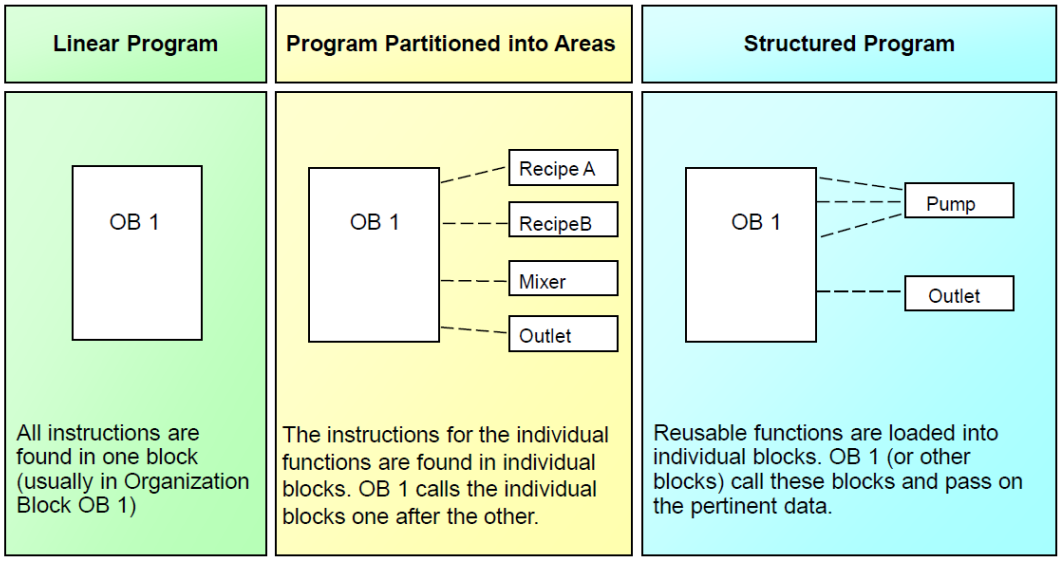
Program Structuring Possibilities



Linear Program: The entire program is found in one continuous program block. This model resembles a hard-wired relay control, that was replaced by a programmable logic controller. The CPU processes the individual instructions one after the other.

Partitioned Program: The program is divided into blocks, whereby every block only contains the program for solving a partial task. Further partitioning through networks is possible within a block. You can generate network templates for networks of the same type. The OB 1 organization block contains instructions that call the other blocks in a defined sequence.

Structured Program: A structured program is divided into blocks. The code in OB1 is kept to a minimum with calls to other blocks containing code. The blocks are parameter assignable. These blocks can be written to pass parameters so they can be used universally. When a parameter assignable block is called, the programming editor lists the local variable names of the blocks. Parameter values are assigned in the calling block and passed to the function or function block.

Example:

• A "pump block" contains instructions for the control of a pump.

• The program blocks, which are responsible for the control of special pumps, call the "pump block" and give it information about which pump is to be controlled with which parameters.

• When the "pump block" has completed the execution of its instructions, the program returns to the calling block (such as OB 1), which continues processing the calling block‘s instructions.

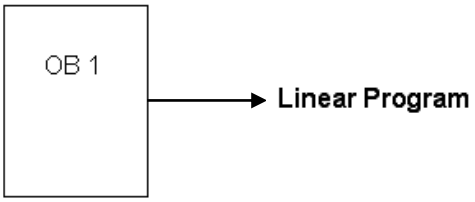
آشنایی با روش های برنامه نویسی

در اکثر نرم افزارهای برنامه نویسی معمولا برنامه نویس می تواند برنامه خود را با توجه به حجم و نوع پروسه کنترلی به روش های مختلفی طراحی نماید. منظور از روش های مختلف برنامه نویسی روش های STL، FBD و LAD نمی باشد. منظور از روش های برنامه نویسی در این قسمت ساختاری است که برنامه نویس برای برنامه خود در نظر می گیرد. معمولا در برنامه کنترلی می توان از ۳ ساختار استفاده نمود. البته لازم به ذکر است که انتخاب ساختار مناسب برای برنامه، بستگی به حجم و نوع برنامه دارد.

- Linear Program
- Divided Program
- Structured Program

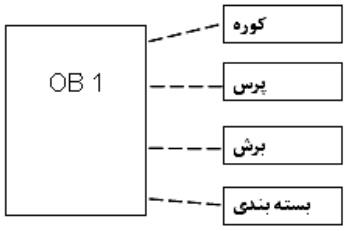
Linear Program

این روش برنامه نویسی تحت عنوان برنامه نویسی خطی شناخته می شود. در این روش تمامی برنامه مربوط به یک پروسه در یک بلوک و به صورت متوالی نوشته می شود. منظور از بلوک همان بلوک OB1 در نرم افزار می باشد. در ابتدا لازم است که ویژگی و نحوه اجرای بلوک OB1 مورد بررسی قرار گیرد. در اکثر PLC های شرکت زیمنس بلوکهایی با نام های مختلف جهت کاربردهای مختلف تعبیه شده است. اما وجه اشتراک بین تمامی PLC های زیمنس استفاده از بلوک OB1 به عنوان بلوک اصلی در برنامه می باشد. زمانی که CPU به حالت RUN سوئیچ می شود، CPU در ابتدا در صورت وجود OB های راه اندازی به آنها مراجعه و سپس نیاز به بلوک OB1 دارد و در واقع سیستم عامل بلوک OB1 را فراخوانی می کند. البته اجرای OB1 را نیز می توان توسط سایر OB ها در هر لحظه قطع نمود. پس در نتیجه بلوک OB1 به عنوان ساختار اصلی برنامه کاربر بوده که اکثر بلوک های برنامه جهت اجرا حتما می بایست در این بلوک فراخوانی شوند. در برنامه نویسی خطی برنامه نویس کل برنامه خود را در بلوک OB1 در Network های متوالی وارد می کند. در پروژه های صنعتی که حجم برنامه کنترلی زیاد می باشد، نوشتن تمامی برنامه در بلوک OB1 کار صحیح و منطقی نمی باشد. به دلیل اینکه تمامی برنامه می بایست پشت سرهم و در یک بلوک طراحی شود. یکی از معایب برنامه نویسی خطی عیب یابی مشکل در برنامه نوشته شده می باشد. همچنین در بسیاری از مواقع نیاز می باشد که قسمتی از برنامه توسط CPU پردازش نشود. در برنامه نویسی خطی به دلیل اینکه تمامی برنامه در یک بلوک می باشد، برنامه نویس معمولا کمتر می تواند بر روی بخش های مختلف برنامه کنترل لازم را داشته باشد.



Divided Program

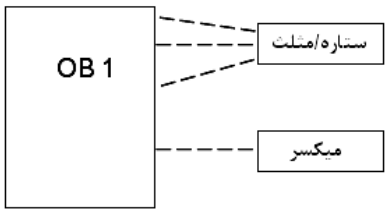
در این روش برنامه نویسی از بلوک های دیگری جهت نوشتن برنامه استفاده می شود. در واقع در این روش قسمت های مختلف مربوط به یک پروژه در بلوک های مجزا از یکدیگر نوشته می شوند. این بلوک ها به عنوان زیربرنامه مورد استفاده قرار می گیرند. به عنوان مثال یک خط تولید را در نظر بگیرید که شامل بخش های مختلفی از جمله کوره ، پرس ، برش و بسته بندی می باشد. در این حالت با استفاده از بلوک های زیربرنامه می توان برنامه مربوط به هر بخش را از بخش دیگر جدا نمود.



این نوع برنامه نویسی که کاربرد فراوانی در پروژه های بزرگ دارد، تحت عنوان برنامه نویسی تقسیم بندی شده شناخته می شود و یکی از مزایای این روش عیب یابی آسان و تحت کنترل درآوردن هر چه بیشتر اجزای مختلف برنامه می باشد. در PLC مختلف شرکت زیمنس نام های مختلفی برای این بلوک ها در نظر گرفته شده است که در ادامه در نرم افزار مورد بحث با این بلوک ها آشنا می شویم.

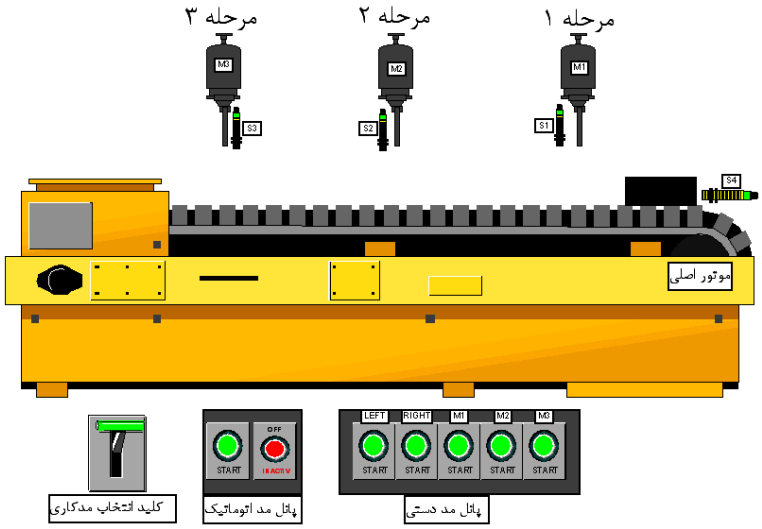
Structured Program

روش دیگری که در پروژه های بزرگ مورد استفاده قرار می گیرد، روش برنامه نویسی ساختار یافته می باشد. از این روش در پروژه هایی که یک منطق بین چندین بخش از پروژه مشابه می باشد استفاده می شود. در واقع استفاده از این روش زمان برنامه نویسی را بسیار کاهش می دهد. به عنوان مثال فرض کنید که قرار است ۱۰ موتور توسط یک PLC کنترل شوند. راه اندازی تمامی موتورها به صورت ستاره/ مثلث می باشند. تنها تفاوت در این پروژه، آدرس های I/O مربوط به هر موتورها می باشد. در این گونه موارد نیازی به تکرار برنامه برای ۱۰ موتور به صورت مجزا نمی باشد. در این صورت برنامه نویس می تواند یک تابع با منطق ستاره/ مثلث طراحی و این تابع را برای تمامی موتورها استفاده کند. در نتیجه استفاده از روش ساختار یافته در بخش هایی از برنامه که قرار است تکرار شود بسیار مفید می باشد. این روش می تواند در پروژه های بزرگ به عنوان کامل ترین روش مورد استفاده قرار گیرد.



تمرین1-(Partitioned Program)

می خواهیم برنامه مربوط به یک دستگاه که دارای دو مد دستی و اتوماتیک می باشد را طراحی کنیم. نحوه کار بدین صورت می باشد که در این ماشین صنعتی از یک موتور که به صورت چپگرد راستگرد کار می کند جهت انتقال قطعه استفاده می شود. این قطعه می بایست در مراحل مختلف جهت انجام عملیات ماشین کاری متوقف شود. زمان توقف نیز در هر مرحله متفاوت می باشد. در ضمن عملیات ماشین کاری تنها در مسیر رفت انجام می گردد.



مد اتوماتیک: در این مد اپراتور با فشردن شاستی استارت در پانل مربوط به مد اتوماتیک، موتور را در جهت چپگرد وارد مدار می کند. با روشن شدن موتور، قطعه تا سنسور S1 جهت انجام عملیات ماشین کاری در مرحله ۱ پیش می رود. با فعال شدن سنسور S1، موتور اصلی متوقف و موتور M1 وارد مدار می شود. زمان انجام عملیات ماشین کاری در این مرحله ۶۰ ثانیه می باشد. پس از سپری شدن این زمان موتور M1 خاموش و مجددا موتور اصلی در جهت چپگرد شروع به حرکت نموده و این سیکل را در مراحل بعدی نیز تکرار می کند. پس از اتمام مرحله ۳، موتور در جهت راستگرد وارد مدار شده و قطعه را به نقطه اولیه که توسط سنسور S4 محدود می شود باز می گرداند. در ضمن در مسیر برگشت عملیات ماشین کاری انجام نمی شود. در هر لحظه با فشردن شاستی استپ نیز کل پروسه متوقف می گردد.

مد دستی: در این مد اپراتور می تواند هر کدام از موتورها را به صورت لحظه ای کنترل کند. در این مد زمان هیچ نقشی در روشن بودن موتورها ندارد. نکته قابل توجه در این مد حفاظت مدار برای موتور اصلی می باشد. فرمان استارت تجهیزات توسط شاستی های تعبیه شده در پانل مربوط به مد دستی انجام می شود. در این مد، حرکت موتور بین دو سنسور S4 و S3 نیز محدود می شود.

تعیین مد کاری: جهت تعیین مد اتوماتیک و دستی از یک کلید دو حالته استفاده می شود. زمانی که کلید در وضعیت 0 می باشد دستگاه در مد اتوماتیک و زمانی که همین کلید در وضعیت یک قرار گرفت دستگاه به مد دستی سوئیچ می شود.

I0.0 : شستی استارت (مد اتوماتیک)

I0.1 : شستی استپ

I0.2 : شستی استارت چپگرد موتور(مد دستی)

I0.3 : شستی استارت راستگرد موتور(مد دستی)

I0.4 : شستی موتور M1 (مد دستی)

I0.5 : شستی موتور M2 (مد دستی)

I0.6 : شستی موتور M3 (مد دستی)

I0.7 : سنسور مرحله ۱ (S1)

I1.0 : سنسور مرحله ۲ (S2)

I1.1 : سنسور مرحله ۳ (S3)

I1.2 : سنسور نقطه ابتدایی (S4)

I1.3 : کلید تعیین مد کاری

Q0.0 : کنتاکتور مربوط به چپگرد

Q0.1 : کنتاکتور مربوط به راستگرد

Q0.2 : موتور M1

Q0.3 : موتور M2

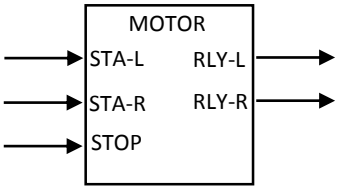
Q0.4 : موتور M3

بلوک FC1 : برنامه مربوط به مد اتوماتیک

بلوک FC2 : برنامه مربوط به مد دستی

تمرین 2- (Structured Program)

یک تابع چپگرد-راستگرد کند توسط بلوک FC طراحی و این تابع را برای ۵ موتور در OB1 با آدرس های متفاوت استفاده و نتیجه را تست کنید. هر موتور دارای ۳ ورودی و ۲ خروجی می باشد.

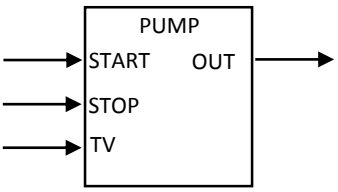


- START_L ✓
- START_R ✓
- STOP_L/R ✓
- RLY_L ✓
- RLY_R ✓

تمرین 3- (Structured Program)

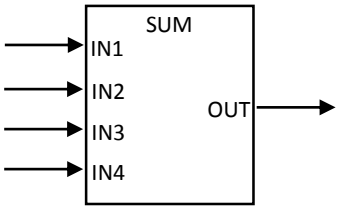
یک تابع توسط بلوک FC با لاجیک زیر برای کنترل ۵ عدد پمپ با زمان های مختلف طراحی کنید.

با اعمال فرمان استارت تابع، خروجی تابع که مربوط به پمپ می باشد، روشن و بعد از یک تاخیر TV ثانیه خاموش شود. تابع دارای ورودی استپ نیز می باشد. در ضمن لازم به ذکر است که زمان خاموش شدن پمپ ها متفاوت می باشد.



تمرین 4- (Structured Program)

توسط بلوک FC یک جمع کننده با ۴ ورودی Real طراحی کنید.

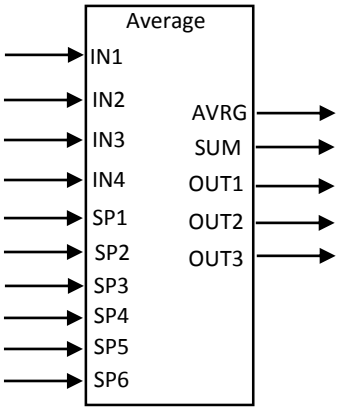


- **نکته:** در تمرین های ارائه شده تنها به I/O های اصلی تابع اشاره شده است . دانشجویان در صورت نیاز می بایست متغیرهای کمکی برای تابع در نظر بگیرند.
- **نکته:** توابع طراحی شده می بایست به گونه ای باشد که قابلیت استفاده به دفعات را در بخش های مختلف برنامه با آدرس ها و مقادیر مختلف داشته باشد.

تمرین 5- (Structured Program)

یک تابع توسط بلوک FC با لاجیک زیر طراحی کنید.

تابع FC1 مقدار ۴ ورودی Real را دریافت و میانگین آنها را محاسبه کند. در ادامه مقدار میانگین با ۶ ورودی SP که از HMI دریافت می شوند، مطابق روابط زیر مقایسه و خروجی های بیتی تابع فعال شوند.



SP1<Average<SP2 ⟶ Out1:True

SP3< Average <SP4 ⟶ Out2:True

SP5< Average <SP6 ⟶ Out3:True

در ضمن مقدار میانگین به همراه جمع ۴ ورودی نیز در خروجی تابع قابل دریافت باشد. این تابع را در OB1 به تعداد ۳ بار فراخوانی و مقادیر مختلف به آن اعمال و نتیجه را تست کنید.

تمرین 6- (Structured Program)

تمرین 5 را با استفاده از متغیرهای TEMP بازنویسی و مجدد تست کنید. ساختار برنامه را در این حالت با حالت قبل مقایسه و ساختار بهینه تر را معرفی کنید.

تمرین 7- (Structured Program)

تمرین 5 را با بلوک FB بازنویسی کنید. مزایای استفاده از بلوک FB را در این حالت بررسی کنید.

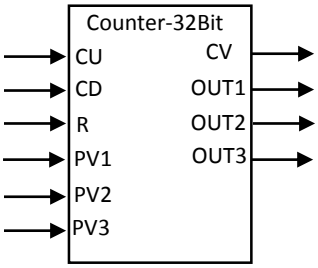
تمرین 8- (Structured Program)

تمرین 5 را با بلوک FB و با کمک متغیرهای STAT بازنویسی کنید. مزایای استفاده از متغیرهای استاتیک نسبت به TEMP در FBها را بررسی و ساختار بهتر را معرفی کنید.

تمرین 9- (Structured Program)

توسط بلوک FB یک تابع شمارنده ۳۲ بیتی بالا/پایین شمار طراحی کنید. در این شمارنده یک ورودی CU و یک ورودی CD برای دریافت پالس داریم.

همچنین از یک ورودی نیز جهت ریست شمارنده استفاده می شود. این شمارنده همچنین دارای ۳ ورودی PV و ۳ خروجی بیتی جهت ارسال فرامین می باشد. خروجی CV شمارنده مقدار جاری شمارنده را نمایش می دهد. در ضمن این شمارنده بازه منفی نداشته و مقدار CV نبایستی کمتر از 0 باشد.



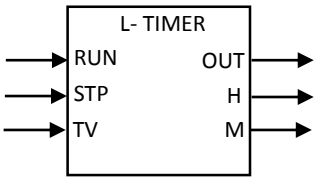
CV>=PV1 ⟶ Out1=True

CV>=PV ⟶ Out2=True

CV>=PV3 ⟶ Out3=True

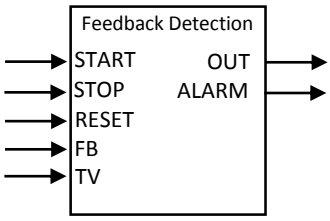
تمرین 10- (Structured Program)

توسط تابع FB یک تایمر تاخیر در قطع با قابلیت ایجاد تاخیر ماکزیمم 10000 ساعت طراحی کنید. با اعمال یک پالس به ورودی Run تابع، خروجی تابع فعال و پس از سپری شدن مدت زمان ست شده در ورودی TV، خروجی خاموش شود. مقدار زمان بر حسب ساعت به تابع اعمال می شود. خروجی تابع نیز بر حسب دقیقه و ساعت مدت زمان سپری شده را نمایش دهد. در این تابع حداقل تعداد تایمر می بایست استفاده شود.



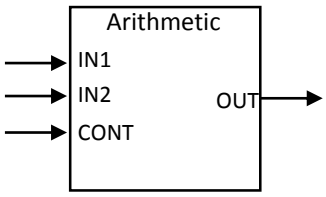
تمرین 11- (Structured Program)

توسط بلوک FC یک تابع تشخیص فیدبک طراحی کنید. این تابع دارای یک ورودی Start می باشد. با اعمال یک پالس به این ورودی، خروجی تابع فعال شود. اگر بعد از زمان ست شده در ورودی TV، فیدبک(FB) تابع برگشت، تابع خروجی را در وضعیت قبلی نگه دارد. در صورتی که فیدبک در زمان تعیین شده دریافت نشد، خروجی تابع خاموش و خروجی Alarm فعال شود. برای ریست کردن آلارم نیز یک ورودی Reset داریم. در ضمن موضوع تشخیص فیدبک در زمان خاموشی با اعمال پالس به ورودی Stop، نیز صادق می باشد. در واقع با اعمال پالس به این ورودی، خروجی تابع می بایست خاموش شود. اگر در مدت زمان تعیین شده، فیدبک برداشته نشد، خروجی آلارم تابع فعال شود.



تمرین 12- (Structured Program)

توسط بلوک FB1 یک تابع با قابلیت انجام ۴ عمل ریاضی طراحی کنید. این تابع دارای ۲ ورودی Real و یک پایه کنترلی Integer می باشد. تابع می بایست به گونه ای طراحی شود که با مقدار دادن به پایه کنترلی بتوان نوع عملیات ریاضی را تعیین کرد.



Control=0 ⟶ IN1+IN2

Control=1 ⟶ IN1-IN2

Control=2 ⟶ IN1×IN2

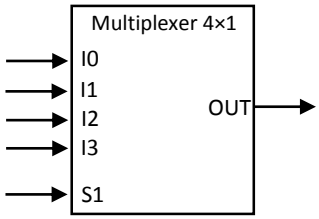
Control=3 ⟶ IN1÷IN2

تمرین13- (Multiple Instance)

تمرین ۲ را با بلوک FB مجدد بازنویسی و این تابع را برای ۳ موتور با نام های MOTOR-1010 ، MOTOR-1011 و MOTOR-1012 در OB1 فراخوانی کنید. مزایای استفاده از روش Multiple Instance را بررسی کنید.

تمرین14- (Structured Program)

توسط بلوک FB، یک تابع مالتی پلکسر 4 به 1 طراحی کنید. این مالتی پلکسر دارای ۴ ورودی دیتا(I0 تا I3) به صورت Real می باشد. با قرار دادن مقادیر 0 تا 3 صحیح به ورودی S1، می توان مقادیر ورودی را در خروجی مالتی پلکسر قرار داد. به عنوان مثال زمانی که ورودی S1=2 باشد، مقدار I2 به خروجی منتقل شود. ورودی S1 یک ورودی Integer می باشد.

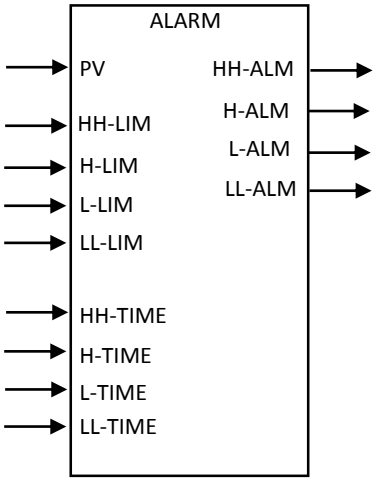


تمرین15- (Structured Program)

توسط بلوک FB، یک تابع تولید آلارم طراحی کنید. این تابع یک مقدار Real را از یک سنسور(PV) دریافت می کند. مقدار دریافت شده با ۴ مقدار زیر مقایسه می شود.

- HH_Limit
- H_Limit
- L_Limit
- LL_Limit

به ازای هر کدام از محدوده های ذکر شده، یک خروجی Alarm وجود دارد. در صورتی که مقدار ورودی(PV) بیشتر از هر کدام از بازه های تعیین شده باشد، آلارم مربوطه بعد از گذشت زمان تعیین شده، فعال شود.



خروجی های آلارم:

- HH_Alarm
- H_Alarm
- L_Alam
- LL_Alarm

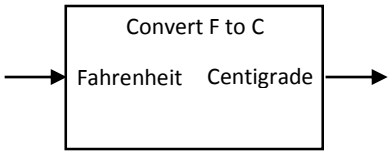
ورودی های زمان :

- HH_Time
- H_Time
- L_Time
- LL_Time

به عنوان مثال فرض کنید، مقدار H_Limit =150.0 و مقدار H_Time =10s باشد. در این صورت زمانی که مقدار ورودی PV به مدت ۱۰ ثانیه بیشتر از 150.0 باشد، آلارم H_Alarm فعال شود. در ادامه با کمتر شدن مقدار PV نسبت به مقدار Limit، آلارم به صورت اتوماتیک برطرف شود.

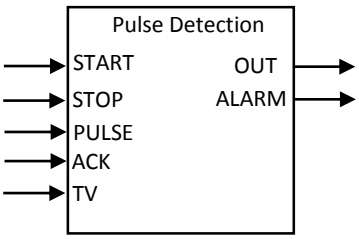
تمرین 16- (Structured Program)

در بخش های مختلف برنامه لازم است که مقدار دمای دریافتی در واحد فارانهایت به سانتیگراد تبدیل و در برنامه استفاده شود. یک تابع با نام Convert توسط بلوک FC یا FB ایجاد کنید که مقدار Real دریافتی بر حسب فارانهایت را دریافت و خروجی را بر حسب سانتیگراد برگرداند.



تمرین 17- (Structured Program)

توسط بلوک FC یک تابع تشخیص وجود پالس طراحی کنید. در بسیاری از پروژه ها جهت تشخیص چرخش موتور، از یک سنسور در مقابل شفت موتور استفاده می کنند. چرخش شفت موتور و عبور زائده تعبیه شده از جلوی سنسور، باعث تولید پالس می شود. تابع دارای یک ورودی استارت و یک ورودی استپ جهت روشن و خاموش کردن موتور می باشد. با اعمال فرمان استارت، خروجی تابع فعال شود. با روشن بودن خروجی، پالس های ارسالی به ورودی Pulse تابع چک می گردد. اگر در مدت زمان ست شده در ورودی TV، ورودی پالس تغییر نکرد، موتور خاموش و خروجی آلارم فعال شود.



به عنوان مثال فرض کنید مقدار TV بر روی 5 ثانیه تنظیم شده است. اگر در مدت زمان 5 ثانیه ورودی Pulse تابع در وضعیت 1 یا 0 ثابت بود، تابع می بایست این وضعیت را به عنوان خطا شناسایی و فرمان را از روی خروجی تابع بردارد. حالت نرمال زمانیست که در مدت زمان 5 ثانیه، تابع مدام تغییرات ورودی پالس را تشخیص دهد. یک ورودی ACK نیز برای برطرف کردن خطای تابع وجود دارد.

تمرین 18- (DB-Absolute Addressing)

یک DB **بدون** Symbolic Name ایجاد و ۴ سطر Real جهت ذخیره ۴ مقدار حاصل از عملیات ریاضی ایجاد کنید. در بلوک OB1 عملیات ریاضی را به دلخواه طراحی و نتیجه را به سطرهای DB منتقل و نتایج را در وضعیت آنلاین مشاهده کنید.

تمرین 19- (DB-Symbolic Addressing)

یک DB با نام Data ایجاد و ۴ سطر Real جهت ذخیره ۴ مقدار حاصل از عملیات ریاضی ایجاد کنید. در بلوک OB1 عملیات ریاضی را طراحی و نتیجه را به سطرهای DB منتقل و نتایج را در وضعیت آنلاین مشاهده کنید. آدرس دهی این تمرین را با تمرین 18 مقایسه و روش بهتر را معرفی کنید.

تمرین20- (DB-Complex Type)

یک DB اشتراکی با نام Temperature ایجاد کنید. در این DB قرار است مقدار دریافت شده از 50 ترموکوپل ثبت شوند. بستر ثبت دیتا را در این DB آماده کنید. در ضمن در برنامه توسط دستور MOVE چند عدد به دلخواه به سطرهای مختلف این DB ارسال و نتیجه را مشاهده کنید.

تمرین 21-(DB-Complex Type)

در یک پروژه قرار است اطلاعات ۱۰ عدد درایو را در یک DB ثبت کنیم. از هر درایو پارامترهای زیر مورد نیاز می باشند.

- ✓ (Bool) Running
- ✓ (Bool) Fault
- ✓ (Bool) Direction
- ✓ (Real) Frequency
- ✓ (Real) Speed

- ✓ (Int) Current
- ✓ (Int) Voltage

مراحل ثبت دیتای درایوها را در ۳ حالت در دیتابلاک های مختلف بررسی کنید:

حالت ۱:

- از تایپ Elementary برای ثبت دیتاها در DB1 استفاده کنید.

حالت۲:

- از تایپ STRUCT برای ثبت دیتاها در DB2 استفاده کنید.

حالت۳:

- از UDT برای ثبت دیتاها در DB3 استفاده کنید.

نتایج حالت های مختلف را با هم مقایسه و بهترین ساختار را معرفی کنید.

تمرین 22-(DB-Complex Type)

در یک DB با نام Data، یک ماتریس با ۵ سطر و ۳ ستون ایجاد کنید. در برنامه مقدار اعشاری 27.25 را به [2,3] و مقدار 100.0 را هم به [5,3] منتقل و نتیجه را در حالت آنلاین مشاهده کنید.

موفق و سربلند باشید