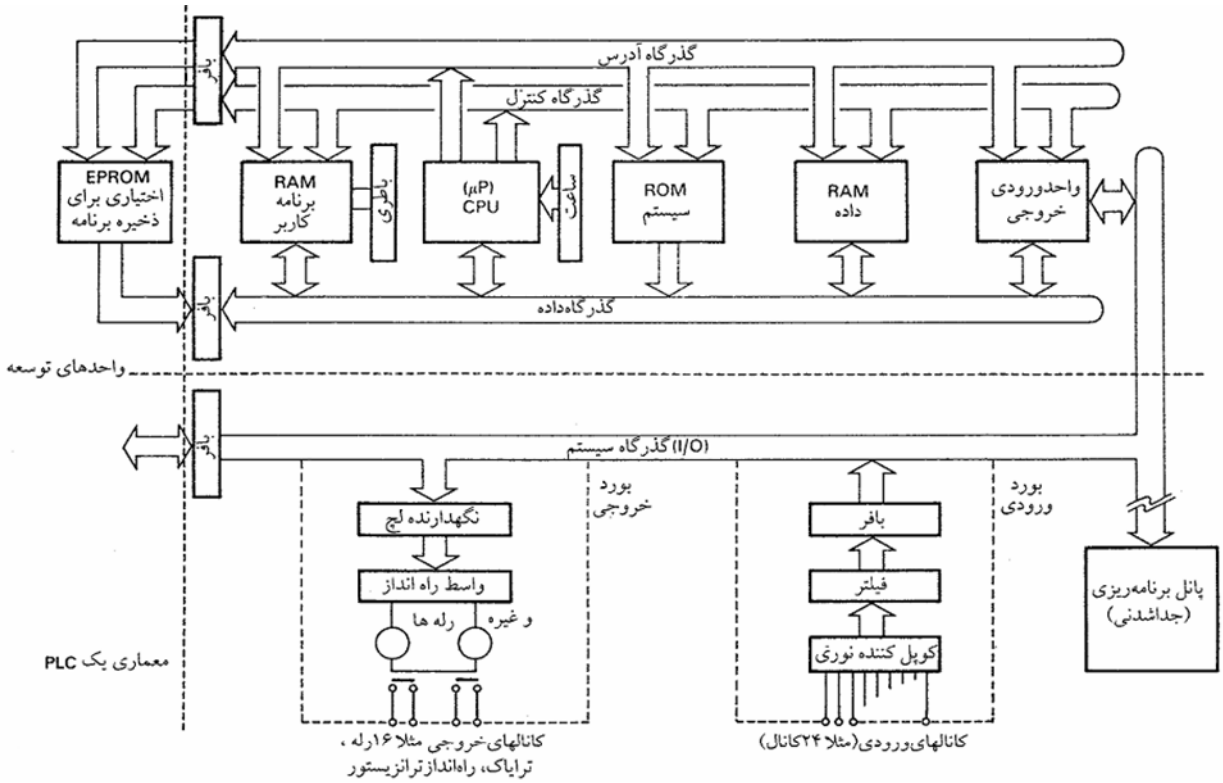
PLC ها کامپیوترهایی، ساخته شده به منظور خاص هستند که شامل سه قسمت اصلی اجرایی می باشند :

۱-پردازشگر ، ۲- ورودی /خروجی ،۳- حافظه –سیگنال های مأخوذ از فرآیند از طریق ورودی به PLC فرستاده شده و آن گاه در حافظه جایی که PLC فرمان های منطقی برنامه ریزی شده را به این ورودی ها اعمال میکند ، ذخیره می شوند. سپس سیگنال های خروجی به منظور راه اندازی تجهیزات مورد نظر ،تولید می شوند. عملی که رخ خواهد داد کاملاً به برنامه کنترل که در حافظه نگهداری می شود بستگی خواهد داشت. در PLC های کوچک تر، این عملیات توسط کارت های ویژه ای انجام می گیرند که به صورت واحد های بسیار فشردهای ساخته شده اند در حالی که ساختار PLCهای بزرگ تر به صورت ماژولار با ما ماژول هایی که بر روی شیارهای تعبیه شده بر روی نصبگاه ها نصب می شود ، بنا گردیده است این امر امکان توسعه سیستم را (در صورت ضرورت) به سادگی فراهم می آورد . در هر دوی این موارد بورد های مداری ویژه ای ، به سادگی تعویض یا برداشته می شود وامکانات تعمیر سیستم نیز به سادگی فراهم می آید . به یک واحد برنامه ریزی سیستم نیز برای بار کردن برنامه های کنترل به حافظه PLC نیاز می باشد . (بار کردن یا down load که طی آن برنامه ها یا داده ها از سیستم بزرگ به سیستم کوچکتر یا یک وسیله وابسته منتقل شود . در مقابل عمل معکوس آن زیر بار کردن نامیده می شود)

۱-۴-۱ واحد پردازش مرکزی (CPU)

CPU بر تمام عملیاتی که درPLC رخ می دهد، کنترل و نظارت دارد ودستورالعمل های برنامه ریزی شده و ذخیره شده را اجرا می کند. یک شاهراه ارتباطاتی درونی یاگذرگاه سیستم، اطلاعات را با نظارتCPU به صورت دو طرفه بین CPU ، حافظه و درگاهها یا پورت های ورودی/خروجی منتقل می کند. CPUبا فرکانس ساعت (CLOCK) توسط یک کریستال کوارتز یا نوسانگرRC تغذیه می شود . این فرکانس بستگی به نوع ریز پردازنده و محدوده عملیاتی نوعاً بین ۱تا ۸مگا هرتز می باشد . مولد ساعت یا CLOCK تعیین کننده سرعت

عملکرد PLC است وشرایط زمان بندی و هم زمـان سـازی را بـرای تمـام اجـزای موجـود در سیستم فراهم می آورد (شکل۱-۳) .



(شکل۱-۳) بلوک دیاگرام داخلی PLC

تقریباً جمیع PLC های مـدرن اسـاس میکـرو پروسـسوری هـستند، کـه از یـک پردازنـده مرکزی به عنوان CPU سیستم استفاده می کنند. برخی PLC های بزرگتر، ریز پردازنده هـای اضافی دیگری نیز برای کنترل توابع پیچیده و زمان بر مانند پردازش های ریاضی ، کنتـرل سـه جمله ای PID و غیره به خدمت می گیرند.

۱-۴-۲ حافظه

حافظه ها دارای دو کاربرد درPLC ها می باشند:

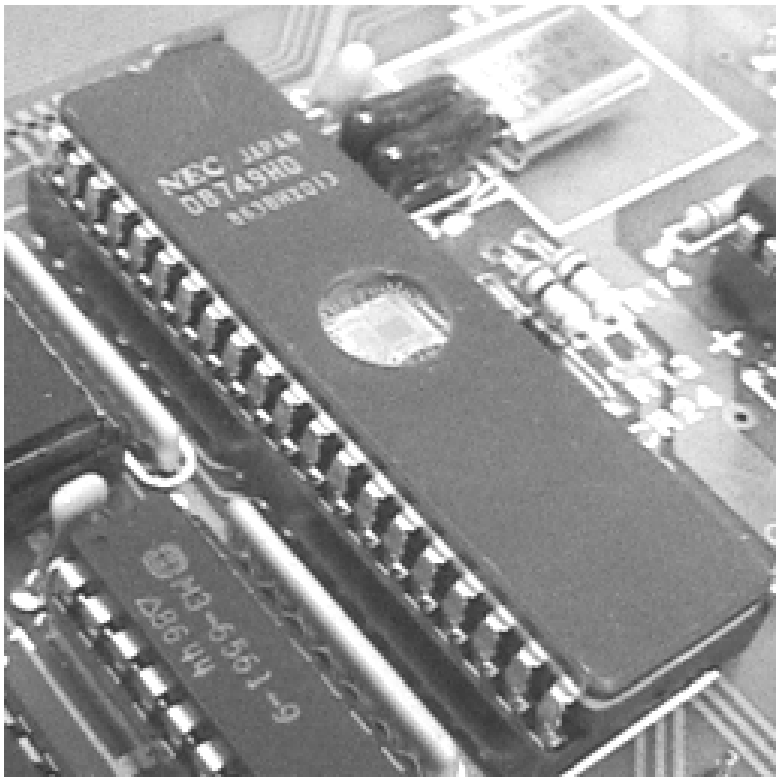
الف) تمام PLC های مدرن برای ذخیره برنامه از حافظه نیمه هادی مانند , EPROM RAM و یاE^2 PROM استفاده می کنند .

عملاً از RAM برای تکمیل برنامه مقدماتی و تست آن استفاده می شود، زیرا که امکان تغییر و اصلاح راحت برنامه را فراهم می آورد. امروزه تمایل کلی به ساخت RAM هایی با استفاده از فن آوری CMOS می باشد. زیرا توان مصرفی آنها پایین بوده و برای نگهداری اطلاعات آن ها در هنگام قطع برق نیز از باطری پشتیبان استفاده میکنند (حافظه RAM ذاتاً فرار است.) این باطری ها قبل از نیاز به تعویض یا جابه جایی ، طول عمری دست کم یکسال دارند. همچنین این

امکان وجود ارد که از نو ع قابل شارژ آن ها استفاده شود تا هر زمان که تغذیه PLC از برق اصلی صورت می گیرد، این باتری شارژ می شود.

این راهکار، ذخیره برنامه در RAM را تقریباً به صورت دایمی در خواهد آورد. در بسیاری از سیستم های PLC ، تنها بر اساس حافظه های RAM با باتری پشتیبان کار می کنند، بنابراین هر گاه ضرورت ایجاب کند، خصوصیات برنامه به سادگی می تواند تغییر یابد.

پس از این که برنامه تکمیل شد و مورد آزمایش قرار گرفت می توان آن را در PROM یا EPROM که اغلب ارزانتر از قطعات RAM می باشند ، بار (load) کرد. برنامه ریزی PROM معمولاً توسط یک برنامه ریز مخصوص (Programer) صورت می گیرد، اگر چه که هم اکنون بیشتر کنترل کننده های قابل برنامه ریزی دارای تسهیلاتی می باشند که اجازه می دهد، برنامه موجود در حافظه RAM کنترل کننده، به درون IC حافظه PROM ی که در سوکتی که برخود PLC تعبیه شده زیر بار (down load) گردد .



شکل (۱-۴) یک حافظه EPROM

ب)علاوه بر ذخیره برنامه، یک PLC به حافظه جهت انجام وظایف دیگری نیز نیازمند است. به عنوان مثال :

۱. I/O RAM جهت کپی کردن ورودی /خروجی.
۲. ذخیره موقت برای وضعیت توابع داخلی مثلاً تایمر ها، شمارنده ها، رله های نشانگر.

از آنجا که داده های درون این حافظه ها مرتباً تغییر می کند (مثلاً تغییر وضعیت یک ترمینال ورودی) بنابراین باید از نوع RAM بوده (قابل نوشتن /خواندن) و ممکن است در بعضی قسمت ها نیازمند باتری پشتیبان باشند.

۱-۴-۳ واحدهای ورودی /خروجی

بیشتر PLC ها با ولتاژداخلی ۵ و ۱۵ ولت dc (ولتاژهای معمول TTL و CMOS) کار می کنند در حالی که می توانند بر روی سیگنال های خیلی بزرگتر، بعنوان نمونه ۲٤ v.dc تا v.ac ۲٤۰ و چندین آمپر عمل پردازش انجام دهند. مطابق شکل (۱-۵). واحدهای ورودی /خروجی پل ارتباطی بین دنیای میکرو الکترونیک دورن PLC و دنیای واقعی بیرونی را تشکیل می دهند ، بنابراین باید دارای توابع بهسازی سیگنال و جدا سازی های لازم (ایزولاسیون) باشند. این ویژگی ها غالباً PLC را قادر می سازند تا مستقیماً و بدون نیاز به واسطه های مداری یا رله ها به محرک ها (actuators) و مبدل های ورودی اتصال یابد .

به منظور فراهم آوردن این امکان، PLC ها چندین نمونه از واحدهای ورودی/ خروجی قابل انتخاب را ارائه می دهند تا با شرایط خاص هر مبدل یا محرک موفق داشته باشند به عنوان مثال :

انتخاب ورودی :

5v (ولتاژ معمول TTL) ورودی سوئیچ (دیجیتالی)

24v ورودی سوئیچی

10v ورودی سوئیچی

240v ورودی سوئیچی

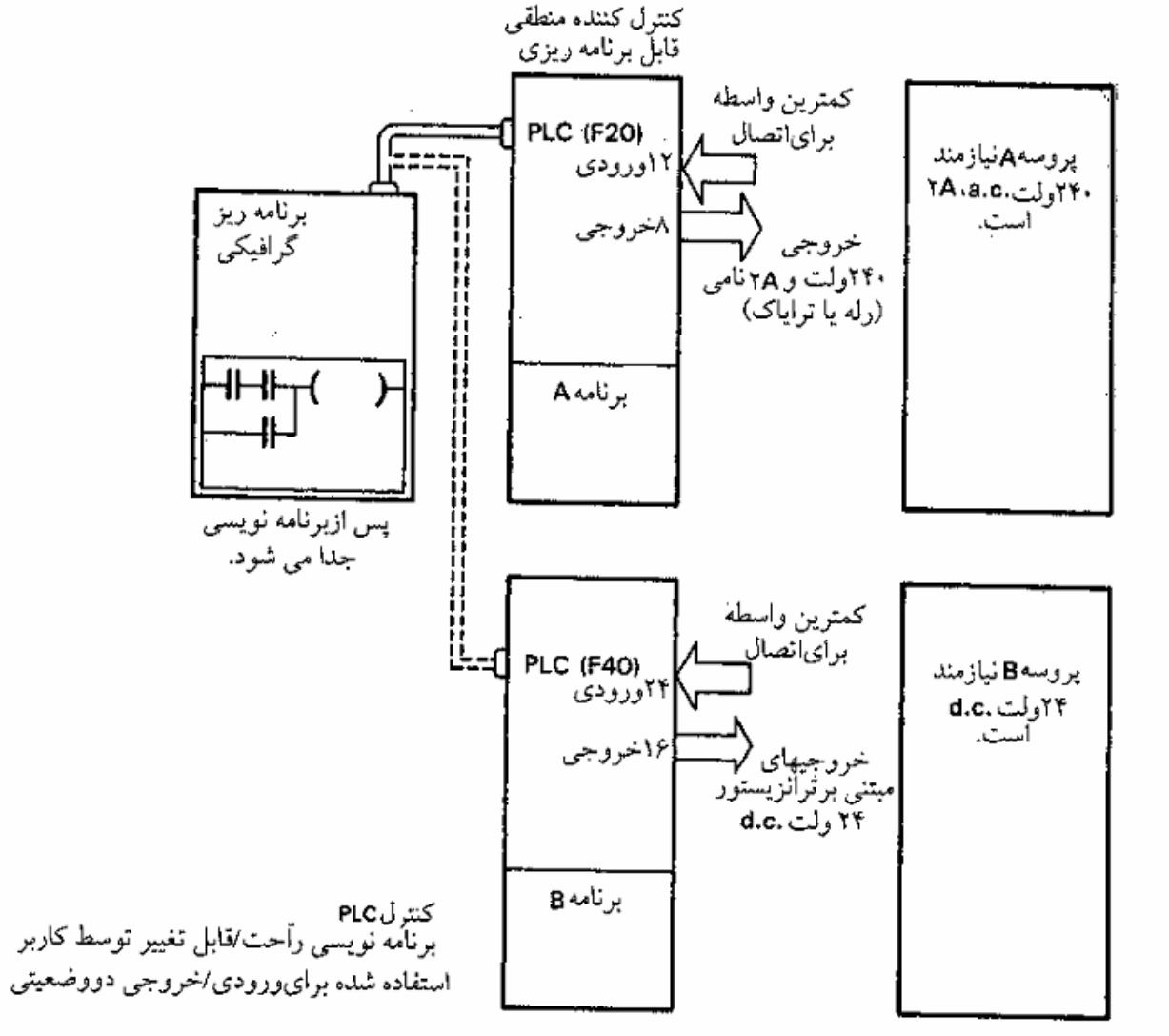
انتخاب های خروجی :

100A , 24v خروجی سوئیچی

10amp , 110v

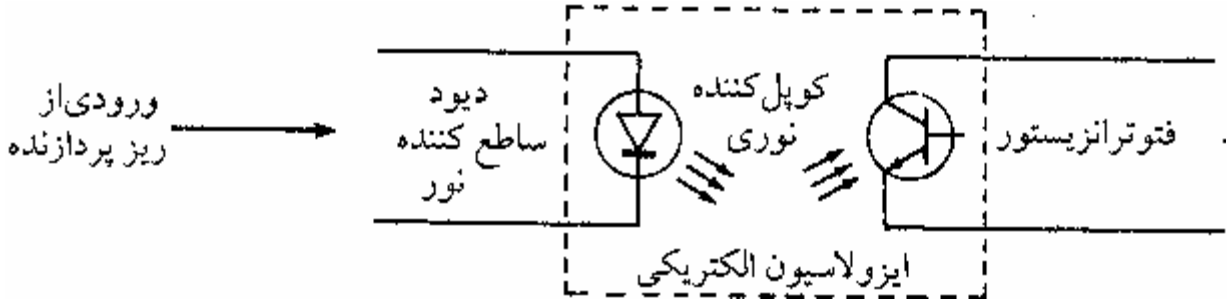
1A(a.c) , 240v (ترایاک)

2A(a.c) , 240v (رله)



شکل (۱-۵) ورودی / خروجی PLC متصل شده به تجهیزات پروسه

شیوه استاندارد برای اتصال تمام کانال های ورودی / خروجی آن است که از لحاظ الکتریکی با فرآیند تحت کنترل ایزوله باشند که این امر با استفاده از جدا کننده نوری در ماژول های ورودی / خروجی انجام می گیرد (شکل ۱-۶) یک مدار جدا کننده نوری شامل یک LED و یک فتو ترانزیستور است که یک زوج کوپل شده توسط نور را به وجود می آورند . این زوج اجازه می دهد تا سیگنال های کوچک عبور کنند. لیکن ولتاژ های زیاد ناگهانی را به همان سطح ولتاژ کوچک قبلی برش خواهد داد . (زیرا ولتاژهای ورودی بیش از مقدار معمول موجب صدمه زدن به CPU خواهد شد . مثلاً اگر ورودی معیوب شده و یا اتصال کوتاه شود ولتاژهای زیاد ورودی مستقیماً به CPU خواهد رسید .)



شکل (۱-۶) مدار جدا کننده نوری

این موضوع سبب حفاظت PLC در برابر ولتاژهای زیاد و ناگهانی (surge) ناشی از ترانزیستورهای سوئیچینگ و منبع تغذیه (که معمولاً به ۱۵۰۰ ولت هم می رسند) می شود . در PLC های کوچک جامع که در آنها همه ورودی / خروجی ها ی در یک محفظه جای داده شده اند ، همه ورودی ها در یک سطح ولتاژ عمل خواهند کرد (مثلاً 24v) و همچین برای خروجی ها این امر صادق است (مثلاً همه 240v ترایاک). علت این امر آنست که تولید کنندگان به دلایل اقتصادی ، فقط بوردهای ویژه با محدوده وظایف ثابت و استاندارد را تولید می کنند. اما PLC های ماژولار در مورد واحدهای ورودی / خروجی انعطاف پذیری بیشتری از خود نشان می دهند. چرا که کاربر می تواند چندین نوع متفاوت از ماژول های ورودی / خروجی و همچنین ترکیبی از آنها را انتخاب کند.

واحدهای ورودی/خروجی با هدف تسهیل اتصال سنسورها یا مبدل های پروسه و نیز محرک ها (actuators) با کنترل کننده های قابل برنامه ریزی طراحی می شوند. به این منظور همه PLC های مجهز به ترمینال مارپیچ استاندارد یا فیش هایی در هر محل ورودی / خروجی می باشند که جابجایی تعویض کارت های ورودی/ خروجی معیوب را سریع و آسان نموده است .

هر ترمینال ورودی / خروجی دارای آدرس منحصر به فرد یا شماره کانالی است که در طی مراحل برنامه نویسی از آن استفاده می شود تا بتوان در حین اجرای برنامه ، مثلاً خواندن یک ورودی یا فعال شدن یک خروجی خاص را مشخص کرد. نمایش وضعیت کانالهای ورودی/ خروجی توسط LED هایی که روی PLC یا واحدهای ورودی / خروجی تعبیه شده است، انجام می گیرد که چک کردن ورودی های وارده از فرآیند به PLC یا خروجی های خارج شده از آن را ساده می سازد.

۱-۵ واحد برنامه ریزی (Programer)

تقریباً بیشتر پانل های برنامه نویسی ساده ، دارای حافظه RAM مورد نیاز جهت ذخیره نیمه دائم برنامه در حال تکمیل یا اصلاح هستند. اگر پانل برنامه نویسی از نوع قابل حمل باشد، در این صورت RAM آن معمولاً از نوع CMOS با باتری پشتیبان بوده تا پانل را قادر به حفظ و نگهداری برنامه ها در حین جابجایی در سطح کارخانه یا کارگاه نماید. یک برنامه تنها زمانی به PLC منتقل شود که آماده استفاده یا تست باشد. هرگاه که برنامه نصب شده به طور کامل تست و اشکال زدایی شد، پانل برنامه نویسی جدا می شود و می تواند برای کنترل کننده دیگری به کار رود .

پانل برنامه نویسی ممکن است دارای امکاناتی نظیر forcing باشد. منظور از اجبار یا forcing تابعی است که برخی از PLC ها از آن ها پشتیبانی می کنند و بوسیله آن ها اعمالی عادی انجام نمی گیرند را به انجام رسانند. مثلاً خروجی که معمولاً بایستی خاموش باشد را مجبور به روشن شدن میکند. در این حالت معمولاً با استفاده از تسهیلات monitoring می توان وضعیت خروجی را بر روی پانل برنامه ریزی مشاهده نمود و مشاهده بلادرنگ (real-time) سوئیچ ها، گیت ها و توابع را ، در حین اجرای برنامه امکان پذیر سازد . این خاصیت می تواند برای عیب یابی، خصوصاً وقتی که فرآیند مورد نظر در فاصله دوری قرار دارد و یا غیر قابل دسترسی است ، بسیار ارزشمند باشد .

PLC های بزرگتر اغلب با یک نمایشگر ویدویویی (VDC) همراه با صفحه کلید کامل و صفحه نمایش ، که به کنترل کننده از طریق یک ارتباط سریال (معمولاً RS232) متصل می گردند ، برنامه ریزی می شوند .

۱-۶ PLC ها - عملکرد درونی وپردازش سیگنال

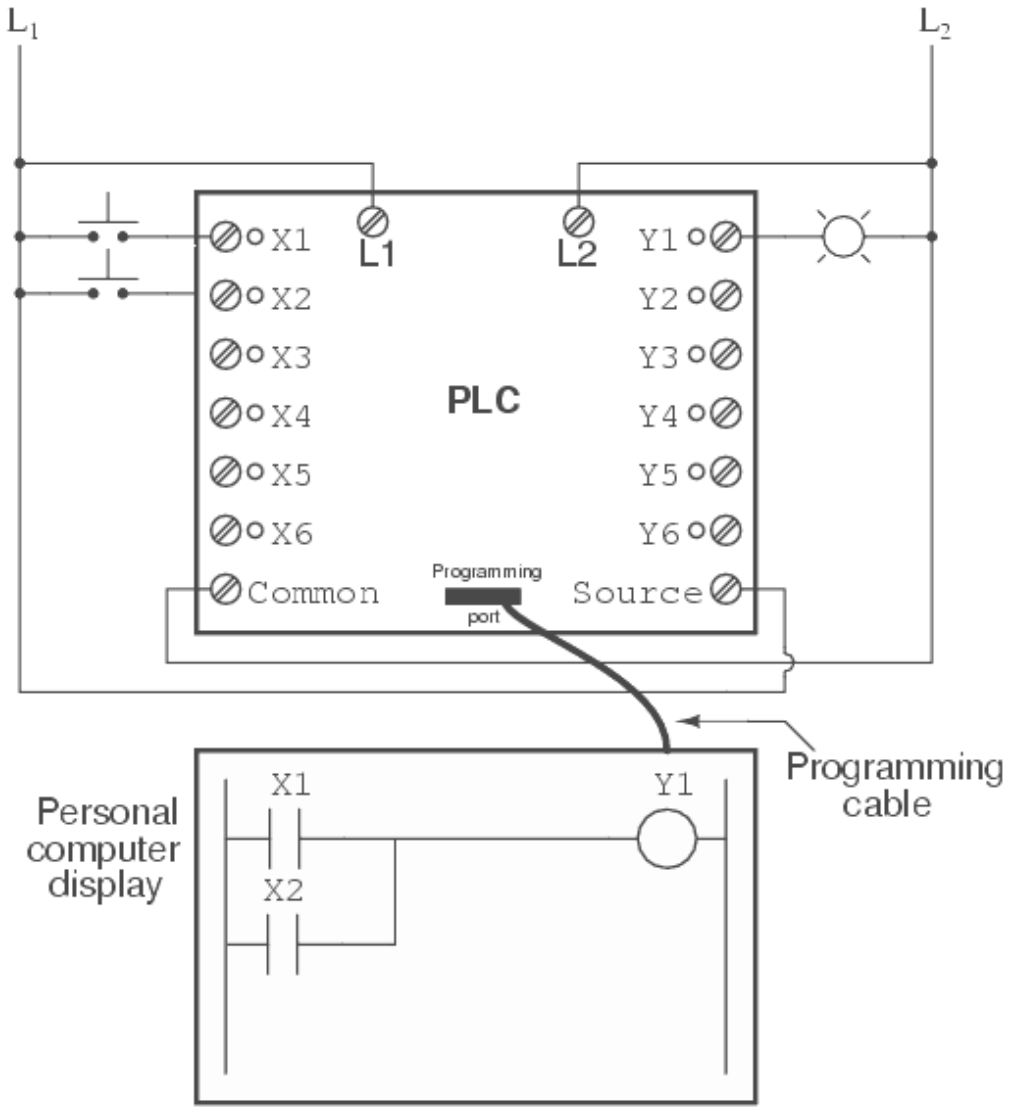
زمانی که یک برنامه در PLC بار(Load) می شود ، هر دستور العمل در یک مکان منحصر به فرد (یا آدرس یکتایی) از حافظه قرار میگیرد .

CPU دارای یک "رجیستر شمارنده برنامه " می باشد که دستور العمل بعدی اشاره میکند تا از حافظه خوانده یا اصطلاحاً واکنشی شود . [fetch یا واکنش عملیاتی است که در طی آن یک دستور العمل از حافظه خوانده شده و در یک رجیستر ذخیره می شود]

هنگامی که یک دستور العمل توسط CPU دریافت می شود در "رجیستر دستورالعمل"قرار می گیرد تا به عملیات درونی یا ریز دستورالعمل های مورد نیاز آن دستور العمل به خصوص ،

دیکد یا کدگشایی شود. نتیجه این کار ممکن است این باشد که مثلاً دستور العمل های بعدی از حافظه خوانده شود و یا یک وسیله فیزیکی توسط CPU راه اندازی گردد.

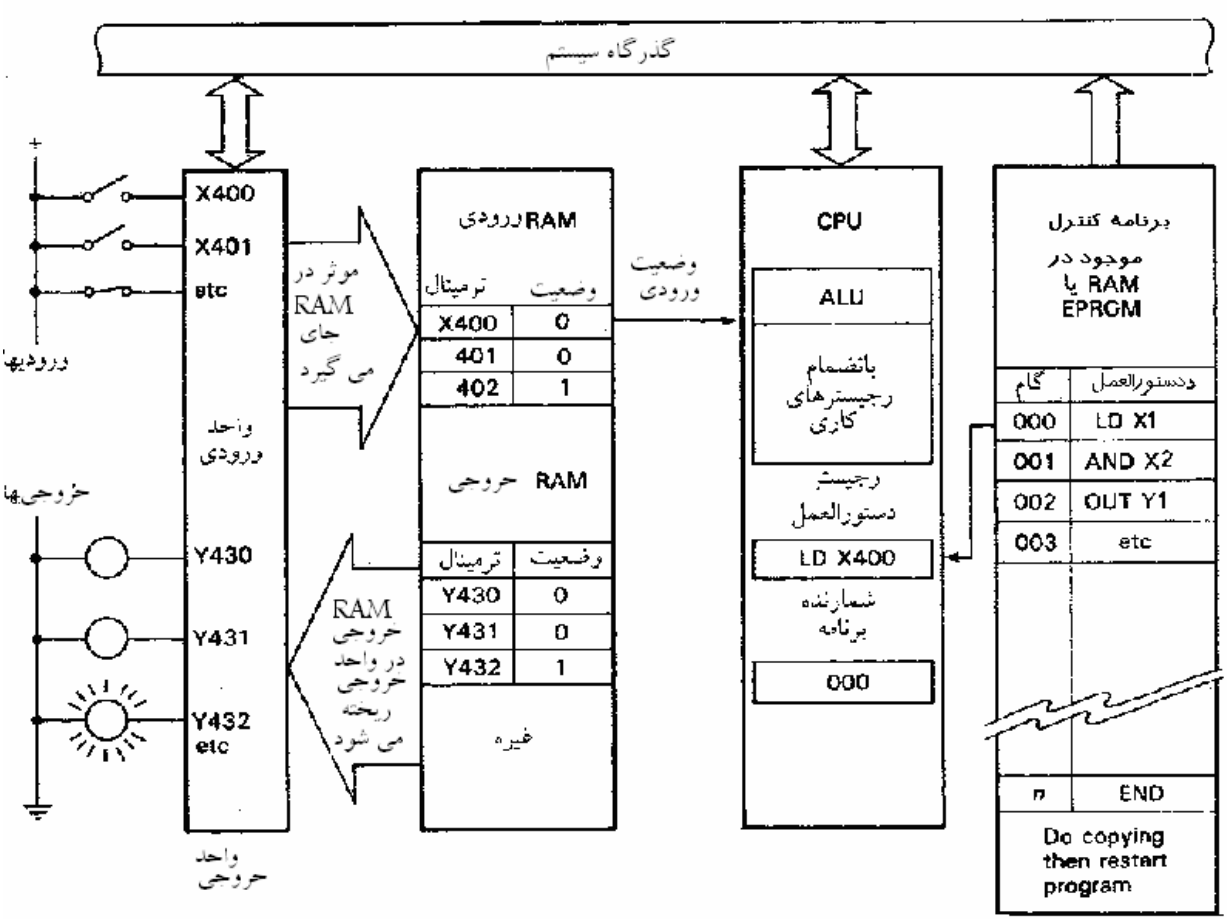
بعنوان نمونه شکل (۱-۷) را در نظر بگیرید. در این شکل می خواهیم برای روشن شدن لامپ مورد نظر از یک PLC استفاده نماییم، بطوریکه در صورت زدن یکی از دو کلید لامپ روشن شود. ورودی ها که دو کلید می باشند به ورودی PLC متصل شده اند. و خروجی PLC به لامپ متصل می باشد، برنامه کنترلی مورد نظر به PLC بار می شود.



شکل (۱-۷) کنترل یک لامپ توسط PLC

مطابق شکل (۱-۸) ابتدا برنامه کنترلی در پانل برنامه ریز نوشته می شود. وسپس به PLC ارسال می گردد. عملکرد درونی PLC بدین صورت است که در درآغاز زمانی که PLC برای شروع به کار ، ست می شود برنامه یا Program counter به آدرس 0000 حافظه RAM و یا

EPROM اشاره خواهد کرد : یعنی محل اولین فرمان حافظه. سپس CPU دستور العمل این آدرس را خوانده، کد گشایی کرده و سپس اجرا می کند که در این جا اولین فرمان LD X1 می باشد. در این حالت CPU متوجه خواهد شد که این دستور مشخص کننده اولین عنصر یک مدار منطقی است که یک کنتاکت باز (LD) متناظر با کانال ورودی (با توجه به حرف X) شماره 1 می باشد. به همین ترتیب CPU شروع به خواندن خط های بعدی برنامه نموده و آنها را اجرا می نمایید. باید توجه داشت که وضعیت ورودی در حافظه I/O RAM نگهداری می شود تا این که CPU خانه حافظه RAM تخصیص داده شده به ورودی X1 را مرور (Scan) نماید.



شکل۱-۸ پردازش سیگنال در CPU

مقدار X1 به داخل رجیسترهای عملیاتی واحد پردازشگر وارد می شود بعد از اتمام اجرای خط اول برنامه شمارنده برنامه یکی افزایش می یابد و به خط بعدی برنامه اشاره می کند، که در اینجا دستور AND X2 می باشد. ابتدا ورودی X2 به ترتیبی که قبلاً ذکر شد خوانده شده به رجیستر CPU منتقل می شود. در این زمان واحد ALU عملیات منطقی AND را بر روی دو ورودی X1 و X2 انجام می دهد.

دستور سوم نیز همانند دستورات بالا اجراءِ می شود. در نتیجه اجرای این دستور حاصل عملیات قبل در حافظه خروجی قرار می گیرد. در انتهای سیکل اجرا مقادیر حافظه های خروجی به خروجی های متناظر اعمال می شود. مطابق شکل (۱-۸) در صورت یک شدن حاصل OR ورودی های X1 و X2 خروجی Y1 که در اینجا به یک لامپ متصل است روشن می شود.

۱-۷ ارتباط CPU با ورودی خروجی ها

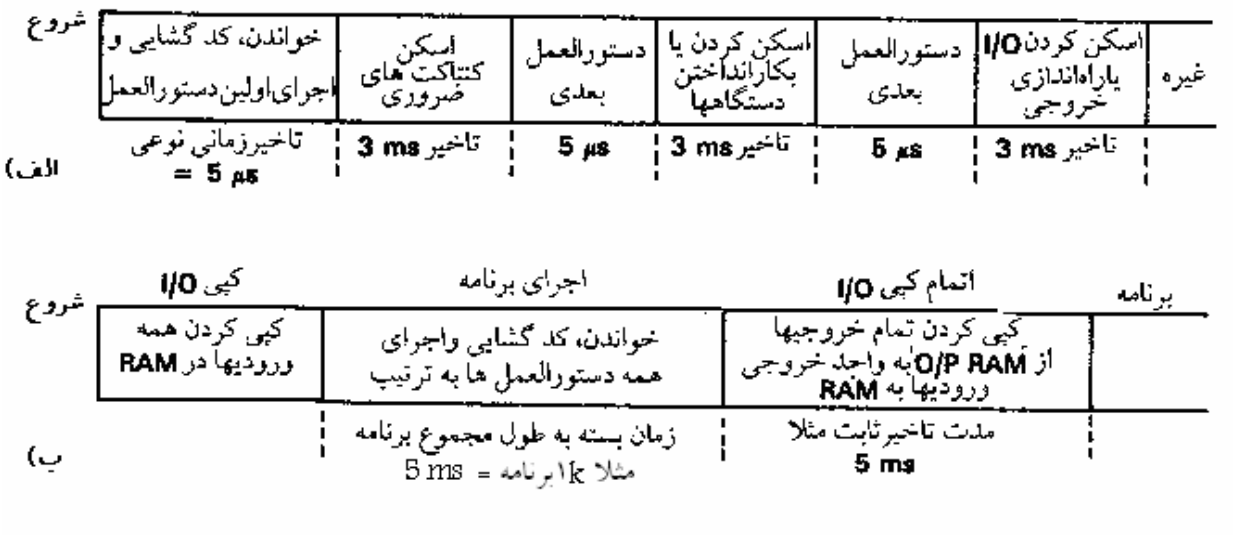
دریافت ورودی ها از واحد ورودی وارسال خروجی ها به واحد خروجی به دو صورت انجام می شود :

۱. **روش نمونه برداری مداوم :** در هر بار که PLC نیاز به یک ورودی داشته باشد مستقیماً آنرا ازآدرس مورد نظر می خواند و هرگاه داده ای را به خروجی نسبت دهد آنرا بلافصله به خروجی ارسال کند این روش در PLC های کوچک مورد استفاده قرار می گیرد. عمل خواندن داددده ها از ورودی ها و یا ارسال داده ها به خروجی ها با کمی تأخیر صورت می گیرد. این تاخیر جهت تضمین این مطلب است که فقط سیگنال های ورودی معتبر، به درون پردازشگر راه خواهند یافت. (تأخیر حدوداً چند میلی ثانیه ای از اتصال پالس های ناشی از اتصال کنترل نشده کنتاکت ها وسایر نویز های ورودی به PLC جلوگیری می کند). همچنین کانال های خروجی ، زمانی که دستور العمل های OUT در یک عملیات منطقی اجرا میشوند ، راه اندازی می گردند . این خروجی ها در واحد I\O نگهداری (Latch) می شوند و وضعیت آن ها توسط این واحدها تا به هنگام رسانی بعدی حفظ می شود شکل(۱-۹ الف).

۲. **کپی یک جای ورودی /خروجی :** PLC های بزرگ دارای چند صد کانال ورودی / خروجی می باشند . از آن جا که در طی اجرای برنامه ، CPU تنها قادر به پردازش یک دستورالعمل در هر لحظه است ، وضعیت هر ترمینال ورودی بایستی جداگانه بررسی شده تا تاثیر آن در برنامه مشخص گردد. نظر به این که در "روش نمونه برداری مداوم " برای هر ورودی به یک تاخیر ۳ میلی ثانیه ای نیازمندیم ، مجموع زمان لازم در هر سیکل برنامه همراه با افزایش تعداد ورودی ها ، افزایش خواهد یافت .

به منظور اجرای سریع برنامه ، می توان به هنگام رسانی ورودی /خروجی را در محل خاصی از برنامه انجام داد . در این روش از یک ناحیه معین حافظه RAM کنترل کننده، به عنوان یک حافظه کمکی یا موقت (Buffer) بین مدار منطقی کنترل و واحد ورودی/خروجی استفاده می شود . هر کانال ورودی و خروجی دارای یک خانه در این I/O RAM می باشد . در جریان کپی کردن ورودی /خروجی ها ،CPU همه ورودی ها را در واحد ورودی /خروجی مرور (Scan)

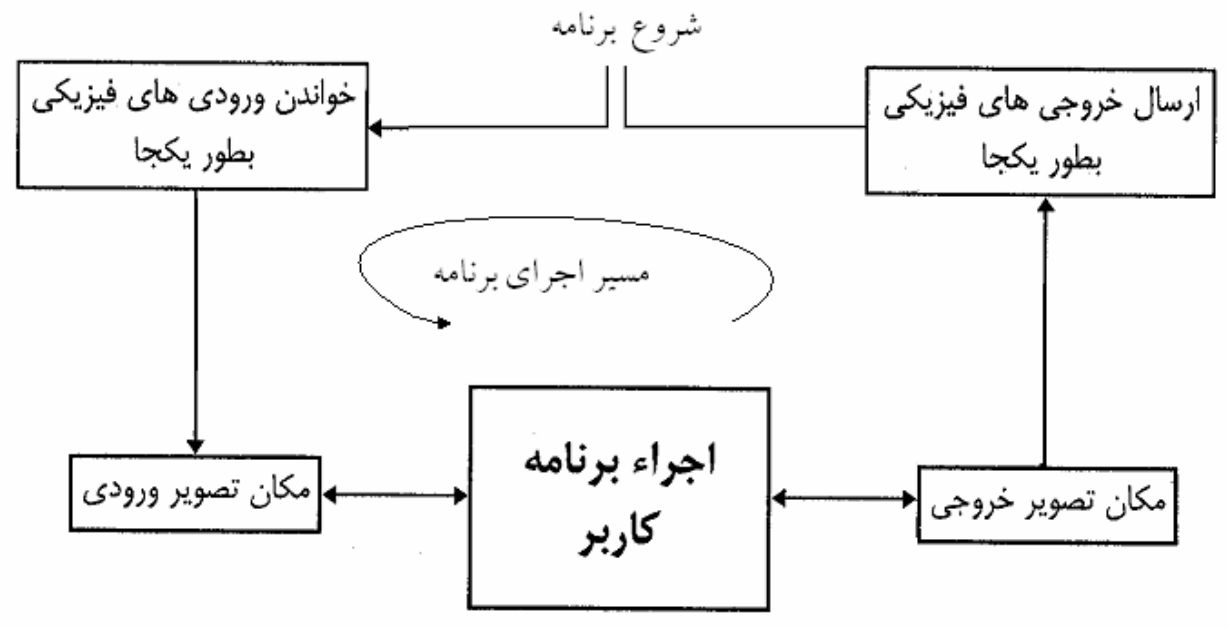
می کند و وضعیت آنها را در خانه های I\O RAM بنام (Input Image Area) ضبط می کند. این روند در ابتدا یا انتهای هر سیکل برنامه انجام می گیرد . شکل(۱-۹ ب).



شکل ۱-۹ اسکن کردن ورودی ها و زمان عکس العمل به طور نمونه : الف)به هنگام رسانی مداوم ب)کپی یک جای ورودی /خروجی

با اجرای برنامه ، داده های ورودی ذخیره شده در I/O RAM ، به صورت " یک خانه در هر لحظه " خوانده می شوند . بر روی این داده ها عملیات منطقی مورد لزوم انجام میگیرد و سیگنال های خروجی منتجه، در قسمت خروجی حافظه I/O RAM بنام (Output Image Area) ذخیره میشوند. سپس در انتهای هر سیکل برنامه ، روتین کپی کننده I/O ، همه سیگنال های خروجی موجود در I/O RAM را به کانال های خروجی مربوطه انتقال می دهد و طبقات خروجی متصل به واحد ورودی/خروجی را راه اندازی می کند . این طبقات خروجی به صورت قفل شده یا Latch شده هستند و وضعیت خود را تا اجرای مجدد روتین کپی کننده ورودی/خروجی حفظ می کنند. روتین اجرای برنامه در زیر آمده است.

کپی کردن یک جای ورودی /خروجی به طور اتوماتیک توسط CPU به عنوان یک زیر روتین از برنامه اصلی انجام میگیرد .(یک زیر روتین یا Subroutine برنامه ای کوچک است که برای انجام وظیفه خاصی طراحی شده و میتواند توسط برنامه اصلی فراخوانی شود. در اینجا زیر روتین ورودی/خروجی در محلی مابین انتهای یک سیکل برنامه و شروع مرحله بعدی آن انجام می گیرد.



شکل ۱- ۱۰ روتین اجرا ی برنامه

۱-۸ ملاحظات زمانی

توجه نمایید که به واسطه سیکلی بودن برنامه "کپی ورودی /خروجی " ، وضعیت ورودی ها و خروجی ها در طی اجرای هر سیکل برنامه قابل تغییر نیست . اگر یک سیگنال ورودی پس از روتین کپی تغییر یابد ، تا اجرای مرحله بعدی برنامه کپی قابل تشخیص نخواهد بود .

مدت زمان به هنگام سازی (update) همه ورودی/خروجی ها ، بستگی به تعداد کل ورودی /خروجی هایی دارد که بایستی کپی شود .

استفاده از روش کپی یک جای ورودی/خروجی ها در عین مزایایی که عنوان شد دارای معایبی نیز می باشد . زمان پاسخ دهی ویا scan cycle time زمانی است که طول می کشد تا PLC تمام برنامه کاربر را پویش نمایید ودر این مدت تغییرات بوجود آمده در ورودی ها وارد Input Image Area نمی شود و خروجی ها نیز به حالتی که در پویش قبلی بودند باقی می ماند. این امر ممکن است در فرآیند های که تغییرات سریع را تجربه می کنند، مشکل ساز باشد. مخصوصاً زمانی که برنامه کاربر طولانی است ومدت زمان زیادی صرف پویش و اجراءِ آن می گردد. همچنین گاهی ملاحظات ایمنی لازم می دارد که تغییرات آنی بعضی ورودی ها همواره مورد توجه قرار گیرد که زمان پاسخ دهی مانع از ثبت به موقع این تغییرات می شود.

نکته : سرعت PLC زمان اجرای ۱۰۰۰ خط دستور Logic مبنای سرعت PLC می باشد.

۱-۹ انواع PLC

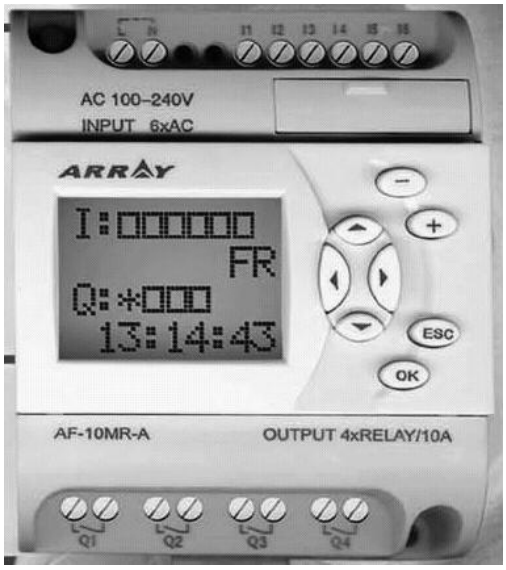
افزایش تقاضا از طرف صنایع برای PLC هایی که قابل به کارگیری در اشکال و ابعاد گوناگون وظایف کنترلی باشند ، سبب شده است که بیشتر تولید کنندگان گستره ای از PLC ها را با تسهیلات و سطوح عملیاتی متفاوتی به بازار عرضه دارند.

معیار اولیه مشخص کننده اندازه PLC ها ، در قالب حجم حافظه برنامه و حداکثر تعداد ورودی و خروجی هایی که سیستم قادر به پشتیبانی از آن هاست ، ارائه می شود.

اما به منظور ارزیابی و محک مناسب هر PLC ، باید خصوصیات دیگری از آن ، از قبیل نوع پردازشگر، زمان اجرای یک سیکل برنامه، تسهیلات زبان برنامه نویسی، توابع (از قبیل شمارنده ، تایمر و . . .) قابلیت توسعه و . . . را نیز در نظر بگیریم . در زیر چشم اندازی کلی از خصوصیات PLC های کوچک ، متوسط و بزرگ ، همراه با کاربردهای نوعی آنها آورده شده است.

۱-۹-۱ PLC های کوچک

معمولاً ، PLC های کوچک و "مینی PLC ها " (شکل ۱-۱۱) به صورت واحدهای قدرتمند ، کارا و فشرده ای طراحی می شوند که قابل جاسازی بر روی ، یا کنار تجهیزات تحت کنترل باشند، آنها عمدتاً به عنوان جایگزین سیستم های رله ای غیر قابل تغییر توسط اپراتور ، تایمر ، شمارنده و غیره مورد استفاده قرار می گیرند تا بخش های مجزا و منفرد کارخانجات یا ماشین آلات را کنترل کنند . اما می توان از آنها برای هماهنگ کردن عملکرد چند ماشین در تلفیق با یکدیگر سود جست .



شکل(۱-۱۱) نمونه ای از یک PLC کوچک

PLC های کوچک قادر به توسعه تعداد کانال های ورودی و خروجی با استفاده از یک یا دو ماژول ورودی /خروجی می باشند . لیکن چنانچه نیازی به افزایش بیشتر تعداد کانال ها باشد ، در این صورت باید PLC را با PLC کامل تر و بزرگ تری تعویض نمود .

ذخیره برنامه در این PLC ها توسط EEPROM یا RAM دارای باتری پشتیبان صورت می گیرد. در حال حاضر گرایش به سمت حافظه های EEPROM همراه با ابزارهای برنامه نویسی است که همر اه با خود PLC عرضه شود .

۱-۹-۲ PLC های متوسط

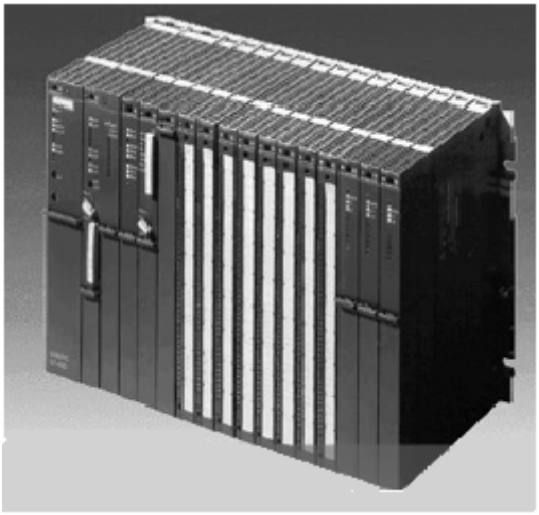
با بزرگ شدن پروسه کنترلی وافزایش تعداد ورودی/خروجی ها دیگر PLC های کوچک جوابگو نیستند. بطور معمول وقتی تعداد ورودی خروجی ها ی پروسه بین ۱۰۰ تا ۱۰۰۰ باشد از PLC های متوسط استفاده می شود. وقتی در این گستره ، ساخت PLC ها با استفاده از ساختار ماژولار رایج است. این ساختار به سادگی امکان توسعه سیستم را تنها با افزودن کارت های ورودی /خروجی به نصبگاه PLC میسر ساخته است. چرا که بیشتر سیستم های نصبگاهی ، فضای لازم برای چندین کارت اضافی را دارا است .

بوردها را اغلب در برابر شرایط نامطلوب طبیعی و مکانیکی مقاوم گردانیده اند تا عملکرد مطمئن دستگاه را به محدوده ای از تغییرات محیطی فراهم آورند . (شکل ۱-۱۲)

شکل (۱-۱۲)PLC متوسط

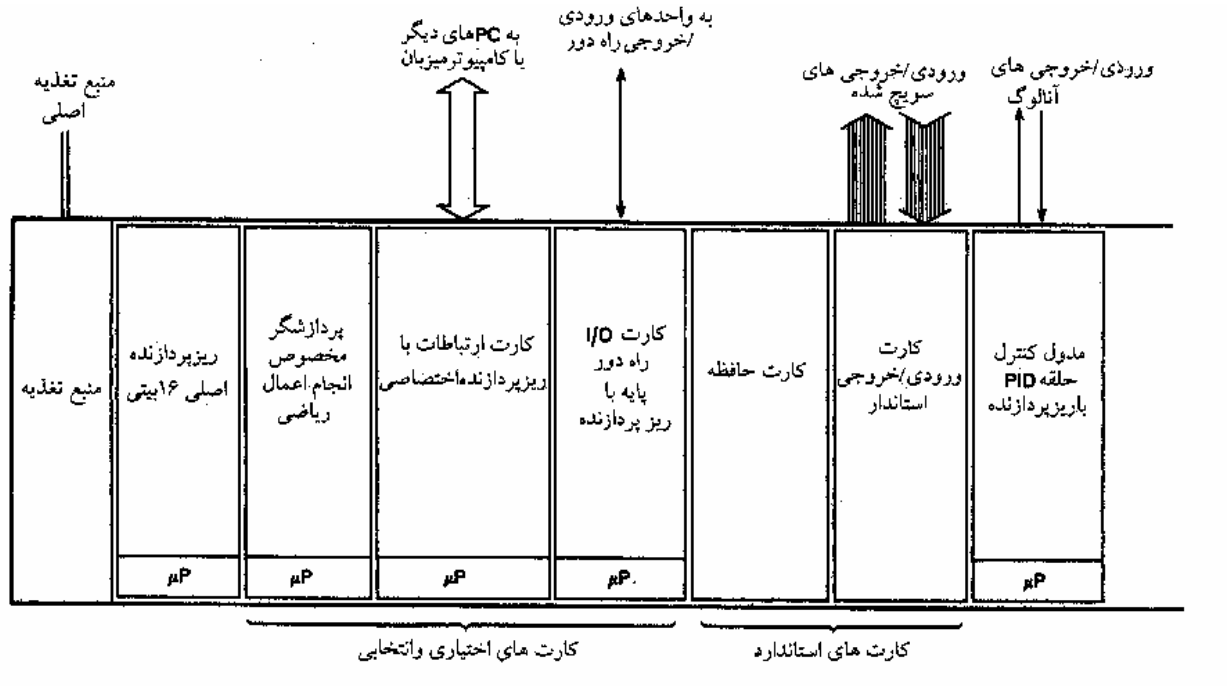
۱-۹-۳ PLC های بزرگ

در مواردی که کنترل تعدا د زیادی از ترمینال های ورودی /خروجی مد نظر بوده و یا به توابع کنترلی پیچیده نیاز باشد، ضرورت کاربرد PLC های بزرگ کاملاً مشهود خواهد شد. از این PLC های به عنوان کنترل کننده ناظر برای نظارت و مونیتور کردن چندین PLC دیگر یا سایر ماشین های هوشمند (نظیر ماشین های CNC)به کار می روند.



شکل (۱-۱۳)یک PLC بزرگ ماژولار

در این PLC ها ساختار ماژولار همراه با گستره وسیعی از کارت های توابع قابل دسترس، (همانند ماژول های ورودی/خروجی آنالوگ) ساختار استاندارد می باشد. به منظور کاربرد مؤثرتر این PLC ها در محدوده گسترده ای از وظایف مختلف کنترلی حرکتی به سوی استفاده از پردازشگر های ۱۶ بیتی و همچنین تکنیک چند پردازشگره توسط سازندگان صورت گرفته است. شکل (۱-۱۴)



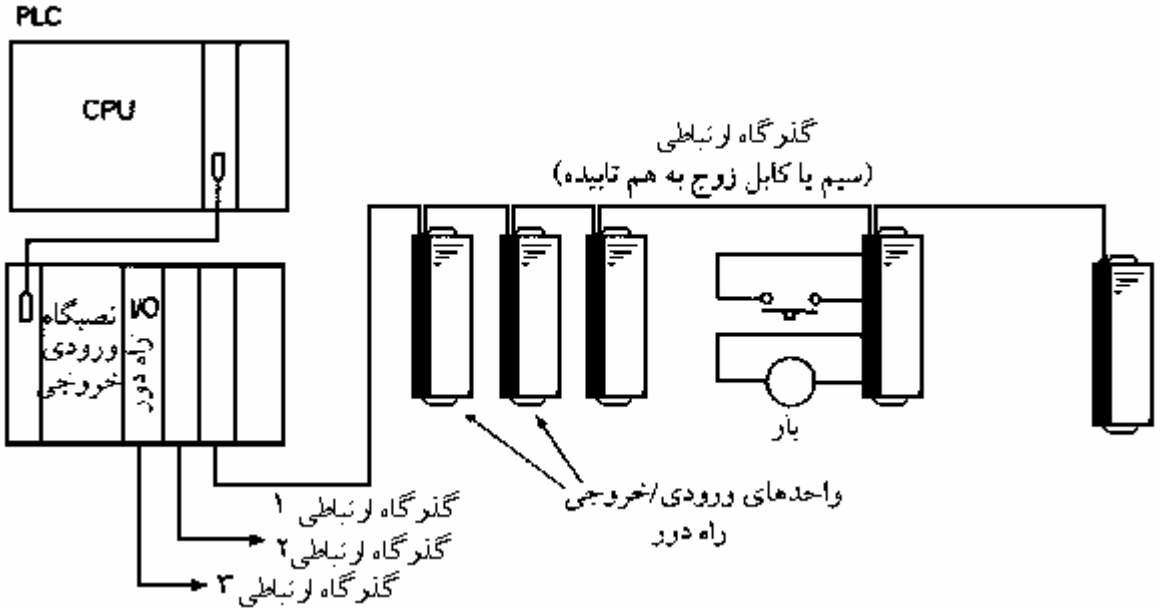
شکل (۱-۱۴)

- پردازشگر 16 بیتی به عنوان پردازشگر اصلی جهت محاسبات دیجیتالی و همچنین به کارگیری متن .
- پردازشگر تک بیتی به عنوان پردازشگر همکار برای محاسبه سریع ، ذخیره سازی و ....
- پردازشگر های جانبی ، برای انجام وظایف اضافی که تابع زمان می باشند و یا زمان آنها امر حیاتی محسوب می شود مانند:
  - کنترل حلقه بسته PID
  - کنترل موقعیت
  - محاسبات عددی با ممیز شناور
  - تشخیص عیب و monitoring
  - ارتباطات بین ماشین های هوشمند برای ورودی /خروجی توزیع شده
  - نصبگاه های ورودی / خروجی با فاصله دور

شیوه چند پردازه در PLC های بزرگ سبب می شود که عملکرد سیستم به خصوص در زمینه تنوع کاربردها و سرعت پردازش بهینه باشد. با این روش PLC قادر خواهد بود برنامه های بزرگ تا ۱۰۰k دستوالعمل یا بیشتر را مدیریت و اجرا نماید. هم اکنون کارت های حافظه ، چندین مگا بایت حافظه را در قالب CMOSRAM یا EPROM فراهم می آورند.

۱-۱۰ ورودی/ خروجی راه دور

زمانی که تعداد متعددی از ترمینال های ورودی /خروجی در مسافت قابل ملاحظه ای دور از PLC جای داده می شوند، کابل کشی به تک تک ترمینال ها ، کاری غیر اقتصادی (ونیز جاگیر) است. یک راه حل برای این مسأله قرار دادن یک واحد ورودی /خروجی راه دور در نزدیکی ترمینال های ورودی /خروجی است. این واحد به صورت متمرکز کننده عمل می کند و همه ورودی ها را رصد کرده و وضعیت آن ها را از طریق یک اتصال ارتباطی سریال به PLC انتقال می دهد.زمانی که PLC سیگنال های خروجی را تولید کرد، آن ها از کابل ارتباطی به واحد ورودی /خروجی راه دور برگشت داده می شوند. در این واحد داده های سریال به سیگنال های خروجی متناظر تبدیل شده و جهت راه اندازی فرآیند استفاده می گردند.مطابق شکل(۱-۱۵).



شکل(۱-۱۵) ورودی / خروجی های راه دور

پرسشهای فصل اول

۱. انواع PLC ها را نام ببرید.

۲. سه قسمت اصلی سخت افزار PLC را نام ببرید.

۳. ارتباط با واحد ورودی و خروجی ها به چند صورت انجام می شود.

فصل دوم

برنامه نویسی STEP7

اهداف آموزشی

۱. آشنایی با نرم افزار SIMATIC Manager
۲. آشنایی با زبان برنامه نویسی STEP7

مقدمه :

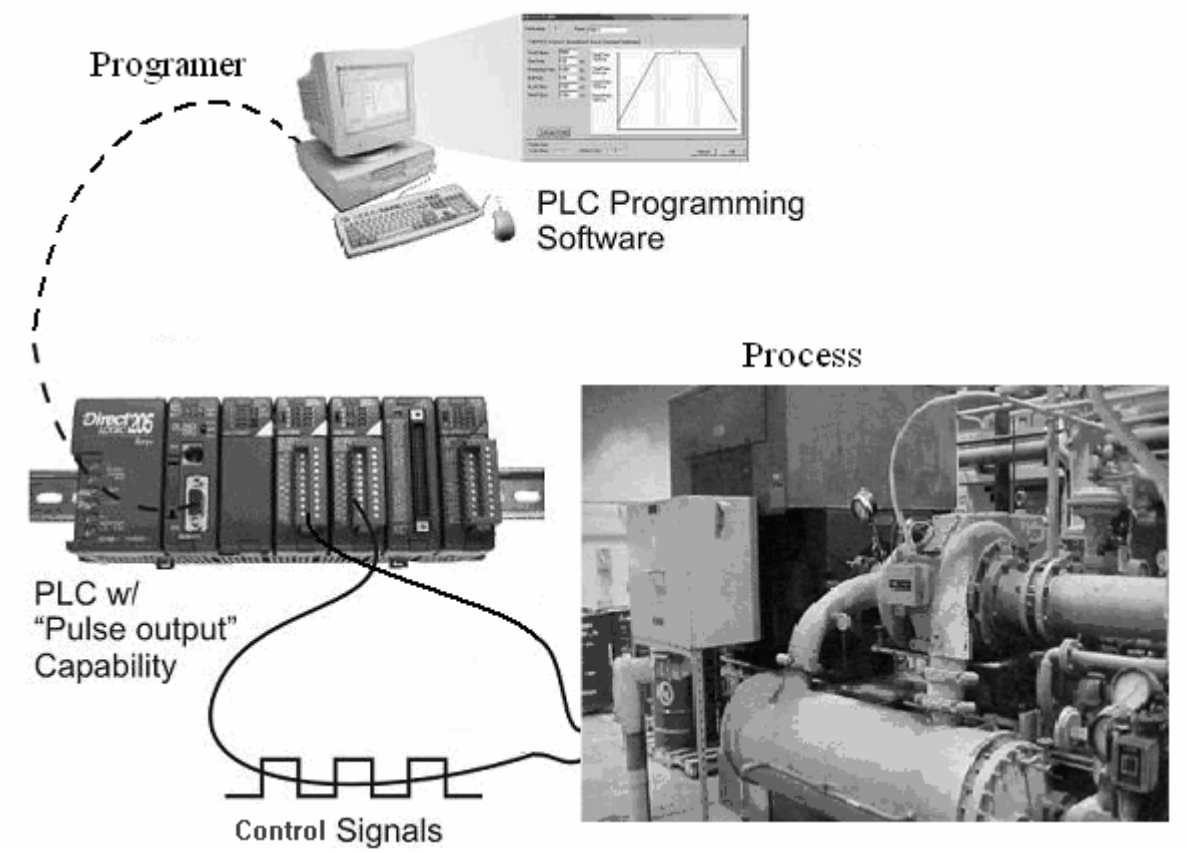
Step7 نرم افزار استاندارد شرکت زیمنس برای نوشتن برنامه های کنترلی به زبانهای Statement Ladder,Logic , Function Block Diagram , بر روی PLC های سری 300/400 S7- میباشد .

در این فصل چگونگی برنامه نویسی با زبان های برنامه نویسی Function Block Diagram, Statement List , Logic Ladder در STEP 7 آشنا خواهید شد.

۲-۱ ترکیب سخت افزار و نرم افزار

با استفاده از نرم افزار STEP 7 میتواند برنامه خود را در غالب یک پروژه ایجاد نماییدPLC های S7 از یک منبع تغذیه ، یک CPU و ماژولهای ورودی و خروجی (I/O Modules) تشکیل شده اند .

وظیفه اصلی PLC ها کنترل پروسه مربوطه بوسیله برنامه نوشته شده واعمال آن به پروسه از طریق ماژول های I/O می باشد. همچنین PLC ها توانایی مونیتورینگ سایر قسمت ها را دارا می باشند.



شکل (۲-۱) شکل کلی یک پروسه کنترلی توسط یک PLC

۲-۲ روش استفاده از STEP 7

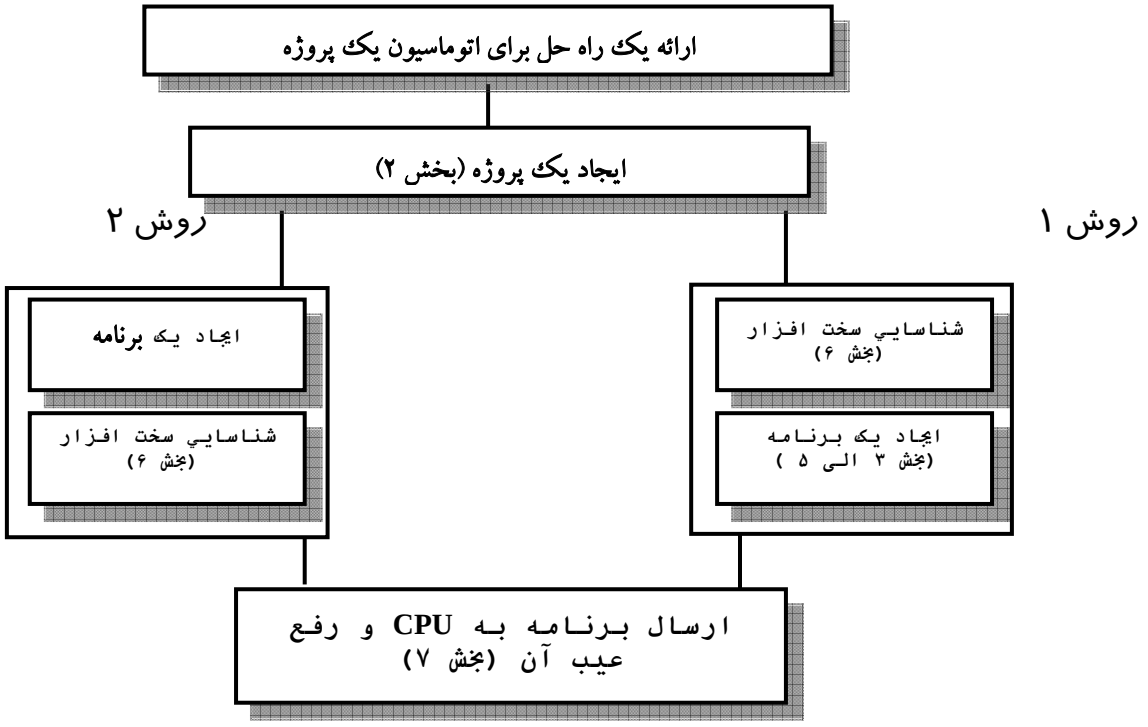
قبل از ایجاد یک پروژه لازم است بدانید که پروژه های STEP 7 به روشهای متفاوتی میتوانند ایجاد شوند .

اگر شما در حال ایجاد یک برنامه با مقدار زیادی ورودی و خروجی میباشید توصیه می شود که ابتدا شناسایی سخت افزار را انجام دهید ، مزیت این کار آن است که STEP 7 آدرس های ممکن را در شناسایی سخت افزار نشان می دهد .

اگر روش دوم را انتخاب نمایید آنگاه می بایست خودتان آدرس ها را بر اساس اجزاءِ انتخابی تعیین کنید و دیگر نمیتوانید آدرس ها را از طریق STEP 7 فراخوانی کنید .

در مرحله تنظیم سخت افزار نه تنها میتوانید آدرسها را تعیین کنید بلکه تغییر پارامترها و مشخصات ماژولها نیز ممکن میباشد، برای این منظور اگر میخواهید از چند CPU استفاده نمایید میبایست آدرس های CPU , MPI ها را با یکدیگر هماهنگ نمایید .

از آنجایی که در مثالهای این جزوه تنها از تعداد محدودی ورودی و خروجی استفاده میشود از مرحله شناسایی سخت افزار صرف نظر کرده و کار را با آموزش برنامه نویسی آغاز میکنیم .



۲-۳ نرم افزار SIMATIC Manager

۲-۳-۱ آغاز به کار برنامه SIMATIC Manager و ایجاد یک پروژه

SIMATIC Manager پنجره اصلی میباشد و هنگامیکه STEP7 اجرا میگردد فعال میشود تنظیمات اولیه بگونه ای است که STEP7 Wizard بطور خودکار راه اندازی شده و شما را هنگام ایجاد پروژه هایSTEP7 پشتیبانی میکند . ساختار پروژه برای ذخیره و منظم کردن تمامی داده ها و برنامه ها استفاده میگردد.

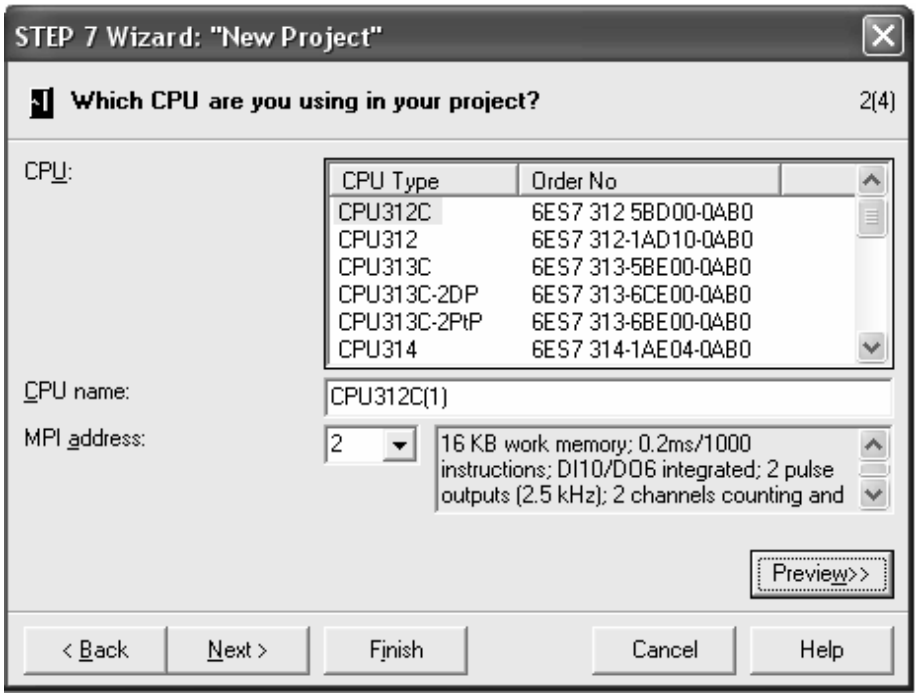
برنامه SIMATIC Manager را اجرا نمایید.در پنجره STEP7 Wizard در بخش Preview میتواند نمایش ساختار پروژه در حال ایجاد را فعال و یا غیر فعال نمایید. شکل (۲-۲)

شکل (۲-۲) STEP7 Wizard

با زدن Next به پنجره بعدی می روید و می توانید برای ایجاد پروژه نوع CPU را انتخاب نمایید.

این پروژه بگونه ای ایجاد میگردد که میتوانید هر نوع CPU که در اختیار دارید را انتخاب کنید

تنظیم پیش فرض برای آدرس MPI ، ۲ میباشد .



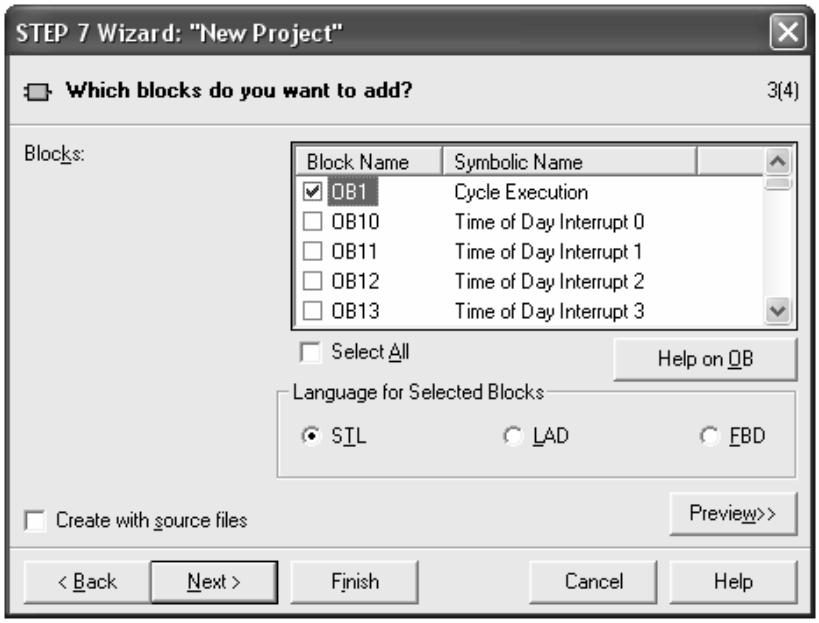
شکل (۲-۳)

نکته :هر CPU مشخصات مخصوص به خود از جمله تنظیمات حافظه و آدرس دهی را دارا می باشد به همین دلیل است که قبل از آغاز برنامه نویسی میبایست نوع CPU را تعیین نمود. آدرس MPI نیز برای ارتباط میان CPU و دستگاه برنامه ریزی یا PC مورد نیاز است.

برای تایید تنظیمات انجام شده کلید Next را فشار داده و به پنجره بعدی بروید.

در پنجره جدید OB1 را انتخاب نمایید و سپس یکی از زبانهای برنامه ریزی را از میان , STL LAD یا FBD انتخاب کنید . OB1 نمایانگر بالاترین سطح برنامه نویسی بوده و بقیه بلوکها را در برنامه S7 سازماندهی میکند شما می توانید زبان برنامه نویسی را در صورت لزوم دوباره تغییر دهید. شکل(۲-٤)

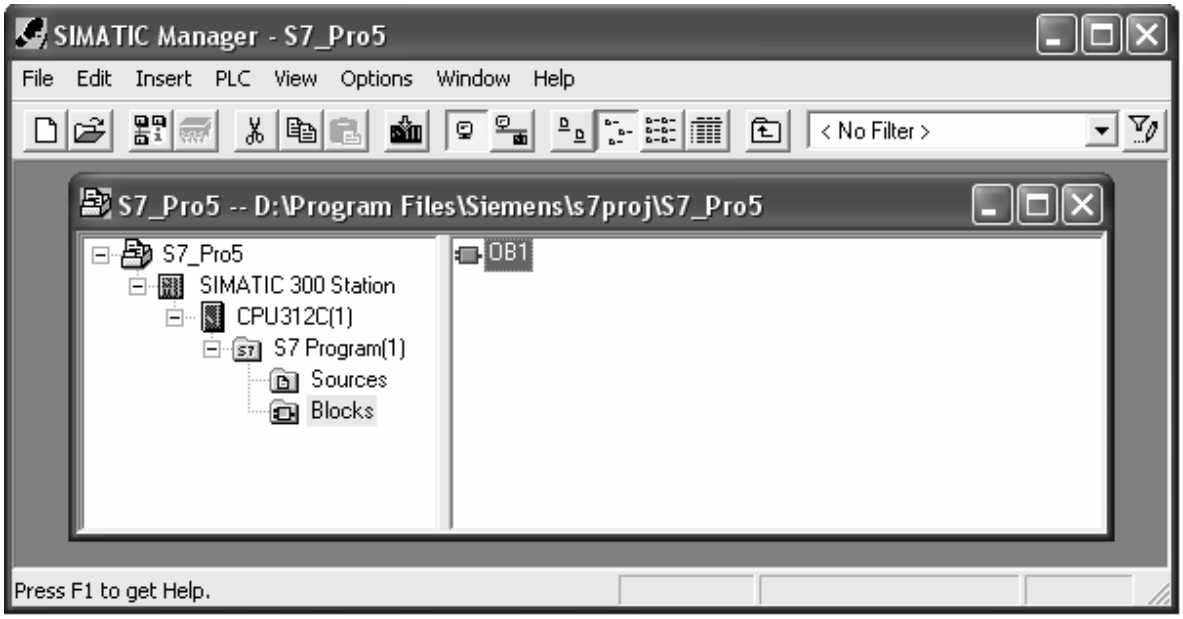
تنظیمات خود را با فشار کلید Next تایید نمایید .



شکل(۲-۴)

در صفحهٔ جدید نام پروژه را در کادر نام پروژه تایپ نمایید ، بر روی Finish کلیک کرده تا پروژه مورد نظر ایجاد گردد.

هنگامیکه دکمه Finish را فشار میدهید Simatic Manager پنجره شروع به کار را که شما ایجاد کرده اید باز میکند در صفحات بعد به شما نشان میدهیم که فایل ها و پوشه های ایجاد شده چه کاربردی دارند و چگونه می توان با آنها کار کرد Wizard STEP7 هر بار که برنامه اجرا میشود فعال میگردد شما میتوانید پیش فرض خود را در اولین پنجره محاوره ای غیر فعال نمایید در این صورت می بایست تمامی دایرکتوری مورد نیاز در پروژه را خودتان ایجاد نمایید . شکل (۲-۵).

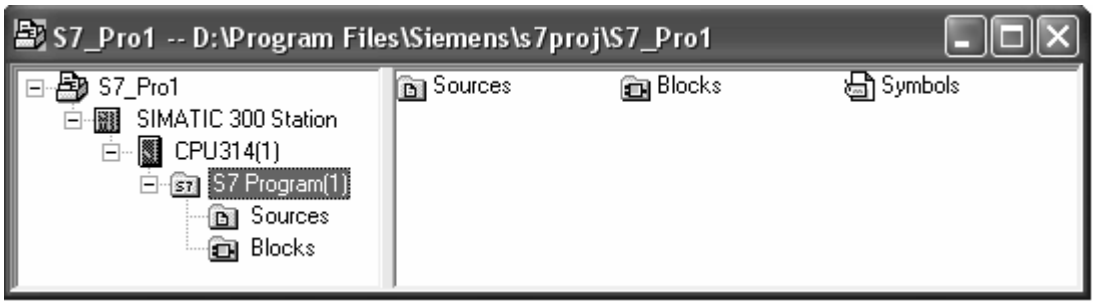


شکل(۲-۵)اجزای یک پروژه

ساختار پروژه در Simatic Manager و طریقه فراخوانی Help بصورت Online همزمان با بسته شدن Simatic Manager , SETUP Wizard به همراه پنجره پروژه First Project ظاهر میگردد حال میتوان تمامی پنجره ها و توابع SETUP را استفاده نمایید .

پروژه ای که اکنون ایجاد نمودید به همراه CPU انتخابی S7 نمایش داده میشود با کلیک کردن بر روی + و یا – میتوانید پوشه ها را باز و بسته نمایید شما همچنین قادر میباشید که توابع و قابلیت های دیگر را نیز با کلیک کردن روی نمادهای سمت راست پنجره فعال نمایید .

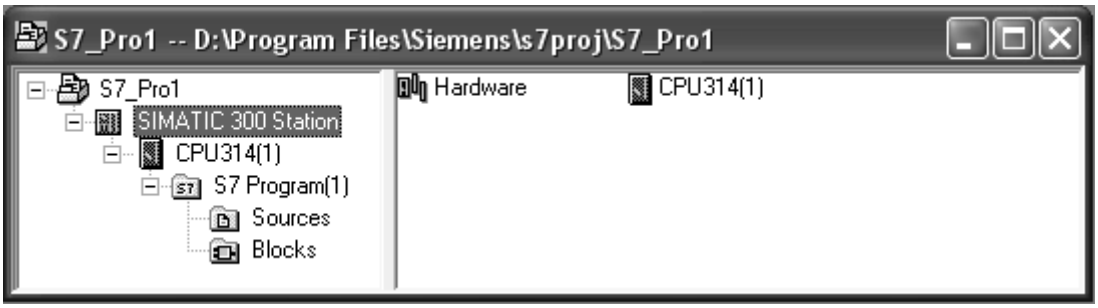
بر روی پوشه "S7 Program 1" کلیک نمایید این پوشه حاوی اجزای ضروروی برنامه میباشد.شکل (۲-۶).



شکل (۲-۶)

فایل Symbols حاوی نامهای نمادین که به آدرس های مختلف تخصیص داده شده اند می باشد. روی پوشه Block کلیک نموده این پوشه حاوی بلوک OB 1 که هم اکنون ایجاد گردید و نیز شامل بقیه بلوک هایی که ایجاد خواهید کرد میباشد. مطابق شکل (۲-۷) بر روی پوشه

Simatic 300 Station کلیک نمایید تمامی داده های سخت افزاری مربوط به پروژه در آنجا ذخیره میگردند.

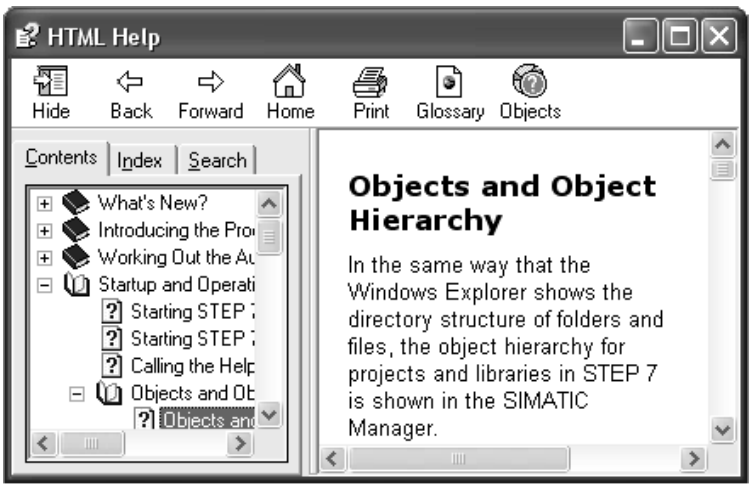


شکل (۲-۷)

حال می توانید برنامه نویسی با Statement List , Ladder Logic Diagram و یا Function Block را آغاز نمایید

۲-۳-۲ فراخوانی Help در SETUP

روش۱ استفاده از F1 : نشانگر ماوس را روی منوی فرمان دلخواه قرار داده و کلید F1 را فشار دهید، مطابق شکل(۲-۸) Help مربوط به آن منوی فرمان ظاهر میگردد .



شکل(۲-۸) استفاده از Help

روش ۲: با استفاده از منو ،onlin-help را باز کرده صفحه ای حاوی عناوین متعدد در قسمت چپ پنجره ظاهر میگردد و عنوان انتخاب شده در سمت راست پنجره ظاهر می گردد و عنوان انتخاب شده در سمت راست پنجره نمایش داده میشود با کلیک کردن روی علامت + عنوان مورد نظر را یافته و آن را انتخاب کنید محتوای عنوان انتخابی در سمت راست نمایش داده میشود با استفاده از index , find میتوانید عنوان مورد نیاز خود را جستجو نمایید .

روش ۳ : با کلیک کردن روی علامت سوال موجود در قسمت toolbar نشانگر ماوس به علامت سوال تغییر کرده و با کلیک بر روی هر شیئی دلخواه help مربوط به آن فعال میگردد .

۲-۴ برنامه نویسی با استفاده از نمادها (Symbols)

۲-۴-۱ آدرس دهی مطلق (Absolute Addresses)

هر ورودی و یا خروجی دارای یک آدرس حقیقی می باشد که در تنظیمات سخت افزار تعیین شده اند این آدرسها بصورت مستقیم اختصاص داده میشود. این آدرسهای مطلق را میتوان با هر نام و نماد دلخواه جایگزین کرد.

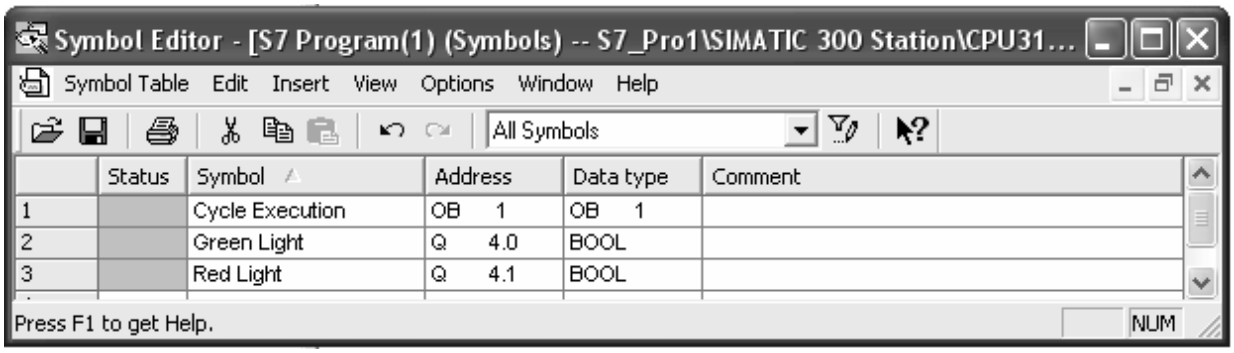
۲-۴-۲ برنامه نویسی نمادین (Symbolic Programming)

از منوی نمادها میتوانید به تمامی آدرسهای مطلق که در برنامه خود از آنها استفاده میکنید یک نام اختصاص داده و نوع داده را نیز مشخص نمایید برای مثال برای ورودی I 0.1 نام نمادین Key1 را میتوان اختصاص داد این نامها در تمامی قسمتهای برنامه شناخته شده میباشند و مانند متغیرهای عمومی عمل مینمایند با استفاده از این نمادها میتوانید درک برنامه خود را بطور قابل ملاحظه ای افزایش دهید .

کار با ویرایشگر نمادها در پنجره پروژه " First Project "منوی (۱) S7 Program را یافته و بر روی Symbols دو بار کلیک نمایید.

جدول نمادها در حال حاضر تنها از بلوک OB1 که از پیش تعریف شده تشکیل شده است .

مطابق شکل (۲-۹) نام "Green Light"و"Q4.0" را در ردیف دوم وارد کنید نوع این داده بطور اتوماتیک اضافه میگردد. نام " Red Light" یا هر نام دلخواه دیگر را به خروجی " Q4.1" اختصاص داده و کلید Enter را بزنید. بر روی ستون Comment در ردیف ۱ یا ۲ کلیک کرده و توضیحات خود را درباره آن نماد وارد نمایید.



شکل (۲-۹)

با این روش میتوانید به تمامی آدرسهای مطلق ورودی ها و خروجی ها که در برنامه استفاده میکنید نام های نمادین اختصاص دهید.

داده ها و تغییراتی را که در جدول نمادها وارد کرده اید را ذخیره کرده و پنجره مربوطه را ببینید.

بطور کلی یک جدول نماد برای هر برنامه S7 وجود دارد که ارتباطی به زبان برنامه نویسی ندارد . استفاده از تمامی کاراکترهای قابل چاپ از جمله کاراکترهای مخصوص در جدول نمادها مجاز میباشد .

نوع داده که بطور اتوماتیک در جدول نمادها اضافه میگردد نوع سیگنالی را که توسط CPU می بایست پردازش گردد را مشخص مینماید STEP7 از نوع داده های زیر استفاده می کند.

در مورد انواع داده های مورد استفاده در برنامه نویسی PLC در بخش های آتی اشاره خواهد شد.

۲-۵ ایجاد برنامه در OB1

۲-۵-۱ باز کردن پنجره برنامه نویسی با LAD\STL\FBD

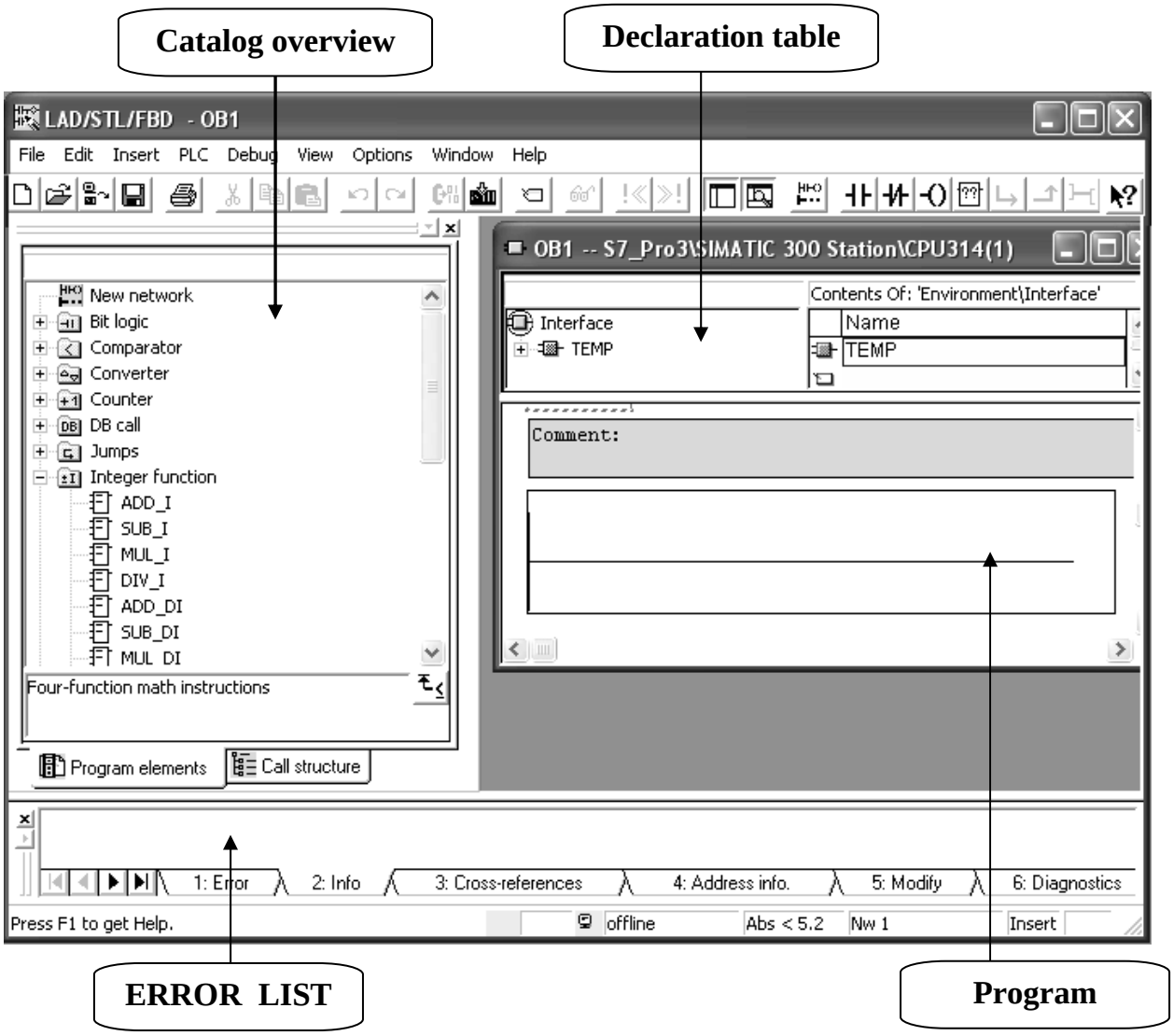
STEP7 زبان های برنامه نویسی مختلفی از قبیل Statement List , Ladder Logic و یا Function Block Diagram دارد. میتوان برنامه های خود را به یکی از زبانهای LAD,STL, FBD نوشت.

با دو بار کلیک کردن OB1 بلوک OB1 مطابق با زبان برنامه نویسی که انتخاب گردیده (LAD,STL یا FBD) باز میشود شما میتوانید زبان برنامه نویسی پیش فرض را در مواقع لزوم تغییر دهید .

Scan cycle Time : در STEP7 ، بلوک OB1 بطور مداوم (cyclically) توسط CPU خط به خط برنامه را خوانده و آن را اجراء میکند زمانیکه CPU به خط اول برنامه بازگشت یک سیکل اجرا کامل میگردد و زمان انجام این سیکل "Scan cycle Time" نامیده میشود .

۲-۵-۲ اجزای پنجره OB1

۱. **Catalog overview:** که در آن لیست دستورات را می توان مشاهده نمود
۲. **Decleration table :** محل تعریف متغیرهاست وتمام این متغیر ها Local هستند. با توجه به اینکه ۲۰ بایت اول توسط خود OB ها استفاده می شود، اگر temp را باز کنیم چند متغیر از پیش تعریف شده وجود دارد.
۳. **ERROR LIST :** لیست خطاهای برنامه نویسی را بعد از Compile کردن نمایش می دهد.



شکل (۲-۱۰) اجزای پنجره OB1

۲-۵-۳ برنامه نویسی OB1 در محیط Ladder logic

در این بخش شما مدارهای سری و موازی و حافظه Set/reset را برنامه نویسی مینمایید .

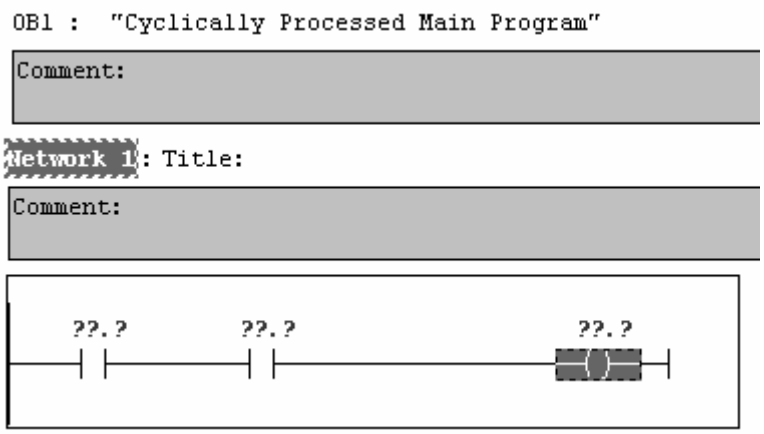
۲-۵-۳-۱ برنامه نویسی یک مدار سری در Ladder Logic

در صورت لزوم در منوی view زبان برنامه نویسی را به LAD تغییر دهید .

مطابق شکل (۲-۱۱) در صفحه باز شده بر روی قسمت عنوان (Title) در OB1 کلیک کرده و جمله " Cyclically Processed Main Program" را بطور مثال وارد نمایید. در قسمت comment توضیحات دلخواه مربوط به آنرا را درج نمایید این توضیحات در برنامه بی تاثیر می باشند وحداکثر تا ۲۰۴۸ کاراکتر می توان نوشت.

به هر قسمت از برنامه در S7 ، Network گویند. Network معادل Segment در S5 می باشد.

بر روی مسیر جاری(خط افقی) برای قرار دادن اولین عنصر خود کلیک کرده تا پر رنگ گردد. سپس کلید کنتاکت باز را در منوی toolbar کلیک کرده تا در در مسیر قرار گیرد. به همین ترتیب دومین کنتاکت باز را قرار دهید. یک خروجی در انتهای راست مسیر قرار دهید.

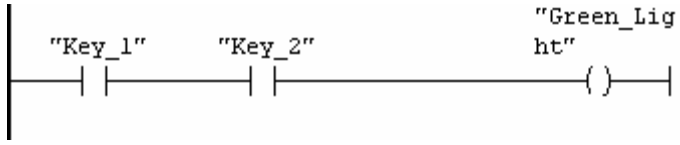


شکل (۲-۱۱)برنامه یک مدار سری در **Ladder Logic**

آدرسهای کنتاکتها و خروجی در این مدار سری مورد نیاز است .

بر روی علامت ؟؟؟ در شکل (۲-۱۲) کلیک کرده و نام نمادین "Key_1" را وارد کرده و کلید Enter را فشاردهید . نام نمادین "Key_2" را نیز برای کنتاکت باز دوم وارد نمایید. نام "Green Light" را برای خروجی وارد نمایید .

حال شما یک مدار سری را بطور کامل برنامه نویسی کرده اید . در صورتیکه هیچ نماد دیگری با رنگ قرمز نشان داده نشود بلوک را Save نمایید .



شکل (۲-۱۲) اختصاص آدرس به متغیر ها

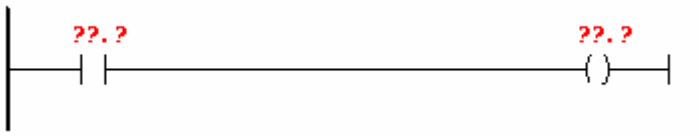
نکته : نمادها در صورتیکه در جدول نمادها موجود نباشند و یا خطای Syntax داشته باشند به رنگ قرمز نمایش داده میشود .

شما همچنین میتوانید نام نمادین را بطور مستقیم از جدول نماد وارد نمایید برای استفاده از این روش بر روی علامت ؟؟؟ کلیک کرده و سپس منوی insert\symbol را انتخاب کنید ، حال از میان نمادهای موجود نماد مربوطه را پیدا کرده و انتخاب کنید .

برای ارسال برنامه فوق به Plc ابتدا گزینه Plc\Download را انتخاب میکنیم ، سپس Plc را بر روی Run می گذاریم.

۲-۳-۵-۲ برنامه ریزی یک مدار موازی در Ladder logic

Netwok1 جدید را وارد نمایید. مسیر جاری را دوباره انتخاب نمایید . یک کنتاکت باز و یک خروجی را انتخاب نمایید .



خط عمودی ابتدای مسیر جاری را انتخاب کنید .

یک شاخه موازی با انتخاب آیکون روبرو وارد نمایید. یک کنتاکت باز دیگر در شاخه موازی وارد کنید.

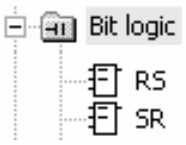
شاخه موازی را با انتخاب آیکون ببندید.

حال آدرس نمادها را مانند آنچه در ایجاد مدار سری انجام دادید وارد نمایید .

کنتاکتها را " Key_3 " و " Key_4 "و خروجی را "Red_Light " بنامید. بلوک را Save نمایید .

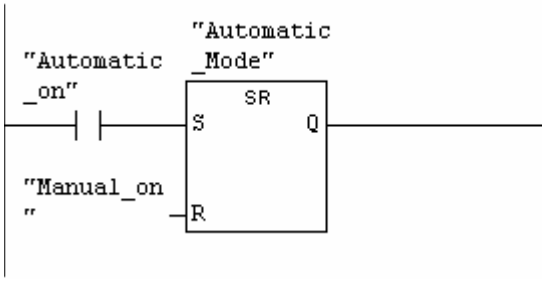
۲-۳-۵-۳ برنامه ریزی یک تابع حافظه در ladder logic

یک Network جدید ایجاد نمایید. مسیر جاری را انتخاب نمایید. مطابق شکل (۲-۱۳) در پنجره Catalog اجزاء برنامه در زیر شاخه " Bit logic " حافظه SR را یافته و آن را وارد نمایید .



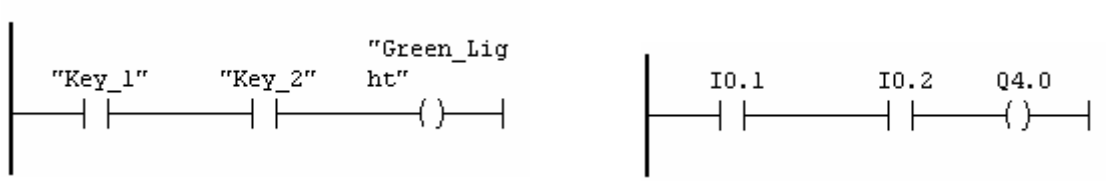
شکل(۲-۱۳) نحوه انتخاب تابع حافظه

یک کنتاکت باز در مقابل هر کدام از ورودی های S وR قرار دهید. نام های نمادین زیر را مطابق شکل (۲-۱٤) واردکرده و سپس بلوک را Save نمایید.



شکل(۲-۱۴) برنامه ریزی یک تابع حافظه در ladder logic

نکته : برای مشاهده تفاوت میان آدرس دهی مطلق و نمادین میتوانید Symbolic representation را در منوی View\Display غیر فعال نمایید.



آدرس های نمادین آدرس های حقیقی

شکل(۲-۱۵) تفاوت بین آدرس دهی مطلق و نمادین

شما میتوانید تعداد کاراکترهای موجود در هر خط از آدرس های نمادین را در منوی Options\Customize تعیین نمایید . برای این کار میبایست در "width of address field" تغییرات لازم را ایجاد نمایید این تغییرات میتواند بین ۱۰ الی ۲٤ کاراکتر باشد .

۲-۵-۴ برنامه ریزی OB1 در Statement List

در این بخش شما دستورات AND , OR و حافظه Statement List برنامه ریزی میکنید .

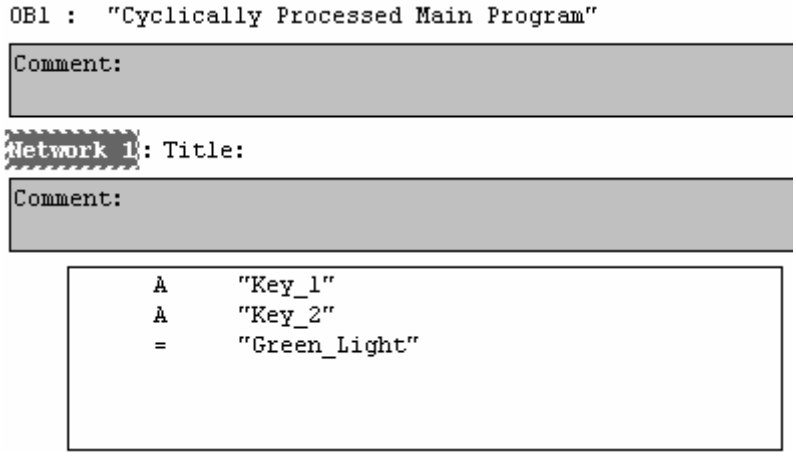
۲-۴-۵-۱ برنامه ریزی دستور العمل AND در محیط Statement List

ابتدا در منوی View زبان برنامه نویسی را به STL تغییر دهید. دقت کنید که گزینه "نمایش نمادین "(Symbolic Representation) فعال باشد.

مطابق شکل(۲-۱۶) عنوان OB1 را در قسمت "Title" به "cyclically processed main program" تغییر دهید.

محیط وارد کردن اولین Statement خود را انتخاب نمایید. حرف A که مخفف AND میباشد را در اولین خط برنامه تایپ نمایید ، و پس از یک فاصله نام نمادین "Key_1" را نیز وارد نمایید .

خط برنامه را با زدن دکمه Enter کامل نمایید با این عمل نشانگر به خط بعدی میرود به همین روش دستورالعمل AND را مطابق شکل کامل نمایید .

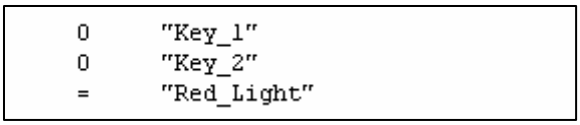


شکل(۲-۱۶) برنامه ریزی دستور العمل AND در محیط STL

حال شما یک دستورالعمل AND را بطور کامل برنامه ریزی کرده اید در صورتیکه هیچ نمادی با رنگ قرمز نشان داده نشود بلوک مورد نظر را ذخیره نمایید .

۲-۴-۵-۲ برنامه ریزی دستورالعمل OR در Statement List

Network1 را انتخاب نمایید. یک Network جدید را وارد کرده و محیط ورودی را دوباره انتخاب نمایید. حرف O که مخفف OR میباشد و نام نمادین "Key_3" را مانند همان روشی که برای دستور العمل AND بکار بردید وارد نمایید. دستورالعمل OR را مطابق شکل۲-۱۷ تکمیل نمایید .



شکل (۲-۱۷)برنامه دستورالعمل **OR** در **STL**

۲-۵-۴-۳ برنامه ریزی دستورالعمل حافظه در محیط Statement List

یک Network جدید را وارد نمایید. مطابق شکل (۲-۱۸) در خط اول برنامه دستور A به همراه نام نمادین "Automatic" وارد نمایید. دستور العمل حافظه را مطابق شکل کامل نموده و ذخیره نمایید سپس Block مورد نظر را ببندید.

```
A     "Automatic_on"
S     "Automatic_Mode"
A     "Manual_on"
R     "Automatic_Mode"
```

شکل (۲-۱۸)برنامه دستورالعمل حافظه در STL

نکته: برای مشاهدات تفاوت میان آدرس دهی مطالق و نمادین منوی فرمان View\display\symbolic Representation را غیر فعال نمایید .

```
A     I     0.1
S     Q     4.0
A     I     0.2
R     Q     4.0
```

شکل (۲-۱۹) آدرس دهی مطلق در STL

۲-۵-۵ برنامه ریزی OB1 توسط (FBD) Function Block Diagram

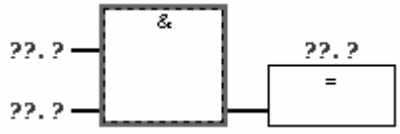
در این روش توابع OR,ANDو حافظه SR را توسط FBD برنامه ریزی میکنیم .

۲-۵-۵-۱ برنامه ریزی تابع AND در FBD

در صورت لزوم در منوی view گزینه FBD را برای زبان برنامه نویسی انتخاب نمایید .

عنوان OB1 را در قسمت Title به "cyclically processed Main Program" تغییر دهید .

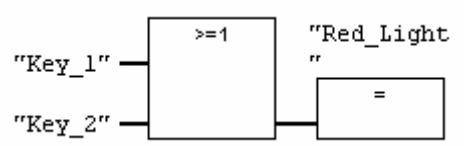
محیط وارد کردن تابع AND را انتخاب نمایید. نماد AND و (=) را وارد نمایید.



شکل (۲-۲۰)برنامه ریزی تابع **AND** در **FBD**

۲-۵-۵-۲ برنامه ریزی تابع OR در FBD

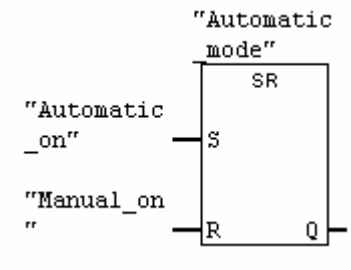
یک Network جدید وارد نمایید. نماد OR و (=) را وارد نمایید. آدرس های تابع OR را وارد نمایید. نام "Key_1" را برای ورودی بالایی و " Key_2 " را برای ورودی پائینی و نیز " Red _ Light " را برای خروجی وارد نمایید . بلوک مورد نظر را ذخیره نمایید .



شکل (۲-۲۱) برنامه ریزی تابع **AND** در **FBD**

۲-۵-۵-۳ برنامه ریزی تابع حافظه در FBD

یک Network جدید وارد نمایید و محیط ایجاد تابع حافظه را نیز انتخاب نمایید . در قسمت Bit Logic در کاتالوگ اجزاء برنامه (Program Elements Catalog) حافظه SR را بیابید و با دو بار کلیک کردن آن را وارد نمایید. نام های نمادین زیر را برای اجزاء حافظه SR انتخاب نمایید. بلوک مورد نظر را ذخیره کرده و پنجره را ببندید.

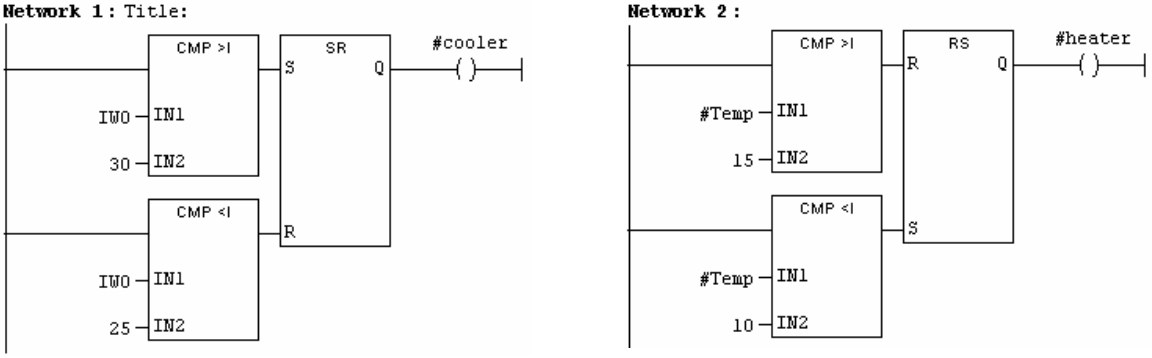


Set	"Automatic_On"
Reset	"Manual_On"
Memory bit	"Automatic_Mode"

شکل (۲-۲۲) برنامه ریزی تابع حافظه در FBD

مثال۱ . برنامه ای برای کنترل دما بنویسید که اگر دما از $^{\circ C}$۳۵ درجه بیشتر شود Cooler روشن شده و اگر از $^{\circ C}$۳۰ درجه کمتر شود خاموش شود. همچنین هرگاه دما بیشتر از $^{\circ C}$۱۵ Heater خاموش شده و در صورت کمتر شدن دما از $^{\circ C}$۱۰ Heater روشن گردد.

OB1

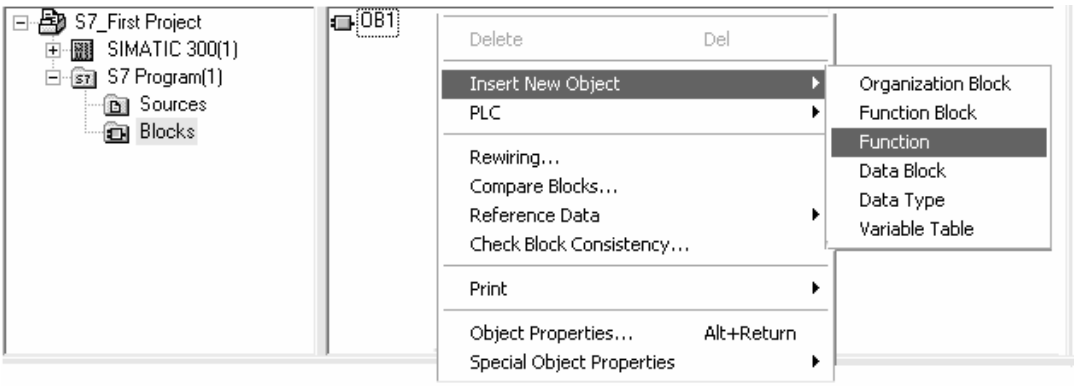


۶-۲ ایجاد و باز نمودن توابع (FC)

در بسیاری موارد برای اجتناب از نوشتن چند باره یک عملیات خاص از توابع استفاده می کنیم. با این روش یک برنامه را می توان به قسمتهای کوچکتر تقسیم نموده که باعث درک سریع تر برنامه می گردد. توابع در سلسله مراتب ساختار برنامه پایین تر از بلوک سازماندهی (OB) قرار میگیرند برای پردازش یک تابع توسط CPU میبایست که آن تابع توسط بلوک های بالاتر(مثلاً OB ها) فراخوانی شود اما بر خلاف بلوک های تابعی نیازی به بلوک داده (DB) ندارند. یک تابع را می توان از توابع دیگر ویا خود تابع (Recursive) نیز فراخوانی کرد.

نکته : در موارد استفاده از توابع recersive ونیز فراخوانی توابع از توابع دیگر باید مراقب زمان بندی باشیم، اگر زمان اجرا بیش از حد باشد Watchdog timer اعلام خطا می کند

برای ایجاد توابع به پوشه بلوک ها رفته و آن را باز نمایید. در بخش راست پنجره کلیک راست نمایید. در منوی pop-up ایجاد شده یک تابع (FC)وارد نمایید .



شکل (۲-۲۳) ایجاد یک تابع

در پنجره محاوره ای "Properties-function" نام FC1 را تایید نموده و زبان برنامه نویسی مناسب را انتخاب نمایید. مابقی تنظیمات پیش فرض را با OK نمودن تایید نمایید. بدین ترتیب تابع FC1 به پوشه بلو ک ها اضافه شده است. با دوبار کلیک کردن FC1 را باز نمایید .

در پنجره برنامه ریزی LAD/STL/FBD گزینه View>LAD را فعال نمایید . توجه کنید که در اینجا FB1 بعنوان Header میباشد زیرا پنجره برنامه نویسی با دو بار کلیک کردن بر روی FB1 باز شده است.

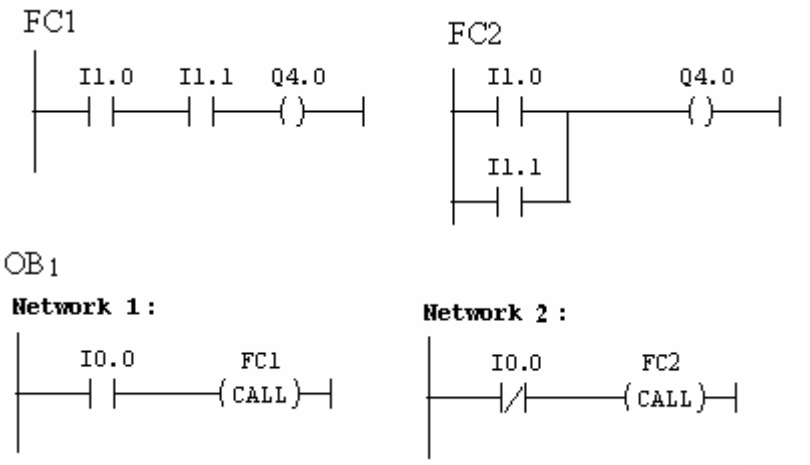
توضیحات زیر را در جدول متغیرها وارد نمایید. برای انجام این کار روی یک خانه (cell) کلیک کرده و نام و توضیح مربوط به آدرس را روی توضیحات زیر وارد نمائید . تنها استفاده از حروف و اعداد Under Score برای پارامترها در جدول توضیح متغیرها (variable Declaration Table) مجاز میباشد.

۲-۶-۱ انواع روش های بکار گیری توابع

• **روش اول :** بعضی توابع داده های مورد نیاز خود رامستقلاً از مسیر دلخواه گرفته وخروجی های مطلوب را به مقصد مورد نظر هدایت می کنند.

مثال۲ .برنامه ای بنویسید که اگر شرط ورودی I0.0 برقرار بود خروجی برابر AND ورودی های I1.1, I1.0 بوده ، واگر شرط مورد نظر برقرار نبود خروجی برابر OR این ورودی ها باشد.

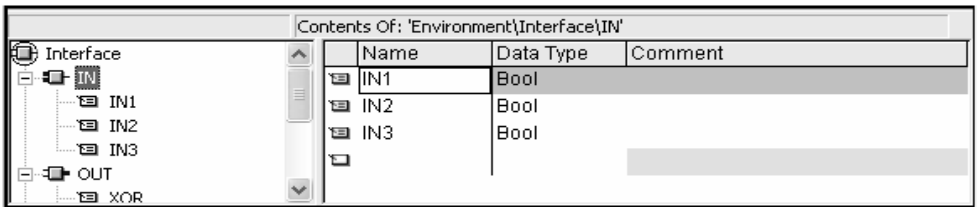
حل:



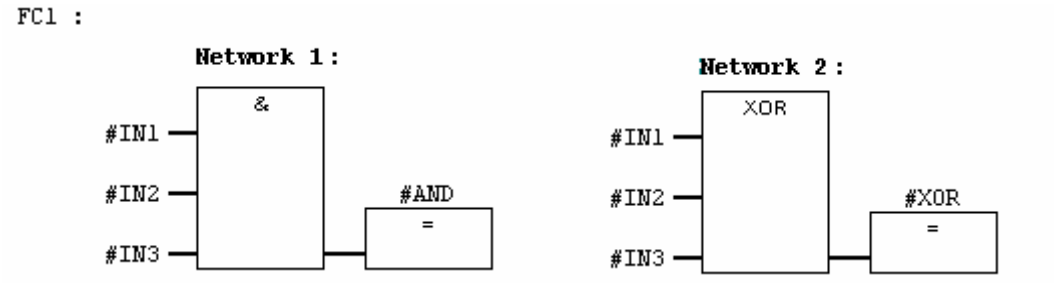
• **روش دوم:** تابع پارامتری می باشد، که درآن تابع را می توان نوشت واز OB ها مقادیر پارامترهای ورودی را ارسال نمود و تابع را فراخوانی نمود و مقدار خروجی را در OB ها بدست آورد.

مثال۳ . تابعی بنویسیم که سه تا بیت را دریافت نموده AND آنرا در خروجی دلخواه قرار داده و XOR آنها را در خروجی دوم قرار داد.

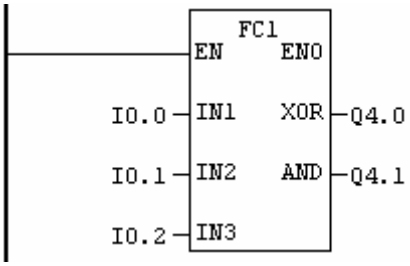
حل : ابتدا تابع FC1 را باز می نماییم در منوی Interface آن، در قسمت IN ، متغیرهای in1 ، in2، in3 و در قسمت OUT ، متغیرهای and ، xor را درج می کنیم.



سپس برنامه مربوط به تابع FC1 را در حالت FBD به صورت شکل بعد می نویسیم.



در نهایت وارد OB1 شده واز پنجره catalog overview گزینه FC BLOCKS ، FC1 را انتخاب کرده تا شکل (۲-۲٤) زیر درج شود. و سپس ورودی ها و خروجی ها را معرفی می کنیم.

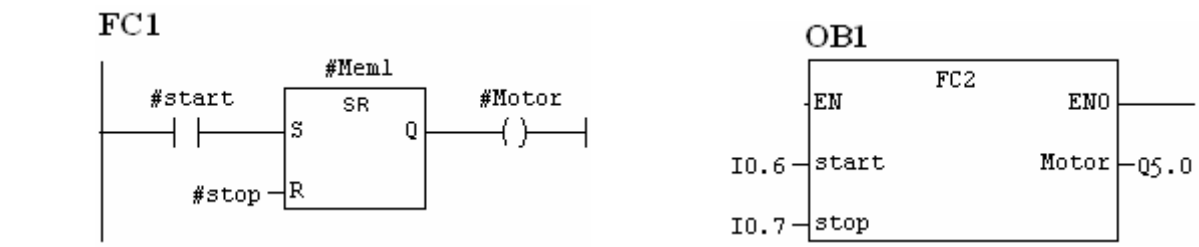


شکل (۲-۲۴)

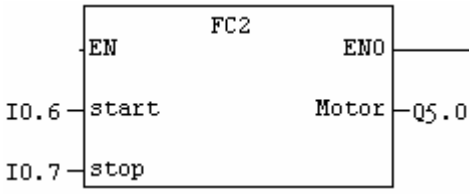
نکته : متغیرهای محلی (Local) با علامت# نشان داده میشود و فقط در بلوک مورد نظر قابل استفاده میباشند. متغیرهای عمومی (global) با علامت " " مشخص میشوند این متغیرها در جدول نمادها (Symbolic Table) تعریف میشوند و در تمام برنامه قابل استفاده می باشند.

مثال٤ . برنامه ای بنویسید که اگر کلید Start زده شود موتور روشن شده واگر کلید Stop زده شود موتور خاموش شود.

حل . یک حافظه SR (چون بعد از فشرده شدن می خواهیم موتور همچنان روشن بماند از حافظه استفاده می کنیم) انتخاب می کنیم و ابتدا برنامه FC1 را می نویسیم، سپس ورودی ها وخروجی های مربوطه را در OB1 اختصاص می دهیم .



حال فرض کنید در جای دیگر از برنامه OB1 ورودی های دیگری مانند زیر به تابع اختصاص بدهیم.



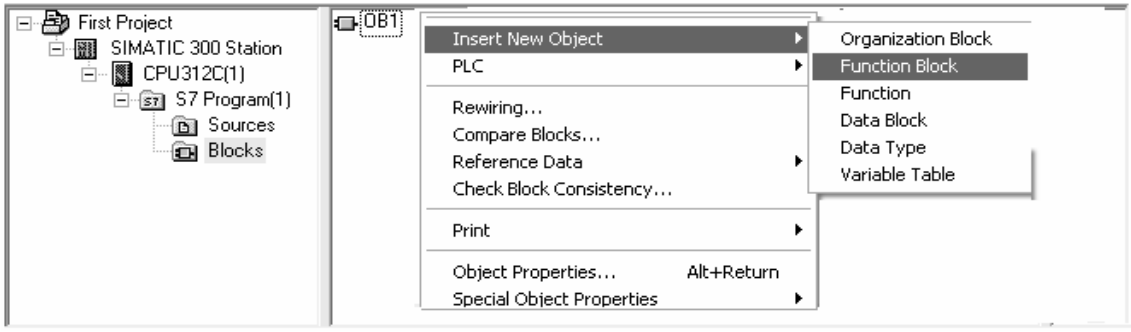
برنامه فوق یک مشکل اساسی دارد و آن این است اگر هریک از ورودی های I0.0 و یا I0.6 ، یک شود خروجی های Q4.0 و Q5.0 با هم یک می شود، در مورد reset شدن نیز وضع برهمین منوال است. چون در توابع FC اجازه استفاده از داده های محلی استاتیک وجود ندارد در چنین مواردی از یک امکان دیگری با عنوان **Function Block** استفاده می شود.

۲-۷ بلوکهای تابعی (Function Blocks) و بلوکهای داده Data Block

۲-۷-۱ ایجاد و باز نمودن توابع بلوکهای تابعی FB

بلوک های تابعی نیز در ساختار برنامه پایین تر از بلوک سازماندهی (OB) قرار میگیرند . این بلوک ها بخشی از برنامه را در خود جای میدهند که میتوانند بارها در OB1 فراخوانی شوند. برنامه ریزی بلوکهای تابعی در پنجره های LAD/STL/FBD همانند برنامه ریزی OB1 و Function می باشد .

ابتدا پروژه "First Project" را باز نموده به پوشه Block رفته و آن را باز نمایید. مطابق شکل (۲-۲۵) در قسمت راست پنجره کلیک راست نمایید. منوی pop-up دکمه راست ماوس فرمانهای menu bar را دارا میباشد یک بلوک تابعی را بعنوان New Objectوارد نمایید .



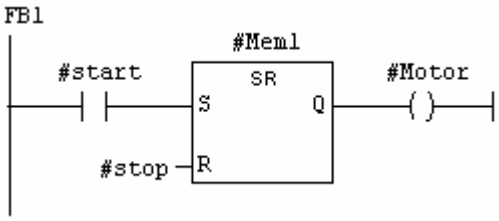
شکل (۲-۲۵) ایجاد توابع بلوکهای تابعی FB

با دو بار کلیک کردن روی FB1 پنجره برنامه نویسی LAD/STL/FBD را باز نمایید. در منوی محاوره ای "properties-Function Block" زبان برنامه نویسی که از آن برای ایجاد بلوک استفاده می کنید را انتخاب و گزینه "Multiple instance FB" را فعال نموده سپس یکی از زبانهای برنامه نویسی را انتخاب و بقیه تنظیمات را نیز با فشردن OK تایید نمایید. حال بلوک تابعی FB1 به پوشه بلوک ها اضافه شده است.

نکته : تفاوت FB ها با توابع در این است که شامل متغیرهای محلی Static می باشد. هرگاه متغیری در FB ها به این صورت تعریف شود هر بار که FB فراخوانی شود یک حافظه مجزا به آن متغیر اختصاص داده می شود، که به این ترتیب از تداخل آن جلوگیری می شود.

مثال ٤ را بااستفاده از FB حل می کنیم.

حل . ابتدا یک FB جدید باز میکنیم و برنامه زیر را در آن درج می کنیم

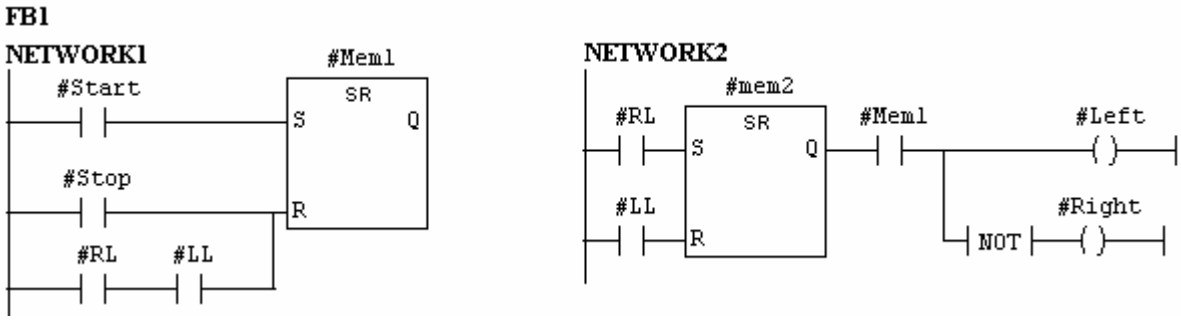


سپس OB1 را باز نموده و از پنجره catalog overview پوشه FB blocks را باز کرده وگزینه FB1 را ۲ بار انتخاب می کنیم. ودر هر بار به هر یک ورودی های مربوطه را اختصاص می دهیم. در بالا ی هر بلوک باید یک DB خاص مشخص نمود که نشانگر حافظه استاتیک مربوطه می باشد.

به این ترتیب با هر بار فراخوانی FB1 یک حافظه مجزا به آن اختصاص می یابد.

مثال۵ . یک موتور چپگرد وراستگرد داریم با زدن کلید Start شروع به حرکت می نماید و بعد از کمی حرکت در جهت چپ به یک کلید محدود کننده برخورد نموده و برمی گردد ودر جهت راست می چرخد بعداز کمی چرخش در جهت راست به محدود کننده راست برخورد نموده وباز برمی گردد ودر جهت چپ می چرخد واین کار را تکرار می نماید تا کلید Stop فشرده شود. برنامه کنترل موتور فوق را بنویسید.

حل. باتوجه به اینکه برنامه فوق نیاز به فلیپ فلاپ دارد بهتر است از FB استفاده نمود، تا فراخوانی برنامه از نقاط مختلف ایجاد تداخل نکند.



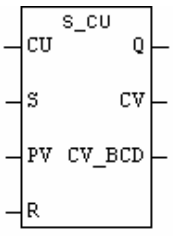
۸-۲ Counters

در بعضی پروسه ها لازم است تعداد دفعات انجام کاری شمارش شود. کانتر ها برای این منظور طراحی شده اند عملکرد آنها بدین گونه است که با ازاءِ اعمال هر پالس ورودی به آن کانتر یکبار می شمارد.

۱-۸-۲ انواع Counter ها

کانتر ها از نظر عملکرد به سه قسمت تقسیم می شوند:

۱. S_CU **کانتر های بالا شمار:** این کانتر به ازای یک پالس لبه مثبت که در ورودی CU ظاهر شود ونیز خروجی کمتر از ۹۹۹ باشد، بصورت BCD یک شماره به بالا می شمارد. در ورودی PV یک مقدار ۲ بایتی قرار می گیرد که در صورت ۱ شدن پایه S این مقدار در خروجی CV بار می شود.



شکل (۲-۲۶) کانتر بالا شمار

سایر ورودی وخروجی ها

پایه CV: خروجی کانتر است و با اعمال پالس به ورودی، این پایه یکی یکی بصورت هگزادسیمال اضافه می شود.

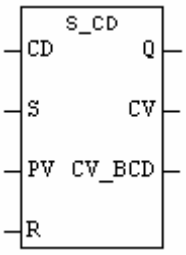
پایه CV_BCD :خروجی کانتر است که بصورت BCD می شمارد.

پایه R : با یک شدن این پایه خروجی **CV**، 0 می شود

پایه Q : به ازاءِ تمام مقادیر بزرگتر از"۱ " CV یک می باشد ، ودر صورت صفر شدن شمارش برابر صفر می شود

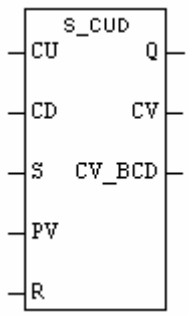
۲. S_CD **کانترهای پایین شمار :** این کانتر به ازای یک پالس لبه مثبت در ورودی CD ، بصورت BCD یک شماره به پایین می شمارد.(در صورتیکه خروجی بیشتر از 0 باشد) با ۱ شدن پایه S مقدار PV در خروجی بار می شود.

پایه Q این کانتر به ازاءِ تمام مقادیر بزرگتر از "۱ " CV ، ۱ می باشد ، و در صورت صفر شدن شمارش برابر صفر می شود. در شکل (۲-۲۷) نمونه ای از این کانترها آمده است.



شکل (۲-۲۷)کانترهای پایین شمار

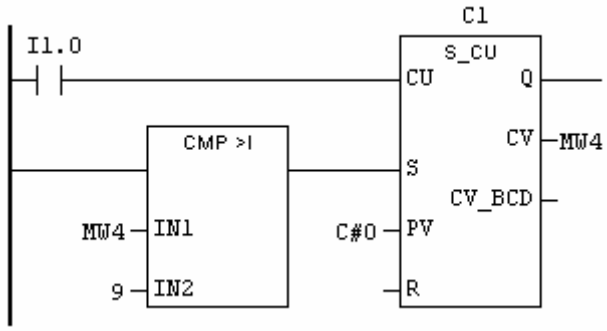
۳. **S_CUD کانترهای بالا_پایین شمار :** با توجه به شکل (۲-۲۸) این کانتر به ازای یک پالس لبه مثبت در ورودی CU رو به بالا و با یک پالس لبه مثبت در ورودی CD رو به پایین می شمارد ،اگر هر دو ورودی CD/CU با هم وارد شود، خروجی یکی بالا ویکی پایین می شمرد.(تغییر نمی کند.) .ما بقی پایه ها بصورت کانترهای بالا می باشد.



شکل (۲-۲۸)کانترهای بالا_پایین شمار

نکته : از استفاده مکرر از یک کانتر ها در نقاط مختلف برنامه بعلت خطا های شمارش پرهیز کنید.

مثال۶ . یک شمارنده طراحی کنید که از ۰ تا ۹ بشمارد و بعد از رسیدن به با هر پالس از صفر شروع کند.



برنامه یکUP_Counter و یک مقایسه کننده است که از 0 می شمارد. هرگاه خروجی کانتر به ۹ رسید خروجی مقایسه کننده ست شده و سبب بار شدنPV در خروجی کانتر می شود

۹-۲ تایمر ها

هرگاه نیاز باشد فاصله زمانی بین دو رویداد را اندازه گرفته و یا عمل خاصی را در مدت زمان مشخص انجام دهیم از تایمرها استفاده می کنیم. ماکزیمم زمان قابل اندازه گیری با تایمرها ۲ ساعت ۴۶ دقیقه و ۳۰ ثانیه بوده و مینیمم زمان قابل اندازه گیری برابر ۱۰ ms می باشد.

عملکرد پایه های مختلف تایمر

پایه S : برای شروع بکار تایمر بکار میرود، که در انواع مختلف تایمرها عملکرد متفاوتی دارد.

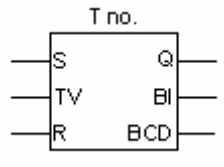
پایه Tv : طول بازه زمانی دلخواه است ودر تایمرهای مختلف عملکرد متفاوتی دارد.

پایه R : در صورت ۱ شدن این پایه در موقع شمردن تایمر reset می شود.

پایه BI : مقدار زمان باقیمانده از زمان tv پس از شروع شمردن که بصورت باینری میباشد.

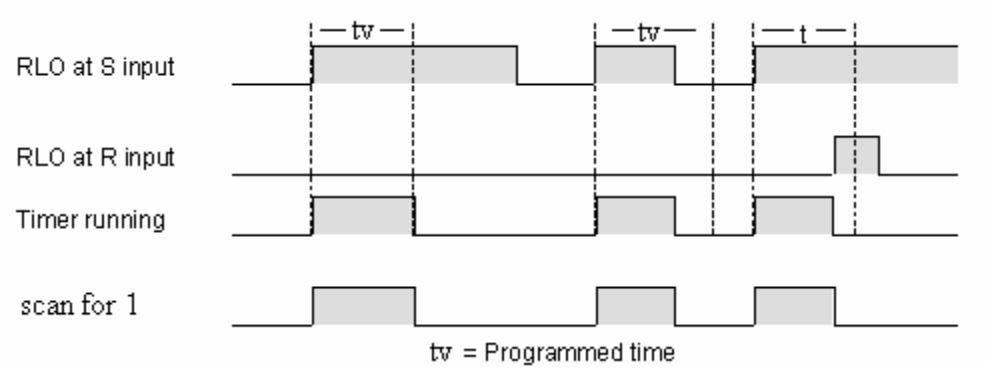
پایه BCD : همان زمان BI می باشد که به شکل دهدهی می باشد.

پایه Q : خروجی تایمر می باشد.



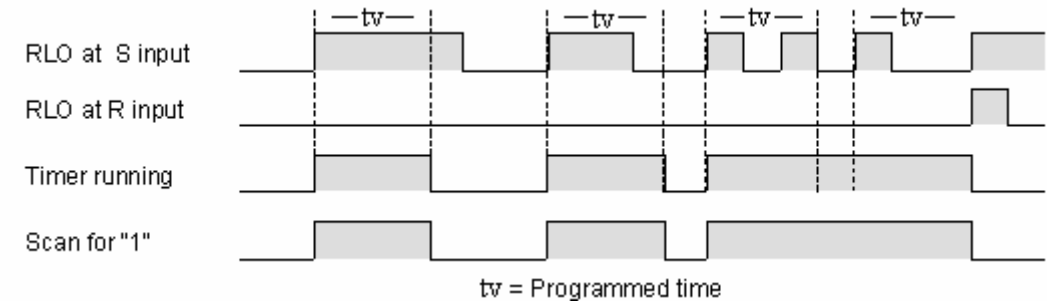
۲-۹-۱ انواع تایمر ها

۱. **Pulse timer (sp) :** مطابق شکل (۲-۲۹) با اعمال لبه مثبت به پایه S تایمر شروع بکار میکند. تایمر به اندازه مدت زمان مشخص شده در پایه TV بعد از اعمال پالس به پایه Sبکار خود ادامه می دهد. اگر قبل از پایان زمان TV سیگنال S از1 به 0 تغییر کند، تایمر متوقف می شود.



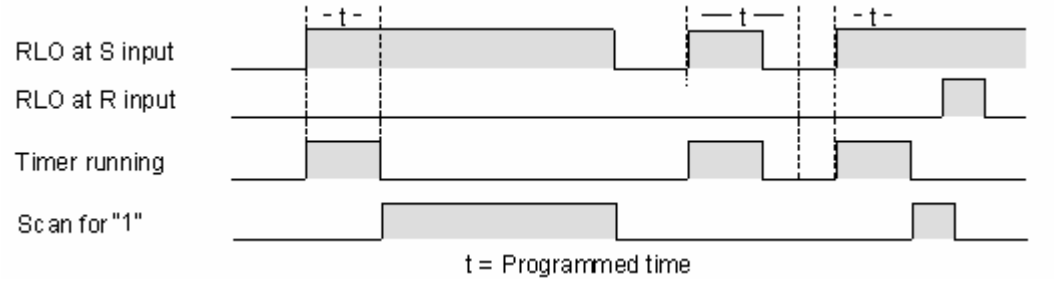
شکل (۲-۲۹) زمان بندی Pulse timer

۲. **SE Extend pulse Timer :** همانطور که در شکل (۲-۳۰) مشاهده می شود، با اعمال لبه مثبت به پایه S تایمر شروع به شمردن میکند. برای ادامه شمردن دیگر نیازی به پالس سطح S نمی باشد، و تایمر بعد از گذشت tv ثانیه متوقف می شود. اگر در زمان روشن بودن تایمر پالس بالا رونده دیگری به پایه S اعمال شود، تایمر از اول شروع به شمردن می کند.



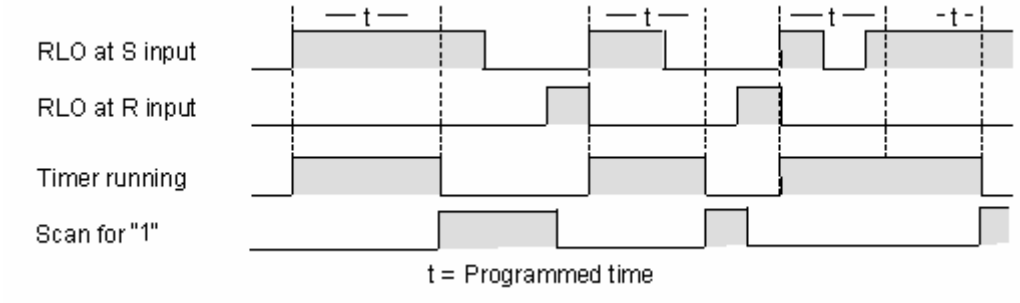
شکل (۲-۳۰) زمان بندی SE Extended pulls Timer

۳. **on-Delay S5 Timer (S_ODT) :** مطابق شکل (۲-۳۱) در این تایمر اگر سیگنال لبه مثبت به پایه S اعمال شود، تایمر شروع به کار میکند وبعد از گذشت tv ثانیه اگر سیگنال پایه S هنوز ۱باشد خروجی ۱ می شود. پس از ۱ شدن خروجی هرگاه پایه S یک شود خروجی 0 می شود.



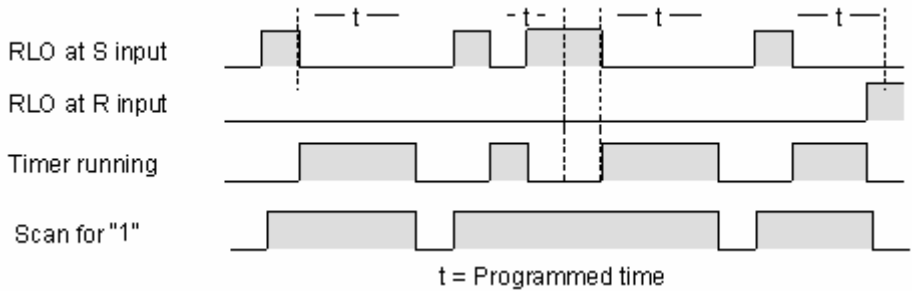
شکل (۲-۳۱) زمان بندی on-Delay S5 Timer

٤. S_ODTS Retentive On-delay : مطابق شکل (۲-۳۲) با اعمال پالش با لبه مثبت به پایه S تایمر شروع به شمردن می کند. برای ادامه شمردن دیگر نیازی به پالس سطح S نمی باشد. خروجی تایمر بعد از گذشت tv ثانیه ۱ می شود. اگر در زمان روشن بودن تایمر پالس بالا رونده دیگری به پایه S اعمال شود، تایمر دوباره شروع به شمردن می کند. وپروسه بالا از نو تکرار می شود.



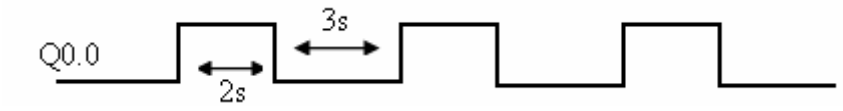
شکل (۲-۳۲) زمان بندی S_ODTS Retentive On-delay

٥. (S_OFFDT) Off-Delay timer : مطابق شکل (۲-۳۳) تا وقتیکه ورودی S ۱ بوده و یا تایمر در حال کار کردن باشد خروجی تایمر ۱ می باشد با اعمال یک پالس لبه منفی به پایه S تایمر روشن می شود. وبعد از گذشت tv ثانیه خروجی ۱ می شود.

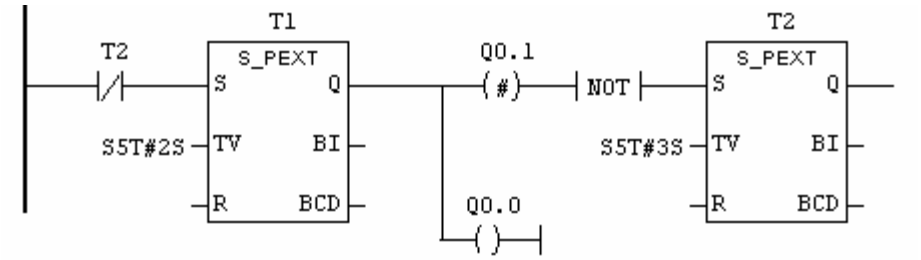


شکل (۲-۳۳) زمان بندی (S_OFFDT) Off-Delay timer

مثال۷ . برنامه ای بنویسید که موج پریودیک زیر را ایجاد کند.



حل: در برنامه شکل (۲-۳٤) فرض کنیم به ورودی S تایمر اول یک لبه مثبت اعمال شود،خروجی Q0.0 ، ۱ می شود. بعد از گذشت 2s خروجی 0 می شود ویک لبه مثبت به تایمر۲ اعمال می شود خروجی این تایمر بعد از 3s یک پالس لبه مثبت به ورودی تایمر ۱ اعمال می کند. و این روند دوباره تکرار می شود.



شکل (۲-۳٤)

المان —(#)— همان خروجی —( )— است با این تفاوت که امکان ایجاد یک انشعاب را می دهد.

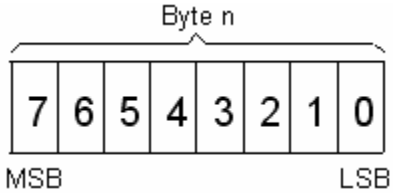
۲-۱۰ انواع داده ها

برای نوشتن یک برنامه از انواع مختلف داده استفاده می شود. هر نوع از داده معرف اندازه و فرمت آن داده می باشد. انواع داده به سه دسته تقسیم بندی می شوند.

۲-۱۰-۱ نوع داده های پایه Elementary Data Types

این نوع داده ها بیشترین کاربرد را در برنامه نویسی دارد. انواع مختلف این نوع داده در زیر آمده است.

- داده های بیتی
  - **BOOL :** که فقط می تواند مقدار 1 بعنوان (True) و 0 بعنوان (False) را داشته باشد.
- متغیر های 1 بایتی :
  - **Byte :** مقادیر بین 00 h تا FF h را قبول می کند.



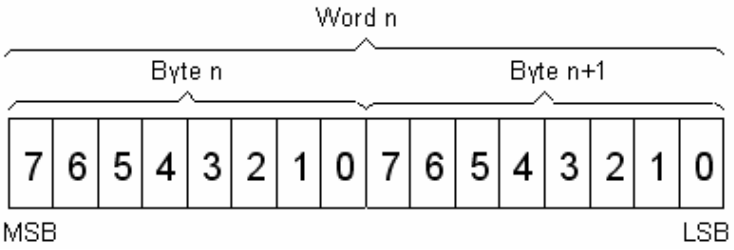
MSB: Most Significant Bit بیت پر ارزش

LSB: Least Significant Bit بیت کم ارزش

➢ متغیرهای 2 بایتی :

- **Word :** مقدار این متغیر بین 0000 H تا FFFF H تغییر می کند. ودر واقع مجموع ۲ بایتی از بیت ها می باشد.

Format	مقدار حداقل	مقدار حداکثر
Binary	2#0	2#1111111111111111
Hexadecimal	W#16#0	W#16#FFFF

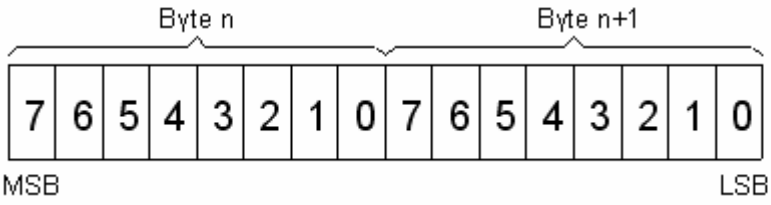


MSB: Most Significant Bit بیت پر ارزش

LSB: Least Significant Bit بیت کم ارزش

- **INT :** این متغیر مقادیر صحیح بین 32768- الی 32767+ را قبول می کند. در این متغیر MSB نشاندهنده بیت علامت می باشد. مقدار 0 بیت علامت نشاندهنده علامت مثبت ومقدار 1 آن نشاندهنده علامت منفی می باشد.

Data Type	Length (bits)	Format	Format Examples	
			Min.	Max.
INT	16	Signed integer	-32768	+32767

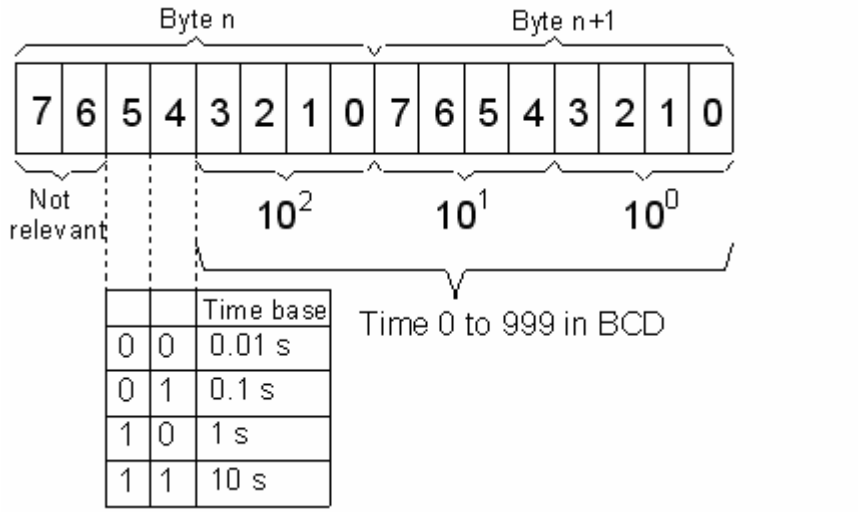


Sign Bit

0 corresponds to the sign "+"

1 corresponds to the sign "-"

- **S5TIME :** در S7 برای نسبت دادن زمان به یک متغیر (در موقع استفاده از تایمر ها) این فرمت بکار می رود.



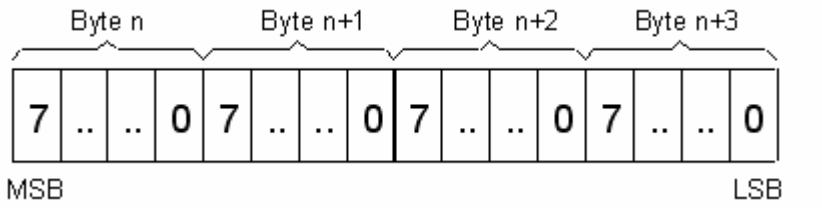
		Time base
0	0	0.01 s
0	1	0.1 s
1	0	1 s
1	1	10 s

- **Date :** در S7 برای اختصاص داددن تاریخ بکار می رود.فرمت آن به شکل زیر است.

Data Type	Length (bits)	Format	Format Examples	
			Min.	Max.
DATE	16	Year-Month-Day	D#1990-01-01	D#2168-12-31

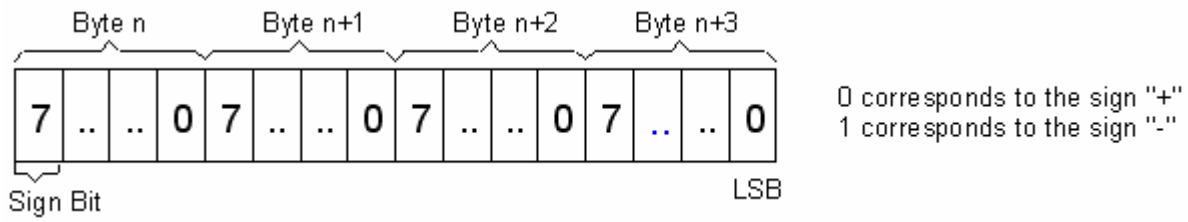
➢ متغیرهای ٤ بایتی

- **DWord :** نوع این داده شبیه Word بوده با این تفاوت که طولش دو برابر است.

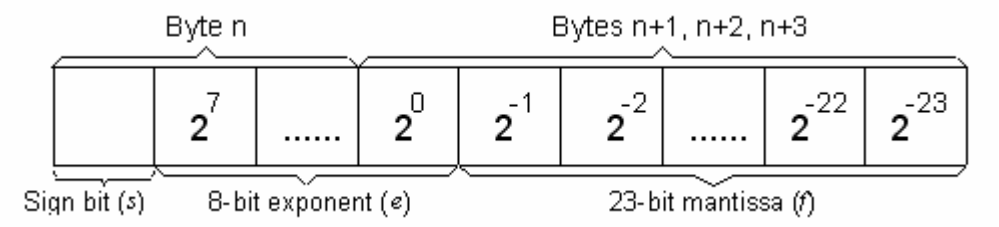


- **DINT :** فرمت این متغیر به شکل زیر می باشد.

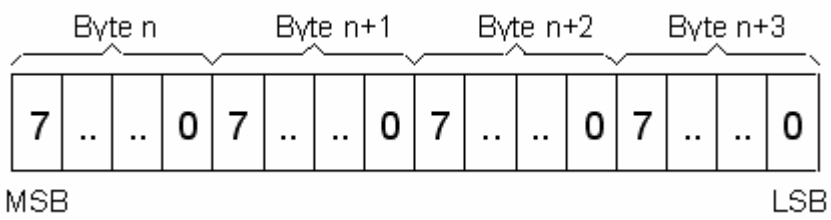
Data Type	Length (bits)	Format	Format Examples	
			Min.	Max.
DINT	32	Signed integer	L# -2147483648	L#+2147483647



- REAL : این فرمت برای متغیرهای اعشاری حقیقی می باشد، ۲۳ بیت کم ارزش این متغیر به عدد اعشاری مورد نظر اختصاص دارد و ۸ بیت بعدی به عنوان توان عدد می باشد. بیت پر ارزش نیز به علامت عدد اختصاص دارد.



- نوع داده TIME برای نشان دادن داده هایی از نوع زمان می توان از این متغیر استفاده نمود. مقدار ماکزیمم ومینیمم وهمچنین فرمت نگارش این داده در زیر آمده است.



T#+24d20h31m23s647ms Max.
T# -24d20h31m23s648ms Min.

که d مخفف day (روز) ، h نشان دهنده ساعت (hour) ،m مخفف minutes (دقیقه) ، s به معنی ثانیه (second) و ms به معنی میلی ثانیه (ms) است.

برای یک داده نوعی لازم نیست تمام واحد های زمان را وارد نمود. به عنوان مثال T#5h10s یک ورودی صحیح است.

۲-۱۰-۲ Complex Data Types

این داده ها شامل انواع زیر می باشد.

- **آرایه ها** : آرایه ها داده هایی پیچیده می باشند که تا ۶ بعد می توانند داشته باشند. تمام داده های یک آرایه باید از یک نوع باشند. محدوده و ابعاد یک آرایه به صورت زیر نمایش داده می شود.

یک بعدی : ARRAY[x1..x2] مثلاً ARRAY[-1..4]

یک آرایه یک بعدی با 6 عنصر ARRAY[0] تا ARRAY[4] می باشد.

سه بعدی : ARRAY[x1..x2 , y1..y2 , z1..z2] مثلاً ARRAY[0..4 ,-2..3,1..7]

محدوده های عناصر یک آرایه می تواند منفی، صفر، مثبت باشند ، اما حتماً با ید x1 از x2 بزرگتر باشد.

- STRUCT ها : که یک مجموعه پیچیده از انواع داده ها می تواند باشد

۳-۱۰-۲ Parameter Type

این نوع داده علاوه بر انواع داده های یاد شده بالا می باشد و فقط برای تعریف پارامتر های IN، IN_OUT، OUT مانند TIMER ها، COUNTER ها، وغیره بکار می رود

پرسشهای فصل دوم

۱. برنامه ای بنویسید که یک موج پریودیک با فرکانس 50 Hz ایجاد کند.
۲. انواع روش های بکار گیری توابع را شرح دهید.
۳. انواع Counter ها را نام برده و عملکرد هر یک را شرح دهید.
٤. انواع تایمر ها را نام برده و عملکرد هر یک را شرح دهید.

فصل سوم

پیکربندی RACK و سخت افزار اصلی PLC های سری S7_ 300

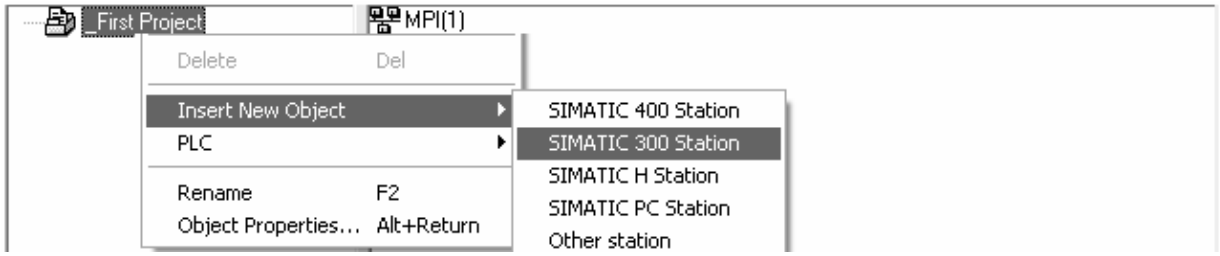
اهداف آموزشی

۱- معرفی کارتهای مختلف CPU به PLC
۲- معرفی کارتهای مختلفremote به PLC

۳-۱ ایجاد سخت افزار

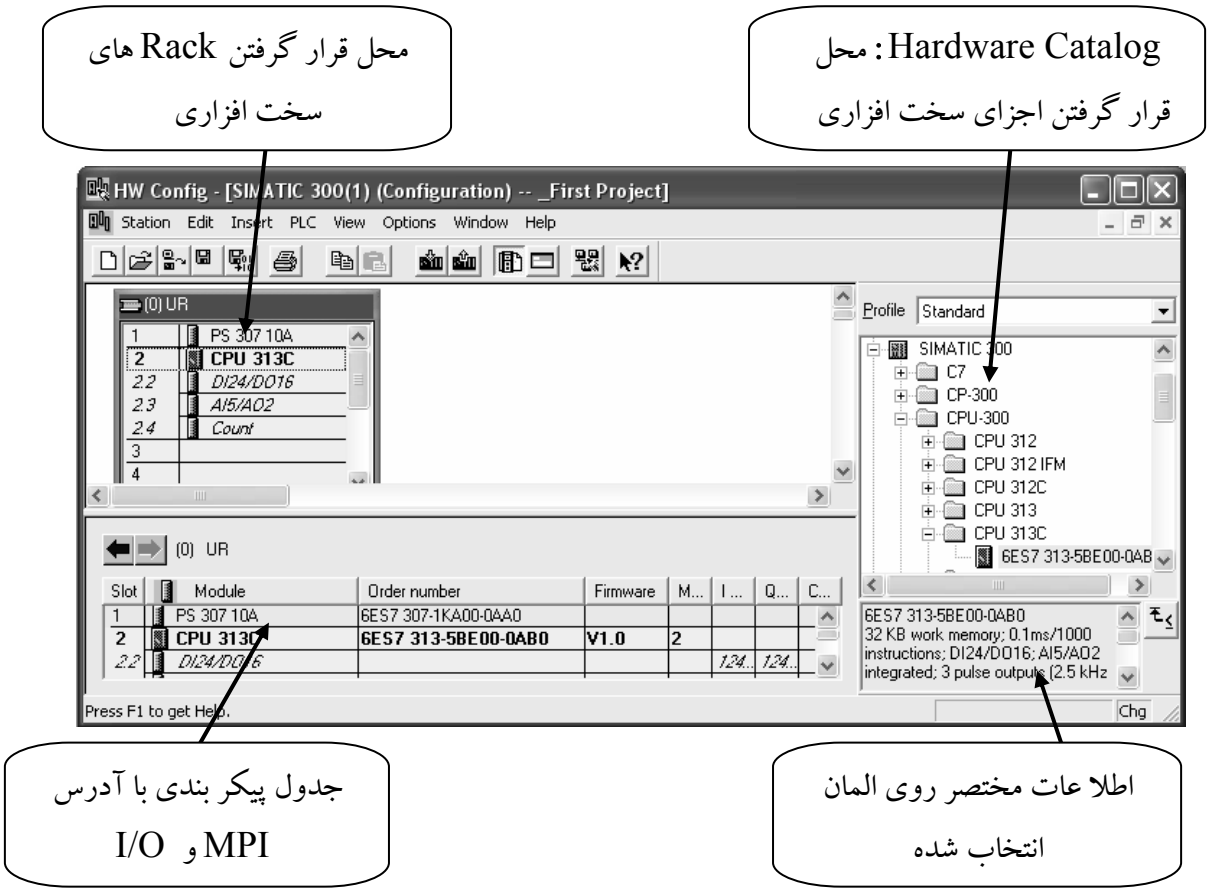
شما می توانید سخت افزار مورد نظر را پس از ایجاد پروژه در قسمت Simatic station تنظیم نمایید. سخت افزار توسط STEP7 تنظیم میگردد ، این تنظیمات در هنگام Downloading به PLC منتقل میگردد .

پوشه Simatic 300 Station را باز نموده و بر روی نماد Hardware دو بار کلیک نمایید .



شکل (۳-۱) ایجاد hardware جدید

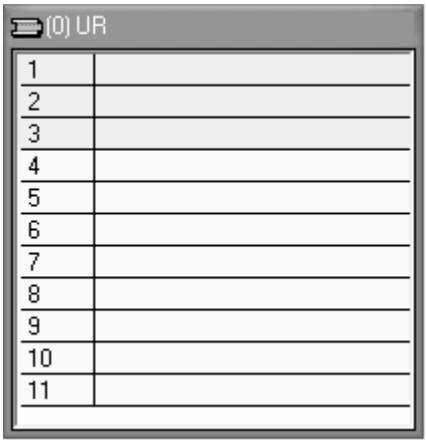
پنجره " HW Config " مانند شکل (۳-۲) باز میگردد و CPU انتخابی شما هنگام ایجاد پروژه نمایش داده میشود . برای پروژه "First Project" CPU از نوع CPU 314 میباشد.



شکل (۳-۲) پنجره hardware configuration

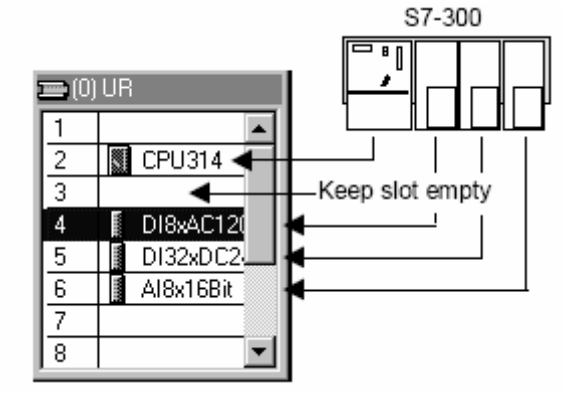
۳-۲ پیکر بندی Slot های مختلف

ابتدا از منوی Catalog شاخه سری Simatic 300 را باز می کنیم. و از گزینه Rack 300 یک Rail انتخاب می کنیم. (dk می کنیم) . مطابق شکل (۳-۳) Rail سخت افزار نمایش داده می شود.



شکل (۳-۳) Rail سخت افزار

مطابق شکل (۳-٤) در هریک از Slot های (خانه های) این Rail باید کارتهای مربوطه را قرار داد. بعبارت دیگر باید هر یک از سخت افزارهای موجود را معرفی در اینجا مشخص و به PLC ، Download کنیم. Slot ها باید بدون جا خالی به ترتیب پر شوند. فقط Slot 3 از این قائده مستثنی می باشد ، که هرگاه از یک Rack استفاده نماییم ، باید خالی گذاریم.



شکل (۳-۴) Rail سخت افزار

۳-۲-۱ Slot 1

مربوط منبع تغذیه می باشد. از شاخه Simatic 300\PS 300 یکی از منابع را انتخاب نموده ، وآنرا به Slot شماره یکDrug & Drop می کنیم. دقت نمایید با انتخاب هر منبع و یا هر المان دیگر اطلاعات مربوط به آن در Box اطلاعات مربوط به المان در زیر صفحه درج می شود.

۳-۲-۲ Slot 2

به CPU اختصاص دارد. شماره PLC موجود را در از شاخه CPU ها پیدا میکنیم وآنرا Drug می کنیم. شماره یک PLC ، در بالای Module آن ونیز بر روی در آن می باشد(در بعضی موارد نیز لازم است در آنرا باز نموده و شماره آنرا مشاهده نمود.) در هنگام انتخاب PLC باید همه این شماره ها با شماره PLC انتخابی همخوانی داشته باشند.

۳-۲-۳ Slot 3

به ماژول Im اختصاص دارد، که در بخش ۳-۳ به آن اشاره خواهد شد.

۳-۲-۴ Slot 4 تا Slot 11

محل قرار گرفتن ماژول های ورودی /خروجی آنالوگ ، دیجیتال، ارتباط پروسسورها و ماژول های تابع می باشد (همچنین می تواند خالی نیز با شد.) ماژول های مربوط به این Slot را می توان در شاخه Simatic 300\ SM 300 پیدا نمود.

۳-۲-۴-۱ ماژول Dummy (DM300)

ماژولی می باشد که میتوان آنرا بجای ماژولی که بعداً نصب می شود قرار داد. به دلخواه می توان یک آدرس را برای این Slot رزرو نمود ویا آدرسی خالی نگذاشت.

۳-۲-۴-۲ ماژول SIM 374

این ماژول میتواند برای شبیه سازی ورودی/خروجی های دیجیتال استفاده شود. این ماژول را نمی توان در پنجره Hardware Catalog پیدا نمود.

۳-۲-۴-۳ ماژول های I/O

ماژول های ورورودی و خروجی دیجیتال وآنالوگ در این Slot ها قرار میگیرند انواع ماژول های ورودی و خروجی با رنج ولتاژ های ± 5 v، ± 10 v، 1_5 v ، ± 20mA، 0_20 mA، 4_20 mAبرای I/O های آنالوگ وdc 24 v ، 15v dc ، 120 v Ac، 230 v Ac با جریان های مختلف برای I/O های دیجیتال را میتوان ازپنجره Catalog پیدا نمود ودر Slot مربوطه قرار داد.

کارت های آنالوگ هم بر مبنای جریان و هم بر مبنای ولتاژ می باشند. بر روی کارت ها یک سلکتور وجود دارد که با آن می توان ولتاژی و یا جریانی بودن کارت را انتخاب کرد.

در سری ۳۰۰ آدرس کارت های آنالوگ از آدرس 256 شروع می شود. برای یک کارت ۱۲ بیتی ۱۶ کاناله به هر کانال ۱۲ بیته ۲ بایت اختصاص داده می شود.

در استفاده از I/O های آنالوگی باید نکات زیر را در نظر گرفت:

الف) نمی توان به محتویات آنالوگ بصورت بیتی دسترسی پیدا کرد.

ب) ورودی خروجی آنالوگ در PII و PIO قرار نمی گیرد، بلکه هر زمان برنامه نیاز داشته باشد به I/O مراجعه می کند.

ج) حداکثر فرکانس نمونه برداری 400 Hz می باشد. اگر نمونه بردار ی با سرعت بالا تر نیاز باشد، باید از کارت I/Oهای fast استفاده کرد.

د) دستورات و توابع PLC که برای متغیرهای دیجیتال بکار برده می شود، را به همان صورت برای متغیرهای آنالوگ نیز می توان بکار برد.

مثال۱. یک ورودی آنالوگ ۱۲ بیتی که با اعمال ولتاژ 0 v عدد 000 H و در ولتاژ 10 v عدد FFF H را نشان می دهد (بین 0 تا 10 ولت تغییر می کند).

الف) 1 ولت معادل چه عددی می باشد؟

ب) عدد ۱ معادل چه ولتاژی است؟

حل: با توجه به اینکه ولتاژ ۱۰ معادل FFF H = 4095 d می باشد داریم.

الف) $\frac{10}{1}=\frac{4095}{x} \quad \Rightarrow \quad x=\frac{4095\times 1}{10}=409$

ب) $\frac{4095}{1}=\frac{10}{x} \quad \Rightarrow \quad x=\frac{10\times 1}{4095}=2.44mv$

۳-۳ RACKهای توسعه یافته

Slot سوم مربوط به ماژول IM(Interface module) می باشد. PLC های S7 دارای قابلیت توسعه پذیری می باشند. با این عمل می توان تعداد ورودی/ خروجی های تحت کنترل را تا چند برابر افزایش داد. S7 به ما امکان می دهد به جز RACK اصلی از RACK های EXPANSION نیز استفاده کنیم. که یا در نزدیکی RACK اصلی بوده و یا در فاصله ای دورتر از آن قرار دارد ماژول های IM وظیفه ارتباط بین Rack اصلی وExpansion را دارد. دو نوع ترکیب برای استفاده از ماژول های بسط یافته در سری 300 وجود دارد.

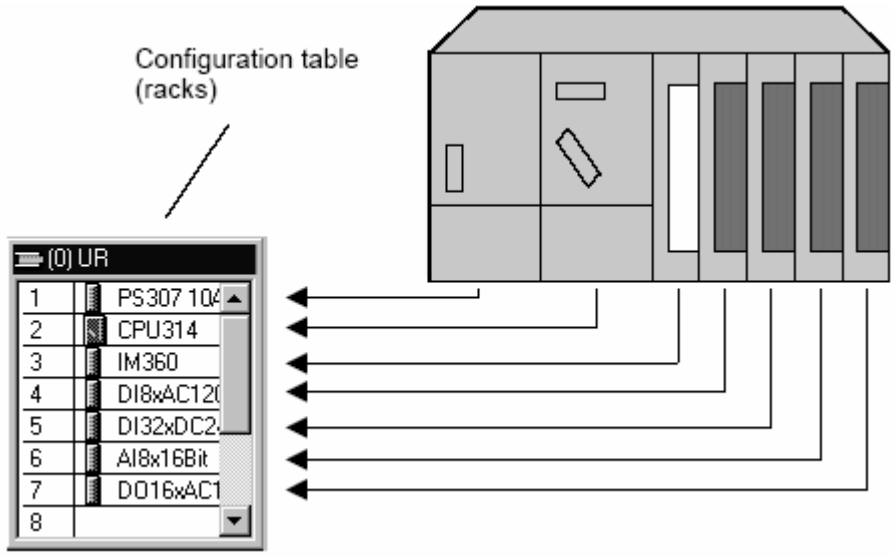
۳-۳-۱ استفاده از ماژول IM 365

با این برد توانایی گسترش یک Rack (Rack 1) توسعه یافته را داریم که محل نصب یک ماژول آن در Rack اصلی بوده و ماژول دیگر آن در Expansion , Rack 1 نصب می شود. در این ترکیب نیازی به Power Supply نمی باشد. حداکثر طول کابل ۱ متر می باشد.

۳-۳-۲ استفاده از ماژولIM 360, IM 361

با این برد توانایی گسترش سه Rack توسعه یافته را داریم. محل نصب IM 360 در Rack اصلی بوده و محل نصب برد IM 361 آن در Expansion , Rack 1,2,3 می باشد. در این ترکیب حداکثر فاصله ۱۰ متر بوده و به Supply Power نیاز دارد.

نکته : برای سری ٤۰۰ می توان حداکثر یک Rack اصلی (main) و Expansion 21 داشت.



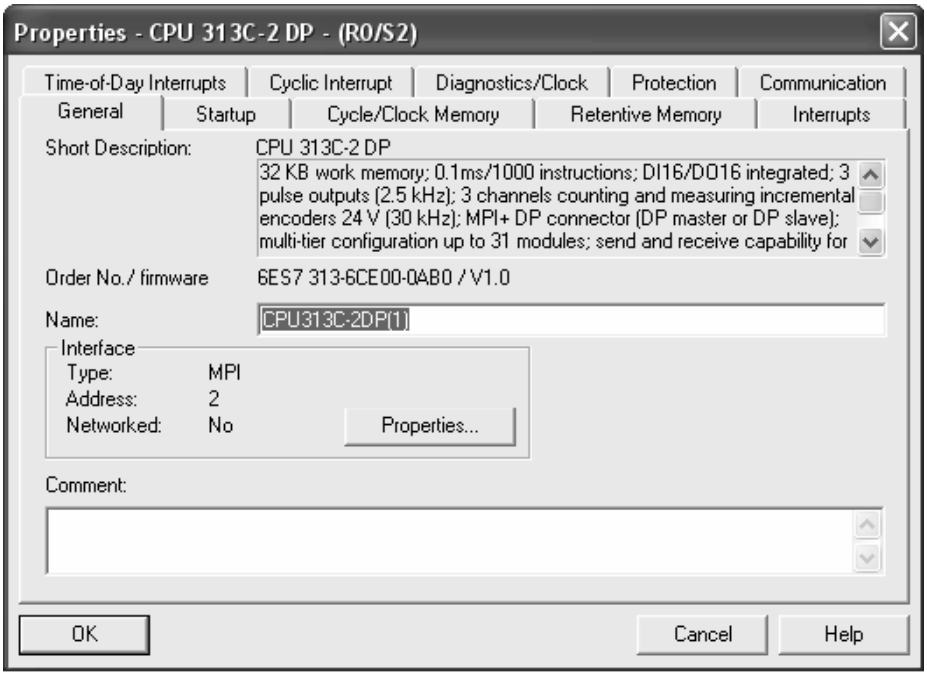
شکل (۳-۵) نحوه نصب و پیکربندی ماژول های IM 360, IM 361

۳-۴ تنظیم پارامترهای سخت افزاری

برای تغییر پارامترهای ماژول (مثلاً آدرسها)، در یک پروژه بر روی ماژول دو بار کلیک نمایید.

۳-۴-۱ تنظیم پارامتر های CPU :

بر روی ماژول CPU در Rack مربوطه db k می نماییم.صفحه شکل(۳-۶) باز می شود.



شکل (۳-۶) پنجره تنظیم پارامتر های CPU

بر روی پنجره شکل چندین برگه وجود دارد، عملکرد هر یک به اختصار شرح داده می شود.

General:

در این صفحه می توان مشخصات CPU ,آدرس و نوع Interface آن را مشاهده نمود.

Startup :

تعاریف

Startup : تغییر وضعیت سیستم از Stop به Start را Startup گویند.

حافظه پایدارretentive : حافظه ای است که با قطع برق بدلیل وجود باتری Back up از بین نمی رود. حافظه ناپایدار حافظه ایست که با قطع جریان برق از بین می رود.

در صفحه Start up میتوان مد Start up را به یکی از صورتهای hot restart, warm restart, cold restart, اختیار کرد. تفاوت این مدها در زیر آمده است.

- **Hot restart :** بعد از Hot restart برنامه از جایی که متوقف شده است Run می شود و هیچکدام از حافظه های پایدار ونا پایدار از بین نمی رود.
- **Warm restart :** برنامه از ابتدا شروع به اجراء می کند و حافظه نا پایدار پاک می شود.
- **Cold restart :** برنامه از ابتدا شروع می شود و حافظه پایدار و ناپایدار پاک می شوند.

Cycle /Clock Memory:

گزینه Scan Cycle Monitoring Time مشخص کننده زمان Watchdog می باشد.در پنجره مربوطه باید یک زمان بر حسب میلی ثانیه وارد نمود اگر زمان Scan time (زمان لازم برای آنکه PLC کل برنامه را یکبار انجام دهد.) بیشتر از زمان Scan Cycle Monitoring باشد، CPU به حالت Stop می رود. دلایلی که ممکن است. زمان Scan time زیاد شود عبارتند از:

۱. پروسه های ارتباطی
۲. رخ دادن یک سری از وقفه ها پشت سر هم
۳. ایجاد یک Error در برنامه CPU

Minimum Scan Cycle Time :

این گزینه در PLCهای سری 300 فعال نمی باشد، اگر Scan time کمتر از حداقل زمان مشخص شده باشد CPU آنقدر منتظر می ماند تا Minimum Scan Cycle Time برسد.

Scan Cycle Load from Communication:

با این پارامتر شما می توانید طول زمان پروسه های ارتباطی را کنترل کنید. بعنوان مثال میتوان از پروسه های انتقال داده به CPU دیگر و یاLoad کردن بلوک ها بوسیله پروگرامر بعنوان نمونه هایی از پروسه های ارتباطی نام برد.

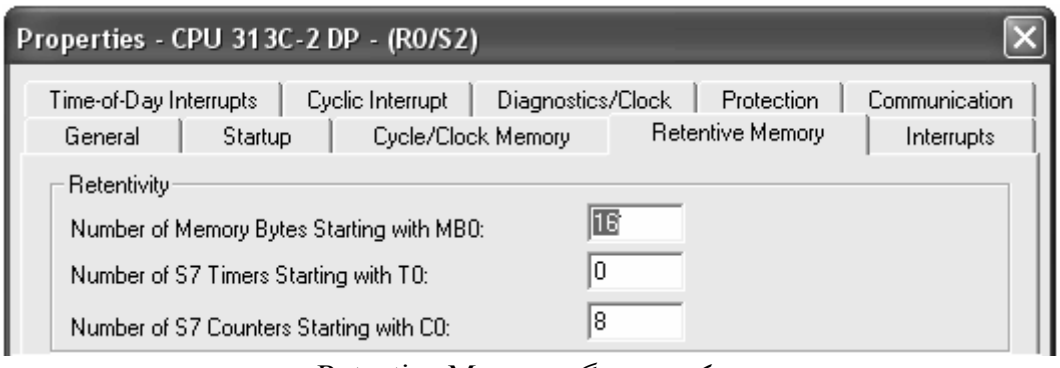
Clock Memory:

اگر گزینه Clock Memory انتخاب شود، CPU این امکان را به ما می دهد که یک بایت دلخواه داشته باشیم که با ۸ فرکانس مختلف تولید پالس نماید.نسبت اندازه طول این پالس ها به طول پریود (duty cycle) 50% می باشد. آدرس بایتی را که می خواهیم پالس های مذکور در آن ظاهر شود با ید در پنجره Memory Byte درج نمود. نرخ فرکانس های پالس های یاد شده در جدول زیر آمده است.

Bit	7	6	5	4	3	2	1	0
Period duration (s):	2	1.6	1	0.8	0.5	0.4	0.2	0.1
Frequency (Hz):	0.5	0.625	1	1.25	2	2.5	5	10

Retentive Memory:

در این صفحه شما می توانید مشخص کنید محتویات کدام قسمت از حافظه بعد از ایجاد خطای قطع در تغذیه باقی بماند. البته این امر در صورت نبود باطری Back up می باشد، در صورت وجود باطری Back up بلوک ها داده های همیشه پایدار خواهد ماند.



شکل(۳-۷) برگ Retentive Memory

درخانه اول باید تعداد حافظه های پایدار با شروع از بایت 0 را وارد کرد.

در خانه دوم و سوم تعداد تایمرها و کانترهای پایدار را انتخاب می کنیم.

در بعضی از CPU این امکان را داریم که DB های دلخواه را پایدار کنیم.

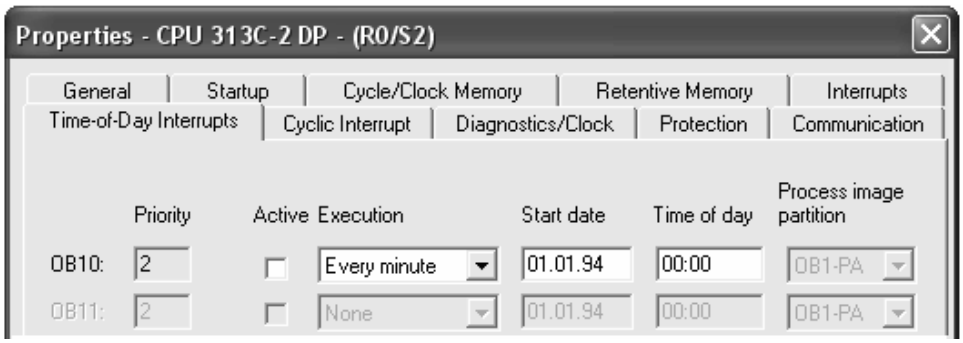
Interrupts :

هر اینتراپت سخت افزاری به یک OB از OB های 40 تا 47 اشاره می کند، ومی توان برنامه سرویس به اینتراپت مورد نظر را در OB مربوطه نوشت. در این صفحه میتوان تقدم Interrupt ها ی سخت افزاری را تعیین نمود.

نکته : باید توجه داشت که برای پذیرش اینتراپت ها باید کارت ورودی اینتراپت پذیر داشته باشیم.

Time-of-Day Interrupts :

اگر بخواهیم برنامه ای بطور پریودیک هر روز ، دقیقه ، ساعت، ماه یا سال و.... انجام شود ابتدا دوره دلخواه را در Execution انتخاب نموده و سپس برنامه مذکور را در OB10 می نویسیم. گزینه های Start date و Time of day شکل (۳-۸) به ترتیب روز شروع وساعت وقوع اینتراپت مربوطه را مشخص می کنند. از OB10 تا OB17 مربوط به این اینتراپت ها می باشد. می توان در این صفحه تقدم این اینتراپت ها را مشخص نمود، در CPU های سری ۳۰۰ فقط OB10 قابل تعریف می باشد.



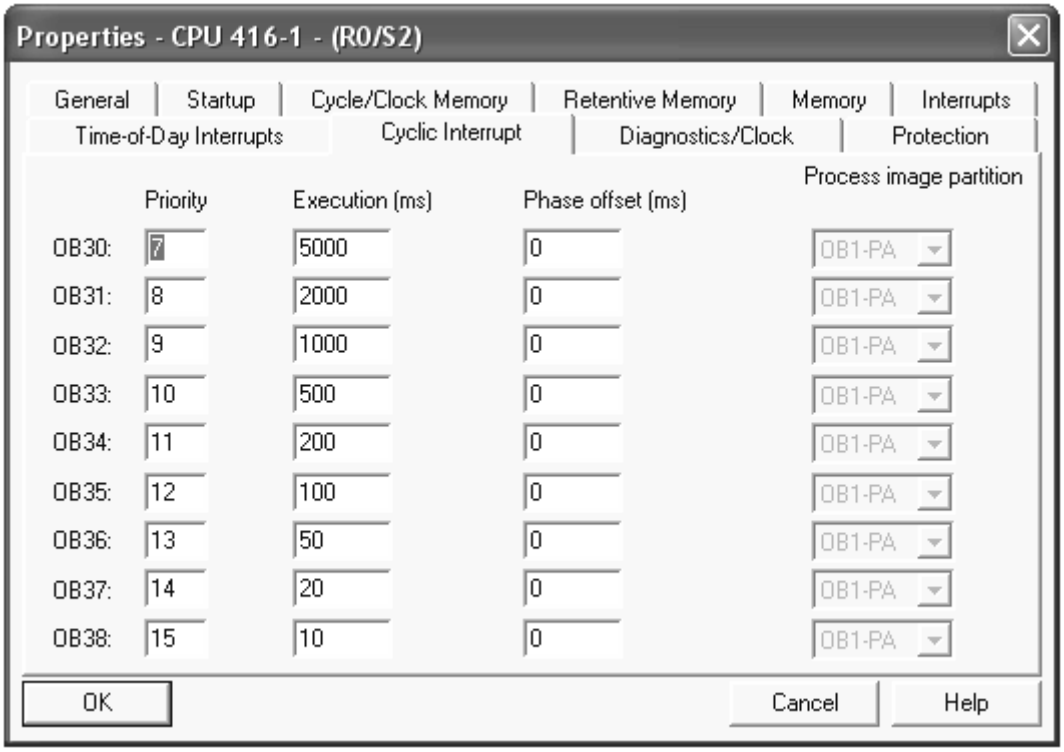
شکل (۳-۸) Time-of-Day Interrupts

Cyclic Interrupts :

همان اینتراپت پریودیک Time-of-Day Interrupts می باشد ، با این تفاوت که فقط زمانهای دوره آن کمتر از دقیقه است. از OB30 تا OB38 به این اینتراپت ها اختصاص دارد.

مطابق شکل (۳-۹) هر کدام از بلوک های OB30 تا OB38 بطور پریودیک هر چند میلی ثانیه اجرا می شوند. این زمان برای ۸ بلوک متفاوت می باشد. پریود مربوط به هر OB در شکل (۳-۹) در مقابلش درج شده است. مطابق شکل (۳-۹) در مقابل هر OB میتوان تقدم آنرا

مشخص نمود که رنج تقدم ها از 0,2,3…24 می باشد. تقدم ۱ به OB1 اختصاص دارد که کمترین تقدم می باشد. در سری ۳۰۰ فقط می توان از OB35 استفاده نمود.

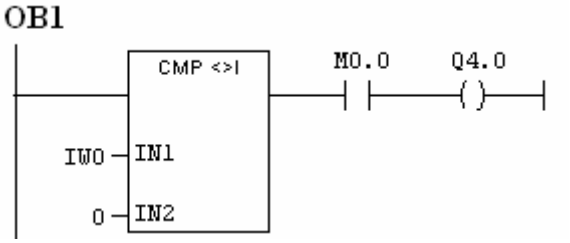


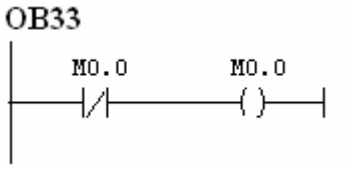
شکل (۳-۹) پنجره تنظیمات Cyclic Interrupts

مثال۲. فرض کنید ۱۶ ورودی داریم برنامه ای بنویسید که اگر هریک از این ورودی ها برابر ۱ شد چراغ error خروجی با پریود T=1s چشمک بزنند.

حل. ورودی ها را به بایت 0 و 1 وصل می کنیم و چراغ error را به خروجی Q4.0 وصل می کنیم.

چون پریود 1s است هر نیم پریود 500ms می شود بنابراین باید از OB33 استفاده نمود. برنامه بصورت زیر است.





Diagnostics/Clock

با انتخاب گزینه Report Cause Of Stop هرگاه سیستم به Stop رفت یک گزارش از علت رفتن به Stop می دهد.

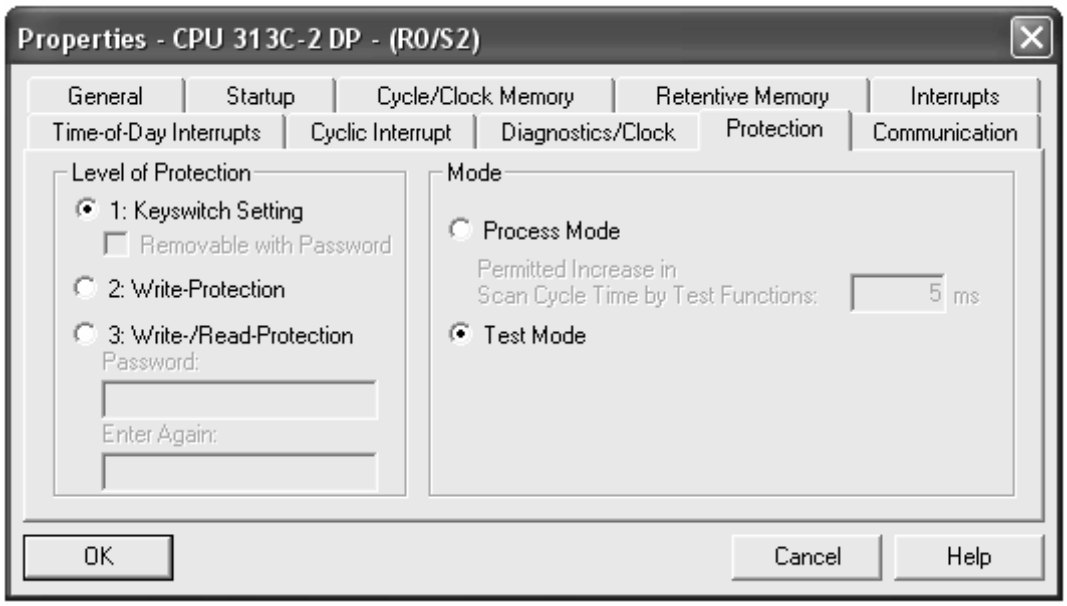
Protection

این صفحه برای حفاظت از برنامه ریزی مجدد و قفل کردن برنامه جاری و یا انتخاب مد اجرای برنامه بکار می رود. .برای حفاظت سه سطح وجود دارد :

۱. **Keyswitch Setting :** در صورت انتخاب این گزینه PLC در صورتی قابل برنامه ریزی می باشد که کلید برنامه ریزی PLC بر روی آن باشد.

۲. **Write_Protection :** با انتخاب این گزینه ابتدا جعبه دریافت Password فعال شده وامکان تعریف یک Password داده می شود. برنامه PLC در صورتی قابل تغییر می باشد، که Password مربوطه داده شود. برای خواندن برنامه نیازی به Password نمی باشد.

۳. **Write/Read_Protection :** با انتخاب این گزینه بدون دادن Password نه می توان برنامه را تغییر داد و نه می توان برنامه را از PLC خواند.



شکل (۳-۱۰) Write/Read_Protection

مطابق شکل (۳-۱۰) برای اجرای برنامه نیز دو مد داریم:

۱. **Test Mode :** دراین مد می توان برنامه را خط به خط اجرا نمود. اجرای خط به خط برنامه به ما امکان اشکال زدایی برنامه را می دهد.

۲. **Process Mode :** در این مد برنامه خط به خط اجرا نمی شود بلکه می توان Cycle Time را تغییر داد. بعد از اجرای هر سیکل CPU منتظر می ماند تا این زمان طی شود.

بعد از انجام تنظیمات یاد شده با استفاده از فرمان Save and Compile داده ها برای انتقال به CPU آماده میشوند.

هنگامیکه پنجره "HW Config" را بستید نماد System DATA در پوشه بلوک ها ظاهر میگردد .

نکته : شما می توانید صحت تنظیمات خود را با استفاده از فرمان Station\Consistency check کنترل نمایید STEP7 راه حل های ممکن برای رفع خطاهایی که احتمال دارد رخ دهد را در اختیارتان قرار میدهد .

پرسشهای فصل سوم

۱. نحوه استفاده از ماژول IM 360, IM 361 را ذکر کنید.
۲. عملکرد بایت Clock Memory را شرح دهید.
۳. در برگه Interrupt پنجره CPU property چه پارامتر هایی را می توان تنظیم نمود.

فصل چهارم
ارسال و عیب یابی برنامه

اهداف آموزشی

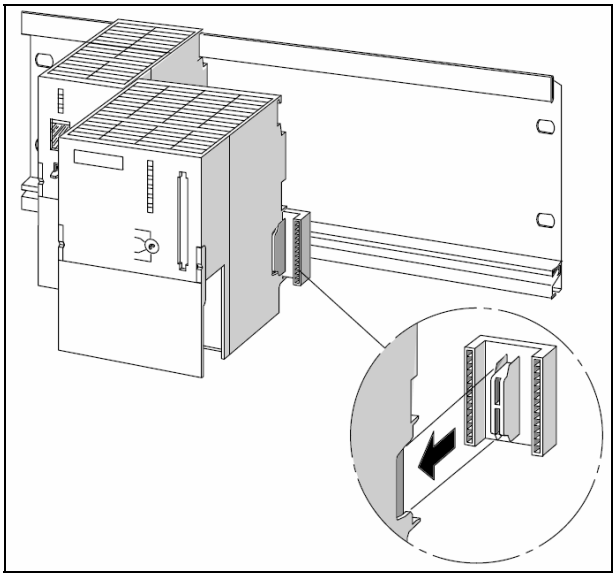
۱- ارسال برنامه به PLC

۲- عیب یابی برنامه

۴-۱ نصب ماژول های PLC

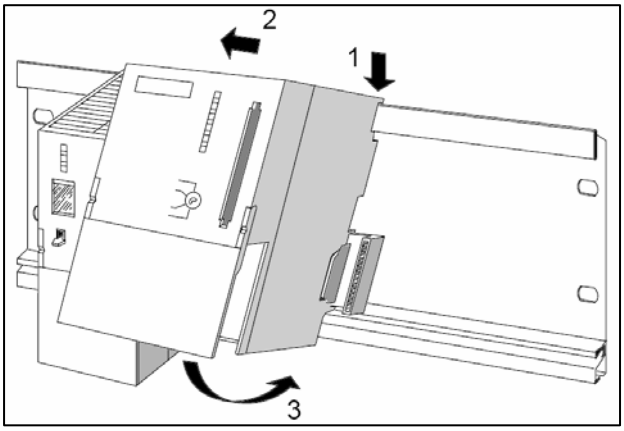
برای قرار دادن ماژول بر روی ریل مراحل زیر را انجام دهید .

با توجه به شکل(٤-١) ماژول را به اتصال دهنده BUS (Connector) وصل نماید .



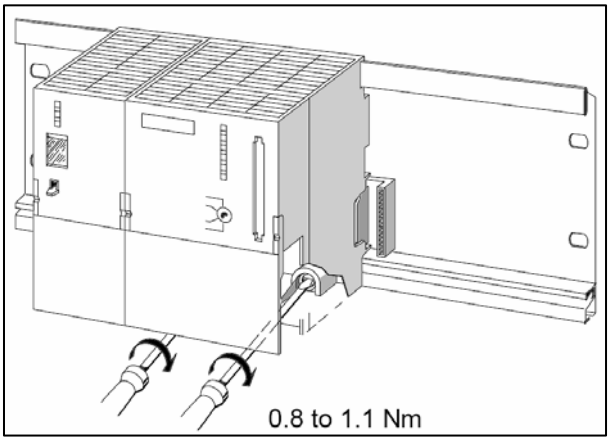
شکل (۴-۱) اتصال ماژول به BUS (Connector)

مطابق شکل (٤_٢) ماژول را بر روی ریل آویزان کرده و به سمت پایین بکشید .



شکل (۴-۲) اتصال ماژول به ریل

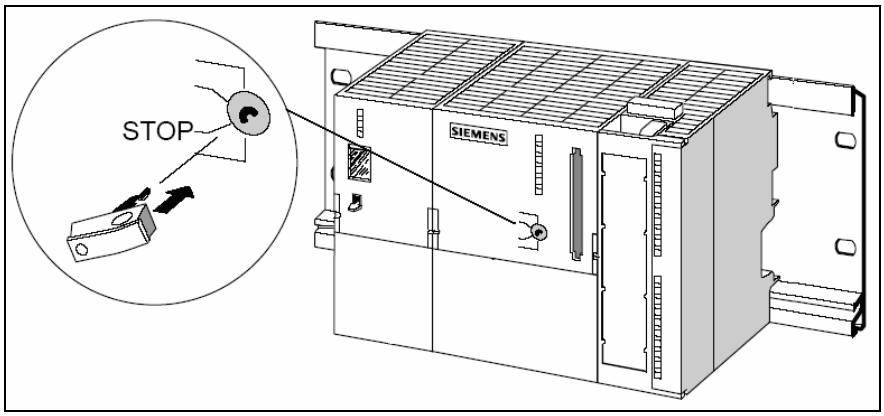
ماژول را در جای خود پیچ نمایید . شکل (٤-٣)



شکل (۴-۳)

ماژولهای باقی مانده را نیز وصل نمایید .

پس از نصب تمامی ماژولها کلید مربوطه را در CPU وارد نمایید . شکل (٤-٤)



شکل (۴-۴)

۴-۲ ارسال برنامه به PLC

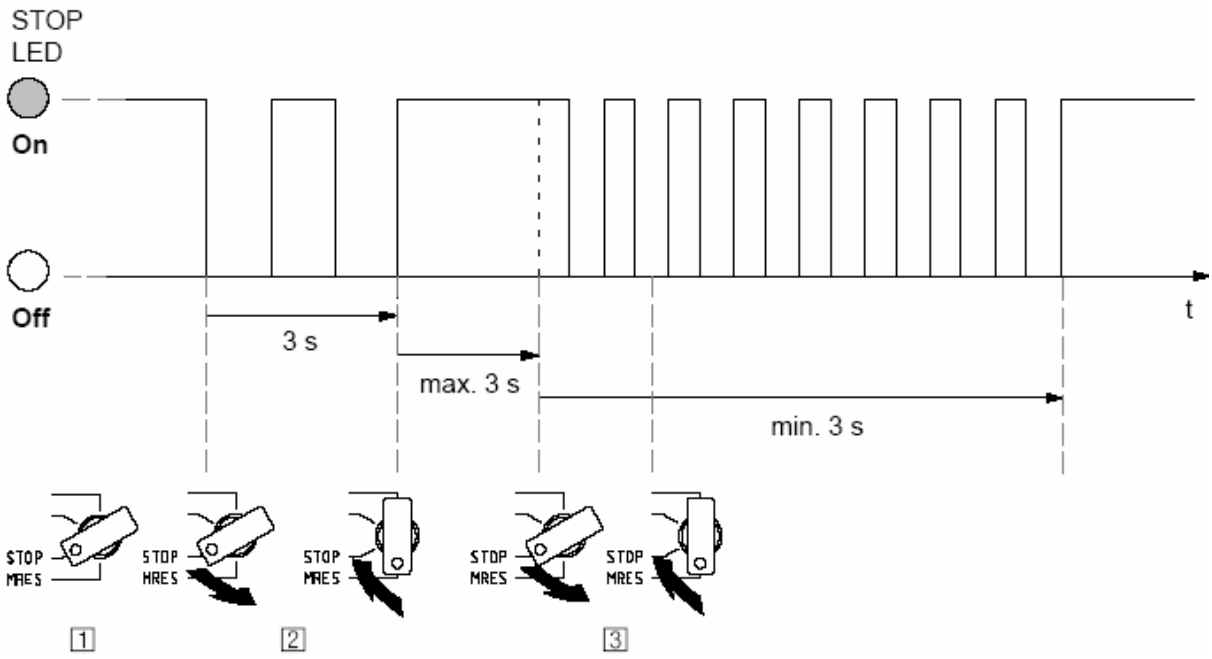
پس از نوشتن برنامه کنترل پروسه نوبت به ارسال آن به PLC می شود. برای این منظور مراحل زیر را طی نمایید.

ابتدا می بایست ارتباط online را برای ارسال کردن برنامه بر قرار نمود. منبع تغذیه را بوسیله کلید ON/OFF روشن نمود . در این حالت دیود "DC 5v" بر روی CPU روشن میگردد . سوئیچ وضعیت عملکرد را به حالت Stop بچرخانید. LED قرمز رنگ "Stop" روشن میگردد .

۴-۳ RESET کردن CPU و بازگشت به حالت RUN

با توجه به شکل(٤-٥) برای ری ست کردن حافظه در PLC مراحل زیر را انجام دهید.

۱. سوئیچ وضعیت عملکرد را به حالت MRES برده و حداقل ۳ ثانیه نگه دارید آنگاه ، LED قرمز رنگ Stop شروع به چشمک زدن بطور آهسته می نمایید
۲. سوئیچ را رها کرده و دوباره به وضعیت MRES برگردانید. هنگامیکه دیود Stop سریعاً شروع به چشمک زدن نمود CPU ، Reset شده است.
۳. اگر دیود Stop شروع به چشمک زدن سریع نکرد پروسه بالا را تکرار نمایید .



شکل(۴-۵) زمان بندی PLC برای RESET کردن CPU

توجه نماییدعمل Reset کردن حافظه تمامی اطلاعات موجود بر روی CPU را پاک می نماید و CPU را به حالت اولیه می برد .

۴-۴ ارسال برنامه به درون CPU

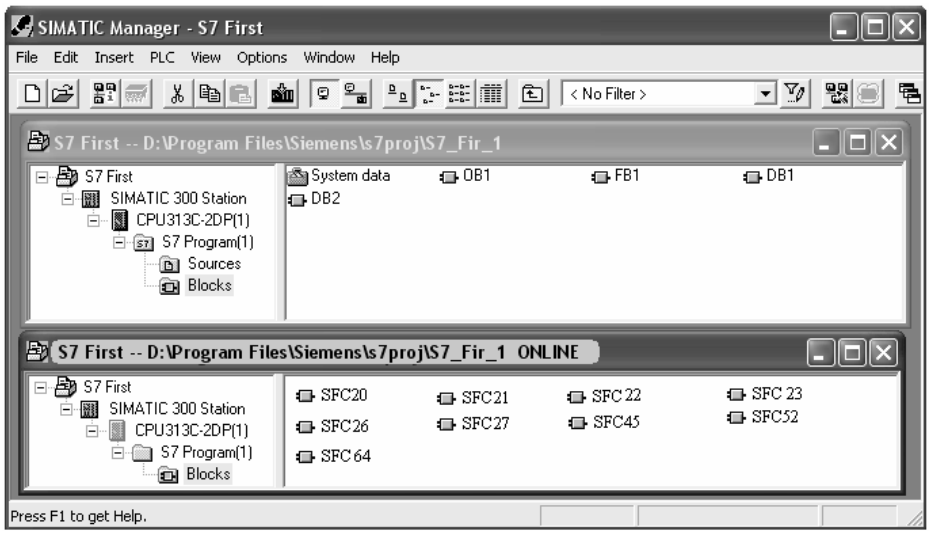
حال سوئیچ وضعیت عملکرد را دوباره به حالت Stop برده تا PLC برای ارسال برنامه آماده گردد .

۴-۵ برقراری ارتباط online

با استفاده از پروژه " Getting Stared" که شما ایجاد کرده اید و یا پروژه "GS-LAD- Example" که موجود میباشد و تنظیم یک تست ساده ، طریقه ارسال برنامه PLC و عیب یابی آن را به شما نشان خواهیم داد .

برنامه Simatic Manager را باز نموده و برنامه مورد نظر را باز نمایید. علاوه بر پنجره "offline" پنجره "online" را نیز باز نمایید حالت online و یا " offline " با رنگ متفاوت در Header پنجره نشان داده میشود .

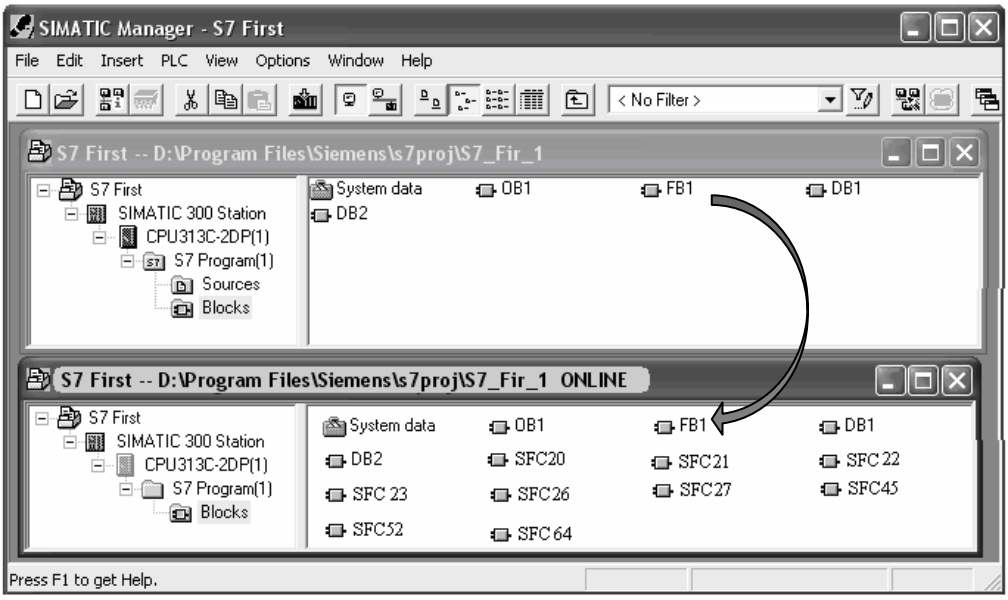
در هر دو پنجره به پوشه Blocks بروید پنجره Offline وضعیت را در دستگاه برنامه ریز و پنجره Online وضعیت CPU را نشان می دهد . شکل (٤-۶)



شکل (۴-۶)

نکته : توابع (SFCs) حتی در صورت reset کردن حافظه ، در CPU باقی می مانند . CPU این توابع مورد نیاز سیستم عامل را تامین می نماید و لزومی به ارسال کردن آنها نبوده و پاک نیز نمیشوند .

پوشه Blocks را در پنجره Offline انتخاب نموده و با استفاده از فرمان PLC/Download برنامه را به CPU ارسال نمایید . عمل ارسال را با فشردن کلید OK تایید نمایید .



شکل (۴-۷)

بلوک های برنامه پس از ارسال شدن در پنجره Online نیز نمایش داده میشوند .

سوئیچ وضعیت عملکرد را به حالت Run-p برده ، دیود سبز RUN روشن شده و دیود Stop خاموش می گردد . CPU برای عملیات آماده می باشد. اگر چراغ قرمز روشن بماند . معنی است که خطا اتفاق افتاده است و برای تشخیص خطا میبایست به Diagnostic buffer مراجعه نمایید .

۴-۵-۱ ارسال بلوک های مجزا

برای اینکه خطاها بسادگی قابل تشخیص باشد میتوان بلوک را بصورت مجزا با استفاده از عمل Drag and Drop به CPU ارسال کرد .

هنگامیکه بلوک ها را ارسال مینمایید کلید وضعیت عملکرد باید روی Stop و یا RUN-P باشد در حالت RUN-P بلوک های ارسال شده سریعاً فعال میشوند .

اگر بلوکهای دارای خطا به جای بلوک های سالم ارسال گردند باعث ایجاد توقف در عملکرد سیستم میشوند برای جلوگیری از این حالت میبایست بلوکها قبل از ارساس تست گردند .

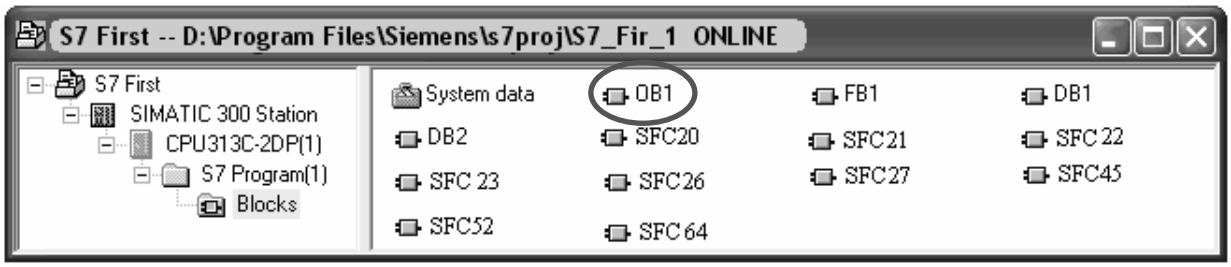
اگر ترتیب ارسال بلوک ها را رعایت ننمایید (ابتدا بلوک های زیری و سپس بلوک های مرتبه بالاتر) CPU به حالت Stop میرود برای جلوگیری از این حالت تمام برنامه را با هم ارسال نمایید.

در عمل گاهی نیاز دارید که بلوک ارسال شده را تغییر دهید برای این کار بر روی بلوک مورد نظر در پنجره Online کلیک نموده پنجره برنامه ریزی LAD\STL\FBD باز میگردد . سپس تغییر لازم را اعمال نمایید دقت کنید که بلوک بلافاصله پس از تغییر فعال میگردد .

۴-۶ عیب یابی برنامه

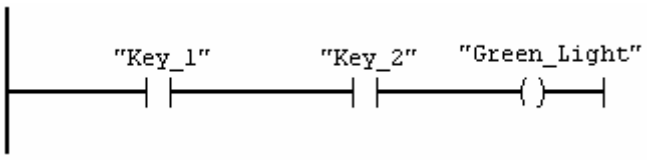
با استفاده از تابعProgram Status می توان بلوک های برنامه را تست نمود برای این کار می بایست یک ارتباط online با CPU برقرار نمود. CPU در حالت RUN و یا RUN-P باشد و برنامه نیز ارسال شده باشد .

OB1 را در پنجره " ONLINE" باز نمایید ، پنجره برنامه ریزی LAD\STL\FBD باز میگردد. تابع Debug/Monitor را فعال نمایید.



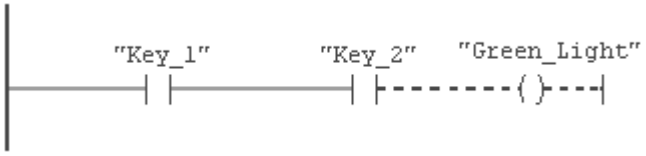
۴-۶-۱ عیب یابی برنامه در LAD

بعد از ارسال برنامه به PLC می توان اجرای مرحله به مرحله آن را مشاهده نمود. برای نمونه یک مدار سری را در نظر بگیرید.



همانطور که اشاره شد برای مشاهده نتایج مرحله به مرحله اجرای دستورات گزینه Debug/Monitor را انتخاب می کنیم. در این صورت مطابق شکل (٤-٧) بعضی از قسمت های مسیر LADER بصورت ممتد سبز رنگ شده و سایر قسمت ها به صورت خط چین می شوند.

خطوط ممتد نشانه آن است که مدار به منبع متصل می باشد و هر جا خطوط مقطع باشد به منزله آن است که سیگنال به آن قسمت نرسیده است .



شکل (۴-۷)

فرض کنید که ورودیKey_1 ، ۱ بوده و ورودی Key_1 ، 0 باشد.در شکل فوق وقتی سیگنال به Key_1 می رسد چون کلید Key_1 بسته می باشد از آن عبور می کند. به همین ترتیب سیگنال به Key_2 می رسد و چون کلید Key_2 کنتاکت باز می باشد بعد از Key_2 مدار به صورت خط چین می شود. در بعلت نرسیدن سیگنال به خروجی Green_light نیز فعال نمی گردد.

۴-۶-۲ عیب یابی برنامه در FBD

در برنامه هایی که به صورت FBD می باشند نیز می توان اجرای مرحله به مرحله برنامه را مشاهده نمود. در این فرمت سیگنال ها توسط ۰و۱ نمایش داده میشوند و خطوط نقطه چین نمایانگر آن است که مدار منطقی مقابل کار نمی کند. (RLO صفر می باشد.)

۴-۶-۳ عیب یابی برنامه در STL

اگر در حالت STL باشیم بعد از مونیتور کردن جدولی حاوی اطلاعات زیر به نمایش در می آید.

حاصل عملیات منطقی (RLO)

بیت Status (STA) (STANDARD)

Standard status

RLO	STA	Standard

حال هر دو کلید را در مدار تست ببندید کلیدهای IO.1 , IO.2 در ماژول ورودی روشن میگردند دیود Q 4.0 در ماژول خروجی نیز روشن میشود .

در زبانهای برنامه نویسی گرافیکی (FBD,LAD) نتیجه تست را از روی تغییر رنگ Network می توان دنبال نمود، این تغییر رنگ نشان میدهد که وضعیت RLO در آن نقطه چگونگی است.
در زبان برنامه نویسی STL رنگ ستـــون RLO و STA هنـــگام ایجاد RLO تغییـر می کند.
حال اگر مثلاً هر دو کلید IO.1 , IO.2 را در مدار تست ببندیم دیود Q 4.0 در ماژول خروجی نیز روشن میشود . نتیجه مونیتور کردن خروجی به صورت زیر است.

RLO	STA	Standard
1	1	0
1	1	0
1	1	0

نکته : توصیه میشود که برنامه های بزرگ را بطور کامل ارسال ننمایید زیرا خطایابی به علت وسعت برنامه و احتمال وجود خطاهای زیاد سخت میگردد. در عوض با ارسال بلوک های مجزا میتوانید به راحتی برنامه را تست و خطایابی نمایید .

پرسشهای فصل چهارم

۱- مراحل RESET کردن CPU را شرح دهید.

۲- نحوه نصب ماژول ها ی PLC را بیان کنید.

مراجع

۱- خودکاری با PLC تألیف : سید حجت سبز پوشان

۲-عملکرد وکاربرد های PLC در اتوماسیون صنعتی تألیف : یان وارناک