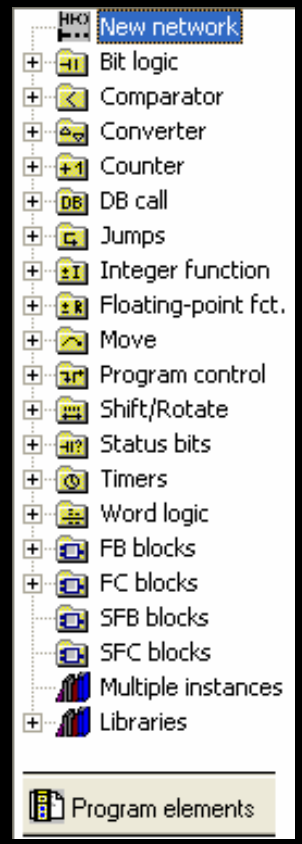
المان های برنامه



کلیّه اجزاء تشکیل دهنده یک سیستم اتوماسیون اعم از کلیدها و کنتاکت ها، تایمرها، شمارنده ها و ... را المان های برنامه می گویند. در این بخش تمامی المان های مربوط به PLC نوع S7 300 با ذکر مثال و نحوه کارتوضیح داده خواهد شد.
توصیه می شود این بخش را به موازات بخش " شروع کار با PLC" و بعد از فصل "نوشتن برنامه در OB1 "فرا بگیرید.
به علّت استفاده از شبیه ساز PLCSIM، در طی این دوره آموزشی، نیازی به اتّصال به دستگاه PLC نیست.

برای خروج کلید Escرا فشار دهید

ورود

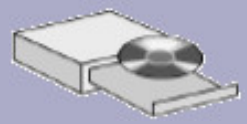
اِلمان های برنامه

- بیت لاجیک ها
- شمارنده ها
- عملیات منطقی روی WORD
- تبدیل کننده ها
- مقایسه گرها
- بیت وضعیّت Status Bit
- تایمر ها
- عملیات ریاضی
- مثال های کاربردی

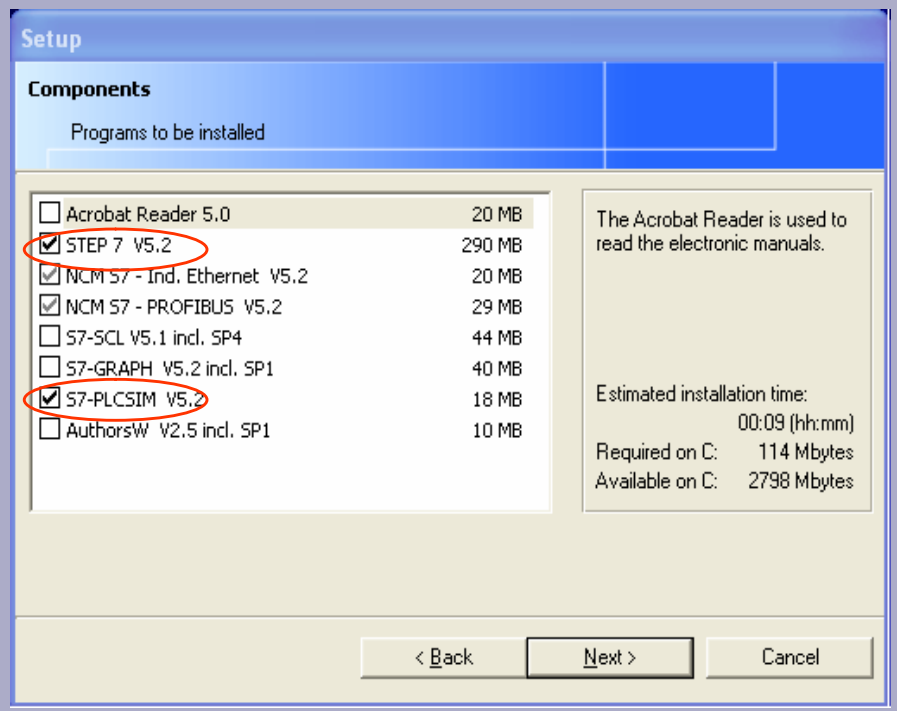
با کلیک روی صفحات یا فشار دگمه Space برنامه را مرحله به مرحله اجرا کنید

نصب نرم افزار STEP 7

برای شروع کار، ابتدا باید نرم افزار
SIMATIC STEP 7 را نصب کنید.
(نحوه نصب این برنامه در بخش "آشنایی با PLC" به طور کامل آمده است.)



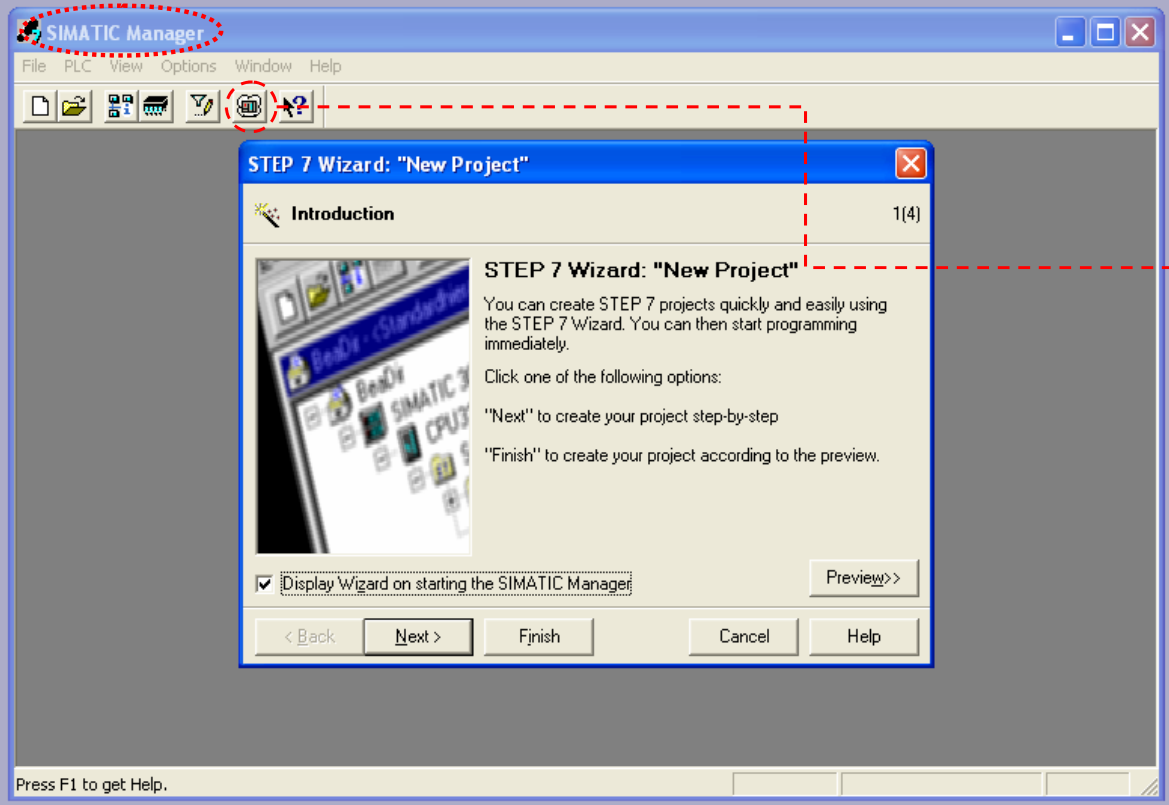
CD با برچسب
SIMATIC STEP 7
را داخل درایو CD قرار دهید.

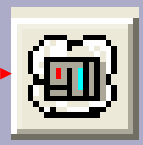


نصب برنامه به طور خودکارشروع می شود.
زبان English را انتخاب کرده و دگمهٔ Nextرا کلیک کنید.
پنجرهٔ مقابل ظاهرمی شود.
فعلاً نرم افزارهایی که مطابق شکل تیک خورده اند را انتخاب کنید.

SIMATIC Manager

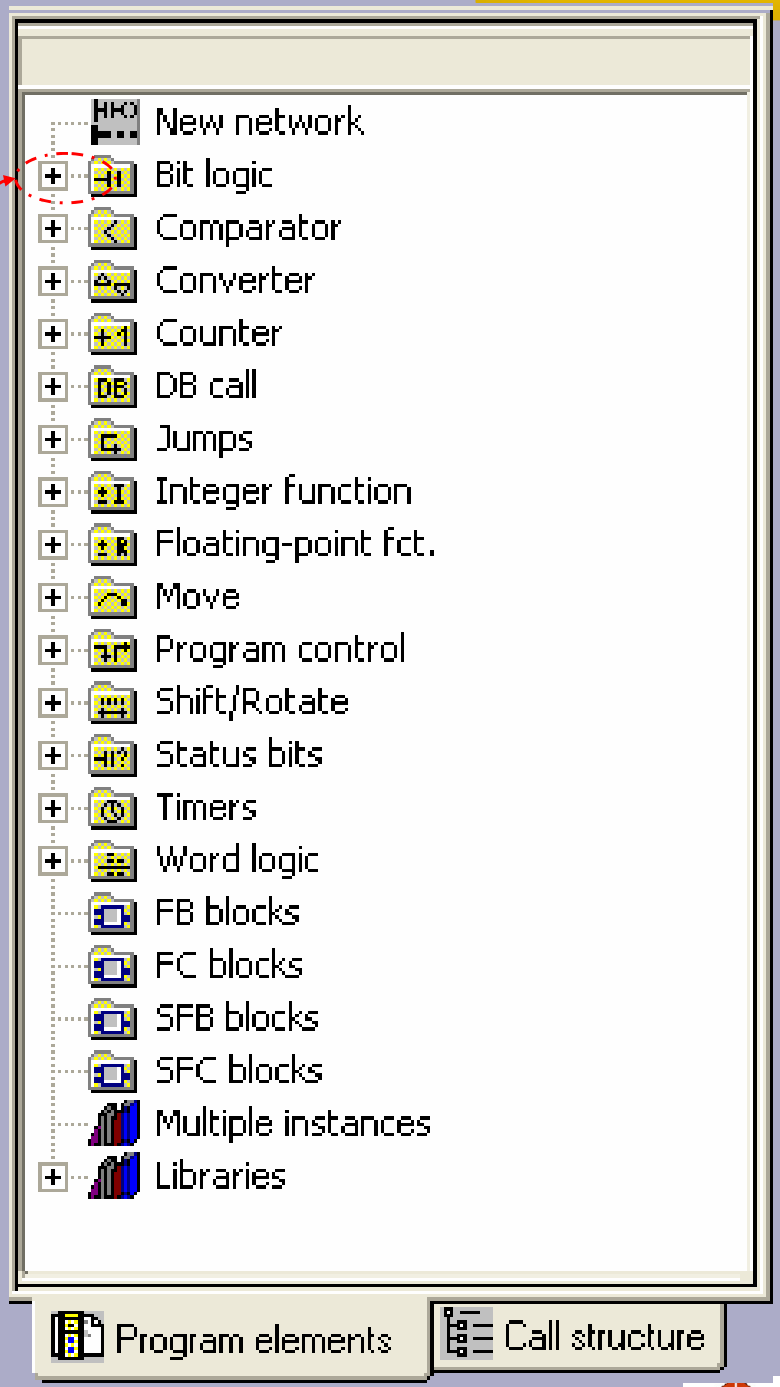
برنامه به طور خودکار نصب می شود.
پس از پایان، آیکون برنامهٔ SIMATIC Manager
در صفحهٔ کامپیوتر ظاهر می شود.
آن را دابل کلیک نمایید.





شبیه ساز S7-PLCSIM

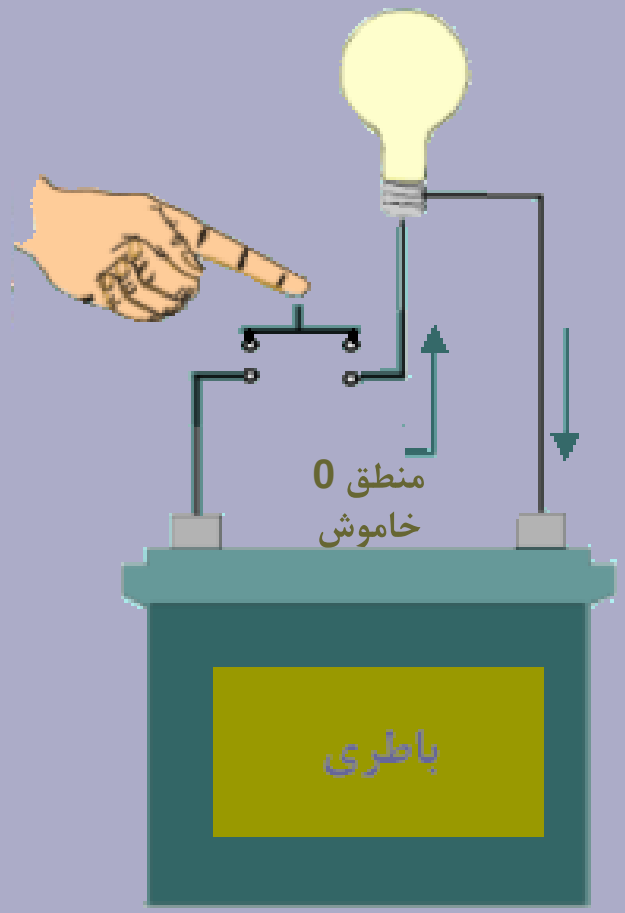
قدم نخست برای برنامه نویسی
شناختن دقیق عناصر و اجزاء
برنامه است.
اِلمان های برنامه در زبان LAD
در پِنجرهٔ سمت چپ واقع شده است
که با کلیک روی علامت + عناصر
آن گروه باز شده
و به برنامه انتقال می یابد.



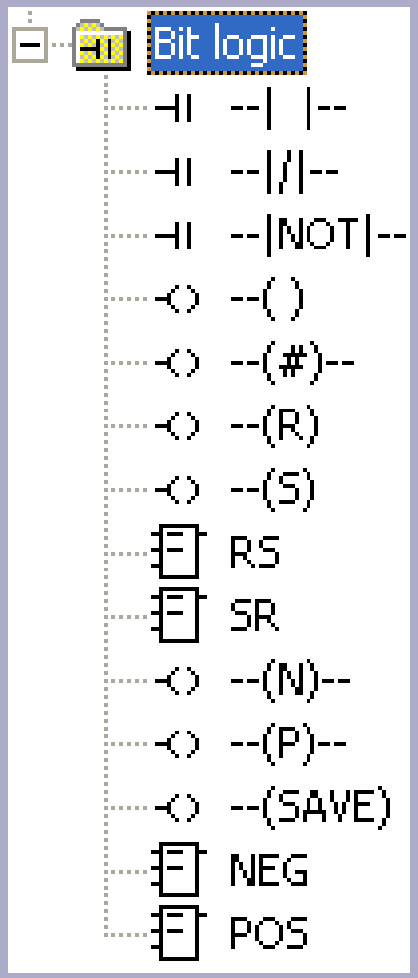
بیت لاجیک ها

دستورات بیت لاجیک با دو رقم 0,1 کار می کنند،
که این دو رقم پایه سیستم اعداد هستند.
رقم های 0,1 رقم های باینری یا بیت نامیده می شوند.
در دنیای کنتاکت ها و کویل ها رقمِ 1 بیانگر حالت فعّال
یا جریان دار، و رقم 0 بیانگر غَیر فعّال بودن
یا بدون جریان می باشند.
د ستورات بیت لاجیک حالت های صفر و یک را ترجمه کرده و
آنها را بر اساس دستورات لاجیکی ترکیب می کنند.

این ترکیبات یک نتیجهٔ صفر یا یک تولید می کنند که به آن
"نتیجهٔ عملکرد منطقی" یا "RLO" نامیده می شوند.



دربخش بیت لاجیک ها ، تعداد ۱۴ المان وجود دارد ، که تک تک آنها را را با ذکر یک مثال تشریح می کنیم

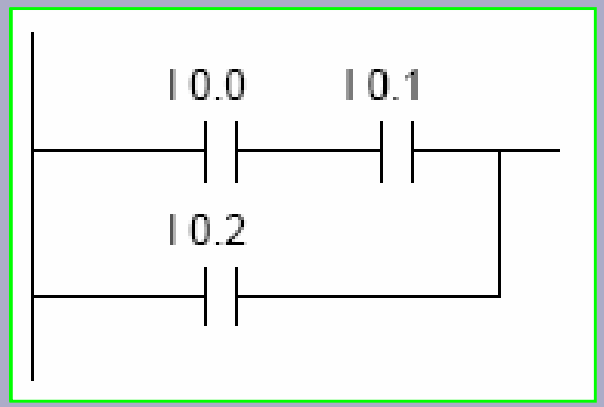


• ---| |---	کنتاکت درحالت عادّی باز
• ---| / |---	کنتاکت در حالت عادّی بسته
.• ---(SAVE)	مقدار RLO در حافظهٔ BR ذخیره می کند
• XOR	گیتِ XOR
• ---( )	بوبینِ خروجی
---(#)---	خروجی واسطه
• ---|NOT|---	معکوس کنندهٔ جریان

• ---(S)	بوبین را Set می کند.
• ---(R)	بوبین را Reset می کند.
• SR	فلیپ فلاپ SR
• RS	فلیپ فلاپ RS

---(N)---	تشخیص لبهٔ منفی RLO
• ---(P)---	تشخیص لبهٔ مثبت RLO
• NEG	تشخیص لبهٔ منفی آدرس
• POS	تشخیص لبهٔ مثبت آدرس

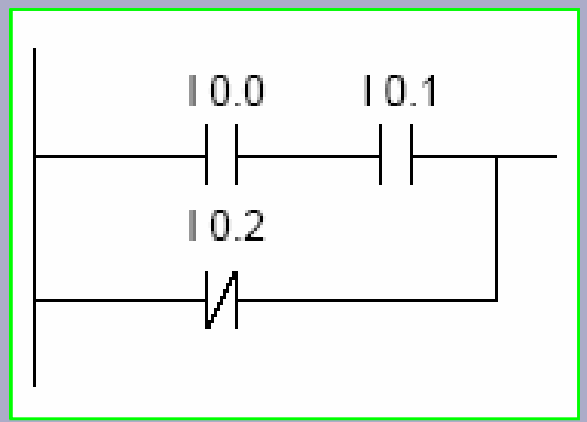
کنتاکت NO



اگر یکی از شرایط زیر مهیّا شود،
جریان برقرار خواهد شد.

وضعیّت هر دو ورودی I0.1 , I0.0 یک باشند
یا
وضعیّت ورودی I0.2 یک باشد.

کنتاکت NC



اگر یکی از شرایط زیر مهیّا شود،
جریان برقرار خواهد شد.

وضعیّت هر دو ورودی I0.1 , I0.0 یک باشند
یا
وضعیّت ورودی I0.2 صفر باشد.

استفاده از شبیه ساز PLCSIM

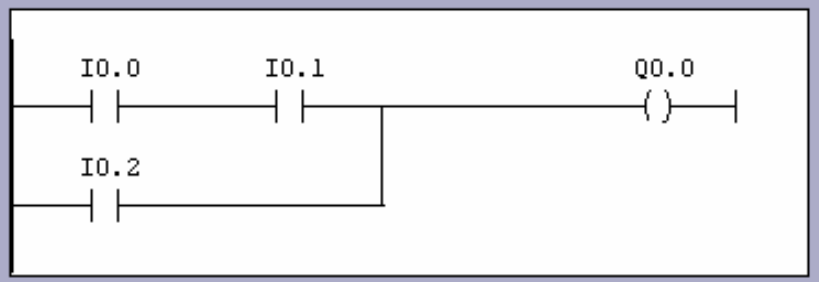
POWEREN.IR

نحوه شبیه سازی برنامه

برای نمایش نحوه تغییر خروجی Q0.0 با ورودی های I0.0, I0.1, I0.2
از شبیه ساز PLCSIM استفاده می کنیم.
مطابق آنچه در بخش "شروع کار با PLC" آموختید، در پنجره LAD/STL/FBD
Network زیر را ایجاد نمایید. برنامه را بعد از ذخیره کردن Download نمایید.

Network 1: Title:

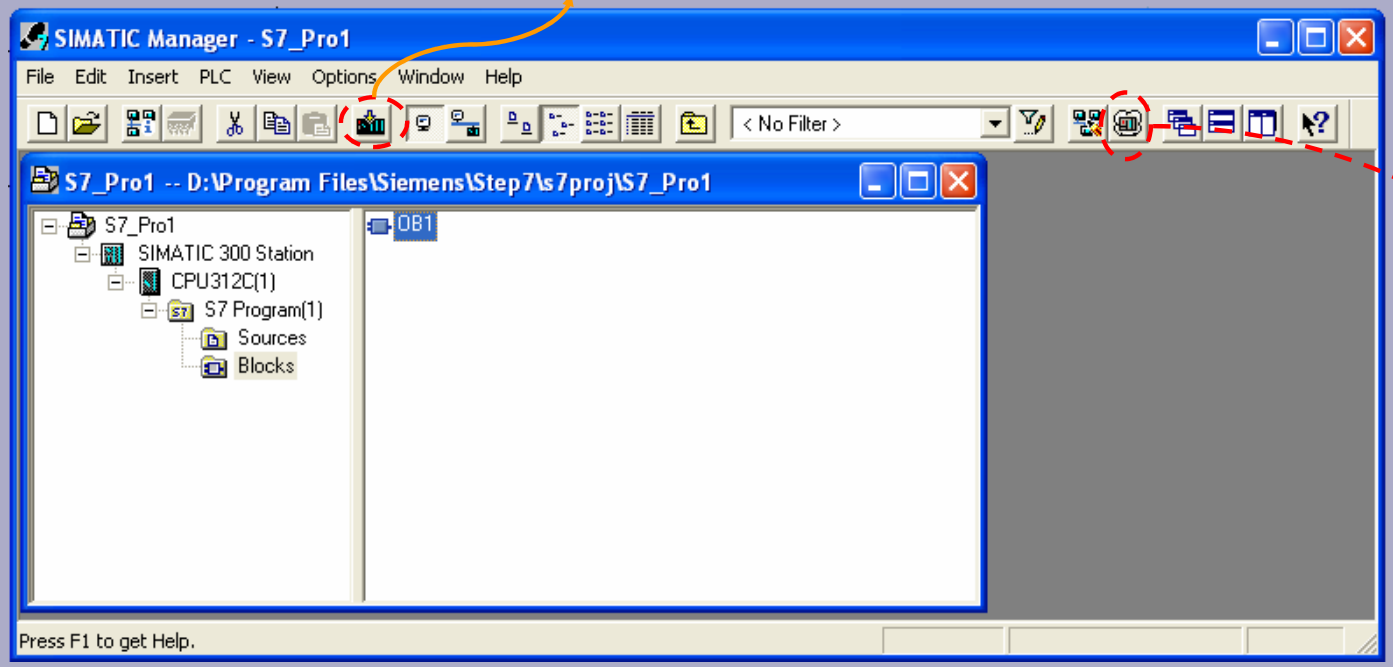
Comment:

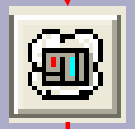


مثال شبیه سازی

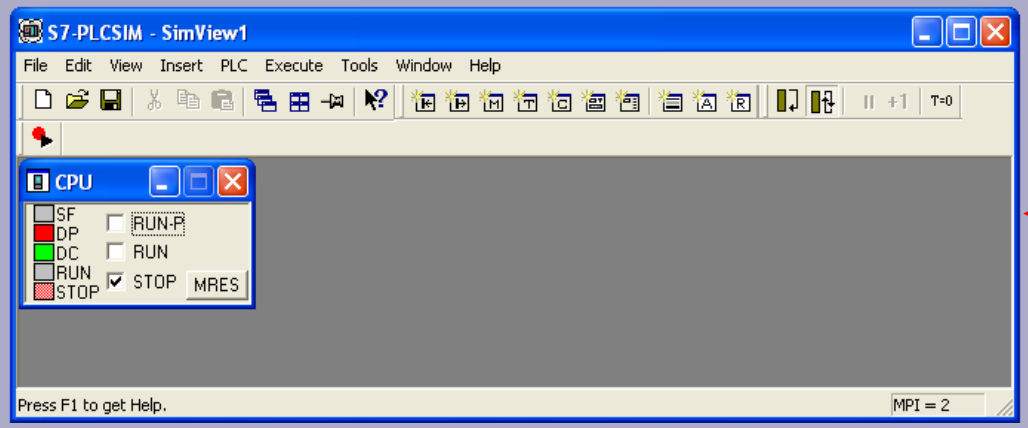
به پنجره SIMATIC Manager باز گشته و مطابق شکل صفحه بعد،
از نوار ابزار، روی PLCSIM کلیک می کنیم.

برنامه را پس از ذخیره کردن، دانلود نمایید.

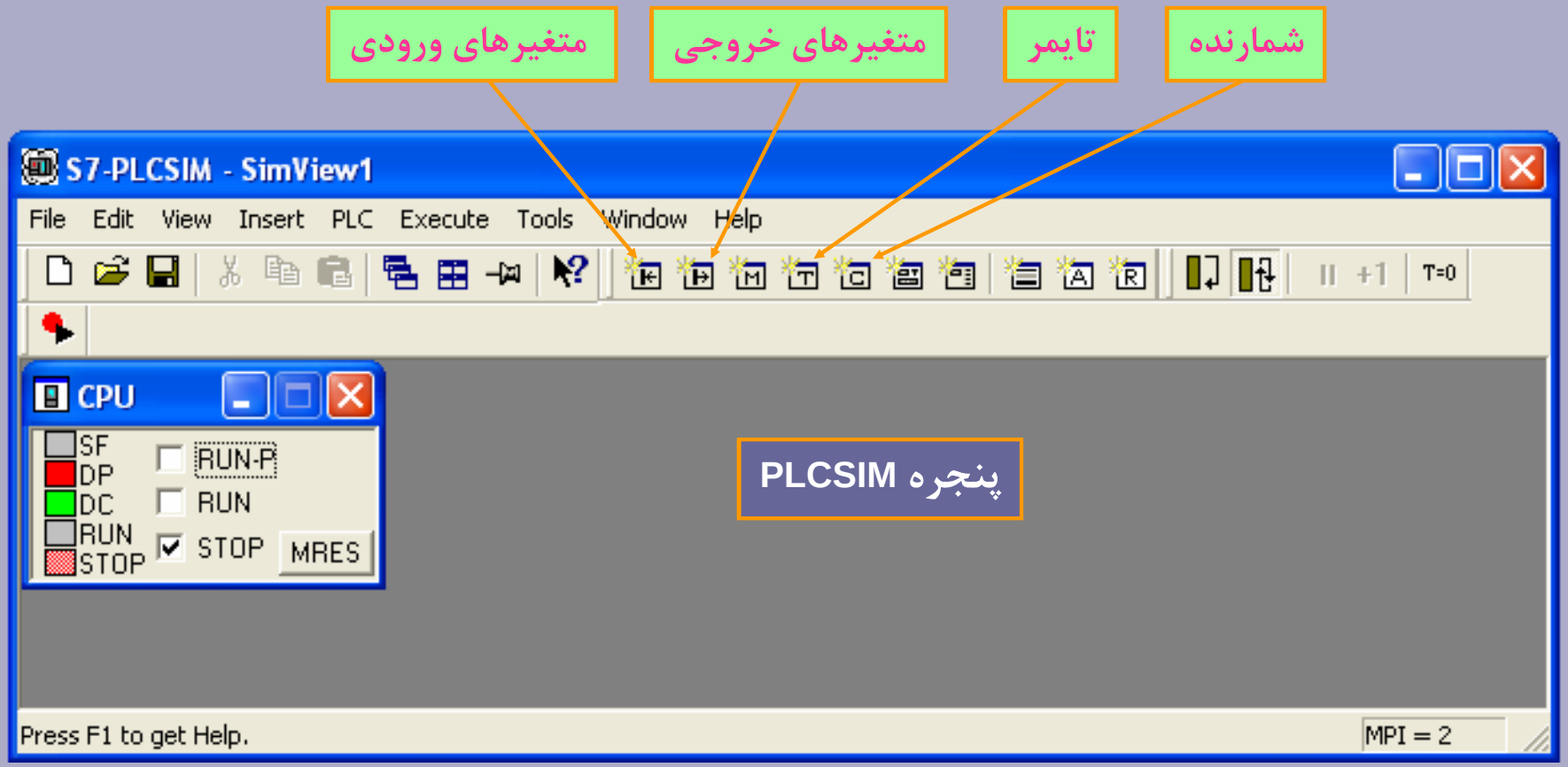




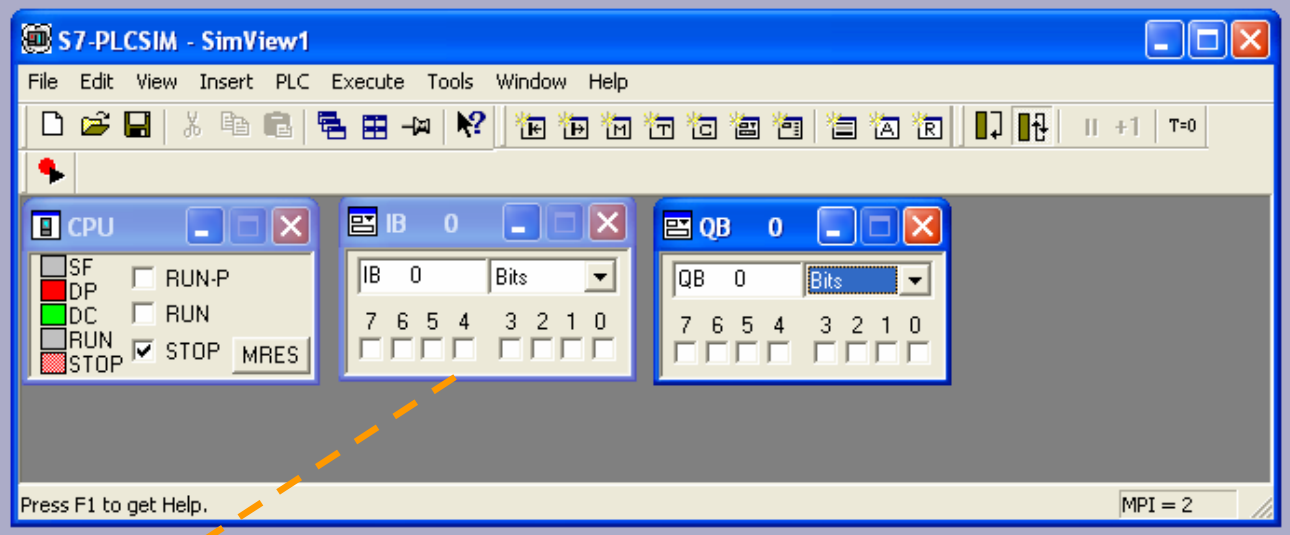
با دابل کلیک روی
PLCSIM
پنجره مقابل باز می شود.



بهتر است قبل از وارد کردن برنامه ها به دستگاه PLC آن ها را شبیه سازی کنیم،
یعنی در یک محیط ویژه، ورودی ها را به صورت نرم افزاری فعّال کرده و عکس العمل خروجی ها را
بر اساس برنامه نوشته شده، مشاهده کنیم.



یکبار روی "متغیر ورودی" و بار دیگر روی "متغیّر خروجی" کلیک می کنیم تا
مطابق شکل صفحه بعد، پنجره تنظیم هر کدام ظاهر شود.
Qبیانگر خروجی، I بیانگر ورودی، و B نیز نشان دهنده بایت است.

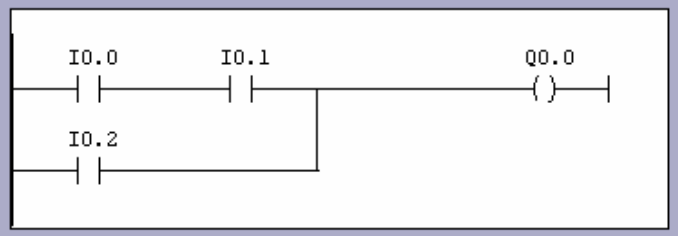


می توان نوع متغیّر ورودی
و خروجی را مطابق شکل از بایت به باینری یا دسیمال
یا هگزا، کاراکتر و .... تغییر داد.
امّا چون متغیر های مثال ما تنها کنتاکت های NO
هستند، همان بیت را انتخاب می کنیم.

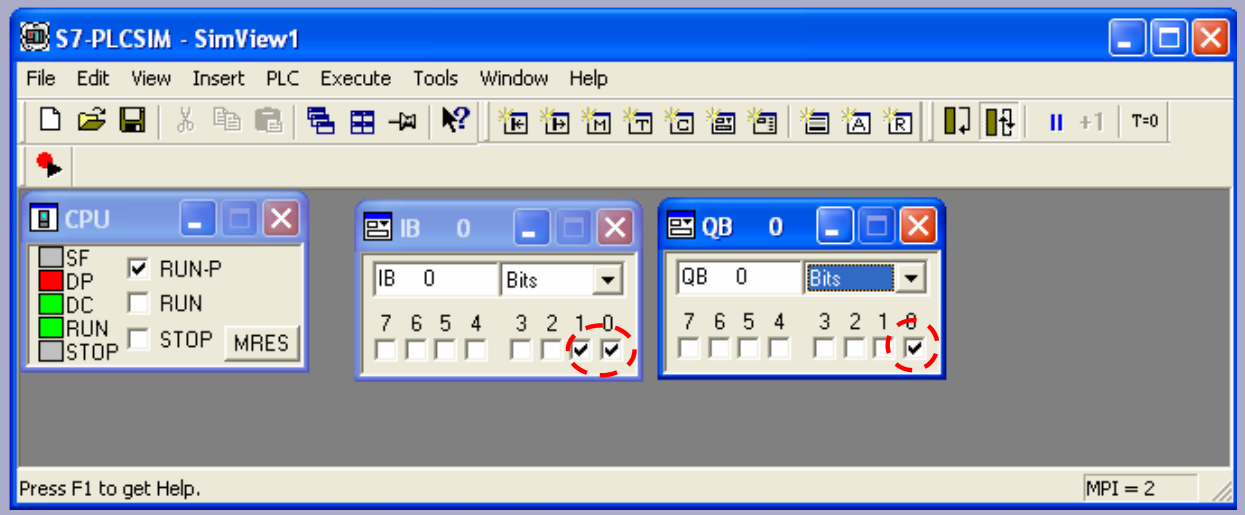
IB 0 یعنی بایت ورودی صفر ، ولی اگر بخواهیم بایت یک ورودی را کنترل کنیم،
روی محل مربوط به آن کلیک کرده و عدد ۱ را تایپ می کنیم.
امّا چون ورودی های مثال ما بیت های صفر تا دوی بایت صفر هستند (I0.0, I0.1 , I0.2)،
بنابراین بایت صفر را انتخاب می کنیم.
چون هر بایت از ۸ بیت تشکیل شده است، بتابراین مکانهای فعال کردن ۸ بیت (از ۰ تا ۷) وجود دارد.

Network 1: Title:

Comment:

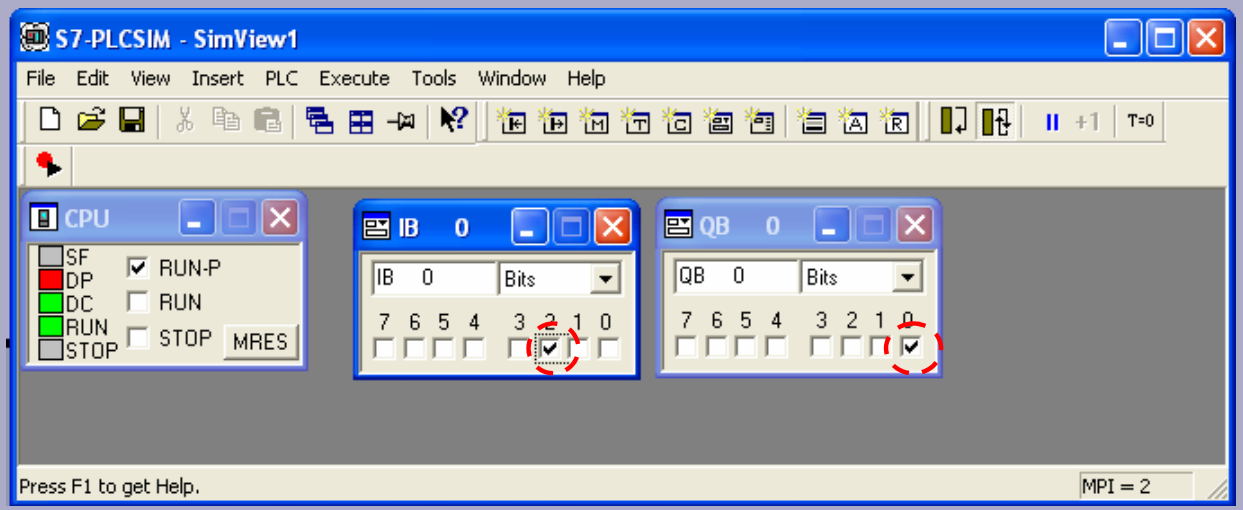


در این مثال اگر "I0.0 و I0.1 " یا I0.2 یک شوند، خروجی Q0.0 روشن می شود.



با فعّال کردن I0.0 , I0.1 خروجی Q0.0یک می شود.

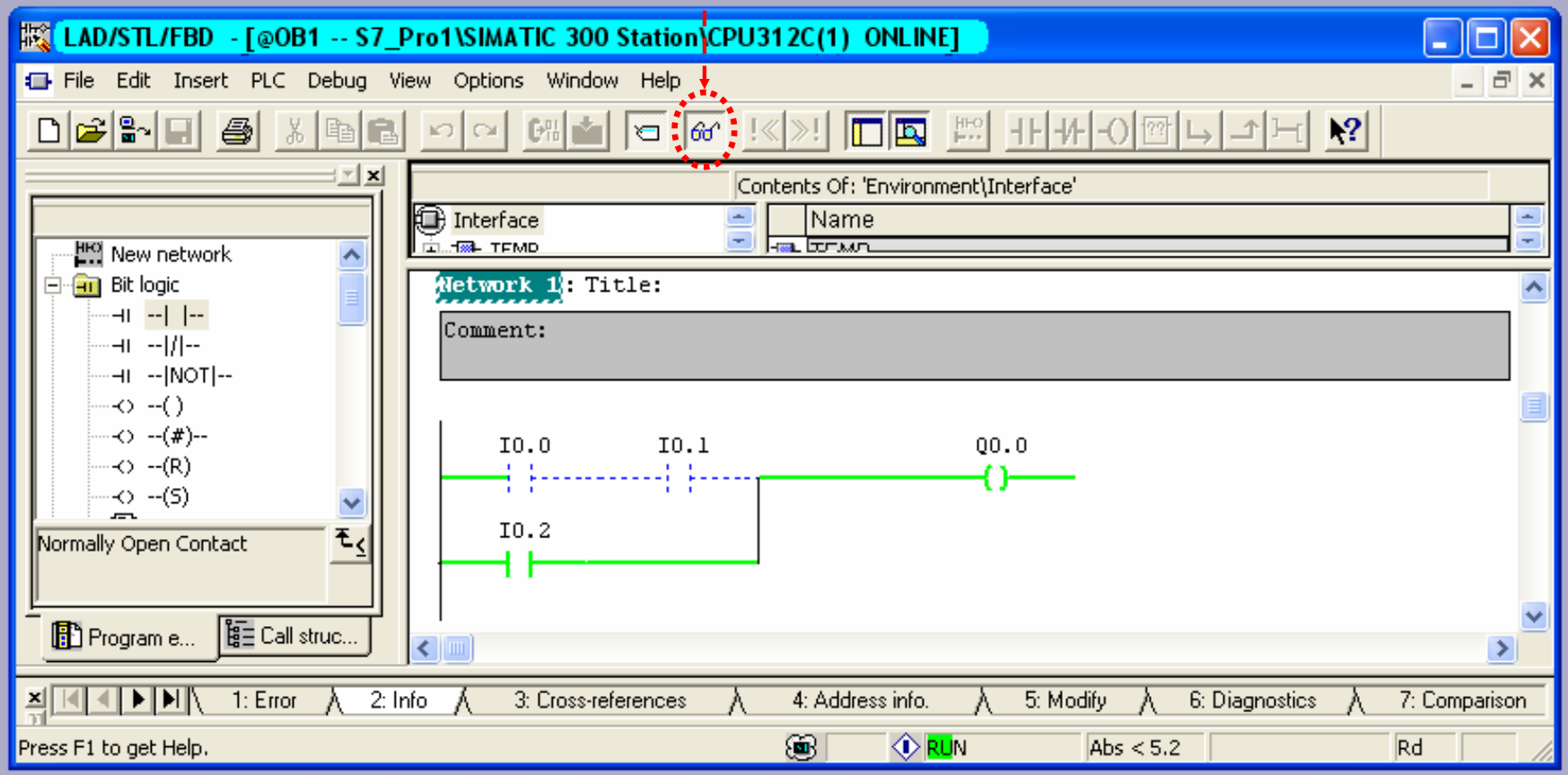
یا



با فعّال شدن I0.2 خروجی Q0.0یک می شود.

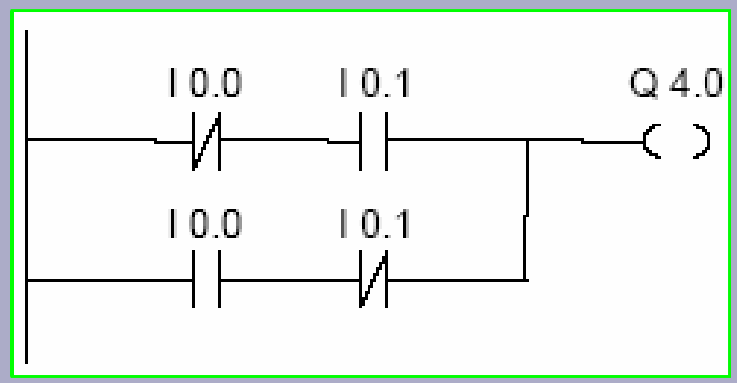
در صورت فعّال کردن شبیه ساز PLCSIM از برنامهٔ SIMATIC Manager و روشن کردن دگمهٔ Monitor از پنجرهٔ LAD/STL/FBD میتوان به ازاء ورودی های مختلف نحوهٔ تبدیل سازی را مشاهده کرد.

Monitor



همانطور که مشاهده می کنید، مسیر وجود برق با توجّه به المان های فعّال شده در PLCSIM با خطوط سبز رنگ قابل رویت است.

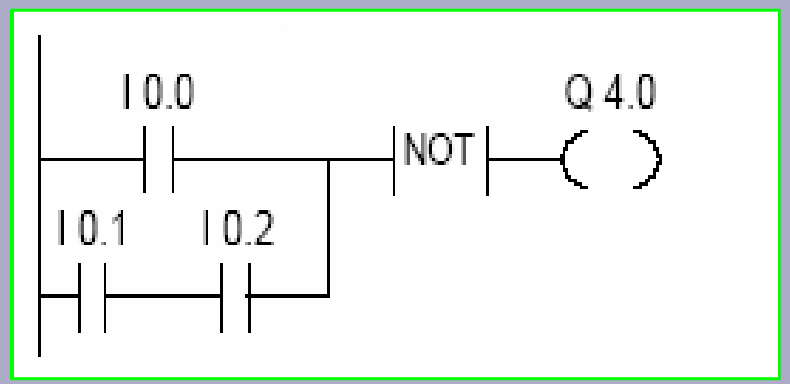
گیتِ XOR



اگر شرایط زیر مهیّا شود،
جریان برقرار خواهد شد.

(I0.0 = "0“ AND I0.1 = "1")
OR
(I0.0 = "1“ AND I0.1 = "0")

NOT



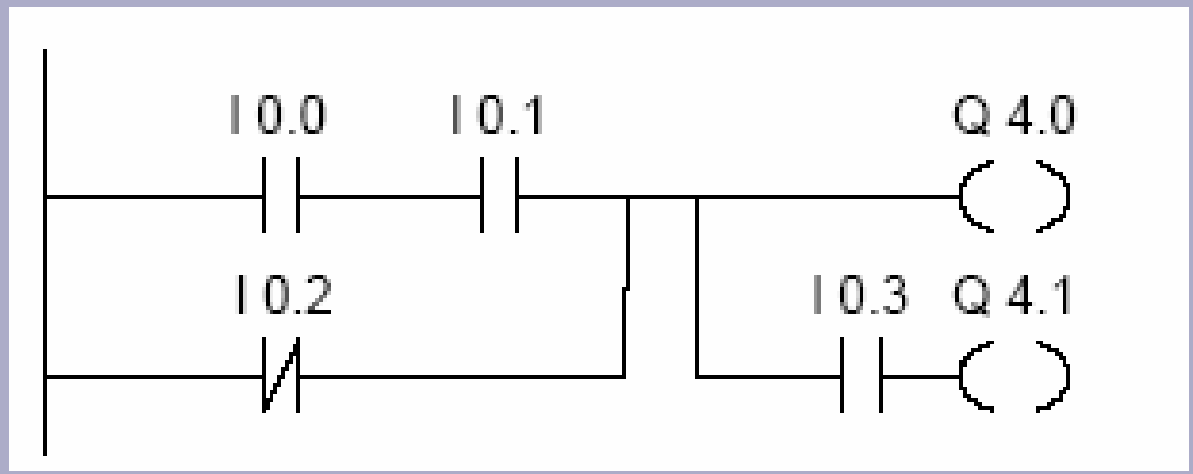
اگر یکی از شرایط زیر مهیّا شود،
خروجی Q4.0 صفر خواهد شد.

وضعیّت ورودی I0.0 یک باشند
یا
وضعیّت هر دو ورودی I0.2 , I0.1 یک باشند.

بوبین خروجی

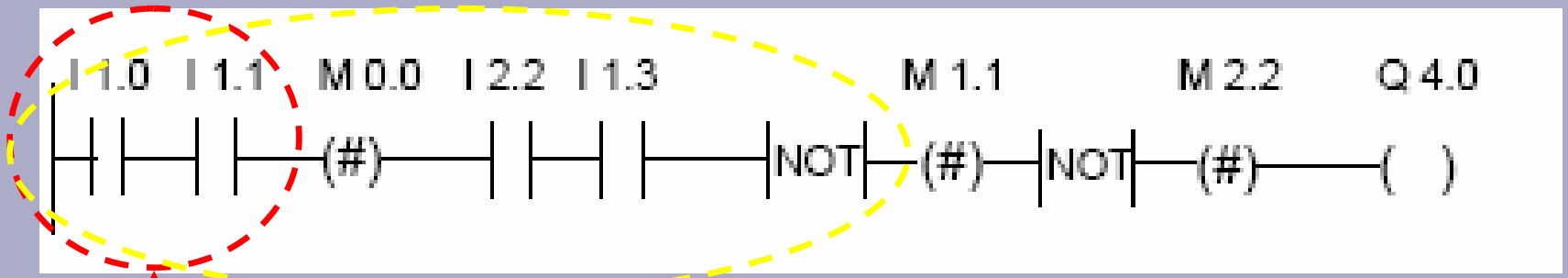
اگر یکی از شرایط زیر مهیّا شود،
خروجی Q4.0 یک خواهد شد:
وضعیّت ورودی های I0.0, I0.1 یک باشند
یا
وضعیّت ورودی I0.2 صفر باشند.

اگر یکی از شرایط زیر مهیّا شود،
خروجی Q4.1 یک خواهد شد:
وضعیّت ورودی های I0.0 , I0.1 یک باشند
یا
وضعیّت ورودی I0.2 صفر و ورودی I0.3یک باشند.



تمرین: مدار فوق را با PLCSIM بررسی کنید.

خروجی واسطه



نتیجهٔ این دو ورودی در M 0.0 ذخیره می شود

نتیجهٔ این قسمت در M 1.1 ذخیره می شود

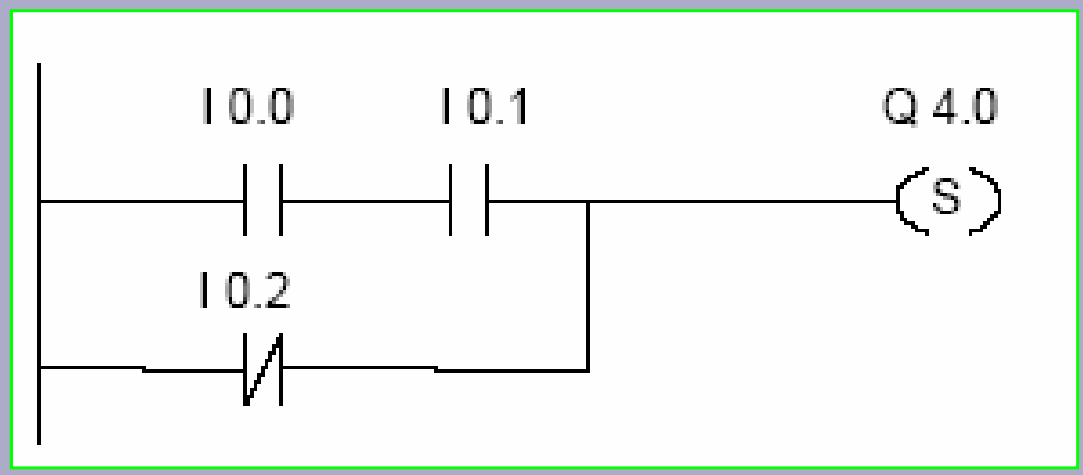
نتیجهٔ کلّ در M2.2 ذخیره می شود.

M0.0, M1.1, M2.2 سه بیت حافظه می باشند.

بوبین سِت یا "یک ماندگار" (Set)

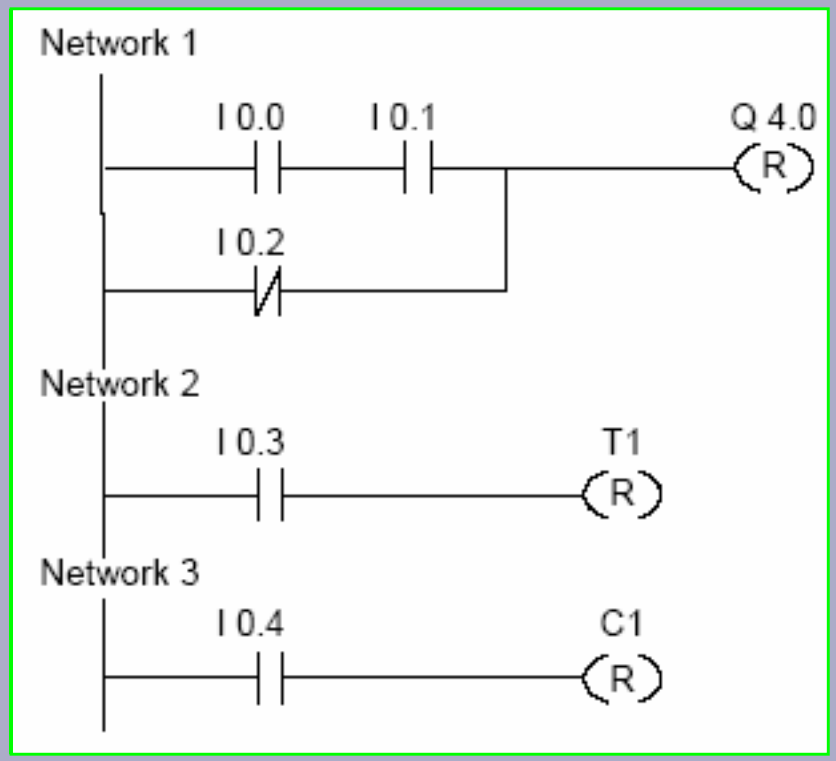
ست شدن یک المان یعنی هنگامی که آن المان تحت شرایطی یک شد،
تا زمان ریست شدن از جای دیگر برنامه در همان حالت یک باقی بماند.
بوبین های خروجی معمولی(مانند رله ها و کنتاکتورها) تا شرایط یک شدن آنها برقرار باشند یک می مانند.
در مثال زیر اگر یکی از شرایط زیر مهیّا شود، خروجی Q4.0 سِت خواهد شد:
وضعیّت ورودی های I0.0, I0.1 یک باشند
یا
وضعیّت ورودی I0.2 صفر باشند.

وقتی خروجی یک بار سِت شد با بر هم زدن شرایط فوق، صفر نمی شود.



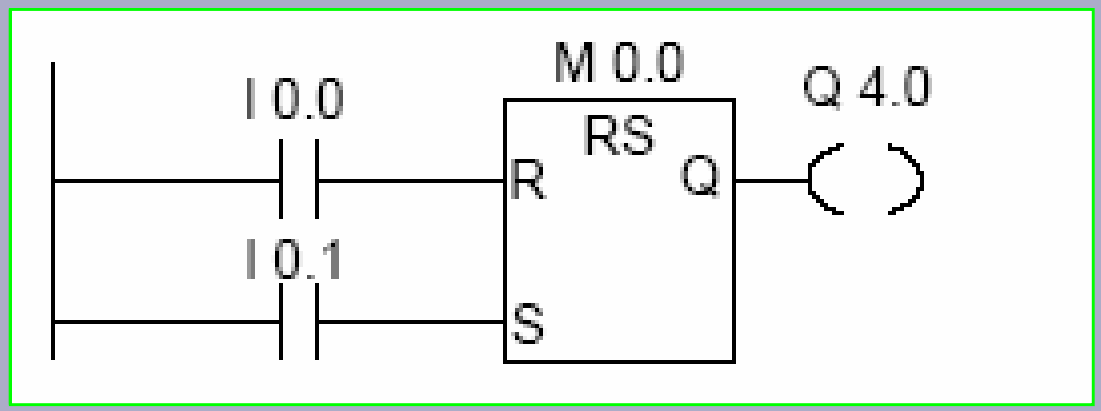
بوبین ریست (صفر ماندگار)

اگر یکی از شرایط زیر مهیّا شود، خروجی Q4.0 **ریست** خواهد شد:
وضعیّت ورودی های I0.0, I0.1 یک باشند
یا
وضعیّت ورودی I0.2 صفر باشند.
وقتی خروجی یک بار ریست شد در همین حالتِ صفر می ماند تا در جای دیگر برنامه ست شوند.
تایمر T1 زمانی ریست می شَود که ورودی I0.3 یک باشد.
شمارندهٔ C1 زمانی ریست می شود که ورودی I0.4 یک باشد.



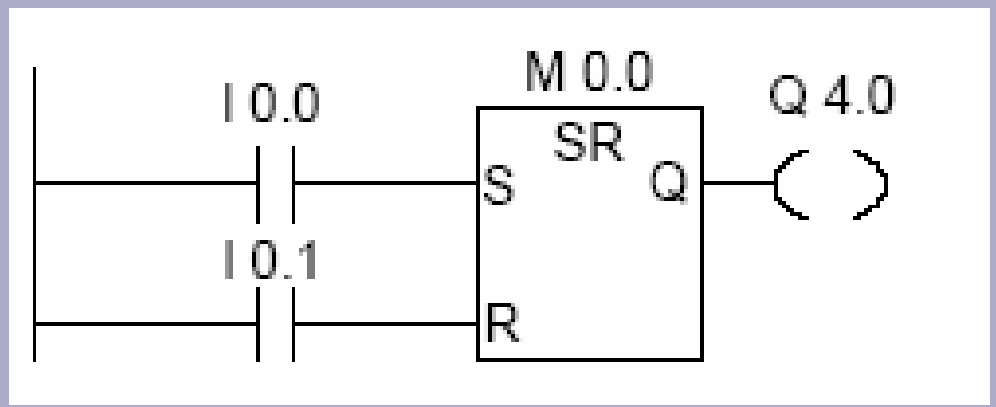
فلیپ فلاپ RS

اگر وضعیّت ورودی I0.0 یک باشد و وضعیّت ورودی I0.1 صفر باشد،
بیت حافظهٔ M0.0 ست می شود.
در غیر اینصورت .اگروضعیّت ورودی I0.0 صفر باشد ووضعیّت ورودی I0.1 یک باشد
بیت حافظهٔ M0.0 ریست شده و خروجی Q4.0 یک می شود.
اگر هر دو ورودی صفر باشند هیچ اتفاقی نمی افتد.
اگر هر دو ورودی یک شوند، به دلیل اولویّت، عمل سِت اتفاق می افتد:
M0.0 ست می شود و Q4.0 یک می شَود.

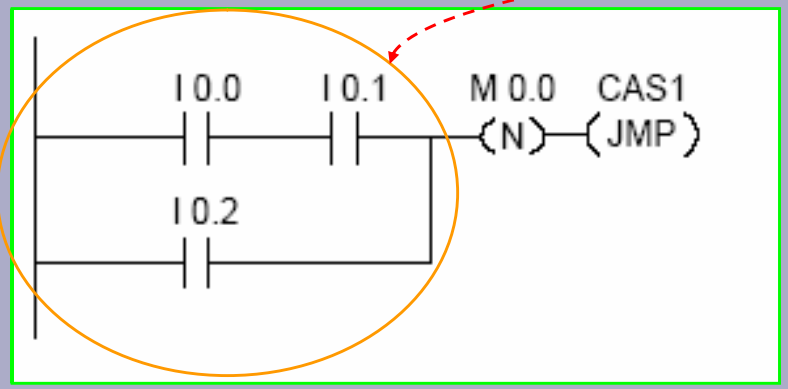


فلیپ فلاپ SR

اگر وضعیّت ورودی I0.0 یک باشد و وضعیّت ورودی I0.1 صفر باشد،
بیت حافظهٔ M0.0 ست می شود.
در غیر اینصورت .اگروضعیّت ورودی I0.0 صفر باشد ووضعیّت ورودی I0.1 یک باشد
بیت حافظهٔ M0.0 ریست شده و خروجی Q4.0 صفر می شود.
اگر هر دو ورودی صفر باشند هیچ اتفاقی نمی افتد.
اگر هر دو ورودی یک شوند، به دلیل اولویّت عمل ریسِت اتفاق می افتد:
M0.0 ریست می شود و Q4.0 صفر می شَود.

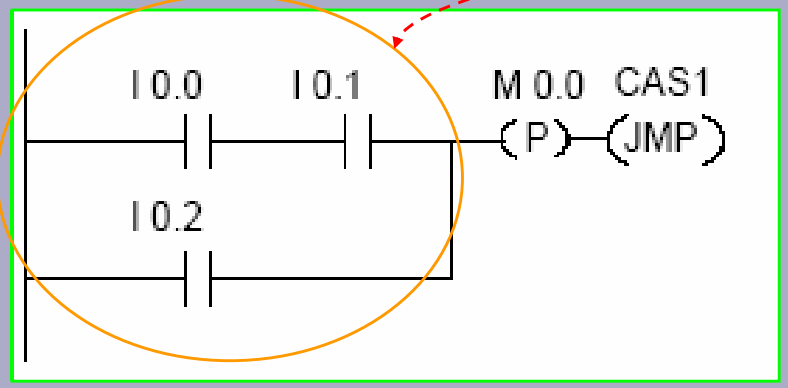


تشخیص لبهٔ منفی



اگر نتیجهٔ سه ورودی NO از یک به صفر تغییر کند برنامه به برچسبِ CAS1 پرش می کند.

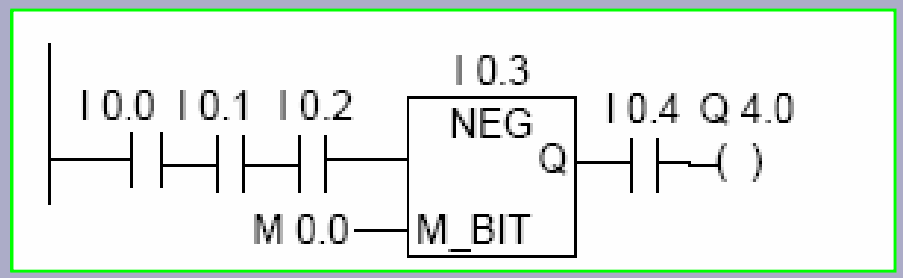
تشخیص لبهٔ مثبت



اگر نتیجهٔ سه ورودی NO از صفر به یک تغییر کند برنامه به برچسبِ CAS1 پرش می کند.

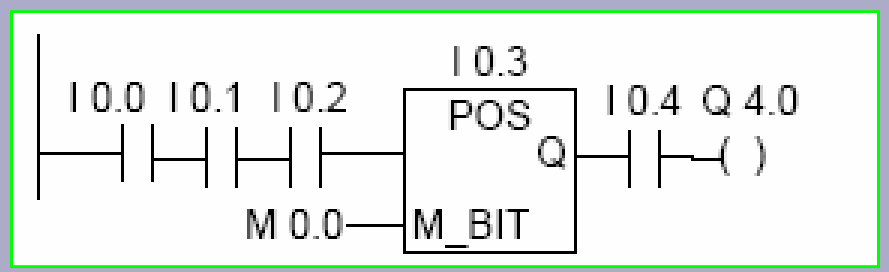
تشخیص لبهٔ منفی آدرس

اگر شرایط زیر مهیّا شوند،
خروجی Q4.0 یک خواهد شد:
وضعیّت ورودی های I0.0, I0.1,I0.2 یک باشند
ورودی I0.3 از یک به صفر تغییر یابد
وضعیّت ورودی I0.4 یک باشد.



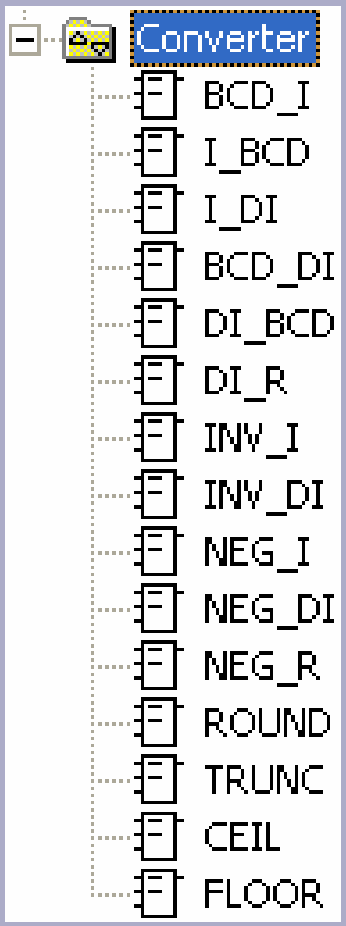
تشخیص لبهٔ مثبت آدرس

اگر شرایط زیر مهیّا شوند،
خروجی Q4.0 یک خواهد شد:
وضعیّت ورودی های I0.0, I0.1,I0.2 یک باشند
ورودی I0.3 از صفر به یک تغییر یابد
وضعیّت ورودی I0.4 یک باشد.



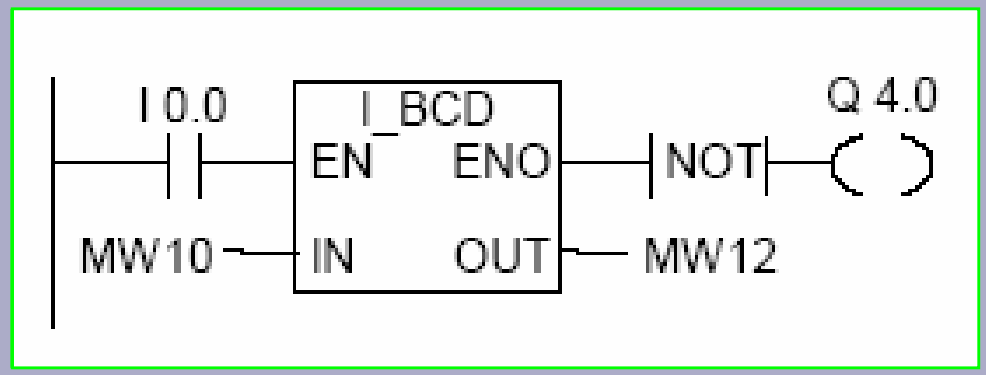
تبدیل کننده ها

در قسمت تبدیل کننده ها ، تعداد ۱۵ نوع المان به صورت بلاکی وجود دارد که هرکدام را با ذکر یک مثال تشریح می کنیم



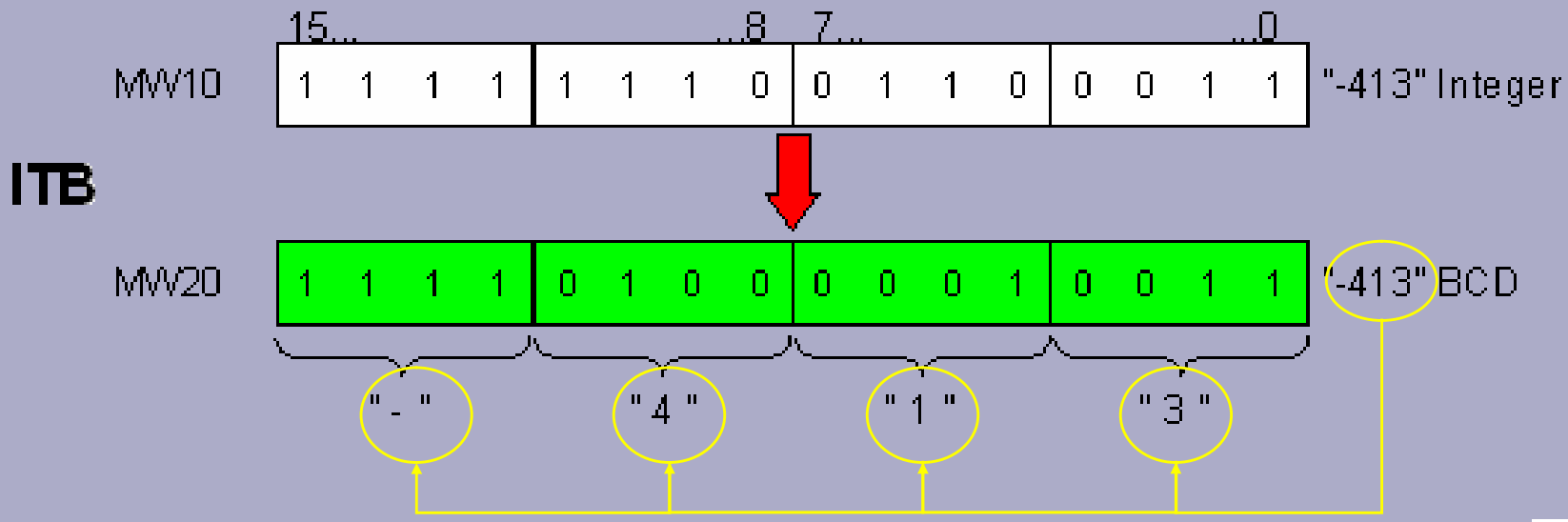
• BCD_I	تبدیل BCD به عدد صحیح
• I_BCD	تبدیل عدد صحیح به BCD
• BCD_DI	تبدیل BCDبه عدد صحیح دوبل
• I_DINT	عدد صحیح دوبل به BCD
• DI_BCD	عدد صحیح دوبل به عدد اعشاری
• DI_REAL	مکمّل گیری تک تک بیت های عدد صحیح
• INV_I	مکمّل گیری تک تک بیت های عدد صحیح دوبل
• INV_DI	مکمّل گیری دو تا دو تای عدد صحیح
• NEG_I	مکمّل گیری دو تا دو تای عدد صحیح دوبل
• NEG_DI	منفی کردن عدد اعشاری
• NEG_R	رند کردن عدد حقیقی
• ROUND	حذف اعشار عدد صحیح
• TRUNC	رند کردن عدد حقیقی به سمت بالا
• CEIL	رند کردن عدد حقیقی به سمت پایین
• FLOOR	

تبدیل عدد صحیح(Integer)به BCD

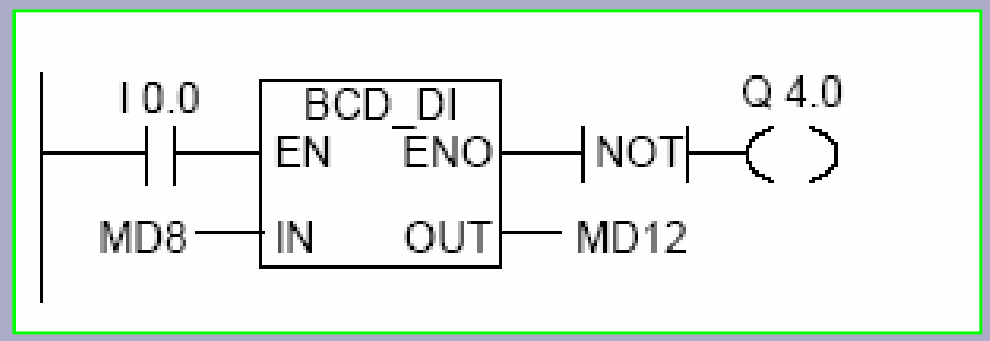


اگر ورودی I0.0یک باشد، محتویّات MW10 به عنوان یک عدد صحیح (Integer) خوانده می شود و به یک عدد BCDتبدیل می شود.
نتیجه در MW12 ذخیره می شود.
خروجی Q4.0 هنگامی که تبدیل انجام نشده باشد، (به خاطر وجودNOT)، یک می باشد.

به عنوان مثال می خواهیم عدد ۹۱۵ که به صورت عدد صحیح میباشد را به BCD تبدیل کنیم.

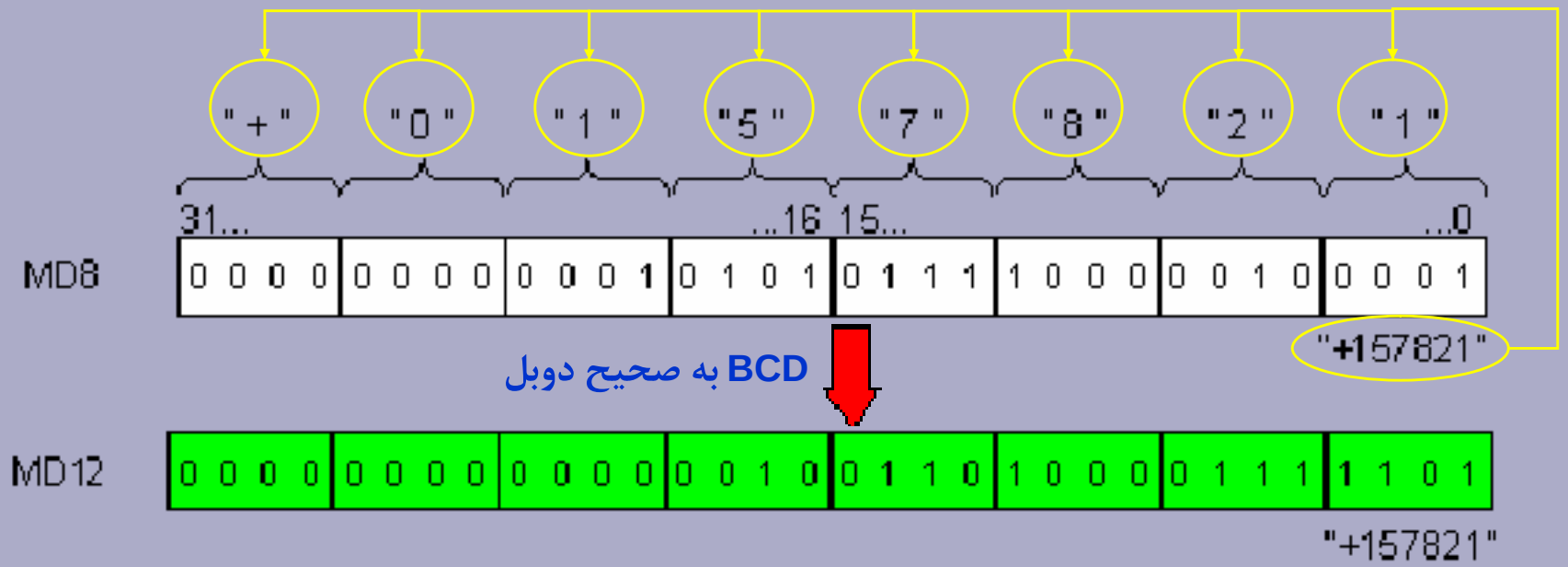


تبدیل BCD به عدد صحیح دوبل (Double Integer)

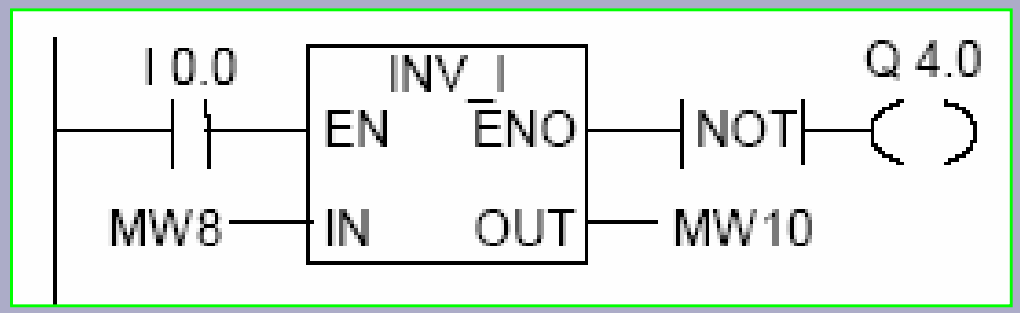


اگر ورودی I0.0 یک باشد، محتویّات MW10 به عنوان یک BCD خوانده می شود و به یک عدد صحیح دوبلتبدیل می شود. نتیجه در MW12 ذخیره می شود.
خروجی Q4.0 هنگامی که تبدیل انجام نشده باشد (به خاطر وجودNOT)، یک می باشد.

به عنوان مثال می خواهیم عدد ۵۰۷۸۲۱ که به صورت BCD میباشد به عدد صحیح دوبل آن تبدیل کنیم.



مکمّل گیرِ بیت ها



اگر ورودی I0.0یک باشد، هر بیت MW8 معکوس می شود و داخل MW10ریخته می شود.

خروجی Q4.0 هنگامی که تبدیل انجام نشده باشد، (به خاطر وجودNOT)، یک می باشد.

0100	0001	1000	0001

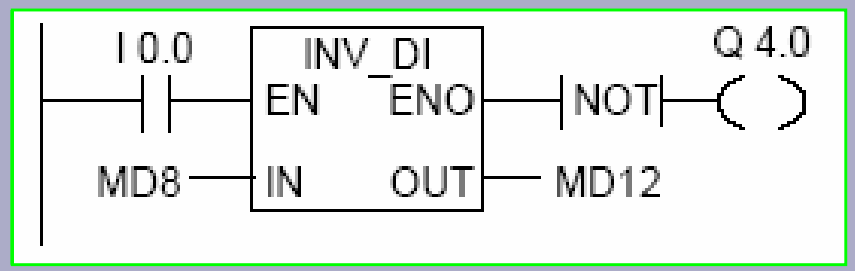
MW8

INV_I

معکوس بیت ها

1011	1110	0111	1110

MW10



مکمّل گیرِ بیت ها برای DI

F0FF FFF0

MW8

INV_DI

معکوس بیت ها

0F00 000F

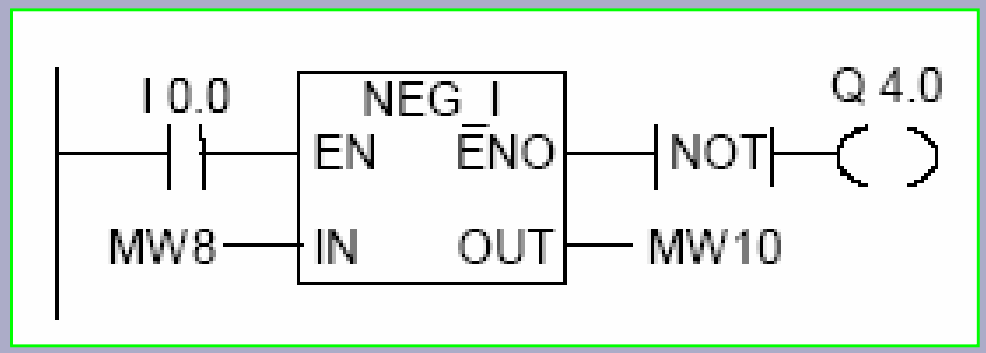
MW12

مثال های مکمّل گیری

قبل از اجرا	0101	1101	0011	1000
بعد از اجرا	1010	0010	1100	1000

قبل از اجرا	0101	1111	0110	0100	0101	1101	0011	1000
بعد از اجرا	1010	0000	1001	1011	1010	0010	1100	1000

منفی کننده عدد صحیح (NEG_I)

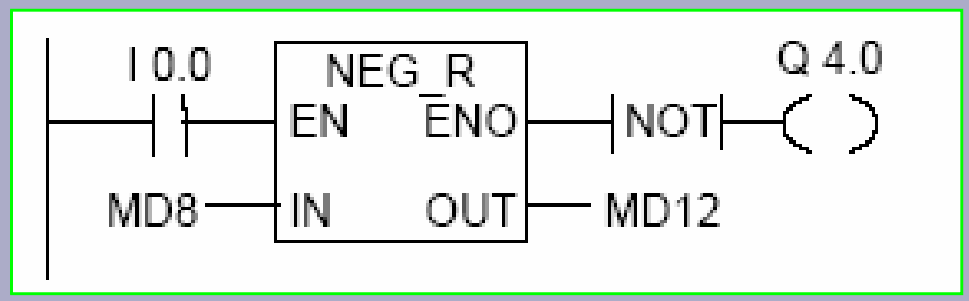


اگر ورودی I0.0یک باشد، مقدار MW8 با علامت منفی داخل MW10 ریخته می شود.

خروجی Q4.0 هنگامی که تبدیل انجام نشده باشد، (به خاطر وجودNOT)، یک می باشد.

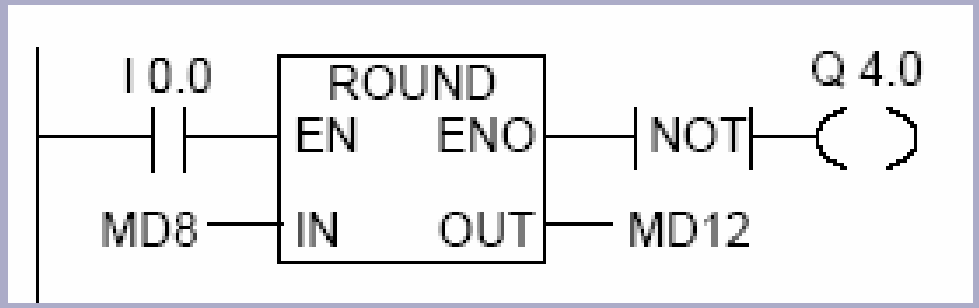
174 → NEG_I → -174

منفی کننده عدد حقیقی (NEG_I)



6.234 → NEG_R → -6.234

رند کننده (ROUND)

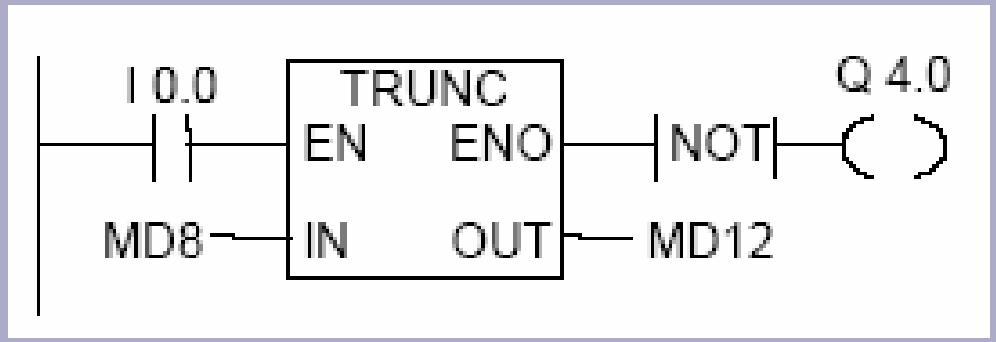


اگر ورودی I0.0 یک باشد، محتویّات MD8 که حاوی عدد اعشاری است به نزدیکترین عدد صحیح دوبل آن تبدیل می شود .
نتیجه داخل MD12ریخته می شود.

خروجی Q4.0 هنگامی که تبدیل انجام نشده باشد، (به خاطر وجودNOT)، یک می باشد.

MD10 = "100.5“	رند کننده →	MD20 = "+100"
MD10 = "-100.5"	← رند کننده	MD20 = "-100"

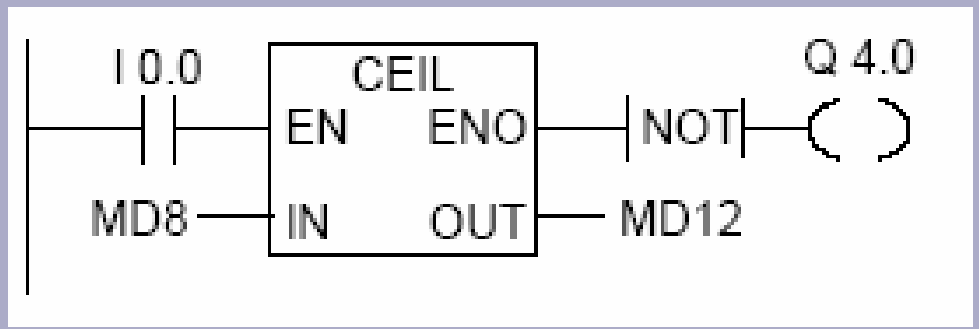
حذف اعشار (TRUNC)



اگر ورودی I0.0یک باشد، قسمت اعشارمقدار MW8 حذف شده و نتیجه داخل MW10 ریخته می شود.

خروجی Q4.0 هنگامی که تبدیل انجام نشده باشد، (به خاطر وجودNOT)، یک می باشد.

حذف اعشار (TRUNC)



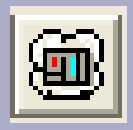
اگر ورودی I0.0یک باشد، قسمت اعشارمقدار MW8 حذف شده و نتیجه داخل MW10 ریخته می شود.

خروجی Q4.0 هنگامی که تبدیل انجام نشده باشد، (به خاطر وجودNOT)، یک می باشد.

استفاده از شبیه ساز PLCSIM

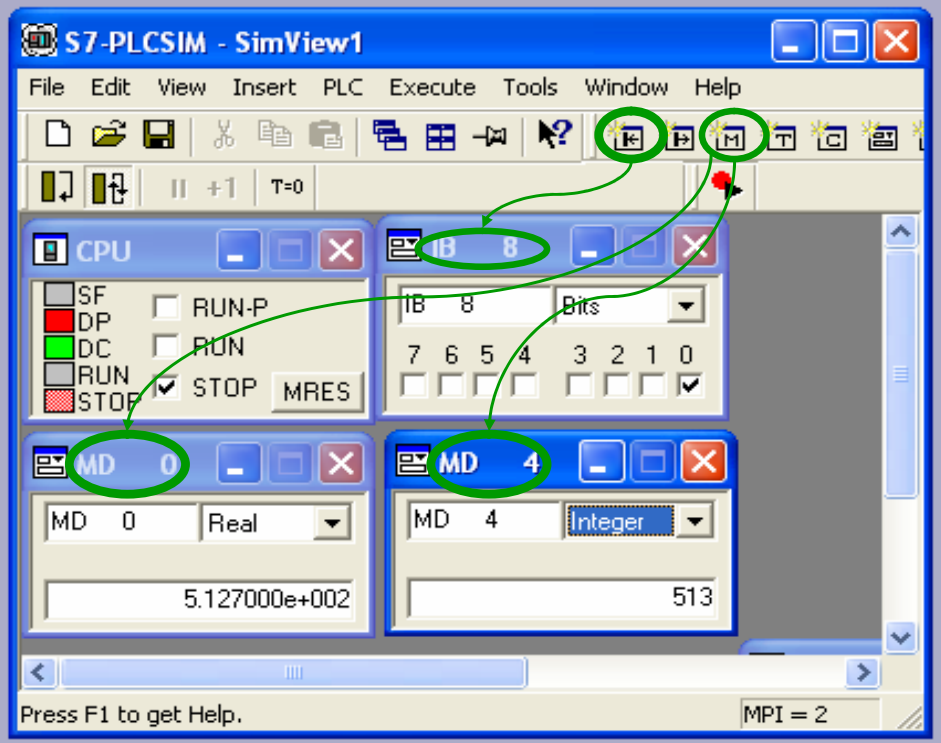
اِلمان های برنامه

دگمهٔ **Monitor**

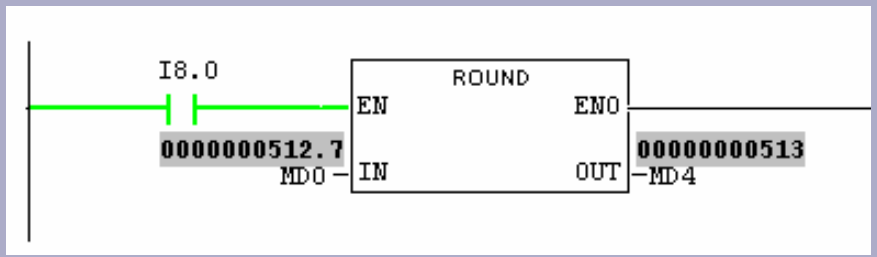


شبیه ساز **PLCSIM**

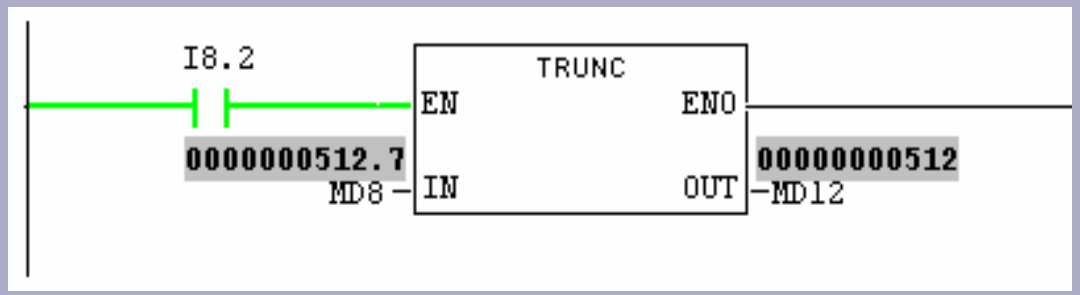
با فعّال کردن شبیه ساز **PLCSIM** از برنامهٔ **SIMATIC Manager** و روشن کردن دگمهٔ **Monitor** از پنجرهٔ **LAD/STL/FBD** میتوان به ازاء ورودی های مختلف نحوهٔ تبدیل سازی را مشاهده کرد.

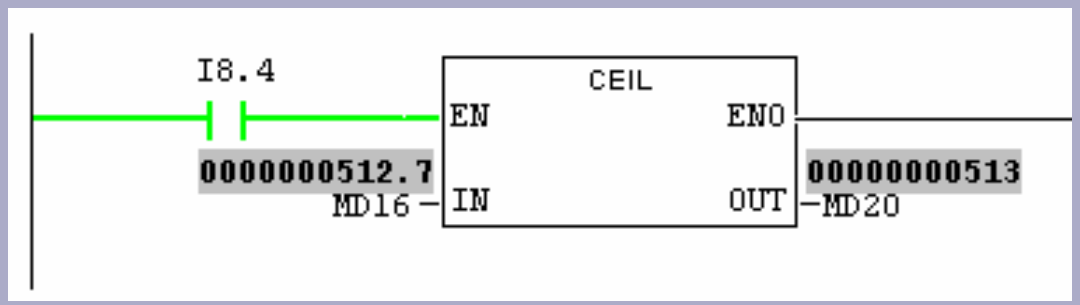


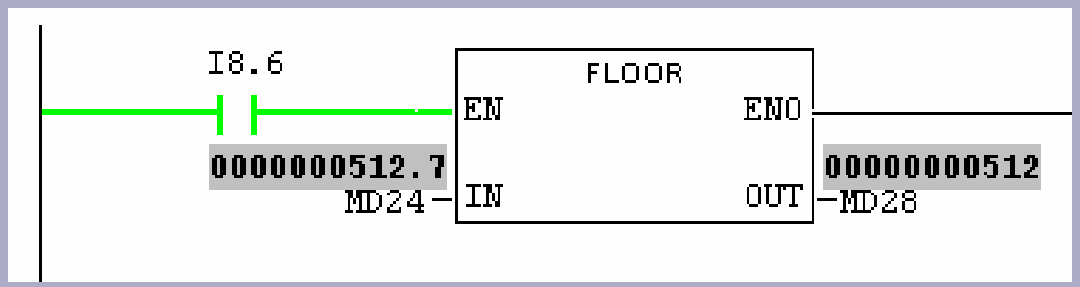
مطابق شکل های روبرو عمل کرده و با تغییر مقدار **MD0** ، مقدار **MD4** را بررسی می کنیم.



POWEREN.IR

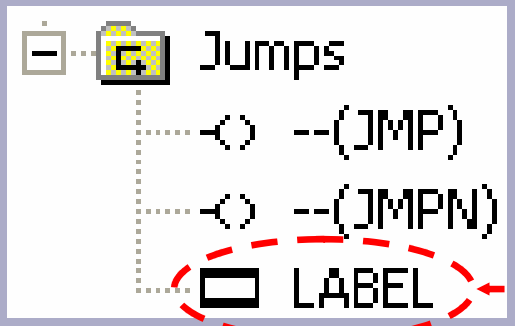


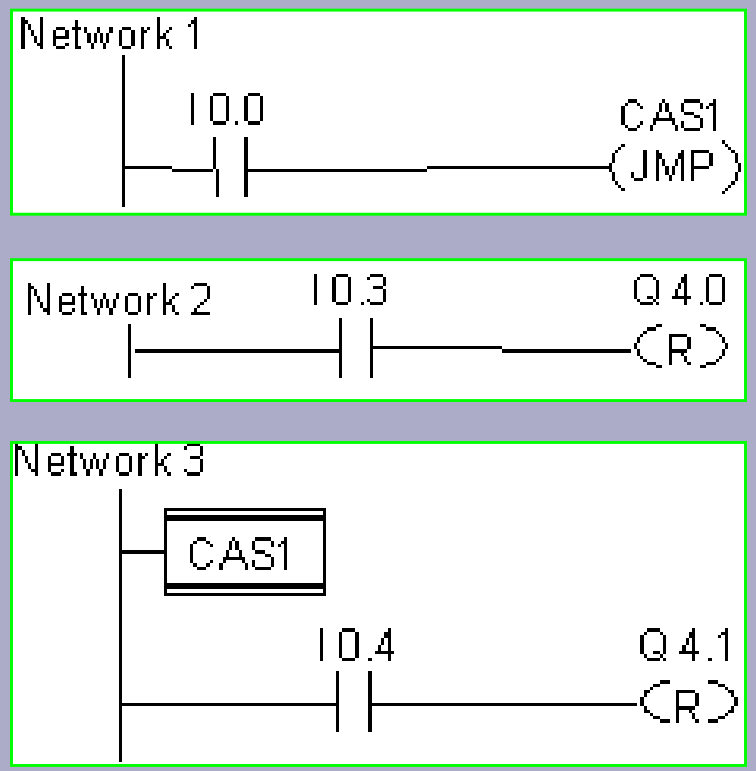




عملیات قبل را برای سه تبدیل کنندهٔ دیگر نیز انجام می دهیم.
مشاهده می شود بلاک CEIL (سقف) عدد اعشاررا سمت بالا رند می کند.
ولی بلاک Floor(کف) عدد اعشار را به سمت پایین رند می نماید.

پرش ها

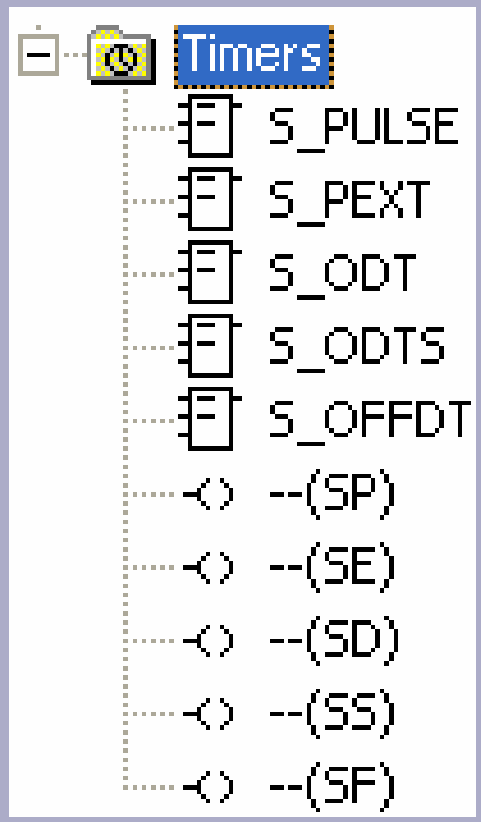




اگر ورودی I0.0 یک باشد، برنامه به برچسبِ CAS1ُ پرش می کند، بدون اینکه Network2َ شود.

تایمرها

در قسمت تایمرها ، تعداد ۵ نوع
به صورت بلاکی و ۴ نوع به صورت
کویلی وجود دارد که هرکدام
را با ذکر یک مثال تشریح می کنیم

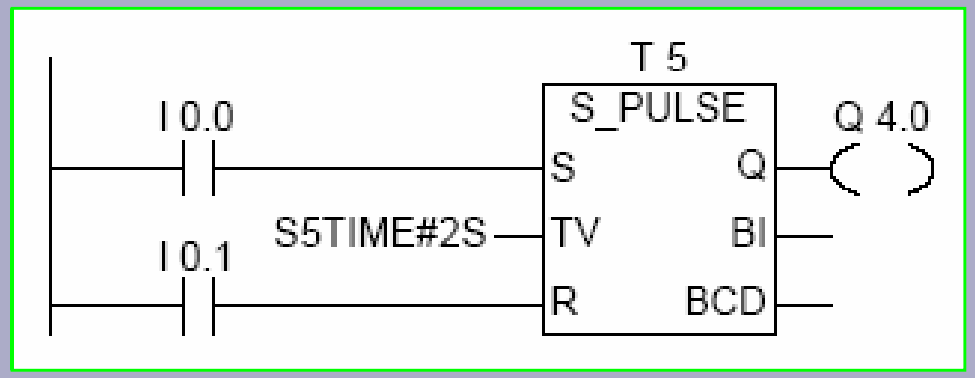


- S_PULSE Pulse S5 Timer
- S_PEXT Extended Pulse S5 Timer
- S_ODT On-Delay S5 Timer
- S_ODTS Retentive On-Delay S5 Timer
- S_OFFDT Off-Delay S5 Timer

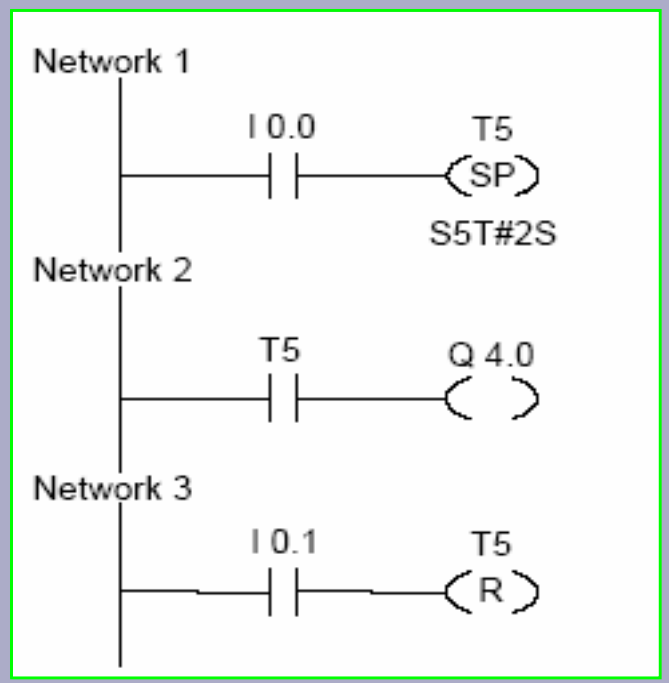
- ---(SP) Pulse Timer Coil
- ---(SE) Extended Pulse Timer Coil
- ---(SD) On-Delay Timer Coil
- ---(SS) Retentive On-Delay Timer Coil
- ---(SF) Off-Delay Timer Coil

لیست هر دو دستهٔ تایمر بلاکی و کویلی مشاهده می گردد.

Pulse Timer



نوع S5

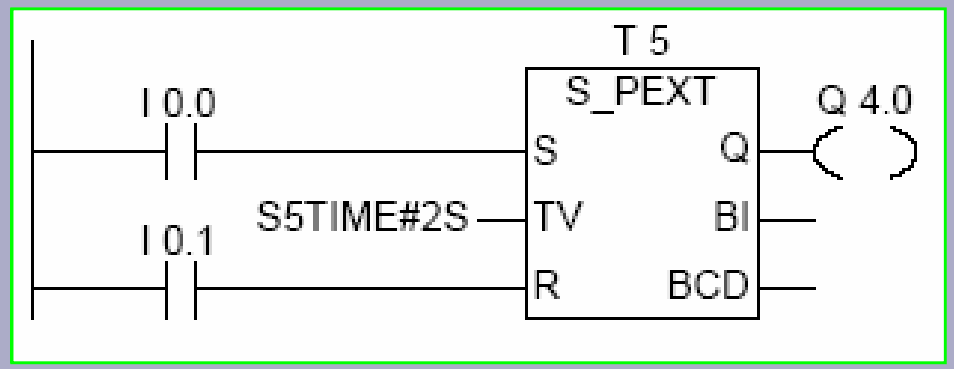


نوع کویلی

اگر ورودی I0.0 روشن شود
تایمر نیز همزمان با آن تا ۲ ثانیه (مادامی که I0.0 یک است) کار می کند. خروجی اش نیز یک می شود.
اگر در حین کارِ تایمر، I0.0 صفر شود
تایمر متوقّف خواهد شد.

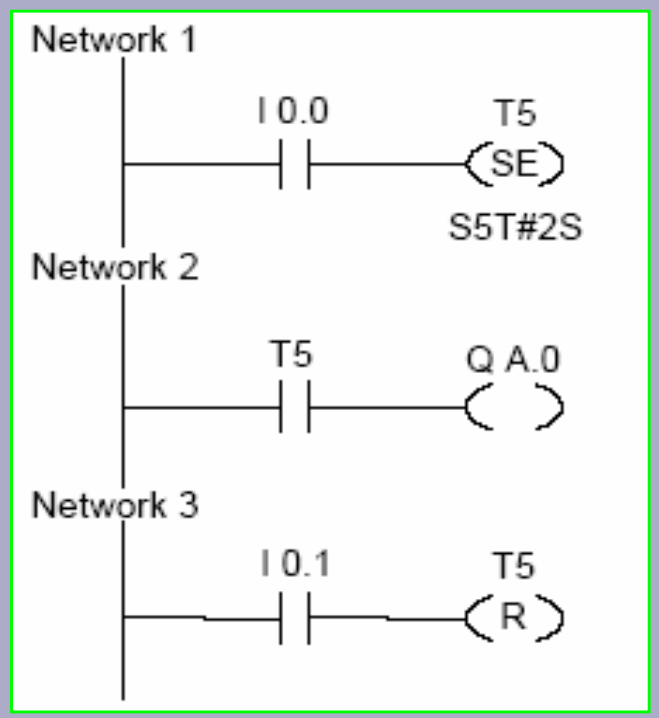
اگر در حین کارِ تایمر ورودی I0.1 زده شود
تایمر ریست می شود.

Extended Pulse Timer



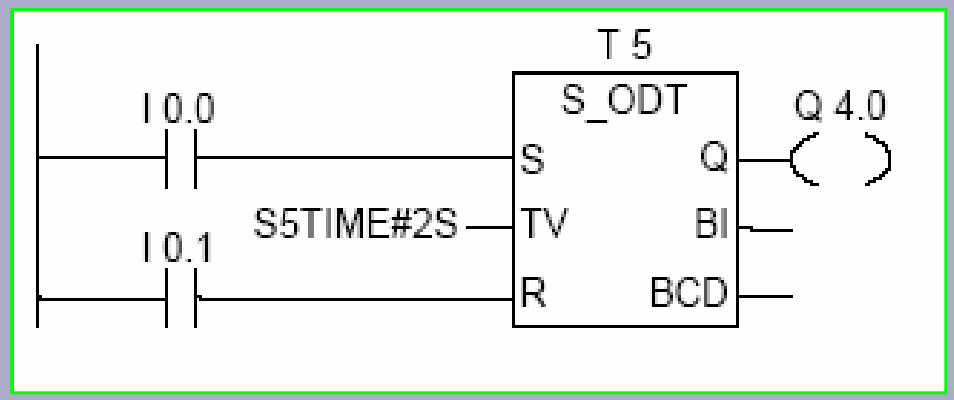
نوع S5

اگر ورودی I0.0 روشن شود،
تایمر نیز همزمان با آن شروع به کار می کند
و به صفر شدن I0.0 اهمیّتی نمی دهد.
اگر در حین کار تایمر مجدداً I0.0 روشن شود،
بلافاصله تایمر از اوّل شروع به کار می کند.
اگر در حین کارِ تایمر ورودی I0.1 زده شود
تایمر ریست می شود.

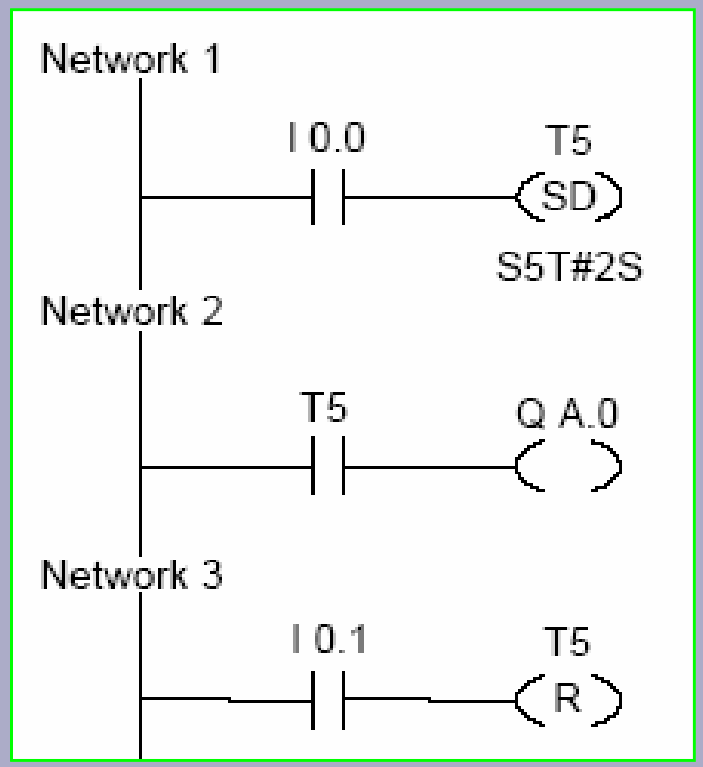


نوع کویلی

On-Delay S5 Timer



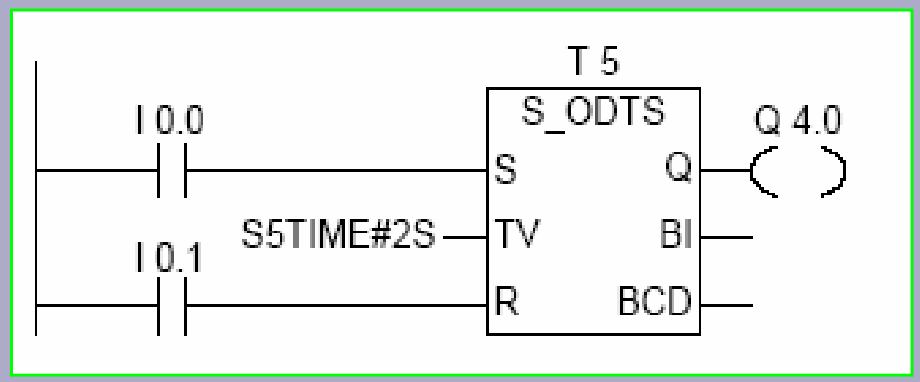
نوع S5



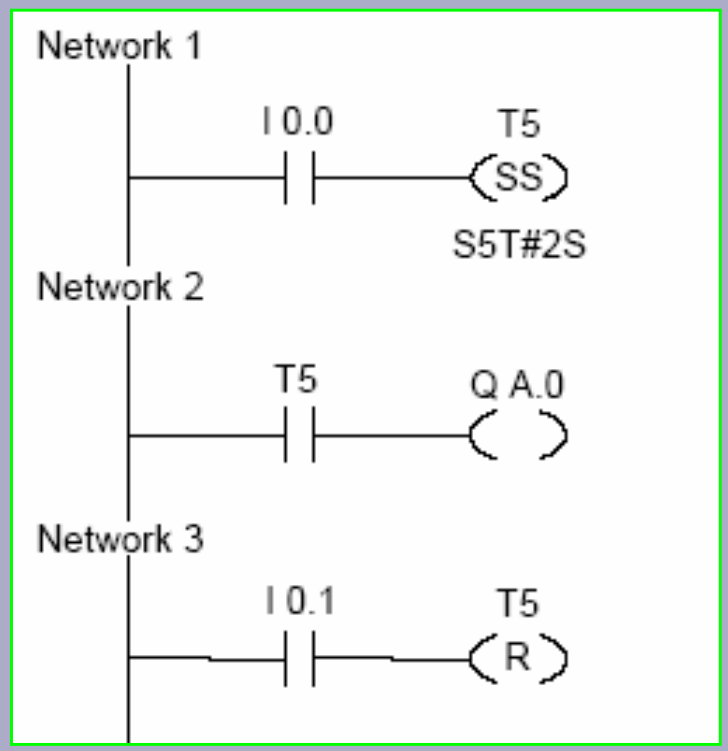
نوع کویلی

اگر ورودی **I0.0** روشن شود،تایمر نیز همزمان با آن تا ۲ ثانیه شروع به کار می کند.
اگر زمان ۲ ثانیه سپری شود و **I0.0** هنوز یک باشد، خروجی، یک خواهد شد.
اگر **I0.0** صفر شود تایمر متوقف شده و خروجی صفر می شود.
اگر ورودی **I0.1** زده شود،
بدون توجه به درحال کار بودن یا نبودن ، تایمر ریست می شود.

Retentive On-Delay Timer



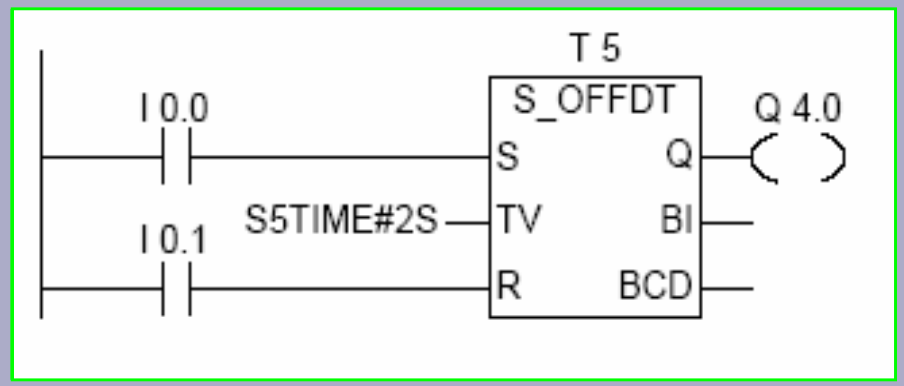
نوع S5



نوع کویلی

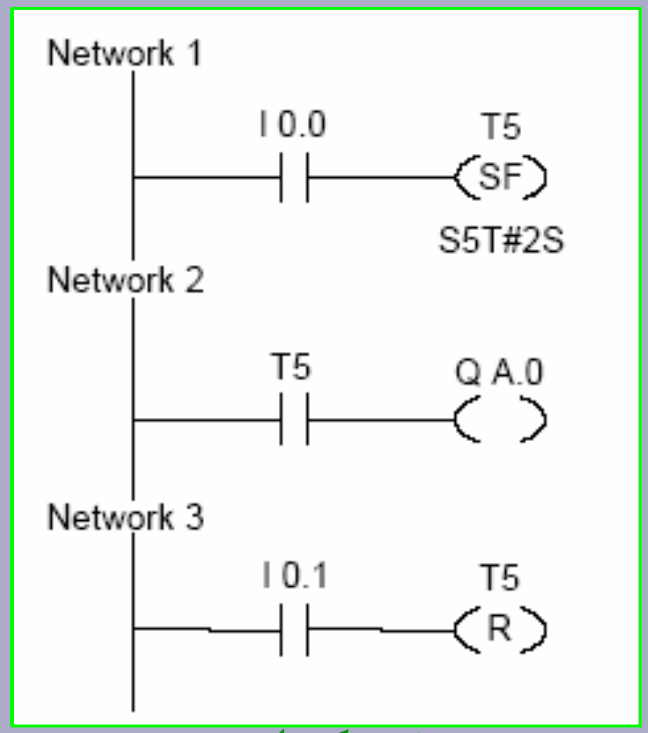
اگر ورودی I0.0 روشن شود،تایمر نیز همزمان با آن شروع به کار می کند.
تایمر بدون توجه به خاموش شدن I0.0 تا ۲ ثانیه کار می کند.
اگر I0.0 قبل از سپری شدن ۲ ثانیه مجدداً روشن شود تایمر از اوّل شروع به کار می کند
بعد از ۲ ثانیه خروجی روشن می شود.
اگر ورودی I0.1 زده شود،
بدون توجه به درحال کار بودن یا نبودن ، تایمر ریست می شود.

Off-Delay Timer Coil



نوع S5

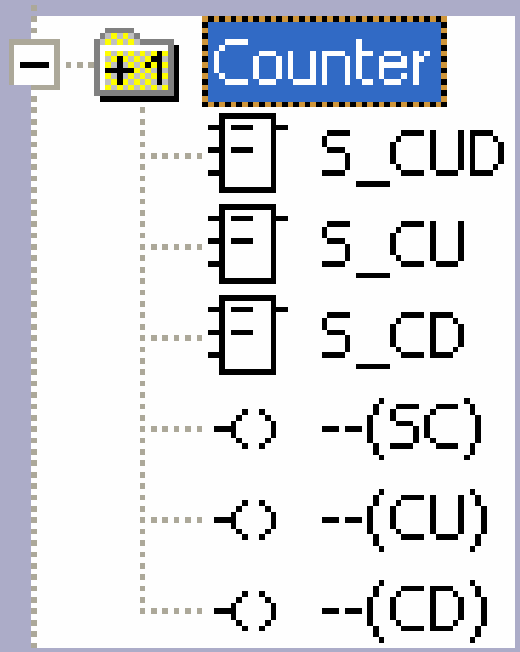
اگر ورودی I0.0 روشن شود،تایمر نیز همزمان با آن شروع به کار می کند.
خروجی هنگامی که تایمر در حال کار است ویا I0.0 روشن باشد، یک می شود.



نوع کویلی

شمارنده ها

در قسمت شمارنده ها ، تعداد ۳ نوع
به صورت بلاکی و ۳ نوع به صورت
کویلی وجود دارد که هرکدام
را با ذکر یک مثال تشریح می کنیم



- S_CUD Up-Down Counter
- S_CD Down Counter
- S_CU Up Counter

- ---(SC) Set Counter Coil
- ---(CU) Up Counter Coil
- ---(CD) Down Counter Coil

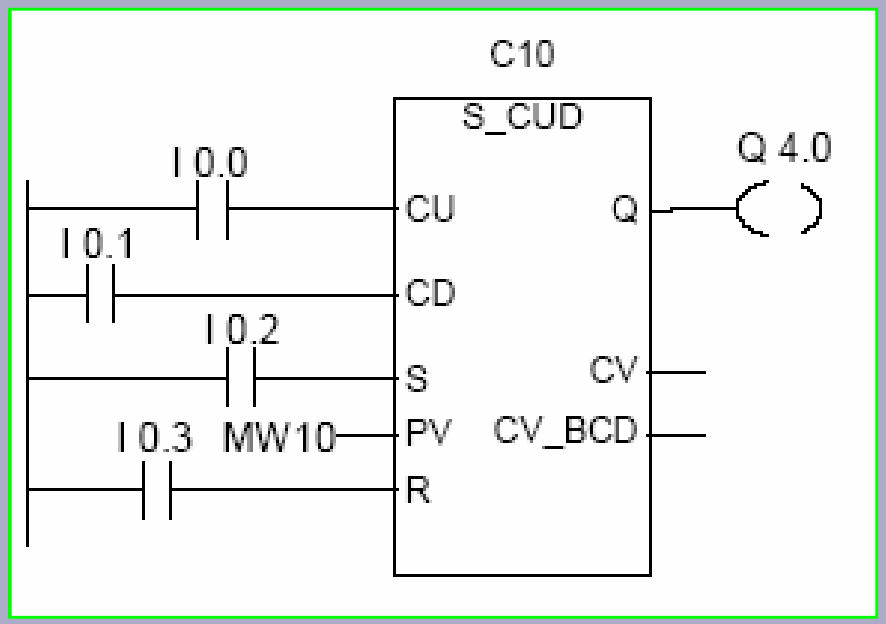
لیست هر دو دستهٔ شمارندهٔ بلاکی و کویلی مشاهده می گردد.

اگر ورودی I0.2 از 0به 1تغییر یابد ،شمارنده با مقدار از پیش تنظیم شدهٔ MW10 پر می شود.

اگر ورودی I0.0 از 0به 1تغییر یابد مقدار شمارندهٔ C10یکی اضافه می شود،مگر اینکه به عدد ۹۹۹رسیده باشد.

اگر ورودی I0.1 از 0به 1تغییر یابد از مقدار شمارنده یکی می شود، مگراینکه مقدار آن صفر شده باشد.

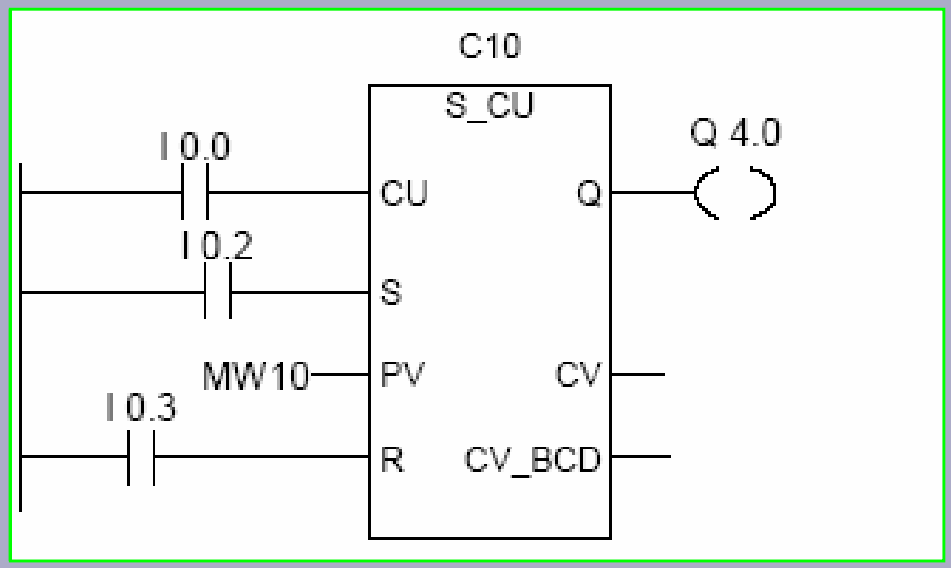
خروجی Q4.0 تا زمانی که مقدار شمارنده صفر نیست، برابر 1 است.



اگر ورودی I0.2 از 0به 1تغییر یابد ،شمارنده با مقدار از پیش تنظیم شدهٔ MW10 پر می شود.

اگر ورودی I0.0 از 0به 1تغییر یابد مقدار شمارندهٔ C10یکی اضافه می شود،مگر اینکه به عدد ۹۹۹رسیده باشد.

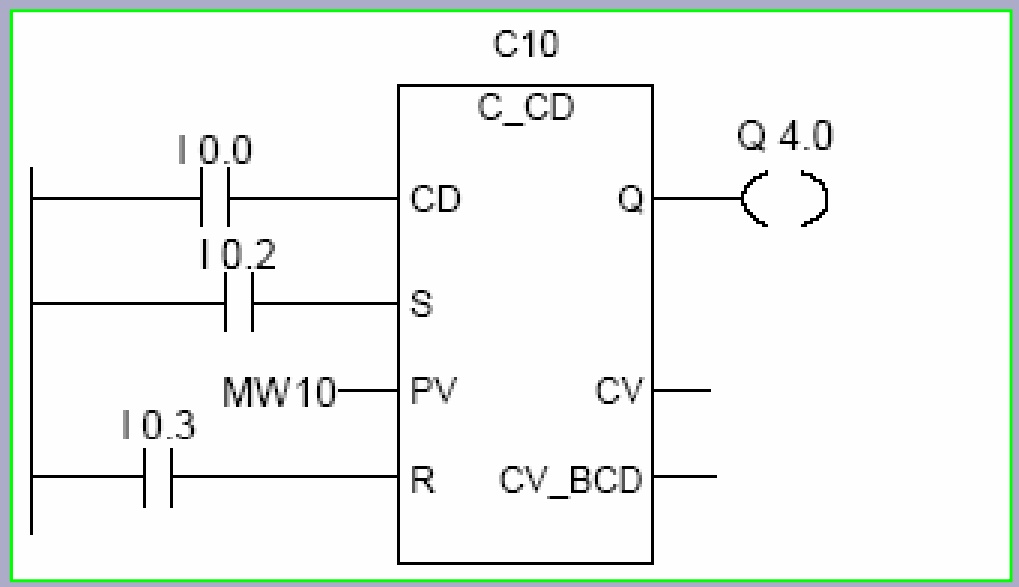
خروجی Q4.0تا زمانی که مقدار شمارنده صفر نیست، برابر 1 است.

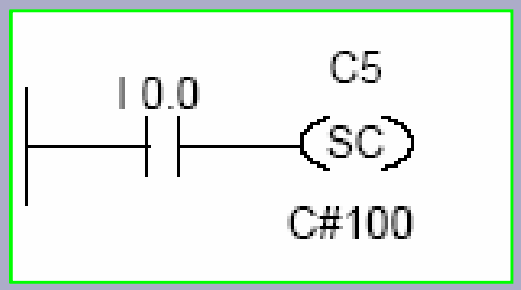


اگر ورودی I0.2 از 0به 1تغییر یابد ،شمارنده با مقدار از پیش تنظیم شدهٔ MW10 پر می شود.

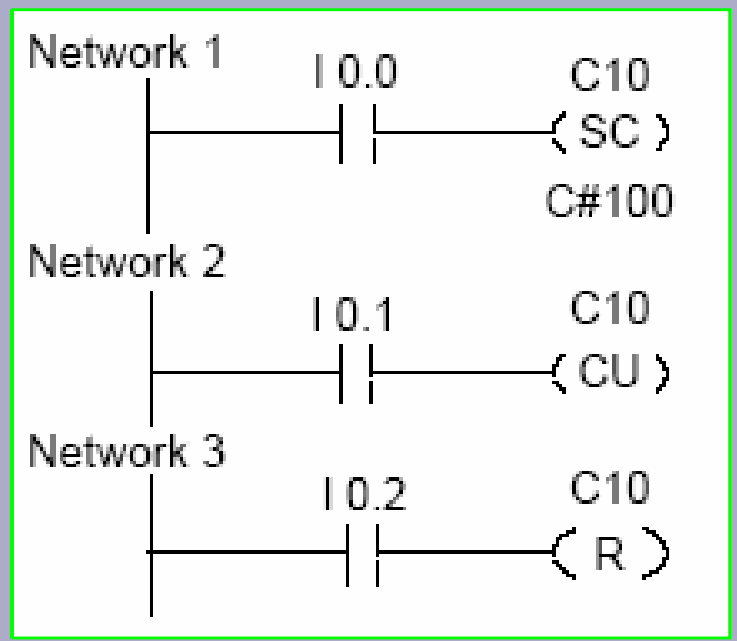
اگر ورودی I0.0 از 0به 1تغییر یابد از مقدار شمارنده یکی کم می شود، مگراینکه مقدار آن صفر شده باشد.

خروجی Q4.0تا زمانی که مقدار شمارنده صفر نیست، برابر 1 است.





اگر ورودی I0.0 از صفر به یک تغییر یابد،
شمارندۀ C5 روی مقدار ۱۰۰ تنظیم می شود.
اگر I0.0 از 0 به 1 تغییر نیابد، شمارنده
بدون تغییر باقی می ماند.



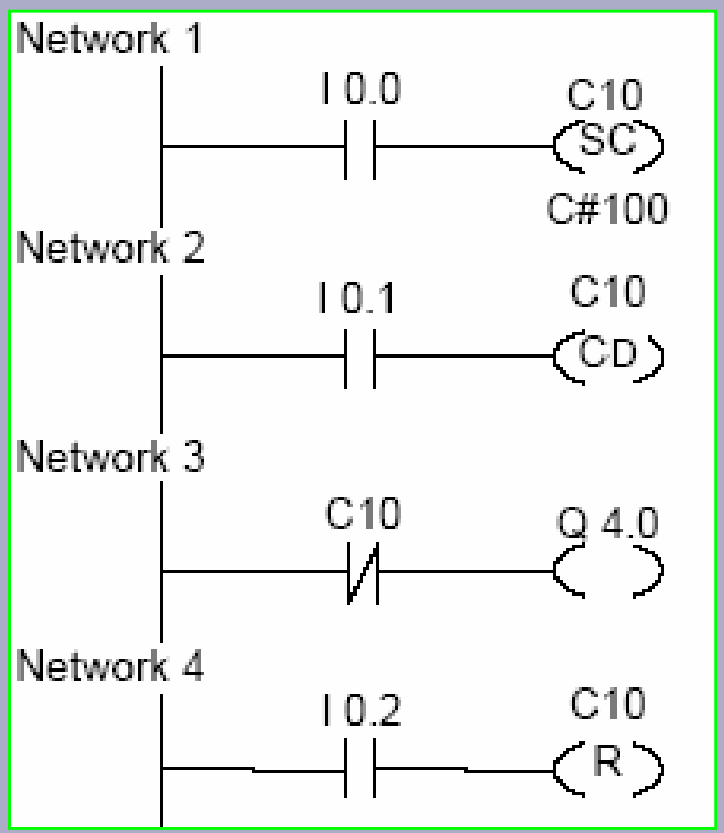
اگر ورودی I0.0 از صفر به یک تغییر یابد،
شمارندۀ C5 روی مقدار ۱۰۰ تنظیم می شود.
اگر ورودی I0.1 از 0به 1تغییر یابد مقدار شمارندۀ C10
یکی اضافه می شود،مگر اینکه به عدد ۹۹۹رسیده باشد.
اگر I0.0 از 0 به 1 تغییر نیابد، شمارنده
بدون تغییر باقی می ماند.
اگر I0.2 یک شود، شمارنده ریست می شود.

اگر ورودی I0.0 از 0به 1 تغییر یابد ، شمارنده با مقدار ۱۰۰ پر می شود.

اگر ورودی I0.1 از 0به 1 تغییر یابد، از مقدار شمارنده یکی کم می شود، مگراینکه مقدار آن صفر شده باشد.

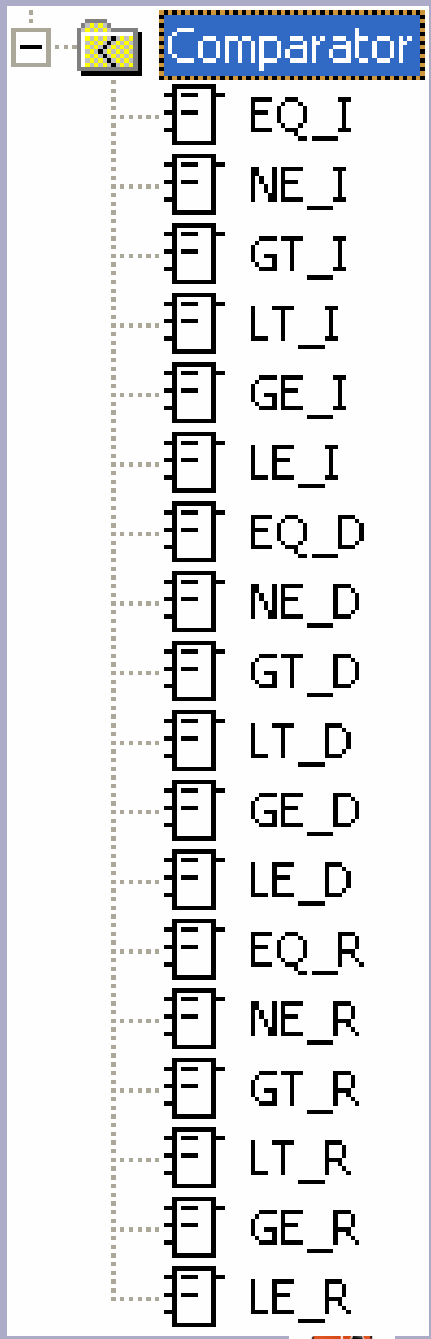
اگر مقدار شمارنده صفر باشد، خروجی Q4.0 برابر 1 می شود.

اگر I0.2 یک شود، شمارنده ریست می شود.



مقایسه گر ها

در بخش مقایسه گر ها ، تعداد ۱۸ المان
به صورت بلاکی وجود دارد ،
که تک تک آنها را
را با ذکر یک مثال تشریح می کنیم

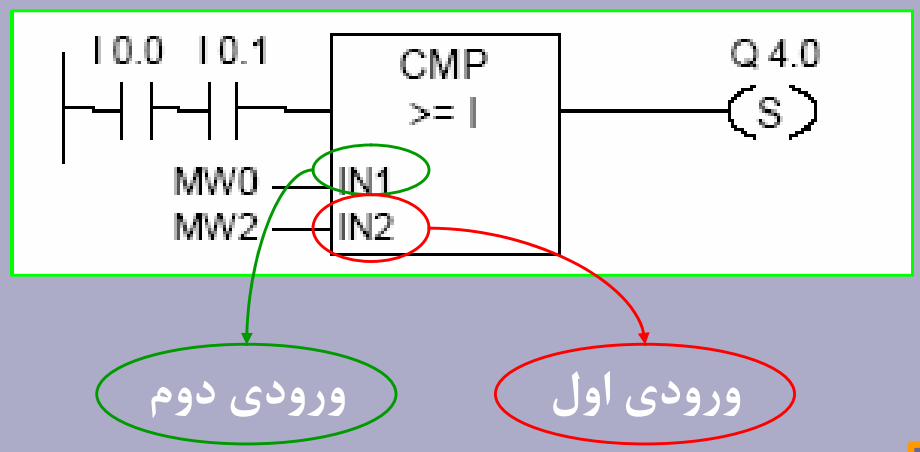


دو ورودی با هم برابر باشند	==
دو ورودی با هم برابر نباشند	<>
ورودی اوّل از ورودی دوّم بزرگتر باشد	>
ورودی اوّل از ورودی دوّم کوچکتر باشد	<
ورودی اوّل، بزرگتر یا مساوی ورودی دوّم باشد	>=
ورودی اوّل، کوچکتر یا مساوی ورودی دوّم باشد	<=

ورودی ها ممکن است از نوع:

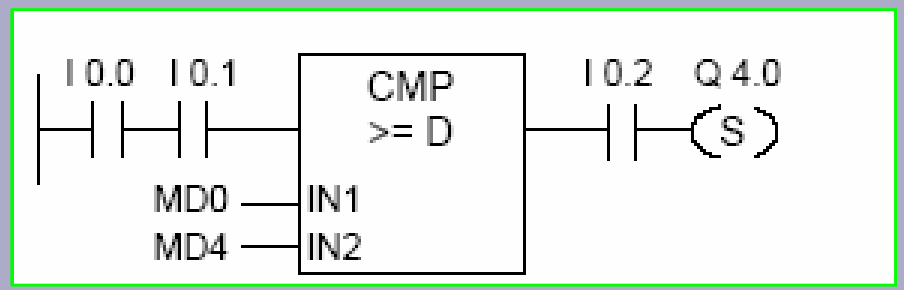
Integer یا عدد صحیح باشند که با I نشان داده می شوند
Double Integer باشند که با DI نشان داده می شوند.
عدد حقیقی یا Real باشند که با R نشان داده می شوند.

ورودی های نوع I



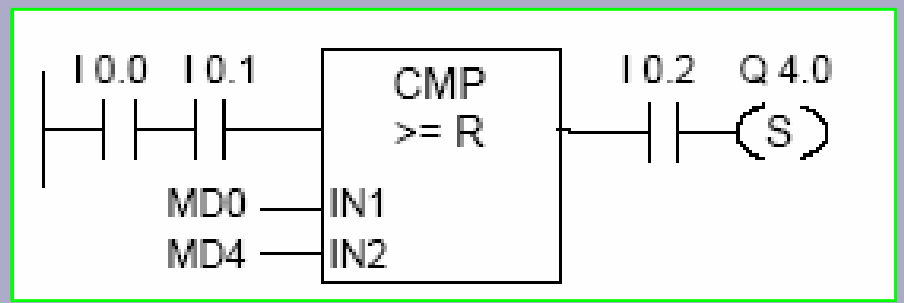
خروجی Q4.0 هنگامی ست می شود که:
ورودی های I0.0,I0.1 هر دو یک باشند.
MW0 بزرگتر یا مساوی MW2 باشد.

ورودی های نوع DI



خروجی Q4.0 هنگامی ست می شود که:
ورودی های I0.0,I0.1 هر دو یک باشند.
MD0 بزرگتر یا مساوی MD4 باشد.
I0.2 نیز یک باشد

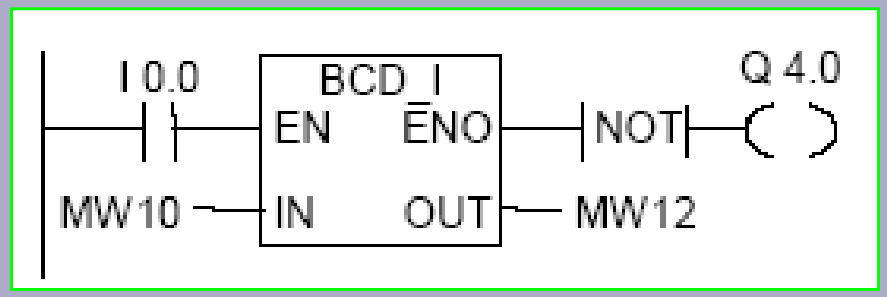
ورودی های نوع R



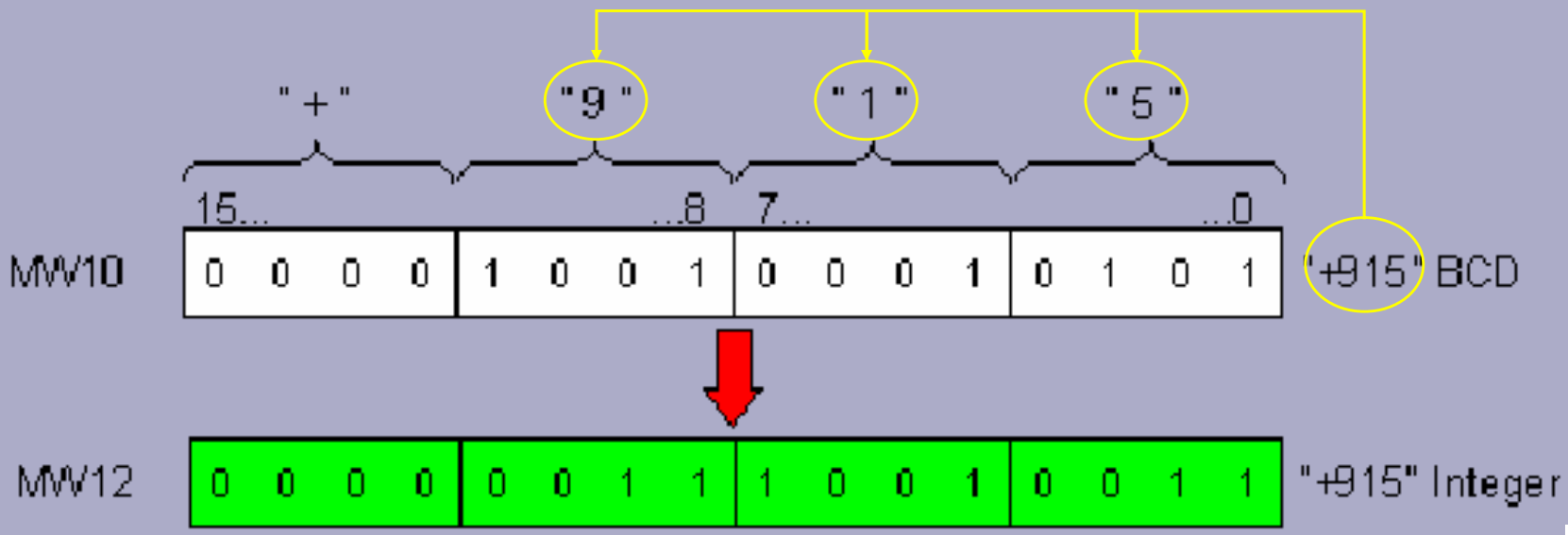
خروجی Q4.0 هنگامی ست می شود که:
ورودی های I0.0,I0.1 هر دو یک باشند.
MD0 بزرگتر یا مساوی MD4 باشد.
I0.2 نیز یک باشد.

تبدیل BCD به عدد صحیح (Integer)

اگر ورودی I0.0 یک باشد، محتویّات MW10 به عنوان یک BCD خوانده می شود و به یک عدد صحیح (Integer) تبدیل می شود. نتیجه در MW12 ذخیره می شود. خروجی Q4.0 هنگامی که تبدیل انجام نشده باشد (به خاطر وجود NOT)، یک می باشد.

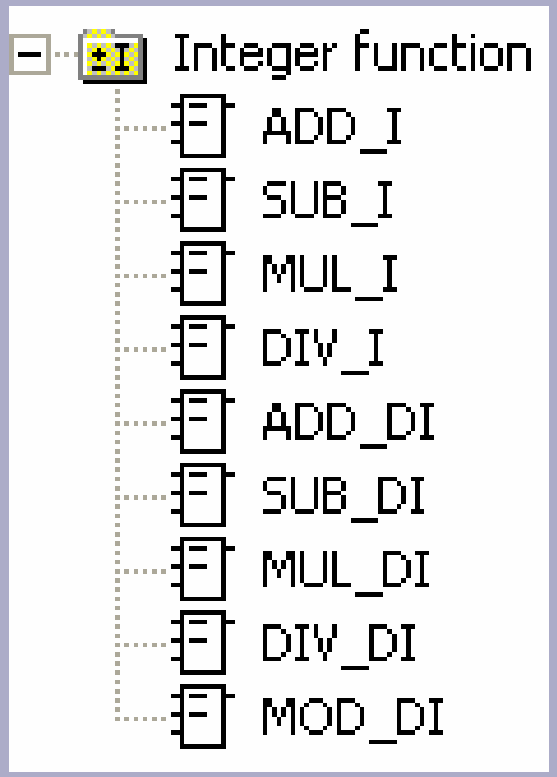


به عنوان مثال می خواهیم عدد ۹۱۵ که به صورت BCD میباشد به عدد صحیح تبدیل کنیم.

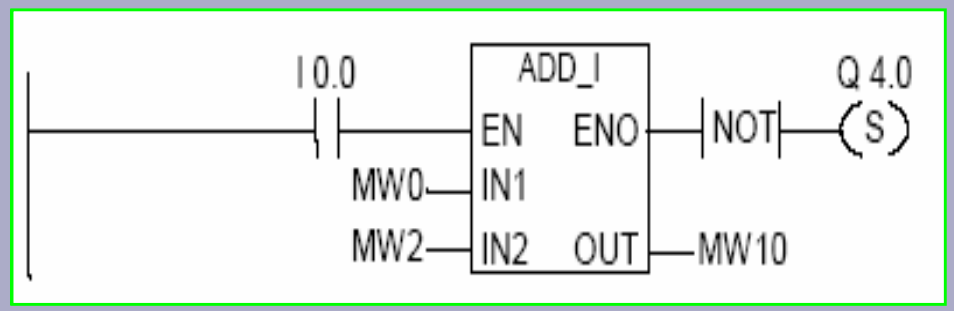


عملیات ریاضی

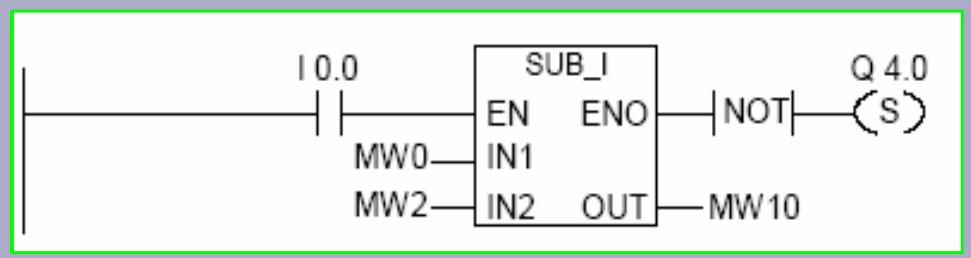
در قسمت عملیات ریاضی روی اعداد صحیح تعداد نه عملگر وجود دارد که با ذکر مثال آن ها را تشریح می کنیم.



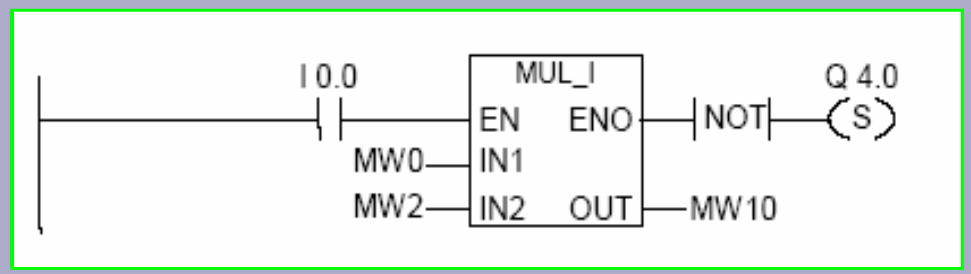
• ADD_I	جمع عدد صحیح
• SUB_I	تفریق عدد صحیح
• MUL_I	ضرب عدد صحیح
• DIV_I	تقسیم عدد صحیح
• ADD_DI	جمع عدد صحیح دوبل
• SUB_DI	تفریق عدد صحیح دوبل
• MUL_DI	ضرب عدد صحیح دوبل
• DIV_DI	تقسیم عدد صحیح دوبل
• MOD_DI	



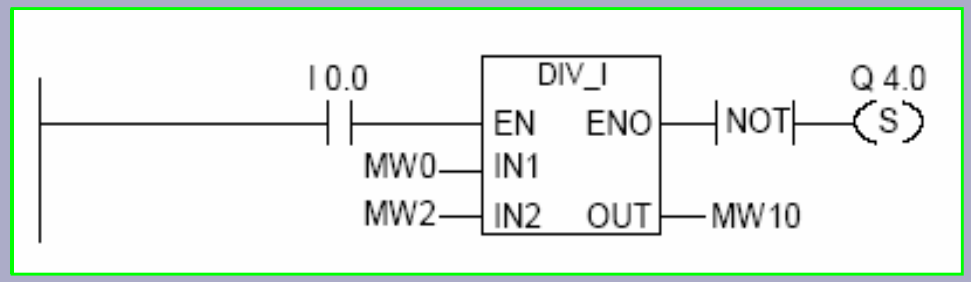
اگر ورودی I0.0 یک باشد، نتیجهٔ جمع MW0+MW2 به خروجی MW10 ریخته می شود.
اگر نتیجه از محدودهٔ مجاز اعداد صحیح خارج باشد، خروجی Q4.0 سِت می گردد.



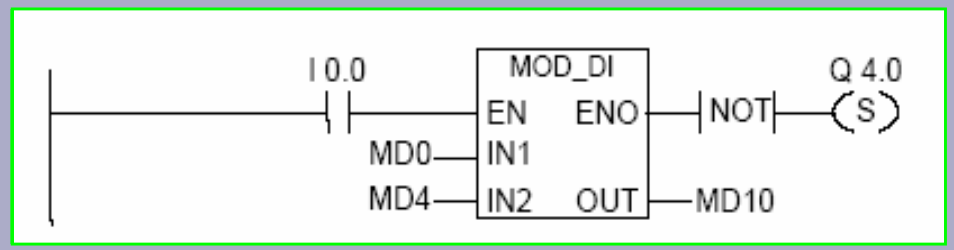
اگر ورودی I0.0 یک باشد، نتیجهٔ تفریق MW0-MW2 به خروجی MW10 ریخته می شود.
اگر نتیجه از محدودهٔ مجاز اعداد صحیح خارج باشد، خروجی Q4.0 سِت می گردد.



اگر ورودی I0.0 یک باشد، نتیجهٔ ضرب MW0 x MW2 به خروجی MW10 ریخته می شود.
اگر نتیجه از محدودهٔ مجاز اعداد صحیح خارج باشد، خروجی Q4.0 سِت می گردد.



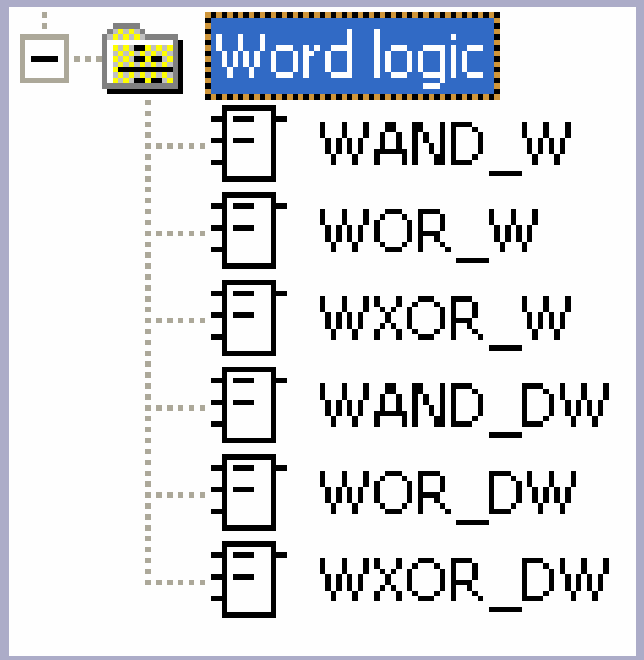
اگر ورودی I0.0 یک باشد، نتیجهٔ تقسیم MW0 بر MW2 به خروجی MW10 ریخته می شود.
اگر نتیجه از محدودهٔ مجاز اعداد صحیح خارج باشد، خروجی Q4.0 سِت می گردد.



اگر ورودی I0.0 یک باشد، باقیماندهٔ تقسیم MD0 بر MD2 به خروجی MD10 ریخته می شود.
اگر نتیجه از محدودهٔ مجاز اعداد صحیح خارج باشد، خروجی Q4.0 سِت می گردد.

عملیات منطقی روی WORD

در قسمت عملیات منطقی روی WORD
تعداد شش بلاک وجود دارد که با ذکر
مثال آن ها را تشریح می کنیم.

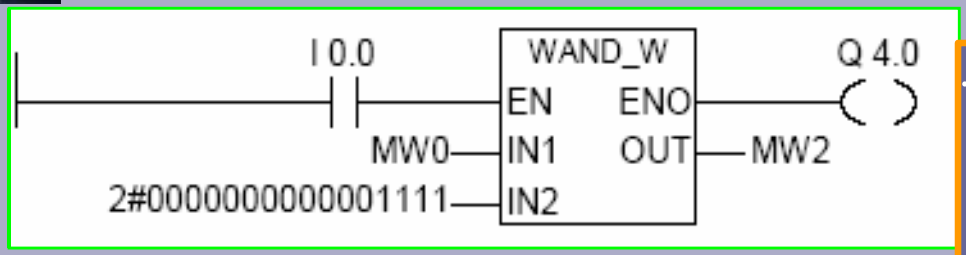


دستورات منطقی WORD، دو جفت WORD یا دو جفت Double Word
را بیت به بیت بر اساس منطق بولی مقایسه می کند.
اگر نتیجهٔ که در خروجی OUT ریخته می شود صفر نباشد بیت CC1 سِت می شود.
ولی اگر صفر باشد CC1 ریست می شود.

- WAND_W (Word) AND Word
- WOR_W (Word) OR Word
- WXOR_W (Word) Exclusive OR Word
- WAND_DW (Word) AND Double Word
- WOR_DW (Word) OR Double Word
- WXOR_DW (Word) Exclusive OR Double Word

اِلمان های برنامه

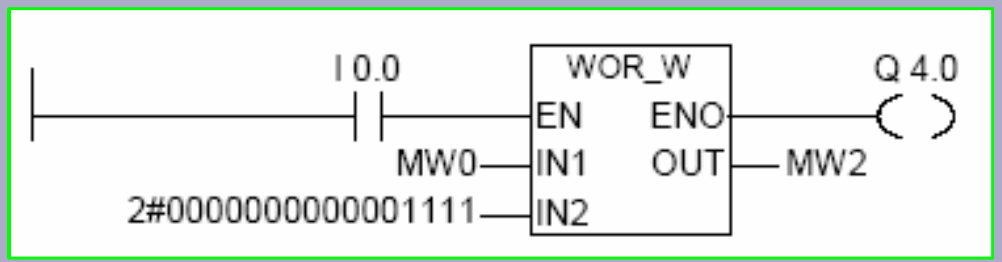
حالت AND دو W



اگر ورودی I0.0 یک باشد، دستورالعمل اجرا می شود.
بیت ها، تک تک با هم AND می شوند نتیجه به MW2 ریخته می شود.
اگر بر نامه اجرا شود، Q4.0 یک می شود.

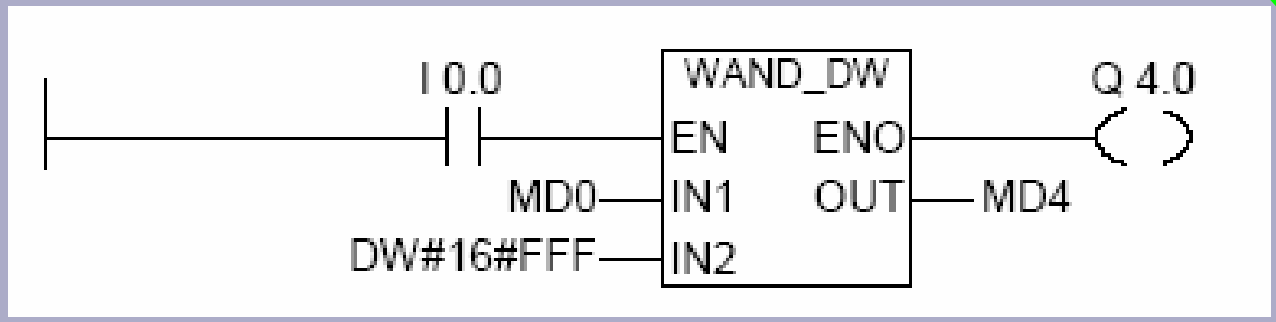
MW0	01010101 01010101			
	AND	=		
IN2	00000000 00001111		00000000 00000101	MW2

حالت OR دو W



MW0	01010101 01010101			
	OR	=		
IN2	00000000 00001111		01010101 01011111	MW2

حالت AND دو DW



MD0 01010101 01010101 01010101 01010101

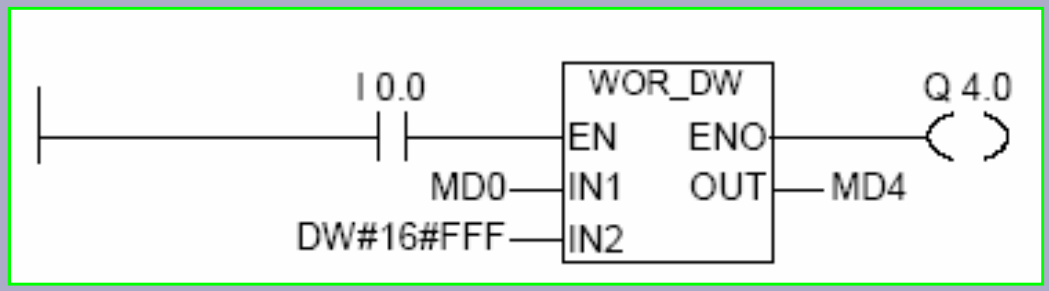
AND

IN2 (FFF) 00000000 00000000 00001111 11111111

=

MD2 00000000 00000000 00000101 01010101

حالت OR دو DW



MD0 01010101 01010101 01010101 01010101

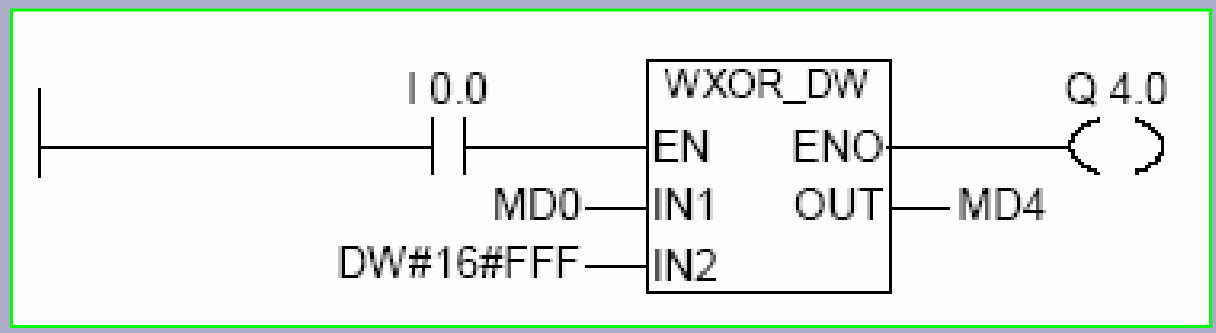
OR

IN2 (FFF) 00000000 00000000 00001111 11111111

=

MD4 01010101 01010101 01011111 11111111

حالت XOR دو DW



MD0 01010101 01010101 01010101 01010101

OR

IN2 (FFF) 00000000 00000000 00001111 11111111

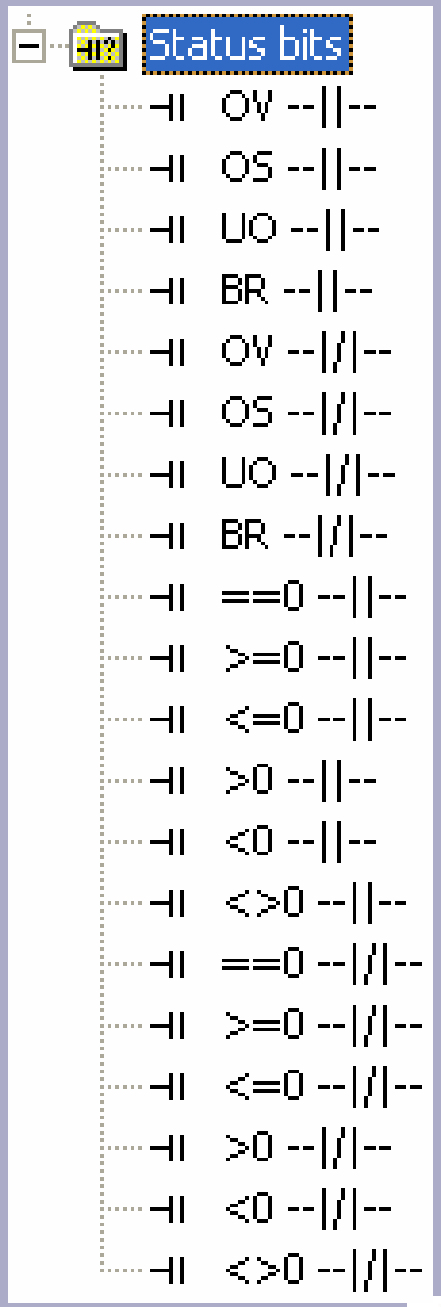
=

MD4 01010101 01010101 01011010 10101010

دستورالعمل های بیت وضعیّت
Status Bit

POWEREN.IR

در قسمت دستورالعملِ بیت وضعیّت،
تعداد بیست المان وجودَ دارد که با ذکر
مثال آن ها را تشریح می کنیم.



بیت وضعیّت (Status Bit)

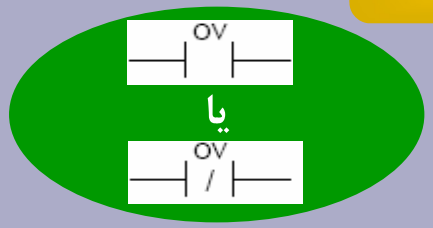
بیت وضعیّت رجیستری در حافظهٔ CPU است،
که حاوی بیت هایی است که می توانید آنها را در آدرس دهی
بیت ها و دستورالعمل منطقی روی Wordها
به عنوان مرجع در نظر بگیرید.
ساختار بیت وضعیّت به صورت زیر است:

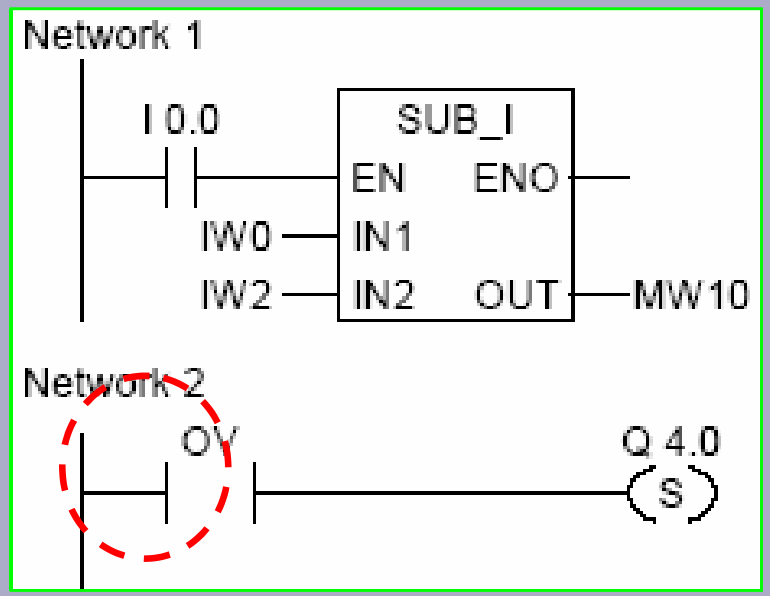
2^{15} ...	...2^{9}	2^{8}	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}
		BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC

دستورالعمل های بیت وضعیّت، دستورالعمل های بیت لاجیکی
هستند که با بیت وضعیّت کار می کنند.
هر کدام از این دستورالعمل ها روی یکی از شرایط زیر اثر می گذارد:

بیت نتیجهٔ باینری (---I I--- BR) ست شود.
عمل ریاضی سر ریز داشته باشد (---I I--- OV) یا سر ریز ذخیره شده (---I I--- OS) داشته باشد.
نتیجهٔ دستور ریاضی نا مرتّب باشد: (---I I--- UO)
نتیجهٔ دستور ریاضی با یکی از راه های زیر با 0 ارتباط داشته باشد:

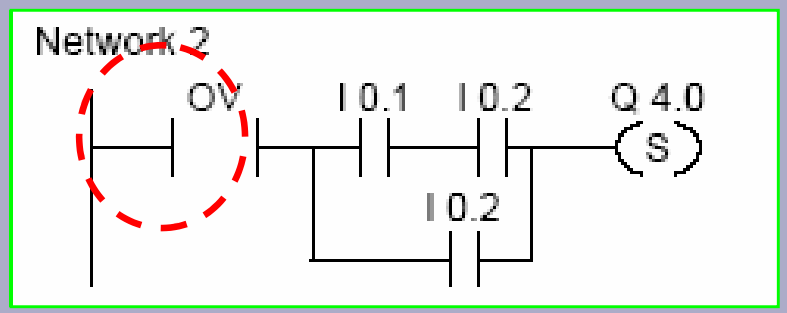
== 0, <> 0, > 0, < 0, >= 0, <= 0.

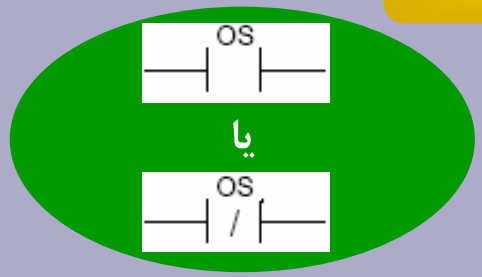


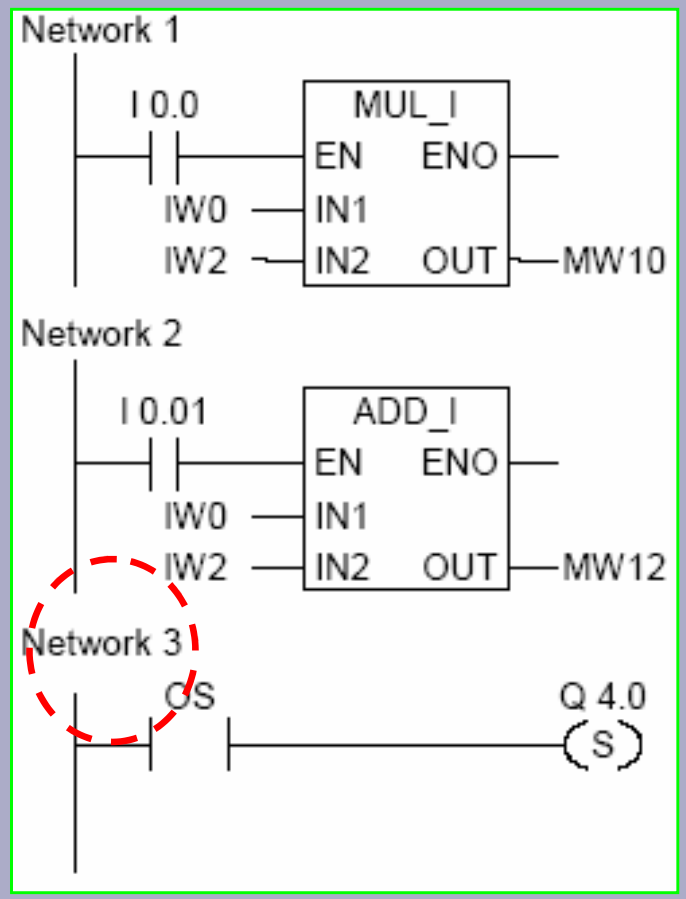


عمل تفریق با یک بودن ورودی I0.0 فعّال می شود.
اگر نتیجهٔ تفریق (IW2 – IW0) خارج محدودهٔ مجاز یک عدد صحیح باشد، بیت OV ست می شود.
با ست شدن OV خروجی Q4.0 یکَ می شود.

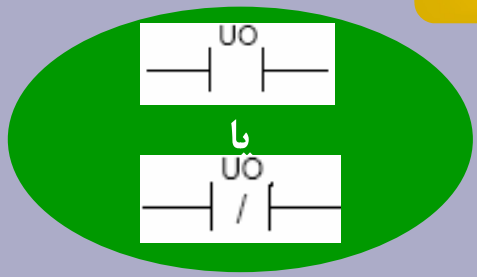
می توانستیم تنها از خروجی ENO استفاده کنیم که در صورت خارج از محدوده بودن تفریق، صفر می شود. ولی ممکن است مانند شکل زیر شرایط دیگری را برای خروجی Q4.0 در نظر داشته باشیم، که در اینصورت باید از بیت وضعیّت OV در یک Network مجزّا استفاده نماییم.

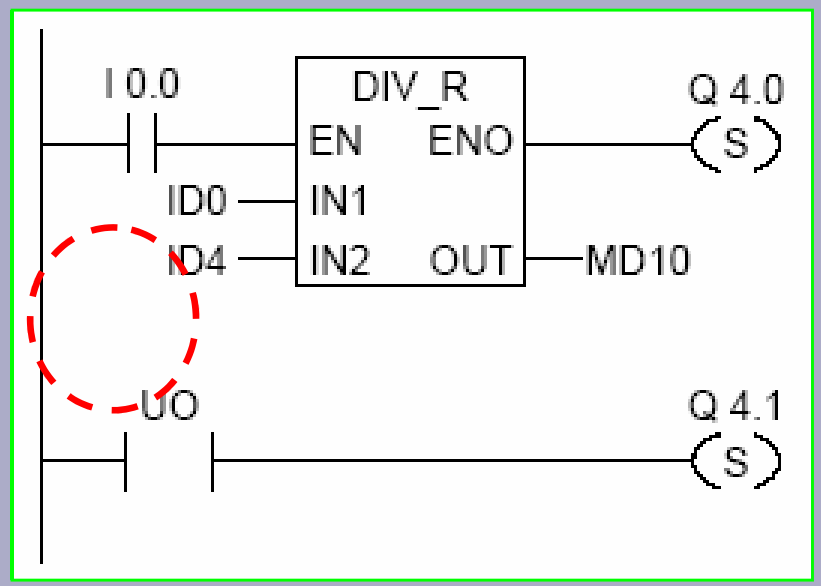




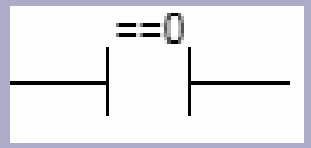


عمل ضرب با یک بودن ورودی I0.0 و عمل جمع با ورودی I0.1 فعّال می شود.
اگر نتیجهٔ یکی از دو عمل ریاضی ، خارج محدودهٔ مجازِ یک عدد صحیح باشد، بیت OS ست می شود.
با ست شدن OS خروجی Q4.0 یکَ می شود.

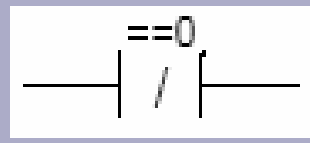


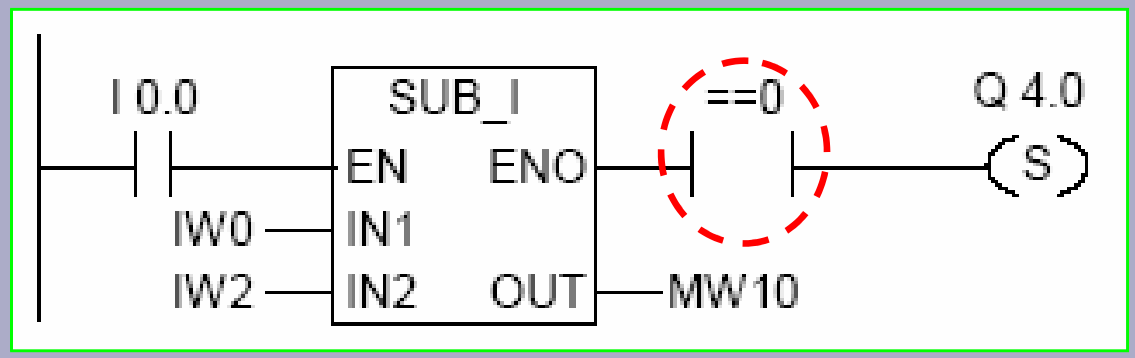


عمل تقسیم دو عدد حقیقی با یک بودن ورودی I0.0 فعّال می شود.
اگر یکی مقادیر عدد غیر مجاز باشد،
بیت UO ستِ می شود.
با ست شدن UO خروجَی Q4.0 یک می شود.



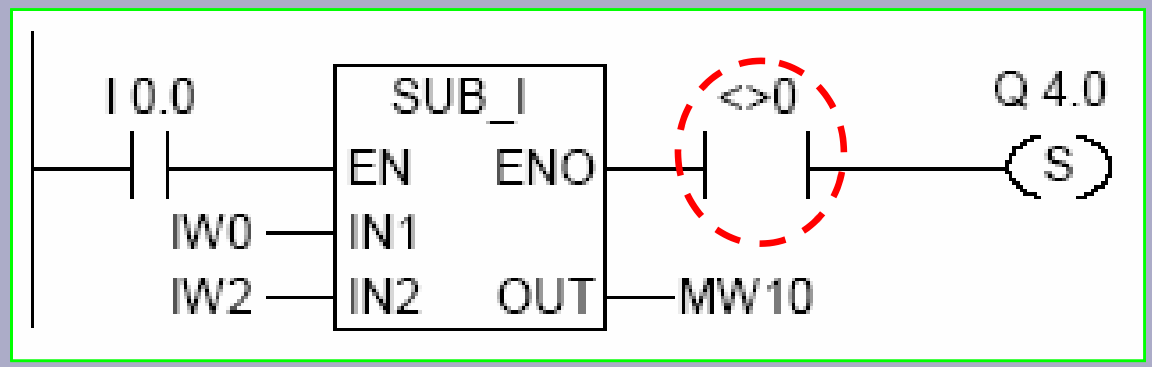
یا



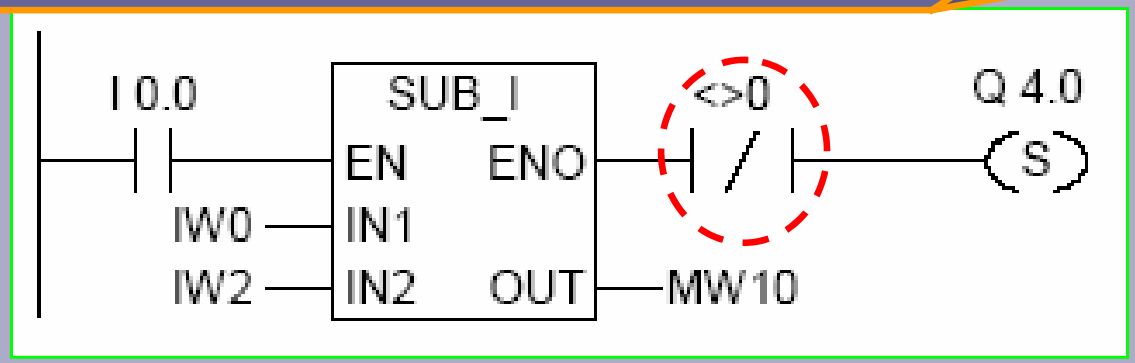


اگر نتیجهٔ عمل تفریق صفر باشد(دو ورودی باهم برابر باشند، خروجی **Q4.0** ست می شود.

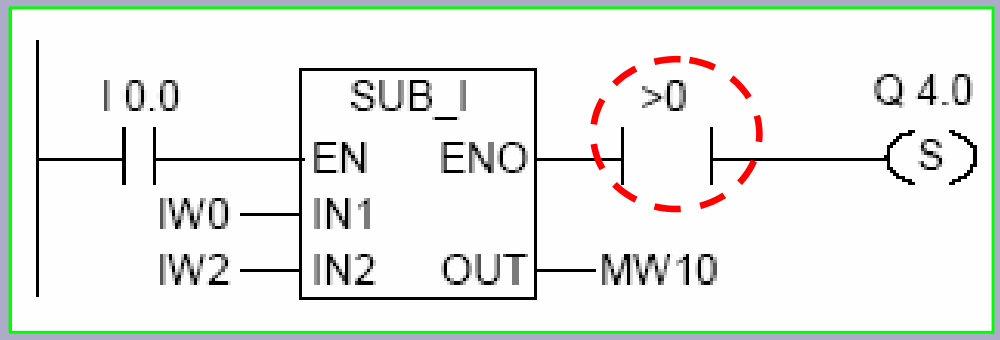
اگر برنامه به خوبی اجرا شود و نتیجهٔ عمل تفریق مخالف صفر باشد خروجی Q4.0 ست می شود.



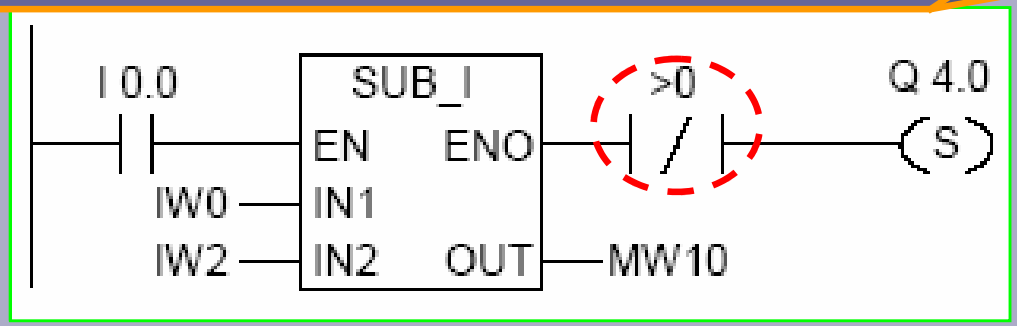
اگر برنامه به خوبی اجرا شود و نتیجهٔ عمل تفریق صفر باشد خروجی Q4.0 ست می شود.



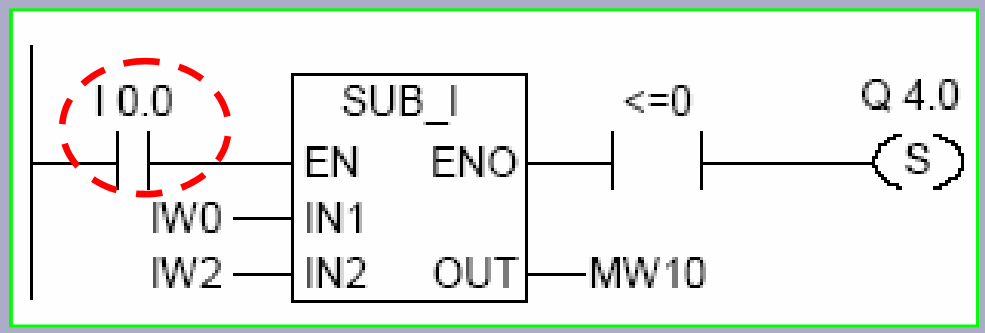
اگر برنامه به خوبی اجرا شود و نتیجهٔ عمل تفریق بزرگتر از صفر باشد خروجی **Q4.0** ست می شود.



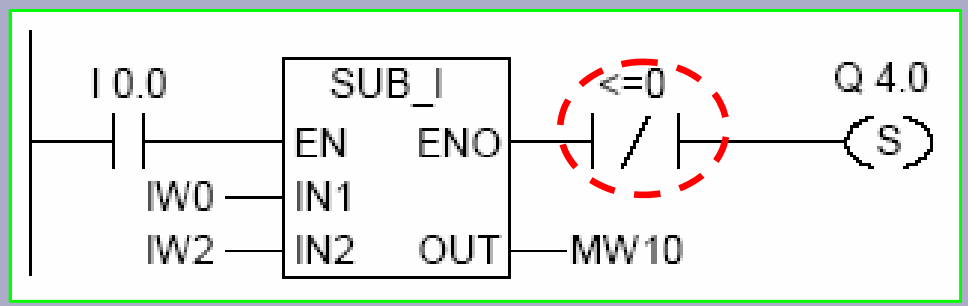
اگر برنامه به خوبی اجرا شود و نتیجهٔ عمل تفریق مخالف صفر نباشد خروجی **Q4.0** ست می شود.



اگر برنامه به خوبی اجرا شود و نتیجهٔ عمل تفریق کوچکتر یامساوی صفر باشد خروجی Q4.0 ست می شود.

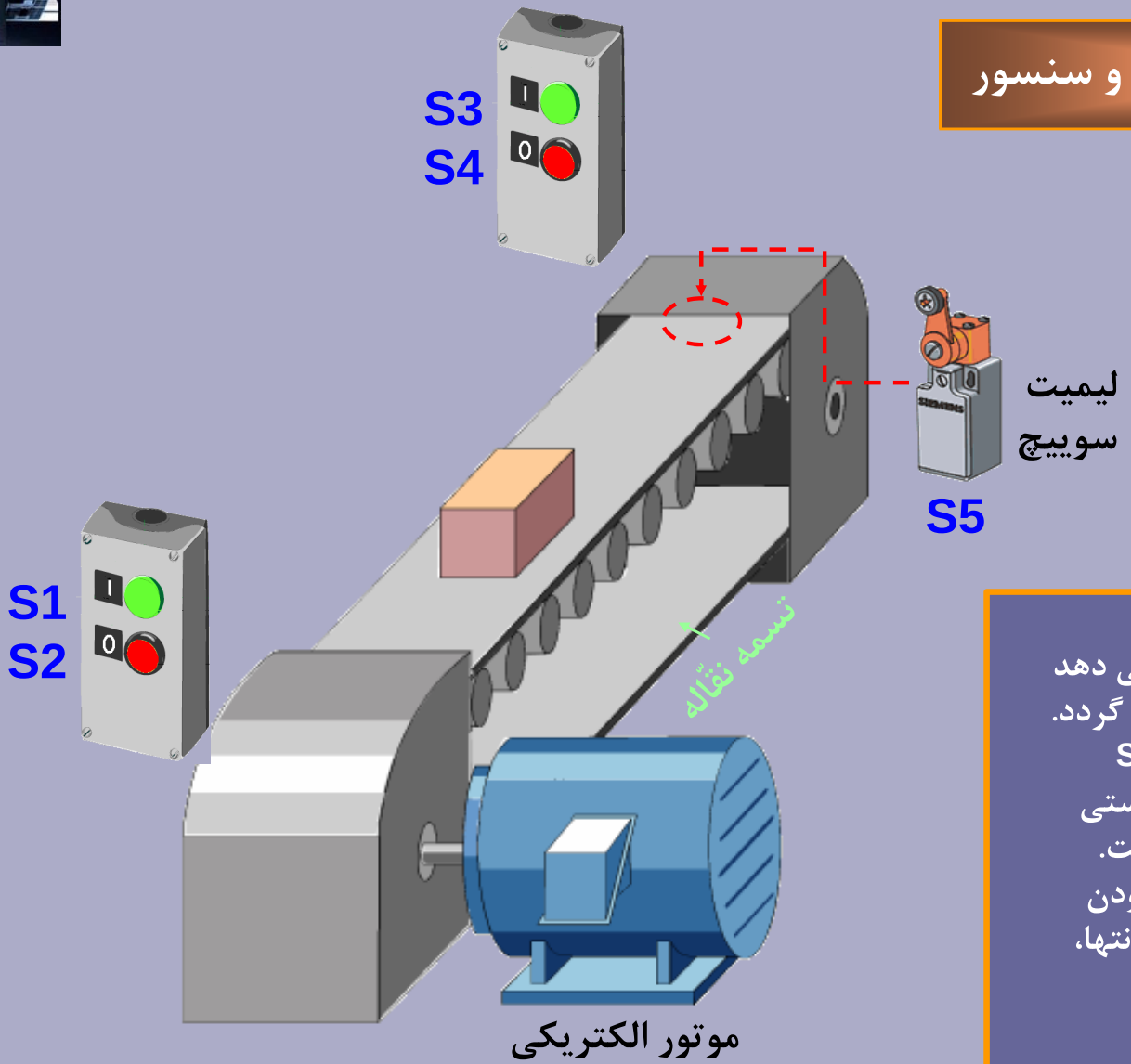


اگر برنامه به خوبی اجرا شود و نتیجهٔ عمل تفریق کوچکتر یامساوی صفر نباشد خروجی Q4.0 ست می شود.



مثال های کاربردی

کنترل تسمه نقاله توسط کلید و سنسور

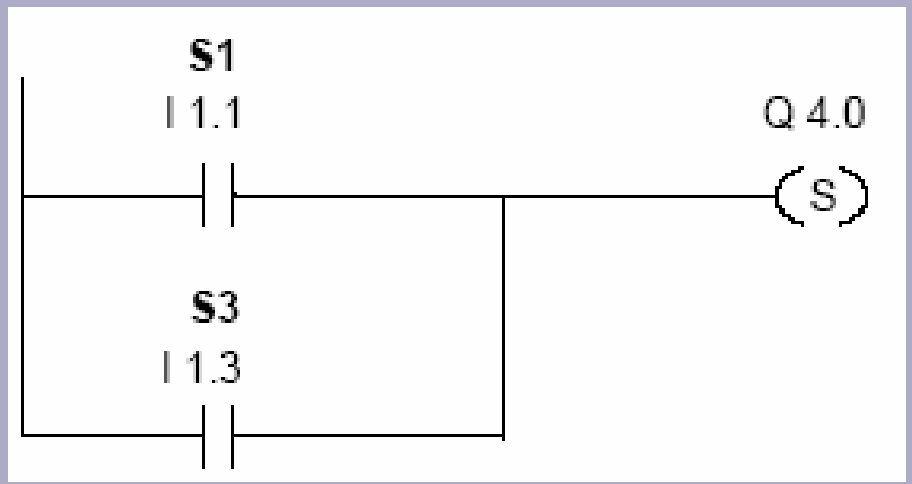


شکل مقابل تسمه نقّاله ای را نشان می دهد که می تواند به صورت الکتریکی فعّال گردد. دو سری پوش باتن Start, Stop در دو طرف آن برای قطع و وصل دستی در هر دو طرف قرار داده شده است. یک لیمیت سوییچ برای متوقف نمودن تسمه نقاله هنگام رسیدن بسته به انتها، قرار داده شده است.

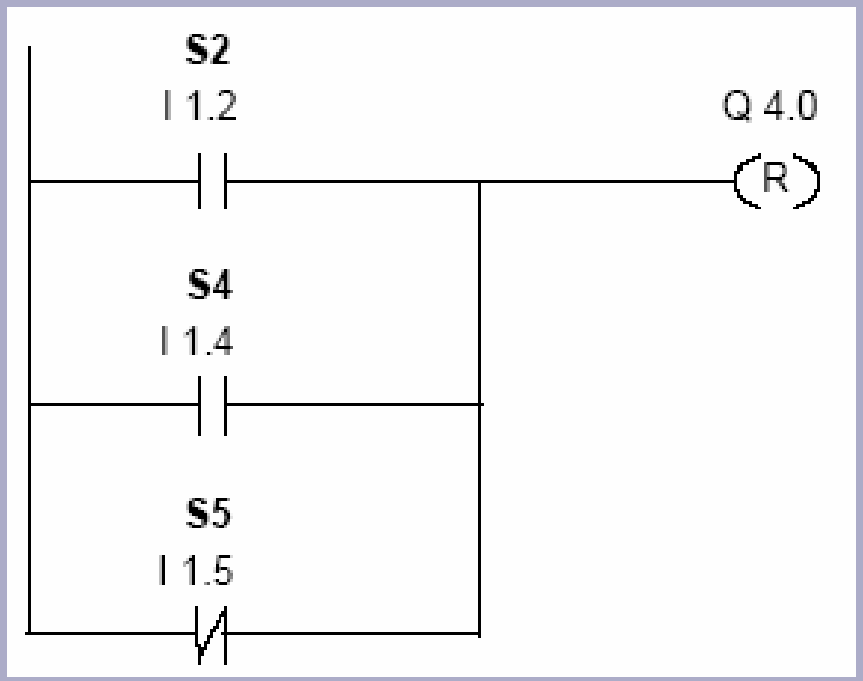
آدرس	سمبل	تجهیزات
I 1.1	S1	پوش باتن استارت طرف چپ
I 1.2	S2	پوش باتن استپ طرف چپ
I 1.3	S3	پوش باتن استارت طرف راست
I 1.4	S4	پوش باتن استپ طرف راست
I 1.5	S5	لیمیت سوییچ
Q 4.0	MOTOR_ON	راه انداز موتور

پوش باتن ها، راه اندازِ موتور و لیمیت سوییچ را مطابق جدول به ورودی –خروجی های PLC وصل می کنیم.

POWEREN.IR

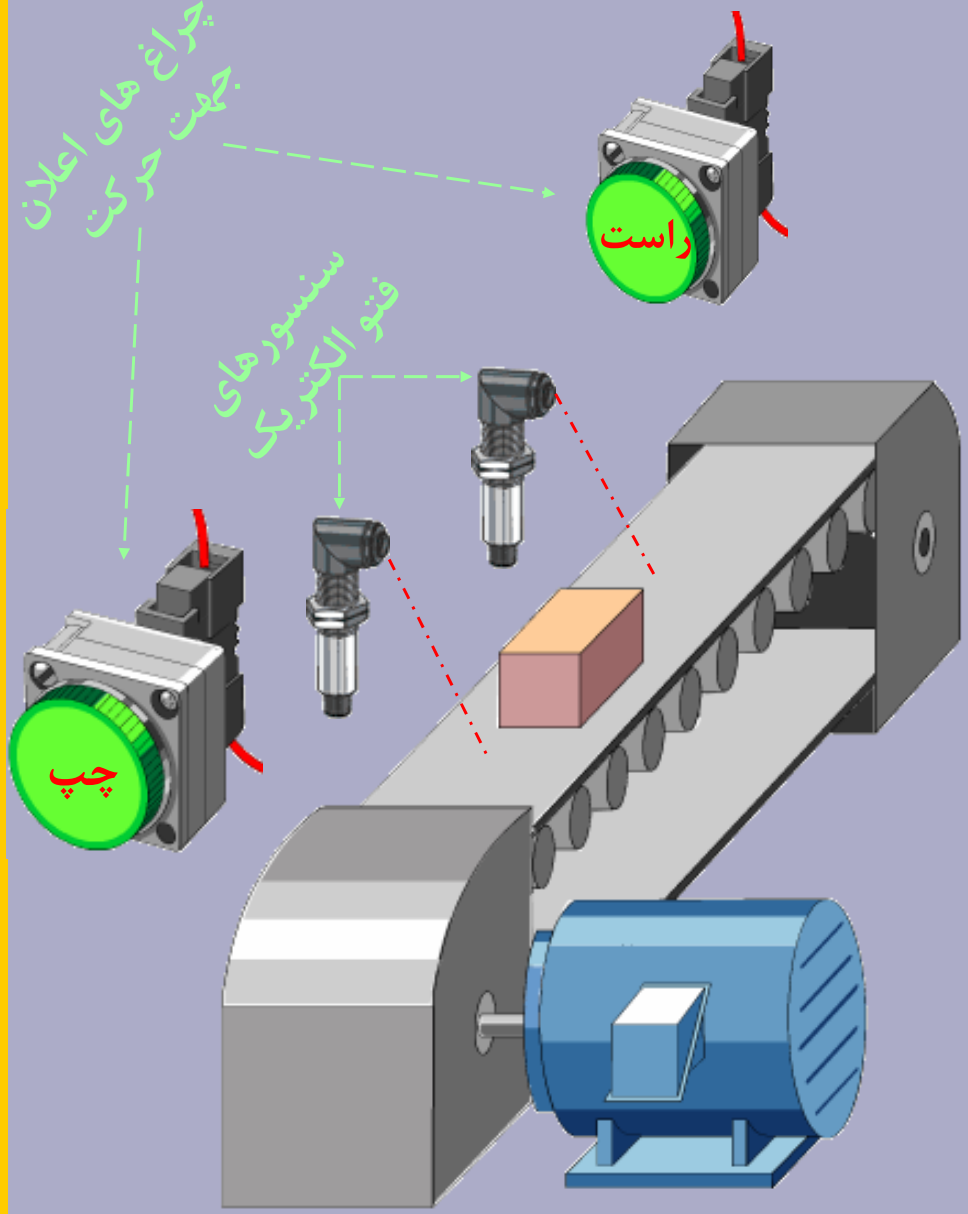


Network1:
هر کدام از استارت ها که زده شود، موتور روشن می شود.



Network2:
هر کدام از استپ ها که زده شود، یا وقتی که لیمیت سوییچ عمل کند، موتور متوقّف می شود.

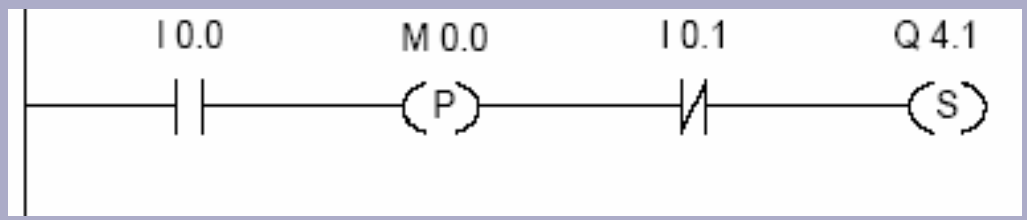
تعیین جهت حرکت بسته روی تسمه نقّاله



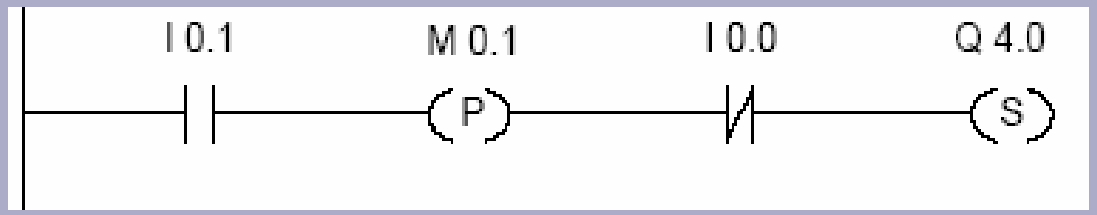
شکل مقابل تسمه نقّاله ای را نشان می دهد
که دو سنسور فتوالکتریک(برای تشخیص حضور شیء)
کنار آن نصب شده است.
باید برنامه ای نوشته شود که جهت حرکت بسته را
روی تسمه نقّاله نشان دهد.
(از کنتاکت های NO سنسورها استفاده می شود.)

آدرس	تجهیزات
I 0.0	سنسور فتوالکتریک سمت راست
I 0.1	سنسور فتوالکتریک سمتچپ
Q 4.0	چراغ نشان دهندهٔ حرکت به راست
Q 4.1	چراغ نشان دهندهٔ حرکت به چپ

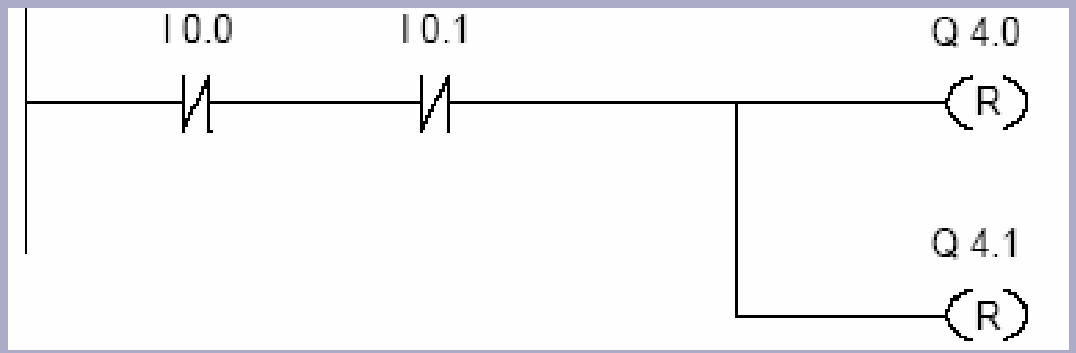
سنسور ها و سیگنال ها را
مطابق جدول، به ورودی –خروجی های PLC
وصل می کنیم.



Network1:
اگر سنسور سمت راست حضور بسته
را تشخیص دهد،در اینصورت ورودی
مطابقِ آن (I0.0) از 0 به 1تغییر می یابد.
در همین زمان سنسور سمت چپ تغییری
حس نمی کند.
در اینحالت بسته به سمت چپ در حال حرکت است.
و چراغ سیگنال "حرکت به چپ"
یا Q4.1روشن می شود



Network2:
اگر سنسور سمت چپ حضور بسته را تشخیص دهد،در اینصورت ورودی مطابقِ آن (I0.1) از 0 به 1تغییر می یابد.
در همین زمان سنسور سمت راست تغییری حس نمی کند.
در اینحالت بسته به سمت راست در حال حرکت است.
و چراغ سیگنال "حرکت به راست" یا Q4.0روشن می شود

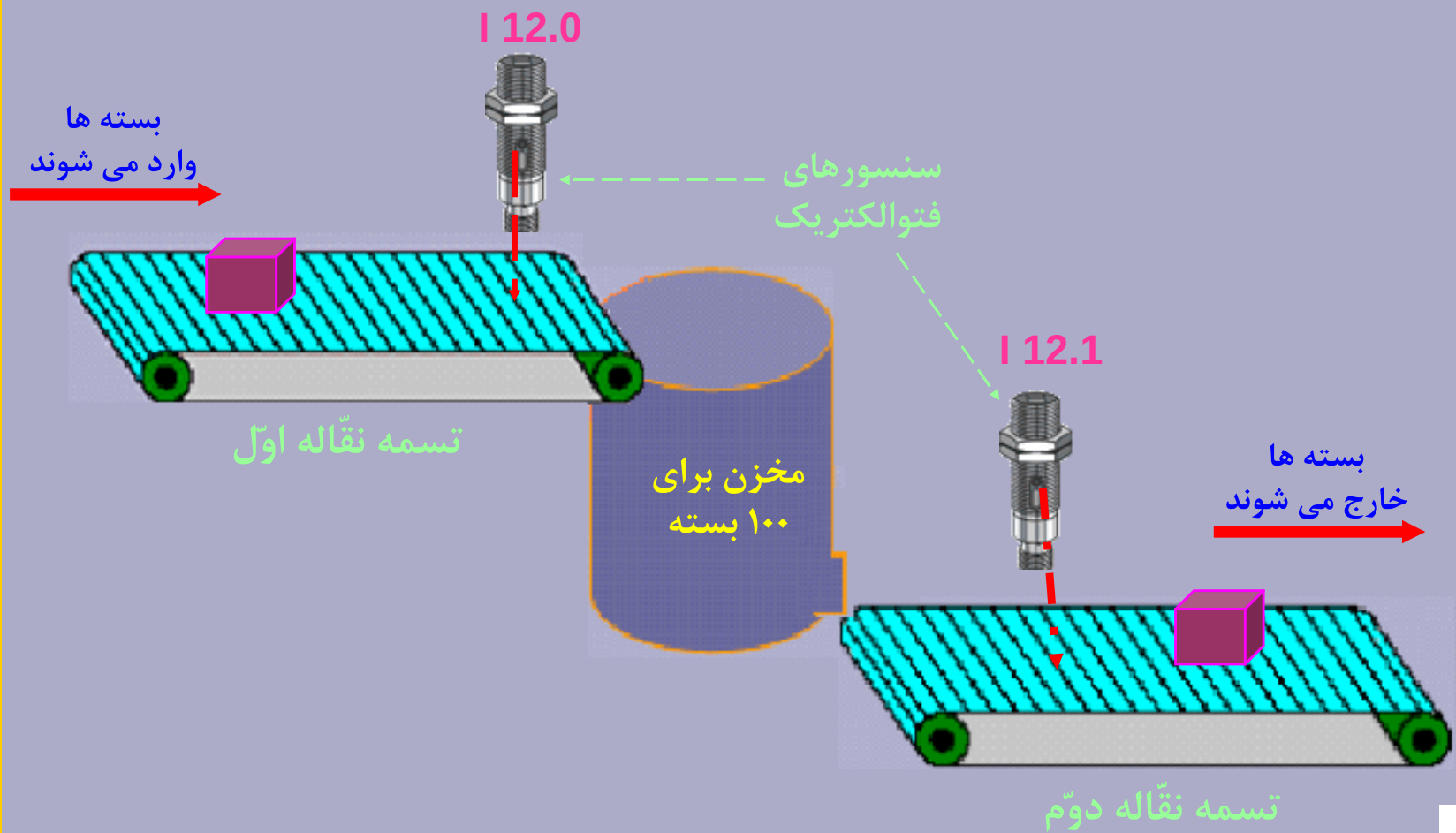


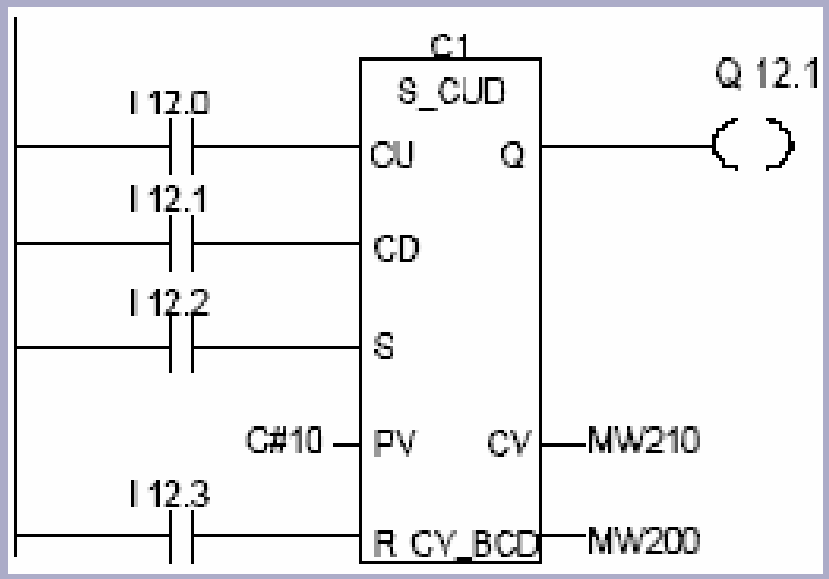
Network3:
اگر سنسور ها هیچ حرکتی را تشخیص ندهند، در این صورت یعنی بسته ای در بین سنسورها وجود نداشته و چراغ سیگنال ها خاموش می شوند.

مخزن ذخیره با شمارنده و مقایسه گر

دو تسمه نقّاله و یک مخزن در این مثال استفاده می شوند.
تسمه نقّالهٔ اوّل، بسته ها را به مخزن تحویل می دهد. یک سنسور فتوالکتریک در انتهای آن نزددیک مخزن تعداد بسته های وارد شده به مخزن را می شمارد.
تسمه نقّاله دوّم، برای تحویل بسته های ذخیره شده به مشتری و حمل ونقل استفاده می شود.
یک سنسور فتوالکتریک هم روی تسمه نقّالهٔ دوّم نزدیک مخزن برای شمارش بسته های خارج شده از مخزن استفاده می شود.
یک صفحهٔ نمایش با ۵ لامپ وضعیّت مخزن را مطابق شکل نشان می دهد.


مخزن
خالی است
(Q12.0)
مخزن
خالی نیست
(Q12.1)
مخزن
50% پر است
(Q15.2)
مخزن
90% پر است
(Q15.3)
مخزن
پر است
(Q15.4)


I 12.0
بسته ها
وارد می شوند
سنسورهای
فتوالکتریک
I 12.1
بسته ها
خارج می شوند
تسمه نقّاله اوّل
مخزن برای
۱۰۰ بسته
تسمه نقّاله دوّم

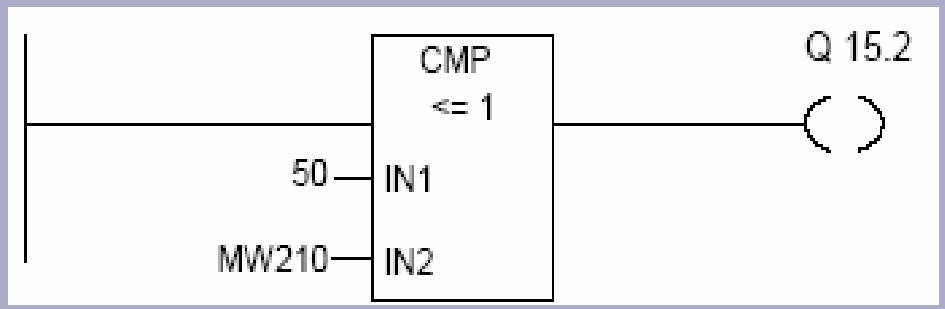


Network1:

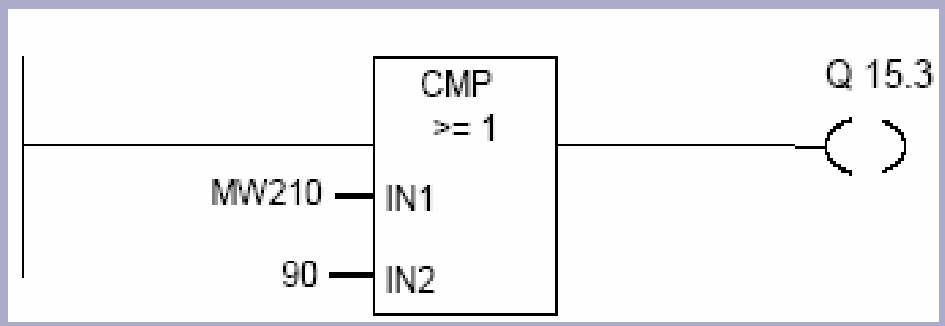
شمارندۀ **C1** وقتی سیگنال **0** به **1** از سنسور ورودی **(I12.0)** دریافت کند، یک شماره بالا می رود.
و وقتی این سیگنال را از سنسور خروجی **(I12.1)** دریافت کند، یک شماره کم می شود.
وروذی **I12.2** برای بارگذاری مقدار پیش فرض(مثلاً بسته های موجود مخزن پس از تعمیرات) استفاده می شود.
MW210 حاوی مقدار شمارندهمیباشد.
خروجی **Q12.1** نشان دهندۀ "مخزن خالی نیست" می باشد.

Network2:

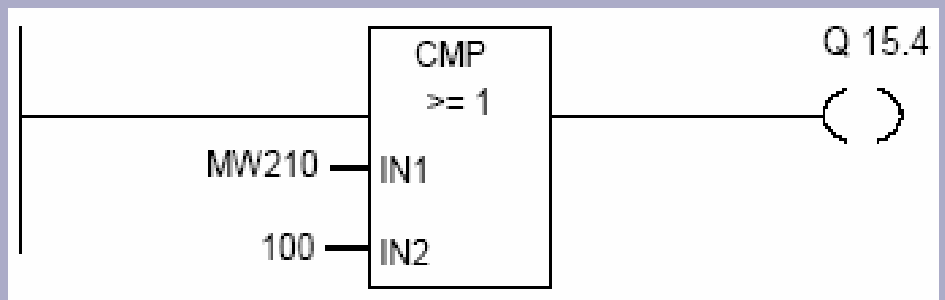
Q12.1 نشانگر "مخزن خالی است" می باشد



Network3:
اگر مقدار شمرده شده کوچکتر یا مساوی ۵۰ باشد ،بیانگر "مخزن 50%پر است" می باشد.



Network4:
اگر مقدار شمرده شده بزرگتر یا مساوی ۹۰ باشد ،بیانگر "مخزن 90%پر است" می باشد.



Network5:
اگر مقدار شمرده شده بزرگتر یا مساوی ۱۰۰ باشد ،بیانگر "مخزن پر است" می باشد.

به دست آوردن حاصل :

MW4 = ((IW0 + DBW3) x 15) / MW0