فصل 1

کتابخانه فانکشن بلوکهاي OMRON

Function Block

پيشگفتار :

با پيشرفت روز افزون صنايع در سرتاسر جهان ، خصوصاً صنعت اتوماسيون و گسترش نيازهاي رو به رشد صنعتي ، ضرورت پرهيز از به كار گيري مدارات پيچيده ، زمان بر و با عيب يابي هاي دشوار بيش از پيش ملموس شده است .

در دهه هاي اخير كنترلرهاي صنعتي در رنج وسيعي جايگزين مدارات رله- كنتاكتوري گشته و با توجه به مزاياي فوق ا لعاده زيادشان از جمله

- قابليت اطمينان بالا
- عيب يابي راحت
- سرعت بالا ي اجراي فرآيند
- امكان استفاده از دستورالعملهاي پيشرفته براي كنترل فرآيندهاي پيچيده

استفاده از اين كنترلر ها به سرعت در اكثر صنايع به يكي از ملزومات تبديل شده است .

لذا كارخانه هاي سازنده كنترلرهاي صنعتي همواره به دنبال ساده سازي هر چه بيشتر عملكرد اين كنترلرها براي كاربران و اضافه نمودن قابليتهاي كارا و جديد به آنها مي باشند .

يكي از قديمي ترين شركتهاي سازندة كنترلر هاي صنعتي ، شركت OMRON مي باشد كه در دهه هاي اخير با سرعت چشمگيري كنترلرهاي صنعتي جديد و با قابليتهاي فراوان را در اختيار كاربران قرار داده و هر روزه بر آن است تا حد امكان استفاده از اين كنترلر ها را ، براي كاربران ساده و راحتتر نمايد .

در راستاي هدف فوق در سري هاي جديد نرم افزار برنامه نويسي كنترلر هاي صنعتي شركت OMRON(CX-Programmer) ، قابليت امكان تعريف فانكشن بلوكها اضافه شده است كه امكان برنامه نويسي هر چه راحتتر و قابل فهم تر ،را براي كاربر فراهم مي نمايد .

اين قابليت مزاياي بسيار زيادي را در اختيار كاربران قرار مي دهد كه از آن جمله مي توان به

- ساز گاري با استاندارد برنامه نويسي IEC 61131- 3
- ساده سازي هر چه بيشتر برنامه نويسي
- درك بهترروند كلي برنامه
- پرهيز از نمايش مراحل داخلي برنامه و دسترسي هر چه راحتتر به ورودي و خروجي هاي اصلي
- وجود فانكشن بلوك هاي از پيش تعريف شده در كتابخانه نرم افزار و امكان استفاده از انها بدون نياز به خطايابي و تست صحت عملكرد
- وجود فانكشن بلوكهاي از پيش تعريف شده مربوط به برقراري ارتباط با تجهيزات صنعتي شركت OMRON در كتابخانه نرم افزار و امكان استفاده از آنها
- اختصاص زمان بيشتر به روند اصلي برنامه، زيرا با استفاده از فانكشن بلوكهاي از پيش تعريف شده ، ديگر نياز به نوشتن برنامه هاي لازم جهت برقراري ارتباط با تجهيزات جانبي متصل به كنترلر نمي باشد .
- امكان نوشتن برنامه هاي فانكشن بلوك به يكي از زبانهاي نردباني و يا ST ، متناسب با نيازتان
- امكان استفاده از بسياري از دستورات محاسباتي و منطقي ، بااستفاده از زبان برنامه نويسي ST

اشاره نمود .

لذا در اين جزوه بر آن هستيم شما را با اين قابليت جديد و كارا آشنا نماييم .

به اميد آنكه بتوانيم گامي هر چند كوچك در آشنايي بهترشما با كنترلرهاي صنعتي و چگونگي برنامه نويسي آنها برداشته باشيم

تهيه وتنظيم :
مهندس آسيه خوش خلق
بهار 1385

مقدمه :

در اين جزوه با

- **چگونگي استفاده از فانكشن بلوكهاي از پيش تعريف شدة موجود در كتابخانه OMRON FB**
- **ايجاد فانكشن بلوكهاي جديد**

با استفاده از نرم افزار CX- Programmer نسخه 5 به بالا،كه قابل استفاده براي برنامه ريزي PLC هاي امرن ، سري CS1/CJ1- H /CJ1M (نسخه 3 به بالا) مي باشند آشنا خواهيد شد .

توجه :

برنامه اي را كه در آن از فانكشن بلوك استفاده نموده ايد را تنها، براي CPU هاي سري CS1 / CJ1- H / CJ1M نسخه 3 به بالا مي توانيد به كار برد .

اگر سعي بر بازنشاني[1] برنامه اي كه در آن از فانكشن بلوك استفاده شده است ،از طريق نر م ا فزار CX- Programmer نسخه 5 بـه پـايين داشته باشيد ، از آنجا كه اين نسخه[2] ها ، فراخواني فانكشن بلوك را پوشش نمـي دهند، قـادر بـه بازنـشاني برنامـه نخواهيد بود و در صورت بازنشاني نيز ، برنامـه نـاقص بازنشاني خواهد شد .

درصورت استفاده از CX- Programmer نسخه 5 :

پيغامهاي زير بعد از اتمام بازنشاني نمايش داده خواهد شد :

The PLC properties which are not supported by this version of CX-Programmer are set in the connecting target PLC.
The PLC properties will not be displayed correctly. Do you wish to continue?

درصورت استفاده از CX- Programmer نسخه 4 :

پيغام زير بعد از اتمام بازنشاني نمايش داده خواهد شد :

Function Block or other data besides Ladder are included in the programs.

درصورت استفاده از CX- Programmer نسخه 3 :

پيغام زير بعد از اتمام بازنشاني نمايش داده خواهد شد :

"Decompile error" :

---

[1] - Upload

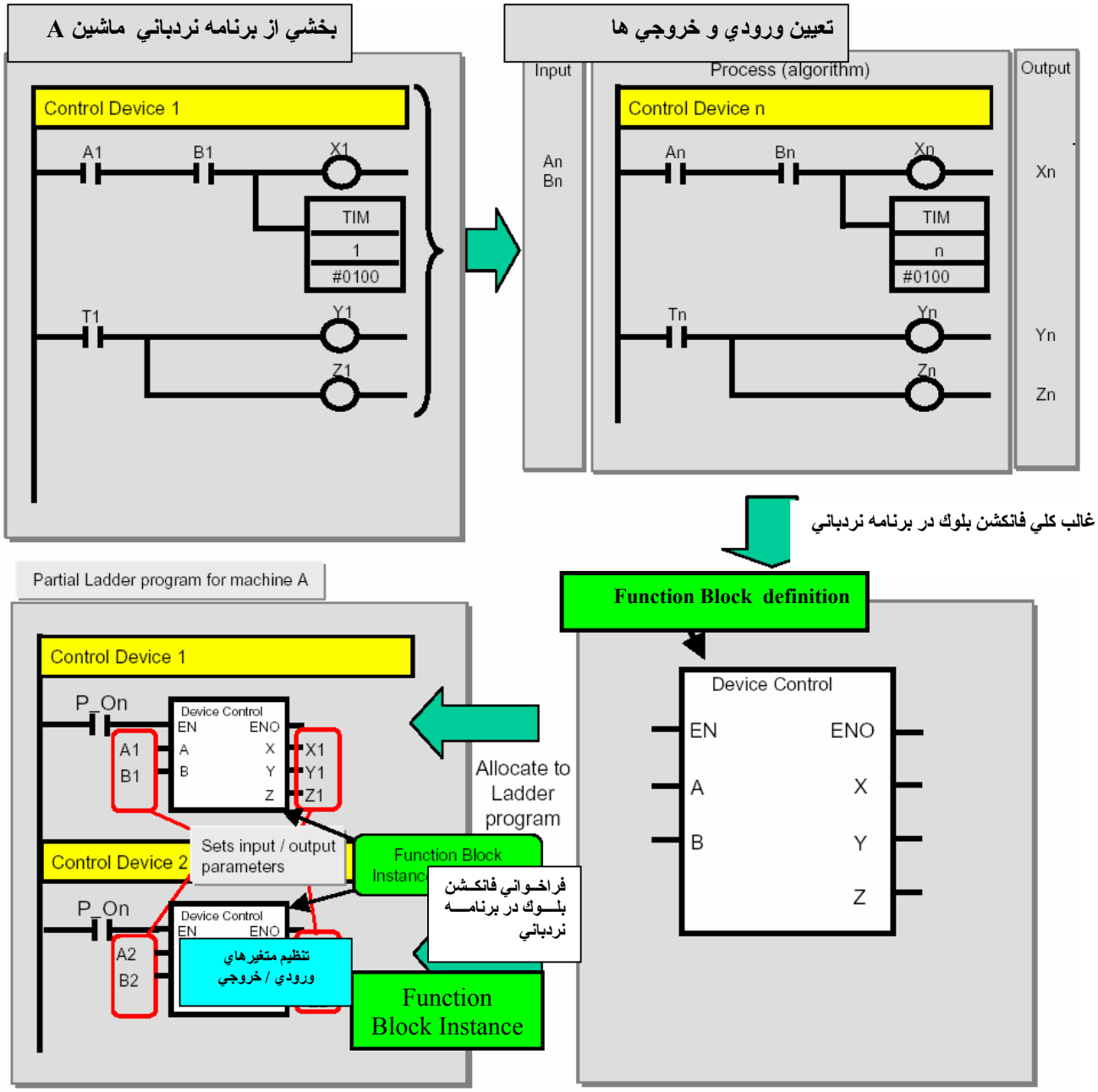
[2] - Version

1- فانكشن بلوك چيست ؟

فانكشن بلوك برنامه اي از پيش تعريف شده مي باشد كه در برنامه اصلي به صورت يك دستور مستقل فراخواني مي شود.در هر فانكشن بلوك وجود يك كنتاكت فعال ساز (EN) براي آغاز به كار، مورد نياز است .اما متغيرهاي ورودي و خروجي تنها در برنامة نردباني كه در آن فانكشن بلوك را فراخواني نموده ايد ،قابل اصلاح ويا تغييرمي باشند .

از يك فانكشن بلوك ميتوان بيش از يك بار در برنامه نردباني و به منظور عمليات كنترلي متفاوت استفاده نمود .

كه اين امر در شكل ز ير كاملاً مشهود است



: Function Block definition

شامل واسط هاي ورودي / خروجي و الگوريتم يك عملكرد مشخص مي باشد .

در تعريف فانكشن بلوك متغيرها را آدرس دهي نمي نماييد بلكه متغيرهاي معادل ورودي / خروجي ها را ، بدون تخصيص آدرس و باعنواني متناسب با عملكردشان تعريف مي نماييد .

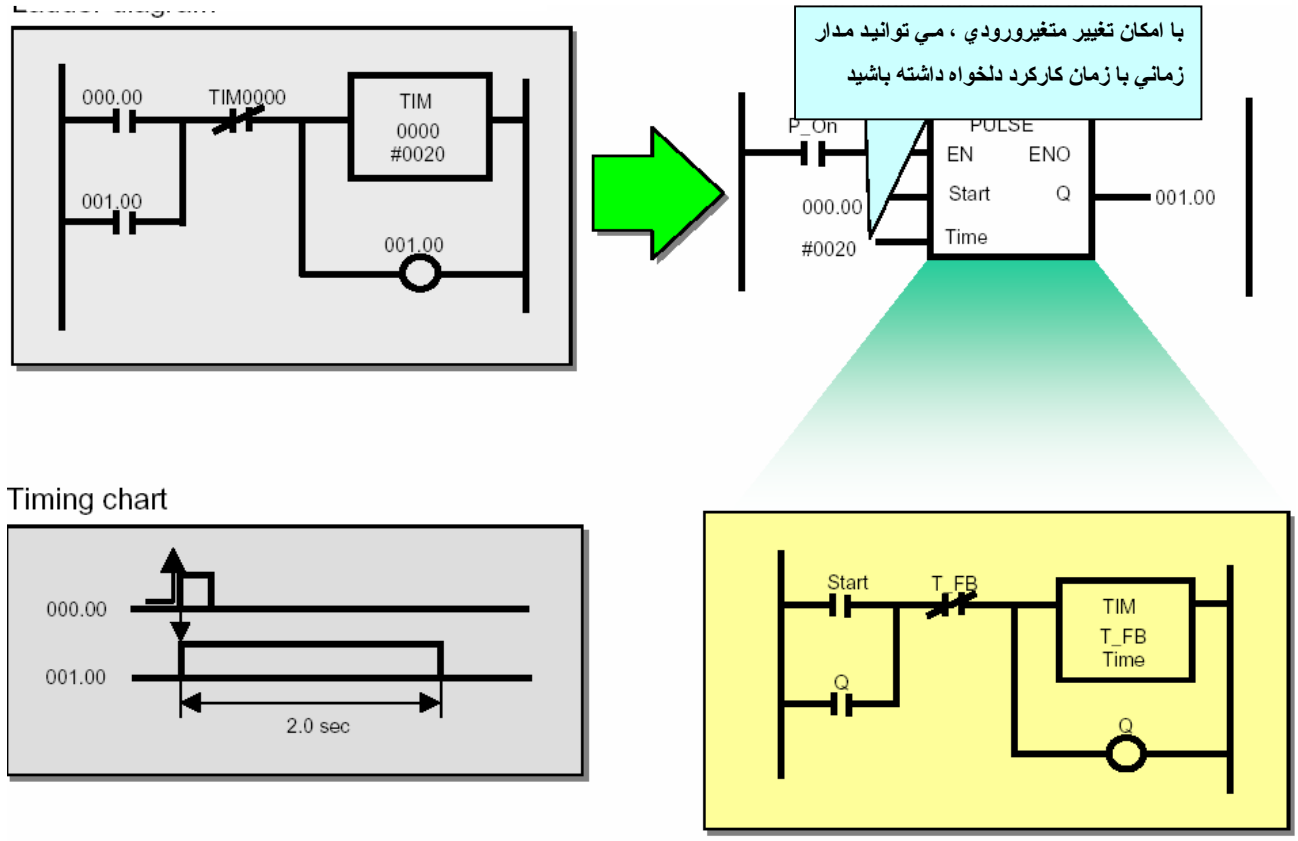
:Function Block Instance

فانكشن بلوك به صورت يك علامت گرافيكي در برنامه نردباني استفاده و فراخواني مي شود و متغيرهاي ورودي و خروجي آن پس از فراخواني در برنامه نردباني ،آدر س دهي مي گردند .

2. يك مثال از فانكشن بلوك :

شكلهاي زير مثالي از كاربرد فانكشن بلوك ، در كنترل يك مدار زماني و مقايسه آن با برنامه نردباني را نشان ميدهد .در برنامه نردباني زير ، با تغيير مقادير تنظيمي تايمر، مي توانيد زمان خاموش شدن خروجي راكنترل نماييد .در صورت استفاده از فانكشن بلوك همانطور كه در شكل نمايش داده شده است اين امكان وجود دارد كه زمان دلخواه كاركرد مدار را تنها با تغيير متغير مربوط به زمان كاركرد (Time) تغيير دهيد .

برنامه نردباني بدون استفاده از فانكشن بلوك



در واقع برنامه نردباني مربوطه را در غالب يك فانكشن بلوك ، به زبان نردباني تعريف مي نماييد و سپس در هر جاي برنامه كه لازم باشد اين فانكشن بلوك را فراخواني و متغيرهاي آن را مقدار دهي مي نماييد .

طرز كار اين فانكشن بلوك :

بلافاصله بعد از تحريك كنتاكت Start و فعال شدن آن ، خروجي Q روشن شده و مستقل از روشن يا خاموش بودن كنتاكت استارت به مدت زمان تنظيمي (قابل تعيين توسط كاربر) روشن بوده و بعد از اين مدت تا زمانيكه مجدداً كنتاكت استارت تحريك گردد خاموش خواهد ماند .

اين برنامه با تعريف در غالب يك فانكشن بلوك كلي و فاقد هر گونه آدرس كه در بالا نمايش داده شده است ، ايجاد شده و به راحتي مي توانيد آن را در برنامه اصلي فراخواني و با آدرس دهي و زمان دهي هاي متفاوت براي كاربردهاي مجزا به كار برد .

3. نمايي كلي از كتابخانه OMRON FB :

اين كتابخانه مجموعه اي از فانكشن بلوكهاي از پيش تعريف شده ميباشد، كه توسط شركت امرن ايجاد شده است .
اين فايلها به منظور كمك به ساده سازي برنامه نويسي ايجاد شده اند و شامل فانكشنهاي استاندارد ، براي برنامه نويسي PLC و نيزتوابع كنترلي مربوط به ايجاد ارتباط با تجهيزات صنعتي OMRON ميباشند.

3-1 مزاياي استفاده از كتابخانه OMRON FB

همانطور كه گفته شد كتابخانه OMRON FB يك مجموعه از فانكشن بلوكهاي استاندارد، كه به منظور بهينه كردن برقراري ارتباط ما بين واحد هاي PLC و ساير تجهيزات كنترلي ، ساخته شده توسط شركت OMRON تهيه شده اند ميباشدو به كار گيري آنها در برنامه نويسي مزاياي بسياري را دارا مي باشد كه در زير خلاصه برخي از آنها را بر خواهيم شمرد :

(1) به منظور برقراري ارتباط PLC با ساير تجهيزات كنترلي (به طور مثال اينورترها ، كنترل كننده هاي دما سرو درايوها و.....) ، ديگر نيازي به نوشتن برنامه نردباني ، جهت اين ارتباط نمي باشد

در نتيجه زمان بيشتري را مي توان به جاي ايجاد برنامه هاي پايه بر روي منطق لازم برنامه ، جهت ارتباط با دستگاه هاي ديگر صرف نمود .
زيرا برنامه هاي پاية ارتباطي با بسياري از تجهيزات امرن از پيش تعريف شده و در كتابخانه موجود مي باشد .

(2) استفاده آسان

برنامه هاي كاربردي به آساني و تنها با فراخواني فانكشن بلوك مربوطه ، فراخواني و اجرا خواهند شد و بدين منظور كافيست علامت گرافيكي فانكشن بلوك مربوطه را د ربرنامه نردباني فراخواني نموده و متغيرهاي آن را مقدار دهي و آدرسهاي ورودي / خروجي هاي آن را تنظيم نماييد .

(3) نيازي به تست عملكرد صحيح برنامه وجود ندارد

شركت OMRON صحت عملكرد فانكشن بلوكهاي از پيش تعريف شده اش را تست نموده است ودر نتيجه لزومي به خطايابي اين برنامه هاي از پيش تعريف شده ، توسط كاربر وجود ندارد .

(4) درك آسان

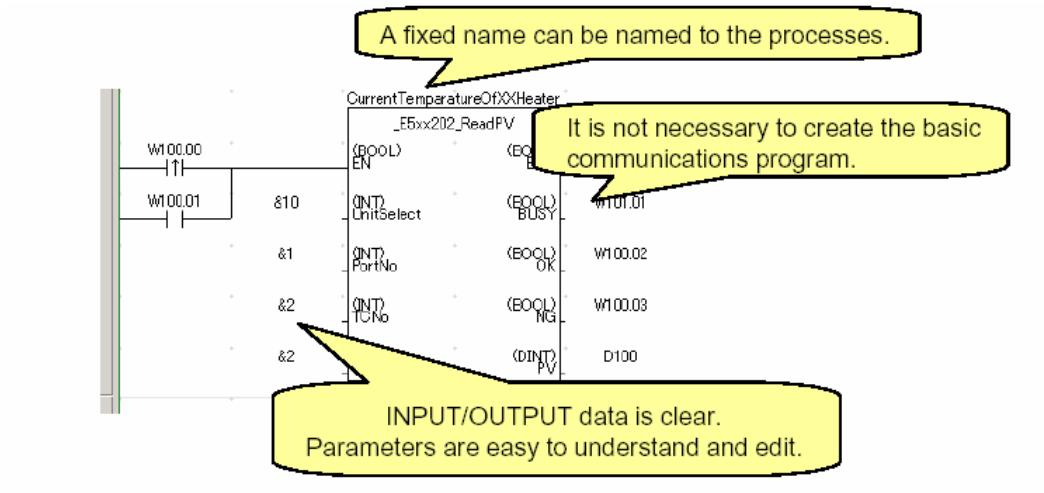
هر فانكشن بلوك يك نام واضح ومتناسب با عملكردش و نيز عنوانهاي واضح و متناسب با كاركرد براي متغيرهايش دارد .و از آنجايي كه حافظه هاي واسط(flag) و همچنين مراحل پردازش اطلاعات دروني فانكشن بلوك نمايش داده نمي شود لذا وضعيت ورودي و خروجي هاي اصلي فانكشن بلوك براي كاربر مشخص ترمي باشد .

از انجا كه مراحل داخلي برنامه مر بوط به فانكشن بلوك نمايش داده نشده و تنها علامت گرافيكي مربوط به آن و نيز وضعيت ورودي / خروجي هايش در برنامه نردباني اصلي نمايش داده مي شود، كاربر هر چه راحتتر متوجه روند كلي برنامه خواهد شد .

(5) گسترش در آينده

شركت امرن ، در آينده واسط مابين برنامه نردباني و فانكشن بلوكها را تغيير نخواهد داد.
در صورت بالا رفتن امكانات PLC يا تجهيزات صنعتي براي بهبود كارايي ، FB متناظر براي تجهيزات جديد ، جايگزين فانكشن بلوك قبلي گشته و تجهيزات به عملكردشا ن ادامه خواهند داد.

به عنوان مثال فانکشن بلوك از پيش تعريف شده براي کارکرد کنترلر دما در زير نمايش داده شده است :



همانطور که مشاهده مي نماييد اين فانکشن بلوك عنواني متناسب با کارکرد خود داشته که اين عنوان قابل تغيير توسط کاربر نمي باشد .

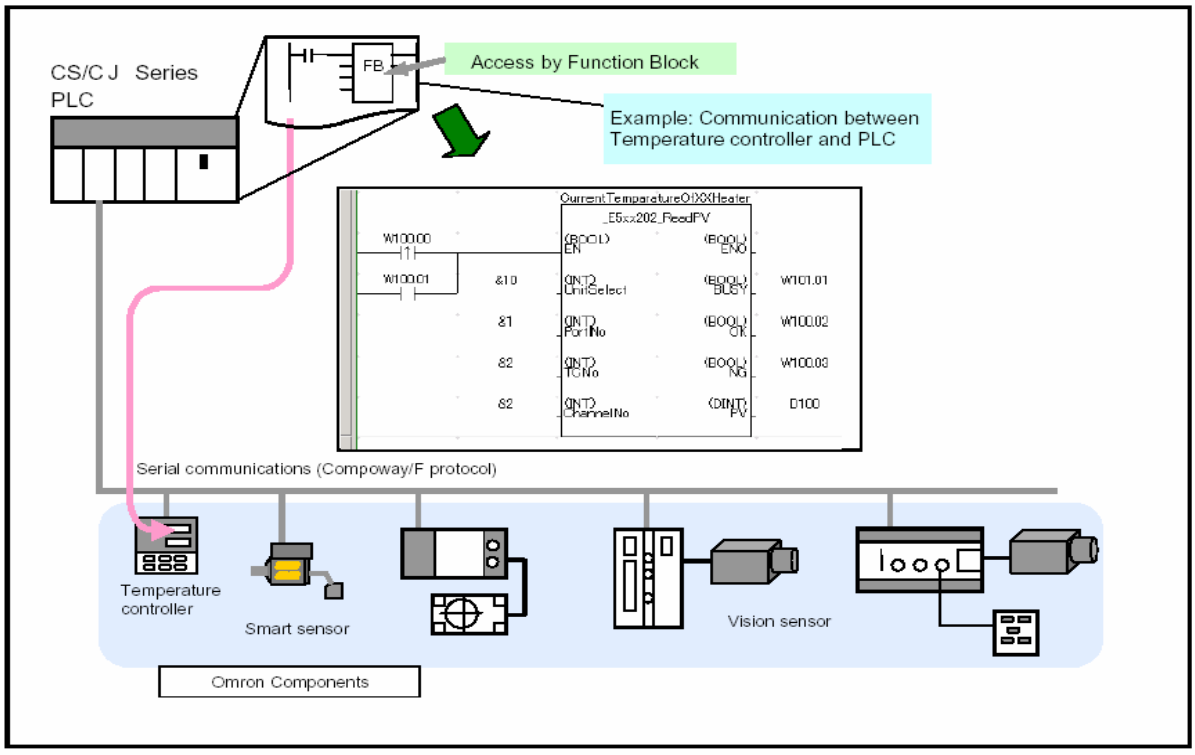
با استفاده از اين بلوك از پيش تعريف شده ، ديگر نيازي به نوشتن برنامه لازم جهت ارتباط با واحد کنترل دما وجود ندارد .

تنها کافيست ورودي / خروجي ها را مقدار دهي نماييد که اين مقادير به وضوح در کنار متغير مربوطه نمايش داده مي شوند ونيز به سادگي مي توانيد مقادير متغيرها را تغيير دهيد.

3- 2- 1 :مثال 1 - استفاده کتابخانه OMRON FB براي ارتباط وکنترل تجهيزات صنعتي :

مشاهده مي نماييد که به راحتي مي توان با استفاده از فانکشن بلوکهاي از پيش تعريف شده با تجهيزات ساخت شرکت امرن ارتباط برقرار نموده و کنترلشان نماييد .

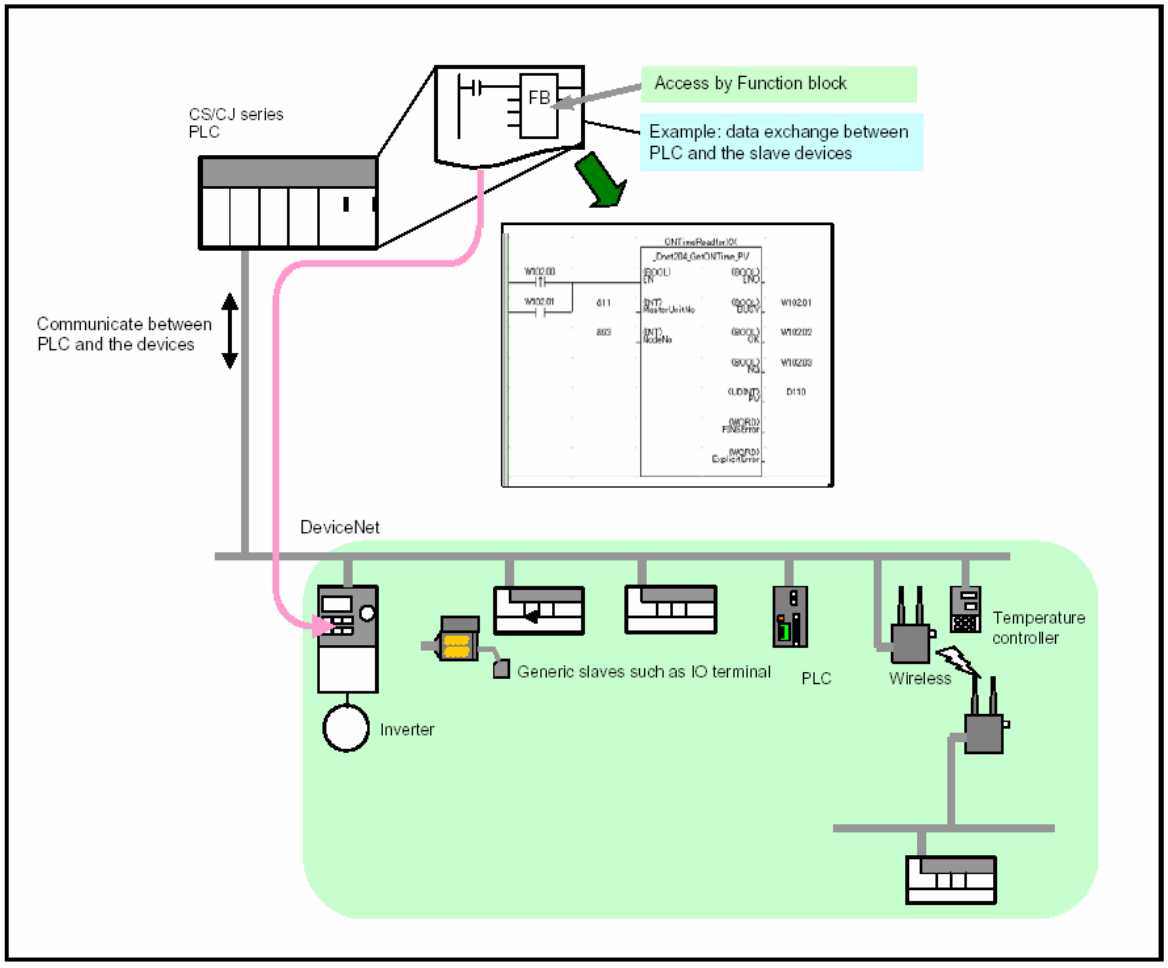
اين مثال مر بوط به اتصال ساده و ارزان قيمت سريال تجهيزات با کنترلر مي باشد



3- 2 -2 :مثال 2 - استفاده كتابخانه OMRON FB براي ارتباط وكنترل تجهيزات صنعتي :

با استفاده از شبكه ارتباطي DeviceNet ، ارتباطات كارآمد تري ايجاد مي گردد .

با استفاده از فانكشنهاي موجود در كتابخانه به آساني مي توانيد تجهيزات به كار رفته در اينگونه ارتباطات (Slaves) را نيز كنترل نماييد .



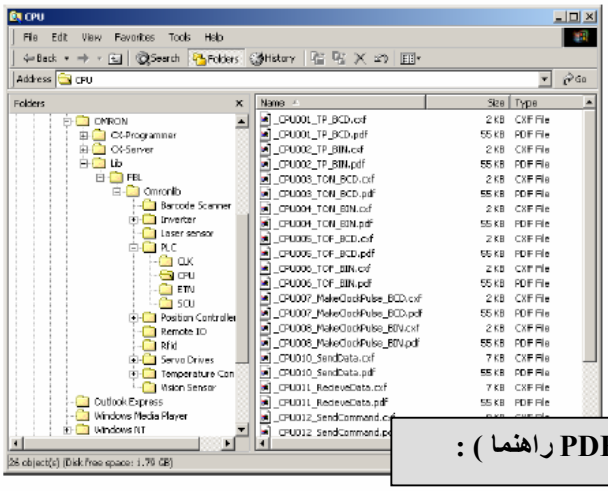
3 -3 : محتويات كتابخانه OMRON FB

محتويات كتابخانه OMRON FB عبارتند از :

3- 3- 1 : فايلهاي مربوط به فانكشن بلوكهاي از پيش تعريف شده :

اين فايلها شامل برنامه هاي از پيش تعريف شده براي ارتباط و كنترل PLC و نيز تجهيزات صنعتي ، شركت امرن بوده كه به زبان نردباني نوشته شده اند و داراي پسوند CXF. مي باشند .عنوان تمامي اين فايلها با '_' آغاز مي شود .

زماني كه كتابخانه OMRON FB را در كامپيوتر شخصيتان نصب مي نماييد اين فايلها در شاخه هايي با عنواني متناسب با نوع PLC و يا تجهيز صنعتي مربوطه طبقه بندي شده و درهمان مسيري كه نرم افزارهاي امرن را نصب نموده ايد نصب مي شوند .



3- 3- 2 : مراجع كتابخانه(فايلهاي PDF راهنما) :

همراه هر فانكشن بلوك موجود در كتابخانه، يك فايل PDF به همان نام نيز به عنوان راهنماي چگونگي كاركرد فانكشن بلوك ، مشخصات مورد نظر ، مشخصات ورودي / خروجيها و متغيرهاي هر فانكشن وجود دارد . با مطالعه فايل راهنماي مربوط به هر فانكشن بلوك مي توانيد ورودي و خروجي ها را مطابق آنچه در راهنمايش آمده مقدار دهي كرده و عملكردش را چك نماييد .يك روش آسان جهت دستيابي به اين فايلهاي راهنما آن است كه فانكشن بلوك مورد نظرتان را از كتابخانه فراخواني نماييد تا علامت گرافيكي معادل آن وارد پنجره Workspace نرم افزار CX- Programmer شود وسپس بر روي آن كليك راست نموده و از پنجره گشوده شده گزينه FB Library Reference را انتخاب نماييد .



3- 4 : فهرست فايلهاي موجود در كتابخانه (فانكشنها و فايلهاي PDF راهنما) و چگونگي دسترسي به آنها :

3- 4- 1: فهرست فايلهاي موجود در كتابخانه OMRON FB :

نوع	تجهيزات مربوطه	تعداد فانكشن بلوكهاي موجود در كتابخانه متناسب با تجهيزات
تجهيزات FA	Temperature Controlller Smart Sensor Vision sensor 2 dimensions bar code reader Wireless terminal	تقريباً 80
PLC	CPU Unit Memory Card Special CPU IO Unit (Ethernet , Controll Link ,DiviceNet, Temperature Control)	تقريباً 95
Motion Control	Posion Control Unit Inverter Servo Motor driver	تقريباً 70

3- 4- 2 : دسترسي به كتابخانه از طريق نصب CD ، CX- One /CX - Programming

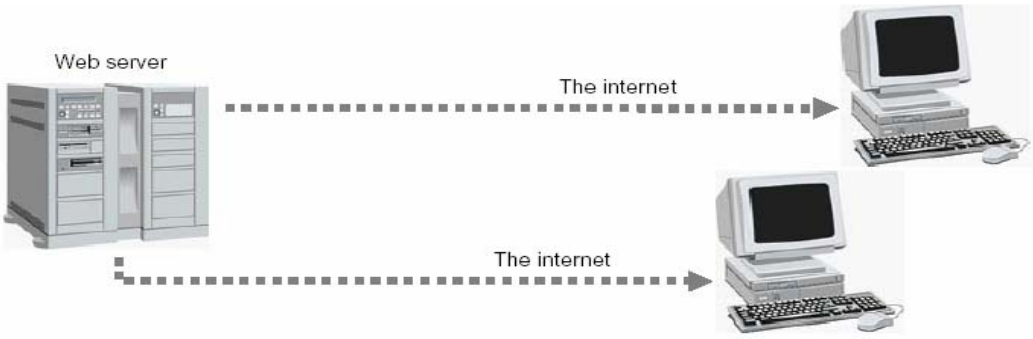
CD مربوط به نصب برنامه CX- Programmer نسخه 6 و يا CX - One شامل كتابخانه OMRON FB مي باشد .

در حين نصب اين نرم افزار مي توانيد كتابخانه فانكشن بلوكها را نيز نصب نماييد .



3- 4- 3 : دسترسي به كتابخانه از طريق وب سايت مربوط به شركت امرن :

آخرين نسخه فايلهاي كتابخانه OMRON FB كه توسط شركت امرن به منظورپشتيباني واحدهاي جديد يا بهبود يافته PLC و يا تجهيزات صنعتي ، ايجاد مي شوند ، در وب سايت اين شركت قرار داده شده و به راحتي مي توانيد آنها را از وب سايت شركت امرن Download نماييد .



فصل 2 :

چگونگي استفاده از OMRON FB :

Function Block

1. توضيح يك برنامه نمونه

در اين فصل چگونگي فراخواني يكي از فانكشنهاي موجود در كتابخانه OMRON FB تحت عنوان Make ON Time /OFF Time Clock Pulse In BCD (تايمر پالس) را توضيح خواهيم داد.
مابقي فانكشن بلوكهاي موجود در كتابخانه نيز به همين روش فراخواني مي گردند .

1- 1 : خصوصيات مورد نياز :

- **به محض آنكه كنترلر وارد حالت RUN يا monitor مي شود يك پالس توليد گردد .**
- **پالس خروجي به آدرس 1.00 منتقل شود.**
- **زمان روشن بودن پالس توسط محتويات آدرس D100 تنظيم گردد.**
- **زمان خاموش بودن پالس ثابت و برابر 2 ثانيه باشد .**

1- 2 : خصوصيات فانكشن بلوك توليد پالس موجود در كتابخانه **FB** OMRON :

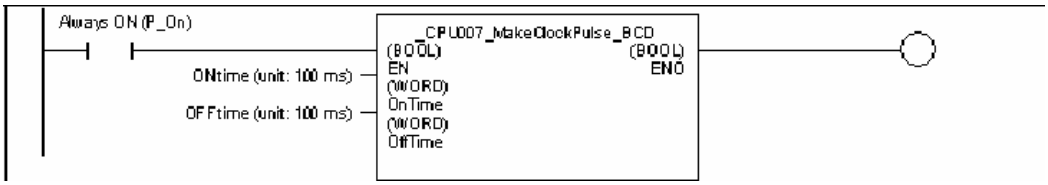
اين فانكشن بلوك يك پالس با زمان روشن وخاموش ، قابل تنظيم (در مبناي BCD) توليد مي نمايد و خصوصيات آن به صورت زير مي باشد :

نام فايل اين فانكشن بلوك :

_CPU007 _ MakeClockPulse _ BCD

كاركرد : **توليد پالسي با زمان روشن وخاموش شدن مشخص .**
پالس خروجي بر روي پايه ENO فانكشن بلوك ايجاد گشته و مي بايست اين پايه را به خروجي كه ميخواهيد پالس را به آن انتقال دهيد وصل نماييد .

سمبل فانكشن بلوك مربوطه :



مسير دستيابي به فايل :

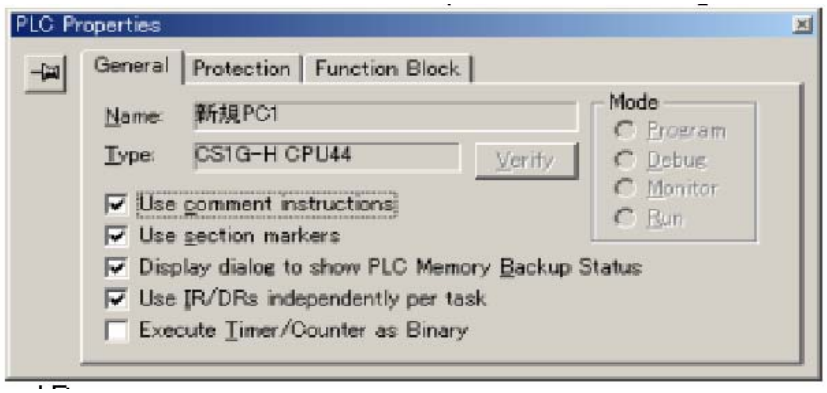
Programm file -> OMRON -> CX- One -> Lib->FBL->English-> OMRON lib -> PLC ->
- CPU007 - MakeClockPulse - BCD

مدلهاي PLC هاي كه امكان استفاده از اين فانكشن بلوك را دارا مي باشند :

CS1H , CS1H- H, CJ1

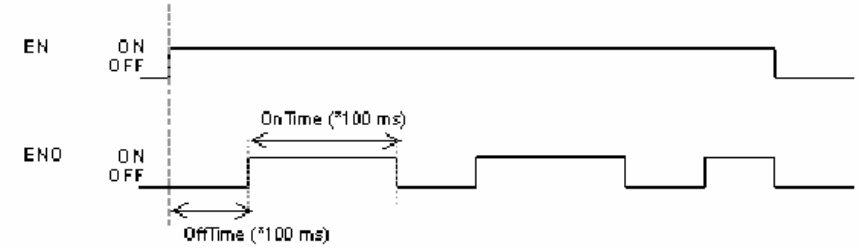
نكاتي كه مي بايست به آن توجه نماييد :

براي استفاده از اين فانكشن بلوك حتماً مي بايست در برنامه CX- Programmer در قسمت PLC Properties ، تيك مربوط به مقدار دهي تايمرها و شمارنده ها به صورت باينري را برداريد در غير اين صورت يك خطا تشخيص داده خواهد شد .



ورودي فعال سازتان (EN) را به فلگ همواره روشن متصل نماييد تا اين فانكشن بلوك همواره به محض قرار گرفتن PLC در حالت RUN يا Monitor شروع به كار نمايد .

خروجي اين فانكشن بلوك :



مشاهده مي شود بعد از اين كه PLC در حالت كاري مناسب قرار گيرد ، تا زماني كه كنتاكت فعالساز روشن است ، پالسي با زمان روشن و خاموشي كه شما تنظيم نموده ايد توليد خواهد شد .

ورودي هاي خارج از محدوده :

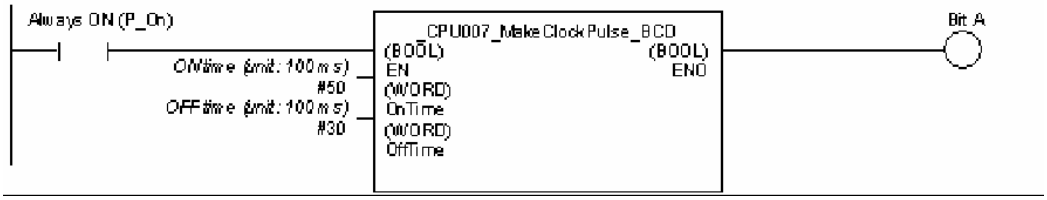
در صورتيكه ورودي ها خارج از محدوده مجازباشند ، خروجي خاموش باقي خواهد ماند و فانكشن بلوك عملكردي نخواهد داشت .

رنج مجاز ورودي هاي مربوط به تنظيم زمان روشن و خاموش بودن پالس ،عددي در رنج 0000 تا 9999(BCD) مي باشد .

البته عددي كه در اين قسمت وارد مي نماييد با مقياس 0.1 ثانيه براي تايمر در نظر گرفته خواهد شد .

يك نمونه از عملكرد :

در اين حالت بيت A متناوباً 5ثانيه روشن و 3 ثانيه خاموش خواهد بود



فانكشن بلوكهاي مرتبط :

فانكشن بلوكي مشابه با اين فانكشن بلوك تحت عنوان

CPU008- MakeClockPulse- BIN -

نيز وجود دارد كه در اين فانكشن بلوك مقدار تايمر به صورت باينري مي باشد

لذا درانتخاب ما بين اين دو فانكشن بلوك دقت نماييد .

متغيرهاي اين فانكشن بلوك :

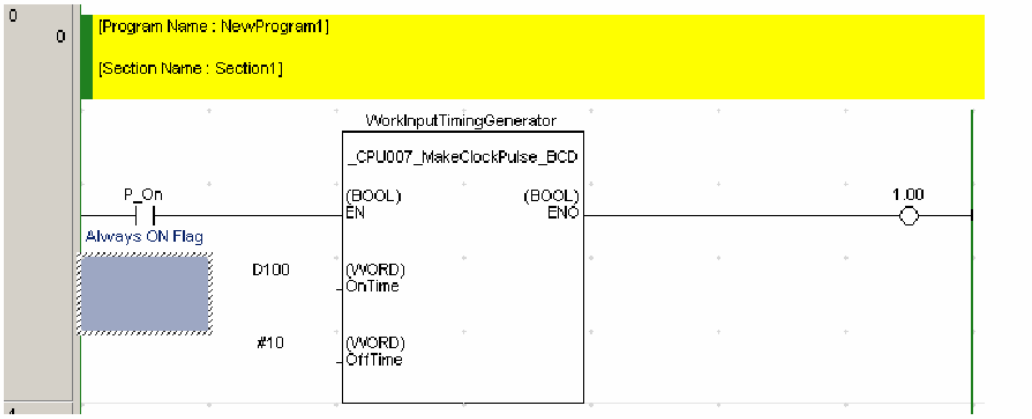
متغيرهاي ورودي

Name	Variable name	Data type	Default	Range	Description
EN	EN	BOOL			1 (ON): FB started 0 (OFF): FB not started.
ON time	OnTime	WORD		#0000 to #9999	Specify the ON time (unit: 100 ms). For example, #30 means 3 seconds.
OFF time	OffTime	WORD		#0000 to #9999	Specify the OFF time (unit: 100 ms). For example, #30 means 3 seconds.

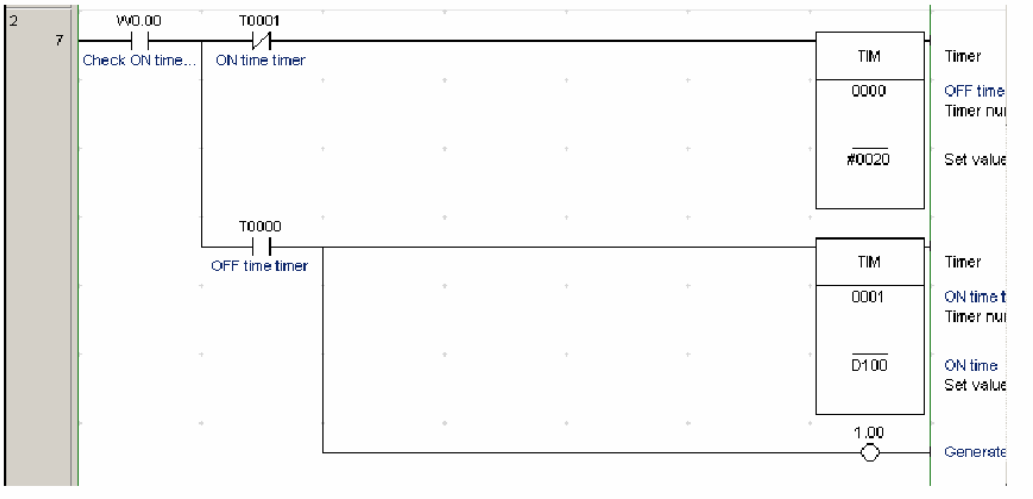
متغيرهاي خروجي

Name	Variable name	Data type	Range	Description
ENO	ENO	BOOL		Turns ON for the OnTime and OFF for the OffTime.

برنامه نردباني در صورت استفاده از فانكشن بلوك توضيح داده شده :



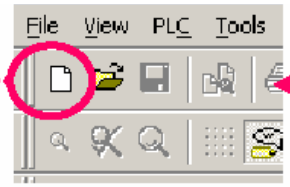
براي مقايسه : اگر اين برنامه را بدون استفاده از فانكشن بلوك بنويسيد ، به صورت زيرخواهد بود :

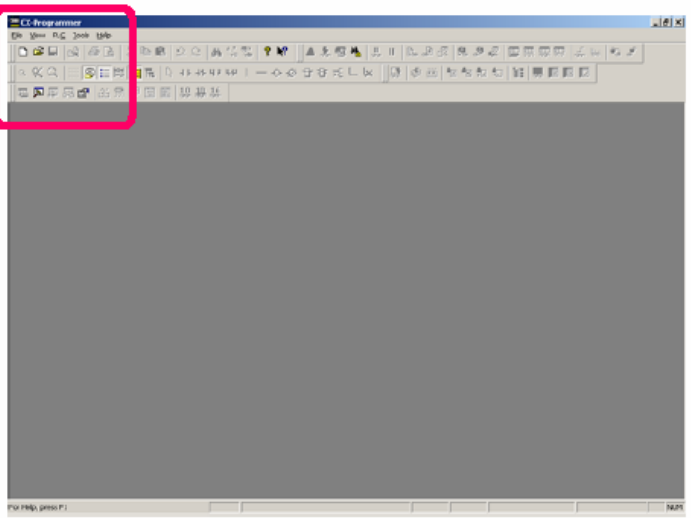


2. باز كردن يك برنامه جديد و تنظيمات مربوط به انتخاب نوع PLC

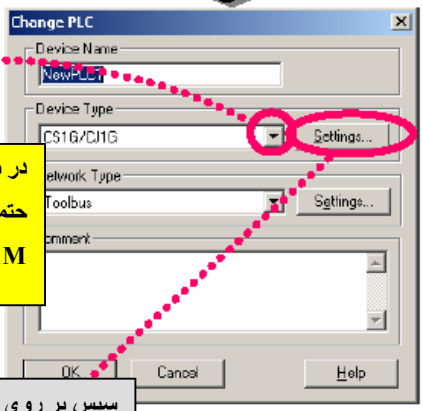
ابتدا از نوار ابزار بالايي نرم افزار ،علامت گرافيكي مربوط به باز كردن يك برنامه جديد را  انتخاب نماييد تا پنجره تنظيمات گشوده شود :







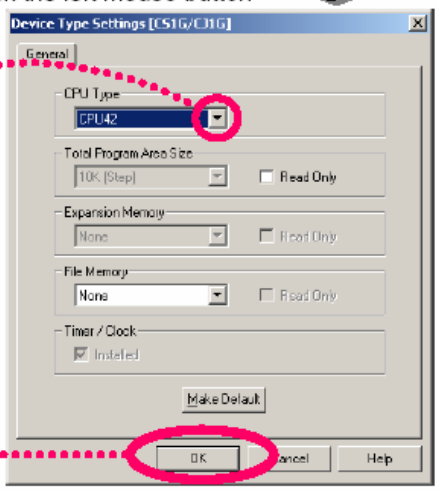
در پنجره باز شده در قسمت
Device Type
نوع PLC مرتبط را انتخاب نماييد



در صورتيكه مي خواهيد در برنامه تان از فانكشن بلوكها استفاده كنيد ، حتماً مي بايست يكي از انواع PLC هاي زير را انتخاب نماييد :
CS1G- H , CS1H- H ,CJ1H- H ,CJ1M

سپس بر روي گزينه Setting كليك نماييد

Click the left mouse button



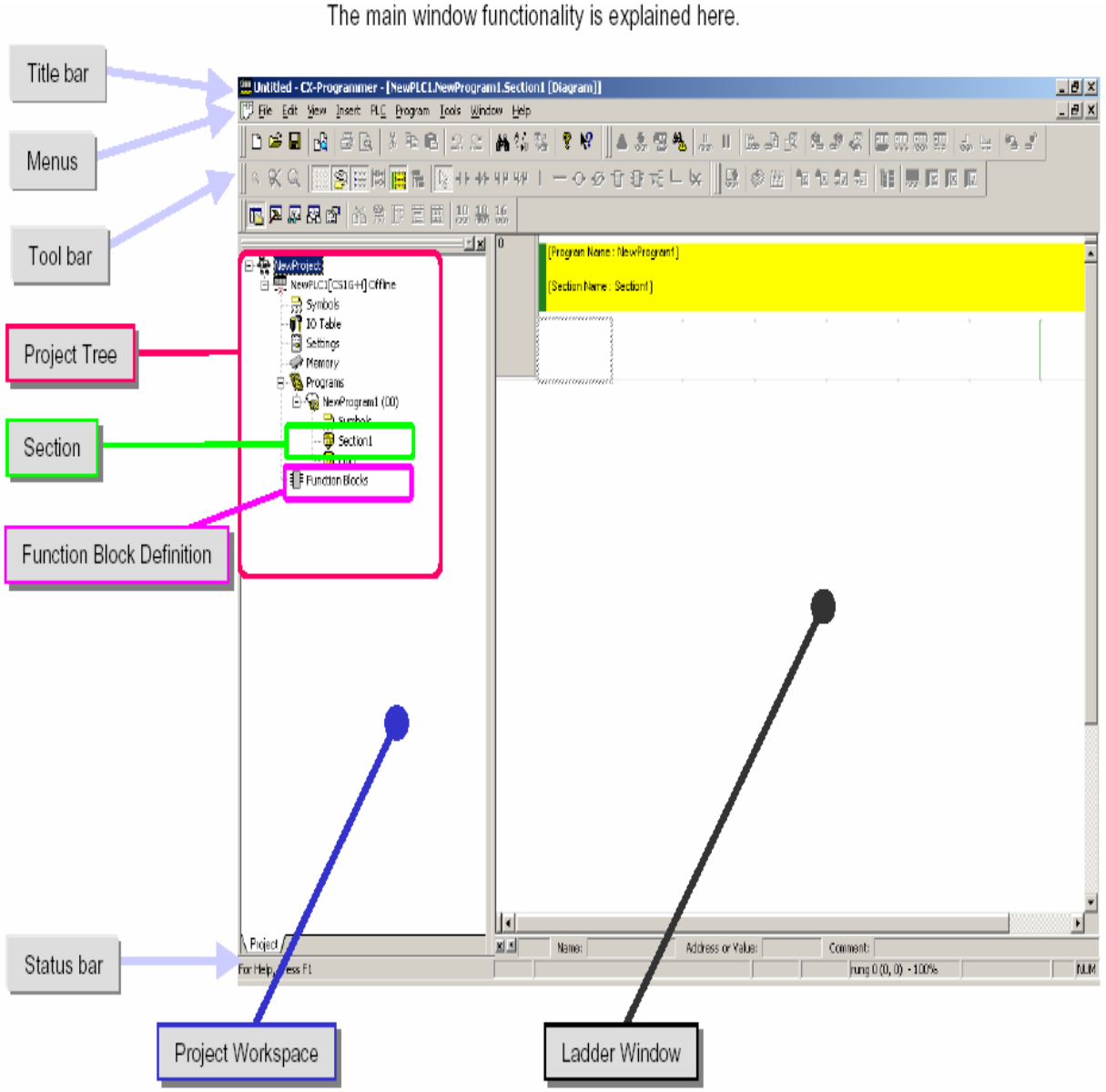
در پنجره گشوده شده با كليك چپ بر روي فلش قسمت مربوط به انتخاب نوع CPU، مدل CPU كنترلر تان را انتخاب نماييد. توجه داشته باشيد كه CPU انتخابي در اين قسمت دقيقاً با CPU كنترلري كه مي خواهيد به ان متصل شويد يكسان باشد .



بعد از انجام تنظيمات OK نماييد تا اين پنجره بسته شود .

قسمتهاي مختلف محيط برنامه نويسي اصلي :

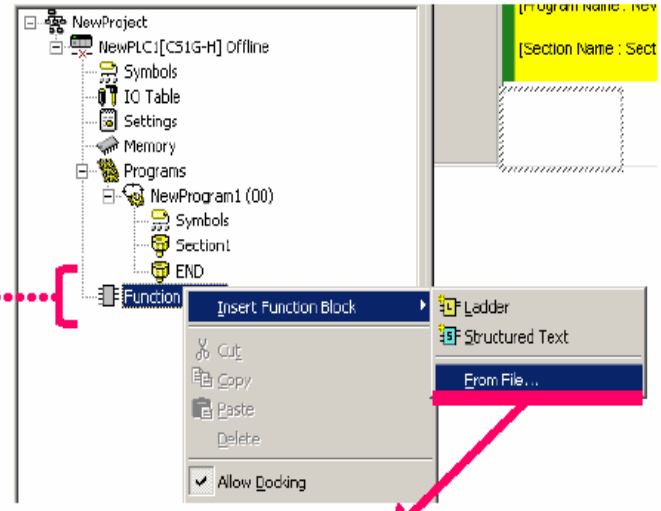
بعد از Ok نمودن پنجره تنظيمات ، وارد محيط برنامه نويسي نرم افزار CX - Programmer خواهيد شد كه شامل قسمت هاي زير است :



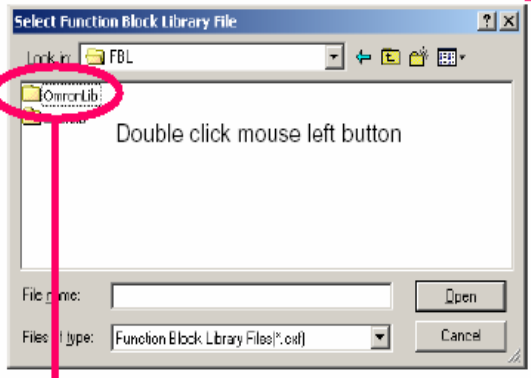
عنوان	محتويات / عملكرد
Title Bar	نام فايل ذخيره سازي برنامه CX - Programmer ايجاد شده در اين قسمت نمايش داده مي شود
Menus	امكان انتخاب منوهاي مختلف را در اختيار شما قرار مي دهد
Toolbars	امكان انتخاب فانكشنها را با كليك بر روي ايكونهاي معادل در اختيار شما قرار مي دهد . براي نمايش اين نوار ابزار مي بايست در منو View گزينه Toolbars را انتخاب نماييد .
Section	امكان تقسيم برنامه به چندين بخش را ايجاد مي نمايد كه هر كدام از اين بخشها مي توانند جداگانه ايجاد و يا نمايش داده شوند
Project WorkSpace Project Tree	اين قسمت مربوط به كنترل برنامه ها و اطلاعات مي باشد . اين امكان وجود دارد كه در صورت نياز Section ها و يا Symbol ها ي يك برنامه را با كشيدن موس و قرار دادن در قسمت مشابه در پروژه اي ديگر، به ان پروژه منتقل نماييد .
Ladder Window	صفحه ايست كه مي توان در آن برنامه نردباني را ايجاد و يا اصلاح نمود
Function Block Definition	اين قسمت محل تعريف فانكشن بلوكهاي جديد ونمايش فانكشن بلوكهايي كه قبلاً در برنامه تعريف شده اند مي باشد . با انتخاب علامت گرافيكي معادل هرفانكشن بلوك مي توانيد ان فانكشن بلوك را كپي و يا حذف نماييد . علامتهاي گرافيكي معادل فانكشن بلوكها : : در صورتي كه فانكشن بلوك استفاده شده جزء فانكشن بلوكهاي از پيش تعريف شده و موجود در كتابخانه نرم افزار باشد با اين علامت گرافيكي نمايش داده خواهد شد . : درصورتي كه فانكشن بلوك مورد نظر را به زبان نردباني نوشته باشيد با اين علامت گرافيكي نمايش داده خواهد شد . : درصورتي كه فانكشن بلوك مورد نظر را به زبان ST نوشته باشيد با اين علامت گرافيكي نمايش داده خواهد شد .
Status Bar	در اين قسمت اطلاعاتي همچون نام PLC وضعيت Online و يا Offline بودن آن و نيز موقعيت سلول فعال در شبكه نمايش داده مي شوند .

3. وارد كردن فانكشن بلوكهاي OMRON در برنامه

در پنجره WorkSpace
بر روي علامت گرافيكي
Function Block كليك راست
نموده و از منو هاي گشوده شده
گزينه هاي زير را انتخاب نماييد :
Insert Function Block
Library File

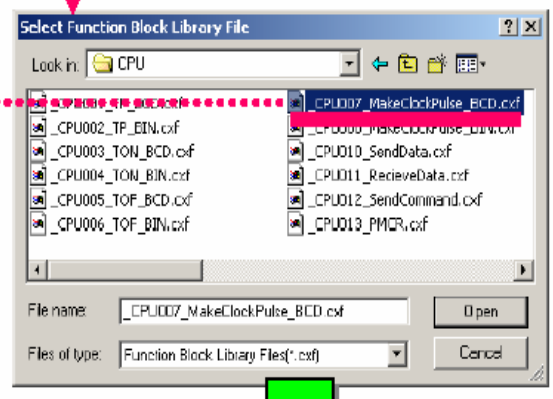


بر روي شاخه ذخيره سازي فانكشن بلوكها
دوبار كليك كرده وآن را باز نماييد.
فايل مورد نظر تان را انتخاب نماييد.
در مثال مورد نظر مسير زير را طي مي كنيم .
Omronlib
Programmable Controller
CPU



مسير پيش فرض شاخه ذخيره سازي كتابخانه فانكشن بلوكهاي امرن
به صورت زير ميباشد :
C:\ Program Files \ Omron \ CX- One \ Lib \ FBL

پس از انتخاب فانكشن بلوك
_CPU007- MakeClock Pulse-BCD.cxf
بر روي آن دوبار كليك كرده و ان را بگشاييد

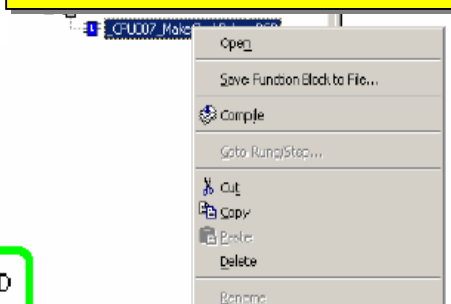


حال فايل ذخيره سازي برنامة تمامي فانكشن بلوكهاي از پيش تعريف شدة موجود دركتابخانة نرم افزار ، فهرست وار در اختيار شما قرار مي گيرد و مي توانيد فانكشن بلوك مورد نظر تان را انتخاب نماييد.
در اين مثال برنامه مربوط به تايمرپالس با مقدار گيري به فرم BCD را انتخاب مي نماييم .اين فانكشن بلوك تحت عنوان CPU007- Make Clock Pulse- BCD.cxf - ذخيره شده است.

يك روش آسان براي دستيابي به فايل pdf راهنماي فانكشن بلوك مورد نظرتان اين است كه، پس از اضافه شدن علامت گرافيكي معادل ان فانكشن بلوك ، به صورت زير شاخه در پنجره project work space بر روي ان كليك راست نموده و از منو گشوده شد گزينه FB library reference را انتخاب نماييد .

در اين حال در برنامه نردباني ، در پنجره Project work Space و در شاخه Function Blocks علامت گرافيكي معادل اين فانكشن بلوك به صورت يك زير شاخه اضافه خواهد شد .



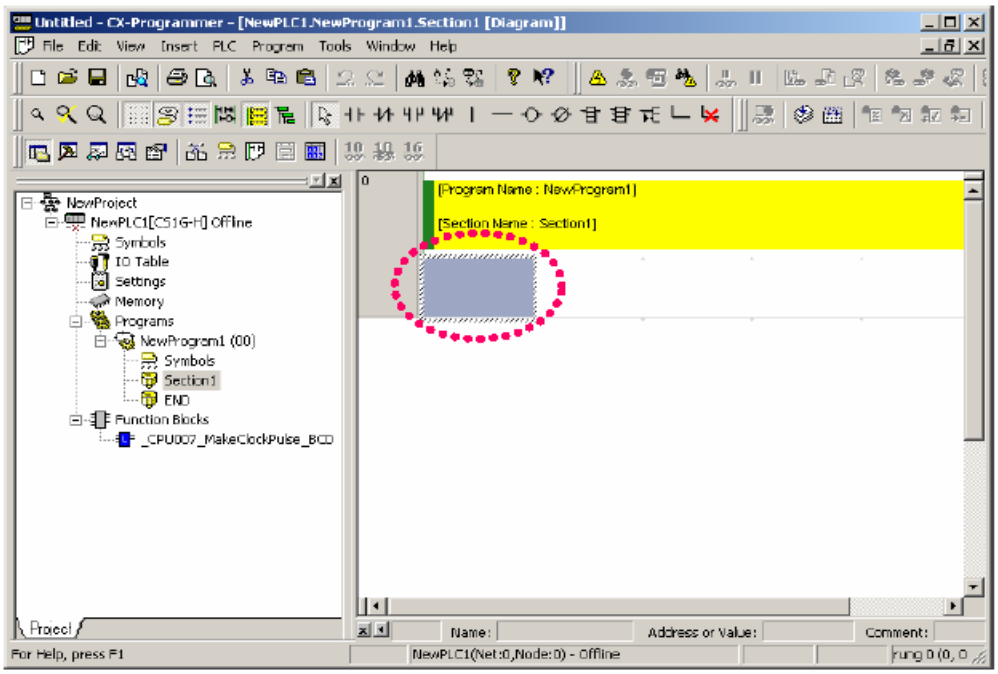


علامت گرافيكي تعريف فانكشن بلوك



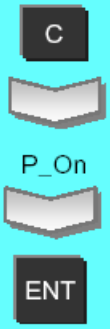
5. ایجاد برنامه

مکان نما را در بالاترین گوشه سمت چپ پنجره Ladder Window قرار دهید .



5- 1 : تعریف یك كنتاكت باز

با فشردن كليد C ، يك كنتاكت باز ايجاد نما ييد .
در اين حال پنجره زير گشوده خواهد شد . با استفاده از فلش كناري محل تعريف كنتاكت جديد ، مي توانيد يكي از كنتاكتهاي از پيش تعريف شده را انتخاب نماييد كه در اين مثال يك پالس همواره روشن را با عنوان P- On انتخاب مينماييم.





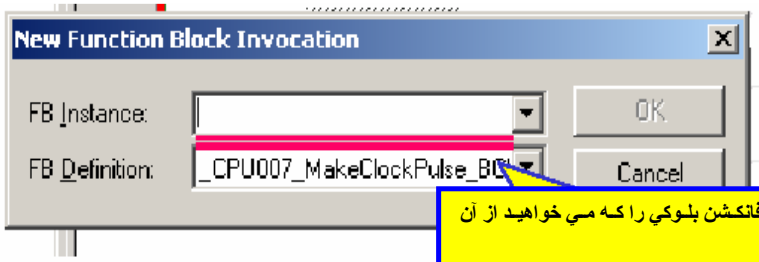


به منظور حذف كنتاكت كافيست ، مكان نما را بر روي ان قرار داده و كليد DEL را بفشاريد و يا آنكه مكان نما را در سمت راست آن قرار داده و كليد BS را بفشاريد

اين كنتاكت از سمبلهاي از پيش تعريف شده بوده و همواره در وضعيت ON مي باشد .

2_5. فراخواني فانكشن بلوك در برنامه اصلي

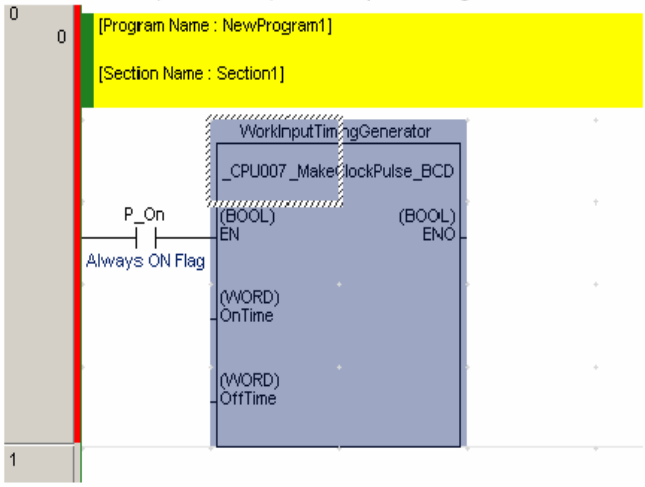
كليد F را فشار دهيد تا پنجره مربوط به فراخواني فانكشن بلوك ، گشوده شود .



در قسمت FB Definition فانكشن بلوكي را كه مي خواهيد از آن استفاده كنيد ،انتخاب نماييد.
توجه داشته باشيد كه حتماً مي بايست فانكشن بلوك مورد نظر در پنجره project workspace تعريف شده باشد .

درقسمت FB Instance نامي متناسب با كاركرد فانكشن بلوك مورد نظرتان را وارد نماييد .در مثال مورد بحث عنوان WorkInput Timing Generator را وارد نموده ايم .

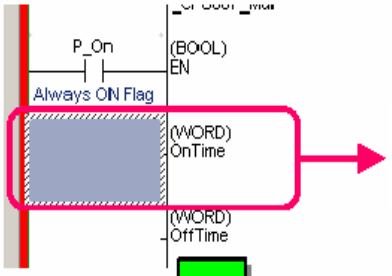
Shows FB instance (invocation) 'WorkInputTimingGenerator'.



بعد از ورود عنوان دلخواه كليد ENTER را بزنيد تا فانكشن بلوك مورد نظرتان وارد محيط برنامه نويسي اصلي شود

2_5. وارد نمودن پارا مترها :

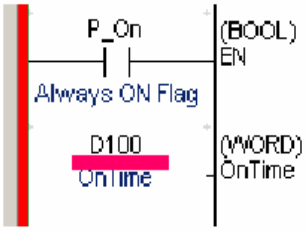
حال مي بايست به پارامتهاي ورودي و خروجي آدرس اختصاص دهيد و بدين منظور مراحل زير را انجام دهيد .
مكان نما را در سمت چپ متغير قرار داده و سپس كليد
Enter و يا P
را بفشاريد تا پنجره مربوط به آدرس دهي متغير ها گشوده شود .



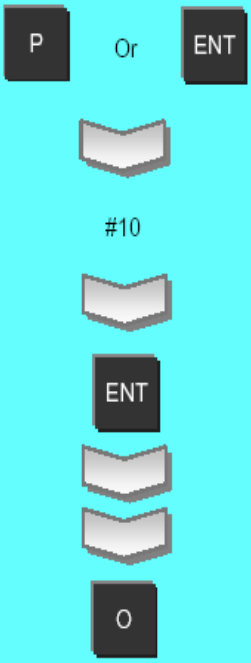
مكان نما را در سمت چپ متغير قرار دهد



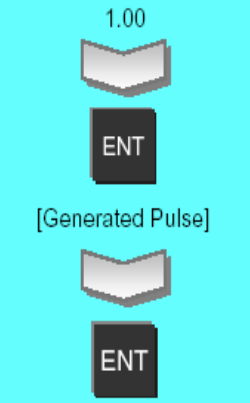
به متغير مورد نظرتان يك آدرس اختصاص دهيد به عنوان مثال به متغير
On Time، آدرس D100
را اختصاص داده
وسپس كليد ENT را بفشاريد

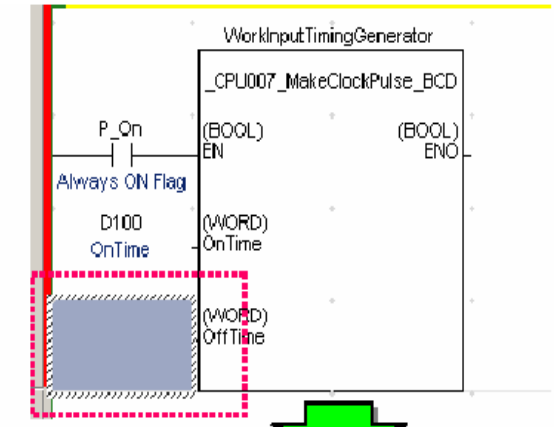


مابقي متغيرها را نيز به همين روش مقدار و يا آدرس دهي نماييد .



براي آدرس دهي كنتاكتهاي خروجي كليد O را بفشاريد تا پنجره زير گشوده شود .



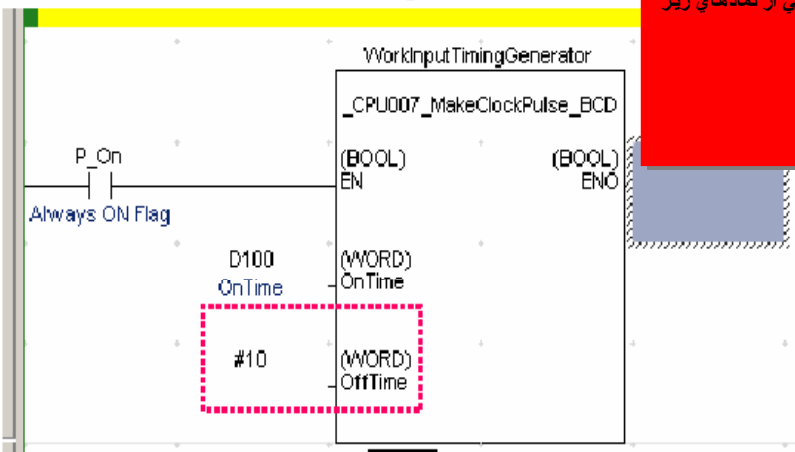




توجه داشته باشيد كه براي ورود مقادير عددي متناسب با فرمت انها، قبل از عدد مورد نظر يكي از نمادهاي زير را قرار دهيد :

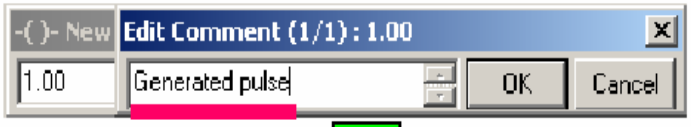
براي مقادير HEX و يا BCD

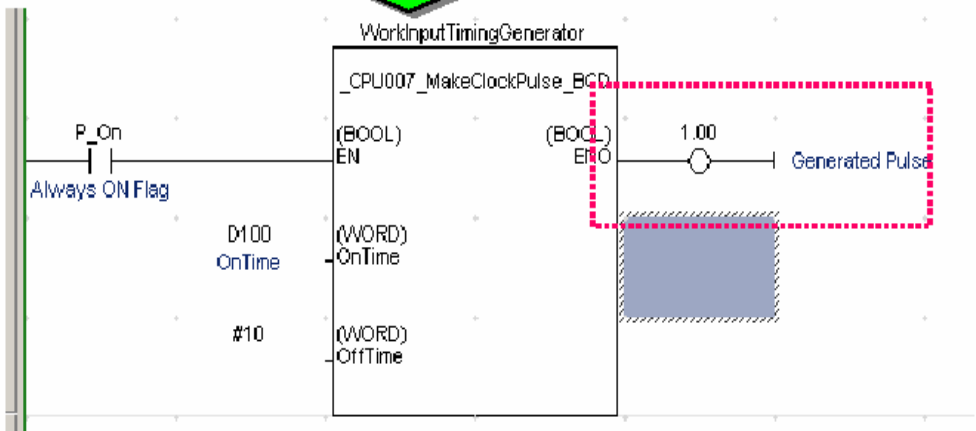
& براي مقادير DEC



در زير مراحل آدرس و يا مقدار دهي متغيرهاي فانكشن بلوك مربوط به تايمر پالس امده است .



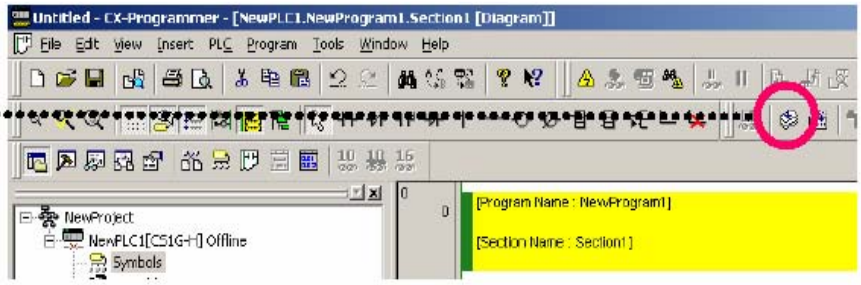


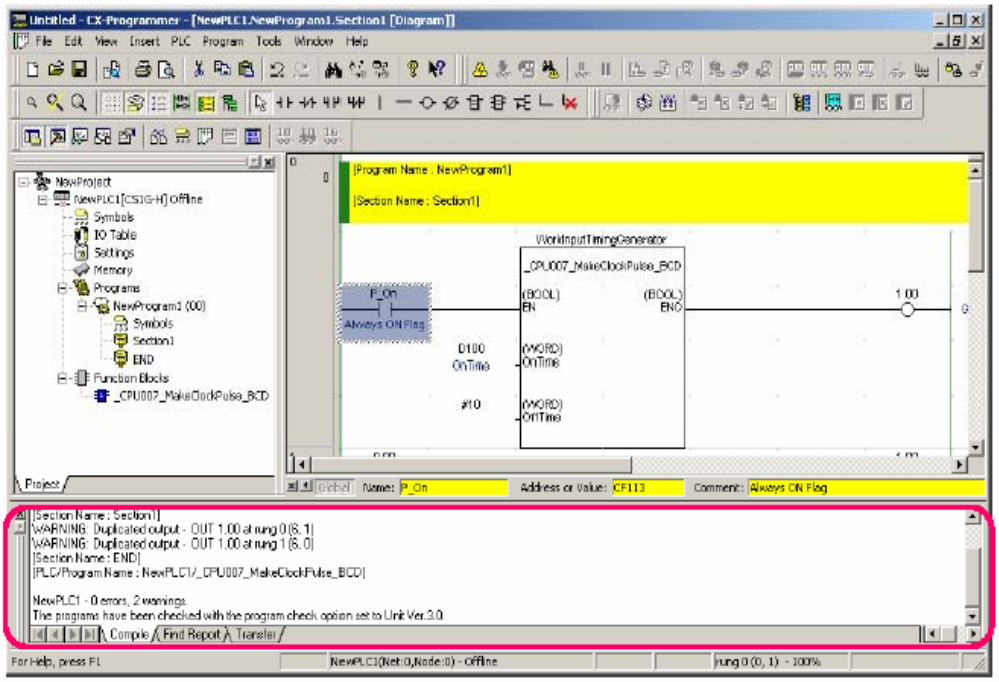


6. خطايابي برنا مه :

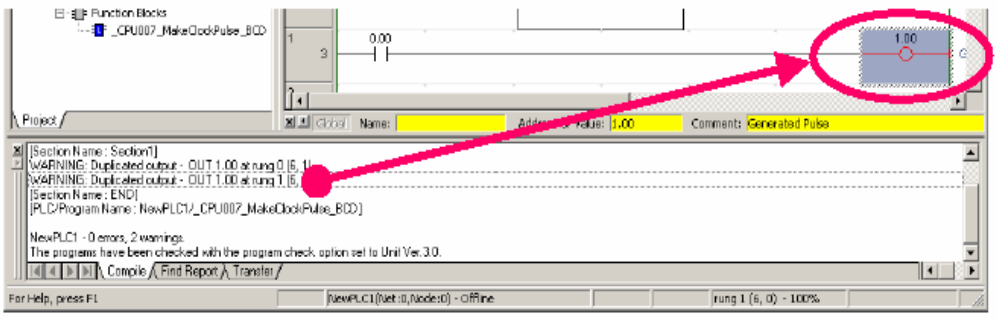
قبل از انتقال ، برنامه را با انتخاب علامت گرافيكي خطايابي  از نوار ابزار بالايي نرم افزار ، خطا يابي نماييد .

Click 





در هنگـام خطـا يـابي در قسمت زيـرين محيط برنامـه نويسي پنجـره اي گشوده خواهـد شـد و در صـورت وجـود خطـا ، خطـا هـا و نيـز آدرسـهاي معـادل آنهـا در ايـن قـسمت فهرسـت وار نمـايش داده خواهند شد .



با دوبار كليك بر روي هر يك از خطا هاي موجود در اين فهرست به محل خطا در برنامه منتقل خواهيد شد . قسمتي از برنامه كه خطا دارد به رنگ قرمز نمايش داده خواهد شد .

اصلاح خطا

بافشردن كليد J و يا F4 مي توانيد به محل خطاهاي موجود در برنامه منتقل شويد .
به منظور بستن پنجرة خطايابي برنامه كه به صورت خودكار در هنگام خطايابي گشوده مي شود كافيست كليد Esc را بفشاريد .

7 . Online شدن :

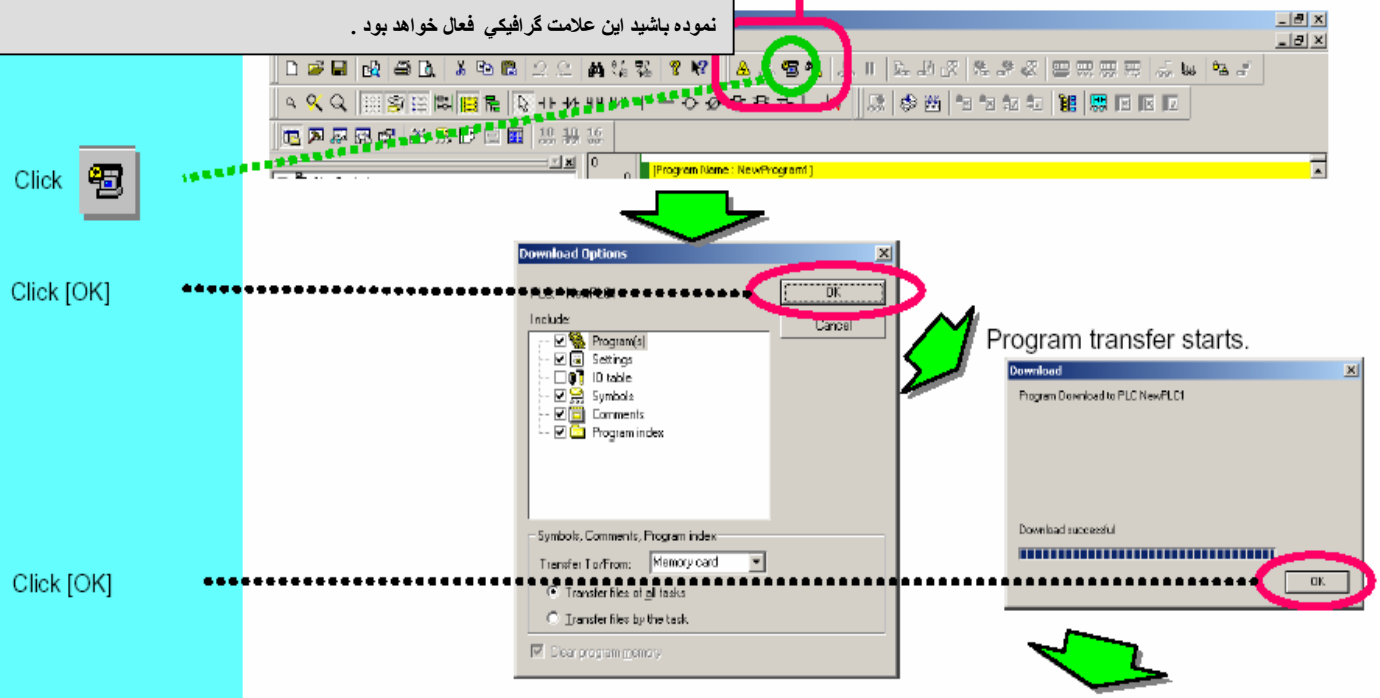
در برنامه CX- Programmer سه روش براي Online شدن وجود دارد :

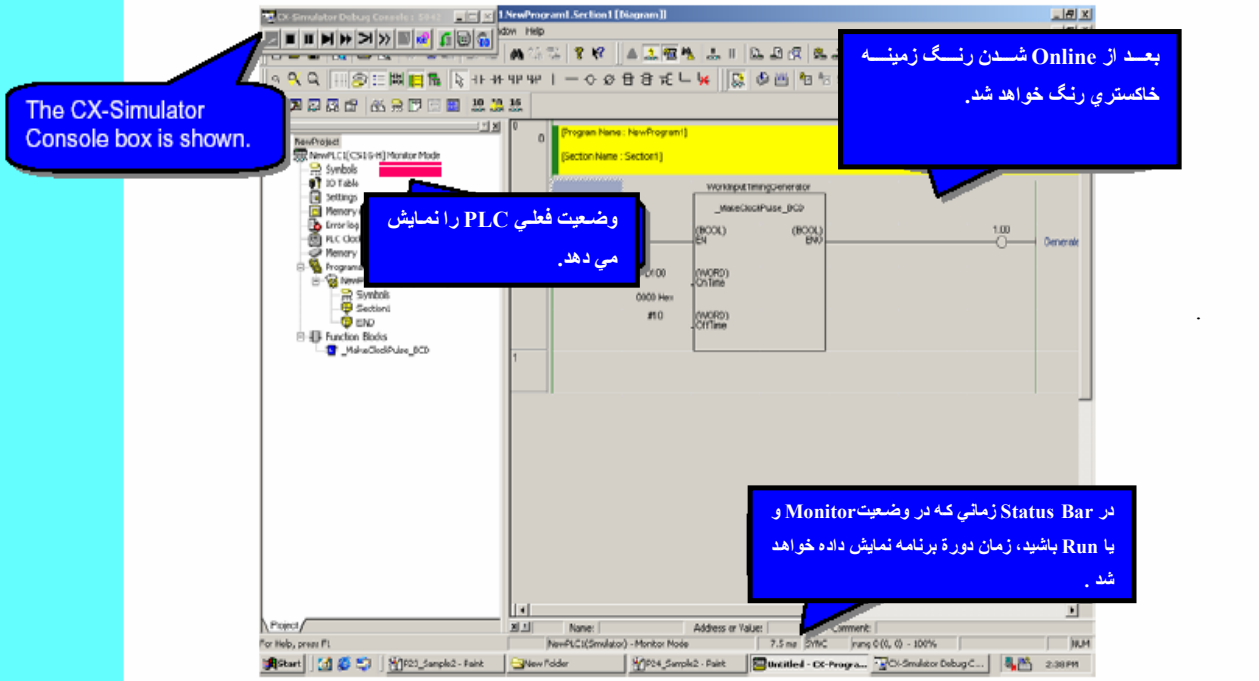
در صورتي كه PLC اي كه مي خواهيد به آن متصل شويد دقيقاً مشابه PLC اي باشد كه در هنگام تعريف برنامه انتخاب نموده ايد با كليك بر روي اين علامت گرافيكي به راحتي به كنترلرتان متصل خواهيد شد .

اتصال اتوماتيك : با كليك بر روي اين علامت گرافيكي به طور اتوماتيك PLC متصل، شناسايي شده و متصل خواهيد شد .

ارتباط مجازي از طريق سيمولاتور :
در اين حالت با كليك بر روي اين علامت گرافيكي مجازاً به كنترلر مجازي متصل خواهيد شد و نيازي به وجود PLC براي اتصال نيست .

توجه داشته باشيد كه تنها در صورتيكه برنامه CX - Simulator را به صورت مجزا نصب نموده باشيد اين علامت گرافيكي فعال خواهد بود .



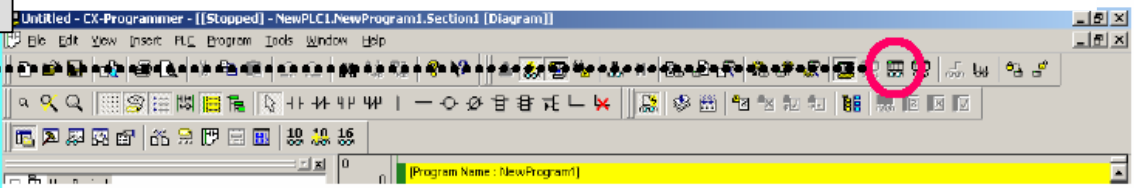


8. مانيتور نمودن برنامه :

در اين حالت مي توانيد وضعيت روشن و يا خاموش بودن كنتاكتها وكويلها ونيز مقدار هر يك از متغير ها را در زمان اجراي برنامه را مشاهده نماييد .

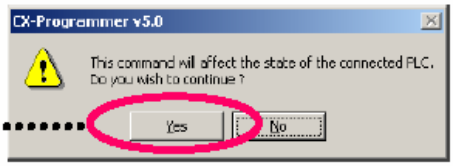
براي تغيير وضعيت كاري PLC به حالت مانيتور بر روي علامت گرافيكي زيركليك نماييد .

Click





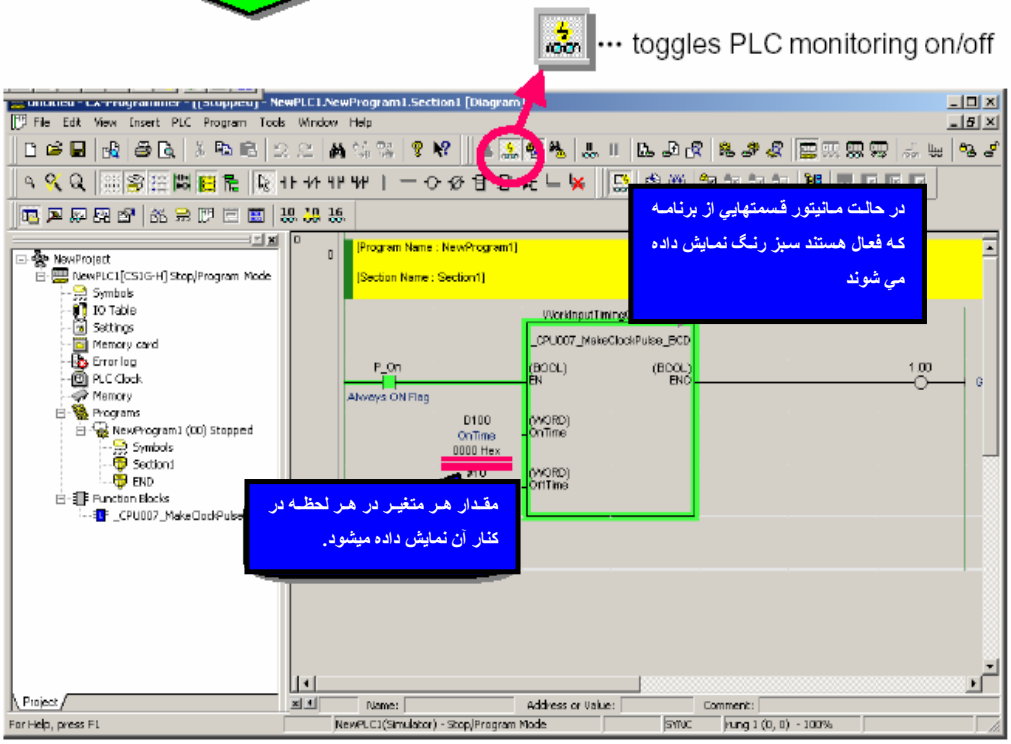




Click [Yes].

در صورتيكه برنامه شما حجم زيادي از اطلاعات را در بر داشته باشد، ممكن است جابجايي در طول برنامه به كندي صورت پذيرد براي رفع اين مشكل با كليك بر روي علامت گرافيكي  از حالت مانيتور خارج شده ، به محل مورد نظر از برنامه كه مي خواهيد منتقل شويد ، رفته و سپس مجدداً وارد حالت مانيتور شويد





9. تغيير مقاديرفعلي متغيرها در حالت مانيتور:

در اين قسمت به چگونگي تغيير و ضعيت كنتاكتها / كويلها و مقادير موجود در حافظه هاي به كار رفته در برنامه خواهيم پرداخت :

به طور مثال مي خواهيم مدت زمان **ON** بودن پالس ايجاد شده توسط تايمر پالسي كه از طريق فانكشن بلوك در برنامه وارد نموده ايم را تغيير دهيم:

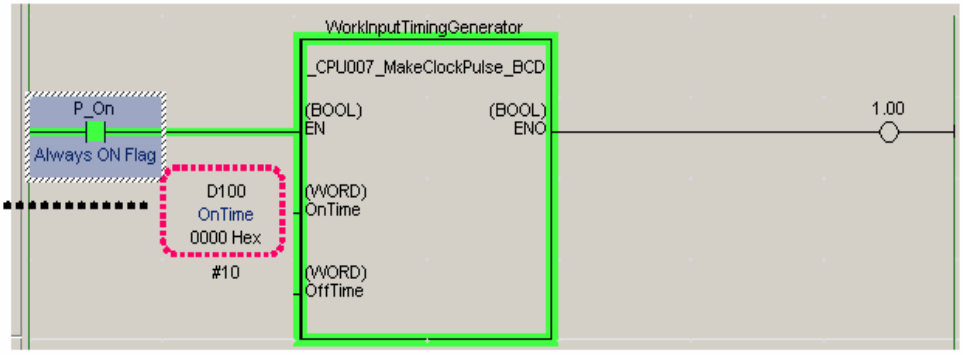
به منظور باز كردن پنجره مربوط به مقدار دهي يك متغير سه روش وجود دارد

1.بر روي متغير مورد نظر (ON Time) كليك راست نموده و از منو گشوده شده گزينه Value -> set را انتخاب نماييد

2.بر روي متغير مورد نظردوبار كليك نماييد

3.بعد از قرار گرفتن مكان نما بر روي متغير مورد نظر كليد Enter را بفشاريد .

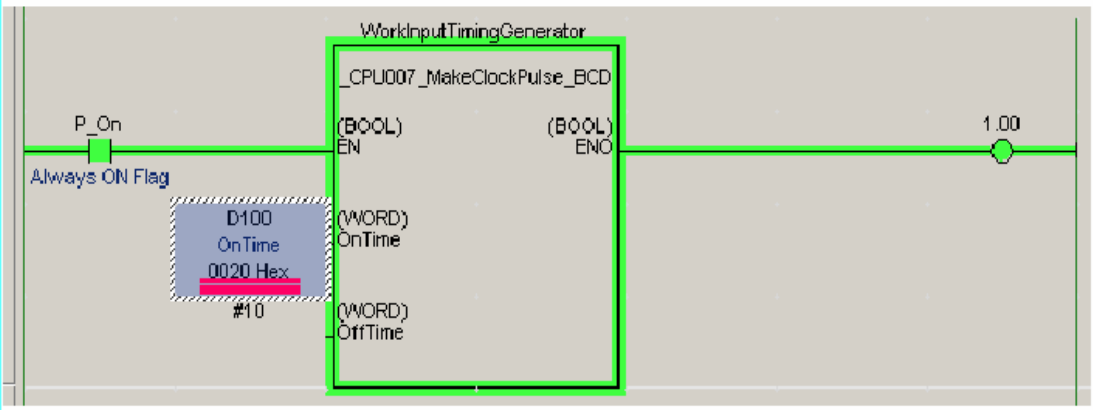
حال در قسمت Value مقدار مورد نظرتان را وارد نماييد .





توجه داشته باشيد اگر مقدار موردنظر شما در مبناي HEX و يا BCD مي باشد قبل از عدد مورد نظرتان علامت # ،

و اگر در مبناي DEC مي باشد قبل از ان حتماً نماد & را قرار دهيد .



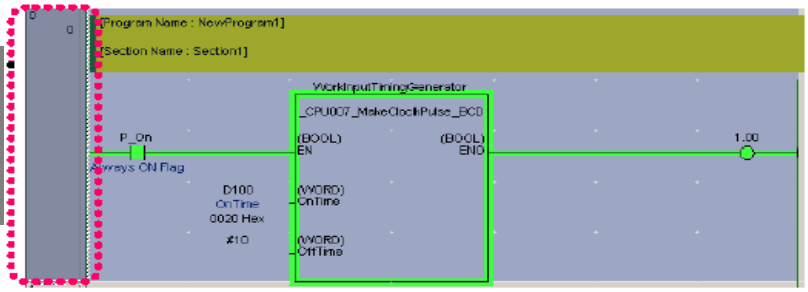
بعد از مقدار دهي متغير موردنظر با رعايت نكته فوق الذكر بر روي گزينه Set كليك نماييد تا پنجره بسته شده ومتغير مورد نظر مقدار دهي شود .

بعد از انجام مراحل فوق متغير مورد نظر مقدار جديد را اتخاذ نموده و در صورتيكه در حالت مانيتور باشيد اين مقدار در كنار آن نمايش داده خواهد شد.

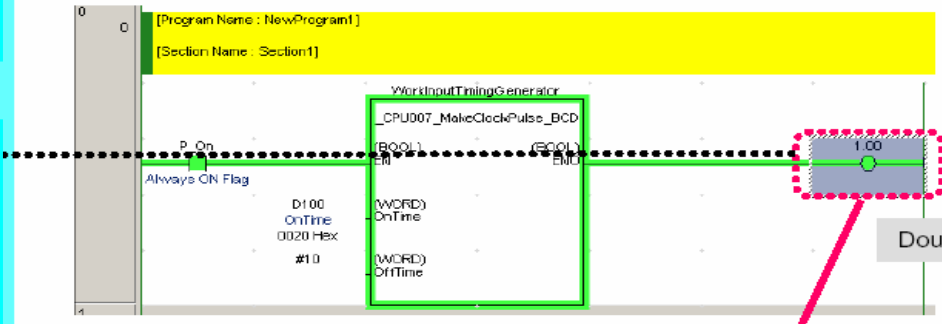
10. تصحيح برنامه در حالت Online

ابتـدا مكـان نمـا را بـر روي خطـي كـه مـي خواهيد تغييراتي در برنامـه آن ايجـاد نماييد قرار دهيد .

همچنين اين امكان وجود دارد كه در صورت نياز به اصلاح چند خط از برنامه ، با كليـك كـردن بـرروي يك خط و كشيدن موس چندين خط را بـه صـورت همزمان انتخاب نماييد



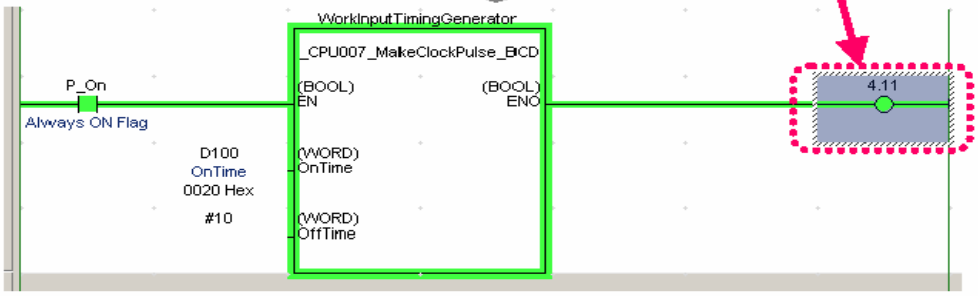
حال از نوار ابـزار بـالايي نـر م افـزار و از منـو **Program** گزينـه **Online Edit** و سـپس **Begin** را انتخاب نموده و بـا آنكـه كليـد هـاي **Ctrl** و **E** را به صورت همزمان بفشاريد .



در اين حال رنگ زمينه خطوطي كه براي تصحيح انتخاب نموده ايد تغيير كرده و سفيد رنگ خواهد شد .حال بر روي هر المانی كه مي خواهيد تغيير ي در آن ايجاد نماييد دوبار كليك نمـوده و تغييرات موردنظر را اعمال نماييد .

Edit the address to the required bit number (4.11 in the example)





بعد از اعمال تمامي اصلاحات مورد نياز، براي بازگر دانـدن خطوط اصلاح شده به كل برنامه اينچنين عمل نماييد :

از نوار ابزار بالايي نر م افزار
و از منو **Program**
گزينه **Online Edit**
و سپس **SendChange**
را انتخاب نماييد
و يا آنكه كليد هاي
Ctrl و **Shift** و **E**
را به صورت همزمان بفشاريد .

پايان

فصل 3

ايجاد تغييرات در فانكشن بلوكهاي موجود در كتابخانه نرم افزار **CX- PROGRAMMER**

Function Block

1. توضيح برنامه مورد نظر

در اين فصل چگونگي ايجاد تغييرات در فانكشن بلوك توليد تايمر پالس موجود در كتابخانه نرم افزار را توضيح خواهيم داد .

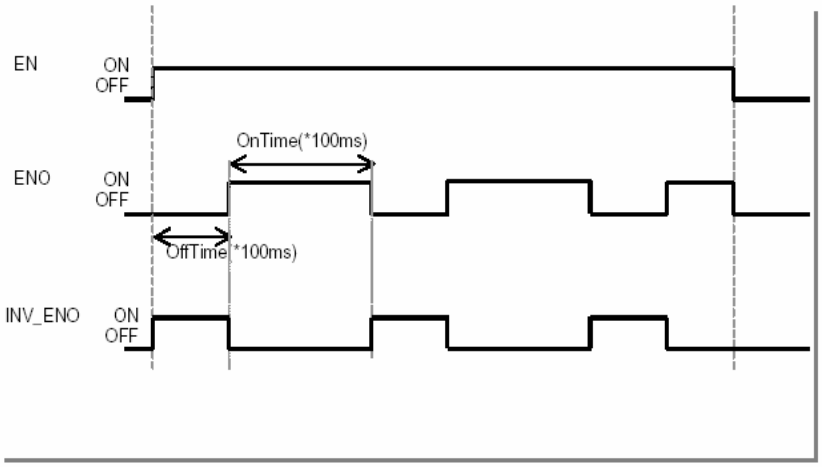
1_1: تغيير ات مورد نظر:

فانكشن بلوك توليد پالس **" Make ON Time / OFF Time Clock Pulse in BCD "** خروجي **ENO** را متناوباً به مدت زمان مشخص **Off Time** خاموش و سپس به مدت زمان مشخص **ON Time** روشن مي نمايد .

حال بر آن هستيم معكوس خروجي فوق را با اضافه نمودن متغير خروجي **"INV - ENO"** ايجاد نماييم .

چگونگي كاركرد اين فانكشن بلوك و نيز خروجي هاي **ENO** و **INV - ENO** در زير آمده است .

INV_ENO :



1_2: تغيير اتي كه مي بايست در محتويات برنامه فانكشن بلوك موجود در كتابخانه ايجاد شود :

براي بر آوردن تغيير فوق الذكر (دستيابي به خروجي معكوس ENO)، تغييرات زير مي بايست در فايل مربوط به ذخيره سازي برنامه فانكشن بلوك مورد نظر (Make ON Time / OFF Time Clock Pulse in BCD) ايجاد شود :

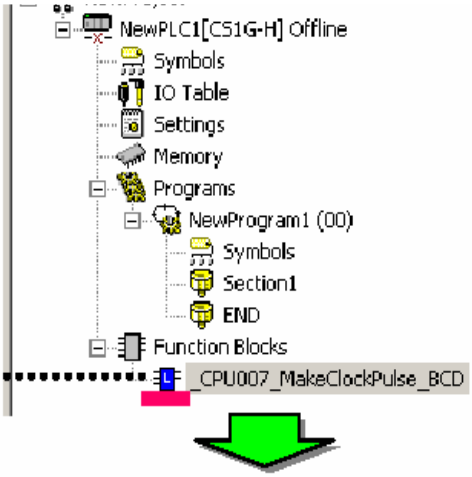
1_ يك خروجي تحت عنوان INV - ENO به متغير هاي خروجي فانكشن بلوك اضافه نماييد .

توجه :

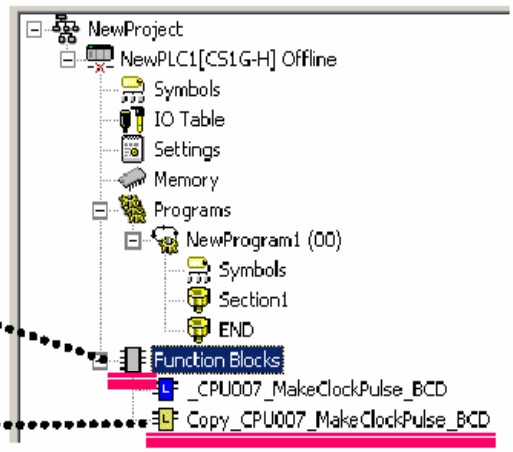
شركت امرن صحت عملكرد فانكشن بلوكهايي كه در آن تغييراتي را ايجاد نموده ايد، تضمين نمي نمايد لذا از صحت عملكرد فانكشن بلوك مورد نظرتان قبل و بعد از انكه در ان تغييري اعما ل نماييد مطمئن شويد .

2. كپي برداري از فا نكشن بلوكهاي موجود در كتابخانه

فانكشن بلوك توليد پالس به مدت زمان مشخص ON و OFF (Make ON Time / OFF Time Clock Pulse in BCD) را، به روشي كه در فصل 1 توضيح داده شد در برنامه تان فراخواني نموده تا علامت گرافيكي معادل ان در پنجره Project WorkSpace نمايش داده شود .

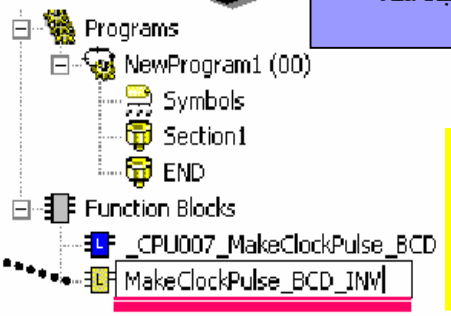


سپس بر روي علامت گرافيكي تعريف ان، ، كليك راست نموده و از منو گشوده شده گزينه Copy را انتخاب نماييد



حال بر روي علامت گرافيكي كليك راست كرده و از منو گشوده شده گزينه Paste را انتخاب نما يد .
علامت گرافيكي معادل فانكشن بلوك جديد كه در واقع كپي از فانكشن بلوك موجود در كتابخانه مي باشد به زير شاخه فانكشن بلوكهاي فراخواني و يا تعريف شده، اضافه خواهد شد .

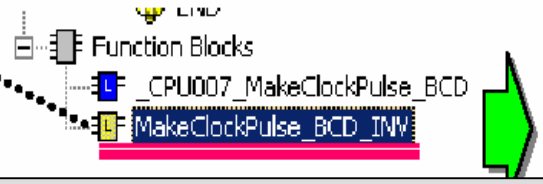
تغيير عنوان فانكشن بلوك ايجاد شده

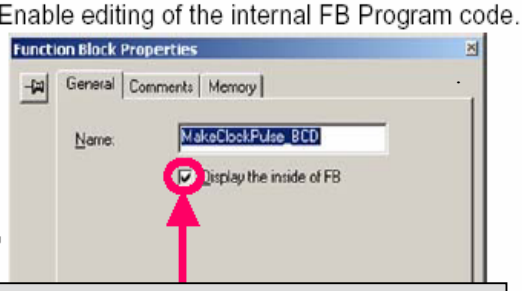


با كليك راست بر روي علامت گرافيكي فانكشن بلوك ايجاد شده و انتخاب گزينه Rename مي توانيد عنوان اين فانكشن بلوك را تغيير دهيد.
ما عنوان MakeClockPulse- BCD- INV را در نظر گرفته ايم

توجه داشته باشيد كه مجاز نيستيد عنوان مورد نظرتان را با '_' آغاز نماييد .
در اين مثال ما عنوان Make Clock Pulse- BCD- INV را براي فانكشن بلوك تغيير يافته مان در نظر گرفته ايم .

بر روي فانكشن بلوك جديد كليك راست نموده و از منو گشوده شده گزينه Property را انتخاب نموده و يا به طور همزمان كليد هاي Alt , ENT را فشار دهيد تا پنجره روبرو گشوده شود



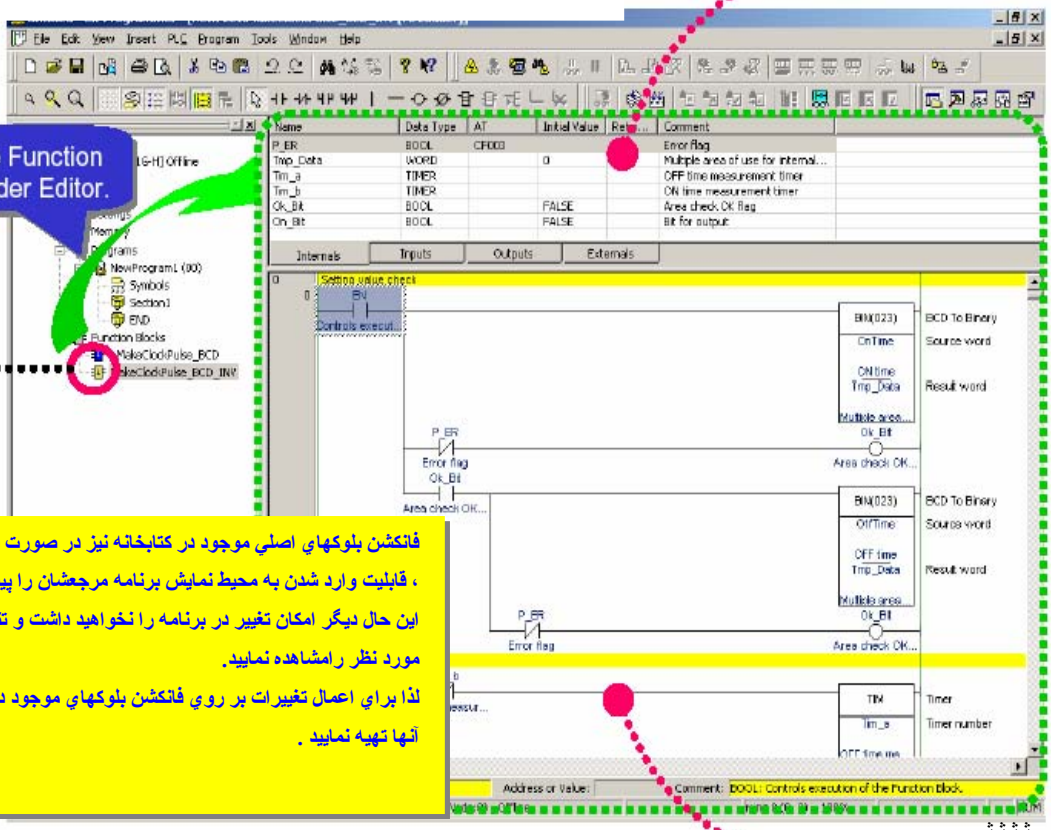


دربخش General اين پنجره ،در قسمت نشان داده شده در شكل، كليك چپ نموده تا در كنار اين گزينه تيك زده شود و قابليت نمايش محتويات برنامه فانكشن بلوك را داشته باشيد

3. اضا فه نمودن يك متغير به فانكشن بلوك

گشايش محيط اصلاح و تغيير برنامه فانكشن بلوك

جدول ورود متغيرها



در ابتـدا بـا انتخـاب علامـت گرافيكـي معادل فانكشن بلوك و دوبـار كليـك بـر روي آن ،وارد محيط برنامه نويسي و اصلاح فانكشن بلوك مورد نظر شويد .

فانكشن بلوكهاي اصلي موجود در كتابخانه نيز در صورت قرار دادن تيك مربوط به نمايش محتوياتشان ، قابليت وارد شدن به محيط نمايش برنامه مرجعشان را پيدا خواهند نمود اما توجه داشته باشيد كه در اين حال ديگر امكان تغيير در برنامه را نخواهيد داشت و تنها مي توانيد برنامه مر تبط با فانكشن بلوك مورد نظر رامشاهده نماييد.

لذا براي اعمال تغييرات بر روي فانكشن بلوكهاي موجود در كتابخانه حتماً مي بايست ابتدا يك كپي از آنها تهيه نماييد .

محيط مربوط به اصلاح برنامه

جدول متغير ها

Name	Data Type	AT	Initial Value	Retai...	Comment
ENO	BOOL		FALSE		Indicates successful execution ...

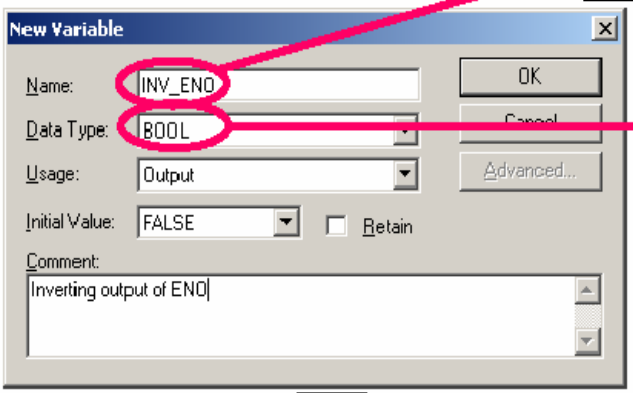
Internals | Inputs | Outputs | Externals

بعد از ورود بـه محيط برنامـه نويسي و اصلاح فانكشن بلوك اينچنين عمل نماييد :

در جدول تعريف متغيرها، قسمت مربوط به تعريف متغيرهاي خروجي را انتخاب نموده و سپس بر روي آن كليك چپ كرده و از منوگشوده شده گزينهInsert variable را انتخاب نماييد تا پنجره مربوط به تعريف يك خروجي جديد گشوده شود .

در ايـن پنجـره در قـسمت Name عنـوان INV-ENO را وارد نمـوده ودر قـسمت DataType نـوع آن را BOOL انتخـاب كرده و سپس OK نماييد.

در اين قسمت نام متغير را وارد نماييد



نـوع متغيـر جديـد را BOOL انتخـاب نماييد

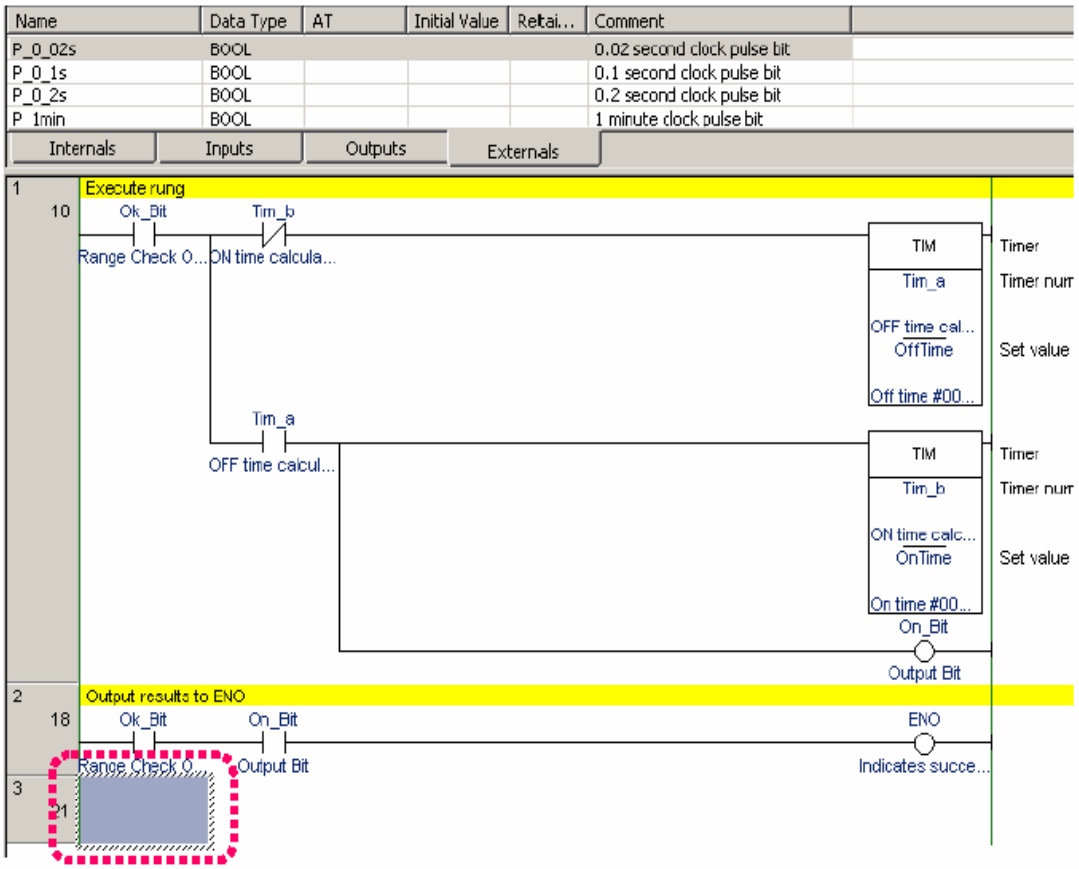
بعد از انجام مراحل فوق متغير خروجي جديد تحت عنوان INV - ENO به فهرست متغير هاي شما اضافه خواهد شد .

Name	Data Type	AT	Initial Value	Retai...	Comment
ENO	BOOL		FALSE		Indicates successful execution ...
INV_ENO	BOOL		FALSE		Inverting output of ENO

4. تغييرات لازم در برنامه مرجع فانكشن بلوك

تغييرات زير را به برنامه نردباني فانكشن بلوك اعمال نماييد :

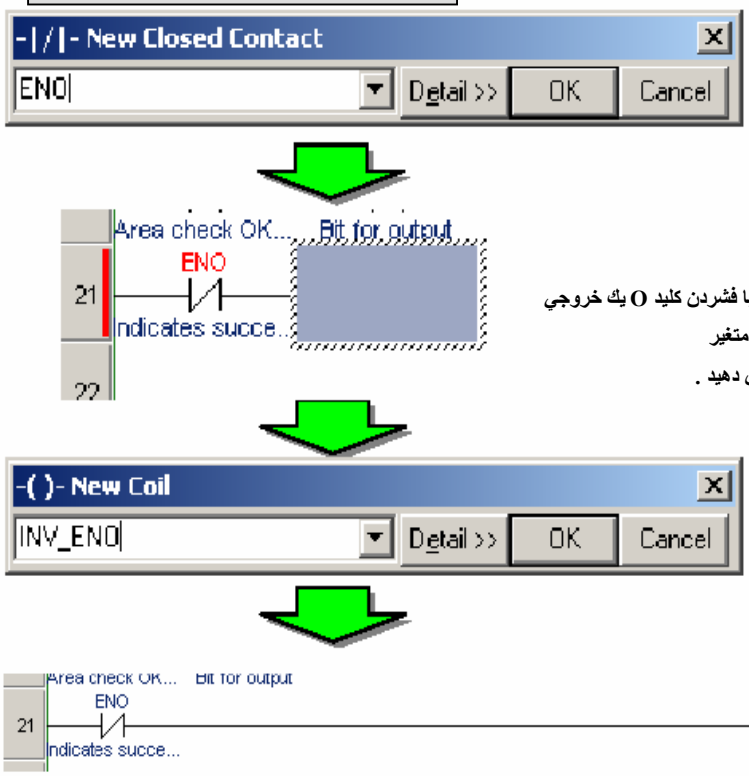
مكان نما را ابتداي خط بعد از انتهايي ترين خط برنا مه قرار دهيد .



با فشردن كليد /
يك كنتاكت بسته را تعريف نماييد
و متغيـر مرجـع ايـن كنتاكـت را
ENO قرار دهيد .



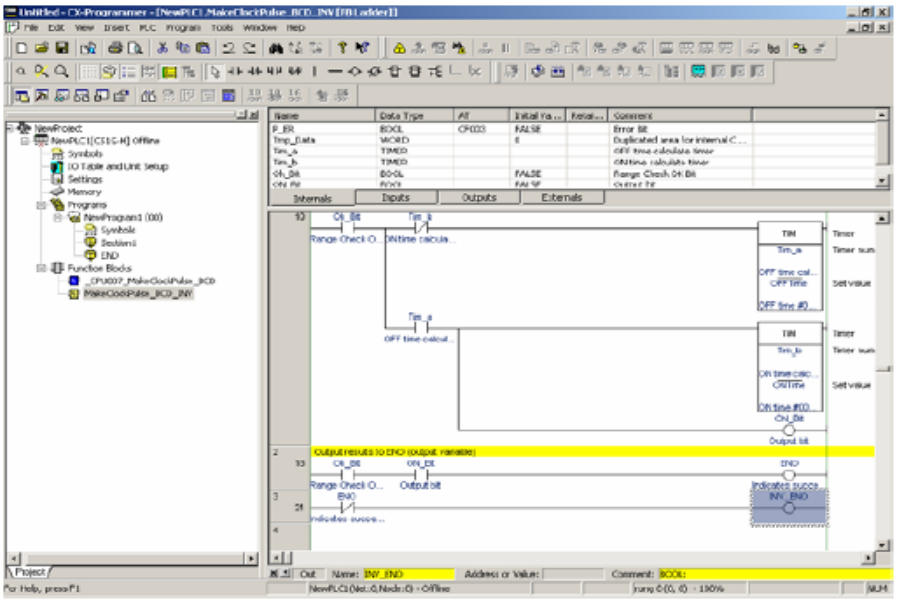
4_1. وارد كردن يك كنتاكت



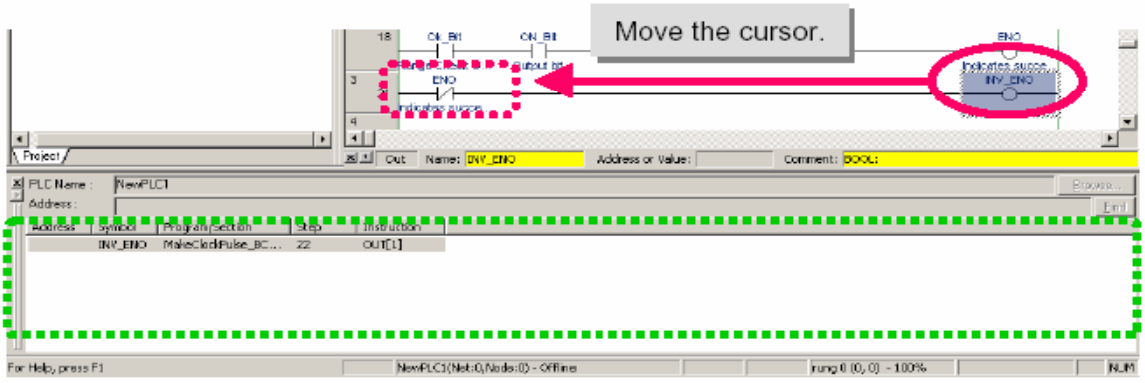
حال بعد از زدن كليد ENT با فشردن كليد O يك خروجي
را فراخواني نموده و به آن متغير
INV - ENO را اختصاص دهيد .

4_2. چك كردن وضعيت متغير هاي به كار رفته در برنامه :

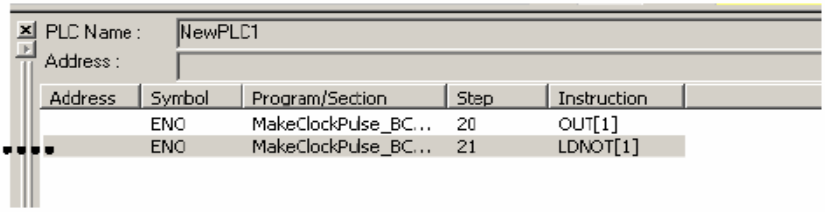
همانند محيط برنامه نويسي اصلي بافشردن همزمان كليدهاي Alt , 4 پنجره نمايش مرجع آدرسهاي به كار رفته در برنامه گشوده خواهد شد(Cross Refrence Window).



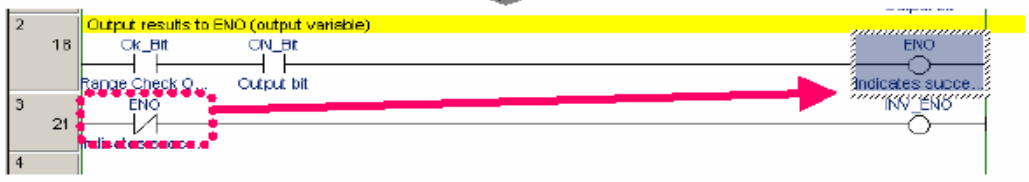
به منظور نمايش پنجره
Cross Reference
به صورت همزمان كليدهاي
Alt , 4
را بفشاريد



با كليك بر روي هر متغير موجود در برنامه ،فهرستي از موارد استفاده آن متغير در برنامه، در اين پنجره نمايش داده خواهد شد .
با كليك بر روي هر يك از خطوط اين فهرست به محل استفاده از آن متغير در برنامه منتقل خواهيد شد همچنين، با استفاده از جهت نما هاي صفحه كليد مي توانيد مابين خطوط مختلف اين فهرست جا بجا شويد .



You can see that variable ENO is used in an output coil in the step No.20 as well.



The cursor in the FB Ladder Editor moves to the output coil in the step No.20.

فصل 4

چگونگي استفاده از زبان برنامه نويسي

(Structured Text) ST

Function Block

1. زبان برنامه نويسي ST چيست ؟

زبان برنامه نويسي ST يك زبان سطح بالا براي كنترلرهاي صنعتي (اكثراً PLC ها) و برمبناي استاندارد IEC 61131- 3 بوده و شامل بسياري از دستورات كنترلي از جمله - IF - THEN - ELSE WHILE - FOR - IF- END ، حلقه و بسياري از دستورات رياضياتي همچون سينوس و لگاريتم مي باشد.

لذا براي كاربردهاي محاسباتي بسيار مناسب است .

نرم افزار CX- Programmer اين زبان را با تطابق بر استاندارد IEC 61131- 3 ساپورت نموده است .

دستورات محاسباتي موجود در نرم افزار CX- Programmer نسخه 5 و6 عبارتند از :

سينوس SIN

كسينوس COS

تانژانت TAN

آرك سينوس ASIN

آرك كسينوس ACOS

آرك تانژانت ATAN

ريشه درجه دو SQRT

قدر مطلق ABS

لگاريتم LOG

لگاريتم طبيعي LN

```
(* Initial Settings *)
XMT[1] := 2;
XMT[2] := 7;
N := 2;

(* CRC16 *)
CRCTMP := 16#FFFF;
FOR I := 1 TO N DO
    CRCTMP := CRCTMP XOR XMT[I];
    FOR J := 1 TO 8 DO
        CT := CRCTMP AND 1;
        IF CRCTMP < 0 THEN
            CH := 1;
            CRCTMP := CRCTMP AND 16#7FFF; (* CRCTMP & 0x7FFF *)
        ELSE
            CH := 0;
        END_IF;
        UINT_CRCTMP := WORD_TO_UINT(CRCTMP) / 2;
        CRCTMP := UINT_TO_WORD(UINT_CRCTMP);
        IF CH = 1 THEN
            CRCTMP := CRCTMP OR 16#A000;      (* CRCTMP OR 0xA000 *)
        END_IF;
        IF CT = 1 THEN
            CRCTMP := CRCTMP XOR 16#A001; (* CRCTMP XOR 0xA001 *)
        END_IF;
    END_FOR;
END_FOR;

IF CRCTMP < 0 THEN
    CL := 1;
    CRCTMP := CRCTMP AND 16#7FFF;     (* CRCTMP & 0x7FFF *)
ELSE
    CL := 0;
END_IF;

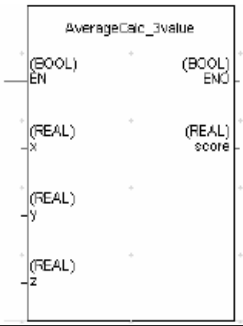
C_1 := CRCTMP AND 16#FF;                (* CRCTMP & 0xFF *)
CRCTMP := CRCTMP AND 16#7F00;     (* CRCTMP & 0x7F00 *)
UINT_CRCTMP := WORD_TO_UINT(CRCTMP) / 256;
C_2 := UINT_TO_WORD(UINT_CRCTMP);
```

مرجع : استاندارد IEC 61131 كه يك استاندارد جهاني براي PLC ها مي باشد . كه توسط كميسيون جهاني الكترو تكنيكال(IEC) تدوين شده است .

اين استاندارد شامل 7 قسمت مي باشد كه قسمت 3 آن مربوط به برنامه نويسي PLC مي باشد .

2. توضيح يك برنامه نمونه

در اين مثال چگونگي ايجاد يك فانكشن بلوك به زبان ST براي محاسبه ميانگين ضخامتهاي اندازه گيري شده را نشان خواهيم داد.



مقادير اندازه گيري شده مي بايست به فرم اعداد حقيقي ذخيره شوند ،

اين فرمت اجازه ذخيره سازي مقادير با طول 32 بيت ودر رنج زير را ممكن مي سازد:

-3.402823 * 10^38 ~ +3.402823*10^38

نام فانكشن بلوك : AverageCalc- 3Value

سمبل هاي ورودي : X (از نوع اعداد حقيقي) Y (از نوع اعداد حقيقي) Z (از نوع اعداد حقيقي)

سمبل خروجي : SCORE (از نوع اعداد حقيقي)

برنامه نوشته شده به زبان ST :

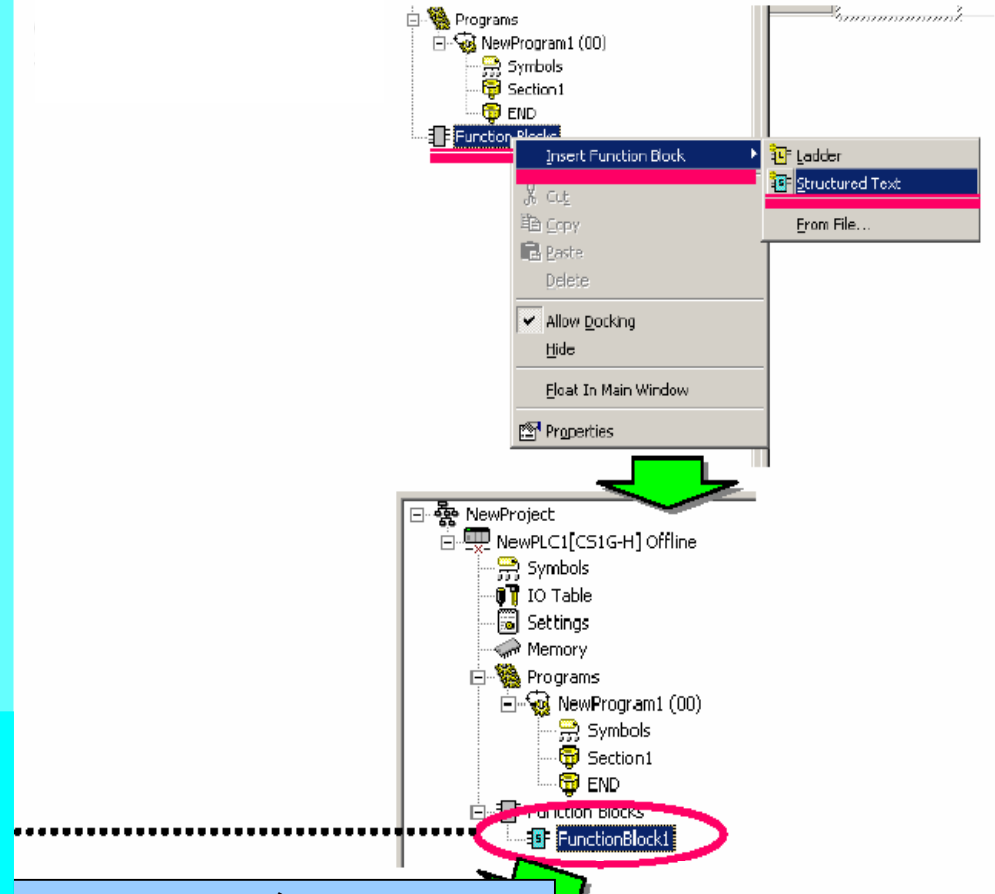
```
SCORE :=( X+Y+Z)/3.0 ;
```

مقداردهي يك عبارت با علامت =: صورت مي پذيرد .

اگر " ; " انتهاي خط فراموش شود خطا تشخيص داده خواهد شد .

4. چگونگي ايجاد يك فانكشن بلوك به زبان ST

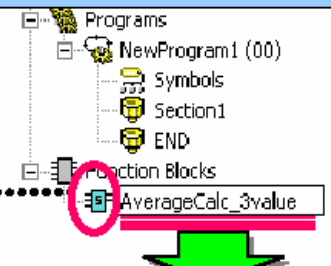
در پنجره Project work Space ، بر روي علامت گرافيكي تعريف فانكشن بلوك ، كليك راست نموده و از منوي گشوده شده گزينه Insert Function Block را انتخا ب كرده ،و سپس گزينه Structure Text را انتخاب نماييد .



در اين صورت يك علامت گرافيكي جديد تعريف فانكشن بلوك در زير شاخه Function Blocks اضافه خواهد شد .

تغيير عنوان فانكشن بلوك

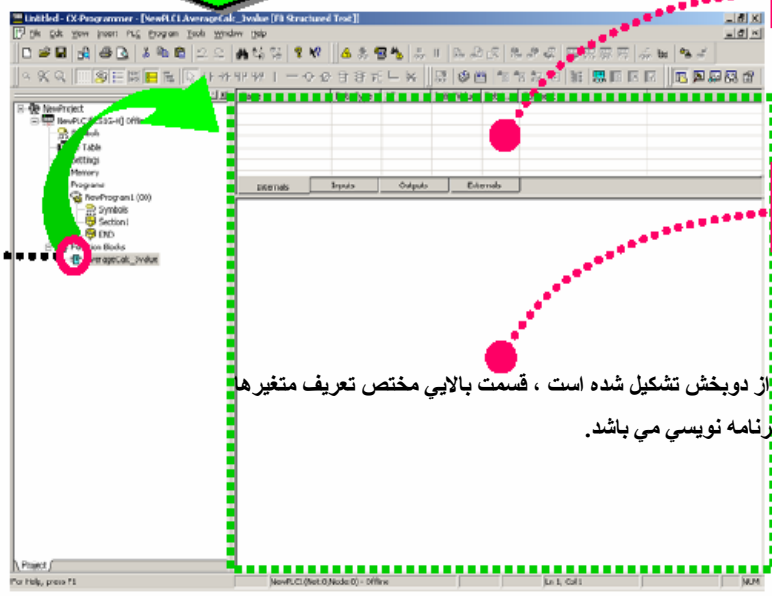
تغيير عنوان فانكشن بلوك : بر روي علامت گرافيكي تعريف فانكشن بلوك ايجاد شده ،كليك راست نماييد و گزينه Rename را انتخاب نماييد و عنوان موردنظرتان را واردنماييد به عنوان مثال [AverageCalce- 3Value]



توجه داشته باشيد كه مجاز نمي باشيد نام مورد نظرتان را با ' -' اغاز نماييد

وارد شدن به محيط برنامه نويسي ST

با دوبار كليك بر روي علامت گرافيكي تعريف فانكشن بلوك به محيط برنامه نويسي ST منتقل شويد .



جدول متغيرها

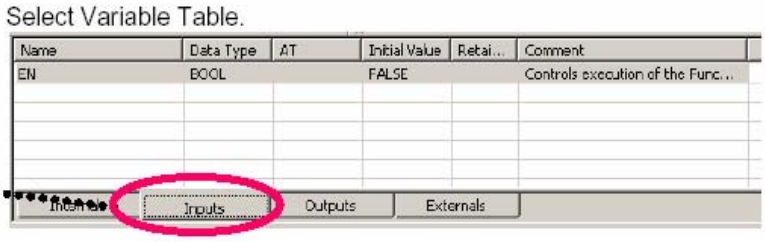
محيط برنامه نويسي ST

محيط برنامه نويسي ST از دوبخش تشكيل شده است ، قسمت بالايي مختص تعريف متغيرها و قسمت پاييني مختص برنامه نويسي مي باشد.

4. واردكردن متغير ها در برنامه فانكشن بلوك

بعد از تعريف يك فانكشن بلوك مـي بايـست متغيرهـاي آن را تعريـف نماييد.

بدين منظور از پنجره اي كـه در قسمت بـالايي محيط برنامـه نويـسي فانكـشن بلـوك وجـود دارد پنجـرة متغيرهـاي ورودي را انتخاب نماييد .



در اين پنجره كليك راست نموده و از منـو گشوده شده گزينه Insert را انتخاب نماييد .

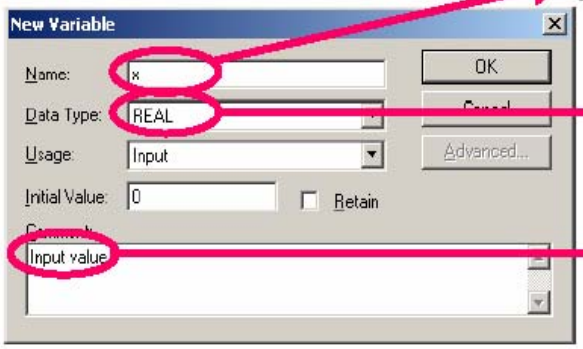
در اين حال پنجره زير گشوده مي شودو مـي توانيـد در ان اطلاعـات مـورد نيـاز از جمله

- نام متغير
- نوع ان
- توضيحات مورد نياز را

وارد نماييد .

بعـد از تعريـف هرمتغيـر ،عنـوان آن متغير

به جدول متغيرها اضافه خواهد شد .

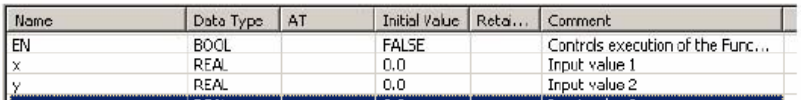


ورود نام متغير: X

انتخاب فرمت متغير: Real

توضيحات مورد نياز را وارد نماييد

متغيرهاي ورودي X, Y,Z ومتغير خروجي SCORE را به همين ترتيب تعريف نماييد



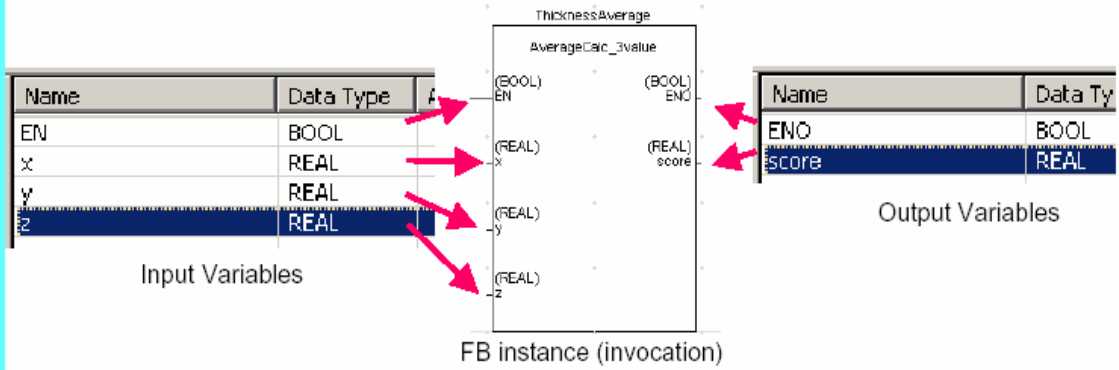
متغير هاي ورودي



متغيرهاي خروجي

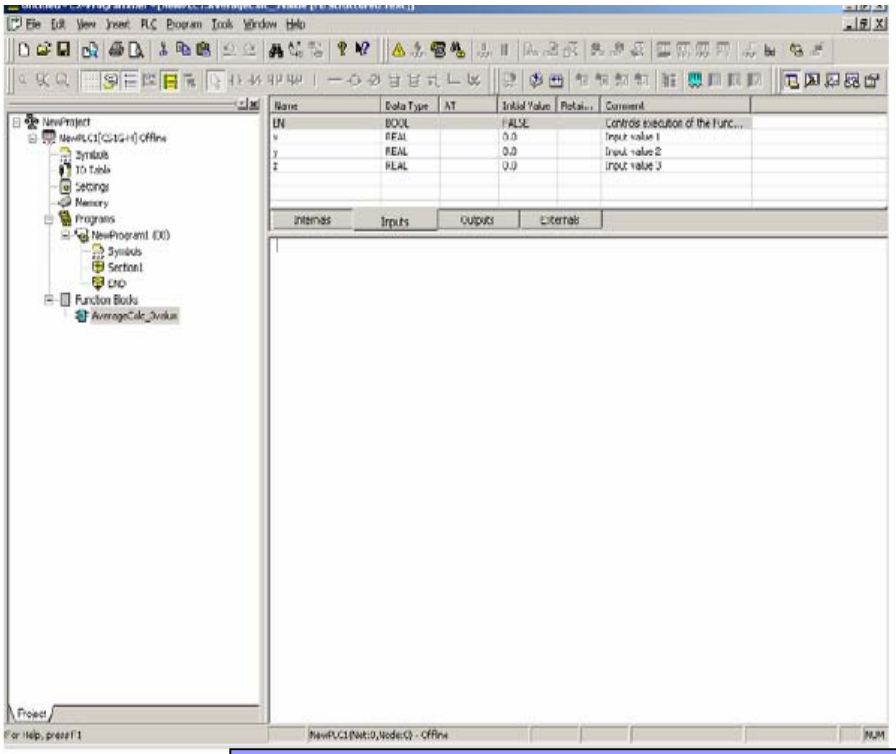
انجام عمليات كپي و Paste در تعريف متغيرها امكان پذير است

ترتيب متغيرها دردستور مستقل معادل فانكشن بلوك در برنامه اصلي ، همسان با ترتيب قرار گرفتن متغير ها در جدول تعريف متغير ها مي باشد و در صورت نياز به تغيير ترتيب آنها ،كافيست متغيرهاي تعريف شده در جدول را با انتخاب و كشيدن توسط موس به ترتيب دلخواه قرار دهيد .



5. برنامه نويسي ST:

در محيط تعريف فانكشن بلوك وارد پنجره پايييني كه مختص برنامه نويسي مي باشد شده سپس دستورات مورد نظر را وارد نماييد .



در برنامه ميانگين گيري 3 عدد ، بعد از تعريف متغير هاي ورودي و خروجي در اين قسمت مي نويسيم :SCORE := (X+ Y+ Z)/3.0

```
score :=(x + y + z) / 3.0;
```

در صورت تعريف متغير ها به صورت Real حتماً مـي بايـست آنهـا را در فرمـت اعـشاري (پوينـت دار) بنويسيد .براي مثال 3 را به صورت 3.0 مي نويسيم .

براي نوشتن توضيحات در ضمن برنامه مي بايست قبل از توضيحات (* و بعد از آن نيز *) ، را قرار دهيد به عنوان مثال :

```
(* Created by Suzuki 5/21/2004 *)
score := (x + y + z) / 3.0;
```



در صورت نياز به راهنمايي در مورد دستورات برنامه نويسي ST ،كافيست در محيط برنامه نويسي كليك راست نموده و از منو گشوده شده گزينه ST Help را انتخاب نماييد تا راهنماي برنامه نويسي به زبان ST گشوده شود

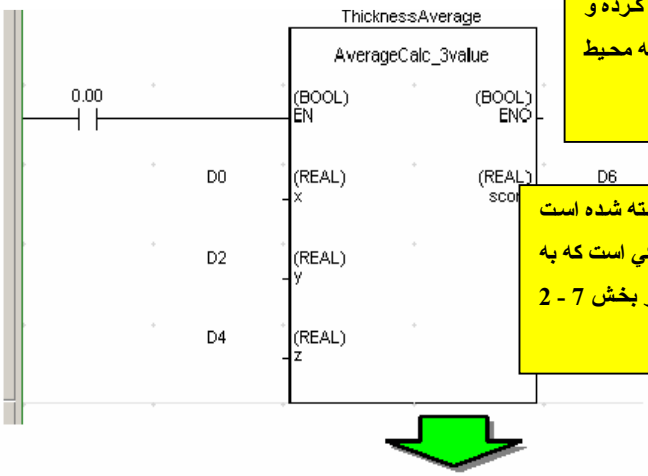
6. وارد كردن فانكشن بلوك در برنامه نردباني و خطا يابي

نام فانكشن بلوك : Thickness Avarage

متغيرهاي ورودي به كار رفته در حافظه هاي D0 , D2 , D4 ذخيره سازي شده اند

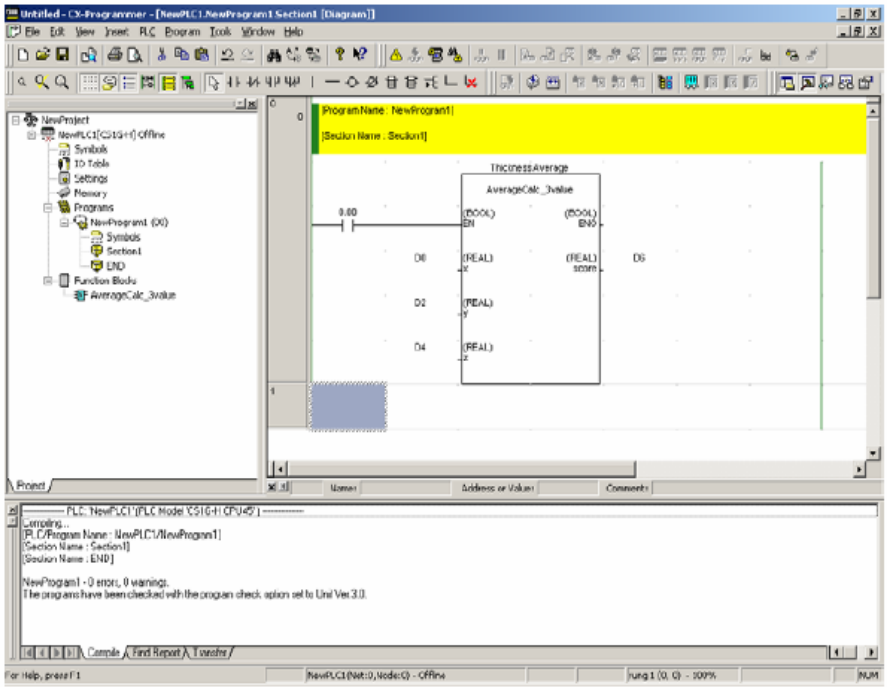
متغير خروجي را در حافظه D6 ذخيره مي نماييم

زماني كه در محيط برنامه نويسي نردباني مي باشيد در صورتيكه بر روي فانكشن بلوك مورد نظر كليك كرده و همزمان كليدهاي SHIFT و F را فشار دهيد به محيط تعريف فانكشن بلوك مورد نظر منتقل مي گرديد .



ورود متغيرهاي فانكشن بلوكي كه با زبان ST نوشته شده است كاملاً همانند وارد كردن پارامترها براي فانكشن بلوكي است كه به صورت نردباني نوشته شده باشد و قبلاً كاملاً در بخش 7 - 2 توضيح داده شد .

Perform a programs check before transferring the program.



به همان روش كه در قسمت 9 -2 توضيح داده شده برنامه را قبل از انتقال به كنترلر چك كرده و از صحت آن اطمينان حاصل نماييد .

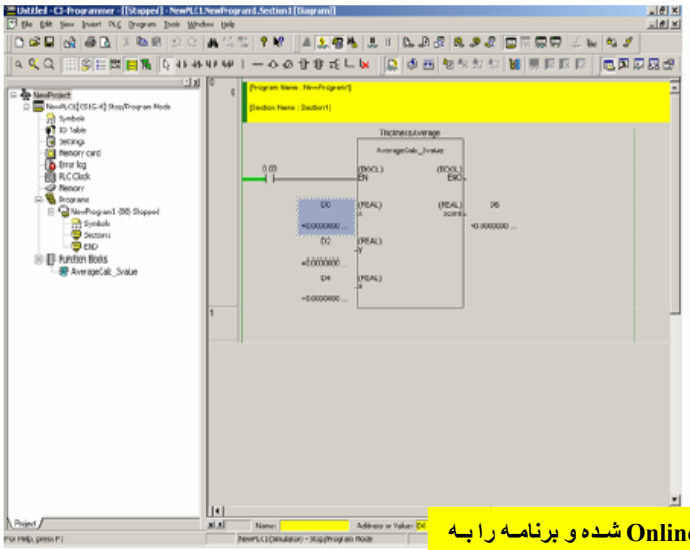
بعد از استفاده از فانكشن بلوك در برنامه اصلي نيز امكان تغيير متغيرهاي آن وجود دارد .

در صورت تغيير، ويرايشگر نردباني رنگ سمت چپ خطي كه فانكشن بلوك آن تغيير كرده است را قرمز مي نمايد .

در اين هنگام بر روي فانكشن بلوك به كار رفته در برنامه اصلي كليك راست كرده و از منو گشوده شده گزينه Update Function block Instance (U) را انتخاب نماييد .

7. انتقال برنامه:

توسط سيمولاتور با كنترلر Online شده و برنامه را به كنترلرمجازي منتقل نماييد



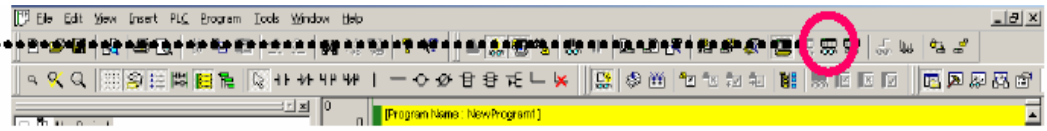
توسط سيمولاتور با كنترلر مجازي Online شده و برنامه را به كنترلر منتقل نماييد كه چگونگي اين امر در بخش 10 -2 توضيح داده شد .

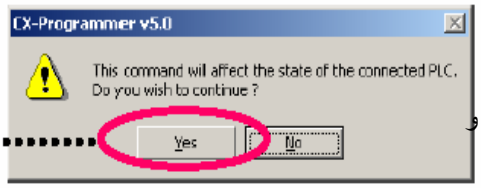
كنترلر را در حالت مانيتور قرار دهيد

بدين منظور بر روي علامت گرافيكي زير كليك نماييد.

در حالت مانيتور وضعيت ON / OFF كنتاكتها و كويلها و نيز مقادير متغيرها در هر لحظه نمايش داده مي شود

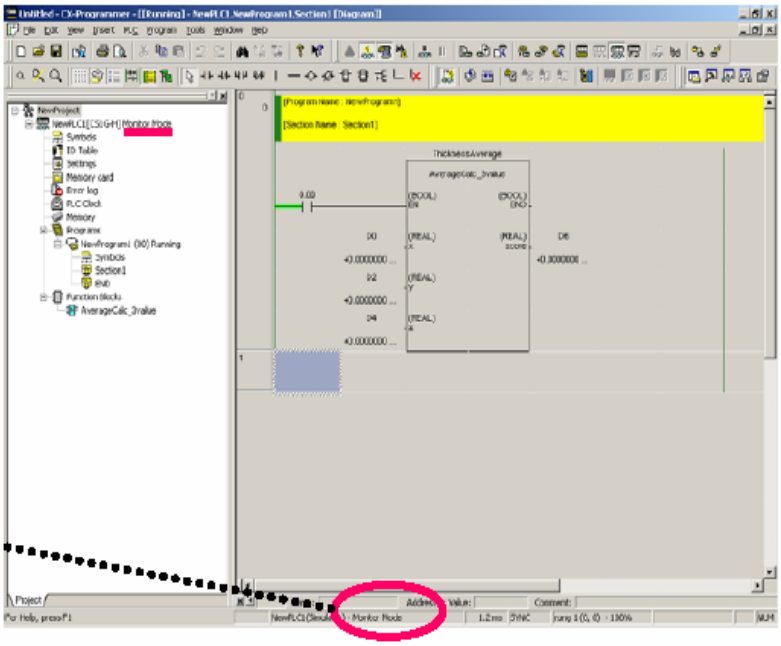
Click





Click [Yes]

در هنگام تغيير حالت كاري پنجره روبرو نمايش داده مي شود تا در صورت اطمينان حالت كاري را تغيير دهيد و به منظور تغيير حالت كاري گزينه yes را انتخاب نماييد .

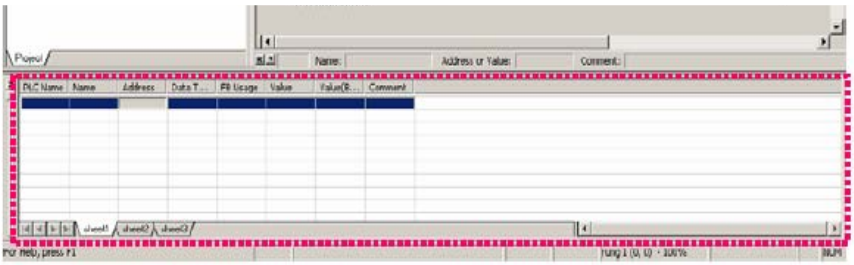


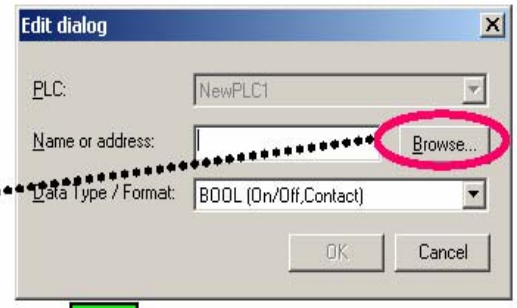
در اين حال در Status Bar ,عبارت Monitor Mode نمايش داده مي شود .

8. مانيتور نمودن چگونگي اجراي فانكشن بلوك:

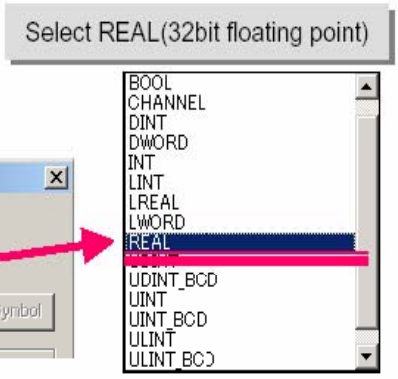
نمايش مقادير پارامتهاي به كار رفته در فانكشن بلوك در هر لحظه با استفاده از Watch window :

بدين منظور با فشردن همزمان كليدهاي ALT + 3 پنجـــره نمـــايش مقـــادير را بگشاييدسپس بـا فشردن كليد ENT پنجره تنظيمات را بگشاييد



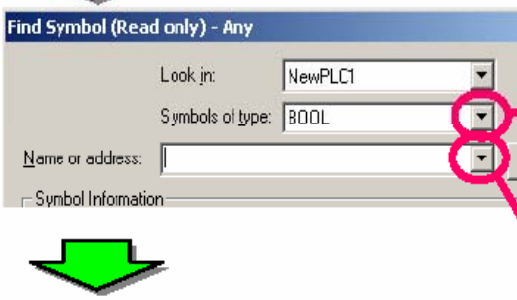


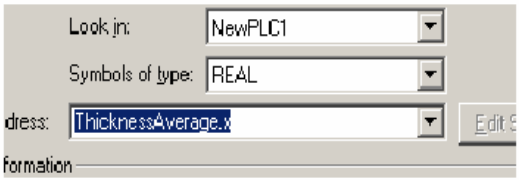
حال گزينه Browse را انتخاب نماييد تا پنجره انتخاب متغيرها گشوده شود

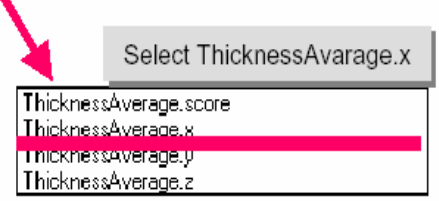


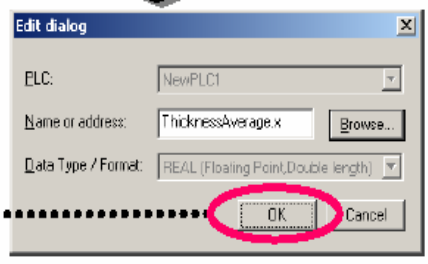
در قسمت Symbols of Type با فشردن فلش كناري از فهرست گشوده شده نوع Real را انتخاب نماييد .

سپس با كليك بر فلش كناري بخش Name of address نام متغير مورد نظرتان را انتخاب نماييد .





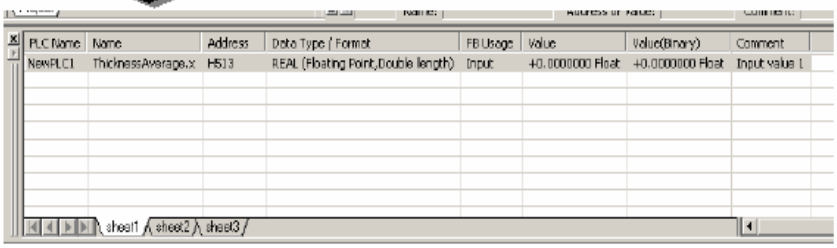




با كليك كردن بر روي گزينهOK اين پنجره بسته و محتويات آدرس مورد نظر ، مانيتور خواهد شد .

زماني كه متغير هاي داخلي را مانيتور مينماييد امكان ثبت نمودن يك مجموعه از متغيرها نيز علاوه برامكان ثبت نمودن مجزاي هر يك از انها ، در پنجره Watch Window وجود دارد .

براي جزئيات بيشتر به بخش 5 - 8 مراجعه نماييد



حال به توضيح يك نمونه برنامه هاي شرطي نوشته شده به زبان ST ،خواهيم پرداخت :

در اين مثال ، ميانگين سه ضخامت اندازه گيري شده ، محاسبه مي گردد ونيز كنترل مي شود كه اين مقدار در يك محدودة خاص كه با تعيين حدود بالاو پايين اش مشخص مي شود قرار دارد يا خير .

عنوان فانكشن بلوك :	**OutputOfDecisionResult**
سمبلهاي ورودي :	**Score (Real) ,set over (Real) ,set under (Real)**
سمبلهاي خروجي :	**OK (Bool) ,OverNG(Bool) , UnderNG (Bool)**

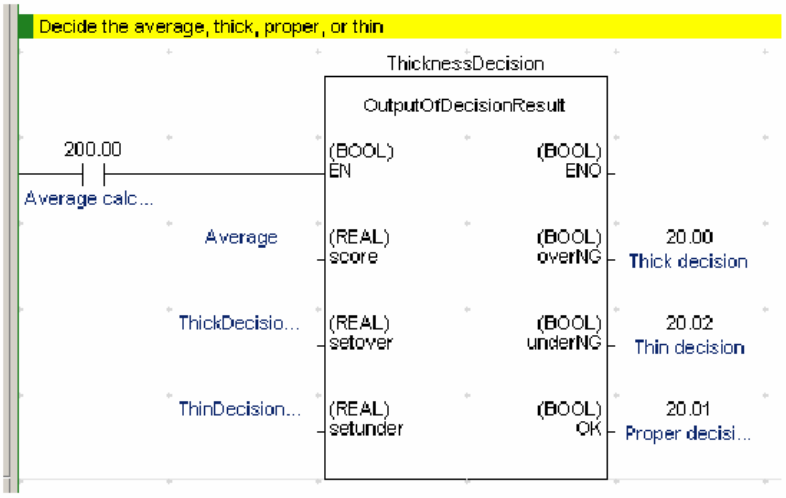
```
ST program:

IF score > setover THEN
  underNG := FALSE;                    (*اگر ميانگين بيشتر از حد بالايي شد *)
  OK := FALSE;                         (*فلگ مربوط به گذشتن از حد پايين را خاموش كن *)
  overNG := TRUE;                      (*فلگ مربوط به وضعيت نرمال را خاموش كن *)
                                       (*فلگ مربوط به گذشتن از حد بالا را روشن كن *)
ELSIF score < setunder THEN            (*اگر ميانگين كمتر و يامساوي حد بالايي ولي كمتراز حد پاييني شد *)
  overNG := FALSE;                     (*فلگ مربوط به گذشتن از حد بالا را خاموش كن *)
  OK := FALSE;                         (*فلگ مربوط به وضعيت نرمال را خاموش كن *)
  underNG := TRUE;                     (*فلگ مربوط به گذشتن از حد پايين را روشن كن *)

ELSE                                   (*اگر ميانگين كمتر از حد بالايي وبيشتراز حد پاييني بود *)
  underNG := FALSE;                    (*فلگ مربوط به گذشتن از حد بالا را خاموش كن *)
  overNG := FALSE;                     (*فلگ مربوط به گذشتن از حد پايين را خاموش كن *)
  OK := TRUE;                          (*فلگ مربوط به وضعيت نرمال را روشن كن *)

END_IF;
```

اين فانكشن بلوك در محيط برنامه نويسي نردباني به فرم زير فراخواني مي شود .



فصل 5:

ايجاد يك برنامه با به كار گيري فانكشن بلوك

Function Block

1. ديد كلي

در اين فصل چگونگي تركيب فانكشن بلوكها رادر برنامه كاربر توضيح خواهيم داد .

2. مراحل ايجاد برنامه :

در زير مراحل كلي تركيب برنامه و فانكشن بلوك را توضيح داده و در اين رابطه مثالي را نيز بيان خواهيم نمود :

همواره مراحل زير را در تعريف هر پروژه در نظر بگيريد :

1. تعريف برنامه
2. ايجاد اجزاء

(1_2) تعريف فانكشن بلوكها

(2 -2) رفع اشكالات فانكشن بلوك

(2 -3) ذخيره سازي فانكشن بلوك ايجاد شده دركتابخانة فانكشن بلوكهاي كاربر

3. فراخواني فانكشن بلوكها در برنامه

(1_3) فراخواني فانكشن بلوكها در برنامه

(3 -2) استفاده از فانكشن بلوكها متناسب با نياز برنامه

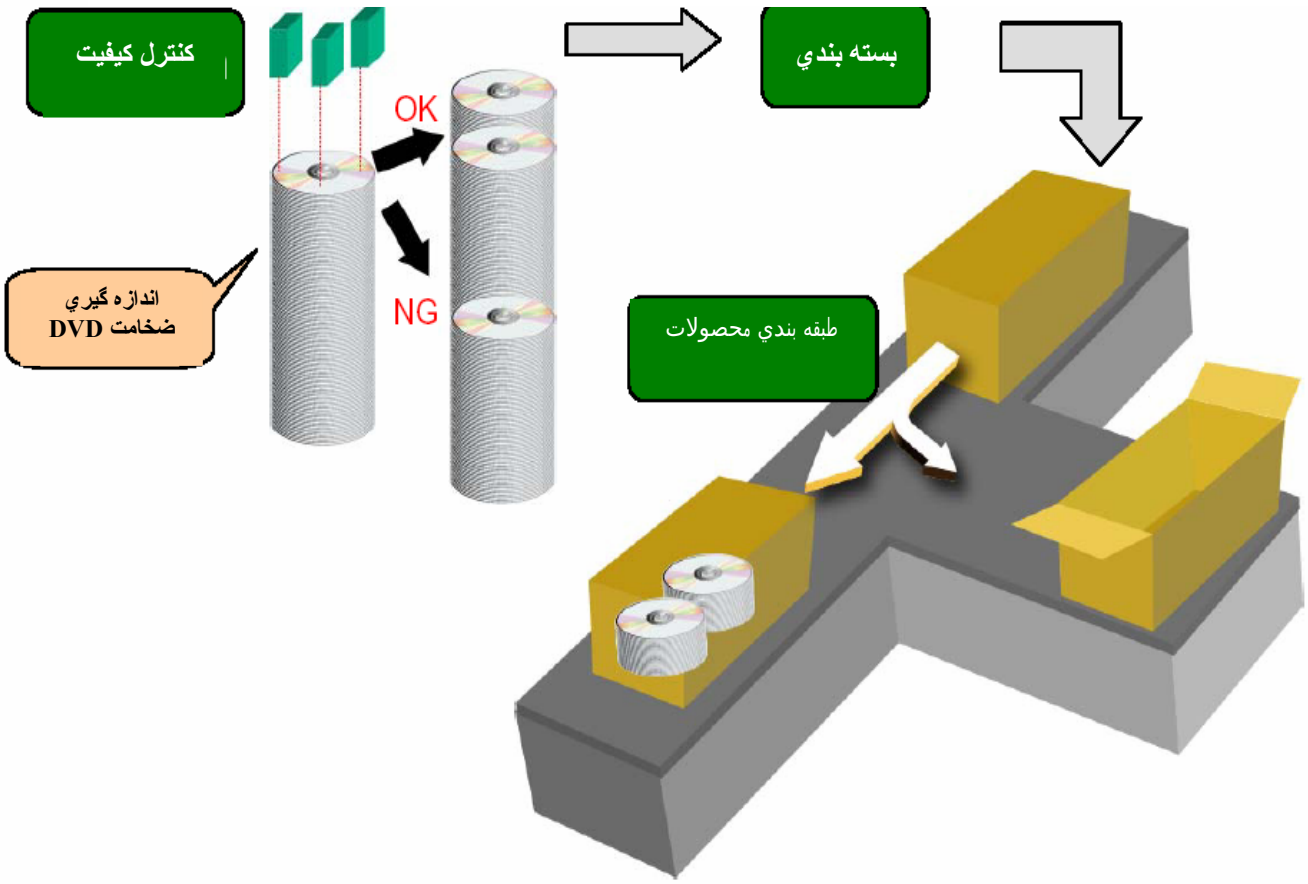
(3 -3) رفع اشكالات برنامه

4. آغاز كار برنامه

3. يك كاربرد نمونه:

مراحل كنترل كيفيت خط توليد DVD ، را به عنوان يك مثال نمونه مطرح كرده ايم .

اين امر به سه مرحلة - كنترل كيفيت(يا اندازه گيري ضخامت DVD) - بسته بندي - طبقه بندي محصولات تفسيم مي شود .

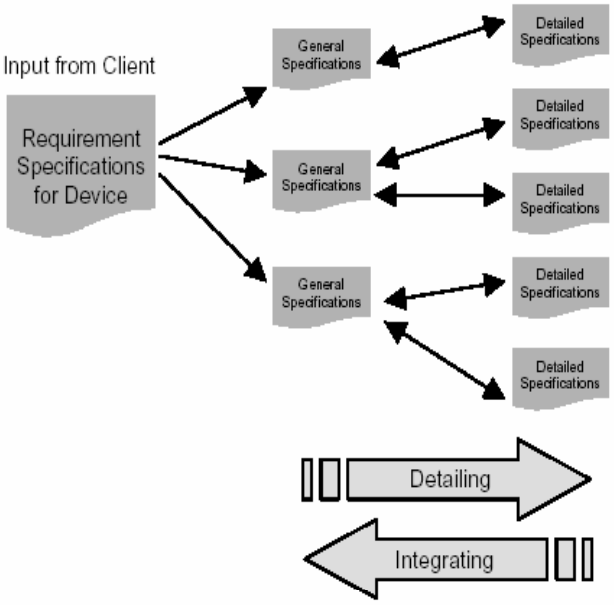


4. چگونگي و مراحل پيشبرد برنامه :

به منظور اجرا ي كاربرد موردنظر، مي بايست سخت افزار را با قابليتهاي نرم افزاري تركيب نموده تا نياز خود را برآورده سازيم .
دربخش بعدبه توضيح مراحل طراحي پروژه براي كاربرد مطرح شده در اول اين بخش خواهيم پرداخت .-

4_1. مروري بر مراحل طراحي :

قبل از هر كاري مي بايست نيازهايتان را متناوباً به قسمتهاي جزئي تر تقسيم كرده و خصوصيات مورد نظرتان را همانطور كه در شكل نشان داده شده است طبقه بندي و به زير نيازها تقسيم نماييد.



4_2. تعيين مراحل مورد نياز :

در مثال مربوط به توليد DVD ، مراحل مورد نياز به صورت زير مي باشند:

مرحله 1 . DVD مي بايست ابتدا از Loader وارد خط شود .

مرحله 2 . ضخامت DVD مي بايست در 3 نقطه اندازه گيري شود .

ميانگين ضخامتها ي اندازه گيري شده محاسبه شده و اگر در رنج مجاز بود ، DVD مي بايست در رده محصولات تاييد شده ذخيره شود و در غير اين صورت در رده محصولات مردود قرار گيرد .

مرحله 3. DVD هاي تاييد شده مي بايست در جعبه بسته بندي شوند .

مرحله 4 . جعبه هاي DVD ميبايست با پوشش كاغذي بسته بندي شوند .

مرحله 5 . بسته ها مي بايست به دو نوع طبقه بندي گردند .وتعداد دفعات اين امر مي بايست براي ارزيابي طول عمر ليميت سويچ مجاور فعال كننده انتخاب مابين دو نوع كالا ، متناوباً شمارش شود .

مرحله 6. . ساير مراحل

بررسي مراحل بالا با جزئيات بيشتر و تشخيص مراحل مشابه

با بررسي جزييات هر مرحله مي توانيم مراحل مشابه را شناسايي نموده و يك مرحله را به طور كلي براي چندين نياز به كار برد .

كنترل فعال كننده (مثالي از مراحل مشابه)

در اين مثال ملاحظه خواهيد نمود كه تقسيم بندي قطعات تاييد شده و معيوب و نيز جعبه هاي بسته بندي شده، مشابه مي باشد و براي هر دو منظور مي توان يك برنامه كنترلي را به كار برد .

در زير نيازهاي مرتبط با فرايند كنترلي فوق الذكر فهرست شده است :

- **فرايند داراي دو فعال كننده براي كنترل حركت در يكي از دو جهت مي باشد كه وابسته به شرايط ورودي عمل مي نمايند .**
- **عملكرد در اين دو جهت مي بايست با هم اينتر لاك داشته باشند.**
- **فرايند يك ورودي RESET موقعيت ، نيز دارا مي باشد .**

محاسبه وچك كردن ميانگين ضخامتها براي خارج نشدن از رنج مجاز(مثالي از يك فرايند عمومي)

اين فرايند مثالي از فرايندهاي عمومي وقابل استفاده مجدد در بسياري از برنامه هاي كنترلي مي باشد .

در اين مثال ، به فرايندي جهت محاسبه ميانگين سه ضخامت اندازه گيري شده و نيز چك نمودن آنكه آيا مقدار محاسبه شده در رنج مجاز مي باشد يا خير ، نياز است .

در زير مراحل مورد نياز براي اين فرايند آمده است .

- **ميانگين 3 مقدار اندازه گيري شده مي بايست محاسبه گردد**
- **مقدار محاسبه شده مي بايست با رنج بالايي و پاييني مقايسه شود تا تشخيص داده شود كه اين مقدار در رنج مجاز است يا خير**

دو مرحله بالا ، اجزاء اصلي برنامه كنترلي ما بوده و براي تعريف آنها دو فانكشن بلوك تحت عنوانهاي "Actuator Control" و " AvgValue- ThresholdCheck " را تعريف خواهيم نمود .

1_3_4. ايجاد مراحل اصلي برنامه در غالب فانكشن بلوك

استفاده مجدد ازيك فانكشن بلوك به منظور تعريف قسمتهاي مشابه در برنامه به طور موثر ، برنامه نويسي را بهبود داده .

لذا براي سهولت بيشتر برنامه نويسي ، فانكشن بلوك متناظر با فرايند مورد نياز را تعريف و در هر جا كه لازم باشد فراخواني مي نماييم. اين امر به درك بهتر روند برنامه و نيز وضعيت ورودي / خروجي ها ،بدون نياز به مطالعه جزئيات آنها ،كمك شاياني مي نمايد .

حال به توضيح چگونگي ايجاد فانكشن بلوكهاي مورد نياز خواهيم پرداخت .

4_3_2. توضيح مراحل ايجاد فانكشن بلوكهاي مورد نياز:

نام فانكشن بلوك: Actuator Control

اين فانكشن بلوك ميبايست به زبان نردباني نوشته شود زيرا براي كنترل مراحل ترتيبي به كار خواهد رفت

جدول متغير هاي ورودي

Name	Data Type	AT	Initial Value	Retained	Comment
EN	BOOL		FALSE		Controls execution of the Function Block.
PosDirInput	BOOL		FALSE		Input for positive direction
NegDirInput	BOOL		FALSE		Input for negative direction
LSpos	BOOL		FALSE		Limit switch for positive direction
LSneg	BOOL		FALSE		Limit switch for negative direction

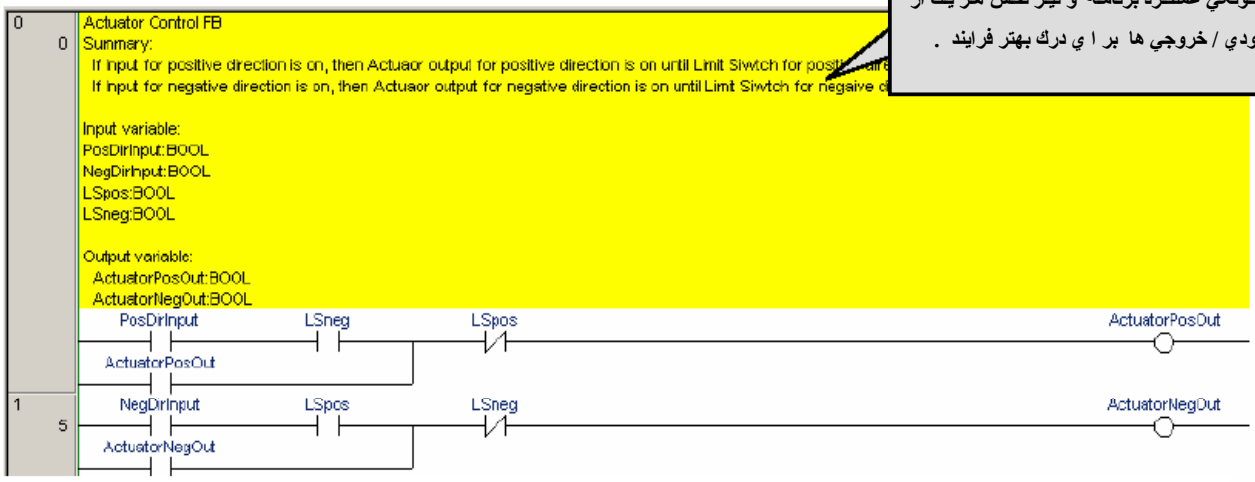
جدول متغير هاي خروجي

pe	AT	Initial Value	Retained	Comment
BOOL		FALSE		Indicates successful execution of the Function Block ...
BOOL		FALSE		Actuator output for positive direction
BOOL		FALSE		Actuator output for negative direction

جدول متغير هاي داخلي

None.

توضيحات خطوط برنامه، به منظور بيان توضيح چگونگي عملكرد برنامه و نيز نقش هر يك از ورودي / خروجي ها بر ا ي درك بهتر فرايند .



نام فانكشن بلوك: AvgValue- ThresholdCheck

اين فانكشن بلوك ميبايست به زبان ST نوشته شود زيرا براي انجام مراحل محاسباتي و مقايسه اي به كار خواهد رفت

جدول متغير هاي ورودي

Name	Data Type	AT	Initial Value	Retained	Comment
EN	BOOL		FALSE		Controls execution of the Function Block.
Input1	REAL		0.0		Input value 1
Input2	REAL		0.0		Input value 2
Input3	REAL		0.0		Input value 3
UpLimit	REAL		0.0		Upper limit value
LowLimit	REAL		0.0		Lower limit value

جدول متغير هاي خروجي

Name	Data Type	AT	Initial Value	Retained	Comment
ENO	BOOL		FALSE		Indicates successful execution of the Function Block...
Result	BOOL		FALSE		OK or NG judge flag

جدول متغير هاي داخلي

	s Type	AT	Initial Value	Retained	Comment
AvgValue	REAL		0.0		

```
(* Agarage value calculation and check of threshould for three values *)

AvgValue := ( Input1 + Input2 + Input3 ) / 3.0;              (* Divides Input 3 values by 3 *)
IF ((AvgValue <=UpLimit) AND (AvgValue >=LowLimit)) THEN (* Compare the agarage value if below of upper limit or above of lower limit *)
    Result := TRUE;
    ELSE
    Result := FALSE;
END_IF;
```

تا آنجا كه ممكن است ، از نامهاي عمومي به جاي نامهاي خاص در تعريف متغيرها و نيز نامگذاري فا نكشن بلوكها استفاده نماييد .

4_4_1. :فراخواني فانكشن بلوكهاي ايجاد شده در برنامه

با تركيب فانكشن بلوكهاي فوق برنامه لازم براي كاربرد مورد نظر مان را ايجاد خواهيم نمود

4_4_2. :تركيب فانكشن بلوكهاي تعريف شده براي اجراي برنامه كنترلي ،انتخاب DVD ها با ضخامت تاييد شده

در مرحله دوم از مراحل بيان شده در فرايند كنترل كيفيت و بسته بندي DVD ، مي بايست ضخامت DVD ها در 3 نقطه اندازه گيري شده و ميانگين آنها محاسبه گردد و در صورتيكه اين مقدار اندازه گيري شده در رنج مجاز باشد محصول در رده محصولات تاييد شده و در غير اين صورت در رده محصولات مردود ذخيره سازي گردد .

بدين منظور مي توانيم دو فانكشن بلوك AvgValue- ThresholdCheck و ActuatorControl را تركيب نموده و يك فانكشن بلوك كلي مشتمل براين دو را ايجاد نماييم .

عنوان اين فانكشن بلوك كلي را DVD- ThickSelectControl انتخاب مي نماييم و در زير چگونگي تعريف آن را توضيح خواهيم داد.

جدول متغير هاي ورودي

Name	Data Type	AT	Initial Value	Retained	Comment
EN	BOOL		FALSE		Controls execution of the Function Block.
LSright	BOOL		FALSE		Limit switch for cylinder right direction
LSleft	BOOL		FALSE		Limit switch for cylinder left direction
Measure1	REAL		0.0		Measurement result 1 of DVD thickness (mm)
Measure2	REAL		0.0		Measurement result 2 of DVD thickness (mm)
Measure3	REAL		0.0		Measurement result 3 of DVD thickness (mm)

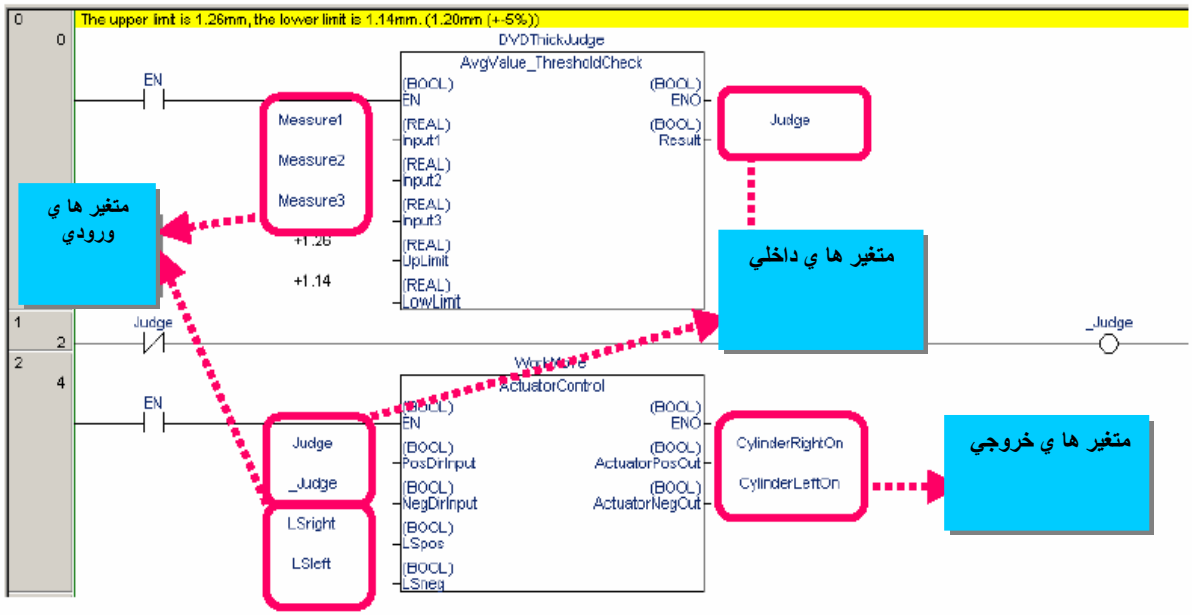
جدول متغيرهاي خروجي

Name	Data Type	AT	Initial Value	Retained	Comment
ENO	BOOL		FALSE		Indicates successful execution of the Function Block ...
CylinderRightOn	BOOL		FALSE		Output for sylinder right direction
CylinderLeftOn	BOOL		FALSE		Output for sylinder left direction

جدول متغيرهاي داخلي

		AT	Initial Value	Retained	Con
WorkMove	FB [ActuatorControl]				
DVDThickJudge	FB [AvgValue_ThresholdCheck]				
Judge	BOOL		FALSE		
_Judge	BOOL		FALSE		

This FB has its specific name and variable names that include "DVD" or "Cylinder" because it is specifically created for application.



يك فانكشن بلوك را مي توان در فانكشن بلوك ديگر فراخواني نمود كه اين كار اصطلاحاً ، آشيانه سازي در آشيانه ، ناميده مي شود و بدين منظور مي بايست فانكشن بلوك مورد نظر تان را در زمره متغيرهاي داخلي و از نوع Function Block تعريف نماييد

2_4_4. تعريف يك فانكشن بلوك براي شمارش دفعات انتخابها به منظور محاسبه طول عمر ليميت سوئيچ ها

در مرحله پنجم از مراحل بيان شده برنامه كنترلي مربوط به بسته بندي DVD ها ، جعبه ها به دو نوع تقسيم شده و در اين حال يك شمارنده مي بايست متناوباً تعداد دفعات انتخاب مابين اين دو نوع را به منظور محاسبه طول عمر ليميت سوييچ مجاور فعال كنندة انتخاب انواع ، بشمارد .بدين منظور مي توان از ON و OFF شدن ليميت سوئيچ ، به عنوان ورودي شمارنده كنترلي عملكرد فعال كننده استفاده نمود .براي براوردن اين نياز فانكشن بلوكي تحت عنوان " WorkMoveControl- LSONcount " را با جزئياتي كه در زير آمده تعريف خواهيم نمود .

متغير هاي ورودي

Name	Data Type	AT	Initial Value	Retained	Comment
EN	BOOL		FALSE		Controls execution of the Function Block.
RightDirInput	BOOL		FALSE		Condition to move actuator to right direction
LeftDirInput	BOOL		FALSE		Condition to move actuator to left direction
LSright	BOOL		FALSE		Limit switch for acutuator right direction
LSleft	BOOL		FALSE		Limit switch for acutuator left direction
Reset	BOOL		FALSE		Resets number of times for opening - closing li...

متغير هاى خروجى

Data Type	AT	Initial Value	Retained	Comment
BOOL		FALSE		Indicates successful execution of the Functio...
BOOL		FALSE		Output for actuator right direction
BOOL		FALSE		Output for actuator left direction
LINT		0		

متغير هاى داخلى

Name	Data Type	AT	Initial Value	Retained	Comment
PrevCycleLS	BOOL		FALSE		
WorkMove	FB [ActuatorControl]				

```
(* Work move control and count of number of times open - close of limit swvitch *)
 (* كنترل عملكرد فعال كننده هاي انتخابگر و نيز شمارش دفعات باز و بسته شدن ليميت سوئيچ ها *)
 (* Reset نمودن تعداد دفعات باز و بسته شدن ليميت سوئيچ *)
    PrevCycleLS := FALSE;
END_IF;

 (* فراخواني فانكشن بلوك ActuatorControl در اين فانكشن بلوك تحت عنوان WorkMove *)

 (* شمارش تعداد دفعات باز وبسته شدن ليميت سوئيچ *)
    LS_ONnumber := LS_ONnumber+1;
END_IF;
PrevCycleLS := LSright;   (* كپي نمودن وضعيت LSright براي مقايسه در اجراي بعدي *)
```

فراخوانى يك فانكشن بلوك نوشته شده به زبان نردبانى در برنامه نوشته شده به زبان ST

چگونگي فراخواني فانكشن بلوكها در برنامه نويسي به زبان ST :

در ابتدا عنوان فانكشن بلوك فراخواني شده و معادل متغير فانكشن بلوكي كه در برنامه ST به ازا آن تعريف شده است قرار خواهد گرفت و سپس ورودي / خروجي هاي فانكشن بلوك فراخواني شده و معادل متغير هاي ورودي / خروجي تعريف شده در زبان ST قرار داده خواهد شد.

به طور مثال :

فانكشن بلوك فراخواني شده : MyFB — متغير فانكشن بلوك معادل در برنامه ST: MyInstance

ورودیهاي فانكشن بلوك: Input1, Input2 — متغيرهاي ورودي معادل در برنامه ST: STInput1 , STInput2

خروجيهاي فانكشن بلوك: Output1 , Output2 — متغيرهاي ورودي معادل در برنامه ST: STOutput1 , STOutput2

در مثال فوق فراخواني فانكشن بلوك نمونه در برنامه ST مي بايست با فرمت زير صورت پذيرد :

MyInstance(Input1:=STInput1, Input2:=STInput2, Output1=> STOutput1 , Output2 =>STOutput2);

زمانيكه تمامي متغيرهاي ورودي / خروجي مورد استفاده قرار گيرند ديگر نيازي به استفاده از فرمت بالا و تعريف يكايك متغيرهاي معادل فانكشن بلوك و متغيرهاي برنامه ST نمي باشد و مي توانيد به فرمت زير عمل نماييد :

MyInstance(STInput1 , STInput2, STOutput1 , STOutput2);

در صورت تعريف متغيرهاي ورودي و خروجي معادل ، مي توانيد تنها قسمتي از متغيرهاي ورودي / خروجي را تعريف نماييد :

MyInstance(Input1:=STInput1, Output2 =>STOutput2);

4_5. توضيح كل برنامه

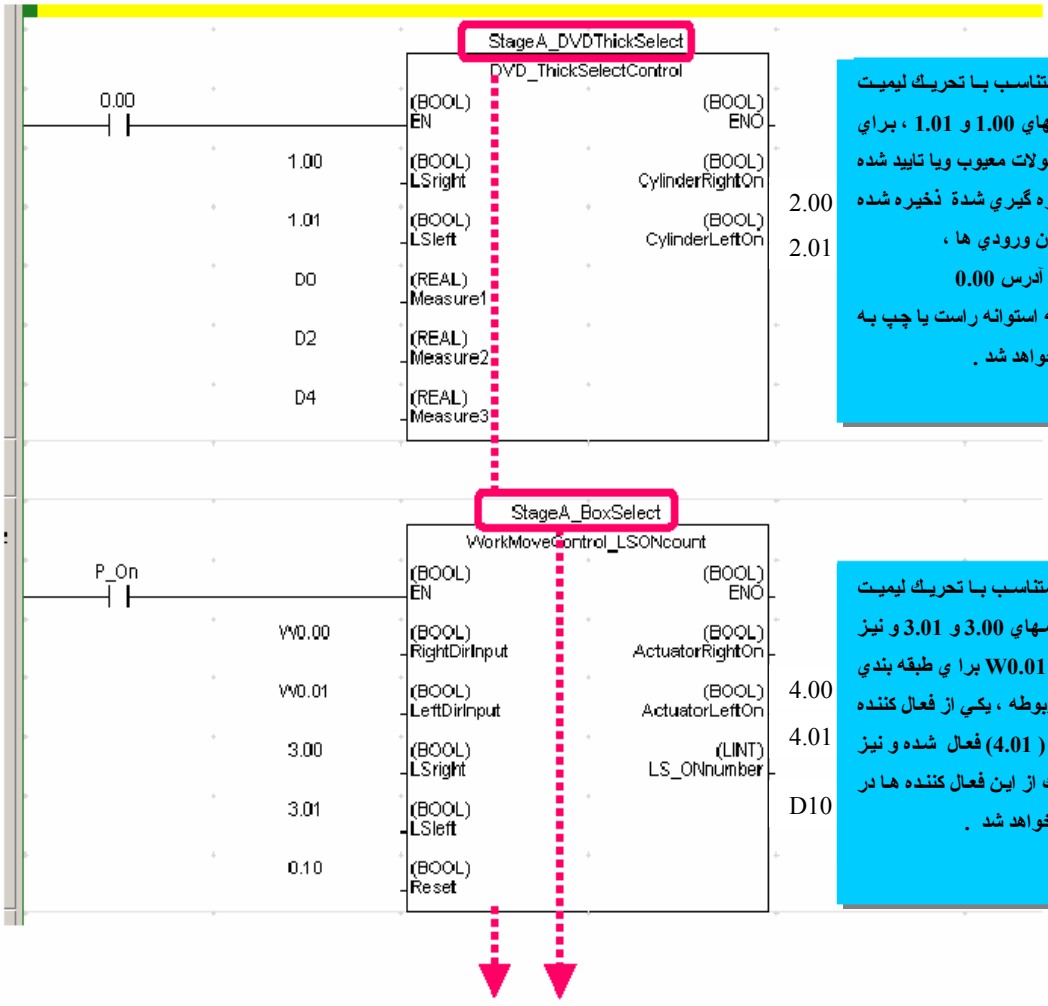
به منظور آنكه فانكشن بلوكهايي كه تا كنون تعريف نموده ايد، بتوانند در غالب يك برنامه عملكرد داشته باشند مي بايست يك برنامه اصلي براي فراخواني و مجتمع سازي اين فانكشن بلوكها تعريف نماييد.

كه چگونگي اين امر در شكلهاي زير آمده است

[Global Variables]

Name	Data Type	Address / Value	Rack Location	Usage
StageA_BoxSelect	FB [WorkMoveControl_LSONcount]	N/A [Auto]		
StageA_DVDThickSelect	FB [DVD_ThickSelectControl]	N/A [Auto]		

* Other instance variables than those to use FB are omitted.



در اين فانكشن بلوك كلي ،متناسب با تحريك ليميت سوييچ راست يا چپ به آدرسهاي 1.00 و 1.01 ، براي تقسيم بندي DVD ها به محصولات معيوب ويا تاييد شده و نيز مقدار سه ضخامت اندازه گيري شدة ذخيره شده در آدرسهاي D0- D5 به عنوان ورودي ها ،

با فعال شدن كنتاكت فعالساز به آدرس 0.00

يكي از خروجي هاي مربوط به استوانه راست يا چپ به آدرسهاي 2.00 و 2.01 فعال خواهد شد .

در اين فانكشن بلوك كلي ، متناسب با تحريك ليميت سوييچ راست يا چپ به آدرسهاي 3.00 و 3.01 و نيز عملكرد ورودي W0.00 و يا W0.01 برا ي طبقه بندي بسته ها به يكي از دو نوع مربوطه ، يكي از فعال كننده هاي راست (4.00) و يا چپ (4.01) فعال شده و نيز تعداد دفعات فعال شدن هر يك از اين فعال كننده ها در آدرسهاي D10- D13 ذخيره خواهد شد .

علت نامگذاري علامت گرافيكي معادل فانكشن بلوك در برنامه تحت عنوان Stage A :

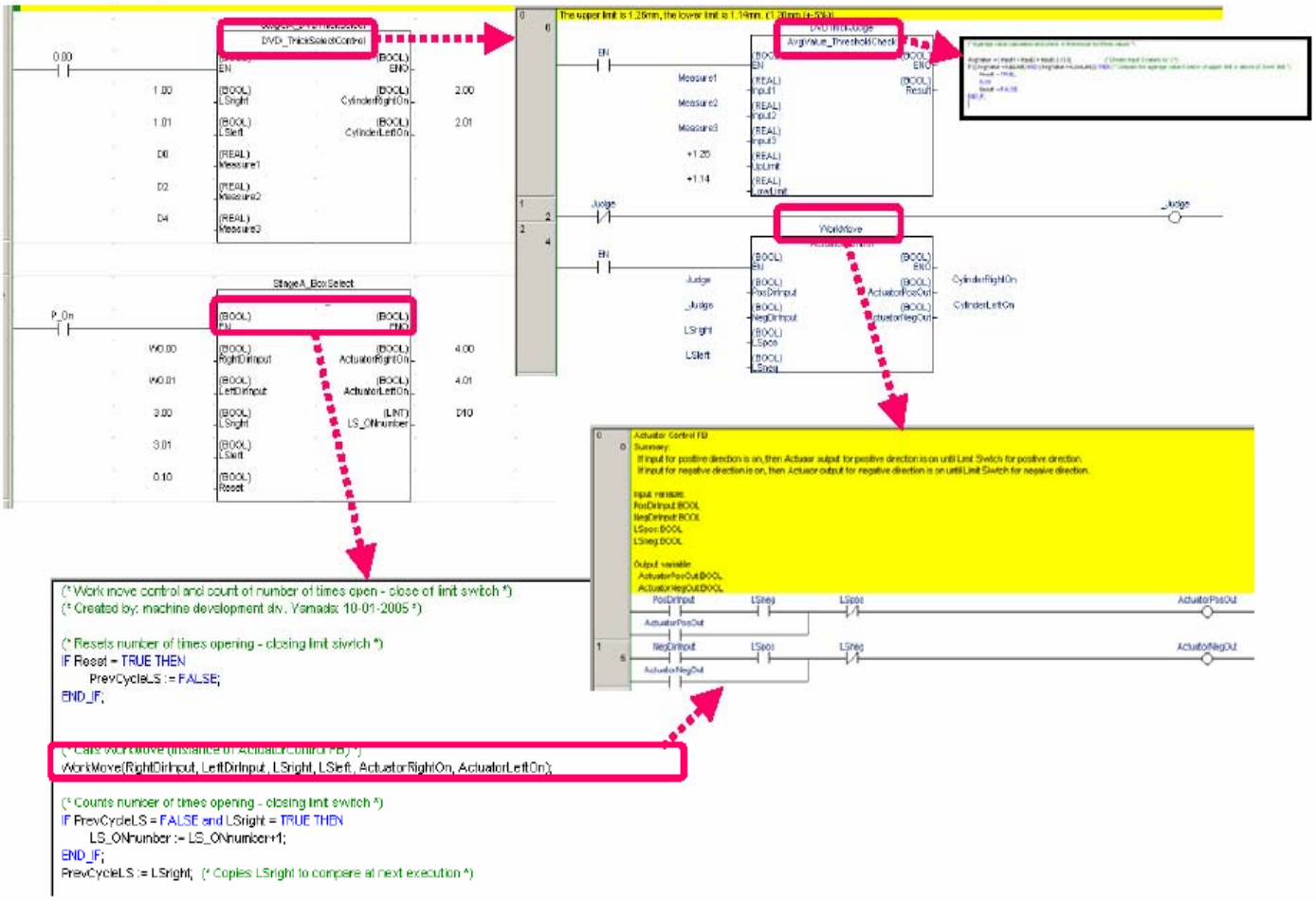
به منظور كنترل يك فرايند مشابه جديد ، كافيست فانكشن بلوك ديگري تحت عنوان Stage B را فراخواني نموده و متغيرهايش را مقدار دهي نماييد .

در اين صورت نيازي به تعريف و ثبت يك فانكشن بلوك جديد نمي باشد .

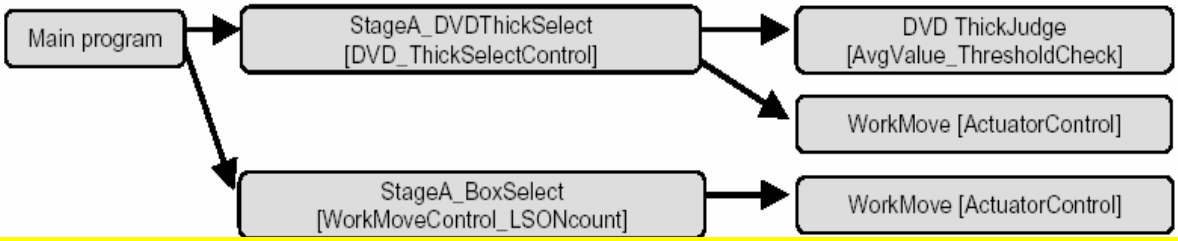
لذا مي توان بعد از تعريف يك فانكشن بلوك در صورت نياز، آنرا بيش از يك بار ، به منظور كنترل فرايندهاي مختلف ، فراخواني نمود .

1_6_4 ساختار كلي برنامه :

در اين قسمت برنا مه ساختار يافته با نمايش جزييات هر بخش نشان داده شده است . همانطور كه ملاحظه مي نماييد اين امكان وجود دارد كه در يك فانكشن بلوك ، فانكشن بلوك هاي ديگر فراخواني و در آنها نيز به نوبه خود فانكشن بلوكهايي فراخواني شده باشد و هر قسمت به زير قسمتهايي تقسيم مي گردد .



ساختار كلي برنامه و طريقه فراخواني فانكشن بلوكها و ز ير فانكشن هاي به صورت دياگرام زير مي باشد

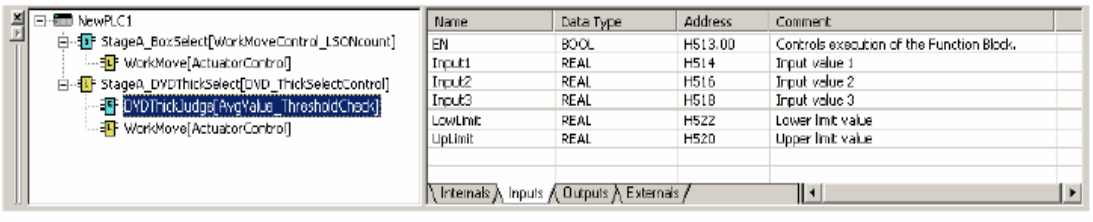


در برنامه ساختاريافته ، خصوصاً براي تغييردر فانكشن بلوكهاي سطح پايين تر ، دانستن ارتباطات مابين بلوكهاي مختلف فانكشن بلوكهاي فراخواني شده در يكديگر در زمان اجراي فرايند بسيار مهم بوده و براي خطايابي برنامه مي بايست اين سطوح و چگونگي ارتباطات آنها واضح گردد .

توصيه مي شود يك دياگرام قابل فهم از كل ساختار برنامه تهيه نماييد .

درنرم افزار CX- Programmer نسخه 6 اين امكان وجود دارد كه با فشردن همزمان كليدهاي Alt , 5 پنجره اي تحت عنوان FB Instance Viewer گشوده شود، كه در آن ساختار ساده تري از فانكشن بلوكهاي استفاده شده در برنامه با نمايش ساختار زير شاخه اي فانكشن بلوكهاي فراخواني شده در يكديگر نمايش داده مي شود .

و نيز با كليك بر روي هر فانكشن بلوك پنجره متغيرهاي مربوط به آن گشوده خواهد شد .



5. فراخواني فانكشن بلوك

اين بخش چگونگي فراخواني فانكشن بلوك در يك برنامه واقعي و خطايي آن را توضيح مي دهد .

يك برنامه جديد ايجاد نموده و فانكشن بلوك " ActuatorControl " را كه چگونگي تعريف آن را در بخش 5- 4 توضيح داديم درآن فراخواني نماييد .

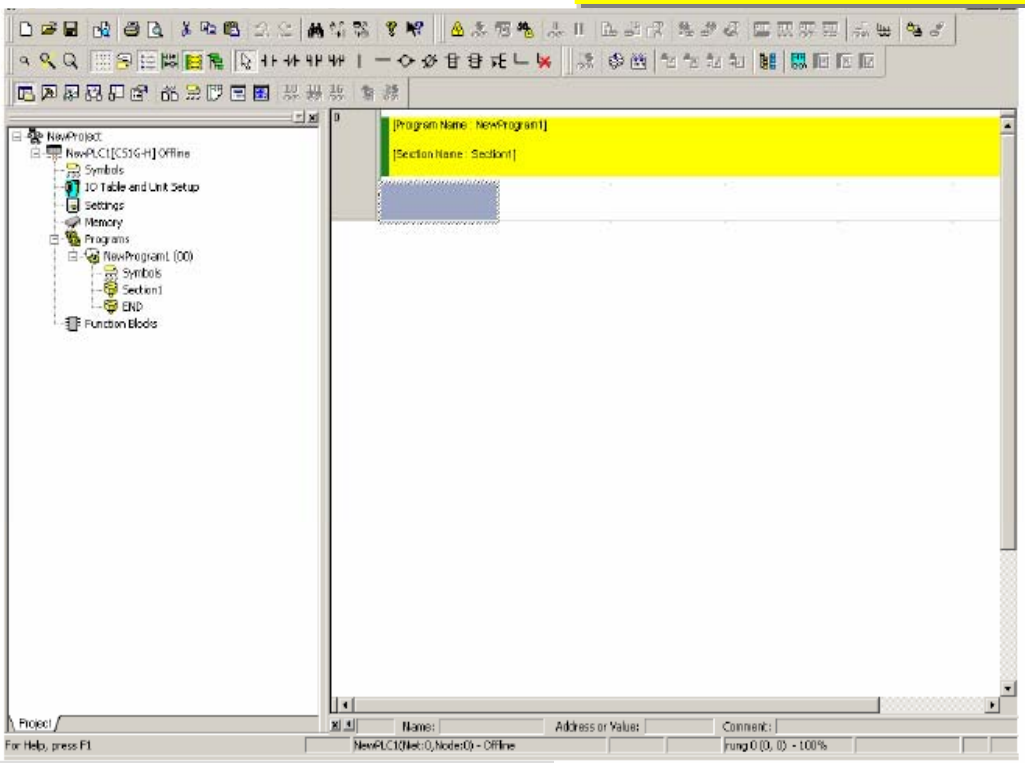
5_1. ايجاد يك برنامه جديد و انجام تنظيمات مربوط به انتخاب مدل PLC و نوع CPU آن

Refer to page 2-3 and create a new project.

چگونگي ايجاد يك برنامه جديد قبلاً در بخش 2- 3 توضيح داده شد .

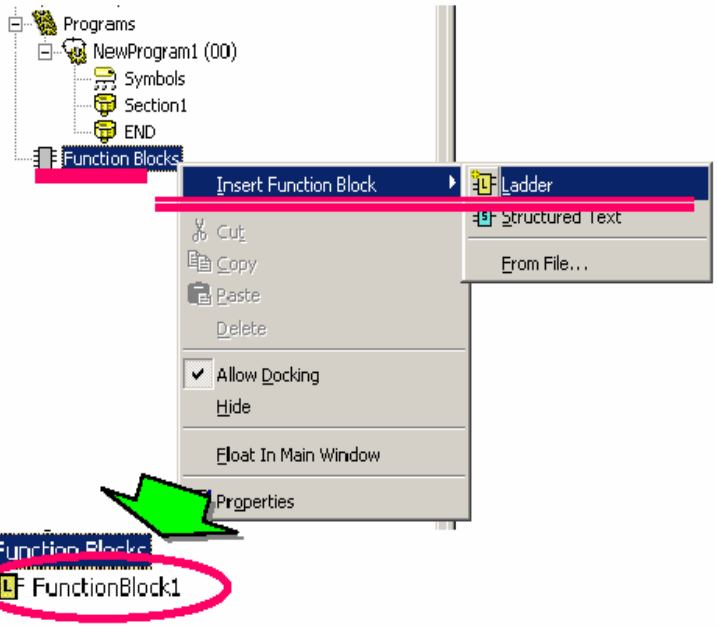
در صورتيكه مي خواهيد از فانكشن بلوك در برنامه تان استفاده نماييد حتماً مي بايست PLC كه مي خواهيد به آن متصل شويد يكي از انواع زير باشد:

CS1G- H , CS1H- H , CJ1G- H , CJ1H- H , CJ1M



5_2. تعريف يك فانكشن بلوك جديد به زبان نردباني

Create Ladder definition FB.



در پنجـــره WorkSpace ، بـــر روي علامـت گرافيكـي تعريـف فانكـشن بلـوك

،

كليك راست نمـوده وگزينـه هـاي زيـر را انتخاب نماييد

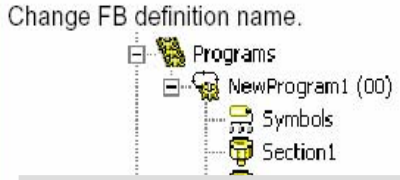
Insert Function Block

Ladder

Now new FB is created.

5_3. ايجاد برنامه نردباني فانكشن بلوك تعريف شده

مكـان نمـا را بـر روي علامـت گرافيكـي مربـوط بـه تعريـف فانكـشن بلـوك ، قـرار دهيـد سپس كليك راست نموده و گزينه Rename را انتخاب نماييد .نام مورد نظر (ActuatorControl) را وارد كرده وسپس كليد Enter را بفشاريد

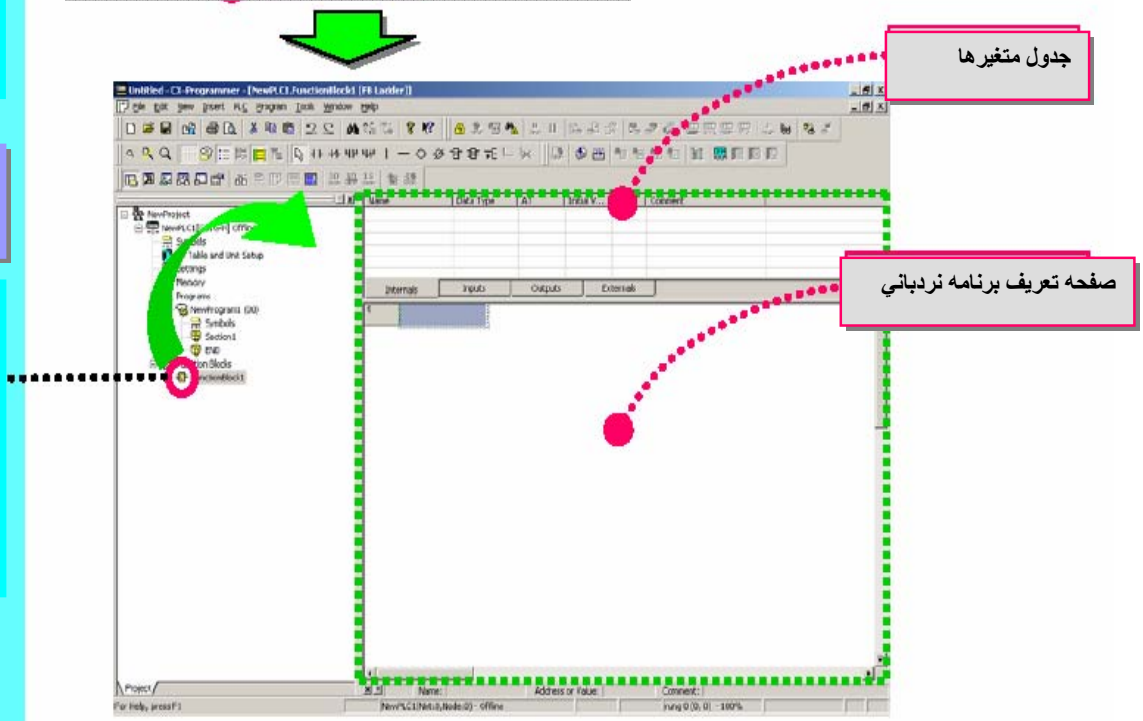


توجه داشته باشيد كه نمي توانيد نام فانكشن بلوك را با " _ " آغاز نماييد

د يك فانكشن بلوك به زبان نردباني

محيط برنامـه نويـسي فانكـشن بلوك را بگشاييد .

مكـان نمـا را بـر روي علامـت گرافيكي تعريف فانكشن بلوك قرار دهيد ، و دوبـار كليـك نماييـد تـا محـيط برنامـه نويـسي فانكـشن بلـوك گشوده شود .



در جدول متغير ها متغيرهاي مورد نيازتان را تعريف نماييد .
تمامي متغير هاي فانكشن بلوك " ActuatorControl " توضيح داده شده در بخش 5- 4 را مي بايست تعريف نماييد .

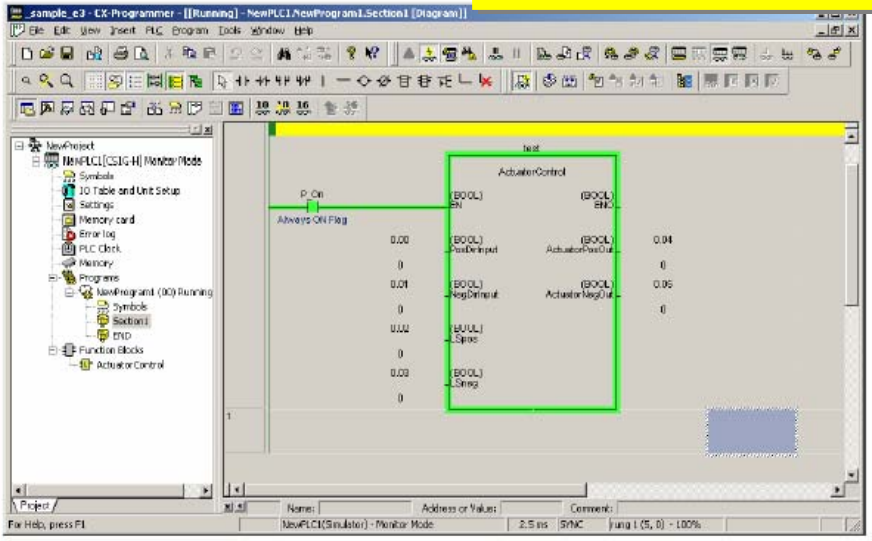
ترتيب متغيرها ، همسان با ترتيب تعريف متغيرها در جدولهاي مربوطه مي باشد و در صورت نياز به تغيير ترتيب متغير ها مي بايست در جدول تعريف متغيرها ، متغير مورد نظرتان را با موس گرفته و بكشيد و در موقعيت جديد قرار دهيد تا ترتيب مورد نظرتان ايجاد شود

وارد قسمت مربوط به برنامه نويسي شده و برنامه نردباني مورد نياز را وارد نماييد .

4_5. انتقال برنامه

از طريق سيمولاتور به كنترلر مجازي متصل شده و برنامه را به ان منتقل نماييد .

براي آشنايي با چگونگي اتصال به كنترلر و انتقال برنامه به آن به بخش 2- 10 مراجعه نماييد



5_5. چك كردن صحت عملكرد- 1 :

در حالت مانيتور، مقادير متغيرها ي FB فراخواني شده در برنامه اصلي را تنظيم نموده ، سپس صحت عملكرد اين فانكشن بلوك را چك نماييد.

ابتدا چگونگي عملكرد و وضعيت متغيرهاي ورودي / خروجي را درعلامت گرافيكي معادل فانكشن بلوك فراخواني شده در برنامه اصلي چك نماييد

مكان نما را بر روي فانكشن بلوك فراخواني شده در برنامه اصلي قرار دهيد و يكي از دو عمليات زير را انجام دهيد تا به برنامه داخلي اين فانكشن بلوك منتقل شويد

1. بر روي آن دوبار كليك نماييد
2. يا انكه بر روي علامت گرافيكي

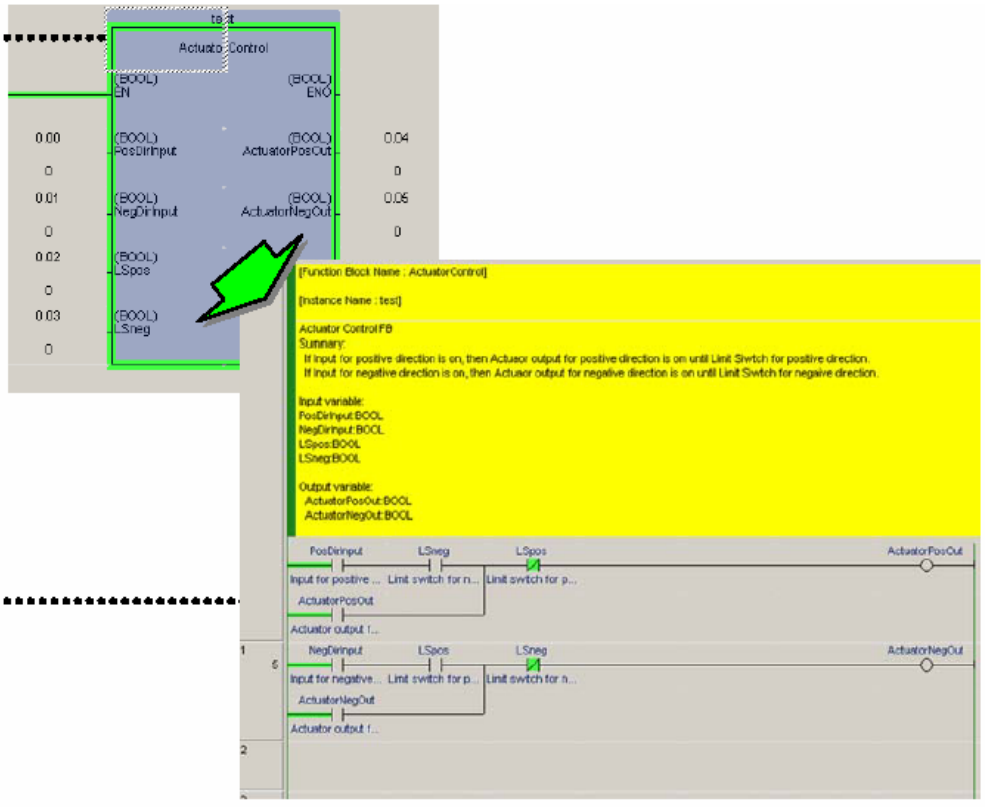


، در نوار ابزار بالا يي نرم افزار ، كليك نماييد

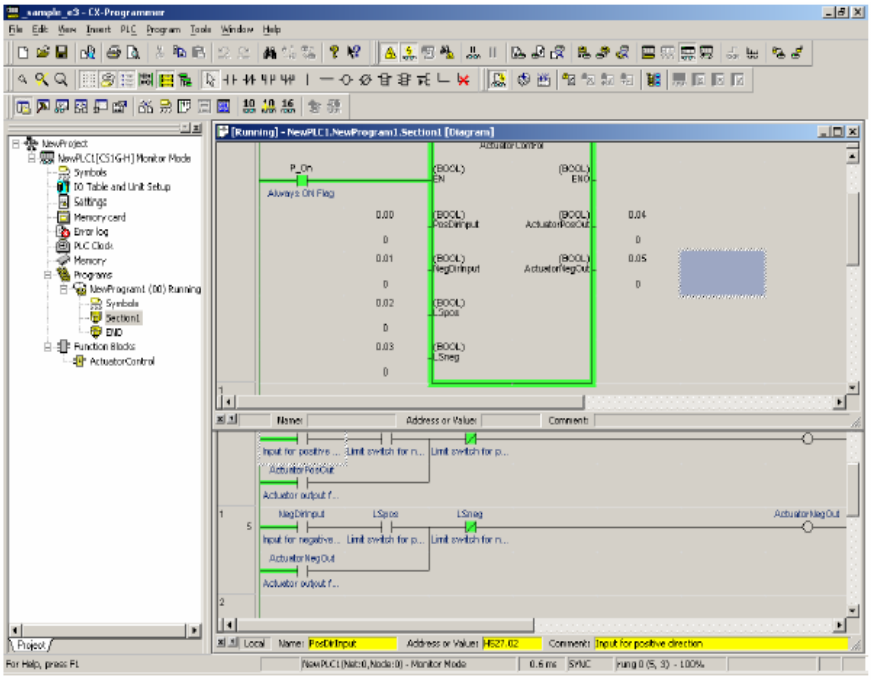


در اين حال برنامه مرجع فانكشن بلوك در حاليكه متغيرهايش مقدار دهي شده اند ، مانيتور شده .

و مي توانيد چگونگي اجرا برنامه آن را دنبا ل نماييد .



صفحات مربوط به برنامه اصلي و نيز برنامه داخلي فانكشن بلوك را به طور همزمان بگشاييد و مراحل عملكرد فانكشن بلوك و تاثير آن بر متغيرهاي ورودي / خروجي را به طور همزمان مشاهده نماييد .



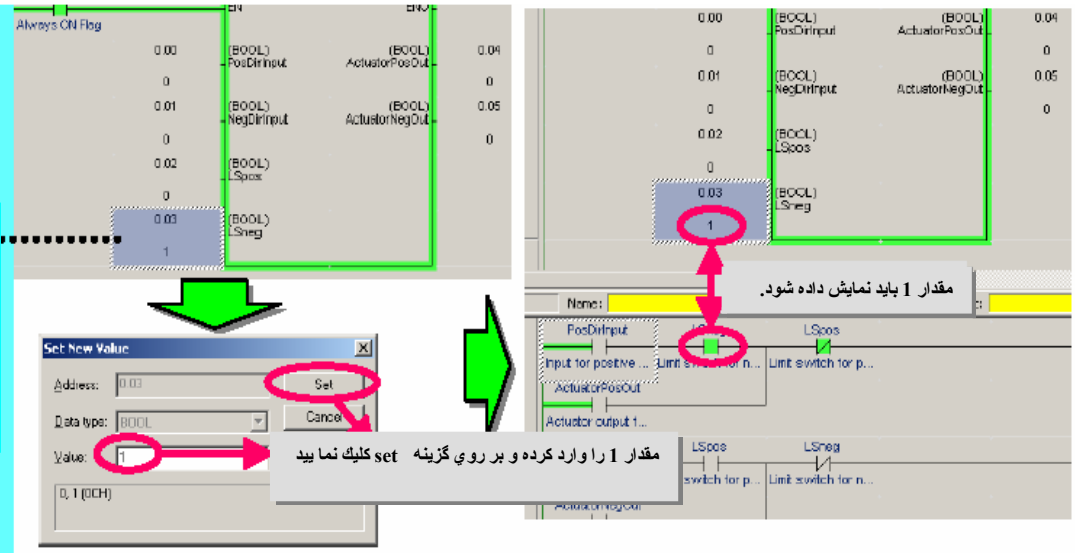
6_5. چك كردن صحت عملكرد- 2 :

پارمترهاي زير را مقداردهي نموده و بررسي نماييد كه آيا خروجي هاي مطلوب ايجاد مي شود ؟

در اين مثال تنها حالت 1 نمايش داده شده است ولي تمامي حالتها مي بايست بررسي شوند .

1. حالت اوليه :با روشن نمودن 0.03 ،خروجي هاي 0.04 و 0.05 مي بايست خاموش شوند
2. عملكرد فعال كننده مربوط به حركت رو به جلو- 1 : ورودي 0.00 را روشن نماييد خروجي 0.04 مي بايست روشن گردد
3. عملكرد فعال كننده مربوط به حركت رو به جلو- 2 : ورودي 0.03 را روشن نماييد خروجي 0.04 مي بايست روشن و خروجي 0.05 مي بايست خاموش شود.
4. عملكرد فعال كننده مربوط به حركت رو به جلو- 3 : ورودي 0.02 را روشن نماييد خروجي 0.04 مي بايست خاموش و خروجي 0.05 مي بايست خاموش شود.

در تمامي موارد فوق الذكر ، كنترلر بايد در وضعيتي باشد كه قابليت نمايش مقدار متغيرها را داشته باشد



مكان نما را بر روي متغير با آدرس 0.03 قرار داده وكليد Enter را بفشاريد تا پنجره مربوط به مقدار دهي اين متغير گشوده شود

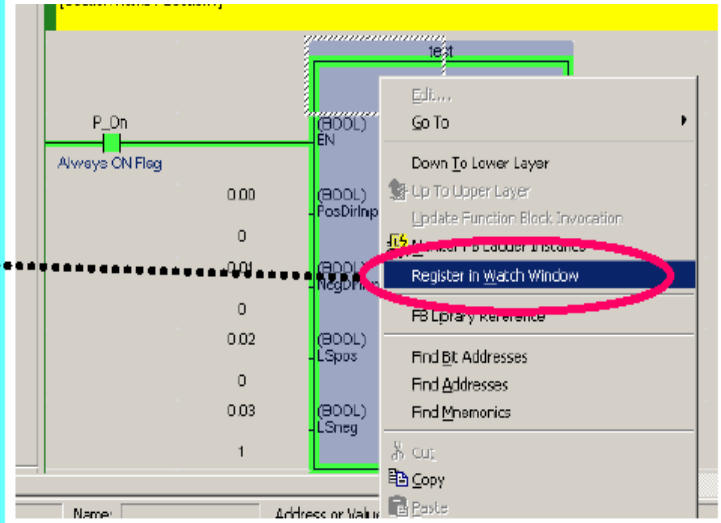
5_7. فراخواني و خطايابي مابقي فانكشن بلوكها

فراخواني و خطايا بي فانكشن بلوك " ActuatorControl " توضيح داده شد .
ما بقي فانكشن بلوكها نيز به همين ترتيب فراخواني و خطا يابي مي شوند.

5_8. نمايش يك دسته از متغيرها به طور همزمان در پنجره Watch Window

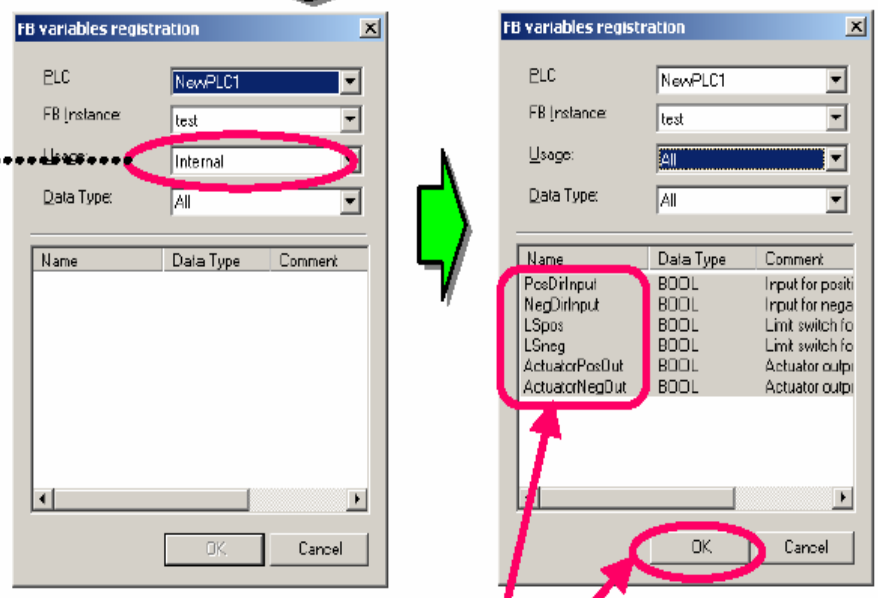
براي خطايابي مي توانيد، به جاي باز كردن محيط برنامه نويسي فانكشن بلوك، از نمايش همزمان وضعيت تمامي يا برخي از آدرسهاي مر بوط به متغيرهاي فانكشن بلوك در پنجره Watch window ، استفاده نماييد

بدين منظور بر روي المان فانكشن بلوك استفاده شده در برنامه، كليك راست نموده و از منو گشوده شده گزينه Register In Watch Window را انتخاب نماييد .
تا پنجره انتخاب متغير هايي كه مي خواهيد به طور همزمان نمايش داده شوند گشوده شود



در پنجره گشوده شده در قسمت Usage و Data Type متغير هايي را كه مي خواهيد وضعيتشان را به صورت همزمان در پنجره Watch Window مشاهده كنيد انتخاب نماييد.

در صورت تمايل به مشاهده همزمان وضعيت تمامي متغير هاي فانكشن بلوك گزينه ALL را انتخاب نماييد .



متغير هاي مورد نظر را انتخاب كرده و OK كنيد.

9-5 . مراحل شبيه سازي برنامه

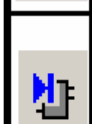
با شبيه سازي فانكشن بلوك مي توانيد چگونگي اجرا مراحل مختلف برنامه را چك نماييد و در ضمن هر زمان كه بخواهيد مي توانيد ، اجرا را متوقف نماييد .

البته قابليت شبيه سازي و كنترل مراحل اجراي فانكشن بلوك ، مختص نرم افزارCX-One نسخه 1.1 به بالا (CX-Programmer نسخه 6.1 و نيز CX-Simulator نسخه 1.6 به بالا) مي باشد

1-9-5 . توضيح كليدهاي مر بوط به محيط شبيه سازي

نوار ابزار زير مخصوص محيط شبيه سازي بوده ودر اين محيط استفاده مي شوند .
عملكرد هر كليد در زير توضيح داده شده است .

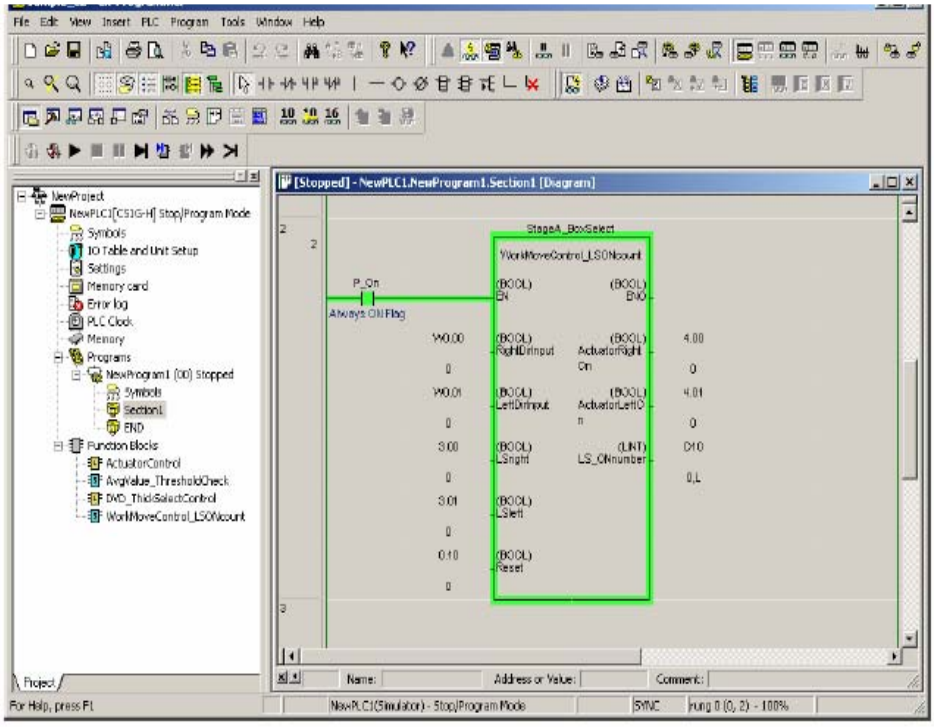


	ايجاد و يا حذف يك نقطه انفصال در برنامه	**مكاني را كه مي خواهيد برنامه در آن نقطه متوقف شود انتخاب كرده و سپس بر روي اين نماد كليك نماييد .تا يك دايره قرمز رنگ در بالاي قسمت مورد نظر شما نمايش داده شود و اجراي برنامه متوقف گردد**
	**حذف تمامي نقاط انفصال برنامه**	**با كليك بر روي اين نماد تمام نقاط انفصال برنامه ، حذف شده و دايره هاي قرمز رنگ موجود در برنامه ازبين خواهند رفت .**
	**تغيير حالت عملكرد PLC به حالت Run و يا Monitor**	**با فشردن اين كليد برنامه از حالت RUN ، وارد حالت مانيتورشده و برنامه كاربر اجرا خواهد شد .**
	**توقف عملكرد برنامه (تغيير حالت عملكرد به حالت Program)**	**با فشردن اين كليد برنامه از حالت RUN ، وارد حالت Progarm شده و برنامه كاربر متوقف خواهد شد .در اين حال براي اجراي مجدد برنامه بايدبر روي كليد قبل كليك نماييد در اين حال برنامه مجدداً اجرا خواهد شد ولي شرايط قبل از توقف برنامه از بين خواهد رفت .**
	**وقفه**	**اجراي برنامه كاربر در محلي كه مكان نما وجود دارد متوقف خواهد شد . بعد از فشردن مجدد اين كليد برنامه كار كرد خود را ادامه خواهد داد ودر اين حال شرايط قبل از وقفه از بين نخواهد رفت .**
	**اجراي مرحله به مرحله برنامه**	**با هر بار فشرد اين نماد يك مرحله از برنامه كه در مورد برنامه هاي نردباني يك دستور و در مورد برنامه هاي ST يك خط مي باشد ، اجرا خواهد شد .و با استفاده از اين نماد مي توانيد برنامه را مرحله به مرحله اجرا نماييد ولي مراحل داخلي فانكشن بلوكها و يا زير برنامه هاي فراخواني شده در برنامه هاي ديگر نمايش داده نخواهد شد**
	**ورود به زير برنامه هاي فراخواني شده در برنامه هاي ديگر**	**با هر بار فشردن اين نماد يك مرحله از برنامه كاربر اجرا خواهد شد و در مواردي كه در برنامه فانكشن بلوكي فراخواني شده باشد و يا آنكه در داخل فانكشن بلوك ، فانشكن بلوكهاي ديگري فراخواني شده باشدو زمانيكه برروي قسمت مربوط به فراخواني زير برنامه ها باشيد ، اين نماد را بفشاريد ، وارد زير برنامه فراخواني شده خواهيد شد و جزئيات ان نمايش داده خواهد شد .**
	**خروج از زير برنامه هاي فراخواني شده در برنامه هاي ديگر**	**با هر بار فشردن اين نماد يك مرحله از برنامه كاربر اجرا خواهد شد و در مواردي كه در برنامه فراخواني شده باشد و يا آنكه در داخل فانكشن بلوك ، فانكشن بلوكهاي ديگري فراخواني شده باشدو زمانيكه برروي قسمت مربوط به فراخواني زير برنامه ها باشيد ، اين نماد را بفشاريد ، از زير برنامه خارج و وارد برنامه اصلي خواهيد شد**
	**اجراي مرحله به مرحله برنامه به صورت پيوسته**	**با فشردن اين نماد يك مرحله از برنامه اجرا خواهد شد اما بعد از پايان آن مرحله، مرحله بعدي پس از يك تاخير زماني مشخص به صورت اتوماتيك اجرا خواهد شد و اين روند ادامه مي يابد تا كل مراحل برنامه ، يكي پس از ديگري اجرا شوند .**
	Scan Run	**برنامه كاربر را به ميزان يك سيكل اجرا خواهد كرد .**

2-9-5 . تنظيم نقطه شكست و مراحل اجرا

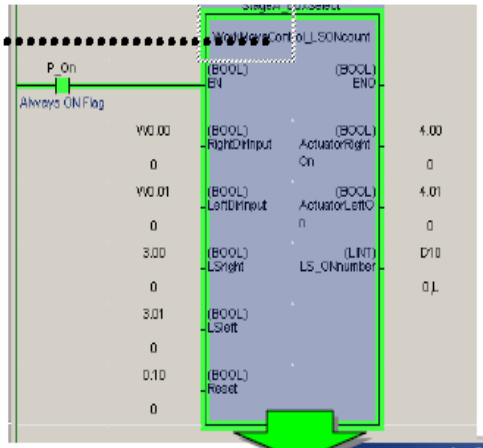
نكته : در صورتيكه نقطه شكست در برنامه داشته باشيد به هنگام رسيدن به نقطه شكست اجراي مر حله به مر حله متوقف خواهد شد و در غير اين صورت حلقه وار اين روند ادامه خواهد يافت

در اين قسمت چگونگي خطا يابي فانكشن بلوك " WorkMoveControl-LSONcount " را در محيط سيمولاتور به عنوان يك مثال توضيح خواهيم داد.



از حالت Run وارد حالت مانيتور شويد

بر روي فانكشن بلوك قرار گرفته و دوبار كليك نماييد و يا آنكه بر روي نماد  كليك نماييد .





مقادير لحظه اي متغيرهايي كه در فانكشن بلوك نوشته شده به زبان ST به كار رفته اند ، در يك پنجره در كنار برنامه اصلي نمايش داده مي شوند .



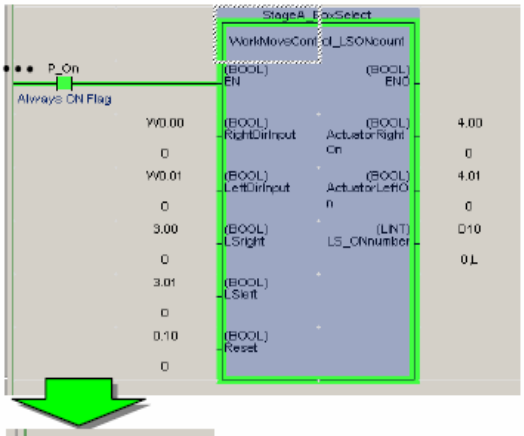
مقادير زير را در فانكشن بلوك فراخواني شده ، مقدار دهي نماييد و از صحت عملكرد فانكشن بلوك مطمئن شويد .

RightDirInput: ON
LeftDirInput: OFF
LSright: OFF
LSleft: ON
Reset: OFF

در اين حالت ، انتظار مي رود كه خروجي ها به صورت زير باشند

ActuatorRightOn: ON
ActuatorLeftOn: OFF
LS_ONnumber: 1

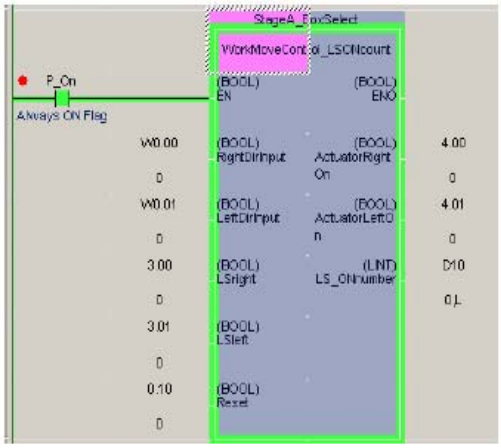
مكان نما را برروي ورودي سمت چپ فانكشن بلوك قرار داده و بر روي نماد كليك نمايد





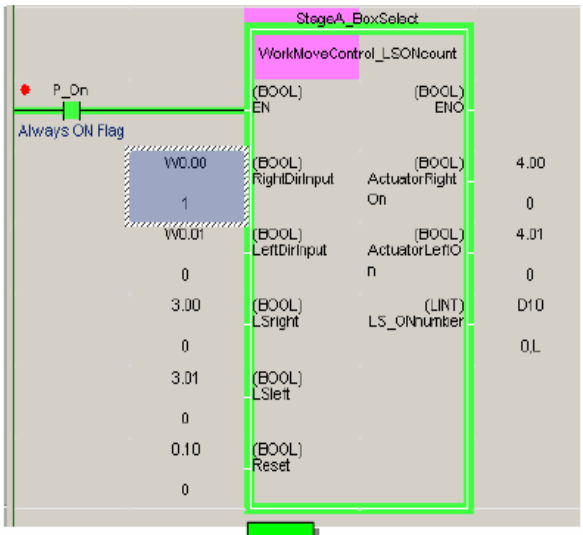
اجرا ي برنامه در نقطه شكست متوقف خواهد شد

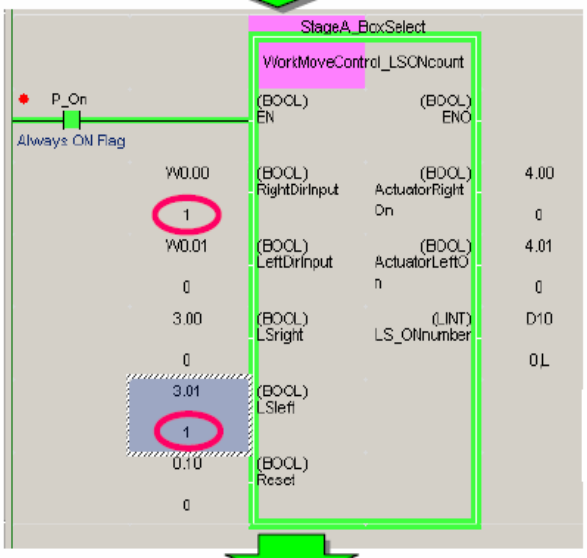
برروي نماد كليك نماييد .



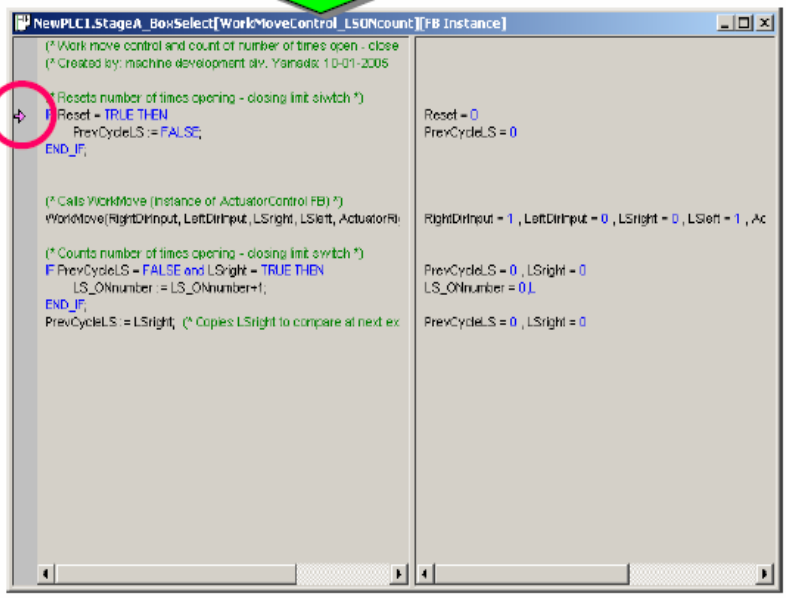
مرحله مر بوط به توقف برنامه اجرا شده و بر روي بعدي كه مر حله مر بوط به فراخواني فانكشن بلوك مي باشد متوقف خواهيد شد

در فانکشن بلوك فراخواني شده ، ورودي هاي "RightDirInput" و "LSleft" را به ترتيبي كه در زير نمايش داده شده است ، فعال نماييد





پارامترهاي مورد نياز تنظيم شد



در ابتدا اين مكان نما بر روي اولين خط برنامه فانكشن شده توسط شما به زبان ST قرار خواهد گرفت

The cursor moves to the first line position of the called ST program.



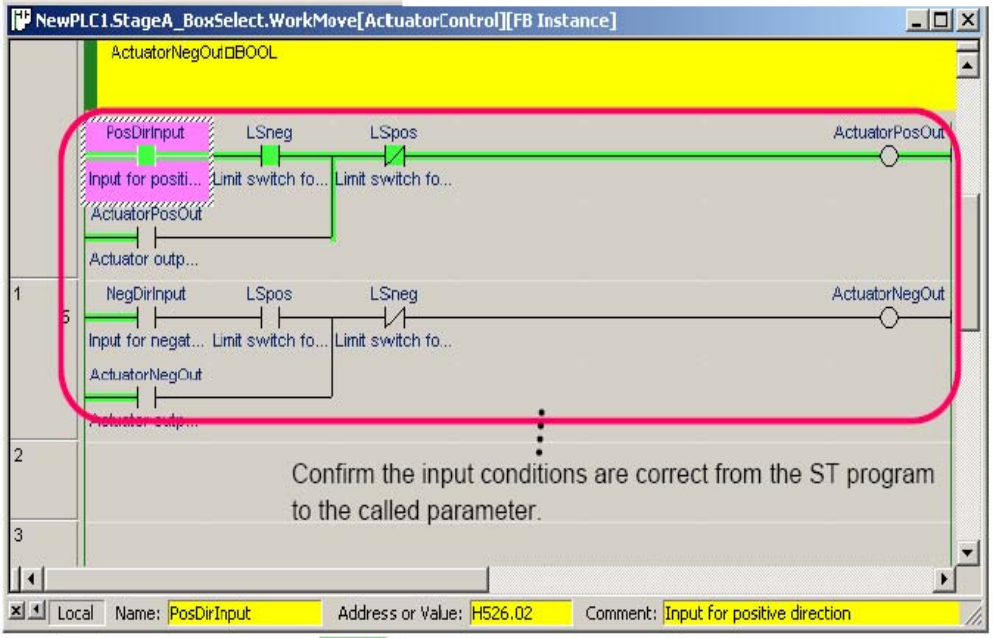
حال بر روي نماد  كليك نماييد تا در ضمن اجراي مرحله به مرحلة برنامه قابليت نمايش محتويات زير برنامه و يا فانكشن بلوكهاي فراخواني شده را نيز داشته باشيد .

بر روي هر مرحله از برنامة ST كه در حال اجرا باشد ، يك مكان نماي صورتي رنگ نمايش داده خواهد شد

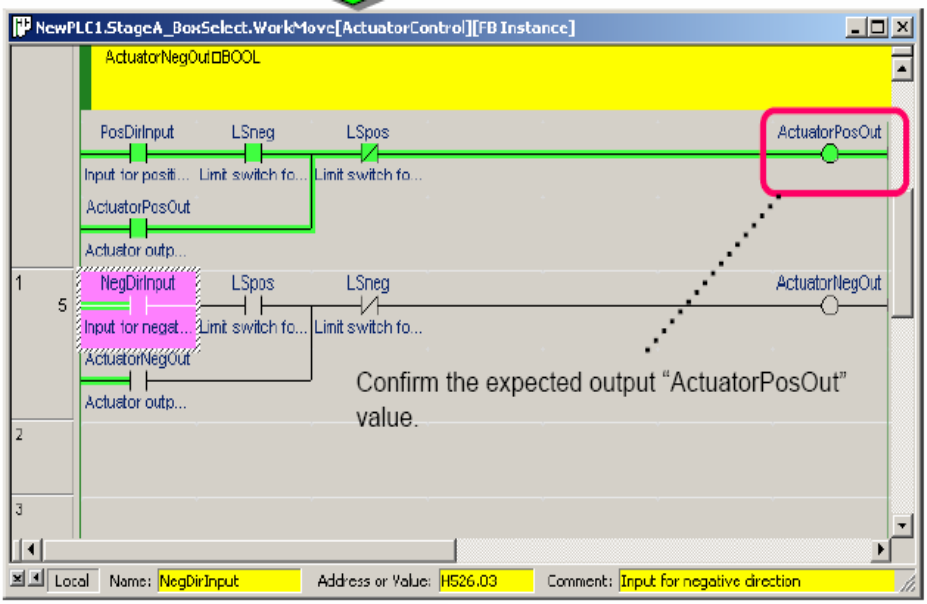
با دو بار فشردن نماد  دو مرحله بعدي برنامه ST (دو خط بعدي) اجرا شده و خطي از برنامه كه در آن فانكشن بلوك كنترل محركها را فراخواني نموده ايد اجرا خواهد شد و وارد اين زير برنامه خواهيد شد .



در اين مرحله از برنامه ST به فانكشن بلوك نردباني فراخواني شده در فانكشن بلوك اصلي ، منتقل خواهيد شد .

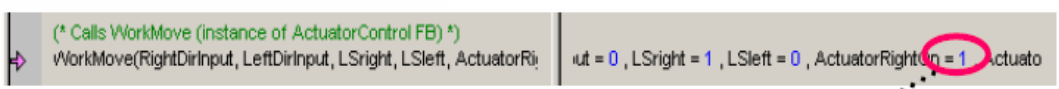


بعد از پنج بار فشردن نماد  و اجراي پنج مرحله از زير برنامه نردباني فراخواني شده ، محرك مورد نظر (ActuatorPosOut) فعال خواهد شد و شما با اطمينان از وصل بودن اين محرك مي توانيد به برنامه ST باز گشته و ادامة برنامه را مرحله به مرحله اجرا نماييد .



بر روي نماد  كليك نماييد تا از برنامه نردباني خارج شده و وارد برنامه ST شويد

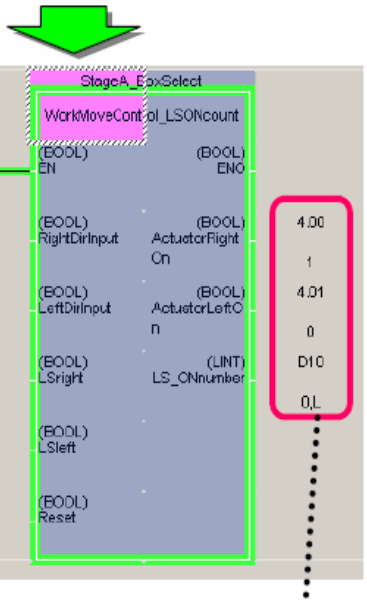
اطمينان از عملكرد كامل شده و مي توانيد به برنامه ST باز گرديد



مطمئن شويد كه نتيجة حاصل ازاجراي مراحل قبل ،به صورت صحيح به برنامه ST منتقل شده است



بـــر روي نمـــاد  كليـــك نماييـــد تـــا بـــه برنامـــة اصـــلي بازگرديد

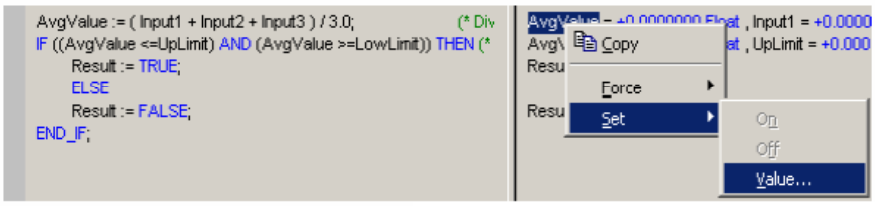


به برنامة نردباني اصلي منتقل شويد

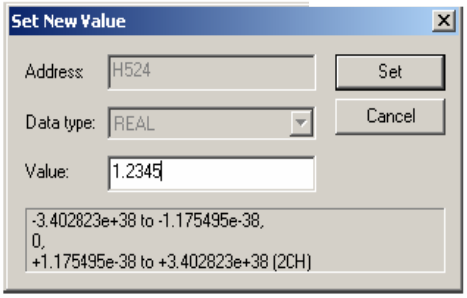
مطمئن شويد كه پارامتر خروجي به صورت صحيح در برنامه منعكس شده است

تذكر

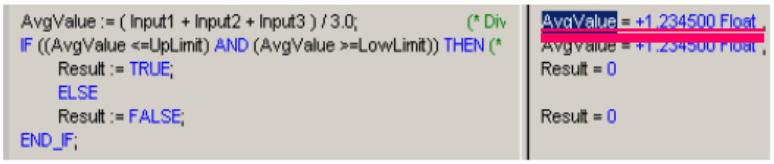
در برنامه ST براي تغيير مقادير لحظه اي پارامترها مي توانيد چنين عمل نماييد



پارامتري را كه مي خواهيد مقدارش را تغيير دهيد انتخاب كرده و بر روي آن كليك راست نماييد حال از منو گشوده شده گزينه Set و سپس Value را انتخاب نماييد .



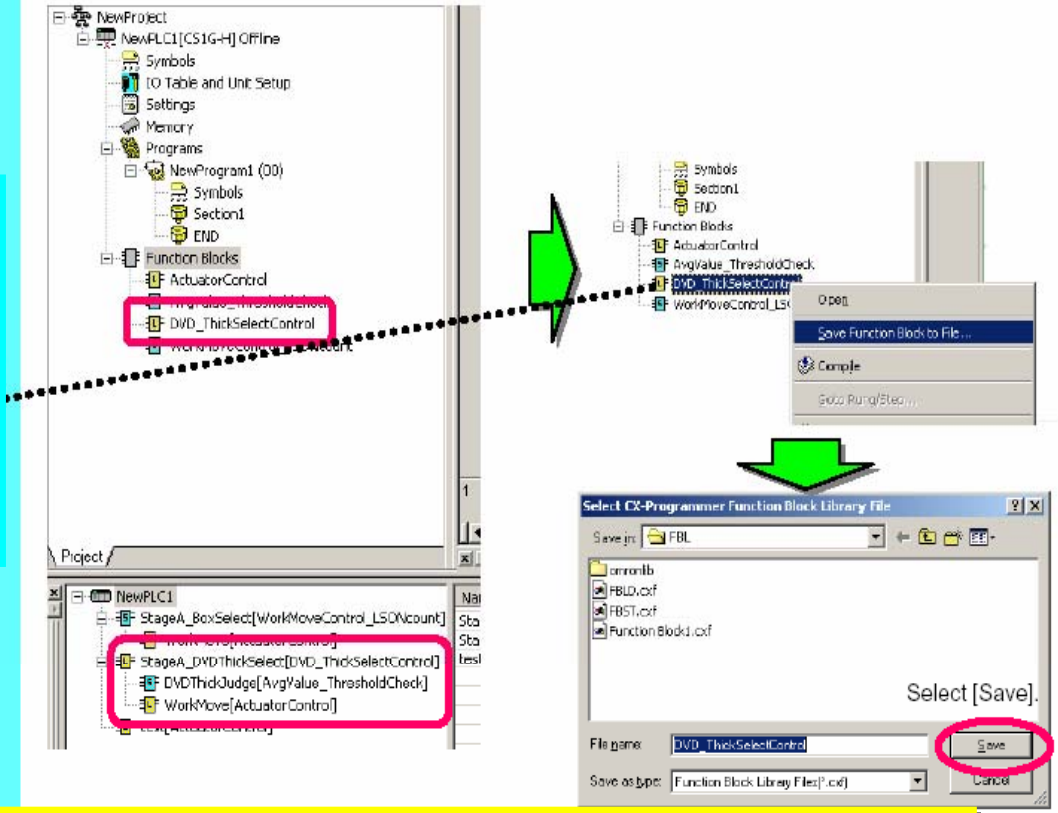
مقدار مورد نظر را وارد نموده وپس بر روي نماد Set كليك نماييد .



6. ايجاد كتابخانه فانكشن بلوكهاي تعريف شده

براي استفاده مجدد از فانكشن بلوكهايي كه تعريف نموده ايد، مي بايست انها را در يك كتابخانه (فايل) ذخيره نماييد .
زير شاخة مر بوط به فانكشن بلوكي كه مي خواهيد ذخيره نماييد ، را انتخاب كنيد .
در اين مثال فانكشن بلوك " DVD- ThickselectControl " را انتخاب نموده ايم .

در پنجرة
Project WorkSpace
برروي علامت گرافيكي مربوط به
فانكشن بلوك
"DVD-ThickSelect Control"
كليك راست نموده و گزينه
Save Function Block To File
را از منو گشوده شده انتخاب نماييد .



مسير پيش فرض ذخيره سازي فانكشن بلوكها به صورت زير مي باشد :
C:\ Progarm Files \ Omron \ CX- One \ FBL
در صورت تمايل به تغيير اين مسير از منوي Tools نرم افزار CX-Programmer گزينه Option را انتخاب نماييد و در پنجره گشوده شده در قسمت General و در بخش " FB Library Storage Folder " مسير مورد نظر براي ذخيره سازي فانكشن بلوكهاي ايجاد شده را وارد نماييد .
فانكشن بلوك هاي از پيش تعريف شده شركت امرن نيز ، در همين مسير و در شاخه Omronlib ذخيره شده اند .
شما نيز براي طبقه بندي و دسترسي راحت تر، مي توانيد يك شاخه ، فرضاً با عنوان Userlib براي ذخيره سازي فانكشن بلوكهايي كه تعريف كرده ايد ، ايجاد نماييد .



زماني كه فانكشن بلوكي را كه در آن فانكشن بلوك هاي ديگري نيز فراخواني شده باشد را ، ذخيره مي نماييد ، تمامي فانكشن بلوك هاي فراخواني شده نيز ذخيره خواهند شد و زماني كه آنها را در پروژه اي ديگر فراخواني مي نماييد ، ترتيب فراخواني فانكشن بلوكها در يكديگر همسان با ترتيبي است كه در فانكشن بلوكي كه ذخيره نموده ايد داشته اند .

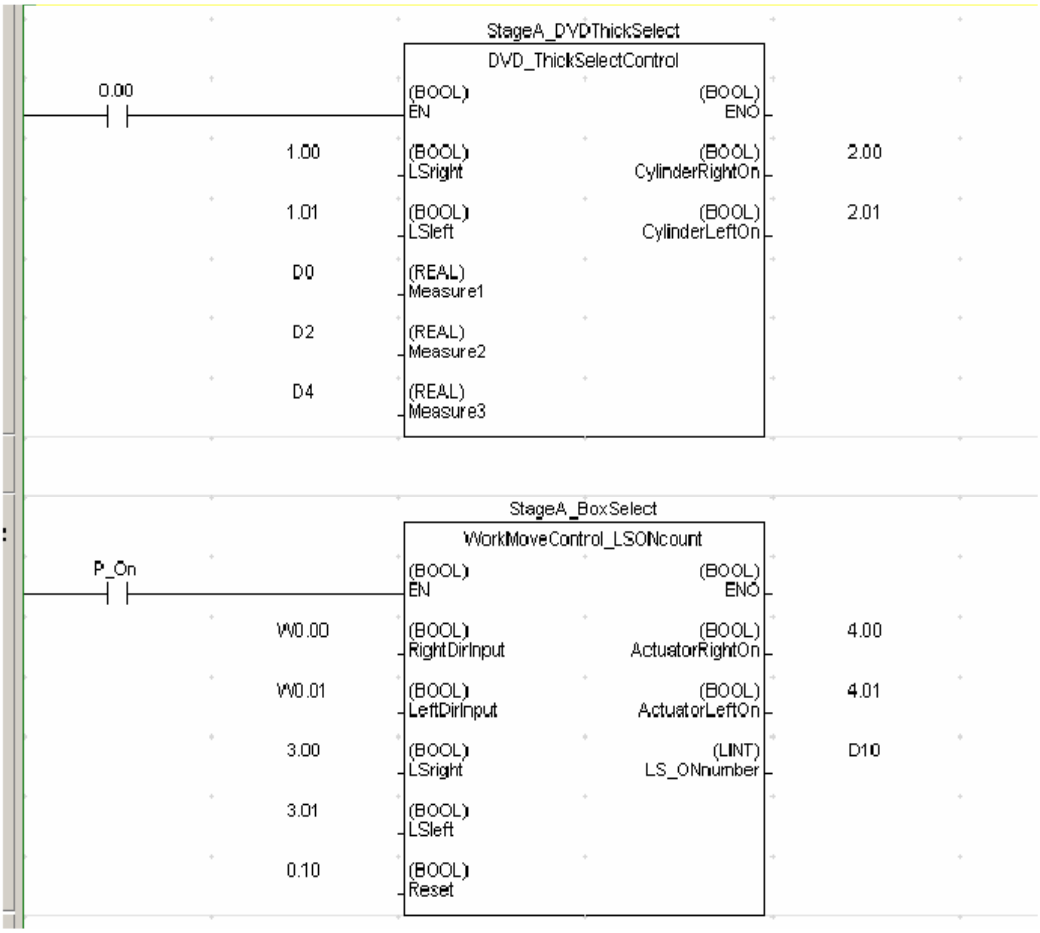
7. وارد كردن فانكشن بلوكها در برنامه اصلي

فانكشن بلوكهاي خطا يابي شده را در برنامه اصلي به كار برده و اين برنامه را به عنوان يك پروژه ذخيره سازي نماييد .
برنامه كلي ، همان ساختاري را دارد كه در بخش 5 -4 توصيح داده شد و نيز توضيح كلي اين برنا مه در بخش7 -5 كاملاً تشريح .

[Global Variables]

Name	Data Type	Address / Value	Rack Location
StageA_BoxSelect	FB [WorkMoveControl_LSONcount]	N/A [Auto]	
StageA_DVDThickSelect	FB [DVD_ThickSelectControl]	N/A [Auto]	

از نمايش متغير هاي داخلي فانكشن بلوكها صرفنظر شده است



در صورت نياز به توضيحات بيشتر به بخشهاي 6 -2 تا 9 -2 مراجعه نماييد

8. خطايابي برنامه اصلي

از آنجا كه فانكشن بلوكها بعد از تعريف خطا يابي شده اند ،
دربرنامه اصلي قسمتهاي زير مي بايست خطا يابي شود :

- قسمتهايي از برنامه كه مستقل از فانكشن بلوك مي باشند
- قسمتهاي از برنامه كه به يك متغير ورودي از فانكشن بلوك مرتبط مي باشند
- قسمتهايي از برنامه كه به يك پارا متر خروجي از فانكشن بلوك رجوع دارند

از آنجايي كه در مثال ما هيچ يك از اين قسمتها وجود ندارد و برنامه تنها شامل فانكشن بلوك مي باشد لذا از توضيح خطا يابي مابقي قسمتها صرفنظر مي نماييم

اطلاعات تكميلي

چگونگي حذف فانكشن بلوك استفاده شده در برنامه

زمانيكه مي خواهيد فانكشن بلوكي را حذف نماييد ، تنها حذف دستور مستقل مربوط به آن از برنامه كافي نيست زيرا متغير مربوط به آن در جدول ، متغيرهاي عمومي ثبت شده است .
در اين موارد ، زمانيكه خطايابي صورت مي پذيرد عنوان فانكشن بلوكهايي كه در جدول متغيرهاي عمومي وجود دارند ولي در برنامه از انها استفاده نشده است ،در پنجره خروجي نمايش داده خواهند شد و بدين ترتيب مي توانيد فانكشن بلوكهايي كه از آن استفاده ننموده ايد را شناسايي نمود ه و حذف نماييد .
فانكشن بلوكها حتي اگر در برنامه استفاده نشوند بخشي از CPU را كه مختص برنامه كاربر مي باشد اشغال خواهند نمود ، لذا پيشنهاد مي شود در صورتيكه ا ز فانكشن بلوكي در برنامه استفاده ننموده ايد قبل از انتقال برنامه به كنترلر ، آن را حذف نماييد .
بدين منظور مراحل زير را انجام دهيد :
براي خطايابي برنامه كليد F7 را بفشاريد

نتيجه خطا يابي در پنجره زيرين محيط برنامه نويسي (Out Window) نمايش داده خواهد شد .



بر روي خط مربوط به هشدار در مورد عدم استفاده از فانكشن بلوك دوبار كليك نماييد .
در اين حال جدول متغير هاي عمومي گشوده و خط مر بوط به تعريف فانكشن بلوك استفاده نشده آبي رنگ خواهد شد.
حال كليد DEL را بفشاريد .
در اين هنگام ديا لوگي گشوده خواهد شد كه شما را از حذف فانكشن بلوك مورد نظر مطلع مي سازد .
در صورت تمايل به حذف بر روي گزينه yes كليك چپ نماييد .

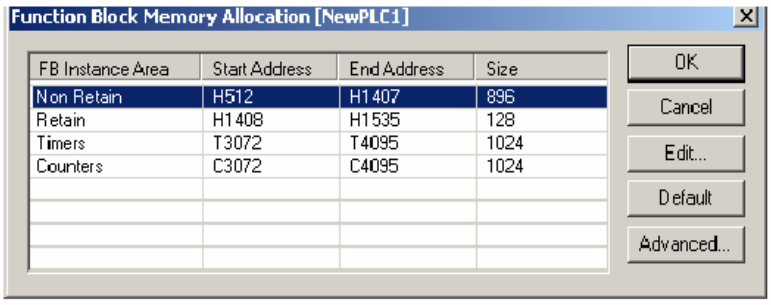
بعد از انجام مراحل ذكر شده ، فانكشن بلوك ، حذف خواهد شد

فضاي حافظه مختص فانكشن بلوكها

در صورت استفاده از فانكشن بلوك در برنامه مي بايست به هر فانكشن بلوك حافظه مورد نيازش را اختصاص دهيد .
نرم افزار cx-Programmer متناسب با فضايي كه ، از مسير زير مي توانيد به آن دسترسي پيدا نماييد، به صورت اتوماتيك به فانكشن بلوكها ، محدوده حافظه مورد نيازشان را اختصاص خواهد داد .
مسير دستيابي به محدوده حافظه ذخيره سازي فانكشن بلوكها : منو PLC ← Function Block Memory ← Function Block Memory Allocation
در اين قسمت 4 محدوده تحت عنوانهاي : - Counters - Timers - Retain - Not retain
وجود دارد و در صورت نياز مي توانيد محدوده حافظه مختص به هر بخش را تغيير دهيد .

- در صورت تغيير محدوده ها ، توجه داشته باشيد كه اگر محدوده مربوط به حافظه هاي ماندگار و يا فرار را تغيير مي دهيد ، وارد محدوده حافظه هاي مربوط به ورودي / خروجي هاي خاص و واحد SIO(Special Input / Output) كنترلر نشويد .
- محدوده حافظه مختص به فانكشن بلوك ها : كنترلر هاي نوع CS1 / CJ1- H / CJ1M از نسخه 3 به بـا لا محدوده حافظه اي مختص فانكشن بلوكها را دارا مي باشند كه عموماً قسمتي از محدوده حافظه ماندگار H مي باشد .

اين محدوده از آدرس H512 تا H1535 مي باشد .
اين محدوده از حافظه، به صورت پيش فرض در نرم افزار به فانكشن بلوكها اختصاص داده شده است و در برنامه نويسي قادر به استفاده از انها در دستورالعملها نمي باشيد .



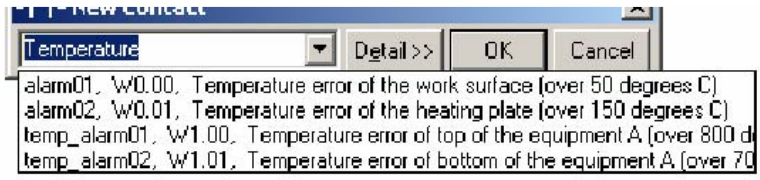
قابليتها

اين امكان وجود دارد كه در هنگام تعريف يك دستورالعمل ، ليستي از اسامي و توضيحات ورو دي / خروجي ها بعد از تعريف دستورالعمل نمايش داده شود . زماني كه يك عملگر تعريف كنتاكت و يا كويل را ، در محيط برنامه نويسي قرار مي دهيد ، و يك حرف را تايپ مي نماييد ، در فهرست زيرين قسمت تعريف متغير كه با كليك بر روي فلش كناري اين قسمت باز مي شود ليستي از تمامي سمبل هايي كه در عنوان آنها و يا توضيحاتشان از آن حرف استفاده شده است ، نمايش داده خواهد شد و مي توانيد متغير مورد نظر تان را از اين فهرست انتخاب نماييد .

به عنوان مثال لغت Temperature را در قسمت مربوط به تعريف متغير تايپ نموده ايم .



با كليك بر روي فلش كناري اين بخش و يا فشردن كليد F4 تمامي سمبلها و يا آدرسهايي كه در عنوان و يا توضيحات مربوط به آنها كلمه Temperature آمده است ، فهرست وار در اختيار شما قرار خواهند گرفت .



به عنوان مثال alarm02 به آدرس W0.01 و با توضيح (Temperature error of the heating Plate (Over 150 degrees C را از اين فهرست انتخاب نموده ايم كه با نماد alarm02 نمايش داده خواهد شد .



ضميمه :
مثالهايي ازدستورالعملهاي زبان برنامه نويسي ST

مثالهايي از دستورالعمل IF

فرمت كلي اين دستور :

IF شرايط اول
THEN فهرست دستورات مربوط به زمان برآورده شدن شرايط اول
ELSEIF شرايط دوم
THEN فهرست دستورات مربوط به زمان برآورده نشدن شرايط اول و برآورده شدن شرايط دوم
ELSE فهرست دستورات مربوط به زمان برآورده نشدن شرايط اول و دوم
ENDIF ;

شرايط 1 و 2 معادل يك مقدار بايناري با ارزش 0 يا 1 مي باشند و فهرست دستورات شامل چندين دستور ساده به طور مثال ; a:=a+1 مي باشد .
عبارت IF فهرست دستورالعملهاي 1 را در صورتيكه شرايط اول برقرار باشد انجام خواهد داد و در صورت صحت شرايط دوم كه پس از ELSEIF آورده شده است و عدم برقراري شرايط 1، فهرست دستورالعملهاي 2 انجام خواهد شد و در صورتيكه هيچ يك از شرايط 1 و 2 بر قرار نباشند در صورت وجود ELSE در برنامه فهرست دستورالعملهاي پس از ELSE انجام خواهد شد .پس از اجراي هر يك از فهرست دستورالعملها بعد از رسيدن به عبارت ENDIF كنترلر از دستورات مربوط به IF خارج و وارد مرحله بعدي برنامه خواهد شد .با به كاربردن يك IF در برنامه مي توانيد در صورت نياز چندين ELSEIF پس از آن به ازاء شرايط متفاوت قرار دهيد. ولي تنها مجاز به استفاده از يك ELSE پس از آن مي باشيد .به علاوه اجرا دستورالعملهاي يك IF مي تواند وابسته به برقراري يك IF ديگر باشد و يا به عبارت بهتر امكان تعريف چندين IF تو در تو را دارا مي باشيد كه براي درك بهتر اين مطلب مي تونيد به مثال 5 مراجعه نماييد.

مثال 1 :

```
Example 1
IF a > 0 THEN
  b := 0;
END_IF;
```

در اين مثال ، اگر متغير a بزرگتر از صفر باشد متغير b مقدار صفر را به خود اختصاص خواهد داد .
و اگر متغير a بزرگتر از صفر نباشد هيچ عملياتي بر روي متغير b انجام نخواهد شد و به مراحل بعدي برنامه كه پس از دستور ENDIF واقع شده اند منتقل خواهيم شد .

مثال 2 :

```
Example 2
IF a THEN
  b := 0;
END_IF;
```

در اين مثال اگر متغير a(بايناري) داراي مقدار 1 باشد متغير b مقدار صفر را به خود اختصاص خواهد داد . و اگر متغير aداراي مقدار صفر باشد هيچ عملياتي بر روي متغير b انجام نخواهد شد و به مراحل بعدي برنامه كه پس از دستور ENDIF واقع شده اند منتقل خواهيم شد .

مثال 3 :

```
Example 3
IF a > 0 THEN
  b := TRUE;
ELSE
  b := FALSE;
END_IF;
```

در اين مثال ،اگر متغير a بزرگتر از صفر باشد متغير b مقدار 1 را به خود اختصاص خواهد داد و به مراحل بعدي برنامه كه پس از دستور ENDIF واقع شده اند منتقل خواهيم شد .
و اگر متغير a بزرگتر از صفر نباشد به مرحله بعد از ELSE منقل شده و متغير b مقدار صفر را به خود اختصاص خواهد داد وسپس به مراحل بعدي برنامه كه پس از دستور ENDIF واقع شده اند منتقل خواهيم شد .

مثال 4 :

```
Example 4
IF a < 10 THEN
  b := TRUE;
  c := 100;
ELSIF a  > 20 THEN
  b := TRUE;
  c := 200;
ELSE
  b := FALSE;
  c := 300;
END_IF;
```

در اين مثال ، اگر متغير a كوچكتر از 10 باشد متغير b مقدار 1 را به خود اختصاص خواهد داد ونيز متغير C مقدار 100 را به خود اختصاص داده و سپس به مراحل بعدي برنامه كه پس از دستور ENDIF واقع شده اند منتقل خواهيم شد .
و اگر متغير a بزرگتر يا برابر 10 باشد به مرحله بعد كه دستور ELSEIF مي باشد منتقل مي شويم و در اين حال اگر متغير a بزرگتر از 0 2باشد متغير b مقدار 1 را به خود اختصاص خواهد داد ونيز متغير C مقدار 200 را به خود اختصاص داده و سپس به مراحل بعدي برنامه كه پس از دستور ENDIF واقع شده اند منتقل خواهيم شد .
و اگر متغير a در محدوده مابين 10 و 20 باشد (شرايط برقراري هيچيك از IF و نيز ELSEIF بر قرار نباشد) به مرحله بعد از ELSE منقل شده و متغير b مقدار صفر را به خود اختصاص خواهد داد ونيز متغير C مقدار 300 را به خود اختصاص داده سپس به مراحل بعدي برنامه كه پس از دستور ENDIF واقع شده اند منتقل خواهيم شد .

مثال 5 :

در اين مثال (يك مثال از IF و THEN هاي تو در تو)، اگر متغير a (باينري) داراي مقدار 1 باشد ، متغير b نيز مقدار 1 را به خود اختصاص خواهد داد و كنترلر به مراحل بعد از دستور ENDIF برنامه منتقل خواهد شد.

در صورتيكه متغير a داراي مقدار صفر باشد هيچ عملياتي بر روي متغير b صورت نپذيرفته و كنترلر به مراحل بعد از دستور ELSE منتقل مي گردد كه خود شامل دستورات شرطي مشابه انچه در مثال 3 بيان شد مي باشد و بعد از اجرا اين مراحل متغير d مقدار 400 را به خود اختصاص داده و كنترل به مراحل بعدي برنامه كه پس از دستور ENDIF واقع شده اند منتقل خواهد شد .

Example 5

```
IF a THEN
  b :=  TRUE;
ELSE
  IF c>0 THEN
    d := 0;
  ELSE
    d := 100;
  END_IF;
  d := 400;
END_IF;
```

مثالهايي از دستورالعمل WHILE

فرمت كلي اين دستور :

WHILE شرايط

DO فهرست دستورات مربوط به زمان برآورده شدن شرايط

END- WHILE ;

شرايط اجراي WHILE معادل يك مقدار باينري با ارزش 0 يا 1 مي باشند و فهرست دستورات شامل جندين دستور ساده مي باشد .
عبارتWHILE متناوباً فهرست دستورالعملهاي خود را در صورتيكه شرايط آن برقرار باشد انجام خواهد داد و در صورت شرايط برقرار نباشد ، كنترلر از اجرا دستورات مربوط به WHILE خارج و به اجراي برنامه هاي نوشته شده پس از دستور END- WHILE خواهد پرداخت .

مثال 1 :

در اين مثال ، در صورت برقراري شرايط اجراي حلقه WHILE (كوچكتر بودن متغير a ، از عدد 10)در هر بار اجرا حلقه دستورالعملهاي زير متناوباً اجرا خواهد شد :
يك واحد به مقدار متغير a افزوده و متغير b نيز در عدد 2 ضرب خواهد شد .
بعد از هر بار اجراي اين دستورات ، كنترلر به ابتداي حلقه WHILE برگشته و بر قراري شرايط را چك مي نمايد و تا زمانيكه شرط اجرا (كوچكتر بودن متغير a از عدد 10) بر قرار باشد عمليات فوق تكرا رخواهد شد .
زماني كه متغير a بزرگتر و يا مساوي عدد 10 شود ديگر دستورات داخل حلقه اجرا نشده و كنترلر
به اجراي مراحل بعدي برنامه كه پس از دستور END- WHILE واقع شده اند خواهد پرداخت .

Example 1

```
WHILE a < 10 DO
  a := a + 1;
  b := b * 2.0;
END_WHILE;
```

مثال 2 :

در اين مثال اگر متغير a(باينري) داراي مقدار 1 باشد در هر بار اجرا يك واحد به مقدار متغير b افزوده شده و نيز با توجه به شرايط ، دستورات شرطي IF ..... THEN اجرا مي شوند وپس از هر بار اجرا به ابتدا حلقه WHILE باز گشته و شرايط اجراي مجدد را چك مي نمايد و تازمانيكه متغير a داراي مقدار 1 باشد اين روند تكرار خواهد شد .
اگر متغير a داراي مقدار 0 باشد به اجراي مراحل بعدي برنامه كه پس از دستور END- WHILE واقع شده اند خواهد پرداخت .

Example 2

```
WHILE a DO
  b := b + 1;
  IF b > 10 THEN
    a:= FALSE;
  END_IF;
END_WHILE;
```

مثال 3 :

در اين مثال اگر عبارت شرط انجام حلقه WHILE داراي مقدار منطقي 1 باشد و يا به عبارت ديگر تا زماني كه پس از افزودن يك واحد به مقدار متغير a ، مقدار معادل ، بزرگتر يا مساوي توان دوم مقدار متغير b باشد ، در هر بار اجراي حلقه، يك واحد به مقدار متغير a افزوده و نيز مقدار متغير b بر مقدارمتغير c تقسيم شده و جايگزين مقدار قبلي متغير b مي گردد.
وپس از هر بار اجرا به ابتدا حلقه WHILE باز گشته و شرط اجرا مجدد را چك مي نمايد و تازمانيكه شرط بر قرار باشد باشد اين روند تكرار خواهد شد و در غير اين صورت به اجراي مراحل بعدي برنامه كه پس از دستور END- WHILE واقع شده اند خواهد پرداخت .

Example 3

```
WHILE (a + 1) >= (b * 2) DO
  a := a + 1;
  b := b / c;
END_WHILE;
```

مثال 4 :

در اين مثال . اگر عبارت معادل شرط انجام حلقه WHILE داراي مقدار منطقي 1 باشد و يا به عبارت ديگر تا زماني كه پس ازكاهش مقدار متغير b از مقدار متغير a ، نتيجه كوچكتر يا مساوي مقدار متغير b به اضافه مقدار متغيرc باشد ، در هر بار اجرا حلقه يك واحد به مقدار متغير a افزوده و نيز متغير bدر متغير a ضرب شده و جايگزين مقدار قبلي متغير b مي گردد.

وپس از هر بار اجرا به ابتدا حلقه WHILE باز گشته و شرط اجرا مجدد را چك مي نمايد و تازمانيكه شرط بر قرار باشد باشد اين روند تكرار خواهد شد و در غير اين صورت به مراحل بعدي برنامه كه پس از دستور END-WHILE واقع شده اند منتقل خواهيم شد .

Example 4

```
WHILE (a - b) <= (b + c) DO
  a := a + 1;
  b := b * a;
END_WHILE;
```

مثالهايي از دستورالعمل REPEAT

فرمت كلي اين دستور :

REPEAT فهرست دستورات مورد نظر

UNTIL شرايط توقف اجرا دستورات فوق

END- REPEAT ;

شرايط اجراي REPEAT معادل يك مقدار بايناري با ارزش 0 يا 1 مي باشند و فهرست دستورات شامل جندين دستور ساده مي باشد .

دستورREPEAT در صورتيكه شرايط توقف آن برقرار نباشد متناوباً فهرست دستورالعملهاي خود را در انجام خواهد داد و در صورتيكه شرايط توقف برقرار شود ، كنترلر از دستورات مربوط به REPEAT خارج و برنامه هاي نوشته شده پس از دستور END- REPEAT را اجرا خواهد نمود .

مثال 1 :

در اين مثال در اولين بار اجرا دستورالعملها كه عبارتند از:

يك واحد افزايش به مقدار متغير a و نيز ضرب متغير b در عدد 2 ، انجام شده و پس از اجرا اين دستورات ، شرايط بعد از دستور UNTIL چك شده و در صورتيكه مقدار منطقي اين عبارت برابر صفر باشد و يا به عبارت ديگر مقدار متغير a كوچكتر يا مساوي 10 باشد، به ابتدا حلقه REPEAT بازگشته و مجدداً دستورالعملهاي بيان شده اجرا مي شود و پس از اجرا بازهم شرط پس از UNTIL چك مي شود و اين روند تا زمانيكه مقدار منطقي اين عبارت برابر 1 گردد (مقدار متغير a بيش از 10 شود) ادامه خواهد يافت .

و اگر متغير a بزرگتر از 0 1شود كنترلر به اجراي مراحل بعدي برنامه كه پس از دستور END- REPEAT واقع شده اند خواهد پرداخت

Example 1

```
REPEAT
  a := a + 1;
  b := b * 2.0;
UNTIL a > 10
END_REPEAT;
```

مثال 2 :

در اين مثال در اولين بار اجرا دستورات مورد نظر كه عبارتند از: يك واحد افزايش به مقدار متغير b و يك سري دستورات شرطي اجرا شده و پس از اجراي اين دستورات ، شرط بعد از دستور UNTIL چك شده و در صورتيكه مقدار منطقي اين عبارت برابر صفر باشد و يا به عبارت ديگر مقدار متغير a برابر0 باشد، به ابتدا حلقه REPEAT بازگشته و مجدداً دستورالعملهاي بيان شده اجرا مي شود و پس از اجرا ، بازهم شرط پس از UNTIL چك مي شود و اين روند تا زمانيكه مقدار منطقي اين عبارت برابر 1 گردد (مقدار متغير aبرابر1 شود) ادامه خواهد يافت . و اگر متغير a برابر 1شود كنترلر به اجراي مراحل بعدي برنامه كه پس از دستور END- REPEAT واقع شده اند خواهد پرداخت .

Example 2

```
REPEAT
  b := b + 1;
  IF b > 10 THEN
    a:= FALSE;
  END_IF;
UNTIL a
END_REPEAT;
```

مثال 3 :

در اين مثال در اولين بار اجرا، دستورات مورد نظر كه عبارتند از: يك واحد افزايش مقدار متغير a و تقسيم متغير b بر متغير c ، اجرا شد ه و پس از اجراي اين دستورات ، شرط بعد از دستور UNTIL چك شده و در صورتيكه مقدار منطقي اين عبارت برابر صفر باشد و يا مقدار متغير a بعلاوه 1 كوچكتر از مقدار حاصل از ضرب متغيرb در عدد 2 باشد به ابتدا حلقه REPEAT بازگشته و مجدداً دستورالعملها ي بيان شده اجرا مي شود و پس از اجرا بازهم شرط پس از UNTIL چك مي شود و اين روند تا زمانيكه مقدار منطقي اين عبارت برابر 1 گردد (مقدار متغير a بعلاوه 1 بزرگتر يا مساوي مقدار حاصل از ضرب متغير b در عدد 2 باشد) ادامه خواهد يافت . و در غير اين صورت كنترلر به اجراي مراحل بعدي برنامه كه پس از دستور END- REPEAT واقع شده اند خواهد پرداخت .

Example 3

```
REPEAT
  a := a + 1;
  b := b / c;
UNTIL (a + 1) >= (b * 2)
END_REPEAT;
```

مثال 4 :

در اين مثال در اولين بار اجرا، دستورات مورد نظر كه عبارتند از: يك واحد افزايش مقدار متغير a و ضرب متغير b در متغير a ،اجرا شده و پس از اجراي اين دستورات ، شرط بعد از دستور UNTIL چك شده و در صورتيكه مقدار منطقي اين عبارت برابر صفر باشد و يا تفاضل متغيرهاي b و a از مقدار حاصل جمع متغيرهاي b , c كوچكتر باشد به ابتدا حلقه REPEAT بازگشته و مجدداً دستورالعملها ي بيان شده اجرا مي شود و پس از اجرا بازهم شرط پس از UNTIL چك مي شود و اين روند تا زمانيكه مقدار منطقي اين عبارت برابر 1 گردد (تفاضل متغيرهاي b و a از مقدار حاصل جمع متغيرهاي b , c بزرگتر و يا مساوي آن باشد) ادامه خواهد يافت . و در غير اين صورت كنترلر به اجراي مراحل بعدي برنامه كه پس از دستور END- REPEAT واقع شده اند خواهد پرداخت .

Example 4

```
REPEAT
  a := a + 1;
  b := b * a;
UNTIL (a - b) <= (b + c)
END_REPEAT;
```

مثالهايي از دستورالعمل FOR

فرمت كلي اين دستور :

For متغير كنترلي := مقدار ابتدايي محدوده

TO مقدار انتهايي محدوده

BY مقدار تغييرات متغير كنترلي در هر مر حله

DO فهرست دستورالعملها

END- FOR ;

متغير كنترلي ميبايست يك عدد صحيح باشد لذا ، اعدا د ابتدايي و انتهايي مر بوط به دفعات اجراي حلقه و نيز مقدارمعادل تغييرات متغير كنترلي در هر مر حله نيز مي بايست اعدادي صحيح باشند. فهرست دستورات شامل چندين دستورالعمل ساده مي باشد .دستورFOR تا زمانيكه متغير كنترلي در رنج مجاز باشد ، متناوباً فهرست دستو رالعملها را اجرا خواهد نمود و در هر مرحله از اجرا مقدار متغير كنترلي به اندازه اي كه بعد ازدستور BY آمده ، افزايش خواهد يافت و درصورتيكه اين قسمت از دستورالعمل را ننوشته باشيد _ (مقدار تغييرات متغير كنترلي در هر مر حله BY)_مقدار متغير كنترلي به طور پيش فرض در هر بار اجرا يك واحد افزايش خواهد يافت . بعد از خارج شدن مقدار متغير كنترلي ،از رنج مجاز ، دستورات بعد از END - FOR اجرا خواهد شد .اين امكان وجود دارد كه در يك حلقه FOR ، حلقه FOR ديگر ي را نيز به كار برد .

مثال 1 :

در اين مثال در اولين بار اجرا، متغير كنترلي a مقدار يك را به خود اختصاص خواهد داد و اين مقدار با مقدار انتهايي رنج كه بعد از دستور TO آمده است (10) مقايسه شده و در صورتيكه از اين مقدار كمتر و يا برابر آن باشد فهرست دستورات (; b := b+a) اجرا خواهد شد .

بعد ازهر بار اجرا يك واحد به مقدار متغير كنترلي a افزوده شده و مجدداً به ابتداي حلقه برگشته و مقدار جديد با انتهاي رنج (10) مقايسه مي شود . تا زمانيكه ، مقدار متغير كنترلي از رنج انتهايي كوچكتر يا مساوي آن باشد اين روتين ادامه خواهد يافت و پس از آنكه مقدار متغير كنترلي بزرگتر از رنج انتهايي گشته از حلقه FORخارج و دستورات بعد از END- FOR اجرا خواهد شد.

```
Example 1
FOR a := 1 TO 10 DO
  b := b + a;
END_FOR;
```

مثال 2 :

در اين مثال در اولين بار اجرا ، متغير كنترلي a مقدار يك را به خود اختصاص خواهد داد و اين مقدار با مقدار انتهايي رنج كه بعد از دستور TO آمده است (10) مقايسه شده و در صورتيكه از اين مقدار كمتر و يا برابر آن باشد فهرست دستورات (; c:= c+10 و; b := b+a) اجرا خواهد شد .

بعد از هر بار اجرا دو واحد به مقدار متغير كنترلي a افزوده شده و مجدداً به ابتداي حلقه برگشته و مقدار جديد با انتهاي رنج (10) مقايسه مي شود . تا زمانيكه ، مقدار متغير كنترلي از رنج انتهايي كوچكتر يا مساوي آن باشد اين روتين ادامه خواهد يافت و پس از آنكه مقدار متغير كنترلي بزرگتر از رنج انتهايي گشته از حلقه FORخارج و دستورات بعد از END- FOR اجرا خواهد شد.

```
Example 2
FOR a := 1 TO 10 BY 2 DO
  b := b + a;
  c := c + 1.0;
END_FOR;
```

مثال 3 :

در اين مثال در اولين بار اجرا، متغير كنترلي a مقدار 10 را به خود اختصاص خواهد دادو اين مقدار با مقدار انتهايي رنج كه بعد از دستور TO آمده است (1) مقايسه شده و در صورتيكه از اين مقدار بزرگتر و يا برابر آن باشد فهرست دستورات (; c:= c+10 و; b := b+a) اجرا خواهد شد .

بعد از هر بار اجرا يك واحد از مقدار متغير كنترلي a كاسته شده و مجدداً به ابتداي حلقه برگشته و مقدار جديد با انتهاي رنج (10) مقايسه مي شود . تا زمانيكه ، مقدار متغير كنترلي از رنج انتهايي بزرگتر يا مساوي آن باشد اين روتين ادامه خواهد يافت و پس از آنكه مقدار متغير كنترلي بزرگتر از رنج انتهايي گشته از حلقه FORخارج و دستورات بعد از END- FOR اجرا خواهد شد.

```
Example 3
FOR a := 10 TO 1 BY -1 DO
  b := b + a;
  c := c + 1.0;
END_FOR;
```

مثال 4 :

در اين مثال در اولين بار اجرا، متغير كنترلي a يك واحد بيشتر از مقدار متغير b را به خود اختصاص خواهد داد و اين مقدار با مقدار انتهايي رنج كه بعد از دستور TO آمده است (دو واحد بيشتر از مقدار متغير c) مقايسه شده و در صورتيكه از اين مقدار كوچكتر و يا برابر آن باشد فهرست دستورات

(; e:= e +1 و; d := d+a) اجرا خواهد شد .

بعد از هر بار اجرا يك واحد به مقدار متغير كنترلي a افزوده شده و مجدداً به ابتداي حلقه برگشته و مقدار جديد با انتهاي رنج (دو واحد بيشتر از مقدار متغير c) مقايسه مي شود .

تا زمانيكه ، مقدار متغير كنترلي از رنج انتهايي كوچكتر يا مساوي آن باشد اين روتين ادامه خواهد يافت و پس از آنكه مقدار متغير كنترلي بزرگتر از رنج انتهايي گشته از حلقه FORخارج و دستورات بعد از END- FOR اجرا خواهد شد.

```
Example 4
FOR a := b + 1 TO c + 2 DO
  d := d + a;
  e := e + 1;
END_FOR;
```

FOR Statement Examples

Example 5

```
FOR a := b + c TO d - e BY f DO
  g := g + a;
  h := h + 1.0;
END_FOR;
```

مثال 5 :

در اين مثال در اولين بار اجرا، متغير كنترلي a برابر مقدار متغير c بعلاوه مقدار متغير b خواهد بود. و اين مقدار با مقدار انتهايي رنج كه بعد از دستور TO آمده است (حاصل تفاضل مقدار متغير e از مقدار متغير d) مقايسه شده و در صورتيكه از اين مقدار كوچكتر و يا برابر آن باشد فهرست دستورات

(h := h + 10 ; g:= g +a ;) اجرا خواهد شد .

بعد از هر بار اجرا به تعدادي برابر با مقدار متغير f به مقدار متغير كنترلي a افزوده شده و مجدداً به ابتداي حلقه برگشته و مقدار جديد با انتهاي رنج (حاصل تفاضل مقدار متغير e از مقدار متغير d) مقايسه مي شود .

تا زمانيكه ، مقدار متغير كنترلي از رنج انتهايي كوچكتر يا مساوي آن باشد اين روتين ادامه خواهد يافت و پس از آنكه مقدار متغير كنترلي بزرگتر از رنج انتهايي گشته از حلقه FORخارج و دستورات بعد از END- FOR اجرا خواهد شد.

مثالهايي از دستورالعمل CASE

فرمت كلي اين دستور :

Case عبارت كنترلي

OF 1 : مقدار مشخصه حالت1 دستورالعملهاي مربوط به حالت

2 : مقدارمشخصه حالت2 دستورالعملهاي مربوط به حالت

.....

ELSE فهرست دستور العملهايي كه در صورتيكه مقدار متغير كنترلي برابر مقدار مشخصه هيچ يك از حالتهاي بالا نباشد مي بايست اجرا گردد

END- CASE ;

مقدار عبارت كنترلي ميبايست يك عدد صحيح باشد ، فهرست دستورات شامل چندين دستورالعمل ساده مي باشد .

دستورCASE مقدار عبارت كنترلي را چك مي نمايد و متناسب با آن مقدار، دستو رالعملها ي حالتي ، كه مقدار مشخصه ان برابر مقدار متغير كنترلي بوده را انجام داده و درصورتيكه اين مقدار برابر مقدار مشخصه هيچ يك از حالتها نباشد در صورت استفاده از دستورالعمل ELSE دستورات بعد ازآن و در غير اين صورت دستورات بعد از دستورالعمل END- CASE اجرا خواهد شد . در صورتيكه مي خواهيد به ازاء چندين مقدار مشخصه متفاوت ، دستورالعملها ي يكساني اجرا شود مابين تمامي اين مقادير مشخصه ' ,' قرار داده و بعد از آخرين مقدار مشخصه پس از نماد ' : ' دستورالعملهاي مورد نظرتان را بنويسيد .

در صورتيكه در نظر داريد يك رنج از اعداد را به عنوان مقدار مشخصه در نظر بگيريد مي بايست از نماد ' .. ' به صورت زير استفاده نماييد :

عدد ابتداي رنج .. عدد انتهاي رنج ، به عنوان مثال ; a:= a+1 : 1..10 در اين حال در صورتيكه مقدار متغير كنترلي عددي در محدوده مابين 1 تا 10 با شد يك واحد به مقدار متغير a افزوده خواهد شد .

Example 1

```
CASE a OF
  2 :  b := 1;
  5 :  c := 1.0;
END_CASE;
```

مثال 1 :

در اين مثال مقدار متغير كنترلي a محاسبه شده و سپس با مقادير مشخصه حالتهاي مختلف مقايسه مي گردد .

در صورتيكه مقدار متغير كنترلي a ، برابر عدد 2 باشد دستورالعمل حالت اول اجرا شده و مقدار متغير b برابر عدد 1 خواهد شد و سپس به اجراي دستورات بعد از دستورالعمل END- CASE خواهد پرداخت

در صورتيكه مقدار متغير كنترلي a برابر عدد 5 باشد دستورالعمل حالت دوم اجرا شده و مقدار متغير c برابر عدد 1.0 خواهد شد و سپس به اجراي دستورات بعد از دستورالعمل END- CASE خواهد پرداخت

در صورتيكه مقدار متغير كنترلي برابر هيچ يك از دومقدار بالا نباشد ، دستورات بعد ازEND- CASE اجرا خواهد شد .

Example 2

```
CASE a + 2 OF
  -2 :  b := 1;
  5 :  c := 1.0;
ELSE
  d := 1.0;
END_CASE;
```

مثال 2 :

در اين مثال مقدار متغير a محاسبه شده و با عدد 2 جمع مي گردد و سپس نتيجه با مقادير مشخصه حالتهاي مختلف مقايسه مي گردد .

در صورتيكه اين مقدار برابر عدد 2- باشد دستورالعمل حالت اول اجرا شده و مقدار متغير b برابر عدد 1 خواهد شد و سپس به اجراي دستورات بعد از دستورالعمل END- CASE خواهد پرداخت

در صورتيكه اين مقدار برابر عدد 5 باشد دستورالعمل حالت دوم اجرا شده و مقدار متغير c برابر عد 1.0 خواهد شد و سپس به اجراي دستورات بعد از دستورالعمل END- CASE خواهد پرداخت

در صورتيكه اين مقدار برابر هيچ يك از دومقدار بالا نباشد ، دستورات بعد ازELSE اجرا شده و مقدار متغير dبرابر 1.0خواهد شد .

مثالهايي از دستور EXIT

```
WHILE expression DO
   statement-list1;
   EXIT;
END_WHILE;
statement-list2;

REPEAT
  statement-list1;
  EXIT;
UNTIL expression
END_REPEAT;
statement-list2;

FOR control variable := integer expression1 TO integer expression2 [ BY integer expression3 ] DO
  statement-list1;
  EXIT;
END_FOR;
statement-list2;
```

دستور EXIT انجام عمليات حلقه اي متناوب را براي انتقال به مر حله بعدي برنامه متوقف مي نمايد و تنها براي دستورالعملهاي تكراري (WHILE , REPEAT , FOR) قابل استفاده مي باشد .

زمانيكه اين دستور در يك حلقه تكرار اجرا مي شود اجراي فهرست دستورالعملهاي اول متوقف و فهرست دستورالعملهاي دوم اجرا خواهد شد .

Example 1

```
WHILE a DO
 IF c = TRUE THEN
   b:=0;EXIT;
 END_IF;
 IF b > 10 THEN
    a:= FALSE;
 END_IF;
END_WHILE;
d:=1;
```

مثال 1 :

اگر شرايط IF اول بر قرار باشد (مقدار متغير c برابر 1 باشد) ،فهرست دستورات مربوطه (b:= 0 ; EXIT)

كه در داخل حلقه WHILEمي باشند اجرا خواهد شد و بعد از اجرا دستور EXIT ، حلقه WHILEمتوقف و مرحله بعدي برنامه كه بعد از دستور END- WHILEقرار دارد اجرا شده و مقدار متغير d برابر 1 خواهد شد .

Example 2

```
 a:=FALSE;
FOR i:=1 TO 20 DO
    FOR j:=0 TO 9 DO
        IF i>=10 THEN
            n:=i*10+j;
            a:=TRUE;EXIT;
        END_IF;
    END_FOR;
    IF a THEN EXIT; END_IF;
END_FOR;
d:=1;
```

مثال 2 :

اگر شرايط IF اول بر قرار باشد (مقدار متغير i بزرگتر ويا برابر عدد 10 باشد) ،

فهرست دستورات مربوطه (n:= i*10+j ;a:= True ; EXIT)

كه در داخل حلقه FOR داخلي ، مي باشند اجرا خواهد شد و بعد از اجرا دستور EXIT ، حلقه FOR داخلي متوقف و مرحله بعدي برنامه كه بعد از دستور END- FOR اول ، قرار دارد اجرا شده

و در صورتيكه مقدار متغير a برابر 1 باشد ، دستور EXIT دوم نيز اجرا شده و از حلقه FOR بيروني نيز خارج شده و دستور بعد از END- FOR دوم اجرا خواهد شد و مقدار متغير d برابر 1 خواهد شد .

مثالهايي از دستور العمل RETURN

```
statement-list1;
RETURN;
statement-list2;
```

با اجراي دستور RETURN اجراي دستورالعملهاي فانكشن بلوك متوقف و بدون اجراي ما بقي فهرست دستورات به برنامه اي كه در آن فانكشن بلوك فرا خواني شده بوده است باز خواهيم گشت .

مثال 1 :

```
Example 1
IF a_1*b>100 THEN
  c:=TRUE;RETURN;
END_IF;
IF a_2*(b+10)>100 THEN
  c:=TRUE;RETURN;
END_IF;
IF a_3*(b+20)>100 THEN
  c:=TRUE;
END_IF;
```

اگر شرايط IF اول و يا دوم بر قرار باشد

(مقدار عبارت a-b و a-2(b+10) ، بزرگتر از عدد 100 باشد) ،

فهرست دستورات مربوطه (RETURN ; TRUE :=c) اجرا شده و بعد از اجرا دستور RETURN ، اجراي دستورات داخلي فانكشن بلوك متوقف و به برنامه اي كه فانكشن بلوك را فراخواني نموده است باز خواهيد گشت .

مثالهايي از متغير هاي آرايه اي

نام متغير[انديس زير نويس]

يك آرايه مجموعه اي از متغير ها مي باشد و سايز آن در جدو ل تعريف متغير هاي فانكشن بلوك مشخص مي گردد .

با استفاده از انديس زير نويس متغير آرايه اي ، مي توان هر يك از متغير هاي زير مجموعه آنرا به صورت مجزا به كار برد .

انديس متغير آرايه اي مي بايست يك مقدار مثبت حقيقي ، يك عبارت با مقدار معادل عدد صحيح و يا يك متغير با مقدار عدد صحيح باشد .

اين انديس به صورت پايه ، صفر است و با اين انديس صفر به متغير اول از مجموعه متغير ها ي آن آرايه دسترسي پيدا خواهيد نمود .

به همين ترتيب با انديس 1 به متغير دوم و ..... دسترسي پيدا خواهيد نمود .

اخطار :

در صورتيكه انديس ، يك عبارت با مقدار معادل ، عدد صحيح و يا يك متغير با مقدار صحيح باشد ، مي بايست از اينكه مقدار انديس در رنج مجاز ، رنج تعداد متغيرهاي آرايه اي مي باشد ، مطمئن گرديد .

براي اطلاع از جزئيات چگونگي نوشتن مطمئن تر، در هنگام استفاده از متغير هاي آرايه اي ، به مثال 5 مراجعه نماييد .

مثال 1 :

```
Example 1
a[0] := 1;
a[1] := -2;
a[2] : = 1+2;
a[3] : = b;
a[4] : = b+1;
```

در اين مثال متغير a ، يك متغير آرايه اي به طول 5 و از نوع اعداد صحيح مي باشد .متغير b نيز از نوع اعداد صحيح مي باشد .

بعد از اجراي اين قسمت از برنامه ، متغير اول از متغير آرايه اي a ، مقدار 1 ،

متغير دوم آن مقدار 2 - ،

متغير سوم آن مقدار 3 ،

متغير چهارم آن مقدار متغير b ،

و متغير پنجم آن يك واحد بيشتر از مقدار متغير b را به خود اختصاص خواهد داد .

مثال 2 :

```
Example 2
c[0] := FALSE;
c[1] := 2>3;
```

در اين مثال متغير c ، يك متغير آرايه اي به طول 2 و از نوع باينري مي باشد .

بعد از اجراي اين قسمت از برنامه ،

متغير اول از متغير آرايه اي a ، مقدار 0 ،

ومتغير دوم آن نيز مقدار 0 را به خود اختصاص خواهد داد .

Array Examples

Example 3

```
d[9]:= 2.0;
```

مثال 3 :

در اين مثال متغير d ، يك متغير آرايه اي به طول 10 و از نوع اعداد حقيقي مي باشد .

بعد از اجراي اين قسمت از برنامه ،

متغيرآخر آن ، مقدار 2.0 را به خود اختصاص خواهد داد .

Example 4

```
a[1] := b[2];
```

مثال 4 :

در اين مثال هر دو متغيرa و b ، متغيرهاي آرايه اي و از يك نوع مي باشند .

بعد از اجراي اين قسمت از برنامه ،

مقدار متغيراول از متغير آرايه اي a ، برابر مقدار متغير دوم متغير آرايه اي b،خواهد شد .

Example 5

```
 a[b] := 1;
a[b+1] := 1;
a[(b+c) *( d-e)] := 1;
```

توجه : در صورتيكه از متغيرهاي صحيح و يا عبارات محاسباتي ، براي تعيين انديس متغيرهاي آرايه اي استفاده مي نماييد ، مقدار واقعي انديس ، در زمان اجرا نمايش داده نخواهد شد ، لذا كاربر مي بايست از اينكه مقدار واقعي انديس در رنج مجاز طول متغير ارايه اي مي باشد مطمئن شود .

براي رفع اين مشكل ، مي توانيد ، از يك روتين " اطمينان از در رنج بودن مقدار انديس " استفاده نماييد ، كه در مثال زير اين امر مشهود مي باشد

```
f := (b+c) *( d-e);
IF (f >0) AND (f<5) THEN
  a[f] := 1;
END_IF;
```

Where variable "f" has an INT data type.

Example 6

```
 a[b[1]]:= c;
a[b[2] + 3]:= c;
```

اين مثال نشان مي دهد كه براي تعيين انديس يك متغير آرايه اي ، مي توانيد از متغير آرايه اي ديگري استفاده نماييد

جدولي از دستورالعملهاي محاسباتي و عددي

Function	Name	Argument data type	Return value type	Operation	Example
ABS(argument)	Absolute value	INT, DINT, LINT, UINT, UDINT, ULINT, REAL, LREAL	NT, DINT, LINT, UINT, UDINT, ULINT, REAL, LREAL	| argument |	a:=ABS(b)
SQRT(argument)	Square root	REAL, LREAL	REAL, LREAL		a:=SQRT(b)
LN(argument)	Natural logarithm	REAL, LREAL	REAL, LREAL		a:=LN(b)
LOG(argument)	Common logarithm	REAL, LREAL	REAL, LREAL		a:=LOG(b)
EXP(argument)	Natural exponential	REAL, LREAL	REAL, LREAL		a:=EXP(b)
SIN(argument)	Sine	REAL, LREAL	REAL, LREAL	SIN(argument)	a:=SIN(b)
COS(argument)	Cosine	REAL, LREAL	REAL, LREAL	COS(argument)	a:=COS(b)
TAN(argument)	Tangent	REAL, LREAL	REAL, LREAL	TAN(argument)	a:=TAN(b)
ASIN(argument)	Arc sine	REAL, LREAL	REAL, LREAL		a:=ASIN(b)
ACOS(argument)	Arc cosine	REAL, LREAL	REAL, LREAL		a:=ACOS(b)
ATAN(argument)	Arc tangent	REAL, LREAL	REAL, LREAL		a:=ATAN(b)
EXPT(base, exponent)	Exponential	Base: REAL, LREAL Exponent: INT, DINT, LINT, UINT, UDINT, ULINT	REAL, LREAL		a:=EXPT(b, c)