مجموعه ی آموزشی

PLC FATEK

CD همراه

فصل اول

مدار فرمان :

بطور کلی مدار فرمان عبارت است از مداری که فرامین کنترلی را برای مدار قدرت صادر می کند . این مدارات با توجه به ورودی مطلوب از طرف اپراتور ، خروجی را برای مدار قدرت ارسال می کنند . مدارات فرمان بدلیل پیچیدگی زیادی که دارند باید یکبار و بطور صحیح پیاده سازی شوند زیرا در صورت نقص ، عیب یابی آن مشکل بوده و معمولا باید برای رفع عیب ، مدار فرمان دوباره طراحی و پیاده سازی شود . در طراحی مدارات فرمان قطعات مختلفی سهیم هستند که مدار فرمان به کمک این قطعات مدار قدرت را کنترل می کند . مهمترین قطعات استفاده شده در مدارات فرمان کنتاکتورها ، رله ها و راه انداز های موتور هستند .

رله ها :

رله یک کلید الکتریکی است که با جریان کمی عمل می کند و کنتاکت های آن می تواند جریان زیادی را عبور دهند . رله در قرن نوزدهم برای استفاده در تلگراف اختراع شد و در قرن بیستم از رله در سیستم های کنترل استفاده گردید . در مقایسه با سایر ادوات الکتریکی رله ها بخاطر داشتن قسمت های فیزیکی متحرک ، سرعت و عمر محدودی دارند . رله های الکتریکی از نظر سرعت ، جریان عبوری ، اندازه ، قابلیت اطمینان و عمر مفید بر رله های الکترومکانیکی ارجحیت دارند . معمولا رله ها برای محدوده کمتر از 5 آمپر مناسب است .

کنتاکتور ها :

کنتاکتور یک کلید مغناطیسی می باشد که یکی از اجزای مهم در مدارات فرمان الکتریکی بحساب می آید . کنتاکتور با استفاده از خاصیت الکترومغناطیس ، مانند رله ها تعدادی کنتاکت را به یکدیگر وصل و یا از یکدیگر جدا می کند ، از این خاصیت جهت قطع و وصل و یا تغییر اتصال در مدارات استفاده می شود . استفاده از کنتاکتور بجای کلیدهای دستی دارای مزایایی می باشد که بشرح زیر است :

1 – کنترل و راه اندازی مصرف کننده یا سیستم از راه دور و چندین نقطه بطور همزمان

2- سرعت بالای قطع و وصل کنتاکتور و طول عمر بالای قطعات

3- امکان طراحی مدارات فرمان خودکار جهت کنترل سیستم های پیچیده

4- کنتاکتور دارای ضریب ایمنی بالا جهت بهره بردار یا اپراتور است

5- در هنگام قطع جریان برق و اتصال مجدد آن ، مصرف کننده یا سیستم خود بخود راه اندازی نمی شود و باید مجددا سیستم را استارت نمود

معمولا کنتاکتورها برای محدوده بیشتر از 15 آمپر مناسب است .

برای هر کنتاکت دو حالت متصور می باشد : NC یا در حالت عادی بسته ، NO یا در حالت عادی باز

موتورهای پله ای (Stepped Motor) :

موتورهای پله ای نمونه ای از موتورهای الکتریکی هستند که بدون استفاده از فیدبک امکان کنترل سرعت و تنظیم موقعیت حرکتی را در اختیار ما قرار می دهند . با تحریک ورودی توسط پالس موتور به اندازه چند درجه حول محور خود دوران می کند . در حقیقت یک موتور پله ای پالس الکتریکی را به حرکت مکانیکی تبدیل می کند . عملکرد اصلی یک موتور پله ای به شفت موتور اجازه می دهد تا به اندازه زاویه ای دقیق مطابق پالس های الکتریکی ارسالی به موتور بچرخد ، از آنجا که شفت موتور فقط به اندازه زاویه طراحی شده هنگام ارسال پالس الکتریکی حرکت می کند ، می توان با کنترل پالس های الکتریکی ارسالی موقعیت و سرعت را کنترل کرد . گشتاور نگهداری به موتورهای پله ای این اجازه را می دهد که موقعیت خود را هنگام توقف بطور محکم حفظ کنند . موتور پله ای عموما در موقعیت توقف بدون انرژی باقی می ماند و هنگامی که تغذیه موتور به کلی قطع شود بصورت مغناطیسی در موقعیت قبلی خود قفل می شود .

موتورهای خود فرمان (Servo Motor) :

موتورهای با قدرت بالایی هستند که برای جابجایی اوزان سنگین مورد استفاده قرار می گیرند و مستقیما توسط ولتاژ AC تغذیه می شوند که البته نوع DC آن نیز موجود می باشد .

لیمیت سوئیچ :

یک قطعه مکانیکی می باشد که از کنتاکت فیزیکی برای آشکارسازی حضور یا عدم حضور یک جسم استفاده می کند . هنگامیکه جسم هدف با محرک کنتاکت فیزیکی پیدا می کند ، محرک از موقعیت عادی خود به موقعیت کاری تغییر مکان می دهد این عمل مکانیکی کنتاکت های بدنه سوئیچ را فعال می کند و خروجی صادر می شود .

سنسور :

یک نوع مبدل سیگنال می باشد که سیگنال غیرالکتریکی ورودی را به سیگنال الکتریکی برای خروجی تبدیل می کند . سنسورها پارامترهای مختلف نظیر سرعت ، دما ، رطوبت ، جابجایی و غیره را به سیگنال الکتریکی تبدیل می کنند . در صنعت طیف وسیعی از سنسورها استفاده می شود که در ادامه به معرفی آنها خواهیم پرداخت .

ترموکوپل :

یک نوع سنسور دما است که از اتصال دو فلز غیر همجنس در یک انتخاب بدست می آید . اصول کار ترموکوپل بر مبنای اثر سیبک است (وقتی دو فلز غیرهمجنس از یک سمت بهم وصل شوند و محل پیوند حرارت داده شود ، در سمت دیگر اختلاف پتانسیل کوچکی بوجود می آید)

لودسل :

یک سنسور نیرو می باشد که نیرو یا وزن را به سیگنال الکتریکی تبدیل می کند . اساسا لودسل از یک مجموعه استرین گیج تشکیل شده است که معمولا چهار عدد هستند و بصورت مدار پل وتسون بهم اتصال دارند .

سنسور فتوالکتریک (مادون قرمز) :

این سنسور از یک پرتو نوری مدوله شده استفاده می کند که توسط هدف شکسته شده یا منعکس می گردد . سنسور فتوالکتریک قادر به تشخیص نور مدوله شده از نور محیط می باشد . منابع نور از طریق این سنسورها در محدوده سبز قابل رؤیت تا مادون قرمز نامرئی در طیف نوری بکار گرفته می شوند .

سنسور القایی :

این سنسور به کمک خاصیت الکترومغناطیس توانایی تشخیص فلز را در میدان دید خود دارد .

سنسور خازنی :

نسبت به تمام مواد (فلز و غیرفلز) حساس بوده و با حضور قطعه موردنظر در نزدیکی آن و تغییر ظرفیت خازنی ، سوئیچ می کند . این سنسور برای کنترل سطح مایعات بکار می رود .

پتانسیومتر :

یک سنسور موقعیت است . پتانسیومتر یک مقاومت متغیر است که با تغییر مکان بازوی آن مقدار مقاومت تغییر می کند . با اندازه گیری میزان مقاومت بین بازوی متحرک و یکی از سرهای ثابت مکان مشخص می گردد .

کد کننده (Encoder) :

ابزاری الکترومکانیکی است که مکان یا حرکت را تشخیص می دهد و شامل یک LED و یک سنسور نوری می باشد . با حرکت یک صفحه مشبک از جلوی LED یک سری پالس در خروجی سنسور نوری ایجاد می شود که به کمک این پالس ها فاصله محاسبه می شود .

واحدهای ورودی و خروجی آنالوگ :

بیشتر سیگنال های طبیعی تغییرات پیوسته دارند و سنسورهایی که کمیت های فیزیکی مانند فشار ، سرعت ، درجه حرارت را تشخیص می دهند خروجی آنالوگ ایجاد می کنند . برای پردازش این سیگنال ها در یک سیستم دیجیتال لازم است این اطلاعات به دیجیتال تبدیل شوند . مبدل های A/D سیگنال پیوسته را تبدیل به یک کد دیجیتال می کنند که معمولا این کد **8** بیتی می باشد . هرچه تعداد بیت خروجی A/D بیشتر باشد دقت تبدیل بیشتر است . ورودی و خروجی آنالوگ معمولا در بازه 10 – 0 ولت یا 20 – 0 میلی آمپر فعالیت می کند .

مدارات منطقی :

مداراتی که در آن متغیرها دارای دو مقدار بوده و بوسیله عملگرهای منطقی بهم مرتبط می گردند را مدار منطقی می نامیم . مدارات منطقی به دو دسته کلی تقسیم می گردند : مدارهای ترکیبی (حلقه باز) ، مدارهای ترتیبی (حلقه بسته) .

مدارهای ترکیبی (حلقه باز) :

در این مدارها خروجی لحظه فعلی به ورودی در همان لحظه بستگی دارد ، بعبارت دیگر هر ورودی اعمال شده به سیستم ، خروجی متناظر خود را تولید می نماید . در مدارات حلقه باز اطلاعاتی از خروجی به ورودی داده نمی شود و کنترل حلقه باز زمانی دچار اختلال می شود که اختلال ناخواسته ای باعث شود خروجی ها از حد مطلوب خارج شوند ، در اینصورت ممکن است سیستم کلی از کنترل خارج شود . بعنوان مثالی از این مدارات می توان به مکانیزم کاری یک ماشین لباسشویی اشاره نمود .

مدارهای ترتیبی (حلقه بسته) :

در مدارات ترتیبی حالت فعلی خروجی علاوه بر وضعیت فعلی ورودی ها به وضعیت قبلی خروجی نیز بستگی دارد یعنی خروجی مدار که در لحظه های قبل بدست آمده و در یک واحد حافظه ذخیره گردیده است ، بر وضعیت فعلی خروجی اثر می گذارد . در این نوع کنترل برای جبران اثر اختلال ، خروجی سیستم اندازه گیری می شود و در صورتیکه خروجی از مقدار مطلوب فاصله داشته باشد تدابیر کنترلی مناسب برای جبران آن اعمال می شود . در مدارات ترتیبی عناصر حافظه وجود دارند که اطلاعات خروجی را برای استفاده ورودی در خود نگهداری می کنند . یکی از عناصر حافظه در مدارات ترتیبی فلیپ فلاپ ها هستند .

حافظه ها :

یک واحد حافظه ابزاری است که اطلاعات دودوئی جهت ذخیره شدن به آن منتقل و یا اطلاعاتی که برای پردازش لازم است از آن دریافت می شود . محلی که اطلاعات ، دستورالعمل ها و نتایج حاصل از عملیات منطقی یا حسابی روی داده ها ، بصورت اطلاعات کد شده برای مدت زمان آنی یا دائم در آنجا نگهداری می شود ، حافظه نامیده می شود . دو نوع حافظه در سیستم دیجیتال وجود دارد : حافظه با دستیابی تصادفی RAM ، حافظه فقط خواندنی ROM .

اطلاعات حافظه RAM بگونه ای است که هم می توان آنها را خواند و هم می توان آنها را تغییر و یا حذف نمود اما اطلاعات حافظه ROM فقط قابل خواندن است و نمی توان آن را تغییر داد . در حافظه از نوع RAM ، محتوای حافظه با قطع جریان برق از بین می رود . حافظه های PROM یک حافظه ROM است با این تفاوت که برنامه توسط برنامه نویس نوشته می شود و توسط پروگرامر PROM در حافظه PROM قرار می گیرد و دیگر قابل تغییر نیست . حافظه های EPROM و EEPROM یک نوع حافظه ROM هستند با این تفاوت که برنامه ای که در حافظه قرار می گیرد را می توان تغییر داد . بدین صورت که حافظه را مدت مشخصی تحت تابش ماوراء بنفش قرار می دهیم ، اتصالات منطقی برنامه از بین می رود و EPROM آماده برنامه ریزی مجدد می شود و همچنین برای پاک کردن برنامه داخلی EEPROM از امواج الکتریکی استفاده می کنیم . حافظه ها از جنس نیمه هادی هستند .

واحدهای حافظه به سه دسته زیر تقسیم می شود :

1 – CPU که واحد پردازش کننده کامپیوتر است

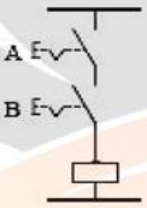
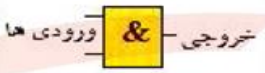
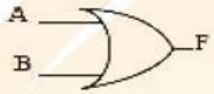
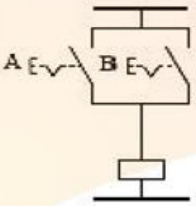
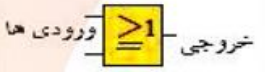
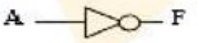
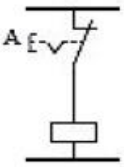
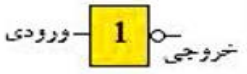
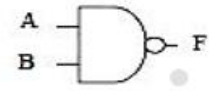
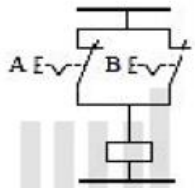
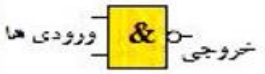
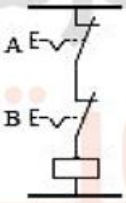
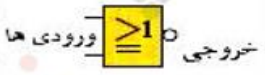
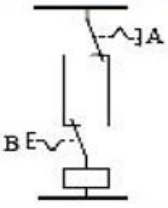
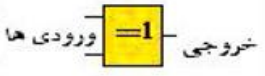
2 – CU که واحد کنترل پردازنده است

3 – ALU که واحد محاسبات منطقی و ریاضی است

منطق دیجیتال :

در مدار منطقی دیجیتال از المان های الکترونیکی نظیر دیود و ترانزیستور استفاده می شود . از ترکیب چند المان توابع منطقی ایجاد شده که هرکدام منطق خاصی را پیروی می کنند . در این مدارها از دو اصطلاح صفر و یک بسیار استفاده می کنند ، مفهوم این دو اصطلاح بدین شرح است : در یک سیستم تنها چیزی که برای المان های الکتریکی قابل فهم است ، بود یا نبود ولتاژ است چون منطق دیجیتال از این خاصیت تبعیت می کند پس باید دو سطح از ولتاژ را برای درک سیستم تعریف نمود مثل 0 ولت و 24 ولت . در این سیستم سطح ولتاژ 24 ولت ، یک و سطح ولتاژ 0 ولت ، صفر تلقی می شود .

توابع منطقی دیجیتال دارای یک یا چند ورودی و یک خروجی می باشند که وضعیت خروجی متناسب با وضعیت ورودی می باشد . در مدارهای منطقی یا دیجیتال عناصری وجود دارد که توانایی انجام عملیات بر روی صفر و یکها را دارند که به آنها گیت (Gate) می گویند . هفت گیت منطقی دیجیتال موجود می باشد : AND ، NOT ، NAND ، OR ، NOR ، XOR ، XNOR

عملگر منطقی	جدول صحت	علامت اختصاری	مدار کلیدی	شکل بلوکی
AND	خروجی ورودی ها A B F 0 0 0 0 1 0 1 0 0 1 1 1			
OR	خروجی ورودی ها A B F 0 0 0 0 1 1 1 0 1 1 1 1			
NOT	A F 0 1 1 0			
NAND	خروجی ورودی ها A B F 0 0 1 0 1 1 1 0 1 1 1 0			
NOR	خروجی ورودی ها A B F 0 0 1 0 1 0 1 0 0 1 1 0			
XOR	خروجی ورودی ها A B F 0 0 0 0 1 1 1 0 1 1 1 0			

مفهوم بیت :

با ترکیب چند تابع منطقی سلول حافظه تشکیل می شود ، این بدان معنی است که وضعیت صفر و یا یک بودن ورودی یا خروجی را در خود حفظ می کند ، به این سلول حافظه یک بیت گفته می شود .

اعداد را می توان در مبناهای عددی مختلف نمایش داد . آشناترین مبنای اعداد ، مبنای ده می باشد . در مبنای ده کلیه اعداد با ترکیبی از اعداد 0 تا 9 حاصل می گردند . از دیگر مبناهای عددی رایج می توان به مبنای دو اشاره نمود ، همانند اعداد مبنای ده هر رقم یک عدد در مبنای دو دارای ارزش خاصی می باشد . در این مبنا تنها اعداد صفر و یک موجود می باشند ، مثلا عدد 01101 یک عدد پنج رقمی در مبنای دو می باشد . هر رقم در مبنای دو را یک بیت و هر هشت بیت را یک بایت و هر دو بایت را یک کلمه می نامند .

- به هر 1024 بایت ، یک کیلوبایت می گویند .
- به هر 1024 کیلوبایت ، یک مگابایت می گویند .
- به هر 1024 مگابایت ، یک گیگابایت می گویند .
- به هر 1024 گیگابایت ، یک ترابایت می گویند .
- اعداد صحیح **32** بیتی در حافظه Dword ذخیره می شود .
- اعداد اعشاری **32** بیتی در حافظه Real ذخیره می شود .

جهت بدست آوردن معادل مبنای دو یک عدد دهدهی این عدد را بطور متناوب بر دو تقسیم می کنیم تا جاییکه خارج قسمت نهایی بر دو قابل تقسیم نباشد ، باقیمانده های بدست آمده را از انتها به ابتدا به ترتیب از چپ به راست بعد از آخرین خارج قسمت می نویسیم و اینگونه معادل دودویی اعداد بدست می آید :

$(41)_{10} = (?)_{2}$

$41 \div 2 \longrightarrow$ خارج قسمت 20 و باقیمانده 1

$20 \div 2 \longrightarrow$ خارج قسمت 10 و باقیمانده 0

$10 \div 2 \longrightarrow$ خارج قسمت 5 و باقیمانده 0

$5 \div 2 \longrightarrow$ خارج قسمت 2 و باقیمانده 1

$2 \div 2 \longrightarrow$ خارج قسمت 1 و باقیمانده 0

چون خارج قسمت بر دو بخش پذیر نیست لذا طبق روش گفته شده معادل باینری عدد را می نویسیم :

$(41)_{10} = (101001)_{2}$

جهت تبدیل یک عدد از مبنای دو به مبنای ده می توان هر رقم را در ارزش مکانی خود ضرب نمود و سپس حاصلضربهای بدست آمده را با هم جمع نمود :

$(101001)_{2} = (?)_{10}$

$101001 \longrightarrow 1 \times 32 + 0 \times 16 + 1 \times 8 + 0 \times 4 + 0 \times 2 + 1 \times 1 = 41$

از دیگر مبناهای عددی پرکاربرد مبنای **16** می باشد . یک عدد در مبنای **16** معادل یک عدد دودویی چهار رقمی است . جدول زیر مبنای شانزده و معادل دهدهی و دودویی آن را نشان می دهد .

مبنای شانزده	مبنای دو	مبنای ده
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
A	1010	10
B	1011	11
C	1100	12
D	1101	13
E	1110	14
F	1111	15

جهت تبدیل یک عدد دودویی به عدد مبنای شانزده کافی است از سمت راست اعداد را چهار رقم چهار رقم جدا نموده و سپس معادل مبنای شانزده آنها را جایگزین نماییم .

$$(10001101)_2 = (\ ?\ )_{16}$$

$$1000\ ,\ 1101 \Longrightarrow 8D$$

$$(10001101)_2 = (8D)_{16}$$

جهت تبدیل یک عدد از مبنای شانزده به مبنای دو به جای هر عدد معادل دودویی چهار رقمی آن را جایگزین می کنیم .

$$(A3B)_{16} = (\ ?\ )_2$$

$$1010\ ,\ 0011\ ,\ 1011 \Longrightarrow A\ ,\ 3\ ,\ B$$
$$(A3B)_{16} = (101000111011)_2$$

جهت تبدیل یک عدد دودویی به عدد مبنای هشت کافی است از سمت راست اعداد را سه رقم سه رقم جدا نموده و سپس معادل مبنای هشت آنها را جایگزین نماییم .

مبنای هشت	مبنای دو
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

$$(001101)_2 = (\ ?\ )_8$$

$$001\ ,\ 101 \Longrightarrow 1\ ,\ 5$$

$$(001101)_2 = (15)_8$$

جهت تبدیل یک عدد از مبنای هشت به مبنای دو به جای هر عدد معادل دودویی سه رقمی آن را جایگزین می کنیم .

$$(23)_8 = (\ ?\ )_2$$

$$010\ ,\ 011 \Longrightarrow 2\ ,\ 3$$

$$(23)_8 = (010011)_2$$

کد BCD : در BCD هر رقم در مبنای دهدهی بطور جداگانه به شکل دودویی کد می شود . هر رقم در چهار بیت کد می شود ، چون بزرگترین رقم دسیمال یعنی **9** در باینری چهار رقمی است .

اعداد صحیح Integer : INT عدد صحیح شانزده بیتی می باشد ، بیت پانزدهم نشان دهنده علامت عدد است . اگر صفر باشد عدد مثبت و اگر یک باشد عدد منفی می باشد . بازه این اعداد بین 32768 – تا 32767 + می باشد .

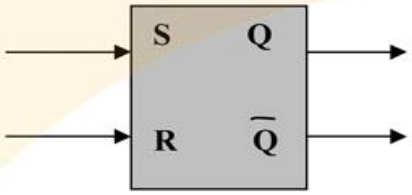
فلیپ فلاپ :

کوچکترین عنصر حافظه در یک مدار ترتیبی را فلیپ فلاپ می نامند . یک فلیپ فلاپ قادر است مادامیکه ورودیهاش تغییر نکرده و جریان تغذیه آن نیز قطع نشده باشد ، یک مقدار را بمدت نامحدود حفظ نماید . انواع مختلفی از فلیپ فلاپ وجود دارد که عبارتند از : فلیپ فلاپ نوع D ، فلیپ فلاپ نوع JK ، فلیپ فلاپ نوع RS ، فلیپ فلاپ نوع T . از آنجا که فلیپ فلاپ کاربردی در PLC فلیپ فلاپ RS می باشد ، به بررسی این فلیپ فلاپ خواهیم پرداخت .

فلیپ فلاپ RS :

جدول درستی و نمای شماتیکی این فلیپ فلاپ بصورت زیر است :

S	R	Q (t + 1)
0	0	Q(t) بدون تغییر
0	1	0
1	0	1
1	1	غیرقابل پیش بینی



فصل دوم

در این فصل مفاهیم اولیه PLC را بررسی می کنیم و زبان های برنامه نویسی و واحدهای تشکیل دهنده آن را معرفی خواهیم کرد . در حقیقت فصل حاضر مقدمه بحث اصلی این مجموعه که آموزش PLC است ، می باشد .

پیشرفت های چشمگیر فناوری نیمه هادی در زمینه ساخت ریزپردازنده و حافظه های با حجم بالا امکان ساخت کنترل کننده های منطقی الکترونیکی برنامه پذیر را فراهم آورد . در این کنترل کننده ها برخلاف کنترل کننده های مبتنی بر قسمت های الکترومکانیکی ، برای تغییر منطق کنترل کافی است بدون تغییری در سیم کشی یا قطعات ، فقط برنامه کنترل را تغییر دهیم . در اینصورت می توانیم از یک کنترل کننده منطقی برنامه پذیر هرجا که خواسته باشیم استفاده نماییم .

مزایای استفاده از کنترل کننده های منطقی برنامه پذیر :

1 – استفاده از PLC حجم تابلوهای فرمان را کاهش می دهد .

2 – استفاده از PLC موجب صرفه جویی فراوان در هزینه می گردد .

3 – PLC استهلاک مکانیکی ندارد بنابراین علاوه بر طول عمر بیشتر ، نیازی به سرویس و تعمیرات دوره ای ندارد .

4 – مصرف انرژی PLC بسیار کمتر از مدارهای رله ای است .

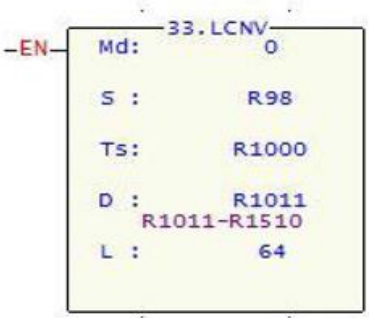
5 – PLC نویزهای صوتی و الکتریکی ایجاد نمی کند .

6 – عیب یابی مدارات کنترل با PLC سریع و آسان است و معمولا PLC خود دارای برنامه عیب یابی می باشد .

PLCها مبتنی بر میکروپروسسور هستند و با داشتن اجزایی مانند زمان سنج ، شمارنده و ثبات انتقالی ، کنترل فرآیندهای پیچیده را آسان می سازند . PLC ها کامپیوترهای تک منظوره ای هستند که از سه بخش تشکیل شده اند : ورودی ، حافظه و پردازش . اطلاعات ورودی از طریق سنسورها دریافت و در حافظه ذخیره می گردند ، این اطلاعات با توجه به فرامین موجود در حافظه پردازش شده و سپس خروجی ها به نحوی مناسب ساخته می شوند . هر PLC دارای زبان برنامه نویسی خاص خود بوده که رابط بین کاربر و سخت افزار PLC می باشد . مهمترین روشهای برنامه نویسی عبارتند از : روش نردبانی ، روش فلوچارتی ، روش لیست جملات .

برنامه نویسی به روش نردبانی :

از آنجا که تمام نقشه های کنترل و فرمان منطقی قبل از ظهور PLC ها بصورت نردبانی و یا چیزی شبیه به آن تهیه و طراحی می شد ، لذا سازندگان PLC این روش برنامه نویسی را بعنوان یکی از روش های ممکن برنامه نویسی انتخاب نمودند . در این روش آن دسته از عناصر نردبان که تابع یا عمل خاص و پیچیده ای را انجام می دهند برای سهولت با یک جعبه نمایش داده می شوند . دستورات نوشته شده به روش نردبانی بترتیب از چپ به راست و از بالا به پایین انجام می گردند . میتوان اینگونه بیان داشت که این زبان براساس نقشه های مدار فرمان ، طراحی شده است .



برنامه نویسی به روش فلوچارتی یا نمایش جعبه ای تابع :

در این روش برنامه بصورت بلوکی نوشته شده که در آن هر بلوک بیانگر یک عملگر می باشد ، بدین ترتیب برنامه های نوشته شده به روش فلوچارتی عبارتند از یک سری جعبه که به یکدیگر متصل گردیده اند . این روش معمولا بطور مستقل کاربرد چندانی ندارد و اغلب برای عیب یابی و یا شناخت منطق کنترل سیستم بسیار مفید است . این زبان براساس مدارهای الکترونیک و دیجیتال طراحی شده است .

برنامه نویسی به روش لیست جملات:

```
00010M   ORG    TD    X2
00011M   AND          M5
00012M   LD           M5
00013M   AND          X2
00014M   ORLD
00015M   RST(  )      M5
```

در این روش هر عمل منطقی توسط یک جمله یا عبارت مناسب نوشته می شود . نکته قابل توجه در این روش برنامه نویسی آن است که هر PLC دارای کد دستورات منحصر بفردی می باشد که این دستورات به نوع CPU بکار رفته بستگی دارد . این زبان براساس زبان برنامه نویسی کامپیوتر ایجاد شده است . زبان برنامه نویسی در حالت لیست جملات مثل زبان بیسیک یا اسمبلی بوده و نوشتاری است . روش لیست جملات نیازهای گرافیکی بسیار کمتری نسبت به دو روش قبل دارد ، لذا نوع و تعداد دستورات قابل درک و اجرا در این روش بیشتر از روش های نردبانی و فلوچارتی می باشد . به همین دلیل برنامه هایی که به روش نردبانی یا فلوچارتی نوشته می شود معمولا قابل تبدیل به لیست جملات می باشد ، درحالیکه عکس این قضیه همواره ممکن نیست . در برنامه نویسی به روش لیست جملات هر چند خط برنامه که عمل خاصی را انجام می دهد یک Segment می گویند .

عبارت یا Statement :

Statement یا هر خط از برنامه نوشته شده به روش لیست جملات ، سطری از برنامه است که معمولا دارای دو بخش زیر است :

عملگر یا Operation : به عمل منطقی که در عبارت صورت می گیرد ، عملگر گفته می شود . عملگرهای مهم عبارتند از : AND ، OR ، NOT

عملوند یا Operand : به قسمتی از عبارت گفته می شود که قرار است یک عمل منطقی (عملکرد) در مورد آن اجرا شود مانند ورودی ها ، خروجی ها . عملوند خود شامل دو بخش آدرس عملوند و نوع عملوند است . نوع عملوند ، همان ورودی ها ، خروجی ها و غیره هستند و آدرس عملوند ، محل عملوند را مشخص می نماید .

انواع PLC :

PLC ها از لحاظ سخت افزاری به دو گونه کلی در دسترس می باشند :

الف : کامپکت (یکپارچه) : PLC یکپارچه خودکفاست . یعنی سخت افزار آن ، منبع تغذیه ، CPU و تعداد محدودی ورودی و خروجی را بصورت یک بسته یکپارچه شامل می شود . این نوع PLC مختصرتر ، ساده تر و ارزان تر و دارای عملکردی محدودتر از گونه دیگر می باشد .

ب : ماژولار : PLC ماژولار از کنار هم قرار گرفتن ماژول های مختلف ، مانند ماژول منبع تغذیه ، ماژول CPU ، ماژول های ورودی و خروجی ، کارت های شبکه ساخته می شوند .

پردازنده CPU :

این واحد اساسی ترین قسمت PLC می باشد . واحد پردازش ، یک ریزپردازنده است و مجموعه اعمال و محاسبات منطقی را انجام می دهد و ارتباط بین واحدهای مختلف را برقرار می سازد . در واحد پردازش عناصر دیگری مثل شمارنده ها و تایمرها تعریف شده اند . اغلب CPUها مجهز به یک باتری پشتیبان هستند ، بنابراین اگر تغذیه ورودی قطع شود و متعاقبا منبع تغذیه نتواند ولتاژ سیستم را تامین کند ، باتری پشتیبان برنامه ذخیره شده در RAM را حفظ می کند .

ماژول های ورودی و خروجی :

ماژول ورودی بصورت الکترونیکی چهار کار اصلی را انجام می دهد ، اولا این ماژول حضور یا عدم حضور سیگنال الکتریکی در تمام ورودی ها را بررسی می کند . ثانیا این ماژول سیگنال مربوط به وصل بودن را از نظر الکتریکی به سطح DC که توسط مدارات الکتریکی ماژول I/O قابل استفاده باشد ، تغییر می دهد . ثالثا این ماژول جداسازی الکترونیکی را با جداکردن خروجی ماژول ورودی از ورودی اش بصورت الکترونیکی انجام می دهد . در نهایت این ماژول سیگنالی را که توسط CPU سیستم PLC قابل تشخیص است ، ایجاد می کند . ماژول خروجی بگونه ای عکس ماژول ورودی عمل می نماید . یک سیگنال DC که از CPU ارسال می گردد ، در هر ماژول خروجی به سیگنال الکتریکی با سطح ولتاژ مناسب بصورت AC یا DC که توسط دستگاه ها قابل استفاده باشد ، تبدیل می گردد .

منبع تغذیه :

منبع انرژی الکتریکی که معمولا استفاده می شود ، منبع جریان متناوب 220 ولت با فرکانس 50 الی 60 هرتز می باشد . از آنجا که اغلب PLCها با ولتاژهای +5 ، - 5 و 24 ولت کار می نمایند لذا هر PLC باید مجهز به مدارهایی باشد که بتواند این تبدیل ولتاژها را انجام دهد . این تبدیل با استفاده از یک منبع تغذیه داخلی انجام می شود .

برنامه ریز PLC :

برای نوشتن برنامه در PLC از وسیله ای بنام PG (Programmer) برنامه ریز دستی استفاده می شود . امروزه برای نوشتن برنامه PLC عمدتاً از دستگاه کامپیوتر استفاده می شود زیرا کاربرها با کامپیوتر و دکمه های آن آشنایی کافی داشته و دستگاهی چند منظوره است که با نصب نرم افزار مربوط به PLC ، به راحتی مورد استفاده قرار می گیرد .

حافظه PLC :

حافظه PLC معمولاً از دو قسمت تشکیل شده است :

یک قسمت در دسترس استفاده کننده بوده و مخصوص نوشتن برنامه کنترل می باشد ، این قسمت قابل پاک کردن و تغییر است و معمولاً از نوع RAM می باشد . قسمت دیگر حافظه سیستم است که مربوط به نحوه عملکرد مدارات داخلی PLC می باشد و معمولاً استفاده کننده از PLC سروکاری با آن نداشته و توسط کمپانی سازنده پر می شود . این قسمت به راحتی قابل پاک کردن و تغییر نیست و معمولاً از نوع EPROM یا EEPROM می باشد .

اجرای متناوب و چرخشی برنامه :

مجموعه دستورالعمل هایی که کاربر جهت کنترل سیستمی در PLC ، به کمک نرم افزار ایجاد می کند را برنامه کاربر می گویند . سرعت اجرای برنامه و عملیات آن به عملکرد CPU ، سرعت آن و همچنین حجم برنامه ، تعداد I/O و ارتباطات بستگی دارد . روش معمول اجرای برنامه در PLC بدین ترتیب است که تمامی ورودی ها خوانده می شوند ، برنامه کاربر پردازش می شود و خروجی ها را ایجاد می نماید ، سپس تمامی خروجی ها اعمال می شوند . بدین ترتیب به یک چرخه ، پیمایش (Scan) و به مدت انجام آن ، زمان چرخه (Cycle Time) گفته می شود .

کوپل کننده های نوری :

جهت حفاظت مدارات داخلی PLC و جلوگیری از نویزهائی که معمولا در محیط های صنعتی وجود دارند ، ارتباط ورودی ها با مدارات داخلی PLC توسط کوپل کننده های نوری (Opto Coupler) انجام می گردد . در داخل PLC ایزولاسیون الکتریکی توسط آپتوکوپلر انجام می شود .

فصل سوم

اولین کنترل کننده منطقی برنامه پذیر در سال **1969** و در راستای کاهش زمان توقف خطوط تولید ، به وسیله کارخانه اتومبیل سازی General Motors بکار گرفته شد . کمپانی FATEK یکی از بزرگترین تولیدکنندگان PLC در تایوان از سال **1992** فعالیت خود را آغاز نمود . از جمله خصوصیات این کمپانی کیفیت و قابلیت بالای آن می باشد . این کمپانی با تولید سری FB در سال **1992** کار خود را آغاز نمود و در سال **2003** ، نسل جدید PLC های خود با نام سری FBs که از تکنولوژی جدید System On Chip بهره می برند ، را وارد بازار نمود . در این فصل خواننده گرامی با مفاهیم سخت افزاری و نرم افزاری PLC FATEK آشنا می شود . در ابتدا مفاهیم سخت افزاری مورد بحث قرار می گیرد و در ادامه به مفاهیم نرم افزاری پرداخته می شود . در فصل آینده برنامه نویسی FATEK و کار با توابع برنامه نویسی ، بیان خواهد شد .

سخت افزار :

PLC های FATEK از سه قسمت اصلی تشکیل شده است :

1 – منبع تغذیه (Power Supply)

2 – برد اصلی (Main board)

3 – واحد ورودی / خروجی (Input / Output)

منبع تغذیه :

این نوع PLC دارای سه نوع تغذیه 220VAC ، 24VDC و 12VDC می باشد که برد تغذیه ، ولتاژ ورودی را به ولتاژ (12VDC یا 5VDC) جهت استفاده ریزپردازنده و واحدهای ورودی و خروجی تبدیل می نماید و همچنین وظیفه ایزوله نمودن ولتاژ مورد استفاده در PLC را ، از برق موجود در سیستم به عهده دارد و بدین ترتیب ایمنی سیستم را در برابر نویز و نوسانات ولتاژ ورودی افزایش می دهد . علاوه بر این ، تمام واحدهای تغذیه در این PLC ، یک خروجی 24VDC نیز در اختیار کاربر قرار می دهند .

برد اصلی :

واحد پردازنده مرکزی : ریزپردازنده با در نظر گرفتن وضعیت ورودی ها ، برنامه موجود در حافظه را اجرا می نماید و براساس آن به واحد خروجی دستور فعال کردن خروجی های موردنظر را می دهد .

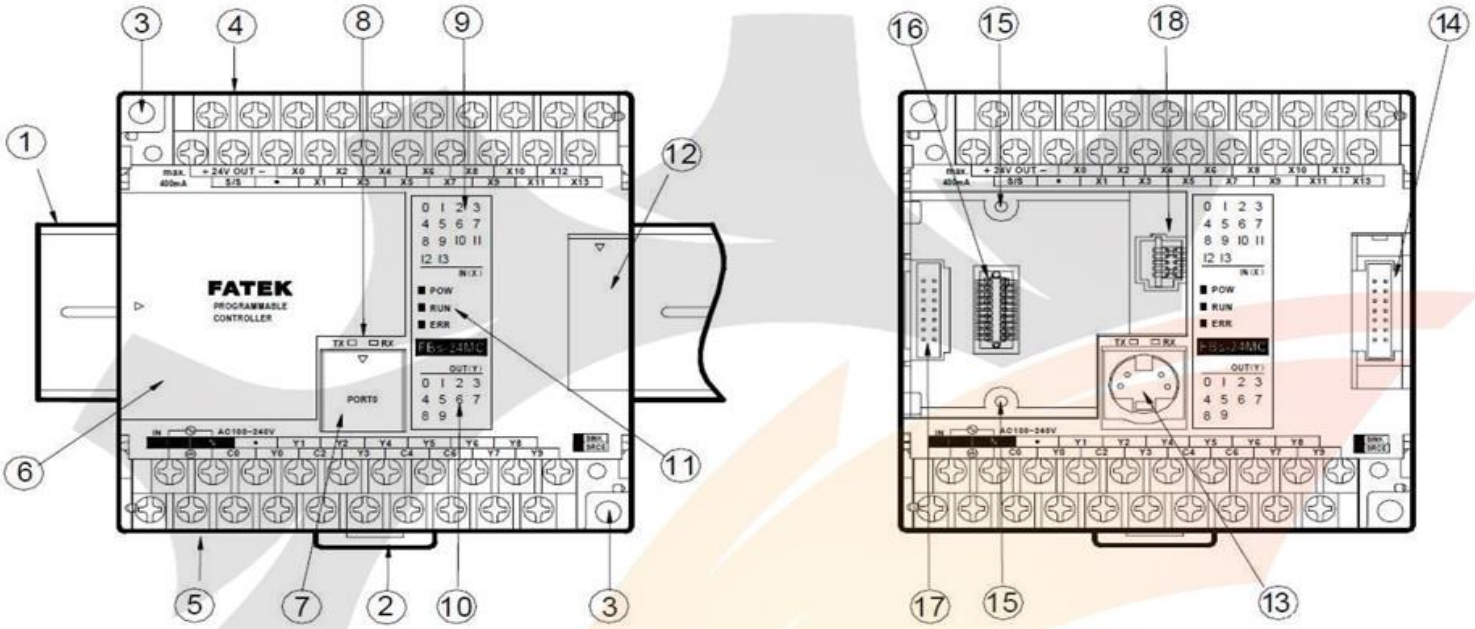
حافظه : جهت ذخیره سازی برنامه و اطلاعات از آن استفاده می شود . ظرفیت حافظه برنامه نویسی Fatek ، 20k Word می باشد . انواع حافظه موجود در PLC عبارت است از :

الف : حافظه سیستم عامل PLC : حافظه فقط خواندنی جهت ذخیره سازی الگوریتم عملکرد PLC استفاده می گردد .

ب : حافظه اطلاعات : حافظه خواندنی / نوشتنی جهت ذخیره اطلاعات لازم در طول اجرای برنامه و همچنین اطلاعات مربوط به ابزارهای برنامه نویسی مانند تایمرها ، شمارنده ها ، و رله های داخلی می باشد .

ج : حافظه جهت ذخیره سازی برنامه : این حافظه جهت نگهداری برنامه در داخل PLC استفاده می شود . این حافظه بصورت CMOS بوده و در صورت قطع برق محتویات آن توسط باتری پشتیبان حفظ خواهد شد . باتری پشتیبان این PLC از نوع لیتیوم با حداکثر طول عمر **10** سال می باشد . همچنین این PLC دارای یک Flash Rom جداگانه ای است که می تواند به صورت اختیاری برای ذخیره برنامه و اطلاعات ، مورد استفاده قرار گیرد و دارای 64Kword حافظه می باشد .

د : پورت ارتباطی : جهت انتقال برنامه از کامپیوتر به PLC و برعکس و همچنین جهت ارتباط PLC با HMI از آن استفاده می شود .



1 – ریل با عرض **35** میلی متر

2 – نگهدارنده PLC بر روی ریل

3 – محل بستن پیچ محکم کننده PLC بر روی تابلو

4 – ترمینال های خروجی 24VDC و ترمینال های ورودی های PLC

5 – ترمینال های ورودی تغذیه 220VAC و ترمینال های خروجی های PLC

6 – پوشش استاندارد PLC بدون برد ارتباطی

7 – پوشش محل اتصال کابل ارتباطی PLC به کامپیوتر

8 – نمایشگر حالت ارسال و دریافت اطلاعات به و از PLC

9 – نمایشگر ورودی های PLC

10 – نمایشگر خروجی های PLC

11 – نمایشگر وضعیت سیستم (RUN ، STOP ، ERROR)

12 – پوشش محل ارتباط ماژول های گسترش

13 – محل اتصال کابل ارتباطی PLC به کامپیوتر (استاندارد RS232 و USB)

14 – محل اتصال ماژول گسترش (این اتصال توسط کابل ماژول گسترش انجام می شود)

15 – سوراخ پیچ برای اتصال برد ارتباطی

16 – کانکتور جهت اتصال ماژول ارتباطی به PLC (برای ماژول های CB25 , CB22 , CB5 , CB2)

17 – کانکتور جهت اتصال ماژول ارتباطی به PLC (برای ماژول های CM55E , CM25E , CM55 , CM22 , CM25)

18 – محل اتصال حافظه خارجی

LED های نمایشگر وضعیت PLC :

POW : این LED قرمز رنگ هنگام اتصال تغذیه PLC ، بصورت ممتد روشن می ماند .

RUN : هرگاه PLC در حالت STOP باشد ، این LED سبز رنگ هر دو ثانیه یکبار چشمک می زند و هرگاه PLC در حالت RUN باشد ، هر 0.25 ثانیه یکبار چشمک می زند .

ERR : هرگاه PLC به طریقی دچار خطا شود ، این LED قرمز رنگ ، چشمک می زند .

خروجی PLC FATEK :

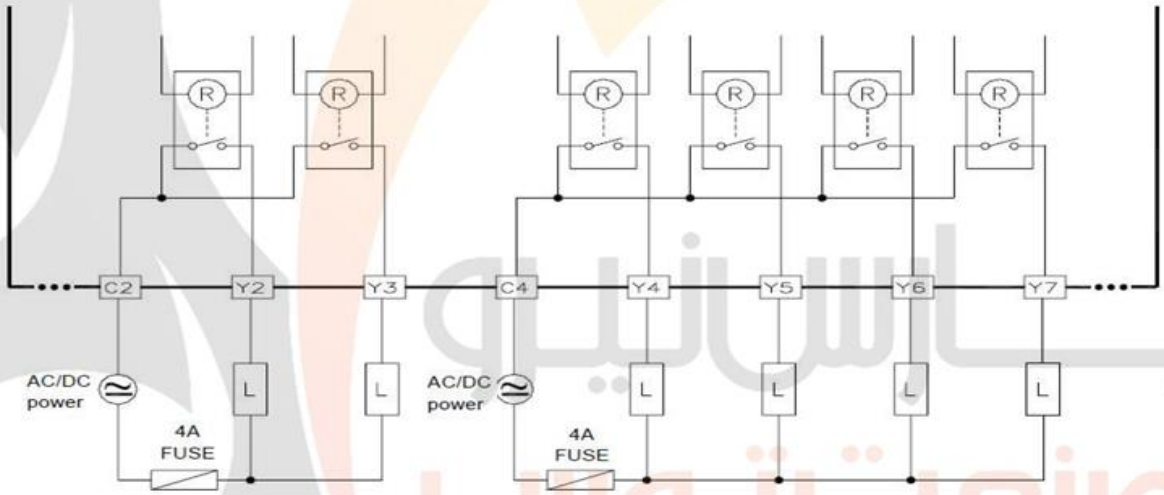
خروجی ها در سه نوع رله ای ، ترانزیستوری (مربوط به وسایل خروجی DC) و تریاکی (مربوط به وسایل خروجی AC) ساخته می گردند که برای حفاظت در برابر اضافه جریان به فیوز مجهز می باشند .

الف : رله ای : در این نوع ، فرمان CPU یک رله را فعال می کند و از طریق کنتاکت این رله ، خروجی فعال می شود . از مزیت های این نوع خروجی می توان به توانایی آن در قطع و وصل خروجی های DC و AC با جریان دو آمپر ، همچنین به استقامت آن در برابر شوک های حاصل از بارهای القایی اشاره کرد . سرعت خروجی های رله ای کند و به دلیل مکانیکی بودن عملکرد رله دارای محدودیت قطع و وصل می باشد .

ب : ترانزیستوری : در این نوع فرمان CPU یک ترانزیستور را فعال می کند و از طریق آن وسیله خروجی فعال می گردد . از مزیت های این نوع خروجی می توان به سرعت بالا و تعداد نامحدود قطع و وصل آن اشاره نمود . اما این نوع خروجی در برابر شوک های ناشی از قطع و وصل بارهای سلفی و جریان های بالا بسیار حساس می باشد و همچنین تنها برای قطع و وصل بارهای DC قابل استفاده می باشد . اگر خروجی ترانزیستوری NPN باشد ، به آن خروجی Sink و اگر PNP باشد به آن Source می گویند .

ج : تریاکی : مانند نوع ترانزیستوری می باشد ولی تنها در مورد بارهای AC قابل استفاده است .

خروجی ها با حرف Y نمایش داده می شوند و هر دو یا چهار خروجی ، دارای یک خروجی مشترک با هم می باشند که با حرف C نشان داده می شوند .



ورودی های دیجیتال در FBs تا **256** عدد قابل افزایش است .

خروجی های دیجیتال نیز تا **256** عدد قابل افزایش است .

حداکثر تا **32** ماژول (دیجیتال و آنالوگ) را می توان به CPU اصلی اضافه کرد .

ورودی ها و خروجی های آنالوگ نیز ، هرکدام تا **64** عدد قابل افزایش هستند .

ورودی ها با حرف X نمایش داده می شوند ، ترمینال مشترک ورودی ها به روی PLC با عبارت S/S مشخص شده است .

هرگاه 24+ به S/S متصل شود ، پالس های منفی برای ورودی ها ، معادل یک در نظر گرفته شده و هرگاه 24- به S/S متصل شود ، پالس های مثبت برای ورودی ها ، معادل یک در نظر گرفته می شود .

پاسخ زمانی PLC :

نحوه انجام عملیات در سیستم PLC بصورت زیر است :

- PLC تمام ورودی ها را بررسی می کند ، ورودی هایی که وصل هستند از نظر PLC معادل یک و کلیدهایی که قطع هستند معادل صفر در نظر گرفته می شوند .
- CPU برنامه موجود در حافظه را خط به خط خوانده و اجرا می کند .
- PLC پس از پایان اجرای برنامه ، وضعیت خروجی ها را به واحد خروجی می فرستد .
- این سیکل مجدد از شماره یک آغاز می شود .

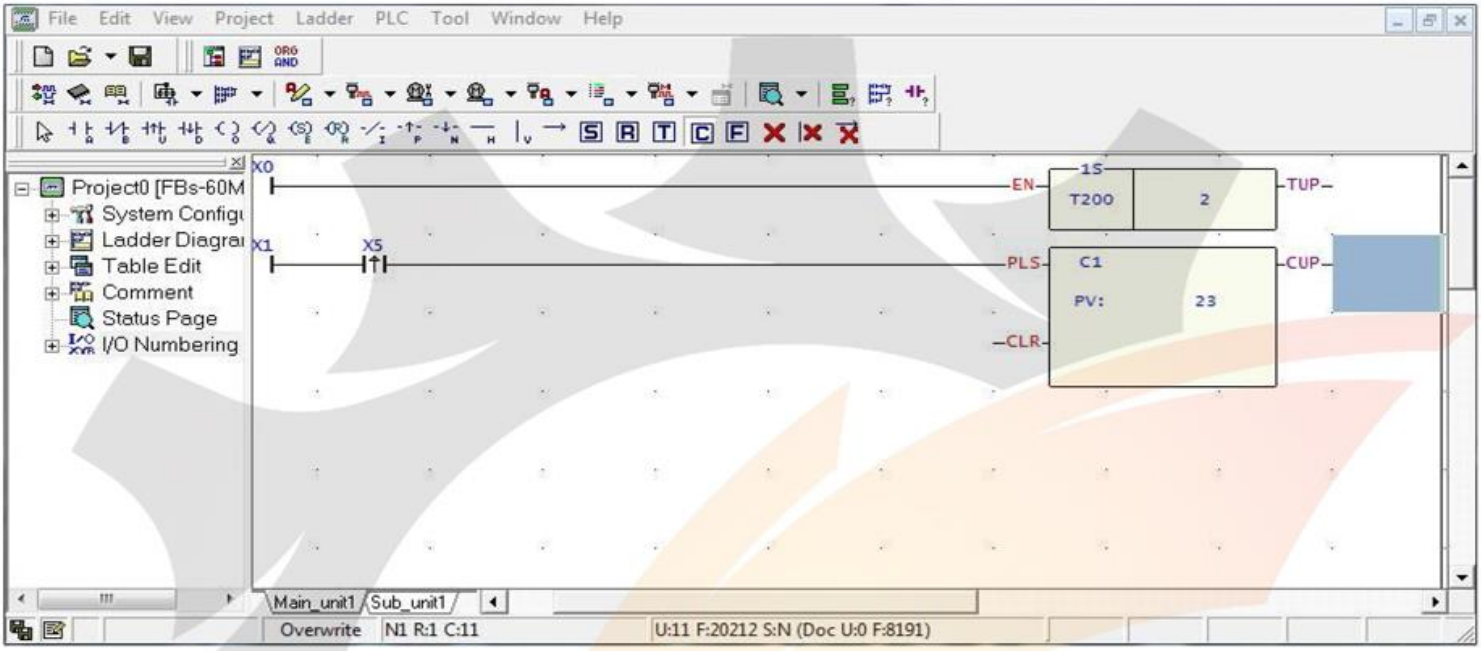
کل زمان انجام مراحل فوق را Scan Time می نامند . چنانچه این زمان بیشتر از **0/25** ثانیه گردد ، نشان دهنده این مطلب می باشد که یکی از قسمت های PLC دچار اشکال شده است . بنابراین تایمر Watch Dog عمل می نماید و تمامی خروجی ها را غیرفعال می کند تا عملکرد اشتباه PLC منجر به حادثه نگردد . این زمان پیش فرض ، از طریق تابع **90** قابل تغییر است .

رابط برنامه نویسی :

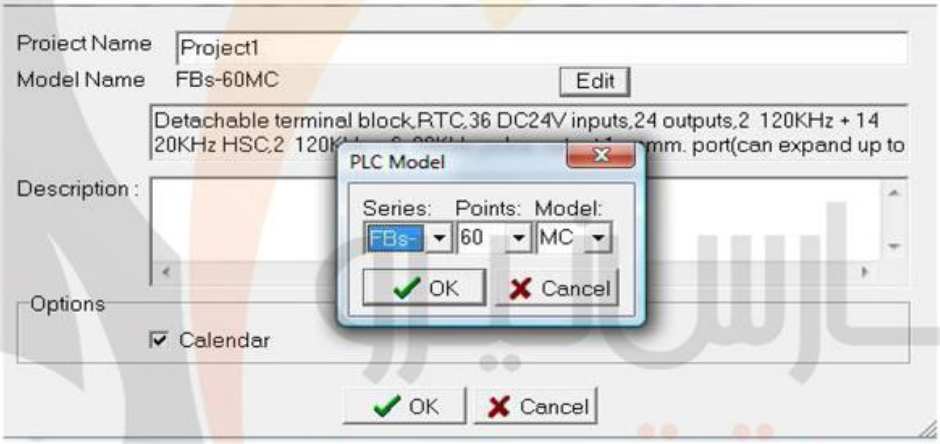
روش برنامه نویسی ، استفاده از کامپیوترهای شخصی و یا رومیزی و نرم افزار ویژه برنامه نویسی (WinProladder) می باشد . بنابراین کاربر از طریق کامپیوتر می تواند مستقیم برنامه موجود در حافظه PLC را مشاهده و تغییر دهد و یا ابتدا برنامه را در داخل کامپیوتر شخصی بنویسد و سپس در موقع مناسب آن را به PLC منتقل نماید . هرگاه برنامه در حالت RUN قرار گیرد ، برنامه اجرا می گردد . برخی قابلیت های نرم افزار برنامه نویسی Fatek (WinProladder) بشرح زیر می باشد :

1 – امکان نوشتن برنامه بصورت Offline و ذخیره آن بصورت یک فایل جهت دسترسی دوباره به برنامه فوق
2 – مشاهده اجرای یک برنامه در حال کار روی PLC
3 – مشاهده اجرای یک برنامه در حال کار بدون استفاده از PLC (محیط شبیه ساز)
4 – قابلیت قطع و وصل هر ورودی ، فعال و غیرفعال کردن هر خروجی در حین اجرای برنامه
5 – امکان تغییر برنامه در حال اجرا
6 – امکان نظارت و تغییر حالت ورودی ها ، خروجی ها و حافظه داخلی PLC از طریق صفحه مانیتورینگ
7 – تهیه نسخه چاپی از برنامه ، پیکربندی ورودی ها و خروجی ها ، جداول تنظیم کننده ، توضیحات و .....
8 – امکان پیدا کردن سریع هر ورودی یا خروجی دلخواه در برنامه و جایگزین نمودن آنها
9 – امکان قراردادن توضیحات اضافی در برنامه
10 – قابلیت چک کردن برنامه از لحاظ خطای برنامه نویسی
11 – امکان قراردادن رمز برای کل برنامه یا فقط زیر برنامه ها
12 – مدیریت پروژه با دسته بندی قسمت های مختلف برنامه
13 – امکان اتصال PLC و PC با روش های متنوع (RS232 ، Ethernet ، Modem)

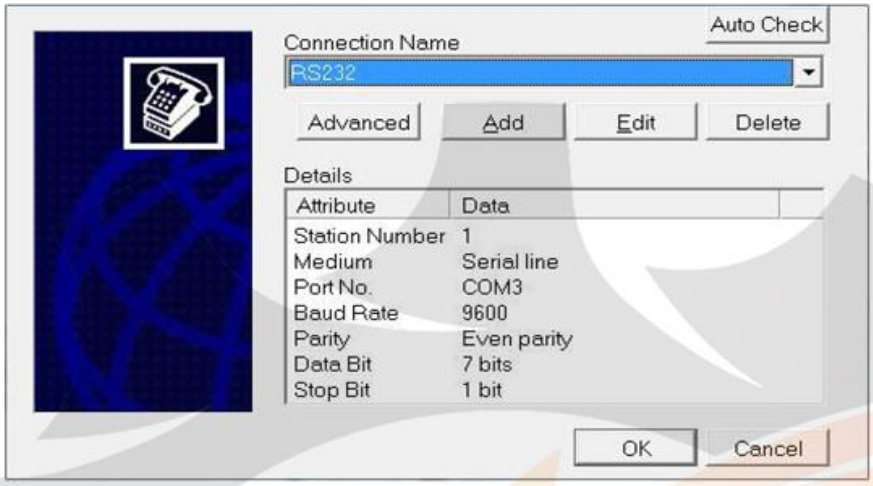
نرم افزار WinProladder :



برای شروع بعد از وارد شدن به برنامه WinProladder ، از منوی File گزینه New Project را انتخاب کنید . در پنجره که باز می شود ، نام پروژه مورد نظر خود را بنویسید و با کلیک بر روی Edit ، مدل PLC مورد استفاده خود را وارد نموده و OK کنید .



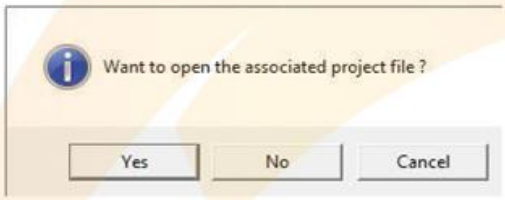
برای اینکه برنامه ای را که در نرم افزار نوشته اید ، در PLC ذخیره کنید ، از منوی File ، گزینه Save As را انتخاب کنید . پنجره ای باز می شود که در آن می توانید نوع ارتباط PLC با PC را تنظیم نمایید . اگر از مشخصات پورت مورد استفاده مطلع نیستید ، به روی گزینه Auto Check در پنجره باز شده کلیک کنید تا بصورت خودکار مشخصات پورت مورد استفاده شما ، مشخص شود .



برای راه اندازی PLC ، پس از Online شدن ، از منوی PLC ، گزینه Run PLC را انتخاب نموده و یا کلید F9 را فشار دهید .

نحوه خواندن برنامه از روی PLC :

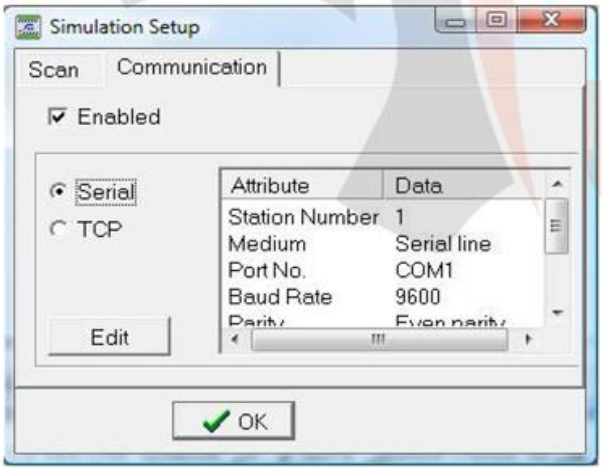
برای انجام این کار ، از منوی File گزینه Open و سپس Connect to PLC را انتخاب نمایید . سپس پنجره زیر ظاهر خواهد شد .



در این مرحله گزینه No را انتخاب نمایید . در پنجره بعد ، پس از انتخاب OK ، نرم افزار شروع به برقراری ارتباط با PLC نموده و در صورت عدم وجود رمز عبور ، برنامه را در اختیار شما قرار می دهد .

شبیه سازی برنامه بدون استفاده از PLC :

پس از نوشتن برنامه خود ، برای شروع شبیه سازی ، از منوی PLC ، گزینه Simulation را انتخاب کرده و برنامه نوشته شده را از گزینه Run PLC راه اندازی کنید . بیشتر توابع در این قسمت شبیه سازی می شوند اما توابع با کاربری پیشرفته که امکان شبیه سازی ندارند با رنگ زرد نمایش داده می شوند . در این حالت امکان اتصال HMI به کامپیوتر نیز وجود دارد . به این ترتیب که پس از راه اندازی Simulation ، از منوی PLC ، گزینه Setup Simulation را انتخاب کرده و وارد قسمت Communication در پنجره ظاهر شده می شویم . با تیک زدن گزینه Enabled ، تنظیمات پورت کامپیوتر ظاهر می شود که باید با تنظیمات پورت HMI منطبق باشد و در قسمت Port No ، شماره Com Port کامپیوتر که به HMI متصل است ، قرار داده می شود .



رمز گذاری (Password) :

برای جلوگیری از بازشدن برنامه از داخل کامپیوتر یا از داخل PLC توسط نرم افزار ، برای برنامه خود ، اسم رمز بگذارید تا امکان دسترسی به برنامه فقط برای افرادی که اسم رمز را دارند ، میسر باشد . برای این کار به ترتیب زیر عمل کنید :

Project > Project Setup > Password

در پنجره ظاهر شده ، رمز مورد نظر خود را در قسمت New Password وارد کرده و مجددا در قسمت Confirm Password وارد کنید . اگر گزینه Protect Sub Program Only را تیک بزنید ، تنها از زیر برنامه های حفاظت شده و دسترسی به برنامه اصلی برای همه میسر خواهد بود . در ضمن بدون داشتن رمز عبور می توان برنامه رمزدار را به داخل PLC انتقال داده و راه اندازی نمود .

نوع دیگری از حفاظت برای برنامه ، گذاشتن Program ID برای برنامه و PLC ID برای PLC (تنها در حالت متصل به PLC و Online) است . به این ترتیب ، برنامه مورد نظر تنها در PLC ای Run می شود که Program ID با PLC ID یکی باشد . برای تعیین Program ID و PLC ID به ترتیب زیر عمل کنید :

Project > Project Setup > Program ID
Project > Project Setup > PLC ID

فصل چهارم

زبان ماشین مجموعه ای از کدهای باینری می باشد که تنها برای ریزپردازنده قابل درک است . از این رو برنامه نویسی با آن برای مهندسین دشوار می باشد . جهت سهولت در امر برنامه نویسی ، شرکت های سازنده PLC نیز هر کدام از زبان های سطح بالا خاص خود بهره می گیرند . در سال **1988** کمیته بین المللی الکتروتکنیکال IEC را به جهت شبیه ساختن زبان های برنامه نویسی در PLC منتشر ساخت . با وجود این هنوز به دلایل بسیاری سازندگان PLC از زبان های مختص به خود استفاده می نمایند . در فصل مفاهیم PLC در مورد زبان های برنامه نویسی مختلف صحبت شد زبان برنامه نویسی PLC FATEK از طریق نرم افزار WinProladder ، ترکیبی از نردبانی و فلوچارتی است .

دیاگرام نردبانی :

برای جایگزین ساختن یک سیستم کنترل مبتنی بر رله با یک PLC نیاز به تبدیل مدارهای فرمان با زبان برنامه نویسی PLC می باشد . استفاده از زبان LD ، این تبدیل را بسیار ساده می نماید . دیاگرام نردبانی از دو خط موازی تشکیل شده است که نشان دهنده خطوط تغذیه مدار می باشند و خطوط افقی که مانند پله های نردبانی می باشند ، خطوط برنامه هستند . هنگام نوشتن برنامه به زبان LD موارد زیر را به خاطر بسپارید :

1 – هر خط از برنامه وظیفه خاصی را به عهده دارد .

2 – در PLC برنامه از از سمت چپ به راست و از بالا به پایین اجرا می گردد و بعد از اجرای کامل برنامه ، اجرای آن دوباره از سر گرفته می شود .

3 – هر خط برنامه با تعدادی کنتاکت باز و یا بسته آغاز و با یک یا چند بوبین رله به انتها می رسد . این رله ها می توانند کمکی و یا خروجی باشند

4 – کنتاکت ها در وضعیت عادی خود در برنامه نشان داده می شوند . بعبارت دیگر کنتاکت های داخلی با فرض غیرفعال بودن رله ها نمایش داده می شوند .

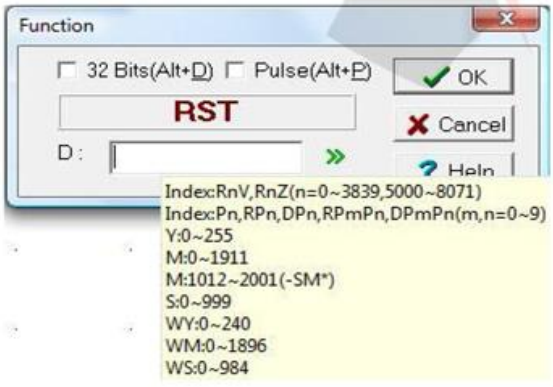
5 – از کنتاکت های یک رله می توان در خطوط مختلف برنامه استفاده نمود

6 – هر کدام از کنتاکت های ورودی و رله های خروجی دارای آدرس منحصر به فرد می باشند . بعنوان مثال : FBs – 40MA دارای **24** ورودی و **16** خروجی می باشد که آدرس آنها به ترتیب ذیل است :

خروجی ها : Y0 ~ Y15 ورودی ها : X0 ~ X23

نوار المان ها :

برای استفاده از این آیتم ها ، ابتدا به روی آنها کلیک کرده تا انتخاب شوند . سپس به روی پنجره دیاگرام نردبانی کلیک کنید و ورودی یا خروجی مربوط را وارد کنید ، یا می توانید مستقیم از طریق صفحه کلید حرف مربوط به هر المان را که در زیر آن آمده است ، به پنجره دیاگرام نردبانی وارد کنید . اگر نشانه گر موس را به روی پنجره باز شده نگهدارید ، تمام المان هایی که می توان در آن قسمت وارد کرد نمایش داده می شود .



کنتاکت باز ورودی

کنتاکت بسته ورودی

کنتاکت ورودی که تنها لحظه ایجاد پالس بالارونده به خروجی فرمان می دهد (هرگاه پالس از **0** به **1** تغییر کند)

کنتاکت ورودی که تنها لحظه ایجاد پالس پایین رونده به خروجی فرمان می دهد (هرگاه پالس از **1** به **0** تغییر کند)

کنتاکت باز خروجی

کنتاکت بسته خروجی

کنتاکت Set کننده خروجی (latch می کند)

کنتاکت reset کننده خروجی (latch می کند)

این المان شرایط قبل خود را معکوس می کند (Not)

هرگاه جلوی کنتاکت ورودی قرار گیرد ، باعث می شود که فرمان ورودی ها تنها در لحظه بالارونده پالس به خروجی اعمال شود

هرگاه جلوی کنتاکت ورودی قرار گیرد ، باعث می شود که فرمان ورودی ها تنها در لحظه پایین رونده پالس به خروجی اعمال شود

خط افقی کوتاه

خط عمودی

خط افقی طولانی

المان مورد نظر را Set می کند (به یک بازنشانی می کند)

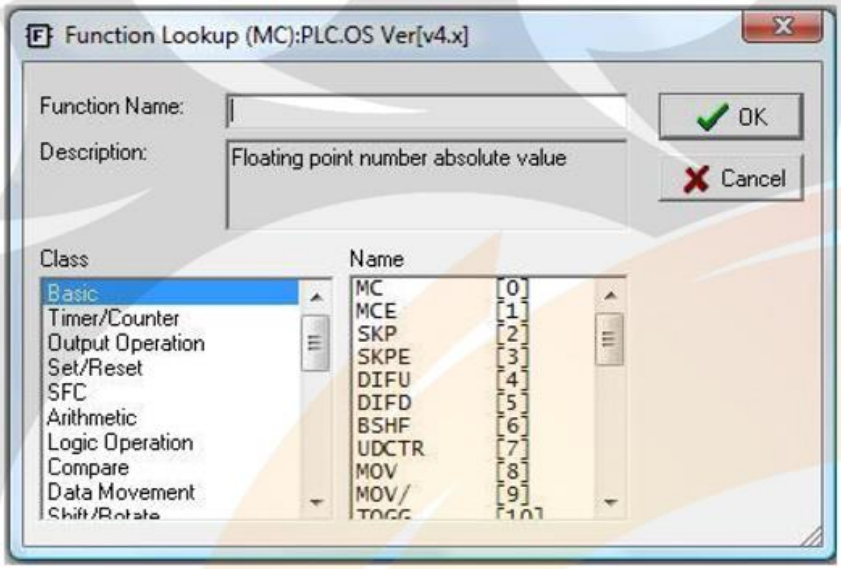
المان مورد نظر را Reset می کند (به صفر بازنشانی می کند)

تایمر

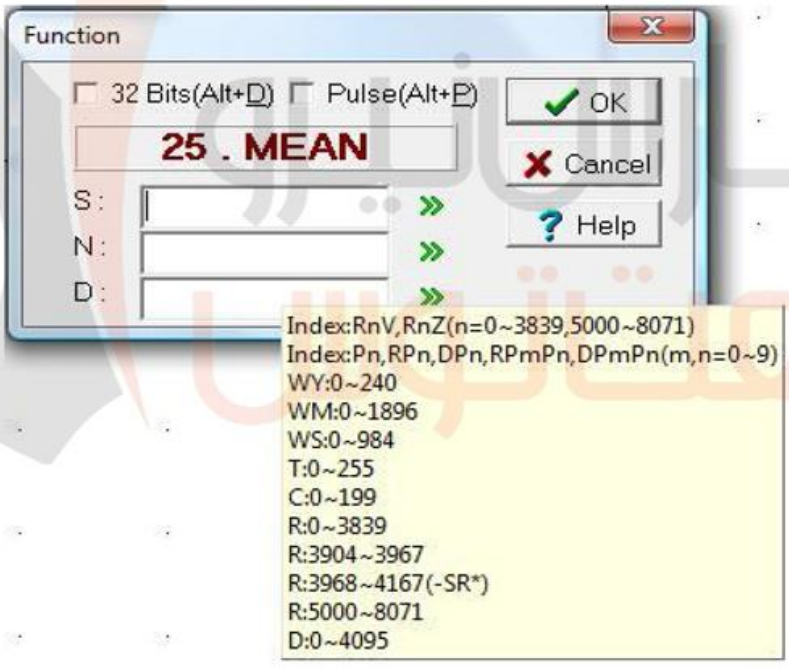
کانتر (شمارنده)

سایر توابع برنامه نویسی

نرم افزار WinProladder دارای حدود **220** تابع در دسته های مختلف عملیات منطقی ، ریاضی ، ارتباطات ، مقایسه ای و ..... می باشد . این توابع از طریق آیکون F در منوی Ladder قابل دسترسی هستند . پس از فشردن این آیکون ، سپس کلیک به روی صفحه برنامه نویسی ، صفحه زیر ظاهر می گردد .



برای انتخاب تابع موردنظر ، ابتدا از قسمت Class ، دسته بندی مرتبط با تابع را انتخاب کرده ، سپس از قسمت Name ، تابع را انتخاب می کنیم . همچنین می توان در قسمت Function Name ، نام تابع یا شماره آن را تایپ کرد . پس از OK کردن ، پنجره ای باز می شود که رجیستر یا بیت دلخواه برای انجام عملیات را وارد خانه های خالی می کنیم . اگر نشانگر موس را برای چند لحظه به روی این خانه های خالی نگه دارید ، تمام بازه های رجیستری و اعداد ثابت قابل استفاده در آن خانه نمایش داده می شود .

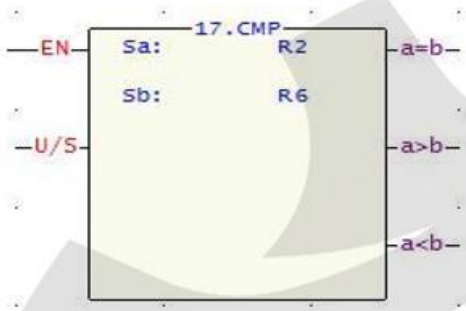


بعضی توابع مانند تابع شماره **25** ، دارای دو گزینه کاربردی می باشند :

32 Bits : هرگاه این گزینه فعال شود ، عملیات به روی دو رجیستر پشت سر هم صورت می گیرد .

Pulse : هرگاه این گزینه فعال شود ، تابع تنها لحظه ای اجرا می شود که ورودی کنترل تابع از **0** به یک تغییر کند (لبه بالارونده پالس)

اگر می خواهید که محتوای رجیسترها در کنار آنها در توابع دیده شوند (در حالت Online) ، از طریق منوی View ، کنار گزینه Register Content علامت تایید بزنید .



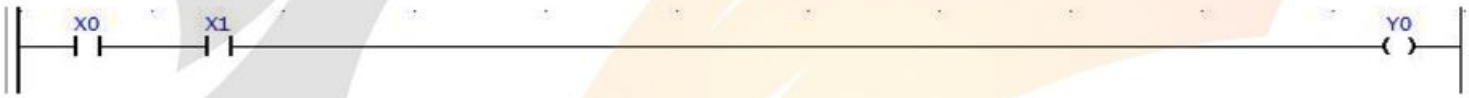
هر تابع براساس عملکرد آن می تواند دارای تعدادی پایه های ورودی و خروجی باشد .

کاربری های منطقی :

در بسیاری از کاربردهای کنترل انجام یک فرآیند تنها در صورت برقرار بودن منطق خاصی امکان پذیر است . در این قسمت با نحوه ایجاد کاربری های منطقی AND ، OR ، NOT ، NAND ، NOR و XOR توسط کنتاکت های باز و بسته ورودی آشنا خواهید شد .

کاربری AND :

در شکل زیر چگونگی ایجاد کاربری AND در یک دیاگرام نردبانی نشان داده شده است . در اینجا خروجی Y0 تنها وقتی فعال می شود که هر دو ورودی X0 و X1 وصل شده باشند .



کاربری OR :

در شکل زیر چگونگی ایجاد کاربری OR در یک دیاگرام نردبانی نشان داده شده است . در اینجا خروجی در صورتی فعال می شود که هر کدام از ورودی های X0 و X1 و یا هر دو وصل شوند .

کاربری NOT :

در شکل زیر چگونگی ایجاد کاربری NOT در یک دیاگرام نردبانی نشان داده شده است . در این وضعیت بوبین خروجی Y0 و ورودی X0 عکس یکدیگر می باشند .

کاربری NAND :

کاربری NOR :

کاربری XOR :

مدار خودنگهدار :

در مدارات فرمان عموما جهت روشن و خاموش کردن موتور از دو شستی استارت و استپ استفاده می شود . هنگامی که شستی استارت را فشار دهیم ، یک رله فعال می شود و از طریق کنتاکت خود نگهدار این رله ، مسیر وصل می ماند . بنابراین با برداشتن دست از روی شستی ، رله همچنان فعال می ماند و تنها راه غیرفعال کردن این رله ، فشار روی شستی استاپ است .

توجه کنید که در تصویر فوق هر دو شستی استارت و استاپ متصل به PLC در حالت عادی دارای کنتاکت باز می باشند .

رله های کمکی (رله های داخلی) :

در مدارهای فرمان عموما می توان رله هایی یافت که مستقیما خروجی را فعال نمی کنند . بلکه از کنتاکت های آن در جهت بوجود آوردن منطق مورد نظر در مدار استفاده می شود . داخل حافظه PLC نیز بیت هایی برای نگهداری اطلاعات وجود دارند که آن ها را رله های کمکی می نامند ، زیرا این رله ها مانند رله های الکترومکانیکی می توانند تحریک شوند و کنتاکت های آن ها پس از تحریک شدن رله ، تغییر وضعیت داده و منطق موردنظر را در برنامه PLC بوجود می آورند و نهایتا منجر به تحریک شدن یک خروجی می شوند . برخلاف رله های الکترومکانیکی که تعداد محدودی کنتاکت باز یا بسته دارند ، در رله های کمکی موجود در داخل PLC می توانیم به دفعات از این کنتاکت ها استفاده نمائیم . وجه تمایز رله های کمکی از یک رله خروجی در شماره آدرس استفاده شده آن در برنامه PLC می باشد . در دیاگرام نردبانی همانند رله های خروجی برای رله های کمکی نیز از نماد —(Y0)— استفاده می گردد . به عنوان مثال از M100 به همراه نماد —(Y0)— نشان دهنده این مطلب است که M100 یک رله کمکی می باشد . کنتاکت های این رله کمکی نیز با نماد —| Y0 |— بعنوان کنتاکت باز NO و —|/ Y0 |— بعنوان کنتاکت بسته NC نمایش داده می شوند و شماره M100 نیز جهت شناسایی این کنتاکت ها به کار می رود . (M از اول کلمه Memory گرفته شده است)

کاربرد رله های کمکی :

استفاده از رله های کمکی در اکثر برنامه ها ، حجم برنامه نویسی را کم می کند و امکان درک برنامه را تا حد زیادی افزایش می دهد . در شکل زیر هنگامیکه کنتاکت X2 بسته باشد ، با وصل حداقل یکی از ورودی های X1 یا X3 رله کمکی M1 تحریک می گردد . بنابراین کنتاکت M1 در خط دوم برنامه تغییر وضعیت داده و به شرط فعال شدن X4 رله خروجی Y1 تحریک می شود .

شکل زیر استفاده از دو رله کمکی جهت فعال کردن یک خروجی را نشان می دهد . در اولین خط برنامه رله کمکی M1 با بسته شدن حداقل یکی از ورودی های X1 یا X2 تحریک می شود و رله کمکی M2 نیز با بسته شدن ورودی های X3 و X4 فعال می گردد که نهایتا رله خروجی Y1 نیز در صورتی که حداقل یکی از رله های کمکی M1 یا M2 تحریک شده باشد ، فعال می گردد .

رله کمکی دارای باطری پشتیبان :

اگر در هنگام کار ، تغذیه ورودی PLC قطع شود ، رله های کمکی فعال ، غیرفعال می شوند . بنابراین با وصل شدن دوباره برق ، کنتاکت های مربوط به این رله ها در وضعیت صحیح قرار ندارند . برای رفع این مشکل تعدادی از رله های کمکی موجود در PLC به گونه ای قابل تنظیم هستند تا در صورت قطع برق با استفاده از تغذیه باتری پشتیبان موجود در PLC ، ارزش خود را حفظ نمایند . این رله ها را پایدار (Retentive) می نامند . برنامه نویس با استفاده از این رله ها می تواند در هنگام قطع و وصل برق از بروز حادثه جلوگیری نماید . در شکل زیر با وصل شدن X4 رله کمکی M300 فعال می شود . اگر در این لحظه برق سیستم قطع شود X4 غیرفعال می شود و در نتیجه کنتاکت X4 باز می شود و کنتاکت X5 بسته می ماند . در اینصورت چون رله M300 پایدار می باشد ، با وصل مجدد تغذیه ، از طریق خود نگهدارش فعال باقی می ماند .

ثبات یا رجیستر :

به طور کلی یک رجیستر به هر نوع حافظه الکترونیکی که بتواند اطلاعات را در خود ذخیره نماید ، اطلاق می گردد . رجیسترها در PLC از بهم پیوستن **16** رله که می توانند فعال یا غیرفعال باشند ، تشکیل می گردد . هر کدام از خانه های رجیستر را یک بیت می نامیم که در سیستم دودوئی ، هر بیت می تواند ارزش صفر به معنی غیرفعال بودن رله و یک به معنی فعال بودن رله را داشته باشد . حافظه در PLC از تعداد زیادی رجیستر تشکیل شده است که هر رجیستر می تواند یک عدد **16** بیتی را در خود ذخیره نماید . رجیسترها در FATEK ، با نماد R و D نمایش داده می شوند .

شیفت رجیستر (Shift Register) :

در خطوط تولید و تسمه نقاله ها عموما بعضی از محصولات معیوب می باشند ، بنابراین این محصولات باید تشخیص داده شوند و در زمان مناسب از خط تولید به خارج هدایت گردند . ردیابی این گونه محصولات معیوب در یک خط تولید از موارد عمده استفاده از شیفت رجیسترها در PLC می باشد . در این بخش با چگونگی استفاده از شیفت رجیسترها و کاربرد آن ها در صنعت آشنا خواهیم شد .

شیفت به معنای انتقال پیداکردن محتویات یک خانه به خانه بعدی می باشد . در شکل زیر ، یک تابع شیفت رجیستر (تابع شماره **6**) در داخل PLC دارای **5** ورودی و خروجی زیر می باشد .

1 – ورودی EN برای فعال سازی تابع

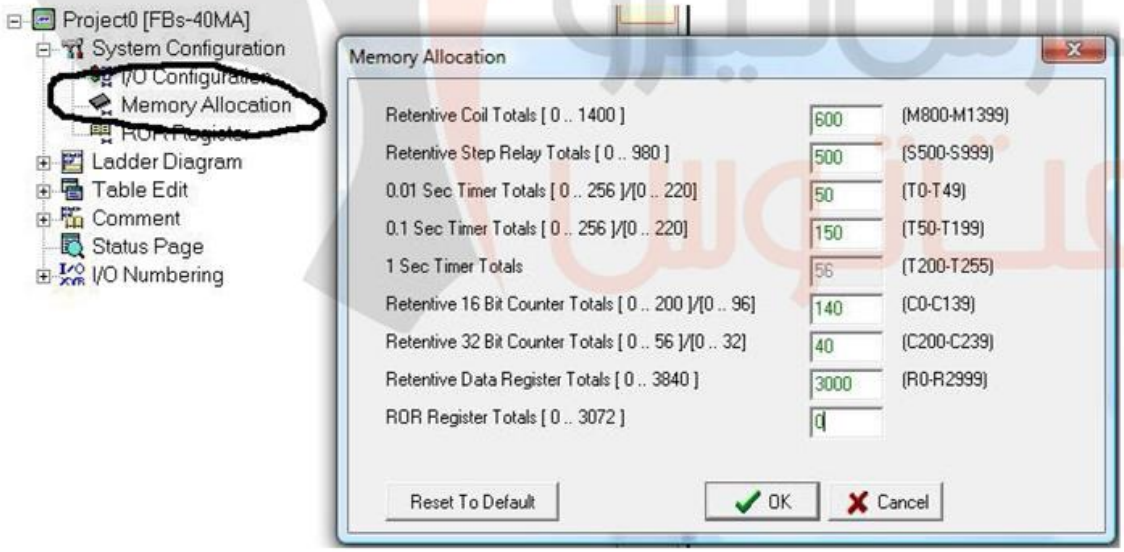
2 – ورودی INB جهت وارد کردن بیت جدید به اولین خانه استفاده می شود

3 – ورودی L/R هرگاه وصل باشد ، شیفت رجیستر به سمت چپ و اگر قطع باشد ، به سمت راست خواهد بود

4 – CLR جهت پاک نمودن محتویات کلیه خانه های رجیستر به کار می رود

5 – محتویات بیت آخر پس از شیفت پیدا کردن در خروجی OTB ظاهر می شود

همانند رله های کمکی پایدار ، گاهی به حفظ مقدار رجیسترها نیز پس از قطع برق نیاز است . به همین دلیل تعدادی از رجیسترها قابل تنظیم هستند که به عنوان رله پایدار استفاده شوند . برای مشاهده و تغییرات بر روی رجیسترهای پایدار به قسمت System configuration > Memory allocation مراجعه کنید .



دستور SET و RESET:

یکی از قابلیت های مهم برنامه نویسی PLC دستور SET و RESET می باشد . با استفاده از دستور SET یک رله فعال شده و تا زمانی که دستور RESET برای آن رله اجرا نشود ، رله فعال می ماند . در شکل زیر با فشار دادن شستی استارت X0 رله خروجی Y1 فعال می گردد و حتی با قطع X0 فعال باقی می ماند . با فشار دادن شستی استاپ X1 رله خروجی Y1 ، RESET شده و غیرفعال می گردد .

در این PLC همچنین بلوک های SET و RESET ، علاوه بر SET و RESET کردن بیت ها ، برای SET و RESET کردن رجیسترها ، تایمرها ، کانترها و ..... نیز استفاده می شود . به این معنی که با SET کردن یک رجیستر ، تمام بیت های آن رجیستر ، یک می شود . هنگام اسکن برنامه ، اگر هر دو بوبین SET و RESET فعال شوند ، دستوری که بعد از دیگری در برنامه قرار می گیرد ، اجرا می شود زیرا PLC در پایان هر اسکن ، نتایج حاصله را در خروجی قرار می دهد . یکی از کاربردهای دستور SET و RESET برقرارکردن اینترلاک (Interlock) می باشد .

اینترلاک ، قرار دادن یک یا چند شرط در مسیر فعال شدن یک خروجی به دلیل حفاظت از تجهیزات ، ایمنی افراد یا انجام صحیح یک فرآیند می باشد .

تایمر (زمان سنج) :

در بسیاری از موارد لازم است تا یک موتور برای مدت زمان معینی روشن بماند و یا بعد از مدت زمان معینی روشن گردد . از این رو تایمرها به عنوان یک قابلیت جهت اندازه گیری زمان در انواع مختلف PLC مورد استفاده قرار می گیرند . همچنین مقدار زمان اندازه گیری شده بصورت عدد در خانه های حافظه PLC ذخیره می شود و در هر لحظه قابل دستیابی می باشند . PLC های FATEK دارای **256** تایمر با سه دقت زمانی می باشد که بصورت زیر تنظیم شده اند :

T0 ~ T49 : 0.01s
T50 ~ T199 : 0.1s
T200 ~ T255 : 1s

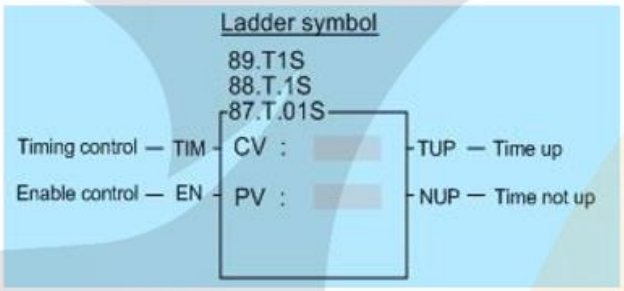
البته می توان این تنظیمات را به دلخواه تغییر داد . برای استفاده از تایمر ، گزینه T را از نوار المان ها انتخاب کرده و به روی محیط برنامه نویسی در محل مورد نظر کلیک کنید . پنجره شکل زیر ظاهر می شود . در گزینه T ، شماره تایمر با دقت دلخواه و در PV ، مقدار پیش فرض تنظیم زمان را وارد کرده و OK کنید .

نکته :

اگر M1957 = 0 ، مقدار جاری CV تا ماکزیمم مقدار رجیستر PV (32767) پیش می رود و اگر M1957 = 1 باشد ، CV بعد از رسیدن به مقدار PV ، دیگر اضافه نمی شود و بر روی همان مقدار ثابت می ماند .

در شکل زیر با فعال شدن ورودی M1 رله خروجی Y1 تایمر تحریک شده و بعد از زمان 5 ثانیه ، کنتاکت تایمر تغییر وضعیت می دهد و رله خروجی Y2 فعال می گردد . بنابراین خروجی Y2 پنج ثانیه پس از فعال شدن خروجی Y1 فعال می شود .

تایمرهایی که تاکنون گفته شد **16** بیتی اند بنابراین مقدار PV آنها نمی تواند بیش از **32767** باشد . برای زمان های تایمرهای طولانی تر ، از توابع **87** ، **88** ، **89** استفاده می شود که اگر تیک 32Bits آن زده شود ، PV آن می تواند تا **2147483647** اضافه شود .



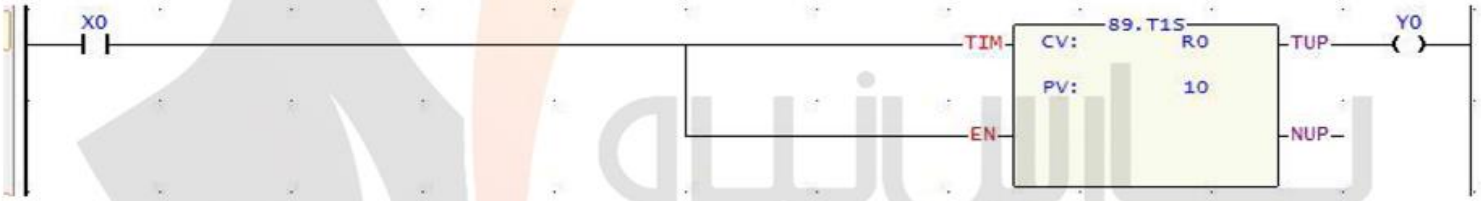
این تابع برخلاف تایمر ساده ، قابلیت نگه داشتن زمان را دارد . در این تابع وقتی ورودی TIM = 1 ، مانند تایمر ساده عمل می کند اما اگر TIM = 0 باشد ، زمان اندازه گیری شده پاک نمی شود . وقتی TIM مجددا یک شود ، محاسبه زمان از ادامه آخرین باری که نگه داشته شده است ، ادامه پیدا می کند .

اگر تایمر احتیاج به بازنشانی داشت باید EN را صفر کرد .

این تابع دو خروجی دارد :

TUP که بعد از اتمام محاسبه زمان یک می شود و NUP که وقتی TUP صفر است یک می شود .

می توان از ترکیب ورودی ها و خروجی ها برای ایجاد تایمرها با کارائی های مختلف استفاده کنید . بعنوان مثال در شکل زیر : وقتی X0 ، ON شود ، بعد از **10** ثانیه ، Y0 ، ON می شود .



شمارنده ها (Counters) :

شمارنده ها جهت مواردی نظیر اندازه گیری دور موتور و یا تعیین تعداد قوطی های کنسرو در هنگام بسته بندی به کار می روند و بدلیل کاربردهای فراوان در تمامی انواع PLC موجود می باشند . شمارنده های موجود در PLC نظیر شمارنده های موجود در مدارات فرمان روی یک تعداد اولیه ، تنظیم می شوند و هنگامی که شمارش به این تعداد تنظیمی رسید ، کنتاکت های شمارنده تغییر وضعیت می دهند . شمارنده های موجود در PLC ، افزایشی و یا کاهشی می باشند . در نوع افزایشی عمل شمارش از صفر شروع می شود و هر بار یک شماره به شماره قبلی اضافه می گردد تا به عدد تنظیمی برسد . ولی در نوع کاهشی شمارش از یک عدد تنظیمی شروع شده و با هر بار شمارش یک عدد از عدد قبلی کاسته می شود تا به صفر برسد . در FATEK ، شمارنده بصورت یک بلوک در نظر گرفته می شود که از طریق آیکون C در نوار المان ها قابل دسترس می باشد . این PLC دارای :

200 کانتر **16** بیتی (C0 ~ C199) با قابلیت شمارش تا **32767** و **56** کانتر **32** بیتی (C200 ~ C255) با قابلیت شمارش تا **2147483647** .

برای تعیین پایدار بودن شمارنده ها به مسیر زیر مراجعه کنید :

System Configuration > Memory Allocation

در C ، شماره کانتر مورد نظر وارد شده و PV جهت وارد کردن تعداد پیش تنظیم اولیه بکار می رود . ورودی PLS برای شمارش پالس ورودی و CLR جهت بازنشانی شمارنده به کار می روند و نهایتا خروجی CUP هرگاه که شمارش به مقدار PV برشد ، روشن می شود .

شمارش همزمان به صورت افزایشی و کاهشی :

شمارنده معمولی تنها به صورت افزایشی شمارش می کند . برای شمارش به صورت کاهشی از تابع شماره **7** که توانایی شمارش به صورت کاهشی و افزایشی را دارد ، استفاده می شود . از یک ورودی جهت شمارش افزایشی و کاهشی (U/D) و یک ورودی جهت ریست کردن شمارنده به حالت اولیه خود استفاده می شود (CLR) . هرگاه ورودی U/D فعال باشد ، شمارش افزایشی و هرگاه غیرفعال باشد ، کاهشی خواهد بود و نهایتا خروجی (CUP) هرگاه که شمارش به مقدار PV برسد ، روشن می شود . CV جهت وارد کردن تعداد اولیه و نمایش تعداد جاری و PV جهت وارد کردن مقدار نهائی استفاده می شود .

پردازش اعداد :

رله های کمکی تنها دارای دو حالت فعال یا غیرفعال می باشند ، به همین دلیل همگی یک خانه از حافظه PLC (یک بیت) را بصورت مجزا اشغال می کنند در حالیکه رجیسترها مجموعه ای از بیت های در کنار هم می باشند و همین امر سبب می شود که از رجیسترها جهت انجام محاسبات ، ذخیره و نمایش اعداد استفاده گردد . در سیستم های کنترل عموما لازم است تا دو عدد را با یکدیگر مقایسه کنیم یا با هم جمع نماییم . هنگامیکه یک عدد وارد PLC گردید ، PLC جهت پردازش روی این عدد نیازمند دستورالعمل های مناسب می باشد . دستورالعمل های انتقال اطلاعات ، دستورهای عملیات حسابی و مقایسه اعداد از جمله این موارد می باشد .

دستورالعمل انتقال اطلاعات :

دستور MOV (تابع شماره **8**) محتویات موجود در آدرس مبدأ حافظه را در آدرس مقصد می نویسد . (با حفظ مقدار آن در آدرس مبدأ)
این دستور جهت انتقال اطلاعات بین رجیسترهای حافظه ، ورودی ها و خروجی ها ، مقادیر ثابت تایمرها و شمارنده ها استفاده می شود . در شکل زیر هنگامی که ورودی وصل گردد ، دستورالعمل MOV اجرا شده و محتویات آدرس مبدأ در آدرس مقصد نیز نوشته می شود . به جای رجیستر مبدأ از عدد ثابت نیز می توان استفاده کرد .

دستورالعمل مقایسه اعداد :

هشت دستورالعمل جهت مقایسه اعداد در PLC وجود دارد که عبارتند از :

مقایسه کلی (تابع شماره **17**)

مقایسه ناحیه ای (تابع شماره **37**)

مساوی (تابع شماره **170**)

بزرگتر (تابع شماره **171**)

کوچکتر (تابع شماره **172**)

نامساوی (تابع شماره **173**)

بزرگتر یا مساوی (**174**)

کوچکتر یا مساوی (**175**)

در این نوع دستورالعمل دو مقدار با یکدیگر مقایسه می شوند و در صورت درست بودن شرط مقایسه ، خروجی فعال می گردد . در شکل زیر هنگامیکه زمان اندازه گیری شده توسط تایمر T4 از **400** بزرگتر شود ، بوبین خروجی تحریک می گردد .

دستورالعمل عملیات حسابی :

در این PLC علاوه بر چهار عمل اصلی ، توابع ریاضی نظیر جذر ، توابع نمایی و لگاریتمی و توابع مثلثاتی و غیره قابل محاسبه می باشند . در شکل زیر چگونگی انجام یک عمل ضرب نمایش داده شده است .

اعداد اعشاری :

در دسته بندی های مختلف توابع FATEK ، دسته ای برای اجرای عملیات به روی اعداد اعشاری یا Float وجود دارد . بنابراین اگر نیاز به اجرای عملیاتی مانند جمع ، ضرب یا مقایسه بین اعشاری بود ، از توابع جمع ، ضرب و مقایسه اعداد Float استفاده می شود . بعنوان مثال اگر بخواهیم یک عدد صحیح را به اعشاری تبدیل کنیم ، از تابع شماره **200** استفاده می کنیم .

پرش (Jump) و برچسب (Label) :

در صورتیکه تحت شرایط خاصی بخواهیم از روی تعدادی از خطوط برنامه بدون اجرا پرش نماید ، از دستور Jump استفاده می شود . پرش به خطی از برنامه صورت می گیرد که برچسب آن مشخص می کند و برنامه از جایی ادامه پیدا می کند که آن برچسب زده شده است . پرش فقط در برنامه اصلی یا زیربرنامه ها صورت می گیرد و با این فانکشن از برنامه اصلی به زیر برنامه و یا برعکس نمی توان پرش کرد . در برنامه نویسی PLC انجام یک پرش در داخل پرش دیگر مجاز می باشد . همچنین دستور پرش به سمت بالای برنامه بایستی با دقت مورد استفاده قرار گیرد زیرا ممکن است ، استفاده غیرصحیح از این دستور منجر به طولانی شدن زمان یک اسکن و در نتیجه خطای Watch Dog گردد .

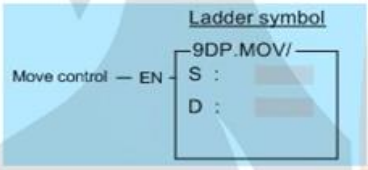
فراخوانی زیر برنامه CALL :

گاهی احتیاج است که یک قسمت از برنامه تنها هنگامی که به آن نیاز است ، اجرا شود . برای این منظور این قسمت از برنامه را در زیر برنامه می نویسیم . برای ایجاد زیربرنامه در درخت پروژه به قسمت Ladder Diagram رفته و به روی Sub Program راست کلیک کرده و New Unit را انتخاب می کنیم . در پنجره ای که باز می شود ، نامی برای زیر برنامه جدید می نویسیم و پس از OK کردن ، زیر برنامه جدید ساخته می شود . جابجایی محیط برنامه نویسی از محیط اصلی به محیط زیر برنامه از طریق تب های پایین محیط برنامه نویسی صورت می گیرد .

حال برای اجرای زیربرنامه ، برچسبی در ابتدای زیر برنامه گذاشته و در انتهای آن نیز تابع شماره **68** (RTS) را می گذاریم .

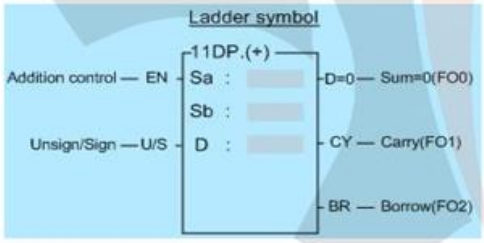
برای اجرا شدن این زیر برنامه ، باید برچسب آن از طریق تابع CALL فراخوانی شود . وقتی برنامه اصلی از طریق CALL یک زیر برنامه را فراخوانی می کند ، آن زیر برنامه نیز می تواند زیر برنامه های دیگر را فراخوانی کند و این کار تا **5** مرحله می تواند انجام بپذیرد .

تابع شماره 9 : Move Inverse



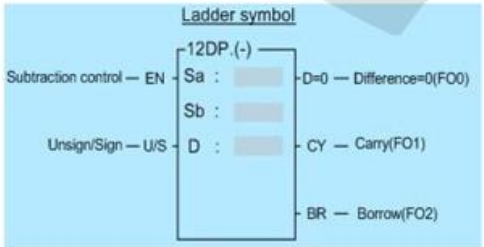
با فعال شدن EN ، اطلاعات موجود در S عکس شده و به D منتقل می شود .
(**0** ها به **1** و **1** ها به **0** تبدیل می شوند)

تابع شماره 11 : Addition

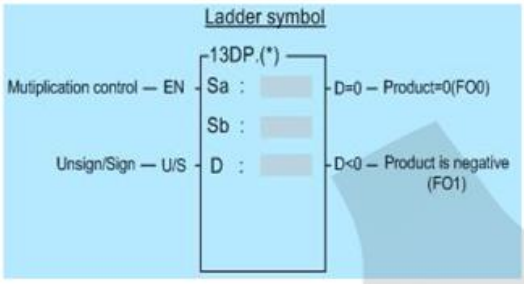


با فعال شدن EN ، داده های موجود در Sa و Sb با هم جمع شده و در D ذخیره می شود .

تابع شماره 12 : Subtraction

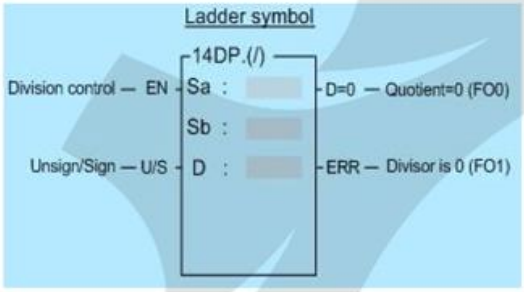


با فعال شدن EN ، داده های موجود در Sa منهای Sb شده و نتیجه در D ذخیره می شود .



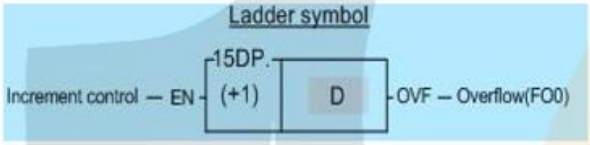
تابع شماره 13 : Multipilication

با فعال شدن EN ، داده های موجود در Sa ضرب در Sb می شود و نتیجه در D ذخیره می شود .



تابع شماره 14 : Division

با فعال شدن EN ، داده های موجود در Sa ، تقسیم بر Sb شده و نتیجه در D ذخیره می شود . هرگاه Sb صفر باشد ، تابع اجرا نمی شود و Error می دهد .

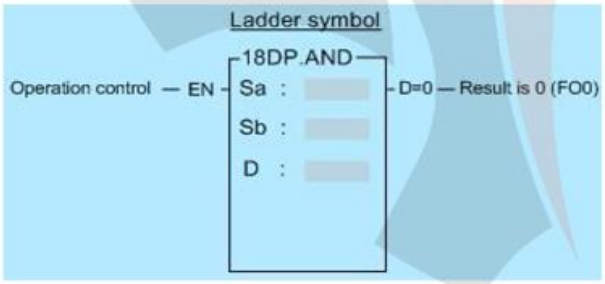


تابع شماره 15 : Increment

هرگاه EN از صفر به یک تغییر کند ، به مقدار رجیستر D ، یک واحد اضافه می شود . اگر این افزایش باعث از محدوده خارج شدن مقدار D شود ، OVF فعال می شود و مقدار D منفی می شود .

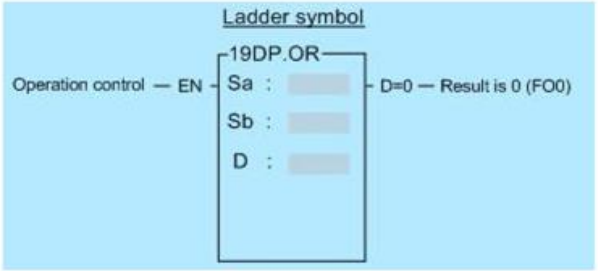
تابع شماره 16 : Decrement

هرگاه EN از صفر به یک تغییر کند ، از مقدار رجیستر D ، یک واحد کم می شود . اگر این کاهش باعث از محدوده خارج شدن مقدار D شود ، UDF فعال می شود و مقدار D مثبت می شود .



تابع شماره 18 : Logical AND

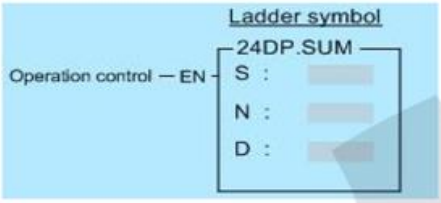
هرگاه EN از صفر به یک تغییر کند ، بیت های موجود در Sa و Sb را با هم AND کرده و نتیجه را در D می ریزد .



تابع شماره 19 : Logical OR

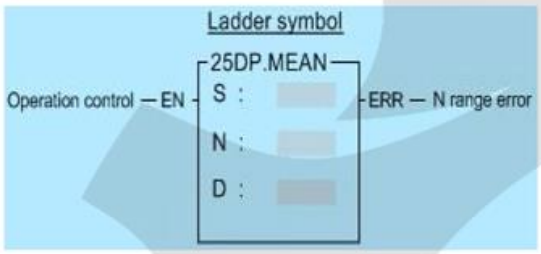
هرگاه EN از صفر به یک تغییر کند ، بیت های موجود در Sa و Sb را با هم OR کرده و نتیجه را در D می ریزد .

تابع شماره 24 : Sum



با این تابع می توان مجموع چند رجیستر متوالی را به یک رجیستر دیگر منتقل کرد . در S اولین رجیستر قرار داده می شود . در N تعداد رجیسترهای متوالی تعیین می شود و مجموع این N تعداد رجیستر در D ریخته می شود .

تابع شماره 25 : Mean



این تابع برای میانگین گرفتن از مقادیر چند رجیستر متوالی کاربرد دارد . رجیستر شروع در S قرار می گیرد و تعداد رجیسترها در N . هرگاه EN از صفر به یک تغییر کند ، مقادیر موجود در رجیسترها با هم جمع شده و تقسیم بر تعداد آنها شده و نتیجه در D ریخته می شود . اگر مقدار بین **2** تا **256** نباشد ، ERR فعال شده و تابع اجرا نمی شود .

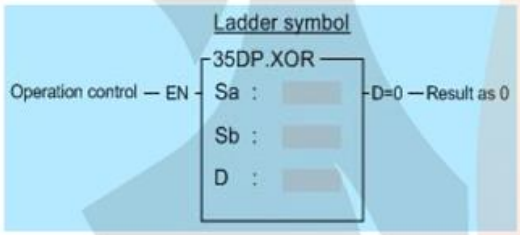
تابع شماره 26 : Square Root

این تابع جذر عدد موجود در S را گرفته و نتیجه را بدون در نظر گرفتن اعشار آن در D ذخیره می کند . اگر مقدار S منفی باشد ، ERR فعال شده و تابع اجرا می شود .

تابع شماره 28 : Absolute

این تابع قدر مطلق مقدار موجود در D را دوباره در D می ریزد .

تابع شماره 35 : Exclusive OR

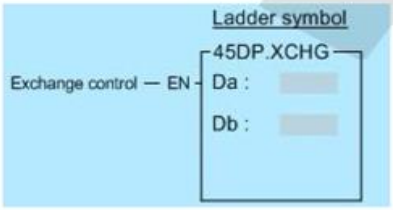


هرگاه EN از صفر به یک تغییر کند ، بیت های موجود در Sa و Sb را با هم XOR کرده و نتیجه را در D می ریزد . عملکرد به این صورت است که هرگاه بیت های متناظر Sa و Sb همانند بودند ، بیت متناظر در D صفر می شود و هرگاه یکی از بیت ها **1** و دیگری **0** بود ، نتیجه در D، یک می شود .

تابع شماره 36 : Exclusive NOR

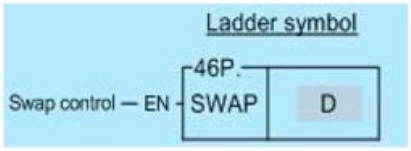
عملکرد این تابع برعکس تابع قبل است یعنی دو بیت همانند در Sa و Sb ، متناظر با یک در D و دو بیت ناهمانند در Sa و Sb متناظر با صفر در D می باشد .

تابع شماره 45 : Exchange



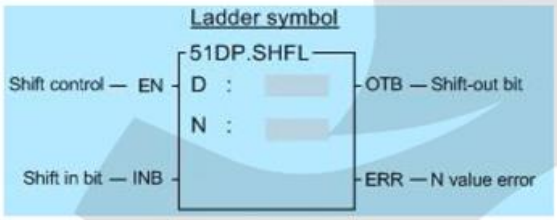
هرگاه EN از صفر به یک تغییر کند ، مقادیر رجیستر Da و Db با هم عوض می شوند .

تابع شماره 46 : Byte Swap



در D یک رجیستر **16** بیتی قرار می گیرد که با فعال شدن EN ، بایت بالا و پایین آن با هم عوض می شوند .

تابع شماره 51 : Shift Left



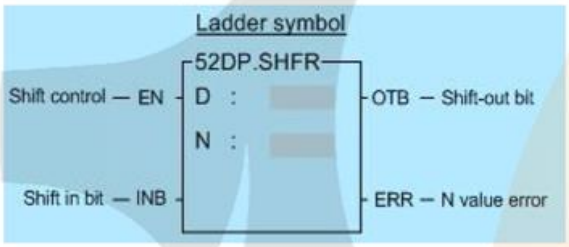
در D رجیستری که قرار است شیفت داده شود ، قرار می گیرد . N تعداد شیفت به چپ را مشخص می کند . جای بیت های شیفت داده شده را مقدار INB پر می کند و N اُمین بیت بالای رجیستر به خروجی OTB می رود .

اگر رجیستر موردنظر **16** بیتی باشد : N : 1~16

اگر رجیستر موردنظر **32** بیتی باشد : N : 1~32

در غیر اینصورت Error می دهد .

تابع شماره 52 : Shift Right



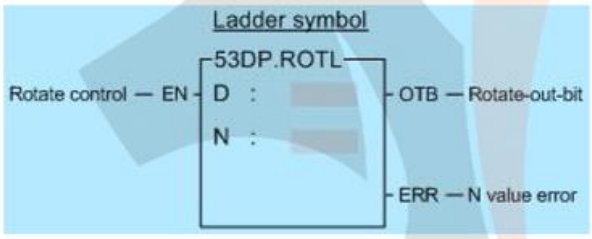
در D رجیستری که قرار است شیفت داده شود ، قرار می گیرد . N تعداد شیفت به راست را مشخص می کند . جای بیت های شیفت داده شده را مقدار INB پر می کند و N اُمین بیت پایین رجیستر به خروجی OTB می رود .

اگر رجیستر موردنظر **16** بیتی باشد : N : 1~16

اگر رجیستر موردنظر **32** بیتی باشد : N : 1~32

در غیر اینصورت Error می دهد .

تابع شماره 53 : Rotate Left



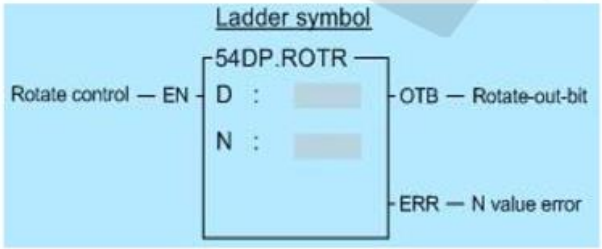
با این تابع چرخش به سمت چپ در محل بیت ها انجام می گیرد و چپ ترین بیت به راست ترین بیت منتقل می شود و یکی کپی از آن به خروجی OTB می رود .

اگر رجیستر موردنظر **16** بیتی باشد : N: 1~16

اگر رجیستر موردنظر **32** بیتی باشد : N: 1~32

در غیر اینصورت Error می دهد .

تابع شماره 54 : Rotate Right



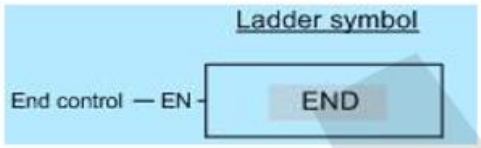
با این تابع یک چرخش به سمت راست در محل بیت ها انجام می گیرد و راست ترین بیت به چپ ترین بیت منتقل می شود و یک کپی از آن به خروجی OTB می رود .

اگر رجیستر موردنظر **16** بیتی باشد : N:1~16

اگر رجیستر موردنظر **32** بیتی باشد : N: 1~32

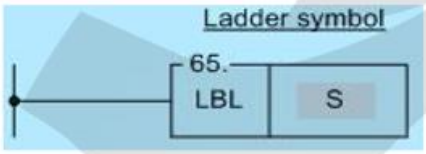
در غیر اینصورت Error می دهد .

تابع Program End :



هرگاه EN از صفر به یک تغییر کند این تابع فعال شده و به ابتدای برنامه می رود . تمام برنامه ها بعد از تابع END دیگر اجرا نمی شوند . وقتی EN=0 ، این تابع نادیده گرفته می شود و برنامه های بعد از این تابع اجرا نخواهند شد . به کار بردن END در برنامه اصلی ضرورتی ندارد زیرا CPU بصورت خودکار وقتی به انتهای برنامه رسید ، به نقطه شروع می رود .

تابع شماره 65 : Lable



این تابع به تنهایی عملی را انجام نمی دهد و بعنوان یک برچسب آدرس برای برنامه ها عمل می کند . بعنوان یک نشانگر برای اجرای تابع های Jump و Call و Interrupt استفاده می شود . همچنین برای آسان خوانی برنامه می توان به کار برد . در S نام برچسب متشکل از حروف و اعداد نوشته می شود که از **1** تا **6** حرف می تواند داشته باشد . اسامی موجود در جدول زیر برای تابع های Interrupt به کار برده می شود ، از آنها به عنوان Lable برنامه های متفرقه استفاده نکنید .

Reserved words	Description
X0+I～X15+I（INT0～INT15） X0–I～X15–I（INT0–～INT15–）	labels for external input (X0～X15) interrupt service routine.
HSC0I～HSC7I	labels for high speed counter HSC0～HSC7 interrupt service routine.
1MSI（1MS）、2MSI（2MS）、3MSI（3MS）、 4MSI（4MS）、5MSI（5MS）、10MSI（10MS）、 50MSI（50MS）、100MSI（100MS）	Labels for 8 kinds of internal timer interrupt service routine.
HSTAI（ATMRI）	Label for High speed fixed timer interrupt service routine.
PSO0I～PSO3I	Labels for the pulse output command finished interrupt service routine.

تابع شماره 69 : RTI

این تابع شبیه تابع RTS است با این فرق که RTS در انتهای اجرای زیر برنامه قرار می گیرد ، در حالی که RTI در انتهای روتین Interrupt قرار می گیرد . در مقایسه با Call که از طریق یک Lable ، زیر برنامه موردنظر را اجرا می کند ، Interrupt مستقیما از طریق سیگنال های سخت افزاری فعال شده و در کار CPU وقفه ایجاد می کند تا به انجام روتین Interrupt بپردازد . اگر وقفه ای در حین اجرای روتین یک وقفه دیگر اتفاق بیافتد ، روتین وقفه ای اجرا خواهد شد که در اولویت بالاتر قرار دارد . توجه داشته باشید که حتما RTI در انتهای روتین وقفه استفاده کنید .

تابع شماره 70 : FOR

این تابع به همراه تابع NEXT تشکیل یک حلقه می دهد که برنامه میان FOR و NEXT مربوطه ، N بار اجرا می شود . حلقه های FOR و NEXT دیگری نیز در میان حلقه های FOR و NEXT اولیه می تواند قرار بگیرد .

تابع شماره 71 : Loop END

این تابع به همراه تابع FOR ، تشکیل یک حلقه را می دهد . اگر قبل از این تابع ، تابع FOR وجود نداشته باشد ، این تابع نادیده گرفته می شود .

تابع شماره 145 : Enable Of Interrupt

هرگاه EN از صفر به یک تغییر کند این تابع برنامه وقفه را اجرا می کند که برچسب آن در دستور نام برده شده است . Lable های مختلف در جدول زیر آورده شده است .

LBL name	Description	LBL name	Description	LBL name	Description
HSTAI	HSTA High speed counter interrupt	X4+I	X4 positive edge interrupt	X10+I	X10 positive edge interrupt
HSC0I	HSC0 High speed counter interrupt	X4–I	X5 negative edge interrupt	X10–I	X10 negative edge interrupt
HSC1I	HSC1 High speed counter interrupt	X5+I	X5 positive edge interrupt	X11+I	X11 positive edge interrupt
HSC2I	HSC2 High speed counter interrupt	X5–I	X5 negative edge interrupt	X11–I	X11 negative edge interrupt
HSC3I	HSC3 High speed counter interrupt	X6+I	X6 positive edge interrupt	X12+I	X12 positive edge interrupt
X0+I	X0 positive edge interrupt	X6–I	X6 negative edge interrupt	X12–I	X12 negative edge interrupt
X0–I	X0 negative edge interrupt	X7+I	X7 positive edge interrupt	X13+I	X13 positive edge interrupt
X1+I	X1 positive edge interrupt	X7–I	X7 negative edge interrupt	X13–I	X13 negative edge interrupt
X1–I	X1 negative edge interrupt	X8+I	X8 positive edge interrupt	X14+I	X14 positive edge interrupt
X2+I	X2 positive edge interrupt	X8–I	X8 negative edge interrupt	X14–I	X14 negative edge interrupt
X2–I	X2 negative edge interrupt	X9+I	X9 positive edge interrupt	X15+I	X15 positive edge interrupt
X3+I	X3 positive edge interrupt	X9–I	X9 negative edge interrupt	X15–I	X15 negative edge interrupt
X3–I	X3 negative edge interrupt				

تابع شماره 146 : Disable Of Interrupt

این تابع برنامه وقفه را که Lable آن در تابع نام برده شده است ، غیرفعال می کند .

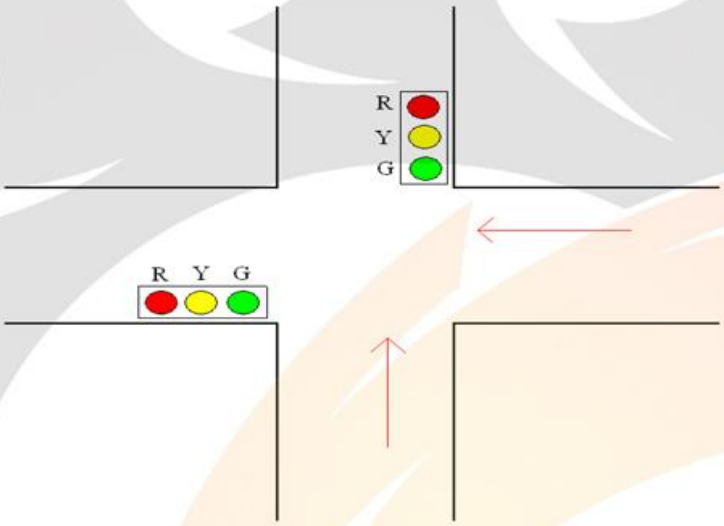
- شایان ذکر است در این جزوه آموزشی فقط توابع برنامه نویسی مهم ذکر شده است و از بیان توابع پیچیده که کاربرد عمومی ندارد ، اجتناب شده است . خواهشمند است جهت کسب اطلاعات جامع از توابع مختلف برنامه نویسی Fatek ، به فهرست منابع مراجعه نمایید .

فصل پنجم

تمرین 1 :

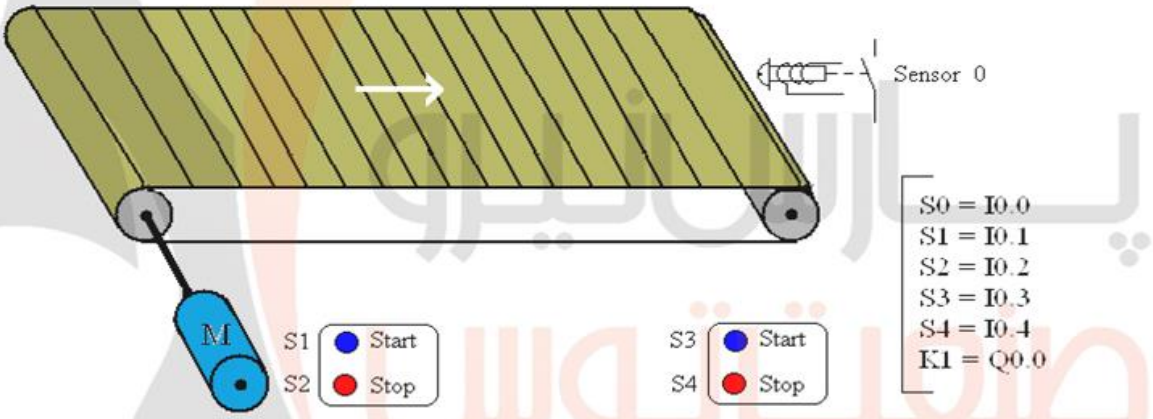
در یک چهارراه سیستم چراغ راهنمایی به صورت زیر است :

مدت زمان چراغ قرمز **30** ثانیه و مدت زمان چراغ زرد **5** ثانیه و مدت زمان چراغ سبز **25** ثانیه می باشد . سیستم کنترل آن را طراحی کنید .



تمرین 2 :

در شکل زیر دو کلید فشاری S1 و S2 به ترتیب برای استارت و استپ در سمت آغازین کانوایر وجود دارد ، همچنین در بخش انتهایی کانوایر دوکلید فشاری S3 و S4 برای استارت و استپ کانوایر تعبیه شده است . از طریق هر دو بخش آغازین و انتهایی کانوایر می توان آن را استپ یا استارت نمود . لازم به ذکر است که سنسور S0 برای توقف کانوایر هنگام رسیدن جسم به انتهای کانوایر نصب شده است . برنامه کنترلی این کانوایر را بنویسید .



تمرین 3 :

سیستم کنترل میز مسابقه سه نفره ای را به گونه ای طراحی کنید که اگر هر کدام از شاسی های S1، S2، S3 را که زودتر فشار داده شود ، چراغ مربوط به آن روشن شده و چراغهای دیگر عمل نکند .

تمرین 4 :

سیستمی را طراحی نمایید که با فشار دادن شاسی S1 کنتاکتور K1 مگنت کرده و در حالت مگنت باقی بماند ، زمانی که برای بار دوم شاسی S1 را فشار دادیم کنتاکتور K1 قطع شود .

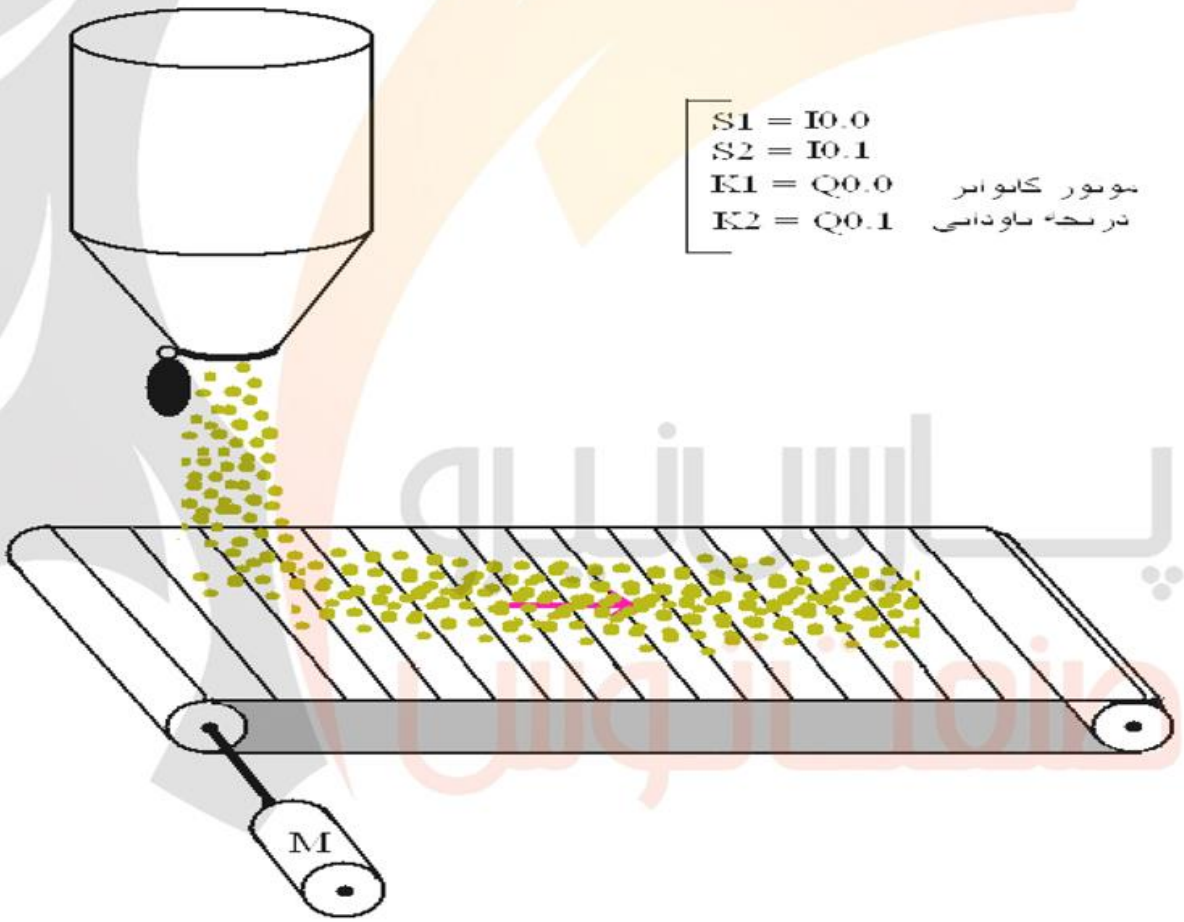
تمرین 5 :

در یک ماشین ابزار سیستم کنترل به صورت زیر است :

با فشار دادن شاسی S1 موتور بطور دائم کار می کند و با فشار دادن شاسی S0 موتور خاموش می گردد . با فشار دادن شاسی S2 موتور بطور لحظه ای کار می کند و هنگامی که شاسی S2 رها شود موتور خاموش می گردد . (سیستم کنترل لحظه ای و دائم)

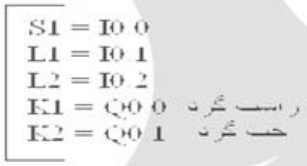
تمرین 6 :

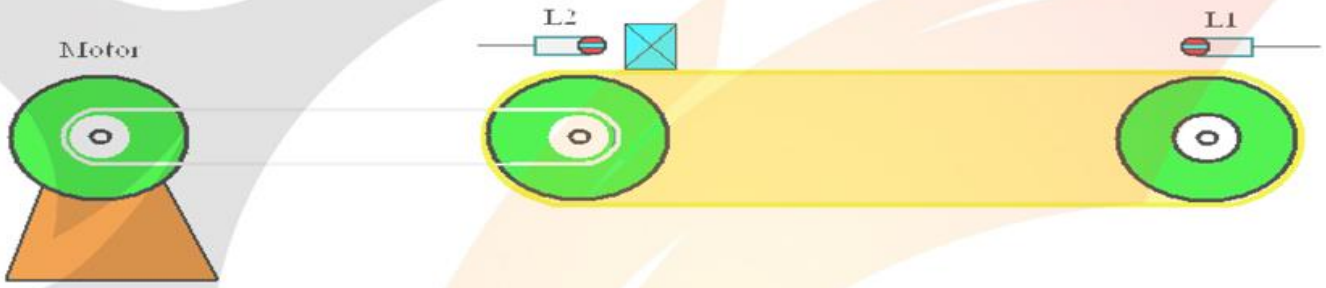
شکل زیر یک انتقال دهنده خرده سنگ از مخزن بر روی کانوایر می باشد . یک دریچه در قسمت ناودانی مخزن خرده سنگ وجود دارد و از آنجا خرده سنگ بر روی کانوایر ریخته می شود ، یک موتور حرکت کانوایر را کنترل می کند . اگر مکانیزم موتور کانوایر متوقف شود و یا مکانیزم عمل معیوب گردد ، دریچه ناودانی مخزن باید بسته شود . وقتی که دریچه مخزن انرژی می گیرد باز می گردد و با قطع انرژی آن بسته می شود شاسی فشاری S1 باعث خاموش شدن سیستم و شاسی S2 باعث روشن شدن سیستم می گردد .



تمرین 7 :

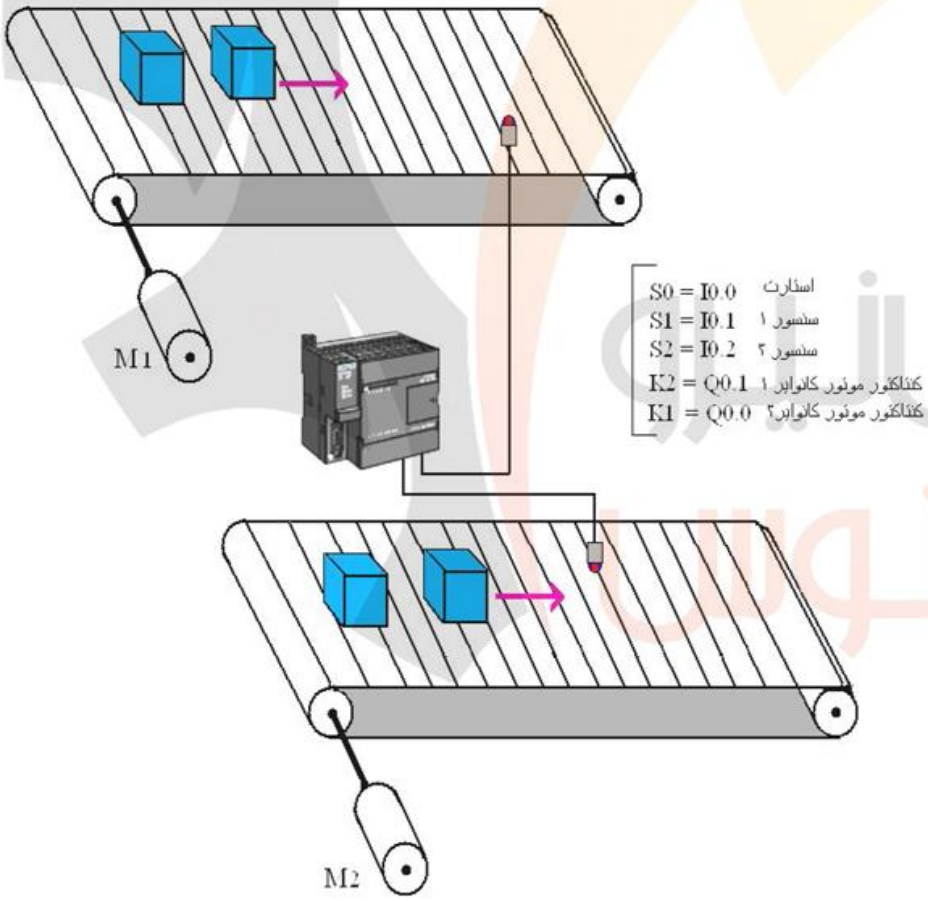
قطعه کاری بر روی یک کانوایر قرار دارد که می تواند در محدوده دو سنسور نوری L1 و L2 حرکت کند . وقتی که شاسی استارت فشار داده می شود ، در صورتی که سنسور L2 توسط قطعه کار تحریک شده باشد کانوایر توسط موتور به سمت جلو حرکت می کند . با حرکت کانوایر و برخورد آن به سنسور L1 کانوایر توسط موتور تغییر جهت می دهد .





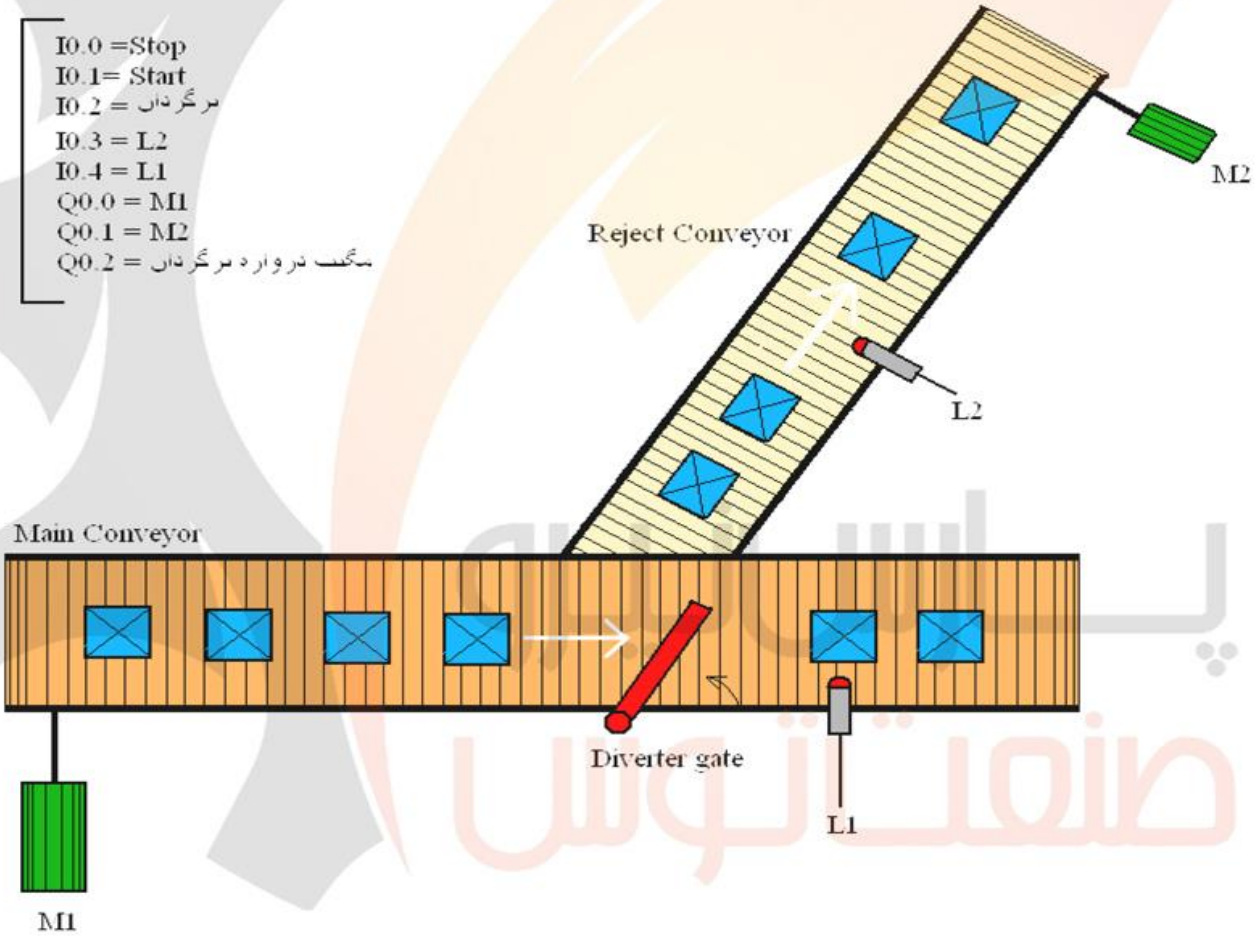
تمرین 8 :

در شکل زیر دو کانوایر که در ساخت یک فرآیند نقش دارند نشان داده شده است . هر کانوایر دارای یک سنسور می باشد که قطعه کارهای عبوری از کنار آن را شمارش می کند . برنامه هایی بنویسید که قطعه کارهای عبوری از هر یک از دو کانوایر را بطور اختصاصی شمارش نماید . زمانی که قطعات کانوایر **1** از **100** عدد گذشت کانوایر **1** خاموش شود و زمانی که سنسور کانوایر **2** از **200** عدد گذشت کانوایر **2** نیز خاموش شود . با زدن مجدد شاسی استارت مراحل از اول آغاز گردد .



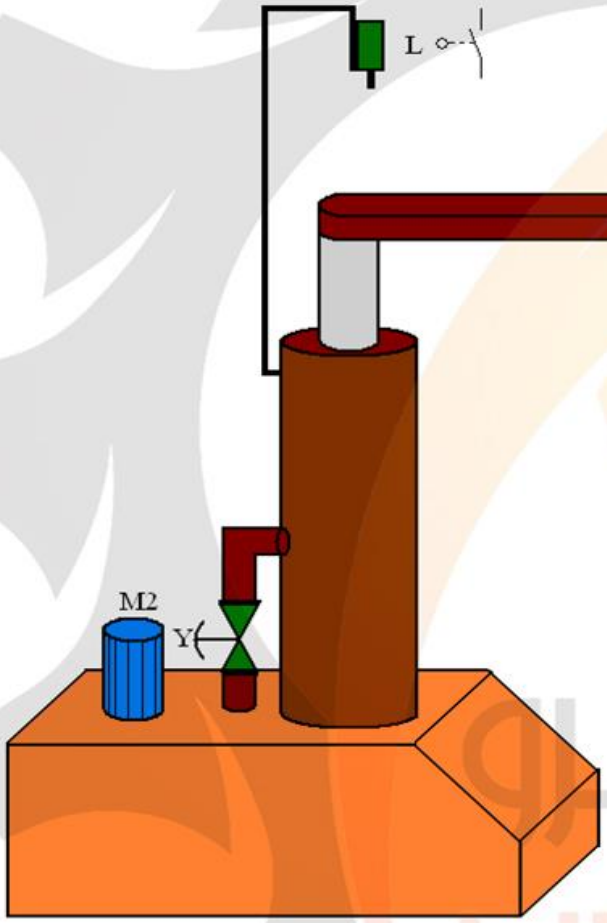
تمرین 9 :

در شکل زیر یک کانوایر اصلی (Main Conveyor) با یک دروازه برگردان (Diverter gate) برای قسمتهای معیوب به داخل کانوایر برگشتی (Reject Conveyor) را نشان می دهد . اگر در بازرسی یک قطعه کار معیوب باشد ، دروازه برگردان توسط اپراتور فعال شده و قطعات به داخل کانوایر برگشتی انتقال پیدا می کند . یک سنسور در کنار کانوایر اصلی و یک سنسور در کنار کانوایر برگشتی وجود دارد و قطعات عبوری را شمارش می کند . برنامه بنویسید که اگر قطعات عبوری از کانوایر برگشتی از **10** عدد و قطعات عبوری از کانوایر اصلی از **50** عدد گذشت سیستم کانوایر متوقف شود (سیستم عملکرد برگردان به این صورت است که با برق دار شدن برگردان عمل کرده و با قطع برق به حالت عادی باز می گردد)

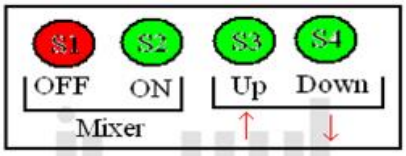


تمرین 10 :

شکل زیر میکسر یک کارخانه صنایع رنگ سازی می باشد سیستم عملکرد میکسر به این صورت است که با زدن شاسی S2 موتور اصلی میکسر (M1) شروع به کار می کند . جهت بالا و پایین کردن پروانه میکسر از یک پمپ هیدرولیک استفاده شده است با روشن شدن موتور M2 پمپ هیدرولیک عمل کرده و پروانه میکسر تا محدوده میکروسوییچ L بالا رفته و با تحریک شدن میکروسوییچ L متوقف می شود . جهت پایین آوردن میکسر شیر برقی Y تعبیه شده که روغن را وارد مخزن اصلی می کند . سیستم کنترل آن را برنامه نویسی نمایید (در صورتی که موتور میکسر دچاراضافه بار گردید سیستم خاموش شده و چراغ نارنجی رنگ چشمک بزند)



Q0.0 = M1 = Motor Mixer
Q0.1 = M2 = Motor Oil Pump
Q0.2 = Y = Valve Coil
Q0.3 = Lamp Green
Q0.4 = Lamp Red
Q0.5 = Lamp Yellow
S1 = I0.0 Stop M1
S2 = I0.1 Start M1
S3 = I0.2 Start M2
S4 = I0.4 Y Valve
L = I0.3 Limit Switch
F5 = I0.7 بیمتال



پانل کنترل

تمرین 11 :

یک شمارنده ماشین برای شمارش تعداد ماشین وارد شده و خارج شده از یک پارکینگ با ظرفیت **25** اتوموبیل مورد نیاز است :

الف : یک ورودی تعداد اتوموبیل های وارد شده و ورودی دیگر تعداد اتوموبیل های خارج شده را می شمارد .

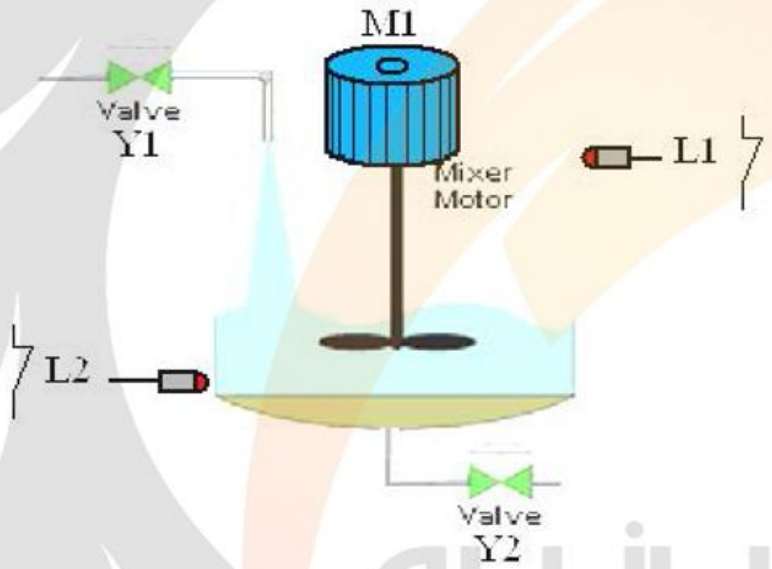
ب : وقتی تعداد اتوموبیل های داخل پارکینگ به **25** عدد رسید ، خروجی کانتر با نمایش عبارت ``Full`` پربودن پارکینگ را مشخص نماید .

ج : وقتی تعداد اتوموبیل های داخل پارکینگ کمتر از **25** عدد بود ، عبارت ``Vacancy`` بمعنی ظرفیت داشتن پارکینگ روشن شود .

د : یک کلید ورودی می تواند توسط متصدی پارکینگ ، پارکینگ را در موقعیت بسته نگهدارد .

تمرین 12 :

شکل زیر یک میکسر را نشان می دهد وقتی که شاسی استارت فشار داده شود سلنوئید Y1 فعال شده و مایع می تواند وارد مخزن شود . سنسورهای L1 و L2 سطح بالا و پایین مایع مخزن را مشخص می کنند و هر دو دارای کنتاکت NC می باشند (وقتی که مخزن خالی است L1 و L2 بسته هستند) زمانی که مخزن پر شد سنسور L1 سلنوئید Y1 را قطع و فرمان شروع به کار موتور میکسر را صادر می کند . موتور میکسر برای **30** ثانیه فعال بوده و سپس خاموش می گردد . وقتی که موتور خاموش شد سلنوئید Y2 فعال شده و مایع مخزن را تخلیه می کند پس از خالی شدن مخزن سنسور L2 به حالت عادی برگشته (بسته شده) و سلنوئید Y2 قطع می گردد . برنامه کنترل آن را بنویسید .



$$\begin{bmatrix} Y1 = Q0.0 \\ Y2 = Q0.1 \\ L1 = I0.1 \\ L2 = I0.2 \\ S1 = I0.0 \\ M1 = Q0.2 \end{bmatrix}$$

تمرین 13 :

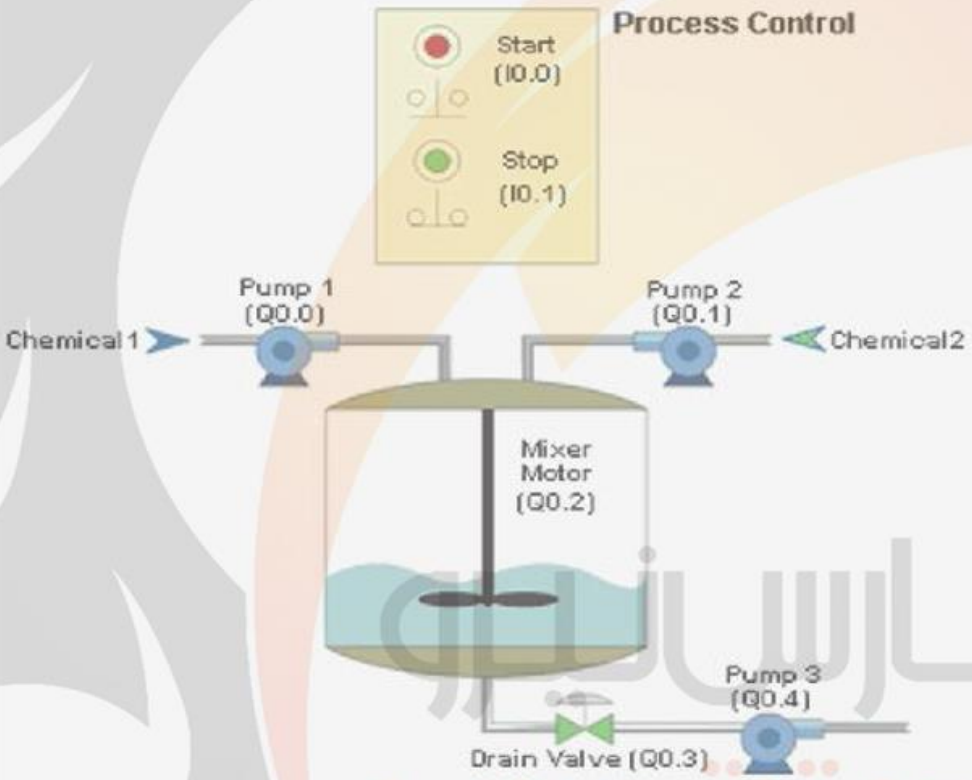
برنامه ای بنویسید مربوط به روشن و خاموش کردن چراغ راه پله های یک ساختمان بطوری که با وارد شدن به طبقه اول ساختمان با فشار دادن کلید چراغ طبقه اول روشن شود و در ابتدای ورودی طبقه دوم با فشار دادن کلید چراغ طبقه دوم روشن و چراغ طبقه اول خاموش شود و این روند تا طبقه آخر ادامه داشته و عکس آن از بالا به پایین نیز صادق باشد . همچنین در هر یک از طبقات در صورت انصراف بتوان برگشت . (سه طبقه)

تمرین 14 :

تایمری طراحی کنید که هم تاخیر در وصل باشد و هم تاخیر در قطع .

تمرین 15 :

شکل زیر یک مخزن به همراه میکسر مواد شیمیایی می باشد . سیستم عملکرد آن به این صورت است که با فشار دادن شاسی استارت ، پمپ یک شروع به کار کرده و یک مایع شیمیایی را وارد مخزن می کند پس از 20 ثانیه از شروع کار پمپ یک ، پمپ **2** نیز روشن شده و یک مایع شیمیایی دیگر را وارد مخزن می کند . پس از **10** ثانیه هر دو پمپ خاموش شده و موتور میکسر به مدت **15** ثانیه مواد شیمیایی را میکس می نماید سپس شیر خروجی باز شده و پمپ **3** به مدت **30** ثانیه روشن شده و مواد مخزن را تخلیه می کند برنامه آن را بنویسید .



تمرین 16 :

میخواهیم دو عدد موتور را بشرح زیر راه اندازی کنیم :

الف : با زدن کلید استارت ، موتور M1 شروع بکار نموده و بمدت **60** ثانیه روشن و سپس خاموش می شود .

ب : موتور M2 ، **15** ثانیه بعد از موتور M1 روشن شده و همراه با M1 خاموش شود .

پارس نیرو صنعت توس

www.parsnst.ir