خود آموز PLC S7 300

- ✓ سخت افزار
- ✓ نرم افزار
- ✓ ۴۳ پروژه تکنیکی و کاربردی

تهیه و تنظیم : منوچهر مومنی

فصل اول

فصل دوم

فصل سوم

فصل چهارم

فصل پنجم

فصل ششم

فصل هفتم

فصل هشتم

فصل نهم

فصل دهم

مقدمه:

در گذشته برای انجام کارهای کنترلی از قطعاتی مانند کلیدها و کنتاکتورها استفاده می شد که امنیت پائین و دشواری در ایجاد روندهای کنترلی ،هزینه زیادی را بر صنعت کشورها اعمال می کرد.پید ایش قطعات محاسبه گر الکتریکی مانند ماشین حساب ها ، محققان را به فکر تولید تجهیزاتی انداخت که عملیات محاسباتی را به صورت الکتریکی انجام دهند.

از زمان پیدایش کامپیوترها به عنوان محاسبه گرهای الکترونیکی ، مهندسان به فکر استفاده از این تکنولوژی در صنعت بوده اند. اما به دلیل ضعف های زیاد سیستم های کامپیوتری در گذشته مانند امنیت پائین، استفاده از این علم در صنعت امکان پذیر نبود. پیدایش ترانزیستورها و در ادامه آن میکروکنترلر ها تحولی عظیم در علم کامپیوتر به وجود آورد. این تحولات راه را برای ورودی کنترل کننده های کامپیوتری به صنعت هموار ساخت.

این کنترل کننده ها تحت نام PLC[1](کنترل کننده های منطقی برنامه پذیر) در جهان شهرت یافتند. از مهمترین تأثیرایت این قطعات می توان به موارد زیر اشاره کرد:

- ✓ افزایش سرعت در محاسبات و تولید محصولات
- ✓ افزایش امنیت و کیفیت محصولات
- ✓ کاهش هزینه تولید کننده هم از لحاظ نیروی کار و هم از لحاظ تعمیرات و نگهداری

با توجه به اهمیت PLC ها در صنایع مختلف ، شرکت های مختلفی در سرتاسر جهان تولید این محصول را آغاز کردند. یکی از این شرکت ها ، شرکت زیمنس آلمان می باشد که PLC های خود را تحت نام سری S7 وارد بازار نموده است که در این سری ، خانواده 300 و 400 از شهرت بیشتری برخوردار هستند. اما پائین بودن هزینه سری 300 باعث گردیده از این خانواده به طور گسترده تری در صنعت استفاده شود. به همین دلیل در این جزوه نیز PLC های سری

[1] Programmable logic controller

s7-300 به طور کامل بررسی می گردند.ابتدا سخت افزار این تجهیز به همراه اطلاعات مورد نیاز معرفی گردیده و در فصل های بعدی ، مباحث نرم افزاری آن به طور جامع توضیح داده شده است.

فصل اول

آشنایی با سخت افزارهای PLCسری ۳۰۰

در این فصل آشنا خواهید شد با .....

۱. قابلیت های مختلف سخت افزار را شناسایی و از آن بهره بجوئید.

۲. انتخاب سخت افزار مطابق با نیاز و نحوه سفارش دادن آن.

۳. عیب یابی در صورت بروز مشکل سخت افزاری

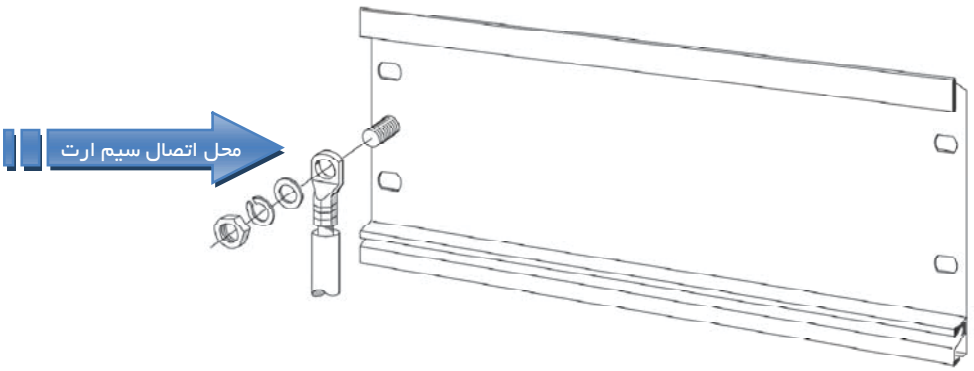
سخت افزار:

برای کار با PLC ها آشنایی به سخت افزار الزامی است زیرا به کاربر کمک می کند تا با قابلیت ها و محدودیت های سیستم خود آشنا باشد. از طرفی شناخت سخت افزار برای کار با نرم افزار اجباری است. در این فصل به معرفی جامع و کاربردی سخت افزار PLC از سری S7300 می پردازیم.

از بین سخت افزارهای موجود ، گروهی وجود دارند که از همه پرکاربردتر هستند و گروهی دیگر برای موارد خاص به کار می روند. در این فصل ابتدا سخت افزار های پر کاربرد به طور کامل توضیح داده خواهند شد و در ادامه آن سخت افزار های خاص معرفی می شوند.

رک (ریل):

این قطعه جهت نگهداری قطعات دیگر به کار می رود. به نوعی سایر سخت افزار ها بر روی این ریل نصب می شوند.در زیر تصویر یک رک خالی نشان داده شده است.



یک رک را می توان هم به صورت افقی و هم به صورت عمودی به بدنه تابلو یا هر جای دیگر پیچ کرد. محل نشان داده شده در شکل جهت اتصال سیم ارت سیستم می باشد و سایر سوراخ ها در ابتدا و انتهای ریل برای عبور پیچ در نظر گرفته شده است.کاربر می تواند با توجه به نیاز خود ، ریل با اندازه های مختلف را سفارش دهد. برای خرید ریل 5 استاندارد تعریف شده است که هر استاندارد اندازه مخصوص به خود دارد.

استاندارد ریل ها:

۱) 160 mm (6es7390 1AB60 0AA0)

۲) 482 mm (6es7390 1AB80 0AA0)

۳) 530 mm (6ES7390-1AF30-0AA0)

۴) 830 mm (6ES7390-1AJ30-0AA0)

۵) 2000 mm (بدون سوراخ، کاربر باید با توجه به نیاز خود برش داده و آنرا سواراخ نماید و استفاده کند)

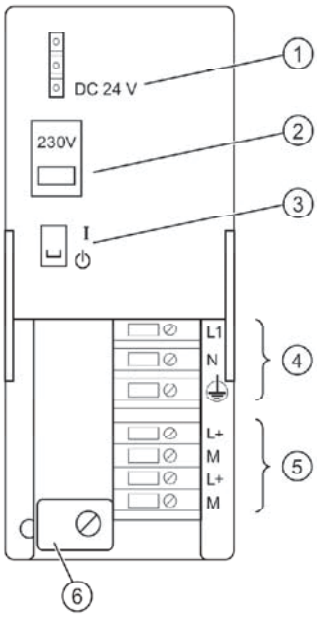
رک به طور فرضی به قسمت هایی با اندازه های مختلف تقسیم بندی می شود که به آن اسلات گفته می شود و از سمت چپ نیز شماره گذاری می گردد. ترتیب نصب سخت افزار ها بر روی رک بسیار مهم است از این رو هر قطعه باید در اسلات مخصوص به خود نصب گردد. نحوه نصب سایر قطعات بر روی رک در ادامه به طور کامل نشان داده خواهد شد.

منبع تغذیه:

جهت راه‌اندازی سیستم های PLC نیاز به منبع تغذیه است تا برق سیستم را تأمین کند. PLC ها با ولتاژ 24VDC تغذیه می شوند. منابع تغذیه با توجه به حداکثر جریان خروجی دسته بندی می شوند که این جریان ها به ترتیب 2,5,10 A می باشند و هر چه میزان جریان دهی منبع تغذیه افزایش یابد اندازه آن نیز افزایش می یابد. در شکل زیر این منابع در کنار هم نشان داده شده اند. منابع تغذیه در رک ها در اسلات شماره یک(اولین اسلات سمت چپ) قرار می گیرند.

معرفی اجزای منابع تغذیه:

یک منبع تغذیه در حالت کلی از اجزای زیر تشکیل می‌شود که در زیر به آنها اشاره شده است. آشنایی کامل با این اجزا، در زمان سیم بندی به کاربر کمک می کند.



۱- نمایشگر وجود ولتاژ 24VDC در خروجی (LED سبز)

۲- سوئیچ انتخاب برق ورودی (با توجه به کشور استفاده کننده ، کاربر آنرا بر روی 230V (مثل ایران) یا 110V (مثل امریکا) قرار می دهد)

۳- کلید روشن و خاموش

۴- ترمینال ورودی برق شهری

۵- ترمینال ولتاژ خروجی 24VDC

۶- نگهدارنده کابل های ورودی یا خروجی

پردازنده اصلی (CPU):

CPU از اصلی ترین قسمت های تشکیل دهنده یک PLC است و دانستن اطلاعات بیشتر و دقیق تر در مورد آن کار با PLC را ساده می سازد.CPU مخفف واژه Central Process Unitبه معنی واحد پردازنده مرکزی بوده که کنترل اصلی سیستم را بر عهده دارد و انجام محاسبات و صدور فرمان نیز بر عهده این ماژول است. قابلیت های یک PLC وابسته به قابلیت هایی است که یک CPU پشتیبانی می کند. پس کاربر باید در مرحله اول نیاز خود و هدف کنترلی خود را به طور کامل بشناسد و سپس به انتخاب CPU بپردازد.

CPU ها از نظر ساختار به دو دسته اصلی تقسیم بندی می شوند:

- ماژولار
- کامپکت

CPU ماژولار: این گروه از CPU ها فقط شامل یک عدد CPU تنها می باشند که عملیات پردازش و کنترل سیستم را بر عهده دارد. در شکل زیر نمایی از این CPU ها نشان داده شده است. CPU ها توسط مدل خود شناخته می شوند که این نام شامل حداقل 3 عدد می باشد که معرف مدل CPU است. از انواع مختلف گروه های CPU ماژولار می توان به موارد زیر اشاره کرد: (دقت شود که هر گروه خود شامل مدل های مختلف با قابلیتهای متفاوت می باشند)

- CPU 312 (6ES7 312-1AE14-0AB0)
- CPU 313 (6ES7 313-1AD03-0AB0)
- CPU 314 (6ES7 314-1AG14-0AB0)
- CPU 315(6ES7 315-1AF03-0AB0)
- CPU 316 (6ES7 316-1AG00-0AB0)

CPU کامپکت: در این دسته از CPU ها علاوه بر پردازنده اصلی ، تعداد مشخصی ورودی و خروجی نیز به همراه CPU قرار داده شده است که نوع و تعداد ورودی و خروجی ها بسته به مدل ها متفاوت می باشند.که در زیر تصویری از یک CPU کامپکت نشان داده شده است.

نام این گروه از CPU ها نیز مانند گروه ماژولار می باشد با این تفاوت که در انتهای نام آنها حرف C به نشانه کامپکت(Compact) بودن CPU قرار داده شده است برای مثال:

- CPU 312 C (6ES7 312-5BE03-0AB0)
- CPU 313 C (6ES7 313-5BF03-0AB0)

CPU ها با توجه به این که قابلیت اتصال به شبکه ها را دارا می باشند یا خیر نیز تقسیم بندی می شوند. CPU هایی که در بالا نام برده شد این قابلیت را دارا نمی باشند. از شبکه هایی که این CPU ها قابلیت اتصال برای آنها تعریف شده است عبارتند از: شبکه پروفیباس DP ، شبکه PTP یا نقطه به نقطه(point to point) و شبکه profinet (PN)

CPU هایی که قابلیت اتصال به شبکه Profibus DP را دارند در انتهای نام مدل آنها عبارت 2DP قرار داده شده است. مثال:

- CPU 313 C 2DP (6ES7 313-6CF03-0AB0)
- CPU 314 C 2DP (6ES7 314-6CG03-0AB0)

برای شبکه PTP هم داریم:

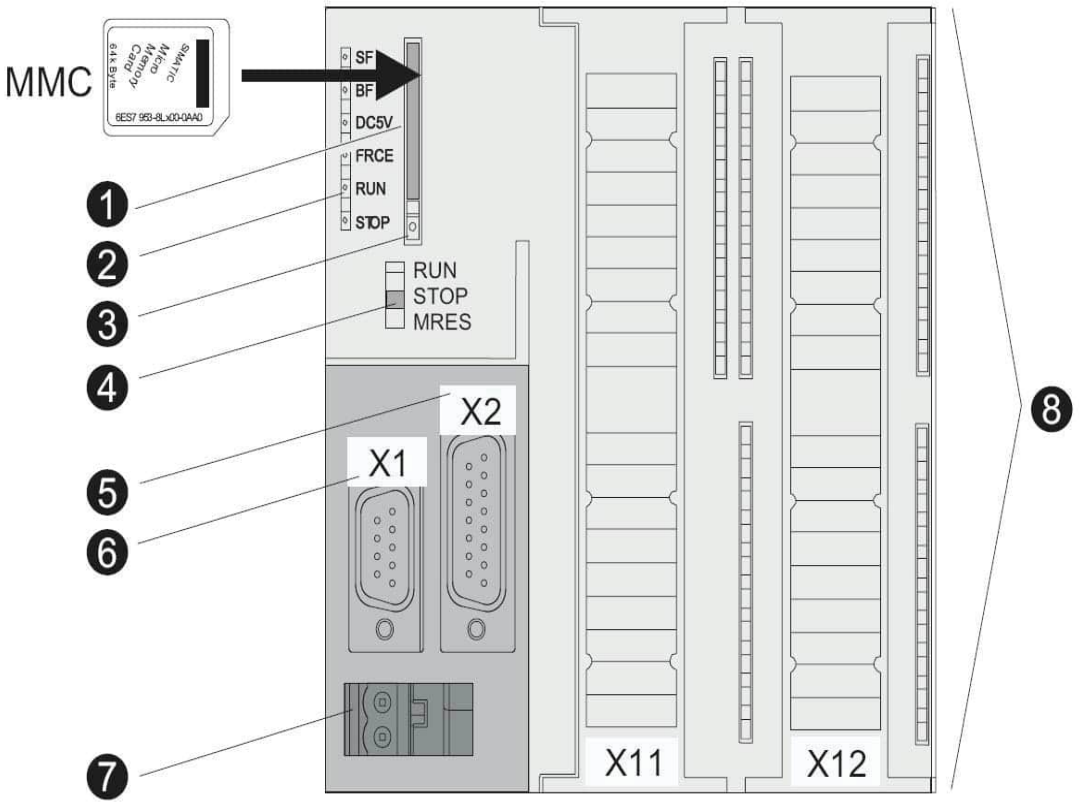
- CPU 314 C PTP (6ES7 314-6BG03-0AB0)

برای شبکه PROFINET نیز داریم:

- CPU 315 2 PN/DP (6ES7 315-2EH13-0AB0)
- CPU 317 2 PN/DP (6ES7 317-2EK13-0AB0)

سخت افزار CPU:

آشنایی با سخت افزار CPU از ضروریات کار با یک PLC است و به همین دلیل در تصویر زیر یک CPU کامپکت به همراه جزئیات سخت افزاری آن نشان داده شده است.



۱) محل قرار دادن کارت حافظه (به طور کامل در مبحث حافظه در مورد آن بحث خواهد شد)

نمایشگر های وضعیت و خطا از نوع LED که در انتهای همین صفحه توضیح داده شده است.

۳) ضامن خروج کارت حافظه

۴) کلید ، جهت تعیین مد کاری CPU که شامل 3 وضعیت کاری می باشد.حالت STOP، RUN و MRES. که به سمت RUN به صورت کلید عمل می نماید و با رها کردن سوئیچ در همان وضعیت باقی می ماند ولی در جهت MRES به صورت شستی عمل می کند و با رها کردن دوباره سوئیچ به حالت STOP بر می گردد که در ادامه توضیحات تکمیلی به هر حالت نشان داده شده است.

۵) محل قرار گیری واسط DP یا PTP

۶) واسط MPI

۷) کانکتور منبع تغذیه CPU

۸) ورودی و خروجی های یکپارچه

LEDهای CPU:

بر روی CPU 6 عدد LED نصب شده است که با توجه به مدل CPU تعداد آن متفاوت خواهد بود در شکل زیر نمای نزدیک L D های CPU دیده می شود.

Stop: روشن بودن LED نارنجی رنگ STOP نشان می دهد که CPU در مد STOP می باشد. در این حالت هیچ سیگنال ورودی را دریافت نمی کند و فرمانی هم در خروجی قرار نمی دهد.

RUN: روشن بودن LED سبز رنگ آن به معنی RUN بودن CPU است یعنی پردازنده در حال کار می باشد و سیگنال خروجی در حال تولید شدن است و عملیات کنترلی در حال اجراست.

FRCE: این حروف از کلمه FORCE بدست آمده است و اگر ورودی یا خروجی به طور نرم افزاری تغییر وضعیت دهند این L D روشن خواهد شد.

DC 5v: هرگاه تغذیه CPU برقرار باشد این L D سبز رنگ روشن می گردد.

BF: (Bus Fault) با بروز خطا در شبکه این L D قرمز رنگ روشن می شود.

SF: (System Fault) در صورت بروز هر گونه خطا ، چه نرم افزاری و چه سخت افزاری این L D قرمز رنگ روشن می شود.

نکته: با روشن شدن L D SF آنگاه CPU به حالت STOP رفته و فرآیند کنترل را متوقف می کند.

در جدول زیر چند نمونه از پر کاربرد ترین CPU ها و قابلیت های آنان نشان داده شده است.

مدل CPU	CPU 312	CPU 313C	CPU 313C 2Dp
ورودی های دیجیتال	۲۵۶	در مجموع ۱۰۱۶	در مجموع ۸۱۹۲
خروجی های دیجیتال	۲۵۶		
ورودی های آنالوگ	۶۴	در مجموع ۲۵۳	در مجموع ۵۱۲
خروجی های آنالوگ	۶۴		

همان طور که در جدول فوق دیده می شود با افزایش مدل CPU حجم ورودی و خروجی هایی که CPUپشتیبانی می کند نیز افزایش می یابد. در CPU312 کاربر فقط می تواند حداکثر 256 ورودی داشته باشد و اگر زمانی کاربر نیاز به 300 ورودی داشته باشد نمی تواند از این مدل CPU استفاده کند ولی در مدل CPU313 کاربر می تواند با توجه به

نیاز خود تعداد ورودی و خروجی دلخواه خود را انتخاب نماید برای مثال می تواند 1000 سیگنال ورودی تعریف نماید و فقط 16 عدد را به عنوان خروجی استفاده کند که مجموعاً به عدد 1016 خواهد رسید. همین روند برای سیگنال های آنالوگ نیز برقرار است.

کارت های ارتباطی:

همان طور که در مبحث CPU دیده شد این ماژول فقط مسئولیت پردازش سیگنال ها را بر عهده دارد و به تنهایی قادر به ارتباط با محیط اطراف خود نمی باشد. به همین جهت برای ارتباط پردازنده و دنیای بیرون نیاز به رابط هایی است که این کار را انجام دهند که این کار توسط کارت های ارتباطی صورت می پذیرد. منظور از ارتباط با دنیای بیرون دریافت یا انتقال اطلاعات بین CPU و دستگاه های خارجی است.این دستگاه های خارجی می توانند سنسور یا محرک ها باشند. کارت های ارتباطی به دو دسته SM ها (Signal Module) ماژول سیگنال و FM ها (Function Module) ماژول های تابعی تقسیم بندی می شوند که هر یک به طور کامل توضیح داده خواهند شد.

سیگنال های ماژول SM: از این ماژول ها برای تبادل سیگنال ها بین CPU و محیط اطراف استفاده می شود که به دو دسته کلی قابل تقسیم هستند.

کارت های ورودی: این دسته از کارت ها ، اطلاعات و سیگنال ها را از محیط اطراف جمع آوری نموده و به CPU تحویل می دهند.

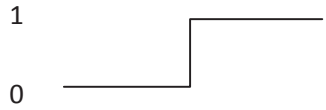
کارت های خروجی: بعد از پردازش سیگنال های دریافتی توسط CPU ، باید سیگنال های فرمان را به محرک ای واقع در دنیای بیرون انتقال داد. برای انجام این کار از کارت های خروجی استفاده می شود.

پس برای راه اندازی یک سیستم کنترلی وجود حداقل یک کارت ورودی و یک کارت خروجی الزامی است.

* البته موارد فوق برای زمانی است که CPU از نوع کامپکت نباشد و یا در صورت کامپکت بودن تعداد ورودی و خروجی برای پروسه مورد نظر کم باشد.

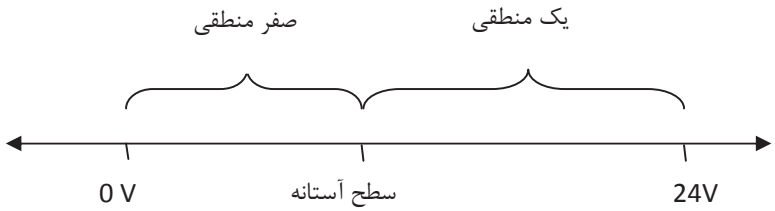
سیگنال ها در دنیای بیرون به دو نوع دیجیتال و آنالوگ دسته بندی می شوند که در اینجا هر یک به اختصار توضیح داده خواهد شد.

سیگنال های دیجیتال: این دسته از سیگنال ها برای انجام عملیات منطقی استفاده می گردند که فقط دارای دو حالت قطع و وصل ستند. به حالت قطع صفر منطقی و به حالت وصل ، یک منطقی گفته می شود.



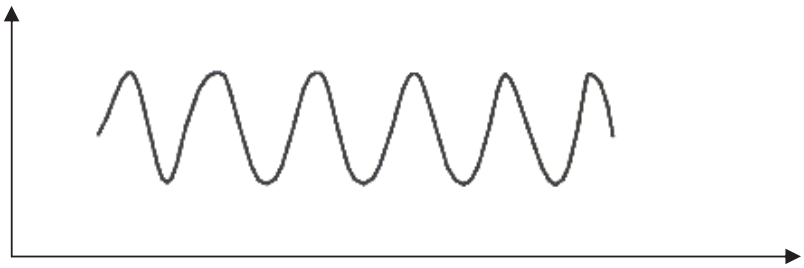
از آنجائیکه ولتاژ کاری PLC از نوع 24VDC می باشد بنابراین مقدار، 24VDC معادل یک منطقی و صفر ولت نیز معادل صفر منطقی می باشد.

نکته: برای سیگنال های دیجیتال یک سطح آستانه تعریف می گردد که اگر ولتاژ بالاتر از این سطح آستانه قرار گیرد به عنوان یک منطقی محسوب شده و در کمتر از سطح آستانه، صفر منطقی تلقی می گردد.



با انجام این تقسیم بندی CPU دیگر دچار سردرگمی نمی شود چون ممکن است بر اثر یک حادثه سیستم دچار افت ولتاژ گردیده و به جای آنکه ولتاژ 24V بر روی ورودی PLC قرار گیرد ولتاژ 16v حاضر شود در آن صورت تشخیص CPU در زمینه صفر یا یک بودن منطقی با مشکل روبرو می‌شد ولی با استفاده از سطح آستانه این مشکل حل می شود.

سیگنال های آنالوگ: سیگنال های آنالوگ بر خلاف سیگنال های دیجیتال می تواند مقدار پیوسته به خود بگیرد و در هر لحظه از زمان مقدار خاصی را دارا باشد.اکثر سیگنال های محیط اطراف از نوع آنالوگ می باشند مانند: دما ، فشار ، سرعت ، فاصله و ... به شکل زیر توجه کنید.



سیگنال های آنالوگ در صنعت به 4 گروه عمده تقسیم می شود:

۱- ولتاژی
۲- جریانی
۳- مقاومتی
۴- ترموکوپلی

از آنجائیکه سیگنال های آنالوگ می توانند مقادیر مختلفی داشته باشند به همین جهت استانداردهایی برای این سیگنال ها در نظر گرفته شده است که برای سیگنال های جریانی می توان به 0...20mA و 4...20mA و برای سیگنال های ولتاژی نیز به+5V ... 5V,-5V ... +10V, 0 ... 10V, -10V ... 0 اشاره نمود.

از بین سیگنال های معرفی شده فوق می‌توان سیگنال های 10V... 0 ، 10V+ ... 10- و 20 mA... 4 را به عنوان پرکاربردترین سیگنال ها در صنعت نام برد و اکثر سنسورهای آنالوگ نیز خروجی هایی در این بازه تولید می کنند.

با توجه به نوع کارت و نوع ورودی در مجموع 4 دسته کارت می توان در نظر گرفت که عبارتند از ورودی دیجیتال ، خروجی دیجیتال ، ورودی آنالوگ و خروجی آنالوگ. اسامی قطعات این گروه با SM آغاز می شود که نشان دهنده Signal Module می باشد و عدد جلوی آن نشان دهنده نوع کارت و نوع سیگنال کاری آن می باشد که در ادامه هر یک را به طور کامل معرفی می کنیم.

۱) **3 ---- خانواده PLC S7300**
4 ---- خانواده PLC S7400

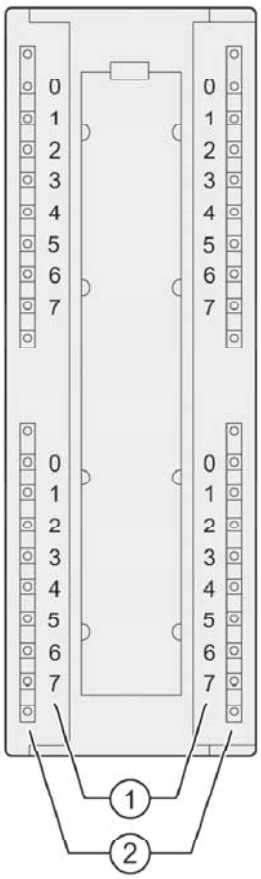
۲) **2 ---- سیگنال دیجیتال**
3 ---- سیگنال آنالوگ

۳) **1 ---- نوع ورودی**
2 ---- نوع خروجی
غیر از ۱ و ۲ ---- کارت‌های خاص

SM X X X
1 2 3

SM 321: این گروه از نوع ورودی دیجیتال بوده که دارای 8 یا 16 یا 32 ورودی دیجیتـال می باشند و ولتـاژ تغذیه آنهـا می توانـد 24 VDC ، 120/230VAC و یا 48VDC باشند. در این کارت ها به علت وجود نویز در محیط های صنعتی، برای ارتباط از کوپل کننده های نوری استفاده شده است. همچنین از مکانیزم های لرزش گیری نیز استفاده شده است تا به کارت آسیبی نرسد. برای هر ورودی نیز یک L D در نظر گرفته شده است که با فعال شدن ورودی مورد نظر ،LED متناظر آن نیز روشن می شود. کاربر می تواند با توجه به توانایی های کارت و موارد استفاده آن، کارت خود را سفارش دهد. در روبرو تصویر آن نیز نشان داده شده است.

در زیر تصویر یک نمونه کارت ورودی دیجیتال، با 32 ورودی نشان داده شده است.



۱- کانال های مختلف یک کارت

۲- LED نشان دهنده وضعیت ورودی

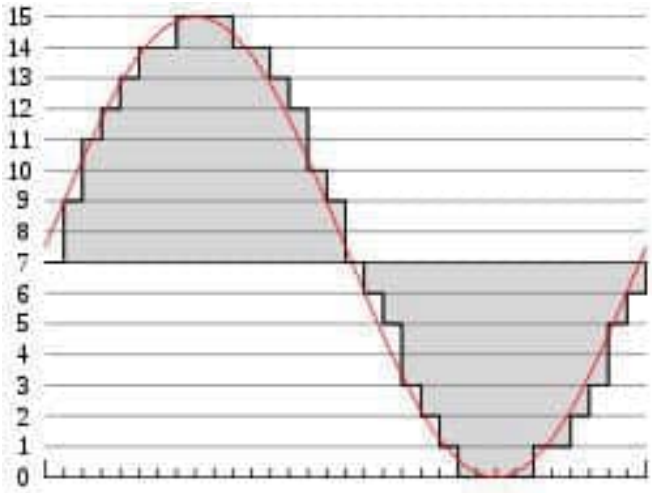
SM 322: این گروه از نوع خروجی دیجیتال می باشند و دارای تعداد 8 یا 16 یا 32 خروجی دیجیتال بوده که بسته به مدل کارت ، خروجی آن می تواند ترانزیستوری یا رله ای باشد.کاربر می تواند با توجه به نیاز خود، کارت مورد نظر خود را انتخاب نماید. شکل زیر یک کارت با 8 خروجی دیجیتال از نوع رله ای را نشان می دهد.

SM 323: این گروه شامل ورودی و خروجی دیجیتال به طور همزمان است یعنی بر روی یک کارت هم ورودی دیجیتال موجود است و هم خروجی دیجیتال که تعداد آنها با توجه به نیاز کاربر قابل انتخاب است.این نوع کارت ها زمانی که کاربر به تعداد محدودی ورودی و خروجی نیاز دارد و از نظر حداکثر تعداد کارت نیز در مضیقه باشد مورد استفاده قرار می گیرد و هزینه را نیز کاهش می دهد. برای مثال کاربر نیاز به حداکثر 16 ورودی دیجیتال و 10 خروجی دیجیتال دارد برای این کار باید دو کارت از خانواده های SM321 وSM322 تهیه نماید که هم با عث افزایش هزینه می گردد و هم یک اسلات از ریل بیشتر اشغال می شود ولی با استفاده از کارت های سری SM323 این مشکل به آسانی و با کمترین هزینه رفع می گردد.

SM 331: این گروه شامل کارت های ورودی آنالوگ می باشند که تعداد ورودی آن می تواند 4,8,12 باشد. ورودی های آنالوگ به 4 دسته تقسیم می شوند: ولتاژی ، جریانی ، مقاومتی و ترموکوپلی. تمام کارت های آنالوگ این ورودی ها را پشتیبانی نمی کنند به همین جهت کاربر باید با توجه به نیاز خود، کارت مورد نظر را انتخاب نماید و سفارش دهد. همان طور که می دانید CPU با مقادیر دیجیتال کار می کند پس ورودی های آنالوگ باید ابتدا تبدیل به دیجیتال شوند که این کار توسط A/D (مبدل آنالوگ به دیجیتال) صورت می پذیرد. یکی از پارامترهای مهم در این تبدیل ، قدرت

تفکیک (رزولوشن) این تبدیل می باشد که به بیت بیان می شود. این کار توسط کوانتیزه کردن (سطح بندی) مقدار آنالوگ صورت می پذیرد.تعداد سطوح کوانتیزه برابر است با $2^{\text{تعدادبیت}}$. با توجه به شکل زیر تعداد سطوح کوانتیزه برابر است با 16 که معادل 4 بیت می باشد.

$$2^4 = 16$$



مثال: اگر برای ولتاژ 10 v-0 از یک کارت 8 بیتی استفاده شود برای قدرت تفکیک آن داریم:

$$R = \frac{10v}{2^8} = 40mV$$

به ازای 40mV تغییر در ورودی ،یک سطح کوانتیزه سیگنال دیجیتال شده، تغییر می کند.

اگر از کارت 12 بیتی استفاده شود با توجه به محاسبات فوق داریم:

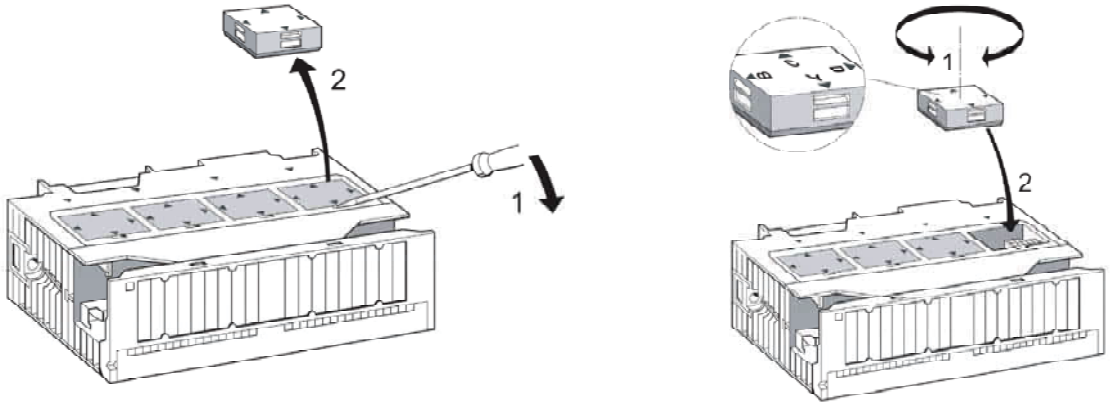
$$R = \frac{10v}{2^{12}} = 2.4mV$$

به ازای تغییر به اندازه 2.4mV ، کارت این تغییر را احساس می کند.

در بعضی از کارت های این گروه با توجه به اینکه ورودی آنالوگ از 4 گروه می تواند انتخاب شود باید نوع ورودی را به طور سخت افزاری نیز تنظیم نمود که این عمل در زیر نشان داده شده است. ابتدا کارت را از پهلو در دست بگیرید به

طوری که قسمت تنظیمات سخت افزاری به سمت بالا باشد و با اهرمی مطابق شکل قطعه مورد نظر را خارج کنید. اگر به قطعه خارج شده دقت کنید حروف A,B,C,D بر روی آن حک شده است. بر روی بدنه کارت توضیحات مربوط به این حروف قید شده است.

حال باید با توجه به توضیحات روی کارت قطعه را طوری در جایش قرار دهید که حرف مورد نظر به سمت فلش روی کارت باشد. برای مثال اگر ورودی ترموکوپل حرف C می باشد باید قطعه را طوری قرار دهید که حرف C در مقابل فلش قرار بگیرد. فراموش نکنید بعد از انجام تنظیمات سخت افزاری ، این تنظیمات باید به صورت نرم افزاری نیز صورت پذیرد.



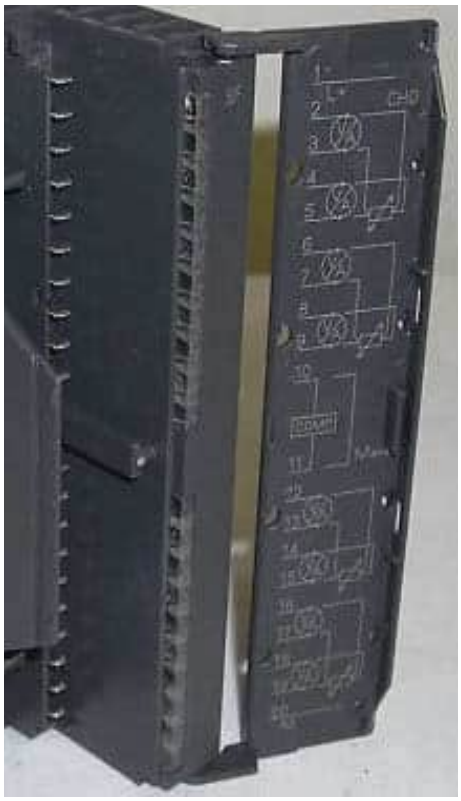
SM 332: این گروه کارت های خروجی آنالوگ را شامل می شوند که تعداد خروجی آن مطابق با نیاز کاربر می تواند 2، 4 یا 8 عدد باشد .خروجی های آنالوگ از نوع ولتاژ یا جریان می باشند که علاوه بر سیم کشی صحیح ، در نرم افزار نیز باید تعریف شود.

SM 334: این گروه شامل کارت هایی است که ورودی و خروجی آنالوگ را به طور همزمان دارا می باشند. که تمام موارد قید شده مانند: تعداد بیت تبدیل و نوع سیگنال های آنالوگ برای این گروه نیز صادق بوده و برای انتخاب کارت نیز باید این موارد را در نظر داشت.

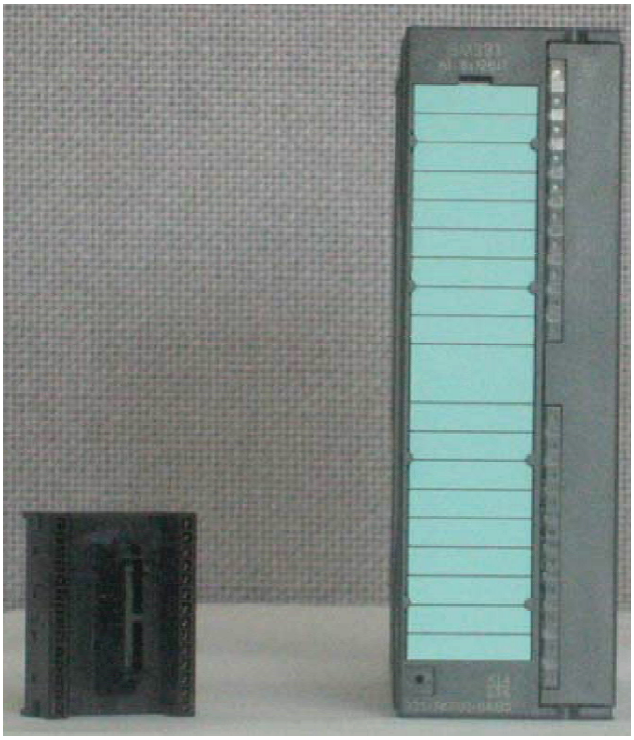
SM327: همان طور که از کد فنی این گروه مشخص است از خانواده سیگنال های دیجیتال می باشد. از ویژگی های این کارت می توان به انتخاب نوع کانال آن اشاره کرد. این کارت که با کد فنی 6ES7 327-1BH00-0AB0 قابل دسترس است شامل 8 ورودی دیجیتال می باشد و 8 کانال دیگر نیز قابلیت انتخاب دارند که به صورت ورودی استفاده شود یا به صورت خروجی. برای نمونه کاربر می تواند در این کارت 16 ورودی دیجیتال یا 8 ورودی دیجیتال و 8 خروجی دیجیتال را داشته باشد.

SM335: این گروه نیز شامل کارت های ترکیبی ورودی و خروجی آنالوگ می باشند. تفاوت این گروه با SM334 قابلیت های تنظیمی بیشتر در این گروه است. علاوه بر این ، دسته SM335 قابلیت ارسال سیگنال Interrupt را نیز دارا می باشند.

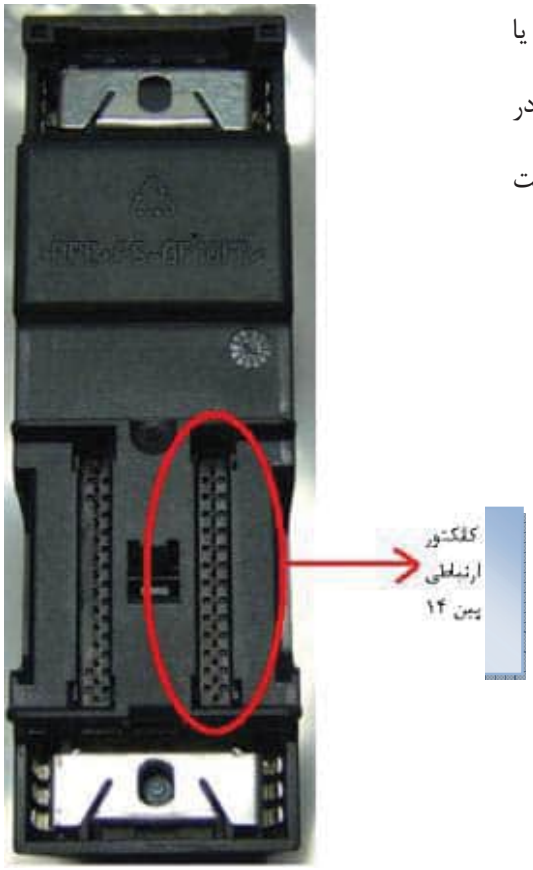
نحوه سیم کشی کارت ها به صورت شماتیک بر روی درب خود کارت ها رسم شده است به شکل زیر توجه نمائید. عدد هایی که قید شده است شماره پین مربوطه را نشان می دهد.



پس از نصب کارت های ارتباطی ، این کارت ها باید بتوانند با CPU ارتباط برقرار کرده تا به آن اطلاعات داده و از آن فرمان دریافت نمایند. این ارتباط از طریق یک کانکتور U شکل از پشت CPU و کارت ها انجام می شود. این کانکتور به همراه کارت ها در یک جعبه عرضه می شود. تصویری از این کانکتور به همراه یک کارت در زیر نشان داده شده است.



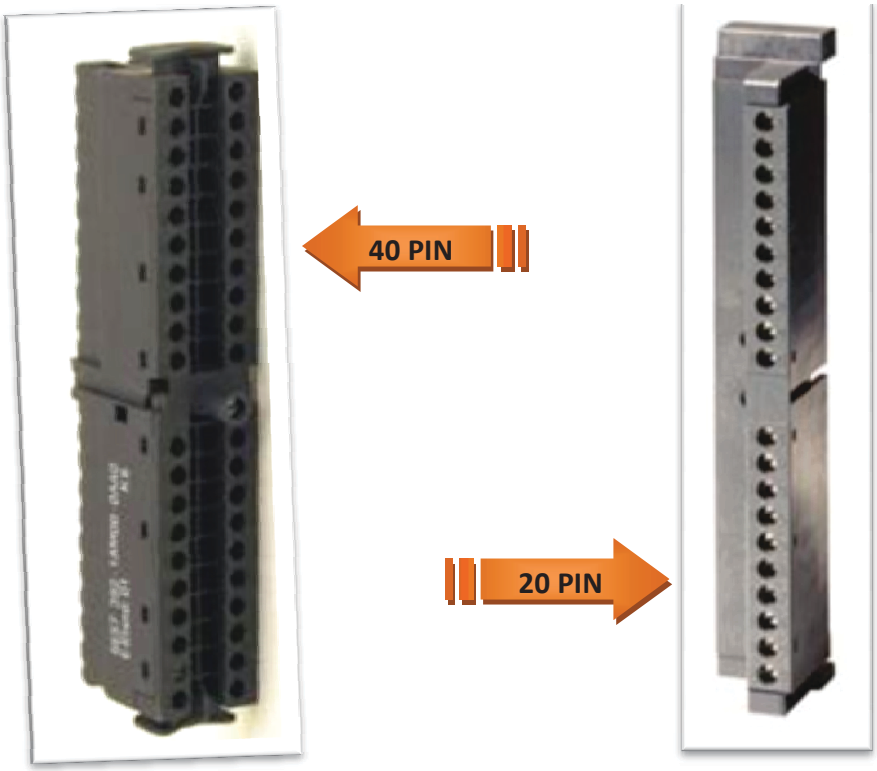
این کانکتور 14 پین از پشت به CPU و کارت وصل شده و این ارتباط را برقرار می نماید.در پشت کارت ها نیز محلی برای نصب این کانکتور در نظر گرفته شده است به شکل زیر توجه کنید.



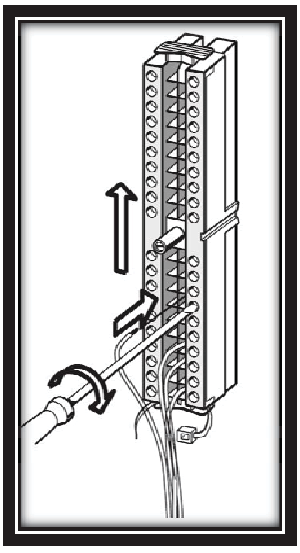
برای نصب این کانکتور باید دقت نمود که قبل از نصب CPU یا کارت بر روی ریل ، این کانکتور باید در جای خود قرار گیرد مگر در زمانی که کاربر بخواهد در اسلات بعدی کارتی قرار ندهد. در قسمت نصب سخت افزاری این مطلب به طور کامل بیان می گردد.

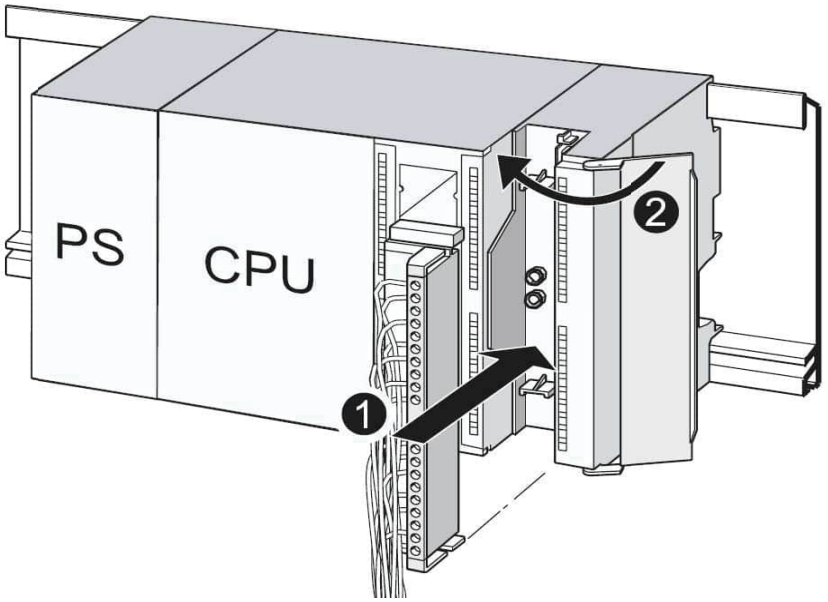
فرانت کانکتور(front connector):

همان طور که در تصویر نمای روبروی این کارت مشاهده نمودید در محل سیم کشی کارت ، محلی برای اتصال سیم در نظر گرفته نشده است. برای عملیات سیم کشی کارت و استفاده از کارت های ارتباطی نیاز به قطعه ای به نام فرانت کانکتور می باشد. این قطعه در دو نوع 20 و 40 پین موجود است که کاربر با توجه به نوع کارت خود، آن را انتخاب می کند. برای مثال کارت با 16 ورودی دیجیتال به یک فرانت کانکتور 20پین نیاز دارد ولی برای کارتی با 16 ورودی دیجیتال و 16خروجی دیجیتال که در مجموع 32 پین می شوند ،نیاز به یک فرانت کانکتور 40 پین می باشد.در یک کارت نمی توان به جای استفاده از یک کانکتور 40 پین از دو کانکتور 20پین استفاده نمود. در زیر تصویری از این دو کانکتور نشان داده شده است.



در شکل زیر نحوه سیم کشی و نصب فرانت کانکتور بر روی کارت نشان داده شده است.





ماژول های تابعی (Function Module):

انجام بعضی امور باعث کندی کار CPU می گردد مانند انجام عمل شمارش با سرعت بالا یا کنترل بعضی سیستم ها نیاز به امکاناتی دارد که ممکن است CPU این قابلیت را نداشته باشد مانند کنترل کننده های صنعتی PID . برای انجام این گونه عملیات های کنترلی ، سخت افزاری جداگانه به نام ماژول های تابعی طراحی شده است که کاربر می تواند با توجه به نیاز خود آن را انتخاب و از آن استفاده نماید. از فواید دیگر این ماژول ها می توان به کاهش حجم کاری CPU اشاره کرد. برای مثال در یک فرآیند ، عمل شمارش را می توان بوسیله یک ماژول شمارنده انجام داد تا حجم عملیاتی CPU کاهش یابد. از FM های پر کاربرد می توان به موارد زیر اشاره کرد:

- FM 350 که از نوع شمارنده های سریع می باشد که علاوه بر انجام عملیات شمارش معمولی قابلیت شمارش با سرعت بالا (500 KHz) را نیز دارد. 6ES7 350-1AH00-0AE0 شماره سفارش یکی از مدل های این نوع کارت می باشد.
- FM 351 عملیات کنترل موقعیت را انجام می دهد. 6ES7 351-1AH00-0AE0

(برای کاربرد کنترل موقعیت می توان به کنترل موقعیت بازوی یک ربات یا یک شافت موتور DC اشاره کرد)

- FM 353 برای کنترل موتور های پله‌ای (Step Motor) استفاده می شود. 6ES7 353-1AH00-0AE0
- FM 354 برای کنترل موتور های سرو کاربرد دارد. 6ES7 354-1AH00-0AE0
- FM 355 از نوع کنترل کننده های صنعتی PID می باشد. 6ES7 355-1VH00-0AE0

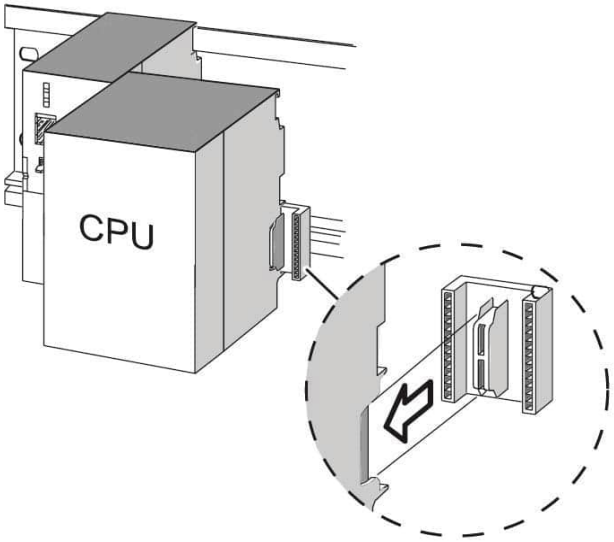
در شکل زیر تصویری از این FM ها نشان داده شده است.

FM 351

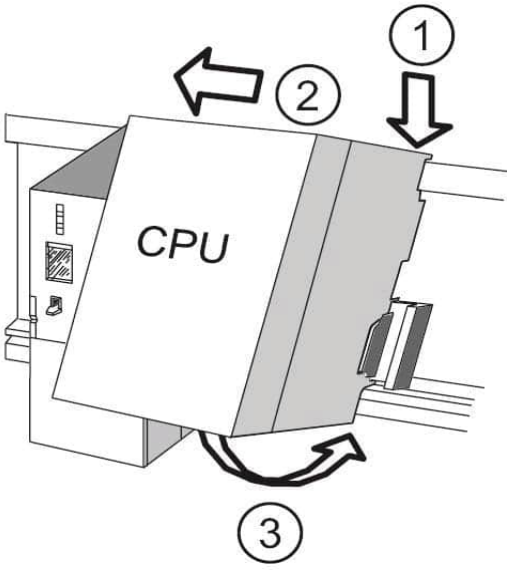
FM 355

مراحل نصب سخت افزار ها:

پس از آشنایی با سخت افزار های اصلی ، نوبت به نصب این سخت افزار ها است. در ابتدا ریل را در محل مناسب نصب کنید و بعد از اطمینان از محکم بدون آن ، منبع تغذیه را بر روی ریل قرار داده و پیچ نمائید. در مرحله بعد باید CPU را در کنار منبع تغذیه قرار داد.قبل از نصب CPU دقت کنید اگر می خواهید در کنار CPU از کارت های ارتباطی استفاده نمائید باید قبل از قرار دادن آن بر روی ریل ، کانکتور U شکل را در پشت آن نصب نمائید. به شکل زیر توجه کنید و به نحوه قرار گیری کانکتور دقت کنید.



بعد از نصب کانکتور CPU را مطابق شکل زیر بر روی ریل نصب کنید.

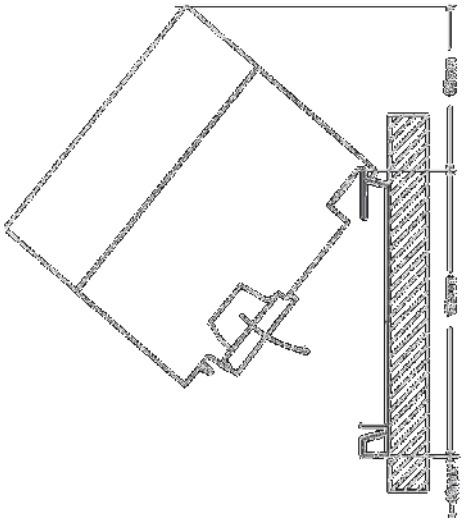


۱- قرار دادن قسمت فوقانی CPU بر روی ریل

۲- حرکت به سمت منبع تغذیه تا کامل در کنار آن قرار گیرد.

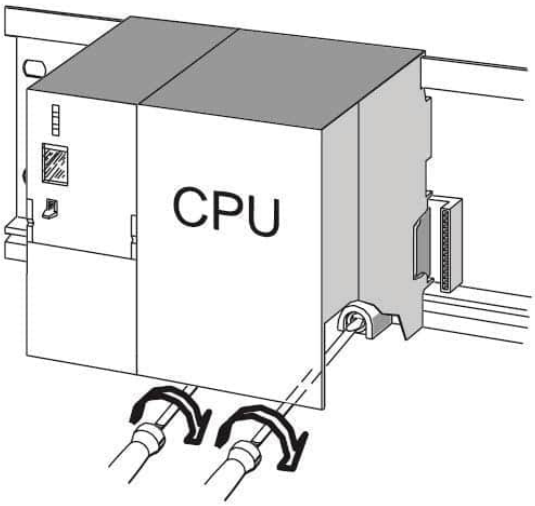
۳- حرکت به سمت پائین

نکته: کانکتور باید قبل از نصب CPU بر روی ریل ، وصل شود.

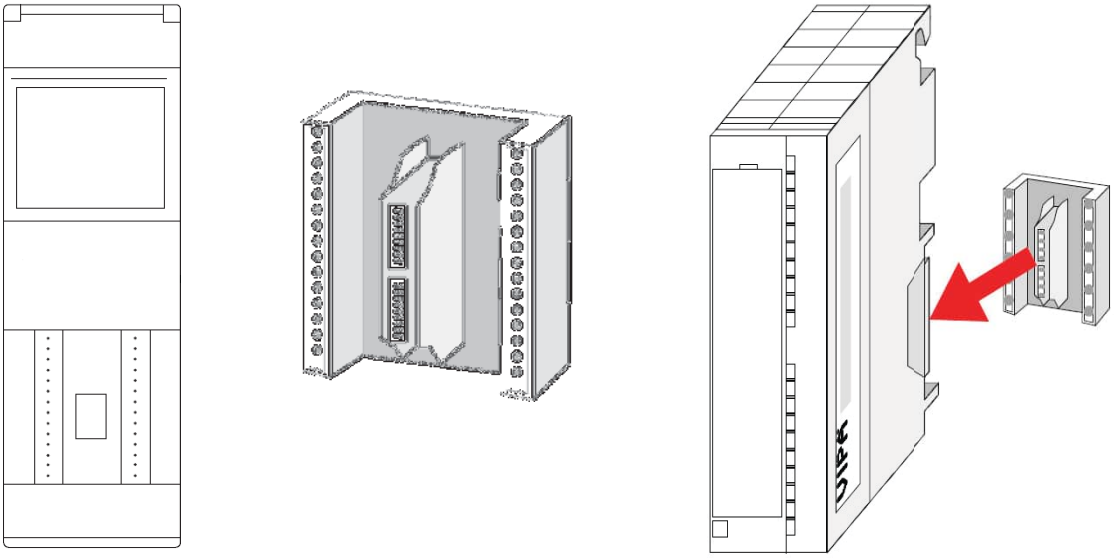


شیوه نصب از زاویه ای دیگر

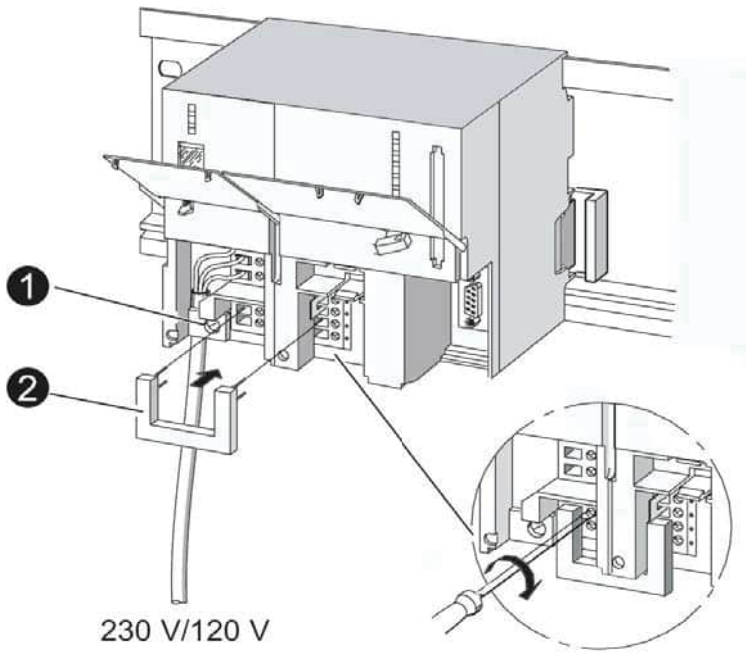
در انتها نیز باید پیچ های نگهدار را محکم کنید مطابق شکل زیر:



برای نصب کارت ها نیز به همین شکل اقدام می شود به شکل های زیر در مورد نحوه نصب کانکتور دقت کنید.



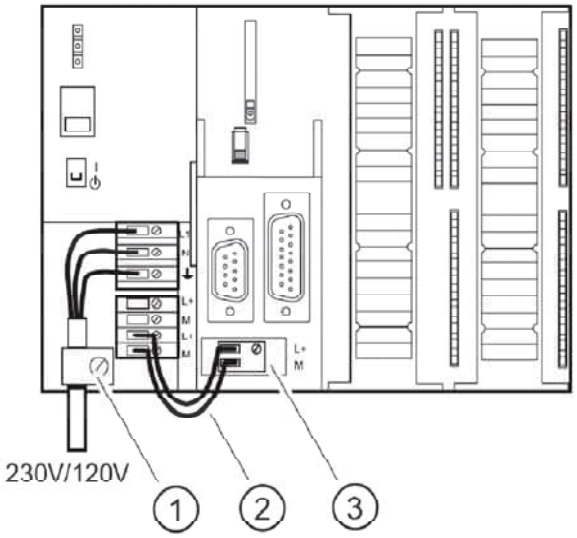
بعد از نصب سخت افزاری در مرحله بعد باید تغذیه CPU را متصل نمود که در شکل زیر نحوه اتصال آن نشان داده شده است.



۱- نوار نگهدار کابل برق ورودی

۲- کانکتور ارتباطی بین منبع تغذیه و CPU که در زیر نیز تصویر آن نشان داده شده است.

نحوه سیم کشی از نمای دیگر نیز در زیر نشان داده شده است.



۱- نوار نگهدارنده کابل ورودی

۲- کابل ارتباطی تغذیه بین منبع تغذیه و CPU

۳- کانکتور تغذیه CPU

حافظه:

(Memory)

حافظه محلی است که برنامه کنترل کننده و اطلاعات در آن ذخیره می گردد همچنین سیستم عاملی که مدیریت کلی PLC را بر عهده دارد در حافظه قرار دارد. حافظه در CPU به 4 دسته تقسیم بندی می شود که در زیر به آن اشاره می کنیم:

۱- حافظه بار گذاری Load memory این قسمت مخصوص برنامه های نوشته شده توسط کاربر است که شامل OB ها ، FB ها ،FC ها و DBها می باشد که در فصل نرم افزار با آن آشنا خواهیم شد.

۲- حافظه کاری (Work Memory) : فقط قسمت اجرایی برنامه به این بخش وارد می شود.برای مثال ممکن است برنامه نوشته شده شامل بلوک های داده باشد که اجرایی نیستند.این بلوک های داده در حافظه بار گذاری می مانند و به حافظه کاری انتقال داده نمی شوند.

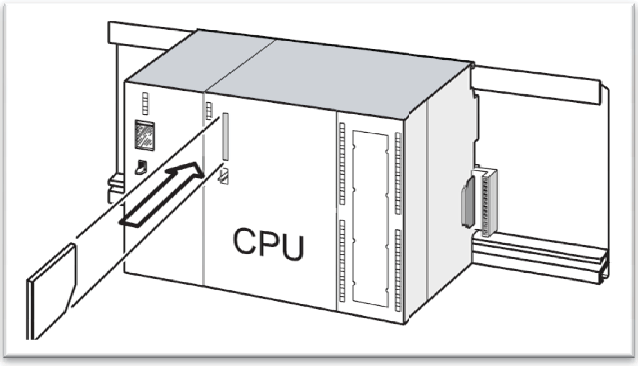
: حافظه بار گذاری را توسط کارت حافظه می توان افزایش داد ولی حافظه کاری به جز در CPU های سری 417 را نمی توان افزایش داد. در زیر تصویر یک کارت حافظه نشان داده شده است.

۳- حافظه سیستمی(System Memory): این ناحیه مخصوص خود CPU است و تایمر ها ، شمارنده ها، جداول PIIو PIQ ، بیت های حافظه و ... در آن قرار دارد.

۴- حافظه ماندگار (Retentive Memory) : به قسمتی از حافظه اطلاق می شود که در صورت قطع برق نیز پاک نخواهد شد. برای استفاده از این قسمت از حافظه باید از باتری ای پشتیبان استفاده کنیم یا برنامه را بر روی کارت ەای حافظه ذخیره کنیم.

کارت حافظه: همان طور که گفته شد این کارت برای توسعه فضای حافظه بار گذاری CPU به کار می رود و از آنجائیکه بعضی از مدل های CPU فاقد حافظه بار گذاری می باشند وجود این کارت برای شروع به کار الزامی است. در شکل زیر نحوه صحیح اتصال کارت به CPU نشان داده شده است.

دقت شود زمانی که CPU در حال اجرای برنامه است کارت را خارج نکنید. جهت اطمینان کار زمانی که CPU خاموش است کارت را وارد کنید.



باز نشانی حافظه (Reset) : برای پاک کردن حافظه سیستمی از روند زیر پیروی کنید.

- کارت را در داخل CPU قرار دهید و برق آن را وصل کنید.

- کلیدی بر روی CPU در نظر گرفته شده است که اگر در وضعیت پائینی باشد در حالت ریست قرار می گیرد.
- این کلید را به مدت 3 ثانیه به سمت پائین نگه دارید تا L D روی آن 3 بار چشمک زده و ثابت شود.
- حال سوئیچ را رها کنید و سپس دوباره برای یک لحظه سوئیچ را به سمت پائین فشار دهید این بار L D با سرعت بیشتری چشمک می زند. بعد از اتمام چشمک ، حافظه سیستمی بازنشانی می شود.

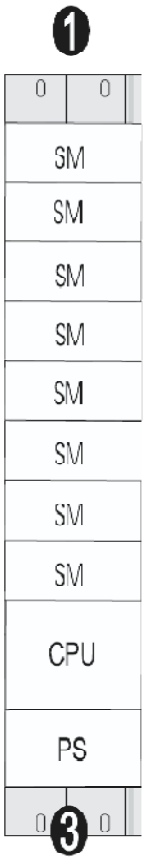
دقت شود این عمل حافظه سیستمی را باز نشانی می کند و در اطلاعات کارت حافظه تغییری ایجاد نمی شود.

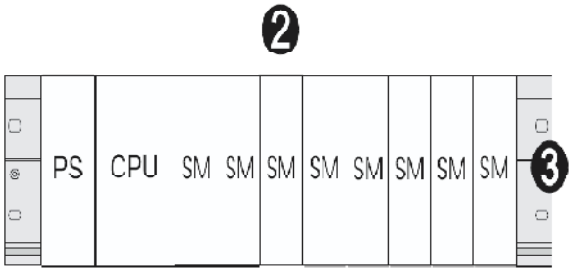
کابل ارتباطی به کامپیوتر:

این قطعه برای انتقال اطلاعات بین CPU و کامپیوتر به کار می رود. انتقال اطلاعات از کامپیوتر به CPU را دانلود کردن و انتقال اطلاعات از CPU به کامپیوتر را آپلود کردن می نامند. برای استفاده از این قطعه ابتدا باید نرم افزار راه انداز آن را نصب کنید که این نرم افزار به همراه خود قطعه می باشد و نیازی به خرید جداگانه آن نیست. این قطعه از دو کابل و یک آداپتور تشکیل شده است. پورت ارتباطی آن با کامپیوتر از نوع USB است و سر دیگر کابل نیز به CPU وصل می شود.در شکل زیر این قطعه نشان داده شده است.

بر روی این آداپتور 3 عددLED به نام های MPI, Power و USB قرار داده شده است. هرگاه اطلاعات بین CPU و کامپیوتر در حال انتقال باشد LED مربوط به MPI چشمک‌زن خواهد شد. هرگاه زمانی که تغذیه این آداپتور برقرار باشد LED مربوط به Power روشن خواهد شد و زمانی که کابل USB آن نیز برقرار باشد LED مربوط به آن نیز روشن خواهد بود.

با استفاده از سخت افزارهای معرفی شده در بالا می توانید یک سیستم کنترلی را راه اندازی کنید. نحوه نصب قطعات در مجموع به شکل زیر می باشد. دقت کنید حداکثر SMهایی که می توانید در یک رک به CPU متصل کنید برابر با 8 عدد می باشد. اطلاعات مربوط به توزیع رک ها در قسمت بعد توضیح داده خواهد شد.





۱- نصب عمودی

۲- نصب افقی

۳- ریل(رک)

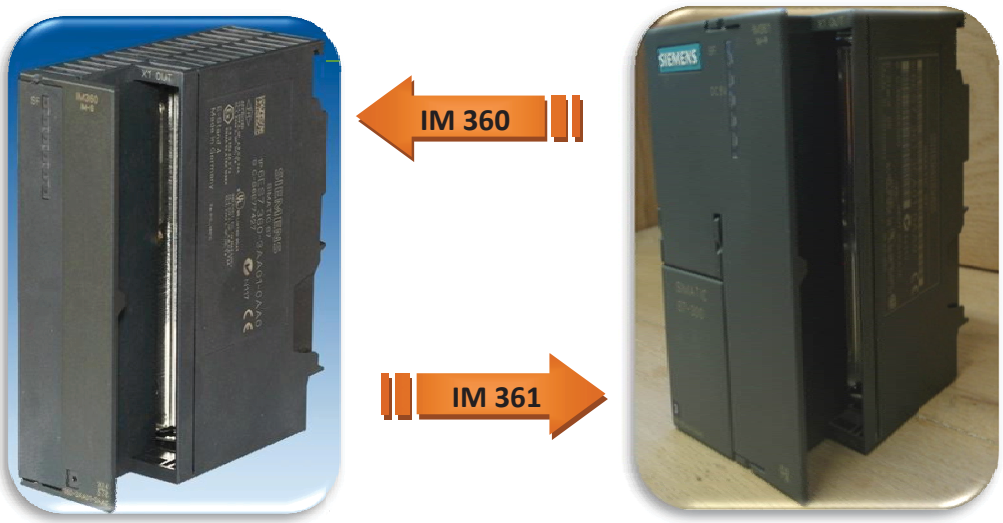
سخت افزار های خاص:

سخت افزار هایی که در قسمت قبل معرفی شدند برای راه اندازی یک PLC در شرایط عادی کافی می باشند. قطعاتی که در این قسمت مورد بررسی قرار خواهند گرفت قطعاتی هستند که در موارد خاص مورد کاربرد قرار می گیرند.

ماژول های واسطه (Interface module) : همان طور که در مباحث قبلی گفته شد یک CPU در یک رک بیشتر از 8 عدد SM را نمی تواند پشتیبانی کند و اگر بخواهیم کارت های دیگری اضافه کنیم باید از رک های توسعه یافته استفاده کنیم. دقت کنید با یک CPU حداکثر می توان 3 رک فرعی دیگر را پشتیبانی کرد و در مجموع 4ریل داشته باشید. برای ارتباط بین رک ها از IM ها استفاده می شود.

IM 360: این ماژول یک ماژول فرستنده است و در رک اول اسلات سه، دقیقاً بعد از CPU قرار می گیرد. شماره سفارش این ماژول 6ES7 360-3AA00-0AA0 است. تصویر این ماژول در زیر نشان داده شده است.

IM 361: یک ماژول گیرنده است و در رک های توسعه یافته به کار می رود. از آنجائیکه در رک های توسعه یافته دیگر نیاز به CPU نیست این ماژول در کنار منبع تغذیه قرار می گیرد ولی باید دقت شود که در نرم افزار باید در اسلات سوم قرار بگیرد و اسلات مربوط به CPU خالی باشد. 6ES7 361-3CA00-0AA0 شماره سفارش این ماژول می باشد.



در شکل زیر نمای کلی ساختار رک های توسعه یافته نشان داده شده است.

۱- رک مرکزی

۲- رک توسعه یافته اول

۳- رک توسعه یافته دوم

۴- رک توسعه یافته سوم

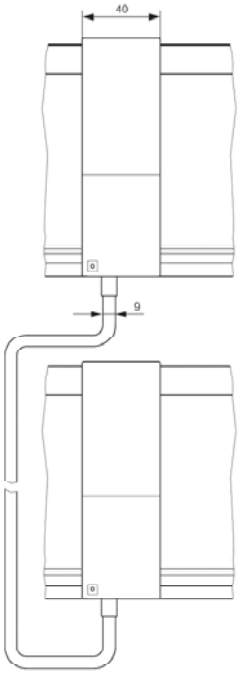
۵- کابل های ارتباطی

۶- آخرین کارتی که می توان اضافه کرد

برای ارتباط بین این دو ماژول از کابل ارتباطی زیر استفاده می شود. همان طور که در شکل هم مشاهده نمودید رک های فرعی دارای CPU نمی باشند ولی به منبع تغذیه جداگانه نیاز دارند و IM361 باید به آن متصل شود.نحوه سیم کشی مانند سیم کشی CPU می باشد.

کابل های ارتباطی در طول های 1,2.5, 5 ,10 متری وجود دارند که ، یک شماره سفارش مخصوص به خود دارند. همانطور که از طول کابل ها مشخص است حداکثر فاصله بین دو رک می تواند 10 متر باشد.اگر محیطی که از این ساختار در آن استفاده می شود دارای نویز زیادی باشد این فاصله کاهش می یابد.

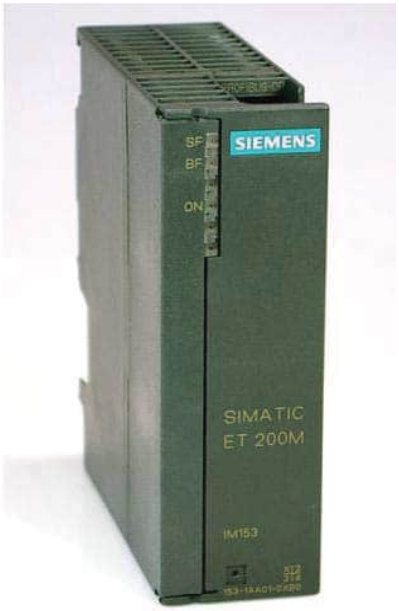
IM 365: این ماژول از دو قسمت send و receive تشکیل شده است و عملکردی مشابه IM 360و IM 361 دارد با این تفاوت که با IM 365 فقط می توان یک رک توسعه یافته داشت و حداکثر فاصله این دو رک نیز 1m می باشد. هزینه استفاده از این روش نسبت به حالت قبلی کمتر است و رک دوم نیز نیاز به منبع تغذیه ندارد. در شکل زیر IM 365 و تصویر شماتیک آن نشان داده شده است.

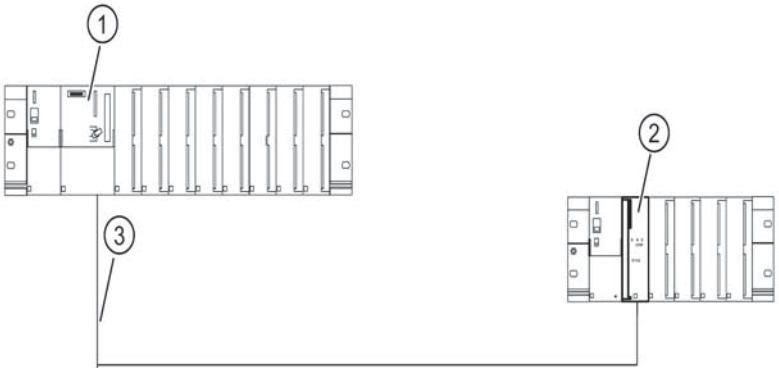


ET 200M: این گروه ، ماژول های واسط تحت شبکه پروفیباس هستند که مهمترین و پرکاربردترین عضو این مجموعه ماژول واسط IM 153-1 است.

IM 153-1: این ماژول برای سیستم های استاندارد به کار می رود و در شکل زیر ساختار مربوطه به استفاده از این ماژول نشان داده شده است. حداکثر تعداد رکی که با این ساختار می توان داشت برابر با 126 عدد است و حداکثر فاصله بین دو رک بدون استفاده از تقویت کننده نیز برابر با 100m خواهد بود. بر روی این قطعه تعدادی دیپ سوئیچ گنجانده شده است که برای تنظیم آدرس شبکه این ماژول می باشد دقت شود که آدرس سخت افزاری و آدرس تنظیمی در نرم افزار یکی باشد.

نمایی از قطعه IM153 در شکل رو به رو نشان داده شده است.



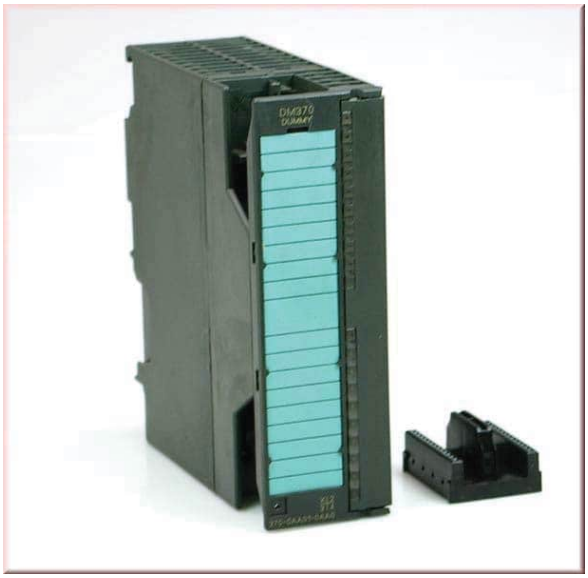


۱- CPU اصلی دارای پورت DP

۲- رک دارای ماژول ET 200

۳- کابل شبکه پروفیباس

ماژول دامی (Dummy Module): این ماژول برای رزرو کردن یک اسلات برای کارت های ارتباطی به کار می رود. از نظر شکل ظاهری کاملاً شبیه ماژول های خانواده SM می باشد و بر روی رک قرار می گیرد تا عملیات نصب سخت افزار انجام شده و پس از رسیدن کارت اصلی ، کارت را جایگزین آن می کنند. همان طور که در شکل دیده می شود این کارت کاملاً شبیه کارت های SM می باشد و یک عدد کانکتور ارتباطی نیز به همراه آن وجود دارد.



پردازنده های ارتباطی: CP (Communication Processor) اسم اختصاری این گروه از ماژول ها می باشد. این گروه برای ایجاد شبکه های مختلف به کار می رود. اطلاعات تکمیلی در بخش شبکه قابل مشاهده است.

CP 341 مخصوص شبکه Modbus ، Cp342-5 جهت برقراری ارتباط با شبکه Profibus ، Cp343-2 برای شبکه های AS-I و CP343-1 نیز برای شبکه اترنت صنعتی می باشد.در شکل زیر یک نمونه Cp342-5 نشان داده شده است.

فصل دوم

آشنایی با مفاهیم اولیه نرم افزار

در این فصل آشنا خواهید شد با

- در این فصل با مفاهیم اولیه PLC
- آدرس دهی به ورودی ، خروجی و فضای حافظه
- آغاز به کار با نرم افزار و پیکربندی سخت افزار

آدرس دهی:

در PLC های سری 300 و 400 برخلاف LOGO عملیات برنامه نویسی با سیم کشی بین بلوک ها انجام نمی گیرد برای ارتباط بین بلوک ها ، باید از آدرس دهی به آنها استفاده نمود جهت ارتباط با فضای خارج و فضای حافظه ، کاربر باید با نحوه آدرس دهی در PLC آشنایی کامل داشته باشد. در PLC فضای حافظه بین 3 گروه تقسیم شده است که عبارتند از: ورودی ها (I) ، خروجی ها (Q) و فلگ ها (M). کوچکترین فضای حافظه **بیت** نامیده می شود که از کنار هم قرار گرفتن 8 بیت یک **بایت** تشکیل می گردد برای آدرس دادن به یک بیت مشخص، باید شماره بیت را به همراه شماره بایت مربوطه ذکر کنیم. به نحوه آدرس دهی بیت ها دقت کنید

شکل روبه رو نشان دهنده یک بایت می باشد.

7	6	5	4	3	2	1	0

مثال:

I 2.5: بیت پنجم از بایت دوم ورودی

Q 4.7: بیت هفتم از بایت چهارم خروجی

M 24.0: بیت صفرم از بایت بیست و چهارم حافظه

I 12.8: چنین آدرسی وجود ندارد چون این آدرس به بیت هشتم اشاره می کند که وجود ندارد

** دقت کنید که بیت ها از عدد صفر شماره گذاری می شوند

از کنار هم قرار گرفتن دو بایت یک WORD تشکیل می شود که شامل 16 بیت می باشد. اگر دو WORD کنار هم قرار گیرند دابل WORD را ایجاد می کنند که شامل 32بیت یا 4 بایت می باشد. برای آدرس دادن به یک بایت فقط ذکر شماره بایت کافی است برای مثال:

IB 2: بایت دوم ورودی

MB 6: بایت ششم از فضای حافظه

QB 3: بایت سوم خروجی

برای آدرس دادن یک WORD از قالب زیر استفاده می کنیم:

QW 3 ، IW 6، MW 0

در آدرس دهی WORD باید نکته مهم زیر را در نظر بگیرید:

MW0=MB 0 +MB 1

MW1=MB 1 + MB 2

همان طور که در دو رابطه بالا می بینید MW 0 از دو بایت صفر و یک تشکیل شده است و MW 1 نیز از دو بایت یک و دو اگر کاربر از دو آدرس MW 0 و MW 1 به طور همزمان استفاده کند چون بایت یک در هر دو وجود دارد به همین دلیل تغییر در یک MW اطلاعات MW دیگر را خراب می کند لذا نحوه صحیح آدرس دهی یک WORD به شکل زیر می باشد: MW 0,MW2,MW4,... یا MW1,MW3,MW5,...

نکته فوق برای دابل WORD نیز باید در نظر گرفته شود. از آنجائیکه دابل WORD شامل 4 بایت می باشد آدرس دهی ها باید به گونه ای باشد که بایت ها با هم تداخل نداشته باشند. برای مثال:....,MD 0,MD4,MD8

فرمت اعداد:

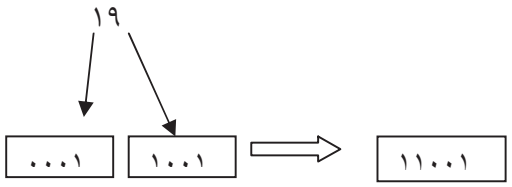
اعداد در PLC به فرمت های مختلفی قابل بیان هستند که در زیر هر کدام را توضیح می دهیم:

۱- **دسیمال**(DECIMAL) : این فرمت همان مبنای دهدهی معمولی است که در حالت عادی افراد استفاده می کنند.

۲- **باینری** Bin**):** این اعداد در مبنای دو می باشند و از ترکیب اعداد 0,1 بدست می آید. برای مثال ترکیب 10010 معادل عدد 18می باشد

۳- **هگزا**(HEX) : این اعداد در مبنای 16 می باشند برای مثال عدد 31 درمبنای HEX به فرم 1F و در فرم باینری به فرم 11111 می باشد

۴- **بی سی دی** BCD[1] : در این فرمت هر عدد به تنهایی به باینری کد می شود و سپس اعداد باینری در کنار هم قرار می گیرند. برای مثال عدد 19 در فرم BCD به صورت زیر بیان می شود



دقت شود که عدد 19 در فرمت باینری به شکل 10011 خواهد بود

اگر عدد باینری زیر معرف یک عدد BCD باشد داریم:

۱۰۰۱۰۰۱۱۰۱۱۱۰۰۰۱

۹ ۳ ۷ ۱

۵- **اعداد صحیح** (integer): این اعداد دارای طول 16 بیتی می باشند و اعداد صحیح بین 32768- الی 32768 را شامل می شوند و در نرم افزار نیز با عبارت INT مشخص می شوند

۶- **اعداد صحیح دابل**(duple integer): همان طور که از اسم این گروه مشخص است طول آن دو برابر گروه INT می باشد یعنی 32 بیت از فضای حافظه را اشغال می نماید این گروه نیز شامل اعداد صحیح می باشند ولی دامنه بیشتری از اعداد را شامل می شوند این اعداد از 214743648 – الی 214743648 است

[1] Binary coded decimal

این گروه از اعداد در نرم افزار با نام DINT مشخص می شوند این اعداد با فرمت (عدد مورد نظر#L) در نرم افزار وارد می شوند

*** دقت شود که اعدادی صحیح که در بازه هر دو گروه INT و DINT هستند را می توان به هر دو فرمت داشته باشیم ولی باید دقت شود که فرمت DINT دو برابر INT فضا اشغال می کند

۷- **اعداد حقیقی یا اعشاری**: این اعداد با نام REAL معرفی می شوند و در نرم افزار نیز با حرف R نشان داده می شوند. طول این اعداد 32 بیت می باشد و از عدد 1.175995e-38- الی 3.402823e+38 را شامل می شود برای وارد کردن این اعداد باید از علامت اعشار استفاده نمود حتی برای اعداد صحیح. برای مثال : 5.0

*** برای اعداد منفی از متمم دو آن استفاده می شود بدین صورت که ابتدا عدد را به صورت مثبت به باینری تبدیل می کنند و سپس تمام یک ها به صفر و تمام صفر ها به یک تبدیل می شود(متمم عدد) در پایان هم عدد به دست آمده را با یک جمع می کنند به مثال زیر توجه کنید:

برای عدد 9- داریم:

9 ⟶ (تبدیل به باینری) 00001001 ⟶ (متمم کردن اعداد) 11110110 ⟶ (مع با عدد یک) 11110111

نحوه عملکرد PLC:

دانستن نحوه عملکرد PLC در اجرای یک برنامه برای کاربر الزامی است زیرا ممکن است کاربر برنامه ای را بنویسد ولی وقتی قصد اجرا با PLC را داشته باشد جواب درست را دریافت نکند که این ناشی از همین عدم آگاهی با نحوه عملکرد PLC است. برای این منظور دانستن چند مفهوم در ابتدا الزامی است

سیکل زمانی: مدت زمانی است که CPU نیاز به انجام محاسبات دارد. که حداکثر این زمان 150 ms می باشد

جدول PII : در این جدول، تصویر تمامی ورودی ها در ابتدای هر سیکل کاری قرار داده می شود و برای انجام محاسبات از تصویر آنها استفاده می گردد و این تصاویر تا پایان سیکل کاری برنامه ثابت هستند

جدول PIQ : نحوه کار این جدول نیز مانند جدول PII بوده با این تفاوت که تصویر خروجی ها در آن قرار داده می شود

در PLC های سری 300 کاربر به این جداول دسترسی ندارد

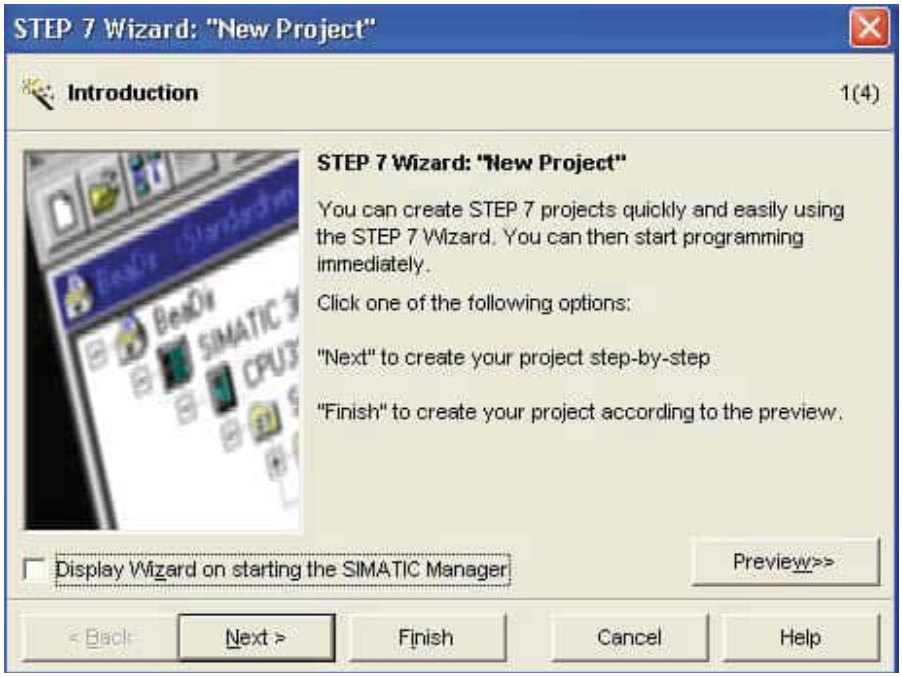
وقتی شما برنامه ای را می نویسید و آن را اجرا می کنید CPU مقدار تمام ورودی ها و وضعیت آنها را می خواند و در جدول PII ذخیره می کند و همین عمل را برای خروجی ها انجام داده و در جدول PIQ ذخیره می نماید. سپس شروع به اجرای برنامه از ابتدا به صورت خط به خط می نماید و در نهایت خروجی های بدست آمده را در جدول PIQ وارد می نماید و دوباره به ابتدای برنامه رفته و جدول PII را آپدیت می کند و برنامه را از نو اجرا می نماید اگر در حین اجرای سیکل برنامه ورودی تغییر کند CPU قادر به تشخیص آن نیست تا زمانی که سیکل به انتها برسد و دوباره جدول PII را آپدیت کند و متوجه تغییر ورودی شود

نکته: دقت شود اگر یک خروجی در دو جای برنامه استفاده شود با توجه به روند عملکر CPU کـه در بـالا توضیـح داده شد همیشه آخریـن مقدار آن خروجـی در جدول PIQ ثـبت می شود و PLC همـان خروجی را به کـارت های خروجی اعمـال می کند شکل زیر نحوه عملکرد CPU را نشان می دهد

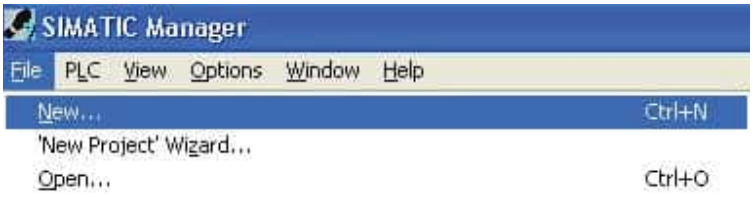
کار با نرم افزار:

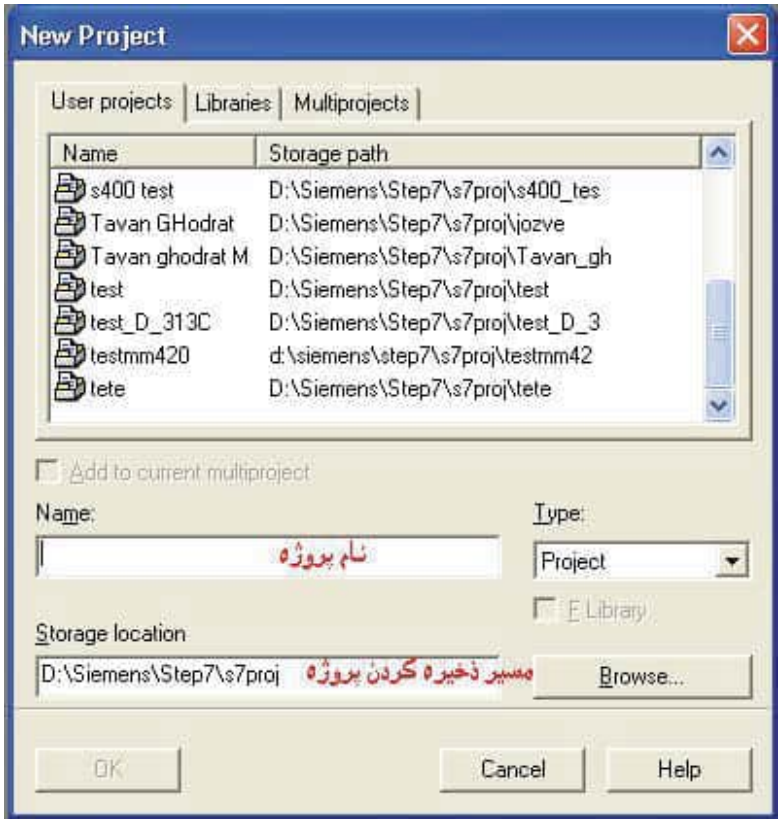
جهت ایجاد ارتباط با PLC های سری S7 از نرم افزار SIMATIC Manager استفاده می گردد. پس از نصب نرم افزار ، آیکن مربوط به آن بر روی صفحه کامپیوتر قابل مشاهده است

با دابل کلیک بر روی این آیکن وارد نرم افزار شده که کادر زیر ظاهر می گردد

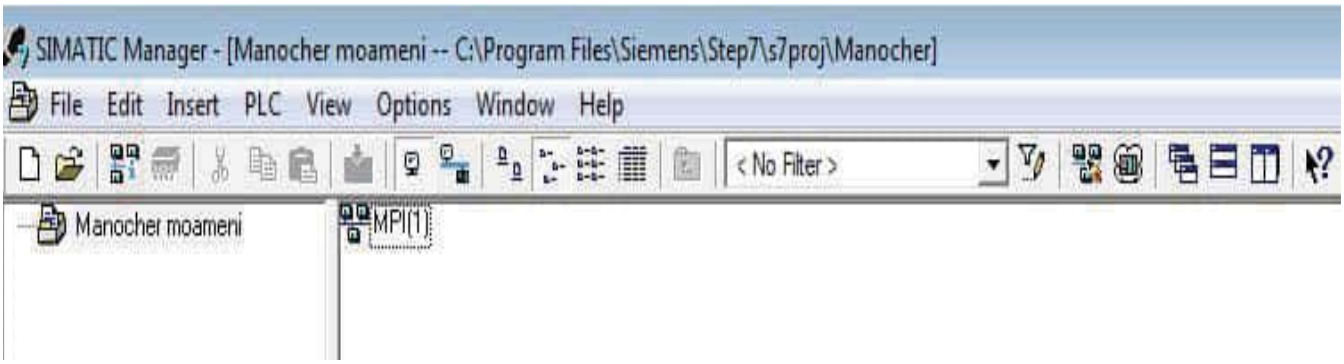


این پنجره مربوط به ایجاد یک پروژه جدید است. که با کلیک بر روی گزینه Next به مرحله بعدی وارد می شویم. گزینه Finish اتمام مراحل را اعلام می کند به علت محدودیت های این شیوه ، کاربران حرفه ای از این روش اقدام نمی کنند. روی گزینه Cancel کلیک کرده تا پنجره فوق بسته شود. حال از منوی File گزینه New را انتخاب کنید تا پنجره مربوط به آن باز شود در این پنجره نام پروژه و محل ایجاد پروژه را وارد نمائید

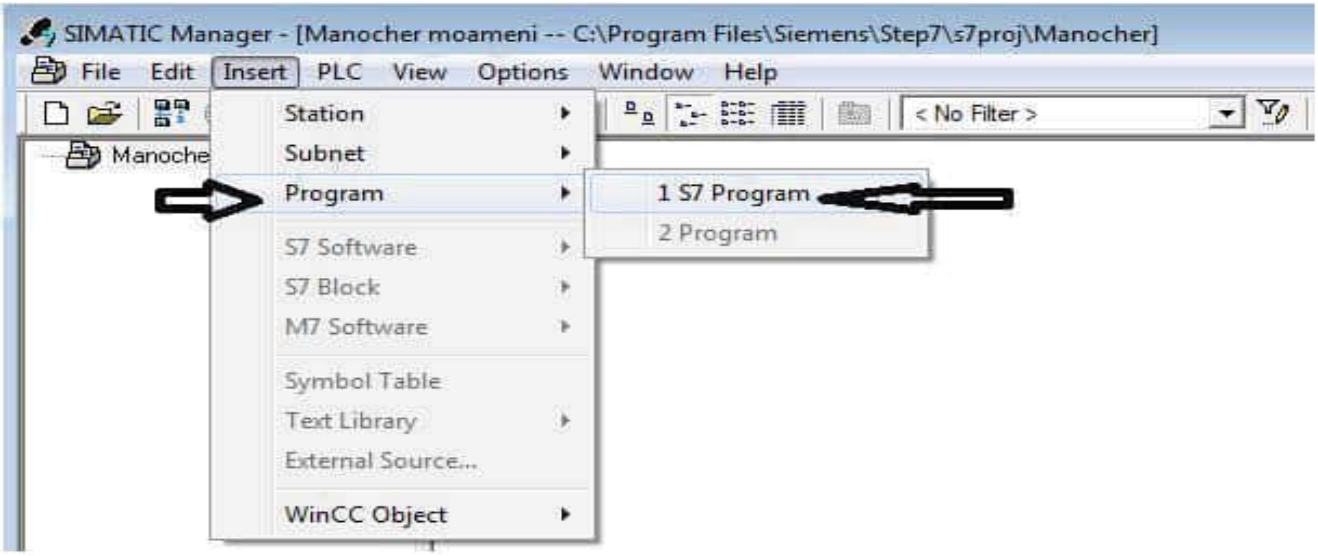




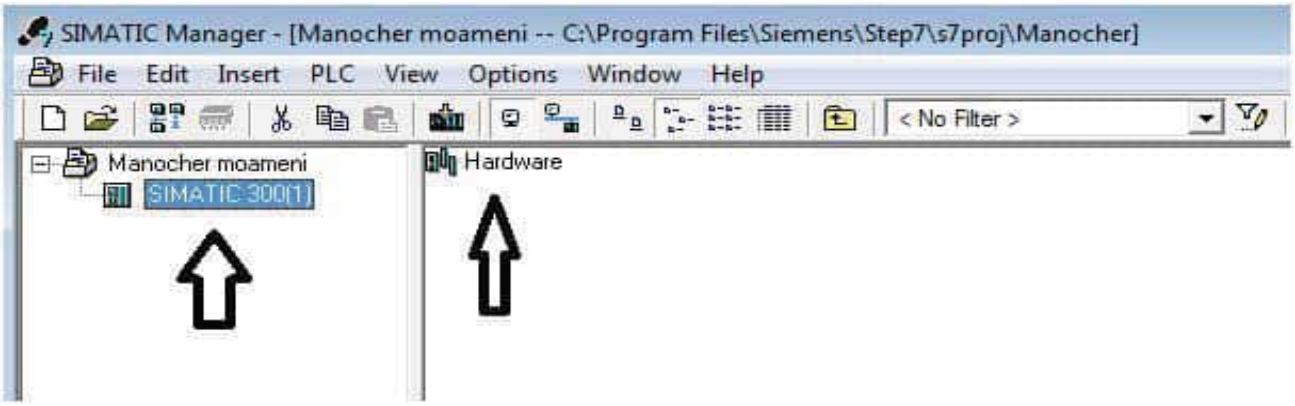
بعد از انجام مراحل فوق بر روی گزینه OK کلیک کنید تا صفحه اصلی برنامه مشاهده گردد



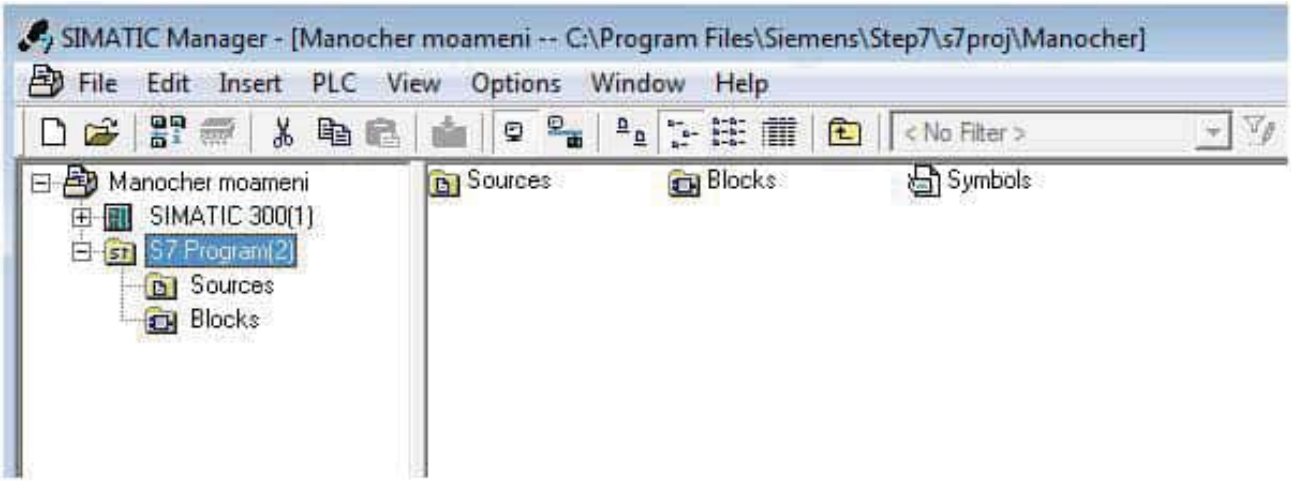
حال روی نام پروژه کلیک کنید و از منوی Insert و از گزینه Station ایستگاه کاری خود را ، که SIMATIC Station 300 است انتخاب کنید و نام دلخواه خود را بر روی ایستگاه قرار دهید



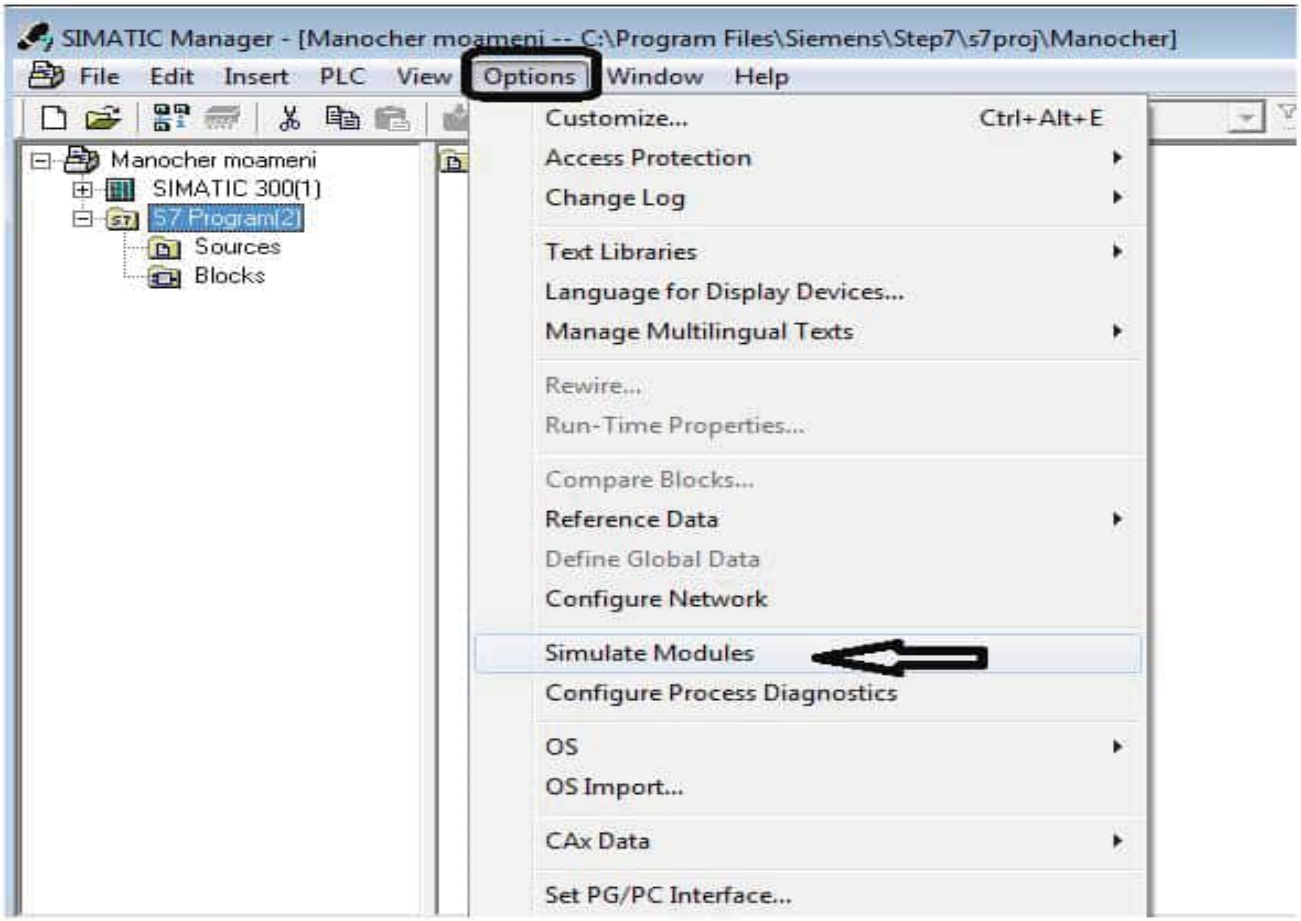
اگر بر روی ایستگاه ایجاد شده کلیک کنید با گزینه Hardware در صفحه سفید سمت راست رو به رو خواهید شد که برای تنظیم سخت افزاری استفاده می شود که انجام این تنظیمات به طور کامل در ادامه توضیح داده شده است



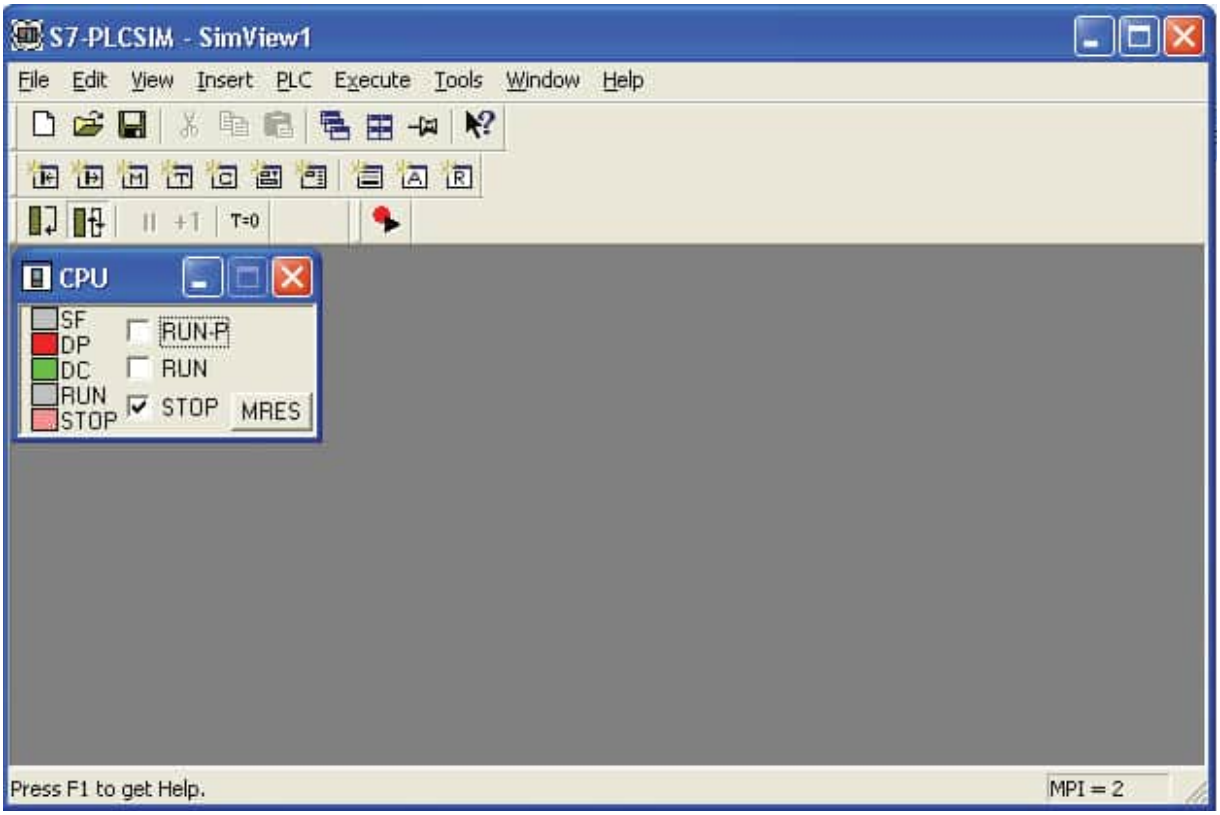
بعد از انجام این مراحل روی نام پروژه خود کلیک کنید و از منوی Insert گزینه Program و سپس گزینه S7 Program را انتخاب کنید تا گزینه مربوط به آن اضافه گردد. در ضمن شما می توانید نام برنامه خود را به دلخواه تغییر دهید



محیط شبیه سازی(Simulation) : این محیط برای انجام تست و بررسی برنامه نوشته شده به کار می رود که در آن کاربر قابلیت اضافه کردن ورودی و خروجی و مشاهده رفتار شمارنده ها و تایمر ها را دارد. از طریق منوی Option گزینه Simulated Module یا انتخاب آیکن در صفحه اصلی  نرم افزار، می توانید این صفحه را مشاهده نمائید



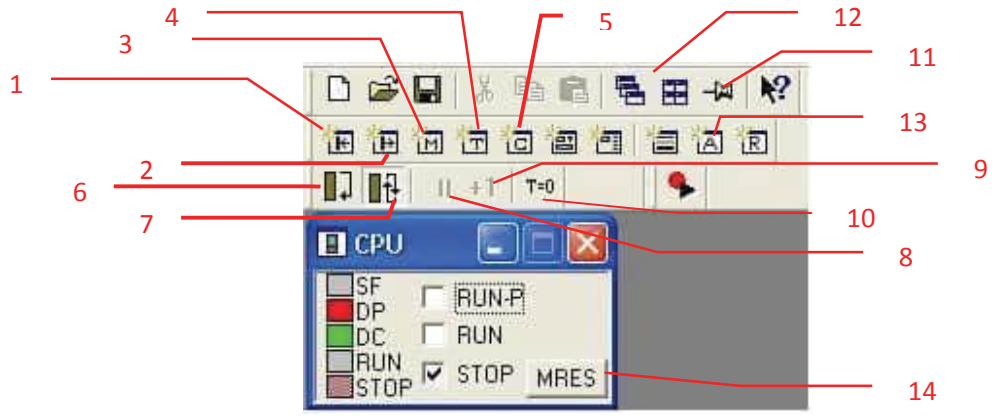
با انتخاب یکی از گزینه های فوق صفحه شبیه سازی مطابق با شکل زیر باز می شود



همان طور که در شکل بالا می توانید ببینید در ابتدا، صفحه مربوط به مشخصات CPU وجود دارد که می توانید با انتخاب گزینه مورد نظر حالت های مختلف آن را مشاهده نمائید

- Stop: به این معنی است که CPU در حالت توقف است و عملی را انجام نمی دهد
- RUN: با انتخاب این گزینه CPU شروع به اجرای برنامه می نماید و چراغ سبز مربوط به آن ابتدا شروع به چشمک زدن می کند و سپس ثابت می شود
- RUN-P: این گزینه همانند گزینه RUN عمل می نماید با این تفاوت که وقتی روی گزینه RUN قرار داشته باشد CPU اجازه Download برنامه جدید را نمی دهد ولی در حالت RUN-P این اجازه داده میشود برای استفاده از شبیه ساز بهتر است این گزینه انتخاب شود تا مشکلی در هنگام کار ایجاد نگردد
- MRES: برای پاک کردن فظای حافظه به کار می رود. با کلیک بر روی این گزینه برنامه دانلود شده نیز پاک می گردد و برای استفاده مجدد باید OB1 را دوباره دانلود نمود

نوار ابزار این صفحه موارد زیر را شامل می شود که در شکل مربوط به آن نشان داده شده است

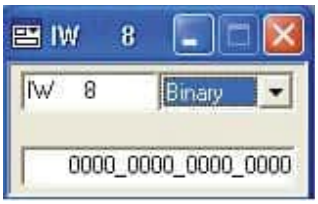


۱- ورودی ها I) ۲- خروجی ها Q) ۳- حافظه M) ۴- تایمرها T) ۵- شمارنده ها C)

۶- اجرای سیکلی ۷- اجرای پیوسته ۸- درنگ در اجرای برنامه Pause) ۹- اجرای اسکن بعدی

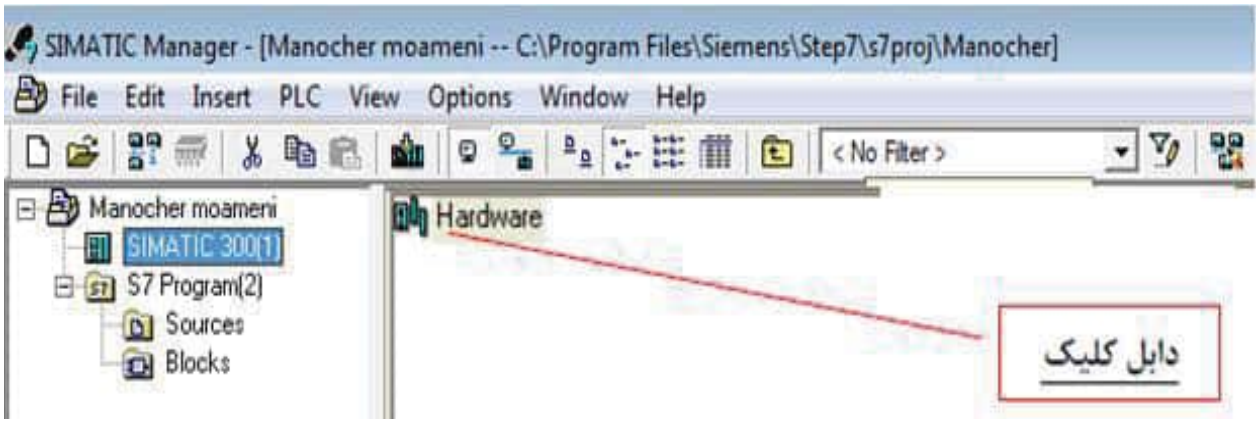
۱۰- صفر کردن همه تایمرها ۱۱- با انتخاب این گزینه صفحه شبیه سازی همیشه بالاترین صفحه باقی می ماند ۱۲- تعیین نحوه چینش پنجره ها ۱۳- بیت وضعیت ۱۴- پاک کردن حافظه

** با انتخاب یک ورودی می توان فرمت های مختلفی برای نمایش آن در نظر گرفت برای مثال اگر یک ورودی را وارد کنید با وارد کردن نوع ورودی مانند : IB,IW,ID به همراه شماره مورد نظر، می توان به آن ورودی دسترسی پیدا کرد. همچنین می توان با استفاده از کادر موجود نحوه نمایش ورودی را نیز انتخاب نمود به شکل های زیر توجه کنید این تغییرات برای خروجی ها و حافظه ها هم قابل اجراست



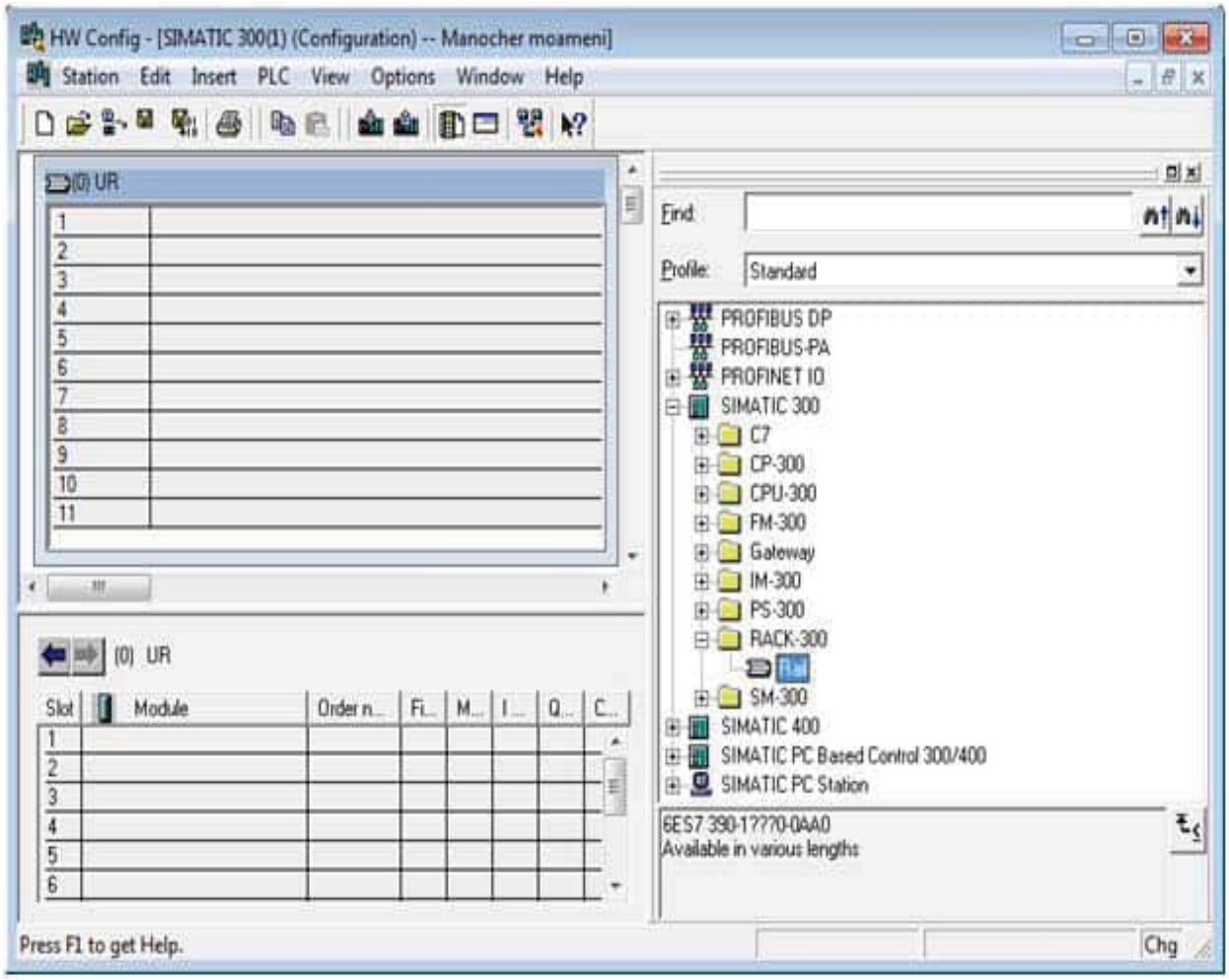
پیکربندی سخت افزار: (HARDWARE Configuration)

برای کار با PLC ابتدا باید سخت افزاری را که درحال کار است برای نرم افزار PLC معرفی گردد با این کار به نرم افزار PLC فهمانده می شود که با چه CPU ای در حال کار کردن است یا منبع تغذیه آن چیست و یا کارت های متصل به CPU به چه تعدادی هستند و نوعشان چیست تا نرم افزار قابلیت ها و محدودیت های موجود برای قطعات را در روند برنامه نویسی در نظر بگیرد. به این عمل پیکر بندی سخت افزاری گفته می شود. برای انجام این عملیات از قسمت Hardware configuration نرم افزار استفاده کنید روند انجام این کار برای یک PLC سری 300 در زیر نشان داده شده است روی ایستگاه مورد نظر کلیک کنید تا در صفحه سمت چپ گزینه Hardware ظاهر شود. به شکل زیر دقت کنید



با دابل کلیک بر روی گزینه مشخص شده وارد قسمت پیکربندی سخت افزار می شویم و پنجره زیر باز می گردد در سمت راست این صفحه می توانید تمام محصولات مرتبط با PLC را ببینید در فضای خالی سمت چپ جائیست که باید پیکربندی PLC را انجام دهید. در ابتدا نیاز به رک (ریل) دارید تا قطعات مورد نظر را بر روی آن قرار دهید. از ستون سمت راست گزینه SIMATIC 300 را انتخاب نموده و در پوشه RACK 300 گزینه RAIL مورد نظر را به صفحه سمت چپ اضافه کنید

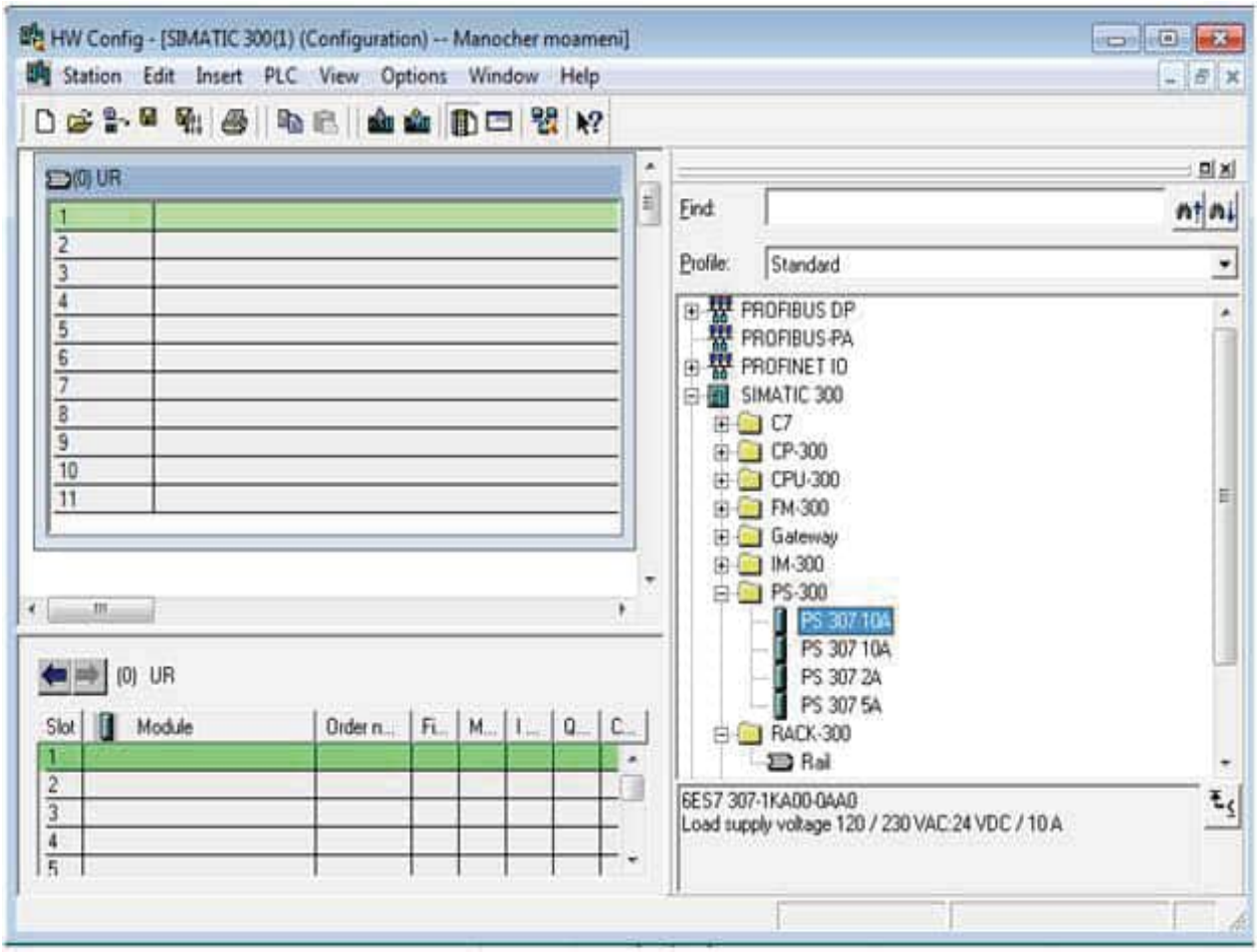

HW Config - [SIMATIC 300(1) (Configuration) -- Manocher moameni]
Station Edit Insert PLC View Options Window Help
محل انجام مراحل پیکر بندی سخت افزار
Find:
Profile: Standard
PROFIBUS DP
PROFIBUS-PA
PROFINET IO
SIMATIC 300
SIMATIC 400
SIMATIC PC Based Control 300/400
SIMATIC PC Station
محصولات مورد نیاز جهت
پیکر بندی سخت افزار
SIMATIC 300(1)
Slot
Designation
محل نمایش جزئیات هر قطعه
PROFIBUS-DP slaves for SIMATIC S7, M7, and C7 (distributed rack)
Press F1 to get Help.



همان طور که در شکل بالا دیده می شود با انتخاب ریل ، شماره سفارش مربوط به آن در زیر صفحه سمت راست آورده شده است. در سمت چپ نیمه پائینی اطلاعات تکمیلی اجزای ریل بالا را نشان می دهد

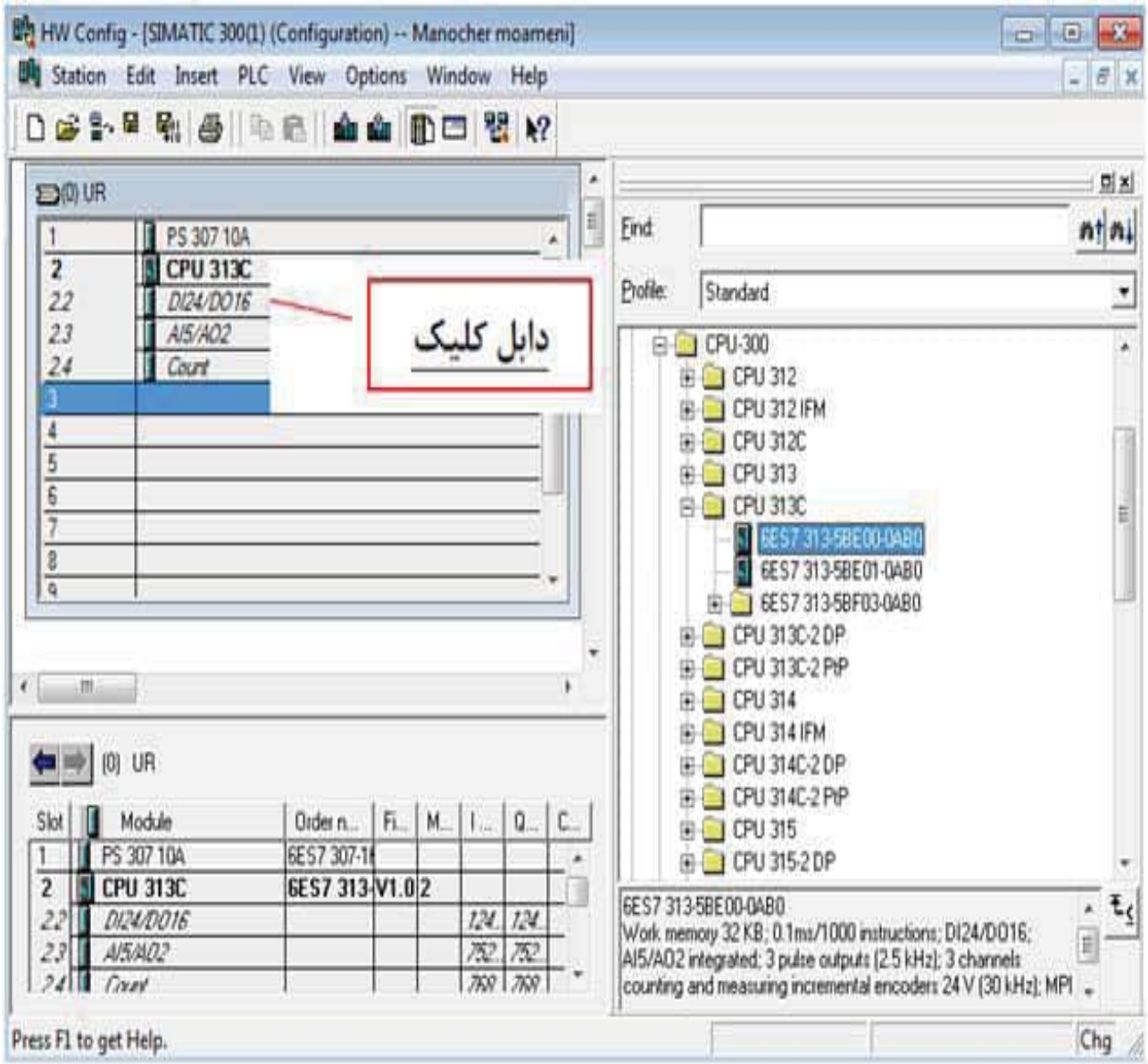
در مرحله بعد باید منبع تغذیه را بر روی ریل قرار دهیم برای این هدف از گزینه PS-300 در لیست سمت راست منبع تغذیه را مطابق با سخت افزار بیرونی خودمان انتخاب می کنیم دقت کنید که اطلاعات قید شده روی سخت افزار با تنظیمات نرم افزار یکی باشد انتخاب منبع تغذیه در شکل زیر نشان داده شده است

همان طور که دیده می شود با انتخاب منبع تغذیه اسلات شماره یک رک به رنگ سبز در می آید این بدین معنی است که کاربر فقط اجازه اضافه کردن منبع تغذیه به اسلات شماره یک را دارد. با دابل کلیک بر روی منبع تغذیه مورد نظر، قطعه به رک اضافه می گردد

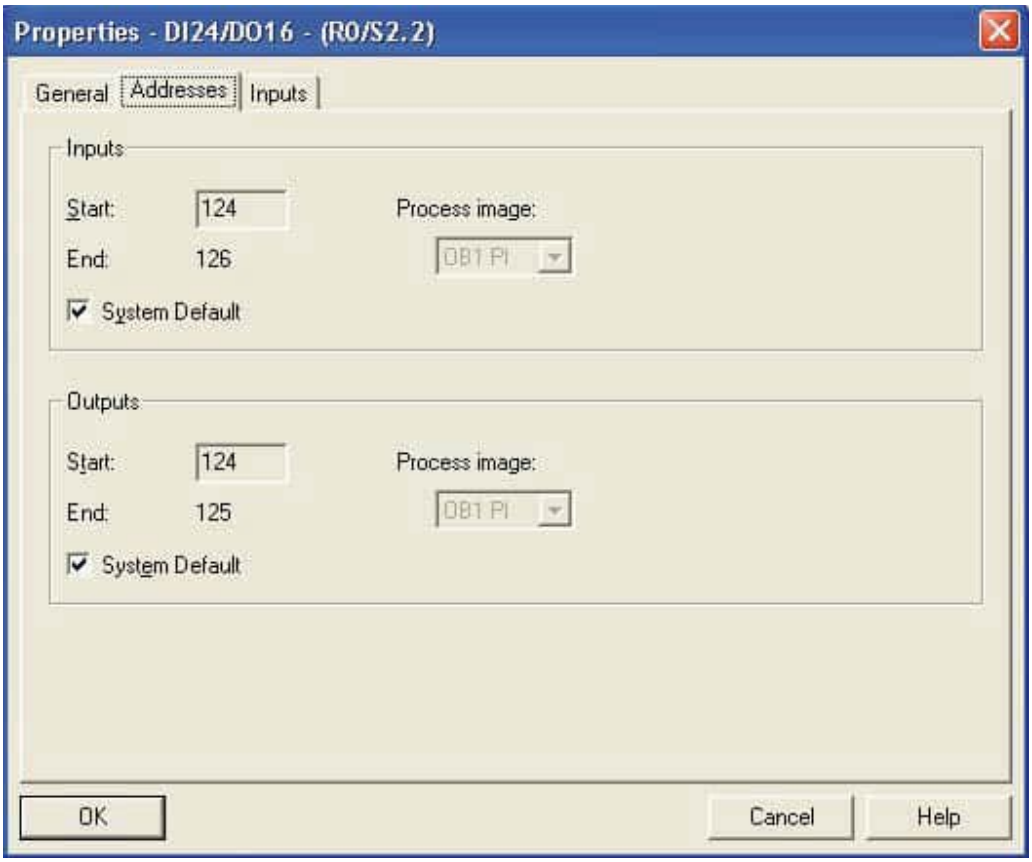


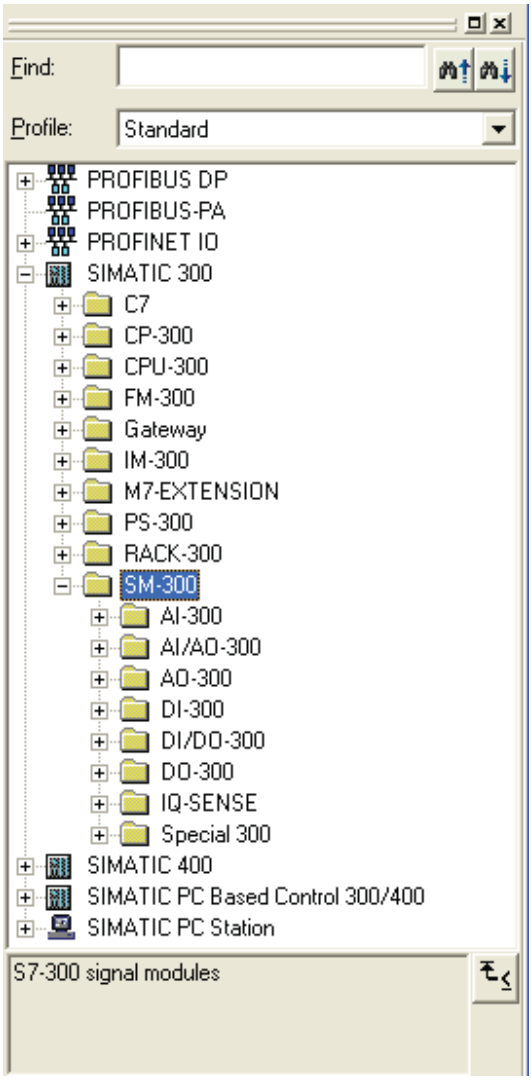
گام بعدی انتخاب نوع CPU است و قرار دادن آن در رک است از گزینه CPU-300 با توجه به مدل CPU قطعه مورد نظر را انتخاب می کنید. دقت شود که CPU فقط در اسلات شماره ۲ قرار می گیرد در شکل زیر یک CPU-C که از نوع کامپکت می باشد قرار داده شده است در نیمه پائینی سمت راست جزئیات مربوط به کارت مانند تعداد ورودی و خروجی ها ، آنالوگ یا دیجیتال بودن آن و حداکثر تعداد کانتر ها نشان داده شده است.

البته کاربر می تواند با دابل کلیک بر روی کارت مورد نظر ، از لبه Addresses آدرس شروع مورد نظر خود را با برداشتن تیک مربوط به پیش فرض سیستم ، وارد کند البته تنظیمات مربوط به هر گروه از کارت ها با هم متفاوت است که در فصل های بعدی هر یک به طور کامل توضیح داده خواهد شد



صفحه مربوط به تنظیمات آدرس کارت ها ، که با دابل کلیک بر روی کارت مورد نظر باز می شود





برای اضافه کردن کارت های ورودی و خروجی از مسیر SM-300 استفاده کنید و کارت مورد نظر خود را انتخاب نمائید اگر بر روی کارت انتخاب شده یک بار کلیک کنید نرم افزار به شما نشان خواهد داد که شما می توانید این کارت را چه خانه های قرار دهید در این حالت کارت را انتخاب کرده و با کشیدن آن در شماره اسلات مورد نظر قرار دهید دقت کنید در پایان کار نباید بین هیچ دو اسلاتی ،اسلات خالی وجود داشته باشد مگر در مواردی که اسلات برای کارت خاصی توسط خود نرم افزار رزور شده است مانند اسلات شماره 3 که برای IM ها رزرو شده است

توضیحات مربوط به اضافه کردن قطعات مربوط به شبکه PROFIBUS DP در فصل مربوطه به شبکه آمده است.

پس از پایان کار و اضافه کردن تمام سخت افزارهای موجود با انتخاب گزینه save and compile ، صفحه موجود توسط نرم افزار هم بررسی شده و هم ذخیره می شود اگر در پیکر بندی سخت افزار مشکلی وجود داشته باشد در این حالت به کاربر اخطار می دهد. پس از انجام مراحل فوق باید سخت افزار تعریف شده را به PLC دانلود کنیم (انتقال دهیم) که این کار با استفاده از انتخاب این گزینه انجام می شود در ضمن می توانید از منوی PLC گزینه Download را نیز انتخاب کنید

*** دقت شود قبل از دانلود کردن حتماً از اتصال کابل کامپیوتر با PLC اطمینان حاصل کنید یا اگر فقط هدف شبیه سازی است حتماً پنجره شبیه سازی باز باشد در غیر این صورت نرم افزار اخطار می دهد و عمل دانلود را انجام نمی دهد

فصل سوم

برنامه نویسی

در این فصل آشنا خواهید شد با

۱- آشنایی با انواع بلوک های موجود در نرم افزار

۲- اصول اولیه برنامه نویسی در plc های سری 300

۳- آشنایی با توابع و بلوک های موجود در نرم افزار plc های سری 300

در PLC به دو روش کلی می توان برنامه نویسی نمود:

۱- روش برنامه نویسی خطی

۲- روش برنامه نویسی ساختار یافته

در روش اول کاربر ، برنامه مورد نظر خود را به طور کامل در محیطی که برنامه در آنجا نوشته و اجرا می شود(OB1) وارد می کند و با RUN کردن آن ، نتایج خود را مشاهده می نماید. این روش برای انجام پروژه های کوچک با تعداد متغییر ورودی و خروجی کم مناسب است اما اگر تعداد متغییرها بسیار زیاد گردد و انجام کار کنترلی شامل چند مرحله پیچیده شود این روش برنامه نویسی دیگر جوابگو نخواهد بود و در صورت بروز مشکل، عیب یابی آن زمان بر و پر هزینه خواهد بود. به همین دلیل از روش برنامه نویسی ساختار یافته استفاده می کنند.

در این روش ابتدا پروژه اصلی را به چند زیر مجموعه تقسیم بندی می کنند و برای هر یک توابعی ایجاد می کنند و در مرحله نهایی با فراخوانی توابع، برنامه اصلی را پیکر بندی می کنند و در صورت بروز مشکل ، تابع دارای مشکل را پیدا کرده و فقط روی آن متمرکز می شوند.

انواع بلوک های برنامه:

برای انجام کار های مختلف بلوک هایی در نظر گرفته شده است که در زیر به آنها اشاره می شود:

بلوک های سازماندهی OB (organization Blocks):

این بلوک ها همان طور که از اسمشان مشخص است برای سازماندهی و اجرای برنامه های نوشته شده به کار می روند. مهمترین وظیفه این بلوک ها ایجاد ارتباط بین سخت افزار و نرم افزار است. این OB ها با استفاده از شماره ای که به آنها داده می شود مشخص می گردند. برای مثال OB1 جایی است که برنامه اصلی نوشته می شود و با RUN کردن PLC ، CPU دستورات این OB را خط به خط اجرا می نماید. مابقی OB ها هم وظایف خاص خود را دارند که در مباحث بعدی با آن آشنا خواهید شد.

توابع FC (Functions)

FC ها بلوک های منطقی هستند که می توان با استفاده از دستورات منطقی برنامه های حجیم یا پرکاربرد را در آن نوشته و به تعداد مورد نیاز ورودی و خروجی برای آن در نظر گرفت و سپس در صورت نیاز به FC ، آن را در OB1 فراخوانی کرد و از آن استفاده نمود. البته می توان درون خود FC نیز یک FC دیگر را نیز فراخوانی کنیم. فقط باید دقت شود که FC ها فاقد حافظه می باشند یعنی با اتمام کار FC همه متغییر های موقت آن از بین می روند. برای رفع این مشکل باید از FB ها استفاده کنیم. نحوه کار با FC ها در فصل های بعدی توضیح داده خواهد شد.

بلوک های توابع FB (Function blocks):

عملکرد این گروه مشابه FC هاست با این تفاوت که دارای حافظه می باشند و وقتی شما یک FB جدید ایجاد می کنید باید یک Data Block اختصاصی هم ایجاد کنید تا مقادیر و وضعیت متغییرها در آن ذخیره گردد که البته این کار را خود نرم افزار با اجازه کاربر انجام می دهد. با این گروه نیز به طور کامل آشنا خواهید شد.

بلوک های داده DB (Data Blocks)

این دسته از بلوک ها بر خلاف گروه های دیگر شامل دستورات منطقی نیستند و فقط برای ذخیره سازی داده ها به کار می روند با استفاده از DB ها می توان برای متغییرها در لحظه راه اندازی مقدار اولیه یا حالت اولیه دلخواه تعیین کرد و در صورت تمایل می توان اطلاعات فعلی متغییر را مشاهده کرد. DB ها به دو دسته تقسیم می شوند که در زیر اشاره می گردد:

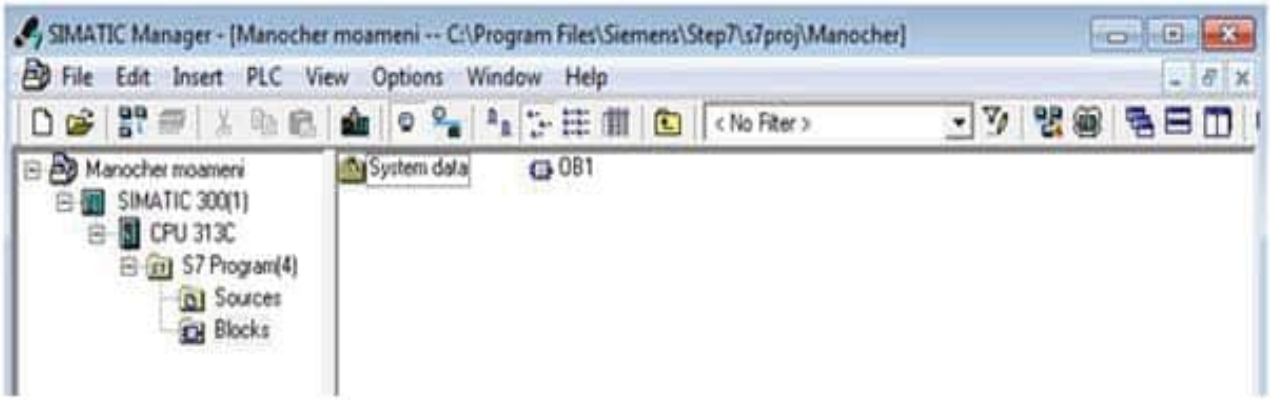
۱- **بلوک های داده اشتراکی**(shared Data Blocks): این بلوک ها از نوع اشتراکی هستند یعنی از سایر بلوک ها می توان به آن دسترسی داشت و اطلاعات آن را خواند یا تغییر داد.

۲- **بلوک های داده اختصاصی**(Instant Data Blocks) : این بلوک ها فقط برای ذخیره اطلاعات مربوط به FB ها به کار می روند. دقت شود که چندین بلوک داده اختصاصی را می توان به یک FB اختصاص داد ولی عکس این قضیه اتفاق نمی افتد یعنی نمی توان به چندین FB یک DB اختصاصی نسبت داد.

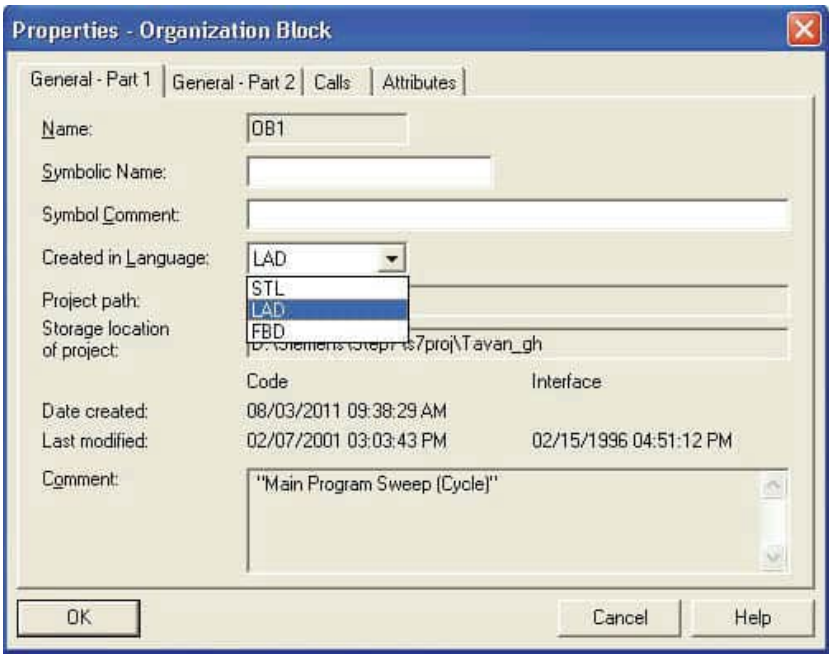
نحوه عملکرد و کار با DB ها در فصل های آینده به طور کامل توضیح داده خواهد شد.

آغاز برنامه نویسی خطی

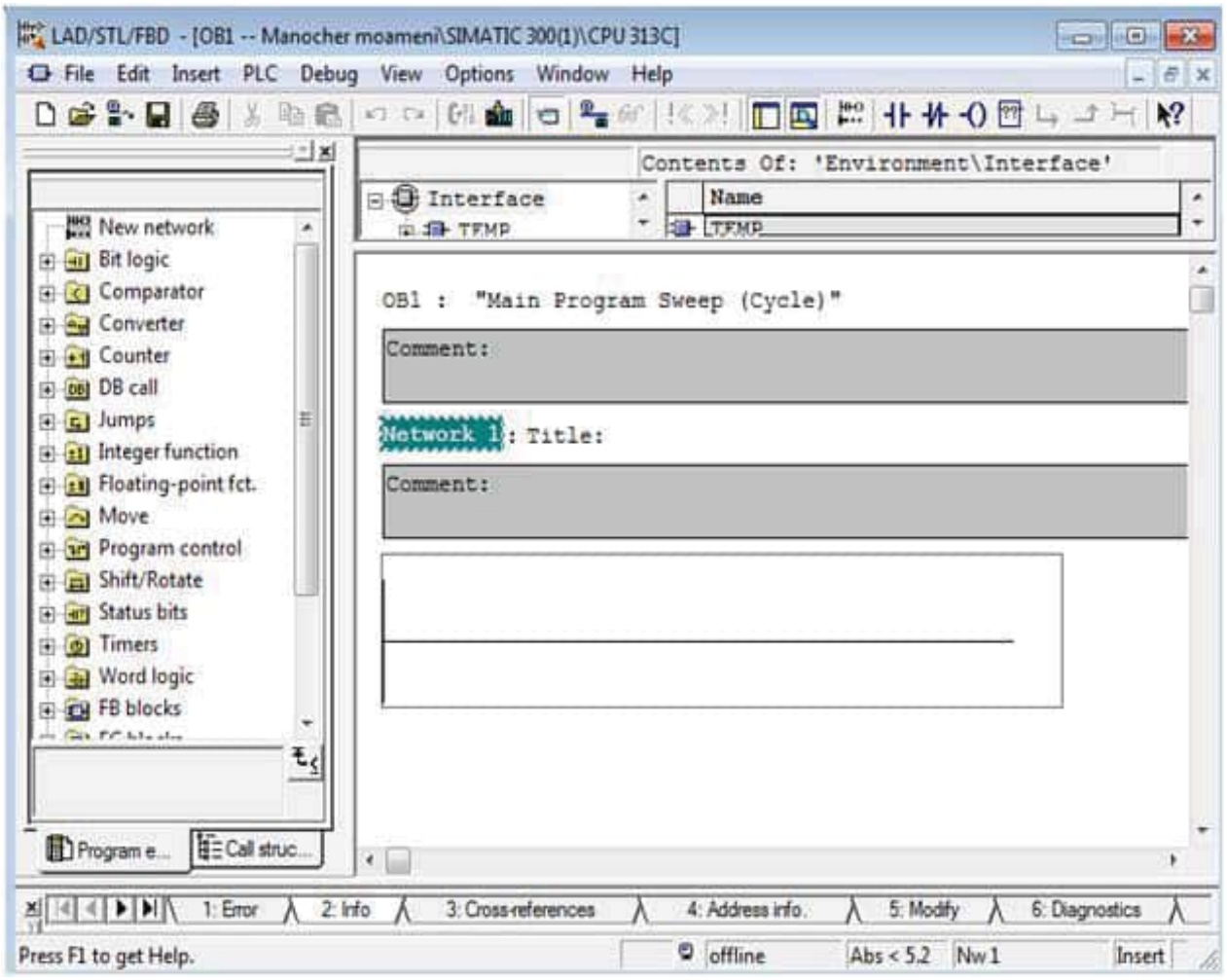
بعد از پیکر بندی سخت افزار پوشه های جدیدی به زیر نام پروژه شما اضافه می شود که نشان دهنده نوع CPU و نام پروژه های آن می باشد که در زیر نشان داده شده است.



با دابل کلیک برروی OB1 پنجره زیر باز می شود که می توان تنظیمات اولیه مانند زبان برنامه نویسی و نام مربوط به این برنامه را انتخاب کرد. با کلیک بر روی OK وارد محیط اصلی برنامه نویسی می شوید. به طور پیش فرض زبان نردبانی (LADDER) انتخاب شده است که می توان به دلخواه آن را تغییر داد. زبان های موجود علاوه بر LAD ، زبان FBD و STL می باشد.



با تائید صفحه فوق ، صفحه برنامه نویسی باز شده و کاربر می تواند برنامه دلخواه خود را با توجه به زبان مورد نظر خود وارد نماید. عملگرهای منطقی ، تایمر و کانترها، عملگر های ریاضی و مابقی توابع همگی در سمت چپ صفحه قابل مشاهده هستند که با کلیک بر روی علامت (+) کنار هر پوشه می توان به بلوک های داخل آن دسترسی پیدا کرد. عمل نوشتن برنامه در محلی به نام Network صورت می گیرد که در وسط صفحه قابل مشاهده است. هر Network حداکثر می تواند یک خروجی داشته باشد و برای داشتن یک خروجی دیگر باید یک Network جدید ایجاد نمائید.



برای ایجاد Network جدید می توانید بر روی آیکون مربوط به آن در بالای صفحه کلیک کرده و یا در صفحه راست کلیک نمائید و گزینه Insert New Network را انتخاب کنید و یا در لیست سمت چپ بر روی اولین گزینه (New Network) دابل کلیک نمائید.

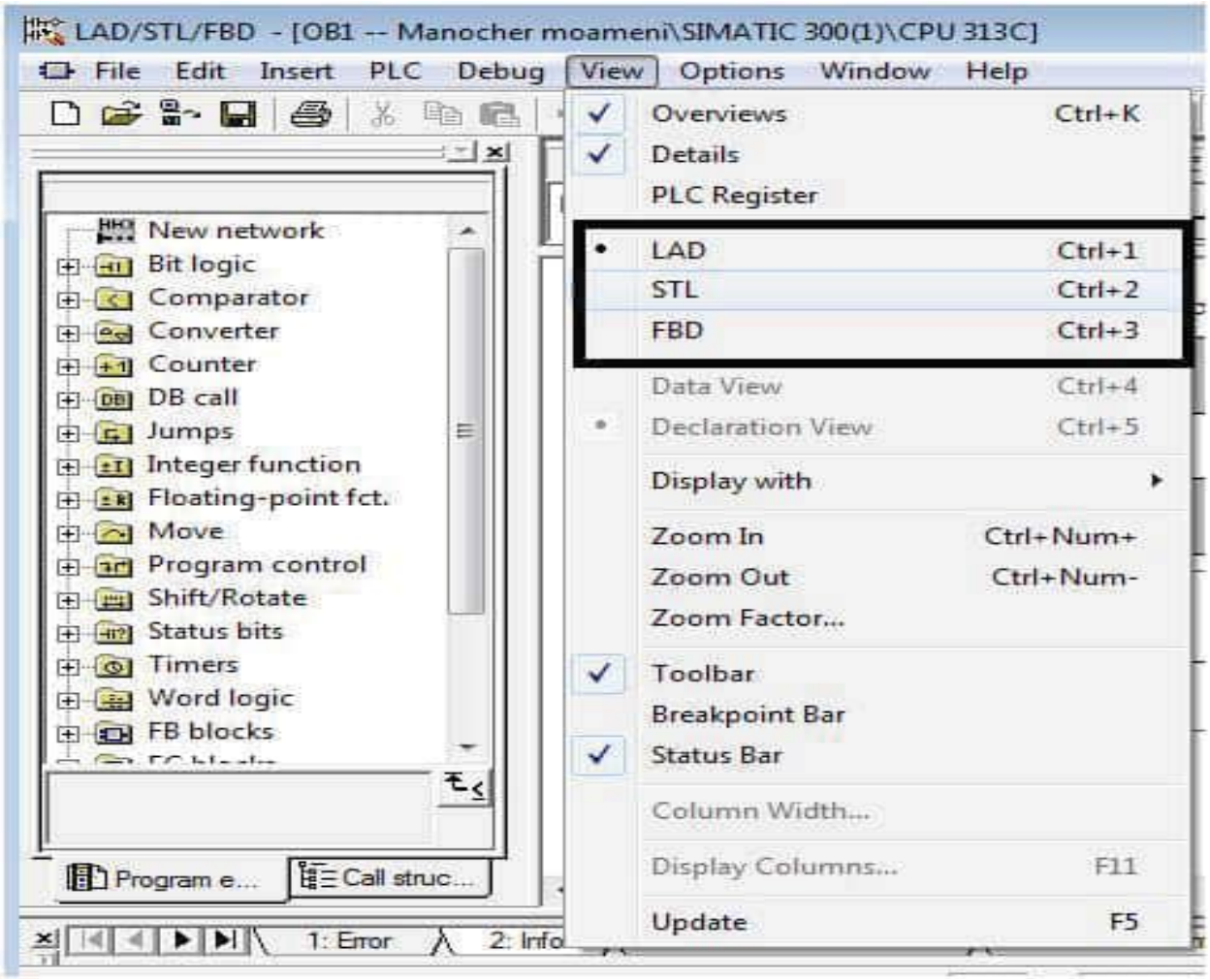
آشنایی با توابع :

در این قسمت نیز توابع مهم و پرکاربرد در ابتدا معرفی می شود و توابع با کاربرد کمتر در ادامه توضیح داده خواهد شد.

زبان های برنامه نویسی در PLC قابل تبدیل به یکدیگر هستند. برای انجام این کار کافیست در محیط برنامه نویسی از گزینه View زبان مورد نظر خود را انتخاب کنید. در شکل زیر مسیر تغییر زبان نشان داده شده است.

تمام برنامه ها از زبان های مختلف به هم تبدیل می شوند

ولی امکان دارد برخی از برنامه های نوشته شده با زبان STL به دو زبان دیگر تبدیل نشو .



✓ **پوشه**Bit LOGIL: همان طور که از اسم پوشه مشخص است عملگر هایی در این پوشه قرار دارند که عملیات منطقی روی بیت ها را انجام می دهند برای مثال عملگرهای AND , OR , XOR , NOT , SET , RESET و ... که گاهی قبلی از نحوه عملکرد آنها از الزامات کار با PLC ها است.

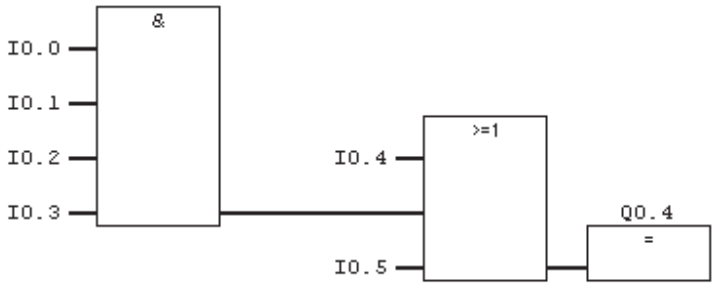
در این پوشه به توضیح برخی از موارد پر کاربرد و شاید هم نا آشنا پرداخته می شود.

- **اضافه کر ن پایه ورودی**: |--| این گزینه فقط در زبان FBD وجود دارد. بلوک هایی که به طور- پیش فرض قرار داده شده اند همگی دارای 2 ورودی می باشند.در روند برنامه نویسی کاربر ممکن است که نیاز به 3 یا 4 ورودی برای یک بلوک باشد با انتخاب بلوک مورد نظر و دابل کلیک بر روی این علامت در لیست سمت چپ تعداد ورودی بلوک مورد نظر افزایش می یابد.

** دقت کنید که بلوک باید قابلیت افزایش پایه را داشته باشد برای مثال این کار برای بلوک RS بی فایده است چون این بلوک فقط دو ورودی دارد.

مثال : برنامه ای بنویسید که 4 ورودی I0.0, I0.1 , I0.2, I0.3 را با هم AND نماید و خروجی را با OR I0.4,I0.5 نماید و حاصل را در Q0.4 قرار دهد.

پاسخ: همان طور که می بینید با استفاده از بلوک بالا تعداد ورودی های بلوک را افزایش پیدا کرده است.



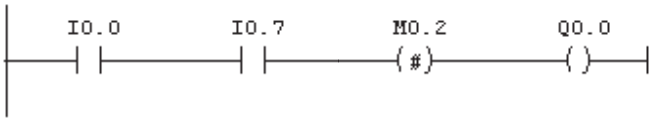
نکته: اگر پایه ای نیاز نبود می توان آنرا حذف کرد برای این کار نیز کافیست برروی پایه مورد نظر کلیک کرده و دکمه Delete را فشار دهید.

- **منفی کردن:** -o| با انتخاب پایه مورد نظر و دابل کلیک بر روی این علامت می توان سیگنال مورد نظر را NOT نمود. البته برای دسترسی آسانتر یک میانبر هم در بالای صفحه قرار داده شده است.
- **خروجی میانی** : -#- (Midline Output) این خروجی بر خلاف خروجی های دیگر که فقط باید در انتهای مسیر قرار بگیرند می تواند در میانه مسیر نیز قرار بگیرد.آدرسی که به این تابع می توان اطلاق کرد می تواند از نوع خروجی یا فضای حافظه باشد.

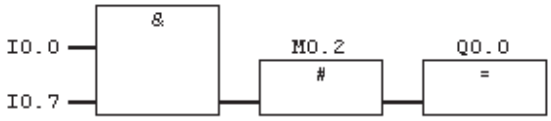
مثال: برنامه ای بنویسید که با تحریک کلید I0.0 و کلید I0.7 خروجی Q0.1 روشن شود و بیت حافظه ای به آدرس M0.2 نیز ست شود.

حل: پاسخ این سوال به دو زبان LAD و FBD نشان داده شده است.

زبان LAD



زبان FBD

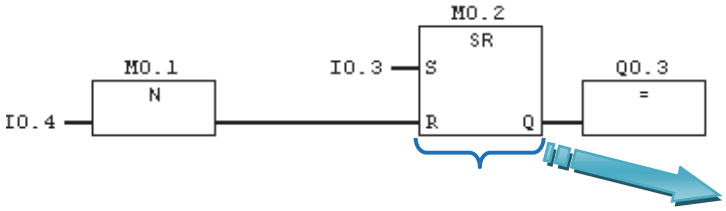


- **تشخیص دهنده های لبه:** --[N]- --[P]- این بلوک ها دارای یک ورودی و یک خروجی می باشند که نسبت به تغییر لبه واکنش نشان می دهند. بلوکی که با نام [P] مشخص شده است نسبت به لبه بالا رونده و بلوک [N] نسبت به لبه پائین رونده حساس بوده و در صورت مشاهده لبه های مذکور در خروجی این بلوک یک پالس به مدت سیکل زمانی CPU مشاهده می شود. عموماً این سیکل زمانی برابر با 150 ms می باشد.

آدرسی که به این بلوک اختصاص داده می شود می تواند از نوع خروجی ، فضای حافظه یا حتی ورودی نیز باشد. به مثال زیر دقت نمائید.

مثال : برنامه ای بنویسید که با فشردن شستی I0.3 خروجی Q0.3 نیز روشن شود و زمانی که شستی I0.4 فشرده می شود خروجی خاموش نشده و زمانی که دست از روی شستی برداشته شده است خروجی خاموش گردد.

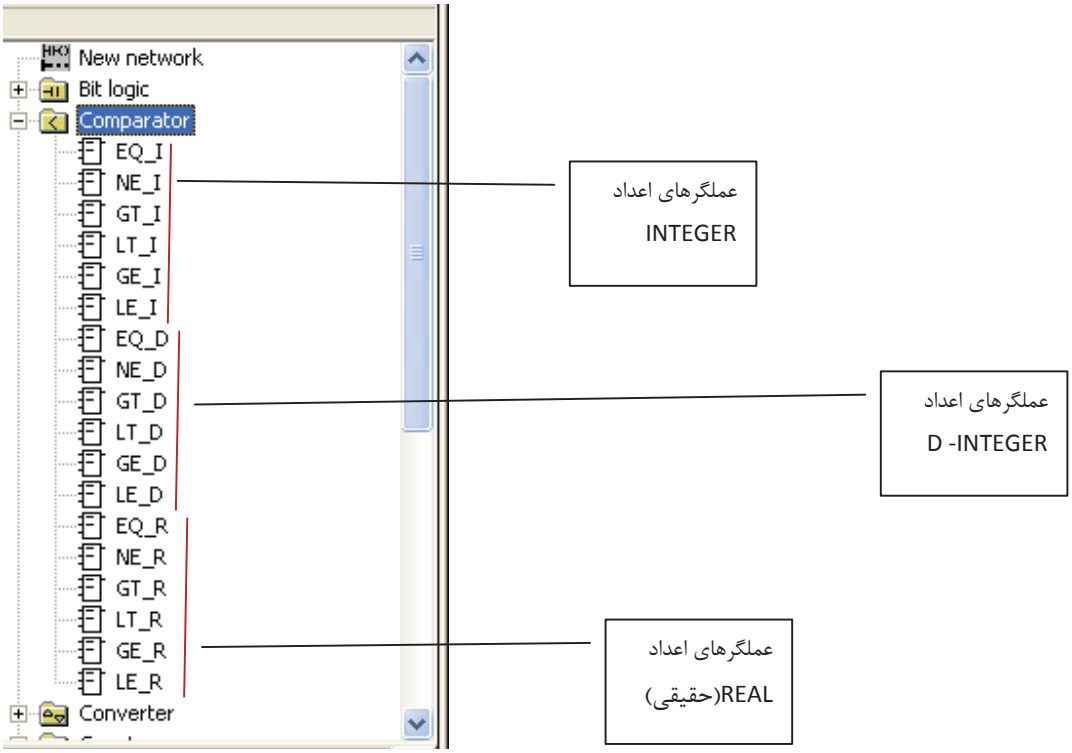
پاسخ: همان طور که در شکل دیده می شود زمانیکه کلید I0.3 تحریک می گردد خروجی روشن می شود ولی اگر کلید I0.4 را فشار دهید اتفاقی نمی افتد چون حساس به لبه پائین رونده می باشد.



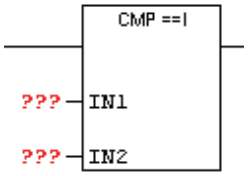
بلوک های SR و RS: این بلوک ها از پر کاربردترین بلوک ها در زبان برنامه نویسی می باشند. همانطور که می دانید با فعال شدن لحظه ای پایه S خروجی این بلوک فعال می شود و با فعال شدن لحظه ای پایه R خروجی ریست می گردد. عملکرد این دو بلوک از این نظر کاملاً شبیه به هم می باشد ولی در یک حالت با هم متفاوت می باشند. زمانی که در هر دو پایه S و R به طور همزمان فعال باشند در بلوک SR اولویت با پایه R می باشد و بر عکس در بلوک RS در صورت فعال شدن همزمان، اولویت با پایه S می باشد.

- ✓ **پوشه مقایسه کننده ها(comparator):**

در این پوشه عملگرهای مقایسه کننده گنجانده شده است. همان طور که می بینید این عملگرها برای ۳ گروه از اعداد INT ، DINT و REAL متمایز شده اند. در اینجا به توضیح عملکرد یک گروه از اعداد می پردازیم و بقیه گروه ها نیز از همین قاعده پیروی می کنند.

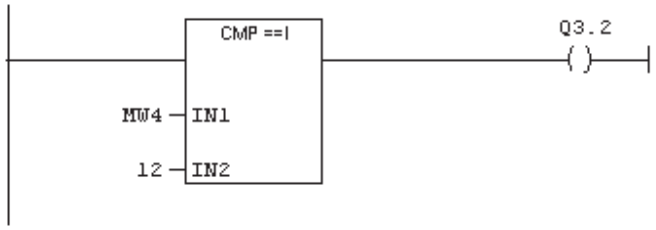


بلوک تساوی EQ_I : ، همان طور که می بینید این بلوک دارای دو ورودی از نوع عدد صحیح می باشد که هر گاه این دو عدد با هم برابر شوند خروجی این بلوک فعال می گردد. علامت سوال قرمز رنگ در ورودی این بلوک نشانگر این است که باید آدرس عدد صحیح یا خود عدد صحیح را وارد نمایید.

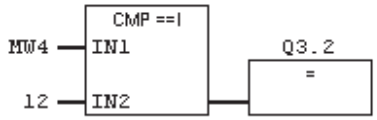


مثال : هرگاه مقدار MW4 برابر با ۱۲ شد خروجی Q 3.2 فعال شود.

نکته: در زبان LAD می توان برای این بلوک فعال ساز نیز قرار داد که با فعال شدن آن ، بلوک عملیات مقایسه را انجام دهد.



زبان FBD:



زبان STL:

```
L     MW     4
L     12
==I
=     Q      3.2
```

در ادامه به جهت راحتی در یادگیری، مثال ها را به زبان LAD یا FBD نشان خواهیم داد.

NE_I **بلوک نامساوی:** هرگاه دو ورودی برابر با هم نباشند خروجی فعال است و اگر برابر شوند خروجی غیر فعال می گردد.

GT_I **بلوک بزرگتر:** این بلوک دارای دو ورودی می باشد هرگاه ورودی IN 1 از IN2 بزرگتر شود خروجی آن فعال می شود.

LT_I **بلوک کوچکتر:** این بلوک نیز دارای دو ورودی می باشد که هرگاه ورودی IN 1 از ورودی IN 2 کوچکتر بشود خروجی آن فعال می گردد.

GE_I **بلوک بزرگتر مساوی**: این بلوک مانند بلوک GT_I عمل می نماید با این تفاوت که در حالتی که دو عدد مساوی هستند هم خروجی آن فعال باقی می ماند.

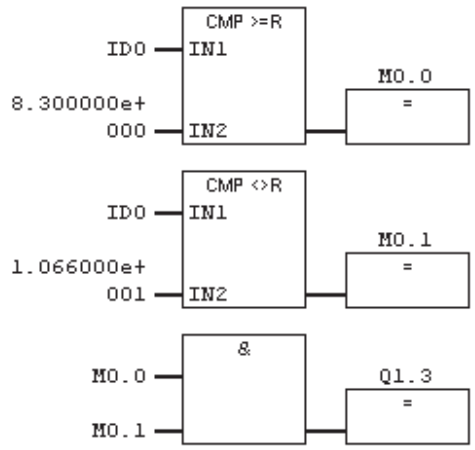
LE_I **بلوک کوچکتر مساوی:** عملکرد این بلوک تقریباً مشابه بلوک LT_I می باشد با این تفاوت که در حالت تساوی دو ورودی هم خروجی آن فعال باقی می ماند.

سایر بلوک ها در این پوشه نیز مشابه هستند. در حالتی که از بلوک های DINT استفاده می کنید فرمت باید به شکل (عدد مورد نظر#L) باشد. برای اعداد REAL نیز باید عدد وارده دارای رقم اعشار باشد. مثلاً برای وارد کردن عدد ۱۰ در فرمت DINT داریم L#10 و با فرمت REAL داریم 10.0

دقت کنید که هر دو ورودی بلوک باید از یک جنس باشند.

مثال : هرگاه ورودی بزرگتر مساوی8.3 شد خروجیQ1.3 روشن گردد ولی اگر برابر با 10.66 شد خروجی Q1.3 خاموش شود.

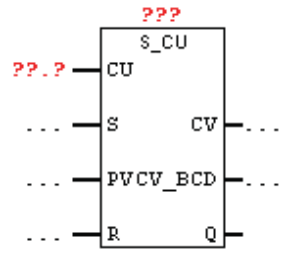
پاسخ: همان طور که در شکل دیده می شود چون اعداد از نوع اعشاری هستند از مقایسه کننده های REAL استفاده شده است. از یک بلوک بزرگ تر مساوی و یک بلوک نامساوی استفاده شده است.



✓ **پوشه شمارنده ها**(counter)**:**

این پوشه جهت انجام عملیات شمارش به کار می رود بلوک های این پوشه به سه دسته تقسیم بندی می شوند: شمارنده های بالارونده، شمارنده های پائین رونده، شمارنده های بالا رونده و پائین رونده که در ادامه هریک را توضیح خواهیم داد.

- **شمارنده بالا رونده(S_CU)**: این بلوک دارای ۴ پایه ورودی و ۳ پایه خروجی می باشد.لازم به ذکر است



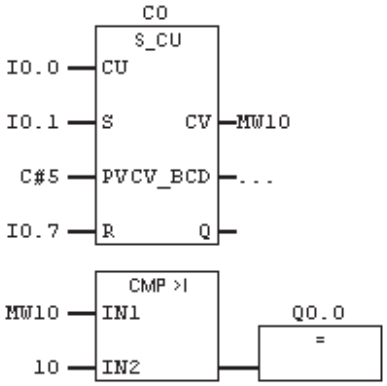
برای استفاده از این بلوک نیازی نیست که همه پایه های آن را متصل نمائید، فقط نامگذاری پایه هایی که دارای علامت سوال با رنگ قرمز می باشند **الزامی** است.

در محل بالای بلوک باید شماره کانتر را ذکر کنید برای مثال C0,C1,... حداکثر تعداد کانتر بستگی به نوع CPU دارد و در مدل های مختلف متفاوت است. نحوه عملکرد این کانتر بدین صورت است که با اعمال لبه بالارونده در پایه CU خروجی کانتر یک واحد افزایش می یابد و در لبه پائین رونده عکس العملی از خود نشان نمی دهد. حداکثر مقداری که کانتر می تواند بشمارد 999 است. هرگاه پایه ورودی S فعال شود مقدار موجود در پایه PV به عنوان مبدا شمارش کانتر قرار می گیرد و با اعمال سیگنال به پایه CU کانتر از آن عدد شروع به شمارش می کند. برای وارد کردن این عدد از فرمت رو به رو استفاده می گردد (عدد مورد نظر#C)

پایه R نیز برای ریست کردن کانتر به کار می رود با اعمال سیگنال به این پایه مقدار کانتر صفر شده و تا زمانی که این پایه فعال باشد کانتر عمل شمارش را انجام نمی دهد. پایه Q خروجی کانتر است و زمانی که کانتر مقداری غیر صفر شود این خروجی فعال می شود. پایه CV مقدار فعلی کانتر را نشان می دهد و می توان عدد موجود در آن را در یک MW ذخیره و مشاهده نمود و در صورت نیاز از آن استفاده کرد. CV_BCD مقدار خروجی کانتر را در قالب BCD نشان می دهد.

مثال : با عبور جسم از مقابل سنسور نوری در خروجی سنسور یک پالس دیده می شود برنامه ای بنویسید که عمل شمارش را انجام دهد و وقتی خروجی کانتر برابر با 10 شد لامپ متصل به خروجی Q 0.0 را روشن نماید تا کاربر بتواند با تحریک کلید I 0.7 کانتر را ریست کند سنسور نوری نیز به ورودی I 0.0 متصل است. با فشردن کلید I0.1 مقدار کانتر برابر با ۵ شود.

پاسخ:



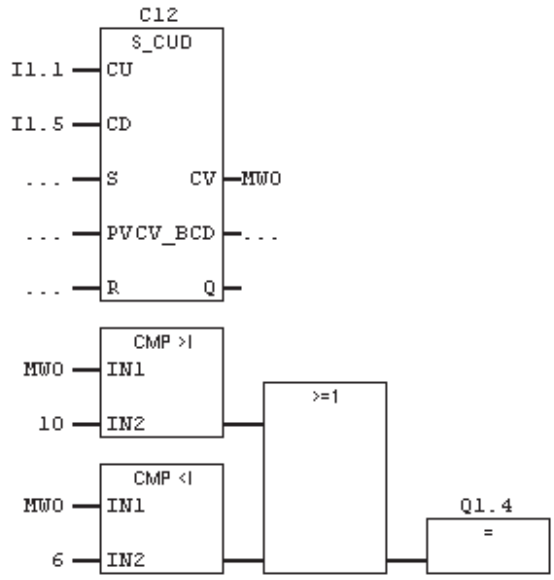
همان طور که دیده می شود ورودیI0.0 به ورودی کانتر وصل شده است خروجی نیز چون به فرم word می باشد در یک MW ذخیره شده و در مرحله بعد با 10 مقایسه می شود. خروجی Q0.0 نیز زمانی فعال می گردد که خروجی کانتر برابر با 10 گردد.

- **شمارنده پائین رونده** (S_CD): عملکرد این بلوک مشابه شمارنده بالارونده است با این تفاوت که به ازای هر لبه بالارونده در پایه CD شمارنده یک واحد کاهش یافته تا به عدد صفر برسد و با صفر شدن مقدار کانتر خروجی آن نیز غیر فعال می شود.
- **شمارنده بالارونده و پائین رونده**(S_CUD)**:** این بلوک تلفیقی از دو بلوک بالایی است و دارای دو پایه مجزا از هم می باشد که با تحریک هر یک کانتر به صورت بالا رونده و یا پائین رونده عمل می کند. اگر هر دو پایه همزمان فعال شوند هر دو عمل نموده و مقدار کانتر تغییر نخواهد کرد.

مثال: ورود و خروج افراد یک اتاق توسط دو سنسور تشخیص داده می شود برنامه ای بنویسید که اگر تعداد افراد اتاق بیشتر از 10 نفر یا کمتر از 6 نفر شد خروجی Q 1.4 فعال شود. سنسور تشخیص ورود به پایه I 1.1 و سنسور تشخیص خروج به پایه I 1.5 متصل شده است.

پاسخ:

در اینجا از کانتر شماره ۱۲ استفاده شده است و سنسور ورود به پایه شمارنده صعودی و سنسور خروج به پایه شمارنده نذولی کانتر متصل یافته. خروجی کانتر نیز در فضای حافظه WORD 0 قرار داده شده است و در مرحله بعد با مقایسه آن با اعداد مورد نظر خروجی آن بدست آمده است.



- **[SC],[CU],[CD]** : این توابع برای کاربردهای خاص و انجام کارها به صورت بیتی به کار می رود و خروجی های Q و CV آن حذف شده است. همان طور که از اسم دستورات مشخص است [SC] برای ست کردن کانتر در یک عدد مشخص به کار می رود و [CU] برای افزایش مقدار یک کانتر و [CD] برای کاهش مقدار یک کانتر کاربرد دارد.

✓ **پوشه توابع اعداد صحیح**(Integer Function): این پوشه برای انجام عملیات بر روی اعداد صحیح در نظر گرفته شده است. توابع این پوشه برای دو سری از اعداد INT و DINT در نظر گرفته شده است که عملکرد مشابه هم دارند.

- **جمع کننده(ADD_I) :** این تابع دارای دو ورودی عدد صحیح و یک ورودی دیجیتال و یک خروجی دیجیتال و یک خروجی صحیح می باشد که با فعال شدن پایه ورودی دیجیتال آن ، دو عدد ورودی را با هم جمع کرده و در خروجی قرار می دهد و پایه خروجی دیجیتال آن نیز فعال می گردد. در صورتی که حاصل جمع دو عدد ورودی از بازه تعریفی برای اعداد صحیح فراتر رود خروجی دیجیتال آن غیر فعال می گردد و این نشان می دهد که عمل جمع صورت نگرفته است.

- **تفریق کننده (SUB_I):** این تابع نیز همانند تابع جمع کننده عمل نموده با فعال شدن ورودی حاصل تفریق دو عدد را در خروجی قرار می دهد و اگر حاصل عملیات در بازه مشخص شده نباشد خروجی دیجیتال آن غیر فعال می گردد.
- **ضرب کننده(MUL_I):** این تابع نیز برای ضرب دو عدد صحیح به کار می رود.
- **تقسیم کننده(DIV_I):** این تابع با فعال شدن ورودی دیجیتال مربوطه IN1 را بر IN2 تقسیم نموده و خارج قسمت آن را در خروجی قرار می دهد. در صورت تقسیم عدد اول بر صفر خروجی دیجیتال آن غیر فعال می گردد.

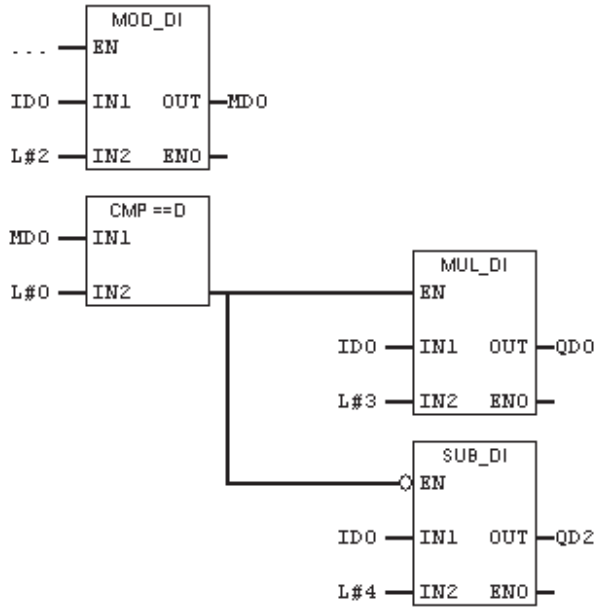
برای اعداد DINT نیز همین توابع با همین ویژگی ها در نظر گرفته شده است که از بازنویسی آن خودداری می کنیم فقط دقت شود که برای وارد کردن اعداد در توابع DINT باید از فرمت (عدد DINT مورد نظر #L) استفاده کنید.

- **باقی مانده (MOD_DI) :** این دستور برای محاسبه باقی مانده تقسیم دو عدد به کار می رود و ورودی های آن نیز از نوع DINT می باشد و در صورت تمایل به ذخیره کردن خروجی باید آن را در یک دابل WORD ذخیره کنید.

مثال: برنامه ای بنویسید که عددی را به صورت دابل INT از ورودی بخواند ، اگر عدد زوج است آن را در 3 ضرب کند و در خروجی QD0 نشان دهد و اگر عدد فرد است آن را با عدد 4 جمع کند و خروجی را در QD 2 نشان دهد. (راهنمایی: باقی مانده عدد زوج بر 2 صفر و باقی مانده عدد فرد 1 است)

پاسخ:

این برنامه ابتدا باقی مانده عدد مورد نظر بر دو را بدست می آورد و آن را با صفر مقایسه می کند. اگر باقی مانده صفر باشد یعنی عدد زوج است و بلوک ضرب کننده فعال می گردد و اگر عدد فرد باشد بلوک جمع کننده فعال می گردد. به فرمت استفاده شده برای عدد 2 دقت کنید. اعداد از نوع DINT می باشند.

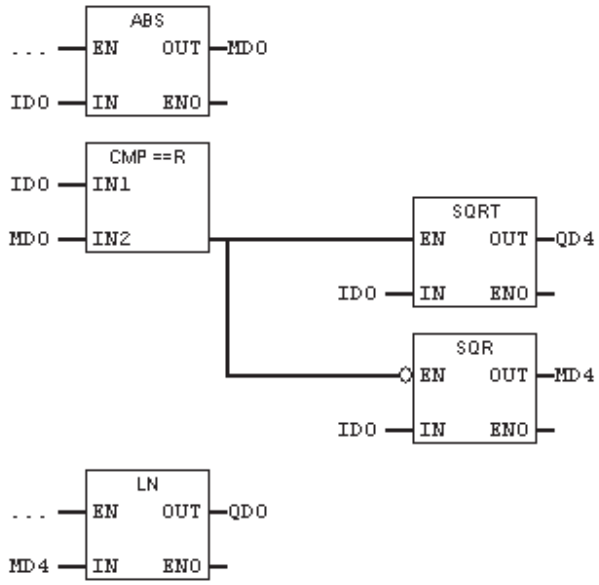


✓ **توابع اعداد حقیقی** (Floating point real) **:** این توابع برای کار کردن با اعداد حقیقی و اعشاری می باشد و نسبت به اعداد INT و DINT تعداد بیشتری را شامل می شوند.

- **توابع** ADD_R , SUB_R , MUL_R , DIV_R : عملکرد این توابع مشابه توابع قبلی است و برای جمع کردن ،تفریق ، ضرب کردن و تقسیم اعداد صحیح به کار می روند.
- **تابع اندازه یا قدر مطلق**(ABS)**:** از این تابع برای محاسبه قدر مطلق عدد ورودی استفاده می گردد.
- **تابع رادیکال**(SQRT)**:** این تابع ریشه دوم عدد موجود در ورودی را محاسبه می کند و اگر عدد زیر رادیکال منفی شود خروجی دیجیتال آن غیر فعال می شود و این یعنی این بلوک عمل نکرده است.
- **توان دوم**(SQR) : از این تابع برای محاسبه توان دوم یک عدد استفاده می گردد.
- **لگاریتم طبیعی**(LN)**:** از این تابع برای محاسبه لگاریتم طبیعی (لگاریتم در مبنای e) استفاده می گردد. اگر عدد وارده منفی باشد خروجی دیجیتال آن غیرفعال می گردد که نشان دهنده عدم کارکرد این بلوک است.
- **تابع نمایی**(EXP)**:** این بلوک برای محاسبه توابع نمایی (e^x) به کار می رود. مقدار تقریبی عدد e برابر با 2.7182می باشد.

- **توابع مثلثاتی:** sin , cos ,tan از جمله مهمترین توابع مثلثاتی هستند که بلوک آنها در دسترس است. ورودی این توابع زوایا بر حسب رادیان می باشد. برای تبدیل زاویه ای از درجه به رادیان کافیست آنرا بر 180 تقسیم و در عدد π ضرب نمائید.
- **توابع معکوس مثلثاتی:** توابع $\underline{Sin^{-1}}$ ، $\underline{Cos^{-1}}$ و $\underline{tan^{-1}}$ توابع معکوس مثلثاتی هستند که برای محاسبات ریاضی به کار می روند. خروجی این توابع بر حسب رادیان می باشد. در توابع آرک سینوس وآرک تانژانت خروجی زاویه ای بین $[-\pi/2, \pi/2]$ است و برای آرک کسینوس خروجی بین $[0, \pi]$ می باشد.

مثال: برنامه ای بنویسید که عددی را از طریق ورودی ID0 دریافت کند اگر عدد ورودی منفی است آن را به توان دو برساند و سپس لگاریتم طبیعی آن را محاسبه کند و درخروجی QD0 قرار دهد و اگر عدد مثبت است ریشه دوم آن را محاسبه نماید و در QD4 قرار دهد. (راهنمایی: قدر مطلق عدد مثبت با خودش برابر است).



پاسخ: این برنامه ابتدا قدر مطلق عدد را محاسبه می کند و جواب حاصل را با خودش مقایسه می نماید اگر با هم برابر بودند یعنی عدد مثبت است و عمل خواسته شده را روی آن انجام می دهد و اگر شرط تساوی برقرار نبود یعنی عدد منفی است محاسبات مربوط به آن اجرا می شود و جواب نهایی نیز به خروجی مربوطه منتقل می شوند.

- ✓ **پوشه انتقال(MOVE):** در این پوشه فقط یک تابع وجود دارد که از جمله توابع پرکاربرد است. تابع MOVE برای انتقال اطلاعات از مکانی به مکان دیگر به کار می رود. بوسیله این دستور می توان داده ای را از مکانی از حافظه یا ورودی، به مکان دیگر از حافظه انتقال داد. یا عددی را به طور دلخواه و دستی در مکانی از حافظه انتقال داد. در انتقال اعداد باید به فرمت آنها توجه کنید که عدد مورد نظر INT,DINT و یا REAL است. برای انتقال اعداد به مکانی از حافظه در قالب مبنای ۱۶ باید از فرمت های زیر استفاده کنیم.

انتقال یک بایت : B#16#00-----B#16#FF

انتقال یک word: w#16#0000 ------- w#16#FFFF

انتقال یک دابل word: w#16#0000 0000 -------- w#16#FFFF FFFF

انتقال اعداد: برای انتقال اعداد کافیست آنها را به طور مستقیم بنویسید فقط به نوع عدد و فرمت عدد دقت کنید.

- ✓ **دستورات پرش**

در روند اجرای برنامه کاربر می تواند با ایجاد شرایطی از محلی از برنامه به محل دیگر برود. این عمل توسط دستورات JUMP (پرش) انجام می گیرد.

[JUMP]: هرگاه سیگنال ورودی این بلوک یک شود یا همان RLO یک شود این بلوک فعال شده و ادامه برنامه از محلی که توسط این بلوک مشخص شده است ادامه می یابد.کاربر محل مورد نظر خود را توسط LABLE (برچسب) مشخص می کند.

[JUMPN]: این بلوک عکس بلوک [JUMP] عمل می نماید به این معنا که با غیر فعال شدن سیگنال ورودی یا RLO این بلوک فعال شده و ادامه برنامه از محل تعیین شده توسط این بلوک ادامه می یابد.

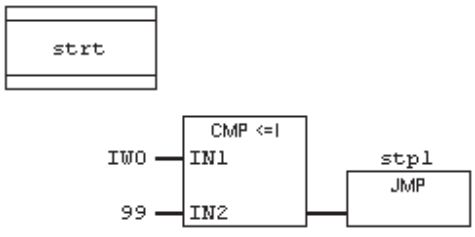
LABLE: این کلمه به معنی برچسب می باشد که کاربر آن را در NETWORK مورد نظر خود وارد می نماید و نامی دلخواه به آن اطلاق می کند و هنگامیکه دستور پرش فعال شود با توجه به برچسب دستور مورد نظر ، NETWORK

مربوطه فعال می شود. در انتخاب نام برچسب دقت شود که این نام حداکثر 4 کارکتر(حروف و اعداد) می تواند داشته باشد و اولین کارکتر آن نمی تواند از نوع عدد باشد. برای مثال: CAS1,mod5 ولی نام های 2nxt و یا loop4 نا معتبر می باشند.

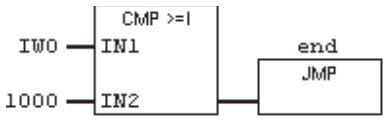
در استفاده از دستورات JUMP باید دقت نمود که عمل پرش هم می تواند به سمت جلو صورت گیرد و عم به سمت عقب. در پرش به جلو دستورات بین محل صدور فرمان و برچسب اجرا نخواهند شد ولی در پرش به عقب باید دقت نمود که عمل پرش به عقب در همان سیکل زمانی اجرای برنامه در حال اجرا می باشد. اگر تعداد دفعات پرش به عقب باعث شود که زمان اجرای برنامه از یک سیکل زمانی CPU بیشتر گردد آنگاه CPU خطای سیستمی(SF) داده و به حالت STOP می رود. برای مثال یک برنامه 10 خطی را در نظر بگیرید که در خط 6 آن شرط پرش به عقب (خط 3) وجود دارد با توجه به روند اجرای دستورات در CPU که قبلاً گفته شد، CPU در مدت 150 ms باید از خط اول به خط دهم برسد و اگر این دستور پرش مانع این عمل شود CPU خطا داده و به حالت STOP می رود. در برنامه هایی که از دستورات پرش به عقب استفاده می کنید به این نکته دقت کنید.

مثال: عددی را از ورودی Iw0 خوانده اگر این عدد دو رقمی است آن را در 2ضرب نماید و در خروجی QW0 نشان دهید و اگر عدد 3رقمی است آن را بر 7 تقسیم نموده و در خروجی مذکور نشان دهید و برای اعداد دیگر عدد صفر را در خروجی نشان دهید.

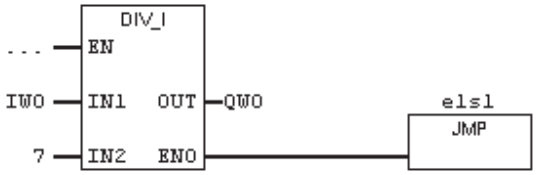
پاسخ: الگوریتم های مختلفی برای پاسخ به این مثال وجود دارد که در اینجا یکی از این برنامه های نوشته شده با دستورات JUMP نشان داده خواهد شد.



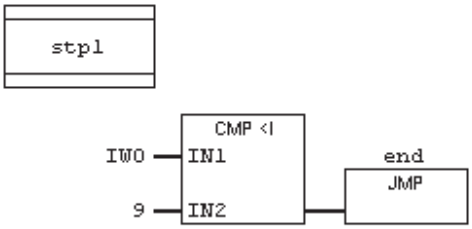
روند الگوریتم به این صورت است که عدد دریافتی از ورودی با عدد 99 مقایسه می شود اگر کوچکتر مساوی این عدد باشد پس یا دو رقمی است یا تک رقمی. برای بررسی آن باید به NETWORK با برچسب stp1 رفت در غیر این صورت NETWORK بعدی اجرا خواهد شد.



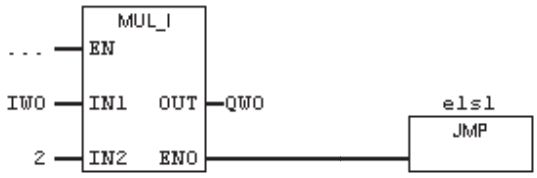
در این NETWORK عدد ورودی با مقدار 1000 مقایسه می شود اگر بزرگتر از آن باشد پس عدد 4 رقمی است و به برچسب end می پرد در غیر این صورت عدد مورد نظر 3 رقمی است و NETWORK بعدی اجرا خواهد شد و پس از پایان عملیات به برچسب els1 می پرد.

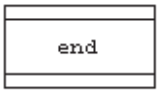


برچسب stp1 در این NETWORK آورده شده است که عدد کوچکتر از 99 را بررسی می نماید که آیا یک رقمی است یا دو رقمی.

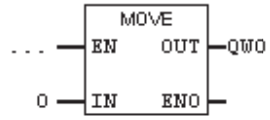


اگر عدد یک رقمی باشد به همان برچسبی می پرد که عدد 4 رقمی نیز به همان می پرد ولی اگر عدد 2 رقمی باشد Network بعدی اجرا خواهد شد و عدد در 2 ضرب می گردد ودر پایان نیز به بر چسب els1 می پرد.





Network شامل برچسب end، عملیات انتقال صفر در خروجی را انجام می دهد. این برچسب زمانی فعال می شود که عدد مورد نظر 4 رقمی یا تک رقمی تشخیص داده می شد.



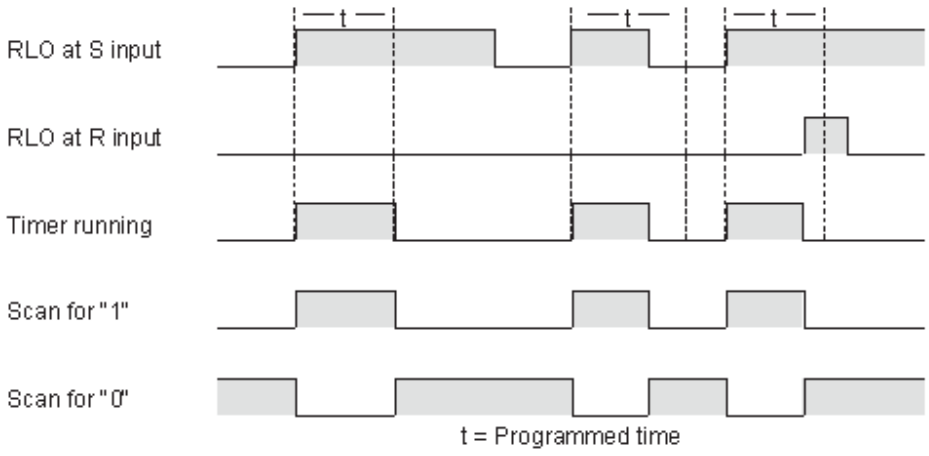
در Network بعد نیز برچسب els1 قرار داده شده است که به زبان stl در آن برنامه ای نوشته شده است به این مضمون که کاری انجام ندهد.(NOP:مخفف No Operation می باشد) در اینNetwork کاری انجام نمی شود و هدف از گنجاندن آن جلوگیری از برش به عقب می باشد. با ورود به این Network روند برنامه در همان سیکل زمانی مورد نظر به پایان می رسد و اسکن برنامه از ابتدا آغاز می شود.

```
elsl: NOP   0
```

✓ **تایمرها:** این پوشه شامل انواع مختلف تایمرها می باشد تعداد تایمرهایی که می توان استفاده نمود بسته به نوع CPU متفاوت است.

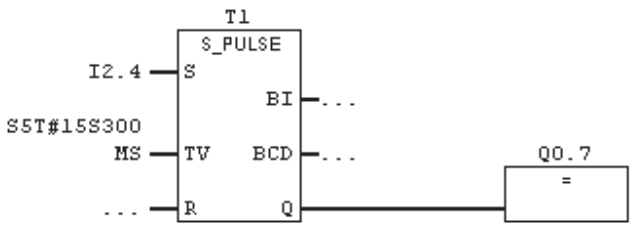
تایمر S_PULSE: این تایمر دارای سه ورودی و سه خروجی است. همان طور که از نمودار زمانی این تایمر مشخص است با فعال شدن پایه S به طور دائم تایمر شروع به زمان گیری می کند و به اندازه زمان تنظیمی در پایه TV (Time value) خروجی دیجیتال آن فعال است و پس از سپری شدن زمان، خروجی غیر فعال می گردد. اگر پایه S در وسط زمان گیری غیر فعال شود تایمر متوقف شده و با فعال شدن مجدد آن زمان گیری را از ابتدا انجام می دهد.

_ با فعال شدن پایه R ریست ، تایمر زمان گیری را متوقف می کند و تا زمانی که این سیگنال فعال است عملی انجام نمی دهد. در پایه BI زمان باقی مانده تا پایان کار تایمر گنجانده می شود که از جنس INTEGER است همین مقدار به صورت BCD در پایه BCD تایمر نیز موجود است تا کاربر با توجه به نیاز خود از آنها استفاده نماید. شماره تایمر نیز در محل بالای بلوک وارد می شود که از فرمت : (شماره تایمرT) استفاده می شود. برای وارد نمودن مقدار زمان در تایمر ها از فرمت (زمان مورد نظر#s5t) استفاده می گردد.زمان مورد نظر می تواند میلی ثانیه ms، ثانیه s، دقیقه m یا ساعت h باشد. حداکثر زمانی که یک تایمر می تواند زمان گیری کند برابر با 2 ساعت و 46 دقیقه و 30 ثانیه است.



مثال: درب اتوماتیک را در نظر بگیرید که وقتی شخص در جلوی سنسور آن قرار دارد درب به مدت 15 ثانیه و 300 میلی ثانیه باز می شود و بعد از این مدت بسته می شود اگر پس از باز شدن درب شخص از جلوی سنسور خارج گردد درب بسته می شود. سنسور به آدرس I 2.4 و درب به خروجی Q 0.7 متصل شده است.

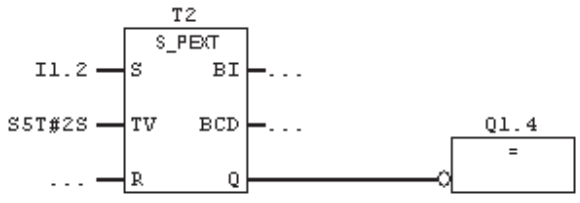
پاسخ: برای انجام این کار از یک تایمر S_Pulse استفاده شده است.



تایمر S_PEXT: عملکرد این تایمر کاملاً مشابه تایمر S_PULSE می باشد با این تفاوت که ورودی تحریک این تایمر حساس به لبه است یعنی با دریافت لبه بالارونده تایمر شروع به زمانگیری می کند اگر در حین زمان گیری لبه دیگری بیاید تایمر مجدداً شروع به زمان گیری می نماید.

مثال: اجسام بر روی نوار نقاله در حال حرکتند فاصله بین دو جسم با توجه به سرعت نوار نقاله دو ثانیه است. وجود اجسام توسط یک سنسور تشخیص داده می شود. با عبور اجسام از جلوی سنسور یک پالس تولید می گردد. برنامه ای بنویسید که اگر فاصله بین دو جسم بیش از 2 ثانیه شد لامپ متصل به خروجی Q1.4 روشن شود. سنسور به ورودی I 1.2 متصل شده است.

پاسخ:



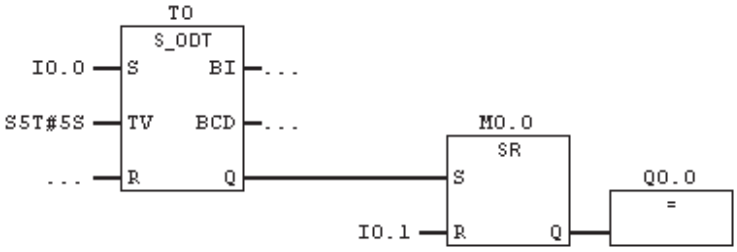
با عبور جسم تایمر روشن می شود و چون از گیت NOT استفاده شده است لامپ مربوطه خاموش می باشد. اگر فاصله بین دو جسم بیشتر از زمان تنظیمی باشد تایمر خاموش شده و خروجی لامپ روشن می گردد.

تایمر تأخیر در وصل(S_ODT): (On Delay Timer) با فعال شدن دائم پایه ورودی، خروجی پس از زمان تنظیمی TV، فعال می شود. اگر در مدت زمان گیری ورودی غیر فعال گردد و دوباره فعال شود زمانگیری از ابتدا آغاز می گردد. عملکرد مابقی پایه ها مشابه عملکرد پایه های توضیح داده شده در تایمر S_PULSE است.

مثال: برای راه اندازی یک الکتروموتور لازم است اپراتور شستی برق را حداقل برای مدت 5 ثانیه فشار دهد و پس از آن با برداشتن دست، موتور به طور دائم کار کند و اگر قبل از 5ثانیه اپراتور دست خود را بردارد عمل زمانگیری متوقف شود و با تحریک شستی STOP موتور متوقف شود. شستی راه انداز به ورودی I 0.0 و کنتاکتور موتور به خروجی Q 0.0 و شستی STOP به ورودی I 0.1 متصل شده است.

پاسخ:

در اینجا از یک تایمر تأخیر در وصل استفاده شده است که با تحریک دائم ورودی پس از 5s خروجی آن فعال می شود.



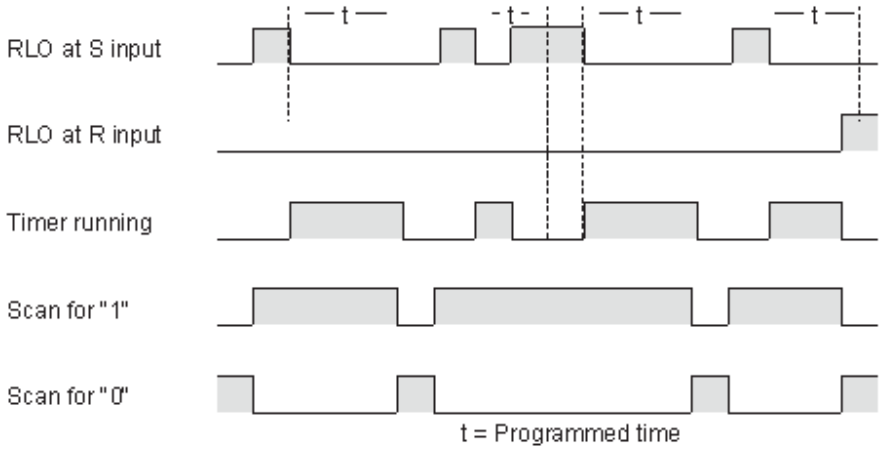
تایمر تأخیر در وصل پایدار (S_ODTS):

عملکرد این تایمر مشابه تایمر تأخیر در وصل است با این تفاوت که ورودی این تایمر حساس به لبه بالا رونده است و با تحریک لبه بالا رونده، تایمر شروع به زمان گیری می کند و اگر لبه بالا رونده دیگری مشاهده کند زمان گیری از ابتدا آغاز می شود.

تایمر تأخیر در قطع(S_OFFDT) Off Delay Timer :

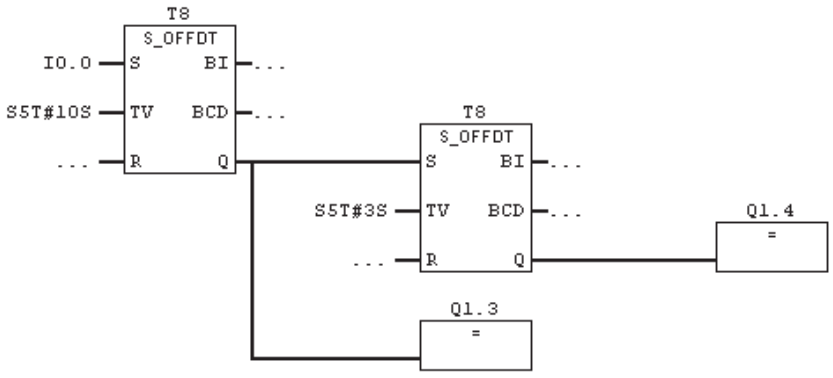
این تایمر حساس به لبه پائین رونده است یعنی زمانی که در ورودی S این بلوک لبه پائین رونده قرار بگیرد تایمر شروع به زمان گیری می کند و پس از سپری شدن زمان تنظیمی در پایه TV خروجی دیجیتال غیر فعال می گردد. با

آمدن لبه بالا رونده بعدی تایمر عکس العملی از خود نشان نمی دهد اما اگر تایمر در حال زمان گیری باشد و لبه پائین رونده بعدی بیاید تایمر زمان گیری را از ابتدا انجام می دهد. نمودار زمانی عملکرد این تایمر نشان داده شده است.



مثال: می خواهیم سیستم خنک کننده یک موتور، 10 ثانیه پس از خاموشی موتور خاموش شود و 3ثانیه پس از خاموشی سیستم خنک کننده نیز خاموش گردد. دو چراغ سیگنال نیز برای بررسی صحت عملکرد این روند در نظر گرفته شده است. با دریافت فرمان خاموش خروجی Q1.3 و Q1.4 روشن می شوند که با خاموش شدن موتور Q1.3 خاموش و با خاموش شدن سیستم خنک کننده Q1.4 خاموش می گردد.

پاسخ:

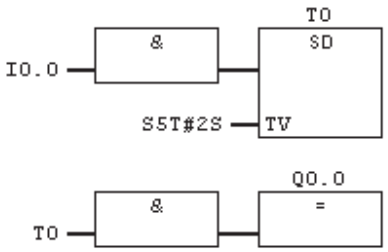


برای نوشتن این برنامه از دو تایمر تأخیر در قطع استفاده شده است. با لبه پائین رونده I1.1 تایمر T10 پس از 10ثانیه خاموش می شود و خروجی

Q1.3 را خاموش می کند و همچنین یک لبه پائین رونده برای تایمر T8 تولید می کند و با آمدن این سیگنال تایمر T8 نیز پس از 3 ثانیه خروجی خود را غیرفعال می کند.

بلوک های [SF],[SS],[SD],[SE],[SP]: این بلوک همان تایمر های فوق می باشند که به طور بیتی عمل می کنند و ترتیب این تایمر ها با ترتیب تایمرهای اصلی آنها یکی می باشد. در اینجا با ذکر یک مثال، نحوه عملکرد یک نمونه تایمر را نشان می دهیم.

مثال:



در برنامه روبه رو از یک تایمر تاخیر در وصل استفاده شده است. با تحریک دائم ورودی I 0.0 تایمر شروع به زمان گیری می نماید و پس از فعال شدن تایمر، خروجی متصل به تایمر فعال می گردد.این مدل برنامه نویسی زمانی کاربرد دارد که کاربر بخواهد بین تایمر و خروجی آن فاصله ایجاد کند یا بخواهد خروجی چند تایمر را به طور یکجا در یک NETWORK ببیند.

بلوک AND با یک ورودی را خود نرم افزار اضافه می کند، کاربر کافیست آدرس را به طور عادی وارد نماید.

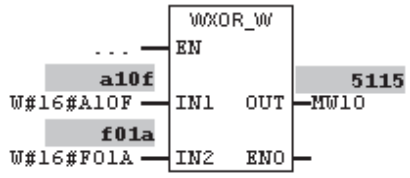
✓ **عملیات منطقی بر کلمات:** word logic

این گروه شامل عملیات منطقی بر روی Word یا دابل Word می باشد. منظور از عملیات منطقی همان عملیات AND , OR و XOR می باشد اما بر روی یک Word یا یک دابل Word . این عملیات به صورت بیتی انجام می گیرد . برای مثال وقتی دو Word با هم AND می شوند تک تک بیت های آن با هم AND شده و در نهایت جواب هر AND ، Word نهایی را تشکیل می دهد.

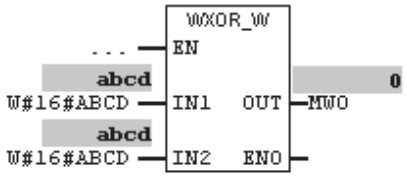
مثال : عدد A10F و عدد F01A را با هم XOR نمائید و عدد حاصل را در MW10 ذخیره نمائید.

پاسخ: از بلوک XOR ویژه Word استفاده شده است عدد حاصل نیز در شکل نشان داده شده است.

با استفاده از Word Logic می توان عملیات منطقی خاصی را پیاده کرد و به نتایج جالبی رسید به مثال های زیر توجه کنید.

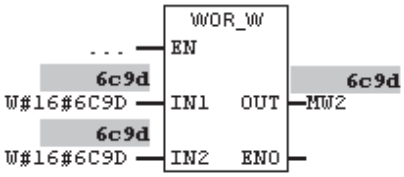


مثال : یک عدد به دلخواه انتخاب نموده و آن عدد را با خودش XOR نمائید حاصل چه عددی است؟



پاسخ: هر عدد با خودش XOR شود حاصل برابر با صفر می گردد.

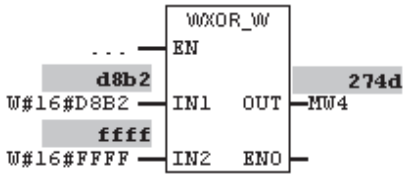
مثال : عددی به دلخواه انتخاب نمائید عدد را با خودش AND یا OR نمائید حاصل چیست؟



پاسخ : در اینجا عدد با خودش OR شد و جواب نهایی برابر با خود عدد اولیه است. هر عددی با خودش OR یا AND شود حاصل همان عدد می گردد.

مثال : عددی را به دلخواه انتخاب کنید حال عدد را با عددw#16#FFFF ، XOR نمائید. حاصل چیست؟

پاسخ: اگر عدد های ورودی و خروجی این بلوک را با هم مقایسه نمائید می بینید که عدد خروجی متمم عدد ورودی است.



عدد d8b2: 1101_1000_1011_0010

عدد 274d: 0010_0111_0100_1101

همان طور که دیده می شود تمامی 1 ها به صفر و صفر ها به 1 تبدیل شده اند.

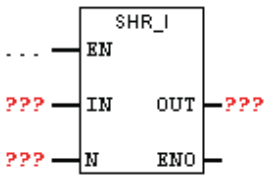
تمامی موارد فوق برای توابع DW نیز برقرارست با این تفاوت که اعداد در این دسته 4 بایت را اشغال می نمایند.

✓ **شیفت و چرخش** Shift / Rotate:

از دستورات این گروه جهت اعمال شیفت و چرخش در بیت ها استفاده می گردد. شیفت دادن یک بیت یا عملیات چرخش می تواند در جهت راست یا چپ صورت گیرد که در نهایت به آن شیفت به راست ، شیفت به چپ ، چرخش به راست و چرخش به چپ گفته می شود.

SHR_I وSHR_DI : عمل شیفت به راست اعداد Integer و دابل Integer را انجام می دهند. از آنجائیکه این اعداد دارای فرمت مثبت ومنفی می باشند فقط عملیات شیفت به راست در آنها معنی پیدا می کند.

در یک عدد از نوع Integer که شامل 2 بایت یا 16 بیت می باشد آخرین بیت سمت چپ بیت علامت می باشد که در روند شیفت به راست نباید تغییر کند. در عمل شیفت به راست اعداد بر 2 تقسیم می شوند.



بلوک این عملگر ها دارای 3 ورودی و دو خروجی می باشد. پایه EN به عنوان فعال ساز این بلوک می باشد و همان طور که می بینید استفاده از آن اختیاری است.

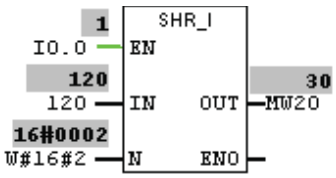
پایه IN عدد ورودی را دریافت می کند که این عدد باید به فرم عدد INT یا DINT باشد .

پایه N برای تعیین تعداد شیفت می باشد که اعداد باید به فرم : (عدد مورد نظر W#16#) وارد شوند.

خروجی OUT نیز مطابق با نوع ورودی ولی شیفت یافته آن می باشد که می تواند از نوع فضای حافظه یا خروجی باشد.

پایه ENO : زمانی که بلوک عملیات شیفت را با موفقیت انجام دهد این پایه در خروجی روشن خواهد شد.

مثال : عدد 120 را به میزان 2 بیت به سمت راست شیفت دهید در مرحله بعد عدد 120- را شیفت دهید و عملکرد این بلوک را مشاهده نمائید.



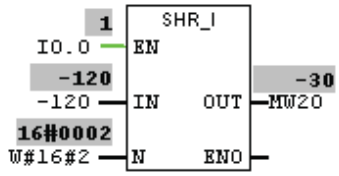
پاسخ : همانطور که می بینید به ازای هر شیفت عدد ورودی بر 2 تقسیم می شود و چون دو بار شیفت داده شده است عدد نهایی بر 4 تقسیم شده است.

به فرمت این اعداد در حالت باینری دقت کنید:

120: 0000_0000_0111_1000

30: 0000_0000_0001_1110

حال عدد 120- را بررسی کنید:



باز هم خروجی بر 4 تقسیم شده است حال به فرمت اعداد در حالت باینری دقت کنید:

120- : 1111_1111_1000_1000

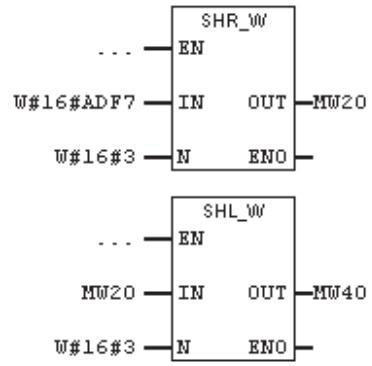
30- : 1111_1111_1110_0010

SHR_W و SHL_W: عمیات شیفت به راست و شیفت به چپ توسط این دو بلوک صورت می گیرد. شکل ظاهری و نحوه عملکرد بلوک ها مشابه نمونه های قبلی می باشد با این تفاوت که فرمت اعداد ورودی و خروجی متفاوت می باشد و پس از عملیات شیفت بیت های خالی با عدد صفر پر می شوند. مثبت و منفی بودن اعداد در اینجا بی معنی است.

مثال: عدد ADF7 را پس از 3 مرحله شیفت دادن به راست به اندازه 3مرتبه به سمت چپ شیفت دهید. آیا عدد نهایی با عدد اولیه یکسان است؟

پاسخ: ابتدا عدد توسط بلوک SHR_W به سمت راست به میزان 3 بیت شیفت داده شده و عدد مربوطه در فضای حافظه MW20 ذخیره شده است و سپس به اندازه 3 بیت به سمت چپ شیفت داده شده است.

حال به بررسی هر یک از خروجی ها در زیر توجه کنید:



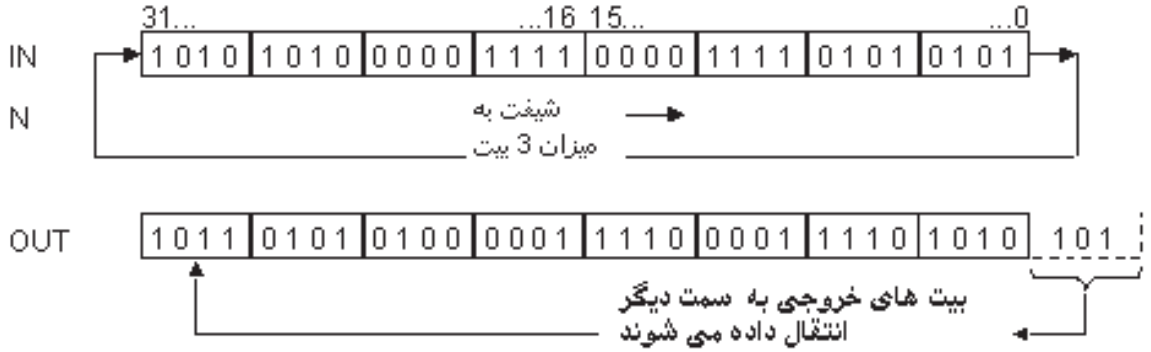
ADF7: 1010_1101_1111_0111

MW20: 0001_0101_1011_1110 این عدد شیفت یافته عدد بالا به میزان 3 بیت به سمت راست می باشد.

MW40: 1010_1101_1111_0000 این عدد شیفت یافته عدد MW20 به سمت چپ به میزان 3بیت می باشد. با مقایسه عدد ابتدایی و عدد نهایی می توان مشاهده کرد که این دو عدد با هم متفاوت می باشند.

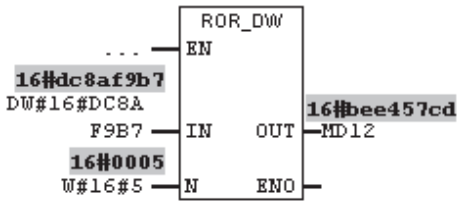
SHL_DW و SHR_DW: عملکرد این گروه دقیقاً مشابه گروه SHI_W می باشد با این تفاوت که ورودی آن از نوع دابلWORD و 4 بایتی می باشد.

ROL_DW و ROR_DW: این بلوک ها عملیات چرخش (Rotate) را انجام می دهند همانطور که از اسم این دستورات مشخص است فقط بر روی داده های با فرمت دابل Word اجرا می شوند. عملکرد چرخش دقیقاً مشابه عمل شیفت است با این تفاوت که ابتدا و انتهای بیت به هم متصل می باشد یعنی در عمل شیفت به راست ، آخرین بیت سمت راست خارج شده و وارد آخرین بیت سمت چپ می شود (تشکیل حلقه می دهند) به شکل زیر توجه کنید.



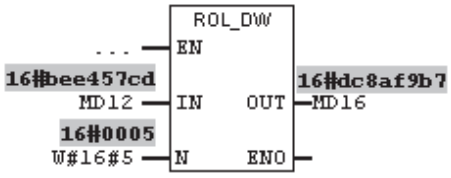
مثال : عدد DC8AF9B7 که از جنس دابل Word می باشد را در نظر بگیرید حال این عدد را 5 بار به سمت راست چرخش داده و عدد حاصل را 5 بار به سمت چپ بچرخانید. آیا عدد حاصل با عدد انتخابی برابر است؟

پاسخ: ابتدا عدد را به بلوک چرخش به راست داده و تعداد چرخش را 5 در نظر بگیرید و حاصل را در MD12 ذخیره نمائید. به فرمت اعداد ورودی دقت کنید.



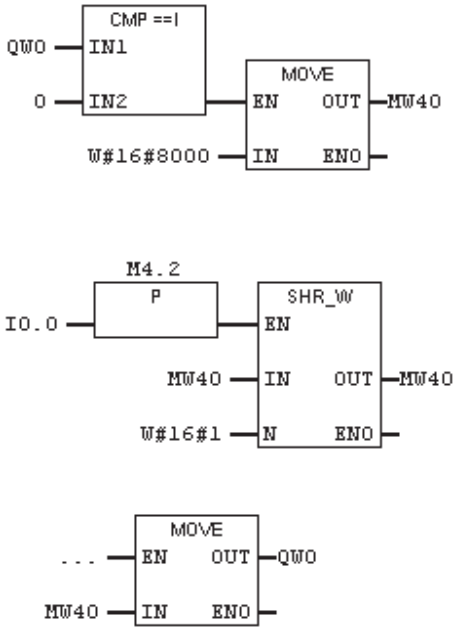
حال با استفاده از بلوک چرخش به چپ عدد فوق را 5 بیت به سمت چپ بچرخانید.

همان طور که می بینید عدد انتهایی و عدد ابتدایی با هم برابر می باشند.



مثال : به16 خروجی یک PLC ، LED متصل شده است و از بین این LED ها یک LED روشن و بقیه خاموش می باشند .کاربر می خواهد با هر بار تحریک ورودی مربوطه این LED روشن به اندازه یک واحد به سمت راست حرکت نماید و پس از رسیدن به آخرین LED سمت راست دو باره LED روشن به انتهای سمت چپ رفته و این عمل همچنان ادامه پیدا کند.

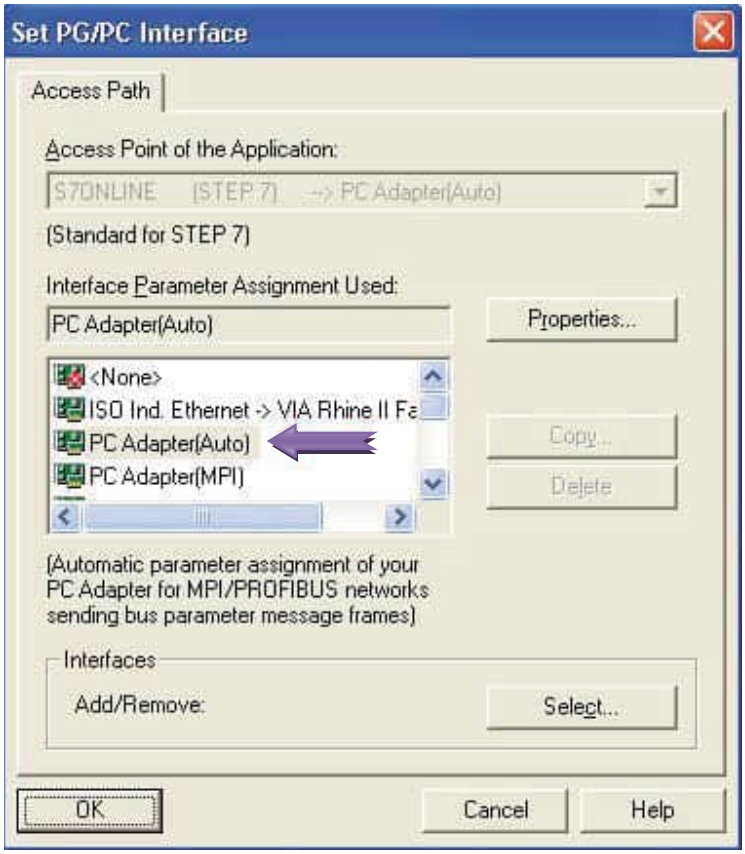
پاسخ: در ابتدا لازم است تا عدد 8000 در مبنای 16 به این خروجی داده شود و با هر بار تحریک ورودی این عدد یک واحد به سمت راست شیفت داده شود. از بلوک تشخیص دهنده لبه استفاده شده است در غیر این صورت تا زمانی که پایه EN فعال باشد عملیات شیفت انجام می گیرد که این مطلوب نیست. پس از انجام شیفت عدد حاصل به خروجی انتقال داده می شود و به ابتدای برنامه برگشته و عدد حاصل را با صفر مقایسه می کند اگر این بلوک فعال شود یعنی LED روشن به انتهای سمت راست رسیده است و با فعال کردن



بلوک MOVE دوباره LED روشن به سمت چپ باز می گردد.

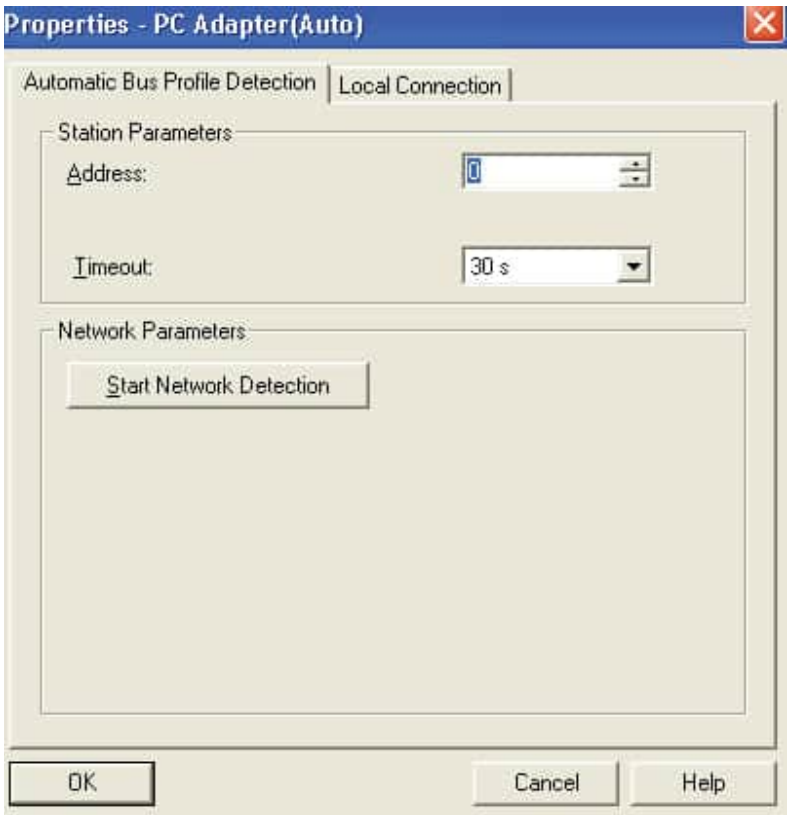
ارتباط با PLC :

توضیحات ارائه شده در این مباحث بر روی محیط شبیه سازی اعمال می شده است برای ارتباط کامپیوتر با PLC جهت انتقال اطلاعات لازم است ابتدا کابل مربوطه را نصب نمائید سپس از منوی Option گزینه Set PG/PC Interface... را انتخاب نمائید تا پنجره زیر باز شود.



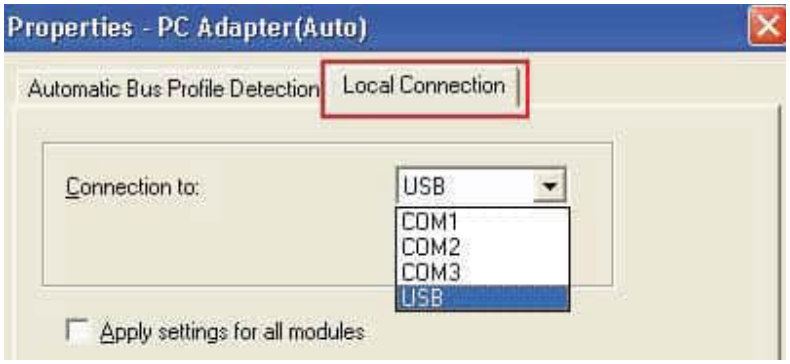
بر روی گزینه مشخص شده در شکل دابل کلیک نمائید یا آن را انتخاب نموده و بر روی گزینه Properties کلیک کنید تا صفحه تنظیمات آن مطابق شکل زیر باز شود.

در قسمت Address می توان آدرس مربوط به ایستگاهی را که CPU در آن قرار دارد تعیین نمود. این عدد به طور پیش فرض صفر می باشد. در قسمت Timeout زمان پاسخ گویی تعیین می گردد. اگر تلاش کامپیوتر برای ارتباط با PLC از این زمان بیشتر شود اخطاری ظاهر می گردد که ارتباط دچار مشکل می باشد.



با انتخاب لبه Local Connection وارد صفحه زیر خواهید شد که تنظیمات دیگری از نحوه ارتباط در آن قرار داده شده است.

با توجه به نوع کابل موجود، ارتباط را بر روی گزینه USB قرار دهید.



حال بر روی گزینه Start Network Detection کلیک نمائید اگر مشکلی در ارتباط وجود نداشته باشد اطلاعات مربوط به Network نمایش داده خواهد شد. در انتها نیز بر روی گزینه OK کلیک نمائید.

فصل چهارم

پردازش سیگنال های آنالوگ

در این فصل آشنا خواهید شد با ...

۱. آشنایی با انواع سیگنال های آنالوگ
۲. آدرس دهی و تنظیمات نرم افزاری کارت های آنالوگ
۳. برنامه نویسی و آشنایی با بلوک های سیگنال های آنالوگ

پردازش سیگنال های آنالوگ:

با مفهوم سیگنال آنالوگ در مبحث آشنایی با کارت ها آشنا شدید و انواع آن را مشاهده نمودید. اکثر پارامترهایی که در محیط اطراف وجود دارند از نوع آنالوگ می باشد مانند : فشار ، دما ، فاصله ، سرعت ، وزن و غیره . به همین جهت سنسور هایی که دارای خروجی آنالوگ می باشند نیز از محبوبیت بیشتری برخوردار هستند از طرفی خروجی این سنسورها حاوی اطلاعات بیشتری نسبت به سیگنال های دیجیتال می باشد.

همان طور که گفته شد سیگنال های آنالوگ برای انتقال به PLC وسپس به کامپیوتر ابتدا باید به حالت دیجیتال تبدیل شوند که این کار توسط عمل کوانتیزه کردن یا سطح بندی سیگنال های آنالوگ صورت می گیرد. میزان این سطوح بستگی به تعداد بیت های رزلوشن دارد و رابطه زیر برقرار است.

$$\text{تعداد سطوح کوانتیزه} = 2^{\text{تعداد بیت ها}}$$

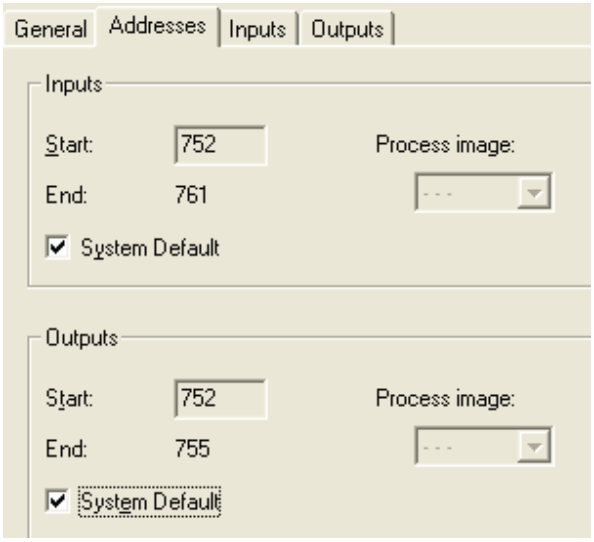
سیگنال های آنالوگ توسط مداراتی موسوم به ADC (Analog Digital Converter) مبدل آنالوگ به دیجیتال صورت می گیرد. این مدار در ورودی هر کارت ورودی آنالوگی تعبیه شده است و CPU دیجیتال شده سیگنال آنالوگ را دریافت می کند. عکس همین عمل نیز برای خروجی آنالوگ صدق می کند. سیگنال های کنترلی تولید شده توسط CPU از نوع دیجیتال می باشند در نتیجه توسط مداراتی موسوم به DAC (Digital Analog Converter) مبدل های دیجیتال به آنالوگ صورت می گیرند. تعداد رزلوشن و همچنین تعداد کانال های یک کارت را کاربر با توجه به نیاز خود می تواند انتخاب نماید.

آدرس دهی در سیگنال های آنالوگ:

CPU جهت پردازش سیگنال های آنالوگ به علت تغییرات دائمی که دارد از جداول PII و PIQ استفاده نمی کند و فضای دیگری برای آن در نظر می گیرد که این فضا با آدرس PIW (peripheral Input Word) برای سیگنال های ورودی و PQW برای سیگنال های خروجی قابل تعریف هستند.سیگنال های آنالوگ پس از تبدیل به دیجیتال فضایی به اندازه یک Word یا همان 2 بیت را اشغال می نمایند.

شماره آدرس کانال های آنالوگ را می توان در پیکر بندی سخت افزار مشاهده نمود و تغییر داد. اگر آدرس شروع سیگنال های آنالوگ PIW 256 باشد آنگاه آدرس شروع کانال بعدی از PIW 258 آغاز می گردد. کاربر نمی تواند از PIW 257 به عنوان ورودی آنالوگ استفاده نماید. سیستم خطایی نمی دهد ولی سیگنالی که دریافت می کند نامعتبر است.

همین موارد برای خروجی آنالوگ نیز صادق می باشند. در شکل زیر این آدرس ها نشان داده شده اند.



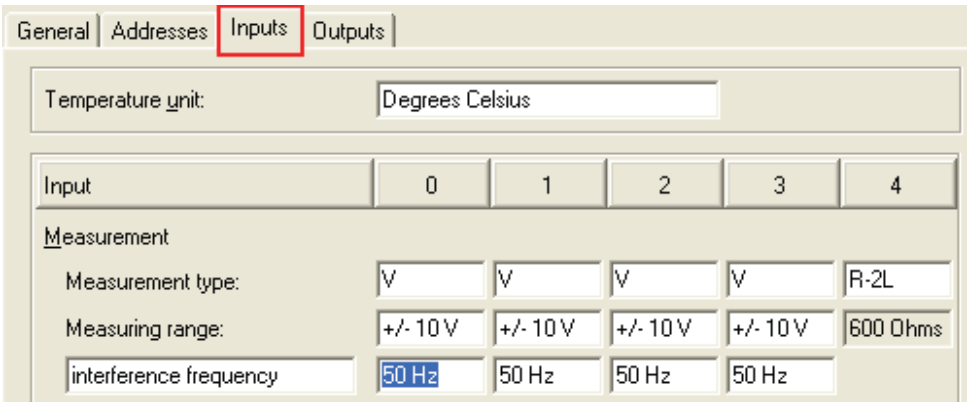
تعیین نوع ورودی از طریق نرم افزار:

سیگنال های آنالوگ به 4 دسته زیر تقسیم بندی می شوند:

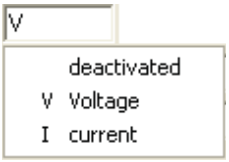
- ولتاژی
- جریانی
- مقاومتی
- ترموکوپلی

با توجه به تعداد سیگنال های آنالوگ و تفاوت سیم بندی این سیگنال ها ، علاوه بر تنظیمات سخت افزاری باید تنظیمات نرم افزاری نیز صورت بگیرد که در قسمت پیکر بندی سخت افزار این کار انجام می شود.

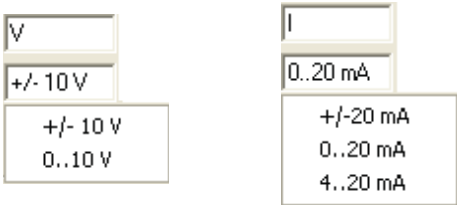
به قسمت پیکر بندی سخت افزار وارد شوید و بر روی کارت آنالوگ موجود دابل کلیک کنید تا کادر مربوط به تنظیمات آن ظاهر شود. ابتدا کارت ورودی آنالوگ توضیح داده خواهد شد. در زیر تنظیمات انواع کارت ها توضیح داده شده است.



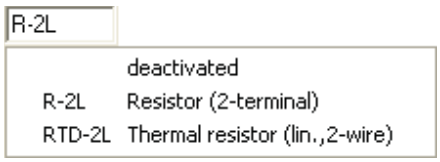
همان طور که در شکل بالا مشاهده می نمائید این کارت دارای 4 ورودی ولتاژ یا جریانی و یک ورودی مقاومتی می باشد. با کلیک بر روی نوع سیگنال ، منوی زیر ظاهر می گردد که می توان ورودی ولتاژ ، جریان و یا حالت غیر فعال را انتخاب نمود. اگر حالتdeactivated(غیر فعال) انتخاب شود باید به صورت سخت افزاری آن ورودی را اتصال کوتاه گردد. این کار باعث افزایش سرعت پردازش کارت می گردد.



در قسمت رنج اندازه گیری می توان بازه سیگنال آنالوگ را مشخص نمود در شکل زیر بازه ولتاژی وجریانی نشان داده شده است.



در مورد ورودی پنجم که از جنس مقاومت می باشد نیز همین مسائل صادق است.

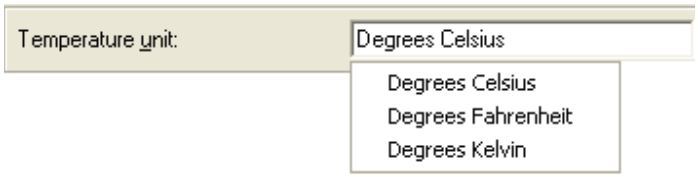


R-2L: این گزینه مربوط به استفاده از خط کش های مقاومتی است که معمولاً برای تعیین موقعیت استفاده می گردد.

RTD-2L: این گزینه برای استفاده از سنسور های دمایی از جنس RTD است که معروفترین آنها PT100 از نوع سه سیمه می باشد ولی همان طور که در گزینه ها دیده می شود واژه 2-wire (دوسیمه) ذکر شده است. در هنگام استفاده از سنسور های 3 سیمه کافیست سیم های هم رنگ سنسور را به هم اتصال دهید.

گزینه interference frequency (فرکانس تداخل) برای جلوگیری از تداخل عملیات تبدیل آنالوگ کانال ها با هم دیگر می باشد که بهتر است بر روی عدد پیش فرض آن 50 Hz قرار داشته باشد.

کاربر با توجه به نحوه اندازه گیری دما می تواند واحد اندازه گیری خود را انتخاب نماید که این واحد ها به ترتیب عبارتند از درجه سلسیوس (سانتی گراد) ، درجه فارنهایت و درجه کلوین. البته این واحد ها با توجه به رابطه زیر قابل تبدیل به هم نیز می باشند.

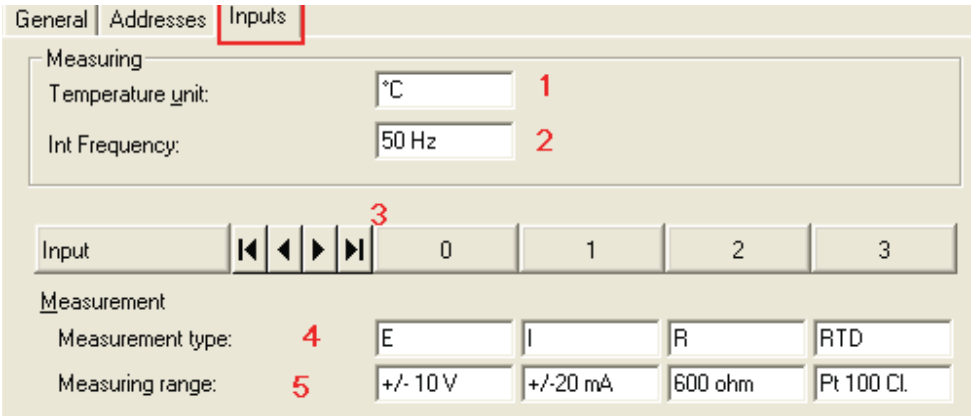


تبدیل درجه سلسیوس به فارنهایت:

تبدیل درجه سلسیوس به کلوین:

با توجه به روابط بالا دمای جوش آب معادل 100 درجه سلسیوس ، 212 درجه فارنهایت و 373 درجه کلوین می باشد.

حال کارت SM 331 با کد فنی 6ES7 331-1KF01-0AB0 را در نظر بگیرید . این کارت دارای 8 ورودی آنالوگ می باشد تنظیمات این کارت با تنظیمات قبلی اندکی متفاوت است. پس از قرار دادن کارت مربوطه در اسلات مورد نظر بر روی آن دابل کلیک نمائید. در لبه Inputs تنظیمات مربوطه موجود می باشد.



۱) تنظیم واحد اندازه گیری

۲) تعیین فرکانس تداخل

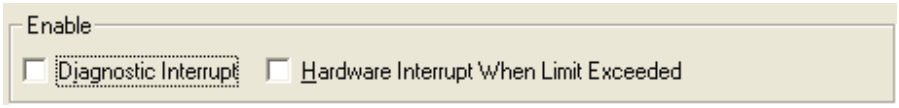
۳) شماره کانال ها و نشانگر های حرکت شماره کانال ها

۴) نوع سیگنال آنالوگ ورودی

۵) رنج اندازه گیری

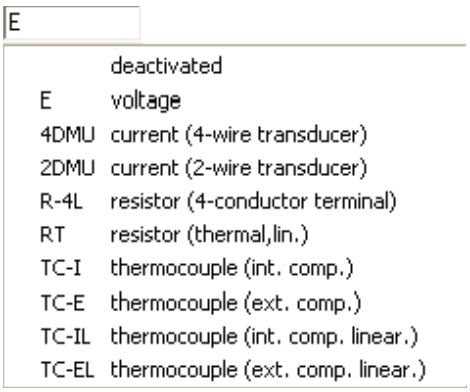
همان طور که در شکل بالا نشان داده شده است هر کانال می تواند ورودی متفاوت دریافت کند.

کارت ورودی آنالوگ SM 331 با دو ورودی آنالوگ 12 بیتی با شماره فنی 6ES7 331-7KB01-0AB0 را در نظر بگیرید. کارت مربوطه را در اسلات مورد نظر قرار دهید حال بر روی کارت دابل کلیک نموده تا صفحه تنظیمات آن باز شود. از لبه Inputs می توان به تنظیمات کارت دسترسی پیدا کرد. در ابتدای این صفحه دو گزینه وجود دارد که فعال ساز وقفه برای این کارت می باشند.



با فعال کردن اولین گزینه سمت چپ در صورت بروز خطا در کارت، کارت به طور خودکار فرمان وقفه صادر می کند و وقفه تنظیمی در CPU اجرا خواهد شد. با فعال نمود گزینه دوم ، زمانی که رنج سیگنال ورودی از رنج تعیین شده بیشتر گردد کارت سیگنال وقفه سخت افزاری را تولید می نماید تا OB تنظیمی در CPU اجرا شود.

در قسمت مربوط به Measuring نیز می توان نوع ورودی را انتخاب نمود. همانطور که در این تنظیمات مشاهده می کنید این کارت قابلیت اندازه گیری سیگنال های آنالوگ از جنس ترموکوپل و جریان های 4 سیمه را نیز دارد.



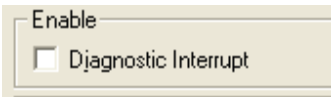
با انتخاب هر یک از موارد روبه رو تنظیمات جزئیات آن نیز ظاهر می شود.

Trigger for Hardware Interrupt: راه اندازی وقفه سخت افزاری زمانی فعال می شود که کاربر گزینه سمت راست در بالای همین صفحه را فعال نماید آن زمان می تواند حداقل و حداکثر مقدار مجاز سیگنال آنالوگ را وارد نماید.

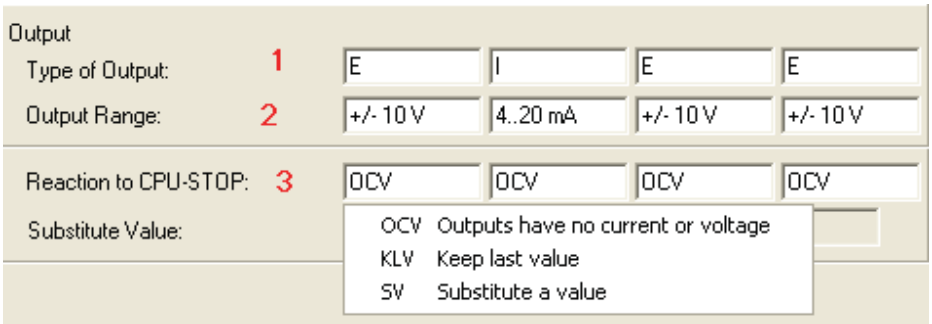
تنظیمات کارت های خروجی آنالوگ:

سیگنال های خروجی به دو دسته ولتاژی و جریانی تقسیم می شوند که این کارت ها نیز مانند کارت ها قبلی دارای تنظیماتی می باشند و کاربر می تواند مطابق نیاز خود آن را انتخاب نماید.

کارت آنالوگ با 4 خروجی 16 بیتی را با کد فنی 6ES7 332-7ND01-0AB0 را در نظر بگیرید. کارت را در اسلات مورد نظر قرار داده و روی آن دابل کلیک نمائید.در صفحه باز شده لبه Output را انتخاب نمائید.



در قسمت Enable با فعال کردن این گزینه وقفه سخت افزاری نیز فعال می گردد.



۱) نوع خروجی

۲) بازه تغییرات خروجی

۳) عکس العمل سیستم زمانی که CPU در مد استپ است.

وقتی CPU در حالت Stop قرار بگیرد برای خروجی آنالوگ آن 3 حالت می توان در نظر گرفت:

۱- OCV: خروجی هیچگونه ولتاژ یا جریانی نداشته باشد.

۲- KLV: آخرین مقدار خود را حفظ نماید.

۳- SV: عدد خاصی جایگزین شود. زمانیکه این گزینه انتخاب می شود قسمت زیرین آن فعال شده که کاربر می تواند عدد دلخواه خود را وارد نماید.

برای مثال در یک کارخانه تولید مواد شیمیایی ، پروانه ظرف مخلوط کن باید همواره بچرخد در غیر این صورت باعث نابودی مواد می شود. اگر بر اثر اشتباه در برنامه نویسی CPU به مد Stop برود با تنظیم خروجی کارت آنالوگ بر روی مقدار دلخواه می توان این پروانه را دائم در حرکت نگه داشت و از خسارت ناشی از آن جلوگیری نمود.

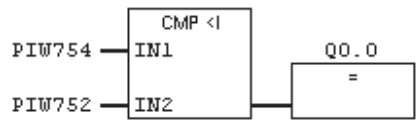
کارت های خروجی به علت محدودیت در نوع سیگنال ، بر خلاف کارت های ورودی از تنوع زیادی برخوردار نمی باشند.

برنامه نویسی با استفاده از سیگنال های آنالوگ:

در برنامه نویسی سیگنال های آنالوگ باید این نکته را در نظر داشت که این سیگنال از جنس Word می باشد و در وارد کردن فرمت اعداد باید دقت نمود.

مثال : ورودی آنالوگ اول را دریافت نماید و با ورودی آنالوگ دوم مقایسه کنید هرگاه ورودی اول بیشتر از ورودی دوم شد خروجی Q0.0 روشن شود.

پاسخ: از بین مقایسه کننده های موجود فقط نوع INT شامل یک WORD می باشد و از آن استفاده شده است.



همان طور که گفته شد سیگنال های آنالوگ در دو بایت یا یک Word قرار می گیرند آخرین بیت سمت چپ بیت علامت می باشد که 1 بودن این بیت به معنی منفی بودن عدد است. بقیه بیت ها نیز مطابق با جدول زیر پر می شوند.

تعداد بیت های رزلوشن	شماره بیت ها															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
8	S	*	*	*	*	*	*	*	*	0	0	0	0	0	0	0
12	S	*	*	*	*	*	*	*	*	*	*	*	*	0	0	0
15	S	*	*	*	*	*	*	*	*	*	*	*	*	*	*	0

علامت S به معنی Sign علامت می باشد. بیت هایی که با ستاره نشان داده شده اند بیت های متغییر هستند و بیت های نشان داده شده با صفر نیز همیشه صفر می باشند. برای مثال کارت با رزلوشن 8 بیت را در نظر بگیرید، 7 بیت سمت راست آن همواره صفر خواهد بود بنابراین فاصله بین دو سطح متوالی 128 خواهد بود که این عدد برای کارت 12 بیتی 32 و برای کارت 15 بیتی 1 خواهد بود.

خروجی سنسور ها برحسب یکی از سیگنال های ولتاژ ، جریان ، مقاومت یا ترموکوپلی می باشد که وقتی برای CPU تعریف می گردد به بازه های تعریف شده برای CPU تبدیل می شوند. برای مثال سیگنال 0-10V به عدد 27648 -0 مقیاس می شود یا سیگنال جریانی 4-20 mA به عدد 27648-0 مقیاس می گردد. حد موجود برای عدد مقیاس شده 27648 می باشد.

نوع سیگنال	ولتاژی		جریانی		مقاومتی برای مثال خط کش 300Ω		سنسور دما برای مثال PT100	
حد بالا	10	27648	20	27648	300Ω	27648	850	8500
صفر	0	0	4	0	0	0	0	0
حد پائین	-10	-27648	----		-----		-200	-2000

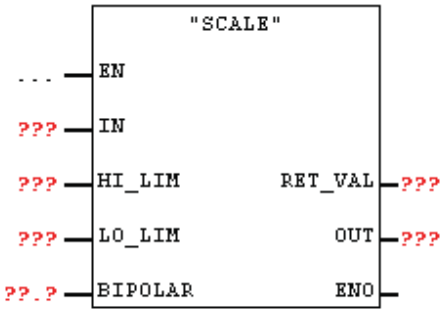
همان طور که از اعداد بالا مشخص است کار کردن با اعداد مقیاس شده برای CPU ، برای کاربر مشکل خواهد بود ، برای مثال کاربر علاقه مند به دانستن ارتفاع تانک مایع است ولی در خروجی CPU عددی به شکل 17348 موجود است که مفهومی برای کاربر ندارد. عمل تبدیل سیگنال زبان ماشین به زبان کاربر و بالعکس تحت نام مقیاس کردن و از مقیاس خارج کردن مطرح می شود که در ادامه با آن آشنا خواهید شد.

مقیاس کردن یک سیگنال آنالوگ:

همان طور که گفته شد عمل مقیاس کردن باید در ورودی CPU انجام شود تا عدد خروجی سنسور که به عدد قابل فهم برای ماشین تبدیل گشته توسط کاربر قابل استفاده باشد. این کار توسط FC 105 که یک فانکشن نوشته شده توسط خود شرکت زیمنس می باشد، انجام می گیرد. این فانکشن از مسیر زیر قابل دسترسی است:

Libraries --- Standard Library ---TI_S7 Converting Blocks --- FC 105 scale convert

شکل این بلوک در روبه‌رو نشان داده شده است همان طور که می بینید این بلوک دارای 5 ورودی و 3 خروجی می باشد.



EN: از نوع بیت می باشد که به عنوان فعال ساز این بلوک به کار می رود و استفاده از آن اختیاری است.

IN: ورودی آنالوگ می باشد که با PIW آغاز می گردد.

HI_LIM: حداکثر مقدار متغییر فیزیکی باید وارد شود این عدد از نوع Real می باشد. برای مثال ارتفاع یک تانک که می تواند 500 cm باشد.

LO_LM: حداقل مقدار متغییر فیزیکی می باشد که این عدد هم از نوع Real می باشد. برای مثال ارتفاع 50 cm

BIPOLAR: این متغییر از نوع بیت می باشد با فعال شدن این پایه خروجی سنسور به عنوان یک سیگنال دارای ناحیه منفی در نظر گرفته می شود. برای مثال اگر این پایه غیر فعال باشد ارتفاع 50 cm به عنوان صفر ولت و 500cm به عنوان 10V در نظر گرفته می شود ولی با فعال کردن این پایه ارتفاع 50cm به عنوان 10V- و ارتفاع 275 cm (نقطه میانی 500 و50 می باشد) به عنوان صفر ولت و ارتفاع 500 cm به عنوان 10V در نظرگرفته می شود.

RET_VALUE: مقدار بازگشتی است که برای بررسی خطا به کار می رود. در حالتی که عدد ورودی در بازه تعریف شده باشد این مقدار برابر با 0 است و در صورت بروز خطا این عدد 8 خواهد شد.

OUT: خروجی اصلی بلوک می باشد که از نوع Real بوده و باید برای آن دابل Word را در نظر گرفت.

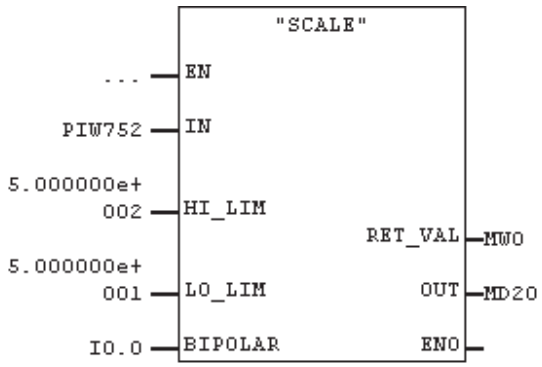
ENO: در صورتی که این بلوک عملیات خود را به درستی انجام دهد این پایه فعال می شود.

برای استفاده از این بلوک باید دقت شود که علاوه بر دانلود کردن OB1 باید FC 105 نیز به طور جداگانه دانلود شود

*** دقت کنید که این فانکشن از طرف شرکت زیمنس مسدود می باشد و کاربر اجازه ورود به آن را ندارد.

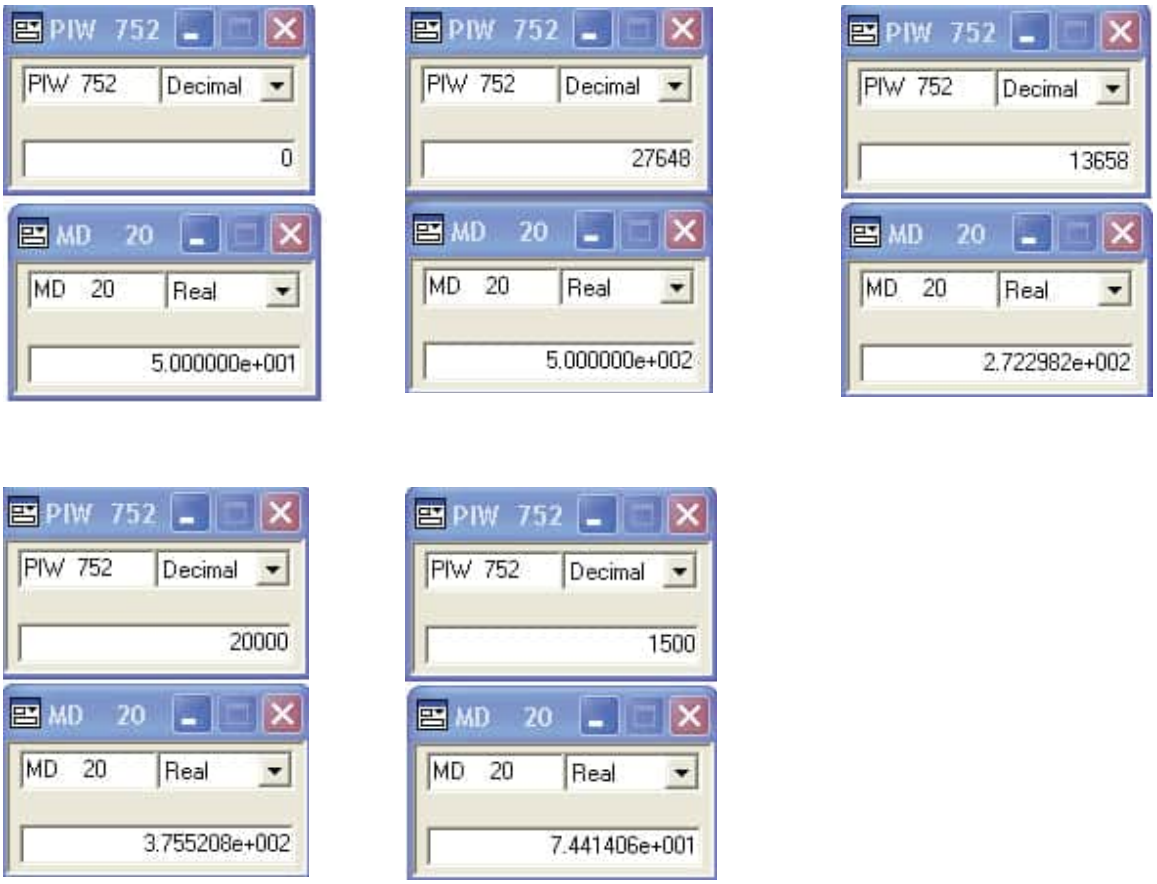
مثال: یک مخزن را در نظر بگیرید حداکثر ارتفاع آن 500cm و حداقل ارتفاع آن 50cm می باشد که به ازای این ارتفاع ها سنسور میزان 0-10V را تولید می کند.حال توسط بلوک fc 105 آنرا مقیاس کنید.

پاسخ:



عدد وارد شده در PIW 752 مقدار سیگنال به زبان ماشین می باشد و عدد موجود در MD 20 عدد قابل فهم برای کاربر.

حال به مقدار حافظه MD 20 را به ازای مقادیر مختلف PIW 752 توجه نمائید.



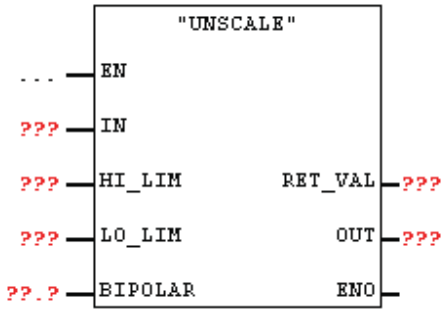
از مقیاس خارج کردن سیگنال آنالوگ:

در قسمت قبل دیده شد که با استفاده از FC 105 می توان عدد قابل فهم برای CPU را به عدد مناسب برای کاربر مقیاس کرد و کاربر می تواند در روند اجرای برنامه با مقادیر فیزیکی کار نماید. بعد از انجام تحلیل های لازم و تولید سیگنال مناسب ، حال باید عدد تولیدی را به زبان CPU تبدیل کرده و به کارت ها انتقال دهید. این عمل تبدیل از زبان کاربر به زبان CPU از مقیاس خارج کردن یا Unscale نامیده می شود.

این عمل توسط بلوک FC 106 UNSCALE CONVERT که توسط خود شرکت زیمنس نوشته شده است صورت می گیرد. این بلوک از مسیر زیر قابل دستیابی است:

Libraries --- Standard Library ---TI_S7 Converting Blocks --- FC 106 unscale convert

نمای کلی این بلوک در زیر نشان داده شده است. پایه های EN , ENo ,RET_VAL و BIPOLAR عملکردی مشابه بلوک قبلی دارند و از توضیح در اینجا خودداری شده است.



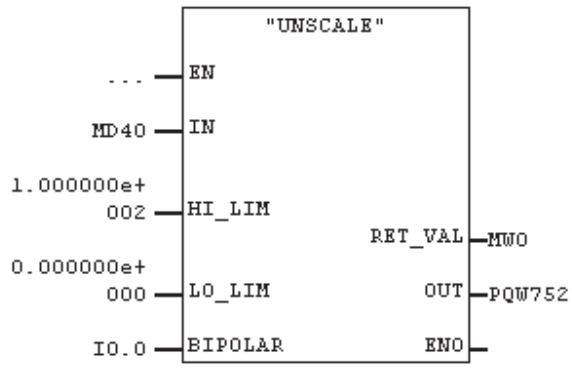
IN: ورودی اصلی این بلوک بوده که از نوع دابل Word می باشد.

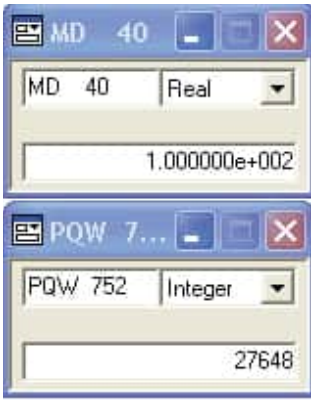
HI_LIM و :LO_LIM مقدار حداکثر و حداقل مقدار فیزیکی می باشد و فرمت این اعداد باید از نوع Real باشد.

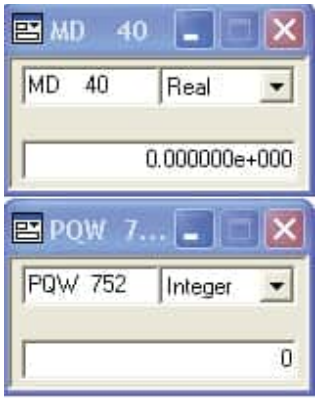
OUT: از نوع Word بوده که می توان به خروجی آنالوگ PQW انتقال داد.

مثال : شیر ورودی یک مخزن را در نظر بگیرید این شیر برای تعیین میزان ارتفاع مایع درون مخزن به کار می رود. میزان باز بودن شیر به در صد بیان می شود یعنی وقتی به آن سیگنال 50 اعمال می شود تا نیمه باز است و با اعمال سیگنال 100 شیر کاملاً باز می گردد. این عمل چگونه صورت می پذیرد؟

پاسخ: از بلوک FC 106 استفاده شده است و در ورودی که بیانگر متغییر های فیزیکی می باشد عدد 0 و عدد 100 وارد گردیده و خروجی این بلوک نیز به کارت آنالوگ اعمال گشته است. خروجی این بلوک به ازای مقادیر مختلف ورودی نشان داده شده است. عدد MD 40 ورودی می باشد که از سمت راست مقدار حداقل ، مقدار میانی و مقدار حداکثر وارد شده و خروجی نیز مقدار حداقل و میانی و حداکثر بازه 27648—0 را نشان می دهد.

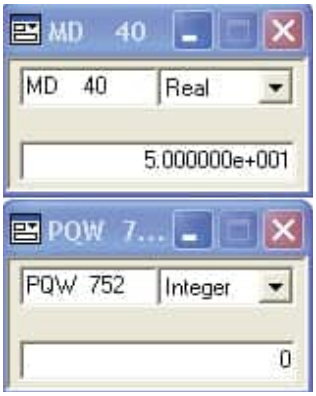




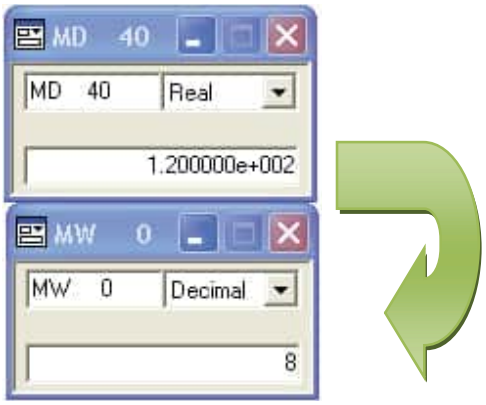


در شبیه سازی های بالا ورودی I0.0 صفر می باشد.

حال I0.0 را یک کنید با این عمل ، خروجی از 27648+ ---- 27648- تغییر خواهد کرد این مقادیر را با مقادیر قبلی مقایسه کنید.

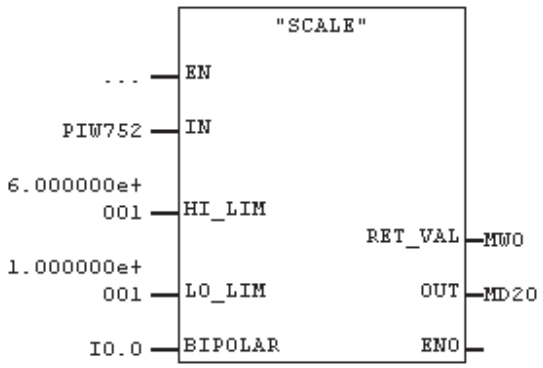


در تمامی موارد فوق عدد ورودی در بازه مجاز می باشد و مقدار RET_VAL برابر با 0 می باشد اگر عدد خارج از این بازه وارد شود مانند: 120 آنگاه مقدار 8 بر گردانده می شود. به شکل زیر دقت کنید:

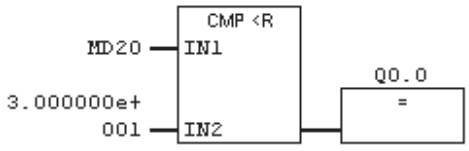


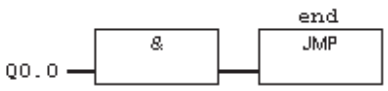
مثال : یک سنسور آلتراسونیک (ما فوق صوت) را در نظر بگیرید. از این سنسور جهت تشخیص فاصله استفاده می گردد. این سنسور به ازای بازه 10cm – 60 cm ، خروجی ولتاژی در بازه 10V– 0 می دهد. اگر جسم در فاصله کمتر از 30cm سنسور قرار گیرد خروجی Q0.0 روشن شود و در خارج از این بازه ، فاصله تا سنسور بر عدد 6 تقسیم شود و به عنوان ولتاژ ورودی به یک کنترل دور موتور ارسال نماید.

پاسخ: ابتدا عدد را از ورودی آنالوگ خوانده و آنرا به FC 105 مقیاس کنید تا در روند برنامه نویسی با کمیت فیزیکی فاصله سروکار داشته باشید. حاصل نیز در دابل Word مربوطه ذخیره می گردد.



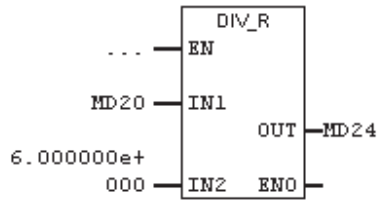
در قدم بعدی باید فاصله را با مقدار 30 مقایسه نمود. اگر این رابطه برقرار باشد خروجی



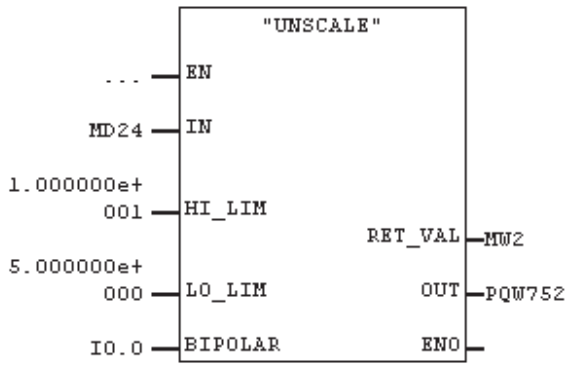


مربوطه روشن خواهد شد در غیر این صورت با استفاده از دستور پرش به انتهای سیکل رفته و کاری را انجام ندهید.

حال طبق صورت مسئله عدد تولیدی باید بر عدد 6 تقسیم شود که این کار با استفاده از بلوک DIV_R انجام می شود.(دقت شود که عدد مورد نظر از نوع Real می باشد)



حال عدد تولید شده را باید به خروجی اعمال کنید که این کار توسط FC 106 صورت می گیرد. حداکثر این مقدار برابر با 10 و حداقل آن برابر با 5 می باشد. این اعداد از تقسیم حداکثر و حداقل مقادیر سنسور بر عدد 6 بدست آمده است.




```
end:  NOP   0
```

در نهایت سیگنال خروجی نیز به کارت ارسال می گردد. و در انتها نیز برچسب end که کاری را انجام نمی دهد قرار داده شده است.

فصل پنجم

برنامه نویسی ساختار یافته (فانکشن و فانکشن بلاک ها)

در این فصل آشنا خواهید شد با .....

نحوه عملکرد فانکشن ها و فانکشن بلاک ها

برنامه نویسی با این بلوک ها

بلوک های برنامه نویسی در STEP

بلوک هایی که در برنامه نویسی توسط STEP 7 به کار می روند به سه دسته تقسیم می شوند.

۱-بلوک های منطقی (Logic Block)

این بلوک ها حاوی دستورات برنامه نویسی هستند که به آنها Code Blocks نیز می گویند و عبارتند از:

Organization Block	OB
Function Block	FB
Function	FC
System Function Block	SFB
System Function	SFC

۲- بلوک های داده (Data Block)

این بلوک ها همان طور که از نامشان پیداست ، حاوی داده می باشند . داده هایی که توسط بلوک های منطقی خوانده یا نوشته می شوند. انواع بلو ک های داده عبارتند از:

Data Block	DB
System Data Block	SDB

۳- توابع تعریف شده توسط کاربرد UDT (User Defined Data Type).

بلوک هایی هستند که حاوی داده بوده و بسته به نیاز همراه با بلو ک داده برای پرهیز از نوشتن داده های تکراری به کار می روند.

مقایسه بلوک های برنامه نویسی STEP7 با STEP 5

آنچه که به دست می آید این است که OB و DB در S5 و S ۷ یک مفهوم را دارند. ولی توابع FB ، PB و SB در S7 فقط به صورت FC شناخته می شوند. همچنین FB و SDB از بلوک های جدید STEP7 می باشد.

انواع بلوک های منطقی در S7

بلوک سازماندهی OB (Organization Block)

OB ها در واقع رابط بین سیستم عامل و برنامه ی کاربر هستند . این بلوک ها توسط سیستم عامل فراخوانده شده و برای مقاصدی چون کنترل سیکلی ، وقفه و راه اندازی به کار می روند. OB1 برای اجرای سیکلی برنامه ی کاربر استفاده می شود. برنامه در این بلوک

می تواند به صورت پشت سر هم (Liner) یا ساختار بندی شده (Structured) باشد. یعنی بخش های برنامه در بلوک های مختلفی نوشته شده که از بلوک ماقبل فراخوانی می شوند . در این روش لازم است توسط STEP7 ابتدا بلوک نهایی و سپس بلوک های ماقبل (مثلاً FB قبل از OB) ایجاد شود.

تابع FC(Functions)

FC ها بلوک های منطقی حاوی دستورات STEP7 هستند که حافظه ندارند . یعنی متغیر های موقت آنها در Local Data Stack ذخیره شده و وقتی اجرای FCتمام می شود . از بین می روند . لذا در صورتی که نیاز به ذخیره سازی داده های آنها باشد باید از نوع shared استفاده کرد. یکی از کاربرد های FC ها این است که جایگزین PB ها در S5 شده اند. به عبارت دیگر می توان برنامه را در FC نوشت و در OB آن را فراخوانی نمود.

بلوک های تابع FB (Functions Block)

FB ها بلوک های منطقی حاوی STEPV هستند که دارای حافظه می باشندو بلوک های داده ازنوع لحظه ای (Instance) را می توانند برای ذخیره سازی پارمترهایی که به آنها ارسال می شوند و متغییرهای استاتیک مورد استفاده قرار دهند .

نکته: SFB ها همانند FB ها دارای حافظه هستند ، ولی SFC ها مانند FC ها حافظه ندارند.

نکته: وقتی پروژه ای به PCیاPG ارسال (Download) می شود ممکن است هیچ SFB یا SFC در آن پوشه موجود نباشد ولی وقتی همان برنامه را از PLC به PC یا PG ، Upload می کنیم SFB و SFC های مختلفی را در پوشه بلوک ها می بینیم که توسط PLC به طور اتوماتیک ایجاد شده اند . محتوای بلوکهای SFC و SFB را نمی توان دید و پیغام The Block is protected ظاهر می شود. در شکل انواع بلوک های مورد استفاده در STEP7 را مشاهده می کنید.

بلوک های سیستمی System Function Block , System Function

SFB ها و SFC ها بلوک هایی هستند که از قبل توسط سازنده برای مقاصد خاصی تهیه شده اند مانند :

- بازبینی یا چک کردن برنامه

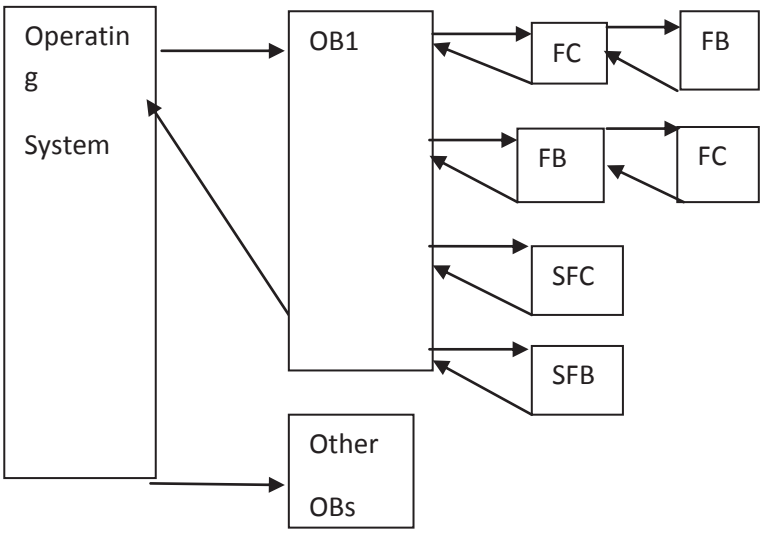
- ساعت سیستم و Run Time Meter

- Error Handling ، به حداقل رساندن احتمال وقوع خطا

-Interrupt Handling ، به حداقل رساندن احتمال وقفه

-Distributed I/O ، ورودی / خروجی توزیع شده

–آدرس دهی ماژول ها



بلوک های داده DB(Data Blocks)

بلوک های داده بر خلاف OB ، FB ، FC حاوی دستورات STEP7 نیستند ، بلکه برای ذخیره سازی داده ها به کار می روند . لذا می توان یک بلوک داده را Write Protect (محافظت در برابر نوشتن) کرد تا PLC در حین اجرای برنامه نتواند داده های آن را عوض کند این کار با کلیک راست روی DB و انتخاب گزینه ی Properties (مشخصات) امکان پذیر است.

بلوک های داده بر دو نوع هستند:

۱-بلوک های داده ی اشتراکی یا سراسر (Shared DB)

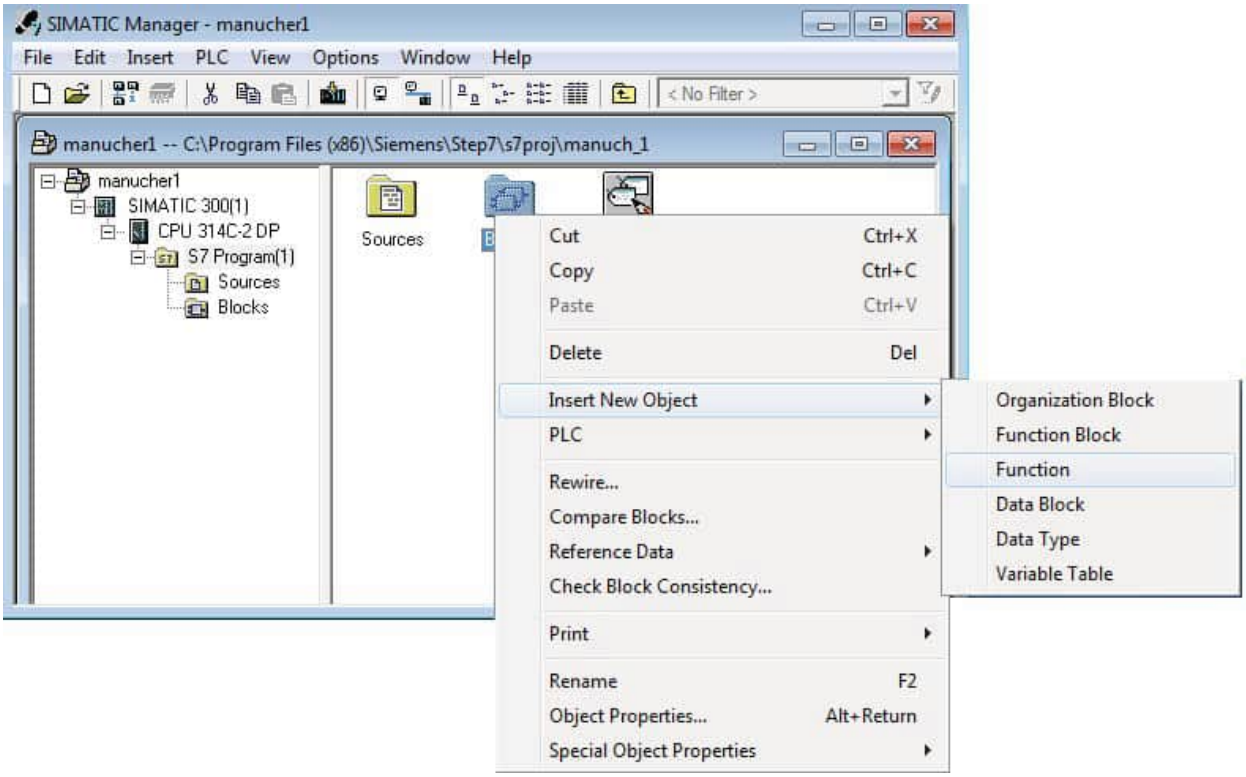
بلوک های داده ی اشتراکی همان طور که از نامشان پیداست ، به صورت اشتراکی بوده و برای تمامی بلوک های برنامه قابل دسترسی هستند. داده های آنها با بسته شدن بلوک داده از بین نمی رود .

۲-بلوک داده ی اختصاصی (Instance DB)

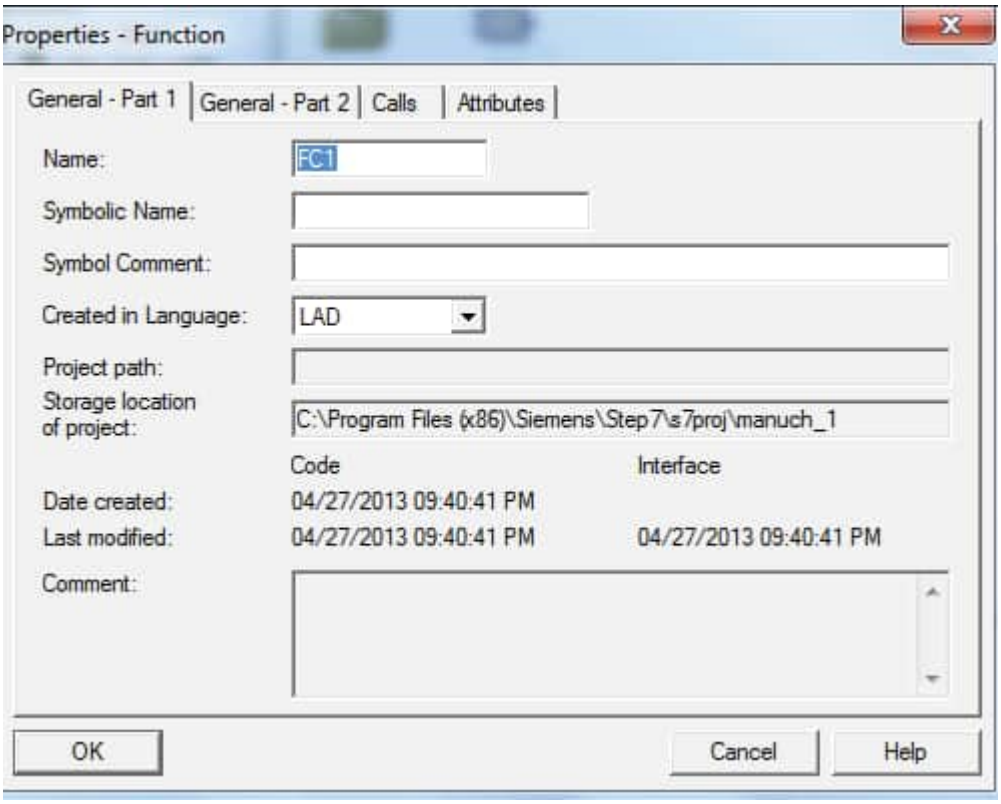
بلوک های اختصاصی به عنوان حافظه ای برای FB ها منظور شده اند. پارامتر هایی که به FB ها ارسال شده و متغیر های استاتیک در این DB ها ذخیره می شوند و وقتی اجرای FB کامل می شود از بین نمی روند.

روش ایجاد و استفاده از تابع FC

در منوی SIMATIC Manager راست کلیک کرده و از گزینه ی Insert New Object گزینه ی Function را انتخاب و نام FC1 را تایپ می کنیم. در شکل ۱۱.۵ و ۶.۱۱ مراحل ایجاد FC را مشاهده می کنید.

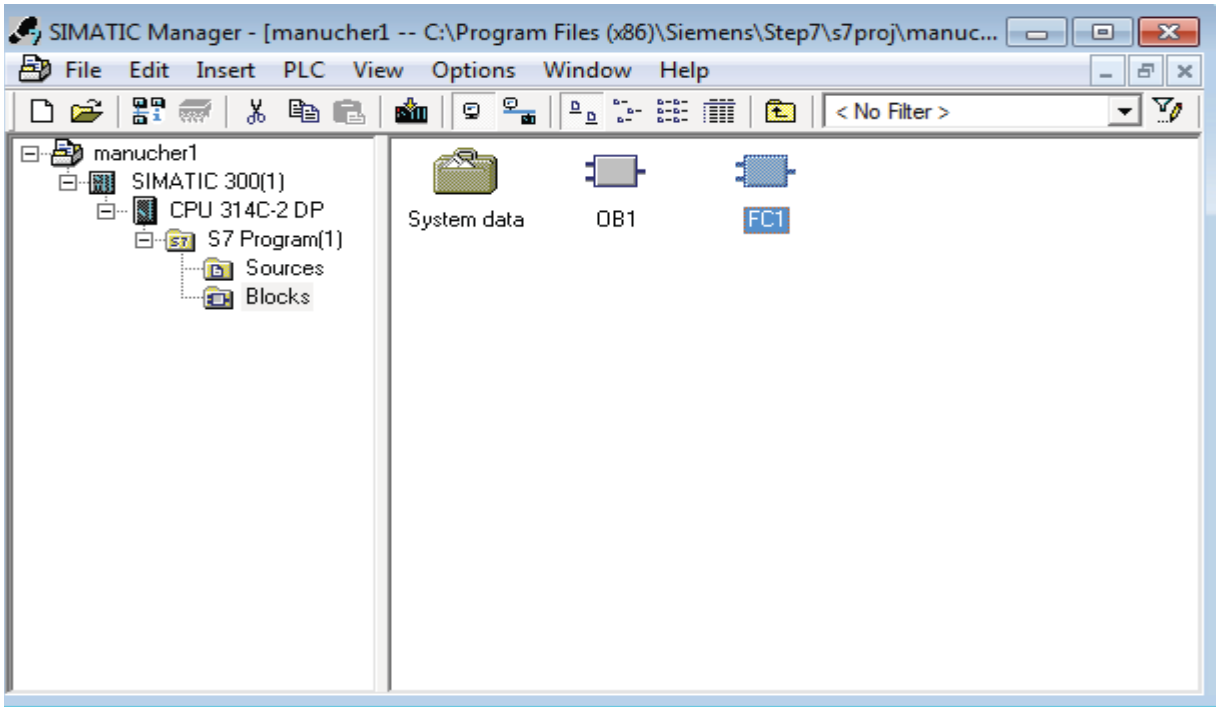


شکل ۱۱.۵



شکل ۶.۱۱

به صورت پیش فرض نام FC1 بر روی این Function گذاشته می شود ، می توانید نامی دیگر و به دلخواه متناسب با پروژه ی مورد نظر بر روی آن بگذارید. سپس بر روی ok کلیک کنید.

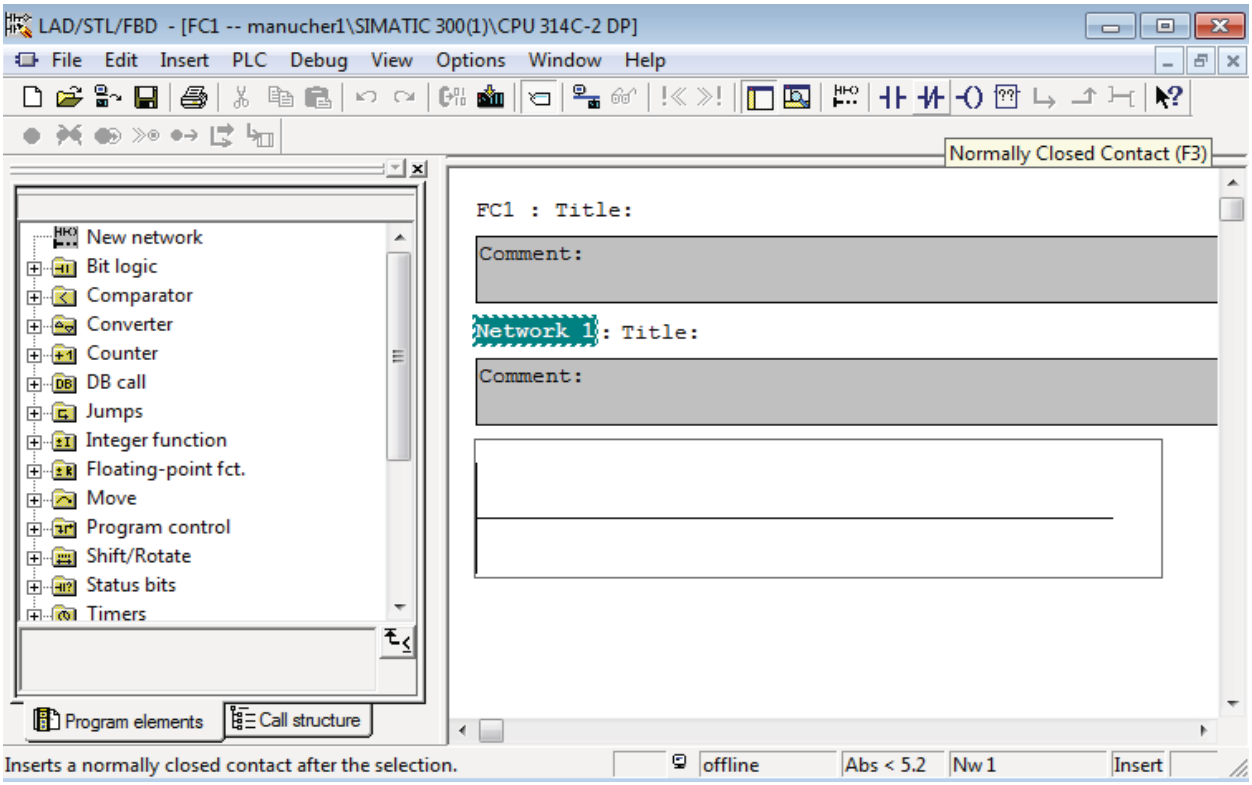


شکل ۷-۱۱

FC1 ایجاد شد حال برای نوشتن برنامه به روش دلخواه با دو بار کلیک بر روی آن می توانید برنامه ی مورد نظر را نوشته و از طریق کلید ابزار Save& Compile نمائید و آن را به PLC و یا شبیه ساز ارسال کنید.

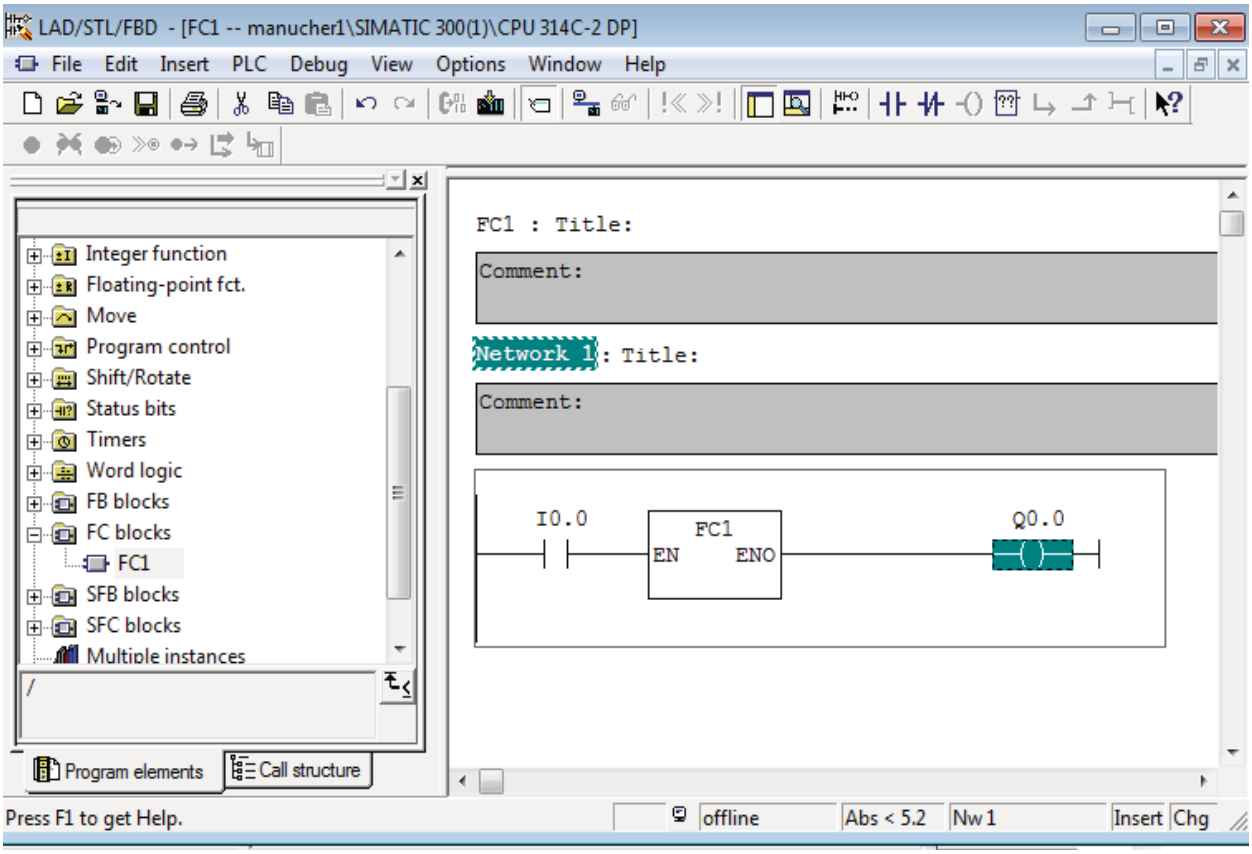
نکته: پس از نوشتن هر برنامه از جمله FC حتماً بایستی آن را ابتدا ذخیره (Save) و سپس ارسال یا(Download) کنیدوسپس به نوشتنOB1اقدام کنیددرصورتی که در OB1 از**FC1** استفاده شده باشدولی آن را ارسال یا(Download) نکرده باشیددر PLC چراغ سیگنال SF روشن وبرنامه OB1 اجرانخواهدشد.

پس از نوشتن برنامه FC1 و ذخیره و ارسال (Download) آن با دو بار کلیک بر روی OB1 وارد آن شوید. (شکل ۸.۱۱).



شکل ۸.۱۱

حال در پنجره ی سمت چپ Program Element و در شاخه ی FC blocks می توانید FC ای را که ساخته اید ، ببینید. اکنون می توانید با انتخاب Network مورد نظر در OB1 که می خواهید FC1 در آن قرار گیرد را انتخاب و سپس توسط ماوس FC1 را به داخل آن Network بکشید. (Drag) کنید ، و یا دو بار بر روی FC1 کلیک کنید. (شکل ۹.۱۱).

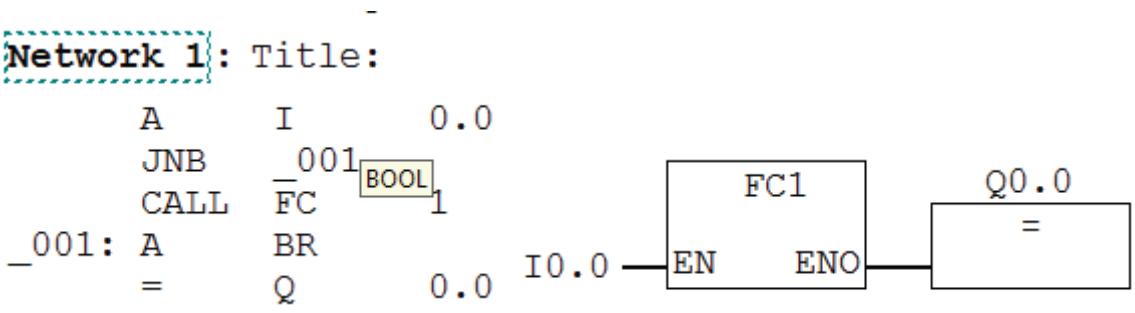


شکل ۹.۱۱

می توان در OB1 به ورودی و خروجی FCI یعنی ENO آدرس دهی نمود. در این صورت FC1 به وضعیت ورودی EN در هنگام کار توجه خواهد کرد.

نمایش FCL به روش STL و FBD به صورت زیر می باشد

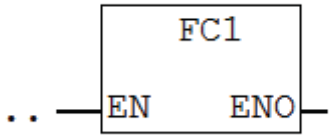
.FC1



در صورتی که ورودی و خروجی آدرس دهی نشود به صورت زیر خواهد بود .

$FC1$

```
Network 1: Title:
      CALL  FC     1
      NOP   0
```



در هنگام استفاده از FC به نکات زیر توجه کنید.

۱-در نرم افزار STEP7 تابع Function با دستور Cll فراخوانی می شود که اگر به EN آدرس داده نشود معادل دستور JU در برنامه نویسی STEP 5 می باشد و در صورتی که به EN آدرس داده شودمعادل دستور JC در برنامه نویسی STEP 5 می باشد(در STEP 7 با عبارت UC غیر شرطی نمایان می شود).

۲- ورودی EN به معنی Enable می باشد و به صورت پیش فرض تا زمانی که آدرس داده نشود یک می باشد و ورودی را به خروجی انتقال می دهد . در صورت دادن آدرس به وضعیت آدرس داده شده توجه می کند . در صورتی که آدرس یک باشد ورودی را به خروجی انتقال می دهد و در غیر این صورت یعنی صفر بودن آدرس ورودی ، ورودی به خروجی انتقال نمی یابد . به عبارت دیگر FC1 فعال نمی شود.

۳- FC1 اگر به صورت شرطی فراخوانی شود ، بعد از اجرای اول ممکن است FC دیگر از شرط تبعیت نکندو آن به دلیل فعال ماندن FC است. باید به گونه ای آن را غیر فعال نمود. یک روش آن است که شرط را در داخل خود FC تعریف کنیم و روش دیگر به موازات شرط کلیدی معرفی کنیم تا FC را غیر فعال نماید.

۴- خروجی ENO نشان دهنده ی درستی محاسبات می باشد . یعنی اگر ENO=O یعنی محاسبات غلط و اگر ENO=1 ، یعنی محاسبات صحیح می باشد.

۵- در روش STL عبارت A BR به معنی AND Bit Register بوده ، این عبارت یک ثبات به نام BR را فراخوانی نموده و نتیجه ی حاصل را در خروجی نمایان می کند.

۶- در روش STL در صورتی که ورودی و خروجی FC آدرس دهی شده باشد عبارت JNB_001 و عبارت A BR:001__ را مشاهده می کنید که 001__ بر چسب (Label) می باشد که به صورت چهار کاراکتری که کاراکتر اول نمی تواند عدد باشد ، همچنین نمی تواند کاراکتر های /و * و-+ باشد ، نمایان می شود و JNB از دستورات پرش به معنی JUMP NOT Bit یعنی اگر RLO=Oپرش به بر چسب صورت می گیرد ولی خطوط وسط اجرا نمی شود.

(بنابراین اگر RLO=1 باشد پرشی صورت نگرفته و خطوط وسط اجرا می شود .)

```
Network 1: Title:
      A     I      0.0
      JNB   _001 BOOL
      CALL  FC     1
_001: A     BR
      =     Q      0.0
```

مثال های کاربردی از FC

مثال ۱.۱۱

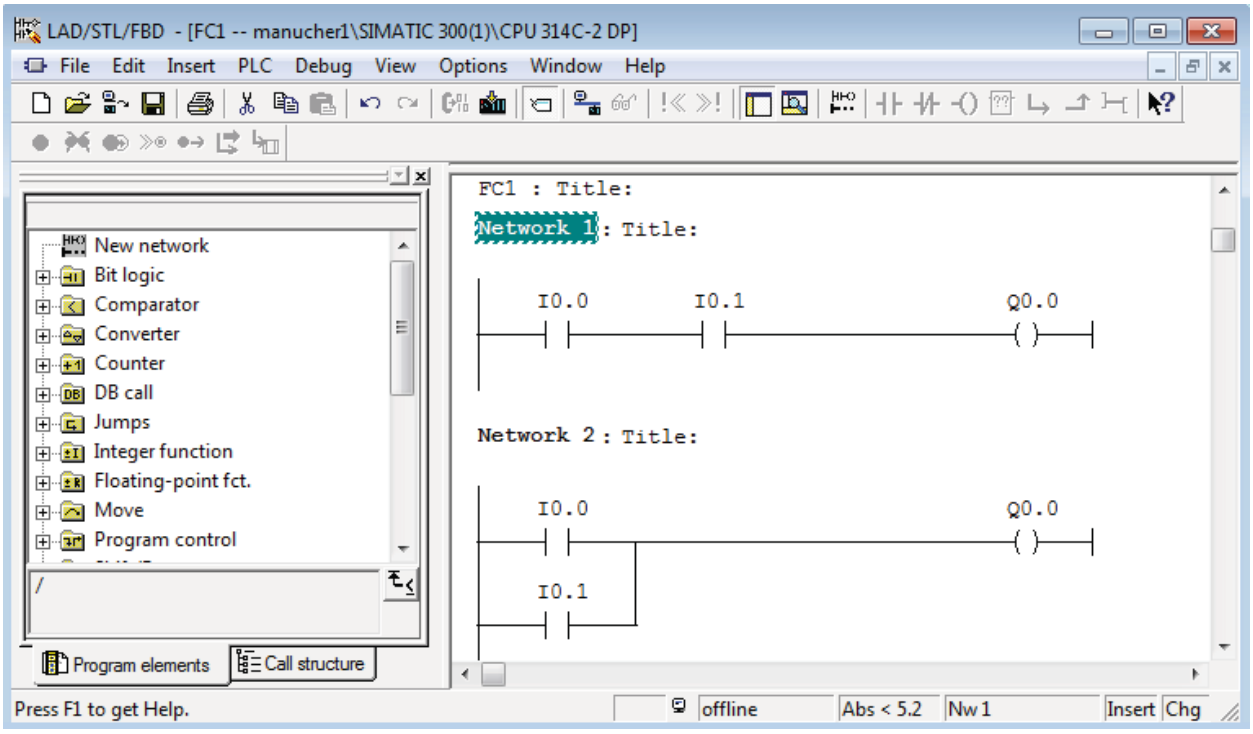
برنامه ای بنویسید که با کلید های I0.0 و I0.1 و خروجی Q0.0 هم عمل AND و هم عمل OR را انجام دهد.

حل :

چون آدرس های ورودی و خروجی هر دو در عمل AND و OR تکراری می باشد عملاً نمی توانیم د ر OB1به مانند زیر برنامه نویسی کنیم زیرا این وضعیت در صنعت خطرناک بوده و ممکن است :

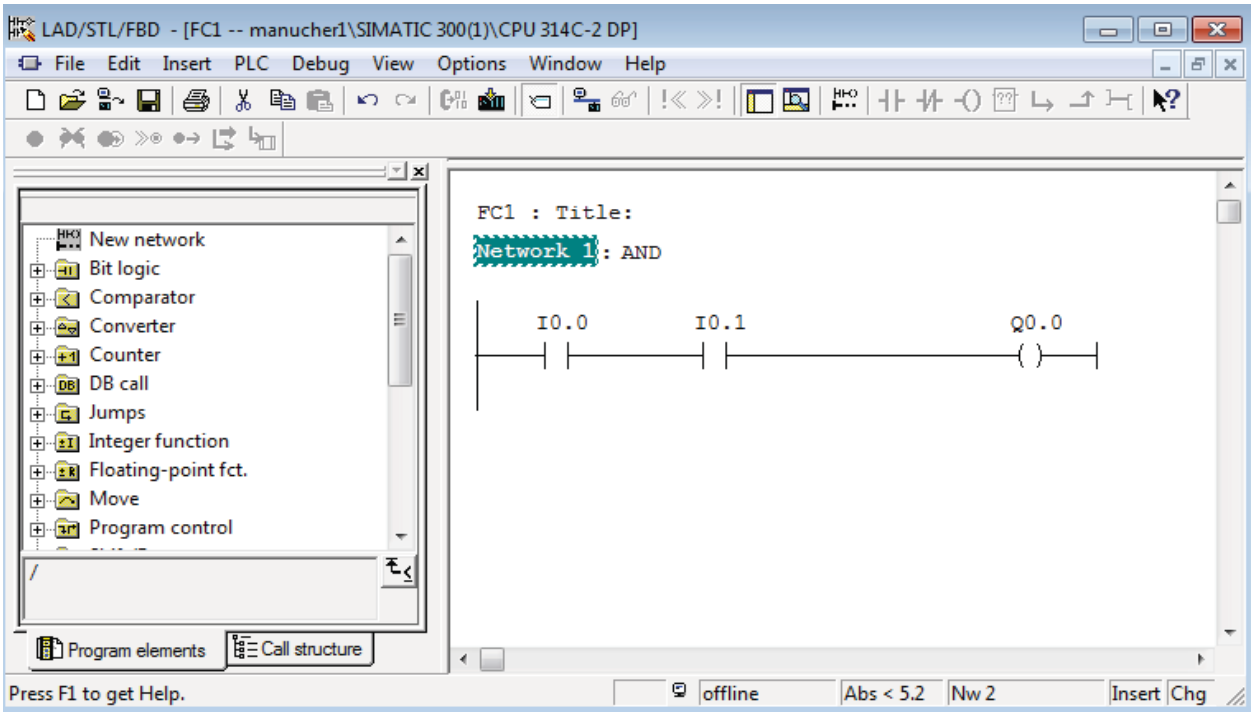
۱-PLC لامپ سیگنال SF را روشن و از اجرای OB1 خودداری کند.

۲-با توجه به اولویت AND نسبت به OR در محاسبات ریاضی فقط اجازه ی اجرای Network۱ را داده و Network2 اجرا نخواهد شد.

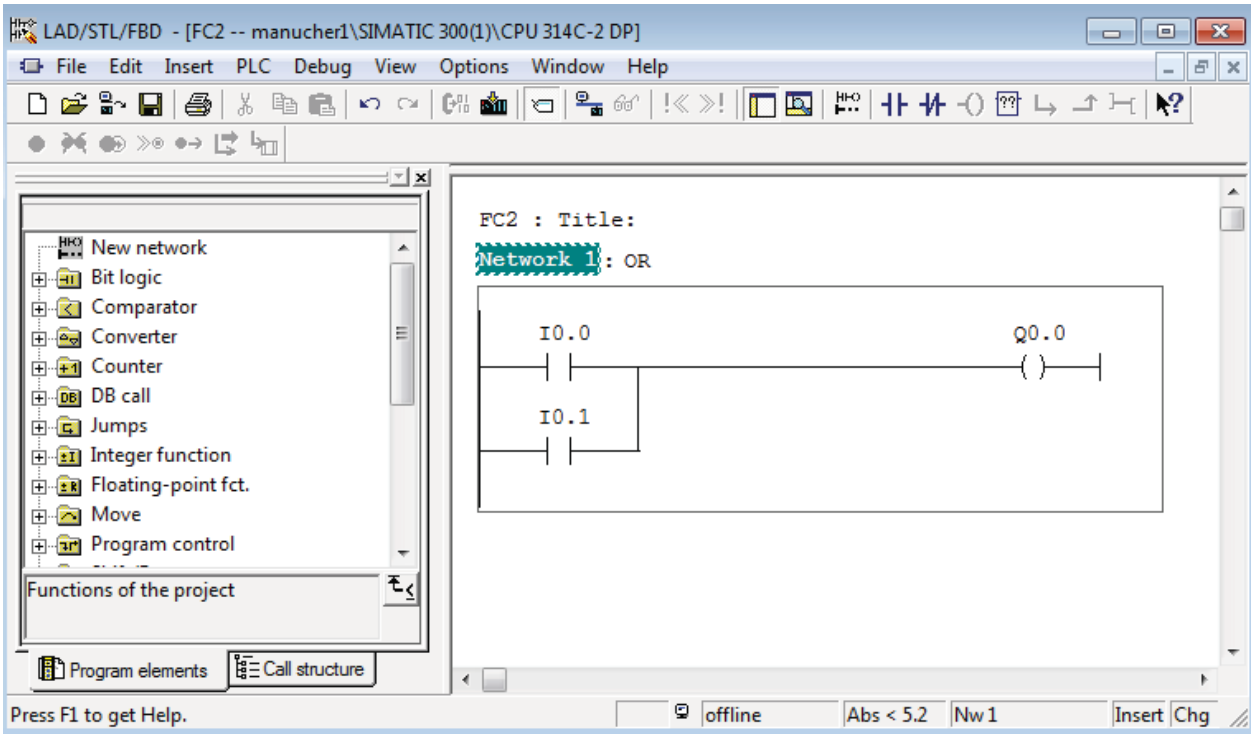


شکل ۱۰.۱۱

برای حل مسئله مطابق شکل های ۱۱.۱۱ و ۱۲.۱۱ ، قسمت AND را در FC1 و قسمت OR را در FC2 نوشته ، ذخیره و ارسال می کنیم.



شکل ۱۱.۱۱

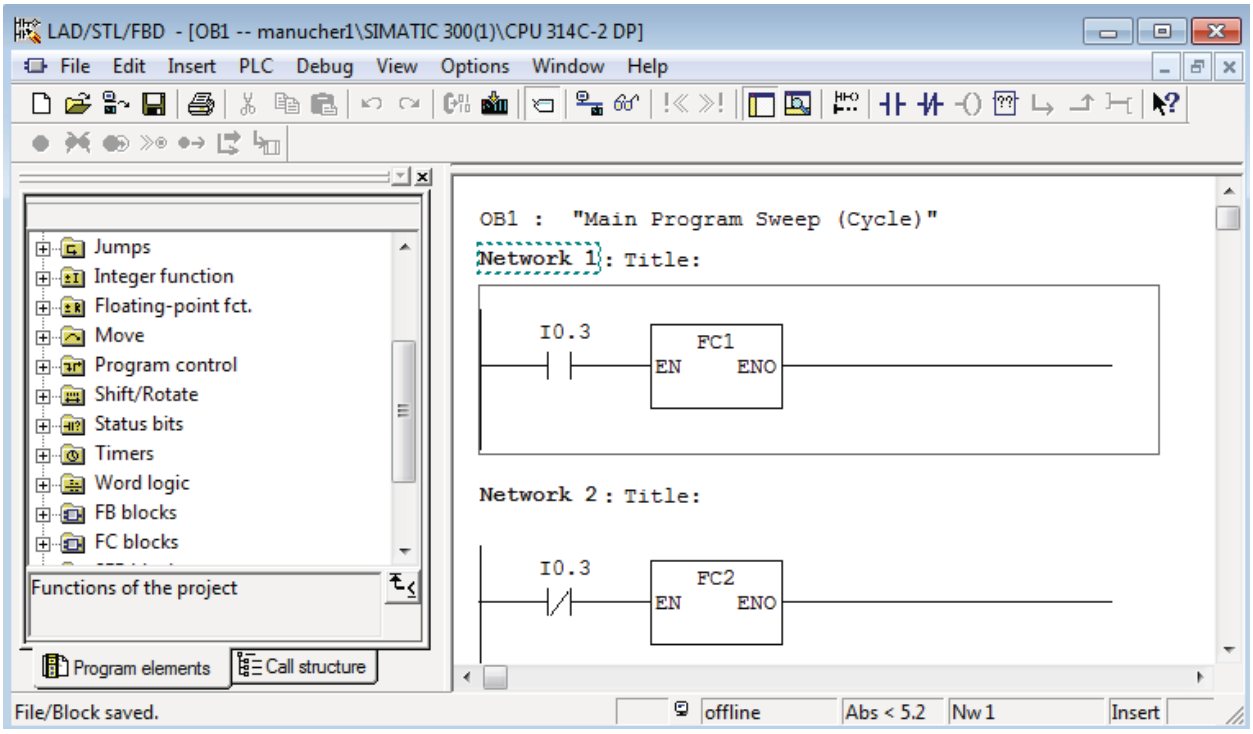


شکل ۱۲.۱۱

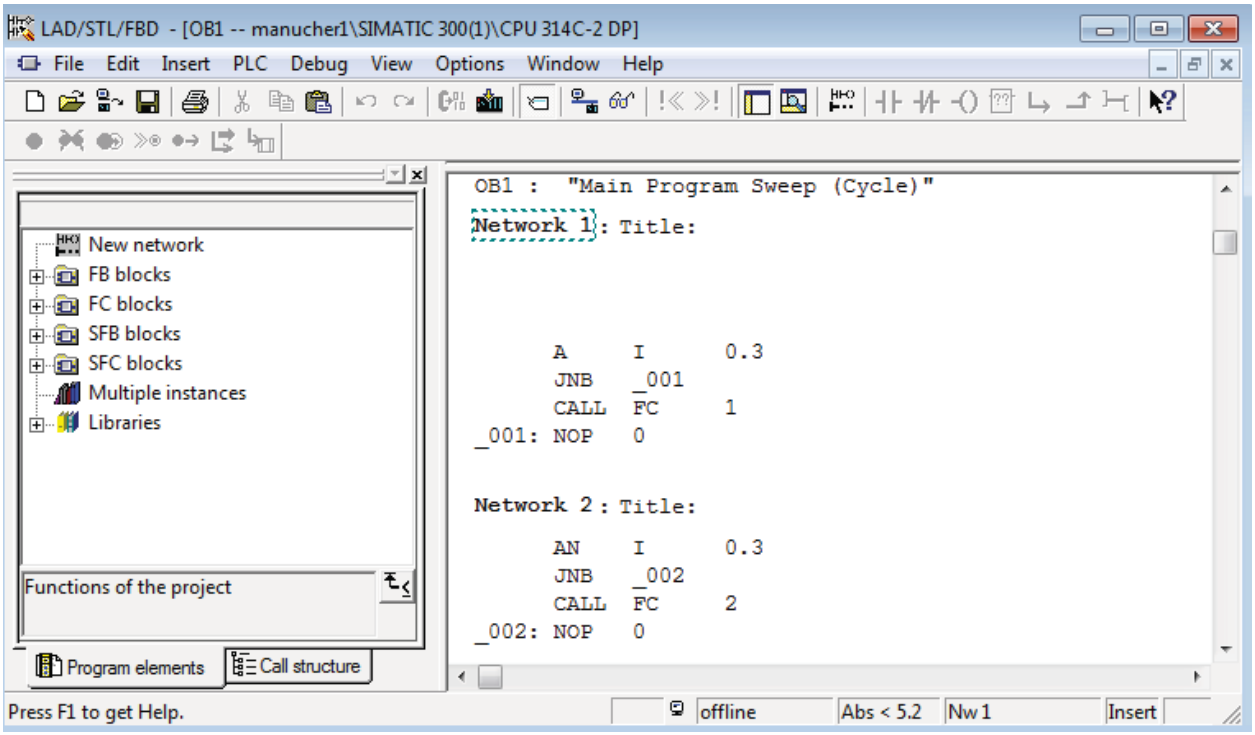
حال در بلوک OB1 با استفاده از کلیدی مانند I0.3 که در FC1وFC2استفاده نشده باشد به عنوان شرط می توانیم اجرای هردو FCراوابسته به وجود و یاعدم وجود I0.3کنیم .با یک شدن FC1، 10.3 و با صفرشدن I0.3 ،آنگاه FC2اجراخواهد شد و هیچ گاه دو FCباهم اجرانخواهد شد.

نکته : می توانیمFC ها را مانند آنچه گفته شد هریک را در یک Network با ماوس کشیده و قرار دهیم و سپس آدرس ورودی I0.3 را در ورودی EN هر یک به صورت بازو بسته قراردهیم . با توجه به اشکال زیر می توانیم با استفاد ازدستور Call از پوشه ی program control سمت چپ صفحه استفاده کنیم .تفاوت این دو روش را می توان در تبدیل برنامه به STL مشاهده کرد که روش اول ازLabel استفاده می شود و در روش دوم ازUC و CC استفاده می شود.

راه حل اول:

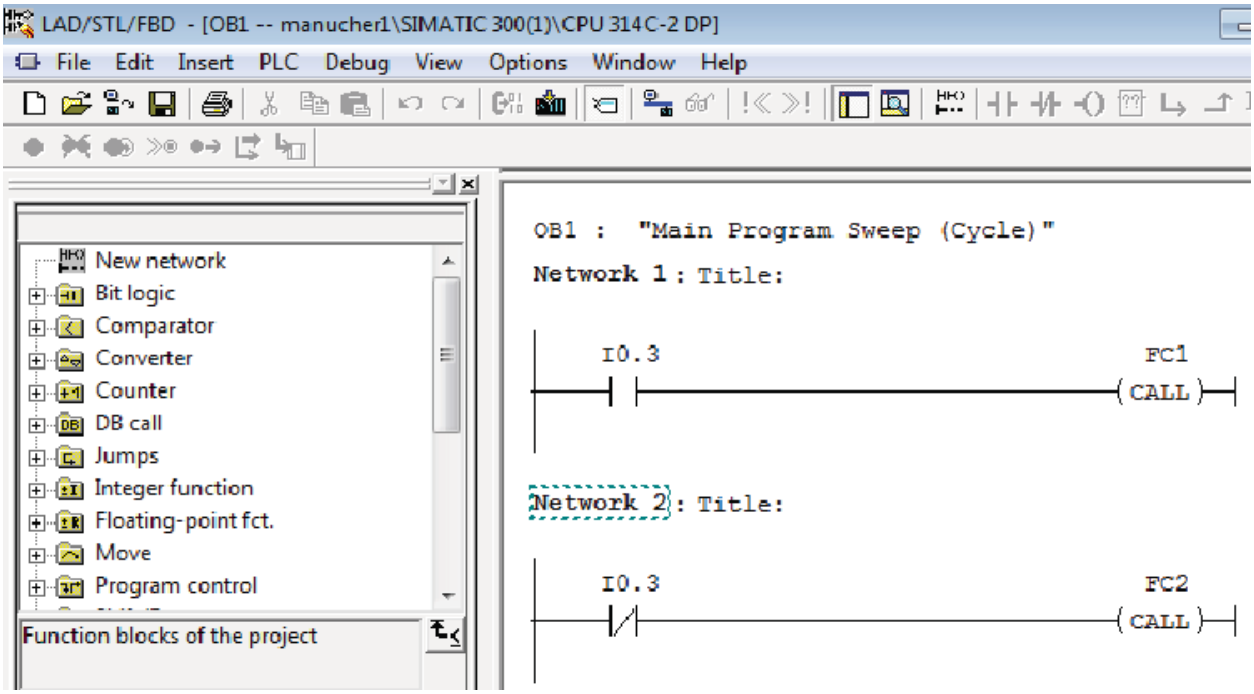


شکل ۱۳.۱۱

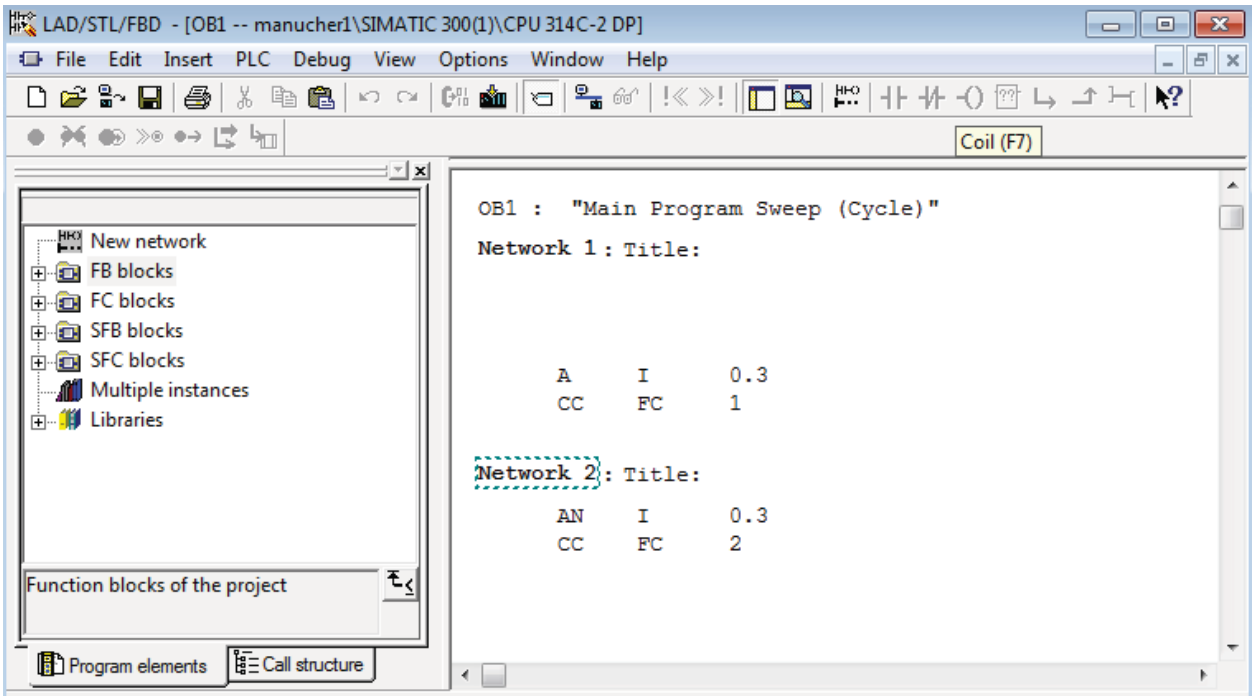


شکل ۱۴.۱۱

راه حل دوم :



شکل ۱۵.۱۱



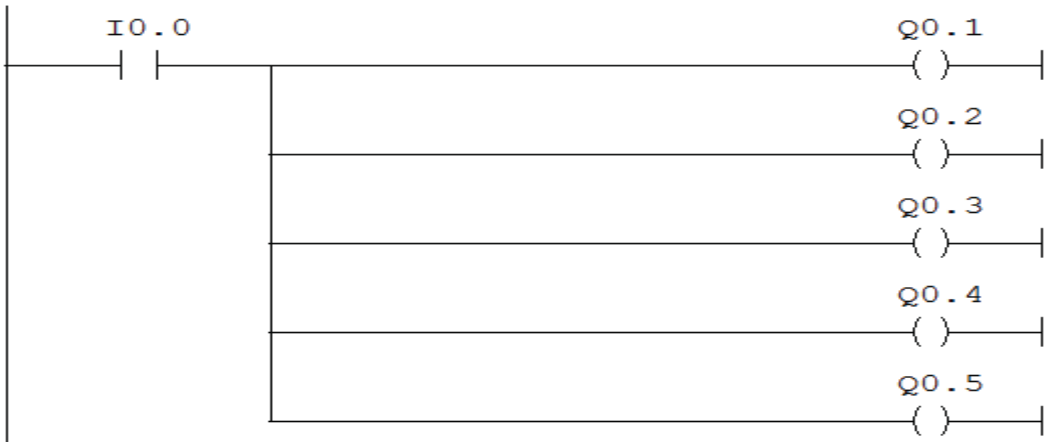
شکل ۱۶.۱۱

مثال ۲.۱۱

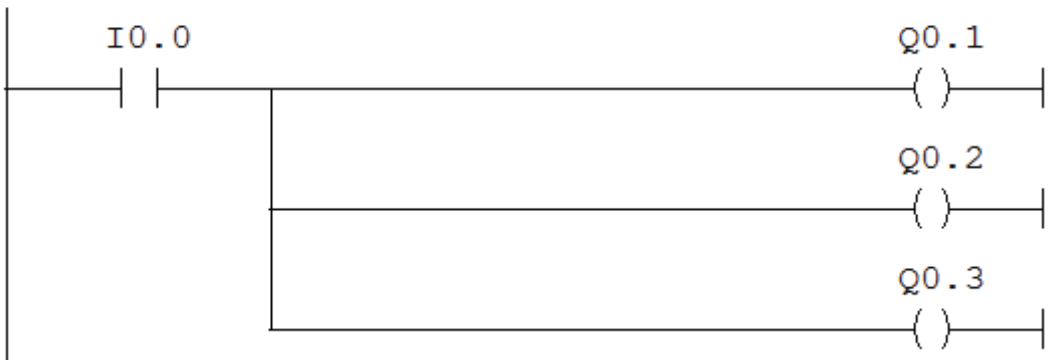
یک سالن بزرگ دارای پنج لامپ H1,H2, H3, H4, H5 و یک کلید ON/OFF می باشد. برای صرفه جویی درمصرف انرژی در نظر است با تحرک کلید ON/OFF اگر روز بود سه لامپ از پنج لامپ روشن شودو اگر شب بود هر پنج لامپ روشن شود . با استفاده از FC برنامه مورد نظر را بنویسید.(تشخیص روز و شب با استفاده از یک فتوسل انجام می گیرد .)

حل:

```
FC1 : Night I0.0 start (on/off)
Network 1: Title:
```




```
FC2 : DAY  i0.0 start(on/off)
Network 1: Title:
```



OB1 : "Main Program Sweep (Cycle)"

Network 1: I0.1 Sensor/Night

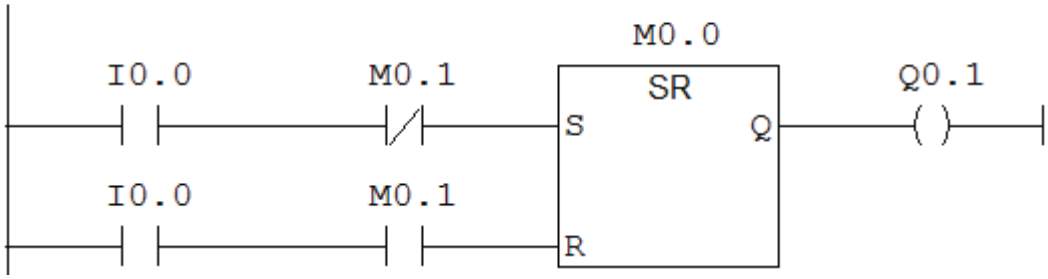
Network 2 : I0.1 Sensor /DAY

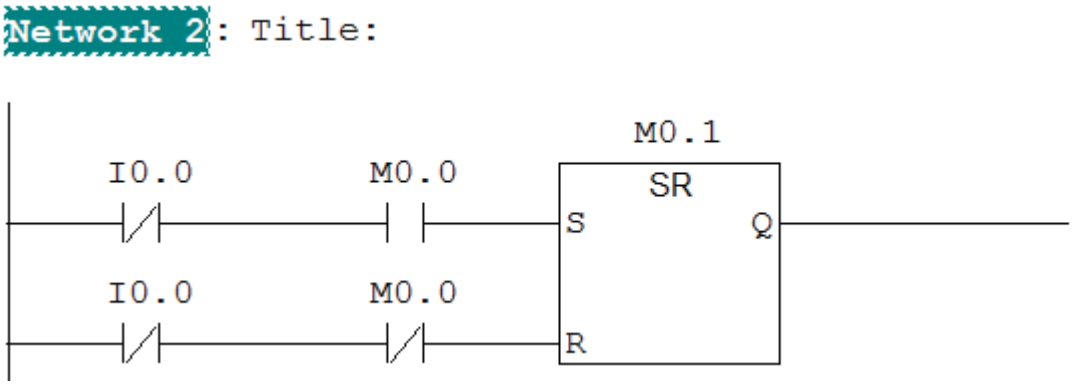
مثال ۳.۱۲

با استفاده از FC برنامه ای بنویسید که با زدن I0.0 برای بار اول Q0.1 روشن و با زدن I0.0برای بار دوم Q0.1 خاموش شود.

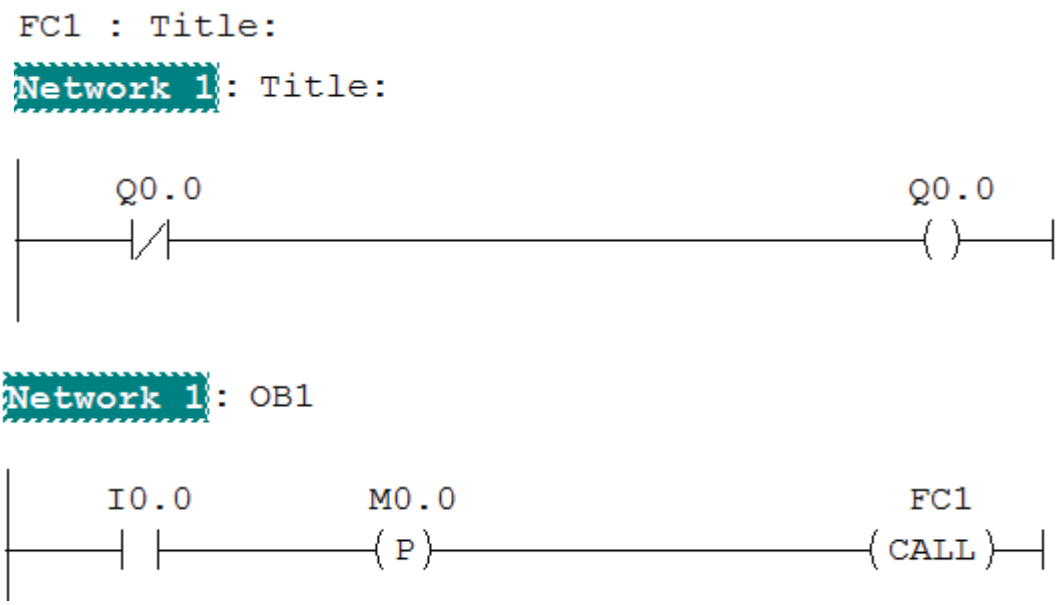
یک راه حل پیشنهادی بدون استفاده از FC

Network 1 : Title:





در اینجا قصد داریم برنامه را با استفاده از FC بنویسیم. مشاهده خواهید کرد که برنامه چقدر ساده و جالب خواهد شد . برای این منظور ابتدا FC1 را نوشته و Save کنید. سپس OB1 را به صورت زیر نوشته و پس از ارسال برنامه درستی عملکرد آن را بررسی کنید .



مثال ۴.۱۱

همچنان که می دانید چراغ های چهار راه راهنمایی رانندگی در روز به صورت زمان دار و در شب به صورت چشمک زن می باشد . برنامه را با استفاده از FC ها به صورتی بنویسید که با تحریک کلید I0.2چراغ های چهار راه به صورت زمان دار و با قطع کلید I0.2چراغ های همان چهار راه با همان ورودی وخروجی ها به صورت چشمک زن تبدیل شود .

-مرحله اول: چراغ سبز اصلی به همراه قرمز فرعی به مدت ۲۰ ثانیه روشن شود.

-مرحله دوم: چراغ زرد اصلی به همراه قرمز به مدت ۵ثانیه روشن شود.

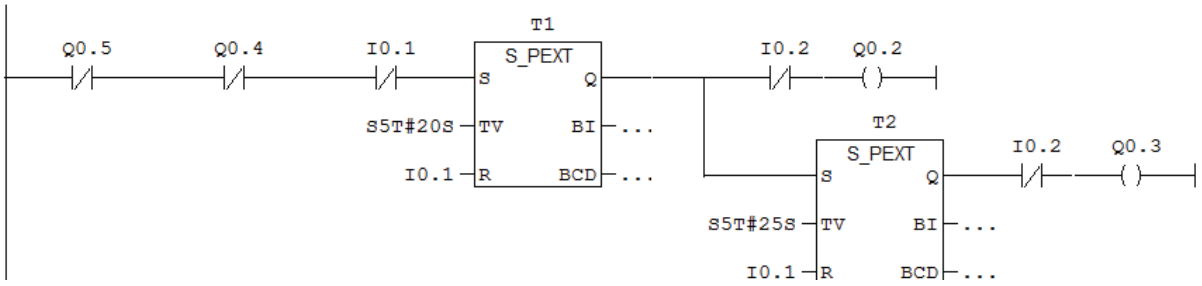
مرحله سوم: چراغ قرمز اصلی به همراه سبز فرعی به مدت ۲۰ ثانیه روشن شود.

مرحله چهارم: چراغ قرمز اصلی به همراه زرد فرعی به مدت ۵ ثانیه روشن شود.

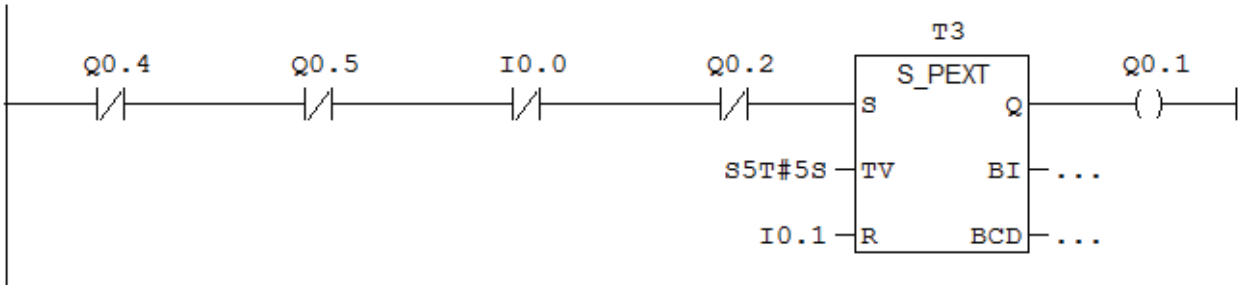
بازگشت به مرحله اول

FC1 : DAY Program

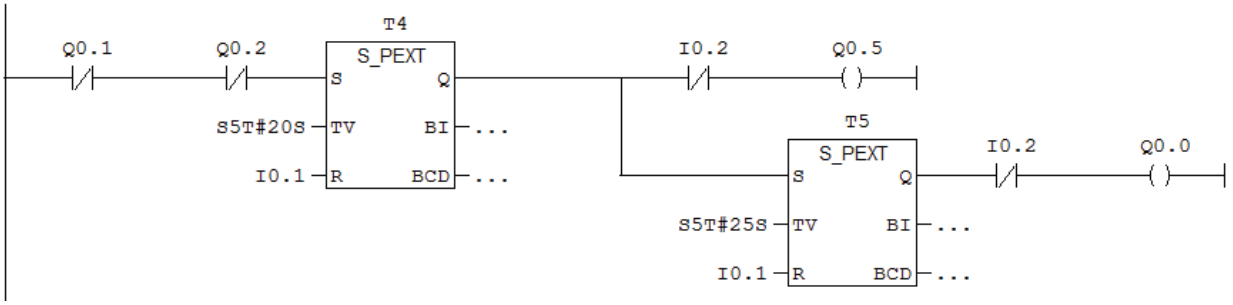
Network 1: Q0.2#GREEN 1(ON20sec)Q0.3#RED2(ON Sec)



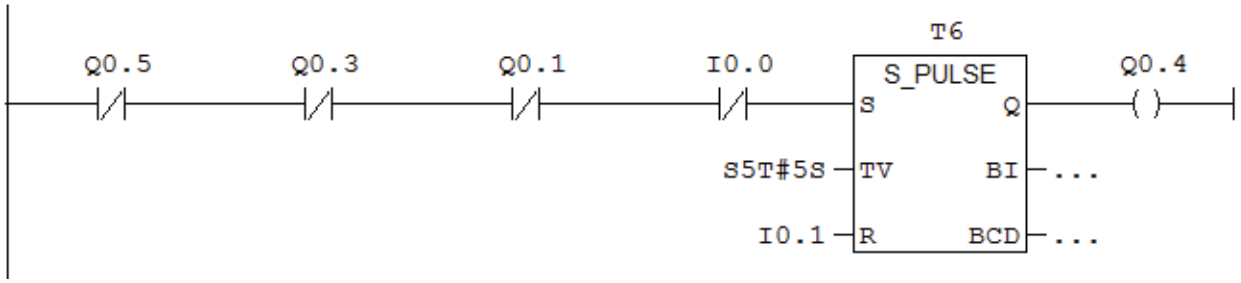
Network 2: q0.1 #YELLOW 1(ON/ 5Sec)



Network 3: Q0.5 #GREEN 2(ON 20 sec)Q0.0#RED 1(ON 25Sec)

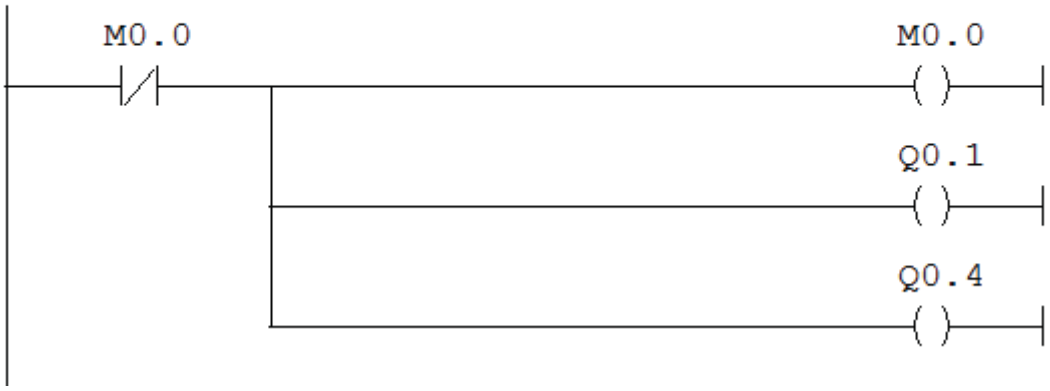


Network 4: Q0.4#YELLOW 2(ON 5Sec)



FC2 : NEGHT PROGRAM

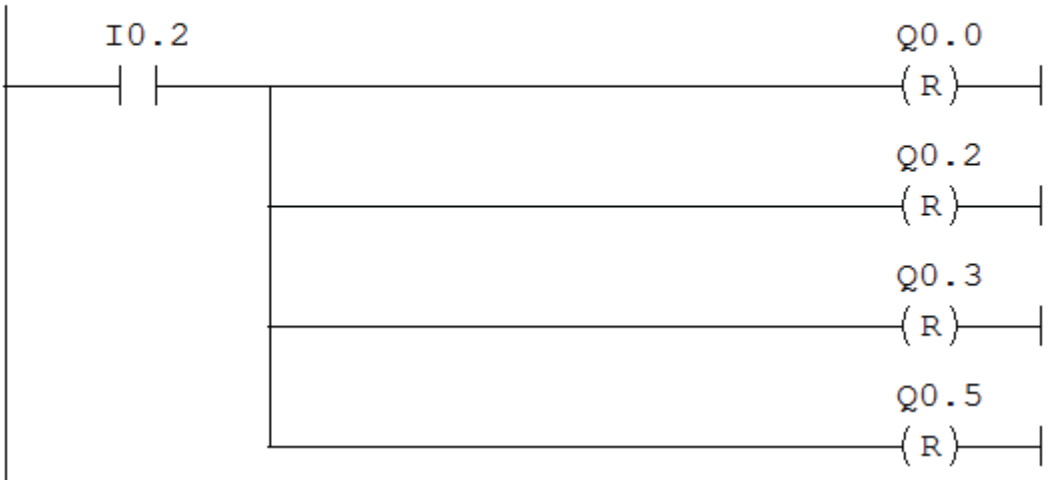
Network 1: Q0.1#YELLOW 1 Q0.4#YELLOW 2



OB1 : "Main Program Sweep (Cycle)"

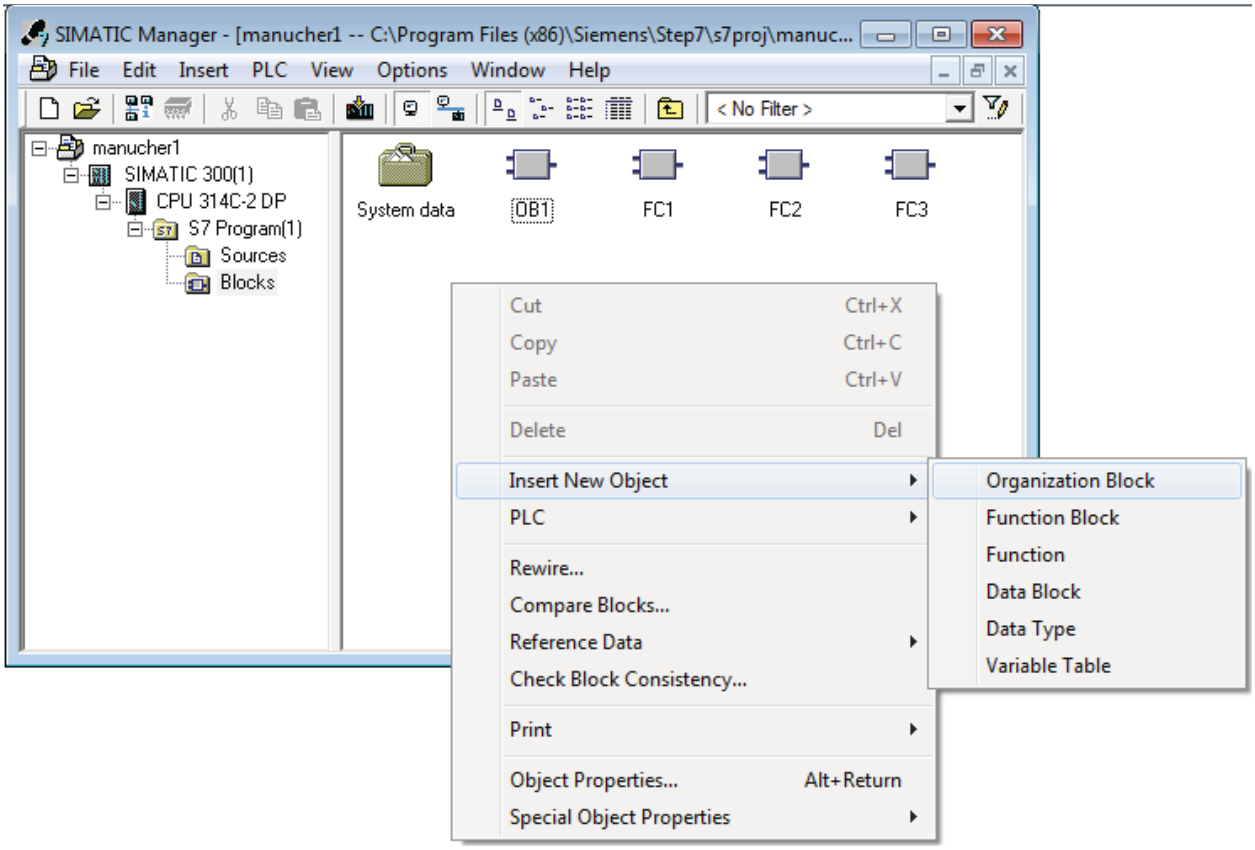
Network 1: Title:

Network 2: Title:

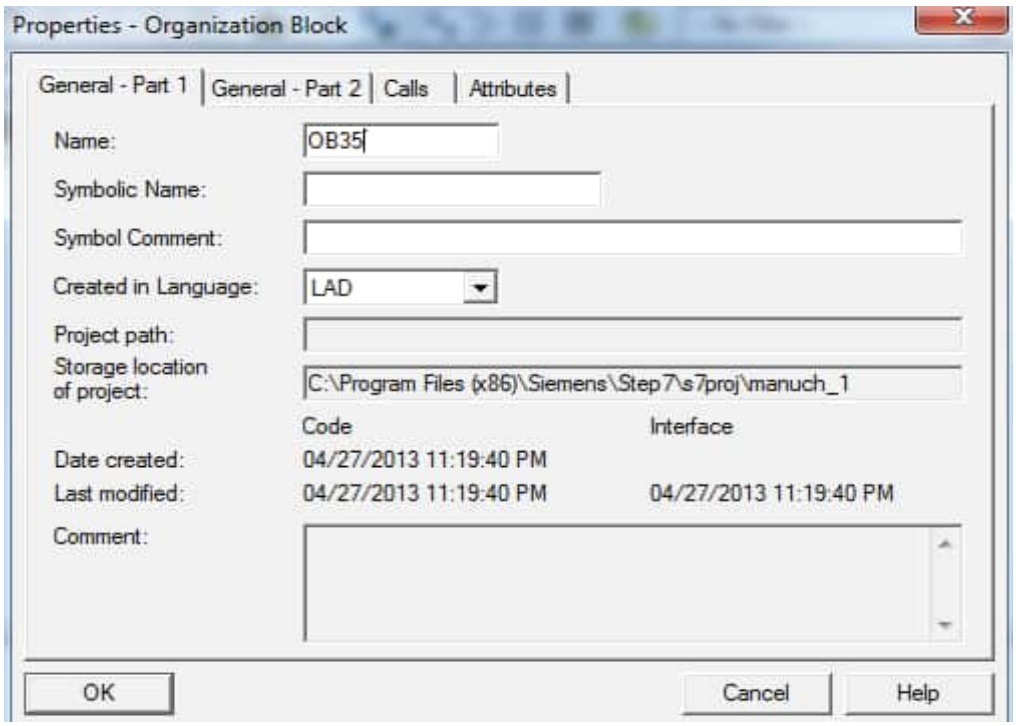


Network 3: Title:

برای درست کردن چراغ چشمک زن می توان از OB 35 به صورت زیر استفاده کرد. روش ایجاد OB35 به صورت زیرمیباشد.

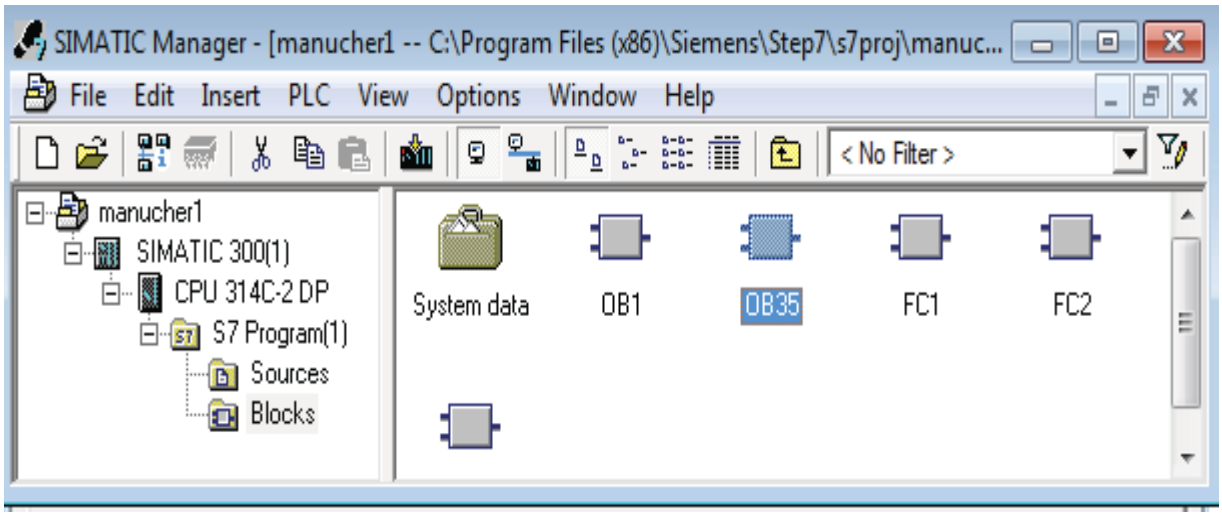


شکل ۱۹.۱۱



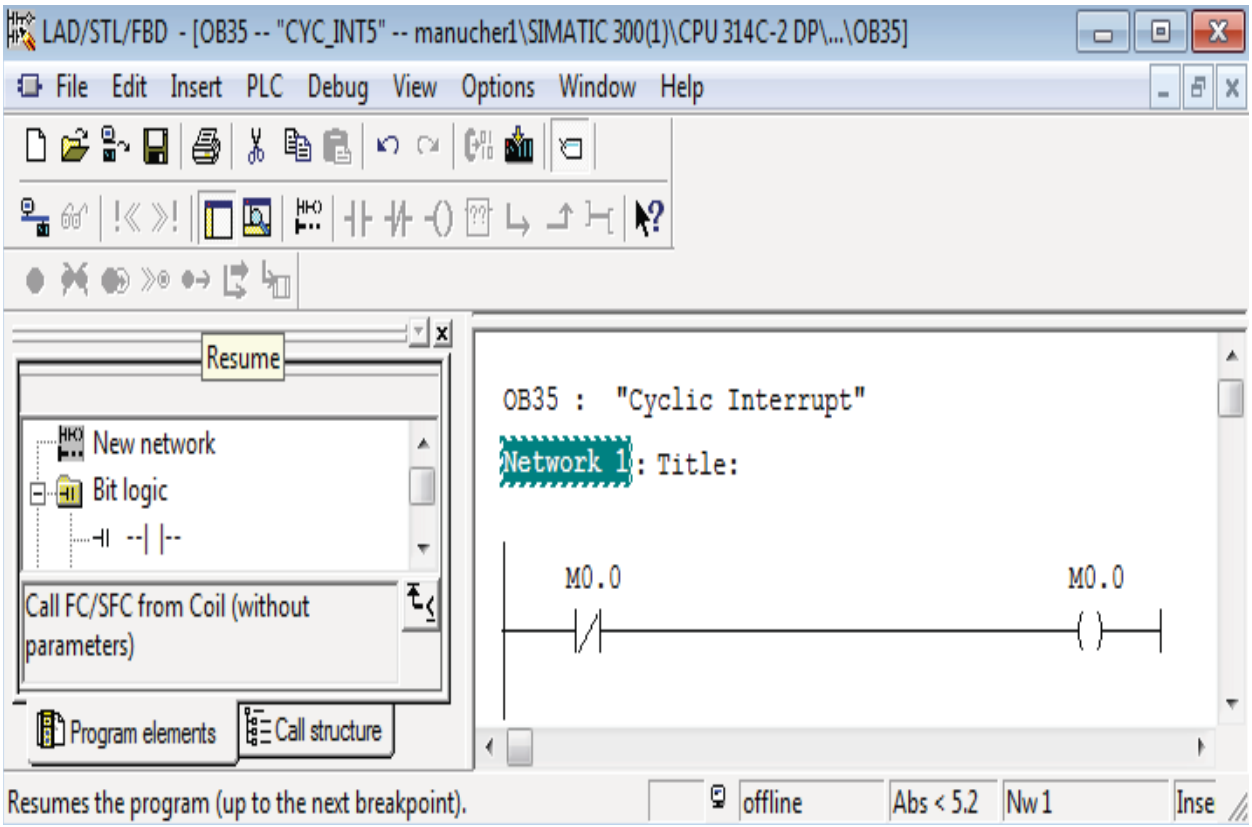
شکل ۲۰.۱۱

با دو بار کلیک بر روی OK بلوک برنامه ی OB35 به منوی SIMATIC Manager افزوده می شود.



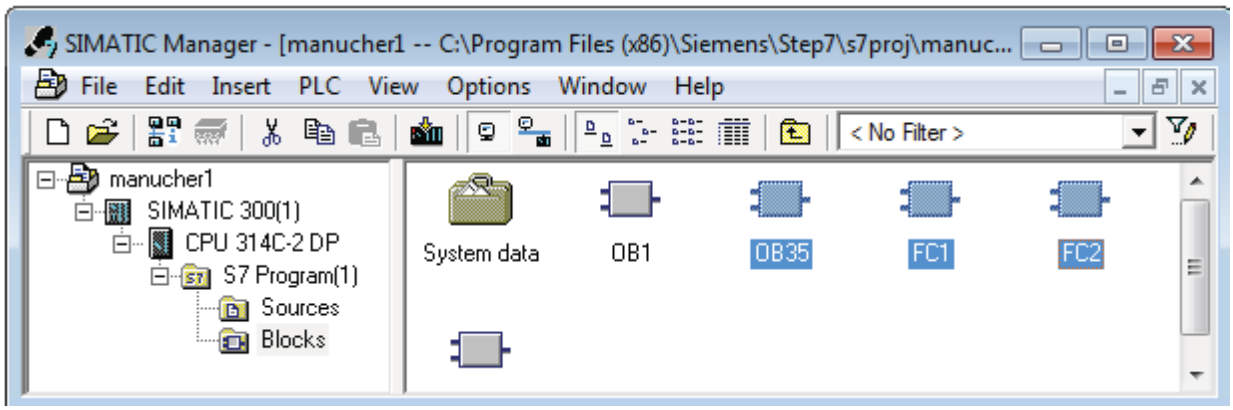
شکل ۲۱.۱۱

برنامه زیر را در Network ۱ نوشته و آن را ذخیره(Save) نموده و به PLC ارسال کنید.



شکل ۲۲.۱۱

با اجرای برنامه ی OB 35 مشاهده می شود که ، خروجی M0.0 به طور متناوب روشن ، خاموش می شود.



شکل ۲۳.۱۱

پس از نوشتن OB 35, FC2 ,FC1 آنها را ذخیره (Save) نموده و به PLC ارسال کنید. سپس OB 1 را ارسال و در وضعیت Monitor اجرای برنامه را مشاهده کنید.

روش ایجاد و استفاده از FC پارامتری

زمانی که بخواهیم با ورودی و خروجی های یکسان اعمال مختلفی را انجام دهیم ، در صورت استفاده از FC بدون پارامتر ممکن است سیستم و برنامه بر اثر تداخل دچار مشکل شده و اجرای برنامه متوقف می شود. لذا برای حل مشکل می توانیم با استفاده از FC های پارامتری دقت کار را افزایش دهیم. در مواردی که می خواهیم با ورودی و خروجی های متفاوت یک عمل مشخصی را انجام دهیم ، مثلاً در یک پروژه نیاز به تعداد ۱۰ برنامه چپ گرد / راست گرد و یا یکی پس از دیگری داریم . به جای آن که ۱۰ بار این برنامه را بنویسیم با استفاده از FC پارامتری به شرح زیر یک بار برنامه و پارامترهای آن را در قالب FC پارامتری تعریف نموده سپس با فراخوانی آن FC در OB1 و در ده Network با تغییر آدرس ورودی و خروجی از همان برنامه استفاده می کنیم. برای روشن شدن موضوع در قالب یک مثال روش ایجاد FC پارامتری را توضیح می دهیم.

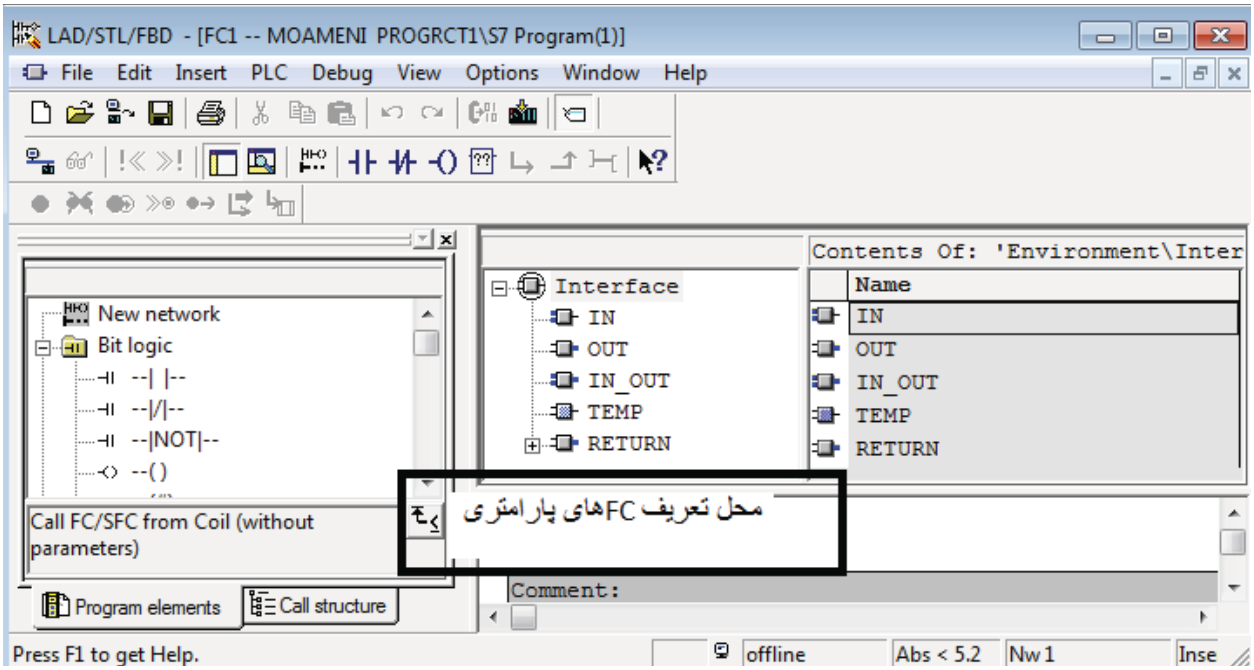
مثال ۵.۱۱

برنامه ای بنویسید که PLC سه ورودی I0.0وI0.1وI0.2را دریافت و اعمال زیر را بر روی آنها انجام داده و نتیجه را در خروجی های Q0.0وQ0.1 نمایش دهد.

۱-AND آنها را در خروجی Q0.0 نمایشدهد.

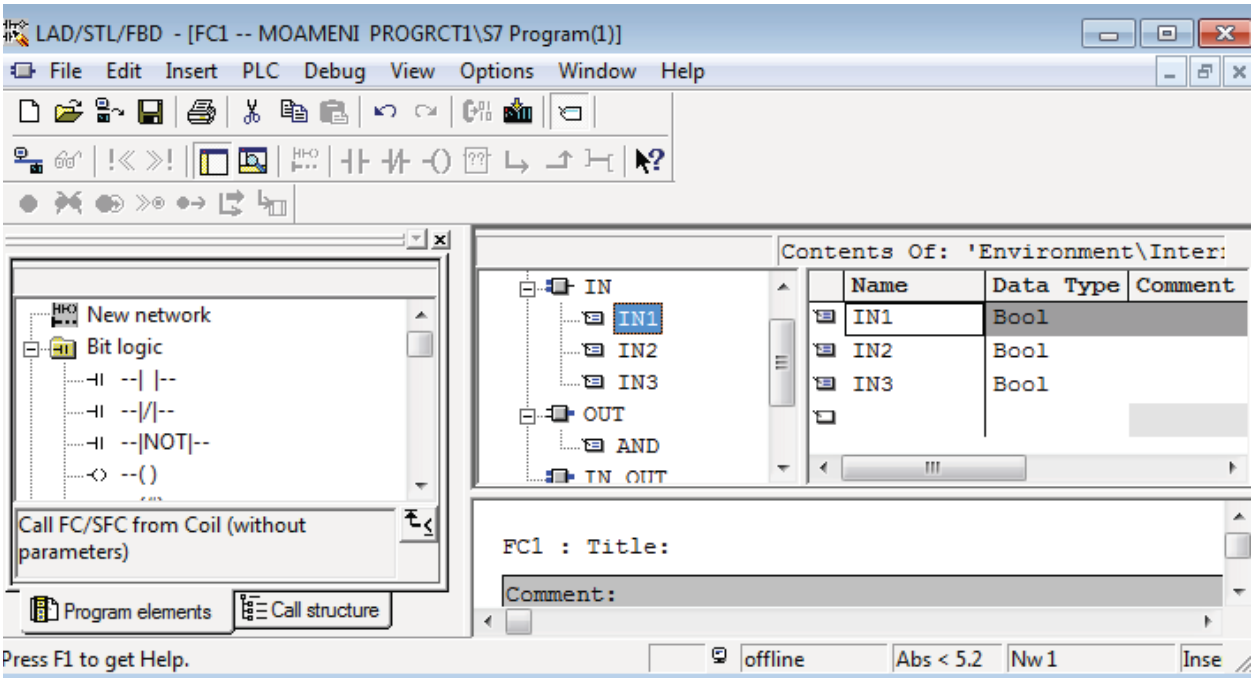
۲- XOR آنها را در خروجی Q0.1 نمایش دهد.

ابتدا به روشی که قبلاً توضیح داده شدیک FC مثلاً FC1 را ایجاد می کنیم. با دو بار کلیک بر روی FC1 آن را باز نموده و نسبت به تکمیل جدول تعریفی (Declaration Table) مانند تصاویر زیر اقدام می کنیم.



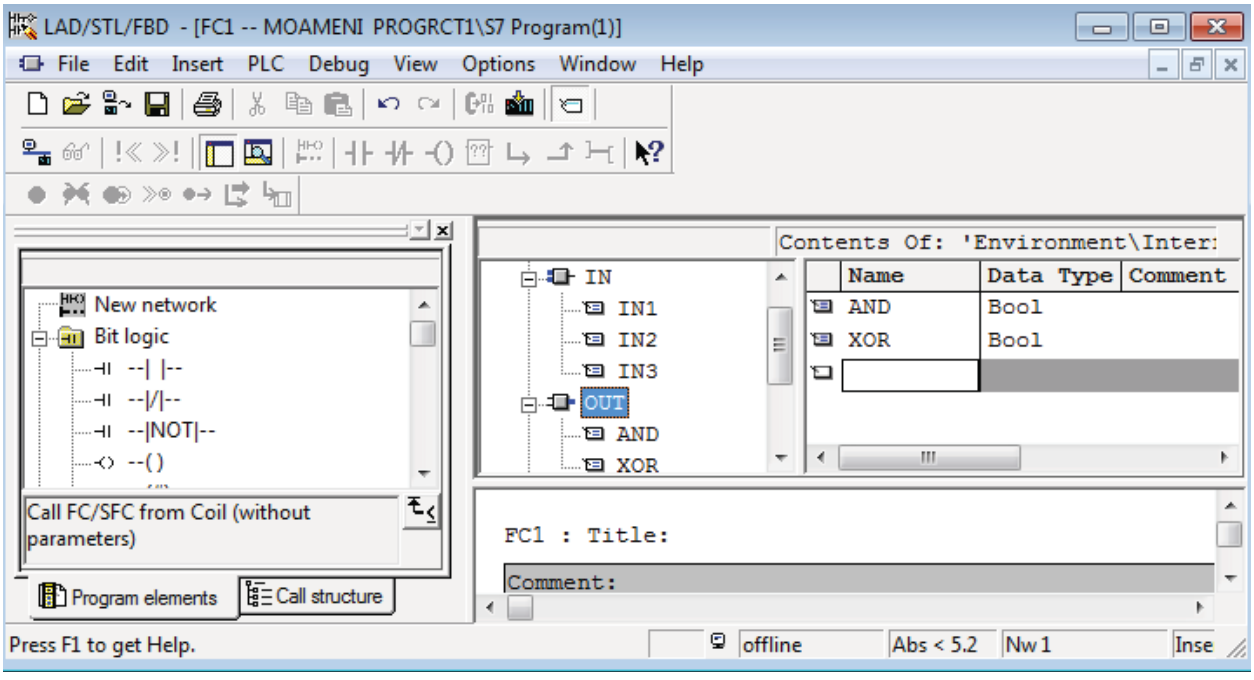
شکل ۲۴.۱۱

در ستون Name به تعداد ورودی ها ، که در این مثال ما سه ورودی داریم نامگذاری می کنیم . در این مثال IN1,IN2,IN3 در ستون Data Type می توان نوع سیستم اعداد ورودی که در اینجا باینری BOOL ، مشخص شده را انتخاب نمود. همچنین در ستون Comment نیز می توان شرحی از مشخصات آن ورودی یا خروجی را نوشت .

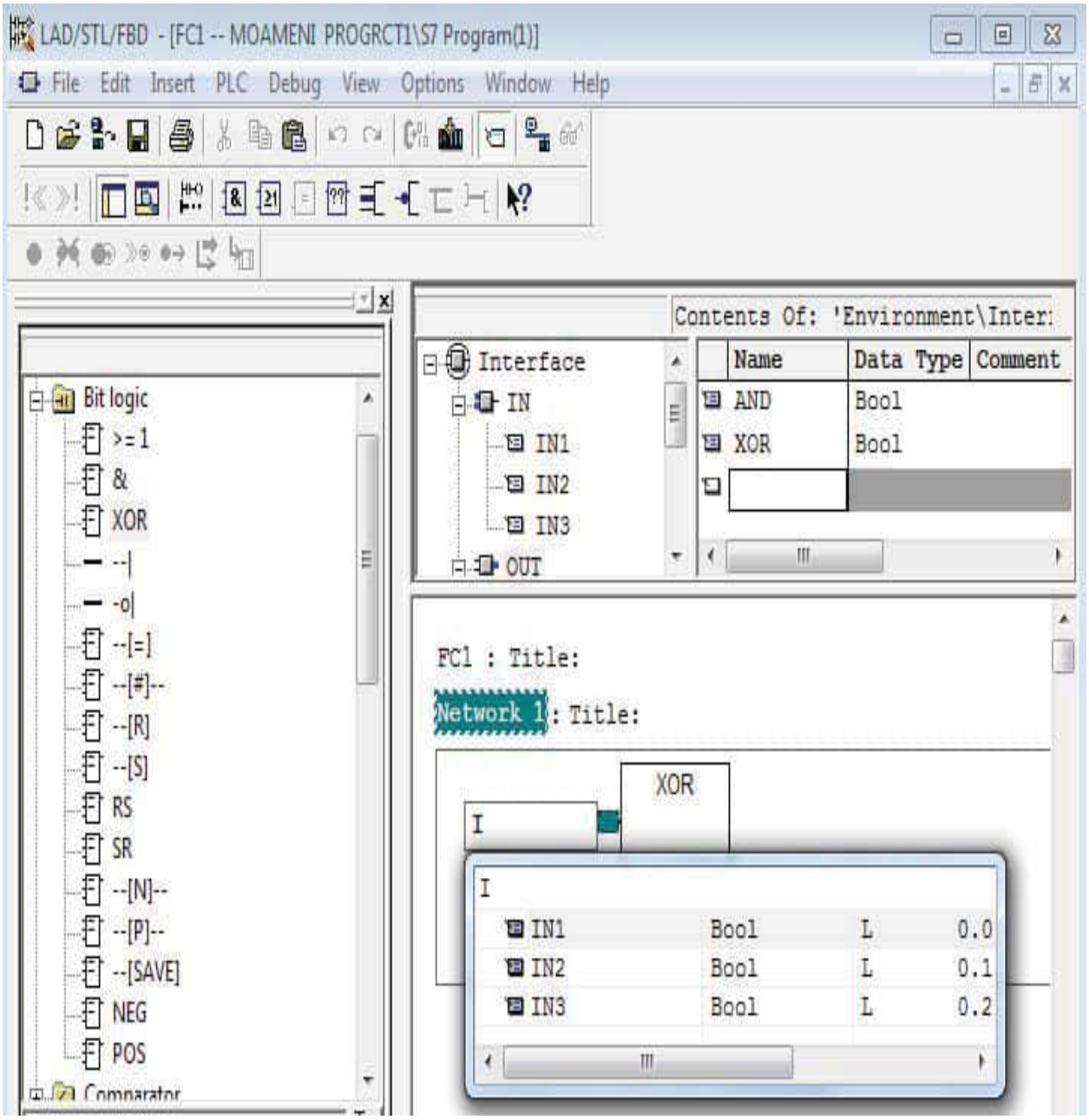


شکل ۲۵.۱۱

پس از خاتمه ی نوشتن ورودی ، حال نوبت نوشتن خروجی می باشد . متناسب با شکل زیر بر روی OUT کلیک نموده و به مانند آنچه در مورد ورودی گفته شد نسبت به تکمیل جدول اقدام می کنیم . ما در این مثال برای روشن شدن موضوع خروجی را با AND و XOR نامگذاری کردیم. در صورتی که قبلاً جدول نمادها فعال شده باشد می توانیم از نمادها نیز استفاده کنیم .

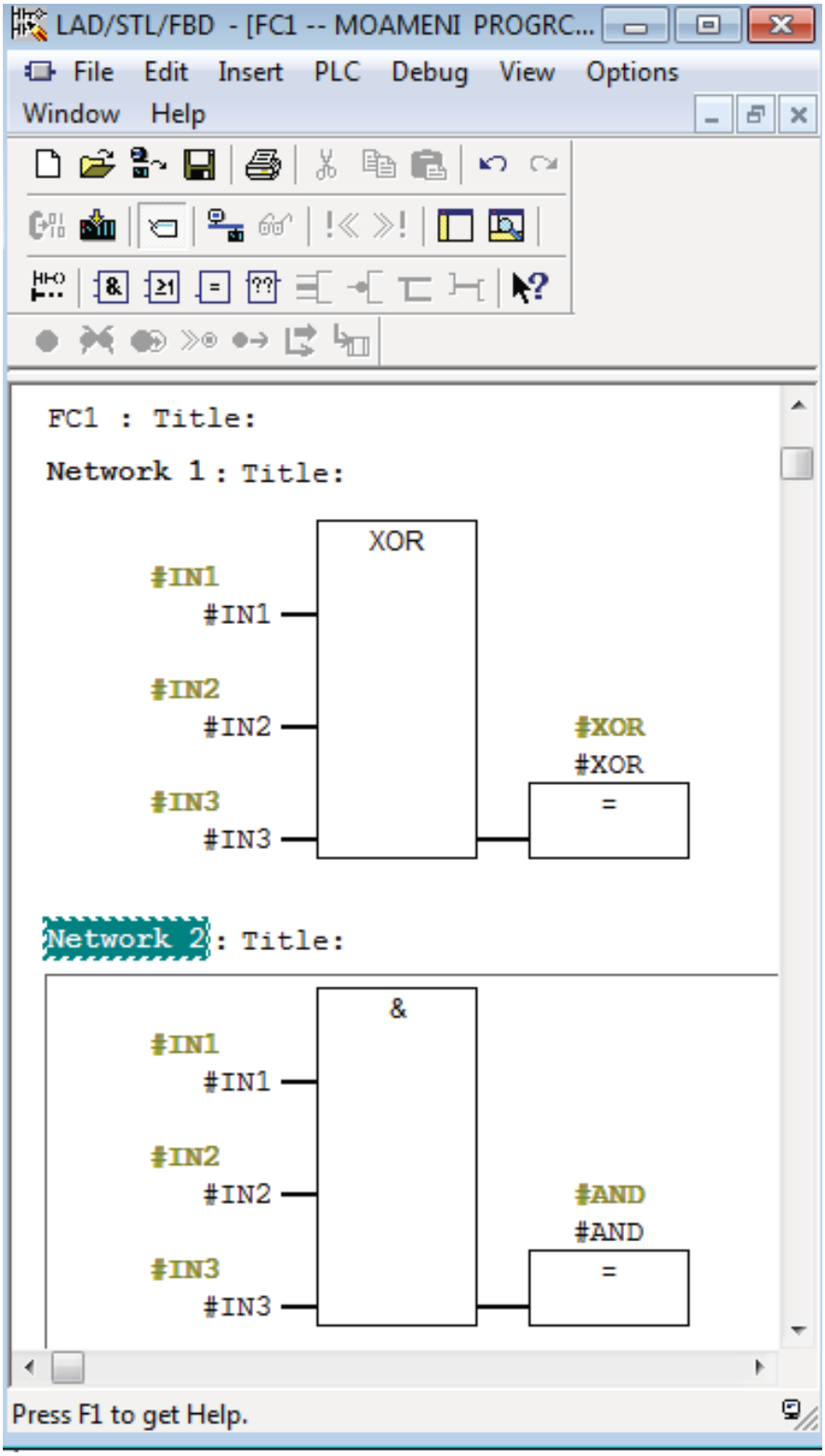


شکل ۲۶.۱۱



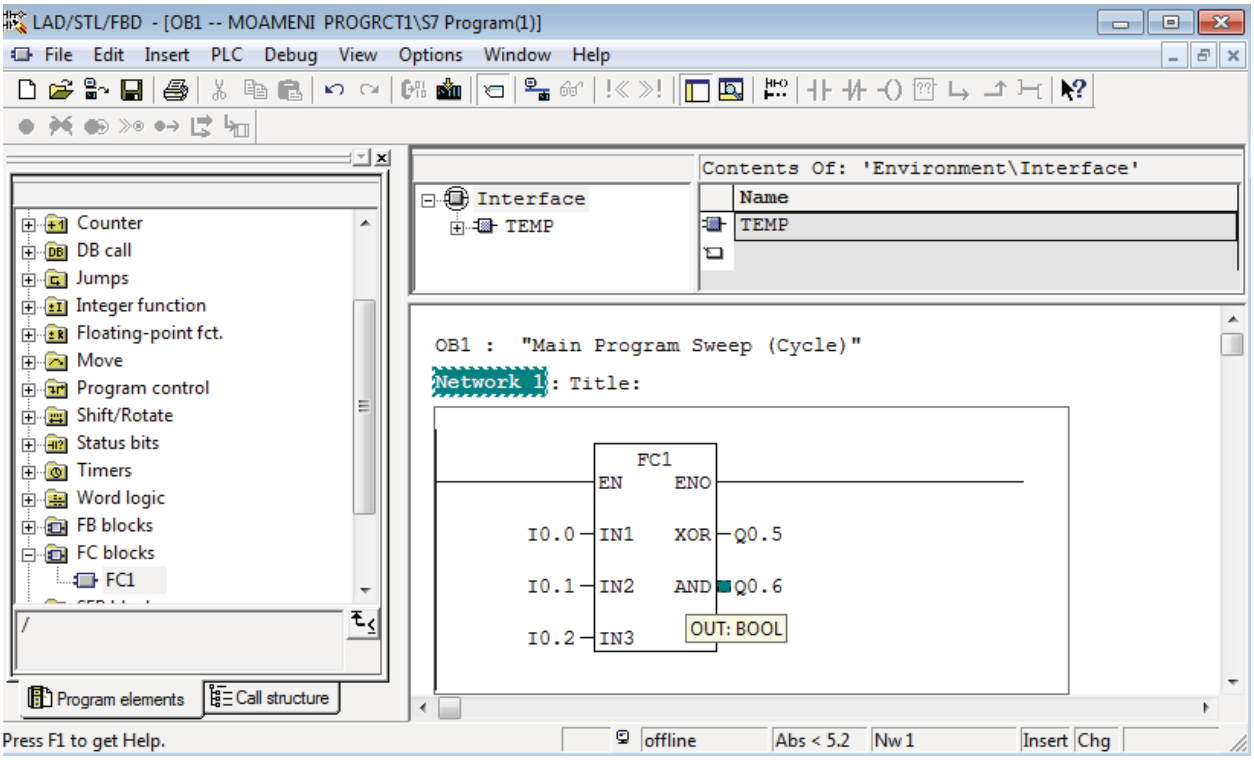
شکل ۲۷.۱۱

پس از تکمیل جدول تعریفی طبق شکل ۲۶.۱۱ سپس FC1 را در دو Network به صورت زیر نوشته و آن را ذخیره و به شبیه ساز یا PLC ارسال می کنیم.


LAD/STL/FBD - [FC1 -- MOAMENI PROGRC...
File Edit Insert PLC Debug View Options
Window Help
FC1 : Title:
Network 1 : Title:
XOR
#IN1
#IN1
#IN2
#IN2
#IN3
#IN3
#XOR
#XOR
=
Network 2 : Title:
&
#IN1
#IN1
#IN2
#IN2
#IN3
#IN3
#AND
#AND
=
Press F1 to get Help.

شکل ۲۸.۱۱

حال OB1 را باز نموده و از طریق فهرست دستورات سمت چپ و از شاخه ی FC blocks بلوک FC1 را در Network ۱ از بلوک OB1 مانند شکل ۲۹.۱۱ نمایان می سازیم.



شکل ۲۹.۱۱

پس از خاتمه ی آدرس دهی ، OB1 را نیز ذخیره و ارسال می کنیم. اکنون با اجرا (RUN) کردن برنامه می توان با همان سه ورودی عملیات XOR, AND را در خروجی های متفاوت مشاهده نمود.

می توان در OB1 در چندین Network یک FC پارامتری (مثلاً FC1 را فراخوانی کرد و با تغییر آدرس های ورودی و خروجی همان اعمال را خواست (در مثال بالا XOR , AND را درخواست نمود) در نوشتن FC از عملگر #IN1,#IN2,#IN3 و خروجی #AND,#XOR استفاده شده است. در OB1 در هر Networkمی توان هر آدرسی را به جای این عملگر(...#)جایگزین نمود.

با توجه به مطالب بیان شده به نکات زیر توجه کنید:

۱-هر تابعی که در IN تعریف شود به عنوان ورودی محسوب می شود.

۲-هر تابعی که در OUT تعریف شود به عنوان خروجی محسوب می شود.

۳- هر تابعی که در IN_OUT تعریف شود ورودی / خروجی محسوب می شود.

۵-RETURN نیز به عنوان خروجی محسوب می شود. تنها زمانی که یک خروجی داشته باشیم.

مزایای FC های پارامتری

-برنامه ی کمتری نوشته می شود.

-به صورت قابل ملاحظه ای حجم برنامه کاهش می یابد.

-در نوشتن یک برنامه ممکن است دچار اشتباه شویم ولی با استفاده از FC های پارمتری خطا به حداقل می رسد.

مثال ۶.۱۱

برنامه ی کنترل دما را با استفاده از FC پارامتری طوری بنویسد که اگر دما از Max (مثلاً ۳۰ درجه) بیشتر شد Cooler روشن و اگر دما از Min (مثلاً ۱۰ درجه)کمتر شد Heater روشن شود.

راهنمایی

به روش گفته شده یک FC می سازیم.

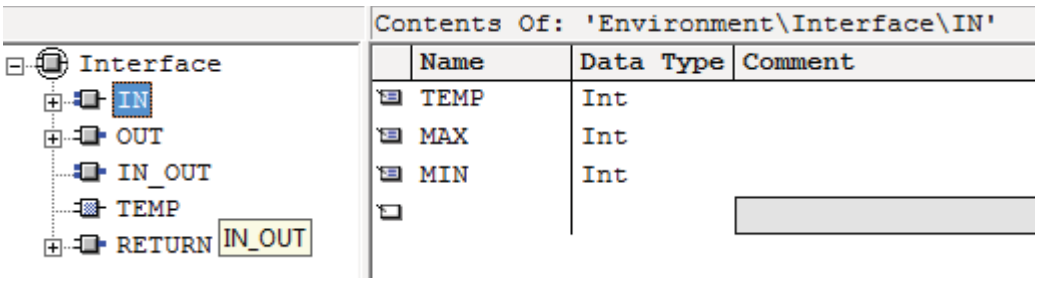
جدول Declaration Table مربوط به FC را تکمیل می کنیم.

کنترل دما Min, Max , Temp را به عنوان ورودی IN و نوع سیستم اعداد Type Data ، از نوع اعداد صحیح INT می باشد.

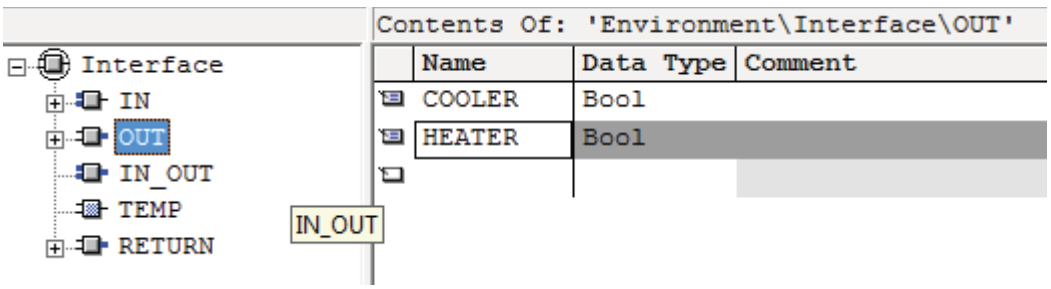
خروجی Cooler و Heater از نوع باینری BOOL می باشد.

پس از تکمیل جدول با استفاده از مقایسه گرها FC را در دو Network می نویسیم و آن را ذخیره و ارسال می کنیم.

سپس در OB1بلوک FC مورد نظر را فراخوانی و آدرس دهی می کنیم و در خاتمه آن را ذخیره ، ارسال و اجرا می کنیم .

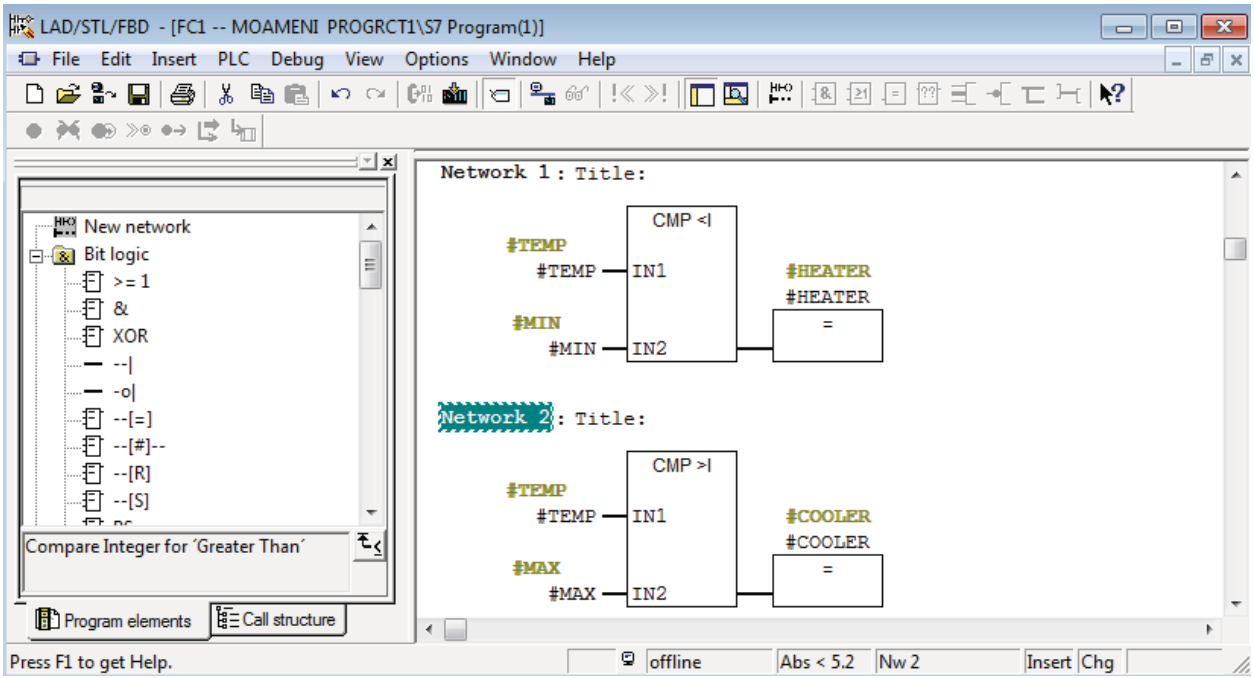


شکل ۳۱.۱۱



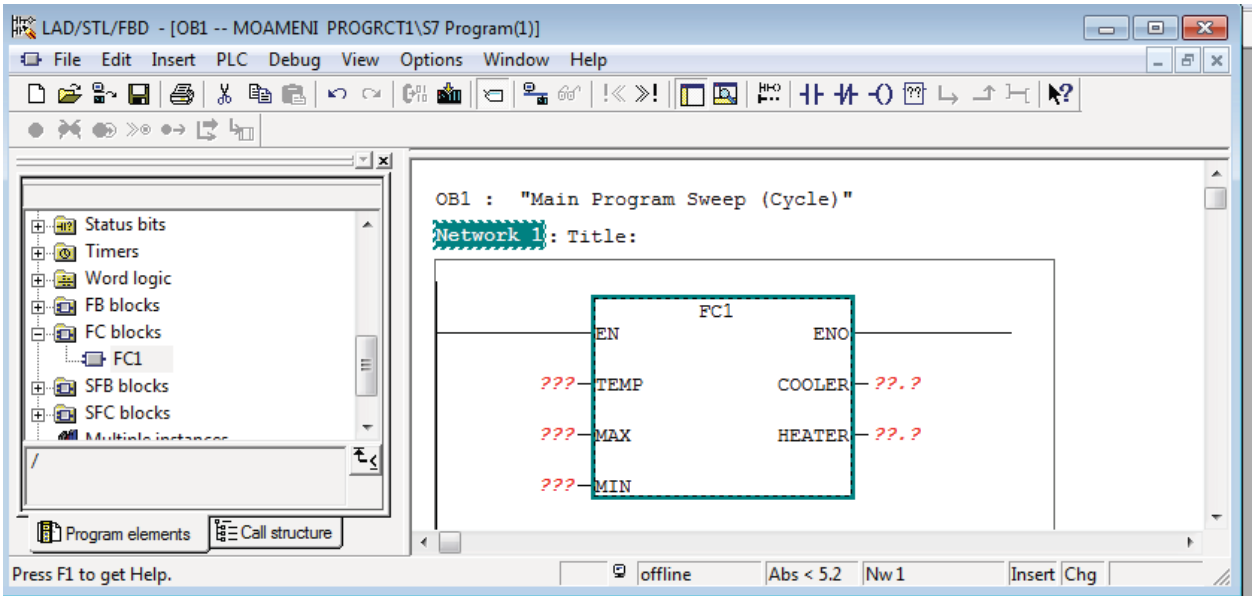
شکل ۳۲.۱۱

پس از تعریف ورودی و خروجی ها FC1را به صورت زیر تکمیل کنید.

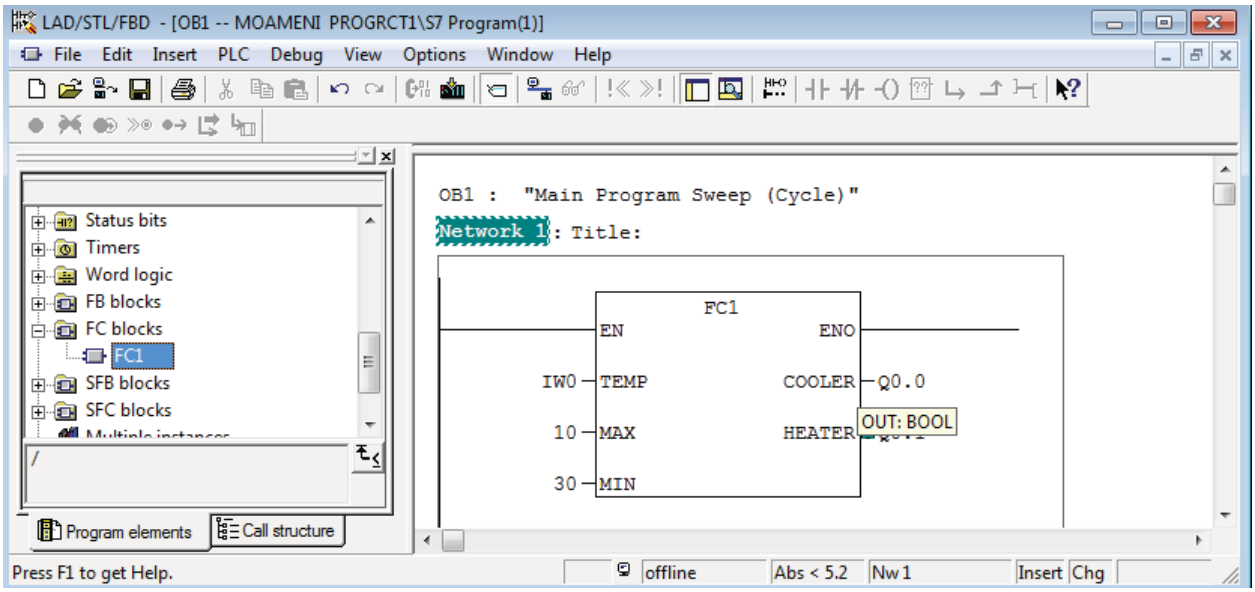


شکل ۳۳.۱۱

برنامه را ذخیره (Save) نموده و درOB1 از قسمت فهرست دستورات بر روی FC2 دوبار کلیک کنید.



شکل ۳۴.۱۱



شکل ۳۵.۱۱

مثال ۷.۱۱

کنترل Valve

با استفاده از FC پارامتری برنامه ای بنویسید که با فشردن کلید پوش باتون Open شیرمخزن باز و با فشردن کلید Close شیر بسته شود. (با باز ماندن شیر مقدار مایع از Max بیشتر و از Min کمتر نشود).

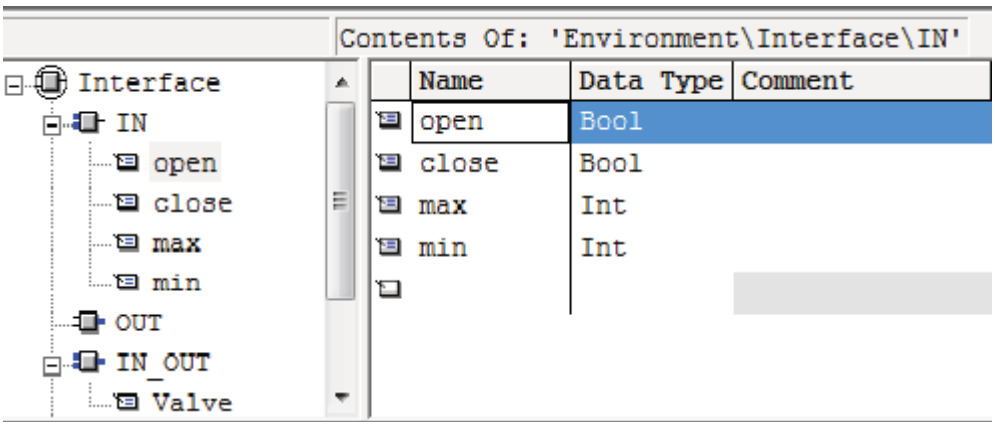
راهنمایی

Openو Close را به عنوان ورودی از نوع BOOL تعریف می کنیم.

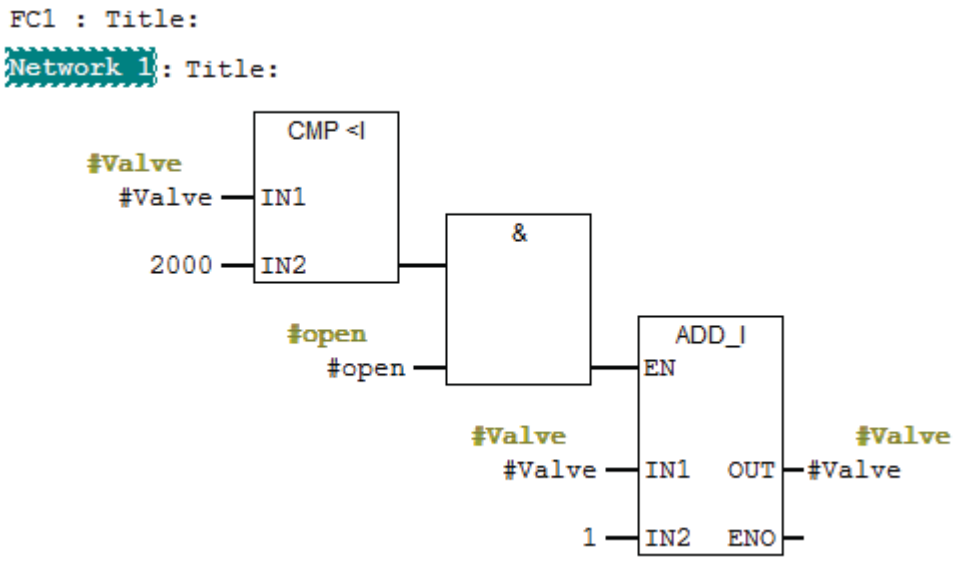
Max و Min را به عنوان ورودی از نوع INT تعریف می کنیم.

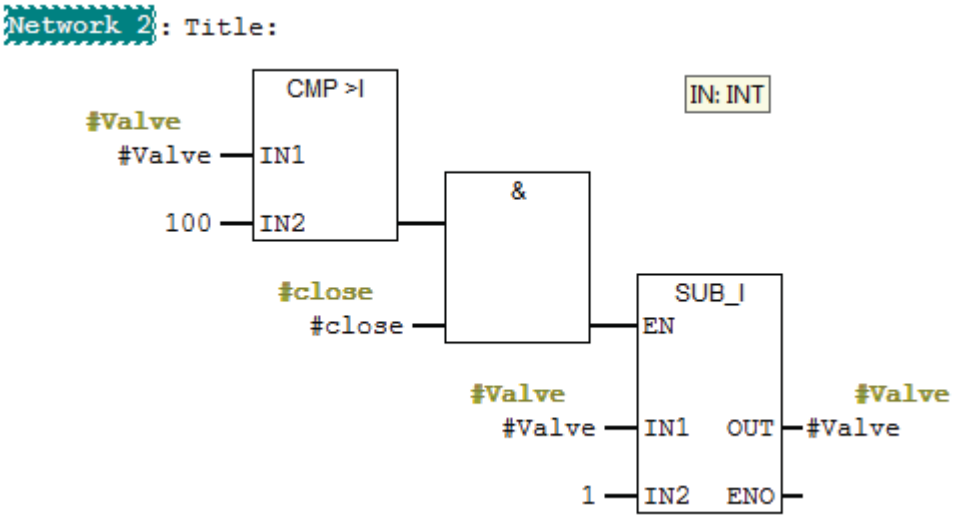
Valve را به عنوان ورودی / خروجی (IN-OUT) از نوع INT تعریف می کنیم. زیرا می خواهیم اگر مایع کم بود اضافه کنیم و اگر زیاد بودکم کنیم. چنین متغییرهایی هم ورودی هستند و هم خروجی .

FC1 را به صورت زیر تکمیل کنید.



شکل ۳۷.۱۱

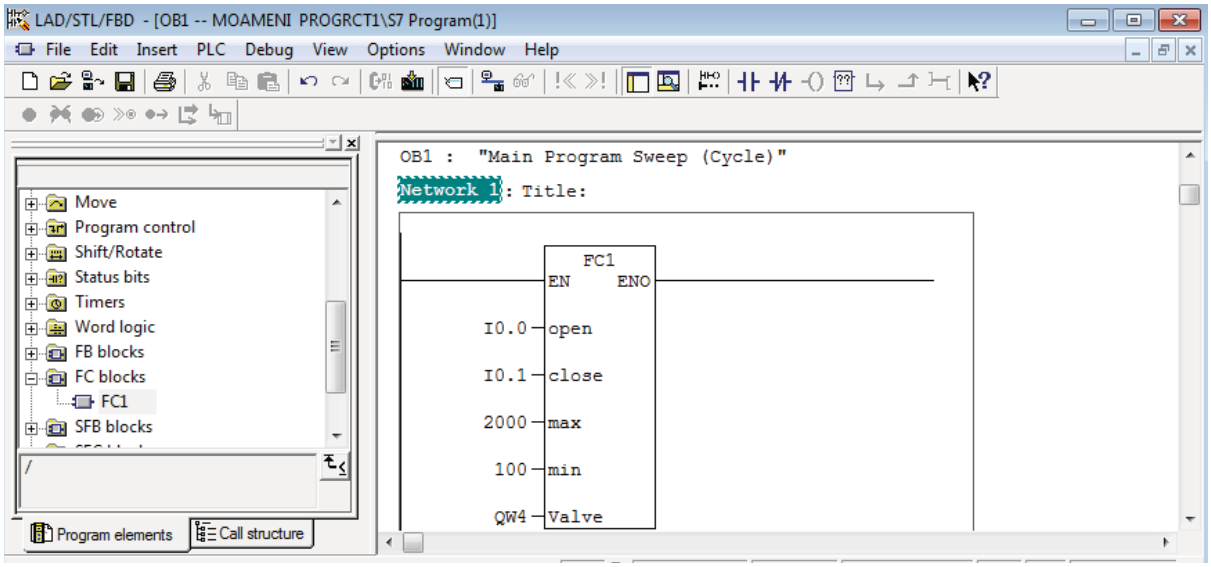




شکل ۳۸.۱۱

FC1 را ذخیره (Save) نموده و به PLC ارسال کنید. مطابق شکلهای زیر OB1 را باز نموده و از قسمت FC blocks ، بر روی FC1 دوبار کلیک نموده و آنرا فراخوانی می کنیم.

شکل ۳۹.۱۱



پس از تکمیل و ذخیره OB1 را به PLC ارسال کنید و عملکرد برنامه را در وضعیت اجرا (RUN) بررسی کنید.

مثال ۸.۱۱

برنامه ای بنویسد که با فعال شدن هر یک از ورودی های I0.0و I0.5خروجی های مربوطه Q4.0 و Q5.0به صورت مجزا چشمک بزنند و با فعال شدن هر یک از ورودی های I0.1و I0.6خروجی های مربوطه Q4.0 و Q5.0 به صورت مجزا خاموش شوند. در حل برنامه از OB 35 به عنوان چشمک زن استفاده کنید.

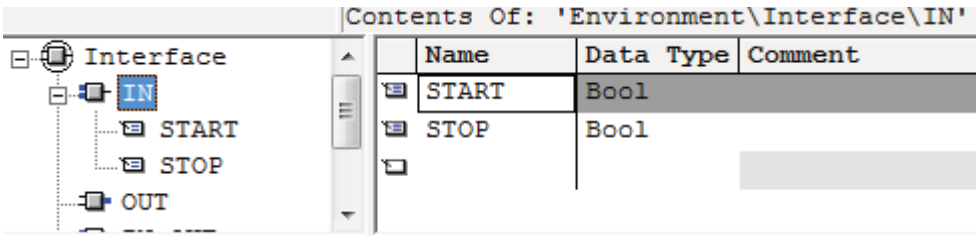
راه حل پیشنهادی

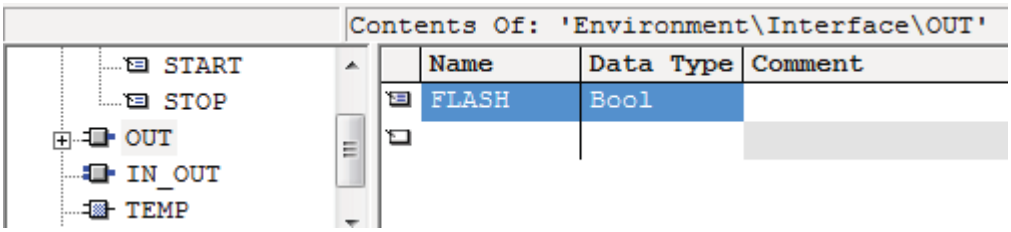
با نگاهی ساده می توان این برنامه را با استفاده از FC پارمتری حل نمود که نمونه آن در زیر اورده شده است . اما با اجرای برنامه مشاهده می شود به علت استفاده از حافظه ی مشترک MO.O با فعال شدن I0.0و I0.5هر دوخروجی فعال و با فعال شدن I0.1 و I0.6 هر دو خروجی خاموش می شوند و هیچ قسمت از ورودی ها و خروجی ها به صورت مستقل عمل نمی کنند .

برنامه ی FC4 ، OB 35 4 و OB1 را به صورت زیر ایجاد کنید. قبل از نوشتن برنامه در FC4 پارمترها را به صورت زیر تعریف کنید.

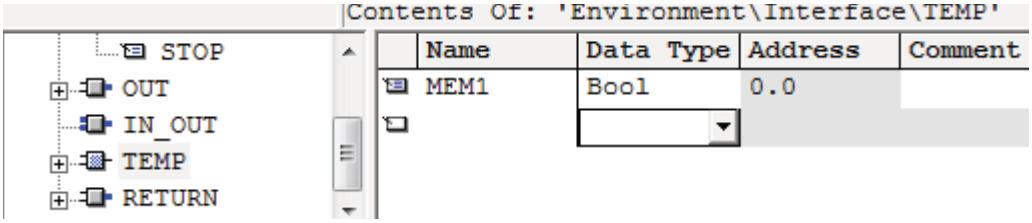
مطابق اشکال (۴۱.۱۱ ، ۴۲.۱۱ ، ۴۳.۱۱) START و STOP را به عنوان ورودی ، MEM1 نه ورودی و نه خروجی (TEMP) و Flash را به عنوان خروجی تعریف می کنیم.

شکل ۴۱.۱۱

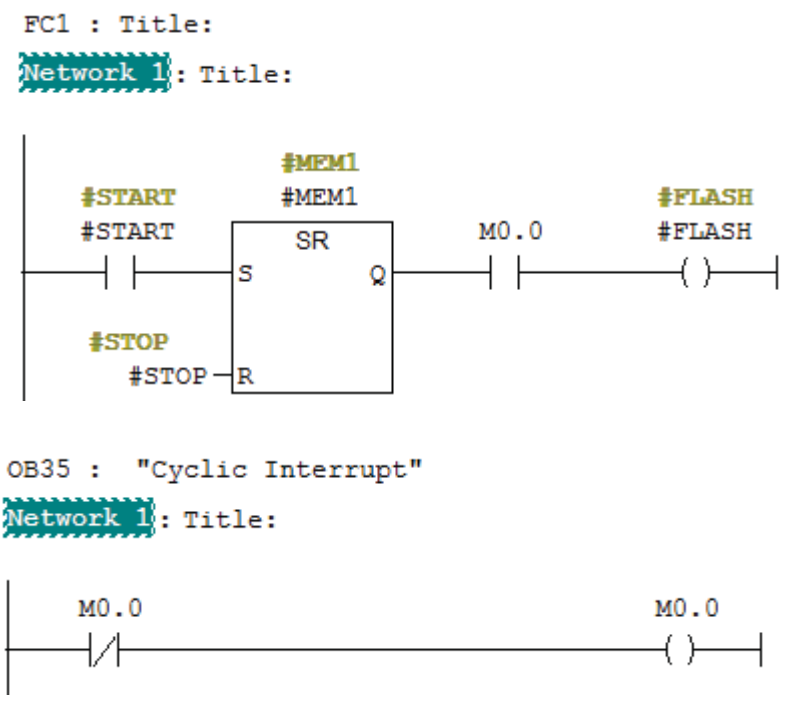


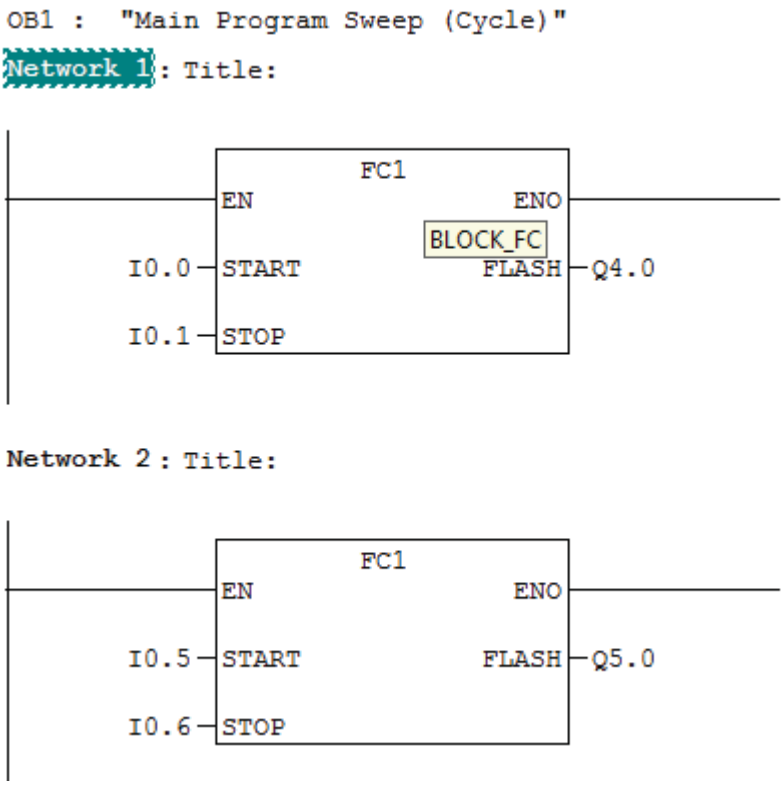


شکل ۱۱-۴۲



شکل ۱۱.۴۳





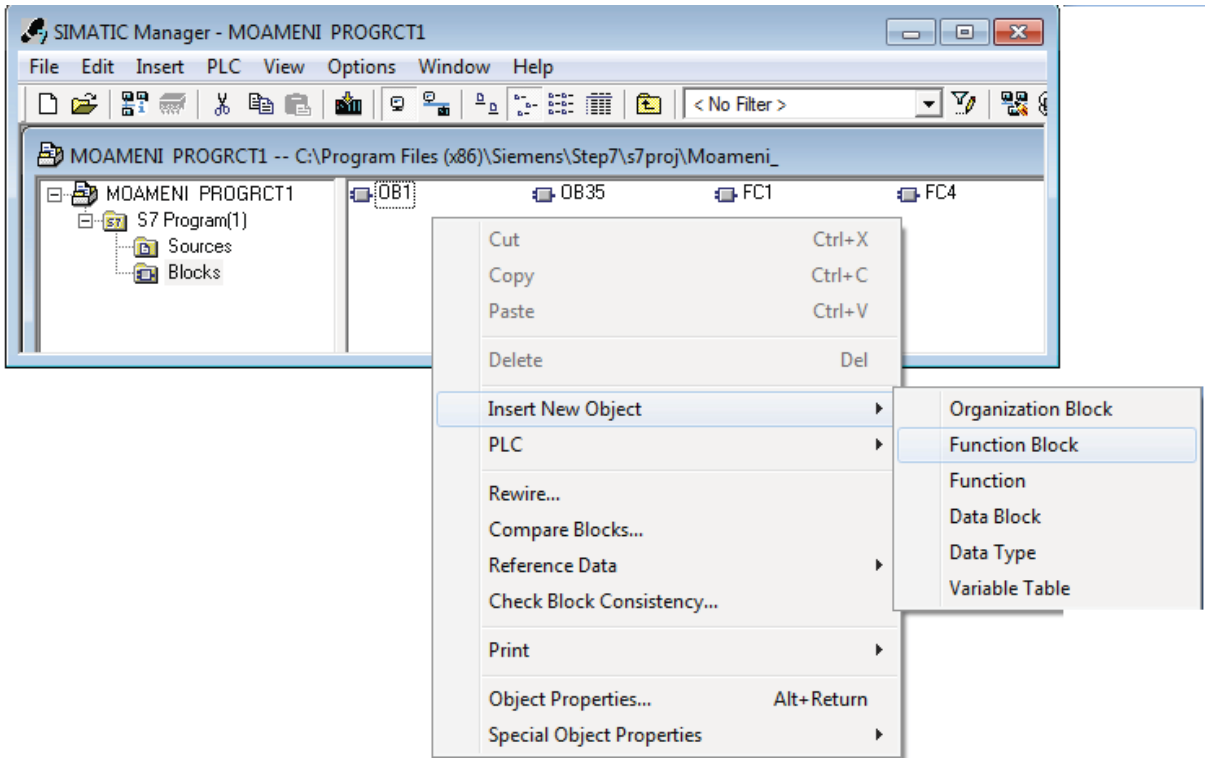
با فعال کردن I0.0 یا I0.5 هر دو خروجی فعال وبافعال کردن I0.1 یا I0.6هر دو خروجی خاموش می شود؟ علت استفاده از یک حافظه ی مشترک مانند MO.O بین دو سیستم می باشد.

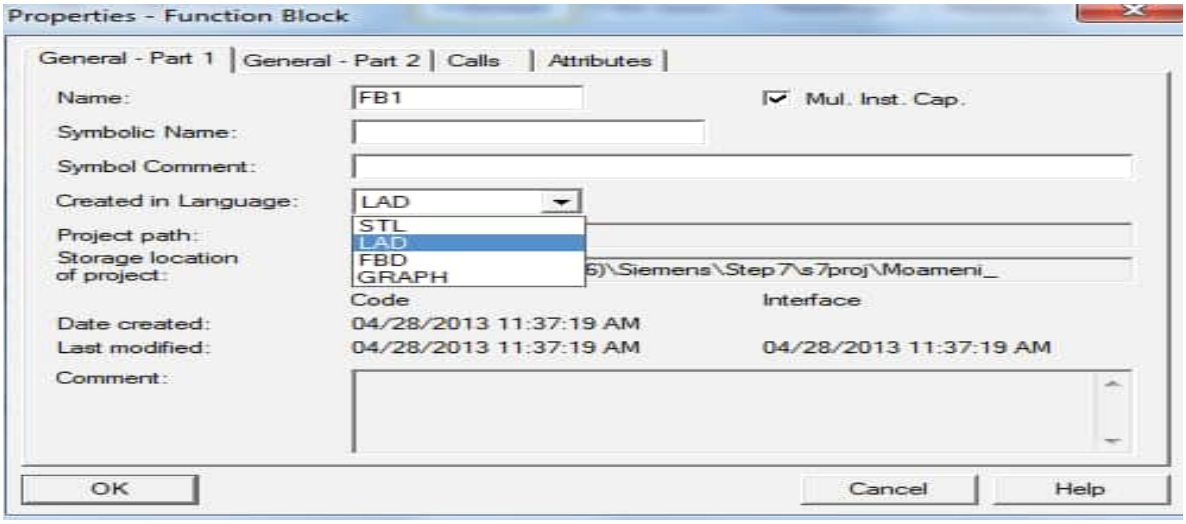
این حافظه ی مشترک ایجاد تداخل می کند. به عبارت دیگر اگر تابع پارامتری باشدو بخواهیم این تابع را در جاهای مختلف استفاده کنیم تداخل ایجاد می شود زیرا همچنان که می دانیم در FC ها نمی توان از عناصر حافظه دار استفاده کرد. برای رفع این مشکل می توان از FB ها که عناصری حافظه دارهستنداستفاده نمودFB هااین قابلیت رادارندکه درهر Networkکه استفاده شوند برای هر کدام یک حافظه مجازی اختصاص می دهند.

نکته: در FC پارامتری به دلیل آنکه فاقد حافظه می باشد ، در صورتی که از آدرس های مشترک یا از Memory های مشترک در Network ها استفاده کنیم ، موجب ایجاد تداخل و به هم ریختن در اجراء برنامه خواهد شد . برای حل مشکل می توانیم از FB استفاده کنیم که دارای حافظه هستند. هر بار که یک FB را در OB1 فراخوانی کنیم با این عمل یک ؟؟DB به این برنامه اختصاص می یابد و هر بار که در برنامه بخواهیم از FB استفاده کنیم برنامه به طور خودکار به سراغ این ؟؟DB می رود و هر بار یک حافظه به آن اختصاص می دهد و از ایجاد تداخل جلوگیری می کند.(مزیتFB نسبت به FC)

روش ایجاد و استفاده از بلوک تابع FB (Function Blocks) و فراخوانی آن. درمنوی SIMATICManagerراست کلیک کرده وازگزینه ی InsertNew Object

Function Blocks را انتخاب و نام FB1 را می رویسیم. در اشکال زیرمراحل ایجاد FB را مشاهده می کنید.





در قسمت Name نام FB1 به طورپیش فرض نوشته شده است.

برای ایجاد یک FB و درک عملکرد آن در قالب یک مثال به شرح آن می پردازیم:

مثال ۹.۱۱

برنامه ای بنویسید که با فعال شدن هر یک از ورودی های I0.0و I0.5 خروجی های مربوطه Q4.0 و Q5.0 به صورت مجزا چشمک بزنند. و با فعال شدن هر یک از ورودی های I0.1 و I0.6 خروجی های مربوطه Q4.0 و Q5.0 به صورت مجزا خاموش شوند. در حل برنامه از OB 35 به عنوان چشمک زن استفاده کنید.

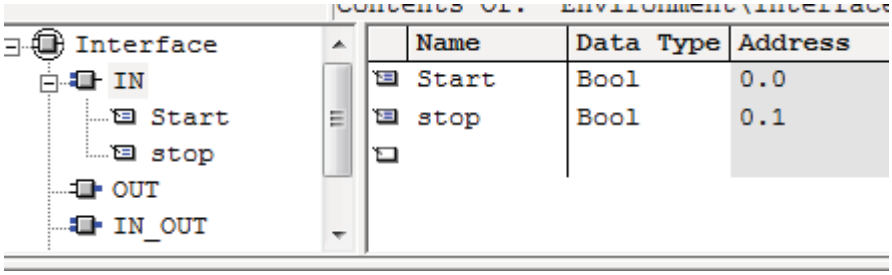
برای حل مسئله از FB استفاده می کنیم که دارای حافظه می باشد.

ابتدا FB را به روشی که گفته شده ایجاد می کنیم و به عنوان مثال آن را FB۱ می نامیم.

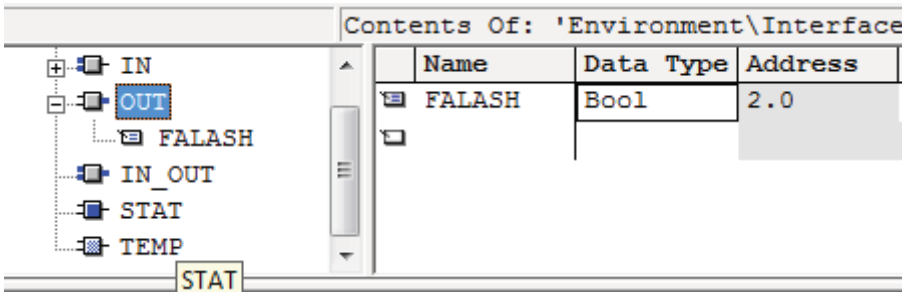
با دوبار کلیک بر روی FB1 وارد آن شده و جدول Declaration Table را به صورت زیر تکمیل می کنیم.

FB1 را ایجاد کرده و IN, OUT و STAT را به صورت زیر تعریف می کنیم.

START و STOP به عنوان ورودی را در ستون IN به صورت باینری BOOL می نویسیم.

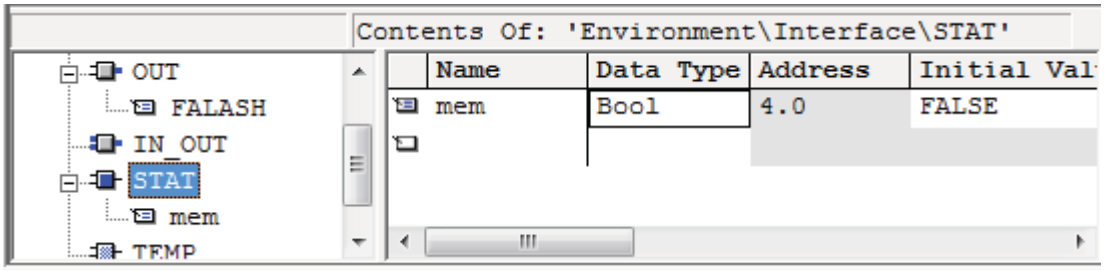


FLASH را به عنوان خروجی در ستون OUT به صورت باینری می نویسیم.

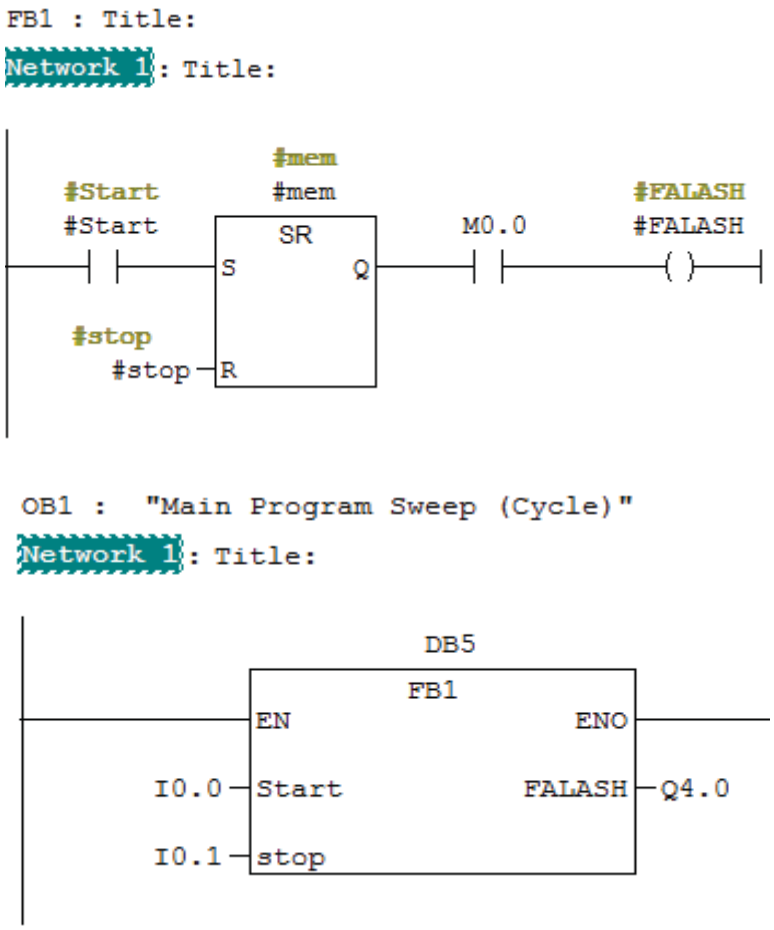


حافظه ی مورد استفاده در متن برنامه FB1 را با نام MEM در ستون STAT به صورت باینری می نویسیم.

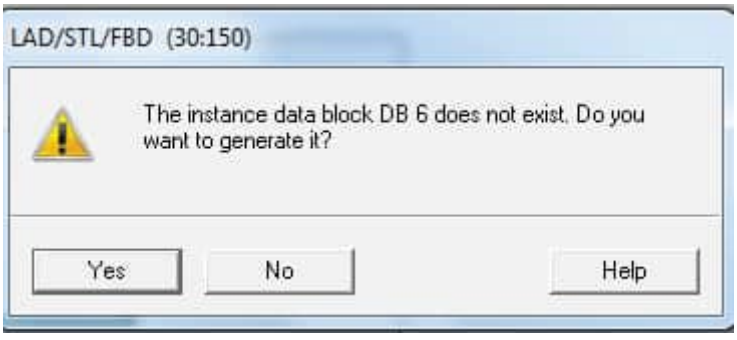
اگر متغیری در FB به صورت STAT تعریف شود ، هر بار که FB را به کار می بریم یا فرا می خوانیم یک حافظه ی مجزا به آن متغیر اختصاص می دهد.

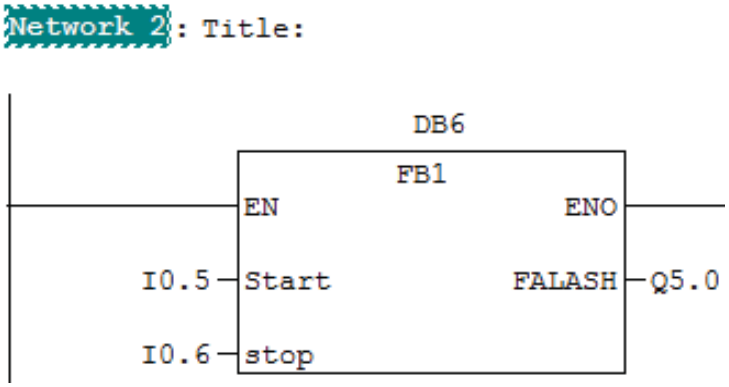


پس از ایجاد FB1 ، آن را ذخیره کرده و در OB1 از قسمت دستورات برنامه نویسی FB1 را فراخوانی می کنیم.



به هر Network که فراخوانی می کنیم یک DB اختصاص می دهیم و شماره ی آن را تایپ می کنیم . در این حالت پنجره ای مبنی بر ایجاد DB و در صورت موجود بودن DB مبنی بر دوباره نویسی آن نمایان می شود که با OK پاسخ داده می شود (مگر این که DB را قبلاً ساخته ایم و نمی خواهیم آن را دوباره نویسی کنیم)





پس از نوشتن برنامه در SLMATIC Manager اگر DB های مختص به OB1 را باز کنیم منوی زیر ظاهر می شوند.

Address	Declaration	Name	Type	Initial value	Actual value
0.0	in	Start	BOOL	FALSE	FALSE
0.1	in	stop	BOOL	FALSE	FALSE
2.0	out	FALASH	BOOL	FALSE	FALSE
4.0	stat	mem	BOOL	FALSE	FALSE

به نکات زیر توجه کنید:

۱-به جای این که MEM را به عنوان TEMP تعریف کنیم آن را به صورت STAT تعریف می کنیم.

۲-اگر متغیری در FB به صورت STAT تعریف شود. هر بار که FB را به کار می بریم یا فرا می خوانیم یک حافظه ی مجزا به آن متغیر اختصاص می دهد.

مثال ۱۰.۱۱

موتوری را در نظر بگیرد که به دو صورت چپ گرد و راست گرد کار می کند. برای راه اندازی این موتور از یک پوش باتون Start و یک پوش باتون Stop و برای تغییر جهت آن از دو لیمیت سوئیچ استفاده می شود. برنامه ی مورد نظر را با استفاده از FB بنویسید.

راهنمایی

-FB جدیدی را ایجاد نموده و در آن LL و RL و Start و Stop را به عنوان IN تعریف می کنیم.

- Left و RIGHT را به عنوان OUT تعریف می کنیم.

در قسمت STAT دو Mem1 و Mem2 را برای وضعیت های (۱) Start ، (۰) Stop و (۱) Left ، (۰) Right تعریف می کنیم.

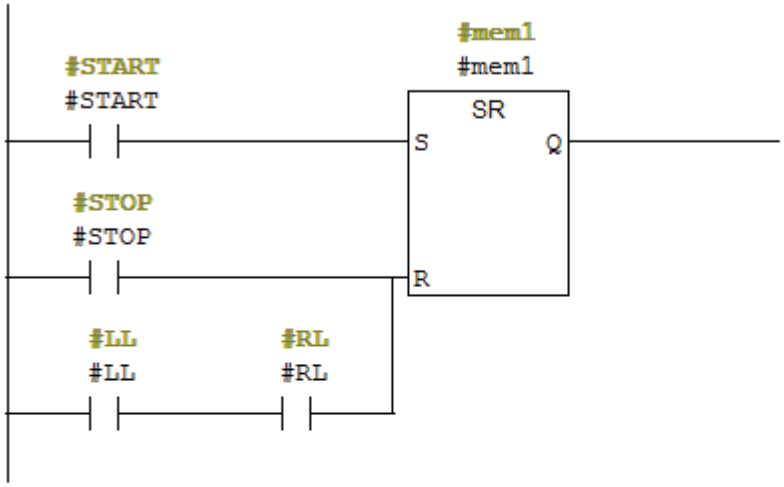
جدول ۲.۱۱

FB1

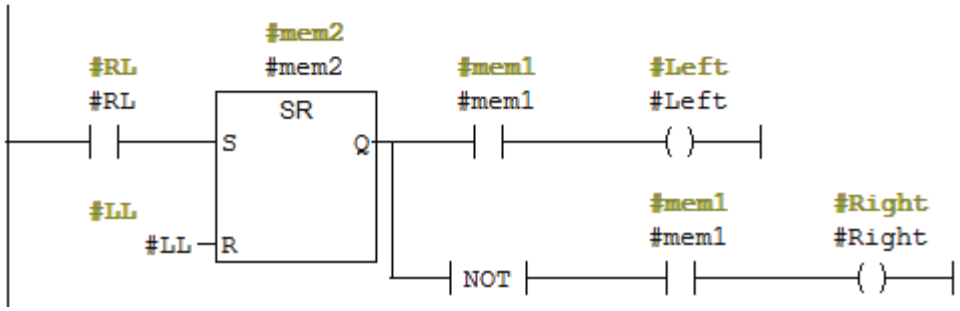
Address	Declaration	Name	Type	Initial value	Actual value
0.0	in	LL	BOOL	FALSE	FALSE
0.1	in	RL	BOOL	FALSE	FALSE
0.2	in	STOP	BOOL	FALSE	FALSE
0.3	in	START	BOOL	FALSE	FALSE
2.0	out	Left	BOOL	FALSE	FALSE
2.1	out	Right	BOOL	FALSE	FALSE
4.0	stat	mem1	BOOL	FALSE	FALSE
4.1	stat	mem2	BOOL	FALSE	FALSE

FB1 : Title:

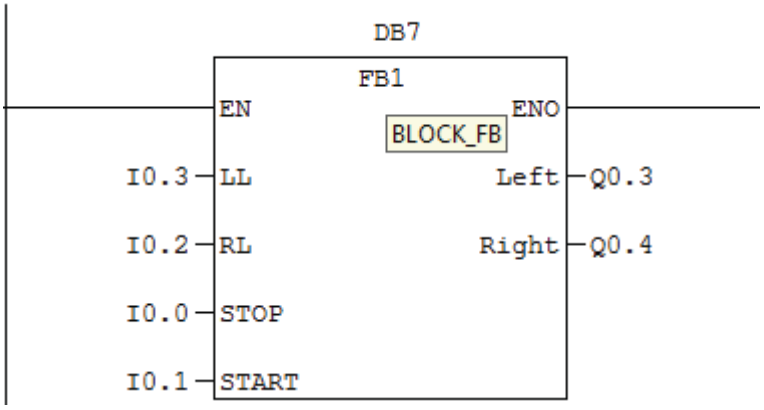
Network 1: Title:



Network 2 : Title:



OB1



نکته ۹- در آدرس دادن به عملگر # توجه کنید.

مثال ۱۱.۱۱

برنامه ی Start/Stop را با بلوک FB به گونه ای بنویسید که دو نفر همزمان از آن استفاده کنند.

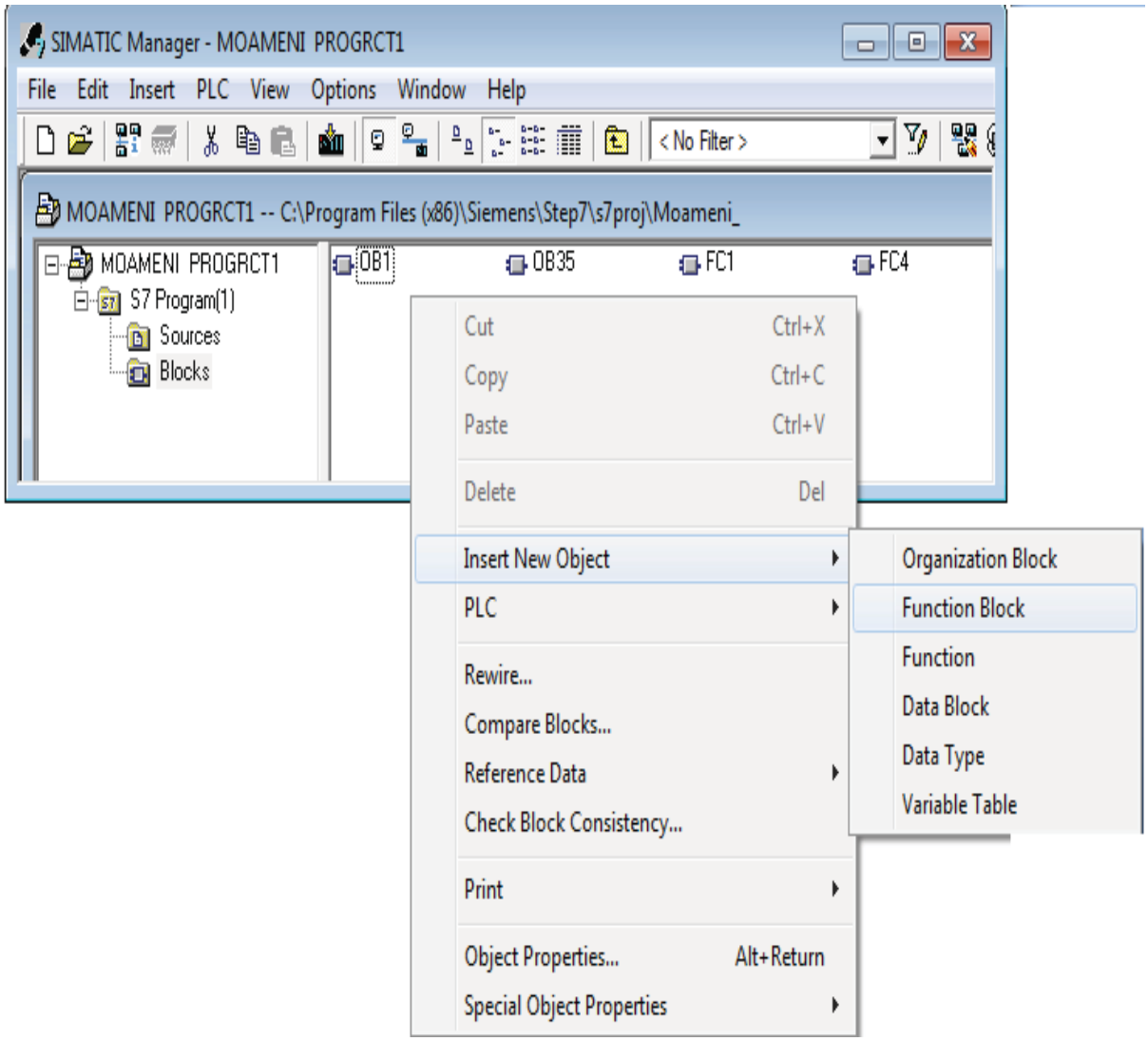
مراحل زیر را اجرا می کنیم.

-دارای دو حافظه می باشیم که یکی را MEM1 و دیگری را MEM2 قرار می دهیم.

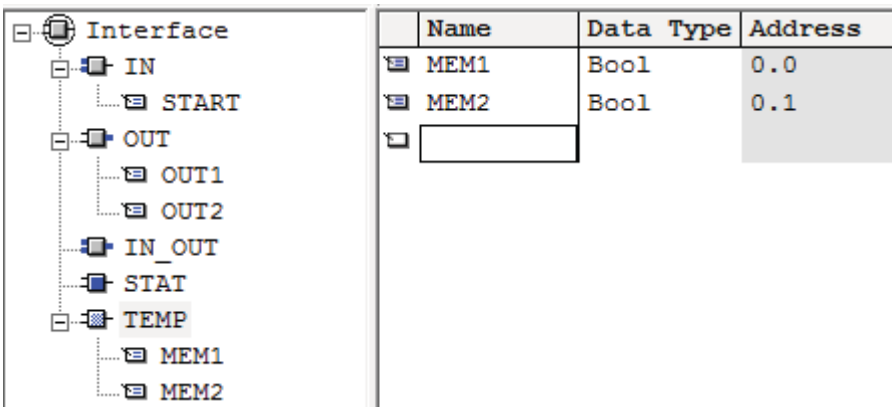
-یک عدد ورودی داریم که آن را در IN ، Start نامگذاری کردیم.

- دو خروجی داریم که آن را در OUT ، OUT1 و OUT2 نامگذاری کردیم.

شکل ۵۲.۱۱

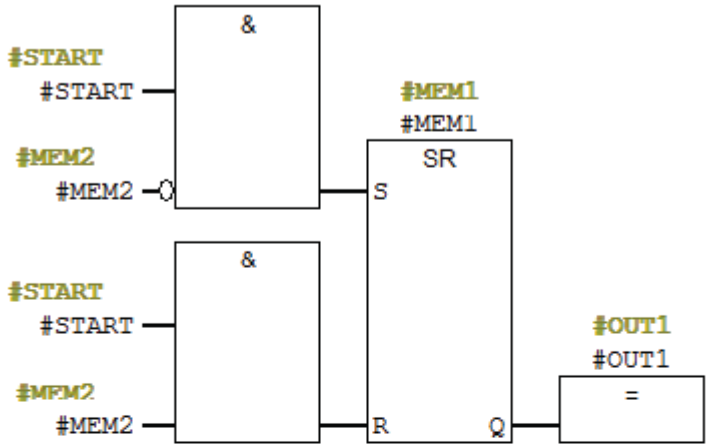


برنامه FB1 را مطابق شکل های زیر نوشته و آن را ذخیره می نماییم .

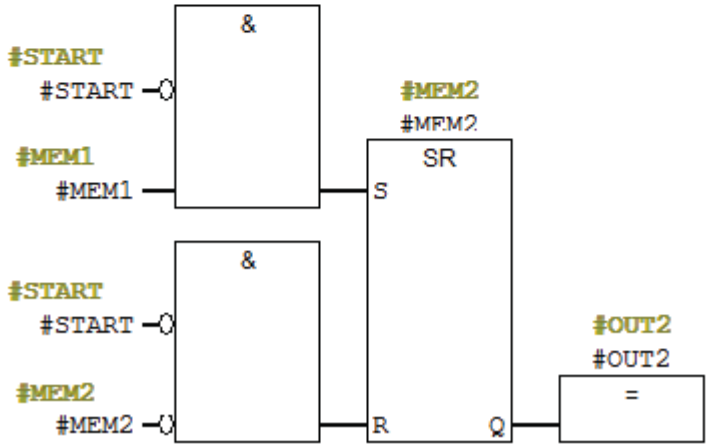


FB2 : manucher project

Network 1: Title:



Network 2 : Title:

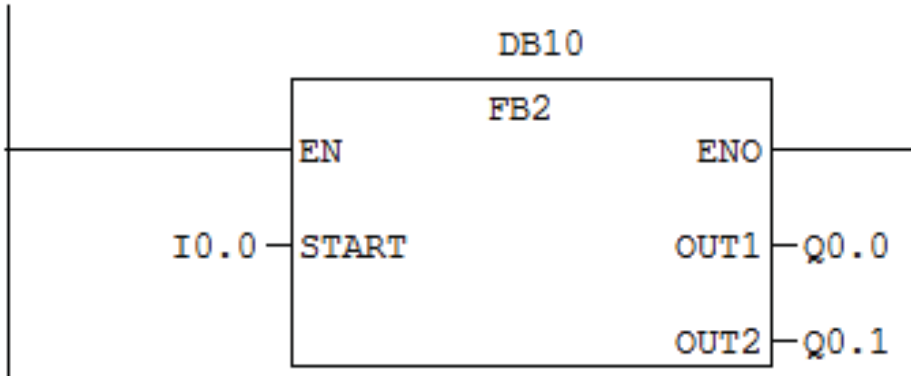


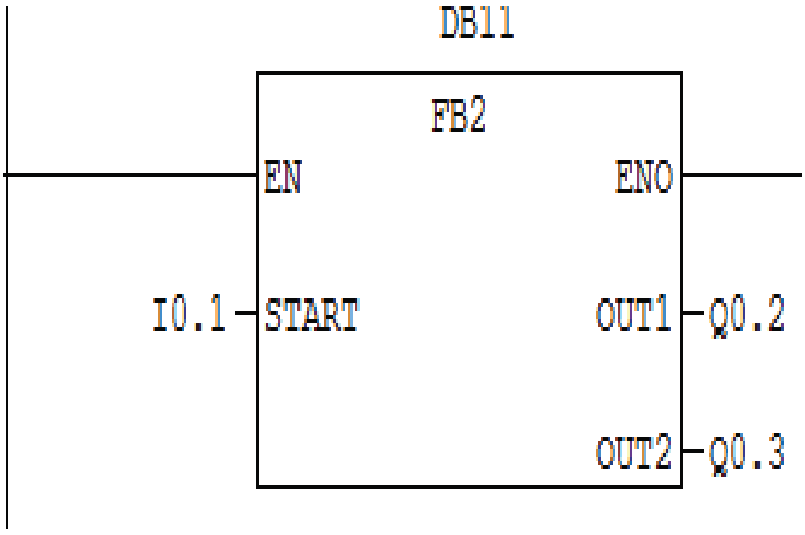
حال OB1 را باز نموده و FB1 را در دو Network مطابق شکل زیر فراخوانی می کنیم. جزوه خود آموز

پس از ذخیره OB1 را به همراه FB1 به PLC یا PLCSIM ارسال و عملکرد برنامه را بررسی کنید.

OB1 : "Main Program Sweep (Cycle)"

Network 1 : Title:





فصل ششم

بلوک های داده (DATA BLOCKS)

در این فصل آشنا خواهید شد با .....

آشنایی با بلوک های داده

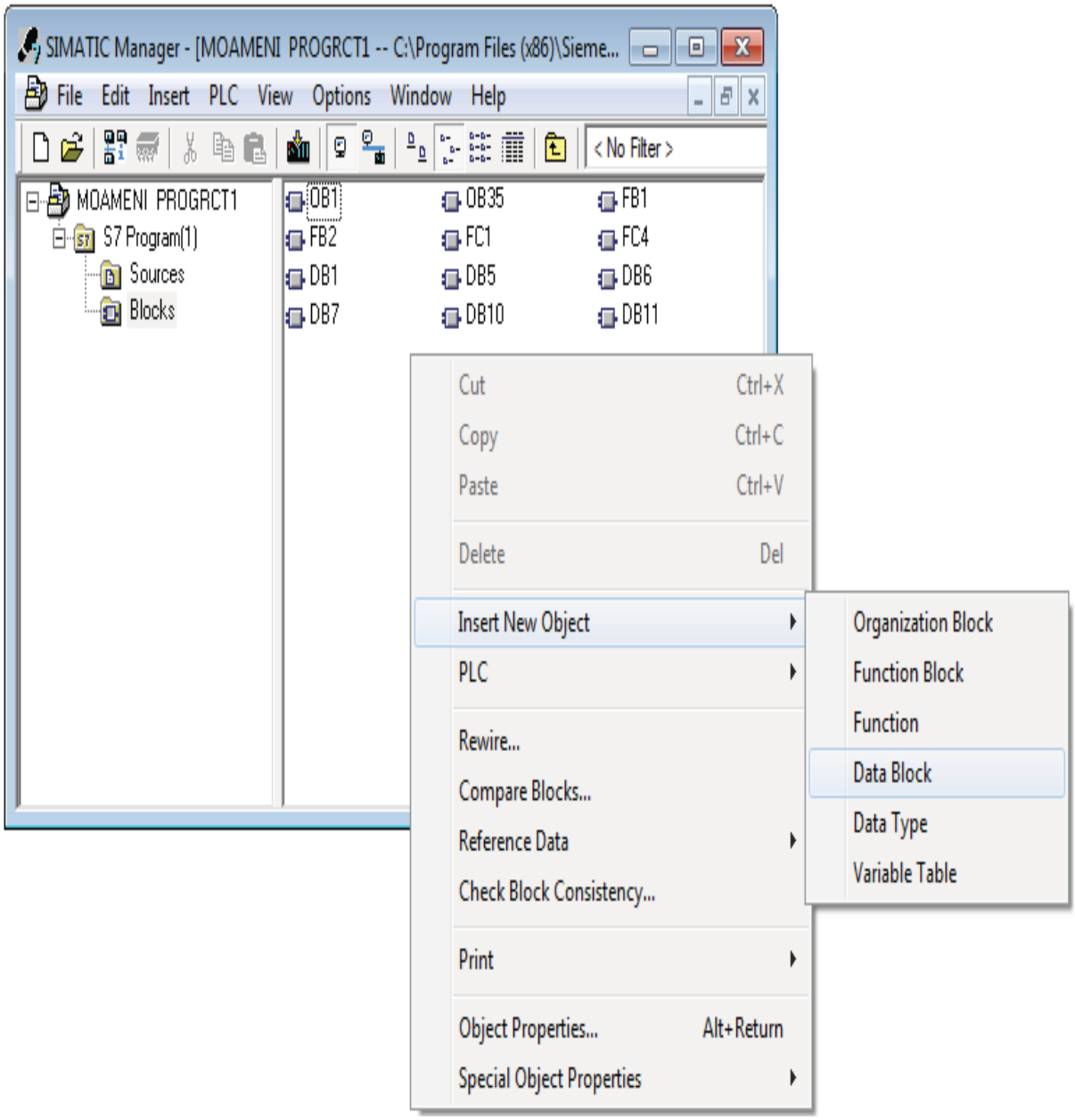
نحوه ارتباط با آنها و برنامه نویسی به وسیله بلوک های داده

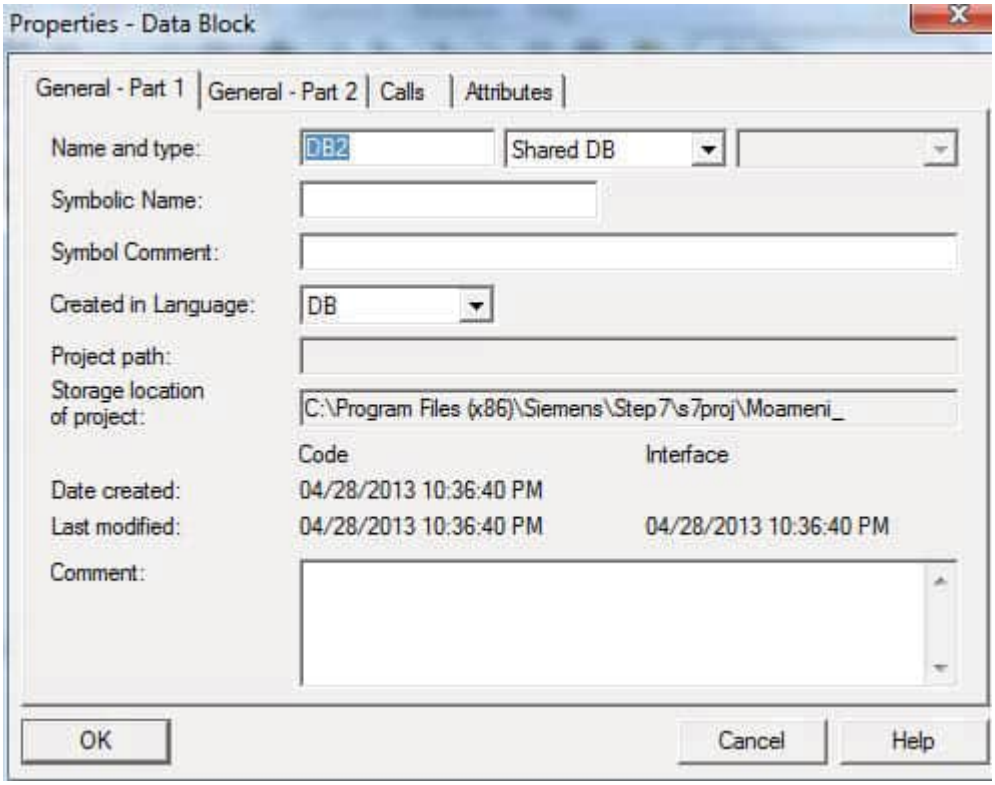
بلوک های داده (DATA BLOCKS)

همان طور که گفته شد این بلوک ها شامل عملیات منطقی نمی با شند و برای ذخیره داده ها به کار می روند در ادامه نحوه ایجاد کردن یک DB و استفاده از آن نشان داده می شود.

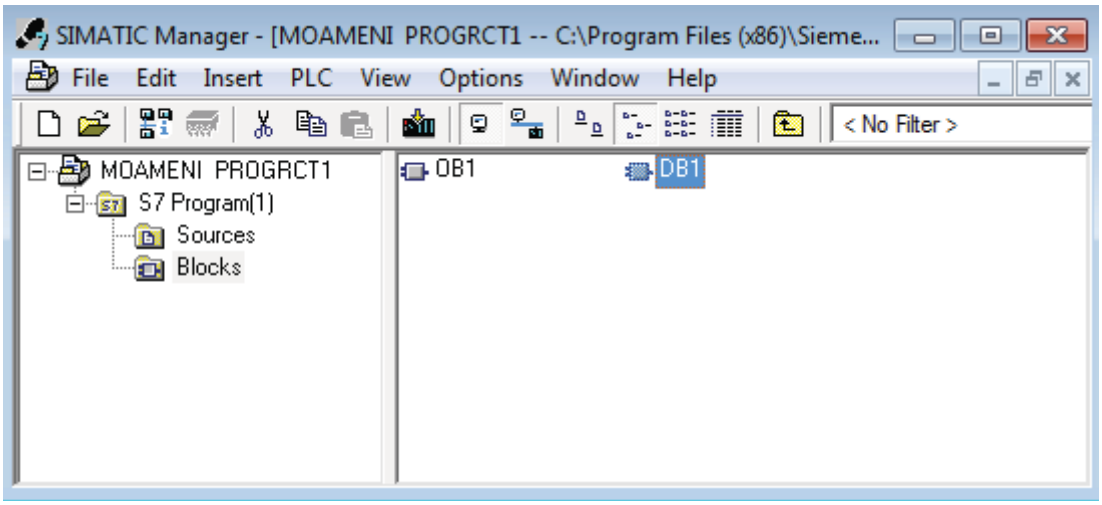
روش ایجاد و استفاده از بلوک داده DB(Data Blocks)

قصد داریم یک DB اشتراکی ایجاد کنیم. برای این منظور در پنجره ی SIMATC Manager راست کلیک نموده و مراحل زیر را پس از ایجاد OB1 طی کنید.





در قسمت Name and type نام و نوع DB را مشخص می کنیم.



با دوبار کلیک بر رویDB که در این مثال به نام DB1 ساخته ایم وارد آن شده و در ابتدا جدول Declaration Table را تکمیل می کنیم.

Address	Name	Type	Initial value
0.0		STRUCT	
+0.0	bit0	BOOL	TRUE
+0.1	bit1	BOOL	FALSE
+0.2	bit2	BOOL	TRUE
+0.3	bit3	BOOL	TRUE
+1.0	byte1	BYTE	B#16#AA
+2.0	int1	INT	100
+4.0	int2	INT	20
=6.0		END_STRUCT	

شکل ۵۹.۱۱

برای درک بیشتر عملکرد DB و چگونگی پر کردن جدول Declaration Table در قالب یک مثال به شرح کار و استفاده از DB می پردازیم.

مثال ۱۲.۱۱

برنامه ای با استفاده از DB به شرح زیر نوشته و عملکرد آن را بررسی کنید.

فرض کنید در سطری از یک DB اطلاعات 0000H وجود دارد . برنامه ای بنویسید که در اجرای هر سیکل برنامه ۱ واحد به مقدار موجود در این کلمه اضافه کند.

حل:

ابتدا DB1 را مطابق آنچه در بالا گفته شد می سازیم.

بر بروی DB1 ساخته شده دو بار کلیک نموده و آنرا باز ونسبت به تکمیل جدول Declaration Table مطابق اشکال زیر اقدام می کنیم.

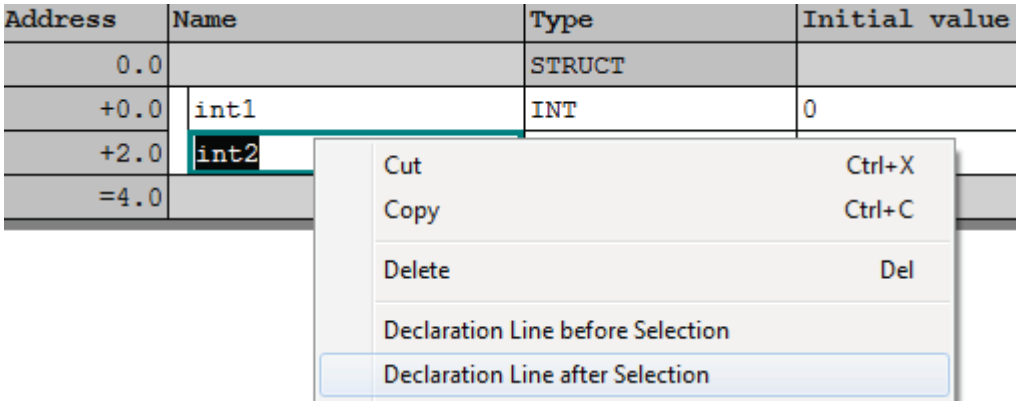
Address	Name	Type	Initial value	Comment
0.0		STRUCT		
+0.0	DB_VAR	INT	0	Temporary placeholder variable
=2.0		END_STRUCT		

شکل ۶۰.۱۱

Address	Name	Type	Initial value	Comment
0.0		STRUCT		
+0.0	int1	INT	0	Temporary placeholder variable
+2.0	int2	INT	1	
=4.0		END_STRUCT		

شکل ۶۱.۱۱

نکته ۱۰: در تکمیل جدول بعد از پر کردن هر ردیف برای ایجاد ردیف جدید ابتدا ردیف را انتخاب و سپس راست کلیک نموده و با انتخاب ... Declaration Line می توانیم قبل و یا بعد از ردیف انتخابی ردیفی اضافه کنیم.



شکل ۶۲.۱۱

نکته ۱-در پر کردن جدول در ستون آدرس Address چیزی تایپ نکنید. خودش پر می شود.

۲- در ستون Name اگر بخواهیم بیت اختصاص دهیم عبارت بیت به همراه شماره مثلاً Bit 0 و Bit2 و ... و اگر بخواهیم بایت اختصاص دهیم عبارت Byte1 ، Byte 2 و ... و اگر بخواهیم کلمه ، Word اختصاص دهیم Int1 ، Int2 و ... می نویسیم. در این مثال با توجه به صورت مسئله از دو کلمه Int1 ، Int2 مطابق با شکل زیر استفاده شده است.

۳- در ستون Type متناسب با نوع سیستم اعداد که در ستون Name انتخاب کرده ایم نوع سیستم عددی Int ، Byte ، Bool را انتخاب نموده و تایپ می کنیم.

۴- در ستون Inittial Value (عدد اصلی) می توان از عبارت True به معنی درست (حقیقی) و یا False به معنی غیر حقیقی استفاده کرد. یا به مانند این مسئله عدد را در این ستون وارد کرد.

نکته : جهت استفاده از DB باید آنها را به صورت زیر نامگذاری نمود:

۱-DBB1 و DBB2 و .. وقتی به صورت بایت باشد.

۲- DBW2 و DBW4 و ... وقتی به صورت کلمه باشد.

۳- در برنامه نویسی DB به ازاء هر Data Word در حافظه رزرو می شود.

۴-DB فقط به صورت غیر شرطی و با دستور OPN که در قسمت (DB Call) قرار دارد فراخوانی می شود.

۵- تا زمانی که DB جدید ی فراخوانی نشود DB قبلی همچنان باز بوده و آماده ی Load و Transfer می باشد.

۶- بلوک اطلاعاتی محدوده ای از حافظه می باشد که جهت ذخیره داده های ثابت و متغیر نظیر مقادیر عددی و .. مورد استفاده قرار می گیرد و بر خلاف سایر بلوک ها بلوک DB شامل برنامه نمی باشد.

۷- هر DB می تواند شامل ۲۵۶ کلمه ی اطلاعاتی (DATA Word) باشد.

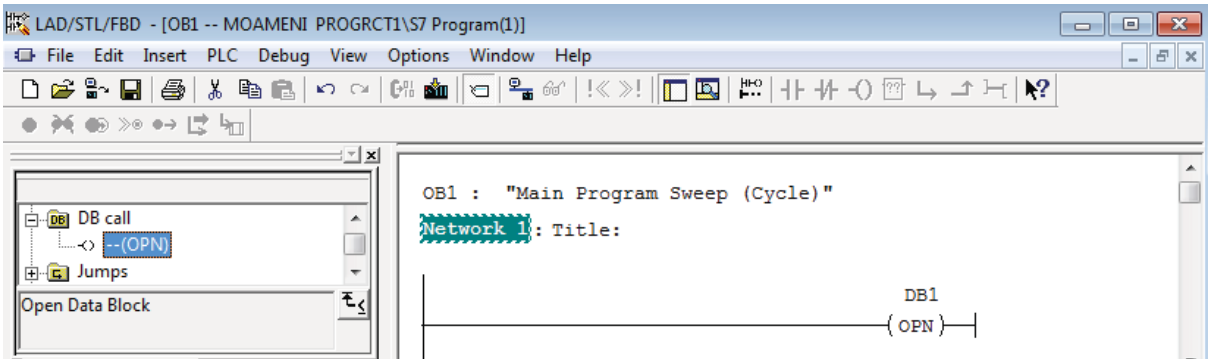
شکل ۶۳.۱۱

Address	Name	Type	Initial value	Comment
0.0		STRUCT		
+0.0	int1	INT	0	Temporary placeholder variable
+2.0	int2	INT	1	
=4.0		END_STRUCT		

در این مثال از DBW 0 و DBW2 استفاده شده است.

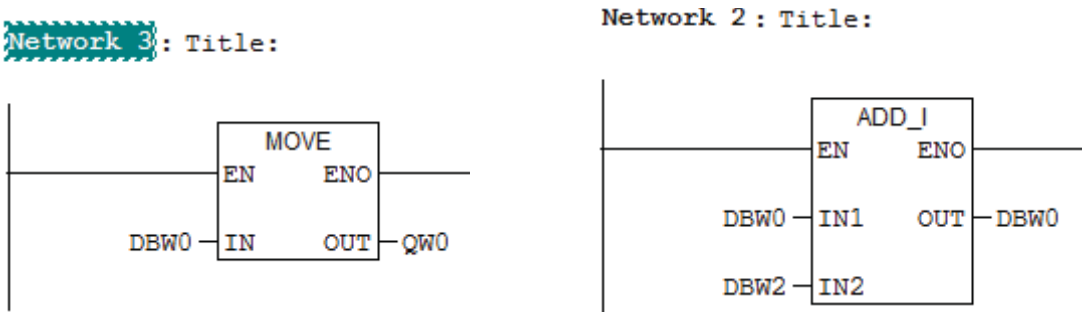
پس ازتکمیل جدول DB1 را ذخیره نموده و به PLC ارسال می کنیم.

حال از پنجره ی SIMATIC Manager بر روی OB1 دو بار کلیک نموده و مطابق شکل های زیر اقدام به نوشتن برنامه می کنیم .



شکل ۶۵.۱۱

برای فراخوانی DB از دستور OPN که در پنجره ی فهرست دستورات و در شاخه ی DB Call قرار دارد استفاده می کنیم. پس از انتخاب Network1 از بلوک OB1 از مسیر گفته شده با دو بار کلیک بر روی شاخه ی OPN آن را مشاهده نموده و نام آن را DB1(در این مثال) قرار می دهیم. Network2 ، Network3 را نیز مطابق با صورت مسئله برنامه نویسی می کنیم که در اشکال زیر نشان داده شده است.



پس از خاتمه ی برنامه نویسی آن را ذخیره و ارسال می کنیم. اکنون با قرار دادن PLC در وضعیت RUN می توانیم عملکرد برنامه و اجرا شدن آن را مشاهده کنیم.

نکته ۱۳: برای دیدن محتوای DB بعد از باز نمودن آن مراحل زیر را طی کنید:

DB1/view/DataView

مثال ۱۳.۱۱

برنامه ای با استفاده از DB بنویسید که بتوان سرعت اجرای یک سیکل زمانی اجرای برنامه توسط ریز پردازنده ی PLC را اندازه گیری نمود.

راهنمایی:

به طور مثال عدد ۲۰۰۰ را در DB می نویسم و در صورتی که برنامه ۲۰۰۰ بار اجرا شود با شروع برنامه زمان گرفته (به صورت دستی) و با روشن شدن Q2.0 زمان گیری را متوقف می کنیم. حال با نوشتن زمان و تقسیم آن بر ۲۰۰۰ می توان یک زمان اجرای سیکل برنامه را بدست آورد.

حل:

ابتدا DB2 را مطابق با آنچه در بالا گفته شد می سازیم.

بر روی DB2 ساخته شده دو بار کلیک نموده و آن را باز و نسبت به تکمیل جدول Declaration Table مطابق با شکل های زیر اقدام می کنیم.

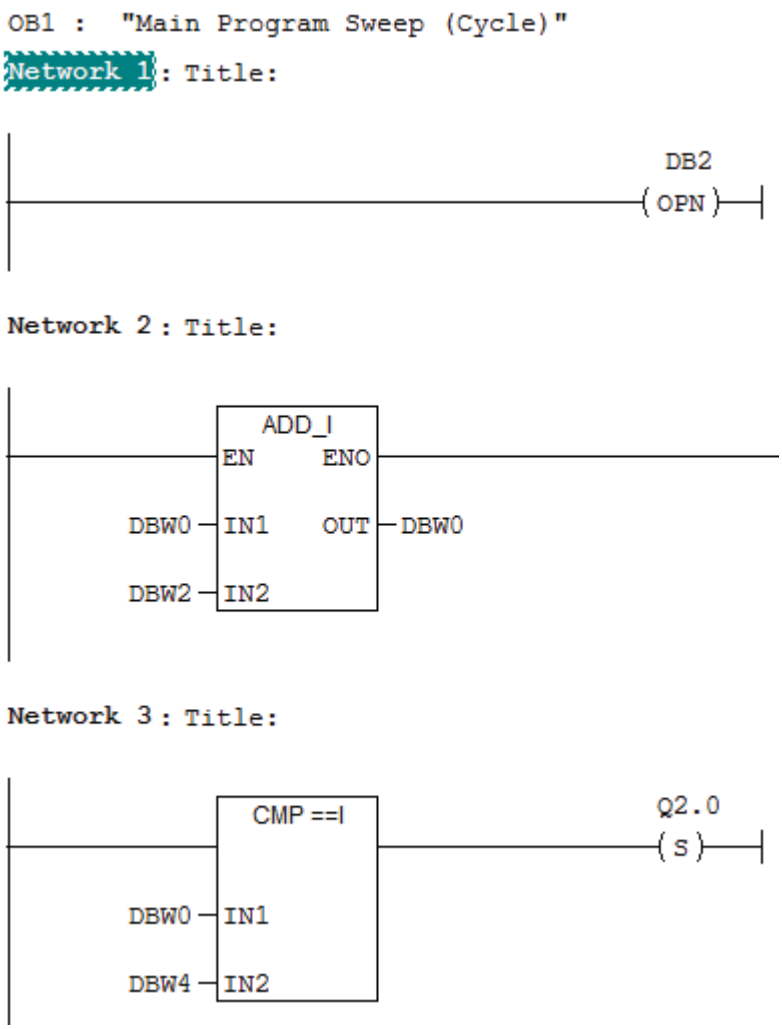
پس از تکمیل DB2 آن را ذخیره و ارسال می کنیم.

درپنجره ی SIMATIC Manager بر روی OB1 دو بار کلیک کرده ومتن برنامه را در سه Network مطابق با اشکال زیر نوشته و سپس آن را ذخیره و ارسال می کنیم.

PLC را در وضعیت RUN قرار داده و اجرای برنامه را مشاهده کنید.

مراحل در شکل های زیر نشان داده شده است.

Address	Name	Type	Initial value	Comment
0.0		STRUCT		
+0.0	INT1	INT	0	Temporary placeholder variable
+2.0	INT2	INT	1	
+4.0	INT3	INT	2000	
=6.0		END_STRUCT		

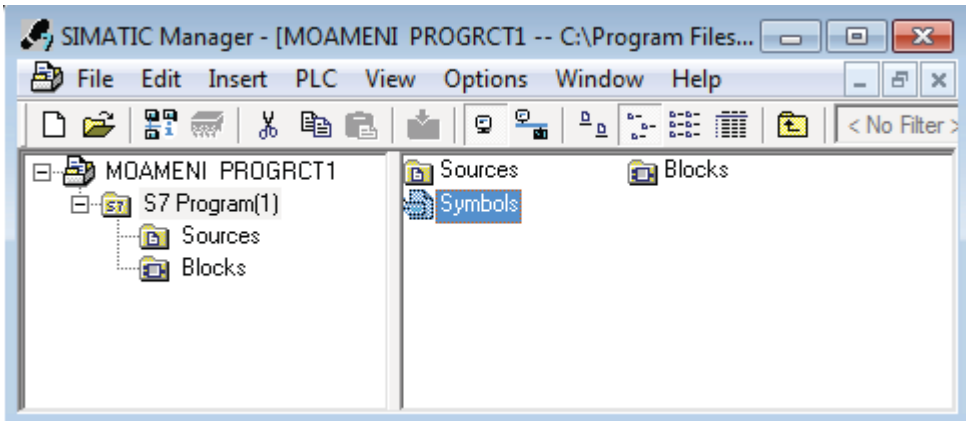


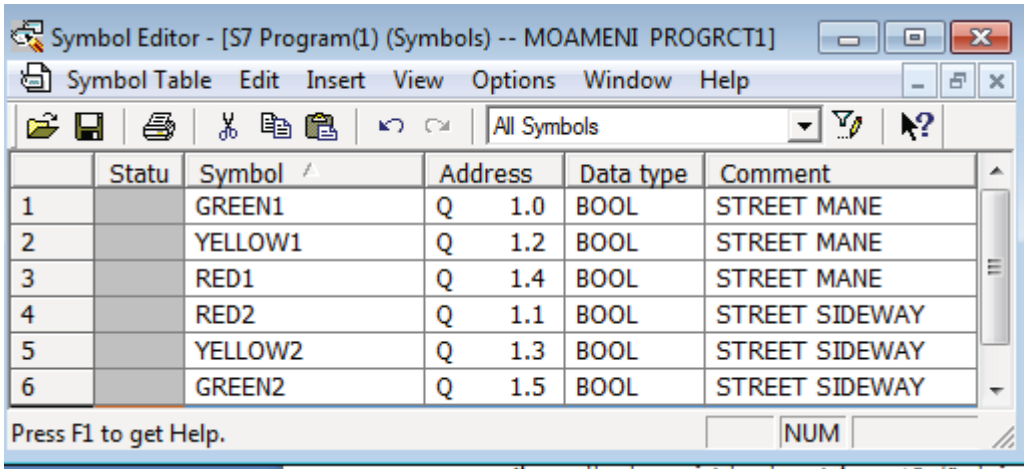
پس از نوشتن OB1 و ذخیره و ارسال به PLC با قرار دادن ماشین در وضعیت RUN ، اجراء برنامه را مشاهده کنید. برای تحلیل برنامه با انتخاب روش STL در برنامه ی OB1 مطابق با شکل زیر می توان مراحل برنامه را مورد نقد و بررسی قرارداد .

```
OB1 :  "Main Program Sweep (Cycle)"
Network 1: Title:
      OPN   DB     2


Network 2 : Title:
      L     DBW    0
      L     DBW    2
      +I
      T     DBW    0
      NOP   0

Network 3 : Title:
      L     DBW    0
      L     DBW    4
      ==I
      S     Q      2.0
```

مثال ۱۴.۱۱

برنامه ی چراغ های راهنمایی و رانندگی یک چهار راه با استفاده از DBبنویسید .فاصله ی زمان ها بین سبز ، زرد ، قرمز ، را در این مثال مساوی در نظر بگیرید.





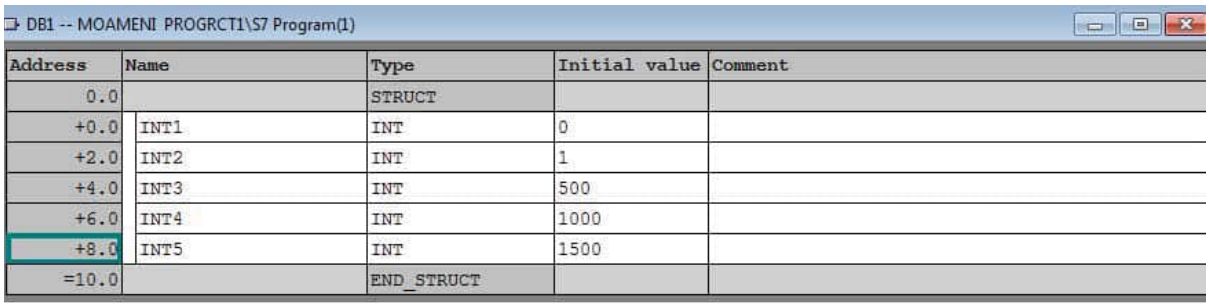
حل:

ابتدا بلوک DB3 و جدول آن را مطابق با تعداد مراحل و زمان بندی خواسته شده پر کرده و آن را ذخیره و ارسال می کنیم .

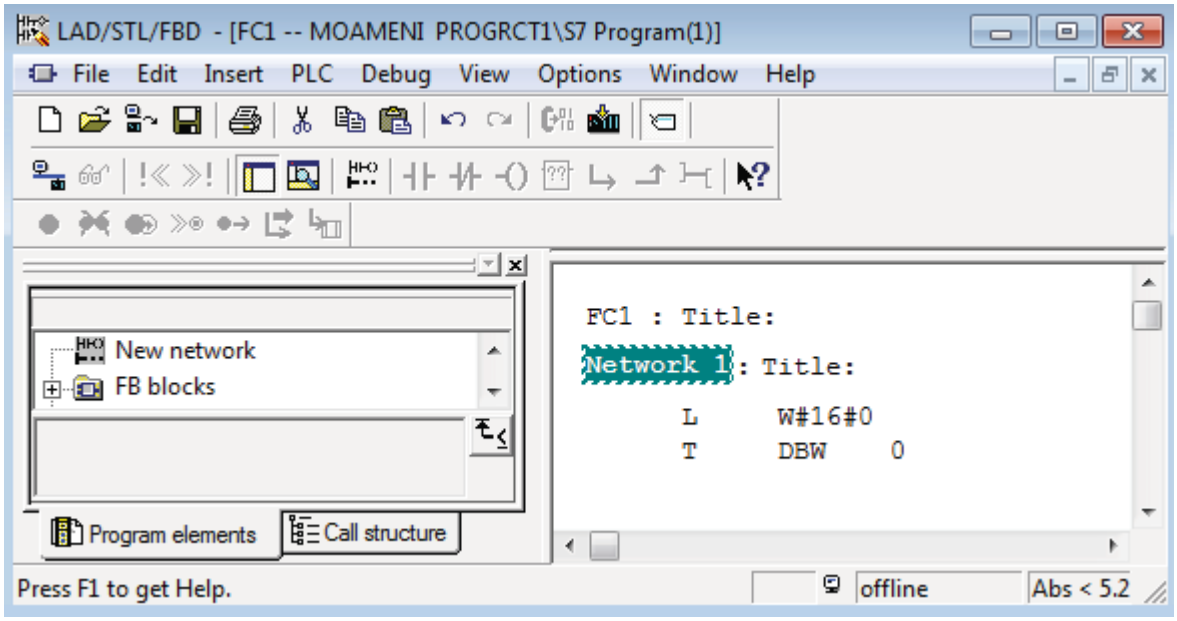
حال بلوک FC1 را به روشی که قبلاً گفته شد ایجاد کنید. آن را باز نموده و برنامه ی نشان داده شده در زیر را در آن نوشته و در خاتمه ذخیره و ارسال کنید.

علت استفاده از FC1 برای تکرار برنامه نویسی DBW 0 بلوک DB3 بعد از خاتمه یافتن تمام مراحل زمان گیری مجدداً با صفر پر می شود. برای انجام این کار از FC1 در با محتویات نشان داده شده در زیر استفاده شده است . بلوک FC1 در بلوک OB1 به صورت شرطی CCFC1 فراخوانی شده است.

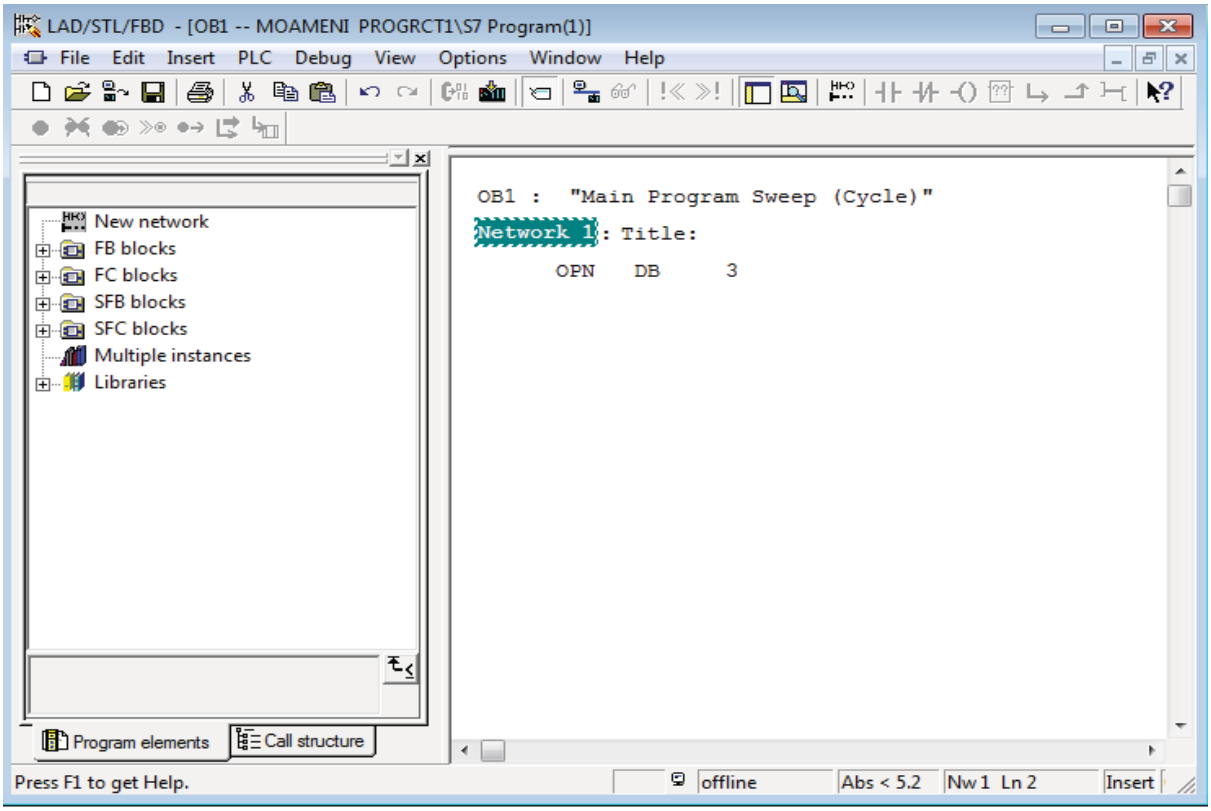
حال بلوک OB1 را باز نموده و محتویات نشان داده شده در زیر را به روش STL در آن وارد و در انتها آن را ذخیره و ارسال کنید.



شکل ۷۴.۱۱



شکل ۷۵.۱۱



Network 2: STREET MANE

```
L     DBW    0
L     DBW    2
+I
T     DBW    0
L     DBW    2
==I
S     "GREEN1"
S     "RED2"
R     "YELLOW1"
R     "YELLOW2"
R     "RED1"
R     "GREEN2"
L     DBW    0
L     DBW    4
==I
S     "YELLOW1"
S     "YELLOW2"
R     "GREEN1"
R     "RED2"
R     "RED1"
R     "GREEN2"
L     DBW    0
L     DBW    6
==I
S     "RED1"
S     "GREEN2"
R     "GREEN1"
R     "RED2"
R     "YELLOW1"
R     "YELLOW2"
L     DBW    0
L     DBW    8
==I
CC    FC     1
```

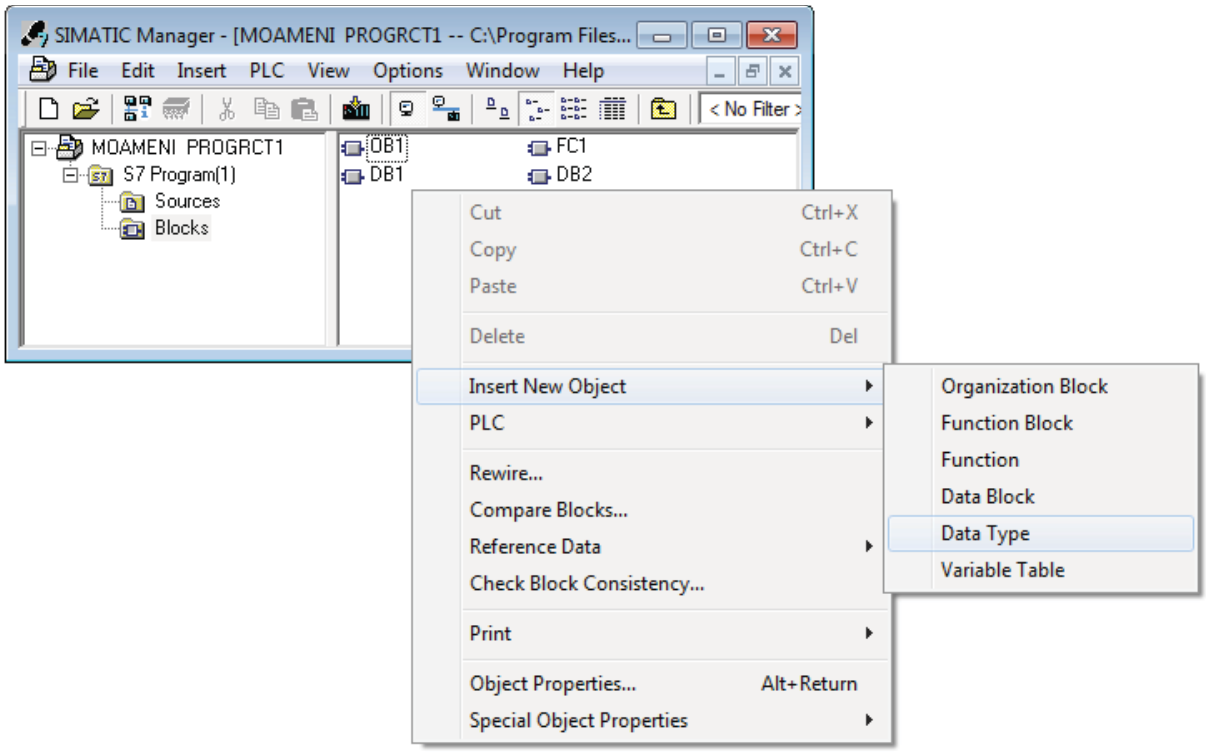
روش ایجاد و استفاده ی UDT (User Defined Data Type)

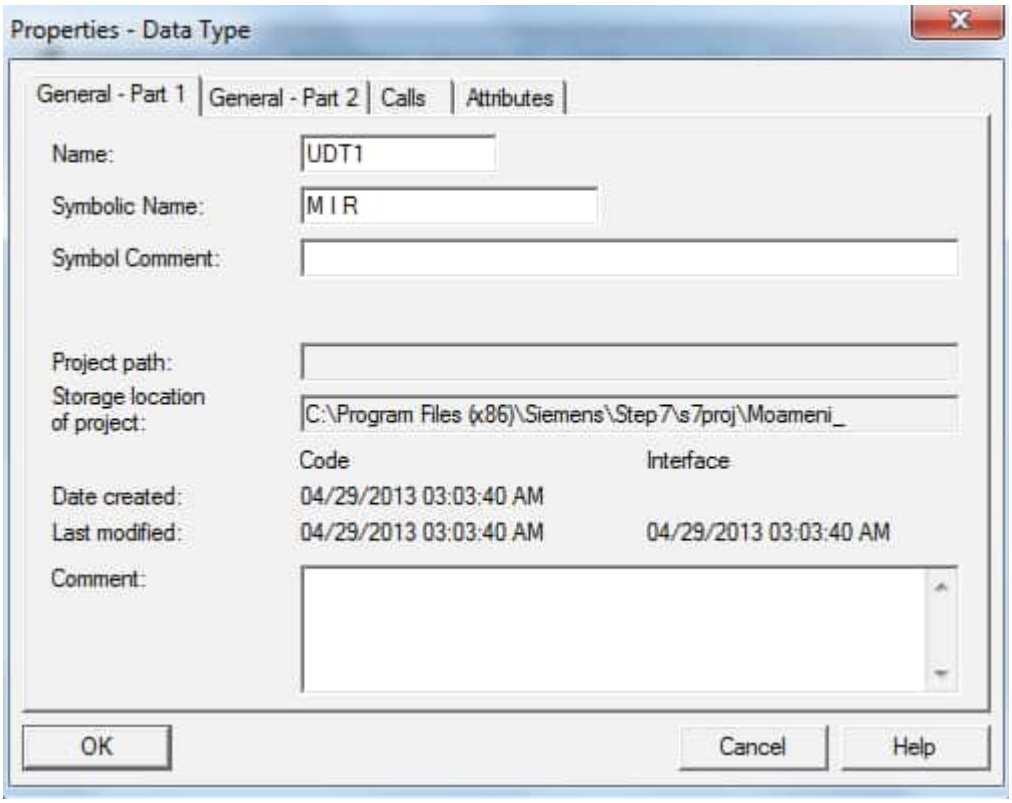
UDT یا

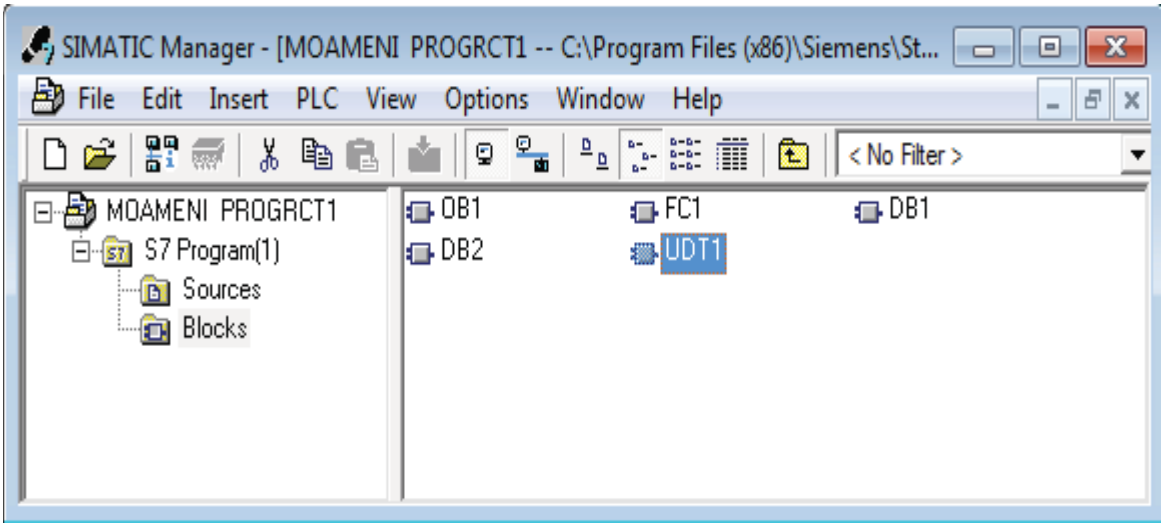
User Defined Data Type نیز از جمله بلوک هایی است که می توان آن را در پوشه ی Blocks برنامه ی SIMATIC Manager ایجادکرد. روش ایجاد آن مشابه FC و FB می باشد. و این بلوک یک متغیر جدید بر اساس متغیر هایی که داریم ساخته می شود .

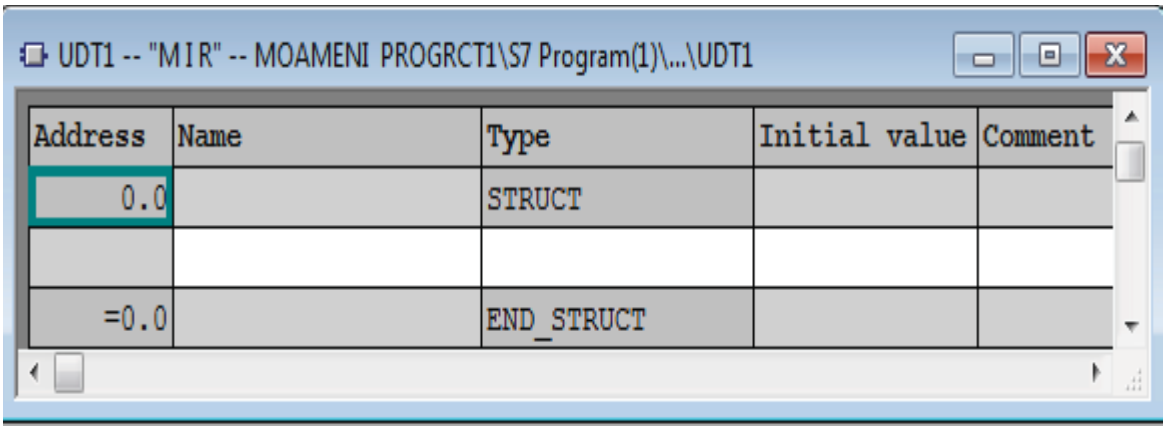
مثال ۱۵.۱۱

مراحل زیررا طی کنید تا باکاربرد UDT آشنا شوید.





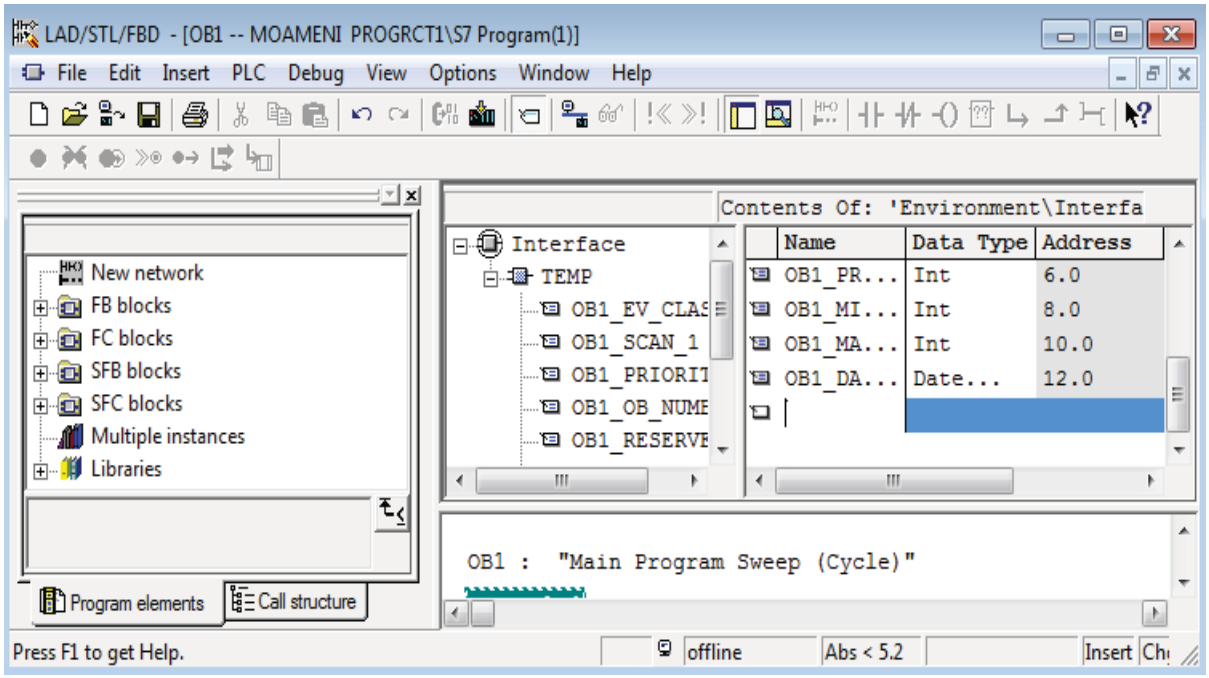




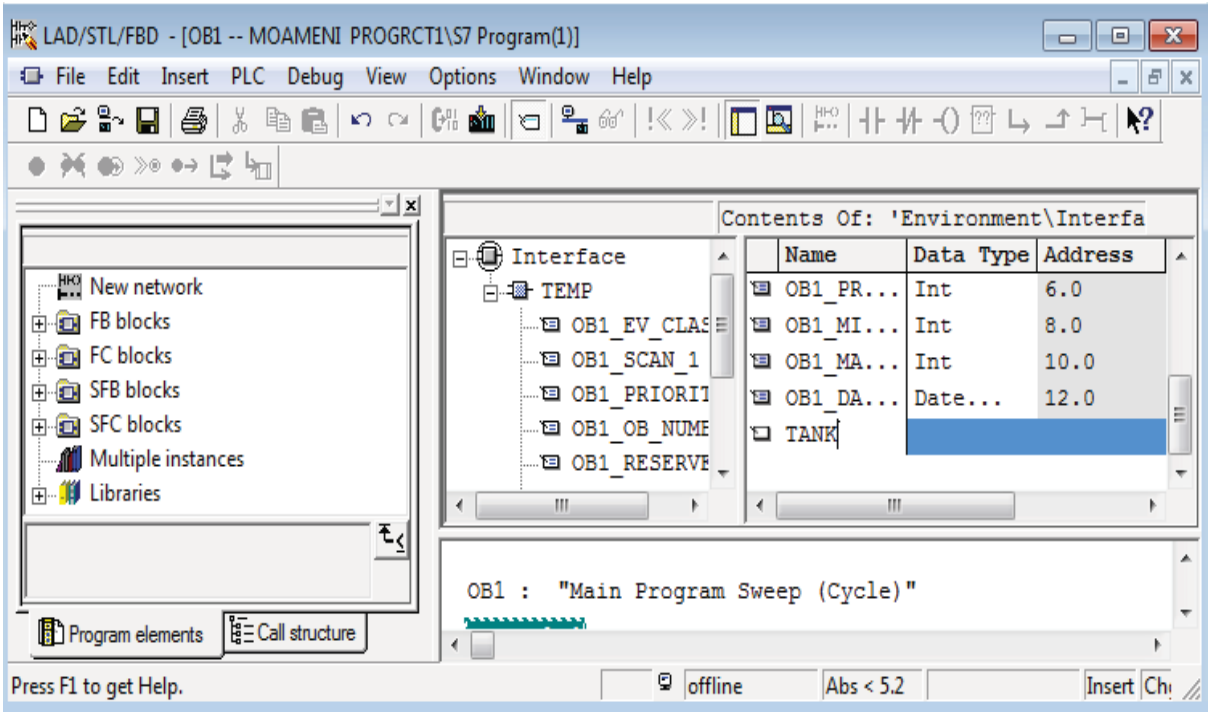
متغیر های زیر را در UDT1 وارد کنید.پس از تکمیل هر سطر کلید Enter را فشار دهید.

Address	Name	Type	Initial value	Comment
0.0		STRUCT		
+0.0	TEMP	INT	10	
+2.0	PRESS	INT	10	
+4.0	COOLER	BOOL	FALSE	
+4.1	HEATER	BOOL	FALSE	
=6.0		END_STRUCT		

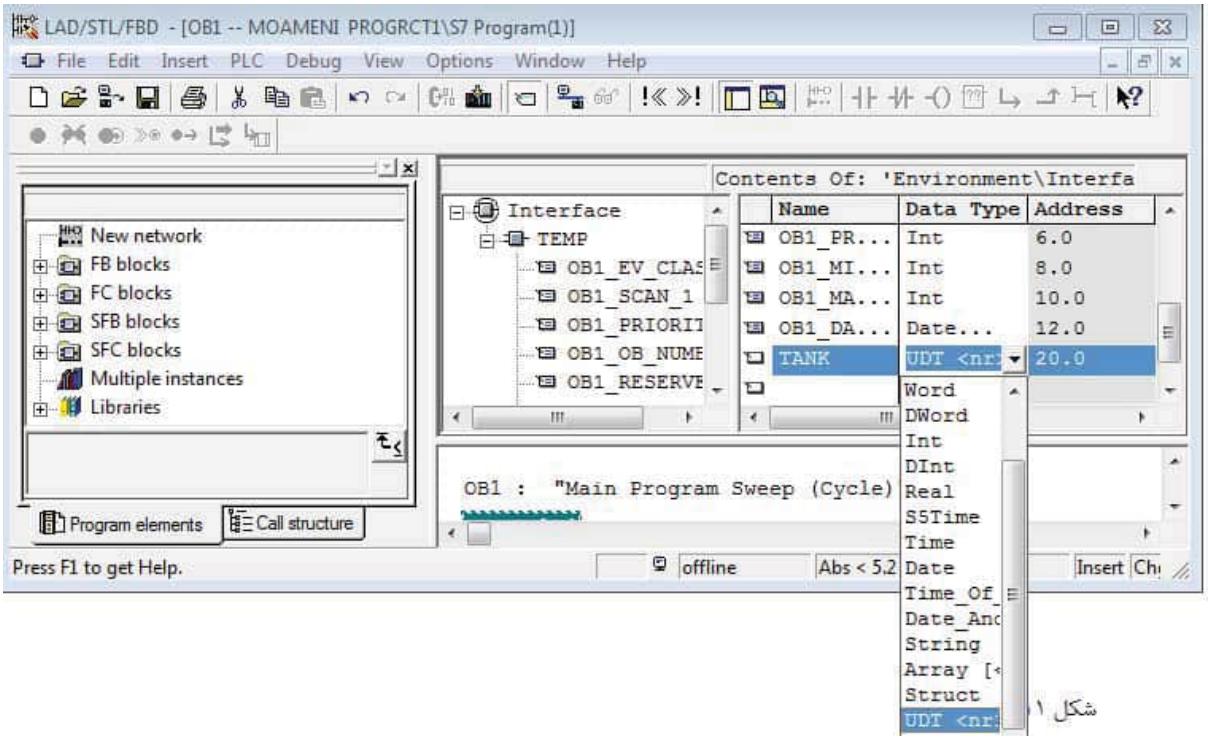
پس از کامل کردن UDT1 آن را ذخیره کنید و OB1 را باز نموده و مراحل زیر را طی کنید.


LAD/STL/FBD - [OB1 -- MOAMENI PROGRCT1\S7 Program(1)]
File Edit Insert PLC Debug View Options Window Help
New network
FB blocks
FC blocks
SFB blocks
SFC blocks
Multiple instances
Libraries
Program elements
Call structure
Contents Of: 'Environment\Interfa
Interface
TEMP
OB1_EV_CLAS
OB1_SCAN_1
OB1_PRIORIT
OB1_OB_NUME
OB1_RESERVE
Name
Data Type
Address
OB1_PR...
Int
6.0
OB1_MI...
Int
8.0
OB1_MA...
Int
10.0
OB1_DA...
Date...
12.0
OB1 : "Main Program Sweep (Cycle)"
Press F1 to get Help.
offline
Abs < 5.2
Insert

(الف)

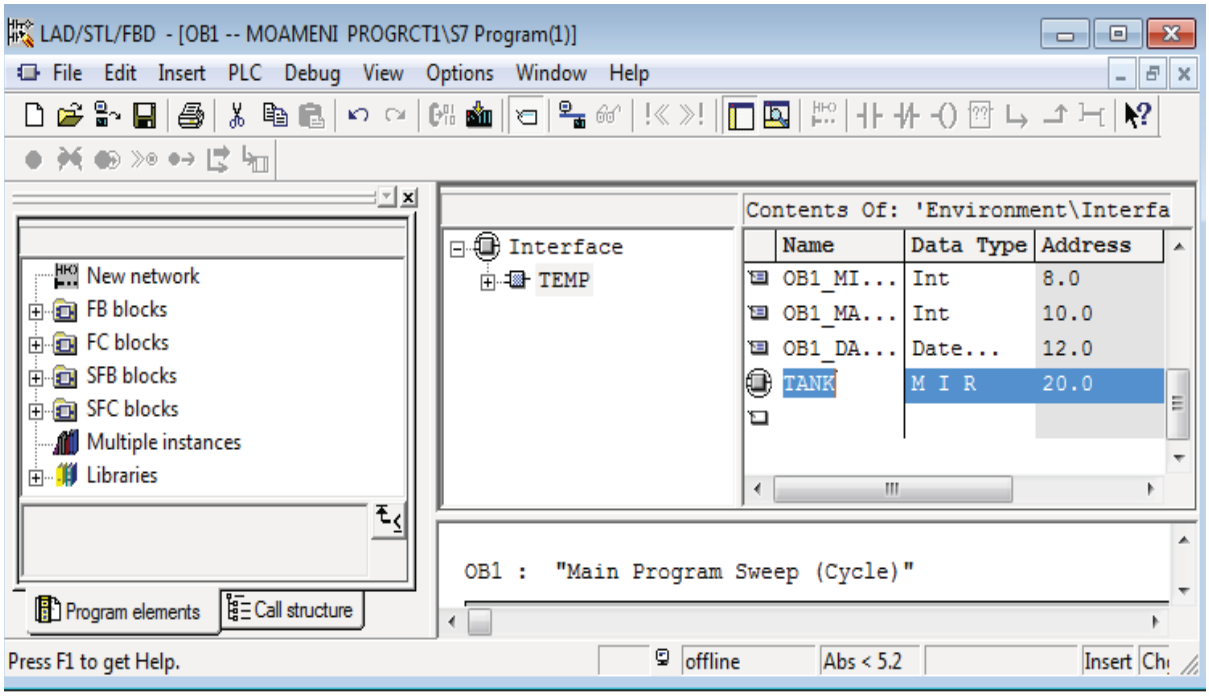

LAD/STL/FBD - [OB1 -- MOAMENI PROGRCT1\S7 Program(1)]
File Edit Insert PLC Debug View Options Window Help
New network
FB blocks
FC blocks
SFB blocks
SFC blocks
Multiple instances
Libraries
Program elements
Call structure
Contents Of: 'Environment\Interfa
Interface
TEMP
OB1_EV_CLAS
OB1_SCAN_1
OB1_PRIORIT
OB1_OB_NUME
OB1_RESERVE
Name
Data Type
Address
OB1_PR...
Int
6.0
OB1_MI...
Int
8.0
OB1_MA...
Int
10.0
OB1_DA...
Date...
12.0
TANK
OB1 : "Main Program Sweep (Cycle)"
Press F1 to get Help.
offline
Abs < 5.2
Insert

(ب)

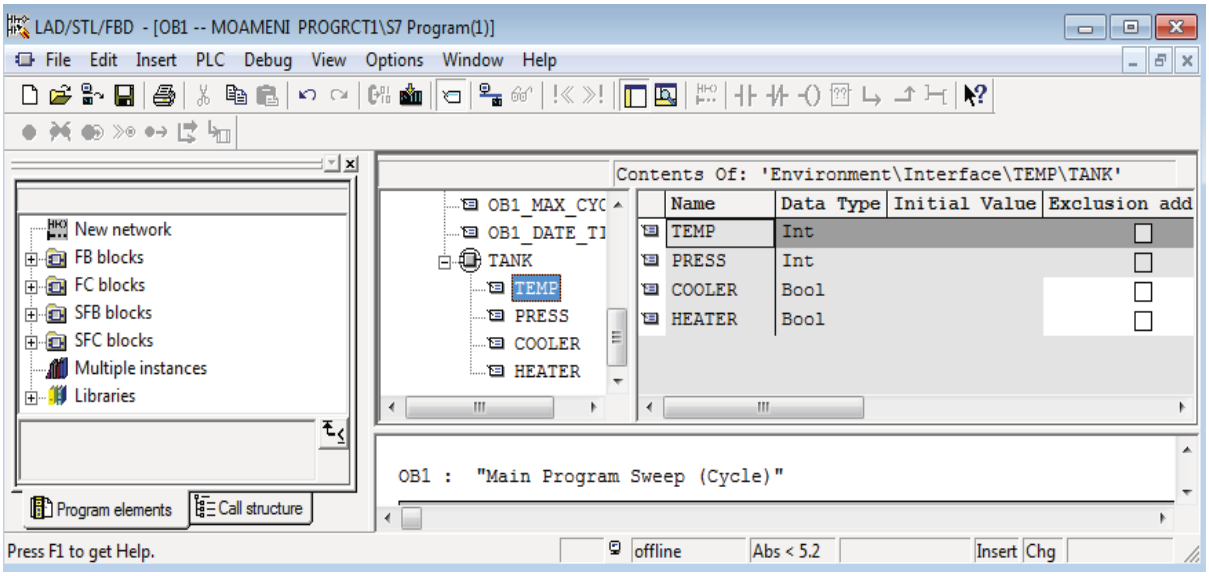

LAD/STL/FBD - [OB1 -- MOAMENI PROGRCT1\S7 Program(1)]
File Edit Insert PLC Debug View Options Window Help
New network
FB blocks
FC blocks
SFB blocks
SFC blocks
Multiple instances
Libraries
Program elements
Call structure
Contents Of: 'Environment\Interfa
Interface
TEMP
OB1_EV_CLAS
OB1_SCAN_1
OB1_PRIORIT
OB1_OB_NUME
OB1_RESERVE
Name
Data Type
Address
OB1_PR...
Int
6.0
OB1_MI...
Int
8.0
OB1_MA...
Int
10.0
OB1_DA...
Date...
12.0
TANK
UDT <nr
20.0
Word
DWord
Int
DInt
Real
S5Time
Time
Date
Time_Of
Date_And
String
Array [
Struct
UDT <nr
OB1 : "Main Program Sweep (Cycle)
Press F1 to get Help.
offline
Abs < 5.2
Insert

شکل ۱

(ج)

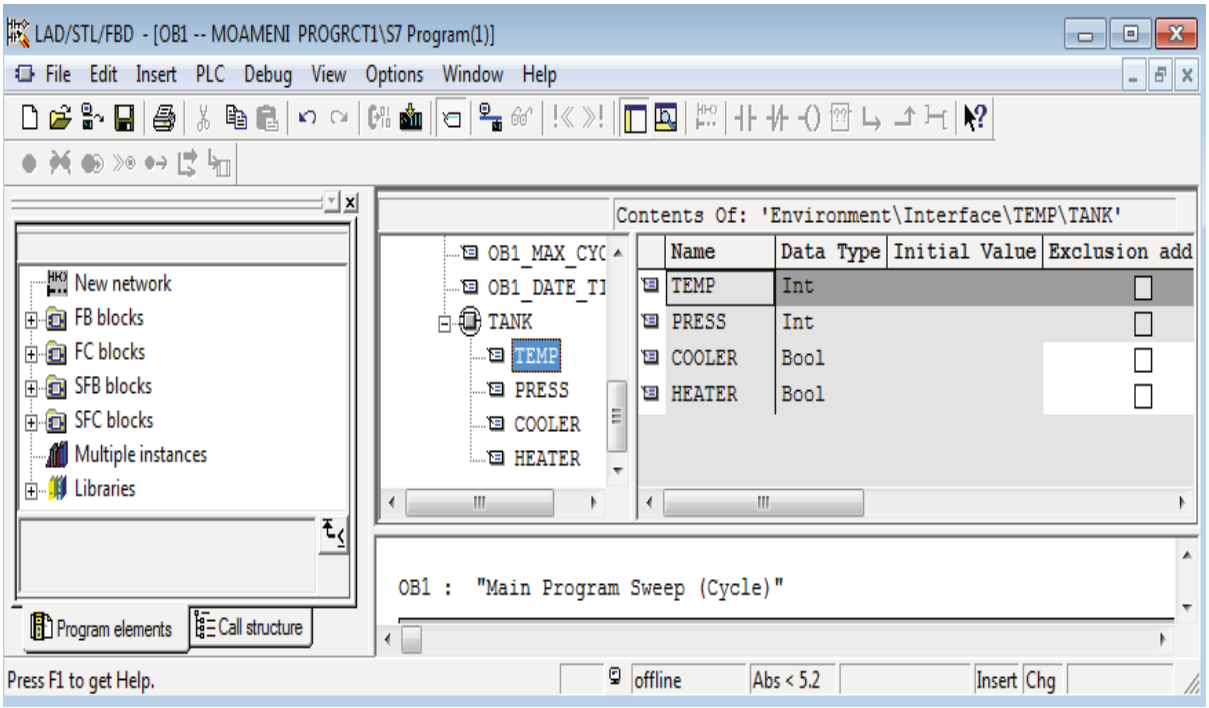

LAD/STL/FBD - [OB1 -- MOAMENI PROGRCT1\S7 Program(1)]
File Edit Insert PLC Debug View Options Window Help
New network
FB blocks
FC blocks
SFB blocks
SFC blocks
Multiple instances
Libraries
Program elements
Call structure
Contents Of: 'Environment\Interfa
Interface
TEMP
Name
Data Type
Address
OB1_MI...
Int
8.0
OB1_MA...
Int
10.0
OB1_DA...
Date...
12.0
TANK
M I R
20.0
OB1 : "Main Program Sweep (Cycle)"
Press F1 to get Help.
offline
Abs < 5.2
Insert

(د)



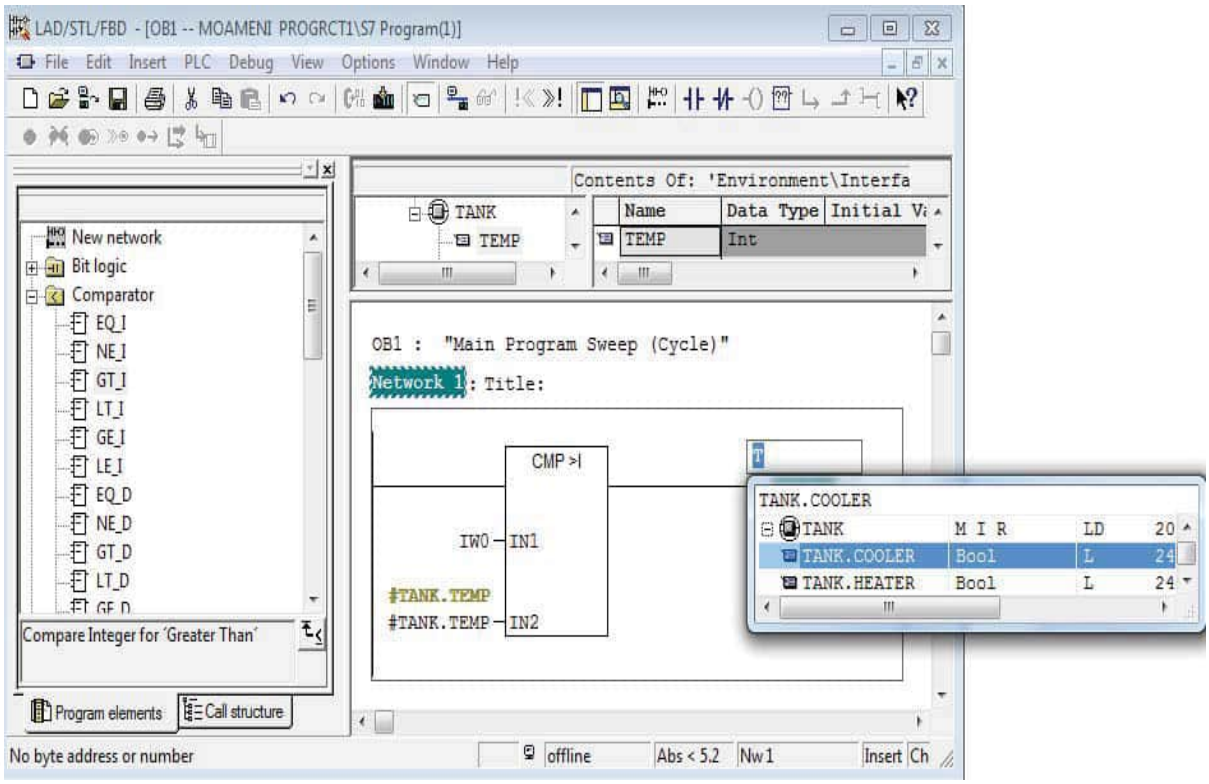
شکل ۸۴.۱۱

سطر مشخص شده را در انتهای جدول وارد می کنیم . اگر متغیر جدید که به نام Tank معرفی شده است را باز کنیم مشاهده می شود که در این بلوک یک متغیر جدید بر اساس متغیر هایی که داریم ساخته شده است .

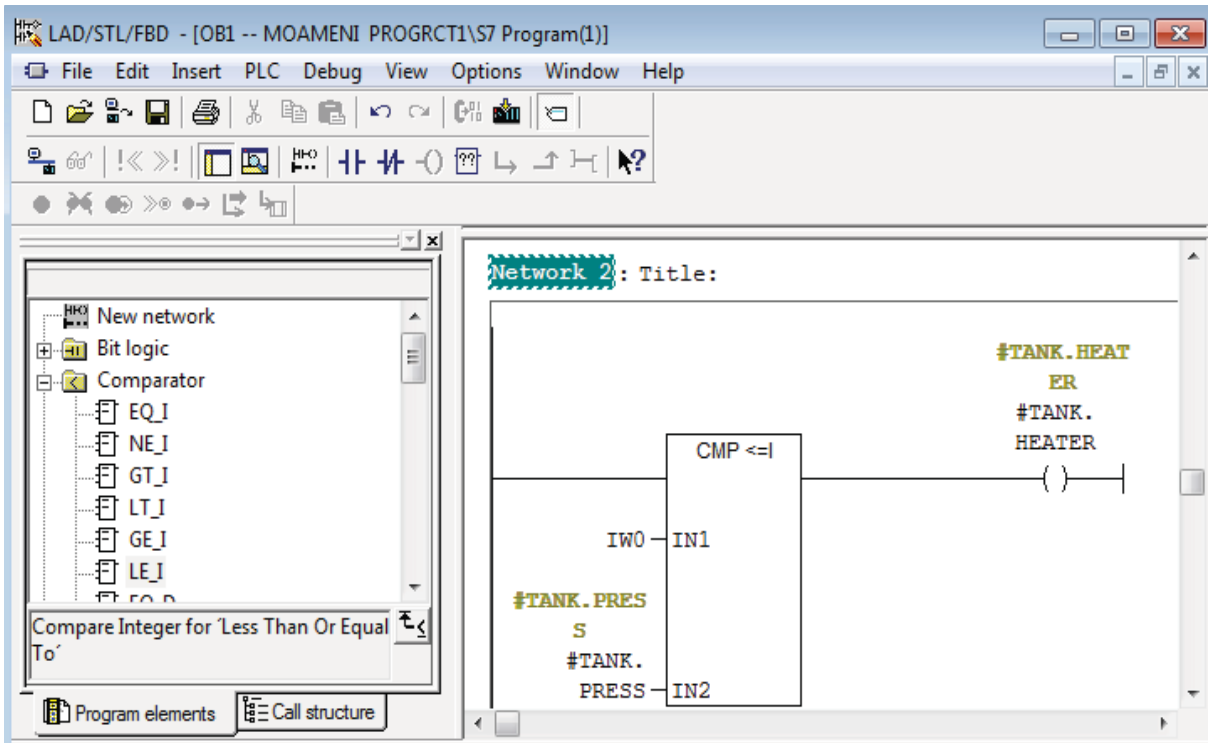


(الف)

(ب)



شکل ۸۵.۱۱



شکل صفحه قبل

نکته: UDT ها حاوی داده هایی هستند که یک بار تعریف شده ولی به مراتب می توانند مورد استفاده قرار گیرند .

۶.۱۱ انواع متغیر های S7

۱-متغیر عمومی یا Global

متغیرهایی هستند که در تمامی Block ها شناخته شده اند و نیازی به تعریف ندراند ، مانند M,Q,I که به معنای M=Memoty , Q=Output , I=Input می باشند.

۲-متغیر محلی با Local

متغیر هایی هستند که فقط در همان بلوک یا ناحیه ای که تعریف شده اند قابل دسترسی هستند و خارج از آن ناحیه یا تعریف نشده اند ، یا قابل دسترسی نیستند . ماند Data Block ها که L و DB به عنوان حافظه های محلی در S7 تعریف شده اند

PB	GLOBAL	LOCAL
OB1	M4.3 1 0	L20.3 1
FC1	1 0	0
FB4	1 0	1

با توجه به جدول فوق متوجه می شویم که اگر متغیری به عنوان Global معرفی شود در تمامی برنامه ها یکسان عمل می کند. یعنی اگر وضعیت متغیر را تغییر دهیم در تمامی بلوک ها تغییر اعمال خواهد شد ولی در مورد متغیر Local این چنین نیست.

تذکر: آدرس متغیرهای Local از بایت ۲۰ شروع می شود.

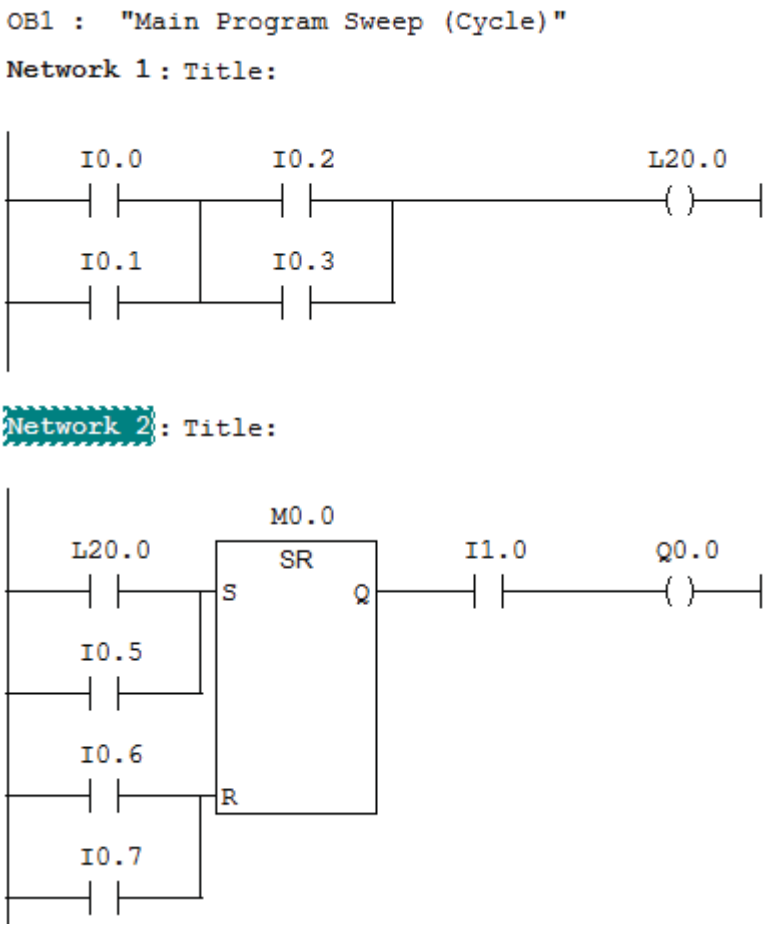
یادآوری: برای متغیرهای یک بیتی می توانیم پسوند قرار ندهیم . اگر بخواهیم این گونه متغیرها را با پسوند استفاده کنیم از X می توان استفاده نمود. به عنوان مثال I4.3 را به صورت IX 4.3 نیز می توان نمایش داد.

یادآوری: برای متغیرهای ۸ بیتی از پسوند B(مانند IB 0 ، QB 4 ، LB 20) و برای متغیرهای ۱۶ بیتی از پسوند W ، برای متغیرهای ۳۲ بیتی از پسوند D و برای متغیرهای ۶۴بیتی از پسوند L استفاده می کنیم.

یادآوری: در سمبل نویسی متغیر عمومی با علامت کوتیشن (" ") و متغیر محلی با علامت شارپ (#) نمایش داده می شود.

مثال ۱۶.۱۱

برنامه زیر را در نظر گرفته و عملکرد آن را توضیح دهید.



در این برنامه L به معنای حافظه ی محلی یا Local می باشد. چون Network ۱ و Network ۲ در یک OB نوشته شده است ، می توان از نتیجه Network ۱ در Network های دیگر استفاده کرد.

در موقع استفاده از حافظه های محلی سعی کنید از آدرس های ۲۰ بایت اول استفاده نشود. زیرا این آدرس ها توسط خود سیستم برای OB ها و موارد دیگر استفاده می شود.

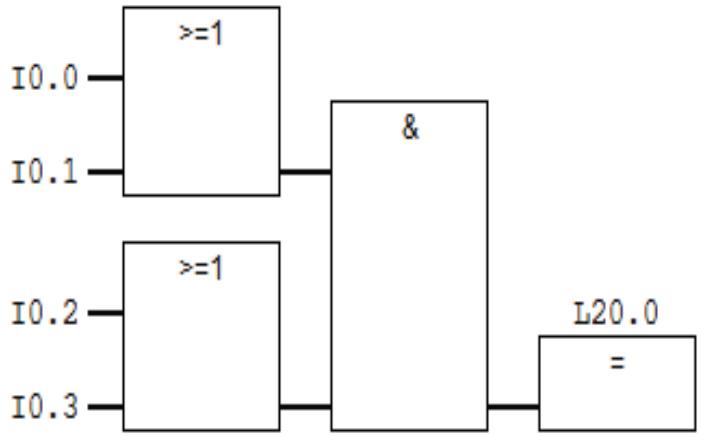
برنامه ی فوق را اجرا نموده تا با کاربرد حافظه های محلی بیشتر آشنا شوید.

مثال ۱۷.۱۱

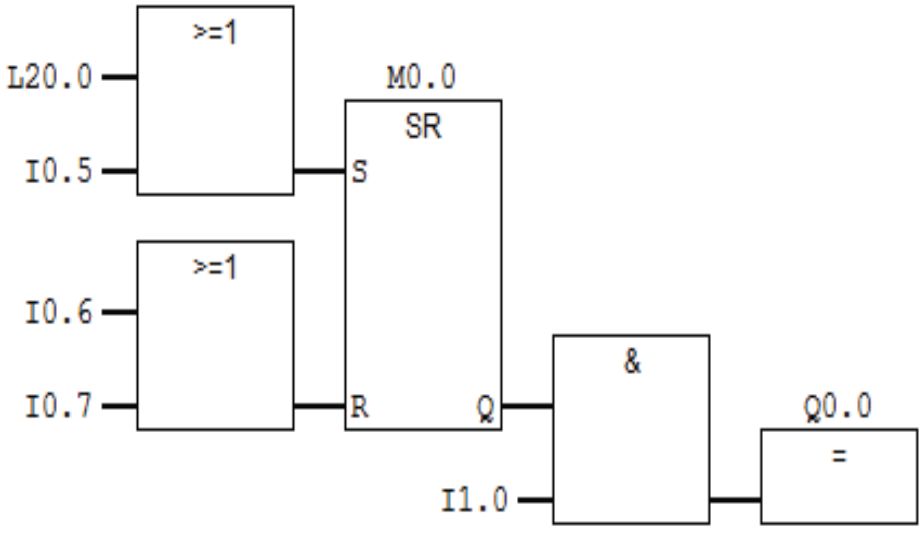
برنامه ی زیر را نوشته و آن را به روش های STL و FBD تبدیل نموده و Statues برنامه را مشاهده کنید.

OB1 : "Main Program Sweep (Cycle)"

Network 1 : Title:



Network 2 : Title:



```
OB1 :  "Main Program Sweep (Cycle)"
Network 1 : Title:
      A(
      O     I     0.0
      O     I     0.1
      )
      A(
      O     I     0.2
      O     I     0.3
      )
      =     L     20.0
```

```
Network 2: Title:
      A(
      A(
      O     L     20.0
      O     I     0.5
      )
      S     M     0.0
      A(
      O     I     0.6
      O     I     0.7
      )
      R     M     0.0
      A     M     0.0
      )
      A     I     1.0
      =     Q     0.0
```

توجه : در حالت STL از L 20.0 در سه جا استفاده شده است .(به محل های انشعاب دقت کنید)

با توجه به مثال فوق به نکات زیر توجه کنید:

۱-سیستم به طور اتوماتیک از حافظه های محلی استفاده می کند . اگر قبلاً از حافظه های محلی استفاده می کند .

(بایت های ۰.۱۹ توسط خود سیستم استفاده شده است).

۲-اگر از خروجی یک فیلپ فلاپ استفاده نکنیم به جای آن از دستور NOP O در S5 استفاده می کنیم. طرز کار BLD 1 0 2 را خواهید دید ، و به تعداد انشعاب ها از حافظه ی محلی استفاده می شود . BLD های دیگر برایم مقاصد دیگر به کار می روند .

۳-دستور NOP O به معنای NO Operation می باشد و تنها بخشی از برنامه STL که حذف شده قابل تبدیل به LAD و FBD باشد به کار می رود ، که آن شانزده صفر می باشد.

پرسش و پاسخ

۱-با استفاده از بلوک ..... می توان از تکرار برنامه هایی که قرار است ، چندین بار در برنامه اجرا شوند اجتناب نمود .

الف.OB ب.FB ج.UTD د. DB

۲- کدام بلوک دارای حافظه می باشد؟

الف. FB ب.FC ج.DB د.OB

۳- کدام بلوک شامل اطلاعات مربوط به پیام ها و پارامترها می باشد؟

الف. OB ب.DB ج.FB د.FC

۴- کدام بلوک در S7 شبیه PB در S5 استفاده می شود ؟

الف. FB ب. FC ج.DB د.الف و ب

۵- PLC برای اجرای برنامه به .... مراجعه می نماید.

الف.OB1 ب. OB22 ج. OB 34 د.OB 100

۶-..... معادل بلوک سازماندهی است.

الف. DB ب.FB ج.FC د.OB

۷- ..... معادل بلوک برنامه می باشد.

الف. OB ب.PB ج.FB د.SB

۸- در بلوک ..... خواندن اطلاعات بدون شرط انجام می شود.

الف. PB ب.DB ج.FB د.SB

۹- DB های اختصاصی برای بلوک ..... به عنوان حافظه عمل می کند.

الف.PB ب.DB ج.FB د.SB

۱۰-..... ها از بلوک های برنامه ریزی هستند.

الف. OB ب.DB ج.FB د.SB

۱۱-بلوک های .....تابع سازگار با IEC استفاده می شوند.

الف. سازماندهی ب .تبدیل S5-S7

ج. تبدیل T1-S7 د. IEC Function Block

۱۲- در PLC های خانواده S7- 3 0 0 بلوک برنامه نویسی FB چه تفاوتی با FC دارد ؟

الف.FB ها قابل برنامه نویسی نیستند ولی FC قابل برنامه نویسی هستند.

ب.FB ها داری حافظه هستندولی در FC ها امکان ذخیره سازی اطلاعات وجود ندارد.

ج.FB ها باید با DB مربوطه فراخوانی شوند ولی FC ها به تنهایی فراخوانی می شوند.

د. موارد الف و ج

فصل هفتم

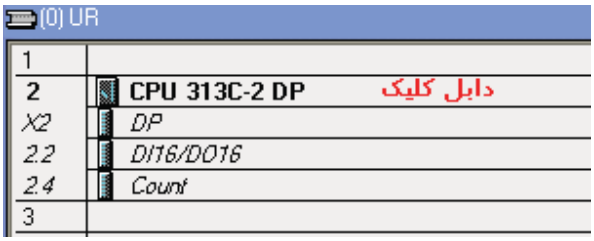
تنظیمات پردازنده (CPU)

در این فصل با مباحث زیر آشنا خواهید شد:

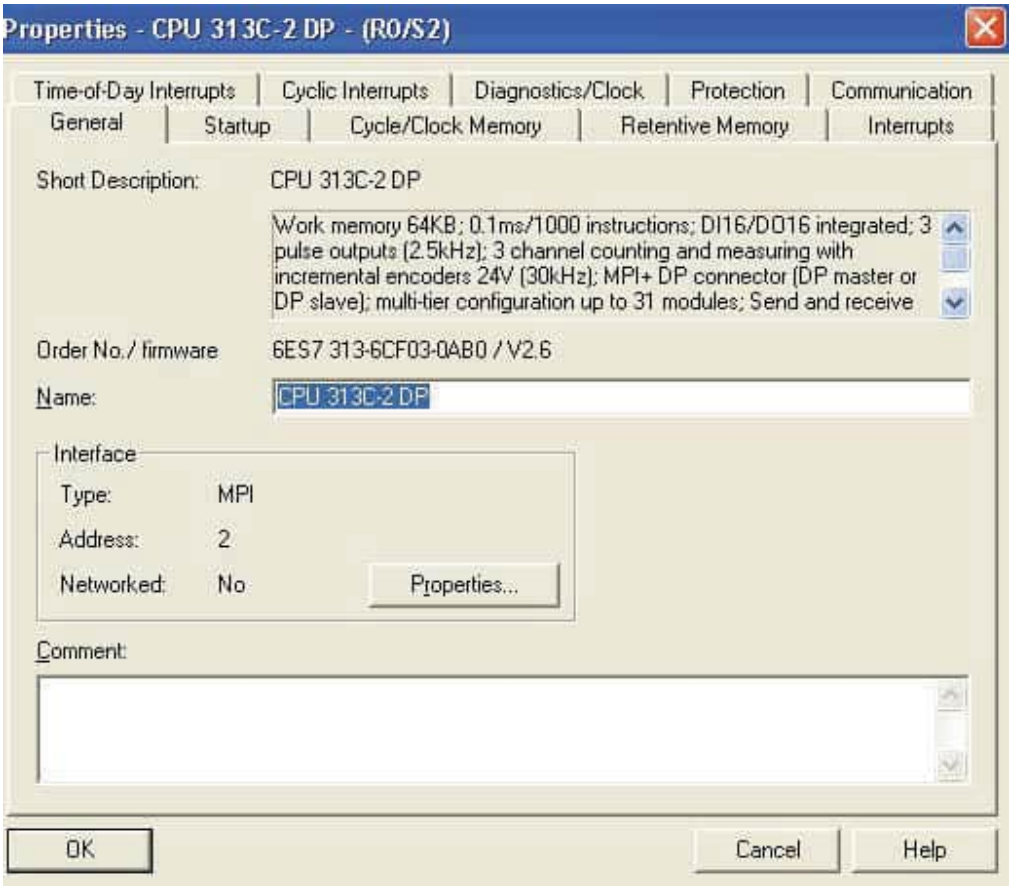
۱. تنظیمات نرم افزاری CPU
۲. تنظیمات وقفه ها

تنظیمات CPU :

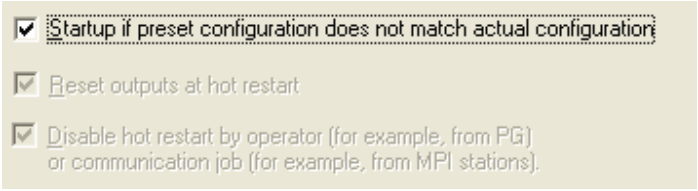
بعد از انجام عملیات پیکر بندی سخت افزاری کاربر می تواند بر روی CPU خود تنظیمات دلخواه مانند شبکه ، حافظه های ماندگار ، شمارنده های ماندگار و ... را اعمال نماید. برای این منظور کاربر باید به صفحه پیکربندی سخت افزار (Hardware Configuration) رفته و در صفحه سمت چپ بر روی نام CPU خود دابل کلیک نماید.



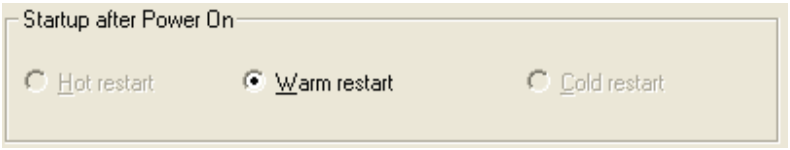
در صفحه باز شده زیر اطلاعات مربوط به تنظیمات CPU در لبه های مختلفی قرار داده شده است.در صفحه اصلی اطلاعات کلی در باره نام CPU و همچنین نوع واسط ارتباطی نشان داده شده است.



لبه Startup : در این لبه تنظیمات مربوط به لحظه راه اندازی CPU می باشد. گزینه های غیر فعال مربوط به CPU هایی می باشد که دارای این قابلیت باشند که اکثراً مربوط به سری 400 است. با فعال بودن تیک این گزینه زمانیکه سیستم واقعی با سیستم پیکربندی شده در نرم افزار یکی نباشد نیز اجازه راه اندازی می دهد.



اگر CPU خاموش و دوباره روشن شود راه اندازی آن به 3 صورت امکان پذیر است که در CPU های سری 300 یک نوع آن فعال می باشد.



Hot restart: این نوع راه اندازی فقط در سری 400 وجود دارد. در این راه اندازی ، پردازش برنامه از جایی ادامه می یابد که تغذیه قطع شده است. برای مثال اگر در خط10 ،تغذیه قطع شود با وصل تغذیه برنامه از خط 11 شروع به کار می کند. در این وضعیت تایمرها و شمارنده ها و فضای حافظه نیز ریست نمی شوند. برای استفاده از این حالت راه اندازی نیاز به باتری پشتیبان می باشد.

Warm restart: در این نوع راه اندازی ، پردازش برنامه از خط اول آغاز می گردد و تایمر ها و شمارنده هایی که به صورت پایدار تعریف نشده اند و همچنین فضای حافظه نیز ریست می گردند.

Cold restart: در این نوع راه اندازی ، پردازش برنامه از خط اول شروع شده و تمامی فضای حافظه ، کانترها و تایمرها ریست می گردند حتی اگر به صورت پایدار (Retentive) تعریف شده باشند.

در این نمونه CPU فوق فقط راه اندازی Warm فعال می باشد.

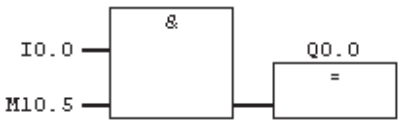
دوره زمانی / حافظه ساعت: در این لبه تنظیمات مربوط به حداکثر و حداقل دوره زمانی موجود می باشد که بسته به نوع CPU در برخی مواقع تنظیم حداقل، غیر فعال می باشد.

در قسمت مربوط به Clock Memory می توان با فعال کردن تیک مربوط به آن ، بایتی از فضای حافظه را انتخاب نمود در این صورت با آغاز به کار CPU ، در بیت های این بایت یک موج مربعی خواهید داشت که فرکانس هر یک متفاوت و به شرح زیر است:

شماره بیت	7	6	5	4	3	2	1	0
دوره تناوب (ثانیه)	2	1.6	1	0.8	0.5	0.4	0.2	0.1
فرکانس	0.5	0.625	1	1.25	2	2.5	5	10

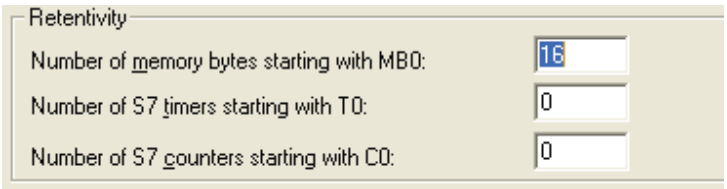
برای مثال اگر کاربر در روند برنامه خود از M10.3 استفاده نماید آنگاه این فضای حافظه با فرکانس 2Hz صفر و یک خواهد شد.

مثال : برنامه ای بنویسید که با فعال شدن ورودی I0.0 خروجی Q0.0 نیز با فرکانس 1Hz روشن و خاموش شود.



پاسخ : کافیست در تنظیمات CPU حافظه مورد نظر زمان را انتخاب کنید که در اینجا MB10 می باشد آنگاه با یک بلوک AND به راحتی می توان این روند را تولید نمود.

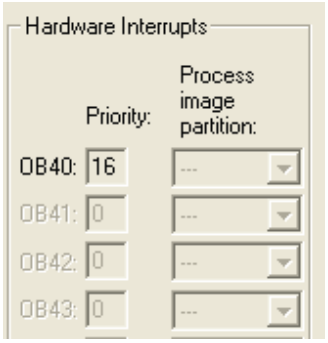
حافظه پایدار Retentive Memory: در این لبه می توان مکان دلخواه از حافظه و همچنین تایمرها و شمارنده های مورد نظر را به صورت پایدار تعریف نمود.



در ردیف اول تعداد بایت های فضای حافظه با شروع از صفر را می توان وارد نمود. در ردیف دوم تعداد تایمرها و در نهایت هم تعداد شمارنده ها را می توان مشخص کرد . برای مثال اگر در ریف دوم عدد 5 را وارد نمائید به این معنی است که تایمر های T0,T1,T2,T3,T4 به عنوان تایمرهای پایدار تعریف شده اند.

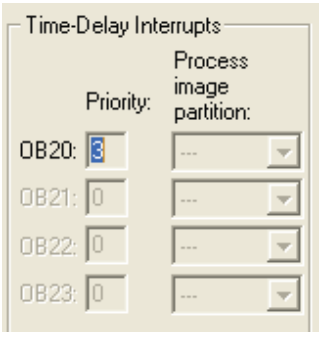
قسمت پائین این لبه مربوط به DB ها می باشد که در سری 300 فعال نیست.

وقفه ها (Interrupts) : با مبحث مربوط به وقفه ها در آینده آشنا خواهید شد. لبه مربوط به تنظیمات آن در CPU در اینجا صورت می گیرد.

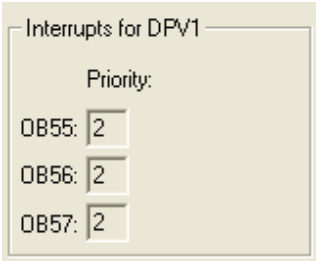


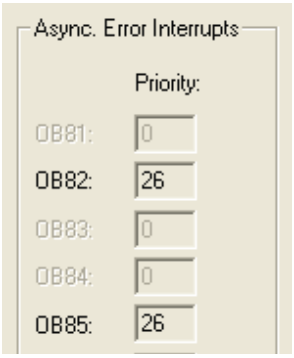
OB مربوط به وقفه های سخت افزاری به این صورت انتخاب می گردد که بسته به مدل CPU تعداد این OB ها متفاوت می باشد.

وقفه های تأخیر زمانی را می توان تعیین نمود.



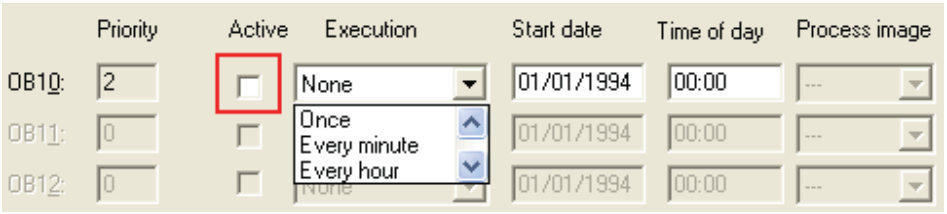
Interrupts for DPV1 وقفه های مربوط به شبکه پروفیباس می باشد.





این قسمت مربوط به تنظیمات وقفه های نا هماهنگ می باشد. برای مثال ممکن است کارت حافظه ناگهان بسوزد یا CPU برای انجام محاسبات فضای کافی در اختیار نداشته باشد در این صورت OB مربوطه اجرا خواهد شد.

لبه وقفه های زمانی از روز (Time of Day): در این لبه کاربر می تواند با وارد کردن زمان مورد نظر خود و همچنین تاریخ آغاز عملیات ، برنامه ای را که در OB مربوطه که در اینجا OB10 می باشد. البته باید دقت نمود که تیک مربوط به فعال کردن این وقفه باید زده شود.



در قسمت Execution (اجرا) کاربر می تواند نحوه اجرای برنامه نوشته شده در OB مر بوطه را تعیین نماید برای مثال : هر دقیقه ، هر ساعت ، هر روز ، هر هفته و ... .

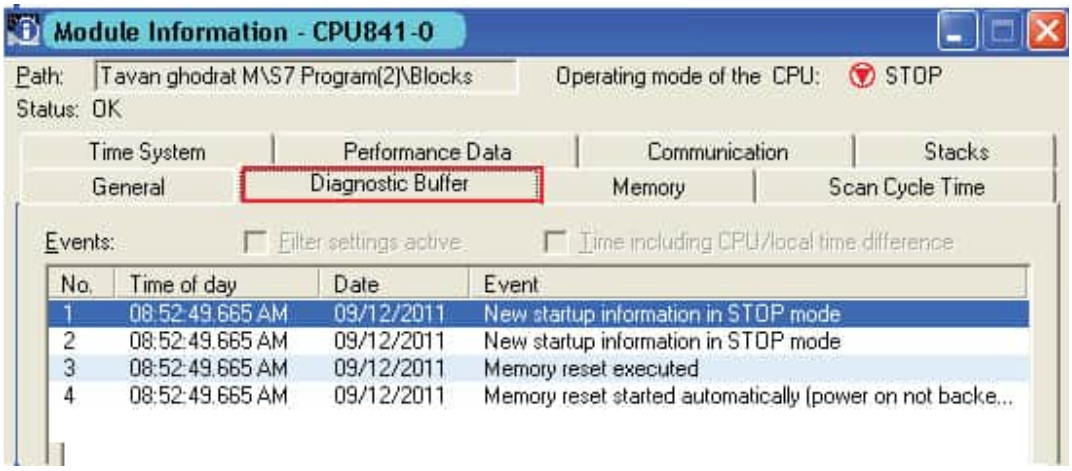
وقفه های دوره ای(Cyclic Interrupt) : این لبه مربوط به تنظیمات وقفه های دوره ای (سیکلیک) می باشد.

برای مثال هر 10 دقیقه می توان شیر تخلیه یک مخزن راتحریک نمود تا فشار از حد مجاز بیشتر نگردد (جهت امنیت بیشتر) یا هر 2 ساعت سیگنال آلارمی را فعال کند تا کاربر شیفت شب، خواب نماند.

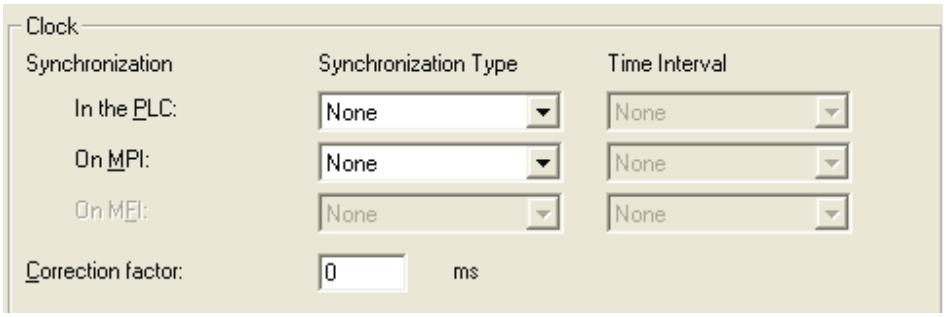
خطایابی / ساعت (Diagnostic / Clock) **:** در قسمت خطایاب می توان با فعال کردن تیک مربوطه نوع خطای رخ داده را مشاهده نمود.

ممکن است CPU در حال کار ، ناگهان Stop کند و کاربر متوجه عیب نشود در این هنگام می توان از خود CPU کمک گرفت برای این کار کافیست کلید های ترکیبی Ctrl+D را فشار دهید، در پنجره باز شده از لبه Diagnostic Buffer تمامی اتفاقاتی که در CPU رخ داده را می توان به همراه جزئیات مربوطه مشاهده نمود.

نکته : برای استفاده از این قابلیت ، باید پنجره محیط شبیه سازی باز باشد یا CPU به کامپیوتر متصل شده باشد.

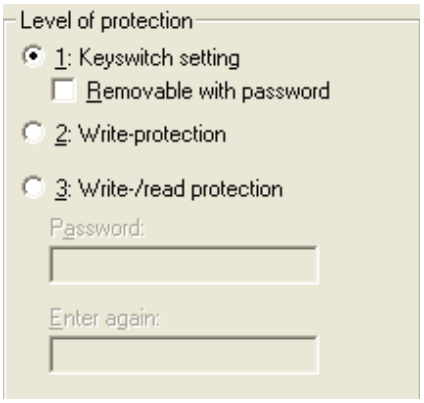


در قسمت مربوط به ساعت سیستم نیز می توان نوع هماهنگ سازی این بین PLC و کامپیوتر را مشخص نمود که در قسمت Synchronization Type مشخص می گردد. که می توان یکی از این دو دستگاه را به عنوان رئیس (Master) و دیگری را نیز به عنوان فرمان بر تعریف نمود. گزینه Time Interval نیز فاصله زمانی بین رئیس و فرمان بر را مشخص می کند که هر چه این مقدار کمتر باشد سرعت سیستم بیشتر خواهد بود.



حفاظت (protection) : در این لبه می توان تنظیمات مربوط به حفاظت و سطح دسترسی کاربران را مشخص نمود.

قسمت سمت چپ مربوط به تنظیمات دانلود و آپلود کردن برنامه از CPU می باشد.

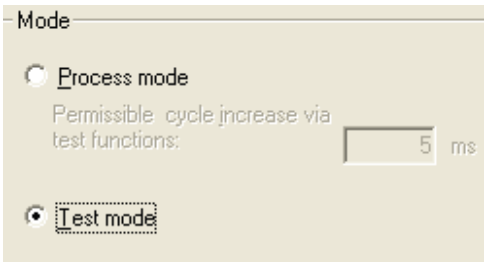


سطح 1 : دانلود و آپلود مجاز می باشد. این مورد برای CPU هایی که دارای سوئیچ می باشند کاربرد دارد که در حالت RUN_P وSTOP اجازه دانلود داده شده است ولی در حالت RUN این اجازه داده نشده است. با انتخاب گزینه (Removable With password) می توان در حالت RUN نیز توسط یک پسورد عمل دانلود را انجام داد.

سطح 2 : آپلود مجاز ولی در دانلود نیاز به پسورد است.

سطح 3 : دانلود و آپلود کردن نیازمند پسورد است.

تعیین مد کاری: زمانی که بر روی حالت Test Mode باشد در هر لحظه کاربر می تواند توسط کامپیوتر به CPU متصل شده و تغییرات را به صورت Online مشاهده نماید ولی در حالت Process Mode (مد فرآیندی) فقط به مدت زمان تعیین شده در این کادر می توان با CPU به صورت Online ارتباط داشت.



فصل هشتم

بلوک های سازماندهی (OB)

در این فصل آشنا خواهید شد با ....

۱. انواع بلوک های سازماندهی
۲. نحوه عملکرد آنها
۳. نحوه استفاده از این بلوک ها در روند برنامه نویسی

بلوک های سازماندهی Organization Blocks:

بلوک های سازمان دهی یا OB ها همان طور که از اسمشان مشخص است برای سازماندهی کار با PLC است و به نوعی یک واسط بین CPU و سیستم عامل می باشند. منظور از سازمان دهی ، مشخص کردن عملکردCPU در صورت بروز اتفاق خاص است. برای مثال اگر مشکل سخت افزاری رخ دهد می توان با فعال کردن OB مربوطه آن عمل مورد نظر کاربر تعریف گردد یا هر یک ساعت عمل مشخصی را انجام دهد. از فواید استفاده از OB ها می توان به این مورد اشاره کرد که در حالت کار عادیCPU در صورت بروز مشکل CPU خطای سیستمی داده و به حالت Stop می رود ولی در صورت وجود OB ، CPU خطای سیستمی داده ولی به حالت STOP نمی رود و با روشن شدن چراغ روی CPU وجود خطا را اعلام می دارد.

به همراه هر OB شماره ای قرار دارد که علاوه بر اینکه نشانگر نام آن است بیانگر ارجعیت اجرایی آن نیز می باشد و از آنجائیکه در هر لحظه فقط یک OB توسط CPU اجرا می گردد اجرای OB در صورت فعال شدن همزمان دو OB توسط این شماره ها مشخص می گردد.

برای مثال اگر OB35 و OB10 به طور همزمان فعال گردند OB35 اجرا خواهد شد.

****اجرای BO1 به علت داشتن کمترین ارجعیت ، در صورت فعال شدن هر OB دیگری متوقف می شود.****

مواردی را که OB برای انجام آن استفاده می شود به طور خلاصه می توان به صورت زیر ذکر کرد:

- لحظه راه اندازی CPU
- انجام کارهای دوره ای و یا مشخص شده با ساعت
- زمانی که خطا رخ دهد
- زمانی که وقفه سخت افزاری رخ دهد.

که در ادامه OB ها به همراه عملکردشان توضیح داده می شوند. دقت شود OB ها هر شماره ای را نمی توانند به خود اختصاص دهند برای مثال OB2 وجود خارجی ندارد.

معرفی OBها:

OB1: محل اجرای دوره ای برنامه اصلی می باشد و همان طور که گفته شد از کمترین درجه ارجعیت بر خوردار است.

OB10-OB17 (Time of Day Interrupts): این OB ها برای انجام کارهایی استفاده می شوند که کاربر بخواهد در ساعت مشخصی از یک روز معلوم، عملیاتی انجام شود. تعداد OB های این گروه بسته به مدل CPU است یعنی یک CPU ممکن است فقط دارای OB10 باشد.

OB20-OB 23(Time Delay Interrupt): این OB ها دارای عملکرد تأخیری می باشند یعنی وقتی ورودی مورد نظر فعال شود با یک تأخیر از قبل مشخص اجرا خواهند شد. تنظیمات در پیکربندی سخت افزار صورت می گیرد. زمان تأخیری که می توان تعریف نمود از 1ms الی 600 ms است.

اگر OB از این گروه فراخوانی شود ولی برنامه ریزی نشده باشد آنگاه OB85 فراخوانی می شود و اگر OB85 نیز برنامه ریزی نشده باشد CPU به حالت Stop می رود.

OB 30- OB 38(Cyclic Interrupts): از این بلوک ها برای انجام کارهای دوره ای استفاده می شود برای مثال هر یک دقیقه شیر تخلیه یک مخزن را باز کند تا فشار در آن افزایش نیابد. نحوه عملکرد این گروه از OB ها در زیر نشان داده شده است.

با صدور فرمان RUN برنامه ، OB1 اجرا خواهد شد و با توجه به زمان مشخص شده، اجرای OB1 متوقف و OB35 اجرا می شود و سپس ادامه OB1 اجرا خواهد شد و به همین ترتیب الی آخر.

OB40-OB47 (Hardware Interrupts): این دسته از OB ها برای انجام وقفه های سخت افزاری استفاده می شوند. منظور از وقفه های سخت افزاری ، اتفاقاتی است که توسط سخت افزار بیرونی رخ می دهد خرابی یک کارت یا قطع شدن سیم ارتباطی می تواند از این دسته باشند. باید دقت کنید که در این حالت سخت افزار نیز باید قابلیت ارسال وقفه سخت افزاری را نیز داشته باشد. اگر وقفه سخت افزاری برای CPU ارسال شود ولی هیچ OB از این دسته برنامه ریزی نشده باشد OB85 فراخوان می گردد و اگر OB85 نیز برنامه ریزی نشده باشد CPU به حالت Stop می رود.

OB55-OB57: این OB ها برای سازمان دهی وقفه هایی است که توسط شبکه پروفیباس اعلام می گردند.

OB 80(Time error): جهت سازماندهی خطاهای زمانی کاربرد دارد برای مثال زمان دوره‌ای اسکن که اگر از حد مجاز بیشتر شود.

OB81 (Power Supply Error): در زمان وجود خطا در منبع تغذیه فعال می گردد. خطا در منبع تغذیه می تواند افت ولتاژ در منبع، دشارژ شدن باتری و یا اتمام باتری پشتیبان باشد که این قابلیت در CPU های سری 400 وجود دارد.

OB 82(Diagnostic Interrupt): برای وقفه های از جنس خطا به کار می رود برای مثال اتصال کوتاه در یک کارت ورودی. البته باید در نظر داشت که کارت نیز قابلیت ارسال این وقفه ها را باید داشته باشد.

OB83 (Insert/Remove Module Interrupt): وقفه وارد کردن یا خارج کردن ماژول در CPU های سری 400 استفاده می شود. پس از روشن کردن PLC آنگاه CPU ماژول های نصب شده و به آن و آنچه که در نرم افزار وجود دارد را با هم مقایسه می نماید و اگر اختلافی مشاهده نماید این OB را فراخوانی می کند.

OB84 (CPU Hardware Fault): در صورت بروز خطای سخت افزاری در CPU این OB فعال می گردد.

OB85 (Program Sequence Error): خطای ترتیب اجرای برنامه را مدیریت می کند. برای مثال شرایط فراخوانی یک OB فراهم شده است اما آن OB وجود نداشته باشد یا CPU به هر دلیلی نتواند جداول PII و PIQ را به روز کند، آنگاه OB85 اجرا خواهد شد.

OB86 (Rack Failure): زمانیکه خطا در ریل (رک) رخ دهد این OB فراخوانی می گردد. اگر بین رک اصلی و رک توسعه یافته قطعی ایجاد گردد نیز می توان از این OB استفاده نمود.

OB87 (Communication Fault): اگر در ارتباطات MPI مشکلی ایجاد شود که این شبکه قادر به خواندن یا نوشتن نباشد این OB فراخوانی می گردد.

OB121 (programming error): زمانیکه خطای برنامه نویسی رخ دهد این OB فراخوانی می گردد برای مثال تایمری که وجود ندارد مورد استفاده قرار گیرد یا بلوک داده ای فراخوانی گردد که وجود خارجی ندارد.

OB122 (I/O Access error): در صورت عدم دسترسی CPU به ورودی و خروجی بیان شده در روند برنامه این OB اجرا می گردد. برای مثال آدرسی در روند برنامه استفاده شده باشد ولی در پیکربندی سخت افزاری آن آدرس را غیر فعال کرده باشید.

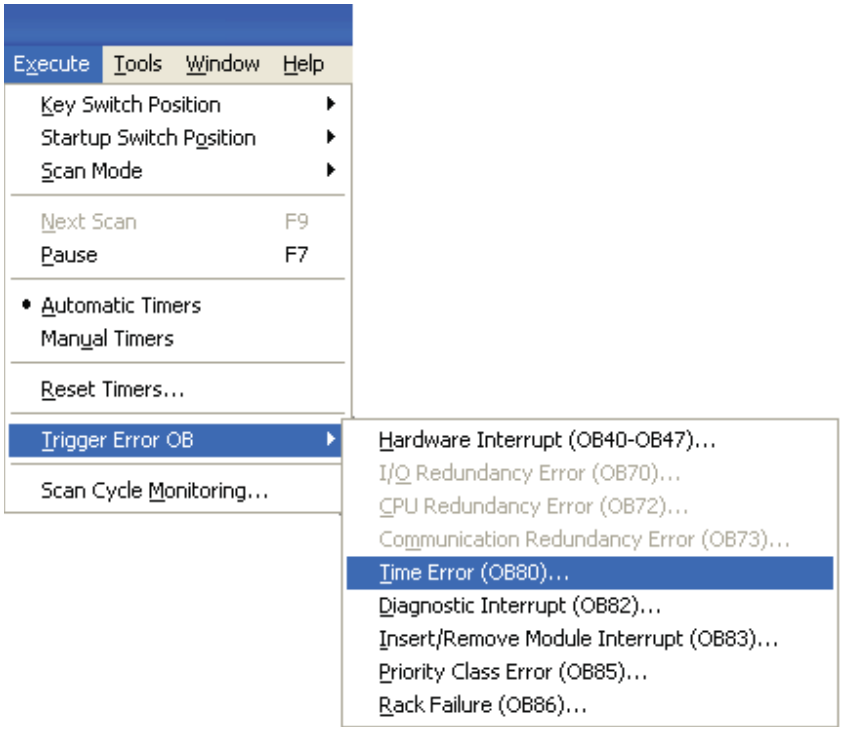
برای استفاده از OB ها باید دقت نمود در زمان دانلود برنامه اصلی (OB1) ، OB مربوطه نیز باید دانلود گردد.

در ادامه نحوه استفاده از OB ها و تأثیر آن در روند اجرای برنامه به همراه یک مثال توضیح داده شده است.

مثال: یک برنامه ساده را در نظر بگیرید که فقط دو ورودی در آن AND می شوند. این برنامه را که در OB1 نوشته اید در CPU دانلود نمائید و سپس در محیط شبیه سازی به آن خطای زمانی (Time error) وارد نمائید و وضعیت CPU را مشاهده کنید. حال به همراه OB1 بلوک OB80 را نیز وارد کرده و سپس همان خطا را اعمال نمائید و رفتار CPU را بررسی نمائید.

پاسخ: برنامه AND دو ورودی I0.0 و I0.1 را در نظر بگیرید که به آن خطای زمانی اعمال شده است. برای اعمال این گونه خطاها باید از مسیر Execute گزینه Trigger Error OB را انتخاب و سپس نوع خطای مورد نظر را کلیک کنید که در شکل زیر خطای زمانی انتخاب شده است.

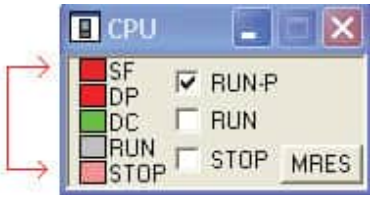
به شکل زیر دقت کنید. برای ساخت یک OB نیز کافیست در صفحه اصلی راست کلیک نموده و از گزینه Insert New Object گزینه Organization Block را انتخاب نمائید.



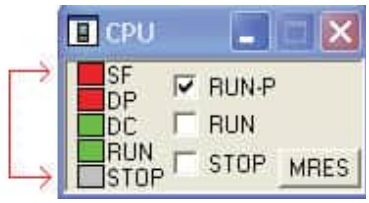
بعد از انتخاب نوع خطای زمانی پنجره ای به شکل زیر باز می شود که نوع خطای زمانی را باید انتخاب نمود که تجاوز از زمان اسکن برنامه را انتخاب کنید و سپس روی گزینه Apply کلیک نمائید تا خطا اعمال شود . در همین لحظه چراغ SF بر روی CPU روشن شده و به مد Stop می رود و روند اجرای برنامه متوقف خواهد شد.

h d i

علاوه بر روشن شدن سیگنال خطای سیستمی، CPU به حالت Stop نیز می رود.



حال اگر OB80 را نیز به همراه OB1 دانلود کنید می توان دید که با اعمال خطای زمانی ، سیگنال خطای سیستمی روشن می گردد ولی CPU همچنان به کار خود ادامه می دهد.



فـصل نـــهم

پروژه های مقدماتی

مشتمل بر:

۹-۱ کنترل یک خروجی به وسیله سه شستی s3,s2,s1

۹-۲ کنترل خروجی به وسیله ۴ شستی به صورت ترکیبی

۹-۳کنترل سه خروجی در بازه زمانی ثابت با فرمان کلید Start

۹-۴ راه اندازی موتور سه فازه رتور قفسی به صورت چپگرد و راستگرد با حفاظت کامل

۹-۵ راه اندازی موتور سه فازه رتور قفسی به صورت راستگرد و چپگرد سریع

۹-۶راه اندازی موتور سه فازه رتور قفسی به صورت ترتیبی ، یکی پس از دیگری

۹-۷ راه اندازی موتور سه فازه رتور قفسی رتور قفسی به صورت ترتیبی ، یکی بجای دیگری

۹-۸ راه اندازی موتور سه فازه رتور قفسی به صورت یکی پس از دیگری با تایمر

۹-۹ راه اندازی موتور سه فاز قفسی به صورت ستاره —مثلث دستی با حفاظت کامل

۹-۱۰ راه اندازی موتور سه فاز قفسی به صورت ستاره —مثلث و چپگرد و راستگرد دستی

۹-۱۱ راه اندازی موتور سه فازه به صورت ستاره و مثلث چپگرد و راستگرد اتوماتیک

۹-۱۲ پروسه کنترل موتور توان بالا به همراه FAN

۹-۱۳ روشن و خاموش کردن تعداد زیادی خروجی به طور همزمان با آرایشی ثابت

۹-۱۴ کنترل پر و خالی شدن مخزن

۹-۱۵ مکانیزیم دسته بندی توپ های سیاه و سفید

۹-۱۶ تعیین اولویت در فشردن کلید زنگ مسابقه

۹-۱۷چراغ چشمک زن به کمک تایمر (S-pulse)

۹-۱۸ چراغ چشمک زن به کمک تایمر (S-PEXT)

۹-۱۹ چراغ راهنمایی و رانندگی چهار راه ، با تایمر (S-pulse)

۹-۲۰ چراغ راهنمایی و رانندگی ، به کمک تنها یک تایمر (S-PEXT)

۹-۲۱ کلید پوش باتن (Push botton)

۹-۲۲ کنترل خروجی ، با ۳ بار فشردن کلید SW1در بازه ۲ ثانیه ای

۹-۲۳ چراغ روشنایی راه پله

۹-۲۴ کنترل چراغ راه پله ، با دو وضعیت موقت و دائم

۹-۲۵ شمارنده صعودی و نزولی با تحریک یک شستی

۹-۲۶ شمارنده صعودی از چند مقدار مختلف

۹-۲۷چشمکزن ۱۶ کاناله به کمک دستور Shiht (حلقه بسته)

۹-۲۸ یکی به جای دیگری ۱۶ کاناله به صورت رفت و برگشت با دستور Shift

۹-۲۹ سیستم جامع کنترل پارکینگ خودرو

مقدمه

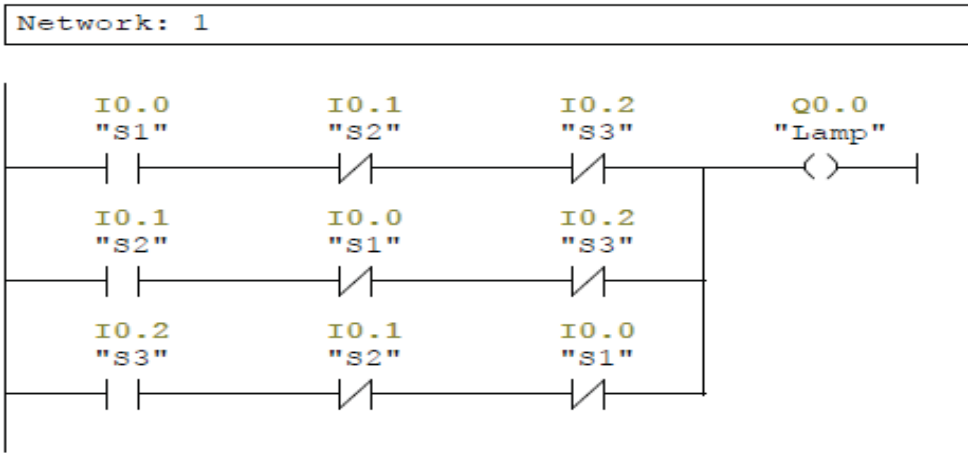
در این فصل به بررسی ۲۹ پروژه تکنیکی و کاربردی در سطح مقدماتی که شامل مباحث اصلی استفاده از Bit Logic ، دستورات مقایسه ، شمارنده ، تایمر و ... می باشد ، می پردازیم . برای حل پروژه های این فصل نیاز است ، دانشجو نسبت به مباحث مقدماتی و پایه تسلط کافی را داشته باشد ، که می توانید با مطالعه فصل های قبل جزوه آشنایی کافی را با ابزار های اصلی پیدا کنید.

۱-9عنوان پروژه : کنترل یک خروجی به وسیله سه شستی S3,S2,21

شرح پروژه : برنامه ای بنویسید که تنها با فشردن یکی از شستی های S1یا S2 و یا S3 خروجی LAMP روشن شود و در صورتی که هر ترکیب دیگری مثلاً S1 و S2 با هم وصل شوند ، خروجی LAMP روشن نشود.

S1	S2	S3	LAMP
1	0	0	1
0	1	0	1
0	0	1	1
1	1	0	0
1	0	1	0
0	1	1	0
1	1	1	0

راه حل پروژه :

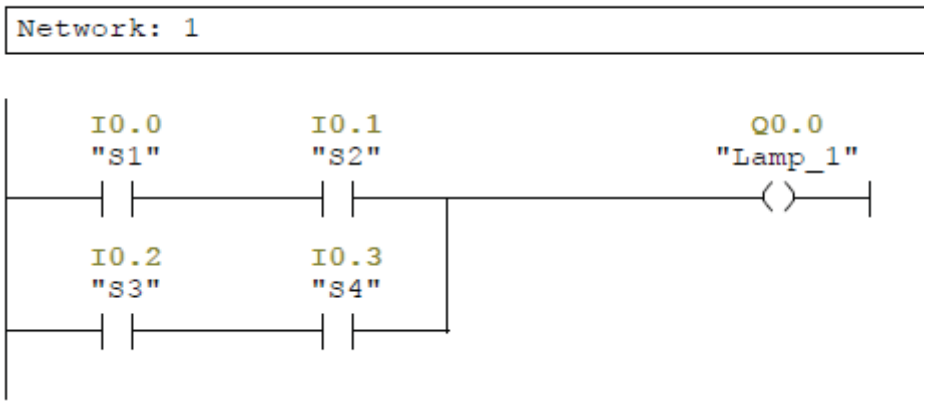


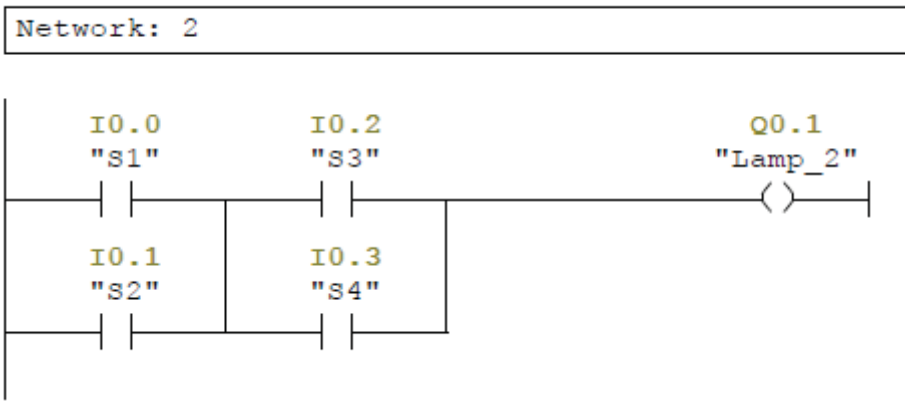
تشریح پروژه : همان طوری که در راه حل پروژه مشاهده می کنید. در Network1 با فشردن کلید S1 در صورتی که کلید های S2 ، S3 فشرده نشده باشند خروجی LAMP روشن می گردد که این منطق با قرار دادن تیغه N.C ، کلید های S2و S3 صورت گرفته است همین طور با فشردن کلید S2 در صورتی که S1 و S3 فشرده نشده باشند خروجی LAMP فعال می شودو و این اتفاق برای ورودی S3 مشروط بر این که ورودی S1 و S2 فشرده نشده باشد مجدداً رخ می دهد.

۲-9عنوان پروژه: کنترل خروجی به وسیله ۴ شستی به صورت ترکیبی

شرح پروژه : برنامه ای بنویسید که در صورتی که شستی S1 و S2 یا S 3 و S 4 با هم فشرده شوند ، خروجی LAMP1 روشن و در غیر این صورت LAMP۲ روشن شود.

راه حل پروژه:



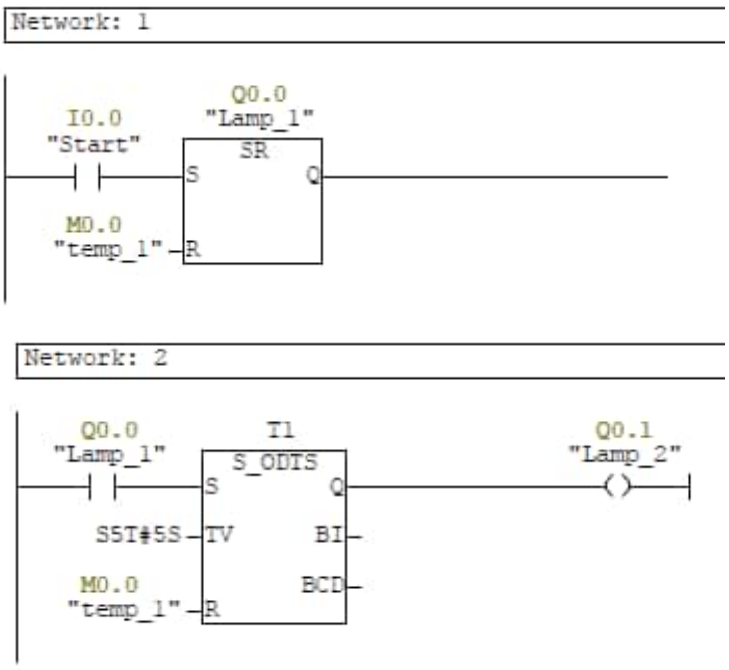


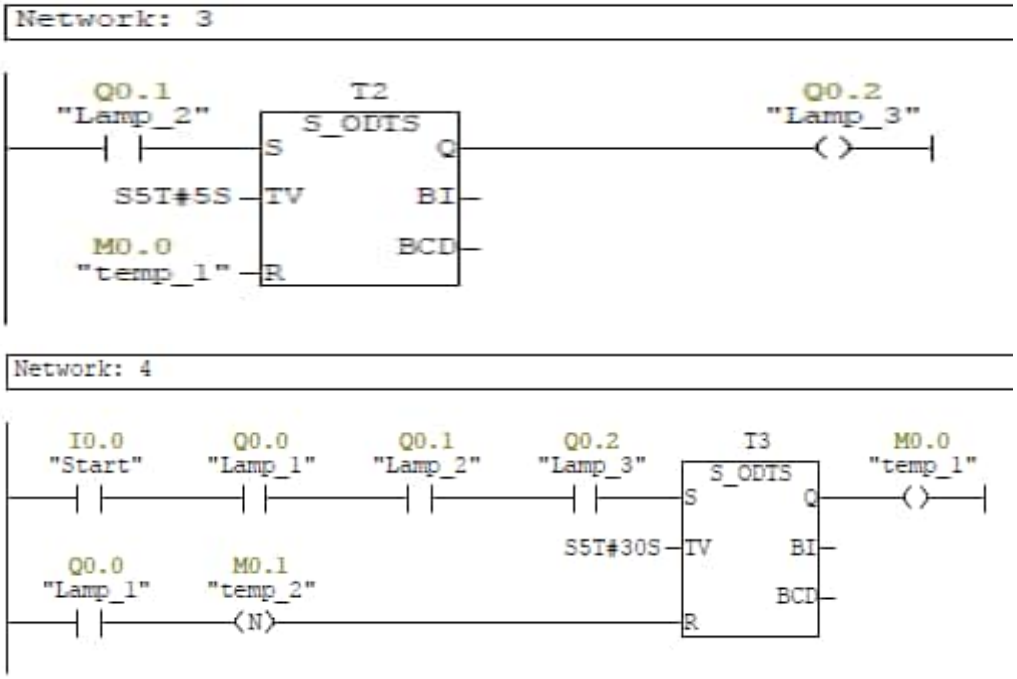
تشریح پروژه: در Network1 با فشردن کلید S1 و S2 که به صورت AND قرار دارند ، خروجی Lamp-1روشن می شود . همچنین همین خروجی در صورتی که S3 و S4 که به صورت AND می باشند ، با هم فشرده شوند ، روشن می گردد . در واقع مطابق شکل ، وضعیت AND ، S1 و S2 با S3 و S4 با هم OR می باشند که برای درک بهتر به راه حل پروژه مراجعه کنید که به صورت FBD مدار را نشان می دهد. در Network1 در صورتی که ترکیبی از S1 و S2 ، S3 وS4 به غیر از فرمت خواسته شده باشند ، خروجی Lamp-2 روشن می گردد.

۳-9به عنوان پروژه: کنترل سه خروجی در بازه زمانی ثابت با فرمان کلید Start

شرح پروژه :برنامه ای بنویسید که با فشردن شستی Start ابتدا Lamp-1 روشن شود پس از ۵ ثانیه Lamp-2 و ۵ ثانیه بعد Lamp-3 روشن گردد. سپس با فشردن مجدد کلید Start به شرطی که هر سه LAMP روشن باشند ، تایمری شروع به زمان گیری کرده و ۳۰ ثانیه بعد هر ۳ خروجی را خاموش کند و پروسه مجدداً تکرار گردد.

راه حل پروژه :





تشریح پروژه: در Network1 با فشردن شستی Start خروجی LAMP_۱ , Setمی شود.

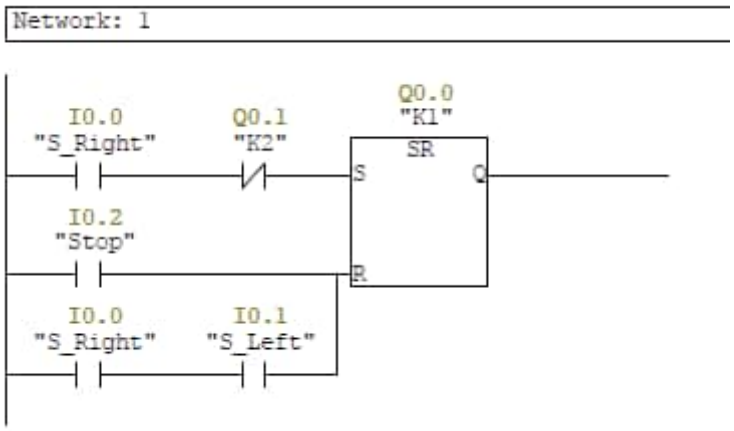
همزمان با Set شدن LAMP_۱ ، در Network2 تایمر T1 که از نوع S-ODTS می باشد شروع به زمان گیری کرده و پس از ۵ ثانیه LAMP_۲ را روشن می کند. در Network3 با روشن شدن LAMP_۲ تایمر دیگر باز از نوع S-ODTS پس از ۵ ثانیه زمان گیری ، خروجی LAMP_۳ را Set می کند. حال هر سه خروجی روشن می باشد . در Network4 تایمر T3 تنها با تحریک Start شروع به زمان گیری به مدت ۳۰ ثانیه می کند و پس از آن خروجی Temp-1را فعال می کند که سبب می شود تمام تایمر ها Reset می کند. دقت کنید که تایمر T3 تنها زمانی با تحریک Start شروع به کار می کند که قبل از آن هر سه خروجی LAMP روشن بوده باشد.

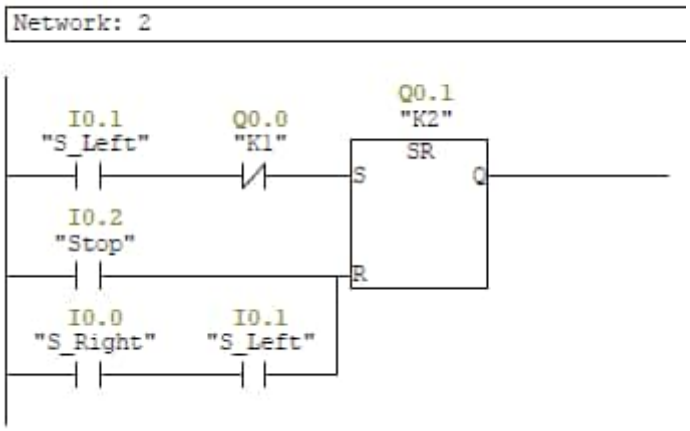
با خاموش شدن LAMP_۱ در Network4 خود تایمر T3 نیز Reset می شود و پروسه برای راه اندازی مجدد آماده می گردد.

9-4عنوان پروژه : راه اندازی موتور سه فازه رتور قفسی به صورت چپگرد و راستگرد با حفاظت کامل

شرح پروژه : با فشردن کلید SR موتور به صورت راستگرد و با فشردن شستی SL موتور به صورت چپگرد کار کند ، با این شرط که تغییر جهت گردش موتور فقط از وضعیت Stop ممکن باشد. در صورتی که موتور در حالت Stop ممکن است ، با فشردنشدن هر دو کلید SL و SR به طور همزمان هیچ کدام از کنتاکتورها جذب نگردند.

راحل پروژه :



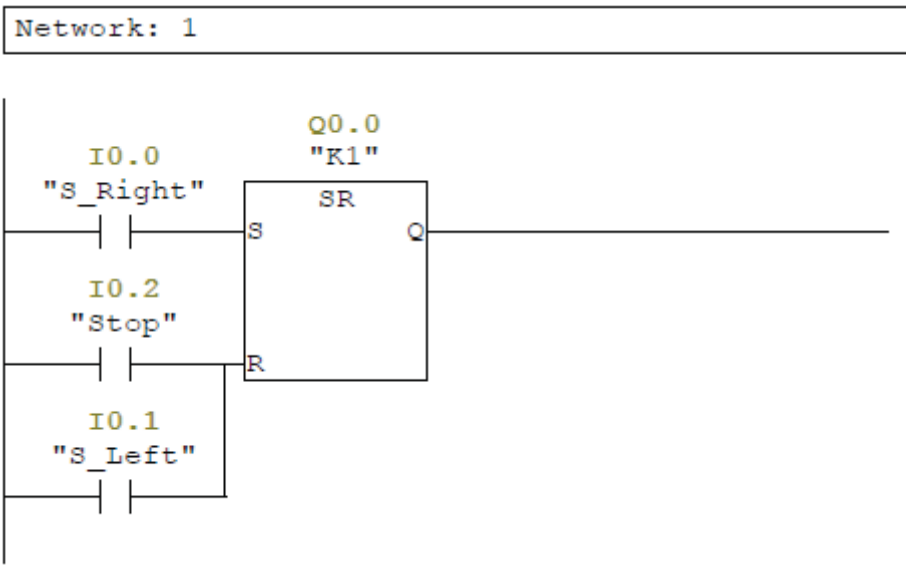


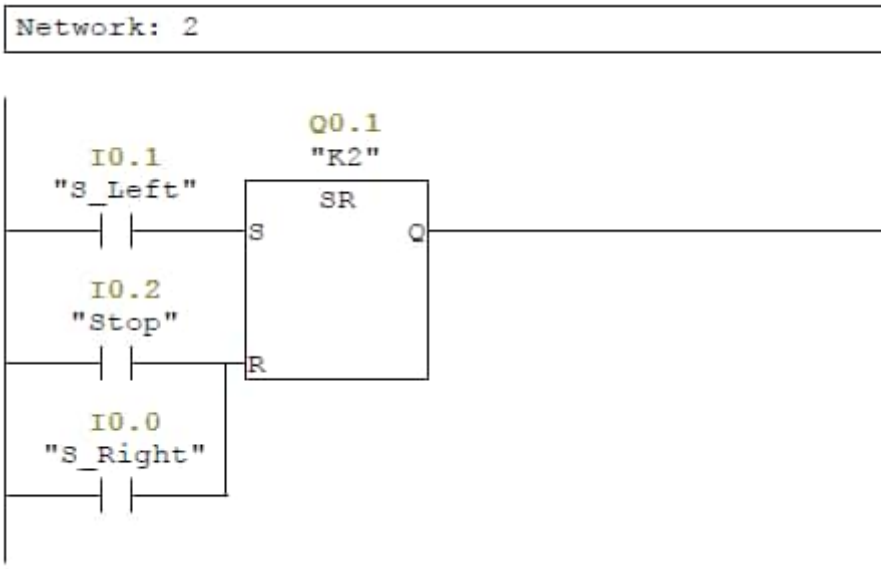
تشریح پروژه : در Network1 با فشردن کلید S-Right کنتاکتور k1 وصل می شود که مطابق با مدار قدرت ، موتور به صورت راستگر شروع به چرخش می کند . حال اگر کلید S-left فشرده شود باعث Reset شدن k1 شده و همزمان در Network2 موجب Set شدن خروجی k2 شده که سبب چرخش موتور به صورت چپگرد می گردد . همین اتفاق به صورت عکس نیز می افتد یعنی اگر موتور در مد چپگرد در حال کار است با فشردن کلید S-Right خروجی k2 قطع و k1 وصل می شود . در هر حالت نیز اگر کلید Stop فشرده شود ، هر دو خروجی Reset می گردند.

5-9 عنوان پروژه : راه اندازی موتور سه فازه رتور قفسی به صورت راستگرد و چپگرد سریع

شرح پروژه : با فشردن کلید S-Right موتور به صورت راست گرد و با فشردن کلید S-Left موتور چپ گرد کار کند . با فشردن کلید Stop موتور ایست کند و تغییر جهت گردش موتور در هر دو حالت راست گرد و چپ گرد به صورت سریع امکان پذیر باشد یعنی در حالی که موتور مثلاً در حال چرخش به سمت راست است با فشردن کلید S-Left بلافاصله تغییر جهت گردش اتفاق بیفتد.

راه حل پروژه :



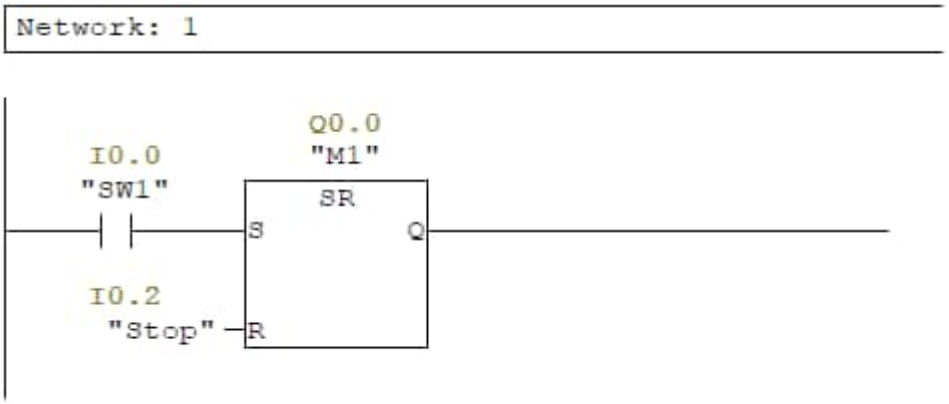


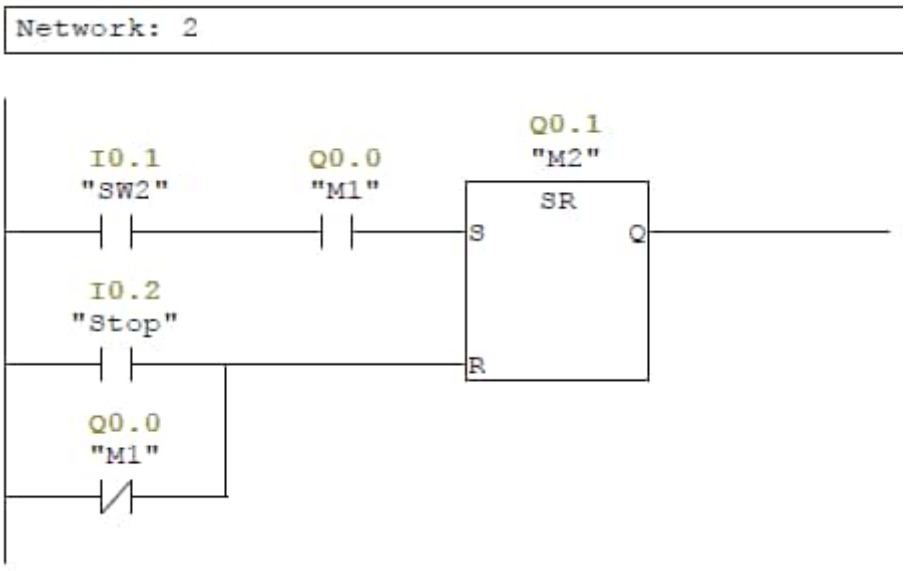
تشریح پروژه : در Network1 بافشردن کلید S-Right کنتاکتور k1 وصل که مطابق با مدار قدرت موتور به صورت راستگرد شروع به چرخش می کند. حال اگر کلید S-left فشرده شود باعث Reset شدن k1 شده و همزمان در Network2 باعث set شدن خروجی k2 شده که سبب چرخش موتور به صورت چپگرد می گردد . همین اتفاق به صورت عکس نیز می افتد یعنی اگر موتور در مد چپگرد در حال کار است با فشردن کلید S-Right خروجی k2 قطع و و k1 وصل می شود در هر حالت نیز اگر کلید Stop فشرده شود ، هر دو خروجی Reset می گردند.

9-6عنوان پروژه : راه اندازی موتور سه فازه رتور قفسی به صورت ترتیبی یکی پس از دیگری

شرح پروژه : با فشردن شستی Sw1 موتور M1 روشن شود. با فشردن شستی Sw2 به شرطی که موتور M1 روشن بود موتور M2 نیز روشن شود همچنین در هر لحظه که شستی Stop فشرده شد هر دو موتور خاموش شوند.

راه حل پروژه :



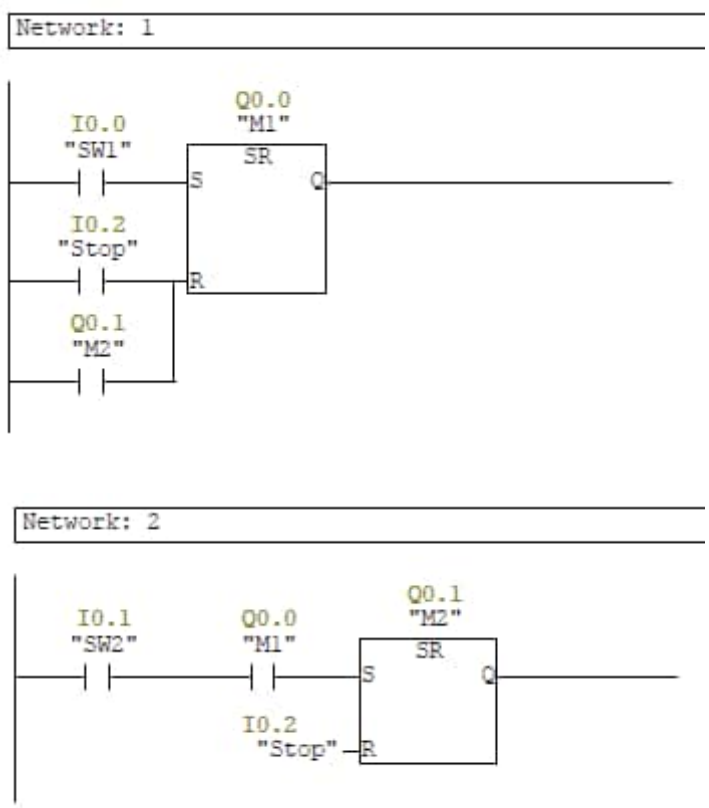


تشریح پروژه : در Network1 با فشردن شستی Sw1 خروجی M1 وصل می شود و تا زمانی که کلید Stop فشرده شود ، روشن می ماند . در Network2 به شرطی که تیغه M1 وصل باشد با فشردن شستی Sw2 خروجی M2 وصل می شود . حال اگر شستی Stop فشرده شود و یاخروجی M1 به هر دلیلی قطع شود باعث Reset شدن M2 می گردد.

7_ 9 عنوان پروژه : راه اندازی موتور سه فازه رتور قفسی به صورت ترتیبی یکی بجای دیگری

شرح پروژه : با فشردن شستی Sw1 موتور M1 شروع به کار کند. با فشردن کلید Sw2 در صورتی که موتور M1 روشن باشد. ابتدا M1 خاموش و سپس M2 روشن شود. با فشردن Stop خروجی Reset شود و در هر حالتی موتورها را خاموش کند .

راه حل پروژه :

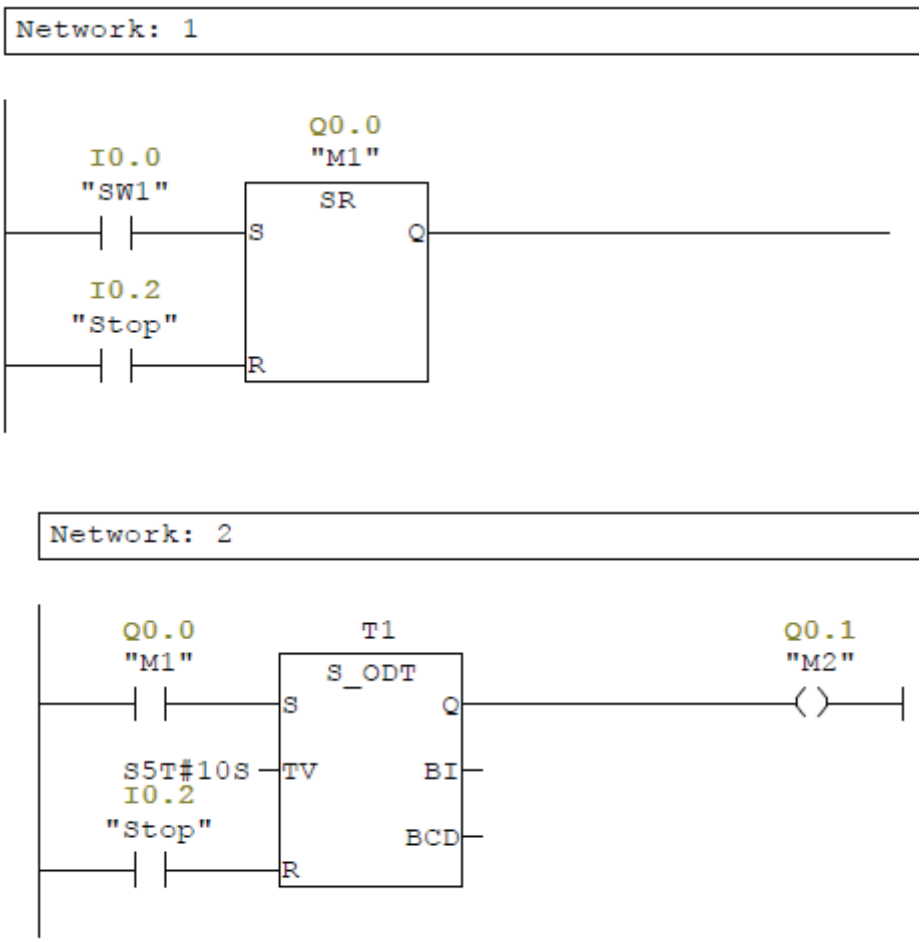


تشریح پروژه : با فشردن شستی Sw1 خروجی M1 وصل می شود و تا زمانی که کلید Stop فشرده شود و یا M2 وصل می شود که همزمان در Network2 باعث Reset شدن M1 می شود موتور M2 زمانی که کلید Stop فشرده شود ، روشن می ماند.

8-9عنوان پروژه : راه اندازی موتور سه فازه رتور قفسی به صورت یکی پس از دیگری با تایمر

شرح پروژه : با فشردن شستی Sw1 موتور M1 روشن شده و پس از ۱۰ ثانیه موتور M2 نیز وصل شود . در صورتی که شستی Stop فشرده شود هر دو خروجی خاموش گردند .

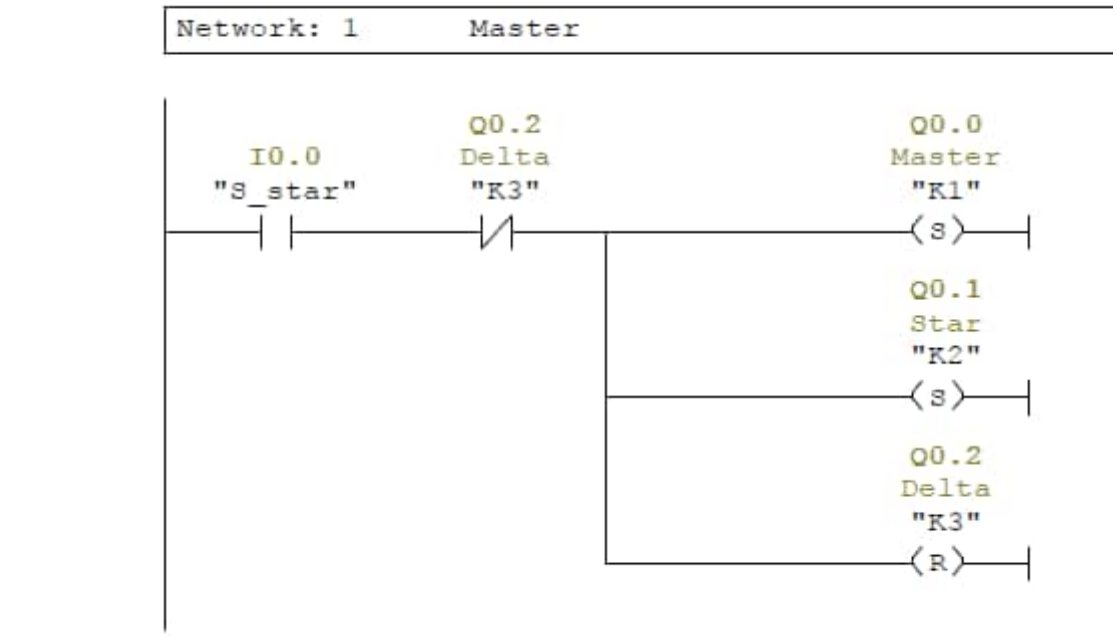
راه حل پروژه :

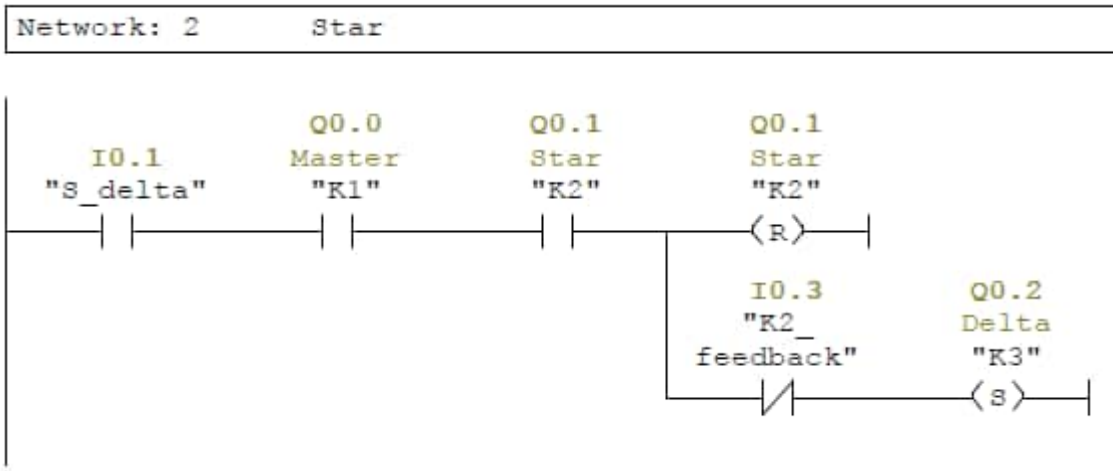


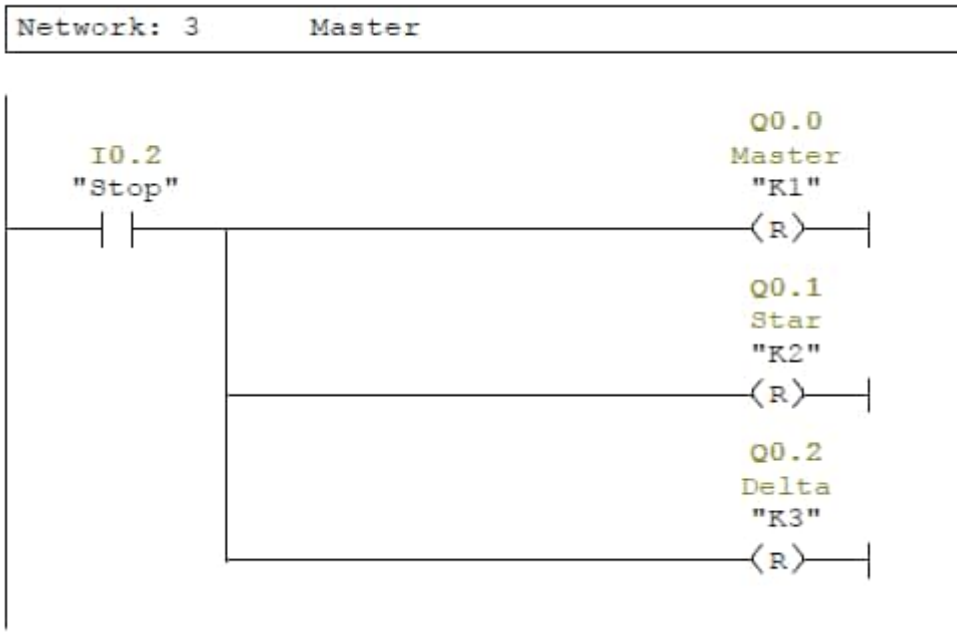
9-9 عنوان پروژه : راه اندازی موتور سه فازه رتور قفسی به صورت - مثلث دستی با حفاظت کامل

شرح پروژه : با فشرده شستی S-star موتور به صورت ستاره شروع به کار کند و رها شدن شستی باعث قطع خروجی نشود.بافشردن شستی S_Delta که می تواندکلیدگریزازمرکزباشد مشروط به این که موتور در حالت ستاره بوده باشد به حال مثلث برود . موتور در حالت مثلث به هیچ وجه با فشردن مجدد کلید S-star به مد ستاره بر نگردد . با فشردن Stop موتور متوقف شود.

راه حل پروژه :


Network: 1 Master
I0.0
"S_star"
Q0.2
Delta
"K3"
Q0.0
Master
"K1"
S
Q0.1
Star
"K2"
S
Q0.2
Delta
"K3"
R


Network: 2 Star
I0.1
"S_delta"
Q0.0
Master
"K1"
Q0.1
Star
"K2"
Q0.1
Star
"K2"
R
I0.3
"K2_
feedback"
Q0.2
Delta
"K3"
S



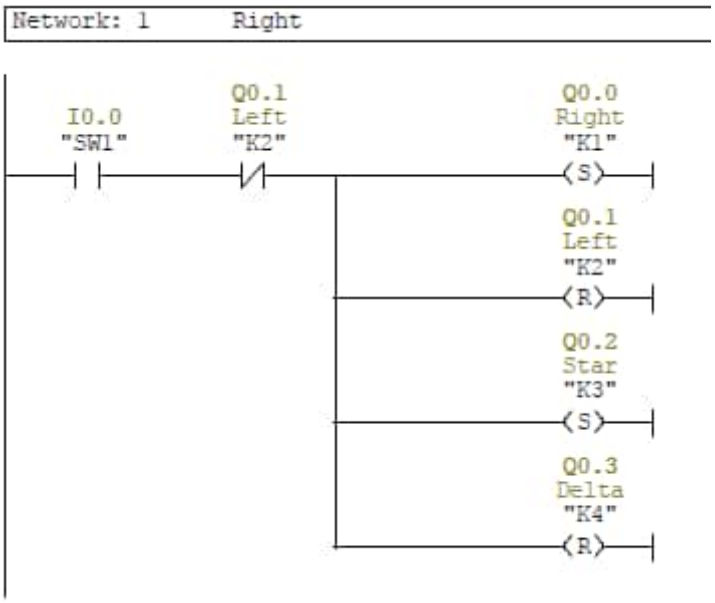
تشریح پروژه : در Network1 با فشردن شستی S-star مشروط بر این که قبلاً K3یا کنتاکتور مثلث نباشد ، k1و k3 وصل می شود که موجب می شود موتور در حالت ستاره راه اندازی ش ود . در Network2 با فشردن یا فعال شدن کلید S-Deltaمشروط به اینکه k1و k2 وصل باشد یعنی موتور در حالت ستاره بوده باشد ابتدا k2قطع می شود پس تیغه N.C کنتاکتور K2 جهت حفاظت از اینکه اطمینان حاصل شود که K2 قطع شده است به k3فرمان می دهد و آن را وصل می کند در نتیجه موتور به صورت مثلث راه اندازی می شود . علت استفاده از تیغه k2-Feadback آن است که اگر به هر دلیلی k2 قطع نشود مانند چسبیدن تیغه ها ، عمل نکردن فنر ، .... و k3 وصل شود ، مطابق نقشه قدرت سه فاز اتصال کوتاه می شوند ! در Network2 با فشردن شستی Stopهر سه خروجی k1 ، k2 و k3ریست می شود در نتیجه موتور خاموش می گردد.

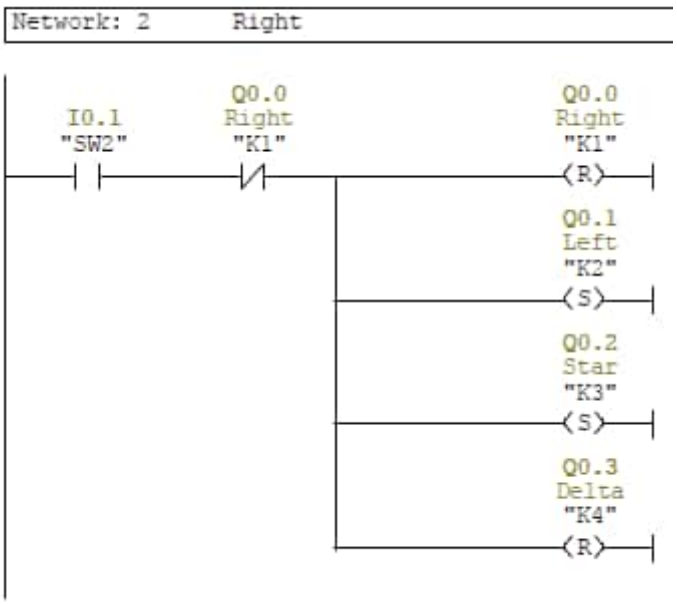
9-10 عنوان پروژه : راه اندازی موتور سه فازه رتور قفسی به صورت ستاره –مثلث و چپگرد و راستگرد دستی

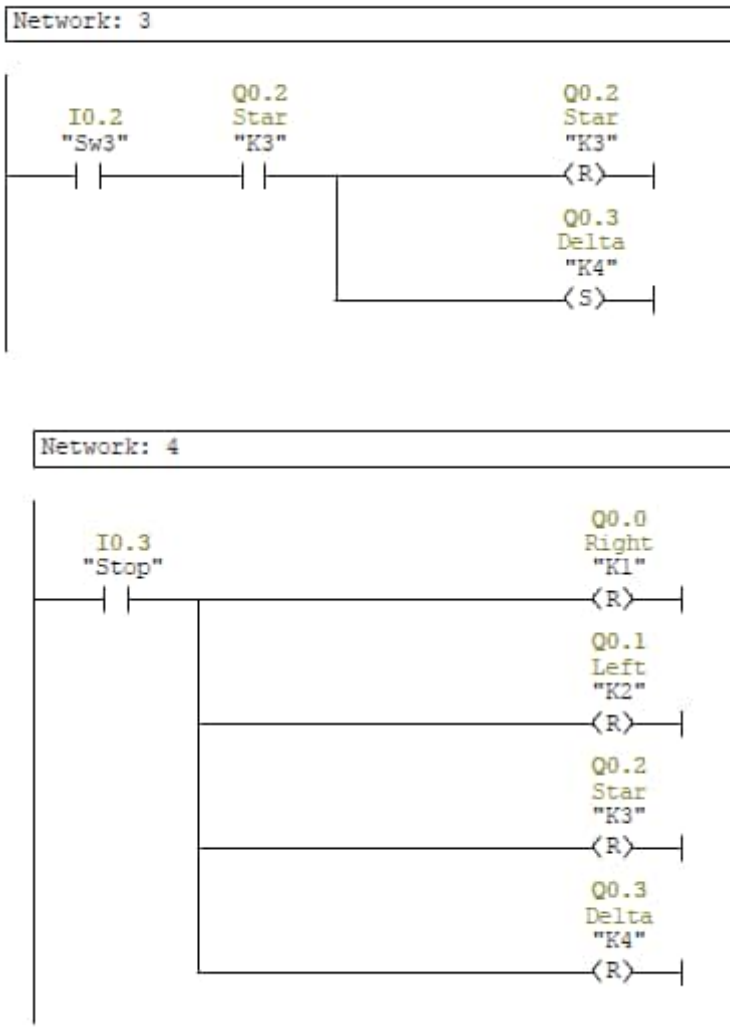
تشریح پروژه : برنامه ای بنویسید که با فشردن شستی Sw1 موتور به صورت ستاره و راستگرد شروع به کار کند . اگر کلید Sw3 فشرده شود در حالی که موتور به صورت ستاره کار می کند ، موتور از حالت ستاره به حالت مثلث برود . با فشردن شستی Sw2 موتور به صورت ستاره و چپگرد شروع شروع به کار کند . در حالی که موتور به صورت ستاره کار می کند ، اگر شستی Sw3 فشرده شود موتور از حالت ستاره به حالت مثلث برود . با فشردن شستی Stopموتور در هر حالتی که باشد ، متوقف می شود . نکته ای که باید به آن توجه کنید آن است که تغیر جهت گردش موتور تنها از حالت توقف

امکان پذیر می باشد یعنی اگر موتور در حال حرکت به صورت راستگرد است ، اگر کلید Sw2 فشرده شود و موتور تغییر جهت ندهد ، تا زمانی که ابتدا Stop شود و سپس عمل کند.

راه حل پروژه :







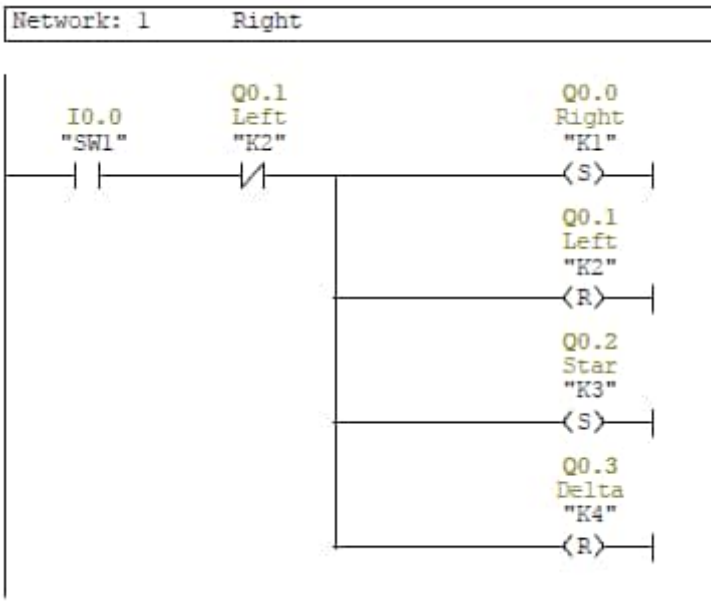
تشریح پروژه : با فشردن شستی Sw1به شرطی که موتور در حالت چپگرد نباشد به صورت ستاره و راس تگرد شروع به کار می کند. در این حالت اگر کلید Sw2 فشرده شود ، اتفاقی نمی افتد زیرا تیغه N.C کنتاکتور K1 که مربوط به مد راستگرد می باشد سر راه Sw2 است و ابتدا باید کلید Stopفشرده شود و موتور متوقف شود سپس Sw2 برای فرمان حالت چپگرد فعال شود در این حالت به صورت عکس برای تغییر جهت از چپگرد به راستگرد نیز صادق است . در Network3 با فشردن کلید Sw2مشروط به این که موتور مد ستاره بوده باشد به مد مثلث می رود ، هر زمان که کلید Stop نیز فشرده شود موتور متوقف می شود .

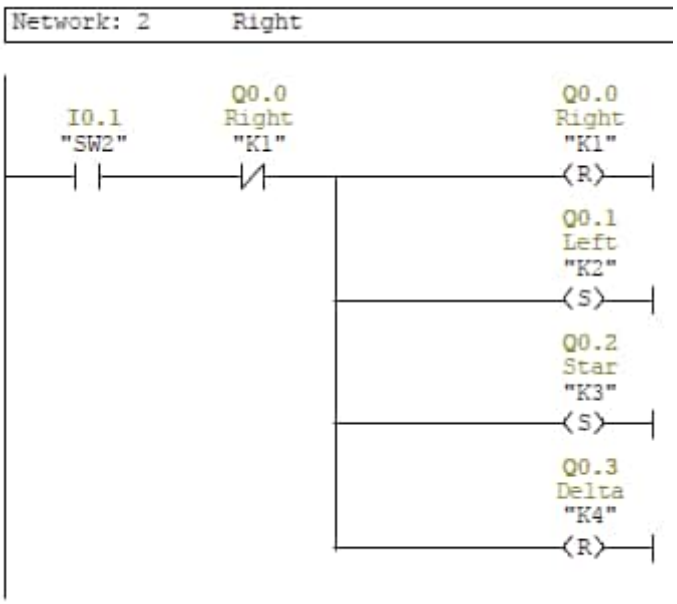
۹-۱۱ عنوان پروژه : راه اندازی موتور سه فازه رتور قفسی به صورت ستاره و مثلث ، چپگرد و راستگرد اتوماتیک

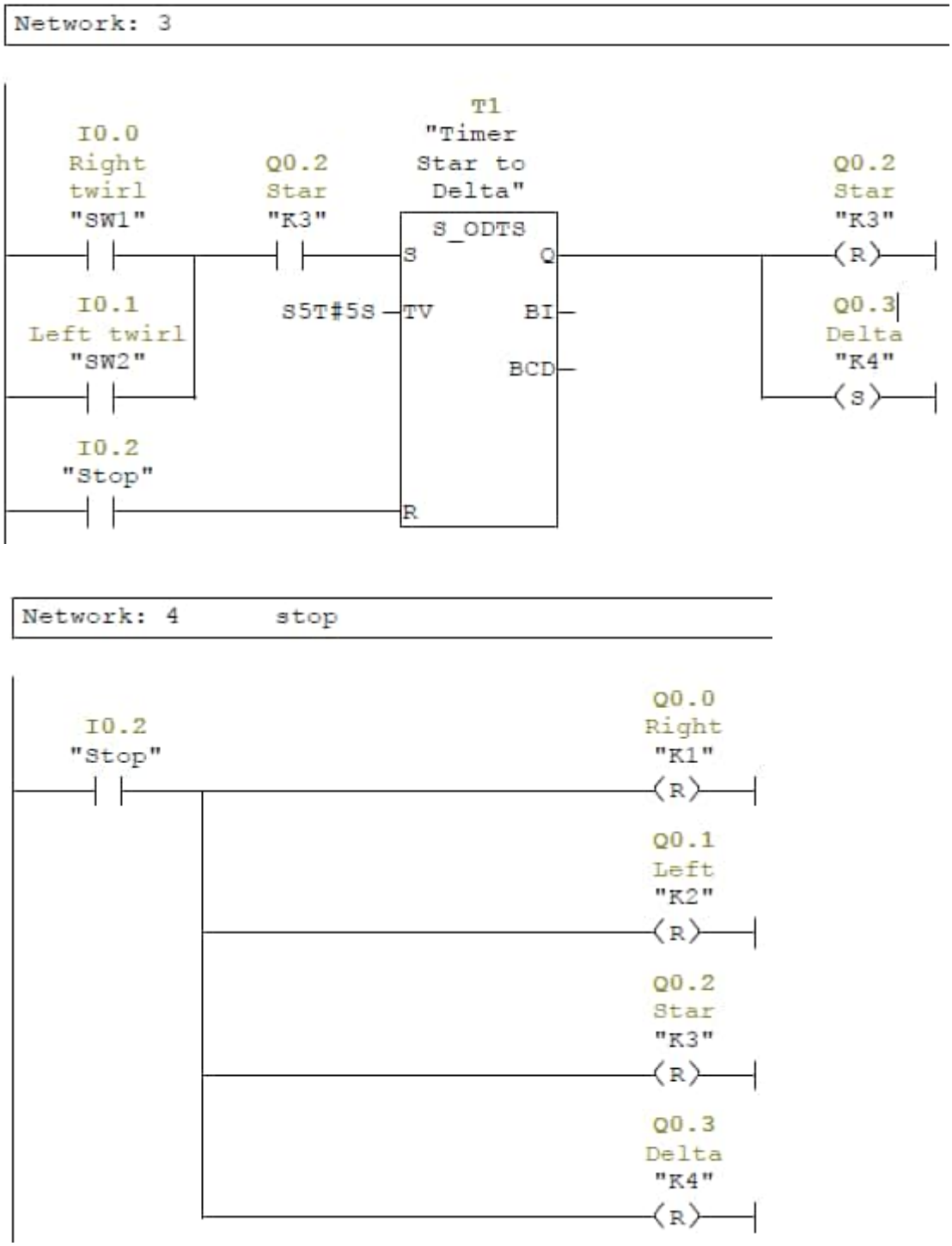
شرح پروژه : با فشردن کلید Sw1 موتور به صورت راستگرد و ستاره راه اندازی شود و پس از ۵ ثانیه از حالت ستاره خارج شده و به صورت مثلث تا زمان فشرده شد ن شستی Stopبه کار خود ادامه می دهد . با فشردن Sw2 موتور به صورت

چپگرد و ستاره شروع به کار کرده و پس از ۵ ثانیه به حالت مثلث برود . تغیر حالت گردش موتور تنها در حالت Stop امکان پذیر باشد.

راه حل پروژه :





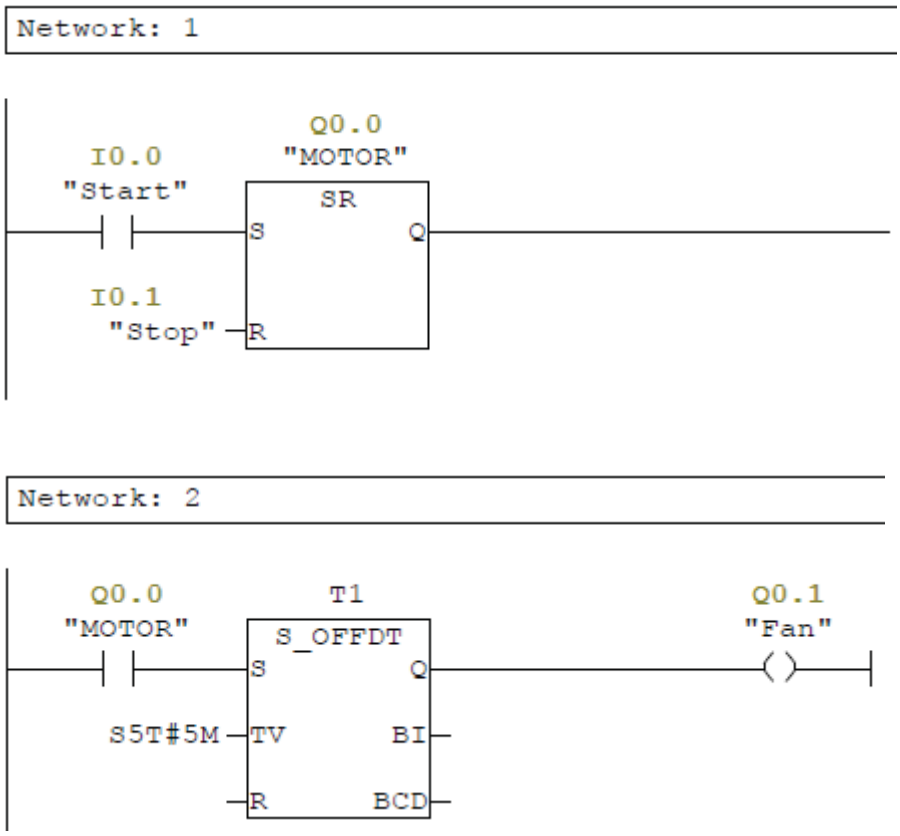


تشریح پروژه : Network1مانندپروژه قبلی طراحی شده است .Network3 با تحریک هریک ازورودی های Sw1یاSw2 به شرطی که موتوردروضعیت ستاره راه اندازی شده باشد ،تایمری ازنوع تاخیر دروصل S-ODTSشروع به زمان گیری کرده وپس از۵ثانیه موتوررا ازحالت ستاره به مثلث می برد .تغییر جهت گردش موتورهم تنها زمانی مقدوراست که ابتدا مطابق ۴ Network مدار Stopشود.

9-12 عنوان پروژه :پروسه کنترل موتور توان بالا به همراه FAN

شرح پروژه:در پروسه ای موتوری با توان کاری بالا که حرارت زیادی ایجاد می کند . می بایست در محیط بسته ای کار کند. برای آنکه بر اثر افزایش حرارت موتور ، آسیبی به آن نرسد به کمک موتور FAN دیگری ،موتور اصلی را خنک نگه داریم .با فشردن کلید start هردو موتور روشن شوند ولی اگر کلید Stop فشرده شود ،موتور اصلی خاموش شود ولی موتورFAN ، پنج دقیقه پس از خاموش شدن موتور اصلی خاموش شود تا موتور را به اندازه کافی خنک کرده باشد.

راه حل پروژه :

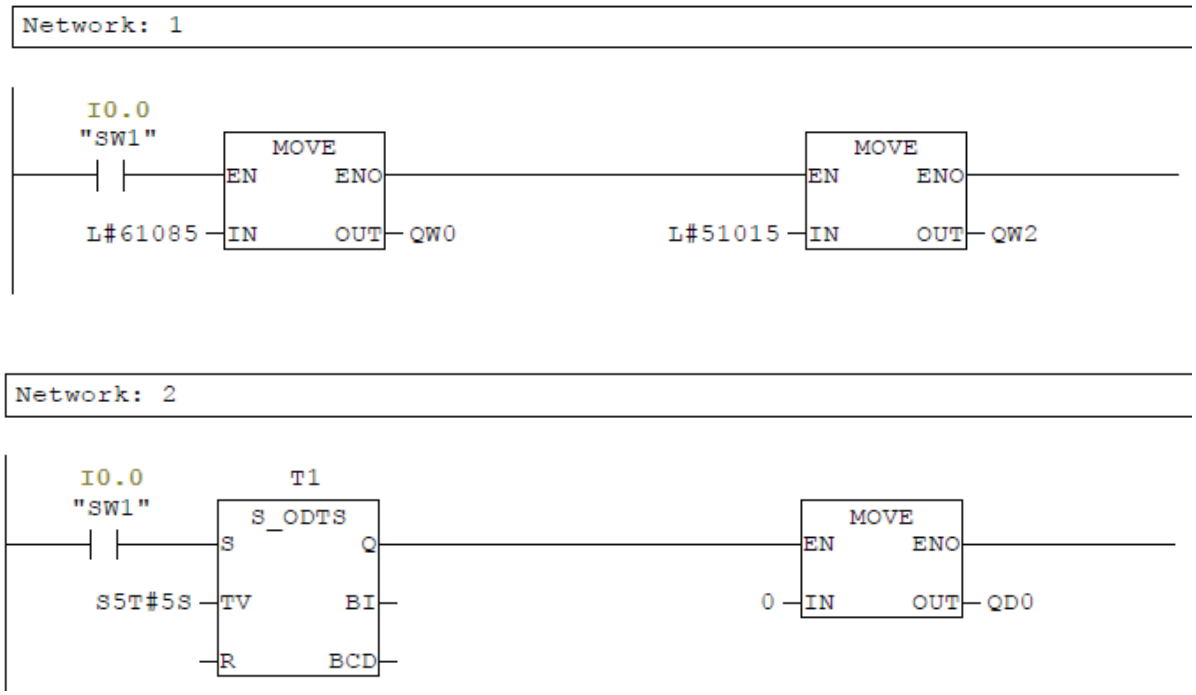


تشریح پروژه:در Network1 با فشردن کلید start خروجی Motor به صورت دائم روشن می گردد که متناظر با آن در Network ۲ چون از تایمر S-OFFDT یعنیoff Delay استفاده شده است خروجی FAN نیز تا زمانی که Motor روشن است روشن می ماند بدون آنکه تایمر شروع به زمان گیری بکند . حال اگر دکمه Reset فشرده شود Motor خاموش میشود تایمر حال شروع به زمان گیری کرد و پس از ۵دقیقه FAN را خاموش می کند.

9-13 عنوان پروژه : روشن و خاموش کردن تعداد زیادی خروجی به طور همزمان با آرایشی ثابت

تشریح پروژه:فرض کنید می خواهیم تعداد 32bitخروجی را باآرایش یکسان روشن و خاموش کنیم .برنامه ای بنویسید که با فشردن کلید Sw1 خروجی Qw0 به فرم ۱۱۱۰۱۱۱۱۰۱۰۰۱۱۱۰۱ و Qw1 بافرم ر: ۱۱۰۰۰۱۱۱۰۱۰۰۰۱۱۱ روشن شده و پس از ۵ ثانیه همه خروجی ها خاموش شوند .

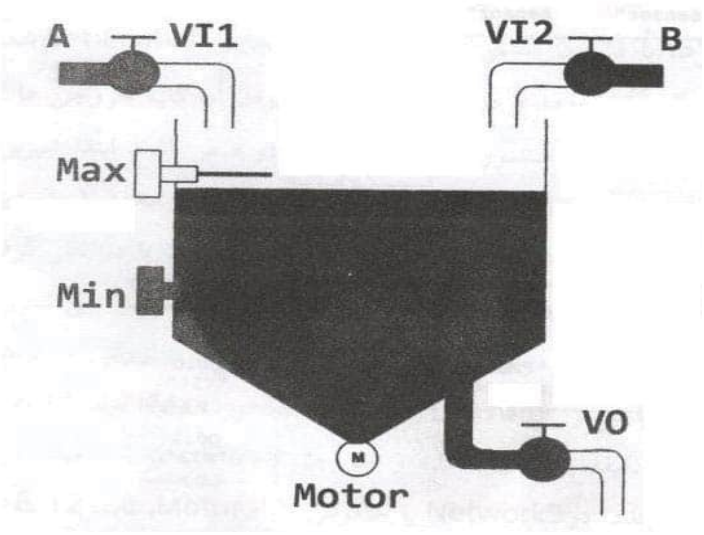
راه حل پروژه :



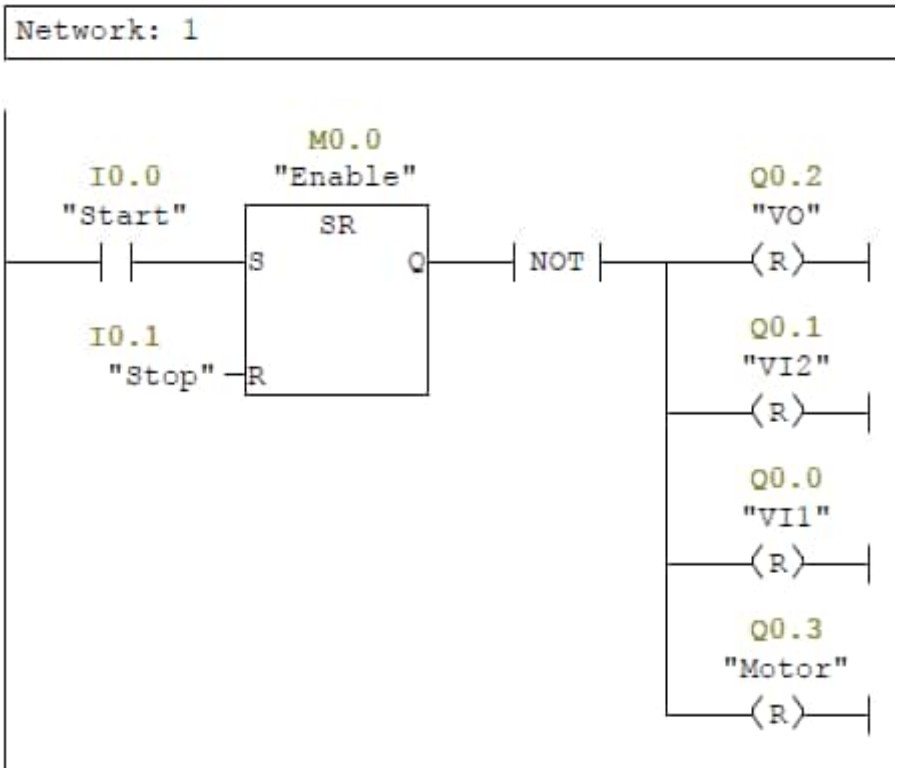
14-9عنوان پروژه: کنترل پر و خالی شدن مخزن

شرح پروژه : با فشردن شستی Start سطح مخزن به کمک سنسور های Min و Max چک شود . اگر مخزن خالی باشد یعنی هر دو سنسورخروجی صفر منطقی را گزارش دهند ابتدا خروجی مخزن یعنی شیر برقی VO بسته شود ، سپس شیر برقی ورودی VI1 باز شود تا زمانی که مواد A به سنسور Min برسد در این لحظه شیر برقی VI۱ بسته شود وشیربرقی VI2 بازشودتامواد Bواردمخزن شودوسطح آن رابه Maxبرساند.دراین لحظه شیربرقی VO2نیزبسته شودوموتور همزن Motor به مدت ۵ دقیقه مواد A و B را با هم مخلوط کند . پس از خاموش شدن موتور ، دستگاه ابتدا ۱۰ ثانیه صبر کند تا تلاتم مواد خاتمه یابد ، سپس شیر برقی خروجی VO را باز کند تا مواد تخلیه شود . بلتخلیه کامل مواد پروسه مجدداً تکرار شود .

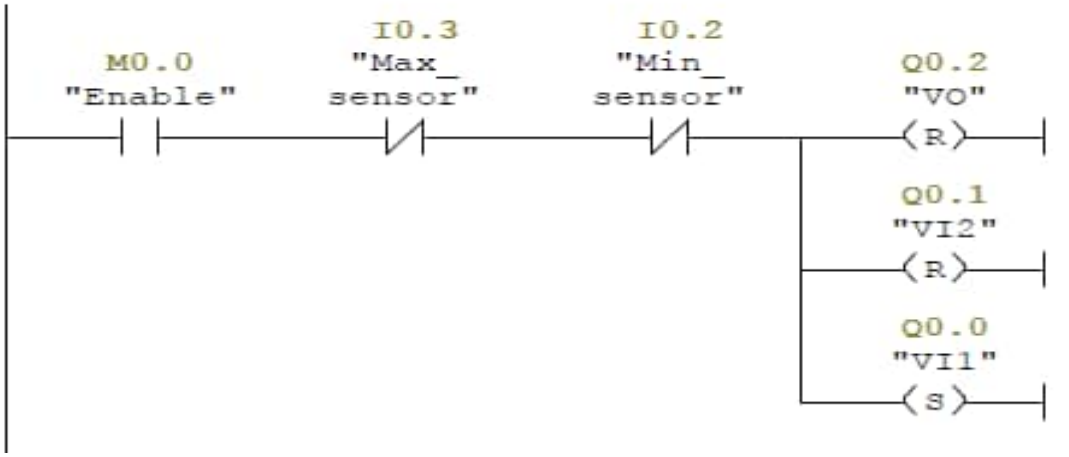
دقت شود که در صورت فعال کردن خروجی شیرهای برقی آنها باز شد و داشتن منطق صفر به معنی بسته بودن شیر برقی است.



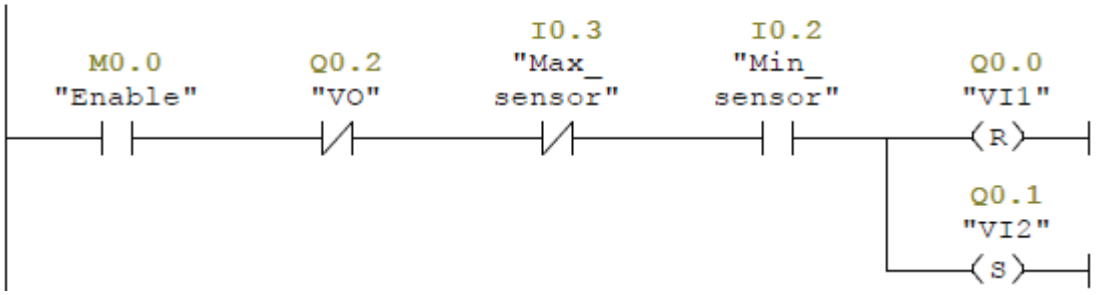
راه حل پروژه:



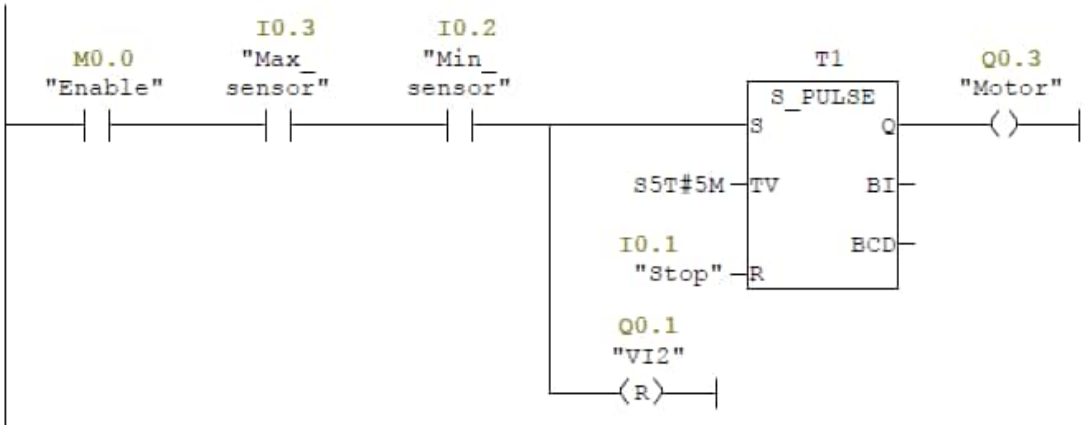
Network: 2

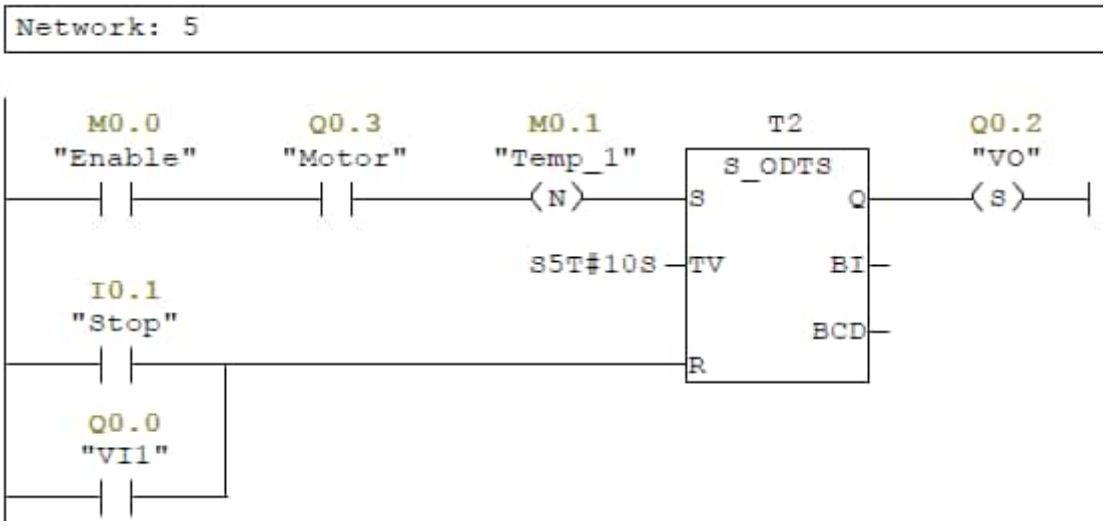


Network: 3



Network: 4



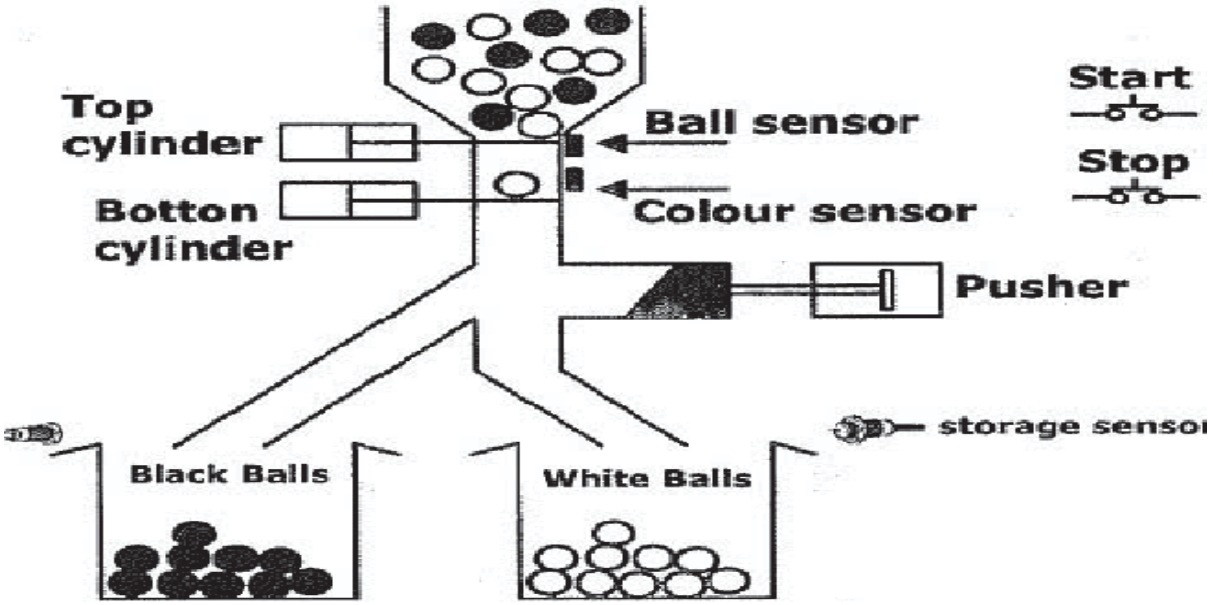


تشریح پروژه : با فشردن شستی Start خروجی مجازی Set,Enable شده و مجوز کار ، Network های دیگر را می دهد و در صورت غیر فعال بودن آن کلیه خروجی ها Reset می شوند. در Network ۲ در صورتی که سنسور Min و Max هر دو صفر باشند ابتدا شیر برقی خروجی Vo بسته شده و پس شیر ورودی VI1 وصل می شود تا مواد A وارد مخزن شود. در Network3 در صورتی که سنسور Min فعال شود و Max همچنان غیر فعال باشد با درنظر گرفتن حفاظت های غیر فعال بودن شیر برقی Vo و Enable ، ابتدا شیر ورودی VI1 بسته شده و پس شیر برقی VI2 باز شود در Network 4 با اضافه شدن مواد B به مخزن سطح آن بالا آمده تا زمانی که ب ه سنسور Max برسد ، در این حالت چون هم سنسور Min فعال است و هم سنسور Max ، تایمر T1 که از نوع S-puls می باشد ، با مقدار ۵ دقیقه بار گذاری شده و خروجی Motor همزن را فعال می کند و همزمان شیر برقی VI2 را می بندد. در Network 5 با خاموش شدن Motor ، تایمر T2 که از نوع S-ODTS تاخیر در وصل می باشد به مدت ۱۰ ثانیه تاخیر ایجاد می کند تا تلایم مواد ایست کند سپس شیر برقی خروجی VO را باز می کند تا مواد خارج شود. با خالی شدن مخزن پروسه مجدداً تکرار می گردد.

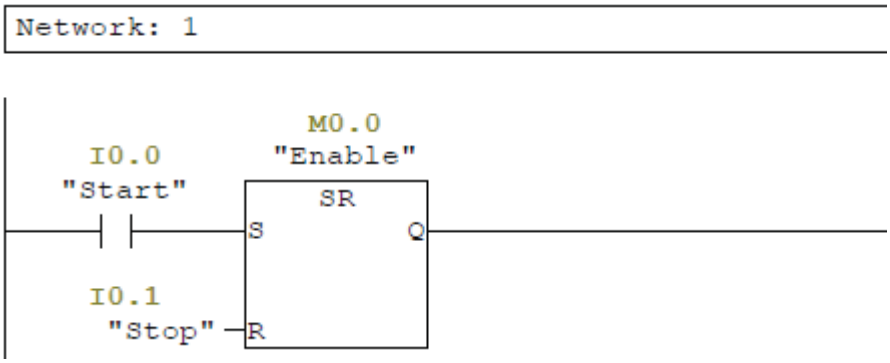
9-15 عنوان پروژه : مکانیزم دسته بندی توپ های سیاه و سفید

شرح پروژه : در این مثال ظرف پر از توپ های سیاه و سفید قرار دارد که طی یک فرآیند باید رنگ آنها تشخیص داده شده و در ظرف مخصوص خود قرار بگیرند . ابتدا موجود بودن توپ در مخزن بررسی می شود ، اگر توپ سفیدبود سیلندر پایین درب را باز می کند و توپ مستقیم وارد مخزن توپ سفید می شود. اگر توپ سیاه بود ، ابتدا سیلندر Pusher به سمت جلو حرکت کرده و سپس سیلندر پایینی درب را باز می کند تا توپ به کمک انحراف Pusher وارد مخزن سیاه شود .این فرآیند تا خالی شدن مخزن ادامه پیدا می کند . پس از خالی شدن مخزن لامپ Empty با فرکانس 1Hz روشن و خاموش شود تا زمانی که مخزن مجدداً پر شود . هر یک از سیلندر هادارای سنسور می باشند که تشخیص می دهند سیلندر تا آخر

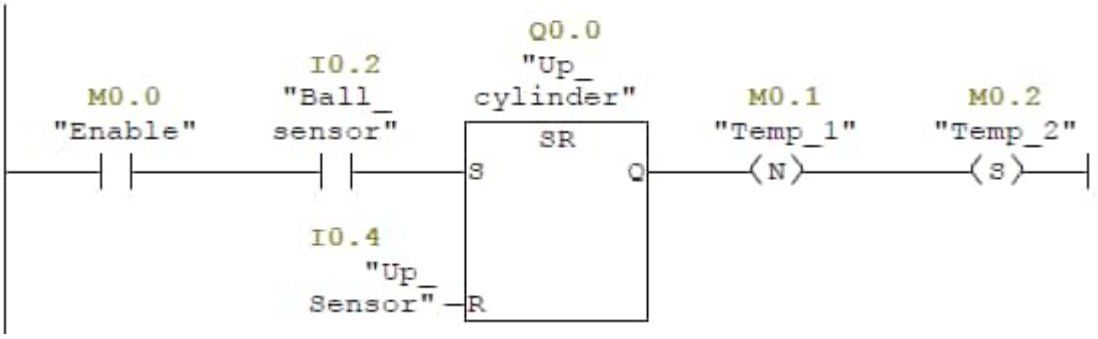
کورس خودحرکت کرده است . همچنین پروژه با کلید Start شروع به کار می کند و با زدن کلید Stop در هر لحظه پروژه متوقف می شود . همان طوری که در شکل مشاهده می کنید سنسور نوری در ابتدای دو ظرف قرار دارد که نشان می دهد توپ وارد ظرف شده است.



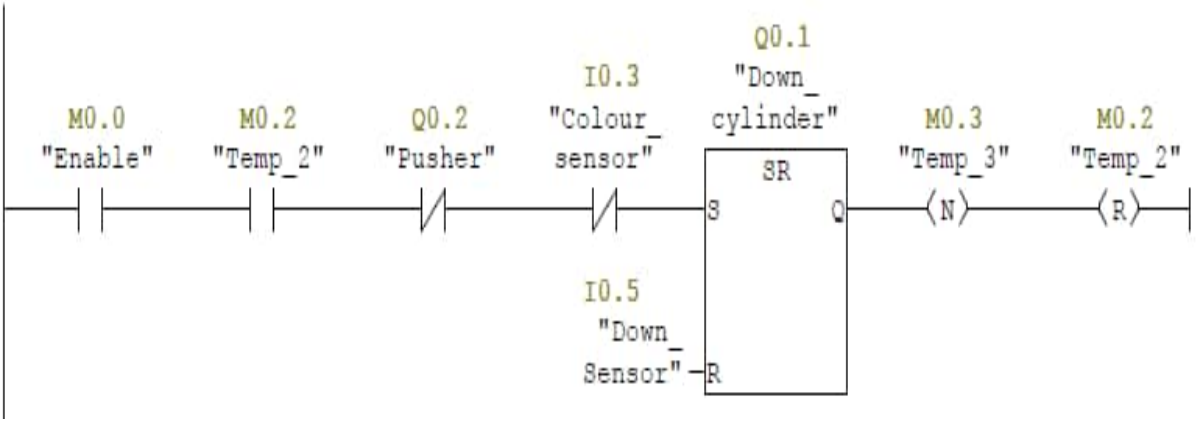
راه حل پروژه :

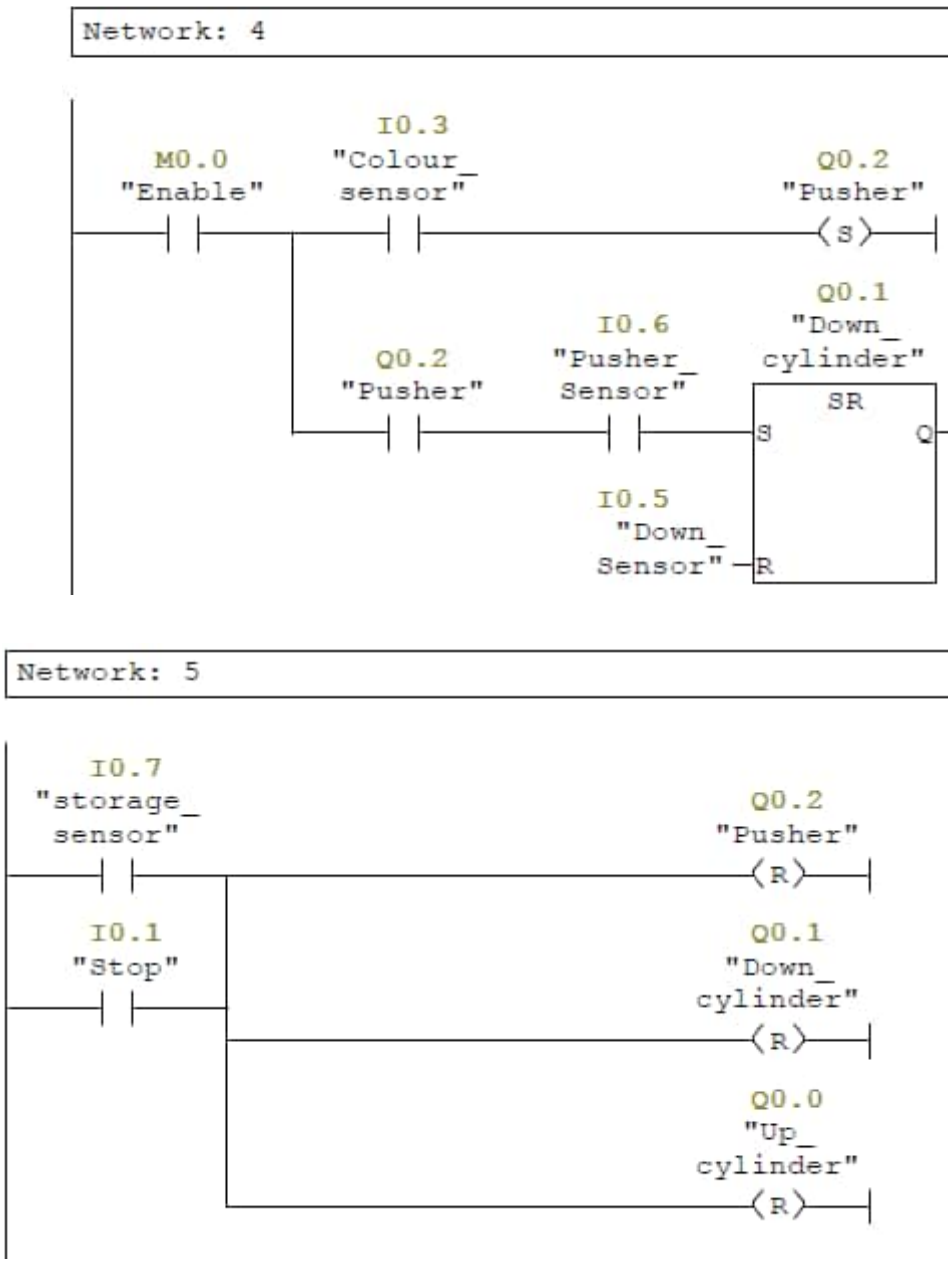


Network: 2



Network: 3



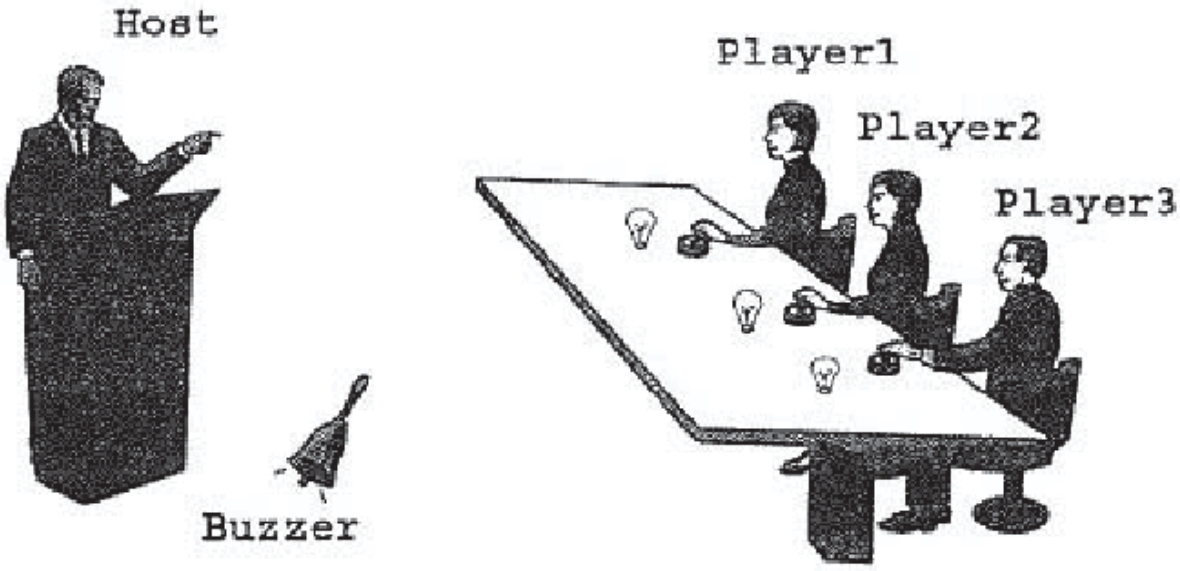


تشریح پروژه : با فشردن کلید Start خروجی مجازی Enable فعال می شود که در سایر Network ها مجوز انجام عملیات را می دهد . در Network ۲ در صورتی که سنسور Ball –sensor تشخیص دهد که توپی درون مخزن است باعث Set شدن خروجی UP-cylinder شده و در نتیجه توپ وارد قسمت پایین می شود . پس از آنکه سیلندر به انتها کورس خود رسید ، سنسور UP- sensor را فعال می کند در نتیجه سیلندر دوباره بسته می شود و مسیر ورودی توپ ازمخزن را می بندد. در این مرحله به کمک سنسور Colour- sensor رنگ توپ تشخیص داده می شود و اگر سیاه باشد ، خروجی سنسور یک است و در نتیجه در Network 4 پس از آنکه سیلندر Pusher را فعال کرد و سیلندر به انتهای کورس رسید سنسور Pusher- sensor فعال شده باعث حرکت سیلندر Down-cylinder می گردد یعنی توپ به سمت پایین حرکت

کرده و به کمک تیغه منحرف کننده -Pusher وارد ظرف توپ های سیاه میشود .پس از آنکه توپ وارد ظرف شد در 5 Network توسط سنسور نوری Storage-sensor حرکت آن تشخیص داده شده و در نتیجه سیلندر Pusher و Down را می بندد و پروسه مجدداً تکرار می شود . اینبار اگر توپ سفید باشد سنسور Colour خروجی صفر داده و تنها Down- cylinder می گردد یعنی توپ به سمت پایین حرکت کرده و به کمک تیغه منحرف کننده Pusher وارد ظرف توپ های سیاه می شود . پس از آنکه توپ وارد ظرف شد در 5 Network توسط سنسور نوری Storage-sensor حرکت آن تشخیص داده شده و در نتیجه سیلندر -Pusher و Down را می بندد و پروسه مجدداً تکرار می شود . اینبار اگر توپ سفید باشد سنسور Colour خروجی صفر داده و تنها Down- cylinder باز شده وتوپ مستقیم وارد ظرف سفید می شود و مجدداً بسته می شود و سیکل تکرار می شود .

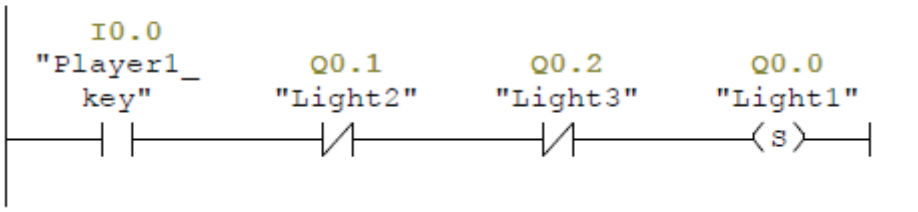
9-16 عنوان پروژه : تعیین اولویت در فشردن کلید زنگ مسابقه

شرح پروژه : پس از آنکه مجری مسابقه سوال را مطرح کرد ، هر شرکت کننده ای که جواب را می داند و زودتر کلید شستی جلوی خود را فشار دهد ، لامپ جلوی شرکت کننده روشن شده و همزمان زنگ مسابقه به مدت ۲ثانیه به صدا در می آید . حال اگر پس از آن هر یک ازشرکت کنندگان مسابقه هر چند بار کلید خود را فشاردهند ، چراغ آنها روشن نمی شود ، یعنی هر کسی که زودتر کلید را فشار دهد اولویت داشته و این امکان را از دیگران سلب می کند . این لامپ توسط کلید Reset که جلوی مجری قرار دارد خاموش می شود و این پروسه مجدداً تکرار می گردد .

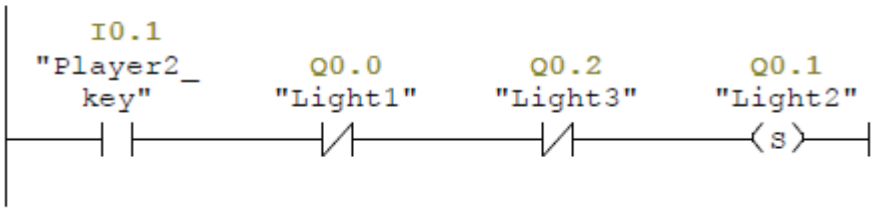


راه حل پروژه :

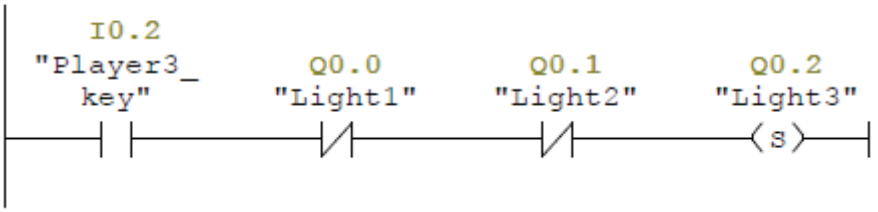
Network: 1



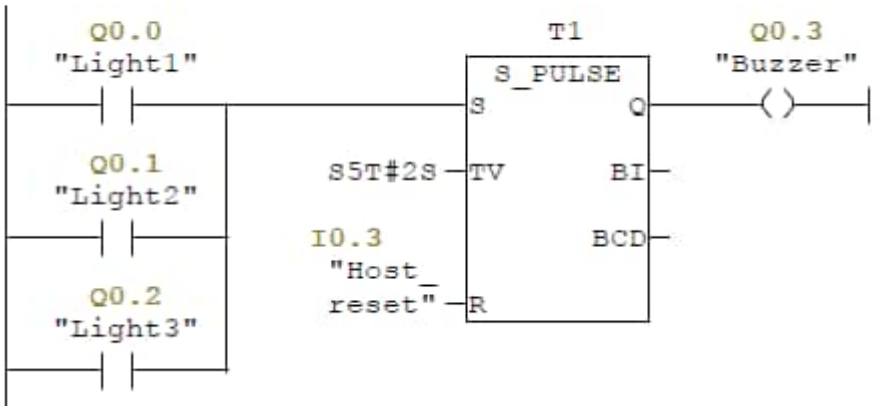
Network: 2

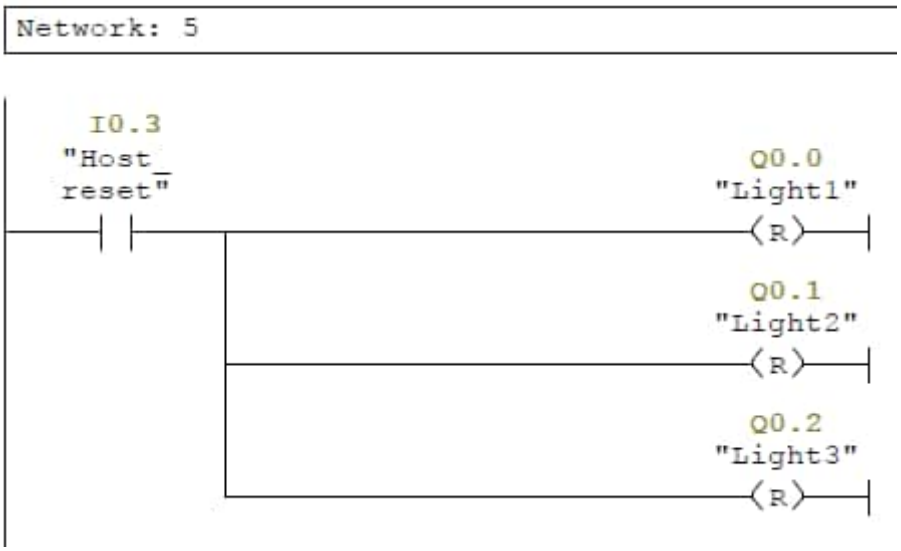


Network: 3



Network: 4



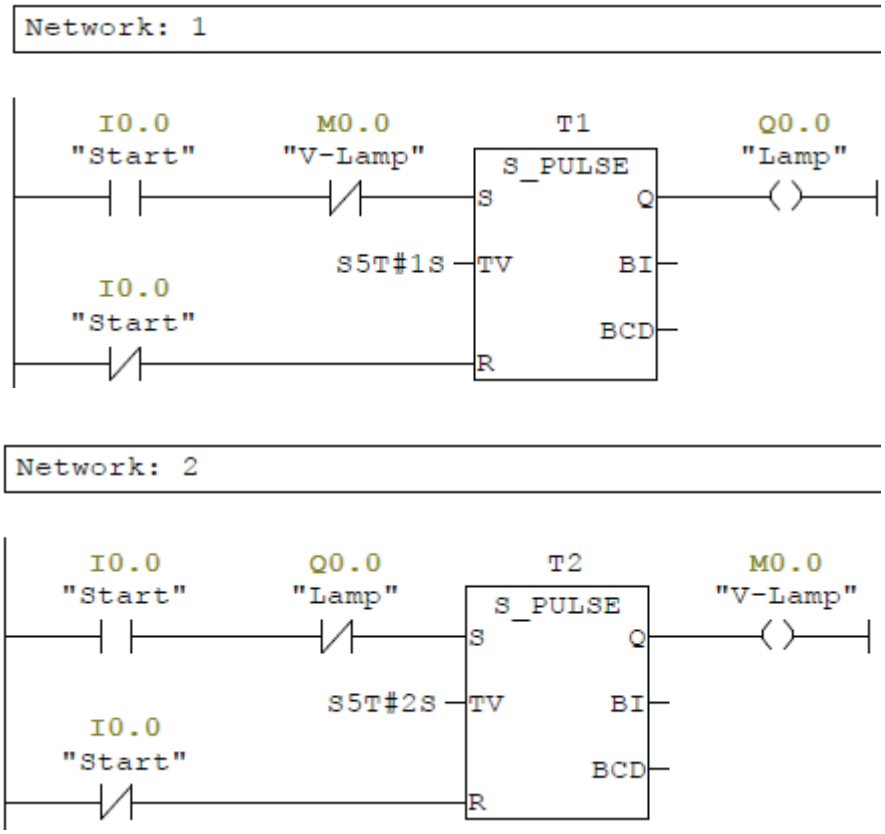


تشریح پروژه : وقتی Player_1 دکمه را فشار دهد ، به شرطی که قبل از او Player_2 و Player_3 دکمه خودرا نزده باشند ، Light1 روشن می شود و این شرط با تیغه بسته Light2 و Light3 که به صورت AND با دکمه Player_1 قرار دارند به وجود می آید . به عنوان مثال اگر قبل از Player_1 ،Player_3 دکمه خود را زده باشد ، چون Light3 روشن می شود تیغه N.C Light3 به حالت باز می رود و تغییر وضعیت کلید های Player_1 و Player_2 تاثیری روی خروجی انها نمی گذارد . در Network 4 ،هر یک از خروجی ها که روشن شود ، به کمک تایمر (S-pulse) ، خروجی Buzzer به مدت ۲ ثانیه روشن و سپس خاموش می گردد . در Network 5 نیز مجری (Host) با فشردن کلید reset کلیه خروجی های Light و در صورت روشن بودن Buzzer را نیز Reset می کند.

17-9عنوان پروژه : چراغ چشمک زن به کمک تایمر (S-pulse)

شرح پروژه : در این پروژه به کمک تایمر S-pulse چراغ چشمک زن تک لامپی را طراحی کنید که پریود زمانی ۱ ثانیه روشن و ۲ ثانیه خاموش شروع به کار کند. البته زمانی که ورودی start وصل باشد.

راه حل پروژه :



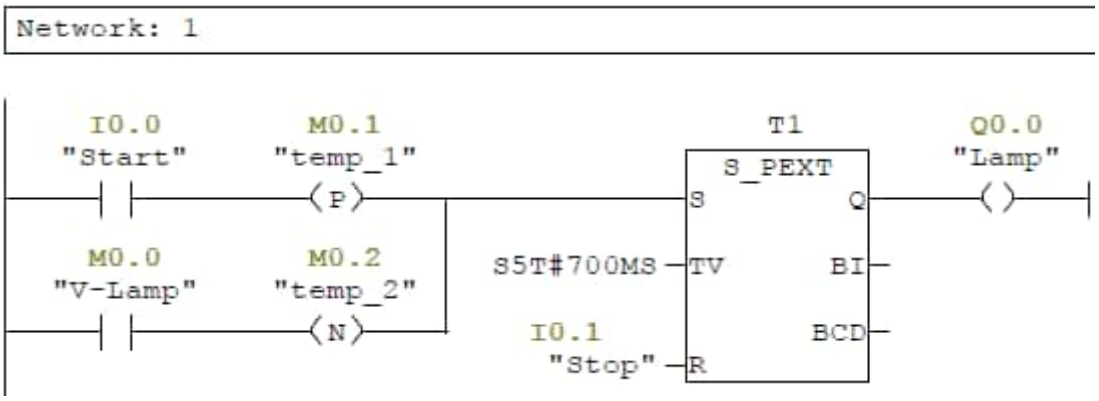
تشریح پروژه: با زدن کلید Start ، چون خروجی مجازی V-LAMP در ابتدا خاموش می باشد و تیغه N.C آن سر راه Start قرار دارد تایمر T1 به مدت ۱ ثانیه خروجی LAMP را روشن نگه می دارد. در

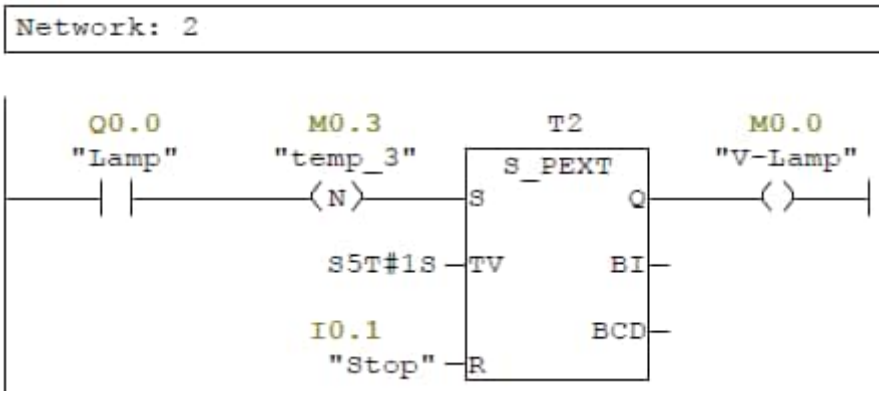
Network 2 با خاموش شدن LAMP چون تیغه N.C آن با Start , AND شده و به T2 وصل است ، خروجی مجازی V-LAMP را به مدت ۲ ثانیه روشن نگه می دارد که درحقیقت زمان Off خروجی اصلی LAMP است. با خاموش شدن V-LAMP ، T1 مجدداً شروع به کار می کندو این عمل تا زمانی که Start فعال باشد ادامه می یابد تیغ ه N.C کلید Start نیز به Reset هر دو تایمر وصل شده که در صورت غیر فعال شدن این ورودی هر دو تایمر را غیر فعال می کند .

18-9عنوان پروژه : چراغ چشمک زن به کمک تایمر (S-PEXT)

شرح پروژه : به کمک تایمر S-PEXT چراغ چشمک زن تک لامپی را طراحی کنید به طوری که ۷۰۰ میلی ثانیه روشن و ۱ ثانیه خاموش باشد و این عمل با زدن لحظه ای یا دائم شستی Start صورت گیرد و تا زمان فشردن شستی Stop ادامه یابد.

راه حل پروژه:



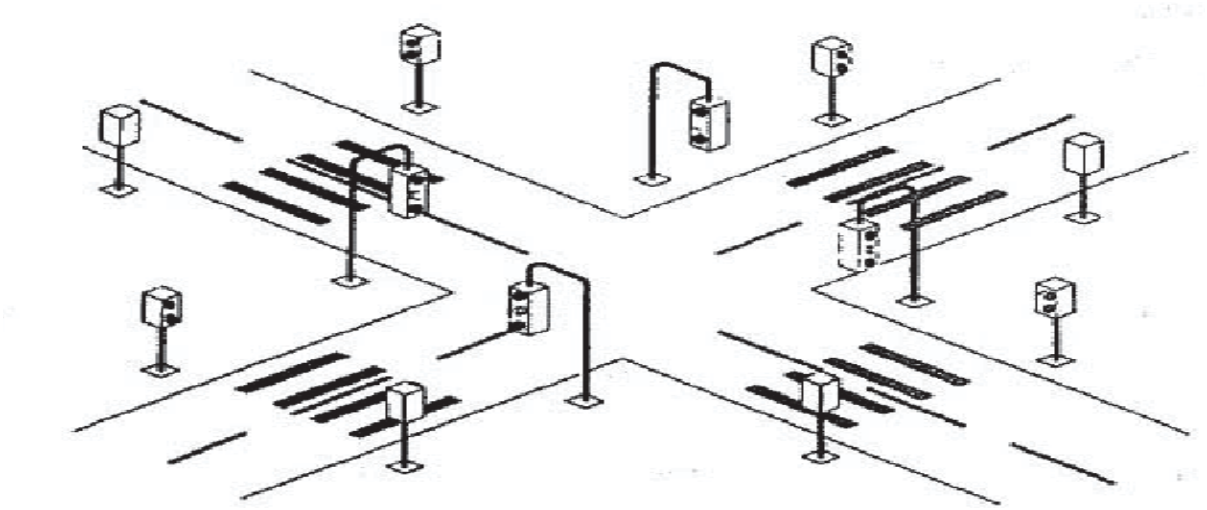


تشریح پروژه : با فشردن کلید Start با لبه بالا رونده آن تایمر T1 ، خروجی LAMP را به مدت ۷۰۰ میلی ثانیه روشن نگه می دارد. در Network2 با خاموش شدن LAMP به کمک تابع تشخیص لبه پایین روند — (N) — تایمر T2 شروع به کار کردن و V-Lamp را به مدت ۱ ثانیه روشن نگه می دارد که درحقیقت زمان off خروجی اصلی LAMP است. همان طور که در Network1 مشاهده می کنید خروجی V-Lamp به کمک تابع — (N)- به صورت موازی با کلید به ورودی T1 وصل شده است که باعث می شود با خاموش شدن V-Lamp تایمر T1 مجدداً فعال شده و شروع به کار کند و این عمل تا زمان زدن کلید Stop ادامه پیدا می کند. در Network1 هدف از قرار دادن تابع — (p)- این است که اگر کاربر کلید

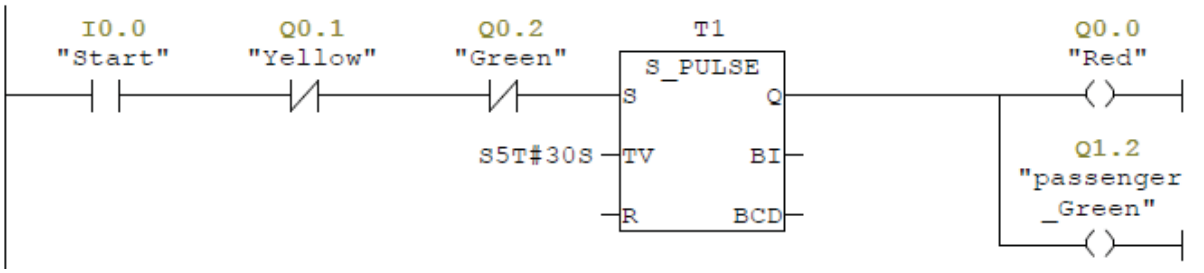
Start را نگه داشته باشد مانع از تحرک مجدد توسط V-Lamp نشود چون این دو با هم OR بوده و اگر Start دائم فعال باشد ، تغییر وضعیت شاخه دوم دیده نمی شود .

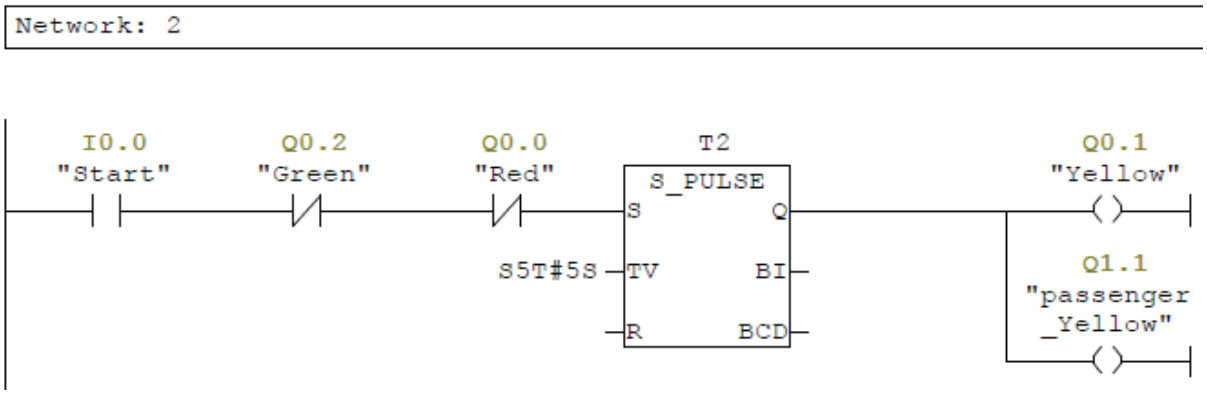
9-19عنوان پروژه : چراغ راهنمایی و رانندگی چهار راه ، با تایمر (S-pulse)

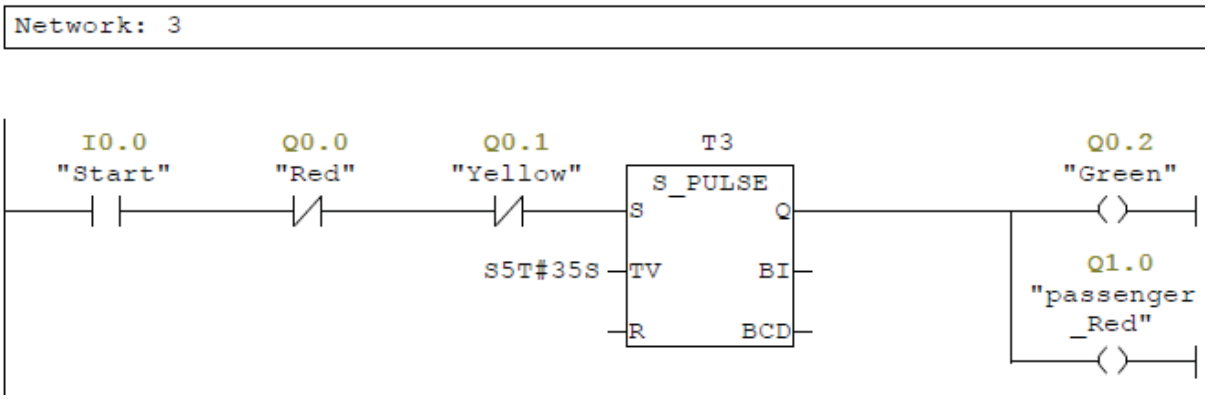
شرح پروژه : با زدن کلید Start سیستم شروع به کار کند . تا زمانی که چراغ سبز برای اتومبیل ها روشن است ، چراغ قرمز مخصوص عابر پیا ده نیز در سمت دیگه چهار راه روشن شود . زمانی که چراغ زرد برای اتومبیل روشن است ، چراغ زرد مخصوص عابر پیاده نیز در سمت دیگر چهار راه روشن باشد . زمانی که چراغ قرمز برای اتومبیل روشن است چراغ سبز مخصوص عابر پیاده نیز سمت دیگر چهار راه روشن باشد و مدت زمان ر وشن بودن چراغ سبز ۳۰ ثانیه چراغ زرد ۵ ثانیه و چراغ قرمز ۳۵ ثانیه باشد. این حلقه تا زمان نگه داشتن کلید Start ادامه پیدا کند.



Network: 1







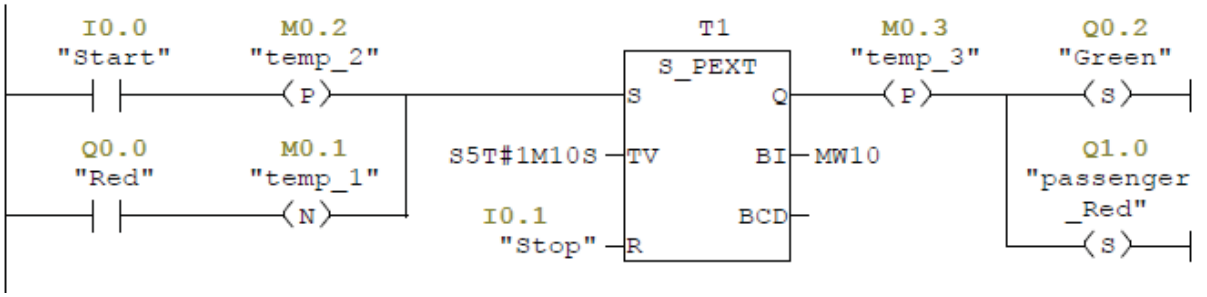
تشریح پروژه : با فشردن کلید start مشروط به این که چراغ زردوسبزچهارراه روشن نباشندتایمر T1به مدت ۳ ثانیه چراغ قرمزوسبزعابرپیاده راروشن نگه می دارد .درNetwork2 با خاموش شدن چراغ قرمز ، T2 با زمان ۵ ثانیه چراغ زرد و زرد عابر پیاده را روشن می کند . در Network3 با خاموش شدن چراغ سبز و قرمز عابر پیاده روشن می شود و پس از ۳۵ ثانیه خاموش شده که این اتفاق در Network1 باعث شروع مجدداً تایمر T1 می شود و این حلقه تا زمان نگه داشتن Start ادامه پیدا می کند.

20-9عنوان پروژه : چراغ راهنمایی و رانندگی ، به کمک تنها یک تایمر (S-PEXT)

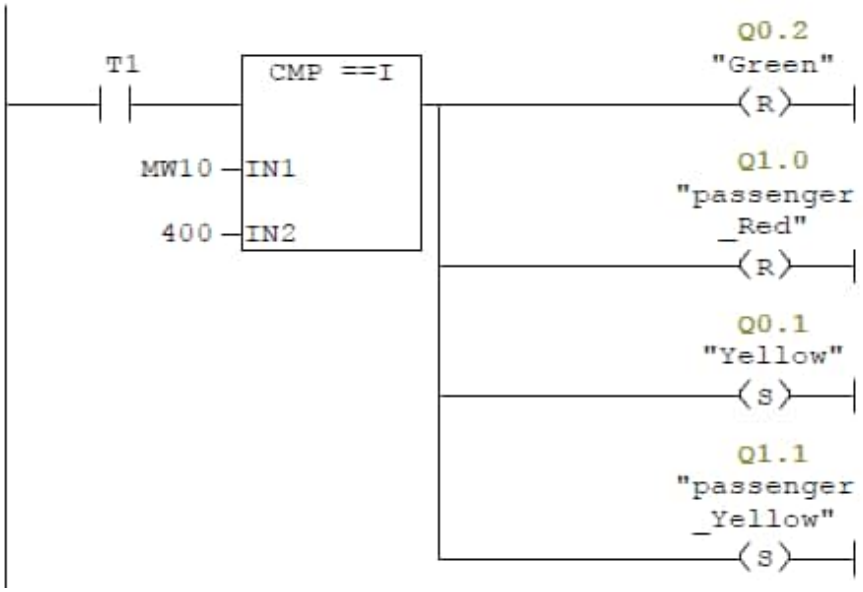
شرح پروژه : در این پروژه به کمک یک تایمر و دستورات مقایسه چراغ راهنمایی رانندگی با شرایط کاری پروژه قبلی (۲۰-۵) ، را طراحی کنید.

راه حل پروژه:

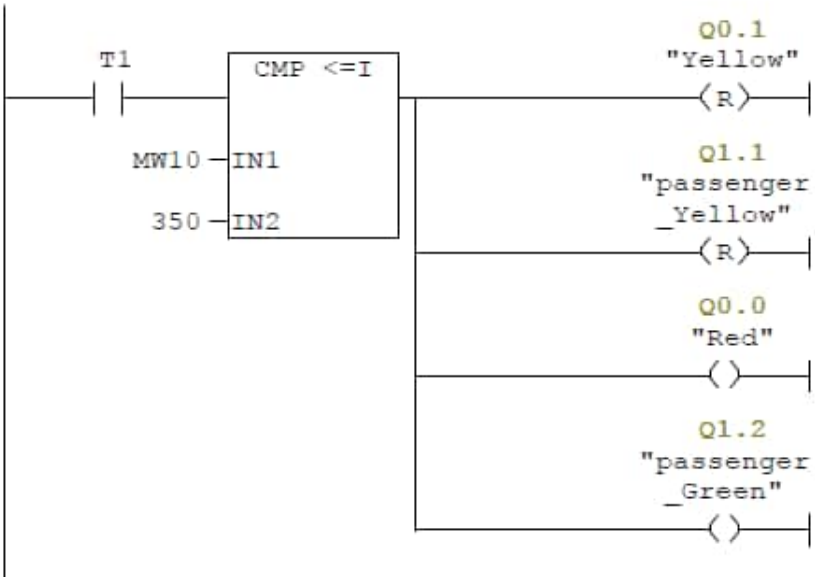
Network: 1

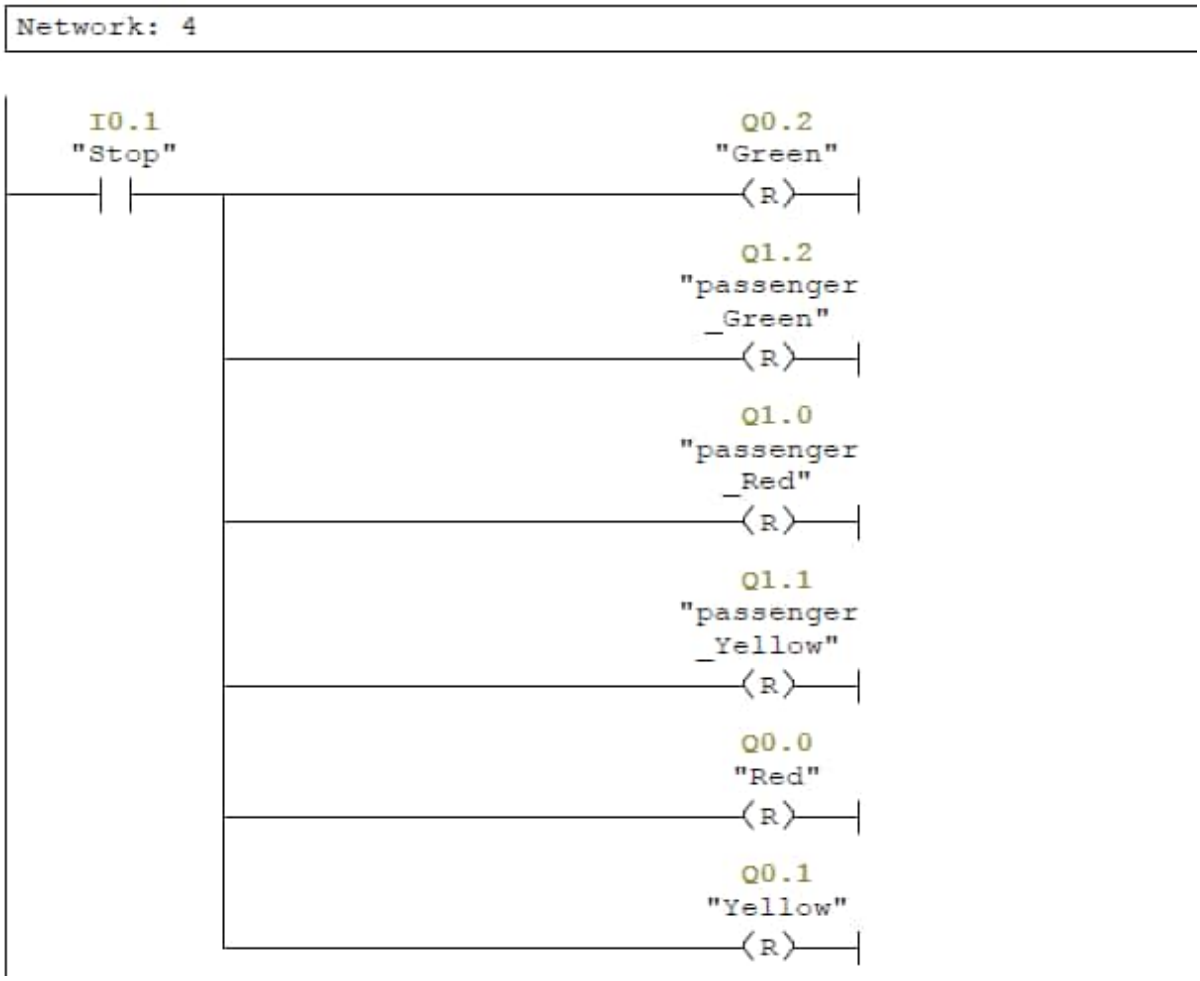


Network: 2



Network: 3





تشریح پروژه : با فشردن کلید Start تایمر T1 با زمان ۷۰ ثانیه که مجموع زمان های زرد ، سبز و قرمزاست شروع به کار می کند و در لحظه شروع ابتدا چراغ سبز و قرمزی بر عابر پیاده را روشن می کند .

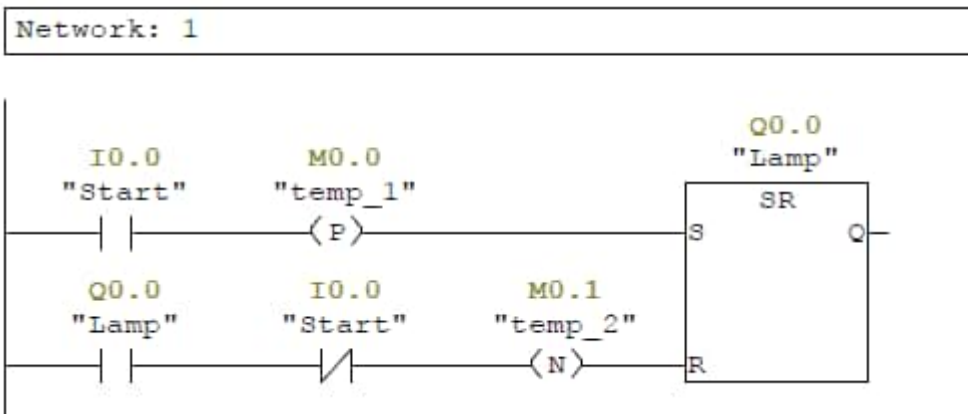
مقدار زمان جاری تایمر در متغییر MW10 قرار می گیرد که این متغییر در Network2 به مقایسگر Cmp==I نسبت داده شده و برابری آن با ۴۰۰ مقایسه می شود. علت قرار دادن ۴۰۰ این است که اولاً تایمر از ۷۰۰ به سمت صفر شروع به حرکت می کند زیرا زمان بیش از ۱۰ ثانیه است دقت پیش فرض آن 10ms است یعنی ۷۰ ثانیه را از ۷۰۰ شروع می کند حال چون می خواهیم ۳۰ ثانیه بعد از شروع به کار چراغ سبز خاموش شده و چراغ زرد روشن شود ، اگر ۷۰ را از ۳۰ کم کنیم ، ۴۰ بدست می آید که معادل آن یعنی ۴۰۰ را قرار می دهیم همین طور در Network3 زمانی که مقدار زمان جاری کوچکترمساوی ۳۵۰ یعنی ۳۵ ثانیه پس از شروع به کار تایمر چراغ زرد هم خاموش شده و در زمان باقی مانند چراغ قرمز وسبز عابر روشن می ماند پس از پایان با خاموش شدن چراغ قرمز در Network1 به کمک تابع —(N)- تایمر مجدداً شروع به کار می کند و این عمل تا زمانی که کلید Stop را فشار ندادیم ، ادامه می یابد. در Network2 و Network3 علت قرار دادن تیغه T1 در سر راه مقایسه گرها این است که در لحظه قبل از راه اندازی مقدار جاری تایمر T1 برابر صفر است

و در Network3 مقایسه کوچکتر مساوی از ۳۵۰ است ، باعث فعال شدن این مقایسگر شده و بدون فرمان ما از چراغ زرد شروع به کار می کند . در Network2 نیز برای حفاظت از اینکه قبل از Start عمل مقایسه صورت نگیرد قرار داده شده است. در Network4 نیز با تحریک Stop کلید خروجی ها Reset می شوند.

21-9عنوان پروژه : کلید پوش باتن (push botton)

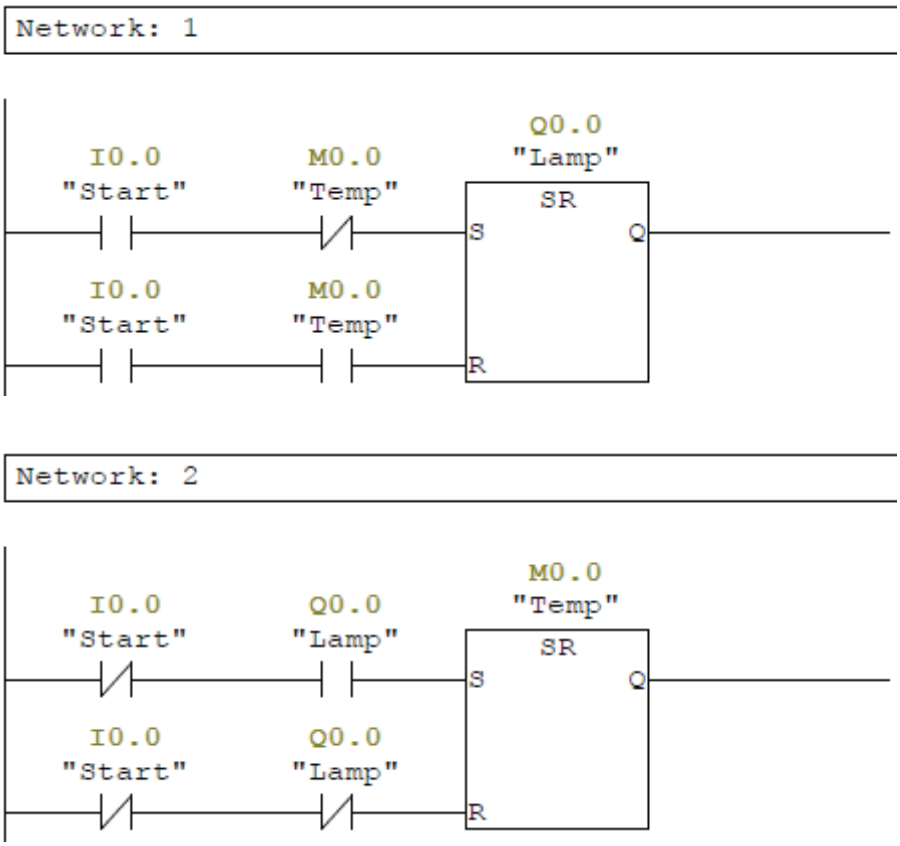
شرح پروژه : برنامه ای بنویسید که با فشردن لحظه ای کلید Start خروجی LAMP روشن شده و به طور دائم روشن بماند. یعنی اگر کلید Start قطع شود ، LAMP خاموش نشود. حال اگر کلید Start مجدداً فشرده شد ، LAMP خاموش شده و تا فشردن مجدد کلید Start خاموش بماند . اگر بخواهیم مثال ساده تری بزنیم مانند کلید Power دستگاه TV یا پخش CD که با یک بار فشردن کلید دستگاه روشن شده و با فشردن مجدد ان دستگاه خاموش شود .

راه حل پروژه (روش اول):



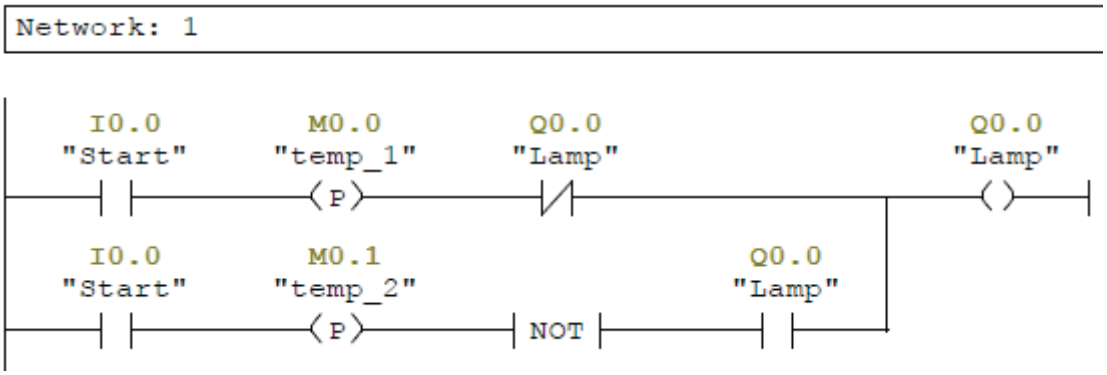
تشریح پروژه (روش اول) : این یکی از ساده ترین و در عین حال حرفه ای ترین روش برای طراحی Push Botton می باشد. با فشردن کلید Start پس از تابع —(p)- که با آن AND شده است ، تابع SR با پالس لحظه ای Set شده و خروجی را به طور دائم روشن نگه می دارد . پس از این اتفاق در پایه R تابع RS تیغه LAMPفعال است چون تیغه N.C کلید Start نیزقراردارد‌که به صورت ANDباLAMP وصل شده است .منطق یک تا ابتدای تابع —(N)- وجوددارد‌که بافشردن مجددکلید Start تیغه N.C کلید Start قطع شده که این اتفاق یک لبه پایین رونده را به وجود می آورد که تابع —(N)- از خود عبور داده و باعث Reset شدن خروجی LAMP می شود و این پروسه آماده شروع مجدد می شود.

راه حل پروژه (روش دوم):



تشریح پروژه (روش دوم): در این روش تابع RS به صورت به الاکلنگی (Flip- Flop) نسبت به یکدیگر روشن و خاموش می شوند. در ابتدا فرض را بر آن می گیریم که خروجی LAMP خاموش است. با فشار دادن کلید در Start تابع RS در Network1 خروجی خود یعنی LAMP را روشن می کند که این اتفاق از انجایی صورت می گیرد که خروجی موقت Temp در Network2 خاموش است و حاصل تیغه N.C آن با ورودی Start که وضعیت ANDرا دارند پایه S را فعال می کند . بلرها کردن کلید Start ، در Network2 چون تیغه N.C ورودی Start با تیغه N.O خروجی LAMP که وضعیت یک قرار دارد و با هم AND شده و به پایه S وصل شده اند تابع را فعال می کنند در این مرحله چون این تابع فعال است و در Network1 در پایه R تابع SR تیغه N.O خروجی موقت Temp فعال است ، با فشردن مجدد کلید Start این تابع Reset شده درنتیجه خروجی LAMP نیز خاموش می شود.

راه حل پروژه (روش سوم):



تشریح پروژه (روش سوم): هدف ازطراحی این پروژه به روش های مختلف آن است که هم این پروژه در طراحی های صنعتی واقعی بسیار کاربردی است و هم اگر روش های مختلف را برای طراحی یک پروژه مشاهده کنید درک شما از استفاده کاربردی از ابزار بالا می رود.

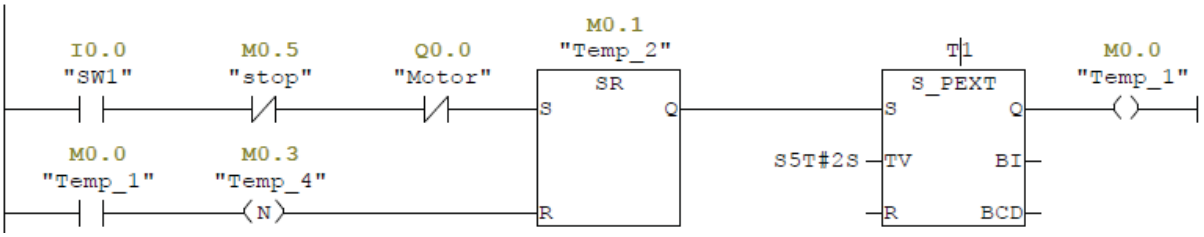
همان طوری که در راه حل پروژه مشاهده می کنید ، این بار توسط منطق X-OR یعنی $Y = A^{-}B + AB^{-}$ طراحی شده است.از آنجایی که طراحی X-OR در زبان LAD به صورت یک بلوک اماده نمی باشد ، می بایست مدار داخلی آن را مطابق شکل طراحی کنیم . با فشردن کلید Start با لبه بالا رونده از آنجایی که تیغه N.C خروجی سرآن قرار دارد برای یک لحظه LAMP را روشن می کند و درقسمت پایین این خط به صورت فیدبک بسته می ماند یعنی اگر دست از روی ورودی Start برداریم خروجی همچنان روشن می ماند حال اگر بار دیگر ورودی Start فشرده شود ، درخط پایین چون برای یک لحظه بلوک NOT باعث قطعی حلقه می شود درنتیجه خروجی LAMP خاموش می شود.

9-22 عنوان پروژه : کنترل خروجی با ۳بارفشردن کلید Sw1 دربازه ۲ثانیه ای :

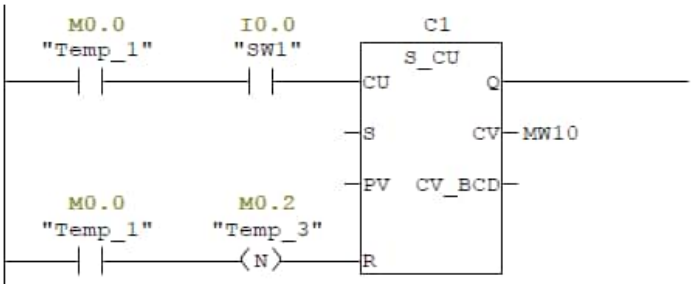
دراین پروژه باید در بازه زمانی حداکثر ۲ ثانیه ای کلید Sw1 را ۳ مرتبه قطع و وصل کنید تا خروجی Motor روشن گردد. زمان گیری از اولین تغییر وضعیت Sw1 شروع می شود و در صورتی که موفق شوید ، خروجی Motor پس از ۵ ثانیه خاموش شود و این عمل مجدداً تکرار شود . اگر در بازه ۲ ثانیه موفق نشوید ، کلید را ۳ فشار دهید مدار Reset شده و از ابتدا عمل شمارش انجام می دهد . همچنین اگر ۵ بار موفق به انجام این کار نشوید خروجی LAMP به مدت ۱۵ دقیقه روشن شده و اجازه فعال سازی Motor را نمی دهد ، پس از آن زمان همه چیز از اول شروع می شود . همچنین تا زمانی که Motor روشن است تغییرات ورودی دیده نشود.

راه حل پروژه :

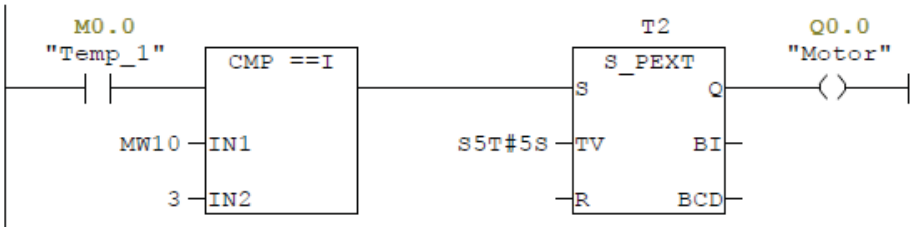
Network: 1



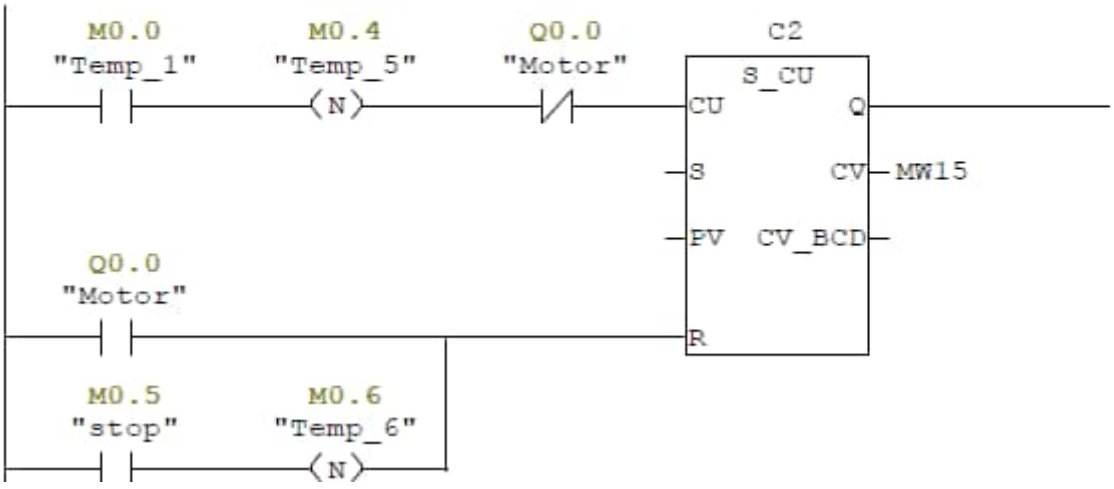
Network: 2



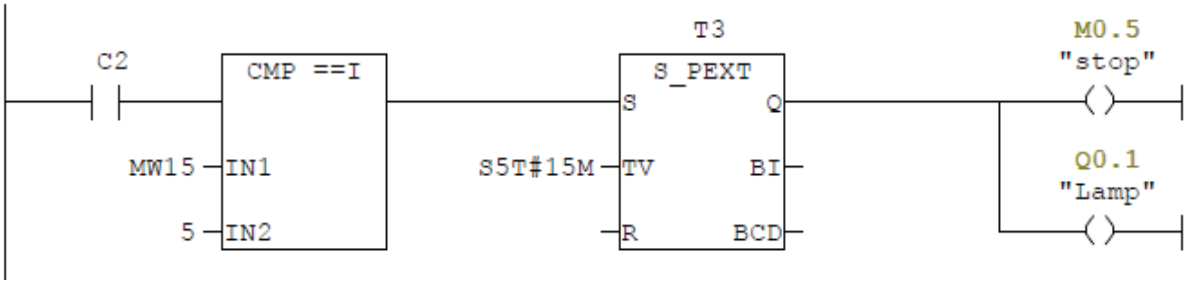
Network: 3



Network: 4



Network: 5

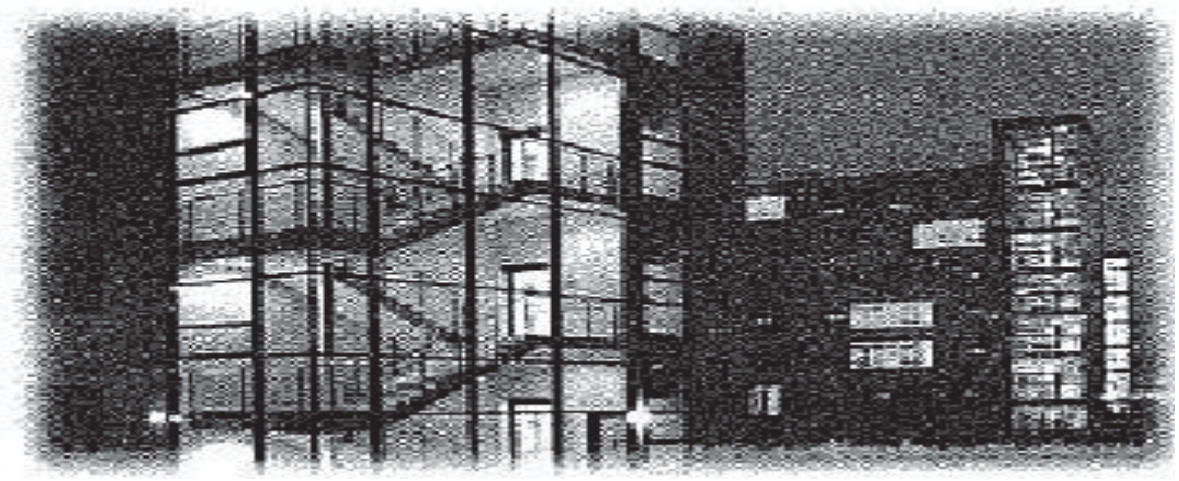


تشریح پروژه : با فشردن کلید Sw1 برای اولین بار چون Motor و تیغه Stop غیر فعال می باشند که به صورت N.C با Sw1 ، AND شده اند تابع SRفعال شده و همراه با آن تایمر T1 زمان گیری را به مدت ۲ ثانیه شروع می کند و در این مدت Temp-1 را روشن نگه می دارد . هدف از قرار گیری تابع RS این است که با تحریک مجدد کلید Sw1 تایمر مجدداً شروع به کار نکند.

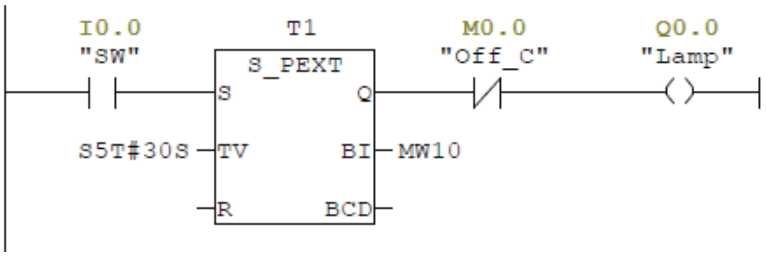
در Network2 تا زمانی که Temp-1 فعال است تغییر وضعیت SW1 توسط شمارنده C1 شمارش می گردد. که با پایان زمان T1 ، خروجی Temp-1 نیز خاموش می شود و چون این تیغه با یک تابع — (N)- به ورودی شمارنده C1 وصل شده است این اتفاق باعث Reset شدن C1 می گردد. در Network3 دستور مقایسگر CMP==I مشروط به فعال بودن T1 خروجی شمارنده C1 را با عدد ۳ مقایسه می کند و اگر در این مدت برابر شوند تایمر T2 را فعال کرده که خود باعث روشن شدن Motor به مدت ۵ ثانیه می گردد . در Network4 حال اگر فرض را به آن بگیریم که در این مدت موفق به ۳ بار فشردن Sw1 نشوید ، چون Motor روشن نشده است با خاموش شدن Temp چون تابع —(N)- متصل است شمارنده C2 را تحریک کرده و یک واحد افزایش می دهد که در واقع تعداد خطاهای ما است. اگر دفعه بعدی هم این اتفاق بیفتد باز هم یک واحد اضافه می گردد و همان طور که در Network5 مشاهده می کنید اگر ۵ بار تکرار شودتایمر T3 به مدت ۱۵ دقیقه فعال شده که علاوه بر روشن نگه داشتن LAMP باعث فعال شدن تیغه Stop می شود که این امر باعث غیر فعال شدن Network1 و عدم دیده شدن تغییر وضعیت Sw1 می گردد . پس از پایان ۱۵ دقیقه این تیغه دوباره به وضعیت اول بر می گردد . که در Network4 باعث Rest شدن شمارنده C2 می گردد. تا همه چیز از اول شروع شود . در ضمن اگر به عنوان مثال ۳ بار موفق به انجام کار نشوید ولی دفعه چهارم بتوانید در ۲ ثانیه ، ۳ بار Sw1 را فعال کنید ، با روشن شدن Motor شمارنده C2 نیز Rest می شود . در این پروژه می توانستیم تیغه Stop را حذف کرده و از تی غه LAMP استفاده ولی برای درک بهتر دانشجویان آن دو را از هم جدا کردیم .

9-23 عنوان پروژه : چراغ روشنایی راه پله (Staircase lighting)

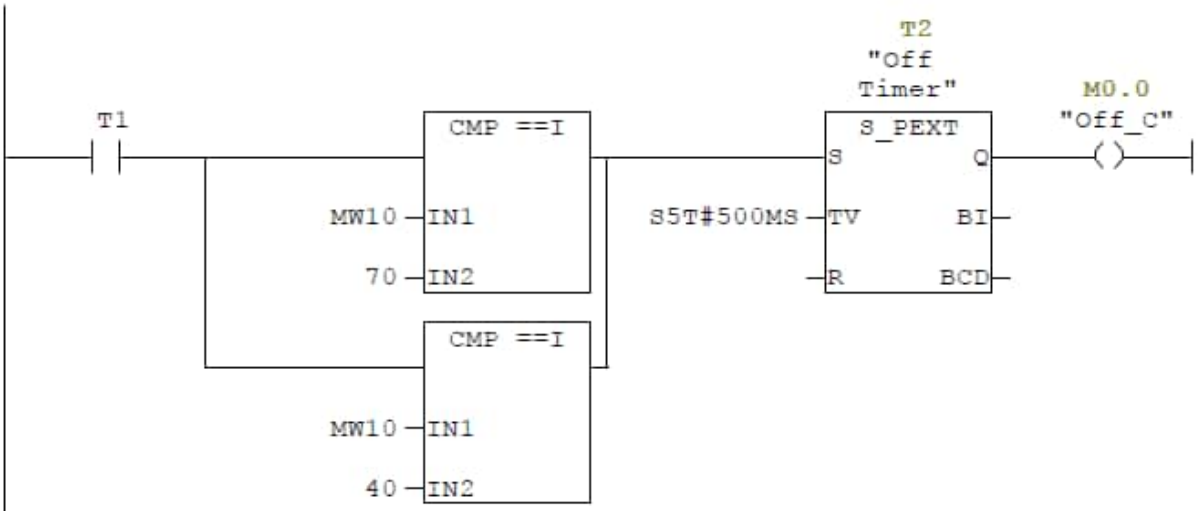
شرح پروژه :پروژه ای طراحی کنید که با فشردن شستی Sw ، چراغ راه پله به مدت ۳۰ ثانیه روشن شود. در ۱۰ ثانیه آخر لامپ برای اخطار اتمام زمان ۲ بار هر کدام به مدت ۰.۵ ثانیه چشمک بزند. یعنی یک بار در ثانیه ۲۳ و بار دیگر ۲۶ ثانیه پس از شروع به کار تایمر روشن شود . توجه کنید اگر کاربر ، در حین زمان گیری شستی Sw را مجدداً فشار داد ، زمان گیری از ابتدا شروع شود.



Network: 1



Network: 2

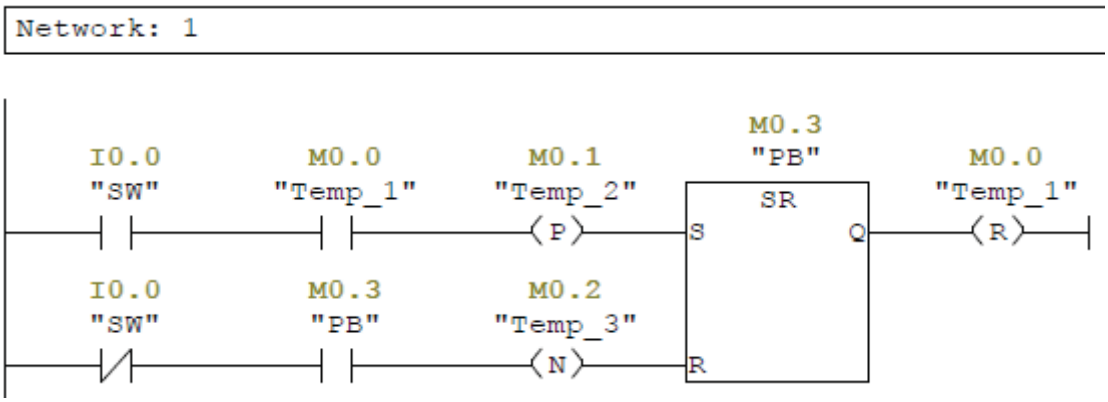


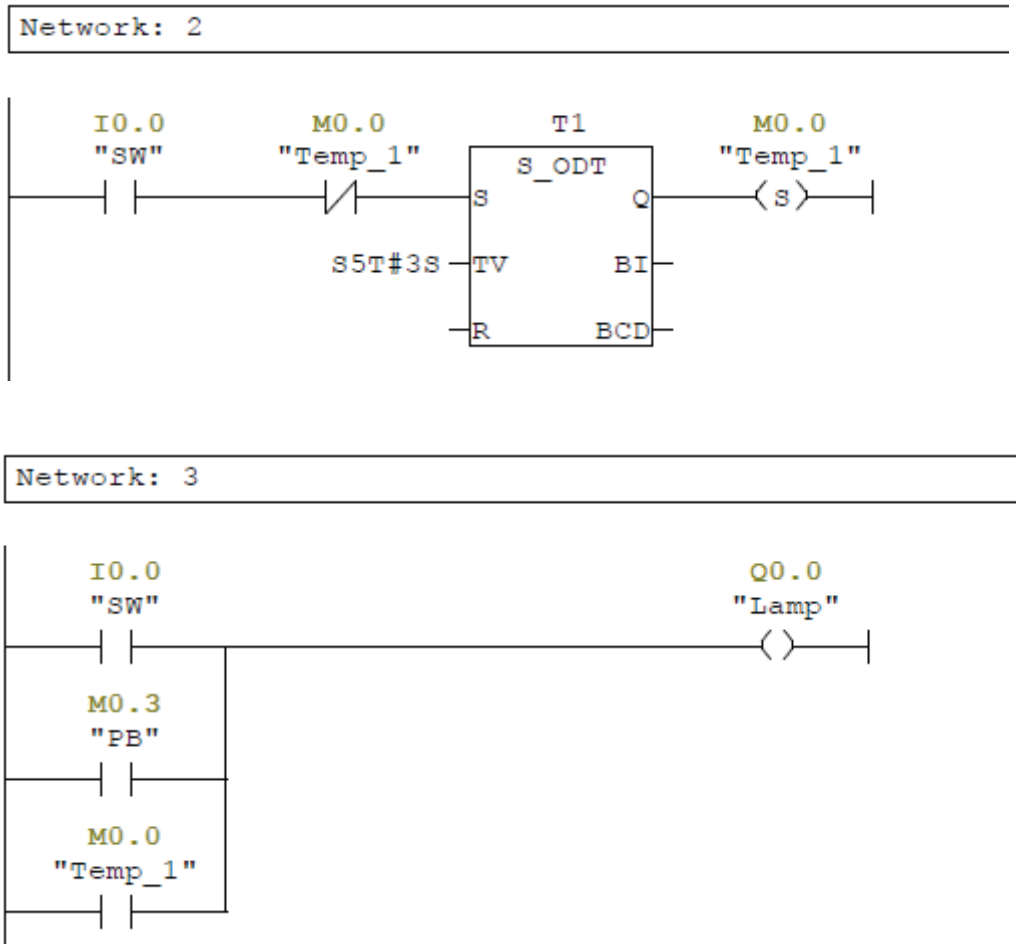
تشریح پروژه : با فشردن شستی Sw و تایمر T1 که از نوع S-PEXT می باشد خروجی LAMP به مدت ۳۰ ثانیه روشن نگه می دارد و به طور همزمان مقدار جاری زمان سپری شده در متغییر MW10 قرار می گیرد در Network2 مقدار جاری با دو مقدار ۷۰ و ۴۰ مقایسه می شود که معرف سپری شدن ۲۳ و ۲۶ ثانیه از شروع می باشد ، زیرا تایمر با ضریب 100ms یعنی از ۳۰۰ شروع به شمارش به صورت معکوس تا رسیدن به صفر می کند . ۲۳ ثانیه بعد یعنی ۲۳۰-۳۰۰ که ۷۰را بدست می آوریم و ۲۶ ثانیه بعد یعنی ۲۶۰-۳۰۰ که ۴۰ را بدست می آوریم و با دستور مقایسه پس از رسیدن به این زمان ها تایمر T2 بازمان 500ms شروع به کار کرده و خروجی مجازی OFF-C را فعال نگه می دارد که این اتفاق در Network1 به صورت موقت باعث خاموش نگه داشتن خروجی لامپ می شود چون تیغه N.C آن با خروجی AND,T1 شده و کنترل می کند.

9-24 عنوان پروژه : کنترل چراغ راه پله ، با دو وضعیت موقت و دائم

شرح پروژه : در این پروژه بازدن لحظه ای شستی Sw ، لامپ متناسب با زمان وصل بودن شستی Sw روشن باشد و در صورتی که شستی بیش از ۳ ثانیه نگه داشته شد ، خروجی لامپ به صورت دائم روشن بماند تا زمانی که شستی Sw مجدداً فشرده شود. در این حالت لامپ خاموش شده و پروسه مجدداً تکرار شود .

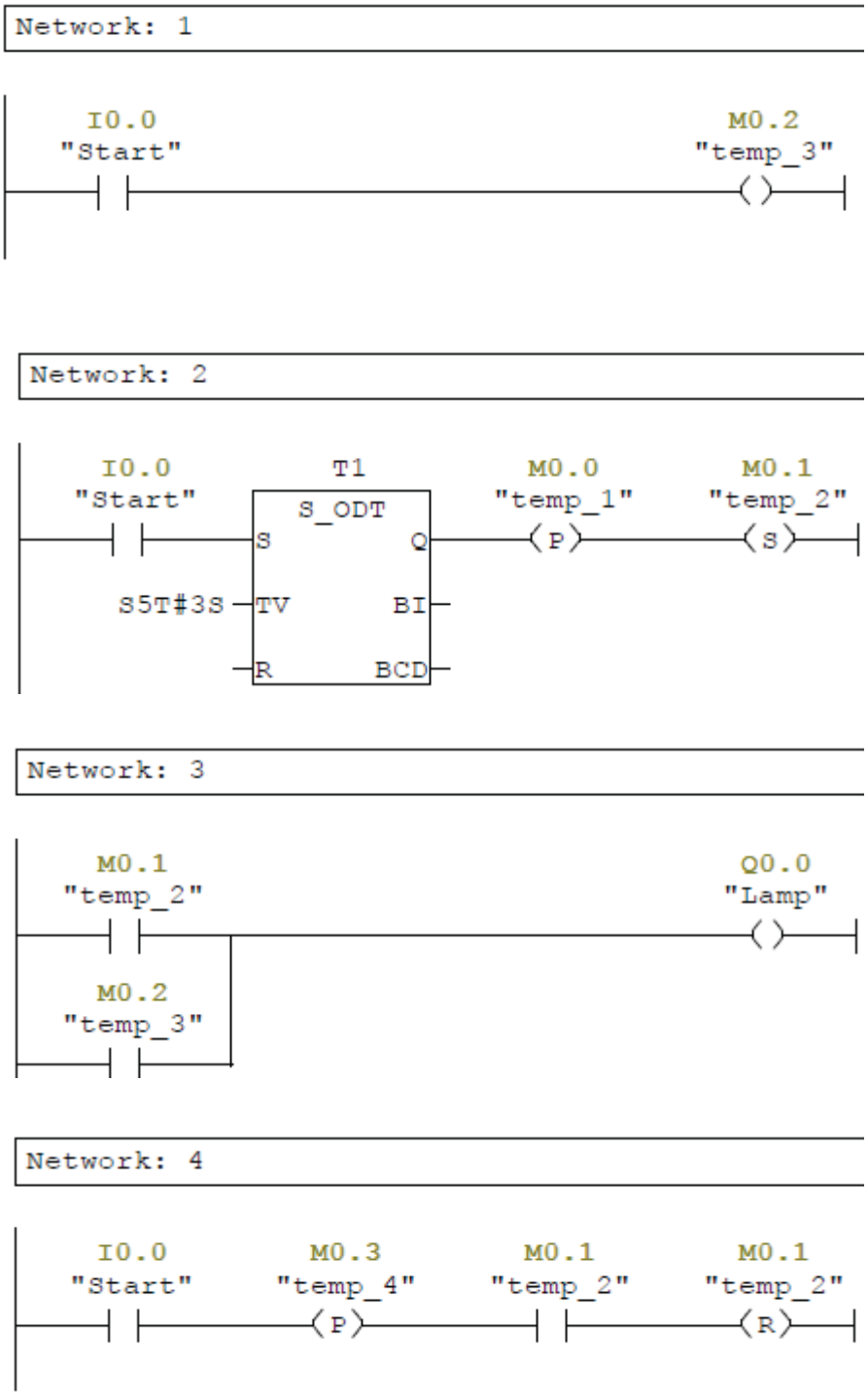
راه حل پروژه (روش اول):





تشریح پروژه (روش اول): این پروژه در حقیقت ترکیبی از مدار push batton و تایمری برای تشخیص نگه داشتن شستی Sw به مدت ۵ ثانیه است. در Network1 به علت قرار داشتن تیغه N.O ، Temp-1سر راه پایه S تابع SR درحقیقت این Network عملاً کار نمی کند . در Network3 با قطع و وصل موقت Sw خروجی LAMP نیز متناسب با آن روشن و خاموش می گردد. در Network3 از تایمر تاخیر در وصل S-ODT برای زمان گیری استفاده شده است بنا به خاصیت این تایمر ، در صورت یک بودن ورودی S آن و سپری شدن زمان تعریف شده خروجی روشن شده و تا زمان فعال بودن پایه S روشن می ماند. در صورتی که کلید Sw بیش از ۵ ثانیه نگه داشته شود خروجی آن یعنی Set -Temp-1 می شود که خود باعث سه اتفاق می شود : ابتدا از مدار خارج شدن این شبکه به علت قرار گیری تیغه N.C این خروجی سرراه ورودی S تایمر. در Network3 با فعال شدن این بیت خروجی Lamp روشن می شود و تا زمان فعال بودن آن روشن می ماند و در آخر در Network1 باعث می شود که تغییر وضعیت بعدی SW بیت PB فعال شود که خود باعث می شود در Network3 خروجی LAMP روشن بماند .بافشردن وقطع بعدی کلید Swدر Network1 مدارعمل کرده وباعث Reset ،PBوTemp-1 می شود که هم منجربه خاموش شدن Lamp و هم مدار را برای کار مجدد آماده می کند.

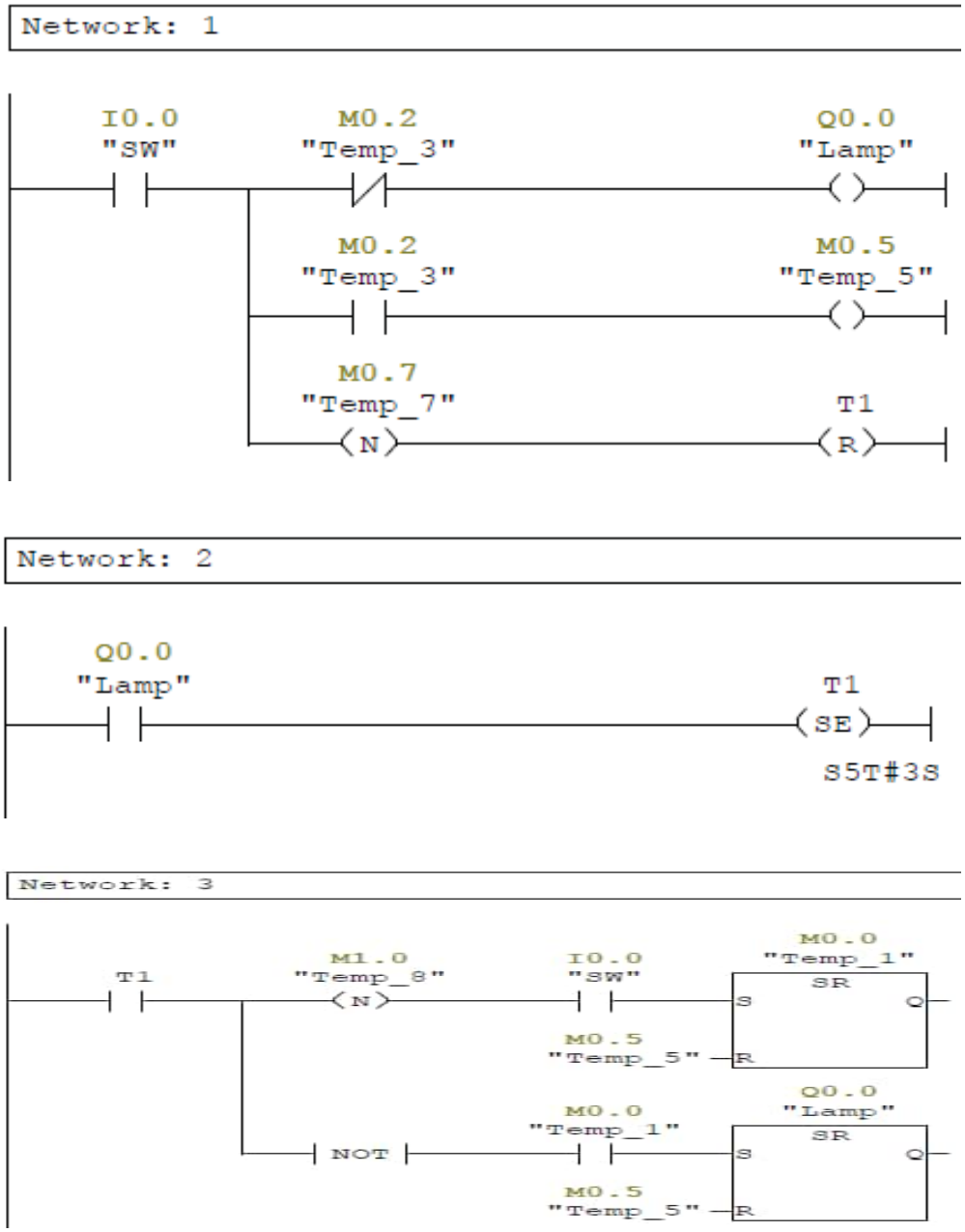
راه حل پروژه (روش دوم):



تشریح پروژه (روش دوم) : در Network1 با فشردن و قطع کلید Start خروجی Temp-3 قطع و وصل می گردد ، که در Network3 موجب روشن و خاموش شدن LAMP می شود. در Network2 همزمان با فشردن و نگه داشتن کلید Start تایمر T1 که از نوع S ODT — می باشد زمان گیری را شروع می کند و چون از نو ع تاخیر در وصل می باشد ، اگر ورودی

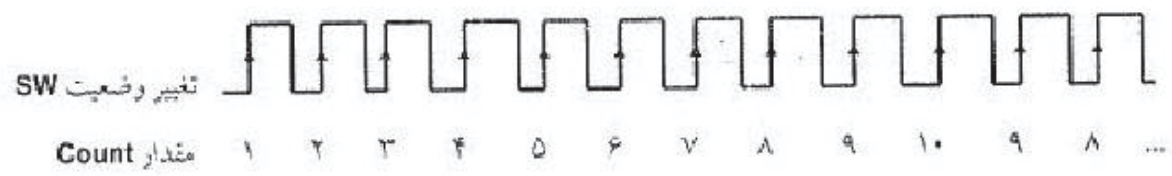
Start بیش از ۳ ثانیه نگه داشته شود ، این تایمر عمل کرد ، و Temp-2 را Set می کند که خود در Network3 موجب روسن ماندن LAMP می گردد. حال اگر ورودی Start را قطع نیز کنیم ، LAMP روشن می ماند . در NetworK4 وقتی Temp-2 فعال است با فشردن مجدد کلید Start ، خروجی Temp-2ریست شده در نتیجه LAMP خاموش می شود.

راه حل پروژه(روش سوم):

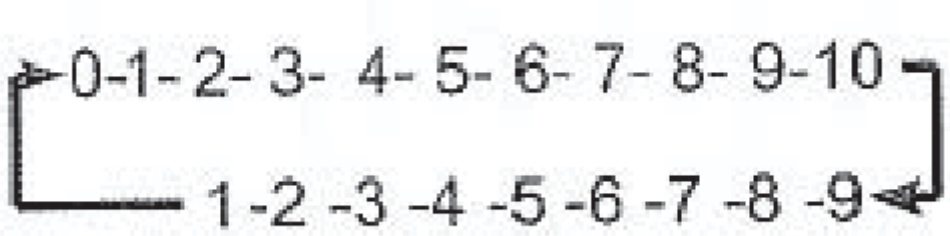


25_9شمارنده صعودی ونزولی باتحریک یک شستی

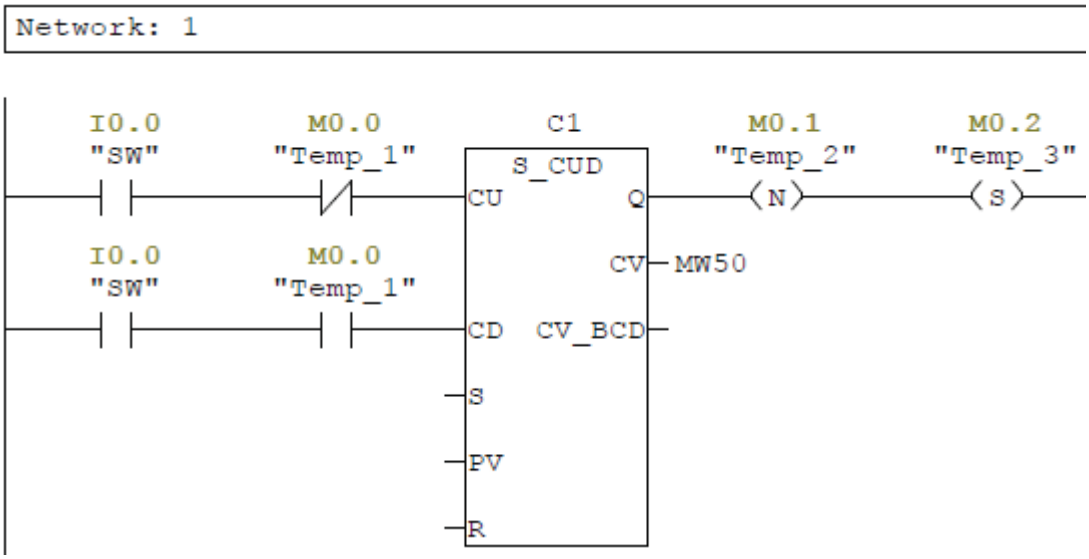
تشریح پروژه : در این پروژه قصد داریم با هر بار فشردن شستی Sw مقدار Count ، یک واحد اضافه شود تا ز مانی که این مقدار به ۱۰ برسد. سپس با هر بار فشردن Sw یک واحد از مقدار Count کم شده تا به صفر برسد ، و این عمل مرتب تکرار شود. برای درک بهتر شما توالی شمارش و تحریک Sw را در ادامه می اوریم.

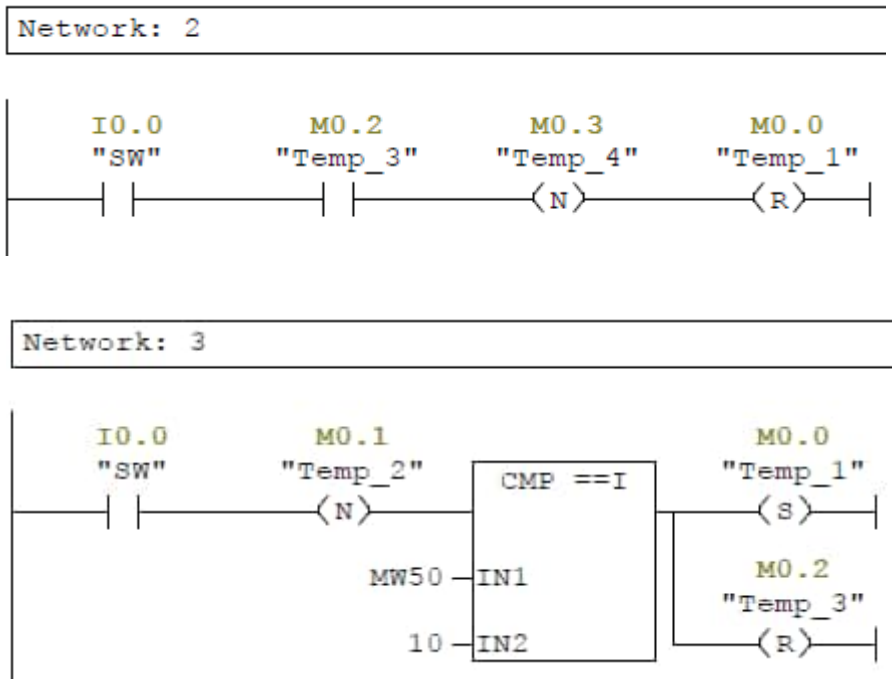


توالی Count :



راه حل پروژه :





تشریح پروژه : از آنجایی که در این پروژه تنها با یک شستی شمارنده به صورت صعودی و نزولی شمارش می کند ، پس باید ورودی Sw را هم به ورودی CU متصل کرد و هم به CD ، ولی با این شرطکه ۱۰ پالس اول را تنها ورودی CU دریافت کند و ۱۰ پالس دوم را ورودی CD.

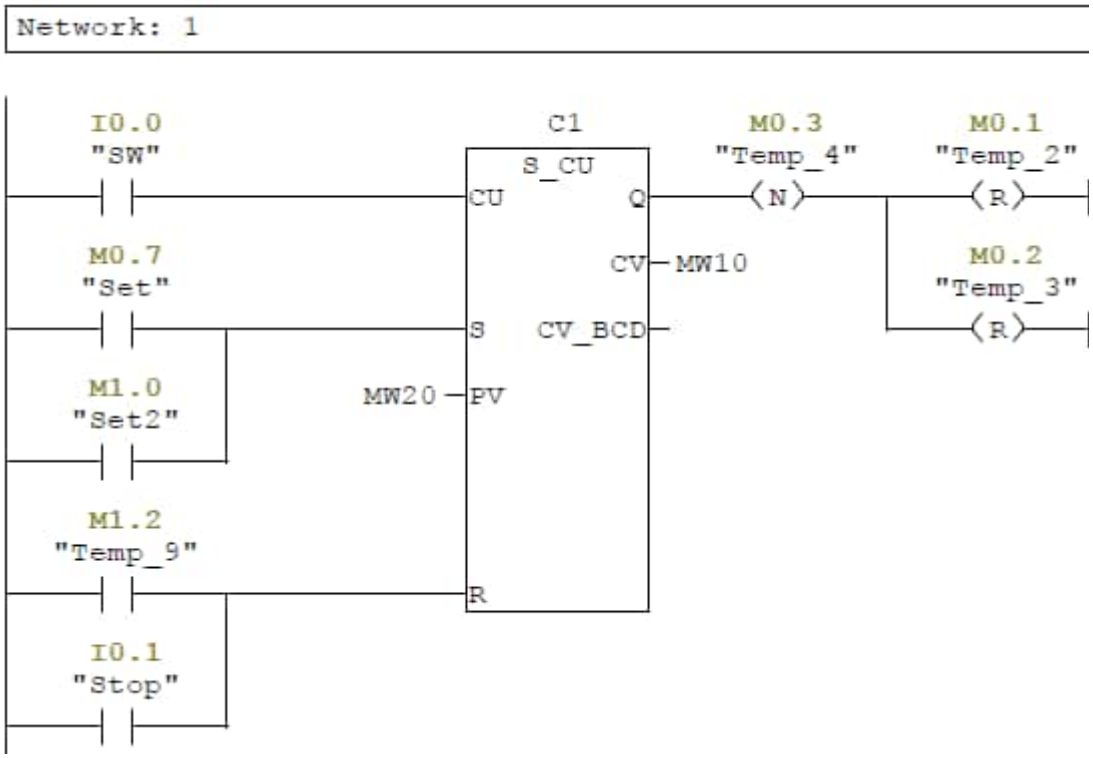
برای این کار تیغه N.C متغییر Temp-1 را سر راه Sw برای تحریک CU و تیغه N.Oآن را سر راه Sw برای CD قرار دادیم. چون در این حالت اولیه Temp-1 خاموش است در نتیجه تنها ورودی CU پالس را دریافت کرده و شمارش را انجام می دهد. در Network3 دستور مقایسه CMP==1 قرار دارد که مقدار جاری شمارنده را با عدد ۱۰ مقایسه می کند و زمانی که برابر شده اند ، ۱ -Temp را Set می کند. برای جلوگیری از این مشکل که برابر ۱۰ بودن آن باعث می شود که خود عدد ۱۰ دیده شود و در عمل تا ۹ شمارش انجام می شود ، دستور مقایسه را با لبه پایین رونده Sw فعال می کنیم تا پس از رها کردن شستی Sw مقایسه انجام شود و عدد ۱۰ نیز دیده شود . در Network2 برنامه تشخیص بازگشت و رسیدن به عدد صفر می باشد . در Network1 پس از آنکه شمارنده به صورت نزولی شمارش را ادامه داد زمانی که به ص فر می رسد خروجی Q آن خاموش می شود و با تابع — (N)- این اتفاق را تشخیص داده و متغیر Temp-3 را Set می کنیم. در Network2 اگر این متغییر روشن باشد و شستی Sw به وضعیت خاموش برود با تابع — (N)- متغیر Temp-1 را Reset می کنیم تا همه چیز از اول شروع شود و علت قرار دادن — (N)- نیز مانند حالت قبل برای آن است که خود عدد صفر نیز دیده شود. با شمارش مجدد صعودی شمارنده و رسیدن به عدد ۱۰ متغییر Temp-3 نیز Reset می شود.

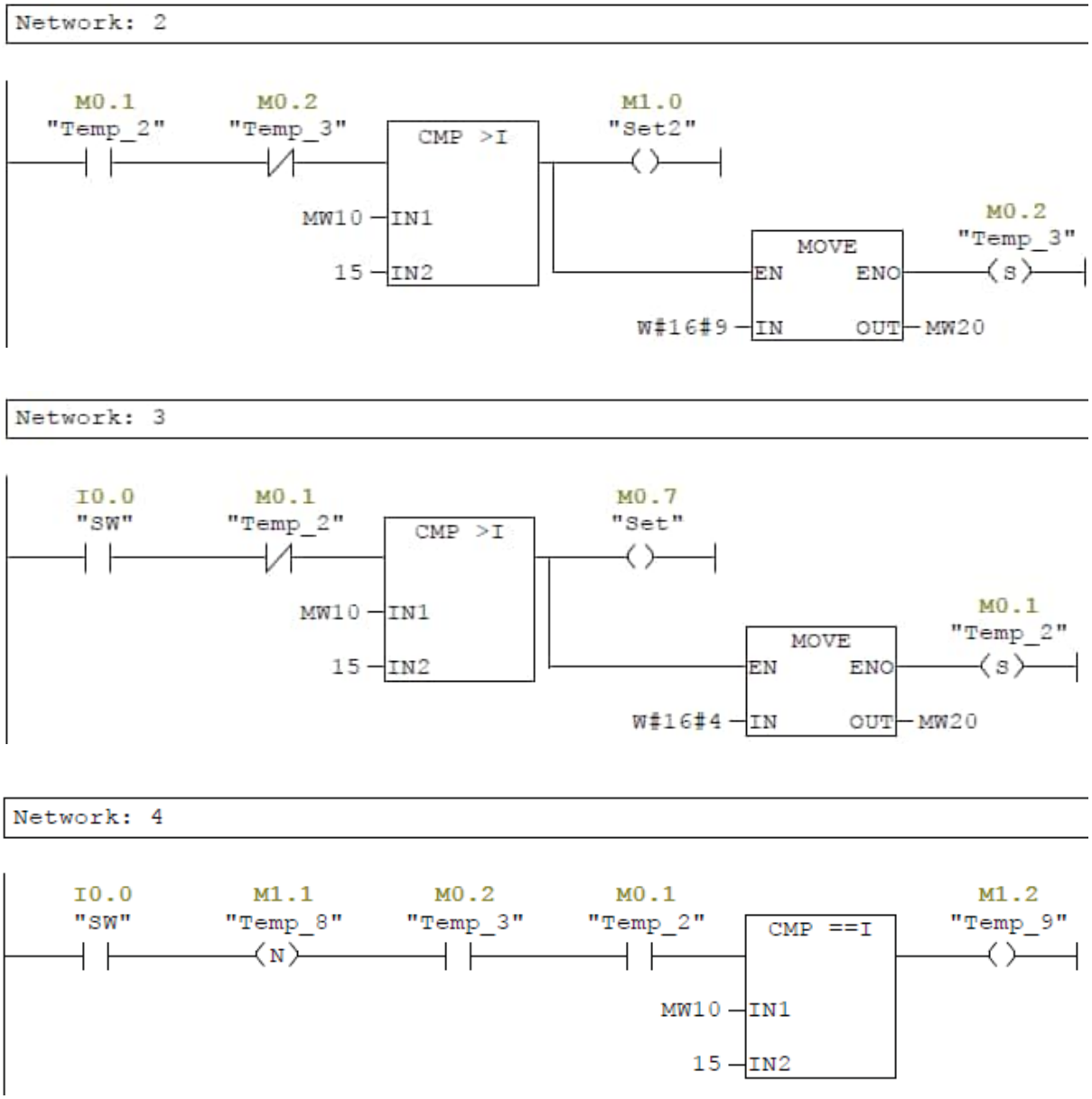
۹-۲۶ عنوان پروژه : شمارنده صعودی از چند مقدار مختلف

شرح پروژه : به کمک شمارنده CU برنامه ای بنویسید که با هر بار فشردن شستی Sw مقدار MW10 یک واحد اضافه شود تا به مقدار ۱۵ برسد ، پس از آن با پالس بعدی شمارنده مجدداً شروع به شمارش کند این بار از عدد ۵ تا۱۵ برسد ، پس از آن از ۱۰ شروع به شمارش کند تا به ۱۵ برسد و این حلقه مجدداً تکرار شود ، برای درک بهتر شما از این مساله ، به توالی زیر توجه کنید:

0-1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-5-6-7-8-9-10-11-12-13-14-15-10-11-12-13-14-15

راه حل پروژه:





تشریح پروژه هدف از طرح این پروژه علاوه بر یادگیری دقیق کاربرد دستورات مقایسه و Move ، به کار گیری ورودی pv برای مقدرا اولیه شمارنده است.

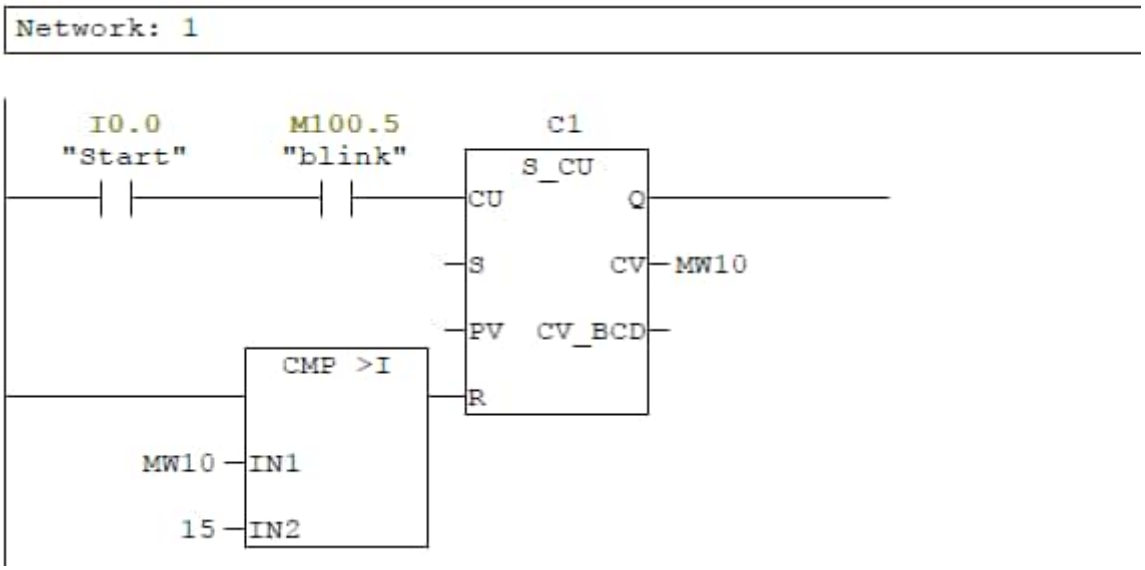
بلهر فشردن شستی Sw شمارنده C1 یک واحد به مقدار صفر MW10 اضافه می کند تا زمانی که در مطابق با Network3 مقدار آن از ۱۵ بیشتر شود یعنی شانزدهیم پالس ورودی . این اتفاق باعث بار گذاری مقدار ۴ در ورودی pv شمارنده می شود که توسط تابع Move اتفاق می افتد. دقت کنید که ورودی آن باید مستقیم با فرمت W # ۱۶# عدد را وارد کنید یا فرمت استاندارد شمارنده یعنی C# که هر دو حالت به صورت Hex دیده می شود. همزمان با فعال شدن مقایسه ، خروجی مجازی Set فعال می شود که پایه Set شمارنده را تحریک می کند و باعث ب ارگذاری شمارنده از عدد ۴ می شود . علت انتخاب عدد ۴ نیز آن است که چون همزمان با بارگذاری ورودی CU نیز تحریک می شود ، در عمل یک واحد بیشتر از

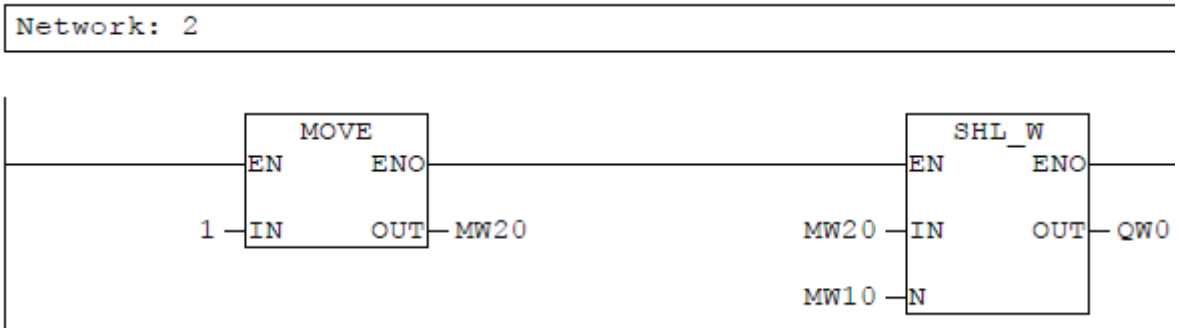
مقدار بارگذاری می باشد و عدد ۵ مدنظر ما دیده می گردد . با فعال شدن MOVE متغییر موقت Temp-2 نیز set می شود که باعث خا رج شدن این خط از برنامه می شود زیرا تیغه N.C آن سر راه این توابع قرار دارد . حال شمارنده از ۵ شروع به شمارش می کند . در Network2 با فعال شدن Temp-2 مقدار جاری مجدداً با ۱۵ مقایسه می شود و پالس بعدی آن باعث بارگذاری و تائید آن توسط فعال شدن Set2 در شمارنده می گردد و همزمان بیت Temp-3 نیز Set می شود که باعث خارج شدن این خط از برنامه می گردد . علت استفاده از دو بیت مختلف Setو Set2 در برنامه برای بارگذاری شمارنده آن است که اگر تنها یک Set را به Network3 های ۲و ۳ نسبت می دادیم ، از آنجایی اولویت در Network های پایین است ، اگر Set در Network2 فعال شود و در Network3 غیر فعال باشد اولویت با غیر فعال بودن است و شمارنده هرگز بارگذاری نمی گردد . به همین علت دو متغییر را در Network1 به صورت OR متصل و به پایه S شمارنده وصل کردیم. این بار شمارنده از ۱۰ شروع به شمارش می کند. در Network4 از آنجایی که قبلاً Temp-3 و Temp ، Set شده اند در صورت رسیدن مجدد مقدار MW10 به عدد ۱۵ باعث فعال شدن متغییر Temp-9 شده که متعاقب آن باعث Reset شدن شمارنده می گردد . علت استفاده از تابع — (N)- قبل از این توالی AND آن است که با لبه پایین رونده ورودی عمل Reset صورت بگیرد تا شما عدد 0 را هم داشته باشید. در Network1 با Reset شدن شمارنده خروجی Q آن نیز غیر فعال شود که با استفاده از تابع — (N)- این وضعیت را تشخیص داده و بیت های Temp-2 و را Reset کردیم تا همه چیز از اول شروع به کار کند.

۹-۲۷ عنوان پروژه: چشمکزن ۱۶ کاناله به کمک دستور Shiht (حلقه بسته)

شرح پروژه : برنامه ای بنویسید که به کمک دستور SHL-W ۱۶ خروجی به صورت یکی به جای دیگری روشن و خاموش شوند و در انتهای کار مجدداً از ابتدا شروع کند. پریود زمانی آن نیز یک ثانیه در نظر گرفته شود .

راه حل پروژه :



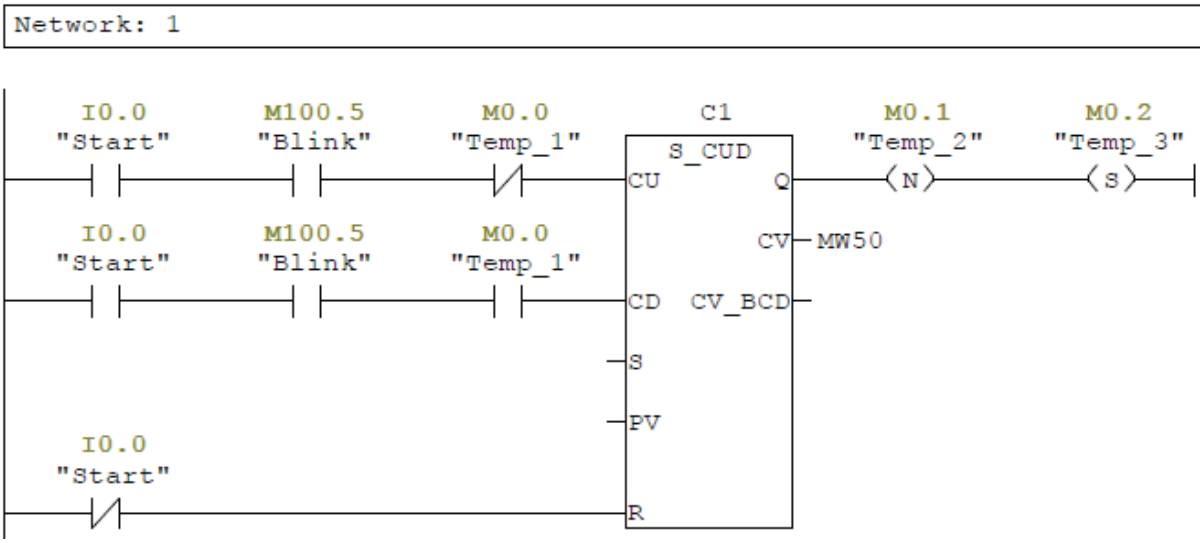


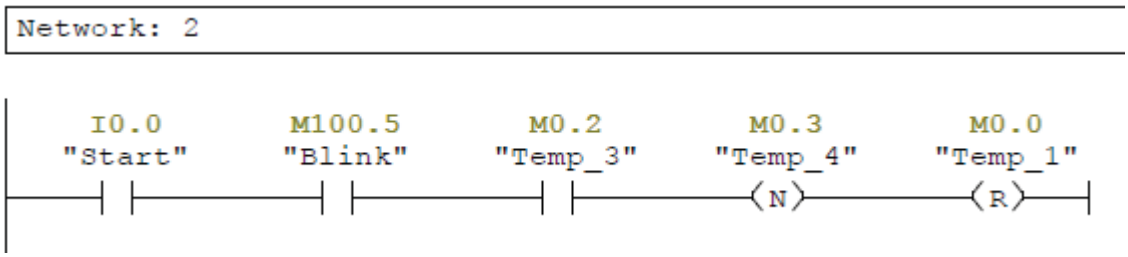
تشریح پروژه : برای حل این مساله Clock Memory را فعال کرده و آدرس Memory Byte را ۱۰۰ در نظر می گیریم . از آنجایی که پریود زمانی مد نظر ما ۱ ثانیه است از آدرس M100.5 استفاده می کنیم . با فعال کردن کلید Start شمارنده C1 با هر بار قطع و وصل تیغه M100.5 یک واحد شمارش را به صورت صعودی انجام داده و مقدار جاری را در متغییر MW10 قرار می دهد . از آنجایی که قصد داریم با رسیدن به بیت ۱۶ ، از ابتدا شروع به کار کند ، با قرار دادن تابع مقایسگر $Cmp >$ ۱ سر راه R شمارنده با رسیدن به ۱۶ شمارنده Reset کرده تا دوباره از ۰ شروع به شمارش کند . در Network2 ابتدا عدد ۱ را در روی تابع شیفت قرار می دهیم تا در ابتدای کار اولین بیت خروجی را از سمت راست روشن کند پس از خروجی شمارنده را به ورودی N تابع شیفت نسبت می دهیم که با هر واحد افزایش MW10 یک بیت به سمت چپ حرکت کند.

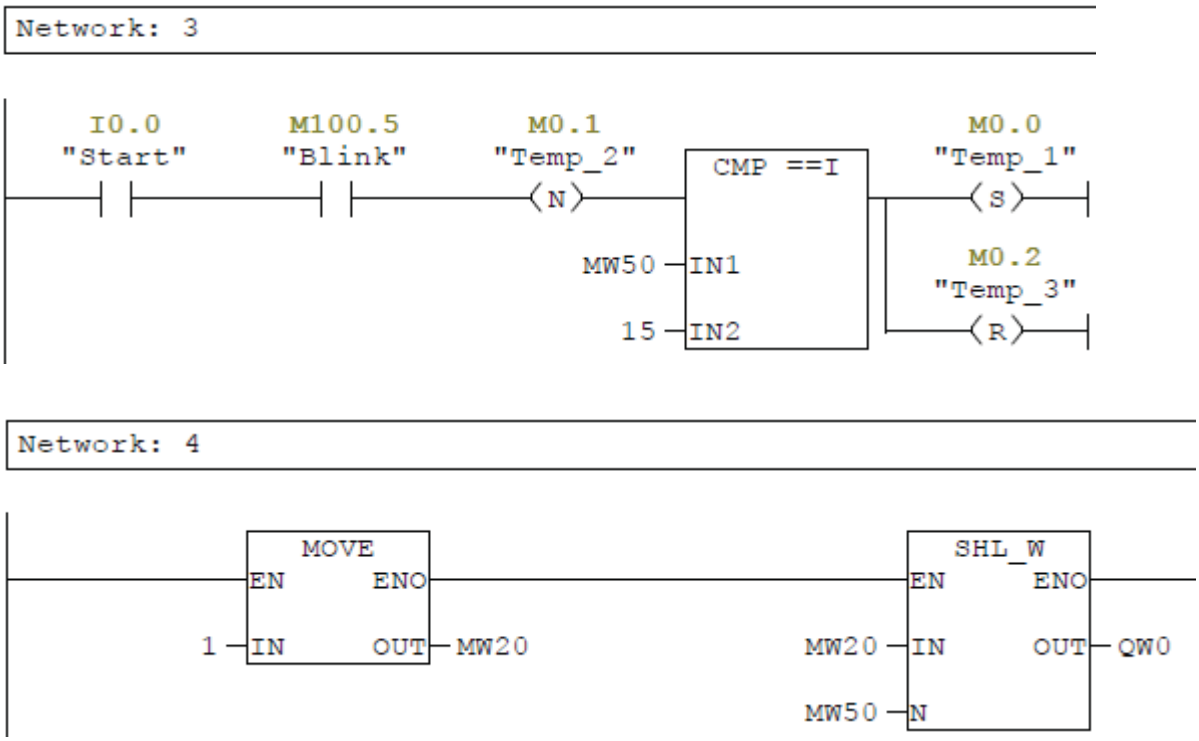
۲۸-9 عنوان پروژه : یکی به جای دیگری ۱۶ کاناله به صورت رفت و برگشت با دستور Shiht

شرح پروژه: این پروژه مشابه مثال قبلی می باشد با این تفاوت که مداری طراحی کنید که توالی روشن و خاموش شدن خروجی ها به صورت رفت و برگشت باشد .

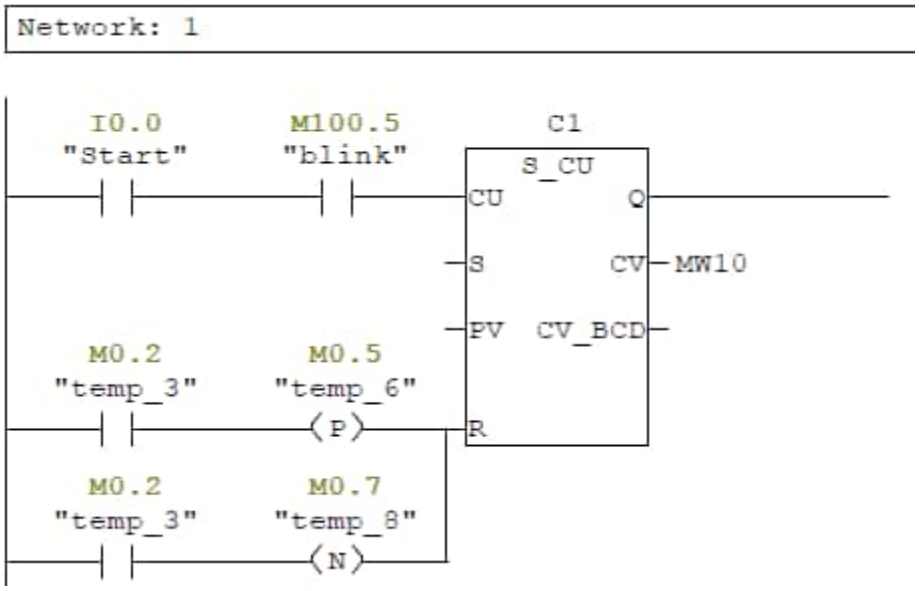
راه حل پروژه (روش اول):

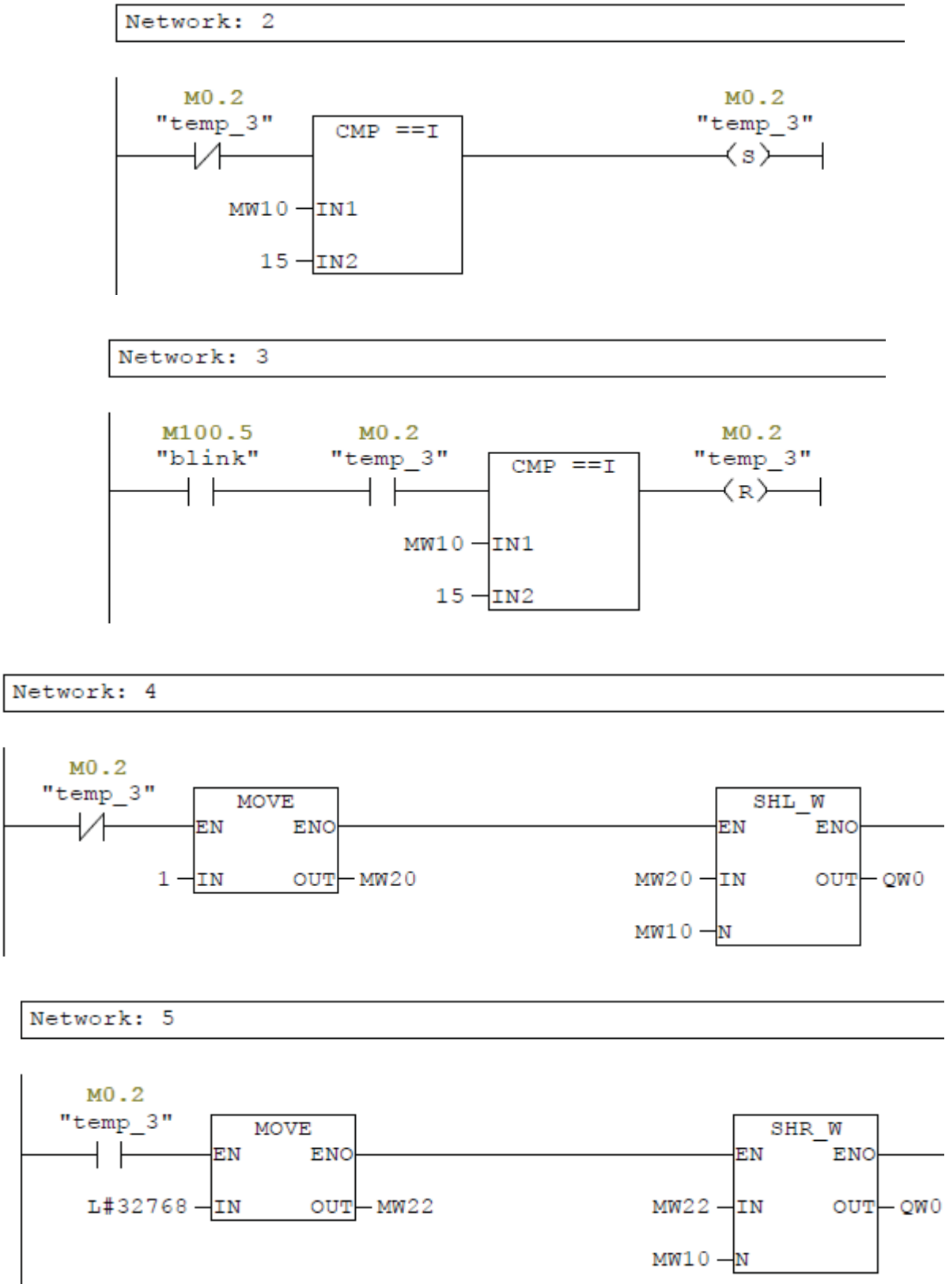






راه حل پروژه (روش دوم):





تشریح پروژه (روش دوم): این پروژه را به دو صورت متفاوت می توان طراحی کرد . روش اول به کمک شمارنده صعودی و نزولی ترکیبی که در مثال آورده شده ، به این صورت که خروجی جاری شمارنده را به ورودی N تابع شیفت نسبت می

دهیم و راه دوم به کمک دو تابع SHL-W و SHR-W که در اینجا به بررسی این روش می پردازیم . ولی در قسمت راه حل هر دو روش بررسی شده است . در روش دوم با زدن کلید Start شمارنده C1 با پالس ۱ ثانیه تیغه M100.5 شروع به شمارش می کند . در Network2 دستور مقایسه Cmp== I مقدار جاری شمارنده را با ۱۵ مقایسه می کند و د ر صورت برابر بودن خروجی مجازی Temp-3 را Set می کند که همزمان خود به علت قرار داشتن تیغه N.C آن سر راه E.n از مدار خارج می گردد . این اتفاق در Network1 نیز باعث Reset شدن شمارنده می شود . در حقیقت زمانی این اتفاق می افتد که بیت روشن شوند از سمت راست حرکت کرده و به آخرین بیت از سمت چپ رسیده است و حالا نوبت به حرکت به سمت راست است. قبل از آن ، Network4 در لحظه اول تابع SHL-W با مقدار ۱ بارگذاری شده و خروجی شمارنده به ورودی N تابع شیفت مرتبط است و باعث حرکت یکی به جای دیگری به ست چپ می شود . پس از Set شدن Temp-3 در Network5 تابع SHR-W با مقدار ۳۲۷۶۸ بارگذاری می شود که در واقع اولین بیت Word از سمت چپ است و این بار شمارنده C1 باعث حرکت بیت یکی پس از دیگری به سمت راست می گردد. در Network2 نیز دوباره مقدار جاری شمارنده با ۱۵ مقایسه می شود و در صورت برابری متغییر Temp-3 را Reset کرده که در Network1 باعث Reset شدن شمارنده C1 می شود و پروژه را از ابتدا راه اندازی می کند و این حلقه تکرار می گردد .

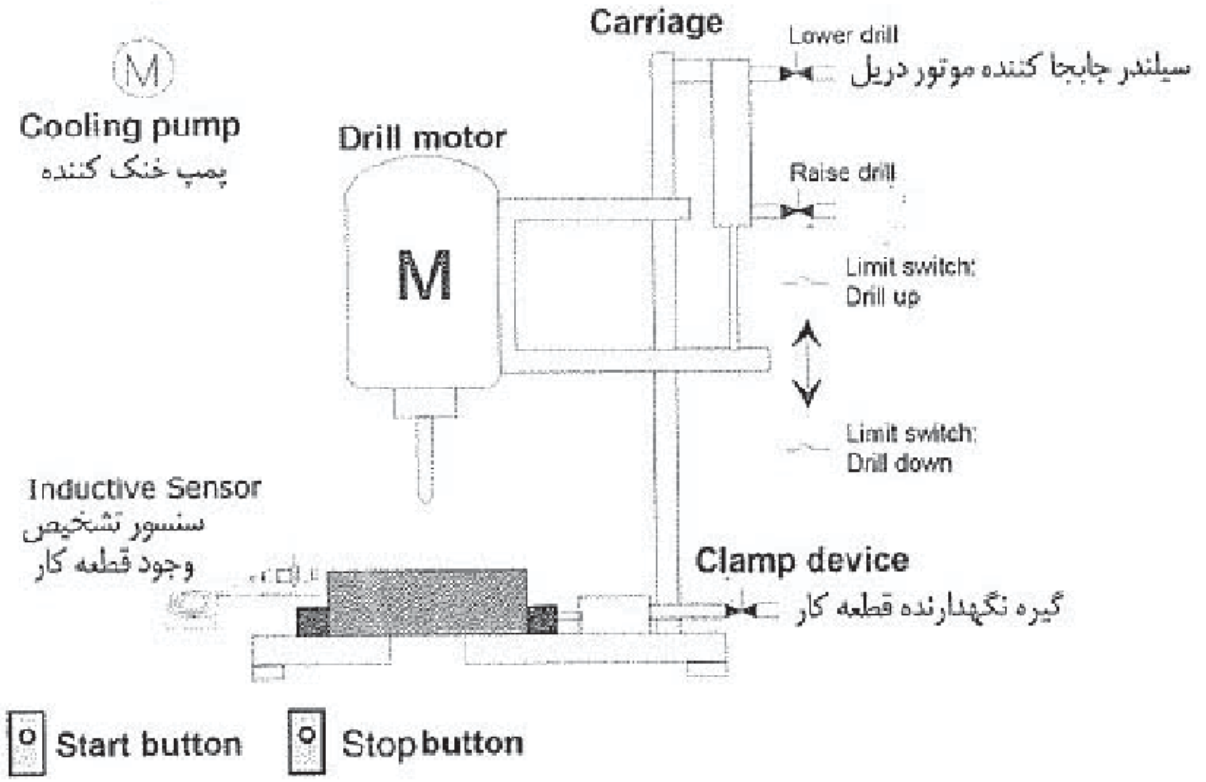
۲۹-9عنوان پروژه : سیستم کنترل دریل عمودی (Vertical Drill)

شرح پروژه : همان طوری که در شکل صفحه بعد مشاهده می کنید ، هدف ، کنترل دستگاه دریل می باشد که پس از قرار گیری قطعه کار روی آن ، به صورت خودکار عمل سوراخ کاری را انجام داده و پس از پایان کار قطعه را خارج و قطعه بعدی را جایگزین می کند.

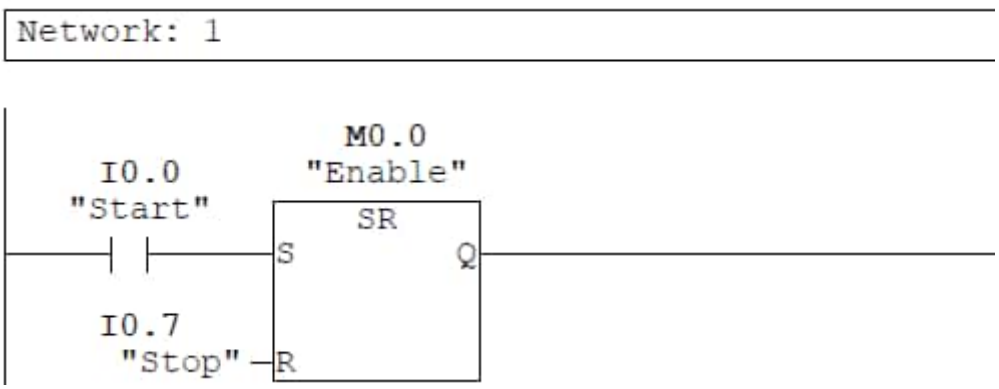
با فشردن کلید Start ابتدا چک شود که حتماً موتور دریل در بالا قرار دارند که این اهرم توسط کلید های Drill up و Drill Down صورت می گیرد ، یعنی باید سنسور Drill up فعال و دیگری غیر فعال باشد یعنی موتور بالای کار قرار دارد. پس توسط سنسور تشخیص دهد که آیا قطعه کار در جای خود قرا ر گرفته است یا نه ، در صورت فعال بودن این سنسور ، فک گیره نگهدارنده بسته شده و قطعه را محکم نگه دارد . ۵ ثانیه پس از بسته شدن فک ، موتور دریل به کمک سیلندر پنوماتیک به سمت پایین حرکت کند که این عمل با فعال کردن شیر برقی Lower drill و غیر فعال کردن Raise drill صورت می گیرد. در این حالت باپایین آمدن موتور ، مته در حال چرخش قطعه کار را سوراخ می کند تا زمانی که کلید Drill Down فعال شود یعنی مته تا انتهای کورس خود وارد قطعه شده ۳ ثانیه پس از فعال شدن این سنسور ،سیلندرپنوماتیک باتعویض وضعیت شیر برقی موتوررابه سمت بالاهدایت کندپس ازرسیدن موتور به بالاترین نقطه و فعال شدن کلید Drill up ، فک گیره باز شده و قطعه کار توسط کارگر تعویض شود و این عمل مرتب تکرار تکرار گردد .

پس از ۱۰ بار عمل سوارخ کاری ، موتور پمپ خنک کننده به مدت ۳۰ ثانیه مته را توسط پمپاژ آب صابون خنک کند . با فشردن کلید Stop در هر مرحله دستگاه متوقف شود .

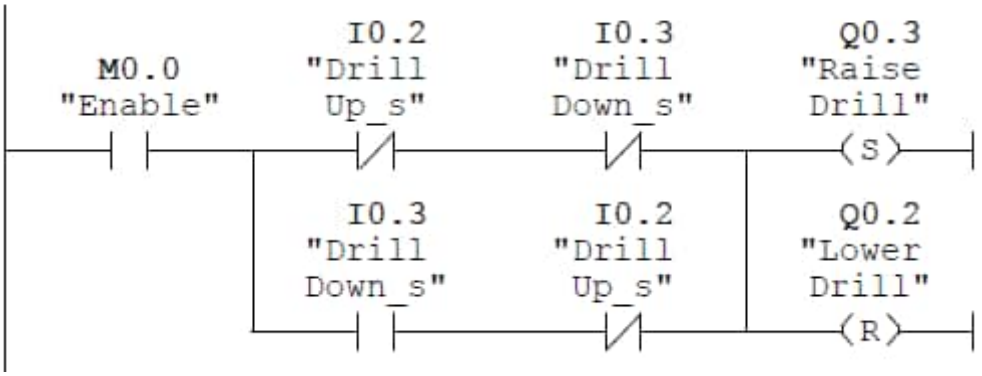
شرایط ایمنی : در صورتی که پس از فشردن کلید Start هیچ یک از کلید های Drill up و Drill Down فعال نبودند یعنی دستگاه موتور دریل در وسط کار قرار داشت ، ابتدا توسط سلندر پنوماتیک ، در بالای کار قرار گیرد ، سپس پروسه شروع به کار کند . همچنین اگر در حین پروسه کلید Stop فشرده شد ، موتور دریل خاموش شده و به کمک سیلندر به سمت بالا حرکت کند!



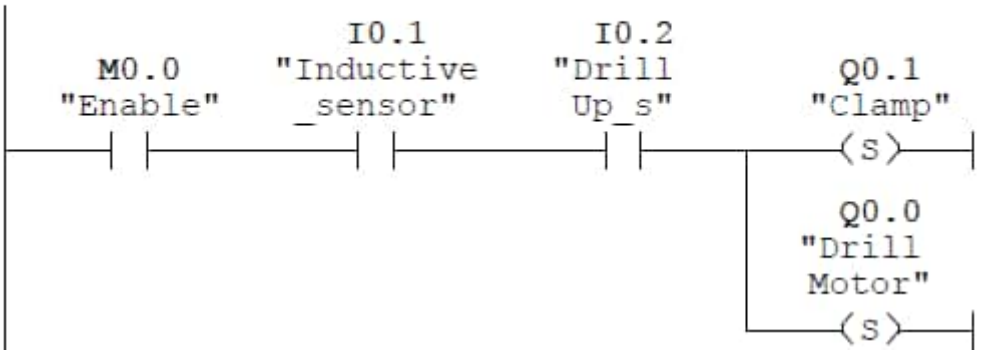
راه حل پروژه :



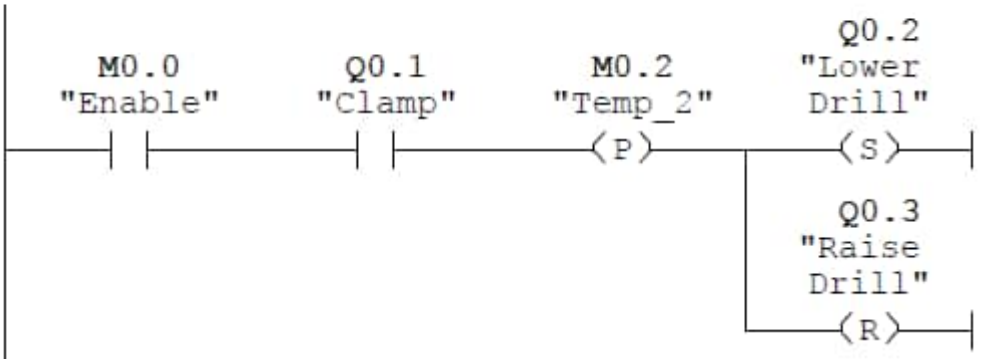
Network: 2



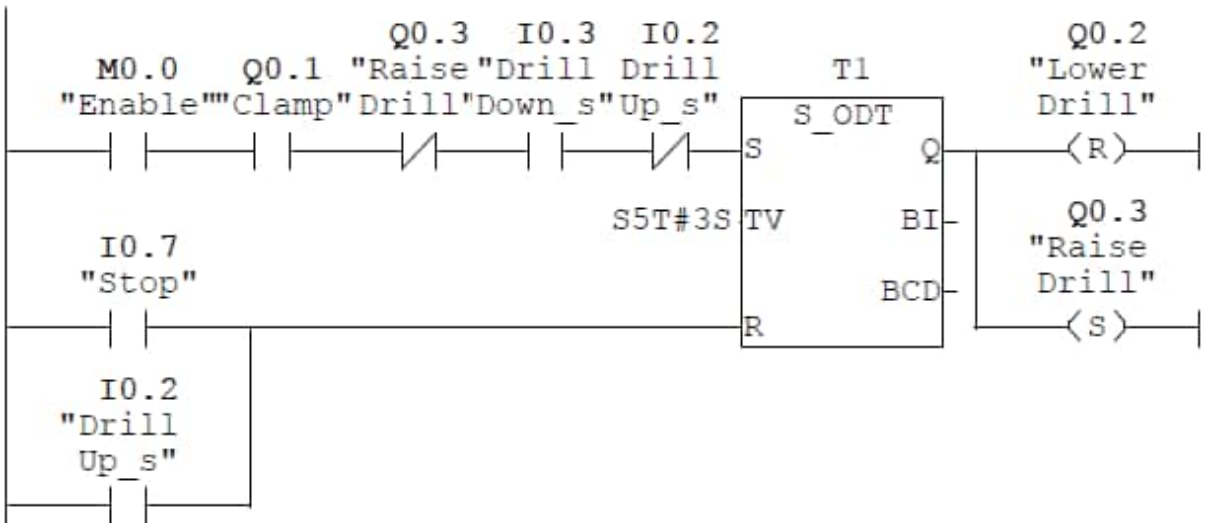
Network: 3



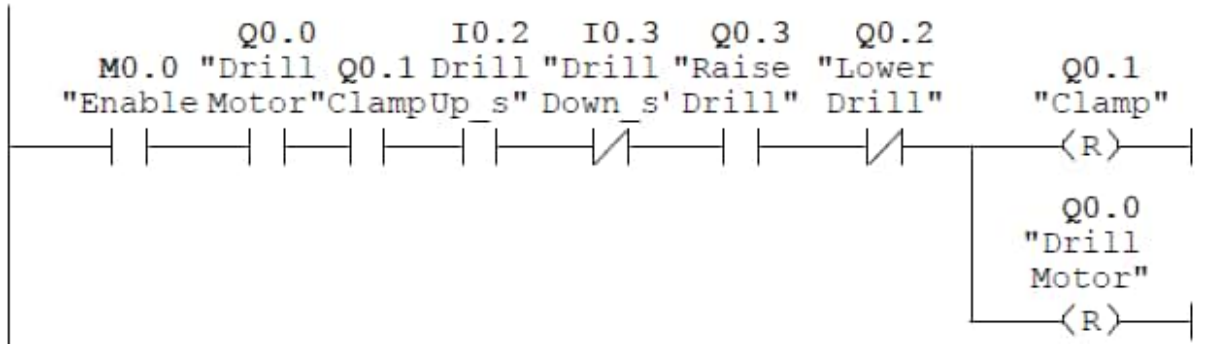
Network: 4

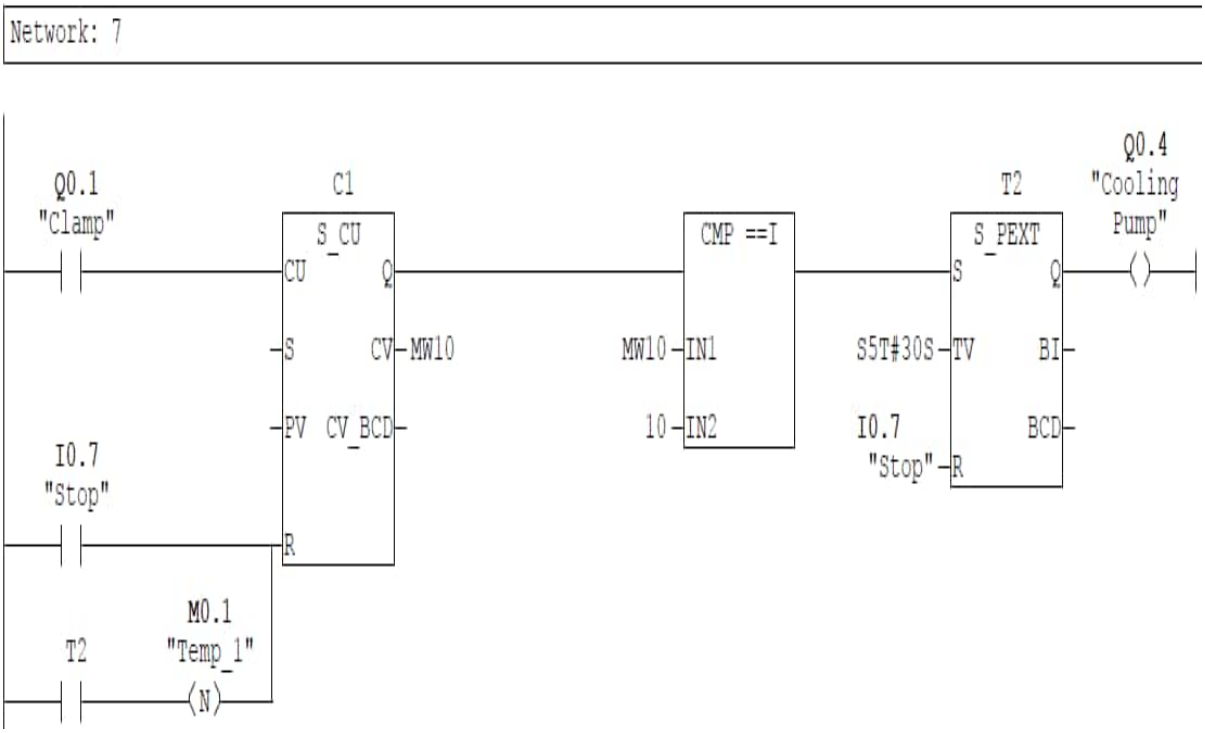


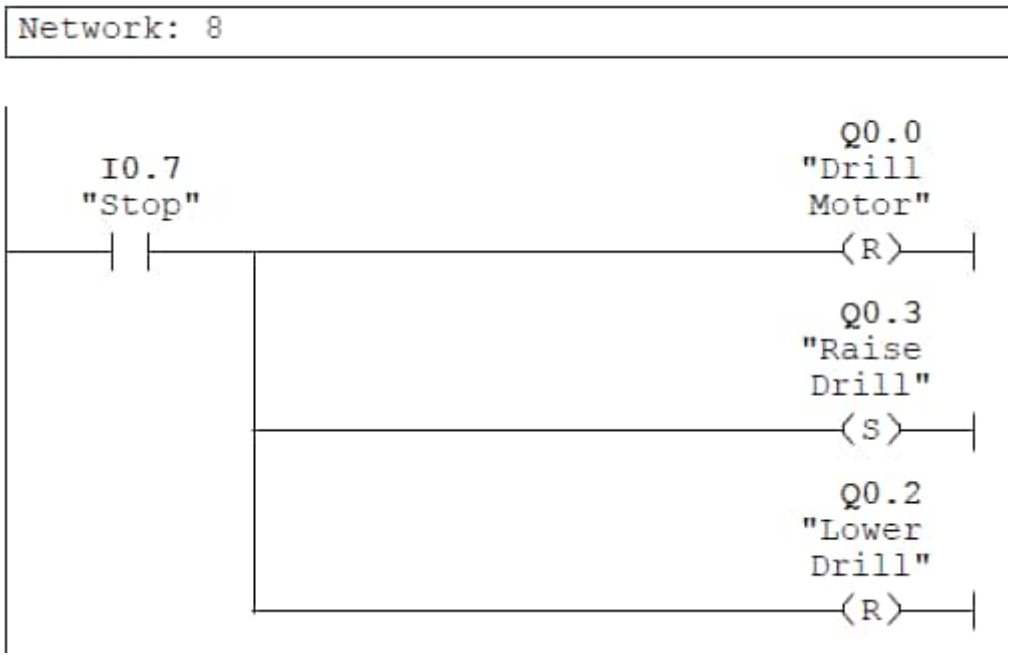
Network: 5



Network: 6



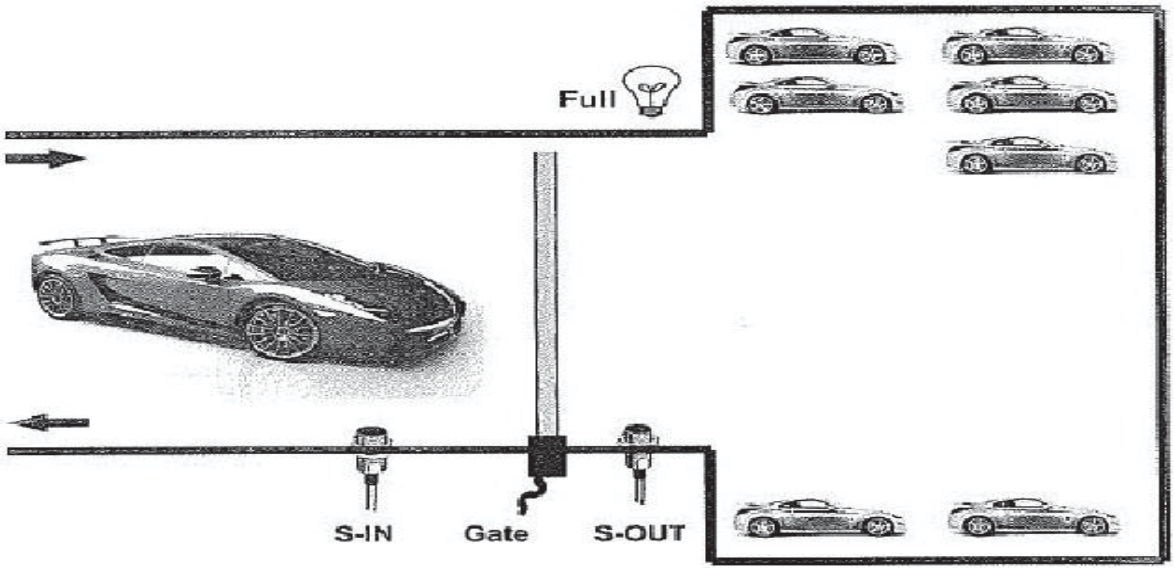




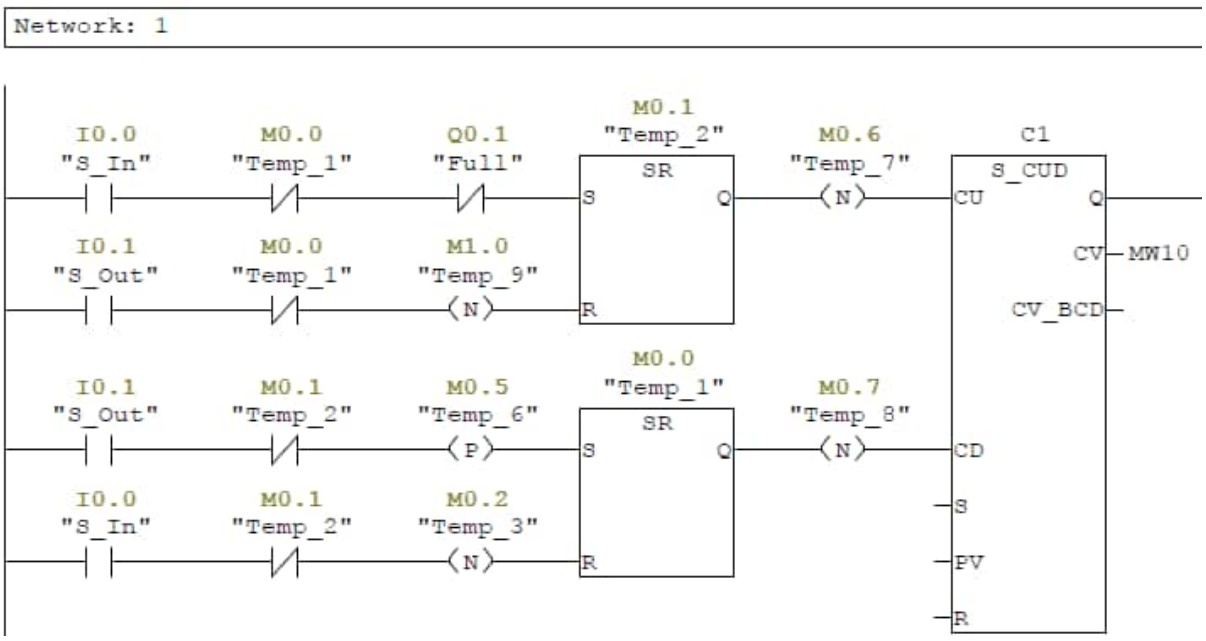
۳۰-9عنوان پروژه : سیستم جامع کنترل پارکینگ خودرو

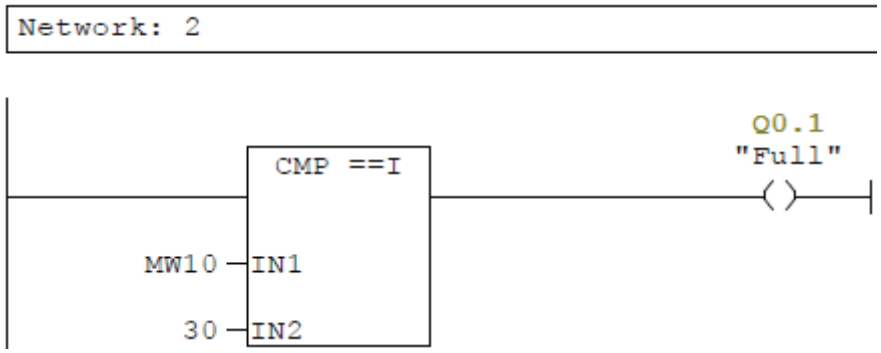
شرح پروژه : مطابق شکل زیر ، با نزدیک شدن خودرو به درب ورودی پارکینگ سنسور نوری S-In تشخیص داده و درب Gate باز شده و ماشین وارد می شود ، و پس از آنکه ماشین از روبروی سنسور نوری S-out عبور کرد ، ابتدا درب ورودی بسته شود و سپس شمارنده ای یک واحد به خودروهای ورودی اضافه کند . حداکثر ظرفیت پارکینگ ۳۰ خودرو می

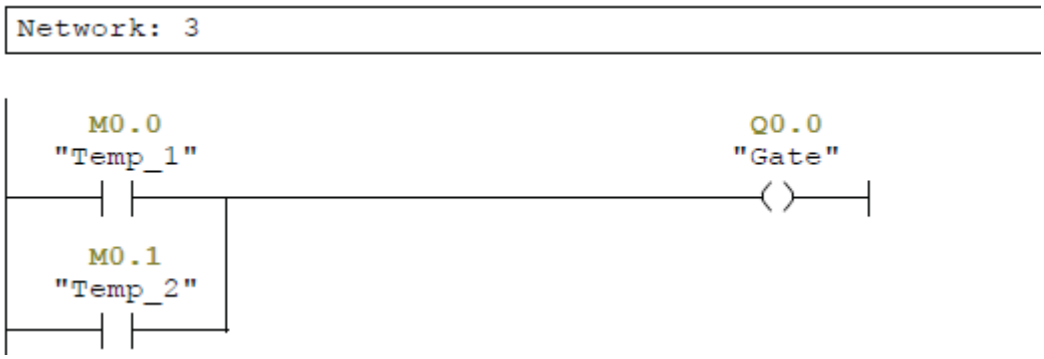
باشد.اگر تعداد خودرو های ورودی به ۳۰ رسید دیگر درب برای ورودی خودرو باز نشود . حال اگر خودرویی از پارکینگ خارج شود با رسیدن به سنسور S-out درب Gate باز شده و پس از عبور از سنسور S-In درب بسته و یک واحد از تعداد C1 کم گردد یعنی اگر به عنوان مثال ۲۵ خودرو وارد پارکینگ شده و ۷ خودرو خارج شود ،شمارنده عدد ۱۸ را نشان داده و اجازه ورود ۱۲ خودرو را به پارکینگ می دهد. در صورتی که پارکینگ پر شود ، لامپ Full روشن شود.



راه حل پروژه :







تشریح پروژه : حل این پروژه نیاز به درک مناسبی از عملکرد تیغه های N.O,N.C و توابع تشخیص لبه –(P) و-(N)- را دارد. در Network1 با رسیدن خودرو به مرز دید سنسور نوری S-In به شرطی که لامپ Full روشن نباشد یعنی پارکینگ پر نباشد ، خروجی Temp-2 را Set می کند که این اتفاق در Network3 باعث باز شدن درب ورودی Gata می شود. خروجی که وارد پارکینگ شده و از سنسور نوری S-out را تشخیص داد باعث Reset ، Temp-2 می شود که متناظر با آن Gata بسته می شود. این اتفاق خود باعث تحریک ورودی CU شمارنده C1 می شود و یک واحد به مقدار MW10 یعنی ظرفیت پارکینگ می شود . حال اگر خودرویی بخواهد از پارکینگ خارج شود با رسیدن به سنسور S-out باعث Set شدن خروجی Temp-1 می شود که در Network3 ، Gata را باز می کند و با عبور از دید سنسور S-in این خروجی Reset شده و علاوه بر بسته شدن Gata باعث کم کردن یک واحد از مقدار C1 می کند. در Network2 نیز مقدار جاری C1 با ۳۰ مقایسه می شود که اگر به آن برسد لامپ Full روشن شده و اجازه باز شدن درب ورودی را نمی دهد.

فصل دهم

پروژه های پیشرفته

مشتمل بر:

۱۰-۱اسیلاتور با زمان on و off متغییر به کمک Functon Biok

۱۰-۲کنترل زنگ مدرسه به کمک ساعت داخلی plc

۱۰ -۳ Set و Reset توالی طولانی از خروجی ها

۱۰-۴شمارش تعداد پیچ های تولیدی کارخانه ، در طول روز

۱۰-۵زمان سنج دقیق فعال بودن(Run) ، (CPU) با دقت 10ms

۱۰-۶غیر فعال کردن موقت وقفه ، Cyclic Inerrupt ، ۸ روز پس از Run شدن CPU

۱۰-۷نشان دهنده مجموع زمان روشن بودن موتور

۱۰-۸سیستم کنترل مخزن به کمک سنسور تعیین سطح آنالوگ

۱۰-۹سیستم کنترل سطح مخزن به کمک سنسور تعیین سطح آنالوگ(Scale)

۱۰-۱۰سیستم کامل کنترل فرایند در مخزن

۱۰-۱۱سیستم کامل کنترل فرآیند در مخزن

۱۰-۱۲ سیستم میانگین گیری وزن باسکول در شالیکوبی

۱۰-۱۳راه اندازی نرم سه الکتروپمپ سیستم جامع بوستر پمپ

۱۰-۱۴ کنترل خروجی آنالوگ به صورت Ramp

مقدمه

در این فصل به بررسی ۱۴ پروژه تکنیکی و کاربردی در سطح پیشرفته که شامل مباحث اصلی استفاده از Function Block , Data Block ، مقادیر آنالوگ ، وقفه های سخت افزاری و نرم افزاری ، SFC,SFB و می باشد ، می پردازیم . برای حل پروژه های این فصل نیاز است ، دانشجو نسبت به مباحث مقدماتی شامل کلیه ابزار اصلی ، Libraries و تنظیمات نرم افزاری تسلط کافی را در داشته باشد.

۱۰-۱عنوان پروژه : اسیلاتور با زمان On و Off متغییر به کمک Function Block

شرح پروژه : به کمک FB ، برنامه ای بنویسید که در OB1 بلوکی دیده شود که زمان ON و OFF را بر حسب ثانیه از شما گرفته و خروجی Output را متناوباً روشن و خاموش کند که در واقع اسیلاتور نامتقارن می باشد . FB به گونه ای طراحی شود که بدون تداخل بارها قابل استفاده باشد .

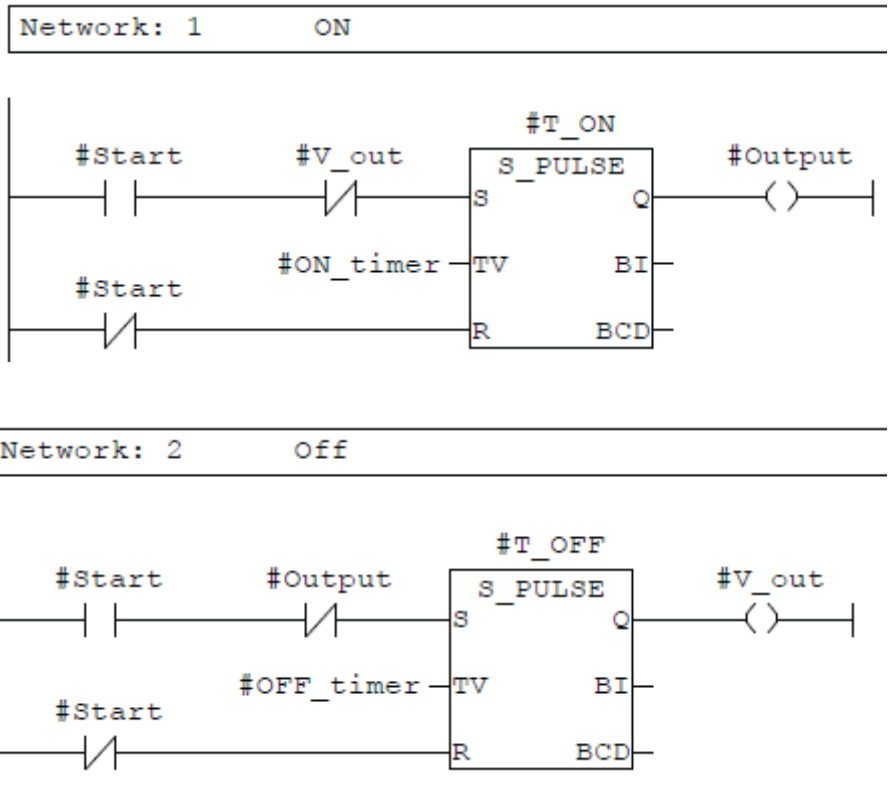
راه حل پروژه : Data Block(DB1)

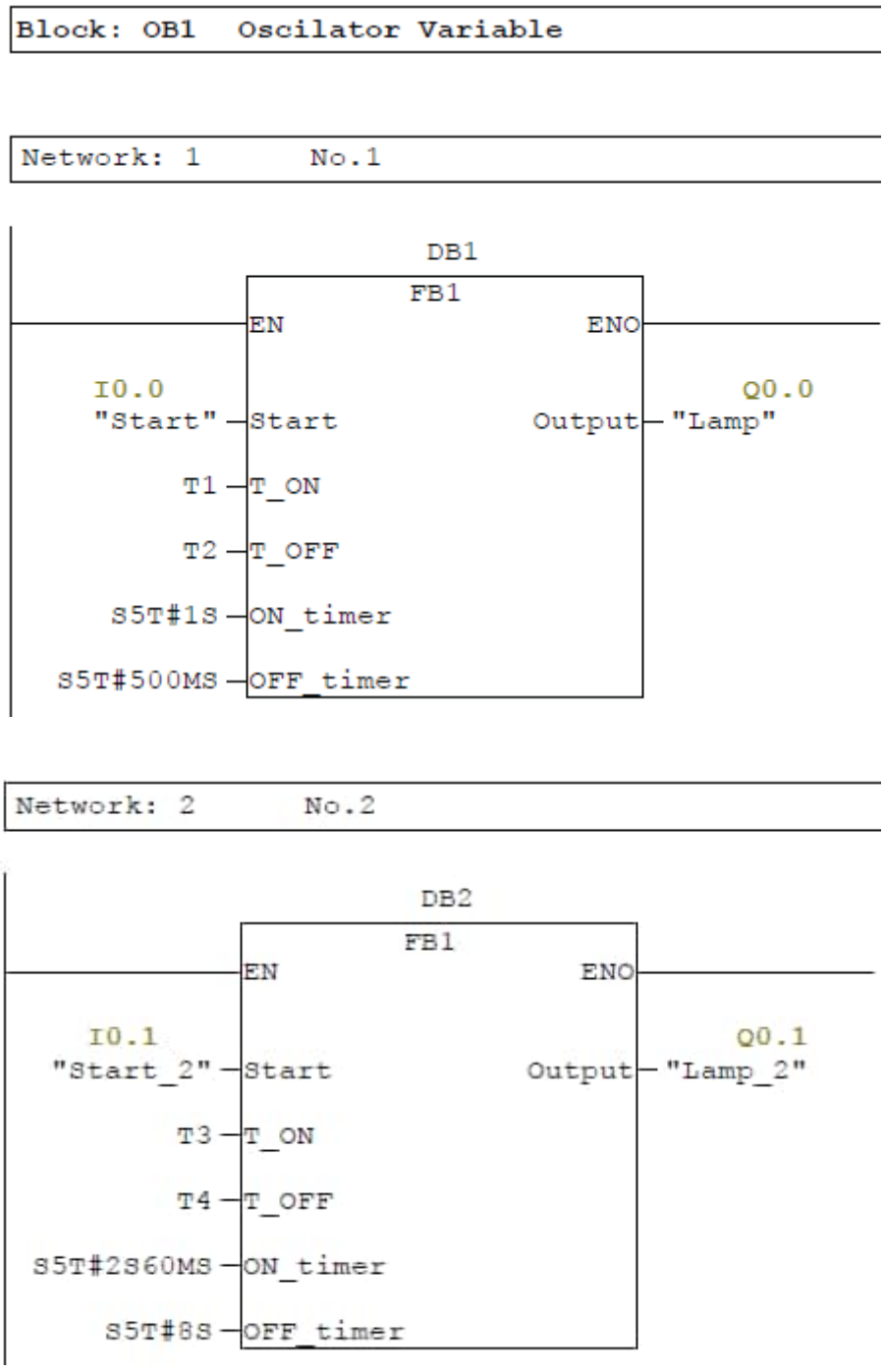
Address	Declaration	Name	Type	Initial value	Actual value
0.0	in	Start	BOOL	FALSE	FALSE
2.0	in	T_ON	TIMER	T 0	T 0
4.0	in	T_OFF	TIMER	T 0	T 0
6.0	in	ON_timer	S5TIME	S5T#0MS	S5T#0MS
8.0	in	OFF_timer	S5TIME	S5T#0MS	S5T#0MS
10.0	out	Output	BOOL	FALSE	FALSE
12.0	stat	V_out	BOOL	FALSE	FALSE

Function Block(FB1)

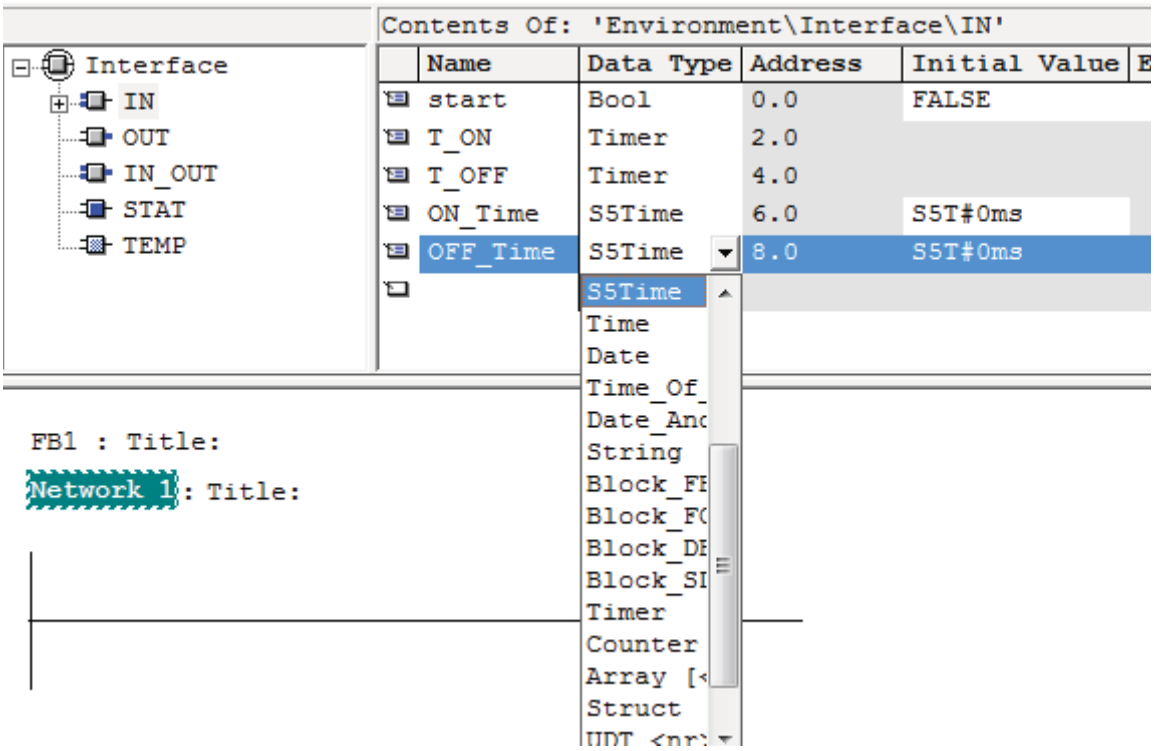
Name	Data Type	Address	Initial Value	Comment
IN		0.0		
Start	Bool	0.0	FALSE	
T_ON	Timer	2.0		
T_OFF	Timer	4.0		
ON_timer	S5Time	6.0	S5T#0MS	
OFF_timer	S5Time	8.0	S5T#0MS	
OUT		0.0		
Output	Bool	10.0	FALSE	
IN_OUT		0.0		
STAT		0.0		
V_out	Bool	12.0	FALSE	
TEMP		0.0		

FB1



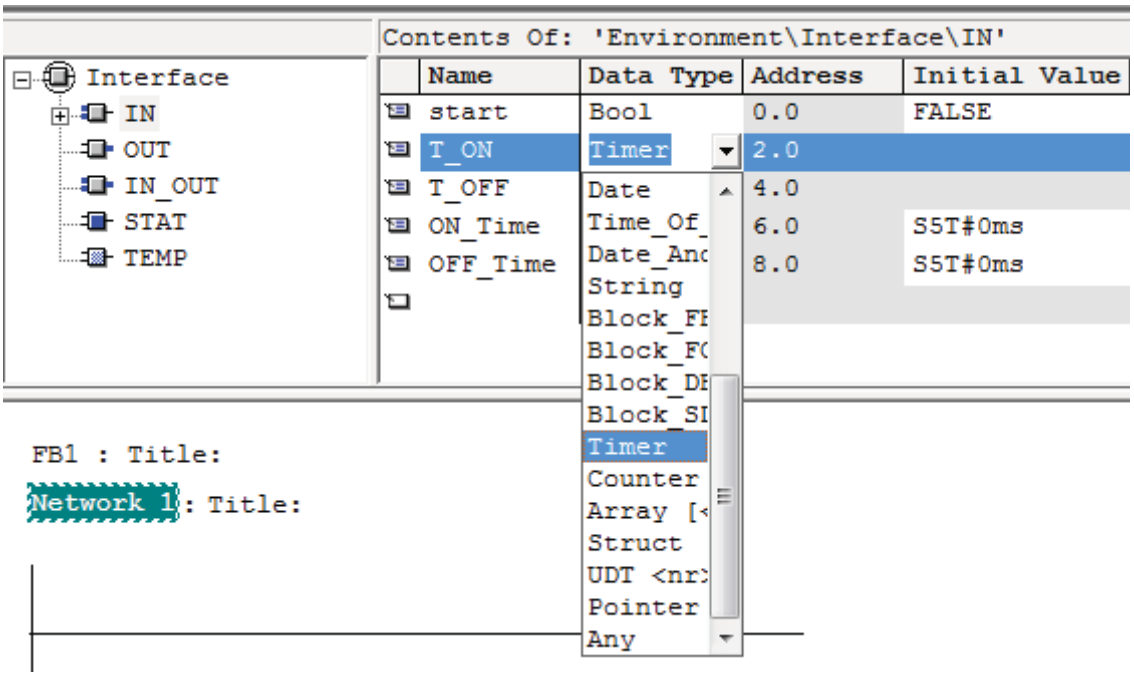


تشریح پروژه : از آنجایی که از ما خواسته شده اسیلاتوری با زمان متغییر طراحی شود پس برنامه چشمک زن ساده را نمی توان استفاده کرد ، بلکه باید آن برنامه را در غالب FB طراحی کنیم ، که زمان TV آن و شماره تایمر Tx متغییر باشد. ابتدا FB1 را ایجاد کرده و مطابق شکل ۱-۹ در قسمت Interface ورودی و خروجی های مورد نظر را ایجاد می کنیم .



در شکل ۲-۹ مشاهده می کنید که متغییری با نام های T-ON و T-OFF ساخته و نوع (Data Type) را Timer انتخاب می کنیم که در واقع شماره تایمر می باشد که می تواند ، TO ، T1 ، T2 و ... انتخاب شود . همچنین ورودی با نام های OFF-Timer و ON-Timer از نوع S5 Timer ایجاد می کنیم که زمان TV تایمرها می باشد . در قسمت Out متغیر با نام Outuput ایجاد می کنیم که خروجی اسیلاتور است . در Stat هم که در واقع نوعی Temp است ولی در DB مربوط به FB ایجاد می شود ، متغییر v-out را برای تایمر Off ایجاد می کنیم . سپس مطابق با پروژه (۱۸-۵) فصل پنجم ، برنامه چشمک زن را طراحی می کنیم ولی با ورودی و خروجی متغیر ، سپس وارد OB1 شده و FB ساخته شده را در برنامه Call می کنیم . همانطوری که در شکل مشاهده می کنیم . FB را دوبار خوانی کرده و با زمان های مختلف مقدار دهی کردیم.

حال اگر شبیه سازی را فعال کنید . مشاهده می کنید با فشردن کلید Start و Start-2 خروجی Lamp و Lamp-2 با زمان های مستقل شروع به روشن و خاموش شدن می کنند.

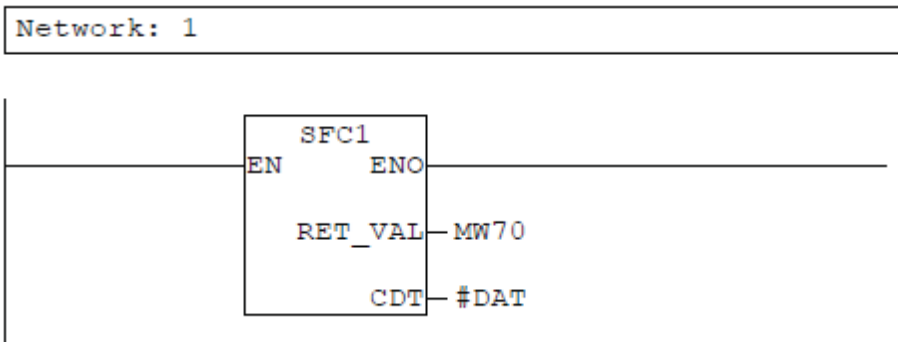


شکل ۲-۹

۲-۱۰عنوان پروژه : کنترل زنگ مدرسه به کمک ساعت داخلی PLC

شرح پروژه : به کمک توابع سیتمی ابتدا ساعت PLC را خوانده و سپس به کمک آن ، برنامه ای بنویسید که روزهای شنبه تا پنج شنبه ، ساعات ۸، ۹:۴۵ ، ۱۰ ، ۱۱:۴۵ ، ۱۲ ، ۱۳:۴۵ زنگ مدرسه به مدت ۳۰ ثانیه به صدا در آید ، که به ترتیب زنگ شروع مدرسه ، زنگ تفریح و زنگ پایان مدرسه است . جهت راهنمایی باید گفته شود که برای حل این پروژه باید ابتدا به کمک تابع SFC1 ساعت ، دقیقه و ایام هفته را از روی واحد RTC[1] ، PLC جدا کنید و سپس برنامه را بنویسید.

راه حل پروژه:



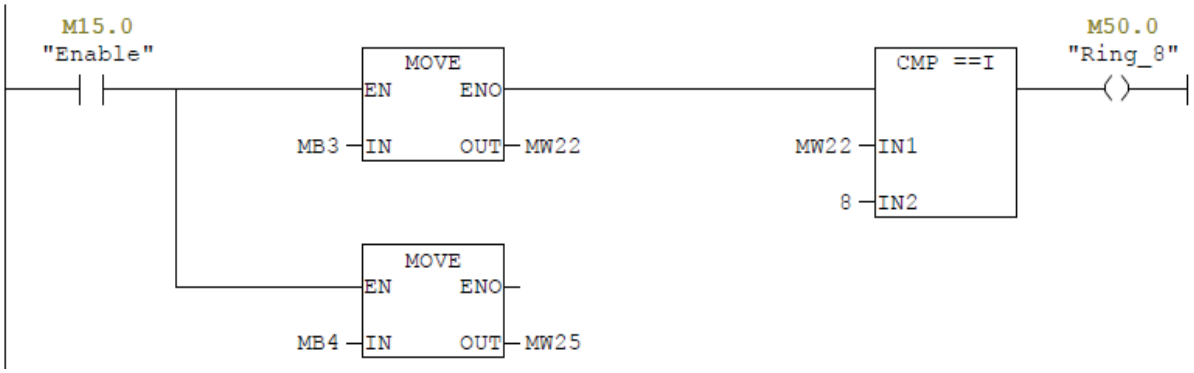
[1] Real Time Clock -

Network: 2 Address Register

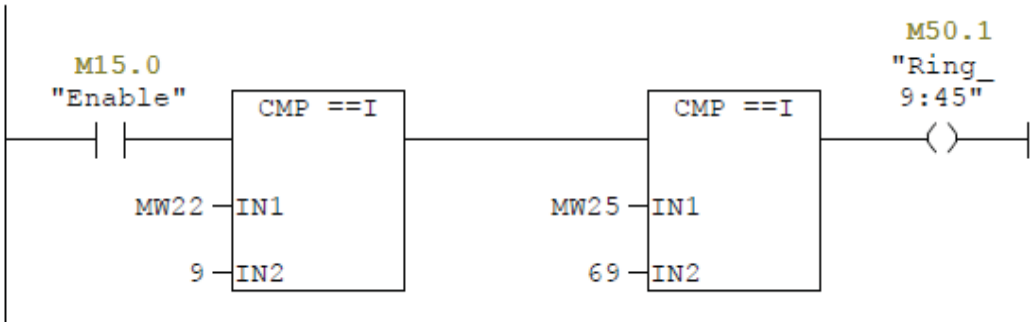
```
LAR1   P##DAT
L      B [AR1,P#0.0]      → سال
T      MB      0
L      B [AR1,P#1.0]      → ماه
T      MB      1
L      B [AR1,P#2.0]      → روز
T      MB      2
L      B [AR1,P#3.0]      → ساعت
T      MB      3
L      B [AR1,P#4.0]      → دقیقه
T      MB      4
L      B [AR1,P#5.0]      → ثانیه
T      MB      5
L      B [AR1,P#7.0]
L      B#16#F
AW
T      MB      6          → روز هفته
```

Network: 3 Enable Ring_Saturday to Thursday

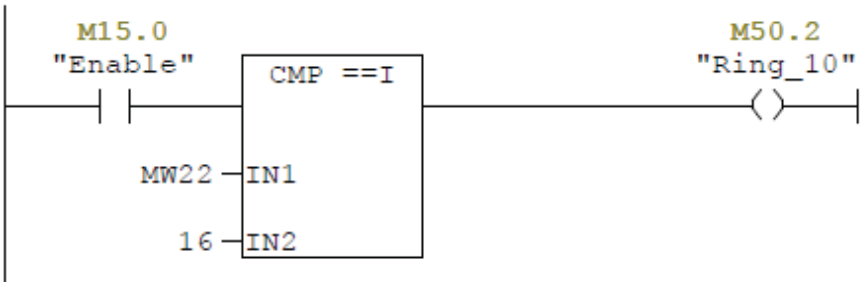
Network: 4



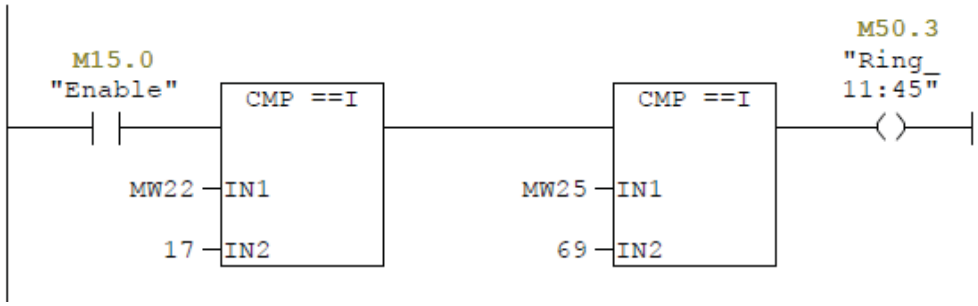
Network: 5



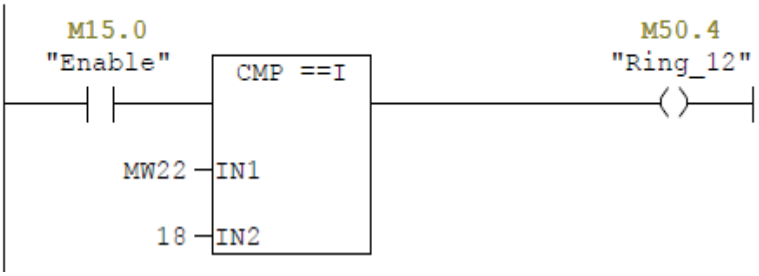
Network: 6



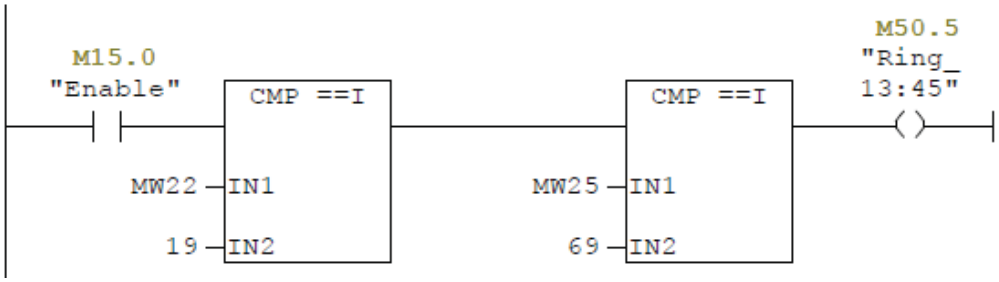
Network: 7



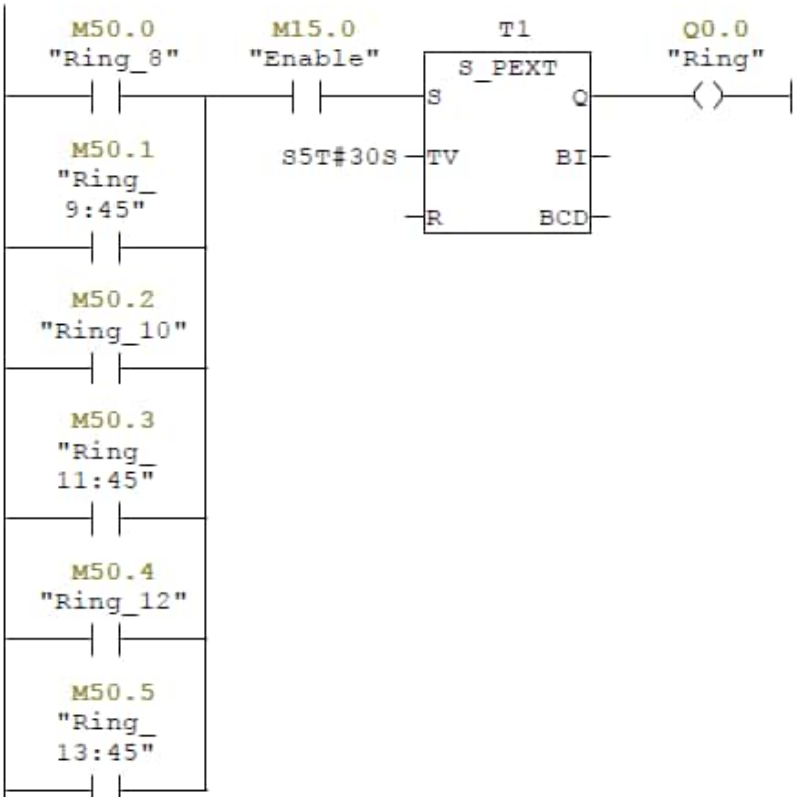
Network: 8



Network: 9



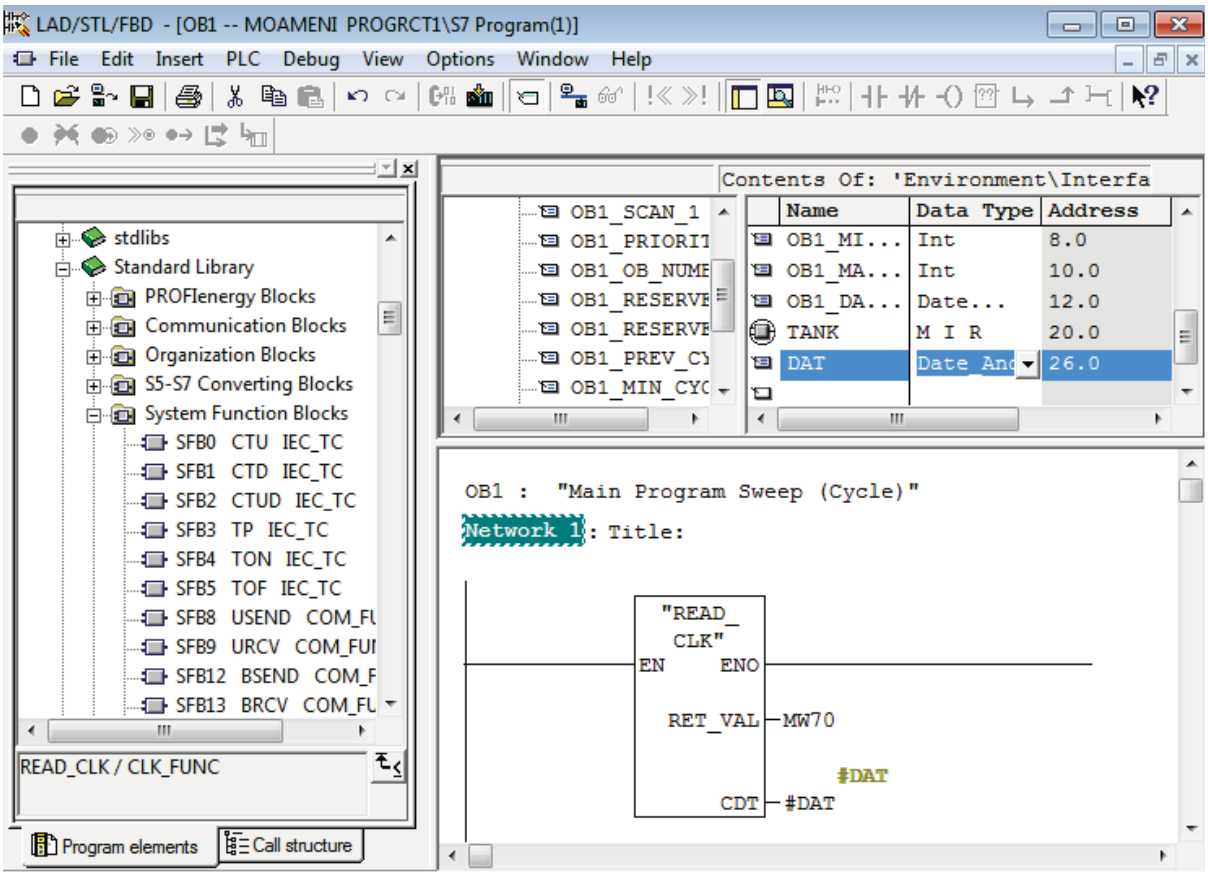
Network: 10



تشریح پروژه : برای حل این پروژه ابتدا مطابق شکل ۳-۶ از مسیر:

Libraries → standaed Librarg→ Syatem Function Block

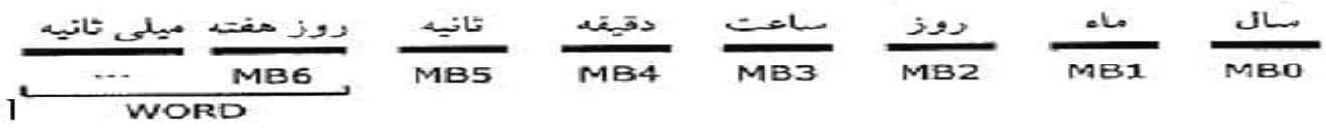
تابعSFC1راوارد Network1 می کنیم. سپس از کادر بالا ، در زیر شاخه Temp → Interface در انتهای آن متغییری با نام دلخواه (DAT) ایجاد کرده و نوع آن را Data_And_Time انتخاب می کنیم . این عمل برای آن است که تابع SFC1 که تاریخ و ساعت PLC را که می خواند ، خروجی را به صورت رشته ای با فرمت فوق به ما می دهد که در هیچ یک از قالب هایی که تاکنون شناخته ایم از قبیل Dword,Word ,Byte نیست ، پس باید متغییری با فرمت فوق ایجاد کنیم تا مقدار جاری زمان در آن قرار دهیم . حال نوبت به آن می رسد که از رشته فوق ، ساعت ، دقیقه و ایام هفته را جدا کرده و در متغییرهایی از جنس Memory Byte(MB) ذخیره کنیم.



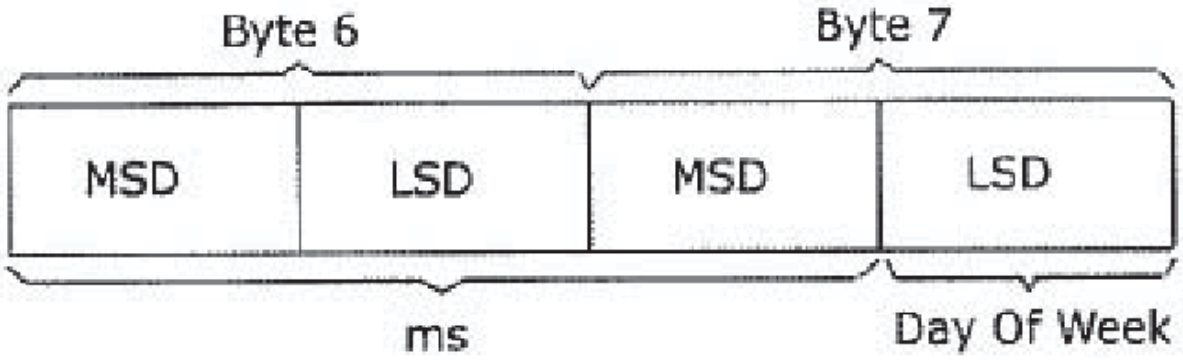
برای درک بهتر خواننده محترم ابتدا در مورد محتویات این رشته و پس تجزیه آن به کمک دستورات STL می پردازیم.

فرمت Date_And_Time به صورت 64bit,BCD معادل 8 Byet فضا را اشغال می کند .حال به کمک دستوراتی به زبان STL ، توسط Address Register با بیت های مختلف آن که معرف سال ، ماه ، روز ، ساعت ، دقیقه ، ثانیه است را جدا و در متغییرهای MB ذخیره می کنیم.

```
Network: 2        Address Register
      LAR1   P##DAT
      L      B [AR1,P#0.0]      → سال
      T      MB      0
      L      B [AR1,P#1.0]      → ماه
      T      MB      1
      L      B [AR1,P#2.0]      → روز
      T      MB      2
      L      B [AR1,P#3.0]      → ساعت
      T      MB      3
      L      B [AR1,P#4.0]      → دقیقه
      T      MB      4
      L      B [AR1,P#5.0]      → ثانیه
      T      MB      5
      L      B [AR1,P#7.0]
      L      B#16#F
      AW
      T      MB      6          → روز هفته
```



در انتهای دستور مشاهده می کنید که بایت هفتم از رشته با مقدار F(۱۱۱) ، AND شده و در نتیجه آن را در MB6 قرار گرفته است که معرف روز هفته (یکشنبه =۱ ، دوشنبه =۲ .... شنبه =۷)است. علت این امر است که بایت ۶و ۷ با هم مقدار میلی ثانیه و ایام هفته را گزارش می دهند که ازاین 16bit ، 4bit مربوط به روز هفته و 12bit مربوط به میلی ثانیه است که برای درک بهتر به شکل زیر توجه کنید:



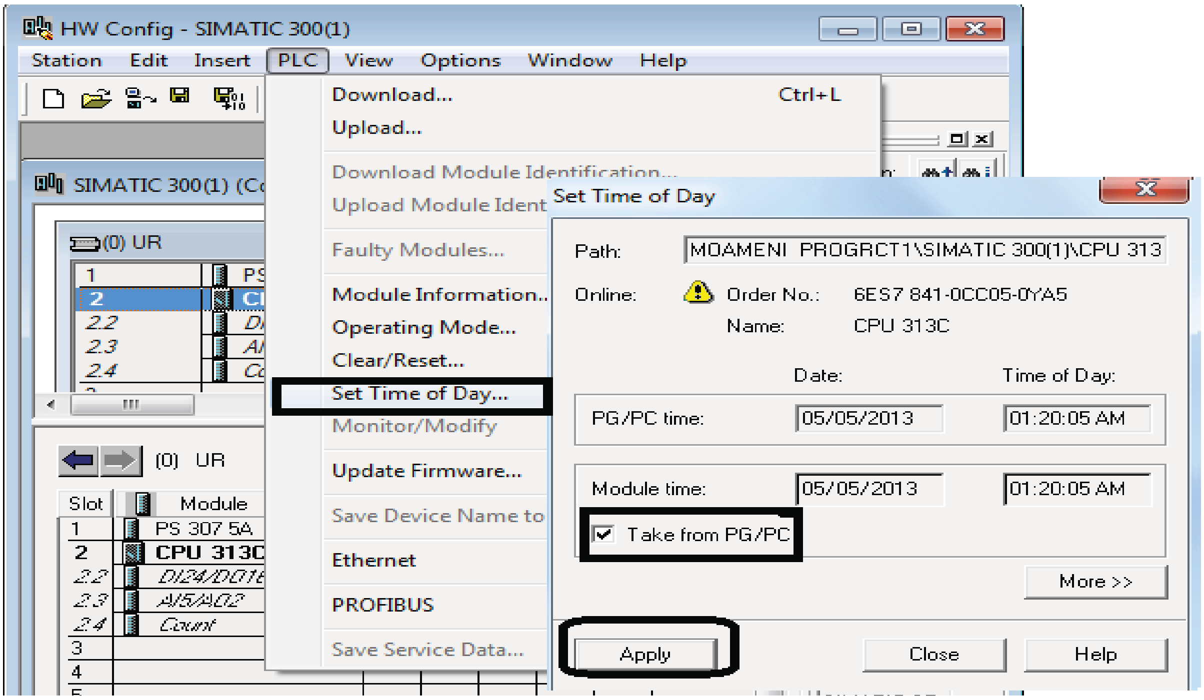
برای حل پروژه فوق ما تنها به ساعت MB3 ، دقیقه MB4و روز هفته MB6 نیاز داریم. نکته بسیار مهم آن است که مقدار قرار گرفته در هر یک از MB فوق با فرمت Hex می باشد ، در حالی که دستورات مقایسه مقادیر را با فرمت Decmal پردازش می کنند ، به عنوان مثالاگر عدد ۱۳ را در مقایسه گر قرار دهیم ، مقایسه گر در زمان ۱۹ خروجی را فعال می کند زیرا معادل Hex به ۱۲ Decimal ، عدد ۱۹ می باشد! پس باید ، ابتدا هر یک از اعداد فوق را که شامل ساعت و دقیقه می باشند را به فرمت Decimal تبدیل کرد و سپس در برنامه استفاده کرد. در جدول صفحه بعد تبدیلات مربوط به پروژه را مشاهده می کنند:

ساعت و دقیقه مورد نظر با فرمت Decimal	معادل HEX
8	8
9	9
45	69
10	16
11	17
12	18
13	19

در Network3 ابتدا MB6 را در MW20 کپی کردیم که علت آن این است که دستورات مقایسه فرمت Byte را قبول نمی کنند و ما با دستور Move محتویات آن را در MW دلخواه قرار دادیم.

از انجایی که در مساله ذکر شد که روزهای شنبه تا پنج شنبه سیستم عمل کند ، و از آنجایی که روز جمعه در MB6 که روز هفته را مشخص می کند معادل عدد ۶ است ، با قرار دادن دستور مقایسه نابرابری با آن در همه روزها به غیر از جمعه سیستم فعال است که با فعال شدن بیت Enable (M15.0) مابقی Network ها فعال می شوند . در Network4 ابتدا مقدار MB3 که معرف ساعت سیستم است و MD4 که دقیقه سیستم را در خود دارد به ترتیب در MW22 و MW25 ذخیره می شوند ، پس به کمک دستور مقایسه Cmp==1مقدار ساعت با عدد ۸ مقایسه می شود ، در صورت برابری بیت Ring__8 را فعال می کند . در Network5 ، در صورتی که ساعت ۹ و دقیقه برابر ۴۵ باشد ، بیت Ring__9:45 فعال می شود. در Network6 اگر ساعت برابر ۱۰ باشد بیت Ring__10 فعال می شود ، Network7 تا ۹ نیز در ساعات ۱۱:۴۵, ۱۲ و ۱۳:۴۵ بیت های مربوط به خود را فعال می کنند . درNetwork10 همان طوری که مشاهده می کنید همه بیت های خروجی مقایسه با آدرس M50.0 تا M50.5 با یکدیگر OR شده و به تایمر S-PEXTبا زمان ۳۰ ثانیه متصل شده اند که همان Ring یا زنگ مدرسه می باشد که خروجی آن وصل شده است و به صدا در می آید.

-در شکل زیر طریقه تنظیم ساعت داخلی PLC را مشاهده می کنید:



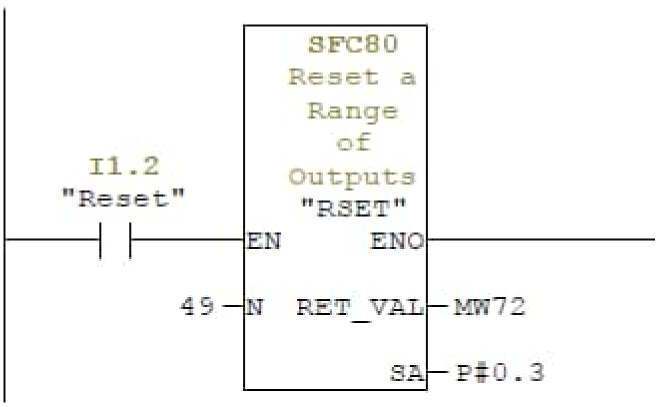
۳-۱۰عنوان پروژه : Set و Reset توالی طولانی از خروجی ها

شرح پروژه : برنامه ای بنویسید که به کمک وقفه سخت افزاری (Hardware Interrupt) و لبه پایین رونده ورودی I ۰.۲ از آدرس Q0.3 تا Q6.2 یعنی 49bit از خروجی به یکباره ، set شوند و با فشردن شستی I1.2 همگی Reset گردند. از آنجایی که تعداد خروجی ها زیاد می باشد ، نمی توانند از دستورات ساده Setو Reset به صورت تکی استفاده کنید!

راه حل پروژه :

Block: OB1 49 Outputs Control

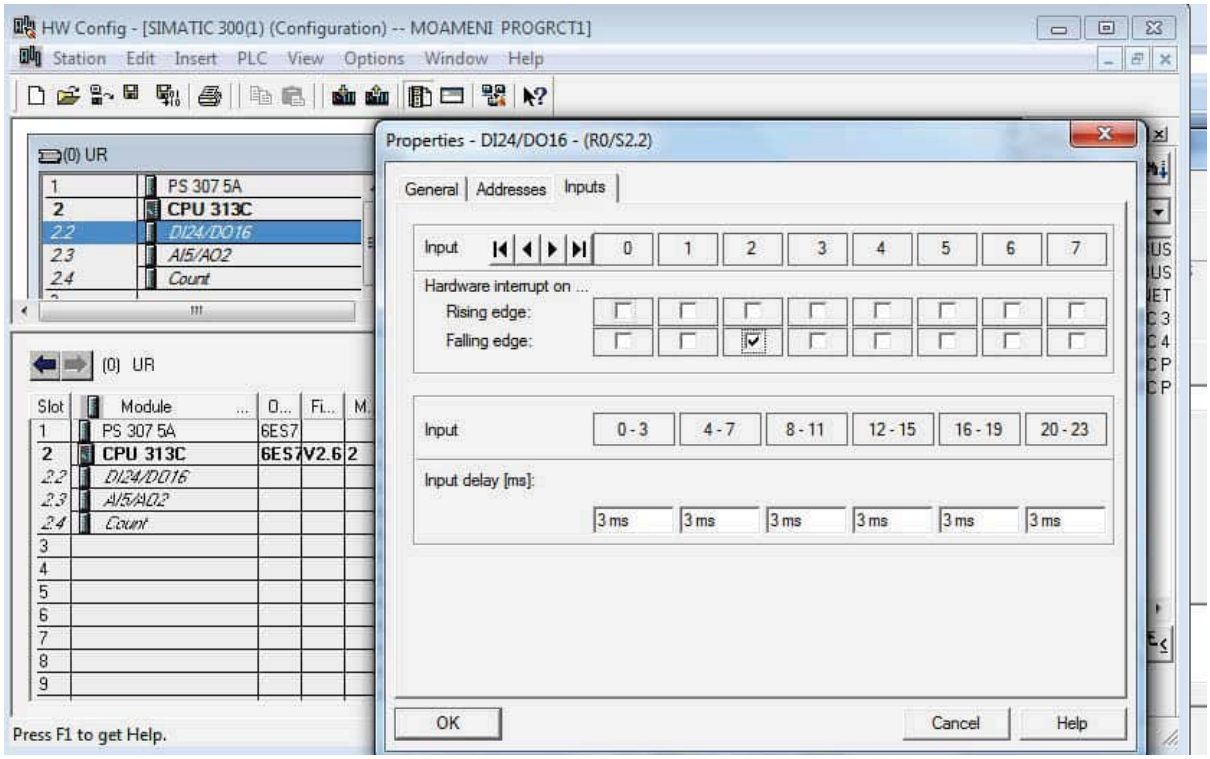
Network: 1



```
Block: OB40    "Hardware Interrupt"

Network: 1

      I1.2
    "Reset"          "SET"
                  EN      ENO
           49 — N   RET_VAL — MW70
                  SA — P#0.3
```

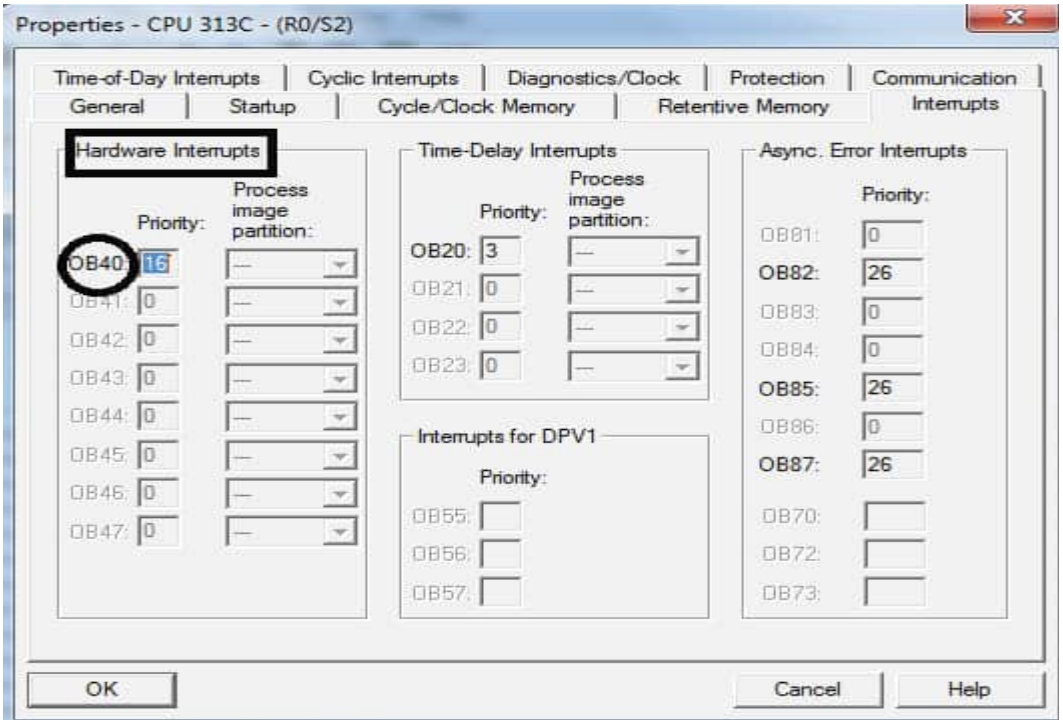
تشریح پروژه : از انجایی که در پروژه از ما خواسته شده با فعال شدن ورودی از نوع وقفه سخت افزاری پروسه انجام شود ، ابتدا مطابق شکل ۴-۶ از مسیر:

Hardware Config → DI24/DO16→Inputs

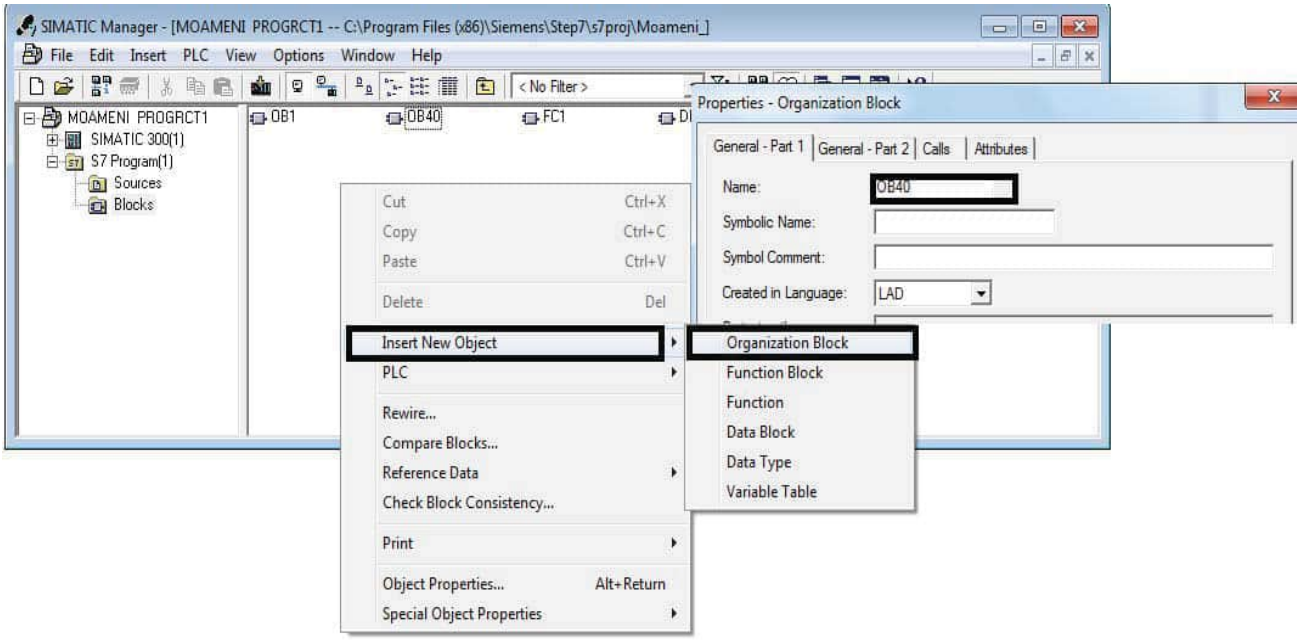


ورودی را انتخاب کرده و در آدرس آن را ۲ که معرف I0.2 می باشد تیک Hardware Intervupt Falling Edge را به منظور فعال شدن وقفه سخت افزاری در لبه پایین رونده می زنیم .

سپس مطابق شکل زیر آدرس OB وقفه را شناسایی می کنیم که در اینجا OB40 می باشد ، سپس در محیط Block ، مطابق شکل ۶-۶ بلوک OB ایجاد کرده و نام آنرا OB40 می گذاریم.



در Network1 ، OB40 همان طور که راه حل پروژه مشاهده می کنید از تابع SFC79 که بلوکی مخصوص این کار است استفاده شده است.



این بلوک در صورت فعال شدن ورودی En به تعدادی که در ورودی N آن تعریف شده است از آدرس (Start SA(Start Address)خروجی واقعی را که در اینجا P#0.3 که معرف Q0.3 می باشد به تعداد 49, Set می کند. اگر در انجام پروسه اشکالی پیش آید کد Hex معادل ان در MW70 قرار می گیرد ، در غیر اینصورت MW70 برابر صفر می باشد. زمانی که لبه پایین رونده I0.2توسط کارت ورودی تشخیص داده شده و OB40 فرا خوانی شده و در صورت غیر فعال بودن کلید Reset خروجی Q0.3 تا Q6.2 , set می شوند. در OB1 نیز در صورت فعال شدن Reset ، تابع SFC80 که مانند SFC79 می باشد با این تفاوت که همان پارامتر های ورودی و خروجی ، خروجی های تعریف شده را Reset می کند یعنی به تعداد 49bit از آدرس شروع Q0.3 هر وضعیتی داشته باشد به منطق صفر تغییرو وضعیت می دهند . این پروژه را به کمک دستور Move نیز می توانید طراحی کنید.

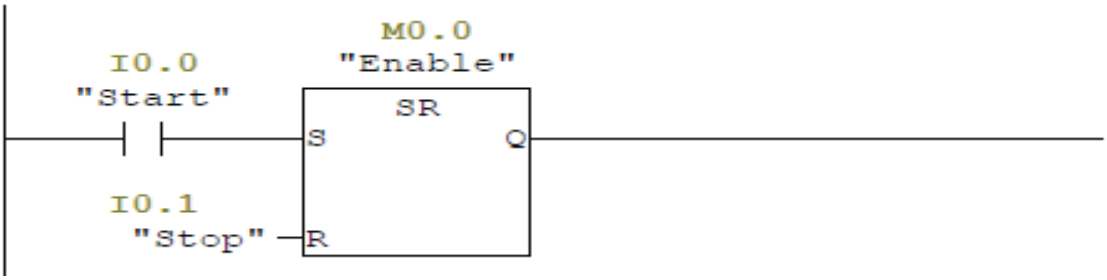
۴-۱۰عنوان پروژه : شمارش تعداد پیچ های تولیدی کارخانه ، در طول روز

شرح پروژه : به کمک شمارنده سیستمی SFB برنامه ای بنویسید که تعداد پیچ های عبوری از مقابل سنسور نوری را شمارش کرده و در DB ذخیره کند با این قابلیت که شمارش از ساعت 8:00 صبح شروع شده و تا 8:00 صبح فردا ادامه یابد. برای روز بعد شمارنده Reset شده و مقدار شمارش شده جاری را در آدرس دیگری ذخیره کند. پس از یک هفته ، در صورتی که در هر یک از روزها ، تعداد تولیدی پیچ کمتر از ۳۰۰ عدد بود ، آن روز را مشخص کرده و عدد متناظر با آن را با پیش فرض معادل یکشنبه و ۷ معادل شنبه در MW70 قرار دهد . پس دقت شود که حداقل تولید محصول در طول ۲۴ ساعت ، ۳۰۰۰ عدد می باشد. برای اندازه گیری زمان ۲۴ ساعت و شروع کار از ۸ صبح از وقفه Time of Day استفاده کنید و تاریخ شروع را از ۰۸/۱۲/۲۰۱۰ فرض کنید.

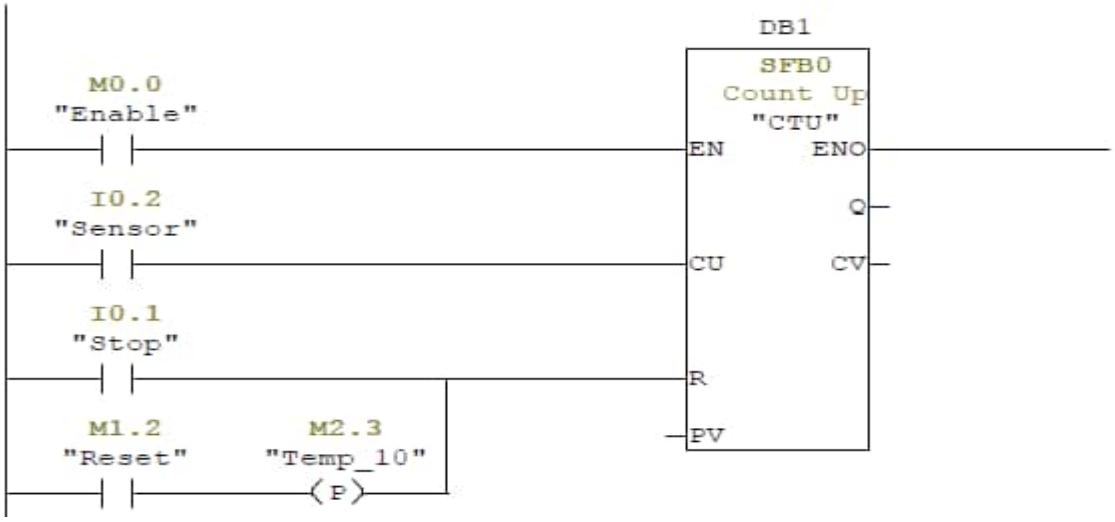
راه حل پروژه : (DB1)

Address	Name	Type	Initial value
0.0		STRUCT	
+0.0	Sample_1	WORD	W#16#0
+2.0	Sample_2	WORD	W#16#0
+4.0	Sample_3	WORD	W#16#0
+6.0	Sample_4	WORD	W#16#0
+8.0	Sample_5	WORD	W#16#0
+10.0	Sample_6	WORD	W#16#0
+12.0	Sample_7	WORD	W#16#0
+14.0	result	WORD	W#16#0
=16.0		END_STRUCT	

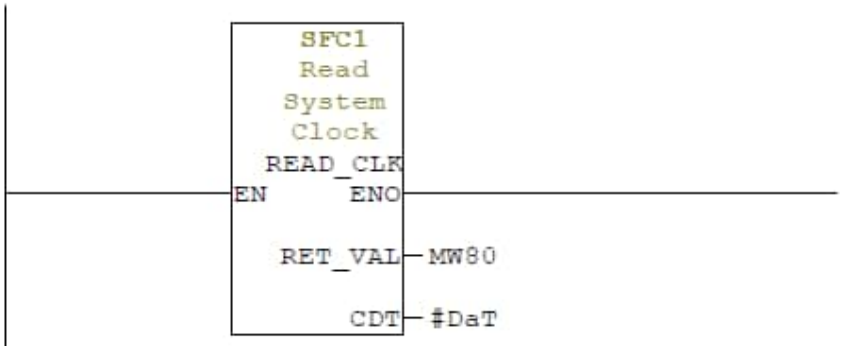
Network: 1 OB1



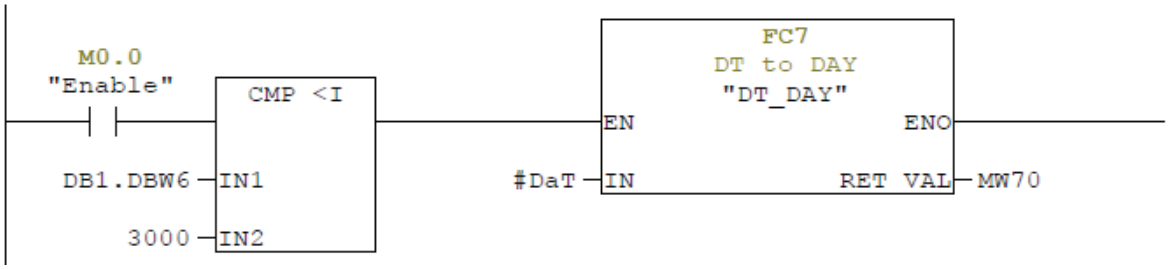
Network: 2



Network: 3



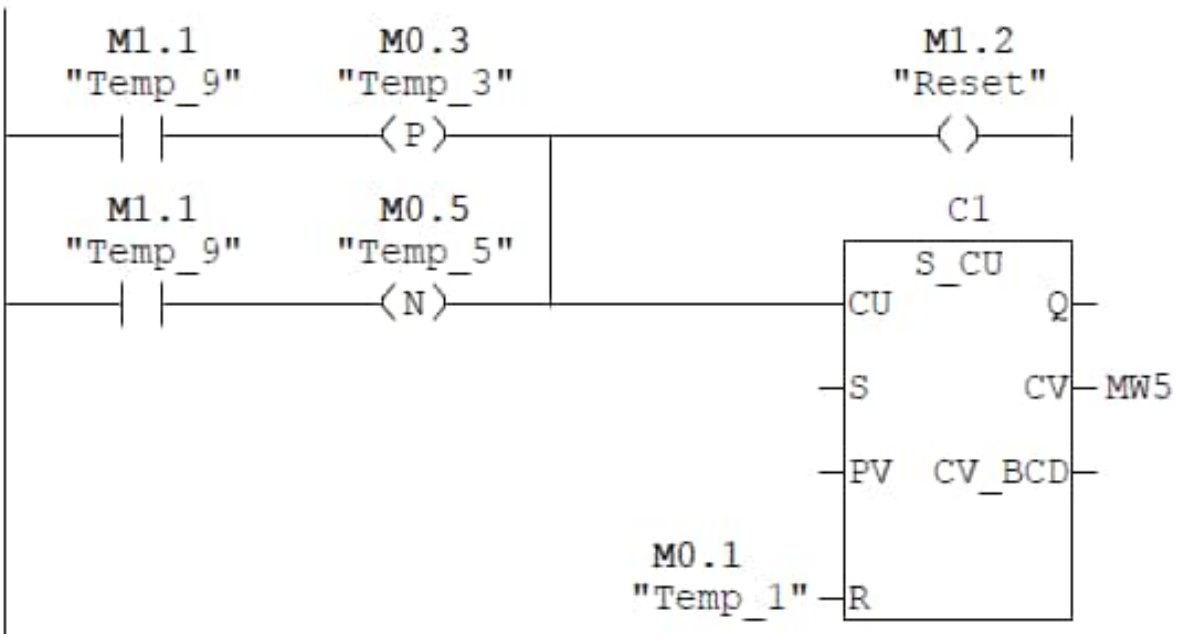
Network: 4



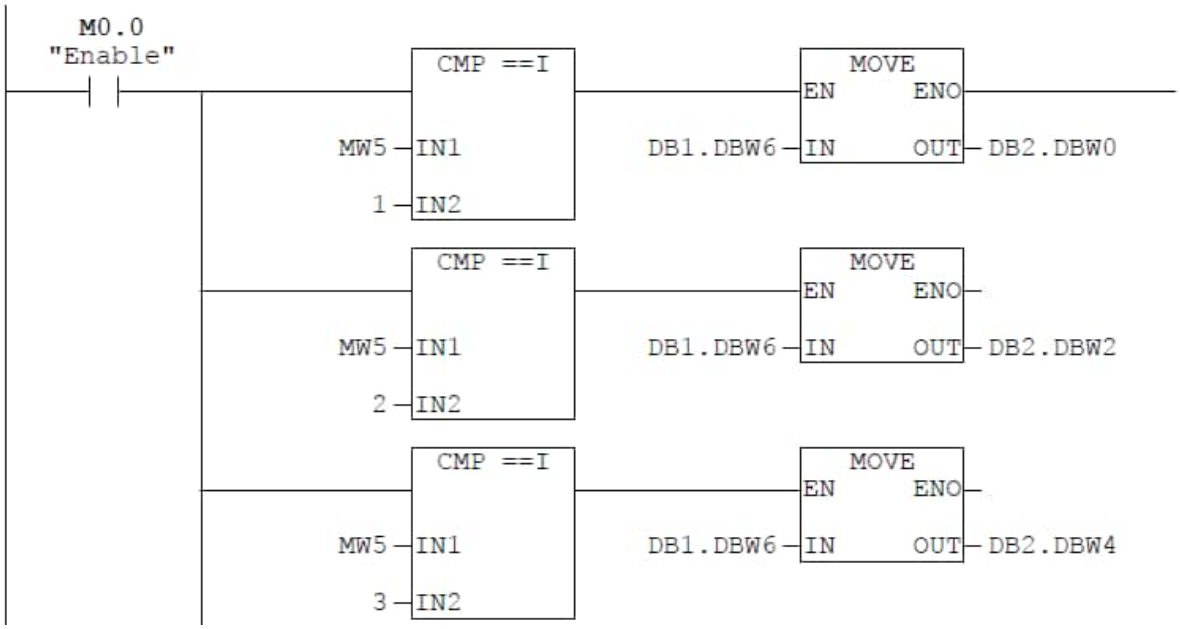
(OB10)

Network: 1 **OB10 "Time of Day Interrupt"**

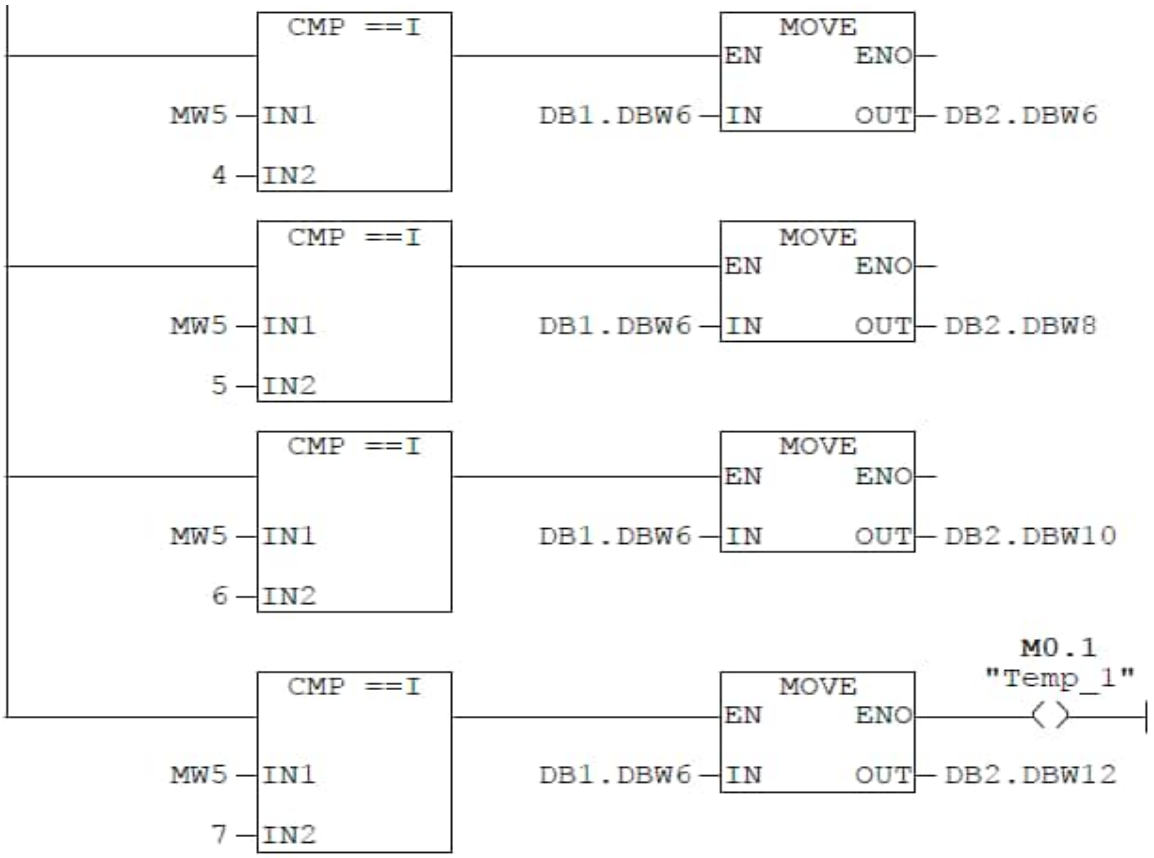
Network: 2



Network: 3



Network: 4

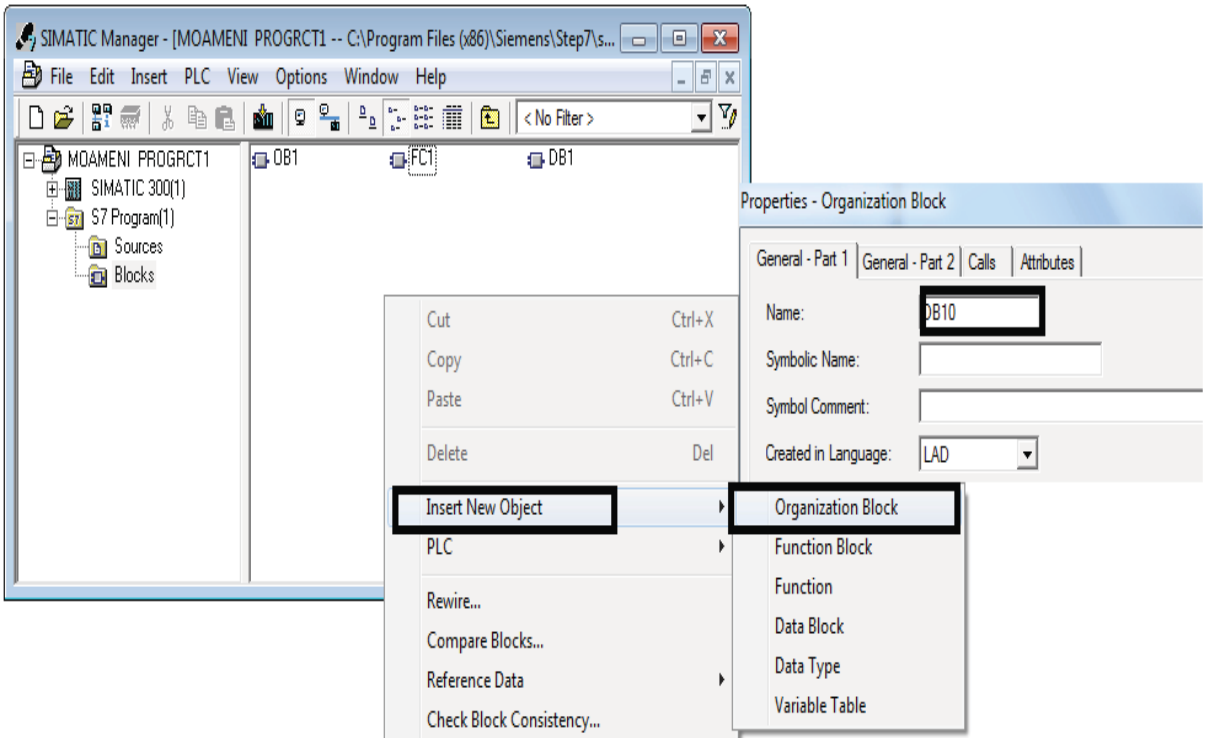


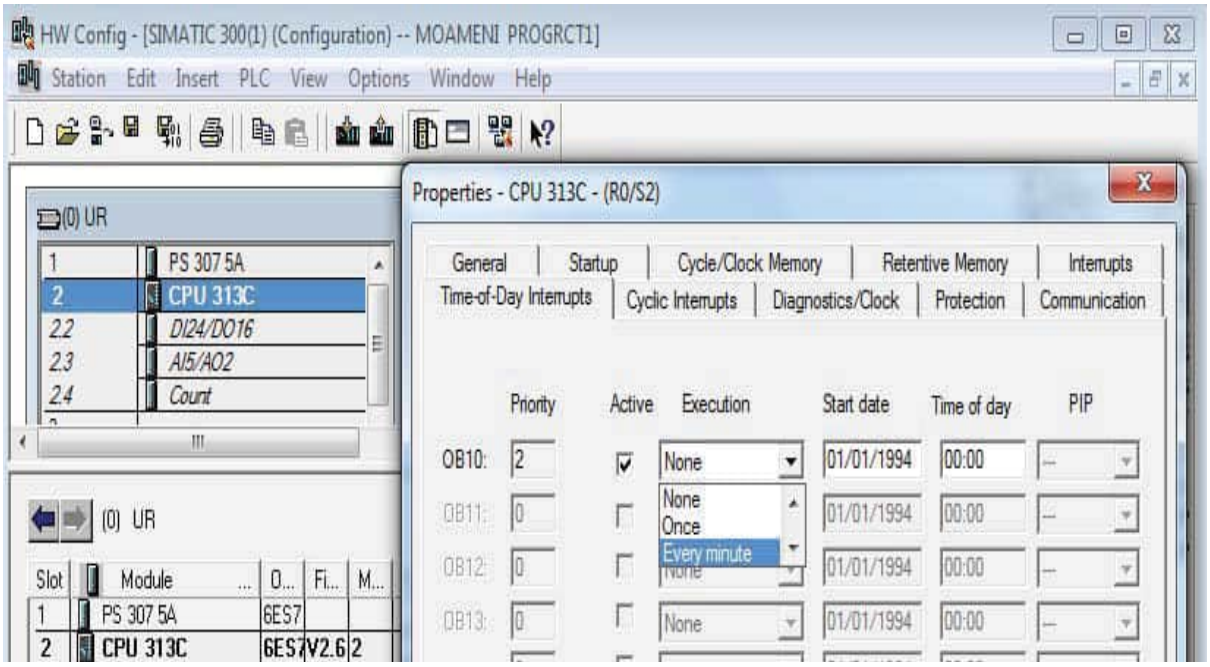
تشریح پروژه: از آنجایی که در پروژه مطرح شده که حداقل شمارش ۳۰۰ عدد می باشد. بهتر است از تابع سیستمی SFBO که شمارنده بالا رونده می باشد و ۳۲۷۶۸ شمارش می کند ، استفاده شود .با فشردن شستی start در Network2 با هرپالس سنسور نوری (sensor) شمارنده یک واحد به مقدار DB1.DBW6 که آدرس داخلی DB1,CV می باشد و به SFBO نسبت داده شده اضافه می گردد . همچنین در Network3 به کمک تابع SFC1 ساعت سیستم خوانده شده و در متغییر DAT که به فرمت Data and Time است قرار می گیرد که در Network4 با عدد ۳۰۰۰ مقایسه می گردد و در صورتی که مقدار آن کمتر از این عدد باشد ، توسط تابع FC7 که Data And Time To day روز وقوع این اتفاق را متناسب با روز هفته از ۱ تا ۷ در متغییر MW70 قرار می دهد . از آنجایی از ما خواسته شد که از تاریخ ۸/۱۲/۲۰۱۰ ساعت 8:00 صبح رهونه گیری شروع شود ، باید از وقفه Time of Day استفاده کنیم ، که از مسیر :

Simatic 300 station → Hardware→Time of Day Interrupts

مطابق شکل پروژه قبل تاریخ و زمان را تنظیم کرده و پس از Save and compile ، درون PLC بار گذاری می کنیم سپس مطابق شکل های زیر وقفه OB10 را ایجاد کرده برنامه را به این گونه ادامه می دهیم که در Network1 از تیغه N.C ،

بیت Temp-9 به خروجی Temp-9 متصل می کنیم که هر بار به OB10می باشد که هر روز یک بار در ساعت 8:00 فراخوانی می شود.





در Network2 با هر بار تغیر وضعیت Tamp-9 چه فعال شدن و چه غیر فعال شدن توسط — (N)- و —(P)- تشخیص داده شده و علاوه بر Reset کردن شمارنده SFBO در OB1 توسط متغییر Reset(M1.2) ، شمارنده C1 که هر روز یک

واحد به مقدار MW5 اضافه می کند ، تریگر می شود . در Network3 ، در روز اول با تابع Cmp==I ، مقدار DB1.DBW6 که در واقع شمارنده SFBO می باشد به Move, DB2.DBWO می شود. در ادامه مقدار Mw5 بااعداد ۲ تا ۷ مقایسه می شود و در هر روز مقدار جاری شمارنده اصلی به درون یک آدرس جدید از Mpve , DB2 می گردد. در انتها و روز هفتم توسط بیت Temp-1 شمارنده Reset,C1 می گردد و هفته بعد مجدداً پروسه تکرار می گردد . در هر مرحله که کلید Stop فشرده شود ، پروسه Reset می شود.

۵-۱۰عنوان پروژه: زمان سنج دقیق فعال بودن (Run) ، CPU با دقت 10ms

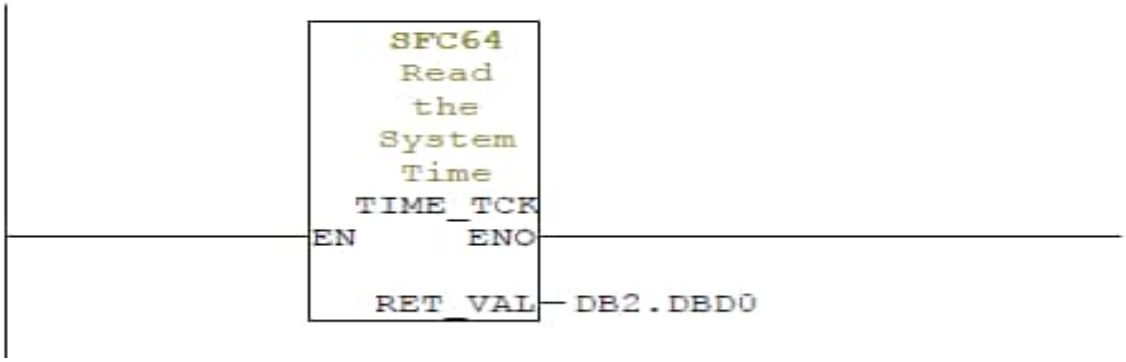
شرح پروژه: برنامه ای بنویسید که ، ۳۰ ثانیه پس از Run شدن CPU ، خروجی Q8.3 ، set شود ، همچنین تا زمانی که CPU در مد Run قرار دارد زمان سنجی دقیق ، مقدار جاری زمان طی شده را در DB2 ذخیره کند.

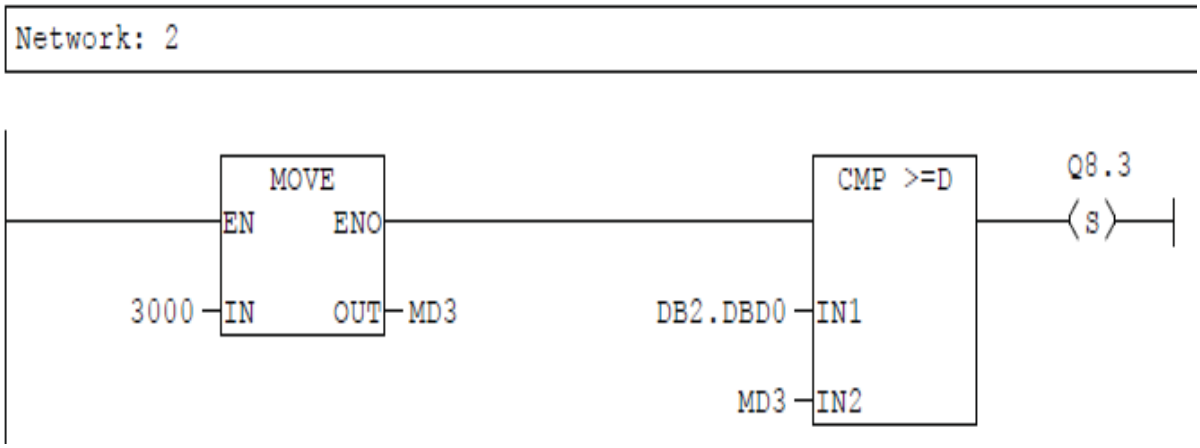
راه حل پروژه :

Block: DB2

Address	Name	Type	Initial value
0.0		STRUCT	
+0.0	SystemTime	DWORD	DW#16#0
=4.0		END_STRUCT	

Network: 1 OB1 Reading the System Time





تشریح پروژه : به کمک تابع SFC64 که به نام Time-Tck شناخته می شود می توان زمان Run بودن CPU را بر حسب میلی ثانیه اندازه گیری کرد که دقت آن در مدل های S7-۴۰۰ و CPU318 برابر یک میلی ثانیه و بقیه مدل های خانواده S7 - ۳۰۰ برابر ۱۰ میلی ثانیه است . حداکثر زمانی را که می تواند اندازه گیری کنید . نیز ۲۱۴۷۴۸۳۶۴۷ میلی ثانیه ، معادل تقریباً ۵۹۶ ساعت می باشد که خود معادل نزدیک به ۲۵ روز کار مداوم می باشد!

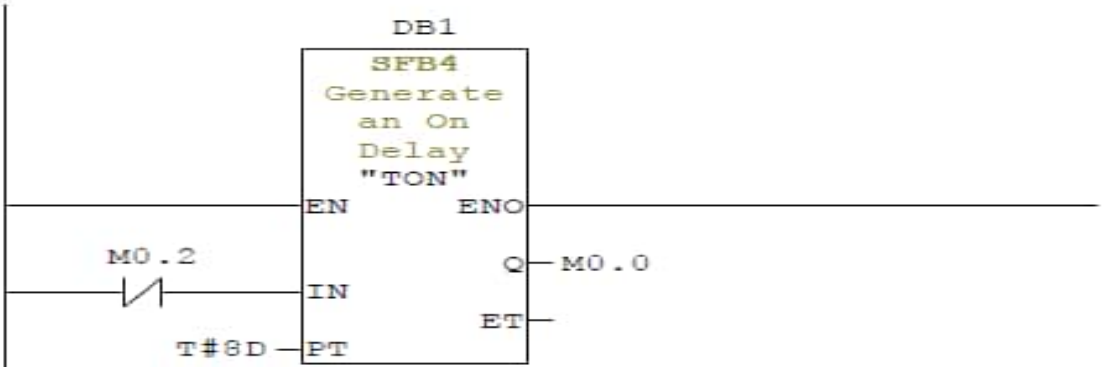
از آنجایی که دقت اندازه گیری در خانواده S7-300 برابر 10ms است ، مقدار جاری زمان که در آدرس DBS.DBDO ذخیره شده است را با عدد ۳۰۰ معادل ۳۰ ثانیه مقایسه کرده و در صورت رسیدن به این زمان خروجی QB.3 را set می کند. دقت شود که در صورت Stop شدن CPU آخرین مقدار نگه داشته می شود ولی اگر مجدداً Run شود ، زمان گیری از صفر شروع می شود.

۶-۱۰عنوان پروژه : غیر فعال کردن موقت وقفه ، Cyclic Inerrupt ، ۸ روز پس از Run شدن CPU

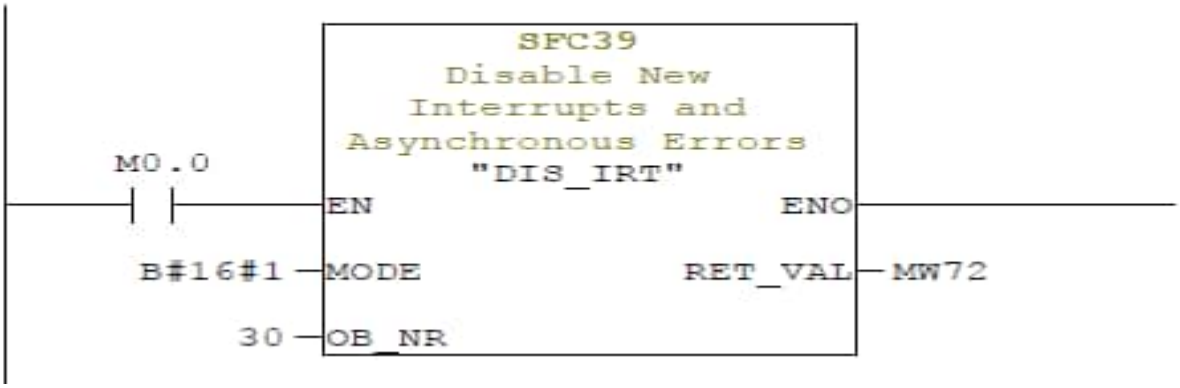
شرح پروژه: برنامه ای بنویسید که پس Run شدن CPU ، تایمری شروع به کار کند و ۸ روز بعد ، وقفه OB35 را که به طور مرتب هر ۲ ثانیه یکبار اجرا می شود را به مدت ۳ روز و ۱۲ ساعت غیر فعال نگه دارد.

راه حل پروژه :

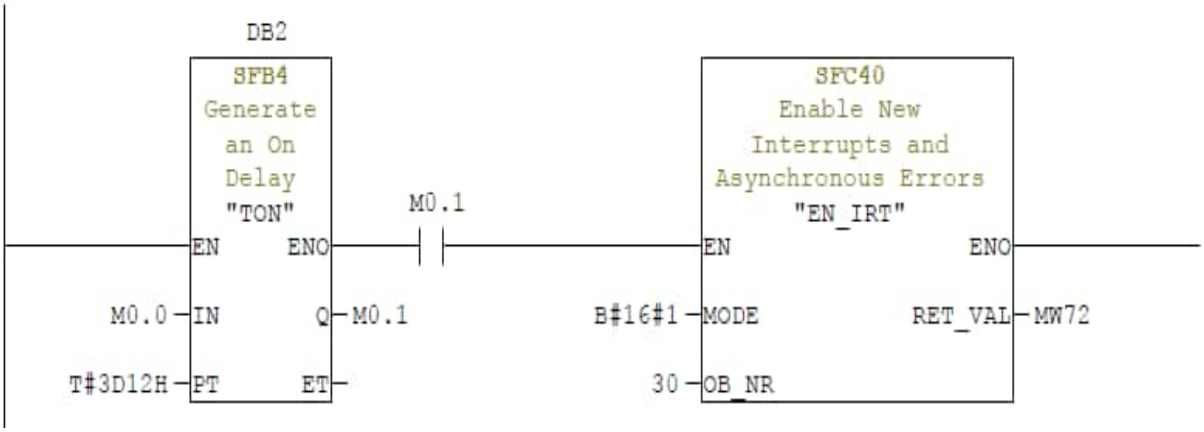
Network: 1



Network: 2



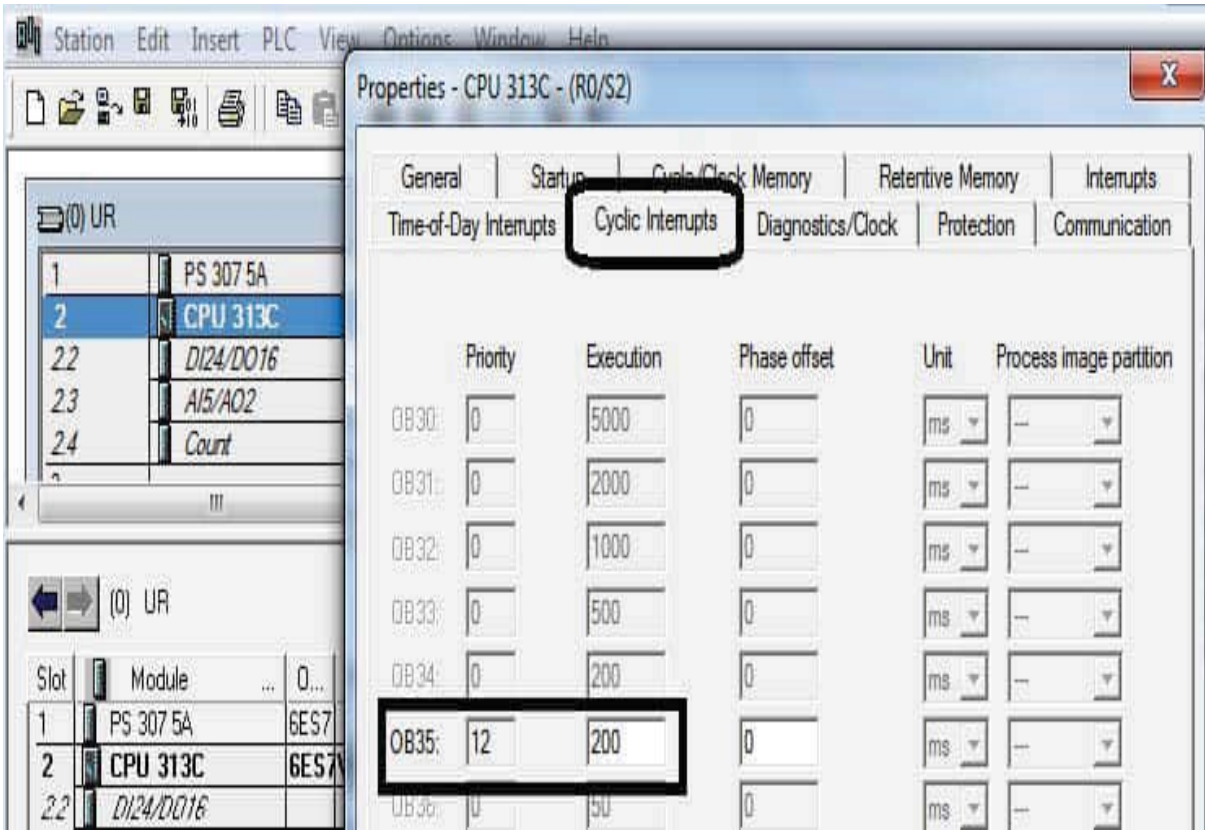
Network: 3



تشریح پروژه : برای حل این پروژه ابتدا از مسیر:

Hardware Config →cpu313c→cyclic Interrupts

مطابق شکل زیر مقدار وقفه OB35 را برابر ۲۰۰۰ قرار می دهیم که معادل ۲ ثانیه می باشد ، سپس Save و Download می کنیم. حال پس از اجرای برنامه و پس از Run شدن cpu هر ۲ ثانیه یک بار این بلوک سیستمی را فراخوانی می کند. در Network1 ، OB1 همزمان با Run شدن cpu تایمری سیستمی از نوع ON Delay با نام SFB4 شروع به زمان گیری می کند.



تفاوت این تایمر با تایمرهای معمولی در دو پارامتر اصلی زمانی است:

۱.زمان اندازه گیری شد ه توسط تایمر معمولی ۲ ساعت و ۴۶ دقیقه و ۳۰ ثانیه می باشد ، در صورتی که با تایمر های IEC-TC می توان تا ۲۴ روز و ۲۰ ساعت و ۲۳ ثانیه زمان گیری کرد.

۲.دقت در تایمرهای معمولی حداکثر ۱۰ میلی ثانیه است ولی در این تایمرها ، ۱ میلی ثانیه است!

۱۰-۷ عنوان پروژه : نشان دهنده مجموع زمان روشن بودن موتور

شرح پروژه : فرض کنید ، موتوری در طول روز به صورت متناوب ، روشن وخاموش می شود . برنامه ای بنویسید که مجموع زمان روشن بودن موتور را بر حسب ساعت در DB قرار دهد.

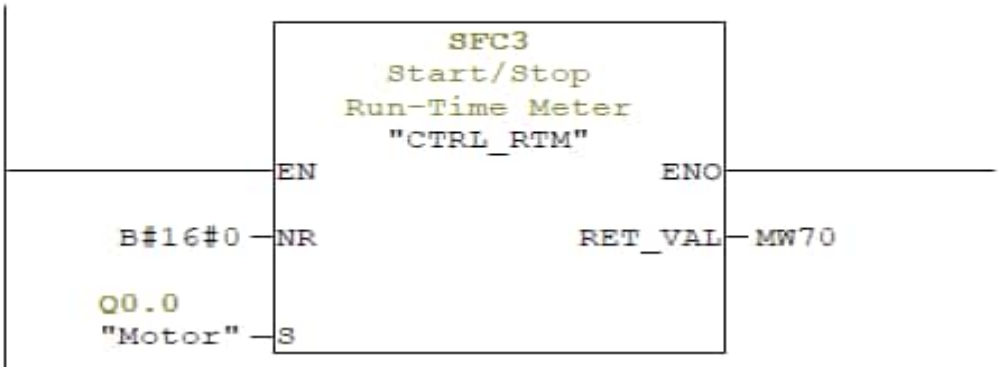
راه حل پروژه :

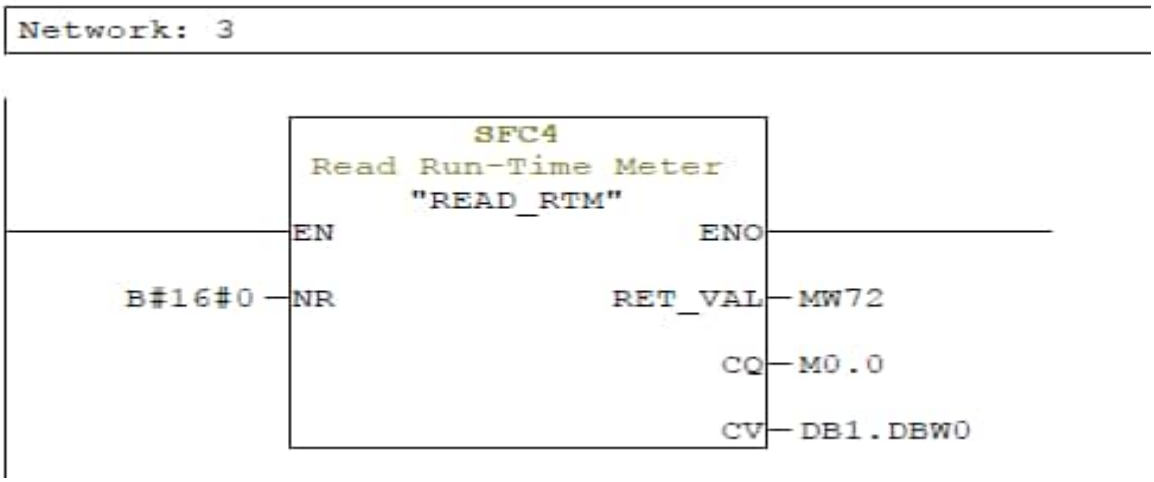
Block: DB1

Address	Name	Type	Initial value
0.0		STRUCT	
+0.0	RTM	WORD	W#16#0
=2.0		END_STRUCT	

Network: 1 OB1 Run Time Meter

Network: 2



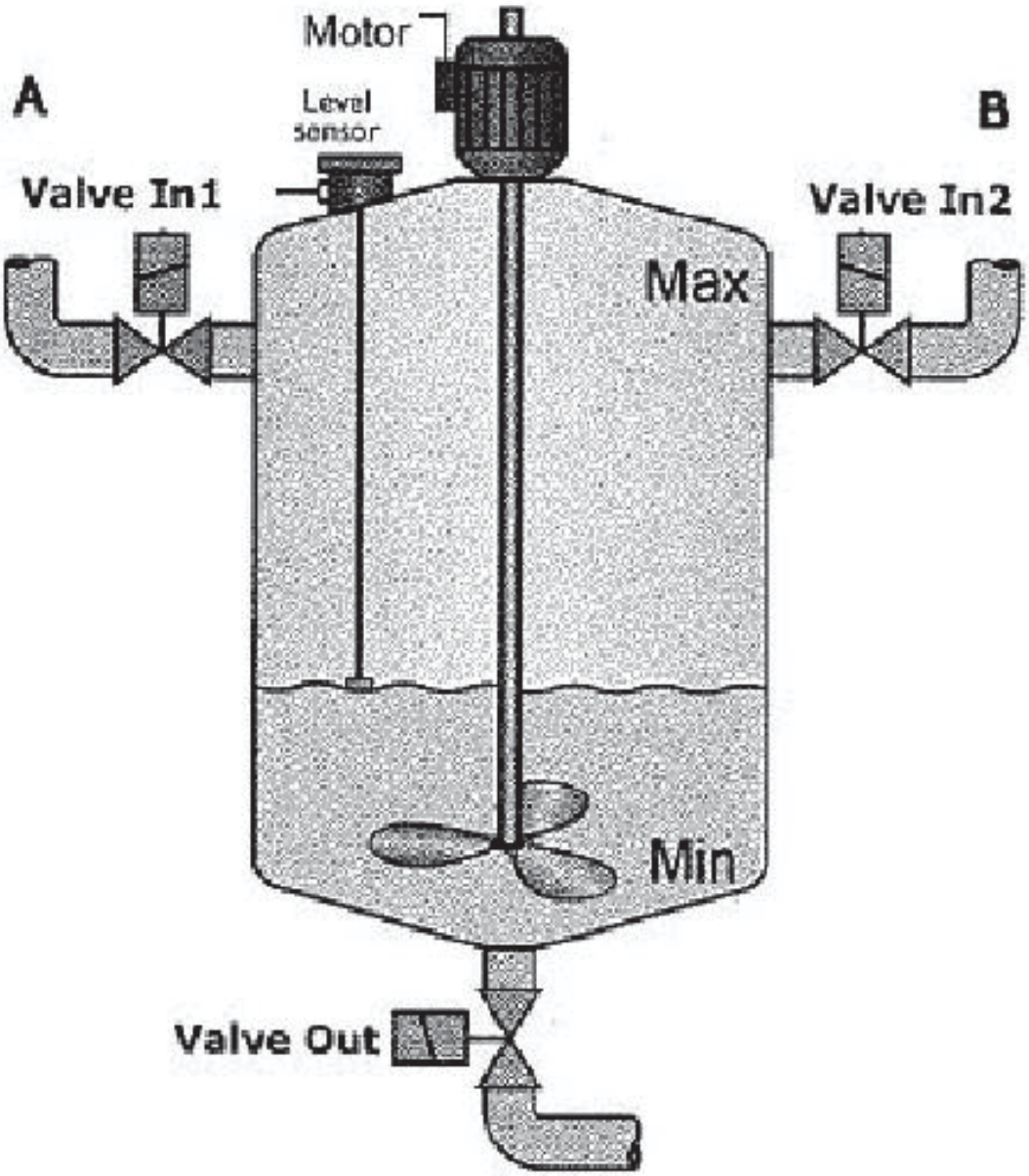


تشریح پروژه : خانواده S7 دارای واحدی به نام RTM (Run Time meter) می باشد که همان طور از نام پیداست ، زمان سنجی است که زمان فعال بودن خروجی خاصی را اندازه گیری می کند . دو دسته از RTM را می توان در برنامه استفاده کرد دسته 16bit , 32bit. در دسته اول تا حداکثر ۳۲۷۶۸ ساعت زمان گیری کرد و حداکثر ۸واحد مستقل را شامل می شود . برای اندازه گیری زمان های بیشتر از RTM ، 32bit استفاده می شود ، که تا ۱۵ واحد مستقل را در اختیار شما قرار می دهد . در این پروژه ازواحد 16bit استفاده شده است . در Network1 ، Motor توسط ورودی SW روشن و خاموش می شود . در Network2 توسط تابع Sfc3 که در واقع فرمان شروع و توقف زمان سنج را صادر می کند با ورودی آدرس NR (Number RTM) که کد B#16#0 به معنی استفاده از RTM شماره صفر می باشد . با فعال و غیر فعال شدن Motor که به ورودی S وصل شده زمان گیری را شروع و متوقف می کند . MW70 گزارش خطای احتمالی را می دهد که در حالت عادی عددی برابر صفر است . در Network3 به کمک SFC4 مجموع زمان RTM مورد نظر خوانده شده و در DB1 ذخیره می شود . MO.O زمانی که واحد RTM فعال است Set می شود. دقت کنید که حداقل باید یک ساعت از روشن بودن Motor گذشته باشد که تغییری در خروجی CV و CQ مشاهده شود.

۸-۱۰عنوان پروژه : سیستم کنترل مخزن به کمک سنسور تعیین سطح آنالوگ

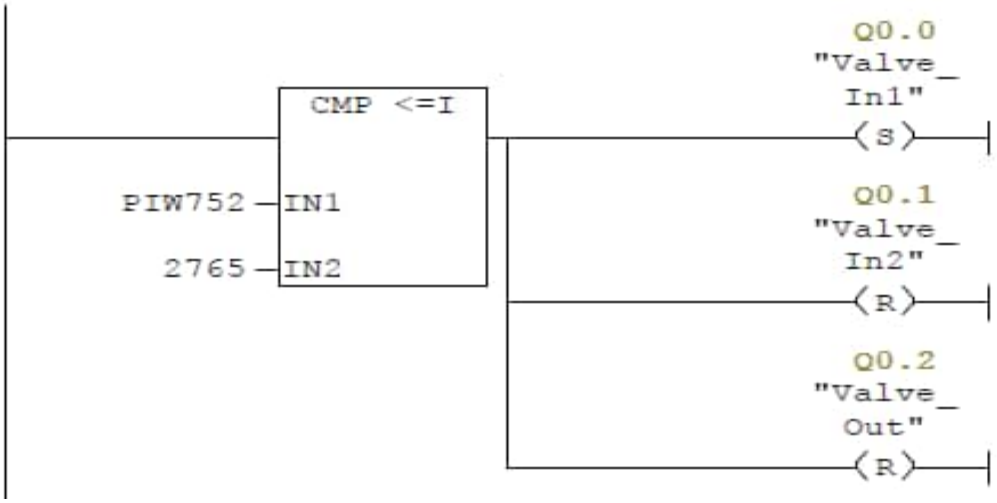
شرح پروژه : مطابق شکل زیر ، سنسور تعیین سطح آنالوگی ، سطح سیال درون مخزن را اندازه گیری کرده و متناسب با آن جریان خروجی $4\sim ۲۰mA$ را به ورودی PLC,AI1 می دهد و نسبت آن هم کاملاً خطی فرض می شود به صورتی که وقتی مخزن خالی است ، جریان ۴ و زمانی که مخزن کاملاً پر است مقدار ۲۰ میلی آمپر را می دهد . می خواهیم برنامه ای بنویسیم که در صورتی که سطح مخزن کمتر از ۱۰% بود ، Valve In2 و شیر برقی خروجی Valve out بسته باشد. مواد ورودی A وارد مخزن شده تا زمانی که سطح مخزن به ۳۵% برسد در این زمان شیر برقی Valae In1 بسته و Valve In2 باز شود. این بار مخزن با ورودی مواد B پر می شود. وقتی سطح مخزن به ۹۰% Valve In2 نیز بسته شود سپس موتور

همزمان به مدت ۳۰ ثانیه دو ماده را با هم مخلوط کند . پس از خاموش شدن موتور همزن ، Valve out باز شده و مواد اماده تخلیه شود . با تخلیه مواد سطح مخزن کم می شود تا زمانی که به زیر ۱۰% برسد ،حال پروسه مجدداً تکرار شود . دقت کنید هنگام کم شدن سطح مخزن و رسیدن به مرز ۳۵% مجدداً حالت تعریف شده قبلی اتفاق نیفتد !

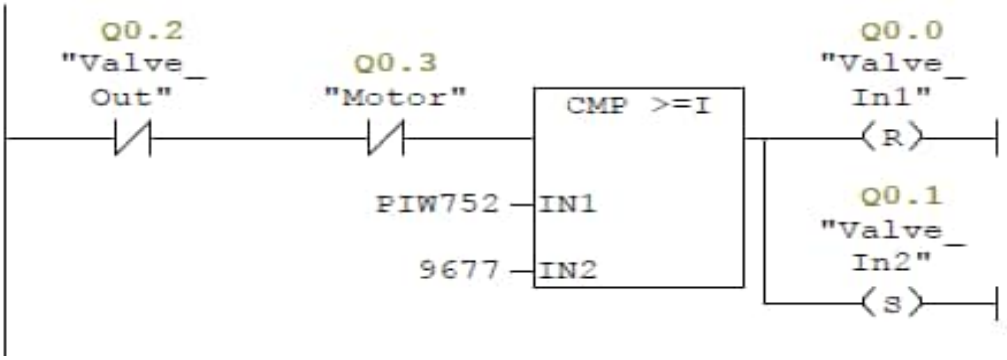


راه حل پروژه :

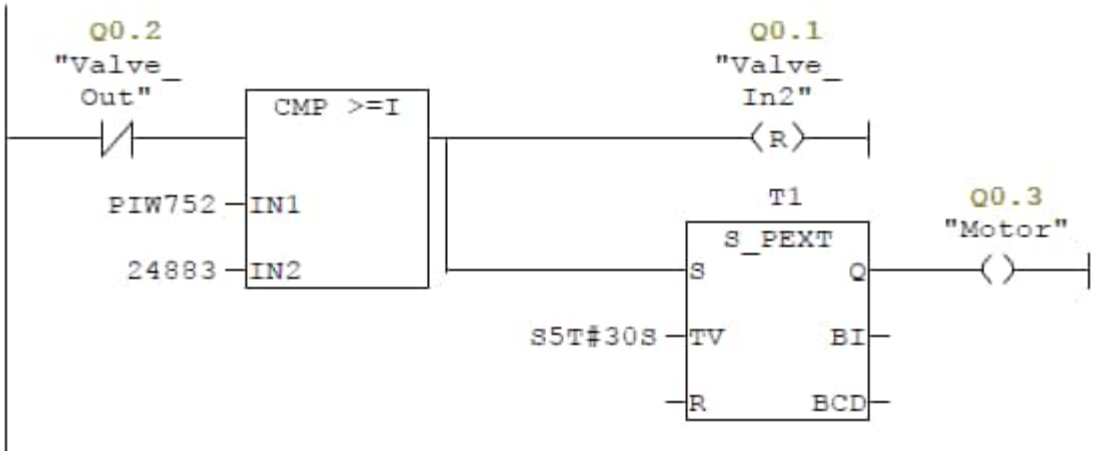
Network: 1 Block: OB1 Cistern Control Analog

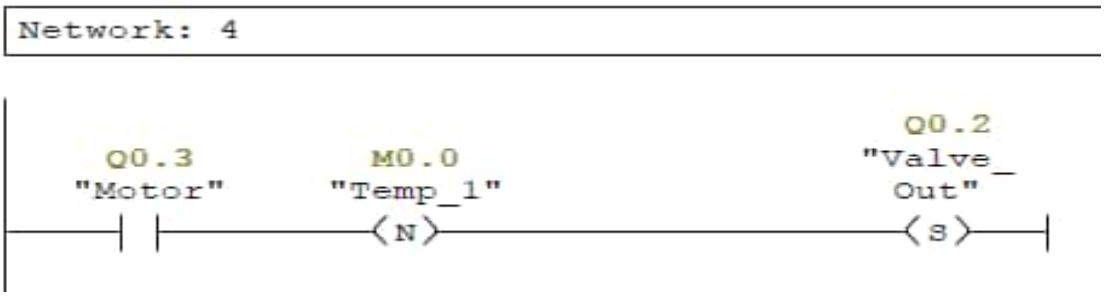


Network: 2



Network: 3

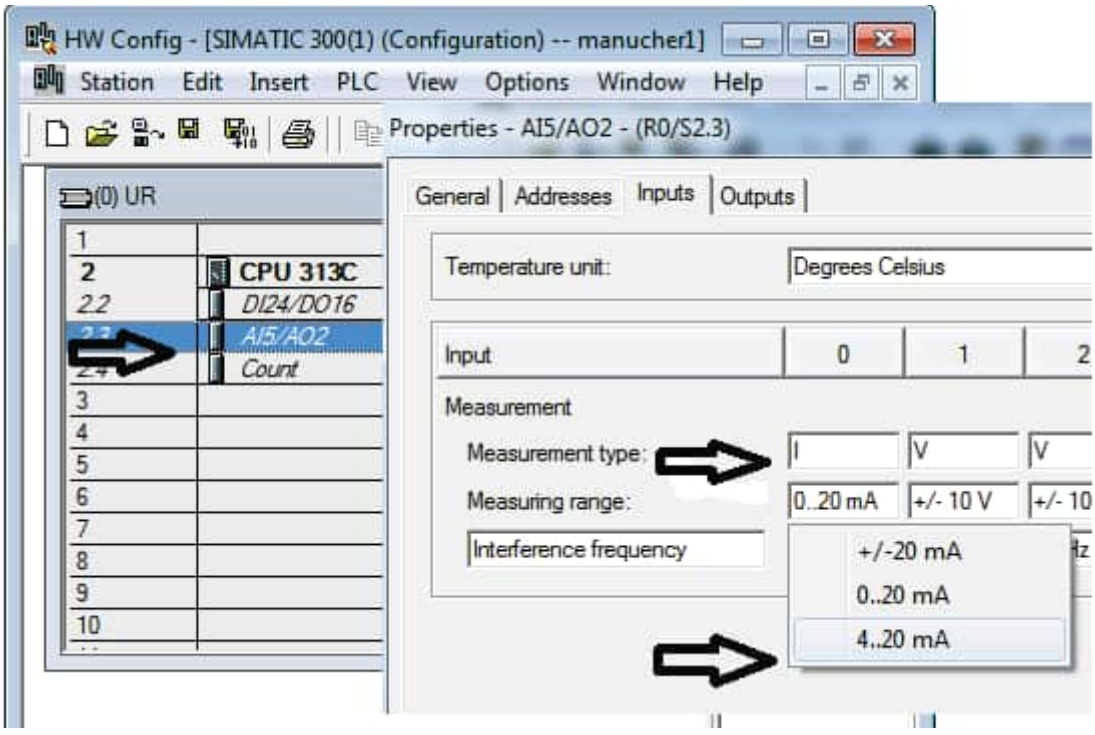




تشریح پروژه: قبل از تشریح حل پروژه ، ابتدا به بررسی تنظیمات کارت ورودی آنالوگ می پردازیم . همان طوری که در صورت مسأله ذکر شده سنسور سطح انالوگ ، خروجی 4-20Ma را به ورودی PLC می دهد. PLC مورد استفاده ما CPU313C می باشد که به صورت Comact ,4 ورودی آنالوگ را در خود دارد. کافی است ازمسیر:

Simatic300 station →Hardware→A15/A02

مطابق شکل ۱۷-۹ ابتدا Measuremenr Type را در Input1 روی جریان (I) قرار داده وسپس رنج آن را روی 4 20Ma ~ قرار می دهیم و در انتها Save و Download می کنیم.



همان طوری که می دانیم واحد AD CPU کارت ورودی انالوگ ، درازای ورودی آنالوگ ولتاژ یا جریان با هر رنجی که باشد مقدار ۰ تا ۲۷۶۴۸ را در متغییر مورد نظر ما قرار می دهد . به عنوان مثال اگر ورودی آنالوگ PIW752 که اولین آدرس در کارت آنالوگ CPU313 C می باشد ، در رنج 20MA ~۴ تعریف شود ، درازای جریان ۴ میلی آمپر مقدار ۰ و در

جریان ۲۰ میلی آمپر مقدار ۲۷۶۴۸ را در PIW7752 قرار می دهد. از انجایی که در پروژه از ما خواسته شده که سطح کمتر از ۱۰% را تشخیص دهد و معیار درصد میباشد باید به کمک ماشین حساب ۱۰ درصد از ۲۷۶۴۸ را محاسبه کنیم که برابر با ۲۷۶۴.۸ می شود که در Network1 این مقدار را گرد کرده و ۲۷۶۵ در نظر گرفته و با ورودی PIW725 مقایسه کردیم.

هر گاه ورودی کمتر از این مقدار بود همان طور که خواسته شد ، Valve-In1 باز و Valve-In2 وValve outبسته می شود.در Network2 بادرنظرگرفتن حفاظت که حتماموتورهمزن و Valve out خاموش است ، در صورتی که سطح مخزن به ۳۵% رسید که معادل گرد شده ۹۶۷۷ است ، Valve-In2 بازو Valve-In1 بسته شود. در Network3 در صورتی که Valve out بسته باشد ، اگر مقدار مخزن به ۹۰% رسید که معادل گرد شده آن ۲۴۸۸۳ است ، ابتدا شیر برقی Valve-In2 بسته و سپس موتور همزن به کمک تایمر S-PEXT به مدت ۳۰ ثانیه مواد را مخلوط می کند.

هدف اصلی ما از قرار دادن تیغه Valve out ,N.C در سر راه Network3 و Network2 آن است که در زمانی که سطح مخزن به اعداد تعریف شده ۹۰ و۳۵ رسید ، شیرهای ورودی فعال نشوند. در Network4 با خاموش شدن Motor ، شیر برقی خروجی Valve out فعال شده و مواد تخلیه می شوند . زمانی که سطح مخزن به کمتر از ۱۰% رسید ، در Network1 پروسه مجدداً تکرار می شود.



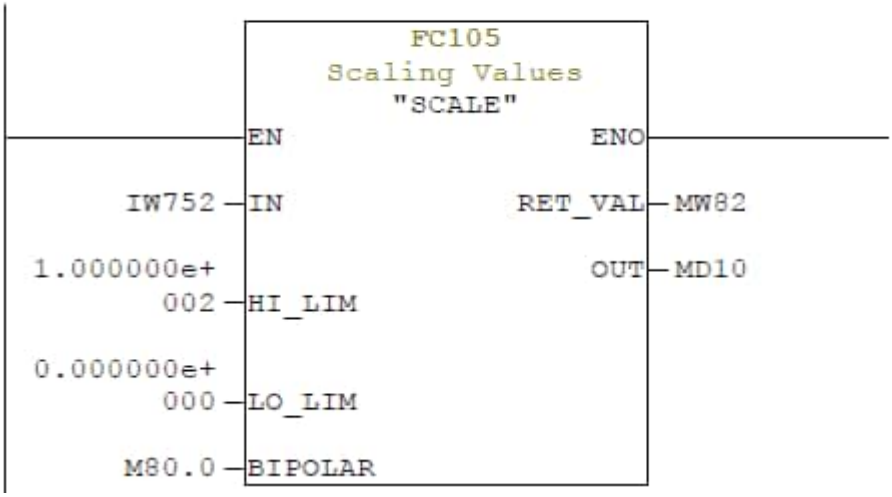
چند مدل از سنسور های اندازه گیری سطح از نوع Ultrasonic

۹-۱۰ عنوان پروژه : سیستم کنترل سطح مخزن به کمک سنسور تعیین سطح آنالوگ (با تابع Scale)

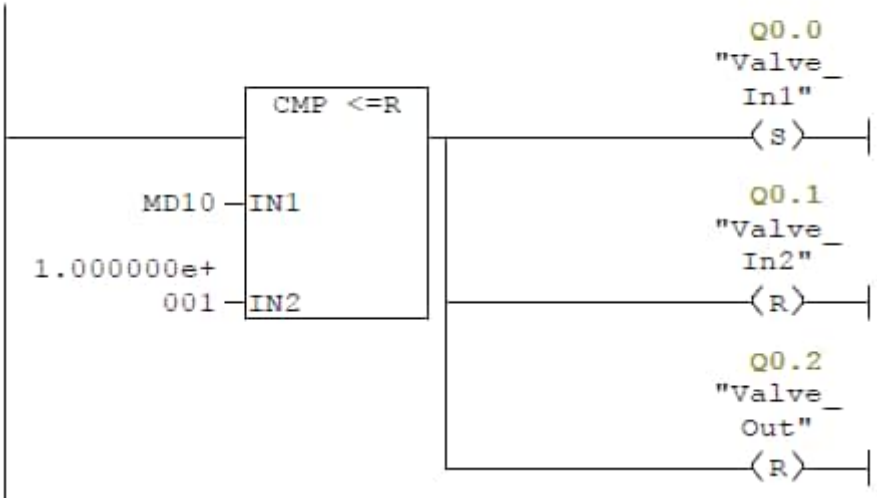
شرح پروژه : شرح این پروژه مانند پروژه قبلی می باشد ، با این تفاوت که آن را به کمک تابع FC105 که مبدل Scale می باشد ، طراحی کنید.

راه حل پروژه :

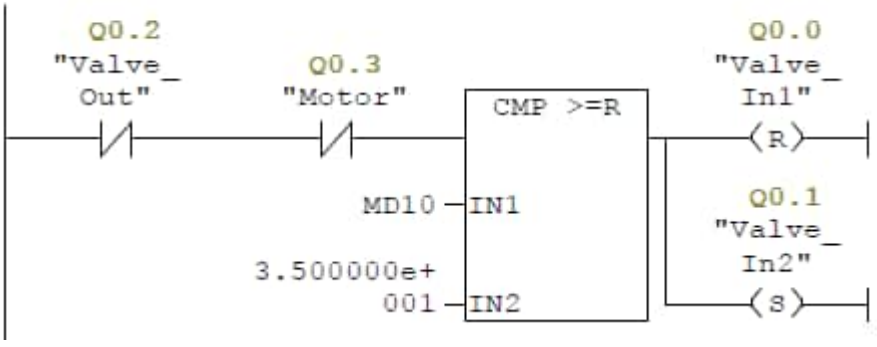
Network: 1 Block: OB1 Cistern Control Analog (Scale)

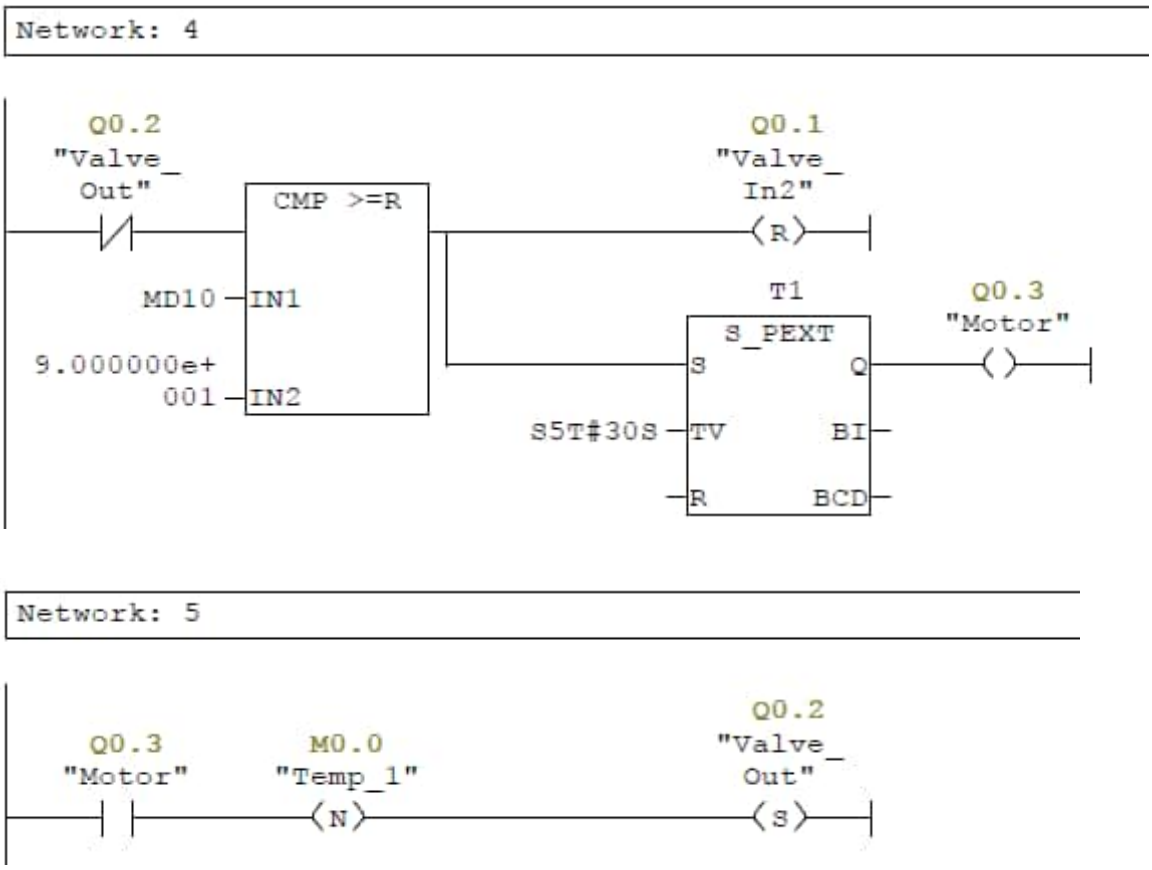


Network: 2



Network: 3



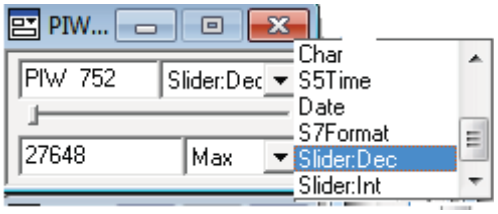


تشریح پروژه : همان طوری که پروژه قبلی مطرح شد ، مقدار تبدیل شده ADC از ورودی آنالوگ ۰~۲۷۶۴۸ می باشد و از انجایی که این عدد به گونه ای است ک کار با آن برای تشخیص مقدار ورودی و نمایش ان روی HMI کاری بسیار مشکل است ، راه چاره استفاده از تابع Scale برای تبدیل این مقدار به رنج مورد نظر ما می باشد . از انجایی که در این پروژه معیار ما درصد می باشد ، بهتر است این مقدار را در بازه ۰~۱۰۰ مقیاس بندی کنیم. مطابق Network1 ابتدا از مسیر:

Libraries →standard Libraries → TI-S7 Converting Block →FC105

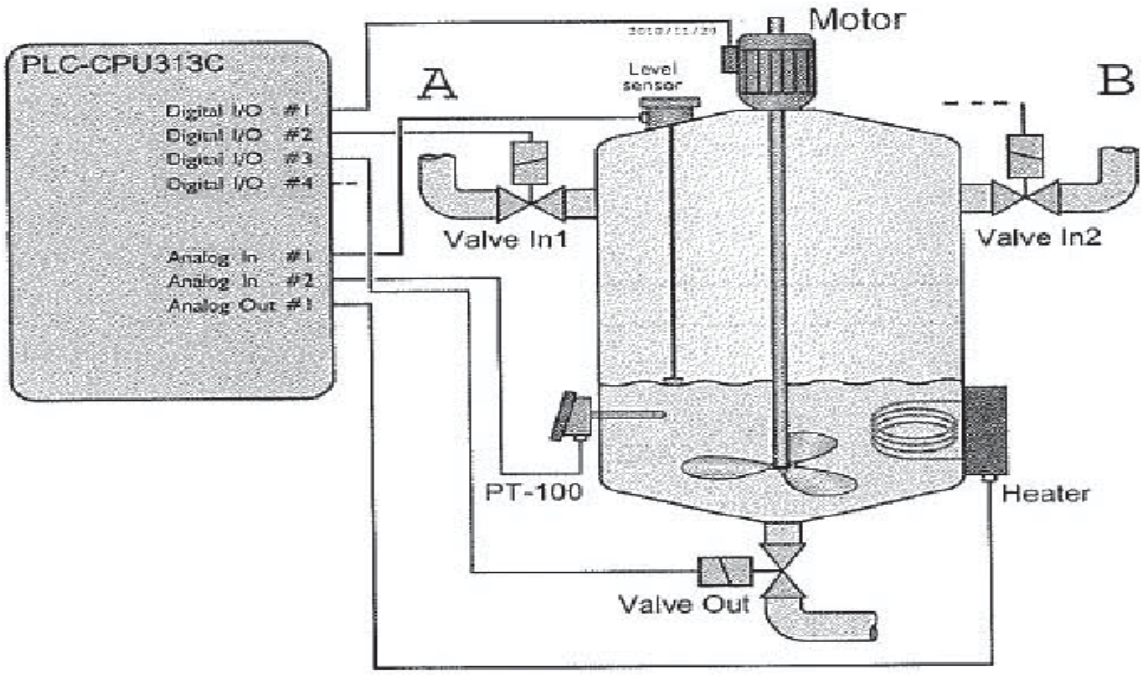
بلوک FC105 را وارد Network۱ می کنیم . ورودی IN این تابع را به سنسور آنالوگ نسبت می دهیم . که در اینجا IW725 می باشد ، بدون آنکه برای ما مهم باشد که جنس آن ولتاژ است و یا جریان مقدار HI-Lim را روی ۱۰۰ قرار می دهیم ولی باید دقت کنید که برای وارد کردن عدد باید حتماً نقطه و صفر را هم بعد از آن در نظر بگیرد زیرا جنس این ورودی اعشاری است پس تایپ می کنیم ۱۰۰.۰ و پس ازفشردن کلید Enter مشاهده می کنید که به صورت 1.000000e+002 دیده می شود. مقدار Lo-Lim را نیز با ۰.۰ مقدار دهی می کنیم . Bipolar در صورتی استفاده می شود که ورودی آنالوگ ما دارای مقادیر منفی نیز باشد مانند $\pm ۱۰v$ یا $\pm ۲۰mA$ که در این پروژه به این صورت نیست .

Ret-val مقدار کدی را بر حسب Hex به ما می دهد که گزارشی از عملکرد تابع است و اگر خطایی رخ نداده باشد مقدار ۰ را بر می گرداند. مقدار تبدیل شده در MD10 قرار می گیرد. مابقی Network ها مانند پروژه قبلی می باشد با این تفاوت که نوع مقایسه گر از نوع R یا اعشاری انتخاب می شود و به عنوان مثال اگر بخواهیم عدد ۳۵% را تشخیص دهد کافی است در ورودی IN2 وارد کنیم ۳۵.۰ برای شبیه سازی نیز بهتر است مطابق شکل زیر ورودی PIW725 از نوع Slider-Dec انتخاب کرده و مقدار Min آن را ۰و مقدار Max را ۲۷۶۴۸ تعیین کنیم تا حدود حرکت لغزنده از ۰ تا ۲۷۶۴۸ باشد که مقدار ورودی را شبیه سازی می کند.



۱۰-۱۰پروژه : سیستم کامل کنترل فرایند در مخزن

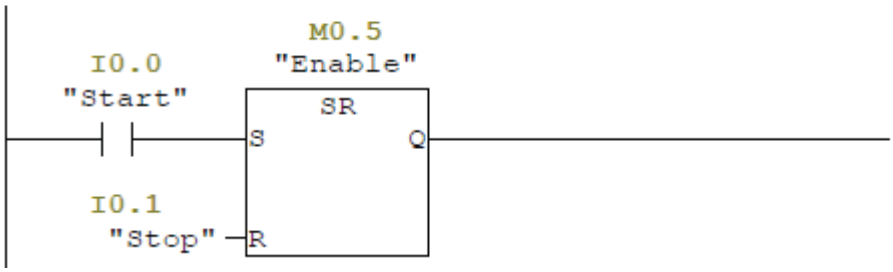
شرح پروژه : با فشردن کلید Start مطابق شکل زیر ، مخزن با نسبت $\frac{۱}{۳}$ مواد A و $\frac{۲}{۳}$ مواد B که توسط شیربرقی Valve-In1 و Valve-In2 واردمخزن می شود . پر گردد . تشخیص سطح مخزن بر عهده سنسور آنالوگ تعیین سطح Level sensor صورت می گیرد . پس از پر شدن مخزن Heater که به صورت آنالوگ و ولتاژ 10V-0 کنترل می شود ، به گونه ای مقیاس بندی شده که در ازای ولتاژ ۰ ولت ورودی خاموش و درازای 10Vورودی دمای ۱۰۰ درجه سانتیگرادراتعیین می کندواین نسبت بین ولتاژورودی ودمای ایجاد شده کاملاً خطی فرض می شود .



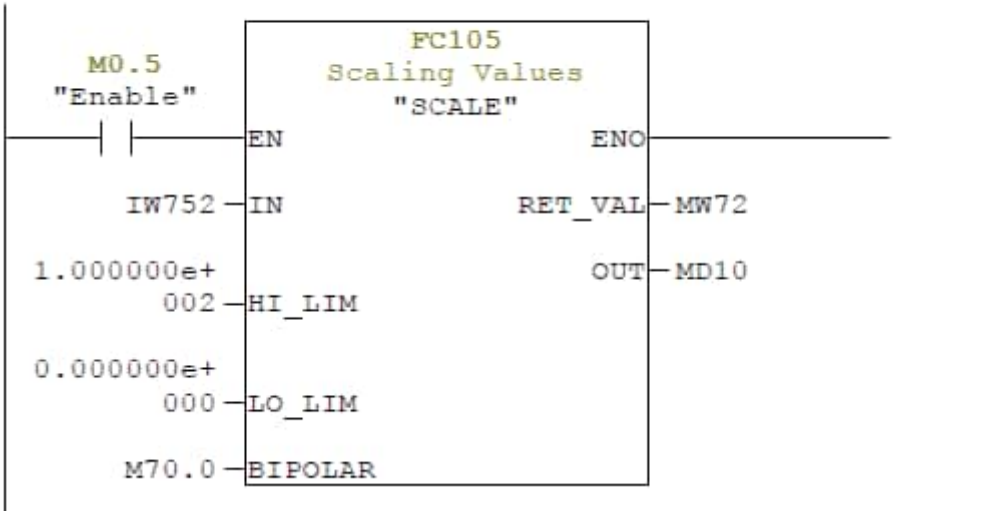
دمای سیال داخی مخزن را تا ۷۰ درجه سانتیگراد بالا می برد و عمل اندازه گیری دما توسط سنسور دمای PT100 صورت می گیرد. پس از رسیدن دمای سیال به ۷۰ درجه سانتیگراد موتور همزن به مدت ۴۰ ثانیه مواد را با هم مخلوط می کند . پس از پایان کار همزن Heater خاموش شده وسیستم آنقدرصبر می کند تا دمای سیال به ۳۰ درجه سانتیگراد برسد. در این حالت شیر برقی خروجی Valve Out باز شده و مواد تخلیه می شود . فرض بر ان گرفته می شود که در ابتدای کار مخزن کاملاً خالی بوده است. پس از تخلیه مواد و خالی شدن مخزن پروسه مجدداً تکرار گردد خالی بودن مخزن را ۵% و پر بودن آن را ۹۸% فرض می کنیم.

راه حل پروژه :

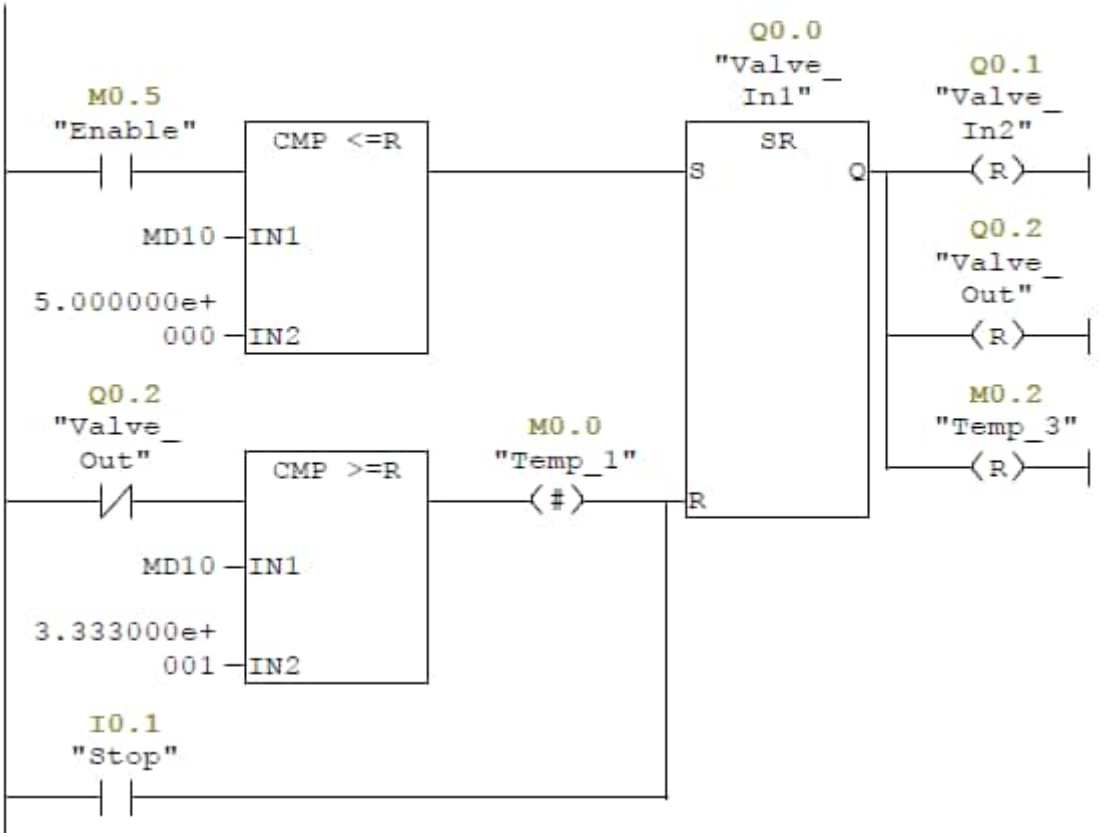
```
Network: 1 Block: OB1  Cistern Control Analog (Scale)
```



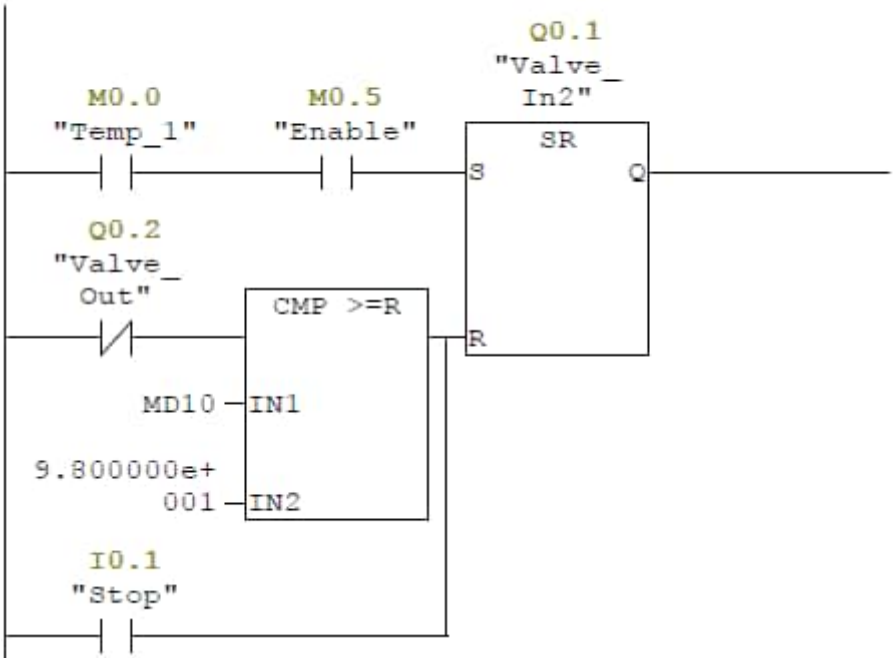
Network: 2



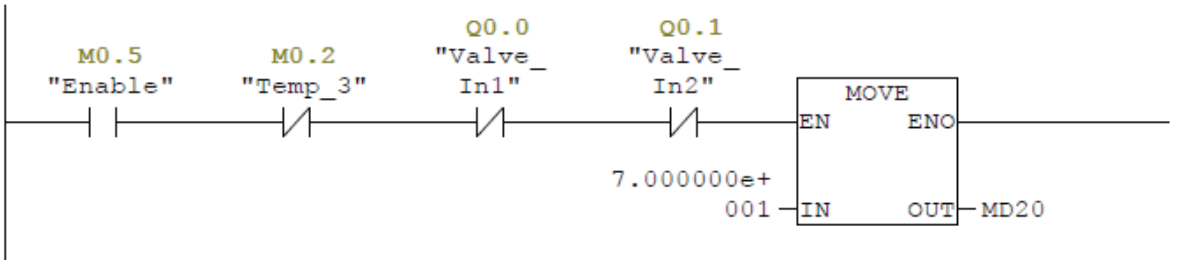
Network: 3



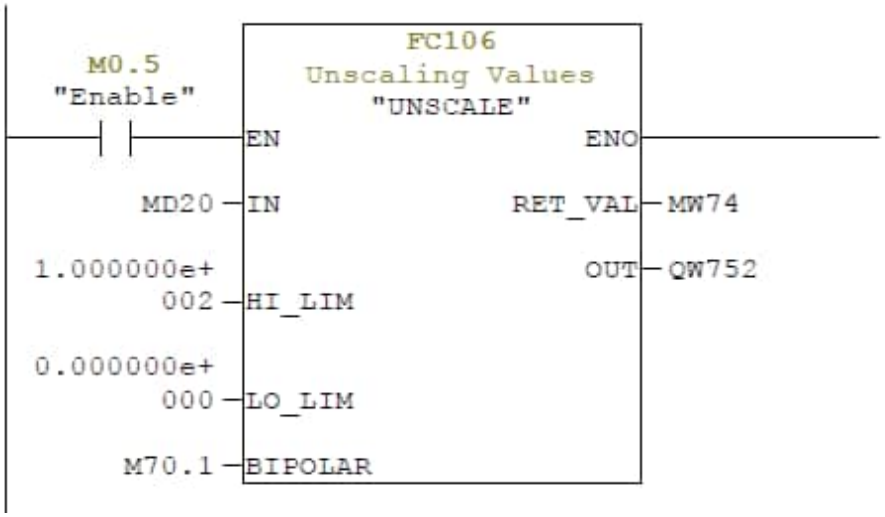
Network: 4



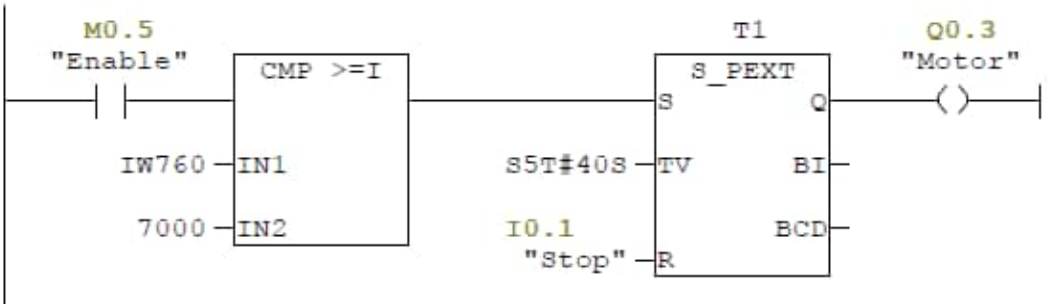
Network: 5



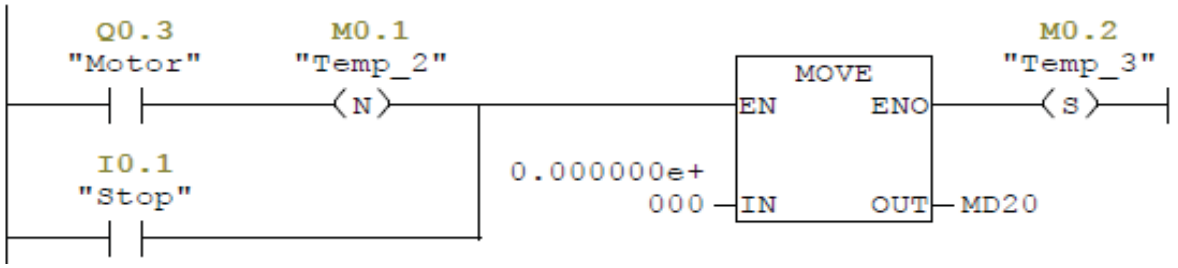
Network: 6



Network: 7



Network: 8



تشریح پروژه : این پروژه در ظاهر کمی پیچیده به نظر می رسد ولی با کمی دقت و ابتکار به راحتی می شود . در Network1 با فشردن کلید Start بیت Enable فعال می شود که در Network های بعدی اجازه کار را به سایر بلوک ها می دهد. در Network2 از تابع Scale (FC105) برای مقیاس بندی ورودی آنالوگ AI1 که مربوط به Level sensor می باشد استفاده شده که متناسب با سطح مخزن عددی بین ۰.۰ تا ۱۰۰.۰ را در MD10 قرار می دهد . در Network3

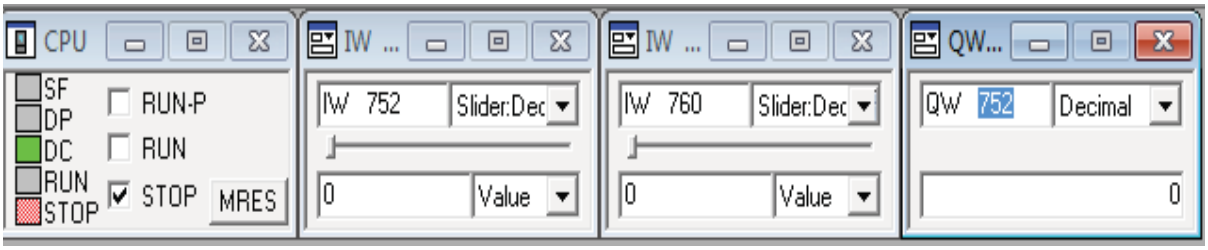
همان طوری که در مساله آمده است اگر سطح مخزن به کمتر از ۵% برسد ، آنرا خالی فرض می کنیم ، پس از قرار دادن تابع مقایسه $Cmp \leq R$ در صورت وقوع آن شیر برقی Valve –In1 را فعال Valvein2 و Valve out را می بندیم. در این حالت تنها مواد A وارد مخزن می شود.

همان طوری که خواسته شد ، $\frac{1}{3}$ مخزن باید از مواد Aپر شود که معادل درصدی ۳۳.۳۳% می باشد. سپس با قرار دادن دستور مقایسه CMP=>Rوقتی به این مقدار رسید ، در Network3 باعث از کار افتادن Valve in1و در Network4 به کمک بیت Temp-1 که به صورت Mid line از Network قبلی و در صورت پر شدن به انداز ۳۳.۳۳% فعال می گردد ، شیر برقی Valve in2 را فعال می کند تا مواد B وارد مخزن شود و تا حداکثر مخزن را پر کند.

چون در صورت مساله ، پر بودن مخزن را ۹۸% فرض کردیم ، با رسیدن به این مقدارشیر برقی Valve In2 نیز بسته می شود. درNetwork5 با این اتفاق مقدار ۷۰۰ وارد MD20 می گردد. در Network۶ از تابع unscale (FC106) استفاده شده که به منظور کنترل Heater می باشد. ورودی این تابع MD20 می باشد و چون در مرحله قبلی با ۷۰۰ بار گذاری شده است ، دمای Heater را روی ۷۰ درجه سانتیگراد قرار می دهد. دمای سیال که بالا می رود ، سنسور PT100 آنرا تشخیص داده و متناسب با آن مقدار IW760 را که به صورت پیش فرض در CPU313C مخصوص آن است را بالا می برد. نکته ای که باید توجه کنید آن است که مقدار قرار گرفته در این متغییر عددی ۴ رقمی است که دو رقم سمت راست آن اعشار و دو رقم سمت چپ آن عدد اصلی است . به عنوان مثال ۴۵۲۳ یعنی ۴۵.۲۳ درجه سانتیگراد. در Network7 با رسیدن دما به ۷۰ درجه سانتیگراد که معادل ۷۰۰۰ می باشد ، موتور همزمان به مدت ۴۰ ثانیه مواد را با هم مخلوط می کند.

در Network8 با خاموش شدن Motor ابتدا مقدار ۰ در MD20 بارگذاری می شود که موجب خاموش شدن Heater می گردد و همزمان بیت موقت Set,Tomp-3 می گردد که این بیت در Network9 مجوز عملگرد دستور مقایسه $Cmpl <= 1$ را می دهد که اگر تشخیص دهد دمای بالا به زیر ۳۰ درجه سانتیگراد رسید ، شیر خروجی Valve out را باز می کند تا مواد تخلیه شوند . همچنین در Network5 ، چون از تیغه N.C آن استفاده شده ، تابع Move دیگر کار نمی کند و در حین تخلیه Heater را روشن نمی کند و همچنین در Network3 در صورت تخلیه مواد و کم شدن سطح مخزن به میزان کمتر از ۵% این بیت Reset شده و پروسه مجدداً تکرار می گردد.

در شکل زیر نمایی از تنظیم ورودی و خروجی ها در شبیه ساز را مشاهده می کنید :



۱۰-۱۱ عنوان پروژه : سیستم میانگین گیری وزن باسکول در شالیکوبی

شرح پروژه : بر روی صفحه bascule ، توسط بازوی پنوماتیک ، هر ۵ثانیه یک کیسه برنج قرار می گیرد که توسط سنسور Load Call وزن آن اندازه گیری می شود.

برنامه را به گونه ای طراحی کنید که با قرار گرفتن اولین کیسه برنج بر روی Bascule نمونه گیری شروع شود و هر ۵ ثانیه یک بار وزن اندازه گیری شده د ر DB قرار بگیرد. پس از ۵ نمونه گیری از وزن ، میانگین آن حساب شده و در متغییر MD80 قرار بگیرد. توجه داشته باشید که سیستم به صورت آماده باش می باشد و کلید Start ندارد و با قرارگیری اولین کیسه روی Bascule به صورت خودکار شروع به کار می کند.

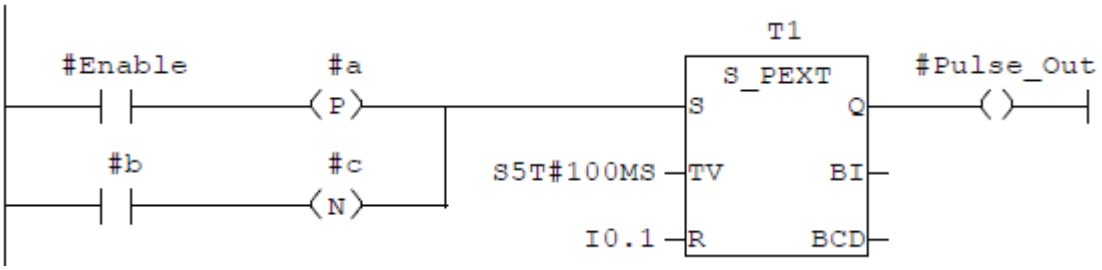
نمونه ای باسکول کفه ای

راه حل پروژه :

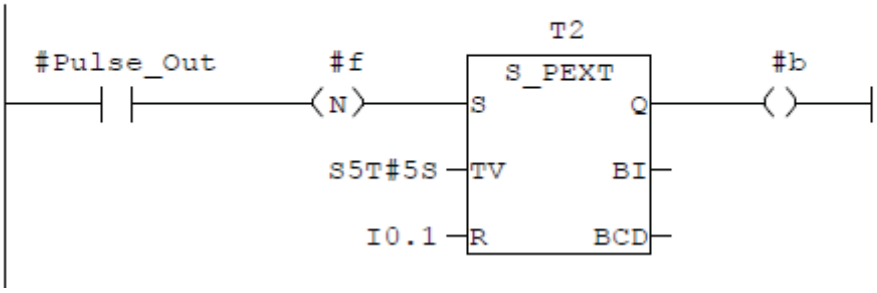
Block: DB1			
Address	Name	Type	Initial value
0.0		STRUCT	
+0.0	sample	REAL	0.000000e+000
+4.0	Sample_Count	WORD	W#16#0
=6.0		END_STRUCT	

Name	Data Type	Address	Initial Value	Comment
IN		0.0		
Enable	Bool	0.0	FALSE	
OUT		0.0		
Pulse_Out	Bool	2.0	FALSE	
IN_OUT		0.0		
STAT		0.0		
a	Bool	4.0	FALSE	
b	Bool	4.1	FALSE	
c	Bool	4.2	FALSE	
d	Bool	4.3	FALSE	
e	Bool	4.4	FALSE	
f	Bool	4.5	FALSE	
TEMP		0.0		

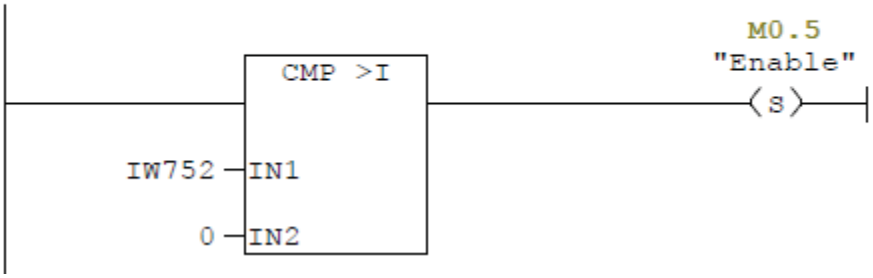
Network: 1 **FB1 Blink Function**



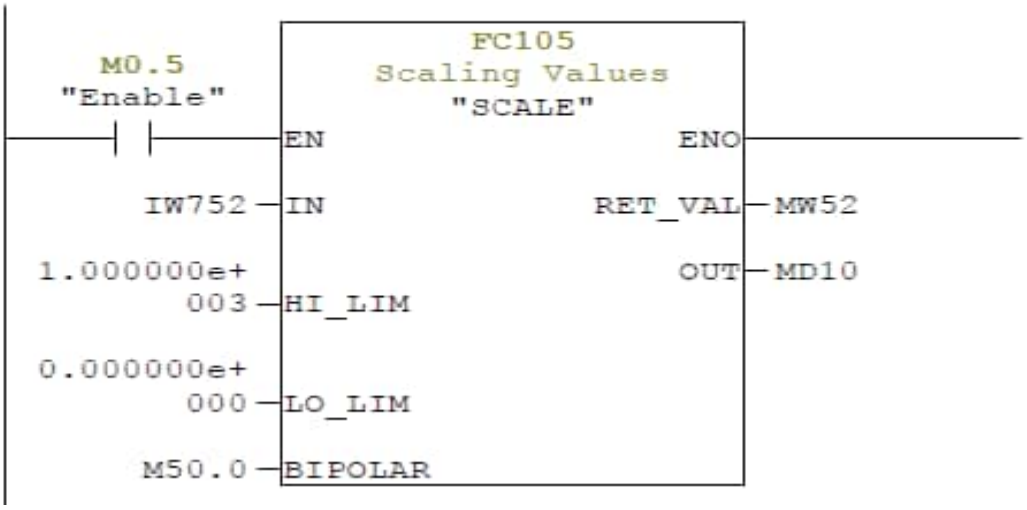
Network: 2



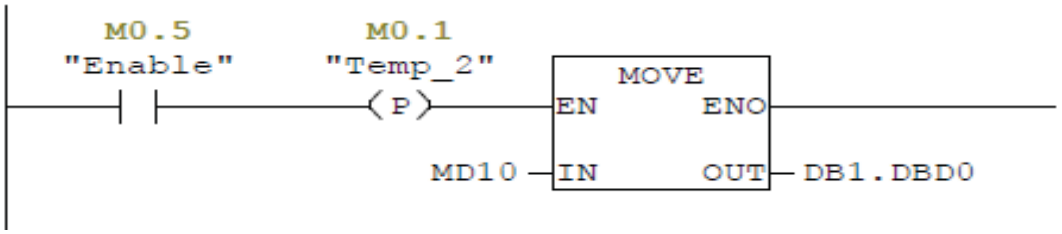
Network: 1 OB1 Average System Weight Weighbridge(Load cell)



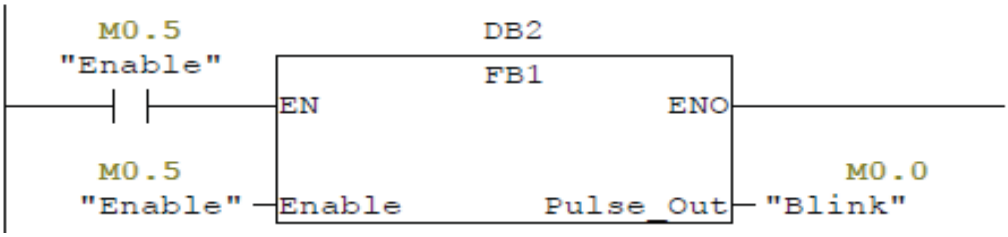
Network: 2

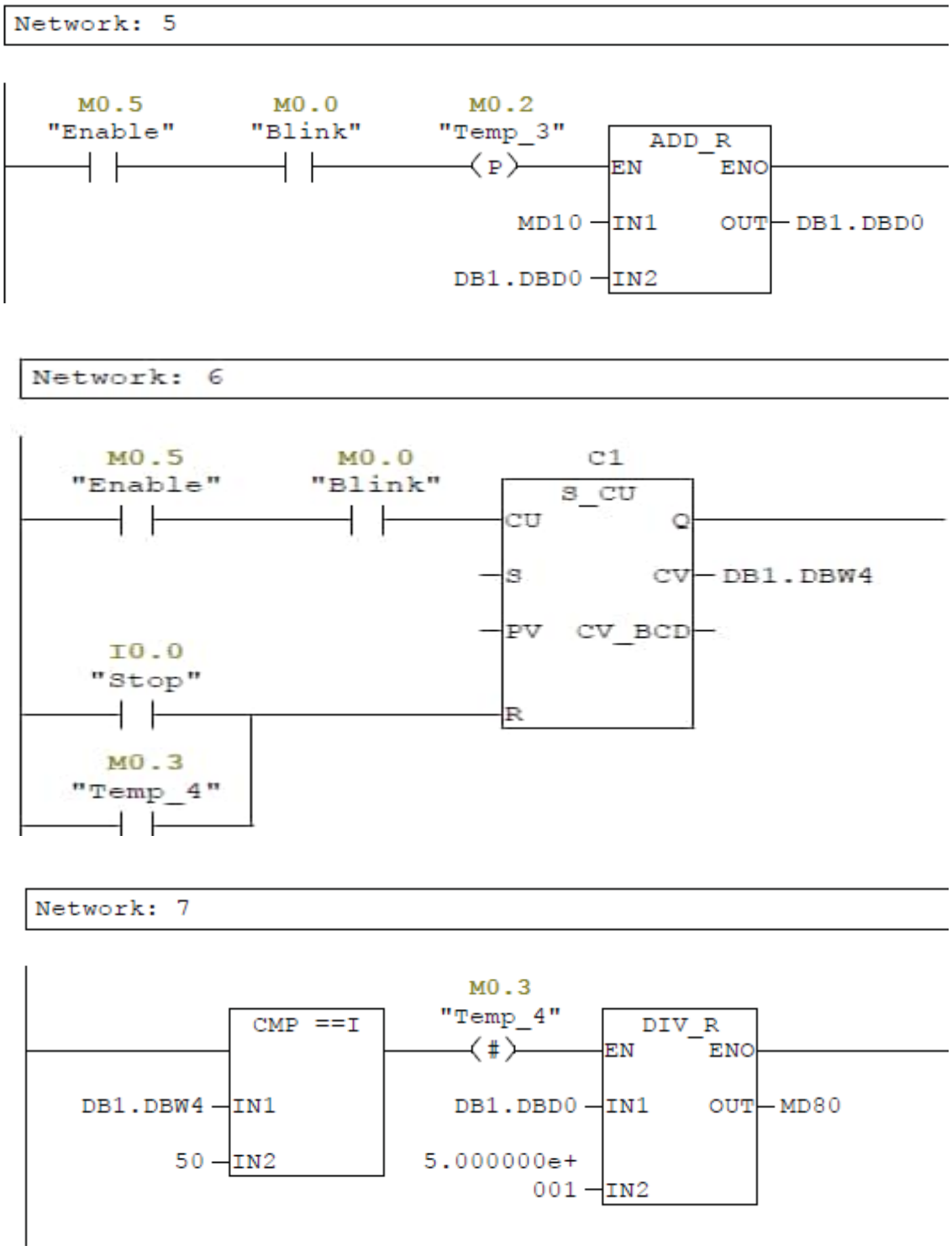


Network: 3



Network: 4 Blink Function





تشریح پروژه : در Network1 مقدار IW75 2 که وردی آنالوگی می باشد که به Loadcell متصل است با م قدار صفر مقایسه می شود و در صورتی که بزرگتر از آن باشد ، بیت Enable را set می کند ، یعنی اگر اولین کیسه برنج روی bascule قرار بگیرد ، سیستم فعال می گردد . در Network2 از تابع Scale برای مقیاس بندی مقدار آنالوگ اندازه گیری شده در رنج ۰ تا ۱۰۰۰ کیلوگرم می باشد . در Network3 با فعال شدن سیستم ، مقدار اولیه MD10 که مقیاس بندی

شده ورودی آنالوگ است در DB1 و آدرس DBDO قرار می گیرد که از نوع Real انتخاب شده است . در Network4 از تابع ساخته شده FB1 که در واقع یک نوساز با پریود زمانی ۱۰۰ میلی ثانیه ON و ۵ ثانیه Off که در شکل FB1 ، نقشه داخلی آنرا مشاهده می کنید استفاده شده است که با فعال شدن Enable متغییر Blink را روشن و خاموش می کند . در Network5 با هر بار پالس بیت Blink مقدار جاری MD10 با مقدار قبلی DB1.DBDO جمع شده و مجدداً در DB1.DBDO قرار می گیرد یعنی در واقع هر ۵ ثانیه مقدار جاری MD10 یا وزن باسکول با مقدار قبلی آن جمع می گردد همزمان در Network6 با هر پالس Blink شمارنده C1 یک واحد به مقدار DB1.DBW4 اضافه می کند . در Network7 درصورتی که شمارنده ۵۰ پالس را شمارش کرده باشد یعنی ۵۰ نمونه از وزن Bascule گرفته شده ، مجموع وزن بدست آمده را تقسیم بر ۵۰ کرده تا میانگین بدست آید و در MD80 قرار می گیرد ، همزمان شمارنده توسط Temp-4 ریست می شود.

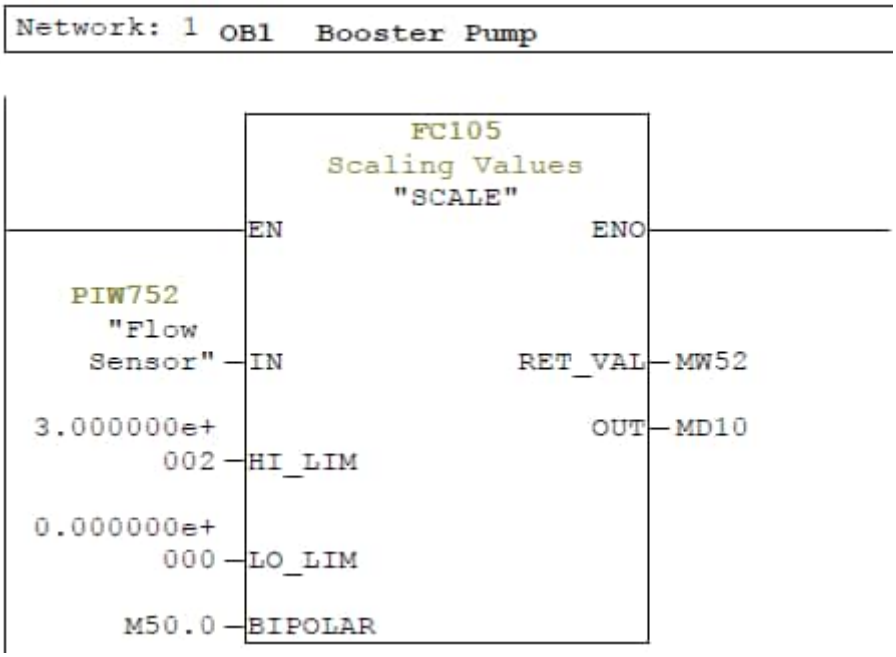
۱۰-۱۲ عنوان پروژه: راه اندازی نرم سه الکتروپمپ سیستم جامع بوستر پمپ

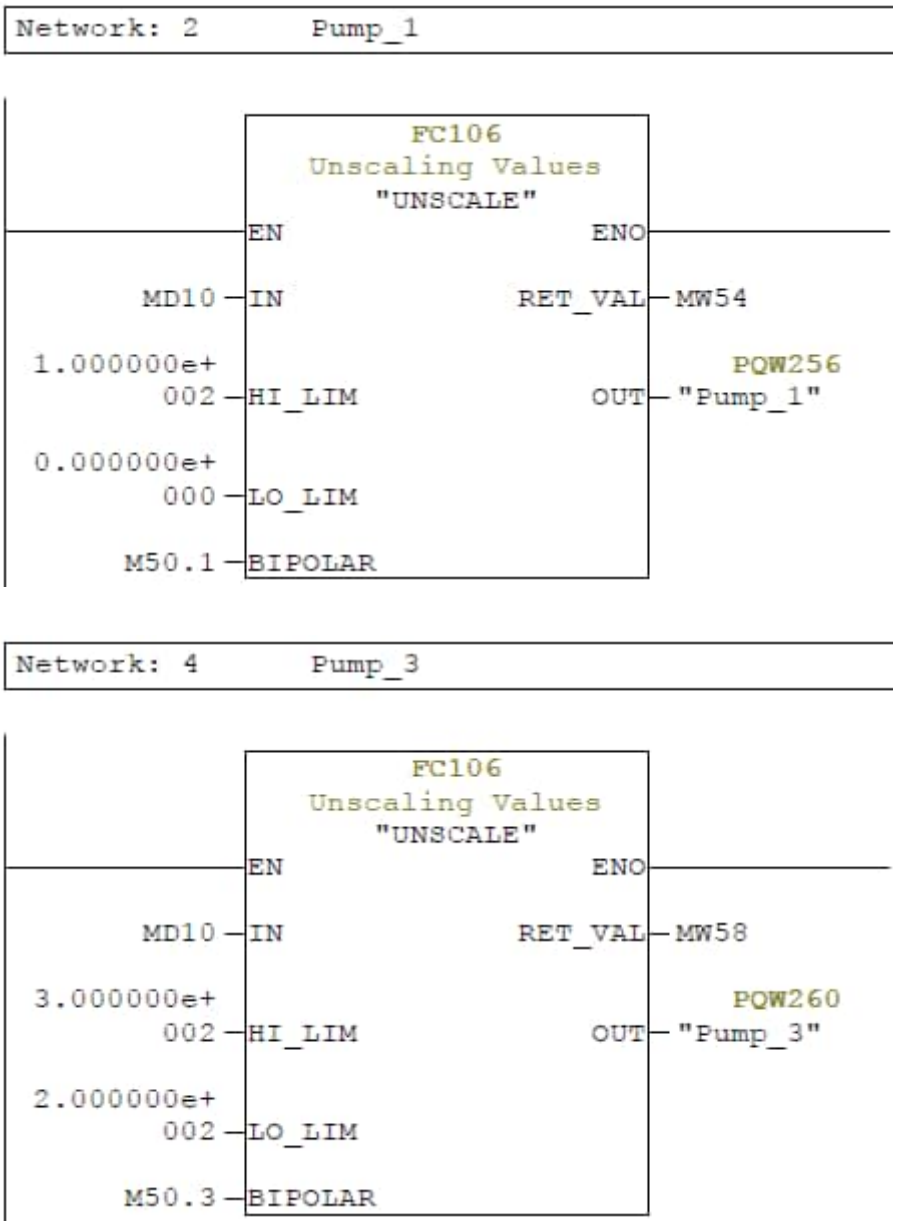
شرح پروژه: در گذشته اگر می خواستیم برای یک ساختمان مثلاً ۴۰ واحدی سیستم پمپ آب تعبیه کنیم . می بایست از یک پمپ با قدرت بسیار بالا استفاده شود که مشکلات بسیاری را به وجود می اورد اگر مشکل مصرف بالای انرژی را نادیده بگیریم فرض کنید این مجتمع مسکونی ، ویلا یی می باشد و زمان هایی از سال تنها چند واحد از آن پر است . حال اگر یکی از ساکنین شیرآب واحد را باز کنند ، دستگاه پمپ با آن قدرت بالا روشن می شود که علاوه بر آسیب جدی ای که به شبکه تاسیسات ساختمان وارد می کند آبی با فشار بسیار زیاد وارد شبکه می کند که موج ب گرم وسرد شدن متوالی آب و نیز قطع و وصل ممتد آب می گردد . حال فرض کنید چندین نفر از ساکنین ساختمان همزمان شیر آب را باز کنند و این بین چند نفر شیرآب را کم و زیاد کنید. این اتفاق باعث می شود جریان و دبی آب در طبقات مختلف مرتباً کم و زیاد شود .

به کمک بوستر پمپ (Booster Pump) می توان به جای یک الکترو پمپ توان بسیار بالا ، ازچند الکتروپمپ توان کم یا متوسط استفاده شود که دور آنها توسط Inverter کنترل می گردد. جریان دبی آب وارد شده به ساختمان توسط سنسور آنالوگ Flow Controler اندازه گیری می شود و متناس ب با آن جریانی بین $۴ - ۲۰mA$ به PLC می دهد . باید متناسب با جریان آب مصرفی سرعت موتورها یکی پس از دیگری اضافه شود یعنی اگر شخص تنها یک شیر آب را باز کرد موتور اول با دور کم شروع به پمپاژ آب کند اگر شیر بعدی باز شد دور موتور بیشتر شود حال اگر مصرف آن‌قدر زیاد شد که دور موتورپمپ اول به حداکثر خود رسید الکتروپمپ دوم با دور نرم به افزایش پمپاژ کمک کند . و الی آخر ... دقت شود که برای حل این پروژه نیاز به یک ورودی آنالوگ $۴ - ۲۰mA$.و ۳ خروجی آنالوگ 0-10v می باشد که به ورودی اینورتر وصل می شوند تا دور آنرا از ۰ تا ۱۰۰% تغییر دهد.

در شکل زیر نمونه ای از بوستر پمپ را مشاهده می کنید:

راه حل پروژه :

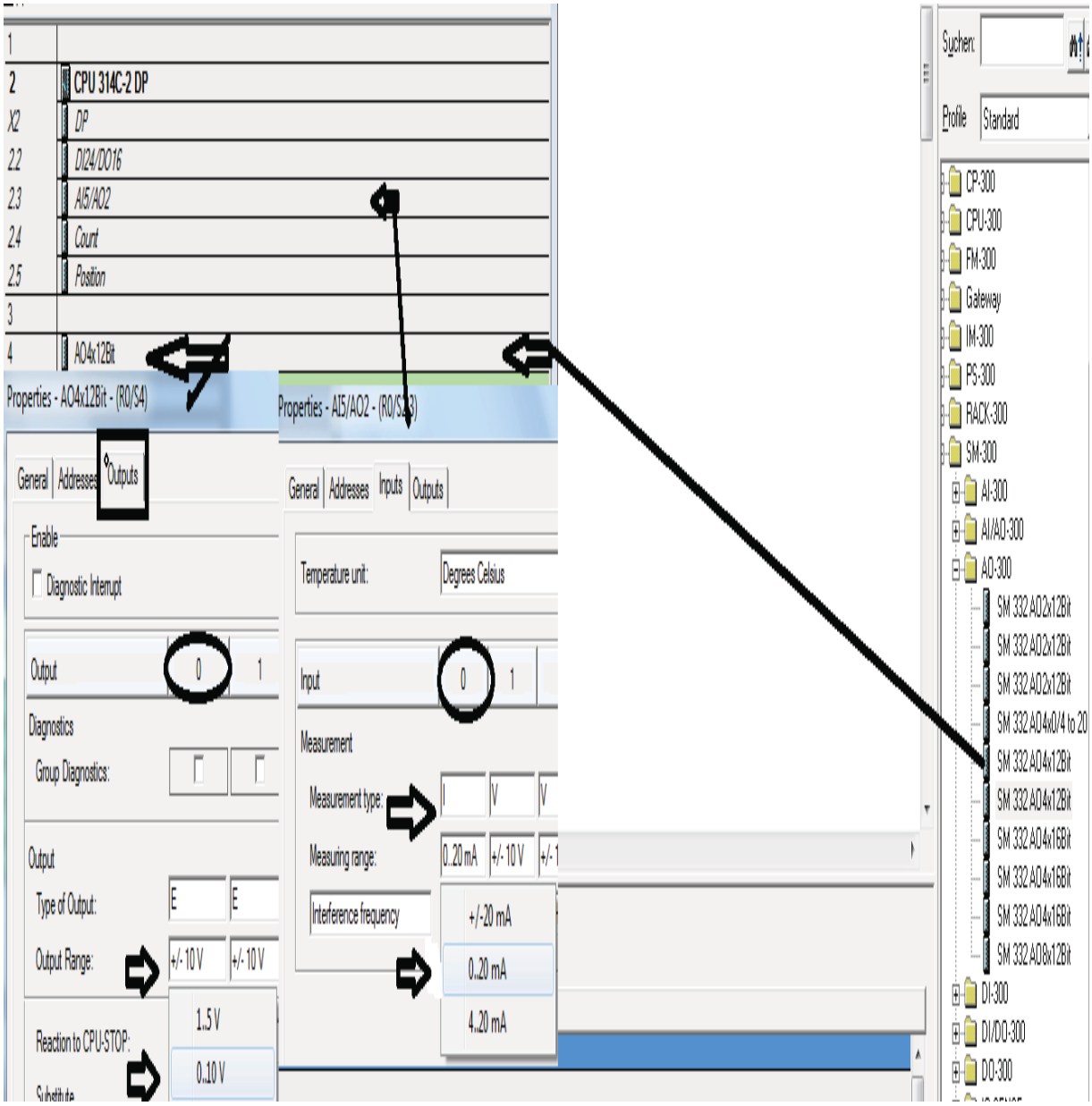




تشریح پروژه : در Network1 از تابع Scale استفاده شده است تا ورودی آنالوگ 4-20Ma را به بازه ۰ تا ۳۰۰ تبدیل کرده و در MD10 قرار دهد. علت انتخاب عدد ۳۰۰ برای HI-Lim این است که خروجی آنالوگ در بازه ۱۰۰-۰ درصد را در نظر گرفته ایم. در Network2 ورودی تابع Unscale خروجی تابع Scale می باشد که از ۰ تا ۳۰۰ تغییر می کند. حال اگر این ورودی از ۰ تا ۱۰۰ تغییر کند خروجی PQW256 از ۰ تا ۲۷۶۴۸ تغییر می کند که در واقع خروجی آنالوگی از ۰ تا ۱۰ ولت می باشد.

در Network3 تابع Unscale این بار بازه 100 تا ۲۰۰ را تشخیص داده و متناسب با آن که در واقع بازه ای ۱۰۰ تایی است خروجی PQW758 را در بازه ۰ تا ۱۰ ولت تغییر می دهد و همین اتفاق در Network4 نیز می افتد با این تفاوت

که بازه ۲۰۰ تا ۳۰۰ را تشخیص می دهد و در خروجی PQW260 اعمال می کند . دقت شود از آنجایی که CPU مورد استفاده ما تنها دارای ۲ خروجی آنالوگ می باشد ، مطابق شکل از کارت خروجی آنالوگ ۴ کاناله استفاده شده که نوع آن را ولتاژ و در رن ج ۰~۱۰V انتخاب می کنیم . حال اگر ورودی آنالوگ 4-20MA تغییر کند که خروجی A/D بازه ۰~۲۷۶۴۸ را ایجاد می کندمتناظربآن ورودی ۰تا۹۲۱۶که دواقع $۱/۳$ ورودی است تغییرکند خروجی - Pump-1 از ۰ تا ۱۰۰ درصد تغییر سرعت می دهد و به همین ترتیب هر $۱/۳$ ورودی که اضافه شود pump بعدی وارد مدار می شود تا زمانی که ورودی آنالوگ 20Ma را دریافت کندیعنی حداکثر دبی عبوری آب اتفاق افتاده است . در این حالت هر سه موتور با حداکثر دور نام خود شروع به پمپاژ آب می کنند .



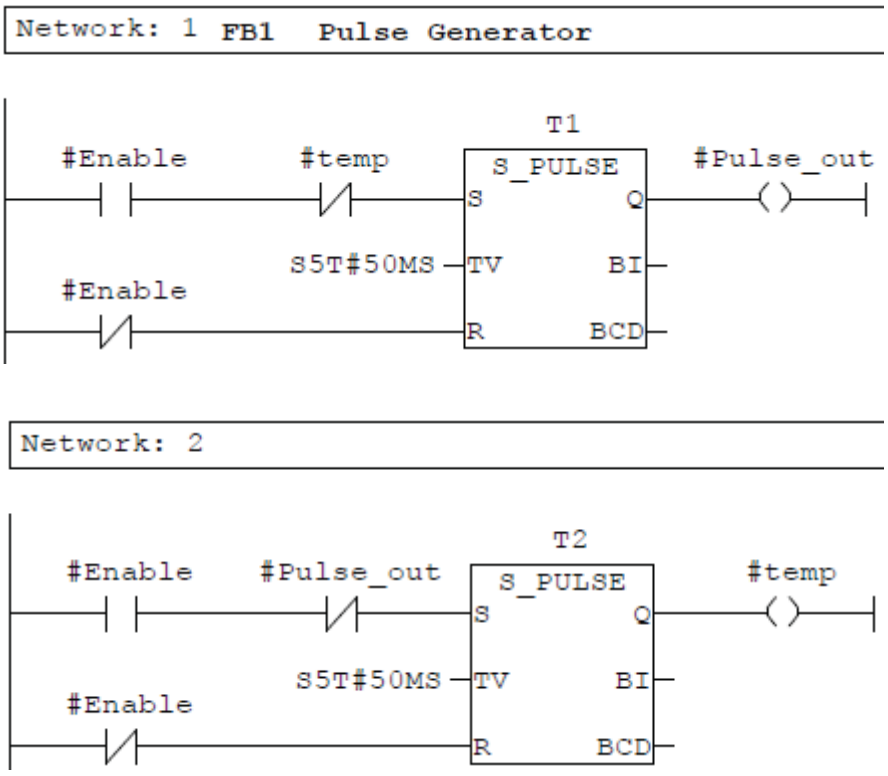
۱۳-۱۰عنوان پروژه : کنترل خروجی آنالوگ به صورت Ramp

شرح پروژه: برنامه ای بنویسید که با فشردن کلید start خروجی آنالوگ PQW754 که به یک شیربرقی آنالوگ (Control Valve) متصل است را در عرض ۱۰ ثانیه از حداقل مقدار تا حداکثر مقدار تغییر دهد و از آنجایی که شیربرقی مورد نظر ، از نوع ۰ تا ۱۰ ولت است ، طوری طراحی شود که در عرض ۱۰ ثانیه ولتاژ خروجی آنالوگ PLC از ۰به ۱۰ ولت برسد که این وضعیت به صورت پیوسته و با حداقل پله باشد .

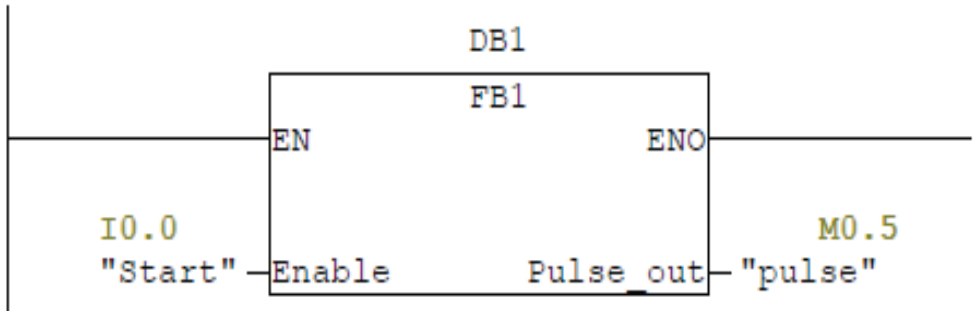
FB1

Name	Data Type	Address	Initial Value	Comment
IN		0.0		
Enable	Bool	0.0	FALSE	
OUT		0.0		
Pulse_out	Bool	2.0	FALSE	
IN_OUT		0.0		
STAT		0.0		
temp	Bool	4.0	FALSE	
TEMP		0.0		

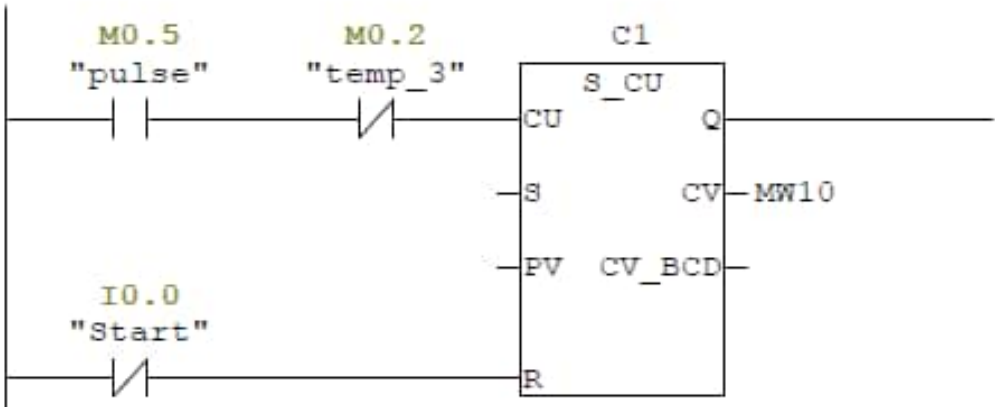
راه حل پروژه :



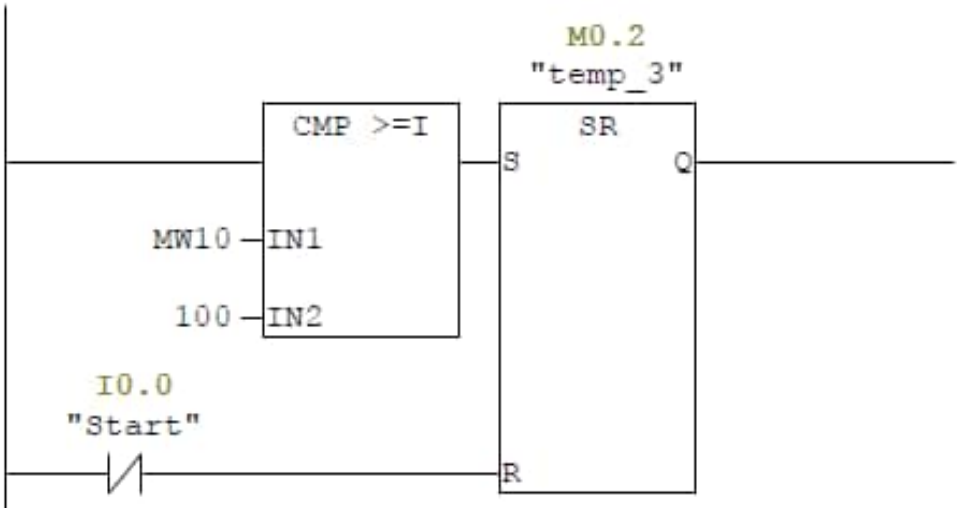
Network: 1 OB1 Analog Output (RAMP)

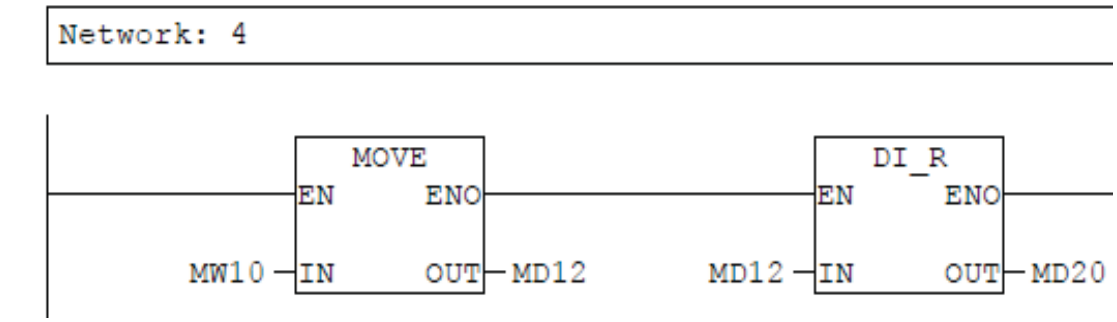


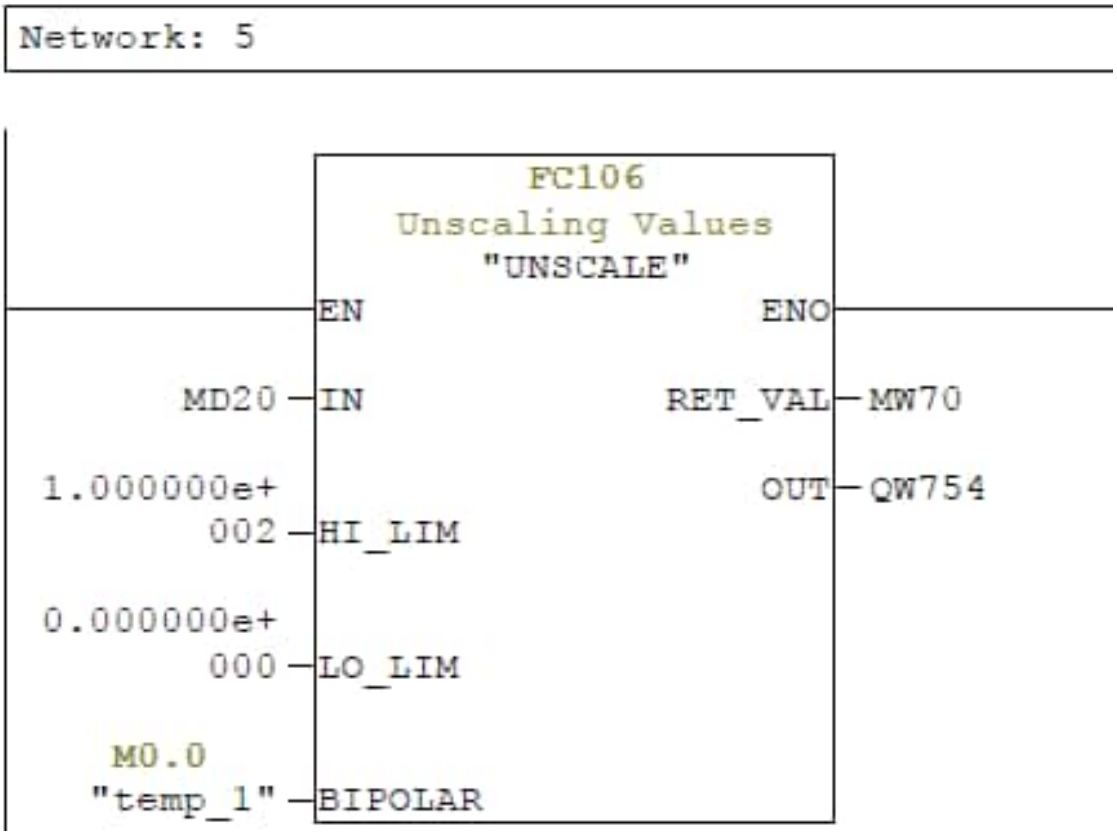
Network: 2



Network: 3







تشریح پروژه : خروجی آنالوگ ۰ تا ۱۰ولت زمانی ایجاد می شود که در آن آدرس مثلاً PQW754 عددی بین ۰ تا ۲۷۶۴۸ تغییر کند. از آنجایی که کار با این عدد کمی مشکل است به کمک تابع Unscale ورودی را در بازه ۰ تا ۱۰۰ قرار می دهیم. حال اگر عدد ورودی بین ۰ تا ۱۰۰ تغییر کند خروجی ما نیز بین ۰ تا ۱۰۰ ولت تغییر می کند. خوب با کمی دقت به این رابطه می رسیم که بین ۰ تا ۱۰۰ و ۰ تا ۱۰ رابطه $\frac{۱}{۱۰}$ بر قرار است یعنی اگر بتوانیم هرثانیه ، ۱۰ واحد به مقدار ورودی Unscale اضافه کنیم در طی ۱۰ ثانیه این مقدار به ۱۰۰ می رسد.

برای این کار ابتدا در FB1 مدار مولد بالس (Pulse Generator) را با پریود زمانی ۵۰ میلی ثانیه روشن و ۵۰ میلی ثانیه خاموش ایجاد کنیم تا در هر ثانیه تقریباً ۱۰ پالس ایجاد کند. در OB1 پس از آنکه در Network1 بلوک FB1 را فراخوانی کردیم. ورودی ان را به Start نسبت داده و خروجی آن رابه ورودی شمارنده C1 متصل می کنیم . حال با هر بار قطع و وصل خروجی Pulse شمارنده یک واحد به مقدار MW10 اضافه می کند.

در Network4 این مقدار پس ازتبدیل به MD وسپس به Real توسط مبدل DI-R در Network5 به ورودی تابع FC106 نسبت داده می شود . با فشردن کلید Start شمارنده شروع به کار می کند و از ۰ تا ۱۰۰ به صورت Real به ورودی Unscale می دهد و متناسب با آن نیز خروجی PQW754 از ۰ تا ۲۷۶۴۸ تغییر می کند که نتیجه آن تغییر خروجی ۰ تا ۱۰ ولت می باشد . Network3 نیز مقدار جاری شمارنده به کمک تابع $Cmp >= 1$ با مقدار ۱۰۰ مقایسه می شود و اگر به مقدار رسید باعث می شود در Network2 شمارنده دیگر پالس ورودی pulse را دریافت نکند زیرا تیغه Temp-3,N.C سر راه آن قرار دارد یعنی پس از ۱۰ ثانیه مقدار ولتاژ خروجی ما روی ۱۰ ولت ثابت می ماند تا زمانی که ورودی Start قطع شود . با این اتفاق در Network2 هم شمارنده C1 ریست می شود و هم در Network3 بلوک SR ، خروجی Temp-3 را ریست می کند و خروجی صفر می شود تا زمانی که مجدداً ورودی فعال گردد .

منابع:

کتاب مجموعه پروژه های PLC : مهندس مبین محسن زاده

جزوه آموزشی کارور PLC آموزشگاه برق و اتوماسیون صنعتی ادیسون : مهندس مجید نقی زاده

جزوه آموزش PLC S7 300 شرکت توان قدرت

جزوه بررسی سخت افزار PLC S7 300 : منوچهر مومنی

جزوه پروژه های کاربردی PLC پتروشیمی خارک : منوچهر مومنی

زمین را گر شوی مالک طمع به آسمان داری

دم مردن همه بینند نه این داری نه آن داری

موفق و سر بلند باشید

منوچهر مومنی

۱۳۹۲/۳/۸