مقدمه ای بر STEP7

1-1 آنچه شما خواهید آموخت

با استفاده از تمرینات عملی ، به شما نشان خواهیم داد که برنامه نویسی بزبان منطق نردبانی، نمایش عبارتی یا دیاگرام بلوکی توسط **STEP7** بسیار ساده می باشد.

دستورات جزئی تر در فصل های مجزا و بصورت قدم به قدم، روشهای کار با **STEP7** را بشما نشان خواهند داد.

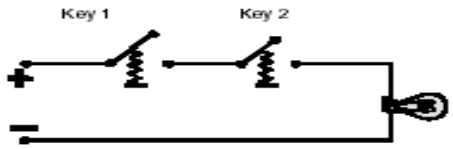
ایجاد یک برنامه با استفاده از منطق دودویی

در فصول 2 تا 7 شما با استفاده از منطق دودویی یک برنامه ایجاد خواهید نمود.

با استفاده از دستورات منطقی برنامه ریزی شده ، شما ورودی های و خروجی های CPU خود را (در صورت وجود) آدرس دهی خواهید نمود.

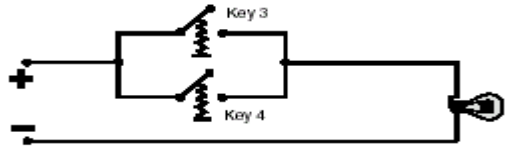
مثال های برنامه نویسی در جزوه مقدماتی از بین روشهای مختلفی که وجود دارد بر مبنای سه دسته دستورالعمل های منطق دودویی اساسی مطرح گردیده اند.

اولین عمل منطق دودویی که شما بعداً بر اساس آن برنامه خواهید نوشت تابع AND (و) می باشد تابع AND را میتوان در یک مدار و با استفاده از دو کلید نمایش داد.



اگر دو کلید 1 و 2 فشار داده شوند لامپ روشن می شود.

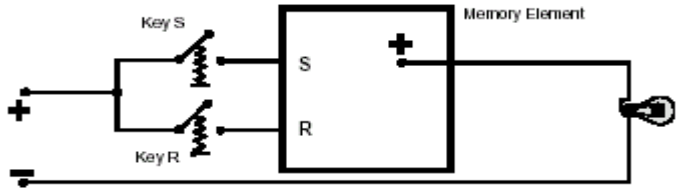
دومین عمل منطقی دودویی، تابع OR (یا) می باشد. تابع OR را میتوان بصورت مدار زیر نمایش داد:



اگر هریک از کلیدهای 3 یا 4 فشرده شود لامپ روشن می شود.

سومین عمل منطق دودویی عنصر حافظه می باشد.

تابع S/R (ست/ری ست) در یک مدار الکتریکی جهت و تامین یک سطح ولتاژ معین بکار میرود.

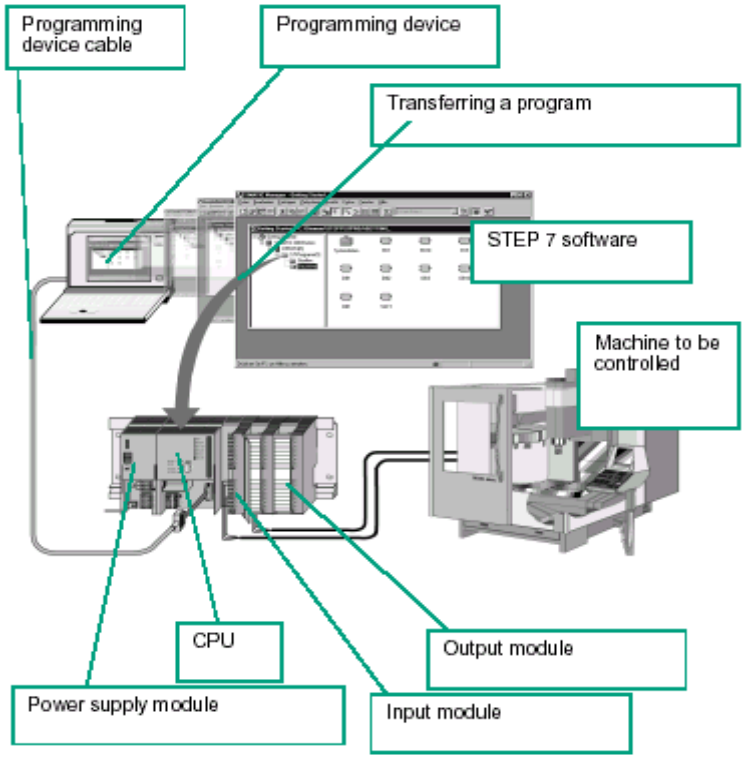


اگر کلید S فشار داده شود چراغ روشن شده و

تا زمانیکه کلید R فشار داده شود روشن می ماند.

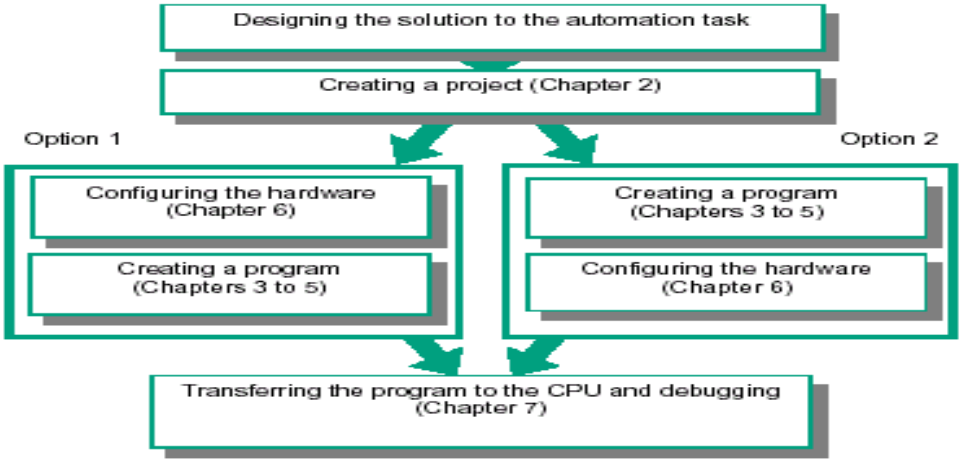
1–2 ترکیب سخت افزار و نرم افزار

با استفاده از نرم افزار STEP7 شما می توانید در قالب یک پروژه، برنامه S7 خویش را بسازید. کنترلرهای قابل برنامه ریزی S7 شامل یک منبع تغذیه، یک CPU و کارت های ورودی و خروجی (I/O Modules) می باشند. کنترلرهای منطقی قابل برنامه ریزی (PLC) ماشین آلات را با استفاده از برنامه S7 مونیتور و کنترل می نمایند.



1–3 مراحل مقدماتی استفاده از STEP7

قبل از ایجاد یک پروژه بد نیست بدانید که پروژه های STEP7 به دو روش مختلف قابل ایجاد می باشند :



اگر در حال ایجاد یک برنامه شامل تعداد زیادی ورودی و خروجی می باشید ، توصیه می شود که نخست به پیکربندی سخت افزار بپردازید. فایده این روش آنست که STEP7 آدرس های قابل استفاده را در ویرایشگر پیکربندی سخت افزاری نمایش میدهد.

اگر روش دوم را برگزینید مجبورید که آدرس ها را بر حسب عناصر انتخاب شده به دلخواه خود برگزینید و قادر به فراخوانی این آدرس ها از طریق STEP7 نمی باشید.

در پیکربندی سخت افرازی ، نه تنها قادر به تعریف آدرس ها میباشید بلکه می توانید پارامترها و مشخصات کارت ها را نیز تغییر دهید.

چون ما قصد داریم تعداد کمی ورودی و خروجی را مورد استفاده قرار دهیم فعلا از پیکربندی سخت افزاری صرف نظر نموده و از برنامه نویسی شروع می نماییم.

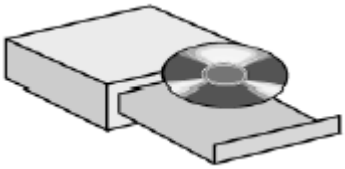
1-4 نصب STEP7

صرفنظر از اینکه بخواهیم از برنامه نویسی یا پیکربندی سخت افزاری آغاز نماییم نخست مجبوریم که STEP7 را نصب نماییم.

اگر از یک ابزار برنامه نویسی SIMATIC (PG) استفاده می نمایید STEP7 نصب شده در اختیار می باشد درصورتیکه از PC استفاده مینمائید میبایست نرم افزار STEP7 را بر روی آن نصب نمائید. هنگامیکه قصد دارید نرم افزار STEP7 را برای اولین بار بر روی یک PC نصب نمایید نخست حداقل پیش نیازهای نرم افزاری و سخت افزاری مورد نیاز را

بررسی نمایید. پیش نیازهای مورد نیاز را میتوانید بر روی CD ، Step7 و در مسیر <Drive>:/STEP7/DISK1 و در داخل فایل Readme.Wri پیدا نمایید.

اگر قصد نصب نرم افزار STEP7 را دارید نخست دیسک STEP7 را در داخل CD-ROM وارد نمائید. برنامه نصب بطور خودکار شروع می گردد ، دستورالعمل نمایش داده شده بر روی مونیتور را تعقیب نمایید تا نرم افزار بطور کامل نصب گردد.



■ اگر برنامه نصب بطور خودکار شروع نمی گردد می توانید برنامه نصب نرم افزار را در مسیر <Drive>:/Step7/Disk1/Setup.exe پیدا نمایید.

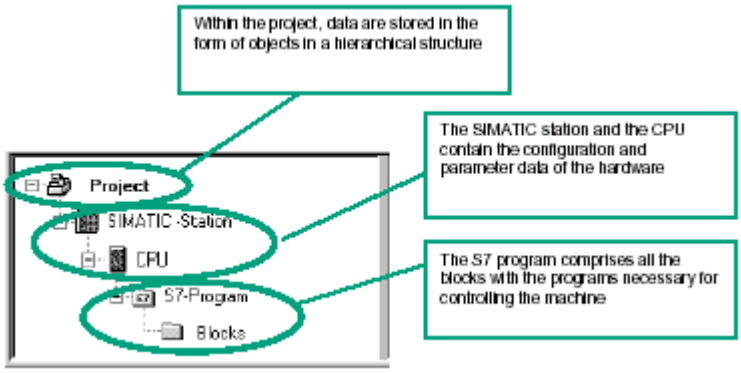
هنگامیکه نصب نرم افزار کامل گردید میبایست کامپیوتر را مجددا راه اندازی نموده آنگاه آیکون " SIMATIC MANAGER" بر روی Desktop محیط ویندوز پدیدار می گردد.

بـا دوبار کلیک نمودن بر روی آیکون "SIMATIC MANAGER" ، Step7Wizard بطور خودکار شروع بکار می نماید.

شـــما مـــی توانـــید بـــا رجـــوع بـــه فـــایل Readme.Wri نکـــات اضـــافی دیگـــری راجـــع بـــه نصـــب نرم افزار STEP7 را بیابید.

2-1 شروع SIMATIC MANAGER و ایجاد یک پروژه

SIMATIC MANAGER پنجره مرکزی ای است که با شروع STEP7 فعال می گردد. تنظیمات پیش گزیده STEP7 بصورتی است که در ابتدای اجرا، STEP7 wizard ، که با استفاده از آن میتوانید یک پروژه STEP7 ایجاد نمایید آغاز می گردد. ساختار پروژه بترتیب برای ذخیره و مرتب سازی کلیه اطلاعات و برنامه ها مورد استفاده قرار می گیرد.



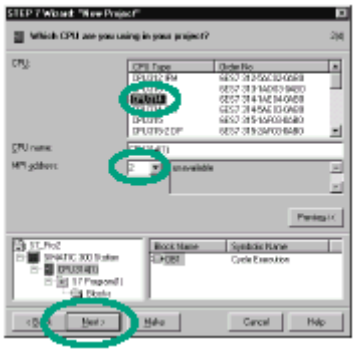
بر روی آیکون SIMATIC Manager دوبار کلیک نموده تا STEP7 Wizardفعال شود.

با استفاده از Preview می توانید نمایش ساختار پروژه در پایین پنجره را فعال یا غیر فعال نمایید.

برای رفتن به کادر مکالمه بعدی بر روی Next کلیک نمایید.

برای پروژه نمونه "Getting Started" ، CPU314 را انتخاب نمایید. مثال بروشی ایجاد گشته که شما در هر لحظه می توانید CPU ای که در اختیار دارید را جایگزین CPU انتخاب شده نمایید.

آدرس پیش گزیده MPI، 2 می باشد.



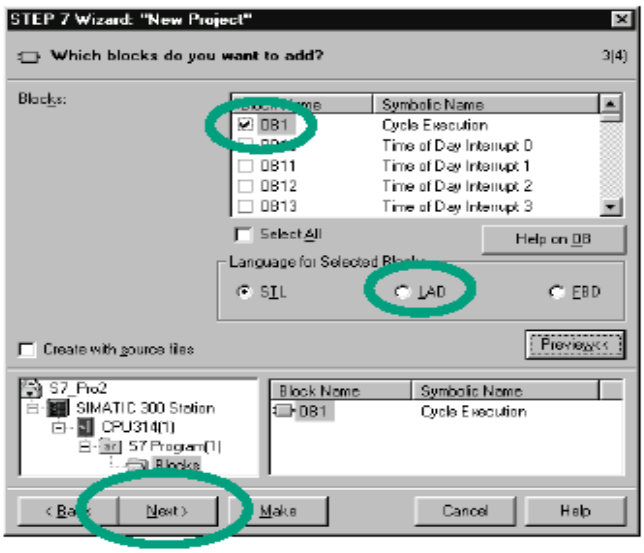
بر روی Next کلیک نموده تا تنظیمات انجام شده را تائید نموده و به کادر مکالمه بعدی بروید.

■ هر CPU مشخصات خاص خودش را دارد، بعنوان مثال از نظر پیکربندی حافظه یا نواحی آدرس دهی. بدین دلیل لازم است قبل از شروع به برنامه نویسی CPU را انتخاب نمود.

CPU برای برقراری ارتباط با دستگاه برنامه نویسی (PG)یا PC لازم است آدرس MPI داشته باشد.

بلوک سازماندهی OB1 را انتخاب نمایید (در صورتی که انتخاب نگشته باشد)

یکی از زبانهای برنامه نویسی را انتخاب نمایید: منطق نردبانی (LAD) ، نمایش عبارتی (STL) یا دیاگرام بلوکی(FBD) .

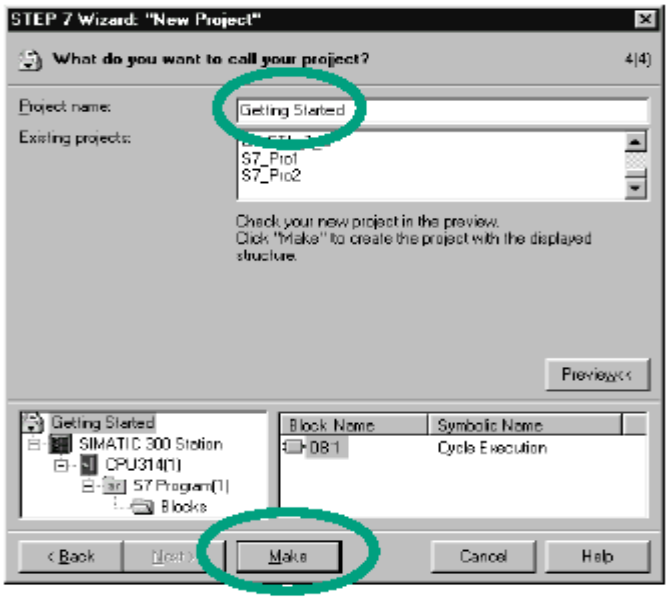


انتخاب های انجام شده را با کلیک کردن بر روی Next تائید نمایید.

■ بلـوک OB1 بالاترین سطح برنامه نویسی محسوب می گردد و سایر بلوک ها را در برنامه S7 سازماندهی میکند. شما می توانید زبان برنامه نویسی را مجدداً در مراحل بعدی تغییر دهید.

در فضـــای"Project Name" و بـــر روی نـــام پیشـــنهادی دوبـــار کلـــیک نمایـــید و آنـــرا بـــا "Getting Started"جایگزین نمایید.

بر روی Make کلیک نمایید تا پروژه تازه تان را بر اساس پیش نمایش (preview) ایجاد نمایید.



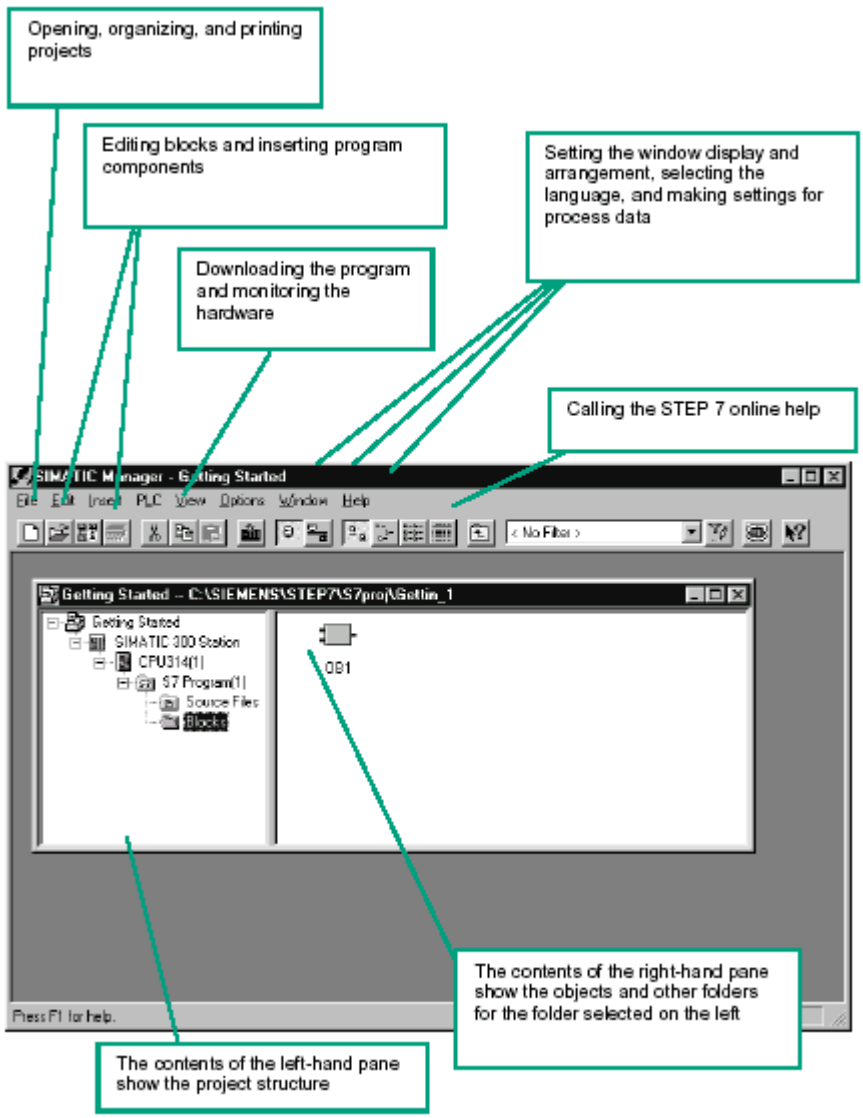
نکـته: هنگامـیکه بـر روی دگمـه Make کلـیک مـی نمایـید، SIMATIC Manager پـنجره ای را جهـت پـروژه "Getting Started" که ایجاد نموده اید باز می کند.

در صـفحات بعـدی نشـان خواهـیم داد فـایل هـا و پوشـه هایـی کـه ایجاد کردیم چه هستند و چگونه می توانید بطور موثـری بـا آنهـا کـار کنـید. هر بار که برنامه را اجرا می نمایید STEP7 Wizard فعال می گردد. شما می توانید این انـتخاب پـیش گـزیده را در شروع برنامه غیر فعال نمایید. اما اگر بدون استفاده از STEP7 Wizard پروژه ای ایجاد نمایید مجبور هستید تمام شاخه های داخل پروژه را خودتان ایجاد نمایید.

راهنمایي: بـرای اطلاعـات بیشـتر می توانید به موضوع "Setting up & editing the project" واقع در Help>Contents مراجعه نمایید.

2.2 ساختار پروژه در SIMATIC Manager و نحوه فراخوانی راهنما

بـه محـض بسـتن SIMATIC Manager , STEP7 Wizard بـا پنجره ای تحت عنوان"Getting Started" پدیدار می گردد. از اینجا شما می توانید کار با کلیه توابع و پنجره های STEP7 را آغاز نمایید.

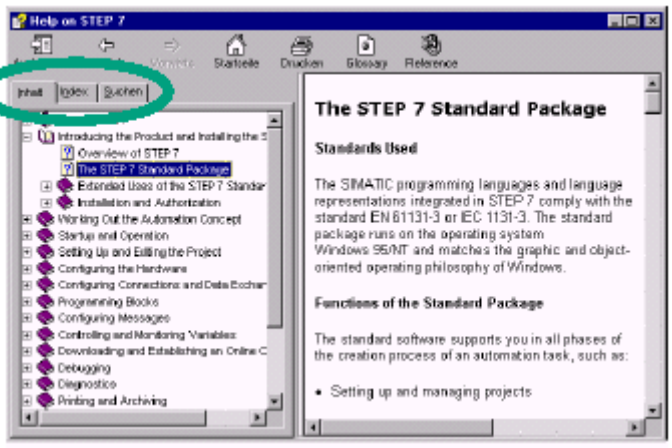


استفاده از راهنمای **STEP7**

روش اول) مکان نما را بر روی هر منوی فرمان قرار داده و کلید **F1** را فشار دهید. راهنمای حساس به وارد نمودن متن برای منوی فرمان انتخاب شده پدیدار می گردد.

F1

روش دوم) در داخل منوی **Help** بر روی **Contents** کلیک نمایید.



صفحه **Contents** با موضوعات راهنمای متعددی، در نیمه چپ صفحه پدیدار شده و موضوع انتخاب شده در نیمه راست صفحه نمایش داده می شود.

برای رسیدن به موضوعی که مورد نظر میباشد می توانید بر روی علامت + در لیست **Contents** کلیک نمایید. در این صورت محتویات موضوع انتخاب شده در نیمه راست صفحه بنمایش در می آید.

با استفاده از **Index** و **Search** می توانید رشته هایی از حروف را وارد کرده و موضوعات مورد نظری که بدنبالش هستید را مشاهده نمایید.

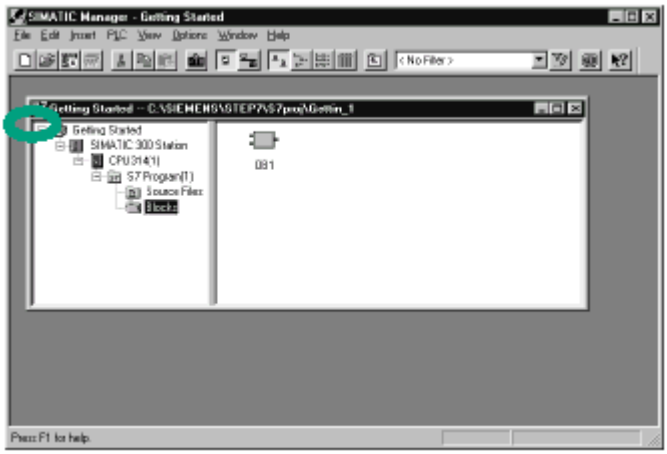
روش سوم) برروی دگمه علامت سوال واقع در خط ابزار (Tool bar) کلیک نمایید تا موس به یک مکان نمای راهنما تبدیل شود. حال در صورت کلیک کردن بر روی هر عنصری خاصی، راهنما فعال

می گردد.

حرکت در ساختار پروژه

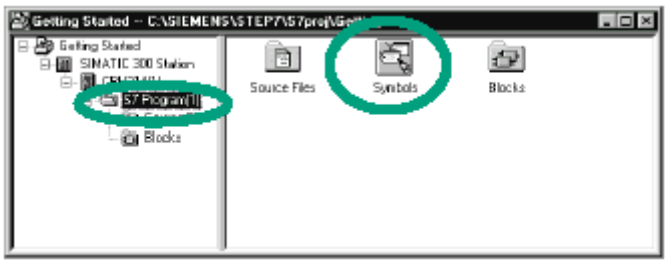
پروژه ای که ایجاد کرده اید بصورت یک ایستگاه **S7 (station)** و **CPU** انتخاب شده بنمایش در می آید.

بر روی علامت های + یا – کلیک نمایید تا پوشه ها را باز کرده یا ببندید.



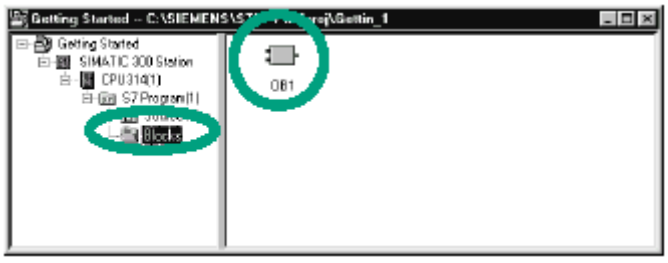
از این مرحله به بعد می توانید با کلیک کردن بر علائمی که در نیمه راست صفحه بنمایش در می آید کارهای دیگری را انجام دهید.

روی پوشه **S7 Program** کلیک نمایید. این پوشه کلیه ملزومات برنامه را در بر می گیرد.



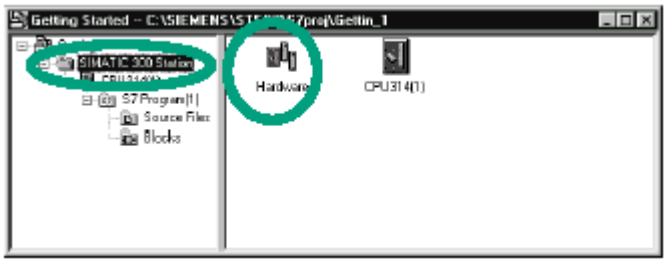
شما عنصر **Symbols** را در فصل 3 جهت نسبت دادن اسامی سمبولیک به آدرسهایتان مورد استفاده قرار خواهید داد. عنصر **Source Files** برای ذخیره برنامه فایلهای مرجع **(source files)** استفاده می شود ولی این موضوع در این کتاب مطرح نمی گردد.

بر روی پوشه **Blocks** کلیک نمایید این پوشه در حال حاضرتنها حاوی **OB1** می باشد ولی بعدا کلیه بلوک های دیگر را نیز شامل خواهد گشت.



در فصول 4 و 5 شما شروع به برنامه نویسی بزبانهای منطق نردبانی، نمایش عبارتی یا دیاگرام بلوکی خواهید نمود.

بـر روی پوشه **SIMATIC 300 Station** کلیک نمایید. کلیه اطلاعات مربوط به مسائل سخت افزاری پروژه در اینجا ذخیره شده است.



در فصل 6 برای تعریف پارامترهای **PLC** تان از عنصر **Hardware** استفاده خواهید نمود.

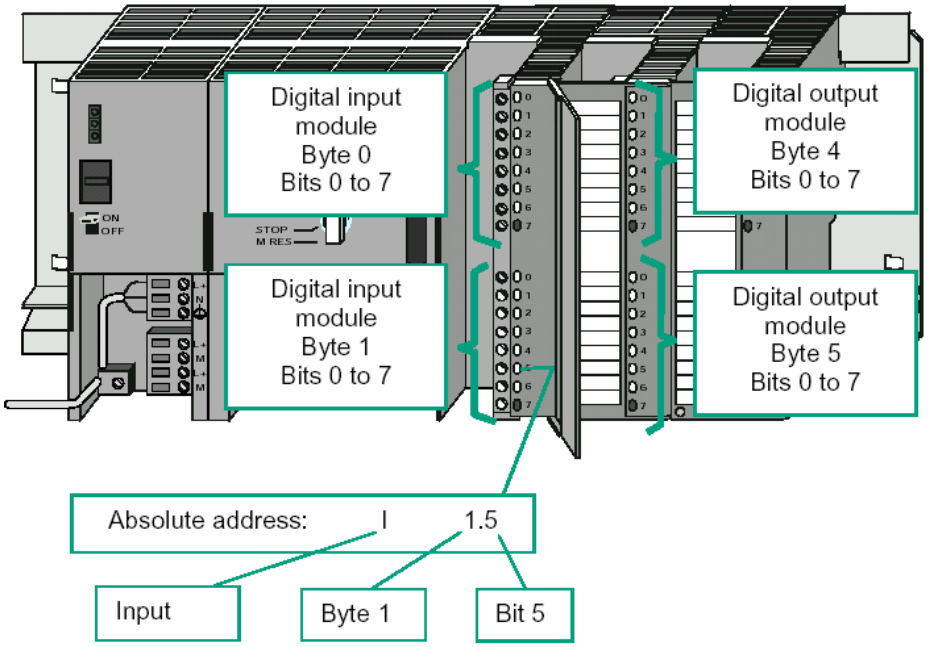
راهنمایی: اگـر جهـت انجـام کارهـای اتوماسـیون نـیاز بـه نـرم افـزارهای دیگـر **SIMATIC** مانـند بسته اختیاری **PLCSIM** (بـرنامه شـبیه سـاز سخت افزار) یا **S7Graph** (زبان برنامه نویسی تصویری) می باشد نیز اینها در **STEP7** اضـافه شـده اند. بعنوان مثال با استفاده از **SIMATIC Manager** میتوانید مستقیماً عناصر مربوطه مثل یک بلوک تابعی **S 7 Graph** را مورد استفاده قرار دهید.

بـــرای اطلاعـــات بیشـــتر میتوانـــید بـــه موضـــوعات "**Working Out the Automation**"و "**Basics of Designing the Program Structure** " واقـع در **Help > Contents** مـراجعه نمایـید. اطلاعـات بیشـتر در ایـن رابطـه تحـت عـنوان"**Component for Completely Integrated Automation**" در نـرم افزار اختیاری"**SIMATIC Catalog ST 70**" قابل دستیابی می باشد.

3. برنامه نویسی با استفاده از سمبل ها

3.1 آدرس دهی مطلق

هـر ورودی و خروجـی یـک آدرس مطلـق دارد کـه هـنگام پیکربـندی سـخت افـزاری ایجـاد مـی گـردد. از آنجایـیکه ایـن آدرس بطـور مستقیم تعریف میگردد لذا آدرس دهی مطلق نامیده شده است.



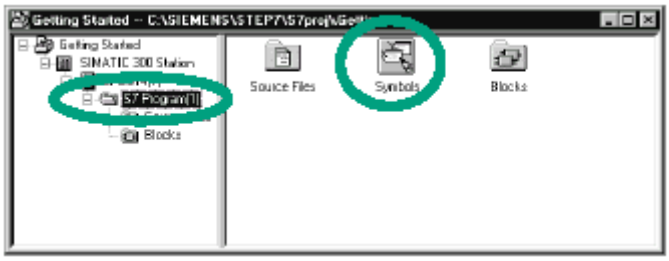
آدرس مطلق می تواند با هرنام سمبلیکی که شما بر می گزینید جایگزین شود.

■ اگـر در بـرنامه **S7** خویـش قصـد اسـتفاده از مقادیر متعدد ورودی و خروجی را ندارید صرفاً میبایست از برنامه نویسی مطلق استفاده نمایید.

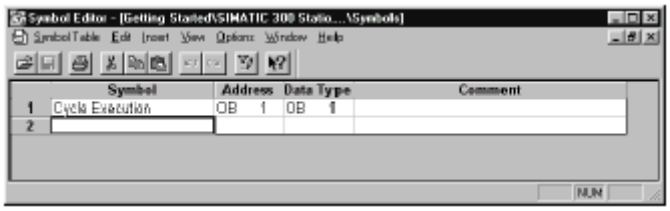
3.2 برنامه نویسی سمبلیک

در جـدول سـمبل هـا بـه تمامـی آدرس هـای مطلقـی کـه بعداً در برنامه استفاده می شوند یک اسم سـمبلیک بهمراه نوع دیتا **(data)** اطلاق خواهد شد بعنوان مثال به ورودی **I0.1** نام سمبلیک **Key1** را میدهیم. این اسامی در تمام قسمت های برنامه اعمال شده و بعنوان مقادیر سراسری **(Global data)** شـناخته می شوند. با استفاده از برنامه نویسی توسط سمبل ها ، برنامه **S7** نوشته شده توسط شما بمیزان زیادی قابل فهم و روان می گردد.

آنقدر پوشه های تو در توی پنجره "Getting Started" را باز نمائید تا به **S7 Program (1)** رسیده و سپس با دوبار کلیک نمودن بر روی **Symbols** آنرا باز نمایید.



جدول سمبل های شما در حال حاضر تنها شامل بلوک **OB1** که قبلاً تعریف شده است میباشد.



بر روی **Cycle Execution** کلیک نموده و عبارت "**Main Program**" را وارد نمایید.

	Symbol	Address	Data Type
1	Cycle Execution	OB 1	OB 1
2			

در سطر دوم "**Green_Light**" و "**Q4.0**" را اضافه نمایید. نوع اطلاعات بصورت اتوماتیک وارد خواهد شد.

	Symbol	Address	Data Type
1	Main Program	OB 1	OB 1
2	Green Light	Q 4.0	BOOL

در سطر اول یا دوم بر روی ستون **Comment** کلیک نمایید تا توضیحی را در رابطه با سمبل مربوطه وارد نمایید. با فشار دادن کلید **Enter** وارد کردن اطلاعات در سطر مربوطه به انجام رسیده و سپس یک سطر جدید ایجاد می گردد.

Comment

در سطر سوم **"Red_Light"**و **"Q4.1"** را وارد نمایید و سپس کلید **Enter** را فشار دهید.

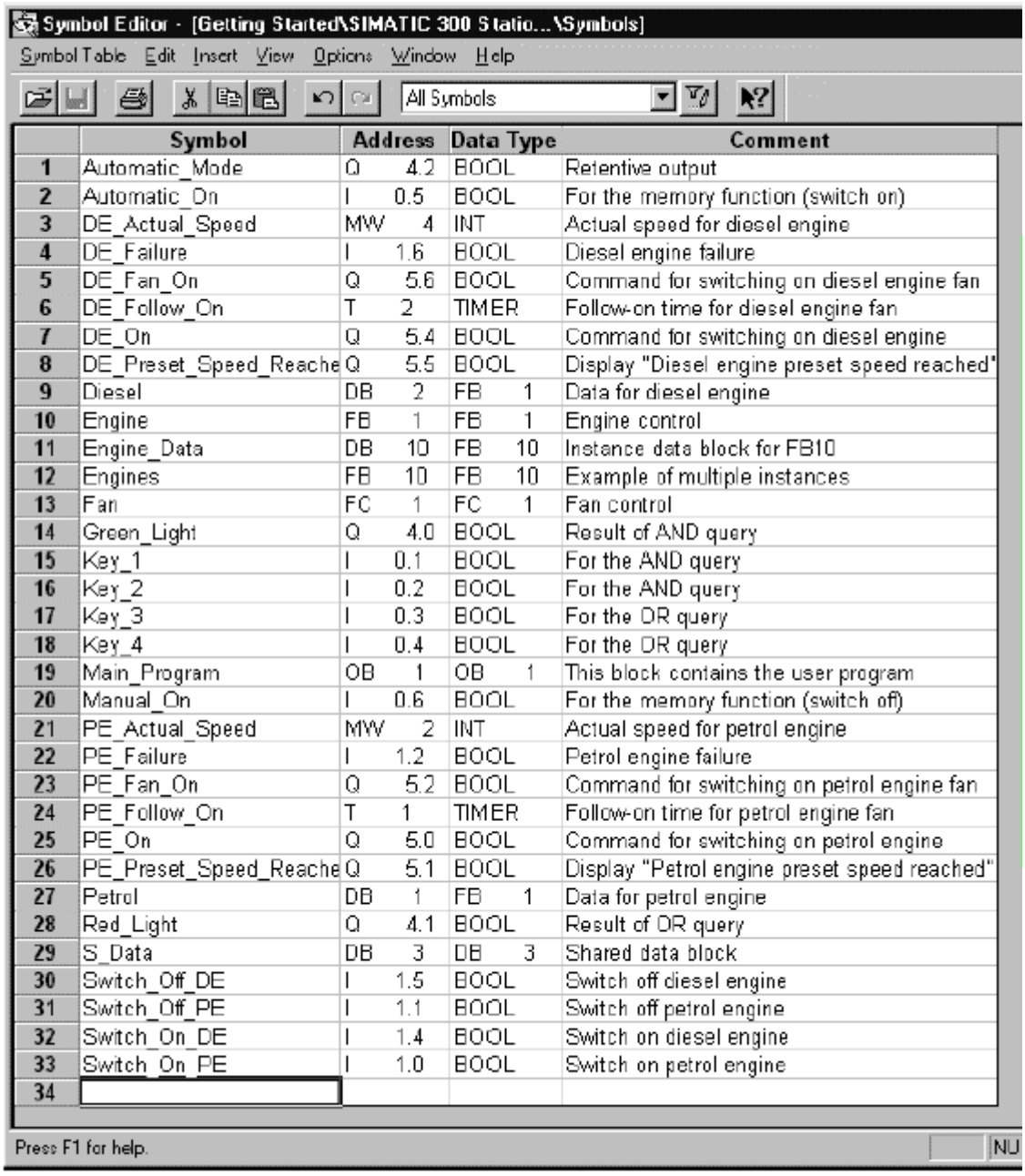
	Symbol	Address	Data Type
1	Main Program	OB 1	OB 1
2	Green Light	Q 4.0	BOOL
3	Red Light	Q 4.1	BOOL

■ بدین روش شما می توانید به تمام آدرس های مطلق ورودی ها و خروجی هایی که قرار است در برنامه مورد استفاده قرار دهید یک اسم سمبلیک نسبت دهید.

کلیه اطلاعات وارد شده و یا تغییراتی که در جدول سمبل ها اعمال نموده اید را ذخیره نموده و

پنجره را ببندید.

از آنجائیکه در سراسر پروژه **"Getting Started"** نامهای زیادی مورد استفاده قرار گرفته اند شما می توانید در بخش 4.1 جدول سمبل ها را به پروژه **"Getting Started"** خویش کپی نمایید.



	Symbol	Address	Data Type	Comment
1	Automatic_Mode	Q 4.2	BOOL	Retentive output
2	Automatic_On	I 0.5	BOOL	For the memory function (switch on)
3	DE_Actual_Speed	MW 4	INT	Actual speed for diesel engine
4	DE_Failure	I 1.6	BOOL	Diesel engine failure
5	DE_Fan_On	Q 5.6	BOOL	Command for switching on diesel engine fan
6	DE_Follow_On	T 2	TIMER	Follow-on time for diesel engine fan
7	DE_On	Q 5.4	BOOL	Command for switching on diesel engine
8	DE_Preset_Speed_Reache	Q 5.5	BOOL	Display "Diesel engine preset speed reached"
9	Diesel	DB 2	FB 1	Data for diesel engine
10	Engine	FB 1	FB 1	Engine control
11	Engine_Data	DB 10	FB 10	Instance data block for FB10
12	Engines	FB 10	FB 10	Example of multiple instances
13	Fan	FC 1	FC 1	Fan control
14	Green_Light	Q 4.0	BOOL	Result of AND query
15	Key_1	I 0.1	BOOL	For the AND query
16	Key_2	I 0.2	BOOL	For the AND query
17	Key_3	I 0.3	BOOL	For the OR query
18	Key_4	I 0.4	BOOL	For the OR query
19	Main_Program	OB 1	OB 1	This block contains the user program
20	Manual_On	I 0.6	BOOL	For the memory function (switch off)
21	PE_Actual_Speed	MW 2	INT	Actual speed for petrol engine
22	PE_Failure	I 1.2	BOOL	Petrol engine failure
23	PE_Fan_On	Q 5.2	BOOL	Command for switching on petrol engine fan
24	PE_Follow_On	T 1	TIMER	Follow-on time for petrol engine fan
25	PE_On	Q 5.0	BOOL	Command for switching on petrol engine
26	PE_Preset_Speed_Reache	Q 5.1	BOOL	Display "Petrol engine preset speed reached"
27	Petrol	DB 1	FB 1	Data for petrol engine
28	Red_Light	Q 4.1	BOOL	Result of OR query
29	S_Data	DB 3	DB 3	Shared data block
30	Switch_Off_DE	I 1.5	BOOL	Switch off diesel engine
31	Switch_Off_PE	I 1.1	BOOL	Switch off petrol engine
32	Switch_On_DE	I 1.4	BOOL	Switch on diesel engine
33	Switch_On_PE	I 1.0	BOOL	Switch on petrol engine
34				

■ در اینجا شما می توانید جدول سمبل های برنامه **S 7** با نام "**Getting Started**" را که بزبان نمایش عبارتی نوشته شده ملاحظه نمایید.

بد نیست بدانید که صرفنظر از زبانی که در نوشتن برنامه انتخاب نمایید تنها یک جدول سمبل ها برای برنامه **S7** تان ایجاد می گردد.

در جدول سمبل ها ، کلیه کاراکترهای قابل چاپ (مانند کاراکترهای خاص ، فاصله ، **....**) قابل استفاده می باشند.

نکته: نوع دیتایی که قبلاً بطور اتومات به جدول سمبل اضافه شد، نوع سیگنالی که میبایست برای **CPU** مورد تجزیه و تحلیل قرار بگیرد را مشخص می کند.

در **STEP7** انواع دیتاهای مورد استفاده بشرح ذیل می باشند.

BOOL **BYTE** **WORD** **DWORD**	این دیتاها بر پایه بیت ها ساخته شده اند، از 1 بیت **(BOOL)** تا 32 بیت **(DWORD)**
CHAR	این نوع از دیتاها دقیقا یک حرف از مجموعه حروف **ASCII** را اشغال میکنند.
INT **DINT** **REAL**	جهت پردازش مقادیر عددی (بعنوان مثال عبارات محاسباتی) مورد استفاده قرار میگیرند.
S5TIME **TIME** **DATE** **TIME_OF_DAY**	جهت نمایش فرمتهای مختلف زمان و تاریخ در داخل **STEP7**. (مثلا برای تنظیم تاریخ جهت وارد نمودن مقدار زمانی یک تایمر)

برای اطلاعات بیشتر می توانید به موضوعات**"Programming Blocks"** و **"Defining Symbols"** واقع در **Help>Contents** مراجعه نمایید.

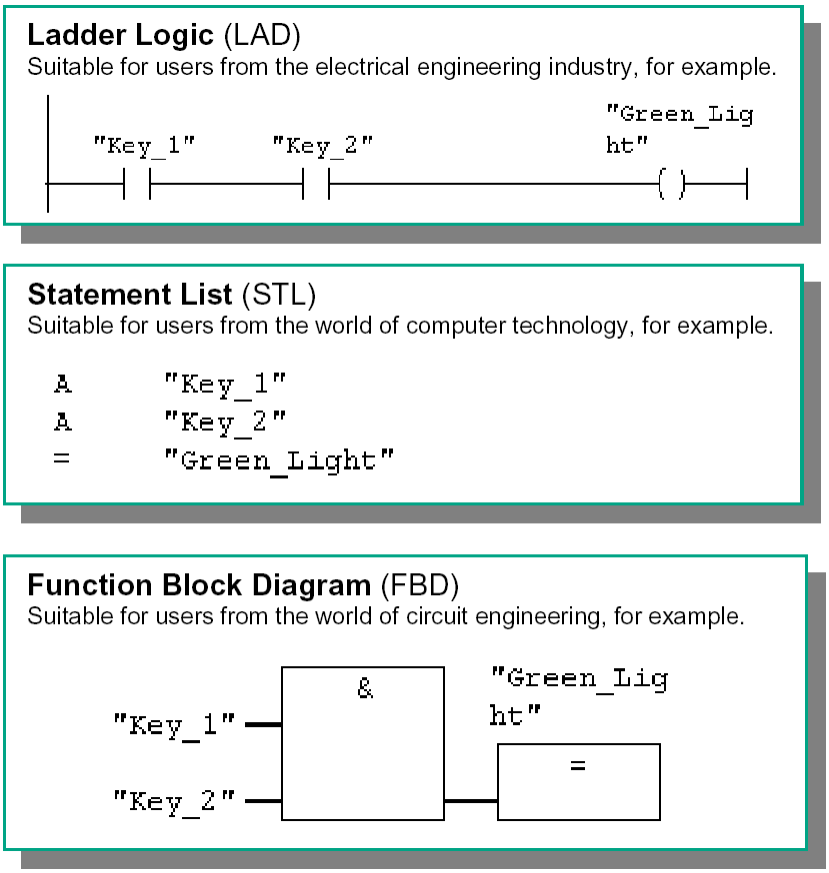
4. ایجاد برنامه در OB1

4.1 باز کردن پنجره برنامه LAD/STL/FBD

انتخاب منطق نردبانی ،نمایش عبارتی یا دیاگرام بلوکی

بـا اسـتفاده از **STEP7** شما می توانید برنامه های **S7** به زبانهای استاندارد منطق نردبانی **(LAD)** ، نمایش عبارتی **(STL)** یا دیاگرام بلوکی **(FBD)** ایجاد نمایید.

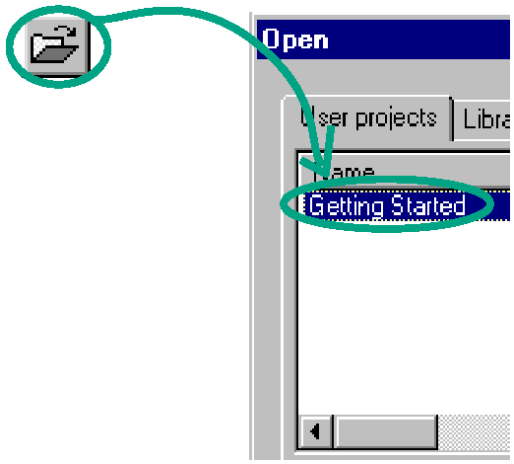
در این فصل شما باید تصمیم بگیرید که کدام زبان را قصد دارید استفاده نمایید



■ بلـوک **OB1** بـه همـان زبانی باز خواهد شد که شما در **Project Wizard** ایجاد کرده بودید. ولی این امکان برای شما وجود دارد که هر زمان بخواهید زبان برنامه نویسی پیش گزیده تان را تغییر دهید.

کپی کردن جدول سمبل ها و باز کردن OB1

در صـورت لزوم پروژه **"Getting Started"** را باز نمایید. بدین منظور روی کلید **Open** در خط ابزار کلـیک نمایـید، پـروژه **"Getting Started"** را کـه ایجاد نموده اید را انتخاب نموده و توسط **OK** تائید نمایید.



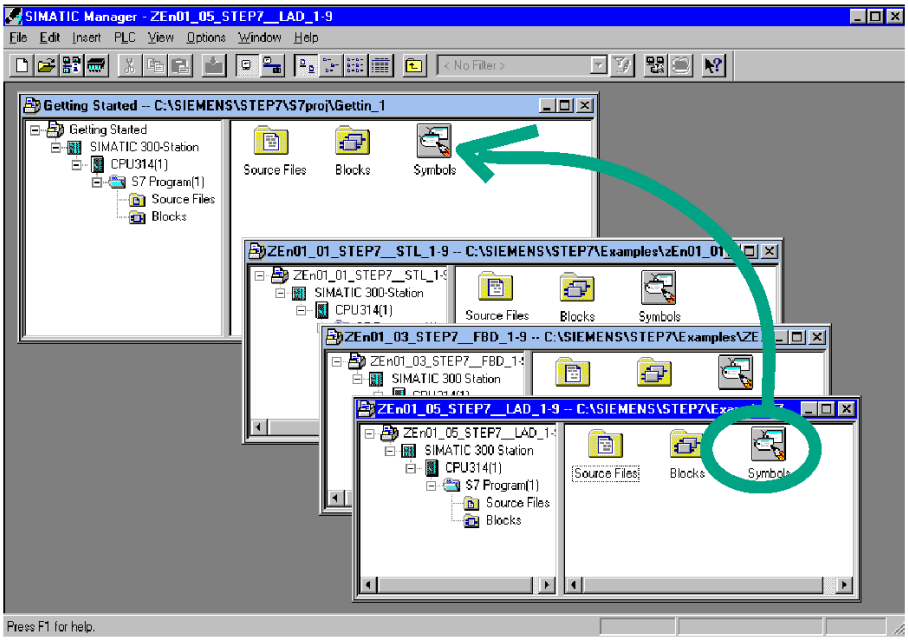
بسته به زبان برنامه نویسی ای که قصد استفاده از آن را دارید یکی از سه پروژه زیر را باز نمایید :

zEn01_06_STEP7_LAD_1-9

zEn01_02_STEP7_STL_1-9

zEn01_04_STEP7_FBD_1-9

اینجا می توانید هر سه نمایش پروژه را ملاحظه نمایید.



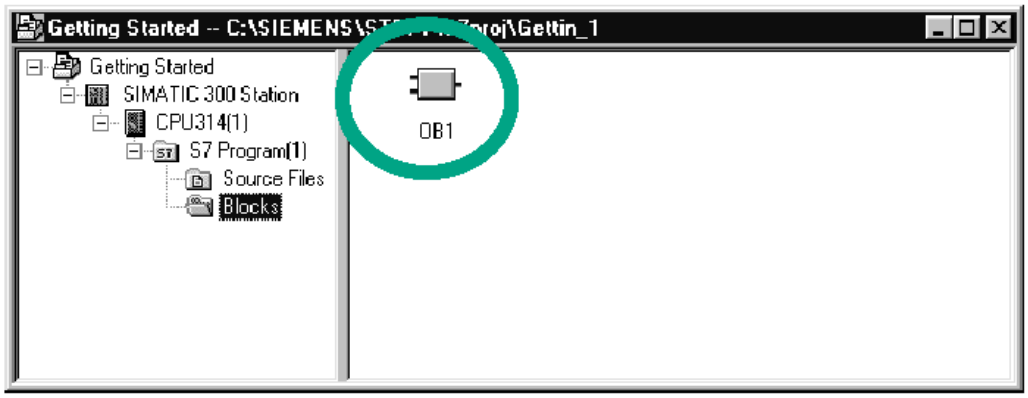
4

مسیرهای تو در توی "**zEn01_XXX**" را آنقدر باز نمایید که علامت **Symbols** را مشاهده نموده و با استفاده از **Drag and Drop** آنرا در پوشه **S7 Program** در پنجره پروژه "**Getting Started**" کپی نمایید.

سپس پنجره "**zEn01_XXX** " را ببندید.

◙ **Drag and Drop** بـه معنـی آنسـت کـه بـر روی آیکـون مـورد نظـرتان کلـیک کـرده و در حالـیکه کلید موس را نگاهداشته آنرا حرکت دهید. با رها کردن کلید موس آیکون مذکور در مکان مورد نظر پدیدار می گردد.

درداخـل پروژه "**Getting Started**" بر روی **OB1** دوبار کلیک نمایید. پنجره نمایش **LAD/STL/FBD** باز می گردد.



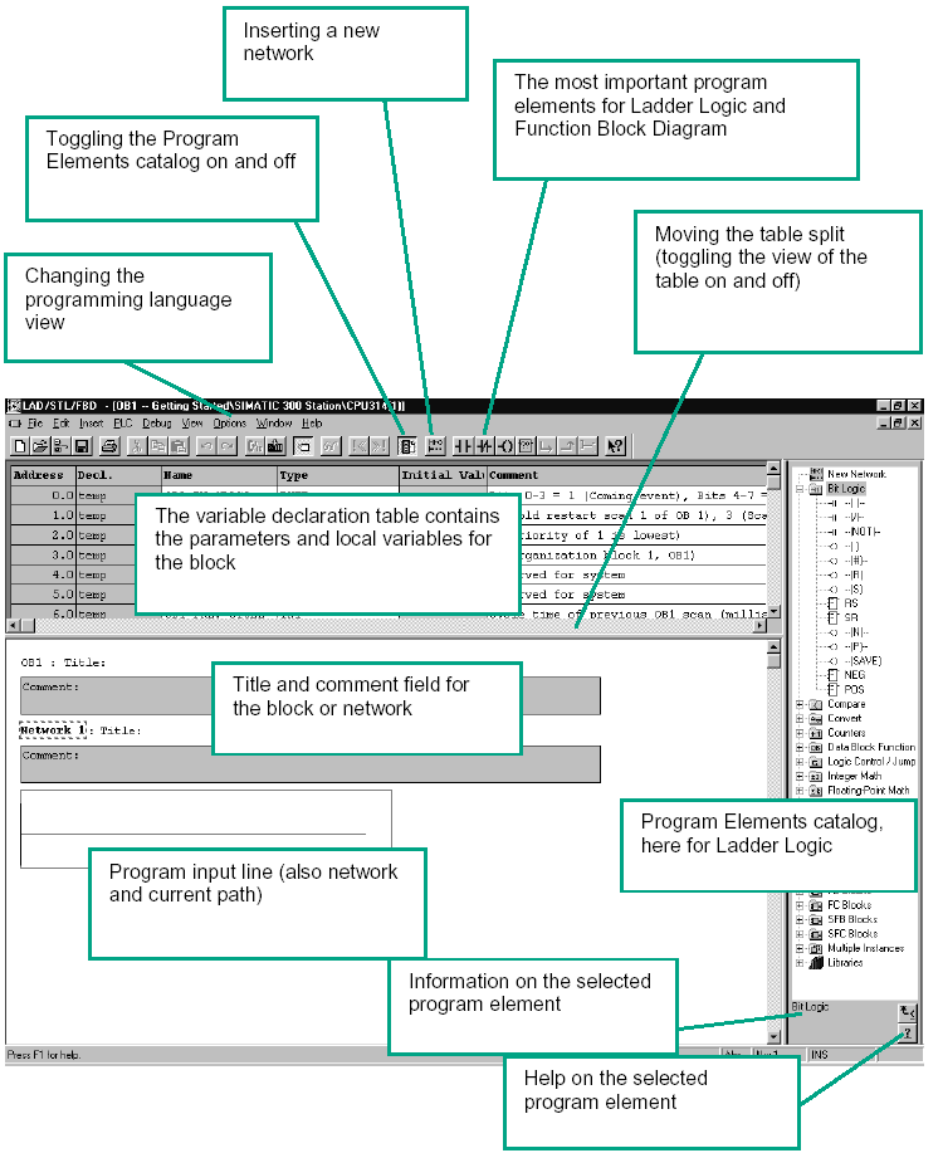
در **STEP7** ، **OB1** دائمـا و بطـور متوالـی توسـط **CPU** در حـال پـردازش مـی باشـد. **CPU** خط به خط **OB1** را خوانـده و دستورات برنامه را اجراء می نماید. هنگامیکه **CPU** به اولین خط برنامه باز می گردد در واقع یک سیکل را بطور کامل انجام داده است. مدت زمانی که برای این کار نیاز است **Cycle Time** نامیده می شود.

بسـته بـه زبانـی که برای برنامه نویسی انتخاب می کنید می توانید برای برنامه نویسی بزبان منطق نردبانی به بخش 4.2 ، بـرای بـرنامه نویسـی بـزبان نمـایش عبارتـی بـه بخـش 4.3 یا برای برنامه نویسی بزبان **Function Block Diagram** به بخش 4.4 رجوع نمایید.

◙ بـــرای اطلاعـــات بیشـــتر مـــی توانـــید بـــه مباحـــث **Programming Block** ، **Creating Blocks and Libraries Libraries** واقع در **Help > Contents** مراجعه نمایید.

پنجره برنامه LAD/STL/FBD

تمـام بلـوک هـا در پـنجره برنامه **LAD/STL/FBD** نوشته می شوند. در اینجا می توانید فرم نمایش برای منطق نردبانی را مشاهده نمایید.

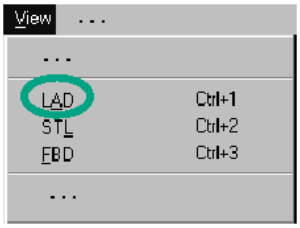


4.2 برنامه نویسی OB1 بزبان منطق نردبانی

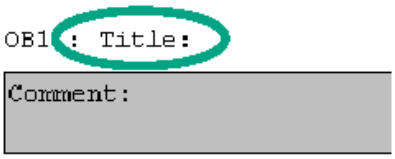
در بخش جاری شما ، یک مدار سری ، یک مدار موازی و تابع حافظه ای ست/ری ست را بزبان منطق نردبانی **(LAD)** برنامه نویسی خواهید نمود.

برنامه نویسی یک مدار سری بزبان منطق نردبانی

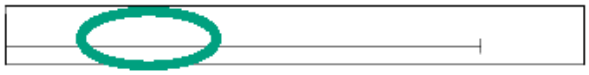
در صورت لزوم در منوی **View** زبان برنامه نویسی را به **LAD** تغییر دهید.



در فضـای **Title** بلوک **OB1** کلیک نموده و عبارت **Cyclically Processed main Program** را بعنوان مثال وارد نمایید.



برای وارد نمودن اولین عنصر مسیر جاری را انتخاب نمایید.



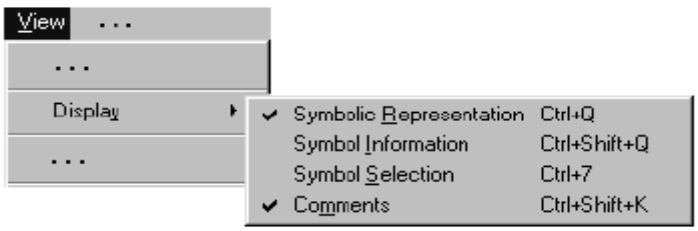
با کلیک نمودن بر روی **Button** مربوطه در منوی ابزار یک کنتاکت حالت عادی باز وارد نمایید.

بروش مشابه یک کنتاکت حالت عادی باز دیگر وارد نمایید.

در انتهای سمت راست مسیر جاری یک سیم پیچ ایجاد نمایید.

آدرس های کنتاکت های حالت عادی باز و سیم پیچ درمدار سری ایجاد شده هنوز وارد نشده اند.

بررسی نمایید که نمایش سمبلیک انتخاب شده باشد.



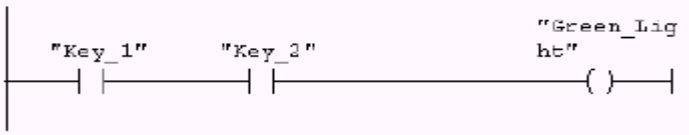
بر روی علامت **??.?** کلیک نموده و نام سمبلیک **Key – 1** (در داخل گیومه) راوارد نمایید.

توسط **Enter** تایید نمایید.

نام سمبلیک **Key – 2** را برای دومین کنتاکت حالت عادی باز وارد نمایید.

نام **Green – Light** را برای سیم پیچ وارد نمایید.

حالا شما یک مدار سری کامل را ایجاد نموده اید.

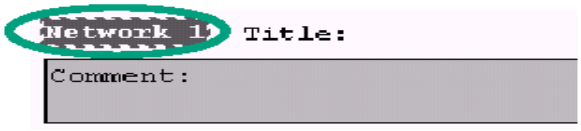


اگر دیگر هیچ سمبلی برنگ قرمز وجود ندارد بلوک را ذخیره نمایید.

■ سمبل هایکه برنگ قرمز نمایش داده می شوند یا در جدول سمبل ها وجود ندارد یا یک خطای **Syntax** وجود دارد. شما می توانید بطور مستقیم نام سمبلیک را از جدول سمبل ها نیز وارد نمایید. بر روی علامت **??.?** کلیک نموده و سپس در داخل منو فرمان **Insert > Symbol** را اجرا نمایید. در لیست **Pull down** نام سمبلیک بطور خودکار اضافه خواهد شد.

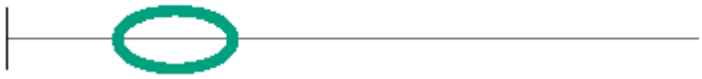
برنامه نویسی یک مدار موازی بزبان منطق نردبانی

Network1 را انتخاب نمایید.



یک **Network** جدید وارد نمایید.

مجدداً مسیر جاری را انتخاب نمایید.



یک کنتاکت حالت عادی باز و یک سیم پیچ وارد نمایید.

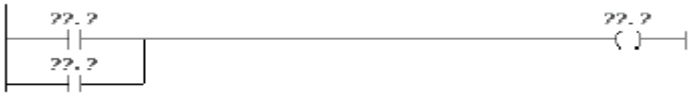
خط عمودی مسیر جاری را انتخاب نمایید.

یک شاخه موازی وارد نمایید.

یک کنتاکت حالت عادی باز دیگر را در شاخه موازی اضافه نمایید.

شاخه را ببندید (در صورت لزوم فلش پایین را انتخاب نمایید)

آدرس ها هنوز در داخل مدار موازی وارد نشده اند.



برای اعمال آدرس های سمبلیک به همان روشی که برای مدار سری انجام دادید عمل نمایید.

کنتاکت حالت عادی باز بالایی را با **Key–3** ، کنتاکت حالت عادی باز پایین را با **Key–4** و سیم پیچ را با **Red–Light** نامگذاری نمایید.

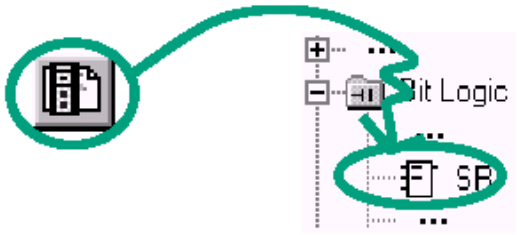
بلوک را ذخیره نمایید.

برنامه نویسی یک تابع حافظه بزبان منطق نردبانی

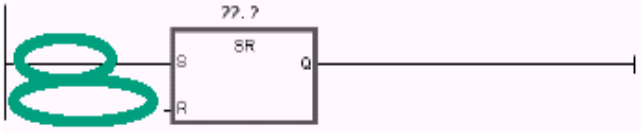
Network2 را انتخاب نموده و **Network** دیگری وارد نمایید.

مجدداً مسیر جاری را انتخاب نمایید.

در کاتالوگ عناصر برنامه زیر **Bit Logic** آنقدر **Navigate** نمایید تا به عنصر **SR** برسید. دوبار کلیک نمایید تا عنصر را وارد نمایید.



یک کنتاکت حالت عادی باز را در جلوی هر کدام از ورودی های **S** و **R** وارد نمایید.

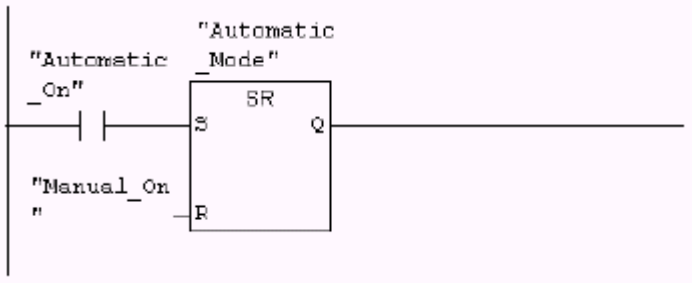


برای عنصر **SR** اسامی سمبلیک زیر راوارد نمایید :

کنتاکت بالایی **" Automatic–On "**

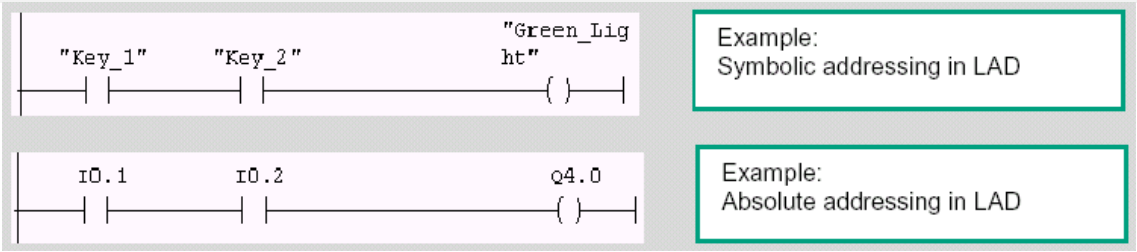
کنتاکت پایینی **Manual–On**

عنصر **SR Automatic–Mode**



بلوک را ذخیره نموده و پنجره را ببندید.

اگر بخواهید میتوانید بین آدرس دهی سمبلیک و مطلق را با غیر فعال نمودن فرمان **View>Display>Symbolic Representation** مشاهده نمایید.



شما می توانید فرمت خط را برای آدرس دهی سمبلیک در پنجره برنامه **LAD/STL/FBD** با استفاده از منو فرمان **Options >Customize** و سپس انتخاب **Width of address field** در **LAD/FBD tab** تغییر دهید.

در اینجا شما می توانید فرمت خط را بین 10 تا 24 حرف تنظیم نمایید.

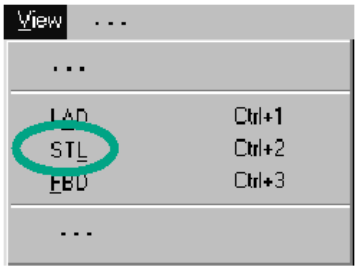
برای اطلاعات بیشتر می توانید به مباحث **Creating Logic Blocks, Programing Block** و **Editing Ladder Instructions** واقع در **Help > Contents** مراجعه نمایید.

4.3 برنامه نویسی OB1 بزبان نمایش عبارتی

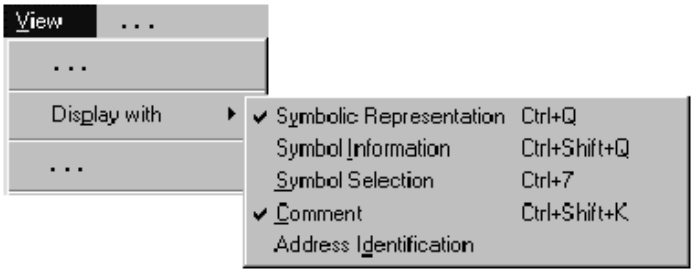
در این بخش شما نوشتن یک دستور **AND** ، یک دستور **OR** و دستور **Set/Rest** حافظه را بزبان نمایش عبارتی (STL) فرا خواهید گرفت.

برنامه نویسی یک دستور AND بزبان نمایش عبارتی

در صورت لزوم زبان برنامه نویسی را در منوی **View** به **STL** تغییر دهید.

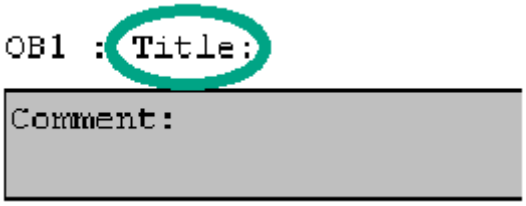


بررسی کنید که نمایش سمبلیک فعال شده باشد.

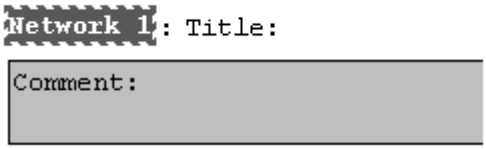


در فضـــــای **Title** بلـــــوک **OB1** کلـــــیک کـــــرده و بعـــــنوان مـــــثال عـــــبارت

" Cyclically Processed main program " را وارد کنید.



در محیط وارد کردن عبارت ها کلیک نمایید.



در ابـتدای خـط یـک **A (AND)** وارد نمایـید و بعد از یک **Space** سپس نام سمبلیک **Key–1** (درداخل گیومه) را وارد نمایید.

خط را توسط **Enter** تکمیل نمایید. مکان نما به خط بعد پرش خواهد کرد.

بروش مشابه دستور **AND** را همانطور که مشاهده می شود تکمیل نمایید.

```
A     "Key_1"
A     "Key_2"
=     "Green_Light"
```

حـال شـما یـک دسـتور **AND** را بطـور کامل ایجاد نموده اید. در صورتیکه هیچ سمبل دیگری برنگ قرمز وجود ندارد بلوک را ذخیره نمایید.

سمبل هایـیکه بـرنگ قرمـز نمـایش داده می شوند یا در جدول سمبل ها وجود ندارد یا یک خطای **Syntax** وجود دارد شـما مـی توانـید بطـور مسـتقیم نـام سـمبلیک را از جدول سمبل ها نیز وارد نمایید. بر روی علامت **???** کلیک نمـوده و سـپس در داخـل مـنو فـرمان **Insert>Symbol** را اجـرا نمایید. در لیست **Pull down** نام سمبلیک بطور خودکار اضافه خواهد شد.

برنامه نویسی یک دستور OR بزبان Statement List

Network1 را انتخاب نمایید.

یک **Network** جدیدایجاد نموده و مجدداً فضای وارد نمودن اطلاعات را انتخاب نمایید.

یک دستور **O (OR)** بنام **Key–3** (همانطوریکه برای دستور **AND** انجام دادید) ایجاد نمایید.

```
O       "Key_3"
```

دستور **OR** را کامل نموده و آنرا ذخیره نمایید.

```
O       "Key_3"
O       "Key_4"
=       "Red_Light"
```

برنامه نویسی یک دستور حافظه ای بزبان نمایش عبارتی

Network2 را انتخاب کرده و **Network** جدیدی ایجاد نمایید.

در خط اول دستور **A** را با نام سمبلیک **Automatic–on** وارد نمایید.

```
A       "Automatic_On"
```

دستورات حافظه را تکمیل نموده و آنرا ذخیره نمایید. بلوک را ببندید.

```
A       "Automatic_On"
S       "Automatic_Mode"
A       "Manual_On"
R       "Automatic_Mode"
```

اگر بخواهید می توانید تفاوت بین آدرس دهی سمبلیک و مطلق را با غیر فعال نمودن فرمان **View > Display > Symbolic Representating** مشاهده نمایید.

```
A       "Key_1"
A       "Key_2"
=       "Green_Light"
```

Example:
Symbolic addressing in STL

```
A       I       0.1
A       I       0.2
=       Q       4.0
```

Example:
Absolute addressing in STL

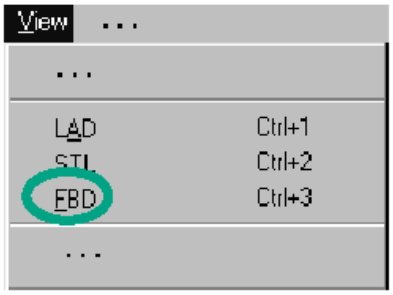
برای اطلاعات بیشتر می توانید به مباحث **Programming Blocks** و **Creating Logic Blocks** و **Editing STL Statements** واقع در **Help > Contents** مراجعه نمایید.

4.4 برنامه نویسی OB1 بزبان Function Block Diagram

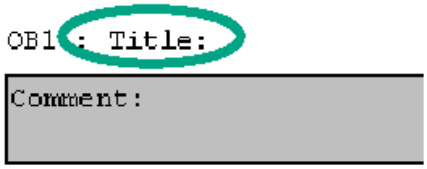
در این بخش شما نوشتن یک دستور **AND** ، یک دستور **OR** و یک دستور مربوط به حافظه را بزبان **Function Block Diagram (FBD)** فرا خواهید گرفت.

برنامه نویسی یک دستور **AND** برنامه **Function Block Diagram**

در صورت لزوم زبان برنامه نویسی را در منوی **View** به **FBD** تغییر دهید.



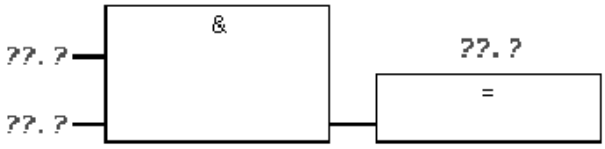
در فضـــــای **Title** بلـــــوک **OB1** کلـــــیک کـــــرده و بعـــــنوان مـــــثال عـــــبارت **Cyclically Processecl main program** را وارد کنید.



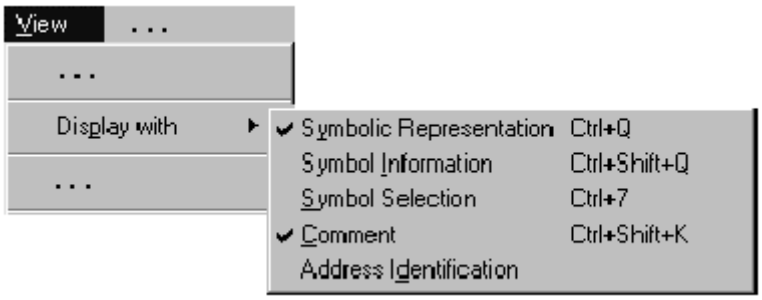
در فضای وارد کردن ورودی ها (زیر فضای توضیحات) کلیک نمایید.

یک **Box** ,**AND (&)** و یک **Assignment (=)** وارد نمایید.

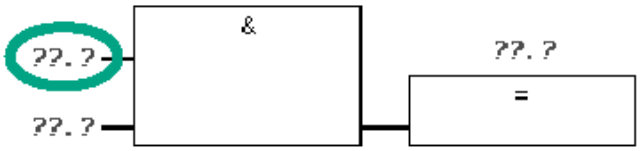
آدرس عناصر بکار رفته در تابع **AND** هنوز وارد نگشته اند.



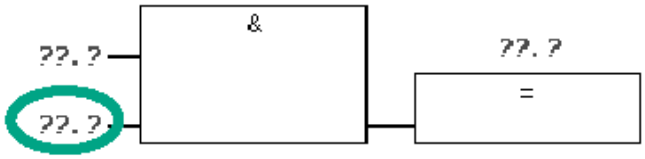
بررسی کنید که نمایش سمبلیک فعال شده باشد.



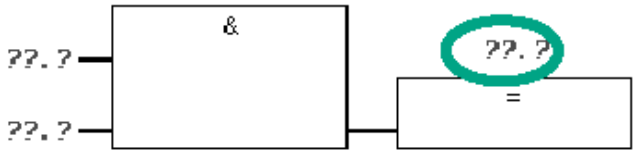
بـر روی نشانه **??.?** کلیک نموده و نام سمبلیک **Key - 1** را (در داخل گیومه) وارد نموده و با **Enter** تایید نمایید.



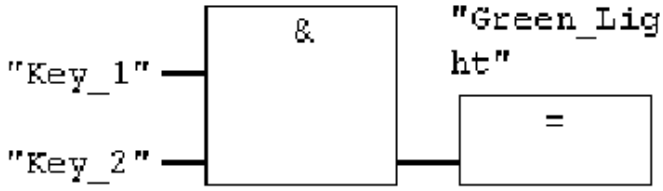
برای ورودی دوم نام سمبلیک **"Key – 2"** را وارد نمایید.



برای **Assignment** نام **"Green – Light"** را وارد نمایید.



هم اکنون شما یک تابع **AND** را برنامه نویسی نموده اید.



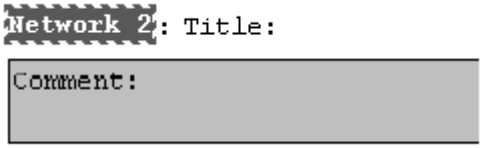
اگر دیگر هیچ سمبلی برنگ قرمز مشاهده نمی شود می توانید بلوک را ذخیره نمایید.

سمبل هایکه برنگ قرمز نمایش داده می شوند یا در جدول سمبل ها وجود ندارد یا یک خطای **Syntax** وجود دارد شما می توانید بطور مستقیم نام سبملیک را از جدول سمبل ها نیز وارد نمایید. بر روی علامت **??.?** کلیک نموده و سپس در داخل منو فرمان **Insert > Symbol** را اجرا نمایید. در لیست **Pull down** نام سمبلیک بطور خودکار اضافه خواهد شد.

نوشتن یک تابع OR بزبان Function Block Diagram

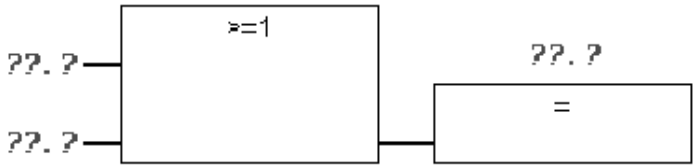
یک **Network** جدید ایجاد نمایید.

فضای وارد نمودن اطلاعات جهت تابع **OR** را انتخاب نمایید.

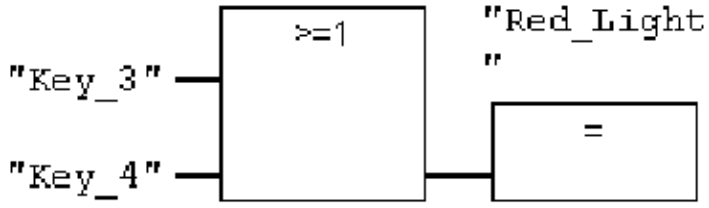


یک **Box** ، **OR** و یک **Assignment** وارد نمایید.

آدرس های تابع **OR** (**1≥**) نوشته شده هنوز وارد نگردیده اند. بهمان روشی که برای تابع **AND** بکار بردید عمل نمایید.



بـرای ورودی بالا **Key – 3** و برای ورودی پایینی **Key – 4** و برای **Assigment** ، **Real-Light** را وارد نمایید.

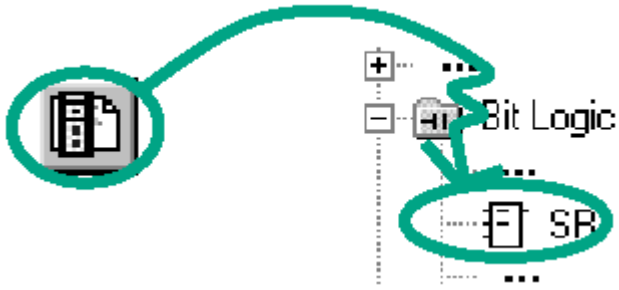


بلوک را ذخیره نمایید.

نوشتن یک تابع حافظه ای بزبان Function Block Diagram

Network2 را انـتخاب نمـوده و **Network** دیگری وارد نمایید. مجدداً فضای وارد نمودن اطلاعات را انتخاب نمایید.(زیر فضای توضیحات)

در داخل کاتالوگ عناصر برنامه زیر **Bit Logic** آنقدر **Navigate** نمائید تا به عنصر **SR** برسید دوبار کلیک نمایید تا عنصر را وارد نمایید.

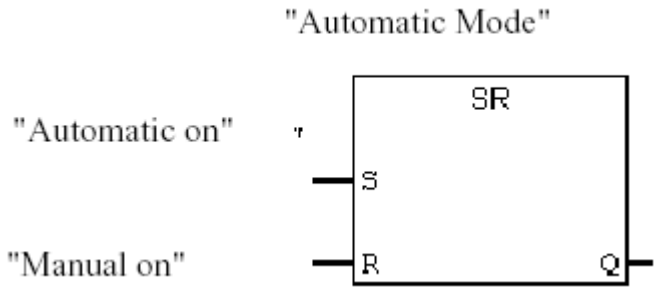


اسامی سمبلیک زیر را برای عنصر **SR** وارد نمایید :

برای ورودی **Set** : **"Automatic – on"**

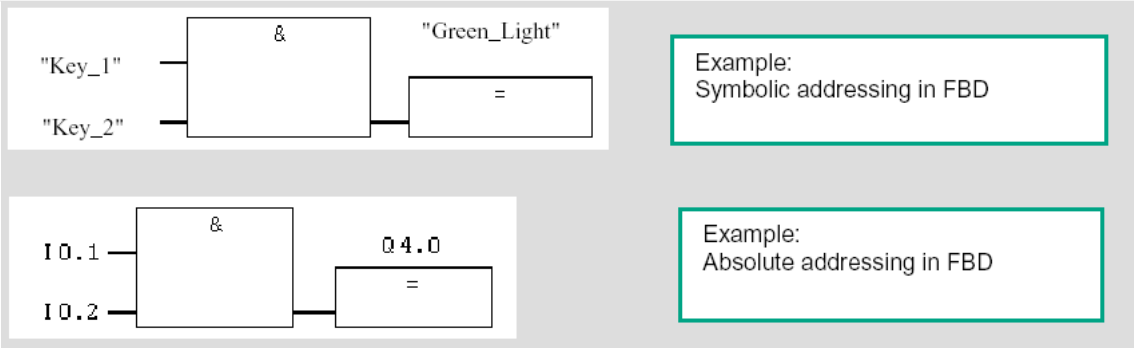
برای ورودی **Reset** : **"Manual – on"**

برای بیت حافظه بکار رفته : **"Automatic – Mode"**



بلوک را ذخیره نموده و پنجره را ببندید.

اگـــر بخواهـــید میتوانـــید بیـــن آدرس دهـــی ســـمبلیک و مطلـــق را بـــا غـــیر فعـــال نمـــودن فـــرمان **View > Display > Symbolic Representation** مشاهده نمایید.



شـما می توانید فرمت خط را برای آدرس دهی سمبلیک در پنجره برنامه **LAD/STL/FBD** با استفاده از منو فرمان **Options >Customize** و سپس انتخاب **Width of address field** در **LAD/FBD** تغییر دهید.

در اینجا شما می توانید فرمت خط را بین 10 تا 24 حرف تنظیم نمایید.

◙ بـرای اطلاعـات بیشـتر مـی توانـید بـه مباحـث **Programming blocksCreating logic blocks,** و **Editing ladder instructions** واقع در **Help > Contents** مراجعه نمایید.

5. ایجاد یک برنامه با استفاده از بلوک های تابعی و بلوک های اطلاعاتی

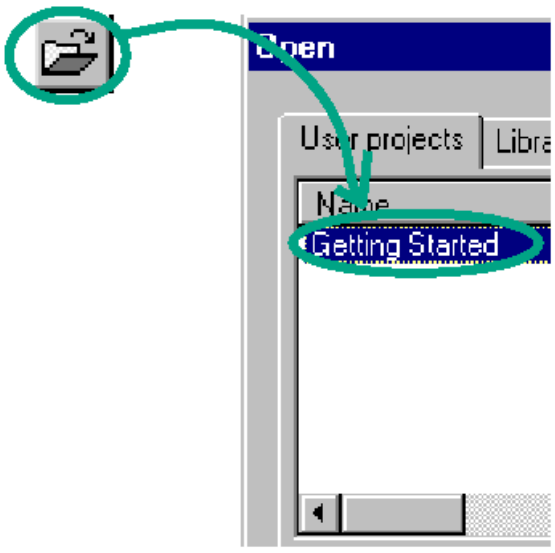
5.1 : ایجاد و باز کردن بلوک های تابعی (FB)

از نظر تقدم برنامه ای بلوک تابعی **(FB)** بعد از بلوک سازماندهی قرار دارد و قسمتی از برنامه را در بر می گیرد می تواند بدفعات در داخل **OB1** فراخوانی شود. کلیه پارامترهای متداول و اطلاعات **Static** بلوک تابعی در بلوک اطلاعاتی **(DB)** مجزایی که به بلوک تابع مذکور منتسب می گردد ذخیره می شوند.

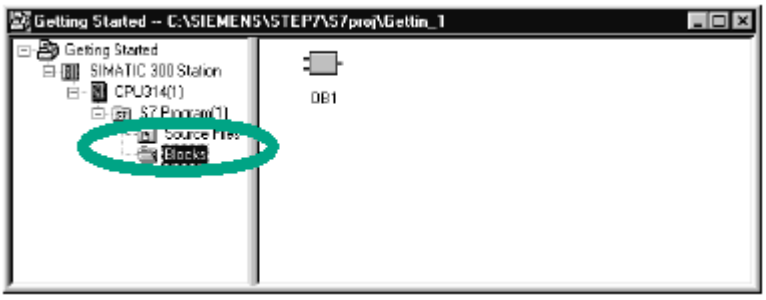
شما بلوک تابعی (**FB1** ، با نام سمبلیک **Engine** ؛ جدول سمبل های صفحه 3-3 را ملاحظه نمایید.) را در پنجره برنامه **LAD/STL/FBD** که قبلاً با آن آشنا شده اید خواهید نوشت. بدین منظور شما باید همان زبان برنامه نویسی ای را که در فصل 4 (برنامه نویسی **OB1**) استفاده نمودید را بکار برید.

همچنین میبایست جدول سمبل ها را در داخل پروژه **Getting started** خود کپی نموده باشید. اگر این کار را انجام نداده اید روش انجام آنرا در صفحه 4-2 مطالعه نموده و پس از کپی کردن جدول سمبل ها به این فصل باز گردید.

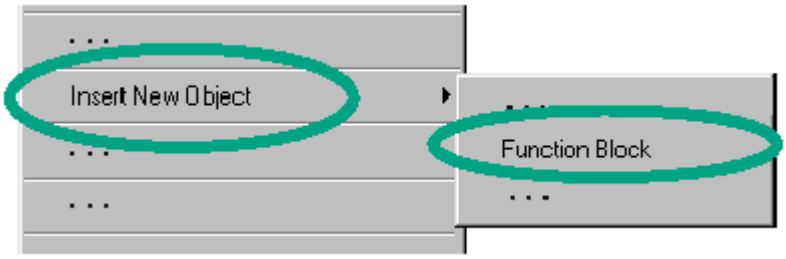
در صورت لزوم پروژه **" Getting Started "** را باز نمایید.



پوشه های تو در تو را باز نموده تا به پوشه **Block** رسیده و آنرا باز نمایید. در نیمه راست صفحه کلیک راست نمایید.

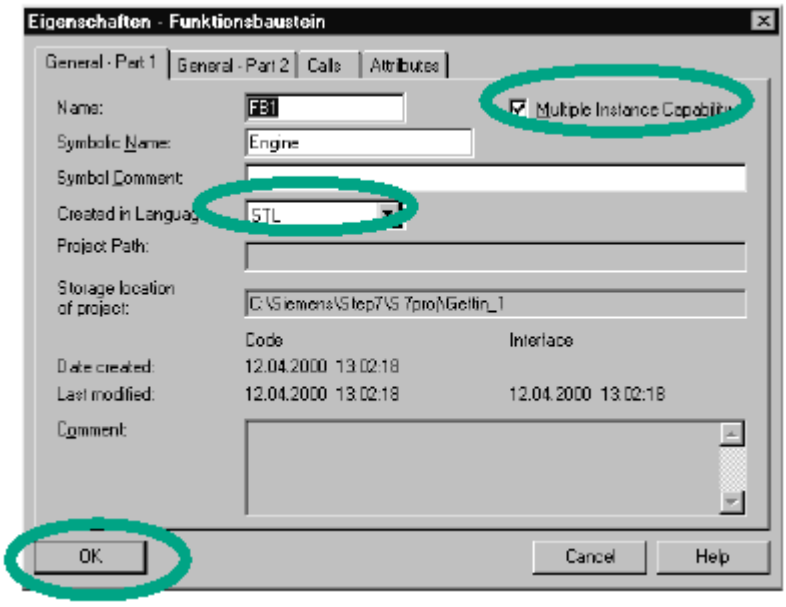


منوی **Pup – up** کلیک راست موس مهمترین دستورات **Menu bar** را شامل می شود. یک **Function Block** را بعنوان یک **New Obeject** وارد نمایید.

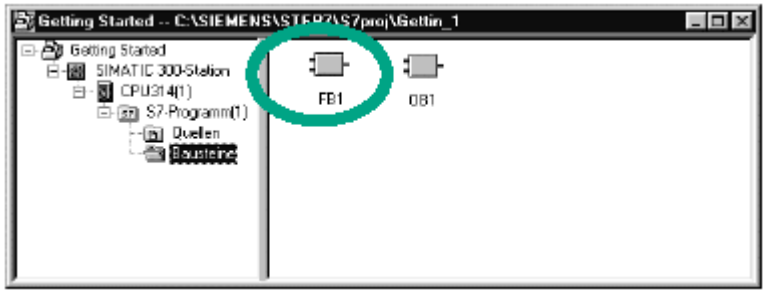


بر روی **FB1** دوبار کلیک نموده تا پنجره برنامه **LAD/STL/FBD** را باز نمایید.

در **dialog box** ، **" Properties – Function Block "** زبانی را که قصد دارید بلوک را با آن بنویسید ایجاد نمایید ، **Multiple instance FB , Check Box** را انتخاب نموده و مابقی تنظیمات را توسط **OK** تایید نمایید.



بلوک تابعی **FB1** در پوشه بلوک ها وارد گردیده است.



بسته به زبان برنامه نویسی ای که انتخاب کرده اید ، مطالعه تان را برای منطق نردبانی در بخش 5.2 ، برای نمایش عبارتی در بخش5.3 یا **Function Block Diagram** را در بخش 5.4 ادامه دهید.

◙ برای اطلاعات بیشتر می توانید به مباحث **Programming Blocks** و **Creating Blocks & Libraries** واقع در **Help > Contents** مراجعه نمایید.

5.2 برنامه نویسی FB1 بزبان منطق نردبانی

حال قصد داریم بشما نشان دهم چگونه میتوان یک بلوک تابعی ای را که بعنوان مثال توسط آن میتوان با استفاده از دو بلوک اطلاعاتی متفاوت یک موتور دیزلی یا بنزینی را مونیتور و کنترل نمود را برنامه نویسی نمود.

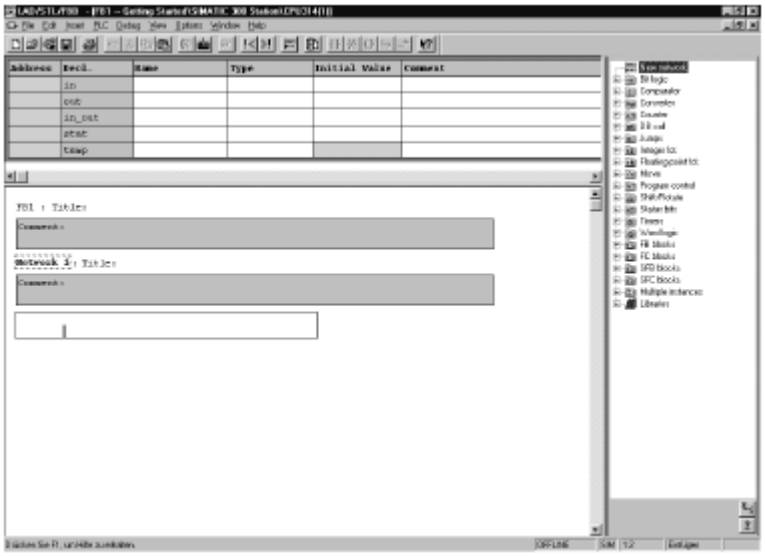
کلیه سیگنال های " Engine – Specific " بین بلوک سازماندهی و بلوک تابعی بصورت پارامترهای بلوکی مبادله می شوند لذا میبایست در جدول Variable Decleration بصورت پارامترهای ورودی و خروجی لیست شوند. (Declaration, In و Out)

در اینجا لازمست بدانید که چگونه میتوان یک مدار سری ، یک مدار موازی و یک تابع حافظه را توسط STEP7 وارد نمود.

1. **پرکردن جدول** Variable Decleration

در حال حاضر پنجره برنامه LAD/STL/FBD باز است و انتخاب View > LAD (زبان برنامه نویسی) فعال است.

توجه داشته باشید که FB1 هم اکنون در Header قرار دارد زیرا برای باز کردن پنجره برنامه دوبار بر روی FB1 کلیک نموده اید.



Declaration های زیر را در داخل Variable Declaration وارد نمایید.

بدین منظور بر روی یک خانه کلیک نمایید و با استفاده از جدولی که در زیر نمایش داده شده است اسم و توضیحات مربوطه را عیناً وارد نمایید.

شما می توانید نوع (Type) را با استفاده از دستور Elementary Types در داخل منوی Pop-Up مکان نما با کلیک راست موس فعال می شود انتخاب نمایید.

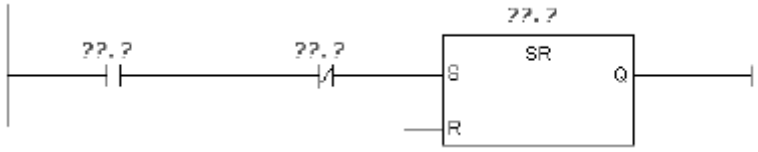
بعد از فشردن کلید **Enter** ، که به ستون بعدی پرش می کند یا اینکه یک ردیف جدید ایجاد می گردد.

Address	Decl.	Name	Type	Initial Value	Comment
0.0	in	Switch_On	BOOL	FALSE	Switch on engine
0.1	in	Switch_Off	BOOL	FALSE	Switch off engine
0.2	in	Failure	BOOL	FALSE	Engine failure, causes the engine to switch off
2.0	in	Actual_Speed	INT	0	Actual engine speed
4.0	out	Engine_On	BOOL	FALSE	Engine is switched on
4.1	out	Preset_Speed_Reached	BOOL	FALSE	Preset speed reached
	in_out				
6.0	stat	Preset_Speed	INT	1500	Requested engine speed
	temp				

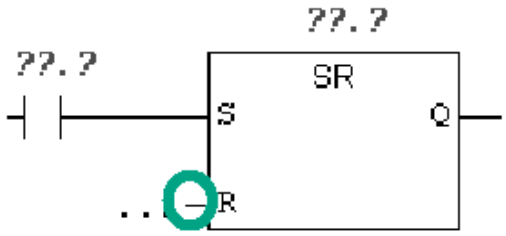
تنها حروف ، اعداد و خط زیر (_) کاراکترهای مجاز برای اسامی بلوک پارامترها در جدول **Variable Declearation** هستند.

2 . **نوشتن برنامه خاموش و روشن کردن یک موتور**

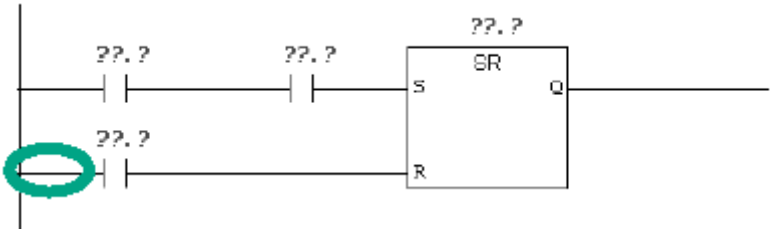
یک کنتاکت حالت عادی باز ، یک کنتاکت حالت عادی بسته و یک عنصر **SR** را بصورت سری و با استفاده از **Button** های مربوطه شان واقع در منوی ابزار یا با استفاده از کاتالوگ عناصر برنامه ایجاد نمایید.



سپس مسیر جاری را بلافاصله پیش از ورودی **R** قرار دهید.

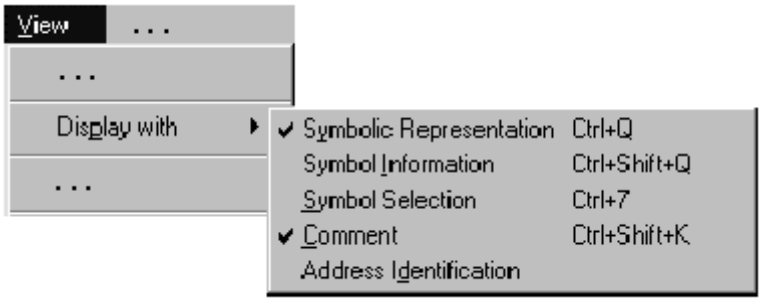


یک کنتاکت حالت عادی باز دیگر وارد نمایید. مسیر جاری را بلافاصله قبل از این کنتاکت قرار دهید.



یک کنتاکت حالت عادی بسته را بصورت موازی با کنتاکت حالت عادی باز ایجاد نمایید.

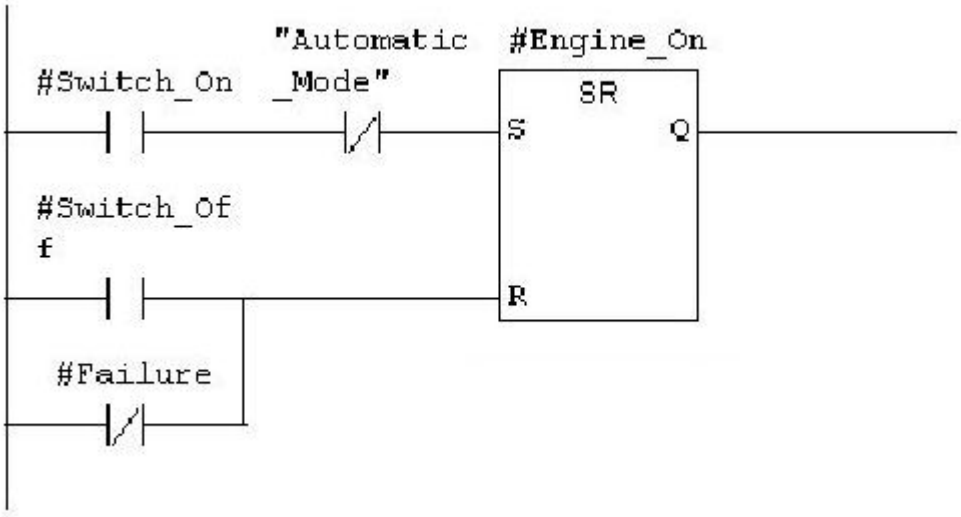
بررسی نمایید که نمایش سیمبلیک فعال شده باشد.



علامت های سئوال را انتخاب نموده و از داخل جدول نمایش متغیرها اسامی مربوطه را وارد نمایید.

(علامت # بطور خودکار اعمال خواهد شد.)

نام سمبلیک **" Automatic – Mode "** را برای کنتاکت حالت عادی بسته مدار سری وارد نمایید.



■ متغیرهای بلوک محلی **(Local)** با علامت # نمایش داده می شوند و فقط در داخل بلوک معتبر می باشند.

متغیرهای عمومی **(Global)** در داخل گیومه نمایش داده می شوند. این متغیرها در جدول سمبل ها نمایش داده شده و در کل برنامه معتبر میباشند.

ارزش سیگنال **" Automatic – Mode "** در **OB1** (**Network3** از صفحه 7 – 4 را مطالعه کنید) و با استفاده از یک عنصر **SR** دیگر تعریف شده است و در حال حاضر در **FB1** مورد استفاده قرار گرفته است.

3 . نوشتن برنامه مونیتورینگ سرعت

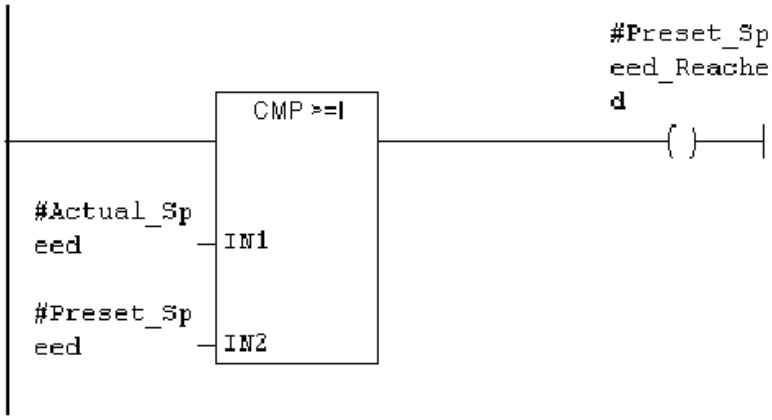
یک **Network** جدید ایجاد نموده و مسیر جاری را انتخاب نمایید.

سپس در کاتالوگ عناصر برنامه **Navigate** نموده تا به تابع **Compare** رسیده و یک **GE-I** را انتخاب نمایید.

آنگاه در مسیر جاری یک سیم پیچ وارد نمایید.

مجدداً علامت های سئوال را انتخاب کرده و مقایسه کننده و سیم پیچ را با نام های مربوطه از داخل جدول نمایش متغیرها نامگذاری نمایید.

سپس برنامه را ذخیره نمایید.



■ چه زمانی موتور روشن یا خاموش خواهد شد ؟

هنگامی که متغیر **On – Switch #** " **1** " را دارد و متغیر " **Automatic – Mode** " مقدار 0 را دارد موتور روشن خواهد شد. تا وقتی که " **Automatic – Mode**" **(Negate)** باشد این تابع فعال نخواهد شد.

(کنتاکت حالت عادی بسته)

هنگامیکه متغیر **Swith – off #** مقدار **"1"** را داشته باشد یا متغیر **Fault #** مقدار 0 را داشته باشند موتور خاموش خواهد شد. این تابع مجدداً زمانی فعال خواهد شد که **Fault#** را **Negate** نماییم. (**Fault #** یک سیگنال **Zero – Active** بوده و در حالت عادی مقدار **1** و در صورت بروز خطا مقدار 0 را خواهد داشت.)

چگونه مقایسه کننده ، سرعت موتور را مونیتور می نماید ؟

مقایسه کننده متغیرهای **Actual – Speed #** و **Setpoint – Speed #** را مقایسه کرده و نتیجه را به سیگنال **Setpoint – Speed – Reached #** اعمال می نماید (مقدار سیگنال **1**)

برای اطلاعات بیشتر میتوانید به مباحث **Programing Blocks** ، **Creating Logic Blocks** و **Editing the Variable declaration table** یا **Editing LAD Instructions** واقع در **Help > Contents** مراجعه نمایید.

5.3 نوشتن FB1 بزبان نمایش عبارتی

حال قصد داریم بشما نشان دهیم چگونه میتوان یک بلوک تابعی که با استفاده از آن بعنوان مثال یک موتور بنزینی یا دیزلی را با بهره گیری از دو بلوک اطلاعاتی مجزا میتوان کنترل و مونیتور نمود را برنامه نویسی نمود.

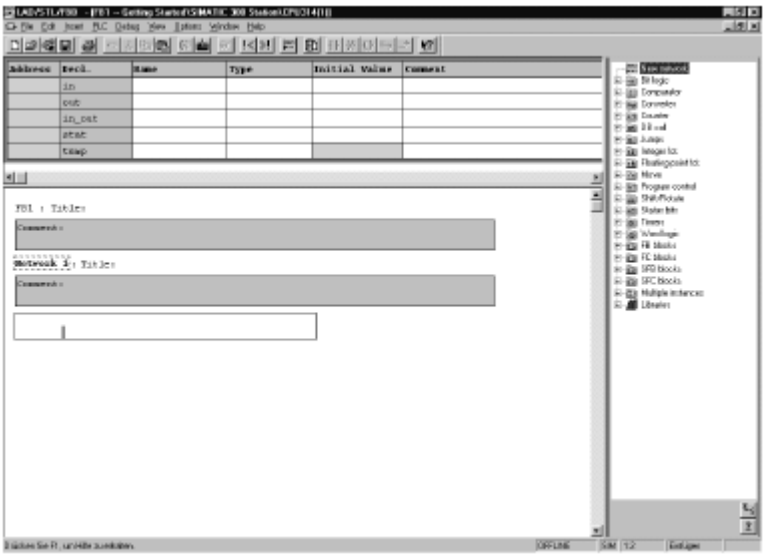
کلیه سیگنال های **" Engine – Specific "** در قالب پارامترهای بلوکی از بلوک سازماندهی به بلوک تابعی منتقل می شوند لذا می بایست در داخل جدول نمایش متغیرها بعنوان پارامترهای ورودی و خروجی درج گردند. (با نمایش **" In "** و **" Out "**)

شما در حال حاضر میبایست بدانید که چگونه یک دستور **AND** ، یک دستور **OR** و دستورات حافظه ای **Set / Reset** را با استفاده از **STEP7** وارد نمایید.

1. تکمیل جدول نمایش متغیرها

در حال حاضر پنجره برنامه **LAD/STL/FBD** باز بوده و انتخاب **View >STL** (زبان برنامه نویسی) فعال میباشد.

توجه داشته باشید که در حال حاضر بدلیل آنکه برای باز کردن پنجره برنامه بر روی **FB1** دوبار کلیک نموده اید ، **FB1** در **Header** قرار دارد.



عبارات زیر را درجدول نمایش متغیرها وارد نمایید.

برای انجام این کار در داخل یک خانه کلیک نموده و نام و توضیح مربوطه مطابق آنچه در ذیل نمایش داده شده است را وارد نمایید.

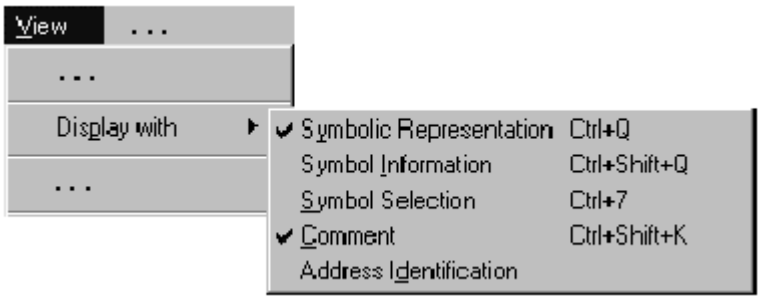
شما می توانید نوع متغیرها را با استفاده از دستور **Elementary types** در داخل منوی **Pop-Up** که با کلیک راست موس پدیدار می شود انتخاب نمایید. با فشردن دکمه **Entter** مکان نما به ستون بعدی رفته و یا ردیف جدیدی ایجاد خواهد شد.

Address	Decl.	Name	Type	Initial Value	Comment
0.0	in	Switch_On	BOOL	FALSE	Switch on engine
0.1	in	Switch_Off	BOOL	FALSE	Switch off engine
0.2	in	Failure	BOOL	FALSE	Engine failure, causes the engine to switch off
2.0	in	Actual_Speed	INT	0	Actual engine speed
4.0	out	Engine_On	BOOL	FALSE	Engine is switched on
4.1	out	Preset_Speed_Reached	BOOL	FALSE	Preset speed reached
	in_out				
6.0	stat	Preset_Speed	INT	1500	Requested engine speed
	temp				

■ تنها حروف ، اعداد و خط زیر (_) کاراکترهای مجاز برای اسامی پارامترهای بلوک در داخل جدول نمایش متغیرها می باشند.

2. نوشتن برنامه روشن و خاموش کردن یک موتور

بررسی نمایید که نمایش سمبلیک فعال باشد.



دستورات مربوطه را در داخل **Network 1** وارد نمایید.

```
A     #Switch_On
AN    "Automatic_Mode"
S     #Engine_On
O     #Switch_Off
ON    #Failure
R     #Engine_On
```

متغیرهای بلوک محلی (**Local**) با علامت # نمایش داده می شوند و فقط در داخل بلوک معتبر می باشند. متغیرهای عمومی (**Global**) در داخل گیومه نمایش داده می شوند. این متغیرها در جدول سمبل ها نمایش داده شده و در کل برنامه معتبر میباشند.

ارزش سیگنال **"Automatic - Mode"** در **OB1** (**Network3**) از صفحه 7 – 4 را مطالعه کنید) و با استفاده از یک عنصر **SR** دیگر تعریف شده است و در حال حاضر در **FB1** مورد استفاده قرار گرفته است.

3. نوشتن برنامه مونیتورینگ سرعت

یک **Network** جدید ایجاد نمود و دستورات مربوطه را وارد نمایید. برنامه تان را ذخیره نمایید.

```
L     #Actual_Speed
L     #Preset_Speed
>=I
=     #Preset_Speed_Reached
```

چه زمانی موتور روشن یا خاموش خواهد شد ؟

هنگامی که متغیر **On – Switch #** مقدار "1" را دارد و متغیر **"Automatic – Mode "** مقدار 0 را دارد موتور روشن خواهد شد. تا وقتی که **"Automatic – Mode"** باشد این تابع فعال نخواهد شد. **(Negate)** (کنتاکت حالت عادی بسته)

هنگامیکه متغیر **Switch – off #** مقدار **"1"** را داشته باشد یا متغیر **Fault #** مقدار 0 را داشته باشند موتور خاموش خواهد شد. این تابع مجدداً زمانی فعال خواهد شد که **Fault#** را **Negate** نماییم. (**Fault #** یک سیگنال **Zero – Active** بوده و در حالت عادی مقدار **1** و در صورت بروز خطا مقدار 0 را خواهد داشت.)

چگونه مقایسه کننده ، سرعت موتور را مونیتور می نماید ؟

مقایسه کننده متغیرهای **Actual – Speed #** و **Setpoint – Speed #** را مقایسه کرده و نتیجه را به سیگنال **Setpoint – Speed – Reached #** اعمال می نماید (مقدار سیگنال **1**)

برای اطلاعات بیشتر میتوانید به مباحث **Programming Blocks** ، **Creating Logic Blocks** و **Editing the Variable Dclaration table** یا **Editing STL Instructions** واقع در **Help > Contents** مراجعه نمایید.

5.4 نوشتن FB1 بزبان Function Block Diagram

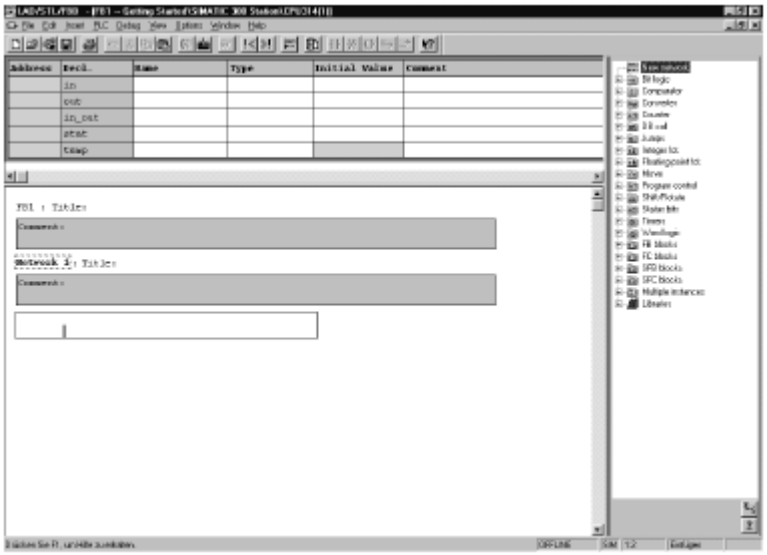
حال به شما نشان خواهیم داد چگونه میتوانید یک بلوک تابعی که با استفاده از آن و بکار بردن دو بلوک اطلاعاتی متفاوت بعنوان مثال می توانید یک موتور بنزینی یا دیزلی را مونیتور و کنترل نمایید را بنویسید.

کلیه سیگنال های **" Engine – Specific "** در قالب پارامترهای بلوکی از بلوک سازماندهی به بلوک تابعی منتقل می شوند لذا میبایست در داخل جدول نمایش متغیرها بعنوان پارامترهای ورودی و خروجی قید شوند (نمایش بصورت **"in"**و**"out"**). شما همچنین میبایست بدانید چگونه یک دستور **AND** ، یک دستور **OR** و یک تابع حافظه ای را در **STEP7** وارد نمایید.

4. تکمیل جدول نمایش متغیرها

در حال حاضر پنجره برنامه **LAD/STL/FBD** باز بوده و انتخاب **View >STL** (زبان برنامه نویسی) فعال میباشد.

توجـه داشـته باشـید که در حال حاضر بدلیل آنکه برای باز کردن پنجره برنامه بر روی **FB1** دوبار کلیک نموده اید ، **FB1** در **Header** قرار دارد.



عبارات زیر را درجدول نمایش متغیرها وارد نمایید.

بـرای انجـام ایـن کـار در مـراحل یک خانه کلیک نموده و نام و توضیح مربوطه مطابق آنچه در ذیل نمایش داده شده است را وارد نمایید.

شـما می توانید نوع متغیرها را با استفاده از دستور **Elementary types** در داخل منوی **Pop-Up** که بـا کلـیک راسـت مـوس پدیـدار مـی شود انتخاب نمایید. با فشردن دکمه **Entter** مکان نما به ستون بعدی رفته و یا ردیف جدیدی ایجاد خواهد شد.

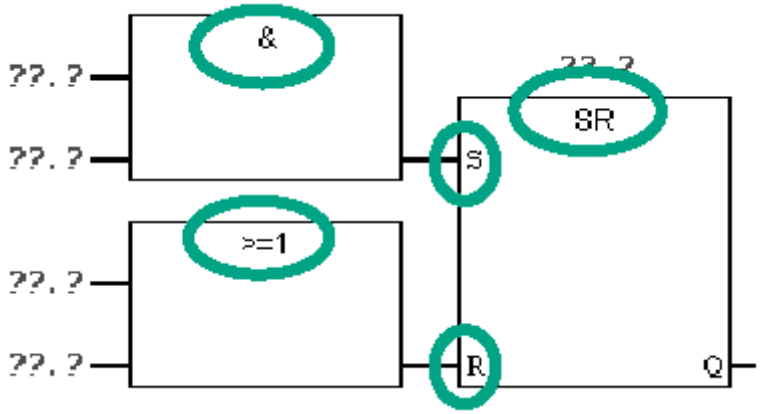
Address	Decl.	Name	Type	Initial Value	Comment
0.0	in	Switch_On	BOOL	FALSE	Switch on engine
0.1	in	Switch_Off	BOOL	FALSE	Switch off engine
0.2	in	Failure	BOOL	FALSE	Engine failure, causes the engine to switch off
2.0	in	Actual_Speed	INT	0	Actual engine speed
4.0	out	Engine_On	BOOL	FALSE	Engine is switched on
4.1	out	Preset_Speed_Reached	BOOL	FALSE	Preset speed reached
	in_out				
6.0	stat	Preset_Speed	INT	1500	Requested engine speed
	temp				

■ تنها حروف ، اعداد و خط زیر (_) کاراکترهای مجاز برای اسامی پارامترهای بلوک در داخل جدول نمایش متغیرها می باشند.

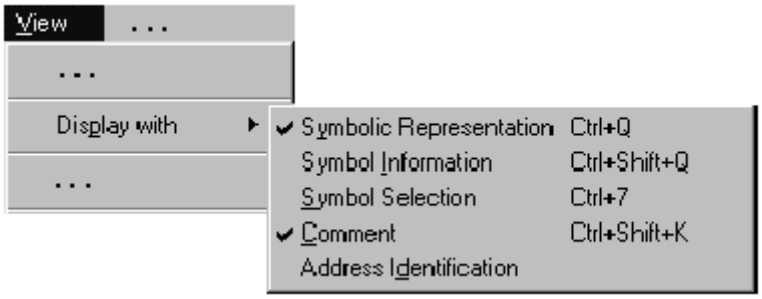
2. نوشتن برنامه خاموش و روشن کردن یک موتور

بـا اسـتفاده از کاتـالوگ عناصـر برنامه یک تابع **SR** را در داخل **Network1** وارد نمایید (پوشه **Bit Logic**)

در ورودی **S (SET)** یک باکس **AND** و در ورودی **R (RESET)** یک باکس **OR** اضافه نمایید.



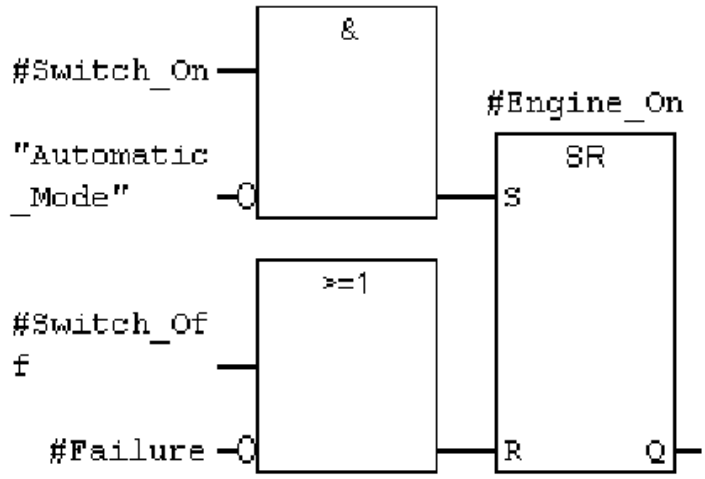
بررسی نمایید که نمایش سمبلیک فعال شده باشد.



بـر روی علامت **??.?** کلیک نموده و اسامی مربوطه را از جدول نمایش وارد نمایید (علامت # بطور خودکار اضافه خواهد شد)

اطمیـنان حاصـل کنـید که یک ورودی تابع **AND** با نام سمبلیک **" Automatic – Mode "** آدرس دهـی شـده باشـد. ورودی هـای **" Automatic – Mode "** و **Fault #** را بـا استفاده از **Button** های موجود در منوی ابزار **(Tool Bar)** **Negate** نمایید.

سپس برنامه تان را ذخیره نمایید.



متغیرهای بلوک محلی **(Local)** با علامت # نمایش داده می شوند و فقط در داخل بلوک معتبر می باشند.

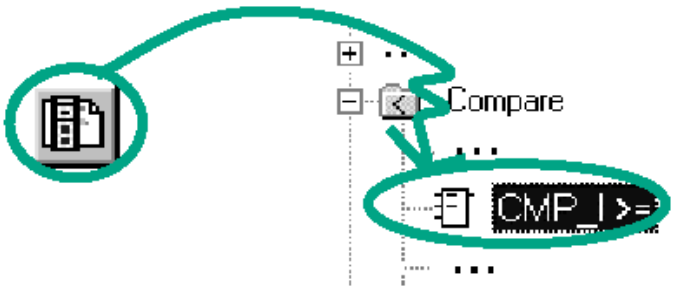
متغیرهای عمومی **(Global)** در داخل گیومه نمایش داده می شوند. این متغیرها در جدول سمبل ها نمایش داده شده و در کل برنامه معتبر میباشند.

ارزش سیگنال **" Automatic – Mode "** در **OB1** (**Network3** ؛ صفحه 7 – 4 را مطالعه کنید) و با استفاده از یک عنصر **SR** دیگر تعریف شده است و در حال حاضر در **FB1** مورد استفاده قرار گرفته است.

3. نوشتن برنامه مونیتورینگ سرعت

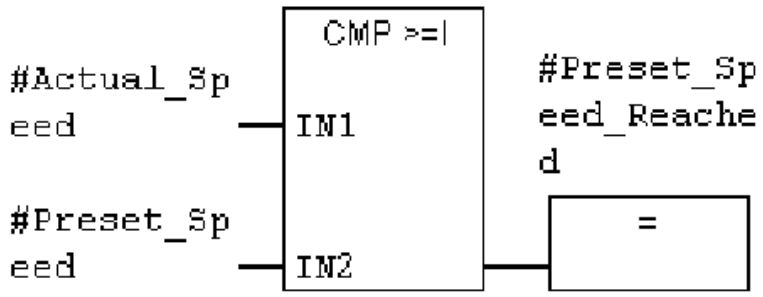
یک **Network** جدید ایجاد نموده و فضای ورودی ها را انتخاب نمایید.

سپس در کاتالوگ عناصر برنامه **Navigate** نموده تا به تابع **Compare** رسیده و یک **CMP>=1** را انتخاب نمایید.



یک **Out put assignment** را به مقایسه کننده افزوده و ورودی ها را با اسامی جدول نمایش متغیرها آدرس دهی نمایید.

سپس برنامه تان را ذخیره نمایید.



چه زمانی موتور روشن یا خاموش خواهد شد ؟

هنگامی که متغیر" **On - Switch** " مقدار 1 را دارد و متغیر " **Automatic – Mode** " مقدار 0 را دارد موتور روشن خواهد شد. تا وقتی که " **Automatic – Mode**" **(Negate)** باشد این تابع فعال نخواهد شد. (کنتاکت حالت عادی بسته)

هنگامیکه متغیر **Swith – off #** مقدار **"1"** را داشته باشد یا متغیر **Fault #** مقدار 0 را داشته باشد موتور خاموش خواهد شد. این تابع مجدداً زمانی فعال خواهد شد که **Fault#** را **Negate** نماییم. (**Fault #** یک سیگنال **Zero – Active** بوده و در حالت عادی مقدار **1** و در صورت بروز خطا مقدار 0 را خواهد داشت.)

چگونه مقایسه کننده ، سرعت موتور را مونیتور می نماید ؟

مقایسه کننده متغیرهای **Actual – Speed #** و **Setpoint – Speed #** را مقایسه کرده و نتیجه را به سیگنال **# Setpoint – Speed – Reached** اعمال می نماید (مقدار سیگنال **1**)

برای اطلاعات بیشتر میتوانید به مباحث **Programming Blocks** ، **Creating Logic Blocks** و **Editing the Variable Declaration table** یا **Editing FBD Instructions** واقع در **Help > Contents** مراجعه نمایید.

5.5 ایجاد بلوکهای اطلاعاتی Instance و تغییر مقادیر واقعی

حال شما برنامه بلوک تابع ساز **FB1** (**Engine**) را نوشته و از بین چیزهای دیگر پارامترهای **Engine _ Specific** را در داخل جدول نمایش متغیرها تعریف نموده اید.

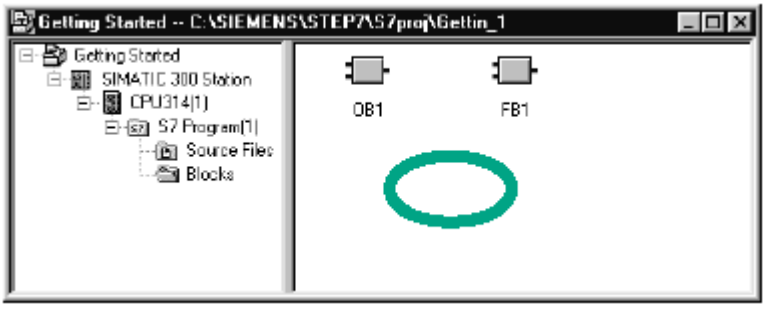
برای اینکه در آینده قادر باشید بلوک تابع ساز را از داخل **OB1** فراخوانی نمایید میبایست بلوک اطلاعاتی مربوطه را ایجاد نمایید.

یک بلوک اطلاعاتی **Instance (DB)** همواره به یک بلوک تابع ساز نسبت داده می شود.

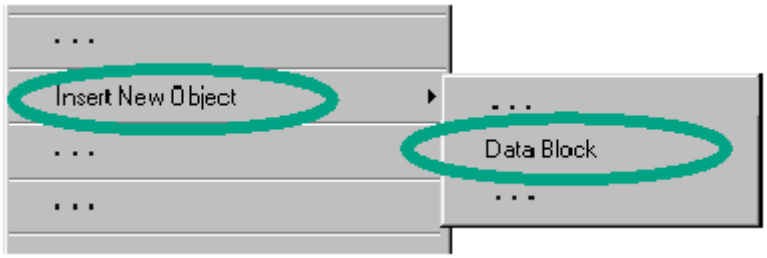
این بلوک تابع ساز برای کنترل و مونیتور نمودن یک موتور بنزینی یا دیزلی بکار می رود.

========متن========

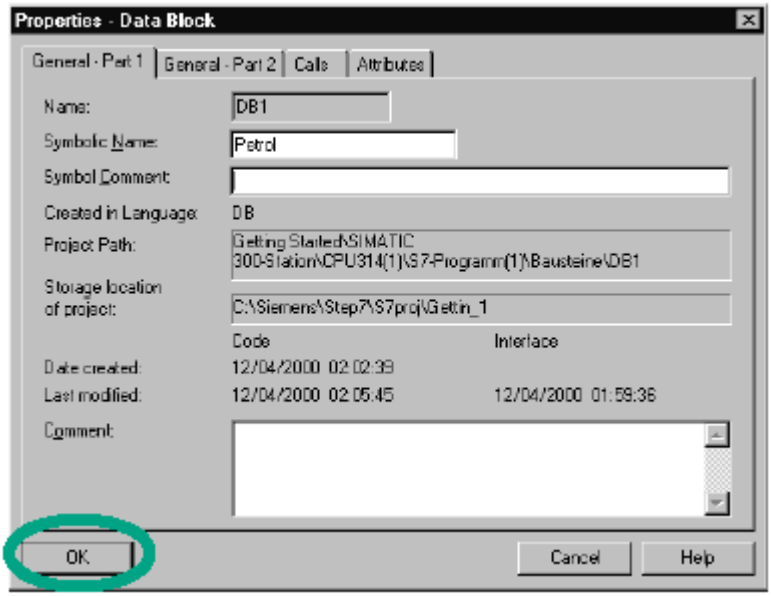
در داخل پروژه **Getting started** که در صفحه **SIMATIC Manager** باز می باشد پوشه **Blocks** را باز کرده و در نیمه راست صفحه با دکمه سمت راست موس کلیک نمایید.



بـا اسـتفاده از منوی گشودنی که توسط دکمه راست موس باز شده است یک بلوک اطلاعاتی **(Data block)** ایجاد نمایید.

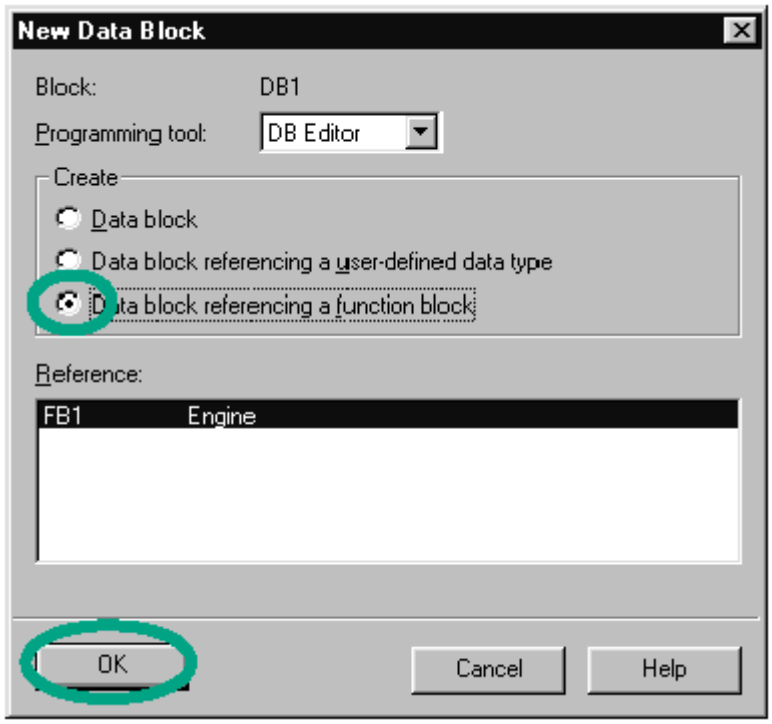


بـا کلـیک کـردن بـر روی **OK** تنظیمات نمایش داده شده در کادر مکالمه **" Properties "** را تائید نمایـید. بلـوک اطلاعاتی **DB1** به پروژه **" Getting Started "** افزوده می شود دوبار کلیک نمایید تا **DB1** باز شود.

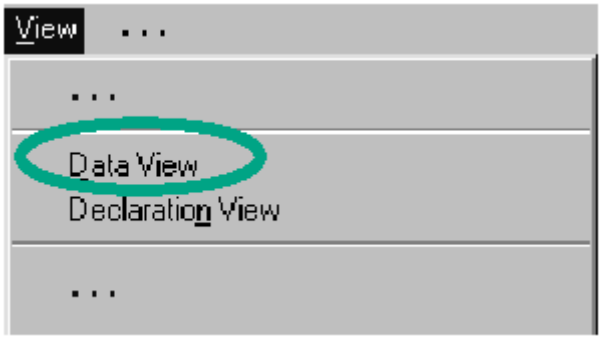


گزینه **Data Block Referencing a Function block** را در کادر مکالمه **New Data Block** فعال نمایید. انتساب " **FB1,Engine** "را توسط **OK** تائید نمایید.

پنجره برنامه **LAD/STL/FBD** با اطلاعات جدول نمایش متغیرهایی که متعلق به **FB1** میباشند باز می شود.

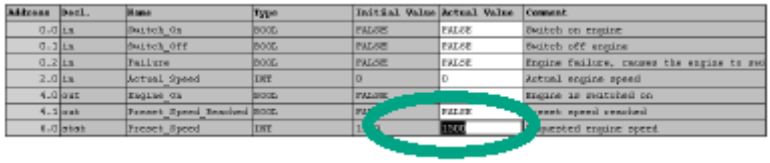


حال **DB1** میبایست اطلاعات مشخصه موتور بنزینی را در بر گیرد. شما میبایست این اطلاعات را وارد نمایید. نخست **Data View** را فعال نمایید.



سپس مقدار **1500** را برای موتور بنزینی در ستون مقدار واقعی (در ردیف **Setpoint – Speed**) وارد نمایید.

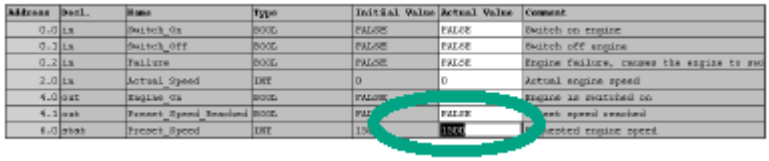
حال شما حداکثر سرعت این موتور را تعریف نموده اید.



DB1 را ذخیره نموده و پنجره برنامه را ببندید.

بهمین روشی که در مورد **DB1** انجام دادید ، بلوک اطلاعاتی دیگر **DB2** را برای **FB1** ایجاد نمایید.

حال مقدار واقعی **1200** را برای موتور دیزلی وارد نمایید.



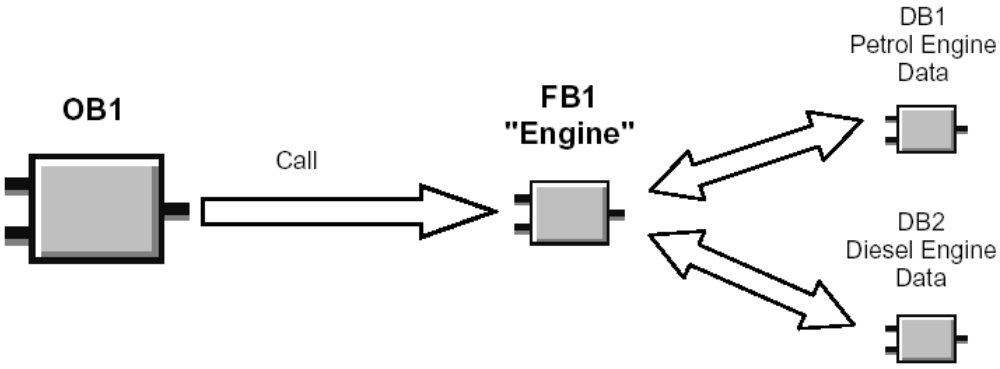
■ با تغییر مقادیر واقعی شما مقدمه چینی کنترل دو موتور تنها بااستفاده از یک بلوک تابع ساز را به انجام رسانیده اید. برای کنترل موتورهای بیشتر فقط میبایست بلوک های اطلاعاتی اضافی ایجاد نمایید.

کار دیگری که میبایست انجام دهید برنامه ریزی جهت فراخوانی بلوک تابع ساز در داخل **OB1** میباشد ، بدین منظور بسته به زبان برنامه نویسی که بر می گزینید برای منطق نردبانی بخش 5.6 ، برای نمایش عبارتی بخش 5.7 یا برای **Function Block Diagram,** بخش 5.8 را مطالعه نمایید.

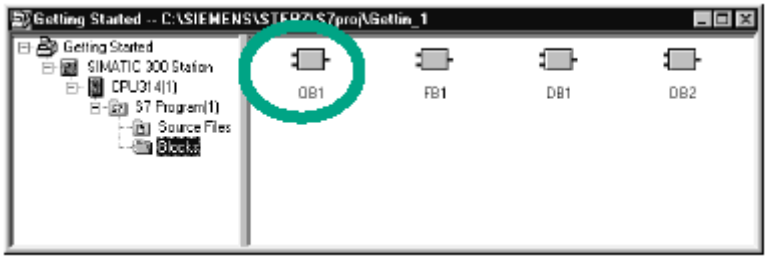
برای اطلاعات بیشتر می توانید به مباحث **Creating Data Blocks, Programming Blocks** واقع در **Help** > **Contents** مراجعه نمایید.

5.6 فراخوانی یک بلوک به زبان منطق نردبانی

کـل کارهایـی که در برنامه نویسی یک بلوک تابع ساز انجام داده اید در صورتی که این بلوک را در داخـل **OB1** فراخوانـی ننمایـید بـیفایده خواهـند بود. برای فراخوانی هر بلوک تابع ساز از یک بلوک اطلاعاتی استفاده می شود و بدین روش میتوانید هر دو موتور را کنترل نمایید.

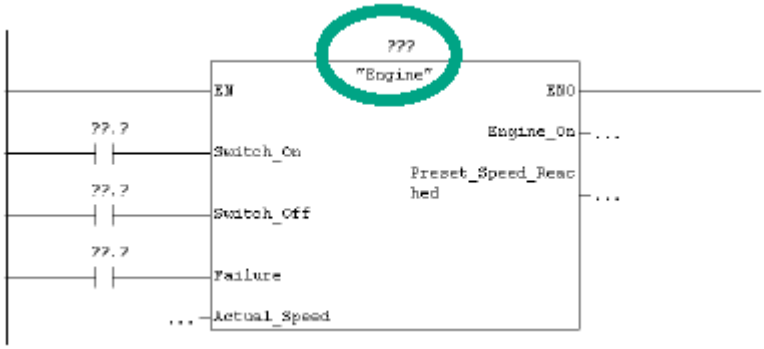


در حالـت ای کـه ، پـروژه **Getting Started** در داخـل پنجره **SIMATIC Manger** باز میباشد پوشه **Blocks** را گشوده و **OB1** را باز نمایید.

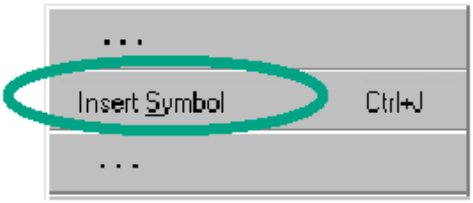


در پـنجره بـرنامه **LAD/STL/FBD** ، **Network 4** را ایجـاد نمایـید. سپس در کاتالوگ عناصر برنامه بلوک **FB1** را یافته و آنرا وارد نمایید.

یـک کنتاکـت حالـت عـادی بـاز را در جلـوی هـریک از عناویـن زیـر ایجـاد نمایـید : **Switch – off , Switch – on** و **Fault**.در بـالای **Engine** روی علامت ??.? کلیک نموده و سپس در همان نقطه در کادر **Input** با دکمه سمت راست موس کلیک نمایید.

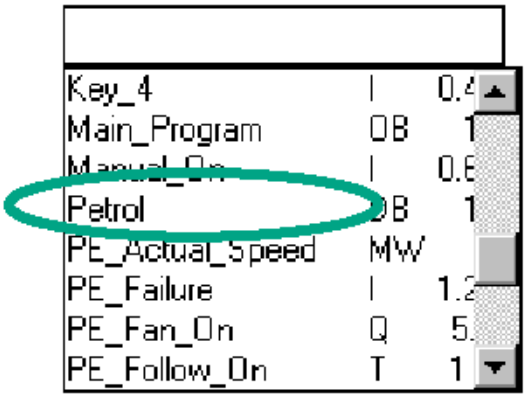


در منوی گشودنی ای که توسط کلیک راست موس ایجاد شده **Insert Symbol** را انتخاب نمایید. یک منوی کشویی نمایان می شود.

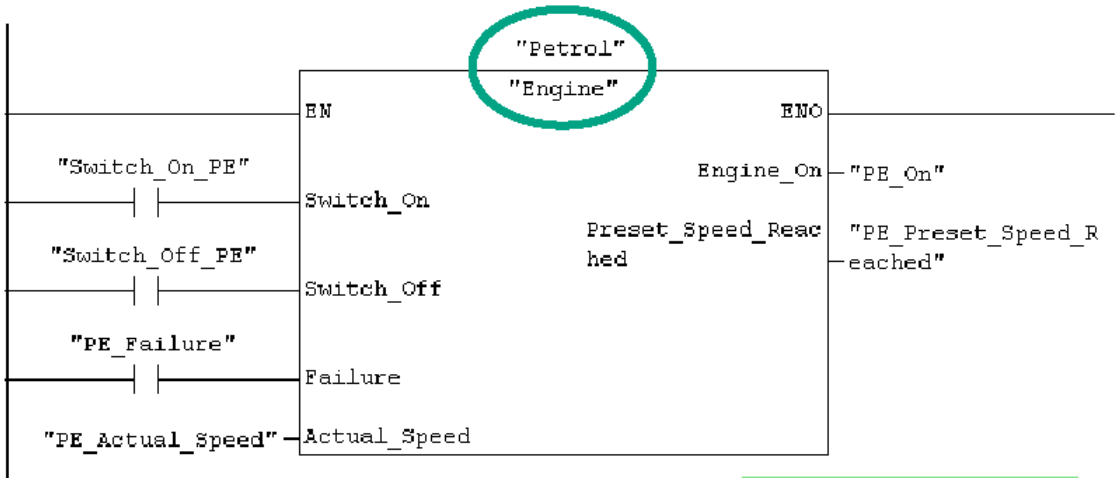


اگر این کار را برای اولین بار انجام میدهید ممکن است کمی نیاز به صرف وقت داشته باشید.

بر روی بلوک اطلاعاتی **Petrol** کلیک نمایید ، آنگاه این بلوک بطور خودکار در کادر ورودی در داخل علامت گیومه قرار می گیرد.



برای مابقی علامت های سئوال کلیک نموده و با استفاده از اسامی سمبلیک داخل منوی کشویی کلیه پارامترهای بلوک تابعی را آدرس دهی نمایید.

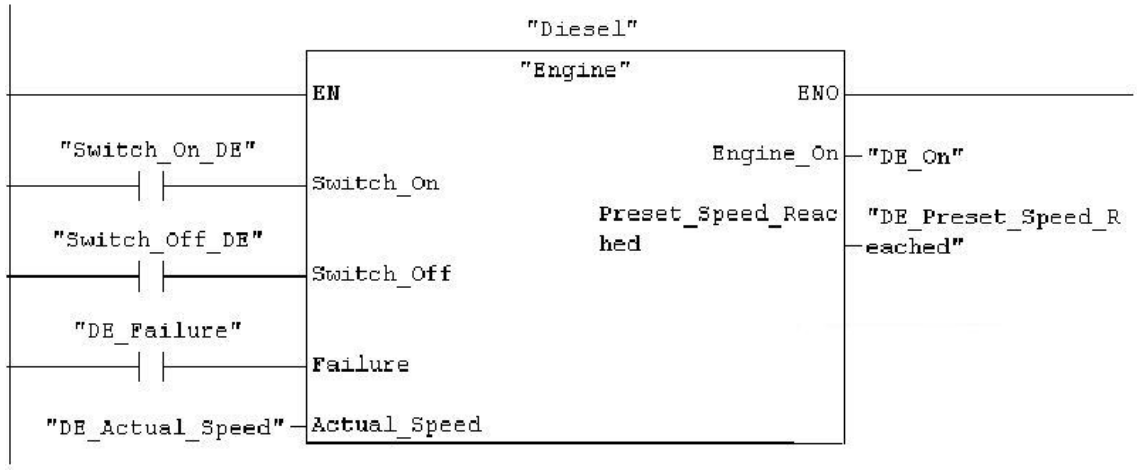


◙ متغیرهای ورودی – خروجی مشخصه موتور (که بصورت " **In** "و" **Out** " نمایش داده می شوند) در بلوک تابعی **Engine** نمایش داده می شوند.

یک سیگنال " **PE – xxx** " به هریک از متغیرهای موتور بنزینی نسبت داده می شود.

برنامه فراخوانی بلوک تابعی " **Engine**" (**FB1**) همراه بلوک اطلاعاتی " **Diesel**"

(**DB2**) را در یک **Network** جدید نوشته و آدرس های مربوطه را از منوی کشویی استخراج نمایید.



◙ یک سیگنال " **DE – xxx** " به هریک از متغیرهای موتور دیزلی نسبت داده می شود.

برنامه تان را ذخیره نموده و بلوک را ببندید.

هنگامیکه با بلوک های سازماندهی ، بلوک های تابعی و بلوک های اطلاعاتی ساختار برنامه را ایجاد می نمایید میبایست فراخوانی بلوک های رده پایینی (مانند **FB1**) را در داخل بلوک های رده بالاتر از آنها (بعنوان مثال **OB1**) انجام دهید.

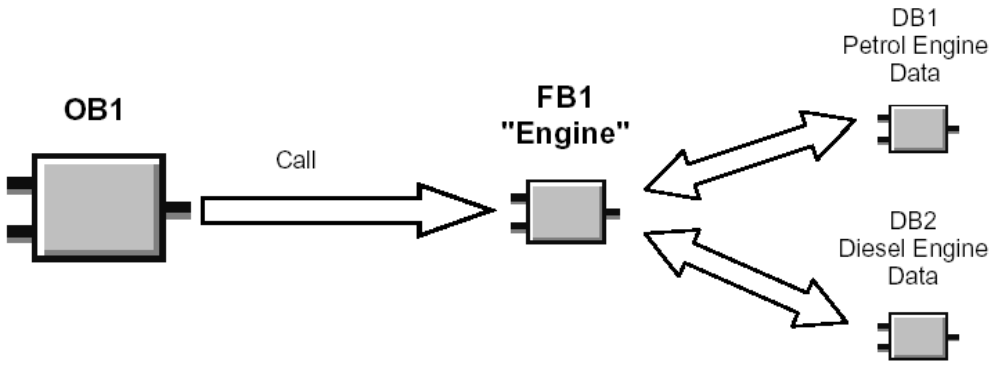
همچنین شما می توانید در داخل جدول سمبل ها به بلوک های مختلف ، نامهای سمبلیک را نسبت دهید. (بعنوان مثال برای **FB1** نام سمبلیک **" Engine "** و برای **DB1** نام سمبلیک **" Petrol "**) هر وقت بخواهید میتوانید بلوک های نوشته شده را چاپ نموده یا بصورت فشرده ذخیره نمایید.

عملیات ذکر شده را میتوانید در **SIMATIC Manager** تحت دستورات **File > Print** یا **File > Archive** انجام دهید.

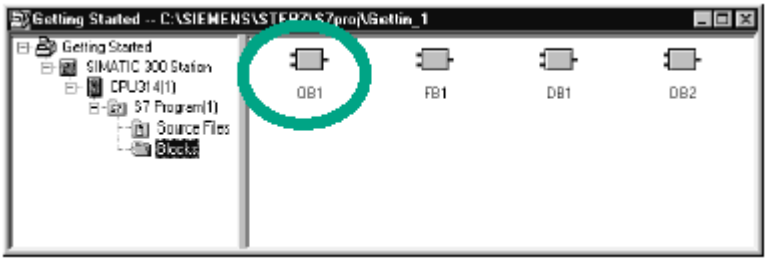
- برای اطلاعات بیشتر می توانید به مباحث **Calling Refrence Helps** ، **Language Description : LAD** و **Program Control Instructions** واقع در **Help > Contents** مراجعه نمایید.

5.7 فراخوانی یک بلوک به زبان نمایش عبارتی

کل کارهایی که در برنامه نویسی یک بلوک تابع ساز انجام داده اید در صورتی که این بلوک را در داخل **OB1** فراخوانی ننمایید، بیفایده خواهند بود. برای فراخوانی هر بلوک تابع ساز از یک بلوک اطلاعاتی استفاده می شود و بدین روش می توانید هر دوموتور را کنترل نمایید.



در حالتی که پروژه **Getting Started** درداخل پنجره **SIMATIC Manager** باز میباشد پوشه **Blocks** را گشوده و **OB1** را باز نمایید.



در پنجره برنامه **LAD/STL/FBD** , **Network 4** را ایجاد نمایید.

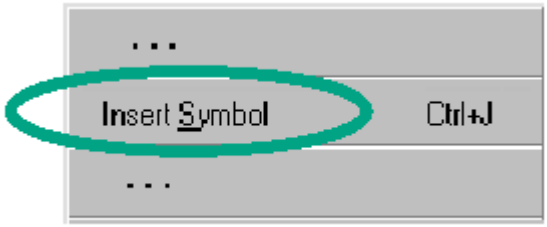
در بخش کدها " **Petrot** " و " **Engine** " **Call** را وارد نموده و سپس **Enter** نمایید.

کلیه پارامترهای بلوک تابعی " **Petrol** " به نمایش در می آیند.

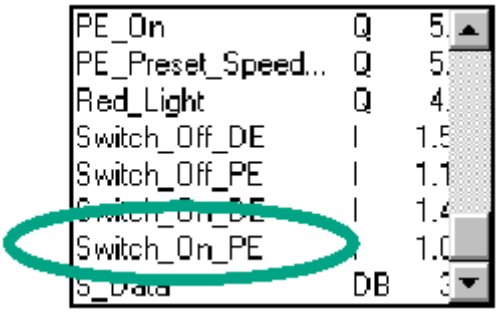
```
CALL "Engine" , "Petrol"
  Switch_On              :=
  Switch_Off             :=
  Failure                :=
  Actual_Speed           :=
  Engine_On              :=
  Preset_Speed_Reached:=
```

مکان نما را بعد از علامت تساوی (=) مربوط به **Switch – on** قرار داده و دکمه سمت راست موس را کلیک نمایید.

در منوی گشودنی ای که با کلیک راست موس ایجاد می شود **Insert Symbol** را انتخاب نمایید. یک منوی کشویی نمایان می شود. اگر این کار را برای اولین بار انجام میدهید ممکن است کمی نیاز به صرف وقت داشته باشد.



بر روی نام **Switeh – on – PE** کلیک نمایید. این نام از داخل منوی کشویی استخراج شده و بصورت خودکار در داخل گیومه قرار خواهد گرفت.



با استفاده از منوی کشویی کلیه آدرس ها ی مورد نیاز را به متغیرهای بلوک تابعی نسبت دهید.

```
CALL  "Engine" , "Petrol"
 Switch_On           :="Switch_On_PE"
 Switch_Off          :="Switch_Off_PE"
 Failure             :="PE_Failure"
 Actual_Speed        :="PE_Actual_Speed"
 Engine_On           :="PE_On"
 Preset_Speed_Reached:="PE_Preset_Speed_Reached"
```

◙ یک سیگنال " **PE – xxx** " به هریک از متغیرهای موتور بنزینی منتسب می گردد.

برنامه فراخوانی بلوک تابعی " **Engine** " **(FB1)** بهمراه بلوک اطلاعاتی " **Diesel** " **(DB2)**

را در داخل یک **Network** جدید بنویسید و بصورت مشابه برای سایر فراخوانی ها عمل نمایید.

```
CALL  "Engine" , "Diesel"
 Switch_On           :="Switch_On_DE"
 Switch_Off          :="Switch_Off_DE"
 Failure             :="DE_Failure"
 Actual_Speed        :="DE_Actual_Speed"
 Engine_On           :="DE_On"
 Preset_Speed_Reached:="DE_Preset_Speed_Reached"
```

برنامه تان را ذخیره نموده و بلوک را ببندید.

◙ هنگامیکه با بلوک های سازماندهی ، بلوک های تابعی و بلوک های اطلاعاتی ساختار برنامه را ایجاد می نمایید میبایست فراخوانی بلوک های رده پایینی (مانند **FB1**) را در داخل بلوک های رده بالاتر از آنها (بعنوان مثال **OB1**) انجام دهید.

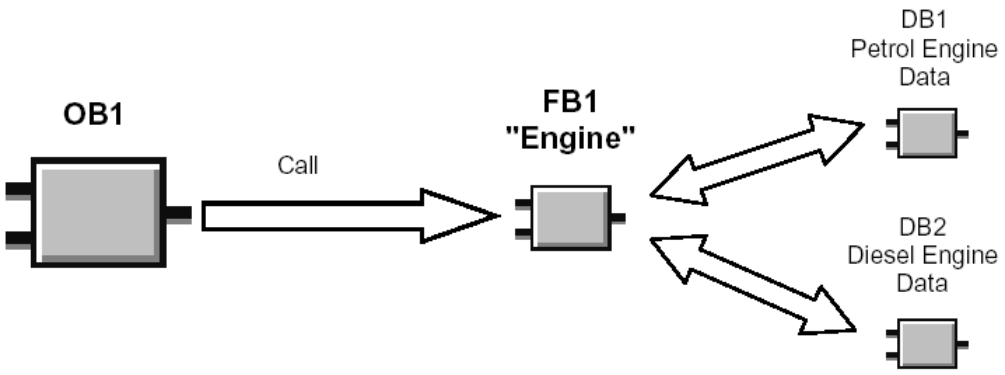
همچنین شما می توانید در داخل جدول سمبل ها به بلوک های مختلف ، نامهای سمبلیک را نسبت دهید. (بعنوان مثال برای **FB1** نام سمبلیک " **Engine** " و برای **DB1** نام سمبلیک " **Petrol** ") هر وقت بخواهید میتوانید بلوک های نوشته شده را چاپ نموده یا بصورت فشرده ذخیره نمایید.

عملیات ذکر شده را میتوانید در **SIMATIC Manager** تحت دستورات **File > Print** یا **File > Archive** انجام دهید.

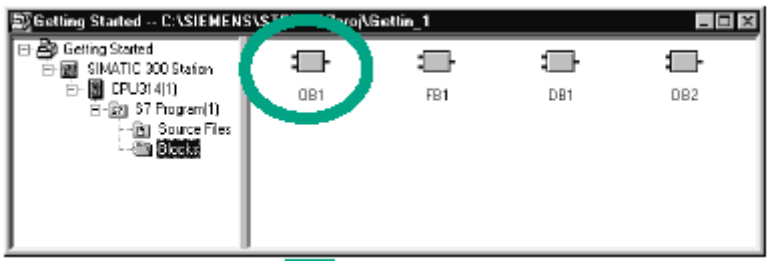
■ برای اطلاعات بیشتر می توانید به مباحث **Calling Refrence Helps** ، **STL : Language Description** و **Program Control Instruction** واقع در **Help > Contents** مراجعه نمایید.

5.8 فراخوانی نمودن یک بلوک به زبان FBD

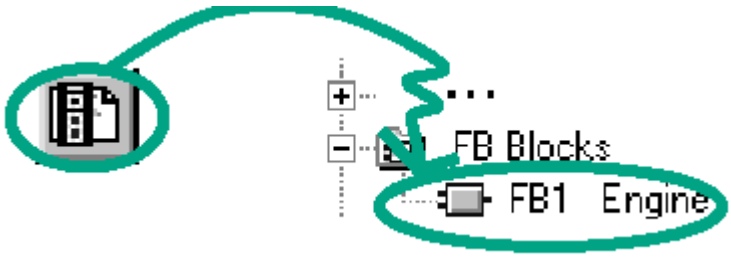
کل کارهایی که در برنامه نویسی یک بلوک تابع ساز انجام داده اید در صورتی که این بلوک را در داخل **OB1** فراخوانی ننمایید، بیفایده خواهند بود. برای فراخوانی هر بلوک تابع ساز از یک بلوک اطلاعاتی استفاده می شود و بدین روش می توانید هر دو موتور را کنترل نمایید.



در حالتی که پروژه **Getting Started** در داخل پنجره **SIMATIC Manager** باز می باشد پوشه **Blocks** را گشوده و **OB1** را باز نمایید.

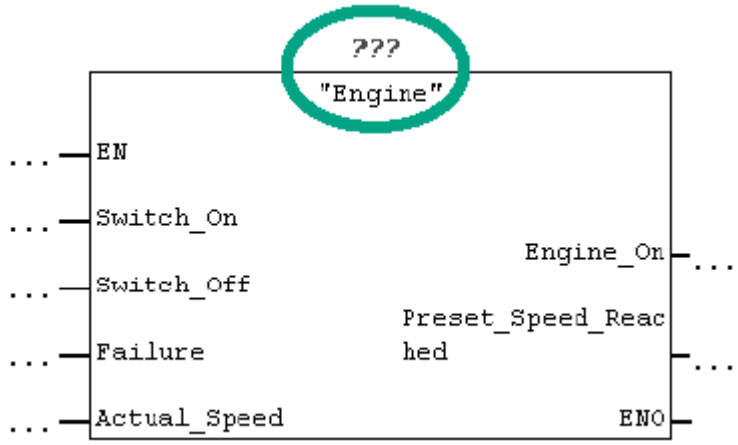


در پنجره **LAD/STL/FBD** , **Network4** را ایجاد نمایید. سپس در داخل کاتالوگ عناصر برنامه، **FB1** را یافته و آنرا وارد نمایید.



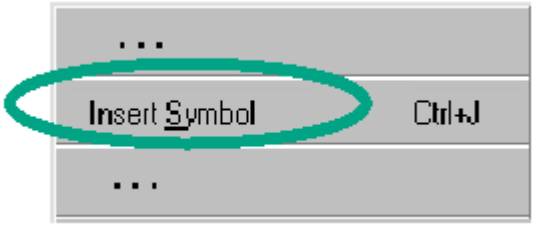
کلیه متغیرهای ورودی و خروجی مشخصه موتور به نمایش در می آیند.

بـر روی علامـت ??.? در بـالای **Engine** کلـیک نموده و سپس در حالیکه مکان نما را در همان مکان نگاهداشته اید در داخل کادر ورودی کلیک راست نمایید.

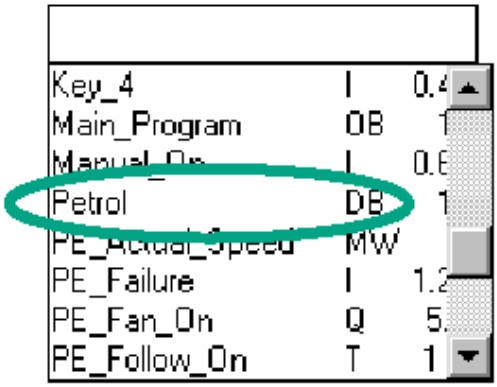


بااستفاده از کلیک راست در داخل منوی گشودنی گزینه **insert Symbol** را انتخاب نمایید.

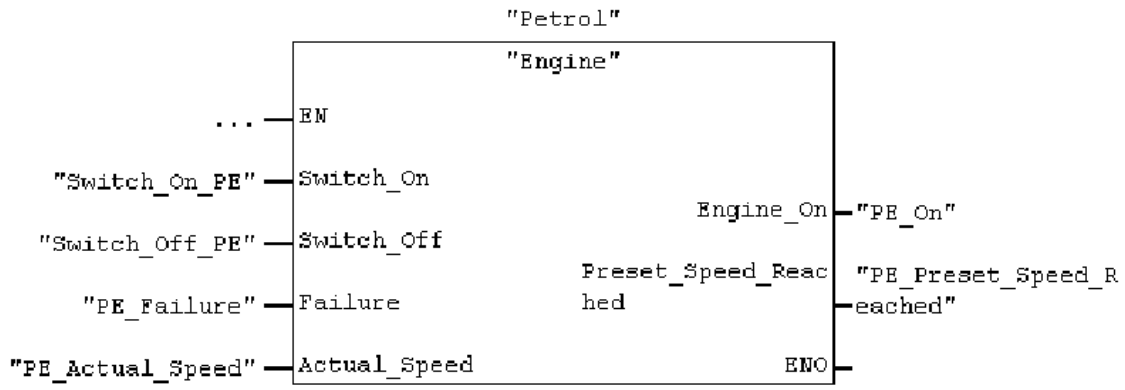
یـک مـنوی کشـوئی پدیـدار خواهد شد. اگر این کار را برای اولین بار انجام میدهید ممکن است کمی نیاز به صرف وقت داشته باشید.



بـر روی بلـوک اطلاعاتی **Petrol** کلیک نمایید تا از داخل منوی کشویی انتخاب شده و بطور خودکار در داخل کادر ورودی درون گیومه قرار گیرد.

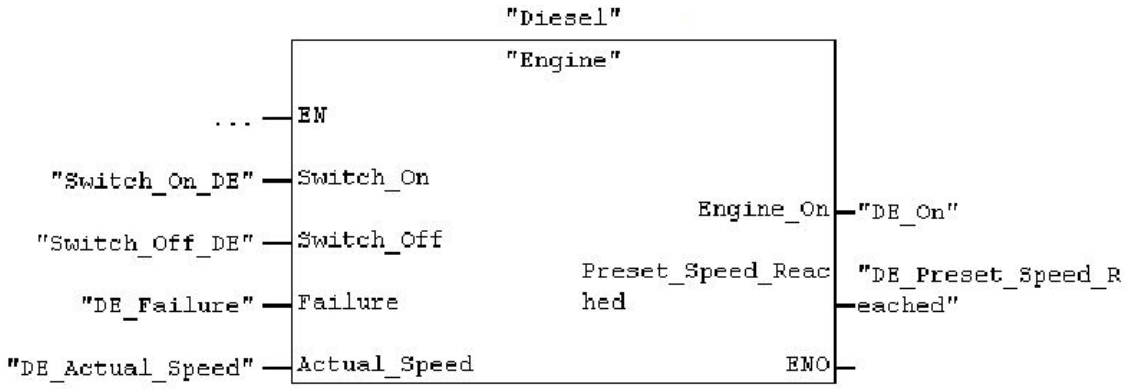


بااستفاده از منوی کشویی کلیه آدرس های مورد نیاز را به سایر پارامترهای بلوک تابعی نسبت دهید.



- یک سیگنال " **PE – xxx** " به هریک از متغیرهای موتور بنزینی منتسب می گردد.

برنامه فراخوانی بلوک تابعی " **Engine** " **(FB1)** بهمراه بلوک اطلاعاتی " **Diesel** " **(DB2)** را در داخل یک **Network** جدید بنویسید و آدرس های مربوطه را از منوی کشویی برگزینید.



- یک سیگنال " **PE - xxx** " به هریک از متغیرهای موتور دیزلی منتسب می گردد.

هنگامیکه با بلوک های سازماندهی ، بلوک های تابعی و بلوک های اطلاعاتی ساختار برنامه را ایجاد می نمایید میبایست فراخوانی بلوک های رده پایینی (مانند **FB1**) را در داخل بلوک های رده بالاتر از آنها (بعنوان مثال **OB1**) انجام دهید.

همچنین شما می توانید در داخل جدول سمبل ها به بلوک های مختلف ، نامهای سمبلیک را نسبت دهید. (بعنوان مثال برای **FB1** نام سمبلیک **" Engine "** و برای **DB1** نام سیمبلیک **" Petrol "**) هر وقت بخواهید میتوانید بلوک های نوشته شده را چاپ نموده یا بصورت فشرده ذخیره نمایید.

عملیات ذکر شده را میتوانید در **SIMATIC Manager** تحت دستورات **File > Print** یا **File > Archive** انجام دهید.

برای اطلاعات بیشتر می توانید به مباحث **Calling Refrence Helps** ، **Language Description : FBD** و **Program Control Instruction** واقع در **Help > Contents** مراجعه نمایید.

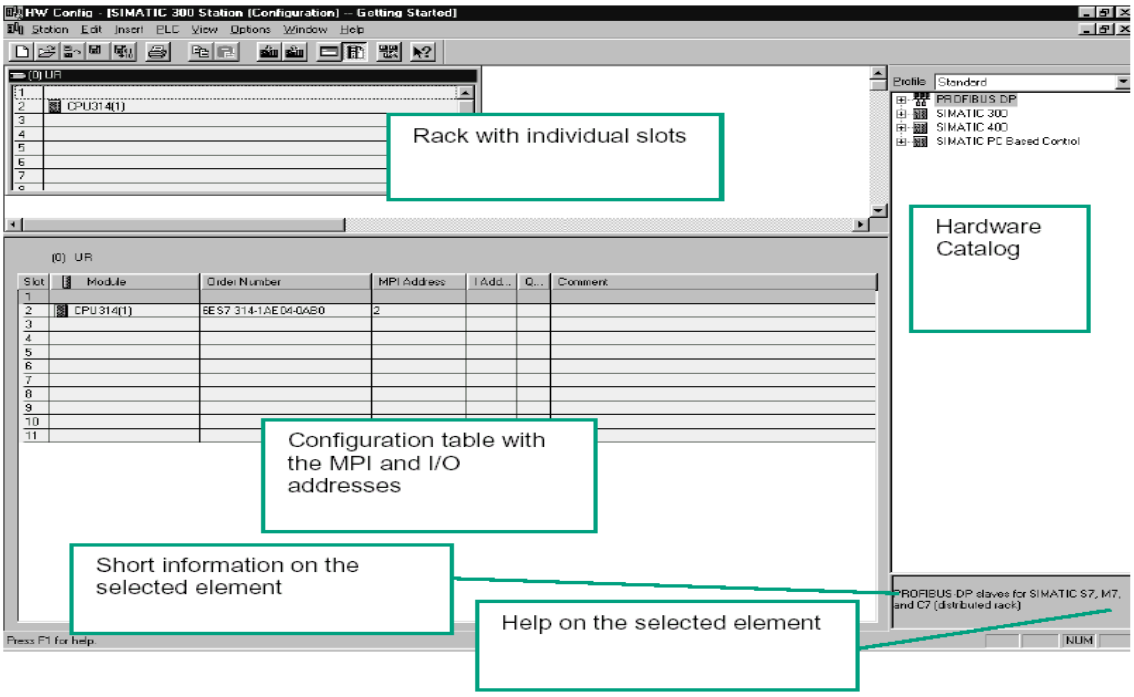
6.1 پیکربندی سخت افزار

شما در ابتدای ایجاد یک پروژه توسط ایستگاه SIMATIC میتوانید به پیکربندی سخت افزارتان اقدام نمایید. ساختار پروژه ای که توسط جادوگر STEP7 در بخش 2.1 ایجاد نمودید تمام این کارها را در بر می گیرد.

هنگامیکه توسط STEP7 سخت افزار را پیکربندی نمودید با استفاده از " Download" می توانید اطلاعات مربوطه را به PLC منتقل نمایید (فصل 7 را مشاهده کنید)

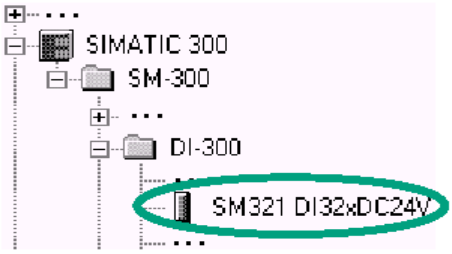
برای شروع پروژه Getting Started را درداخل SIMATIC Manager باز نمایید.

پوشه SIMATIC 300 Station را باز نموده و بر روی سمبل Hardware دوبار کلیک نمایید. پنجره " HW Config " باز می شود. CPU ای که هنگام ایجاد پروژه انتخاب نموده بودید نمایش داده می شود که برای پروژه " Getting Started " CPU 314 میباشد.

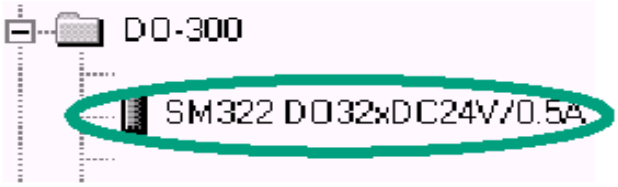


اولین چیزی که مورد نیازتان میباشد یک کارت منبع تغذیه میباشد. درداخل کاتالوگ آنقدر پیش بروید تا به **PS307 2A** برسید و آنرا برداشته و درداخل شیار 1 رها نمایید **(Drag & Drop)** .

حال در داخل کاتالوگ جستجو نموده تا کارت ورودی (**DI** / ورودی دیجیتال) **SM 321 DI32 × DC 24V** را یافته و آنرا در داخل شیار 4 وارد نمایید. شیار 3 خالی باقی می ماند.



بهمین روش کارت خروجی **SM322 DO32×DC 24V/0.5A** را در شیار 5 وارد نمایید.

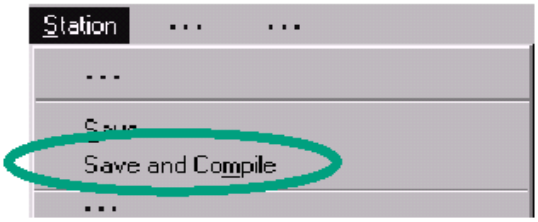


در صـورتی کـه قصد داشته باشید پارامترهای (بعنوان مثال ، آدرس) یک کارت را درون پروژه ای تغییر دهید بر روی کارت مذکور دوبار کلیک نمایید. هرچند توجه داشته باشید که فقط در صورتیکه از نـتایج حاصـله از تغییراتی که میدهید بر روی **PLC** اطلاع دقیق دارید مبادرت به تغییر پارامترها ورزید.

Slot	Module	Order Number	MPI Address	I Add...	Q...	Comment
1	PS307 2A	6ES7 307-1BA00-0AA0				
2	CPU314(1)	6ES7 314-1AE04-0AB0	2			
3						
4	DI32xDC24V	6ES7 321-1BL00-0AA0		0...3		
5	DO32xDC24V/0.5A	6ES7 322-1BL00-0AA0			4...7	
6						
7						
8						
9						
10						
11						

در پروژه **Getting Started** هیچ نیازی به تغییرت در این زمینه ندارید.

با اجرای فرمان **Save and compile** اطلاعات آماده انتقال به **CPU** میباشند.



هنگامیکه برای اولین بار **Application** ، **" HW Config "** را می بندید سمبل **System data** در پوشه بلوک ها پدیدار می گردد.

- شـما مـی توانـید با استفاده از دستور **Station > Consistency check** خطاهای موجود در پیکربندی تان را بررسی نمایید. **STEP7** برای هر خطایی که ممکن است بروز نماید یک سری راه حل های عملی ارائه میدهد.
- بـــرای اطلاعـــات بیشـــتر مـــی توانـــید بـــه مباحـــث **Cinfiguring the hardware** و **Configuring Centrals Racks** واقع در **Help > Contents** مراجعه نمایید.

7. Downlouad نمودن و غلط گیری برنامه

7.1 برقراری یک ارتباط زنده

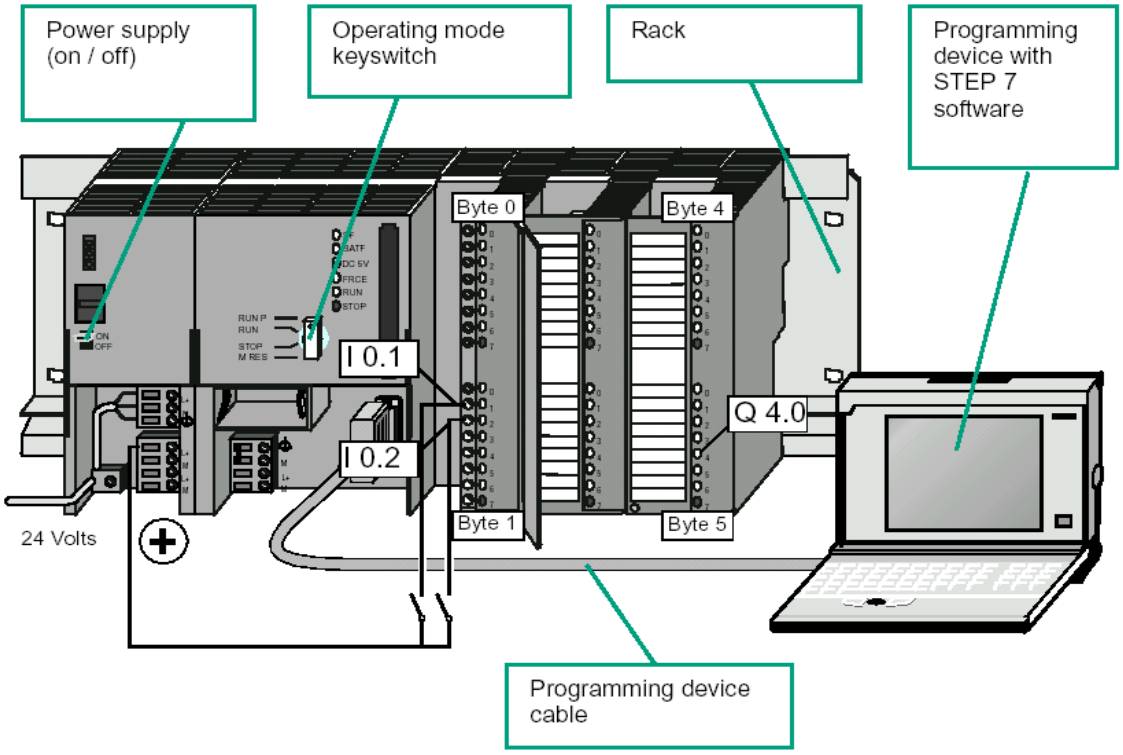
بـــا اسـتفاده از پـروژه موجـود **GS - LAD – Example** یـا پـروژه **Getting started** کـه خودتـان ایجـاد نمـوده ایـد و یک پیکربندی امتحانی ساده به شما نشان خواهیم داد چگونه برنامه را به کنترل کننده منطقی قابل برنامه ریزی **(PLC)** ، **Download** نموده و سپس آنرا غلط گیری نمایید.

شما میبایست:

- سخت افزار پروژه **" Getting Started "** را پیکربندی نموده باشید. (فصل 6 را مطالعه نمایید.)
- سخت افزار را براساس راهنمای نصب ، **Setup** نموده باشید.

مثال یک مدار سری (تابع **AND**) :

خروجـی **Q4.0** نمـی بایسـت روشـن شود (دیود **Q4.0** بر روی کارت خروجی دیجیتال روشن می شـود) مگـر آنکه هر دو کلید **I0.1** و کلید **I0.2** فشرده شوند. پیکربندی امتحانی زیر را با استفاده از مقداری سیم و **CPU** تان انجام دهید.



پیکربندی سخت افزار

برای سوار کردن **(Assemble)** یک کارت بر روی ریل بروش زیر عمل کنید :

- کانکتور **Bus** را در پشت کارت قرار دهید.
- کارت را از ریل آویخته و بطرف پایین بچرخانید.
- کارت را در جای خود پیچ نمایید.
- کارتهای باقیمانده را سوار نمایید.
- هنگامیکه کلیه کارتها رادر جای خود نصب نمودید کلید را در داخل **CPU** وارد نمایید.

حتی اگر شما از سخت افزاری متفاوت از آنچه که در شکل نمایش داده شده است استفاده نمایید نیز می توانید کار را انجام دهید. تنها لازمست که ورودی ها و خروجی ها را آدرس دهی نمایید.

STEP7 روش های مختلفی را جهت غلط گیری برنامه تان در اختیار شما قرار میدهد بعنوان مثال استفاده از **Program Status** یا اینکه از طریق جدول متغیرها.

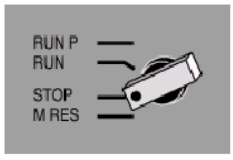
برای کسب اطلاعات بیشتر راجع به پیکربندی **Rack** مرکزی می توانید به کتابچه های **S7-400 / M7–400–Hardware** و **S7–300 Hardware and installation/Module specification** مراجعه نمایید.

7.2 Downloud نمودن برنامه به PLC

برای **Downloud** نمودن برنامه میبایست یک ارتباط **Online** ایجاد نموده باشید.

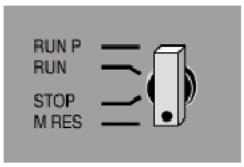
منبع تغذیه را با استفاده از کلید **ON / OFF** روشن نمایید. دیود نورانی **" DC 5V "** بر روی **CPU** روشن می گردد.

کلید تعیین وضعیت کاری را (اگر هم اکنون در وضعیت **STOP** قرار ندارد) بر روی موقعیت **STOP** قرار دهید. **LED** قرمز رنگ **STOP** روشن خواهد شد.



ری ست نمودن CPU و برگردانیدن آن به وضعیت RUN

کلید تعیین وضعیت را به موقعیت **MRES** چرخانیده و حداقل بمدت 3 ثانیه نگاهداشته تا **LED** قرمز رنگ **STOP** به آرامی شروع به چشمک زدن نماید.



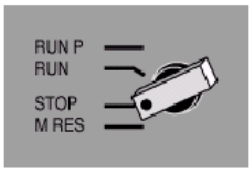
کلید را رها نموده و حداکثر بعد از 3 ثانیه مجدداً آن را به وضعیت **MRES** بچرخانید. هنگامیکه **LED** مربوط به **STOP** بسرعت چشمک بزند. **CPU** ری ست شده است.

در صورتیکه **LED** مربوط به **STOP** بسرعت چشمک نزند مراحل فوق را تکرار نمایید.

■ ری ست نمودن حافظه کلیه اطلاعات موجود بر روی **CPU** را خذف نموده و **CPU** را در وضعیت اولیه قرار میدهد.

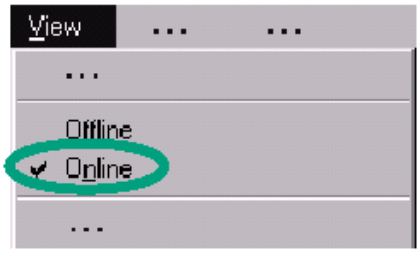
Downloud نمودن برنامه به CPU

حال کلید تعیین وضعیت را مجدداً به موقعیت **STOP** برگردانیده تا برنامه را **Download** نمایید.



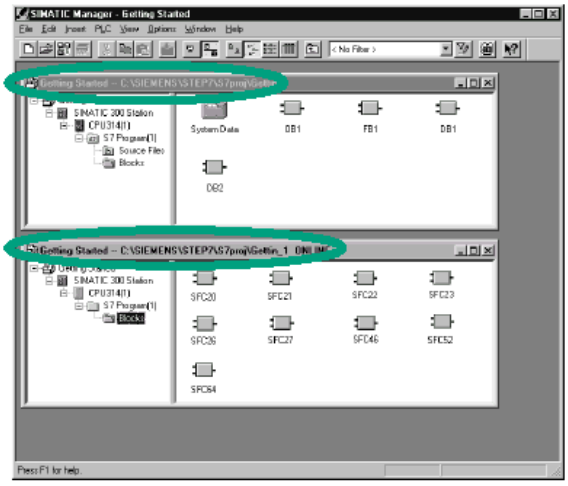
برنامه **SIMATIC Manager** را اجراءنموده و پروژه **Getting Started** را (در صورتیکه هنوز باز ننموده اید) از طریق کادر مکالمه **" Open "** باز نمایید.

علاوه بر پنجره **" Getting Started Offline "** پنجره **" Getting Started Online "** را نیز باز نمایید.



وضعیت **Online** یا **Offline** توسط رنگ های متفاوت هدرها مشخص می گردد.

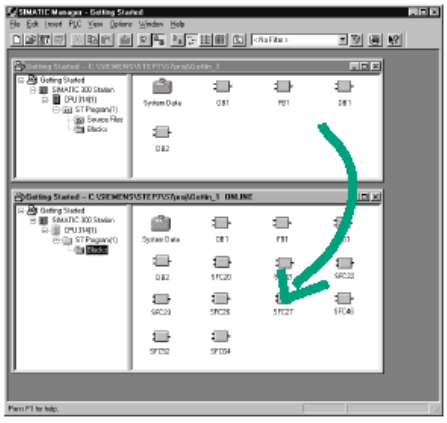
در هر دو پنجره پوشه **Block** را یافته و باز نمایید. پنجره **Offline** وضعیت ابزار برنامه نویسی (**PG** یا **PC**) و پنجره **Online** وضعیت **CPU** را نمایش میدهند.



◙ توابـع سیسـتمی (**SFC** هـا) تا زمانیکه حافظه را ری ست ننمایید در **CPU** باقی می مانند. این توابع که به سیستم عـامل مـربوط می باشند توسط **CPU** ایجاد می شوند. این توابع قابل پاک کردن نبوده و همچنین نمی بایست آنها را **Download** نمود.

پوشـه **Block** را در پـنجره **Offline** انـتخاب نموده و سپس با استفاده از فرمان **PLC>Downloud** برنامه را به **CPU**، **Download** نمایید. اعلان را با استفاده از **OK** تائید نمایید.

بلوک های برنامه پس از **Download** نمودن آنها در پنجره **Online**نمایش داده می شوند.



◙ شـما همچنیـن مـی توانـید بـا اسـتفاده از دکمـه هـای مربوطه واقع خط ابزار یا توسط کلیک راست موس از منوی گشودنی ، فرمان **PLC>Downloud** را صادر نمایید.

روشن نمودن CPU و بررسی وضعیت کاری

کلید تعیین وضعیت کاری را به **RUN-P** بچرخانید.

نمایشگر سبز رنگ **" RUN "** روشن شده نمایشگر قرمز **" STOP "** خاموش می شود. حال **CPU** آماده کار میباشد.

زمانیکه نمایشگر سبز روشن شود میتوانید آزمایش برنامه را شروع نمایید.

اگر نمایشگر قرمز رنگ روشن ماند، خطایی رخ داده است. در آنصورت میبایست بافر خطا را بمنظور یافتن خطا بررسی نمایید.

Download نمودن بلوک ها بطور جداگانه

درعمل برای اینکه خطا سریعتر بر طرف شود ، بلوک ها می توانند بطور جداگانه و با استفاده از عمل کشیدن و رها کردن **(Drag and Drop)** به **CPU** منتقل شوند. برای **Download** نمودن بلوک ها کلید تعیین وضعیت کاری بر روی **CPU** میبایست در یکی از وضعیت های **" RUN – P "** یا **" Stop "** باشد. بلوک هایی که در وضعیت **" Run - P "** شوند نور افعال می گردند.**Download**

لذا می بایست بخاطر داشته باشید :

- در صورتیکه بلوک های بدون خطا با بلوک های خطا دار مورد بازنویسی **(Over written)** قرار گیرند این عمل به خطای سیستم کنترل منجر خواهد شد. شما می توانید با امتحان نمودن بلوک هایتان قبل از **Down load** نمودن آنها از این مساله جلوگیری نمایید.
- اگر شما در **Download** نمودن بلوک ها ، بترتیب عمل ننمائید – ابتدا بلوک های رده پایین و سپس بلوک های رده بالاتر – **CPU** به وضعیت **" STOP "** خواهد رفت.

شما می توانید با **Download** نمودن کل برنامه به **CPU** از این مورد جلوگیری نمایید.

برنامه نویسی بصورت Online

در عمـل ممکن است نیاز پیدا کنید تا بلوک هایی را که به **Download CPU** نموده اید را تغییر دهید. بدین منظور در پـنجره **Online** بـر روی بلـوک مـورد نظـر دوبار کلیک نموده تا پنجره برنامه **LAD/STL/FBD** باز شود. سپس بروش معمول برنامه بلوک را تغییر دهید. توجه نمایید که بلوک تغییر داده شده بلافاصله در **CPU** فعال می گردد.

برای اطلاعات بیشتر می توانید به مباحث

" Establishing an Online connection and Making CPU Setting "و

"Downloading from the PG/PC to the programming controller " واقـــع در

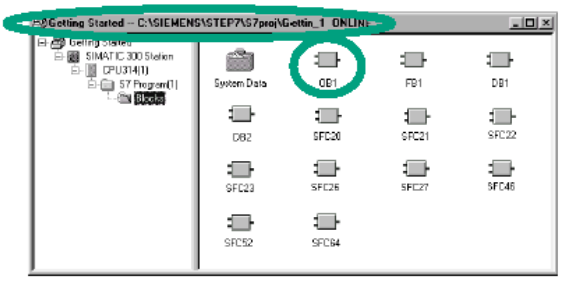
Help > Contents مراجعه نمایید.

7.3 تست برنامه توسط Status گیری

با استفاده از عمل **Status** گیری از برنامه می توانید برنامه یک بلوک را تست نمایید. لازمه این عمل آنسـت کـه یـک ارتـباط **Online** را بـا **CPU** برقرار نموده ، **CPU** دروضعیت **RUN** یا **RUN – P** بوده و برنامه **Down load** شده باشد.

OB1 را در پنجره پروژه **" Getting Started Online "**باز نمایید.

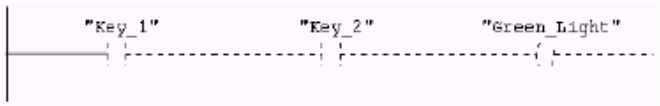
پنجره برنامه **LAD/STL/FBD** باز می شود.



تابع **Monitor** > **Debug** را فعال نمایید.

غلط گیری بزبان منطق نردبانی

مـدار سری واقع در **Network 1** بزبان منطق نردبانی نمایش داده شده است. مسیر مدار تا **Key-1 (I0.1)** با یک خط توپر نمایش داده شده است ، این بدان معنی است که برق به مدار اعمال شده است.



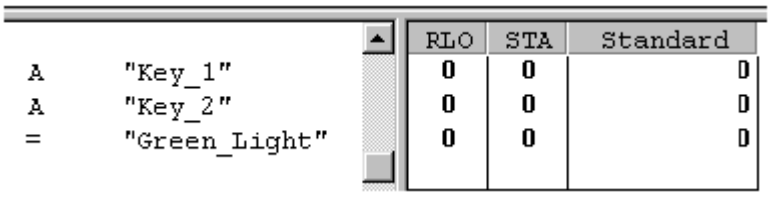
غلط گیری بزبان (Function Block Diagram).

وضـعیت سـیگنال هـا بـا **0** و **1** نمـایش داده شـده اسـت. خـط نقطـه چیـن بـدان معنـی اسـت نتیجه ای از عملیات منطقی حاصل نشده است.

غلط گیری بزبان نمایش عبارتی

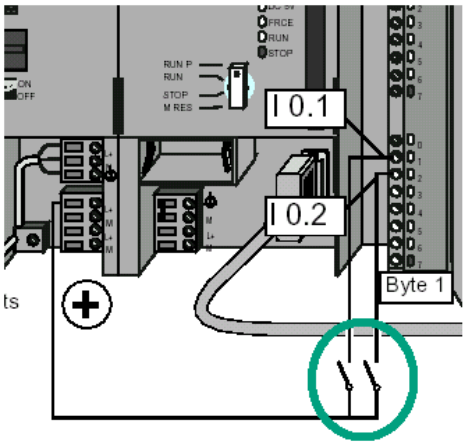
برای زبان نمایش عبارتی نتایج زیر بصورت جدولی نمایش داده می شود.

- نتیجه عملیات منطقی **(RLO)**
- بیت **Status (STA)**
- **Status** استاندارد **(STANDARD)**



- با استفاده از **Options > Customize** میتوانید زبان برنامه نویسی را که هنگام تست نمایش داده می شود را تغییر دهید.

حال هر دو کلید سخت افزار بسته شده را فشار دهید.

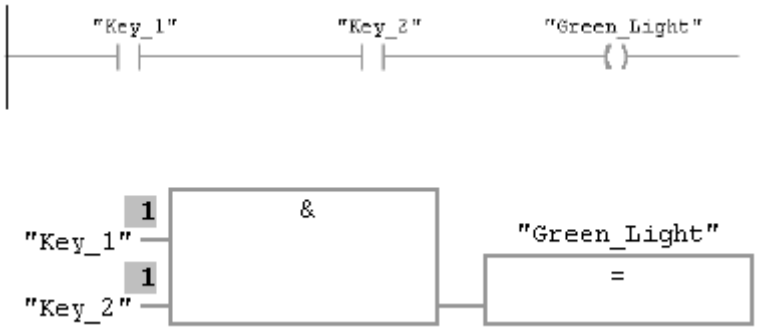


دیودهای ورودی **I0.1** و **I0.2** بر روی کارت ورودی روشن می شوند.

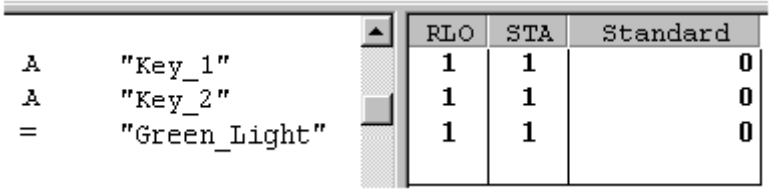
دیود خروجی **Q4.0** بر روی کارت خروجی نیز روشن می شود.

در زبانهای برنامه نویسی گرافیکی منطق نردبانی و **Function Block Diagram** شما می توانید با استفاده از تغییر در رنگ **Network** نوشته شده نتیجه تست را پیگیری نمایید.

این تغییر رنگ نشان میدهد که نتیجه عملیات منطقی تا این نقطه برآورده شده است.

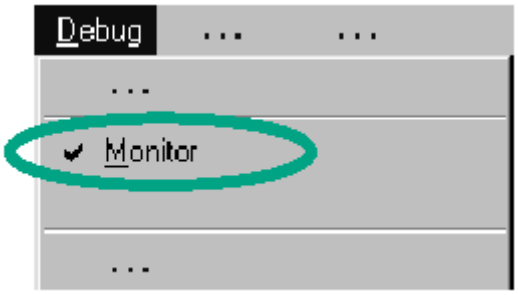


در زبان برنامه نویسی نمایش عبارتی محتویات ستونهای **STA** و **RLO** در اثر برآورده شدن نتیجه عملیات منطقی تغییر می کند.



		RLO	STA	Standard
A	"Key_1"	1	1	0
A	"Key_2"	1	1	0
=	"Green_Light"	1	1	0

تابع **Monitor** > **Debug** را غیر فعال ساخته و پنجره را ببندید.



سپس پنجره **Online** را در **SIMATIC Manager** ببندید.

توصیه می شود که بمنظور اجرای برنامه های موجود آنها را یکجا به **CPU** ، **Download** ننمایید ، زیرا در آنصورت بخاطر تعدد منابع ایجاد خطا ، بطرف کردن منشاء بروز خطا مشکل تر خواهد بود. در عوض برای دستیابی به نتیجه بهتر میبایست بلوک ها را بطور مجزا **Download** نموده و سپس آنها را تست نمایید.

برای اطلاعات بیشتر میتوانید به مباحث "**Debugging**" و "**Testing with program status**" واقع در **Help>Contents** مراجعه نمائید.

4.7 امتحان نمودن برنامه توسط جدول متغیرها

شما می توانید متغیرهای برنامه را بطور مجزا از طریق مونیتور و اصلاح نمودن ، تست نمایید.

لازمه این کار آنست که یک ارتباط **Online** با **CPU** برقرار نموده ، **CPU** دروضعیت **RUN-P** بوده و برنامه **Download** شده باشد.

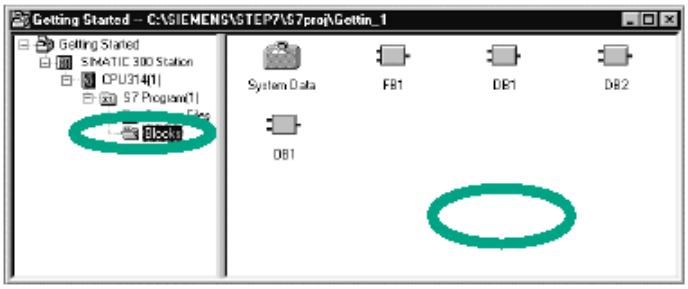
همزمان با تست توسط وضعیت برنامه **(Program status)** شما می توانید در جدول متغیرها ورودیها و خروجیها را در **Network1** (مدار سری یا تابع **AND**) مونیتور نمایید.

شما بهمین صورت می توانید مقایسه کننده سرعت موتور در **FB1** را با تنظیم **(presseting)** نمودن سرعت واقعی ، تست نمایید.

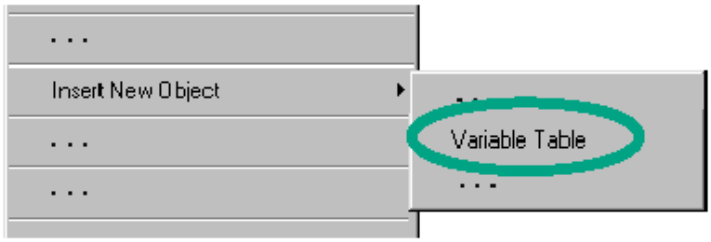
ایجاد جدول متغیرها

نقطه شروع مجدداً پنجره باز " **Getting Started Offline**" در **SIMATIC Manager** میباشد.

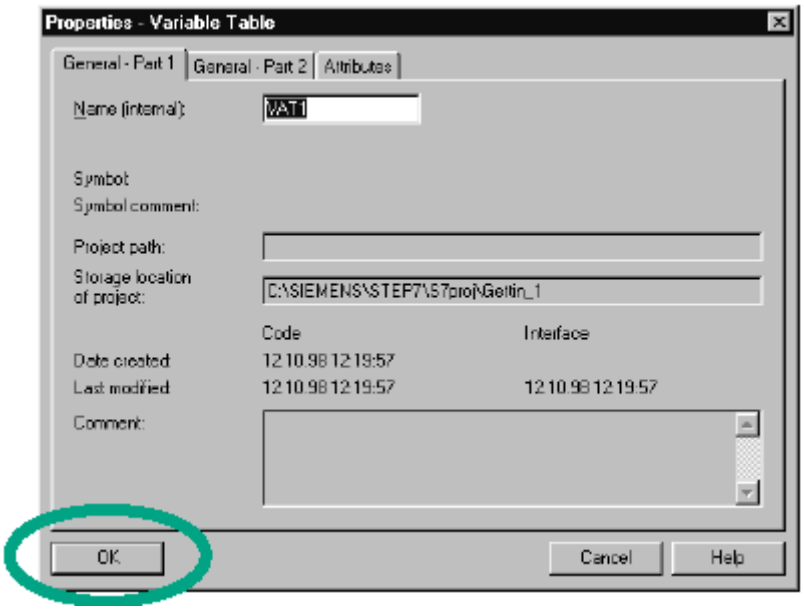
بسراغ پوشه **Blocks** رفته و در نیمه راست پنجره کلیک راست نمایید.



با استفاده از دکمه راست موس **Variable Table** را از داخل منوی گشودنی وارد نمایید.

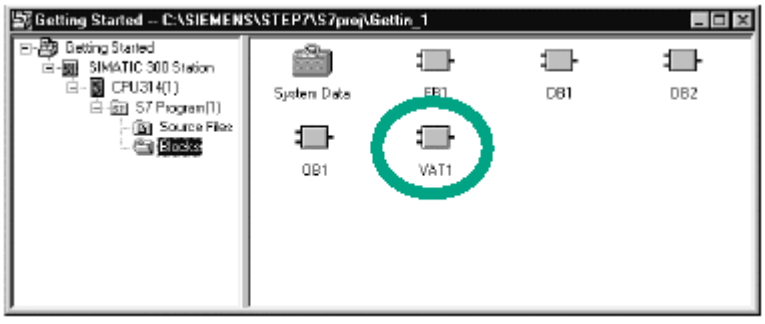


تنظیمات پیش گزیده را با تائید نمودن کادر مکالمه **" Properties "** انتخاب نمایید.



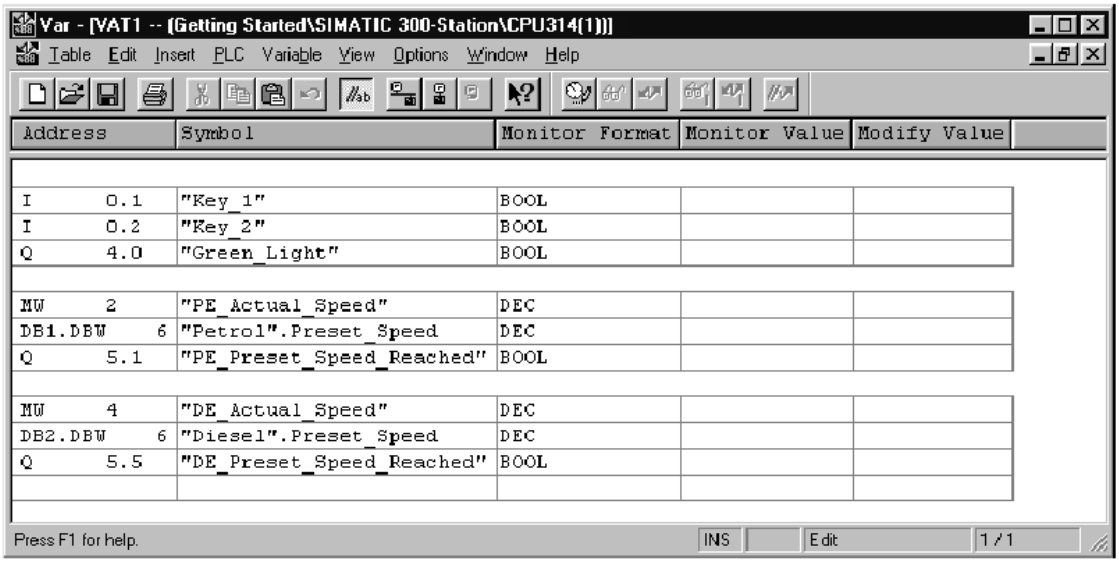
در داخل پوشه **Blocks** جدول **VAT 1** (جدول متغیرها) ایجاد می گردد.

دوبار کلیک نموده تا **VAT 1** باز شود ؛ پنجره **"Monitoring and Modifiying"** باز خواهد شد.



در ابتدا جدول متغیرها خالی میباشد. همانطوریکه در زیر نمایش داده شده است اسامی سمبلیک یا آدرس های مثال " **Getting Started**" را در داخل جدول وارد نمایید. مابقی جزئیات با زدن کلید **Enter** بخودی خود اضافه خواهند شد.

فرمت نمایش کلیه سرعت ها را به فرمت **DEC** (دسیمال) تغییر دهید. برای انجام این کار در داخل خانه مورد نظر کلیک نمایید. (مکان نما به یک پیکان در بالای ستون فرمت نمایش تبدیل خواهد شد) و فورمت **DEC** را با استفاده از کلیک راست موس انتخاب نمایید.

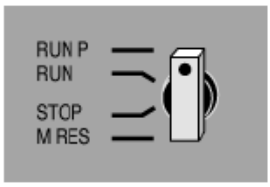


جدول متغیرهایتان را ذخیره نمایید.

Online کردن جدول متغیرها

در داخل خط ابزار پنجره "**Monitoring and Modifying Variables**" بر روی آیکون مربوطه کلیک نموده تا یک ارتباط **Online** با **CPU** پیکربندی شده برقرار شود. عبارت **ONLINE** در خط وضعیت ظاهر خواهد شد.

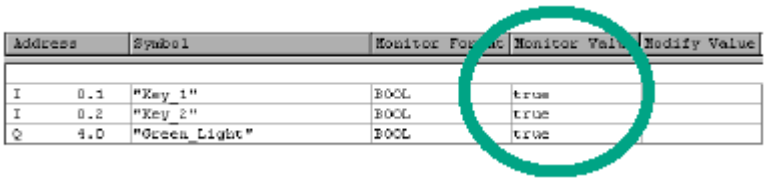
کلید تعیین وضعیت را بر روی **RUN – P** قرار دهید. (در صورتیکه هنوز این کار را نکرده باشید.)



مونیتور نمودن متغیرها

در داخـل خـط ابزار بر روی آیکون **Monitor Variables** کلیک نمایید. وضعیت کاری **CPU** در خط وضعیت بنمایش در می آید.

Key 1 و **Key 2** را فشار داده و نتیجه را در جدول متغیرها مشاهده نمایید.

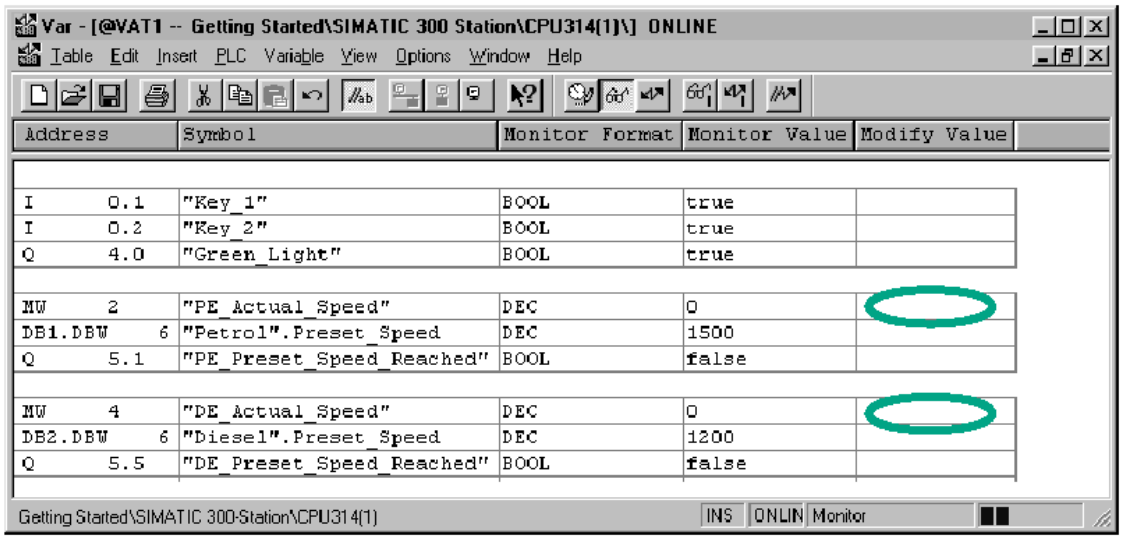


Address	Symbol	Monitor Format	Monitor Value	Modify Value
I 0.1	"Key_1"	BOOL	true	
I 0.2	"Key_2"	BOOL	true	
Q 4.0	"Green_Light"	BOOL	true	

وضعیت مونیتورینگ در جدول متغیرها از **False** به **True** تغییر خواهد کرد.

اصلاح متغیرها

در سـتون **Modify Value** مقـدار **1500** را بـرای آدرس **MW2** و **1300** را بـرای آدرس **MW4** وارد نمایید.

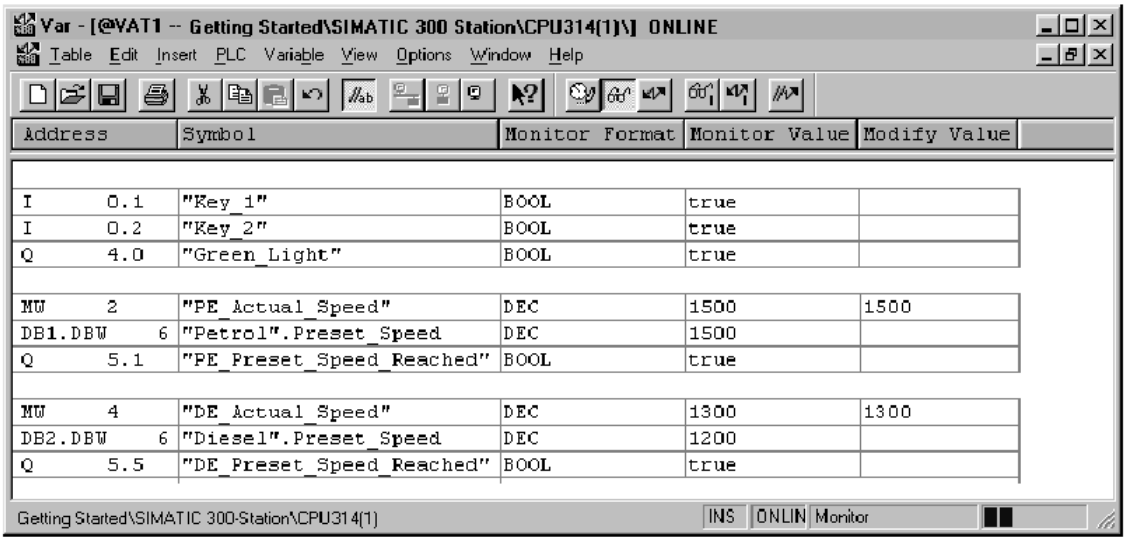


مقادیر اصلاح شده را به **CPU** تان منتقل نمایید.

عملـیات انـتقال را پیگیری نمایید ، این مقادیر در داخل **CPU** تان پردازش خواهند شد. نتیجه حاصل از مقایسه پدیدار خواهد شد.

مونیتورینگ متغیرها را متوقف نمایید. (با کلیک مجدد بر روی آیکون مربوطه در خط ابزار) و پنجره را ببندید.

کلیه پر سش ها را با **Yes** یا **OK** تایید نمایید.



معمـولاً جـداول متغـیرهای حجـیم بدلیل محدودیت در فضای صفحه نمایش ، قابلیت نمایش بطور کامل را ندارند. اگر شـما جداول متغیر بزرگی دارید ، توصیه می شود با استفاده از **STEP7** برای هر برنامه **S7** چند جدول متغیر ایجاد نمایید.

شما می توانید جداول متغیرها را دقیقاً متناسب با نیازهایتان مرتب نمایید.

شما می توانید بهمان روشی که در مورد بلوک ها انجام میدادید اسامی متفاوتی را به جداول متغیرهایتان نسبت دهید (بعـنوان مـثال نـام **OB1 – Network 1** بجـای **VAT 1**). از جـدول سمبل ها برای انتخاب اسامی جدید می توانید استفاده نمایید.

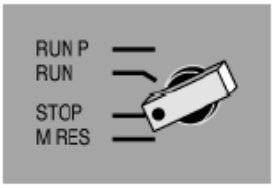
برای اطلاعات بیشتر به مباحث "Debugging" و "Testing with the variable table" واقع در Help>Contents مراجعه نمایید.

7.5 ارزیابی بافر خطا

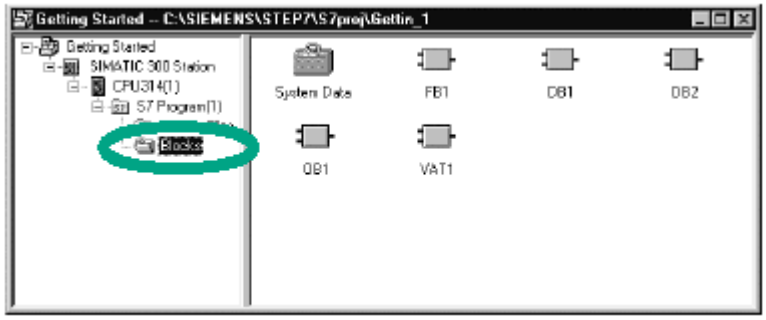
اگر در شرایطی حاد ، CPU هنگام پردازش یک برنامه S7 به حالت Stop برود و یا اینکه شما پس از Download نمودن برنامه قادر به تغییر وضعیت CPU به حالت Run نباشید، میتوانید علت وقوع خطا را از وقایع لیست شده در بافر خطا (Diagnostic Buffer) بیابید.

برای این منظور لازمست CPU در حالت Stop بوده و شما یک ارتباط Online با CPU برقرار نموده باشید.

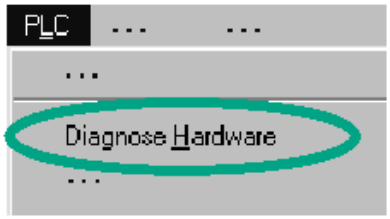
ابتدا سوئیچ تعیین وضعیت کاری CPU را بر روی STOP قرار دهید.



مجدداً نقطه شروع SIMATIC Manager و باز کردن پنجره پروژه Getting Started Offline میباشد. پوشه Block را انتخاب نمایید.

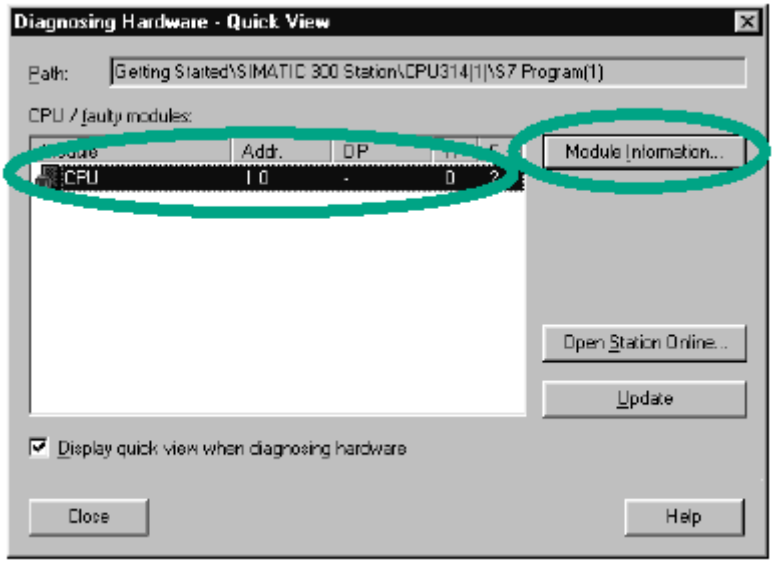


اگر در پروژه تان چندین CPU موجود است، نخست تعیین نمایید که کدام یک به حالت Stop رفته است.



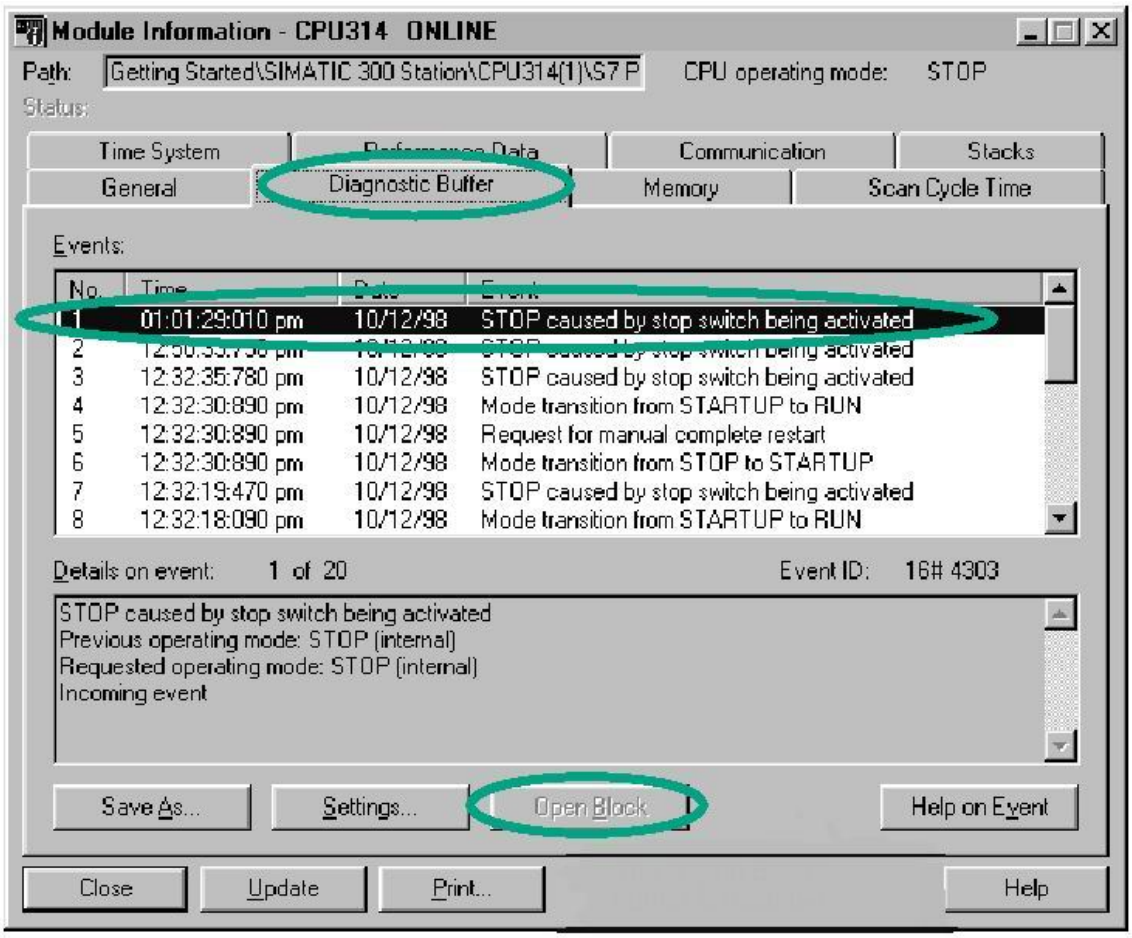
کلیه **CPU** های قابل دسترسی در کادر مکالمه **Diagnosing Hardware** لیست شده اند.

CPU ای که در حالت **Stop** قرار دارد **Highlight** (بصورت رنگی) مشخص شده است. پروژه **Getting started** تنها همان یک **CPU** ای را که دارا است نمایش میدهد. بر روی **Module Information** کلیک نموده تا بافر خطای این **CPU** را ملاحظه نمایید.



■ اگر تنها یک **CPU** در اختیار داشته باشید ، می توانید برای دسترسی اطلاعات این **CPU** مستقیماً از دستور **PLC > Module Information** استفاده نمایید.

پنجره **Module information** کلیه اطلاعات در رابطه با مشخصات و پارامترهای **CPU** تان را فراهم می نماید. حال بمنظور یافتن علت وضعیت **Stop** پوشه **Diagnostic Buffer** را انتخاب نمایید.



◙ بدلیل آنکه در بلوک پروژه **Getting Started** خطایی موجود نبود دکمه **Open Block** غیرفعال می باشد.

آخرین خطا (شماره 1) در بالای لیست قرار داشته و علت وضعیت **Stop** نمایش داده می شود. کلیه پنجره ها بجز **SIMATIC Manager** را ببندید.

◙ اگر یک خطای برنامه نویسی منجر به **STOP** رفتن **CPU** داشته باشید ، خطا را انتخاب کرده و بر روی دکمه **Open Block** کلیک نمایید.

سپس بلوک در پنجره آشنای **LAD/STL/FBD** باز شده و **Network** ای که منجر به بروز خطا شده **Highlight** (رنگی) می شود.

با مطالعه این فصل شما با موفقیت پروژه نمونه **Getting Started** را از ایجاد پروژه تا غلط گیری برنامه نهایی تکمیل نمودید. شما در فصول بعدی می توانید با کار بر روی تمرینات انتخابی ، اطلاعاتتان را بیشتر گسترش دهید.

◙ برای اطلاعات بیشتر به مبحث" **Calling the Module information**" واقع در **Help > Contents** مراجعه نمایید.

8. نوشتن یک تابع

8.1 ایجاد و باز نمودن توابع (FC)

از لحاظ سلسله مراتب برنامه توابع نیز همانند بلوک های تابعی **(FB)** ، تحت بلوک سازماندهی قرار دارند.

تابعی که بخواهد توسط **CPU** مورد پردازش قرار گیرد لازمست در بلوکی که از لحاظ سلسله مراتب در موقعیت بالاتری قرار دارد فراخوانی شود. برخلاف بلوک های تابعی (**FB**) توابع (**FC**) نیازی به وجود بلوک های اطلاعاتی (**DB**) ندارند.

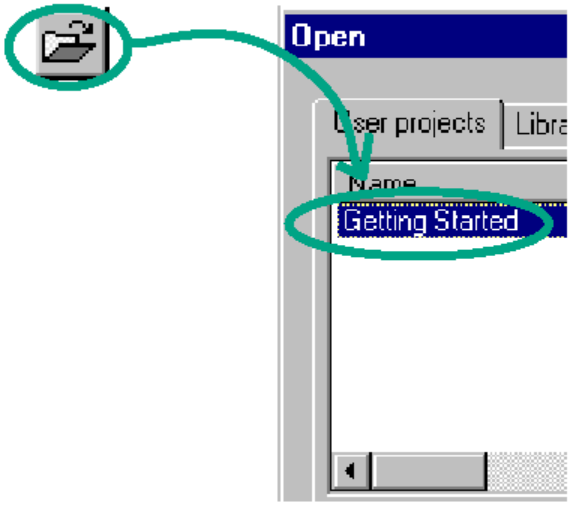
پارامترها نیز بهمراه توابع در جدول نمایش متغیرها لیست می شوند اما در مورد **Static local data** مجاز به اینکار نمی باشیم. شما می توانید بهمان صورتی که در پنجره برنامه **LAD/STL/FBD** یک بلوک تابعی ایجاد نمودید ، یک تابع را بنویسید.

برای این منظور لازمست که با برنامه نویسی به زبانهای منطق نردبانی ، **FBD** یا نمایش عبارتی (فصول 4 و 5 را ببیند) و همچنین برنامه نویسی سمبلیک (فصل 3 را ببینید) آشنا باشید.

اگر در قالب پروژه نمونه **"Getting Started"** در فصل 1 و 7 کار نموده اید آنرا باز نمایید.

در غیر اینصورت در **SIMATIC Manager** با استفاده از دستور **File>"New Project" Wizard**یک پروژه جدید ایجاد نمایید.

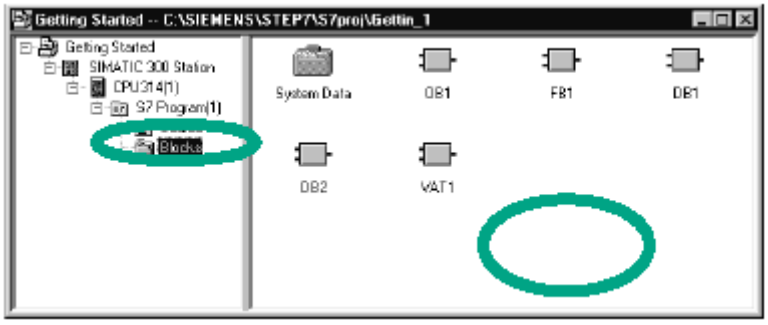
بدین منظور دستورات بخش 2.1 را تعقیب نموده و پروژه را به **"Getting Started Function"** تغییر نام دهید.



ما کار را با پروژه "**Getting Started**" ادامه می دهیم اما شما می توانید همچنان هر مرحله را توسط یک پروژه جدید به انجام برسانید.

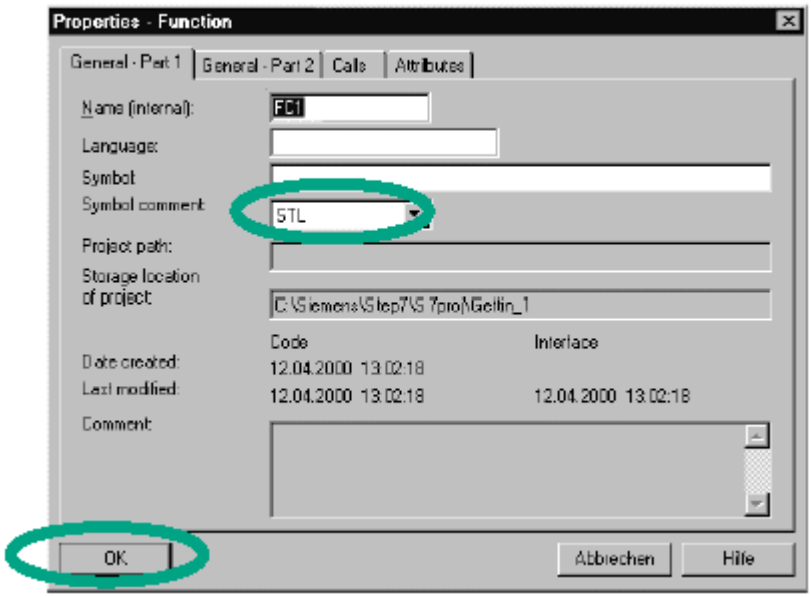
پوشه **Blocks** را یافته و آنرا باز نمایید.

بااستفاده از دکمه راست موس در نیمه راست صفحه کلیک نمایید.



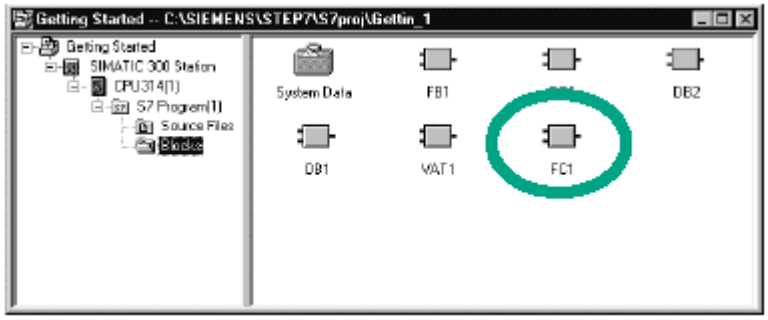
از منوی گشودنی یک **Function** (تابع – **FC**) را وارد نمایید.

در کـادر مکالمــه " **Properties–Function**" نام **FC1** را پذیرفته و زبان برنامه نویسی مورد نظر را انتخاب نمایید. با استفاده از **OK** مابقی تنظیمات پیش گزیده را تائید نمایید.



تابع **FC1** به پوشه بلوک ها افزوده می شود.

دوبار کلیک نموده تا **FC1** باز شود.



بـر خـلاف بلـوک هـای تابعی **(FB)** هیچ **Static data** ای را نمیتوان در جدول نمایش متغیرها برای یک تابع **(FC)** تعریف نمود.

Static data هایـی کـه در یک بلوک تابعی **(FB)** تعریف می شوند پس از بستن بلوک باقی می مانند.. **Static data** مثلا می تواند بیت های حافظه ای که برای مقدار محدود کننده سرعت استفاده شد باشند. (فصل 5 را ببینید)

برای نوشتن یک تابع **(FC)** می توانید اسامی سمبلیک جدول سمبل ها را مورد استفاده قرار دهید.

- بـــرای اطلاعـــات بیشـــتر بـــه مباحـــث "**Working Out the Automation**"، "**Basics of designing a program structure**" و "**Blocks in the user program**" واقـع در **Help** > **Contents** مراجعه نمایید.

8.2 نوشتن توابع (FC)

در ایـن بخش شما یک تابع تایمر **(Timer)** را در قالب یک مثال خواهید نوشت. یک تابع تایمر هنگام روشـن شـدن موتـور باعـث روشـن شدن فن مربوطه می گردد. (فصل 5 را ببینید) و سپس فن تا 4 ثانیه پس از خاموش شدن موتور کار خواهد کرد (تاخیر در قطع).

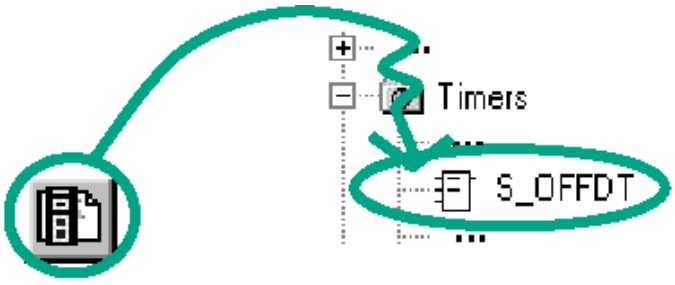
همانطوریکـه قبلاً گفته شد ، میبایست در جدول نمایش متغیرها ، پارامترهای ورودی و خروجی تابع تعییـن گردند پنجره برنامه **LAD/STL/FBD** باز میباشد. شما بهمان ترتیبی که در مورد بلوک های تابعـی **(FB)** عمـل نمودیـد بـا این جدول نمایش متغیرها کار خواهید نمود (فصل 5 را ببینید) عبارات زیر را وارد نمایید.

Address	Decl.	Name	Type	Initial Val	Comment
0.0	in	Engine_On	BOOL		Signal for switching on the engine
2.0	in	Timer_Function	TIMER		Timer function used for the switch-off delay
4.0	out	Fan_On	BOOL		Signal for switching on the fan
	in_out				
	temp				

نوشتن تابع تایمر بزبان منطق نردبانی

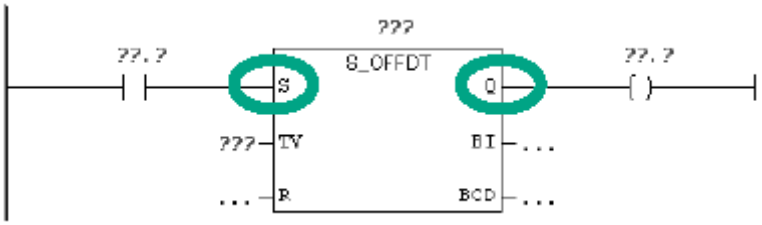
مسیر جاری را جهت وارد نمودن دستورات نردبانی انتخاب نمایید.

در داخل کاتالوگ عناصر برنامه عنصر **S_OFFDT (Start off - Delay Timer)** را وارد نموده و آنرا انتخاب نمایید.



یک کنتاکت حالت عادی باز در جلوی ورودی **S** وارد نمایید.

یک سیم پیچ **(Coil)** را بعد از خروجی **Q** وارد نمایید.

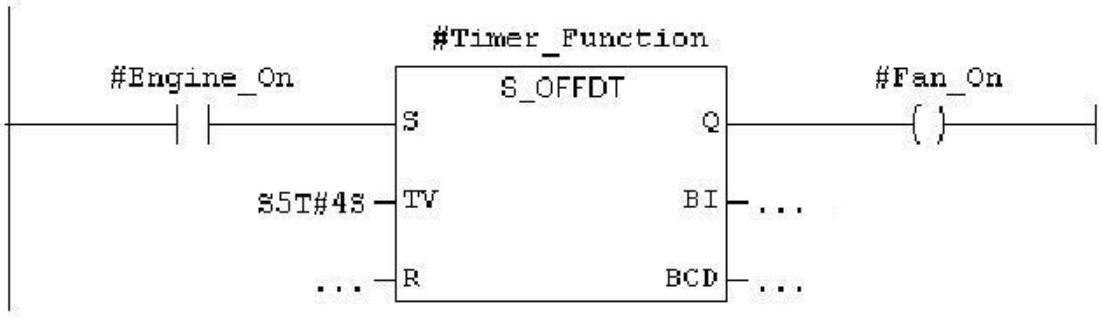


علامـت هـای سئوال را انتخاب نموده و از جدول نمایش متغیرها اسامی مربوطه تان را وارد نمایید.

(علامت # خودبخود اعمال خواهد شد)

زمـان تاخـیر را در ورودی **TV** تایمر **S _OFFDT** تنظیم نمایید. در اینجا **S5T # 4S** بمعنای آنست کـه یـک مقـدار ثابـت کـه باندازه 4 ثانیه طول می کشد (**4S**) با دیتا از نوع**s5Time#** **(S5T#)** تعریف گردیده است.

سپس تابع را ذخیره نموده و پنجره را ببندید.



◙ تابع **Timer _ Function #** با پارامتر ورودی **Engine _ On #** استارت می گردد.

بعـدا هنگامـیکه قصد فراخوانی تابع در **OB1** داشته باشیم، یکبار با پارامترهای موتور بنزینی و یکبار با پارامترهای موتور دیزلی آنرا انجام خواهیم داد (بعنوان مثال **T1** برای**"PE_Follow_On"**). شما اسامی سمبلیک این پارامترها را بعداً در جدول سمبل ها وارد خواهید نمود.

نوشتن تابع تایمر بزبان نمایش عبارتی

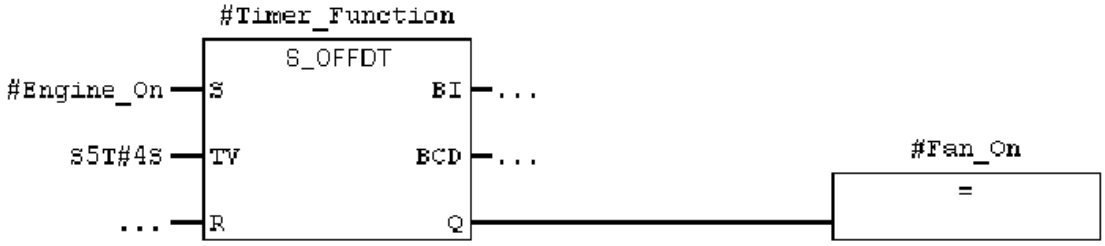
اگـر قصـد داریـد کـه بـرنامه را بـزبان نمایش عبارتی بنویسید، فضای ورودی در زیر **Network** را انتخاب نموده و عبارات را بصورتی که در مقابل نمایش داده شده است وارد نمایید.

سپس تابع را ذخیره نموده و پنجره را ببندید.

```
A     #Engine_On
L     S5T#4S
SF    #Timer_Function
A     #Timer_Function
=     #Fan_On
```

نوشتن تابع تایمر بزبان FBD

اگـر قصـد داریـد کـه بـرنامه را بـزبان **FBD** بنویسید، فضای ورودی در زیر **Network** را انتخاب نموده و برنامه **FBD** زیر را برای تابع تایمر وارد نمایید.



سپس تابع را ذخیره نموده و پنجره را ببندید.

■ تـابع تایمـر تـنها در صـورتی مورد پردازش قرار می گیرد که شما تابع را در یک بلوک که از نظر سلسه مراتب در رده بالاتری قرار داشته باشد (در مثال ما ، **OB1**) فراخوانی نمایید.

بـــــــرای اطلاعـــــــات بیشـــــــتر بـــــــه مباحـــــــث" **Helps Reference Calling** "، **"The STL,FBD,Or LAD Language Description"** و " **Timer Instruction** "واقـع در **Help** > **Contents** مراجعه نمایید.

8.3 فراخوانی تابع در OB1

فراخوانی تابع **FC1** همانند فراخوانی بلوک تابعی **(FB)**، در **OB1** صورت می پذیرد.

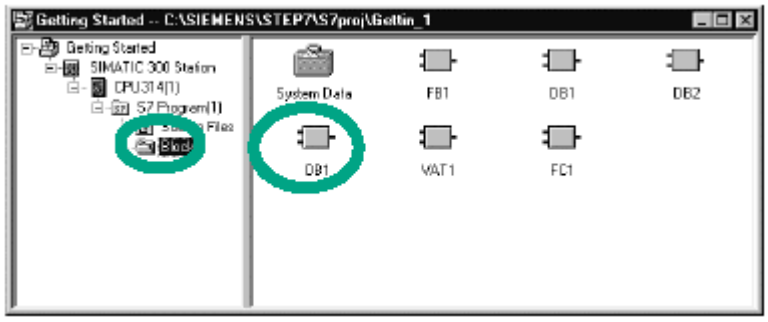
در داخل **OB1** با استفاده از آدرس های مربوطه به موتور بنزین یادیز لی میتوان به کلیه پارامترهای تـابع دسترسـی داشـت. بدلـیل آنکـه این آدرس ها هنوز در جدول سمبل ها تعریف نگردیده اند ، در اینجا اسامی سمبلیک آدرس ها را اضافه خواهیم نمود.

■ یـک آدرس قسـمتی از یک عبارت **STEP7** بشمار می آید و تعیین می کند که پردازشگر اجرای دستور را بر روی چه چیزی انجام دهد. آدرس ها می توانند مطلق یا سمبلیک باشند.

حـال **SIMATIC Manager** بـازبوده و پـروژه **"Getting Started"** یـا پـروژه جدیـدی که خودتان تعریف نموده اید را شامل می شود.

پوشه **Blocks** را یافته و **OB1** را باز نمایید.

پنجره برنامه **LAD/STL/FBD** گشوده می گردد.



■ اگر شما در فصل 4 جدول سمبل ها را از یک پروژه نمونه (**GS_STL_ExampleGas_LAD_Example,** یا **GS_FBD_Example**)بـه پروژه **Getting started** تان کپی نموده باشید دیگر نیازی به اضافه نمودن سمبل ها در اینجا نخواهید داشت.

افزودن اسامی سمبلیک در مرحله ثانویه

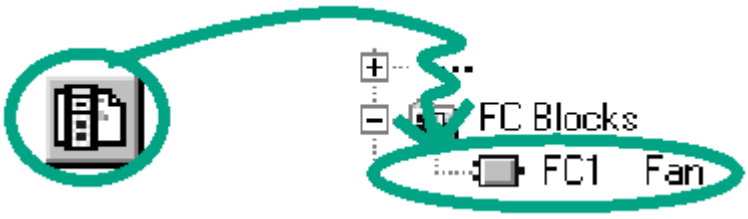
از درون پـنجره بـرنامه **LAD/STL/FBD** و توسط دستور **Options > Symbol table** جدول سمبل هـا را بـاز نمایـید و از نـوار پیمایش گوشه نیمه راست صفحه پنجره برای دیدن کل جدول سمبل ها استفاده نمایید.

حال سمبل های زیر را به جدول سمبل ها بیفزایید.

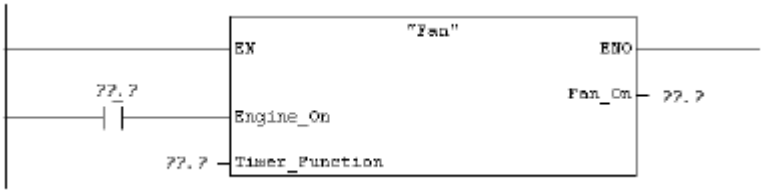
Symbol	Address	Data Type	Comment
DE_Follow_On	T 2	TIMER	Follow-on time for diesel engine fan
PE_Follow_On	T 1	TIMER	Follow-on time for petrol engine fan
Fan	FC 1	FC 1	Fan control
PE_Fan_On	Q 5.2	BOOL	Command for switching on petrol engine fan
DE_Fan_On	Q 5.6	BOOL	Command for switching on diesel engine fan

نحوه انجام فراخوانی در زبان منطق نردبانی:

در حال حاضر در محیط نمایش **LAD** قراردارید. **network** جدیدی ایجاد نمایید(شماره6). سپس در کاتالوگ عناصر برنامه **FC1** را یافته و آنرا وارد نمایید.

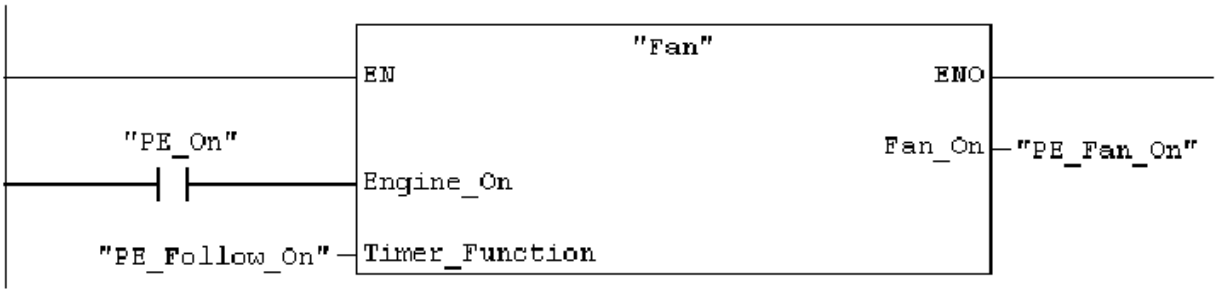


در جلوی "**Engine_On**" یک کنتاکت حالت عادی باز وارد نمایید.

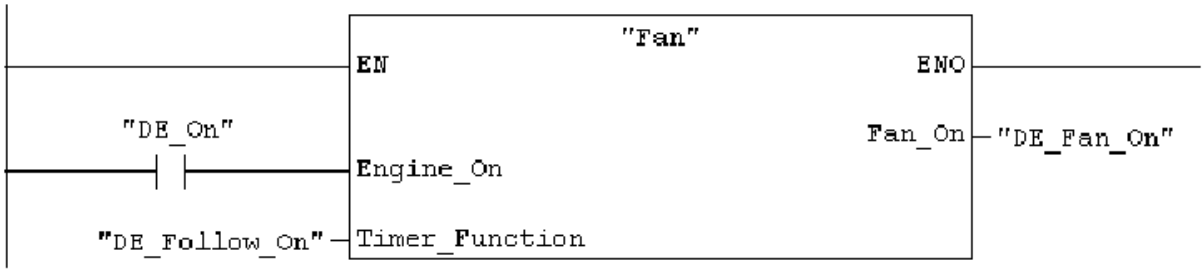


◙ با استفاده از دستور **View>Display>Symbolic Representation** میتوانید نحوه نمایش آدرسها را از سمبولیک به مطلق و بالعکس تغییر دهید.

بر روی علامت سئوالهای فراخوانی **FC1** کلیک نموده و اسامی سمبولیک را وارد نمایید.



توسط آدرسهای موتور دیزلی برنامه فراخوانی تابع **FC1** را در **network** 7 بنویسید. اینکاررا به همان روشی که برای **network** قبلی بکار بردید انجام دهید (آدرسهای موتور دیزلی را قبلا به جدول سمبلها افزوده اید).



بلوک را ذخیره نموده و پنجره را ببندید.

◙ جهت مشاهده اطلاعات مربوط به آدرسهای بکار رفته در هر **network** دستور

View>Display>Symbol Information را فعال نمایید.

جهت بنمایش درآمدن چند **network** در یک صفحه دستور**View>Display>Comment** و در صورت نیاز

View>Display>Symbol Information را غیر فعال نمایید.

با استفاده از دستور **View>Zoom Factor** میتوانید سایز صفحه نمایش**network** ها را تغییر دهید.

نحوه انجام فراخوانی در زبان نمایش عبارتی:

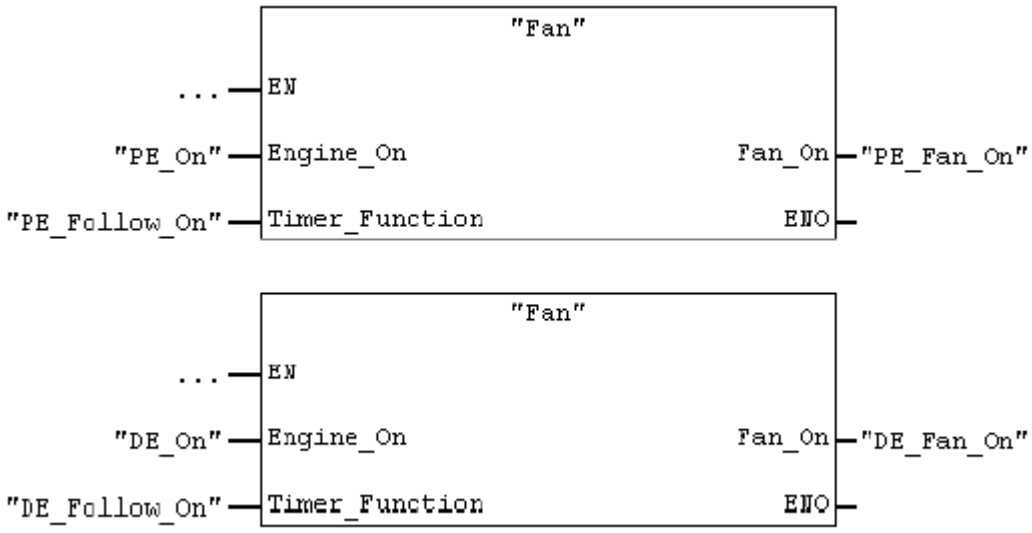
درحالیکه در حال برنامه نویسی بزبان نمایش عبارتی هستید فضای ورودی ها واقع در زیر **network** جدید ایجاد شده را انتخاب نمایید و عناصر **STL** ای را که در زیر نمایش داده شده است را وارد نمایید.

```
Network 6 : Controlling the Fan for the Petrol Engine
    CALL  "Fan"
     Engine_On      :="PE_On"
     Timer_Function:="PE_Follow_On"
     Fan_On         :="PE_Fan_On"

Network 7 : Controlling the Fan for the Diesel Engine
     CALL  "Fan"
      Engine_On      :="DE_On"
      Timer_Function:="DE_Follow_On"
      Fan_On         :="DE_Fan_On"
```

پس از ذخیره نمودن؛ پنجره را ببندید.

نحوه انجام فراخوانی در زبان دیاگرام بلوکی:

درحالیکه در حال برنامه نویسی بزبان دیاگرام بلوکی هستید فضای ورودی ها واقع در زیر **network** جدید ایجاد شده را انتخاب نمایید و عناصر **FBD** ای را که در زیر نمایش داده شده است را وارد نمایید.



پس از ذخیره نمودن؛ پنجره را ببندید.

فراخوانی هایی که برای توابعمان مورد مثالمان نوشتیم از نوع فراخوانی غیر شرطی بوده که همواره در حال پردازش میباشند.

در صورت نیاز میتوان فراخوانی یک تابع یا بلوک تابعی را منوط به براورده بودن شرط یا شرایط خاص مثل یک ورودی و یا یک پیش شرط منطقی نمود. ورودی **EN** و خروجی **ENO** بمنظور برنامه نویسی شرطی تعبیه شده اند.

◙ برای اطلاعات بیشتر به مباحث "**Helps Refrence Calling**" , "**The LAD, FBD or STL Language Description**" یا "**Program Control Instructions**" واقع در **Help > Contents** مراجعه نمایید.

9. برنامه نویسی یک بلوک اطلاعاتی اشتراکی

9.1 ایجاد و باز نمودن بلوک های اطلاعاتی اشتراکی

در صورتیکه میزان حافظه داخلی موجود در **CPU** برای ذخیره نمودن کلیه اطلاعات کافی نباشد میتوانید اطلاعات مشخصی را در یک بلوک اطلاعاتی اشتراکی ذخیره نمایید.

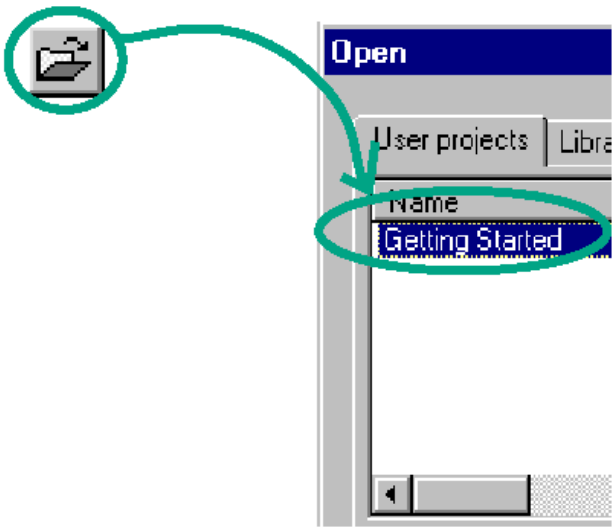
اطلاعات موجود در یک بلوک اطلاعاتی اشتراکی در دسترس تمام بلوک های دیگر می باشد. از طرف دیگر یک بلوک اطلاعاتی **Instance** به یک بلوک تابع ساز خاص منتسب می گردد و اطلاعات آن تنها قابل دسترسی برای این بلوک تابع ساز میباشد (بخش 5.5 را ملاحظه کنید).

حال شما میبایست با برنامه نویسی بزبانهای منطق نردبانی، نمایش عبارتی یا **STL** (فصول 4 و 5 را ملاحظه کنید) و همچنین برنامه نویسی سمبلیک (فصل 3 را ببینید) آشنا باشید.

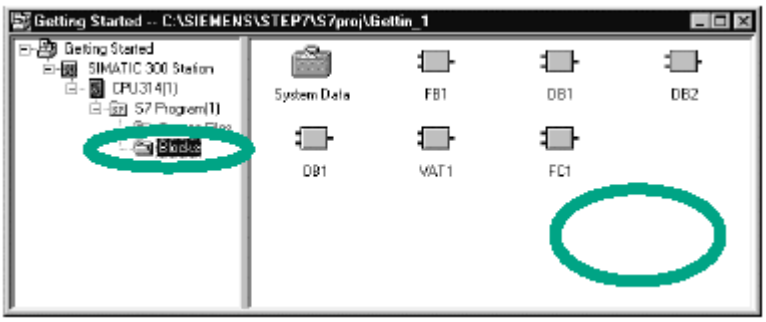
اگر طی فصول 1 تا 7 در قالب پروژه نمونه **"Getting Started"** کار کرده اید هم اکنون آنرا باز نمایید. در غیر اینصورت با استفاده از ویزارد **File>"New project"** یک پروژه جدید را در **SIMATIC Manager** ایجاد نمایید.

برای انجام این کار ، دستورات بخش 2.1 را تعقیب نموده و پروژه را به **"Getting Started Function"** تغییر نام دهید.

ما کارمان را با پروژه **Getting Started** ادامه میدهیم ولی شما می توانید همچنان هر مرحله را با استفاده از یک پروژه جدید طی نمایید.



بسراغ پوشه **Blocks** رفته و آنرا باز نمایید.



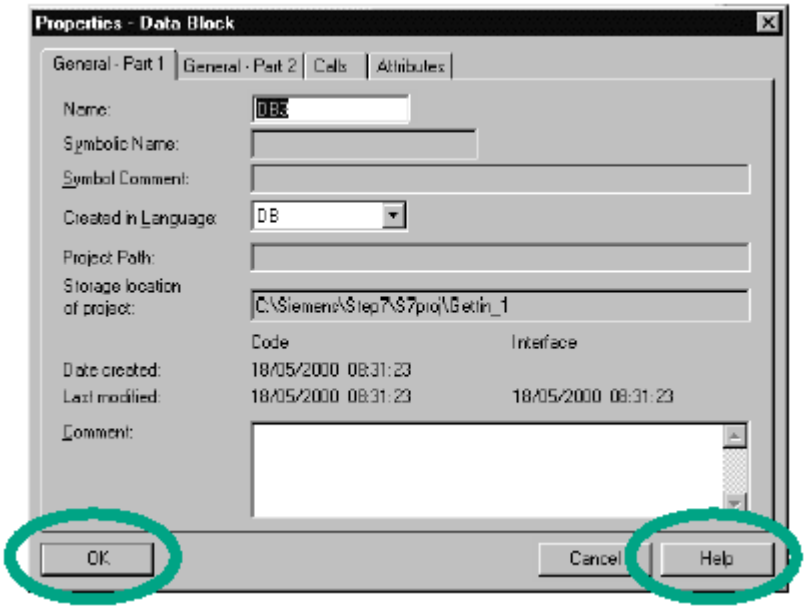
در نیمه راست پنجره کلیک راست نمایید.

یک بلوک اطلاعاتی **(DB)** را از طریق منوی گشودنی ایجاد نمایید.

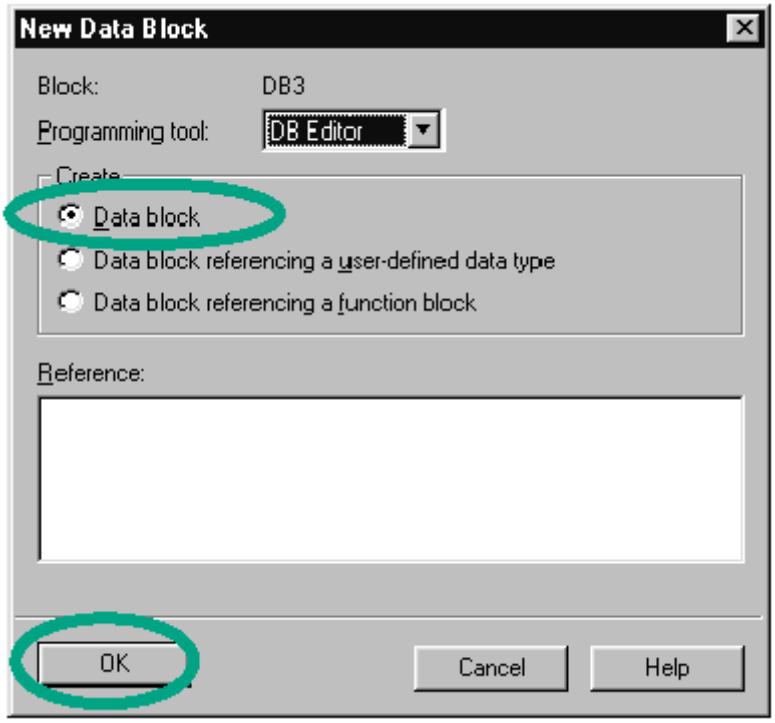
در کادر مکالمه **"Properties–Data Block"** کلیه تنظیمات پیش گزیده را توسط **OK** بپذیرید. در صورت نیاز به اطلاعات بیشتر از **Button , "Help"** استفاده نمایید.

بلوک اطلاعاتی **DB3** به پوشه **Blocks** افزوده شده است.

دوبار کلیک نموده تا **DB3** باز شود.



در کـادر مکالمـه "**New Data Block**" کـه پدیـدار مـی گـردد انـتخاب **Data Block** را فعال نمایید. با استفاده از **OK** کادر مکالمه را ببندید.



Instance Data Block

یـــــادآوری : شـــــما در بخـــــش 5.5 بـــــا فعـــــال نمـــــودن انـــــتخاب "**Data Block referncing a function block**" یـک بلـوک اطلاعاتـی **Instance** ایجـاد نمودیـد. در مقـابل ، بـا استفاده از **Data Block** شما یک بلوک اطلاعاتی اشتراکی ایجاد می نمایید.

وارد نمودن متغیرها درون بلوک اطلاعاتی

در ستون نام (Name) "PE_Actual_Speed" را وارد نمایید.

با دکمه راست موس کلیک نموده تا با استفاده از منو ، فرمان **Elementary Types >INT** را از داخل برای انتخاب نوع ، دکمه راست موس را کلیک نموده و از درون منوی گشودنی پدیدار شده فرمان **Elementary Types >INT** را انتخاب نمایید.

Address	Name	Type	Initial Value	Comment
0.0		STRUCT		
=0.0		END_STRUCT		

در مثال زیر ، سه دیتای اشتراکی در **DB3** تعریف شده است.

این اطلاعات را بهمین صورت در جدول نمایش متغیرها واردنمایید.

Address	Name	Type	Initial Value	Comment
0.0		STRUCT		
+0.0	PE_Actual_Speed	INT	0	Actual speed for petrol engine
+2.0	DE_Actual_Speed	INT	0	Actual speed for diesel engine
+4.0	Preset_Speed_Reached	BOOL	FALSE	Both engines have reached the preset speed
=6.0		END_STRUCT		

◙ در بلوک اطلاعاتی در مورد متغیرهای سرعت واقعی "PE_Actual_Speed" و"DE_Actual_Speed" همانند کلمه های حافظه **(PE-Actual – Speed) MW2** و **(DE – Actual – Speed) MW4** عمل می نماییم. این را میتوانید در فصل بعد ملاحظه نمایید.

بلوک اطلاعاتی را ذخیره نمایید.

نسبت دادن سمبل

این امکان برای شما وجود دارد که به بلوک های اطلاعاتی اسامی سمبلیک نسبت دهید جدول سمبل ها **(Symbol Table)** را باز نموده و نام سمبلیک "S_Data" را برای بلوک اطلاعاتی **DB3** وارد نمایید.

◙ اگـر شـما جـدول سـمبل هـا را از یـک پـروژه نمونـه (zEn01_02_STEP7_STL_1–10 ، zEn01_04_STEP7_FBD_1–10 یـا zEn01_06_STEP7_LAD_1–10) بـه پـروژه "Getting Started" تان در فصل 4 کپی نموده اید دیگر نیازی به افزودن سمبل دیگری ندارید.

Symbol	Address	Data Type	Comment
...	...	...	...
S_Data	DB 3	DB 3	Shared data block

جدول سمبل ها را ذخیره نموده و پنجره **Symbol Editor** را ببندید.

جدول نمایش متغیرهای بلوک اطلاعاتی اشتراکی را نیز ببندید.

بلوک های اطلاعاتی اشتراکی در جدول نمایش متغیرها:

با استفاده از فرمان **View > Date View** شما می توانید برای بلوک اطلاعاتی اشتراکی مقادیر واقعی دیتاهای از نوع **INT** را در جدول تغییر دهید. (بخش 5.5 را مشاهده کنید.)

بلوک های اطلاعاتی اشتراکی در جدول سمبل ها :

بـر خـلاف بلـوک اطلاعاتـی **Instance** در بلـوک اطلاعاتی مشترک نوع دیتا در داخل جدول سمبل ها همواره آدرس مطلـق میباشـد. در مـثال مـا نـوع دیتا "**DB3**" میباشد. برای بلوک اطلاعاتی **Instance** همواره بلوک تابعی مربوطه بعنوان نوع دیتا مشخص می گردد.

◙ بـرای اطلاعـات بیشتر به مباحث "**Programming Blocks** " و "**Creating Data Blocks**" واقع در **Help > Contents** مراجعه نمایید.

10 نوشتن یک Multiple Instance

10 . 1 ایجاد و باز نمودن یک بلوک تابعی بالا دستی (Higher – Level)

در فصل 5 شما با استفاده از بلوک تابعی "Engine" (FB1) برنامه کنترل یک موتور را ایجاد نمودید بلوک تابعی FB1 هنگام فراخوانی در OB1 بلوک های اطلاعاتی "Petrol" (DB1) و "Diesel" (DB2) را مورد استفاده قرار میداد.

هر بلوک اطلاعاتی ، اطلاعات متفاوتی از موتورها را در برمی گرفت (مثلاً #Setpoint_Speed)

حال فرض کنید برای کار اتوماسیون تان نیاز به برنامه های دیگری جهت کنترل موتور داشته باشید؛ مثلا یک برنامه کنترل موتور روغنی یا یک موتور هیدروژنی و غیره.

اگر بخواهید با روندی که تا بحال آموخته اید اینکار را انجام دهید هر بار برای هر موتور اضافی یک بلوک اطلاعاتی (DB) جدید که حاوی اطلاعات موتور میباشد را به FB1 نسبت میدهید؛ بعنوان مثال FB1 بهمراه DB3 برای کنترل موتور روغنی، FB1 بهمراه DB4 برای موتورهیدروژنی و غیره.

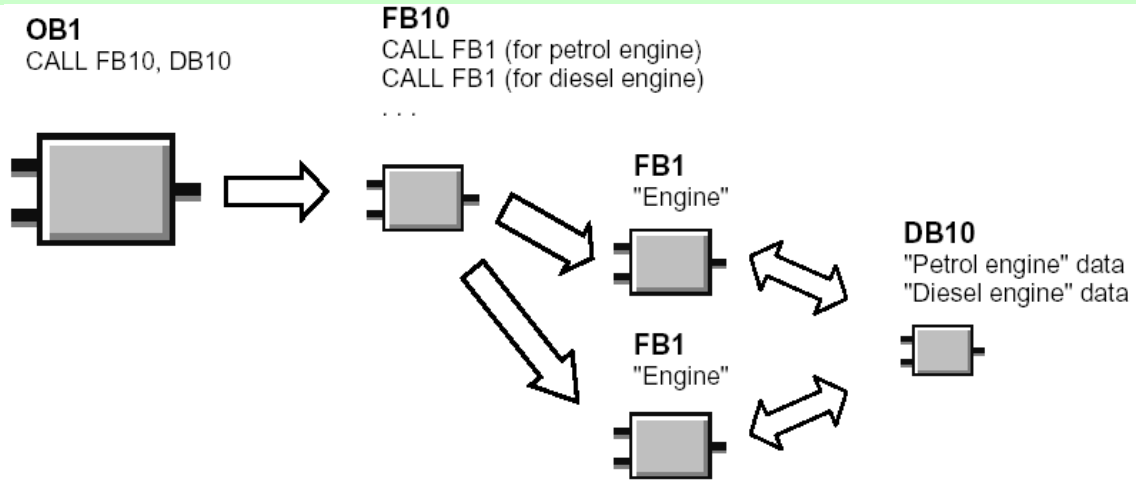
بدین ترتیب با افزایش برنامه های کنترل موتور جدید تر تعداد بلوک ها بطور قابل ملاحظه ای افزایش خواهد یافت. از طرف دیگر با استفاده از Multiple Instance ها می توانید تعداد بلوک ها را کاهش دهید.

برای این منظور شما میبایست یک بلوک تابعی (FB) بالا دستی جدید ایجاد نمایید. (در مثال ما FB 10) و FB1 را بدون هیچگونه تغییری در داخل آن و بعنوان یک Local Instance فراخوانی نمایید.

در هر فراخوانی Subordinate FB1 اطلاعاتش رادر داخل بلوک اطلاعاتی DB10 مربوط به FB10 بالادستی ذخیره می نماید. این بدان معناست که شما مجبور به نسبت دادن کلیه بلوک های اطلاعاتی به FB1 نمی باشید. کلیه بلوک های تابعی (FB) به یک بلوک اطلاعاتی واحد باز می گردند. (در اینجا DB10).

- بلوک های اطلاعاتی DB1 ، DB2 در داخل DB10 ، بکار گرفته شده اند.

برای انجام این کار شما می بایست **FB1** را در دیتاهای محلی **Static** مربوط به **FB10** درج نمایید.



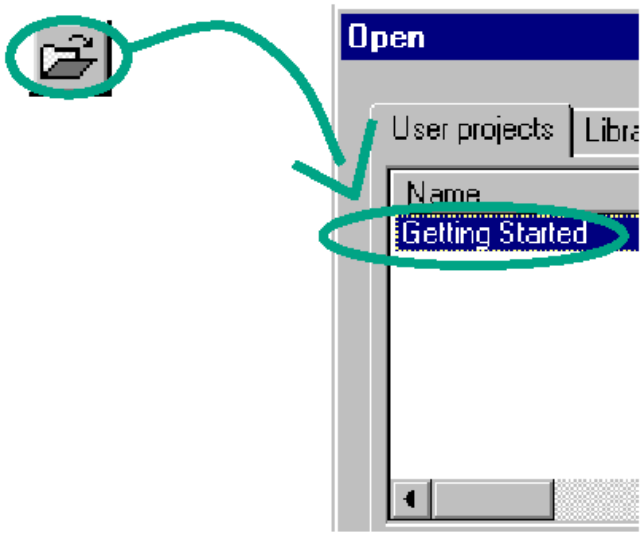
در حال حاضر شما با برنامه نویسی بزبان منطق نردبانی ، **FBD** یا نمایش عبارتی (فصول 4 و 5 را ببینید) و همچنین برنامه نویسی سمبلیک (فصل 3 را ببینید) آشنا شده اید.

در صورتیکه در فصول 1 تا 7 با مثال **"Getting Started"** کارکرده اید پروژه **Getting Started** را باز نمایید.

در غیر اینصورت یکی از پروژه های زیر را در داخل **SIMATIC Manager** باز نمایید :

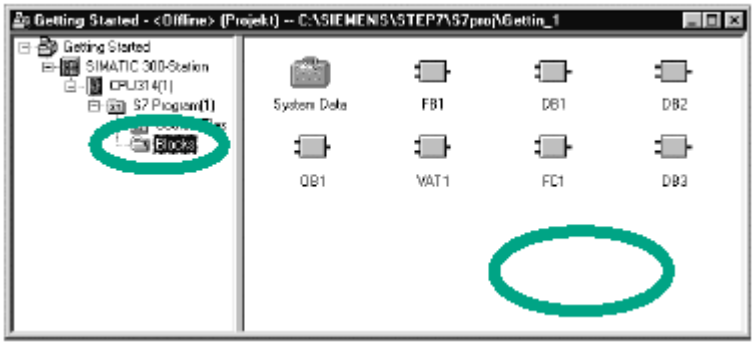
zEn01_06_STEP7_LAD_1–9 برای منطق نردبانی ، **zEn01_02_STEP7_STL_1–9**

برای نمایش عبارتی و **zEn01_04_STEP7_FBD_1–9** برای **FBD**.



پوشه **Blocks** را یافته و آنرا باز نمایید.

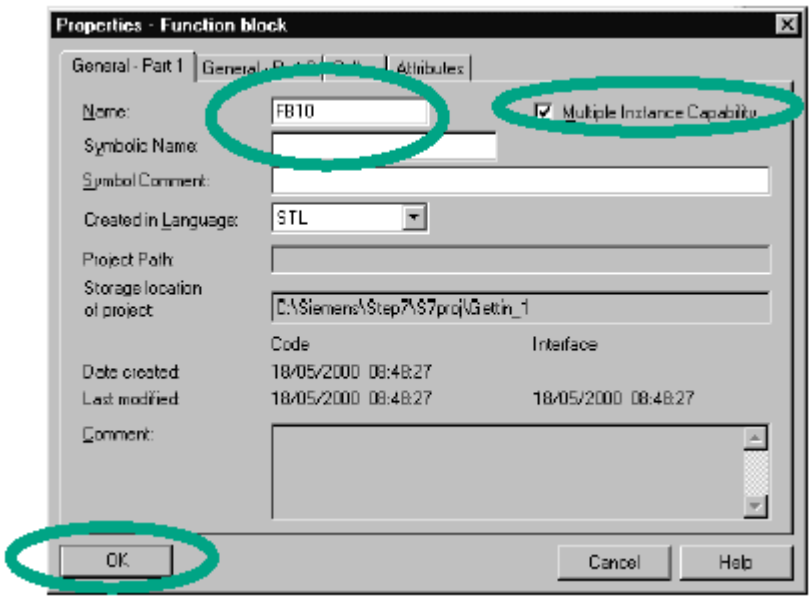
در نیمه راست پنجره کلیک راست نموده و با استفاده از منوی گشودنی، یک بلوک تابعی ایجاد نمایید.



نام بلوک را به **FB10** تغییر داده و زبان برنامه نویسی مورد نظر را انتخاب نمایید.

FB Multiple Instance را (در صورت لزوم) فعال نموده و ما بقی تنظیمات پیش گزیده را با **OK** تایید نماید.

FB10 به پوشه **blocks** افزوده می شود. دوبار کلیک نمایید تا **FB10** باز شود.



شما می توانید **Multiple Instance** ها را برای هر بلوک تابعی ای حتی بعنوان مثال برای برنامه های کنترل شیر ایجاد نمایید. در صورتیکه قصد استفاده از **Multiple Instance** ها را داشته باشید دقت کنید که هر دوی بلوک های فراخوان و فراخوانده شده میبایست قابلیت **Multiple Instance** را دارا باشند.

◙ برای اطلاعات بیشتر می توانید به مباحث " **Blocks Programming**" و "**Creating Blocks and Libraries**" واقع در **Help > Contents** مراجعه نمایید.

10.2 نوشتن FB10

بمنظور فراخوانی **FB1** تحت عنوان یک **Local Instance** مربوط به **FB10** میبایست یک **Static Variable** با یک نام متفاوت برای هر بار فراخوانی **FB1** ، **declare** گردد.

در اینجا نوع دیتا **FB1** "**Enginre**" میباشد.

پرکردن جدول نمایش متغیرها

با باز نمودن پنجره برنامه **LAD/STL/FBD** متغیرهای زیر را بمنظور فراخوانی **FB1** وارد نمایید :

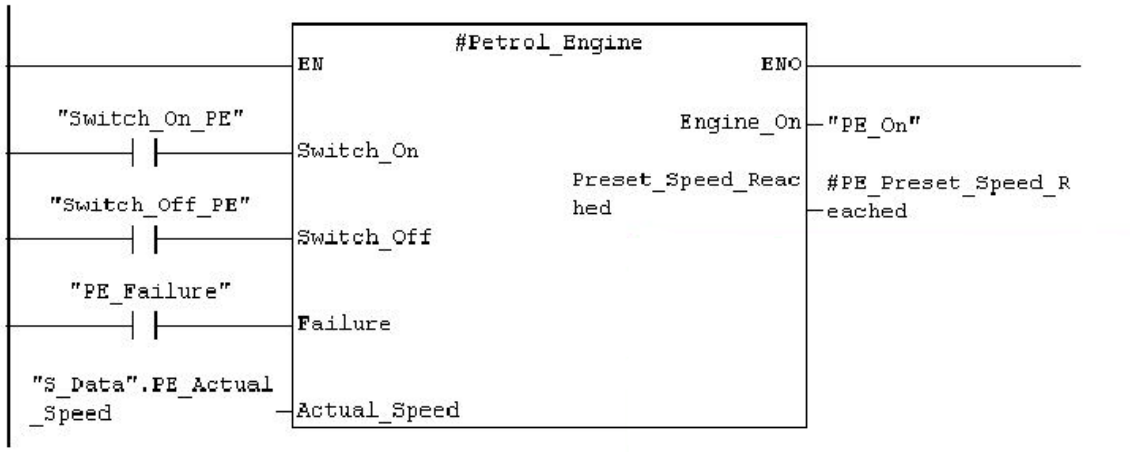
Address	Decl.	Name	Type	Initial Valu	Comment
	in				
0.0	out	Preset_Speed_Reached	BOOL	FALSE	Both engines have reached the preset speed
	in_out				
2.0	stat	Petrol_Engine	"Engine"		First local instance of FB1 "Engine"
10.0	stat	Diesel_Engine	"Engine"		Second local instance of FB1 "Engine"
0.0	temp	PE_Preset_Speed_Reached	BOOL		Preset speed reached (petrol engine)
0.1	temp	DE_Preset_Speed_Reached	BOOL		Preset speed reached (diesel engine)

- **Local Instance** های وارد شده در کاتالوگ عناصر برنامه ، تحت "**Multiple Instances**" پدیدار خواهند شد.

نوشتن **FB10** بزبان منطق نردبانی

فراخوانی "**Petrol – Engine**" را بعنوان بلوک **Multiple – Instance** ، "**Petrol – Engine**" در **Network 1** وارد نمایید.

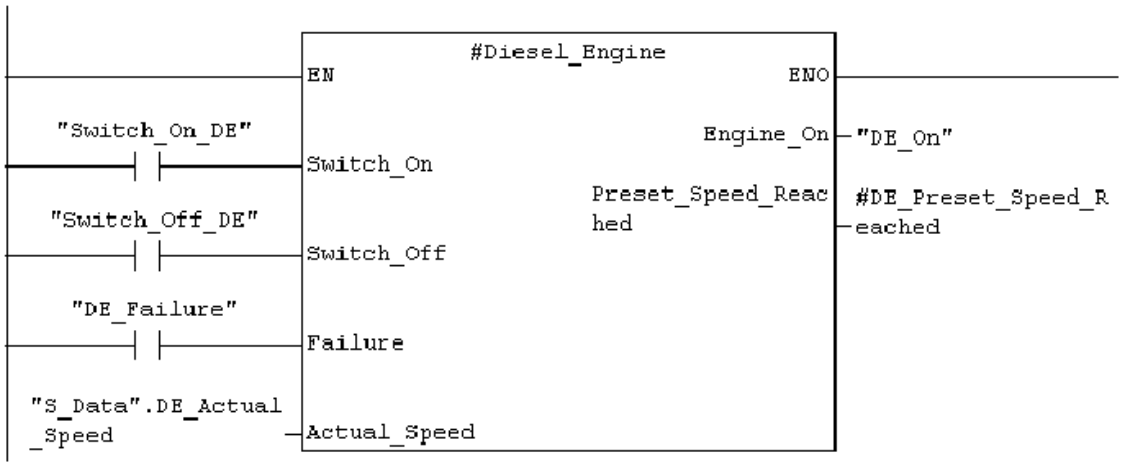
سپس کنتاکت های حالت عادی باز مورد نیاز را وارد نموده و توسط نامهای سمبلیک ، فراخوانی را کامل نمایید.



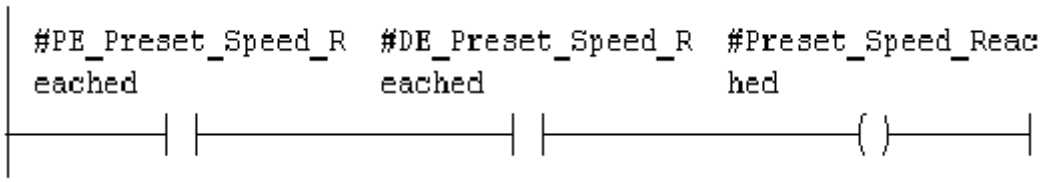
◙ **"Actual – Speed"** برای موتورها از یک خانه حافظه مورد استفاده قرار نمی گیرد (بخش 5.6 را ببینید **Onwards**) بلکه ازیک بلوک اطلاعاتی اشتراکی استخراج می گردد.(بخش 9.1 را ببینید) بطور عمومی آدرس دهی بصورت زیر صورت می گیرد :

Data-block".address" بعنوان مثال: **."S-Data".PE-Actual-Speed**

یک **Network** جدید ایجاد نموده و فراخوانی مربوط به موتور دیزلی را انجام دهید. بهمان ترتیبی که در **Network 1** عمل شد پیش بروید.



یک **Network** جدید بازکرده و یک مدار سری با آدرس های مشخص شده ایجاد نمایید. سپس برنامه تان را ذخیره نموده و بلوک را ببندید.



◙ متغیرهای موقتی "**PE-Setpoint-Reached**" و "**DE-Setpoint-Reached**" به پارامتر خروجی " – **Setpoint Reached**" که بعداً در داخل **OB1** مورد پردازش قرارمی گیرد اعمال می شوند.

نوشتن FB10 بزبان نمایش عبارتی

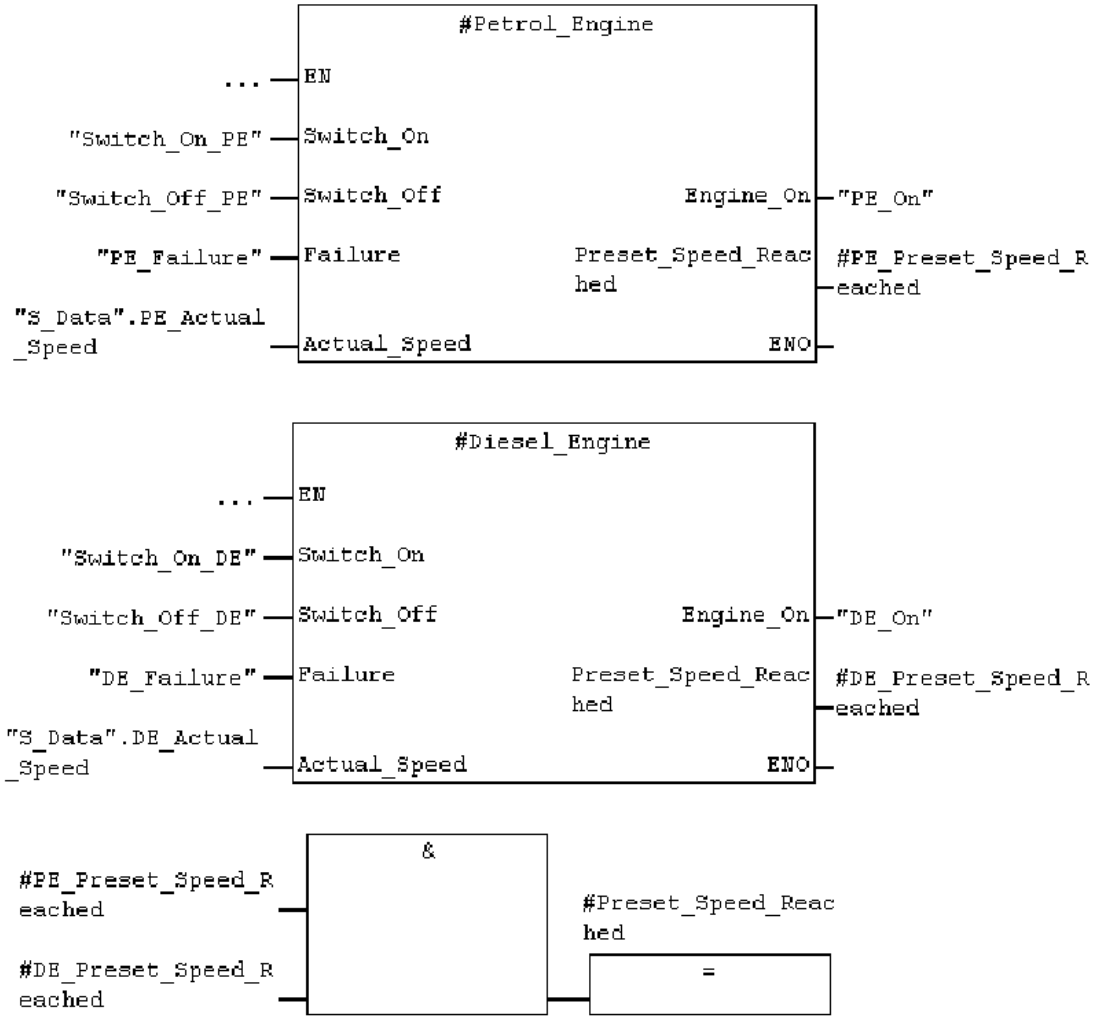
اگـر در حـال نوشـتن بـرنامه بـزبان نمـایش عبارتـی هستید فضای وارد نمودن ورودی در زیر یک **Network** جدیـد را انـتخاب نمـوده و دسـتورات **STL** ای را که در اینجا نشان داده شده است وارد نمایید. سپس برنامه تان را ذخیره نموده و بلوک را ببندید.

```
CALL  #Petrol_Engine
 Switch_On           :="Switch_On_PE"
 Switch_Off          :="Switch_Off_PE"
 Failure             :="PE_Failure"
 Actual_Speed        :="S_Data".PE_Actual_Speed
 Engine_On           :="PE_On"
 Preset_Speed_Reached:=#PE_Preset_Speed_Reached

CALL  #Diesel_Engine
 Switch_On           :="Switch_On_DE"
 Switch_Off          :="Switch_Off_DE"
 Failure             :="DE_Failure"
 Actual_Speed        :="S_Data".DE_Actual_Speed
 Engine_On           :="DE_On"
 Preset_Speed_Reached:=#DE_Preset_Speed_Reached

A     #PE_Preset_Speed_Reached
A     #DE_Preset_Speed_Reached
=     #Preset_Speed_Reached
```

نوشتن FB10 بزبان FBD

اگـر در حـال نوشـتن بـرنامه بـزبان **FBD** هستید فضای وارد نمودن در زیر یک **Network** جدید را انتخاب نموده و دستورات **FBD** زیر را وارد نمایید.

سپس برنامه را ذخیره نموده و بلوک را ببندید.



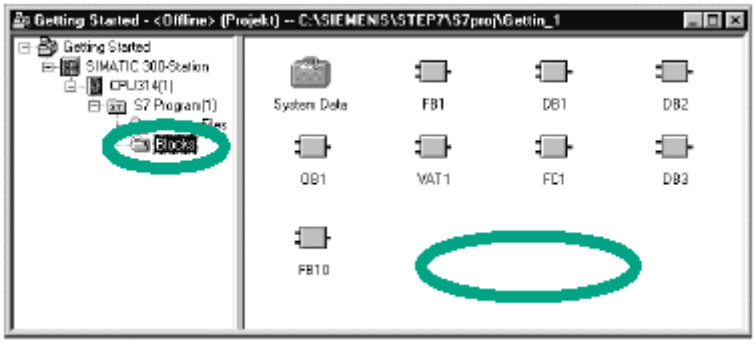
بـرای ویـرایش هر دو فراخوانی **FB1** در داخل **FB10** لازمست **FB10** فراخوانی شده باشد. **Multiple Instance** هـا تنها می توانند برای بلوک های تابعی نوشته شوند. ایجاد **Multiple Instance** برای توابع **(FC** ها) امکان پذیر نمی باشد.

بـرای اطلاعـات بیشـتر می توانید به مباحث **Programming Blocks**، **Creating Logic Blocks** و **Multiple Instances in the variable Declaration Table** واقع در **Help > Contents** مراجعه نمایید.

10.3 ایجاد DB10 و تغییر مقادیر واقعی

بلـوک اطلاعاتی جدید **DB10** جایگزین بلوک های اطلاعاتی **DB1** و **DB2** خواهد شد. اطلاعات مربوط بـه موتـور بنزینـی و موتـور دیزلی در **DB10** ذخیره شده و بعداً نیاز است که **FB10** در داخل **OB1** فراخوانی گردد (مبحث "فراخوانی **FB1** در داخل **OB1**" را در بخش 5.6 به بعد ملاحظه نمایید)

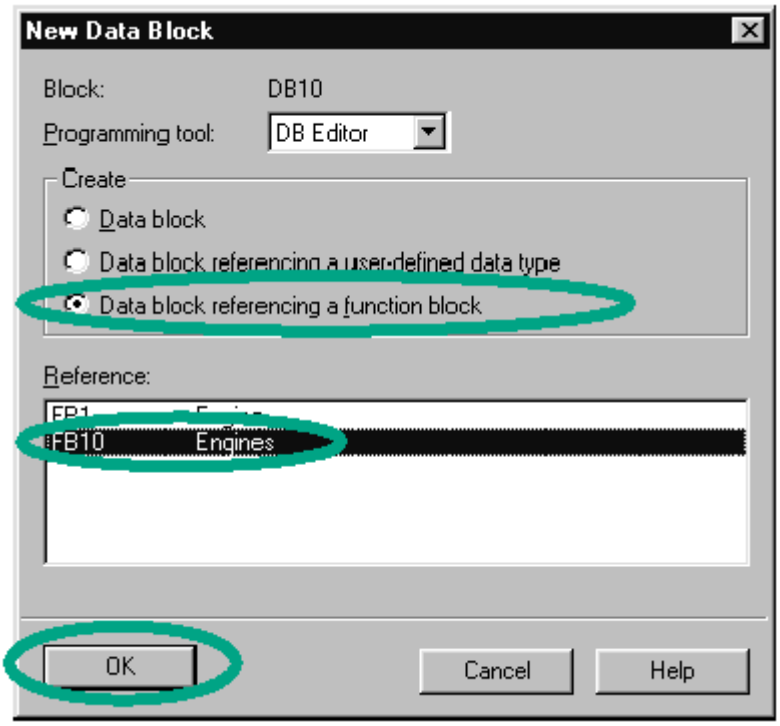
با استفاده منوی گشودنی ، بلوک اطلاعاتی **DB10** را در داخل پوشه **Blocks** مربوط به پروژه " **Getting Started** " در محیط **SIMATIC Manager** ایجاد نمایید.



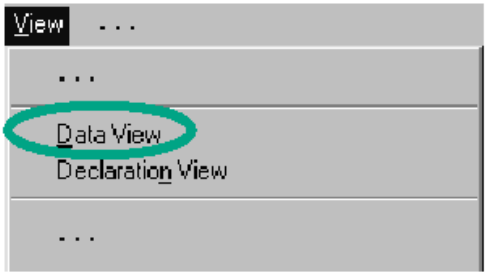
برای انجام این کار ، در کادر مکالمه ای که باز می شود نام بلوک اطلاعاتی را به **DB10** تغییرداده و ما بقی تنظیمات را با استفاده از **OK** تایید نمایید.

بلوک اطلاعاتی **DB10** ایجاد می شود. این بلوک را باز نموده تا کادر مکالمه "**New Data Block**" را مشاهده نمایید.

انتخاب **Data Block Refering a Function Block** را فعال نموده و **FB10** را انتخاب نمایید. تنظیمات را با استفاده از **OK** تایید نمایید.



بلوک اطلاعاتی **DB10** باز می شود. فرمان **View > Data View** را انتخاب نمایید.



◙ **Data View** هـر متغـیر مسـتقل (**Individual**) را در **DB10** کـه شـامل متغـیرهای داخلـی **(Internal)** در فراخوانی مربوط به **FB1 (Local Instances)** میشود را نمایش میدهد. **declaration View** متغیرها را همانگونه که در **FB10** نشان داده شده اند نمایش میدهد.

مقدار واقعی موتور دیزلی را به **1300** تغییر داده ، بلوک را ذخیره نموده و آنرا ببندید.

Address	Decl.	Name	Type	Initial Value	Actual Value	Comment
0.0	out	Preset_Speed_Reached	BOOL	FALSE	FALSE	Both engines have reached
2.0	stat:in	Petrol_Engine.Switch_On	BOOL	FALSE	FALSE	Switch on engine
2.1	stat:in	Petrol_Engine.Switch_Off	BOOL	FALSE	FALSE	Switch off engine
2.2	stat:in	Petrol_Engine.Failure	BOOL	FALSE	FALSE	Engine failure, causes ths
4.0	stat:in	Petrol_Engine.Actual_Speed	INT	0	0	Actual engine speed
6.0	stat:out	Petrol_Engine.Engine_On	BOOL	FALSE	FALSE	Engine is switched on
6.1	stat:out	Petrol_Engine.Preset_Speed_Reached	BOOL	FALSE	FALSE	Preset speed reached
8.0	stat	Petrol_Engine.Preset_Speed	INT	1500	1500	Requested engine speed
10.0	stat:in	Diesel_Engine.Switch_On	BOOL	FALSE	FALSE	Switch on engine
10.1	stat:in	Diesel_Engine.Switch_Off	BOOL	FALSE	FALSE	Switch off engine
10.2	stat:in	Diesel_Engine.Failure	BOOL	FALSE	FALSE	Engine failure, causes ths
12.0	stat:in	Diesel_Engine.Actual_Speed	INT	0	0	Actual engine speed
14.0	stat:out	Diesel_Engine.Engine_On	BOOL	FALSE	FALSE	Engine is switched on
14.1	stat:out	Diesel_Engine.Preset_Speed_Reached	BOOL	FALSE	FALSE	Preset speed reached
16.0	stat	Diesel_Engine.Preset_Speed	INT	1500	1300	Requested engine speed

حـال جـدول نمـایش متغـیرهای **DB10** کلـیه متغـیرها را در بـر مـی گیرد. شما در نیمه اول آن متغیرهای مربوط به فراخوانی بلوک تابعی "**Petrol – Engine**" و در نیمه دوم آن متغیرهای مربوط به فراخوانی بلوک تابعی " **Diesel – Engine**" را می توانید ملاحظه کنید. (بخش 5.5 را ببینید.)

متغـیرهای داخلی **FB1 (Internal)** اسامی سمبلیک خود را حفظ می کنند مثل: "**Switch-on**". در صورتیکه اسامی **Local Instance** ها در جلوی اسامی شان قرار می گیرد مثل:

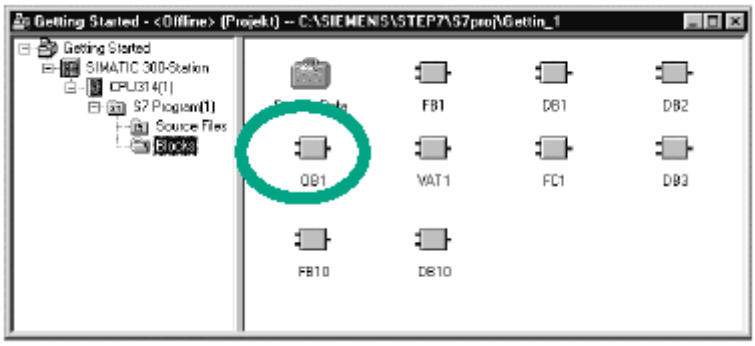
"**Petrol_Engine.Switch_On**"

بـرای اطلاعـات بیشـتر می توانید به مباحث "**Programming Blocks**" و "**Creating Data Blocks**" واقع در **Help > Contents** مراجعه نمایید.

10.4 فراخوانی FB10 در داخل OB1

ما در مثالمان فراخوانی **FB10** را در داخل **OB1** انجام داده ایم. این فراخوانی بهمان صورتی که قبلا در مـورد فراخوانـی **FB10** در داخل **OB1** عمل نمودید انجام می پذیرد. (بخش 5.6 به بعد را ببینید.) شـما با استفاده از **Multiple instance** ها می توانید **Network**های 4 و 5 را که از بخش 5.6 به بعد ایجاد نمودید جایگزین نمایید.

در پروژه ای که هم اکنون **FB10** را درون آن ایجاد نمودید **OB1** را باز نمایید.



■ در صـــورتیکه شـــما در فصـــل 4 جـــدول ســـمبل هـــا را از یـــک پـــروژه نمونـــه **zEn01_04_STEP7_FBD_1-10** ،**zEn01_06_STEP7_LAD_1-10** ،**zEn01_02_STEP7_STL_1-10)** به پروژه **Getting Started** کپی نموده اید دیگر نیازی به اضافه نمودن سمبل دیگری ندارید.

تعریف اسامی سمبلیک

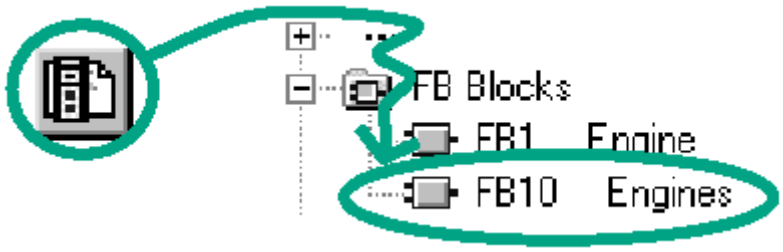
در حالـــیکه پـــنجره بـــرنامه **LAD/STL/FBD** بـــاز اســـت بـــا اســـتفاده از دســـتور **Options>Symbol table** جدول سمبل ها را باز نموده و اسامی سمبلیک مربوط به بلوک تابعی **FB10** و بلوک اطلاعاتی **DB10** را در آن وارد نمایید.

سپس جدول سمبل ها را ذخیره نموده و پنجره را ببندید.

Symbol	Address	Data Type	Comment
...	...	...	...
Engines	FB 10	FB 10	Example of multiple instances
Engine_Data	DB 10	FB 10	Instance data block for FB10 10
...	...	...	...

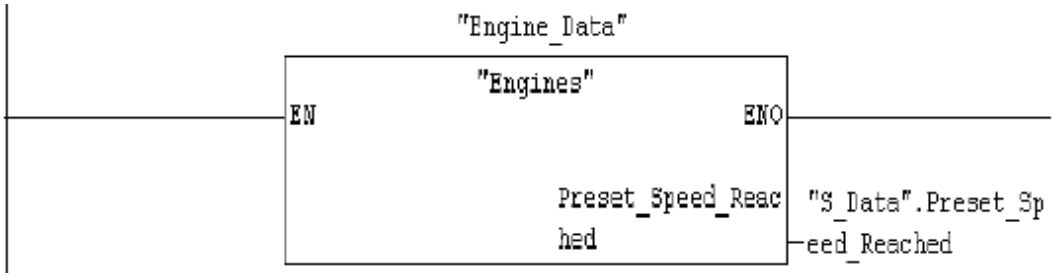
نحوه فراخوانی بزبان منطق نردبانی

در انـتهای **OB1** یـک **Network** جدیـد ایجـاد نمـوده و فراخوانـی **FB10 (Engines)** را در آن تعریف نمایید.



بااسـتفاده از اسـامی سمبلیک مربوطه ، دستور فراخوانی را بصورت زیر تکمیل نمایید. چون حالا ما **FB1** را درداخل **FB10** فراخوانی می نماییم قسمت های مربوط به فراخوانی **FB1** در داخل **OB1** را (**Network** های 4 و 5 از بخش 5.6 به بعد) حذف نمایید.

سپس برنامه تان را ذخیره نموده و بلوک را ببندید.



◙ سیگنال خروجی "**Setpoint_Reached**" مربوط به **FB10** ("**Engines**") به یک متغیر مربوط به بلوک اطلاعاتی مشترک اعمال شده است.

نحوه فراخوانی بزبان نمایش عبارتی

اگـر در حال برنامه نویسی به زبان نمایش عبارتی هستید ، فضای وارد نمودن ورودیها واقع در زیر یک **Network**جدید را انتخاب نموده و دستورات **STL** زیر را وارد نمایید.

بـرای ایـن مـنظور در داخل کاتالوگ عناصر برنامه از **FB Blocks** > **FB10 Engines** استفاده نمایید.

چـون حـالا ، **FB1** را در داخـل **FB10** فراخوانی می نماییم قسمت های مربوط به فراخوانی **FB1** در

داخـل **OB1** را (**Network** هـای 5 و 4 از بخش 5.6به بعد) حذف نمایید. سپس برنامه تان را ذخیره نموده و بلوک را ببندید.

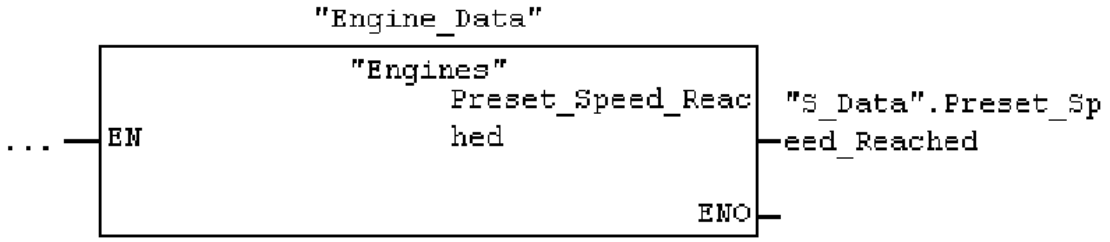
```
CALL  "Engines" , "Engine_Data"
 Preset_Speed_Reached:="S_Data".Preset_Speed_Reached
```

نحوه فراخوانی بزبان FBD

اگـر در حـال نوشـتن بـرنامه بـزبان **FBD** میباشـید ، فضـای وارد نمـودن ورودی واقع در زیر یک **Network** جدید را انتخاب نموده و دستورات **FBD** زیر را وارد نمایید.

بـرای ایـن مـنظور در کاتـالوگ عناصـر بـرنامه از **FB Blocks > FB10 Engines** اسـتفاده نمایـید.

چـون حـالا ما **FB1** را در داخل **FB10** فراخوانی می نماییم قسمت های مربوط به فراخوانی **FB1** در داخـل **OB1** را (**Network** های 4 و 5 از بخش 5.6 به بعد) حذف نمایید. سپس برنامه تان را ذخیره نموده و بلوک را ببندید.



◙ در صـورتیکه در کار اتوماسیون تان نیاز به برنامه های کنترل موتور بیشتری داشته باشید ؛ مثلاً برای موتورهای گازسـوز ، موتورهـای هیدروژنـی و غیره ، میتوانید آنها را بهمان ترتیب و بصورت **Multiple Instance** نوشته و در داخل **FB10** فراخوانی نمایید.

بـرای ایـن مـنظور همانطوریکـه در جدول نمایش متغیرهای مربوط به**FB10 (Engines)** نمایش داده شد موتورهای اضافی را قید نموده و **FB1** را در داخل **FB10** فراخوانی نمایید.

(**Multiple Instance** در کاتالوگ عناصر برنامه).

شما میتوانید اسامی سیمبلیک جدیدی مثل مراحل روشن شدن و خاموش شدن را در جدول سمبل ها تعریف نمایید.

◙ بـرای اطلاعـات بیشـتر به مباحث "**The STL , FBD Or LAD Language Description**" و " **Program Control Instruction**" واقع در **Help > Contents** مراجعه نمایید.

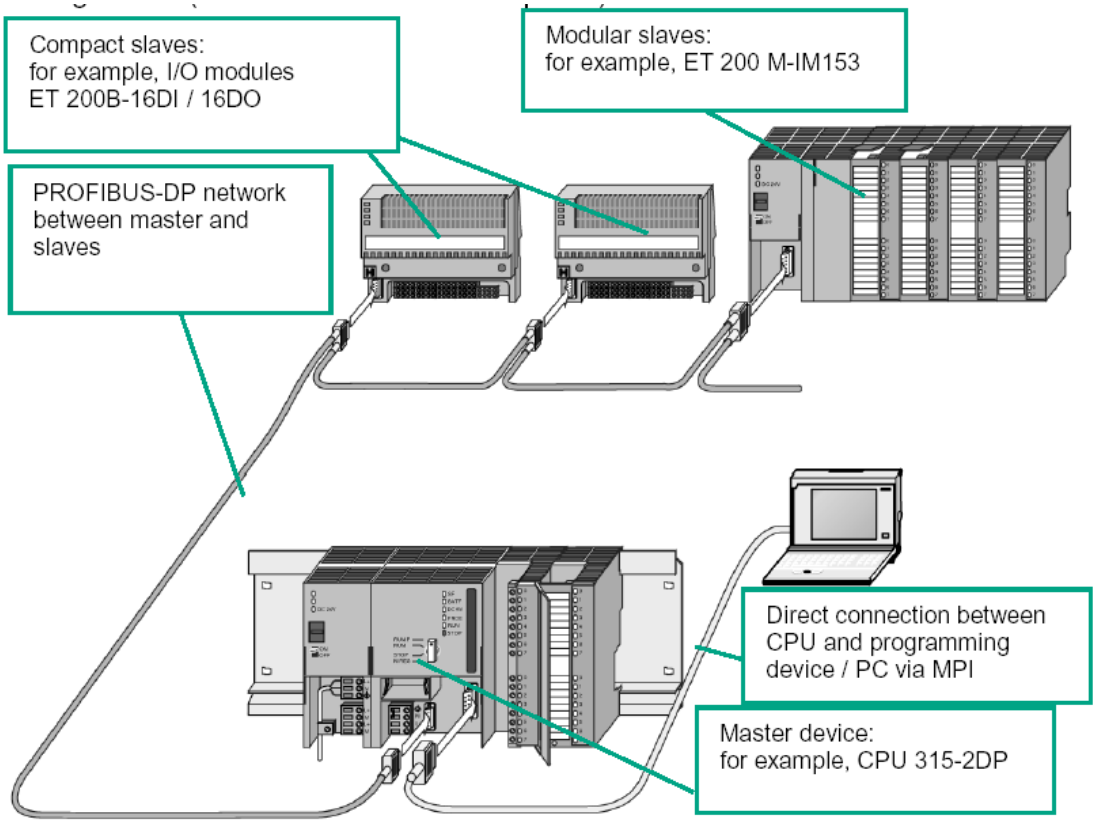
11 پیکربندی ورودی / خروجی توسعه یافته

11.1 پیکربندی ورودی / خروجی توسعه یافته توسط PROFIBUS DP

در پیکربندی سیستم های اتوماسیون معمول ، سنسورها وعملگرها بطور مستقیم با استفاده از کابل کشی به کارتهای ورودی / خروجی **PLC** مرکزی متصل می شوند. این به معنای یک مقدار قابل توجه عملیات کابل کشی میباشد.

با استفاده از پیکربندی توسعه یافته می توان از طریق قراردادن کارتهای ورودی و خروجی نزدیک سنسورها و عملگرها میزان کابل کشی را تا حد قابل ملاحظه ای کاهش داد. شما می توانید ارتباط بین **PLC** ، کارتهای ورودی / خروجی و تجهیزات **Field** را با استفاده از **Profibus DP** برقرار سازید. برای اطلاع از نحوه انجام پیکربندی می توانید به فصل 6 مراجعه نمایید. تفاوتی بین ایجاد یک پیکربندی مرکزی یا یک پیکربندی توسعه یافته وجود ندارد. شما میبایست از داخل کاتالوگ سخت افزاری کارتهای مورد نیازتان را انتخاب نموده و آنها را در داخل **Rack** قرار دهید و سپس مشخصات آنها را بر حسب نیازتان تنظیم نمایید.

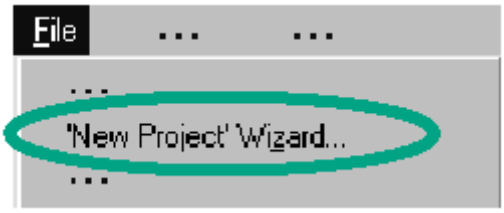
قبل از خواندن این فصل بهتر است با ایجاد یک پروژه و برنامه نویسی یک پیکربندی مرکزی آشنا باشید (بخش 2.1 و فصل 6 را مطالعه کنید.)



ایجاد یک پروژه جدید

نقطه شروع از **SIMATIC Manager** میباشد. برای راحتی کارتان تمام پروژه های فعال را ببندید.

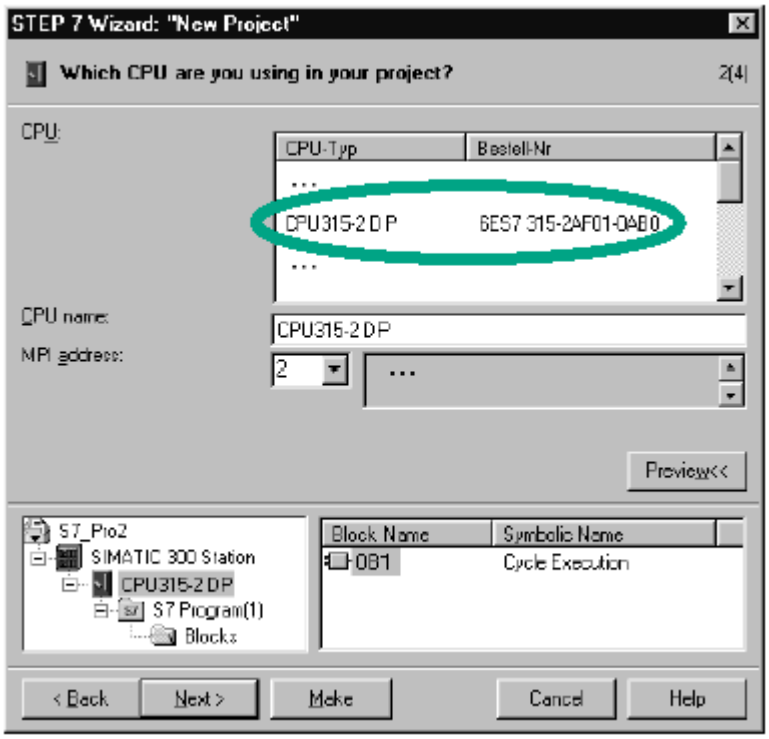
یک پروژه جدید ایجاد نمایید.



از کادر مکالمه مربوطه **CPU 315-2DP** را انتخاب نمایید (**CPU** همراه با شبکه **Profibus-DP**).

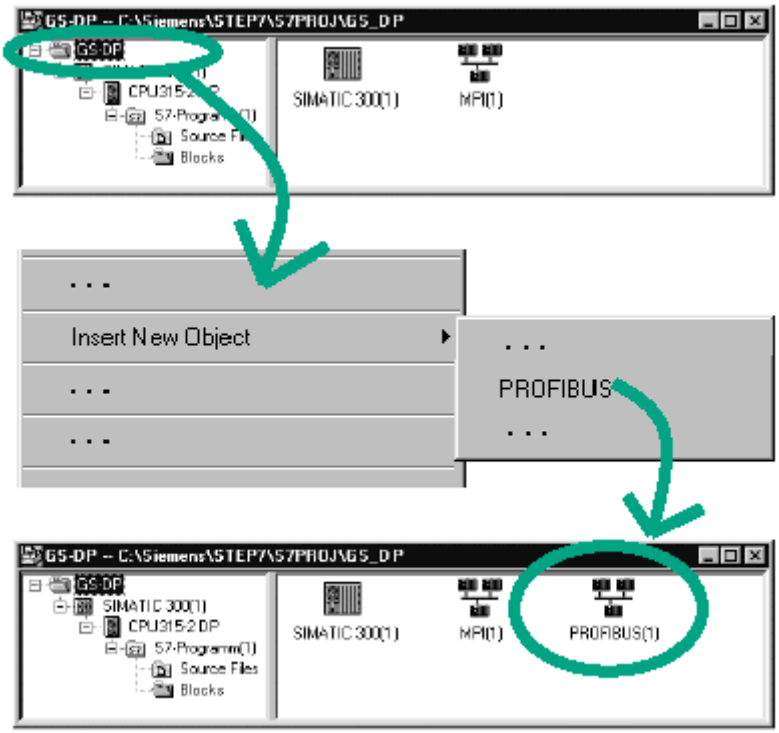
حـال مطابق بخش 2.1 عمل نموده ونام **"GS_DP"(Getting Started Distributed I/O)** را به پروژه نسبت دهید.

اگـر قصـد داریـد پیکربـندی را دلخـواه خودتـان تعریف نمایید میبایست **CPU** را هم اکنون مشخص نمایـید. توجـه داشـته باشید که **CPU** تان میبایست ورودی / خروجی های توسعه یافته را پشتیبانی نماید.



اضافه نمودن شبکه Profibus

پوشـه **GS – DP** را انـتخاب نمـوده و بـا استفاده از کلیک راست موس در نیمه راست پنجره ، شبکه **PROFIBUS** را ایجاد نمایید.

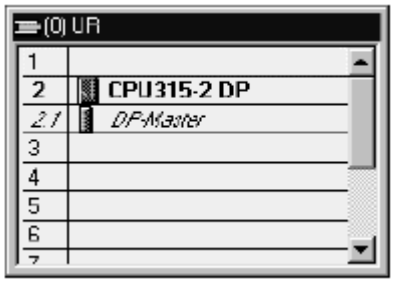


پیکربندی ایستگاه

پوشــه **SIMATIC 300 Station** را انـتخاب نموده و بر روی **Hardware** دوبار کلیک نمایید. پنجره "**HW Config**" باز می شود. (بخش 6.1 را ملاحظه کنید)

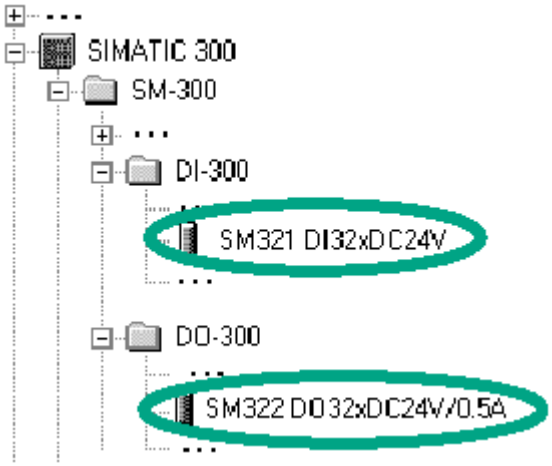
CPU 315-2 DP درداخل **Rack** پدیدار می شود.

در صـورت لـزوم بااسـتفاده از دسـتور **View > Hardware Catalog** یـا دکمه های مربوطه در خط ابزار ، کاتالوگ سخت افزاری را بازنمایید.



کارت منبع تغذیه **PS 307 2A** را کشیده و در شیار **1** رها کنید.

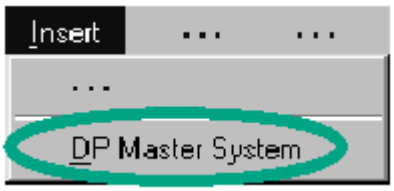
بهمان صورت کارتهای ورودی / خروجی **DI32×DC24V** و **DO32×DC24V/0.5A** را در شیارهای 4 و 5 قرار دهید.

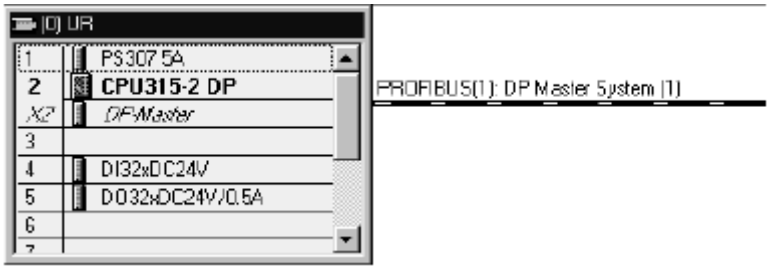


◙ شما علاوه بر **CPU** تان که ورودی / خروجی های توسعه یافته را پشتیبانی می کند می توانید **CPU** های دیگری را نیز در همان **Rack** قرار دهید (در اینجا توضیح داده نمی شود).

پیکربندی سیستم DP-master

DP Master را در شیار 2.1 انتخاب کرده و یک **DP – master system** ایجاد نمایید.

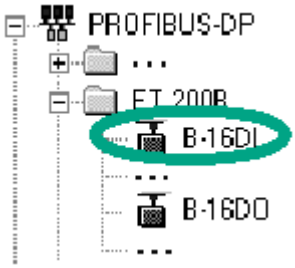




■ شـما می توانید هر چیزی را بخواهید با استفاده از کشیدن آنها در حالیکه دکمه چپ موس را فشار میدهید به مکان مورد نظرتان در **Master System** منتقل سازید.

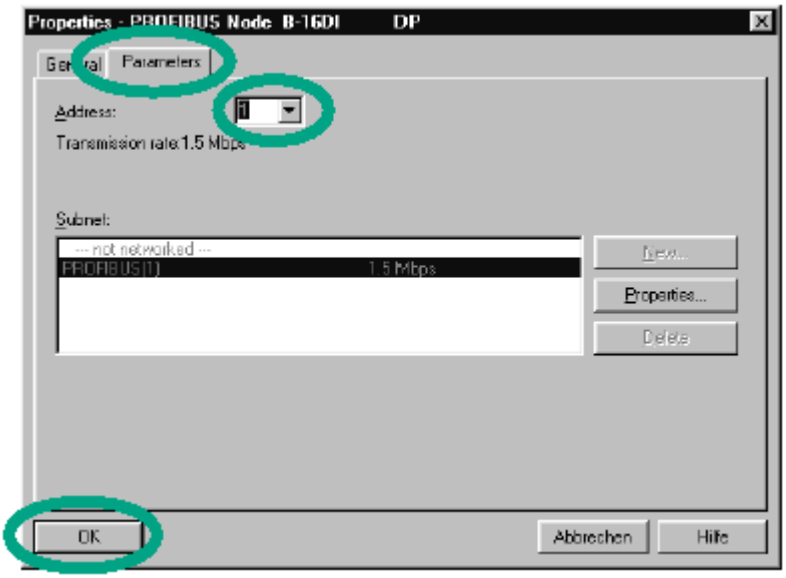
در قسـمت کاتـالوگ سـخت افزاری کارت **B – 16DI** را یافته و آنرا در **Master System** وارد نمایید.

(آنرا به **Master System** بکشید تا وقتیکه مکان نما به علامت "+" تبدیل شود سپس آنرا رها سازید.)



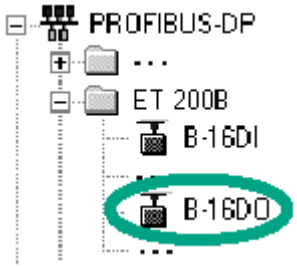
شـما بوسـیله**tab** "**Network Connection**" مـربوط بـه کـار مکالمـه "**Properties**" می توانید آدرس **Node** کارت را تغییر دهید.

آدرس پیشنهادی **1** را توسط **OK** تایید نمایید.

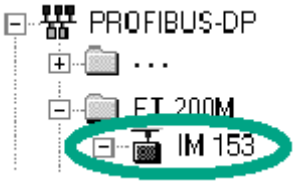


بهمین روش کارت **B – 16 DO** را به **Master System** منتقل نمایید.

آدرس **Node** بطـور خودکـار در کـادر مکالمه تعریف می گردد. این مقدار را با استفاده از **OK** تایید نمایید.

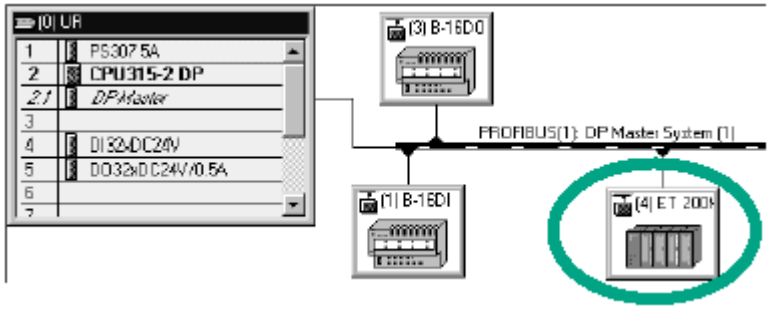


کـارت **Interface** **IM 153** را بـه **Master System** منـتقل نموده و آدرس **Node** را مجدداً توسط **OK** تایید نمایید.

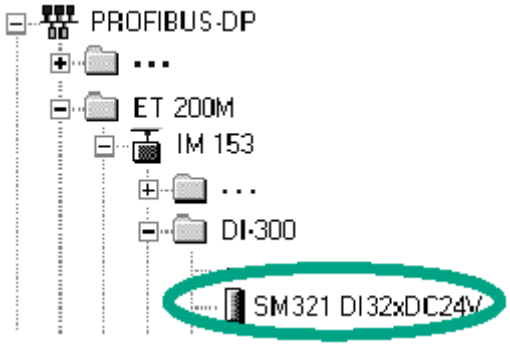


■ در ایـنجا ما از آدرس های **Node** پیش گزیده استفاده می کنیم اما شما در هر زمانی که نیاز پیدا کردید می توانید این آدرس ها را تغییر دهید.

M 200 ET را در **Network** انـتخاب نمایـید. شـیارهای خالـی مـربوط بـه **M 200 ET** در جـدول پیکربندی پایینی بنمایش در می آید. در اینجا شیار 4 را انتخاب نمایید.



خـود **ET 200 M** مـی توانـد کارتهـای ورودی / خروجـی دیگـری نـیز داشـته باشـد. کـارت **DI32×DC24V** را بعـنوان نمونه برای شیار 4 انتخاب کرده و دوبار بر روی آن کلیک نموده تا وارد شود.

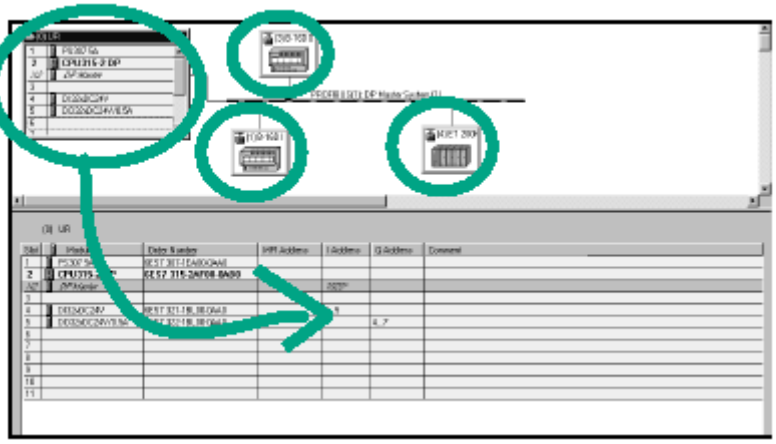


■ هـنگام اسـتفاده از کاتالوگ سخت افزاری همواره از بودن در پوشه مورد نظرتان اطمینان حاصل نمایید. مثلاً برای انتخاب کارت جهت **ET 200M** باید بداخل **ET 200M** رجوع نمایید.

تغییر آدرس Node

در مـثال مطـرح شده ما نیازی به تغییر آدرس **Node** نداریم. اما اغلب در عمل نیاز به انجام این کار وجود دارد.

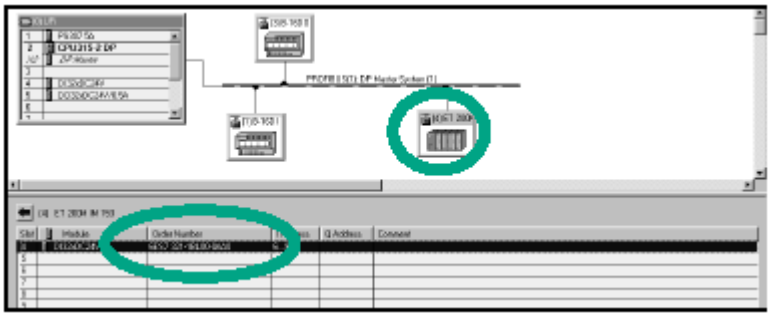
Node را یکی پس از دیگری انتخاب کرده و آدرس های ورودی و خروجی را بررسی نمایید.



همانطور که ملاحظه می شود "**Configuring Hardware**" کلیه آدرس ها را طوری ترتیب داده است که هیچ آدرس مشترکی وجود ندارد.

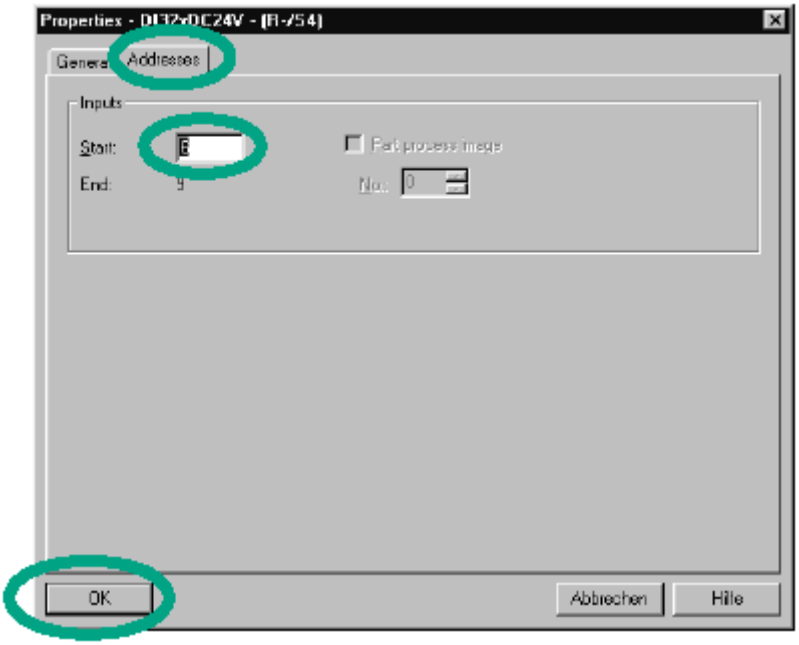
فرض کنیم قصد داشته باشید که آدرس **ET 200 M** را تغییر دهید :

ET 200 M را انتخاب نموده و بر روی **DO32×DC24V/0.4A** (شیار 4) دوبار کلیک نمایید.



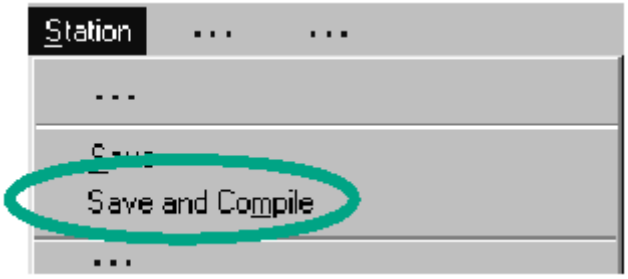
حـال در **tab** "**Addresses**" مربوطه به کادر مکالمه "**Properties**" آدرس ورودی را از 6 به 12 تغییر دهید.

با استفاده از **OK** کادرمکالمه را ببندید.



سرانجام پیکربندی ورودی / خروجی توسعه یافته را **Save and Compile** نمایید.

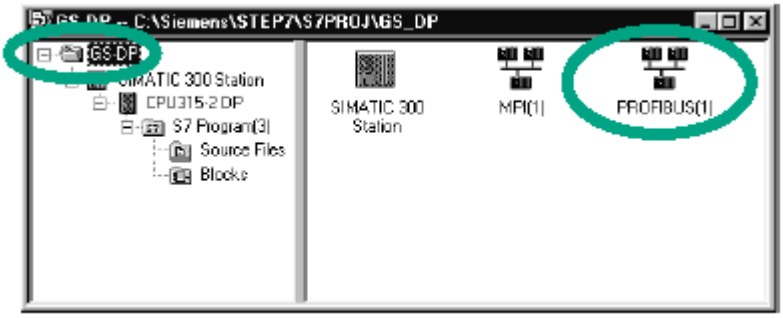
پنجره را ببندید.



◙ فرمان **Save and Complile** بمنزله آنست که پیکربندی بطور خودکار از لحاظ **Consistency**مورد بررسی قرار می گیرد. اگر خطایی وجود نداشته باشد ، اطلاعات سیستمی ایجاد شده و می تواند به **PLC** منتقل **(Download)** شود. با استفاده از **Save** شما می توانید پیکربندی تان راحتی در صورت وجود خطا ذخیره نمایید. اما در مرحله بعد شما قادر به انتقال **(Download)** پیکربندی تان به **PLC** نخواهید بود.

اختیاری : پیکربندی شبکه ها

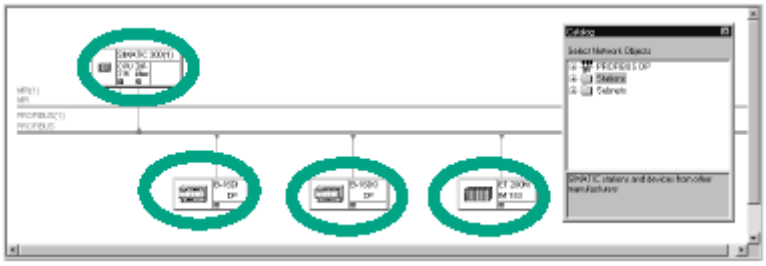
شما همچنین با استفاده از بسته نرم افزاری **"Configuring network"** قادر به پیکربندی ورودی / خروجی توسعه یافته میباشید. در **SIMATIC Manager** بر روی**(1) Profibus** دوبار کلیک نمایید.



پنجره "NETPRO" گشوده می شود.

شما از طریق کاتالوگ عناصر شبکه می توانید **DP Slave** های دیگری را بر روی **Profibus DP** بکشید و رها نمایید.

بر روی هر عنصری که بخواهید آنرا پیکربندی نمایید دوبار کلیک نمایید. پنجره "**Configuring Hardware**" باز می شود.



◙ با استفاده از دستور **Station>Consistency check** (در پنجره"**Configuring Hardware**") و **Network > Consistenency check** (در پنجره"**Configuring Network**") می توانید قبل از ذخیره نمودن پیکربندی ، آنرا از نظر خطا بررسی نمایید.

هر خطایی که در داخل **STEP7** نمایش داده شود راه حل پیشنهادی ای نیز برای آن نمایش داده خواهد شد.

◙ برای اطلاعات بیشتر به مباحث "**Configuring the Hardware**" و "**Configuring the distributed I/O**" واقع در **Help > Contents** مراجعه نمایید