



نحوه استفاده و آشنایی با plc
خانواده زیمنس
و نرم افزار مربوطه تحت عنوان

Simatic s7 siemens
Industrial automation

تهیه و تدوین بر اساس داکيومنت ها و مراجع اصلي
کمپانی زیمنس

تهیه کننده و مولف

محمد یادگار توچاهی

کارشناس فنی درجه ۳

اداره کل تعمیرات مونتاژ یک

اداره پشتیبانی و مهندسی تعمیرات

پاییز ۱۳۸۴

فهرست :

۴	پیش گفتار مولف
۷	مقدمه
۱۰	آشنایی با محیط نرم افزار
۱۱	ساختن یک پروژه جدید در برنامه
۱۵	نحوه کانفیگوریشن و تنظیمات سخت افزار
۲۰	ساختن بلوک ها در برنامه
۲۲	انواع بلوک در s7
۲۵	فرم های نوشتن برنامه در s7
۲۵	کتابخانه s7
۲۶	BIT LOGIC
۳۲	FLIP FLOP
۴۱	مقایسه کننده ها (COMPARE)
۴۴	FunctionBlock Diagram (FBD)
۵۴	تبدیل کننده ها (CONVENTORS)
۶۴	شمارنده ها (COUNTERS)
۷۰	دستورات پرش (JUMP)

ادامه فهرست :

تایمرها (TIMERS)..... ۷۴

شیفترها (SHIFTERS) ۸۹

مقایسه FBD و LADD ۹۶

چکیده و تقدیر و تشکر..... ۹۸



پیش گفتار مولف :

همانگونه که تمامی همکاران عزیز مستحضر می باشید امروزه با پیشرفت علم و تکنولوژی کاربرد کامیوتر در صنعت نیز نمود پیدا کرده است بدین گونه که بجای تابلوهای فرمان و قدرت دستگاههای خط تولید که تماماً بصورت رله کنتاکتوری ساخته می شد و در نتیجه برای تعمیر و عیب یابی هر ایراد اولاً به تجربه و شناخت کافی از تابلو نیاز بود و ثانیاً بایستی مرحله به مرحله تمامی تابلو از طریق نقشه کنترل می گردید تا ایراد مشخص گردد اما امروزه از دستگاهی بنام plc استفاده می گردد و بوسیله plc علاوه بر اینکه می توان هر ایرادی را مونیتور نموده و بر روی صفحه op نمایش داد بدون نیاز به نقشه و براحتی می توان ایرادات دستگاه ها را در کوتاه ترین زمان ممکن مشخص و رفع عیب نمود .

از دیگر مزایای plc قابلیت اتصال آنها به یکدیگر و برقراری ارتباط شبکه می باشد که بدین وسیله می توان از طریق یک کامپیوتر مرکزی در اتاق کنترل تمامی تجهیزات و plc های داخل خط تولید را کنترل و مورد بررسی قرار داد که این موضوع باعث کاهش نیروی ماهر در خط تولید و افزایش دقت و راندمان تجهیزات می گردد .

در کارخانه معظم ایران خودرو از سال ۱۳۷۹ تا کنون تقریباً تمامی تجهیزات خطوط تولید سالن های مختلف از قبیل مونتاژ ، رنگ ، بدنه سازی ، پرس ، موتورسازی و ریخته گری از حالت قدیمی خارج شده و مجهز به سیستم اتوماسیون صنعتی و Plc گردیده است .
از این رو احتیاج به متخصصین plc در این مدت روز بروز افزایش یافته و در حال گسترش نیز می باشد .

از انجاییکه در داخل کشور این تکنولوژی و رشته فنی بصورت یک رشته تخصصی در دانشگاهها و آموزشگاههای کشور تدریس نمی گردد و یا بصورت بسیار محدود وجود دارد لذا بندرت شرکت ایران خودرو می تواند کارشناسان با تجربه را در این رشته تخصصی استخدام نماید

بنابراین بهترین گزینه آموزش افراد در داخل شرکت می باشد که در سالهای اخیر مرکز محترم آموزش ایران خودرو در این زمینه گامهای اساسی برداشته و اقدام به راه اندازی آزمایشگاه plc نموده است .

اگر دوستان و همکاران عزیز جهت خرید کتابهای مربوط به plc به روبروی دانشگاه تهران مراجعه نموده باشند در می یابند که اکثریت کتابها ی نوشته شده در این زمینه مربوط به تئوری و مفهوم Plc و شناخت قطعات داخل Plc می باشد و کتابی که در رابطه با استفاده از نرم افزار plc زیمنس سری خانواده s7 نوشته شده باشد بندرت یافت می گردد

از انجاییکه در کارخانه ایران خودرو استفاده از زیمنس سری s7 متداول می باشد از این رو نیاز به یک مرجع اصلی جهت استفاده از نرم افزار simatic s7 بعنوان راهنمای کارشناسان و تکنسین های محترم الکترونیک شرکت بیش از پیش احساس می گردد

در این جزوه فرض بر این می باشد که همکاران عزیز مفهوم plc و مدارات منطقی و علم لاجیک را که در کتابهای مختلف به تفسیر آموزش داده شده است دانسته و بجهت استفاده سریع از نرم افزار s7 از این مجموعه استفاده نمایند .

امید است که تلاش چندین ماهه این حقیر بعنوان یک مرجع مورد استفاده همکاران محترم قرار گرفته و مفید واقع گردد

در پایان از همکاران و صاحب نظران این رشته بابت نواقص موجود عذر خواهی نموده و استدعا دارم تا ضمن مطالعه این مجموعه ایرادات موجود را مشخص و نظرات خود را برای اینجانب ارسال نمایند تا بتوانم در اسرع وقت مشکلات را برطرف و مجموعه ایی نسبتاً کامل را در اختیار همکاران عزیز قرار بدهم .

در اینجا جا دارد از تمامی همکاران عزیز و خصوصاً روسای محترم اداره کل و اداره پشتیبانی تعمیرات مونتاژ که در تهیه این مجموعه بنده را مورد لطف و پشتیبانی خود قرار داده اند کمال سپاسگزاری را بنمایم .

یا تشکر

محمد یادگار

کارشناس فنی تعمیرات مونتاژیک

مقدمه :

• ساختار PLC

PLC از عبارت programable logic control به معنای کنترل کننده منطقی قابل برنامه ریزی گرفته شده است. PLC کنترل کننده ای نرم افزاری است که در قسمت ورودی اطلاعاتی را به صورت باینری یا آنالوگ دریافت و آنها را طبق برنامه ای که در حافظه اش ذخیره شده است پردازش می نماید و نتیجه عملیات را نیز از قسمت خروجی به صورت فرمانهایی به گیرنده ها و اجرا کننده های فرمان ارسال می کند. به عبارت دیگر PLC عبارت از یک کنترل کننده منطقی است که می تواند منطق کنترل را توسط برنامه برای آن تعریف نمود و در صورت نیاز بر راحتی آن را تغییر داد.

وظیفه PLC قبلاً بر عهده مدارات فرمان و رله های کنتاکتوری بود که امروز استفاده از آنها منسوخ شده است. از اشکالات عمده این رله ها این بود که با افزایش این رله ها حجم و وزن مدارات فرمان بسیار بزرگ شده و قیمت آنها نیز افزایش می یافت و نهایتاً عیب یابی اینگونه مدارات بسیار پیچیده و زمان بر می گردید.

برای رفع این معطل مدارات فرمان الکترونیکی ساخته شدند که آنها نیز به علت اینکه تک کار بودند و برای استفاده در چند مدار می بایستی تغییرات عمده در آنها ایجاد می شد کارایی کمی داشتند.

با استفاده از PLC تغییر در روند تولید یا عملکرد ماشین به راحتی صورت می گیرد زیرا دیگر لازم نیست سیم کشی ها و سخت افزار سیستم کنترل تغییر کند و تنها کافی است چند سطر برنامه نوشت و به PLC ارسال کرد تا کنترل مورد نظر تحقق یابد.

از طرف دیگر قدرت PLC در انجام عملیات منطقی و محاسباتی و مقایسه ای و نگهداری اطلاعات به مراتب بیشتر از تابلوهای فرمان معمولی است. PLC به طراحان این امکان را میدهد که آنچه را که در ذهن دارند در اسرع وقت بیازمایند.

هر کس با مدارات فرمان رله ای کار کرده باشد به خوبی می داند که پس از طراحی تابلو اگر نکته ای از کار افتاده باشد مشکلات بسیاری برای رفع آن پیش روست و زمان زیادی نیز صرف خواهد شد.

اکنون برای توجه بیشتر به تفاوت ها و مزایای PLC نسبت به مدارات کنتاکتوری موارد زیر را بر می شماریم:

- ۱- استفاده از PLC موجب کاهش حجم تابلوی فرمان می گردد.
- ۲- استفاده از PLC مخصوصاً در فرایندهای عظیم موجب صرفه جویی قابل توجهی در هزینه لوازم و قطعات میشود.
- ۳- PLC استهلاک مکانیکی ندارد بنا براین علاوه بر عمر بیشتر نیازی به تعمیرات و سرویس های دوره ای نخواهد داشت.
- ۴- PLC انرژی کمتری مصرف می کند.
- ۵- PLC ها بر خلاف مدارات رله کنتاکتوری نویز الکتریکی و صوتی ایجاد نمی کند.
- ۶- استفاده از یک PLC منحصر به یک پروسه و فرایند خاصی نیست و با تغییراتی در برنامه می توان به آسانی از آن برای کنترل پروسه های دیگر استفاده کرد.
- ۷- طراحی و اجرای مدارات کنترل و فرمان با استفاده از PLC بسیار سریع و آسان است.

۸- برای عیب یابی مدارات کنتاکتوری الگوریتم و روش خاصی نداریم اما در عیب یابی مدارات PLC بر راحتی با تغییرات در نرم افزار و *SIMULATION* کردن آن می توان عیب یابی کرد..

• کاربرد های PLC در صنایع مختلف :

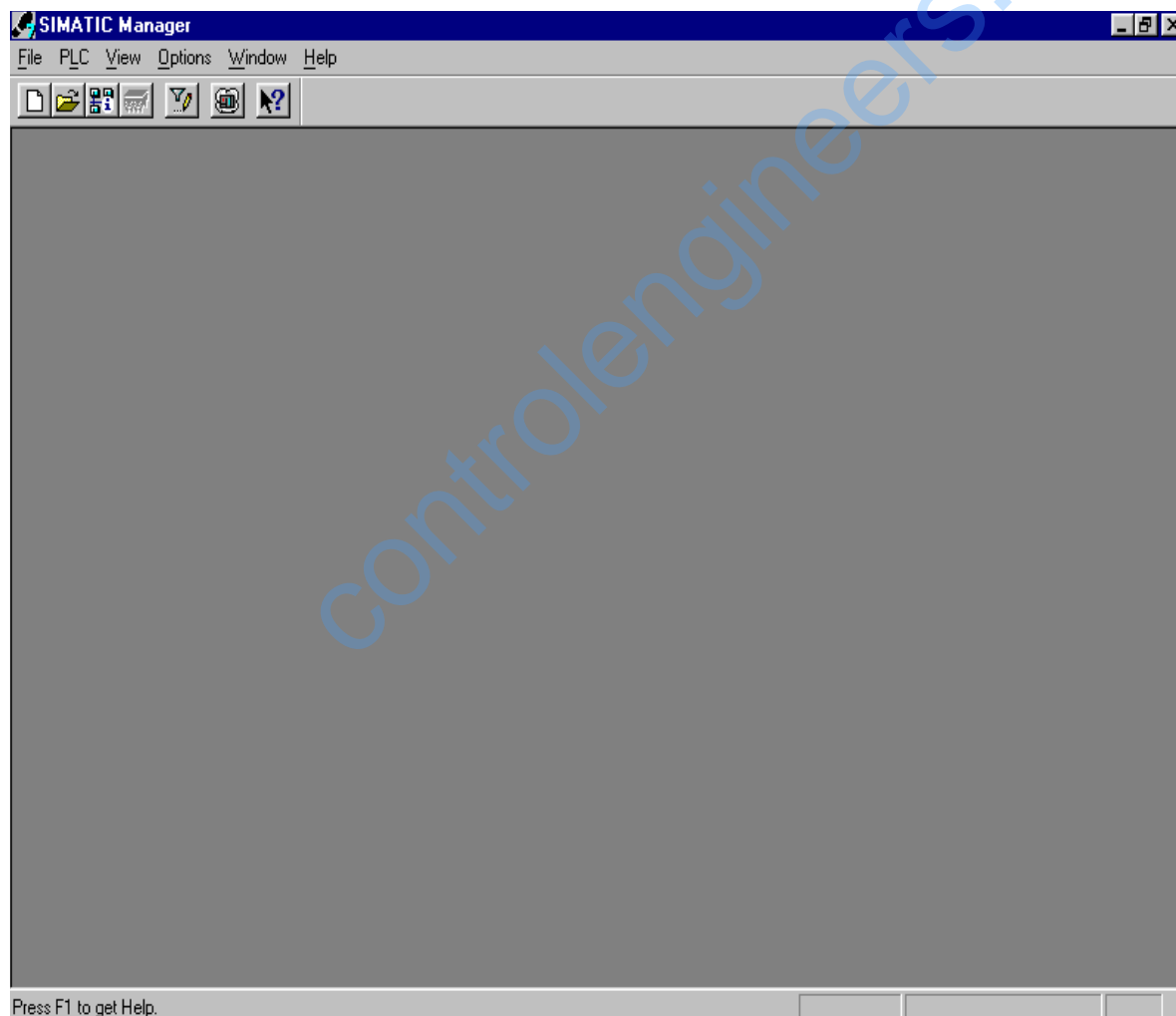
امروزه کاربرد های فراوانی از PLC در پروسه های مختلف صنعتی به چشم می خورد که خود نشانگر اهمیت فراوان PLC در صنعت است. از جمله این استفاده ها می توان به موارد زیر اشاره کرد:

- صنایع اتومبیل سازی شامل سوراخ کاری و پائش رنگ و حمل موتور LIFT و DROP .
- صنایع پلاستیک سازی شامل ذوب قالبگیری و دمش هوا
- صنایع سنگین شامل کوره های صنعتی کنترل دمای اتوماتیک
- صنایع شیمیایی شامل دستگاه های مخلوط شیمیایی
- خدمات ساختمانی شامل آسانسور تهویه هوا و...
- سیستم های حمل و نقل شامل جرثقیل ها سیستم کانوایر و...
- و..

آشنایی با محیط نرم افزار

همانند نرم افزار های دیگر با کلیک روی آیکون آن راه اندازی می شود . و همانطوریکه در شکل S7 نرم افزار ملاحظه میشود دارای یک باریکه شامل چند قسمت از جمله همانند سیستم عامل ویندوز می باشد :

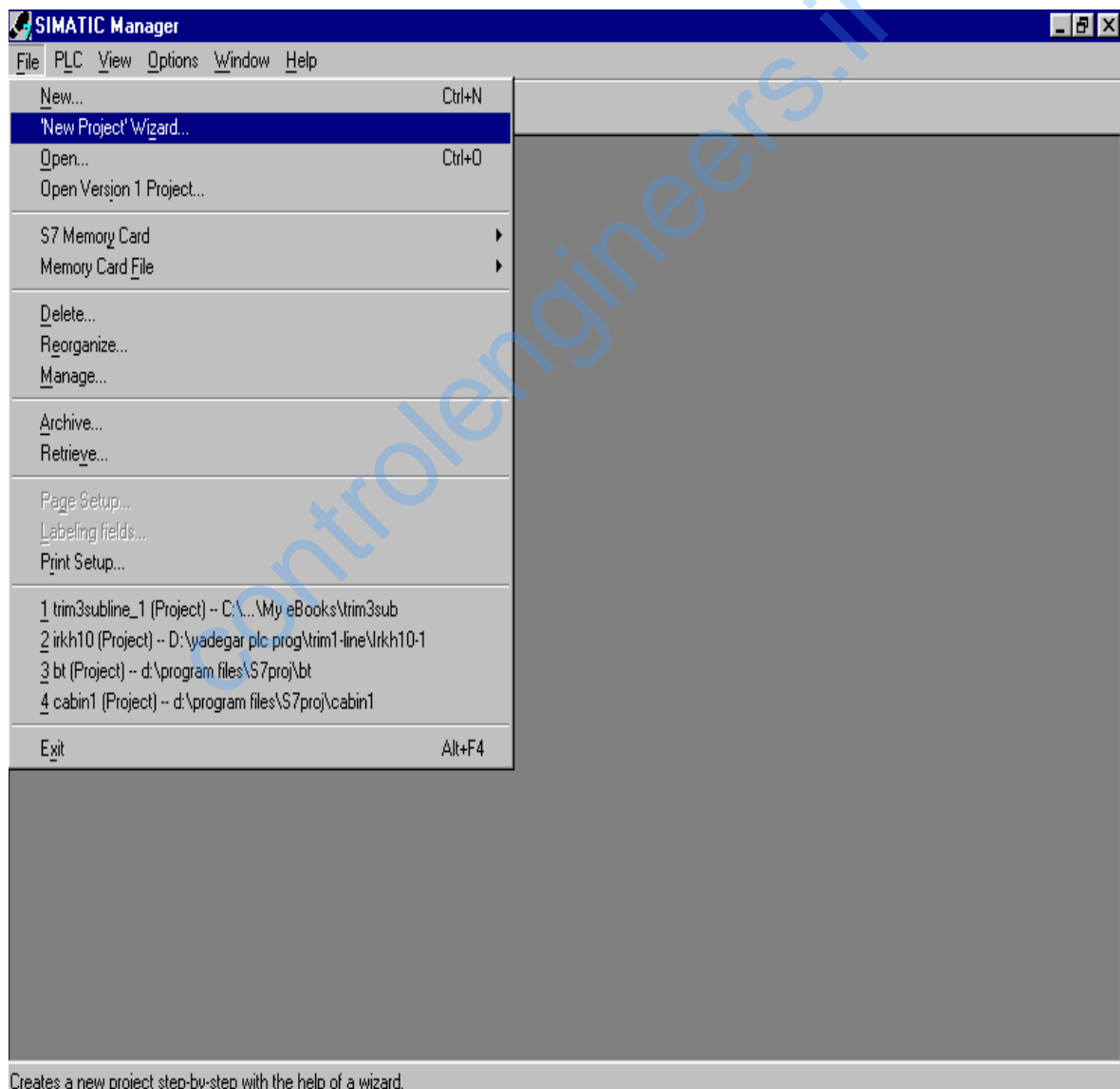
FILE/PLC/VIEW/OPTION/WINDOW/HELP



• ساختن یک پروژه جدید در برنامه s7

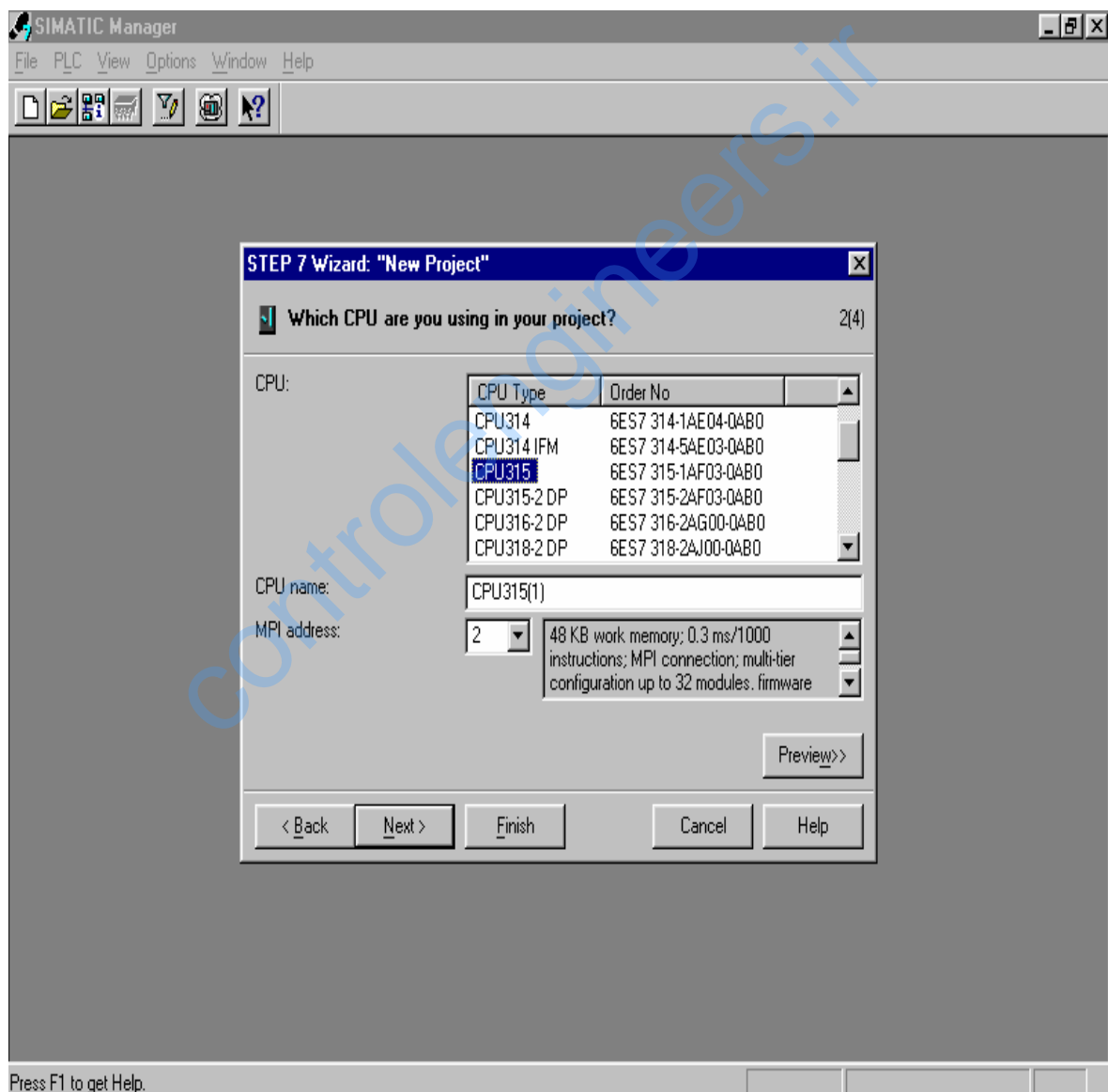
حال با انتخاب منوی FILE یک منوی کرکره ای ظاهر میشود این منو در حقیقت ابتدای کار با نرم افزار S7 می باشد .

در این مرحله با انتخاب گزینه NEW PROJECT WIZARD وارد صفحه جدیدی خواهیم شد. این منو در حقیقت جایی است که در آن میتوانیم قسمت های سخت افزاری مورد نظر را انتخاب کنیم .



• نحوه انتخاب سخت افزار پروژه

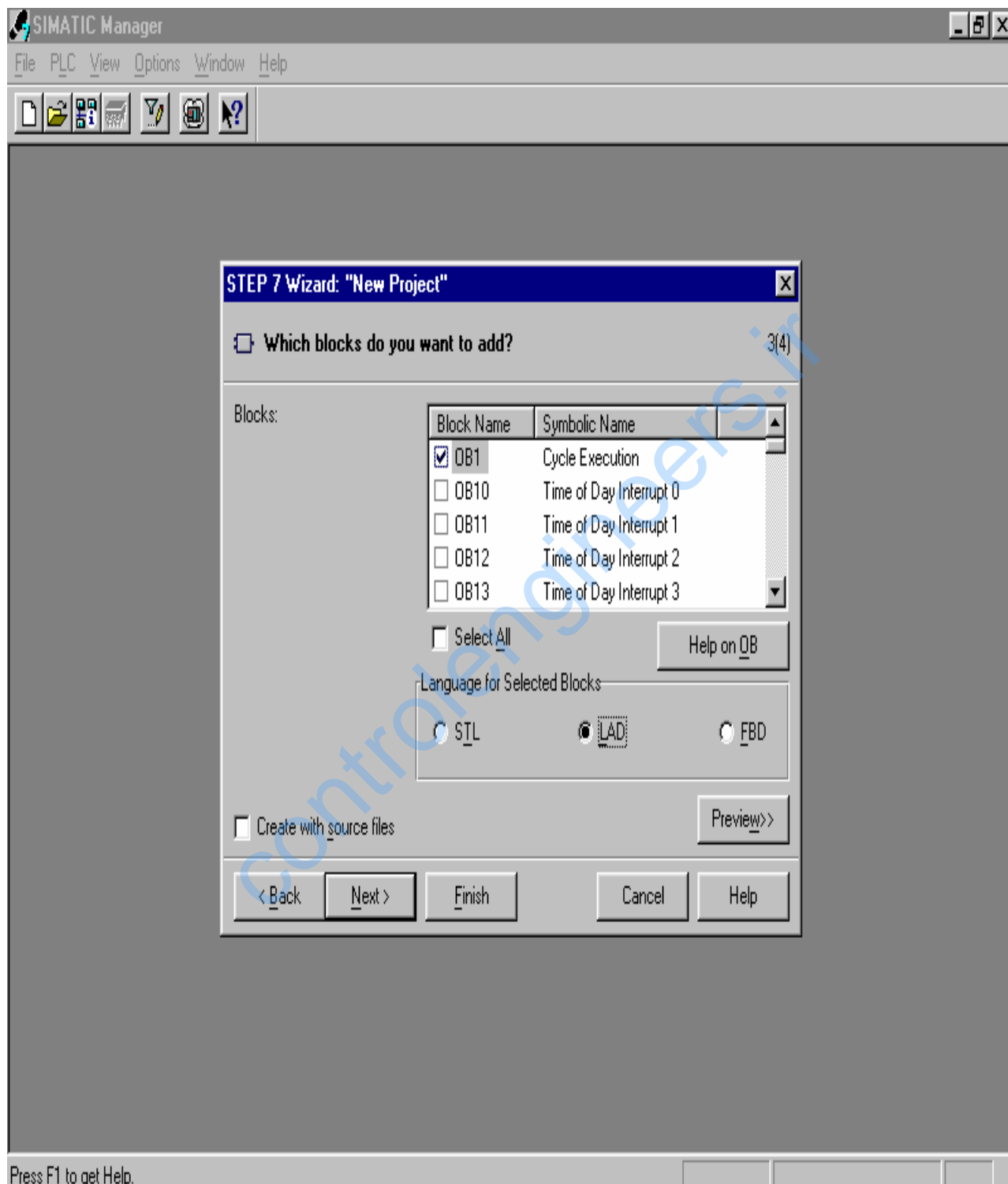
حال میتوانیم CPU مورد نظر را از CPU TYPE های پیشنهادی نرم افزار گزینش کنیم. همچنین در این صفحه به MPI ADDRESS بر میخوریم که مقدار آن را همیشه ۲ در نظر میگیریم. با زدن دکمه NEXT به صفحه بعد می رویم



در این صفحه ORGANIZATION BLOCK(OB) مورد نظر را گزینش میکنیم.که در این لیست OB های غیر از OB1 نیز تعریف گردیده است که طراح بنا بر احتیاجات خود میتواند از آنها نیز استفاده کند.همچنین در استیل های S7 آمده که کاربر میتواند یکی از سه گزینه LAD,FBD ,STL را بنا به استفاده خود انتخاب کند.

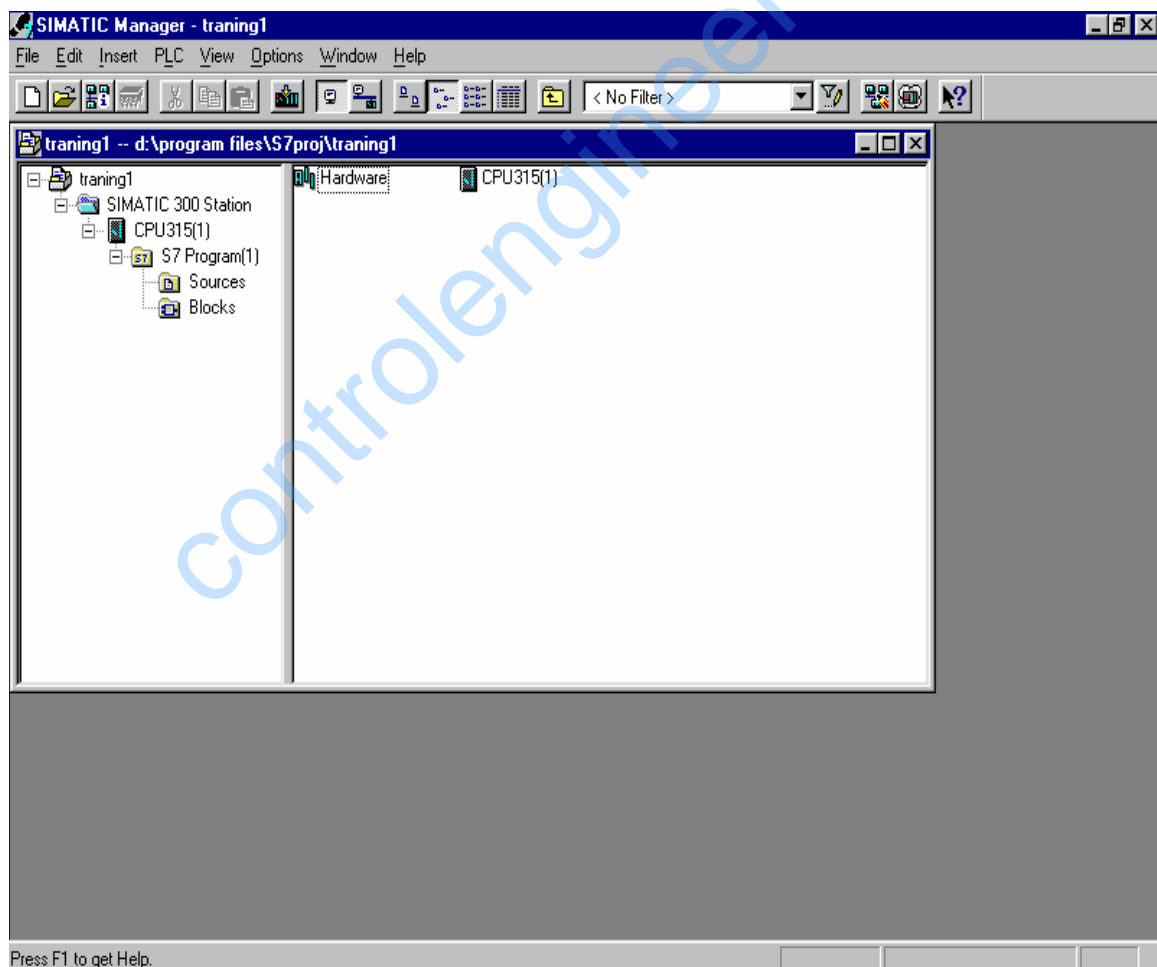
با زدن دکمه FINISH این مرحله به پایان رسیده و صفحه SIMANTIC MANAGE باز میشود.

در صفحه بعد نحوه عملکرد توضیح داده شده در فوق در داخل نرم افزار نمایش داده شده است .



• نحوه کانفیگوریشن و تنظیمات سخت افزار (HARDWARE)

حال در سمت راست صفحه بلوک های مورد استفاده در برنامه و در سمت چپ ساختار درختی آنچه در سمت راست انجام شده در پروژه 1 TRAINING دیده می شود. برای رفتن به قسمت SOURCE میتوانیم بقیه سخت افزار های مورد نیاز از جمله کارتهای ورودی و خروجی ، کارت های شبکه ، منبع تغیه ، پردازشگرها و... را از آنچه خود نرم افزار ارائه کرده است گزینش کنیم. برای این منظور باید به HARDWARE رفته و المانها را گزینش کنیم.



هم اینک صفحه HW CONFIG باز شده است که سمت چپ آن با سخت افزار های انتخابی پر میشود CPU که در ابتدا انتخاب کردیم در SLOT دوم نشسته و ما باید از سمت راست و از کتابخانه آن سخت افزار ها را انتخاب کنیم
ابتدا باید SIMANTIC300 یا SIMANTIC 400 را انتخاب کنیم.
و سپس المانها را از آنجا پیدا نماییم .

ما در این مثال محیط SIMANTIC300 را در نظر گرفته ایم. ابتدا PS (POWER SUPPLY) دلخواه را از منوی سمت راست DRAG و در اولین SLOT موجود در جدول DROP میکنیم . یاد آوری این نکته ضروری است که SLOT سوم همیشه خالیست که ما در اینجا PS 307 (منبع تغذیه ۵ آمپر) را بر داشتیم

HW Config - [SIMATIC 300 Station (Configuration) -- training1]

Station Edit Insert PLC View Options Window Help

Profile: Standard

(0) UR

1	PS 307
2	CPU315
3	
4	
5	
6	
7	
8	
9	
10	
11	

(0) UR

Slot	Module	Order number	Firmware	MPI addr...	I...	Q...	C...
1	PS 307 10A	6ES7 307-1KA00-0AA0					
2	CPU315(1)	6ES7 315-1AF03-0AB0		2			
3							
4							
5							
6							
7							
8							
9							
10							
11							

6ES7 307-1KA00-0AA0
Load supply voltage 120 / 230 VAC:24 VDC / 10 A

Press F1 to get Help.

اکنون نوبت انتخاب کارتهای رودی و خروجی است

AI/AO

مربوط به کارتهای آنالوگ است. ANALOG INPUT/OUTPUT. که ما در اینجا کارت AI4/A02 (۴ ورودی آنالوگ, و دو خروجی آنالوگ) را انتخاب کردیم.

DI/DO

مربوط به کارتهای دیجیتال است. DIGITAL INPUT/OUTPUT. که ما در اینجا کارت 16X12V/.5 A (۱۶ ورودی خروجی دیجیتال ۱۲ ولت نیم آمپر) را انتخاب کردیم.

همچنین در جدول پایینی مشخصات کامل و محل آدرس آنها در حافظه مشهود است.

ORDER NUMBER

شماره مشخص هر المان را نشان میدهد که هر شماره مختص یک المان است.

HW Config - [SIMATIC 300 Station (Configuration) -- training1]

Station Edit Insert PLC View Options Window Help

Profile: Standard

Slot	Module	Order number	Firmware	MPI addr...	I...	Q...	C...
1	PS 307 10A	6ES7 307-1KA00-0AA0					
2	CPU315(1)	6ES7 315-1AF03-0AB0		2			
3							
4	AI4/AO2	6ES7 334-0KE00-0AB0			256...	256...	
5	DI/DO 16x24V/0.5A	6ES7 323-1BL00-0AA0			4...5	4...5	
6							
7							
8							
9							
10							
11							

6ES7 323-1BL00-0AA0
Digital I/O module DI16 24 V + 16DO
24 V / 0.5 A

Press F1 to get Help.

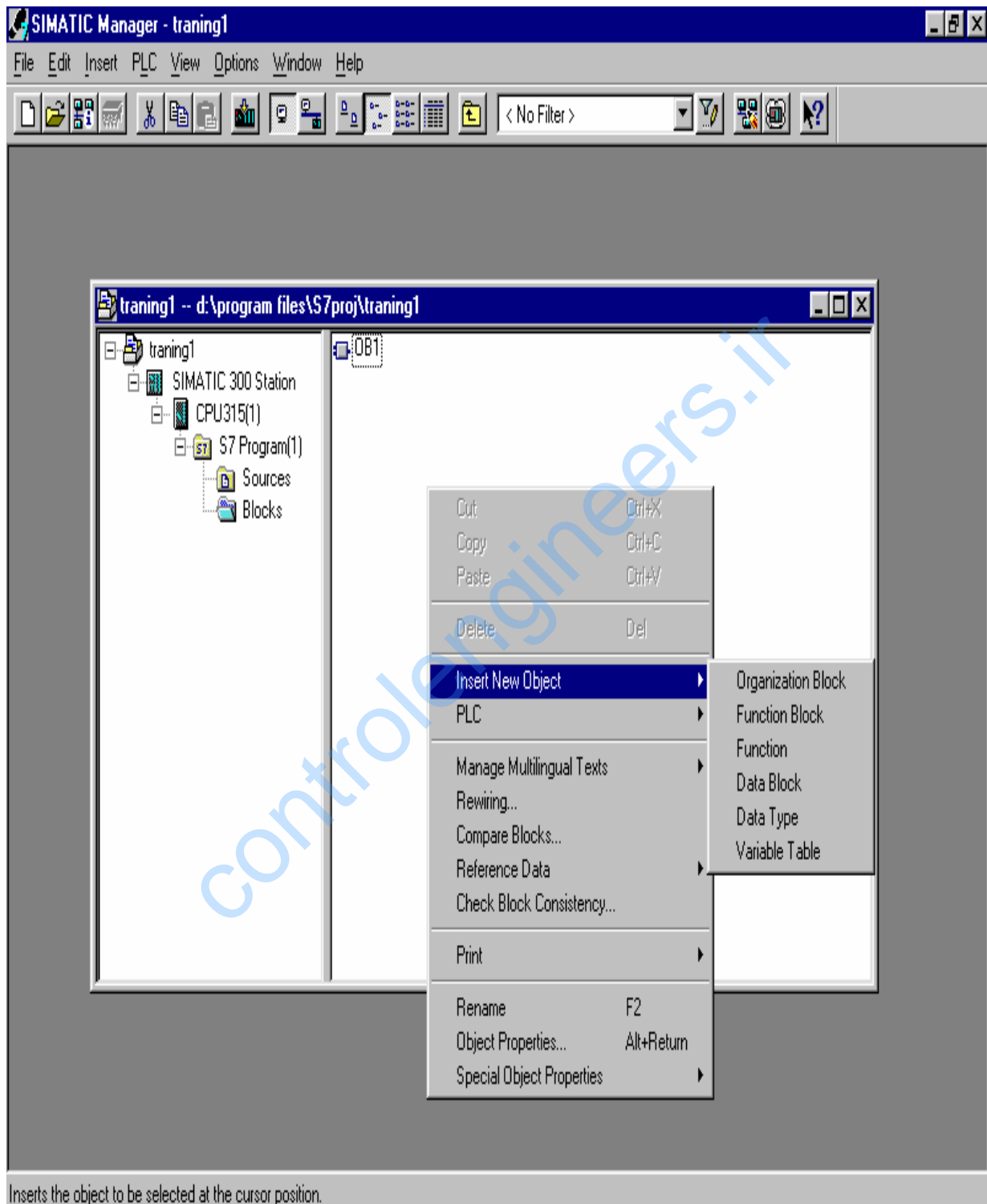
- ساختن بلوک ها در بر نامه

همانطور که گفته خواهد شد بلوک های مختلفی با وظایف مشخص در s7 وجود دارد. اکنون می بایست این بلوک ها در بر نامه فراخوانی شوند. برای این منظور ابتدا باید از ساختار درختی سمت چپ BLOCK را انتخاب کنیم. حال با کلیک چپ در سمت راست صفحه ساختار درختی ملاحظه می شود

باانتخاب INSERT NEW OBJECT میتوانیم هر یک از بلوک های :

FUNCTION BLOCK, DATABLOCK,
ORGANIZATION BLOCK, VARIABLE TABLE

را براساس احتیاج مطابق شکل زیر بسازیم.



انواع بلوک در S7 :

ORGANIZATION BLOC (OB) •

INTEGRATED SPESIAL OBs •

SYSTEM FUNCTION BLOCKS(SFBs) •

SYSTEM FUNCTIONS(SFC) •

FUNCTION BLOCK (FC) •

DATA BLOCK (DB) •

SYSTEM DATA BLOCK (SDB) •

: ORGANIZATION BLOCK –OB

در حقیقت OB توصیف کننده وظایف هر قسمت است و به عبارت دیگر واسطه بین استفاده کننده از سیستم و نرم افزار است. مهم ترین قسمت در بلوک ها OB است. نکته قابل توجه در OB ها انواع و تقسیم بندی آنها است.

: OB1

بلوک اصلی موجود است. و OB های دیگر به عنوان وقفه عمل کرده و وارد سیکل اجرای برنامه شده و برنامه را برای اجرای OB مورد نظر آماده میکند. انواع OB در برنامه و عملکرد آنها و شماره آنها در جداول پایین آمده است سمت راست شماره آنها و تفاوتهایشان در S5, S7 آمده است. نکته مهم در OB ها PRIORITY آنهاست بدین گونه که هر گاه دو OB همزمان فراخوانی شوند آن OB که دارای PRIORITY بیشتری باشد در اجرا اولویت دارد. مقادیر آنها در زیر آمده است.

OB10-OB23 که در آنها PRIORITY 2-23 است.

OB70-OB72 که در آنها PRIORITY 23-26 است.

...

Function	S5 Block	Replacement in S7
Cycle time triggering	OB31	SFC43 RE_TRIGR
Battery failure	OB34	OB81 (Error reaction can be programmed by user)
Access to condition code byte	OB110	STEP 7 instruction: L STW/T STW
Delete ACCU 1 – 4	OB111	STEP 7 instruction sequence: L 0; PUSH; PUSH; PUSH
Roll up ACCU	OB112	Function not identical: STEP 7 instruction: PUSH
Roll down ACCU	OB113	Function not identical STEP 7 instruction: POP
Disable all interrupts on/off	OB120	SFC41 DIS_AIRT SFC42 EN_AIRT
Disable cyclic (timed) interrupts individually on/off	OB121	SFC39 DIS_IRT SFC40 EN_IRT
Delay all interrupts on/off	OB122	SFC41 DIS_AIRT SFC42 EN_AIRT
Delay cyclic (timed) interrupts individually on/off	OB123	SFC39 DIS_IRT SFC40 EN_IRT
Set/read CPU time <i>(continued on next page)</i>	OB150	SFC0 SET_CLK SFC1 READ_CLK

که برنامه اصلی در ۱ OB ها و برنامه های که در طول اجرای برنامه دائماً فراخوانی می شوند در FC (فانکشن) و (فانکشن بلاک) FB ریخته می شوند .

فرمهای نوشتن برنامه در S7 :

(۱) فرم نردبانی (LADDER)

(۲) فرم نرم افزاری (STATEMENT LIST)

(۳) فرم بلوکی (FUNCTION BLOCK DATA)

فرم نردبانی (LAD):

هر گاه در بر نامه المانها به صورت مدارات پارالل رسم گردند یا به عبارت دیگر به صورت قطعات الکترونیکی موجود در کتابخانه قرار گیرند در اینصورت مدار به صورت LADDER بسته شده است.

حال باید در ابتدا با فرمان ها و المانهای موجود در کتابخانه S7 آشنا شویم تا بتوانیم از چگونگی کاربرد آنها در مدارات منطقی آگاه شویم:

بیت های لاجیک

---| |--- (Normally open contact) :

کنتاکتور در حالت عادی باز که برای هر کنتاکتور به صورت زیر آدرس دهی میکنیم.
I 0.1 که از سمت چپ میخوانیم عدد اول مربوط به آدرس بایت و عدد دوم مربوط به بیت اشغال شده از حافظه توسط این کنتاکتور است یعنی در آنجا آدرس بایت ۰ و بیت ۱ از حافظه اشغال شده است.

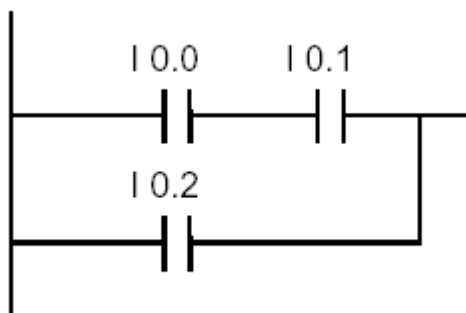
برای هر کدام از این المانها جدولی است که حاوی اطلاعاتی در مورد نوع DATA که توسط این آدرس داده میشود و MEMORY AREA آن است. که بعضی از این جداول را در پایین خواهیم آورد.

<address>

---| |---

Parameter	Data Type	Memory Area	Description
<address>	BOOL	I, Q, M, L, D, T, C	Checked bit

مثال :



: (Normally closed contact) ---|/|---

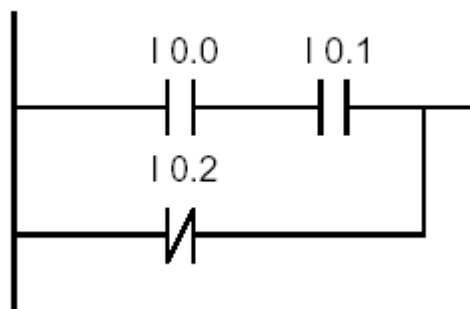
کنتاکتوری که در حالت عادی بسته است. آدرس دهی آن نیز مانند بالاست. تفاوت این دو کنتاکتور در اینست که کنتاکتور بالای در زمانیکه بیت RLO یک است فعال میشود اما در پایینی وقتی ۰ است یا به عبارت دیگر کنتاکتور پایینی با ۰ فعال و بالایی با آمدن ۱ فعال میشود.

<address>

---|/|---

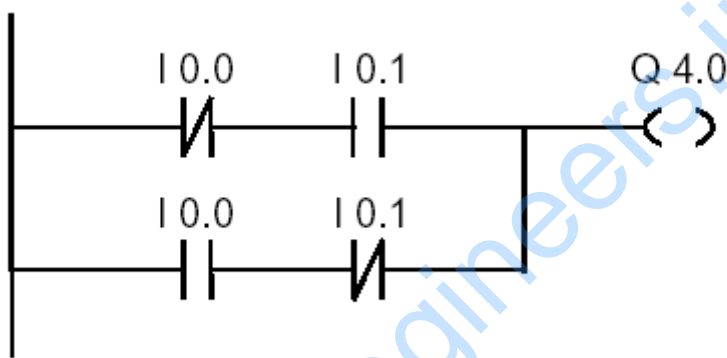
Parameter	Data Type	Memory Area	Description
<address>	BOOL	I, Q, M, L, D, T, C	Checked bit

مثال:



: (Xclusive OR) XOR

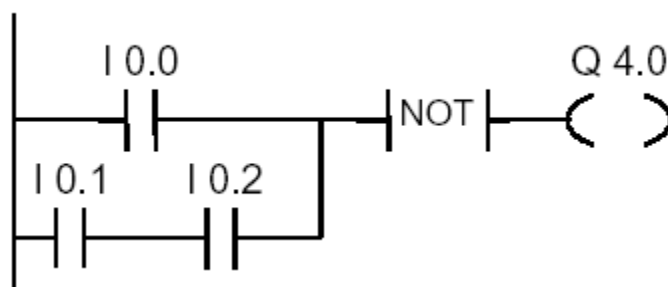
در مدار XOR جریان یا از شاخه بالایی و یا از پایینی باید برقرار شود.
این مدار شامل هر دو کنتاکتور باز و بسته است.



که نتیجه سیگنال در خروجی مورد نظر که در اینجا آدرس Q4.0 است ذخیره میشود.

بیت RLO را منفی میکند. ---|NOT|---

مثال:

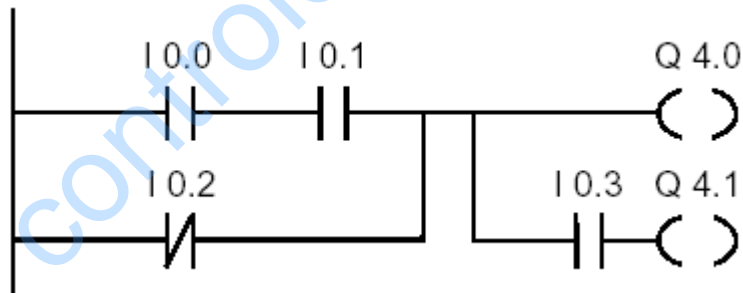


---()
(OUTPUT COIL)

به این صورت کار میکند که اگر بیت RLO یک شود آدرس این COIL نیز 1 میشود و اگر هم RLO صفر باشد بیت آدرس نیز صفر میشود.

مثال:

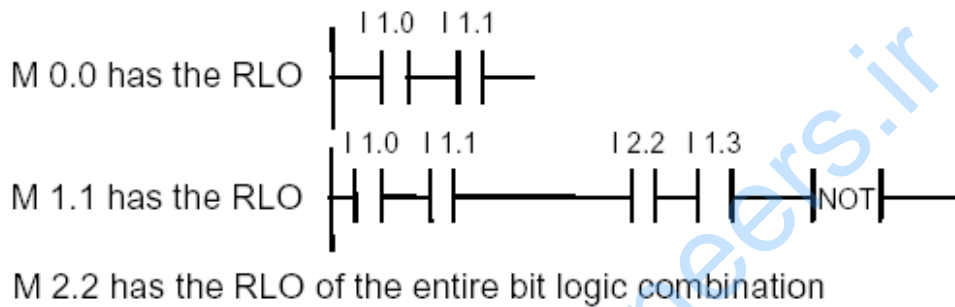
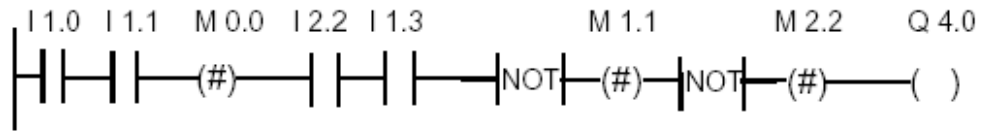
خروجی Q4.0 در یکی از حالات زیر 1 میشود:
سیگنال 1 باشد و AND I0.1, I0.0 رخ دهد.
سیگنال 0 باشد و جریان I0.2 را عبور دهد.
و یا خروجی Q4.1 زمانی 1 میشود که:
سیگنال 0 باشد و I0.2 صفر باشد و I0.3
سیگنال 1 باشد و AND I0.1, I0.0 رخ دهد و یک باشد.



---(#)---

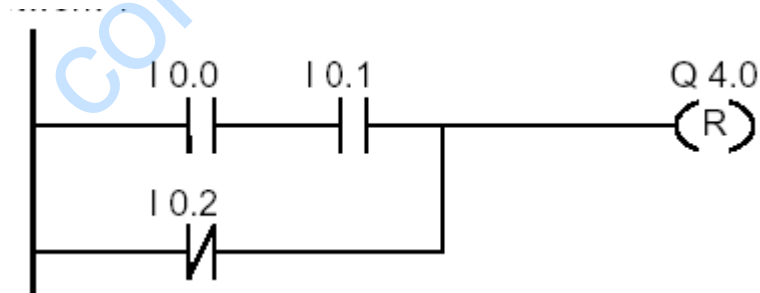
المانی که بیت RLO را در آدرس مورد نظر SAVE می کند .

مثال:



---(R)

بیت RLO را RESET میکند یعنی اگر ۱ باشد ۰ و اگر صفر باشد یک میکند.

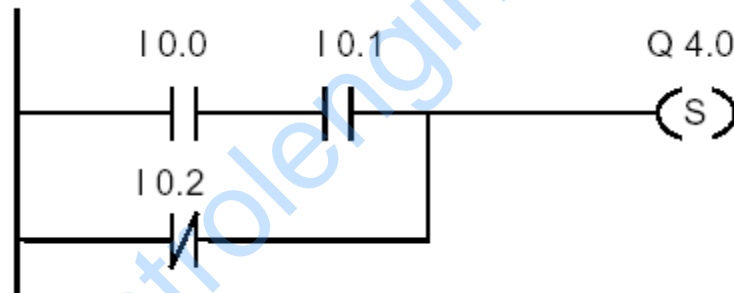


Parameter	Data Type	Memory Area	Description
<address>	BOOL	I, Q, M, L, D, T, C	Reset bit

---(S)

خروجی سیگنال هر چه باشد SET میکند یعنی ۱ می کند.

مثال:



Parameter	Data Type	Memory Area	Description
<address>	BOOL	I, Q, M, L, D	Set bit

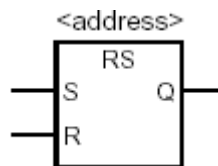
: FLIP FLOP

برای ثابت نگه داشتن ورودی در حالت صفر و یا یک به فلیپ فلاپ احتیاج داریم. اصطلاحاً بعنوان یک تیغه خود نگه دار عمل می کند .

: RESET SET FLIP FLOP(RS)

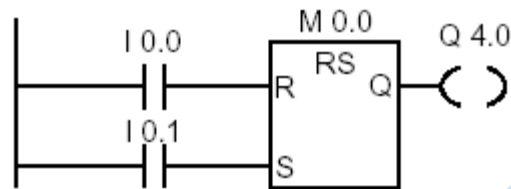
در این فلیپ فلاپ هنگامی تغییر رخ میدهد که بیت RLO ۱ باشد و زمانی که ۰ باشد هیچ تغییری روی مقادیر SET , RESET رخ نمی دهد.

در جدول پایین دیاگرام و مقادیر یک فلیپ فلاپ را مشاهده می کنیم



Parameter	Data Type	Memory Area	Description
<address>	BOOL	I, Q, M, L, D	Set or reset bit
S	BOOL	I, Q, M, L, D	Enabled reset instruction
R	BOOL	I, Q, M, L, D	Enabled reset instruction
Q	BOOL	I, Q, M, L, D	Signal state of <address>

مثال:



اگر I0.0 یک باشد و I0.1 صفر باشد بیت حافظه M0.0 SET می شود. و Q4.0 مقدار ۰ را به خود می گیرد.

نکته:

(در نرم افزار S7 برای اختصاص فضاي حافظه داخلي از اصطلاح M و آدرس دهی مربوطه استفاده می گردد و در نوع از Cpu ها تعداد آدرس هاي حافظه M که می توان استفاده نمود مشخص گردیده و از طریق گرفتن REFERENCE در نرم افزار می توانیم آنرا بفهمیم)

اگر I0.0 صفر باشد و I0.1 یک باشد بیت M0.0 RESET میشود و Q4.0 مقدار ۱ را به خود می گیرد.

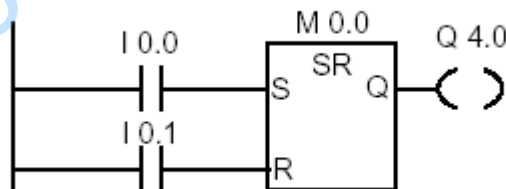
اگر هر ۲ صفر باشند هیچ تغییری رخ نمی دهد و اگر هر دو ۱ باشند مقدار SET بعنوان مقدم در نظر گرفته می شود.

: SET RESET FLIP FLOP(SR)



Parameter	Data Type	Memory Area	Description
<address>	BOOL	I, Q, M, L, D	Set or reset bit
S	BOOL	I, Q, M, L, D	Enable set instruction
R	BOOL	I, Q, M, L, D	Enable reset instruction
Q	BOOL	I, Q, M, L, D	Signal state of <address>

مثال:



اگر I 0.0 یک باشد و I 0.1 صفر باشد بیت M 0.0 SET می شود و Q 4.0 مقدار ۱ را به خود می گیرد.

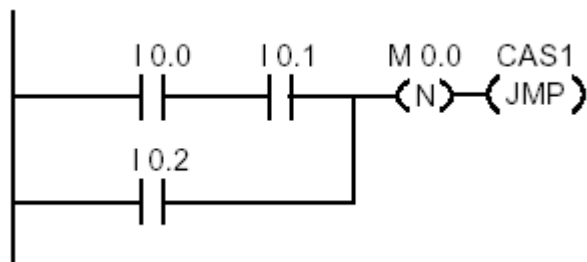
اگر I0.0 صفر باشد و I0.1 یک باشد بیت M0.0 RESET میشود و Q4.0 مقدار ۰ را به خود می‌گیرد.
اگر هر ۲ صفر باشند هیچ تغییری رخ نمی‌دهد و اگر هر دو ۱ باشند مقدار RESET ارجح قرار می‌گیرد.

---(N)---

زمانیکه آدرس از ۱ به ۰ تغییر می‌یابد بیت RLO را یک می‌کند.

Parameter	Data Type	Memory Area	Description
<address>	BOOL	I, Q, M, L, D	Edge memory bit, storing the previous signal state of RLO

مثال:



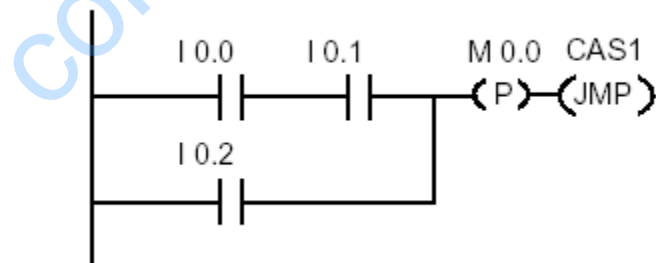
بیت حافظه M0.0 در RLO قدیم SAVE شده است هنگامی که سیگنال تغییر کند و سیگنال از ۱ به ۰ تغییر کند برنامه به آدرس CAS1 پرش می کند.

---(P)---

زمانیکه آدرس از ۰ به ۱ تغییر می یابد بیت RLO را یک می کند.

Parameter	Data Type	Memory Area	Description
<address>	BOOL	I, Q, M, L, D	Edge memory bit, storing the previous signal state of RLO

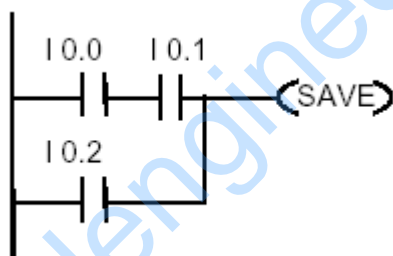
مثال:



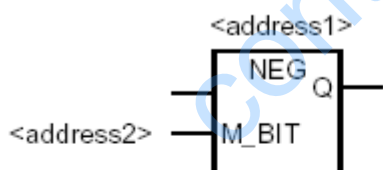
بیت M0.0 در RLO قدیم SAVE شده است هنگامی که سیگنال از ۰ به ۱ تغییر کند برنامه به آدرس CAS1 پرش می کند.

---(SAVE)

اطلاعات RLO را در BR MEMORY ذخیره می کند.
مثال:

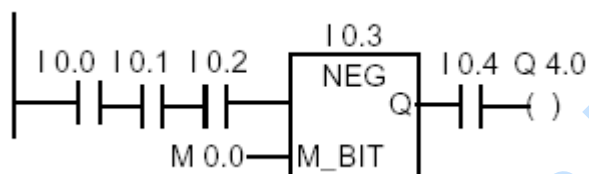


: NEG (NEGATIV EDG DETECTION)



Parameter	Data Type	Memory Area	Description
<address1>	BOOL	I, Q, M, L, D	Scanned signal
<address2>	BOOL	I, Q, M, L, D	M_BIT edge memory bit, storing the previous signal state of <address1>
Q	BOOL	I, Q, M, L, D	One shot output

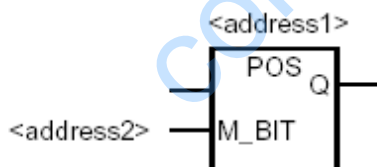
محتویات RLO قدیم و جدید را مقایسه میکند و در صورتیکه RLO جدید ۱ و RLO قدیمی ۰ باشد، بیت RLO را ۱ میکند.
مثال:



در I0.3 لبه پایین رونده داریم.
I0.0 و I0.1 و I0.2 سیگنال های ۱ می باشند
با لبه پایین رونده در I0.4 سیگنال ۱ گردیده و خروجی Q4.0 یک می گردد.

---(P)---

(POSITIVE EDGE DETECTION)

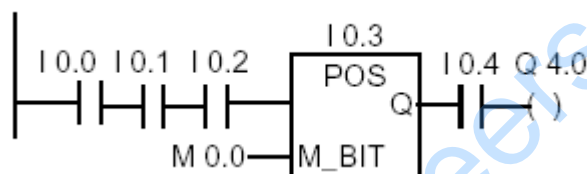


Parameter	Data Type	Memory Area	Description
<address1>	BOOL	I, Q, M, L, D	Scanned signal
<address2>	BOOL	I, Q, M, L, D	M_BIT edge memory bit, storing the previous signal state of <address1>
Q	BOOL	I, Q, M, L, D	One shot output

محتویات RLO قدیم و جدید را مقایسه میکند و در صورتیکه RLO جدید 1 و RLO قدیمی 0 باشد، بیت RLO را 0 میکند.

با لبه بالا رونده عمل می نماید و حساس به سطح سیگنال نمی باشد و زمانی که سیگنالی از حال صفر به حالت یک تغییر حالت یابد با اولین لبه بالارونده این بلوک عمل می نماید

مثال:



در I0.3 لبه بالا رونده داریم.

در صورتیکه I0.0 و I0.1 و I0.2 به سیگنال 1 تغییر حالت بدهند .
در این حالت در I0.4 و خروجی Q4.0 سیگنال یک می گردد .

نکته :

در شکل زیر در منوی سمت چپ که کتابخانه S7 می باشد تمامی دستورات فوق بصورت گرافیکی نمایش داده شده است که می توان بر اکتی بوسیله موس و با روش DRAG AND DROP آنها را به داخل برنامه و در هر NETWORK دلخواه که با باز کردن یک FC بعنوان مثال بصورت اتوماتیک ساخته می شود قرار داد .

LAD/STL/FBD - [OB1 -- traning1\SIMATIC 300 Station\CPU315(1)]

File Edit Insert PLC Debug View Options Window Help

Address	Declaration	Name	Type	Initial v	Comment
0.0	temp	OB1_EV_CLASS	BYTE		Bits 0-3 = 1

OB1 : "Main Program Sweep (Cycle)"

Comment:

Network 1: Title:

Comment:

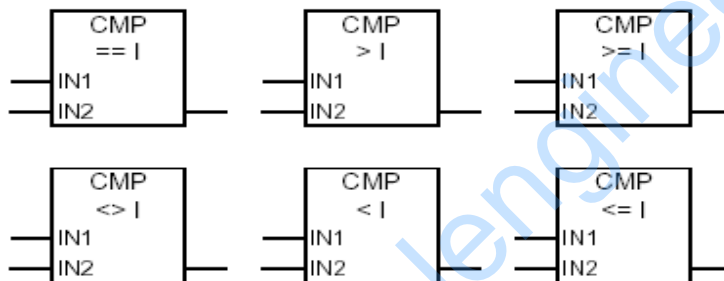
- |-|-
- |-
- |NOT|-
- ()
- (#)-
- (R)
- (S)
- RS
- SR
- (N)-
- (P)-
- (SAVE)
- NEG
- POS
- Comparator
- Converter
- Counter
- DB call
- Jumps
- Integer fct.
- Floating-point fct.
- Move
- Program control
- Shift/Rotate
- Status bits
- Timers
- Word logic
- FB blocks
- Normally Open Contact

Press F1 to get Help. offline Abs Nw 1 Insert

مقایسه کننده ها (COMPARATOR) :

(CMPI) COMPARE INTEGER

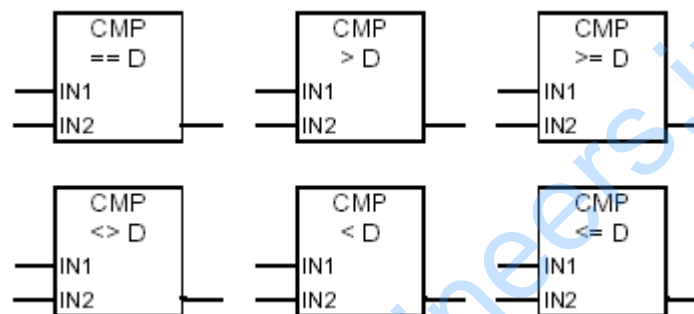
مقایسه کننده فوق برای مقایسه دو عدد INTEGER (اعداد صحیح بدون اعشار) استفاده می گردد که انواع مختلف کوچکتر مساوی ، بزرگتر مساوی و مساوی را می توان انتخاب نمود



Parameter	Data Type	Memory Area	Description
box input	BOOL	I, Q, M, L, D	Result of the previous logic operation
box output	BOOL	I, Q, M, L, D	Result of the comparison, is only processed further if the RLO at the box input = 1
IN1	INT	I, Q, M, L, D or constant	First value to compare
IN2	INT	I, Q, M, L, D or constant	Second value to compare

:(CMP D) COMPARE DOUBLE INTEGER

برای مقایسه اعداد INTEGER که بزرگ بوده و بیش از دو WORD برای نخبیره آنها در حافظه مورد نیاز می باشد



Parameter	Data Type	Memory Area	Description
box input	BOOL	I, Q, M, L, D	Result of the previous logic operation
box output	BOOL	I, Q, M, L, D	Result of the comparison, is only processed further if the RLO at the box input = 1
IN1	DINT	I, Q, M, L, D or constant	First value to compare
IN2	DINT	I, Q, M, L, D or constant	Second value to compare

• یادآوری مدارات منطقی

$0 \text{ AND } 1 = \text{BIT}$

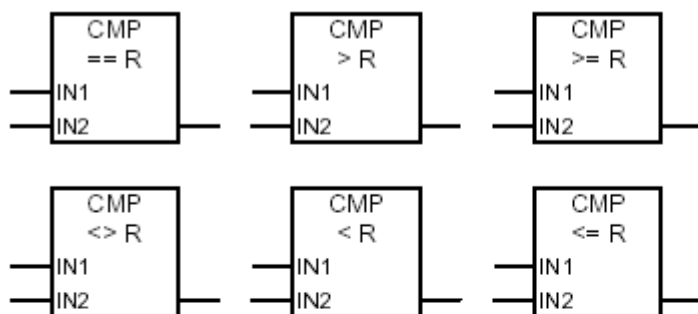
$8\text{BIT} = \text{BYTE}$

$2 \text{ BYTE} = \text{WORD}$

$4\text{BYTE OR } 2 \text{ WORD} = \text{DOUBLE WORD}$

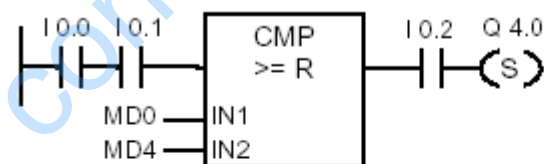
COMPARE REAL(CMP R)

برای مقایسه اعداد اعشاری استفاده می‌گردد



Parameter	Data Type	Memory Area	Description
box input	BOOL	I, Q, M, L, D	Result of the previous logic operation
box output	BOOL	I, Q, M, L, D	Result of the comparison, is only processed further if the RLO at the box input = 1
IN1	REAL	I, Q, M, L, D or constant	First value to compare
IN2	REAL	I, Q, M, L, D or constant	Second value to compare

مثال :



Output Q4.0 is set if the following conditions exist:

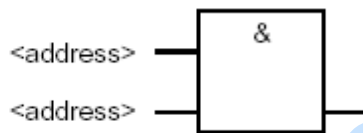
- There is a signal state of "1" at inputs I0.0 and at I0.1
- And MD0 >= MD4
- And there is a signal state of "1" at input I0.2

از آنجایی که در برنامه نویسی با s7 بعلت نزدیک بودن روش به مدارات منطقی معمولاً از FBD (FUNCTION BLOCK DIAGRAM) استفاده می کنیم و کاربرد المانها در تمام فرمهای یکی است. لذا ادامه LIBRARY را در FBD ادامه می دهیم.

: FUNCTION BLOCK DIAGRAM (FBD)

: AND(&)

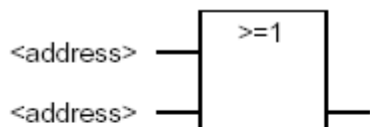
آدرس ها را در محل نشان داده شده می نویسیم و از بلوک AND استفاده می کنیم.



Parameter	Data Type	Memory Area	Description
<address>	BOOL	I, Q, M, T, C, D, L	The address indicates the bit whose signal state will be checked.

: OR(>=1)

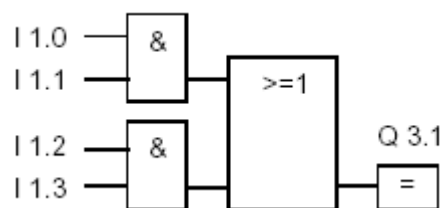
آدرس ها را در محل نشان داده شده می نویسیم و از بلوک AND استفاده می کنیم.



Parameter	Data Type	Memory Area	Description
<address>	BOOL	I, Q, M, T, C, D, L	The address indicates the bit whose signal state will be checked.

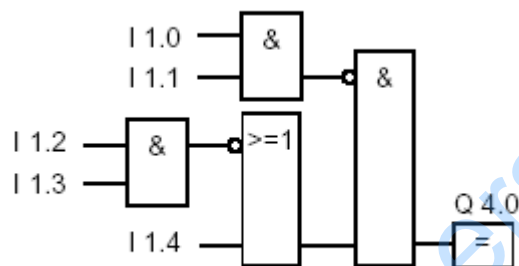
ترکیب AND ,OR :

I1.0, I1.1 با هم AND میشوند و I1.2, I1.3 نیز با هم AND میشوند نتایج با هم OR و در Q3.1 ریخته می شود.



: NEGATE BINARY INPUT (\neg)

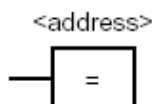
هر گاه انجام نشدن یک رخداد برای ما مطرح باشد در این صورت از این فرم استفاده می کنیم. یعنی اینکه بخواهیم معکوس یک ورودی (NOT) را به سیستم بدهیم.



مفهوم بالا اینکه (AND I1.2 , I2.3) یک نشود با I1.4 OR شوند . NOT جواب AND بلوک اول (بالایی) و بلوک دوم AND (پایینی) میشوند و جواب در Q4.0 ریخته میشود.

: ADDRESSING

آدرس دهی در s7



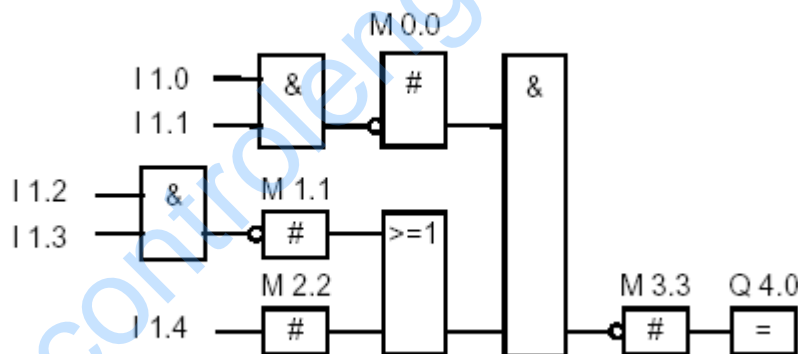
Parameter	Data Type	Memory Area	Description
<address>	BOOL	I, Q, M, D, L	The address specifies the bit to which the signal state of the string of logic operations is assigned.

در s7 مقداری به ADDRESS یک بلوک تخصیص داده می شود.
که در بالا این آدرس Q4.0 است.

: MIDLINE OUTPUT (#)

هدف آن SAVE کردن بیت RLO است.

مثال:



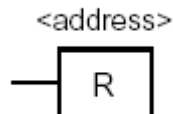
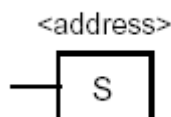
M0.0 منفی (NEGATIVE) RLO بلوک اول را SAVE میکند.
M1.1 منفی (NEGATIVE) RLO بلوک دوم را SAVE میکند.
M2.2 : RLO بیت I1.4 را SAVE میکند.
M3.3 منفی (NEGATIVE) RLO بلوک آخر را SAVE میکند.

: RESET (R)

همانند آنچه در روش LADD گفته شد.

: SET(S)

همانند آنچه در روش LADD گفته شد.

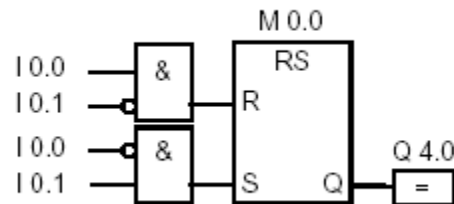


فلیپ فلاپ:

: RESET_SET FIP FLOP

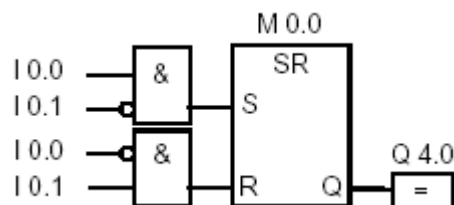
توضیحات همانند گذشته که در روش ladder توضیح داده شده می باشد.

مثال:



اگر I0.0 یک و I0.1 صفر باشد بیت M0.0 RESET و Q4.0 صفر است.
 اگر I0.0 یک و I0.1 صفر باشد بیت M0.0 SET و Q4.0 یک است.
 اگر هر دو صفر باشند اتفاقی نمی افتد.
 اگر هر دو یک باشند SET رخ می دهد. Q4.0 یک میشود.

SET_RESET FLIP FLOP :

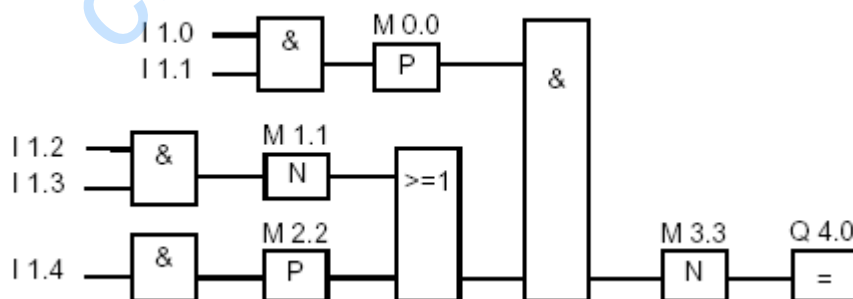


اگر I0.0 یک و I0.1 صفر باشد بیت M0.0 SET و Q4.0 یک است.
 اگر I0.0 یک و I0.1 صفر باشد بیت M0.0 RESET و Q4.0 صفر است.

اگر هر دو صفر باشند اتفاقی نمی افتد.
اگر هر دو یک باشند RESET رخ می دهد. Q4.0 یک میشود.



آشکار سازهای لبه بالارونده و پایین رونده بوده که توضیحات در روش قبلی ارسال گردیده است
مثال :

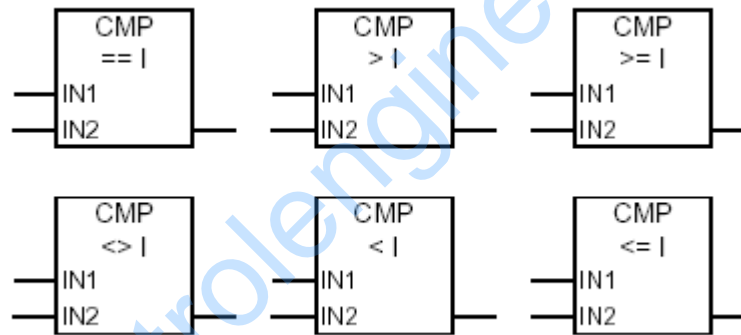


در N اگر از ۱ به ۰ برویم RLO را ۱ میکند.
در P اگر از ۰ به ۱ برویم RLO را ۰ میکند.

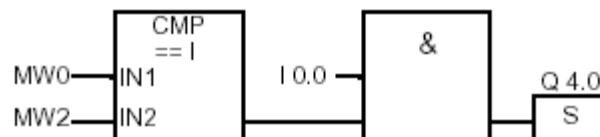
مقایسه کننده ها در FBD :

:(COMPARE INTEGER) CMPI

توضیحات همانند قبل .



مثال:

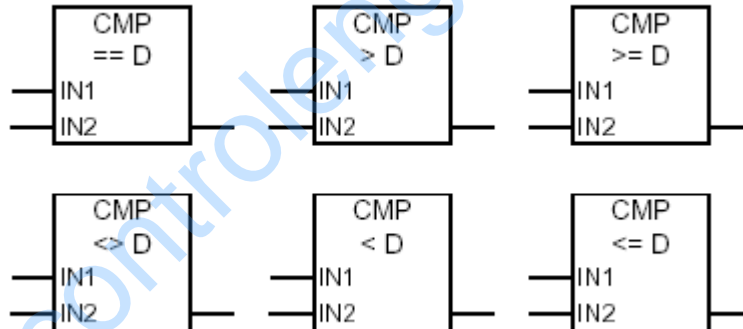


SET Q4.0 است زمانی که MWO, MW1 برابر باشند و با I0.0 AND شوند.

نکته :

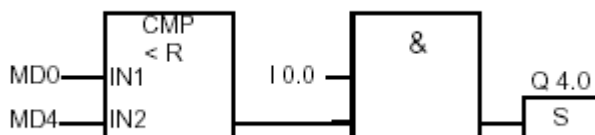
MW (MEMORY WORD) به حافظه های بالاتر از یک BIT حتی یک BYTE اطلاق می گردد که دارای دو بایت می باشد و برای ذخیره مقادیر بزرگ و یا اعداد اعشاری کاربرد دارد

: CMPD



توضیحات همانند قبل.

: CMPR



MD0 از MD4 کوچکتر باشد I0.0 یک باشد

در این صورت Q4.0 SET است.

نکته :

MD (MEMORY DOUBLE) به حافظه دارای 2WORD اطلاق می گردد

تبدیل کننده ها:

CONVERSION INSTRUCTIONS

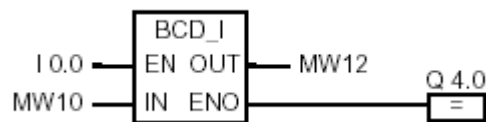
: BCD_I

مبدل تبدیل اعداد باینری به فرمت BCD INTEGER می باشد



Parameter	Data Type	Memory Area	Description
EN	BOOL	I, Q, M, D, L, T, C	Enable input
IN	WORD	I, Q, M, D, L or constant	Number in BCD format
OUT	INT	I, Q, M, D, L	Integer value of the BCD number
ENO	BOOL	I, Q, M, D, L	Enable output

مثال:

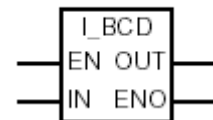


در مثال فوق

MW10 به صورت یک عدد BCD خوانده میشود و تبدیل به INTEGER می شود. و نتیجه در MW12 ریخته می شود. در صورت انجام کار Q4.0 یک میشود. (ENO=EN)

: I_BCD

Parameter	Data Type	Memory Area	Description
EN	BOOL	I, Q, M, D, L, T, C	Enable input
IN	INT	I, Q, M, D, L or constant	Integer
OUT	WORD	I, Q, M, D, L	BCD value of the integer
ENO	BOOL	I, Q, M, D, L	Enable output



این بلوک ورودی بصورت INTEGER بوده و خروجی بصورت BCD INTEGER می باشد

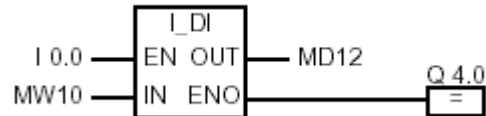
: BCD TO DOUBLE INTEGER(BCD_DI)

Parameter	Data Type	Memory Area	Description
EN	BOOL	I, Q, M, D, L, T, C	Enable input
IN	DWORD	I, Q, M, D, L or constant	Number in BCD format
OUT	DINT	I, Q, M, D, L	Double integer value of the BCD number
ENO	BOOL	I, Q, M, D, L	Enable output

: INTEGER TO DOUBLE INTEGER(I_DI)

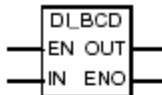
Parameter	Data Type	Memory Area	Description
EN	BOOL	I, Q, M, D, L, T, C	Enable input
IN	INT	I, Q, M, D, L or constant	Value to be converted
OUT	DINT	I, Q, M, D, L	Result
ENO	BOOL	I, Q, M, D, L	Enable output

مثال :



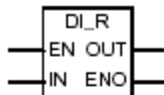
I0.0 را از MW10 به صورت INTEGER می گیرد تبدیل میکند به DOUBLE و جواب را در MW12 میریزد.

: DOUBLE INTEGER TO BCD(DI_BCD)



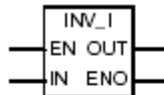
Parameter	Data Type	Memory Area	Description
EN	BOOL	I, Q, M, D, L, T, C	Enable input
IN	DINT	I, Q, M, D, L or constant	Double Integer
OUT	DWORD	I, Q, M, D, L	BCD value of the double integer
ENO	BOOL	I, Q, M, D, L	Enable output

: DOUBLE INTEGER TO REAL(DI_R)



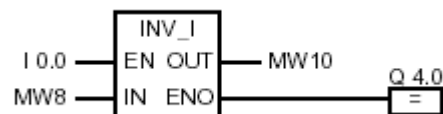
Parameter	Data Type	Memory Area	Description
EN	BOOL	I, Q, M, D, L, T, C	Enable input
IN	DINT	I, Q, M, D, L or constant	Value to be converted
OUT	REAL	I, Q, M, D, L	Result
ENO	BOOL	I, Q, M, D, L	Enable output

: ONES COMPLEMENT INTEGER(INV_I)



Parameter	Data Type	Memory Area	Description
EN	BOOL	I, Q, M, D, L, T, C	Enable input
IN	INT	I, Q, M, D, L or constant	Input value
OUT	INT	I, Q, M, D, L	Ones complement of the integer
ENO	BOOL	I, Q, M, D, L	Enable output

مثال:



در مثال بالا داریم: MW8=01000001 10000001 که فرم INTEGER است. و در خروجی مکمل ۱ آن یعنی:
10111110 01111110 در MW8 داریم.

: ONES COMPELEMENT DOUBLE INTEGER (INV_DI)



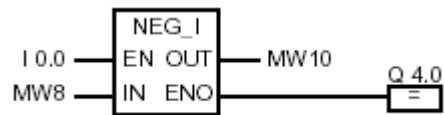
Parameter	Data Type	Memory Area	Description
EN	BOOL	I, Q, M, D, L, T, C	Enable input
IN	DINT	I, Q, M, D, L or constant	Input value
OUT	DINT	I, Q, M, D, L	Ones complement of the double integer
ENO	BOOL	I, Q, M, D, L	Enable output

: TWOS COMPELEMENT INTEGER(NEG_I)



Parameter	Data Type	Memory Area	Description
EN	BOOL	I, Q, M, D, L, T, C	Enable input
IN	INT	I, Q, M, D, L or constant	Input value
OUT	INT	I, Q, M, D, L	Twos complement of the integer
ENO	BOOL	I, Q, M, D, L	Enable output

مثال:



در مثال بالا $MW8=+10$ و $MW10=-10$ است.

: TWOS COMPELEMENT DOUBLE INTEGER(NEG_DI)



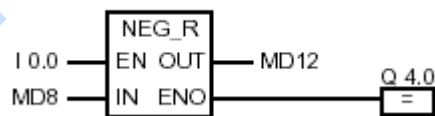
Parameter	Data Type	Memory Area	Description
EN	BOOL	I, Q, M, D, L, T, C	Enable input
IN	DINT	I, Q, M, D, L or constant	Input value
OUT	DINT	I, Q, M, D, L	Twos complement of the double integer
ENO	BOOL	I, Q, M, D, L	Enable output

: NEGATIV REAL NUMBER(NEG_R)

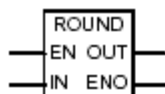


Parameter	Data Type	Memory Area	Description
EN	BOOL	I, Q, M, D, L, T, C	Enable input
IN	REAL	I, Q, M, D, L or constant	Input value
OUT	REAL	I, Q, M, D, L	The result is the negated input value.
ENO	BOOL	I, Q, M, D, L	Enable output

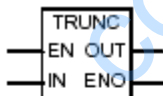
مثال:



در بالا داریم $MD12 = -6.234$ ، $MD8 = +6.234$ قسمت حقیقی را مثبت کرده است.
در زیر به نام بردن باقی المابها بسنده میکنیم:

: ROUND

Parameter	Data Type	Memory Area	Description
EN	BOOL	I, Q, M, D, L, T, C	Enable input
IN	REAL	I, Q, M, D, L or constant	Value to be rounded
OUT	DINT	I, Q, M, D, L	IN rounded to the next double integer
ENO	BOOL	I, Q, M, D, L	Enable output

: TRUNC

Parameter	Data Type	Memory Area	Description
EN	BOOL	I, Q, M, D, L, T, C	Enable input
IN	REAL	I, Q, M, D, L or constant	Value to be truncated
OUT	DINT	I, Q, M, D, L	Integer component of IN
ENO	BOOL	I, Q, M, D, L	Enable output

LAD/STL/FBD - [OB1 -- training1\SIMATIC 300 Station\CPU315(1)]

File Edit Insert PLC Debug View Options Window Help

Address	Declaration	Name	Type	Initial v	Comment
0.0	temp	OB1_EV_CLASS	BYTE		Bits 0-3 = 1

OB1 : "Main Program Sweep (Cycle)"

Comment:

Network 1: Title:

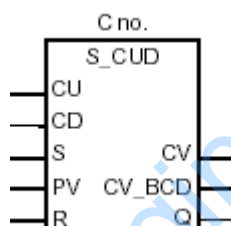
Comment:

- New network
- Bit logic
- Comparator
- Converter**
 - BCD_I
 - L_BCD
 - L_DI
 - BCD_DI
 - DI_BCD
 - DI_R
 - INV_I
 - INV_DI
 - NEG_I
 - NEG_DI
 - NEG_R
 - ROUND
 - TRUNC
 - CEIL
 - FLOOR
- Counter
- DB call
- Jumps
- Integer fct.
- Floating-point fct.
- Move
- Program control
- Shift/Rotate
- Status bits
- Conversion instructions

Press F1 to get Help. offline Abs Nw 1 Insert

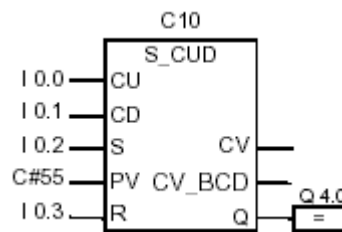
شمارنده ها (COUNTER) :

: ASSIGN PARAMETER AND COUNT UP/DOWN (S_CUD)



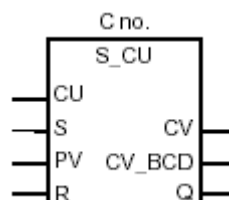
Parameter English	Parameter German	Data Type	Memory Area	Description
no.	Nr.	COUNTER	C	Counter identification number. The range depends on the CPU.
CU	ZV	BOOL	I, Q, M, D, L	ZV input: Up Counter
CD	ZR	BOOL	I, Q, M, D, L	ZR input: Down Counter
S	S	BOOL	I, Q, M, D, L, T, C	Input for presetting the counter
PV	ZW	WORD	I, Q, M, D, L or constant	Count value in the range between 0 and 999 or Count value entered as C#<value> in BCD format
R	R	BOOL	I, Q, M, D, L, T, C	Reset input
CV	DUAL	WORD	I, Q, M, D, L	Current count value (hexadecimal number)
CV_BCD	DEZ	WORD	I, Q, M, D, L	Current count value (BCD format)
Q	Q	BOOL	I, Q, M, D, L	Status of the counter

مثال:



در شمارنده فوق با تغییر سیگنال از ۰ به ۱ در $I0.2$ SET می شود. $C\#55$ یعنی مقدار شمارش تا ۵۵ دفعه می باشد. هنگامی که $I0.0$ از ۰ به ۱ تغییر کند شمارنده به صورت افزایشی تا ۹۹۹ به بالا می شمارد. و اگر $I0.1$ از ۰ به ۱ تغییر کند به صورت کاهش تا ۰ می شمارد. هنگامی که $I0.3$ از ۰ به ۱ تغییر کند مقدار COUNTER در ۰ ست می شود.

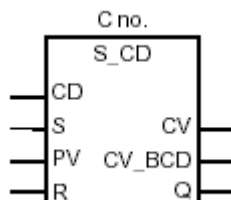
:ASSIGN PARAMETER AND COUNT UP (S_CU)



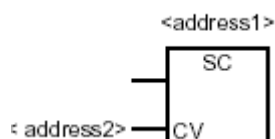
Parameter English	Parameter German	Data Type	Memory Area	Description
no.	Nr.	COUNTER	C	Counter identification number. The range depends on the CPU.
CU	ZV	BOOL	I, Q, M, D, L	ZV input: Up Counter
S	S	BOOL	I, Q, M, D, L, T, C	Input for presetting the counter
PV	ZW	WORD	I, Q, M, D, L or constant	Count value in the range between 0 and 999 or Count value entered as C#<value> in BCD format
R	R	BOOL	I, Q, M, D, L, T, C	Reset input
CV	DUAL	WORD	I, Q, M, D, L	Current count value (hexadecimal number)
CV_BCD	DEZ	WORD	I, Q, M, D, L	Current count value (BCD format)
Q	Q	BOOL	I, Q, M, D, L	Status of the counter

: ASSIGN PARAMETER AND COUNT DOWN (S_CD)

با جدول مشابه بالا

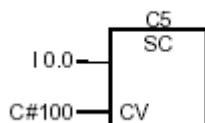


: SET COUNTER VALUE

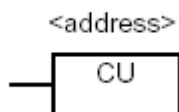


Parameter English	Parameter German	Data Type	Memory Area	Description
Counter no.	Counter no	COUNTER	C	The address1 specifies the number of the counter which is to be preset with a value.
CV	ZW	WORD	I, Q, M, D, L or constant	The value to be preset (address2) can lie between 0 and 999. When you enter a constant, C# must stand in front of the value specified by the BCD format, for example, C#100.

مثال:



شمارنده فوق مقدار ۱۰۰ به COUNTER VALUE داده شده و با تحریک I0.0 مقدار کانتر به ۱۰۰ می رود در غیر اینصورت اتفاقی نمی افتد. (از ۱ به ۰ برویم)

: UP COUNTER(CU)

Parameter	Data Type	Memory Area	Description
Counter no.	COUNTER	C	The address specifies the number of the counter which is to be incremented.

شمارنده فوق به صورت صعودی تا ۹۹۹ می شمارد.

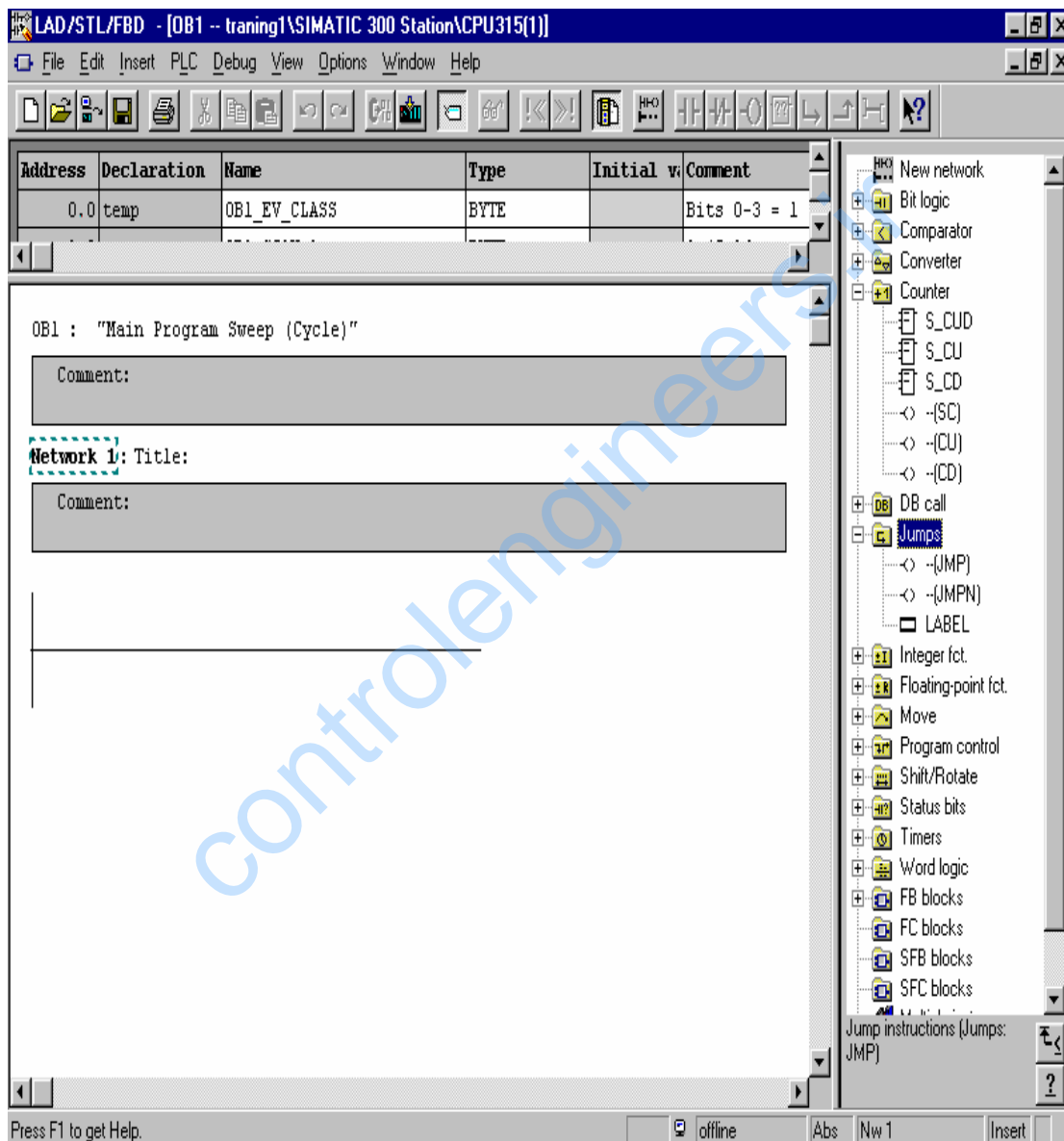
DOWN COUNTER(CD)

شمارنده معکوس



Parameter	Data Type	Memory Area	Description
Counter no.	COUNTER	C	The address specifies the number of the counter which is to be decremented.

در شکل زیر در قسمت چپ روش انتخاب انواع شمارنده ها از داخل کتابخانه S7 مشخص گردیده است

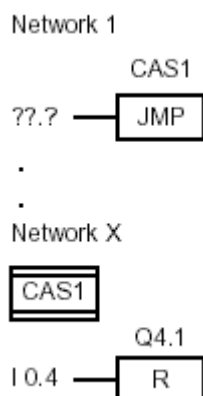


دستورات پرش (JUMP) :

: UN CONDITIONAL JUMP TO BLOCK

Parameter	Data Type	Memory Area	Description
Name of a jump label	-	-	The address specifies the label to which the program will jump unconditionally.

مثال:

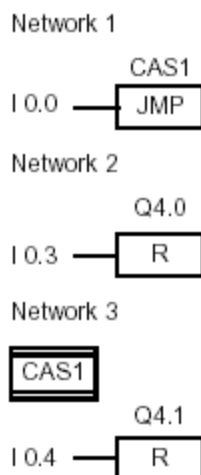


: CONDITIONAL JUMP IN A BLOCK



Parameter	Data Type	Memory Area	Description
Name of a jump label	-	-	The address specifies the label to which the program will jump if the RLO is 1.

مثال:



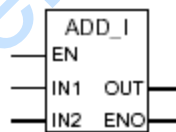
زمانیکه IO.0 فعال شود (یک شود) CASI اجرا می شود.
 زمانیکه IO.3 فعال شود (یک شود) فرمان RESET می آید.

:INTEGER MATH INSTRUCTION

کاربرد دستورات ریاضی برای اعداد صحیح در نرم افزار

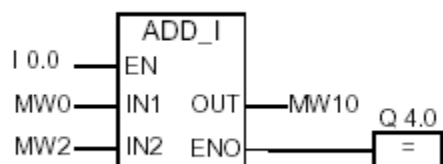
ADD INTEGER(ADD_I)

دستور جمع اعداد صحیح



Parameter	Data Type	Memory Area	Description
EN	BOOL	I, Q, M, D, L, T, C	Enable input
IN1	INT	I, Q, M, D, L or constant	First value for addition
IN2	INT	I, Q, M, D, L or constant	Second value for addition
OUT	INT	I, Q, M, D, L	Result of addition
ENO	BOOL	I, Q, M, D, L	Enable output

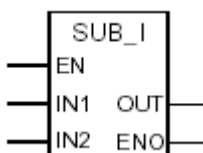
مثال:



با فعال شدن این بلوک بوسیله ورودی I0.0 شروع به جمع می کند.
 ۲ ورودی MW0 و MW2 را می گیرد و با هم جمع میکند و خروجی را در MW10 میریزد

اگر حاصل در رنج مورد نظر نباشد یا به هر دلیل غیر قابل قبول باشد I0.0 صفر میشود و Q4.0 را صفر می کند.

SUBTRACT INTEGER(SUB_I)



Parameter	Data Type	Memory Area	Description
EN	BOOL	I, Q, M, D, L, T, C	Enable input
IN1	INT	I, Q, M, D, L or constant	Minuend (value from which second value is subtracted)
IN2	INT	I, Q, M, D, L or constant	Subtrahend (value subtracted from the first value)
OUT	INT	I, Q, M, D, L	Result of subtraction
ENO	BOOL	I, Q, M, D, L	Enable output

تایمرها (TIMER INSTRUCTIONS) :

انواع تایمر عبارتند از:

(۱)

(PULSE TIMER) S_PULSE

(۲)

(EXTENDED PULSETIMER) S_PEXT

(۳)

(ON-DELAY TIMER) S_ODT

(۴)

(RETENTIVE ON -DELAY TIMER) S_ODTS

(۵)

(OF DELAY TIMER) S_OFFDT

(۶)

SP

(۷)

SD

(۸)

SE

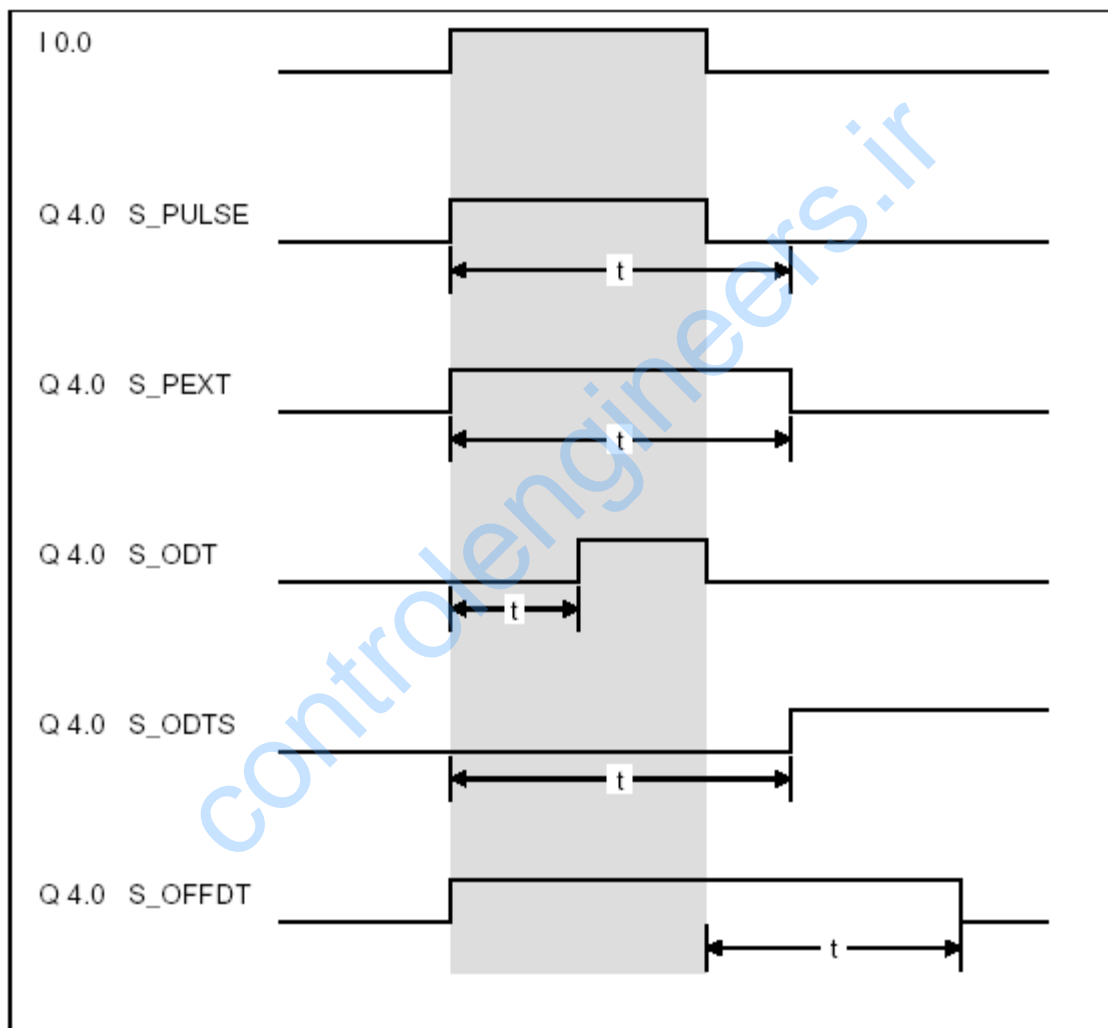
(۹)

SS

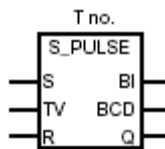
(۱۰)

SF

ابتدا مقایسه ای کلی در مورد تایمر ها و تفاوت های بین آنها از روی شکل در زیر ارائه می‌دهیم:

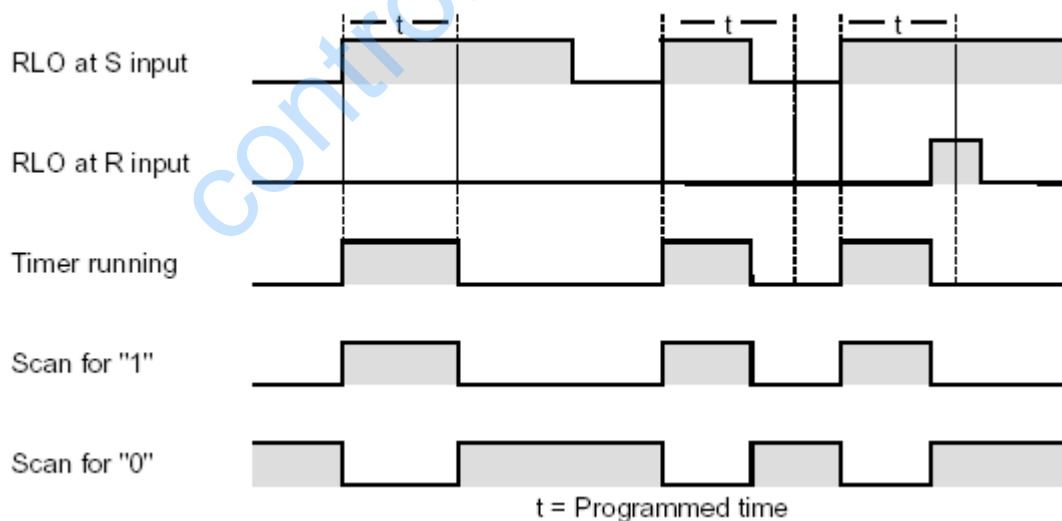


S_PULSE



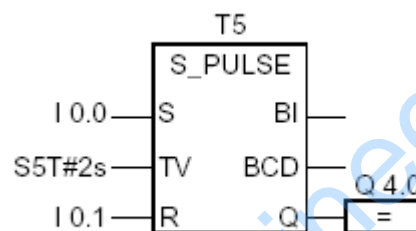
Parameter English	Parameter German	Data Type	Memory Area	Description
no.	Nr.	TIMER	T	Timer identification number. The range depends on the CPU.
S	S	BOOL	I, Q, M, D, L, T, C	Start input
TV	TW	S5TIME	I, Q, M, D, L or constant	Preset time value (range 0-9999)
R	R	BOOL	I, Q, M, D, L, T, C	Reset input
BI	DUAL	WORD	I, Q, M, D, L	Time remaining (value in integer format)
BCD	DEZ	WORD	I, Q, M, D, L	Time remaining (value in BCD format)
Q	Q	BOOL	I, Q, M, D, L	Status of the timer

پاسخ این تایمر را به ورودی های مختلف SET, RESET را ملاحظه میکنیم:



که پالس RLO می آید و با توجه به RESET پاسخ را ملاحظه می کنیم:
 TIMER بدون توجه به پالس به مقدار زمان خود روشن است و پس از پایان یافتن زمانش
 به پایان می رسد

مثال



در ابتدا باید در مورد نامگذاری تایمر ها و چگونگی زمان دادن به آن به نکات زیر توجه کرد:

Time Base	Binary Code for the Time Base
10 ms	00
100 ms	01
1 s	10
10 s	11

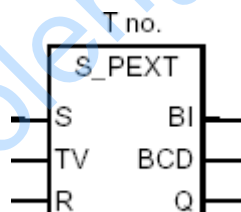
شکل باینری زمان ها به فرم بالاست.

S5T شکل استاندارد نمایش تایمر هاست.

زمان مورد نظر را به فرم #5 نشان داده میشود که در پایه TV نوشته می شود.

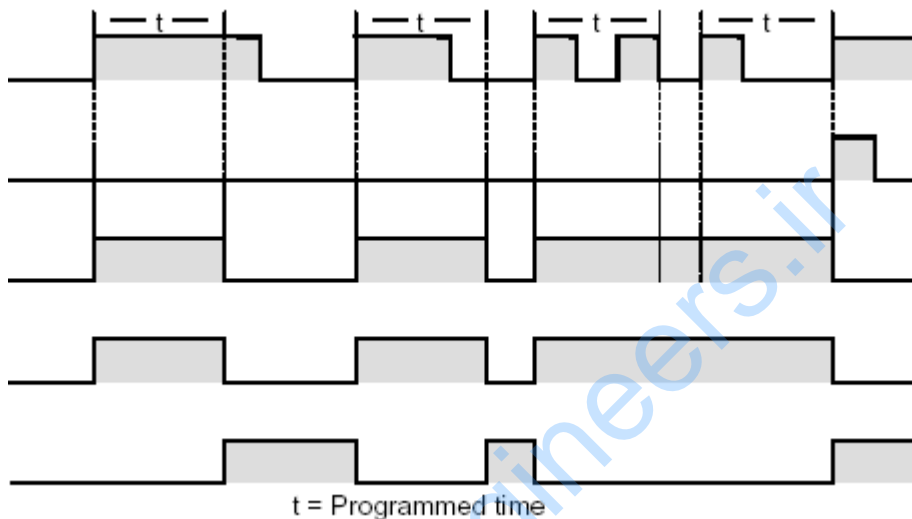
در مثال بالا با تحریک I0.0 یعنی SET تایمر فعال و ۲ ثانیه می شمارد و با فعال شدن RESET دیگر تایمر نمی شمارد. خروجی در Q4.0 ریخته می شود و به ۲ فرم باینری و BCD نمایش می دهد.

S_PEXT

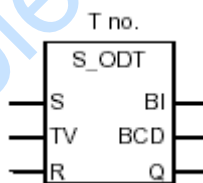


Parameter English	Parameter German	Data Type	Memory Area	Description
no.	Nr.	TIMER	T	Timer identification number. The range depends on the CPU.
S	S	BOOL	I, Q, M, D, L, T, C	Start input
TV	TW	S5TIME	I, Q, M, D, L or constant	Preset time value (range 0-9999)
R	R	BOOL	I, Q, M, D, L, T, C	Reset input
BI	DUAL	WORD	I, Q, M, D, L	Time remaining (value in integer format)
BCD	DEZ	WORD	I, Q, M, D, L	Time remaining (value in BCD format)
Q	Q	BOOL	I, Q, M, D, L	Status of the timer

تا انتهای TIME می‌رود و منتظر لبه بالا رونده برای تحریک بعدی می‌ماند و اگر لبه نیاید ادامه می‌یابد

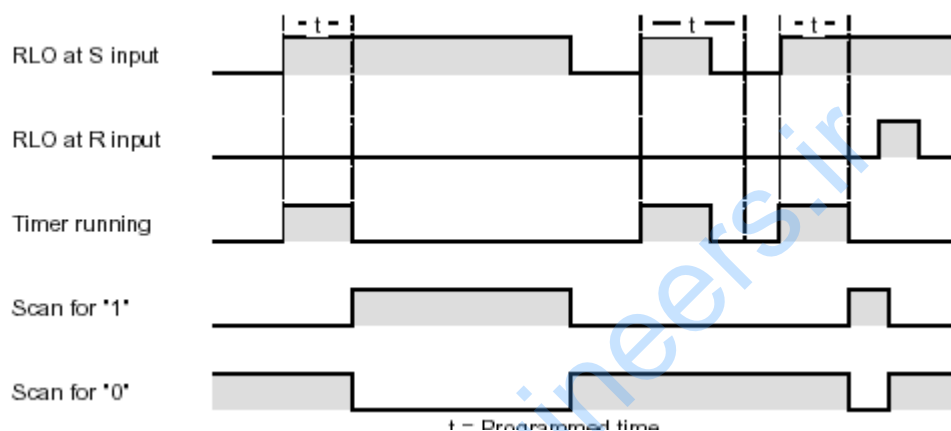


: S_ODT

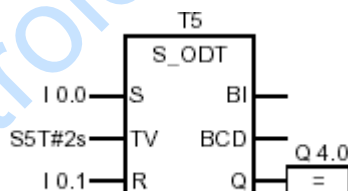


Parameter English	Parameter German	Data Type	Memory Area	Description
no.	Nr.	TIMER	T	Timer identification number. The range depends on the CPU.
S	S	BOOL	I, Q, M, D, L, T, C	Start input
TV	TW	S5TIME	I, Q, M, D, L or constant	Preset time value (range 0-9999)
R	R	BOOL	I, Q, M, D, L, T, C	Reset input
BI	DUAL	WORD	I, Q, M, D, L	Time remaining (value in integer format)
BCD	DEZ	WORD	I, Q, M, D, L	Time remaining (value in BCD format)
Q	Q	BOOL	I, Q, M, D, L	Status of the timer

با تحریک TIMER پالس به اندازه زمان خود می آید و منتظر لبه بالا رونده بعدی می ماند و
 TIMER بدون توجه به اندازه پالس و با توجه به لبه تحریک می شود.

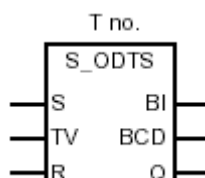


مثال:



با تحریک I0.0 تایمر به کار می افتد و اگر ۲ ثانیه به پایان رسد و I0.0 همچنان فعال باشد
 Q4.0 همچنان ۱ است و اگر I0.0 تغییر حالت دهد Q4.0 صفر می شود اگر I0.1 تغییر
 حالت دهد از ۰ به ۱ در زمانیکه TIMER کار می کند تایمر RESTART می شود.

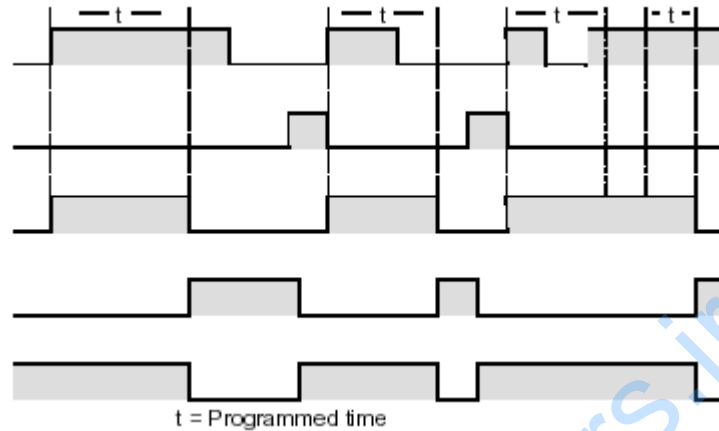
: S_ODTS



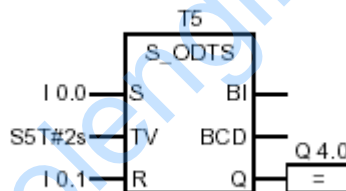
Parameter English	Parameter German	Data Type	Memory Area	Description
no.	Nr.	TIMER	T	Timer identification number. The range depends on the CPU.
S	S	BOOL	I, Q, M, D, L, T, C	Start input
TV	TW	S5TIME	I, Q, M, D, L or constant	Preset time value (range 0-9999)
R	R	BOOL	I, Q, M, D, L, T, C	Reset input
BI	DUAL	WORD	I, Q, M, D, L	Time remaining (value in integer format)
BCD	DEZ	WORD	I, Q, M, D, L	Time remaining (value in BCD format)
Q	Q	BOOL	I, Q, M, D, L	Status of the timer

نکته :

در این تایمر بعد از آمدن فرمان تایمر زمان مورد نظر را سپری می کند و تا آمدن لبه بالا رونده ادامه می یابد و اگر لبه نیاید پالس هم نداریم.

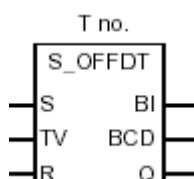


مثال:



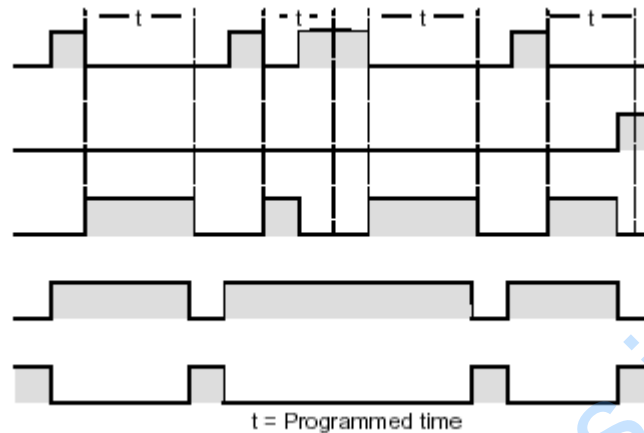
وقتی که I0.0 از ۰ به ۱ (لبه بالا رونده) تغییر حالت دهد تایمر بر خلاف تغییر سیگنال عمل میکند یعنی اگر I0.0 از ۰ به ۱ تغییر حالت پیدا کند بعد از گذشت زمان T تایمر RESTART می شود و اگر در زمانیکه تایمر کار میکند I0.1 از ۰ به ۱ تغییر وضعیت دهد تایمر RESTART می شود. و Q4.0 در زمانیکه زمان تایمر تمام شده و I0.1 صفر است روشن است.

: S_OFFDT

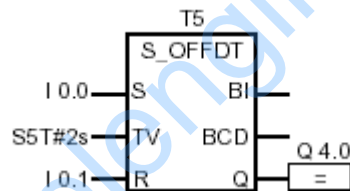


Parameter English	Parameter German	Data Type	Memory Area	Description
no.	Nr.	TIMER	T	Timer identification number. The range depends on the CPU.
S	S	BOOL	I, Q, M, D, L, T, C	Start input
TV	TW	S5TIME	I, Q, M, D, L or constant	Preset time value (range 0-9999)
R	R	BOOL	I, Q, M, D, L, T, C	Reset input
BI	DUAL	WORD	I, Q, M, D, L	Time remaining (value in integer format)
BCD	DEZ	WORD	I, Q, M, D, L	Time remaining (value in BCD format)
Q	Q	BOOL	I, Q, M, D, L	Status of the timer

این تایمر با لبه پایین رونده فعال میشود و تا پایان TIME میروود اگر در این زمان پالس پایین رونده دیگری بیاید پالس به پایان میروود و اگر نه تا انتهای پالس میروود.



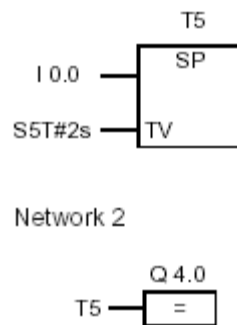
مثال:



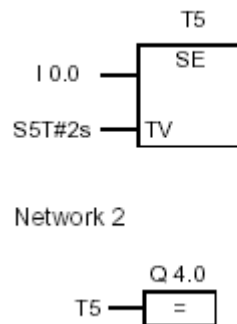
اگر I0.0 از ۱ به ۰ تغییر کند تایمر بکار می افتد و Q4.0 زمانی ۱ است که I0.0 یک باشد یا تایمر در حال کار باشد و I0.1 اگر در زمان کار کردن تایمر فعال شود تایمر RESTART میشود.

: SP(START PULS TIMER)

مثال:



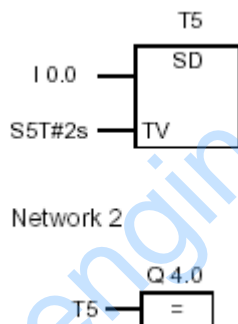
اگر I0.0 از ۱ به ۰ تغییر کند تایمر بکار می افتد و تا زمانی که SIGNAL STATE یک است تایمر می شمارد و تایمر مرتباً پالس مربعی با فواصل معین ایجاد می کند.

: SE (START EXTENDED PULSE TIMER)

اگر I0.0 از ۱ به ۰ تغییر کند تایمر بکار می افتد و بدون تاثیر از لبه پایین رونده ادامه می یابد و اگر I0.0 بعد از اتمام TIME فعال شود تایمر RESTART می شود.

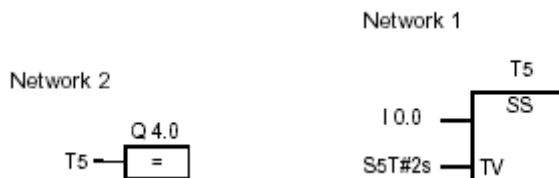
: SD (START ON DELAY TIMER)

مثال:

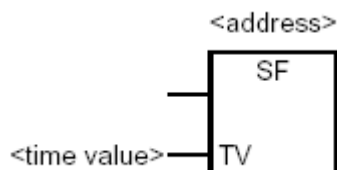


اگر I0.0 از ۱ به ۰ تغییر کند تایمر بکار می افتد و اگر I0.0 یک باشد و تایمر نیز از شمارش خارج Q4.0 یک است و اگر SIGNAL STATE از ۱ به ۰ تغییر کند TIMER ری استارت می شود.

: SS (START RETENTIVE ON DELAY)

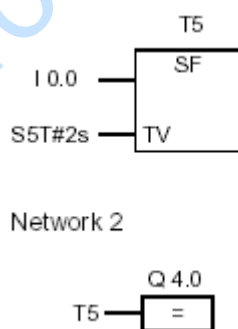


: SF (STSRT OFF DELAY TIMER)



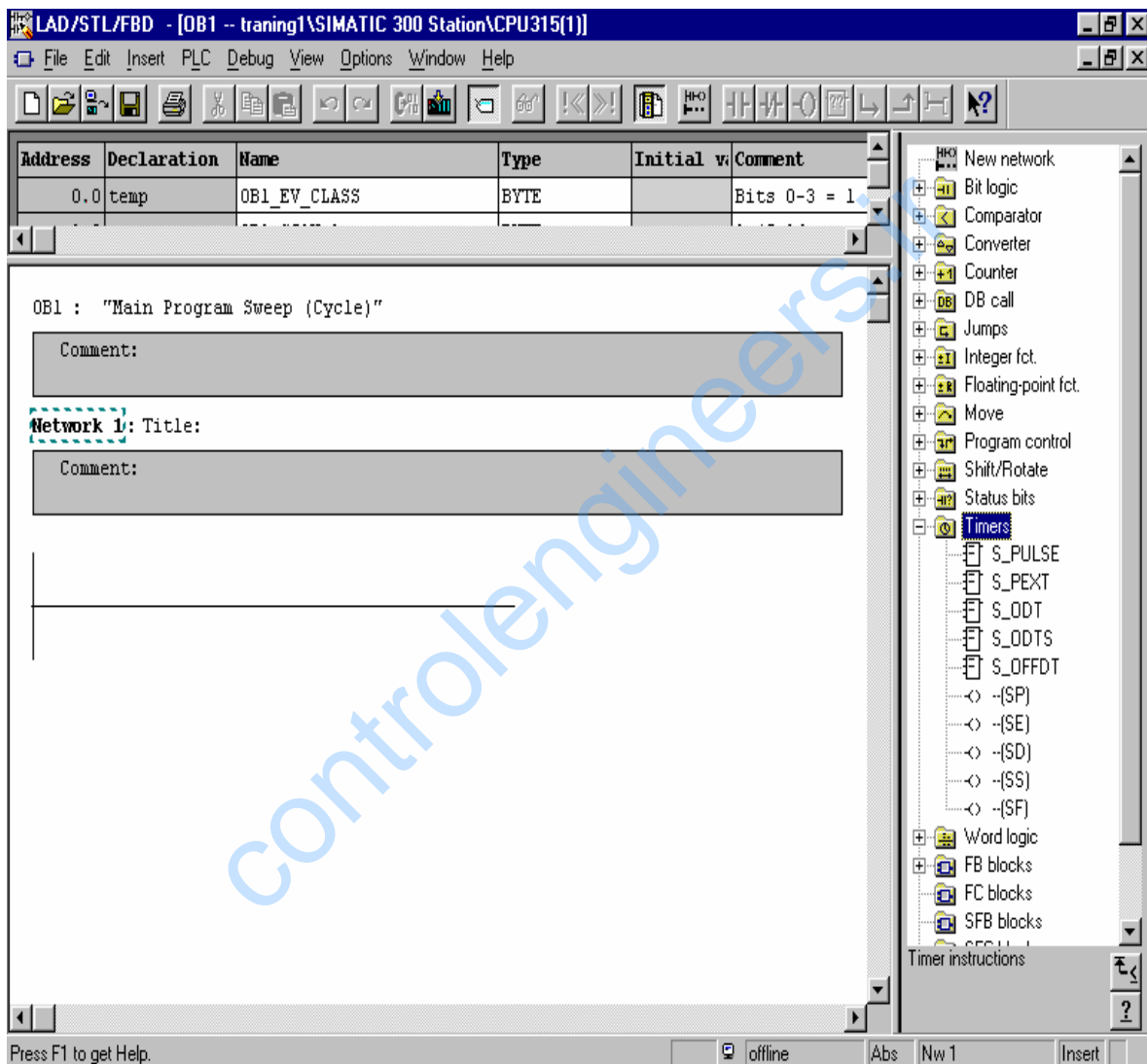
Parameter English	Parameter German	Data Type	Memory Area	Description
Timer no.	Timer no.	TIMER	T	The address specifies the number of the timer which is to be started.
TV	TW	S5TIME	E, A, M, D, L or constant	Timer value (S5TIME format)

مثال:



زمانیکه **SIGNSL STSTE** از ۱ به ۰ تغییر کند تایمر بکار می افتد و اگر از ۰ به ۱ برود **TIMER** ری استارت می شود **Q4.0** زمانی ۱ است که یا **IO.0** یک باشد یا تایمر در حال کار باشد.

در شکل زیر نمای تایمر ها را در برنامه S7 در قسمت library ملاحظه می کنیم:



شیفترها (SHIFT INSTRUCTION)

انواع شیفتر عبارتند از:

SHR_I (SHIFT RIGH INTEGER)

SHR_DI (SHIFT RIGH DOUBLE INTEGER)

SHL_W (SHIFT LEFT WORD)

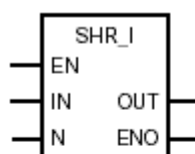
SHR_W (SHIFT RIGHT WORD)

SHL_DW (SHIFT LEFT DOUBLE WORD)

SHR_DW (SHIFTRIGH DOUBLE WORD)

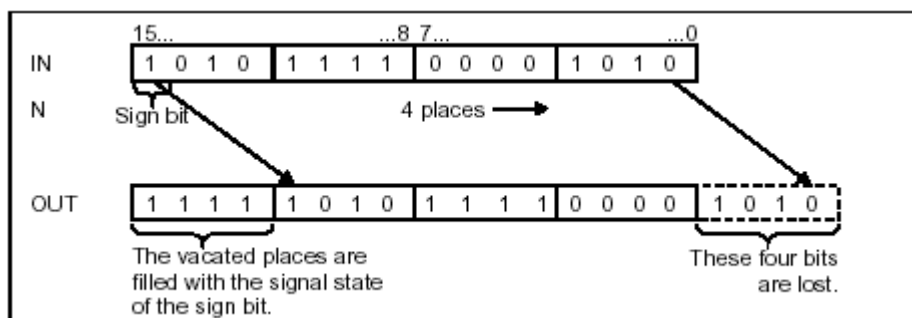
که در زیر به معرفی چندی از آنها می پردازیم:

: SHR_I

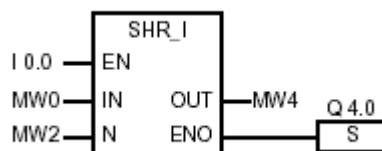


Parameter	Data Type	Memory Area	Description
EN	BOOL	I, Q, M, L, D, T, C	Enable input
IN	INT	I, Q, M, L, D	Value to be shifted
N	WORD	I, Q, M, L, D	Number of bit positions by which the value will be shifted
OUT	INT	I, Q, M, L, D	Result of the shift instruction
ENO	BOOL	I, Q, M, L, D	Enable output

که در شکل نحوه شیفت دادن آن را ملاحظه میکنیم:

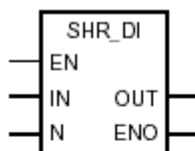


مثال:



با فعال شدن I0.0 شیفتر فعال شده و آنچه در MW0 است را به تعداد بیتی که MW2 اعلام می کند به راست شیفت می دهد و نتیجه را در MW4 می ریزد.

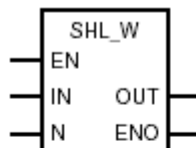
SHR_DI



Parameter	Data Type	Memory Area	Description
EN	BOOL	I, Q, M, L, D, T, C	Enable input
IN	DINT	I, Q, M, L, D	Value to be shifted
N	WORD	I, Q, M, L, D	Number of bit positions by which the value will be shifted
OUT	DINT	I, Q, M, L, D	Result of the shift instruction
ENO	BOOL	I, Q, M, L, D	Enable output

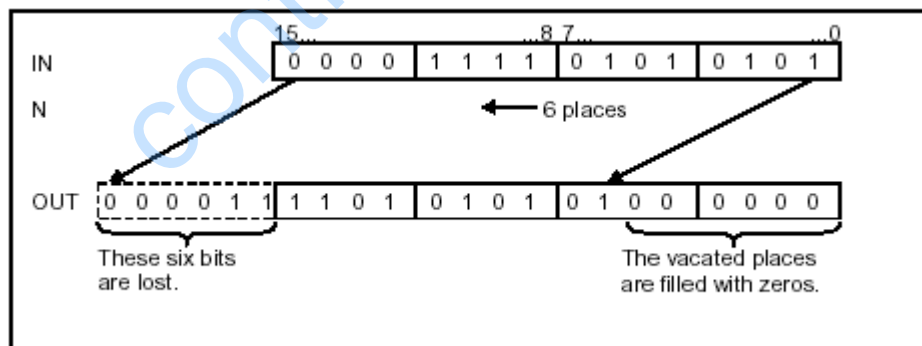
همانند قبل است اما این بار DOUBLE INTEGER است.

: SHL_W (SHIFT LEFT WORD)



Parameter	Data Type	Memory Area	Description
EN	BOOL	I, Q, M, L, D, T, C	Enable input
IN	WORD	I, Q, M, L, D	Value to be shifted
N	WORD	I, Q, M, L, D	Number of bit positions by which the value will be shifted
OUT	WORD	I, Q, M, L, D	Result of the shift instruction
ENO	BOOL	I, Q, M, L, D	Enable output

که فرم عددی که میگیرد WORD است و همانند شکل زیر به چپ شیفت می دهد.

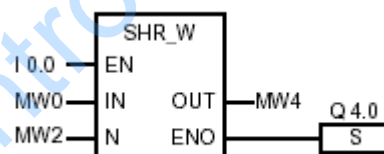


SHR_W(SHIFT RIGHT WORD)

همانند بالاست با این تفاوت که به سمت راست شیفت می دهد.

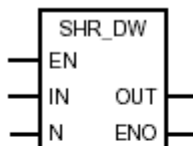
Parameter	Data Type	Memory Area	Description
EN	BOOL	I, Q, M, L, D, T, C	Enable input
IN	WORD	I, Q, M, L, D	Value to be shifted
N	WORD	I, Q, M, L, D	Number of bit positions by which the value will be shifted
OUT	WORD	I, Q, M, L, D	Result of the shift instruction
ENO	BOOL	I, Q, M, L, D	Enable output

مثال:

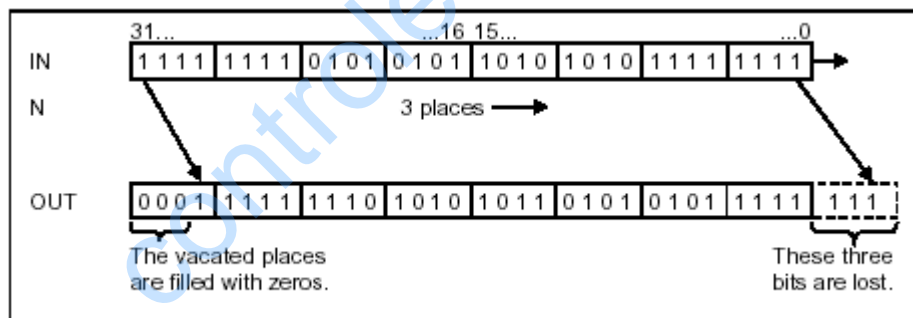


با ۱ شدن I0.0 فعال می شود و محتویات MW0 را به تعداد بیت موجود در MW2 به صورت WORD به سمت راست شیفت می دهد و نتیجه را در MW4 میریزد.

SHR_DW (SHIFT RIGHT DOUBLE WORD)



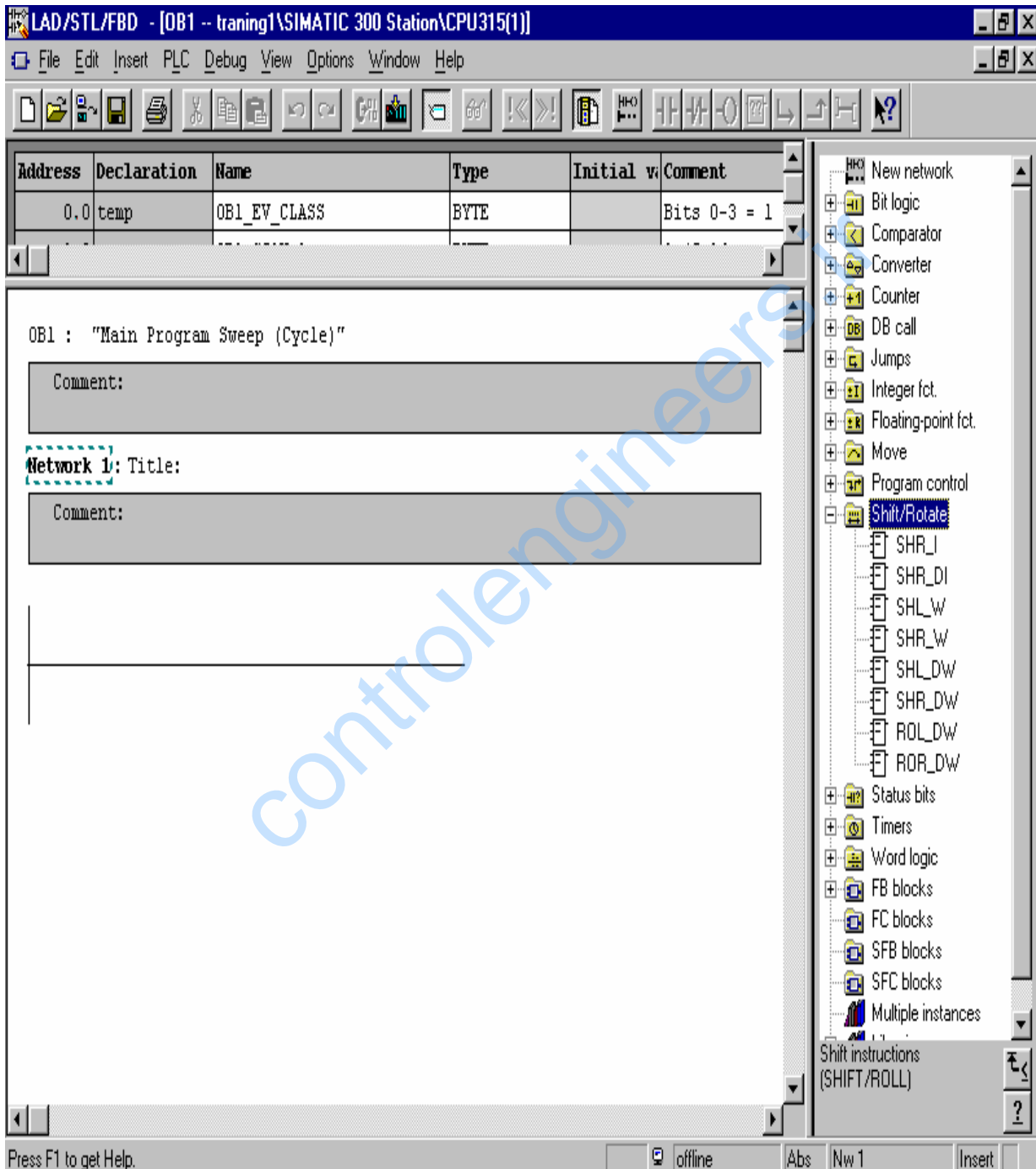
Parameter	Data Type	Memory Area	Description
EN	BOOL	I, Q, M, L, D, T, C	Enable input
IN	DWORD	I, Q, M, L, D	Value to be shifted
N	WORD	I, Q, M, L, D	Number of bit positions by which the value will be shifted
OUT	DWORD	I, Q, M, L, D	Result of the shift instruction
ENO	BOOL	I, Q, M, L, D	Enable output



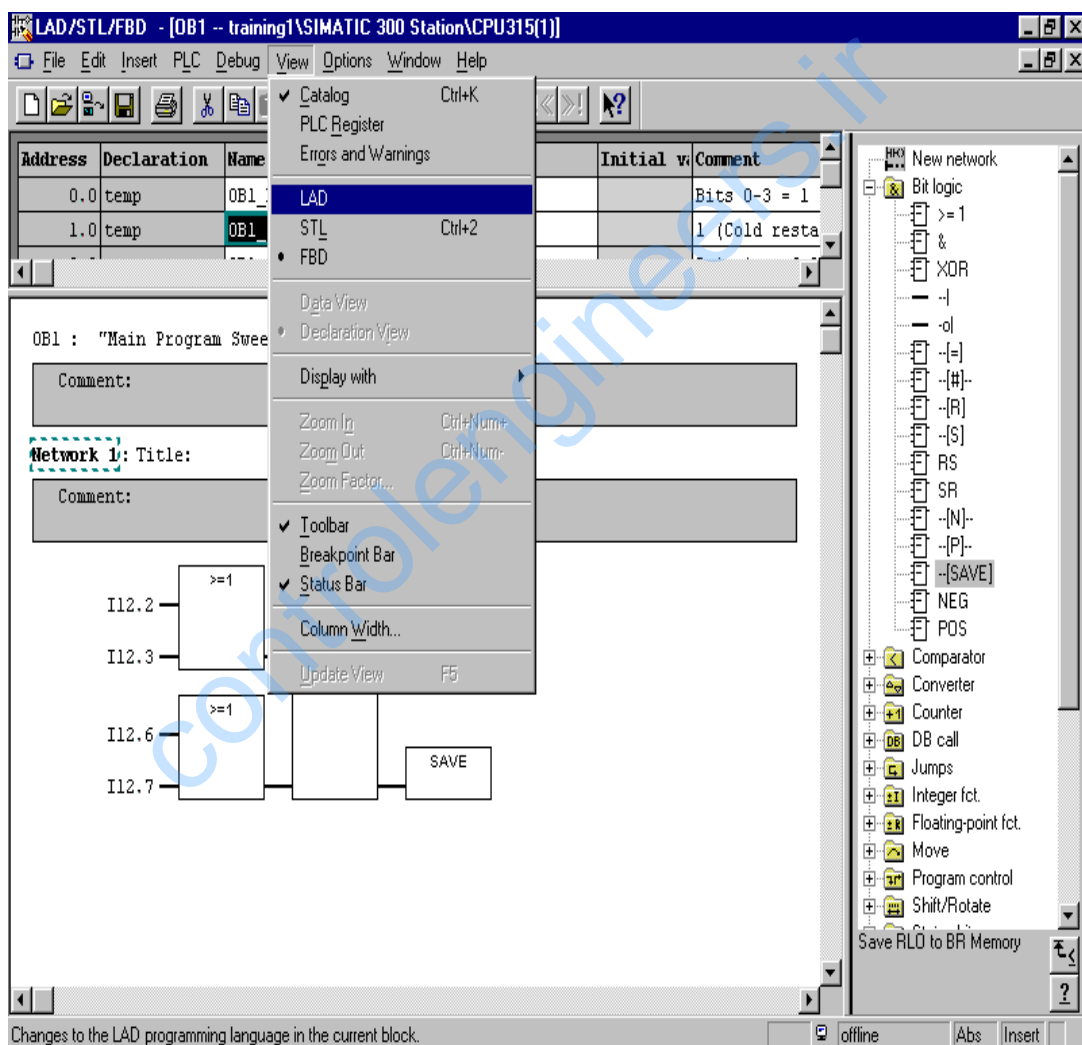
که فرم اطلاعات به صورت DOUBLE WORD را می گیرد و به سمت راست

شیفت میدهد

در این شکل نحوه انتخاب دستور شیفت را از داخل library برنامه s7 مشاهده می کنیم



در نهایت گزارش را با طرح بر نامه ای ساده در محیط FBD و LADD به پایان می
بریم:



LAD/STL/FBD - [OB1 -- training1\SIMATIC 300 Station\CPU315(1)]

File Edit Insert PLC Debug View Options Window Help

Symbolic name	Symbolic comment	Address	Data type	Initial value	Cycle time of update
8.0 temp	OB1_MIN_CYCLE	INT			Minimum cycle
10.0 temp	OB1_MAX_CYCLE	INT			Maximum cycle
12.0 temp	OB1_DATE_TIME	DATE_AND_TIME			Date and time

OB1 : "Main Program Sweep (Cycle)"

Comment:

Network 1: Title:

Comment:

Function blocks of the project

- New network
- Bit logic
- Comparator
- Converter
- Counter
- DB call
- Jumps
- Integer fct.
- Floating-point fct.
- Move
- Program control
- Shift/Rotate
- Status bits
- Timers
- Word logic
- FB blocks
 - FB1
- FC blocks
- SFB blocks
- SFC blocks
- Multiple instances
- Libraries

Press F1 to get Help. offline Abs Insert

تقدیر و تشکر :

در نهایت جا دارد از تمام دوستان بخاطر قصوراتی که در این جزوه ملاحظه نموده اند عذر خواهی نمایم و از ایشان تمنا دارم که بنده را برای رفع مشکلات و ایرادات موجود راهنمایی و یاری بفرمایند .

همانطور که کارشناسان محترم این رشته اطلاع دارند مطالب موجود در مراجع زیمنس بسیار حجیم و زیاد بوده و نمی توان تمامی مطالب را بصورت یک کتابچه تدوین نمود .

مطالب ذکر شده در فوق در جقیقت قسمت اول شروع کار با نرم افزار s7 و Plc زیمنس می باشد که امیدوارم برای تمامی همکاران عزیز بعنوان یک **reference** مفید واقع گردد

انشالله در آینده نزدیک سعی در ارائه قسمت دوم این کتابچه که کاربرد برنامه نویسی با نرم افزار s7 بوده و مثالهای عملی و مفید در آن بکار برده شده باشد ادامه می دهیم و همچنین مطالبی درباره نحوه **network configuration** و ارتباط شبکه ایی plc با قطعات داخلی و همچنین plc ها با یکدیگر , و همچنین نحوه **setting** پانل های اپراتوری زیمنس **op3-7-27-37-270** و کار با نرم افزار های **graph & simulation** را به استحضار خوانندگان عزیز می رسانم .

امیدوارم که خداوند در این راه فرصت کافی را در اختیار اینجانب قرار دهد تا بتوانیم خدمتی را برای کشور عزیزمان و جامعه صنعتی و اتوماسیون بنامیم.

چکیده:

آشنایی با PLC و نرم افزار s7 , انواع بلوک در s7 ,
فرم های نمایش در s7 , تایمر ها , شمارنده ها,
مقایسه کننده ها , تبدیل کننده ها و شیفت دهنده ها و سخت افزار در s7

منابع بکار گرفته شده در این مجموعه :

- Working with step 7 (siemens manuals)
- Configuring hardware with step 7 (siemens manuals)
- Getting started with step 7 (siemens manuals)
- Ladder logic for s7-300 and s7-400 programming (siemens manuals)
- Function block diagram for s7-300 and s7-400 programming (siemens manuals)

ومن الله توفيق

محمد یادگار

کارشناس فنی برق و الکترونیک

اداره کل تعمیرات مونتاز یک

اداره پشتیبانی مهندسی