sbargh.ir
sbargh.ir@yahoo.com

# جزوه PLC درجه۲

## مطابق با استاندارد سازمان فنی وحرفه ای

## تهیه شده در واحد آموزشی طبس

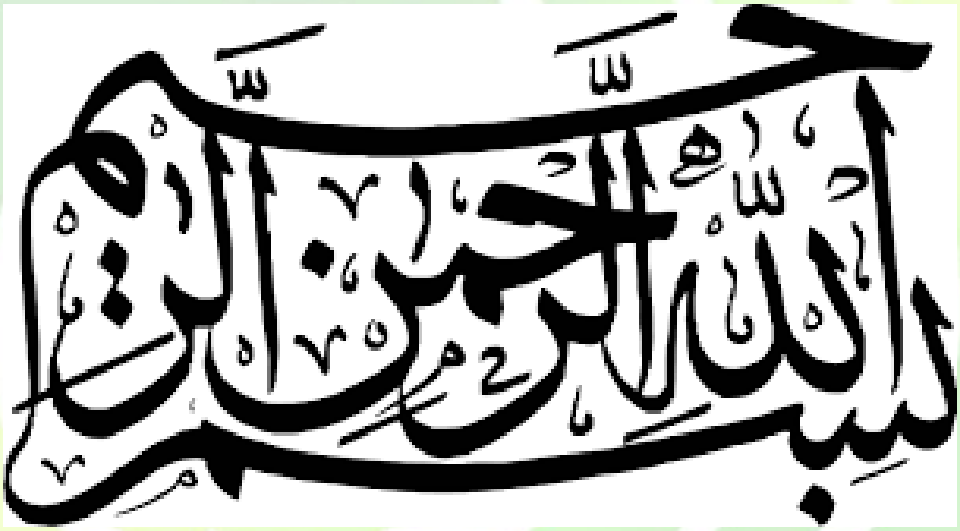
## مدرس: دکتر سالکی

## گردآوری و تنظیم: مسعود پرویزی مقدم

## تابستان ۱۳۹۷

جزوه PLC تهیه شده در مرکز آموزش فنی و حرفه ای طبس

مدرس: دکتر سالکی  جمع آوری و تنظیم: مسعود پرویزی مقدم



sbargh.ir

sbargh.ir

## انواع حافظه ها

- RAM: حافظه ای موقت که هم خواندنی و هم نوشتنی است. با قطع تغذیه، اطلاعات آن از بین خواهد رفت.
- ROM: حافظه ای دائم که فقط خواندنی است (Read Only Memory). فقط یکبار و توسط کارخانه سازنده برنامه نویسی می شود.
- PROM: حافظه ای دائم که فقط خواندنی است و فقط یکبار توسط کاربر برنامه نویسی می شود.
- EPROM: حافظه ای خواندنی و نوشتنی است. بصورت الکتریکی برنامه نویسی شده و توسط اشعه فرابنفش پاک می شود.
- EEPROM: حافظه ای خواندنی و نوشتنی است. بوسیله جریان الکتریکی برنامه نویسی شده و با آن نیز پاک می شود. تعداد دفعاتی که حافظه ی EEPROM قابل برنامه نویسی است بیشتر از حافظه ی EPROM است.
- FLASH: حافظه هایی خواندنی و نوشتنی اند. اطلاعات آن با جریان الکتریکی نوشته و پاک می شود. تکنولوژی بالاتر و حجم بیشتری دارند.
  - ♦ مثال: هارد کامپیوتر، حافظه ی فلش

## مبانی اعداد

- **مبنای ۱۰:** در این مبنا، ده ابزار برای شمردن یا نشان دادن اطلاعات داریم:

sbargh.ir

بزرگترین مقدار در مبنای ۱۰ با دو جایگاه: ۹۹

جایگاه ها در توان های مبنا اضافه می شوند.

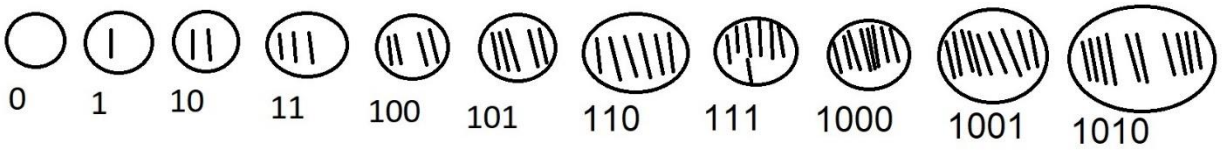
**روش رقم مبنا:**

$$\text{ارزش هر عدد} = \text{رقم} * \text{مبنا}^{\text{نما}}$$

**نما** را میتوان **تعداد ارقام قبل از عدد** تعریف کرد.

مثال: $۴ * ۱۰^{۲} + ۹ * ۱۰^{۱} + ۲ * ۱۰^{۰} = ۴۹۲$

- **مبنای ۲ (باینری):** در این مبنا، دو ابزار داریم: ۰ و ۱




جایگاه ها در مبنای دو:

۱ و ۲ و ۴ و ۸ و ۱۶ و ۳۲ و ۶۴ و ۱۲۸ و ۲۵۶ و ۵۱۲ و ۱۰۲۴ و ۲۰۴۸ و ۴۰۹۶ و ... .

sbargh.ir

مثال:

اعداد زیر را به مبنای ده ببرید.

$(101101)_2 = (\quad)_{10}$

$(111101)_2 = (\quad)_{10}$

- روش بسته ها: یعنی بررسی می کنیم که چند بسته چندتایی در عدد وجود دارد.

مثال:

اعداد زیر را به مبنای دو ببرید.

$(74)_{10} = (\quad)_2$

$(129)_{10} = (\quad)_2$

- روش تقسیمات متوالی: بدین گونه که تقسیمات پشت سرهم (یعنی خارج قسمت تقسیم قبل به عنوان مقسوم تقسیم بعد استفاده می شود. مقسوم علیه عددی خواهد بود که میخواهیم عدد را در مبنای آن به دست بیاوریم.) انجام می شود تاجایی که خارج قسمت صفر گردد. آنگاه باقیمانده ها را از آخرین به اولین مرتب می کنیم.

مثال: با استفاده از تقسیم های متوالی، عدد ۹۴ را از مبنای ۱۰ به مبنای ۲ ببرید.

مثال: با استفاده از روش رقم مبنا عدد زیر را از مبنای ۲ به مبنای ۱۰ منتقل کنید.

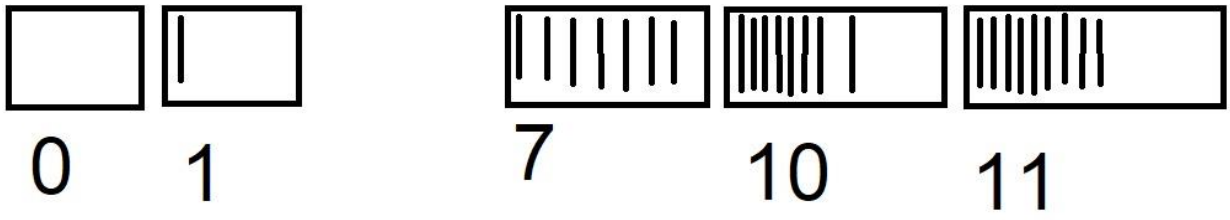
$(1110111)_2 = (\quad)_{10}$

$۱*۲^۶ + ۱*۲^۵ + ۱*۲^۴ + ۰*۲^۳ + ۱*۲^۲ + ۱*۲^۱ + ۱*۲^۰ = ۱۱۹$

- **مبنای ۸ (اکتان):** از هشت ابزار برای نمایش اطلاعات استفاده می کنیم.

sbargh.ir

۰،۱،۲،۳،۴،۵،۶،۷

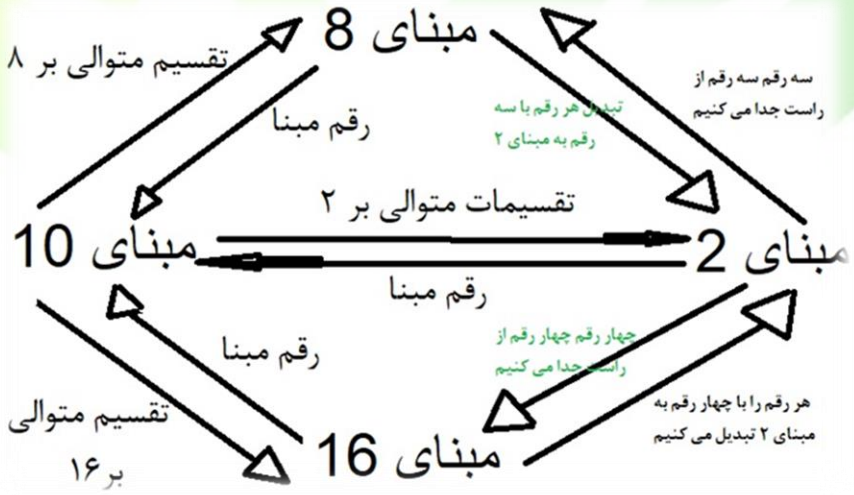



## جایگاه ها در مبنای ۸: یکی ۸تایی ۶۴تایی

مثال: عدد ۴۳۲ را به مبنای ۸ ببرید.

مثال: اگر ۳۲۱ در مبنای ۸ قرار داشته باشد، معادل آن را درمبنای ۱۰ بیابید.

برای تبدیل از مبنای ۸ به مبنای ۲ میتوان هر رقم در مبنای ۸ را به ۳ رقم در مبنای ۲ تبدیل کرد.



sbargh.ir

## سایز انواع متغیرها

- بیت: کوچکترین واحد حافظه که فقط دو حالت دارد. یا صفر یا یک.
- بایت: هر هشت بیت را یک بایت گوییم.
- Word: هردوبایت یا به عبارتی هر ۱۶ بیت را یک Word گوییم.
- Dword (دابل ورد): هر دو Word یا هر ۴ بایت یا هر ۳۲ بیت را یک Dword می نامیم.

| تعداد بیت | تعداد حالت | بزرگترین عدد |
|---|---|---|
| ۱ | ۱ | ۱ |
| ۲ | ۴ | ۳ |
| ۳ | ۸ | ۷ |
| . | . | . |
| . | . | . |
| n | $۲^{n}$ | $۲^{n}-۱$ |

## عملگرها

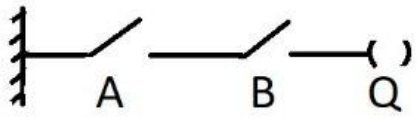
مثال: در عبارت ۸=۵+۳ به علامت جمع، عملگر(Operator) و به اعداد بکار رفته، عملوند(Operand) می گوییم.

- عملگرهای منطقی

در طی نوشتن برنامه، حالت هایی پیش رو داریم که باید براساس آن تصمیم گیری شود.

sbargh.ir

- And: هم این هم آن، هردو باهم، همه باهم




عملوند: بیت

در این حالت، جدول صحت عملکرد بدین گونه خواهد بود:

اگر حالت عادی را حالت تحریک نشده(معادل صفر) تعریف کنیم، آنگاه:

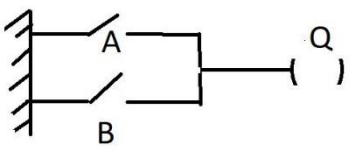
NC: در حالت عادی بسته

NO: در حالت عادی باز

تعریف می شوند.

| A | B | Q |
|---|---|---|
| ۰ | ۰ | ۰ |
| ۰ | ۱ | ۰ |
| ۱ | ۰ | ۰ |
| ۱ | ۱ | ۱ |

- OR: یا این یا آن، حداقل یکی

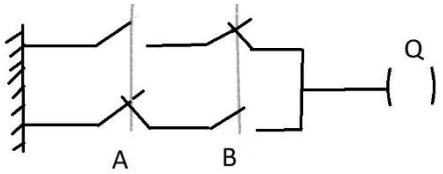



| A | B | Q |
|---|---|---|
| ۰ | ۰ | ۰ |
| ۰ | ۱ | ۱ |
| ۱ | ۰ | ۱ |

sbargh.ir

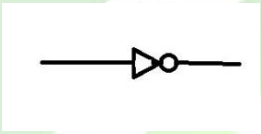
| | | |
|---|---|---|
| ۱ | ۱ | ۱ |

- XOR: فقط یکی، یای انحصاری




| A | B | Q |
|---|---|---|
| ۰ | ۰ | ۰ |
| ۰ | ۱ | ۱ |
| ۱ | ۰ | ۱ |
| ۱ | ۱ | ۰ |

- Not: عملگر بیت را معکوس می کند. برای خواندن صفر ازین عملگر استفاده می شود.



| A | Not |
|---|---|
| ۱ | ۰ |
| ۰ | ۱ |

## نواحی مختلف PLC

- ناحیه ی ورودی(I): کلید، سنسورها، و فرمان هایی که کاربر به هرطریقی به آن می دهد.
- ناحیه ی خروجی(Q): برای ایجاد فرمان های صادر شده بکار می رود.
- ناحیه ی حافظه ی داخلی (M) Flag: برای ذخیره ی فرمان های میانی استفاده می شود.

## روش های آدرس دهی:

- **آدرس دهی بیتی**: بصورت {شماره بیت . شماره بایت ناحیه} انجام می شود.

sbargh.ir

مثال:

| ناحیه ی I | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| ۰ | ۷ | ۶ | ۵ | ۴ | ۳ | ۲ | ۱ | ۰ |
| ۱ | | | | | | | | |
| ۲ | | | | | | | | |
| ۳ | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |

در جدول بالا برای آدرس دهی بیت مشخص شده از آدرس I 0.2 استفاده می کنیم.

| ناحیه ی M | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| ۰ | ۷ | ۶ | ۵ | ۴ | ۳ | ۲ | ۱ | ۰ |
| ۱ | | | | | | | | |
| ۲ | | | | | | | | |
| ۳ | | | | | | | | |
| ۴ | | | | | | | | |
| ۵ | ۷ | ۶ | ۵ | ۴ | ۳ | ۲ | ۱ | ۰ |
| | | | | | | | | |

در این جدول از آدرس M 5.5 استفاده می کنیم.

- **آدرس دهی بایتی:** بصورت {شماره بایت B ناحیه} انجام می پذیرد.

در جدول زیر از آدرس I B 19 برای آدرس دهی استفاده می کنیم.

| ناحیه ی I | |
|---|---|
| ۰ | |
| ۱ | |

sbargh.ir

| | |
|---|---|
| ۲ | |
| ۳ | |
| | |
| | |
| ۱۹ | |
| ۲۰ | |
| | |
| | |

همچنین برای آدرس دهی جدول ذیل از آدرس Q B 10 استفاده می کنیم.

| ناحیه ی Q | |
|---|---|
| ۰ | |
| ۱ | |
| ۲ | |
| ۳ | |
| | |
| ۱۰ | |
| | |

- **آدرس دهی Word:** بصورت {شماره بایت آغازین W ناحیه} انجام می پذیرد.

به عنوان مثال در جدول ذیل از آدرس M W 11 برای آدرس دهی استفاده می کنیم.

| ناحیه ی M | |
|---|---|
| ۰ | |

| | |
|---|---|
| ۱ | |
| ۲ | |
| ۳ | |
| | |
| ۱۱ | |
| ۱۲ | |
| | |

در هنگام آدرس دهی مراقب تداخل آدرس باشید.

به عنوان مثال دو آدرس M W 3 و M W 4 دارای بایت مشترک ۴ می باشند.

توجه داشته باشید که برنامه به این تداخل هیچگونه خطایی نشان نمی دهد اما در اجرا با مشکل مواجه می شود.

- **آدرس دهی Dword:** بصورت {شماره بایت آغازین D ناحیه} انجام می پذیرد.

به عنوان مثال برای آدرس دهی جدول زیر از آدرس Q D 8 استفاده می شود.

| ناحیه ی Q | |
|---|---|
| ۰ | |
| ۱ | |
| ۲ | |
| | |

| | |
|---|---|
| ۸ | |
| ۹ | |
| ۱۰ | |
| ۱۱ | |
| | |

# کنترل

**کنترل** به معنای **تحت فرمان و اختیار در آوردن سیستم ها و پروسه ها** می باشد.

**مدار فرمان** : مداری که فرامین کنترلی را برای مدار قدرت صادر می کند.

- کنترل صنعتی:
- کنترل دستی
- کنترلر های الکترو مکانیکی ( رله، کنتاکتور، پنوماتیک، هیدرولیک و ... )
- کنترلر های الکترونیکی ( میکرو کنترلر ها )
- کامپیوتر
- PLC (کنترلرهای منطقی برنامه پذیر)

## معایب مدارات فرمان الکترومکانیکی :

- حجم بالای مدار فرمان برای پروسه های کنترلی بزرگ و پیچیده
- بالارفتن هزینه تمام شده
- مصرف بیشتر انرژی در مقایسه با مدارات الکترونیکی

- استهلاک قطعات مکانیکی
- فرآیند پیچیده و طولانی نصب و راه اندازی
- عیب یابی سخت و مشکل
- کاربرد منحصر به فرد و غیر قابل استفاده برای پروسه های دیگر
- هزینه بالا و اتلاف وقت زیاد برای توسعهٔ مدار فرمان
- و ....

## مزایای استفاده ازPLC :

- کاهش حجم تابلو و مدارات فرمان
- صرفه جویی در هزینه ، لوازم و قطعات
- نداشتن استهلاک مکانیکی ،عمربیشترونیاز کمتر به تعمیرات
- انرژی مصرفی کمتر
- عدم ایجاد نویز الکتریکی و صوت
- استفاده برای پروسه های گوناگون فقط با تغییربرنامه
- روش های عیب یابی آسان و ساخت یافته
- تشخیص عیب درسایرقسمت های پروسه تحت کنترل توسط خود PLC

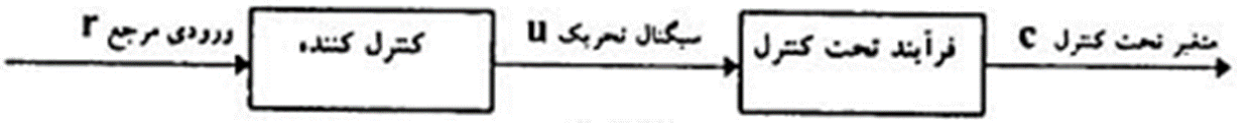
# مزایای PLC نسبت به سایر کنترل کننده ها

| | کامپیوتر | مدارهای منطقی الکترونیکی | مدارهای رله‌ای | PLC |
|---|---|---|---|---|
| قیمت با توجه به عملکرد | گران قیمت | ارزان | نسبتاً ارزان | ارزان |
| حجم و ابعاد | نسبتاً کوچک | خیلی کوچک | بزرگ و حجیم | خیلی کوچک |
| سرعت کنترل | خیلی سریع | نسبتاً سریع | کند | خیلی سریع |
| نویز الکتریکی | کاملاً خوب | خوب | عالی | خوب |
| نصب وبهره‌برداری | برنامه‌نویسی مشکل است | طراحی مشکل است | طراحی ونصب مشکل است | نصب وبرنامه‌نویسی ساده است |
| توانایی محاسبات پیچیده را دارد؟ | آری | خیر | خیر | آری |
| تغییر نحوهٔ کنترل و ایجاد تغییرات | آسان | مشکل | خیلی مشکل | بسیار آسان |

## سیستم های کنترل از لحاظ ساختاری

الف – سیستم های کنترل حلقه باز :

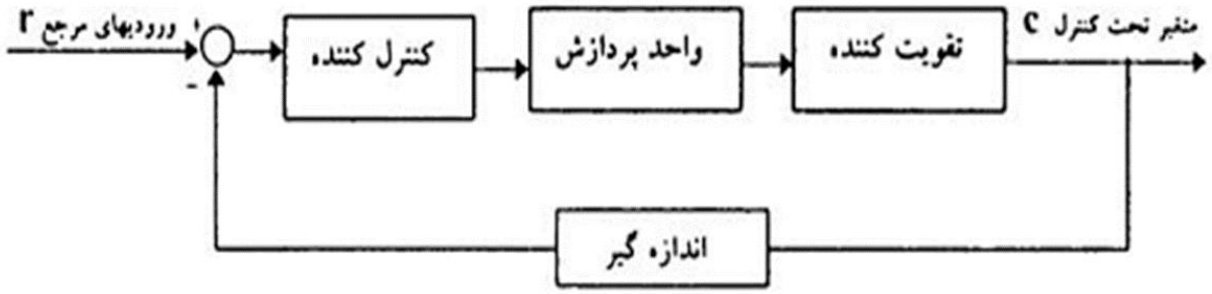
خواسته های ما از سیستم کنترل به خوبی برآورده نمی شود و تنها به دلیل سادگی و اقتصادی بودن از آنها استفاده می کنیم. کنترل سیستم های حلقه باز در مقابل اغتشاشات بیرونی مشکل تر است.




دیاگرام یک سیستم کنترل حلقه باز ساده

ب – سیستم های کنترل حلقه بسته:

آنچه برای کنترل دقیق تر و قابل انعطاف تر لازم است یک فیدبک از خروجی به ورودی سیستم است . سیستمی با یک یا چند مسیر فیدبک ، سیستم کنترل حلقه بسته می نامیم

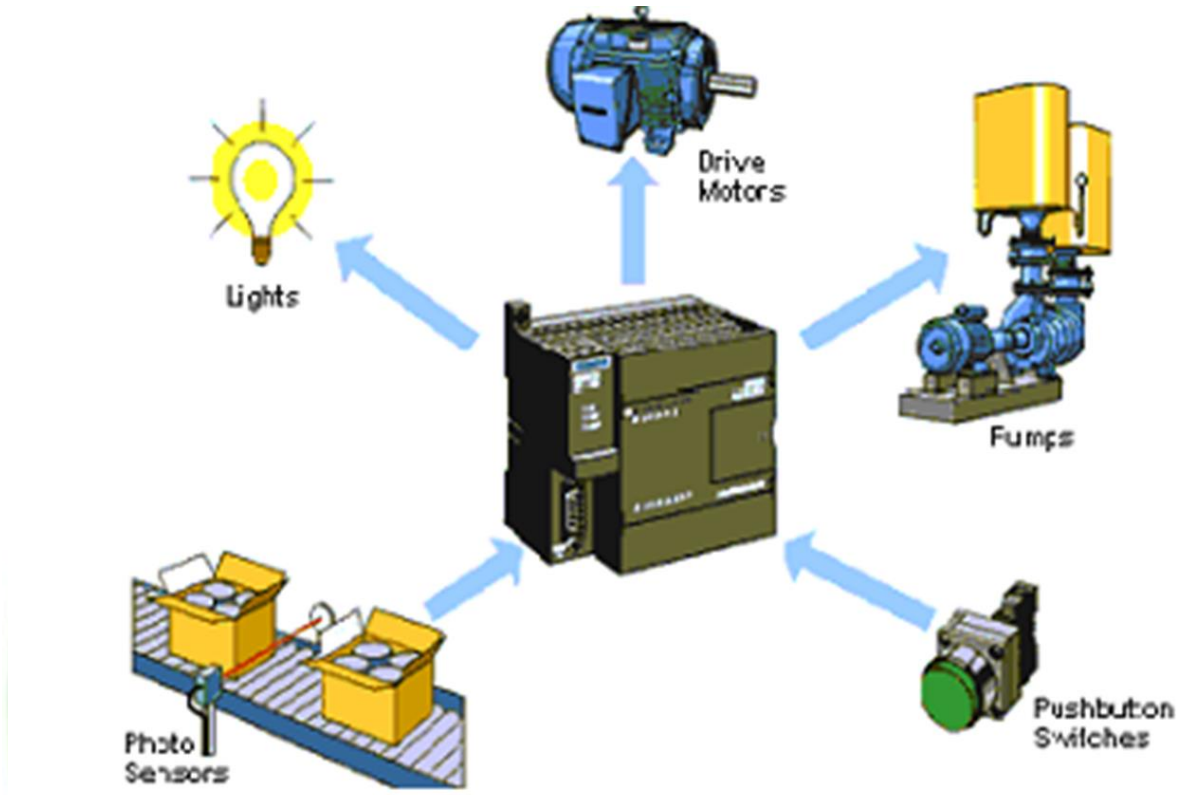



دیاگرام یک سیستم کنترل حلقه بسته

## *PLC و تاریخچه ی آن*

ازعبارت Programmable Logic Controller به معنی کنترل‌کننده منطقی قابل برنامه ریزی گرفته شده اسـت.

کنترل کننده ای نرم افزاری از خانواده کامبیترها که عمدتاً در مقاصد صـنعتی بکار می رود . درقسمت ورودی اطلاعاتی را به صورت باینری دریافت و آنها را طبق برنامه ای که درحافظه اش ذخیره شـده پردازش می نماید و نتیجه عملیات را نیز از قسمت خروجی برای ماشین یا




فرایندهایی که تحت کنترل آن است صادر می نماید.

۱۹۶۸ = ساخت و استفاده از اولینPLC ها در صنایع خودروسازی

دهۀ ۷۰ = اضافه شدن قابلیت برقراری ارتباط ( شبکه شدن ) به PLC

دهۀ ۸۰ = استاندارد شدن پروتکل های ارتباطی

دهۀ ۹۰ = ایجاد استاندارد IEC61131

## استاندارد IEC61131

این استاندارد در سال ۱۹۹۱ در کمیتهٔ بین المللی الکترو تکنیک ( IEC ) شکل گرفت و شامل بخش های زیر است :

بخش ۱ – اطلاعات کلی

بخش ۲- ملزومات سخت افزاری و آزمایشها

بخش ۳- زبانهای برنامه نویسی

بخش ۴- راهنمای کاربران

بخش ۵- ارتباطات

بخش ۶- رزرو

بخش ۷- برنامه نویسی کنترل فازی ( ۲۰۰۱ )

بخش ۸-راهنمای کاربرد زبانهای برنامه نویسی

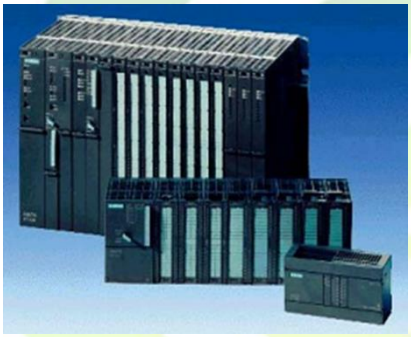
## برخی از شرکت های سازندهٔ PLC

| | |
|---|---|
| زیمنس ( SIEMENS ) آلمان | S7200,300,400,LOGO |
| OMRON ژاپن | ZEN |
| MITSUBISHI ژاپن | ALPHA |
| Schneider (Telemecanique) | ZELIO |
| THEBEN | PHARAO |
| MOELLER | EASY |
| Allen Bradly | |

# PLCهای زیمنس

## Simatic S5 :

این نوع دارای مدلهایی چون S5-90U , S5-95U به صورت Compactو S5-115U,S5-100U به صورت Modular می باشد .برنامه نویسی آن با STEP 5 انجام می گیرد.



## Simatic S7 :

به سه خانواده تقسیم می شوند S7-200. برای سیستم های کنترلی کوچک بکار می رود و باMicro win برنامه نویسی می شود S7-300 .عملکرد متوسط دارد و S7-400 که حوزه عملکرد وسیعی دارد و هر دو با Simatic Manager برنامه نویسی می شوند.

## LOGO:

یک کنترلر ساده و ارزان قیمت که برای کارهای کنترلی کوچک (ساختمان ها یا ماشین های کوچک) بکارمی رود. برنامه ریزی LOGO توسط کلیدهای روی آن و یا برنامه LOGO!Soft Comfort انجام می شود.

## Simatic C7 :

ترکیبی از S7-300 و Operator Control.علاوه بر اینکه کار کنترلی انجام می دهد بر روی نمایشگر آن می توان پیغام ها، رخدادها و مقادیر مربوط به فرآیند را دید. C7 کمپکت بوده و انواع مختلف دارد. برای برنامه نویسی آن علاوه برSTEP7 به نرم افزار Protool هم نیاز است.

## خانواده S7

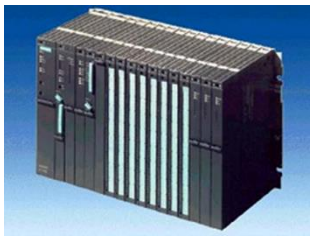
S7-200 : یک micro plc ارزان قیمت

S7-300 :یک mini plc با حوزه عملکرد متوسط

S7-300F: برای سیستم های نیازمند به ایمنی بالا.

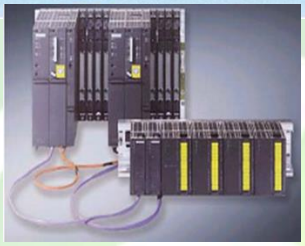
درانتهای کد CPU حرف F معرف این نوع است.مثلا CPU 315F

S7-300C: CPU با مدول ورودی/ خروجی به صورت کمپکت عرضه شده است.در انتهای کد CPU حرف C معرف این نوع است.



S7-400: یک PLC با حوزه عملکرد وسیع

S7-400H: برای پروسه های خیلی حساس

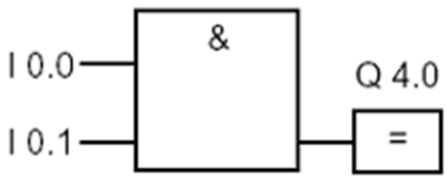


S7-400FH: برای کاربردهای با درجه ایمنی بالا

## زبانهای برنامه نویسی در STEP7

۱ – FBD (Function Block Diagram) :

یک نمایش گرافیکی از دستورات و توابع STEP7 است و از نمادهایی به شکل مربع برای هر عمل منطقی یا دستور استفاده می کند.در این روش توابع ریاضی و منطقی به راحتی توسط این جعبه ها نمایش داده می شوند.



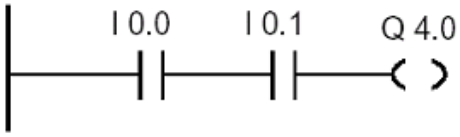

برای مثال بلوک مقابل ورودیهایش را با هم AND می کند.

این روش در STEP5 به روش CSF معروف است.

**۲**– LAD (LADDER **یا نردبانی**) :

این روش نیز یک نمایش گرافیکی از دستورات برنامه نویسی است و بسیار شبیه به منطق نردبانی دیاگرام رله ای است. توسط LAD می توان مسیر عبور جریان از منبع به عناصر مختلف را تعقیب نمود.

نمونه ای از دستورات به زبان LAD :




۳- STL ( Statement List ) :

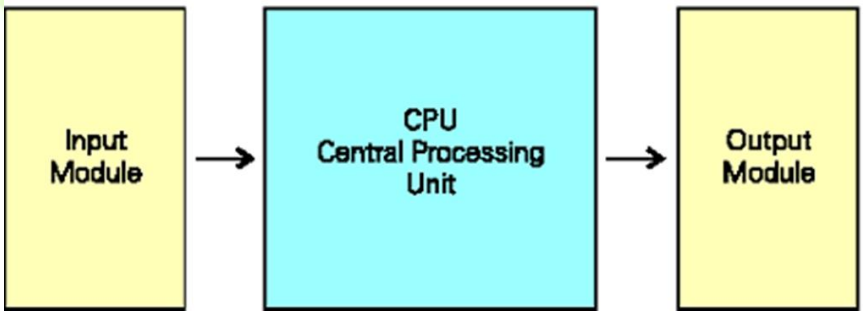
این روش یک نمایش متنی از دستورات برنامه نویسی است و وظایف کنترلی توسط عبارات و دستورات کنترلی مجزا مشخص می شوند. هر عمل توسط یک یا چند حرف مخفف نمایش داده می شود.برای مثال دستور مقابل دو بیت را با هم AND کرده و نتیجه را در بیت خروجی قرار می دهد.

```
A    I 0.0
A    I 0.1
=    Q 4.0
```

یک برنامه را می توان به هر یک از سه زبان بیان شده نوشت. زبانهای LAD و FBD قابل تبدیل به یکدیگر و قابل تبدیل به STL می باشند. STL کامل تر از دو روش دیگر است و محدودیت های آنها را ندارد اما در برخی موارد قابلیت تبدیل به دیگر زبان ها را ندارد.

زبانهای تکمیلی نظیر SCL-SFC-Graph-Higraph نیز به همراه برنامه STEP7 ارائه می شوند.

## بلوک دیاگرام کلی PLC




**ترمینال های ورودی: ( Input Module )**

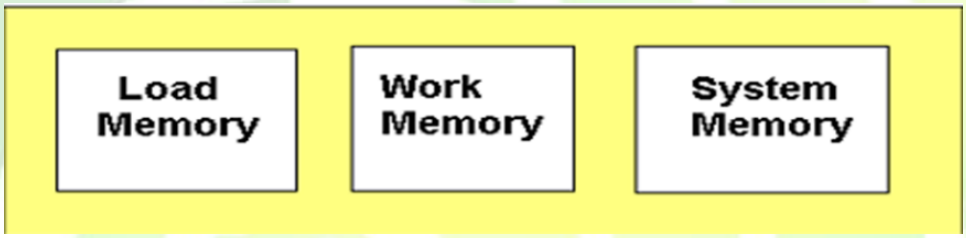
این واحد محل دریافت اطلاعات فرآیندها یا پروسه تحت کنترل است.تعداد و نوع ورودیها در PLC های مختلف متفاوت است .

## ترمینالهای خروجی: ( Output Module )

این واحد محل صدور فرمان های PLC به پروسه تحت کنترل می باشد. تعداد و نوع خروجیها نیز برای PLC های مختلف متفاوت است.

## واحد پردازندۀ مرکزی: ( CPU )

محل ذخیره ، پردازش و ارسال اطلاعات است . حافظۀ CPU های S7 ساختاری شبیه شکل زیر دارد:

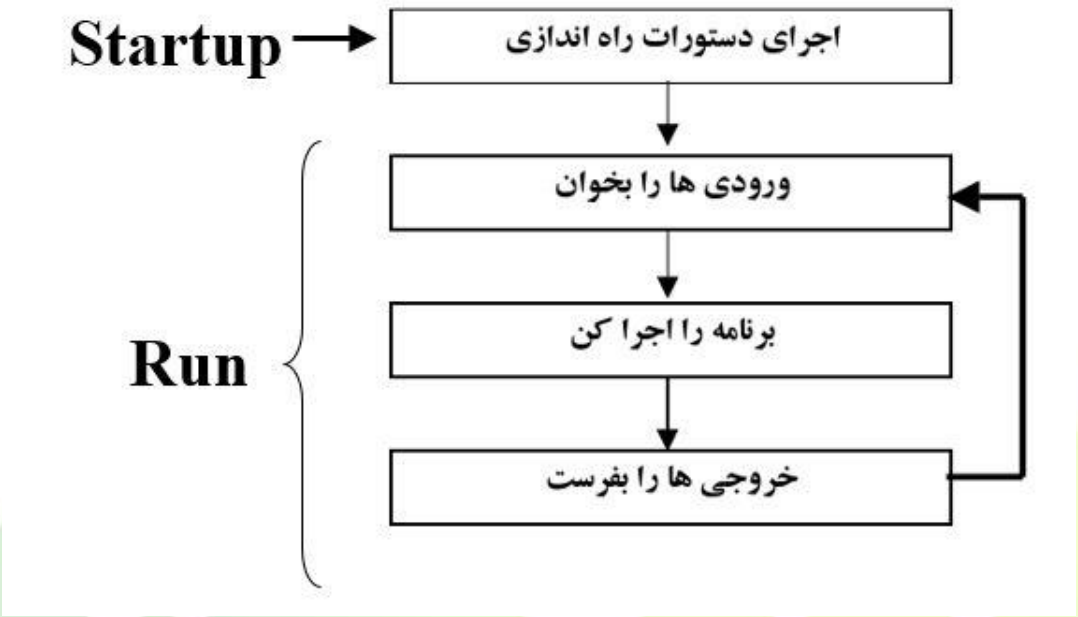



Load Memory: وقتی برنامه به CPU ارسال (Download) می شود در این قسمت قرار می گیرد. امروزه برای این قسمت از کارت حافظه استفاده می شود

Work Memory: این حافظه بخشی از برنامه را که اجرایی است در برمی گیرد. مثلاً یک فانکشن فقط وقتی که برای اجرا صدا زده می شود از Load Memoryدر این بخش حافظه قرار می گیرد.در صورت ریست شدن محتویات این بخش پاک می شود

System Memory: این بخش عناصر حافظه مانند جداول PIQ، PII، تایمرها ، کانترها ، فلگ ها و ... را در بر می گیرد. در صورت ریست شدن محتویات این بخش نیز پاک می شود.

# مراحل اجرایی PLC پس از روشن شدن




## آشنایی با برخی از مفاهیمPLC

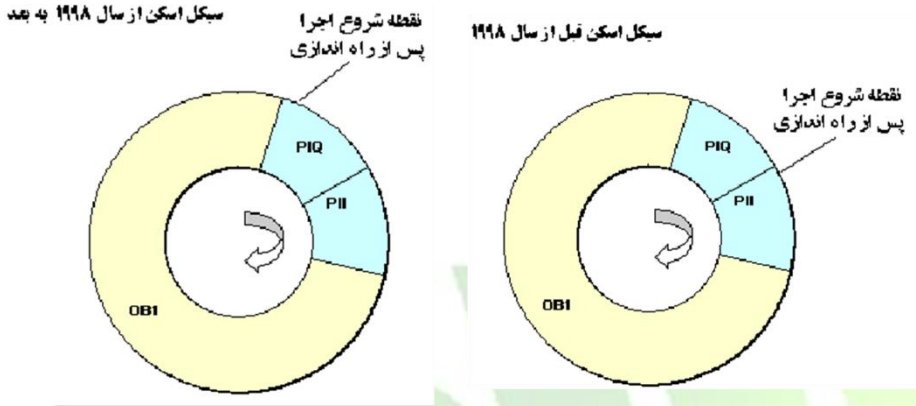
### PII (Process Image Input):

در هر سیکل CPU تصویری از ورودیها را گرفته و در این قسمت ذخیره می کند . ( مثل اینکه از مقادیر ورودی در یک لحظهٔ کوتاه عکس برداری شود) و در اجرای برنامه از این مقادیر استفاده می شود.مقادیر موجود در PII در طول یک سیکل اسکن تغییر نمی کنند.

### PIQیا PIO (Process Image Output) :

پس از اجرای برنامه، خروجی های تولید شده در این بخش ذخیره شده و از آنجا به کارت های خروجی ارسال می گردند. مقادیر موجود در PIQ در سیکل اسکن بعد تغییر نمی کنند.

# سیکل اسکن در CPU های ساخت زیمنس




در حالت Run عملاً تغییری به وجود نیامده ، ولی سیکل اسکن دوم PLC می تواند خروجی های احتمالی ایجاد شده در حالت راه اندازی را بفرستد و بعد برنامه را اجرا کند.

**تعریف HMS:** رابط انسان و PLC است. برای اینکه ببینیم در سیستم چه اتفاقی می افتد.

نکته: باتری Back Up باید دارای ولتاژ ۲,۸ تا ۳,۶ ولت باشد.

**تعریف Scan Cycle:** هر بار اجرای برنامه از خط اول تا آخر و دوباره برگشتن به همان خط.

**تعریف Scan Time:** به مدت زمان اجرای Scan Cycle گفته می شود. به بیان دیگر: مدت زمانی که طول می کشد تا یک برنامه اجرا شود.

حافظه های Retentive پایدار و غیرفرار (Non-Volatile) هستند.

حافظه های Non-Retentive ناپایدار و فرار (Volatile) هستند.

**PLC جایگزین مدار قدرت نمی شود.**

sbargh.ir

# شروع کار با نرم افزار

نرم افزار Simatic Manager را می توان نرم افزار مادر فرض کرد. زیرا با استفاده از آن می توان تمامی نرم افزارهای Step 7 را باز و اجرا نمود.

MPI 1 نوعی شبکه که بصورت پیش فرض برای ارتباط کامپیوتر با PLC در نظر گرفته می شود.

پس از بازکردن نرم افزار،برای ایجاد پروژه جدید از سربرگ File گزینه New را انتخاب نمایید.
سپس برای مشخص کردن Station از سربرگ Insert گزینه Station و سپس ۳۰۰ را انتخاب نمایید.

حال برروی Simatic ظاهر شده در سمت چپ کلیک کنید. در قسمت راست Hardware ظاهر می شود.

برروی آن دابل کلیک کنید تا نرم افزار HW Config باز شود. از این نرم افزار برای پیکربندی سخت افزار PLC استفاده می شود.

در پنجره ی قسمت سمت راست این نرم افزار، بخش کاتالوگ رو مشاهده خواهید کرد که شامل محصولات شرکت زیمنس که قادر به برقراری ارتباط هستند، می شود.

در صورت تمایل میتوان این کاتالوگ را آپدیت کرد.

برای نمایش/عدم نمایش این پنجره از آدرس View/Catalog یا کلید میانبر Ctrl + k استفاده کنید.

پنجره سمت چپ نرم افزار دارای دو بخش مشابه است. که قسمت پایینی آن دارای توضیحات بیشتر خواهد بود.

ابتدا باید ریل (Rack) مشخص شود. از قسمت Catalog بخش Simatic 300 و از پوشه RACK-300 باید Rail را انتخاب نمایید.

sbargh.ir

همچنین میتوانید با کلیک راست روی پنجره سمت چپ و انتخاب Insert Object میتوان ریل را انتخاب کرد.

فقط یک نوع ریل داریم که هم به عنوان Rack اصلی و هم به عنوان به عنوان Rack اضافی استفاده می شود.

در S7 300 ریل دارای ۱۱ جایگاه (اسلات) است و فقط نقش نگهدارندگی ماژول ها را برعهده دارد.

اسلات شماره ۱ فقط برای تعریف منبع تغذیه (PS) کاربرد دارد. برای اضافه کردن منبع تغذیه ی ساخت شرکت زیمنس، می توانید از پنجره کاتالوگ، بخش Simatic 300 و پوشه PS 300 منبع تغذیه را تعیین کنید.

در پنجره کاتالوگ با یک بار کلیک برروی ابزار، جایگاه (اسلات) آن در جدول نشان داده می شود. بدیهی است دابل کلیک، ابزار را به پنجره مربوط به خود اضافه خواهد کرد.

در روش دیگر افزودن، باید در جدول کلیک راست کرده و New Object را انتخاب کنید. منابع تغذیه در Step 7 300 به ۲ ، ۵ و ۱۰ آمپری تقسیم می شوند. اینکه از کدام منبع تغذیه استفاده کنیم به تعداد ماژول ها و مصرف هرکدام بستگی دارد. مانند: ورودی، خروجی، CPU، FM و ... .

می توانید از منبع تغذیه ساخت زیمنس استفاده نکنید. در صورت استفاده از یک منبع تغذیه عادی می توانید جایگاه (اسلات) Power Supply را جدول خالی بگذارید.

در اسلات دوم فقط می توان CPU را تعریف کرد. انتخاب نوع CPU به نوع پروژه بستگی دارد. برخی حروف و علامت های CPU قبلا توضیح داده شد که در اینجا نیز به آنها اشاره ای داریم. علامت F نشان دهنده ی سطح حفاظتی، C نشان دهنده ی کامپکت بودن آن، H به معنی این است که CPU دارای قسمت یدکی است و ۲□□ نشان دهنده ی وجود کانکتور شبکه پروفی باس می باشد. به عنوان مثال: CPU 313C-2DP .

مابقی اسلات ها بعدا توضیح داده می شود.

در آخر علامت از ردیف بالا علامت پنجم از سمت چپ با نام Save and Compile (کلید میانبر Ctrl + S) را انتخاب کرده تا تنظیمات ذخیره شود. همچنین می توانید برای چک کردن عدم وجود خطا مسیر Station/Consistency Check را انتخاب کنید که در صورت نبود خطا، با پیغام No error مواجه خواهید شد.

حال نرم افزار را بسته و نرم افزار اصلی برگردید. بر روی نام CPU انتخاب شده کلیک کنید. حال S7 Program و بعد روی Blocks دابل کلیک کنید. توجه کنید که در این قسمت باید پوشه System Data وجود داشته باشد وگرنه Compile ما دچار مشکل بوده و اشتباه انجام شده است.

اکنون به OB1 می رسیم که اولین بلوک از بلوک های سازمانی است.این بلوک رابط کاربر و سیستم عامل بوده و کاربر باید برنامه خود را در این بلوک بنویسد و سیستم عامل نیز برای اجرای برنامه به این بلوک مراجعه می کند.

بر روی OB1 دابل کلیک کنید تا Properties آن باز شود. در این پنجره در قسمت Symbol Name می توانید تا ۶۴ کاراکتر وارد کنید. از قسمت Language زبان را انتخاب کنید (فعلا FBD) و Ok را بزنید. حال برنامه LAD/STL/FBD باز می شود که از آن برای برنامه نویسی استفاده می کنیم.

قسمت سمت چپ Program Element است که ابزار برنامه نویسی در آن قرار دارد. پنجره سمت راست پایین، محیط اصلی برنامه نویسی است. در این بخش در Title می توان نام و آیکون را وارد کنیم. البته این بخش اجباری نبوده و فقط برای عیب یابی بعدی به کمک کاربر خواهند آمد.

پوشه اول با نام Bit logic (دستورات منطق بیت) شامل دستورات بیتی می باشد. با دابل کلیک بر روی هرکدام، آن دستور وارد نرم افزار خواهد شد. در هر دستور، علامت ؟؟؟ اجبارا باید آدرس دهی شود. اما علامت سه نقطه ( . . . ) اجباری برای آدرس دهی ندارد.

**متغیر Bool دارای سایز یک بیت می باشد.**

با کلیک بر روی هر بلوک و فشردن F1 قسمت راهنمای (Help) آن همراه با توضیحات مربوطه باز می شود.

بلوک اول OR، بلوک دوم And و بلوک سوم XOR می باشد. از بلوک چهارم برای اضافه کردن ورودی استفاده می شود

بلوک پنجم Not نام دارد که برای خواندن مقدار صفر کاربرد دارد. ابزار بعدی تساوی است که برای خروجی استفاده می شود.

از ابزار Middle Out Put (...#...) برای ذخیره کردن نتیجه قسمتی از برنامه که خروجی نهایی نیست استفاده می شود. از این مقدار ذخیره شده میتوان در قسمتهای دیگر برنامه استفاده کرد.

دو نماد بعدی Set و Reset می باشد. در ابزار Set، ورودی اگر برای یک لحظه هم یک شود، خروجی یک می ماند. در ابزار Reset نیز هنگامی که ورودی برای یک لحظه یک شود، خروجی ریست خواهد شد.

> در برنامه نویسی اولویت اجرا با دستوری است که به خط انتهایی برنامه نزدیکتر است.
>
> اگر یک خروجی چند شرط دارد، بهتر است آن را در یک Network بنویسید.

ابزار های SR و RS تشکیل شده از ابزار های Set و Reset است. تنها تفاوت این دو هنگامی است هردو ورودی Set و Reset همزمان فعال باشند. در فلیپ فلاپ SR اولویت با Reset و در RS اولویت با Set خواهد بود.

نرم افزار Simatic Manager دارای نرم افزار شبیه ساز PLC SIM است که از آن میتوان برای اجرای مجازی برنامه نوشته شده، شبیه سازی آنها و مشاهده نتایج استفاده کرد. برای باز کردن آن از داخل نرم افزار Simatic Manager گزینه Simulation On/Off (گزینه پنجم از سمت راست در ردیف بالا) را انتخاب نمایید. نرم افزار PLC SIM دارای گزینه هایی برای افزودن ورودی و خروجی است. در قسمت View با انتخاب گزینه Always on Top این پنجره در جلوی همه ی پنجره های باز دیگر، نمایش داده خواهد شد.

پس از اتمام نوشتن برنامه، از نرم افزار LAD/FBD/STL آیکون Download را انتخاب کنید تا برنامه نوشته شده در نرم افزار Simulation دانلود شود. با استفاده از آیکون Monitor (علامت عینک) میتوانید عملکرد برنامه را بصورت تصویری مشاهده کنید.

در هر Network فقط یک سری دستور منتهی به خروجی نوشته می شوند.

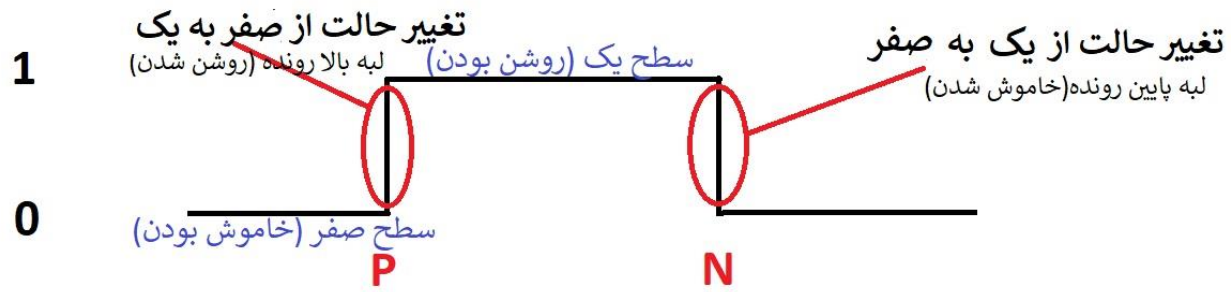
بنابراین به ازای هر خروجی حداقل یک Network لازم دارید.

دقت کنید که در هنگام دانلود برنامه، نرم افزار شبیه ساز (یا حتی خود CPU) در حالت STOP باشد. همچنین در صورتی که Monitor فعال باشد، برنامه قابل ویرایش نیست.

- روند انجام پروژه:

۱. تعیین ورودی ها
۲. تعیین خروجی ها
۳. اختصاص آدرس به ورودی و خروجی ها
۴. تقسیم مسئله به زیر مسائل کوچکتر
۵. انجام زیرپروژه ها و اتصال آنها به یکدیگر
۶. عیب یابی (تداخل آدرس و... .)

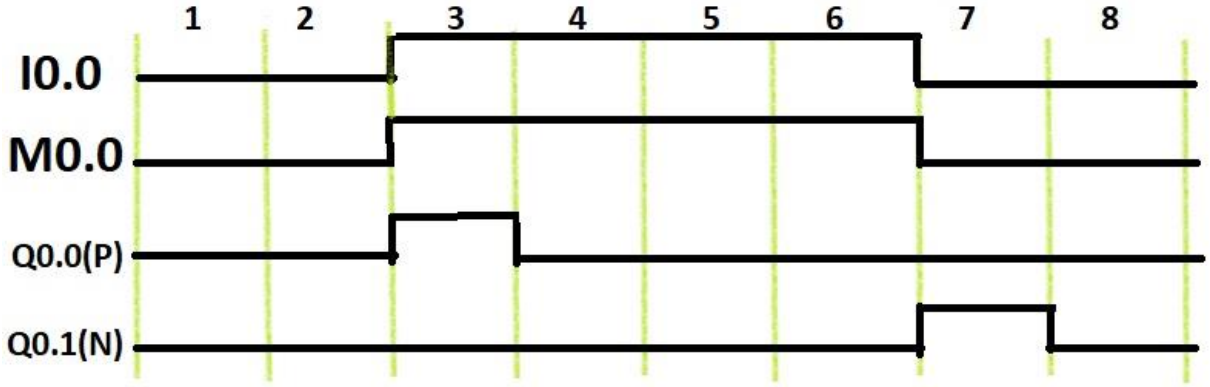
## قسمت های مختلف سیگنال دیجیتال




ابزار های P و N را داریم.

P: برای تشخیص لبه ی بالا رونده

N: برای تشخیص لبه ی پایین رونده




## ادامه تنظیمات در نرم افزار HW Config

اسلات شماره ۳ فقط مخصوص IM است. IM رابط بین رک اصلی و رک اضافی است. در اسلات های بعدی ترتیب مهم نیست و در هر کدام میتواند SM، CP، FM، DM و... جای گیرد.

کارت های موجود در پوشه SM شامل موارد ذیل می باشد:

تنوع SM در S7-400 کمتر از S7-300 می باشد.

- ورودی دیجیتال (DI)
- خروجی دیجیتال (DO)
- ورودی و خروجی دیجیتال (DO,DI) مخصوص S7-400 می باشد.
- ورودی آنالوگ (AI)
- خروجی آنالوگ (AO)
- ورودی و خروجی آنالوگ (AI,AO) مخصوص S7-400 می باشد.

CP: ماژول شبکه بوده و برای شبکه کردن CPU هایی که کانکتور شبکه ندارند، کاربرد دارد.

FM: قطعه ای است که یک وظیفه ی خاص را برعهده دارد و به اصطلاح باری را از دوش CPU برمی دارد. این ماژول با CPU تبادل اطلاعات داشته و از آن فرمان میگیرد.

DM: به ماژول کاذب معروف است. برای رزرو کردن یک اسلات ازین ماژول استفاده می کنیم. درواقع وظیفه این ماژول ارتباط ماژول قبل با ماژول بعد از خودش است.

## تایمرها

از تایمر برای زمانگیری استفاده می شود. دارای چند پایه بوده که از پایه S به عنوان پایه راه انداز تایمر استفاده می شود. در پایه TV مقدار زمان موردنظر کاربر به تایمر داده می شود. برای زمان دادن به تایمر از فرمت ذیل استفاده کنید.

S5T# ساعت H دقیقه M ثانیه S میلی ثانیه MS

فعال شدن پایه R نیز زمان گیری را متوقف می کند و تایمر را خاموش می کند. از پایه ی Q نیز به عنوان خروجی استفاده می کنیم.

حداکثر زمانی که هر تایمر میتواند دریافت کند، ۹۹۹۰ ثانیه یا ۰۰:"۳۰:'۴۵:۲ می باشد.

دو پایه دیگر نیز زمان باقیمانده تایمر را به فرمت باینری و BCD نشان می دهند.

انواع تایمرها و ویژگی های آن ها به شرح زیر می باشد:

- تایمر S-Pulse

۱. آن را به نام SP می شناسیم.
۲. برای راه اندازی به لبه بالارونده ی S نیاز دارد.
۳. برای کارکردن به سطح یک S نیاز دارد.
۴. اگر Set را برداریم و دوباره بگذاریم، تایمر از اول شروع به شمارش می کند.
۵. اگر تایمر را Reset کردیم، باید Set را برداشته و دوباره بگذاریم.
۶. زمان گیری منطبق بر خروجی است.

- تایمر S-PEXT

۱. آن را با نام SE می شناسیم.
۲. برای راه اندازی نیاز به لبه ی بالا رونده ی S دارد.
۳. برای کارکردن به سطح یک S نیاز ندارد.
۴. زمان گیری منطبق بر خروجی است.
۵. قبل از تمام شدن زمان، اگر Set را برداریم و دوباره بگذاریم، تایمر از اول شروع به شمارش می کند.

- تایمر S-ODT

۱. برای راه اندازی به لبه ی بالا رونده ی S نیاز دارد.
۲. برای کار کردن به یک بودن Set نیاز دارد.
۳. بعد از اینکه زمان گیری تمام شد، اگر پایه Set یک باشد، آنگاه خروجی داریم.
۴. قبل از تمام شدن زمان اگر Set را برداریم دیگر خروجی نخواهیم داشت.
۵. تایمر تاخیر در وصل
۶. اگر Reset کردیم، باید Set را برداریم و دوباره بگذاریم.

- تایمر S-ODTS

۱. تمام ویژگی های تایمر S-ODT را دارد.
۲. تنها تفاوت آن این است که کارکرد آن به وجود Set بستگی ندارد.
۳. پس از زدن Set تنها راه خاموش کردن آن فقط پایه Reset خواهد بود.

- تایمر S-Off DT

۱. با زدن Set خروجی داریم ولی زمان گیری نداریم.
۲. با برداشتن Set زمان گیری شروع می شود.
۳. پس از پایان زمان گیری خروجی قطع می شود.
۴. شروع زمان گیری با لبه ی پایین رونده Set خواهد بود.
۵. خروجی فقط به وجود Set بستگی دارد و نه لبه ی بالا رونده ی آن.

# مروری بر زبان های دیگر

برای جابجایی بین زبان ها می توانید از زبانه ی View آن ها را انتخاب کنید. همچنین می توانید از کلید میانبر Ctrl و هرکدام از کلیدهای ۱، ۲ ویا ۳ استفاده کنید.

نکته: در زبان LADDER عملگرهای And، OR و XOR به طور مستقیم وجود ندارند.

در زبان STL برای نوشتن دستورات از سمت چپ بلوک (قسمت ورودی ها) شروع می کنیم.

A: برای خواندن و And کردن یک سیگنال با RLO استفاده میشود.

```
A   I0.0
A   I0.1
=   Q0.0
```

Status Word: آن را با نام کلمه وضعیت می شناسیم. سیستم عامل PLC برای اطلاع از وضعیت اجرای برنامه در هر خط، به این کلمه مراجعه می کند. تاکنون ۹ بیت از SW مورد استفاده قرار گرفته است.

**SW**

FC: یا First Check که اگر اجرای برنامه در خط اول Network یا پرانتز باشد، مقدار این بیت برابر صفر و در غیر این صورت مقدار آن یک خواهد بود.

RLO: یا Result of Logic Operation که نتیجه عملیات منطقی هر خط در این بیت ذخیره می شود.

STA: وضعیت سیگنال خوانده شده در هر خط در این بیت ذخیره می شود.

مثال:

| STA | RLO | FC |
|---|---|---|
| 0 | 0 | |
| 0 | | |
| 1 | 0 | |
| 0 | | |

اگر I0.0 مقدار صفر و I0.1 یک داشته باشد. داریم:

در خط اول FC صفر است و برنامه مقدار I0.0 را خوانده و در RLO قرار می دهد. در خط بعدی FC یک شده و برنامه مقدار I0.1 را خوانده و با مقدار قبلی And کرده و در RLO جایگزاری می شود. در نهایت با رسیدن به دستور انتساب (=) RLO مقدار خود را در آدرس مقابل این دستور قرار می دهد.

برای دستور OR از O و برای دستور XOR از X استفاده می کنیم.

دستورات O، A و X با RLO کار می کنند اما دستورات S و R به RLO وابسته اند یعنی اگر RLO=1 باشد این دستورات اجرا می شوند و اگر RLO=0 باشد این دستورات اجرا نمی شوند (یعنی از روی این دستورات پرش می شود).

برای خواندن صفر (Not کردن) در حالت STL از دستورات AN، ON و XN استفاده می کنیم.

مثال:

فلیپ فلاپ SR

```
A  I0.0
S  M0.0
A  I0.1
R  M0.0
A  M0.0
= Q0.0
```

نکته: اگر بخشی از برنامه حذف شده باشد در هنگام تبدیل به STL دستور NOP0 ظاهر می شود. (به سیستم عامل نرم افزار کمک می کند که آن را از STL به FBD تبدیل کند.)

- از دستورات FP و FN برای خواندن لبه ی بالارونده و لبه ی پایین رونده استفاده می شود.

مثال:

لبه بالا رونده

```
A  I0.0
FP  M0.0
=  Q0.0
```

## انواع متغیرها

Bool = با سایز بیت

S5T# = با سایز Word

Int = با سایز Word / عدد صحیح علامت دار / دارای ۱۵ بیت ارزش و یک بیت علامت ( ۱ بودن برای منفی و حالت صفر برای مثبت بودن است.) / دارای بازه ی قابل پذیرش ۳۲۷۶۸- تا ۳۲۷۶۷+ می باشد.

Dint = مشابه متغیر Int اما باسایز Dword و اعداد صحیح علامت دار بزرگتر از آن

Real = با سایز Dword / دارای یک بیت علامت / در برگیرنده ی اعداد حقیقی / یک عدد بین صفر تا ده به همراه توانی از ۱۰ را می پذیرد.

Word = با سایز Word / اعداد حسابی از صفر تا ۶۵۵۳۵ / بیت علامت ندارد

Dword = با سایز Dword / بیت علامت نداشته و فقط اعداد حسابی را می پذیرد.

## مقایسه کننده ها(Comparators)

سه گروه مقایسه کننده داریم: Int، Dint، Real

| انواع مقایسه کننده ها | | | |
|---|---|---|---|
| مقایسه گر | علامت | مثال برای Int | |
| مساوی | == | ==I | EQ |
| نامساوی | <> | <>I | NE |
| بزرگتر | > | >I | GT |
| کوچکتر | < | <I | LT |
| بزرگتر مساوی | >= | >=I | GE |
| کوچکتر مساوی | <= | <=I | LE |

برای اینکه در یک بلوک Dint یک ورودی Int به کار ببریم، از #L در اول آن استفاده می کنیم.

زمانیکه میخواهیم یک عدد صحیح را در بلوک Real وارد کنیم، باید آن را با قسمت اعشاری صفر وارد کنیم.

## دونوع کارت مورد استفاده

رک اضافی IM 365-SR

۱. هم در رک اصلی و هم در رک اضافی از همین کارت استفاده میکنیم.
۲. رک اضافی نیازی به منبع تغذیه ندارد.
۳. حداکثر فاصله ی رک اضافی با رک اصلی یک متر است.
۴. فقط یک رک اضافی می توانیم داشته باشیم.

کارت IM 360-S، IM 361-R

۱. در رک اصلی از IM360-S و در رک اضافی IM 361-R استفاده می شود.
۲. رک اضافی نیاز به منبع تغذیه ندارد.
۳. حداکثر فاصله رک اضافی می تواند ۱۰ متر باشد.(۲،۵ و ۱۰ متر)
۴. حداکثر ۳ رک اضافی می توانیم داشته باشیم.

# لامپ های سیگنال موجود بر روی CPU

**SF:** اگر در حین اجرای برنامه، خطای سخت افزاری یا نرم افزاری روی دهد، CPU به حالت Stop رفته و این چراغ روشن می شود. (قرمز)

**BF:** اگر در شبکه کردن CPU مشکلی ایجاد شود، این چراغ روشن می شود. (قرمز)

**DC5V**: اگر تغذیه مدارات داخلی CPU به درستی تامین شده باشد، این چراغ روشن می شود. (سبز)

**FRCE:** اگر در برنامه حداقل یک متغیر force شده وجود داشته باشد، این چراغ روشن می شود. (نارنجی)

**Run:** اگر برنامه در حال اجرا باشد این چراغ روشن می شود.

**Stop:** اگر CPU در حال Stop باشد این چراغ روشن می شود.

# آکومولاتور

برای انتقال اطلاعاتی با سایز بزرگتر از بیت از آن استفاده می کنیم.

در S7 300 دو عدد آکومولاتور ۳۲ بیتی داریم.

در S7 400 چهار عدد آکومولاتور ۳۲ بیتی داریم.

## دستورات بارگذاری و انتقال از ACCU

L: این دستور محتویات آدرس مقابل خود را به ACCU 1 منتقل می کند.

T: این دستور محتویات ACCU 1 را به آدرس مقابل خود منتقل می کند.

کاربر فقط به ACCU 1 دسترسی دارد.

**چند مثال:**

**L IB 0**: این دستور، محتویات IB0 را به ACCU 1-L-L منتقل میکند.

**L MW0**: این دستور محتویات MW0 را به ACCU 1-L منتقل میکند.

**L ID4**: این دستور محتویات ID4 را به ACCU1 منتقل میکند.

**1** **L IW0**

**2** **L IW2**

با دستور ۱ محتویات IW0 به ACCU1-L منتقل می شود. با اجرای دستور ۲ ابتدا محتویات ACCU1 در ACCU2 کپی می شود. سپس محتویات IW2 در ACCU1 کپی می شود.

اگر ۳ دستور بارگذاری پشت سرهم داشته باشیم، اطلاعات بارگذاری اول از بین می رود.

sbargh.ir

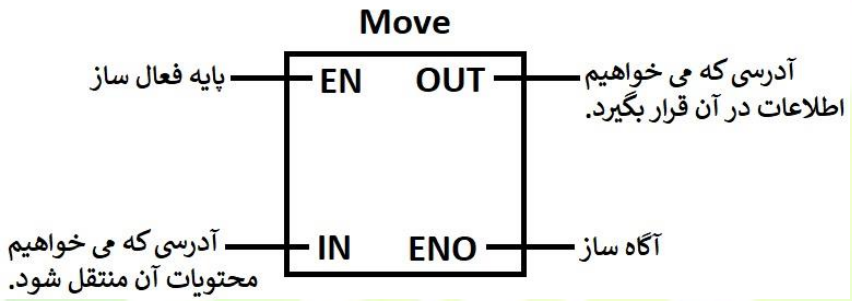
**TQB2:** محتویات ACCU1-L-L را به آدرس QB2 منتقل می کند.

**TMW2**: محتویات ACCU1-L را به آدرس MW0 منتقل می کند.

**TQD6**: محتویات ACCU1 را به آدرس QD6 منتقل می کند.

## دستور Move

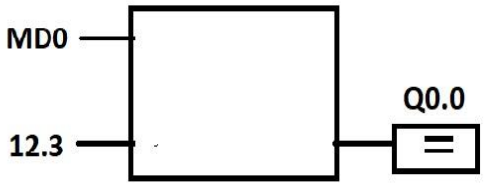
اگر پایه EN آدرس دهی نشود، سیستم عامل بدون هیچ پیش شرطی آن را انجام می دهد. اما اگر آدرس دهی شود، وقتی سیستم عامل به آن رسید، باید حتما یک باشد تا اجرا شود. اگر عملیات بلوک به درستی انجام شود، پایه آگاه ساز یک خواهد شد.




### مقایسه گرها

مثال:

```
L MW0
L 5
== I
= Q0.0
```




```
LMD0
L12.3
>R
==Q0.0
```

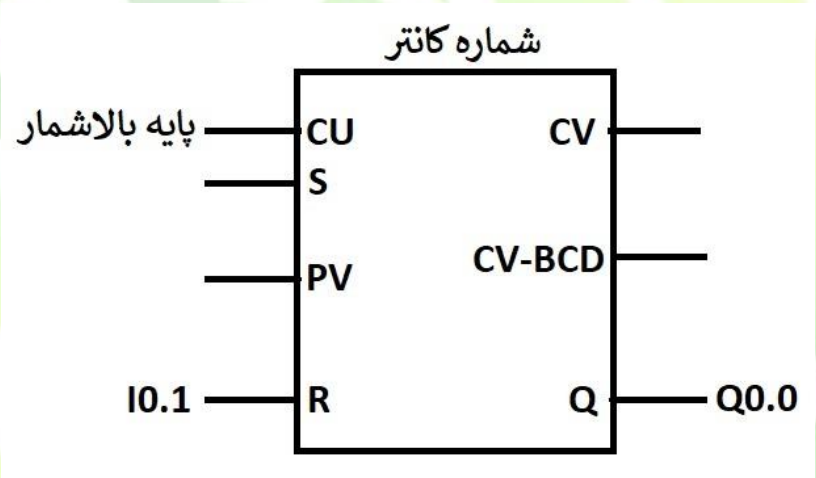
# BCD

دسیمال کد شده بصورت باینری : هر رقم را ۴ رقم به مبنای ۲ تبدیل می کنیم. برای تبدیل BCD به مبنای ۱۰ ، کافی است از راست ۴ رقم ۴ رقم جدا کنیم.

به عنوان مثال عدد ۴۹۳ در BCD بصورت ۱۰۰۱۰۰۱۰۰۱۱ می باشد اما همین عدد در مبنای دو، برابر ۱۱۱۱۰۱۱۰۱ خواهد بود.

## کانترها (Counters)

از کانترها برای شمارش استفاده می کنیم.




در کانتر بالا شمار، با هر لبه ی بالارونده ی پایه ی بالا شمار، مقدار جاری کانتر یک واحد افزایش می یابد. با فعال شدن کانتر و افزایش مقدار جاری آن به بیشتر از صفر، خروجی Q روشن می شود. از پایه R برای ریست کانتر به صفر استفاده می کنیم بنابراین با فعال شدن این پایه مقدار جاری کانتر صفر شده و خروجی خاموش می شود. از پایه CV مقدار جاری کانتر را به فرمت هگزادسیمال نشان می دهد. کاربر از این پایه بیشتر استفاده میکند. پایه CV-BCD نیز مقدار جاری کانتر را به فرمت BCD نمایش می دهد. آدرس دهی پایه S اجباری نیست، اما اگر آدرس دهی شود پایه PV نیز باید مقدار بگیرد. همچنین اگر پایه PV مقدار بگیرد، پایه Set نیز باید

sbargh.ir

آدرس دهی شود. با فعال شدن پایه S، مقدار اولیه PV در کانتر بارگذاری می شود. برای مقدار دهی پایه PV از C# استفاده میکنیم.

- کانترها حداکثر به میزان ۹۹۹ واحد می شمارند.
- کانترها منفی شمار نیستند.

کانتر پایین شمار نیز مشابه کانتر بالا شمار بوده اما با هر بار لبه ی بالارونده این پایه، یک واحد از مقدار جاری کانتر کم خواهد شد.

- یک کانتر ترکیبی نیز در دسترس داریم که هم بالاشمار و هم پایین شمار خواهد بود.

## ساعت های کاری PLC

Stop: در این حالت برنامه قابل ویرایش است.

Run: برنامه فقط قابل خواندن بوده و امکان ویرایش وجود ندارد.

Run-p: برنامه در حالت اجرا بوده و همزمان قابلیت ویرایش و دانلود را نیز دارد.

MRes: برای ریست کردن حافظه ی داخلی CPU استفاده می شود.

## پالس ساعت

یک روش برای ایجاد پالس در فواصل زمانی منظم است. برای ایجاد آن ابتدا وارد نرم افزار HW Config شوید. روی CPU در اسلات دوم (قبلا نحوه تنظیمات سخت افزاری توضیح داده شد) کلیک راست کرده و وارد پنجره سوم شوید. از قسمت پایین، تیک را زده و در قسمت byte یک عدد به عنوان مثال ۱۰۰ را وارد کرده و سپس دانلود را بزنید.(در صورتی که از شبیه ساز استفاده می کنید، پس از دانلود، در پنجره بازشده کلید View را زده و پس از انتخاب CPU کلید Ok را بزنید.)

sbargh.ir

حال برای استفاده از آن، وارد نرم افزار برنامه نویسی شده و در یک ورودی دلخواه فرمت زیر را وارد کنید: M 100. (0 or 1 or 2 ... 7)

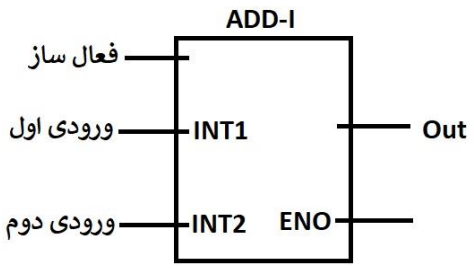
به عنوان مثال: M100.5

هریک از اعداد وارد شده، فرکانس خاص مربوط به خود را ایجاد می کنند.

| عدد | ۰ | ۱ | ۲ | ۳ | ۴ | ۵ | ۶ | ۷ |
|---|---|---|---|---|---|---|---|---|
| T | ۰.۱ | ۰.۲ | ۰.۴ | ۰.۵ | ۰.۸ | ۱ | ۱.۶ | ۲ |
| F | ۱۰ | ۵ | ۲.۵ | ۲ | ۱.۲۵ | ۱ | ۰.۶۲۵ | ۰.۵ |

## دستورات ریاضی اعداد صحیح علامت دار

INT1 یا INT2: بسته به نوع بلوک از نوع INT یا Dint با اندازه ملاک Word یا Dword می باشد.




```
L  MW0
L  +5
+I
TMW20
```

| +D | +I | ADD | دو عدد را با هم جمع می کند. |
|---|---|---|---|
| -D | -I | SUB | عدد دومی را از اولی کم می کند. |
| *D | *I | MOL | دو عدد را در هم ضرب می کند. |
| /D | /I | DIV | خارج قسمت تقسیم عدد اول بر دوم |
| MOD | ------ | | باقیمانده تقسیم عدد اول بر دوم |

## دستورات ریاضی متغیرهای Real (Floating Point Function)

sbargh.ir

ورودی آنها از نوع Real و سایز Dword است.

| معادل STL | دستور | وظیفه |
|---|---|---|
| +R | ADD-R | جمع |
| -R | SUB-R | تفریق |
| /R | DIV-R | تقسیم |
| *R | MUL-R | ضرب |

ABS: قدر مطلق عدد ورودی را در خروجی قرار می دهد.

Sqrt: جذر متغیر ورودی را در خروجی قرار می دهد.

Sqr: توان دوم متغیر ورودی را در خروجی قرار می دهد.

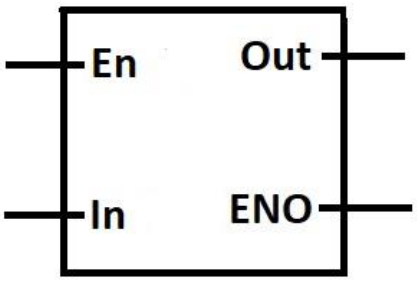
Ln: لگاریتم طبیعی متغیر ورودی را در خروجی قرار می دهد.

Exp: تابع نمایی

SIN, COS, TAN: توابع مثلثاتی (متغیر ورودی برحسب رادیان)

ASIN, ACOS, ATAN: توابع معکوس مثلثاتی (متغیر خروجی برحسب رادیان خواهد بود.)

## مبدل ها (Convertor)




BCD-I: برای تبدیل فرمت BCD به INT استفاده می شود.
( حداکثر ۹۹۹) [BTI]

I-BCD: برای تبدیل INT به BCD استفاده می شود.
(حداکثر ۹۹۹) [ITB]

I-DI: برای تبدیل INT به DINT استفاده می شود. [ITD]

BCD-DI: برای تبدیل BCD به DI استفاده می شود. (حداکثر ۹٬۹۹۹٬۹۹۹) [BTD]

sbargh.ir

DI-BCD: برای تبدیل DI به BCD استفاده می شود. (۹٬۹۹۹٬۹۹۹) [DTB]

DI-R: برای تبدیل DI به Real استفاده می شود. [DTR]

Inv-R: متمم یک متغیر ورودی را در خروجی قرار می دهد. (حالت Int) [InvI]

Inv-R: متمم یک متغیر ورودی را در خروجی قرار می دهد. (حالت DINT) [InvD]

- متمم یک: برای محاسبه متمم یک کافی است تمام بیت های یک متغیر را معکوس کنیم.

NEG-I: قرینه ی متغیر ورودی را در خروجی قرار می دهد. (متمم ۲) (حالت Int)

NEG-DI: قرینه ی متغیر ورودی را در خروجی قرار می دهد. (متمم ۲) (حالت Dint)

NEG-R: قرینه ی متغیر ورودی را در خروجی قرار می دهد. (متمم ۲) (حالت Real)

- متمم ۲: با افزودن ۱ به متمم یک، متمم ۲ به دست می آید.

در ادامه برای تبدیل Real به Dint ، ۴ بلوک در نظر گرفته شده است.

- Round: جهت گرد کردن عدد بکار می رود.

۱۲.۵ = ۱۲ -۱۳.۴ = -۱۳ ۱۳.۵ = ۱۴ ۷.۸ = ۸ ۲۲.۴ = ۲۲

- Trunc: جهت حذف قسمت اعشار بکار می رود.

۶.۳ = 6 ۴.۵ = ۴ ۱۶.۹۹ = ۱۶ ۱۲.۲ = ۱۲

- CEIL: جهت گرد کردن به بالا به کار می رود.

۷.۶ = ۸ -۱۴.۲۵ = -۱۴ ۱۲.۳ = ۱۲

- Floor: جهت گرد کردن به پایین به کار می رود. (کاربرد آن شبیه جزء صحیح است.)

sbargh.ir

در دستورات پوشه Word Logic ورودی ها باهم بیت به بیت با هم ترکیب منطقی می شوند.

مثال: در دستور WAND، ورودی ها بیت به بیت باهم And شده و نتیجه به عنوان خروجی اعلام می شود. مثلا اگر ورودی ها به شرح ذیل باشند:

۱۰’۱۰۰’۱۱۱’۰۱۰’۰۰۱

۱۱’۱۰۱’۱۰۰’۰۱۱’۱۱۱

خروجی برابر ۱۰’۱۰۰’۱۰۰’۰۱۰’۰۰۱ خواهد بود.

۱۶.۹۵ = ۱۶ ۱۱.۵۳ = ۱۱ -۱۴.۲ = -۱۵ ۱۲.۷ = ۱۳

## حالت های راه اندازی PLC

Cold: تمامی متغیرهای برنامه، چه آنها که بصورت پایدار تعریف شده اند و چه آنها که به صورت ناپایدار تعریف شده اند، پاک می شوند. همچنین برنامه از خط اول OB1 اجرا می شود.

Worm: متغیرهایی که بصورت پایدار تعریف شده اند پاک نمی شوند و فقط متغیرهای ناپایدار پاک خواهند شد. برنامه از خط اول OB1 اجرا خواهد شد.

Hot: هیچکدام از متغیرها چه پایدار و چه ناپایدار پاک نمی شوند. با اجرای مجدد آن، برنامه از همان خطی که متوقف شده بود ادامه می یابد. این حالت فقط در S7-400 در دسترس است.

## بلوک های سازمانی OB (Organization Block)

**وظایف:**

sbargh.ir

۱. رابط کاربر و سیستم عامل

- OB1: کاربر برنامه خود را در OB1 می نویسد وسیستم عامل هم برای اجرای برنامه به OB1 مراجعه می کند.

۲. مدیریت وقفه:

۲,۱. مبتنی بر رویداد (Event) هستند

۲,۱,۱. وقفه های TOD: OB1X

۲,۱,۲. وقفه های تاخیر زمانی: OB2X

۲,۱,۳. وقفه های سیکلی: OB3X

۲,۱,۴. وقفه های سخت افزاری: OB4X

۲,۲. مبتنی بر خطا (Error) هستند.

۲,۲,۱. وقفه های خطای سخت افزاری: OB8X

۲,۲,۲. وقفه های نرم افزاری : OB12X

که هر دو دسته را OB ها مدیریت می کنند.

۳. دستورات راه اندازی:

- Cold: OB102
- Warm: OB100
- Hot: OB101

معمولا هرچه عدد OB بزرگتر باشد، اولویت اجرای بالاتری دارد. برای استفاده از وقفه های زمانی، کارت ورودی نیز باید وقفه پذیر باشد.

در نرم افزار Simatic Manager می توان بلوک های سازمانی را تعریف کرد. پس از تعریف سخت افزار، در داخل پوشه Blocks، روی صفحه کلیک راست کرده واز قسمت Insert New Object گزینه ی Organization block را انتخاب کنید.

- بعضی از بلوک ها بلافاصله پس از ایجاد شدن کار می کنند ام برخی دیگر خیر.

sbargh.ir

# Data Block

این بلاک ها حاوی دستورات برنامه نویسی نیستند و فقط برای ذخیره ی اطلاعات، متغیرها و پیغام ها استفاده می شوند.

دو نوع دارند:

- نوع عمومی(Share): توسط همه بلوک ها قابل دسترسی است.
- نوع اختصاصی(Instance): فقط توسط FB مخصوص به خود قابل دسترسی و اعمال تغییرات است.

Data Block ها از طریق محیط Blocks قابل ایجاد هستند. برای ایجاد آن ابتدا کلیک راست کرده و از Insert New Object گزینه ی Data Block را انتخاب کنید. گزینه ای با نام DB1 ایجاد خواهد شد که با دابل کلیک بر روی آن میتوان وارد آن شده و آن را ویرایش کرد.

تمرین: یک Data Block ایجاد کرده و اطلاعات زیر را در آن وارد کنید.

## آدرس دهی متغیرهای دیتا بلاک

- روش اول

آدرس دهی بیتی:

آدرس بیت.DBX.شماره دیتا بلاک

بیت شماره صفر از بایت شماره صفر از دیتا بلاک ۱ DB1.DBX.0.0

آدرس دهی بایتی:

آدرس بایت.DBB.شماره دیتا بلاک

DB1.DBB1

آدرس دهی وردی:

sbargh.ir

آدرس.DBW.شماره دیتا بلاک

DB1.DBW2

آدرس دهی دابل وردی:

آدرس.DBD.شماره دیتا بلاک

DB1.DBD4

- روش دوم

در پوشه DB Call، بلوک Open را انتخاب کنید.

در قسمت مربوط به آدرس دهی، شماره دیتا بلاک را انتخاب کنید.

آدرس دهی بیتی: آدرس بیت .DBX

آدرس دهی بایتی: آدرس بایت .DBB

آدرس دهی Word: آدرس . DBW

آدرس دهی Dword: آدرس .DBD

## فانکشن Function

برای برنامه نویسی ساختار یافته و نوشتن دستورات تکراری از آن ها استفاده می کنیم. برای ایجاد یک FC باید به پوشه Blocks مراجعه کنید. Fc ها مانند OB قابلیت برنامه نویسی دارند (به زبان دلخواه).

در بخش بالای نرم افزار STL/LAD/FBD متغیر های ورودی و خروجی را تعیین کرده و آن را در بخش پایین برنامه نویسی به کار می گیریم.

برای استفاده از ورودی و خروجی که تعیین کرده ایم، پس از نوشتن برنامه در مکان آدرس دهی دلخواه کلیک کرده و قسمتی از نامی که به آن متغیر داده ایم تایپ کرده و خود نرم افزار ان را به ما پیشنهاد می دهد و با کلیک بر روی آن، می توان آن را انتخاب کرد.

متغیری که با # شروع می شود متغیر محلی است.

## Function Block (FB)

از لحاظ عملکرد دقیقا مشابه FC ها بوده اما با این تفاوت که FB و FC در این است که FB ها دارای حافظه هستند و متغیرهای آن ها در DB مخصوص به FB ذخیره می شوند.

- هر FB با DB مخصوص خود فراخوانی می شوند.

در قسمت آدرس دهی آن، باید آدرس DB مربوطه را وارد کنیم که اگر قبلا آن را ایجاد نکرده باشیم، پس از وارد کردن آدرس و فشردن Enter از ما برای ایجاد آن سوال می شود.

روش دیگر این است که پس از تعریف FB در محیط Simatic Manager کلیک راست کرده و از Insert New Object گزینه Data Block را انتخاب می کنیم. سپس نوع آن را Instance قرار داده و FB دلخواه را انتخاب کنید.