

آموزش

PLC FATEK



PowerEn.ir



کنترل کننده های منطقی برنامه پذیر PLC

کنترل به بیان ساده عبارت است از هدایت یک فرایند در جهت رسیدن به نقطه مورد نظر. امروزه صنایع به منظور تولید بهینه و حفظ ایمنی نیازمند سیستمهای کنترل می باشند. در این فصل با ساختمان و اساس کار کنترل کننده های منطقی برنامه پذیر آشنا خواهید شد.

۱-۱) سیستم های کنترل

یک سیستم کنترل می تواند به سادگی ثابت نگه داشتن دمای یک خانه در 20°C و یا به پیچیدگی هدایت خط تولید در یک کارخانه اتومبیل سازی باشد. با این وجود هر سیستم کنترل اساساً از سه جزء اصلی تشکیل شده است:

۳-خروجی ها

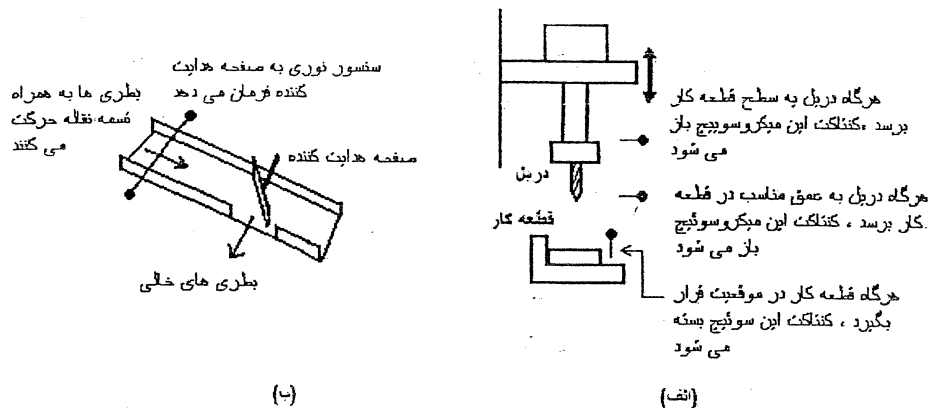
۲-منطق کنترل

۱-ورودی ها

شکل (الف-۱-۱) سیستم کنترل یک مته برقی را نشان می دهد. هنگامی که قطعه کار در مکان صحیح قرار گرفت، اهرم مربوطه، مته برقی را پایین می برد تا به سطح قطعه کار برسد، در اینجا موتور دریل روشن شده و مته شروع به سوراخ کردن قطعه می کند تا اینکه عمق لازم روع، قطعه کار ایجاد شده. سپس، موتور متوقف گشته و اهرم باعث می شود که مته به جای اولیه ی خود بازگردد.

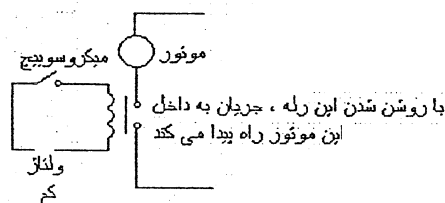
در این سیستم کنترل، ورودی ها، میکروسوییچ هایی هستند که وجود قطعه کار و رسیدن مته به سطح و عمق مناسب را نشان می دهند، و موتور مته و اهرم نیز خروجی ها می باشند، نحوه ی انجام عملیات نیز منطق کنترل را بیان می کند. همان طور که مشاهده می نمایید هر ورودی می تواند به صورت قطع یا وصل و هر خروجی نیز قادر است فعال یا غیرفعال باشد.

شکل (ب-۱-۱) سیستم کنترل یک تسمه نقاله را نشان می دهد. در اینجا بطری های نوشابه به سمت دستگاه بسته بندی هدایت می شوند و یک سنسور نوری، بطری های خالی را تشخیص داده و با دادن فرمان به صفحه ی هدایت کننده، بطری های خالی را به خارج از خط تولید منتقل می کند، در این مثال سنسور نوری به عنوان ورودی و صفحه ی هدایت کننده به عنوان خروجی تلقی می گردند.



شکل ۱-۱. الف) سیستم کنترل متی برقی ، ب) سیستم کنترل تسمه نقاله

در سیستم های کنترل مبتنی بر رله (مدارهای فرمان)، منطق سیستم کنترل با انجام سیم کشی و استفاده از کنتاکت رله ها حاصل می گردد. به عنوان مثال در شکل ۱-۲، با وصل میکروسوییچ بویتر رله فعال می شود و از طریق کنتاکت آن موتور متی روشن می گردد.



شکل ۱-۲. کنترل مبتنی بر رله

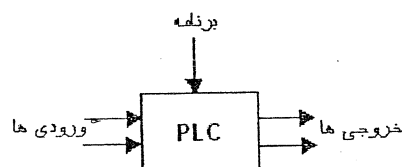
۱-۲) سیستم های کنترل کننده منطق برنامه پذیر (PLC)

در دهه ی ۱۹۶۰ میلادی با رشد صنایع اتومبیل سازی و به وجود آمدن رقابت شدید بین سازندگان نیاز به سیستم های کنترل پیچیده و تغییرات مداوم در خطوط تولید احساس می گردید. از یک طرف برای تعویض هر خط تولید مدت زمان زیادی صرف تغییر منطق کنترل (سیم کشی و رله ها) می شد و از طرف دیگر در صورت بروز عیب در سیستم های کنترل مبتنی بر رله، یافتن رله ی معیوب بسیار دشوار بود.

اولین کنترل کننده منطقی برنامه پذیر (PLC) در سال ۱۹۶۹ و در همین راستا یعنی کاهش زمان توقف خطوط تولید به وسیله ی کارخانه ی اتومبیل سازی General Motors بکار گرفته شد.

اصولاً PLC یک سیستم کنترل کننده ی مبتنی بر ریزپردازنده (میکروپروسسور) می باشد که:

- ۱- ورودی ها مانند شستی، کلید، سنسور ها و... و خروجی ها نیز کنتاکتور، سلونوئید و... شش کنترا و غیره مستقیماً به PLC سیم کشی می شوند، اما منطق کنترل یک برنامه کامپیوتری است که توسط کاربر وارد حافظه PLC می شود و اصول کار PLC نیز بدین ترتیب است که ابتدا ورودی ها را بررسی می کند، سپس مطابق برنامه موجود در حافظه خروجی ها را فعال یا غیرفعال می نماید. (شکل ۱-۳)



شکل ۱-۳. PLC به صورت شماتیک

- ۲- دارای زبان برنامه نویسی ساده و امکانات نرم افزاری متناسب می باشد به نحوی که کاربر می تواند به راحتی برنامه ی موجود در حافظه PLC را مشاهده و اشکال یابی نماید. همچنین در صورت وجود عیب به راحتی قادر است تا برنامه را تغییر داده و در صورت لزوم برنامه جدیدی را بنویسد.
- ۳- تمامی ابزار های کنترل موجود در مدارهای فرمان نظیر تایمر، شمارنده، توالی سنج، توابع محاسباتی و منطقی به صورت نرم افزاری در حافظه PLC شبیه سازی شده اند.
- ۴- قابلیت اتصال به انواع مختلف وسایل ورودی و خروجی را دارا می باشد
- ۵- قطع یا وصل بودن هر وسیله ورودی و خروجی توسط یک لامپ نشان دهنده (LED) مشخص می گردد.
- ۶- طراحی ملابیکی PLC به صورتی می باشد که تعویض بخش یا بخش هایی از PLC به راحتی امکان پذیر بوده تا بدین وسیله مجموعه زمان توقف سیستم کنترل به حداقل برسد.
- ۷- قابلیت تحمل حرارت، رطوبت، لرزش، نویز و تغییرات ولتاژ ورودی موجود در محیط های صنعتی را دارا است.
- ۸- PLC در تمامی مراحل اجرای برنامه به طور پیوسته اجزاء داخلی خود از قبیل: حافظه، پردازنده، سیستم ورودی/خروجی را امتحان و از صحت کارکرد آنها مطمئن می شود، و در صورت بروز عیب در هر کدام از آنها اجرای برنامه را متوقف می سازد.

FATEK (۱-۳)

کمپانی FATEK ، یکی از بزرگترین تولید کنندگان PLC در تایوان از سال ۱۹۹۲ است. از جمله خصوصیات این کمپانی کیفیت و قابلیت اطمینان بالای آن می باشد.

این کمپانی با تولید سری FB در سال ۱۹۹۲ کار خود را آغاز نمود و در سال ۲۰۰۳ ، نسل جدید PLC های خود ، با نام سری FBs که از تکنولوژی جدید System On Chip بهره می برند را وارد بازار نمود . کلیه محصولات FATEK دارای استاندارد های معتبر CE و UL می باشند .

شرکت درنامهر همکاری خود را با FATEK از سال ۱۳۷۸ آغاز نمود و تاکنون توانسته سهم قابل توجهی از بازار اتوماسیون ماشین آلات و خطوط تولید را به خود اختصاص دهد .



شکل دوم

سخت افزار

(۲-۱) سخت افزار

FATEK- PLC از سه قسمت اصلی تشکیل شده است.

(۲-۱-۱) منبع تغذیه (Power Supply)

FATEK- PLC بنا به انتخاب کاربر، دارای ۳ مدل تغذیه: ۲۲۰ VAC، ۲۴VDC و ۱۲VDC می باشد که برد تغذیه، ولتاژ ورودی را به ولتاژ (۱۲VDC \pm یا ۲۴VDC \pm) جهت استفاده ریزپردازنده و واحدهای ورودی/خروجی تبدیل می نماید و علاوه بر این واحد تغذیه 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. جدول زیر، ویژگی های تغذیه ی ورودی FATEK-PLC را نمایش می دهد.

AC		10-14 I/O	20-24 I/O	32-40 I/O	60 I/O
	نارمه ی ولتاژ ورودی	100 ~ 240 VAC – 15 % / + 10 %			
	نارمه ی فرکانس ورودی	50/60 Hz ± 5 %			
	حداکثر توان مصرفی	21 W	36 W		
DC	نارمه ی ولتاژ ورودی	12 VDC/24 VDC – 15 % / + 20 %			
	حداکثر توان مصرفی	15 W	24 W		

علاوه بر این، تمام واحد های تغذیه در این PLC، یک خروجی 24V DC خروجی نیز در اختیار کاربر قرار می دهند.

(۲-۱-۲) برد اصلی (Main board)

شامل:

- واحد پردازنده مرکزی (CPU): 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 مورد نظر را می دهد.

- حافظه (Memory): جهت ذخیره سازی برنامه و اطلاعات از آن استفاده می شود. برای آشنایی با حافظه می توان آن را به صورت ماتریسی از عناصر الکترونیکی در نظر گرفت که هر عنصر توانایی ذخیره کردن یک بیت (BIT) را در خود دارد (هر بیت می تواند ارزش صفر یا یک داشته باشد).

در این PLC، هر سطر حافظه حاوی ۱۶ بیت می باشد که به آن یک کلمه (Word) می گویند.

به خانه های ۱۶ بیتی از حافظه (Word) که از آن ها برای ذخیره و بازخوانی اطلاعات (Data) استفاده

می شود یک رجیستر (Register) گفته می شود. هر ستون از این ماتریس دارای آدرس منحصر به فردی

می باشد و CPU می تواند از طریق خطوط آدرس (ADDRESS BUS) به هر Word دلخواه از حافظه

دسترسی پیدا کند.

ظرفیت حافظه برنامه نویسی FATEK 20 K Word می باشد (بر خلاف سیستم ده دهی که هر کیلو معادل

۱۰۰۰ می باشد، در سیستم باینری هر کیلو معادل 1024 می باشد) بنابراین ظرفیت این حافظه معادل

$$(2 \times 1024 \times 16 = 32768) \text{ بیت می باشد.}$$

انواع حافظه موجود در PLC عبارت است از:

۱- حافظه سیستم عام. PLC (System ROM): حافظه فقط خواندنی. (Read Only Memory)، جهت

ذخیره سازی الگوریتم عملکرد PLC استفاده می گردد

۲- حافظه اطلاعات (Data RAM): حافظه خواندنی/نوشتنی جهت ذخیره اطلاعات لازم در طول اجرای

برنامه و همچنین اطلاعات مربوط به ابزارهای برنامه نویسی مانند تایمر ها، شمارنده ها و رله های داخلی

می باشد (به فصل ۴ مراجعه شود)

۳- حافظه جهت ذخیره سازی برنامه (User Program Memory): این حافظه جهت نگهداری برنامه در

داخل PLC استفاده شده و به صورت زیر می باشد:

CMOS RAM: حافظه خواندنی/نوشتنی که در صورت قطع برق محتویات آن توسط باتری پشتیبان (back up

battery) حفظ خواهد شد. CMOS RAM نوعی حافظه RAM کم مصرف می باشد. (

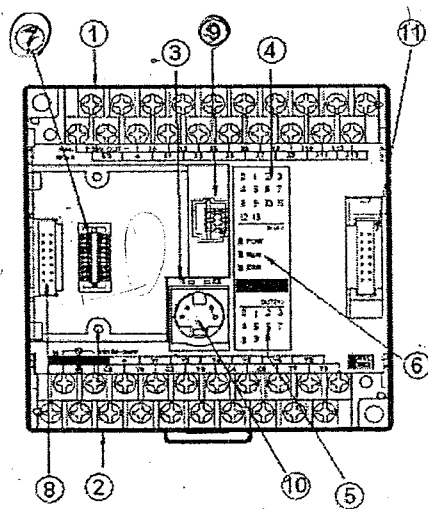
باتری پشتیبان این PLC از نوع لیتیوم با حداکثر طول عمر ۱۰ سال می باشد.

همچنین این PLC دارای یک Flash Rom جداگانه است که می تواند به صورت اختیاری برای ذخیره ی برنامه و اطلاعات ، مورد استفاده قرار گیرد (به عنوان پشتیبانی بیشتر از برنامه و یا جابه جایی برنامه و Data) و دارای ۶۴KWord حافظه می باشد.

- پورت ارتباط (communication port) : جهت انتقال برنامه از PC به PLC و برعکس، همچنین جهت

ارتباط PLC با HMI از آن استفاده می شود.

به شکل ۱-۲ توجه کنید .



- ① ترمینال ۲۴ VDC خروجی
- ② ترمینال های تغذیه اصلی ورودی و خروجی های دیجیتال
- ③ LED های نمایشگر انتقال (TX) و دریافت (RX) پورت 0
- ④ نمایشگر های ورودی های دیجیتال
- ⑤ نمایشگر های خروجی دیجیتال
- ⑥ نمایشگر های وضعیت PLC (POW, RUN, ERR)
- ⑦ محل اتصال بردهای ارتباطی (CB2, CB22, CB5, CB55, CB25, CBE) و بردهای آنالوگ (B2A1D, B4AD, B2DA) و برد های نمایشگر (LCD) (BDAP, PEPD)
- ⑧ محل اتصال به ماژول های ارتباطی (تنها برای مدل های MC/MN)
- ⑨ محل اتصال Memory Pack (CM55, CM22, CM25, CM25E, CM55E, CMGSM)
- ⑩ پورت 0 (RS232)
- ⑪ محل اتصال ماژول های توسعه (ورودی / خروجی و آنالوگ ها) تنها به روی PLC های I/O 20 به بالا موجود است

LED های نمایشگر وضعیت PLC :

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

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

کند) و هرگاه PLC در حالت RUN باشد ، هر ۰.۲۵ ثانیه یکبار چشمک می زند (چشمک تند)

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

۳-۱-۲) واحد ورودی/خروجی (I/O)

ارتباط PLC با دنیای خارج را برقرار می کند بدین صورت که PLC از طریق این واحد ولتاژ مورد استفاده در وسایل ورودی را به ولتاژ قابل درک برای CPU تبدیل می نماید.

همچنین بعد از اجرای برنامه توسط CPU وظیفه تبدیل فرمان CPU به ولتاژ مورد استفاده در وسایل خروجی را برعهده دارد. علاوه بر این واحد I/O با ایزوله نمودن ولتاژ وسایل ورودی/خروجی از PLC ایمنی سیستم را در برابر نویزهای موجود در محیط های صنعتی افزایش می دهند.

وسایل ورودی خروجی به دو دسته دیجیتال (Digital or Discrete) و آنالوگ (Analog) تقسیم بندی می گردند.

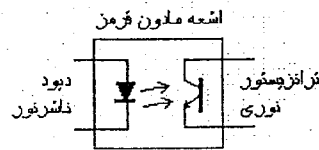
وسایل ورودی/خروجی دیجیتال تنها دارای دو وضعیت ON/OFF (فعال یا غیرفعال / قطع یا وصل) می باشند. اساس کار CPU موجود در PLC نیز بر مبنای سیستم باینری دودویی می باشد. در سیستم باینری هر بیت می تواند صفر یا یک باشد، بنابراین با یک بیت می توان تنها وضعیت یک کلید ورودی یا یک خروجی را نمایش داد. به عنوان مثال با وصل یک کلید، ولتاژ موجود در سر ترمینال کارت ورودی به ولتاژ 5 تبدیل می گردد و CPU وجود این ولتاژ را معادل 1 در سیستم باینری می بیند. همچنین در صورت قطع کلید، ولتاژ موجود در سر ترمینال ورودی قطع شده و CPU عدم وجود این ولتاژ را معادل صفر می بیند. بدین ترتیب CPU با 0 و 1 شدن یک بیت متوجه قطع و وصل یک کلید می گردد.

اما بسیاری از کمیت های فیزیکی نظیر دما، فشار، طول و ... دارای مقادیر پیوسته می باشند. مثلاً یک ترموکوپل متناسب با دمایی که در آن قرار دارد ولتاژی پیوسته تولید می نماید که برای تبدیل این ولتاژ به اعداد دیجیتال قابل فهم در PLC از ورودی آنالوگ استفاده می گردد و از طرف دیگر فرمان های صادر شده به وسیله CPU نیز به صورت باینری می باشد و باید از طریق یک خروجی آنالوگ تبدیل به ولتاژ پیوسته گردد.

در ادامه این بخش به معرفی انواع ورودی/خروجی های دیجیتال (Digital I/O) می پردازیم. عموماً وسایل ورودی و خروجی با ولتاژهای صنعتی نظیر 24 vdc، 48 vdc، 110 vdc، 110 vac و

۲۲۰ vac کار می نمایند، درحالی که ولتاژ قابل درک برای CPU، ۵ VDC + می باشد. برای انجام این تبدیل در

کارت های ورودی/خروجی از اپتوکوپلر استفاده می شود. (شکل ۲-۲)



شکل ۲-۲. اپتوکوپلر

هنگام وصل شدن سنسور ورودی، جریانی از دیود نورانی (LED) عبور می کند (حدود 10mA) برخورد نور فوق به فتوترانزیستور باعث وصل شدن آن شده. در نتیجه یک سیگنال ON/OFF در ورودی (به وسیله ی سنسور) با ولتاژ صنعتی به یک سیگنال ON/OFF با ولتاژ ۵ VDC تبدیل می گردد. همچنین فاصله ی هوایی موجود بین دیود نورانی و فتوترانزیستور از ولتاژهای لازم را تامین می کند. زیرا امکان اتصال تصادفی وسایل ورودی و خروجی با ولتاژ بالاتر همیشه وجود دارد، که در این صورت نباید آسیبی به CPU و مدارات داخلی PLC وارد شود.

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

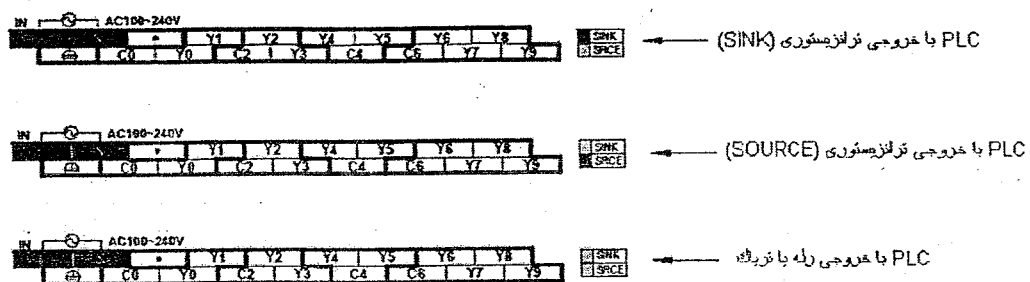
۱- نوع رله ای: در این نوع، فرمان CPU یک رله را فعال می کند و از طریق کنتاکت این رله، خروجی فعال می شود. از مزیت های این نوع خروجی می توان به توانی که آن در قطع و وصل خروجی های DC و AC با جریان ۲ آمپر همچنین به استقامت آن در برابر شوک های جاصل از بارهای القایی اشاره کرد. سرعت خروجی های رله ای کند (حدود 10ms) و به دلیل مکانیکی بودن عملکرد رله دارای محدودیت قطع و وصل (حدود چند میلیون بار) می باشد.

۲- نوع ترانزیستوری: در این نوع فرمان CPU یک ترانزیستور را فعال می کند و از طریق آن وسیله ی خروجی فعال می گردد. از مزیت های این نوع خروجی می توان به سرعت بالا (حدود 0.5ms) 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.

(5~30 VDC) اگر خروجی ترانزیستوری، NPN باشد، به آن خروجی Sink و اگر PNP باشد به آن Source

می گویند. (شکل ۲-۳)

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



شکل ۲-۳ خروجی های PLC

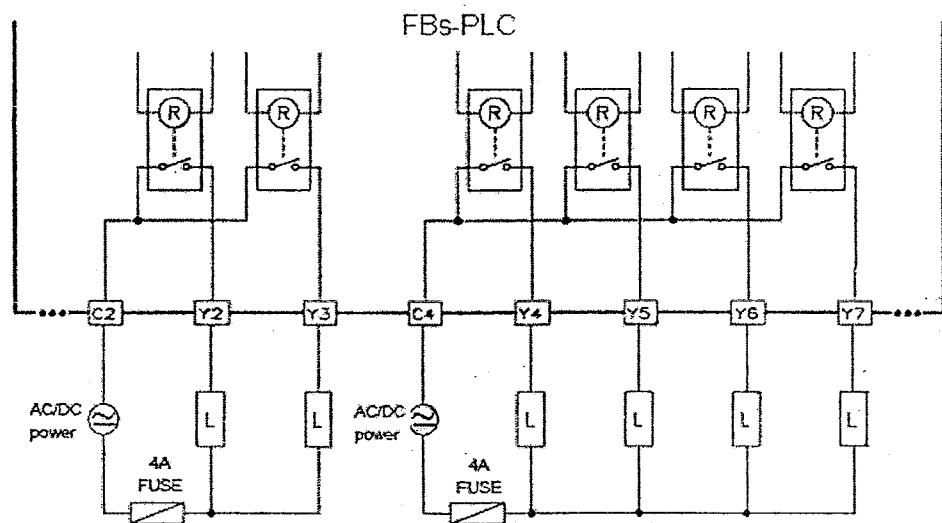
خروجی ها در FATEK PLC با حرف "Y" نمایش داده می شوند

هر دو یا چهار خروجی، دارای یک خروجی مشترک با هم می باشند که با حرف "C" نشان داده

شده و به دور Y ها و C مربوط به روی PLC یک خط پررنگ برای تمایز از خروجی های دیگر کشیده

می شود.

شکل ۲-۴ یک نمونه نحوه ی اتصال بار به خروجی های رله ای PLC را نمایش می دهد.



شکل ۴-۲. خروجی C2 برای رله های Y2 و Y3 مشترک است.
خروجی C4 برای رله های Y4، Y5، Y6، Y7 مشترک است.

در انواع یکپارچه این PLC تعداد ورودی ها و خروجی های دیجیتال در CPU اصلی به صورت جدول زیر می باشد:

تعداد خروجی ها	تعداد ورودی ها	مدل PLC
4	6	FBs-10MA/T FBs-10MC/T FBs-10MN/T
6	8	FBs-14MA/T FBs-14MC/T FBs-14MN/T
8	12	FBs-20MA/T FBs-20MC/T FBs-20MN/T
10	14	FBs-24MA/T FBs-24MC/T FBs-24MN/T
12	20	FBs-32MA/T FBs-32MC/T FBs-32MN/T

16	24	FBs-40MA/T FBs-40MC/T
16	28	FBs-44MN/T
24	36	FBs-60MA/T FBs-60MC/T FBs-60MN/T

ورودی های دیجیتال در FBS-PLC تا ۲۵۶ عدد قابل افزایش است.

خروجی های دیجیتال نیز تا ۲۵۶ عدد قابل افزایش است.

(به جز مدل های ۱۰ و ۱۴ ورودی-خروجی که غیر قابل افزایش می باشند)

تعداد خروجی ها	تعداد ورودی ها	مدل ماژول
4	4	FBs-8EA/T
-	8	FBs-8EX
8	-	FBs-8EY/T
8	8	FBs-16EA/T
-	20	FBs-20EX
16	-	FBs-16EY/T
10	14	FBs-24EA/T
16	24	FBs-40EA/T
24	36	FBs-60EA/T

حداکثر تا ۳۲ ماژول (دیجیتال و آنالوگ) را می توان به CPU اصلی اضافه کرد.

علاوه بر مدل های فوق دو ماژول گسترش دیگر وجود دارد که ورودی ها و خروجی های دیجیتال از طریق یک

سوکت (30pin) به ماژول متصل می شوند.

تعداد خروجی ها	تعداد ورودی ها	مدل ماژول
-	24	FBs-24EX
24	-	FBs-24EYT

ورودی ها و خروجی های آنالوگ نیز، هر کدام تا ۶۴ عدد قابل افزایش هستند
 ورودی ها در ولتاژ ۲۴ vdc ساخته شده و با حرف "X" نمایش داده می شوند.
 ترمینال مشترک ورودی ها به روی PLC با عبارت S/S مشخص شده است.
 هرگاه ۲۴۷ + به S/S متصل شود، پالس های منفی برای ورودی ها، معادل یک در نظر گرفته شده و هرگاه
 ۲۴۷- به S/S متصل شود، پالس های مثبت برای ورودی ها، معادل یک در نظر گرفته می شود

۲-۲) آدرس دهی ورودی/خروجی

PLC برای اجرای برنامه باید هر ورودی و یا خروجی را تشخیص دهد و این کار را با دادن یک آدرس
 منحصر به فرد به هر ورودی یا خروجی انجام می گیرد. هم چنین برنامه نویس نیز جهت نوشتن برنامه باید با
 نحوه ی این آدرس دهی آشنا باشد. در FATEK PLC آدرس مذکور بر روی هر ترمینال نوشته شده است، و
 از یک حرف و یک شماره در ادامه ی آن تشکیل گردیده است، به عنوان مثال ورودی ها به صورت ...X0, X1 و
 خروجی ها به صورت ...Y0, Y1 نامگذاری شده اند.

۲-۳) پاسخ زمانی PLC

نحوه ی انجام عملیات در سیستم PLC به صورت زیر است:

- ۱- PLC تمام ورودی ها را امتحان می کند (Scan Inputs)، ورودی هایی که وصل هستند از نظر PLC معادل یک و کلیدهایی که قطع هستند معادل صفر در نظر گرفته می شوند.
- ۲- CPU برنامه موجود در حافظه را خط به خط خوانده و اجرا می کند.
- ۳- PLC پس از پایان اجرای برنامه ، وضعیت خروجی ها را به واحد خروجی می فرستد.
- ۴- این سیکل مجدداً از شماره یک آغاز می شود.

کل زمان انجام مراحل ۱ تا ۳ برابر است با:

Scan Inputs + Scan Program + Scan Outputs

و آن را Scan Time می نامند.

چنانچه این زمان بیشتر از ۲۵/۰ ثانیه گردد، نشان دهنده ی این مطلب می باشد، که یکی از قسمت های PLC دچار اشکال شده بنابراین تایمر سگ نگهبان (Watch Dog Timer) 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 ۹۰ قابل تغییر است.

فرض نمایید که در یک برنامه با وصل یک ورودی یک خروجی باید فعال گردد. حال اگر تصادفاً ورودی در لحظه ای وصل شود که PLC در مرحله خواندن ورودی ها را به انجام رسانده باشد، در این صورت باید به اندازه ی یک Scan (مجموعه زمان Scan ورودی ها، Scan برنامه و Scan خروجی ها) صبر کند تا PLC متوجه وصل شدن این ورودی گردد و سپس به اندازه ی یک Scan دیگر صبر نماید تا خروجی فعال گردد. بنابراین حداکثر تاخیر در اجرای این برنامه، دو برابر زمان Scan است که این تاخیر را تاخیر نرم افزاری PLC می نامند.

از طرف دیگر به دلیل نویزهای موجود در محیط های صنعتی ورودی ها عموماً دارای فیلتری می باشند که این نیز به نوبه ی خود تاخیری را در دریافت ورودی ایجاد می نماید (حدود ۱۰ ms). همچنین اگر خروجی از نوع رله ای باشد مدت زمانی حدود ۱۰ ms نیز برای وصل رله ی خروجی خواهیم داشت، مجموع این دو زمان را تاخیر سخت افزاری PLC می نامند.

بنابراین پاسخ زمانی PLC حاصل جمع تاخیر نرم افزاری و سخت افزاری موجود در آن می باشد.

۴-۲) ویژگی های محیطی

دمای عملکرد	محیط بسته	حداقل	5°C
		حداکثر	40°C
	محیط باز	حداقل	5°C
		حداکثر	55°C
دمای نگهداری	-25 °C ~ +70 °C		
رطوبت نسبی دما	5 % ~ 95 %		

فصل سوم

معرفی نرم افزار

۱-۳) رابط برنامه نویسی (Programmer)

روش برنامه نویسی ، استفاده از کامپیوترهای شخصی (PC) و یا رومیزی (Lap Top) و نرم افزار ویژه برنامه نویسی (Win Proladder) می باشد. بنابراین کاربر از طریق کامپیوتر می تواند مستقیماً برنامه موجود در حافظه PLC را مشاهده و تغییر دهد (On Line Programming) 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 (از طریق Win Proladder) برنامه اجرا می گردد.

برخی قابلیت های نرم افزار برنامه نویسی (Win Proladder) FATEK به شرح زیر می باشد:

۱- امکان نوشتن برنامه به صورت Off Line و ذخیره آن به صورت یک فایل جهت دسترسی دوباره به

برنامه فوق .

۲- مشاهده ی اجرای یک برنامه در حال کار روی PLC (On Line Monitoring).

۳- مشاهده ی اجرای یک برنامه در حال کار بدون استفاده از PLC (Offline Simulation)

۴- قابلیت قطع و وصل هر ورودی یا فعال و غیرفعال کردن هر خروجی در حین اجرای برنامه (forcing).

بایستی دقت شود که عمل forcing حفاظت های موجود در سیستم را در نظر نمی گیرد، بنابراین

هنگام استفاده از آن باید دقت نمود!

۵- امکان تغییر برنامه در حال اجرا (RUN) که باید با دقت کامل انجام پذیرد.

۶- بعد از نوشتن برنامه تغییر و اشکال یابی آن باید به سهولت انجام می پذیرد.

از جمله دارای قابلیت های زیر می باشد :

الف) امکان نظارت و تغییر حالت ورودی ها (به صورت مجازی) ، خروجی ها و حافظه ی داخلی PLC از

طریق صفحه ی مانیتورینگ (Status Page)

ب) تهیه پرینت از برنامه ، پیکربندی ورودی ها و خروجی ها، جداول تنظیم کننده، توضیحات اضافی (Comments) و...

ج) امکان پیدا کردن سریع هر ورودی یا خروجی دلخواه (search) در برنامه و جایگزین نمودن آن ها.

د) امکان قرار دادن توضیحات اضافی در برنامه (Comments)

ه) قابلیت چک کردن برنامه از لحاظ خطای برنامه نویسی.

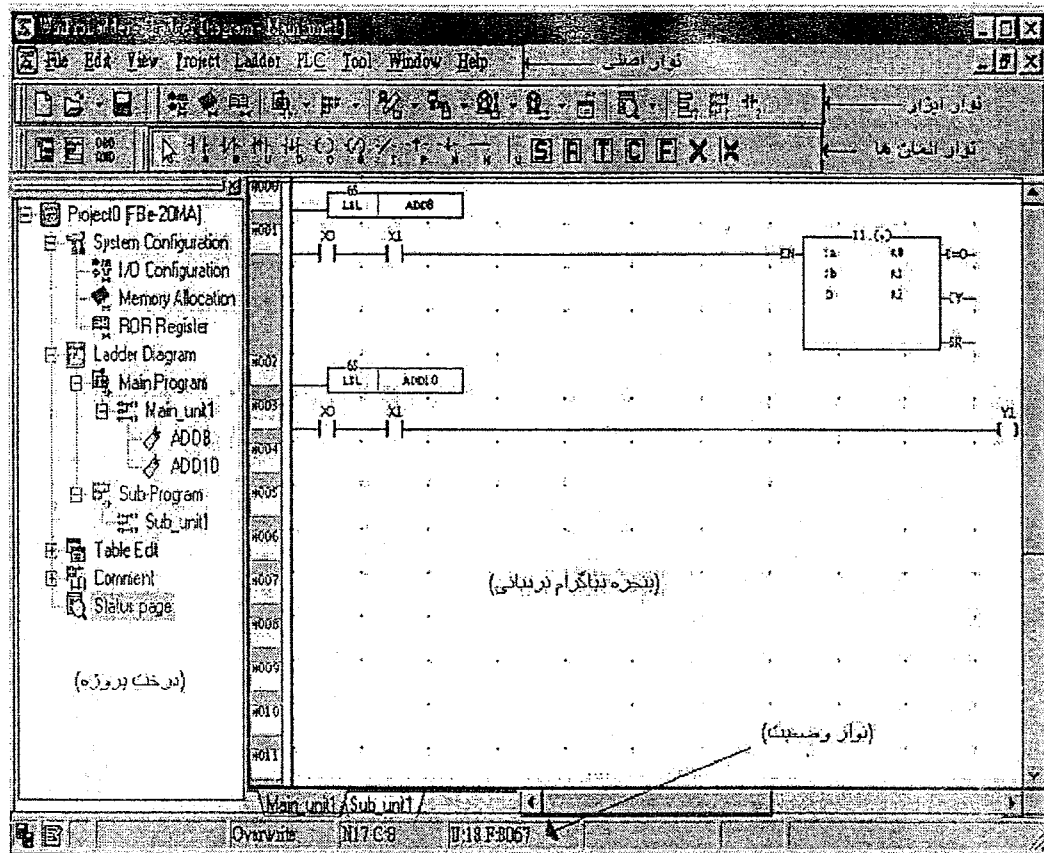
و) امکان قرار دادن رمز (Password) برای کل برنامه یا فقط زیر برنامه ها.

۶- مدیریت پروژه با دسته بندی قسمت های مختلف برنامه از طریق درخت مرتبه بندی شده ی پروژه (Project Tree)

۷- امکان اتصال PLC و PC با روش های متنوع (اتصال مستقیم از طریق RS232 و Ethernet و اتصال از راه دور به کمک مودم)

۳-۲) آشنایی با محیط نرم افزار WinProLadder

شکل ۳-۱ محیط کار عمومی نرم افزار WinProLadder را نمایش می دهد.



شکل ۳-۱

نوار اصلی

اکثر عملیات با وارد شدن به منوهای این نوار قابل اجرا است.

نوار ابزار

اکثر عملیات خاص، می توانند از طریق آیتم های این نوار فعال شوند

نوار المان ها

دکمه های این نوار، برای نوشتن یا تصحیح دیاگرام نردبانی به کار می روند. عملکرد این دکمه ها به طور مفصل در فصل ۴ آمده است.

نوار وضعیت

اطلاعاتی در مورد وضعیت اتصال PLC، موقعیت فعلی نشانه گر در صفحه و ... را به نمایش می گذارد.

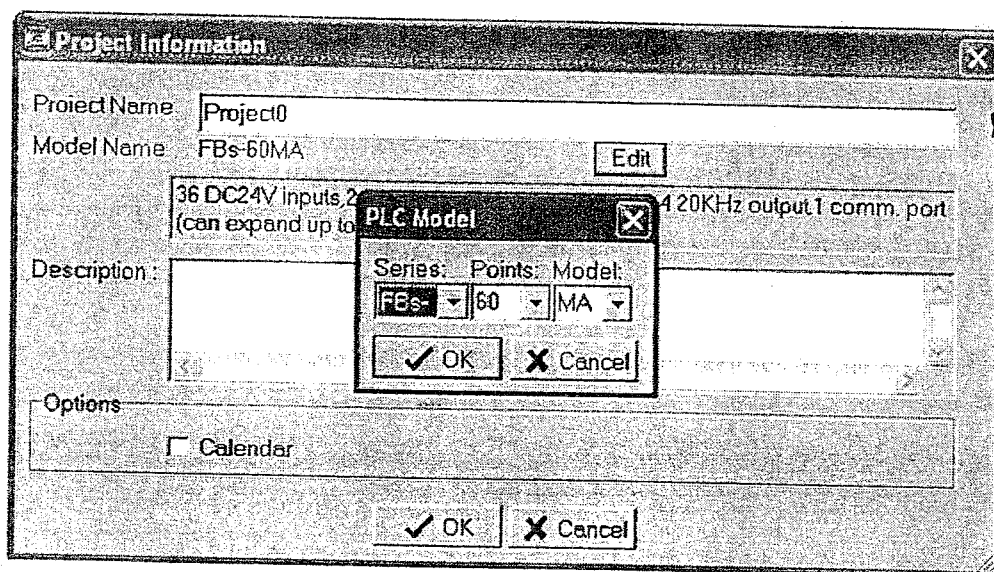
درخت پروژه

این قسمت طرح کلی پروژه را با کمک درخت مرتبه بندی شده نمایش داده و دسترسی به کاربرد های مختلف PLC را برای کاربر آسان نموده است.

پنجره ی دیاگرام نردبانی کاربر می تواند به ایجاد یا نظارت برنامه ی Ladder در این محیط بپردازد.

۳-۳) ایجاد یک برنامه ی جدید در PLC

برای شروع بعد از وارد شدن به برنامه ی WinProLadder، از منوی File، گزینه ی New Project را انتخاب کنید. در پنجره ای که باز می شود، نام پروژه ی مورد نظر خود را بنویسید و با کلیک بر روی Edit، مدل PLC مورد استفاده خود را وارد نموده و OK کنید. (شکل ۲-۳)

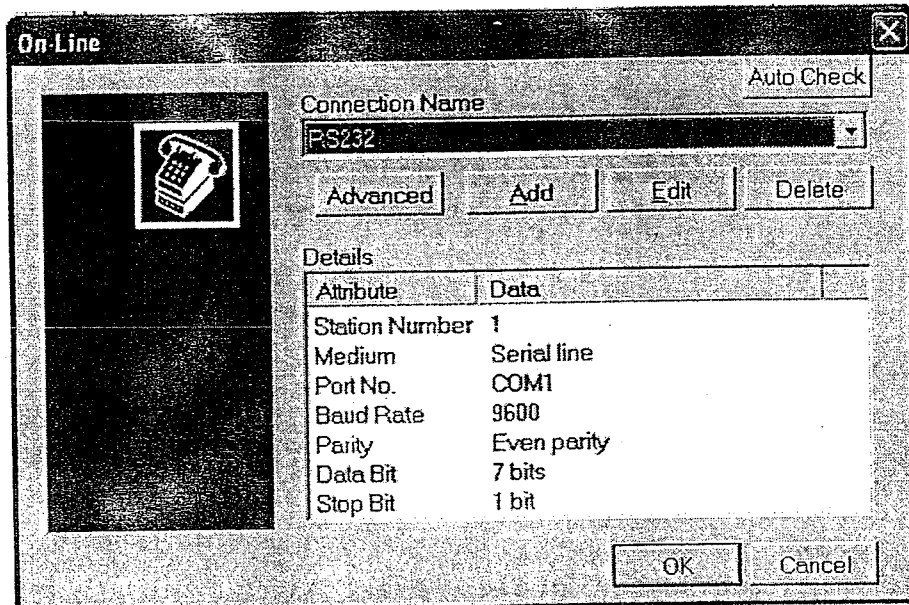


شکل ۲-۳

۳-۴) ذخیره ی برنامه ی موجود در PC به PLC

(برای اطلاع از نحوه ی اتصال کابل های مربوطه به پیوست ۱ مراجعه کنید.)

برای این که برنامه ای را که در WinProLadder نوشته اید، در PLC ذخیره کنید، از منوی File، گزینه ی Save As – To PLC را انتخاب کنید. پنجره ای باز می شود که در آن می توانید نوع ارتباط PLC با PC را تنظیم نمایید. (شکل ۳-۳) اگر از مشخصات پورت مورد استفاده مطلع نیستید، به روی گزینه ی Auto Check در پنجره ی باز شده کلیک کنید تا به صورت خودکار مشخصات پورت مورد استفاده ی شما، مشخص شود.



شکل ۳-۳

تنظیمات استاندارد در FATEK-PLC به صورت زیر می باشد:

۱. شماره ایستگاه PLC یا Station Number : تنظیم کارخانه = 1

۲. Baud Rate : تنظیم کارخانه = 9600

۳. Parity : تنظیم کارخانه = Even Parity

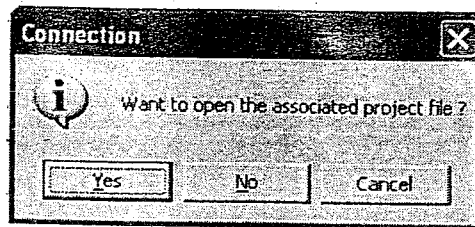
۴. Data Bit : تنظیم کارخانه = 7 bits

۵. Stop Bit : تنظیم کارخانه = 1bit

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

۳-۵ نحوه ی خواندن برنامه از روی PLC

برای انجام این کار ، از منوی File گزینه ی Open سبب ، Connect to PLC را انتخاب نمایید یا مستقیم کلید Ctrl+L را فشار دهید. سپس پنجره ی شکل ۴-۳ ظاهر می شود.



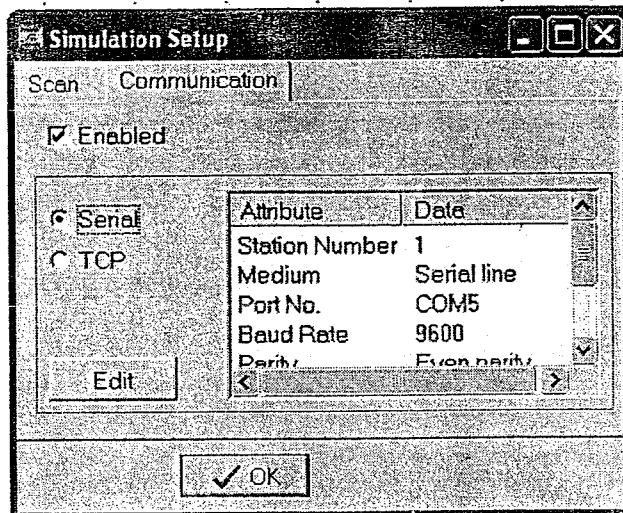
شکل ۳-۴

در این مرحله گزینه ی No را انتخاب نمایید . حال پنجره ی شکل ۳-۳ ظاهر می شود.
پس از انتخاب OK ، نرم افزار شروع به برقراری ارتباط با PLC نموده و در صورت عدم وجود رمز عبور (Password) برنامه را در اختیار شما قرار می دهد.

۳-۶) شبیه سازی برنامه بدون استفاده از PLC (Offline Simulation)

در حال حاضر برای شبیه سازی باید نسخه ی نرم افزار مورد استفاده 2.5 باشد و مطالبی که بیان می شود برای نسخه های قبل تر یا نسخه ی ۳ صدق نمی کند
پس از نوشتن برنامه ی خود ، به اء شروع شبیه سازی ، از منوی PLC ، گزینه ی Simulation را انتخاب کرده و برنامه ی نوشته شده را از گزینه ی 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 ی Setup Simulation را انتخاب کرده و وارد کشوی Communication در پنجره ی ظاهر شده می شویم . با تیک زدن گزینه ی Enabled ، تنظیمات پورت کامپیوتر ظاهر می شود که باید با تنظیمات پورت HMI منطبق باشد و در قسمت Port No ، شماره ی COM Port کامپیوتر که به HMI متصل است ، قرار داده می شود . (شکل ۳-۵)



شکل ۲-۵

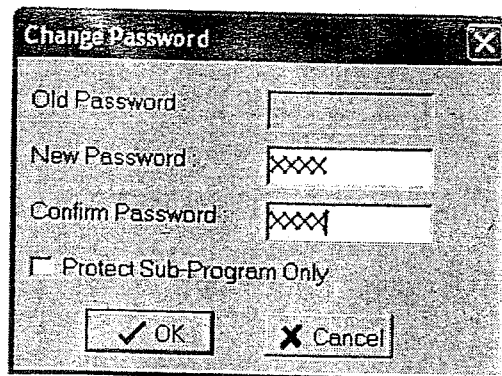
۳-۷ رمز گذاری (Password)

برای جلوگیری از باز شدن برنامه از داخل کامپیوتر یا از داخل PLC توسط نرم افزار ، برای برنامه ی 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 ی ظاهر شده ، رمز مورد نظر خود را در قسمت New Password وارد کرده و مجددا در قسمت Confirm Password وارد کنید . اگر گزینه ی Protect Sub-Program Only را تیک بزنید ، تنها از زیر برنامه ها حفاظت شده و دسترسی به برنامه ی اصلی برای همگان میسر خواهد بود . در ضمن بدون داشتن Password می توان برنامه ی رمز دار را به داخل PLC انتقال داده و راه اندازی (Run) نمود .

(شکل ۳-۶)



شکل ۴-۲

نوع دیگری از حفاظت برای برنامه ، گذاشتن Program ID برای برنامه و PLC ID برای PLC (تنها در حالت

Online و متصل به PLC) است . به این ترتیب ، برنامه ی 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

برنامه نویسی PLC

زبان ماشین مجموعه ای از کدهای باینری می باشد که تنها برای ریزپردازنده قابل درک است. از این رو برنامه نویسی با آن برای مهندسين دشوار می باشد. جهت سهولت در امر برنامه نویسی PLC- همانند کامپیوتر که ابتدا برنامه به زبان های سطح بالا نظیر C و Basic نوشته شده و سپس توسط کامپایلر به زبان ماشین تبدیل می شود- شرکت های سازنده ی PLC نیز هر کدام از زبان های سطح بالای خاص خود بهره می گیرند. در سال ۱۹۸۸ کمیته ی بین المللی الکتروتکنیکال (IEC) استاندارد IEC 1131-3 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.

زبان های برنامه نویسی PLC را می توان به دو دسته اصلی تقسیم بندی نمود:

۱- زبان های ترسیمی (گرافیکی) ۲- زبان های نوشتاری

دیاگرام نردبانی (Ladder Diagram) و بلوک دیاگرام (Function Block Diagram) دو روش

عمده ی برنامه نویسی به زبان ترسیمی می باشند.

زبان برنامه نویسی FBS-PLC از طریق نرم افزار Win ProLadder، ترکیبی از Ladder و Function Block است.

۴-۱) دیاگرام نردبانی

مدارهای فرمان عموماً به صورت دیاگرام نردبانی رسم می گردند. شکل ۴-۱-الف یک مدار الکتریکی و شکل

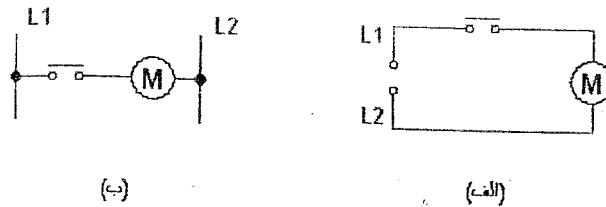
۴-۱-ب دیاگرام نردبانی معادل آن را در مدارهای فرمان نشان می دهد.

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 (دیاگرام نردبانی) 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

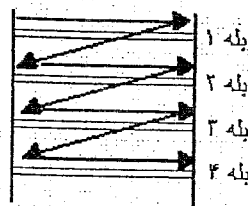
افقی که مانند پله های نردبانی می باشند خطوط برنامه هستند.



شکل ۴-۱. ترسیم یک مدار الکتریکی به صورت: الف) شمشیر، ب) دیگرام نردبانی

هنگام نوشتن برنامه به زبان LD (دیاگرام نردبانی) موارد ذیل را به خاطر بسپارید:

- ۱- هر خط از برنامه (هر پله نردبان) وظیفه‌ی خاصی را به عهده دارد.
 - ۲- در PLC برنامه از سمت چپ به راست و از بالا به پایین اجرا می‌گردد و بعد از اجرای کامل برنامه، اجرای آن دوباره از سر گرفته می‌شود. توجه فرمایید که اگرچه شکل ظاهری دیاگرام نردبانی در مدارهای فرمان و برنامه‌های PLC یکسان است اما نحوه‌ی پردازش آن‌ها متفاوت می‌باشد.
- (شکل ۴-۲) (رجوع کنید به طرز کار PLC در فصل یک).



شکل ۴-۲. چگونگی اجرای یک برنامه در PLC

- ۳- هر خط برنامه با تعدادی کنتاکت باز و یا بسته آغاز و با یک یا چند بوبین رله به انتها می‌رسد (این رله‌ها می‌توانند کمکی و یا خروجی باشند. در بخش ۵-۴ با رله‌های کمکی آشنا خواهید شد).
- ۴- کنتاکت‌ها در وضعیت عادی خود در برنامه نشان داده می‌شوند به عبارت دیگر کنتاکت‌های داخلی (نظیر کنتاکت‌های مربوط به رله‌های موجود در برنامه) با فرض غیرفعال بودن رله‌ها نمایش داده می‌شوند. در مورد کنتاکت‌های مربوط به وسائل ورودی بایستی دقت شود؛ چنانچه یک شستی NO به PLC متصل شده است و شما از یک کنتاکت باز جهت نمایش آن در برنامه استفاده کنید با فشار شستی NO کنتاکت فوق در برنامه بسته می‌شود و اگر از یک کنتاکت بسته جهت نمایش شستی NO در برنامه استفاده شود با فشار شستی NO این کنتاکت در برنامه باز می‌شود. اما در مورد اتصال یک

شستی NC به PLC مطلب فوق برعکس می باشد یعنی چنان چه جهت نمایش این شستی NC در برنامه PLC از یک کنتاکت بسته استفاده شود با فشار دادن شستی NC این کنتاکت بسته می شود و چنان چه جهت نمایش این شستی NC در برنامه PLC از یک کنتاکت باز استفاده گردد با فشار دادن شستی NC کنتاکت مذکور باز می گردد.

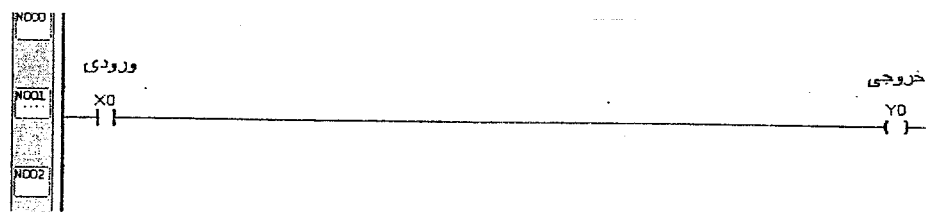
۵- از کنتاکت های یک رله می توان در خطوط مختلف برنامه استفاده نمود.

۶- هر کدام از کنتاکت های ورودی و رله های خروجی دارای آدرس منحصر به فرد می باشند. به عنوان مثال 40MA - FBs دارای ۲۴ ورودی و ۱۶ خروجی می باشد که آدرس آن ها به ترتیب ذیل است:

خروجی ها: Y0 ~ Y15 ورودی ها: X0 ~ X23

به عنوان مثال در شکل ۳-۴ با وصل شدن کنتاکت ورودی رله ی خروجی فعال می گردد و با باز شدن این

کنتاکت خروجی نیز غیرفعال می شود.

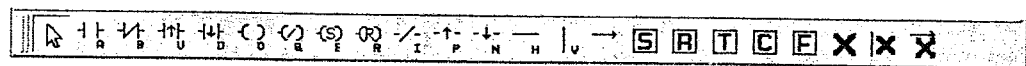


شکل ۳-۴. یک خط برنامه در دیگرام نردبانی

۲-۴) نوار المان ها

همانطور که در فصل پیش گفته شد ، آیتم های این نوار برای ایجاد و ویرایش برنامه به کار گرفته می شوند.

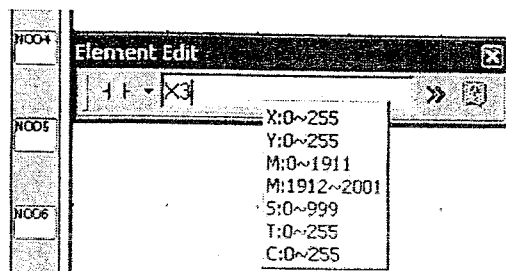
(شکل ۴-۴)



شکل ۴-۴

برای استفاده از این آیتم ها ، ابتدا به روی آن ها کلیک کرده تا انتخاب شوند سپس به روی پنجره ی دیگرام نردبانی کلیک کنید و ورودی یا خروجی مربوطه را وارد کنید یا می توانید مستقیم از طریق کیبورد حروف

مربوط به هر حرف را که در زیر هر المان آمده است به روی پنجره ی دیاگرام نردبانی وارد کنید . اگر نشانه -
 کر موس را به روی پنجره ی باز شده نگهدارید ، تمام المان هایی که می توان در آن قسمت وارد کرد نمایش
 داده می شود. (شکل ۵-۴)



شکل ۴-۵

کنتاکت باز ورودی :

کنتاکت بسته ی ورودی :

کنتاکت ورودی که تنها لحظه ی ایجاد پالس بالارونده به خروجی فرمان می دهد . (هرگاه پالس از ۰ به ۱

تغییر کند)

کنتاکت ورودی که تنها لحظه ی ایجاد پالس پایین رونده به خروجی فرمان می دهد . (هرگاه پالس از ۱

به ۰ تغییر کند)

کنتاکت باز خروجی :

کنتاکت بسته ی خروجی :

کنتاکت set کننده ی خروجی (latch می کند)

کنتاکت reset کننده ی خروجی (latch می کند)

این المان شرایط قبل خود را معکوس می کند (Inverse)

هرگاه جلوی کنتاکت ورودی قرار گیرد ، باعث می شود که فرمان ورودی ها تنها در لحظه ی

بالارونده ی پالس به خروجی اعمال شود

۴-۸ : هرگاه جلوی کنتاکت ورودی قرار گیرد، باعث می شود که فرمان ورودی ها تنها در لحظه ی

پایین رونده ی پالس به خروجی اعمال شود

۴-۸ : خط افقی کوتاه

۴-۸ : خط عمودی

→ : خط افقی طولانی

۴-۸ : set می کند (توضیح کامل در بخش ۴-۸)

۴-۸ : reset می کند (توضیح کامل در بخش ۴-۸)

۴-۹ : تایمر (توضیح کامل در بخش ۴-۹)

۴-۱۰ : کانتر (توضیح کامل در بخش ۴-۱۰)

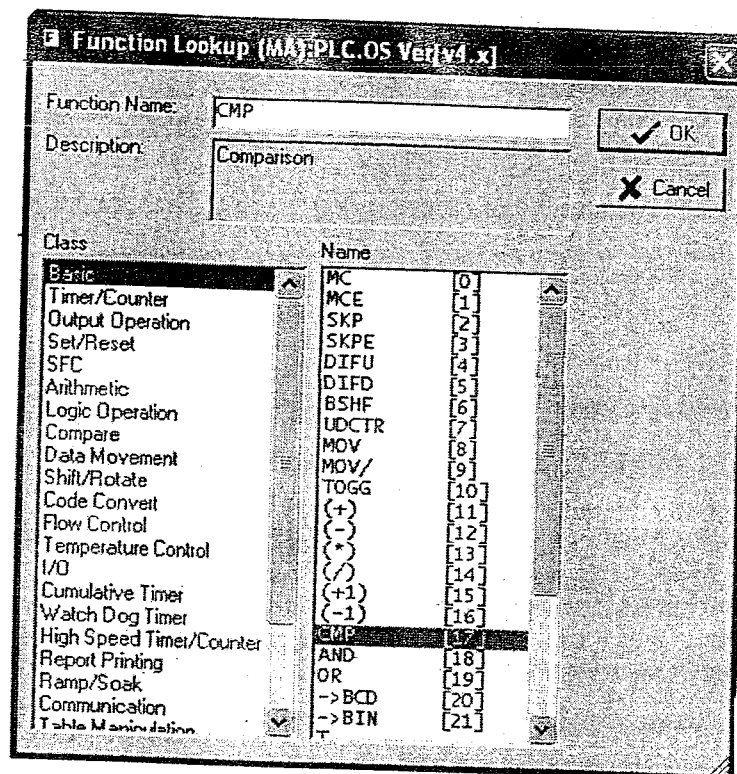
۴-۱۰ : دیگر توابع

در این زبان، هر تابع به صورت یک بلوک که دارای تعدادی ورودی و خروجی می باشد تعریف می گردد و در بالای بلوک نوع و شماره ی تابع نوشته می شود.

نرم افزار Win Proladder دارای حدود ۲۲۰ تابع در دسته های مختلف عملیات منطقی، ریاضی، ارتباطات،

مقایسه ای و ... می باشد. این توابع از طریق آیکون ۴-۱۱ در منوی Ladder قابل دسترسی هستند. پس از فشردن

این آیکون، سپس کلیک به روی صفحه ی برنامه نویسی، صفحه ی زیر ظاهر می گردد (شکل ۴-۶)



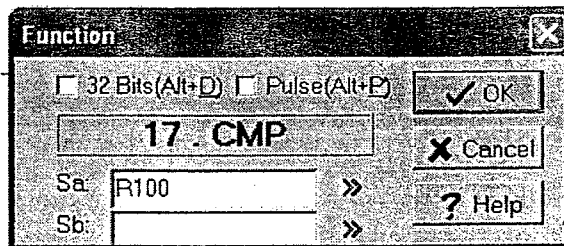
شکل ۴-۶

برای انتخاب تابع مورد نظر، ابتدا از قسمت Class، دسته بندی مرتبط با تابع را انتخاب کرده، سپس از قسمت Name، تابع را انتخاب می کنیم، یا م. توان در قسمت Function Name، نام تابع یا شماره ی آن را تایپ کرد. (به عنوان مثال تابع 17-Compare-مقایسه)

پس از OK کردن، پنجره ای باز می شود که رجیستر یا بیت دلخواه برای انجام عملیات را وارد خانه های خالی می کنیم. (برای اطلاعات بیشتر در مورد رجیسترها به بخش ۴-۶ رجیستر- مراجعه کنید)

اگر نشانگر موس را برای چند لحظه به روی این خانه های خالی نگه دارید، تمام بازه های رجیستری و اعداد

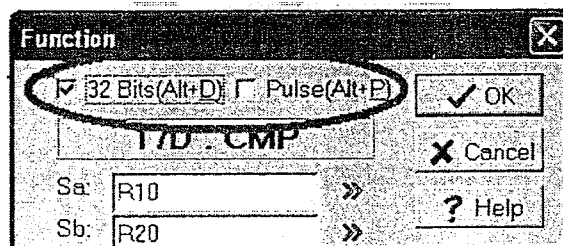
ثابت قابل استفاده در آن خانه نمایش داده می شود. مانند شکل ۴-۷



Index: RnY, RnZ (n=0~4167, 5000~8071)
 Index: Pn, RPn, DPn, RPNPn, DPNPn (m, n=0~9)
 WX: 0~240
 WY: 0~240
 WM: 0~1896
 WS: 0~984
 T: 0~255
 C: 0~199
 R: 0~3839
 R: 3840~3903
 R: 3904~3967
 R: 3968~4167
 R: 5000~8071 (ROR)
 R: 5000~8071
 D: 0~4095
 -32768~32767 (16-bit signed number)

شکل ۴-۷

بعضی توابع مانند تابع ۱۷ دارای دو گزینه کاربری می باشند (شکل ۴-۸)



شکل ۴-۸

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

(Double Registers)

به عنوان مثال تابع شکل ۴-۸، مقدار رجیسترهای R10-R11 (DR10) را با مقدار رجیسترهای R20-R21

(DR20) مقایسه می کند.

Pulse: هرگاه این گزینه فعال شود، تابع تنها لحظه ای اجرا می شود که ورودی کنترل تابع از ۰ به ۱ تغییر کند.

(لبه ی بالارونده ی پالس)

اگر می خواهید که محتوای رجیستر ها در کنار آن ها در توابع دیده شوند (در حالت On-line و Run)، از

طریق منوی View، کنار گزینه ی Register Content علامت تایید بزنید.

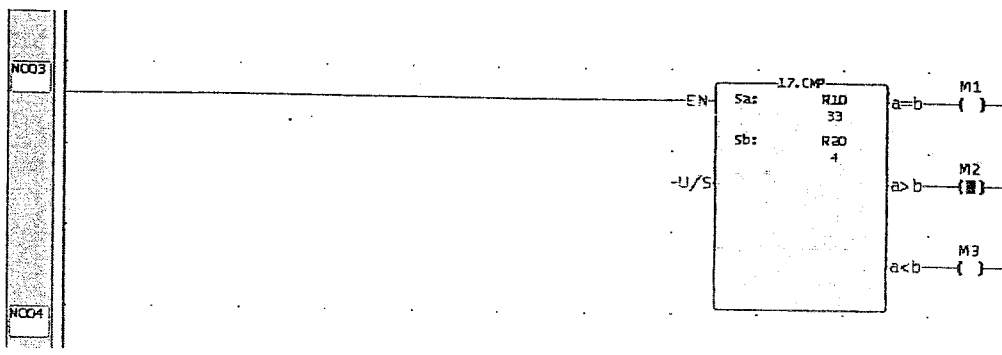
نمادهای پر کاربرد استفاده شده در توابع و کاربری آن ها در جدول زیر آمده است:

نماد	نام	کاربرد
S	Source - مبدا	محل قرار گیری عملوند مبدا که تنها از اطلاعات آن به عنوان مرجع استفاده شده و پس از اجرای تابع، مقدار آن تغییر نمی کند. اگر بیش از یک مبدا در یک تابع باشد، هر کدام با یک زیرنویس مشخص می شوند مثلاً Sa و Sb
D	Destination - مقصد	عملوند D برای ذخیره ی نتیجه ی اطلاعات استفاده شده و مقدار آن پس از انجام عملیات تغییر می کند
L	Length - طول	طول داده یا طول جدول را مشخص می کند
N	Number - تعداد	عدد ثابتی که به عنوان تعیین تعداد استفاده می شود و اگر در یک تابع بیش از یکی باشد، هر کدام با یک زیرنویس مشخص می شوند مثلاً Na ، Nb ، Ns و ...
Pr	Pointer - اشاره گر	برای اشاره به یک رجیستر یا بلوک داده ی مشخص استفاده شده و نمی توان از عدد ثابت استفاده کرد
CV	Current Value - مقدار جاری	در تایمر ها و کانتر ها، برای ذخیره ی مقدار جاری تایمر و کانتر استفاده می شود
PV	Preset Value - مقدار پیش تنظیم شده	در تایمر ها و کانتر ها، برای ذخیره ی مقدار مرجع یک تایمر یا کانتر به عنوان set point یا مقایسه استفاده می شود
T	Table - جدول	از ترکیب چند رجیستر متوالی یک جدول ایجاد شده و عملیات به روی رجیسترهای جدول صورت می گیرد. اگر در یک تابع بیش از یک جدول استفاده شود، هر کدام با یک زیرنویس مشخص می شوند مانند Ta ، Tb ، Ts و ...
M	Matrix - ماتریس	از ترکیب چند رجیستر متوالی یک ماتریس ایجاد شده و عملیات به روی یتهای ماتریس صورت می گیرد. اگر در یک تابع بیش از یک ماتریس استفاده شود، هر کدام با یک زیرنویس مشخص می شوند مانند Ma ، Mb ، Ms و ...

پس از تعیین پارامترهای تابع، OK کنید تا تابع در محیط برنامه نویسی ثبت شود. هر تابع بر اساس عملکرد آن

می تواند دارای تعدادی پایه ی ورودی و خروجی باشد. به عنوان مثال در تابع ۱۷، هرگاه مقدار رجیستر

$R10 > R20$ باشد، پایه خروجی 'a > b' فعال خواهد شد. (شکل ۹-۴)



شکل ۹-۴

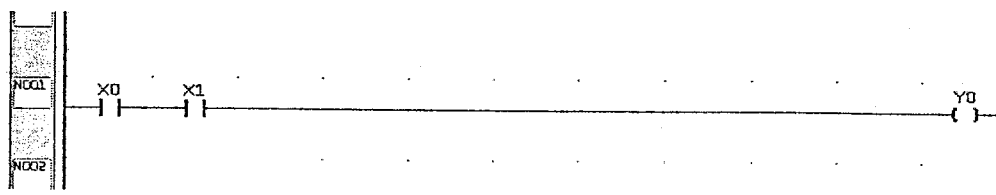
برای آشنایی با توابع مختلف FBS-PLC به فصل ۶ مراجعه کنید.

۳-۴ (کاربری های منطقی

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

۱-۳-۴ کاربری AND

شکل ۴-۱۰ چگونه ی ایجاد کاربری AND را در یک دیاگرام نردبانی نشان می دهد. در اینجا خروجی Y0 تنها وقتی فعال می شود که هر دو ورودی X0 و X1 وصل شده باشند. به عنوان مثال یک مه برقی زمانی روشن می شود که اولاً قطعه کار در مکان مناسب قرار گرفته باشد و ثانیاً تا سطح قطعه کار پایین آمده باشد.



شکل ۴-۱۰

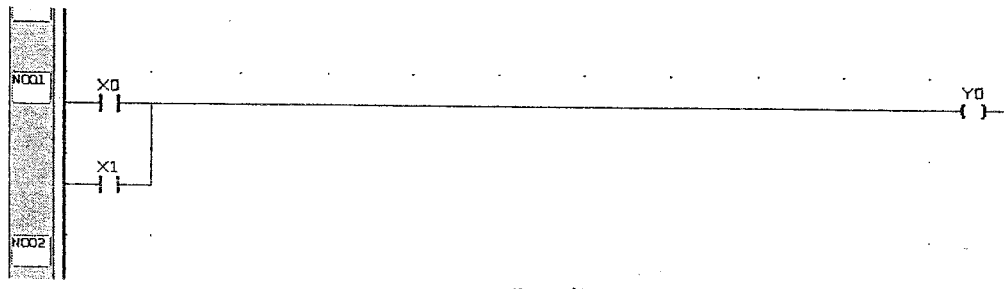
جدول درستی این تابع در زیر نمایش داده شده است.

خروجی $X0 \cdot X1 = Y0$

X0	X1	Y0 خروجی
۰	۰	۰
۰	۱	۰
۱	۰	۰
۱	۱	۱

OR کاربری (۴-۳-۲)

شکل ۴-۱۱ چگونه استفاده از کاربری OR در یک دیاگرام نردبانی را نمایش می دهد. در این جا خروجی در صورتی فعال می شود که هر کدام از ورودی های A و B یا هر دو وصل شوند.
به عنوان مثال یک بطری نوشابه به دلیل نداشتن درپوش و یا پرنبودن باید از خط تولید خارج شود.



شکل ۴-۱۱

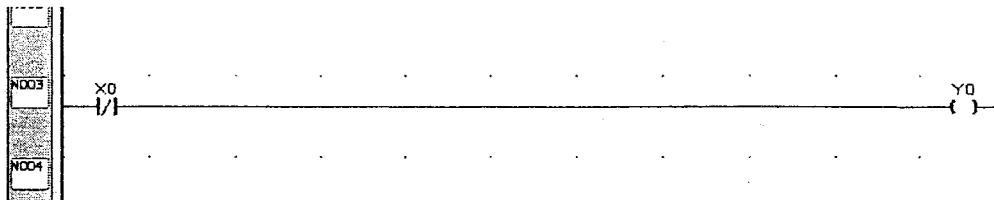
جدول درستی این تابع در زیر نمایش داده شده است.

$$X0 + X1 = Y0 \text{ خروجی}$$

X0	X1	خروجی Y0
۰	۰	۰
۰	۱	۱
۱	۰	۱
۱	۱	۱

NOT کاربری (۴-۳-۳)

در شکل ۴-۱۲ نحوه ی ایجاد کاربری NOT در یک دیاگرام نردبانی را مشاهده می کنید. در این وضعیت بوبین خروجی Y0 و ورودی X0 عکس یکدیگر می باشند. به عنوان مثال هنگامی که نور نباشد چراغ های موجود در خیابان روشن می شوند و هنگامی که نور خورشید وجود داشته باشد این چراغ ها خاموش می گردند.



شکل ۴-۱۲

جدول درستی این تابع در زیر آمده است.

X0	Y0 خروجی
۰	۱
۱	۰

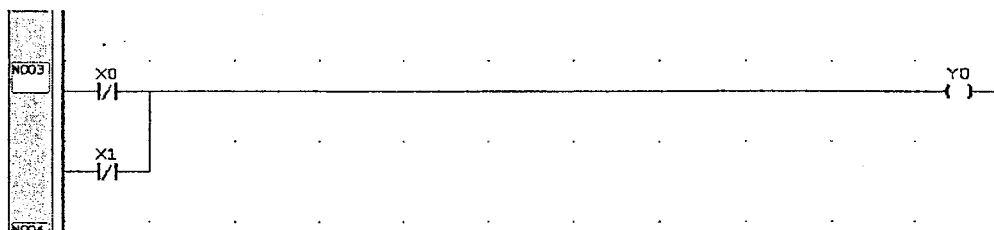
۴-۳-۴) کاربری NAND

این تابع از قرار گرفتن یک تابع NOT بعد از یک تابع AND به دست می آید که جدول درستی آن در زیر

آمده است:

X0	X1	Y0 خروجی
۰	۰	۱
۰	۱	۱
۱	۰	۱
۱	۱	۰

شکل ۴-۱۳) چگونه استفاده از تابع NAND را نشان می دهد.



شکل ۴-۱۳

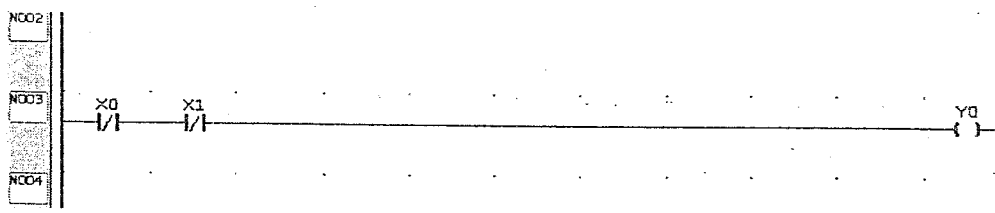
۴-۳-۵) کاربری NOR

این تابع از قرار دادن یک تابع NOT بعد از تابع OR به دست می آید.

جدول درستی این تابع در زیر نشان داده شده است.

X0	X1	خروجی Y0
۰	۰	۱
۰	۱	۰
۱	۰	۰
۱	۱	۰

شکل ۴-۱۴ نحوه ی استفاده از تابع NOR را نمایش می دهد.



شکل ۴-۱۴

۴-۳-۶) کاربری XOR

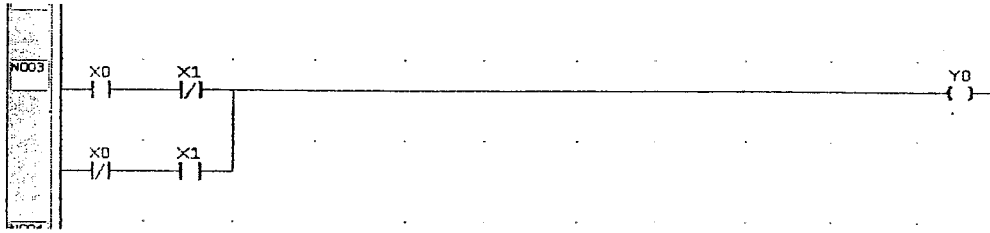
در تابع OR اگر هر کدام از ورودی ها و یا هر دو ورودی وصل شوند خروجی فعال می گردد اما در تابع

XOR خروجی تنها در صورت وصل شدن یکی از ورودی ها فعال می گردد.

جدول درستی این تابع در زیر نمایش داده شده است.

X0	X1	خروجی Y0
۰	۰	۰
۰	۱	۱
۱	۰	۱
۱	۱	۰

شکل ۱۵- ۴ نحوه ی استفاده از تابع XOR را نشان می دهد.

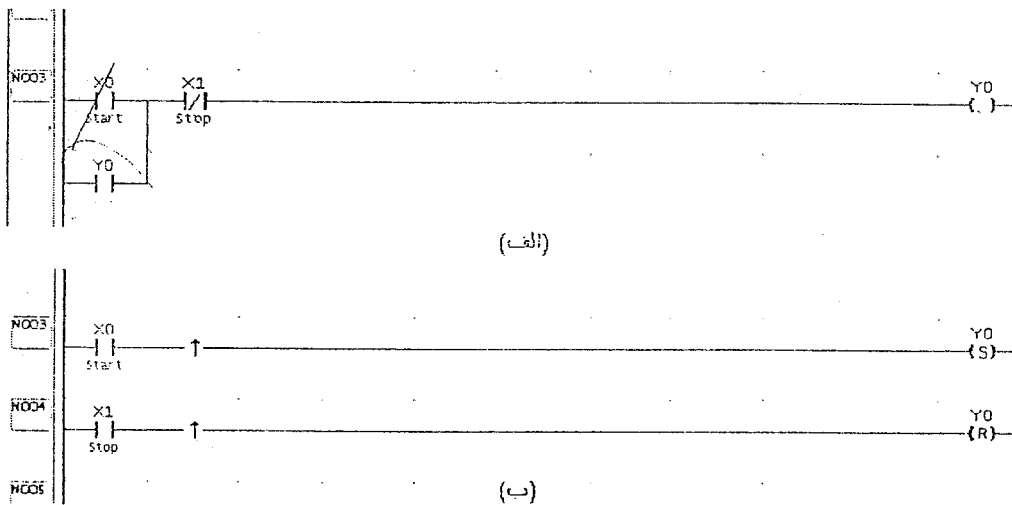


شکل ۱۵- ۴

۴-۴) مدار خودنگهدار (Latching)

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

شکل ۱۶- ۴ دو حالت ایجاد دیاگرام نردبانی مدار خودنگهدار را نمایش می دهد.



شکل ۱۶- ۴ استفاده از کنتاکت خود نگهدار

نکته: توجه کنید که در این مثال هر دو شستی استارت و استاپ متصل به PLC در حالت عادی دارای کنتاکت باز می باشند. در شستی استارت در برنامه نیز از کنتاکت باز استفاده شده است بنابراین با فشار این شستی



کنتاکت آن در برنامه بسته می شود. در شستی استاپ نیز، در برنامه (الف) از کنتاکت بسته استفاده شده است بنابراین با فشار این شستی کنتاکت آن در برنامه باز می گردد.

۵-۴) رله های کمکی (رله های داخلی)

در مدارهای فرمان عموماً می توان رله هایی را یافت که مستقیماً خروجی را فعال نمی کنند بلکه از کنتاکت های آن در جهت به وجود آوردن منطق موردنظر در مدار استفاده می شود. در داخل حافظه ی 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 مورد نظر را در برنامه PLC به وجود می آورند و نهایتاً منجر به تحریک شدن یک خروجی می شوند.

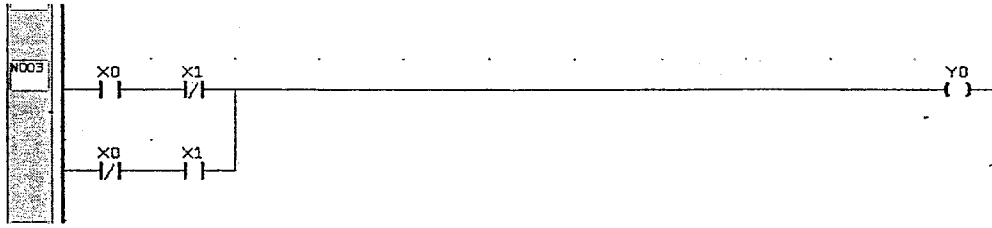
برخلاف رله های الکترومکانیکی که تعداد محدودی کنتاکت باز یا بسته دارند در رله های کمکی موجود در داخل PLC می توانیم به دفعات از این کنتاکت ها استفاده نمائیم. وجه تمایز رله های کمکی از یک رله ی خروجی در شماره ی آدرس استفاده شده آن در برنامه PLC می باشد.

۱-۵-۴) نحوه ی استفاده از رله های کمکی در برنامه

در دیاگرام نردبانی همانند رله های خروجی برای رله های کمکی نیز از نماد (-) استفاده می گردد. به عنوان مثال استفاده از M100 به همراه نماد (-) نشان دهنده ی این مطلب است که M100 یک رله کمکی می باشد. کنتاکت های این رله ی کمکی نیز با نماد  به عنوان کنتاکت باز (NO) و  به عنوان کنتاکت بسته (NC) نمایش داده می شوند. و شماره ی M100 نیز جهت شناسایی این کنتاکت ها به کار می رود. (M از اول کلمه ی Memory گرفته شده است).

جهت فعال یا غیر فعال کردن رله های کمکی در حالت Online (run)، به روی رله ها در محیط برنامه، راست کلیک کرده و گزینه ی On یا Off را انتخاب کنید.

شکل ۱۵- ۴ نحوه ی استفاده از تابع XOR را نشان می دهد.

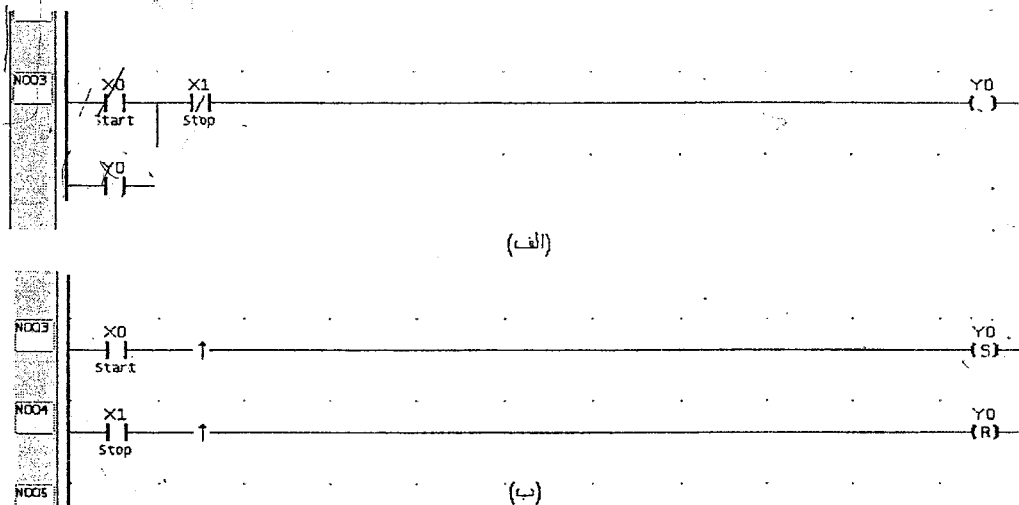


شکل ۱۵- ۴

۴-۴) مدار خودنگهدار (Latching)

در مدارات فرمان عموماً جهت روشن و خاموش کردن موتور از دو شستی استارت و استاپ استفاده می شود. هنگامی که شستی استارت را فشار دهیم یک رله فعال می شود و از طریق کنتاکت خودنگهدار این رله مسیر وصل می ماند. بنابراین با برداشتن دست از روی شستی، رله هم چنان فعال می ماند و تنها راه غیر فعال کردن این رله فشار روی شستی استاپ است.

شکل ۱۶- ۴ دو حالت ایجاد دیاگرام نردبانی مدار خودنگهدار را نمایش می دهد.



شکل ۱۶- ۴. استفاده از کنتاکت خود نگهدار

نکته: توجه کنید که در این مثال هر دو شستی استارت و استاپ متصل به PLC در حالت عادی دارای کنتاکت باز می باشند. در شستی استارت در برنامه نیز از کنتاکت باز استفاده شده است بنابراین با فشار این شستی

کنتاکت آن در برنامه بسته می شود. در شستی استاپ نیز، در برنامه (الف) از کنتاکت بسته استفاده شده است بنابراین با فشار این شستی کنتاکت آن در برنامه باز می گردد.

۵-۴) رله های کمکی (رله های داخلی)

در مدارهای فرمان عموماً می توان رله هایی را یافت که مستقیماً خروجی را فعال نمی کنند بلکه از کنتاکت های آن در جهت به وجود آوردن منطق موردنظر در مدار استفاده می شود. در داخل حافظه ی 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 مورد نظر را در برنامه PLC به وجود می آورند و نهایتاً 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 ی خروجی در شماره ی آدرس استفاده شده ان در برنامه PLC می باشد.

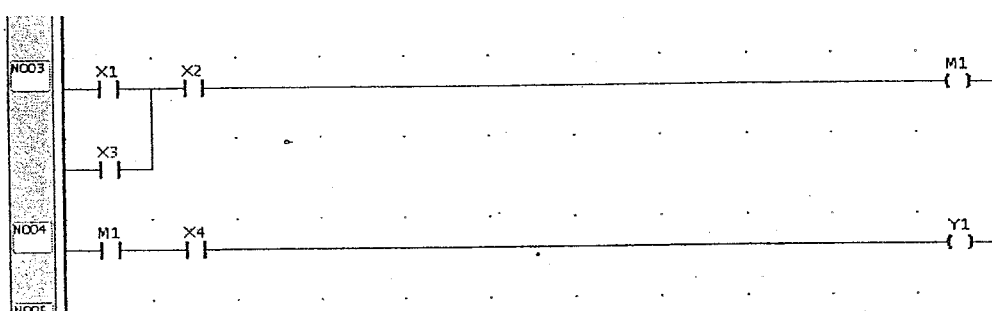
۱-۵-۴) نحوه ی استفاده از رله های کمکی در برنامه

در دیاگرام نردبان، همانند رله های خروجی برای رله های کمکی نیز از نماد () - استفاده می گردد. به عنوان مثال استفاده از M100 به همراه نماد () - نشان دهنده ی این مطلب است که M100 یک رله کمکی می باشد. کنتاکت های این رله ی کمکی نیز با نماد $\left| \text{---} \right|$ به عنوان کنتاکت باز (NO) و $\left| \text{---} / \right|$ به عنوان کنتاکت بسته (NC) نمایش داده می شوند. و شماره ی M100 نیز جهت شناسایی این کنتاکت ها به کار می رود. (M از اول کلمه ی Memory گرفته شده است).

جهت فعال یا غیر فعال کردن رله های کمکی در حالت Online (run)، به روی رله ها در محیط برنامه، راست کلیک کرده و گزینه ی On یا Off را انتخاب کنید.

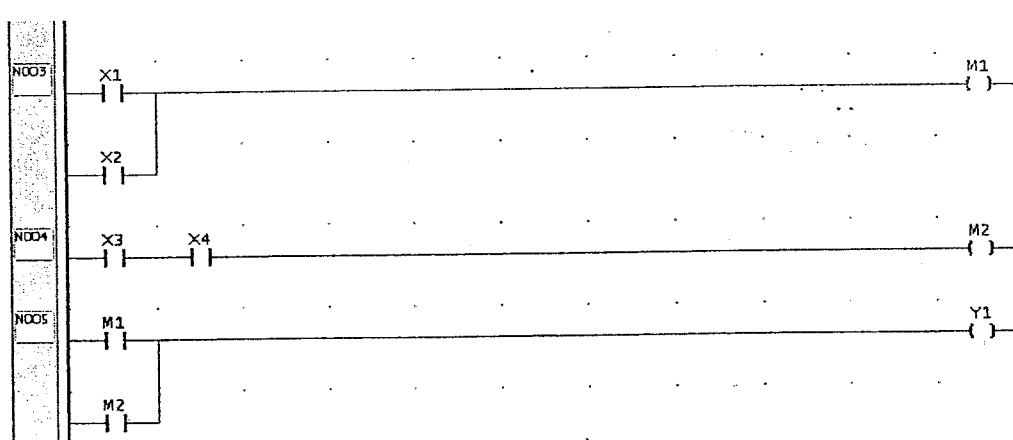
۴-۵-۲) کاربرد رله های کمکی

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 ۴-۱۷ هنگامی که کنتاکت X2 بسته باشد، با وصل حداقل یکی از ورودی های X1 یا X3 رله ی 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.



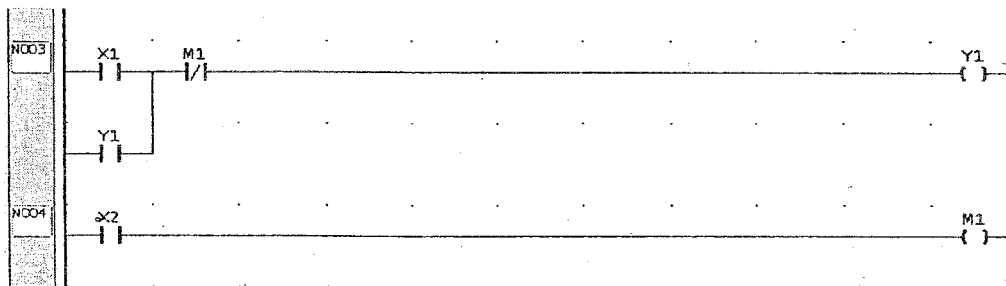
شکل ۴-۱۷

شکل ۴-۱۸ 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 ها X1 تا X2 تحریک می شود و رله ی کمکی M2 نیز با بسته شدن ورودی های X3 و X4 فعال می گردد که نهایتاً رله ی خروجی Y1 نیز در صورتی که حداقل یکی از رله های کمکی M1 یا M2 تحریک شده باشد فعال می گردد.



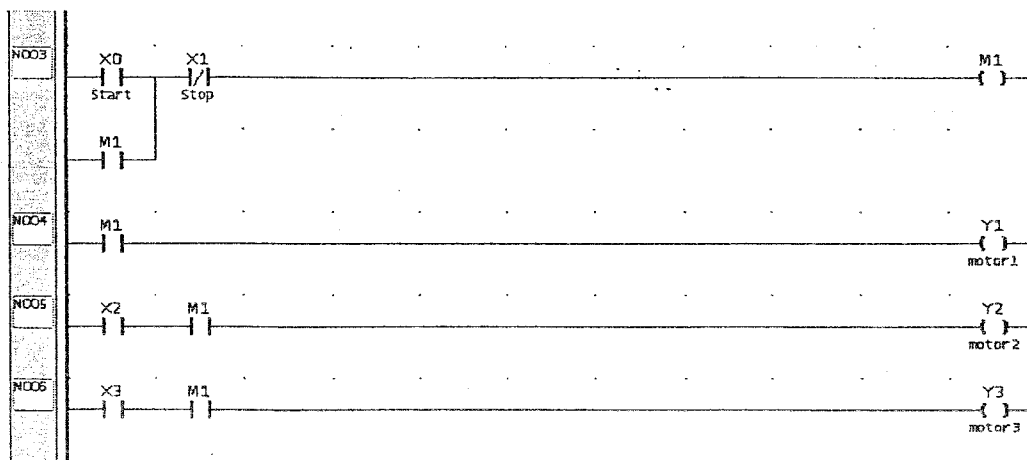
شکل ۴-۱۸

در شکل ۴-۱۹ هنگامی که شستی X1 وصل شود رله خروجی Y1 تحریک می گردد و توسط کنتاکت خودنگهدار Y1 این رله تحریک باقی می ماند که اصطلاحاً این عمل latch می گویند. با وصل شدن ورودی X2 رله Y1 کمکی M1 تحریک می شود و در اولین خط برنامه کنتاکت بسته M1 تغییر وضعیت می دهد و رله خروجی Y1 غیرفعال می شود. (unlatch).



شکل ۴-۱۹

در دیاگرام نردبانی شکل ۴-۲۰ با فشار شستی استارت رله کمکی M1 تحریک می شود و توسط کنتاکت خودنگهدار M1 تحریک می ماند. در خط دوم با تحریک شدن M1 خروجی Y1 فعال و موتور شماره یک روشن می شود. در خط سوم و چهارم با به وجود آمدن شرایط لازم دیگر رله های خروجی Y2, Y3 فعال می گردند و در نتیجه موتورهای دوم و سوم روشن می شوند و با فشار شستی استاپ هر سه موتور (در صورت روشن بودن) متوقف می شوند.



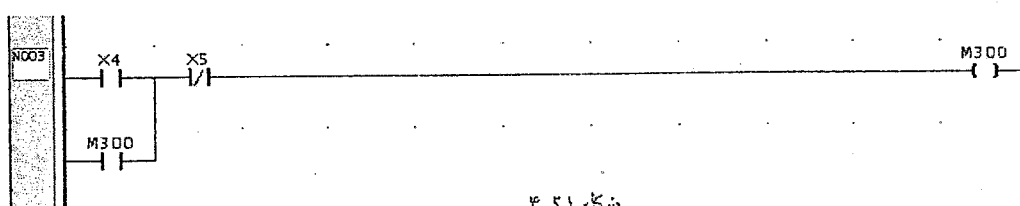
شکل ۴-۲۰

۳-۵-۴) رله های کمکی دارای باطری پشتیبان (retentive)

اگر در هنگام کار برق 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 باطری پشتیبان موجود در PLC، ارزش خود را حفظ نمایند. این رله ها را پایا (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 ۴-۲۱ با وصل شدن X4 رله ی کمکی M300 فعال می شود، اگر در این لحظه برق سیستم قطع شود X4 غیرفعال می شود. (در نتیجه کنتاکت X4 باز می شود و کنتاکت X5 بسته می ماند). در این صورت چون رله ی M300 پایا می باشد با وصل مجدد برق از طریق خودنگهدارش فعال باقی می ماند.



شکل ۴-۲۱

جدول زیر بازه ی رله های کمکی FATEK PLC را نمایش می دهد.

رله های کمکی	M0 ~ M1399 (1400)	قابل تنظیم به عنوان رله های Retentive یا NonRetentive
	M1400 ~ M1911 (512)	NonRetentive
	M1912 ~ M2001 (90)	رله های خاص
Step Relays	S0 ~ S19 (20)	NonRetentive (no control)
	S20 ~ S999 (980)	قابل تنظیم به عنوان Retentive یا NonRetentive

نحوه ی تنظیم رله های Retentive در بخش ۷-۴ آمده است.

۴-۶) رجیستر (register)

به طور کلی یک رجیستر به هر نوع حافظه ی الکترونیکی که بتواند اطلاعات را در خود ذخیره نماید اطلاق می گردد. رجیسترها در PLC از بهم پیوستن ۱۶ 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 (باینری) 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 ۱۶ بیتی (کلمه word) را در خود ذخیره نماید.

عدد چهار بیتی ۱۱۱۱ را در نظر بگیرید. بیت سمت راست را بیت کم ارزش (LSB) و بیت سمت چپ را بیت پرارزش MSB می نامند. ارزش بیت کم ارزش در مبنای دسیمال (دهی) $2^0 = 1$ و بیت پرارزش $2^3 = 8$ می باشد و ارزش عدد چهار بیتی ۱۱۱۱ در مبنای ده (دسیمال) به صورت ذیل محاسبه می شود.

$$(1111)_2 = 2^3 + 2^2 + 2^1 + 2^0 = 8 + 4 + 2 + 1 = (15)_{10}$$

و این بدان معناست که یک رجیستر چهار بیتی حداکثر توان نمایش ۱۶ عدد یعنی از عدد ۰ تا عدد ۱۵ را در مبنای دسیمال دارد. بنابراین یک رجیستر ۸ بیتی توان نمایش $2^8 = 256$ عدد را از صفر تا ۲۵۵ و یک رجیستر ۱۶ بیتی توان نمایش $2^{16} = 65536$ عدد را از صفر تا ۶۵۵۳۵ دارد می باشد. (در این PLC از صفر تا ۳۲۷۶۷ است و یک بیت مربوط به علامت می باشد)

رجیسترها در FATEK PLC، با نماد R و D نمایش داده می شوند.

همچنین از به هم پیوستن ۱۶ رله ی کمکی پی در پی نیز یک کلمه (word) به وجود می آید.

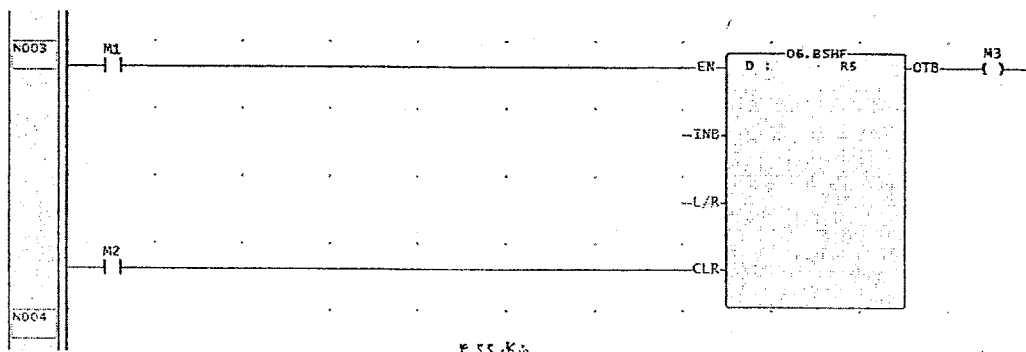
به عنوان مثال WMO که به معنای یک word (M0-M15) می باشد

۴-۶-۱) شیفت رجیستر (Shift register)

در خطوط تولید و تسمه نقاله ها عموماً 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 ی استفاده از شیفت رجیسترها در PLC می باشد. در این بخش با چگونگی استفاده از شیفت رجیسترها و کاربرد آن ها در صنعت آشنا خواهید شد.

- شیفت به معنای انتقال پیدا کردن محتویات یک خانه (bit) به خانه بعدی می باشد.

یک تابع شیفت رجیستر (تابع شماره ۴) در داخل PLC دارای ۵ ورودی و خروجی زیر می باشد (شکل ۴-۲۲):



شکل ۴-۲۲

۱- ورودی 'EN' برای فعال سازی تابع.

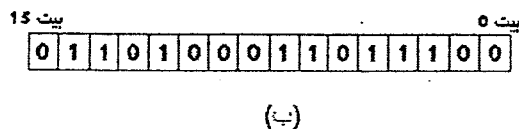
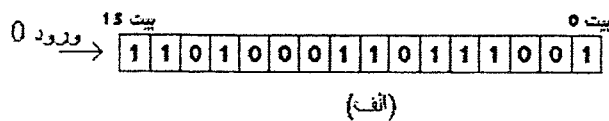
۲- ورودی 'INB' جهت وارد کردن بیت جدید به اولین خانه استفاده می شود.

۳- ورودی 'L/R' هرگاه وصل باشد، شیفت رجیستر به سمت چپ و اگر قطع باشد، به سمت راست خواهد بود.

۴- 'CLR' جهت پاک نمودن محتویات کلیه خانه های رجیستر به کار می رود.

۵- محتویات بیت آخر پس از شیفت پیدا کردن در خروجی 'OTB' ظاهر می شود.

رجیستر ۱۶ بیتی شکل ۴-۲۲ الف را در نظر بگیرید. با فعال شدن ورودی EN در صورتی که ورودی INB غیرفعال باشد یک ۰ وارد رجیستر می شود و اگر ورودی L/R غیرفعال باشد، شیفت رجیستر به سمت راست خواهد بود و نهایتاً شکل ۴-۲۲ ب را خواهیم داشت.



شکل ۴-۲۳

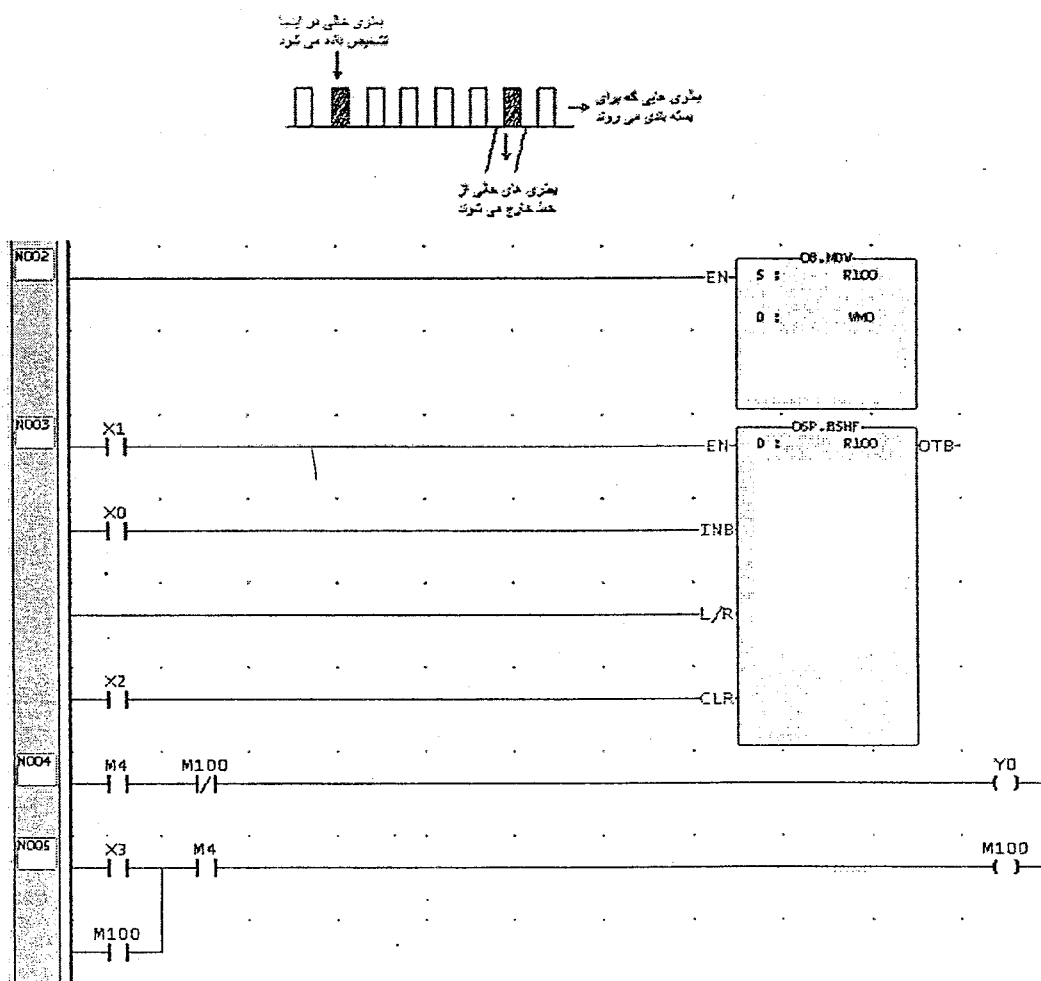
با فعال شدن ورودی 'CLR' محتویات تمام خانه های رجیستر صفر می گردد.

۲-۶-۴) ردیابی محصولات معیوب در خط تولید با کمک شیفت رجیستر

در شکل ۴-۲۴ سنسور نوری X0 در زمان خالی بودن بطری نوشابه به صورت لحظه ای وصل می شود و یک بیت با ارزش یک به اولین خانه ی رجیستر R100 وارد می کند. سنسور X1 نیز با عبور هر بطری نوشابه (اعم از پر یا خالی) یک بار به صورت لحظه ای وصل می شود، و با هر بار وصل شدن لحظه ای X1 یک عمل جابجایی در محتویات رجیستر انجام می گیرد.

بنابراین پس از عبور ۴ بطری نوشابه رله ی M4 فعال شده و نشان دهنده ی این مطلب است که بطری خالی به مکان مناسب رسیده و در این حالت Y0 فعال شده و اهرم مربوط به خارج کردن بطری خالی عمل کرده و بطری را از خط تولید خارج می نماید. سپس اهرم مربوطه به میکروسوئیچ X3 برخورد کرده و سبب فعال شدن M100 و غیرفعال شدن Y0 گردیده و اهرم را به جای اولیه ی خود باز می گرداند.

تابع ۸ (MOV) برای انتقال R100 به WM0 استفاده شده است. (به بخش ۴-۱۱-۱ مراجعه کنید)



شکل ۴-۲۴

۳-۶-۴) رجیسترهای دارای باتری پشتیبان (retentive)

همانند رله های کمکی Retentive، گاهی به حفظ مقدار رجیستر ها نیز پس از قطع برق نیاز است. به همین

دلیل تعدادی از رجیستر ها قابل تنظیم هستند که به عنوان رله ی Retentive استفاده شوند.

جدول زیربازه ی رجیسترهای FATEK PLC را نمایش می دهد.

R0~R3839	NonRetentive یا Retentive
R3840~R4160	رجیسترهای خاص
R5000~R8071	Retentive
D0~D3999	Retentive

۴-۷) نحوه ی تنظیم رله های کمکی و رجیسترهای Retentive

از روی درخت پروژه ، به روی گزینه ی

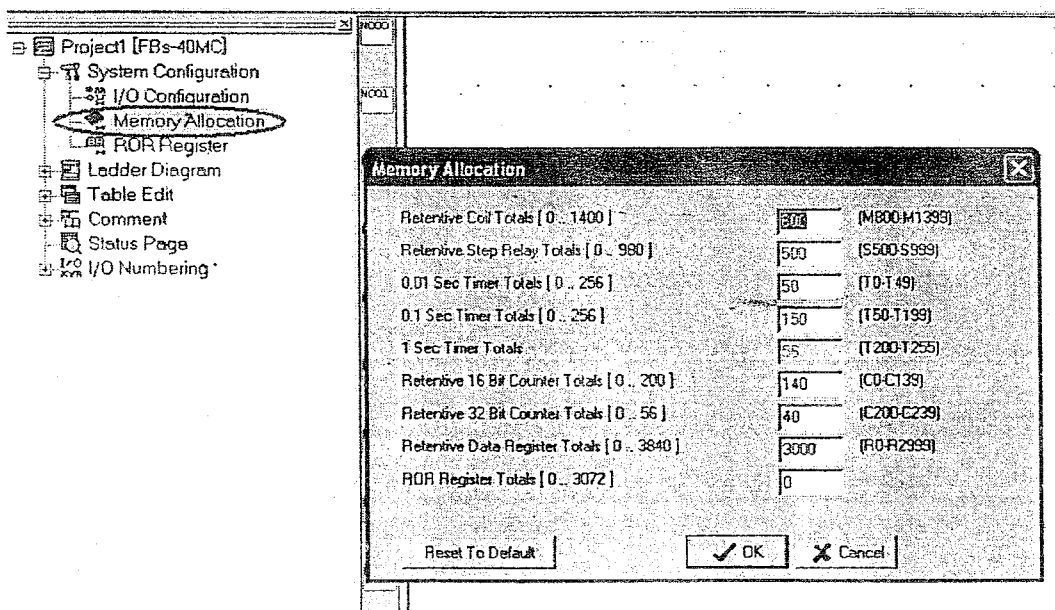
System Configuration > Memory Allocation کلیک کنید. پنجره ی شکل ۴-۲۵ ظاهر می شود.

همانطور که در ردیف اول ملاحظه می کنید ، مقدار بیش فرض رله های Retentive کمکی M ، ۶۰۰ عدد است

(M800~M1399) که این مقدار قابل تغییر است. به همین ترتیب می توانید تعداد کانترها و رجیسترهای

Retentive را هم تغییر دهید. در مورد تایمرها به بخش ۴-۹ مراجعه کنید.

پس از OK کردن ، تغییرات اعمال خواهد شد.

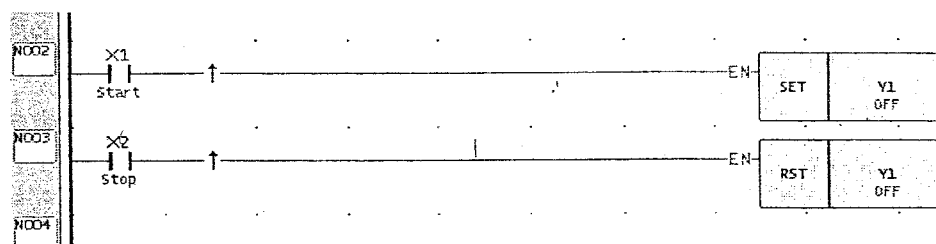


شکل ۴-۲۵

۴-۸) 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 احداث نشود رله فعال می ماند.

در شکل ۴-۲۶ با فشار دادن شستی استارت X0 رله خروجی Y1 فعال می گردد و حتی با قطع X0 فعال باقی می ماند. (این عمل در شکل ۴-۱۹ با استفاده از کنتاکت خودنگهدار انجام می گرفت) با فشار دادن شستی استاپ X1 رله خروجی Y1 reset شده و غیر فعال می گردد.



شکل ۴-۲۶

در این PLC هم چنین، بلوک های **S** set و **R** 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، ۱ می شود.

هنگام برنامه scan اگر هر دو بوبین set و reset فعال شوند، دستوری که بعد از دیگری در برنامه قرار می گیرد کنترل را به دست می گیرد زیرا PLC در پایان هر scan، نتایج حاصله را در خروجی قرار می دهد. یکی از کاربرد های دستور set و reset برقرار کردن 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 فرایند

۴-۹) 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، هم چنین مقدار زمان اندازه گیری شده به صورت عدد در خانه های حافظه PLC

ذخیره می شود و در هر لحظه قابل دستیابی می باشد. در این فصل با چگونگی استفاده از تایمرها در داخل برنامه آشنا خواهید شد.

۴-۹-۱) نحوه ی عملکرد

FATEK PLC دارای ۲۵۶ تایمر ۳ دقت زمانی می باشد:

۱ ثانیه ، ۰.۱ ثانیه و ۰.۰۱ ثانیه

تایمر ها به طور پیش فرض ، به صورت زیر تنظیم شده اند :

T0~T49 : 0.01s

T50~T199 : 0.1s

T200~T255 : 1s

که می توان این تنظیم را به دلخواه تغییر داد. برای تغییر در درخت پروژه ، به روی گزینه ی

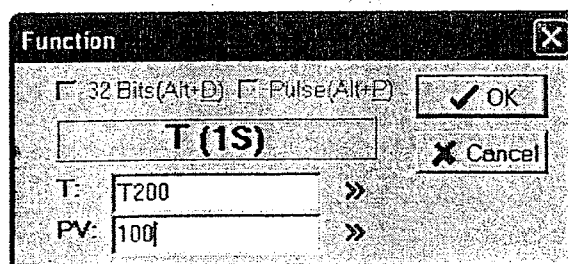
System Configuration > Memory Allocation کلیک کنید. پنجره ی شکل ۴-۲۵ ظاهر می شود.

در ردیف سوم ، چهارم و پنجم می توانید مقادیر پیش فرض تایمر ها را تغییر دهید.

تایمر روی یک تعداد اولیه تنظیم می شود و هنگامی که شمارش به این تعداد تنظیمی رسید کنتاکت خروجی شمارنده تغییر وضعیت می دهد.

برای استفاده از تایمر ، گزینه ی **T** را از نوار ایماها انتخاب کرده و به روی محیط برنامه نویسی در محل مورد نظر خود کلیک کنید.

پنجره ی شکل ۴-۲۷ ظاهر می شود. در گزینه ی T ، شماره ی تایمر با دقت دلخواه و در PV ، مقدار پیش تنظیم زمان را وارد کرده و OK کنید.



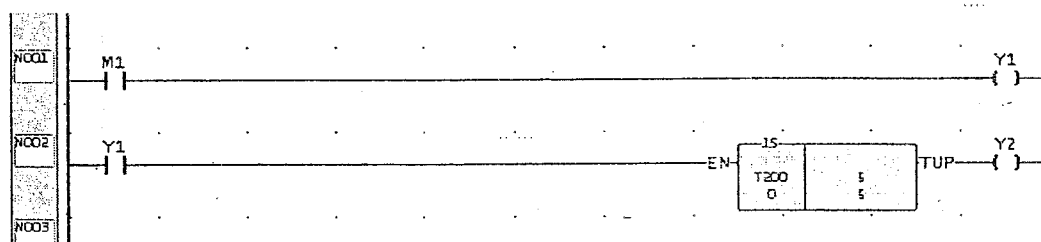
شکل ۴-۲۷

محاسبه ی زمان تایمر بدین صورت خواهد بود: $PV \times$ ادقت تایمر

هرگاه ورودی 'EN' تایمر فعال شود، پس از سپری شدن زمان تایمر، خروجی 'TUP' فعال خواهد شد.
اگر $M1957=0$ مقدار جاری تایمر (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.

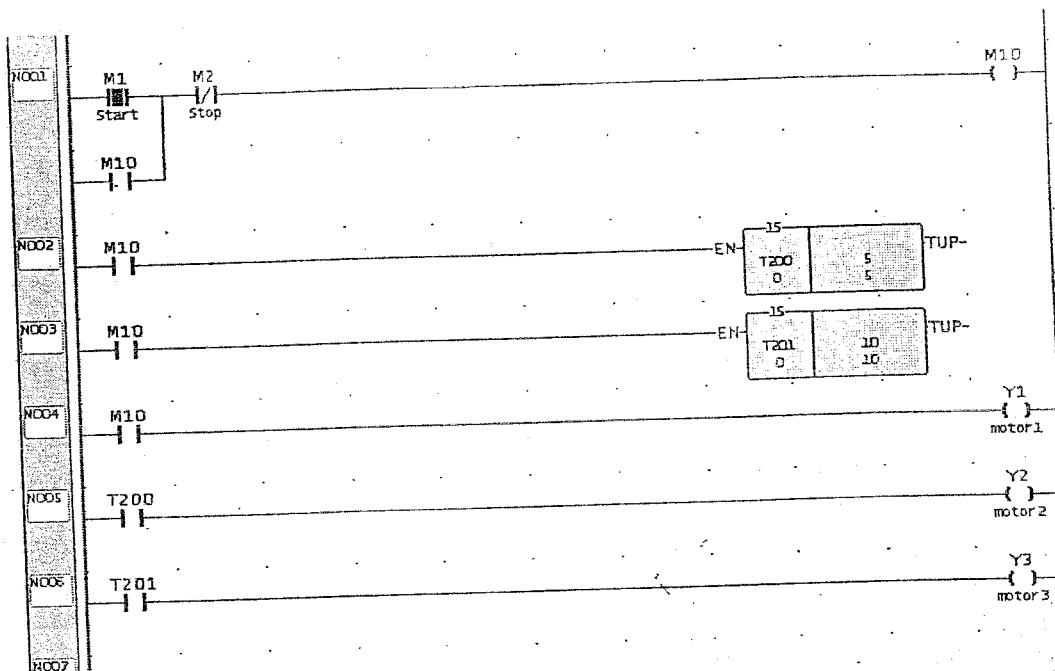
۴-۹-۲) زمان بندی

در شکل ۴-۲۸ با فعال شدن ورودی M1 رله ی خروجی Y1 تایمر تحریک شده و بعد از زمان 5sec کنتاکت تایمر تغییر وضعیت می دهد و رله ی خروجی Y2 فعال می گردد.
بنابراین خروجی Y2، ۵ ثانیه پس از فعال شدن خروجی Y1 فعال می شود.



شکل ۴-۲۸

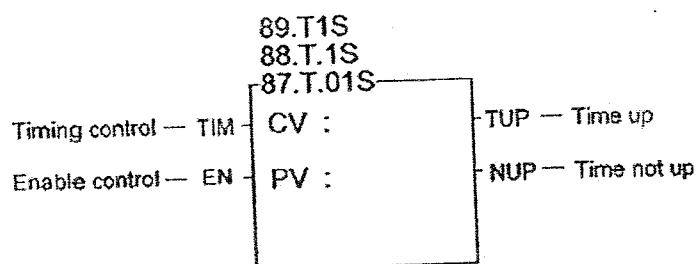
در شکل ۴-۲۹ با زدن شستی استارت، رله ی داخلی M10 فعال می شود و تحریک باقی می ماند در نتیجه تایمرهای T200 و T201 فعال می شوند و موتور ۱ نیز همزمان روشن می گردد. بعد از گذشت زمان T1 موتور ۲ و بعد از گذشت زمان T2 موتور ۳ روشن می شوند و با زدن شستی استاپ در هر مرحله می توان هر سه موتور را خاموش نمود.



شکل ۴-۲۹

۳-۹-۴) زمان‌های طولانی

تایمرهایی که تاکنون گفته شد ۱۶ بیتی اند بنابراین مقدار PV آنها نمی‌تواند بیش از ۲۷۶۷۰ باشد. برای زمان‌های طولانی‌تر، از توابع ۸۷، ۸۸ و ۸۹ استفاده می‌شود که اگر تک 32Bits آن زده شود، PV آن می‌تواند تا ۲۷۶۸۳۶۴۷ اضافه شود. (شکل ۴-۳۰)



شکل ۴-۳۰

این تابع برخلاف تایمر ساده، قابلیت تکه داشتن زمان را دارد.

در این تابع وقتی ورودی $TIM=1$ ، مانند تایمر ساده عمل می‌کند، اما اگر $TIM=0$ باشد، زمان اندازه گیری شده پاک نمی‌شود. وقتی TIM مجدداً ۱ شود، محاسبه‌ی زمان از ادامه‌ی آخرین باری که تکه داشته شده است ادامه پیدا می‌کند.

اگر تایمر احتیاج به اری ست شدن داشت، 'EN' را کنید.

این تابع دو خروجی دارد:

'TUP' که بعد از اتمام محاسبه ی زمان، ۱ می شود و

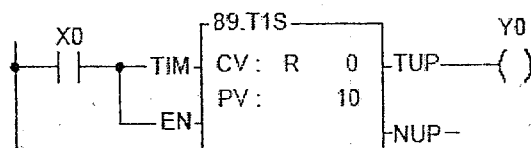
'NUP' که وقتی 'TUP' صفر است، ۱ می شود.

می توانید از ترکیب ورودی ها و خروجی ها برای ایجاد تایمرها با کارایی های مختلف استفاده کنید. (on-delay،

off-delay، ...)

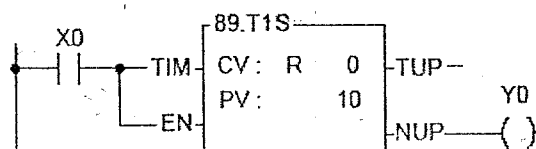
به عنوان مثال:

- وقتی X0، ON شود، بعد از ۱۰ ثانیه، Y0، ON می شود. (شکل ۴-۳۱)



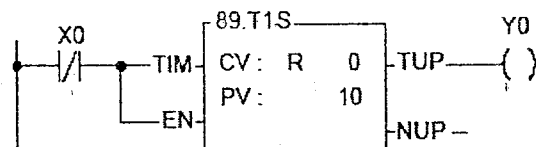
شکل ۴-۳۱

- Y0 به طور عادی ON است، وقتی X0، ON شود، بعد از ۱۰ ثانیه، Y0، OFF می شود. (شکل ۴-۳۲)



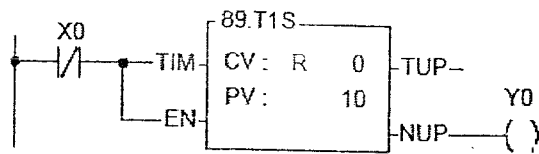
شکل ۴-۳۲

- Y0 به طور عادی OFF است، وقتی X0، OFF شود، بعد از ۱۰ ثانیه، Y0، ON می شود. (شکل ۴-۳۳)



شکل ۴-۳۳

- Y0 به طور عادی ON است، وقتی X0، OFF شود، بعد از ۱۰ ثانیه، Y0، OFF می شود. (شکل ۴-۳۴)




شکل ۴-۳۴

۴-۱۰) 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 به دلیل کاربردهای فراوان در تمامی انواع PLC موجود می باشند. حتی تایمر که در فصل قبل مورد بررسی قرار گرفت در واقع یک نوع شمارنده به حساب می آید، که پالس های ساعت متصل به پردازنده را می شمارد. در این فصل انواع شمارنده ها و کاربرد آن ها در صنعت مورد بررسی قرار خواهد گرفت.

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 (Up Counter) و یا کاهشی (Down Counter) می باشند. در نوع افزایشی عمل شمارش از صفر شروع می شود و هر بار یک شماره به شماره ی 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.

۴-۱۰-۱) شمارنده ی معمولی

در FATEK PLC، شمارنده به صورت یک بلوک در نظر گرفته می شود که از طریق دکمه ی  در نوار

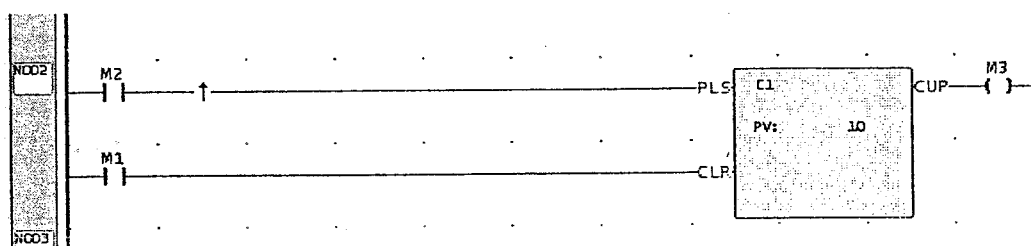
المان ها قابل دسترس می باشد. شکل ۴-۳۵. این PLC دارای :

۲۰ کانتر ۱۶ بیت (C0~C199) با قابلیت شمارش تا ۳۲۷۶۷ و

۵۶ کانتر ۲۲ بیت (C200~C255) با قابلیت شمارش تا ۲۱۴۷۴۸۳۶۴۷.

برای همین با بودن (Retentive) کانتر ها، در درخت پروژه به گزینه ی

System Configuration > Memory Allocation مراجعه کنید.



شکل ۴-۳۵

در C، شماره ی کانتر مورد نظر وارد شده و PV جهت وارد کردن تعداد پیش تنظیم اولیه به کار می رود. ورودی PLS برای شمارش پالس ورودی و CLR جهت reset کردن شمارنده به کار می روند و نهایتاً خروجی CUP هر گاه که شمارش به مقدار PV برسد، روشن می شود. در شکل ۴-۳۵ با یک بار فعال شدن M1 شمارنده reset می گردد و با هر بار فعال شدن M2 یک شمارنده به شمارنده اضافه می شود و هنگامی که شمارش به تعداد تنظیم شده روی شمارنده که در این جا عدد ۱۰ می باشد رسیده، شمارنده تغییر وضعیت داده و رله ی خروجی M3 فعال می شود.

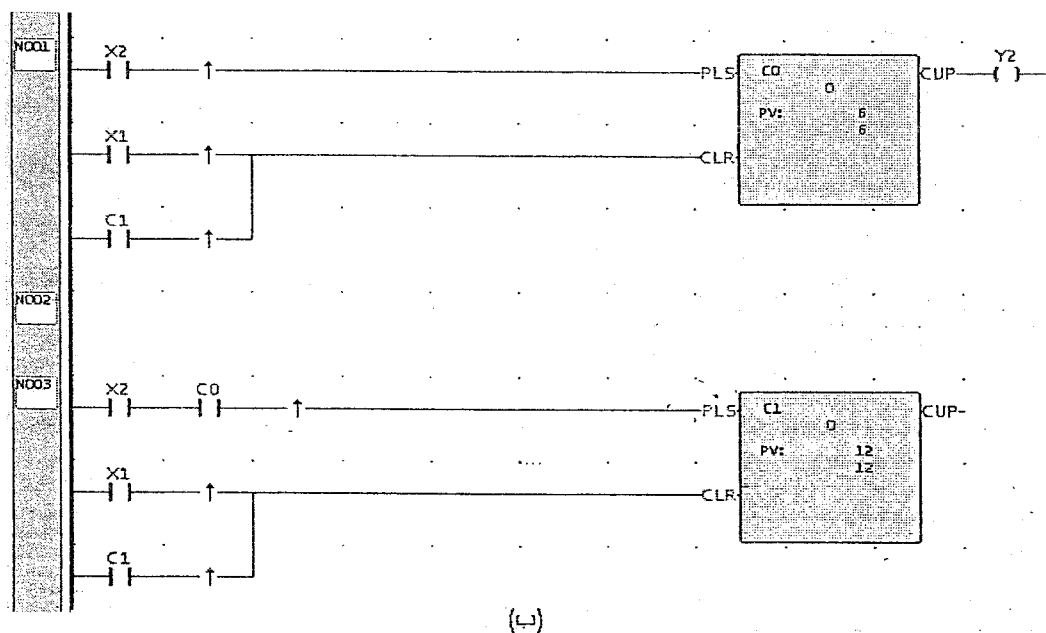
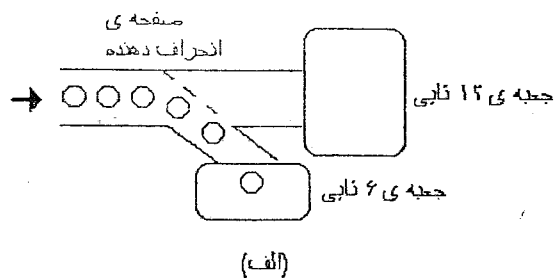
۲-۱۰-۴) کاربرد شمارنده ها

در شکل ۴-۳۶-الف، ابتدا ۶ عدد قوطی کنسرو توسط صفحه ی هدایت کننده به سمت بسته بندی ۶ تایی و سپس ۱۲ عدد از آن ها به سمت بسته بندی ۱۲ تایی هدایت می شوند. شکل ۴-۳۶-ب، دیاگرام نردبانی این خط تولید را در FATEK-PLC نمایش می دهد.

شستی X1 هر دو شمارنده را reset کرده و نوار نقاله ی خط تولید را روشن می نماید. X2 کنتاکت سنسور نوری (فتوسل) می باشد که عبور هر قوطی را تشخیص می دهد.

شمارنده ی C0 پس از دریافت ۶ پالس از طریق فتوسل X2 رله ی خروجی Y2 را فعال کرده تا از طریق آن فرمان لازم جهت تغییر وضعیت صفحه ی هدایت کننده صادر شود و قوطی ها به سمت بسته بندی ۱۲ تایی هدایت گردند.

در این لحظه شمارنده ی C1 شروع به شمارش ۱۲ عدد قوطی نموده و پس از اتمام شمارش هر دو شمارنده ی C0 و C1 را reset نموده و دستگاه مجدداً شروع به فعالیت در این سیکل بسته می نماید.



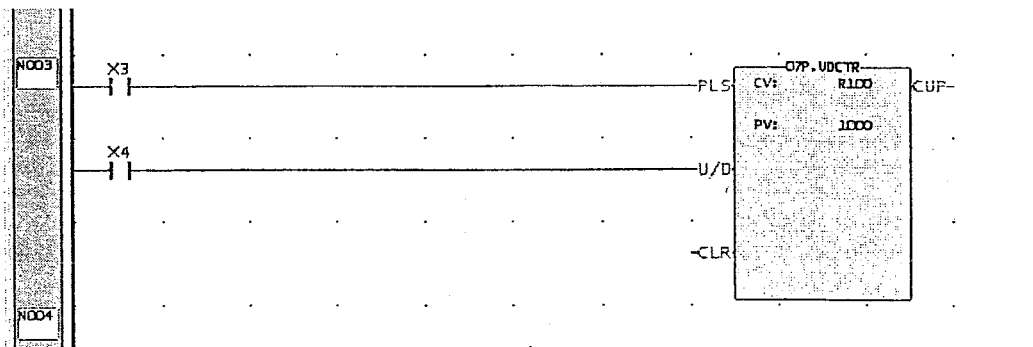
شکل ۴-۳۶

۳-۱۰-۴) شمارش همزمان به صورت افزایشی و کاهشی

کانتور معمولی تنها به صورت افزایشی شمارش می کند، برای شمارش به صورت کاهشی از تابع شماره ۷ که توانایی شمارش به صورت کاهشی و افزایشی را دارد استفاده می شود.

از یک بوبین جهت شمارش (افزایشی) (U) و (کاهشی) (D) 'U/D' و یک بوبین جهت reset کردن شمارنده به حالت اولیه خود استفاده می شود 'CLR'. هرگاه ورودی 'U/D' فعال باشد، شمارش افزایشی و هرگاه غیر فعال باشد کاهشی خواهد بود و نهایتاً خروجی 'CUP' هرگاه که شمارش به مقدار PV برسد، روشن می شود.

(شکل ۴-۳۷)

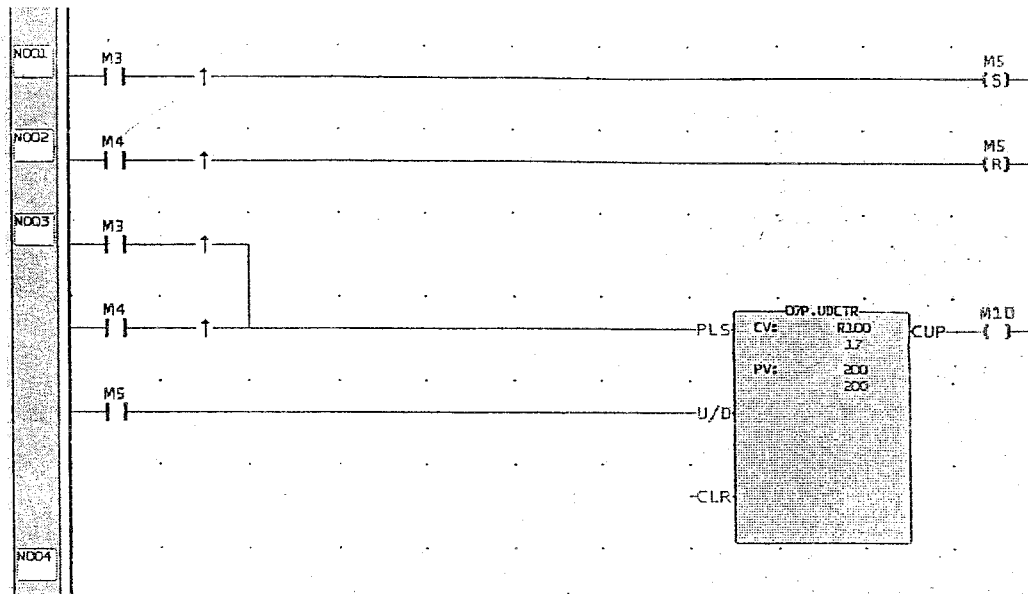


شکل ۴-۳۷

CV جهت وارد کردن تعداد اولیه و نمایش تعداد جاری و PV جهت وارد کردن مقدار نهایی استفاده می شود.

فرض کنید که می خواهیم تعداد ماشین های موجود در یک پارکینگ را بشماریم و هنگامی که این ماشین ها به تعداد معینی رسیدند علامت ظرفیت پارکینگ تکمیل است را در جلوی درب ورودی پارکینگ ظاهر نمائیم. در این صورت باید یک سنسور در جلوی درب ورودی، ورود ماشین ها را تشخیص و سنسور دیگری در جلوی درب خروجی، خروج آن ها را مشخص نماید، که سنسور اولی افزایشی و سنسور دومی کاهش عمل می کند. در شکل ۴-۳۸ دیاگرام نردبانی این برنامه را مشاهده می کنید. وصل بودن پایه ی U/D، نشان دهنده ی شمارنده ی افزایشی و قطع بودن آن نشان دهنده ی شمارنده ی کاهش می باشد.

رله ی M10 در خروجی کانتر، جهت نمایش علامت ظرفیت پارکینگ تکمیل است در HMI به کار رفته و هرگاه نتیجه ی شمارش برابر ۲۰ شود روشن می شود. X3 سنسور نوری درب ورودی و X4 سنسور نوری درب خروجی می باشد.



شکل ۴-۳۸

۴-۱۱) پردازش اعداد

برای عملیات محاسباتی و ذخیره و نمایش اعداد نیز استفاده کرد. حافظه PLC (یک بیت) را به صورت مجزا اشغال می کنند در حالی که رجیسترها مجموعه ای از بیت های (۱۶ بیت) در کنار هم می باشند (به بخش ۶-۴ رجیستر رجوع کنید) و همین امر سبب می شود که از رجیسترها جهت انجام محاسبات، ذخیره و نمایش اعداد نیز استفاده گردد.

در سیستم های کنترل عموماً لازم است تا دو عدد را با یکدیگر مقایسه کنیم یا با هم جمع نماییم. در این فصل با چگونگی پردازش اعداد توسط PLC و موارد استفاده آن در صنایع آشنا خواهید شد.

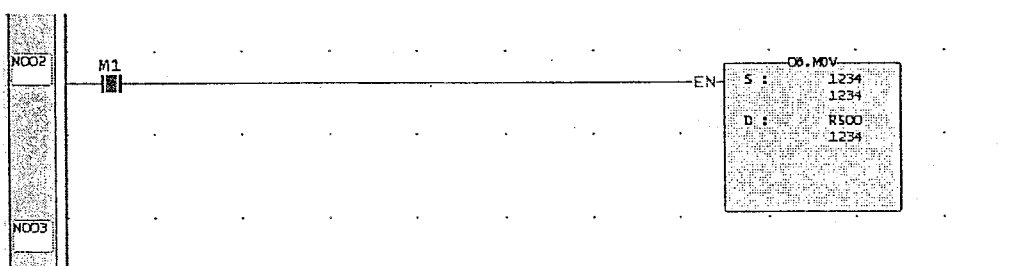
هنگامی که یک عدد وارد PLC گردید PLC جهت پردازش روی این عدد نیازمند دستورالعمل های مناسب می باشد. دستورالعمل های انتقال اطلاعات، دستورالعمل های عملیات حسابی و مقایسه ی اعداد از جمله ی این موارد می باشند.

۴-۱۱-۱) دستورالعمل انتقال اطلاعات

دستور MOV (شماره تابع : ۸) 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.

شکل ۴-۳۹ دیاگرام نردبانی مربوط به این دستورالعمل را در FATEK-PLC نشان می دهد. 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.



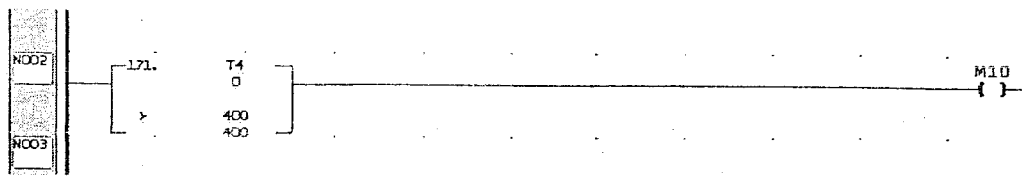
شکل ۴-۳۹

۴-۱۱-۲) دستورالعمل مقایسه اعداد

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

- [۱۷] مقایسه کلی
- [۳۷] مقایسه ناحیه ای
- [۱۷۰] مساوی
- [۱۷۱] بزرگتر
- [۱۷۲] کوچکتر
- [۱۷۳] نامساوی
- [۱۷۴] بزرگتر یا مساوی
- [۱۷۵] کوچکتر یا مساوی

در این نوع دستورالعمل دو مقدار با یکدیگر مقایسه می شوند و در صورت درست بودن شرط مقایسه، خروجی فعال می گردد. نحوه ی استفاده از دستور ۱۷ در بخش ۴-۲ (نوار المان ها) تشریح شده است. در شکل ۴-۴۰ نحوه ی استفاده از دستور "بزرگتر" را در FATEK-PLC مشاهده می کنید. هنگامی که زمان اندازه گیری شده توسط تایمر T4 از ۴۰۰ بزرگتر شود بوبین خروجی تحریک می گردد.



شکل ۴-۴۰

در شکل ۴-۴۱، نحوه ی استفاده از دستور مقایسه ناحیه ای (Zone compare) [تابع ۳۷] را مشاهده می کنید.

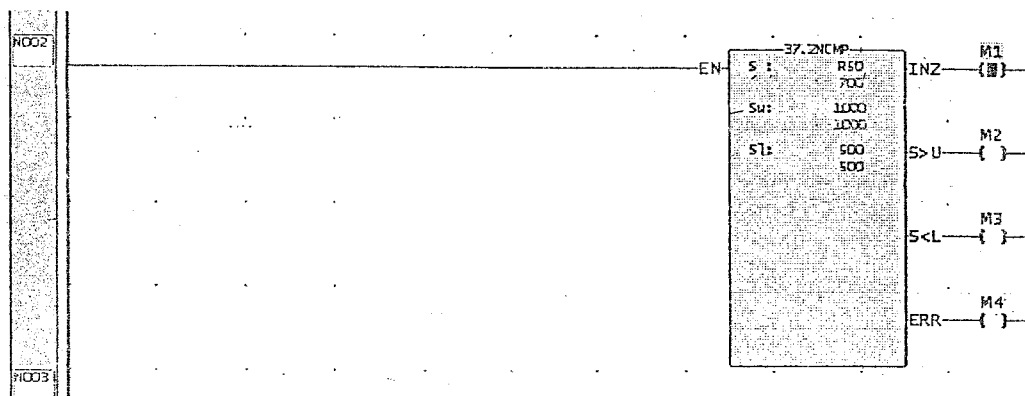
در این مثال، هرگاه مقدار R50 بین ۵۰۰ و ۱۰۰۰ باشد، خروجی "INZ" (در اینجا M1) فعال می شود.

هرگاه مقدار R50 کمتر از ۵۰۰ شود، خروجی 'S<L' (در اینجا M3) فعال می شود.

هرگاه مقدار R50 بیشتر از ۱۰۰۰ شود، خروجی 'S>U' (در اینجا M2) فعال می شود.

مقداری که در SU قرار می گیرد، باید بزرگتر از SL باشد، در غیر این صورت، خروجی 'ERR' فعال

می شود (در اینجا M4) و تابع عمل نمی کند.



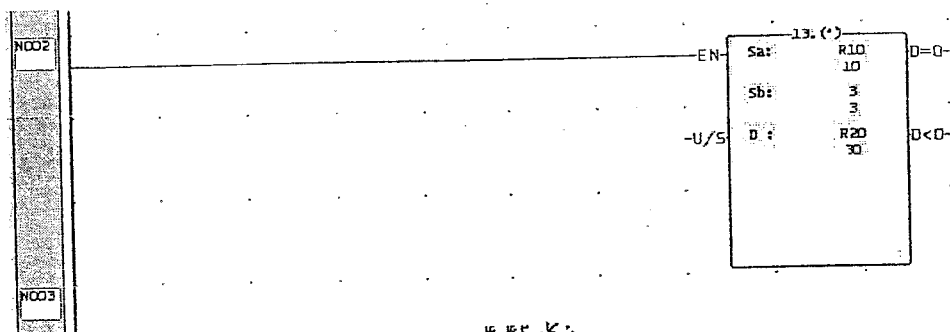
شکل ۴-۴۱

۳-۱۱-۴) دستورالعمل عملیات حسابی

در FATEK-PLC علاوه بر چهار عمل اصلی، توابع ریاضی نظیر جذر، توابع نمایی و لگاریتمی و توابع مثلثاتی

و غیره قابل محاسبه می باشند.

شکل ۴-۴۲، چگونگی انجام یک عمل ضرب در FATEK-PLC را نمایش می دهد.



شکل ۴-۴۲

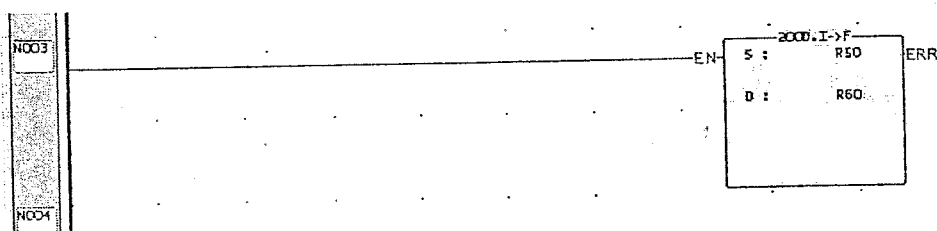
جزئیات دیگر توابع این دسته را در فصل ۶ مشاهده کنید.

۴-۱۱-۴ اعداد اعشاری

در دسته بندی های مختلف توابع FATEK، دسته ای برای اجرای عملیات به روی اعداد اعشاری یا Float وجود دارد. بنابر این اگر نیاز به اجرای عملیاتی مانند جمع، ضرب یا مقایسه بین اعداد اعشاری بود، از توابع جمع، ضرب و مقایسه ای مخصوص اعداد Float استفاده می شود.

به عنوان مثال اگر بخواهیم یک عدد صحیح را به اعشاری تبدیل کنیم، از تابع شماره ۲۰۰ استفاده می کنیم.

(شکل ۴-۴۳)



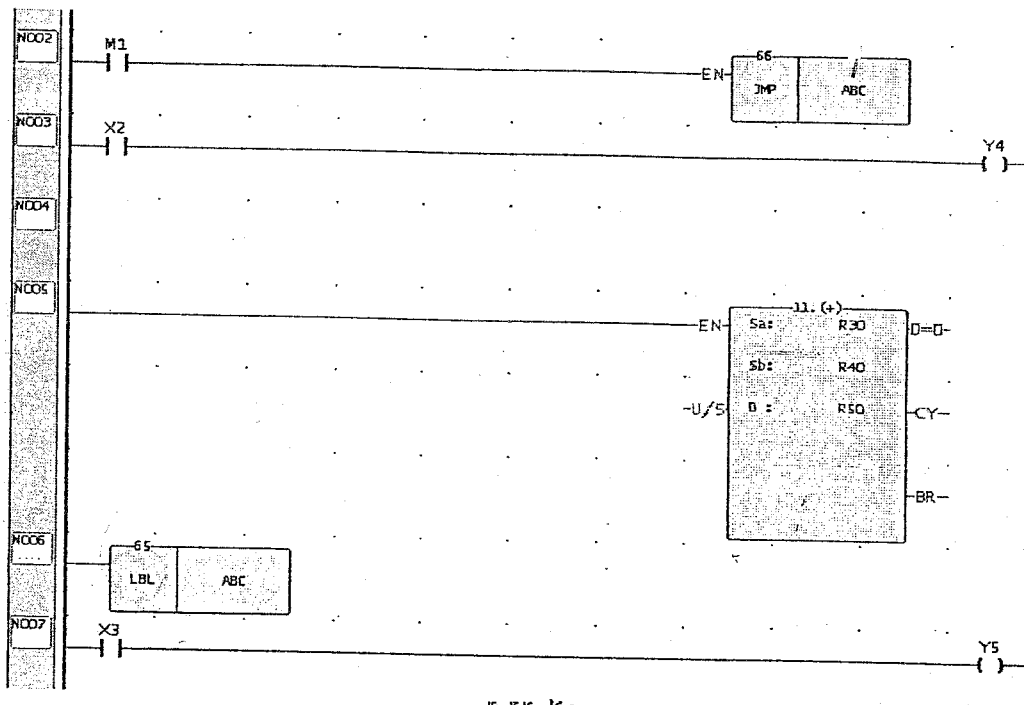
شکل ۴-۴۳

۲-۱۲ پرش (Jump) و برچسب (Label)

در صورتی که تحت شرایط خاصی بخواهیم پردازنده از روی تعدادی از خطوط برنامه بدون اجرا پرش نماید از دستور پرش (شماره تابع : ۴۴) استفاده می شود. پرش به خطی از برنامه صورت می گیرد که برچسب (Label) آن مشخص می کند و برنامه از جایی ادامه پیدا می کند که آن برچسب زده شده است.

شماره تابع Label: ۴۵

در شکل ۴-۴۴ با وصل شدن کنتاکت M1 تابع Jump با برچسب ABC فعال شده و پردازنده مستقیماً به خط Y و برچسب ABC می رود. در صورتی که M1 وصل نباشد خطوط برنامه به ترتیب از ابتدا تا انتها اجرا می گردند.



شکل ۴-۴۴

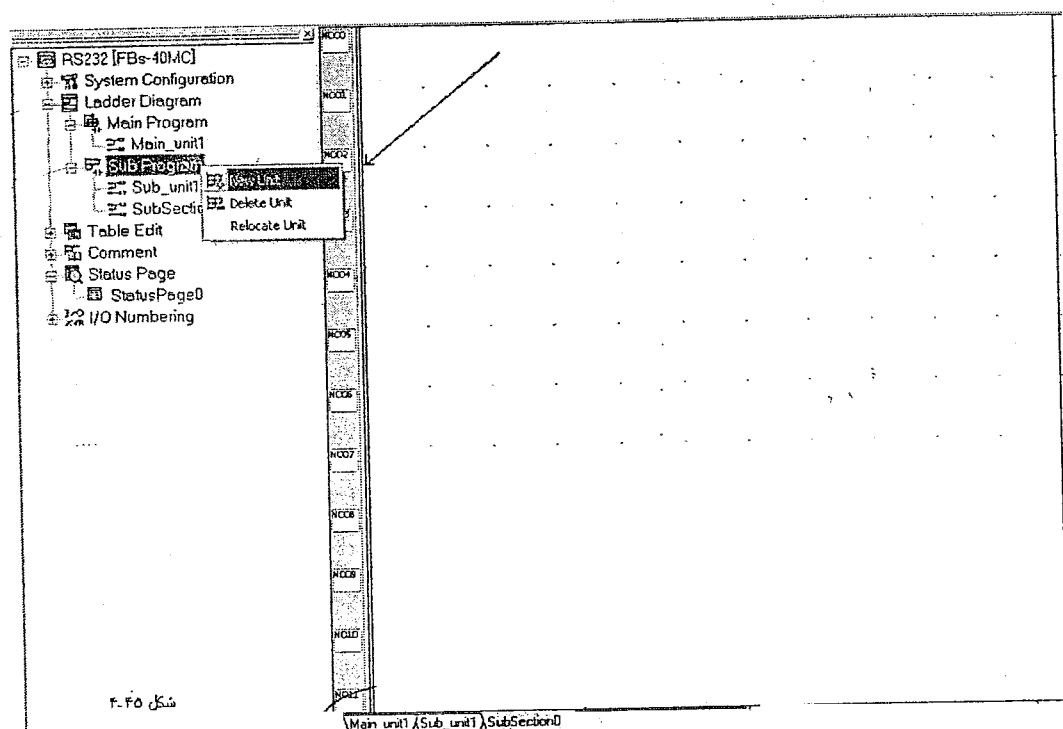
پرش فقط در برنامه اصلی یا زیربرنامه ها صورت می گیرد و باین فانکشن از برنامه اصلی به زیر برنامه و یا برعکس نمی توان پرش کرد.

در برنامه نویسی PLC انجام یک پرش در داخل پرش دیگر مجاز می باشد. هم چنین دستور پرش به سمت بالای برنامه بایستی با دقت مورد استفاده قرار گیرد زیرا ممکن است استفاده ی غیر صحیح از این دستور منجر به طولانی شدن زمان یک scan و در نتیجه خطای Watch Dog Timer (تایمر سگ نگهبان) گردد.

۴-۱۳ فراخوانی زیر برنامه (CALL)

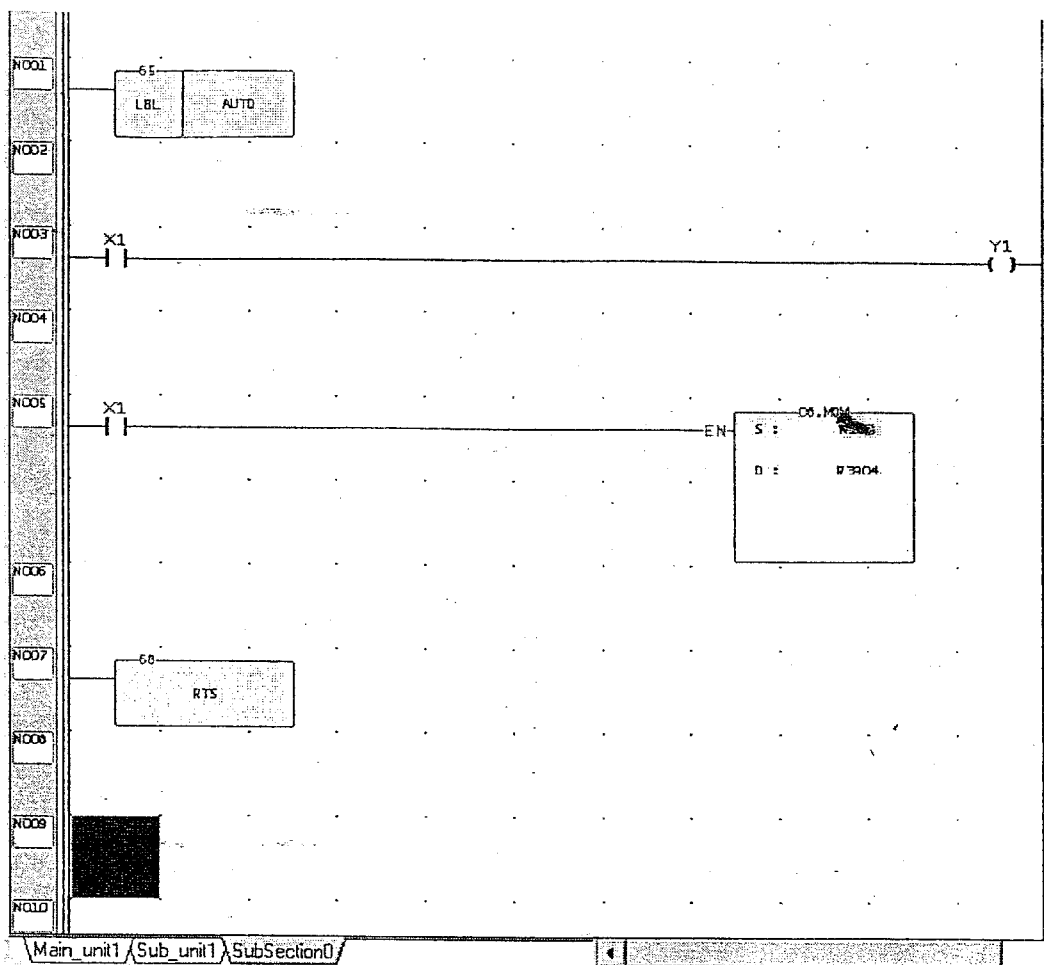
گاهی احتیاج است که یک قسمت از برنامه تنها هنگامی که به آن نیاز است اجرا شود. برای این منظور این قسمت از برنامه را در زیر برنامه می نویسیم. برای ایجاد زیربرنامه در درخت پروژه به قسمت

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 ی جدید می نویسیم و پس از OK کردن زیر برنامه ی جدید ساخته می شود. جا به جایی محیط برنامه نویسی از محیط اصلی به محیط زیر برنامه ، از طریق تب های کشویی پایین محیط برنامه نویسی صورت می گیرد. (شکل ۴۵-۴)



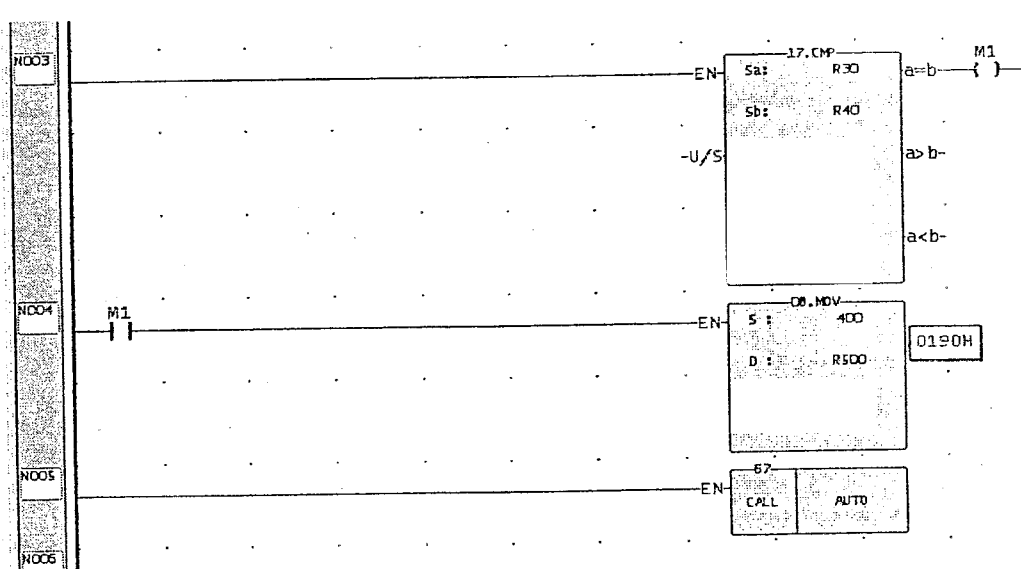
حال برای اجرای زیر برنامه ، برجستی (Label) در ابتدای زیر برنامه گذاشته و در انتهای آن نیز تابع شماره

۶۸ Return From Subroutine = RTS را می گذاریم. (شکل ۴۶-۴)



شکل ۴-۴۶

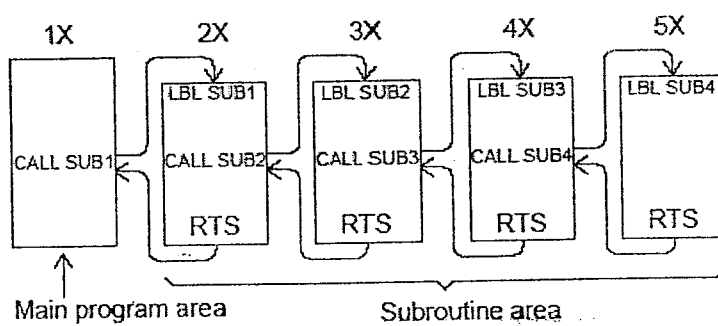
برای اجرا شدن این زیر برنامه ، باید برجست آن از طریق تابع CALL (شماره ۶۷) فراخوانی شود .
 در شکل ۴-۴۷ ، هرگاه ورودی تابع CALL فعال شود ، زیر برنامه با برجست AUTO فراخوانی و اجرا می شود .



شکل ۴-۴۷

وقتی برنامه اصلی از طریق CALL یک زیربرنامه را فراخوانی می کند، آن زیربرنامه نیز می تواند زیربرنامه

های دیگر را فراخوانی کند و این کار تا ۵ مرحله می تواند انجام یابد. (شکل ۴-۴۸)



شکل ۴-۴۸

مسائل فصل ۴

۴-۱) در FATEK-PLC به زبان Ladder، برنامه ای بنویسید که در آن یک چراغ در صورت فشار کلید

تست لامپ و یا در صورت روشن شدن یک موتور و تامین فشار کافی روشن گردد.

۴-۲) در چهار طرف یک ماشین پرس برای حفظ ایمنی افراد چهار سنسور نوری قرار گرفته است که در صورت عمل کردن هر کدام از این سنسورها ماشین متوقف می شود. این برنامه را به زبان Ladder بنویسید.

۴-۳) دیاگرام نردبانی طراحی نمائید که با وصل ورودی، خروجی Y1 فعال شود و تنها ۵ ثانیه فعال بماند.

۴-۴) دیاگرام نردبانی طراحی نمائید که با فشار یک شستی لامپ روشن شده و با برداشتن دست از شستی همچنان روشن بماند و با فشار دوباره ی شستی لامپ خاموش شود و با برداشتن دست از روی شستی خاموش بماند.

۴-۵) دیاگرام نردبانی طراحی نمائید که فرمان یک روبات که از چهار طرف با سنسورهای نوری احاطه شده است را به صورتی صادر نماید که هنگام تشخیص ورود هرگونه جسم خارجی به محدوده ی کار روبات از طریق این سنسورها، سیستم متوقف شده و آلارم فعال شود و تا زمانی که مانع، خارج نشده، امکان آغاز دوباره ی فعالیت روبات وجود نداشته باشد.

فصل پنجم

طراحی، آزمایش و اشکال یابی برنامه

در این فصل در ارتباط با نحوه ی طراحی برنامه صحبت خواهیم کرد و در ادامه به ذکر چند مثال کاربردی می پردازیم.

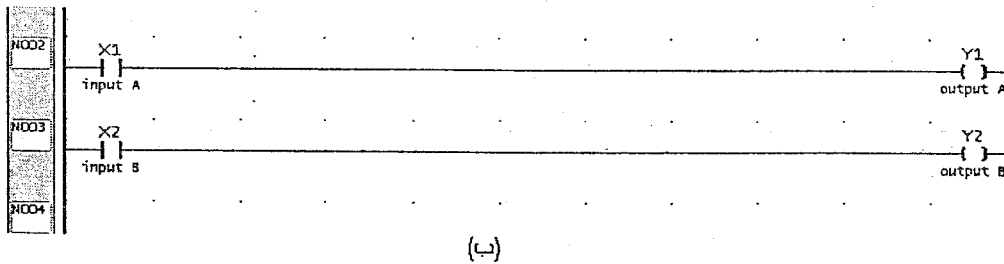
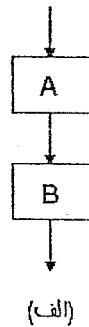
۵-۱) طراحی برنامه

جهت طراحی یک برنامه لازم است مراحل ذیل صورت گیرد.

- ۱- ورودی ها و خروجی ها سیستم کنترل را دقیقاً تعیین نمائید.
 - ۲- الگوریتم برنامه را به صورت فلوچارت (flow chart) و یا دستورالعمل های برنامه نویسی کامپیوتر نظیر WHILE-DO, BEGIN, DO, END, IF-THEN-ELSE معین نمائید. (توجه شود که مهمترین مرحله از طراحی برنامه مرحله ی فوق است و لازم است این مرحله به دقت انجام و طی چندین مرحله اصلاح گردد).
 - ۳- ورودی ها و خروجی های موجود را به ادرس های مناسب ورودی ها و خروجی ها I/O نسبت دهید. سپس الگوریتم برنامه را به زبان برنامه نویسی PLC تبدیل نمائید.
 - ۴- برنامه به دست آمده را وارد PLC نموده و مراحل انجام کار را آزمایش و رفع اشکال نمائید. (در حالت On-Line)
 - ۵- برنامه را به صورت مناسب بایگانی نمائید تا به آسانی قابل درک و تغییر باشد.
- هنگامی که فلوچارت یک برنامه را رسم می نمائید خواهید دید که هر برنامه از کنار هم قرار دادن چند الگوریتم ساده تشکیل می گردد. در ادامه این بخش با ذکر چند مثال از این گونه الگوریتم ها، با نحوه ترسیم فلوچارت آشنا خواهید شد.

۱- الگوریتم ترتیبی

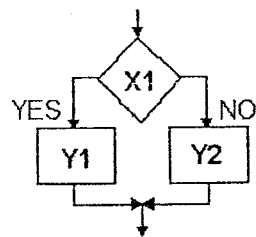
فرایند ترتیبی بدین صورت است که انجام یک فعالیت (فرآیند B) بعد از انجام فرآیند دیگر (فرایند A) روی می دهد. شکل ۵-۱ الف فلوجارت این فرآیند و شکل ۵-۱ ب. دیاگرام نردبانی معادل این فرآیند را نشان می دهد.



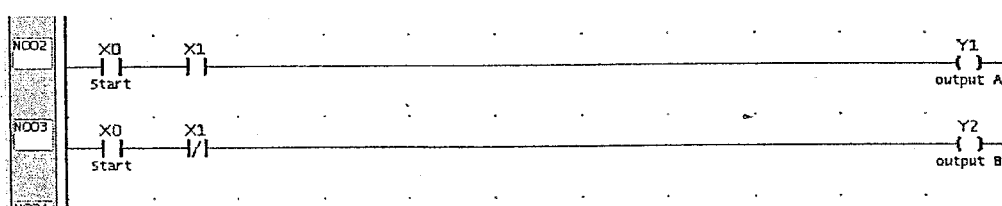
شکل ۵-۱ . الف) فلوجارت ترتیبی ب) دیاگرام نردبانی الگوریتم ترتیبی

۲- الگوریتم شرطی

شکل ۵-۲ الف فلوجارت مربوط به یک فرایند شرطی و شکل ۵-۲ ب دیاگرام نردبانی معادل این فرآیند شرطی را نشان می دهد.



(الف)

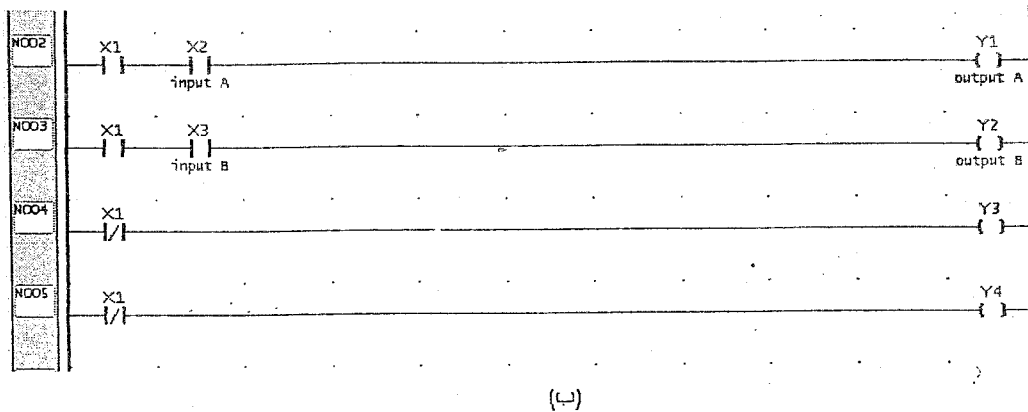
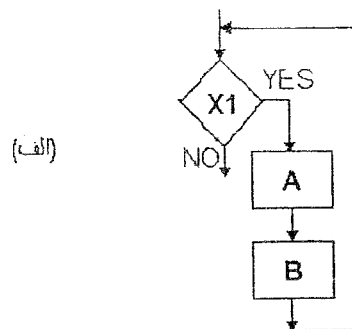


(ب)

شکل ۵-۲. الف) فلوجارت شرطی ب) دیاگرام نردبانی الگوریتم شرطی

۳- الگوریتم حلقه

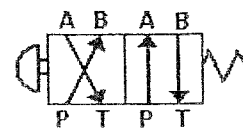
حلقه، اجرای بخش هایی از یک برنامه به صورت مکرر می باشد. شکل ۳-۵. الف فلوجارت مربوط به یک حلقه شرطی را نمایش می دهد و شکل ۳-۵. ب دیاگرام نردبانی معادل آن را نشان می دهد.



شکل ۵-۲. الف) فلوجارت حلقه ب) الگوریتم حلقه

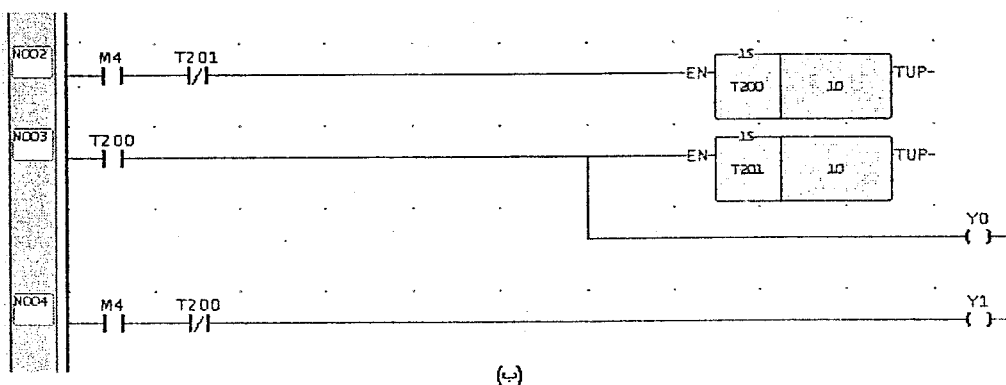
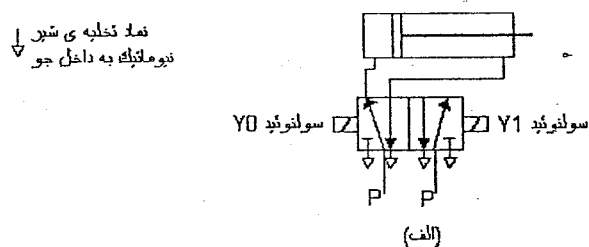
۵-۲) مسائل حل شده

۱- شکل ۵-۴ یک ولو ۴/۲ را نشان می دهد. هنگامی که شستی فشار داده شود دهانه ی A به T و دهانه ی P به B وصل می شود (T نشان دهنده ی خروجی به هوای آزاد در سیستم های نیوماتیکی و یا بازگشت روغن هیدرولیک به مخزن می باشد و P نشان دهنده ی وصل به خط فشار روغن یا هوا است). با برداشتن دست از روی شستی، فنر ولو را به حالت اولیه ی خود بازمی گرداند یعنی دهانه ی P به A و دهانه ی B به T وصل می شود.



شکل ۵-۴

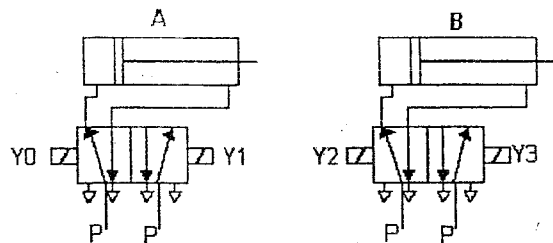
در شکل ۵-۵ زمانی که M4 وصل شود خروجی Y1 تحریک می گردد. با فعال شدن سلونوئید Y1 فشار سیال به سمت راست سیلندر وارد شده و نهایتاً پیستون به سمت چپ حرکت می کند. بعد از گذشت ۱۰ ثانیه، کنتاکت های تایمر T0 تغییر وضعیت داده و باعث غیرفعال شدن Y1 و فعال شدن Y0 می شوند، با فعال شدن Y0، فشار روغن به سمت چپ سیلندر وارد شده و پیستون به سمت راست حرکت می کند بعد از گذشت ۱۰ ثانیه این بار کنتاکت های تایمر T1 فعال شده و پیستون مجدداً به جای اولیه خود باز می گردد و تا زمانی که M4 فعال باشد این عمل به طور پیوسته انجام می گردد.



شکل ۵-۵. حرکت پیستون به صورت متناوب

۲- در شکل ۵-۶ سه پیستون A, B باید به ترتیب زیر فعال گردند:

ابتدا پیستون A به سمت راست و سپس به سمت چپ حرکت کند. در مرحله ی بعد پیستون B به سمت راست و بعد به سمت چپ حرکت کند که می توان ترتیب انجام این عملیات را به صورت ذیل نیز نمایش داد.

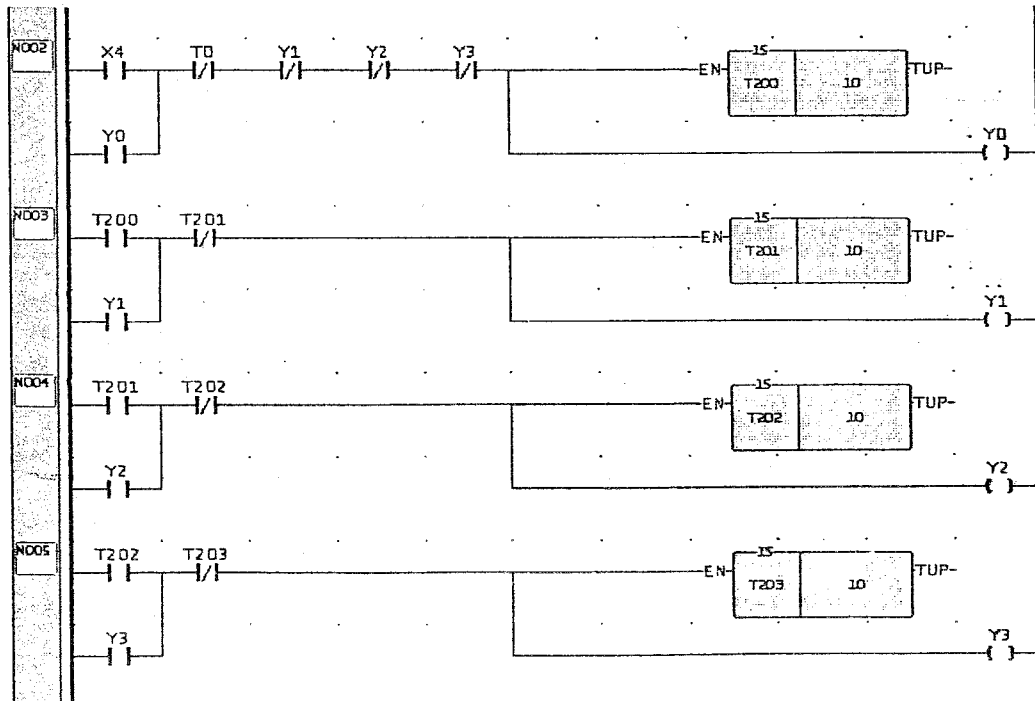


شکل ۵-۶

دیاگرام نردبانی برنامه فوق در شکل ۵-۷ قابل مشاهده است.

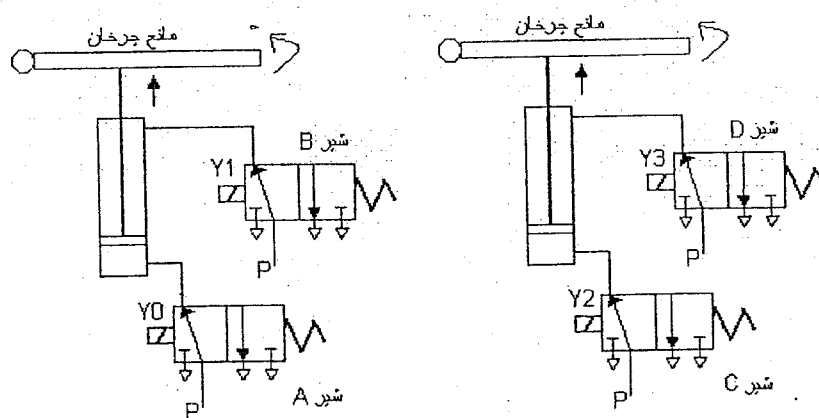
با فعال شدن X4، رله ی خروجی Y0 به همراه تایمر T200 فعال شده و رله ی خروجی از طریق کنتاکت خودنگهدار فعال باقی می ماند. در این حالت پیستون A به سمت راست حرکت کرده اما بعد از گذشت ۱۰ ثانیه کنتاکت های T200 تغییر وضعیت داده و باعث غیرفعال شدن رله ی خروجی Y0 به همراه تایمر T200 شده و هم زمان رله ی خروجی Y1 به همراه تایمر T201 را فعال می کند و در این حالت پیستون A به سمت چپ حرکت می کند و بعد از گذشت ده ثانیه، فرمان فوق جهت حرکت پیستون B به

سمت راست و سپس به سمت چپ از طریق تایمرهای T202 و T203 صادر می شود



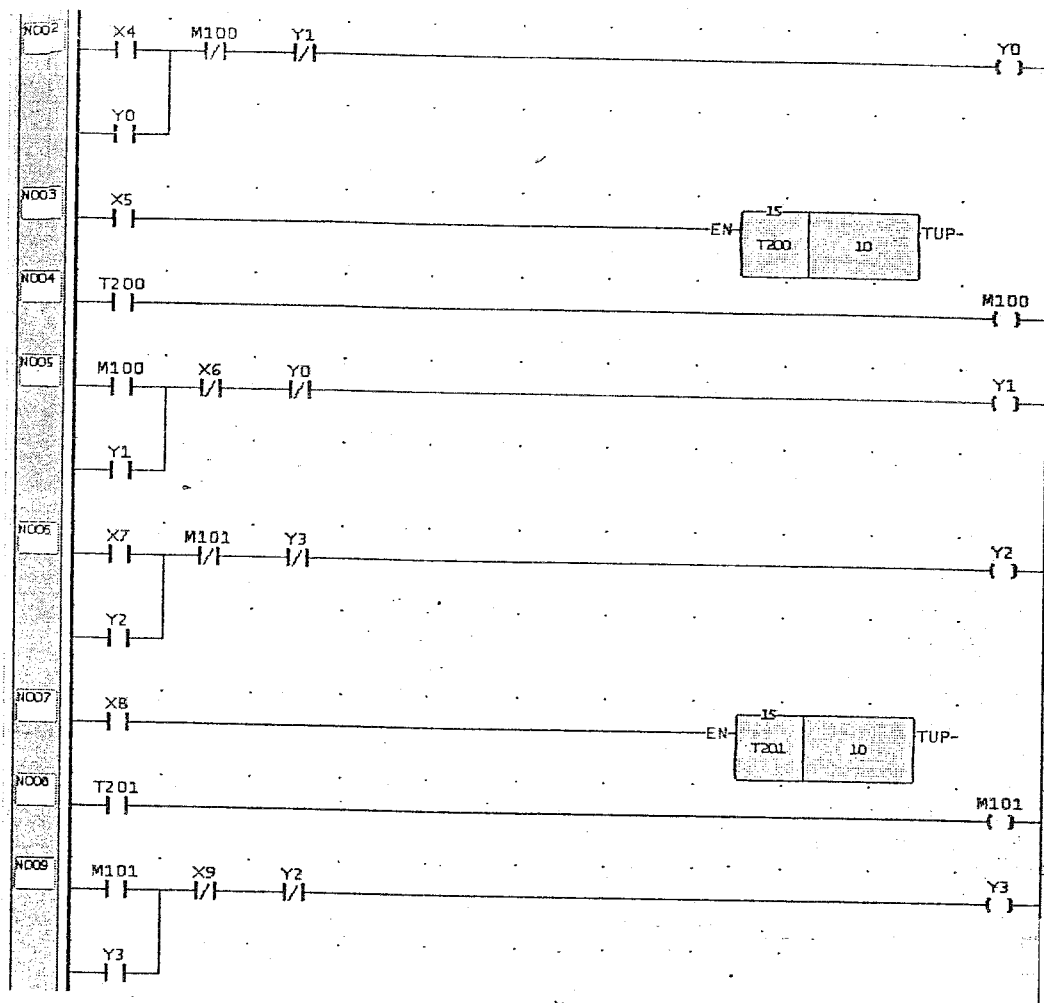
۳ - جهت باز و بسته نمودن درب یک پارکینگ از چهار شیر نیوماتیکی استفاده شده است. زمانی که پول مناسب در داخل صندوق انداخته شود، درب ورودی باز و هنگامی که سیستم متوجه وجود یک ماشین پشت درب خروجی می شود درب خروجی باز می گردد.

همان طور که از شکل ۸-۵ مشخص است ، شیرها با تحریک شدن سلونوئید، تغییر وضعیت می دهند و با غیرفعال شدن همان سلونوئید به توسط نیروی فنر به حالت اولیه خود باز می گردند.



شکل ۵-۸

شیرهای نیوماتیک A و B جهت فرمان باز و بسته شدن درب ورودی و شیرهای C و D جهت فرمان باز و بسته شدن درب خروجی پارکینگ استفاده می شوند. شکل ۹-۵ دیاگرام نردبانی این مثال را نشان می دهد. با انداختن سکه درون صندوق X4 وصل می شود و رله ی Y0 را که دارای خودنگهدار می باشد فعال می کند و با فعال شدن این رله شیر A تحریک شده و باعث باز شدن درب ورودی می گردد. بعد از باز شدن کامل درب، میکروسوییچ X5 وصل می شود و ۱۰ ثانیه بعد سلونوئید Y0 غیرفعال می گردد و نیروی فنر شیر را به حالت اولیه بر می گرداند و در همین حین رله ی Y1 تحریک شده و باعث بسته شدن درب ورودی به وسیله ی فرمان شیر B روی سیلندر می گردد. پس از بسته شدن درب ، میکروسوییچ X6 وصل می شود. مدار فوق دقیقاً برای درب خروجی تکرار می گردد با این تفاوت که فرمان باز شدن درب خروجی از طریق یک سنسور نوری X7 ، روی شیر C آمده و فرمان بسته شدن درب نیز به وسیله ی میکروسوییچ X8 بعد از زمان ۱۰ ثانیه روی شیر D می آید.



شکل ۵-۶

۴- در یک خط تولید توسط یک نوار نقاله بطری های خالی نوشابه پرگشته، و درپوش آن ها گذاشته

می شود و سپس در بسته بندی های چهارتایی آماده می کردند.

فصل ششم

دیگر توابع FATEK

PLC

16.DECREMENT

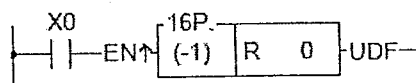


هرگاه 'EN' از ۰ به ۱ تغییر کند، از مقدار رجیستر D، ۱ کم می شود. (۱- می شود)

اگر این کاهش باعث از رنج (range) خارج شدن مقدار D شود، 'UDF' فعال می شود و مقدار D مثبت

می شود.

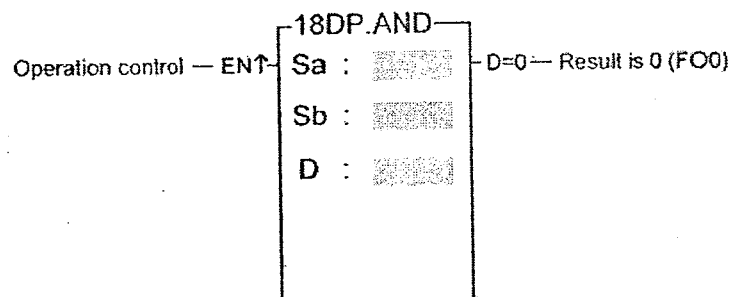
مثال:



⇓ X0 = 1

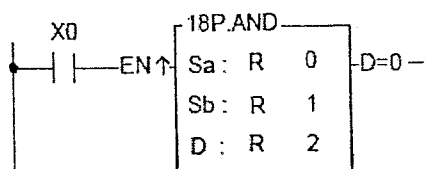


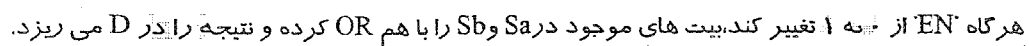
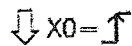
18.LOGICAL AND



هرگاه 'EN' از ۰ به ۱ تغییر کند، بیت های موجود در Sa و Sb را با هم AND کرده و نتیجه را در D می ریزد.

مثال:

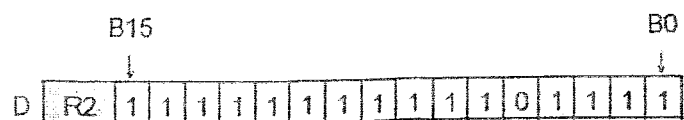
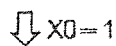




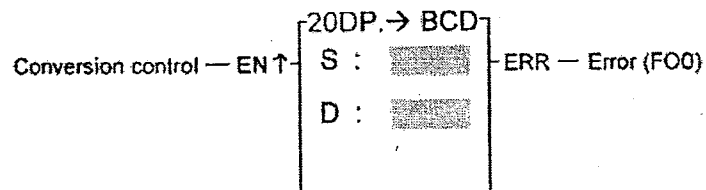
19. OR

Sa :	R	0
Sb :	R	1
D :	R	2

D=0



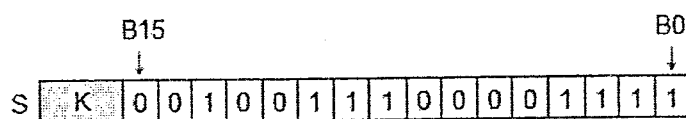
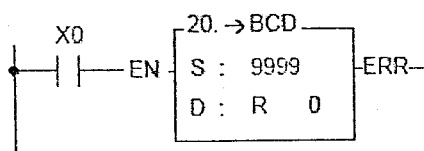
20. BIN TO BCD CONVERSION



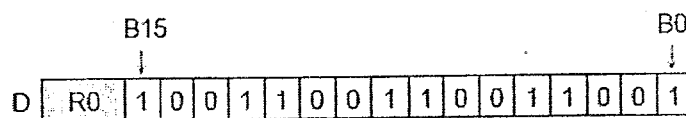
هرگاه 'EN' از ۰ به ۱ تغییر کند، داده های موجود در S را که به صورت باینری است، به صورت BCD درآورده و در D می ریزد.

اگر داده S در BCD Range نباشد، 'ERR' فعال می شود و اطلاعات قبلی D بدون تغییر باقی می ماند.

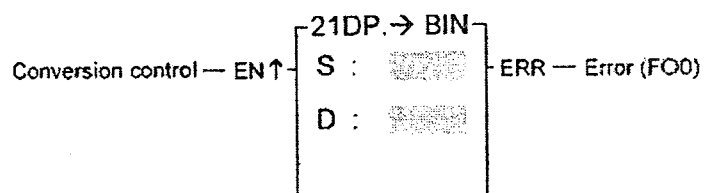
مثال:



↓ X0=1



21. BCD TO BIN CONVERSION

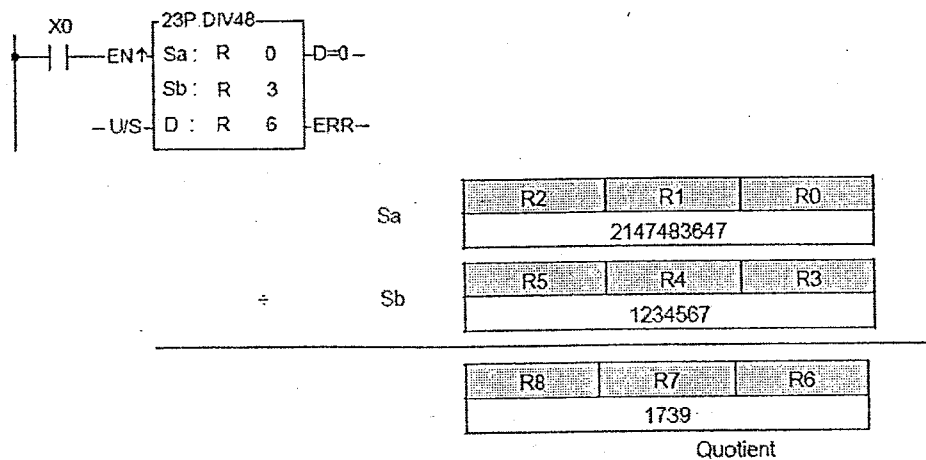


برعکس BIN TO BCD عمل می کند.

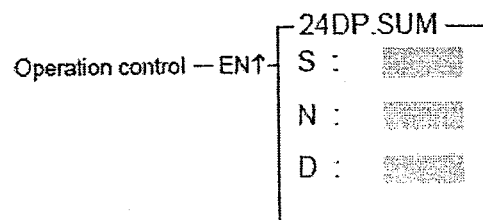
اگر نتیجه صفر باشد 'D=0' فعال می شود.

اگر Sb صفر باشد 'ERR' فعال می شود.

مثال:



24.SUM

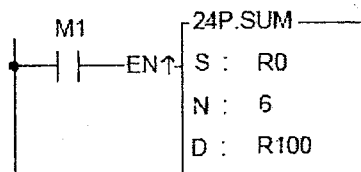


با این تابع می توان مجموع چند رجیستر متوالی را به یک رجیستر دیگر منتقل کرد.

در S اولین رجیستر قرار داده می شود. در N تعداد رجیسترهای متوالی تعیین می شود و مجموع این N تعداد

رجیستر در رجیستر می شود.

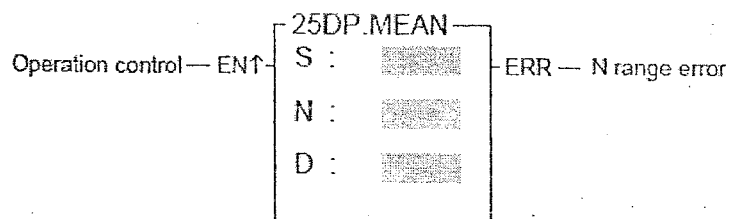
مثال:



R0=0030H
 R1=0031H
 R2=0032H
 R3=0033H
 R4=0034H
 R5=0035H

} → R100=012FH

25.MEAN



این تابع برای میانگین گرفتن از مقادیر چند رجیستر متوالی کاربرد دارد.

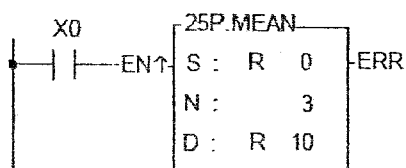
رجیستر شروع در S قرار می گیرد. تعداد رجیسترها در N

هرگاه EN از ۰ به ۱ 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 N بین ۲ تا ۲۵۶ نباشد، ERR فعال شده و تابع اجرا نمی شود.

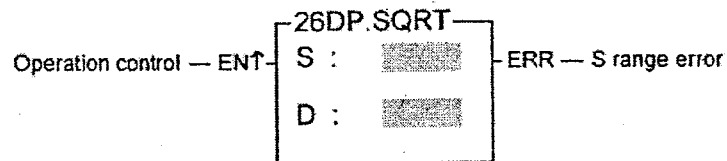
مثال:



S (N=3)	R0	123	$\frac{123 + 9 + 788}{3}$ = 306 (Rounding off the remainder)
	R1	9	
	R2	788	
$\Downarrow x_0 = \Uparrow$			
D	R10	306	

26.SQUARE ROOT

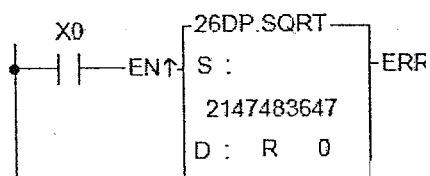
Ladder symbol



این تابع، جذر عدد موجود در S را گرفته و نتیجه را بدون در نظر گرفتن اعشار آن در D می ریزد.

اگر مقدار S منفی باشد، 'ERR' فعال شده و تابع اجرا نمی شود.

مثال:



S K 2147483647

↕ X0 = ↑

D R1 R0 46340

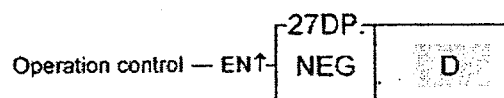
R1 R0

$$\sqrt{2147483647} = 46340.95$$

↑

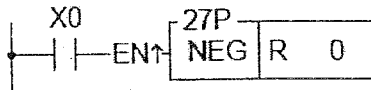
rounding off

27.NEGATION



هرگاه 'ENT' از ۰ به ۱ تغییر کند، مقدار D منفی می شود و دوباره در همان رجیستر ریخته می شود.

مثال:

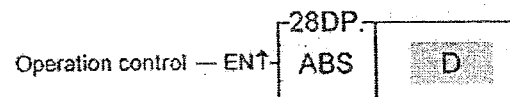


D R0 12345 → 3039H

↓ X0 = 1

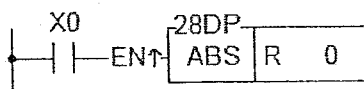
D R0 -12345 → CFC7H

28.ABSOLUTE



این تابع قدر مطلق مقدار موجود در D را دوباره در D می‌ریزد.

مثال:

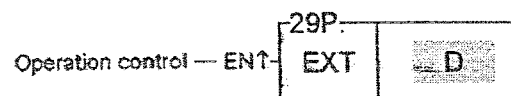


D R1 R0 -12345 → CFC7H

↓ X0 = 1

D R1 R0 12345 → 3039H

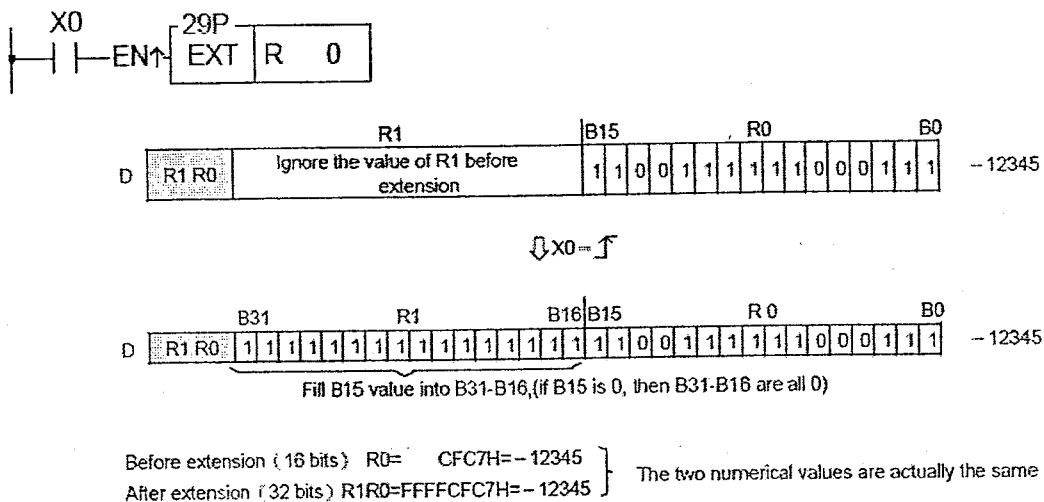
29.SIGN EXTENSION



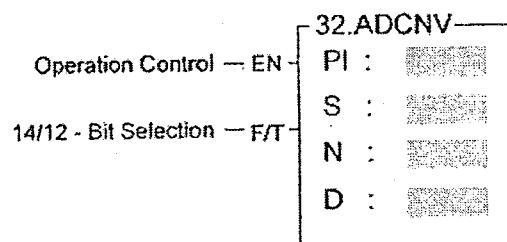
در D یک مقدار ۱۶ بیتی قرار دارد، با فعال شدن 'ENT' همین مقدار ۳۲ بیتی می‌شود.

(برای این کار از رجیستر D+1 استفاده می‌شود)

مثال:



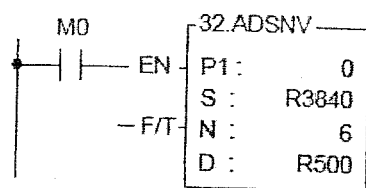
32.ADCNV



این تابع ورودی آنالوگ را به حساب میلی آمپر را که در رنج (4~20mA) باشد به عددی در رنج (0~20mA) تبدیل می کند (0~16383) که برای کار PLC مناسب تر است.

وقتی ورودی F/T = 0 باشد، عدد تبدیل شده در رنج (12-bit) 0~4095 قرار می گیرد.
 و اگر ورودی F/T = 1 باشد، عدد تبدیل شده در رنج (14-bit) 0~16383 قرار می گیرد.
 وقتی EN = 1 است، تبدیل را از S به طول N شروع کرده و نتایج را در D ذخیره می کند.

مثال:



S

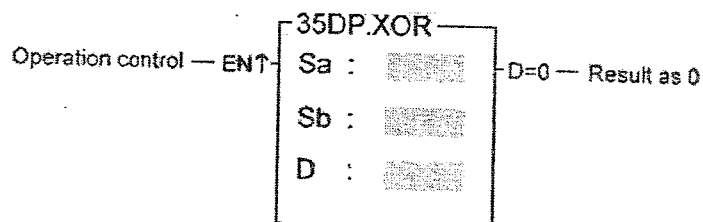
R3840	-1229
R3841	409
R3842	2047
R3843	-2048
R3844	-2048
R3845	-2048



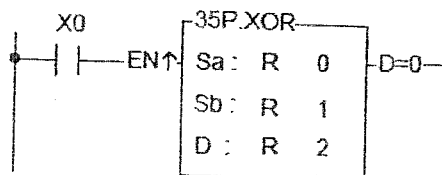
D

R500	0	(4 mA)
R501	2047	(12 mA)
R502	4095	(20 mA)
R503	0	(0 mA)
R504	0	(0 mA)
R505	0	(0 mA)

35.EXCLUSIVE OR (XOR)



هرگاه 'EN' از ۰ به ۱ تغییر کند، بیت‌های موجود در Sa و Sb را با هم XOR کرده و نتیجه را در D می‌ریزد. عملکرد به این صورت است که هرگاه بیت‌های متناظر Sa و Sb همانند بودند، بیت متناظر در D، ۰ می‌شود و هرگاه یکی ۱ و آن یکی ۰ بود، نتیجه در D، ۱ می‌شود. هرگاه تمام بیت‌های D، صفر شود، 'D=0' فعال می‌شود.

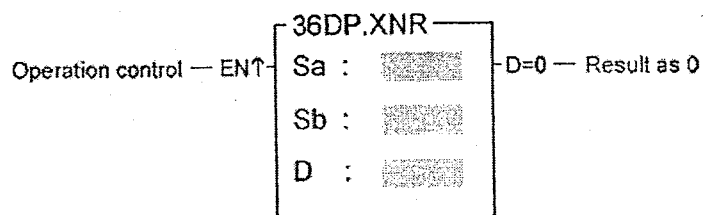


Sa	R0	1	0	1	1	1	0	1	1	0	1	1	0	1	1	0	1
Sb	R1	1	1	1	0	1	1	1	0	1	0	1	0	0	1	1	0

$\Downarrow X0 = \Uparrow$

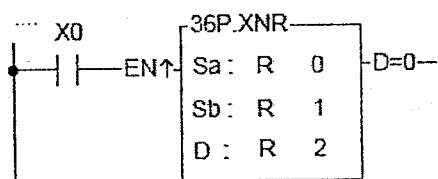
D	R2	0	1	0	1	0	1	0	1	1	1	0	0	1	0	1	1
---	----	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

36.EXCLUSIVE NOR (XNOR)



عملکرد این تابع بر عکس تابع قبل است یعنی دو بیت همانند در Sa و Sb، متناظر یا 1 در D و دو بیت ناهمانند

در Sa و Sb، متناظر یا 0 در D می باشد.

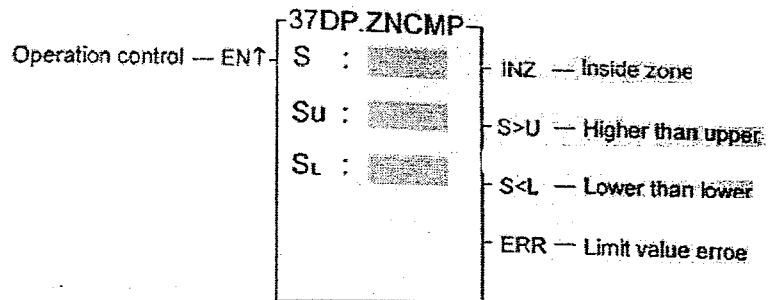


Sa	R0	1	0	1	1	1	0	1	1	0	1	1	0	1	1	0	1
Sb	R1	1	1	1	0	1	1	1	0	1	0	1	0	0	1	1	0

$\Downarrow X0 = \Uparrow$

D	R2	1	0	1	0	1	0	1	0	0	0	1	1	0	1	0	0
---	----	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

37.ZONE COMPARE



هرگاه EN به 1 تغییر کند، مقدار S با مقدار S_U و S_L مقایسه می شود.

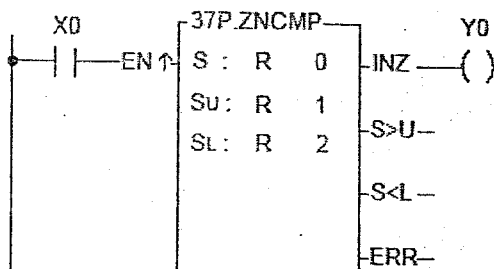
اگر بین این دو باشد، INZ فعال می شود

اگر بزرگتر از S_U باشد، S>U فعال می شود

اگر کوچکتر از S_L باشد، S<L فعال می شود.



اگر $S_U < S_L$ باشد، ERROR داده خواهد

مثال:



S	R0	200
Su	R1	300
Sl	R2	100

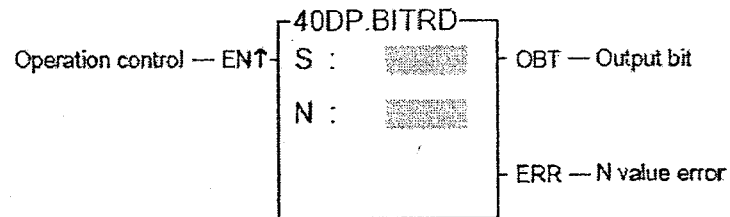
Before-execution

(Upper limit value) X0 = 
(Lower limit value) 

Y0


Results of execution

40.BIT READ



داده های منبع در S ریخته می شود.

هرگاه 'ENT' از ۰ به ۱ تغییر کند:

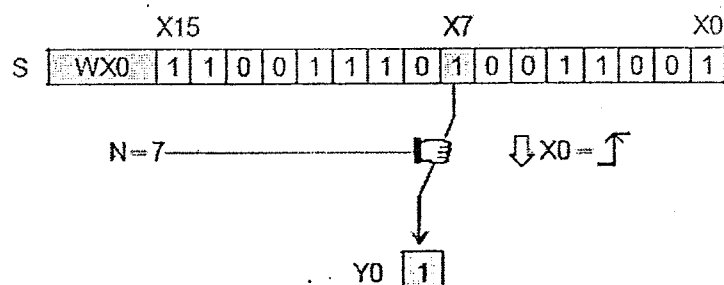
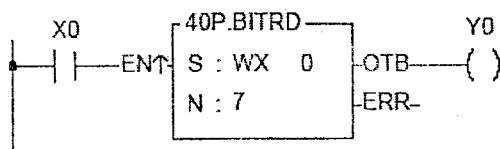
N آمین بیت داده ی S در خروجی 'OBT' قرار می گیرد.

اگر داده ی S، ۱۶ بیتی باشد: N:0~15

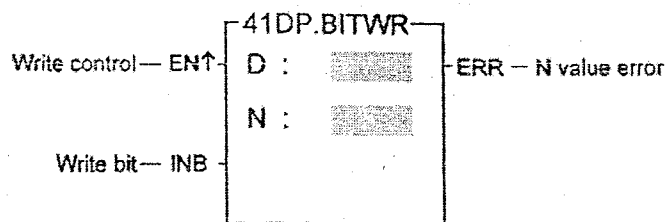
اگر داده ی S، ۳۲ بیتی باشد: N:0~31

در غیر این صورت error می دهد.

مثال:



41.BIR WRITE



هرگاه 'EN' از ۰ به ۱ تغییر کند:

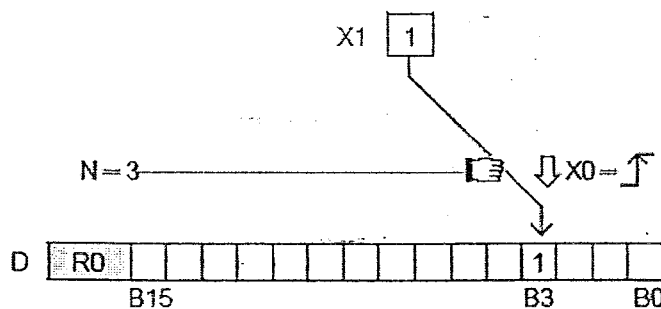
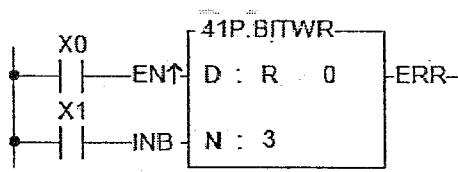
مقدار 'INB' در N^{امین} بیت رجیستر D ریخته می شود.

اگر عملوند D، ۱۶ بیتی باشد : N:0~15

اگر عملوند D، ۳۲ بیتی باشد : N:0~31

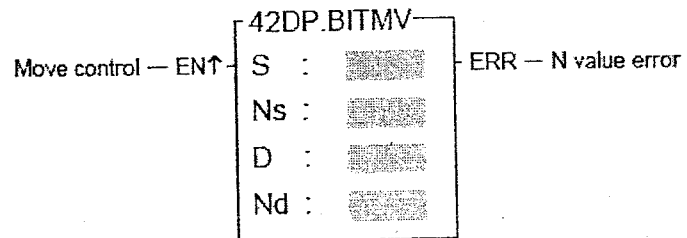
در غیر این صورت ERROR می دهد.

مثال:



تمام بیت ها غیر از B3 دست نخورده باقی می مانند.

42.BIT MOVE



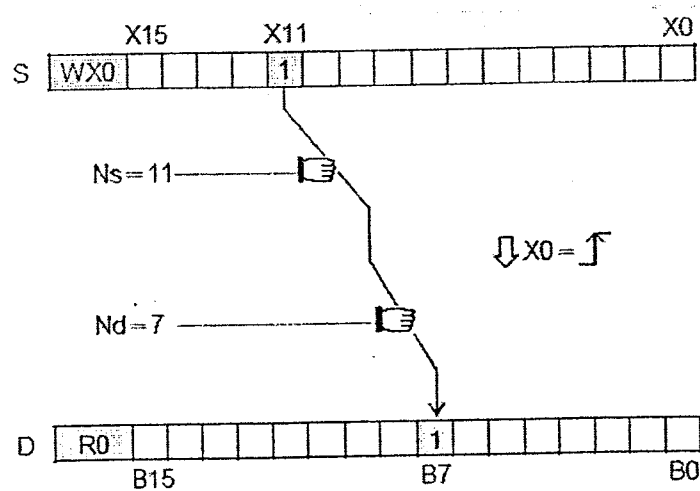
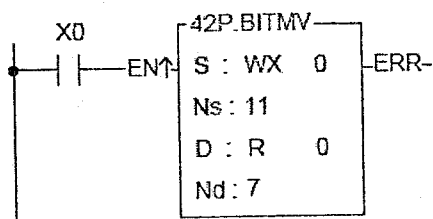
هرگاه 'EN' از ۰ به ۱ تغییر کند:

Ns آمین بیت رجیستر S به

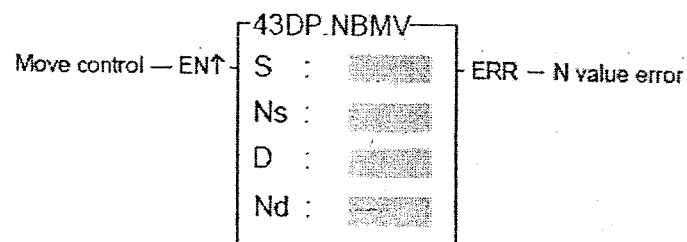
Nd آمین بیت رجیستر D ریخته می شود.

هرگاه مقادیر Ns و Nd متناسب با مقادیر S و D نباشند، error فعال می شود.

مثال:



43. NIBBLE MOVE

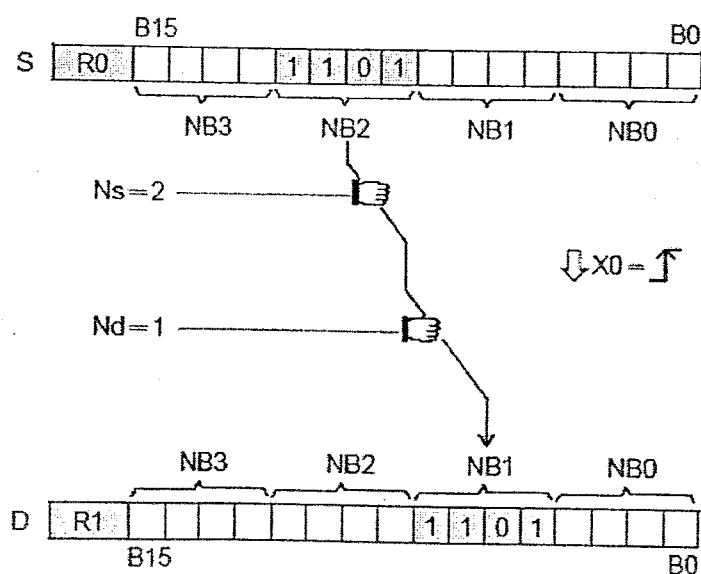
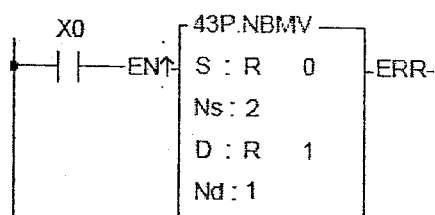


هرگاه 'EN' از ۰ به ۱ تغییر کند:

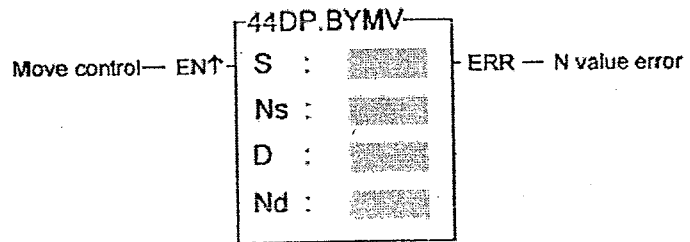
Ns آمین نیبل (Nibble=4bits) از رجیستر S را به

Nd آمین نیبل از رجیستر D منتقل می کند.

مثال:



44.BYTE MOVE



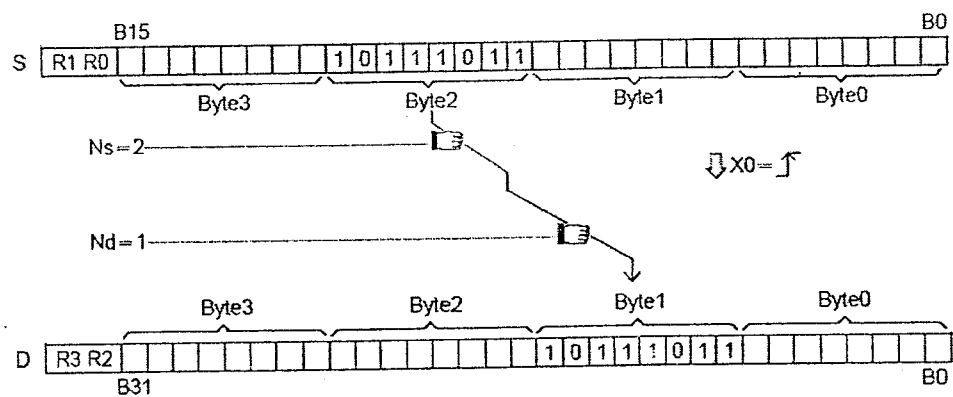
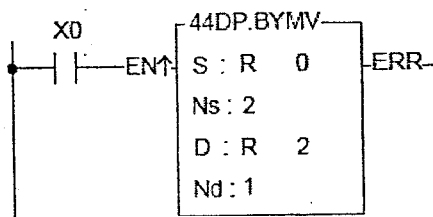
هرگاه 'EN' از ۰ به ۱ تغییر کند:

Ns آمین بایت رجیستر S به

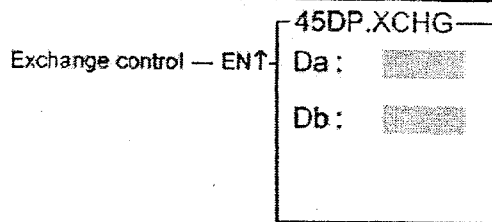
Nd آمین بایت رجیستر D ریخته می شود.

هرگاه مقادیر Ns و Nd متناسب با مقادیر S و D نباشد، error فعال می شود.

مثال:



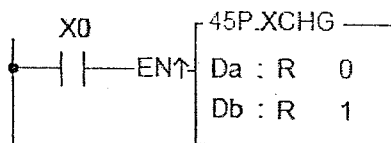
45. EXCHANGE



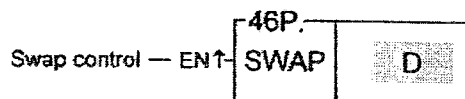
هرگاه 'EN' از ۰ به ۱ تغییر کند:

مقادیر رجیستر Da و Db با هم عوض می شوند.

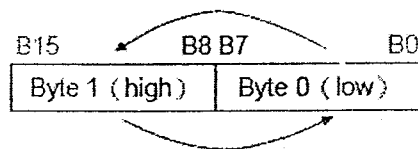
مثال:

[illegible]
$$\Downarrow x0 = \uparrow$$
[illegible]

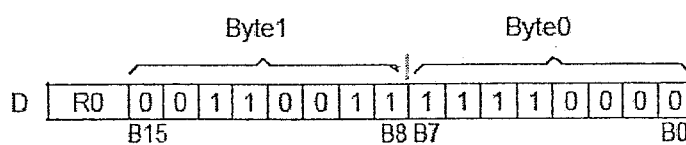
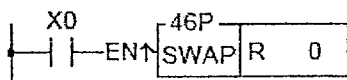
46.BYTE SWAP



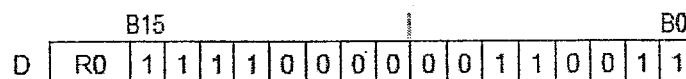
در D یک رجیستر ۱۶ بیتی قرار می‌گیرد که با فعال شدن 'EN'، بایت بالا و پایین آن با هم عوض می‌شوند.



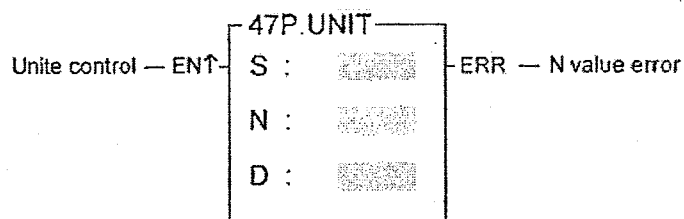
مثال:



⇓ X0 = ⇑



47. NIBBLE UNITE



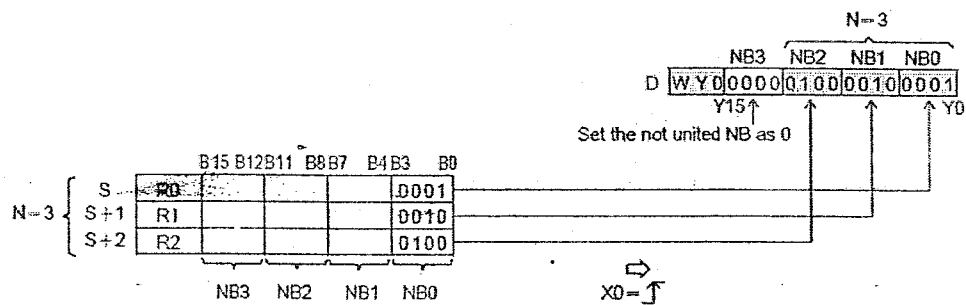
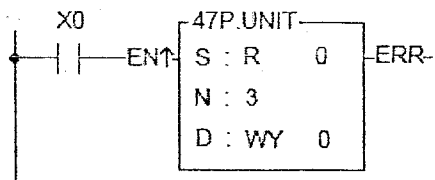
در S اولین رجیستر ۱۶ بیتی مورد نظر قرار می گیرد.

این تابع نابل های پایین N تعداد از رجیستر ها را به ترتیب در رجیستر D قرار می دهد.

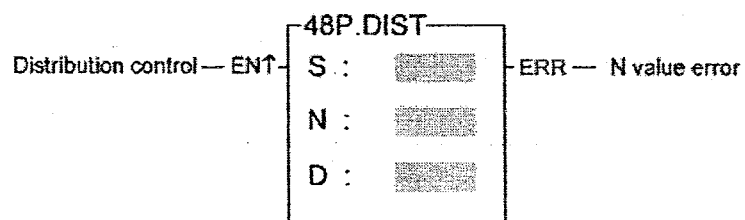
اگر N فرد باشد، باقی رجیستر D با ۰ پر می شود.

مقدار N باید ۱~۴ باشد، در غیر این صورت error داده می شود.

مثال:



48.NIBBLE DISTRIBUTE

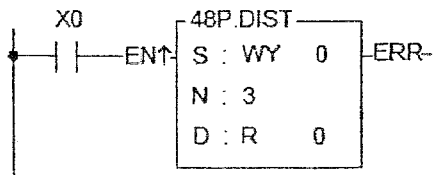


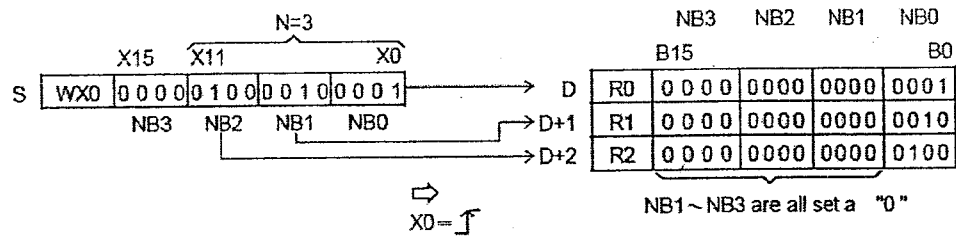
در S یک رجیستر ۱۶ بیتی قرار می گیرد که به ترتیب N تعداد از نیبل های آن به رجیستر D+1.D و منتقل می شود.

این نیبل ها، نیبل پایین رجیسترهای D+1.D و را اشغال می کنند و باقی فضای این رجیسترهای با ۰ پر می شود.

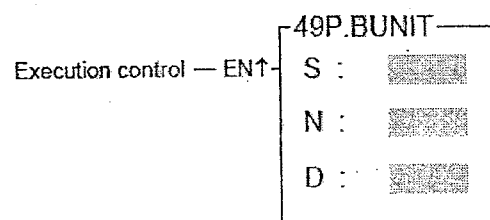
اگر N: 1~4 نباشد، ERROR فعال می شود.

مثال:





49.BITE UNITE

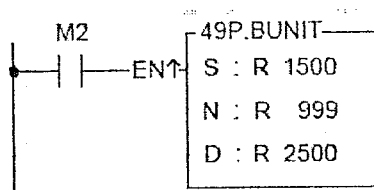


در S اولین رجیستر از رجیستر های مورد نظر قرار می گیرد:

این تابع بیت های پایین N تعداد از رجیستر های مشخص شده در S را به ترتیب در رجیستر های D, D+1, ...

قرار می دهد.

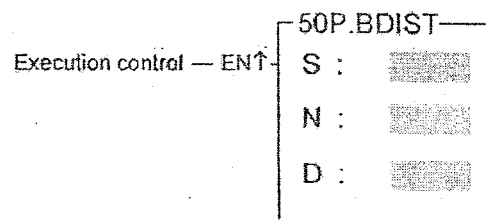
مثال:



	S	
	High Byte	Low Byte
R1500	Don't care	Byte-0
R1501	Don't care	Byte-1
R1502	Don't care	Byte-2
R1503	Don't care	Byte-3
R1504	Don't care	Byte-4
R1505	Don't care	Byte-5
R1506	Don't care	Byte-6
R1507	Don't care	Byte-7
R1508	Don't care	Byte-8
R1509	Don't care	Byte-9

	D	
	High Byte	Low Byte
R2500	Byte-0	Byte-1
R2501	Byte-2	Byte-3
R2502	Byte-4	Byte-5
R2503	Byte-6	Byte-7
R2504	Byte-8	Byte-9

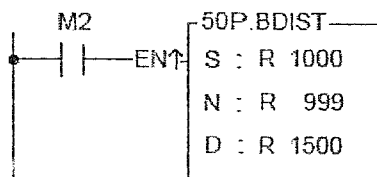
50.BYTE DISTRIBUTE



در S اولین رجیستر از رجیستر های مورد نظر قرار می گیرد.

این تابع با بیت های میبدار (N تعداد) به تایت های پایین رجیستر D+I.D منتقل می کند.

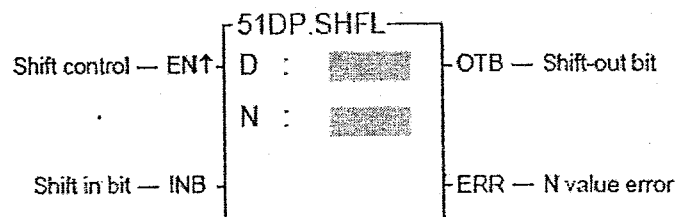
مثال:



	S	
	High Byte	Low Byte
R1000	Byte-0	Byte-1
R1001	Byte-2	Byte-3
R1002	Byte-4	Byte-5
R1003	Byte-6	Byte-7
R1004	Byte-8	Don't care

	D	
	High Byte	Low Byte
R1500	00	Byte-0
R1501	00	Byte-1
R1502	00	Byte-2
R1503	00	Byte-3
R1504	00	Byte-4
R1505	00	Byte-5
R1506	00	Byte-6
R1507	00	Byte-7
R1508	00	Byte-8

51.SHIFT LEFT



در رجیستری که قرار است شیفت داده شود قرار می گیرد

N، تعداد شیفت به چپ را مشخص می کنند.

جای بیت های شیفت داده شده را مقدار INB پر می کند و

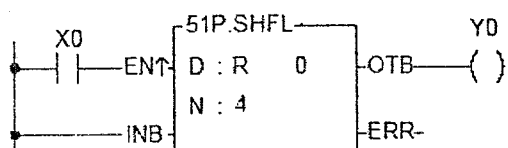
N آمین بیت بالای رجیستر به خروجی OTB می رود.

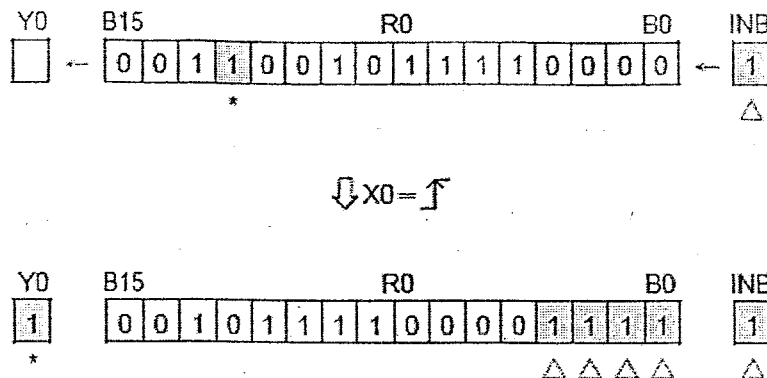
اگر رجیستر مورد نظر ۱۶ بیتی باشد: N:1~16

اگر رجیستر مورد نظر ۳۲ بیتی باشد: N:1~32

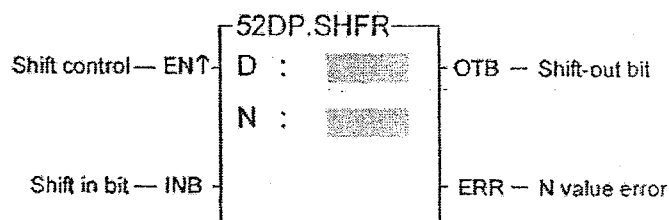
در غیر این صورت error می دهند.

مثال:





52.SHIFT RIGHT



در 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

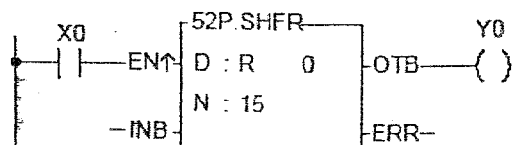
N آمین بیت پایین رجیستر به خروجی OTB می رود.

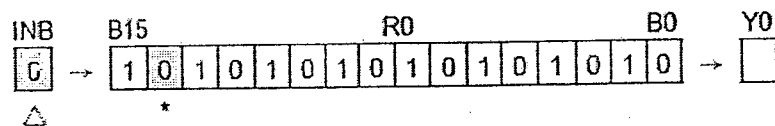
اگر رجیستر مورد نظر ۱۶ بیتی باشد: N:1~16

اگر رجیستر مورد نظر ۳۲ بیتی باشد: N:1~32

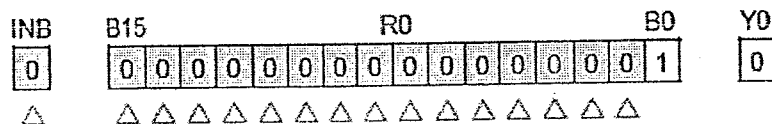
در غیر این صورت error می دهد.

مثال:

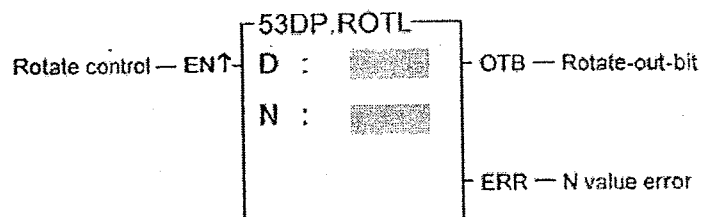




↓ X0 = ∫



53. ROTATE LEFT



با این تابع یک چرخش به سمت چپ در محل بیت ها انجام می گیرد و

چپ ترین بیت به راست ترین بیت منتقل می شود و

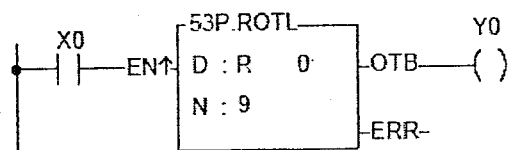
یک کپی از آن به خروجی 'OTB' می رود.

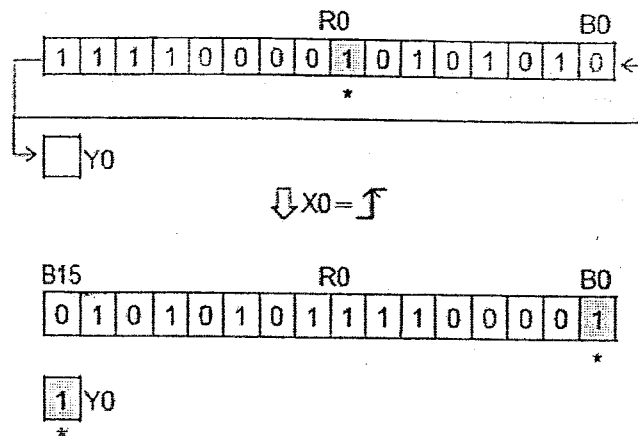
اگر رجیستر مورد نظر ۱۶ بیتی باشد: N:1~16

اگر رجیستر مورد نظر ۳۲ بیتی باشد: N:1~32

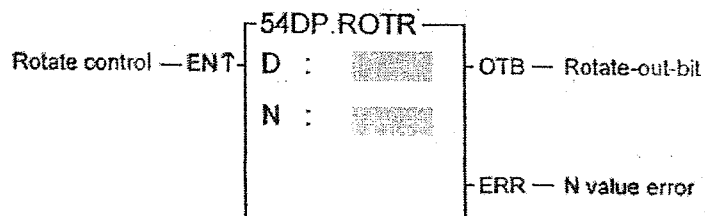
در غیر این صورت error می دهد

مثال:





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

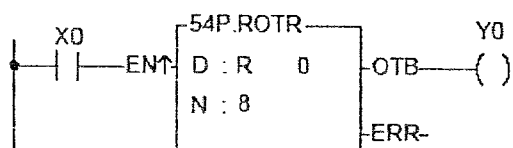
یک کپی از آن به خروجی OTB می رود

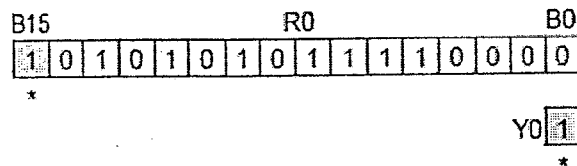
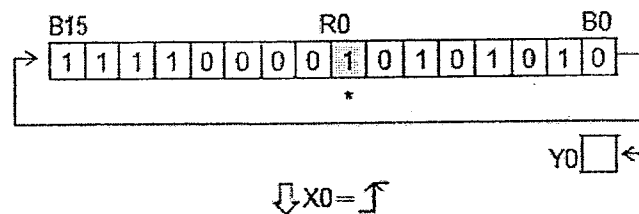
اگر رجیستر مورد نظر ۱۶ بیتی باشد: N:1~16

اگر رجیستر مورد نظر ۳۲ بیتی باشد: N:1~32

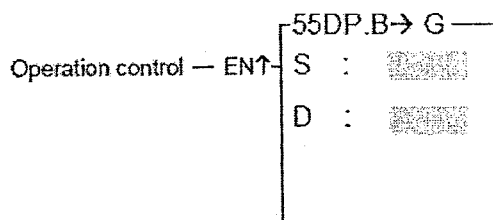
در غیر این صورت error می دهد.

مثال:





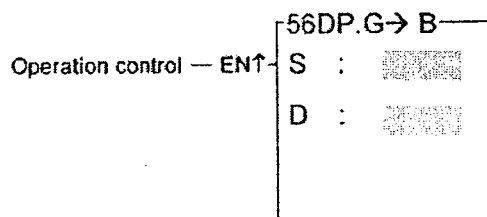
55. BINARY TO GRAY



هرگاه EN از 0 به 1 تغییر کند:

مقدار باینری موجود در رجیستر S به صورت کد گری، کدبندی می شود و نتیجه در D ریخته می شود.

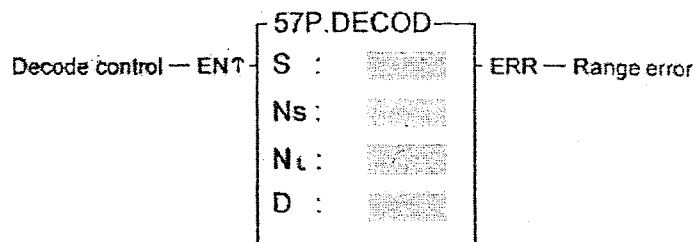
56. GRAY TO BINARY



هرگاه EN از 0 به 1 تغییر کند:

مقدار گری موجود در رجیستر S به صورت باینری، کدبندی می شود و نتیجه در D ریخته می شود.

57.DECODE



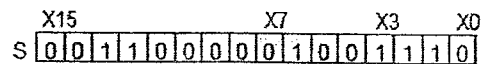
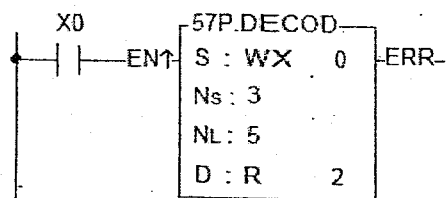
رجیستری که کدگشایی می شود در S ریخته می شود و نتیجه در رجیستر D+1.D ریخته می شود.

طول رجیستر D: 2^{NL} بیت می شود.

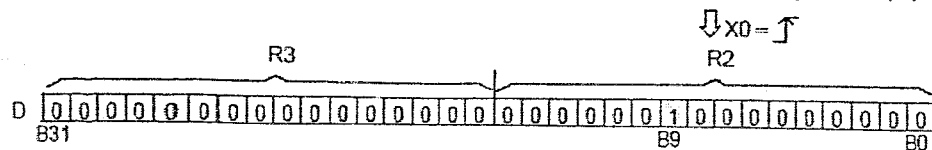
مقداری که کدگشایی می شود از Ns آیین بیت رجیستر مبدأ (S) آغاز شده و طول آن N_L بیت خواهد بود. معادل دسیمال این مقدار (به عنوان مثال ۹) باعث فعال شدن بیت ۹م (B9) از رجیستر مقصد (D) خواهد شد.

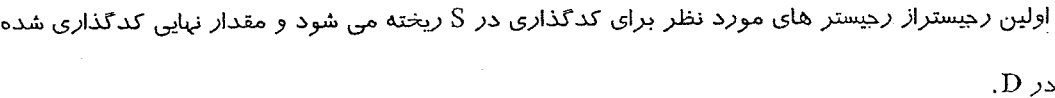
در S یک رجیستر ۱۶ بیتی ریخته می شود پس اگر مقدار N_L یا N_s متناسب با این رنج نباشد، error داده خواهد شد.

مثال:



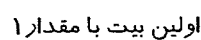
Length of decode value $N_L=5$, so bit value is formed by X7-X3 (equal 9)





هرگاه $H/L = 1$ باشد، کدگذاری با اولویت بالا انجام می شود و

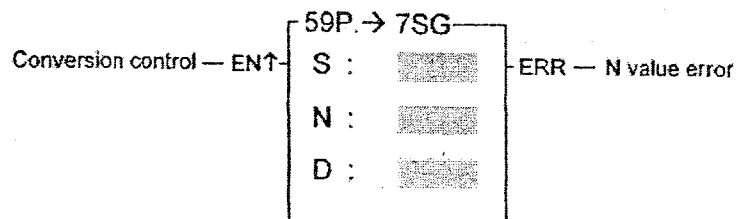
مثال:



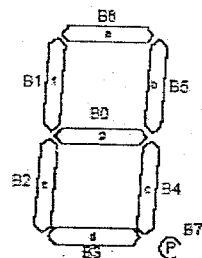
برای کدگذاری با اولویت بالا



59.7-SEGMENT CONVERSION



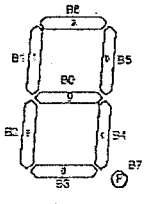
در S یک رجیستر ۱۶ بیتی قرار می گیرد که هر نیبل (۴ بیت) آن معرف یک عدد هگز است که از طریق این تبدیل به یک عدد ۸ بیتی تبدیل می شود (کد 7-seg) که این عدد در D ذخیره می شود. سگمنت های 7-seg با حروف زیر، علامت گذاری شده اند



هرگاه بیت B6 از رجیستر D، ۱ شود، معادل نمایش سگمنت a از 7-seg می باشد.
هرگاه بیت B5 از رجیستر D، ۱ شود، معادل نمایش سگمنت b از 7-seg می باشد و...

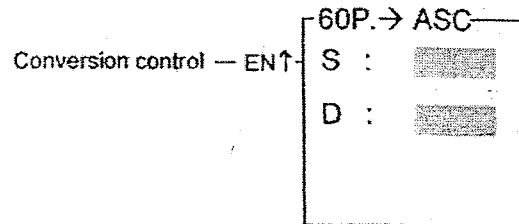
تعداد نیبل هایی که تبدیل 7-seg روی آنها انجام می گیرد N+۱ است.

رنج مؤثر N، ۰~۳ است. در غیر این صورت error داده خواهد شد.

Nibble data of S		7-segment display format	Low byte of D								Display pattern
Hexadecimal number	Binary number		B7 ●	B6 a	B5 b	B4 c	B3 d	B2 e	B1 f	B0 g	
0	0000		0	1	1	1	1	1	1	0	0
1	0001		0	0	1	1	0	0	0	0	1
2	0010		0	1	1	0	1	1	0	1	2
3	0011		0	1	1	1	1	0	0	1	3
4	0100		0	0	1	1	0	0	1	1	4
5	0101		0	1	0	1	1	0	1	1	5
6	0110		0	1	0	1	1	1	1	1	6
7	0111		0	1	1	1	0	0	1	0	7
8	1000		0	1	1	1	1	1	1	1	8
9	1001		0	1	1	1	1	0	1	1	9
A	1010		0	1	1	1	0	1	1	1	A
B	1011		0	0	0	1	1	1	1	1	b
C	1100		0	1	0	0	1	1	1	0	c
D	1101		0	0	1	1	1	1	0	1	d
E	1110		0	1	0	0	1	1	1	1	E
F	1111		0	1	0	0	0	1	1	1	F

جدول الكوى نمایشی 7-seg

60.ASCII CONVERSION



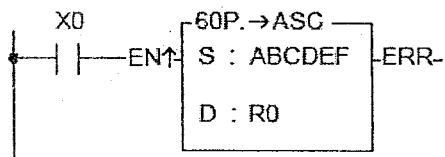
هرگاه EN از ۰ به ۱ تغییر کند:

تبدیل ASCII به روی اعداد و حروف ذخیره شده در S، انجام می شود و کد ASCII آن حروف یا اعداد در D ذخیره می شود.


در S حداکثر ۶ رجیستر ۱۶ بیتی ذخیره می شود. (در هر رجیستر ۲×ASCII قرار می گیرد).

کاربرد این تابع برای دستگاه های نمایشگری است که ورودی را به صورت کد ASCII می پذیرند.

مثال:

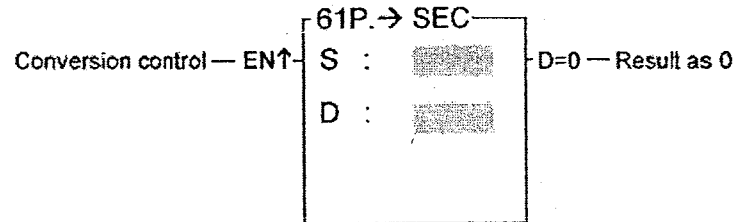


Alphabet
ABCDEF

X0 = 

	D	
	High Byte	Low Byte
R0	42 (B)	41 (A)
R1	44 (D)	43 (C)
R2	46 (F)	45 (E)

61.H:M:S to SECONDS CONVERSION



هرگاه 'EN' از ۰ به ۱ تغییر کند:

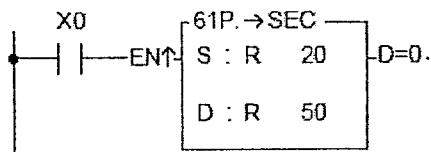
این تابع زمانی را که به صورت ساعت، دقیقه و ثانیه در رجیسترهای S~S+2 ذخیره شده است را بر حسب ثانیه

در رجیستر D ذخیره می کند.

اگر نتیجه ۰ باشد، 'D=0' فعال می شود.

مقدار ذخیره شده در S بر حسب ثانیه، S+1 بر حسب دقیقه و S+2 بر حسب ساعت است.

مثال:

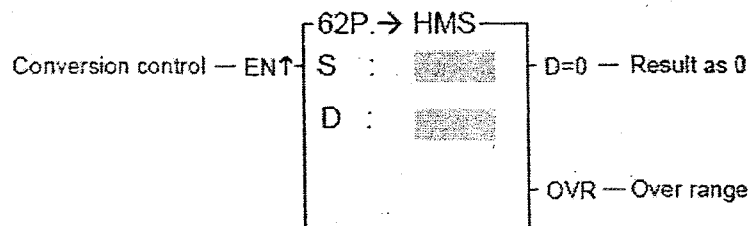


S	R20	0E11H	= 3601 sec
	R21	FD2FH	= -721 min
	R22	03F3H	= 1011 hr

↓ X0 = ↑

D	R50	EE45H	} = 3599941 sec
	R51	0036H	

62.SECOND to H:M:S



هرگاه 'ENT' از ۰ به ۱ تغییر کند:

این تابع بر عکس تابع قبل عمل می کند و زمان بر حسب ثانیه را بر حسب ساعت، دقیقه و ثانیه تبدیل می کند.

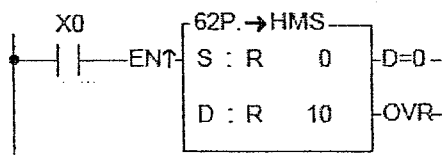
مقداری که در D ذخیره می شود بر حسب ثانیه، D+1 بر حسب دقیقه و D+2 بر حسب ساعت است.

اگر مقدار موجود در S، ۰ باشد، D=0 فعال می شود.

مقدار مناسب برای S، 117964799-117968399 ثانیه می باشد در غیر این صورت

'OVR' (OVER RANGE) فعال می شود.

مثال:

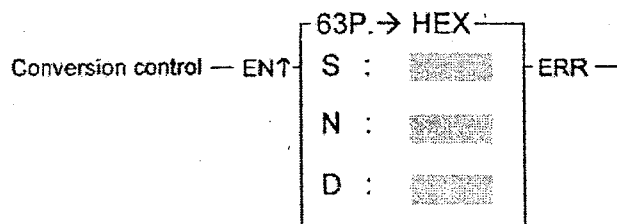


R0	5D17H	} 6315287 sec
R1	0060H	

↓ X0 = ↑

R10	002FH	47 sec
R11	000EH	14 min
R12	06DAH	1754 hr

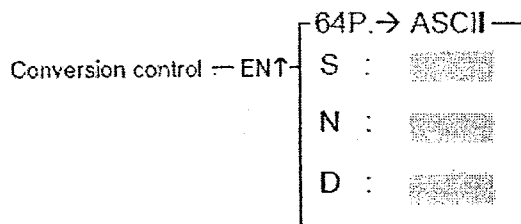
63.ASCII to HEX



هرگاه 'EN' از ۰ به ۱ تغییر کند:

N تعداد از کدهای ASCII که در S ذخیره شده اند، به معادل هگز خود تبدیل شده و در D ذخیره می شوند. هدف اصلی این تابع تبدیل کد ASCII کاراکترهای ('0'~'9' و 'A'~'F') به معادل هگز آنها می باشد و برای دستگاه هایی کاربرد دارد که CPU می تواند کد هگز را پردازش کند. بنابراین اگر مقدار موجود در S معادل ASCII کاراکترهای ('0'~'9' و 'A'~'F') نباشد، 'ERR' فعال خواهد شد.

64.HEX to ASCII



هرگاه 'EN' از ۰ به ۱ تغییر کند:

برعکس تابع قبل عمل کرده و N تعداد از نیبل های S را به صورت معادل ASCII در D+1.D ذخیره می کند. کاربرد این تابع در این است که اعداد هگز پردازش شده توسط PLC را به کد ASCII برای ارتباط از طریق پورت ها تبدیل کند.

PROGRAM END



هرگاه 'EN' از ۰ به ۱ تغییر کند:

این تابع فعال شده و به ابتدای برنامه می رود.

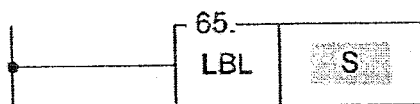
تمام برنامه ها بعد از تابع END دیگر اجرا نمی شوند. وقتی $EN=0$ ، این تابع نادیده گرفته می شود و برنامه

های بعد از این تابع اجرا نخواهند شد.

به کار بردن END در برنامه اصلی ضرورتی ندارد زیرا CPU، به صورت خودکار وقتی به انتهای برنامه

رسیده به نقطه شروع می رود.

65.LABLE



خود این تابع صرفاً عملی انجام نمی دهد و به عنوان یک برچسب آدرس برای برنامه ها عمل می کند.

به عنوان یک نشانگر برای اجرای تابع های INTERRUPT, CALL, JUMP 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 ۱ تا ۶ 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 ها ی 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.
PS00I~PS03I	Labels for the pulse output command finished interrupt service routine.

69.RTI



این تابع شبیه تابع RTS است با این فرق که RTS در انتهای اجرای زیربرنامه قرار

می گیرد در حالی که RTI در انتهای روتین INTERRUPT قرار می گیرد.

در مقایسه با CALL که از طریق یک LABEL، زیربرنامه مورد نظر را اجرا می کند.

INTERRUPT مستقیماً از طریق سیگنال های سخت افزاری فعال شده و در کار CPU وقفه ایجاد می کند تا به

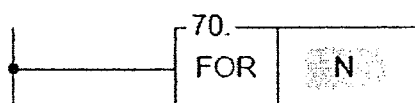
انجام روتین INTERRUPT بپردازد.

اگر اینترپتی در حین اجرای روتین یک اینترپت دیگر اتفاق بیافتد روتین اینترپتی اجرا خواهد شد که در اولویت

بالاتر قرار دارد.

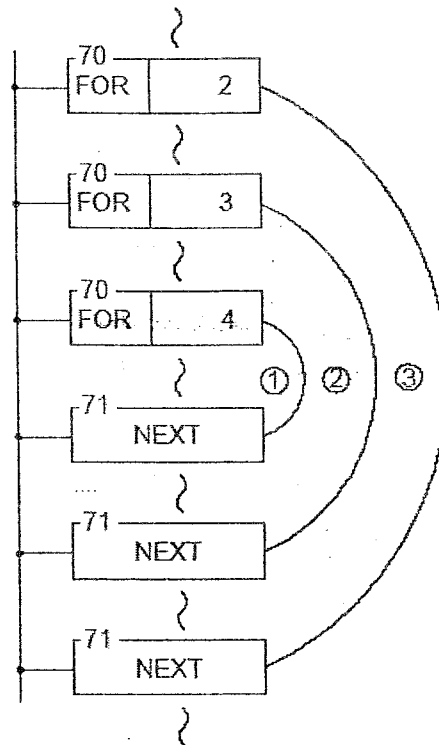
توجه داشته باشید که حتماً از RTI در انتهای روتین اینترپت استفاده کنید.

70.FOR

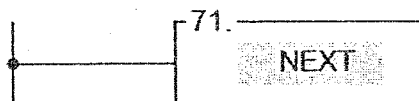


این تابع به همراه تابع NEXT، تشکیل یک حلقه را می دهد که برنامه ی میان FOR و NEXT مربوطه N بار اجرا می شود. حلقه های FOR و NEXT دیگری نیز در میان حلقه های FOR و NEXT اولیه می تواند قرار بگیرد.

به مثال توجه کنید:

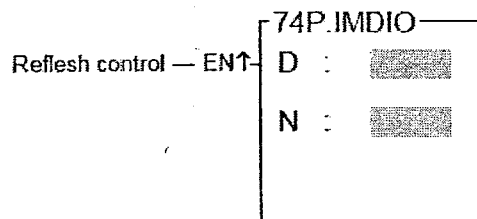


71.LOOP END



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.

74.IMMEDIATE I/O



این تابع سیگنال های ورودی و خروجی را فوراً، update می کند.

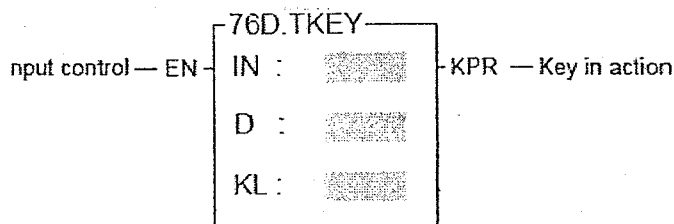
هرگاه 'EN' از ۰ به ۱ تغییر کند:

N سیگنال ورودی یا خروجی، refresh خواهند شد.

جدول زیر شماره ورودی و خروجی های مجاز برای استفاده این تابع را نشان می دهد.

Main-unit type Permissible numbers	20 points	32 points	40 points	60 points
Input signals	X0~X11	X0~X19	X0~X23	X0~X35
Output signals	Y0~Y7	Y0~Y11	Y0~Y15	Y0~Y23

76.DECIMAL-KEY INPUT



این تابع می تواند ورودی ۰~۹ را توسط کیبورد دریافت کند که این تعداد معادل با ۹~IN تا IN قرار

می گیرند. بر اساس ترتیب کلیدهای فشرده شده، یک عدد دهدهی ۴ یا ۸ رقمی می تواند در رجیستر مشخص

شده در D ذخیره شود.

اگر رجیستر موجود در D، ۱۶ بیتی باشد، این عدد می تواند ۴ رقمی باشد و اگر ۳۲ بیتی باشد، ۸ رقمی خواهد

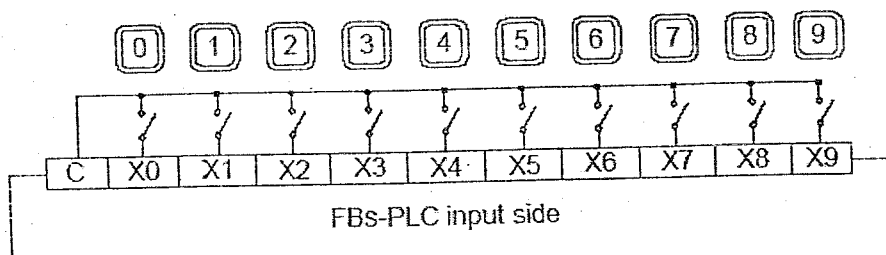
بود.

این تابع در صورتی عمل می کند که 'EN' = ۱ باشد.

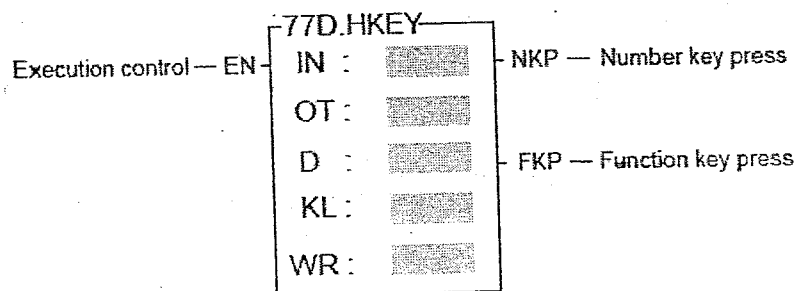
هرگاه هر یک از کلیدها فشرده شود، خروجی 'KPR' به اندازه ی همان مدت فشرده شدن کلید، ۱ خواهد ماند.

هرگاه هریک از اعداد فشرده شود، رجیستر معادل آن عدد که در KL مشخص می شود، ۱ خواهد شد و حتی اگر کلید مذکور رها شود بازهم ۱ خواهد ماند تا زمانی که عدد دیگری غیر از عدد قبلی فشرده شود؛ آنگاه رجیستر معادل عدد فشرده شده جدید در KL، ۱ خواهد شد.

با فشرده شدن هر عدد، معادل دهمی آن در D ذخیره می شود و هرگاه عدد جدیدی فشرده شود، عدد قبلی به سمت چپ شیفت پیدا می کند.

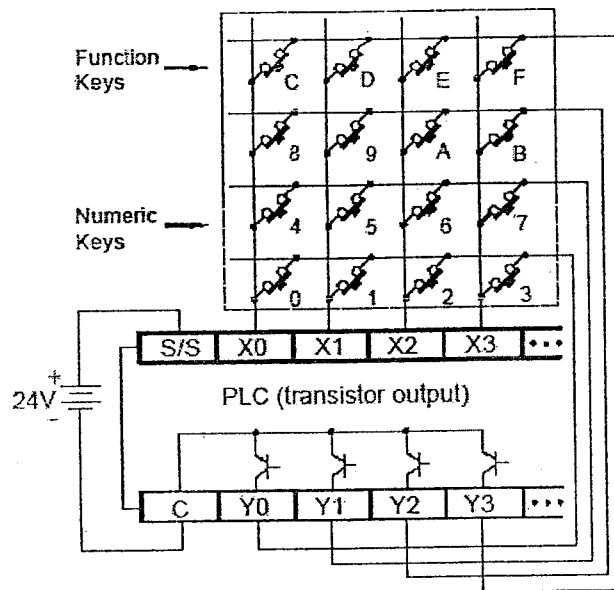


77. HEX-KEY INPUT



این تابع مشابه تابع قبل است با این تفاوت که علاوه بر اعداد ۰~۹ می تواند ۶ ورودی دیگر نیز دریافت کند (A~F) که هر کدام می توانند معادل یک دستور عمل باشند.

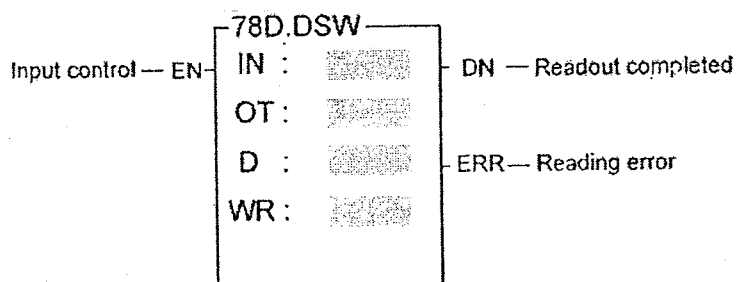
همچنین اتصال سخت افزاری این تابع با تابع قبل متفاوت است. ۴ ورودی اول PLC به ۴ خروجی اول PLC طوری متصل می شوند که اتصال هر کدام از آنها یکی از خروجی ها را بدهد.



هرگاه هر یک از اعداد 0~9 فشرده شود، خروجی 'NKP' به اندازه همان مدت فشرده شدن کلید 1، خواهد ماند و هرگاه یکی از دستورات A~F فشرده شوند، خروجی 'FKP' به اندازه همان مدت فشرده شدن کلید 1، خواهد ماند.

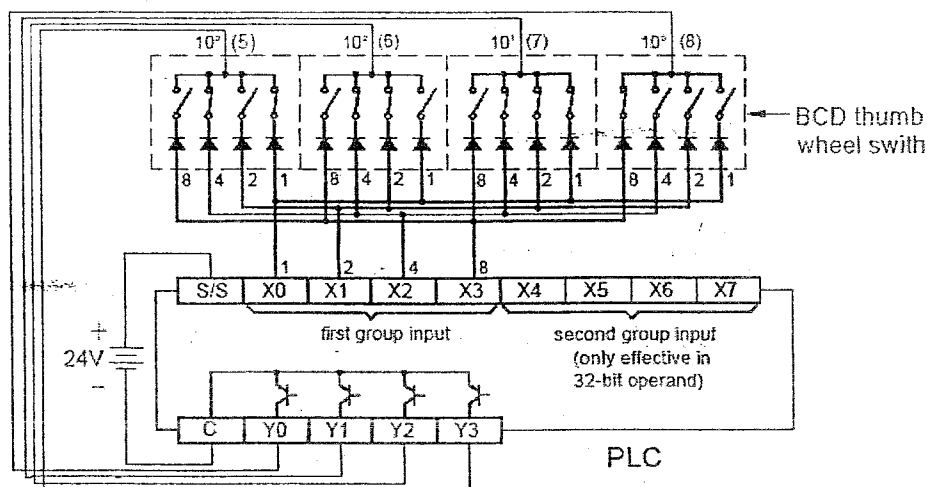
رجیستر ذخیره شونده در WR برای استعمال تابع است و در جای دیگری نباید از آن استفاده کرد.

78.DIGITAL SWITCH INPUT



هرگاه $EN = 1$ بشود، این تابع 4 رقم دهدهی را از سوئیچ BCD دستی، بازخوانی می کند و آنها را در D ذخیره می کند.

بیت های هم رقم باید به هم وصل شوند و از طریق یک دیود سری بشوند.



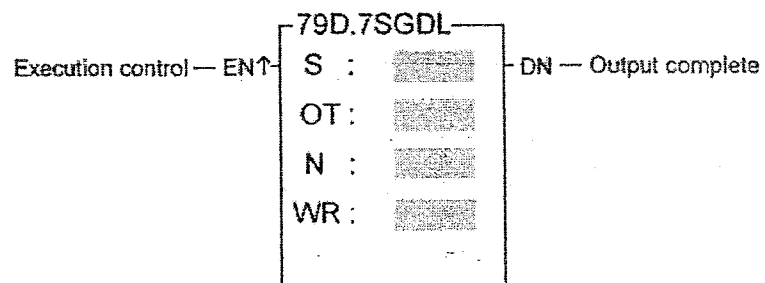
هر بار که ۴ رقم از سوئیچ ها خوانده شد، $DN=1$ می شود.

اگر هریک از اعداد خوانده شده در رنج BCD (0~9) نباشد، ERR فعال می شود.

PLC مورد استفاده باید دارای خروجی ترانزیستوری باشد.

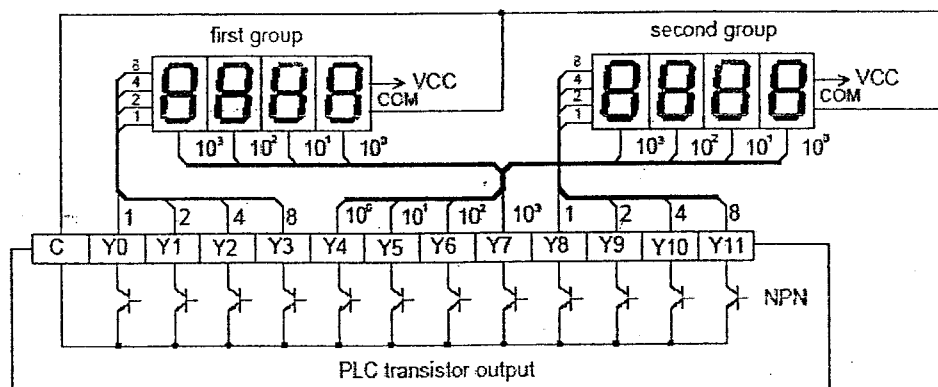
رجیستر ذخیره شوئنده در WR برای کاربری تابع است و در جای دیگری نباید از آن استفاده کرد.

79.7-SEG OUTPUT WITH LATCH

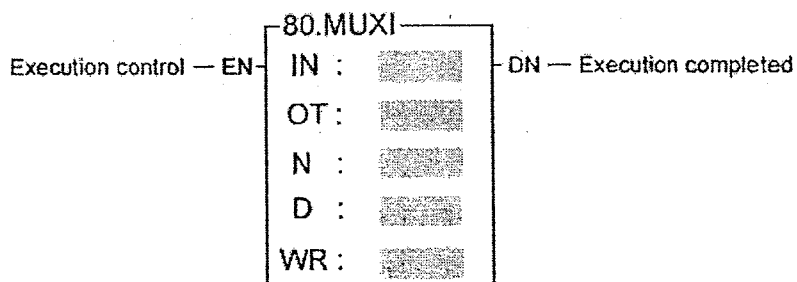


با توجه به شکل بعدی، هرگاه $EN=1$ ، ۴ نیبل رجیستر مشخص شده در S برای نمایش به 7-SEG سری اول

منتقل می شوند. ۴ نیبل $S+1$ به 7-SEG سری دوم منتقل می شوند.



80.MULTIPLEX INPUT



این تابع از مبدل پلکس برای خواندن $8 \times N$ ورودی استفاده می کند که فقط ۸ ورودی و N خروجی از PLC را استفاده می کند.

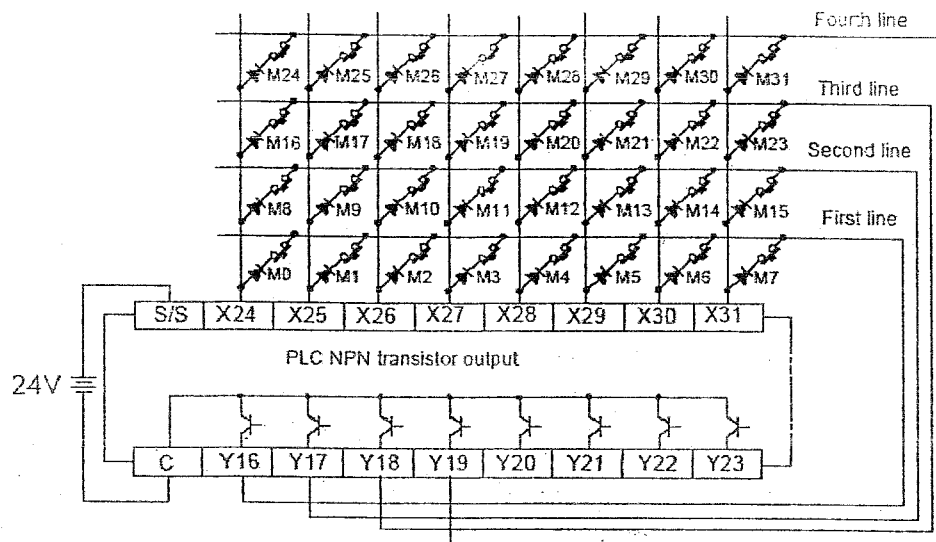
۸ ورودی با IN شروع می شوند و N خروجی از OT شروع می شوند.

در هر اسکن یکی از N خروجی ۱ می شود و خط مربوط به آن خروجی انتخاب می شود.

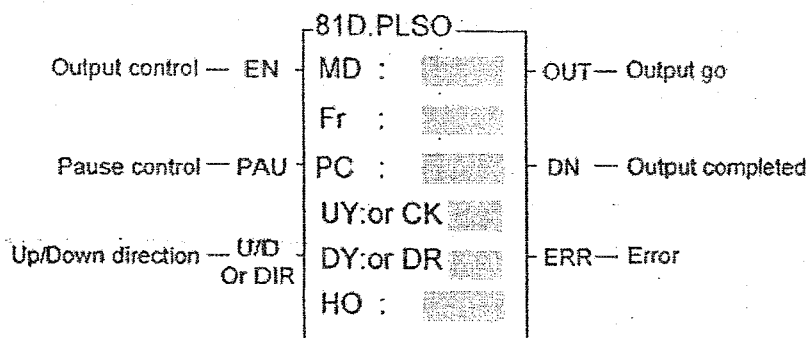
$OT0$ مربوط به خط اول، $OT1$ مربوط به خط دوم و ... است.

پس از اسکن تمام خطوط (N خط)، $8 \times N$ مشخصه در رجیستر D ذخیره می شود، سپس DN یک می شود. اما

فقط به اندازه یک اسکن یک می ماند.



81.PULSE OUTPUT



وقتی $MD=0$:

هرگاه 'EN' از 0 به 1 تغییر کند، ابتدای ست می کند که باعث پاک کردن 'OUT' و 'DN' و صفر شدن رجیستر HO می شود. سپس 'PAU' را چک می کند. اگر $PAU=1$ باشد، عمل صورت نمی گیرد. اگر $PAU=0$ ، در فرکانس Fr شروع به دادن پالس به خروجی UY یا DY می کند.

اگر $U/D=1$: پالس به UY می رود.

اگر $U/D=0$: پالس به DY می رود.

مقدار رجیستر HO، هر بار که پالس به خروجی می رود اضافه می شود تا زمانی که برابر یا بزرگتر از مقدار رجیستر PC شود. در این حالت $DN=1$ می شود. در طول مدتی که پالس در حال انتقال است، $OUT=1$ می ماند در غیر این صورت 0 خواهد بود.

هنگام انتقال پالس باید $EN=1$ نگاه داشته شود، اگر به ۰ تغییر یابد، فرستادن پالس متوقف می شود و $OUT=0$ می شود. اما داده های دیگر تغییر نمی کنند. وقتی EN مجدداً از ۰ به ۱ تغییر می کند، عملیات از اول آغاز می شود.

اگر می خواهید برنامه را طوری متوقف کنید که باعث ری ست شدن کل برنامه نشود، از ورودی PAU برای توقف برنامه استفاده کنید.

در طول انتقال پالس، این تابع به چک کردن مقدار فرکانس (FR) و تعداد پالس های نهایی (PC) می پردازد. بنابراین تا زمانی که فرستادن پالس پایان نیافته، می توانید Fr و PC را تغییر دهید. اما تنظیم جهت پالس (U/D) تنها هنگام ری ست کردن قابل اجرا خواهد بود. (وقتی EN از ۰ به ۱ تغییر می کند) و به همان حالت باقی می ماند تا زمانی که انتقال پالس متوقف شود یا ری ست دیگری اتفاق بیافتد.

هدف اصلی این تابع، راه اندازی استپ موتور یا دو کنترل پالس جهت دار راست گرد و چپ گرد است. اگر شما فقط احتیاج به گردش در یک سمت را داشته باشید، می توانید آنها به یکی از مقادیر UY یا DY مقدار بدهید. در این حالت تابع دیگر توجهی به مقدار U/D نخواهد کرد.

وقتی $MD=1$:

پالس خروجی به روی پایه کنترل DIR و رجیستر CK انداخته می شود.

این تابع تنها یک بار می تواند استفاده شود و $UY(CK)$ و $DY(DR)$ باید از خروجی های ترانزیستوری PLC گرفته شوند.

رنج موثر PC برای رجیستر ۱۶ بیتی: 0~32767

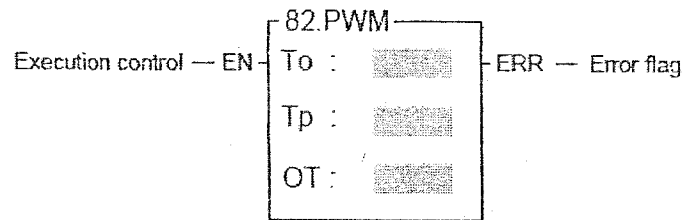
رنج موثر PC برای رجیستر 32 بیتی: 0~2147483647

اگر $PC=0$ ، بدون توقف پالس خواهد داد و $DN=0$ خواهد ماند.

رنج موثر FR : 8~2000

اگر مقدار PC یا Fr از رنج تعیین شده فراتر باشد، این دستور اجرا نشده و ERR فعال می شود.

82.PULSE WIDTH MODULATION



وقتی $EN=1$ است پالسی به خروجی OT می فرستد که To میلی ثانیه ON است و پریود آن Tp میلی ثانیه می باشد. OT باید از یک خروجی ترانزیستوری PLC گرفته شود.

حداقل مقدار To ، ۰ است که در این موقعیت خروجی همیشه به حالت OFF خواهد بود.

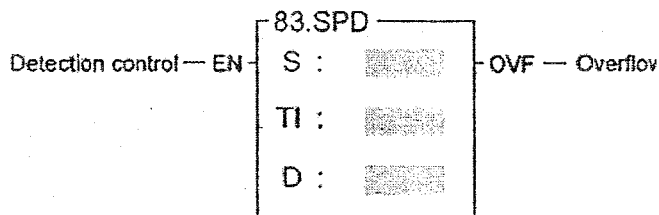
حداکثر مقدار To ، برابر Tp است که در این موقعیت خروجی همیشه به حالت ON خواهد بود.



اگر $To > Tp$ ، 'ERR' فعال شده و تابع عمل نخواهد کرد.

این تابع تنها یک بار می تواند استفاده شود.

83.SPEED DETECTION



این تابع برای به دست آوردن سرعت گردش دستگاه های چرخنده (مانند موتور) بر حسب rpm استفاده

می شود؛ به این صورت که فرکانس سیگنال ورودی را از طریق ۸ ورودی سرعت بالای PLC ($X0 \sim X7$) به

دست می آورد و با کمک زمان نمونه برداری (TI)، تعداد پالس ورودی S را مشخص می کند.

مطلوب است که هنگام استفاده از این تابع، طراحی به صورتی انجام گیرد که پالس بیشتری در هر چرخش تولید

شود تا نتیجه ی بهتری به دست آید. اما مجموع فرکانس تمام سیگنال های آشکار شده باید کمتر از 5KHz

باشد. در غیر این صورت WDT (Watch Dog Timer) فعال می شود. [تابع ۹۰]

رجیستر D برای ذخیره نتایج از سه رجیستر ۱۶ بیتی متوالی استفاده می کند (D0~D2)

در D0 نتیجه ی محاسبه ذخیره می شود.

در D1 مقدار شمارش کنونی ذخیره می شود.

در D2 زمان نمونه برداری ذخیره می شود.

وقتی $EN=1$ ، شروع به محاسبه ی تعداد پالس برای ورودی S می کند که در رجیستر D1 نشان داده می شود. در همین هنگام تایمر نمونه برداری (D2) فعال شده و به شمارش ادامه می دهد. تا زمانی که مقدار D2 برابر با پریود نمونه برداری (TI) شود. مقدار محاسبه شده نهایتاً در رجیستر D0 ذخیره شده، سپس چرخه ی محاسبه ی جدیدی، آغاز می شود. این کار تا وقتی $EN=0$ شود ادامه می یابد.

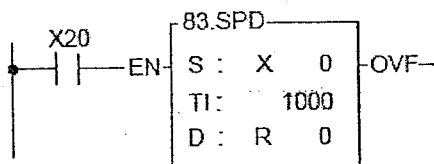
D0 یک رجیستر ۱۶ بیتی است پس حداکثر آن می تواند ۳۲۷۶۷ باشد. اگر پریود نمونه برداری خیلی طولانی باشد یا سرعت پالس ورودی خیلی زیاد باشد، مقدار D0 ممکن است از ۳۲۷۶۷ بیشتر شود در این صورت $OVF=1$ شده و عملیات متوقف می شود.

اگر هر چرخش یک دستگاه چرخنده n پالس تولید کند، می توان سرعت را از معادله زیر محاسبه کرد:

$$N = \frac{(D0) \times 60}{n \times TI} \times 10^3 \quad (\text{rpm})$$

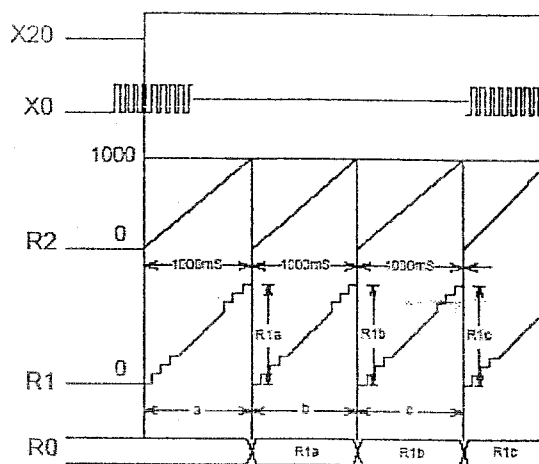
سرعت:

مثال:

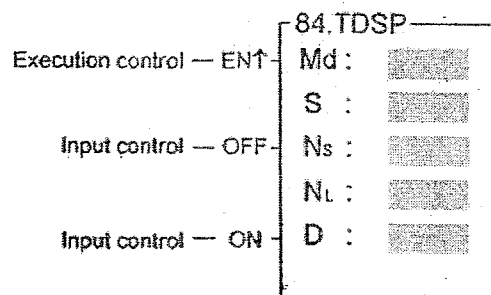


اگر $n=60$ باشد:

$$N = \frac{(200) \times 60}{60 \times 1000} \times 10^3 = 200 \text{ rpm}$$



84.16/7-SEG DISPLAY



این تابع برای مدل های FBs-7SG1 و FBs-7SG2 کاربرد دارد و می تواند کاراکترهای مختلف را برای نمایش در 16-seg و 17-seg آماده کند.

S آدرس شروع کاراکترهای مورد نظر برای تبدیل ذخیره می شود.

Ns مقداری برای pointer است که مشخص می کند کاراکتر، دقیقاً از کجا شروع می شود.

N_L طول کاراکتر مورد نظر را مشخص می کند.

D آدرس شروع محل ذخیره ی نتایج تبدیل را مشخص می کند.

بعد از اجرای تابع، هر یک کاراکتر (8-bits) مبدا (S) به الگوی نمایشی ۱۶ بیتی مربوطه تبدیل می شود.

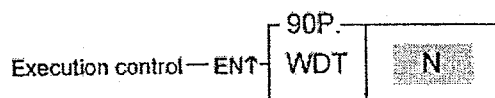
وقتی $EN=1$ ، $OFF=0$ ، $ON=0$ و $Md=0$ باشد، این تابع تبدیل را اجرا می کند. وقتی $OFF=1$ و $Md=0$ تمام بیت های مربوط به الگوی نمایشی 16-seg صفر خواهند شد. این بدین معنیست که اگر 16-seg در این حالت به PLC متصل شود، تمام LED های آن خاموش خواهد بود.

وقتی $OFF=1$ و $Md=1$ ، تمام بیت های مربوط به نمایش 7-seg صفر می شود.

وقتی $ON=1$ و $Md=0$ ، تمام بیت های مربوط به نمایش 16-seg، 1 می شود.

وقتی $ON=1$ و $Md=1$ ، تمام بیت های مربوط به نمایش 7-seg، 1 می شود.

90.WATCH DOG TIMER



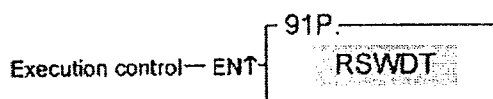
هدف اصلی از طراحی تایمر WDT، محافظت و نظره ایست که از طرف سیستم اعمال می شود. برای مثال، اگر CPUی PLC ناگهان آسیب ببیند و راهی برای اجرای برنامه یا I/O refresh نباشد، بعد از سپری شدن زمان تعیین شده در WDT، WDT به صورت خودکار تمام I/O ها را خاموش می کند. در کاربردهای خاص اگر scan time خیلی طولانی باشد، ممکن است باعث ایجاد مشکلات امنیتی یا عدم پیروی از کنترل شود که این تابع می تواند برای اعمال محدودیت مورد نیاز به روی scan time استفاده شود.

هرگاه EN از 0 به 1 تغییر کند:

WDT شمارش زمان $N \times 10 \text{ ms}$ را آغاز می کند و اگر در این مدت EN تغییر نکند پس از زمان تعیین شده PLC خاموش می شود.

یک بار که WDT set شد برای همیشه باقی می ماند و دیگر نیازی به ست کردن مجدد در هر اسکن ندارد.

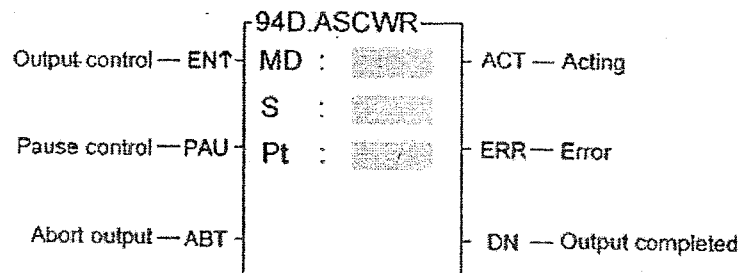
91.RESET WDT



وقتی EN 1 بشود:

WDT ری ست شده و محاسبه زمان را از 0 شروع می کند.

94.ASCII WRITE



هرگاه 'ENT' از ۰ به ۱ تغییر کند و MD=0 باشد، اطلاعاتی را که به صورت ASCII کد شده اند و از S شروع می شوند را به Port1 منتقل می کند تا زمانی که به پایان فایل برسد.

محتویات فایل S می تواند از طریق نرم افزار WinProLadder پردازش شود.

اطلاعات پردازش شده باید به فرمت ASCII باشد. در غیر این صورت این تابع تبادل اطلاعات را متوقف می کند و 'ERR' ۱ خواهد شد. اگر تمام فایل به صورت صحیح منتقل شود، خروجی 'DN' ۱ می شود.

در طول مدت انتقال اطلاعات، خروجی 'ACT' ۱ خواهد بود. تنها هنگام توقف خروجی، error یا کنسل کردن 'ACT' به ۰ بر می گردد.

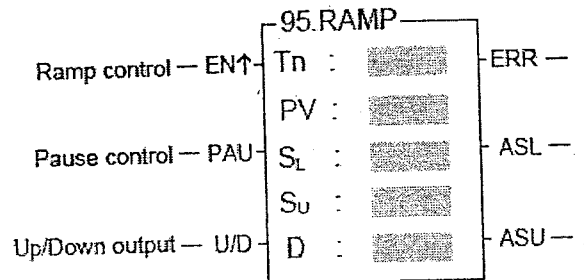
این تابع می تواند به صورت پی در پی به کار برده شود اما در یک زمان مشخص، تنها یکی از آنها اجرا خواهد شد.

وقتی این تابع در حال اجراست، اگر PAU ۱ بشود، این تابع انتقال اطلاعات را متوقف می کند؛ با برگشتن PAU به ۰، انتقال اطلاعات ادامه می یابد.

وقتی این تابع در حال اجراست، اگر ABT ۱ بشود، این تابع انتقال اطلاعات را رها می کند، سپس توانایی اجرای تابع بعدی را خواهد داشت.

95.RAMP

تابع شیب برای خروجی دیجیتال به آنالوگ



در T_n ، تایمری قرار داده می شود که زمان پایه آن $0.01s$ باشد و در جای دیگری استفاده نشود.

PV برای تنظیم تایمر شیب می باشد.

S_L حد پایین شیب و S_u حد بالای شیب را مشخص می کند.

هر گاه 'EN' از ۰ به ۱ تغییر کند:

ابتدا تایمر T_n ریست شده و ۰ می شود. سپس اگر $U/D = 1$ مقدار S_L در رجیستر D ذخیره می شود و وقتی

رجیستر $M1974=0$ هر $0.01s$ مقدار D به اندازه $PV - S_L$ افزایش می یابد.

وقتی مقدار D به S_u رسید، خروجی $ASU = 1$ می شود.

اگر $U/D = 0$ ، مقدار S_u در رجیستر D ذخیره می شود. وقتی $M1974=0$ هر $0.01s$ ، مقدار D به اندازه ی PV

$S_u - S_L$ کاهش می یابد.

وقتی مقدار D به S_L رسید، خروجی $ASL = 1$ می شود.

همزمان با تغییر 'EN' از ۰ به ۱، جهت شیب 'U/D' اندازه گرفته می شود. بعد از اینکه خروجی D شروع به شیب

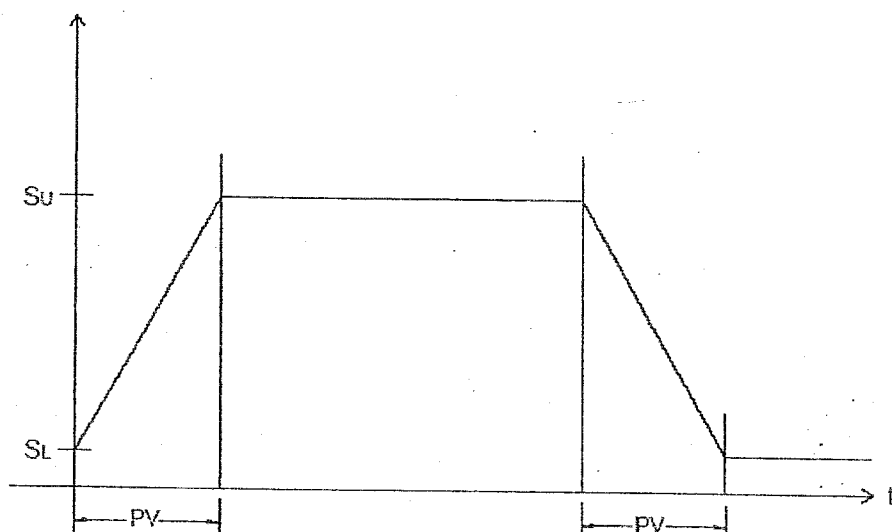
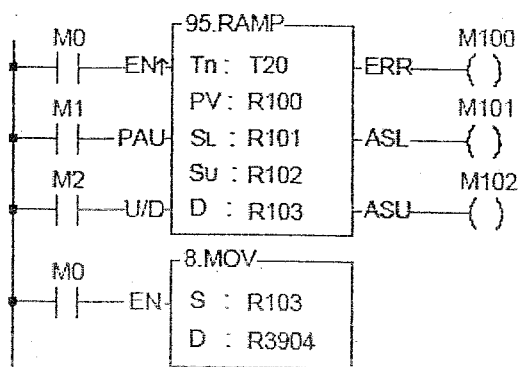
گرفتن کرد، تغییر 'U/D' فایده ای ندارد.

اگر نیاز به توقف عمل شیب دادن بود، باید $PAU = 1$ بشود. وقتی $PAU = 0$ کنید و هنوز شیب دادن به پایان

نرسیده باشد، عمل شیب دادن تا هنگامی که پایان یابد ادامه خواهد یافت.

مقدار S_u باید بزرگتر از S_L داده شود، در غیر این صورت تابع اجرا نشده و خروجی 'ERR' ۱ خواهد شد.

مثال:



توابع جدولی

Fun No.	Mnemonic	Functionality	Fun No.	Mnemonic	Functionality
100	R→T	Register to table data move	107	T_FIL	Table fill
101	T→R	Table to register data move	108	T_SHF	Table shift
102	T→T	Table to table data move	109	T_ROT	Table rotate
103	BT_M	Block table move	110	QUEUE	Queue
104	T_SWP	Block table swap	111	STACK	Stack
105	R-T_S	Register to table search	112	BKCMP	Block compare
106	T-T_C	Table to table compare	113	SORT	Data Sort

یک جدول شامل دو یا چند رجیستر متوالی است (۱۶ یا ۳۲ بیتی)

تعداد رجیستر ها که تشکیل یک جدول را می دهند، طول جدول می نامند. (L)

اجرای توابع جدولی بیشتر برای پردازش داده ها استفاده می شود، مانند انتقال ، کپی ، مقایسه ، جستجو و ... بین

جدول ها و رجیستر ها یا بین جدول ها.

در میان توابع جدولی، بیشتر توابع از یک Pointer استفاده می کنند تا مشخص کنند کدام رجیستر جدول، هدف

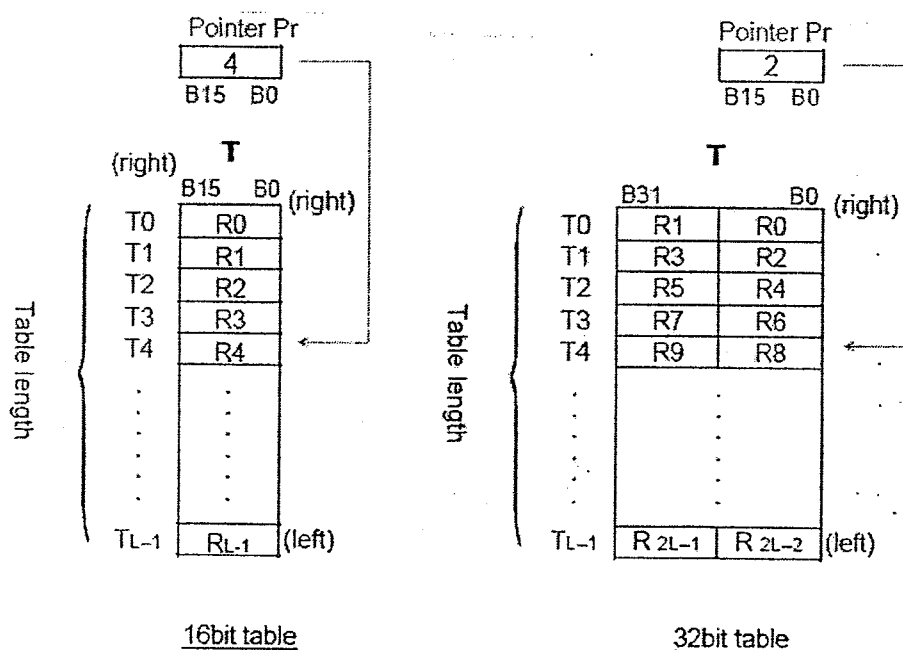
اجرا باشد. Pointer برای هر دو جدول ۱۶ و ۳۲ بیتی ، یک رجیستر ۱۶ بیتی است. رنج موثر Pointer، $0 \sim L-1$ ،

است.

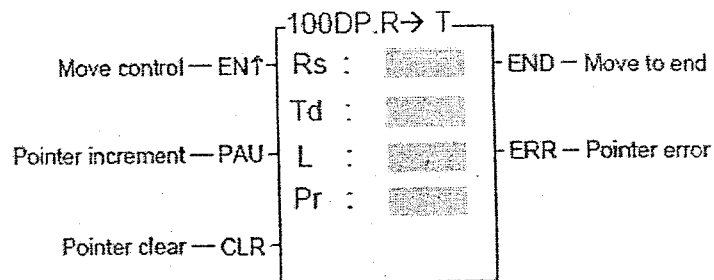
در هنگام عملیات جدولی، عملیاتی مانند شیفت به راست یا چپ، چرخش به راست یا چپ به این ترتیب مشخص

می شود که جهت به سمت رجیستر بالاتر باشد، چپ گفته می شود و اگر جهت به سمت رجیستر پایین تر باشد

راست گفته می شود.



100.REGISTER TO TABLE MOVE



هرگاه 'EN' از ۰ به ۱ تغییر کند:

محتوای رجیستر مبدا (Rs) به داخل رجیستری از جدول ریخته می شود که به آن اشاره می کند.

در رجیستر شروع جدول قرار می گیرد به (L).

این تابع قبل از اجرا، ابتدا سیگنال ورودی 'CLR' را چک می کند. اگر 'CLR' مقدار 'Pr' را چک

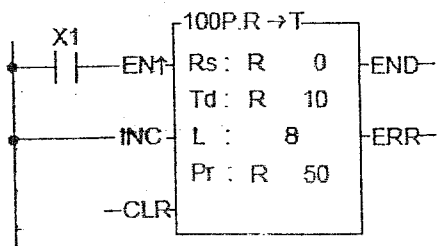
می کند. اگر مقدار 'Pr' به L-1 رسیده بود (یعنی به آخرین رجیستر جدول اشاره می کرد) خروجی 'END'، ۱

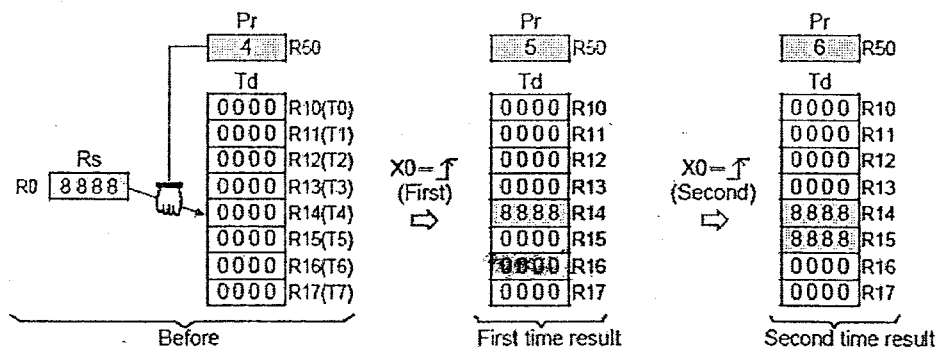
می شود و اجرای این تابع پایان می یابد.

اگر مقدار 'Pr' کمتر از L-1 باشد، سپس مجدداً 'INC' را چک می کند. اگر 'INC'، ۱ باشد، مقدار 'Pr' افزایش

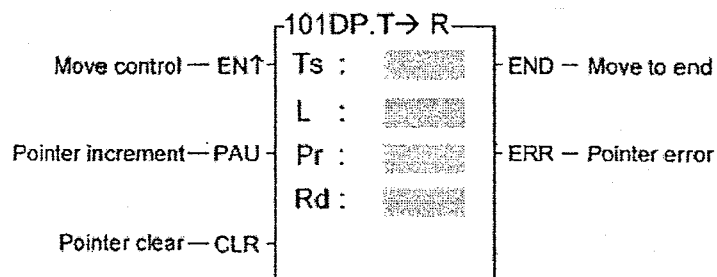
می یابد.

مثال:



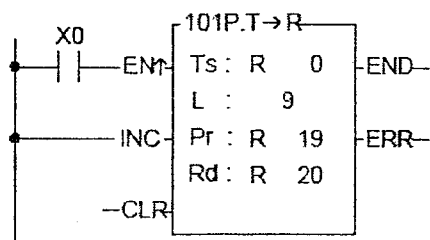


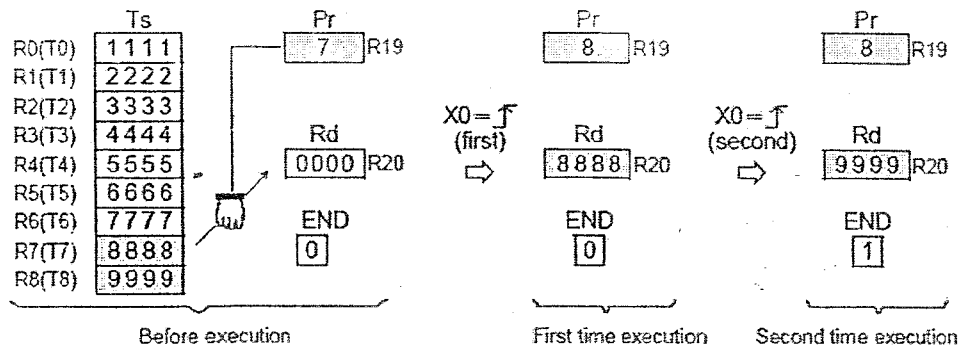
101. TABLE TO REGISTER MOVE



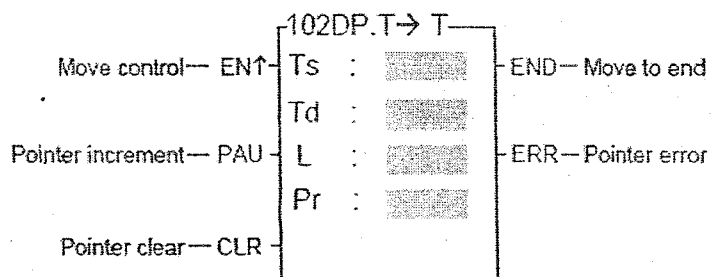
این تابع بر عکس تابع قبل عمل کرده و محتویات یک رجیستر از جدول مورد نظر را به رجیستر مقصد منتقل می کند.

مثال:





102.TABLE TO TABLE MOVE



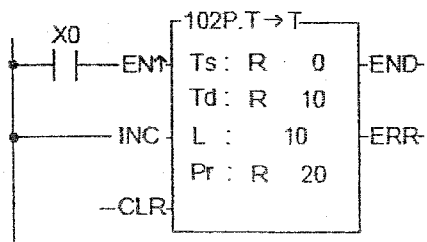
در Ts، رجیستر شروع جدول مبدا قرار می گیرد و

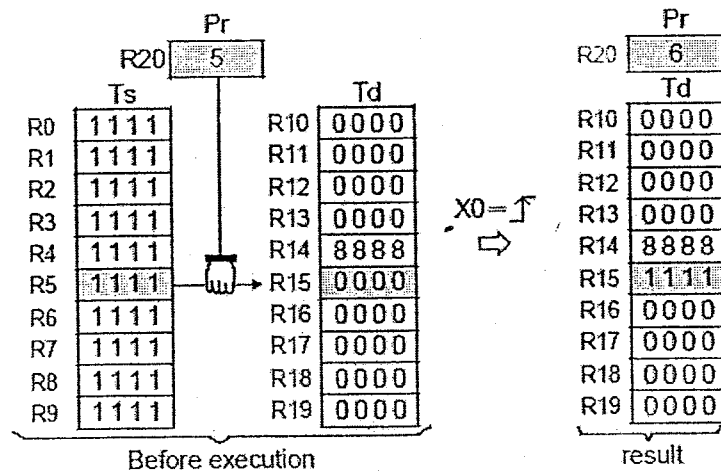
در Td، رجیستر شروع جدول مقصد قرار می گیرد.

L، طول جدول مقصد و مبدا را مشخص می کند.

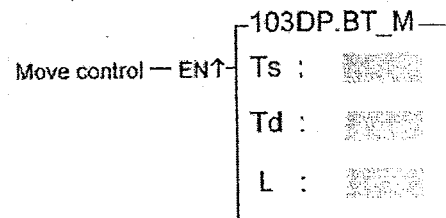
هرگاه 'EN' از ۰ به ۱ تغییر کند:

محتویات آن رجیستری از جدول مبدا که Pr به آن اشاره می کند به رجیستر معادلش در جدول مقصد منتقل می شود.





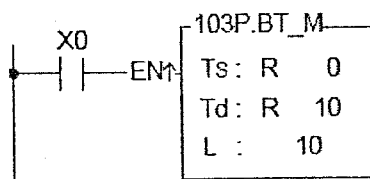
103.BLOCK TABLE MOVE

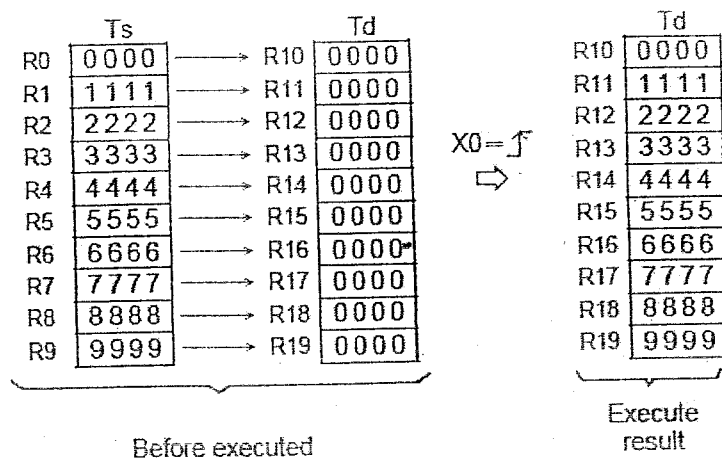


هر گاه 'EN' به ۱ تغییر کند:

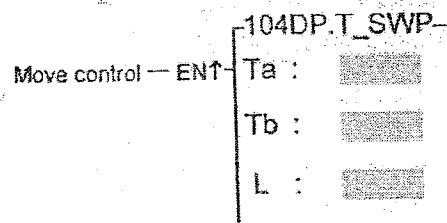
محتویات رجیسترهای جدول مبدا به داخل رجیسترهای معادلشان در جدول مقصد کپی می شوند.

مثال:





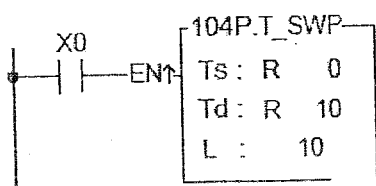
104.BLOCK TABLE SWAP

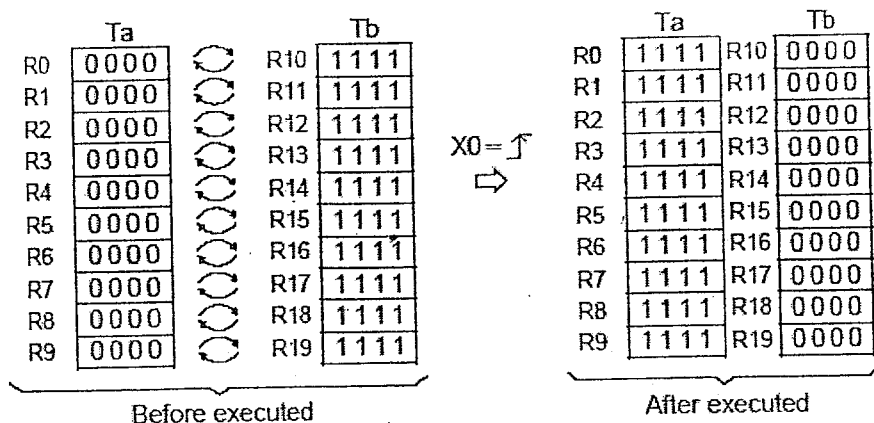


هرگاه EN از 0 به 1 تغییر کند:

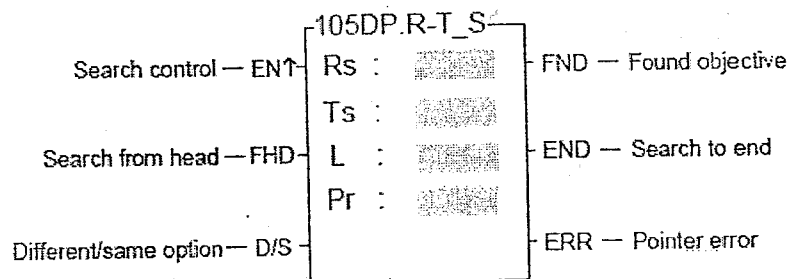
محتویات رجیسترهای جدول a با محتویات رجیسترهای معادلشان در جدول b، جابجا می شوند.

مثال:





105. REGISTER TO TABLE SEARCH



هرگاه 'EN' از ۰ به ۱ تغییر کند:

وقتی $FHD = 1$ با مقدار Pointer (Pr) به $L-1$ رسیده است، جستجو از اولین رجیستر جدول Ts آغاز می شود.

وقتی $FHD = 0$ و مقدار Pr کمتر از $L-1$ باشد، جستجو از اولین رجیستر بعد از جایی که Pr اشاره می کند، شروع می شود.

وقتی $D/S = 1$ ، جستجو برای یافتن اولین رجیستری که محتوای آن با Rs متفاوت باشد صورت می گیرد.

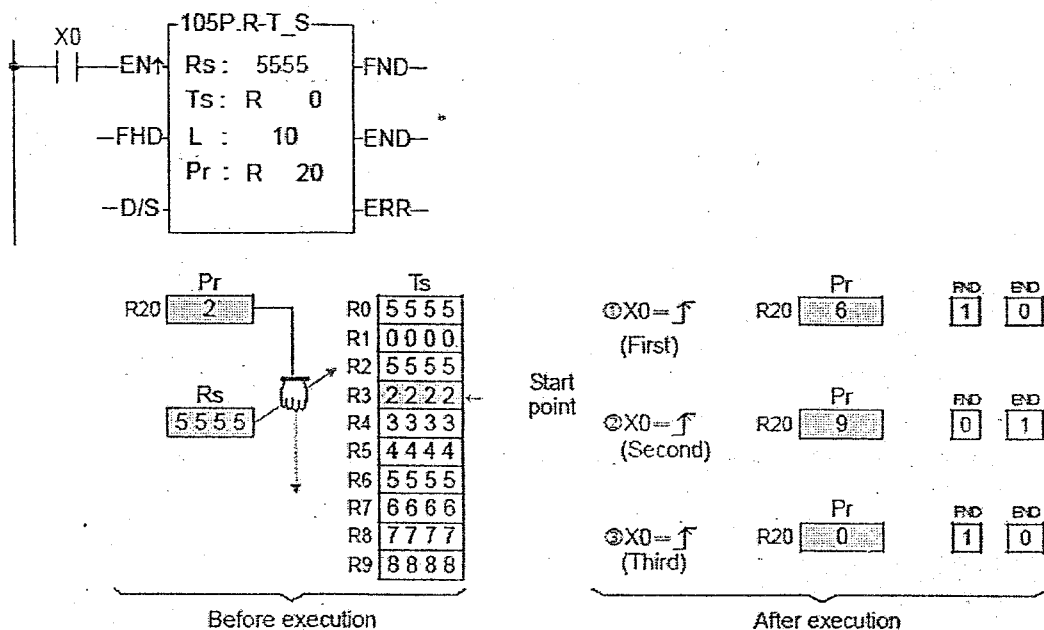
وقتی $D/S = 0$ ، جستجو برای یافتن اولین رجیستری که محتوای آن با Rs یکسان باشد صورت می گیرد.

بعد از یافتن اطلاعات مورد نظر، جستجو متوقف می شود و خروجی 'FND'، ۱ می شود.

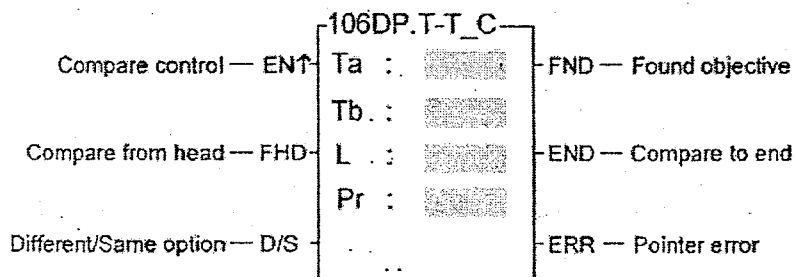
وقتی Pr به $L-1$ رسید چه محتوای مورد نظر را یافته باشد چه نیافته باشد، 'END' فعال شده و جستجو متوقف

می شود.

مثال:

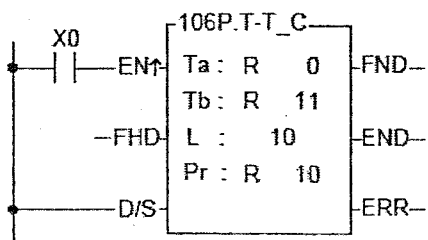


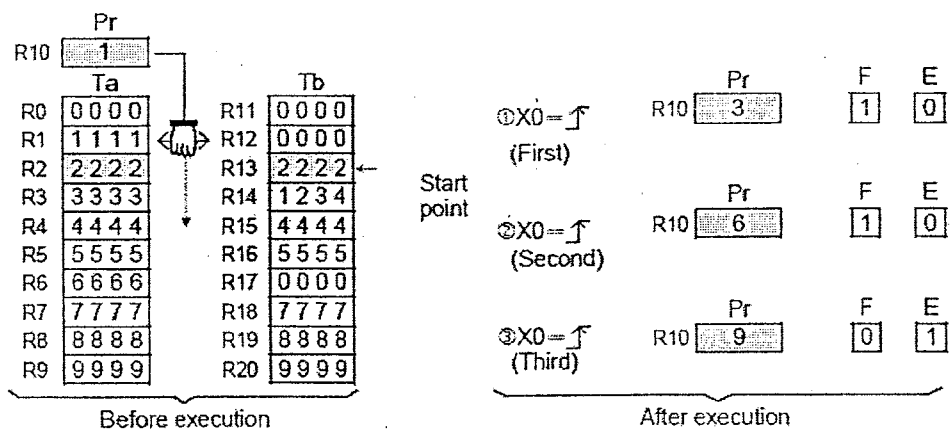
106.TABLE TO TABLE COMPARE



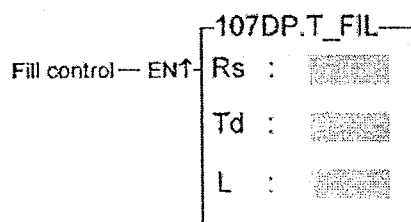
این تابع مانند تابع قبل است با این تفاوت که محتوای رجیسترهای معادل از دو جدول Ta و Tb با هم مقایسه

می شوند.





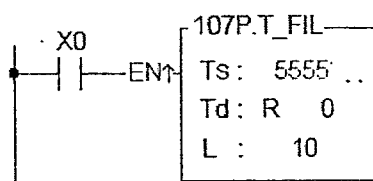
107.TABLE FILL

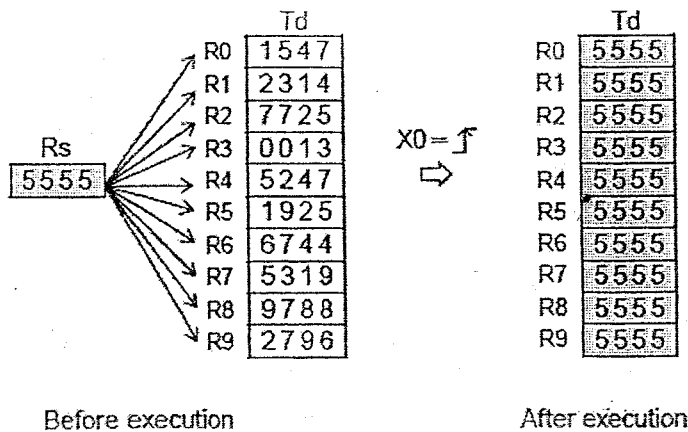


هر گاه 'EN' از ۰ به ۱ تغییر کند:

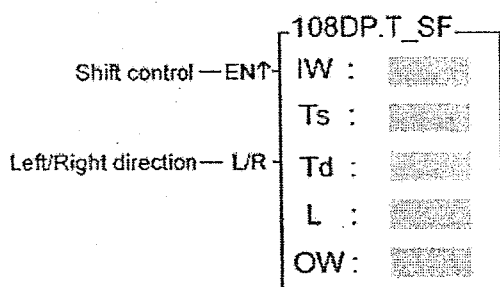
محتوای رجیستر Rs، تمام رجیسترهای جدول Td را پر می کند.

کاربرد اصلی این تابع در صفر کردن کل یک جدول یا یکی کردن محتوای تمام رجیسترهای یک جدول است.





108.TABLE SHIFT



هرگاه EN^{*} از ۰ به ۱ تغییر کند

محتوای رجیسترهای جدول Ts یک رجیستر به سمت چپ یا راست شیفت پیدا می کند. وقتی $L/R = 1$ شیفت به

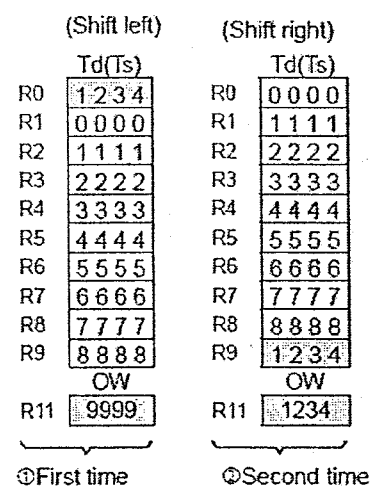
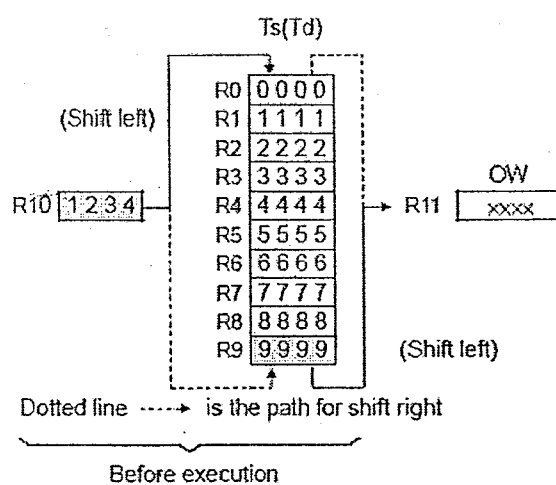
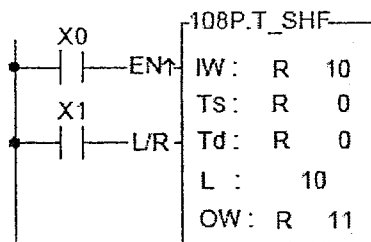
چپ صورت می گیرد و وقتی $L/R = 0$ شیفت به راست صورت می گیرد.

محتوای رجیستری که پس از اعمال شیفت از جدول خارج می شود، به رجیستر مشخص شده در OW منتقل

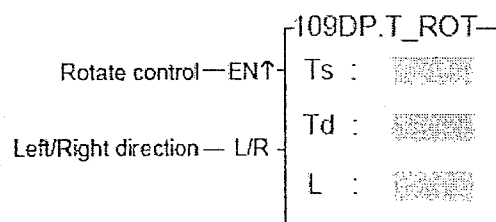
می شود.

جدول شیفت داده شده می تواند در جدول Td یا خود Ts ذخیره شود.

مثال:



109.TABLE ROTATE



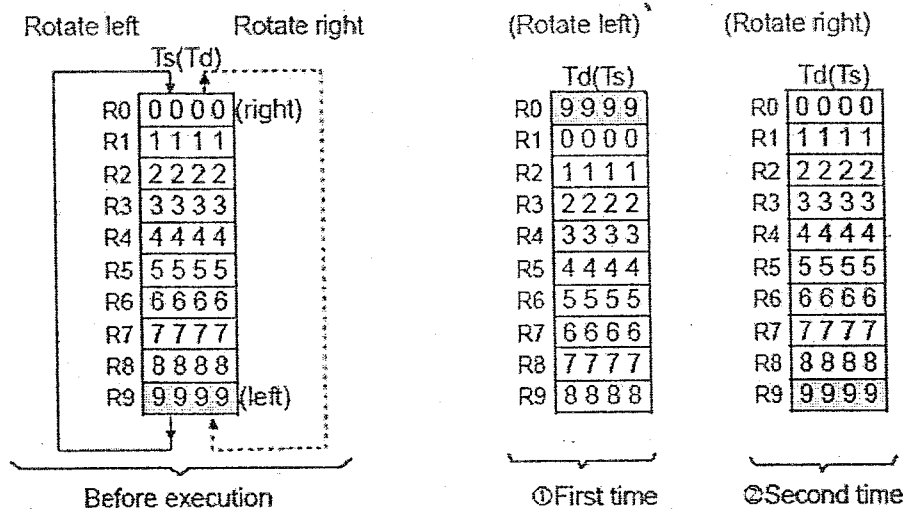
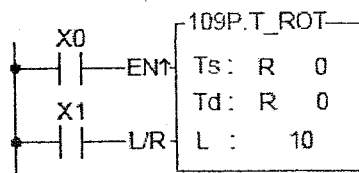
هرگاه 'EN' از ۰ به ۱ تغییر کند:

محتوای رجیسترهای جدول Ts، یک رجیستر به سمت چپ یا راست می چرخند و نتیجه در Td ذخیره می شود.

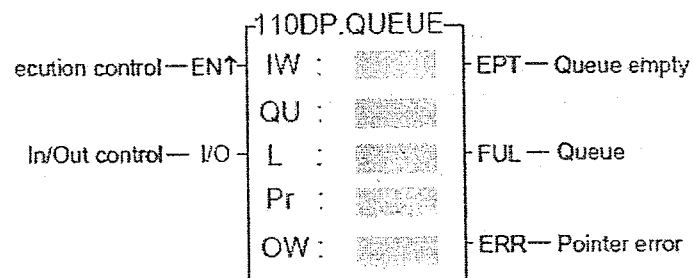
وقتی $L/R=1$ ، چرخش به چپ صورت می گیرد

وقتی $L/R=0$ ، چرخش به راست صورت می گیرد.

مثال:



110. QUENE



QUENE نیز نوعی جدول است. با این تفاوت که شماره رجیسترهای جدول معمولی از

0~L-1 است اما شماره جدول های QUENE 1~L است و Pr=0 برای نمایش خالی بودن QUENE استفاده می شود.

در QUENE:

همان داده ای که ابتدا در QUENE قرار می گیرد، (عمل Push)، اولین داده ای خواهد بود که از QUENE برداشته می شود. (عمل Pop)

یک QUENE از L رجیستر ۱۶ یا ۳۲ بیتی تشکیل شده که از رجیستر مشخص شده در QU شروع می شوند.

هرگاه $I/O = 1$ ، محتوای IW به داخل QUENE push می شود و

هرگاه $I/O = 0$ ، اولین داده ی درون QUENE (قدیمی ترین) به داخل OW pop می شود.

محتوای IW همیشه به داخل اولین رجیستر QUENE push می شود و پس از هر push یکی به

مقدار pointer (Pr) اضافه می شود و همیشه هنگام pop کردن، قدیمی ترین محتوا از داخل QUENE به OW،

pop می شود و یکی از مقدار Pr کم می شود.

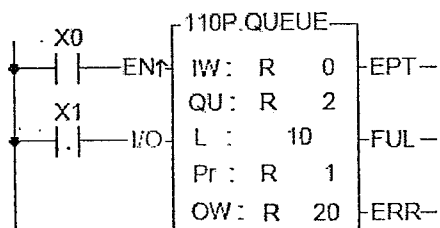
اگر QUENE خالی از داده باشد ($Pr=0$ باشد)، $EPT=1$ خواهد شد.

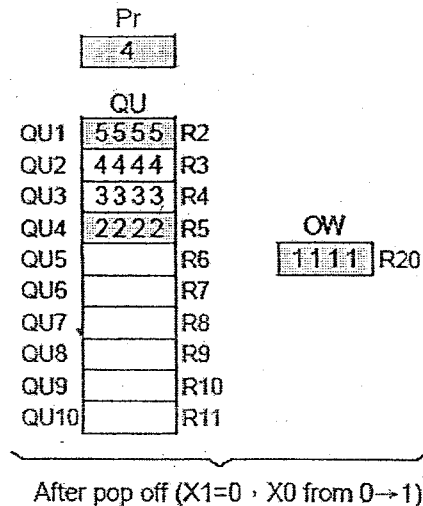
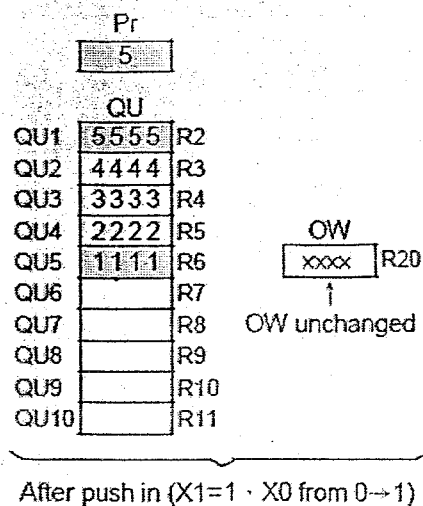
اگر QUENE کاملاً پر باشد (Pr به رجیستر L از QUENE اشاره کند)، $FUL=1$

خواهد شد.

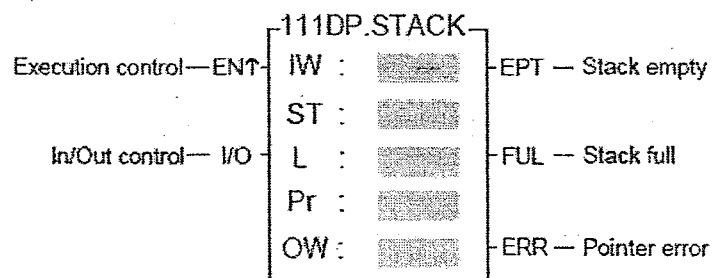
اگر مقدار Pr فراتر از رنج 0~L داده شود $ERR=1$ خواهد شد.

مثال:





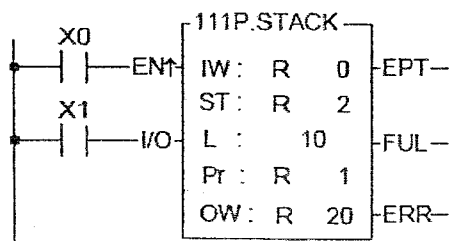
111.STACK

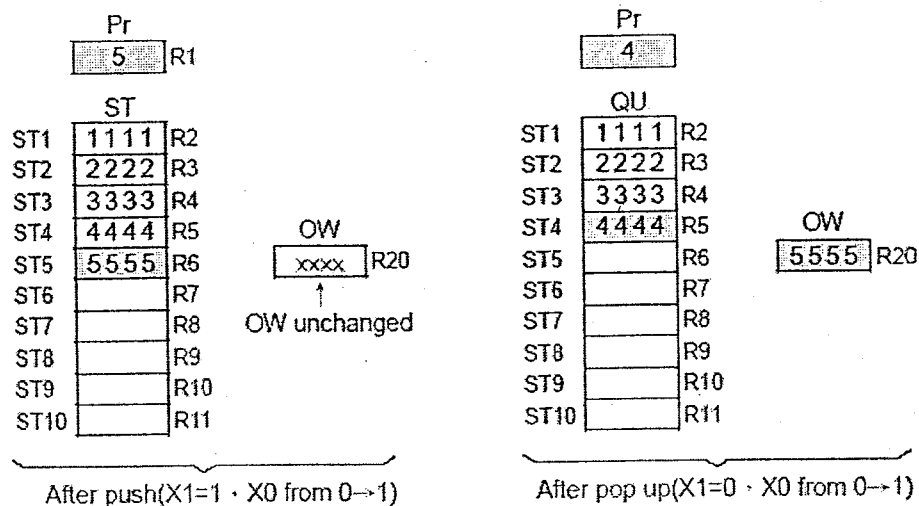


STACK نیز جدولی مانند QUEUE است یا یک تفاوت.

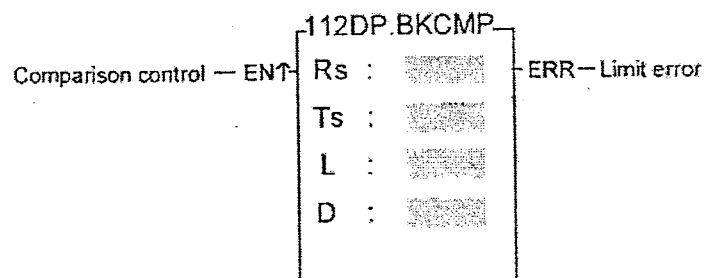
در QUEUE اولین داده ای که push می شود اولین داده ای خواهد بود که pop می شود اما در

STACK، آخرین داده ای که push می شود اولین داده ای خواهد بود که pop می شود.





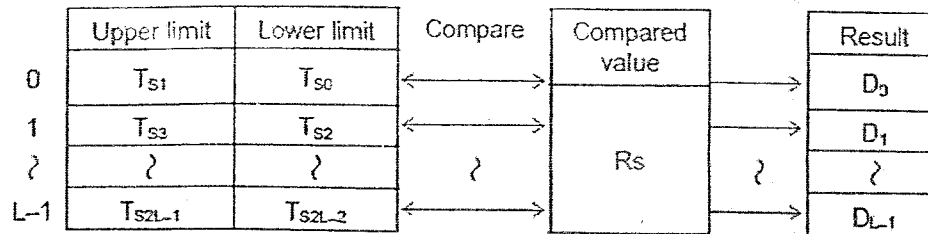
112.DRUM



هرگاه 'EN' از ۰ به ۱ تغییر کند:

مقایسه بین محتوای R_s و یک جفت از رجیسترهای جدول که از T_s شروع می شوند، صورت می گیرد. (به عنوان مثال یک جفت شامل Ts_0 که به آن حد پایین و Ts_1 که به آن حد بالایی گویند، می شود). بعد از مقایسه بین R_s و اولین جفت، نتیجه در D_0 ذخیره می شود. سپس، مقایسه بین R_s و دومین جفت صورت می گیرد و نتیجه در D_1 ذخیره می شود و...

اگر مقدار R_s در محدوده حد بالا و حد پایین یک جفت باشد، بیت D متناظر با آن جفت، ۱ می شود در غیر این صورت ۰ خواهد بود.



L در این تعداد جفت ها را مشخص می کند.

مقایسه تا زمانی ادامه می یابد که Rs با تمام جفت ها مقایسه شود.

وقتی رجیستر $M1975=0$ است و در جفتی، حد بالا کوچک تر از حد پایین باشد، 'ERR' فعال می شود و نتیجه

مقایسه برای آن جفت، ۰ خواهد بود.

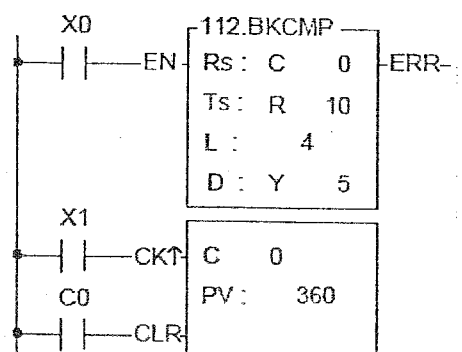
وقتی رجیستر $M1975=1$ است محدوده دیتی برای بزرگتر یا کوچکتر بودن حد بالا نسبت به حد پایین وجود

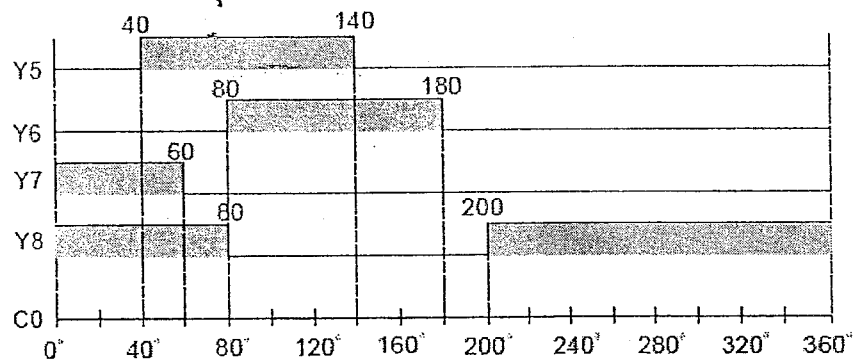
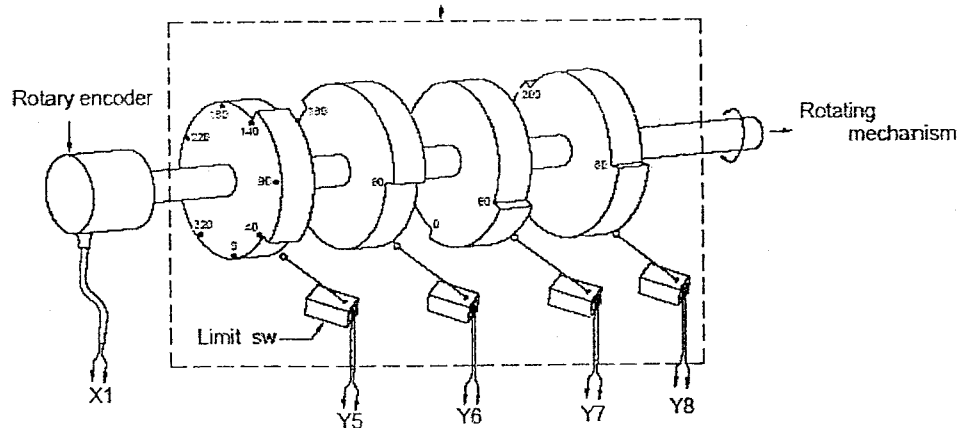
ندارد. این حالت می تواند برای سوئیچ DRUM الکترونیکی چرخنده 360° ای، کاربرد داشته باشد.

این تابع در واقع سوئیچی برای محورهای الکترونیکی محسوب می شود که اگر به همراه برنامه INTERRUPT

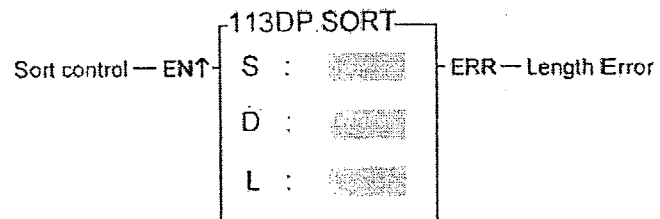
و دستور IMMEDIATE I/O به کار گرفته شود، می تواند محور الکترونیکی دقیقی به دست دهد.

مثال:





113.DATA SORTING



هرگاه 'EN' از ۰ به ۱ تغییر کند:

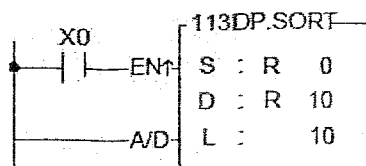
این تابع رجیسترها به تعداد L را که از S شروع می شوند، به صورت افزایشی یا کاهش مرتب می کند و نتیجه را در D ذخیره می کند.


اگر $A/D=1$ رجیسترها به صورت افزایشی مرتب می شوند و

اگر $A/D=0$ رجیسترها به صورت کاهش مرتب می شوند.

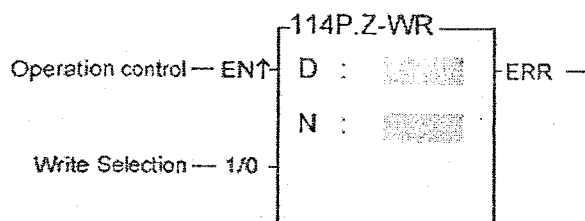
L باید در رنج 2~127 باشد در غیر این صورت $ERR=1$.

مثال:



S			D	
R0	1547	X0 = 	R10	0013
R1	2314		R11	1547
R2	7725		R12	1925
R3	0013		R13	2314
R4	5247		R14	2796
R5	1925		R15	5247
R6	6744		R16	5319
R7	5319		R17	6744
R8	9788		R18	7725
R9	2796		R19	9788
Before			After	

114.ZONE WRITE

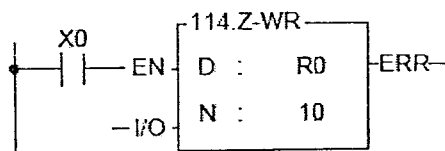


هرگاه 'EN' از ۰ به ۱ تغییر کند:

این تابع N تعداد از رجیسترها که از D شروع می شوند را set (۱) یا reset (۰) می کند.

اگر ۱/۰ = ۱ باشد set می کند و اگر ۱/۰ = ۰ باشد reset می کند.

مثال:



در این مثال هرگاه $X0=1$ ، رجیسترهای $R0 \sim R9$ ، reset شده و ۰ می شوند.

توابع ماتریسی

Fun No.	Mnemonic	Functionality	Fun No.	Mnemonic	Functionality
120	MAND	Matrix AND	126	MBRD	Matrix Bit Read
121	MOR	Matrix OR	127	MBWR	Matrix Bit Write
122	MXOR	Matrix XOR	128	MBSHF	Matrix Bit Shift
123	MXNR	Matrix XNOR	129	MBROT	Matrix Bit Rotate
124	MINV	Matrix Inverse	130	MBCNT	Matrix Bit Count
125	MCMP	Matrix Compare			

یک ماتریس از ۲ یا چند رجیستر پی در پی ۱۶ بیتی تشکیل شده است. به تعداد رجیسترهای تشکیل دهنده

ماتریس، طول ماتریس می گویند و با (L) نمایش می دهند.

پس یک ماتریس $E \times 16$ بیت (آرایه) دارد و واحد اصلی اجرای این توابع، بیت می باشد.

توابع ماتریسی عمدتاً برای پردازش گسسته صورت می گیرد مانند انتقال، کپی، مقایسه، جستجو و غیره یک

آرایه به ماتریس یا ماتریس به ماتریس.

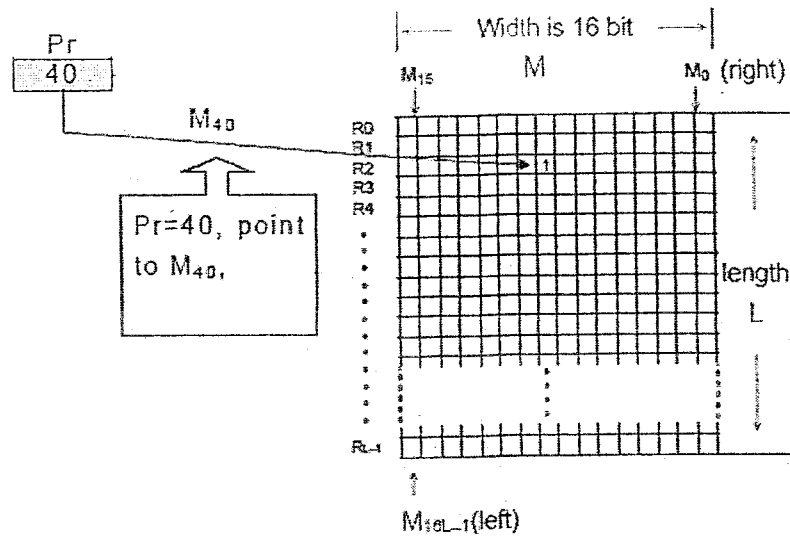
در میان این توابع، بیشتر آنها به یک اشاره گر (pointer) ۱۶ بیتی نیاز دارند تا به یک آرایه خاص در یک ماتریس

اشاره کنند.

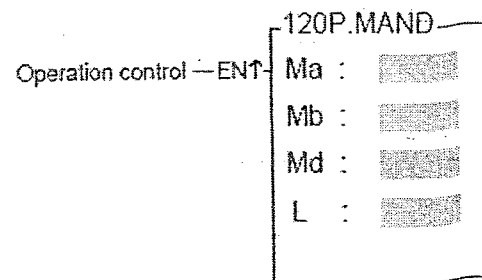
رنج موثر این pointer، $0 \sim 16L-1$ است.

جهت اجرای دستوراتی مانند شیفت و چرخش، حرکت به سمت بیت کم ارزش را جهت راست می دانیم و حرکت

به سمت بیت با ارزش تر را جهت چپ می دانیم.

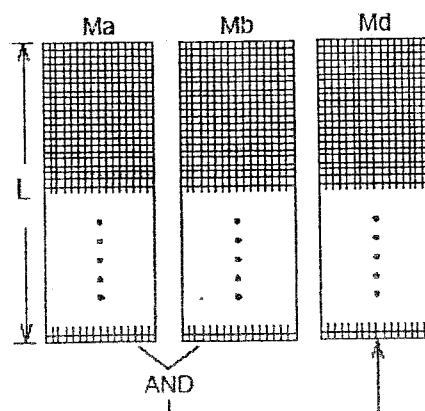


120.MATRIX AND

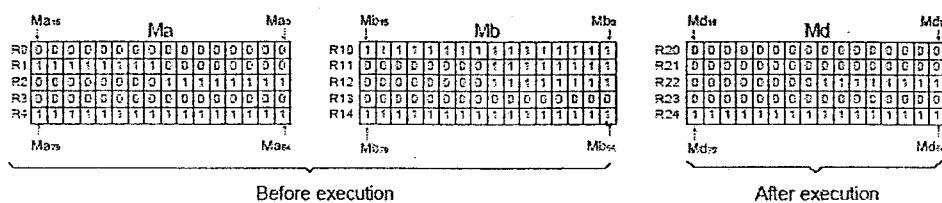
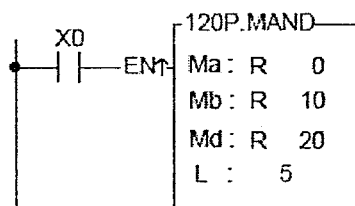


هرگاه 'ENT' از ۰ به ۱ تغییر کند:

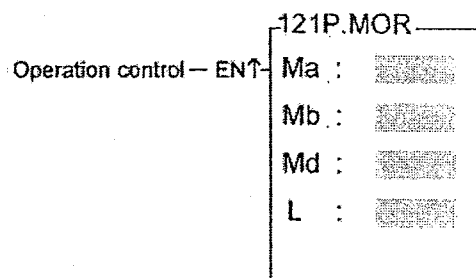
بیت های متناظر در ماتریس Ma و Mb را با یکدیگر AND کرده و نتیجه در Md ریخته می شود.



مثال:

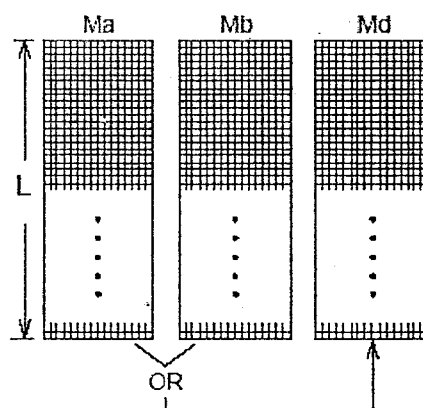


121.MATRIX OR

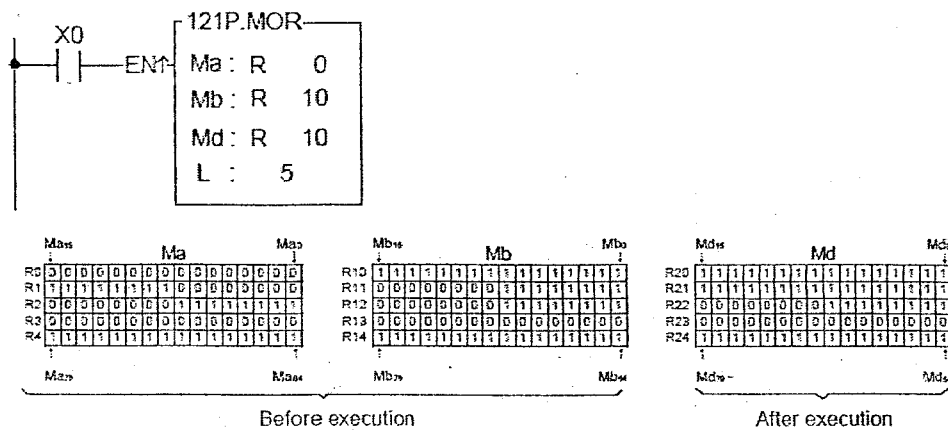


هرگاه 'EN' از ۰ به ۱ تغییر کند:

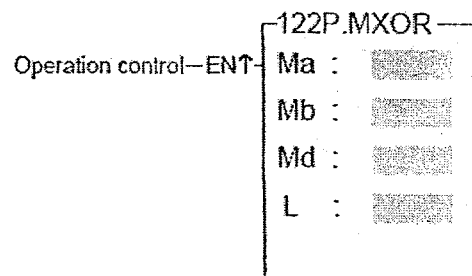
بیت های متناظر در ماتریس Ma و Mb را با یکدیگر OR کرده و نتیجه در Md ریخته می شود.



مثال:

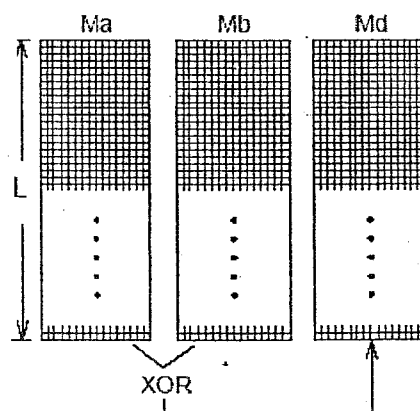


122.MATRIX XOR

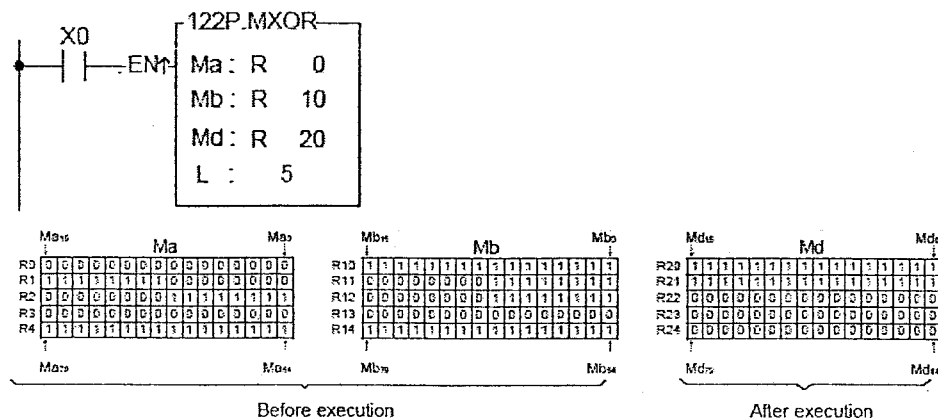


هرگاه 'ENT' از ۰ به ۱ تغییر کند:

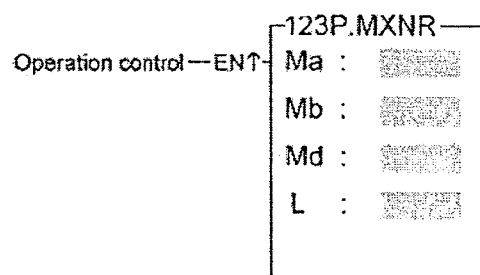
بیت های متناظر در ماتریس Ma و Mb را با یکدیگر XOR کرده و نتیجه در Md ریخته می شود.



مثال:

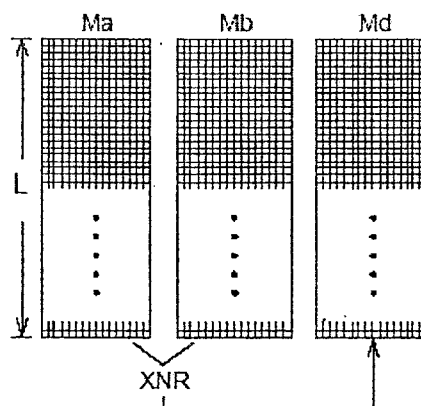


123.MATRIX XNOR

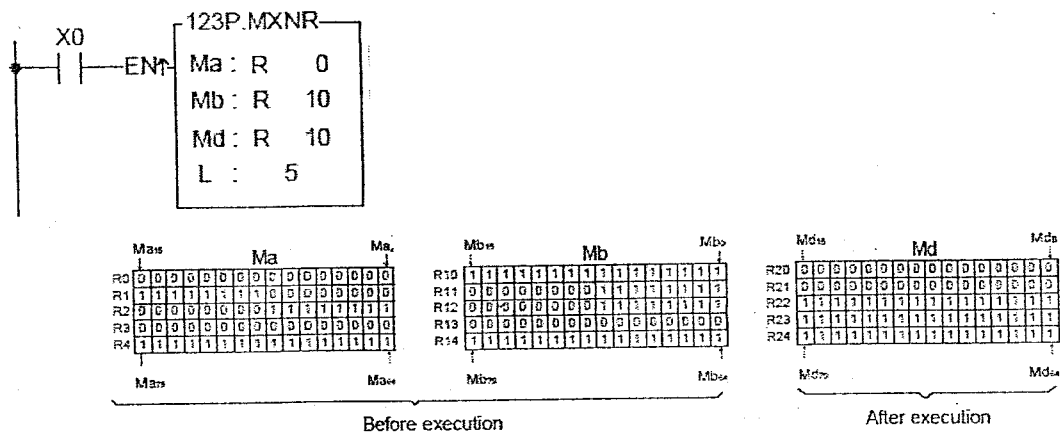


هرگاه 'EN' از ۰ به ۱ تغییر کند:

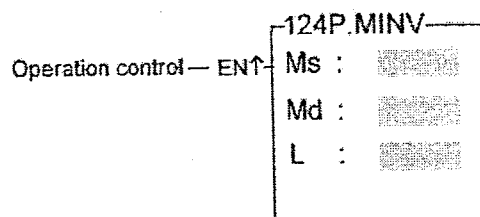
بیت های متناظر در ماتریس Ma و Mb را با یکدیگر XNOR کرده و نتیجه در Md ریخته می شود.



مثال:

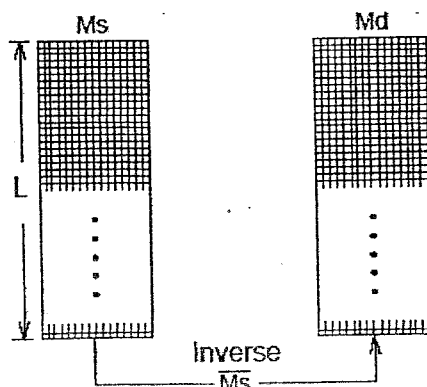


124.MATRIX INVERSE

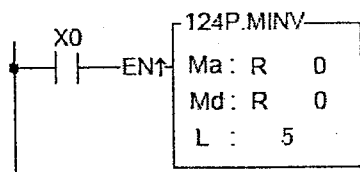


هرگاه 'ENT' از ۰ به ۱ تغییر کند:

تمام بیت های ماتریس Ms معکوس می شوند (۰ ها ۱ شده و ۱ ها ۰ می شوند) و نتیجه در Md ذخیره می شود.



مثال:



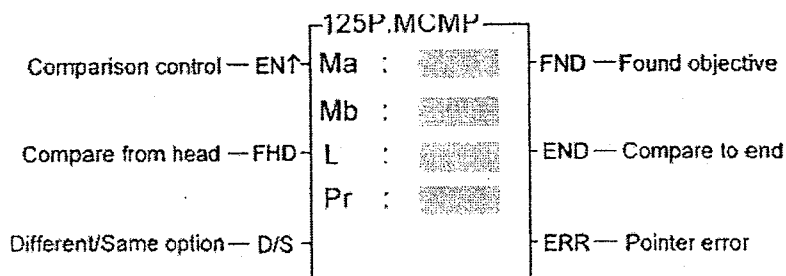
	MS ₁₅	MS ₁₄	MS ₁₃	MS ₁₂	MS ₁₁	MS ₁₀	MS ₉	MS ₈	MS ₇	MS ₆	MS ₅	MS ₄	MS ₃	MS ₂	MS ₁	MS ₀
R0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
R2	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1
R3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R4	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Before execution

	MD ₁₅	MD ₁₄	MD ₁₃	MD ₁₂	MD ₁₁	MD ₁₀	MD ₉	MD ₈	MD ₇	MD ₆	MD ₅	MD ₄	MD ₃	MD ₂	MD ₁	MD ₀
R0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
R1	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
R2	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
R3	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
R4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

After execution

125.MATRIX COMPARE



هرگاه 'EN' از ۰ به ۱ تغییر کند:

وقتی $FHD=1$ یا مقدار Pr به $16L-1$ رسیده است، مقایسه بین دو بیت متناظر از ماتریس های Ma و Mb از

آغاز ماتریس ها (اولین بیت)، شروع می شود.

وقتی $FHD=0$ و مقدار Pr کمتر از $16L-1$ باشد، مقایسه از اولین بیت، بعد از جایی که Pr اشاره می کند، شروع

می شود.

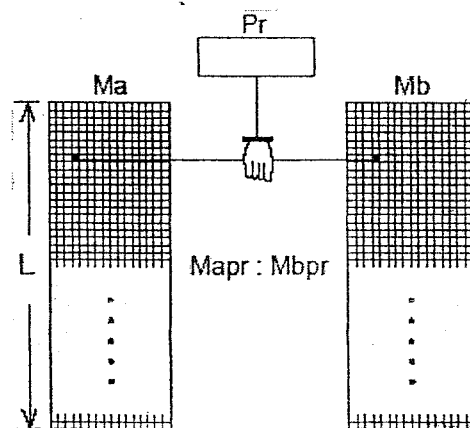
وقتی $D/S=1$ ، مقایسه برای یافتن اولین جفت بیت متناظری که محتوای آنها متفاوت باشد صورت می گیرد.

وقتی $D/S=0$ ، مقایسه برای یافتن اولین جفت بیت متناظری که محتوای آنها یکسان باشد صورت می گیرد.

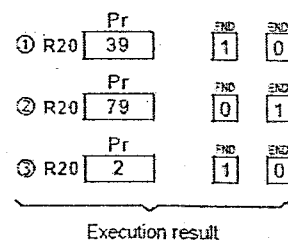
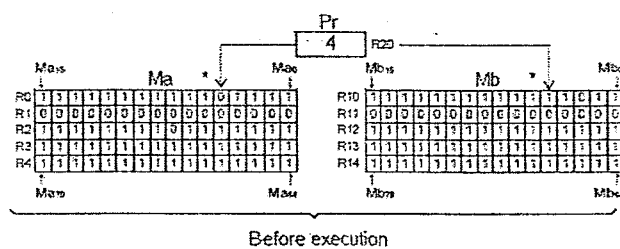
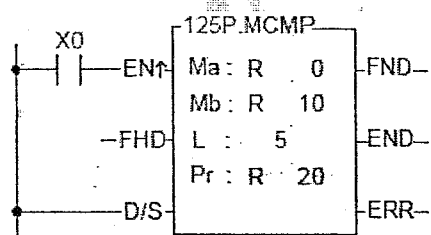
بعد از یافتن اطلاعات مورد نظر، مقایسه متوقف می شود و خروجی 'FND' ۱ می شود.

وقتی Pr به 16L-1 رسید، چه بیت های مورد نظر را یافته باشد چه نیافته باشد، 'END' فعال شده و مقایسه

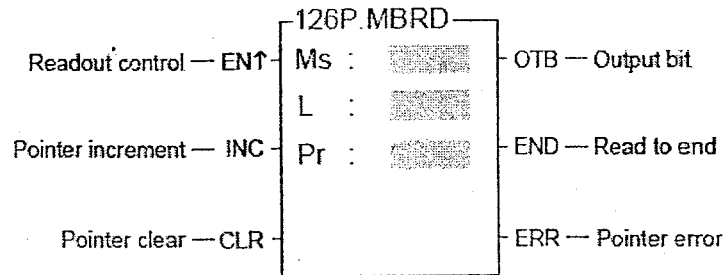
متوقف می شود.



مثال:



126.MATRIX BIT READ



هرگاه 'EN' از ۰ به ۱ تغییر کند:

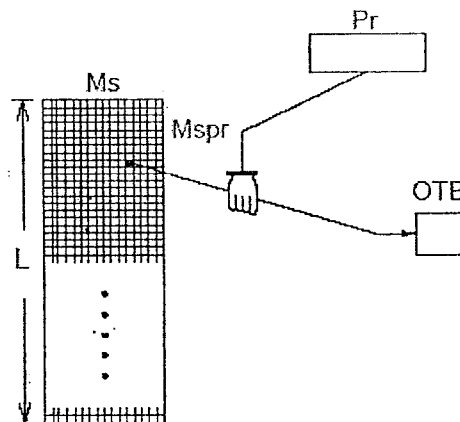
مقدار بیت Pr به آن اشاره می کند، در خروجی 'OTB' ظاهر می شود.

این تابع قبل از اجرا، ابتدا 'CLR' را چک می کند، اگر $CLR=1$ ، مقدار Pr را ۰ می کند سپس بیت را به خروجی 'OTB' می فرستد. بعد از آن دوباره مقدار Pr را چک می کند. اگر مقدار Pr به $16L-1$ رسیده بود (یعنی به آخرین بیت ماتریس اشاره می کرد) خروجی 'END' ۱ می شود و اجرای این تابع پایان می یابد.

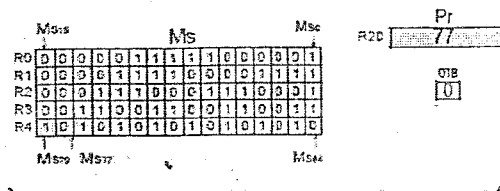
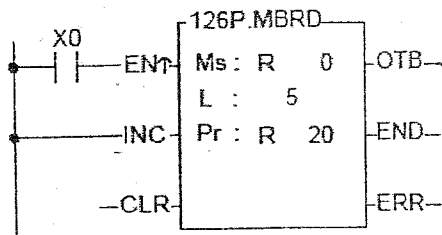
اگر مقدار Pr کمتر از $16L-1$ باشد، مجدداً 'INC' را چک می کند. اگر 'INC' ۱ باشد، مقدار Pr افزایش می یابد.

'CLR' می تواند بدون این که تحت تاثیر ورودی های دیگر باشد، مستقلاً اجرا شود.

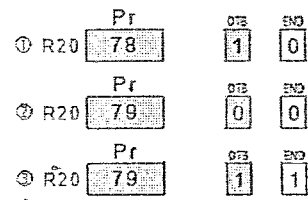
رنج موثر Pr ، $0 \sim 16L-1$ است، فراتر از این رنج $ERR=1$ خواهد شد و این تابع اجرا نمی شود.



مثال:

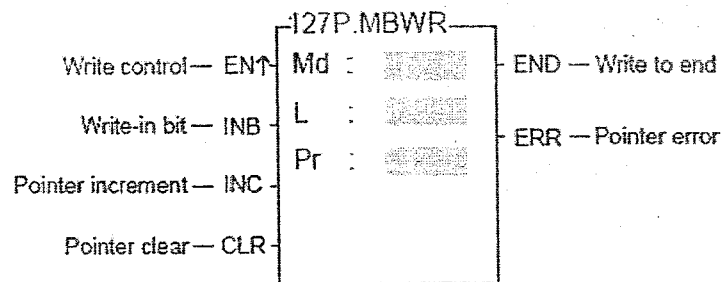


Before execution



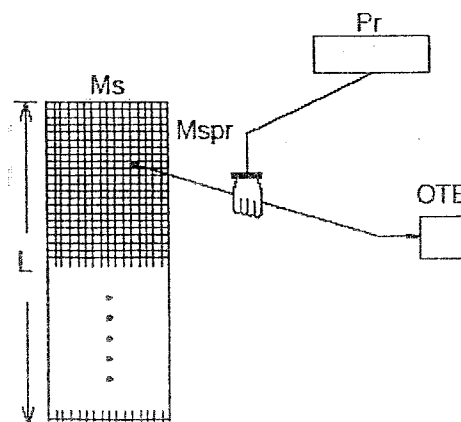
Execution result

127.MATRIX BIT WRITE

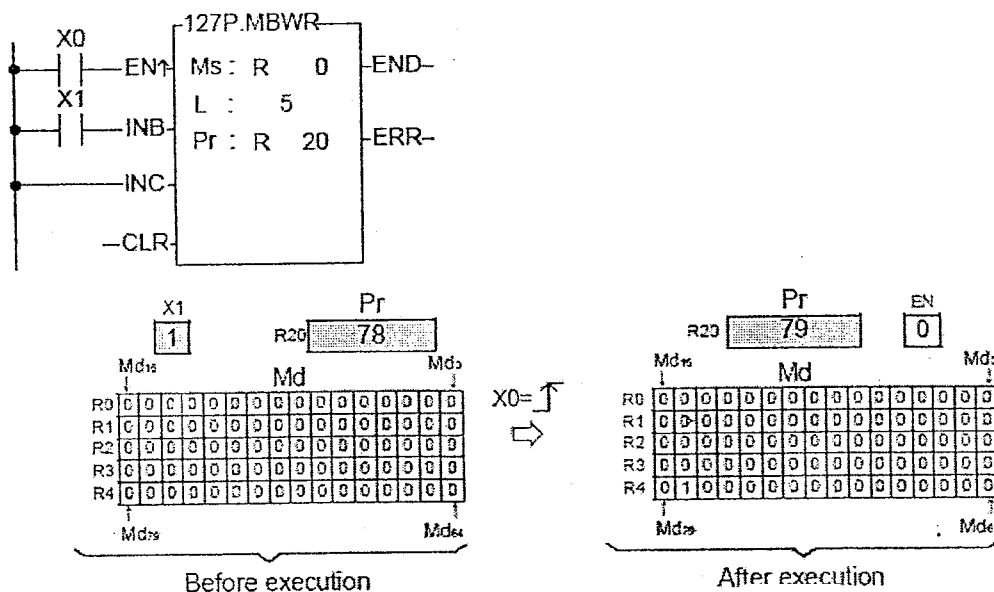


مانند تابع قبل است با این تفاوت که هنگام اجرا، بیت مشخص شده در ورودی 'INB' به روی بیتی که Pr اشاره

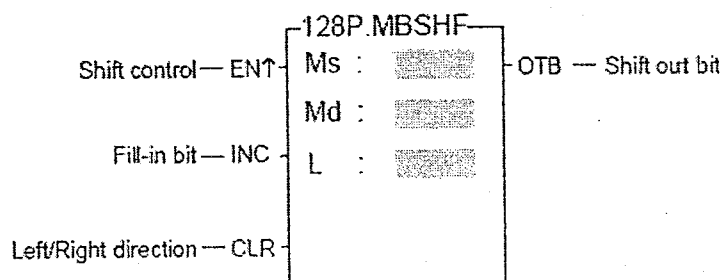
می کند، ریخته می شود.



مثال:



128.MATRIX BIT SHIFT



هرگاه 'EN' از ۰ به ۱ تغییر کند

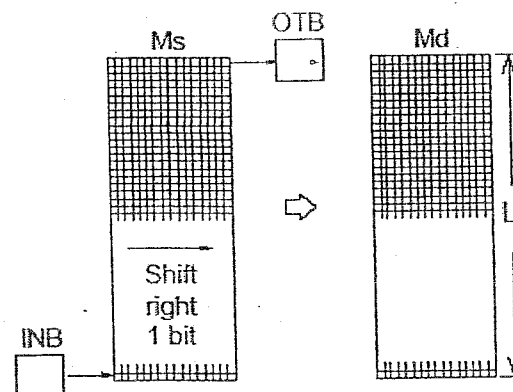
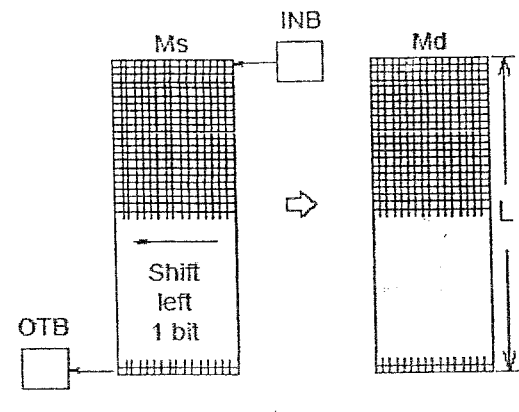
بیت های ماتریس Ms، ۱ بیت شیفت پیدا می کنند و نتیجه در Md ذخیره می شود.

اگر $L/R=1$ باشد، شیفت به چپ صورت می گیرد و

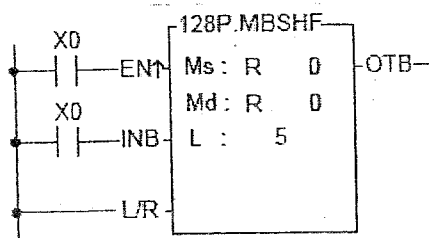
اگر $L/R=0$ باشد، شیفت به راست صورت می گیرد.

بیت خالی ایجاد شده بعد از اعمال شیفت، توسط 'INB' پر می شود، محتوای بیتی که پس از اعمال شیفت از

جدول خارج می شود، به خروجی 'OTB' منتقل می شود.



مثال:



	Ms ₁₅	Ms	Ms ₀
R0	0	0	0
R1	1	1	1
R2	1	1	1
R3	0	0	0
R4	0	1	1

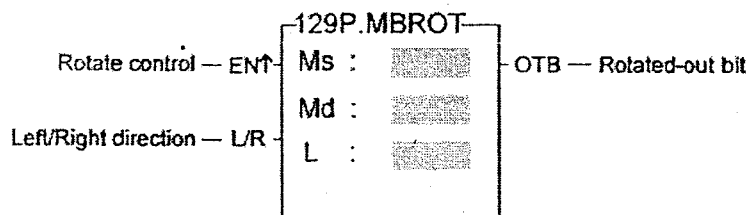
X0 = 1
⇒

	Md ₁₅	Md	Md ₀
R0	0	0	0
R1	1	1	1
R2	1	1	1
R3	0	0	0
R4	1	1	1

Before execution

After execution

129.MATRIX BIT ROTATE



هرگاه 'EN' از ۰ به ۱ تغییر کند:

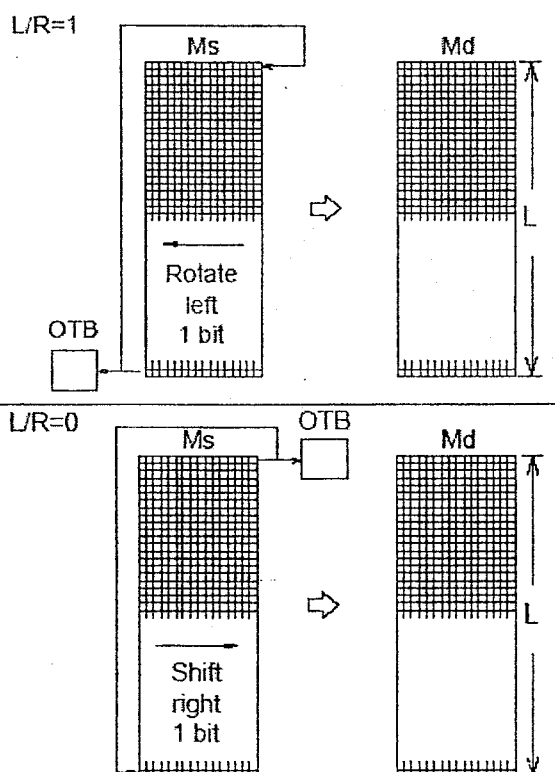
بیت های ماتریس Ms، یک بیت به سمت راست یا چپ می چرخند و نتیجه در ماتریس Md ذخیره می شود.

وقتی $L/R=1$ ، چرخش به چپ صورت می گیرد.

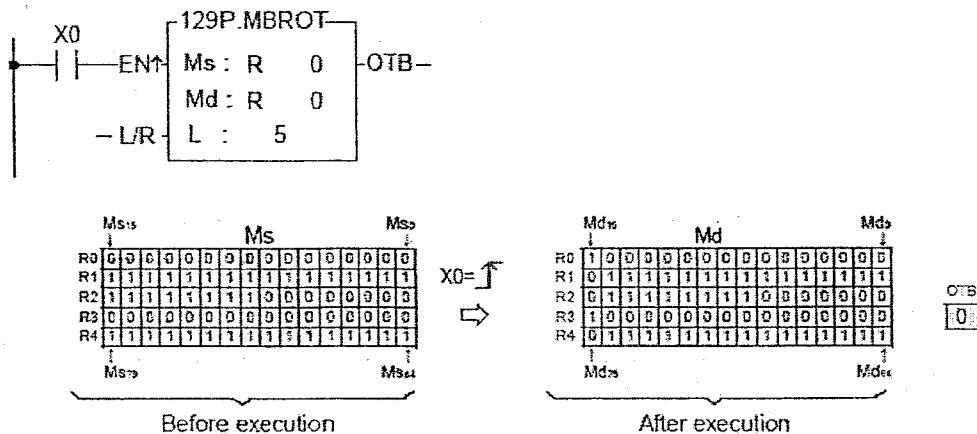
وقتی $L/R=0$ ، چرخش به راست صورت می گیرد.

یک کپی از بیتی که بر اثر چرخش از یک سر ماتریس خارج شده و در سنز دیگر آن قرار

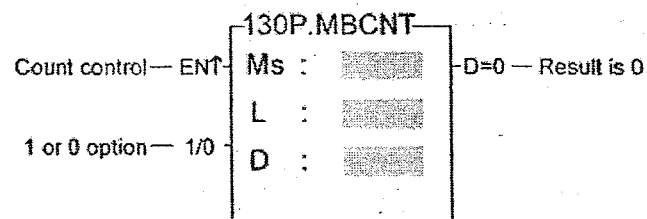
می گیرد، به خروجی OTB نیز می رود.



مثال:



130.MATRIX BIT STATUS COUNT



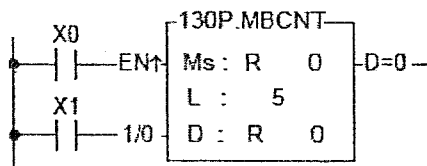
هرگاه 'EN' از ۰ به ۱ تغییر کند:

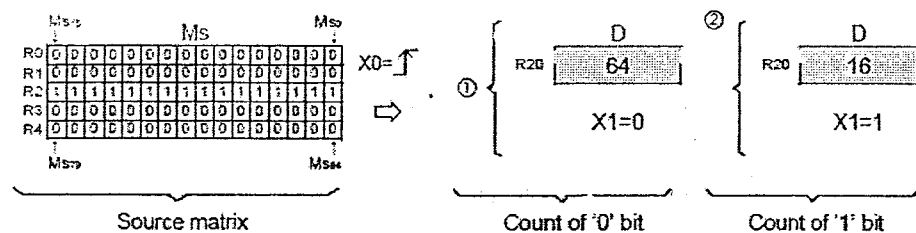
اگر $1/0=1$ باشد، تعداد بیت های ۱ موجود در ماتریس Ms را شمرده و تعداد در D ذخیره می شود.

اگر $1/0=0$ باشد، تعداد بیت های ۰ موجود در ماتریس Ms را شمرده و تعداد در D ذخیره می شود.

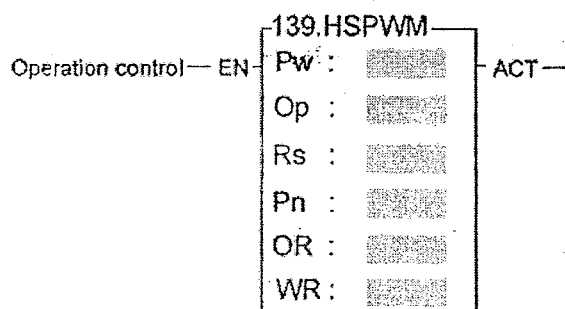
اگر هیچ تعداد از آن بیت مورد نظر موجود نباشد، $D=0$ ، ۱ خواهد شد.

مثال:





139.HSPWM



وقتی $EN=1$ ، این تابع پالسی را به خروجی می فرستد که فرکانس و پریود آن بر اساس فرمول های زیر

محاسبه می شود

در PW خروجی مورد نظر انتخاب می شود:

($0=Y0$, $1=Y2$, $2=Y4$, $3=Y6$)

در Rs، رزولوشن یک دوره پالس تعیین می شود:

$0=1/100$ $1=1/1000$

(۱) اگر $Rs=1/100$:

$$f_{pwm} = \frac{184320}{(P_n + 1)}$$

(۲) اگر $Rs=1/1000$:

$$f_{pwm} = \frac{18432}{(P_n + 1)}$$

پریود:

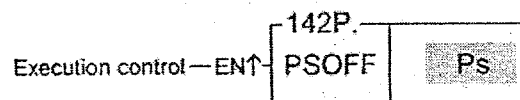
$$T(\text{Period}) = \frac{1}{f_{\text{pwm}}}$$

در OR عرض یک پالس در هر دوره مشخص می شود.

برای $Rs=1/100$: $OR=0\sim100$

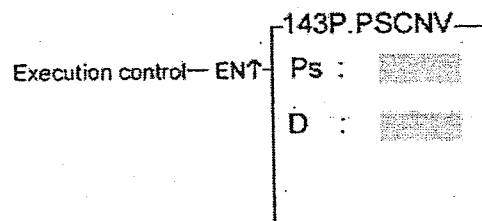
برای $Rs=1/1000$: $OR=0\sim1000$

142.STOP PULSE OUTPUT



این تابع فرستادن پالس به خروجی را متوقف می کند.

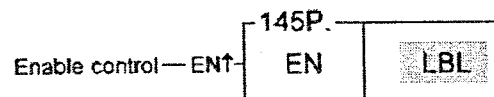
143.PSCNV



این تابع مقدار پالس جاری را به مقداری برای نمایش تبدیل می کند. بر حسب mm تا درجه، inch و پالس.

این تابع فقط وقتی تابع ۱۶۰ اجرا می شود، می تواند تبدیل را انجام دهد.

145.ENABLE OF INTERRUPT



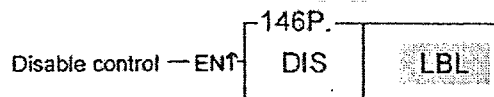
هرگاه 'EN' از ۰ به ۱ تغییر کند:

این تابع برنامه فرعی یا اینترپتی را فعال می کند که بر حسب (LABEL) آن در دستور نام برده شده است.

LABEL اینترپت های مختلف در جدول زیر آورده شده است.

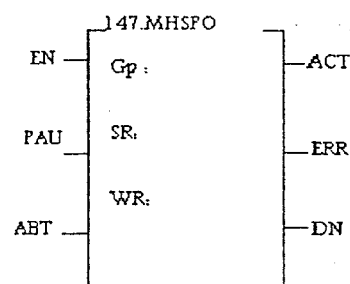
LBL name	Description	LBL name	Description	LBL name	Description
HSTA I	HSTA High speed counter interrupt	X4+I	X4 positive edge interrupt	X10+I	X10 positive edge interrupt
HSC0 I	HSC0 High speed counter interrupt	X4-I	X4 negative edge interrupt	X10-I	X10 negative edge interrupt
HSC1 I	HSC1 High speed counter interrupt	X5+I	X5 positive edge interrupt	X11+I	X11 positive edge interrupt
HSC2 I	HSC2 High speed counter interrupt	X5-I	X5 negative edge interrupt	X11-I	X11 negative edge interrupt
HSC3 I	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				

146.DISABLE OF INTERRUPT



این تابع برنامه فرعی یا اینترپتی را که LABEL آن در تابع نام برده شده است، غیر فعال می کند.

147.MULTI-AXIS HIGH SPEED PULSE OUTPUT



این تابع برای پشتیبانی از interpolation خطی برای کنترل حرکت چند محوره، استفاده می شود که شامل

برنامه حرکتی که توسط برنامه متنی نوشته شده است، می شود. ما هر نقطه از مکان را یک استپ (step)

می‌نامیم. هر استپ 15 رجیستر برای کد کردن در اختیار دارد.

این تابع می‌تواند تا 4 محور را برای interpolation خطی همزمان، پشتیبانی کند.

یا interpolation خطی دو مجموعه ی دو محوری.

(Gp0=محورهای Ps0 و Ps1 و Gp1=محورهای Ps2 و Ps3)

خروجی ها برای این تابع باید تنظیم بشوند تا در یکی از مدهای U/D یا A/B خروجی مناسب بدهند در غیر

این صورت مانند خروجی های عادی رفتار خواهند کرد.

مد U/D (UP/DOWN): Y0, Y2, Y4, Y6 و پالس بالارونده می‌فرستد.

Y1 (Y3, Y5, Y7) و پالس پایین رونده می‌فرستد

مد A/B: Y0 (Y2, Y4, Y6) و پالس فاز A می‌فرستد

Y1 (Y3, Y5, Y7) و پالس فاز B می‌فرستد.

ارتباطات برای کنترل موقعیت

M1991	ON: پالس را کند کرده سپس قطع می‌کند. OFF: فوراً پالس را قطع می‌کند.
M1992	ON: Ps0 آماده است OFF: Ps0 فعال است.
M1993	ON: Ps1 آماده است. OFF: Ps1 فعال است.
M1994	ON: Ps2 آماده است. OFF: Ps2 فعال است.
M1995	ON: Ps3 آماده است. OFF: Ps3 فعال است.
M1934	ON: آخرین step را تمام کرد.
M1935	ON: آخرین step را تمام کرد.

ON : M2000 ، چند محور همزمان عمل می کنند.

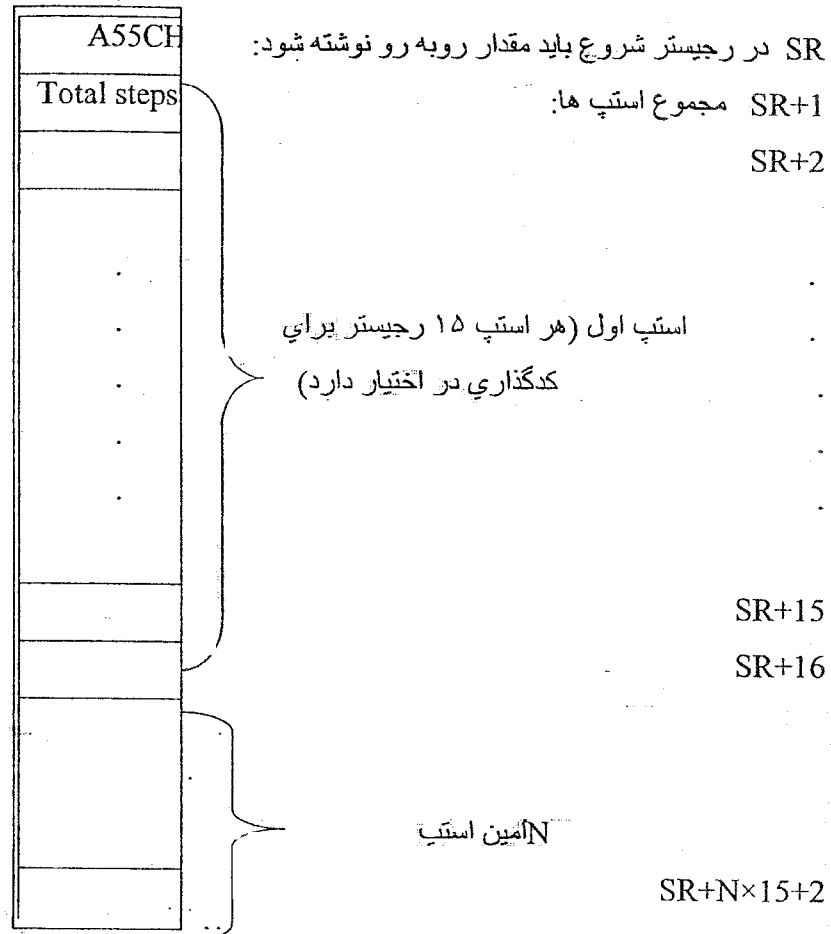
سرعت بردار Gp0	DR4068
سرعت بردار Gp1	DR4070
Gp0 error code	D4060
Gp1 error code	D4061
شماره استپی که به Gp0 مربوط است (شماره استپی که کامل شده است)	D4062
شماره استپی که به Gp1 مربوط است (شماره استپی که کامل شده است)	D4063

Ps No	فرکانس خروجی جاری	موقعیت پالس جاری	تعداد پالس باقی مانده برای انتقال
Ps0	DR4080	DR4088	DR4072
Ps1	DR4082	DR4090	DR4074
Ps2	DR4084	DR4092	DR4076
Ps3	DR4086	DR4094	DR4078

در تابع ۱۴۷، نمی توان فرکانس خروجی را در حین انتقال پالس تغییر داد.

در SR، رجیستر شروع رجیسترهایی ذخیره می شود که برنامه مکان یابی در آنها ذخیره

می شود:



WR نقطه شروع رجیسترهای عملگر (سیستمی) این تابع می باشد.

استپ اجرا شده یا متوقف شده	WR+0
Working flag	WR+1
توسط سیستم کنترل می شود	WR+2
توسط سیستم کنترل می شود	WR+3
توسط سیستم کنترل می شود	WR+4
توسط سیستم کنترل می شود	WR+5

توسط سیستم کنترل می شود	WR+6
توسط سیستم کنترل می شود	WR+7
توسط سیستم کنترل می شود	WR+8

WR+0 : هنگام اجرای این تابع ، محتویات این رجیستر ، استپ را که اجرا می شود نمایش می دهد (1~N). اگر

تابع در حال اجرا نباشد ، محتویات این رجیستر ، استپی را که در آن جا متوقف شده است نشان می دهد. وقتی

(EN=1) ، استپ بعدی اجرا می شود. (اگر استپ جاری ، آخرین استپ باشد، اجرا از اولین استپ شروع می شود)

WR+1 : B0~B7 (بیت ۰ ~ بیت ۷) مجموع استپ ها

B8=ON ، خروجی نگه داشته شده است. (pause)

B9=ON ، منتظر شرایط انتقال

B10=ON ، خروجی بی انتها

B12=ON ، در حال انتقال پالس (بیت مخصوص خروجی ACI)

B13=ON ، error در اجرای تابع (بیت مخصوص خروجی ERR)

B14=ON ، پایان اجرای یک استپ (بیت مخصوص خروجی DN)

وقتی step کامل می شود 'DN' روشن می شود و این وضعیت باقی می ماند. کاربر می تواند از لبه بالا رونده

DN برای پاک کردن WR+1 استفاده کند.

Error	کد	مشخصه Error	
R4060(PS0)	0	Error فاقد	
R4061(PS1)	1	ارور پارامتر ۰	<p>کدهای اروری که ممکن است در هنگام اجرای تابع ۱۴۱ به وجود بیایند</p>
R4062(PS2)	2	ارور پارامتر ۱	
R4063(PS3)	3	ارور پارامتر ۲	
R4060(Gp0)	4	ارور پارامتر ۳	
R4061(GP1)	5	ارور پارامتر ۴	
	6	ارور پارامتر ۵	
	7	ارور پارامتر ۶	
	8	ارور پارامتر ۷	
	9	ارور پارامتر ۸	
	10	ارور پارامتر ۹	
	13	ارور پارامتر ۱۲	
	14	ارور پارامتر ۱۳	
	15	ارور پارامتر ۱۴	
	30	Error of variable address for speed setting	<p>140 147</p>
	31	Error of setting value for speed setting	
	32	Error of variable address for stroke setting	
	33	Error of setting value for stroke setting	
	34	Illegal positioning program	
	35	length error of total step	
	36	Over the maximum step	
	37	Limited frequency error	
	38	initiate/stop frequency error	
	39	Over range of compensation value for movement	
	40	Over range of moving stroke	
	41	ABS positioning is not allowed within DRVC commands	
	42	DRVZ cant follow DRVC	
	50	Illegal operation mod of DRVZ	
	51	Illegal DOG input number	
	52	Illegal PG0 input number	
	53	Illegal CLR output number	

60 : Illegal linear interpolation command

توجه : محتوای رجیستر مشخصه ارور، آخرین کد ارور را حفظ می کند. برای اطمینان از این که ارور بیشتری روی نمی دهد، می توانید این رجیستر را ۰ کنید.

تغییر جدول برنامه Servo توسط WinProladder

برای استفاده از تابع ۱۴۷ باید ابتدا تنظیمات زیر صورت گیرد.

برای انجام تنظیمات به ترتیب زیر کلیک کنید تا جدول تنظیمات بیاید:

Project name / Table Edit / Servo Program Table

به روی Servo Program Table راست کلیک کنید و New Table را انتخاب کنید.

Table Edit

Table Properties

Table Type: Multi-Axis positioning table

Table Name: LINE

Table starting address: R6000

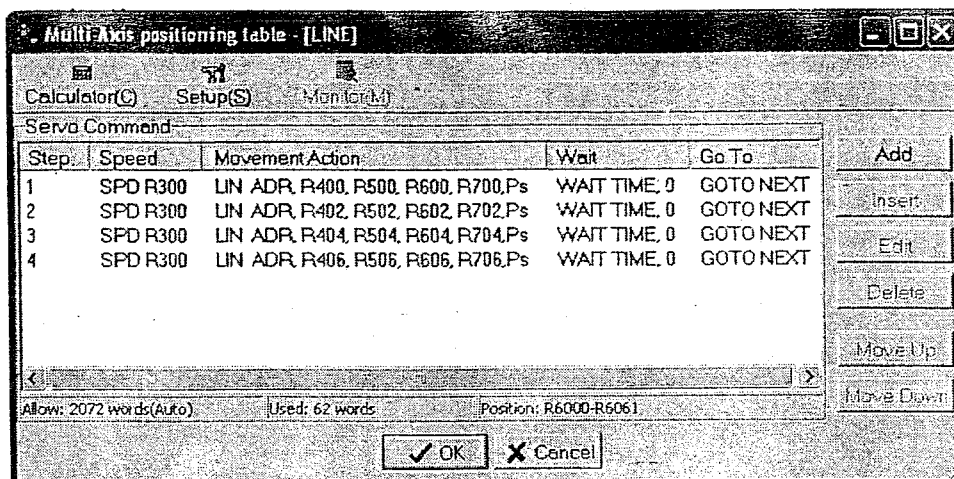
Table Capacity: ☒ Dynamic Allocation
☐ Fixed Length

☐ Load Table From PLC

☐ Load Table From ROP

Description

OK Cancel



دستورات مکان یابی برای interpolation خطی به صورت زیر هستند:

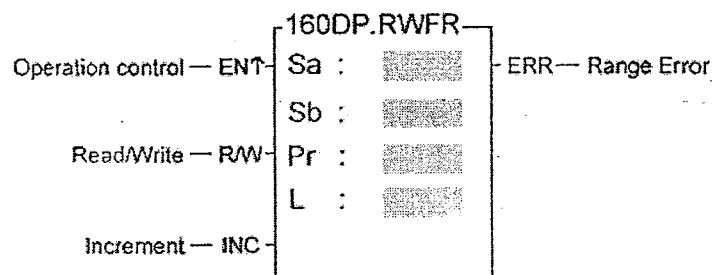
توضیحات	رجیستر عملوند	دستور
<p>سرعت بردار را برای interpolation خطی تنظیم می کند. سرعت حرکت بر حسب فرکانس یا m/s بیان می شود.</p> <p>(پارامتر -1 یا ۲ بر حسب فرکانس سیستم به صورت پیش فرض در حالت فرکانس است)</p> <p>عملوند می تواند عدد ثابت یا یک رجیستر باشد که اگر رجیستری باشد به ۲ رجیستر، فضا احتیاج دارد.</p> <p>فرکانس محور مربوطه برای خروجی از روی مشخصات سرعت بردار محاسبه خواهد شد.</p> <p>$921600 \text{ Hz} \geq \text{رنج فرکانس خروجی} \geq 1$</p>	<p>XXXXXX یا</p> <p>Rxxxx یا</p> <p>Dxxxx</p>	SPD
<p>جهت و اندازه حرکت را مشخص می کند بر حسب پالس یا mm، درجه، inch (وقتی پارامتر -1=0 از تابع ۱۴۱ تنظیمات بر حسب پالس خواهد بود، پیش فرض سیستم پالس است)</p> <p>وقتی ششمین عملوند LIN، Ut باشد، بر اساس مشخصات پارامترهای ۱، ۲، ۳ از تابع ۱۴۱، سیستم، شمارش پالس مربوطه را به خروجی مورد نظر تبدیل می کند.</p>	<p>ADR/ABS, X, Y, Z, W, Ut/ Ps</p> <p>X: مختصات محور Ps0</p> <p>Y: مختصات محور Ps1</p> <p>Z: مختصات محور Ps2</p> <p>W: مختصات محور Ps3</p>	LIN

<p>عملوند اول: ADR یا ABS</p> <p>ADR : حرکت نسبی</p> <p>ABS : حرکت مطلق</p> <p>عملوندهای دوم تا پنجم: مختصات محوره‌های X,Y,Z,W را مشخص می کند.</p> <p>وقتی مختصات یک محور ۰ است یا جای خالی (در دستورات متنی) گذاشته می شود و عملوند اول نیز ADR است، یعنی حرکتی در راستای آن محور نخواهد بود.</p> <p>وقتی به جای مختصات یک محور جای خالی (در دستورات متنی) گذاشته می شود و عملوند اول نیز ABS است، یعنی حرکتی در راستای آن محور نخواهد بود</p> <p>حداکثر اندازه حرکت باید کمتر از ± 1999999 پالس باشد.</p> <p>عملوند ششم: این عملوند واحد حرکت را مشخص می کند.</p> <p>Ut : هر عدد معادل یک واحد خواهد بود (توسط پارامتر ۰.۳ دستور ۱۴۱ مشخص می شود)</p> <p>Ps: رزولوشن اجرا شونده، معادل یک پالس خواهد بود.</p>		
<p>LINE برای interpolation خطی در حرکت بی انتها استفاده می شود.</p> <p>در این دستور، رابطه بین محورها در خروجی، طوری است که بقیه محورها از محوری که بیشترین مختصات را دارد تبعیت خواهند کرد.</p> <p>به عنوان مثال: در حالتی که عملوند ششم Ps است و مختصات محورها</p> <p>$Ps0=1000, Ps1=500, Ps2=300, Ps3=0$</p> <p>باشد، بدین معنی است که اگر محور Ps0، ۱۰۰۰ پالس می فرستد، سپس Ps1 و Ps2 به ترتیب ۵۰۰ و ۳۰۰ پالس خواهند فرستاد. (Ps3 کار نمی</p>	<p>ADR/ABS,X,Y,Z,W,Ut/Ps</p> <p>X: مختصات محور Ps0</p> <p>Y: مختصات محور Ps1</p> <p>Z: مختصات محور Ps2</p> <p>W: مختصات محور Ps3</p>	<p>LINE</p>

<p>کند چون مقدارش ۰ است)</p> <p>این دستور این نسبت ها را در دادن پالس به خروجی ادامه می دهد تا زمانی که تابع ۱۴۷ متوقف شود.</p>			
<p>پس از اتمام خروجی پالس، این دستور باعث انتظار سیستم برای زمان معین می شود سپس به مرحله ی معین شده می رود.</p> <p>این دستور ۵ مدل عملوند دارد:</p> <p>Time: (زمان پایه ۰.۰۱ ثانیه) وقتی زمان تعیین شده به پایان رسید، مرحله ای را اجرا می کند که توسط GOTO مشخص شده است.</p>		<p>Time,XXXXX</p> <p>یا Rxxxxx</p> <p>یا Dxxxxx</p> <p>یا X0~X255</p> <p>یا Y0~Y255</p> <p>یا M0~M1911</p> <p>S0~S999</p>	WAIT
<p>تا زمانی صبر می کند که مشخصه مورد نظر ON شود، سپس مرحله ای را اجرا می کند که توسط GOTO مشخص شده است</p>	<p>X0~X255</p> <p>Y0~Y255</p> <p>M0~M1911</p> <p>M0~M1911</p>		
<p>اگر مشخصه مورد نظر این دستور در حین پالس دهی به خروجی، ON شود، فوراً مرحله ای را که توسط GOTO مشخص شده انجام می دهد. اگر تا آخر پالس دهی ON نشود، این دستور مانند WAIT عمل می کند</p>		<p>یا X0~X255</p> <p>یا Y0~Y255</p> <p>یا M0~M1911</p> <p>S0~S999</p>	EXT
<p>پس از پایان یافتن دستورات ACT, WAIT و EXT مرحله ای را اجرا می کند که GOTO به آن اشاره می کند.</p> <p>NEXT: بدین معنی است که مرحله ی بعد اجرا شود.</p> <p>1~N: مرحله ای را انجام می دهد که LABLE آن عدد مورد نظر باشد.</p>		<p>یا NEXT</p> <p>یا 1~N</p> <p>یا Rxxxxx</p> <p>Dxxxxx</p>	GOTO
<p>مرحله ای که اجرا می شود در این رجیستر ها ذخیره شده است.</p>	<p>Rxxxxx</p> <p>Dxxxxx</p>		

پایان برنامه مکان یابی	MEND
------------------------	------

160.READ/WRITE FILE REGISTER



برای برنامه Ladder، این تابع تنها تابعی است که می تواند به پردازش رجیسترهای فایل، بپردازد.

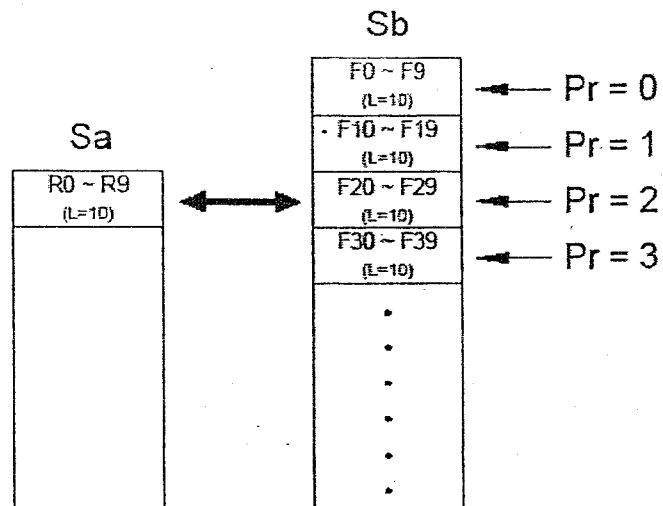
هرگاه $ENT = 0$ به 1 تغییر کند:

اگر $P/W = 1$ محتویات رجیسترهای فایل با آدرس پایه Sb به طول L از جایی که Pr اشاره می کند به داخل رجیسترهای شروع شونده از Sa ریخته می شود.

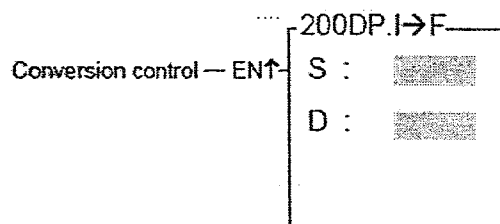
اگر $R/W = 0$ محتویات رجیسترهای شروع شونده از Sa به داخل رجیسترهای فایل با آدرس پایه Sb به طول L از جایی که Pr اشاره می کند، ریخته می شود.

اگر $INC = 1$ ، بعد از اجرا، Pr یکی اضافه خواهد شد

اگر $L = 0$ یا $L > 511$ باشد یا عملیات بخواند خارج از رنج رجیسترهای فایل ($F0 \sim F8191$) اجرا شود، ERR فعال خواهد شد.



200.CONVERSION of INT to FLOAT

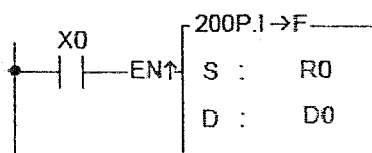


هرگاه 'EN' از ۰ به ۱ تغییر کند:

حتویات رجیستر ۱۶ بیتی (INTEGER) شروع شوند از S درداخل رجیستر ۳۲ بیتی (FLOAT) شروع

نمونه از D ذخیره می شود.

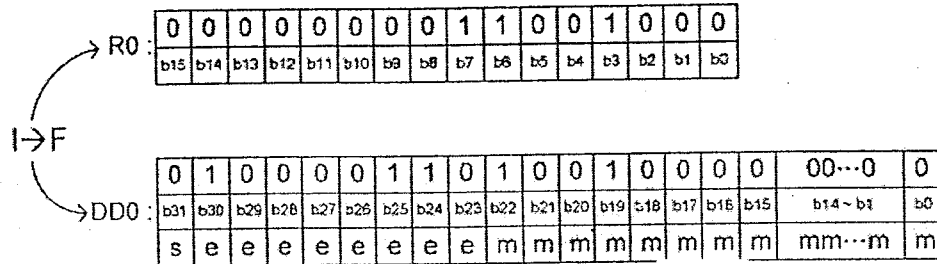
مثال:



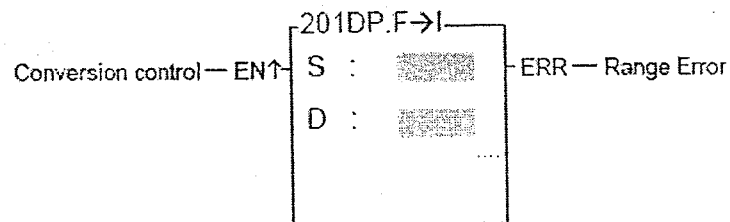
※ R0 = 200 (0000000011001000)

Integer To Floating ←

→ DD0 = 43480000H



201. FLOAT to INTEGER



هرگاه 'EN' از ۰ به ۱ تغییر کند:

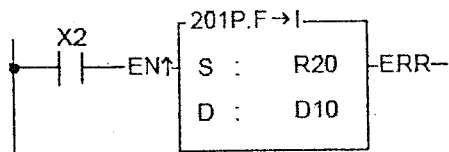
محتویات رجیستر ۳۲ بیتی (FLOAT) شروع شوند از S در داخل رجیستر ۱۶ بیتی (INTEGER) شروع

شونده از D ذخیره می شود.

اگر مقدار مبدا فراتر از رنج مقصد باشد و قابل تبدیل شدن نباشد ، 'ERR' فعال می شود و مقدار D دست

نخورده باقی می ماند.

مثال:



※ DR20 = 123.45 → Normalize → 42F6E666H

Floating To Integer

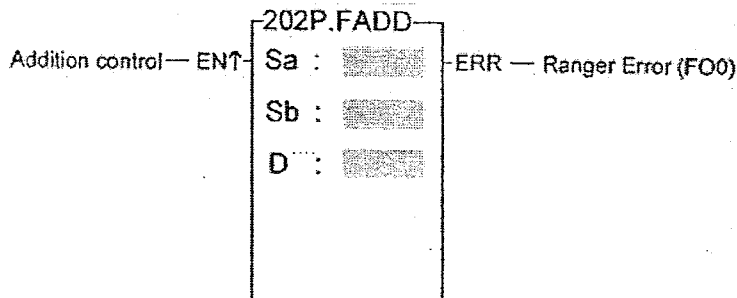
→ D10 = 007BH

[illegible]
$$\begin{array}{c} \downarrow \\ F \rightarrow I \\ \downarrow \end{array}$$

D10:

0	0	0	0	0	0	0	0	0	1	1	1	1	0	1	1
b13	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0

202. FLOATING POINT NUMBER ADDITION

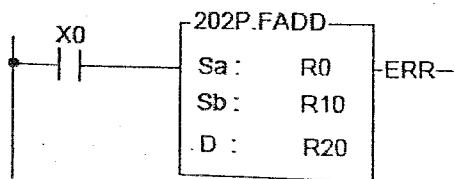


هرگاه 'EN' از ۰ به ۱ تغییر کند:

مقدار FLOAT (۳۲ بیتی) S_a با مقدار FLOAT S_b جمع شده و نتیجه در D ریخته می شود.

اگر مجموع دو عدد فراتر از رنج یک رهیستر ۳۲ بیتی باشد ($\pm 3.4 \times 10^{38}$)، 'ERR' فعال می‌شود.

مثال:



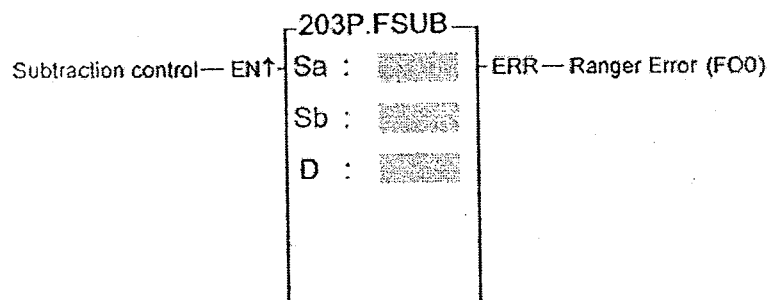
DR0 200 ⇒ Floating Point Number : DR0 43480000H

DR10 150 ⇒ Floating Point Number : DR10 43160000H

+

DR20 43AF0000H

203. FLOATING POINT NUMBER SUBTRACTION

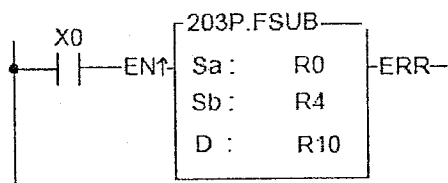


هرگاه 'EN' از ۰ به ۱ تغییر کند:

مقدار FLOAT (۳۲ بیتی) Sa منتهای مقدار FLOAT، Sb شده و نتیجه در D ریخته می شود.

اگر نتیجه تفریق دو عدد فراتر از رنج یک رجیستر ۳۲ بیتی باشد ($\pm 3.4 \times 10^{38}$)، 'ERR' فعال می شود.

مثال:

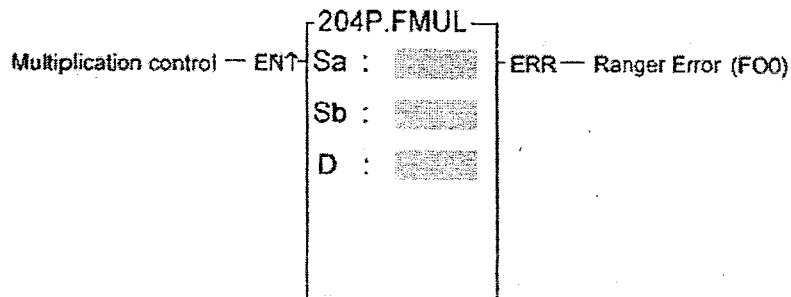


DR0 200 ⇒ Floating Point Number : DR0 43480000H

DR4 500 ⇒ Floating Point Number : DR4 43FA0000H

DR10 C3960000H

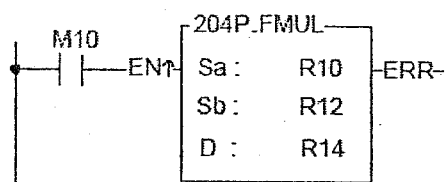
204. FLOATING POINT NUMBER MULTIPLICATION



هرگاه 'EN' از ۰ به ۱ تغییر کند:

مقدار FLOAT (۳۲ بیتی) Sa ضرب در مقدار FLOAT، Sb شده و نتیجه در D ریخته می شود.
اگر نتیجه ضرب دو عدد فراتر از رنج یک رجیستر ۳۲ بیتی باشد ($\pm 3.4 \times 10^{38}$) ERR فعال می شود.

مثال:



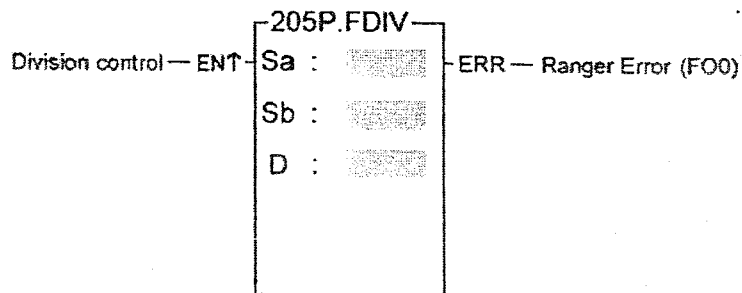
DR10 | 1 2 3 . 4 5 | ⇒ Floating Point Number : DR10 | 4 2 F 6 E 6 6 6 H

DR12 | 6 7 8 . 5 4 | ⇒ Floating Point Number : DR12 | 4 4 2 9 A 2 8 F H

×

DR14 | 4 7 A 3 9 A E 2 H

205. FLOATING POINT NUMBER DIVISION

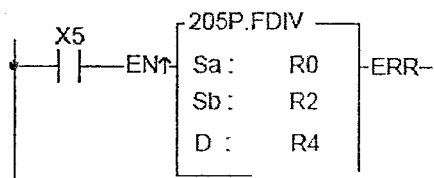


هرگاه 'EN' از ۰ به ۱ تغییر کند:

مقدار FLOAT (۳۲ بیتی) Sa را بر مقدار FLOAT، Sb کرده و نتیجه در D ریخته می شود.

اگر نتیجه تقسیم دو عدد فراتر از رنج یک رجیستر ۳۲ بیتی باشد ($\pm 3.4 \times 10^{38}$)، 'ERR' فعال می شود.

مثال:



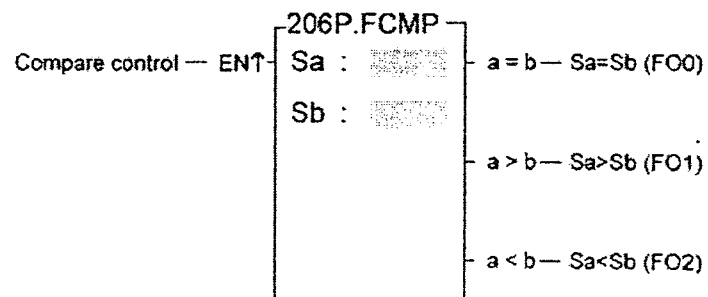
DR0 | 1 2 5 . 2 5 ⇒ Floating Point Number : DR0 | 4 2 F A 8 0 0 0 H

DR2 | 5 ⇒ Floating Point Number : DR2 | 4 0 A 0 0 0 0 0 H

÷

DR4 | 4 1 C 8 6 6 6 6 H

206. FLOATING POINT NUMBER COMPARE



هرگاه 'EN' از ۰ به ۱ تغییر کند:

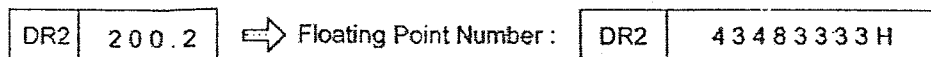
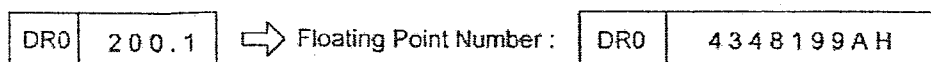
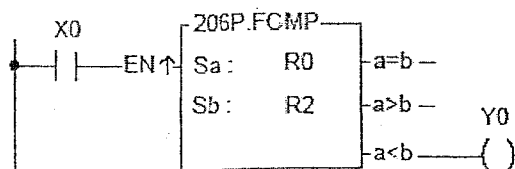
مقدار دو رجیستر ۳۲ بیتی Sa و Sb را با هم مقایسه کرده.

اگر $Sa > Sb$: $a > b = 1$ می شود.

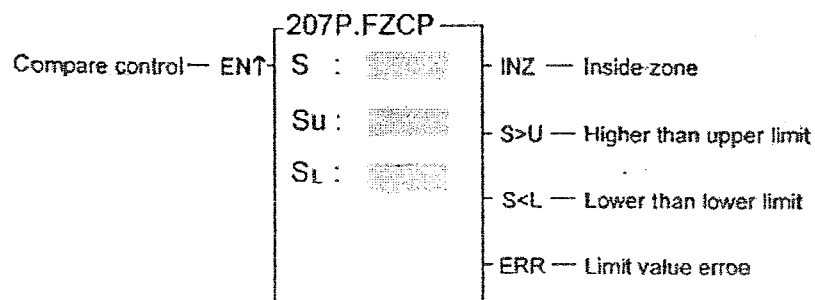
اگر $Sa < Sb$: $a < b = 1$ می شود.

اگر $Sa = Sb$: $a = b = 1$ می شود.

مثال:



207 FLOATING POINT NUMBER ZONE COMPARE



هرگاه 'EN' از ۱ به ۰ تغییر کند:

مقدار ۳۲ بیتی S را با Su و SL مقایسه کرده.

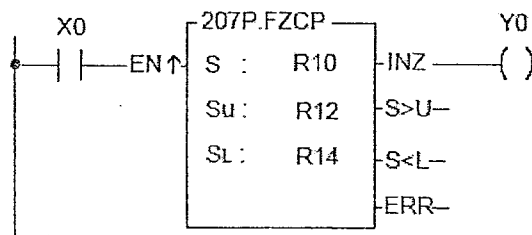
اگر $S > U = 1$: $S > S_u$ می شود.

اگر $S < L = 1$: $S < S_L$ می شود.

اگر $INZ = 1$: $S_L < S < S_u$ می شود.

اگر $S_L > S_u$: 'ERR' داده خواهد شد.

مثال:

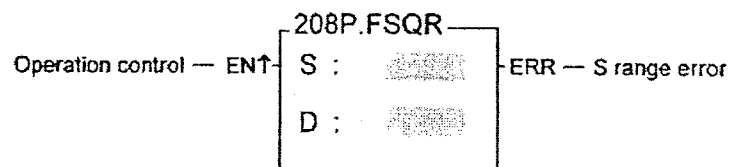


S	DR10	2 0 0 0 . 2	⇒ Floating Point Number :	DR10	4 4 F A 0 6 6 6 H	
Su	DR12	3 0 0 0 . 3	⇒ Floating Point Number :	DR12	4 5 3 B 8 4 C D H	{ Upper limit value }
Sl	DR14	1 0 0 0 . 1	⇒ Floating Point Number :	DR14	4 4 7 A 0 6 6 6 H	{ Lower limit value }

Before-execution

$X0 = \text{true} \rightarrow \text{FLOATING ZONE COMPARE} \rightarrow Y0 = \boxed{1}$
Results of execution

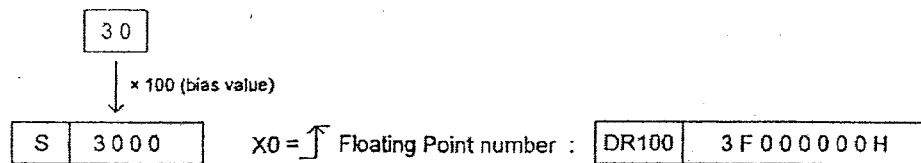
208. FLOATING POINT NUMBER SQUARE ROOT



هرگاه 'EN' از ۰ به ۱ تغییر کند:

جذر عدد ۳۲ بیتی S را گرفته و نتیجه را در D می ریزد

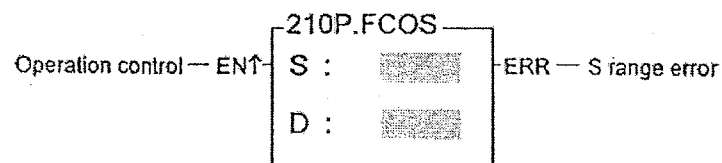
اگر مقدار S، منفی باشد، 'ERR' فعال می شود.



$$\sin(30) = 0.5$$

210.COS INSTRUCTION

تابع مثلثاتی کسینوس

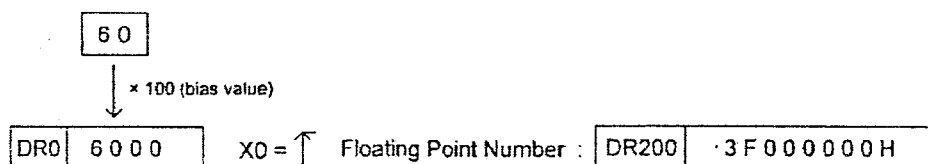
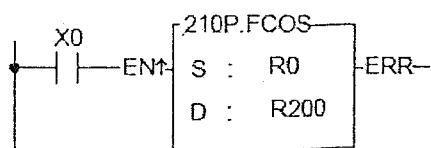


هرگاه 'EN' از ۰ به ۱ تغییر کند:

از مقدار موجود در رجیستر S، کسینوس گرفته و نتیجه را به صورت ۴۲ بیتی در D و D+1 می ریزد.
رنج S در واحد ۰.۱ درجه، -18000~+18000 می باشد که اگر فراتر از این رنج داده شود، 'ERR' فعال می

شود.

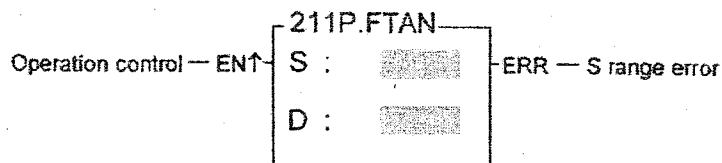
مثال:



$$\cos(60) = 0.5$$

211.TAN INSTRUCTION

تابع مثلثاتی تانژانت

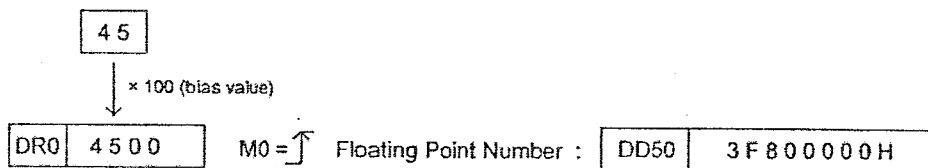
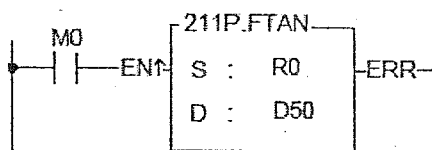


هرگاه 'EN' از ۰ به ۱ تغییر کند:

از مقدار موجود در رجیستر S، تانژانت گرفته و نتیجه را به صورت ۳۲ بیتی در D و D+1 می ریزد.
رنج S در واحد ۰.۱ درجه، ۱۸۰۰۰~+۱۸۰۰۰ می باشد که اگر فراتر از این رنج داده شود، 'ERR' فعال می

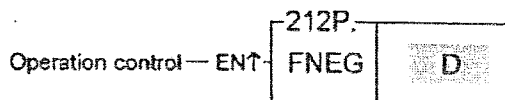
شود.

مثال:



$$\text{TAN}(45) = 1$$

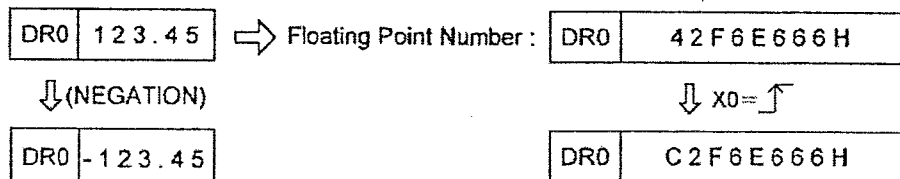
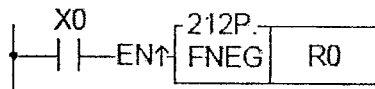
212.CHANGE SIGN OF FLOAT



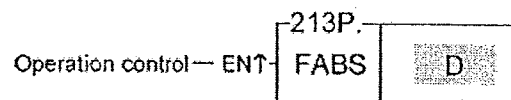
هرگاه 'EN' از ۰ به ۱ تغییر کند:

علامت عدد FLOAT (۳۲ بیتی) D را، عکس می کند و نتیجه را در خود D ذخیره می کند.

مثال:



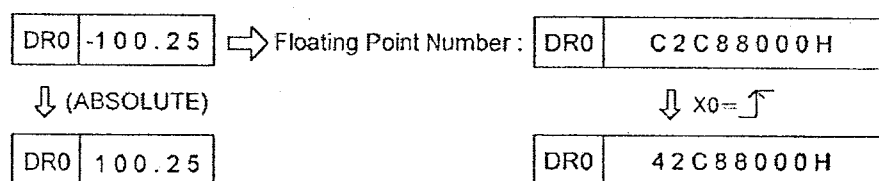
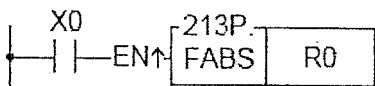
213.FLOAT ABSOLUTE



هرگاه 'EN' از ۰ به ۱ تغییر کند:

قدر مطلق عدد D.FLOAT را گرفته و نتیجه را در خود D ذخیره می کند.

مثال:



فصل هفتم

وسایل ورودی و خروجی

کلید شستی و کنتاکتور از جمله وسایل ورودی و خروجی Digital می باشند. زیرا در هر لحظه می توانند تنها وصل یا قطع باشند. بنابراین مستقیماً به ورودی و خروجی دیجیتال متصل می شوند. ترموکوپل یک وسیله ی ورودی آنالوگ می باشد زیرا متناسب با دمای محیطی که در آن قرار گرفته، در دو سر آن ولتاژ ایجاد می شود. این ولتاژ سپس توسط مدارات الکترونیکی به ولتاژ و یا جریان مناسبی تبدیل شده و به ماژول ورودی دمای PLC وصل می گردد.

در این فصل با بعضی از انواع وسایل ورودی و خروجی، دیجیتال و آنالوگ که در صنعت، به PLC متصل می گردند آشنا خواهید شد.

۷-۱) انواع وسایل ورودی

سنسورها وسایلی هستند که کمیت های فیزیکی نظیر دما، فشار، جریان سیال، سطح مایع در مخزن، وزن، حرکت مکانیکی، سرعت، شتاب، رطوبت، میدان مغناطیسی و ... را حس می نمایند و نسبت به آن عکس العمل نشان می دهند. که این عکس العمل می تواند به صورت دیجیتال (باز و بسته شدن یک کنتاکت) و یا آنالوگ (ولتاژ پیوسته) آشکار گردد.

وسایل ورودی دیجیتال (سوئیچ ها) عموماً دارای یک سنسور و یک کنتاکت می باشند. به عنوان مثال یک سوئیچ فشار (Pressure Switch) دارای یک قسمت حس کننده فشار (سنسور فشار) می باشد. هنگامی که فشار به حد معینی برسد عکس العمل سنسور موجب وصل شدن یک کنتاکت در داخل سوئیچ

می گردد. بنابراین یک سوئیچ فشار می تواند به صورت مستقیم به ورودی دیجیتال PLC متصل شود.

سوئیچ ها می توانند در حالت عادی باز یا بسته باشند. به عنوان مثال سوئیچ فشار ذکر شده، در حالت عادی کنتاکت آن باز می باشد و با رسیدن فشار به نقطه ی معینی کنتاکت آن بسته می گردد. از این رو سوئیچ فوق را NO (Normally open) می نامند. اما اگر در حالت عادی کنتاکت این سوئیچ بسته باشد و بعد از اعمال فشار این کنتاکت باز گردد آن را NC (Normally Close) می گویند. علاوه بر این باید

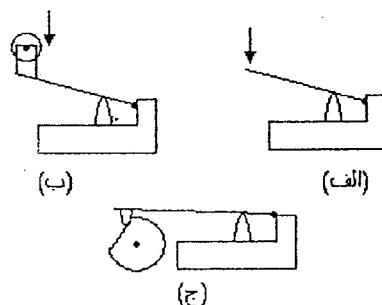
توجه داشت که بعضی از انواع سوئیچ ها، به جای استفاده از کنتاکت های مکانیکی از کلیدهای الکترونیکی نظیر ترانزیستور و یا تریاک استفاده می کنند.

در طبیعت کمیت های فیزیکی همگی پیوسته می باشند. بنابراین برای اندازه گیری آن ها از سنسورها به همراه مدارات الکترونیکی مورد نیاز استفاده می گردد (کل این مجموعه را ترانسمیتر می نامند). متناسب با کمیت فیزیکی اندازه گیری شده جریان و یا ولتاژی در خروجی ترانسمیتر ایجاد می شود و در داخل کارت آفالوک PLC توسط یک A/D، این ولتاژ به یک عدد دیجیتال قابل فهم برای CPU تبدیل می گردد.

۷-۱-۱) سنسورهای تشخیص اشیاء (Object Detector Sensors)

در خطوط تولید جهت تشخیص عبور یک شی، وجود یک قطعه ورود انسانی به محدوده ی کار یک روبات، محدود کردن کورس یک پیستون و ... با هدف کنترل سیستم و همچنین جلوگیری از بروز صدمات به تجهیزات، کاربر و مواد اولیه از این گونه سنسورها استفاده می گردد و انواع آن به شرح ذیل می باشد:

الف) **لیمیت سوئیچ (Limit Switch):** در اثر تماس مستقیم و مکانیکی شی با لیمیت سوئیچ کنتاکت های آن تغییر وضعیت می دهند. دستگاه مته برقی در فصل یک را به خاطر آورید، در آن جا جهت اطمینان از قرار گرفتن قطعه کار در مکان مناسب، رسیدن مته به سطح قطعه و سوراخ کردن قطعه تا عمق لازم از سه لیمیت سوئیچ مجزا استفاده شده بود. شکل ۷-۱ سه نوع مختلف لیمیت سوئیچ را نشان می دهد.



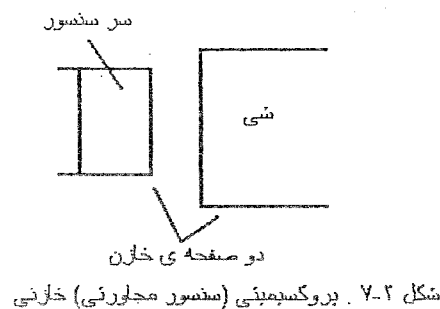
شکل ۷-۱. محرک لیمیت سوئیچ به وسیله ی: الف) اهرم
ب) غلطک، ج) بلامک

ب) پروکسیمیتی سوئیچ (Proximity Switch): در این نوع سوئیچ بدون برقرار شدن تماس مکانیکی، عبور یا وجود شی تشخیص داده می شود. بنابراین به دلیل نداشتن تماس مستقیم دارای استهلاک کمی بوده و در انواع ذیل می باشد:

ب-۱- پروکسیمیتی سوئیچ القایی: این سوئیچ از یک اسیلاتور RF (Radio Frequency) و یک مدار LC تشکیل شده است. طبیعتاً به دلیل نوسانات ناشی از اسیلاتور میدان مغناطیسی اطراف مدار LC القا می گردد که با نزدیک شدن یک قطعه ی فلزی به این میدان، جریانی در آن قطعه القا شده و میدان ناشی از این جریان (نیروی ضد محرکه) عملاً باعث برهم خوردن تعادل اسیلاتور شده و آن را متوقف می نماید که حاصل آن وصل شدن یک کلید الکترونیکی مانند ترانزیستور می باشد. به همین دلیل به این نوع سوئیچ ECKO (Eddy Current Killed Oscillator) نیز می گویند.

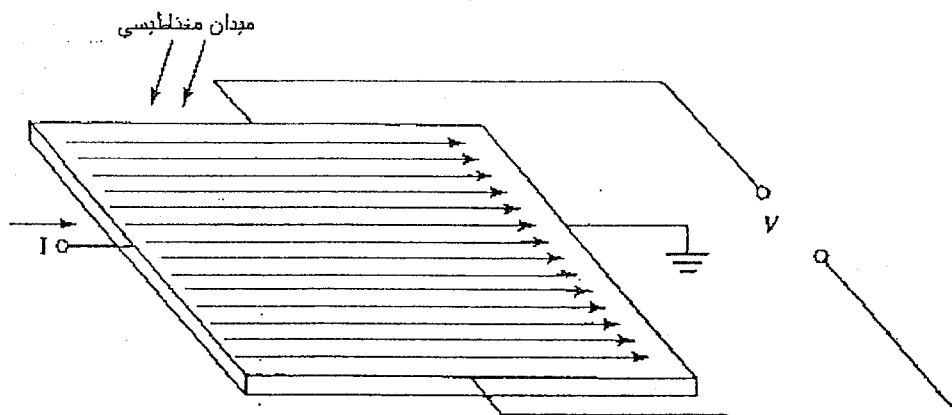
دامنه کاربرد این سنسور بین 0.1 mm تا 10 mm می باشد و حساسیت آن به فلزات مغناطیسی نظیر آهن بیشتر از فلزات دیگر است.

ب-۲- سوئیچ پروکسیمیتی خازنی: این نوع سوئیچ قادر به تشخیص اشیاء فلزی و غیر فلزی می باشد. همانطور که می دانید ظرفیت یک خازن با تغییر عایق بین صفحات آن تغییر می نماید. در نوع فلزی سنسور نقش یکی از صفحات خازن را ایفا نموده و قطعه ی فلزی به عنوان صفحه ی دیگر خازن به کار می رود. با نزدیک شدن قطعه ی فلزی به سنسور فاصله ی هوایی کم شده و ظرفیت خازن تغییر می کند. (شکل ۲-۷)



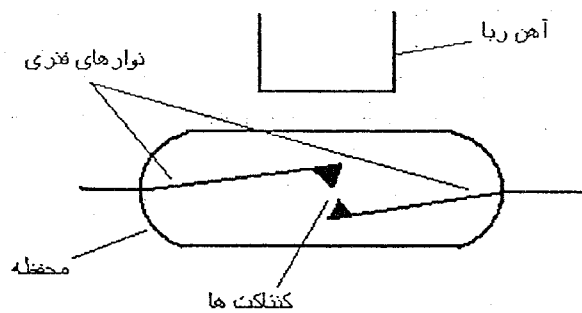
در نوع غیرفلزی صفحه ی دیگر خازن، زمین می باشد و شی غیرفلزی نیز نقش عایق موجود بین صفحات را به عهده دارد. دامنه ی کاربرد این سنسور بین 1 mm تا 6 mm می باشد.

ج) سنسور اثر هال (Hall Effect): هنگامی که جریان ثابتی از دو سر یک صفحه ی فلزی عبور داده شود، در صورتی که این صفحه در معرض میدان مغناطیسی عمود بر آن قرار بگیرد، در دو سر دیگر صفحه ولتاژی القاء می گردد که متناسب با شدت میدان مغناطیسی می باشد. بنابراین با استفاده از سنسور هال می توان متوجه حرکت یک شی مغناطیسی (نظیر آهنربا یا یک کویل حاوی جریان) گردید (شکل ۷-۳)



شکل ۷-۳. اثر هال

د) رید سوئیچ (Reed Switch): این سوئیچ از دو کنتاکت قابل انعطاف که در داخل یک محفظه شیشه ای قرار می گیرند تشکیل شده و همان طور که در شکل ۷-۴ مشاهده می نمایید با نزدیک شدن شیء مغناطیسی این دو کنتاکت جذب یکدیگر می شوند.



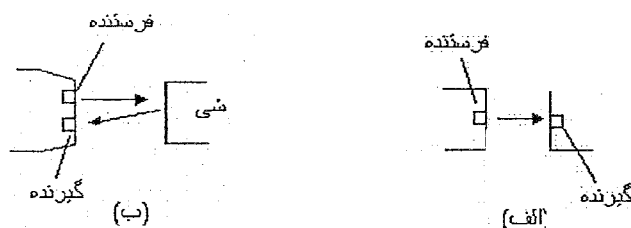
شکل ۷-۴. رید سوئیچ

ه) سنسور های نوری (Photoelectric Sensor): در دو نوع مستقیم و انعکاسی ساخته می شوند.

نوع مستقیم مطابق شکل الف-۵-۷ از یک فرستنده (LED مادون قرمز) و یک گیرنده (فتو ترانزیستور) تشکیل شده و هر کدام دارای یک عدسی جهت جلوگیری از پراکندگی نور بوده و در محفظه ای جداگانه قرار گرفته اند.

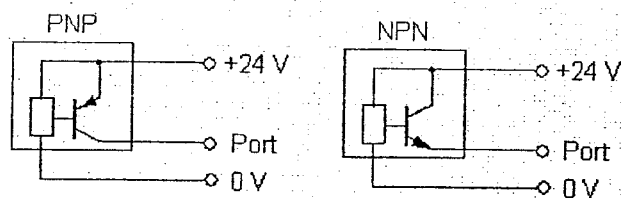
به عنوان مثال می توان در چهار گوشه محل کار یک روبات از این سنسور استفاده نمود. بدین صورت که هنگام وارد شدن انسان به محدوده ی کار این روبات مسیر نور قطع شده و فتوترانزیستور گیرنده غیرفعال می گردد و فرمان توقف را به سیستم روبات صادر می کند. در نوع انعکاسی گیرنده و فرستنده در داخل یک مجموعه قرار دارند و بنابراین تنها نیاز به یک مسیر سیم کشی می باشد. انعکاس نور نیز از طریق جسم مورد نظر یا با استفاده از صفحه منعکس کننده جداگانه ای تامین می گردد.

(شکل ب-۵-۷)



شکل ۷-۵ انواع سنسورهای نوری (الف) مستقیم، (ب) انعکاسی

نکته: شکل ۷-۶ طبقه ی خروجی ده نوع سوئیچ پروکسیستی DC را نمایش می دهد. اگر طبقه ی آخر این سوئیچ، ترانزیستور نوع NPN باشد ورودی PLC باید به صورت Source بسته شود. (یعنی ترمینال مشترک ورودی S/S به ولتاژ +۲۴ و ترمینال مربوط به ورودی مورد نظر به سر Port در روی ترانزیستور NPN متصل می گردد) اما اگر خروجی این سوئیچ، ترانزیستور نوع PNP باشد، ورودی PLC باید به صورت Sink بسته شود. (یعنی ترمینال مشترک ورودی S/S به ولتاژ ۰V و ترمینال مربوط به ورودی مورد نظر به سر Port در روی ترانزیستور PNP متصل گردد). بنابراین چون یک ورودی معمولاً به صورت Sink یا Source می باشد، در استفاده ی همزمان از چند سنسور در روی یک کارت ورودی باید دقت نمود که همگی از یک نوع باشند.



شکل ۶-۷. سوئیچ پروکسیمی

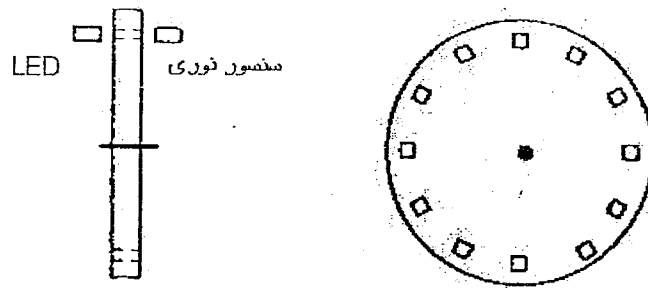
۲-۱-۷) سنسورهای جابجایی (Position Displacement Sensor)

این گونه سنسورها جهت اندازه گیری جابجایی خطی یا دورانی یک جسم بکار می روند. افزون بر این در بسیاری از سنسورهای دیگر نظیر سنسور فشار ابتدا کمیت مورد اندازه گیری تبدیل به حرکت مکانیکی می شود و سپس این حرکت اندازه گیری می گردد.

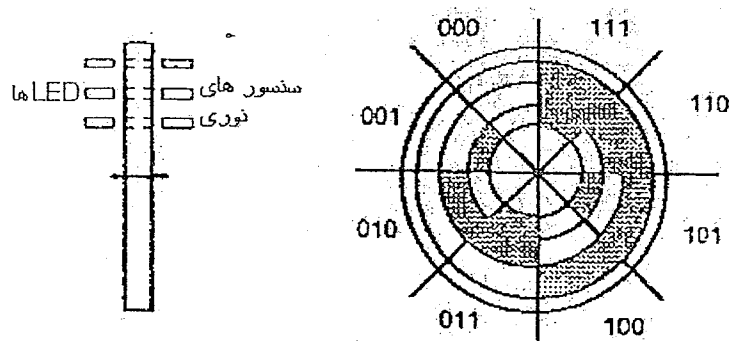
پتانسیومتر (مقاومت متغیر) یکی از ساده ترین وسایل اندازه گیری حرکت می باشد که در نوع خطی و دورانی ساخته می شود. در ادامه با چند نوع دیگر از این سنسورها آشنا خواهید شد.

الف) LVDT (Linear Variable Differential Transformer)

مطابق شکل ۷-۷ از یک سیم پیچ اولیه و دو سیم پیچ ثانویه که در روی یک استوانه تو خالی پیچیده می شوند تشکیل شده است. در درون این استوانه یک هسته آهنی قرار دارد که یک سر آن از بیرون به وسیله ای که می خواهیم جابجا کنیم آن را اندازه گیری نمائیم متصل است. سیم پیچ اولیه تحت ولتاژ متناوب با دامنه ی ثابت قرار می گیرد و دو سیم پیچ ثانویه به نحوی با یکدیگر سری می شوند که ولتاژ خروجی از تفاضل این دو ولتاژ القایی حاصل می گردد. زمانی که هسته در وسط استوانه قرار دارد، روی دو سیم پیچ ثانویه ولتاژ AC یکسانی القا شده و در این حالت ولتاژ خروجی صفر می باشد. هنگامی که هسته به سمت بیرون می آید $V_1 - V_2$ دارای مقدار منفی شده و هنگامی که هسته به سمت داخل رود مقدار $V_1 - V_2$ مثبت می گردد. سپس این ولتاژ متناوب خروجی را به ولتاژ مستقیم تبدیل می نمایند که اندازه این ولتاژ متناسب با موقعیت هسته در درون استوانه است.



شکل ۷-۸. اجزای تشکیل دهنده ی اینکودر افزایشی



شکل ۷-۹. اجزای تشکیل دهنده ی اینکودر مطلق سه بیتی

۳-۱-۷ کرنش سنچ (Strain Guage)

یک قطعه سیم فلزی را در نظر بگیرید. با اعمال کشش به این سیم (تنش) طول آن متناسب با نیروی اعمالی افزایش می یابد (کرنس). سنسورهای کرنش سنچ بر همین اساس ساخته می شوند. نسبت تغییرات مقاومت به تغییرات طول را ضریب کرنش سنچ می نامند که به صورت ذیل بیان می گردد (Guage Factor)

$$GF = \frac{\Delta R/R}{\Delta L/L}$$

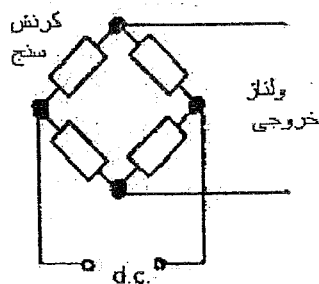
ضریب کرنش سنچ برای فلزات بین ۲ تا ۴ می باشد، در عمل برای این که تغییرات مقاومت سیم قابل توجه باشد از سیم بلند یا صفحه ی فلزی نازک مطابق شکل ۱۰-۷ برای ساخت کرنش سنچ ها استفاده می گردد. این قطعه مانند یک تمبر با استفاده از چسب اپوکسی به صورت یکنواخت روی یک سطح صاف پلاستیکی چسبانیده می شود و مجموعه ی فوق نهایتاً روی جسمی که می خواهیم تنش موجود روی آن اندازه گیری شود، نصب می گردد.



شکل ۷-۱۰. کرنش سنج

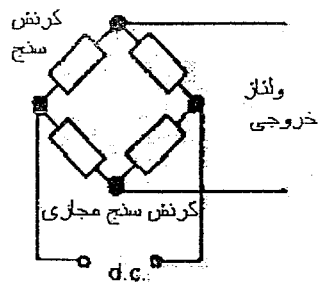
شکل ۷-۱۱ نحوه ی تبدیل کردن تغییرات مقاومت به تغییرات ولتاژ را به وسیله ی پل و تستون نمایش می دهد. هنگامی که پل در حال تعادل است یعنی چهار مقاومت موجود در روی آن با هم برابرند، ولتاژ خروجی وجود ندارد. با اعمال تنش مقاومت کرنش سنج تغییر می کند و ولتاژ خروجی با شرط $R \gg \Delta R$ برابر است

با $\Delta V = \frac{E \Delta R}{4R}$ که در این جا E ولتاژ تغذیه پل و تستون می باشد.



شکل ۷-۱۱. مدار پل و تستون

یکی از مشکلات موجود در این مدار، تغییر کرنش سنج در اثر تغییرات دمای محیط می باشد. برای جبران این اثر از دو کرنش سنج در دو طرف پل و تستون استفاده شده (شکل ۷-۱۲) که یکی از سنسورها به گونه ای نصب می شود تا تحت تنش قرار گرفته و سنسور دوم تنها تحت دمای محیط سنسور اولی باشد (بدون تنش). که بدین وسیله تاثیر دمای محیط از بین می رود.



شکل ۷-۱۲. جریان اثر دما به وسیله ی کرنش سنچ مجاری در پل وینستون

با نصب مناسب کرنش سنچ ها می توان از آن ها جهت اندازه گیری فشار، نیرو و وزن استفاده نمود. شکل الف-

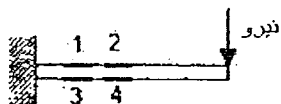
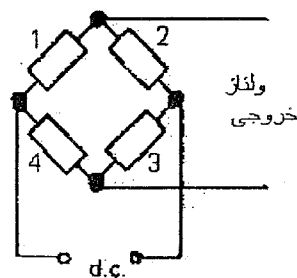
۷-۱۳ میله ی از یک طرف درگیر را نشان می دهد. هنگامی که این میله تحت تنش قرار گیرد خم شده، بنابراین

قسمت بالای آن تحت کشش قرار می گیرد (افزایش مقاومت کرنش سنچ های ۲ و ۱ و در قسمت تحتانی میله

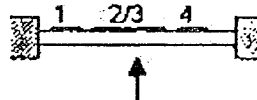
کاهش مقاومت کرنش سنچ های ۳ و ۴ را خواهیم داشت).

همچنین شکل ب- ۷-۱۳ نحوه ی نصب چهار کرنش سنچ روی یک میله از دو سمت درگیر یا دیافراگم را نشان

می دهد که هنگام نصب سنسورهای ۲ و ۱ بایستی 90° نسبت به سنسورهای ۳ و ۴ زاویه داشته باشند.



(الف)



(ب)

شکل ۷-۱۲. کاربرد کرنش سنچ: الف) سنسور نیرو، ب) سنسور فشار

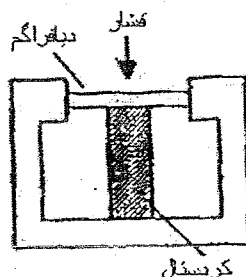
۴-۱-۷) اندازه گیری فشار سیال

نیروی وارده بر واحد سطح را فشار می نامند. این فشار می تواند نسبت به خلأ اندازه گیری شود، که آن را فشار مطلق می نامیم و یا می توان آن را نسبت به فشار اتمسفر اندازه گیری نمود که آن را فشار نسبی یا فشار گیج می نامند. همچنین سنسور اختلاف فشار ΔP نیز تفاضل دو فشار را اندازه گیری می کند. بنابراین ملاحظه می فرمایید که در هر سه حالت عملاً یک فشار نسبت به فشار دیگر اندازه گیری می شود.

سنسورهای فشار ابتدا فشار را به حرکت مکانیکی تبدیل می نمایند، سپس این جابه جایی می تواند به تغییرات ولتاژ منجر شود.

یکی از روش های تبدیل فشار به جابه جایی استفاده از دیافراگم است. دیافراگم یک صفحه ی قابل انعطاف از جنس فلز یا لاستیک می باشد که دو طرف یک محفظه را به طور کامل از هم جدا می نماید و در یک طرف محفظه، فشار اتمسفر و در طرف دیگر آن فشار مورد اندازه گیری اعمال می شود.

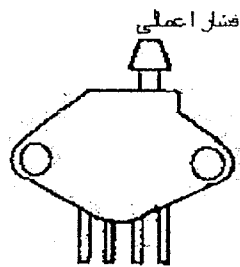
جابه جایی مرکز دیافراگم در اثر اختلاف این دو فشار می تواند به وسیله ی کرنش سنج (شکل ب-۱۳-۷)، LVDT، پتانسیومتر، تغییرات خازنی و یا کریستال پیزوالکتریک (شکل ۴-۷) تبدیل به تغییرات ولتاژ گردد. (در کریستال پیزوالکتریک هنگام اعمال نیرو متناسب با آن ولتاژ ضعیفی تولید می گردد).



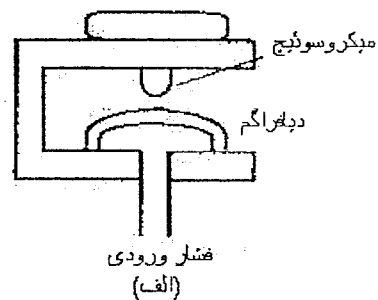
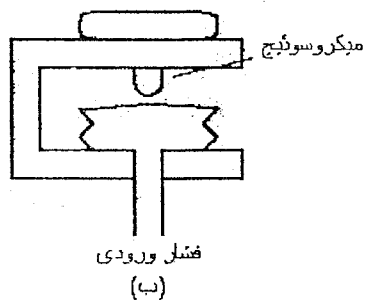
شکل ۱۴-۷. سنسور فشار پیزوالکتریک

شکل ۱۵-۷ یک سنسور فشار با نام $MP\ X100\ AP$ ساخت کارخانه ی موتورولا را نشان می دهد که براساس پدیده ی پیزوالکتریک ساخته شده است که توسط آن فشار مطلق را اندازه گیری می نمایند و خروجی آن $16\ mv/kpa$ است.

همچنین در شکل ۱۶-۷ دو نوع سوئیچ فشار (Pressure Switch) را مشاهده می نمائید که در اثر اعمال فشار معینی، یک میکروسوئیچ را فعال می کنند.

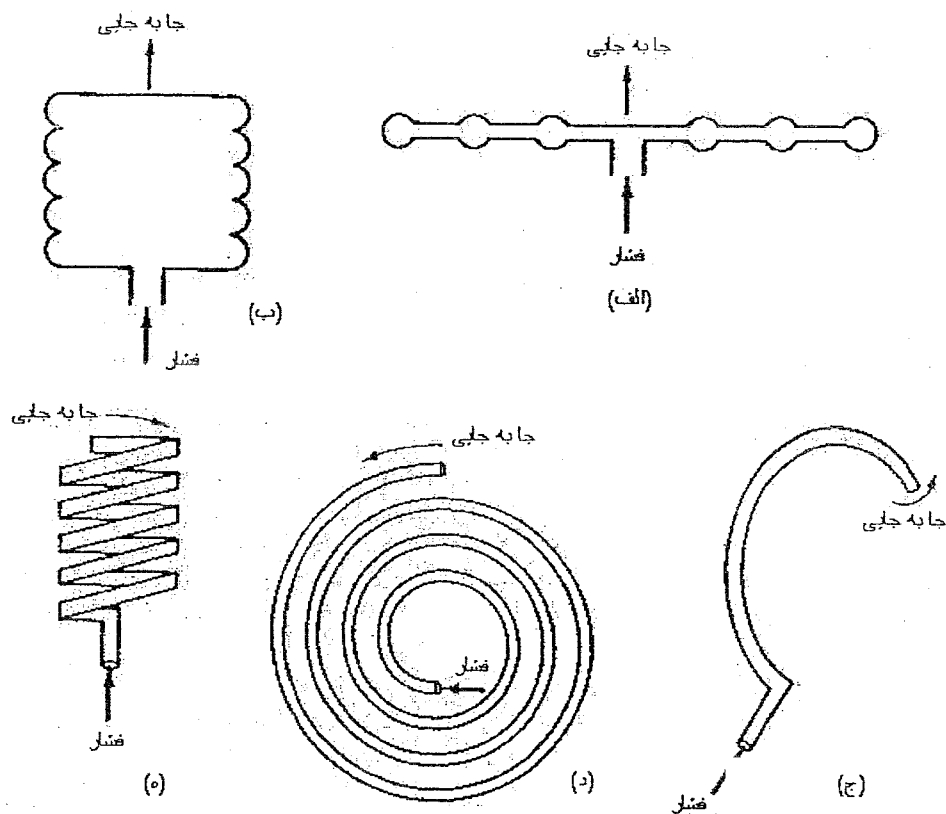


شکل ۷-۱۵ MPX100AP



شکل ۷-۱۶. استفاده از: (الف) دیافراگم، (ب) بلوز در سوئیچ فشار

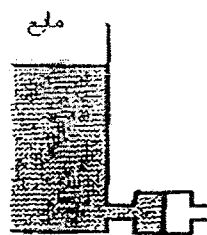
برای تبدیل فشار به جابه جایی علاوه بر دیافراگم از کپسول دیافراگمی، بلوز یا محفظه ی خرطومی شکل (Bellows)، بردن تیوب (Bourdon Tube) نیز استفاده می شود. (شکل ۷-۱۷) بلوز یک محفظه ی آکاردئونی شکل فلزی می باشد که در اثر اعمال فشار صفحات آن از یکدیگر باز می شوند و جابه جایی ایجاد می گردد. بردن تیوب یک لوله به شکل C و یا مارپیچ است که یک سمت آن مسدود و سمت دیگر آن به فشار تحت اندازه گیری وصل می باشد و متناسب با فشار لوله تغییر شکل داده و باز می شود.



شکل ۷-۱۷. تبدیل فشار به جابه جایی: (الف) کیسول، (ب) بلوز، (ج) بردن معمولی، (د) بردن حلغری، (ه) بردن مارپیچ

۷-۱-۵) اندازه گیری سطح مایعات

سنسورهای فشار همچنین برای اندازه گیری سطح مایعات در مخزن به کار می روند (شکل ۷-۱۸)

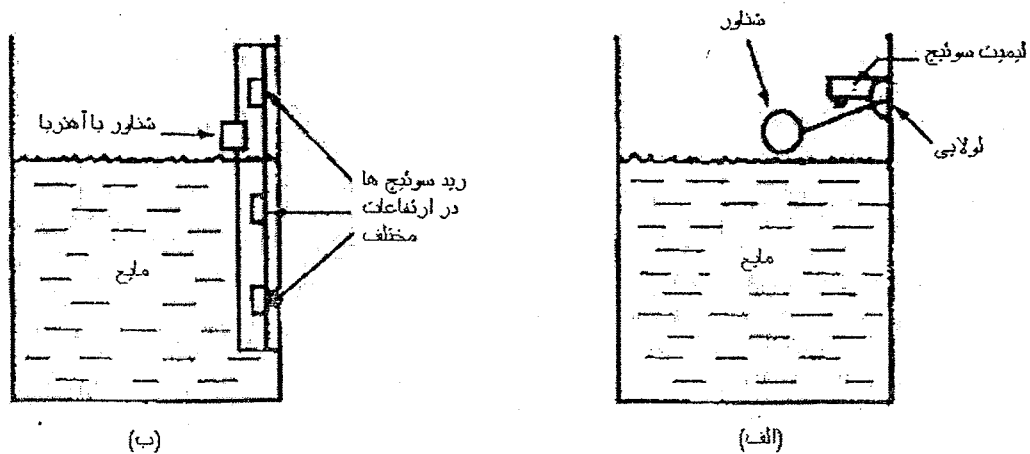


شکل ۷-۱۸. اندازه گیری سطح مایع توسط فشارسنج

فشار نسبی در کف یک مخزن که به ارتفاع h حاوی مایعی به جرم حجمی ρ باشد برابر است با:

$$P = h\rho g$$

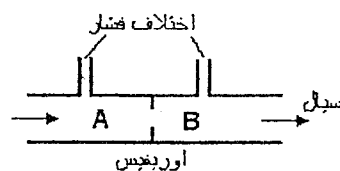
گاهی اوقات لازم است تا هنگامی که ارتفاع مایع به حد معینی در مخزن رسید یک سوئیچ قطع یا وصل گردد (Level Switch). شکل ۷-۱۹ دو روش استفاده از شناور برای این منظور را نشان می دهد.



شکل ۷-۱۹

۷-۱-۶ اندازه گیری جریان عبوری سیال (دبی)

یکی دیگر از استفاده های سنسورهای فشار، اندازه گیری دبی (flow) سیالات می باشد. بدین صورت که در مسیر عبور سیال یک محدودکننده ی جریان (یک صفحه با سوراخ معین در وسط آن که به اوریفیس (orifice) معروف است) نصب می شود. طبیعتاً در دو طرف این اوریفیس اختلاف فشار پدید می آید که با مجذور دبی متناسب است. (شکل ۷-۲۰)



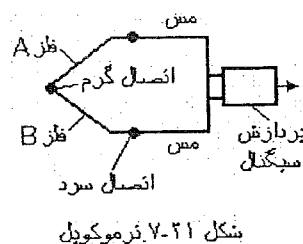
شکل ۷-۲۰

۷-۱-۷) اندازه گیری دما

الف) RTD (Resistive Temperature Detector): اساس کار RTD بر پایه ی افزایش خطی مقاومت فلزات در اثر افزایش دما است. رایجترین نوع RTD با نام تجاری PT100 (مقاومت آن در دمای 0°C معادل $100\ \Omega$) و ضریب افزایش این مقاومت $0.39\ \Omega/^{\circ}\text{C}$ است) از جنس پلاتین ساخته می شود.

باید توجه داشت که در RTD همانند کرنش سنج، جهت تبدیل تغییرات مقاومت به تغییرات ولتاژ از پل وتستون استفاده می شود.

ب) ترموکوپل: ترموکوپل نیز یکی از سنسورهای دما می باشد که از دو فلز غیرهم جنس A و B تشکیل شده است که در یک سر به یکدیگر متصل شده اند و هنگامی که این نقطه گرم می شود در دو سر دیگر این دو فلز ولتاژی تولید می گردد که متناسب با دمای نقطه ی اتصال و جنس دو فلز A و B می باشد (شکل ۷-۲۱).



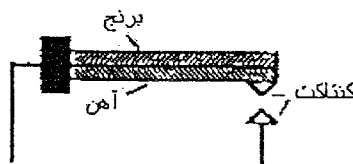
عملاً هنگام اندازه گیری این ولتاژ دو فلز A و B باید در ترمینال به دو سیم مسی متصل شوند بنابراین دو ترموکوپل دیگر به صورت ناخواسته در محل ترمینال ایجاد می گردد که ولتاژ تولیدی آن ها در جهت عکس ولتاژ ترموکوپل می باشد. بنابراین ولتاژ اندازه گیری شده در واقع متناسب با اختلاف دمای اندازه گیری توسط ترموکوپل و دمای محیط اتصال در نقطه ی ترمینال می باشد. به همین جهت این نقطه را اتصال سرد می نامند. برای اینکه دمای نقطه اتصال گرم به درستی اندازه گیری شود باید توسط یک سنسور دیگر دمای نقطه اتصال سرد اندازه گیری گردد. همچنین ولتاژ تولید شده توسط ترموکوپل کاملاً خطی نیست و عملاً در داخل مازول های آنالوگ مرتبط به ترموکوپل تغییرات این ولتاژ خطی می گردد.

گاهی اوقات لازم است تا هنگامی که دمای نقطه ی مورد نظر به میزان معینی رسید یک کنتاکت قطع و یا وصل گردد (Temperature Switch). برای این منظور می توان از ترمیستور، بی متال یا سیستم های کاپیلاری (لوله موئین) استفاده نمود.

اساس کار ترمیستور بر پایه ی کاهش شدید مقاومت نیمه هادی ها در اثر افزایش دما می باشد اما متاسفانه این تغییرات غیر خطی می باشد.

در شکل ۷-۲۲ یک بی مثال را مشاهده می کنید که از اتصال دو فلز غیرهم جنس تشکیل شده است. هنگام افزایش دما هر دو فلز منبسط می شوند ولی چون ضریب انبساط آن ها متفاوت است، مجموعه به شکلی خم می شود که فلزی که افزایش طولی آن بیشتر است قوس بیرونی را تشکیل می دهد و با خم شدن بی مثال یک کنتاکت وصل می گردد.

دماسنج کاپیلاری از یک فشارسنج بردن تیوب، لوله ی رابط موئین و یک مخزن محتوی سیال قابل انبساط تشکیل شده است. در اثر گرما سیال درون مخزن منبسط شده و از طریق لوله ی موئین سبب عملکرد سنسور فشار می گردد.

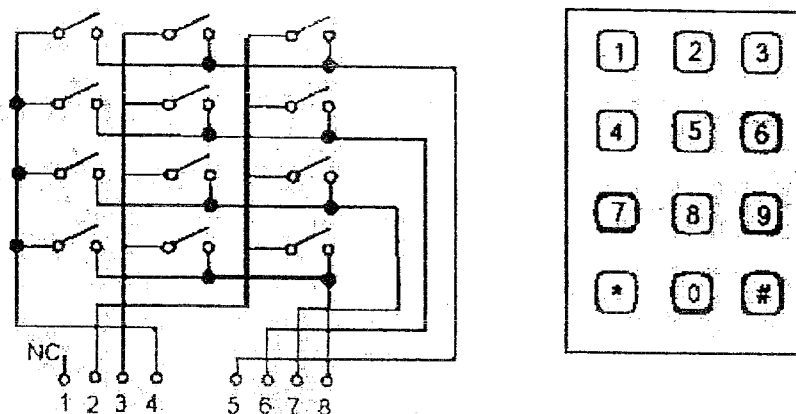


شکل ۷-۲۲. بی مثال

۸-۱-۷) صفحه کلید (Key Board)

از صفحه کلید جهت وازد کردن پارامترهای مورد نیاز سیستم های کنترل استفاده می شود. هر صفحه کلید از تعدادی کلید تشکیل شده که با فشار هر کدام یک کنتاکت وصل می شود.

شکل ۷-۲۳ یک صفحه کلید ۱۲ تایی را نشان می دهد که کلیدها به فرم ماتریسی بسته شده اند؛ بنابراین تعداد کمتری از ورودی های یک کارت PLC اشغال می شود.



شکل ۷-۲۲ صفحه کلید

۷-۲ انواع وسایل خروجی

وسایل خروجی که به PLC متصل می شوند نیز مانند وسایل ورودی می توانند از نوع دیجیتال (ON/OFF) یا آنالوگ باشند. در ادامه این بخش به معرفی کنتاکتور و سولنوئید والو که از جمله وسایل خروجی دیجیتال به حساب می آیند و شیر کنترل، از وسایل خروجی آنالوگ می پردازیم

۷-۲-۱ وسایل خروجی دیجیتال

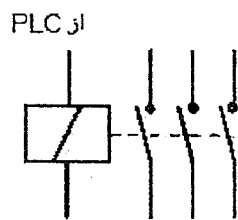
خروجی های دیجیتال PLC عموماً برای قطع و وصل مصرف کننده های کوچک (کمتر از چند آمپر) مناسب می باشند اما عملاً برای کنترل یک فرایند به سیگنال های کنترل با قدرت بیشتری نیاز می باشد. برای این منظور از وسایل خروجی استفاده می نمایم.

سولنوئید (Solonoid): اساس کار اکثر وسایل خروجی دیجیتال (ON/OFF) می باشد، و سیگنال الکتریکی را به حرکت مکانیکی تبدیل می نماید.

سولنوئید از یک سیم پیچ که به دور یک هسته تو خالی پیچیده می شود تشکیل شده است و با عبور جریان از این سیم پیچ میدان مغناطیسی ایجاد شده توسط آن هسته متحرک را به داخل می کشد.

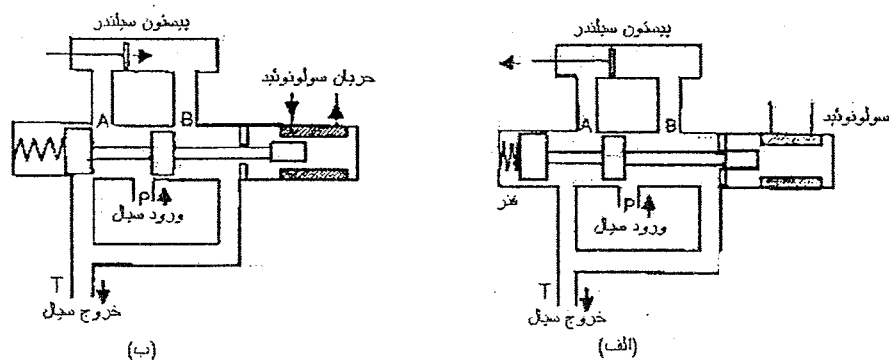
هنگامی که می خواهیم توسط PLC یک موتور پر قدرت را روشن نماییم، ابتدا با وصل شدن رله ی خروجی PLC جریانی به سمت سولنوئید کنتاکتور هدایت شده و میدان مغناطیسی ایجاد شده در آن، هسته متحرک را به سمت داخل کشیده و سبب وصل شدن کنتاکت های قدرت کنتاکتور می گردد و نهایتاً مسیر عبور جریان برای

موتور برقرار می شود (شکل ۷-۲۴) و با قطع شدن جریان فوق هسته ی متحرک به وسیله ی فنر به حالت اول خود بازگشته و موتور متوقف می گردد.



شکل ۷-۲۴. کنترل کنتور

اگر حرکت این هسته متحرک منجر به قطع و وصل شدن مسیر عبور سیال گردد آن را سولونوئید والو (شیر برقی) می نامند. شکل ۷-۲۵ طرز کار یک سولونوئید والو را نشان می دهد.



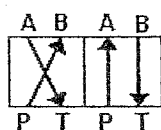
شکل ۷-۲۵. سولونوئید والو (الف) سولونوئید غیر فعال ، (ب) سولونوئید فعال

هوای فشرده (تولید شده توسط کمپرسور) و یا روغن هیدرولیکی تحت فشار (از طریق پمپ) از مسیر P وارد والو می شود و مسیر T نیز برای خارج شدن هوا به اتمسفر و یا برگشت روغن به مخزن ذخیره می باشد. شکل الف ۷-۲۵ والو را در حالتی که سیم پیچ سولونوئید آن غیرفعال است نمایش می دهد. بنابراین سیال به سمت راست سیلندر وارد شده و پیستون را به چپ می راند. همچنین سیالی که در طرف راست سیلندر، پشت پیستون قرار گرفته از طریق اتصال A وارد والو شده و از مسیر اتصال T به مخزن ذخیره (در روغن هیدرولیکی) و یا به اتمسفر (در هوای پنوماتیکی) برمی گردد.

در شکل ب-۲۵-۷ والو را در حالتی که سیم پیچ سولونوئید آن فعال است مشاهده می کنید با فعال شدن سولونوئید هسته ی متحرک به داخل کشیده می شود و در نتیجه سیال به سمت چپ سیلندر وارد شده و پیستون به سمت راست حرکت می نماید و از طریق مسیر B به T سیال پشت پیستون خارج می گردد. با غیرفعال شدن سولونوئید نیروی فنر ولو را به حالت اولیه باز می گرداند.

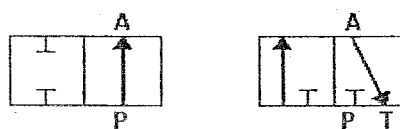
سولونوئید والوها براساس تعداد درجه های ورود و خروج سیال و تعداد وضعیت های موجود طبقه بندی می شوند. به عنوان مثال والو فوق الذکر یک شیر ۴/۲ می باشد. یعنی ۴ درجه A, B, T, P و دو وضعیت فعال و غیرفعال دارد.

در نقشه کشی شماتیک هر وضعیت را به صورت یک مربع نشان می دهد که در داخل هر مربع مسیرهای ارتباطی سیال در آن وضعیت نمایش داده شده و جهت فلش ها بیانگر جهت حرکت سیال می باشد. بنابراین نقشه شماتیک شیر ۴/۲ ذکر شده مطابق شکل ۲۶-۷ می باشد.



شکل ۲۶-۷. شیر ۴/۲

همچنین در شکل ۲۷-۷ شماتیک دو شیر ۲/۲ و ۳/۲ را می بینید که در این شماتیک روش فعال شدن شیر نمایش داده نشده است.

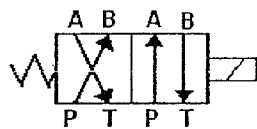


2/2

3/2

شکل ۲۷-۷. شیرهای جهت دار

شکل ۲۸-۷ روش های فعال و غیرفعال شدن سولونوئید والو را به صورت شماتیک و با استفاده از فنر، سولونوئید یا به صورت دستی نمایش می دهد. بنابراین شماتیک کامل والو نشان داده شده در شکل ۲۶-۷ مطابق شکل ۲۹-۷ خواهد بود.



شکل ۷-۲۹ شیر برقی



(ج)



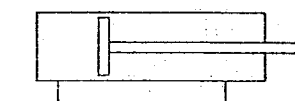
(ب)



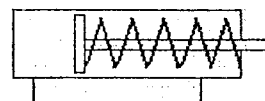
(الف)

شکل ۷-۲۸ حرکت توسط: (الف) سولنوئید (ب) شستی (ج) فنر

سیلندر تک کاره (Single Acting Cylinder): سیلندر تک کاره سیلندری است که سیال تحت فشار به یک طرف آن وارد می شود و بازگشت پیستون در طرف دیگر توسط یک فنر انجام می پذیرد. (شکل الف - ۷-۳۰). شکل ۷-۳۱ نحوه ی کنترل یک سیلندر تک کاره را توسط یک شیر ۳/۲ نشان می دهد.

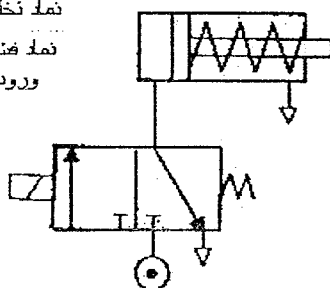
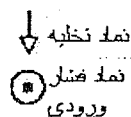


(ب)

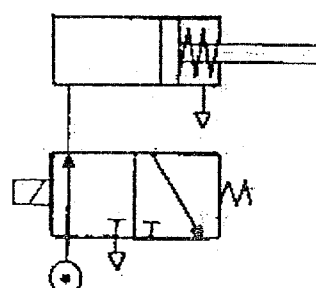


(الف)

شکل ۷-۳۰ سیلندر: (الف) تک کاره (ب) دو کاره



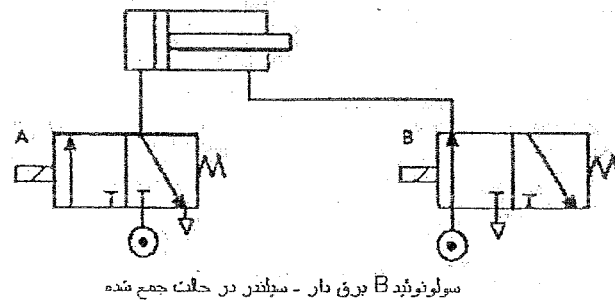
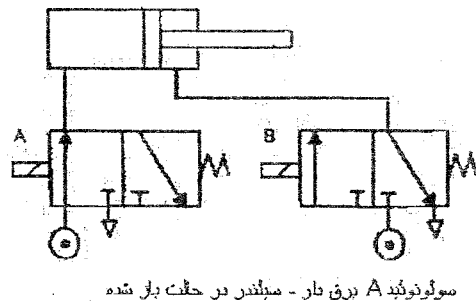
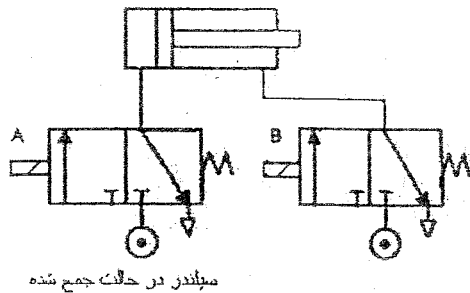
سیلندر در حالت جمع شده
سولنوئید بی برق



سیلندر در حالت باز شده
سولنوئید برق دار

شکل ۷-۳۱ نحوه ی کنترل سیلندر تک کاره توسط یک شیر ۳/۲

سیلندر دو کاره (Double Acting Cylinder): در سیلندر دو کاره، سیال تحت فشار به دو طرف سیلندر وارد می شود (شکل ب- ۷-۳۰). شکل ۷-۳۲ نحوه ی کنترل این سیلندر توسط دو وایو ۳/۲ را نشان می دهد.



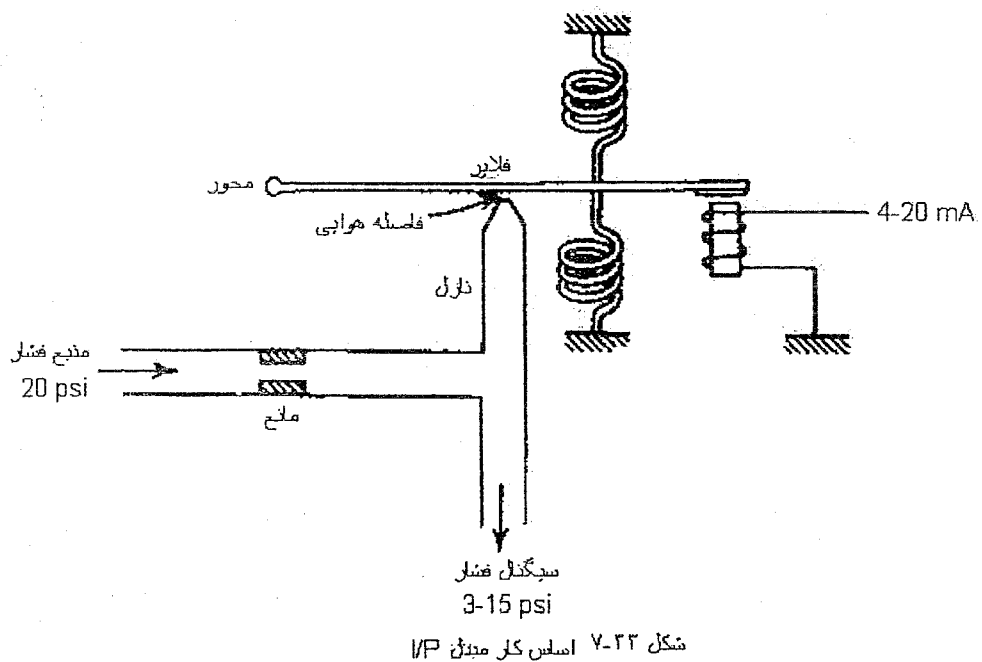
شکل ۷-۲۲. نحوه ی کنترل یک سیلندر دو کاره توسط دو شیر ۲/۲

۷-۲-۲) وسایل خروجی آنالوگ

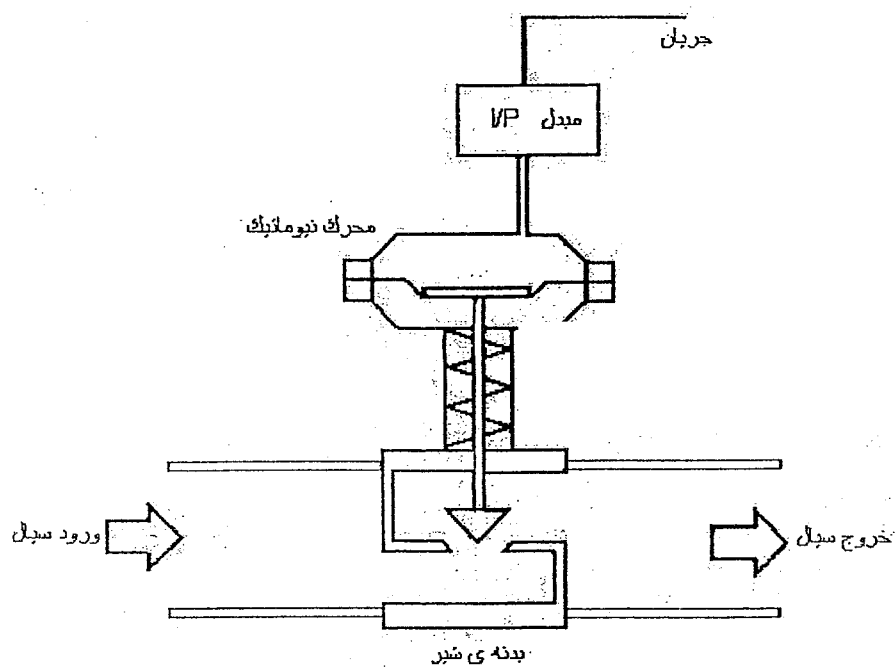
شیر کنترل: در کلیه ی صنایع شیمیایی لازم است که دبی عبوری سیال را کنترل نمائیم تا بتوانیم کیفیت محصولات را ثابت نگه داریم. جهت نیل به امر فوقی از شیر کنترل استفاده می گردد. با استفاده از شیر کنترل قادر خواهیم بود تا مسیر عبور یک سیال را از صفر تا ۱۰۰٪ در جهت نیاز باز و یا بسته نمائیم که نحوه ی کنترل فرایند به طور مثال به شرح زیر است:

مازول خروجی آنالوگ PLC سیگنال ۴ تا ۲۰ mA را به مبدل جریان الکتریکی به هوای فشرده (I to P یا I/P) می فرستد، در I/P این سیگنال تبدیل به هوای فشرده ۱۵ psi تا ۳ می شود (psi فشار معادل یک پوند بر یک اینچ مربع می باشد). شکل ۷-۳۳ ساختمان داخلی یک مبدل جریان الکتریکی به هوای فشرده را نشان می دهد. در I/P متناسب با جریان اعمالی، سولنوئید، فلاپر (flapper) را جذب می نماید و در یک نقطه با نیروی فنر به تعادل می رسد. هرچه این جریان بیشتر شود، فاصله ی هوایی کمتر شده و مسیر عبور هوا از نازل بسته تر می گردد و بنابراین فشار هوای خروجی افزایش پیدا می کند. به عنوان مثال اگر جریان ۴ mA به سولنوئید اعمال شود فشار هوای خروجی ۲ psi و در صورتی که جریان ۱۲ mA به سولنوئید اعمال گردد، فشار هوای خروجی به ۹ psi و هنگامی که جریان ۲۰ mA به سولنوئید داده شود فشار هوای خروجی معادل

۱۵ psi می گردد.



شکل ۷-۲۴ اساس شیر کنترل را نشان می دهد. خروجی I/P به پشت دیافراگم لاستیکی کنترل والو وصل می شود و با فشار فنر به تعادل می رسد و درجه متناسب با فشار مرکز دیافراگم حرکت می نماید و مسیر عبور سیال را باز یا بسته می کند.



شکل ۷-۲۴. ساختمان شیر کنترل

