فصل سوم

برنامه نویسی

در این فصل آشنا خواهید شد با ۰۰۰۰

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:

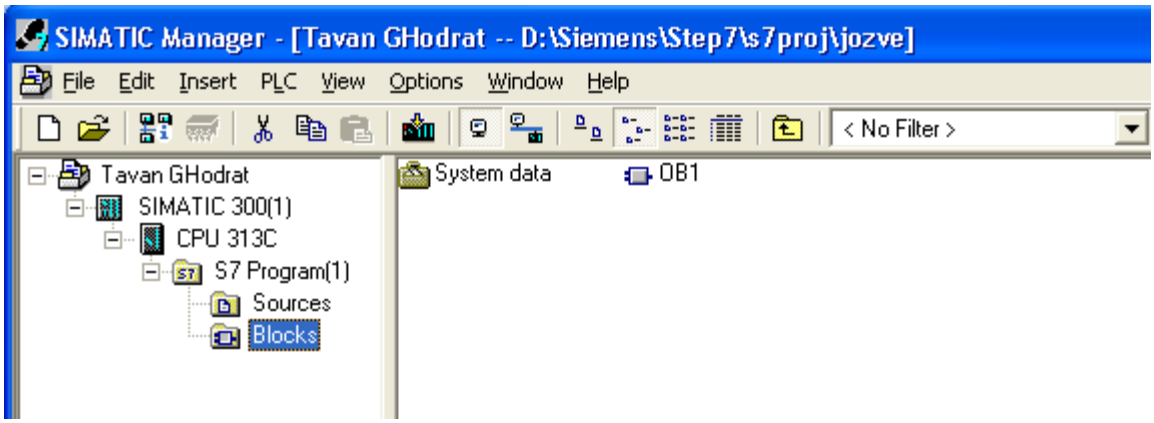
۱- **بلوک های داده اشتراکی**(shared Data Blocks): این بلوک ها از نوع اشتراکی هستند یعنی از سایر بلوک ها می توان به آن دسترسی داشت و اطلاعات آن را خواند یا تغییر داد.

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.

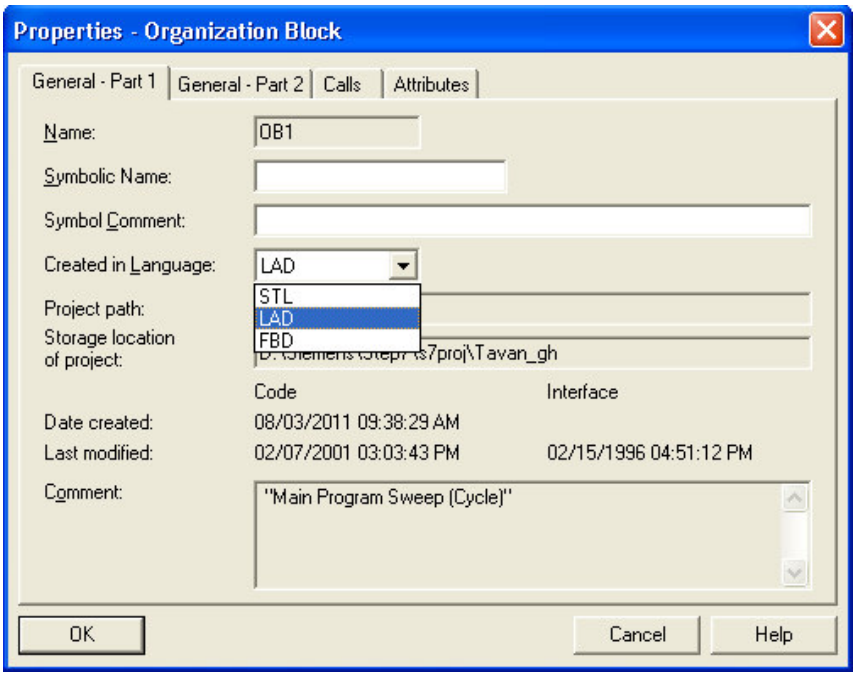
نحوه عملکرد و کار با DB ها در فصل های آینده به طور کامل توضیح داده خواهد شد.

آغاز برنامه نویسی خطی

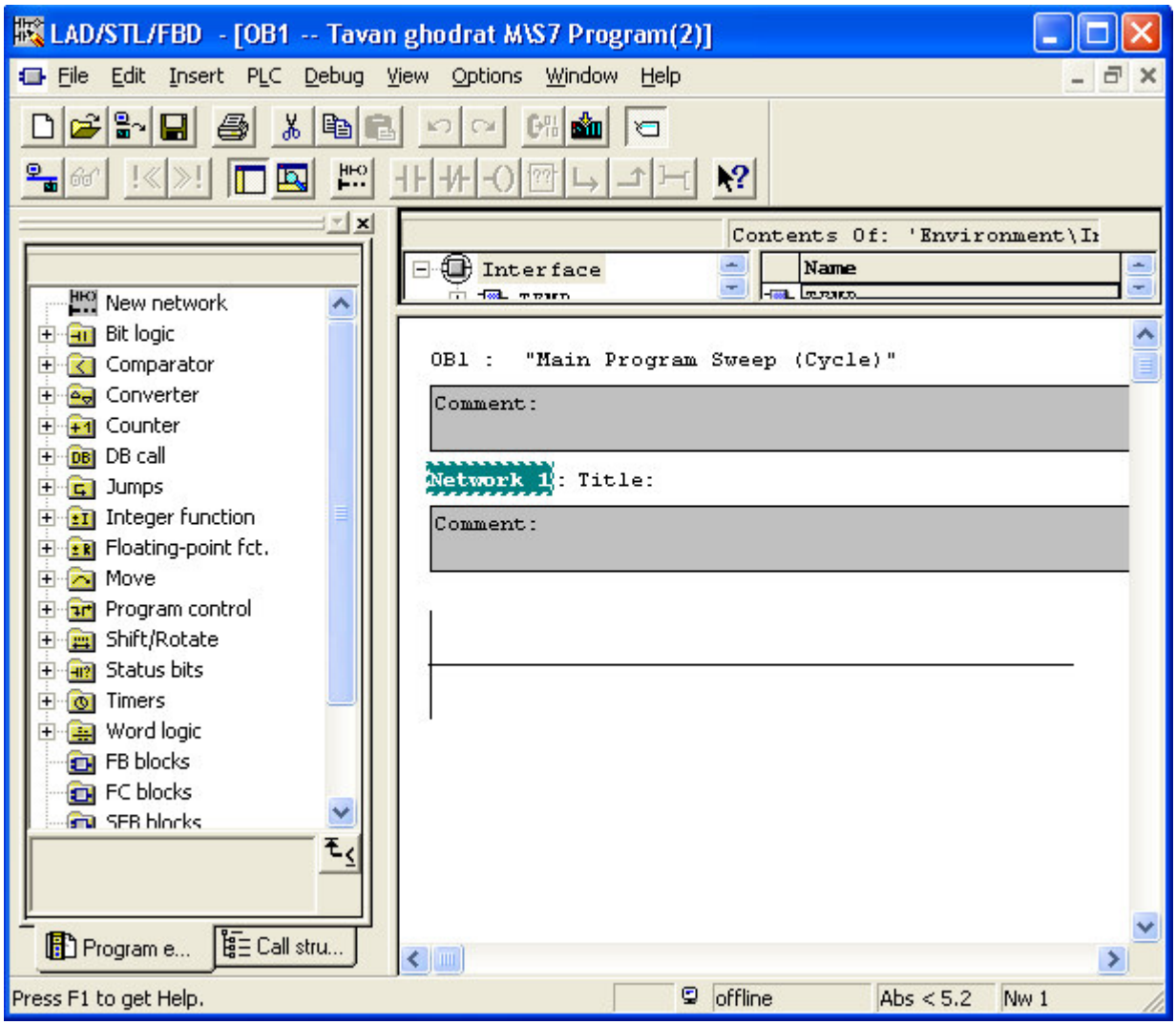
بعد از پیکر بندی سخت افزار پوشه های جدیدی به زیر نام پروژه شما اضافه می شود که نشان دهنده نوع CPU و نام پروژه های آن می باشد که در زیر نشان داده شده است.



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.

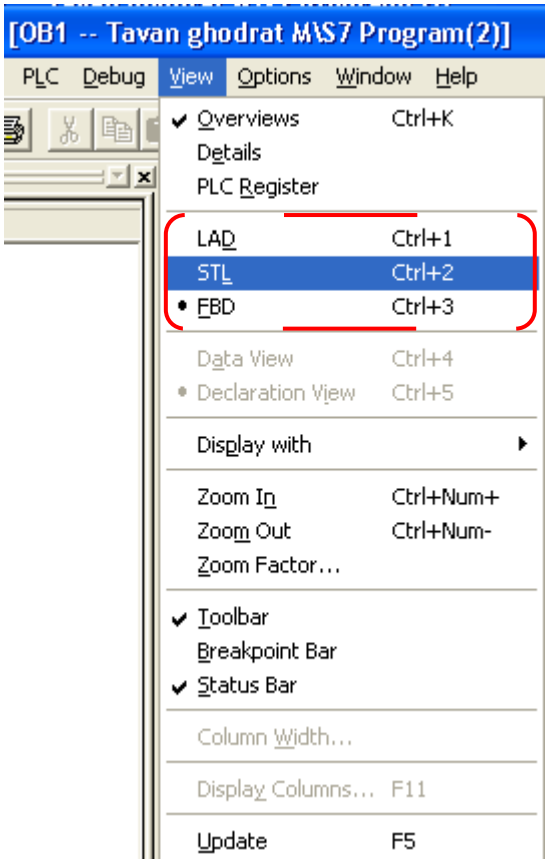
آشنایی با توابع :

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 Logic**: 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 آگاهی قبلی از نحوه عملکرد آنها از الزامات کار با PLC ها است.

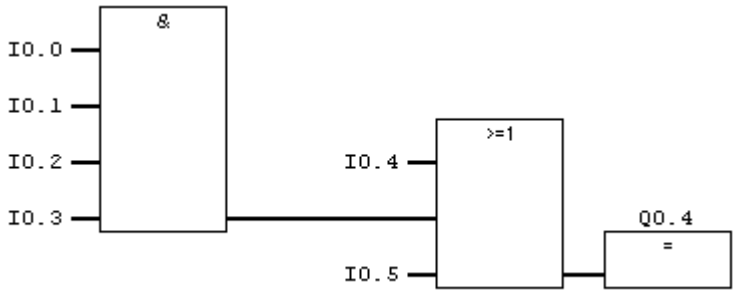
در این پوشه به توضیح برخی از موارد پر کاربرد و شاید هم نا آشنا پرداخته می شود.

- **اضافه کردن 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.



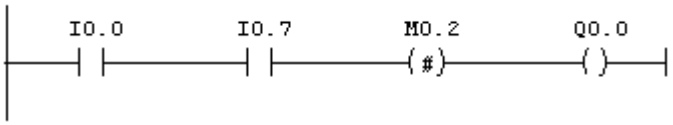
نکته: **اگر پایه ای نیاز نبود می توان آنرا حذف کرد برای این کار نیز کافیست** برروی پایه مورد نظر کلیک کرده و دکمه Delete را فشار دهید.

- **منفی کردن:** -|/|- 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.

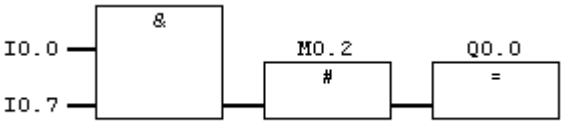
مثال: برنامه ای بنویسید که با تحریک کلید I0.0 و کلید I0.7 خروجی Q0.1 روشن شود و بیت حافظه ای به آدرس M0.2 نیز ست شود.

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

زبان LAD



زبان FBD

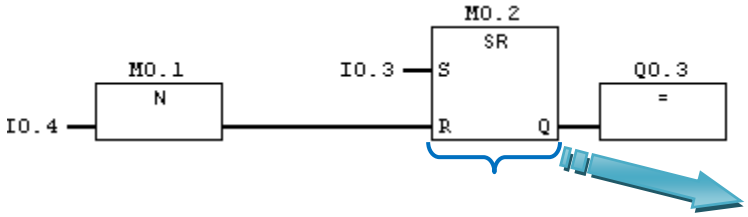


- **تشخیص دهنده های لبه**: --[N]- --[P]-- این بلوک ها دارای یک ورودی و یک خروجی می باشند که نسبت به تغییر لبه واکنش نشان می دهند. بلوکی که با نام [P] مشخص شده است نسبت به لبه بالا رونده و بلوک [N] نسبت به لبه پائین رونده حساس بوده و در صورت مشاهده لبه های مذکور در خروجی این بلوک یک پالس به مدت سیکل زمانی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

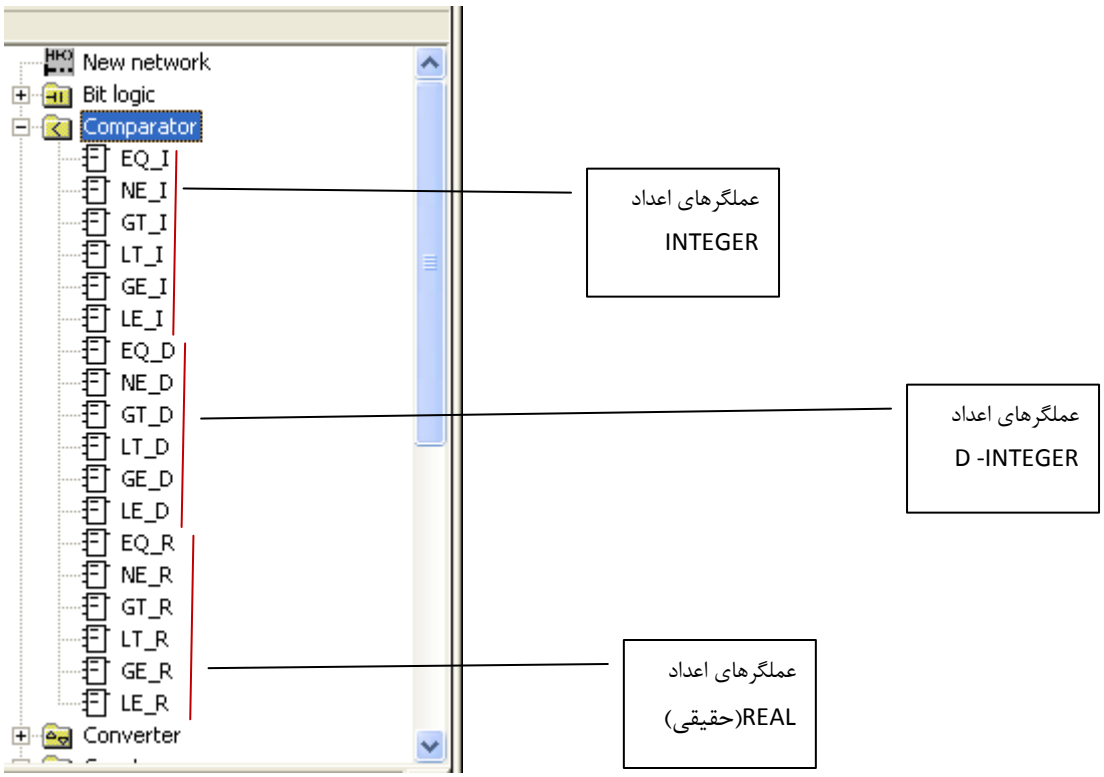
پاسخ: همان طور که در شکل دیده می شود زمانیکه کلید I0.3 تحریک می گردد خروجی روشن می شود ولی اگر کلید I0.4 را فشار دهید اتفاقی نمی افتد چون حساس به لبه پائین رونده می باشد.



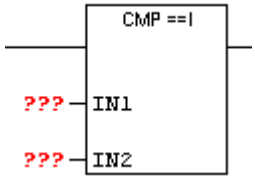
بلوک های SR و RS: این بلوک ها از پر کاربردترین بلوک ها در زبان برنامه نویسی می باشند. همانطور که می دانید با فعال شدن لحظه ای پایه S خروجی این بلوک فعال می شود و با فعال شدن لحظه ای پایه R خروجی ریست می گردد. عملکرد این دو بلوک از این نظر کاملاً شبیه به هم می باشد ولی در یک حالت با هم متفاوت می باشند.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

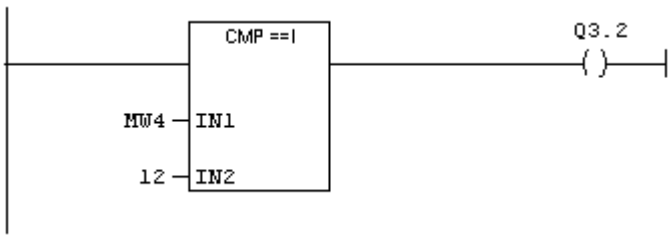


بلوک تساوی EQ_I : ، همان طور که می بینید این بلوک دارای دو ورودی از نوع عدد صحیح می باشد که هر گاه این دو عدد با هم برابر شوند خروجی این بلوک فعال می گردد. علامت سوال قرمز رنگ در ورودی این بلوک نشانگر این است که باید آدرس عدد صحیح یا خود عدد صحیح را وارد نمایید.

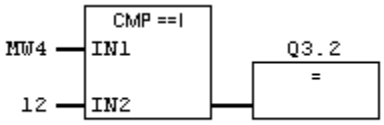


مثال : هرگاه مقدار MW4 برابر با ۱۲ شد خروجی Q 3.2 فعال شود.

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.

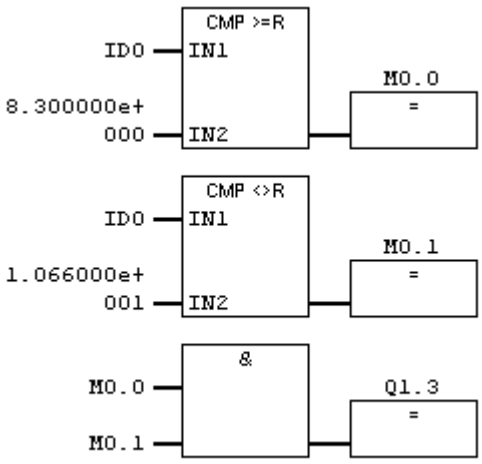
LE_I **بلوک کوچکتر مساوی:** عملکرد این بلوک تقریباً مشابه بلوک LT_I می باشد با این تفاوت که در حالت تساوی دو ورودی هم خروجی آن فعال باقی می ماند.

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.

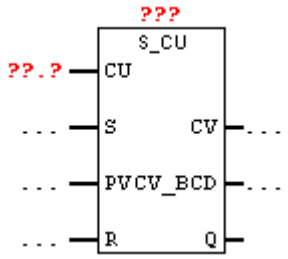
پاسخ: همان طور که در شکل دیده می شود چون اعداد از نوع اعشاری هستند از مقایسه کننده های REAL استفاده شده است. از یک بلوک بزرگ تر مساوی و یک بلوک نامساوی استفاده شده است.



✓ **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



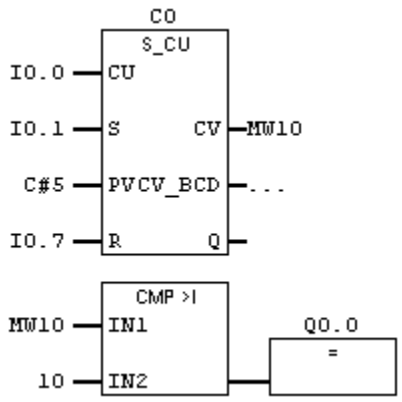
برای استفاده از این بلوک نیازی نیست که همه پایه های آن را متصل نمائید، فقط نامگذاری پایه هایی که دارای علامت سوال با رنگ قرمز می باشند **الزامی** است.

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:



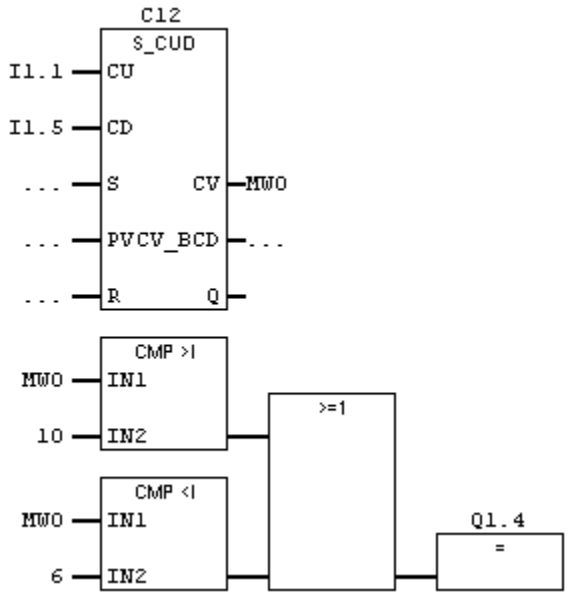
همان طور که دیده می شود ورودیI0.0 به ورودی کانتر وصل شده است خروجی نیز چون به فرم word می باشد در یک MW ذخیره شده و در مرحله بعد با 10 مقایسه می شود. خروجی Q0.0 نیز زمانی فعال می گردد که خروجی کانتر برابر با 10 گردد.

- **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.

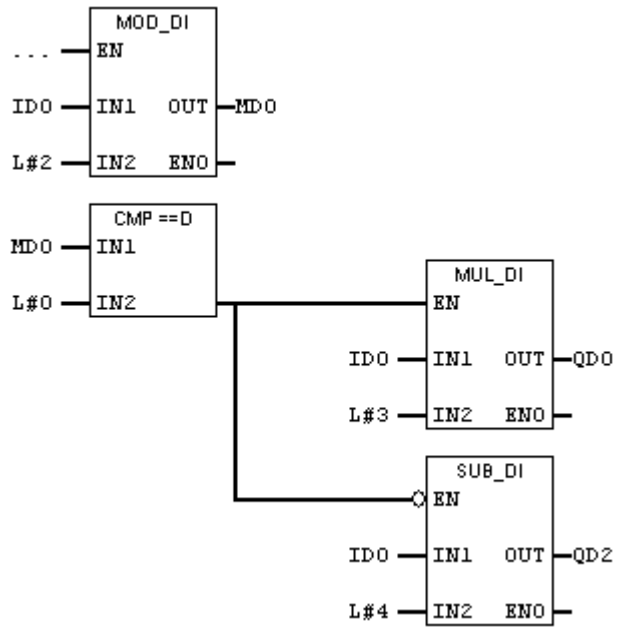
برای اعداد DINT نیز همین توابع با همین ویژگی ها در نظر گرفته شده است که از بازنویسی آن خودداری می کنیم فقط دقت شود که برای وارد کردن اعداد در توابع DINT باید از فرمت (عدد DINT مورد نظر L#) استفاده کنید.

- **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.

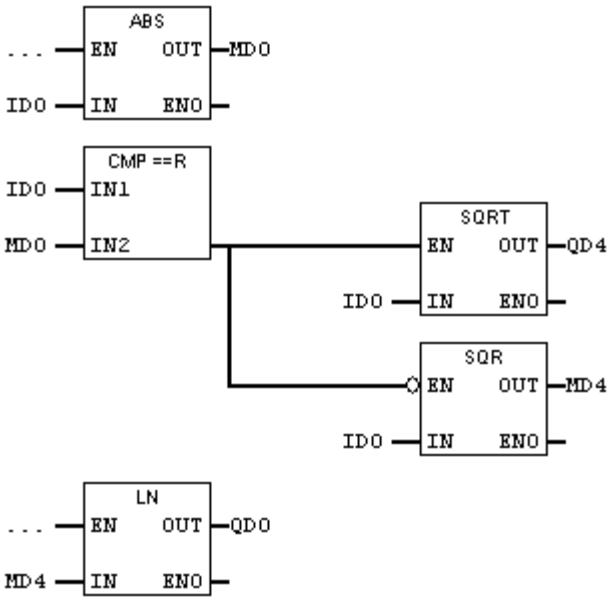


✓ **توابع اعداد حقیقی** (Floating point real) **:** این توابع برای کار کردن با اعداد حقیقی و اعشاری می باشد و نسبت به اعداد INT و DINT تعداد بیشتری را شامل می شوند.

- **توابع** DIV_R , MUL_R , SUB_R , ADD_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.

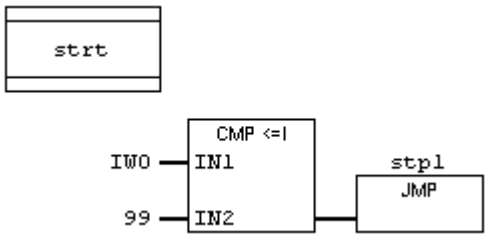
LABLE: این کلمه به معنی برچسب می باشد که کاربر آن را در NETWORK مورد نظر خود وارد می نماید و نامی دلخواه به آن اطلاق می کند و هنگامیکه دستور پرش فعال شود با توجه به برچسب دستور مورد نظر ، NETWORK

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.

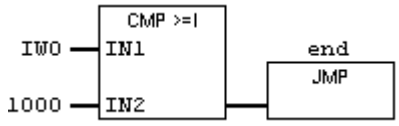
در استفاده از دستورات JUMP باید دقت نمود که عمل پرش هم می تواند به سمت جلو صورت گیرد و عم به سمت عقب. در پرش به جلو دستورات بین محل صدور فرمان و برچسب اجرا نخواهند شد ولی در پرش به عقب باید دقت نمود که عمل پرش به عقب در همان سیکل زمانی اجرای برنامه در حال اجرا می باشد. اگر تعداد دفعات پرش به عقب باعث 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.

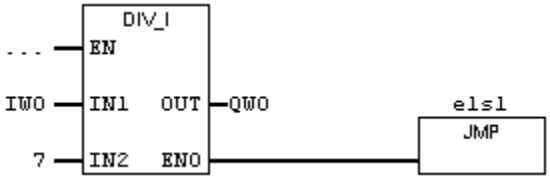
پاسخ: الگوریتم های مختلفی برای پاسخ به این مثال وجود دارد که در اینجا یکی از این برنامه های نوشته شده با دستورات JUMP نشان داده خواهد شد.

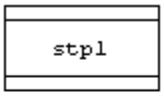


روند الگوریتم به این صورت است که عدد دریافتی از ورودی با عدد 99 مقایسه می شود اگر کوچکتر مساوی این عدد باشد پس یا دو رقمی است یا تک رقمی. برای بررسی آن باید به NETWORK با برچسب stp1 رفت در غیر این صورت NETWORK بعدی اجرا خواهد شد.

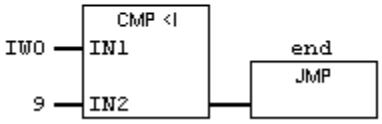


در این NETWORK عدد ورودی با مقدار 1000 مقایسه می شود اگر بزرگتر از آن باشد پس عدد 4 رقمی است و به برچسب end می پرد در غیر این صورت عدد مورد نظر 3 رقمی است و NETWORK بعدی اجرا خواهد شد و پس از پایان عملیات به برچسب els1 می پرد.

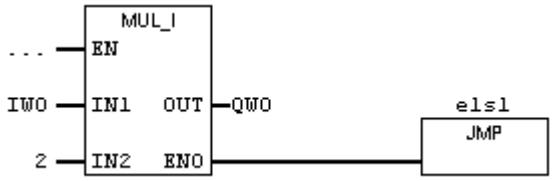


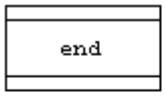


برچسب stp1 در این NETWORK آورده شده است که عدد کوچکتر از 99 را بررسی می نماید که آیا یک رقمی است یا دو رقمی.

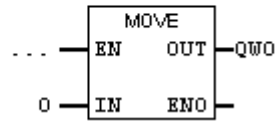


اگر عدد یک رقمی باشد به همان برچسبی می پرد که عدد 4 رقمی نیز به همان می پرد ولی اگر عدد 2 رقمی باشد Network بعدی اجرا خواهد شد و عدد در 2 ضرب می گردد ودر پایان نیز به بر چسب els1 می پرد.





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.



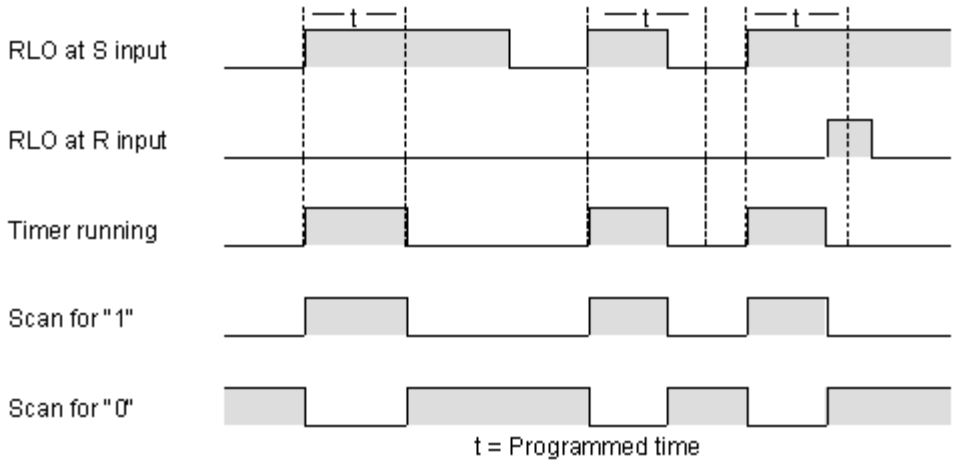
در Network بعد نیز برچسب els1 قرار داده شده است که به زبان stl در آن برنامه ای نوشته شده است به این مضمون که کاری انجام ندهد.(NOP:مخفف No Operation می باشد) در این Network کاری انجام نمی شود و هدف از گنجاندن آن جلوگیری از برش به عقب می باشد. با ورود به این Network روند برنامه در همان سیکل زمانی مورد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

```
els1: NOP   0
```

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

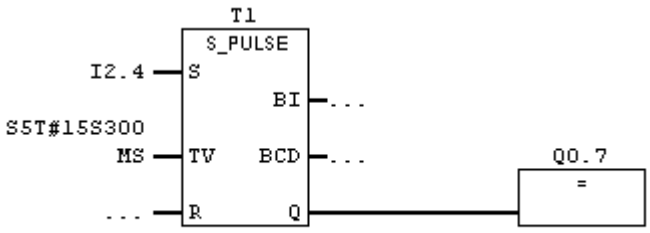
تایمر S_PULSE: این تایمر دارای سه ورودی و سه خروجی است. همان طور که از نمودار زمانی این تایمر مشخص است با فعال شدن پایه S به طور دائم تایمر شروع به زمان گیری می کند و به اندازه زمان تنظیمی در پایه TV (Time value) خروجی دیجیتال آن فعال است و پس از سپری شدن زمان، خروجی غیر فعال می گردد. اگر پایه S در وسط 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 نظرs5t#) 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.

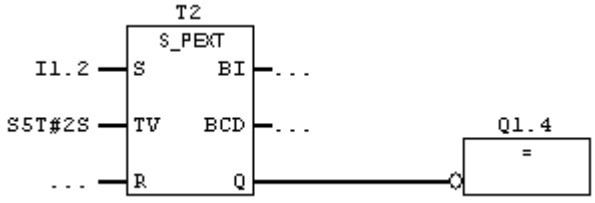
پاسخ: برای انجام این کار از یک تایمر S_Pulse استفاده شده است.



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:



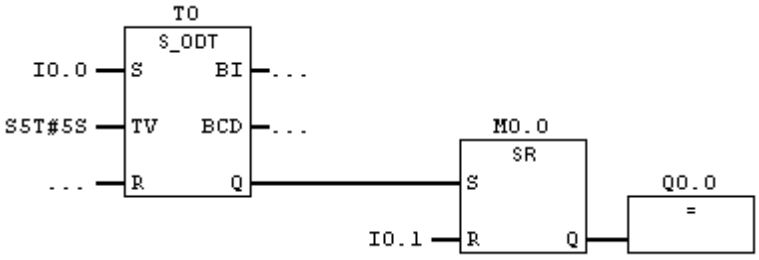
با عبور جسم تایمر روشن می شود و چون از گیت NOT استفاده شده است لامپ مربوطه خاموش می باشد. اگر فاصله بین دو جسم بیشتر از زمان تنظیمی باشد تایمر خاموش شده و خروجی لامپ روشن می گردد.

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.



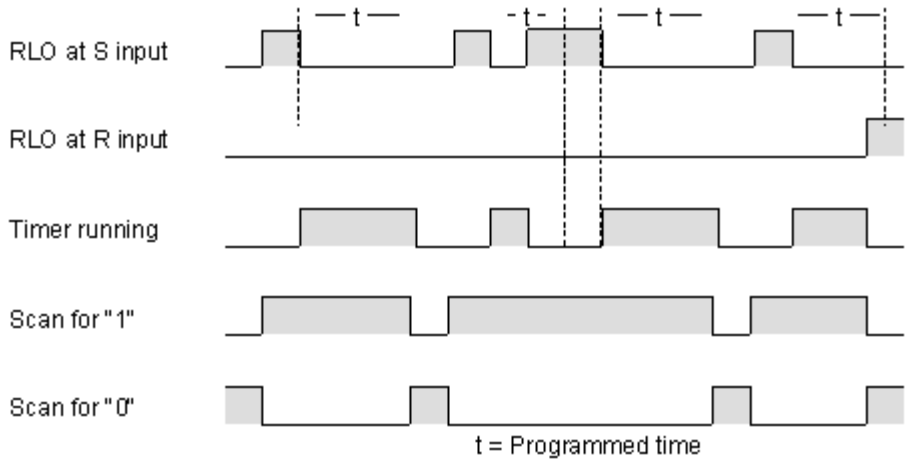
تایمر تأخیر در وصل پایدار (S_ODTS):

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 :

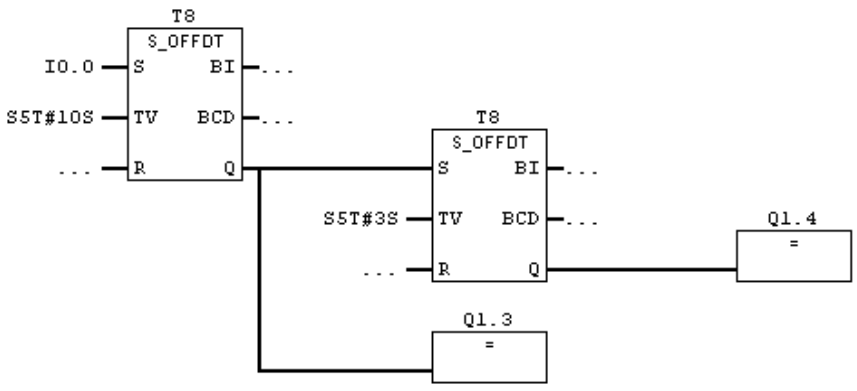
این تایمر حساس به لبه پائین رونده است یعنی زمانی که در ورودی S این بلوک لبه پائین رونده قرار بگیرد تایمر شروع به زمان گیری می کند و پس از سپری شدن زمان تنظیمی در پایه TV خروجی دیجیتال غیر فعال می گردد. با

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:

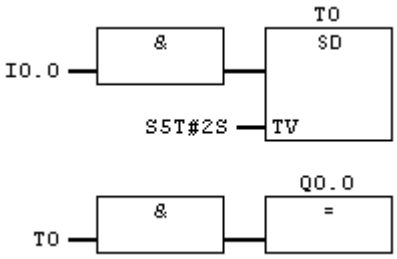


برای نوشتن این برنامه از دو تایمر تأخیر در قطع استفاده شده است. با لبه پائین رونده I1.1 تایمر T10 پس از 10ثانیه خاموش می شود و خروجی

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.

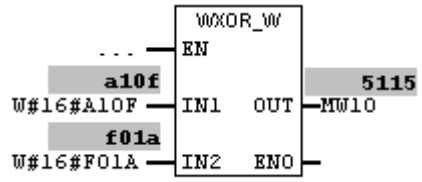
✓ **عملیات منطقی بر کلمات:** word logic

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.

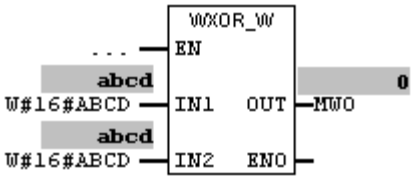
مثال : عدد A10F و عدد F01A را با هم XOR نمائید و عدد حاصل را در MW10 ذخیره نمائید.

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.

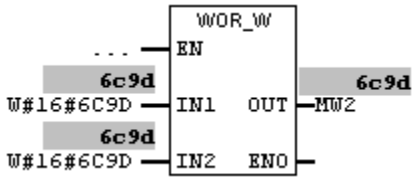


مثال : یک عدد به دلخواه انتخاب نموده و آن عدد را با خودش XOR نمائید حاصل چه عددی است؟



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

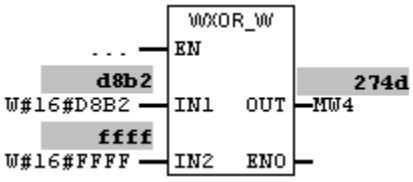
مثال : عددی به دلخواه انتخاب نمائید عدد را با خودش AND یا OR نمائید حاصل چیست؟



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.

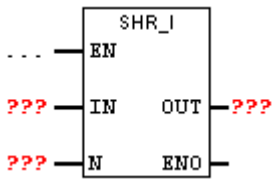
تمامی موارد فوق برای توابع DW نیز برقرارست با این تفاوت که اعداد در این دسته 4 بایت را اشغال می نمایند.

✓ **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.

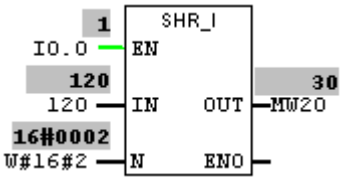
پایه IN عدد ورودی را دریافت می کند که این عدد باید به فرم عدد INT یا DINT باشد .

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.



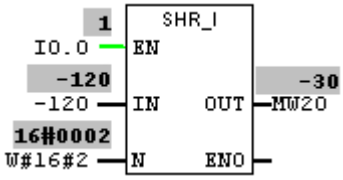
پاسخ : همانطور که می بینید به ازای هر شیفت عدد ورودی بر 2 تقسیم می شود و چون دو بار شیفت داده شده است عدد نهایی بر 4 تقسیم شده است.

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: عملیات شیفت به راست و شیفت به چپ توسط این دو بلوک صورت می گیرد. شکل ظاهری و وه عملکرد بلوک ها مشابه نمونه های قبلی می باشد با این تفاوت که فرمت اعداد ورودی و خروجی متفاوت می باشد

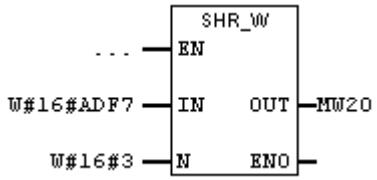
نح

پس از عملیات شیفت بیت های خالی با عدد صفر پر می شوند

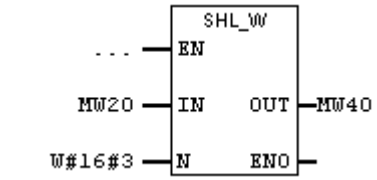
عدد ADF7 را پس از 3 مرحله شیفت دادن به راست به اندازه 3 مرتبه به اعداد ~~مثبت~~ و ~~منفی~~ بودن ~~به اعداد~~ در ~~چپ~~ شیفت ~~دهید~~ آیا عدد نهایی با عدد اولیه یکسان است؟

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

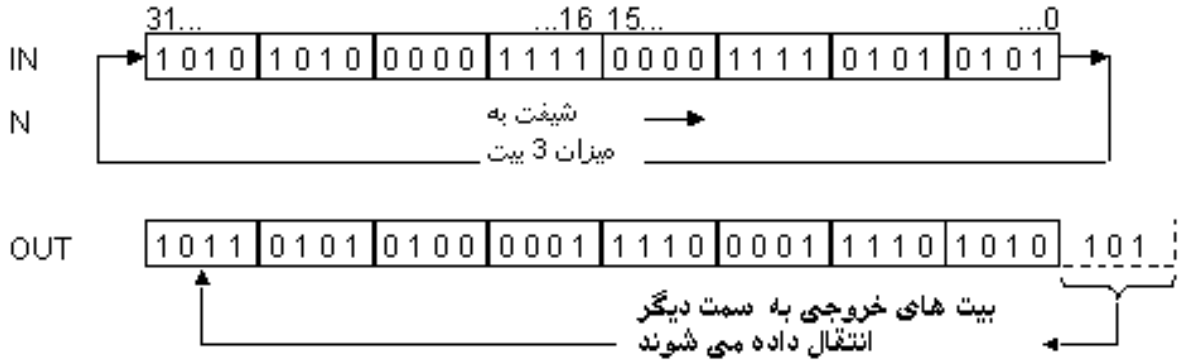


MW20: 0001_0101_1011_1110 این عدد شیفت یافته عدد بالا به میزان 3 بیت به سمت راست می باشد.

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.

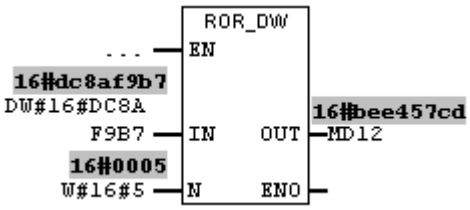
SHL_DW و SHR_DW: عملکرد این گروه دقیقاً مشابه گروه SHI_W می باشد با این تفاوت که ورودی آن از نوع دابلWORD و 4 بایتی می باشد.

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.



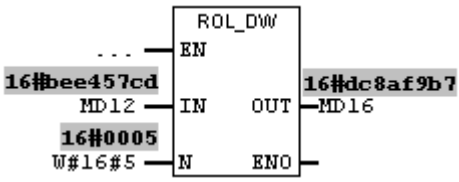
مثال : عدد DC8AF9B7 که از جنس دابل Word می باشد را در نظر بگیرید حال این عدد را 5 بار به سمت راست چرخش داده و عدد حاصل را 5 بار به سمت چپ بچرخانید. آیا عدد حاصل با عدد انتخابی برابر است؟

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.



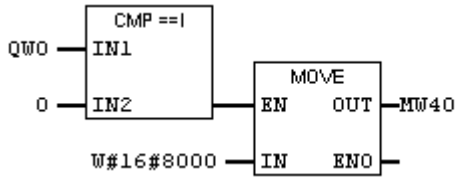
حال با استفاده از بلوک چرخش به چپ عدد فوق را 5 بیت به سمت چپ بچرخانید.

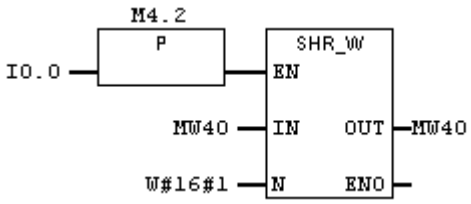
همان طور که می بینید عدد انتهایی و عدد ابتدایی با هم برابر می باشند.

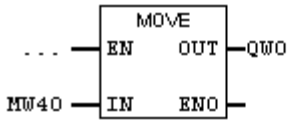


مثال : به16 خروجی یک PLC ، LED متصل شده است و از بین این LED ها یک LED روشن و بقیه خاموش می باشند .کاربر می خواهد با هر بار تحریک ورودی مربوطه این LED روشن به اندازه یک واحد به سمت راست حرکت نماید و پس از رسیدن به آخرین LED سمت راست دو باره LED روشن به انتهای سمت چپ رفته و این عمل همچنان ادامه پیدا کند.

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



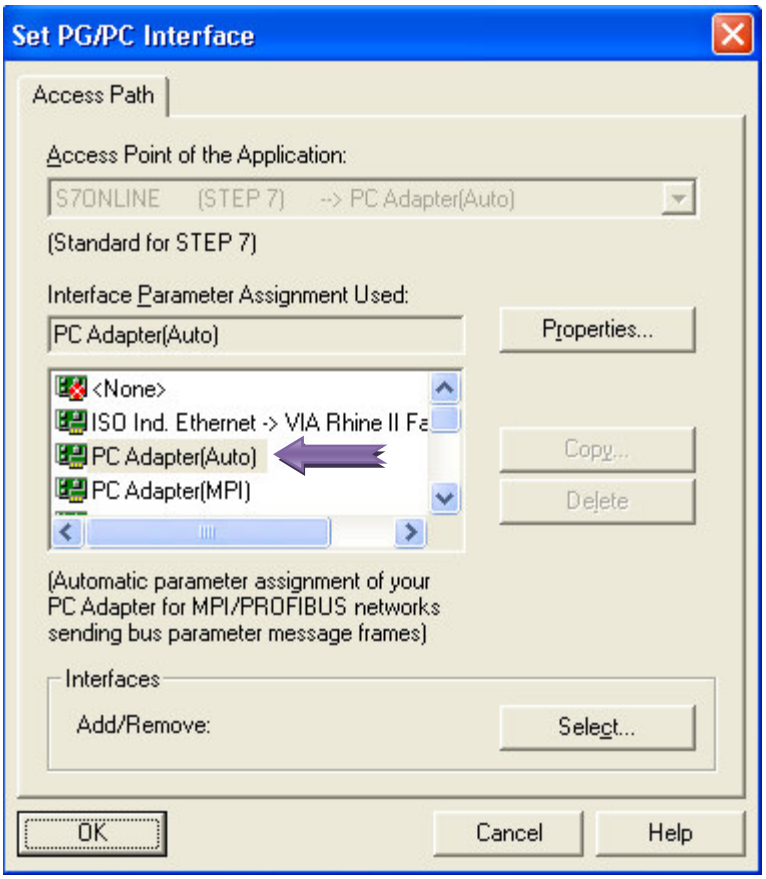




بلوک MOVE دوباره LED روشن به سمت چپ باز می گردد.

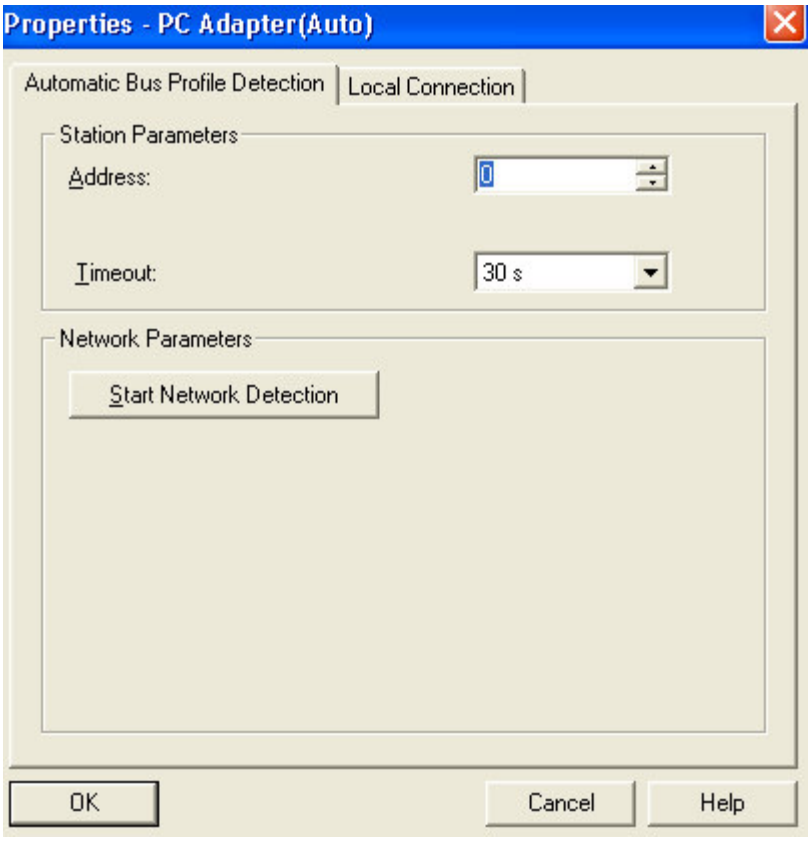
ارتباط با PLC :

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.



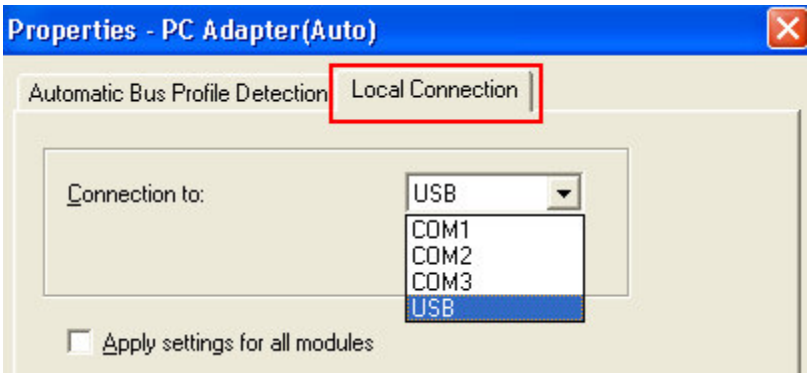
بر روی گزینه مشخص شده در شکل دابل کلیک نمائید یا آن را انتخاب نموده و بر روی گزینه Properties کلیک کنید تا صفحه تنظیمات آن مطابق شکل زیر باز شود.

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.