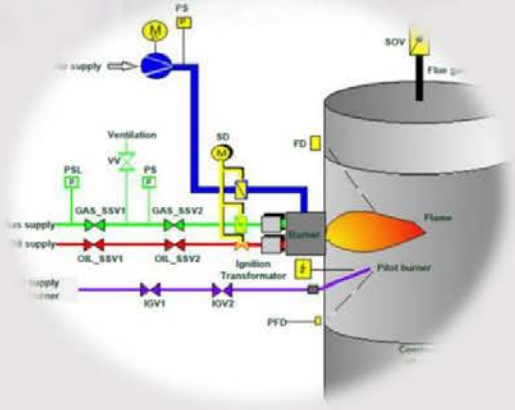
PLC S7-300.400

LEVEL(II)



مهندس میثـم زارع
مهندس مهرنوش اصغری

SIEMENS

بلوک های

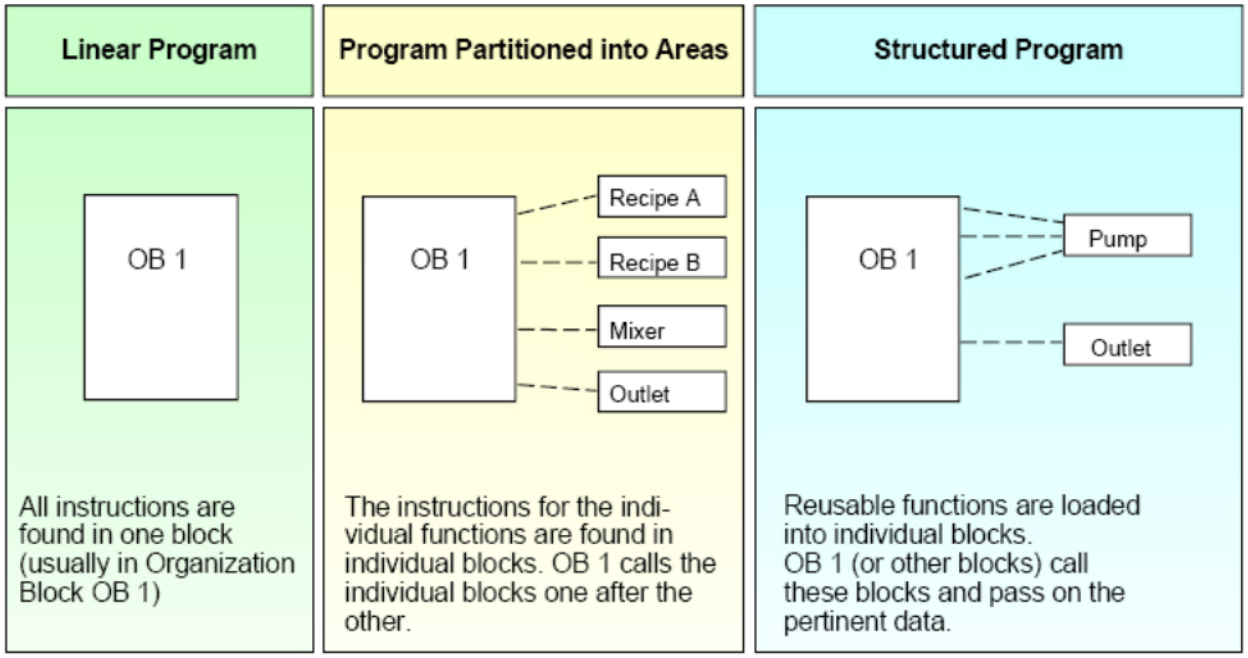
برنامه نویسی

Meysam.zare.4@gmail.com
Mehrnoush_asghari@yahoo.com

همانطور که میدانیم با RUNشدن CPU اگر بلوکهای Startup وجود نداشته باشد، CPUبه سراغ بلوک OB1 آمده و به صورت پیوسته برنامه نوشته شده در OB1 را اجرا می کند.

اگرچه PLC های شرکت زیمنس همگی دارای بلوک اصلی برنامه نویسی OB1 هستند اما علاوه بر آن بلوک های متفاوتی برای کاربردهای مختلف در نظر گرفته شده است،که در این فصل به معرفی این بلوک ها می پردازیم.

1-انواع ساختار های برنامه نویسی



الف)ساختار برنامه نویسی خطی

در این روش تمام برنامه مربوط به یک پروسه در بلوکOB1 پشت سر هم نوشته میشود.

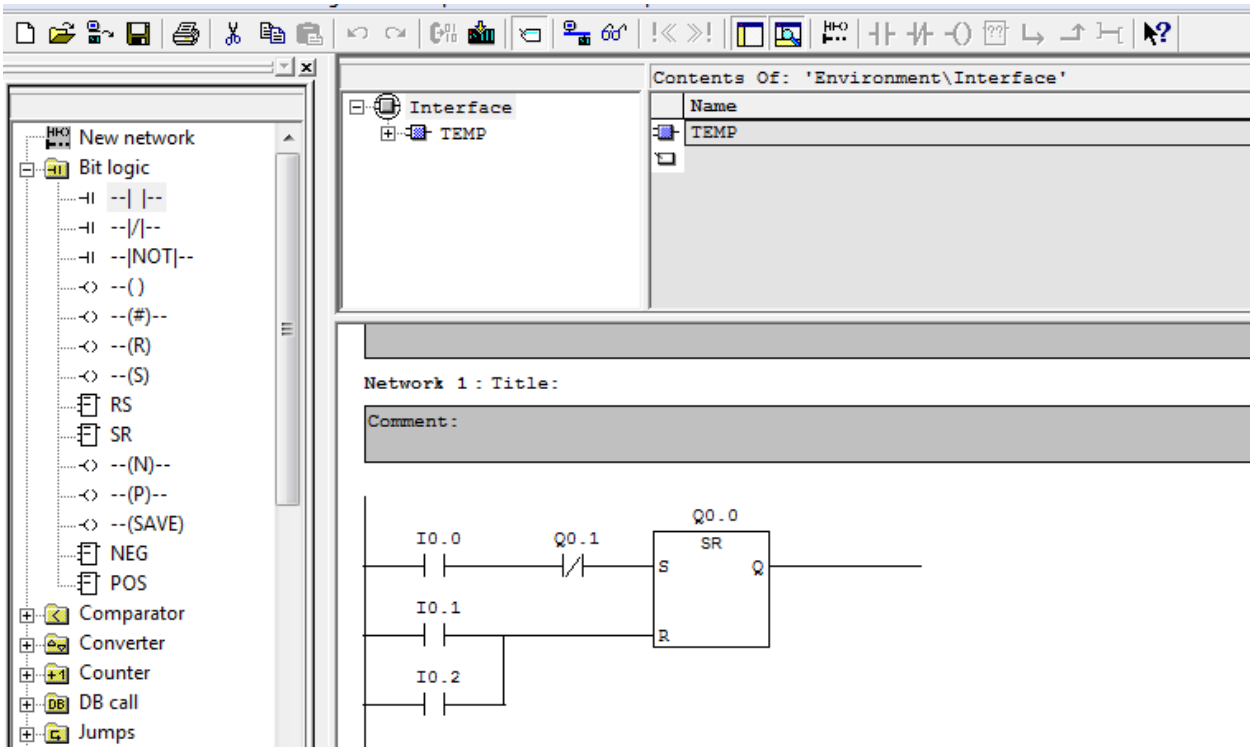
این روش برای برنامه های کوچک پیشنهاد می شود ولی در عمل و در پروژه های صنعتی به دلیل حجم بالای برنامه نوشتن تمامی برنامه در بلوک OB1 اصلا کار منطقی نمی باشد.

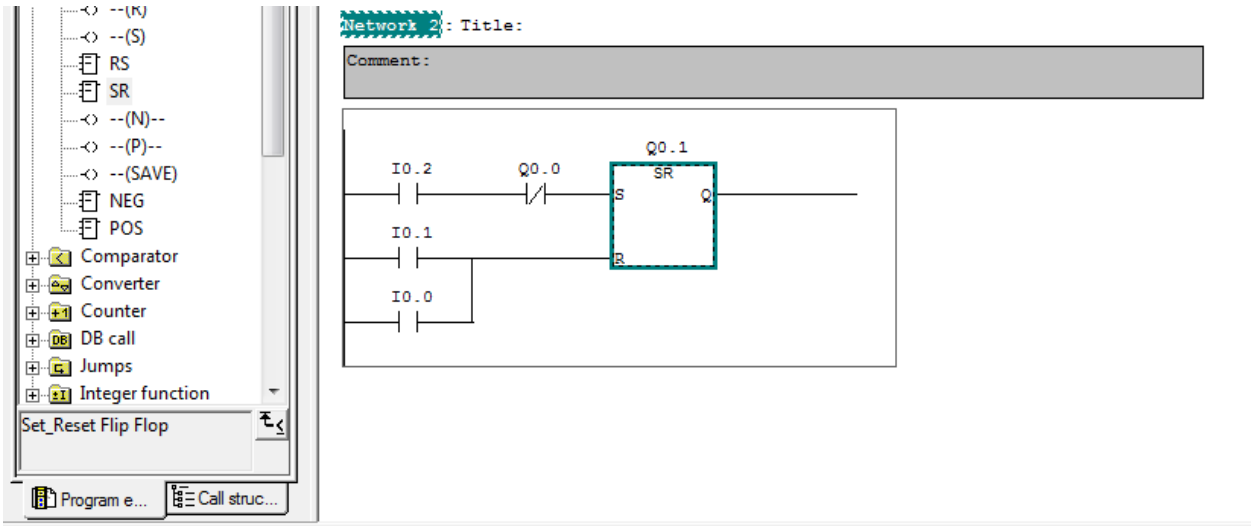
مشکلات این روش:

- عیب یابی دشوار
- مشکل بودن اضافه کردن برنامه به برنامه قبلی

- عدم کنترل کافی روی قسمت های مختلف برنامه

در شکل های زیر یک نمونه برنامه نویسی به صورت خطی که برنامه مربط به یک موتور چپگرد-راستگرد را نشان میدهد، مشاهده میکنید.





ب) ساختار برنامه نویسی تقسیم شده

در این روش برنامه مربوط به قسمت های مختلف یک پروسه در بلوک های مختلف نوشته می شود. این بلوکها به عنوان زیر برنامه مورد استفاده قرار میگیرند.

به عنوان مثال یک خط تولید که دارای بخش هایی مانند دستگاه پرس،کوره می باشد را در نظر بگیرید.

برای هر بخش نیاز به برنامه جدا داریم.

بنابراین برنامه قسمت دستگاه پرس در یک بلوک و برنامه کوره را در یک بلوک دیگر می نویسیم.

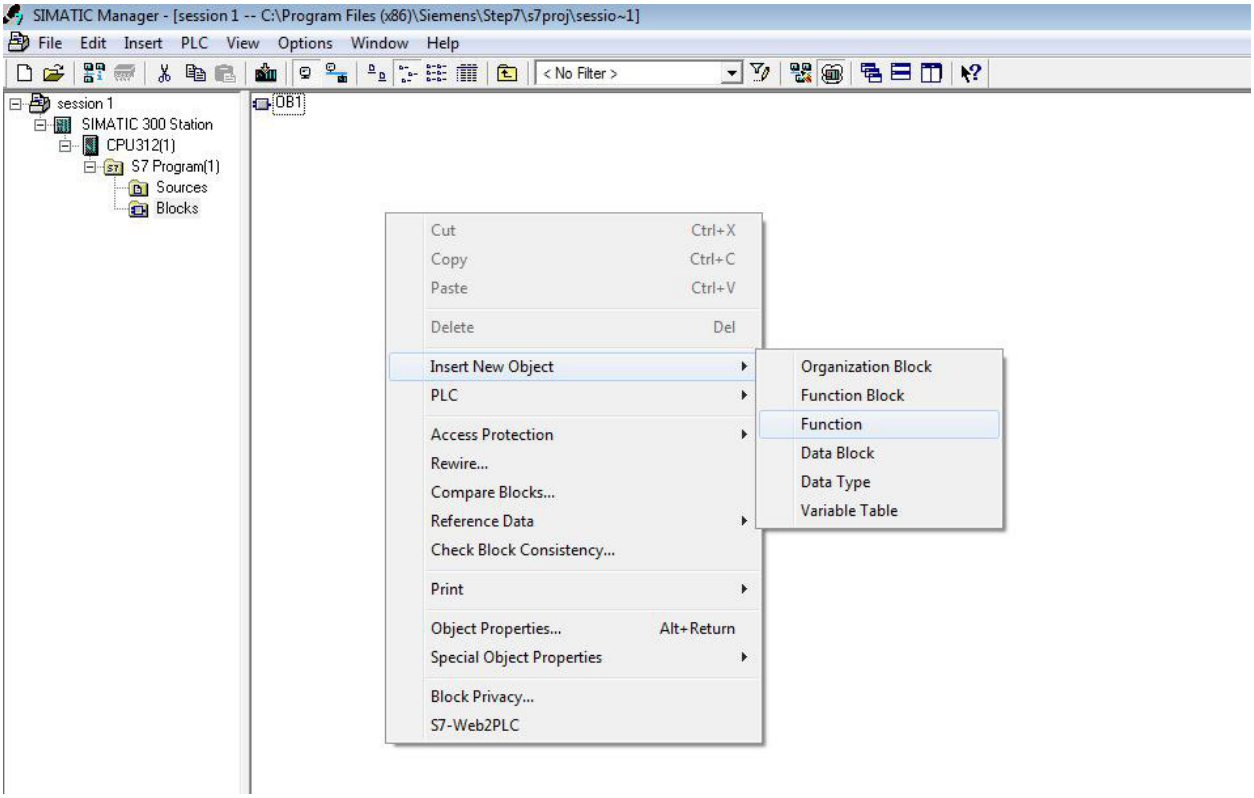
بلوک هایی که بدین منظور استفاده می شوند در PLC های S7 ، بلوک های FB-FC می باشند

مزایای استفاده از این روش:

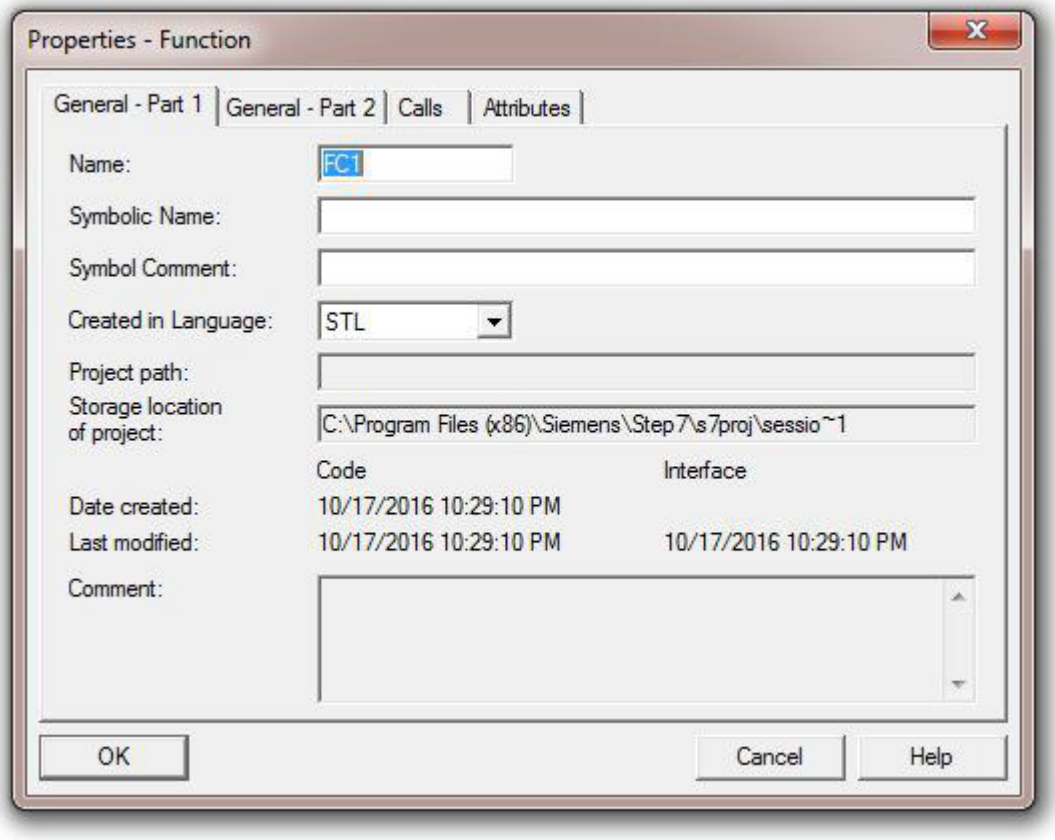
- عیب یابی سریعتر و راحت تر
- سهولت در اضافه کردن برنامه به برنامه قبلی
- کنترل راحت بر روی قسمت های مختلف برنامه

بلوک FC

برای ساخت بلوک های FC به مسیر زیر مراجعه کنید.

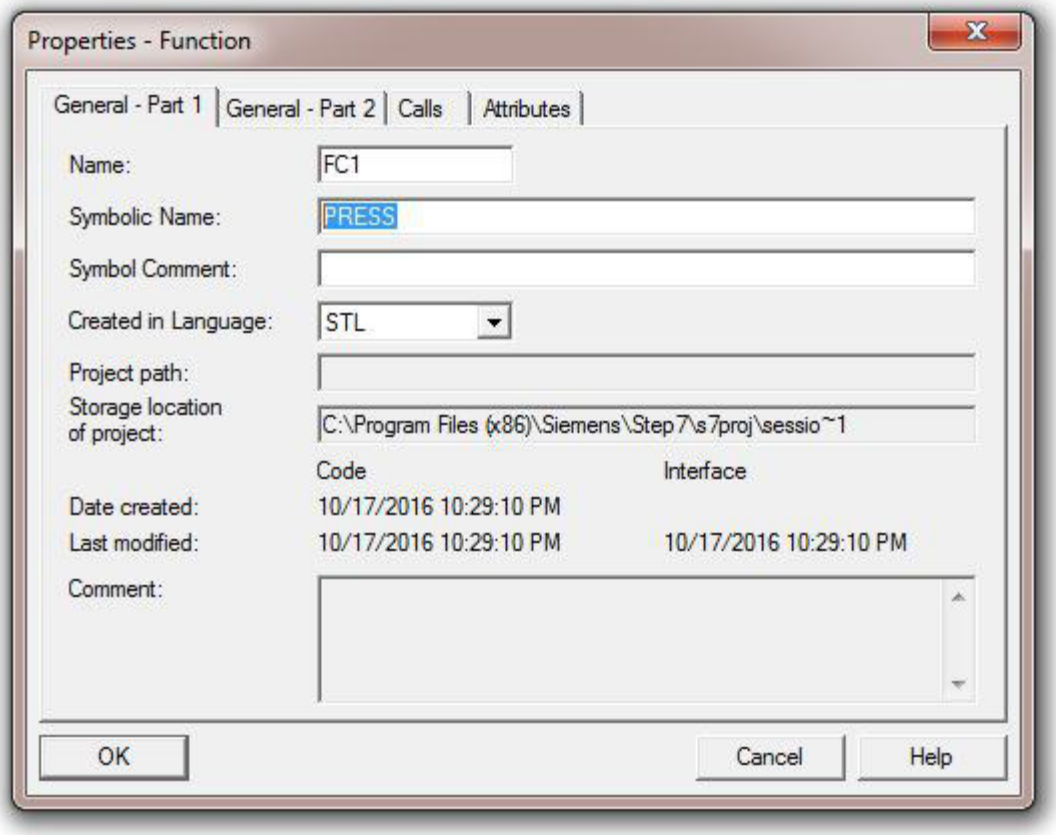


با کلیک بر روی گزینه مشخص شده در شکل صفحه قبل، پنجره شکل زیر نمایان می شود.

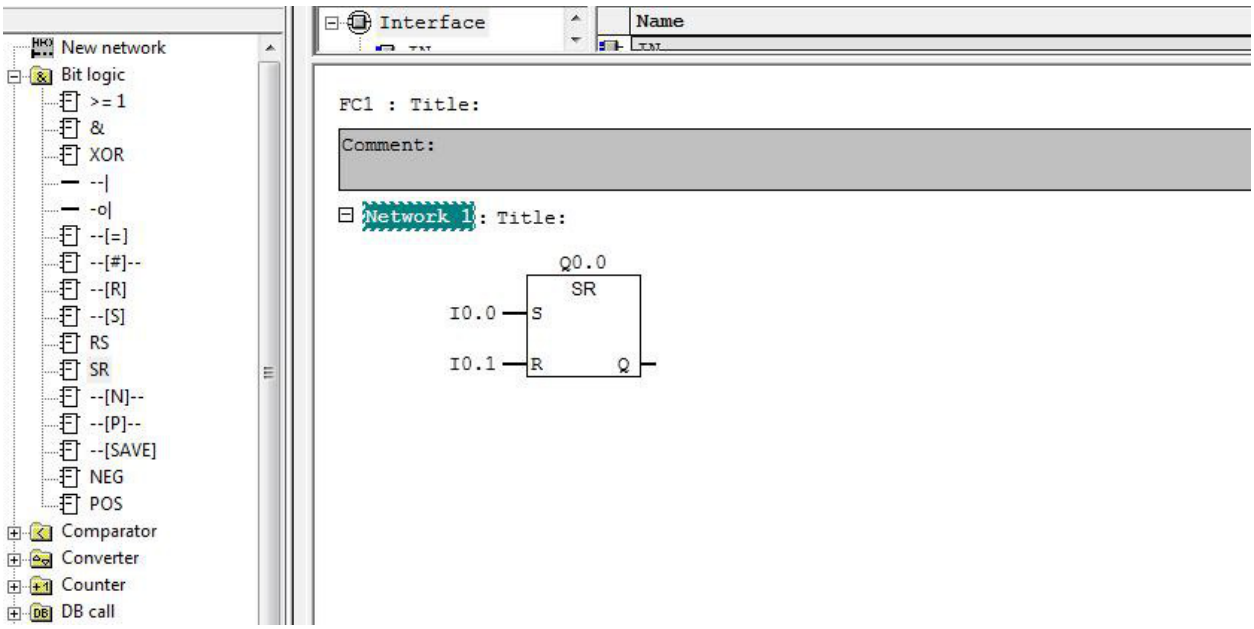


حال در مثال خط تولید با داشتن دو دستگاه پرس و کوره را بررسی میکنیم.فرض کنید میخواهیم برنامه مربوط به دستگاه پرس را در بلوک FC1 بنویسیم.

بعد از نمایان شدن پنجره فوق،در قسمت Symbolic Name عبارت PRESS را مطابق شکل زیر وارد می کنیم

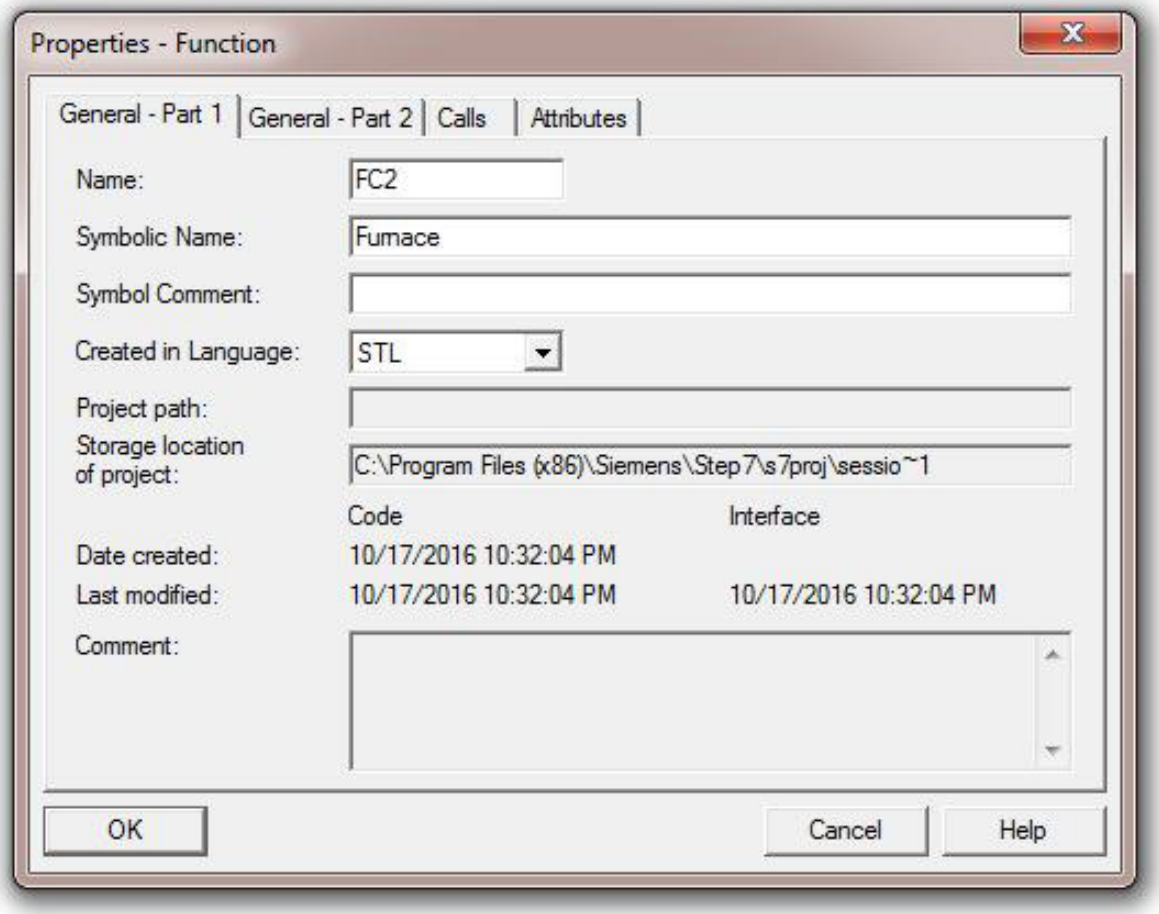


وارد بلوک FC1 می شویم و برنامه مربوط به دستگاه پرس را مینویسیم.

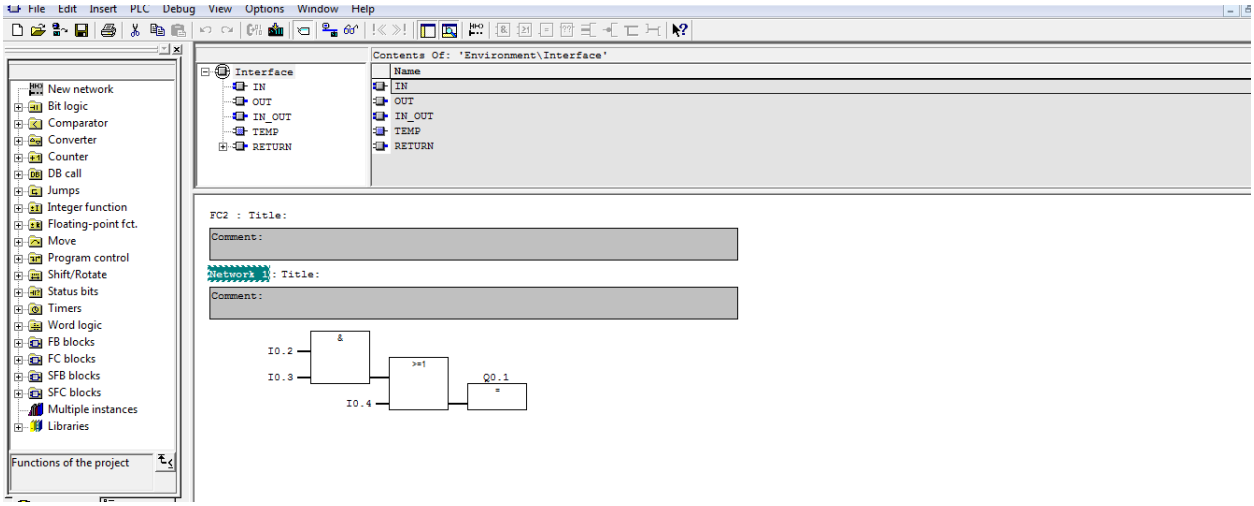


این بلوک را ذخیره کرده و وارد محیط اصلی برنامه میشویم و با تکرار مراحل بیان شده یک بلوک FC دیگر به نام FC2 برای کوره ایجاد میکنیم.

نام این بلوک را Furnace می گذاریم.



پس از وارد شدن به محیط FC2 برنامه مربوط به کوره را در آن می نویسیم و سپس آن را ذخیره می کنیم.



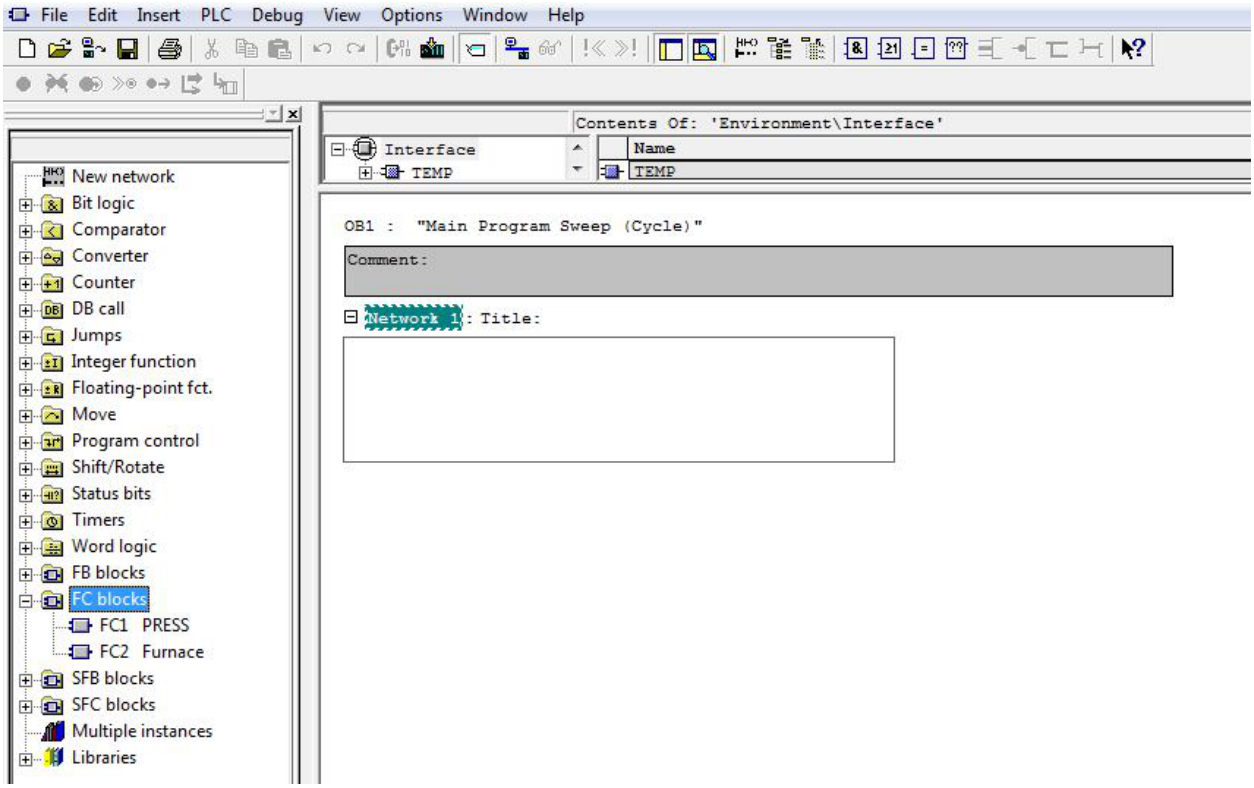
نکته: با کمی دقت در آدرس های داده شده در دو بلوک مشاهده میکنید که آدرسها کاملا مجزا از یکدیگرند.

در ادامه حتما باید بلوک های FC در OB1 فراخوانی شوند

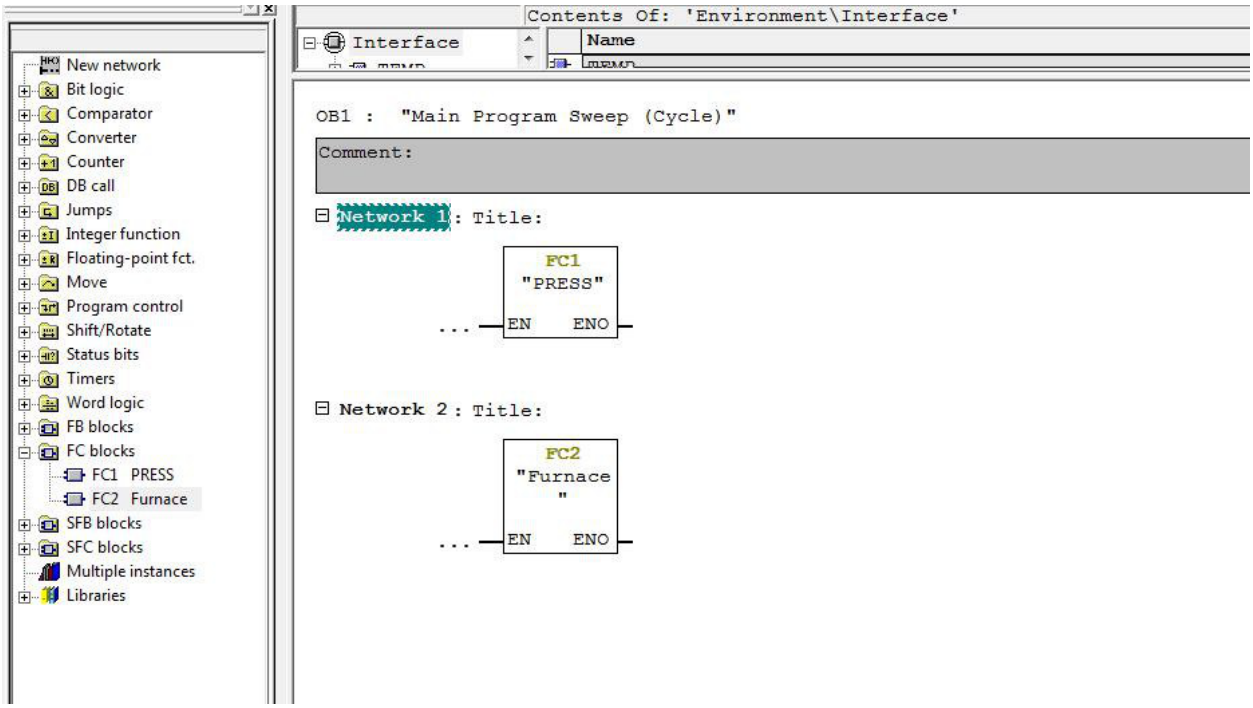
اگر این بلوک ها در OB1 فراخوانی نشوند، CPUبه هیچ وجه این بلوک ها را پردازش نمی کند

برای این منظور به محیط اصلی برنامه بازگشته و وارد OB1 میشویم.

به ستون سمت چپ و قسمت FC blocks مراجعه میکنیم.(شکل زیر)



همانطور که مشاهده میکنید هر دو بلوک FC1 و FC2 در زیرمجموعه این گزینه هستند.هر کدام را در یک Network وارد میکنیم.

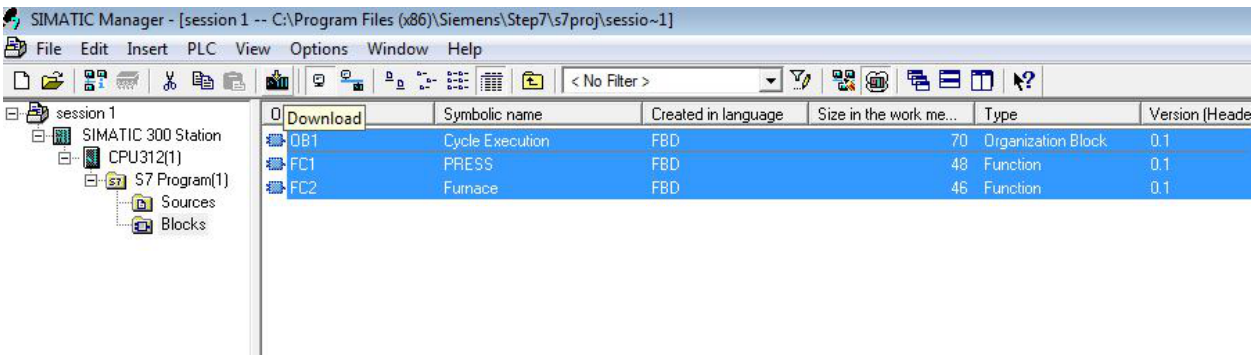


اگر ورودی EN خالی گذاشته شود، یعنی نوع فراخوانی غیرشرطی است و این بدان معناست که مادامی که CPU در وضعیت RUN قرار گیرد، بدون هیچ گونه شرطی برنامه هر دو بلوک را می خواند

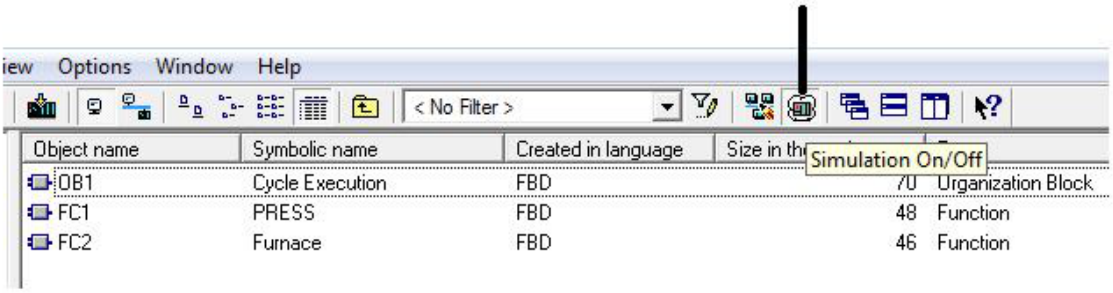
برای این مثال مشکلی ایجاد نمی شود، به این دلیل که برنامه ها اصلا به هم ارتباطی ندارند و مربوط به دو قسمت مجزا می باشد

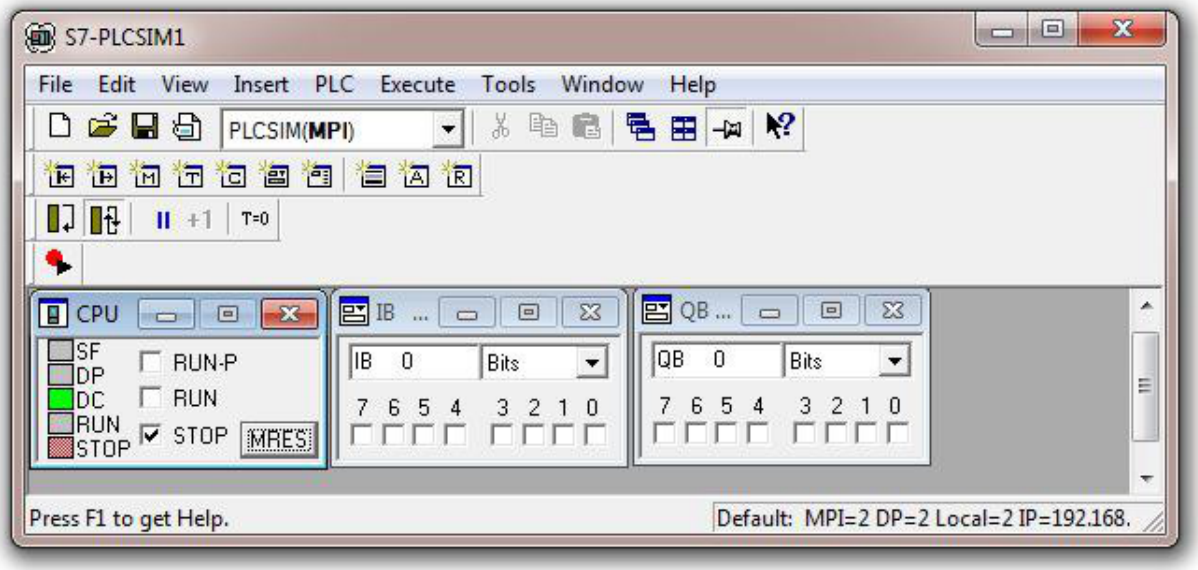
در ادامه برای اجرا، باید در محیط اصلی برنامه هر سه بلوک را انتخاب کنبم و بر روی گزینه Download کلیک کنید.

البته راه دیگر این است که گزینه Blocks را انتخاب وسپس بر روی گزینه Download کلیک نمایید.



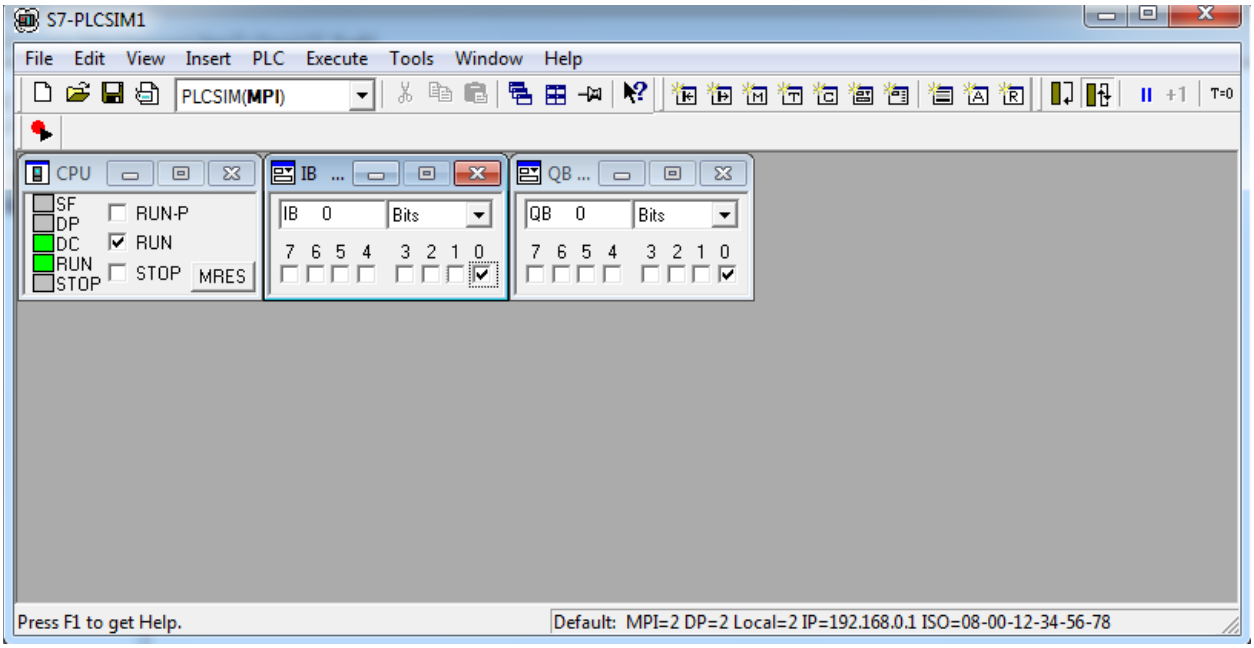
لازم به ذکر است که قبل از کلیک بر روی گزینه Download بایستی محیط PLCSIM باز شده باشد و همچنین شبیه ساز باید در وضعیت Stop باشد.

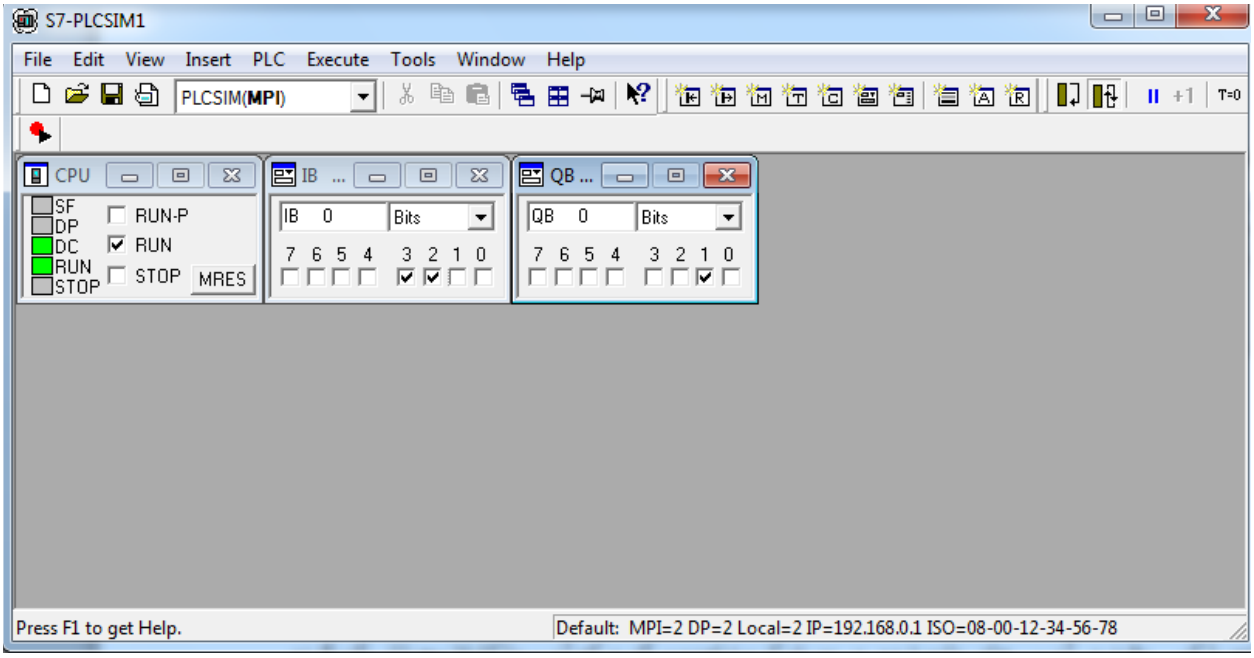


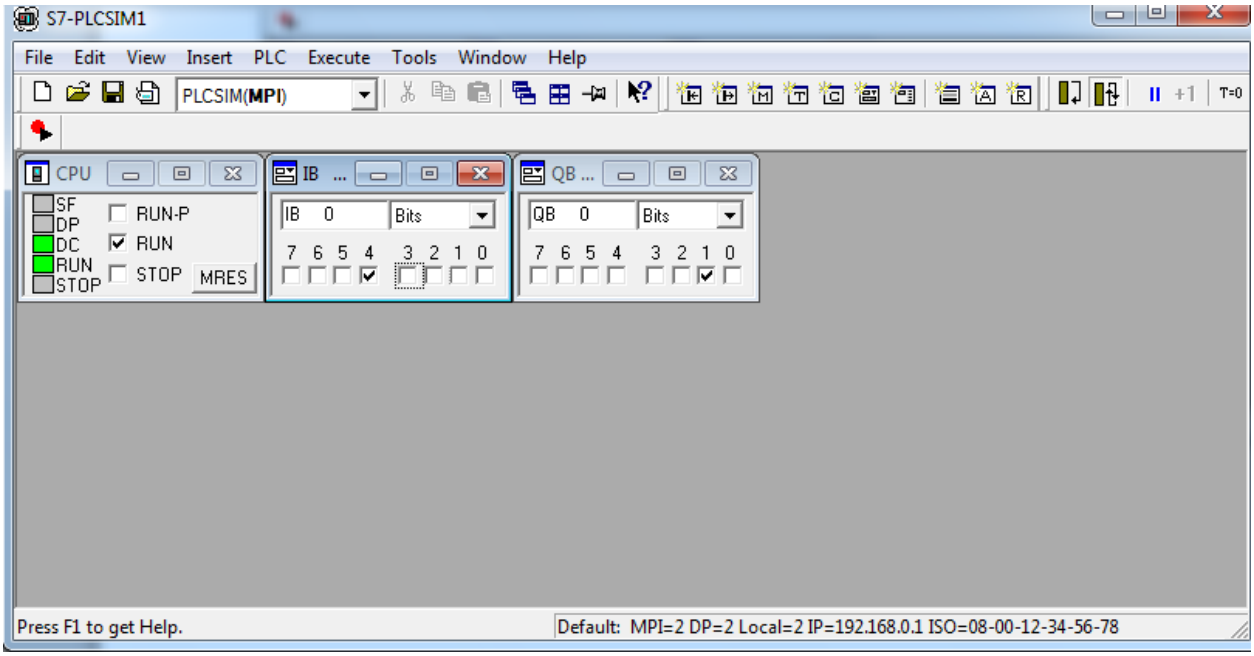


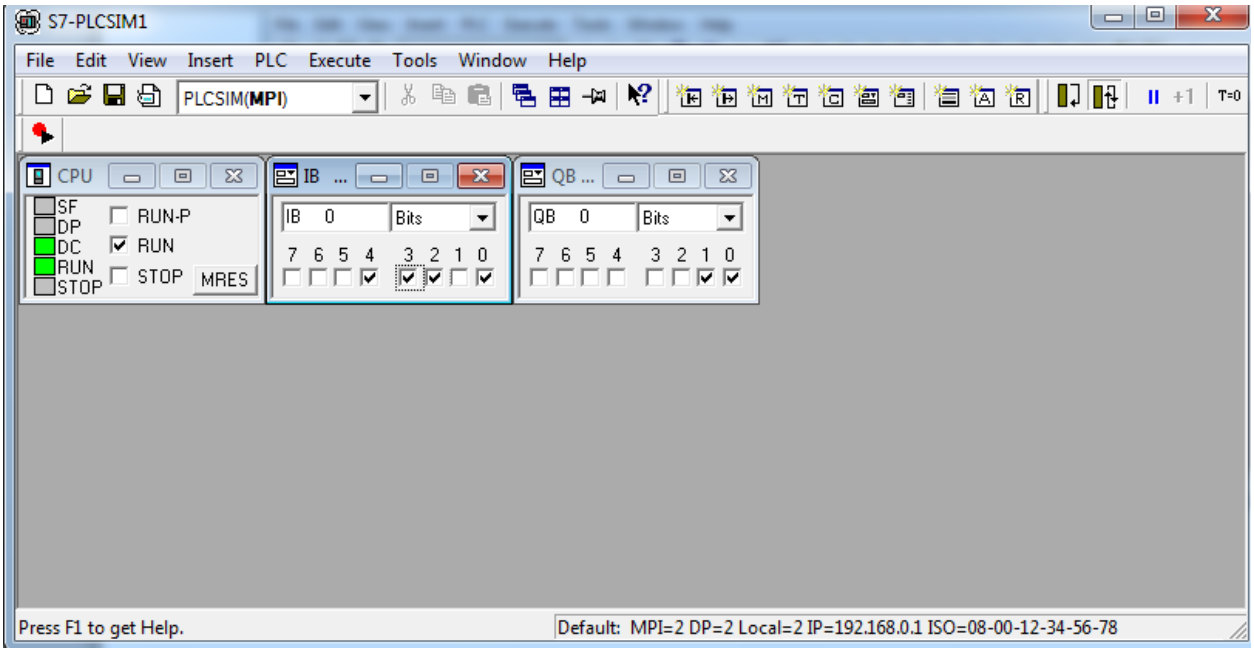
برای تست این برنامه یک بایت ورودی یک بایت خروجی لازم است.

حالا با توجه به برنامه نوشته شده می توان ورودی ها را فعال و برنامه را تست کرد.


S7-PLCSIM1
File Edit View Insert PLC Execute Tools Window Help
PLCSIM(MPI)
CPU
SF
DP
DC
RUN
STOP
RUN-P
RUN
STOP
MRES
IB ...
IB 0
Bits
7 6 5 4 3 2 1 0
QB ...
QB 0
Bits
7 6 5 4 3 2 1 0
Press F1 to get Help.
Default: MPI=2 DP=2 Local=2 IP=192.168.0.1 ISO=08-00-12-34-56-78


S7-PLCSIM1
File Edit View Insert PLC Execute Tools Window Help
PLCSIM(MPI)
CPU
SF
DP
DC
RUN
STOP
RUN-P
RUN
STOP
MRES
IB ...
IB 0
Bits
7 6 5 4 3 2 1 0
QB ...
QB 0
Bits
7 6 5 4 3 2 1 0
Press F1 to get Help.
Default: MPI=2 DP=2 Local=2 IP=192.168.0.1 ISO=08-00-12-34-56-78

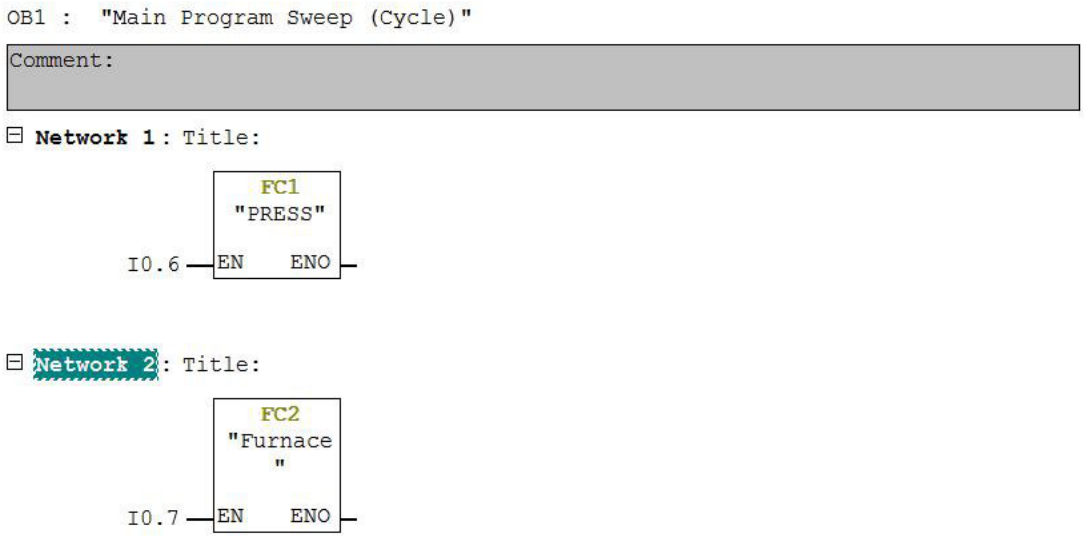




حال فرض کنید می خواهیم برنامه هر قسمت توسط یک بیت وارد مدار شود یعنی برای اجرا شدن برنامه هر بخش از خط تولید فعال شدن بیت مربوطه الزامی باشد.

فراخوانی در این حالت فراخوانی شرطی نامیده می شود.

بدین منظور،برنامه بلوک های FC1 و FC2 بدون تغییر می مانند ولی برنامه بلوک OB1 به شکل زیر تغییر می کند.



یعنی برای اجرا شدن برنامه دستگاه پرس حتما لازم است ورودی I0.6 فعال شود و برای اجرا شدن برنامه FC2 یا برنامه کوره نیز باید I0.7 فعال شود.

اگر این ورودی ها فعال نشوند، هیچ برنامه ای اجرا نمی شود.

در این مثال اگر هر دو وروردی با هم فعال شوند ، باز مشکلی پیش نمی آید چون آدرس ها و منطق ها متفاوت می باشد.

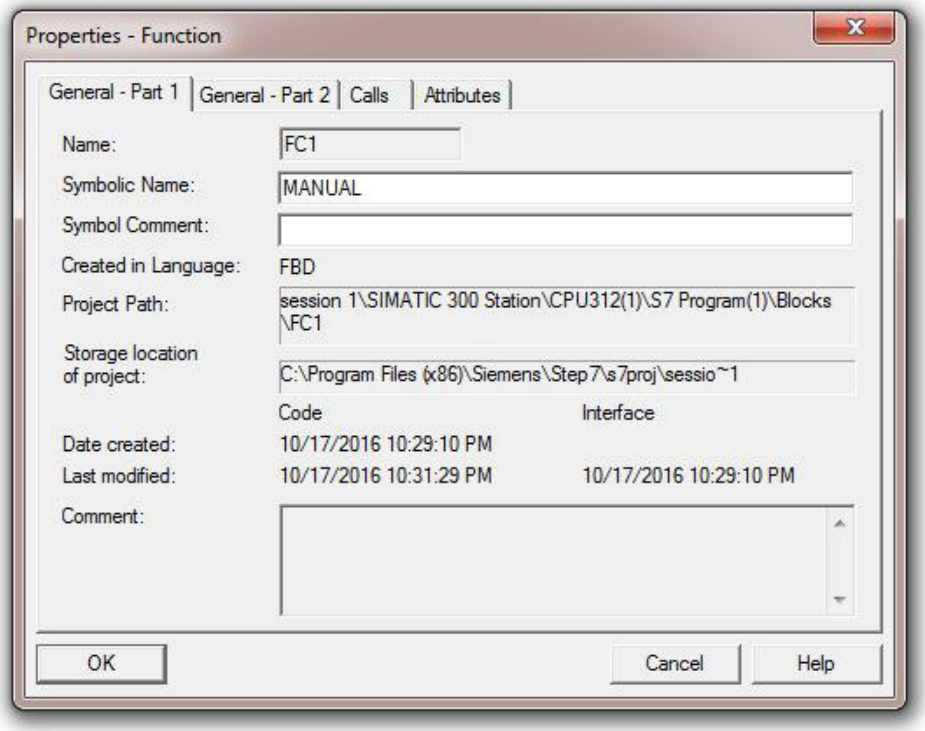
این شرط می تواند توسط HMI کنترل شود.

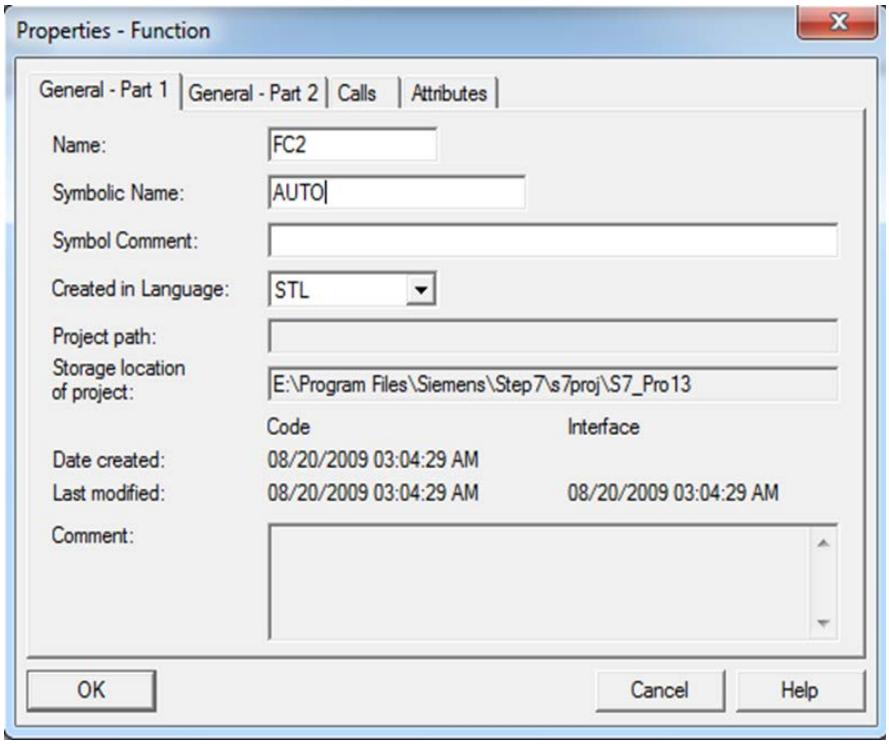
مثال) فرض کنید یک موتور در مد دستی بدون در نظر گرفتن زمان و توسط دو تا شاستی روشن و خاموش می شود و در مد اتوماتیک وقتی استارت زده میشود، موتور روشن و 10 ثانیه بعد خاموش می شود.

دقت کنند در این مثال آدرس ها در دو بلوک مشابه می باشند چون برنامه مربوط به یک سیستم می باشد.

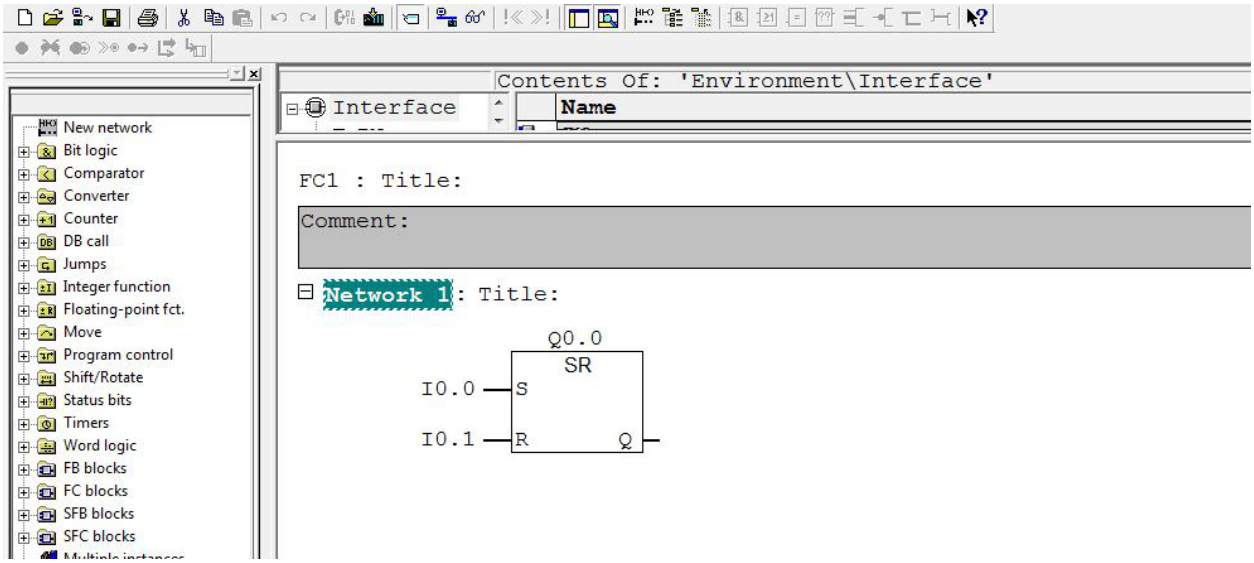
ابتدا در پروژه جدید دو بلوک FC بسازید.

نام یکی را AUTO و نام دیگری را MANUAL در نظر بگیرند.

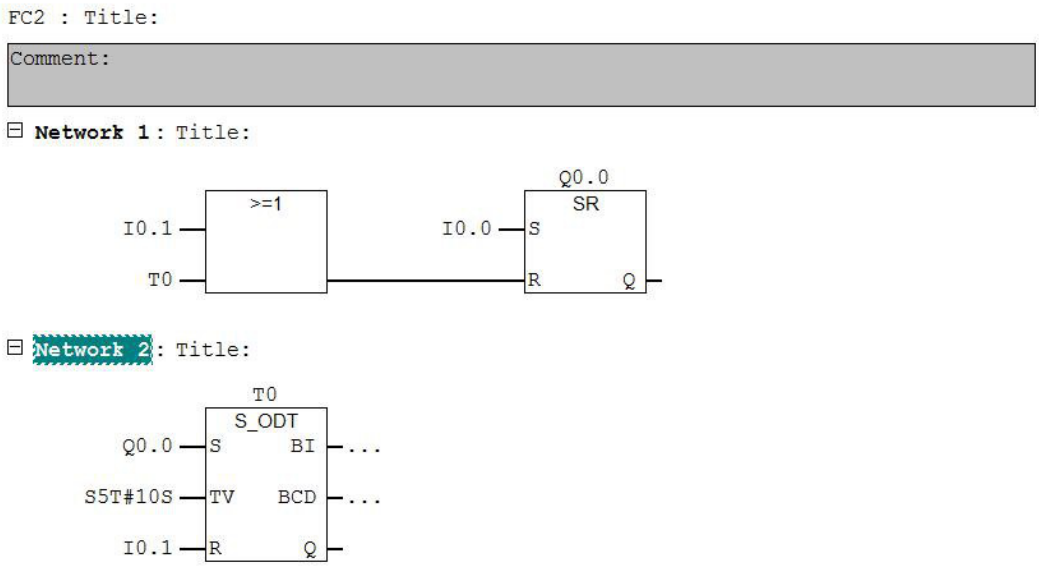

Properties - Function
General - Part 1
General - Part 2
Calls
Attributes
Name:
FC1
Symbolic Name:
MANUAL
Symbol Comment:
Created in Language:
FBD
Project Path:
session 1\SIMATIC 300 Station\CPU312(1)\S7 Program(1)\Blocks\FC1
Storage location of project:
C:\Program Files (x86)\Siemens\Step7\s7proj\sessio~1
Code
Interface
Date created:
10/17/2016 10:29:10 PM
Last modified:
10/17/2016 10:31:29 PM
10/17/2016 10:29:10 PM
Comment:
OK
Cancel
Help


Properties - Function
General - Part 1
General - Part 2
Calls
Attributes
Name:
FC2
Symbolic Name:
AUTO
Symbol Comment:
Created in Language:
STL
Project path:
Storage location of project:
E:\Program Files\Siemens\Step7\s7proj\S7_Pro13
Code
Interface
Date created:
08/20/2009 03:04:29 AM
Last modified:
08/20/2009 03:04:29 AM
08/20/2009 03:04:29 AM
Comment:
OK
Cancel
Help

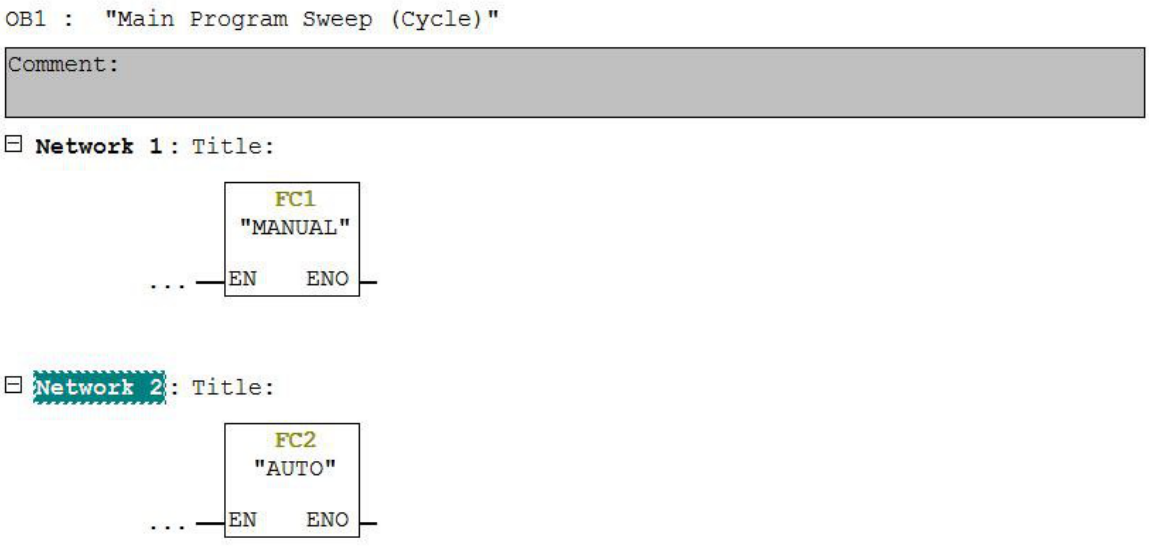
سپس وارد محیط بلوک FC1 شده و برنامه مربوط به مد دستی دستگاه را می نویسیم.



پس از ذخیره وارد محیط FC2 شده و برنامه مد اتوماتیک را می نویسیم.



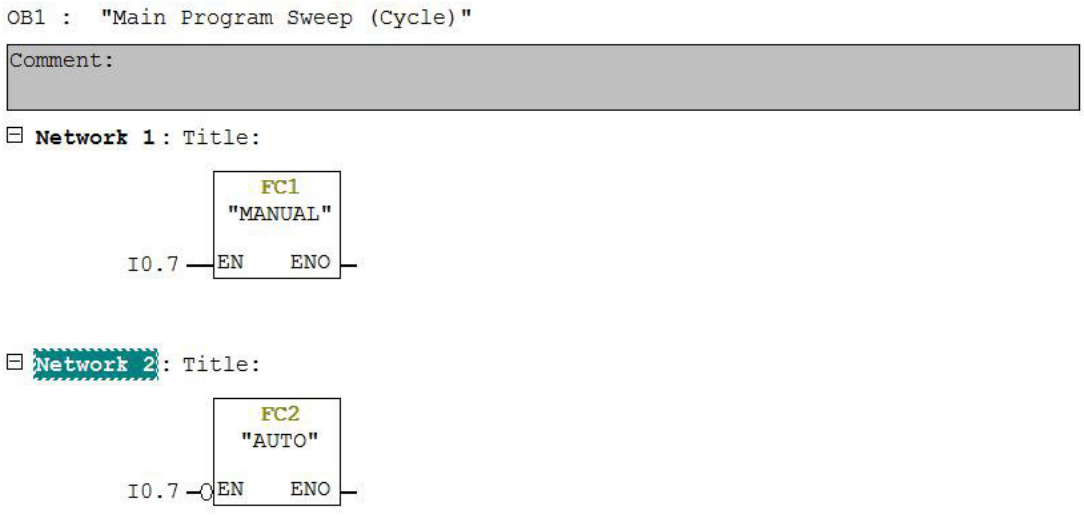
بعد از نوشتن برنامه و ذخیره آن، وارد محیط OB1 شوید و فرآیند فراخوانی زیربرنامه ها را انجام دهید.



در این مثال نمی توان ورودی های EN را خالی گذاشت زیرا هر دو بلوک اجرا می شوند و با توجه به صورت مثال امکان اینکه دستگاه هم در مد اتوماتیک باشد و هم دستی وجود ندارد.

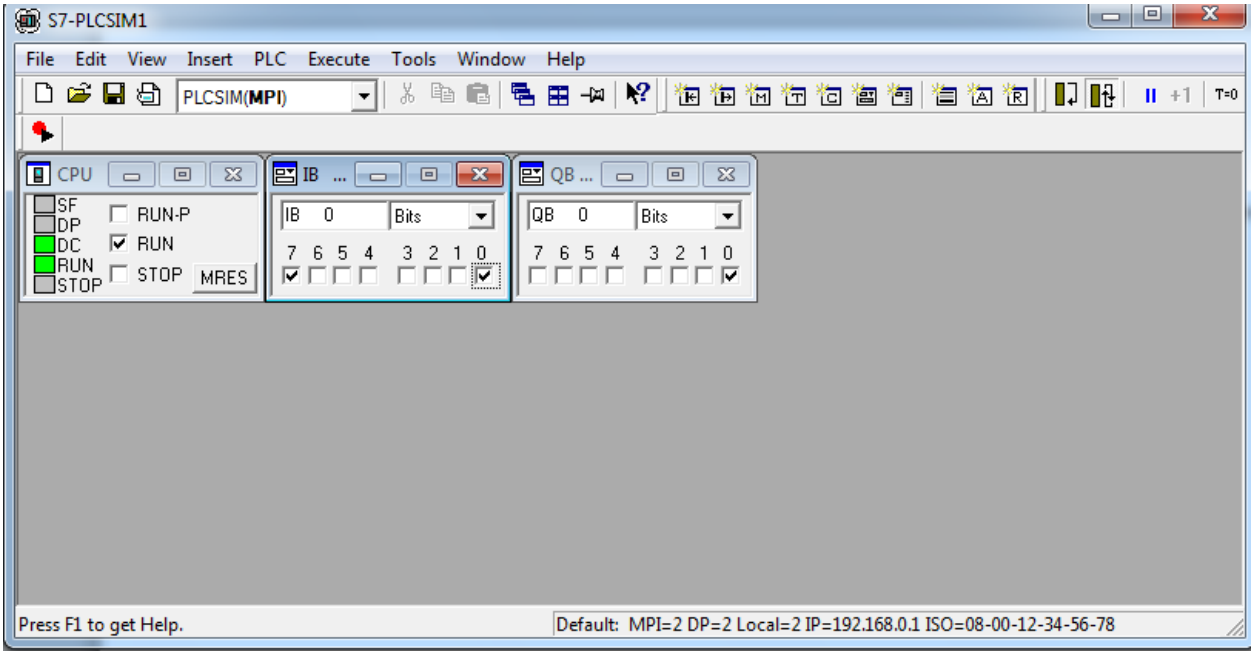
در این مثال در هر لحظه فقط می بایست یک بلوک اجرا شود.

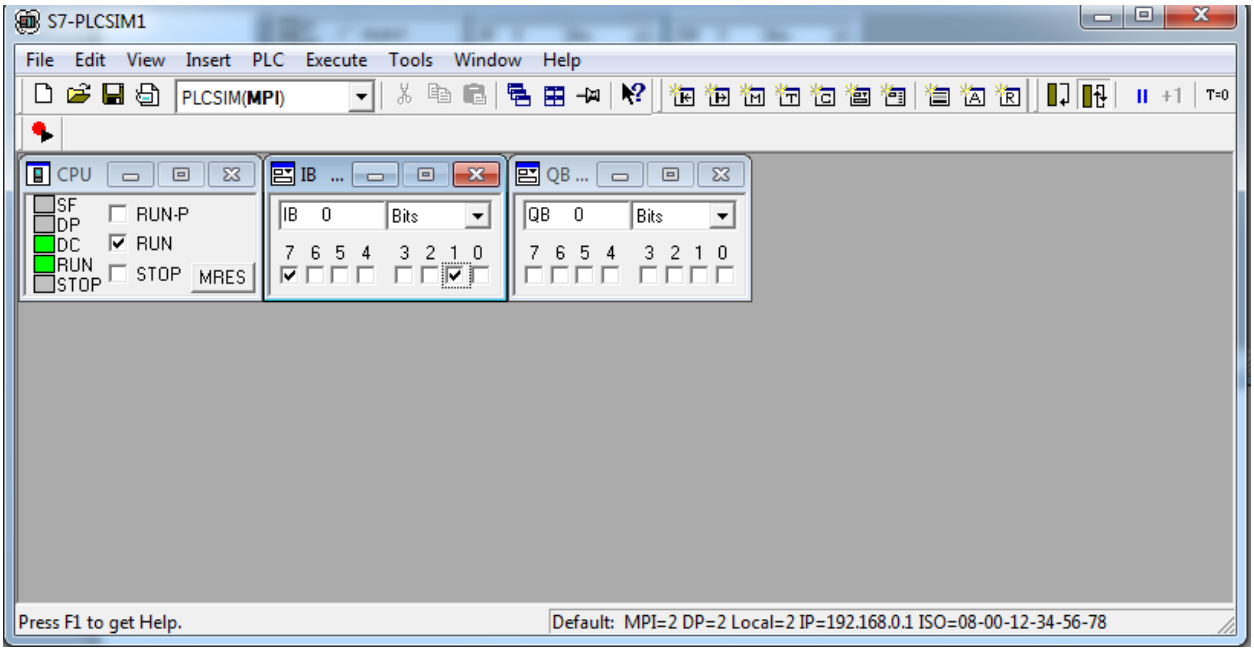
پس شرطی مانند شکل زیر باستی در نظر گرفته شود.

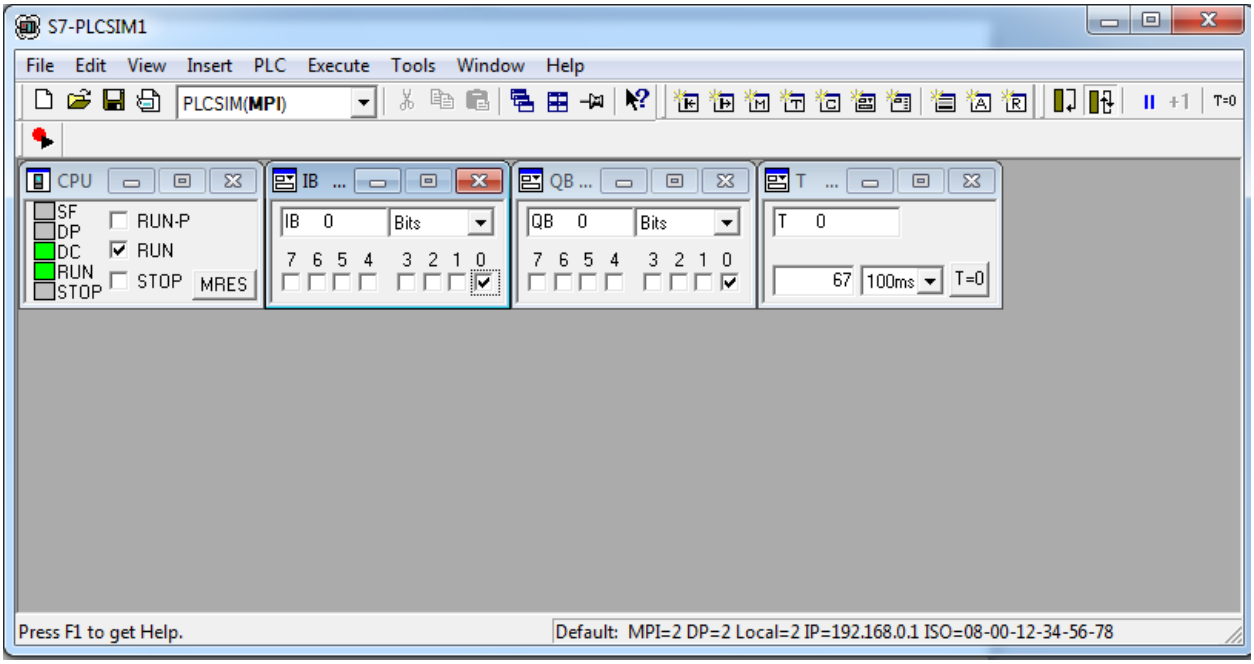


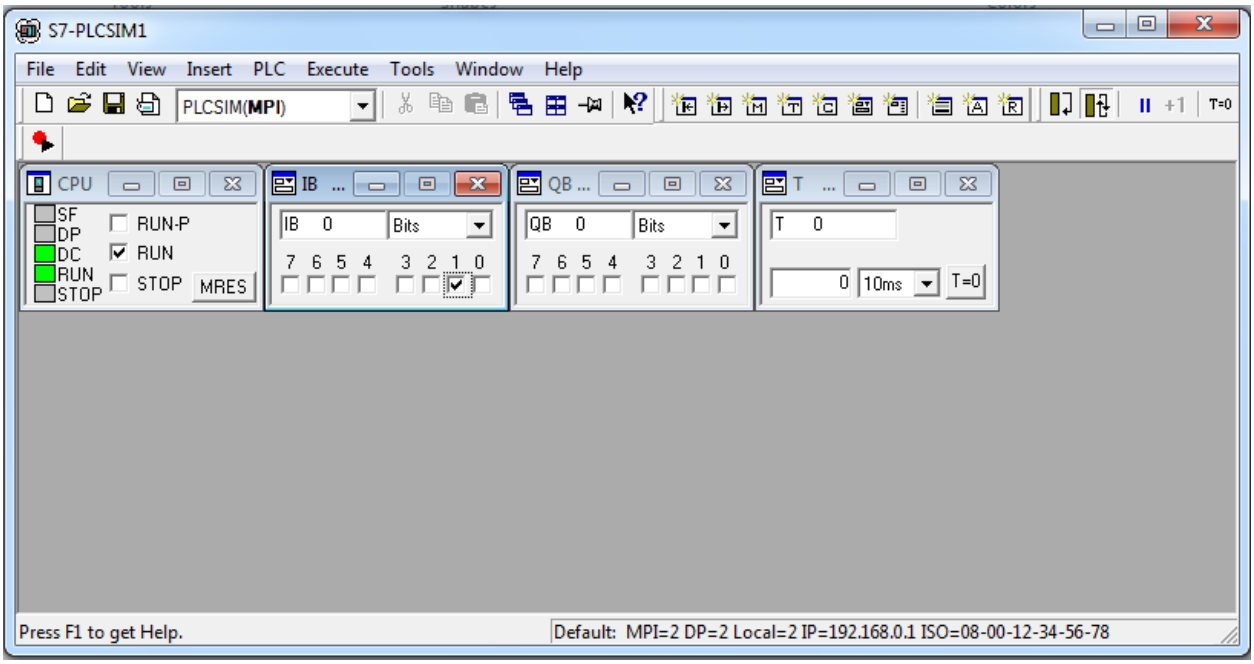
برای اینکه مطمئن شویم که هیچ گاه هر دو بلوک با هم اجرا نمی شوند، NOT یک ورودی را در ورودی EN بلوک بعدی قرار می دهیم.

سپس به محیط اصلی برنامه رفته و برنامه را Download میکنیم.









مشاهده می شود که موتور Q0.0 با دو برنامه متفاوت کنترل می شود.

در واقع نوع برنامه را وضعیت I0.7 مشخص می کند،یعنی با تغییر وضعیتI0.7 ، برنامه کنترلی برای موتور تغییر می کند.

پ) ساختار برنامه نویسی ساختار یافته

همانطور که ملاحظه شد در روش دوم آدرس ها در هر بلوک وارد می شوند و تنها برنامه مربوط به یک دستگاه در بلوک های مختلف پخش میگردد.

اما گاهی یک برنامه برای چندین درایو یکسان است بدین منظور نوشتن به کرات یک برنامه معقول به نظر نمی آید زیرا اگر قرار باشد برای هر درایو یا موتور یک بلوک اشغال کنیم حجم برنامه زیاد و زمان برنامه نویسی هم به همان نسبت افزایش می یابد.

در اینگونه موارد از برنامه نویسی ساختار یافته استفاده میکنیم.

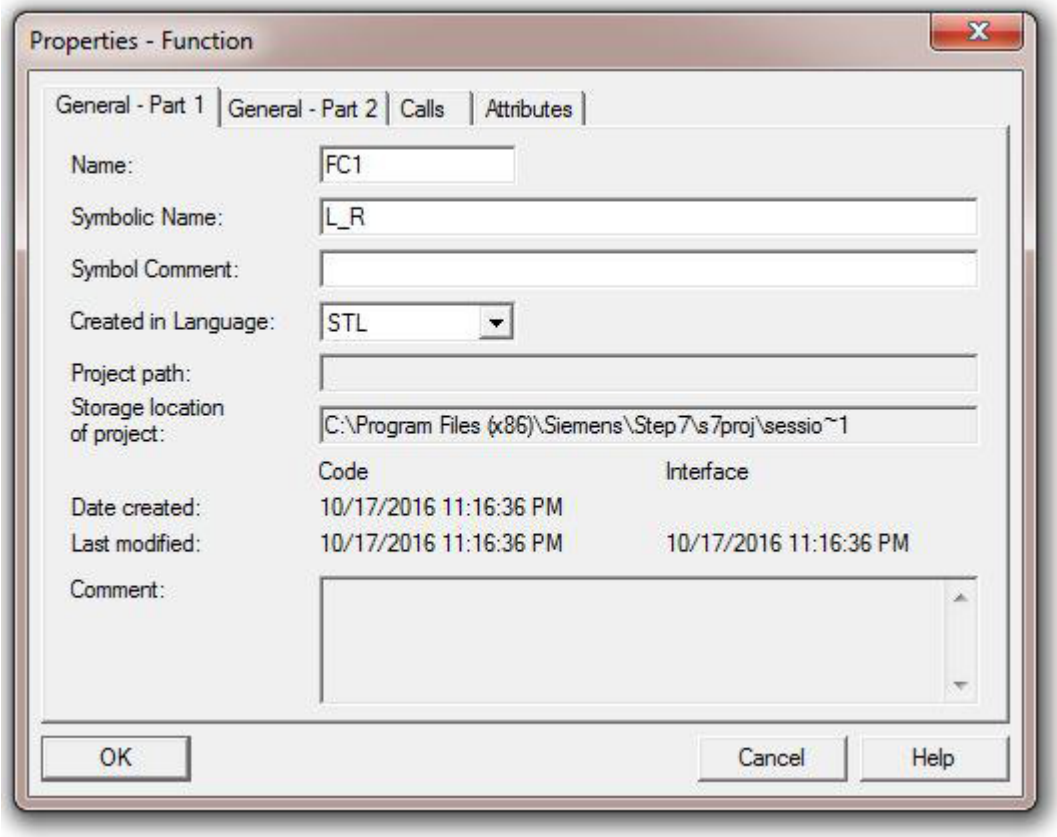
در این روش یک برنامه را یکبار می نویسیم و هر چند بار که بخواهیم در برنامه استفاده می کنیم.

نوشتن برنامه بدین صورت را تابع نویسی می گویند.

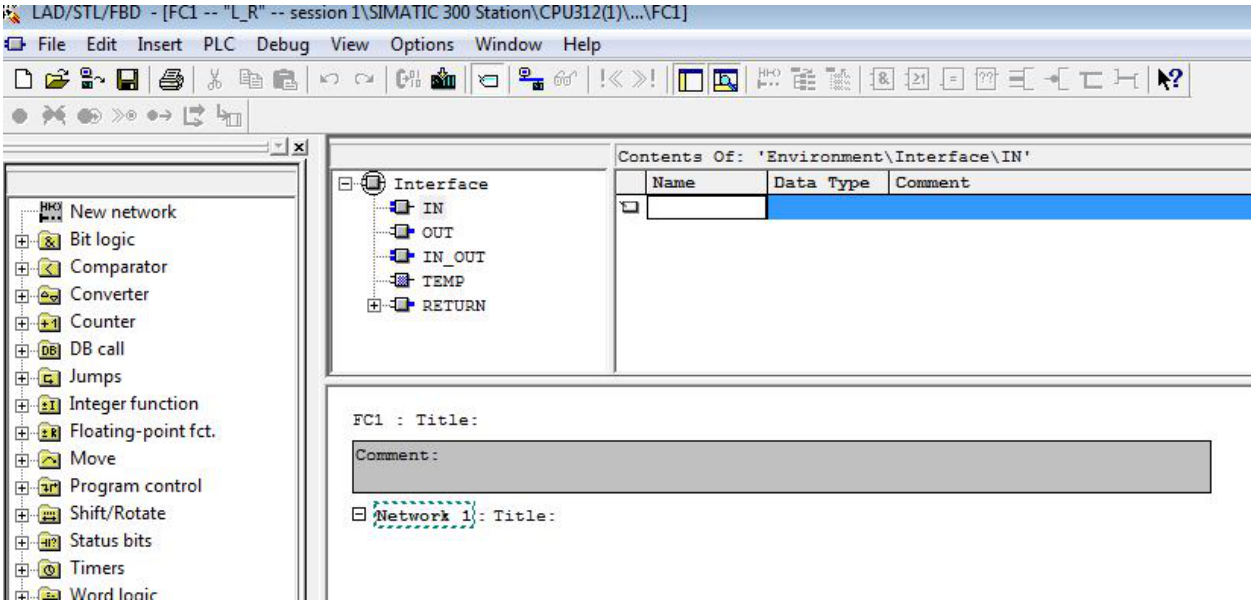
حال با ذکر یک مثال به توضیح بیشتر این روش می پردازیم.

مثال) فرض کنید می خواهیم یک تابع چپگرد راستگرد بنویسیم و از آن برای 5 عدد موتور استفاده کنیم.

ابتدا یه پروژه جدید ایجاد کرده و سپس مطابق مراحلی که قبلا گفته شد یک بلوک FC میسازیم.



پس از وارد شدن به محیط FC1 ، در قسمت بالای محیط، جدول پارامترهای تابع وجود دارد.



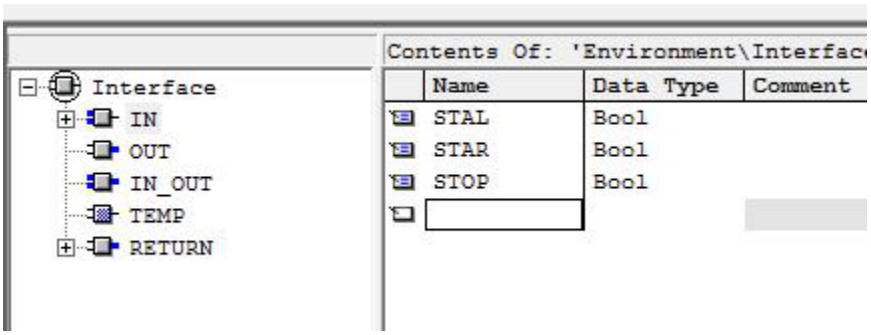
قبل از نوشتن برنامه باید مشخص کنیم که برنامه ای که قرار است بنویسیم چه تعداد I/O دارد.

در این مثال 3 ورودی و 2 خروجی داریم.

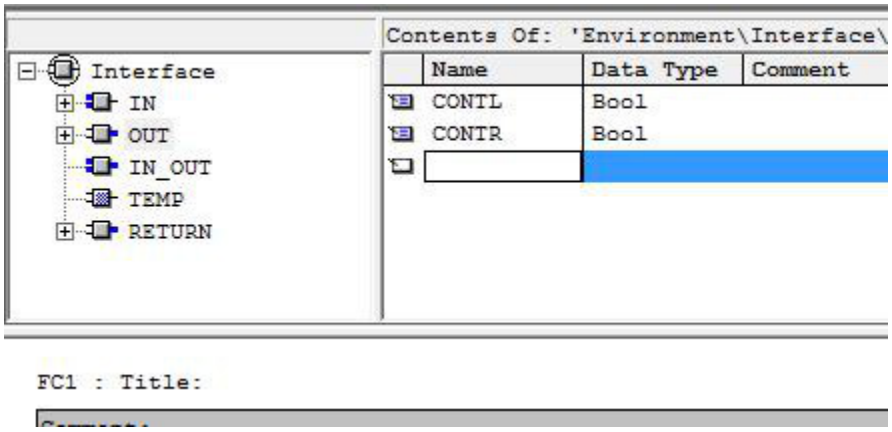
ورودیها: ورودی استارت چپگرد،ورودی استارت راستگرد وStop

خروجی ها: خروجی کنتاکتور چپگرد و خروجی کنتاکتور راستگرد

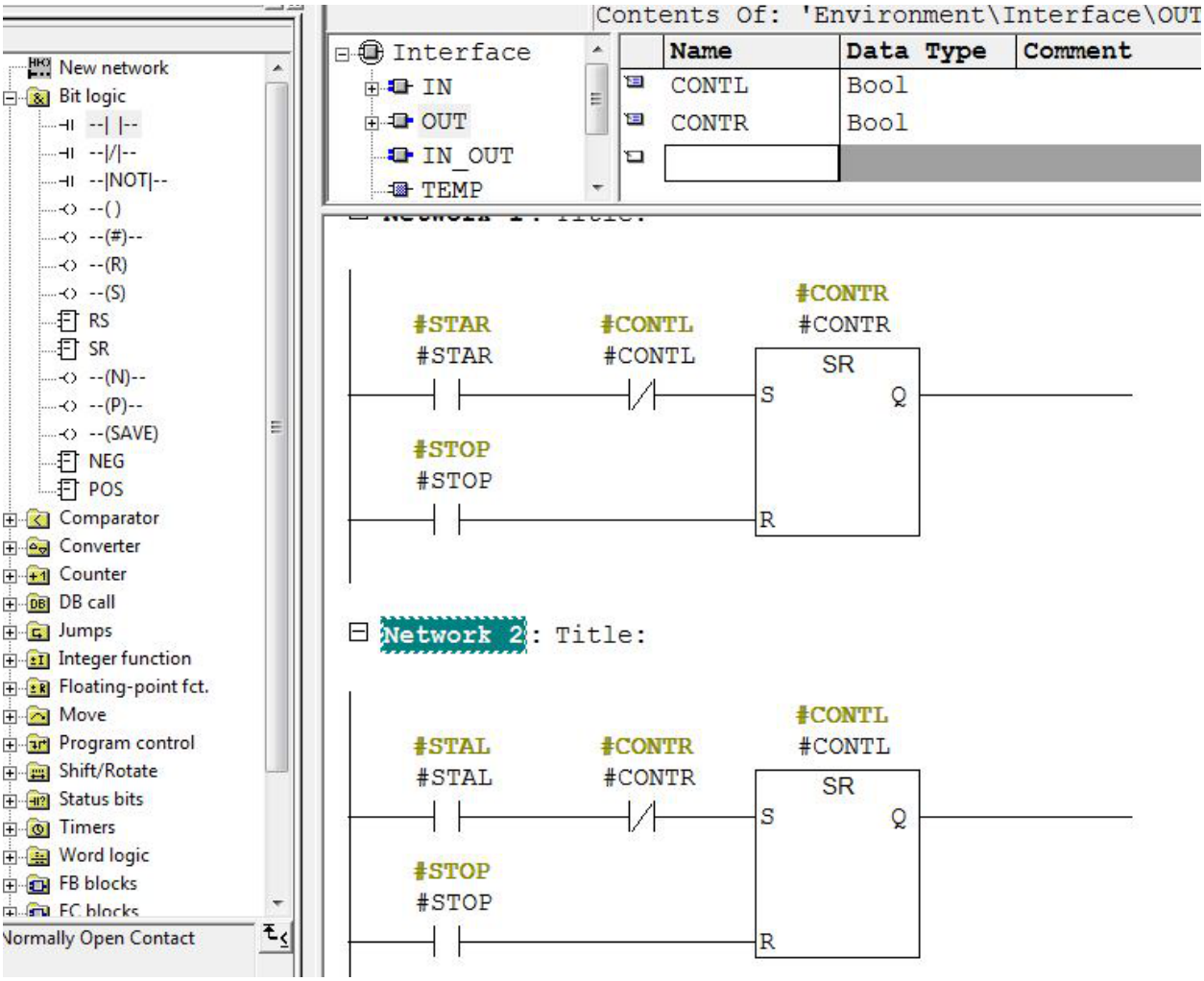
با کلیک بر روی گزینه IN در قسمت چپ جدول توابع اقدام به وارد کردن ورودیها می نماییم.



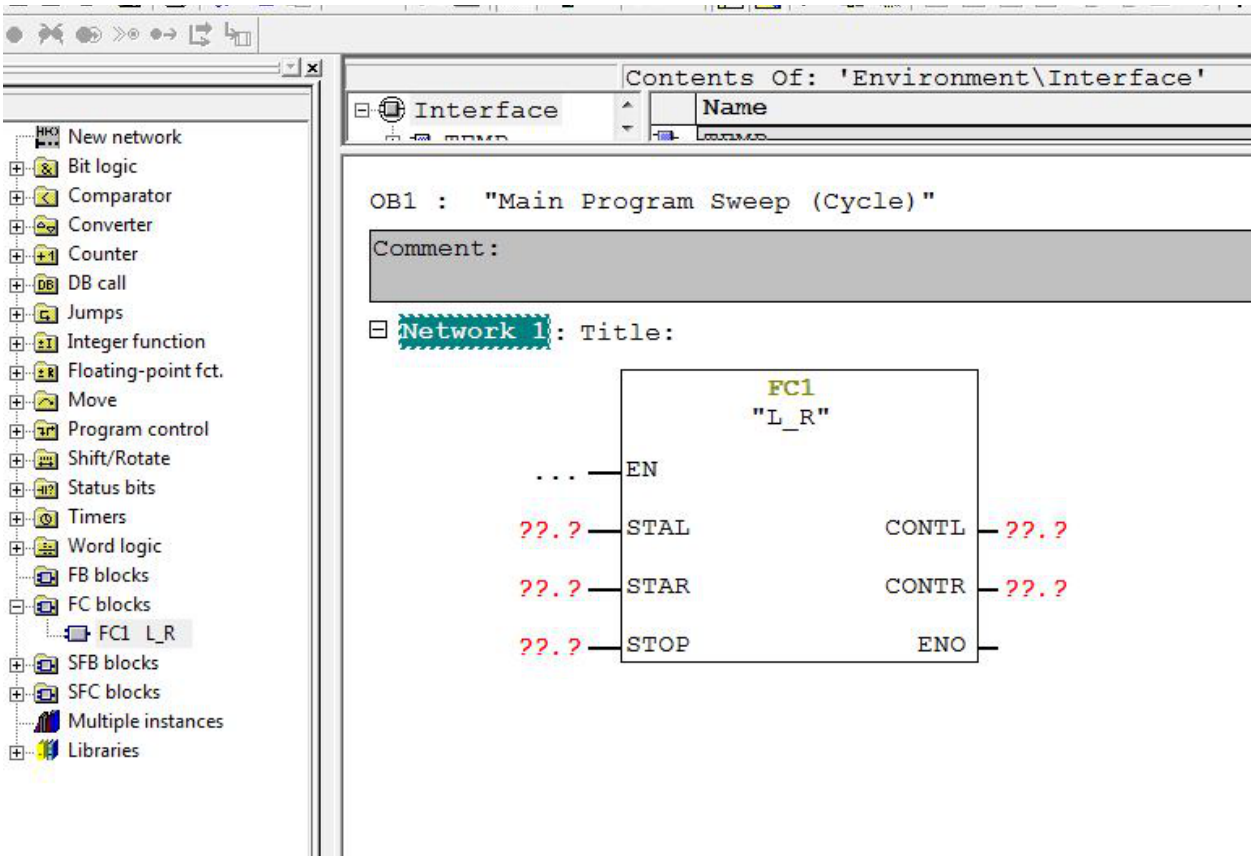
و با کلیک بر روی OUT خروجی ها را وارد میکنیم.



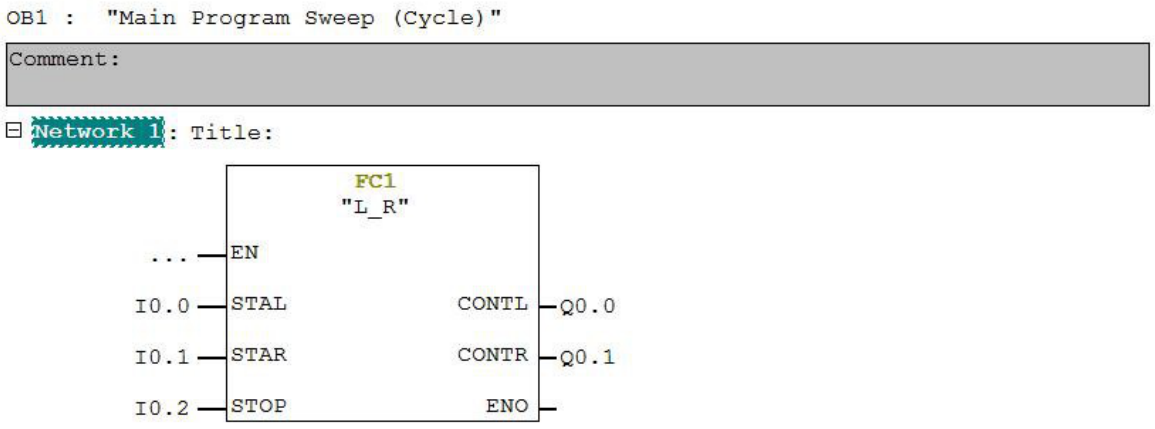
سپس برنامه مربوط به موتور چپگرد- راستگرد را بدون آدرس دادن و با اسامی فوق در FC1 می نویسیم.



در ادامه بلوک FC1 را در بلوک OB1 برای موتور اول فراخوانی می کنیم.



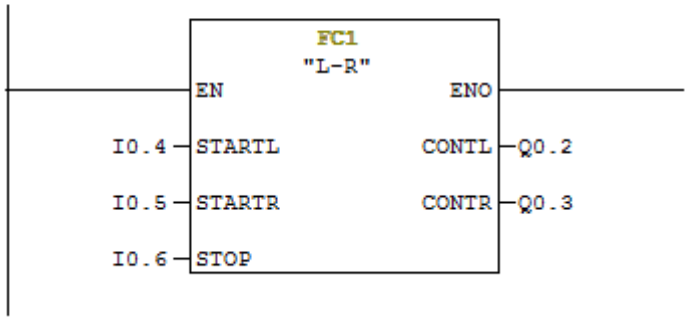
بعد از آن آدرسهای مربوط به موتور اول را وارد میکنیم.



و همین کار را برای 4 موتور باقی مانده انجام میدهیم.

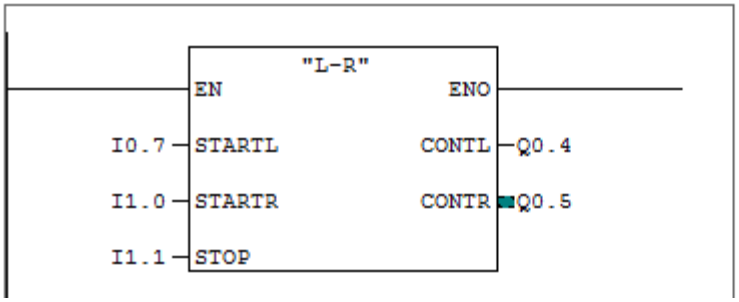
Network 2 : Title:

Comment:



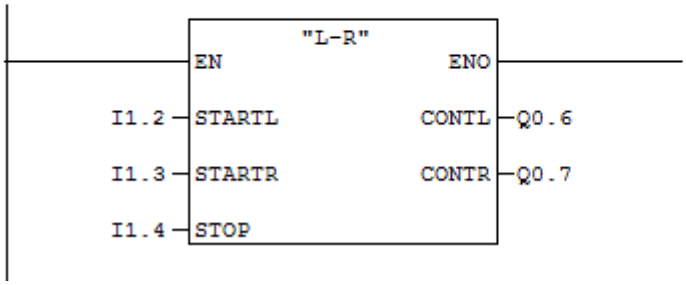
Network 3 : Title:

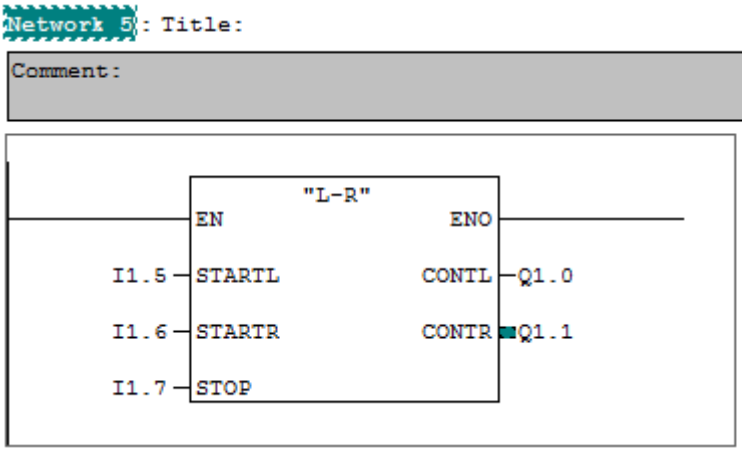
Comment:



Network 4 : Title:

Comment:





مروری بر انواع حافظه:

حافظه بارگذاری(Load Memory)

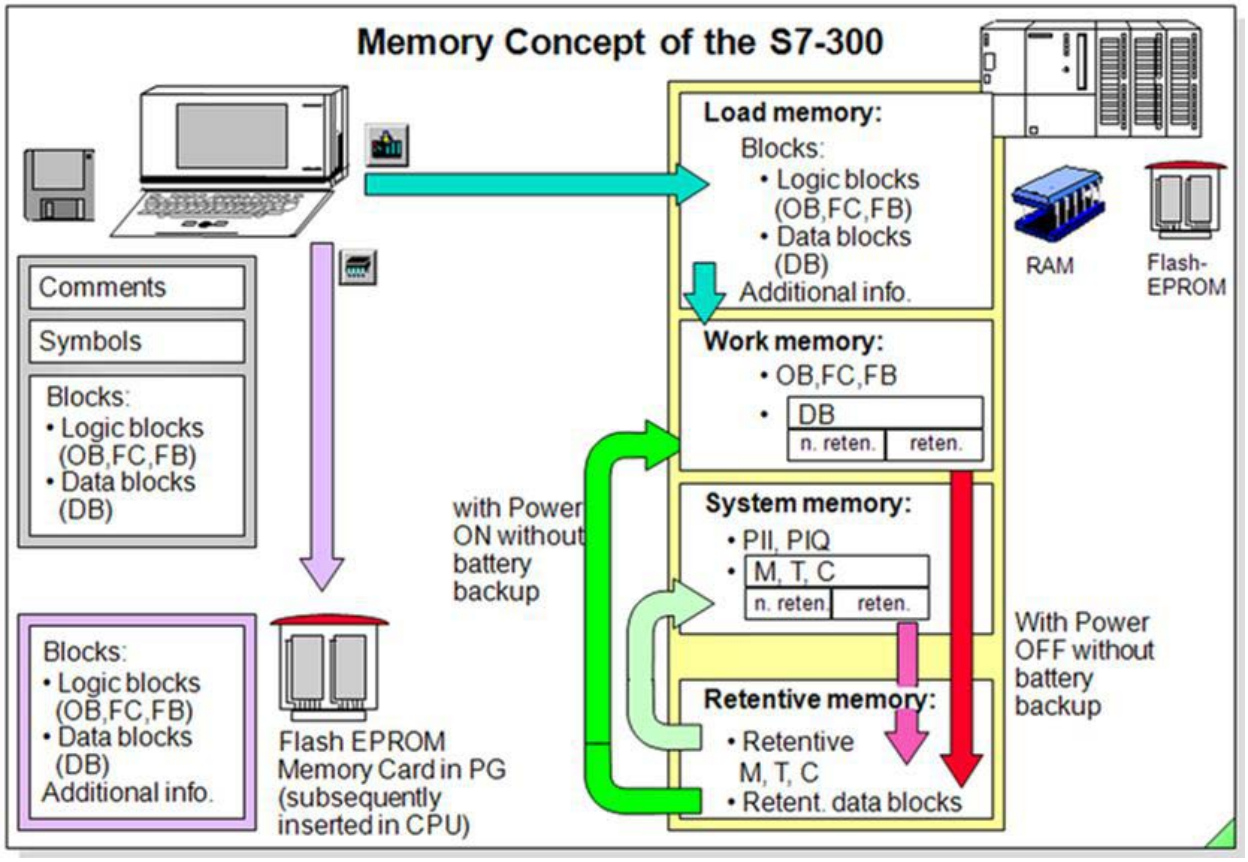
این حافظه بخشی از ماژول قابل برنامه نویسی است که شامل اجزای ایجاد شده در PLC توسط کاربر می باشد.این حافظه میتواند به صورت RAM یا یک کارت حافظه باشد.

حافظه کاری(Work Memory)

این حافظه درون CPU قرار دارد و شامل داده های در حال اجرا می باشد.

حافظه سیستمی(System Memory)

این بخش از حافظه به منظور ذخیره اطلاعات مختلف از جمله:جداول پردازش ورودی و خروجی،حافظه بیتیM، تایمرها،کانترها و L-STACK مورد استفاده قرار می گیرد.



معرفی انواع متغیرها

متغیرهای Global: این متغیرها در کل برنامه معتبر هستند.حافظه M و بلوک Data Block از این دسته هستند که در تمامی بلوک ها قابل استفاده می باشند.

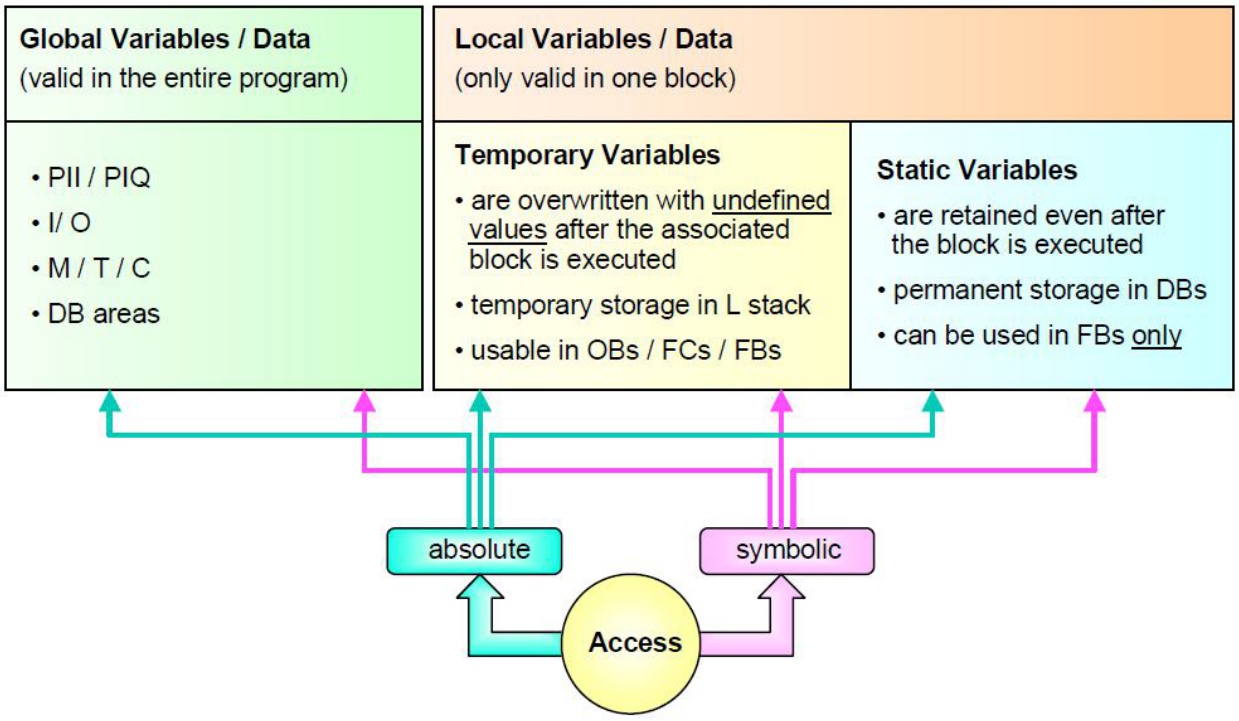
متغیرهای محلی: این متغیرها تنها در بلوک مورد استفاده شده معتبر هستند و به دو دسته زیر تقسیم میشوند.

Temporary Variable: دسترسی به متغیرهای این گروه تنها زمانی که بلوک در حال اجرا است،امکانپذیر می باشد.محل ذخیره این متغیرها که در حقیقت خود نوعی حافظه های موقت هستند، در حافظه L-STACK می باشد و همانطور که در قسمت قبل اشاره شد L-STACK جزو حافظه سیستمی CPU است و همانطور که میدانیم مقادیر موجود در حافظه L-STACK با قطع برق CPU از بین میروند.

Static Variable: متغیرهای استاتیک تنها در بلوکهای خاصی به نام Function Blockاستفاده میشوند.به این بلوکها،بلوکهای حافظه دار گفته می شود.

هر FB دارای یک بلوک DB می باشد که اطلاعات FB در آن با ذخیره میشود و جزو حافظه بارگذاری می باشد،به همین دلیل اطلاعات این بلوک علاوه بر بلوک FBدر سایر قسمتهای برنامه نیز قابل دسترسی است.

لازم به ذکر است که محتوای DB با قطع برق از بین نمی رود.



آشنایی با بلوک FC

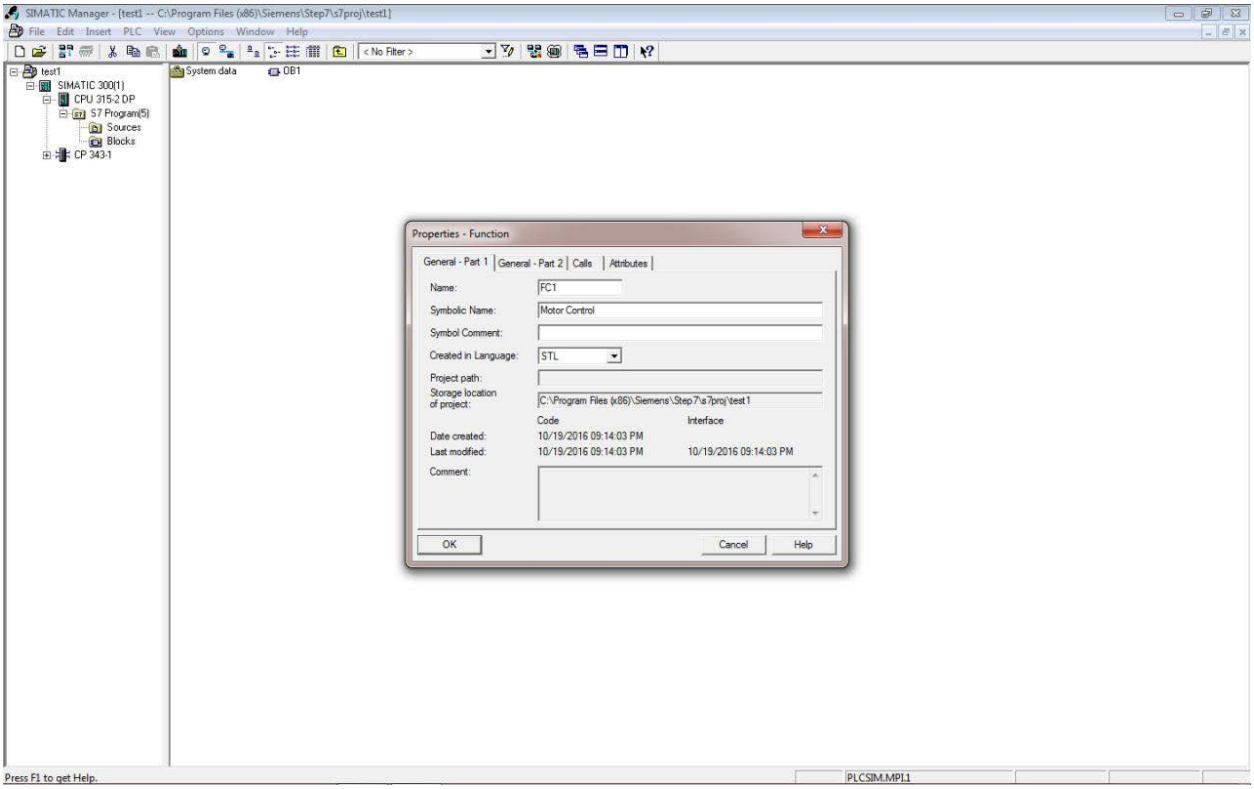
با بیان چند مثال به بررسی بیشتر مطالب فوق می پردازیم.

مثال) فرض کنید چند موتور با منطق زیر کار می کنند

زمانی که شاستی استارت زده میشود،موتورها روشن می شوند و بعد از گذشت یک بازه زمانی موتورها خاموش می شوند.شاستی استپ نیز در هر لحظه موتور را خاموش می کند مدت زمان تاخیر برای هر موتور نیز متفاوت می باشد ولی عملکرد کلی یکی است وتنها آدرس ها متفاوت می باشند

در این حالت یک بلوک طراحی می کنیم و از آن به تعداد دلخواه استفاده می کنیم.

مرحله1:ساخت بلوک FC1



نام بلوک را motor control قرار می دهیم

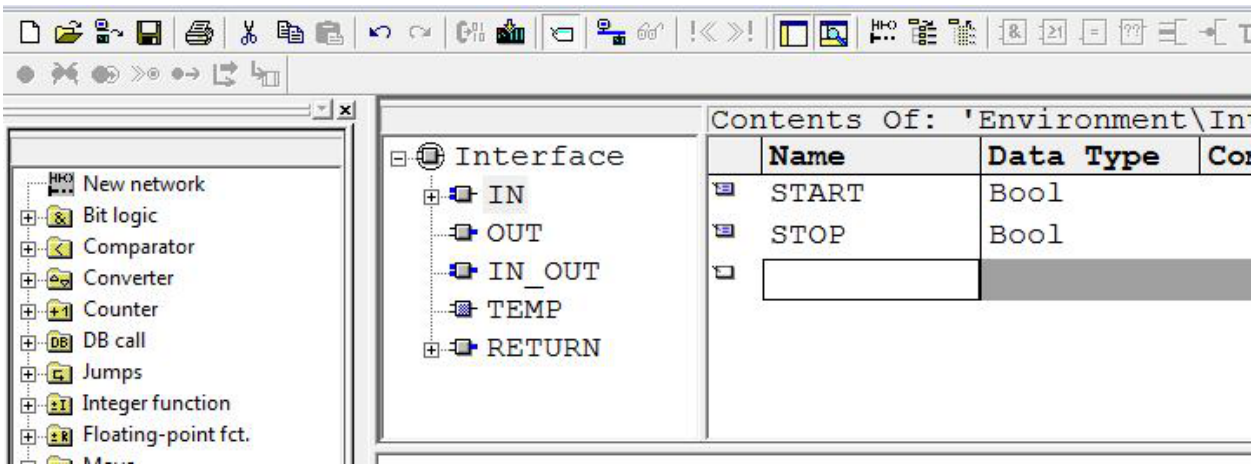
مرحله2-ایجاد پارامترهای تابع

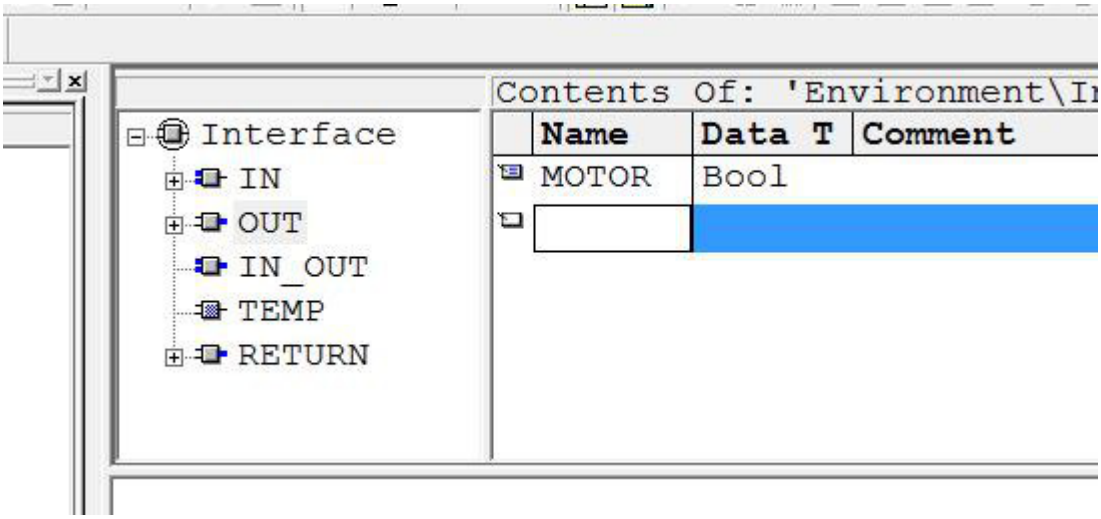
در ابتدا تعداد پارامتر اولیه ا ایجاد می کنیم

start

stop

motor



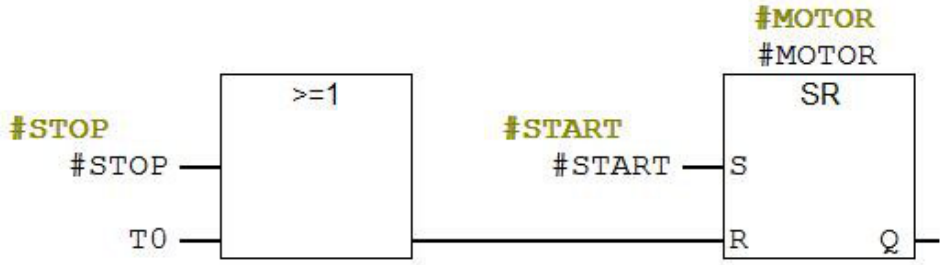


البته به پارامترهای بیشتری نیاز می باشد.در ادامه اصلاحات لازم را انجام می دهیم

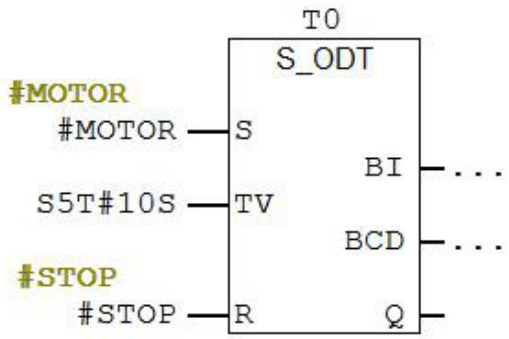
در مرحله بعد برنامه را می نویسیم.

Comment:

Network 1 : Title:



Network 2 : Title:



در این مثال با زدن استارت موتور روشن می شود.در ادامه یک تایمر S-ODT فعال می گردد.زمانی که تیغه تایمر فعال شد، موتور ریست می شود

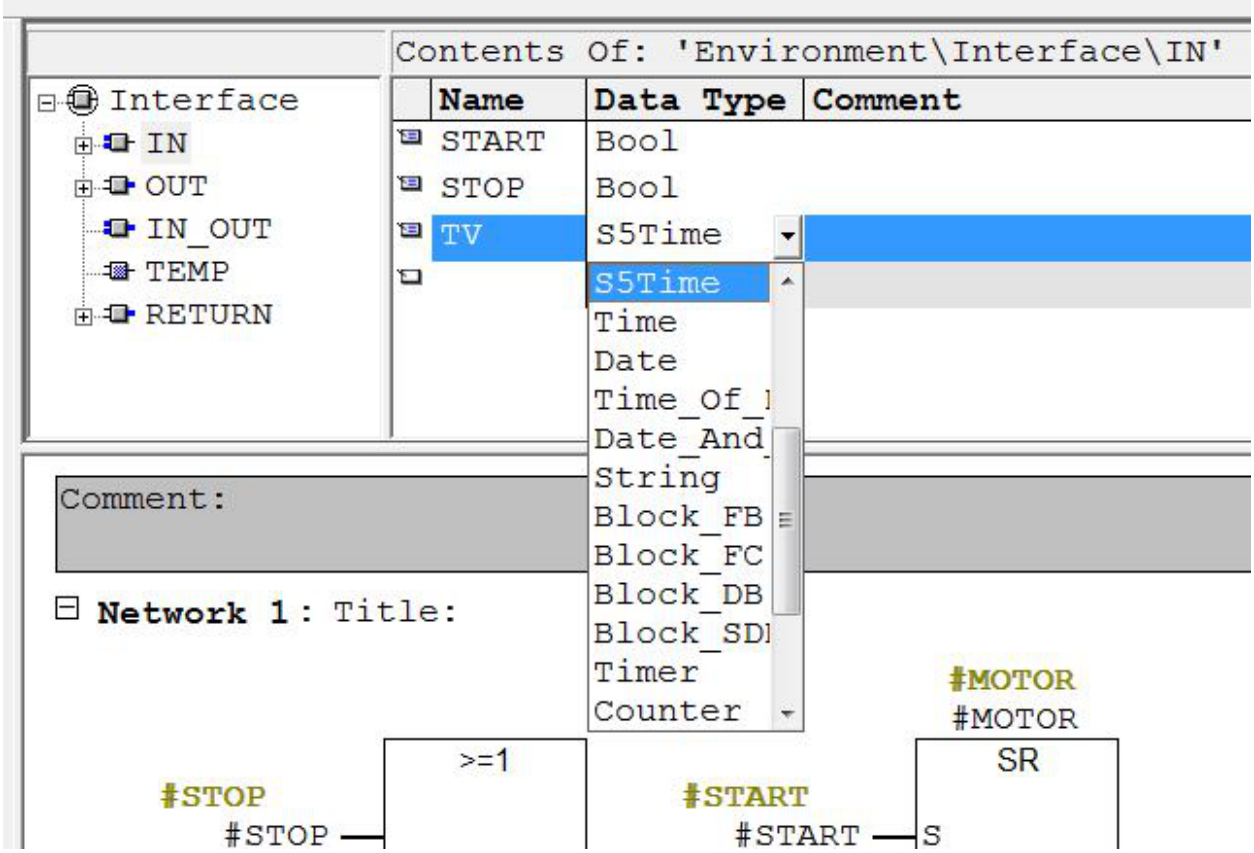
با کمی دقت در برنامه نوشته شده در میابیم که چند مشکل در این مثال وجود دارد.

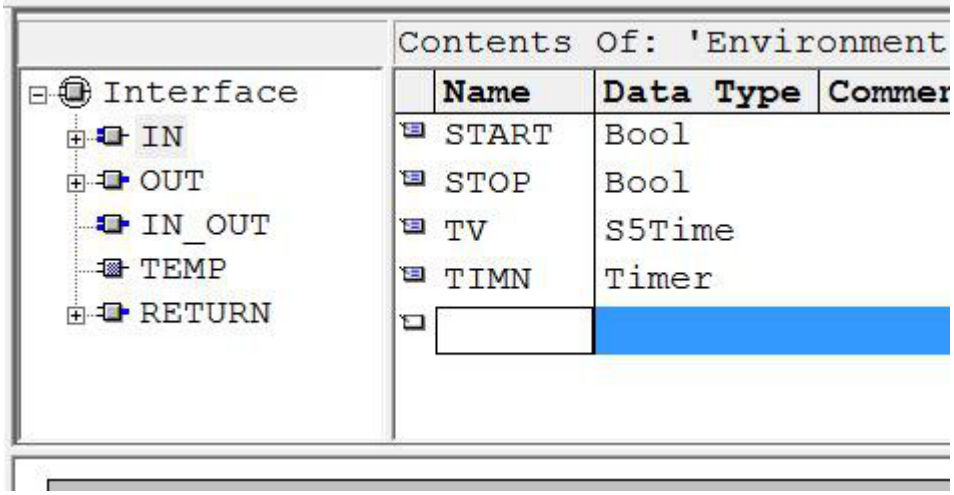
در مورد پایه TV تایمر،اگر قرار باشد زمان را روی 10 ثانیه ثابت کنیم دیگر قابل تغییر نخواهد بود و ما میدانیم که زمان تاخیر برای موتورها متفاوت است.

از طرفی یک تایمر با شماره T0 هم نمی تواند چندین زمان مختلف را شمارش کند

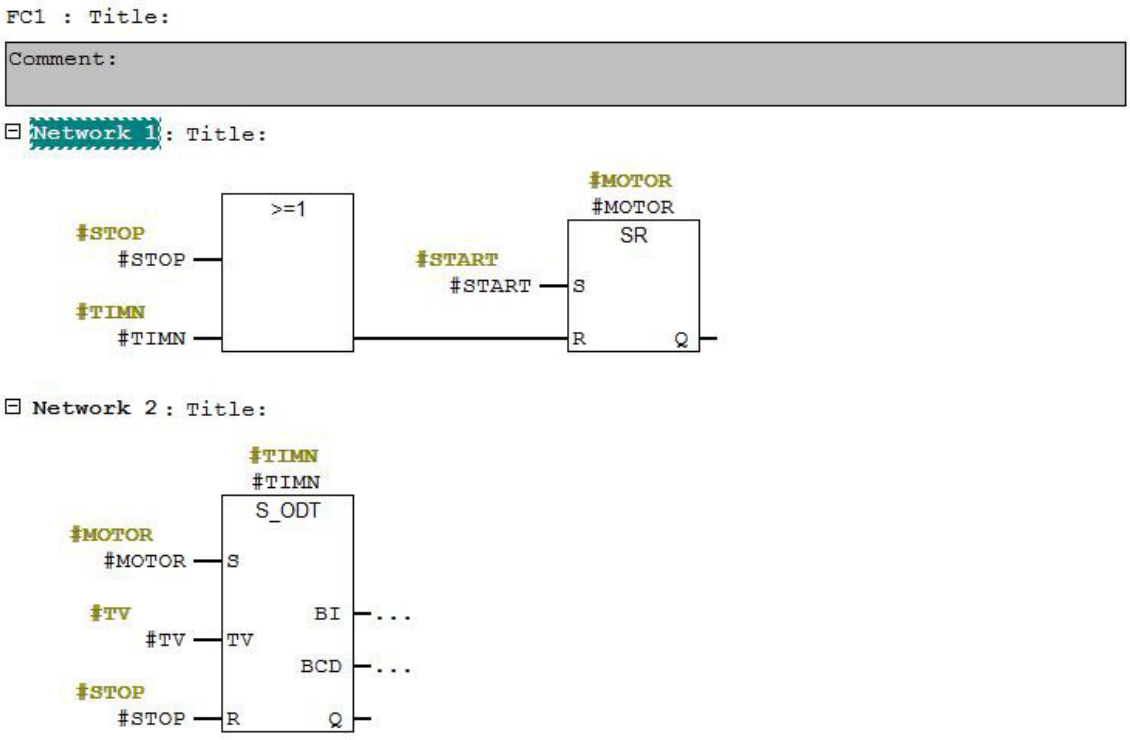
بنابراین این دو پارامتر باید به عنوان متغیر در جدول توابع تعریف شوند.

حال جدول و برنامه را اصلاح میکنیم.



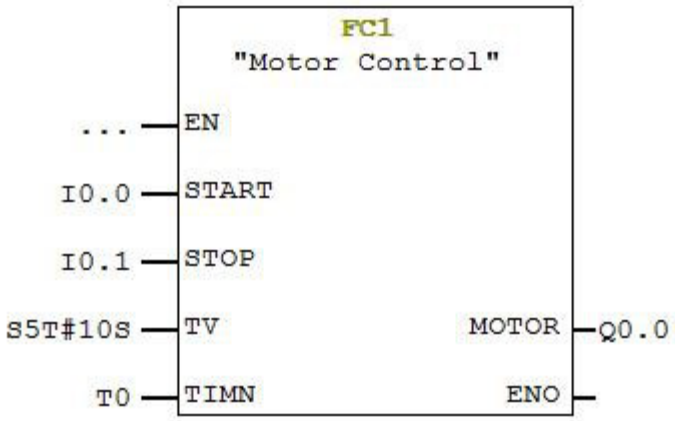


به نوع مقادیر جدید دقت کنید.

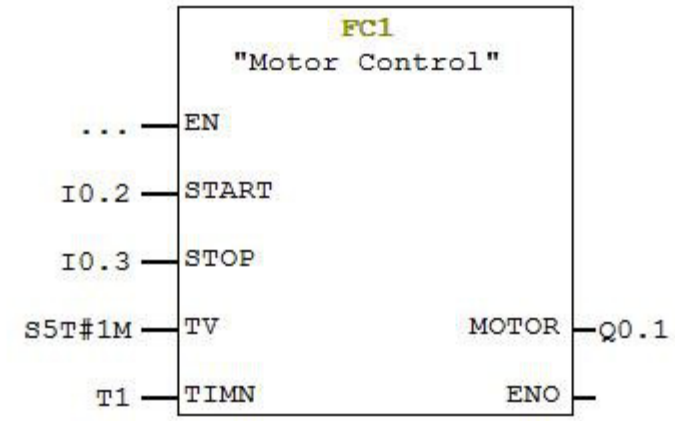


همانطور که ملاحظه میکنید تمامی مقادیر متغیر شدند.

مرحله بعد فراخوانی به تعداد دلخواه در بلوک OB1 می باشد.



Network 2 : Title:



در شکل فوق این تابع برای کنترل دو موتور با آدرس ها، زمان، شماره تایمرو خروجی های مختلف استفاده شده است.

مثال) فرض کنید 4ترموکوپل داریم،ترموکوپل ها بعد از یکسری محاسبات ، مقدار دما را در حافظه های مختلفی به صورت زیر منتقل می کنند

TC1=MD0

TC2=MD4

TC3=MD8

TC4=MD12

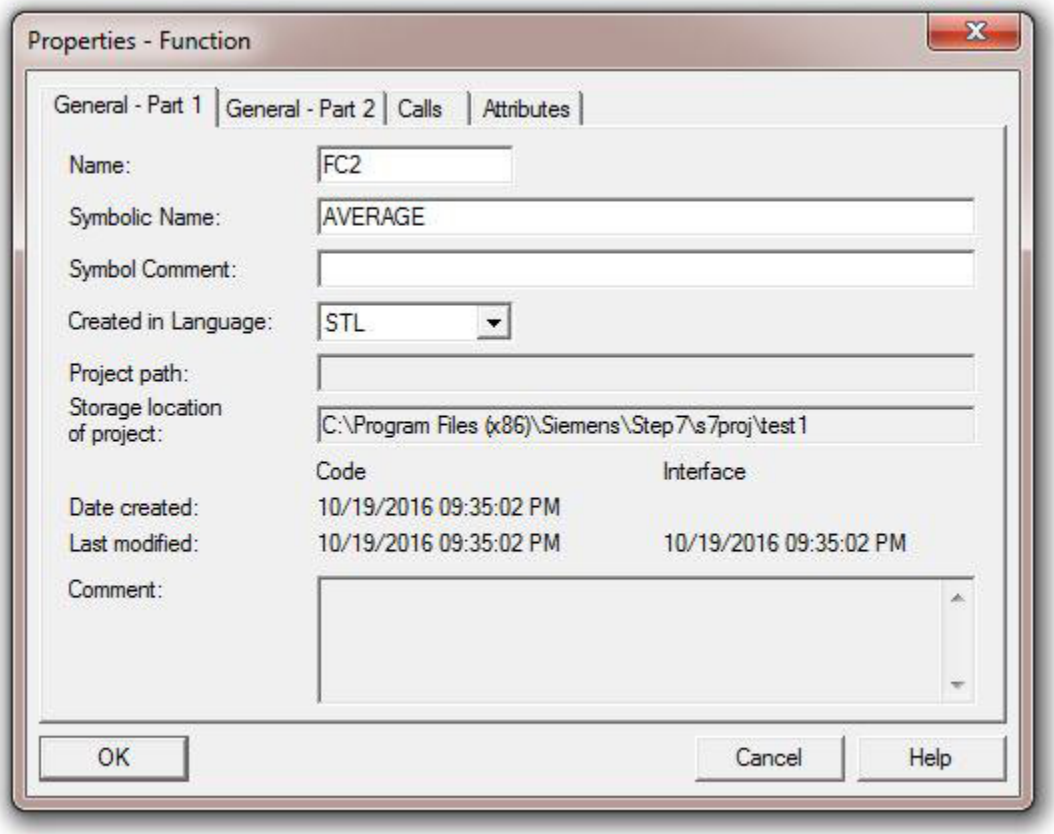
عبارت TC نام در نظر گرفته شده برای ترموکوپل ها می باشند و بازه دمایی 0 تا 350 درجه می باشد.حافظه MDنیز جهت ذخیره مقدار دما استفاده شده است.

هدف از این مثال نوشتن برنامه ای است که مقدار 4 دما را دریافت کند و مقدار میانگین آنها را محاسبه کند و در خروجی AVRG نیز نمایش دهد و سپس مقدار میانگین را با ورودی SP مقایسه کند

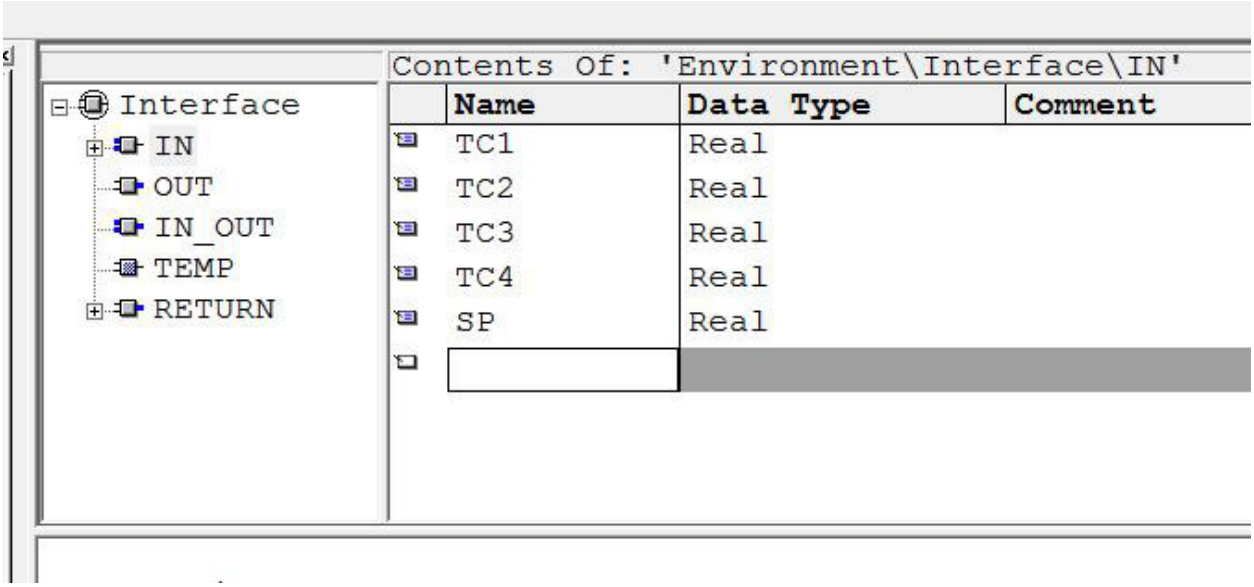
اگر مقدار AVRG بیشتر از مقدار SP بود، خروجی V1 را فعال کند خروجی V2 غیر فعال باشد و اگر مقدار AVRGکمتر از SP بود، خروجی V2 روشن و V1 خاموش گردد از این تابع به دفعات در برنامه با آدرس های مختلف نیاز می باشد.

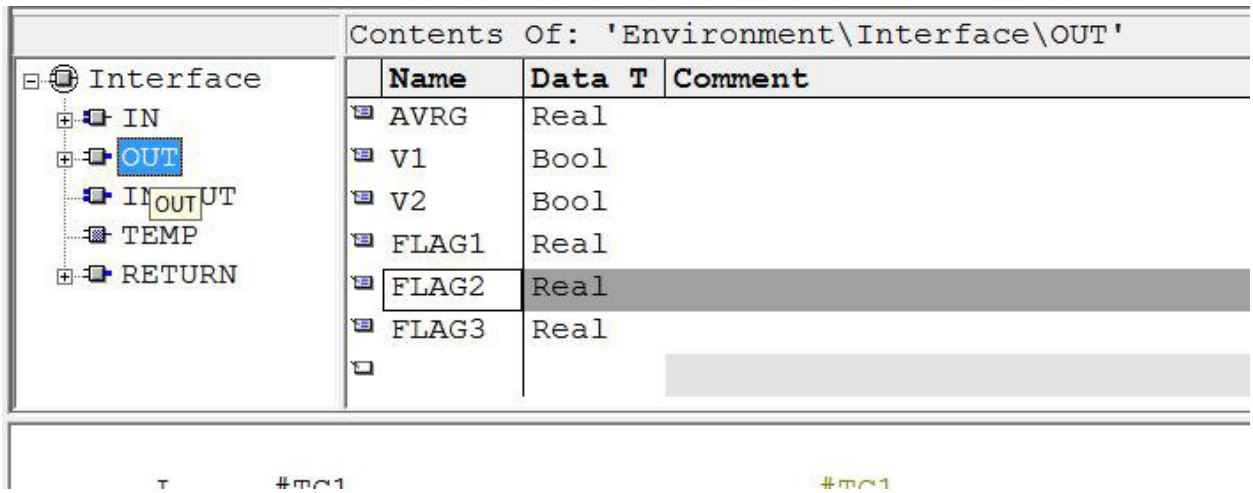
همانطور که میدانیم دما از نوع REAL می باشد.

مرحله 1:ساخت تابع FC1



تعریف متغیرهای تابع





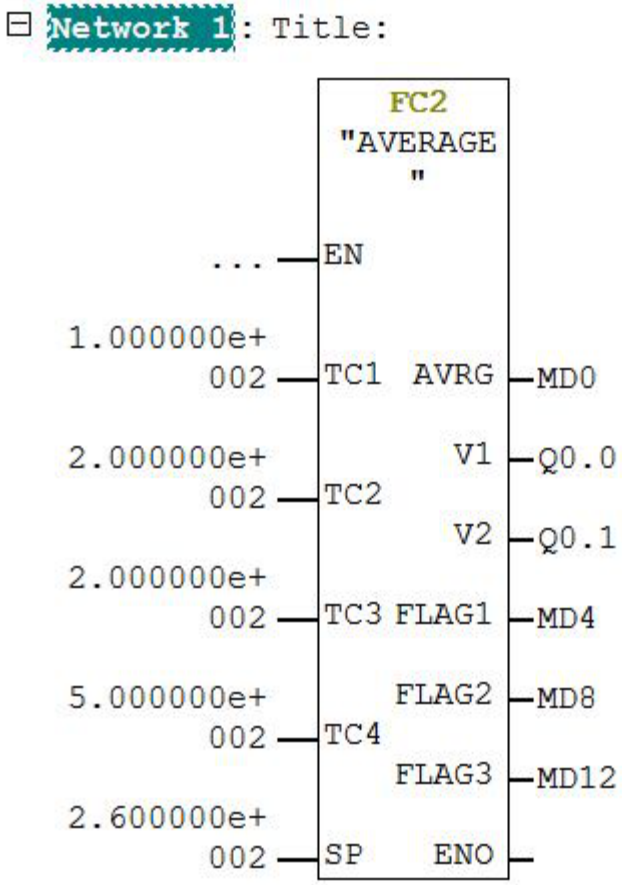
FLAG2,FLAG1 و FLAG3 برای انجام عملیات ریاضی به منظور محاسبه میانگین دماها تعریف شده اند.

برنامه این مثال را جهت تمرین بیشتر به روش STL می نویسیم

```
L     #TC1
L     #TC2
+R
T     #FLAG1
L     #TC3
L     #TC4
+R
T     #FLAG2
L     #FLAG1
L     #FLAG2
+R
T     #FLAG3
L     #FLAG3
L     4.000000e+000
/R
T     #AVRG
L     #AVRG
L     #SP
>R
=     #V1
AN    #V1
=     #V2
```

حال تابع را در بلوک اصلی برنامه فراخوانی می کنیم.

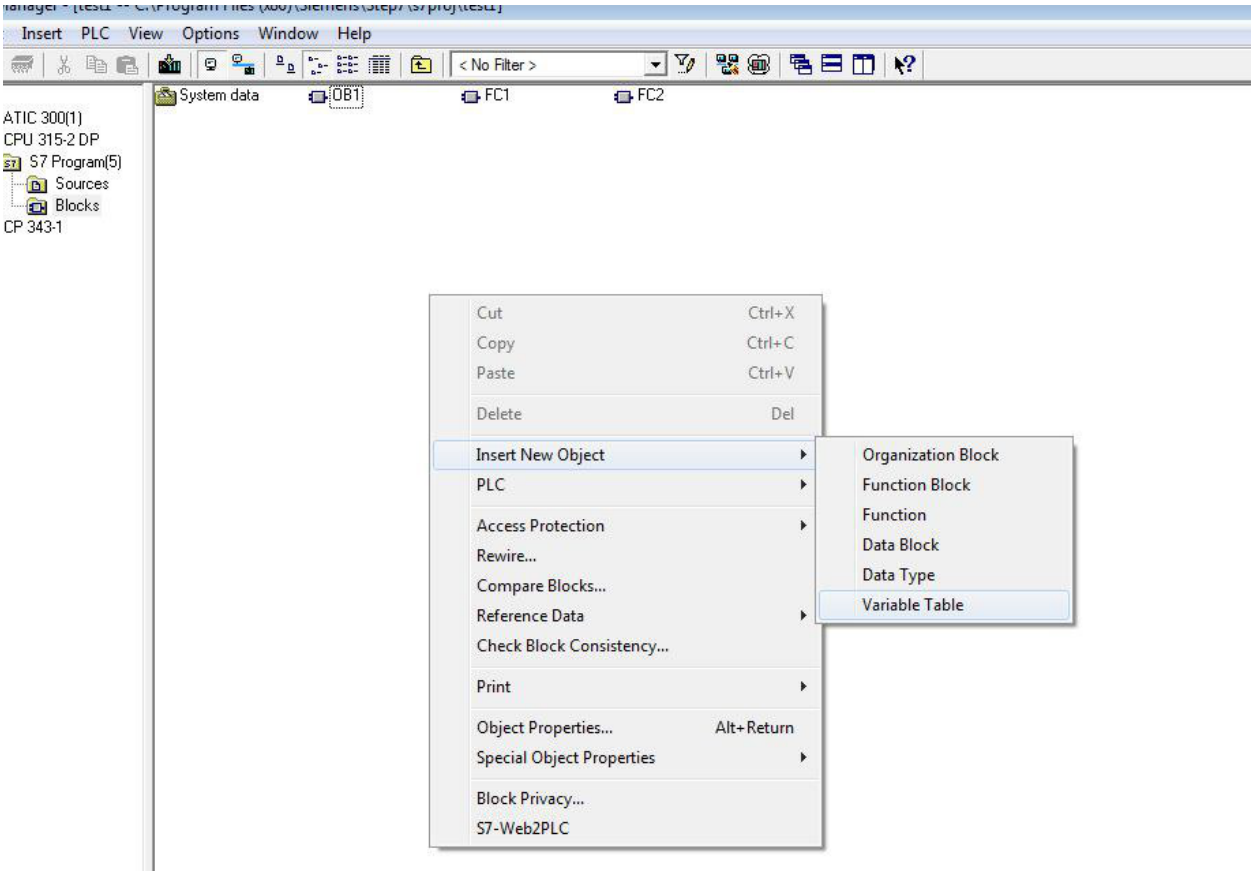
برای راحتی کار می توانیم به ورودی تابع مقادیر ثابت می دهیم.

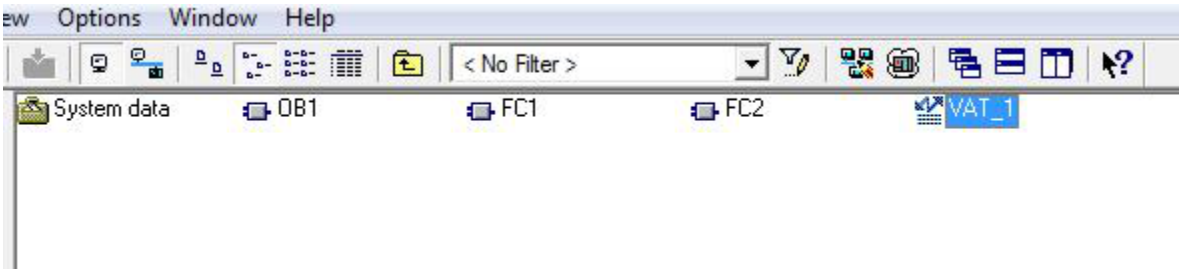


مقدار SP=260 واد شده است و مقدار میانگین اعداد وارد شده 250 می باشد.

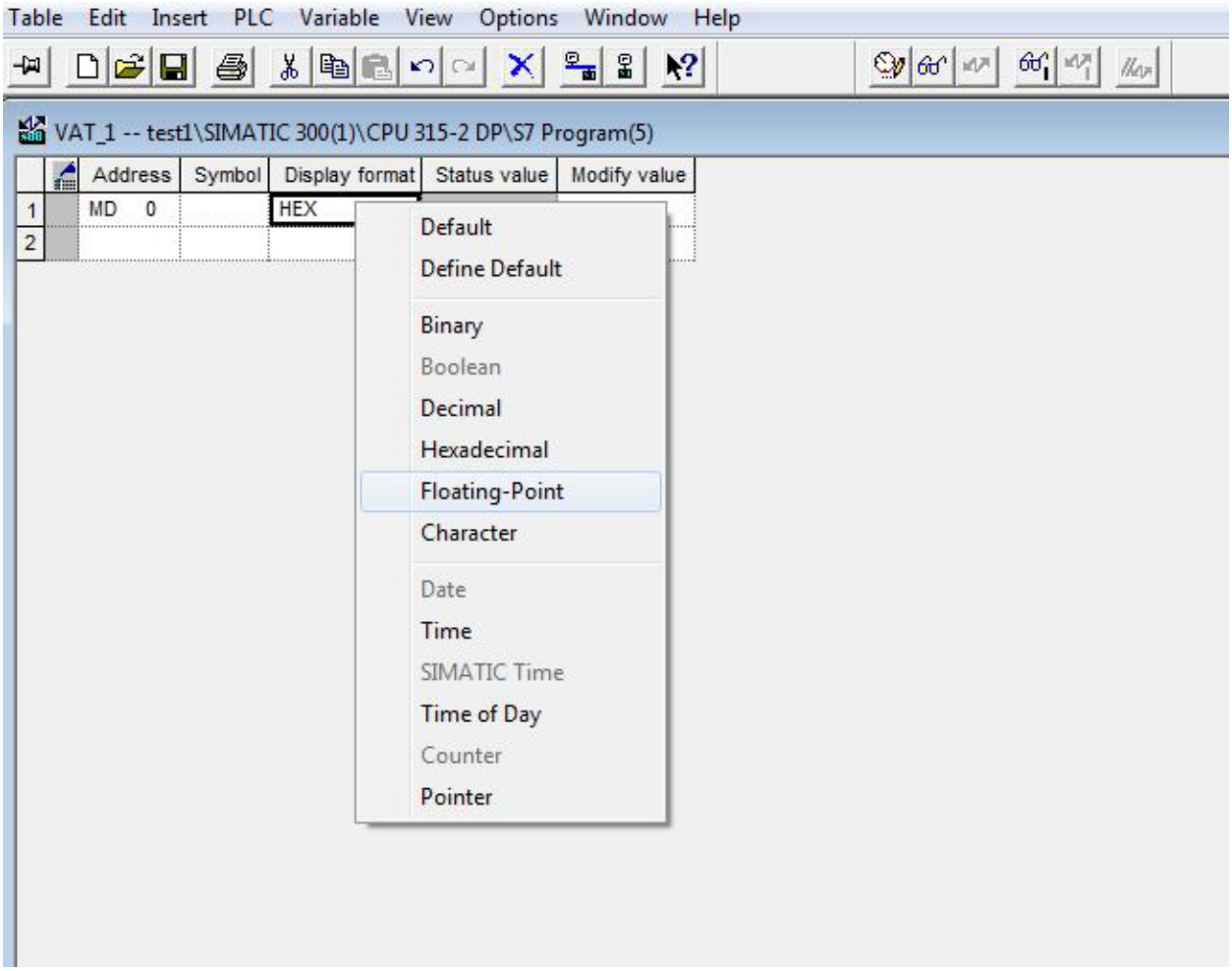
برنامه را تست می کنیم

برای مشاهده مقادیر از جدول VAT که یکی از قابلیت های نرم افزار است،استفاده می کنیم



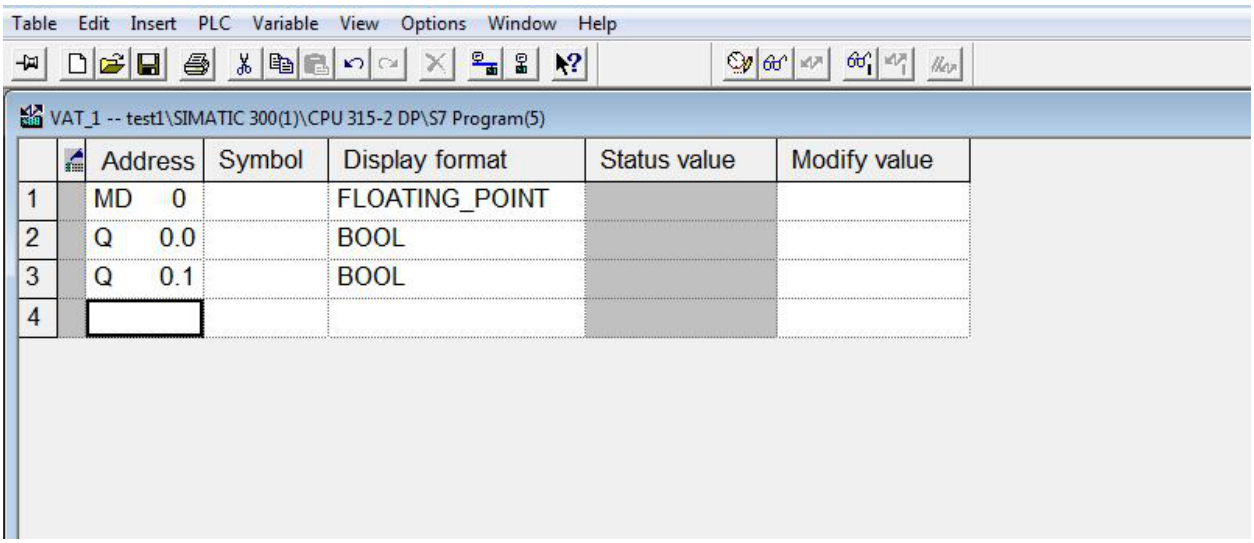


با توجه به برنامه مقدار MD0 برابر مقدار AVRG می باشد که باید مانیتور شود



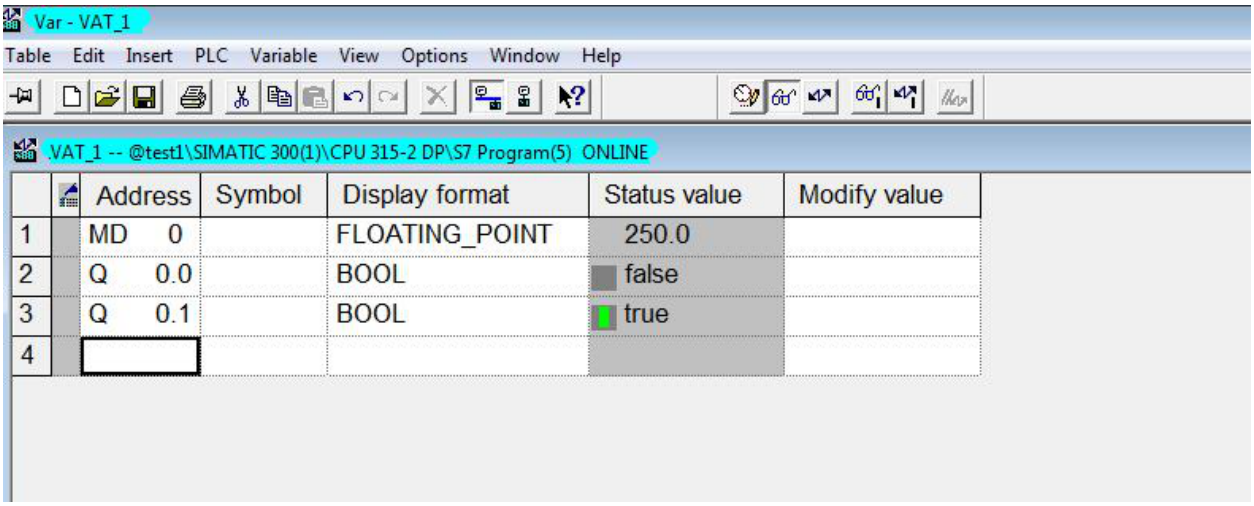
فرمت نمایش MD0 را نیز بر روی FLOAT می گذاریم

دو خروجی برنامه Q0.0و Q0.1 را نیز در جدول وارد می کنیم.



بعد از دانلود FC و OB بر روی شبیه ساز سیستم را RUN می کنیم

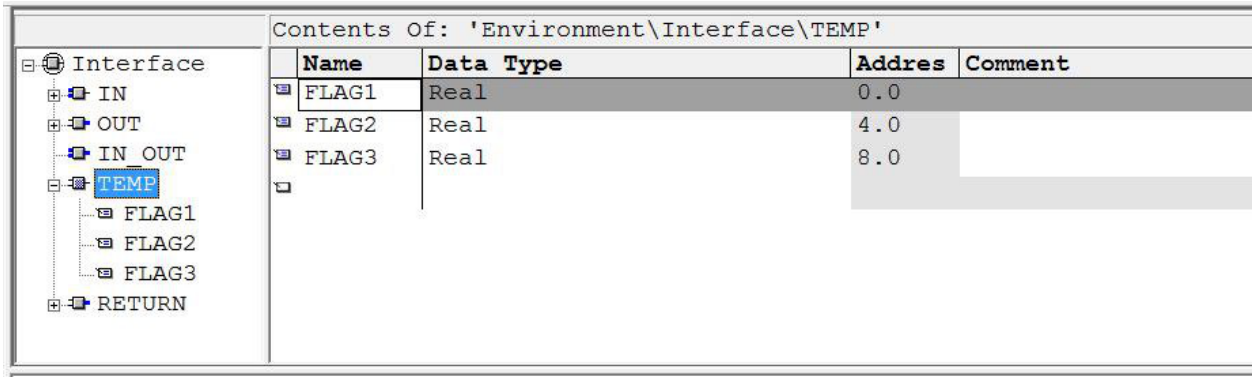
در جدول VAT آیکون عینک را فعال می کنیم



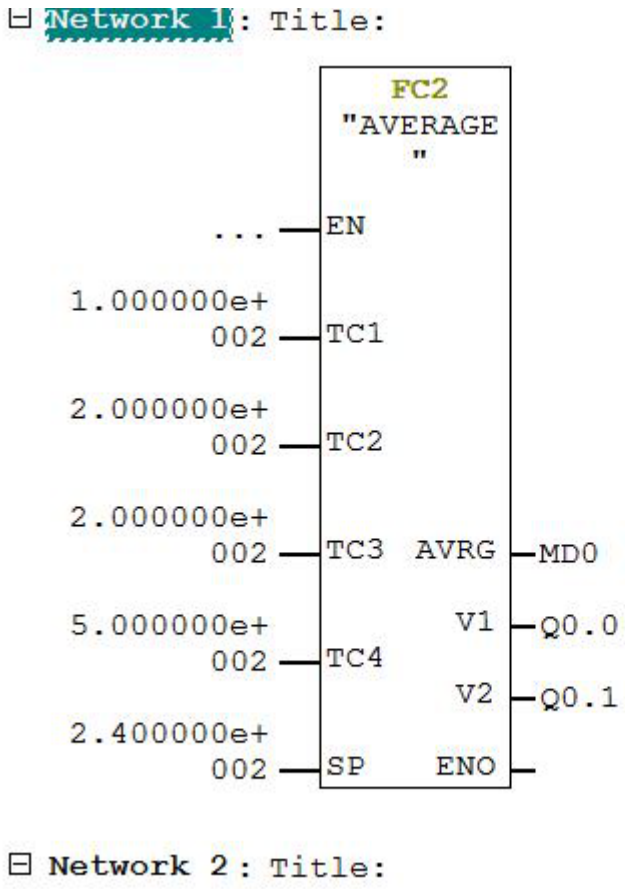
همانطور که مشاهده میکنید خروجی V2 روشن و V1 خاموش می باشد مقدار میانگین نیز نمایش داده شده است.

با کمی دقت در میابیم که نیازی به نمایش مقادیر نتایج میان برنامه یعنی همان مقادیر FLAG ها نمی باشد بنابراین می توانیم جهت کاهش زمان برنامه نویسی و همچنین اشغال فضای کمتری از حافظه،از حافظه های موقت در تعریف FLAG ها استفاده کنیم.

این بار متغیرهای FLAG را به داخل TEMP منتقل می کنیم



با فراخوانی مجدد بلوک FC در بلوک اصلی برنامه شکل زیر مشاهده می شود.



توجه به این نکته بسیار حایز اهمیت است که حجم حافظه موقت در CPU محدود است و این مقدار از قبل توسط سازنده تعریف شده است و چنانچه مقدار حافظه مورد استفاده در برنامه بیشتر از مقدار تعریف شده باشد CPU به صورت خودکار خاموش می شود.

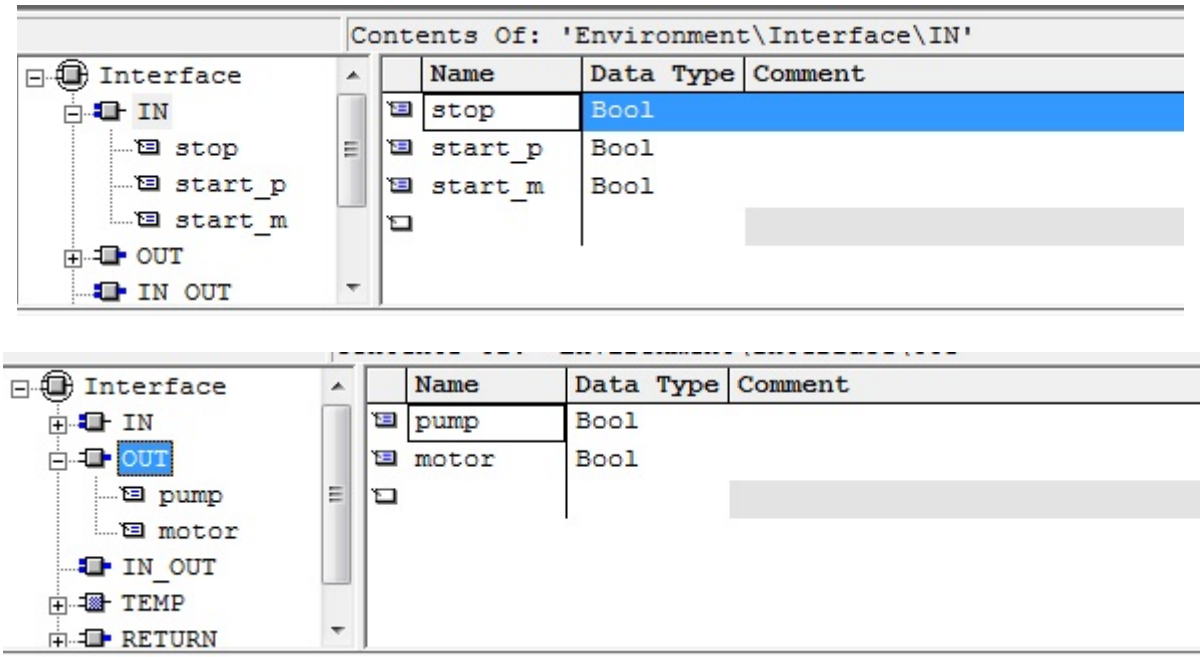
حال میخواهیم ضعف بلوک FC را با طرح یک مثال بررسی کنیم. همانطور که قبلا بیان شد حافظه در FCها به صورت Temporary یا موقت است و این حافظه ها فقط در بلوک مربوط خود معتبر میباشد.

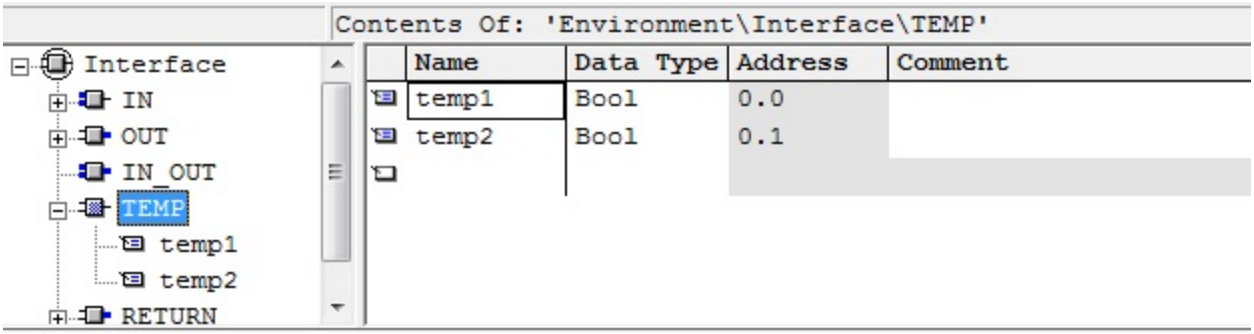
فرض کنیم برنامه ساده ی پمپ و موتور یکی پس از دیگری را به صورت فانکشن و به شکل زیر برنامه نویسی کنیم:

موتور تنها در صورتی که پمپ قبل از آن روشن شده باشد کار میکند.میخاهیم تابع این برنامه را بنویسیم و دوبار آن را فراخانی کنیم.

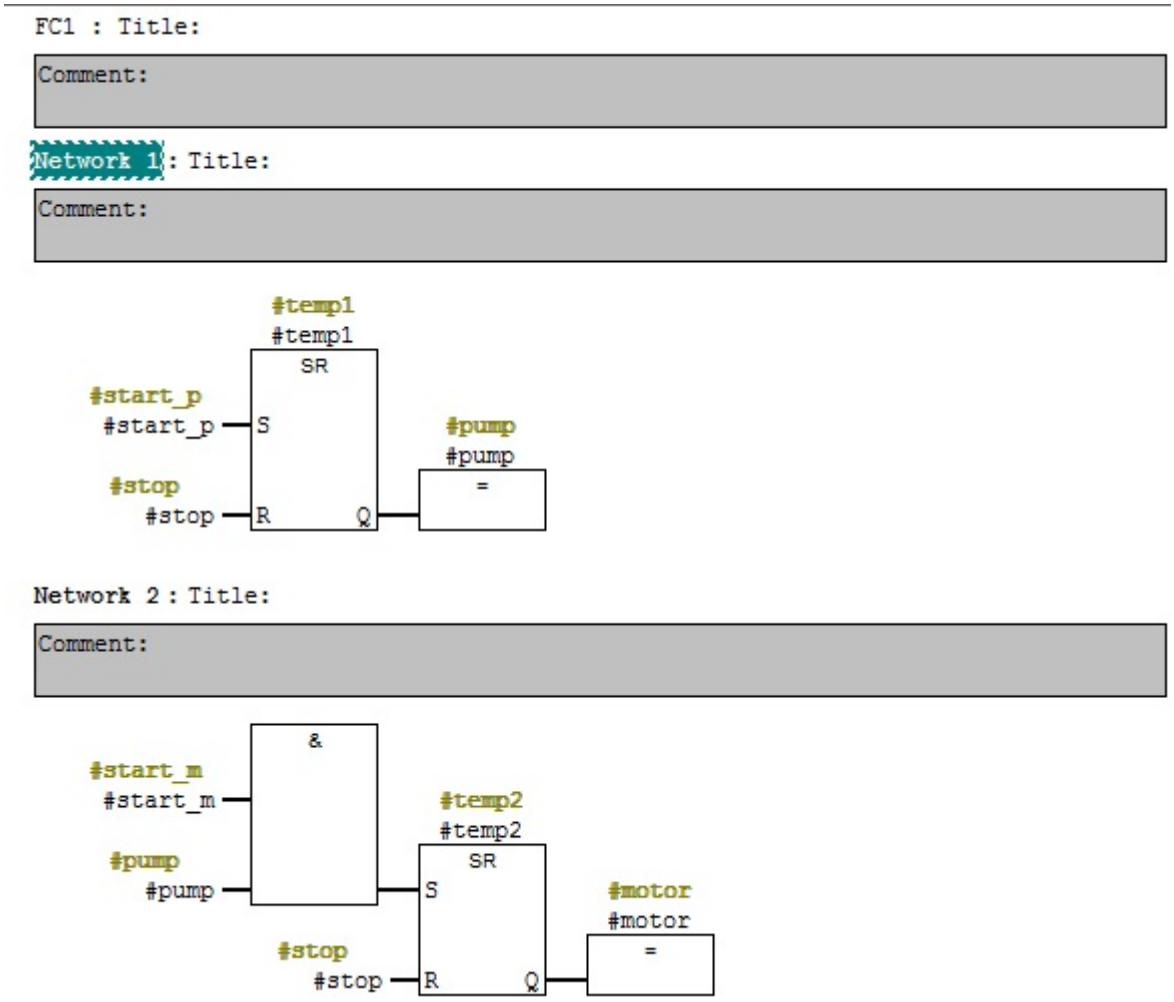
برنامه را در FC1 مینویسیم

ابتدا مقداری ورودی و خروجی و حافظه موقت را برای تابع مشخص میکنیم





برنامه یکی پس از دیگری را با استفاده از حافظه Temp مینویسیم:



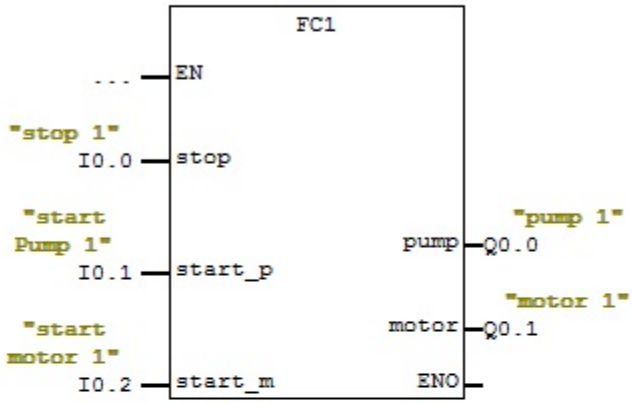
برنامه را save کرده و در OB1 دوبار فراخانی میکنیم و سپس آدرس های مورد نظر را برای قسمت اول و بعد از آن برای قسمت دوم وارد میکنیم.

OB1 : "Main Program Sweep (Cycle)"

Comment:

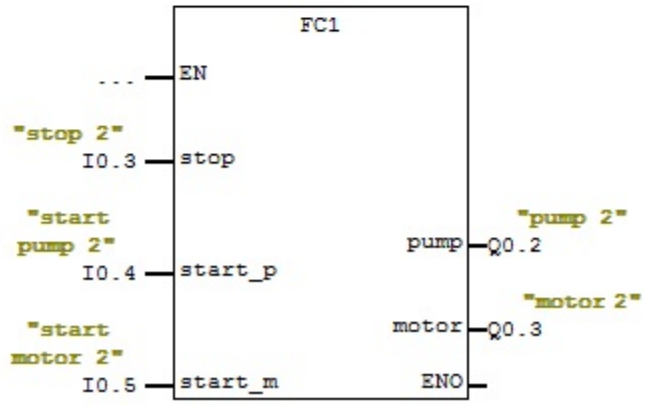
Network 1: Title:

Comment:



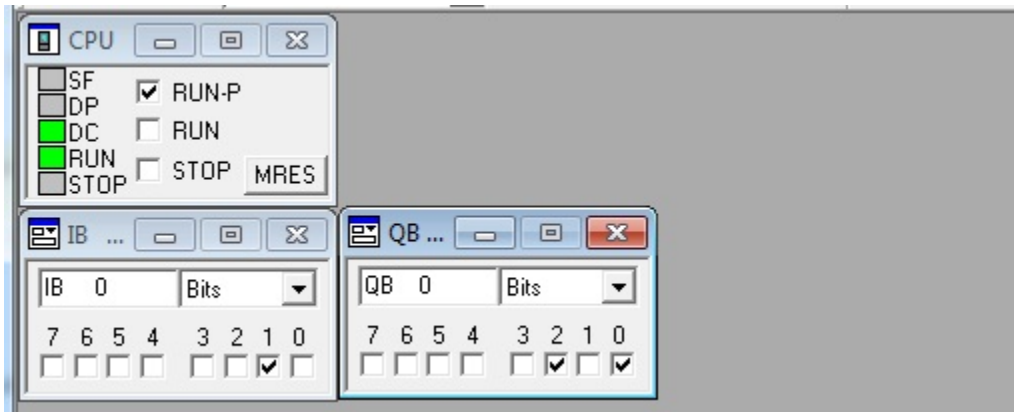
Network 2 : Title:

Comment:



برنامه راsave میکنیم و OB1 و FC1 را در سیمولیشن دانلود میکنیم.

با فعال کردن ورودی start pump 1 مشاهده میکنیم که علاوه بر pump 1 خروجی مربوط به pump 2 نیز فعال میشود. علت این امر به اشتراک گذاری حافظه موقت در FC هاست و چنانچه یک مقدار مشخص شده از temp در یک network فعال شود این مقدار در network های بعدی و قبلی نیز فعال میشود.



به همین دلیل به بلوک های FC بلوک های بدون حافظه میگویند و برای رفع این مشکل بلوک های FB که در ادامه با آن آشنا میشوید بکار گرفته میشوند.

آشنایی با بلوک FB

بلوک های FB به عنوان بلوک های حافظه دار، دارای عملکردی همانند بلوک های FC می باشند. تفاوت بین این دو بلوک در این است که پارامترهای یک FB در یک DB خاص ذخیره می شود. بلوکی که حتما می بایست به یک FB اختصاص داد DB نوع Instance می باشد. در واقع هر بلوک FB پارامترهای خود را در یک بلوک DB ذخیره می کند. از این رو به بلوک های FB بلوک های حافظه دار می گویند و برای نوشتن توابع کلی تر از این بلوک ها استفاده می شود.

دیتا بلوکی که به یک FB اختصاص داده می شود به صورت اتوماتیک ایجاد می گردد،اما نکته قابل توجه در مورد بلوک FB این است که دیتا بلاک ها محل نوشتن برنامه نیستند بلکه مثل فضای M ، حافظه CPU محسوب می شوند و از آنها جهت ذخیره سازی دیتا استفاده می شود.

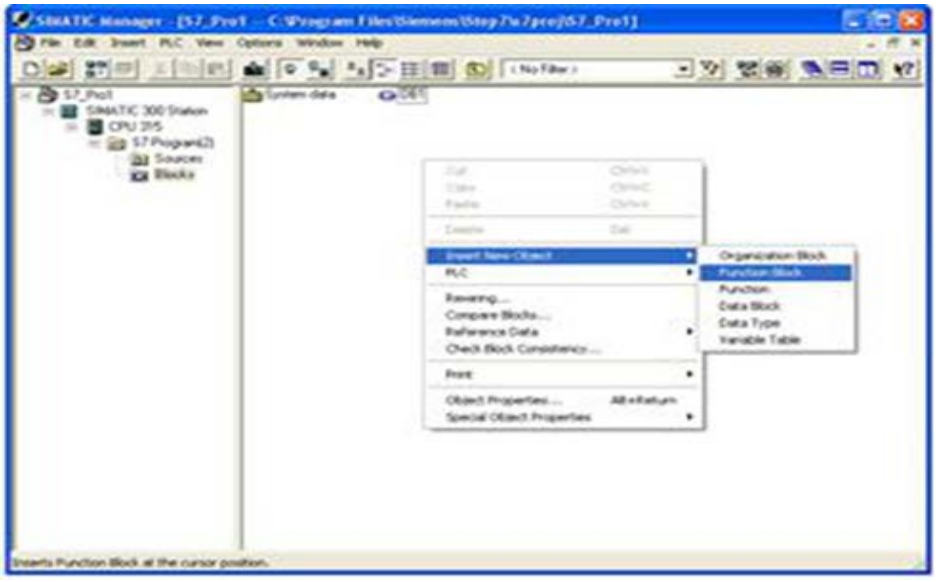
به عبارت دیگر با هر بار فراخوانی FB می بایست یک بلوک DB به آن اختصاص داد.در واقع زمانی که FB را فراخوانی می کنیم، یک فضای حافظه به آن اختصاص می دهیم. بلوک فراخوانی شده FB ، پارامترهای I/O خود را در این بلوک ذخیره می کند بنابراین به تعداد I/O های بلوک FB ، در دیتا بلاک مربوطه سطر ایجاد می شود. این سطرها همان محل ذخیره سازی I/O های FB می باشند که به صورت اتوماتیک ایجاد می شوند.پس این بلوک ها ، بلوک های اختصاصی می باشند.

در واقع زمانی که از یک بلوک FB استفاده می کنیم، اگر به I/O های تابع ، حافظه یا مقداری اختصاص ندهیم، نرم افزار خطایی نمی گیرد به این دلیل که هر I/O به طور اتوماتیک دارای یک آدرس از بلوک هایDB اختصاص داده شده می باشد.

استفاده از FB وقتی در برنامه اصلی نیاز به یک تابع تکراری باشد که فقط هر بار ورودی آن تغییر می کند یا آدرس خروجی های آن عوض می شود بسیار مفید است و منجر به کاهش برنامه می شود.

مثال) در این مثال می خواهیم یک تابع میانگین برای 4 ورودی که با فرمت 16 بیتی صحیح می باشند را طراحی کنیم(ساخت تابع میانگین توسط بلوکFB)

از مسیر نشان داده شده در شکل زیر یک بلوک FB ایجاد می کنیم



پس از وارد شدن به بلوک FB در جدول توابع متغیرهای مورد نیاز را تعریف میکنیم.

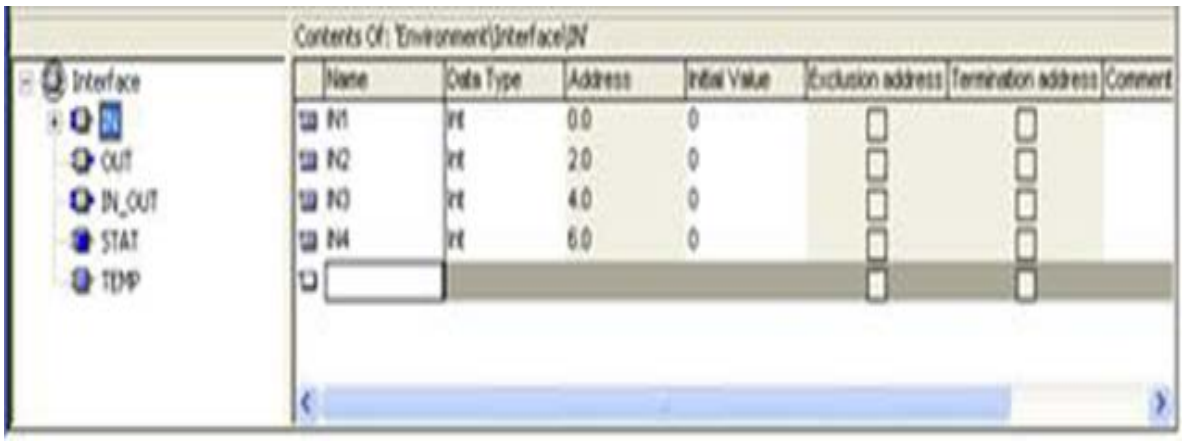
پارامترهای IN :

IN1(ورودی 1 -16 بیتی صحیح)

IN2(ورودی 2 -16 بیتی صحیح)

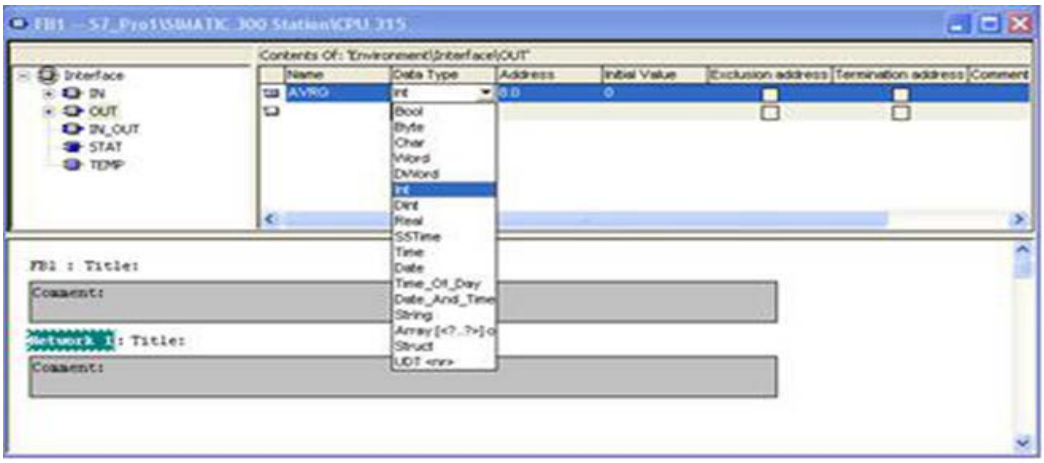
IN3(ورودی 3 -16 بیتی صحیح)

IN4 (ورودی 4 -16 بیتی صحیح)



پارامترهای OUT :

AVRG (خروجی تابع-16 بیتی صحیح)

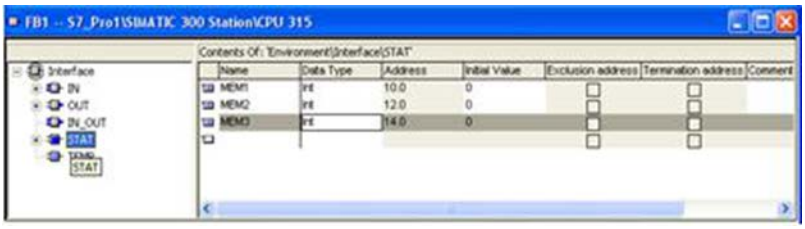


پارامترهای STAT (حافظه های کمکی):

MEM1

MEM2

MEM3

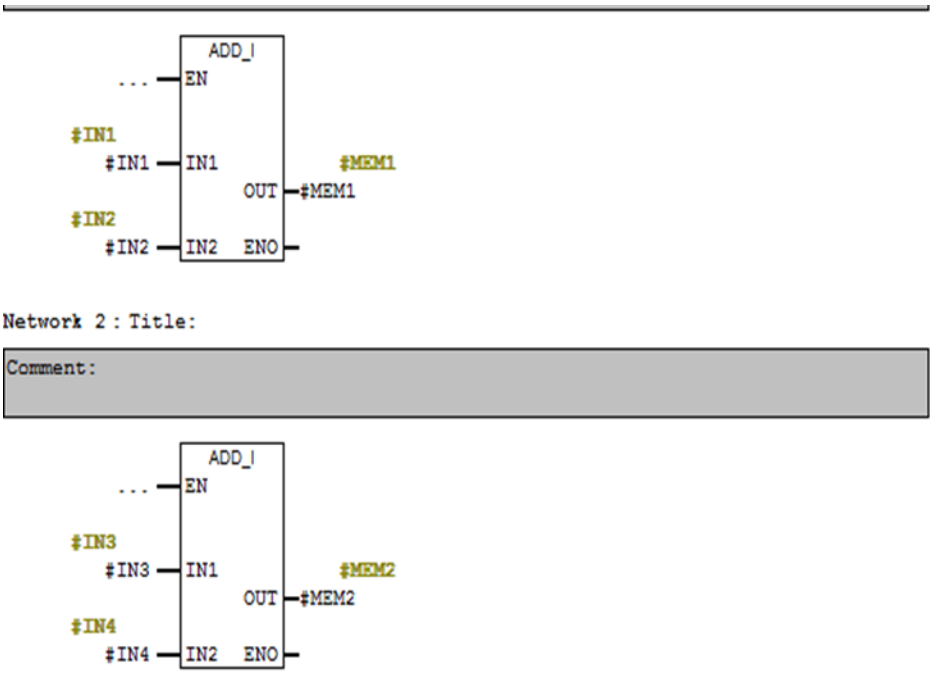


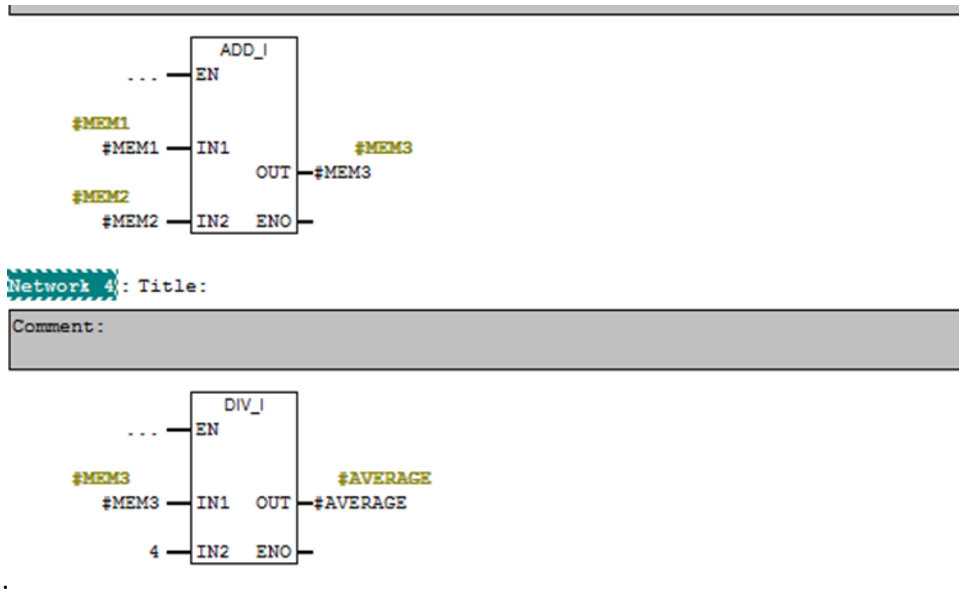
همانطور که مشاهده میکنید بلوک های FB مجهز به متغیرهای با نام STAT یا استاتیک نیز می باشند

همانطور که در بحث های قبلی بیان شد متغیرهای استاتیک مختص بلوک های FB می باشند. محل ذخیره شدن این متغیرها بر خلاف متغیر TEMP در بلوک های DB می باشد. از این رو دیتای ذخیره شده در متغیرهای محلیDB بعد از اجرای بلوک نیز در دسترس می باشند. این قابلیت باعث می شود که در برنامه با هر بار فراخوانی بلوک FB بتوان پارامترها و مقادیر مختلفی به تابع اعمال نمودکه این داده ها با اجرا نشدن بلوک، در بلوک DB باقی می مانند.

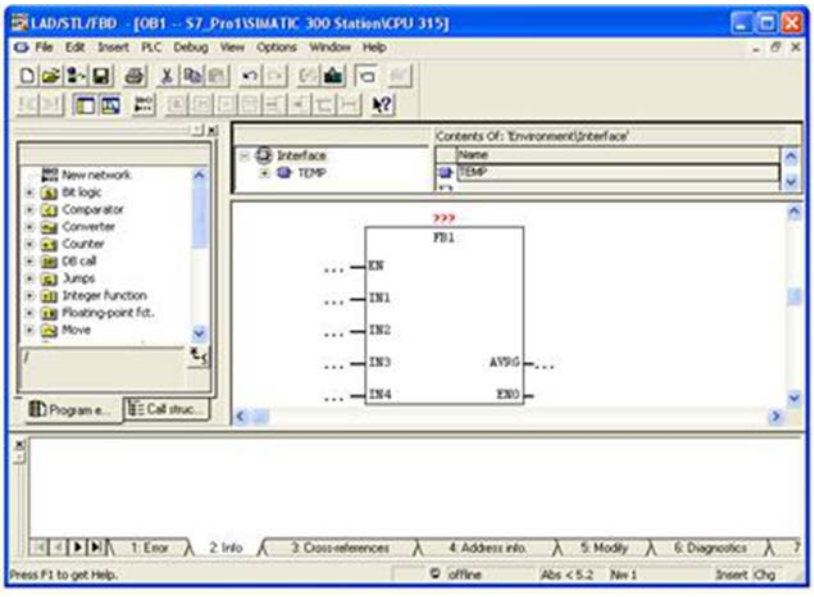
در واقع اگر متغیر را از نوع TEMP تعریف کنیم، در زمان فراخوانی تابعFB ، به عنوان I/O تابع ظاهر نمی شود.ولی این متغیر در DB قرار دارد .پس دارای آدرس می باشد و هر کجای برنامه که نیاز باشد قابل استفاده می باشد

برنامه FB1 :

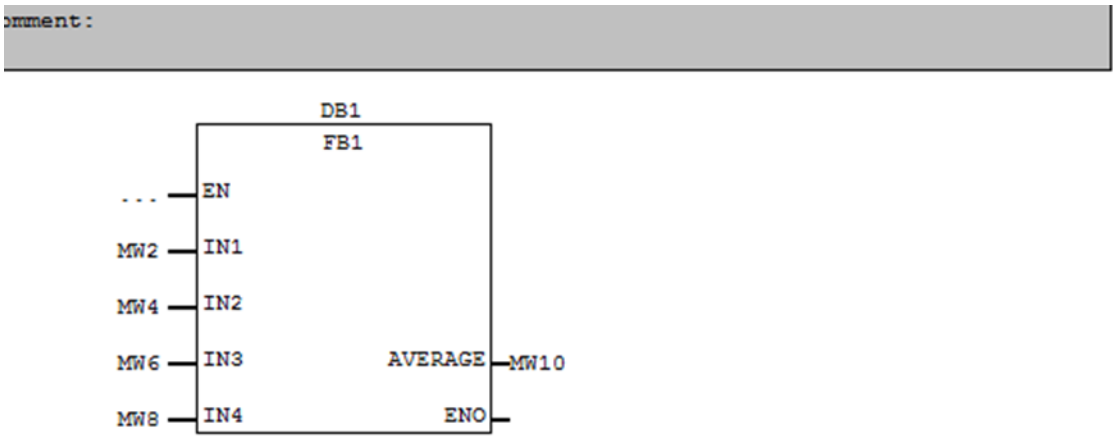




فراخوانی تابع FB1 در OB1:



در تابع فراخوانی شده در ابتدا یک DB با شماره 1 به تابع مورد نظر اختصاص و سپس ورودی هایی که قصد گرفتن میانگین آنها را داریم وارد می کنیم. جواب نهایی نیز به خروجی AVRG منتقل می گردد. اگر این تابع برای آدرس های دیگر در Network بعدی مورد استفاده قرار گیرد حتما می بایست یک DB دیگر به آن تابع اختصاص داده شود.



با وارد کردن بلوک DB1 و با تایید آن ، بلوک DB1 در صفحه اصلی به طور اتوماتیک ساخته می شود.

در این مثال در زمان فراخوانی چند حافظه به ورودی تابع اختصاص داده شد.فرض را بر این گرفتیم که مقادیر موجود در این حافظه ها قرار است میانگین گیری شوند.

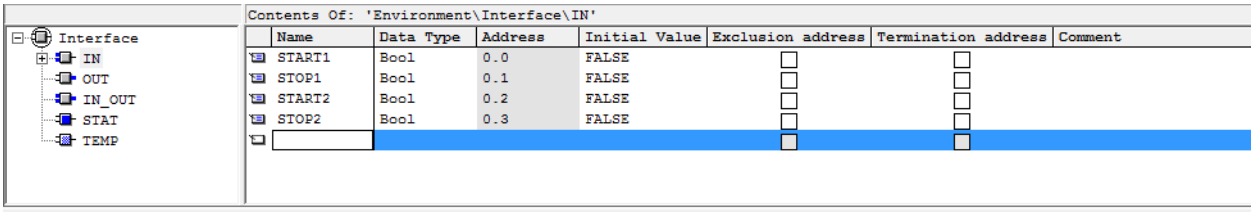
با دقت در مثال فوق در میابیم که با هر بار فراخوانی بلوک FB، از بلوک های DB متفاوتی استفاده می شد و با توجه به محدود بودن تعداد DB ها به بررسی نحوه الختصاص یک بلوک DB به چندین بلوک FB که از آن با نام قابلیت Multiple Instance یاد می شود،می پرازیم.

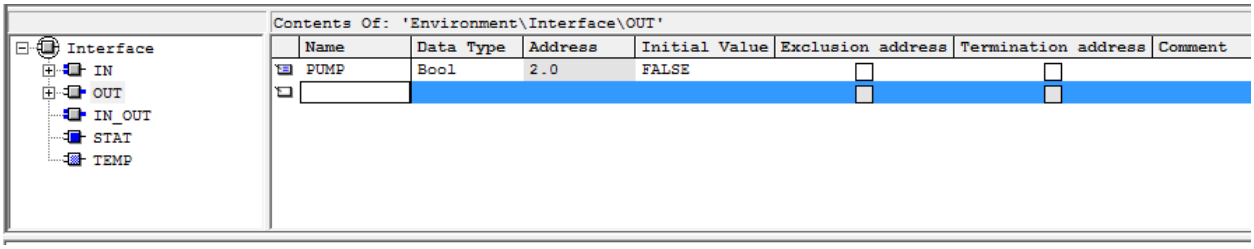
در روش Multiple Instance از یک FB اضافی برای کنترل و مدیریت سایر فراخوانی ها استفاده می شود به این صورت که با هر بار فراخوانی بلوک سطح پایین می بایست یک متغیر STAT در بلوک سطح بالاتر که نقش کنترلر را دارد، تعریف کرد بدین ترتیب در هنگام فراخوانی دیگر نیازی به اختصاص بلوک DB تمی باشد. در این حالت با فراخوانی سطح بالاتر تنها بایستی یک DB برای یکبار اختصاص داد.

به مثال زیر توجه نمایید.

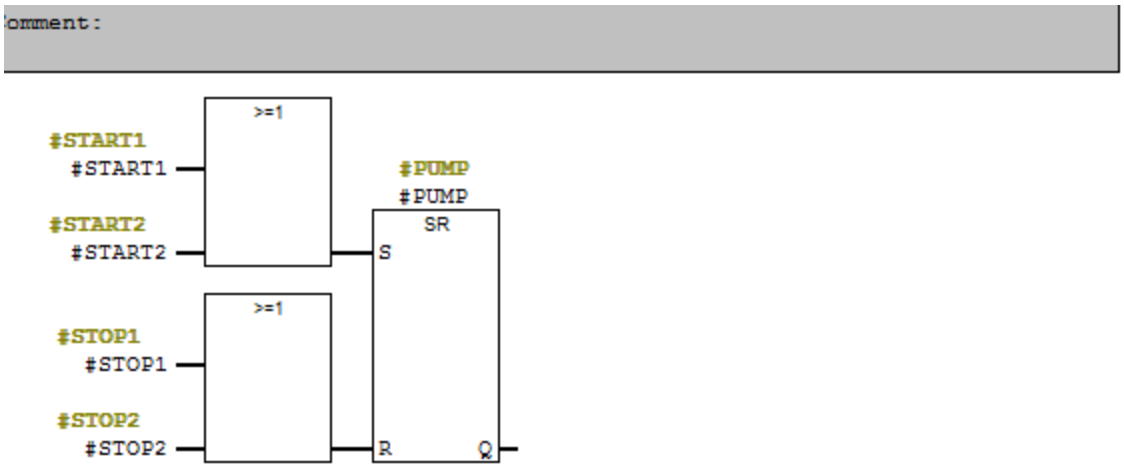
در این مثال ابتدا می خواهیم یک تابع برای کنترل یک پمپ که از دو نقطه فرمان می گیرد را به روش Instance Model برای 4 پمپ طراحی کنیم.

ابتدا بلوک FB1 را ایجاد و پارامترهای ورودی و خروجی آن را تعریف می کنیم.





حال در بلوک FB1 برنامه را می نویسیم.

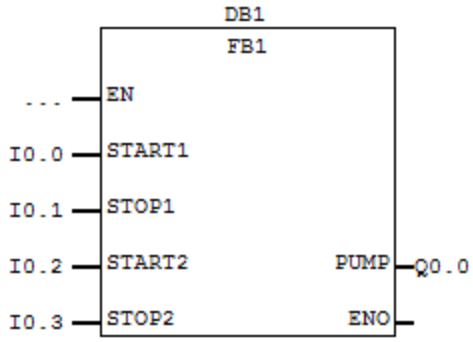


حال بلوک FB1 را در OB1 برای 4 پمپ فراخوانی می کنیم.

Comment:

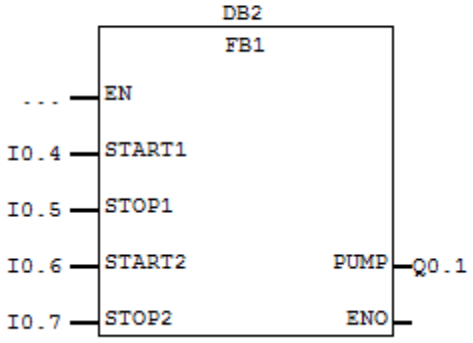
Network 1 : PUMP1

Comment:



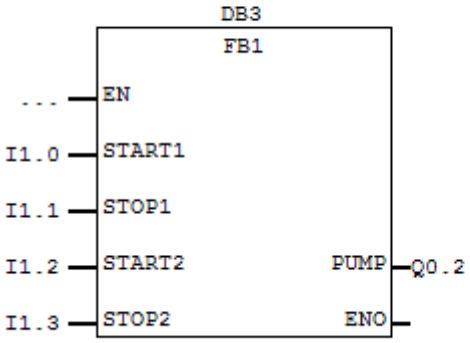
Network 2 : PUMP2

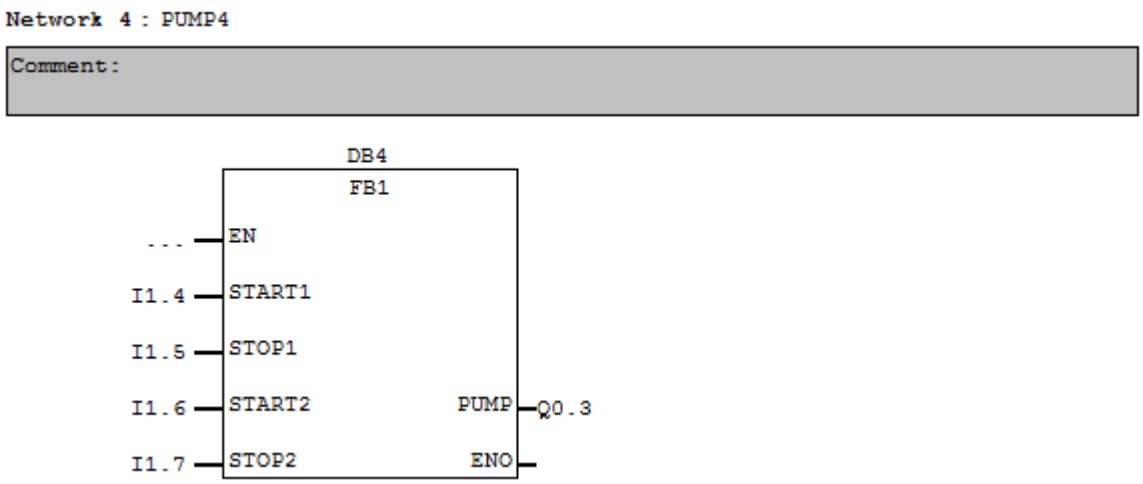
Comment:



Network 3 : PUMP3

Comment:



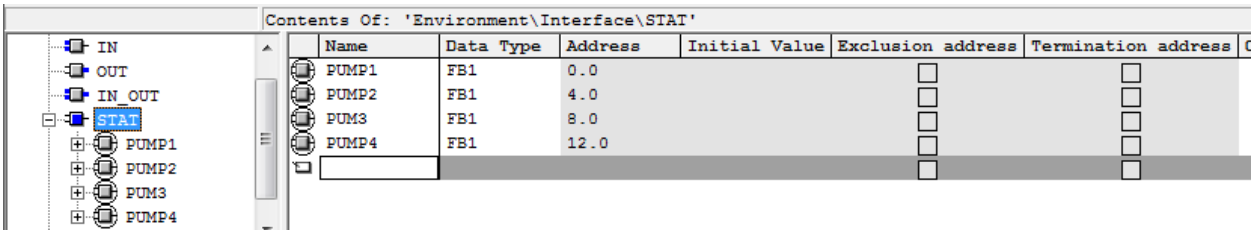


در این حالت از چهار بلوک DB برای ذخیره مقادیر FB استفاده شده است.حال مثال فوق را با استفاده از روش Multiple Instance که در آن یک بلوک DB برای هر 4 تابع فراخوانی استفاده میشود،بازنویسی می کنیم.

در این حالت ابتدا بلوک FB1 را مانند روش قبل ایجاد میکنیم و سپس بلوک FB2 را برای مدیریت فراخوانی ها ایجاد می کنیم.

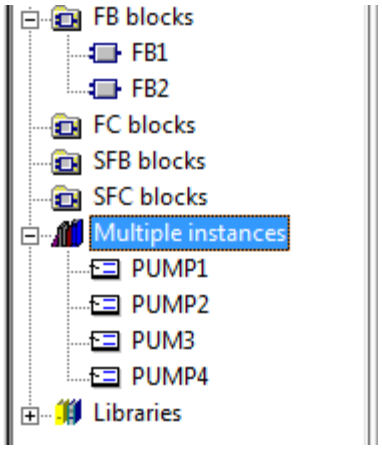
در این مرحله قبل از فراخوانی بلوک FB1 می بایست در بخش STAT به تعداد دفعاتی که قرار است تابع FB1 فراخوانی شود متغیرهایی با نام دلخوان تعریف شوند.

برای 4 پمپ،4 متغیر تعریف می کنیم.



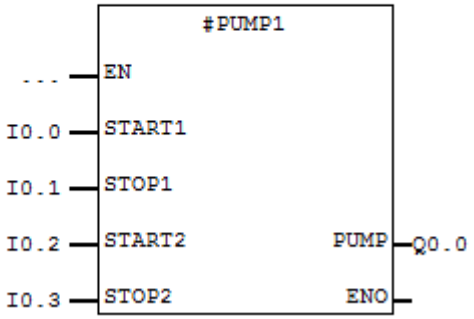
به نوع متغیرهای تعریف شده دقت نمایید که از نوع FB1 تعریف شده اند.

حال تابع را از بخش **Multiple Instance** فراخوانی میکنیم.



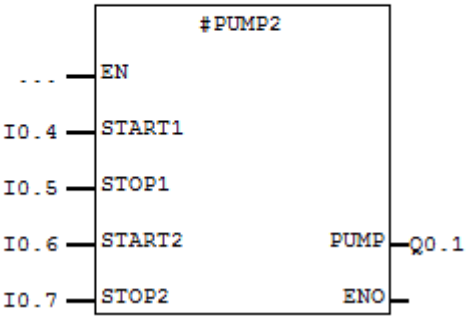
Network 1 : Title:

Comment:



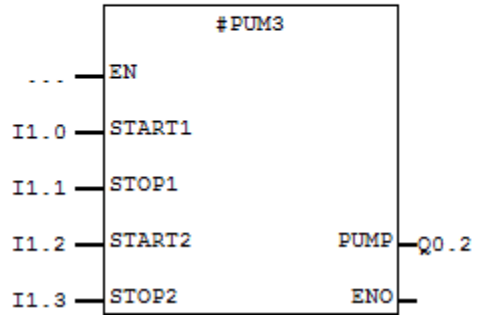
Network 2 : Title:

Comment:



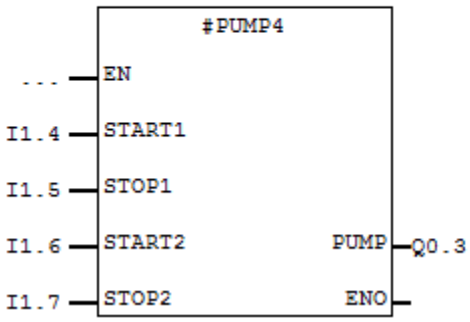
Network 3 : Title:

Comment:



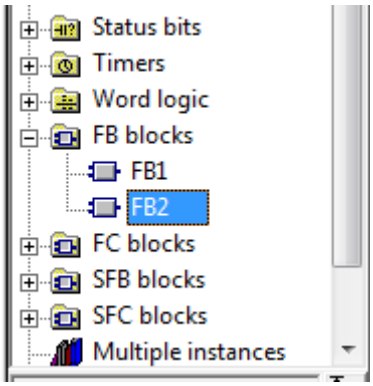
Network 4 : Title:

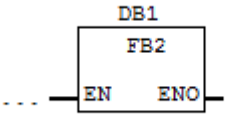
Comment:



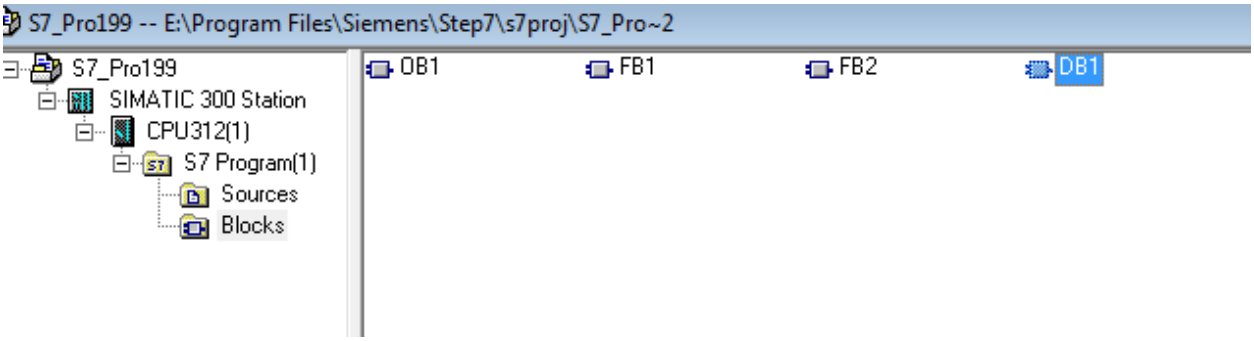
با دقت در شکل های فوق در می یابیم که در این حالت برای هر بار فراخوانی یک بلوک DB ایجاد نمی شود.

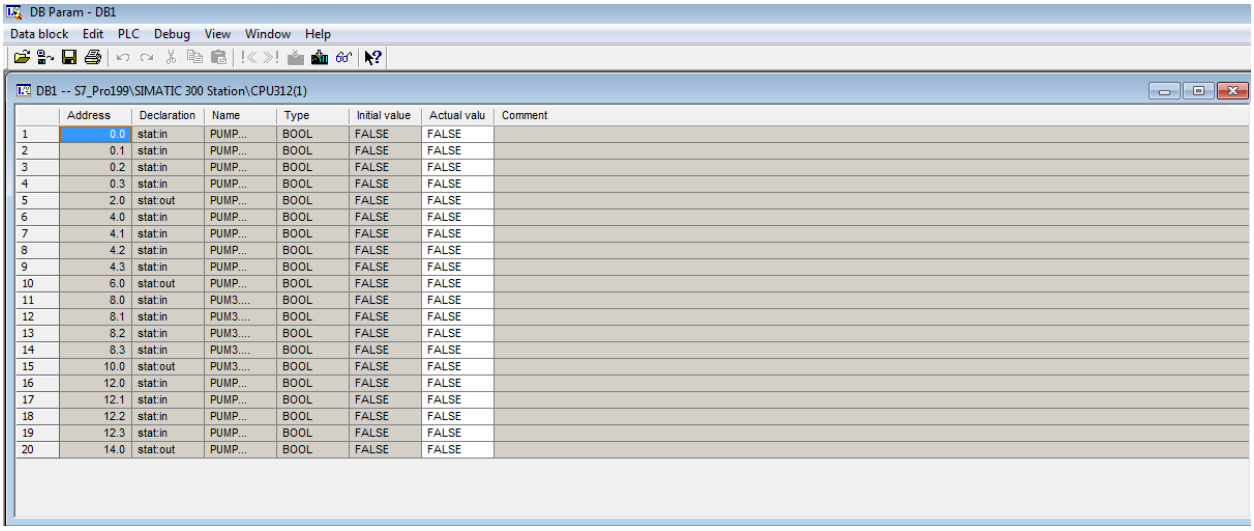
حال بلوک FB2 را در OB1 فراخوانی می کنیم.





با وارد شدن به محیط DB1 مشاهده می شود که دیتاهای هر 4 تابع در آن ذخیره شده است.بدین ترتیب در استفاده از فضای حافظه صرفه جویی قابل توجهی انجام می شود.





	Address	Declaration	Name	Type	Initial value	Actual valu	Comment
1	0.0	stat:in	PUMP...	BOOL	FALSE	FALSE	
2	0.1	stat:in	PUMP...	BOOL	FALSE	FALSE	
3	0.2	stat:in	PUMP...	BOOL	FALSE	FALSE	
4	0.3	stat:in	PUMP...	BOOL	FALSE	FALSE	
5	2.0	stat:out	PUMP...	BOOL	FALSE	FALSE	
6	4.0	stat:in	PUMP...	BOOL	FALSE	FALSE	
7	4.1	stat:in	PUMP...	BOOL	FALSE	FALSE	
8	4.2	stat:in	PUMP...	BOOL	FALSE	FALSE	
9	4.3	stat:in	PUMP...	BOOL	FALSE	FALSE	
10	6.0	stat:out	PUMP...	BOOL	FALSE	FALSE	
11	8.0	stat:in	PUM3....	BOOL	FALSE	FALSE	
12	8.1	stat:in	PUM3....	BOOL	FALSE	FALSE	
13	8.2	stat:in	PUM3....	BOOL	FALSE	FALSE	
14	8.3	stat:in	PUM3....	BOOL	FALSE	FALSE	
15	10.0	stat:out	PUM3....	BOOL	FALSE	FALSE	
16	12.0	stat:in	PUMP...	BOOL	FALSE	FALSE	
17	12.1	stat:in	PUMP...	BOOL	FALSE	FALSE	
18	12.2	stat:in	PUMP...	BOOL	FALSE	FALSE	
19	12.3	stat:in	PUMP...	BOOL	FALSE	FALSE	
20	14.0	stat:out	PUMP...	BOOL	FALSE	FALSE	

:(DB) Data Block

دیتا بلوکها بر خلاف سایر بلوک ها حاوی دستورات برنامه نیستند بلکه اطلاعات را در خود نگه می دارند و بردو نوع زیر هستند:

- دیتا بلوک اشتراکی (Shared DB)

از بین DB می توان در هر بلوک دیگر استفاده کرد اطلاعات آنها با بسته شدن DB از بین نمیرود.

- دیتا بلوک اختصاصی(Instance DB)

این بلوک به عنوان حافظه برای FB ها استفاده می شود.

حال به تشریح DB اشتراکی با بیان یک مثال می پردازیم

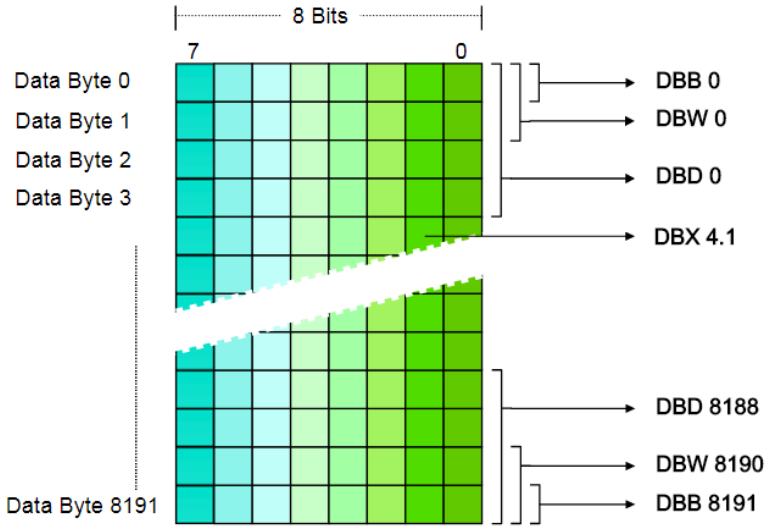
بلوک های DB را می توان در حالت اشتراکی در هر کجای برنامه به کار برد و در واقع توسط کاربر به وجود می آیند و از آنها به عنوان فضای حافظه استفاده می شود.

آدرس دهی برای DB

برای دسترسی به سطر های یک DB ابتدا لازم است شماره DB موردنظر مشخص شود.

شماره DBها به صورت عدد صحیح در بازه ی 1 تا 65535 است.

هر خط یک DB به صورت یک بایت است که در کاربرد های مختلف میتوان در محدوده BIT، Byte، Word، Dword و ...استفاده نمود.



آدرس دهی DB در محدوده ی Bit:

DBn.DBXm.u

که در آن n شماره DB . m شماره بایت. u شماره بیت

آدرس دهی DB در محدوده ی Byte:

DBn.DBBm

که در آن n شماره DB . m شماره بایت.

آدرس دهی DB در محدوده ی Word:

DBn.DBWm

که در آن n شماره DB . m شماره بایت شروع است و بایت بعدی راشامل میشود

آدرس دهی DB در محدوده ی Dword:

DBn.DBDm

که در آن n شماره DB . m شماره بایت شروع است و سه بایت بعدی را شامل میشود.

مثال :

DB2.DBW4 : آدرس در محدوده ی دو بایتی یا Word و آدرس بایت شروع 4 و در DB شماره 2

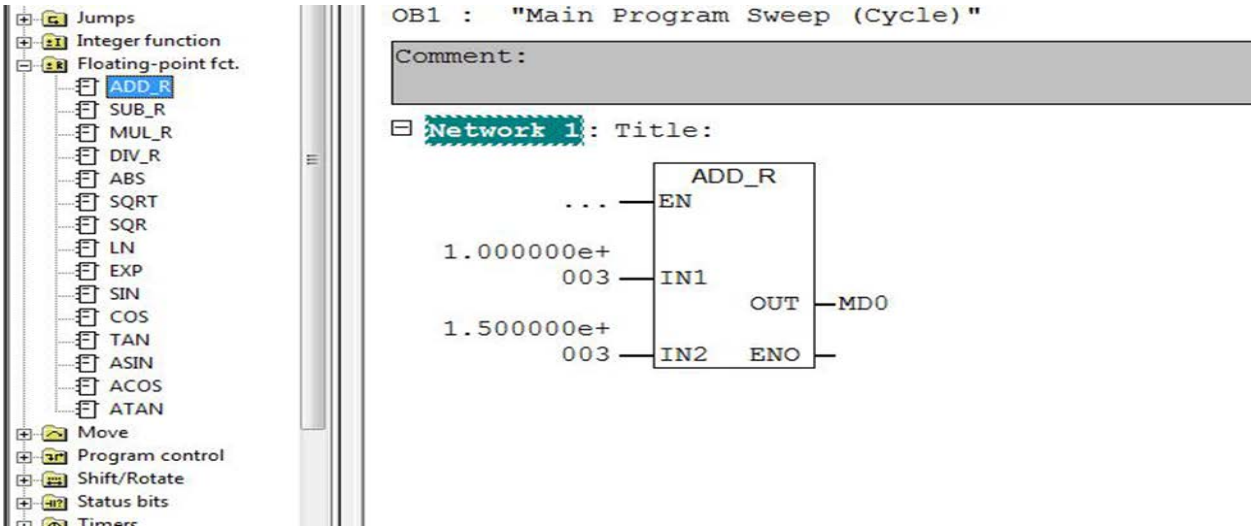
DB1.DBX3.5 : آدرس دهی در محدوده ی بیتی و بیت شماره 5 در بایت 3 و در DB شماره 1

DB43.DBD74 : آدرس دهی در محدودیه Dword یا چهاربایتی که آدرس بایت شروع 74 ودر DB شماره 43

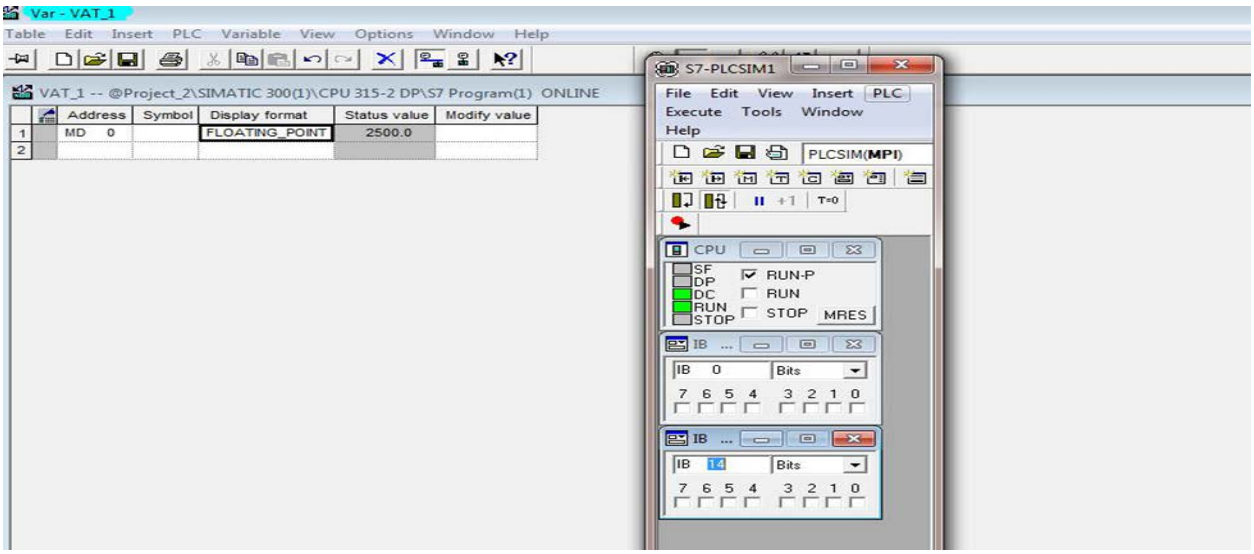
DB250.DBB203 : آدرس در محدوده بایت و آدرس 203 از DB شماره 250

حال با یک مثال به تشریح بیشتر DB اشتراکی می پرذازیم.

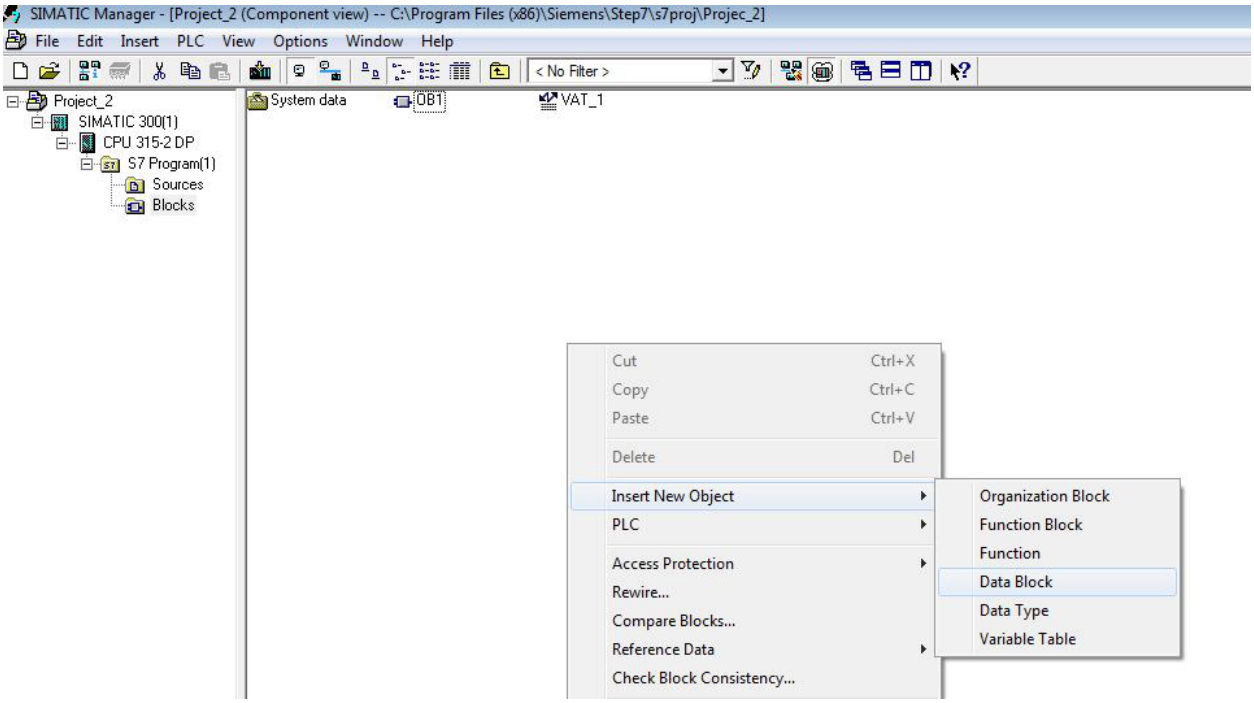
تا به حال برای ذخیره نتیجه جمع دو مقدار 1000 و 1500 از MD 0 استفاده میکردیم..یک بار دیگر این مثال را بازنویسی می کنیم.



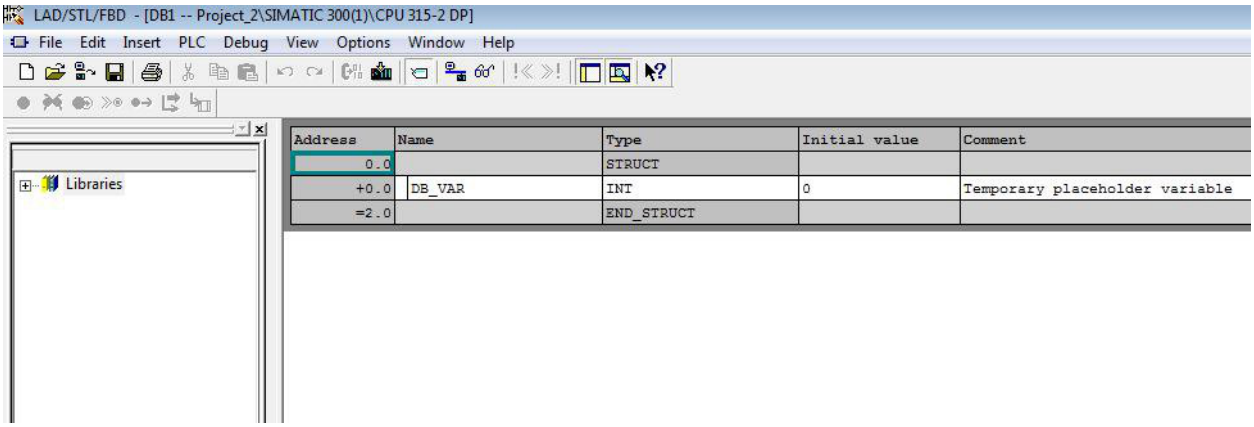
بعد از دانلود برنامه برای مشاهده نتیجه جدول VAT را ایجاد میکنیم



حال همین مثال را با استفاده از DB بازنویسی میکنیم.ابتدا برای ایجاد یک بلوکDB به مسیر نشان داده شده در شکل زیر میرویم.

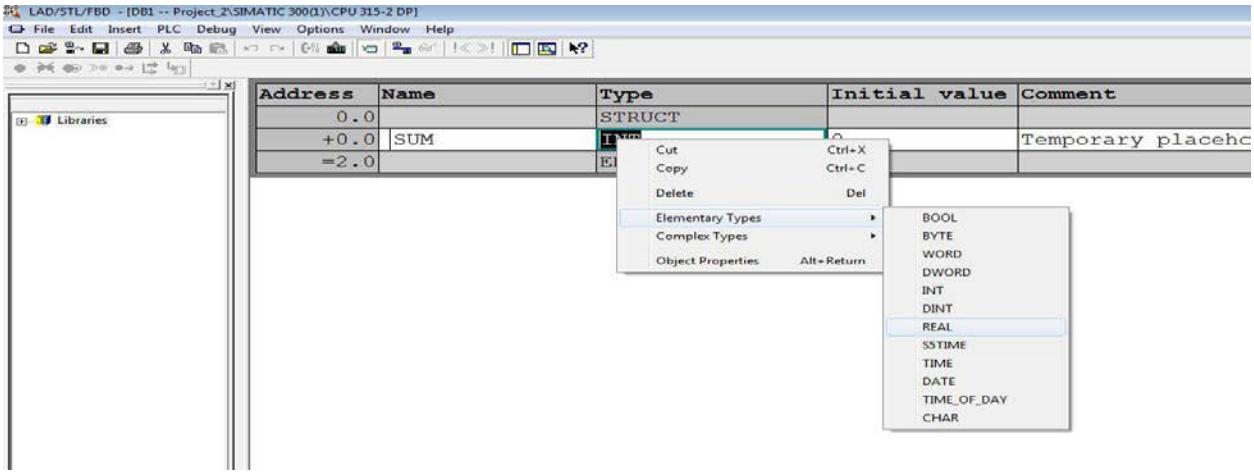


پس از ایجاد بلوک DB وارد محیط آن میشویم



ستون اول آدرس،ستون دوم نام دلخواه و ستون سوم نوع داده و در ستون بعد مقدار اولیه و در ستون پایانی پیغام خود را وارد می کنیم.

در قسمت NAME یک نام به دلخواه وارد می کنیم

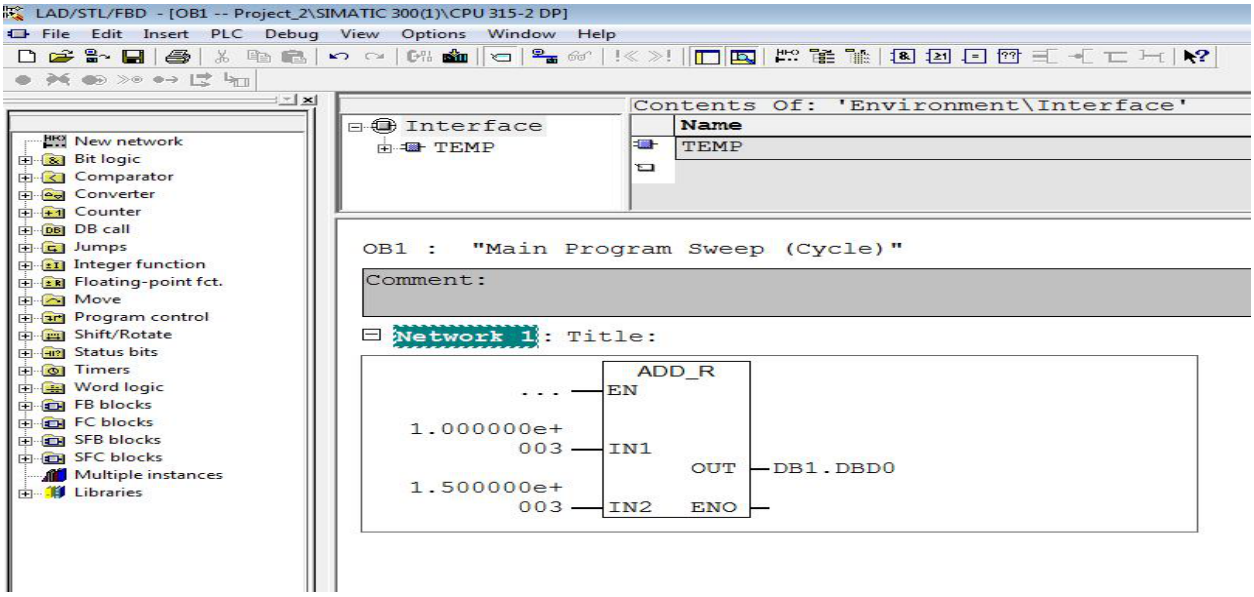


نوع داده و مقدار اولیه آن را در ستونهای مربوطه مشخص می کنیم.

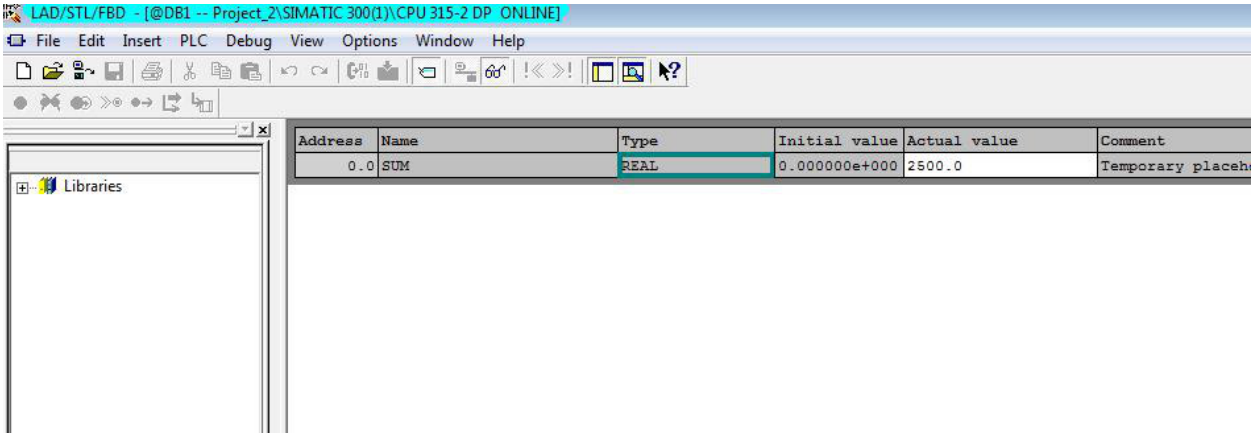
Address	Name	Type	Initial value	Comment
0.0		STRUCT		
+0.0	SUM	REAL	0.000000e+000	Temporary placeholder variable
=4.0		END_STRUCT		

آدرس این سطر DB1.DBD0 می باشد.

پس از ذخیره این بلوک وارد OB1 می شویم و برنامه مربوط به جمع را مینویسیم.



در ادامه در صفحه اصلی OB1 را به همراه DB1 به شبیه ساز دانلود می کنیم سپس وارد DB1 می شویم و آیکون عینک را فعال می کنیم

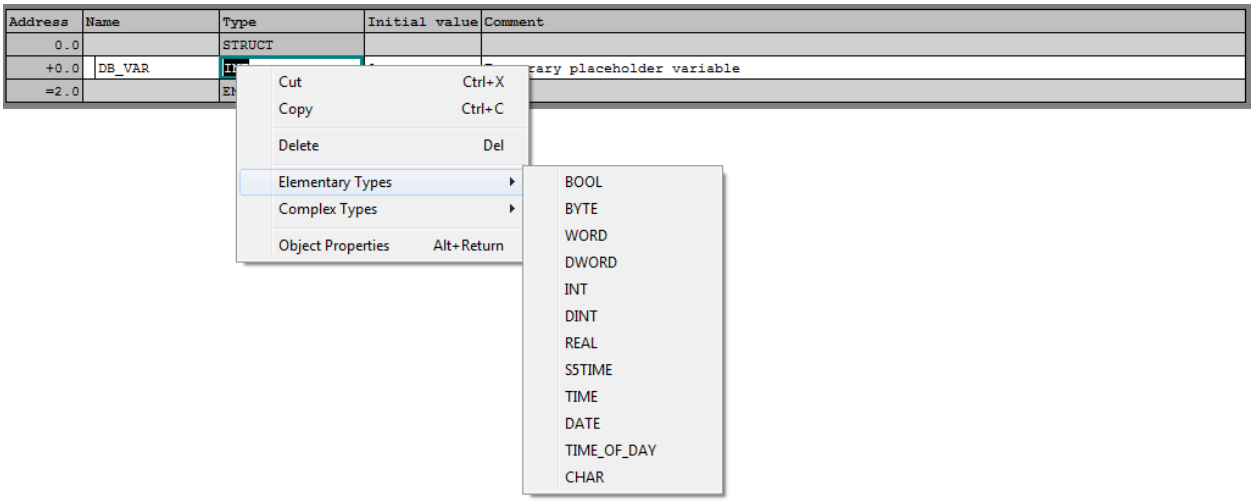


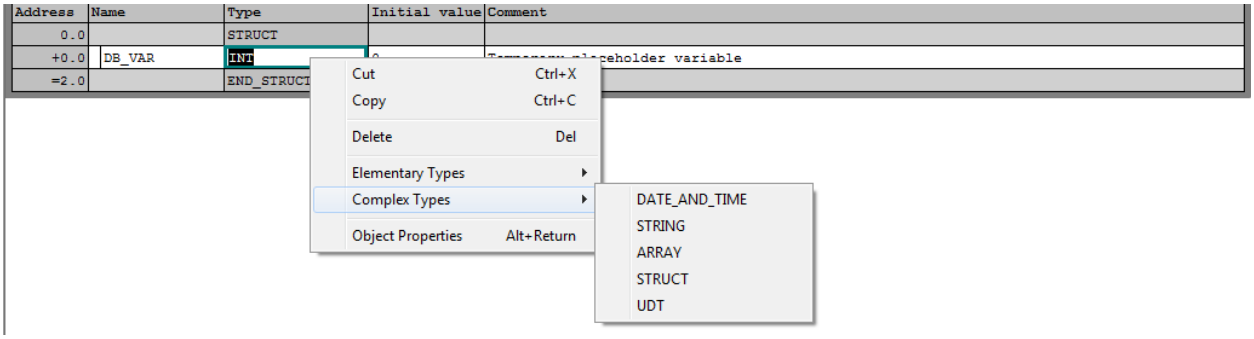
همانطور که ملاحظه می کنید، مقد ار 2500 در سطر اول ذخیره شد این سطر دارای آدرس می باشد و در تمام برنامه قابل دسترسی است.

معرفی انواع فرمت داده در DB اشتراکی

همانطور که میدانیم آدرس های DB از صفر شروع میشود و با توجه به تعداد بایت تعریف شده به شکل اتوماتیک آدرس ایجاد میشود.

در DB دو نوع تایپ در نظر گرفته شده است.یکی برای داده های تا 32 بیت(Elementry Types) مانند داده های با فرمت BOOL، BYTE، WORD،REAL و غیره و دیگری برای داده های با حجم بیش از 32 بیت(Complex Types).





با انواع داده های نوع Elementry Types قبلا آشنا شده ایم.

حال به معرفی فرمت داده ها با حجم بیش از 32 بیت می پردازیم.

فرمت Date_And_Time

از این دیتا برای ذخیره سازی تاریخ و زمان استفاده می شود و فضایی که اشغال میکند 64 بیت است.ساختار این داده به شکل D#YY.MM.DD.HH:MM:S:MS میباشد.

Address	Name	Type	Initial value	Comment
0.0		STRUCT		
+0.0	DB_VAR	DATE_AND_TIME	DT#16-8-12-10:36:15.000	
=8.0		END_STRUCT		

فرمت STRING

حداکثر حجم این نوع داده 254 بایت میباشد و برای نمایش متن در سیستم های مانیتورینگ استفاده می شود.نحوه نمایش ان به صورت نشان داده شده در شکل زیر است.

Address	Name	Type	Initial value	Comment
0.0		STRUCT		
+0.0	DB_VAR	STRING[254]	'text'	
=8.0		END_STRUCT		

پیغام مورد نظر را در قسمت Initial value وارد میکنیم.لازم به ذکر است که این پیغام در محیط دیتا بلاک قابل مانیتور کردن نمی باشد.

فرمت ARRAY

از این فرمت برای جلوگیری از تکرار داده ها با فرمت یکسان استفاده می شود.در واقع متغیرهای با تایپ یکسان توسط فرمت آرایه اعریف می شوند.

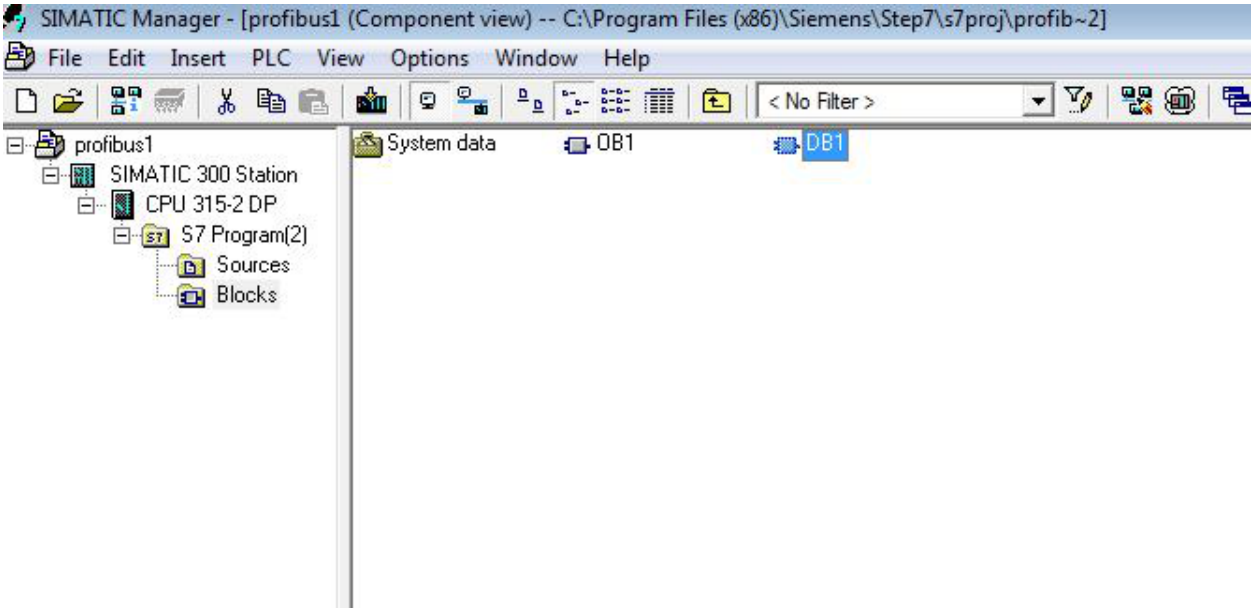
آرایه تایپی است که می تواند شامل n المان باشد.تمامی عناصر یا المان های آرایه باید یک تایپ مشابه داشته باشند.(مثلا همگی از نوع Real)

یک آرایه همانند یک ماتریس می تواند یک بعدی و یا چند بعدی تعریف شود که هر بعد نیز دارای سطر و ستون هستند.

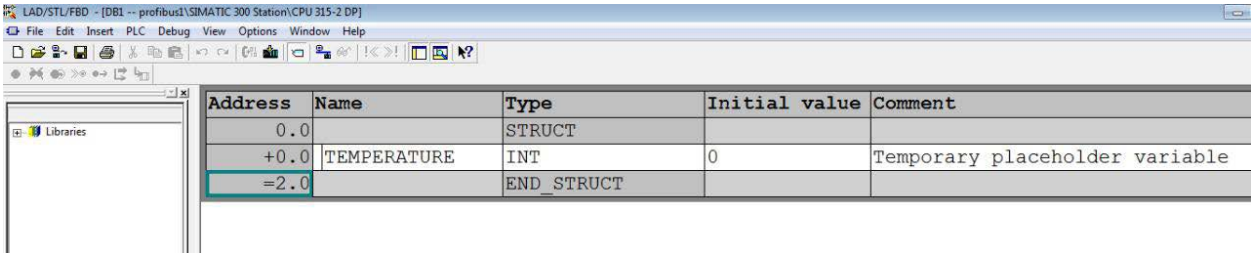
مثال) فرض کنید در یک کوره صنعتی 50 سنسور دما وجود دارد می خواهیم مقادیر دما را از سنسورها بخوانیم و در یک دیتا بلاک ذخیره کنیم(مقدار دما هم real می باشد)

در این مثال فرض بر این است که همگی داده ها از نوع Real هستند.

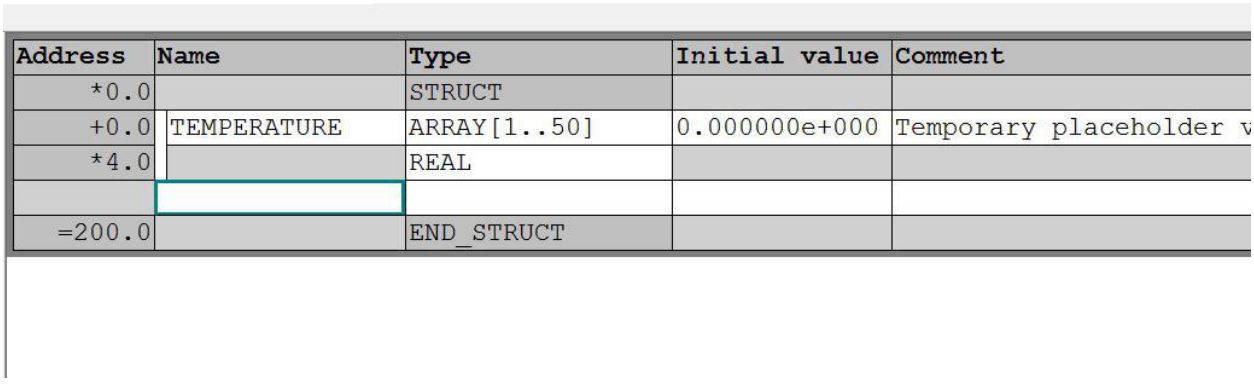
ابتدا یک DB ایجاد می کنیم.



وارد DB شوید و نام را TEMPERATURE وارد کنید

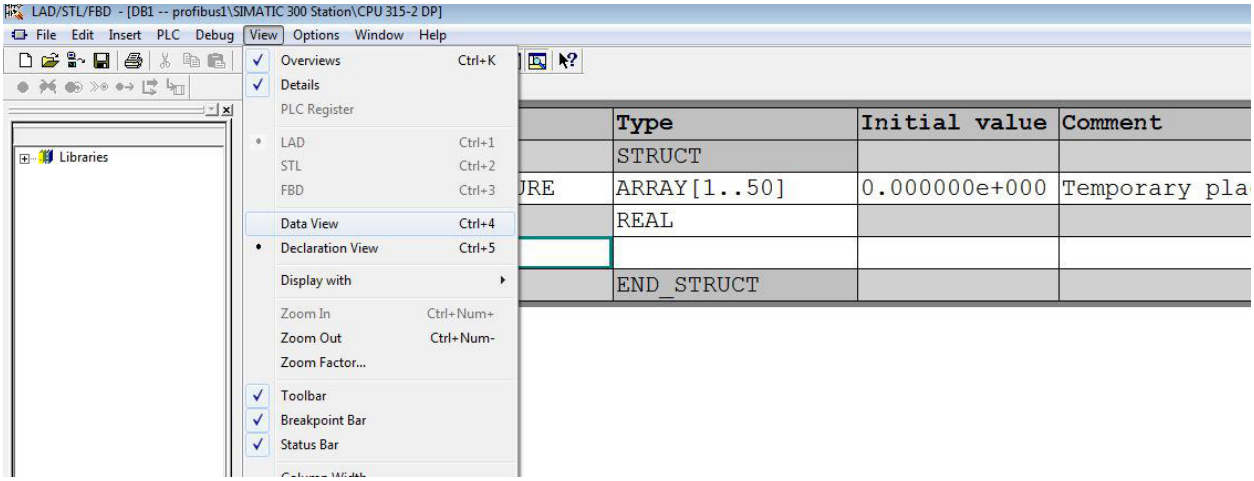


سپس با توجه به مثال داده شده فرمت داده را یک آرایه 50 سطری وارد کنید و مقدار اولیه را 0.0 وارد کنید. تایپ المان ها را REAL قرار دهید.

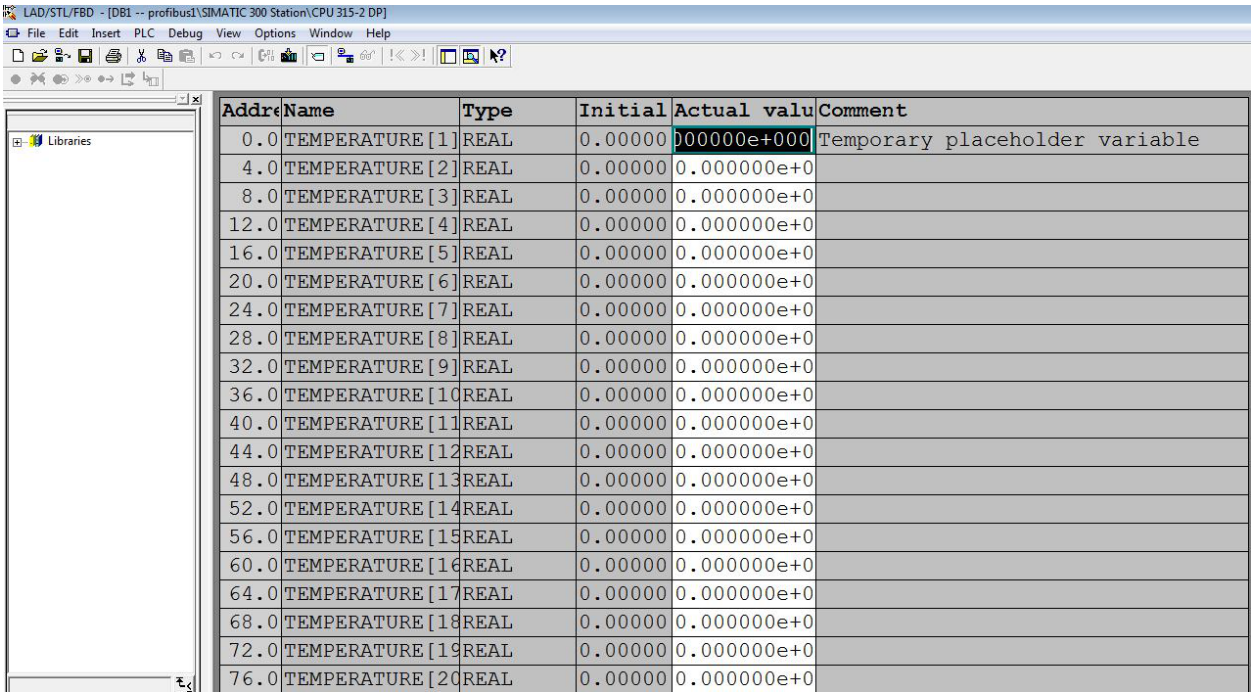


Address	Name	Type	Initial value	Comment
*0.0		STRUCT		
+0.0	TEMPERATURE	ARRAY[1..50]	0.000000e+000	Temporary placeholder v
*4.0		REAL		
=200.0		END_STRUCT		

برای مشاهده المان های تابع تعریف شده به مسیر نشان داده شده در شکل زیر میرویم.



با زدن گزینه فوق لیست المان های تابع ظاهر می شود.

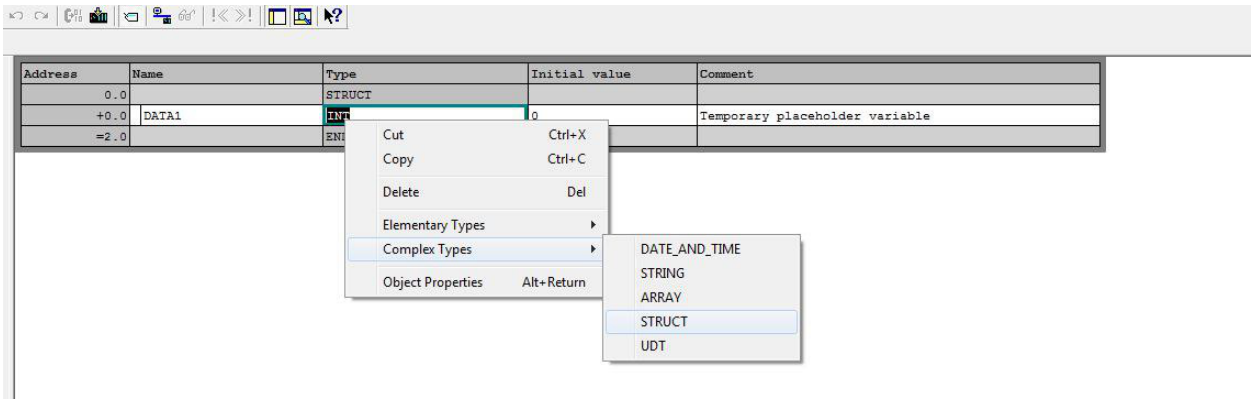


فرمت STRUCT

از این فرمت می توانیم برای دسته بندی داده ها با تایپ متفاوت استفاده کرد .

به مثال زیر توجه کنید

یک DB بسازید



Address	Name	Type	Initial value	Comment
0.0		STRUCT		
+0.0	DATA1	STRUCT		Temporary placeholder variable
=0.0		END_STRUCT		
=0.0		END_STRUCT		

Address	Name	Type	Initial value	Comment
0.0		STRUCT		
+0.0	DATA1	STRUCT		Temporary placeholder variable
+0.0	D1	REAL	0.000000e+000	
+4.0	D2	BOOL	FALSE	
+6.0	D3	INT	0	
+8.0	D4	REAL	0.000000e+000	
=12.0		END_STRUCT		
=12.0		END_STRUCT		

یک استراکچر با نام DATA1 داریم که دارای 4 زیرمجموعه است که تایپ المانها با هم فرق می کند

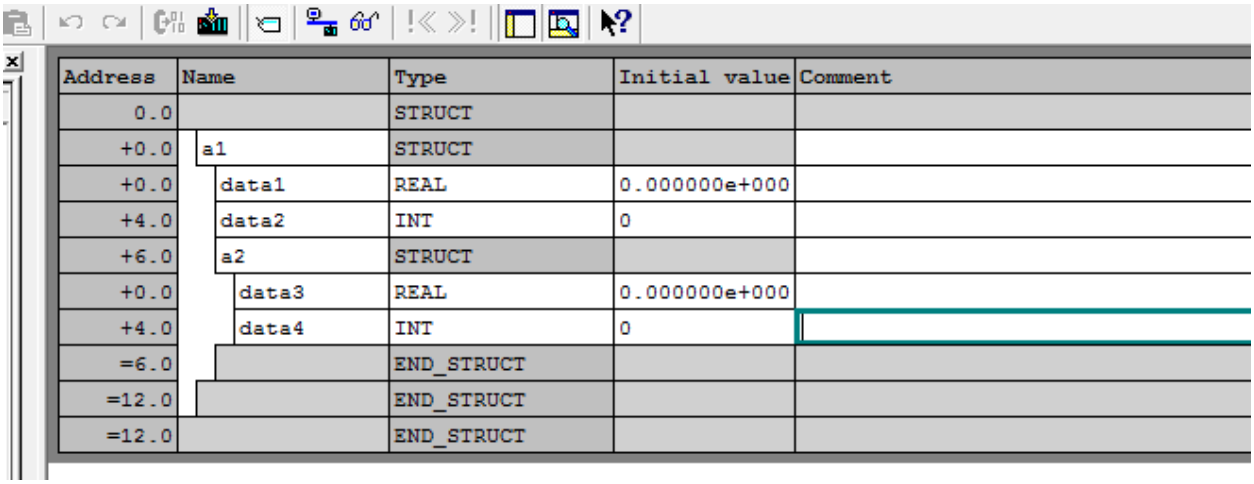
پس در استراکچر برخلاف آرایه ، می توان از داده های با تایپ های متفاوت استفاده کرد.

با کمی دقت در آدرس ها درمیابیم که ظاهرا آدرس المانها در داخل استراکچر با آدرس های خارج از محدوده استراکچر تداخل دارد ولی در واقع چنین نیست زیرا آدرس هر پارامتر در داخل استراکچر با آدرس خود استراکچر جمع می شود.

برای مشاهده آدرسهای واقی می توان به مسیر View>>>Data view مراجعه کنید.

به مثال زیر دقت کنید.

همانطور که پیشتر گفته شد در استراکچر می توانیم داده ها با تایپ متفاوت را به صورت معمولی یا تو در تو دسته بندی کنیم.به این صورت که برای یک ساختار نامی را انتخاب و سپس داده ها با تایپ متفاوت را در قرار میدهیم.این عمل در در نهایب به دستیابی راحت تر داده ها می انجامد.



Address	Name	Type	Initial value	Comment
0.0		STRUCT		
+0.0	a1	STRUCT		
+0.0	data1	REAL	0.000000e+000	
+4.0	data2	INT	0	
+6.0	a2	STRUCT		
+0.0	data3	REAL	0.000000e+000	
+4.0	data4	INT	0	
=6.0		END_STRUCT		
=12.0		END_STRUCT		
=12.0		END_STRUCT		

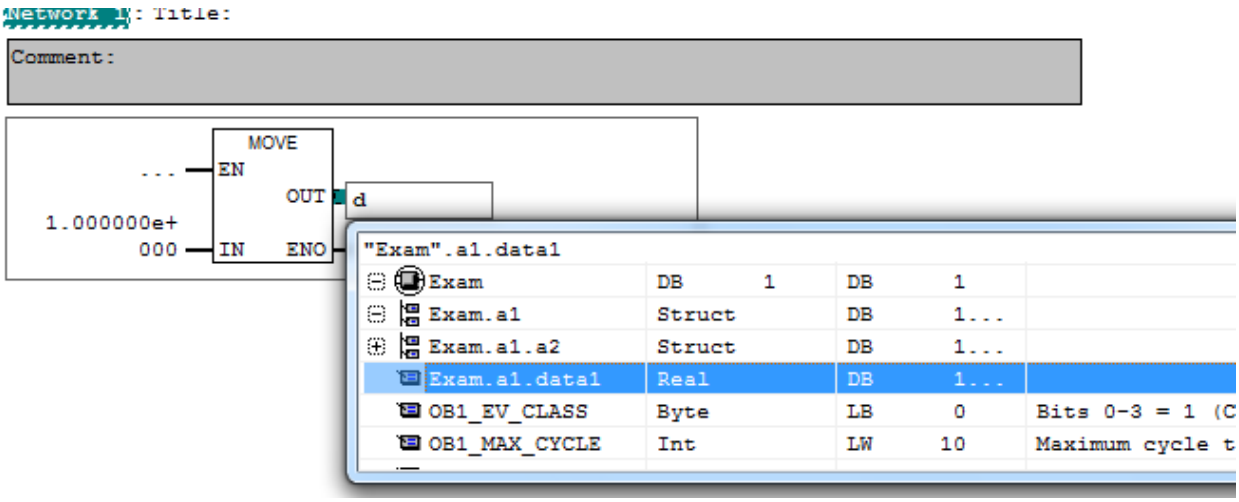
همانطور که در مثال فوق مشاهده می کنید در ستون آدرس آدرس های 0.0 و 4.0 تکرار شده اند علت این امر این است که آدرس هر پارامتر در داخل استراکچر با آدرس خود استراکچر جمع می شود.

در این مثال a2 زیر مجموعه a1 و a1 زیر مجموعه استراکچر است به همین دلیل در a2 نیز با آدرس 0.0 و 0.4 مواجه میشویم.

اما در واقع داده ها در یک آدرس قرار ندارند.

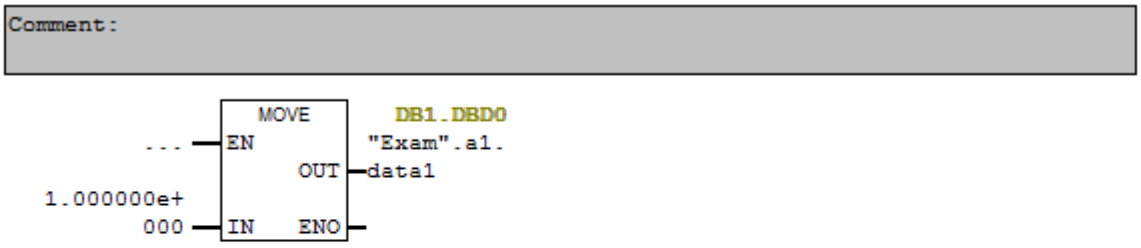
به محیط OB1 رفته و یک برنامه فرضی می نویسیم(میدانیم که در داده های با حجم 32 بیت نمیتوان از دستورات معمولی مانند MOVE استفاده کرد و باید بلوک های سیستمی را به کار برد ولی به دلیل اینکه در حال حاضر با بلوکهای سیستمی آشنایی نداریم و هدف تنها مشاهده آدرس واقعی data1 تا data4 است از دستور MOVE به شکل فرضی استفاده کرده ایم)

در زمان ایجاد بلوک DB برای آن سمبل Exam انتخاب شده است.حال برای دستیابی به هر دیتا به مسیر نشان داده شده در شکل زیر می رویم.

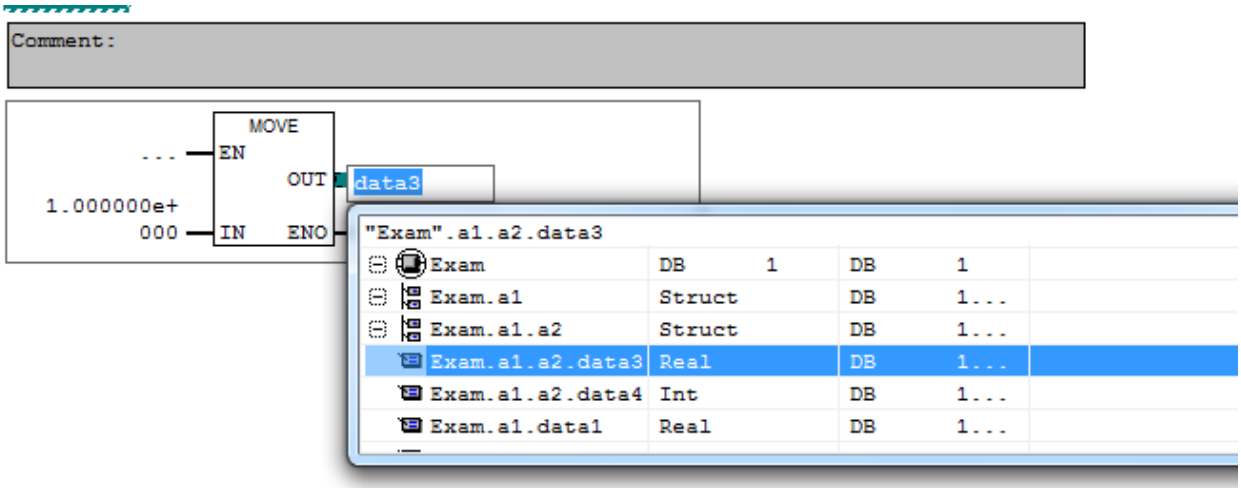


با کمی دقت در شکل فوق در می یابیم که data1 زیرمجموعه a1 و خود a1 زیر مجموعه Exam است.

با انتخاب data1 و زدن اینتر به شکل اتوماتیک آدرس آن ظاهر می شود

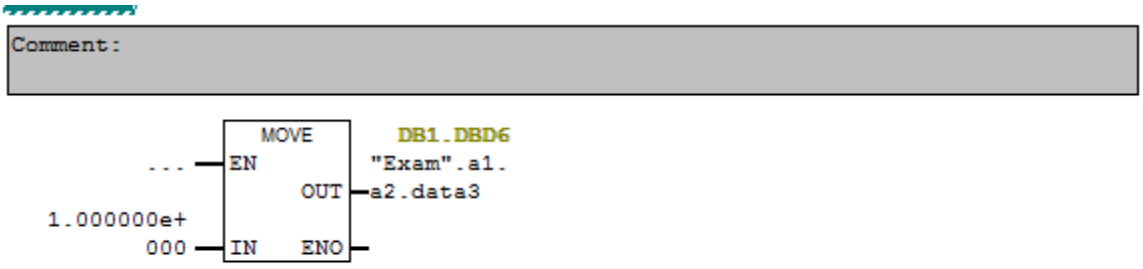


برای وارد کردن data3 بار دیگر مراحل قبل را تکرار می کنیم.

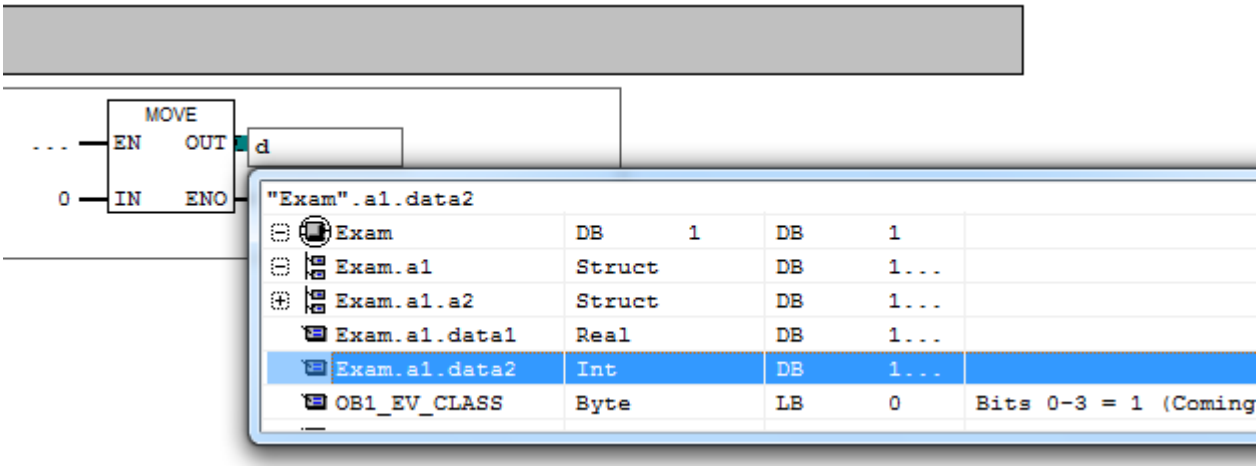


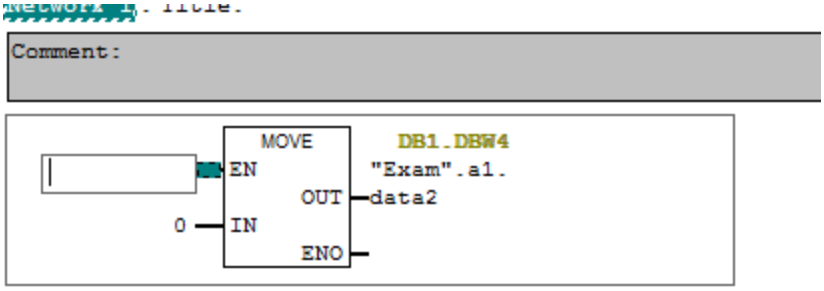
همانطور که در شکل فوق مشخص است data3 زیر مجموعه a2 است و a2 خود زیر مجموعه a1 می باشد

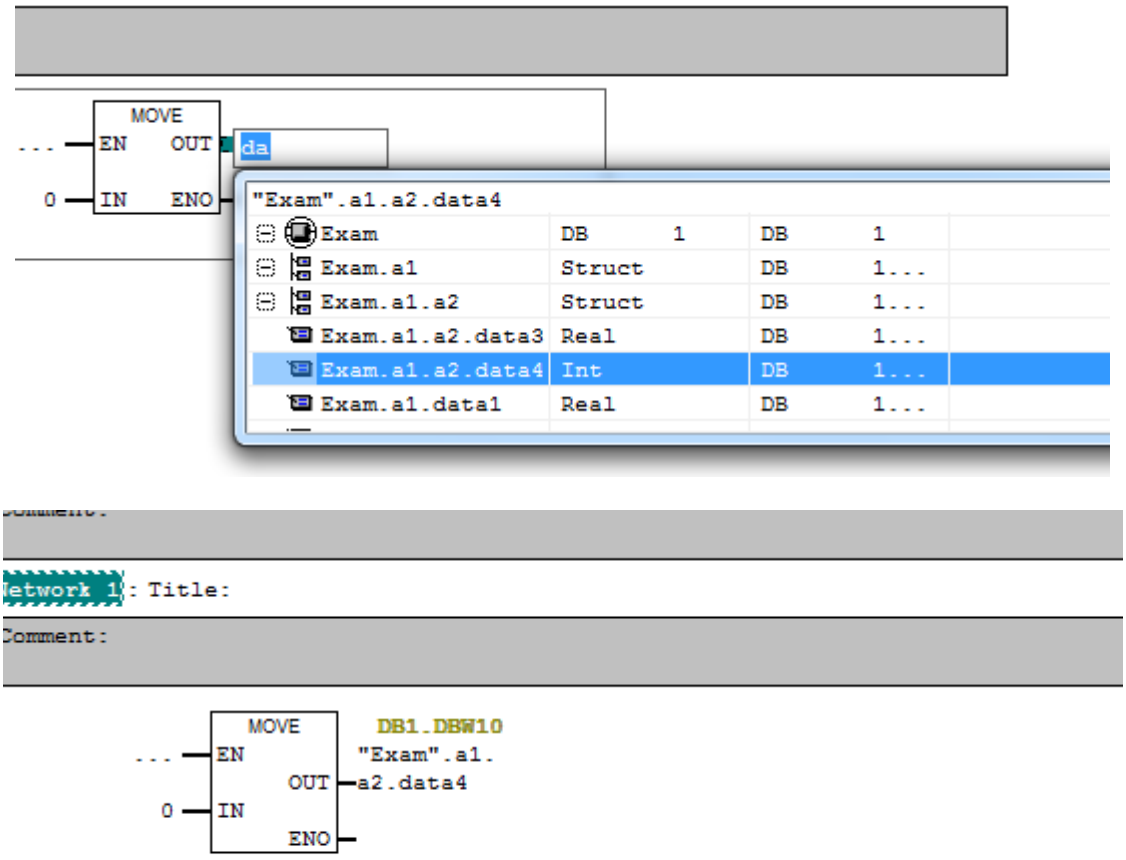
با زدن کلید اینتر و ظاهر شدن آدرس data3 می بینیم که اگر چه در ظاهر در داخل دیتا بلاک dara1 و data3 آدرس مشابه دارند ولی در حقیقت چنین نیست و این تکرار آدرس تنها به دلیل نوع خاص آدرس دهی در استراکچرهای تو در تو است.



و در مورد data2 و data4 نیز دقیقا همین مطلب صادق است


MOVE
EN
OUT
d
0
IN
ENO
"Exam".a1.data2
Exam DB 1 DB 1
Exam.a1 Struct DB 1...
Exam.a1.a2 Struct DB 1...
Exam.a1.data1 Real DB 1...
Exam.a1.data2 Int DB 1...
OB1_EV_CLASS Byte LB 0 Bits 0-3 = 1 (Coming


Comment:
MOVE
EN
OUT
0
IN
ENO
DB1.DBW4
"Exam".a1.
data2



فرمت UDT

از این فرمت برای ذخیره تعدادی از پارامترها جهت بکارگیری در بلوک های DB و سایر بلوک ها استفاده می شود.در مورد فرمت UDT توجه به این نکته که یک UDT می تواند شامل انواع داده های Elementary و Complex حایز اهمیت است.لازم به ذکر است که بلوک UDT به PLC دانلود نمی شود و همانطور که گفته شد تنها برای استفاده در بلوکهای DB و سایر بلوکها مورد استفاده قرار می گیرد.به دلیل امکان تعریف داده ها با انواع مختلف و همچنین استفاده از UDT برای چند ساختار مشابه،زمان ایجاد دیتاها در بلوک های DB تا حد قابل توجهی کاهش می یابد.

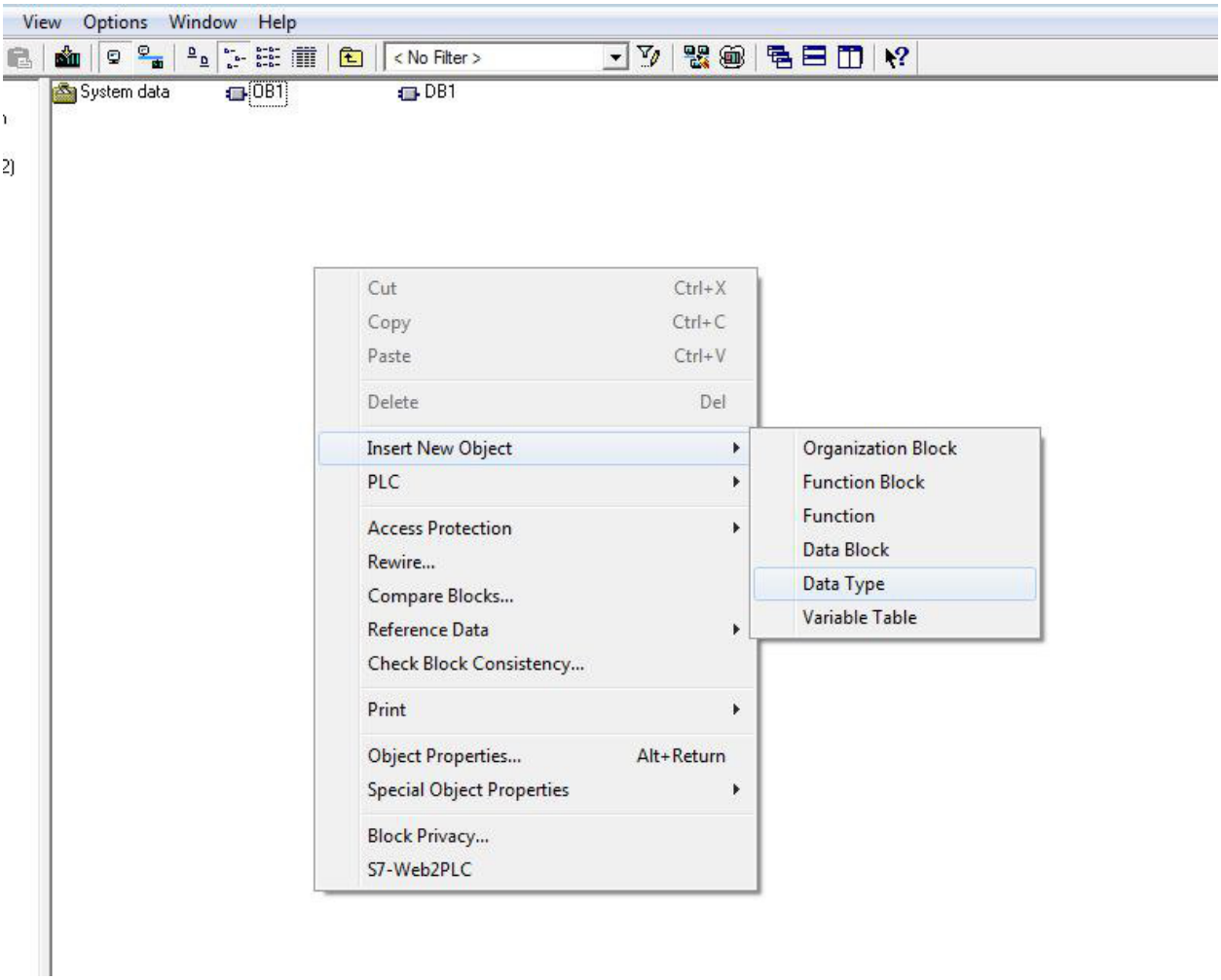
به مثال زیر توجه کنید

مثال)در یک پروسه صنعتی فرض کنید می خواهیم از 5 موتور دیتاهای زیر را بخوانیم و در یک DB ثبت کنیم.

سرعت:REAL،گشتاور: REAL ،جهت: BIT،خطا: BIT ،جریان موتور: INT،فعال بودن موتور: BIT و

مقدار فرکانس: INT

بر روی گزینه DATA TYPE کلیک کنید



وارد محیط UDT1 می شویم

پارامترهای مربوط به یک موتور را در UDT1 وارد می کنیم

Address	Name	Type	Initial value	Comment
0.0		STRUCT		
+0.0	FRQ	REAL	0.000000e+000	
+4.0	SPEED	INT	0	
+6.0	TRQ	REAL	0.000000e+000	
+10.0	FAULT	BOOL	FALSE	
+10.1	DIRECTION	BOOL	FALSE	
+10.2	RUNNING	BOOL	FALSE	
+12.0	CURRENT	INT	0	
=14.0		END_STRUCT		

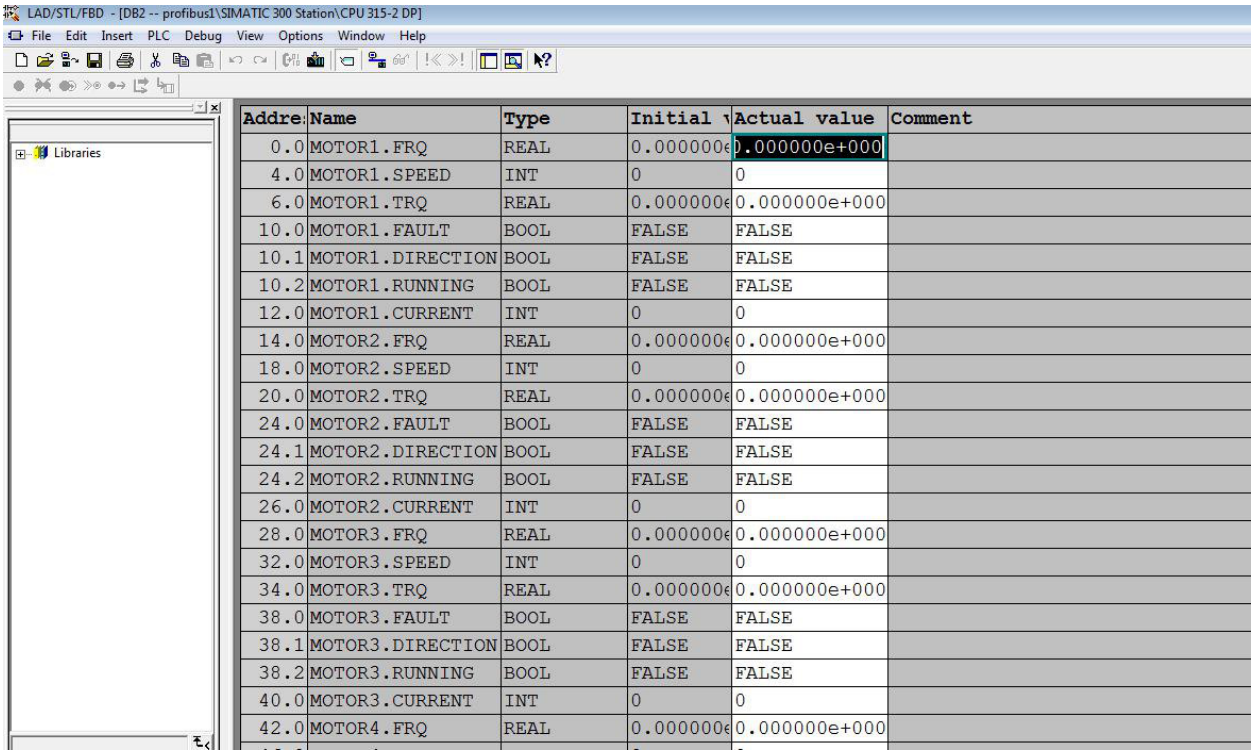
بعد از ذخیره UDT از آن خارج ویک DB ایجاد کرده و سپس وارد محیط DB میشویم.

Address	Name	Type	Initial value	Comment
0.0		STRUCT		
+0.0	MOTOR1	UDT1		Temporary placeholder variable
=14.0		END_STRUCT		

این کار را برای 5 موتور تکرار می کنیم

Address	Name	Type	Initial value	Comment
0.0		STRUCT		
+0.0	MOTOR1	UDT1		Temporary placeholder variable
+14.0	MOTOR2	UDT1		
+28.0	MOTOR3	UDT1		
+42.0	MOTOR4	UDT1		
+56.0	MOTOR5	UDT1		
=70.0		END_STRUCT		

سپس گزینه DATA VIEW را فعال می کنیم



Addre	Name	Type	Initial	Actual value	Comment
0.0	MOTOR1.FRQ	REAL	0.000000e	0.000000e+000	
4.0	MOTOR1.SPEED	INT	0	0	
6.0	MOTOR1.TRQ	REAL	0.000000e	0.000000e+000	
10.0	MOTOR1.FAULT	BOOL	FALSE	FALSE	
10.1	MOTOR1.DIRECTION	BOOL	FALSE	FALSE	
10.2	MOTOR1.RUNNING	BOOL	FALSE	FALSE	
12.0	MOTOR1.CURRENT	INT	0	0	
14.0	MOTOR2.FRQ	REAL	0.000000e	0.000000e+000	
18.0	MOTOR2.SPEED	INT	0	0	
20.0	MOTOR2.TRQ	REAL	0.000000e	0.000000e+000	
24.0	MOTOR2.FAULT	BOOL	FALSE	FALSE	
24.1	MOTOR2.DIRECTION	BOOL	FALSE	FALSE	
24.2	MOTOR2.RUNNING	BOOL	FALSE	FALSE	
26.0	MOTOR2.CURRENT	INT	0	0	
28.0	MOTOR3.FRQ	REAL	0.000000e	0.000000e+000	
32.0	MOTOR3.SPEED	INT	0	0	
34.0	MOTOR3.TRQ	REAL	0.000000e	0.000000e+000	
38.0	MOTOR3.FAULT	BOOL	FALSE	FALSE	
38.1	MOTOR3.DIRECTION	BOOL	FALSE	FALSE	
38.2	MOTOR3.RUNNING	BOOL	FALSE	FALSE	
40.0	MOTOR3.CURRENT	INT	0	0	
42.0	MOTOR4.FRQ	REAL	0.000000e	0.000000e+000	

همانطور که مشاهده می کنید، UDTبه تمامی موتورها اختصاص داده شد و در نتیجه کار خیلی سریعتر انجام شد

پس توسط UDT می توان یکسری داده با تایپ مختلف را آماده کرد و در DB مدام از آن استفاده نمود.

سنسورها

و

کارتهای آنالوگ

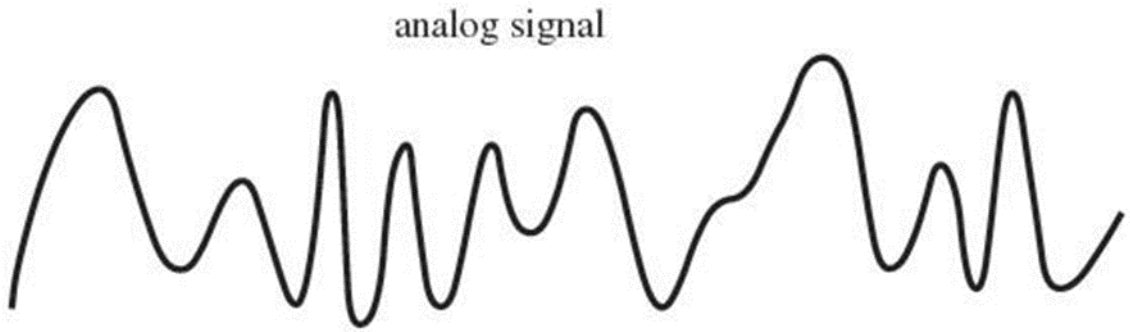
تعریف کمیت آنالوگ: به کمیت هایی گفته می شود که مقدارشان در بازه زمان متغیر است. برخی کمیت های آنالوگ عبارتند از: دما، فشار، رطوبت، سرعت، فلو(میزان جریان مایع)، فاصله و... .

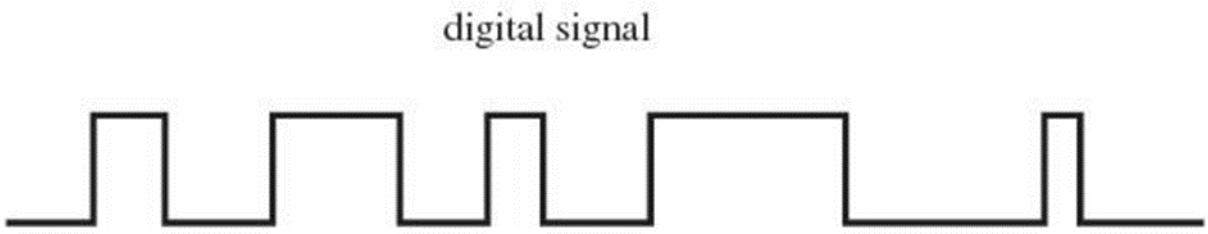
سنسورهایی که این کمیت ها را اندازه گیری می کنند نیز به عنوان سنسورهای آنالوگ نام گذاری می شوند.

سنسورهای آنالوگ ورودی

برای پردازش سیگنال های آنالوگ در PLC نیاز به کارت ورودی آنالوگ می باشد.

برخلاف سیگنال های دیجیتال که ماهیتی گسسته بین دو مقدار صفر منطقی و یک منطقی دارند سیگنال های آنالوگ ماهیتی پیوسته دارند.





سنسورهای آنالوگ یک کمیت فیزیکی با ماهیت پیوسته را به سیگنال الکتریکی پیوسته تبدیل میکند. خروجی این سنسورها عموماً از نوع ولتاژ یا جریان یا مقاومت می باشد.

هر وسیله اندازه گیری آنالوگ از دو بخش تشکیل می شوند:

1- **ترانسدیوسر:** مبدلی که کمیت مورد اندازه گیری را حس کرده و نتیجه را به سیگنال های الکتریکی تبدیل می نماید.

2- **ترانسمیتر:** وسیله ای است که سیگنال های الکتریکی غیر استاندارد را از ترانسدیوسر دریافت کرده و آن را به سیگنال الکتریکی استاندارد جهت ارسال به PLC تبدیل میکند.

سیگنالهای الکتریکی استاندارد:

جریانی:	[4 , 20] mA , [0 , 20] mA , [-20 , +20] mA
ولتاژی:	[0, 10] V , [1 , 5] V , [-10 , +10] V , [-5 , +5] V

در بحث کارتهای ورودی آنالوگ نکات زیر بایستی مورد توجه قرار گیرد: تعداد کانال کارت، دقت کارت،رنج جریان و ولتاژ،....

کارت نشان داده شده در شکل زیر دارای 8 کانال ورودی میباشد که به صورت AI*8 نشان داده می شود.

هر چقدر دقت کارت مورد استفاده بیشتر باشد، کارت قادر است سیگنال های کوچکتری را آشکار کند،یعنی کارت با قدرت تفکیک یا رزولوشن بالاتر، که به همان میزان قیمت کارت نیز بیشتر میشود.

حال به بررسی برخی سنسورهای آنالوگ می پردازیم.

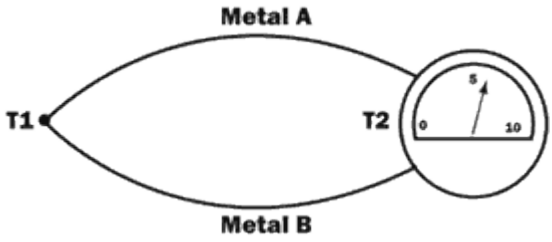
سنسورهایی مثل :

- سنسور دما که شامل دو سنسور زیر است:
  - ترموکوپل (TC)
  - ترمومتر(RTD)
- سنسور وزن(Load Cell)
- فلومتر
- سنسور فشار (Pressure Transmiter)
- سنسور سطح (Level Transmiter)
- سنسور فاصله

لازم به ذکر است که خروجی هر سنسور به ازای اندازه گیری کمیت،یک سیگنال از جنس ولتاژ ، جریان و یا مقاومت می باشد.

ترموکوپل (TC)

ترموکوپل یکی از پر مصرف ترین سنسورهای اندازه گیری دما محسوب میگردد. این سنسورهای ساده با استفاده از اتصال یک فلز و آلیاژ آن و با توجه به اثر سیبک تولید میشوند و با همین مکانیزم ساده قادر به اندازه گیری دما به راحتی در رنج وسیعی از زیر صفر تا حدود 1700 درجه میباشند.



با توجه به پدیده سیبک اتصال هر دو فلز از نوع مختلف باعث ایجاد ترموکوپل شده و تولید میلی ولتی متناسب با دمای اعمال شده میکند .از این رو انواع مختلفی از ترموکوپل میتوان تولید کرد در حالی که در استانداردهای متداول میتوان از انواع 10 یا 12 ترموکوپل یاد کرد که در مدلهای های مختلف و با اشکال گوناگون تولید میشود.

سیگنال خروجی ترموکوپل یک ولتاژ ضعیف است که توسط یک مبدل به یک سیگنال استاندارد تبدیل می شود.یکی از ویژگی های ترموکوپلها استفاده در دماهای بالا می باشد زیرا معمولا در دماهای پایین دارای دقت قابل قبولی نمی باشند.

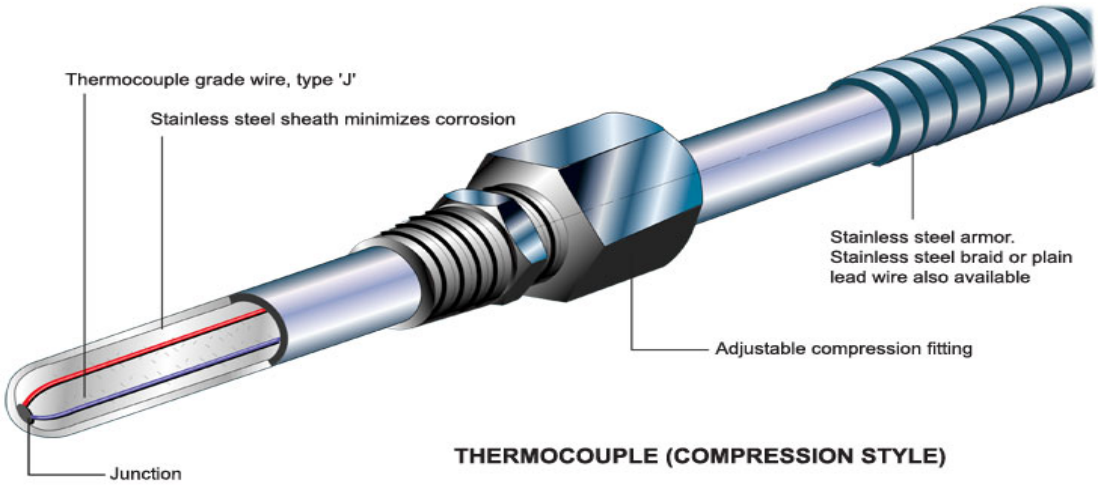
حال به مرور اجمالی برخی از پرکاربردترین انواع ترموکوپل ها می پردازیم.

ترموکوپل تیپ K (CHROMEL-ALUMEL)

ترموکوپل نوع K از سیم فلزی Ni-Cr (به نام تجاری کرومل Chromel) و Ni-Al (به نام تجاری آلومل Alumel) ساخته می شود.این ترموکوپل ارزان قیمت است و یکی از پرکاربردترین ترموکوپل ها می باشد. رنج عملکرد دمایی آن بین C ° 200 - و°C 135+ و حساسیت آن تقریبا µV/°C 41 است و معمولا در دماهای بالا مورد استفاده قرار می گیرد. ترموکوپل نوع K بخاطر استفاده از مس خاصیت ضد اکسیداسیون دارد لذا در کوره ها که اکسیداسیون رخ می دهد مناسب تر می باشد.

ترموکوپل تیپ J

این ترموکوپل از فلز آهن (Fe) و آلیژهای مس- نیکل(Cu-Ni)ساخته می شود. رنج دمایی این ترموکوپل بین 180 ° C - و750 ° C+ است. به دلیل احتمال اکسید شدن آهن این ترموکوپل ، در صنایع قالب ریزی پلاستیک استفاده می شود. حساسیت ترموکوپل نوع J ، به اندازه ی 55 µV/°C است و برای طرح های جدید توصیه می شود.ترموکوپل نوع J به علت وجود آهن در مکانهایی که امکان اکسیداسیون وجود دارد استفاده نمی شود.



THERMOCOUPLE (COMPRESSION STYLE)

ترموکوپل تیپT

ترموکوپل نوع T از مس (Cu) و آلیاژ نیکل – مس((Cu-Ni) (کنستانتان Constantan)) ساخته می شود. محدوده ی عملکرد دمایی این نوع ترموکوپل ، بین 250 ° C - و °C 400+ است. این ترموکوپل نسبتا ارزان و برای کاربردهای با دمای پایین مناسب است و در برابر رطوبت مقاومت خوبی دارد. حساسیت این ترموکوپل، 46 µV/°C است.ترموکوپل نوع T به دلیل اینکه نسبت به تمام انواع ترموکوپل کم خطرتر است و رنج درجه حرارت مناسبی دارد و همچنین از حساسیت خوبی برخوردار است در صنعت بیشتر مورد استفاده میگیرد.

از دیگر انواع ترموکوپل میتوان به ترموکوپل نوع E:کرومل و کنستانتان (chromel and constantan)، ترموکوپل نوع N: نیکروسیل و نیسیل (nicrosil and nisil) و ترموکوپل نوع S، B ، R :پلاتین و رودیوم (platinum and rhodium) اشاره کرد.

ترمومتر (RTD)

ترمومترها، برای اندازه گیری دما به کار میروند.خروجی این سنسورها به صورت اهم می باشد به همین دلیل به سنسورهای مقاومت خطی معروف هستند.

معروفترین سنسور RTD که خیلی کاربرد دارد PT100 است که دارای دقت بالایی می باشد و برای اندازه گیری تا دمای 850 درجه مورد استفاده قرار می گیرد..این سنسور در دمای 0 دارای مقاومت مشخص 100 اهم می باشد و به ازای هر درجه افزایش دما 0.385 اهم به مقاومتش افزوده می شود و از پلاتین که دارای مقاومت دقیقی در دماهای مختلف است ساخته شده است.

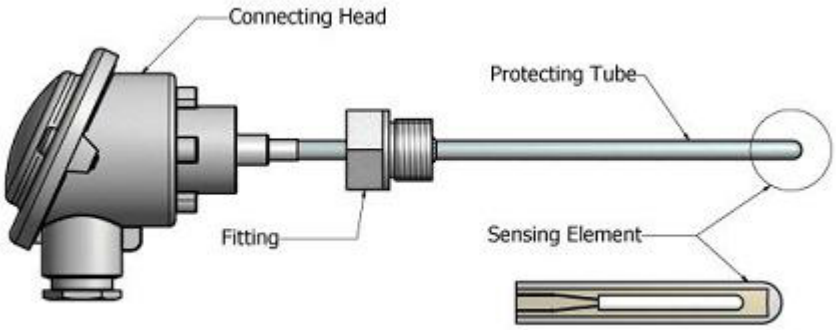
سنسورهای PT100 در واقع با سیم پلاتین پیچانده شده دور سرامیک ساخته شده اند. اما در حال حاضر برای ارزانتر شدن از یک فیلم پلاتین بر روی یک بستر سرامیکی ساخته می شوند.

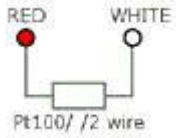
این سنسورها در مقاومتهای 200 ، 500 و 1000 اهم نیز ساخته میشوند. اگر چه این سنسورها کم به فروش میرود ، معمولاً آنها همراه با پروب از جنس استیل برای نصب در محیطهای صنعتی خرید میشوند.

از مزایای استفاده از ترمومتر میتوان به دقت بالا،پایداری طولانی مدت،محدوده دمایی مناسب و توانایی خوب در تبادل اشاره نمود.

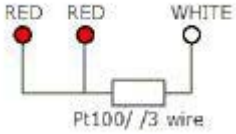
سنسورها ی ترمومتر در 3 دسته،تقسیم بندی می شوند که عبارتند از:

- 2 سیمه

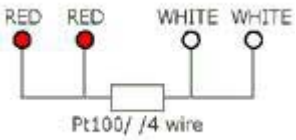




- 3 سیمه



- 4 سیمه

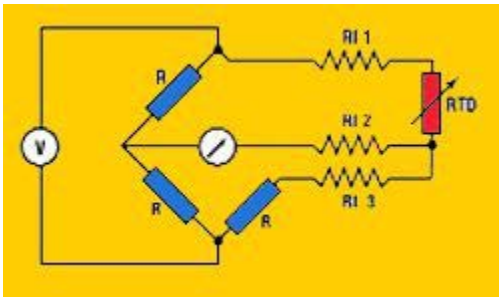


- در بین 3 نوع ترمومتر،مدل 4 سیمه دارای بیشترین دقت و مدل 3 سیمه متداولتر است.

خطا در ترمومترها

o مقاومت کم PT100 بدان معنی است که مقاومت کابل میتواند خطای قابل توجهی را تحمیل کند. مقاومت کابل دو نوع خطا ایجاد میکند ، یکی خطای آفست که به خاطر مقاومت داخلی کابل است (که ناچیز است) و دیگری تغییر در مقاومت کابل در تغییر دما (که نمیتوان نادیده گرفت). هر دو این خطاها را میتوان با استفاده از مدار جبران ساز 3 سیمه و 4 سیمه از بین برد.

بیشتر پروبها به صورت 3 سیمه ساخته میشوند. همانطور که در شکل زیر مشاهده میشود یک سمت PT100 یک سیم و در سمت دیگر آن 2 سیم متصل است ، این سه سیم به دستگاه اندازه گیری ولتاژ که یک پل مقاومتی است متصل شده و همانطور که ملاحظه می شود چون مقاومتهای Rl1 و Rl3 با هم برابرند مقاومت کابل در دستگاه اندازه گیری دیده نمیشود. بنابر این با این روش خطای مقاومت کابل از بین میرود.



توجه داشته باشید که جبران ساز 3 سیمه به طور نظری مدارهای پل ثابت کامل است. پل ولتاژ ثابت تنها زمانی کامل هستند که پل متوازن باشد. به هر حال در اکثر حالات خطا ناچیز است.

همچنین سنسور PT100 دارای یک جدول استاندارد می باشد که خروجی مقاومت به ازای دماهای مختلف در آن درج شده است.

°C	0	1	2	3	4	5	6	7	8	9	°C
-200	18.4932										-200
-190	22.8031	22.3737	21.9439	21.5139	21.0834	20.6526	20.2215	19.7899	19.3580	18.9258	-190
-180	27.0779	26.6520	26.2257	25.7990	25.3720	24.9447	24.5171	24.0891	23.6608	23.2321	-180
-170	31.3200	30.8972	30.4741	30.0507	29.6270	29.2029	28.7786	28.3539	27.9289	27.5036	-170
-160	35.5313	35.1115	34.6914	34.2710	33.8503	33.4294	33.0081	32.5865	32.1646	31.7425	-160
-150	39.7137	39.2967	38.8794	38.4619	38.0440	37.6260	37.2076	36.7889	36.3700	35.9508	-150
-140	43.8691	43.4547	43.0401	42.6252	42.2101	41.7946	41.3790	40.9631	40.5469	40.1304	-140
-130	47.9993	47.5873	47.1752	46.7628	46.3501	45.9372	45.5241	45.1107	44.6971	44.2832	-130
-120	52.1058	51.6962	51.2863	50.8762	50.4659	50.0554	49.6446	49.2336	48.8224	48.4109	-120
-110	56.1903	55.7828	55.3751	54.9672	54.5591	54.1507	53.7422	53.3334	52.9244	52.5152	-110
-100	60.2541	59.8486	59.4429	59.0371	58.6310	58.2247	57.8182	57.4115	57.0047	56.5976	-100
-90	64.2987	63.8950	63.4912	63.0873	62.6831	62.2787	61.8742	61.4695	61.0645	60.6594	-90
-80	68.3251	67.9233	67.5212	67.1190	66.7166	66.3141	65.9114	65.5084	65.1054	64.7021	-80
-70	72.3346	71.9344	71.5340	71.1335	70.7328	70.3319	69.9309	69.5297	69.1284	68.7268	-70
-60	76.3282	75.9296	75.5307	75.1318	74.7326	74.3334	73.9339	73.5343	73.1346	72.7347	-60
-50	80.3068	79.9096	79.5123	79.1148	78.7171	78.3194	77.9214	77.5234	77.1251	76.7268	-50
-40	84.2713	83.8754	83.4795	83.0834	82.6871	82.2908	81.8943	81.4976	81.1008	80.7039	-40
-30	88.2222	87.8277	87.4331	87.0383	86.6434	86.2484	85.8532	85.4579	85.0625	84.6669	-30
-20	92.1603	91.7671	91.3737	90.9802	90.5866	90.1929	89.7990	89.4050	89.0109	88.6166	-20
-10	96.0861	95.6941	95.3019	94.9097	94.5173	94.1247	93.7321	93.3394	92.9465	92.5535	-10
0	100.0000	99.6091	99.2182	98.8271	98.4359	98.0445	97.6531	97.2615	96.8698	96.4780	0
0	100.0000	100.3907	100.7814	101.1719	101.5623	101.9526	102.3427	102.7328	103.1227	103.5125	0
10	103.9022	104.2918	104.6813	105.0706	105.4599	105.8490	106.2380	106.6269	107.0156	107.4043	10
20	107.7928	108.1813	108.5696	108.9578	109.3458	109.7338	110.1216	110.5094	110.8970	111.2845	20
30	111.6718	112.0591	112.4463	112.8333	113.2202	113.6070	113.9937	114.3802	114.7667	115.1530	30
40	115.5392	115.9254	116.3113	116.6972	117.0830	117.4686	117.8541	118.2395	118.6248	119.0100	40
50	119.3951	119.7800	120.1648	120.5495	120.9341	121.3186	121.7030	122.0872	122.4713	122.8554	50
60	123.2392	123.6230	124.0067	124.3902	124.7737	125.1570	125.5402	125.9233	126.3063	126.6891	60
70	127.0718	127.4545	127.8370	128.2194	128.6016	128.9838	129.3658	129.7478	130.1296	130.5113	70

بازه اندازه گیری این سنسور از 200- درجه تا 850 + درجه می باشد.

- یکی دیگر از علل خطا،حرارت داخلی است. به خاطر اینکه یک جریان باید از داخل سنسور عبور کند تا یک سیگنال ولتاژ برای تجهیزات الکترونیک ایجاد شود همین جریان باعث به وجود آمدن حرارت اندکی داخل سنسور و در نتیجه تغییر مقاومت سنسور می شود. یک جریان زیاد میتواند ولتاژ مطلوبی را ارائه دهد اما باعث ایجاد خطای حرارت داخلی بزرگتری میشود. یک جریان کوچک این خطا را کاهش میدهد ، اما کاهش تغییر در خروجی الکترونیک برای به حداقل رساندن خطا در مدار مورد نیاز است. بهترین راه سعی و خطا روی کاربرد است با این حال به طور کلی جریانهای 1mAیا کمتر استفاده میشود. خطای حرارت داخلی وقتی دمای گاز اندازه گیری شود بیشتر است زیرا در این حالت سنسور، حرارت داخلی خود را کمتر میتواند از دست دهد.

سنسور وزن Load Cell

لودسل یک نوع حسگر الکترونیکی برای اندازه گیری وزن و نیرو است که در سیستم های توزین مورد استفاده قرار می گیرد . این محصول تغییرات وزن را بر اساس تغییر ولتاژ و وزن بار وارده حس کرده و آن را به نشان دهنده الکترونیکی یا اندیکاتور منتقل می نماید . هر کدام از انواع لودسل به کلاس های مختلفی تقسیم می شود. هر لودسل دارای مشخصاتی نظیر کلاس ارائه شده می باشد، کلاس هر لودسل بیانگر موارد مختلفی از جمله دقت، ظرفیت و تعداد قسمت های تقسیم شده بر حسب استاندارد است.

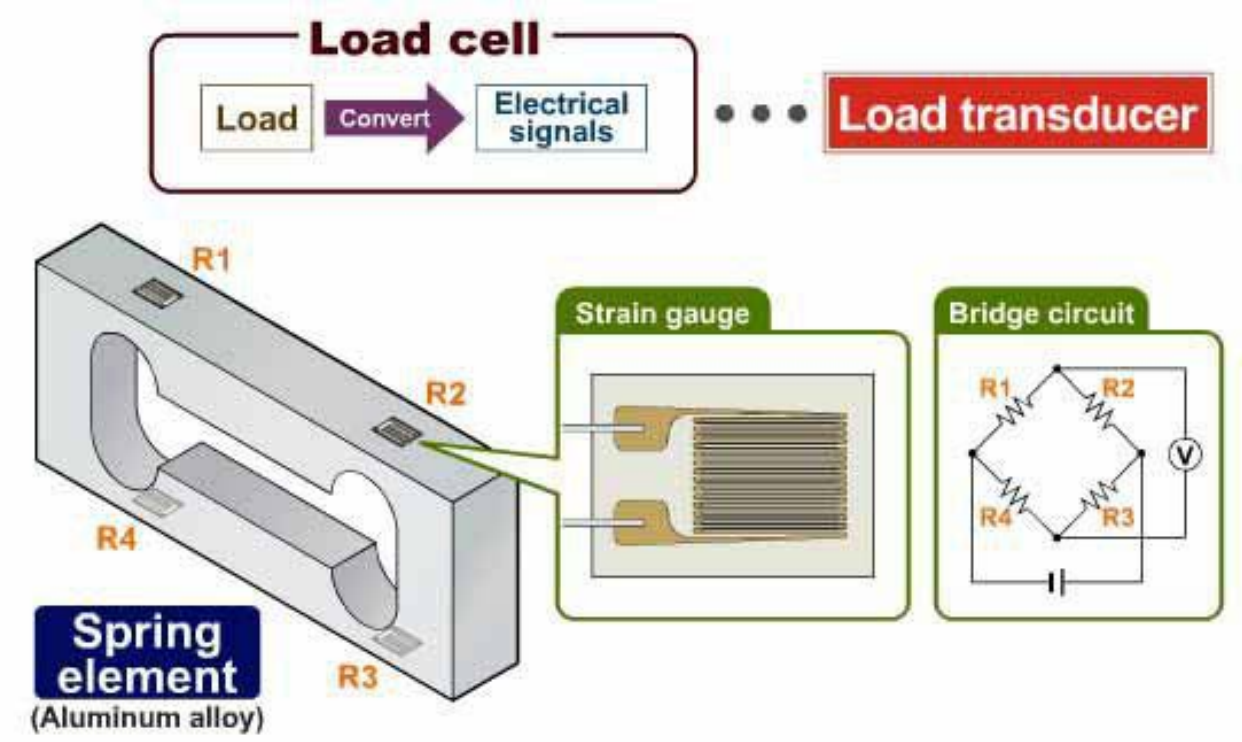
دقت لودسل به عنوان یکی از پارامترهای مهم در انتخاب لودسل می باشد و به صورت درصدی از ظرفیت کل در نظر گرفته می شود.

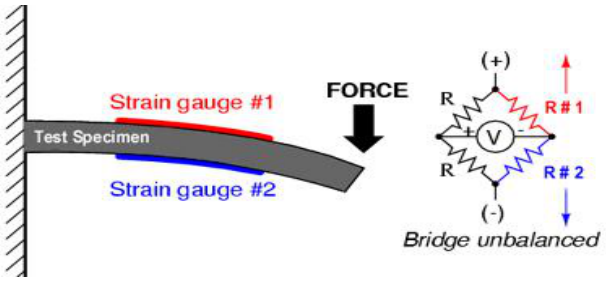
لودسل های مرغوب و دقیق دارای استاندارد OIML جهانی می باشند که این استاندارد مبین کیفیت و دقت ساخت این محصول است . بنابراین عملیات سنجش وزن بوسیله لودسل هایی که فاقد این استاندارد باشند قابل اطمینان نخواهد بود .

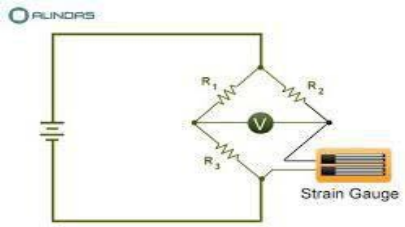
نحوه عملکرد لودسل

مکانیسم عملکرد لودسل بر اساس تغییرات طول ناشی از وارد شدن بار می باشد که سبب تغییر خروجی آن می شود. این وسیله فشار وارده از سوی جسم و موارد مورد نظر را به سیگنال الکتریکی تبدیل می کند و این امکان را به وجود می آورد که لودسل به وسیله آن وزن دقیق مواد را بدست آورد .

لودسل شامل یک هسته فلزی (از آلیاژ خاص) و تعدادی Strain Gauge (مجموعه ای از مقاومت های الکتریکی) می‌باشد که در اثر اعمال نیرو تمام مواد تغییر شکل می‌یابد اما پس از برداشتن نیرو به حالت اولیه خود برمی‌گردد . میزان برگشت‌پذیری این ماده تعیین کننده کیفیت و دقت و دیرپایی لودسل است.





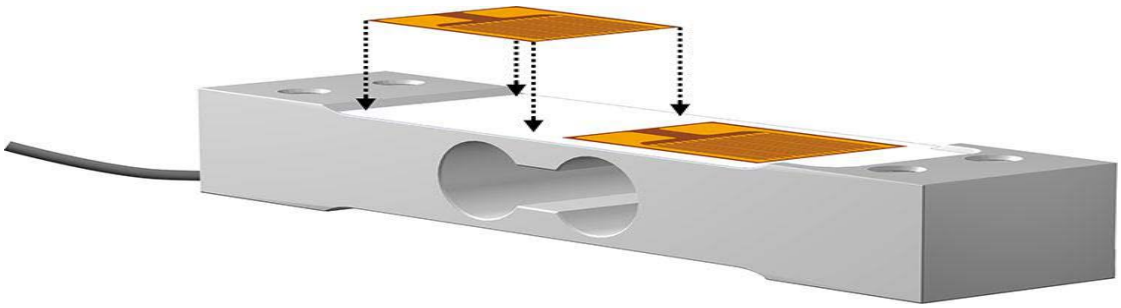


اصلی ترین قسمت لودسل استرین گیج (Strain Gauge) می باشد که توسط چسب مخصوصی بر روی هسته داخلی لودسل نصب می گردد.

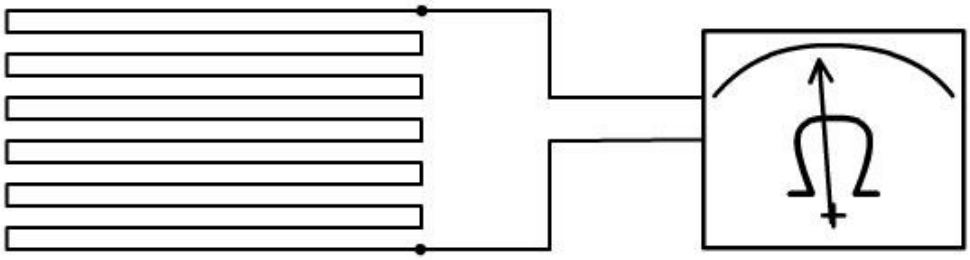
استرین گیج سیم رسانایی است که بصورت زیگزاگ روی هسته نصب شده و در اثر کشیده شدن، طول آن افزایش یافته و مقاومتش بالا می رود.

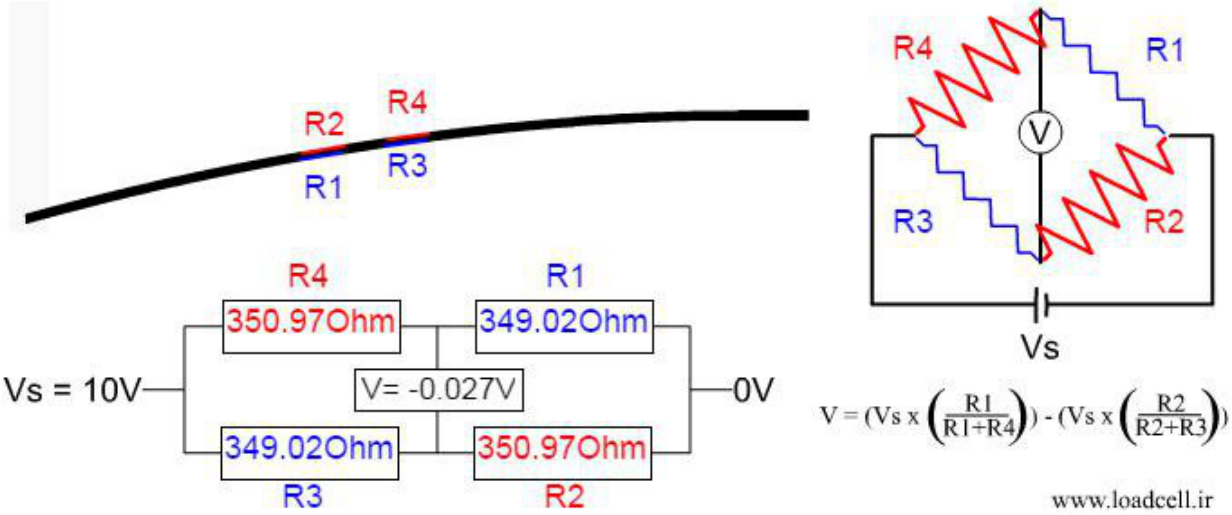
زمانی که نیرو بر لودسل وارد می شود، لودسل از شکل طبیعی خود خارج شده (این تغییر شکل قابل رویت نیست) و استرین گیج را در حالت کشش قرار می دهد و باعث تغییر مقاومت آن می شود که اندازه این تغییر مقاومت به مقدار نیروی وارده بر لودسل بستگی دارد.

یک لودسل معمولا دارای چندین استرین گیج مشابه می باشد.



استرین گیج ها بصورت یکسان در جهت کشش نصب می شوند و اغلب چهارتایی یک پل وتستون (Wheatstone Bridge) کامل را تشکیل می دهند.





لودسل ها معمولا از آلیاژ فلزاتی مانند آلومینیوم، فولاد و استیل ساخته می شوند که نوع این آلیاژ در طول عمر لودسل موثر است.

باید به این نکته توجه کرد که ظرفیت تعیین شده برای هر لودسل بسیار مهم است و شوک ناگهانی یا نیروی وارده ای که بیش از ظرفیت تعیین شده ی لودسل باشد به لودسل صدمه وارد می کند.

همانطور که گفته شد لودسل مبدلی است که نیرو را به صورت بار دریافت کرده و آن را به سیگنال های الکتریکی تبدیل می کند. دستگاه هایی مانند نشاندهنده های دیجیتالی وزن، کامپیوتر های صنعتی و سایر

وسایل اندازه گیری برای نمایش وزن، از سیگنال های الکتریکی لودسل استفاده می کنند. وزن نمایش داده شده توسط همین دستگاه ها به شکل های گوناگون مانند چاپ، ذخیره اطلاعات و ... پردازش می شود.

کابل لودسل از دو سیم مثبت و منفی تغذیه ورودی(Input Excitation) ، دو سیم مثبت و منفی سیگنال خروجی (Output Signal)و یک سیم اتصال به زمین (Shield) تشکیل شده است که در کل شامل ۵ رشته سیم مجزا می باشد. گاهی به منظور کاهش خطا در کابل لودسل ها از ۷ رشته سیم استفاده می شود که شامل ۵ سیم نامبرده و دو سیم دیگر برای تشخیص درستی ارتباط (Sense) است.

انتخاب لودسل

انتخاب یک لودسل مناسب به نوع، ظرفیت ، دقت و جنس آن بستگی دارد که به شرح هر کدام از موارد فوق می پردازیم:

الف) انواع لودسل

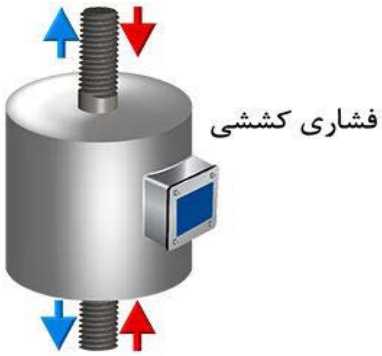
انتخاب نوع لودسل براساس نحوه ی بارگذاری می باشد یعنی مطابق با نیروهای وارده بر لودسل مانند کشش، فشار، خمش و...

لودسل ها با توجه به کاربردی که دارند به انواع مختلفی تقسیم می شوند:

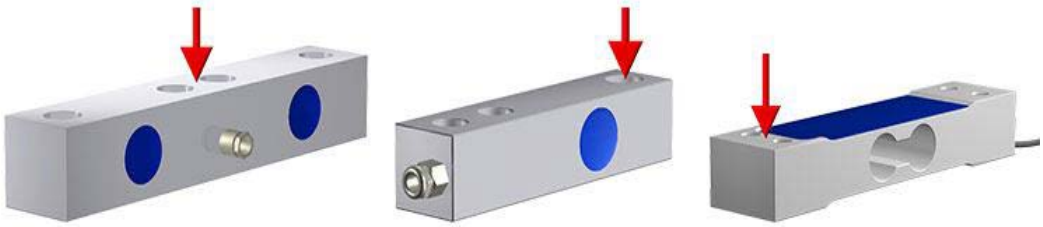
- لودسل فشاریCompression
- لودسل خمشیShear Beam
- لودسل تک پایهSingle point
- لودسل کششی S-Type

در دسته بندی دیگری لودسل ها به گروه های زیر تقسیم می شوند:

- لودسل ستونی (Column Load Cells)



- لودسل تیغه ای یا میله ای (Beam Load Cells)



تک پایه | خمشی یا برشی یکطرفه | خمشی یا برشی دوطرفه

- لودسل S شکل (S Type Load Cells)

فشاری کششی

در این نوع لودسل فشار وارده بر لودسل منجر به تغییر کوچکی در طول لودسل میگردد. در این نوع سنسور وزن و فشار وارده بایستی به صورت عمود بر مرکز لودسل وارد شود.این لودسل ها در ظرفیت 20kg تا 10000kg موجود می باشند.

ب)ظرفیت لودسل(Capacity)

تشخیص ظرفیت لودسل براساس فرمول زیر محاسبه می گردد:

$$C_{Lc} \geq \frac{Q \times (Tare + Clive)}{N} \text{ X K}$$

C_{Lc} : حداقل ظرفیت لودسل بر حسب کیلو گرم

Tare : وزن مرده یا پلتفرم بر حسب کیلو گرم

C_{Live} : حداکثر ظرفیت توزین لودسل برحسب کیلوگرم

Q : ضریب اطمینان لودسل

N : تعداد لودسل

K : نسبت یا ضریب کاهش لودسل

1)ظریب اطمینان لودسل(Q)

یک فرمول خاص و دقیق جهت محاسبه مقدار ضریب اطمینان به صورت عام وجود ندارد و با توجه به شرایط محیطی و نوع کارکرد سیستم توزین تعیین می شود . از عوامل موثر بر این مقدار می توان به موارد زیر اشاره نمود :

- شرایط اعمال بار خارج از مرکز (غیر متقارن)
- اثر بار ضربه ای و دینامیکی
- اثر وزش باد
- طراحی و ویژگی های صفحه توزین (پلتفرم)

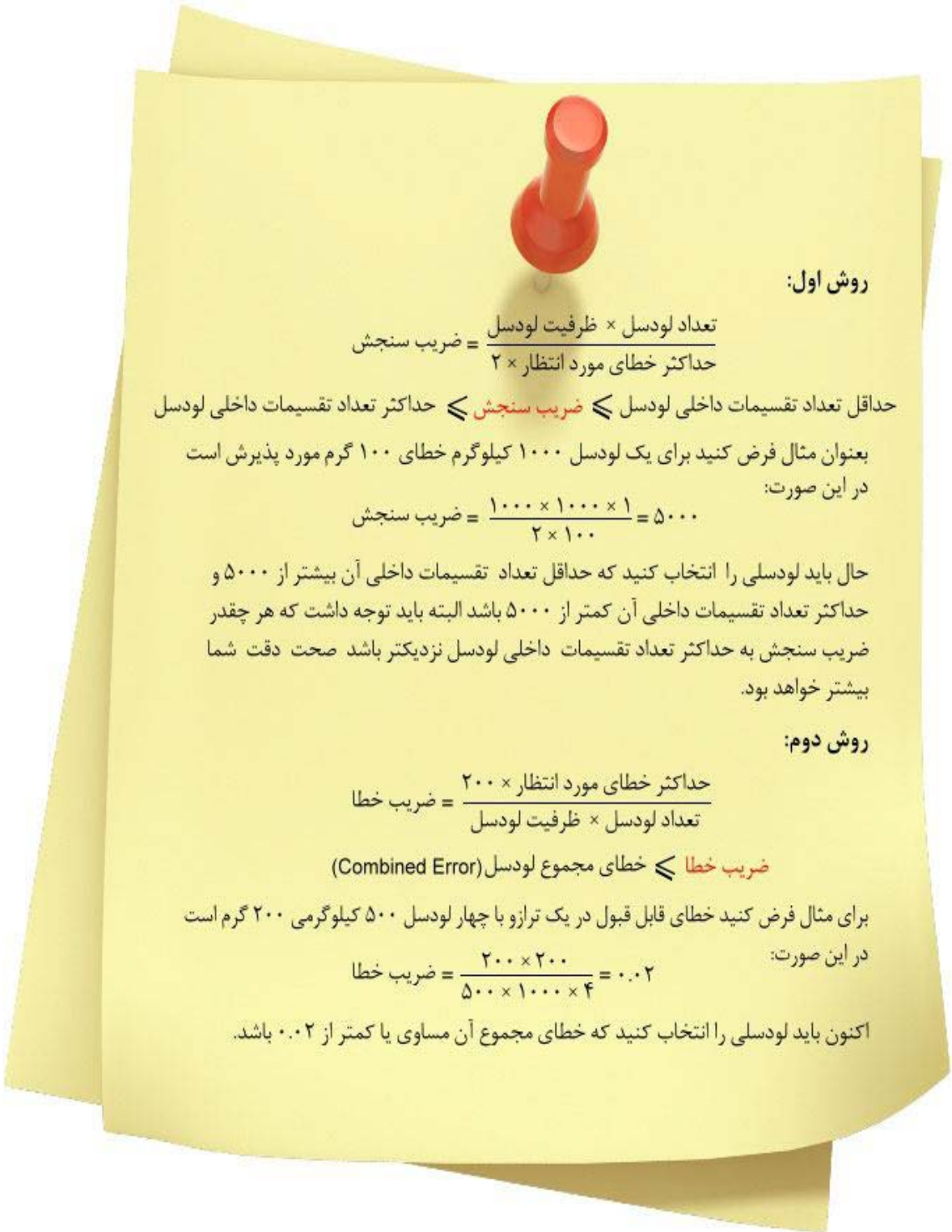
2) ضریب کاهش (K)

این مقدار مطابق فرمول ذیل محاسبه می گردد :

$$K = \frac{\text{مقدار بار اعمالی بر لودسل}}{\text{مقدار بار اعمالی بر صفحه توزین}}$$

این مقدار عموماً در لودسل های مکانیکی که از اهرم بندی استفاده می شود مخالف 1 است و در سایر سیستم های تمام الکترونیک معادل یک می باشد .

ج)دقت لودسل(Accuracy) عوامل نسبتا زیادی در تعیین دقت لودسل سهیم می باشند اما ساده ترین راه برای انتخاب دقت یک لودسل، یکی از دو روش زیر می باشد.



روش اول:

$$\text{ضریب سنجش} = \frac{\text{تعداد لودسل} \times \text{ظرفیت لودسل}}{\text{حداکثر خطای مورد انتظار} \times ۲}$$

حداقل تعداد تقسیمات داخلی لودسل ⩽ ضریب سنجش ⩽ حداکثر تعداد تقسیمات داخلی لودسل

بعنوان مثال فرض کنید برای یک لودسل ۱۰۰۰ کیلوگرم خطای ۱۰۰ گرم مورد پذیرش است در این صورت:

$$\text{ضریب سنجش} = \frac{۱ \times ۱۰۰۰ \times ۱۰۰۰}{۲ \times ۱۰۰} = ۵۰۰۰$$

حال باید لودسلی را انتخاب کنید که حداقل تعداد تقسیمات داخلی آن بیشتر از ۵۰۰۰ و حداکثر تعداد تقسیمات داخلی آن کمتر از ۵۰۰۰ باشد البته باید توجه داشت که هر چقدر ضریب سنجش به حداکثر تعداد تقسیمات داخلی لودسل نزدیکتر باشد صحت دقت شما بیشتر خواهد بود.

روش دوم:

$$\text{ضریب خطا} = \frac{\text{حداکثر خطای مورد انتظار} \times ۲۰۰}{\text{تعداد لودسل} \times \text{ظرفیت لودسل}}$$

ضریب خطا ⩽ خطای مجموع لودسل(Combined Error)

برای مثال فرض کنید خطای قابل قبول در یک ترازو با چهار لودسل ۵۰۰ کیلوگرمی ۲۰۰ گرم است در این صورت:

$$\text{ضریب خطا} = \frac{۲۰۰ \times ۲۰۰}{۵۰۰ \times ۱۰۰۰ \times ۴} = ۰.۰۲$$

اکنون باید لودسلی را انتخاب کنید که خطای مجموع آن مساوی یا کمتر از ۰.۰۲ باشد.

د)جنس لودسل

با توجه به محیط استفاده ی لودسل، جنس لودسل انتخاب می شود. مثلا در محیط های مرطوب و اسیدی که درجه خوردگی بالا می باشد بهتر است جنس لودسل از آلیاژ استیل، در محیط های مغروق در آب جنس لودسل از استنلس استیل با درجه حفاظت IP68 و در محیط های معمولی جنس لودسل از آلیاژ آلومینیوم انتخاب شود.

قیمت لودسل ها بسیار متفاوت است و این تفاوت بستگی به عواملی مانند ظرفیت، جنس، دقت، نوع، استانداردها، سازنده یا برند و ... دارد.

سنسور فشار

فشار یکی از مهمترین پارامترهای استفاده شده در صنعت برای کنترل پروسه های مختلف میباشد و از تقسیم نیرو بر سطح به دست می آید. این کمیت با توجه به این که در مواردی نظیر ارتفاع سنجی سطح مایعات و همچنین اندازه گیری سرعت سیالات و فلو نیز کاربرد دارد اهمیت ویژه داشته و تجهیزات و وسائل مختلفی جهت اندازه گیری آن در شرایط مختلف ابداع گردیده است.
این سنسور نیز همانند سایر سنسورها دارای انواع مختلفی می باشد.

اندازه گیری فشار با دیافراگم

اندازه گیری فشار با دیافراگم یکی از روش های اندازه گیری فشار در ابزاردقیق می باشد،این سنسورها که معمولاً فلزی یا از انواع خاصی از پلیمرهای ارتجاع پذیر می باشند، بصورت یک صفحه صاف یا چین خورده ساخته شده اند. عمدتاً این نوع از دستگاه ها برای اندازه گیری اختلاف فشار استفاده می شوند. در یک سلول اختلاف فشار یا Dp cell تغییر شکل مکانیکی دیافراگم موجب یک جابجایی کوچک و نیز کشیدگی سطح دیافراگم میشود.

از این دو خاصیت یعنی جابجایی و نیز کشیدگی سطح ایجاد شده، به طور مستقل میتوان به منظور اندازه گیری میزان اختلاف فشار اعمالی استفاده نمود. در واقع میزان کشیدگی و میزان جابجایی ایجاد شده، رابطه مشخصی با اختلاف فشار اعمالی دارد. لذا با دانستن یکی از دو فشار، می توان میزان فشار مجهول را به دست آورد.

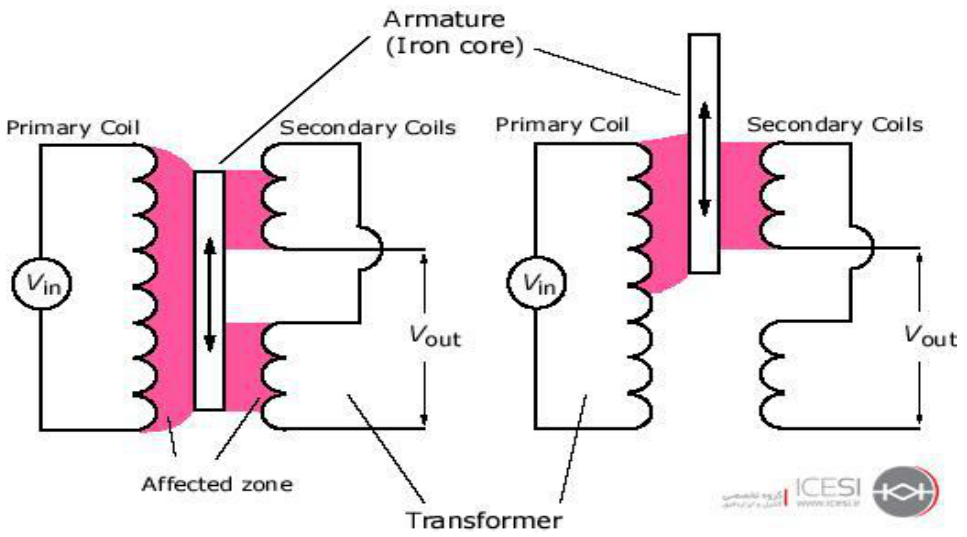
مزایای اندازه گیری فشار با دیافراگم

از محاسن مهم دیافراگم ها، قابلیت محافظت از دیگر ابزارهای اندازه گیری فشار در برابر شرایط نامناسب، مثل دماهای بالا و سیالات خورنده و نیز توانایی بالای آنها در آب بندی می باشد.

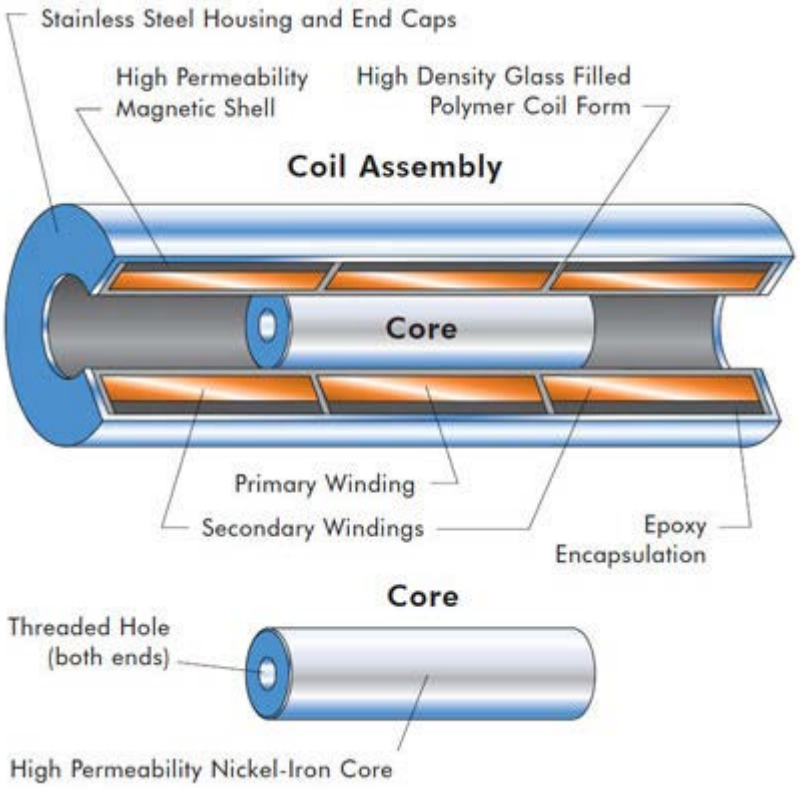
فشارسنج های مغناطیسی (LVDT)

LVDT یک تجهیز الکترومکانیکی است که توسط جابجایی دو هسته می تواند تولید سیگنال کند و در دو نوع کلی AC و DC می باشند.

فشارسنج های مغناطیسی در واقع یکی از روشهای اندازه گیری فشار می باشد که به معرفی آن می پردازیم ،دو سیم پیچ را در نظر بگیرید که یکی از آنها به یک منبع پتانسیل متصل میباشد. چنانچه جریانی که از این سیم پیچ می گذرد، تغییر کند و یا اینکه خود سیم پیچ جابجا شود، به علت تغییر میدان الکترومغناطیسی، شار عبوری از سیم پیچ دوم نیز تغییر می کند. تغییر شار الکترومغناطیسی سبب بوجود آمدن جریان القایی در سیم پیچ دوم می شود. این تغییر میدان را می توان با تغییر مکان یک هسته آهنی که بین این دو سیم پیچ قرار دارد، نیز بوجود آورد. یعنی با جابجایی هسته، میدان الکترومغناطیسی تغییر کرده و در نتیجه شار عبوری از سیم پیچ دوم نیز تغییر می یابد.



حال اگر این هسته را به یک سنسور فشار مانند دیافراگم وصل نماییم، جابجایی تولیده شده توسط سنسور، سبب حرکت هسته در داخل سیم پیچ ها می شود. با اندازه گیری میزان جریان القایی در سیم پیچ ثانویه، میزان جابجایی هسته و در نتیجه تغییرات فشار محاسبه می شود. لازم به ذکر است میزان تغییر در جریان القایی با میزان جابجایی هسته به طور خطی متناسب نیست. برای رفع این مشکل از دو سیم پیچ ثانویه استفاده میشود.



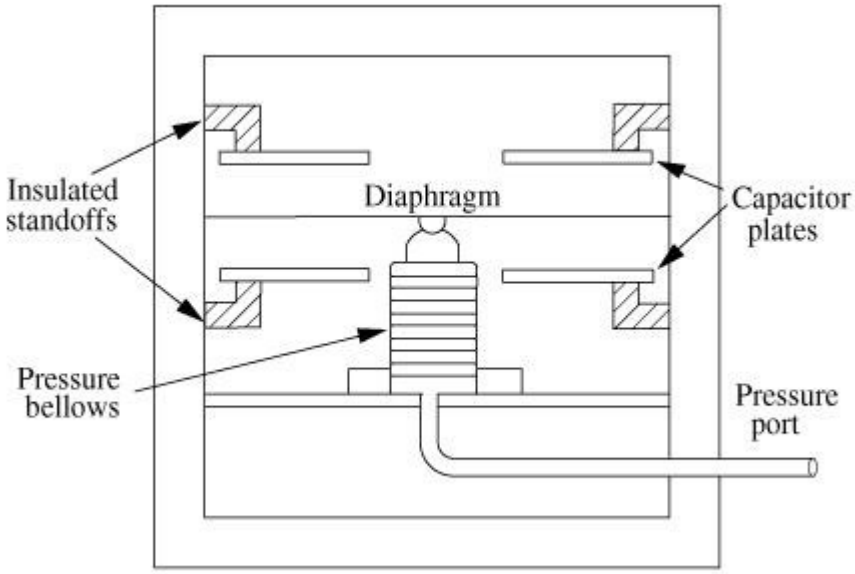
اندازه گیری فشار به روش خازنی(Capacitive)

اندازه گیری فشار به روش خازنی در واقع یکی از روشهای اندازه گیری فشار است که سنسورهای آن در تجهیزات ابزاردقیق و ترانسمیترها استفاده می شود، ظرفیت الکتریکی خازن تابعی از فاصله صفحات، مساحت آنها و ضریب دی الکتریک ماده دی الکتریک میباشد و با فاصله صفحات رابطه عکس دارد. به عبارت دیگر با کاهش فاصله بین صفحات ظرفیت الکتریکی آن افزایش می یابد.

مقاومت ظاهری خازن ها که بیشتر در مدارهای جریان متناوب بکار می روند، متناسب با ظرفیت الکتریکی آنها می باشد. لذا می توان با اندازه گیری مقاومت ظاهری خازن در یک مدار پل وتستون که با ولتاژ AC تحریک می شود، ظرفیت الکتریکی خازن را محاسبه نمود.

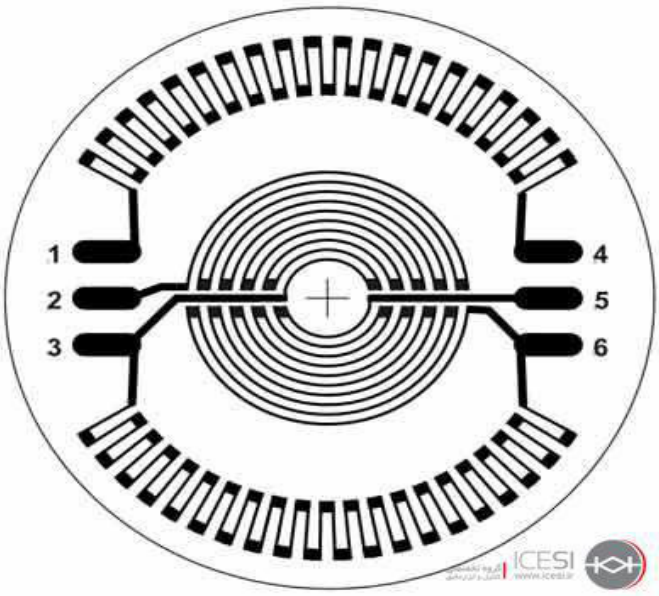
دستگاه های اندازه گیری خازنی از یک سنسور فشار و یک ترنسدیوسر خازنی تشکیل یافته است. سنسور فشار که می تواند یک دیافراگم باشد، پس از دریافت تغییرات فشار سیال، آنرا به صورت جابجایی به یکی از صفحات

خازن ترنسدیوسر منتقل می نماید و سبب تغییر ظرفیت آن می شود. در نتیجه با اندازه گیری تغییرات ظرفیت الکتریکی خازن، تغییرات فشار محاسبه می گردد.



بازه اندازه گیری فشار به روش خازنی

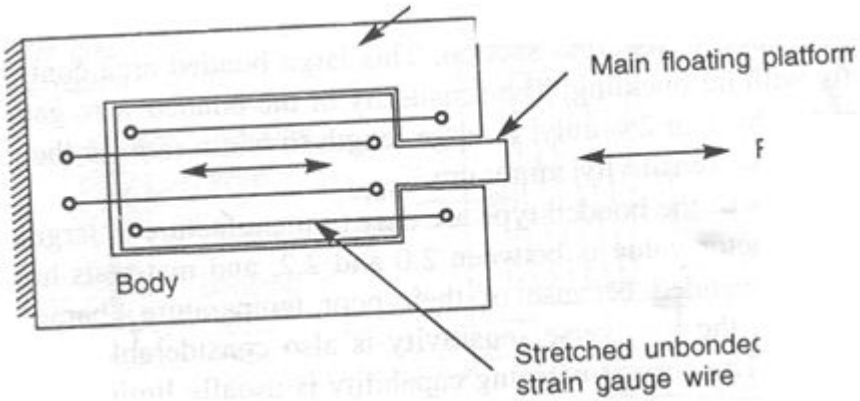
مبدلهای فشار خازنی میتوانند در خلاءهای بسیار زیاد تا فشارهای بالا در حد 10000PSI استفاده شوند و به همین دلیل کاربردگسترده ای دارند و در مقایسه با کرنش سنج ها، خطای Drift زیادی ندارند.معمولا بیشترین دقت این ترنسدیوسرها ۰،۱ درصد و کمترین دقت آنها ۰،۲ درصد می باشد. علی رغم اینکه این روش حساس و دقیق است ولی بسیار وابسته به تغییر ات دمایی می باشد. به همین دلیل سیستمی برای تصحیح خطای تغییرات دمایی در کنار این ترنسدیوسرها وجود دارد.



اندازه گیری فشار با استرین گیج(Strain Gauge)

اندازه گیری فشار با استرین گیج از مرسوم ترین و پرکاربرد ترین روش های اندازه گیری فشار در ترانسمیترهای فشار در صنایع بزرگ می باشد. مقاومت الکتریکی یک سیم با طول سیم، رابطه مستقیم و با سطح مقطع آن رابطه معکوس دارد. به عبارت دیگر هرچه طول سیم بیشتر شود و یا سطح مقطع آن کمتر گردد، مقاومت الکتریکی سیم بیشتر خواهد شد.

کرنش سنج ها یا استرین گیج ها یکی از متداولترین ترانسدیوسرهای فشار هستند. این دستگاه اندازه گیر شامل یک سنسور دیافراگمی است که روی آن یک یا چند مقاومت الکتریکی چسبانده شده است.معمولا این مقاومت ها روی صفحه جداگانه ای نصب شده اند و مجموعهٔ آنها روی دیافراگم قرار گرفته اند. کشیدگی سنسور در اثر اعمال فشار، منجر به کشیده شدن صفحه در حد چند میکرون، می شود و چون مقاومت الکتریکی کاملاً به صفحه چسبانده شده، تغییر شکل صفحه باعث تغییرطول مقاومت و در نتیجه تغییر مقاومت الکتریکی آن می شود. این تغییر مقاومت الکتریکی که متناسب با میزان فشار اعمالی است، توسط یک پل وتستون اندازه گرفته می شود و در نهایت میزان تغییرات فشار محاسبه گردیده و به نمایشگر یا سیستم کنترلی ارسال می گردد.



اندازه گیری فشار با استرین گیج دارای مزایای زیر است:

- طول عمر بالا
- عملکرد بسیار آسان
- قیمت مناسب

سنسور فشار نوع پیزوالکتریک

اصطلاح پیزو بیانگر تغییر در مقاومت الکتریکی ماده ای است که در معرض یک نیروی مکانیکی همچون فشار قرار می گیرد.این پدیده در کریستالهای خاص رخ داده و به خوبی در نیمه هادی ها نمایان می شود.در این سنسور کریستالهای خاص را تحت فشار قرار میدهند که این فشار منجر به تولید سیگنال توسط مدارهای مختلف می گردد.

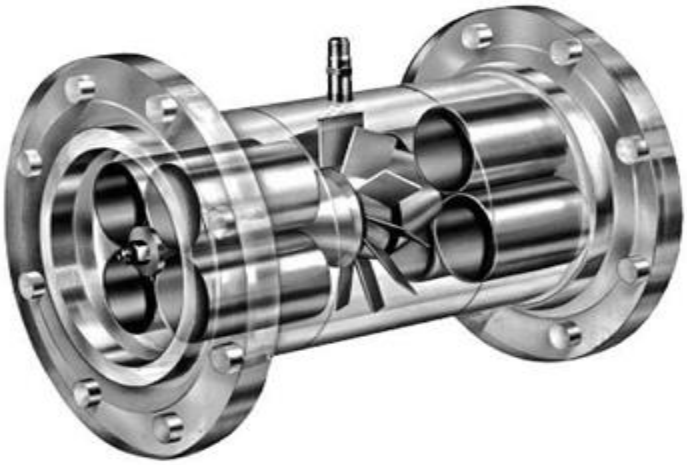
سنسورهای فلومتر

فلومتر یا دبی سنج (Flow Meter) وسیله ایست که حجم مواد عبوری را نسبت به زمان اندازه گیری می کند. در بیشتر صنایع از جمله صنایع نفت و پتروشیمی و...کاربرد دارد. دانستن مقدار دقیق فلو یا جریان عبوری سیالی مانند گاز و یا نفت و همچنین توجه به میزان کل فلو که معرف جرم ماده مصرفی است در بسیاری موارد نظیر پروسه های صنعتی کاربردی و مهم است. همچنین اندازه گیری دقیق فلو به دلیل ملاحظات مالی در فروش وخرید حجم مشخصی از ماده و کالا مانند کنتور گاز خانگی و تبادلات نفتی و ...اهمیت ویژه ای دارد.

انواع روش های اندازه گیری فلو

فلو متر توربینی(Turbine Flowmeter)

فلومترهای توربینی مکانیزم ساده ای دارند و از یک توربین با پره های مشخص تشکیل شده اند که در کنار یک سنسور(برای مثال پراکسیمتی سوئیچ) با هر گردش توربین در اثر عبور جریان و فلو تعدادی پالس تولید میکنند که با کالیبرهٔ دستگاه بر اساس مقدار فلو و تعداد پالس ایجاد شده رنج فلومتر بدست می آید.



فلومترهای جرمی یا مس فلومتر(Mass Flowmeter)

یکی دیگر از روشهای اندازه گیری فلو بر حسب وزن میباشد که به فلومترهای جرمی یا مس فلومتر مشهورند. در روشهای متداول قبل برای اندازه گیری جرم سیال عبوری نیاز به اندازه گیری دانسیته مواد داشت که عملا محاسبه جرم با خطای بسیار همراه می گردید. در حال حاضر انواع فلومترهای جرمی با تکنولوژیهای کوریالیس و ترمال مس این اندازه گیری را با دقت بالا انجام می دهند.در این فلومترها، فلوی سیال مستقل از فشار قابل اندازه گیری است و شما را از خطاهای اندازه گیری فلوی سیالات گازی ناشی از تغییرات دما و فشار مصون نگه میدارد.

فلومتر گردابی یا ورتکس (Vortex Flowmeter)

در عمل فلومتر ورتکس با مکانیزمی بر اساس اندازه گیری نوسانات ایجاد شده در پشت یک مانع در سیال کار میکند. اساس کار این سنسورها بدین ترتیب است که با قرار دادن یک مانع بر سر راه سیال، جریان گردابی در پشت مانع تولید می شود که باعث به وجود آمدن افت فشار می گردد. این افت فشار متناسب با سرعت سیال است. حال توسط یک سنسور پیزوالکتریک (سنسور فشار) این نوسانات مکانیکی را به سیگنال الکتریکی تبدیل می کنیم.

فلومتر ورتکس در بسیاری از صنایع برای اندازه گیری مایعات و گازها و بخار مورد استفاده قرار میگیرد. برای مثال در صنایع شیمیائی و پتروشیمی در قسمت تولید برق ... در سیالاتی نظیر بخار اشباع، بخار بسیار داغ،

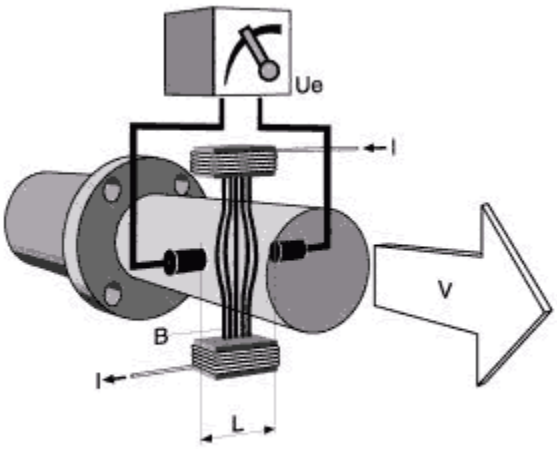
هوای فشرده، نیتروژن، گازهای دو فازی، گاز دودکش، دی اکسید کربن، آب دمینرال، حلالها، روغن انتقال حرارت، آب تغذیه بویلر و .... میتوان از این فلومتر استفاده کرد.

اندازه گیری فلو در فلومتر ورتکس به صورت حجمی بوده که البته برای اندازه گیری به صورت جرمی در مواردی مانند بخار بایستی با استفاده ترانسمیتر فشار جداگانه و همچنین سنسور دما و اتصال آنها به فلو کامپیوتر به محاسبه جرم عبوری یا مقدار انرژی جابجا شده پرداخت. در تصاویر زیر شکل کلی این فلومترهای آمده است.

فلومترهای الکترومغناطیسی(Electromagnetic Flowmeter)

فلومترهای الکترومغناطیسی با استفاده از قانون القای الکترومغناطیسی فارادی و اندازه گیری تغییرات شار مغناطیسی در اثر سرعت سیال جاری میتوانند مقدار فلو را با دقت قابل قبولی اندازه گیری کنند. در حقیقت با ایجاد شار مغناطیسی در سیال عبوری از فلومتر و اندازه گیری آن میتوان به یک فلومتر الکترومغناطیسی دست پیدا کرد.

به دلیل مکانیزم فلومتر مغناطیسی، آب یا سیال عبوری باید تا اندازه ای خواص الکتریکی و رسانائی داشته باشد و در غیر این صورت اندازه گیری با اخلال مواجه میشود و بایستی از فلومترهائی با مکانیزم های متفاوت نظیر فلومتر ورتکس یا فلومتر توربینی استفاده نمود برای مثال از فلومتر مغناطیسی برای اندازه گیری فلوی آب دمینرال یا مقطر نمیتوان استفاده نمود.



$Ue = B \cdot L \cdot v$
$Q = A \cdot v$
Ue = induced voltage
B = magnetic induction (magnetic field)
L = electrode gap
v = flow velocity
Q = volume flow
A = pipe cross-section

I = current strength

فلومترهای آلتراسونیک(Ultrasonic Flowmeter)

امروزه روشهای آلتراسونیک جزء لاینفک بیشتر تجهیزات اندازه گیری ابزار دقیق مخصوصا نمونه های غیر تماسی گردیده است. در فلومترهای آلتراسونیک که در حوزه وسیعی از سیالات اعم از گازها و مایعات کاربرد دارند با توجه به خواص دوپلر که در فیزیک با آن آشنا هستید فرکانسی از یه منبع تولید شده و از طرف دیگر یک گیرنده این فرکانس را لحظه به لحظه چک میکند. در صورتی که فلوی سیال دچار تغییر میشود این فرکانس رفت و برگشت تغییر به خاطر سرعت سیال تغییر محسوسی خواهد داشت. اندازه گیری و کالیبره این تغییرات بر حسب فلو به شما امکان اندازه گیری فلو به روش آلتراسونیک را خواهد داد. در عمل تکنیک اندازه گیری فلو به روش آلتراسونیک پیچیدگی های خود را خواهد داشت ولی اساس اندازه گیری اکثر فلومترهای اولتراسونیک مکانیزم دوپلر میباشد.

فلومتر جرمی دمایی

از فلومترهای دمایی برای اندازه‌گیری فلوی جرمی گازها به‌طور مستقیم با فشار کم و سرعت پایین استفاده می‌شود.

اندازه‌گیری فلو در روش دمایی بر این اساس است که در هنگام جریان یافتن گاز، حرارت از جسم گرم منتقل می‌شود. یک فلومتر دمایی برای اندازه‌گیری فلو دارای دو سنسور اندازه‌گیری دمای PT100 می‌باشد. یکی از سنسورها به‌عنوان اندازه‌گیر دمای مرجع می‌باشد، سنسور دیگر گرم شده و هنگامیکه فلو جریان ندارد یک اختلاف دمای ثابت و مشخص با سنسور اول به وجود می آید به‌محض اینکه سیال جریان می‌یابد سنسوری که گرم شده بود شروع به خنک شدن می‌کند هرچه سیال سریع‌تر حرکت می‌کند سریع‌تر خنک می‌شود . انرژی الکتریکی موردنیاز برای گرم کردن سنسور دوم به منظور ثابت نگه‌داشتن اختلاف دما متناسب با فلوی جرمی می‌باشد.

Meysam.zare.4@gmail.com
Mehrnoush_asghari@yahoo.com

- **مزایای فلومتر دمایی**

1- اندازه‌گیری مستقیم فلوی جرمی و دمای سیال

2- بدون نیاز به جبران دما و فشار

3- قابلیت اندازه‌گیری رنج وسیع

4- واکنش سریع به جریان کم سیال

5-افت فشار ناچیز

6- بدون نیاز به تعمیر و نگهداری ، بدون قطعات متحرک

فلومترهای دمایی به طور گسترده در صنایع برای اندازه‌گیری فلوی حجمی به‌کاربرده شده است ازجمله:

- هوای فشرده
- دی‌اکسید کربن (برای تخمیر و خنک‌کننده)
- آرگون (در تولید فولاد)
- نیتروژن و اکسیژن (تولید)
- گاز طبیعی (برای کنترل سوخت کوره و بویلر)
- تهویه و گازهای گلخانه ای

هنگامیکه درگازها اندازه‌گیری جریان در رنج وسیع و افت فشار کم‌اهمیت باشد فلومترهای دمایی بهترین پیشنهاد به‌جای روش‌های سنتی برای کنترل فرآیند ، مانیتورینگ و فروش، تشخیص نشتی و مانیتور شبکه‌های توزیع می‌باشد. با استفاده فلومتر گرمایی داخل شونده می‌توان فلوی لوله‌های بزرگ و داکت های مستطیل شکل هوا را اندازه‌گیری کرد.

سطح سنج(Level Transmiter)

سطح سنج ها و یا لول سنج ها ابزارهایی هستند که به کمک آنها می توان سطح سیالات را با دقت بالا اندازه گیری کرد. سطح سنج ها با توجه به نوع تکنولوژی و کاربری آنها به انواع مختلفی تقسیم بندی می شوند . در حالت کلی با توجه به نوع اندازه گیری به دو دسته نقطه ای و پیوسته تقسیم بندی می شوند، در اندازه گیری نقطه ای سطح، سطح سیال در یک نقطه خاص اندازه گیری می شود و در اندازه گیری پیوسته سطح، ارتفاع سیال درون مخزن بطور پیوسته اندازه گیری می شود.

انواع سطح سنج های پیوسته

1-سطح سنج شناوری(Float Level Switch)

از خاصیت شناوری هم در اندازه گیری نقطه ای سطح گسسته و هم پیوسته استفاده می شود. در اندازه گیری پیوسته می توان به کمک مکانیزم هایی، موقعیت شناور را در هر نقطه داخل مخزن اندازه گیری و به اتاق کنترل ارسال کرد. یکی از روش ها استفاده از وزنه و کابل می باشد. در این نوع از سطح سنج یک وزنه تعادل به وسیله کابل به شناوری که درون مخزن قرار گرفته متصل می شود. این وزنه تعادل همزمان با بالا و پایین رفتن شناور در اثر تغییر ارتفاع سیال درون مخزن، جابجا و ارتفاع سیال را بر روی درجه بندی که در بیرون از مخزن نصب شده است نمایش می دهد. وزنه تعادل کشیدگی کابل را هنگام بالا و پایین رفتن شناور حفظ می کند. به جای کابل می توان از زنجیر نیز استفاده کرد. این سطح سنج ها دقیق می باشند اما معایب مکانیکی همچون گیر کردن کابل و یا اصطکاک میان کابل و چرخ دنده ها از دقت آنها می کاهد. هنگام نصب باید دقت شود که بخش های مکانیکی سنسور از تماس با هوا و سیال در امان بمانند زیرا این مساله موجب خوردگی و فرسایش تدریجی این قطعات می شود.این سنسورها به ندرت جهت ارسال سیگنال الکتریکی به کار می روند و تنها برای نمایش محلی ارتفاع سیال مناسب هستند و نصب آنها در مخازن با سقف بسته و تحت فشار نیازمند رعایت شرایط خاص مخزن خواهد بود. سطح سنج های شناوری تکنولوژی ساخت پیچیده ای ندارند و نسبت به انواع دیگر سطح سنج های پیوسته ارزانتر بوده و اندازه گیری دقیق و قابل اطمینانی را فراهم می نمایند. با توجه به اینکه شناور و لوله آن به طور کامل درون سیال فرو می روند باید از مواردی که در برابر خوردگی مقاوم

هستند، ساخته شوند. باید دقت شود که در هنگام نصب سنسور میله سنسور بصورت کاملا قائم درون سیال قرار گیرد و شناور به راحتی و آزادانه درون آن جابجا شود.

2-سطح سنج های تغییر مکانی

سنسورهای تغییر مکانی نیز براساس اصل شناوری ارشمیدس طراحی می شوند. همانطور که گفته شد براساس این اصل هرگاه یک جسم درون مایعی غوطه ور باشد، نیرویی رو به بالا و برابر با وزنش به آن وارد می شود که به آن نیروی شناوری گفته می شود.طبق قانون ارشمیدس وقتی جسمی در مایع فرو می رود به اندازه وزن مایعی که جسم را در برمیگیرد، از وزن آن کاسته می شود. از این قاعده در ساخت سنسورهای تغییر مکانی استفاده می شود. در این سنسورها یک جسم جابجا شونده که به اهرمی متصل است، وارد مخزن می شود. هنگامیکه مخزن خالی است تمام نیروی وزن جسم جابجا شونده، به اهرم وارد شده و ترانسمیتر این نیرو را به عنوان سطح صفر سیال منظور می کند. وقتی سیال وارد مخزن می شود و بخشی از جسم جابجا شونده را در برمی گیرد، همزمان با افزایش نیروی شناوری سیال، از میزان نیروی وزن جسم، که به اهرم وارد می شود کاسته می شود. ترانسمیتر سطح با اندازه گیری این تغییر نیرو، ارتفاع سیال درون مخزن را محاسبه می کند.

- ✓ **تفاوت سنسور تغییر مکانی با سنسور شناوری**

در سنسورهای شناوری، شناور تابع سطح سیال است و با تغییر سطح سیال بالا و پایین می رود و با کاهش ارتفاع سیال شناور می تواند تا کف مخزن هم پایین برود. شناور همیشه در سطح سیال قرار دارد و ترانسمیتر با تعیین موقعیت آن، ارتفاع سیال را اندازه گیری می کند. اما در سنسورهای تغییر مکانی جسم جابجا شونده به طور کامل در سیال فرورفته و جابجایی آن با تغییر ارتفاع سیال بسیار محدود است. ترانسمیتر با اندازه گیری میزان تغییر نیروی وزن جسم جابجا شونده، در اثر افزایش یا کاهش ارتفاع سیال، سطح سیال درون مخزن

را اندازه گیری می کند. تغییر نیروی وزن جسم جابجا شونده که با تغییر ارتفاع سیال متناسب است را می توان با روشهای مختلفی اندازه گیری کرد. محاسبه تغییر نیروی وارده به یک فنر و یا محاسبه تغییرات گشتاور وارده به یک میله از پرکاربردترین تکنیک هایی هستند که بدین منظور بکار گرفته می شوند.

3-سطح سنج اختلاف فشاری (Differential Pressure Level Transmitters)

همان طور که میدانیم فشار را می توان براساس نحوه عملکرد مانومترها و با محاسبه فشار ستونی از مایع اندازه گیری کرد. در سطح سنج هایی که براساس محاسبه فشار کار می کنند می توان طبق رابطه زیر با اندازه گیری فشار و معلوم بودن مقدار چگالی سیال(ρ) ارتفاع ستون مایع درون مخزن را بدست آورد:

$$Pabs=P+(\rho*g*H)$$

در این رابطه P فشار گازها و بخارات بالای مخزن است که در مخازن بدون سقف با فشار جو در منطقه برابر می باشد. جهت محاسبه ارتفاع سیال درون یک مخزن با این روش، نیاز است که فشار ستون مایع و فشار بالای مخزن اندازه گیری شوند. بدین منظور ترانسمیتر های اختلاف فشاری ساخته شده اند که با محاسبه اختلاف فشار مخزن، سیگنالی متناسب با ارتفاع سیال درون آن ارسال می کنند. فشاری که در این حالت اندازه گیری می شود فشار استاتیک مایعی می باشدکه در حالت سکون درون مخزن قرار دارد لذا به این روش، روش اندازه گیری هیدرواستاتیک سطح نیز گفته می شود. اندازه گیری سطح با این روش پیچیدگی خاصی ندارد و استفاده از این نوع سنسورها به دلیل سادگی و قیمت پایین بسیار فراگیر شده است.

4-سطح سنج اولتراسونیک(Ultrasonic Level Transmitters)

در سنسورهای اولتراسونیک و یا مافوق صوت از اصل انعکاس امواج استفاده می شود، ترانسمیتر امواج اولتراسونیک صوت را تولید و به سطح سیال ارسال می کند. این امواج پس از برخورد با سطح سیال برگشت داده می شوند، یک گیرنده امواج برگشت شده از سطح سیال را دریافت می کند و ترانسمیتر با محاسبه زمان رفت و برگشت امواج، ارتفاع سیال را اندازه گیری می کند. در اندازه گیری سطح در مواردی که امکان تماس با سطح سیال وجود نداشته باشد، می توان از سطح سنج های اولتراسونیک استفاده کرد. ارتفاع مخزن به عنوان اطلاعات اولیه به ترانسمیتر داده شده است و ترانسمیتر با اندازه گیری فضای خالی مخزن، ارتفاع سیال درون آن را محاسبه می کند. دقت این سنسورها به میزان بازگشت امواج از سطح سیال بستگی دارد. در مخازنی که سطح سیال آرام و بی تلاطم است، امواج به خوبی بازگشت می شوند، اما حضور حباب، بخار و ذرات گرد و غبار غلیظ در سطح سیال موجب جذب امواج شده و دقت اندازه گیری کاهش می یابد. همچنین هرچه امواج فاصله بیشتری را طی کنند از قدرت آنها کاسته می شود و میزان امواج بازگشت داده شده به سمت سنسور کمتر می شود لذا در صورتی که ارتفاع مخزن زیاد باشد باید از ترانسمیترهایی استفاده شود که امواج پرقدرت و با زاویه کمتری ارسال می کنند.

5-سطح سنج راداری(Radar Level Transmitters)

امواج راداری مخفف Radio Detection and Ranging و به معنای تشخیص رادیویی و مسافت یابی می باشند. در حدود 50 سال است که با ساخت سنسورهای راداری از این امواج جهت اندازه گیری ارتفاع سیال درون مخزن نیز استفاده می شود. امواج راداری در حقیقت امواج الکترومغناطیس هستند، طیف وسیعی از امواج همچون امواج رادیویی و تلویزیونی، امواج تلفن همراه و رادیواکتیو ایکس و گاما در محدوده امواج الکترومغناطیسی قرار می گیرند. در استفاده از امواج الکترومغناطیسی مساله ایمنی بسیار مهم است و امواج راداری مورد استفاده در ترانسمیترهای راداری از این حیث کاملا ایمن می باشند. فرکانس کاری این امواج در محدوده 3 تا 30 گیگاهرتز می باشد. این محدوده کاملا ایمن می باشد. میزان انرژی لازم جهت ارسال امواج رادار نیز در یک محدوده ایمن قرار دارد و از انرژی امواج ارسالی در وسایل خانگی نیز کمتر می باشد. انرژی امواج ارسالی ترانسمیترهای راداری در حدود 3 درصد انرژی نشتی مجازی است که وسایل خانگی نشت میکند.

سطح سنج راداری با ارسال امواج به سطح سیال، فضای خالی مخزن را اندازه گیری می کنند و با کسر مقدار اندازه گیری شده از ارتفاع واقعی مخزن که به عنوان مقدار اولیه به ترانسمیتر داده شده است، ارتفاع مایع درون مخزن را محاسبه می کنند.

6-سطح سنج رادیواکتیو

نوع دیگری از سنسورهای اندازه گیری سطح، از ارسال و دریافت امواج رادیواکتیو استفاده می کنند. این سنسورها بیرون از مخزن نصب می شوند و یک منبع رادیواکتیو امواج را به درون مخزن می فرستد، امواج ارسالی توسط

گیرنده رادیواکتیو که در سمت دیگر مخزن نصب شده است، دریافت و به سیگنال الکتریکی متناسب با سیستم کنترل تبدیل می شوند. وقتی که مخزنی خالی است میزان امواج دریافتی بیشتر می باشد و هنگامیکه سیال در مسیر امواج قرار می گیرد مقداری از امواج ارسالی منحرف شده و به گیرنده نمی رسند. امواج رادیو اکتیو در محدوده خطرناک برای انسان قرار دارند، لذا منبع ارسال امواج رادیواکتیو باید به گونه ای طراحی شود که هیچگونه خطری برای نفرات شاغل در واحد صنعتی نداشته باشد. این منبع مجموعه ای از امواج رادیواکتیو را بصورت موازی با مخزن ارسال می کند و از انتشار امواج به جهت های دیگر جلوگیری می کند. نیرو و انرژی امواج ارسالی وابسته به ضخامت مخزن و فاصله بین منبع رادیواکتیو و گیرنده امواج و چگالی سیال درون مخزن می باشد.

7-سطح سنج سروو موتوری(Servo Level Transmitter)

سطح سنج های سروو موتوری در دهه 1950 به بازار معرفی شدند این سنسورها در حقیقت ارتقاء یافته سطح سنج های تغییر مکانی می باشند. در این سطح سنج ها جسم جابجا شونده به یک وزنه به قطر تقریبا 50 میلیمتر و با چگالی بالاتر از چگالی سیال مورد اندازه گیری تبدیل شده است. مبنای کار اندازه گیری در این سطح سنج ها نیز بر اساس اندازه گیری نیروی شناوری ارشمیدوس می باشد. استفاده از فناوری زیرپردازنده امکان تصحیح

بسیاری از خطاهای محیطی از قبیل وزن و یا تغییر شکل مخزن که به علت وجود فشار هیدرواستاتیکی به وقوع می پیوندد را فراهم نموده است.

انواع سطح سنج نقطه ای

1- سطح سنج شناوری(Float Level Switch)

این سطح سنج در مورد اول سطح سنج های پیوسته بررسی شد.

2-سطح سنج خازنی یا رسانایی(Conductivity/Capacitive Level Switches)

یکی دیگر از روشهای اندازه گیری نقطه ای سطح استفاده از اثر خازنی و یا هدایت الکتریکی است. این سطح سنج ها به سنسورهای RF (Radio Frequency) معروف می باشند. سنسورهای RF از طریق ورود یک میله فلزی به داخل مخزن با سیال در تماس می باشند. با اعمال ولتاژ به میله، تغییرات هدایت الکتریکی و یا خاصیت خازنی اندازه گیری می شود. تفاوت اصلی بین سنسورهای هدایت الکتریکی و خازنی درنوع و فرکانس ولتاژ اعمال شده به میله می باشد. سطح سنج های خازنی قادرند به صورت گسسته و پیوسته به مانیتور کردن سطح مخزن بپردازند. این موضوع بخصوص برای مواد پودری که یکی از گزینه ها برای انتخاب، سوئیچ سطح خازنی میباشد اهمیت دارد. عملکرد سطح سنج طبق خاصیت خازنی ایجاد شده بین سطح مخزن و الکترود سنسور میباشد. خاصیت دی الکتریک خازن وقتی بیشتر میشود که مقدار مواد بین سطح مخزن و الکترود بیشتر شود درنتیجه خروجی که با ارتفاع مخزن متناسب است را میتوان محاسبه کرد.

در نمونه های گسسته سوئیچ سطح یا خروجی سوئیچ ، یک سنسور مجاورتی خازنی میباشد. یکی از مزیت های این روش پایداری عملکرد در مدت زمان طولانی و اندازه گیری جزء متحرک میباشد.

سطح سنج های خازنی در مواد پودری پرک یا گرانول و نیز برای تعیین نقاط مرزی دو مایعی که در مخزن چگالی متفاوتی دارند، کاربردی هستند.

3-سطح سنج ارتعاشی(Vibrating Level Switch)

سنسورهای ارتعاشی شامل یک تیغه می باشند که با فرکانس طبیعی نوسان می کند، چنانچه تیغه با سیال داخل مخزن برخورد نماید فرکانس نوسان آن تغییر خواهد کرد. میتوان با اندازه گیری این تغییر فرکانس، سطح سیال را در ارتفاع مورد نظر اندازه گیری کرد. سنسورهای ارتعاشی در دو نوع میله ای و دیاپازونی ساخته می شوند. سنسورهای دیاپازونی از دو شاخه که چنگاله گفته می شود ساخته شده اند. شاخه ها با فرکانس طبیعی نوسان می کنند، هنگامیکه سیال درون مخزن به سنسور می رسد و با شاخه های در حال نوسان برخورد می کند موجب تغییر فرکانس نوسان آنها می شود، با اندازه گیری این تغییر فرکانس، سطح سیال درون مخزن اندازه گیری می شود. سنسورهای دیاپازونی برای مایعات مناسب می باشند اما چنانچه مایع مورد نظر حاوی ذرات جامد باشد، احتمال گیر کردن این ذرات بین تیغه های سنسور وجود دارد لذا برای جامدات، بهتر است که از سنسورهای میله ای استفاده شود.

4-سطح سنج با پره متحرک(Paddle Wheel Switches)

این سنسورها از پره هایی تشکیل شده اند که به کمک یک موتور الکتریکی و با دورکم به چرخش در می آیند. وقتی که سطح سیال بالا می آید و با این پره ها برخورد می کند، مانع چرخش پره ها می شود، در این زمان سوییچ عمل کرده و سطح سیال را نشان می دهد. این سنسورها بیشتر در اندازه گیری سطح جامداتی همچون سیلوهای گندم و سیمان بکار می روند و بسیار ساده، ارزان و قابل اطمینان هستند. تغییرات دما و چگالی برنحوه عملکرد این سنسورها تاثیر گذار نیست. چنانچه این سنسورها برای اندازه گیری سطح مایعات به کار گرفته شوند باید به این مساله توجه داشت که چسبندگی سیال، از میزان طراحی شده برای سنسور بیشتر نباشد. این سنسورها نسبت به شوک و ارتعاش در سیستم حساس هستند و نباید در محلی که لرزش های زیادی وجود دارد نصب شوند. قبل از تعیین محل نصب سنسور باید محل ورودی و خروجی مواد جامد به مخزن و نحوه

قرارگرفتن مواد و شیب آنها درون مخزن بررسی شود، سقوط اجسام از بالا نیز منجر به شکستن پره های سنسور خواهد شد، لذا سنسور نباید در مسیر ورودی مخزن نصب شود. هنگامیکه سنسور در کنار مخزن نصب می شود باید شیبی به سمت پایین داشته باشد تا مواد اطراف پره ها نیز همزمان با پایین رفتن سطح مواد درون مخزن، به سمت پایین هدایت شود.

5-سطح سنج های نوری

سطح سنج های نوری از یک فرستنده و گیرنده نوری ساخته شده اند و براساس بازتابش نور از سطح اجسام و یا عبور نور از درون سیال عمل می کنند. سوییچ های نوری انواع مختلفی دارند، نوع اول سنسورهایی هستند که گیرنده و فرستنده آنها در مقابل یکدیگر نصب می شوند. نوری که از فرستنده ساطع می شود تا زمانیکه به مانعی برخورد ننماید (مانند سطح مایع درون مخزن) درمسیر مستقیم حرکت کرده و به گیرنده میرسد. زمانی که سطح سیال بالا می آید و مابین گیرنده و فرستنده قرار می گیرد، پرتوهای نور در اثر وقوع پدیده شکست منحرف شده و به گیرنده نمی رسند، دراین زمان خروجی سطح سنج فعال شده و وجود سیال در ارتفاع مشخص را گزارش می نماید. در نوع دیگری از این سنسورها گیرنده و فرستنده در بالای مخزن و در کنار یکدیگر نصب می شوند. نوری که از فرستنده ساطع می شود، سنسور با اندازه گیری زمان رفت و برگشت نور، سطح سیال درون

مخزن را اندازه گیری می کند. چنانچه ارتفاع سیال درون مخزن افزایش یابد زمان رفت و برگشت نور کاهش خواهد یافت.

جهت اندازه گیری گسسته سطح سیال از سنسورهای دیگری نظیر سنسور حرارتی و دیافراگمی نیز استفاده می شود که از کاربرد کمتری نسبت به سنسورهای معرفی شده برخوردار هستند.

ترانسمیتر (Transmitter)

ترانسمیتر از ترکیب دوکلمه ای انتقال (Transfer) و اندازه گیری (Metering) تشکیل شده است و به معنی تجهیزی می باشد که بتواند کمیت فیزیکی را اندازه گیری کرده و سپس سیگنال اندازه گیری شده را برای کنترل کننده ارسال نماید.

در واقع ترانسمیتر سیگنالهای غیر استاندارد خروجی سنسورها را دریافت و به سطح قابل قبولی برای کنترلرها تبدیل میکند بنابراین خروجی یک ترانسمیتر یک سیگنال استاندارد است.

ترانسمیترها در دو مدل الکترونیکی و یا نیوماتیکی می باشندکه در هر دوحالت، سیگنالی استاندارد را ارسال می نمایند تا برای تجهیزاتی که در LOOP کنترل قراردارند، قابل فهم می باشد. در ترانسمیتر های نوع الکترونیکی جریان4 تا 20 میلی آمپر و درنوع نیوماتیکی فشار هوای 3 PSI تا 15 PSI از سوی ترانسمیتر به کنترلرهای الکترونیکی و یا نیوماتیکی ارسال می شود.

انواع ترانسمیتر

ترانسمیتر ها از نظر نوع و کاربرد به چند دسته تقسیم می شوند:

- ترانسمیتر فشار

- ترانسمیتر دما

- ترانسمیتر رطوبت

- ترانسمیتر فلو یا جریان سیالات

- ترانسمیتر سطح یا ارتفاع مخازن

- ترانسمیتر وزن

- ترانسمیتر سرعت

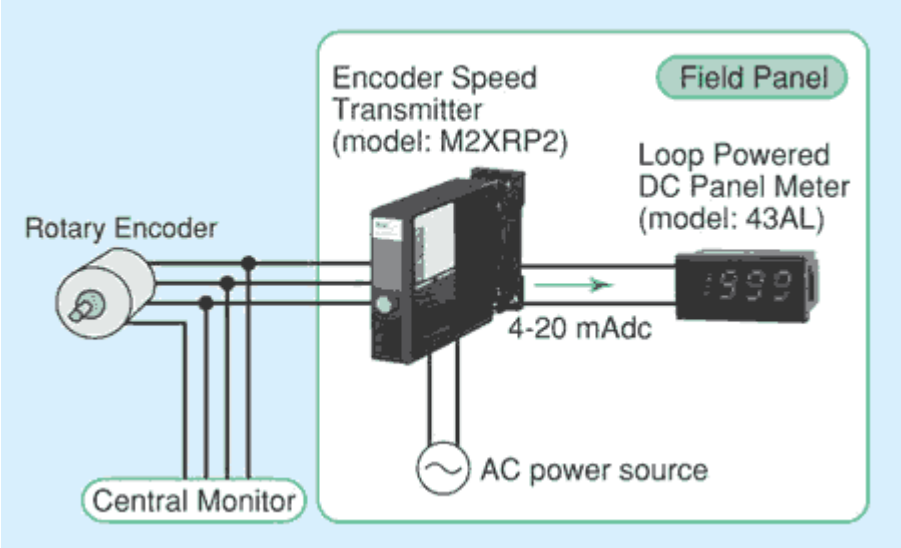
و.....

ترانسمیترهای فوق بسته به نوع استفاده به صورت های زیر قابلیت نصب در محل های گوناگون را دارند.

- Rail mount
- Head mount
- Field mount
- Wall mount

تفاوت ترانسمیتر با ترانسدیوسر

یک ترانسدیوسر بنا به تعریف ، قطعه ای است که وظیفه تبدیل حالات انرژی به یکدیگر را برعهده دارد، بدین معنی که اگر یک سنسور فشار همراه یک ترانسدیوسر باشد، سنسور فشار پارامتر را اندازه می گیرد و مقدار تعیین شده را به ترانسدیوسر تحویل می دهد،سپس ترانسدیوسر آن را به یک سیگنال الکتریکی قابل درک برای کنترلر و صد البته قابل ارسال توسط سیم های فلزی ، تبدیل می کند.



ترانسمیتر وسیله ای است که یک سیگنال الکتریکی ضعیف را دریافت کرده و به سطوح قابل قبول برای کنترلرها و مدارهای الکترونیکی تبدیل می کند ، مثلاً یک حلقه فیدبک سیگنالی در سطح ماکروولت یا میلی ولت یا میلی آمپر تولید می کند و این سیگنال ضعیف می تواند با عبور از ترانسمیتر به سیگنالی در سطوح صفر تا 10 ولت و یا 4 تا 20 میلی آمپر تبدیل شود. ترانسمیترها عموماً از قطعاتی مثل op-amp برای تقویت و خطی کردن این سطوح ضعیف سیگنال استفاده می کنند. سنسورها و ملحقات آنها مثل ترانسدیوسرها را در گروه های بزرگی تحت عنوان ابزار دقیق قرار داده و آنها را بر اساس نوع انرژی قابل استفاده و روشهای تبدیل ، دسته بندی می کنند .

ترانسدیوسر های اندازه گیری، تغییرات حس شده توسط سنسورها را به سیگنالهای آنالوگ استاندارد از قبیل 4± تا 20 میلی آمپر، 500 mV ± تا 10 V تبدیل می کنند.

✓ **انواع ترانسدیوسر**

- ترانسدیوسر ولتاژ
- ترانسدیوسر جریان
- ترانسدیوسر فرکانس
- ترانسدیوسر توان اکتیو و راکتیو
- ترانسدیوسر ضریب توان (کسینوس فی)
- ترانسدیوسر قابل برنامه ریزی
- ترانسدیوسر مقاومت
- ترانسدیوسر دما
- ترانسدیوسر ایزوله
- ترانسدیوسر فشار
- ترانسدیوسر خازنی

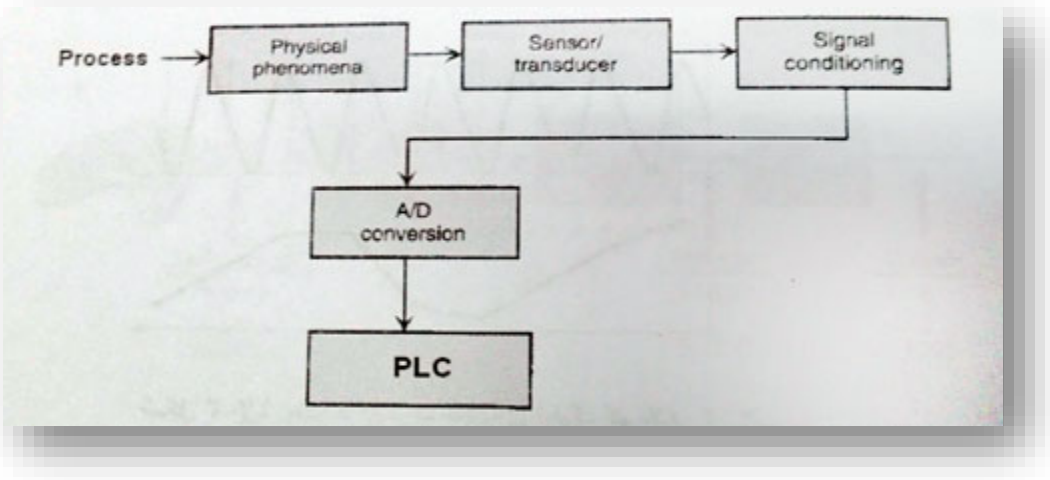
کارتهای ورودی و خروجی آنالوگ

کارت های ورودی آنالوگ

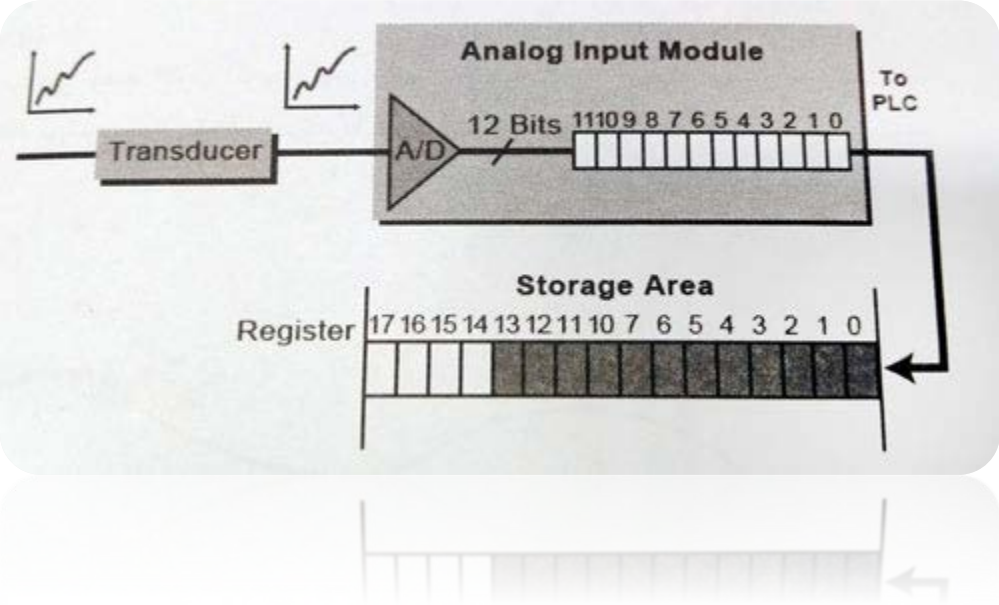
سیگنال های ورودی آنالوگ بر خلاف نوع DISCRETE (دیجیتال) مقادیر پیوسته داشته و دو وضعیتی نیستند. پارامترهای فرآیندی مانند دما،فشار،فلو و ارتفاع سطح که در یک بازه مشخص تغییر می کنند از جمله مهمترین سیگنالهای آنالوگ ورودی برای PLC محسوب می شوند که در مطالب قبل به تفصیل مورد بررسی قرار گرفتند.این نوع ورودی ها در واقع سطح سیگنال آنالوگ را به دیجیتال تبدیل می کنند. برخی از مهمترین سطوح استاندارد برای ورودی های آنالوگ عبارتند از:

- 4-20mA
- 1-5V
- 0-10V
- -2.5V/+2.5V
- -5V/+5V
- -10V/+10V

پارامتر فرآیندی توسط سنسور یا Transdusr به سیگنال الکتریکی تبدیل شده سپس توسط ترانسمیتر با تبدیل به سیگنال الکتریکی استاندارد به کارت ورودی آنالوگ ارسال می شود. سیگنال الکتریکی پس از تبدیل به دیتا در اختیار CPU قرارداده می شود. معمولا ترانسدیوسر و ترانسمیتر در یک دستگاه عرضه می شوند.



نحوه تبدیل سیگنال آنالوگ ورودی در پی ال سی



تبدیل و ذخیره سازی آنالوگ ورودی در پی ال سی

در سیگنال دیجیتال اگر نویزی روی خط ورودی به کارت بیفتد و سطح ولتاژ را کمی تغییر دهد اهمیتی ندارد زیرا سطح سیگنال معرف فقط یک وضعیت است ولی در سیگنال آنالوگ اگر نویزی روی سیگنال سوار شود توسط مبدل A/D تبدیل می گردد و به غلط معرف تغییر پارامتر های فرآیندی خواهد بود . برای جلوگیری از این تأثیر لازم است تمهیدات لازم برای حفاظت سیگنال نسبت به نویز پیش بینی شود از مهمترین کارها، انتقال سیگنال با کابل شیلد دار است که شیلد آن زمین شده باشد.

کارت های آنالوگ معمولاً بیش از یک ورودی را می پذیرند بسیاری از سازندگان کارت های آنالوگ 2 و 4 و 8 و حتی 16 ورودی عرضه می کنند. از آنجا که معمولاً فقط یک A/D داخل کارت وجود دارد نیاز به یک Multiplexer می باشد که به ترتیب کانال ها را روی مبدل A/D بفرستد.

قبل از ورود سیگنال به مبدل A/D لازم است عمل تقویت فیلتر سازی و حفاظت انجام شده باشد. کانال های ولتاژی کارت آنالوگ معمولاً High Impeddance(درحد مگا اهم) هستند، بنابراین تأثیر نامطلوبی روی سیگنال اندازه گیری شده نمیگذارد.

کانال های جریانی کارت آنالوگ دارای امپدانس پایین (OHM500-250) است تا امکان انتقال جریان از ترانسمیتر به کانال وجود داشته باشد.

ورودی های ولتازی آنالوگ می توانند از یک COM مشترک در کارت استفاده کنند که به آنها Single Ended گفته میشود این نوع ورودی ها نویز پذیری زیادی دارند . اگر هر ورودی دارای COM خاص باشد به آنها Double Ended یا Differential گفته می شود و مشکل نویز پذیری ندارد ولی تعداد کانال بیشتری از کارت را اشغال می کند.

همانطور که می دانیم کارت های ورودی انالوگ در PLCS7-300 به صورت SM331 نشان داده میشوند.

در S7-300 سه مدل کارت آنالوگ وجود دارد:

- ✓ SM331 یک کارت عمومی می باشد.
- ✓ SM331TC یک کارت مختص ترموکوپل می باشد.
- ✓ SM331RTD یک کارت متخص RTD می باشد.

SM331 TC

در این کارت ها امکان اتصال مستقیم ترموکوپل،جهت اندازه گیری حرارت وجود دارد.

انواع ترموکوپل ها عبارتند از : B،N،J،K،S،T،U،E،L

SM331 RTD

این کارت برای محاسبه حرارت با استفاده از ترمومترهای استاندارد با دقت بالا مناسب است. زمانی که از کارتهای خاص جهت اندازه گیری دما استفاده می شود دیگر نیازی به ترانسمیتر نمی باشد بنابراین در این کارت می توان

یک PT100 را مستقیماً به آن متصل کرد. در حالی که در کارتهای عمومی برای اتصال PT100حتماً بایستی از ترانسمیتر استفاده کرد.

سایر ترمومترهای قابل اتصال به این کارت عبارتنداز: PT200، PT500، PT1000، NI100، NI200، NI500

مبدل A/D

کارتهای ورودی آنالوگ یک مبدل (A/D) Analog to Digital هستند. این کارتها سیگنالهای الکتریکی خروجی سنسور را دریافت و با پردازش، آن ها را به مقادیر دیجیتال قابل فهم برای CPU تبدیل میکنند.

این مقدار دیجیتال که در محدوده ی word و به صورت عدد صحیح تعریف شده است به صورت ذیل است:

تک قطبی	[0 , 27648]
دو قطبی	[-27648 , +27648]

تک قطبی: سیگنالهای استاندارد که شامل بازه ی منفی نیستند، مانند mA [4 , 20]

دو قطبی : سیگنالهای الکتریکی که شامل بازه ی منفی هستند، مانند mA [-20 , +20]

دقت در کارت ورودی آنالوگ

معمولاً مبدل های A/D دارای قدرت تفکیک 8 تا 16 بیتی هستند. منظور از دقت، توانایی تشخیص حداقل تغیرات سیگنال است. هر چه دقت کارت عدد بزرگ تری باشد یعنی تغییرات ریز تری را مشاهده میکند و دقیق تر است.

در این میان کارت های 16BIT دارای بیشترین میزان دقت و 8BIT ها دارای کمترین میزان دقت هستند.

آدرس کانال ورودی در کارت آنالوگ:

آدرس این کانال ها در محدوده ی WORD است.

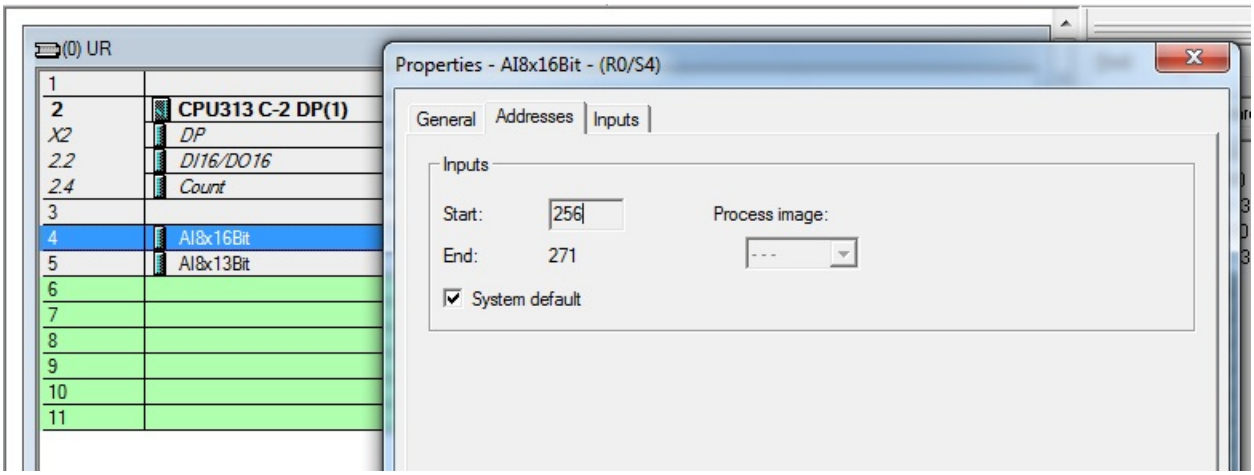
فرمت نوشتن آدرس ها به صورت ذیل میباشد

PIW XXX

که در آن XXX عدد بایت آغازین این محدوده است.

نکته : این آدرس ها به هیچ عنوان نباید تغییر پیدا کنند زیرا آدرس ها مخصوص فضای P هستند که در این فضا تغییرات به صورت لحظه ای اعمال میشود.

در محیط HW Config یک کارت ورودی آنالوگ از قسمت simatic 300 / SM-300/ AI-300 می آوریم. مشخصات کارت در قسمت Properties قابل مشاهده است.



همانطور که مشاهده می کنید آدرس کانال اول PIW256 میباشد.

کارت از نوع ورودی است که دارای هشت کانال مجزاست که به صورت جفتی قابل تنظیم به صورت ولتاژی یا جریانی چهار سیمه اند.

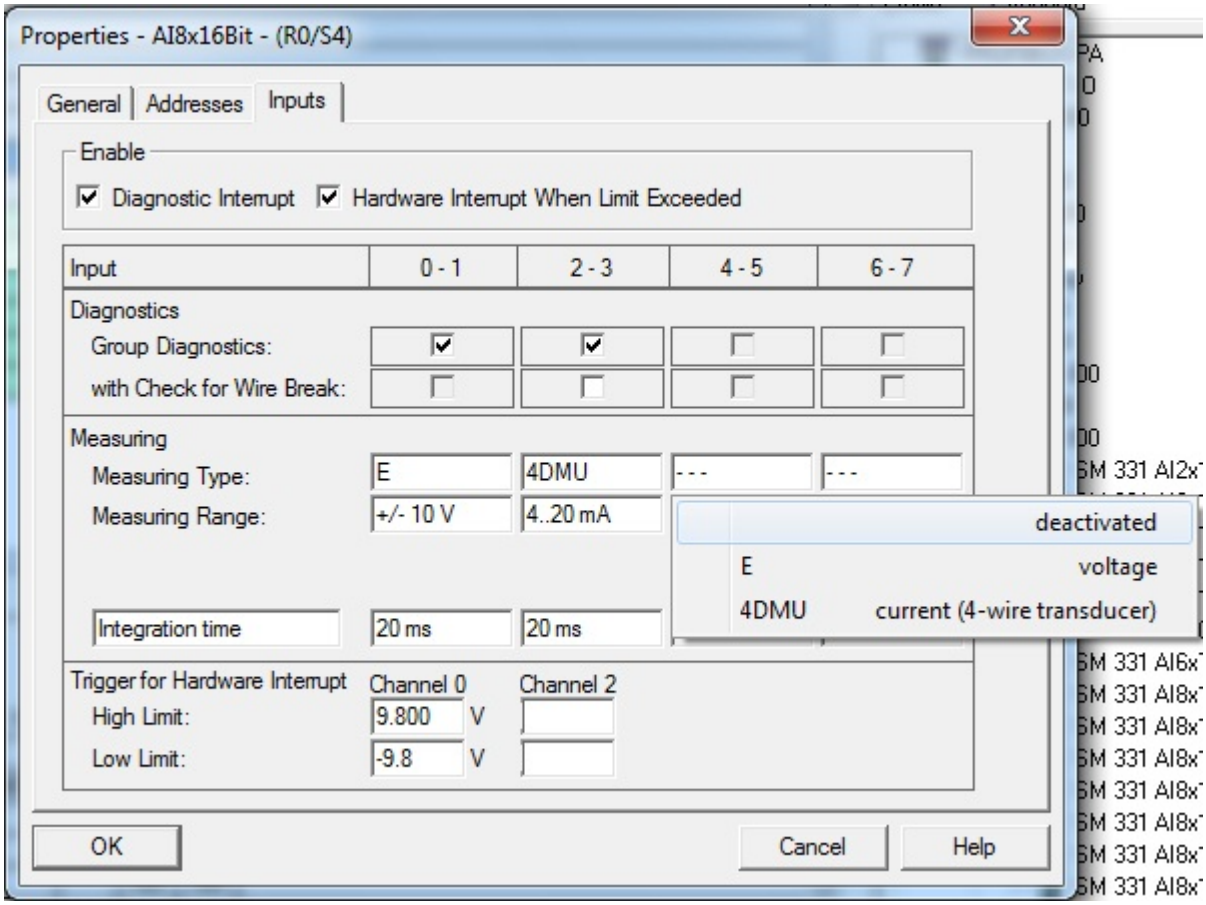
با کلیک در قسمت Measuring type میتوان نوع جریانی یا ولتاژی یا اهمی یا در مواردی ترموکوپلی یا ترمومتری سنسور را مشخص کرد.

در قسمت Measuring range میتوان رنج هر کدام از تایپ های ذکر شده را تنظیم کرد.

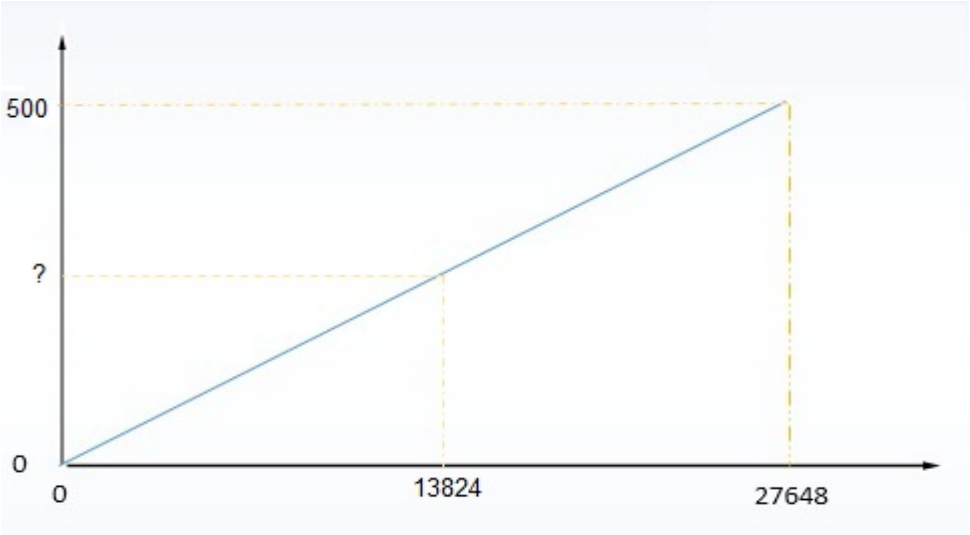
این کارت ها دارای قابلیت وقفه هنگام خطا (Diagnostic interrupt) و وقفه هنگام رویداد (Hardware interrupt) هستند.

High limit و low limit مربوط به وقفه رویداد hardware interrupt میباشد که با توجه به تایپ سیگنال ورودی برای آن حد بالا و پایین تعریف میشود و با گذر از این حد ها بلوک وقفه مربوطه فراخوانی میشود.

اگر از کانالی استفاده نمیشود در این تنظیمات به جهت افزایش سرعت بهتر است در حالت deactivated قرار گیرد.



معادله خط درمبدل A/D



فرض کنید یک سنسور دما مقادیر 0 تا 500 درجه را Sens میکند. این سنسور به ازای 0 درجه 0 ولت و به ازای 500 درجه مقدار 10 ولت را به کارت ورودی آنالوگ میدهد.

مقدارآنالوگ ولتاژی V [10 , 0] به کارت ورودی آنالوگ وارد میشود، این مقدار در مبدل A/D به مقادیر [27648 , 0] دیجیتالی تبدیل میشود. این مقدار دیجیتال اعداد صحیح مطلوب ما نیست و میبایست با استفاده از معادله خط این مقادیر را به مقادیر حقیقی کمیت فزیکی اولیه (در اینجای دمای [500 , 0] درجه) برای برنامه نویسی تبدیل کنیم.

این کار توسط فرمول FC105 انجام میشود:

OUT= [(FLOAT_IN – K1) / (K2 – K1) * (HI_LIM – LO_LIM)] + LO_LIM

OUT : خروجی کمیت حقیقی در لحظه

FLOAT_IN : آدرس کانال ورودی کارت

K2 : 27648

K1 : 0 (تک قطبی) یا 27648 - (دو قطبی)

LO_LIM : مینیموم مقدار کمیت فزیکی

HI_LIM : ماکزیموم مقدار کمیت فزیکی

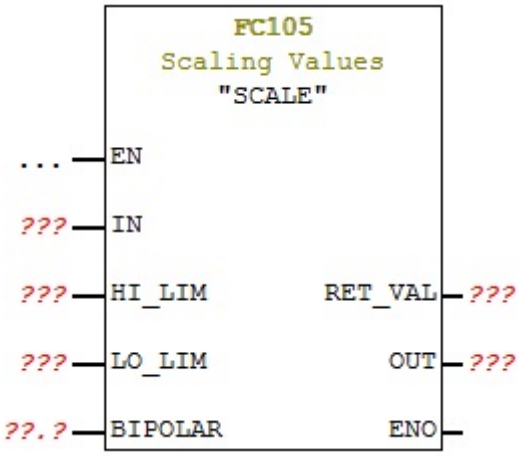
فراخوانی بلوک **FC105 “SCALE”**

برای تبدیل مقدار کمیت فیزیکی تبدیل شده توسط سنسور به ولتاژ و جریان و ... و تبدیل این سیگنال الکتریکی به اعداد دیجیتال در بازه مبدل A/D به کمیت واقعی اولیه، از این بلوک سیستمی که قابلیت فراخوانی دارد استفاده میکنیم.

ابتدا وارد محیط برنامه نویسی میشویم.

در قسمت نوار ابزار سمت چپ برنامه و در مسیر زیر میتوان یک FC105 را درون Network اضافه کرد

Libraries / Standard Libraries / TI-S7 Converting Blocks / FC105 SCALE CONVERT



بلوک FC105 به صورت یک تابع نوشته شده است. توجه شود که این بلوک، یک FC به Block های موجود اضافه میکند که باید به همراه OB1 در سیمولیشن دانلود شود.

EN : برای شرطی کردن اجرای تابع FC105.

IN : ورودی تابع که آدرس کانال ورودی کارت آنالوگ میباشد. فرمت آن به صورت PIWXXX است.

HI_LIM : بالاترین حد از کمیتی که سنسور میتواند اندازه بگیرد (مثلا 500.0 درجه).

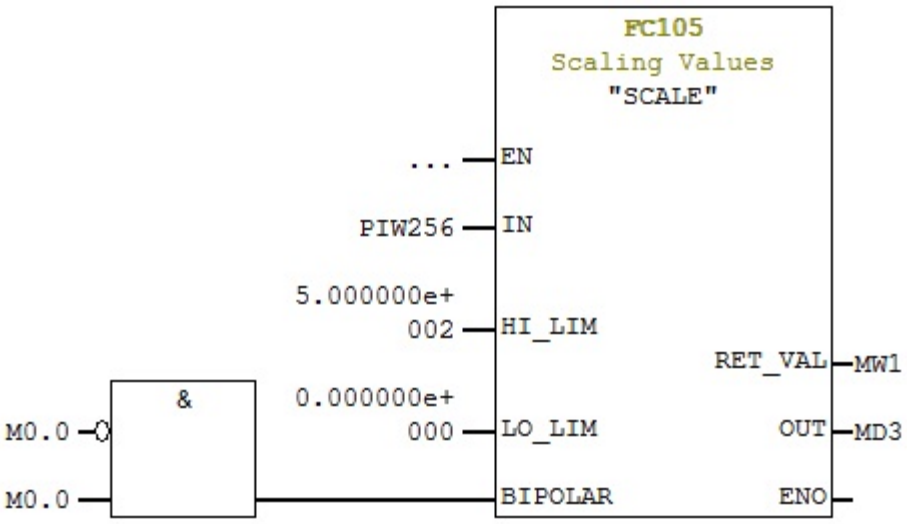
LO_LIM : پایینترین حد از کمیتی که سنسور میتواند اندازه بگیرد (مثلا 0.0 درجه).

BIPOLAR : مشخص کننده تک قطبی یا دوقطبی بودن سنسور. مقدار صفر منطقی برای تک قطبی ها و مقدار یک منطقی برای دوقطبی ها

RET_VAL : خروجی کد خطا های ممکن که باید در یک WORD از حافظه ذخیره شود.

OUT : خروجی که مقدار حقیقی از کمیت سنس شده توسط سنسور که باید در یک DWORD از حافظه ذخیره شود.

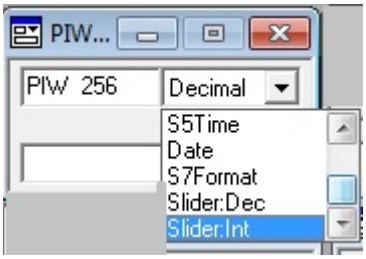
ENO : خروجی تابع هنگامی که تابع در صورت برقراری شرط اجرا مقدار یک منطقی میدهد.

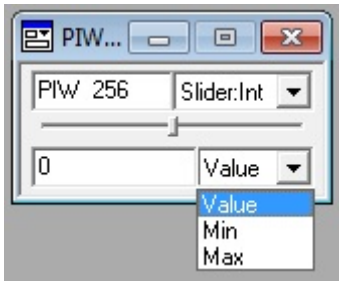


برای ساخت صفر منطقی یک بیت را با نات همان بیت AND میکنیم.

برای ساخت یک منطقی یک بیت را با نات همان بیت OR میکنیم.

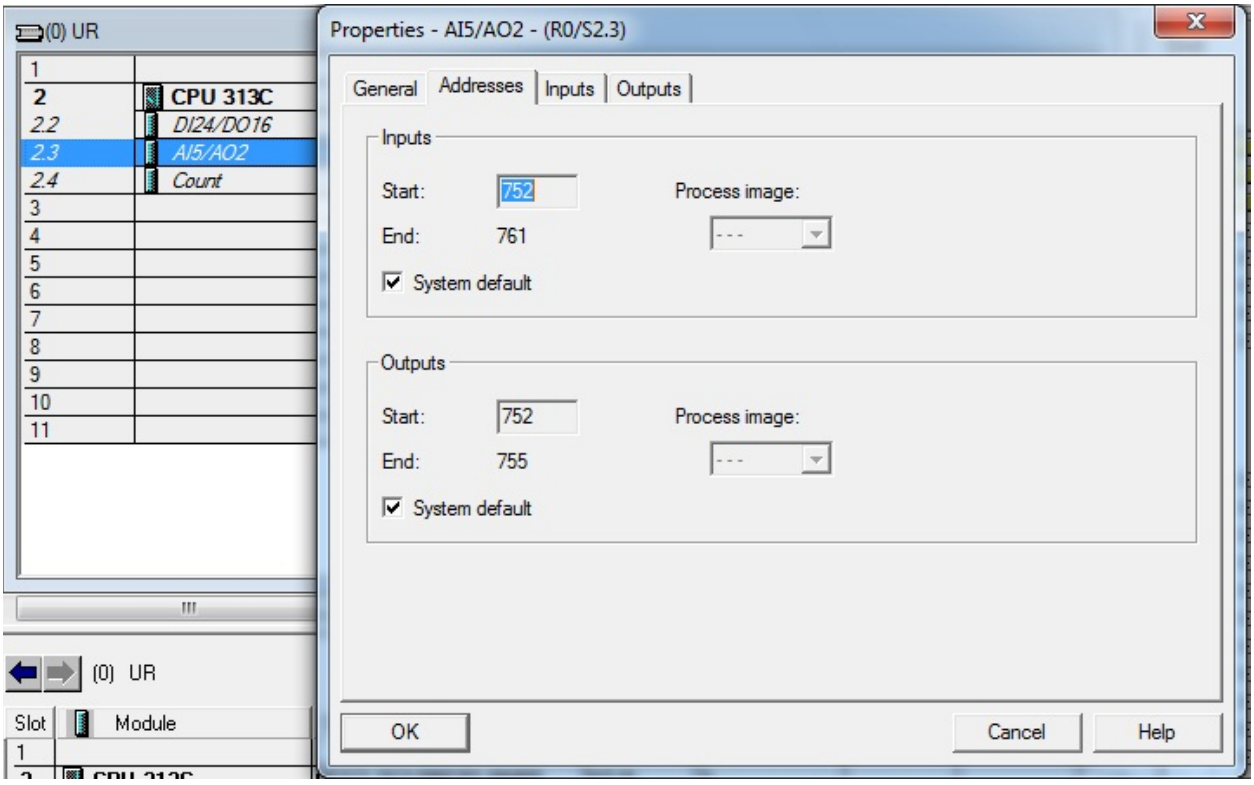
برای شبیه سازی سنسور در سیمولیشن در قسمت Insert یک Input میاوریم و آدرس کانال آنالوگ مورد نظر را مینوسیم. در قسمت Type آن Slider int را انتخاب میکنیم و مقدار Max آن را عدد 27648 و Min آن با توجه به تک قطبی یا دوقطبی بودن به ترتیب مقادیر صفر و 27648- میدهیم. با حرکت دادن اسلایدر میتوان نقش سنسور را در سیمولیشن اجرا کرد.



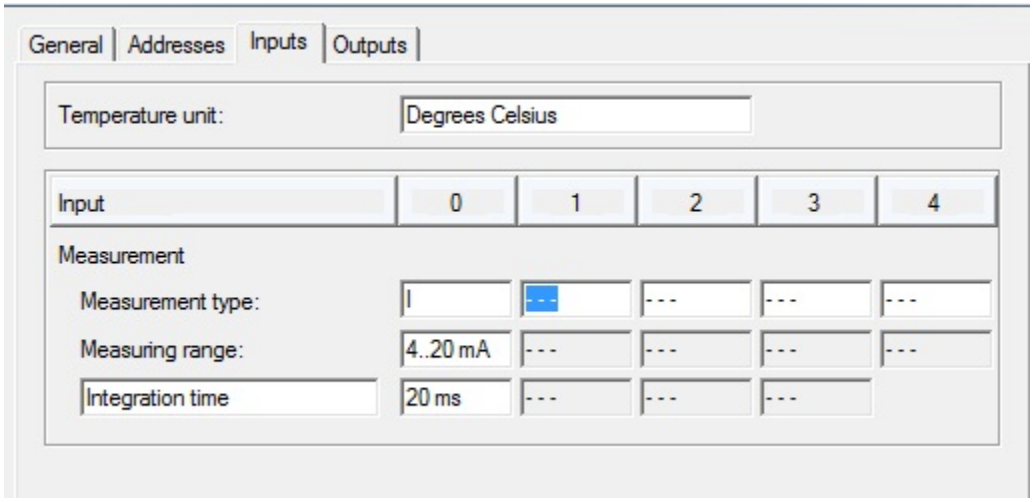


مثال : یک سنسور دما PT100 دمای صفر تا 350 درجه را در یک کوره سنس میکند و به ازای صفر درجه مقدار 4mA و به ازای 350 درجه مقدار 20mA در خروجی ترانسمیتر خود به کانال اول از کارت آنالوگ CPU 313C ارسال میکند. برنامه ای بنویسید که مقدار دمای کوره را نشان دهد و هنگامی که دما به 300 درجه رسید خروجی آلارم Q0.0 را فعال کند.

مرحله اول انتخاب CPU مور نظر و مشاهده آدرس کارت آنالوگ ، این CPU compact است.

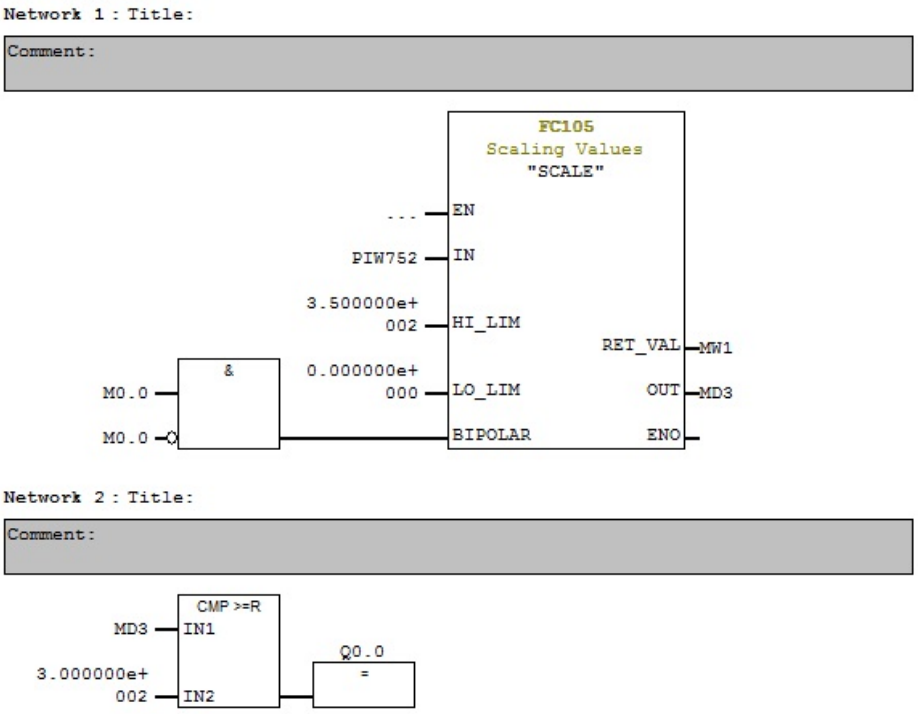


مرحله بعدی تعیین تایپ ورودی و رنج آن در تب Input است.

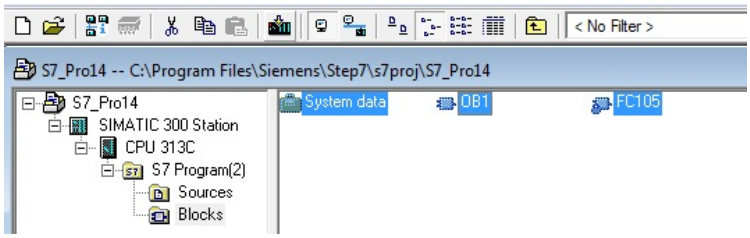


محیط HW Config را Save& Compile میکنیم.

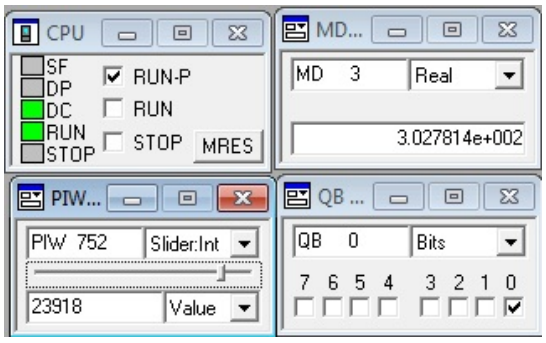
در OB1 تابع FC105 را فراخوانی میکنیم و مقادیر مربوطه را میدهیم.



OB1 را SAVE میکنیم و سیمولیشن را باز میکنیم و بلوک های موجود را به همراه System Data روی سیمولیشن دانلود میکنیم.



با تغیر اسلایدر PIW752 دما در MD3 نمایش داده شده و هنگامی که از 300 بیشتر شود خروجی Q0.0 روشن میشود



معرفی برخی کارتهای ورودی آنالوگ (AI)

SM331 AI2*12BIT

کارت فوق یک کارت 2 کاناله و 12 بیتی با Order Number: 6ES7 331-7KB00-0AB0 است. ورودی این کارت می تواند: ولتاژ، جریان، مقاومت، ترمومتر و ترموکوپل باشد.

رنج قابل دسترس برای ورودی های مختلف این کارت به صورت زیر می باشد:

ولتاژ: ±80 mV ، ±250 mV، ± 500 mV، ± 1 V، ± 2.5 V، ± 5 V ، 1/5 V، ± 10 V

جریان: ±3.2 mA ، ± 10 mA ، 0/20 mA، 4/20 mA، ±20 mA

مقاومت: 600 Ohm،300 Ohm ،150 Ohm

ترمومتر: PT100 استاندارد، NI100 استاندارد

ترموکوپل: نوع E،K،J، N

فرکانس تداخل این کارت 50Hz می باشد.

SM331 AI4*0/4to 20mA,Ex

یک کارت 4 کاناله و 15بیتی برای ورودی جریان باOrder Number: 6ES7 331-7RD00-0AB0 می باشد. رنج ورودی این کارت 0/20 mA و 4/20 mA می باشد. فرکانس تداخل این کارت 50Hz می باشد.

SM331 AI6*TC

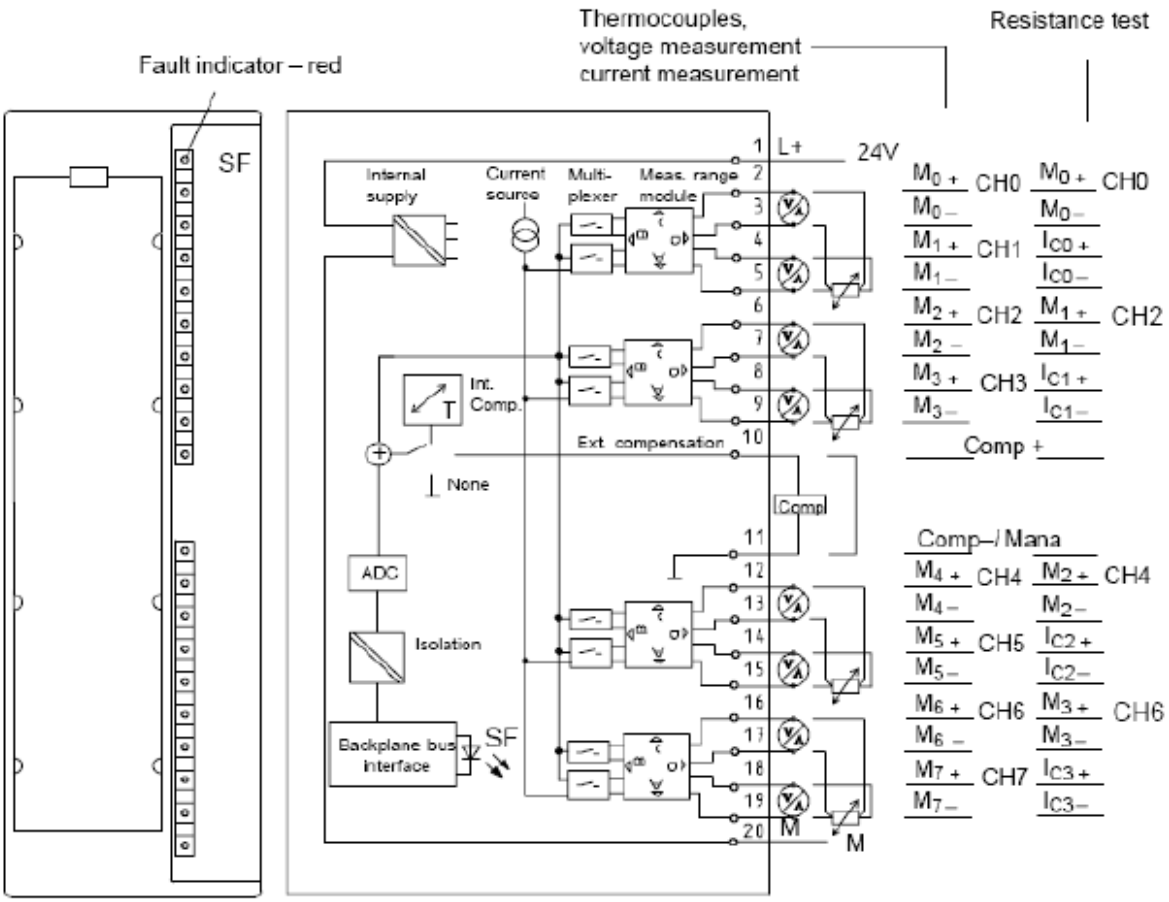
این کارت یک کارت 6 کاناله و 16 بیتی برای ورودی ولتاژ و ترموکوپل با OrderNumber:6ES7331-7PE10-0AB0 است. رنج قابل قبول این کارت به صورت زیر است.

ولتاژ:± 80 mV ، ±250 mV، ± 500 mV، ± 1 V، ± 25 V، ± 50 V

ترموکوپل: نوع TXK/XKL،C،U،T،L،S،R،B،E،K،J،N

SM331 AI8*12BIT

کارت فوق یک کارت 8 کاناله و 12 بیتی با Order Number: 6ES7 331-7KF00-0AB0 است. ورودی این کارت می تواند:ولتاژ،جریان،مقاومت و ترموکوپل باشد.



رنج قابل دسترس برای ورودی های مختلف این کارت به صورت زیر می باشد:

ولتاژ: ±80 mV ، ±250 mV، ± 500 mV، ± 1 V، ± 2.5 V، ± 5 V ، 1/5 V، ± 10 V

جریان: ±3.2 mA ، ± 10 mA ، 0/20 mA، 4/20 mA، ±20 mA

مقاومت: 150 Ohm، 300 Ohm،600 Ohm

ترموکوپل: نوع E،KJ، N

فرکانس تداخل این کارت 50Hz می باشد.

SM331 AI8*13BIT

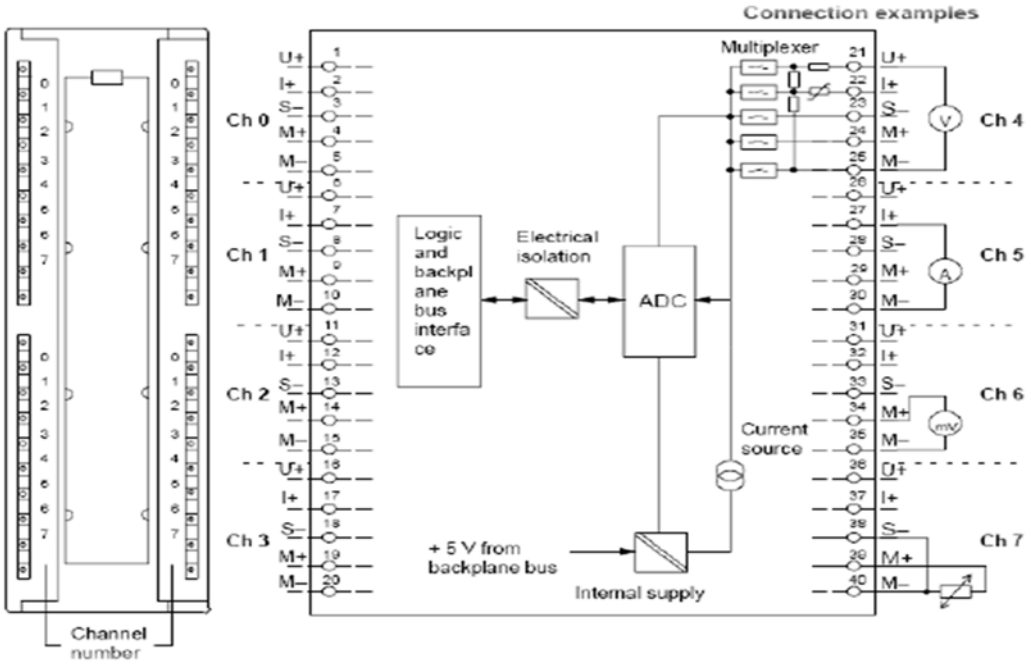
کارتی 8 کاناله و 13 بیتی با Order Number: 6ES7 331-1KF00-0AB0 است. ورودی آن ولتاژ، جریان، مقاومت و ترمومتر در رنج های زیر است.

ولتاژ: ±50 mV ، ±500 mV، ± 1 V، ±5 V، ±10 V ، 1/5 V، ±0/10 V

جریان: 0/20 mA، 4/20 mA، ±20 mA

مقاومت: 600 Ohm، 6 KOhm

ترمومتر: PT100 استاندارد، PT100 cl.



SM331 AI8*14BIT

کارتی 8 کاناله و 14 بیتی با Order Number: 6ES7 331-7HF00-0AB0 است. ورودی آن ولتاژ و جریان در رنج های زیر است:

ولتاژ: ± 1 V، ±5 V، ±10 V، 1/5 V

جریان: 0/20 mA، 4/20 mA، ±20 mA

SM331 AI8*16BIT

کارت فوق یک کارت 8 کاناله و 16 بیتی با Order Number: 6ES7 331-7NF00-0AB0 است.

ورودی این کارت ولتاژ و جریان در رنج های زیر است:

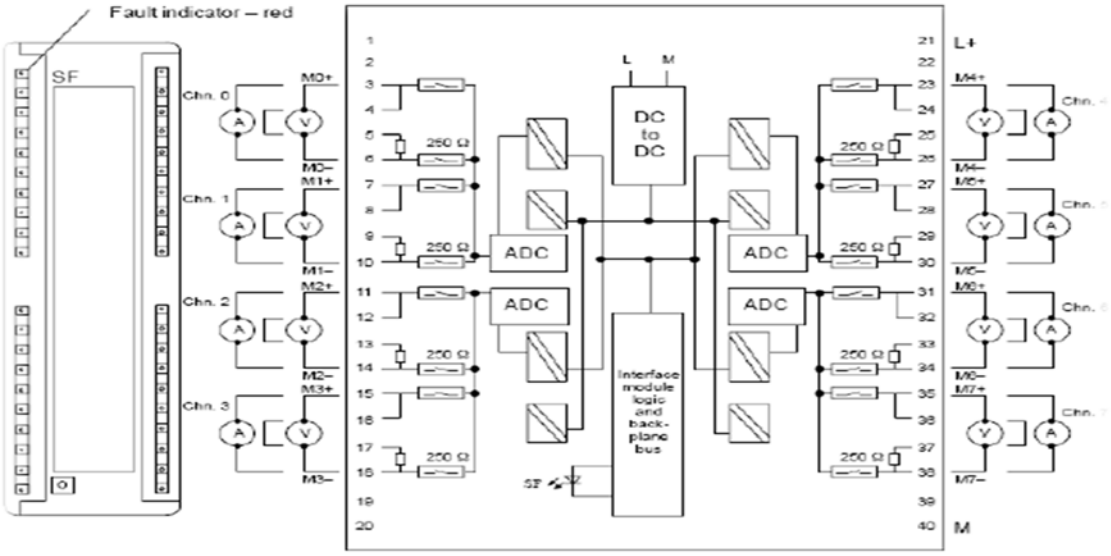
ولتاژ: ±5 V، ±10 V، 1/5 V

جریان: 0/20 mA، 4/20 mA، ±20 mA می باشد.

SM331 AI8*RTD

کارت 8 کاناله 16 بیتی با قابلیت اتصال مستقیم به ترمومتر و با

Order Number: 6ES7 331-7PF00-0AB0 می باشد.



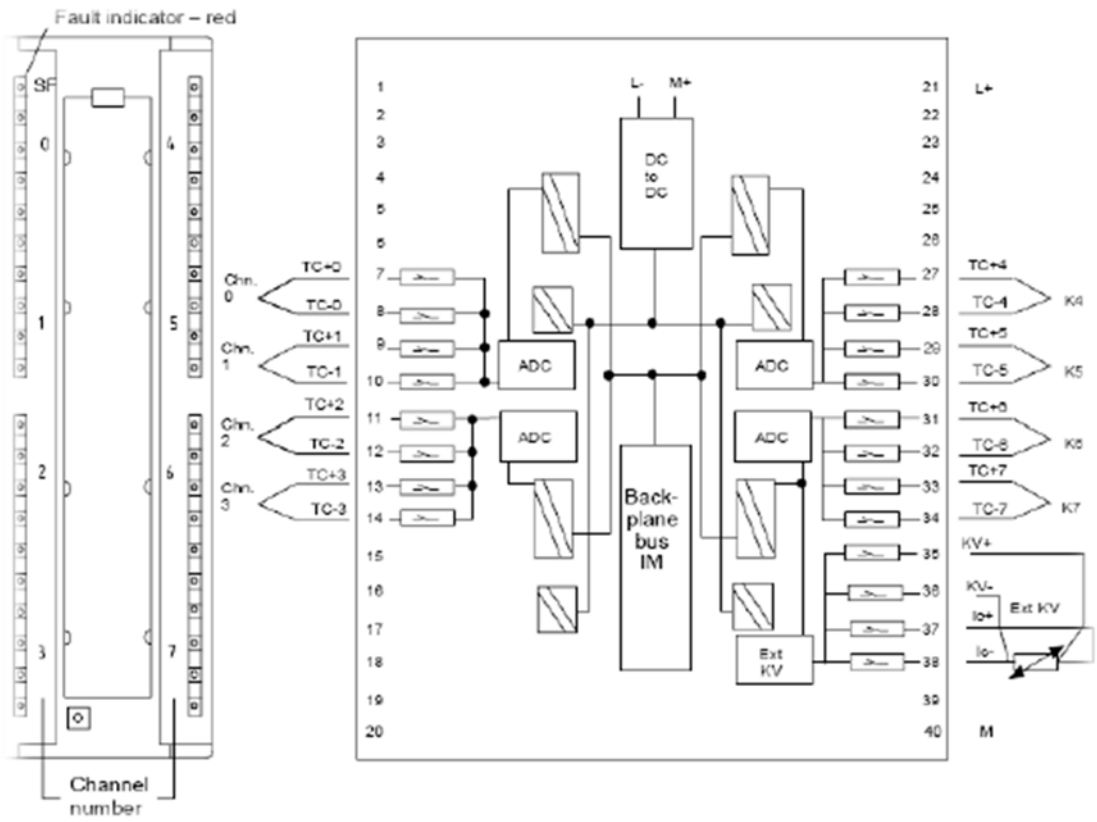
ورودی این کارت در رنج مقاومت: 150 Ohm، 300 Ohm، 600 Ohm و انواع ترمومترهای NI،PT و CU10 در دو رنج استاندارد و cl می باشد.

SM331 AI8*TC

کارت 8 کاناله 16 بیتی با قابلیت اتصال مستقیم به ترموکوپل و با

Order Number: 6ES7 331-7PF10-0AB0 است.

ورودی این کارت ترموکوپل های نوع N ، J ، K ، E ، B ، R ، S ، L ، T ، U ، C می باشد.



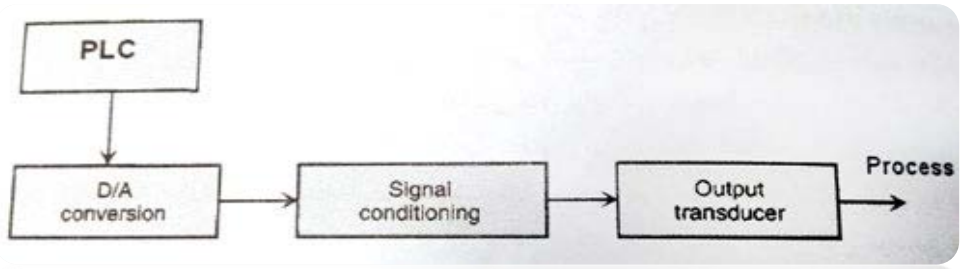
SM331 AI8*TC/4*RTD,Ex

کارت 4 کاناله 15بیتی که هر دو کانال جهت اتصال یک ترمومتر و یا به عنوان یک کارت 8 کاناله برای اتصال ترموکوپل به کار میرود. شماره سفارش این کارت Order Number: 6ES7 331-7SF00-0AB0 می باشد. ورودی این کارت علاوه بر ترمومتر PT100 و PT200 و NI100 در دو رنج استاندارد و cl و همچنین ترموکوپل نوع U،T،L،S،R،B،E،K،J،N و C میتواند ولتاژ در رنج ±25 mV ، ±50 mV ، ±80 mV، ±250 mV، ± 500 mV و ±1 V و نیز مقاومت در بازه 150 Ohm، 300 Ohm، 600 Ohm مورد استفاده قرار گیرد.

کارت خروجی آنالوگ (DO)

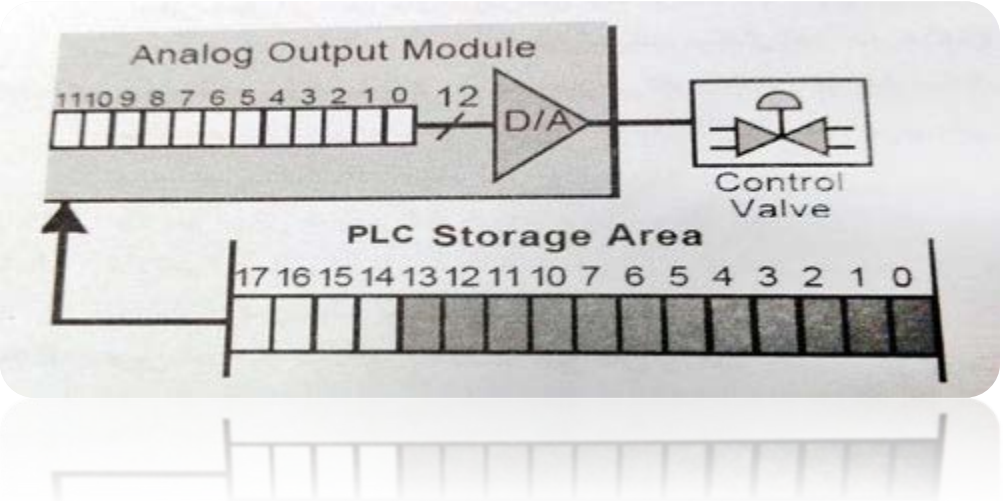
سیگنال های خروجی آنالوگ دارای یک وضعیت پیوسته هستند. در نوع DISCRETE (دیجیتال) مصرف کننده دو وضعیتی است (ON/OFF) ولی در اینجا مصرف کننده نیاز به سیگنال پیوسته دارد. به عنوان مثال یک ولو کنترلی که بین صفر تا 100 درصد باز و بسته میشود ، با یک سیگنال آنالوگ خروجی که به عنوان مثال می تواند بین 4 تا 20 میلی آمپر باشد از سمت PLC کنترل می گردد.

اصول عملکرد کارت آنالوگ خروجی بر عکس کارت آنالوگ ورودی است یعنی PLC فرمان خود را به صورت رشته ای از صفر و یک دیجیتال (دیتا) به کارت ارسال میکند . کارت از طریق یک مبدل دیجیتال به آنالوگ (D/A) دیتا را تبدیل به جریان یا ولتاژکرده و به مصرف کننده ارسال می نماید.



نحوه تبدیل سیگنال آنالوگ خروجی در پی ال سی ها

عملکرد کارت آنالوگ خروجی ساده تر و سریع تر از کارت آنالوگ ورودی است. این کارت به سهولت یک عدد را به ولتاژ یا جریان تبدیل می کند و به زمان بندی و مسائلی مانند نمونه برداری نیاز ندارد.



تبدیل فرمان آنالوگ PLC

کارت آنالوگ خروجی می تواند چندین کانال داشته باشد انواع متداول 2 یا 4 یا 8 کاناله هستند. از آنجا که معمولا یک مبدل D/A در کارت تعبیه شده است، خروجی این مبدل توسط یک Mutiplexer به کانال انتقال می یابد.

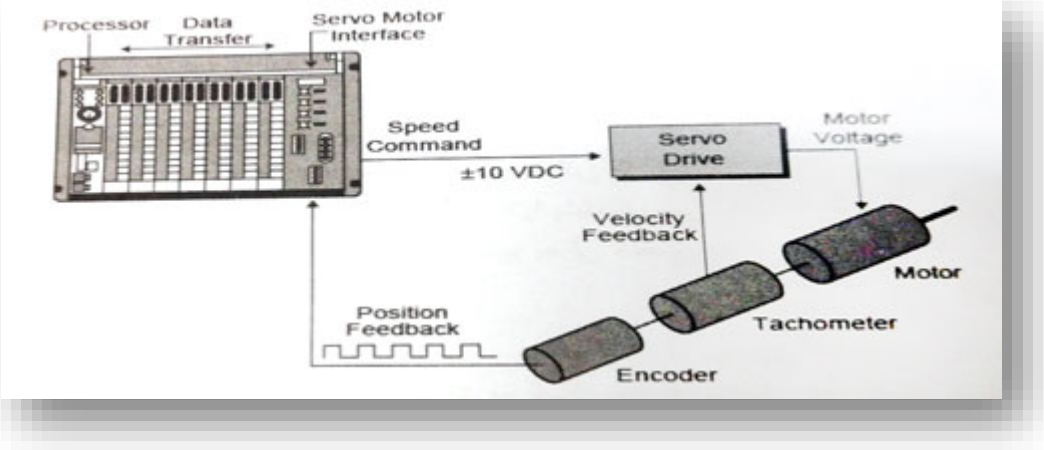
اتصال می تواند به صورت Single Ended یا Differntial به کارت انجام شود.

کانال های کارت آنالوگ نه تنها از یکدیگر ایزوله هستند، بلکه از PLC نیز ایزوله شده اند از اینرو درصورت بروز اضافه ولتاژ درسمت خروجی، کارت حفاظت خواهد شد.

ماژول های مربوط به I/O های خاص

PLC بایستی امکان ارتباط با ورودی و خروجی های خاص که از طریق کارت های معمولی قابل استفاده نیستند را داشته باشد این I/O ها کاربرد خاص دارند و خیلی متداول نیستند. حدود 5 تا 10 درصد I/O ها ممکن است از این نوع باشند. نمونه این کارت ها می توان به مواردی که قابلیت دریافت سیگنال ها یا پالس های سریع مانند پالس های انکودر را دارند اشاره نمود.کارت هایی که برای کنترل لوپ و کنترل موقعیت و کنترل فازی به کار

میروند نیز از این جمله هستند این کارت ها به طور مستقل از CPU می توانند کار پردازش را به طور کامل انجام داده و فرامین لازم را تولید و ارسال کنند.



❖ کارت های خروجی آنالوگ در PLCS7-300 به صورت SM332 نشان داده میشوند.

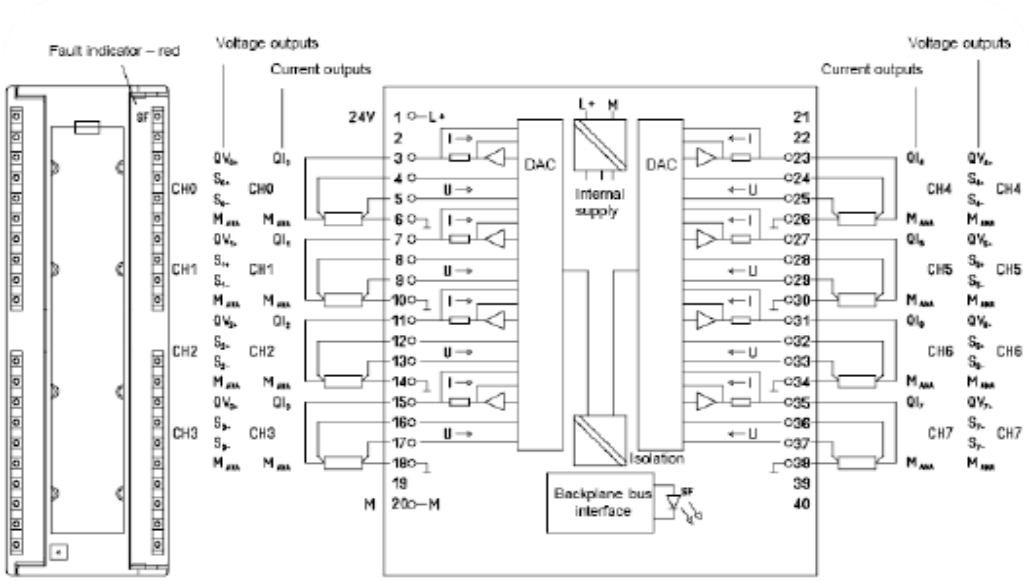
معرفی انواع کارت های خروجی آنالوگ

SM332 AO2*12BIT

این کارت یک کارت 8 کاناله 12 بیتی با Order Number: 6ES7 332-5HB00-0AB0 است که قادر به تولید سیگنال به هر دو جنس ولتاژ و جریان در رنج های زیر در خروجی است:

ولتاژ: 0/10 V،1/5 V و ±10 V

جریان: 0/20 mA، 4/20 mA، ± 20 mA



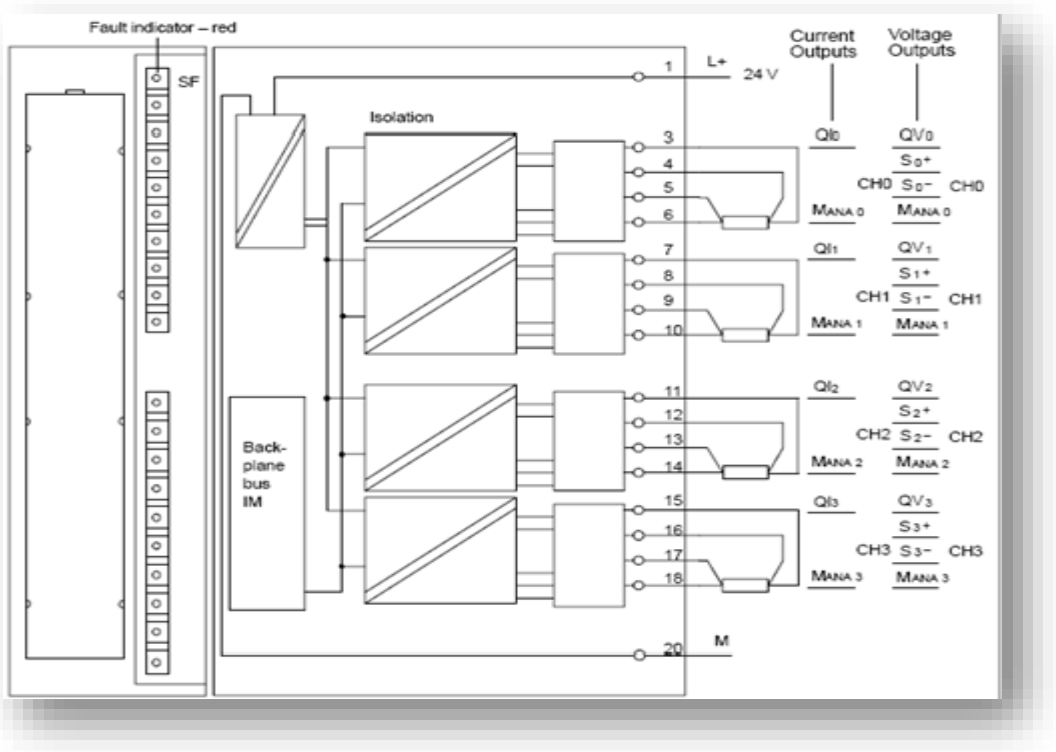
SM332 AO4*0/4 to 20mA,Ex

این کارت دارای 4 کانال و 15 بیتی با Order Number: 6ES7 332-5RD00-0AB0 است.
خروجی این کارت از نوع جریان با رنج 0/20 mA، 4/20 mA می باشد.

SM332 AO4*12BIT

این کارت 4 کاناله 12 بیتی با Order Number: 6ES7 332-5HD00-0AB0 است. خروجی این کانال از جنس ولتاژ با رنج 1/5 V،0/10 V و ±10 V و جریان با رنج های 0/20 mA، 4/20 mA، ± 20 mA می باشد.

SM332 AO4*16BIT



کارت خروجی 4 کاناله 16 بیتی با Order Number: 6ES7 332-7ND00-0AB0 است. خروجی این کارت، ولتاژ با رنج: 1/5 V، 0/10 V و ±10 V و جریان با رنج: 0/20 mA، 4/20 mA، ± 20 mA می باشد.

SM332 AO8*12BIT

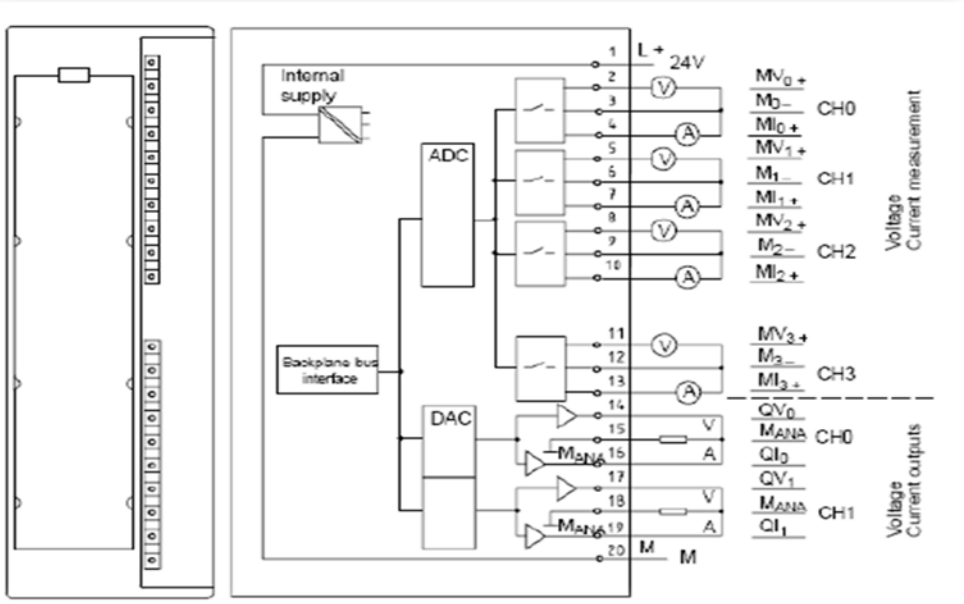
این کارت نیز دارای 8 کانال و 12 بیتی با Order Number: 6ES7 332-5HF00-0AB0 و خروجی های ولتاژ با رنج: 1/5 V، 0/10 V و ±10 V و جریان با رنج: 0/20 mA، 4/20 mA، ± 20 mA می باشد.

- ✓ **نکته**: لازم به ذکر است که نوع دیگری از کارتهای آنالوگ که هم به عنوان ورودی و هم به عنوان خروجی(AI/AO) میتوانند مورد استفاده قرار بگیرند نیز وجود دارد که این دسته از کارت ها را به شکل SM334 و SM335 نمایش میدهند و عبارتند از:

❖ SM334 AI4/AO2

- SM334 AI4/AO2*12BIT
- SM334 AI4/AO2*8/8BIT
- SM335 AI4/AO4*14/12BIT

در شکل زیر یک کارت SM334 نشان داده شده است.



دقت در کارت خروجی آنالوگ

کارتهای خروجی دارای قدرت تفکیک 12 تا 16 بیتی هستند. منظور از دقت، توانایی ایجاد حداقل تغیرات سیگنال است. هر چه دقت ما عدد بزرگ تری باشد یعنی تغیرات ریز تری را میتواند تولید کند و دقیق تر است.

در این میان کارت های 16Bit دارای بیشتر میزان دقت و 12Bit ها دارای کمترین میزان دقت هستند.

مبدل D/A

کارتهای خروجی آنالوگ مبدل های Digital to Analog (DTA) هستند.

بسته به نوع تک قطبی یا دوقطبی بودن سیگنال خروجی، بازه اعداد صحیح [27648 , 0] رادریافت و آن را به سیگنال تک قطبی ولتاژی و جریانی (بازه های بدون مقدار منفی مثلا V[10, 0]) و بازه اعداد صحیح

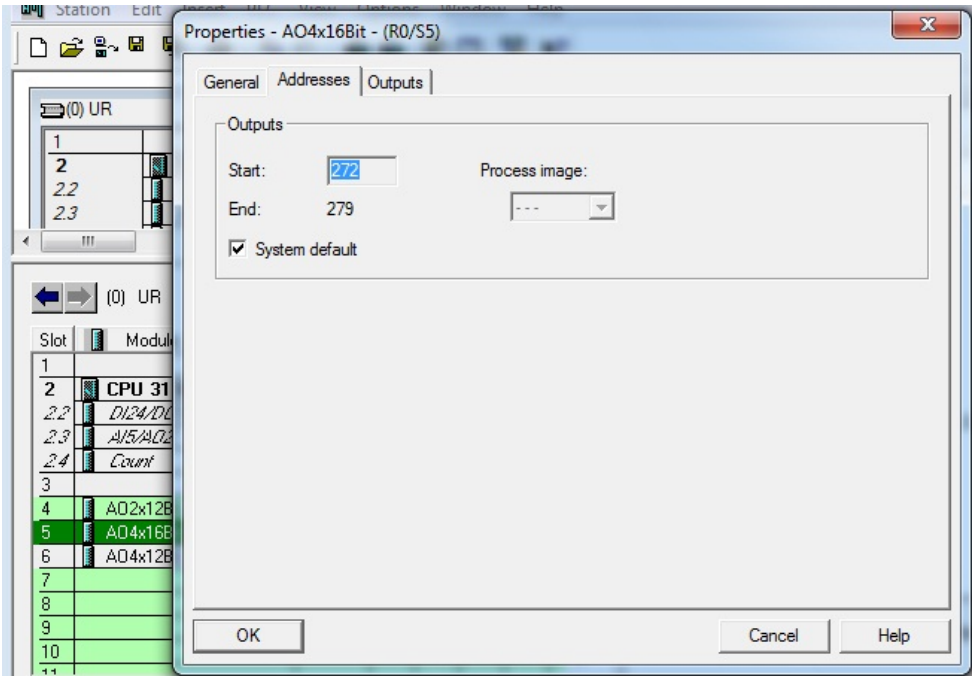
[27648+ , 27648-] را دریافت و به سیگنال دو قطبی ولتاژی و جریانی (شامل بازه ی منفی [10+ , 10-]V) را تولید میکند.

آدرس دهی کارت خروجی آنالوگ:

کارت های خروجی دارای آدرس در فضای P هستند که در اطلاعات کارت، آدرس شروع کانال ها موجود است و این آدرس ها هم در محدوده ی WORD هستند. آدرس دهی کانال های کارت خروجی آنالوگ به صورت زیر است

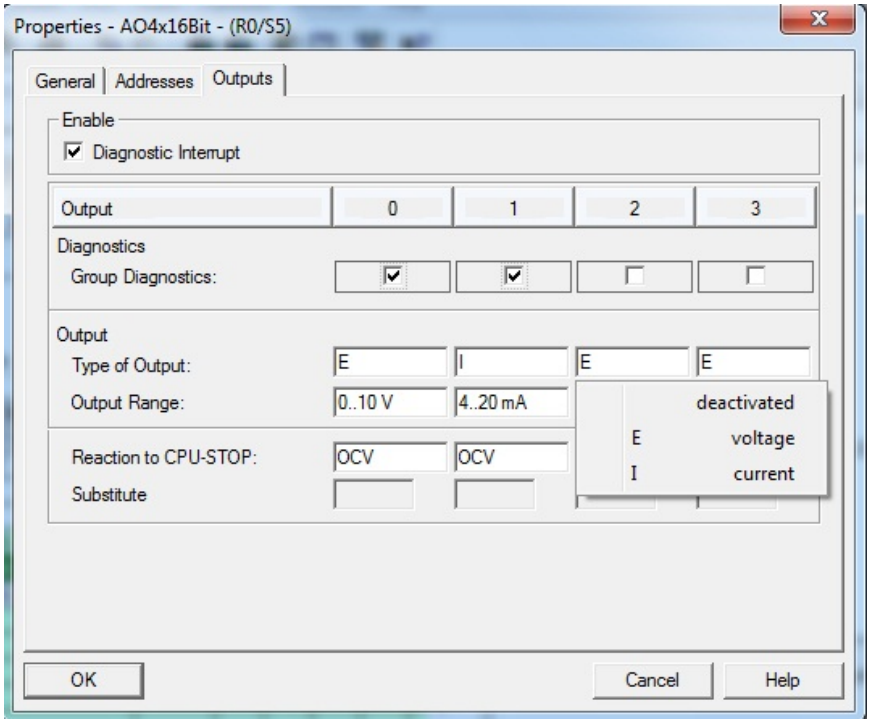
PQWXXX

که XXX شماره بایت شروع کانال میباشد.



در قسمت Output مانند کارت های ورودی آنالوگ Diagnostic Interrupt داریم که با فعال کردن آن و انتخاب هر کانال، وقفه خطای آن کانال را فعال میکنیم.

از قسمت Type of Output و Output Range میتوان نوع ولتاژی یا جریانی سیگنال خروجی و رنج آن را تنظیم نمود.

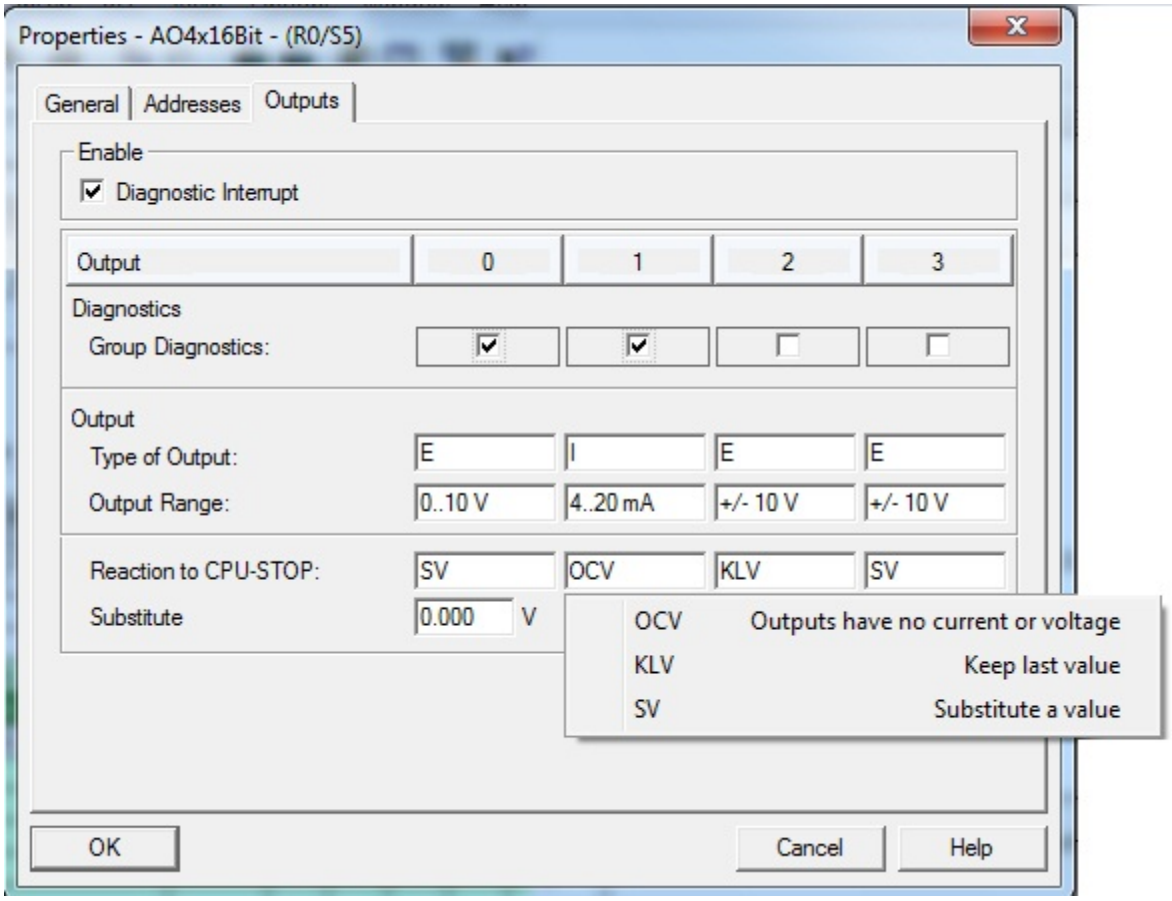


در قسمت Reaction to CPU-STOP عملکرد کارت خروجی هنگامی که CPU به مد STOP میرود را مشخص میکنیم.

OCV : خروجی جریان یا ولتاژ قطع میشود.

KLV : آخرین مقدار خروجی ثبت شده.

SV : مقداری که مشخص میکنیم اعمال میشود که با فعال کردن این گزینه میتوان در Substitute مقدار مورد نظر را وارد کرد.



معادله خط در کارت خروجی آنالوگ

فرض کنید از طریق یک اسلایدر در HMI سرعت موتور متصل به درایو را کم و زیاد کنیم و اسلایدر بازه حقیقی بین صفر تا 50 هرتز است. معادله خطی میخواهیم که بازه Hz [50 , 0] را به بازه [27648 , 0] تک قطبی (سیگنال های در رنج مثبت) و [27648+ , 27648-] دوقطبی (سیگنال شامل رنج منفی و مثبت) تبدیل کند.

بعد از تبدیل باید این مقدار به آدرس کانال کارت خروجی فرستاده شود تا مبدل ATD سیگنال آنالوگ را تولید کند.

این کار توسط بلوک سیستمی FC106 انجام میشود.

OUT= [((IN – LO_LIM) / (HI_LIM – LO_LIM)) * (K2–K1)] + K1

OUT : خروجی با آدرس کانال کارت خروجی

IN : اسلایدر کنترلی در بازه REAL

K2 : 27648

K1 : 0 (تک قطبی) یا 27648 - (دو قطبی)

LO_LIM : مینیمم مقدار اسلایدر کنترلی

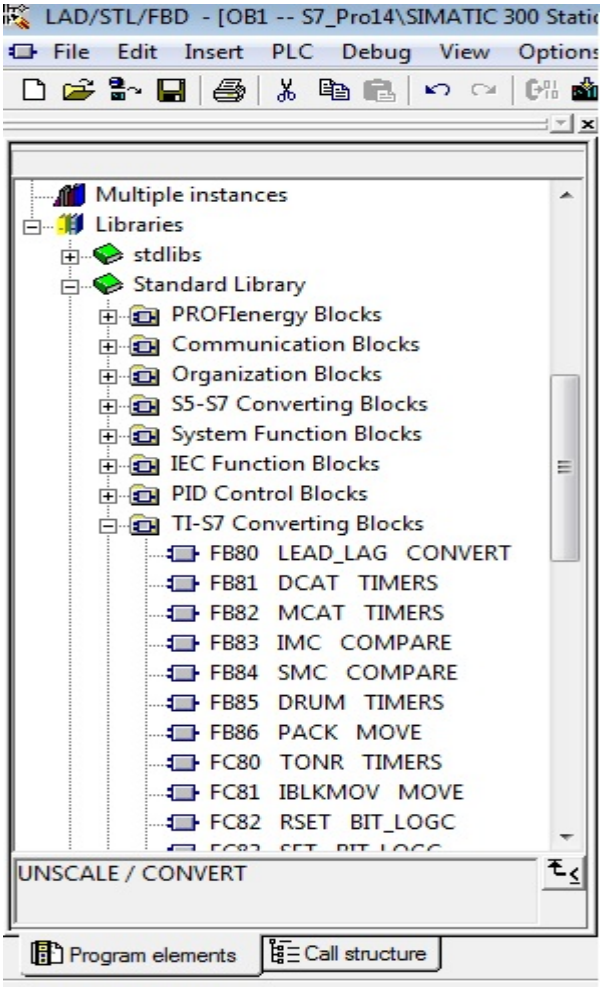
HI_LIM : ماکزیموم مقدار اسلایدر کنترلی

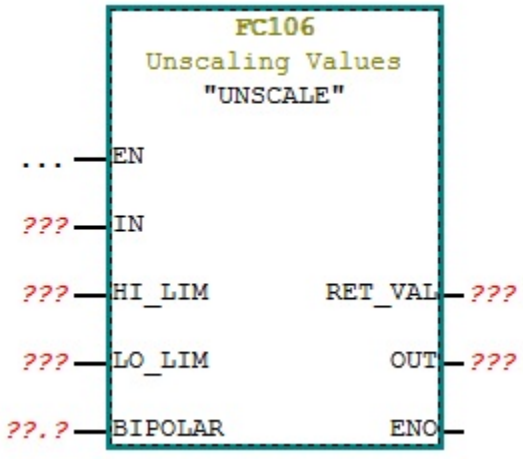
فراخوانی بلوک سیستمی FC106

برای تبدیل مقدار حقیقی اسلایدر کنترلی توسط HMI یا پتاسیومتر یا ... به بازه های دیجیتالی ، مبدل DTA برای تولید سیگنال الکتریکی استاندارد ولتاژ و جریان ، از این بلوک سیستمی که قابلیت فراخوانی دارد استفاده میکنیم.

وارد محیط برنامه نویسی میشویم، در قسمت نوار ابزار سمت چپ برنامه و در مسیر زیر میتوان یک FC106 را درون نت ورک اضافه کرد:

Libraries / Standard Libraries / TI-S7 Converting Blocks / FC106 UNSCALE CONVERT


LAD/STL/FBD - [OB1 -- S7_Pro14\SIMATIC 300 Stati
File Edit Insert PLC Debug View Options
Multiple instances
Libraries
stdlibs
Standard Library
PROFIenergy Blocks
Communication Blocks
Organization Blocks
S5-S7 Converting Blocks
System Function Blocks
IEC Function Blocks
PID Control Blocks
TI-S7 Converting Blocks
FB80 LEAD_LAG CONVERT
FB81 DCAT TIMERS
FB82 MCAT TIMERS
FB83 IMC COMPARE
FB84 SMC COMPARE
FB85 DRUM TIMERS
FB86 PACK MOVE
FC80 TONR TIMERS
FC81 IBLKMOV MOVE
FC82 RSET BIT_LOGC
UNSCALE / CONVERT
Program elements
Call structure


FC106
Unscaling Values
"UNSCALE"
EN
IN
HI_LIM
LO_LIM
BIPOLAR
RET_VAL
OUT
ENO

بلوک FC106 به صورت یک تابع نوشته شده است. توجه شود که این بلوک یک FC به Block های موجود اضافه میکند که باید به همراه OB1 در سیمولیشن دانلود شود.

EN : برای شرطی کردن اجرای تابع FC106

IN : ورودی کنترلی تابع که یک مقدار حقیقی در بازه ی مشخص شده است (مثلا فرکانس در بازه صفر تا 50Hz)

HI_LIM : بالاترین حد از ورودی کنترلی حقیقی (مثلاً 50.0 هرتز)

LO_LIM : پایینترین حد از ورودی کنترلی حقیقی (مثلاً 0.0 هرتز)

BIPOLAR : مشخص کننده تک قطبی یا دوقطبی بودن سیگنال الکتریکی خروجی. صفر منطقی برای تک قطبی ها و مقدار یک منطقی برای دوقطبی ها

RET_VAL : خروجی کد خطا های ممکن که باید در یک WORD از حافظه ذخیره میشود.

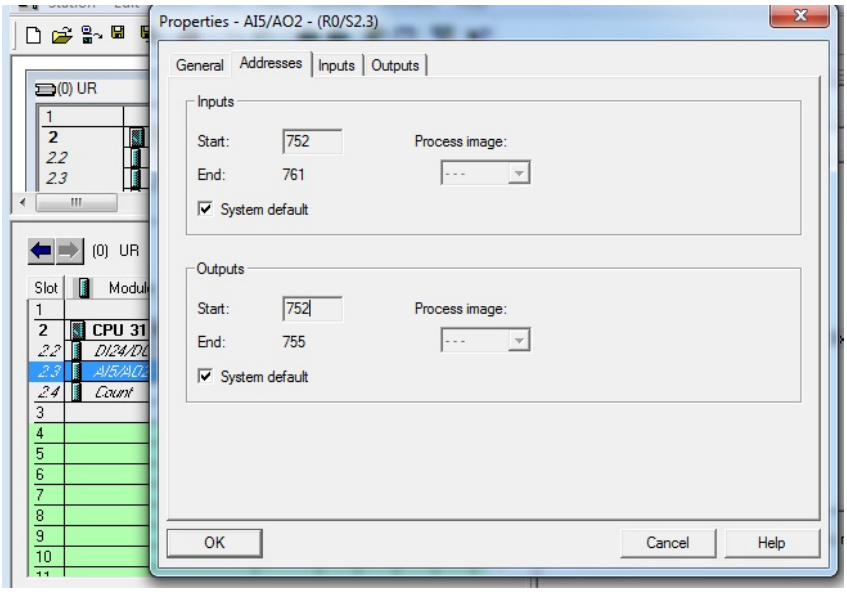
OUT : خروجی که باید آدرس کانال کارت خروجی با فرمت PQWXXX وارد شود. مقدار صحیح در بازه [0 , 27648] برای خروجی تک قطبی و یا مقدار صحیح [+27648 , -27648] برای دوقبی را شامل میشود.

ENO : خروجی تابع هنگامی که تابع در صورت برقراری شرط اجرا مقدار یک منطقی میدهد.

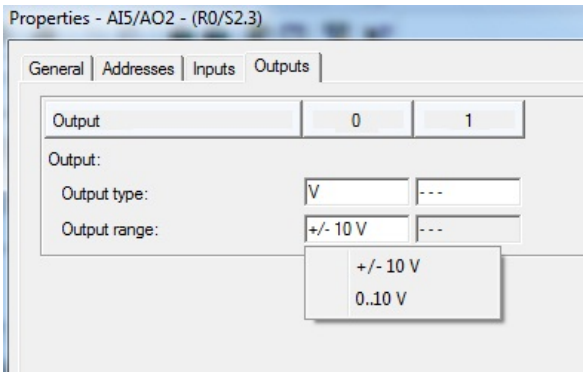
مثال: میخواهیم با یک اسلایدر در HMI (MD3)، فرکانس موتور متصل به درایو با ورودی آنالوگ ولتاژی در رنج [+10 , -10]V را در محدوده [0 , 50]Hz کنترل کنیم.

CPU 313C را در نظر بگیرید. هرگاه فرکانس بزرگ تر از 30Hz شد چراغ Q0.0 روشن شود.

مرحله اول انتخاب CPU مورد نظر و مشاهده آدرس کارت آنالوگ این CPU Compact است.

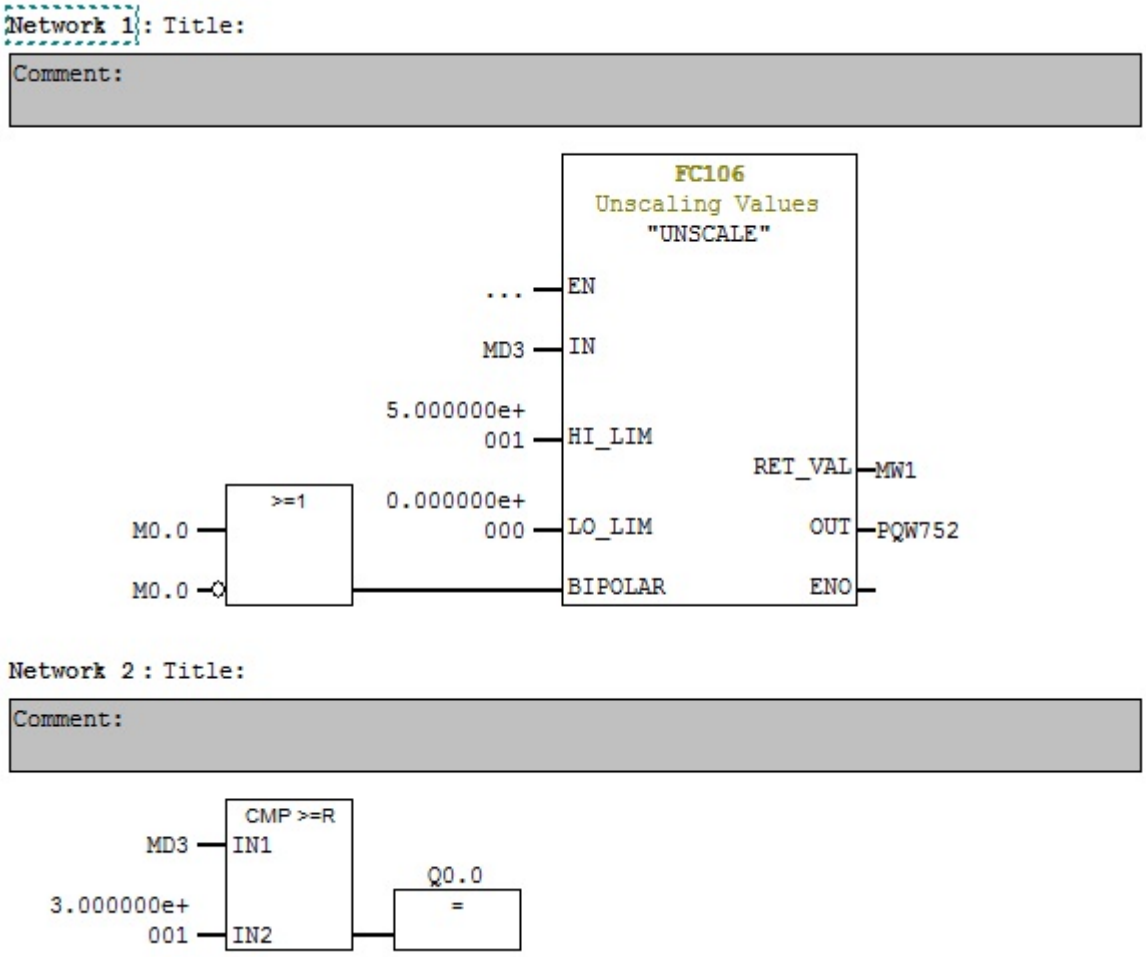


مرحله بعدی تعیین تایپ و رنج کارت خروجی آنالوگ است.

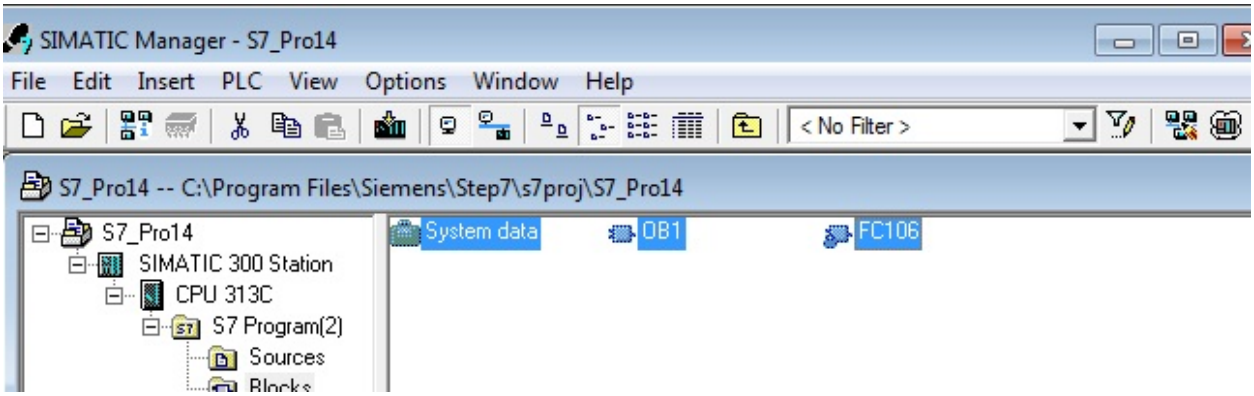


محیط HW Config را Save& Compile میکنیم.

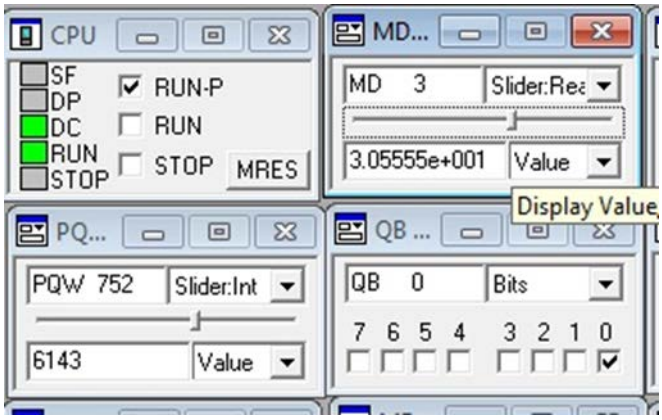
در OB1 تابع FC106 را فراخوانی میکنیم و مقادیر مربوطه را میدهیم.



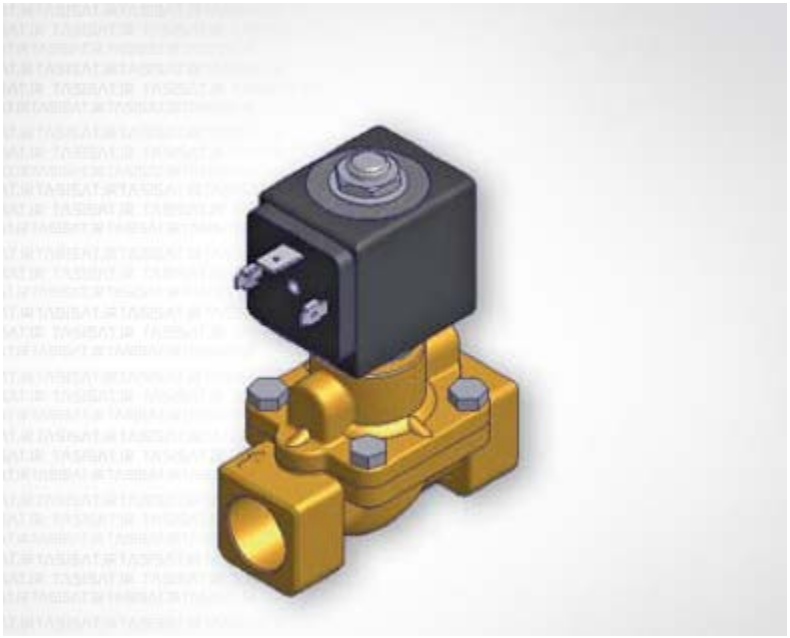
OB1 را ذخیره کرده و سیمولیشن را باز میکنیم و بلوک های موجود را به همراه System Data روی سیمولیشن دانلود میکنیم.



با تغییر اسلایدر MD3 مشاهده میشود مقدار PQW752 از 27648- تا 27648+ تغیر میکند.



نمونه از شیرهای برقی آنالوگ



بلوک های سازمانی

OBها

انواع بلوک های سازمانی یا OBها

به طور کلی OBها به سه دسته اصلی تقسیم میشوند:

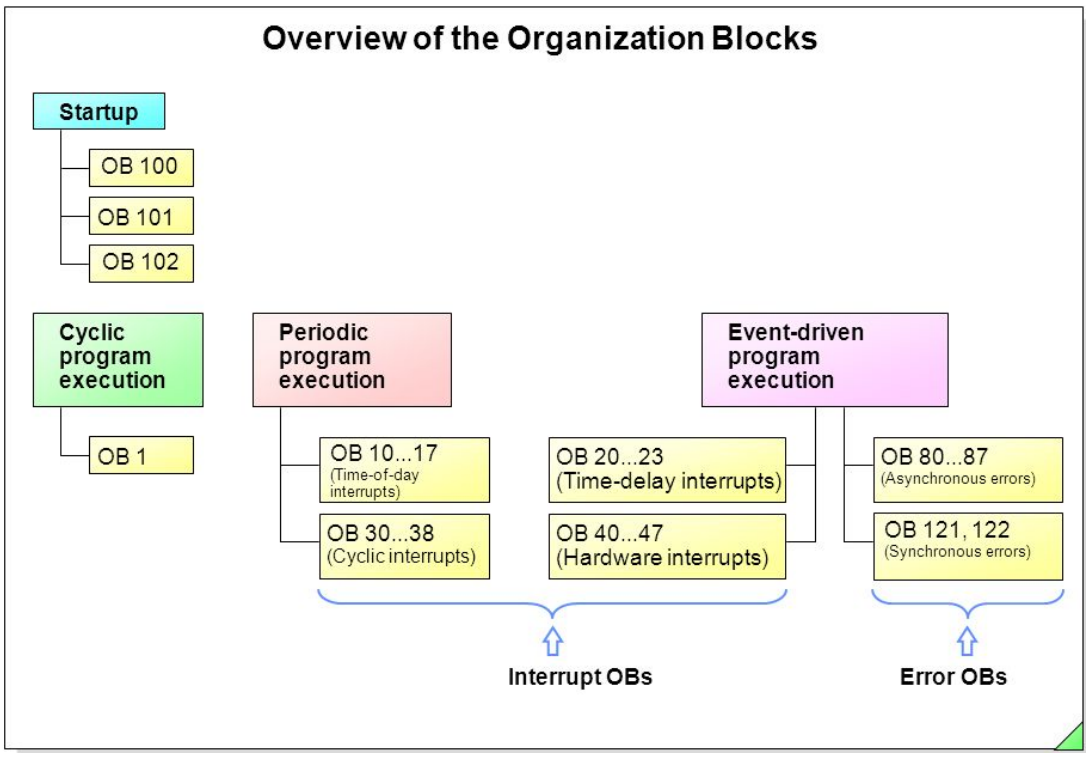
OB اجرای سیکل برنامه

OB های راه اندازی

OB های وقفه

هنگامی که PLC شروع به کار میکند بلوک های سازمانی راه اندازی (STARTUP OB) یک بار اجرا میشوند و پس از آن وارد بلوک سازمانی اجرای برنامه (CYCLIC PROGRAM EXECUTION - OB1) میشود. تا زمانی که وقفه ای پیش نیاید CPU به اجرای سیکلی این بلوک می پردازد و هنگام روی دادن وقفه، CPU پردازش برنامه ی OB1 را متوقف می کند و به بلاک های سازمانی مربوط به وقفه (INTERRUPT OB) میرود و برنامه نوشته شده داخل آن بلوک را اجرا میکند. بسته به نوع وقفه و نوع برنامه آن بعد از اجرای بلوک وقفه در صورتی که فرآیند وقفه پایان یافته باشد CPU به بلوک OB1 برمیگردد و اجرای برنامه را ادامه میدهد.

نمای کلی از بلوک های سازمانی (OBها)



ترتیب الویت فراخوانی

اجرای برنامه یک OB می تواند توسط اتفاق با الویت بالاتر در مرزهای مشخص شده ای متوقف و برنامه OB با تقدم بالاتر اجرا شود. سپس دنباله برنامه باقیمانده از OB متوقف شده، پس از پایان اجرای برنامه OB با تقدم بالاتر اجرا خواهد شد.

ترتیب اولویت OBها از 1 تا 28 رتبه بندی شده است که در این بین الویت 1 کم ارزش ترین و الویت 28 پر ارزش ترین الویت محسوب میشود. یعنی اجرای برنامه با الویت بالاتر تقدم دارد.

الویت اجرای برخی از OBها

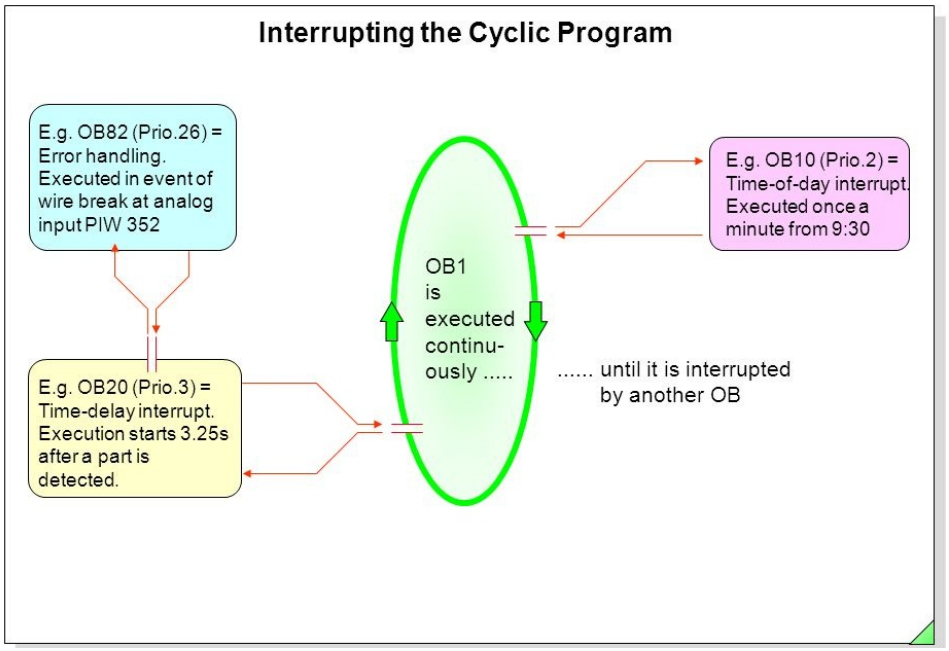
OB No.	OB Type	Priority
OB 1	Cyclic program	1
OB 10	Time-of-day interrupt	2
OB 20	Time-delay interrupt	3
OB 35	Cyclic interrupt	12
OB 40	Hardware interrupt	16
OB 82	Error handling	26 / 28

نکته قابل توجه در مورد PLC های سری 400 این است که برخی از OBها دارای الویت اجرای برابر اند. در این صورت این OB ها یک دیگر را متوقف نمیکنند بلکه بعد از پایان اجرای یکی، دیگری بر اساس تشخیص OPERATION SYSTEM اجرا میشوند.

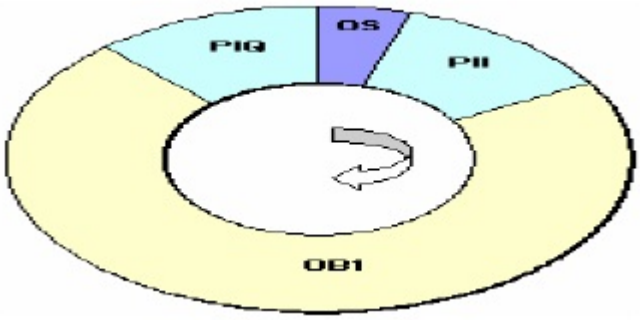
OB اجرای سیکل برنامه (CYCLIC PROGRAM EXECUTION OB)

این بلوک که تنها شامل OB1 میشود که محل نوشتن برنامه اصلی است. درجه اولویت این بلوک 1 یعنی دارای کم ارزش ترین درجه الویت اجرا میباشد. اجرای برنامه در این بلوک به صورت سیکلی است و اجرای آن تا ورود وقفه ادامه میابد.

لازم به ذکر است که بعد از آمدن وقفه و اجرای آن در صورتی که وقفه پایان یابد CPU مجددا وارد اجرای برنامه این بلوک میشود و در غیر این صورت CPUبه مد استپ میرود.



اجرای سیکلی برنامه به این صورت است که ابتدا تصویری از همه ی ورودی های برنامه گرفته میشود (PII) و بر اساس آن برنامه اجرا میشود و پس از پایان برنامه تصویری از خروجی ایجاد میشود (PIQ) و به سیستم عامل برای اعمال آن به خروجی ها فرستاده میشود و این روند ادامه دار است.



OBهای راه اندازی (START UP OB)

با روشن شدن PLC یا تغیر مد CPU از STOP به مد START، قبل از اجرای برنامه OB1، CPU یک بار برنامه مربوط به راه اندازی را در صورت وجود اجرا میکند. الویت اجرای این OB ها 27 است که بالا ترین الویت است.

در بسیار از برنامه ها نیاز است که مقادیری به عنوان مقادیر اولیه در حافضه یا خروجی و ... قرار گیرد. این مقدار دهی ها و یا سایر تغیراتی که لازم است قبل از اجرای برنامه اصلی ایجاد شود را میتوان در بلوک های مربوط به راه اندازی برنامه ریزی کرد.

نکته: در صورتی که CPU در مد RUN باشد و با قطع برق مواجه شود، بعد از وصل برق نیز بلوک های راه اندازی اجرا میشوند.

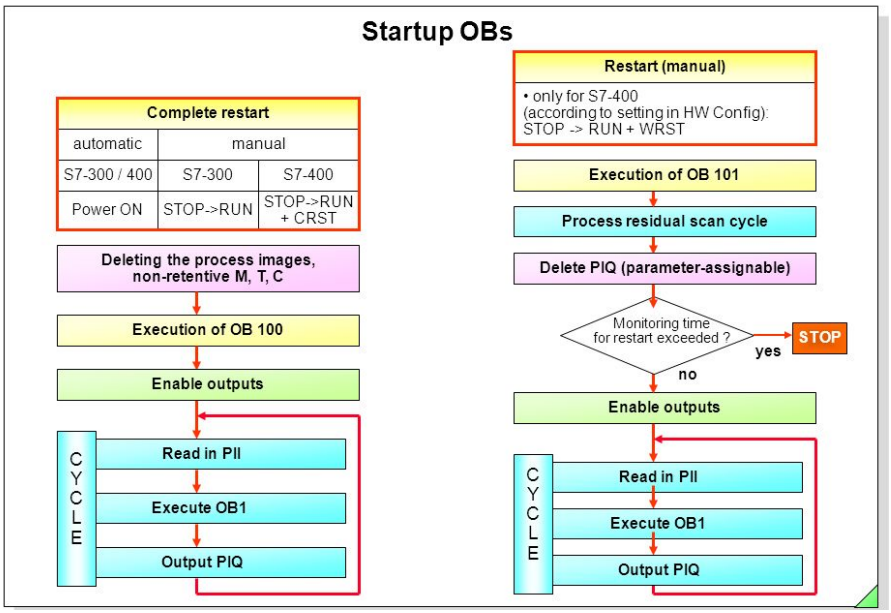
انواع بلوک های راه اندازی

1- OB100 (برای راه اندازی گرم یا WARM RESTART)
2- OB101 (برای راه اندازی داغ یا HOT RESTART)
3- OB102 (برای راه اندازی سرد یا COLD RESTART)

برای PLC های سری 300 راه اندازی فقط به صورت WARM انجام میشود ولی در سری 400 راه اندازی میتواند به هر سه روش انجام شود.

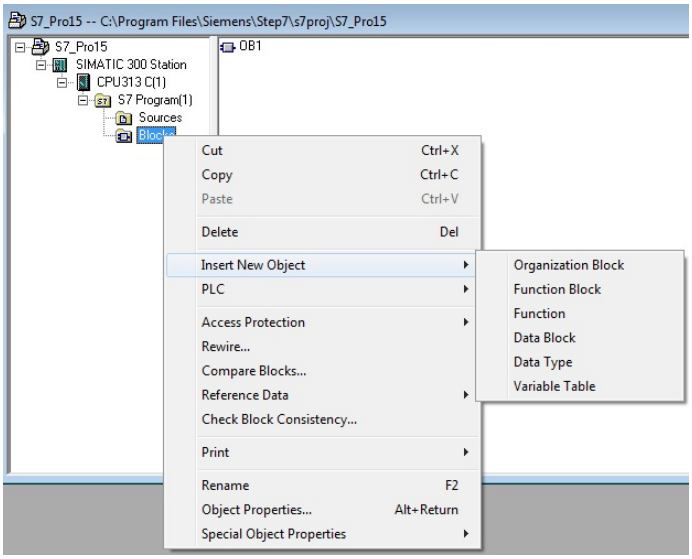
تذکر: CPU 318-2 تنها CPU موجود در سری 300 است که علاوه بر راه اندازی WARM میتواند به صورت COLD هم راه اندازی شود.

نحوه عملکرد CPU هنگام شروع بکار



مثال: میخاهیم هرگاه CPU راه اندازی میشود تمامی مقادیر MD0 برابر با صفر شود.

قدم اول ساخت بلوک OB100 است.برای این کار کافی است در قسمت BLOCK در زیر مجموعه CPU انتخابی در پروژه کلیک راست کرده و در قسمت INSER OBJECT یک ORGANIZATION BLOCK ایجاد کنیم، و شماره OB مورد نظر که اینجا OB100 است را تایپ کنیم. بدیهی است که ساخت سایر OBها که در ادامه به آن میپردازیم به همین صورت است.



بعد از ساخت OB100 در آن برنامه مورد نظر را مینوسیم

```
OB100 :  "Complete Restart"
Comment:

Network 1: Title:
Comment:

      L     0
      T     MD    0
      BE
```

توجه داشته باشید که اگر برنامه فوق را درون OB1 بنویسیم در اجرای هر سیکل از برنامه مقادیر حافظه MD0 صفر میشود ولی در اینجا این مقدار تنها در زمان شروع برنامه صفر میشود و دیگر در روند اجرای برنامه وارد نمیشود تا زمانی که CPU به مد استپ برود و دوباره RUN شود.

OBهای وقفه (INTERRUPT OB'S)

وقفه ها به دو دسته ی کلی تقسیم میشوند

1- وقفه مبتنی بر خطا (ERROR) : وابسته به وقوع خطا نرم افزاری یا اشکال سخت افزاری است، و برای تشخیص خطا استفاده میشود. این وقفه ها شامل بلوک های OB 80 تا OB 87 و OB121, OB122 میباشد.

2- وقفه های مبتنی بر رویداد (EVENT) : وابسته به رویداد تنظیم شده از قبل است و پای خطا در میان نیست. این وقفه ها شامل بلوک های OB10 تا OB17 - OB20 تا OB23 - OB30 تا OB38 – OB40 تا OB47 میباشد.

در صورت بوجود امدن وقفه از نوع رویداد CPU اجرای OB1 را متوقف میکند و برنامه OB مربوط به آن وقفه را یک بار اجرا میکند و به OB1باز میگردد.

در صورت بوجود امدن خطا در برنامه یا اشکال در سخت افزار، CPU از OB1 خارج و به OB مربوط به خطا میرود و چراغ SF روشن میشود، در صورتی که خطا در برنامه OBمربوط رفع شود CPU به کار خود ادامه میدهد و در غیر اسن صورت به مد STOP میرود.

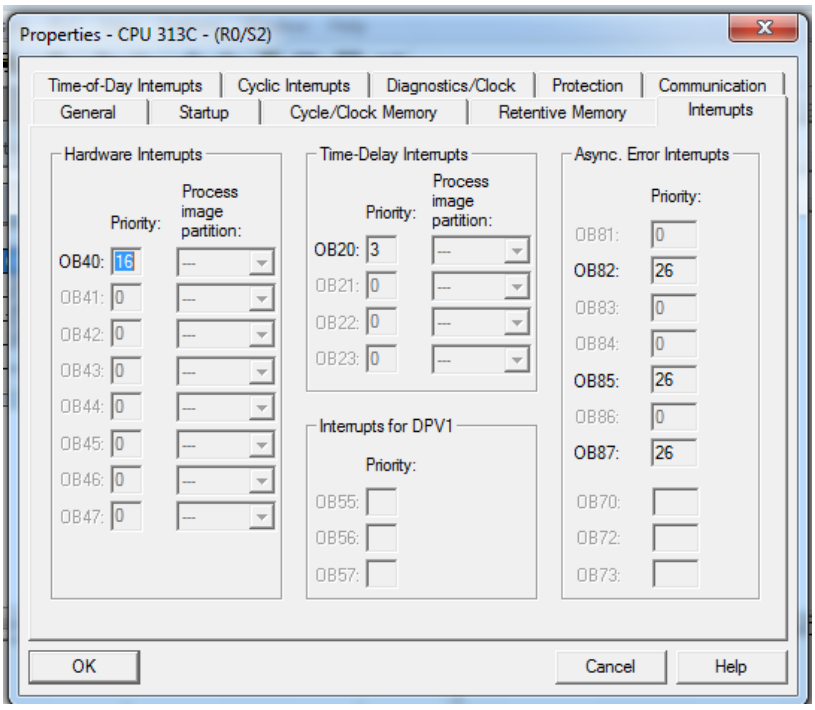
انواع وقفه های مبتنی بر ERROR و درجه الویت

Organization block		Priority in S7-300
Function	Number	
Asynchronous error	OB 80 ... 87	26 , 28
Synchronous error	OB 121, 122	Priority of the OB causing the erro

انواع وقفه های مبتنی بر EVENT و درجه الویت

Organization block		Priority in S7-300
Function	Number	
Time-of-day interrupt	OB 10 ... 17	2
Cyclic interrupt	OB 30 ... 38	12
Time-delay interrupt	OB 20 ... 23	3
Hardware interrupt	OB 40 ... 47	16
Diagnostic interrupt	OB 81 ... 87	26

بسته به نوع CPU انتخابی امکان دارد برخی از OBها برای استفاده اکتیو نباشند. برای مشاهد بلوک های قابل استفاده در CPU مورد نظر، به تنظیمات CPU در زیربرنامه Hwconfig مراجعه می کنیم. در تب های interrupt و time of day interrupt و cyclic interrupt انواع OBها موجود است.



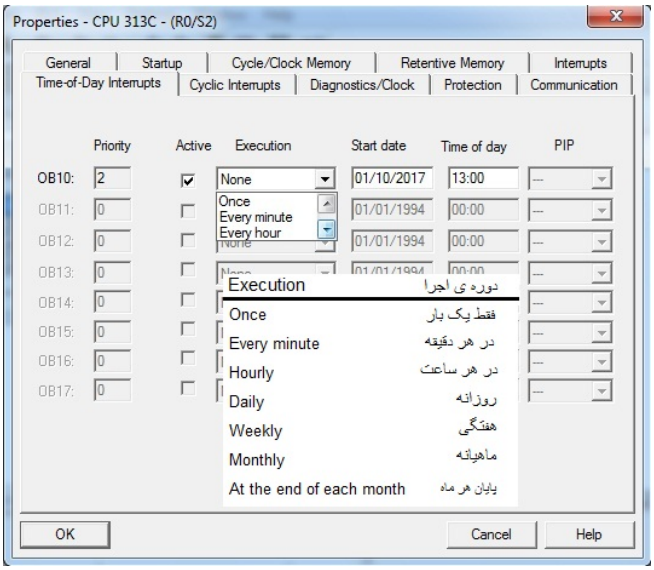
در ادامه به معرفی انواع وقفه ها می پردازیم.

وقفه های زمان تاریخ (TIME OF DAY INTERRUPT)

برای اعمال وقفه در تاریخ و زمان مشخص بکار میرود. این وقفه ها شامل OB10 تا OB17 میشود که بسته به CPU انتخابی تعداد OBهای فعال آن متفاوت است. هر OB اجرای یک برنامه وقفه مورد استفاده قرار میگیرد.

برای استفاده از این OBها در زیربرنامه HWconfig و تنظیمات CPU (تب Time-of-Day)، باید OB مربوطه را فعال (Active) و دوره ی اجرا و همچنین تاریخ و زمان اجرای آن را تنظیم نمود.

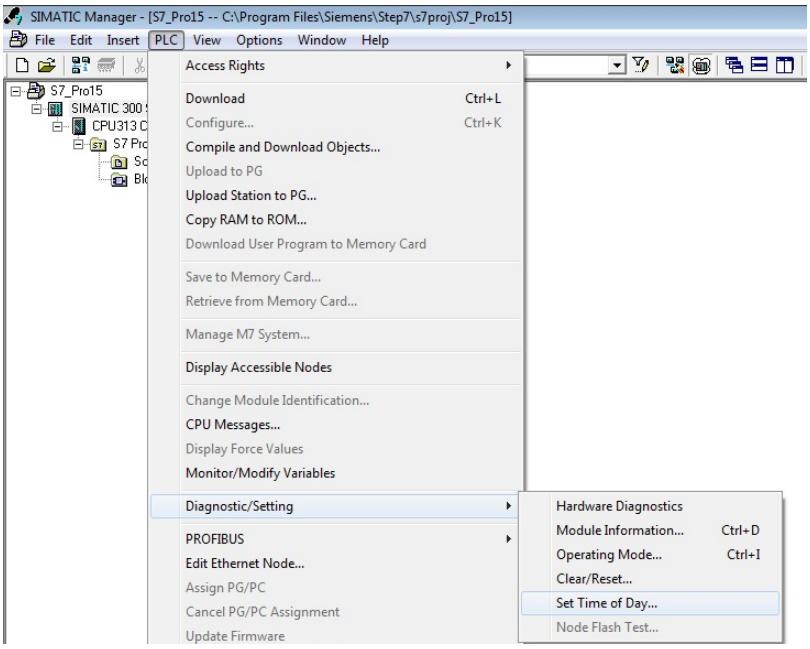
دوره اجرا برای مشخص کردن تکرار یا عدم تکرار اجرای برنامه در این بلاک ها است.



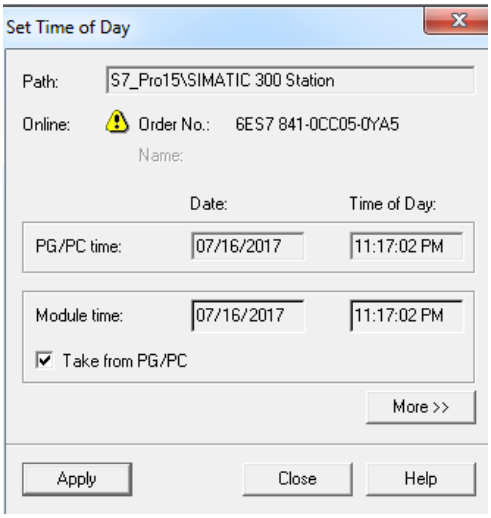
همانگونه که مشاهده میشود CPU313C فقط یک بلوک فعال برای وقفه ی TIME OF DAY دارد و الویت اجرای آن 2 است.

بعد از این مرحله نوبت به ساخت OB10 در BLOKs در برنامه SIMATIC Manager است و برنامه مربوط به وقفه TIME OF DAY در آن مینوسیم.

نکته قابل توجه تنظیم ساعت و تاریخ PLC است. برای تنظیم ساعت و تاریخ باید به PLC متصل بود یا سیمولیشن فعال باشد در مسیر زیر میرویم.



در پنجره ی باز شده میتوان تاریخ و زمان را با سیستم متصل به PLC هماهنگ کرد.



- نکته قابل توجه این است که اگر تاریخ اجرا قبل از راه اندازی PLC باشد هرگز به بلوک وقفه وارد نمیشود.

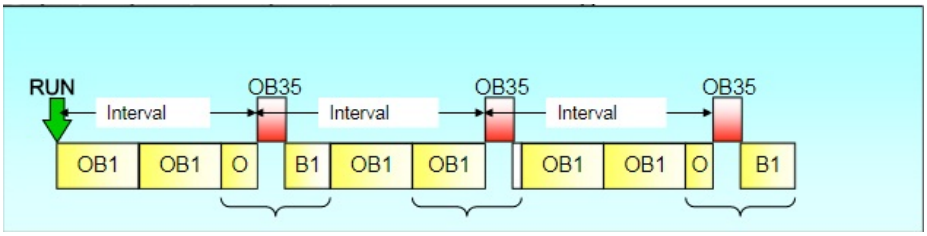
- نکته: اگر OB وقفه در تنظیمات CPU فعال باشد ولی خود آن موجود نباشد یا دانلود نشده باشد SF روشن و CPU به مد STOP میرود.
- برای تست در سیمولیشن باید حتما تنظیمات انجام شده در HWconfig را SAVE & COMPILE کنیم و برای دانلود، SYSTEM DATA به همراه بلوک ها باید دانلود در سیمولیشن دانلود شود.

وقفه های سیکلی (CYCLIC INTERRUPT)

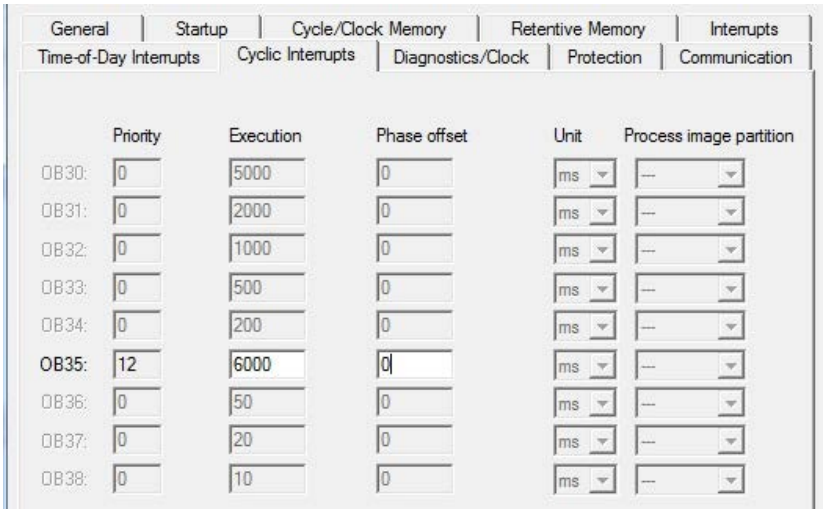
این وقفه ها به صورت سیکلی و در دوره های زمانی مشخص و مداوم اجرا میشوند. یکی از کاربرد های مفید این بلوک ها در کنترل کننده های PID و نمونه برداری از سنور های آنالوگ است. مزیت این روش این است که سرعت پردازش CPU را افزایش میدهد. این وقفه ها شامل OB30 تا OB38 میشود که بسته به CPU انتخابی تعداد OBهای فعال آن متفاوت است. درجه الویت هر یک را مشاهده میکنید.

OB	Start Event	Default Priority Class
OB30	Cyclic interrupt 0 (default interval: 5 s)	7
OB31	Cyclic interrupt 1 (default interval: 2 s)	8
OB32	Cyclic interrupt 2 (default interval: 1 s)	9
OB33	Cyclic interrupt 3 (default interval: 500 ms)	10
OB34	Cyclic interrupt 4 (default interval: 200 ms)	11
OB35	Cyclic interrupt 5 (default interval: 100 ms)	12
OB36	Cyclic interrupt 6 (default interval: 50 ms)	13
OB37	Cyclic interrupt 7 (default interval: 20 ms)	14
OB38	Cyclic interrupt 8 (default interval: 10 ms)	15

نحوه ی اجرا این OB ها نمایش داده شده است.



برای استفاده از این OBها در زیربرنامه HWconfig و تنظیمات CPU (Cyclic Interrupt)، میتوان OB فعال جهت استفاده را شناسایی کرد و فاصله ی زمانی اجرای این OBها در قسمت Execution با واحد میلی ثانیه وارد کرد و همچنین شماره بلوک فعال دلخواه را در قسمت BLOKs در برنامه SIMATIC Manager ایجاد و برنامه نویس کرد.



در اینجا برای وقفه سیکلی فقط میتوان از OB35 استفاده کرد.

- حداکثر فاصله ی زمانی برای اجرای وقفه سیکلی 60000Ms است.
- این وقفه برخلاف وقفه تاریخ زمانی نیازی به فعال کردن ندارن و در صورت وجود بلوک مربوطه با شروع به کار cpu اجرا میشوند.
- درجه اولویت این بلوک ها از وقفه تاریخ زمانی بیشتر است
- فاصله ی زمانی اجرای این وقفه باید از زمان لازم برای اجرای آن بیشتر باشد، در غیر این صورت قبل از پایان یافتن اجرای این OB بار دیگر توسط سیستم عامل فراخانی میشود و در این صورت خطای TIME ERROR اتفاق می افتد و SF روشن میشود و CPU به مد STOP میرود.
- در صورتی که بیش از یک OB وقفه سیکلی داشته باشیم ممکن است در هنگام اجرا تداخل پیش بیاید و ممکن است این تداخل ها زمان سیکل CPUرا از حد مجاز بالاتر ببرد و خطا ایجاد کند. باید توجه کرد که این تداخل ایجاد نشود، برای ایجاد نشدن تداخل میتوان مقدار Phase offset به این OB ها

اختصاص داد. باید توجه داشت که مقدار Phase offset از زمان لازم برای اجرای OB وقفه دیگر بیشتر باشد.

- با استفاده از بلوک های سیستمی SFC39 و SFC43 میتوان وقفه سیکلی را متوقف کرد یا به آنها تاخیر داد یا مجددا فعال نمود.

وقفه های سخت افزاری (Hardware Interrupt)

این وقفه ها از سیگنالی که سخت افزاری میفرستد بوجود میایند و بر اساس ERROR نیست. در واقع سخت افزار با قابلیت ایجاد Interrupt میباشد. مثلا تغیر وضعیت یک ورودی در کارت SM321 DI16xDC24V, Interrupt میتواند وقفه ایجاد کند.

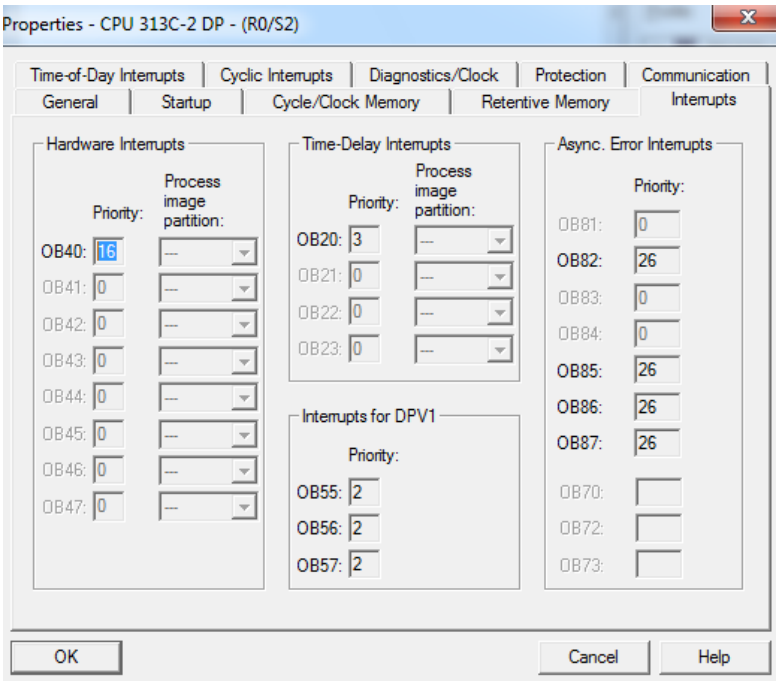
این سخت افزار ها میتواند کارت SM ، CP و FM باشند که دارای قابلیت Hardware interrupt باشند.

این وقفه ها شامل OB40 تا OB47 میشود که بسته به CPU انتخابی تعداد OBهای فعال آن متفاوت است.

درجه الویت هرکدام در زیر نمایش داده شده است.

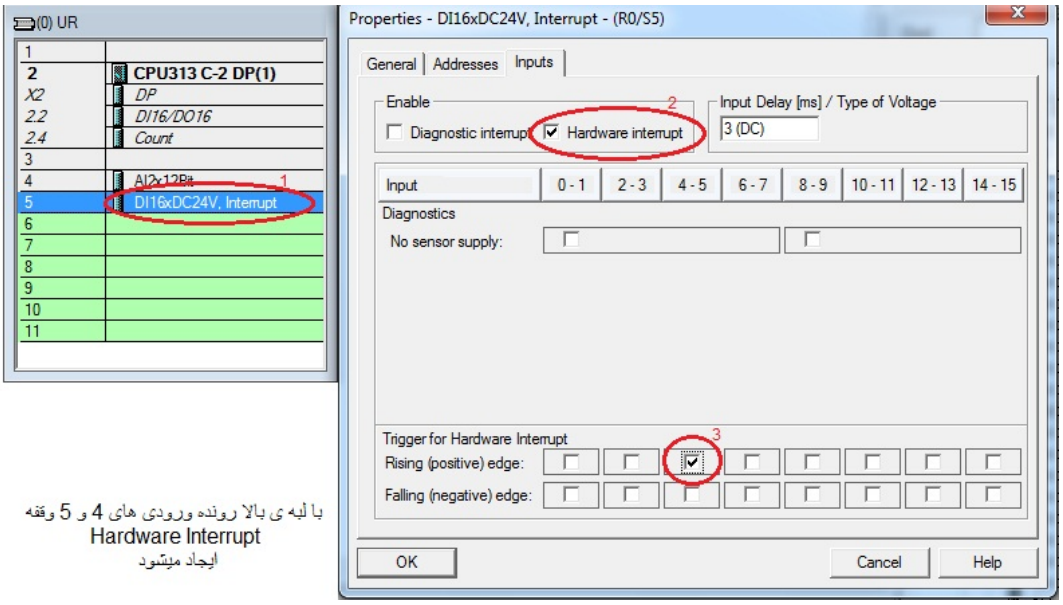
OB	Start Event	Default Priority Class
OB40	Hardware interrupt 0	16
OB41	Hardware interrupt 1	17
OB42	Hardware interrupt 2	18
OB43	Hardware interrupt 3	19
OB44	Hardware interrupt 4	20
OB45	Hardware interrupt 5	21
OB46	Hardware interrupt 6	22
OB47	Hardware interrupt 7	23

برای استفاده از این OBها در زیربرنامه HWconfig و تنظیمات CPU (تب Interrupt)، میتوان OB فعال جهت استفاده را شناسایی کرد و همچنین شماره بلوک فعال دلخواه را در قسمت BLOKs در برنامه SIMATIC Manager ایجاد و برنامه نویس کرد.



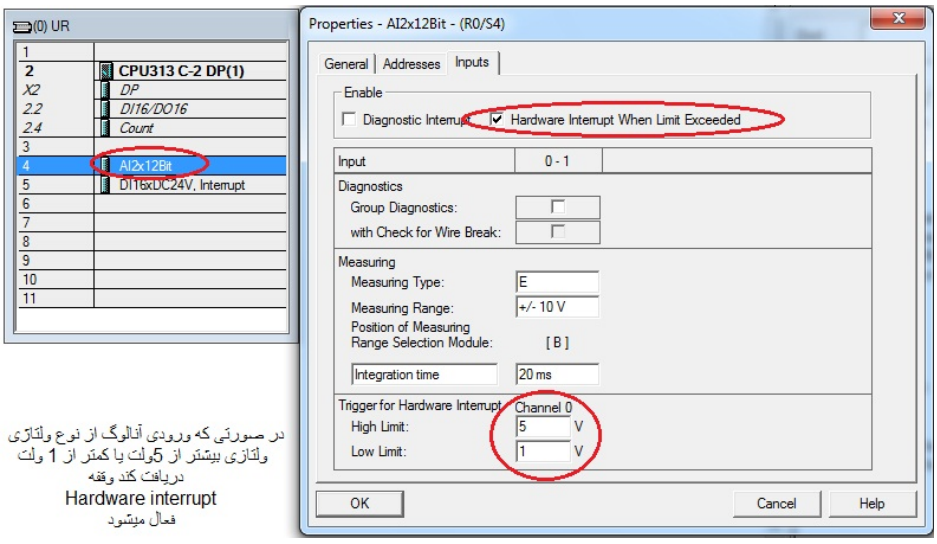
در CPU313C-2DP فقط OB40 را میتوان استفاده کرد.

- توجه شود برای استفاده از این وقفه باید حتما سخت افزاری با قابلیت ایجادHardware Interrupt استفاده شده باشد و تنظیمات آن در زیر برنامه HWconfig انجام شده باشد.



با لبه ی بالا رونده ورودی های 4 و 5 وقفه
Hardware Interrupt
ایجاد میشود

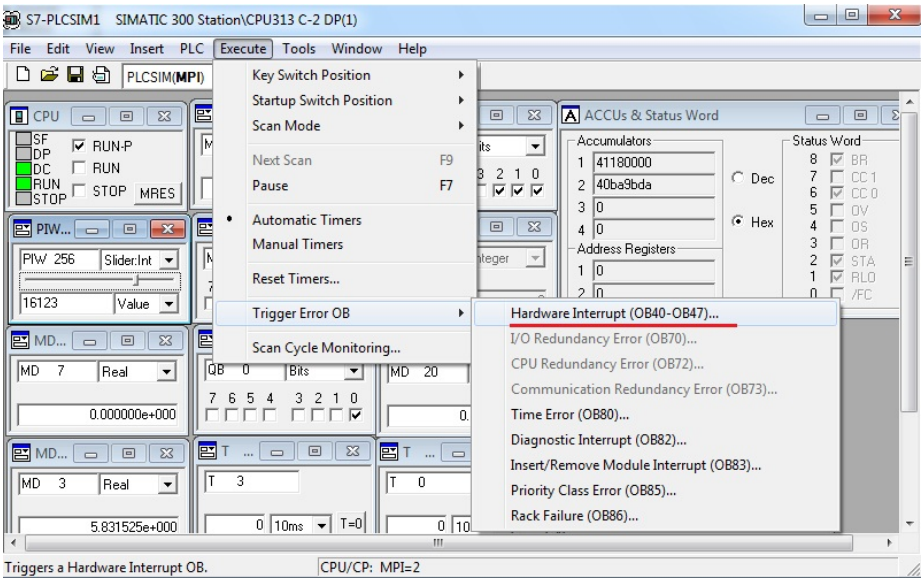
تنظیمات Hardware Interrupt در کارت ورودی دیجیتال



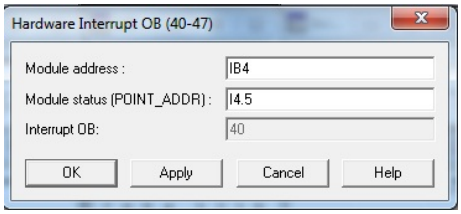
در صورتی که ورودی آنالوگ از نوع ولتاژی ولتاژی بیشتر از 5ولت یا کمتر از 1 ولت دریافت کند وقفه
Hardware interrupt
فعال میشود

تنظیمات Hardware Interrupt در کارت ورودی آنالوگ

- برای شبیه سازی ورودی اینتراپت دار در سیمولیشن باید در مسیر زیر در PLCSIM وارد شد.

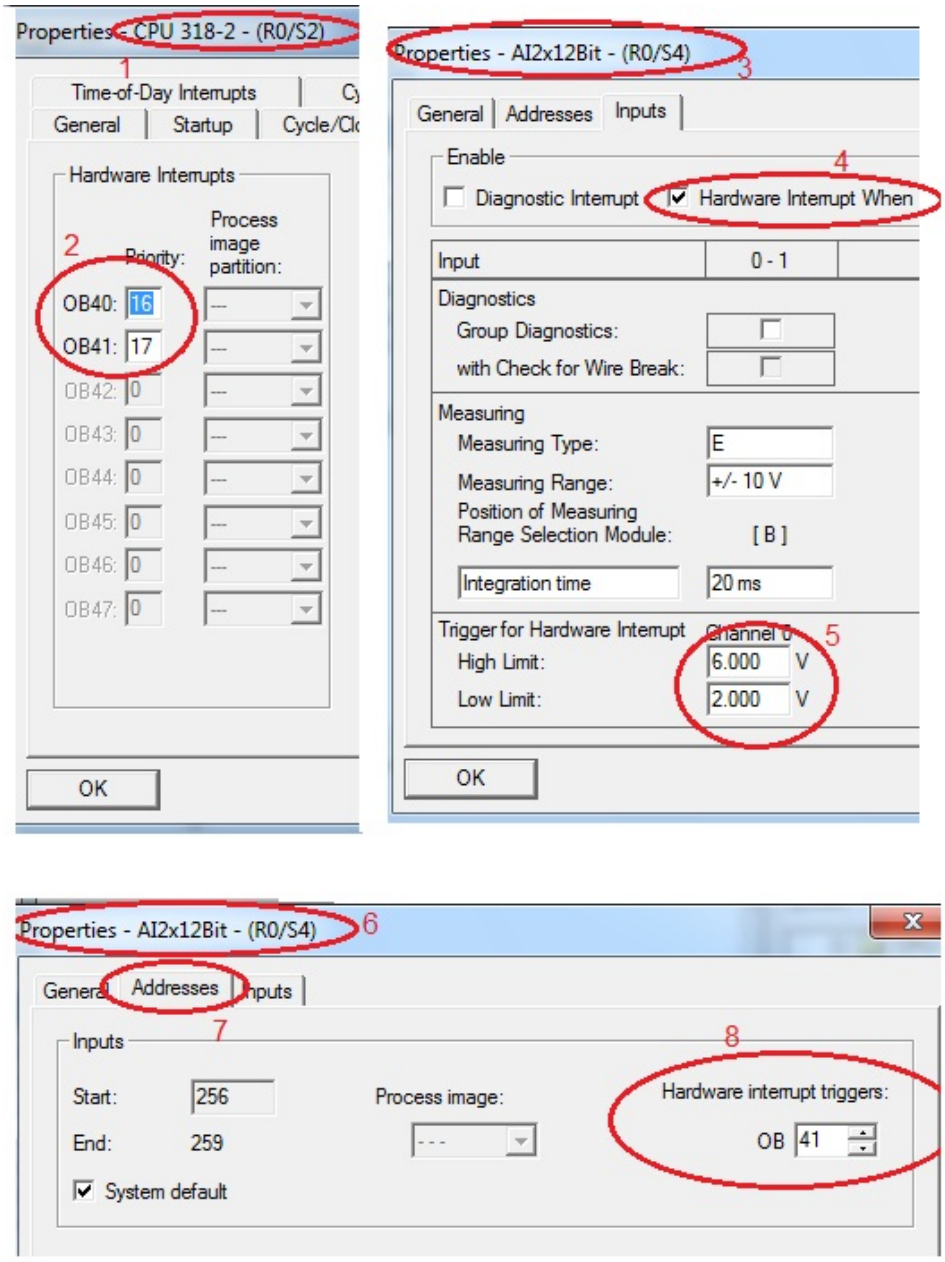


در پنجره باز شده آدرس پایه برای ورودی اینتراپت دار را در قسمت Module address وارد و آدرس ورودی دقیق را در Module sratus وارد میکنیم. با Apply کردن وقفه آن فعال میشود.



- فرض کنیم که بر اساس لبه بالا رونده یک ورودی دیجیتال یک وقفه سخت افزاری فراخانی شود. تا زمانی که CPU در اجرای وقفه سخت افزاری باشد وقفه های جدید سخت افزاری را مشاهده نمیکند.

- در صورتی که CPU بیش از یک OB برای وقفه سخت افزاری داشته باشد در قسمت Addresses کارت های دارای اینتراپت، باید شماره OB وقفه سخت افزاری مربوط به آن کارت را مشخص کرد.



وقفه های مبتنی بر ERROR

این وقفه ها که پیش تر اشاره شد ناشی از خطا هستند و نه رویداد از قبل تنظیم شده. این OB ها برای تشخیص خطا اهمیت زیادی دارند و هریک برای تشخیص های خاصی استفاده میشوند. به کار گیری مناسب این بلوک ها موجب تشخیص سریع خطا و همچنین نمایش نوع خطا ومحل خطا در HMI است.

خطا ها به طور کلی به دو دسته تقسیم میشوند

- خطای آسنکرون: ناشی از خرابی کارتها و خطای سخت افزاری. از OB80 تا OB87
- خطای سنکرون: ناشی از خطاهای برنامه و اجرای آن. OB121 و OB122

System Functions for Controlling Interrupt OBs

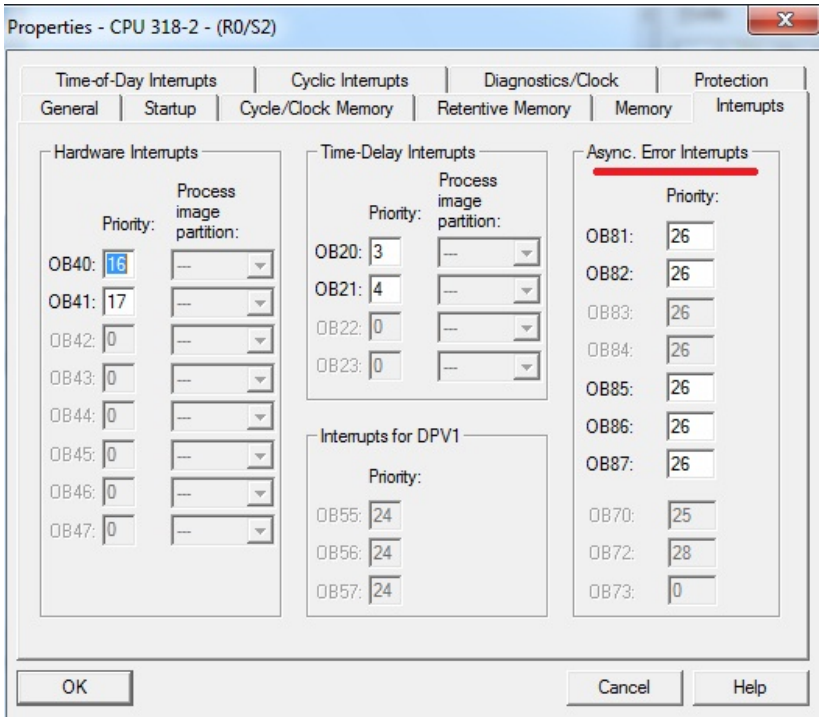
Organization block		Priority in S7-300
Function	Number	
Asynchronous error	OB 80 ... 87	26 , 28
Synchronous error	OB 121, 122	Priority of the OB causing the erro

خطاهای آسنکرون (Asynchronous error)

Type of error	Example	OB	Priority
Time error	Maximum scan cycle time exceeded	OB80	26
Power supply fault	Backup battery failure	OB81	
Diagnostic interrupt	Wire break at input of diagnostics-capable module	OB82	
Insert / remove interrupt	Removal of a signal module during operation of an S7-400	OB83	
CPU hardware fault	Incorrect signal level at the MPI interface	OB84	
Program execution error	Error in updating the process image (module defective)	OB85	
Rack fault	Failure of an expansion device or a DP slave	OB86	
Communication error	Error in reading message frame	OB87	26 / 28

درجه الویت و نام هر یک از این OB ها را مشاهده میکنید.

- این OB ها در قسمت Interrupt در تنظیمات CPU قابل بررسی است و بسته به نوع CPU میتواند فعال یا غیر فعال باشد. درجه اولویت این OBها از دیگر OBها بالاتر است و در صورت همزمانی این بلوک ها اجرا میشوند.

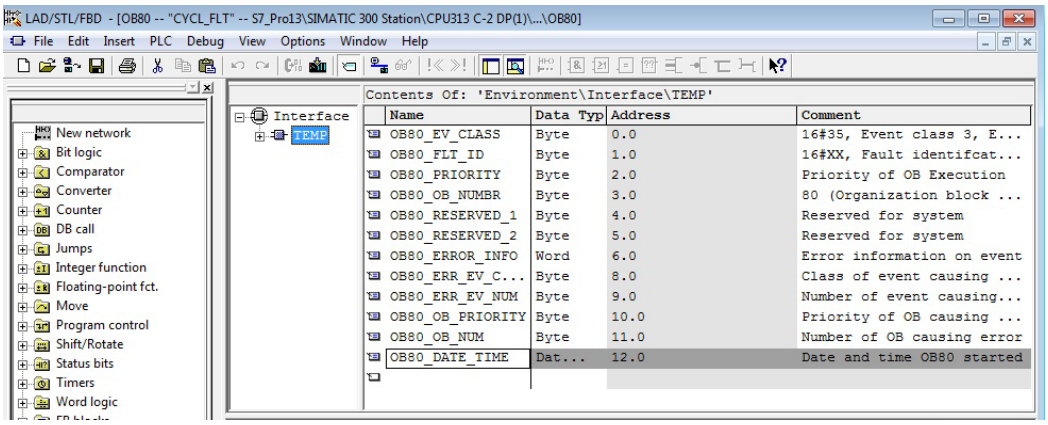


خطاهای سنکرون (Synchronous error)

Type of error	Example	OB	Priority
Programming error	A block that is not present in the CPU is called in the program	OB121	Same as that of the OB interrupted as a result of the error
Access error	A module which is either defective or not present is addressed in the program (e.g. direct access to a non-existent I/O module)	OB122	

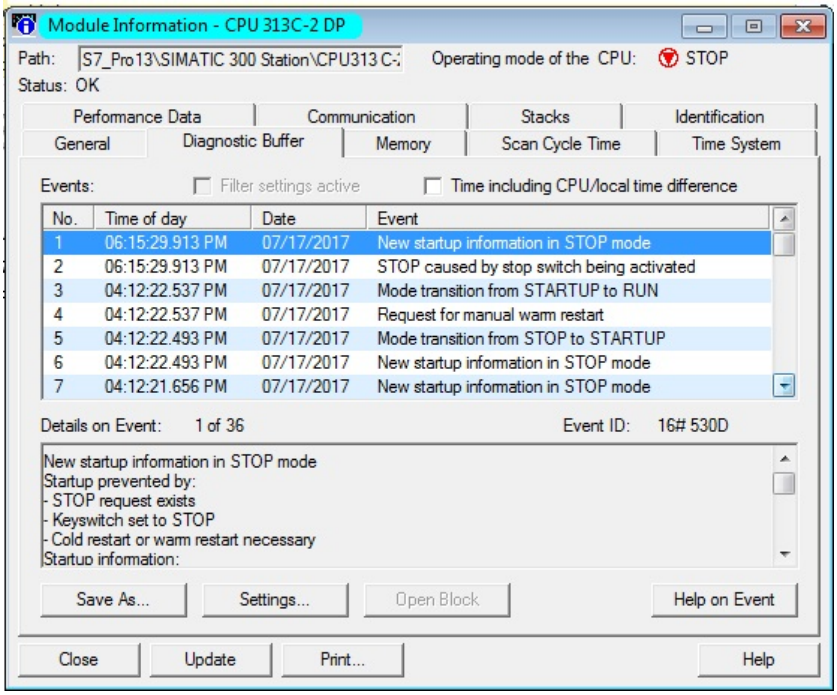
شماره OBها این خطاها و نام هر یک را مشاهده میکنید.

- در این بلاک های مربوط به وقفه های مبتنی بر خطا، در قسمت Interface متغیر های temp موجود است که برخلاف OB های دیگر که به آنها نیاز نداشتیم، این temp ها برای تشخیص خطا بسیار کاربردی و ضروری اند.



- درصورت بوجود آمدن خطا و نبود بلوک خطای مربوطه با روشن شدن SF، CPU به مد STOP میرود. در مسیر زیر میتوان علت خطا را بررسی کرد. بدیهی است این امکان تنها وقتی ممکن است که PC/PG به CPU متصل باشند.

در برنامه SIMATIC Manager و منوی PLC و در قسمت Diagonstic setting و در Module information در تب Diagonstic Buffer میتوان تمامی پیغام های CPU و تاریخ و حادثه ی پیش آمده را مشاهده کرد. در تب های بعدی مانند Scan Cycle Time میتوان زمان سیکل برنامه را مشاهده کرد.در تب Stacks اخرین اطلاعات حافظه را میتوان مشاهده کرد.



OB80 TIME ERROR

در هنگام رخ دادن خطاهای زمانی، سیستم عامل سراغ بلوک OB80 میرود. اگر این بلوک درCPU دانلود شده باشد آنرا اجرا ودر غیر این صورت به مد STOP میرود.

برخی از خطاهای زمانی عبارت اند از :

- تجاوز از سیکل اسکن
- فراخانی یک OB توسط سیستم عامل قبل از تمام شدن اجرای قبلی آن
- پرش از روی وقفه ی Time of Day توسط جلو کشیدن ساعت سیستم

در متغیرهای محلی (TEMP) این بلوک پارامتری به نام OB80_FLT_ID با وزن بایت وجود دارد که کد خطای خاص بسته به نوع خطا در این بایت ذخیره میشود .میتوان به راحتی با نوشتن برنامه ی ساده به صورت مقایسه ای نوع خطا را با ست کردن یک بیت مموری یا دیتا بلاک در سیستم HMI مانیتور کرد.

در TEMP دیگر به نام OB80_DATE_TIME زمان و تاریخ خطا که باعث فراخانی OB80 شده است در این متغیر قرار میگیرد.

خطای تجاوز از سیکل اسکن

این خطا به دلیل لوپ بینهایت، طولانی شدن برنامه OB1، فراخانی بلوک های وقفه متعدد و افزایش زمان سیکل اسکن منجر به ایجاد این خطا میشوند.

در این صورات تغییرات زیر رخ میدهد

OB80_FLT_ID = مقدار 01

OB80_ERROR_INFO = آخرین مقدار زمان سیکل اسکن

OB80_OB_NUM = شماره آخرین OB اجرا شده

فراخانی یک OB توسط سیستم عامل قبل از تمام شدن اجرای قبلی آن

در این صورات تغییرات زیر رخ میدهد:

OB80_FLT_ID = مقدار 02

OB80_OB_NUM = شماره آخرین OB اجرا شده

پرش از روی وقفه ی Time of Day توسط جلو کشیدن ساعت سیستم

مقدار 05 = OB80_FLT_ID

با توجه به شماره time of day پرش شده بیت صفر تا 7 به نشانه ی OB10 تا OB17 = OB80_ERROR_INFO

ساده ترین برنامه برای OB80 در این صورت فقط میتوان متوجه شد علت خطا سیکل اسکن برنامه بوده است. بدیهی است که برای ریست شدن این بیت در OB1 باید برنامه بنویسیم.

```
SET
= M 0.0
```

OB81 Power Supply Error

در صورت بروز خطای منبع تغذیه، سیستم عامل این بلوک را فراخانی میکند ولی در صورتی که این بلوک موجود نباشد برخلاف سایر خطاها CPU به مد استپ نمیرود.

برخی از خطاها عبارت اند از

- خراب شدن باطری Backup مربوط به منبع تغزیه رک (در سری 400)
- اشکال در تغذیه Backup مربوط به رک (در سری 400)
- اشکال در تغذیه 24 ولت مربوط به رک (در سری 400)

متغیر های محلی این بلوک که برای نوشتن برنامه در همین بلوک استفاده میشود به صورت زیر تغیر میکنند:

کد خطا = OB81_FLT_ID

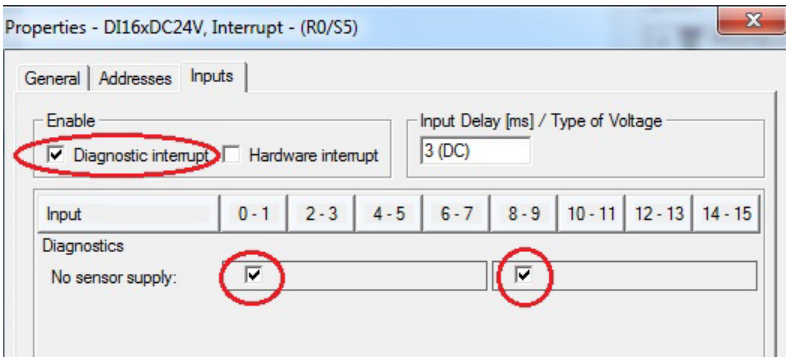
تاریخ و زمان بروز خطا=OB81_DATE_TIME

به علت اینکه این بلاک در PLCهای سری 300 موجود نیست از پرداختن مفصل تر به آن خود داری شده است.

OB82 Diagnostic Interrupt

این خطاها توسط کارتهای ورودی و خروجی که قابلیت Diagnostic یا تشخیص عیب را دارند آشکار میشود.

برای استفاده از این وقفه باید کارت های استفاده شده دارای این قابلیت باشند و در تنظیمات آنها Diagnostic Interrupt فعال شده باشد و OB82 در BLOKs موجود باشد. تنظیمات کارت را در شکل زیر مشاهده میکنید.



متغیر های محلی این بلوک که برای نوشتن برنامه در همین بلوک استفاده میشود به صورت زیر تغیر میکنند:

برابر با مقدار OB82_FLT_ID =42

مقدار 84 مبین خطا در کارت ورودی و مقدار 85 مبین خطا در کارت خروجی = OB82_IO_FLAG

آدرس پایه کارت با فورمت WORD قرار میگیرد = OB82_MDL_ADDR

به صورت بیتی و یک شدن آن مبین خطای داخلی است = OB82_INT_FAULT

بیه صورت بیتی و یک شدن آن مبین خطای خارجی است= OB82_EXT_FAULT

بیه صورت بیتی و یک شدن آن مبین خطا در تغذیه بیرونی کارت = OB82_EXT_VOLTAGE

به صورت بیتی و یک شدن آن مبین خطا در Front Connector = OB82_FLD_CONNCTR

به صورت بیتی و یک شدن آن مبین خطا در Submodule = OB82_SUB_MDL_ERR

به صورت بیتی و یک شدن آن مبین خطا در مبدل ATD یا DTA = OB82_ADU_FLT

تاریخ و زمان بروز خطا= OB81_DATE_TIME

OB83 Insert/Remove Module Interrupt

خطای اضافه یا برداشته شدن کارت های SM و FM و CP . این خطا بیشتر در سری 400 نمود پیدا میکند چون در سری 300 فقط کارت آخر امکان جدا شدن دارد و اگر کارتی در میان برداشته شود باس کانکتور و ارتباط سایر کارت ها قطع میشود.

متغیر های محلی این بلوک که برای نوشتن برنامه در همین بلوک استفاده میشود به صورت زیر تغیر میکنند:

OB83_MDL_ADDR = ادرس پایه کارت

OB83_DATE_TIME = تاریخ و زمان وقوع خطا

OB84 CPU Hardware Fault

در صورت بروز خطا در ارتباط CPU با شبکه MPI و همچنین خطا در ارتباط CPU با Communication Bus و در هنگام وقوع و برطرف شدن، این بلوک فراخانی میشود

متغیر های محلی این بلوک که برای نوشتن برنامه در همین بلوک استفاده میشود به صورت زیر تغیر میکنند:

OB84_EV_CLASS = Outgoing مقدار 38 برای و Incoming مقدار 39 برای

OB85 Priority Class Error

خطای کلاس الویت در شراط زیر رخ میدهد

1- فراخانی OBخاص و عدم وجود آن درBLOCKs
2- عدم دسترسی سیستم عامل به کارتها
3- عدم دسترسی به I/O در هنگام آپدیت کردن PII/PIQ

متغیر های محلی این بلوک که برای نوشتن برنامه در همین بلوک استفاده میشود به صورت زیر تغیر میکنند:

OB85_DATE_TIME = تاریخ و زمان وقوع خطا

OB85_FLT_ID = مقدارA1 برای خطای شماره 1

مقدار A3 برای خطای نوع 2

مقدار B1 خطای نوع 3 برای Input

مقدار B2 برای خطای نوع 3 برای Output

OB86 Rack Failure

این بلوک در صورت رخ دادن خطاهای زیر فراخانی میشود

1- خطای رک برای IM یا قطعی کابل ارتباطی بین IMها
2- اشکال در DP Slave در شبکه PROFIBUS

متغیر های محلی این بلوک که برای نوشتن برنامه در همین بلوک استفاده میشود به صورت زیر تغیر میکنند:

وقوع خطا مقدار 39 و برطرف شدن خطا مقدار OB86_EV_CLASS = 38

مقدار C1 برای خطای نوع 1 و مقدار C3 خطای نوع OB86_FLT_ID = 2

آدرس بیس (در نوع 1 برای IM و در نوع 2 برای DP Slave) = OB86_MDL_ADDR

متغیر با وزن DWORD که بیت های آن در خطای نوع یک مبین رک خطا دیده است = OB86_Z23

به این صورت که بیت 0 همیشه صفر و بیت 1 تا 21 هر کدام یک بود خطا در رک همشماره در بیت اتفاق افتاده است.

در خطای نوع دوم بایت دوم (بیت 8 تا 15) شماره ID مربوط به SLAVE را در خود دارد.

OB121 Programming Error

خطاهای ناشی از برنامه نویسی است و هر جا CPU در پردازش برنامه با اشکال روبرو شود این بلوک فراخانی میشود.

- بلاک FC یا FB که فراخانی شده موجود نباشد.
- آدرسی که برای تایمر، کانتر،یا سایر متغیرهای حافظه داده شده خارج از حد مجاز برای CPU است.

متغیر های محلی این بلوک که برای نوشتن برنامه در همین بلوک استفاده میشود به صورت زیر تغیر میکنند:

OB121_SW_FLT =	21	خطا در تبدیل BCD
	26	خطای شماره تایمر
	27	خطای شماره کانتر
	30	خطای نوشتن روی DBاشتراکی که Write protect شده
	31	خطای نوشتن روی DB خاص که Write protect شده
	32	خطای شماره DB اشتراکی
	33	خطای شماره DB اختصاصی
	34	خطای شماره FC هنگام فراخانی
	35	خطای شماره FB هنگام فراخانی
	3A	عدم دسترسی به DB
	3C	عدم دسترسی به FC
	3D	

OB122 I/O Access Error

در صورت بروز خطای دسترسی به خواندن ورودی ها و نوشتن خروجی ها، سیستم عامل OB122 را فراخانی میکند و اگر موجود نباشد، CPU به مد STOP میرود.

متغیر های محلی این بلوک که برای نوشتن برنامه در همین بلوک استفاده میشود به صورت زیر تغیر میکنند:

مقدار 42 خطا در خواندن ورودی، مقدار 43 خطا در نوشتن خروجی = OB122_SW_FLT

یک Word که آدرس پایه کارت دارای مشکل را نمایش میدهد = OB122_MEM_ADDR

بلوکی که در آن خطا رخ داده: 88 برای OB، 8C برای FC، 8E برای FB = OB122_BLK_TYPE

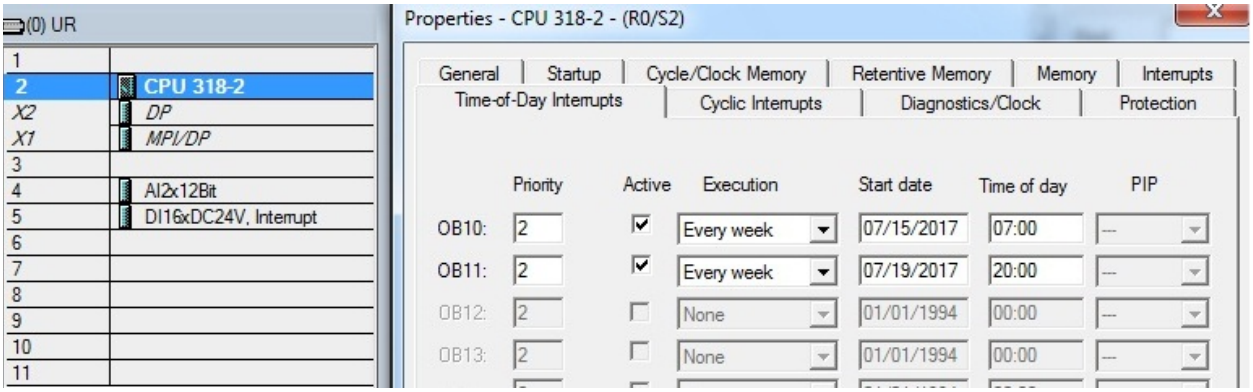
تاریخ وزمان وقوع خطا = OB122_DATE_TIME

مثال برای وقفه Time of Day

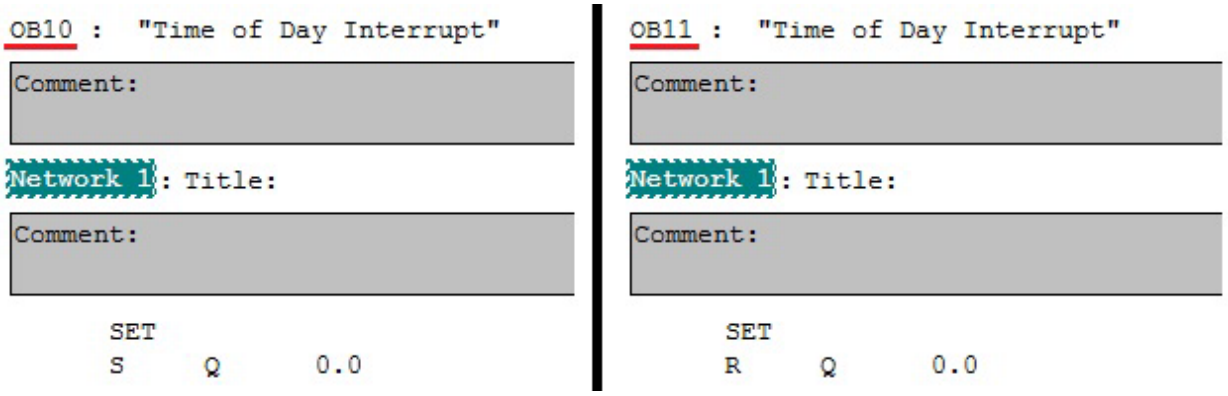
میخاهیم یک خروجی شنبه ساعت 7 صب روشن و چهار شنبه ساعت 8 شب خاموش شود و این سیکل ادامه دار باشد:

برای این کار یک CPU با دو OB Time of Day نیاز داریم مثلا CPU 318-2

تاریخ اولین شنبه آینده و چهارشنبه بعد آن را برای فعال کردن OB ها تنظیم میکنیم و Active میکنیم و Save & Compile میکنیم.



در OB10 برنامه ی روشن شدن و در OB11 برنامه خاموش شدن را مینوسیم.

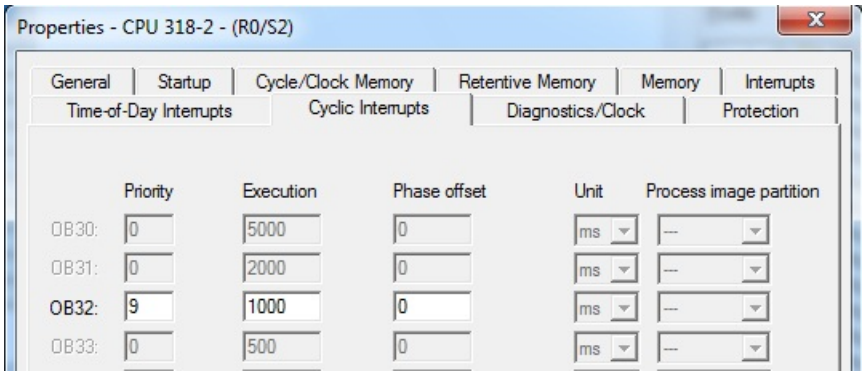


حال باید بلوک های موجود و Systemdata را دانلود کنیم . خروجیQ0.0 در شنبه ساعت 7:00 روشن و چهارشنبه ساعت 20:00 خاموش میشود.

مثال برای Cyclic Interrupt

یک چراغ چشمک زن توسط بلوک وقفه سیکلی با فرکانس 1Hz

ابتدا باید فاصله ی زمانی 1000 mS برای وقفه سیکلی تنظیم کرد و بعد از آن ساخت OB مربوط و نوشتن برنامه.



```
OB32 :  "Cyclic Interrupt"
Comment:

Network 1: Title:
Comment:

      AN    Q       0.0
      =     Q       0.0
```

مثال برای برنامه نویسی OB80 TIME ERROR

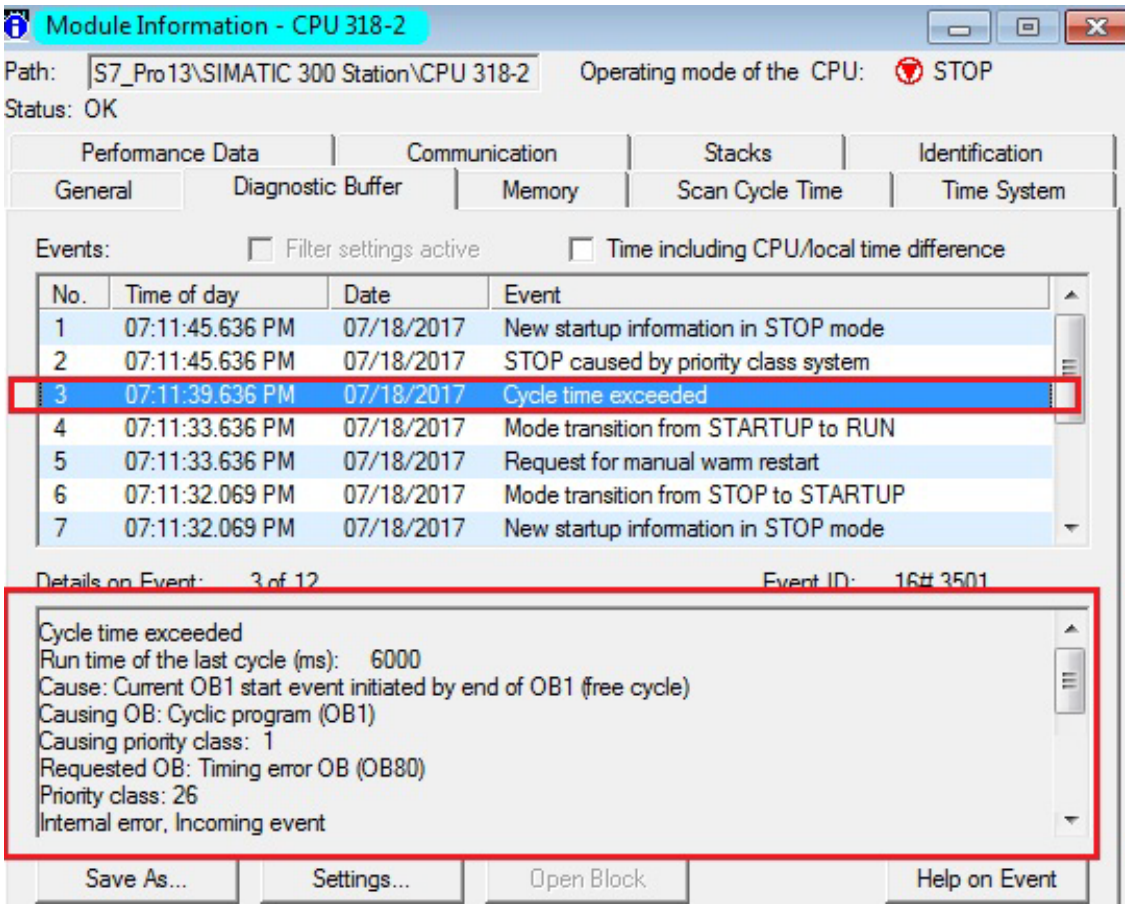
در بلاک OB80 برنامه زیر را مینویسیم تا برای اوپراتور از طرق مانیتورینگ پیام مربوط به خطا نمایش داده شود.

این برنامه با ست کردن بیت هایی در دیتا بلاک، و استفاده از آنها در مانیتورینگ میتواند مشخص کند چه نوع خطای زمانی رخ داده است و شماره OB که باعث ایجاد خطا شده را هم نمایش داد.

```
OB80 :  "Cycle Time Fault"
Comment:

Network 1: Title:
Comment:

      L     #OB80_FLT_ID              #OB80_FLT_ID       -- 16#XX, Fault identifcation code
      L     1
      ==I
      JC    M001
      L     #OB80_FLT_ID              #OB80_FLT_ID       -- 16#XX, Fault identifcation code
      L     2
      ==I
      JC    M002
      L     #OB80_FLT_ID              #OB80_FLT_ID       -- 16#XX, Fault identifcation code
      L     5
      ==I
      JC    M003
      L     #OB80_OB_NUM              #OB80_OB_NUM       -- Number of OB causing error
      T     DB1.DBB    0
      BEU
M001: S     DB1.DBX    1.0
      BEU
M002: S     DB1.DBX    1.1
      BEU
M003: S     DB1.DBX    1.2
      BE
```

نکته: برنامه ی ریست شدن تمامی مقادیر ست شده در برنامه بالا از یک شاسی مخصوص یا یک بیت حافظه در مانیتورینگ و نوشتن برنامه ریست شدن آنها در OB1 استفاده میکنیم.

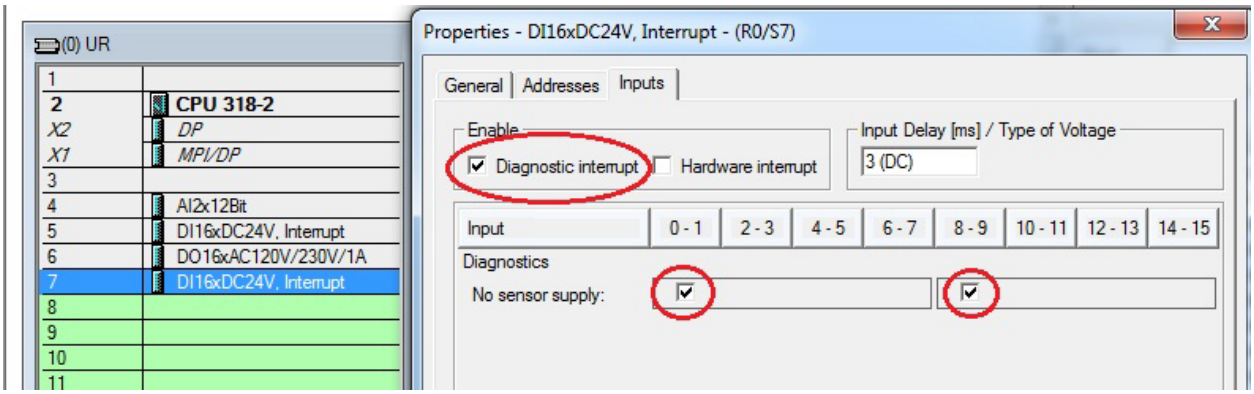
همانطور که قبلا ذکر شده برای تشخیص نوع خطا میتوان به Module Information مراجعه کرد. برای خطای لوپ بینهایت در زیر مشاهده میکنید. مشخص شده خطا از چه نوعی است و توضیحات کامل داده شده است.

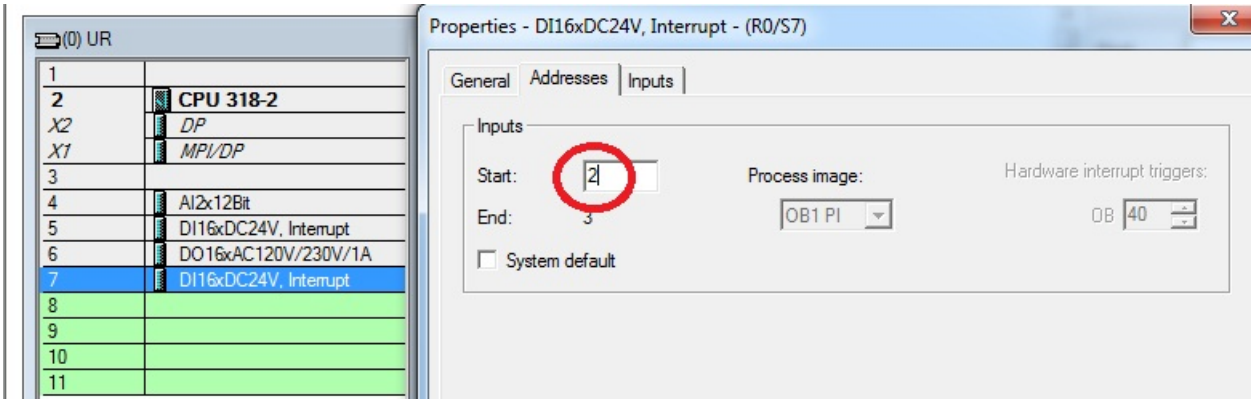


مثال برای OB82 Diagnostic Interrupt

برای شبیه سازی بررسی این خطا باید کارتی دارای قابلیت Diagnostic Interrupt داشته باشیم. تنظیمات آدرس را روی کارت انجام میدهیم و برنامه مورد نظر را در OB82 مینوسیم. از کارت DI16xDC24V, SM321 Interrupt با ادرس بیس 2 استفاده میکنیم.

تنظیمات را Save & Compile میکنیم.



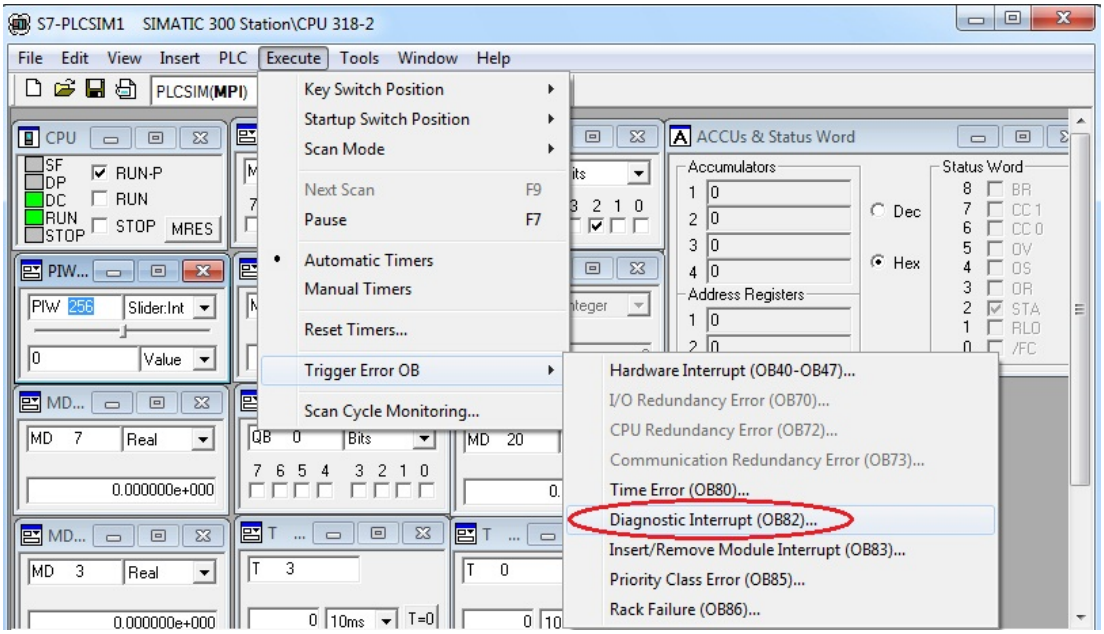


در OB82 برنامه زیر را مینوسیم.

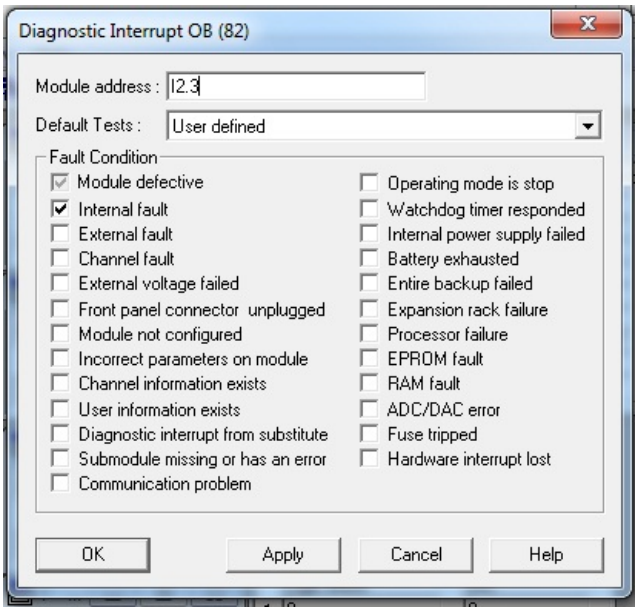
```
OB82 :  "I/O Point Fault"
Comment:

Network 1: Title:
Comment:

      L     #OB82_IO_FLAG                                  #OB82_IO_FLAG      -- Input (01010100),
      L     84
      ==I
      =     M      0.0                 // INPUT  MODULE
      L     #OB82_IO_FLAG                                  #OB82_IO_FLAG      -- Input (01010100),
      L     85
      ==I
      =     M      0.1                 // OUTPUT MODULE
      L     #OB82_MDL_ADDR                                 #OB82_MDL_ADDR     -- Base address of r
      T     MW     1                   // BASE ADDRES
      A     #OB82_INT_FAULT                                #OB82_INT_FAULT    -- Internal fault
      =     M      0.2                 // INTERNAL FAULT
      A     #OB82_EXT_FAULT                                #OB82_EXT_FAULT    -- External fault
      =     M      0.3                 // EXTERNAL FAULT
      A     #OB82_FLD_CONNCTR                              #OB82_FLD_CONNCTR  -- Field wiring conn
      =     M      0.4                 // FRONTCONNECTOR FAULT
      A     #OB82_ADU_FLT                                  #OB82_ADU_FLT      -- ADU fault
      =     M      0.5                 // ATD/DTA FAULT
```

برنامه نوشته شده را Save میکنیم و بلوک های موجود و Systemdata را در سیمولیشن دانلود میکنیم.

برای شبیه سازی خطای در سیمولیشن به مسیر زیر میرویم.

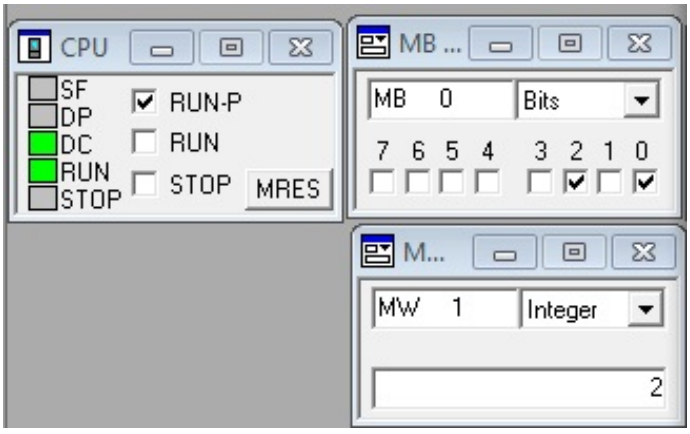


در پنجره ی باز شده نوع خطارا انتخاب و آدرس یک بیت از کارت را وارد میکنیم و در انتها Apply میکنیم.



مشاهده میکنیم که بیت های نوشته شده در برنامه با توجه به نوع خطا فعال شده است.

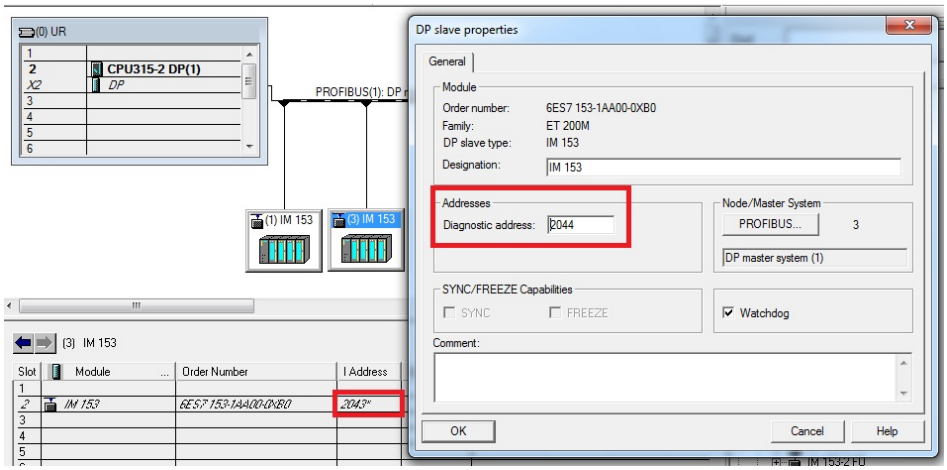
متوجه میشویم خطایی در کارت ورودی با آدرس بیس 2 از نوع داخلی ایجاد شده است.



برنامه نویسی و تست در سیمولیشن (در صورت وجود آن در Trigger Error OB) به همین صورت با توجه به متغیر های محلی بلوک مورد نظر انجام میشود.

مثال برای OB86 DP Slave

فرض کنید در شبکه پروفیباس یک CPU 315-2DP داریم و دو واحد ET-200M که آدرس diagnostic این دو تجهیز 2044 و 2046 میباشد.



برنامه را در OB86 مینویسیم

```
OB86 :  "Loss Of Rack Fault"
Comment:

Network 1: Title:
Comment:

      L     #OB86_MDL_ADDR              #OB86_MDL_ADDR
      L     2044
      ==I
      S     M    100.0

      L     #OB86_MDL_ADDR              #OB86_MDL_ADDR
      L     2046
      ==I
      S     M    100.1
```

بیت های M100.0 و M100.1 در HMI تگهای مربوط به آلارم خطای فوق هستند.

مثال برای OB85 Priority Class Error

برنامه را در OB85 مینوسیم تا در صورت رخ داد خطاهای مربوطه، در HMI پیغام ها متناسب با آن ظاهر شود.

در HMI تگ های مورد نظر را با آدرس های M100.0 تا M100.5 ایجاد میکنیم

```
OB85 :  "Organization Block (OB) Not Loaded Fault"
Comment:

Network 1: Title:
OB85_OB_NUM

      A(
      L     #OB85_EV_CLASS
      L     B#16#35
      )
      JNB   M000

      L     #OB85_FLT_ID
      L     B#16#A1
      ==I
      =     M    100.0                 // OB that is not loaded on the CPU.

      L     #OB85_FLT_ID
      L     B#16#A3
      ==I
      =     M    100.1                 //Error when the operating system accesses a module

      L     #OB85_FLT_ID
      L     B#16#A4
      ==I
      =     M    100.2                 //PROFINET Interface DB cannot be addresse

M000: A(
      L     #OB85_EV_CLASS
      L     34
      ==I
      )
      A(
      L     #OB85_FLT_ID
      L     B#16#A4
      ==I
      )
      =     M    100.3                 //PROFINET Interface DB can be addressed again

      A(
      L     #OB85_EV_CLASS
      L     B#16#39
      ==I
      )
      JNB   M001

      L     #OB85_FLT_ID
      L     B#16#B1
      ==I
      =     M    100.4                 // I/O access error when updating the process image of the inputs

      L     #OB85_FLT_ID
      L     B#16#B2
      ==I
      =     M    100.5                 // I/O access error when transferring the output process image to the output modules

M001: NOP   0
```

مثال برای OB121 Programming Error

برنامه ای در OB121 مینوسیم که در صورت ایجاد خطای برنامه نویسی بتوانیم علت خطا را در HMI ببینیم

```
OB121 :  "Programming Error"
Comment:

Network 1: Title:
Comment:

      L     #OB121_SW_FLT
      L     B#16#21
      ==I
      =     M    101.0                //BCD conversion error

      L     #OB121_SW_FLT
      L     B#16#26
      ==I
      =     M    101.1                // Error for timer number

      L     #OB121_SW_FLT
      L     B#16#27
      ==I
      =     M    101.2                // Error for counter number

      L     #OB121_SW_FLT
      L     B#16#30
      ==I
      =     M    101.3                //Write access to a write-protected global DB

      L     #OB121_SW_FLT
      L     B#16#31
      ==I
      =     M    101.4                //Write access to a write-protected instance DB

      L     #OB121_SW_FLT
      L     B#16#32
      ==I
      =     M    101.5                //DB number error accessing a global DB

      L     #OB121_SW_FLT
      L     B#16#33
      ==I
      =     M    101.6                //DB number error accessing an instance DB

      L     #OB121_SW_FLT
      L     B#16#34
      ==I
      =     M    101.7                //FC number error in FC call

      L     #OB121_SW_FLT
      L     B#16#35
      ==I
      =     M    102.0                //FB number error in FB call

      L     #OB121_SW_FLT
      L     B#16#3A
      ==I
      =     M    102.1                //Access to a DB that has not been loade


      L     #OB121_SW_FLT
      L     B#16#3C
      ==I
      =     M    102.2                //Access to an FC that has not been loaded


      L     #OB121_SW_FLT
      L     B#16#3E
      ==I
      =     M    102.3                //Access to an FB that has not been loaded
```

مثال برای OB122 I/O Access Error

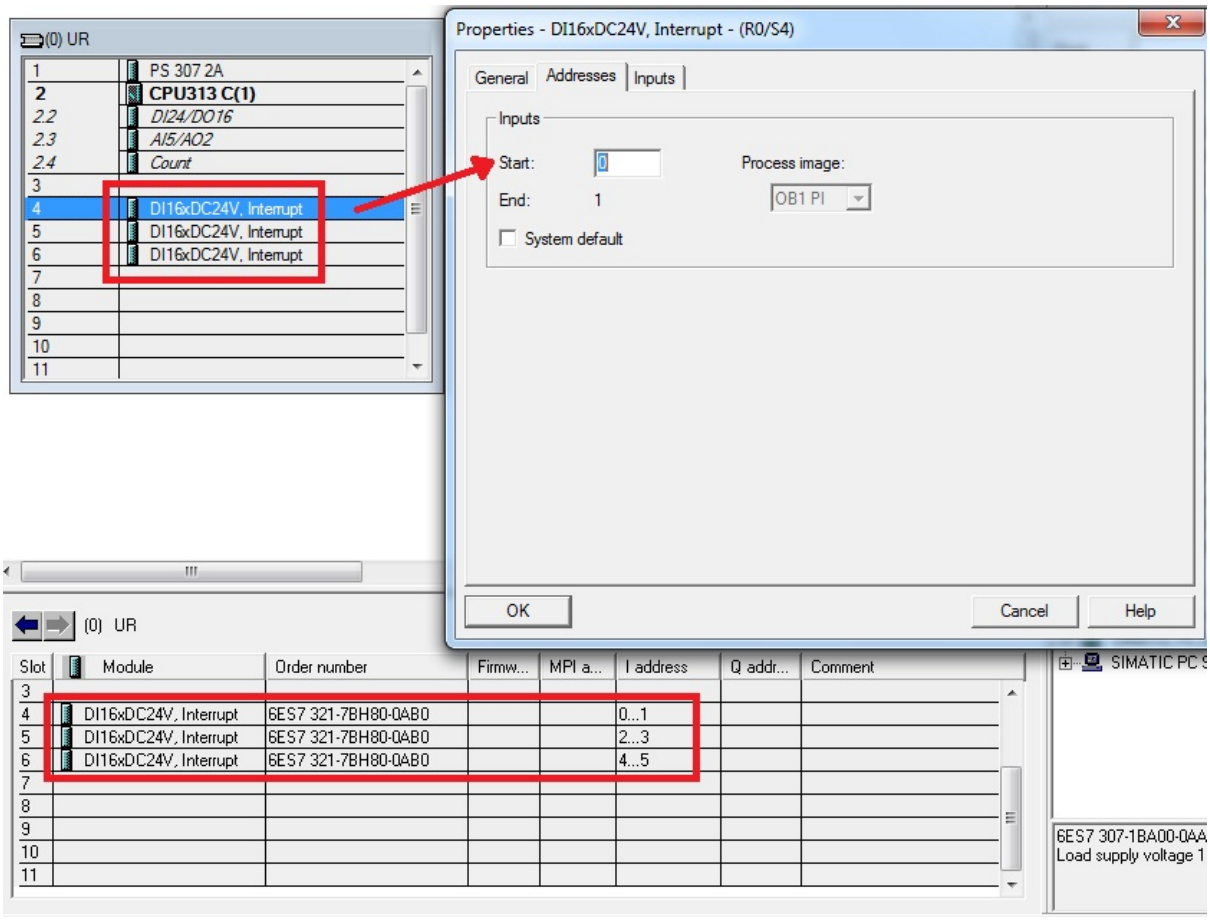
نوع خطا و آدرس بیس خطا در هنگام خطای دسترسی به ورودی و خروجی مشخص شود .

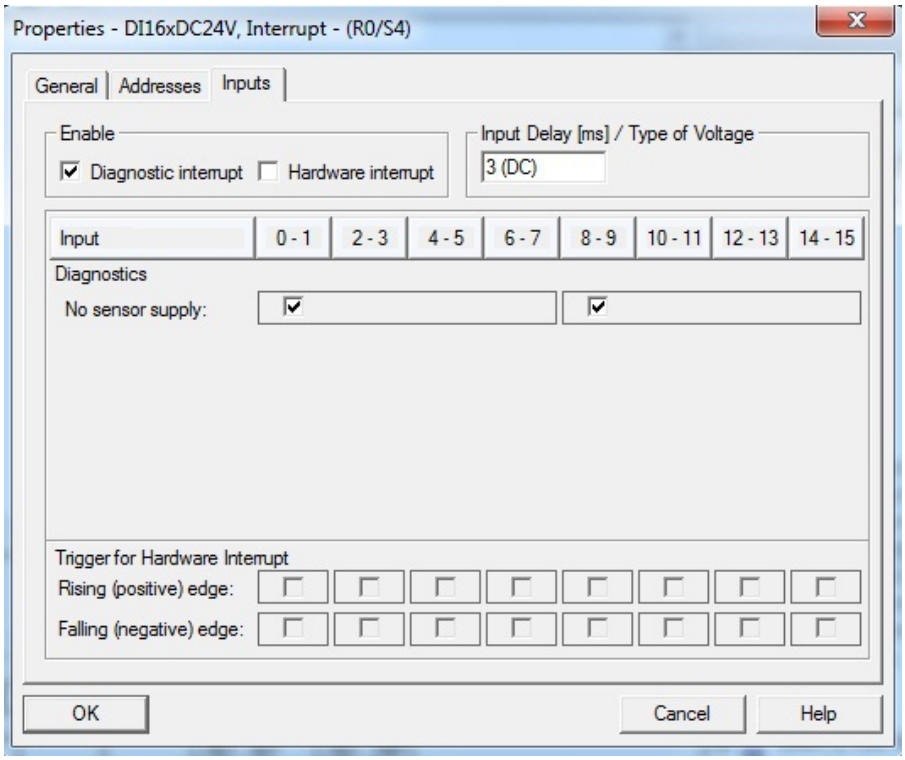
```
OB122 :  "Module Access Error"
Comment:

Network 1: Title:
Comment:

      L     #OB122_SW_FLT
      L     B#16#42
      ==I
      =     M    103.0                 //I/O access error, reading

      L     #OB122_SW_FLT
      L     B#16#43
      ==I
      =     M    103.1                 //I/O access error,Writing

      L     #OB122_SW_FLT
      L     B#16#43
      ==I
      =     M    103.2                 //I/O access error,Writing

      L     #OB122_MEM_ADDR
      T     MW   104                   //Memory address where the error occurred
```

مثال برای OB82

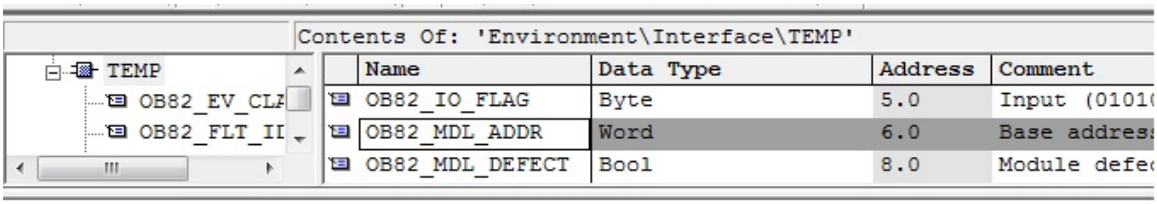
سه کارت DI با قابلیت DIAGNOSTIC داریم، برنامه ای بنویسیدکه اگر فیوز هر یک از کارت ها سوخت آدرس بیس کارت در MW1 نمایش داده شود و همچنین شماره اسلات کارت را در ریل نمایش دهد.

ابتدا تنظیمات سخت افزاری را برای مثال فوق انجام میدهیم. سه کارت DI16xDC24V,Interrupt را در اسلات های 4 و 5 و 6 قرار میدهیم. آدرس بیس هرکدام را به ترتیب 0 ، 2 ، 4 تنظیم میکنیم و همچنین قابلیت Diagnostic را برای هر سه کارت فعال می کنیم.


(0) UR
1 PS 307 2A
2 CPU313 C(1)
2.2 DI24/DO16
2.3 AI5/AO2
2.4 Count
3
4 DI16xDC24V, Interrupt
5 DI16xDC24V, Interrupt
6 DI16xDC24V, Interrupt
7
8
9
10
11
Properties - DI16xDC24V, Interrupt - (R0/S4)
General
Addresses
Inputs
Inputs
Start: 0
End: 1
Process image:
OB1 PI
System default
OK
Cancel
Help
(0) UR
Slot
Module
Order number
Firmw...
MPI a...
I address
Q addr...
Comment
3
4 DI16xDC24V, Interrupt 6ES7 321-7BH80-0AB0 0...1
5 DI16xDC24V, Interrupt 6ES7 321-7BH80-0AB0 2...3
6 DI16xDC24V, Interrupt 6ES7 321-7BH80-0AB0 4...5
7
8
9
10
11
SIMATIC PC S
6ES7 307-1BA00-0AA
Load supply voltage 1

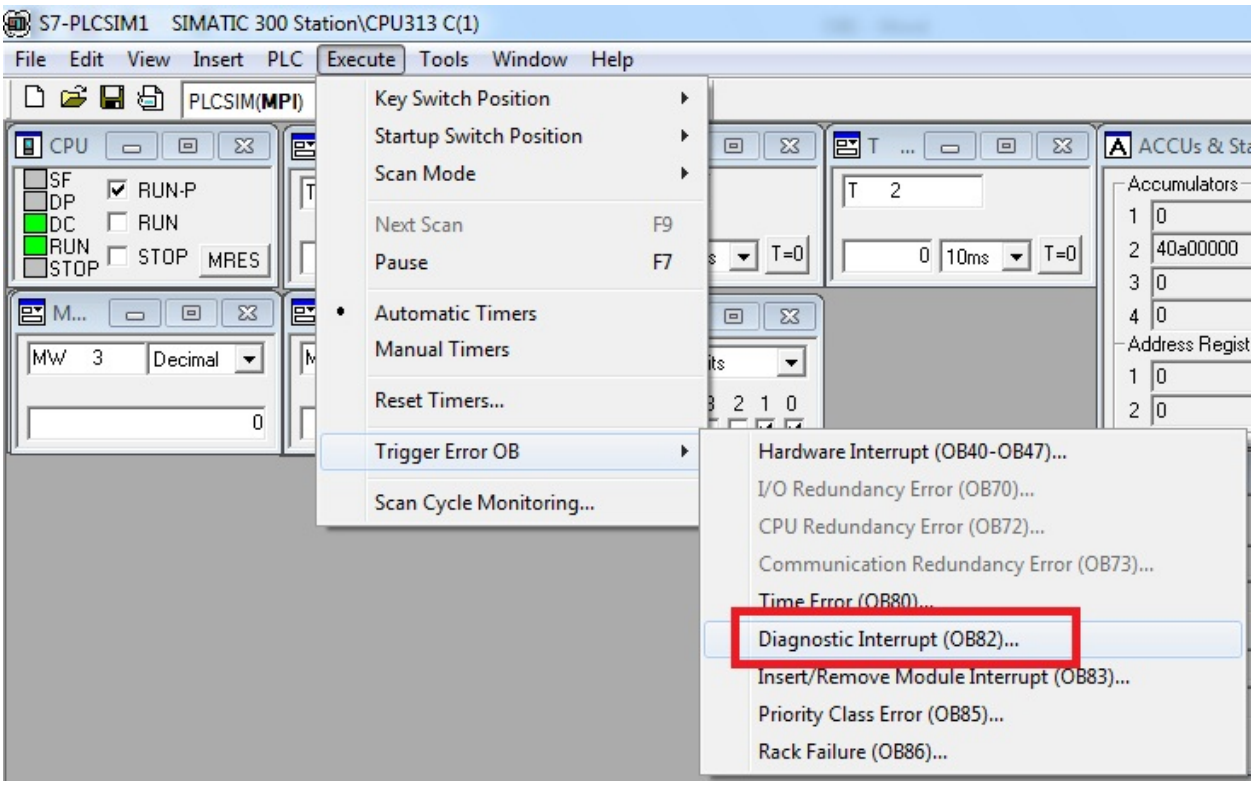

Properties - DI16xDC24V, Interrupt - (R0/S4)
General
Addresses
Inputs
Enable
Diagnostic interrupt
Hardware interrupt
Input Delay [ms] / Type of Voltage
3 (DC)
Input
0 - 1
2 - 3
4 - 5
6 - 7
8 - 9
10 - 11
12 - 13
14 - 15
Diagnostics
No sensor supply:
Trigger for Hardware Interrupt
Rising (positive) edge:
Falling (negative) edge:
OK
Cancel
Help

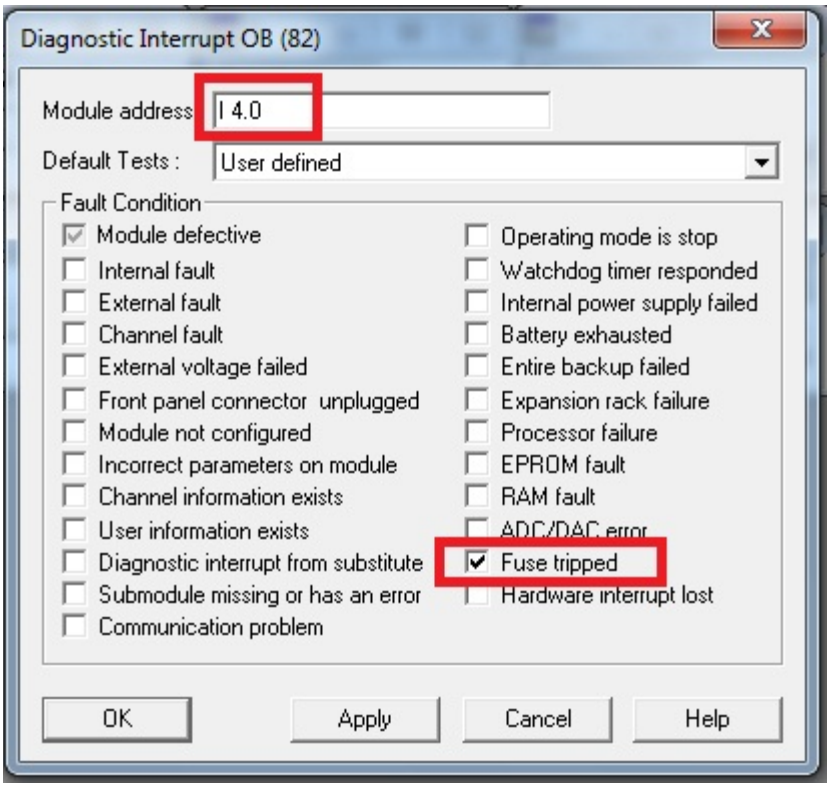
سپس بلوک OB82را میسازیم و برنامه را با استفاده از متغیرهای محلی موجود در این بلوک، مینویسیم.



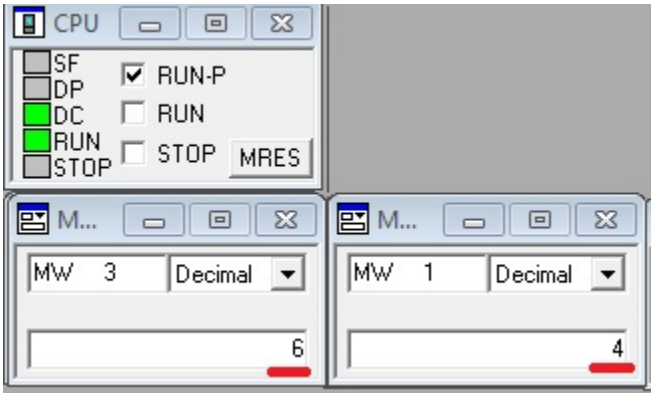

```
OB82 :  "I/O Point Fault"
Comment:
Network 1: Title:
Comment:

      A     #OB82_FUSE_FLT                                  #OB82_FUSE_FLT
      =     M      0.0                 // TAG FOR  FUSE FAULT
      L     #OB82_MDL_ADDR                                  #OB82_MDL_ADDR
      T     MW     1                   // TAG FOR BASE ADDRESS
      L     MW     1
      L     0
      ==I
      JC    m000
      L     MW     1
      L     2
      ==I
      JC    m001
      L     MW     1
      L     4
      ==I
      JC    m002
      BEU
m000: L     4
      T     MW     3
      BEU
m001: L     5
      T     MW     3
      BEU
m002: L     6
      T     MW     3
      BE
```

حال با ایجاد خطا در شبیه سازی میتوانیم آدرس بیس و شماره اسلات ماژول معیوب را در MW1 و MW3 مشاهده کنیم.


S7-PLCSIM1 SIMATIC 300 Station\CPU313 C(1)
File Edit View Insert PLC Execute Tools Window Help
PLCSIM(MPI)
Key Switch Position
Startup Switch Position
Scan Mode
Next Scan F9
Pause F7
Automatic Timers
Manual Timers
Reset Timers...
Trigger Error OB
Scan Cycle Monitoring...
Hardware Interrupt (OB40-OB47)...
I/O Redundancy Error (OB70)...
CPU Redundancy Error (OB72)...
Communication Redundancy Error (OB73)...
Time Error (OB80)...
Diagnostic Interrupt (OB82)...
Insert/Remove Module Interrupt (OB83)...
Priority Class Error (OB85)...
Rack Failure (OB86)...
CPU
SF
DP
DC
RUN
STOP
RUN-P
RUN
STOP
MRES
Accumulators
Address Regist


Diagnostic Interrupt OB (82)
Module address I 4.0
Default Tests : User defined
Fault Condition
Module defective
Internal fault
External fault
Channel fault
External voltage failed
Front panel connector unplugged
Module not configured
Incorrect parameters on module
Channel information exists
User information exists
Diagnostic interrupt from substitute
Submodule missing or has an error
Communication problem
Operating mode is stop
Watchdog timer responded
Internal power supply failed
Battery exhausted
Entire backup failed
Expansion rack failure
Processor failure
EPROM fault
RAM fault
ADC/DAC error
Fuse tripped
Hardware interrupt lost
OK
Apply
Cancel
Help

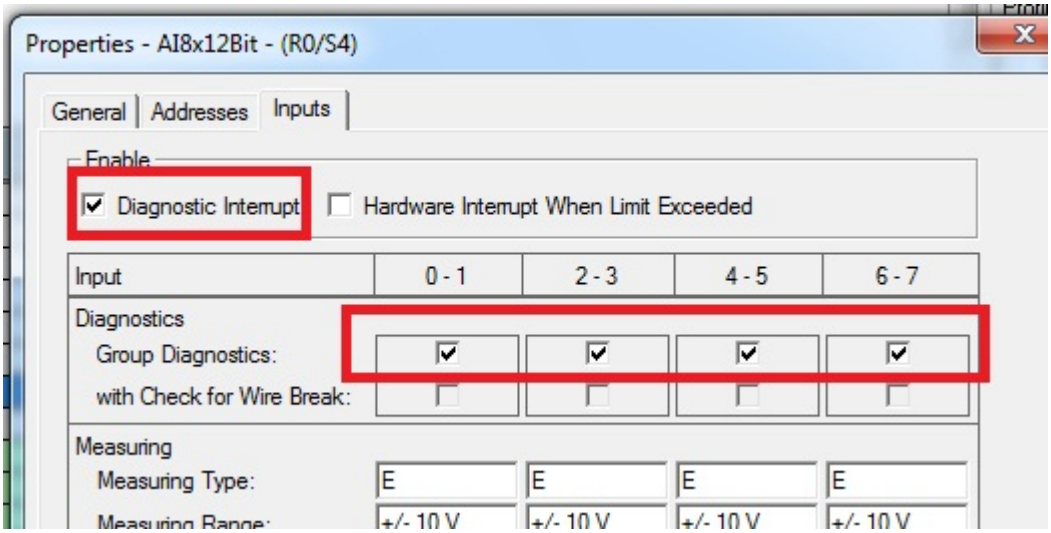
آدرس پایه ماژول در MW1 و شماره اسلات آن در MW3 قابل مشاهده است.

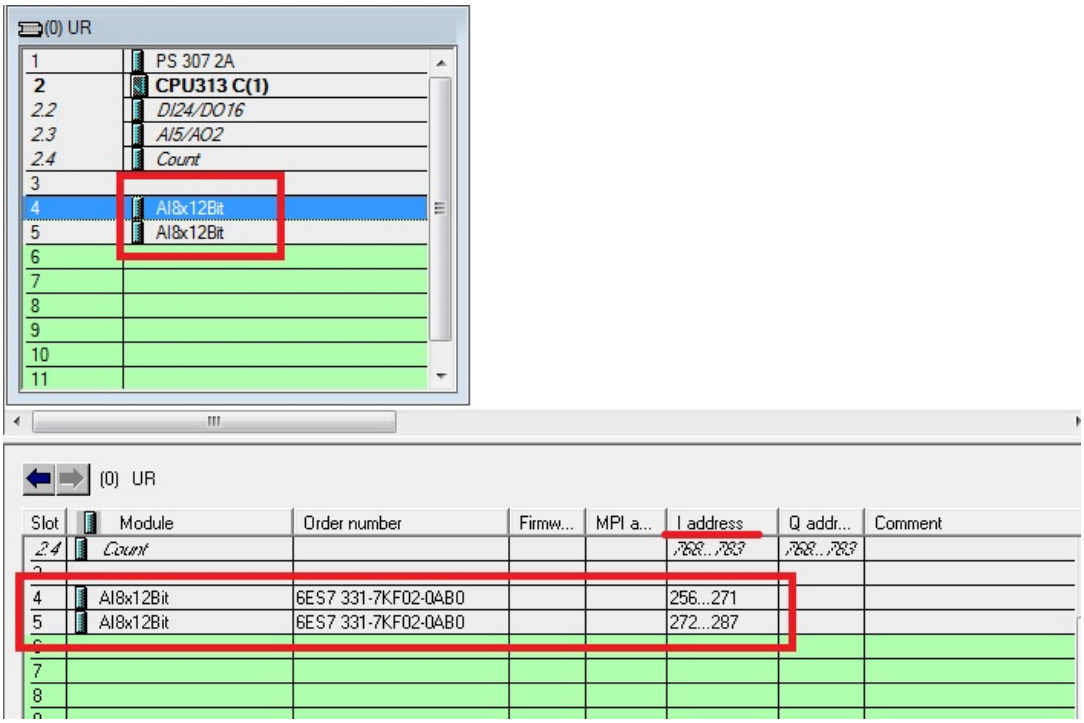


مثال برای OB82

دو عدد کارت AI داریم. اگر خطایی در کارت اول اتفاق افتاد خروجی Q0.0 فعال شود و برای کارت دوم Q0.1و همچنین آدرس شروع کارت در MW20 نمایش داده شود.

دو کارت ورودی آنالوگ با قابلیت Diagnostic در محیط سخت افزاری میاوریم. مثلا AI8x12Bit سپس قابلیت Diagnostic را برای هر دو کارت فعال می کنیم.



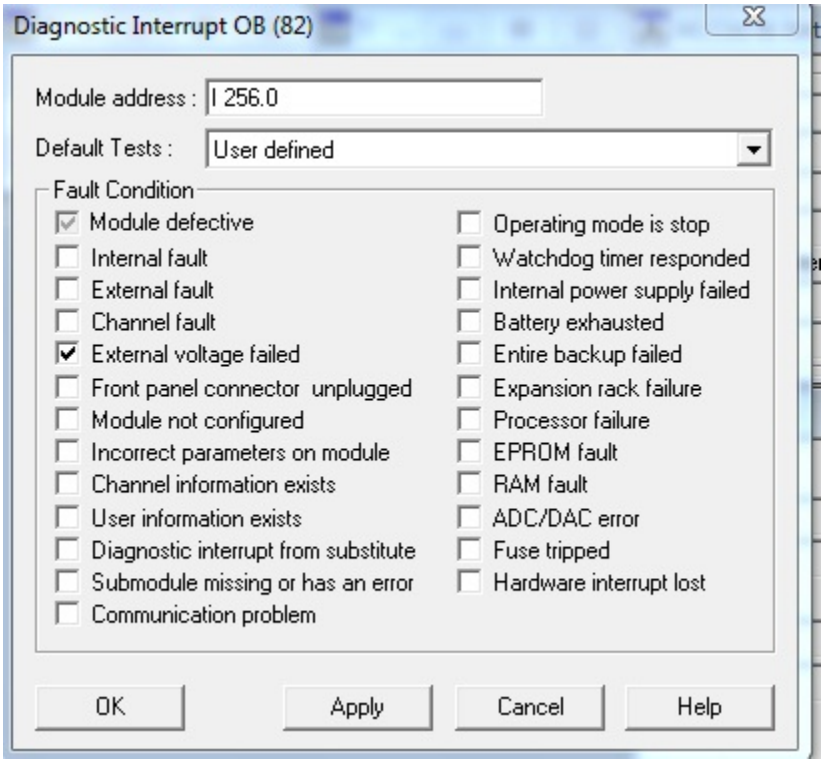


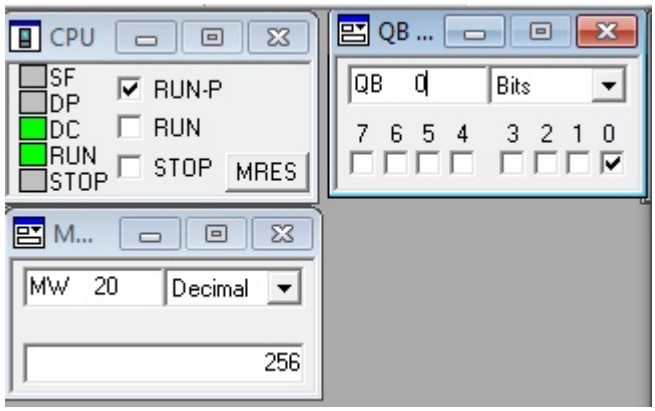
همانطور که ملاحضه میکنید آدرس های مربوط به هر کارت در قسمت پایین تصویر مشخص شده است.

حالا برنامه مربوطه را در OB82 مینویسیم.

```
OB82 :  "I/O Point Fault"
Comment:

Network 1: Title:
Comment:

      L     #OB82_MDL_ADDR
      T     MW    20
      L     MW    20
      L     256
      ==I
      JC    m000
      L     MW    20
      L     272
      ==I
      JC    m001
      BEU
m000: SET
      =     Q     0.0
      BEU
m001: SET
      =     Q     0.1
      BE
```

با ایجاد خطا بر روی هر یک از کارت ها خروجی مربوط به آن کارت فعال و آدرس پایه در MW20 نمایش داده میشود.

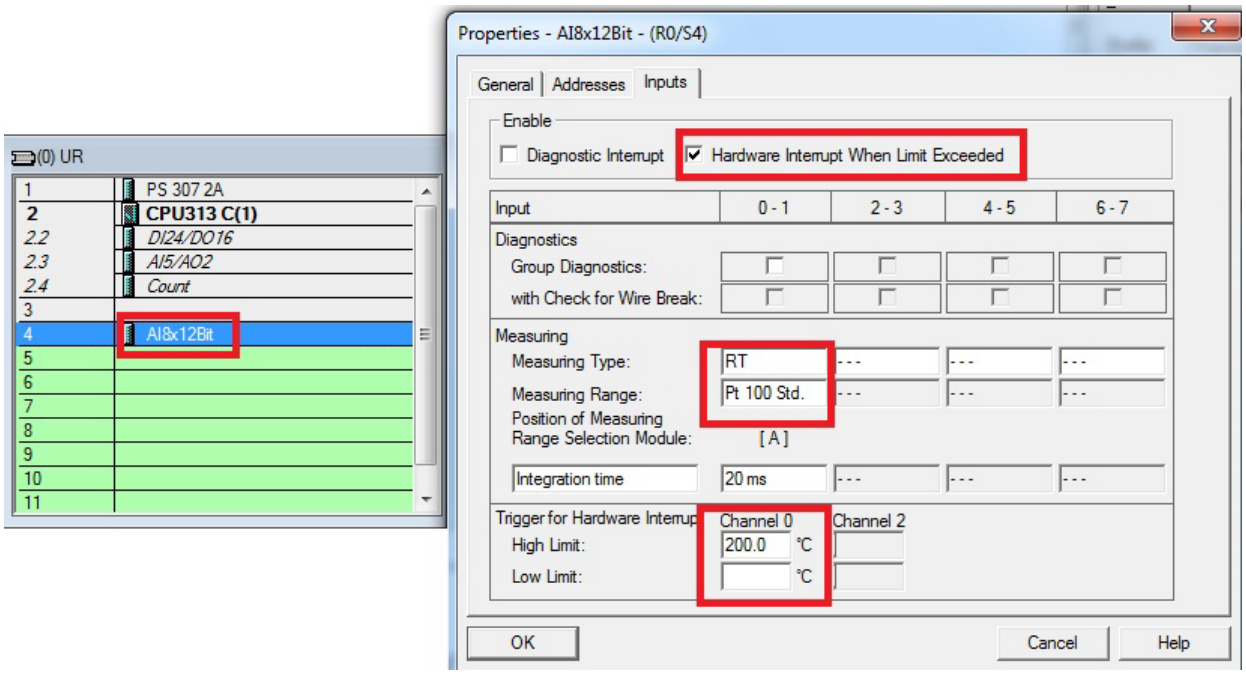




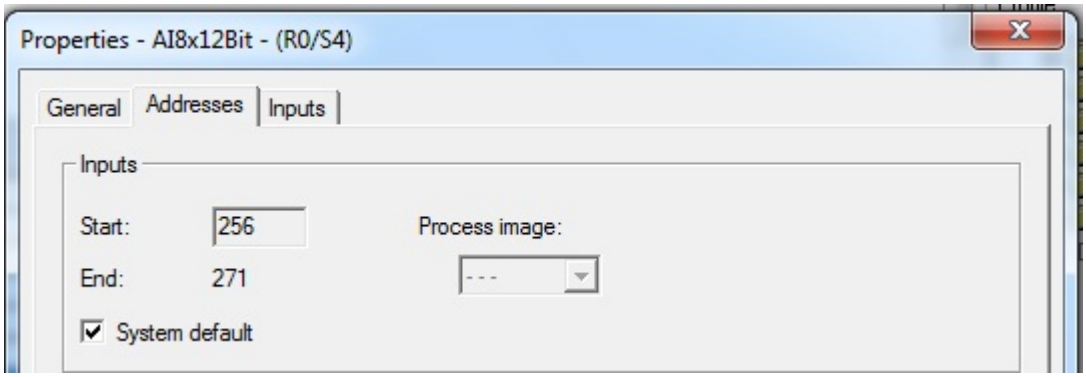
مثال برای OB40

با استفاده از قابلیت Hardware Interrupt برنامه ای بنویسید که اگر دمای حس شده توسط یک PT100 بیشتر از 200 درجه شد خروجی Q0.0 فعال شود:

یک کارت ورودی آنالوگ با قابلیت فوق می آوریم و تنظیمات آن را انجام میدهیم.

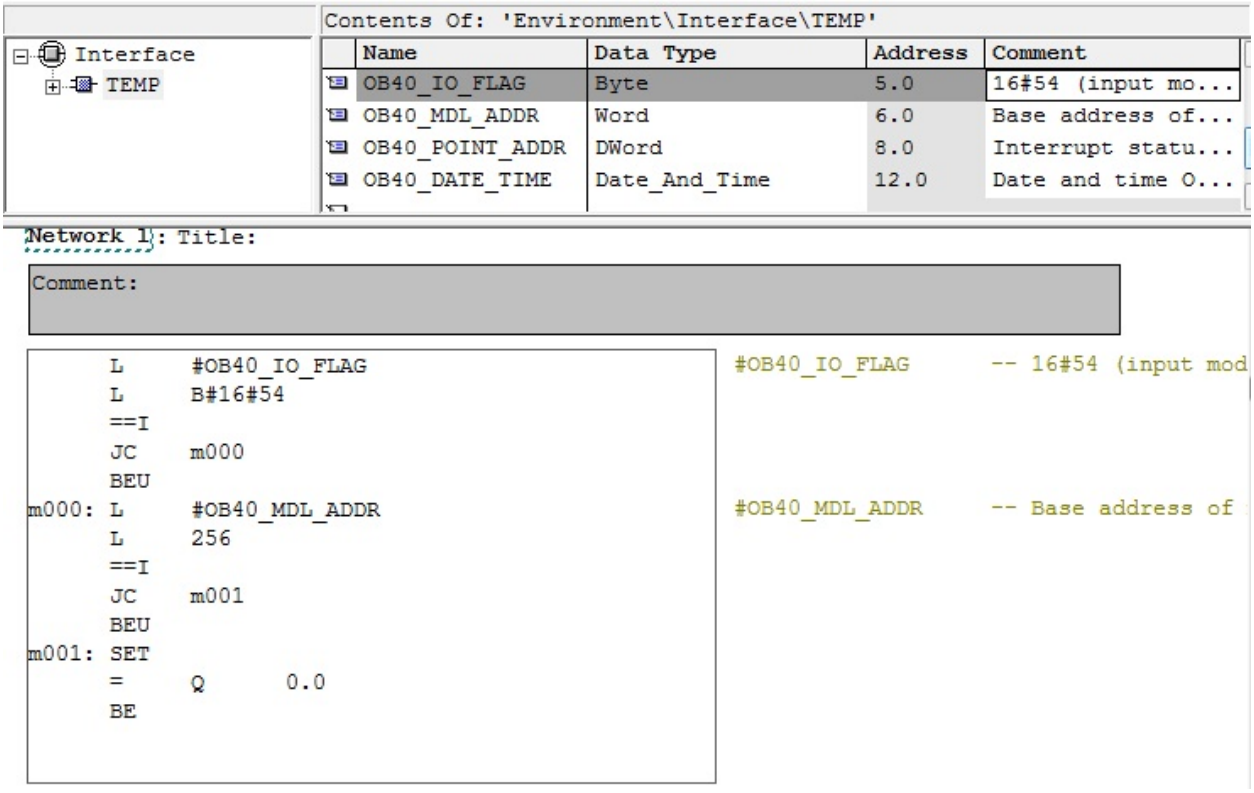


آدرس شروع کارت را بخاطر میسپاریم

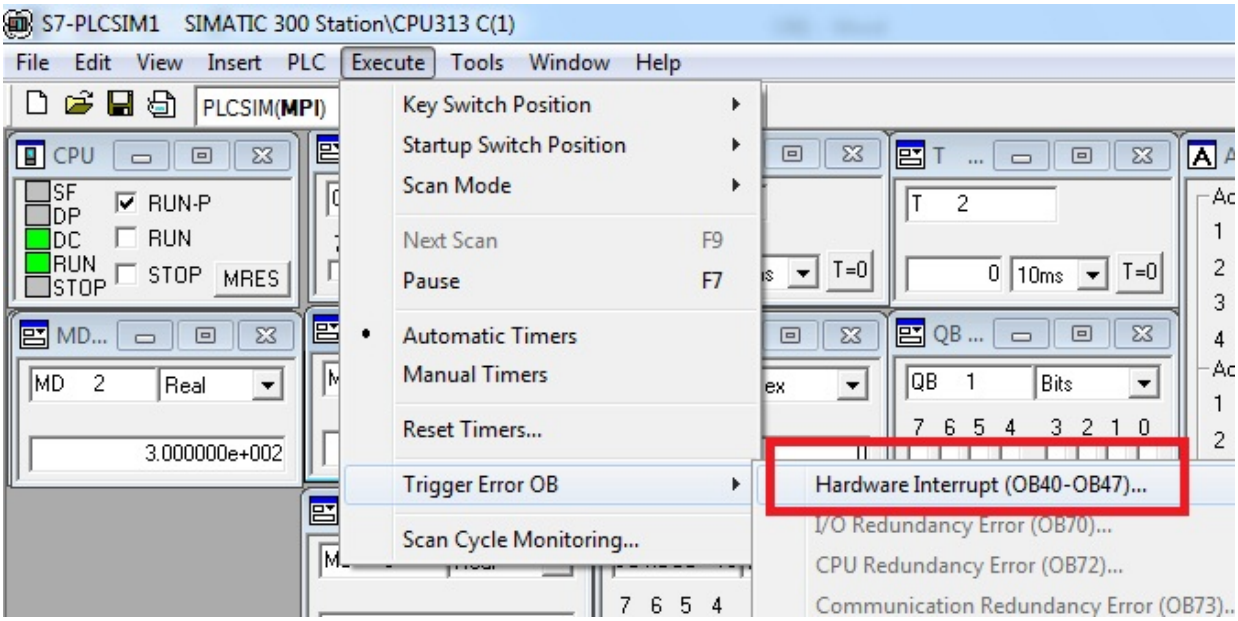


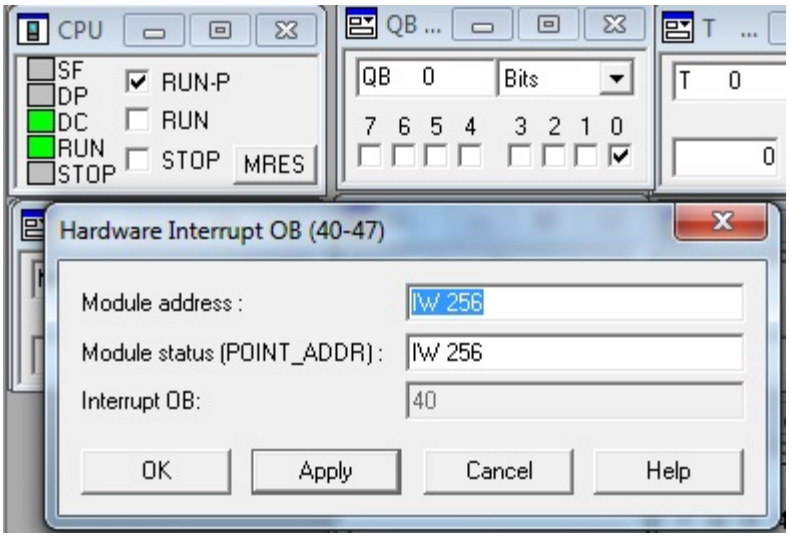
برنامه مورد نظر را در OB40مینویسیم. برنامه را با فرض اینکه سایر تجهیزات میتوانند وقفه ایجاد کنند به نحوی مینویسیم که فقط در صورتی که این کارت ورودی آنالوگ وقفه را ایجاد کند خروجی فعال شود.

ابتدا نوع ورودی و خروجی بودن را شرط ادامه برنامه و سپس آدرس پایه کارت را شرط قرار داده ایم.



برای تست برنامه باید وقفه سخت افزاری را ایجاد نماییم




CPU
SF
DP
DC
RUN
STOP
RUN-P
RUN
STOP
MRES
QB ...
QB 0
Bits
7 6 5 4 3 2 1 0
T ...
T 0
0
Hardware Interrupt OB (40-47)
Module address :
IW 256
Module status (POINT_ADDR) :
IW 256
Interrupt OB:
40
OK
Apply
Cancel
Help

بلوکهای سیستمی

بلوک های سیستمی توابع ای هستند که توسط برنامه نویسان نرم افزار برای کاربرد های خاصی آماده شده اند. این بلوک های سیستمی به صورت کتاب خانه در نرم افزار قرار داده شده است تا کاربران بسته به نیاز از آن ها استفاده کنند.

توابعی مانند تایمر های مختلف که مقادیر بیشتر از 2H46M30S میتوانند شمارش کنند و همچنین کانتر هایی 32 بیتی که مقادیر بیشتر از 999 را میشمارند. همچنین بلوک های برای کاربرد های مختلف در شبکه از قبیل ارسال و دریافت دیتا بین مستر و اسلیو و آی اسلیو ها و بسیاری از بلوک های خاص برای کاربرد های خاص مانند کنترل کننده های PID و ... .

بلوک های سیستمی به دو دسته تقسیم بندی شده است

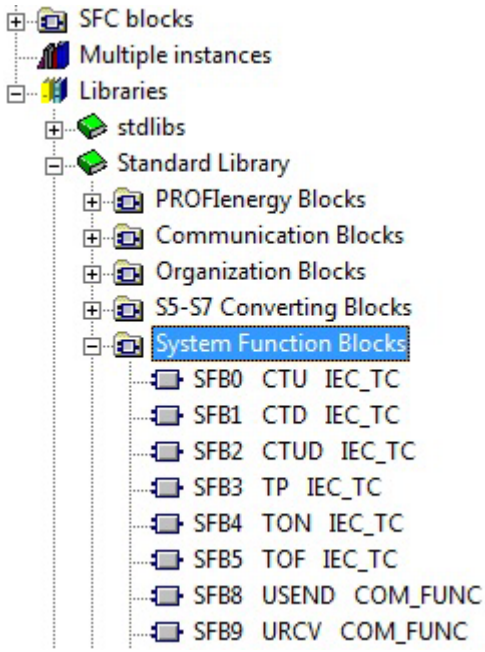
- (System Function Block) SFB
- (System Function) SFC

مانند آنچه در مورد بلوک های برنامه نویسی FB و FC داشتیم بلوک های SFB دارای حافظه هستند و با اضافه کردن آنها باید یک DB اختصاصی برای آنها در نظر بگیریم. همچنین نیاز نیست تا همه ی پایه های ورودی و خروجی این بلوک ها آدرس دهی شوند زیرا از طریق DB اختصاصی میتوان مقادیر مورد نیاز را وارد کرد.
بلوک های SFC بدون حافظه هستند و تمامی پایه های ورودی و خروجی باید آدرس دهی شوند.

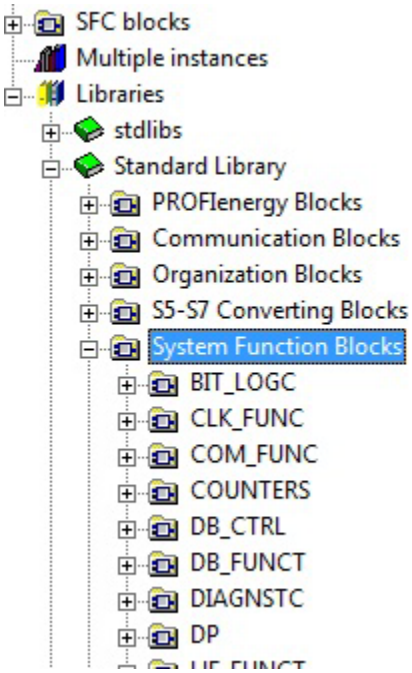
این بلوک ها به صورت Protect شده هستند و کاربران نمیتوانند در آنها تغییر ایجاد کنند یا برنامه آنهارا مشاهده کنند، فقط امکان استفاده یا فراخانی این بلوک ها وجود دارد.

بلوک های سیستمی در کتابخانه یا Library برنامه قرار دارند که از مسیر زیر میتوان انواع این بلوک هارا مشاهده و از آنها استفاده نمود.

Libraries > Standard Library > System Function Blocks

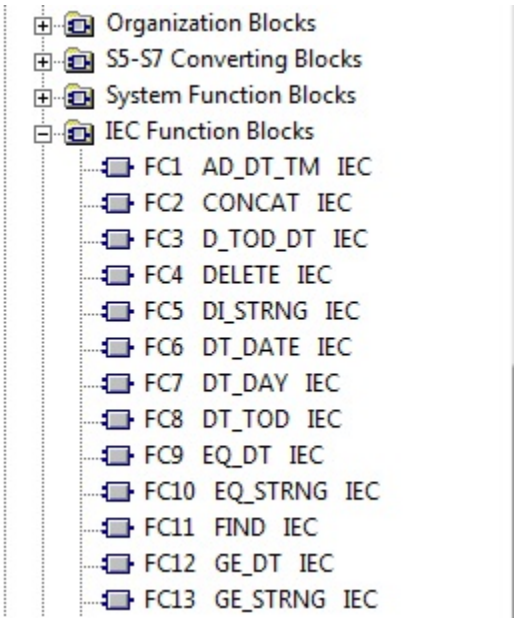


در این قسمت لیست فانکشن های موجود قابل مشاهده است. میتوانیم برای مشاهده به صورت گروه بندی شده روی فولدر System Function Block کلیک راست کرده و از قسمت Sort Library Block By گزینه Block Family را انتخاب میکنیم.



مشاهده میکنیم که بلوک های سیستمی بسته به نوع عملکرد داخل فولدر های مرتبط قرار میگیرند.

بلوک های پرکاربرد دیگر که در قسمت Standard Libraries موجود است در قسمت IEC Function Block لیست شده اند. از جمله پرکاربرد ترین فانکشن ها در این قسمت بلوک های مربوط به تنظیمات فرمت تاریخ و زمان است و در ادامه با برخی از آنها آشنا میشویم.



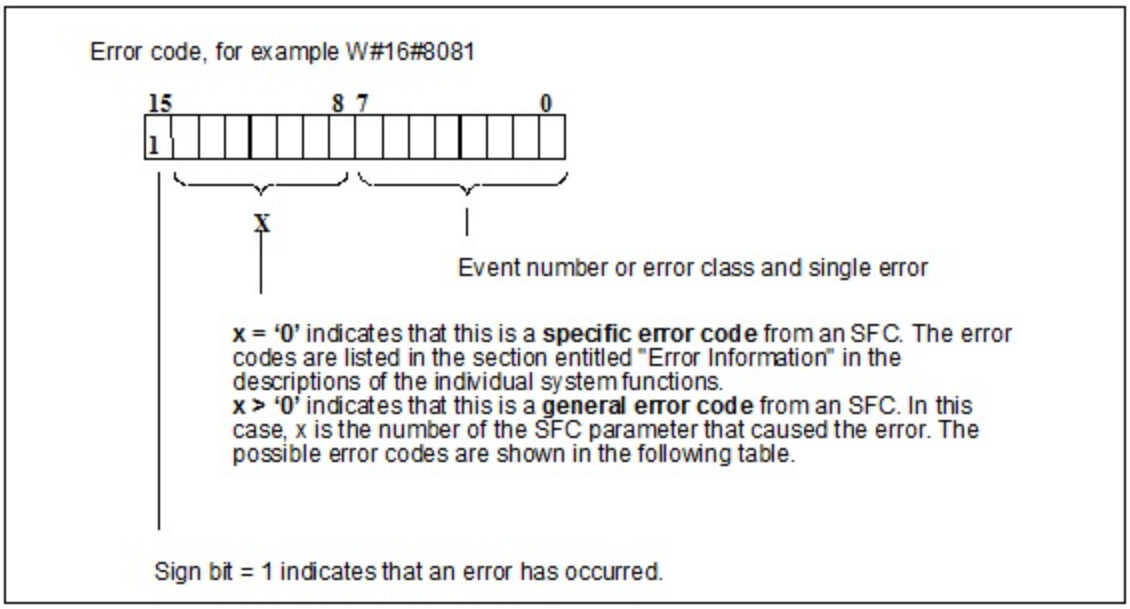
SFC ها (فانکشن های سیستمی)

این فانکشن ها به صورت FC برنامه نویسی شده اند و بدون حافظه هستند.

در تمامی SFC ها یک مقدار بازگشت یا Return Value وجود دارد که با RET_VAL در خروجی بلاک ها نمایش داده شده است. وظیفه این مقدار بازگشتی ارسال کد خطای پیش آماده در بلوک است که در یک ممووری ورد که توسط کاربر آدرس دهی میشود، ذخیره میشود.

خطاها دو نوع عمومی و اختصای دارند. نوع عمومی برای همه SFC ها است و نوع اختصاص مربوط به یک نوع خاص از SFC است. کد خطا به صورت Hex نمایش داده میشود و هر کد بیان گر خطایی خاص است که در تصویر مشاهده میکنیم.

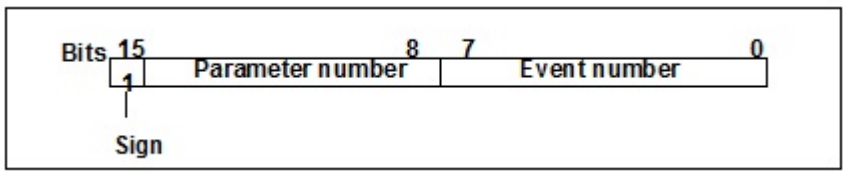
در صورتی که خطایی اتفاق افتد چه خطا نوع عمومی باشد و چه نوع خصوصی، بیت شماره 15 از RET_Val برابر با 1 منطقی میشود.



مطابق آنچه در Help برنامه مربوط به RET_VAL برای SFC ها آورده شده مشاهد میکنیم که که هفت بیت مشخص شده با X دو حالت کلی میتواند داشته باشد که نمایانگر نوع خطا است.

- **خطای عمومی**

اگر X عددی بزرگتر از 0 داشته باشد نمایشگر یک خطای عمومی است. در این صورت این قسمت که Parameter Number نام دارد عددی بین 1 تا 111 است و آن عدد، شماره پارامتری از SFC را نمایش میدهد که باعث ایجاد خطا شده است. بایت با ارزش کمتر مشخص کننده نوع رویداد است که عدد بین 0 تا 127 دارد و مطابق جدول نوع خطا را مشخص میکند.

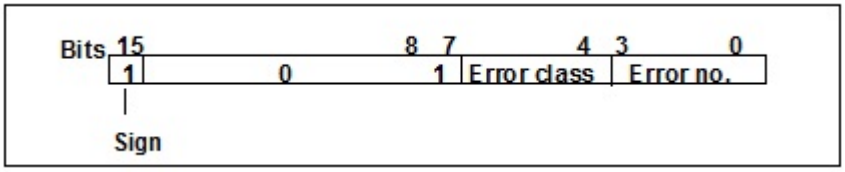


در جدول زیر کد خطاهای عمومی لیست شده است. توجه داشته باشید که بجای x در کد های زیر عدد پارامتری از SFC که خطا را ایجاد کرده است قرار میگیرد.

Error Code (W#16#...)	Explanation
8x7F	Internal error This error code indicates an internal error at parameter x.
8x01	Illegal syntax ID at an ANYparameter
8x22	Range length error when reading a parameter.
8x23	Range length error when writing a parameter. This error code indicates that the parameter x is located either entirely or partly outside the range of an address or that the length of a bit range is not a multiple of 8 with an ANY parameter.
8x24	Range error when reading a parameter.
8x25	Range error when writing a parameter. This error code indicates that the parameter x is located in a range that is illegal for the system function. Refer to the descriptions of the individual functions for information about the illegal ranges.
8x24 8x25	Range error when reading a parameter. Range error when writing a parameter. This error code indicates that the parameter x is located in a range that is illegal for the system function. Refer to the descriptions of the individual functions for information about the illegal ranges.
8x26	The parameter contains a timer number that is too high. This error code indicates that the timer specified in parameter x does not exist.
8x27	The parameter contains a counter number that is too high (counter number error). This error code indicates that the counter specified in parameter x does not exist.
8x28 8x29	Alignment error when reading a parameter. Alignment error when writing a parameter. This error code indicates that the reference to parameter x is a bit address that is not equal to 0.
8x30 8x31	The parameter is located in a read-only global DB. The parameter is located in a read-only instance DB. This error code indicates that parameter x is located in a read-only data block. If the data block was opened by the system function itself, the system function always returns the value W#16#8x30.
8x32 8x34 8x35	The parameter contains a DB number that is too high (DB number error). The parameter contains an FC number that is too high (FC number error). The parameter contains an FB number that is too high (FB number error). This error code indicates that parameter x contains a block number higher than the highest permitted number.
8x3A 8x3C 8x3E	The parameter contains the number of a DB that is not loaded. The parameter contains the number of an FC that is not loaded. The parameter contains the number of an FB that is not loaded.
8x42	An access error occurred while the system was attempting to read a parameter from the peripheral input area.
8x43	An access error occurred while the system was attempting to write a parameter to the peripheral output area.
8x44	Error in the nth (n > 1) read access after an error occurred.
8x45	Error in the nth (n > 1) write access after an error occurred. This error code indicates that access to the required parameter is denied.

- **خطای اختصاصی**

اگر مقدار X برابر با 0 باشد، بیانگرخطای خاص مربوط به SFC خاص است. خطاهای اختصاصی برای هر SFC از قسمت Help مربوط به آن و در Error Information مشخص میشود.



در این حالت بیت شماره 7 برابر با 1 میشود و سه بیت مربوط به Error class عددی بین 0 تا 7 است و همچنین 4 بیت آخر که Error no است شماره خطا را مشخص میکند و عددی بین 0 تا 15 است. همانطور که گفته شد این خطا برای هر SFC متفاوت است و با توجه به Error Information موجود در Help آن، نوع خطا مشخص میشود.

معرفی برخی از SFC های پرکاربرد

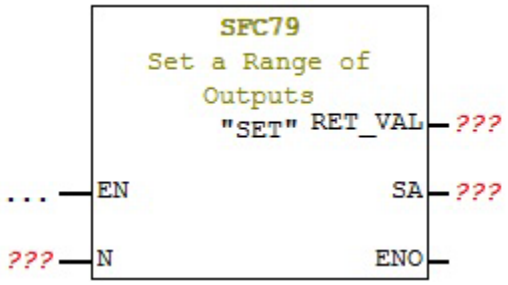
در این قسمت برخی از SFC های پرکاربرد به ترتیب خانواده آنها بررسی میشود.

BIT_LOGK

در این قسمت دو بلوک سیستمی برای ست و ریست کردن خروجی (Q) مورد استفاده قرار میگیرد. ویژگی این بلوک ست و ریست کردن مقادیر بیش از 32 بیت است که در یک سیکل انجام میدهد.

(SET BIT_LOGK) SFC 79

برای ست کردن تعداد بیت مشخص شده خروجی مورد استفاده قرار میگیرد.



EN : ورودی فعال سازی بلوک

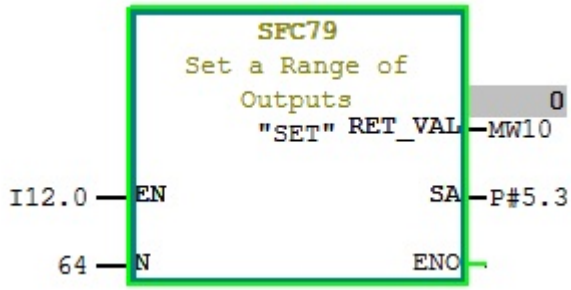
N : تعداد بیست هایی از خروجی که می خواهیم ست شود را وارد میکنیم.

RET_VAL : یک مموری ورد برای ذخیره کد خطا مشخص میکنیم.

SA : آدرس بیت شروع را به صورت Pointer با فرمت P#m.n که m شماره بایت و n شماره بیت است مشخص میکنیم.

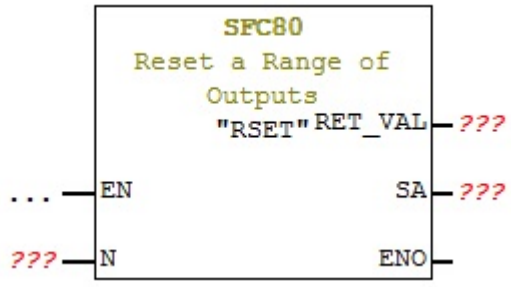
ENO : خروجی بیتی نشانگر فعالیت بلوک سیستمی

برای مثال فرض کنید میخواهیم تعداد 64 خروجی را از آدرس شروع Q5.3 ست کنیم. داریم:



(RSET BIT_LOGK) SFC 80

برای ریست کردن تعداد بیت مشخص شده خروجی کاربرد دارد.



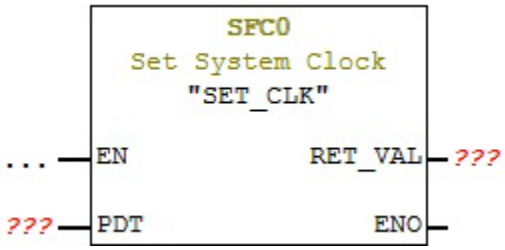
پارامترهای این بلوک دقیقا مانند بلوک قبلی است. این دو خطای اختصاصی ندارند.

CLK_FUNC

این فانکشن ها که Clock Function نام دلرند مربوط به زمان و زماندهی و زمان سنجی است.

(SET_CLK CLK_FUNC) SFC 0

توسط این بلوک زمان و تاریخ مورد نظر را به CPU اعمال میکنیم. البته راه حل ساده تر برای زمان دهی به CPU ممکن است که در ادامه بیان خواهیم کرد.



EN : ورودی فعال سازی بلوک

PDT : برای وارد کردن زمان و تاریخ مورد نظر است .

RET_VAL : یک ممووری ورد برای ذخیره کد خطا

ENO : خروجی بیتی نشانگر فعالیت بلوک سیستمی

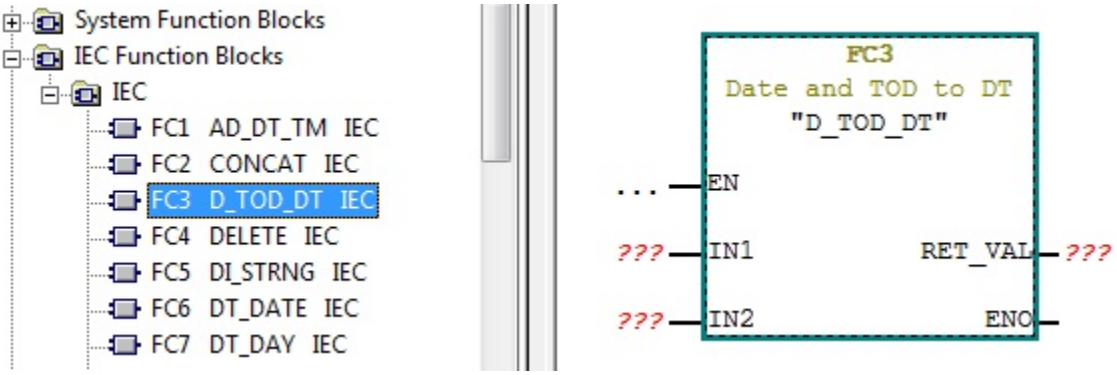
فرمت زمان و تاریخ، به صورت DT#1995-02-02-08:30:01.999 با نام DateAndTime است که 8 بایت یا 64 بیتی است و با روش معمول به صورت مستقیم نمیتوان زمان و تاریخ را به وارد کرد. برای اینکار نیاز به بلوک های قسمت IEC Function Blocks داریم تا فرمت زمان و فرمت تاریخ را در هم ادغام کند و در آخر فرمت مورد نظر یعنی DateAndTime را در 8 بایت ذخیره کند.

The DATE_AND_TIME data type is stored in BCD format:

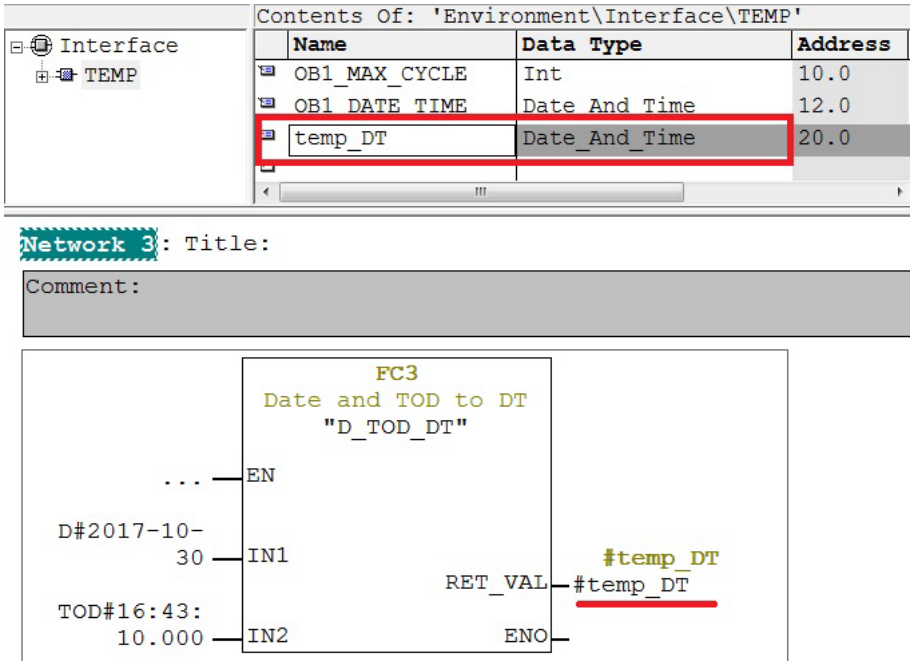
Byte	Contents	Range
0	Year	1990 to 2089
1	Month	01 to 12
2	Day	1 to 31
3	Hour	0 to 23
4	Minute	0 to 59
5	Second	0 to 59
6	2 MSD of ms	00 to 99
7 (4 MSB)	LSD of ms	0 to 9
7 (4 LSB)	Day of week	1 to 7 (1 = Sunday)

نحوه ی قسمت بندی این 8 بایت را در جدول بالا مشاهده میکنید.

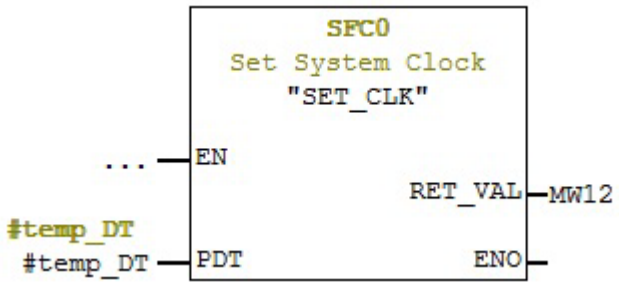
توسط بلوک FC3 در قسمت IEC Function Blocks ورودی مورد نیاز PDT را میسازیم. فرمت تاریخ را در ورودی IN1 به صورت D#yyyy-mm-dd (D#2017-10-30) و فرمت زمان روز را به ورودی IN2 به صورت TOD#hh:mm:ss.mmm (TOD#16:43:00.000) می دهیم و خروجی را در حافظه Temp ذخیره میکنیم.



برای این کار لازم است در Temp سطر با فرمت DATE_AND_TIME ایجاد کرده و نام گذاری کنیم. سپس نام متغیر ایجاد شده را برای خروجی این FC3 قرار دهیم.



بعد از این کار این حافظه Temp ساخته شده را برای ورودی PDT برای SFC 0 قرار میدهیم.

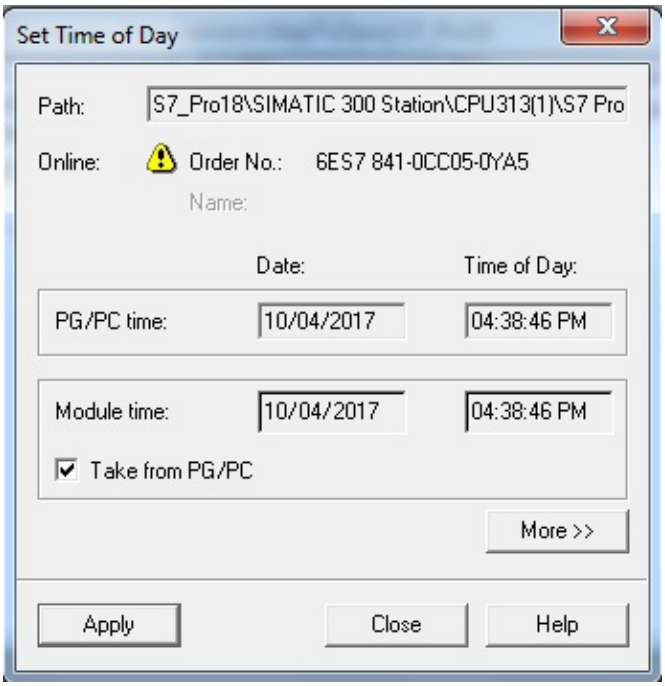


استفاده از این بلوک در OB1 موجب میشود مرتبا تاریخ و زمان CPUبه مقادیر فوق عوض شود و باید به صورت کنترل شده در بلوک های startup یا در صورت بروز وقفه باشد که باعث پیچیدگی بیشتر آن میشود و یا توسط یک ورودی برای پایه EN کنترل انجام شود.

خطای اختصاصی

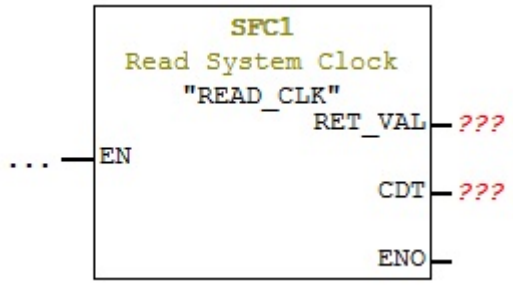
8080	Error in date
8081	Error in time

بهترین روش برای تنظیم کردن تاریخ و ساعت CPU استفاده از امکان فراهم شده در نرم افزار Simatic manager است. کافی است در محیط نرم افزار وارد منوی PLC شویم و از قسمت Diagnostic/setting گزینه ی Set time of day را انتخاب کنیم.



SFC1 (Read_CLK CLK_FUNC)

توسط این فانکشن امکان خواندن زمان و تاریخ CPU فراهم میشود و از این قابلیت در برنامه تعمیرات و نگهداری، سرویس دستگاهها و... میتوان استفاده کرد .در کل توسط این بلوک و بلوک های مقایسه گر میتوانیم در زمان های مشخص اقداماتی را انجام دهیم.این بلوک خطای اختصاصی ندارد.



EN : فعال ساز بلوک

RET_VAL : یک ممووری ورد برای ذخیره کد خطا

CDT : خروجی جهت نمایش زمان و تاریخ CPU به فرمت #DT

ENO : خروجی بیتی نشانگر فعالیت بلوک سیستمی

نکاتی که در مورد ورودی PDT در بلوک SFC0 بیان شد برای این پارامتر نیز صادق است.

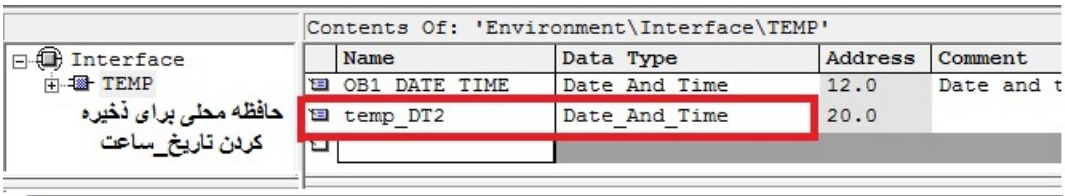
یک حافظه Temp با تایپ DATE_AND_TIME ایجاد و برای خروجی CDT تنظیم میکنیم. اطلاعات زمان و تاریخ را در CDT با استفاده از روش های برنامه نویسی ویا بلوک های IEC Function Blocks استخراج نمود.

برای استخراج مقدار زمان و تاریخ از حافظه Temp موجود برای خروجی CDT، از بلوک های FC6 (DT_DATE) و FC7 (DT_DAY) و FC8 (DT_TOD) در IEC Function block میتوانیم استفاده کنیم.

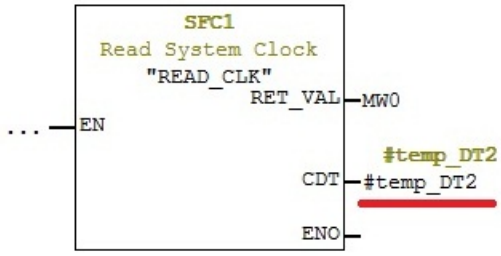
خروجی این بلوک ها RET_VAL است و برای تاریخ (FC6) و روز (FC7) خروجی به تایپ Word و برای زمان_روز (FC8) خروجی به صورت دابل ورد است.

توجه داشته باشید که پارامتر RET_VAL مخفف شده ی Return Value در بلوک های IEC Function Blocks خروجی است و مانند پارامتر RET_VAL در System Function Blocks کد خطا نیست.

با استفاده از خروجی های تفکیک شده ی پارامتر CDT توسط بلوک های FC6 و FC7 و FC8 میتوانیم در برنامه نویسی برای انجام عملکرد مورد نظر در تاریخ و زمان مشخص استفاده کنیم. کافی است تاریخ و ساعت مشخص را توسط سیمولیشن برنامه وارد کنیم و کد اینتیجر و دابل اینتیجر آنها را استراج کرده و توسط مقایسه گر ها برنامه شرطی مورد نظر را بنویسیم.

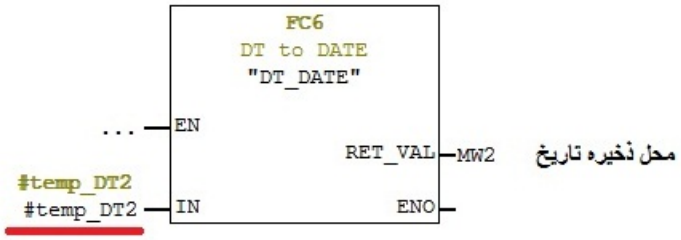


Comment:



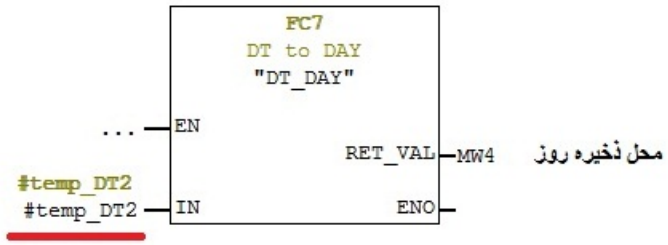
Network 2 : Title:

Comment:



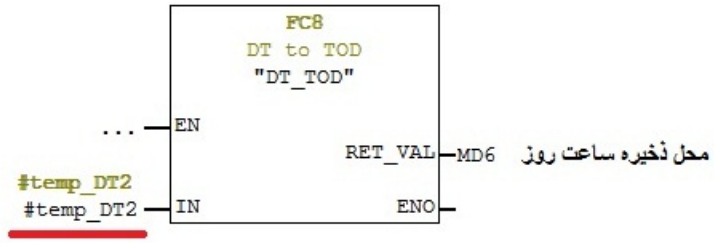
Network 3 : t

Comment:

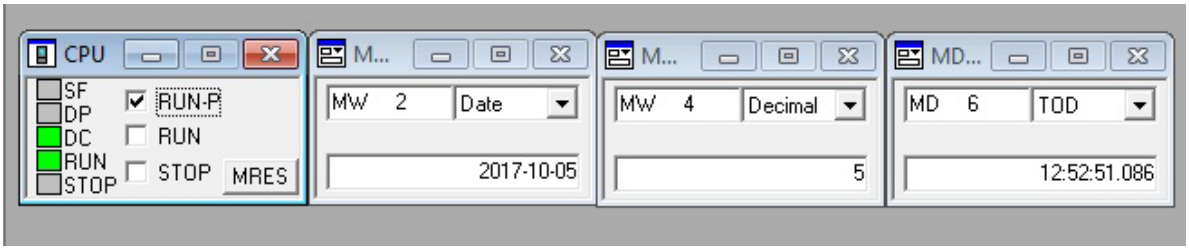


Network 4: Title:

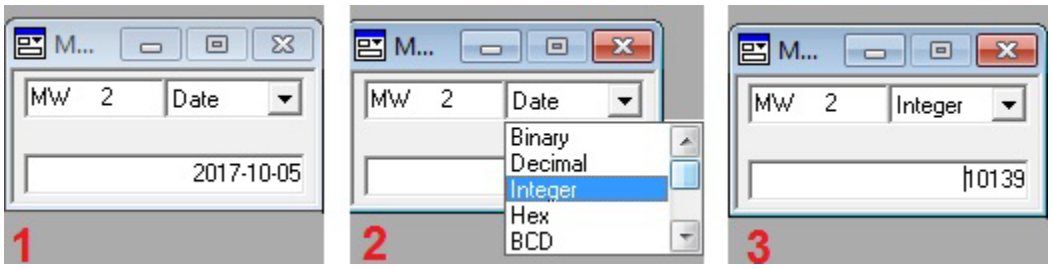
Comment:



با دانلود تمامی بلاک ها در سیمولیشن ملاحظه میکنیم:



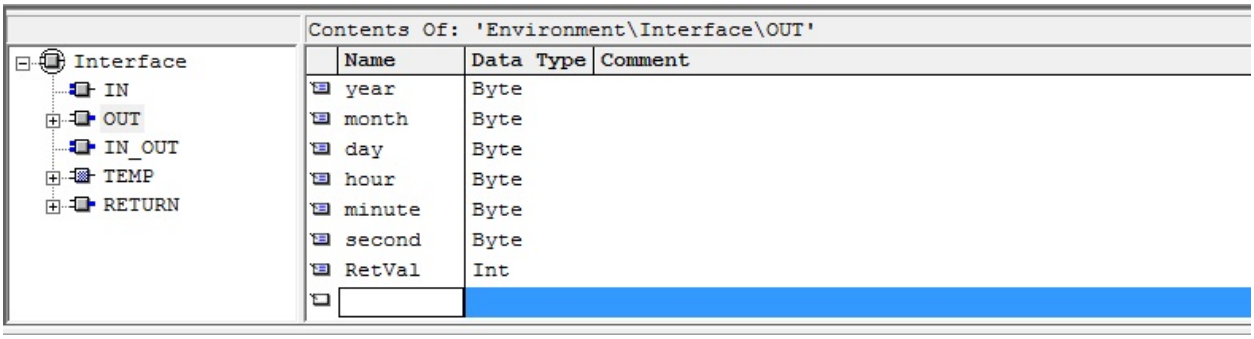
برای بدست آوردن کد اینتیجر و دابل اینتیجر مقادیر فوق مطابق مراحل شکل زیر عمل میکنیم.



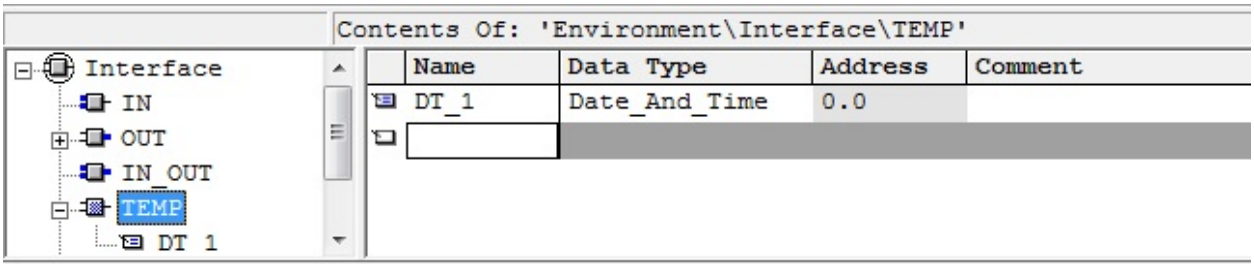
روش دیگر برای تفکیک پارامتر های CDT استفاده از روش برنامه نویسی با آدرس دهی غیر مستقیم است.
با ذکر یک مثال نحوه ی برنامه نویسی این مورد بیان میکنیم.

مثال: میخواهیم یک تابع برای ذخیره کردن مقادیر تاریخ‌وزمان در حافظه بنویسم و بعد مقادیر را در حافظه های زیر قرار دهیم. MB100 : سال ، MB101 : ماه ، MB102 : روز ، MB103 : ساعت ، MB104 : دقیقه MB105 : ثانیه

1.ساخت بلوک FC1 و تعریف پارامتر های مورد نظر در قسمت OUT



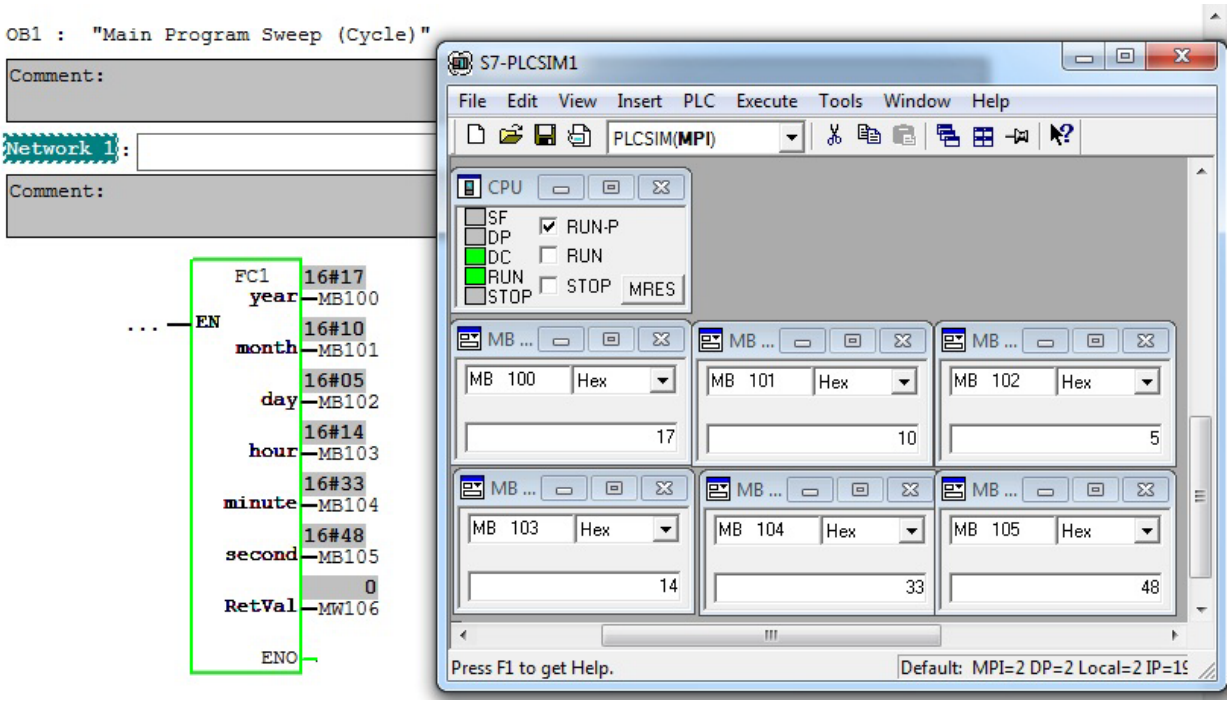
2. تعریف پارامتر Temp برای خروجی CDT با تایپ DATE_AND_TIME



3. نوشتن برنامه کنترلی جهت تفکیک پارامترهای سال ,ماه ,روز و... به روش آدرس دهی غیر مستقیم در بلوک FC1.

```
FC1 : Title:
Comment:

Network 1: Title:
Comment:

CALL  "READ_CLK"                SFC1              -- Read System Clock
 RET_VAL:=#RetVal               #RetVal
 CDT    :=#DT_1                 #DT_1
NOP   0
LAR1  P##DT_1
L     B [AR1,P#0.0]
T     #year                     #year
L     B [AR1,P#1.0]
T     #month                    #month
L     B [AR1,P#2.0]
T     #day                      #day
L     B [AR1,P#3.0]
T     #hour                     #hour
L     B [AR1,P#4.0]
T     #minute                   #minute
L     B [AR1,P#5.0]
T     #second                   #second
```

4. فراخوانی FC1 در بلوک OB1 و قرار دادن فضاهای حافظه

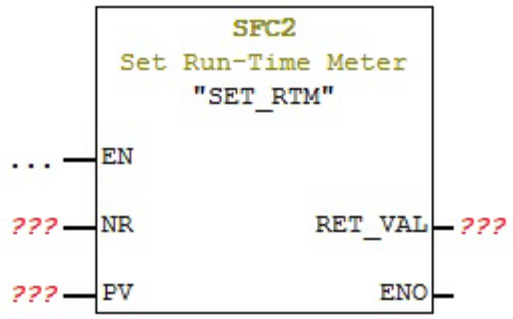


توجه داشته باشید که تمامی پارامترها با تایپ Hex عدد دقیق را بیان میکنند.

SFC2 (SET_RTM CLK_FUNC)

RTM مخفف شده ی Run Time Meter زمان سنج های داخل CPU هستند که میتواند مقدار ساعت کارکرد یک خروجی را در خود ذخیره کند. این ویژگی کار برد زیادی در تعمیرات و نگهداری دارد. حداکثر هشت RTM به صورت 16 بیتی وجود دارد و حداکثر 32767 ساعت را ثبت میکنند که میتوان 8 خروجی را به صورت جداگانه و همزمان زمان سنجی کند. هر چند این کار را با تایمر های سخت افزاری برای وسیله های بیشتری میتوان انجام داد ولی استفاده از این بلوک ها ساده تر است.

SFC2 برای SET کردن RTM استفاده میشود. توسط این بلوک فعال سازی یکی از هشت RTM موجود انجام میشود. در این بلوک تنها شماره RTM که قرار است ساعت کاری یک خروجی را ضبط کند و مقدار کارکرد قبلی آن خروجی داده میشود و مشخص کردن آدرس خروجی در بلوک سیستمی بعدی انجام میشود.



EN : فعال ساز بلوک

NR : این ورودی جهت تعیین شماره RTM است.. تایپ این ورودی بایت بوده و به صورت B#16#x که x در آن عدد بین 0 تا 7 است، باید وارد شود. مثلا B#16#3 به معنای استفاده از RTM شماره 3 است.

PV: در این ورودی مقدار اولیه RTM را به فرمت integer قرار میدهیم. مثلا اگر این مقدار 50 باشد به این معناست که وسیله مورد نظر 50 ساعت تاکنون کار کرده است.

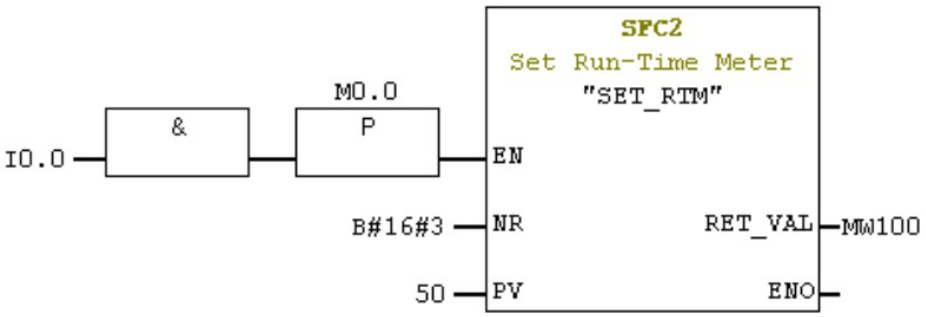
RET_VAL : یک ممووری ورد برای ذخیره کد خطا

ENO : خروجی بیتی نشانگر فعالیت بلوک سیستمی

خطای اختصاصی

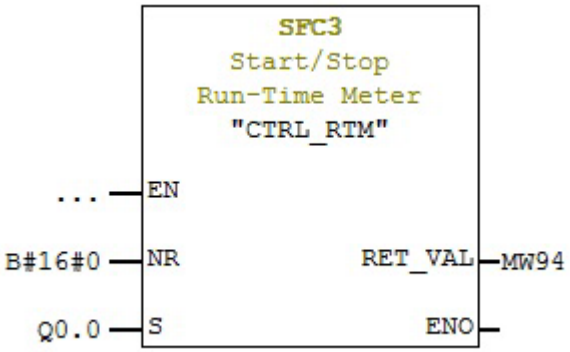
8080	Wrong number for the runtime meter
8081	A negative value was transferred to the PV parameter.

نکته قابل توجه در این بلوک این است که باید از پایه EN با یک لبه بالارونده استفاده شود چون در صورتی که این بلوک همواره فعال باشد مقادیر PV در هر سیکل در شماره RTM وارد شده قرار میگیرد و عملا این RTM هیچ مقداری را ضبط نمیکند و مرتبا مقدار PV در آن قرار میگیرد.



SFC3 (CTRL_RTM CLK_FUNC)

این فانکشن برای SET کردن شماره RTM که قبلا در SFC2 تنظیم شده بود به کار میرود. هنگامی که ورودی S در این بلوک مقدار 1 منطقی داشته باشد زمان سنج فعال میشود و هنگامی که 0 منطقی باشد زمان سنج متوقف میشود.



EN : فعال ساز بلوک

NR : ورودی برای تعیین شماره RTM

S : آدرس خروجی که قرار است مقدار کارکرد آن ثبت شود. اگر این پایه یک شود به معنای روشن بودن و اگر صفر باشد به معنای خاموشی . مثلا Q0.0

RET_VAL : یک ممووری ورد برای ذخیره کد خطا

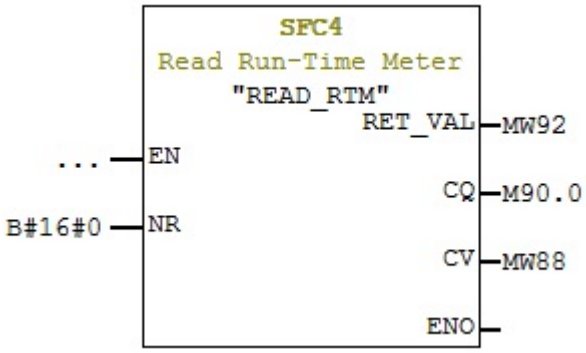
ENO : خروجی بیتی نشانگر فعالیت بلوک سیستمی

خطای اختصاصی

8080	Wrong number for the runtime meter

SFC4 (READ_RTM CLK_FUNC)

از SFC4 برای نمایش وضعیت RTM مورد نظر که قبلا توسط 2 فانکشن قبلی تنظیم و SET شده است استفاده میشود. با تعیین کردن شماره RTM میتوانیم از وضعیت تعداد ساعت شمارش کرده و فعالیت یا عدم فعالیت زمان سنج مطلع شویم.



EN : فعال ساز بلوک

NR : شماره RTM

CQ : برای نمایش فعال بودن RTM مورد نظر به صورت بیتی

CV : مقدار فعلی RTM را نمایش میدهد.(integer)

RET_VAL : یک ممووری ورد برای ذخیره کد خطا

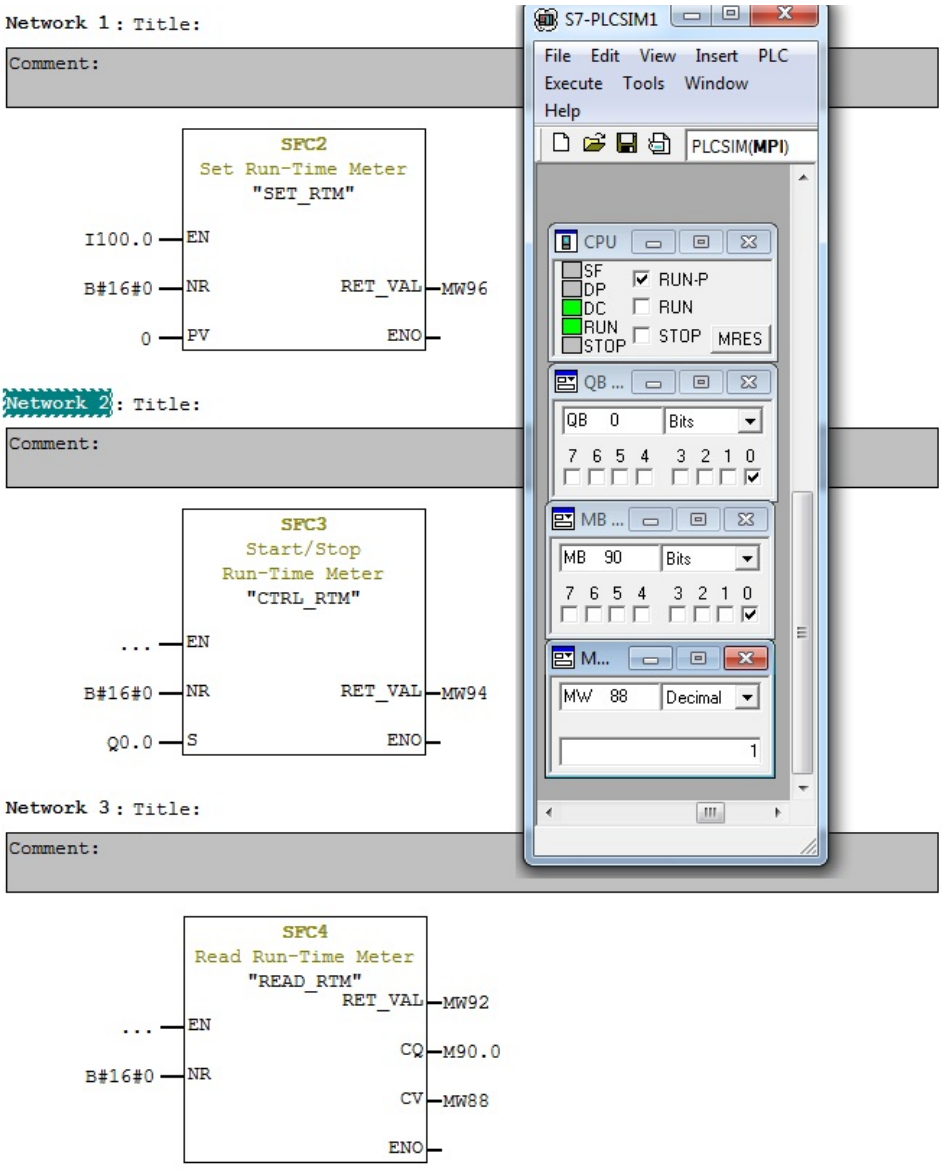
ENO : خروجی بیتی نشانگر فعالیت بلوک سیستمی

خطای اختصاصی

8080	Wrong number for the runtime meter
8081	Overflow of the runtime meter

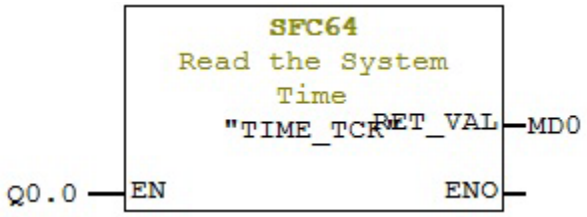
مثال : میزان کار کرد Q0.0 ثبت شود.

1- تنظیم RTM توسط SFC2
2- set کردن RTM با فانکشن SFC3
3- خواندن RTM با استفاده از SFC4
4- مشاهده نتیجه بعد از 1ساعت روشن بودن Q0.0 در جدول



(TIME_TCK CLCK_FUNC) SFC64

این بلوک یک کرنومتر با دقت یک میلی ثانیه است. تایپ خروجی این بلوک دابل ورد است و مقدار 0 تا حداکثر 2147483647 میلی ثانیه را شمارش میکند. در صورت وقوع سرریز، مقدار این شمارنده زمان صفر میشود. زمان سیستم تنها تحت تأثیر حالت های عملکرد CPU است. این بلوک هیچگونه خطایی ندارد.



EN : فعال ساز بلوک

RET_VAL : این بلوک سیستمی هیچگونه خطایی ندارد و RET_VAL خروجی بلاک با تایپ Dword برای نمایش زمان شمارش شده است.

ENO : خروجی بیتی نشانگر فعالیت بلوک سیستمی

برای ثبت زمان کاری یک خروجی، میتوانیم از پایه EN بلوک استفاده کنیم.

این بلوک با توجه به نوع راه اندازی CPU مقادیر قبلی خود را حفظ میکند یا صفر میشود.

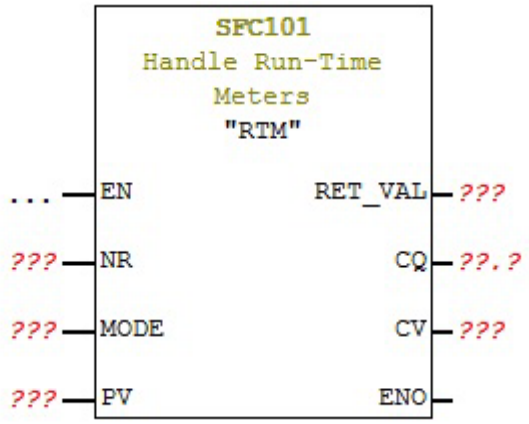
Mode	System Time
Startup	... is constantly updated
RUN	... is constantly updated
STOP	... is stopped and retains the current value
Hot restart (not with S7-300 and S7-400 H)	... continues with the value saved at the change to the STOP mode
Warm restart	... is deleted and restarts with "0"
Cold restart	

SFC101 (RTM CLCK_FUNC):

این فانکشن بلاک برای فعال کردن و استارت و استپ و خاندن مقدار RTM های 32 بیتی استفاده میشود.

RTM های 32 بیتی زمانی بیشتر از RTM های 16 بیتی ذخیره میکنند.

فانکشن های SFC2, SFC3و SFC4 دارای هشت RTM که هر کدام 16 بیتی میباشند و بازه شمارش انها ساعت 32768 است. اگر بخواهیم بازه شمارش بیشتری داشته باشیم باید از RTM های 32بیتی استفاده شود. بدین منظور باید از فانکشن SFC101 که دارای 16 (0....15) RTM میباشد استفاده کنیم. این فانکشن دارای تمامی امکانات تنظیم SET و خواندن RTM را دارد.



EN : فعال ساز بلوک

NR : در این ورودی شماره RTM که عددی بین 0 تا 15 است قرار میدهیم.

MODE : با قرار دادن عددی بین 0 تا 6 میتوان از مد های مختلف بهره برد.

- MODE 0 :
- MODE 1 : RTM استارت میشود.
- MODE 2 : توقف RTM
- MODE 4 : RTM با مقدار اولیه PV ست میشود.
- MODE 5 : RTM با مقدار اولیه PV ست میشود و سپس شروع به کار میکند.
- MODE 6 : RTM با مقدار اولیه PV ست میشود و سپس متوقف میشود.

PV : جهت وارد کردن مقدار اولیه RTM به صورت 32 بیتی.

CQ : در صورت فعال شدن RTM این خروجی به صورت بیتی یک میشود.

CV: این خروجی برای نمایش مقدار RTM به فرمت 32 بیتی میباشد

RET_VAL : یک ممووری ورد برای ذخیره کد خطا

ENO : خروجی بیتی نشانگر فعالیت بلوک سیستمی

خطای اختصاصی

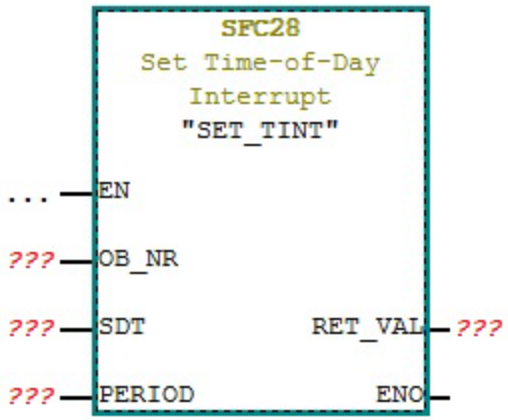
8080	Wrong number for the runtime meter
8081	A negative value was passed to parameter PV.
8082	Overflow of the runtime meter.
8091	Illegal value in input parameter MODE.

PGM_CNTL

در این قسمت فانکشن های Program Control شامل فانکشن های کنترلی وقفه ها و کنترلی مد CPU است که در ادامه بررسی میشوند.

SFC28 (SET_TINT PGM_CNTL)

ابن بلوک برای فعال سازی بلوک های وقفه Time Of Day Interrupt به صورت برنامه نویسی شده کاربرد دارد. همانطور که میدانیم این بلوک های سازمانی از OB10 تا OB17 شماره گذاری شده اند و بسته به نوع CPU یک یا تعداد بیشتری از این وقفه ها قابل استفاده هستند.



EN : فعال ساز بلوک

OB_Nr : شماره OB وقفه زمان روز که در CPU فعال است. شامل اعداد 10 تا 17

SDT : تاریخ و زمان شروع برای وقفه که به فرمت DATE_AND_TIME است و باید توسط بلوک FC3 در IEC Function Blocks در یک متغیر Temp وارد شود و از متغیر در این پایه اسفاده کنیم.

PERIOD : دوره ی اجرای این بلوک وقفه که به صورت کد های Hex و با توجه به مقادیر زیر باید وارد شود.

W#16#0000	once
W#16#0201	every minute
W#16#0401	hourly
W#16#1001	daily
W#16#1202	weekly
W#16#1401	monthly
W#16#1801	yearly
W#16#2001	at month's end

RET_VAL : یک مموری ورد برای ذخیره کردن کد خطا

ENO : خروجی بیتی نشانگر فعالیت بلوک سیستمی

نکته قابل توجه که باید به آن توجه نمود این است که شماره OB نوشته شده در این بلوک باید حتما آن بلوک وقفه در CPU موجود باشد.

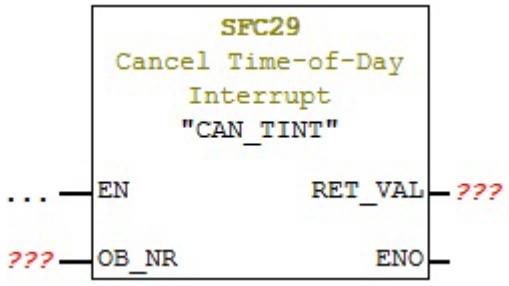
در صورت فعال کردن بلوک وقفه حتما باید بلوک آن در لیست بلوک های دانلود شده موجود باشد، در غیر این صورت با رسیدن به زمان فعال سازی بلوک وقفه، ال ای دی SF روشن میشود و CPU به مد Stop میرود.

خطاهای اختصاصی

8090	Incorrect parameter OB_NR
8091	Incorrect parameter SDT
8092	Incorrect parameter PERIOD
80A1	The set start time is in the past (This error code occurs only when PERIOD = W#16#0000)

SFC29 (CAN_TINT PGM_CNTL)

این بلوک برای غیر فعال کردن بلوک وقفه Time of Day به کار میرود.



EN : فعال ساز بلوک

OB_NR : شماره OB وقفه زمان-تاریخ که میخواهیم غیر فعال شود (شامل اعداد 10 تا17)

توسط پایه EN میتوانیم این بلوک سیستمی را با یک ورودی یا توسط HMI با یک تگ بیتی از مموری فعال کنیم. در صورت فعال شدن این بلوک سیستمی، وقفه زمان-تاریخ (OB10-OB17) غیر فعال میشود.

RET_VAL : یک مموری ورد برای ذخیره کردن کد خطا

ENO : خروجی بیتی نشانگر فعالیت بلوک سیستمی

خطای اختصاصی

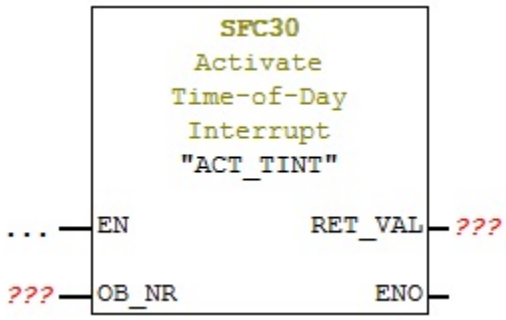
8090	Incorrect parameter OB_NR
80A0	No start date/time specified for the time-of-day interrupt OB

در صورتی که بلوک وقفه مورد نظر را توسط این بلوک سیستمی، غیر فعال کنیم و نیاز باشد دوباره فعال کنیم، باید توسط بلوک بعدی فعال سازی شود.

(ACT_TINT PGM_CNTL) SFC30

این بلوک برای فعال کردن بلوک وقفه ی Time of Day که توسط بلوک سیستمیSFC29 غیر فعال شده است، به کار میرود.

توجه کنید که فعال سازی بلوک بصورت لحظه ای انجام میشود.



EN : فعال ساز بلوک

OB_NR : شماره OB وقفه زمان-تاریخ که میخواهیم فعال شود (شامل اعداد 10 تا17)

RET_VAL : یک مموری ورد برای ذخیره کردن کد خطا

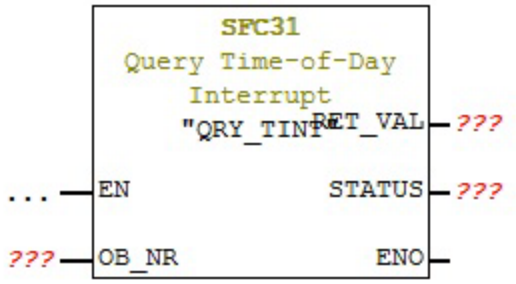
ENO : خروجی بیتی نشانگر فعالیت بلوک سیستمی

خطای اختصاصی

8090	Incorrect parameter OB_NR
80A0	Start date/time-of day not set for the respective time-of-day interrupt OB.
80A1	The activated time is in the past. This error only occurs if execution = once is selected.

(QRY_TINT PGM_CNTL) SFC31

برای نمایش وضعیت فعال بودن یا عدم فعال بودن بلوک های وقفه زمان تاریخ (OB10-OB17) از این بلوک سیستمی استفاده میکنیم.



EN : فعال ساز بلوک

OB_NR : شماره OB وقفه زمان-تاریخ که وضعیت آن را میخواهیم (شامل اعداد 10 تا17)

STATUS : یک ورد وضعیت که نمایان گر وضعیت بلوک وقفه با شماره وارد شده در OB_NR وارد شده است.

RET_VAL : یک مموری ورد برای ذخیره کردن کد خطا

ENO : خروجی بیتی نشانگر فعالیت بلوک سیستمی

خطای اختصاصی

8090	Incorrect parameter OB_NR

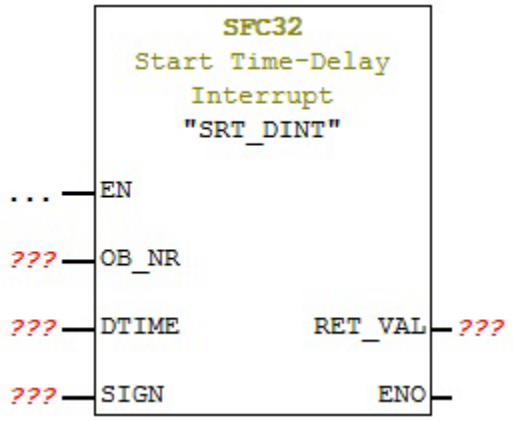
بیت های وضعیت این ورد معرف حالات زیر است:

BIT number	STATUS
0	=0 : CPU is in RUN. =1 : CPU is starting up.
1	=0 : The interrupt is enabled. =1 : The interrupt was disabled by calling SFC39 "DIS_IRT".
2	=0 : The interrupt is not active or has elapsed. =1 : The interrupt is active.
3	always 0
4	=0: An OB with the number of OB_NR does not exist. =1: An OB with the number of OB_NR is loaded.
Other	Always 0

(SRT_DINT PGM_CNTL) SFC32

بلوک سیستمی برای فعال کردن وقفه های Time-Delay Interrupt که شامل بلوک های سازمانی OB20 تا OB23 است.

با فعال سازی این بلوک سیستمی، شماره OB_NR مربوط به بلوک وقفه بعد از طی کردن زمان DTIME یک بار اجرا میشود. توجه شود زمانی که پایه EN در بلوک سیستمی SFC32 تغییر وضعیت میدهد، فراخانی بلوک وقفه انجام میشود و اجرای آن یک سیکل است تا زمانی که مجددا تغیر وضعیت ایجاد شود. پس فعال سازی این بلوک سیستمی باید به صورت لحظه ای و با لبه بالارونده یا پایین رونده انجام شود.



EN : ورودی فعال سازی بلوک

OB_NR : شماره OB وقفه تاخیر-زمان که میخواهیم اجرا شود (شامل اعداد 20 تا 23)

DTIME : زمان تاخیر مورد نیاز برای اجرای برنامه وقفه موجود، با فرمت T# وارد میکنیم. (T#10s)

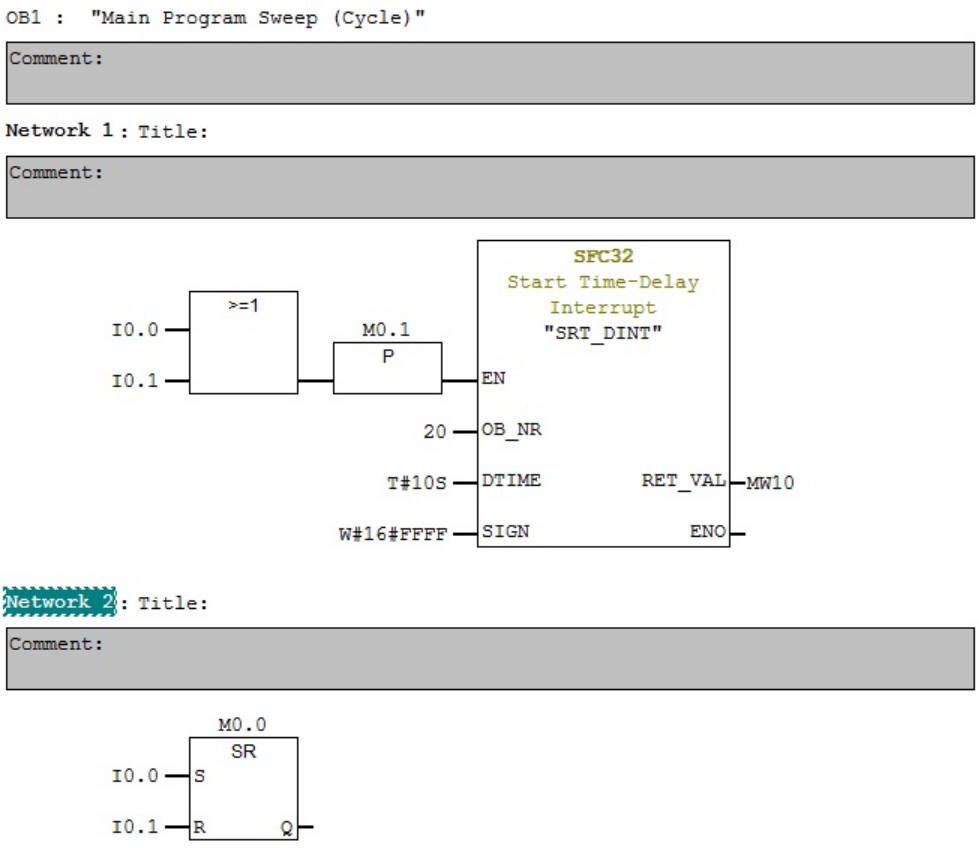
SIGN : شناسه ای که در زمان شروع بکار OBوقفه ظاهر می شود و میتوانیم از آن در بلوک برنامه نویسی وقفه استفاده کنیم.

RET_VAL : یک مموری ورد برای ذخیره کردن کد خطا مشخص می کنیم.

ENO : خروجی بیتی نشانگر فعالیت بلوک سیستمی

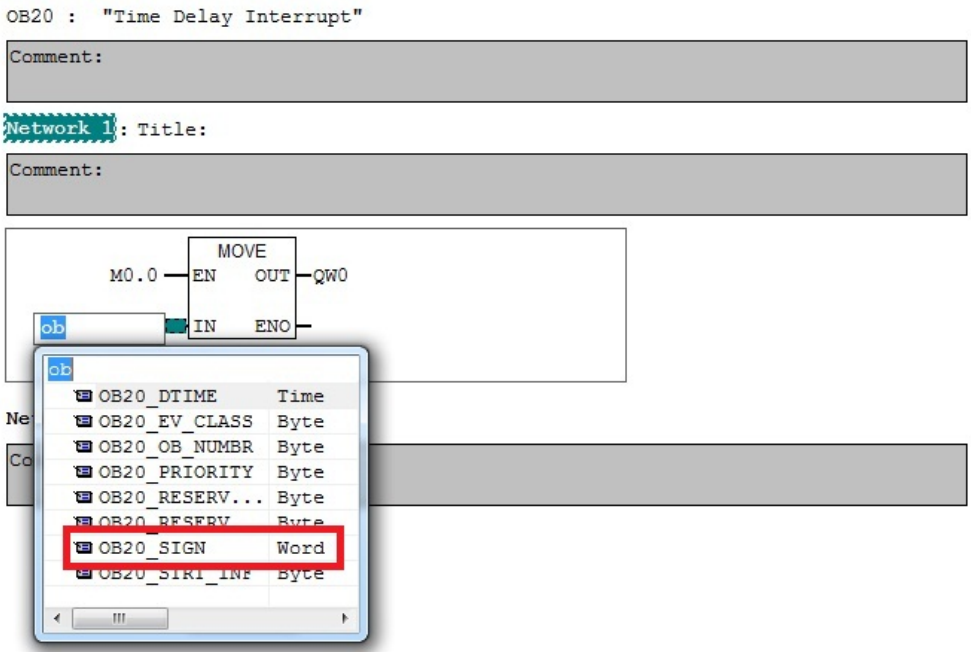
مثال : میخاهیم با فعال شدن ورودی I0.0 یک ورد از خروجی با تاخیر ده ثانیه روشن شود و با فعال شدن I0.1 آن ورد از خروجی خاموش شود.

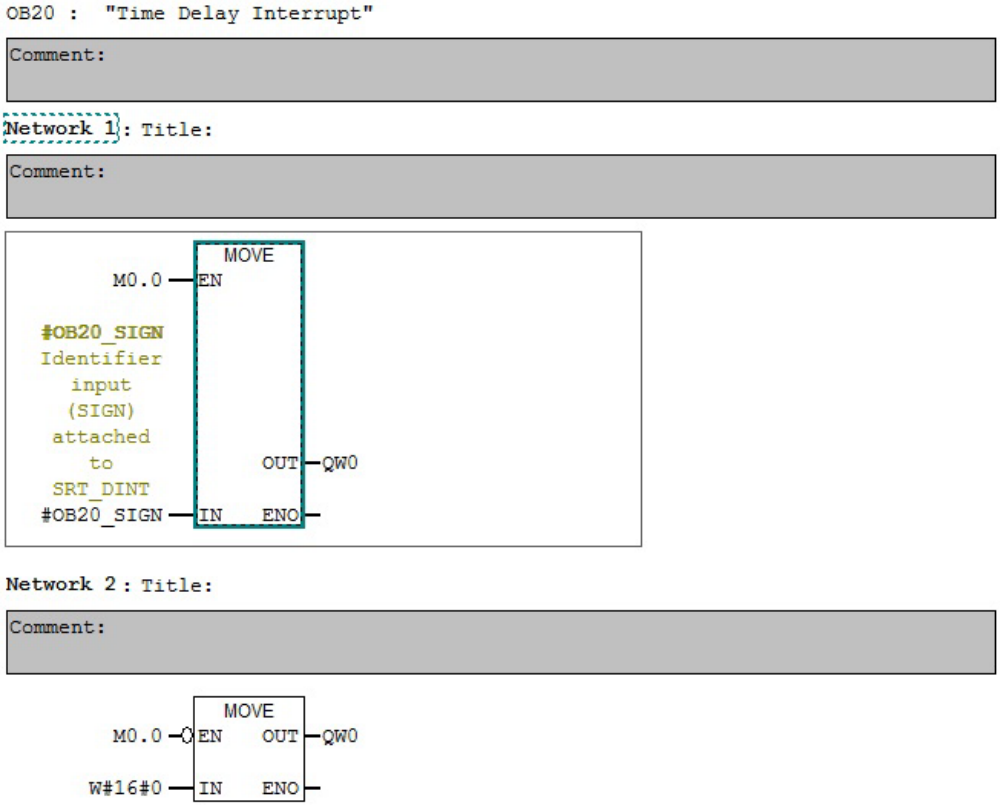
ابتدا در OB1 بلوک SFC32 را فراخانی میکنیم



در اینجا بعد از فعال شدن هر یک از ورودی ها یک لبه به بلوک سیستمی SFC32 وارد میشود. در این لحظه بعد از گذشت 10 ثانیه بلوک وقفه اجرا میشود و تغیرات را اعمال میکند.

حالا OB20را ایجاد کرده و برنامه زیر را مینویسیم. میتوانیم در بلوک وقفه OB20 از SIGN مشخص شده در بلوک سیستمی آن استفاده کنیم.





با دانلود برنامه در سیمولیشن مشاهده میکنیم بعد از زدن I0.0 بلوک SFC32 فعال میشود و وقفه ی OB20 بعد از طی شدن زمان ده ثانیه با توجه به مقادیر نتغیر ها در همان لحظه عمل میکند.

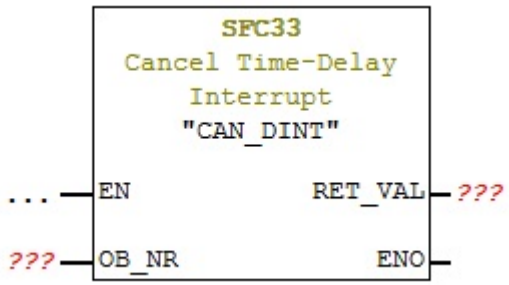
خطای اختصاصی

8090	Incorrect parameter OB_NR
8091	Incorrect parameter DTIME

(CAN_DINT PGM_CNTL) SFC33

بلوک سیستمی برای غیر فعال کردن وقفه های TIME DELAY INTERRUPTکه توسط بلوک SFC32 فعال شده اند.

فعال سازی این بلوک نیز مانند بلوک قبلی با تغییر وضعیت (استفاده از لبه بالارونده و پایین رونده) باید انجام شود.



EN : ورودی فعال سازی بلوک

OB_NR : شماره OB وقفه تاخیر-زمان که میخواهیم غیر فعال شود (شامل اعداد 20 تا 23)

RET_VAL : یک مموری ورد برای ذخیره کردن کد خطا مشخص می کنیم.

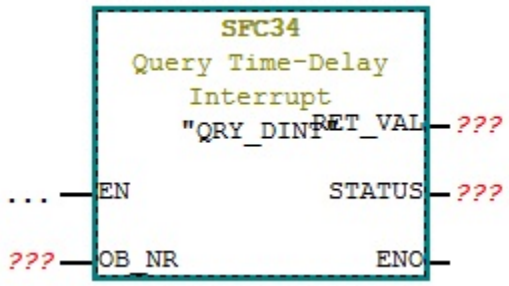
ENO : خروجی بیتی نشانگر فعالیت بلوک سیستمی

خطای اختصاصی

8090	Incorrect parameter OB_NR
80A0	Time-delay interrupt has not started.

(QRY_DINT PGM_CNTL) SFC34

با استفاده از این بلوک سیستمی می توان وضعیت یک OBوقفه تاخیر زمانی (OB20-OB23) را مورد بررسی قرار دهیم.



EN : ورودی فعال سازی بلوک

OB_NR : شماره OB وقفه تاخیر زمانی که میخواهیم وضعیت آنرا مورد بررسی کنیم(شامل اعداد 20 تا 23)

RET_VAL : یک مموری ورد برای ذخیره کردن کد خطا

STATUS : یک ورد وضعیت که نمایانگر وضعیت بلوک وقفه ای است که شماره آن در OB_NR وارد شده است. بین های وضعیت این ورد معرف حالات زیر است.

BIT number	STATUS
0	=0 : CPU is in RUN. =1 : CPU is starting up.
1	=0 : The interrupt is enabled. =1 : The interrupt was disabled by calling SFC39 "DIS_IRT".
2	=0 : The interrupt is not active or has elapsed. =1 : The interrupt is active.
3	always 0
4	=0: An OB with the number of OB_NR does not exist. =1: An OB with the number of OB_NR is loaded.
Other	Always 0

ENO : خروجی بیتی نشانگر فعالیت بلوک سیستمی

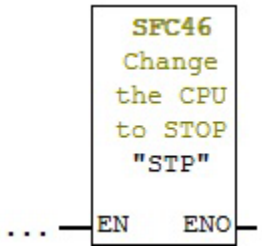
خطای اختصاصی

8090	Incorrect parameter OB_NR

SFC46 (STP PGM_CNTL)

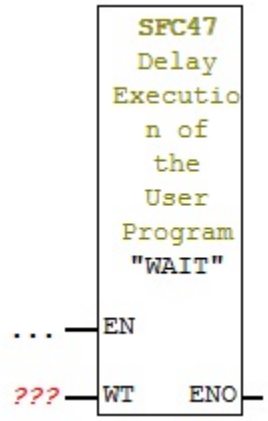
توسط این بلوک سیستمی میتوانیم به صورت نرم افزاری CPU را در مد STOP ببریم. این بلوک پارامتری ندارد و همچنین خطایی در آن صورت نمیگیرد.

از پایه EN میتوانیم برای شرط فعال سازی بلوک استفاده کنیم تا در صورت رخ دادن اتفاق مورد نظر این بلوک فعال شود. همچنین در بلوک های وقفه TIME OF DAY میتوانیم این بلوک سیستمی را در OBوقفه نوشته تا در تاریخ و ساعت مشخص CPU متوقف شود.



SFC47 (WAIT PGM_CNTL)

این بلوک سیستمی در صورت فعال شدن باعث ایجاد تاخیر در اجرای خط بعدی برنامه میشود. این تاخیر تا مقدار 32767 میکروثانیه امکان پذیر است.کمترین مقدار این تاخیر به نوع CPU و زمان اجرای سیکل بلوک سیستمی SFC47 بستگی دارد.



EN : ورودی فعال سازی بلوک

WT : زمان تاخیر با تایپ INT (میکرو ثانیه)

ENO : خروجی بیتی نشانگر فعالیت بلوک سیستمی

عملکرد این بلوک سیستمی توسط OB با اولویت بالاتر متوقف میشود. این بلوک هیچگ.نه خطایی ندارد.

IRT_FNC

در این قسمت فانکشن های فعال و غیر فعال سازی انواع وقفه ها قرار دارد که به بررسی آنها می پردازیم.

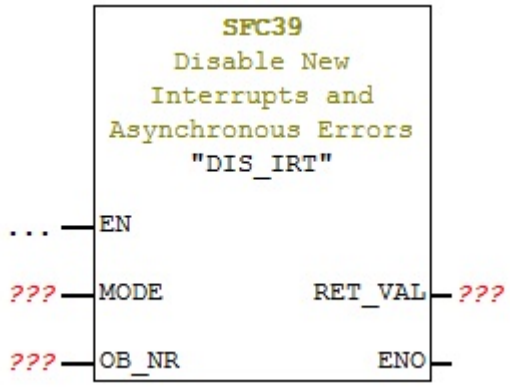
(DIS_IRT IRT_FNC) SFC39

این بلوک سیستمی برای غیر فعال کردن همه وقفه ها یا بخشی از وقفه به کار میرود. فعال شدن این بلوک سیستمی، پردازش وقفه های جدید و خطاهای آسنکرون را غیرفعال می کند. این بدان معنی است که اگر وقفه رخ دهد، سیستم عامل CPU به صورت زیر واکنش نشان می دهد:

- فراخانی برای OB وقفه یا OBهای خطای آسنکرون انجام نمیشود.
- اگر یک وقفه رویداد یا خطای آسنکرون روی دهد و OB های مربوط به آنها برنامه ریزی نشده باشد ، واکنش طبیعی(در برخی موارد روشن شدن SF و STOP شدن CPU ، دربرخی تنها روشن شدن SF و ...) انجام نمی شود.

غیر فعال کردن وقفه ها توسط این بلوک سیسنمی در تمام کلاس های اولویتی تاثیر می گذارد.
فعال سازی مجدد وقفه ها که توسط این بلوک سیستمی غیر فعال شده است ، با بلوک SFC 40 (EN_IRT) و همچنین با راه اندازی مجدد به صورت Warm یا Cold امکان پذیر است.

این که آیا سیستم عامل، وقفه های رویداد یا وقفه های خطا را در بافر تشخیص (Diagnostic Buffer) در هنگام وقوع آن ثبت کند یا خیر بستگی به تنظیم پارامتر ورودی MODE دارد.



EN : ورودی فعال سازی بلوک

MODE : این پارامتر مشخص میکند کدام وقفه یا خطا غیر فعال شود. مد کاری این بلوک در محدوده بایت و با فرمت Hex نوشته میشود و شامل کد های زیر میباشد:

MODE(B#16#..)	
00	تمام وقفه های رویداد و خطاهای آسنکرون غیر فعال میشوند (سنکرون ها غیر فعال نمیشوند) در این حالت پارامتر OB_NR را برابر 0 قرار میدهیم.
01	در این حالت یک کلاس از وقفه ها غیر فعال میشود. در این حالت شماره شروع OB های کلاس مورد نظر را در قسمت OB_NR مینویسیم. Time- of-day interrupts: 10 Time-delay interrupts: 20 Cyclic interrupts: 30 Hardware interrupts: 40 Interrupts for DPV1: 50 Multicomputing interrupts: 60 Redundancy error interrupts: 70 Asynchronous error interrupts: 80
02	تنها یک OBوقفه با شماره مشخص که در OB_NR نوشته میشود غیر فعال میشود.
80	حالت 00 بدون ثبت وقفه بوجود آمده در بافر تشخیص (Diagnostic Buffer)
81	حالت 01 بدون ثبت وقفه بوجود آمده در بافر تشخیص (Diagnostic Buffer)
82	حالت 02 بدون ثبت وقفه بوجود آمده در بافر تشخیص (Diagnostic Buffer)

OB_NR : با توجه به مد انتخابی، مقدار شرح داده شده در جدول بالا قرار میگیرد.

RET_VAL : یک مموری ورد برای ذخیره کردن کد خطا مشخص می کنیم.

ENO : خروجی بیتی نشانگر فعالیت بلوک سیستمی

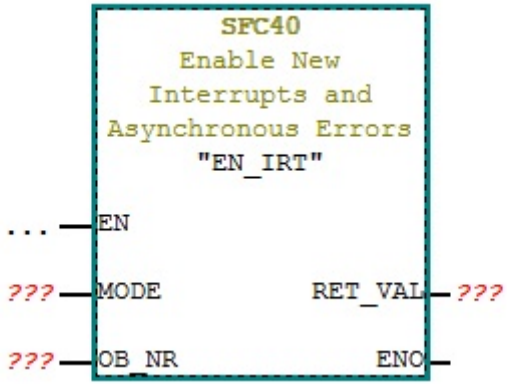
خطای اختصاصی

8090	The input parameter OB_NR contains an illegal value.
8091	The input parameter MODE contains an illegal value.

(EN_IRT IRT_FNC) SFC40

توسط این بلوک سیستمی، وقفه ها و خطاهای آسنکرون که قبلا با SFC 39 غیر فعال شده بود را میتوان دوباره فعال کرد. اگر وقفه ای رخ دهد، سیستم عامل CPU در یکی از روش های زیر واکنش نشان می دهد:

- OBمربوط به آن وقفه یا خطا فراخانی و اجرا میشود.
- اگر OB مربوط به آن وقفه برنامه ریزی نشده باشد و در لیست برنامه های دانلود شده موجود نباشد، با توجه به نوع وقفه واکنش استاندارد توسط سیستم عامل اجرا میشود.



EN : ورودی فعال سازی بلوک

MODE : این پارامتر مشخص میکند کدام وقفه غیر فعال شده، فعال شود. مد کاری این بلوک در محدوده بایت و با فرمت Hex نوشته میشود و شامل کد های زیر میباشد:

MODE(B#16#..)	
00	تمام وقفه های رویداد و خطاهای آسنکرون فعال میشوند. در این حالت پارامتر OB_NR را برابر 0 قرار میدهیم.
01	در این حالت یک کلاس از وقفه ها فعال میشود. در این حالت شماره شروع OB های کلاس مورد نظر را در قسمت OB_NR مینویسیم. Time- of-day interrupts: 10 Time-delay interrupts: 20 Cyclic interrupts: 30 Hardware interrupts: 40 Interrupts for DPV1: 50 Multicomputing interrupts: 60 Redundancy error interrupts: 70 Asynchronous error interrupts: 80

تنها یک OBوقفه با شماره مشخص که در OB_NR نوشته میشود، فعال میشود.	**02**

OB_NR : با توجه به مد انتخابی، مقدار شرح داده شده در جدول بالا قرار میگیرد.

RET_VAL : یک مموری ورد برای ذخیره کردن کد خطا مشخص می کنیم.

ENO : خروجی بیتی نشانگر فعالیت بلوک سیستمی

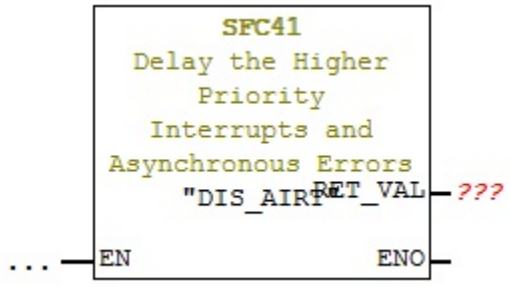
خطای اختصاصی

8090	The input parameter OB_NR contains an illegal value.
8091	The input parameter MODE contains an illegal value.

(DIS_AIRT IRT_FNC) SFC41

وظیفه ی این بلوک سیستمی ایجاد تاخیر در شروع به خواندن برنامه ی OB های وقفه و OB های خطای آسنکرون که اولویت بالاتری نسبت به OB فعلی دارد، است. این تاخیر تا زمانی که برنامه OB فعلی پایان یابد ادامه میابد. برای هر OB می توان بیشتر از یک بار SFC 41 را فراخانی کرد.

برای لغو هر یک از این بلوک سیستمی فراخانی شده باید از SFC42 استفاده شود.



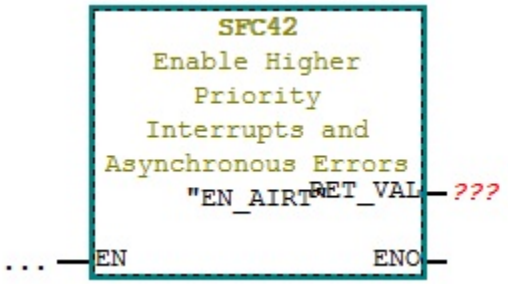
EN : ورودی فعال سازی بلوک

RET_VAL : تعداد دفعاتی که این بلوک در فراخانی OB با اولویت بالاتر تاخیر ایجاد کرده است.

ENO : خروجی بیتی نشانگر فعالیت بلوک سیستمی

SFC42 (EN_AIRT IRT_FNC)

توسط این بلوک سیستمی ، اجرای وقفه های با اولویت بالاتر که قبلا توسط SFC41 غیر فعال شده بود مجددا فعال میشود. توجه داشته باشید به تعداد غیر فعال سازی اولویت اجرای OB ها (تعداد بلوک استفاده شده ی SFC41) باید فعال سازی انجام شود.



EN : ورودی فعال سازی بلوک

RET_VAL : این پارامتر دو حالت دارد

نمایش به صورت "n" که n تعداد بلوک های SFC41 که هنوز غیر فعال نشده اند را نمایش میدهد.(الویت ها زمانی دوباره اجرایی میشوند که تعداد بلوک های SFC41 غیر فعال شده 0 باشد.)

نمایش به صورت W#16#8080 که به معنی استفاده مجدد از این بلاک در صورتی که قبلا فعال سازی الویت ها انجام شده است.

ENO : خروجی بیتی نشانگر فعالیت بلوک سیستمی

MOVE

در این قسمت فانکشن های انتقال اطلاعات بین حافظه با حجم بالا را داریم که به بررسی آنها میپردازیم.

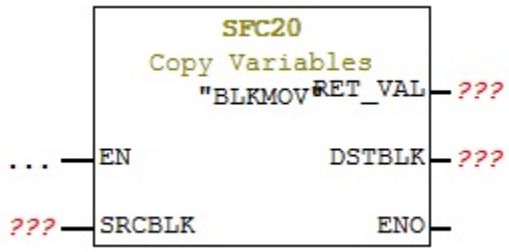
SFC20 (BLKMOV MOVE)

این بلوک سیستمی برای کپی کردن بخشی از حافظه مورد استفاده قرار میگیرد. در دستورات MOVE نرم افزار حداکثر تا 4 بایت را میتواند انتقال داد ولی در این بلوک سیستمی محدودیت برای انتقال وجود ندارد و متغیرهای از جنس TD و ARRAY و STRUCT با این بلوک سیستمی میتوان انتقال داد.

اطلاعات قابل انتقال شامل اطلاعات Db ها، اطلاعات Memory، تصاویر وردی و تصاویر خروجی است.

باید توجه داشت مبدا انتقال با مقصد آن تداخلی از نظر آدرس نداشته باشند.

عملکرد انتقال در این بلوک سیستمی میتواند توسط وقفه با اولویت بالاتر متوقف شود و بعد از انجام وقفه عملیات انتقال ادامه نمیابد.



EN : ورودی فعال سازی بلوک

SRCBLK : آدرس شروع مبدا انتقال و تعداد بایت انتقالی با فرمت ANY (P#...BYTE...) داده میشود.

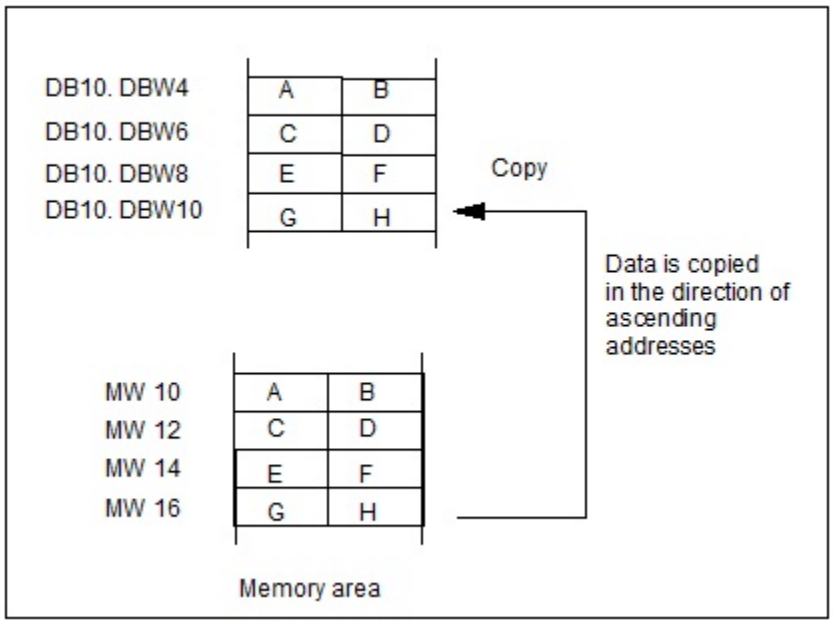
برای مثال P#M20.0 BYTE 20 یعنی 20 بایت از آدرس شروع 20.0 در ناحیه M

برای مثال P#DB3.DBX53.0 BYTE 10 یعنی 10 بایت با ادرس شروع 53.0 از DB3

DSTBLK : آدرس شروع مقصد و تعداد بایت انتقالی با فرمت ANY داده میشود.

RET_VAL : یک ممووری ورد برای ذخیره کردن کد خطا

ENO : خروجی بیتی نشانگر فعالیت بلوک سیستمی



در صورتی که مقدار بایت های مقصد از مقدار بایت های مبدا بیشتر باشد، در بایت های اضافی اطلاعاتی قرار نمیگیرد.

در صورتی که تعداد بایت های مبدا بیشتر باشد به تعداد بایت های مقصد اطلاعات منتقل میشود.

در صورتی که تعداد بایت های مشخص شده بعد از آدرس شروع، از محدوده تعداد بایت های موجود بیشتر شود بسته به نوع CPU عکس العمل های زیر رخ میدهد:

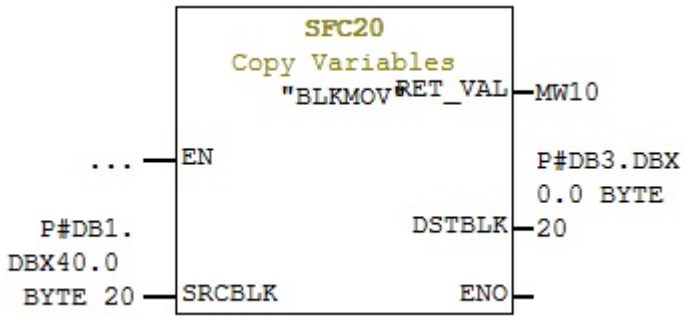
در S7-300 انتقال اطلاعات انجام نمی شود و کد خطای W#16#837F در RET_VAL قرار میگیرد.

در S7-400 و CPUبا ورژن 4 انتقال اطلاعت انجام نمیشود و در سایر CPUهای سری 400 انتقال تا حد موجود انجام میشود و کد خطای W#16#8122 یا W#16#8323 در RET_VAL قرار میگیرد.

خطای اختصاصی

8090	Nesting depth exceeded.
8092	SFC 20 "BLKMOV" cannot be executed because a unusable ((write-protected, unrunnable on unloaded) data block was accessed..

مثال : انتقال محتوای 20 بایت از DB1 با آدرس شروع 40.0 به DB3 با آدرس شروع 0.0

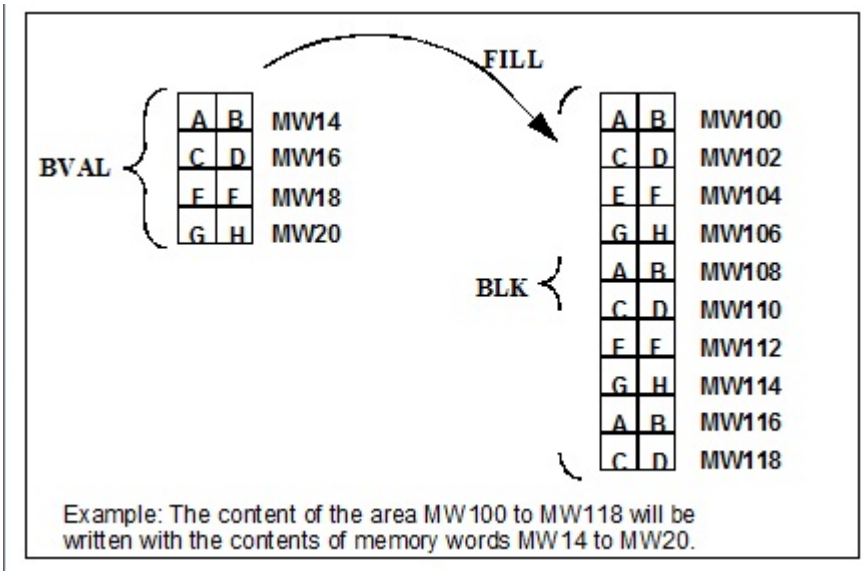


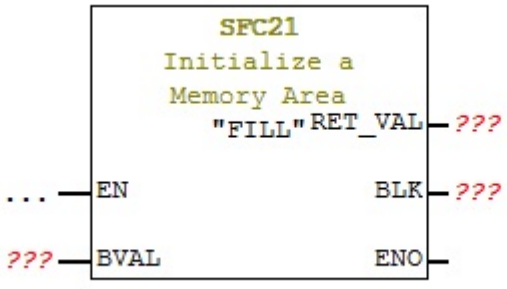
(Fill MOVE) SFC21

این بلوک مانند بلوک قبلی عملیات انتقال را انجام می دهد، تنها با این تفاوت که در این بلوک اگر مقصد دارای ناحیه بزرگ تر باشد عملیات انتقال با تکرار داده ها انجام شده تا کل فضای مشخص شده پر شود.

نکات گفته شده در بلوک قبل در اینجا صادق است.

طریقه پر شدن حافظه مقصد در صورتی که بزرگ تر از مبدا باشد را در شکل زیر ملاحضه میکنید.





EN : ورودی فعال سازی بلوک

BVAL : آدرس شروع مبدا انتقال و تعداد بایت انتقالی با فرمت ANY (P#...BYTE...) داده میشود.

BLK : آدرس شروع مقصد و تعداد بایت انتقالی با فرمت ANY داده میشود.

RET_VAL : یک مموری ورد برای ذخیره کردن کد خطا مشخص میکنیم.

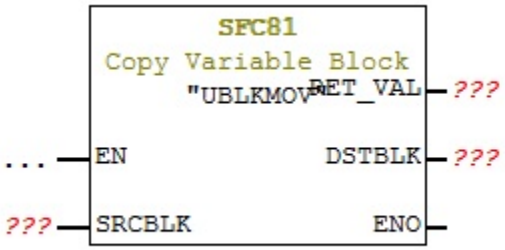
ENO : خروجی بیتی نشانگر فعالیت بلوک سیستمی

(UBLKMOV MOVE) SFC81

این بلوک سیستمی که UBLKMOV (uninterruptible block move) نام دارد دقیقا همانند بلوک SFC20 عمل میکند با این تفاوت که اگر در حین انتقال وقفه با الویت بالاتر ایجاد شود، عملیات انتقال متوقف نمیشود و برنامه وقفه با الویت بالاتر بعد از پایان عملیات انتقال اجرا میشود.

می توان حداکثر 512 بایت داده را توسط این بلوک سیستمی منتقل کرد.

سایر نکات گفته شده در مورد SFC20 نیز در اینجا صادق است.



EN : ورودی فعال سازی بلوک

SRCBLK : آدرس شروع مبدا انتقال و تعداد بایت انتقالی با فرمت ANY (P#...BYTE...) داده میشود.

DSTBLK : آدرس شروع مقصد و تعداد بایت انتقالی با فرمت ANY داده میشود.

RET_VAL : یک مموری ورد برای ذخیره کردن کد خطا

ENO : خروجی بیتی نشانگر فعالیت بلوک سیستمی

خطای اختصاصی

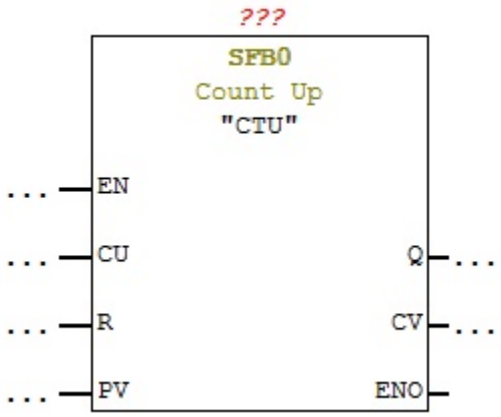
8091	The source area is in an unlinked data block.

SFB

بلوک های سیستمی حافظه دار یا SFB ها شامل فانکشن های تخصصی تر از SFC ها است که در شبکه و کنترلر های PID و ... کاربرد فراوان دارند. در آدامه به معرفی خانواده پر کاربرد IEC_TC این بلوک های سیستمی میپردازیم.

SFB0 (CTU IEC_TC)

این بلوک سیستمی شمارنده صعودی شمار دارای مقایسه گر است. برخلاف کانتر های CPU، این شمارنده در محدوده ی INT شمارش انجام میدهد و تا تعداد 32767 را شمارش میکند. در حالی که کانتر های CPU تا 999 تعداد را شمارش میکردند. چون این بلوک سیستمی به صورت Function Block است پس نیاز به یک DB اختصاصی دارند. در این شمارنده زمانی که بازه شمارش به مقدار مشخص شده در پایه PV برسد، خروجی فعال شده و برخلاف کانترهای قبلی پایه PV برای مقدار دهی اولیه نیست و مقایسه گر است.



EN: ورودی فعال ساز بلوک

CU: ورودی بالا شمار

R: پایه ریست

PV: مقدار مقایسه شونده برای فعال سازی بیت خروجی Q

Q: خروجی بیتی

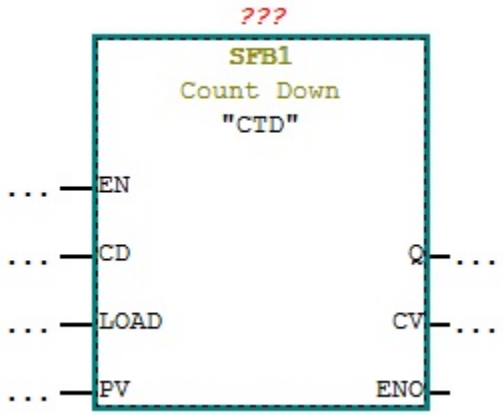
CV: مقدار جاری شمارنده

ENO: خروجی نمایش دهنده فعال بودن بلاک

مقدار شمارش شده این کانتر که در پایه CV ثبت میشود، با راه اندازی COLD ریست میشود. در صورت نیاز به ریست کردن این مقدار در راه اندازیWARM میتوانیم در بلوک OB100 این بلوک را فراخانی و پایه R را یک منطقی دهیم.

(CTD IEC_TC) SFB1

این بلوک سیستمی یک شمارنده پایین شمار است. زمانی که مقدار جاری شمارش کوچکتر و برابر با صفر باشد خروجی بیتی فعال میشود. زملنی که مقدار CV برابر با عدد 32767- شود شمارش متوقف میشود.
تفاوت دیگر این بلوک با بلوک سیستمی قبلی در لود شدن مقدار اولیه است. در اینجا پایه PV برای لود شدن مقدار اولیه بکار میرود. زمانی که ورودی LOAD در این بلوک 1 شود، مقدار شمارش شده کانتر (CV) برابر با مقدار اولیه (PV) میشود. بازه شمارش در این شمارنده 32767+ تا 32767- است.



EN: ورودی فعال ساز بلوک

CD: ورودی پایین شمار

LOAD: ورودی جهت بار کردن مقدار PV

PV: ورودی مقدار اولیه

Q: خروجی بیتی

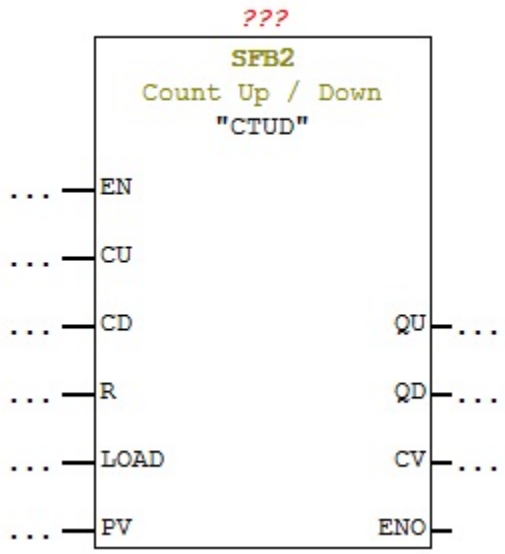
CV: مقدار جاری شمارنده

ENO: خروجی بیتی نشان دهنده فعالیت بلوک سیستمی

مقدار شمارش شده این کانتر که در پایه CV ثبت میشود، با راه اندازی COLD ریست میشود. در صورت نیاز به ریست کردن این مقدار در راه اندازیWARM میتوانیم در بلوک OB100 این بلوک را فراخانی و پایه R را یک منطقی دهیم.

(CTUD IEC_TC) SFB2

این بلوک یک شمارنده صعودی/ نزولی است که دارای مقایسه گر نیز هست. پایه های CU جهت شمارش به بالا و پایه CD جهت شمارش به پایین است. این شمارنده دارای 2 خروجی QD و QC جهت نمایش وضعیت CV میباشد و چنانچه مقدار جاری کوچکتر و مساوی عدد صفر باشد، خروجی QD روشن و اگر مقدار جاری بزرگتر مساوی مقدار تعیین شده در پایه PV باشد خروجی QU روشن میشود.
حد بالا و پایین شمارش این شمارشگر به ترتیب 32767+ و 32767- است.



EN: ورودی فعال ساز بلوک

CU: ورودی بالا شمار

CD: ورودی پایین شمار

R: پایه ریست

PV: مقدار مقایسه شونده برای فعال سازی بیت خروجی QU (LOAD=0) / ورودی مقدار اولیه (LOAD=1)

LOAD: ورودی جهت بار کردن مقدار PV

QD: خروجی بیتی مقایسه کننده CV با مقدار0 (در صورتی که CV>=0 باشد QD=1)

QU : خروجی بیتی مقایسه کننده CV با مقدار PV (در صورتی که CV>=PV باشد QU=1)

CV: مقدار جاری شمارنده

ENO: خروجی بیتی نشان دهنده فعالیت بلوک سیستمی

چناچه ورودی LOAD دارای مقدار منطقی 1 باشد، مقادیر PV را بدون توجه به ورودی CU و CD در CV ست میشود. این مقدار تا زمانی که پایه LOAD برابر با 0 منطقی نشود ثابت میماند.

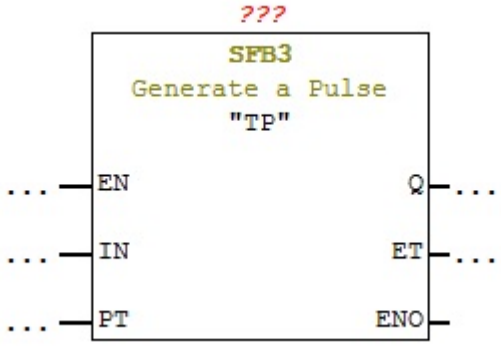
سطح سیگنال 1 در ورودی R، مقادیر 0 را بدون توجه به مقادیر ورودی CU، CD و LOAD در خروجی CV ست میکند.

مقدار شمارش شده این کانتر که در پایه CV ثبت میشود، با راه اندازی COLD ریست میشود. در صورت نیاز به ریست کردن این مقدار در راه اندازیWARM میتوانیم در بلوک OB100 این بلوک را فراخانی و پایه R را یک منطقی دهیم.

(TP IEC_TC) SFB3

این بلوک سیستمی عملکرد شمارنده زمان یا تایمر را دارد که با تایمر های سخت افزاریCPU ساختار متفاوتی دارند. این تایمر از نوع تاخیر در قطع است که با اعمال لبه بالا رونده به پایه IN تایمر شروع به شمارش مقدار PT میکند و خروجی Q آن تا پایان شمارش زمان دارای سطح سیگنال 1 میشود.

این تایمر تا مقدار 24D-20H-31M-23S را میتواند شمارش کند.



EN: ورودی فعال ساز بلوک

IN: ورودی فعال سازی تایمر با مقدار PT

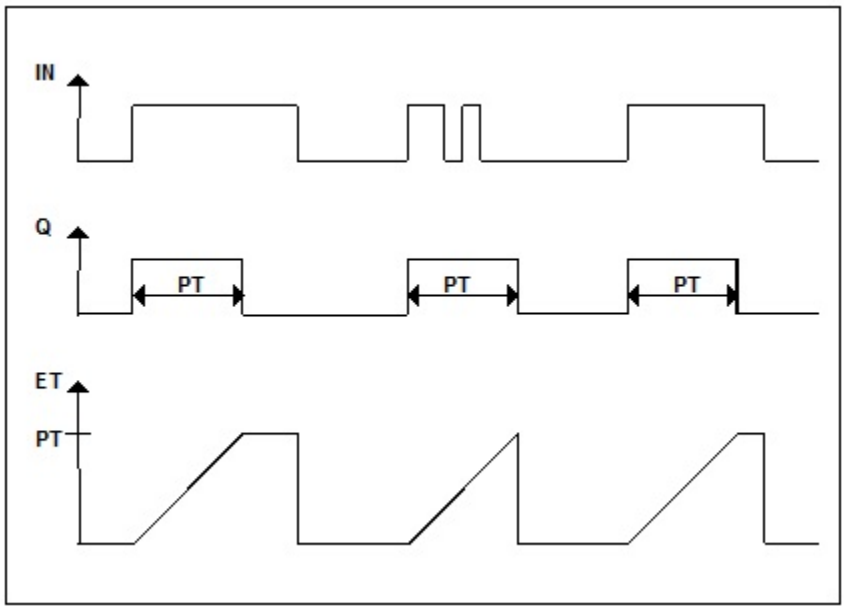
PT: مقدار زمان مورد نیاز برای تایمر که با فرمت T#... وارد میشود. (T#24D20H31M23S)

Q: خروجی تایمر

ET: : مقدار زمان جاری تایمر(تایپ TIME با وزن دابل ورد)

ENO: خروجی بیتی نشان دهنده فعالیت بلوک سیستمی

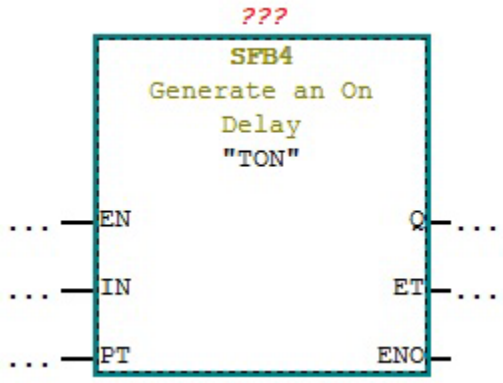
این تایمر با لبه بالارونده در پایه IN شروع به شمارش زمان میکند و تا پایان یافتن زمانPT ، تغییر وضعیت ورودی IN در روند آن اثر گذار نیست. (به عبارت دیگر قبل از پایان یافتن زمان مشخص شده برای تایمر حتی زمانی که ورودی IN از 0 به 1 میرود تغییر در آن ایجاد نمیشود.)



نکته: با استفاده از یک برنامه نویسی صحیح (multiple instances) میتوانیم از یک تایمر نرم افزاری به دفعات متعدد با بازه زمانی های مختلف استفاده کنیم.

SFB4 (TON IEC_TC)

این بلوک سیستمی یک تایمر تاخیر در وصل است که به هر 2 لبه بالا و پایین در ورودی فعال سازی آن حساس است. از این تایمر به عنوان تایمر TON(On Delay Timer) نام میبرند. در این تایمر با اعمال لبه بالا رونده در پایه IN تایمر شروع به شمارش زمان PT کرده و پس از سپری کردن این زمان، خروجی Q تایمر روشن شده و تا زمانی که لبه پایین رونده در ورودی IN اعمال شود (ورودی IN از 1 به 0 تغییر وضعیت دهد) روشن میماند.



EN: ورودی فعال ساز بلوک

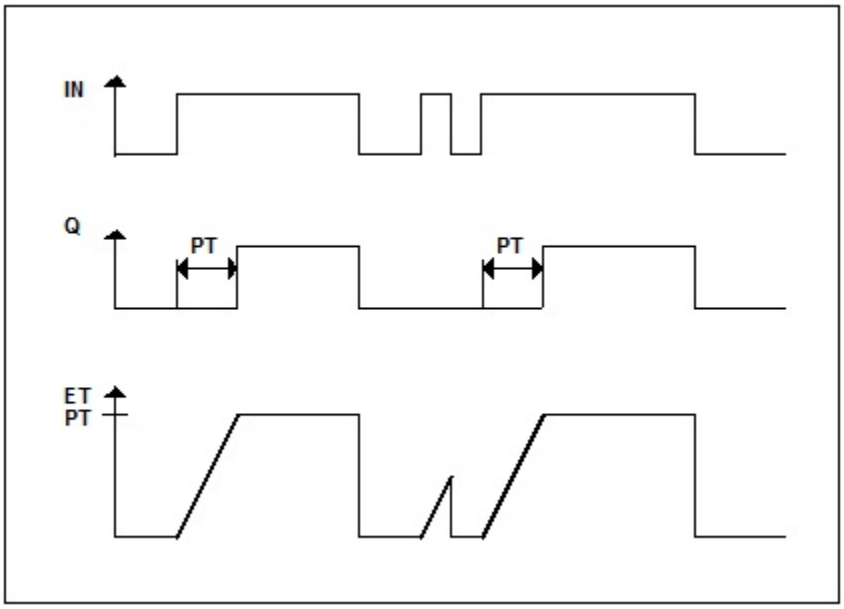
IN: ورودی فعال سازی تایمر با مقدار PT

PT: مقدار زمان مورد نیاز برای تایمر که با فرمت T#... وارد میشود. (T#24D20H31M23S)

Q: خروجی تایمر

ET: مقدار زمان جاری تایمر(تایپ TIME با وزن دابل ورد)

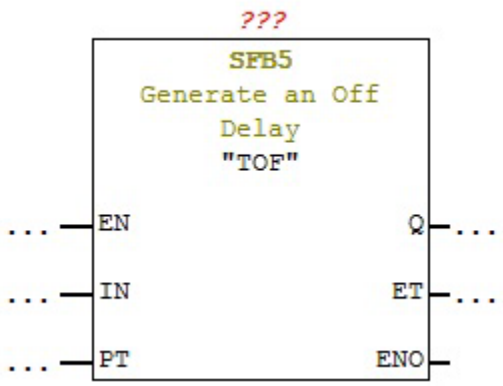
ENO: خروجی بیتی نشان دهنده فعالیت بلوک سیستمی



سیستم عامل در راه اندازی سرد مقدار زمان تایمر را ریست میکند. برای ریست کردن مقدار تایمر در راه اندازی سرد باید بلوک SFC4 را در OB100 فراخانی کرده و مقدار PT را برابر با 0 منطقی کنیم.

SFB5 (TOF IEC_TC)

این بلوک سیستمی به تایمر TOF (Time Off Delay) معروف است. عملکرد این بدین صورت است که با قرار گرفتن لبه بالا رونده در پایه IN، خروجی Q تایمر فعال میشود. زمانی که لبه پایین رونده اعمال شود تایمر شروع به شمارش کرده و بعد از گذراندن زمان تعیین شده در پایه PT (بعد از تاخیر زمان PT پس از غیرفعال شدن ورودی IN) ، خروجی Q صفر منطقی میشود.



EN: ورودی فعال ساز بلوک

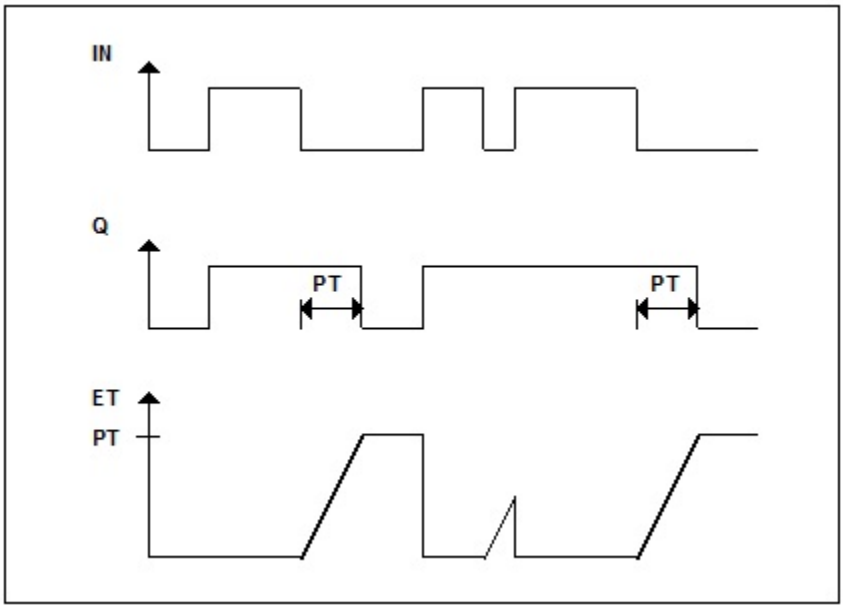
IN: ورودی فعال سازی تایمر با مقدار PT

PT: مقدار زمان مورد نیاز برای تایمر که با فرمت T#... وارد میشود. (T#24D20H31M23S)

Q: خروجی تایمر

ET: مقدار زمان جاری تایمر(تایپ TIME با وزن دابل ورد)

ENO: خروجی بیتی نشان دهنده فعالیت بلوک سیستمی


IN
Q
PT
PT
ET
PT