بنام خداوند بخشنده ومهربان

عنوان مدرک :	شروع برنامه نویسی FATEK PLC
توضیحات :	معرفی و بررسی نرم افزار Winproladder
تعداد صفحه :	19
شماره ویرایش :	-
ویرایش کننده :	ا.رضایی
تاریخ ویرایش :	1392.04.02

شروع برنامه نویسی با نرم افزار WINPROLLADER

حافظه داخلی FATEK PLC :

در این پی ال سی ها ، رجیسترها بصورت تک بیتی یا 16بیتی قابل دسترسی می باشند (رجیسترهای 32بیتی از استفاده دو رجیستر 16بیتی بوجود می آیند)

رجیسترها دارای دو دسته هستند :

- **رجیسترهای معمولی :این رجیسترها توسط برنامه نویس در برنامه برای نگهداری اطلاعات و اعداد قابل استفاده هستند و بر دو دسته می باشند**

1- **Retentive** : **مقدار موجود در این رجیسترها با قطع و وصل برق صفر نمی شوند و ماندگار می مانند.**

2- **Non retentive** : **مقدار موجود در این رجیسترها با قطع و وصل برق پاک (صفر) می شوند .**

- **رجیسترهای خاص : (Special Registers) این رجیسترها رابط بین اطلاعات CPU و برنامه کاربر هستند برای مثال CPU مقدار ساعت و تاریخ داخلی PLC در رجیسترهای R4128~R4133 قابل دسترسی می باشد.**

رجیسترهای بیتی :

X0 ~ X255	**ورودی**	X
Y0 ~ Y255	**خروجی**	Y
M1400~M1911	**رجیسترهای بیتی معمولی غیر ماندگار**	M
M0~M1399	**رجیسترهای بیتی معمولی قابل تنظیم برای ماندگار بودن یا غیر ماندگار بودن**	
M1912 ~ M2001	**رجیسترهای بیتی خاص**	

تعدادی از رجیسترهای بیتی خاص :

(تمام رجیسترهای خاص از مسیر Special Registers << HELP قابل دسترسی می باشند)

شماره رجیستر بیتی خاص	توضیحات
M1912	با یک شدن این بیت CPU به حالت استپ می رود و تموم خروجی ها خاموش می شوند . با خاموش و روشن کردن ، PLC دوباره RUN می شود
M1913	با یک شدن این بیت خروجی های سخت افزاری خاموش می شوند ولی وضعیت خروجی ها در برنامه در همان حالت باقی می مانند.
M1914	با یک شدن این بیت ، رجیسترهای بیتی غیر ماندگار صفر می شوند
M1915	با یک شدن این بیت ، رجیسترهای بیتی ماندگار صفر می شوند
M1916	با یک شدن این بیت ، رجیسترهای 16بیتی غیر ماندگار صفر می شوند
M1917	با یک شدن این بیت ، رجیسترهای 16بیتی ماندگار صفر می شوند
M1920	بوسیله این بیت ، CPU هر 0.01 ثانیه یک پالس تولید می کند.
M1921	بوسیله این بیت ، CPU هر 0.1 ثانیه یک پالس تولید می کند.
M1922	بوسیله این بیت ، CPU هر 1 ثانیه یک پالس تولید می کند.
M1923	بوسیله این بیت ، CPU هر 60 ثانیه یک پالس تولید می کند.
M1924	وقتی CPU از حالت STOP به RUN یا از حالت خاموش به روشن می رود ، فقط در اسکن اول برنامه این بیت 1 می شود و تا آخر صفر می ماند.
M1952	برای تغییر ساعت و تاریخ داخلی PLC این بیت باید ابتدا 1 شود و پس از تنظیم ساعت و تاریخ ، باید صفر شود .
M1957	با یک شدن این بیت ، وقتی تایمر به مقدار تنظیم شده رسید در این مقدار باقی می ماند.

رجیسترهای 16 بیتی :

D0~D4000	رجیسترهای 16بیتی معمولی	D
D4001~D4095	رجیسترهای 16بیتی خاص	
R0~R3839	رجیسترهای 16بیتی معمولی قابل تنظیم برای ماندگار بودن یا غیر ماندگار بودن	R
R5000~R8071	رجیسترهای 16بیتی معمولی ماندگار	
R3840~R4200	رجیسترهای 16بیتی خاص	

تعدادی از رجیسترهای 16بیتی خاص :

(تمام رجیسترهای خاص از مسیر HELP >> Special Registers قابل دسترسی می باشند)

توضیحات	Register No.
رجیسترهای ورودی آنالوگ	R3840 ~ R3903
رجیسترهای خروجی آنالوگ	R3904 ~ R3967
تنظیم Baud Rate پورت 0	R4050
مقدار مربوط به تنظیمات پارامترهای پورت 1	R4146
مقدار مربوط به تنظیمات پارامترهای پورت 2	R4158
مقدار مربوط به تنظیمات پارامترهای پورت High Speed 2	R4161
مقدار مربوط به تنظیمات پارامترهای پورت 3	R4043
مقدار مربوط به تنظیمات پارامترهای پورت 4	R4044
تنظیم پروتکل های ارتباطی پورتهای 1~4	R4047
8 بیت کم ارزش شماره استیشن می باشد (مقدار 8 بیتهای باارزش را برابر با مقدار H55 قرار می دهیم)	R4055
"پارامتر ثانیه" مربوط به ساعت داخلی CPU	R4128
"پارامتر دقیقه" مربوط به ساعت داخلی CPU	R4129
"پارامتر ساعت" مربوط به ساعت داخلی CPU	R4130
"پارامتر روز" مربوط به تاریخ داخلی CPU	R4131
"پارامتر ماه" مربوط به تاریخ داخلی CPU	R4132
"پارامتر سال" مربوط به تاریخ داخلی CPU	R4133
چندمین روز هفته	R4134
زمان اسکن برنامه	R4136
رجیستر V برای آدرس دهی غیر مستقیم	R4164
رجیستر Z برای آدرس دهی غیر مستقیم	R4165
رجیستر P0 برای آدرس دهی غیر مستقیم	D4080
رجیستر P1 برای آدرس دهی غیر مستقیم	D4081
رجیستر P2 برای آدرس دهی غیر مستقیم	D4082
رجیستر P3 برای آدرس دهی غیر مستقیم	D4083
رجیستر P4 برای آدرس دهی غیر مستقیم	D4084
رجیستر P5 برای آدرس دهی غیر مستقیم	D4085
رجیستر P6 برای آدرس دهی غیر مستقیم	D4086
رجیستر P7 برای آدرس دهی غیر مستقیم	D4087
رجیستر P8 برای آدرس دهی غیر مستقیم	D4088
رجیستر P9 برای آدرس دهی غیر مستقیم	D4089

برنامه نویسی PLC به زبان LADDER :

نحوه ی انجام عملیات در سیستمPLC به صورت زیر است :

PLC تمام ورودی ها را چک می کند (Scan Inputs)، ورودی هایی که وصل هستند از نظر PLC معادل " یک" و ورودی هایی که قطع هستند معادل "صفر" قرار داده می شوند .

CPU برنامه موجود در حافظه را خط به خط خوانده و اجرا می کند و پس از پایان اجرای برنامه ، وضعیت خروجی ها را به واحد خروجی می فرستد و این سیکل مجدداً از ابتدا آغاز می شود.

کل زمان انجام مراحل ۱تا۳برابر است با Scan Inputs + Scan Program+ Scan Outputsو آن را Scan Time می نامند.

چنانچه این زمان بیشتر از 0.25 ثانیه گردد، نشان دهنده ی این مطلب می باشد، که یکی از قسمت هایPLC دچار اشکال شده بنابراین تایمر سگ نگهبان (Watch Dog Timer) عمل نموده و تمامی خروجی ها را غیرفعال می کند تا عملکرد اشتباه PLCمنجر به حادثه نگردد. این زمان پیش فرض، از طریق تابع ۹۰ قابل تغییر است .

فرض کنید که در یک برنامه باید با وصل یک ورودی ، یک خروجی فعال گردد.حال اگر تصادفاً ورودی در لحظه ای وصل شود که PLC ، مرحله خواندن ورودی ها را به انجام رسانده باشد،در این صورت بایدبه اندازه ی یک اسکن کامل صبر کند تا وضعیت این ورودی به CPU انتقال یابد ، 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 باشندکه این نیز به نوبه ی خود تاخیری را در دریافت ورودی ایجاد می نماید(حدود 10ms)، همچنین اگر خروجی از نوع رله ای باشد مدت زمانی حدود 10ms نیز برای وصل رله ی خروجی خواهیم داشت ، مجموع این دو زمان را تاخیر سخت افزاریPLC می نامند.

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

نرم افزار برنامه نویسی PLC FATEK "WinProladder" می باشد. کاربر بوسیله این نرم افزار می تواند مستقیماً برنامه موجود در حافظهPLC را مشاهده و تغییر دهد و یا ابتدا برنامه را در داخل کامپیوتر شخصی بنویسد و سپس در موقع مناسب آن را به PLC منتقل نماید و قابلیت اجرای برنامه "RUN" یا "STOP" از محیط این نرم افزار انجام می شود .

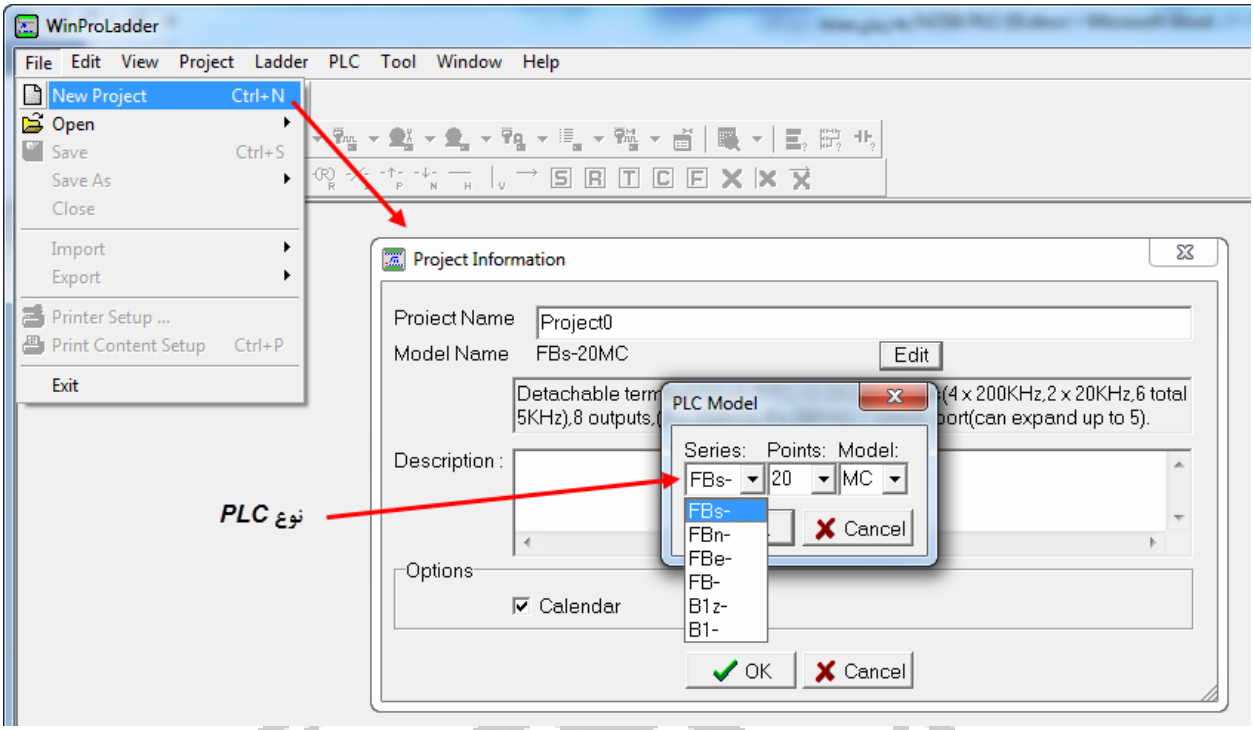
برخی قابلیت های نرم افزار برنامه نویسی FATEK (Win Proladder) به شرح زیر می باشد:

- **امکان نوشتن برنامه به صورت Off Line و ذخیره آن به صورت یک فایل جهت دسترسی دوباره به برنامه فوق .**
- **مشاهده ی اجرای یک برنامه روی PLC (On Line Monitoring).**
- **مشاهده ی اجرای یک برنامه بدون استفاده از PLC (Offline Simulation)**
- **قابلیت قطع و وصل هر ورودی یا فعال و غیرفعال کردن هر خروجی .**
- **امکان تغییر برنامه در حالت (RUN) ، که از قابلیت های منحصر به فرد این نرم افزار می باشد.**
- **امکان نظارت و تغییر حافظه ی داخلی PLC از طریق صفحه ی مانیتورینگ (Status Page)**
- **امکان پیدا کردن سریع هر ورودی یا خروجی دلخواه (search) در برنامه و جایگزین نمودن آن ها.**
- **امکان قرار دادن توضیحات اضافی در برنامه(Comments)**
- **امکان قرار دادن رمز (Password) برای کل برنامه یا فقط زیر برنامه ها.**
- **امکان اتصال PLC و PC با روش های متنوع (اتصال مستقیم از طریق RS232 ، USB ، Ethernet و اتصال از راه دور به کمک مودم خط تلفن).**

معرفی منوهای نرم افزار Winprollader :

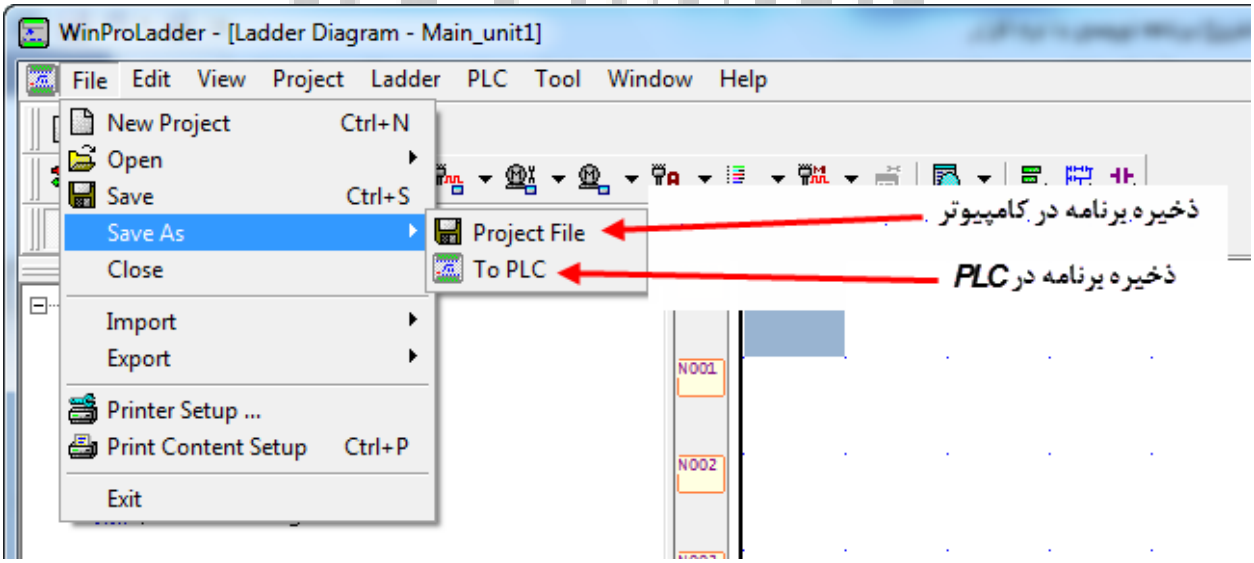
ایجاد پروژه جدید :

از منوی File و انتخاب گزینه New Project و انتخاب نوع PLC می توان پروژه جدیدی را ایجاد کرد و در آن برنامه مورد نظر را نوشت .

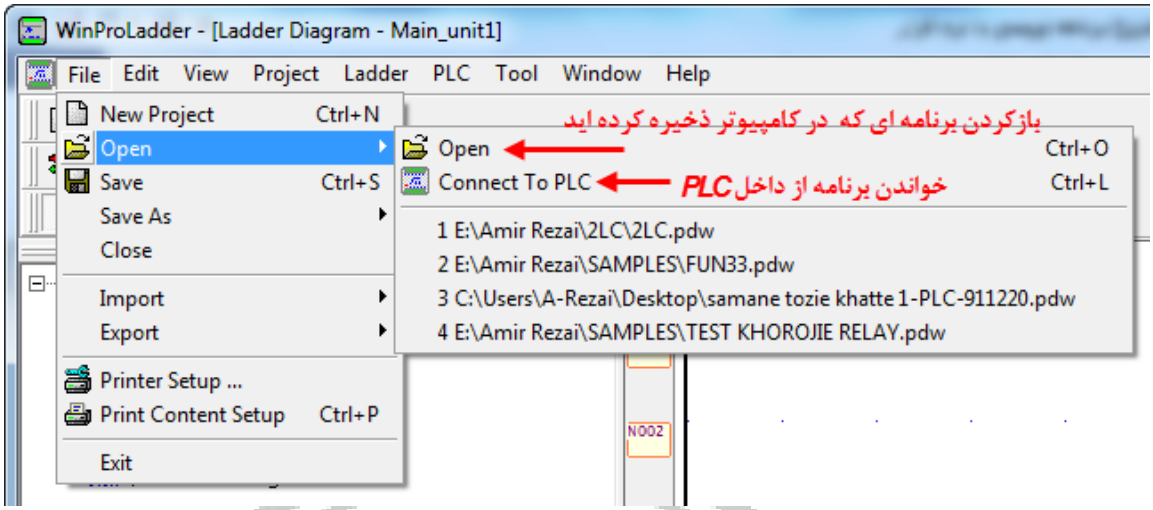


ذخیره برنامه :

در منوی **File** با استفاده از گزینه **Save As >> Project File** می توان برنامه را بعنوان یک فایل با فرمت **pdw.** در کامپیوتر ذخیره کرد یا با استفاده از گزینه **Save As >> To PLC** برنامه نوشته شده را در حافظه **PLC** ذخیره کرد.

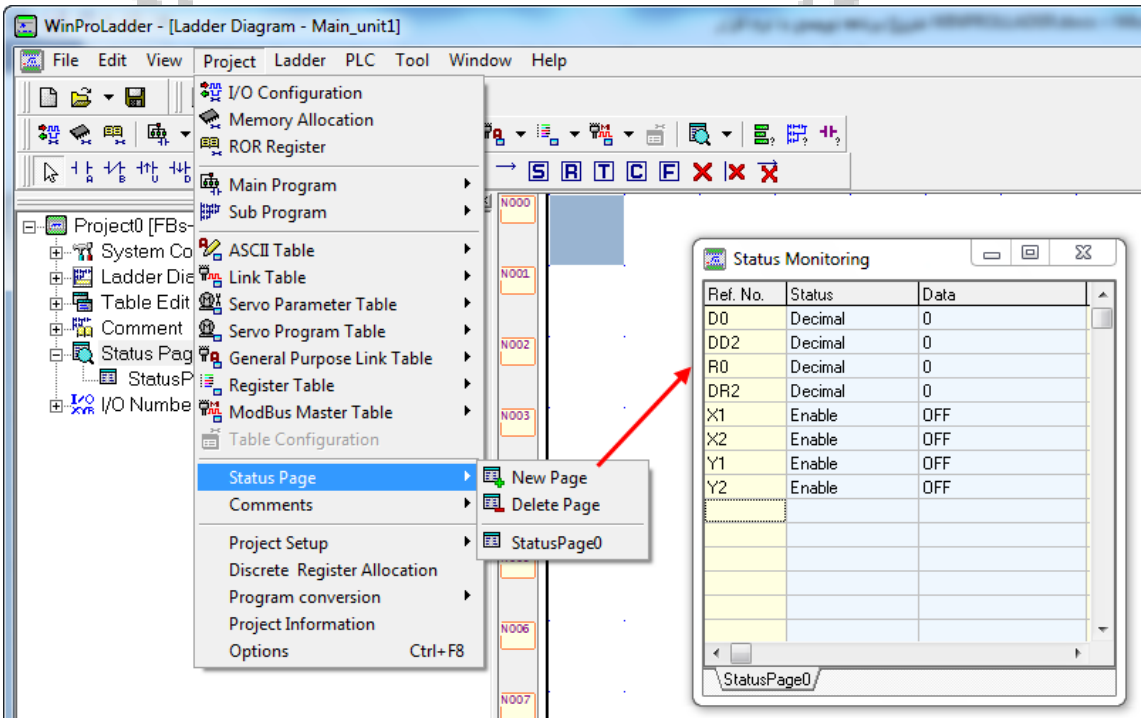


باز کردن برنامه یا خواندن برنامه از PLC :



مانیتور کردن رجیسترها و ورودی ها و خروجی های PLC :

با استفاده از صفحات Status Page می توان نام رجیسترها یا ورودی و خروجی های مورد نیاز را در ستون Ref. No. نوشت و مقادیر جاری آنها را مشاهده یا تغییر داد .



اختصاص کلمه عبور به پروژه :

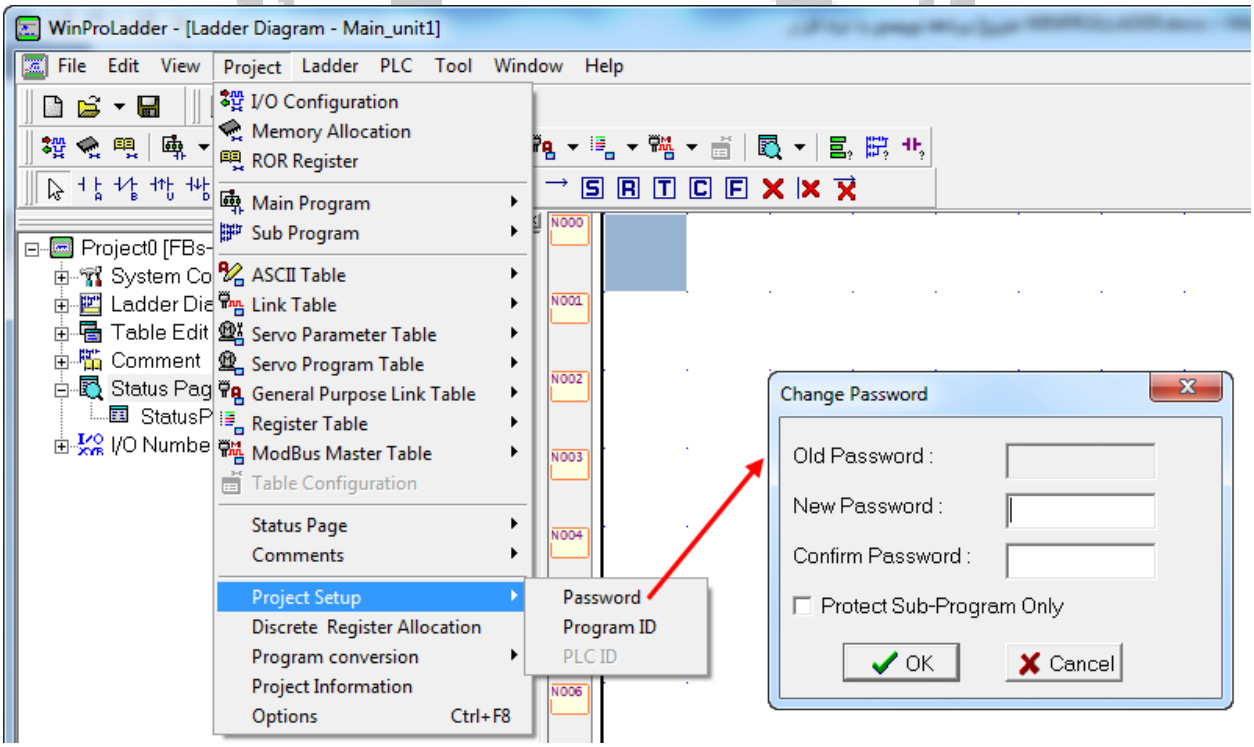
دو نوع کلمه عبور را می توان به برنامه نوشته اختصاص داد :

1- کلمه عبور به کل برنامه : با انتخاب نکردن عبارت Protect Sub-program Only ، کلمه عبور به کل برنامه های نوشته شده اختصاص می یابد ،در این حالت برای اینکه برنامه را باز کنیم ابتدا باید کلمه عبور را وارد کنیم .

2- اختصاص دادن کلمه عبور به زیر برنامه ها : با تیک زدن عبارت Protect Sub-program Only کلمه عبور به زیربرنامه های نوشته شده اختصاص می یابد ، در این حالت برنامه باز می شود ولی فقط صفحه Main برنامه باز می شود و برای باز کردن صفحات زیر برنامه باید کلمه عبور را وارد کنیم .

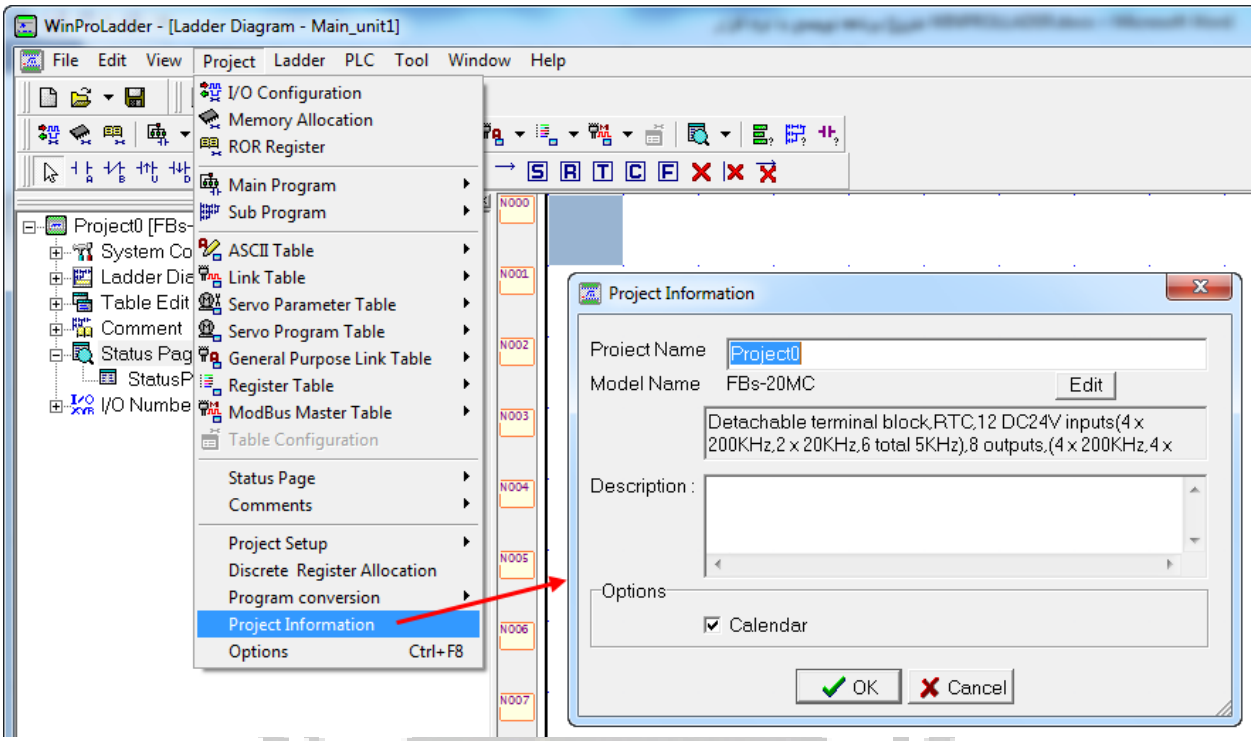
PLC ID :

در PLC های FATEK می توان به برنامه نوشته شده یک مشخصه (ID) که می تواند شامل حروف و اعداد باشد اختصاص داده می شود و به PLC نیز یک مشخصه اختصاص داده می شود، اگر برنامه ای را بخواهیم در یک PLC اجرا کنیم باید Program ID و PLC ID با یکدیگر برابر باشند . گزینه PLC ID زمانی که به PLC آنلاین باشیم فعال می شود.



تغییر نوع PLC در برنامه :

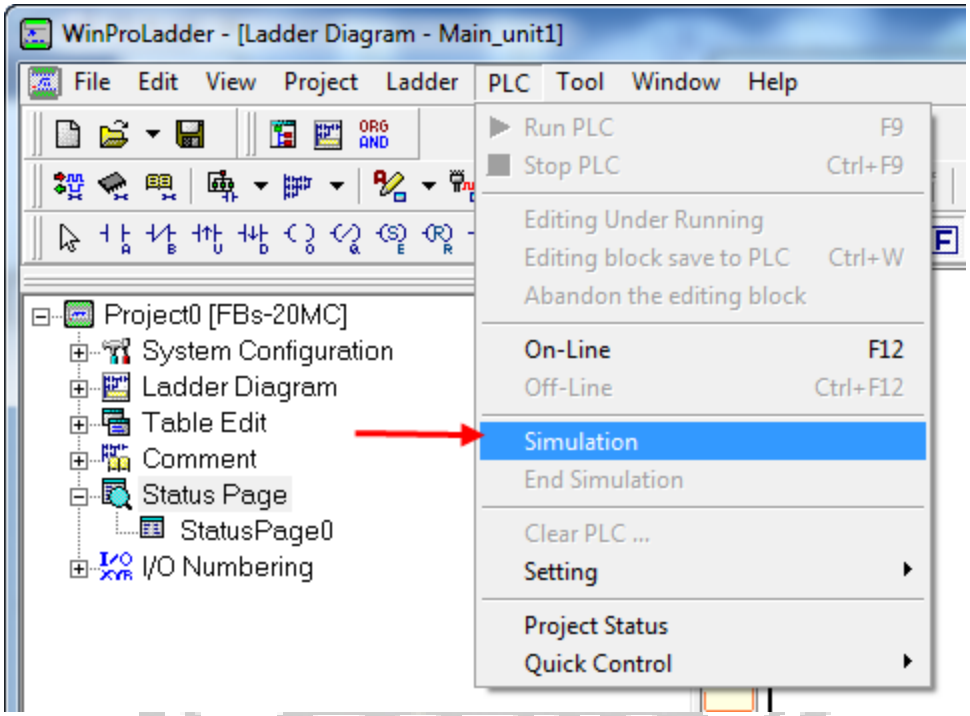
در منوی Project و با انتخاب گزینه Project information می توان صفحه مشخصات PLC معرفی شده به برنامه را باز کرد و نوع PLC انتخاب شده را عوض کرد .



شبیه سازی برنامه :

در مواردی که بدون PLC می خواهیم برنامه را تست کنیم می توانیم از مد Simulation استفاده کنیم .

با فعال کردن گزینه Simulation و سپس اجرای Run ، برنامه اجرا می شود .



تنظیم پورتهای ارتباطی PLC :

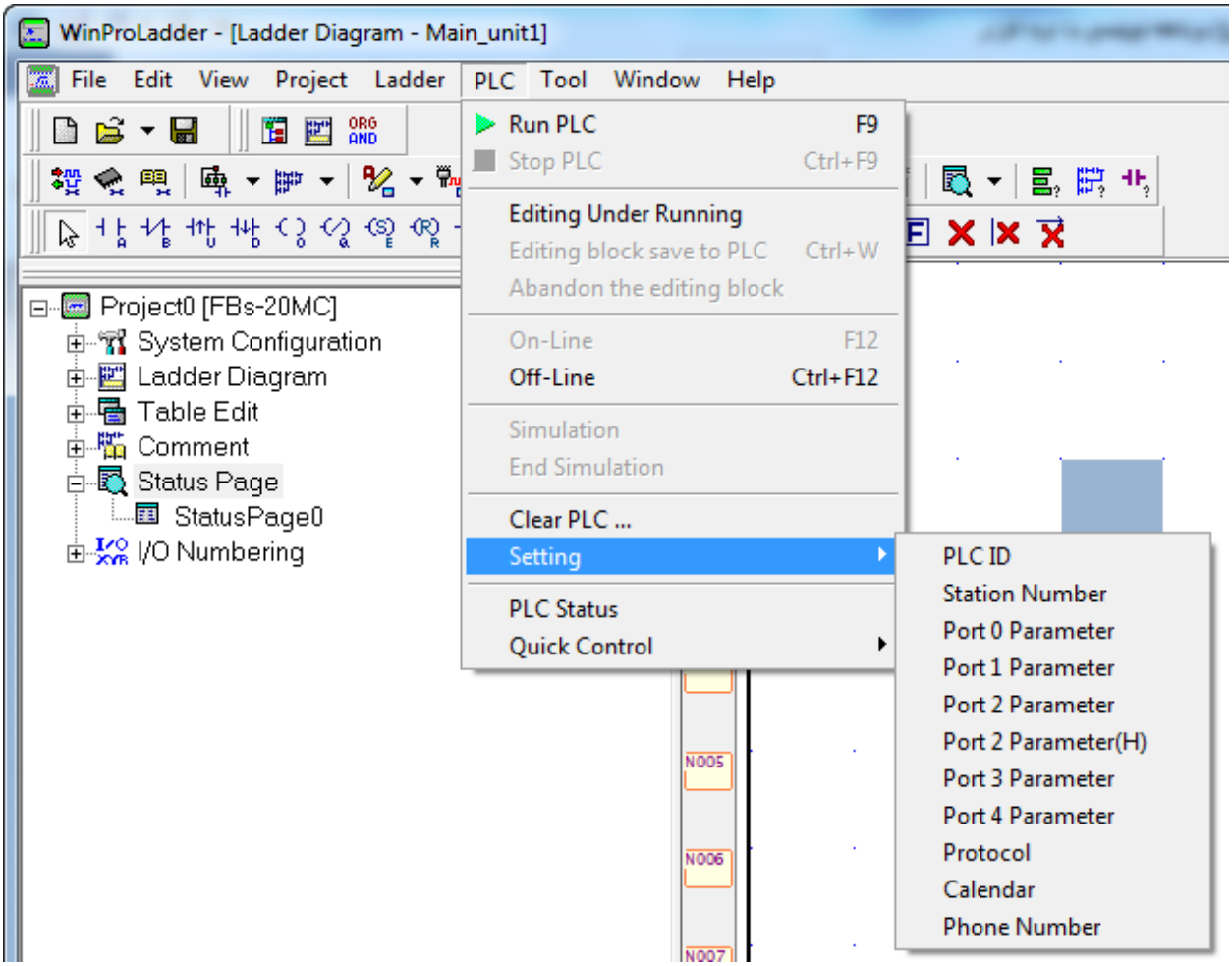
با انتخاب منوی PLC و گزینه Setting در حالتی که برنامه به PLC آنلاین می باشد ، گزینه های تنظیمات پورتهای ارتباطی فعال می باشند، توسط این گزینه ها می توان Baude rate , Parity , Data bits , Stop bit و پروتکل ارتباطی که عبارتند از : Fatek Communication protocol , Modbus RTU (Slave) , Modbus Ascii (Slave) .

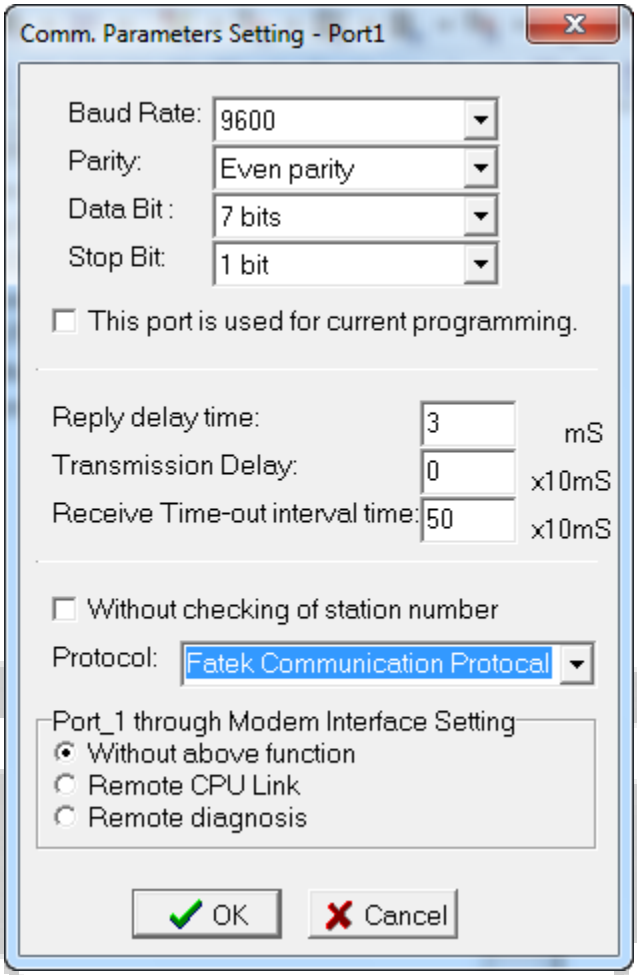
لازم به ذکر است که در پارامترهای پورت صفر فقط Baude rate قابل تغییر است و بقیه پارامترها برابر غیر قابل تغییر می باشند

پارامترهای پیش فرض پورتها مطابق جدول زیر می باشند :

Baude Rate	۹۶۰۰
Parity	Even
Data bits	۷
Stop bit	۱

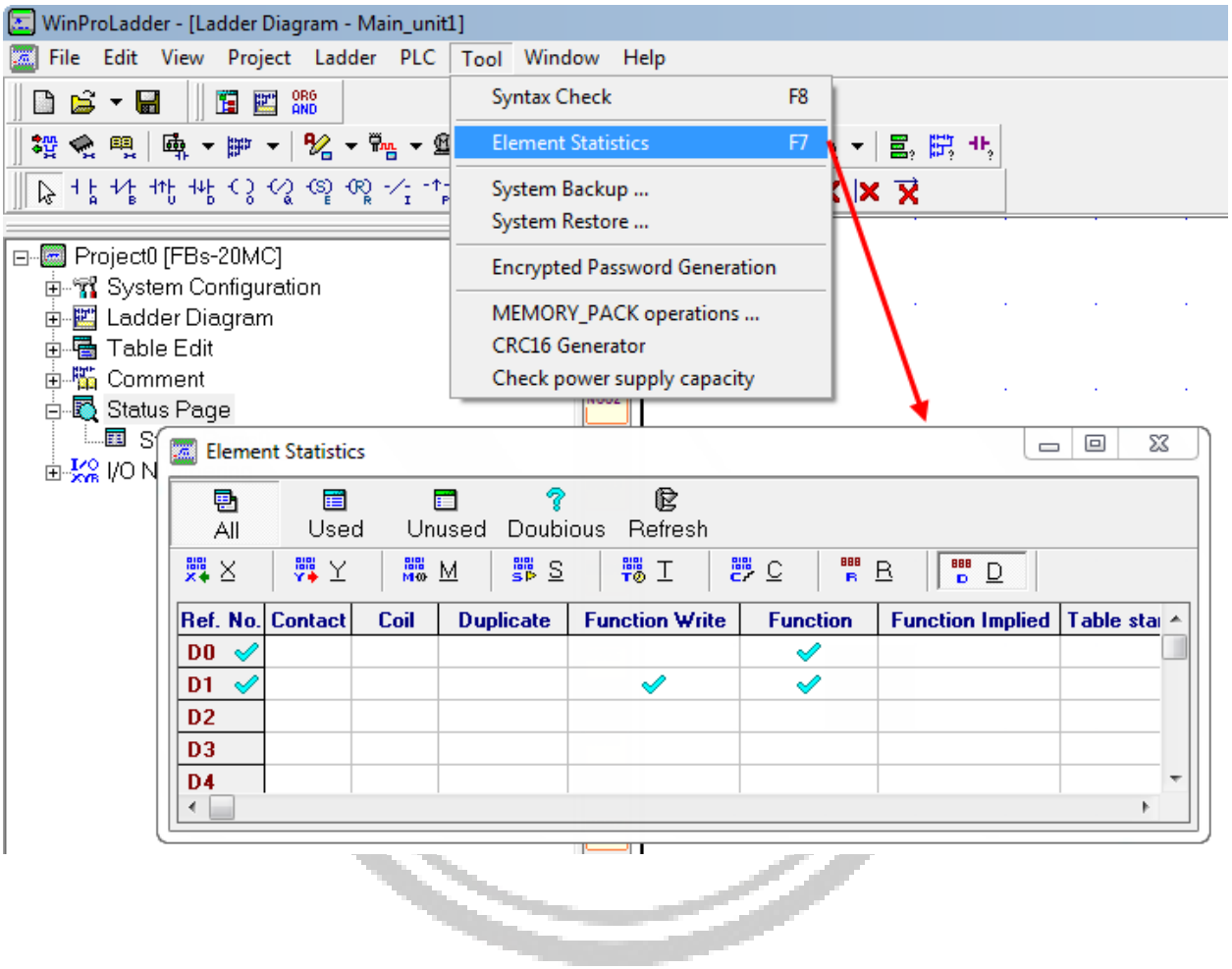
Reply delay time	۳
Transmission delay	۰
Receive time out interval time	۵۰
protocol	Fatek Communication protocol




Comm. Parameters Setting - Port1
Baud Rate: 9600
Parity: Even parity
Data Bit : 7 bits
Stop Bit: 1 bit
This port is used for current programming.
Reply delay time: 3 mS
Transmission Delay: 0 x10mS
Receive Time-out interval time: 50 x10mS
Without checking of station number
Protocol: Fatek Communication Protocal
Port_1 through Modem Interface Setting
Without above function
Remote CPU Link
Remote diagnosis
OK
Cancel

متغیرهای استفاده شده و استفاده نشده در برنامه :

با انتخاب از منوی Tool و گزینه Element Statistics می توان از شماره متغیرهای استفاده شده (Used) و یا استفاده نشده (Unused) آگاهی یافت .



برنامه نویسیPLC

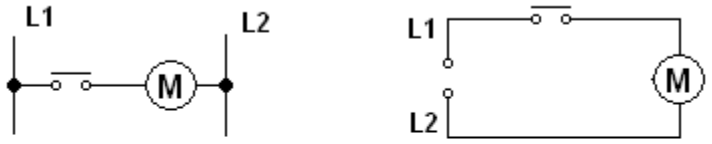
زبان ماشین مجموعه ای از کدهای باینری می باشد که تنها برای ریزپردازنده قابل درک است. از این رو برنامه نویسی با آن برای مهندسین دشوار می باشد. جهت سهولت در امر برنامه نویسیPLC- همانند کامپیوتر که ابتدا برنامه به زبان های سطح بالا نظیر C و Basic نوشته شده و سپس توسط کامپایلر به زبان ماشین تبدیل می شود- شرکت های سازنده یPLC نیز هر کدام از زبان های سطح بالا ی 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 درPLC منتشر ساخت. باوجود این هنوز به دلایل بسیاری، سازندگانPLC از زبان های مختص به خود استفاده می نمایند.

۱-۴) 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 یکPLC نیاز به تبدیل مدارهای فرمان با زبان برنامه نویسیPLC می باشد.استفاده از زبانLD (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.



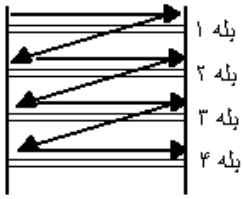
(ب) (الف)

شکل ۱-۴ . ترسیم یك مدار الکتریکی به صورت : الف) شماتیك ، ب) دیاگرام نردبانی

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

1- هر خط از برنامه(هر پله نردبان) وظیفه ی خاصی را به عهده دارد.

2- درPLC برنامه از سمت چپ به راست و از بالا به پایین اجرا می گردد و بعد از اجرای کامل برنامه ، اجرای آن دوباره از سرگرفته می شود توجه فرمایید که اگرچه شکل ظاهری دیاگرام نردبانی در مدارهای فرمان و برنامه های PLC یکسان است اما نحوه ی پردازش آن ها متفاوت می باشد.



شکل ۲-۴. چگونگی اجرای یک برنامه در PLC

3- هر خط برنامه با تعدادی کنتاکت باز و یا بسته آغاز و با یک یا چند بوبین رله به انتها می رسد

4- کنتاکت ها در وضعیت عادی خود در برنامه نشان داده می شوند به عبارت دیگر کنتاکت های کمکی ، با فرض غیرفعال بودن رله ها نمایش داده می شوند

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

6- هرکدام از کنتاکت های ورودی و رله های خروجی دارای آدرس منحصربه فرد می باشند. به عنوان مثالFBs - 40MA دارای ۲۴ورودی و۱۶ 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 دیاگرام نردبانی

۲-۴) نوار المان ها

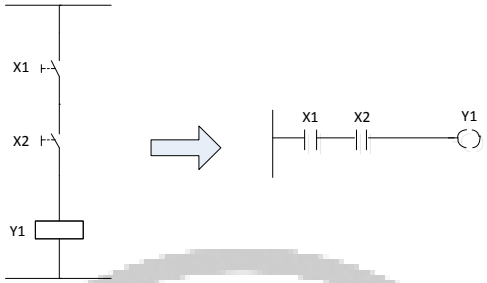
آیتم های این نوار برای ایجاد و ویرایش برنامه به کار گرفته می شوند.

⊣ ⊢	"یک" بودن بیت مورد نظر را نشان می دهد
⊣/⊢	" صفر"بودن بیت مورد نظر را نشان می دهد
⊣↑⊢	"لبه بالا رونده" بیت مورد نظر را نشان می دهد
⊣↓⊢	"لبه پایین رونده" بیت مورد نظر را نشان می دهد
—( )	خروجی بیتی
—(/)	معکوس خروجی بیتی
—(S)	یک کردن بیت
—(R)	صفر کردن بیت
—/—	معکوس کردن خط
—↑—	لبه بالا رونده خط
—↓—	لبه پایین رونده خط
S	یک کردن رجیستر (۱بیتی ، ۱۶بیتی ، ۳۲بیتی)
R	صفر کردن رجیستر (۱بیتی ، ۱۶بیتی ، ۳۲بیتی)
T	تایمر
C	شمارنده
F	تمام دستورات نرم افزار

با ذکر چند مثال برنامه نویسی LADDER را توضیح می دهیم

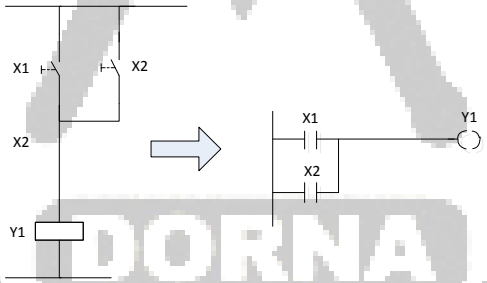
❖ مثال 1) کاربري AND:

در اينجا خروجي Y0 تنها وقتي فعال مي شود كه هر دو ورودي X0 و X1 وصل شده باشند.



مثال 2) كاربري OR

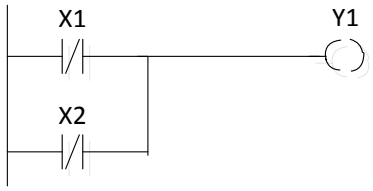
در اين جا خروجي در صورتي فعال مي شود كه هر كدام از ورودي هاي A يا B يا هر دو وصل شوند.



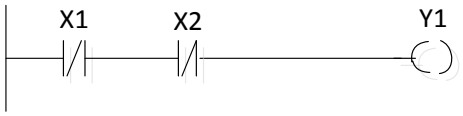
مثال 3) كاربريNOT

در اين وضعيت بوبين خروجي Y0 و ورودي X0 عكس يكديگر مي باشند.

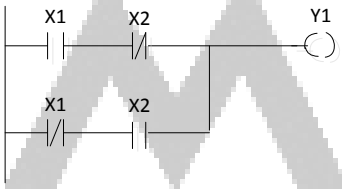
مثال 4) كاربري NAND



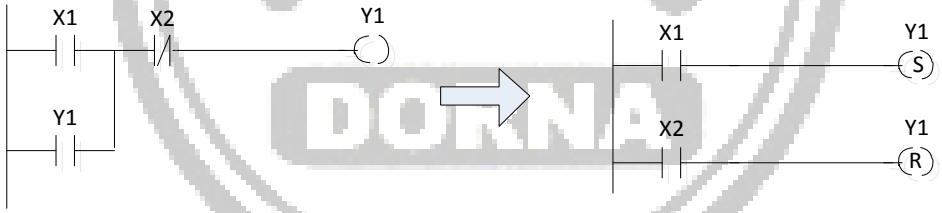
مثال 5) کاربري NOR



مثال 6) کاربري XOR



مثال 7) مدار خود نگهدار



بنام خداوند بخشنده ومهربان

عنوان مدرک :	سخت افزار FATEK PLC FBs
توضیحات :	ماژولهای آنالوگ
تعداد صفحه :	6
شماره ویرایش :	-
ویرایش کننده :	ا.رضایی
تاریخ ویرایش :	1392.02.04

ماژولهای آنالوگ

ماژولهای آنالوگ به دو دسته تقسیم می شوند

1- ماژولهایی که برروی PLC قرار می گیرند

2- ماژولهایی که از سمت راست به PLC اضافه می شوند

ماژولهای آنالوگ که برروی PLC قرار می گیرند با نام FBs-B... نامگذاری می شوند:

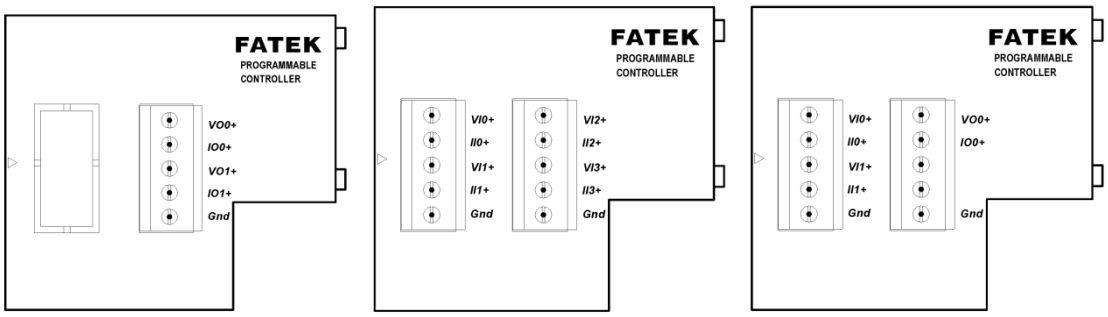
FBs-B2DA	(0~10V or 0~20mA)	2 کانال خروجی آنالوگ
FBs-B2A1D	(0~10V or 0~20mA)	2 کانال ورودی آنالوگ + 1 کانال خروجی آنالوگ
FBs-B4AD	(0~10V or 0~20mA)	4 کانال ورودی آنالوگ

کانالهای ورودی آنالوگ مقادیر ورودی را در آدرسهای D4072~D4075 ، بصورت عددی در بازه [0 , 16380] نشان می دهد

D4072	ریجیستر مربوط به کانال **1** ورودی
D4073	ریجیستر مربوط به کانال **2** ورودی
D4074	ریجیستر مربوط به کانال **3** ورودی
D4075	ریجیستر مربوط به کانال **4** ورودی

وقتی عددی در بازه [0 , 16380] را در ریجیسترهای D4076~D4077 قرار دهیممقدار ولتاژ یا جریان آنالوگ ، متناسب با عدد وارد شده در خروجی ظاهر می شود

D4076	ریجیستر مربوط به کانال 1 خروجی
D4077	ریجیستر مربوط به کانال 2 خروجی



FBs-B2DA FBs-B4AD FBs-B2A1D

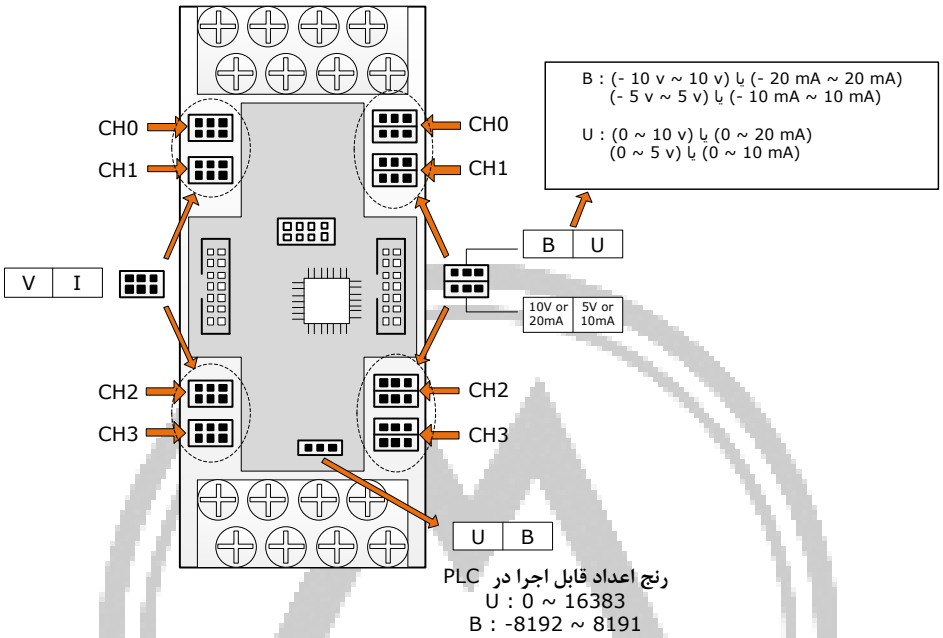
ماژولهای آنالوگ که از سمت راست به PLC اضافه می شوند :

FBs-2DA	(-10~10V, 0~10V or -20~20mA, 0~20mA)	2 کانال خروجی آنالوگ
FBs-4DA	(-10~10V, 0~10V or -20~20mA, 0~20mA)	4 کانال خروجی آنالوگ
FBs-4A2D	(-10~10V, 0~10V or -20~20mA, 0~20mA)	4 کانال ورودی آنالوگ + 2 کانال خروجی آنالوگ
FBs-6AD	(-10~10V, 0~10V or -20~20mA, 0~20mA)	6 کانال ورودی آنالوگ

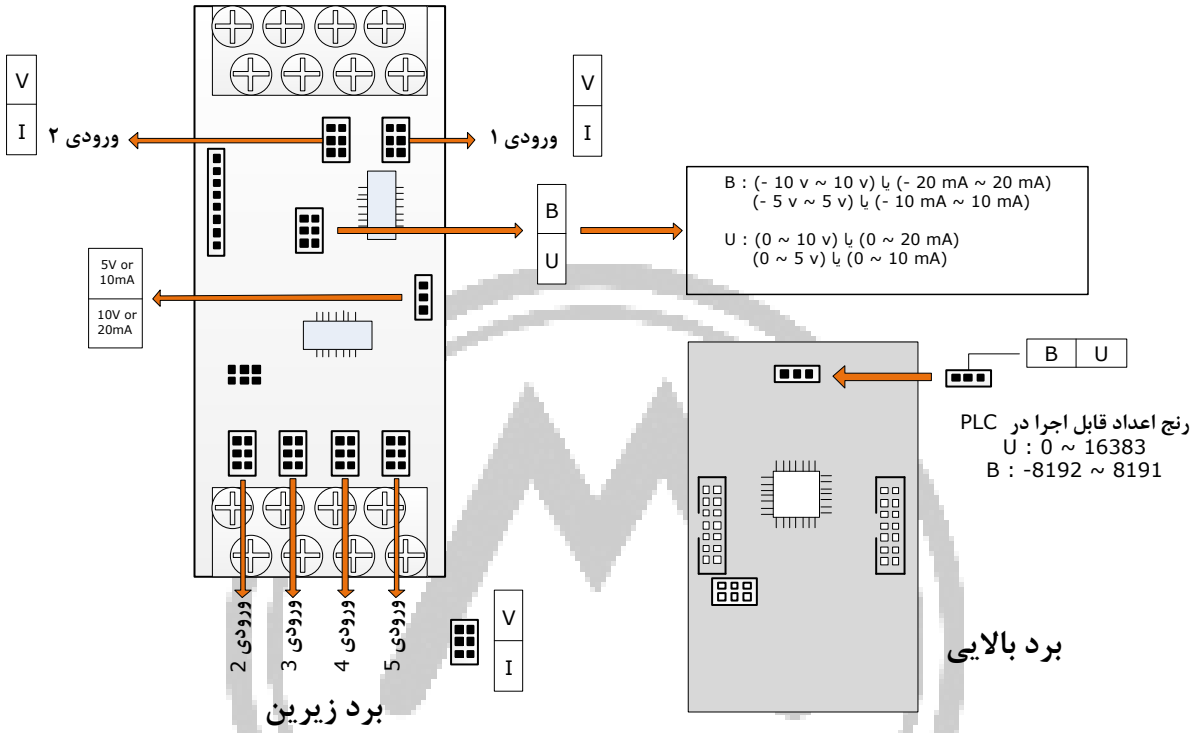
ریجیسترهای مربوط به کانالهای ورودی آنالوگ آدرسهای R3840~R3903 می باشند (64 کانال مجزا) .

ریجیسترهای مربوط به کانالهای خروجی آنالوگ آدرسهای R3904~R3967 می باشند (64 کانال مجزا).

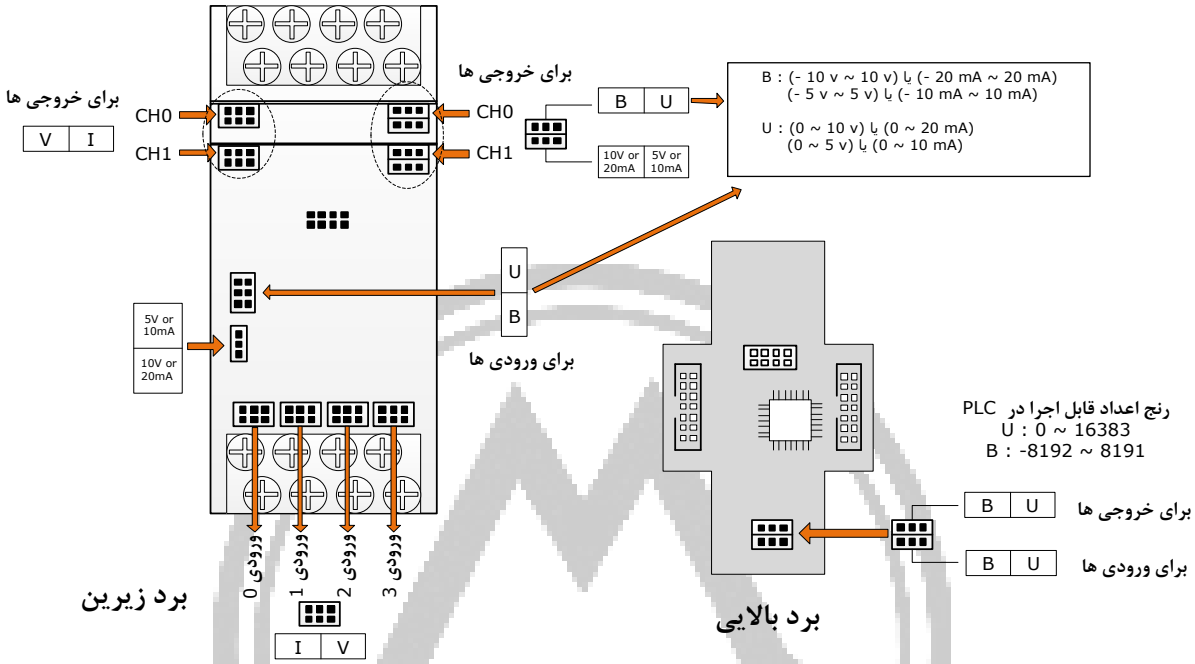
و ، (8191 ~ 8192 -)Bipolar تنظیم شده هستند .در صورت نیاز به تغییر این تنظیمات، پس از باز نمودن پوشش روی این ماژول، می توان مد کاری را بصورت سخت افزاری و با تنظیم کردن جامپرها تنظیم کرد .



تنظیم جامپرهای مربوط به ماژول آنالوگ FBs-4DA/2DA



تنظیم جامپرهای مربوط به ماژول آنالوگ FBs-6AD



تنظیم جامپرهای مربوط به ماژول آنالوگ FBs-4A2D

بنام خداوند بخشنده ومهربان

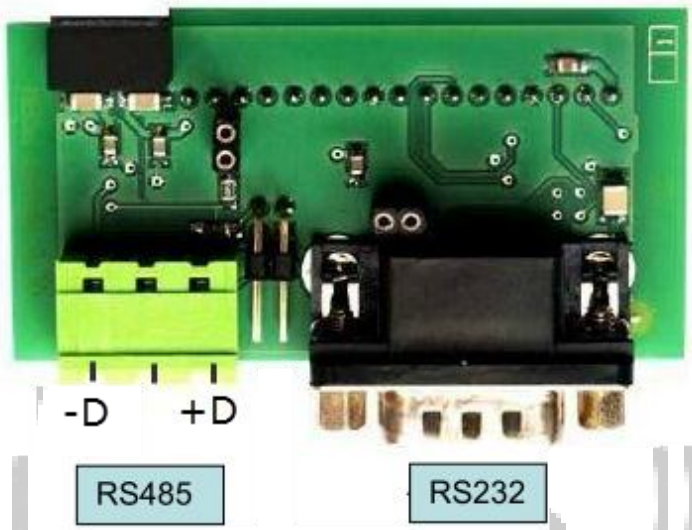
عنوان مدرک :	برنامه نویسی FATEK PLC
توضیحات :	ارتباط Modbus بین FATEK PLC و Comap InteliDrive
تعداد صفحه :	3
شماره ویرایش :	0
ویرایش کننده :	ا.رضایی
تاریخ ویرایش :	1391.9.10

تنظیمات روی کنترلر ژنراتور :

پورت Com1 دارای استاندارد RS-232 و پورت Com2 دارای استاندارد RS-485 می باشد.

چون با PLC و از طریق RS-485 می خواهیم ارتباط برقرار کنیم ، باید COM2 را در حالت MODBUS قرار دهیم.

شماره control address را عدد 1 قرار می دهیم.

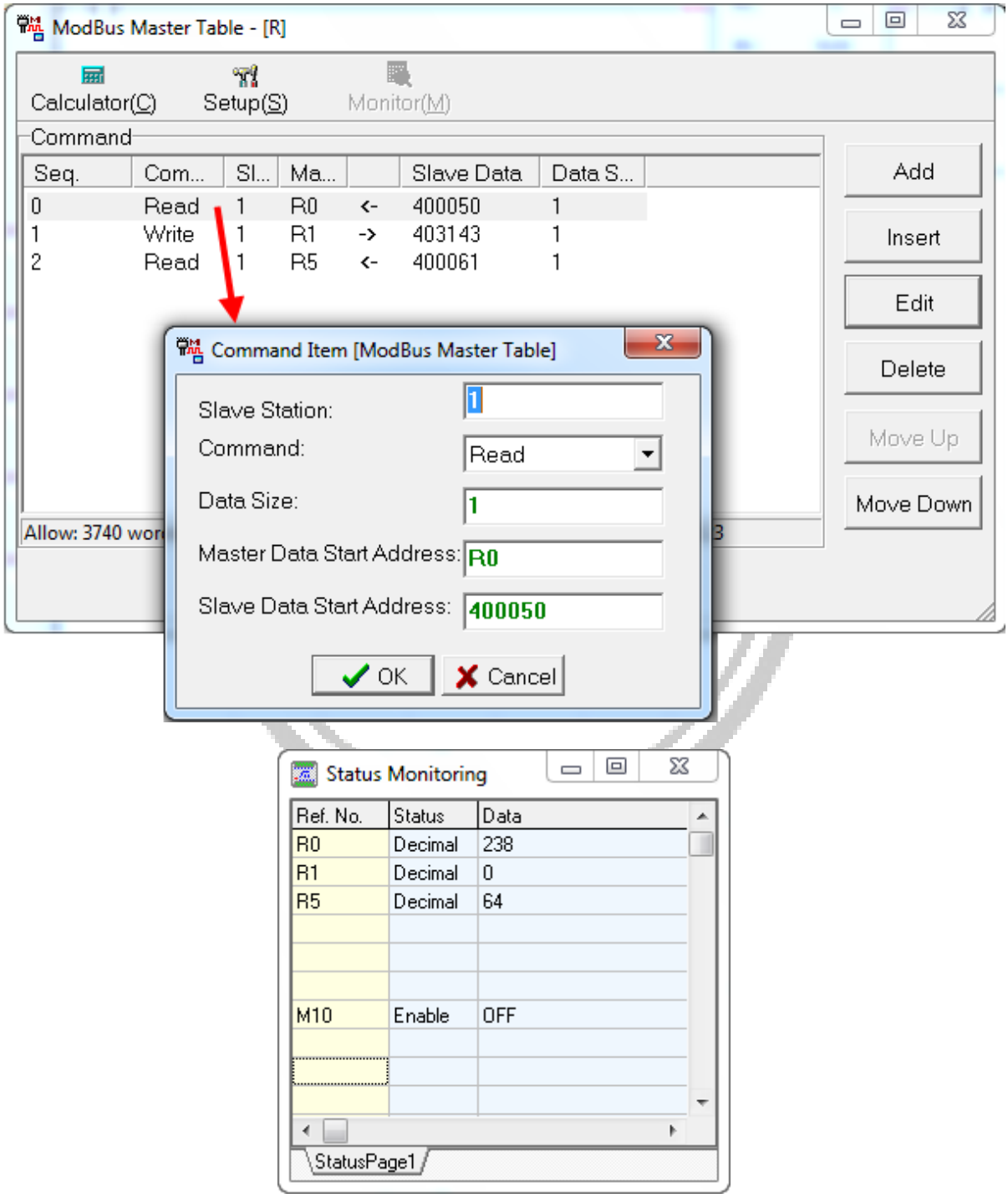


مقادیر Baude rate – Data bit – Stop bit – Parity مربوط به PLC و COMAP باید با هم برابر باشند.

برنامه PLC :

برای مثال ریجیسترهای کنترلر ژنراتور جدول زیر را می خواهیم بوسیله PLC تغییر دهیم.

Register(s)	Com.Obj.	Name	Dim	Type	Len	Dec	Group
40050	8213	Ubat	V	Integer	2	1	Analog CU
40061	8235	BIN		Binary#1	2	-	Binary I/O
43143	8315	ControllerMode		List#6	1	-	Basic settings



بنام خداوند بخشنده ومهربان

عنوان مدرک :	برنامه نویسی PLC FATEK
توضیحات :	نحوه ارتباط PLC FATEK و نرم افزار WINCC
تعداد صفحه :	8
شماره ویرایش :	1
ویرایش کننده :	ا.رضایی
تاریخ ویرایش :	1393.02.27

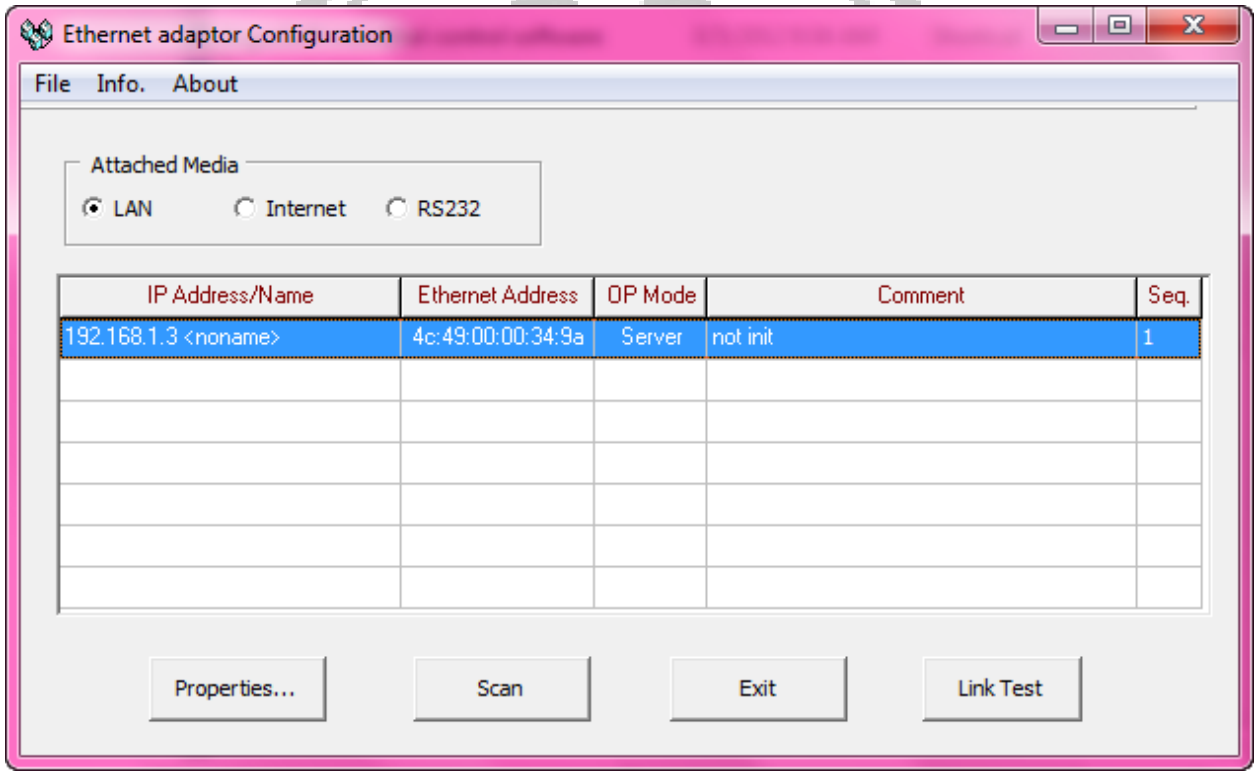
1- این ارتباط با پروتکل MODBUS TCP/IP در نرم افزار wincc و پروتکل MODBUS RTU SLAVE در PLC FATEK انجام می شود .

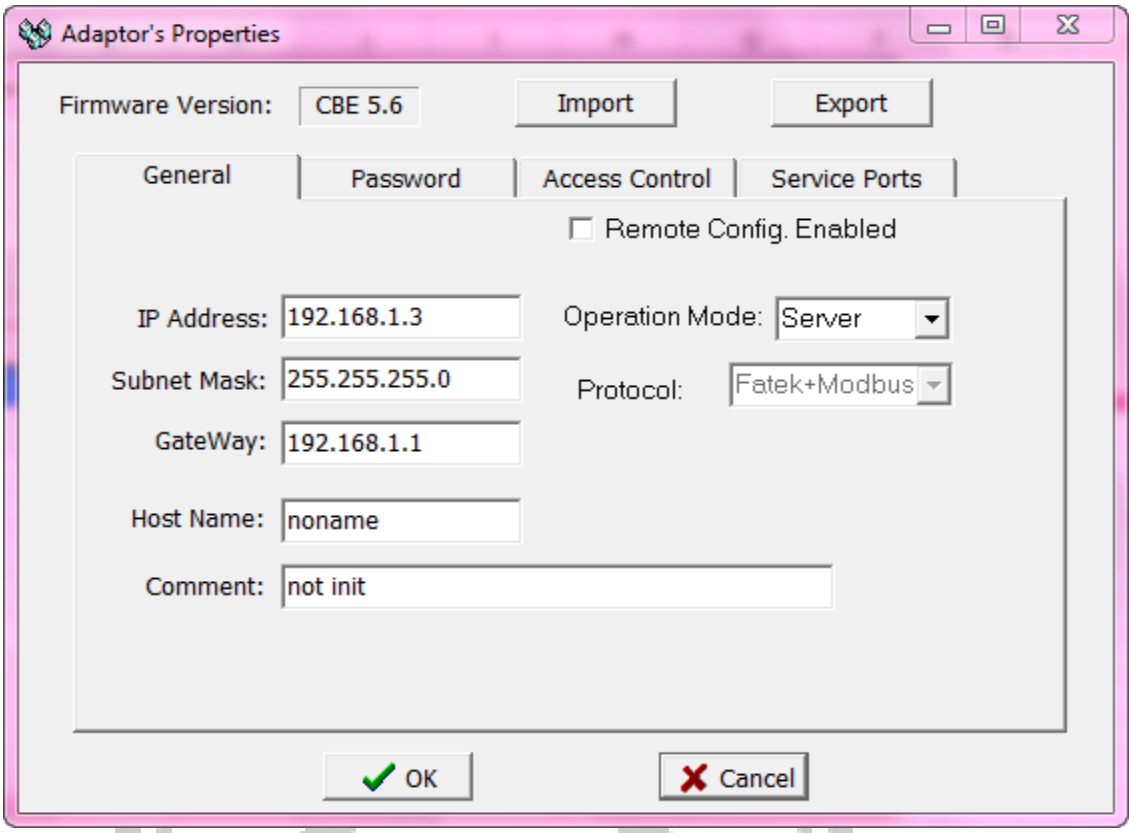
2- نرم افزار WinCC فقط از طریق پورت 2H و بوسیله ماژول FBs-CBEH یا FATEK FBs-CBE می تواند به PLC کانکت شود

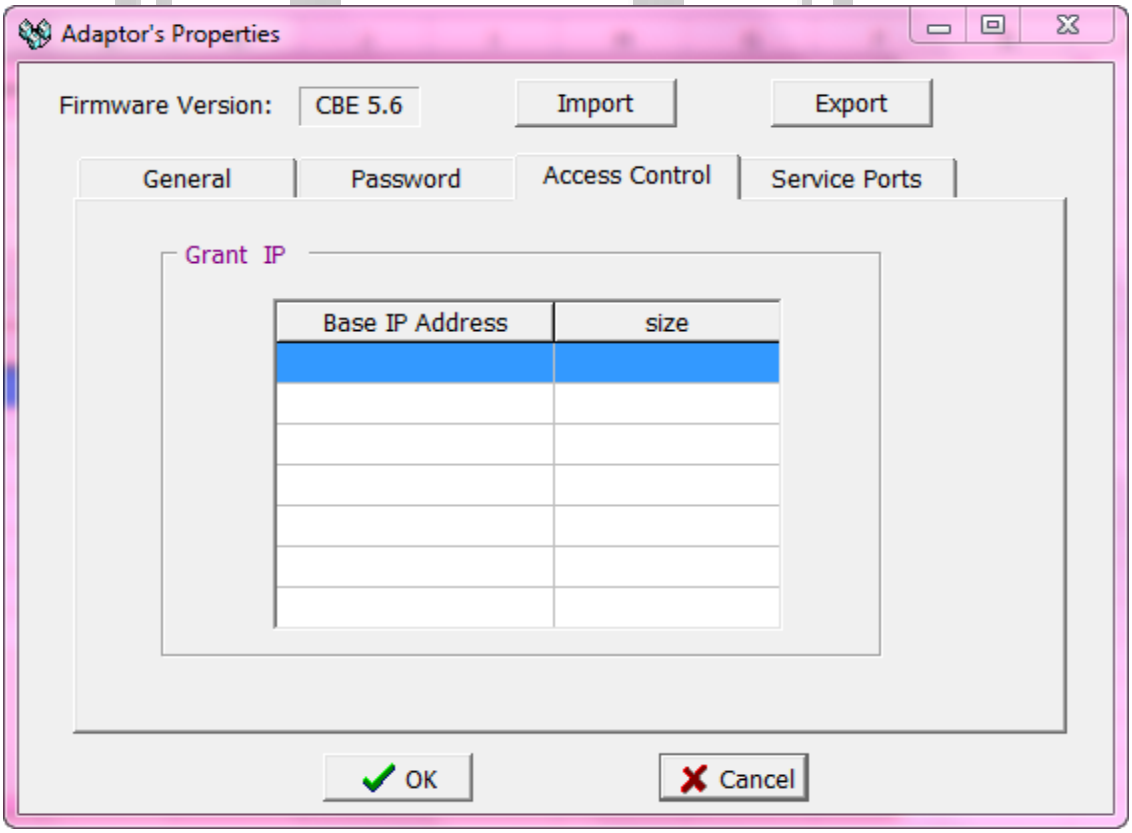
3- تنظیمات در 3 نرم افزار باید انجام شود : WINCC , Fatek Ethernet Module Configuration tool , Winproladder

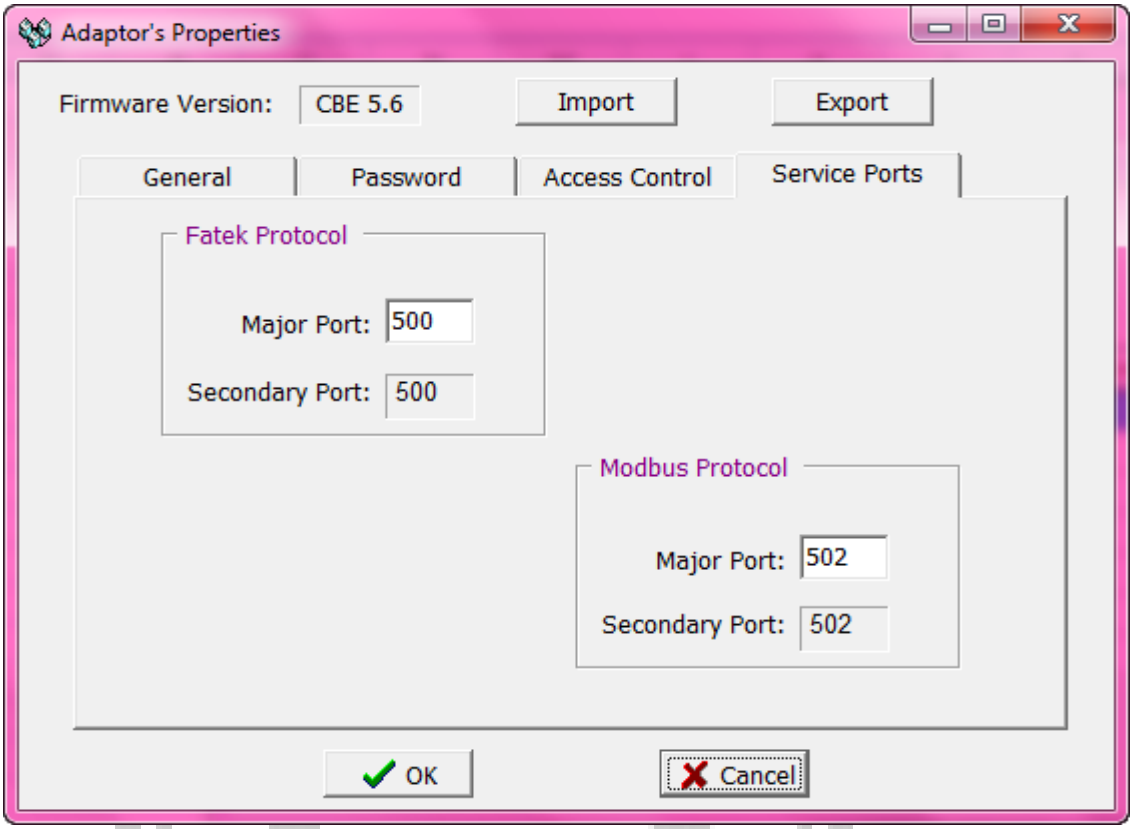
4- با تست های انجام شده ، رجیسترهای 32 بیتی قابل خواندن یا نوشتن نبودند.

تنظیمات در نرم افزار Fatek Ethernet Module Configuration tool

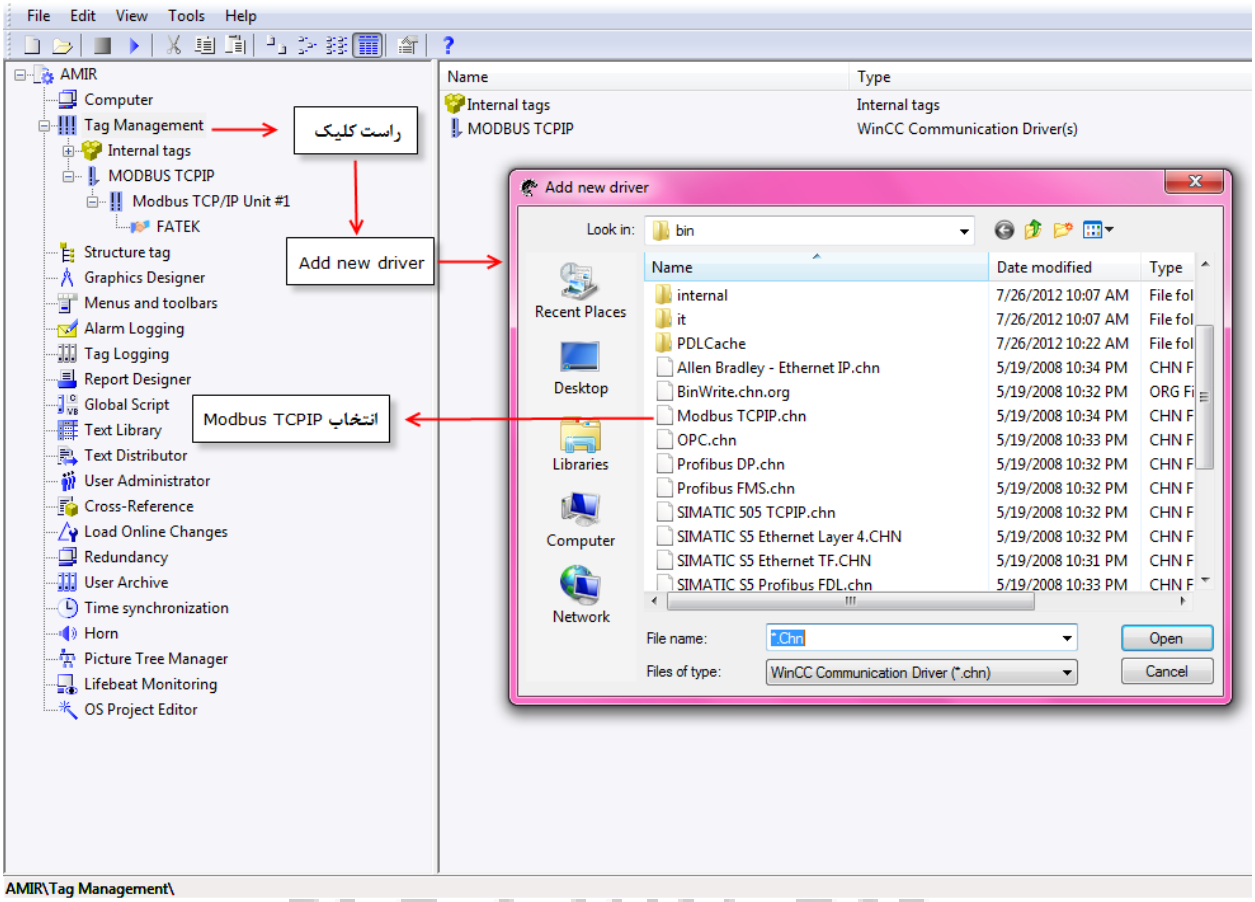


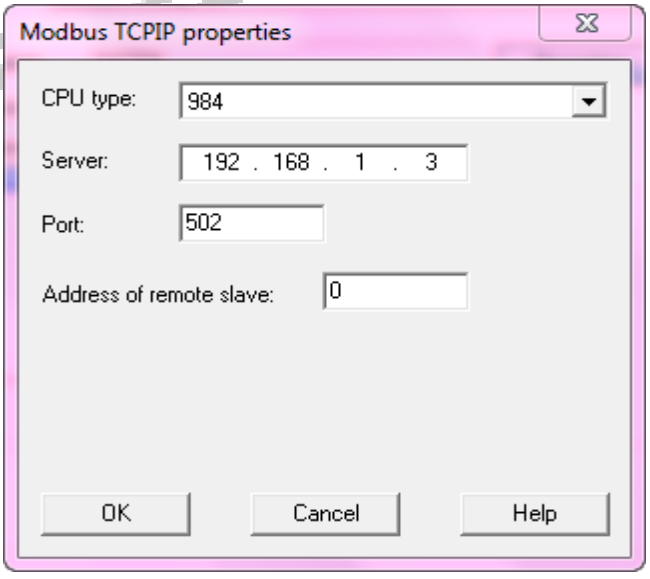

Adaptor's Properties
Firmware Version:
CBE 5.6
Import
Export
General
Password
Access Control
Service Ports
Remote Config. Enabled
IP Address:
192.168.1.3
Operation Mode:
Server
Subnet Mask:
255.255.255.0
Protocol:
Fatek+Modbus
GateWay:
192.168.1.1
Host Name:
noname
Comment:
not init
OK
Cancel

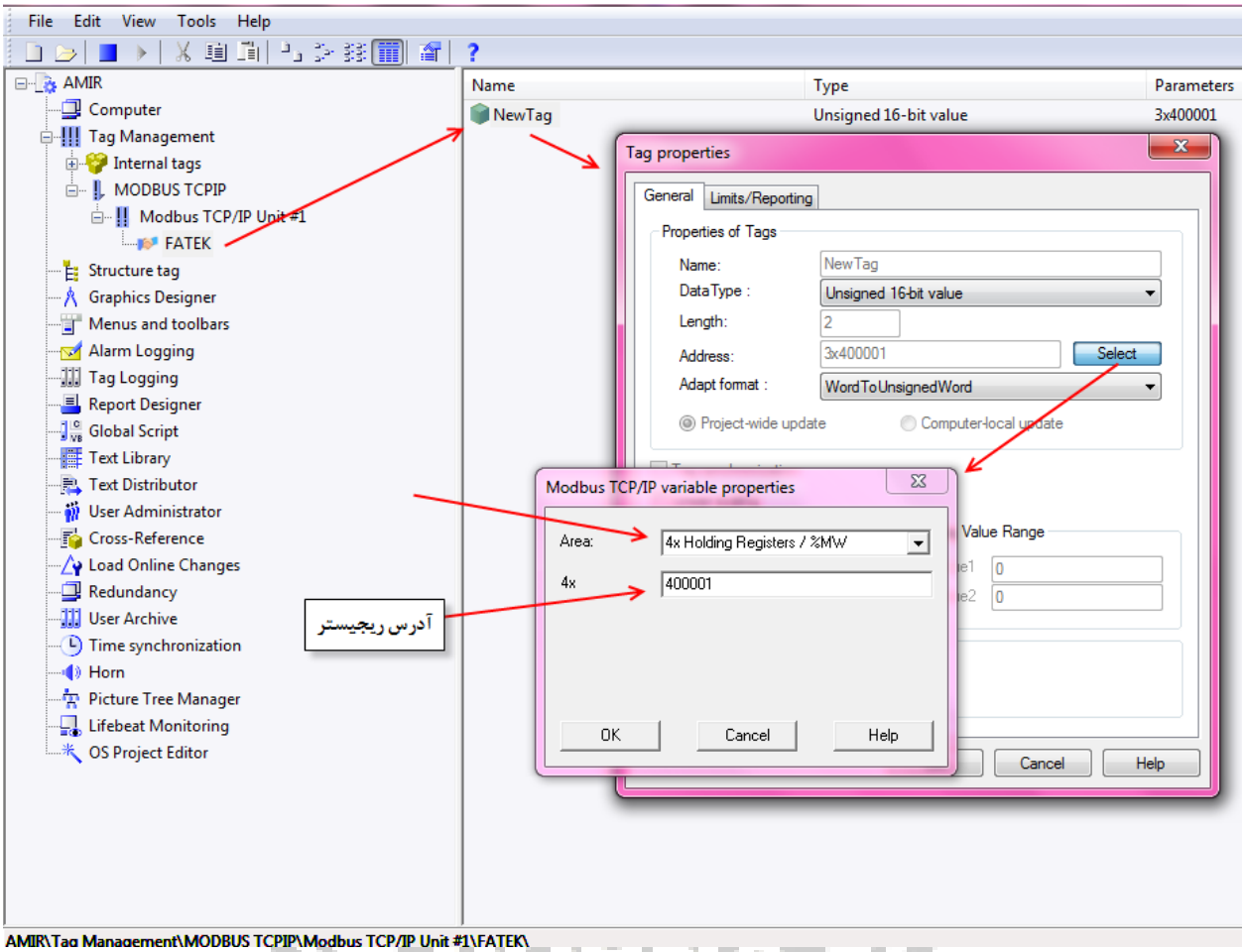

Adaptor's Properties
Firmware Version:
CBE 5.6
Import
Export
General
Password
Access Control
Service Ports
Grant IP
Base IP Address
size
OK
Cancel


Adaptor's Properties
Firmware Version:
CBE 5.6
Import
Export
General
Password
Access Control
Service Ports
Fatek Protocol
Major Port:
500
Secondary Port:
500
Modbus Protocol
Major Port:
502
Secondary Port:
502
OK
Cancel

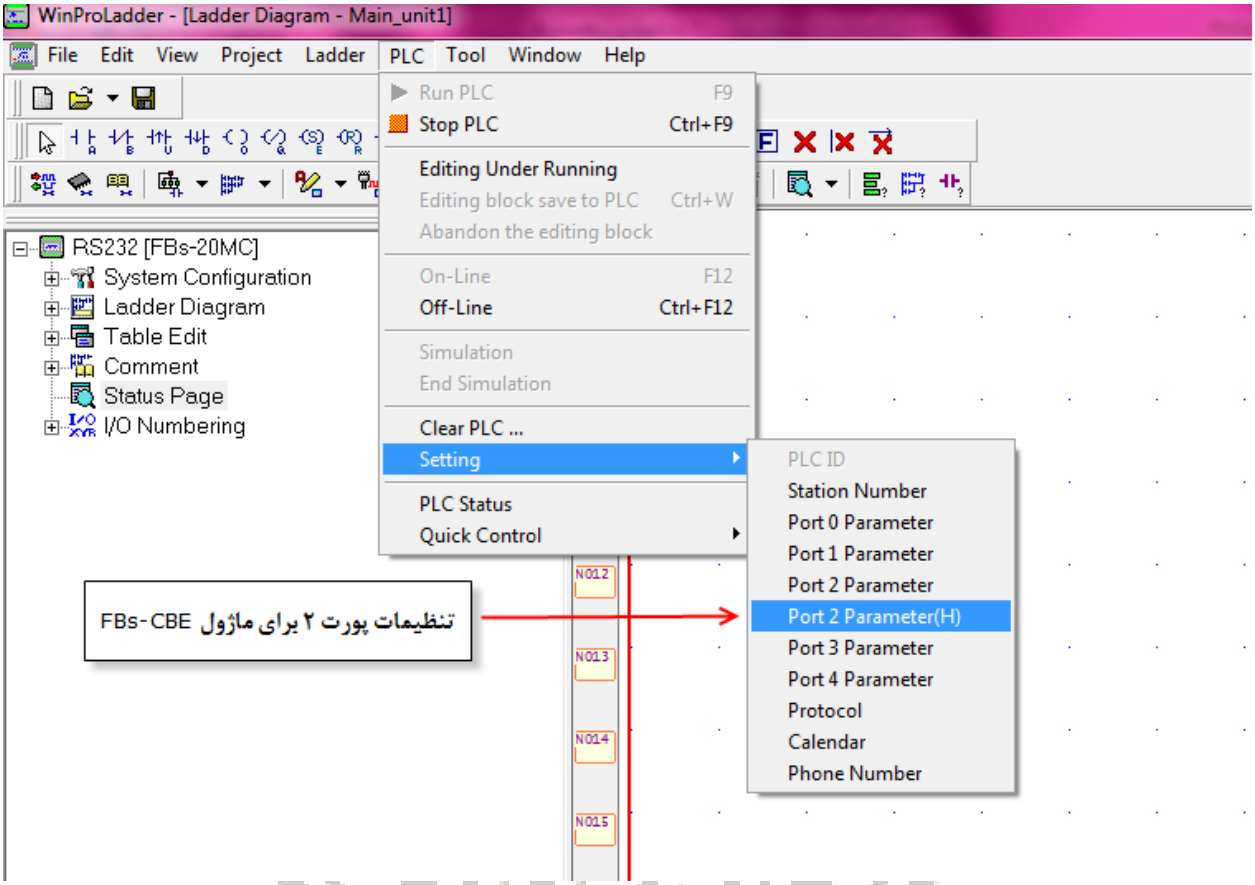
تنظیمات در نرم افزار WINCC

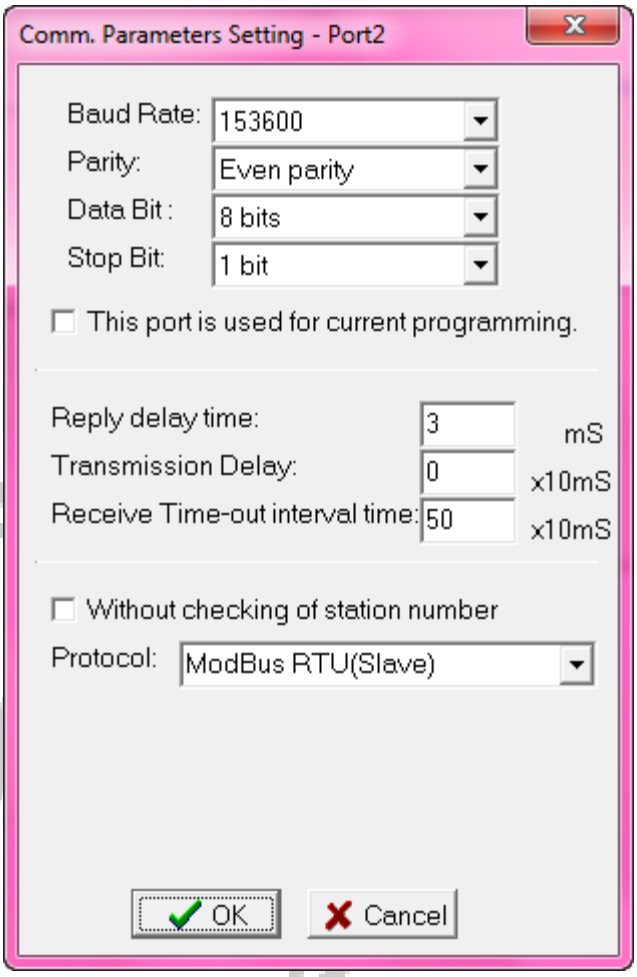


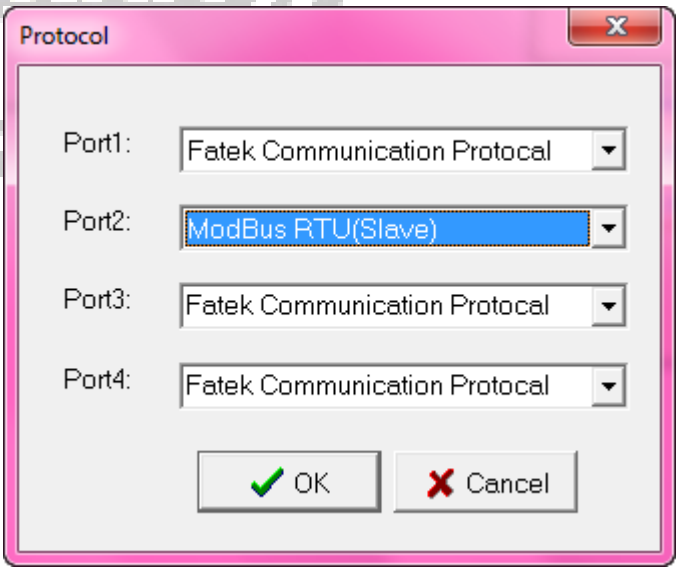



File Edit View Tools Help
AMIR
Computer
Tag Management
Internal tags
MODBUS TCPIP
Modbus TCP/IP Unit #1
FATEK
Structure tag
Graphics Designer
Menus and toolbars
Alarm Logging
Tag Logging
Report Designer
Global Script
Text Library
Text Distributor
User Administrator
Cross-Reference
Load Online Changes
Redundancy
User Archive
Time synchronization
Horn
Picture Tree Manager
Lifebeat Monitoring
OS Project Editor
Name
Type
Parameters
NewTag
Unsigned 16-bit value
3x400001
Tag properties
General
Limits/Reporting
Properties of Tags
Name:
NewTag
DataType :
Unsigned 16-bit value
Length:
2
Address:
3x400001
Select
Adapt format :
WordToUnsignedWord
Project-wide update
Computer-local update
Modbus TCP/IP variable properties
Area:
4x Holding Registers / %MW
4x
400001
OK
Cancel
Help
Value Range
Cancel
Help
آدرس ریجیستر
AMIR\Tag Management\MODBUS TCPIP\Modbus TCP/IP Unit #1\FATEK\

تنظیمات در نرم افزار Winproladder




Comm. Parameters Setting - Port2
Baud Rate: 153600
Parity: Even parity
Data Bit : 8 bits
Stop Bit: 1 bit
This port is used for current programming.
Reply delay time: 3 mS
Transmission Delay: 0 x10mS
Receive Time-out interval time: 50 x10mS
Without checking of station number
Protocol: ModBus RTU(Slave)
OK
Cancel


Protocol
Port1: Fatek Communication Protocal
Port2: ModBus RTU(Slave)
Port3: Fatek Communication Protocal
Port4: Fatek Communication Protocal
OK
Cancel

بنام خداوند بخشنده ومهربان

عنوان مدرک :	برنامه نویسی PLC FATEK
توضیحات :	Configuration of FBs-1LC , 2LC
تعداد صفحه :	5
شماره ویرایش :	3
ویرایش کننده :	ا.رضایی
تاریخ ویرایش :	1393.07.05

FBs-1LC

Specifications

Total Channels - One channel
Resolution- 16 bit (include signed bit)
I/O Points Occupied - 1 RI(Input Register) and 8 DO
Conversion Rate- 5/10/20/25 Hz selectable
Non-Linearity- 0.01% F.S. (@25℃)
Zero Drift- 0.2 μV/℃
Gain Drift- 10 ppm/℃
Excitation Voltage – 5V with 250Ω load
Sensitivity - 2mV/V, 5mV/V, 10mV/V, 20mV/V
Software Filter- Moving average
Average Samples- 1~8 configurable
Isolation- Transformer(Power) and photo-coupler(Signal)
Indicator(s) - 5V PWR LED
Supply Power- 24V-15%/+20%, 2VA
Internal Power Consumption- 5V, 100mA
Operating Temperature- 0 ~ 60 ℃
Storage Temperature- -20 ~ 80 ℃

FBs-2LC

Specifications

Total Channels - Two channels
Resolution- 16 bit (include signed bit)
I/O Points Occupied - 1 RI(Input Register) and 8 DO
Conversion Rate- 0.6/1.2/2.5/3 Hz
Non-Linearity- 0.01% F.S. (@25℃)
Zero Drift- 0.2 μV/℃
Gain Drift- 10 ppm/℃
Excitation Voltage – 5V with 250Ω load
Sensitivity - 2mV/V, 5mV/V, 10mV/V, 20mV/V
Software Filter- Moving average
Average Samples- 1~8 configurable
Isolation- Transformer(Power) and photo-coupler(Signal)
Indicator(s) - 5V PWR LED
Supply Power- 24V-15%/+20%, 2VA
Internal Power Consumption- 5V, 100mA
Operating Temperature- 0 ~ 60 ℃
Storage Temperature- -20 ~ 80 ℃

کارتهای **FBs-1LC , 2LC** مقدار مربوط به لودسل را در ورودی آنالوگ **(R3840~R3903)** نشان می دهند .

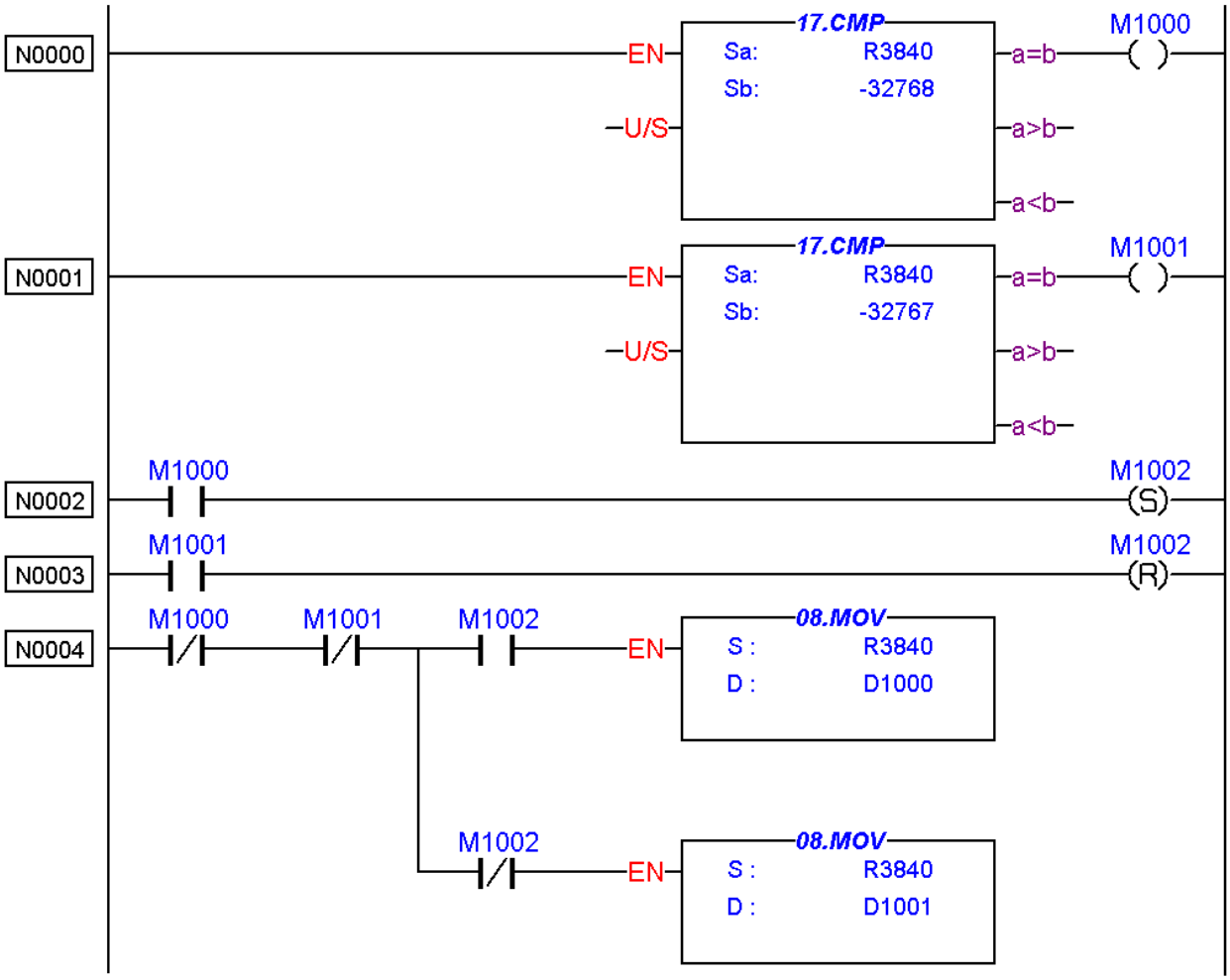
در کارت **2LC** ریجیستر خروجی لودسل (ورودی آنالوگ) برای هر دو کانال فقط یک ریجیستر 16 بیتی می باشد (از 32768 تا 32768-) . این ریجیستر دائما در بین مقادیر زیر متغیر می باشد.

رجیستر ورودی آنالوگ	شرح عدد
-32760	سنسور قطع شده است
-32768	چند لحظه بعد مقدار کانال **0** لودسل نشان داده می شود
-32767	چند لحظه بعد مقدار کانال **1** لودسل نشان داده می شود
اعداد دیگر	مقدار کانال لودسل

برای استفاده از ماژول **FBs-2LC** در نرم افزار **WinProllader** ورژن های **3.22** و قبل تر از آن ، باید برنامه زیر را نوشت.

برنامه ای برای خواندن از کارت لودسل دوکاناله از روی ریجیستر **R3840**

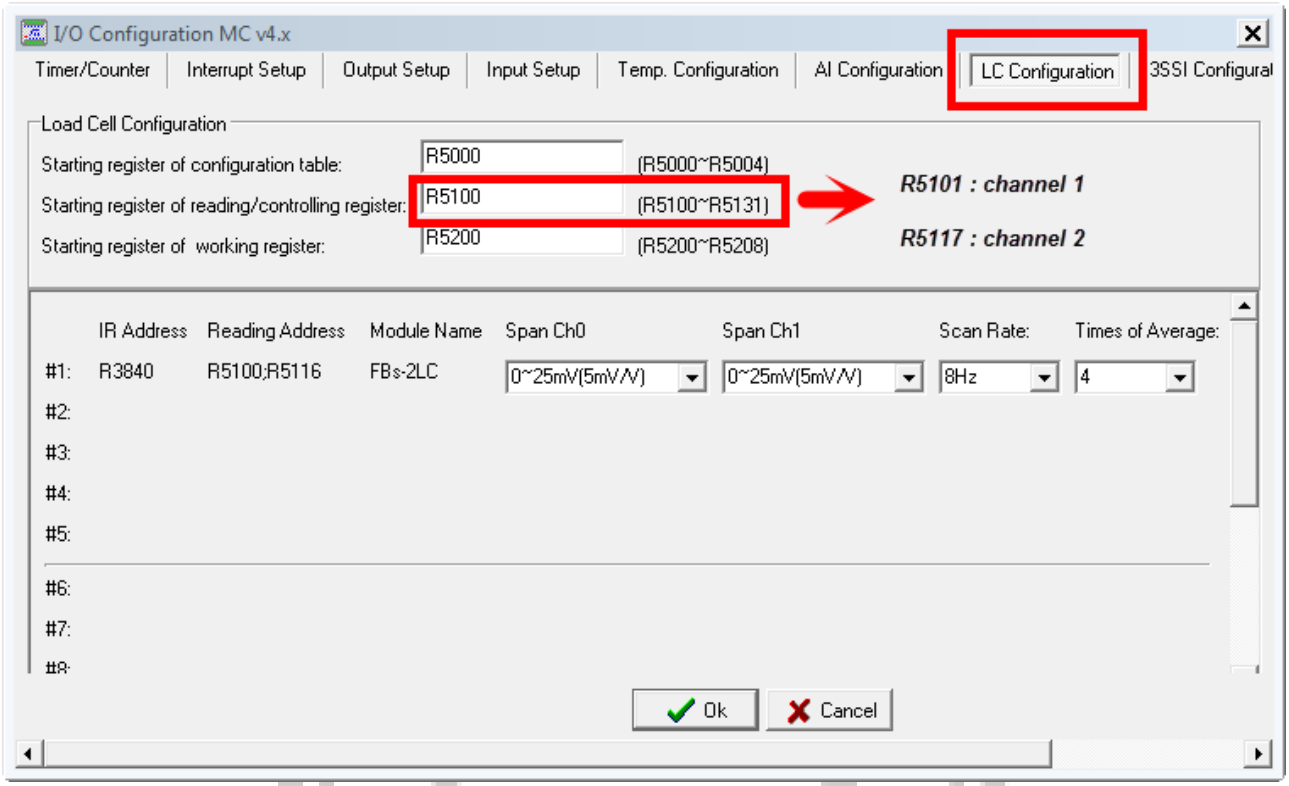
Printed Item: Ladder Diagram - Main_unit1



D1000 مربوط به کانال "صفر" کارت **2LC** می باشد .

D1001 مربوط به کانال "یک" کارت **2LC** می باشد .

در نرم افزار WinProllader ورژن های 3.23 و بعد از آن، با تنظیم جدول در منوی I/O configuration می توان هر دو کانال لودسل را خواند.



FBs-2LC

Signal	Name	Function Description	
Y_{s+1},Y_{s+0}	Span CH #0	00	0~10mV(2mV/V)
		01	0~25mV(5mV/V)
		10	0~50mV(10mV/V)
		11	0~100mV(20mV/V)
Y_{s+1},Y_{s+0}	Span CH #1	00	0~10mV(2mV/V)
		01	0~25mV(5mV/V)
		10	0~50mV(10mV/V)
		11	0~100mV(20mV/V)
Y_{s+5},Y_{s+4}	Conversion Rate	00	0.6 Hz
		01	1.2 Hz
		10	2.5 Hz
		11	3 Hz
Y_{s+7},Y_{s+6}	Average Count	00	No Average
		01	2 Samples
		10	4 Samples
		11	8 Samples

FBs-1LC

Signal	Name	Function Description	
Y_s+1,Y_s+0	SPAN	00	0~10mV(2mV/V)
		01	0~25mV(5mV/V)
		10	0~50mV(10mV/V)
		11	0~100mV(20mV/V)
Y_s+3,Y_s+2	RESERVED	Reserved	
Y_s+5,Y_s+4	CONVSERSION RATE	00	5Hz
		01	10Hz
		10	20Hz
		11	25Hz
Y_s+7,Y_s+6	AVERAGE COUNT	00	No Average
		01	2 Samples
		10	4 Samples
		11	8 Samples

بنام خداوند بخشنده ومهربان

عنوان مدرک :	برنامه نویسی FATEK PLC
توضیحات :	نحوه راه اندازی ماژول های آنالوگ FATEK B1
تعداد صفحه :	2
شماره ویرایش :	1
ویرایش کننده :	ا.رضایی
تاریخ ویرایش :	1391.12.5

نحوه راه اندازی ماژول های آنالوگ FATEK B1

CPU های سری B1 نمی توانند مستقیما کارت آنالوگ متصل را شناسایی کنند برای معرفی کارت آنالوگ به CPU باید یکی از مقادیر زیر را در ریجیستر D4071 قرار داد:

ریجیسترهای مربوط به ورودی آنالوگ : D4072 ~ D4075
ریجیسترهای مربوط به خروجی آنالوگ : D4077 ~ D4076
به ازای هر کدام از ماژولها ، مقدار مربوط به آن ماژول را در D4071قرار دهیم
B1-L2A1D, B1-L2A2D : D4071=23042 OR 23047
B1-L2DA : D4071=23043
B1-L4AD : D4071=23044

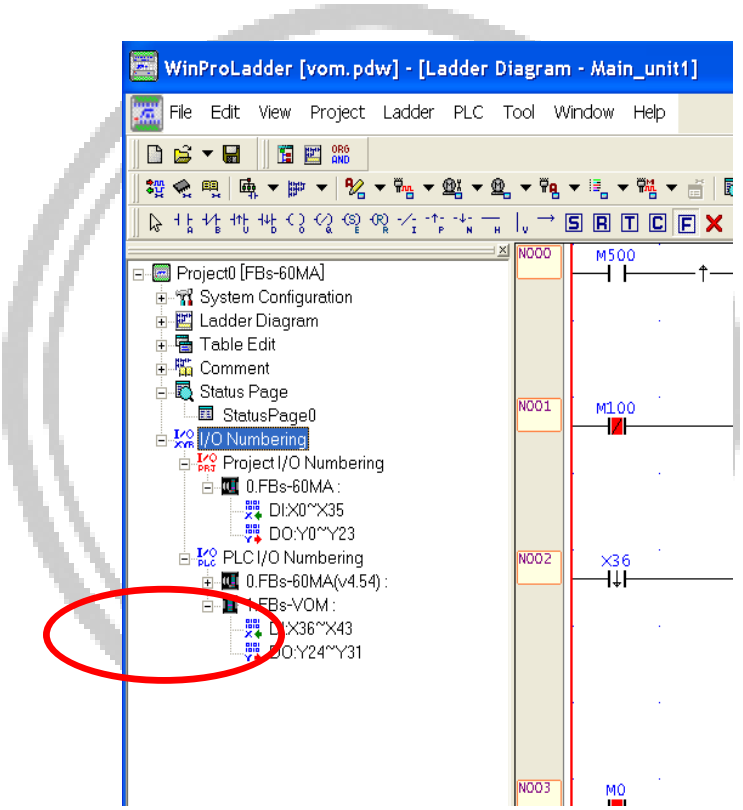
بنام خداوند بخشنده ومهربان

عنوان مدرک :	FATEK PLC FBs VOM
توضیحات :	نحوه راه اندازی کارت FBs-VOM
تعداد صفحه :	13
شماره ویرایش :	1
ویرایش کننده :	ا.رضایی
تاریخ ویرایش :	1392.04.04

FATEK FBs-VOM " Voice Output Module"

FBs-VOM ، ماژولی برای پخش اصوات مختلف از طریق بلند گویی که به آن نصب می شود می باشد. این ماژول قابلیت نصب SD memory card را بر روی خود دارد و حداکثر 245 پیغام را ذخیره می کند. نرم افزار VOM Edit ، پیغام ها با پسوند wav. را به پسوند wrk. تبدیل می کند .

ورودی ها و خروجی های دیجیتال مجازی مرتبط با این ماژول در شاخه ی درختی نشان داده می شوند.



مشخصه	ویژگی ها
حداکثر تعداد پیغام ها	245 پیغام
محل ذخیره	حافظه ی داخلی یا کارت حافظه ی SD
حافظه ی داخلی	1MB به مدت 2 دقیقه
کارت حافظه ی SD	یک کارت 2 GB می تواند تا 4000 دقیقه ، پیغام صوتی در خود ذخیره کند
مشخصات فایل wave مورد استفاده در نرم افزار VOM Edit	Mono 8 bit 8KHz
کنترل پخش صوت ها	از طریق PLC یا به صورت دستی برای انجام تست (از طریق کلیدهای تعبیه شده به روی ماژول)
کنترل ولوم صوت ها(volume)	از طریق PLC تا 10 درجه
اشغال ورودی ها و خروجی ها	8 ورودی دیجیتال و 8 خروجی دیجیتال
نمایشگر وضعیت	3 LEDs

در صورت نبود کارت حافظه ، از حافظه ی داخلی ماژول برای پخش اصوات ، استفاده می شود.

1.پخش صدا

ارتباط PLC و VOM از طریق 8 ورودی دیجیتال مجازی (DI)و 8 خروجی دیجیتال مجازی (DO) می باشد .هر پیغام در ماژول با یک عدد (از 1 تا 245) شناخته می شود که نسبت دادن اعداد به پیغام ها، در نرم افزار VOM Edit صورت می گیرد.

PN = عددی که در 8 خروجی دیجیتال مجازی می توان نوشت از(0 تا 255)،

- **PN=0 ← دستور Stop را صادر کرده اید.**
- **1 ≤ PN ≤ 245 ← دستور پخش یکی از 245 پیغام را صادر کرده اید.**
- Volume #0 ~ Volume #9 ← PN= 246 ~ 255

8 ورودی دیجیتال مجازی برای نمایش وضعیت فعلی پخش به کار می روند.

ورودی	نام	توضیح
X+0	TOGGLE	این بیت ، هرگاه فرمان پخش جدید صادر شود، تغییر می کند
X+1	BUSY	در طول مدت پخش ، این بیت 1 می شود، بعد از پایان پخش 0 می شود
X+2	VOL-CMD	وقتی فرمان صادر شده مربوط به کنترل ولوم باشد، این بیت 1 می شود ، در غیر این صورت 0 می شود
X+3	ERROR	وقتی شماره فرمان صادر شده ، بیشتر از تعداد پیغام های ذخیره شده در حافظه ی داخلی باشد، این بیت 1 می شود
X+4 ~ X+7	VERSION	نسخه ی نرم افزار ماژول را نشان می دهد 1~15

2. نحوه ی بارگذاری محتویات صوتی SD Card به داخل حافظه ی داخلی ماژول

ابتدا PLC را به برق وصل نموده ، سپس SD را داخل ماژول قرار دهید.

☚ فایل ذخیره شده به روی کارت حتما باید با نام و پسوند VOM.wrk ذخیره شود. ایجاد فایل با این پسوند از طریق نرم افزار Vom Edit صورت می گیرد .

اگر می خواهید محتویات کارت SD به روی حافظه ی داخلی ماژول منتقل شود، محتویات صوتی موجود در کارت باید کمتر از 1MB باشد در غیر این صورت انتقال صورت نمی گیرد.

بعد از قرار دادن کارت به روی سوکت ماژول ، اگر محتوای آن کمتر از 1MB باشد، LED2 شروع به چشمک زدن می کند.در این حالت در کمتر از 5 ثانیه ، SW1 (سوئیچ1) را فشار دهید تا وارد مد بارگذاری کارت بر حافظه ی داخلی شوید، آنگاه LED2 سریع تر چشمک می زند. در این حالت SW1 را به مدت حداقل 1.5 ثانیه بفشارید. محتوای کارت به روی حافظه ی داخلی ریخته خواهد شد. در مدتی که محتوای کارت در حال انتقال به روی حافظه ی ماژول است ، LED3 (قرمز رنگ) به صورت ممتد روشن خواهد بود و بعد از اتمام انتقال ، خاموش می شود.

اگر حجم پیغام های صوتی شما ، بیشتر از 1MB است، ماژول پیغام ها را مستقیما از روی کارت پخش کرده و دیگرنمی تواند به روی حافظه ی داخلی انتقال دهد.

3. تست پخش صدا به صورت دستی (از طریق سوئیچ ها)

SW1 را بفشارید تا وارد مد تست شوید.سپس دوباره SW1 یا SW2 را در کمتر از 2 ثانیه فشار دهید تا پیغام پخش شود .

با فشردن SW1 ، پیغام بعدی پخش می شود و با فشردن SW2 پیغام قبلی، فشردن هرکدام از سوئیچ ها در هنگام پخش ، پخش را متوقف می کند(Pause) . هرگاه سوئیچ ها بیشتر از 2 ثانیه فشرده نشوند، ماژول به حالت معمول خود باز می گردد. (وقتی ماژول در مد تست است ، فرامین PLC نادیده گرفته می شود)

نمایشگر های وضعیت

LED1~LED3 **بسته به وضعیتی که در آن قرار دارند ، روشن خواهن شد.**

ردیف	وضعیت	LED1	LED2	LED3	توضیح
s0	Error	چشمک سریع	چشمک سریع	چشمک سریع	هیچ پیغامی برای پخش وجود ندارد
s1	مقداردهی اولیه	OFF	OFF	OFF	
s2	انتظار پخش	چشمک عادی	OFF	OFF	انتظار پخش از حافظه ی ماژول (وقتی SD وجود ندارد)
s3	انتظار پخش	چشمک سریع	OFF	OFF	انتظار پخش از SD
s4	پخش پیغام با فرمان PLC	چشمک عادی	ON	OFF	PLCدر حالت s2 فرمان داده است
s5	پخش پیغام با	چشمک	ON	OFF	PLCدر حالت s3 فرمان داده است

	فرمان PLC	سریع			
s6	انتظار برای پخش دستی	چشمک کند	چشمک عادی	OFF	SW1 در حالت s2 یا s3 فشرده شده است
s7	پخش دستی	چشمک کند	ON	OFF	SW1 یا SW2 در حالت s6 فشرده شده است و پیغام در حال پخش است
s8	تایید شناخت SD	چشمک کند	چشمک عادی	OFF	SD با محتوای قابل پخش از طریق ماژول، به روی آن نصب شده است
s9	ورود به مد بارگذاری	چشمک کند	چشمک سریع	OFF	SW1 در حالت s8 فشرده شده است، آماده بارگذاری SD به روی ماژول
s10	بارگذاری به روی ماژول	چشمک کند	OFF	ON	SW1 در حالت s9 برای حداقل 1.5 ثانیه فشرده شده است

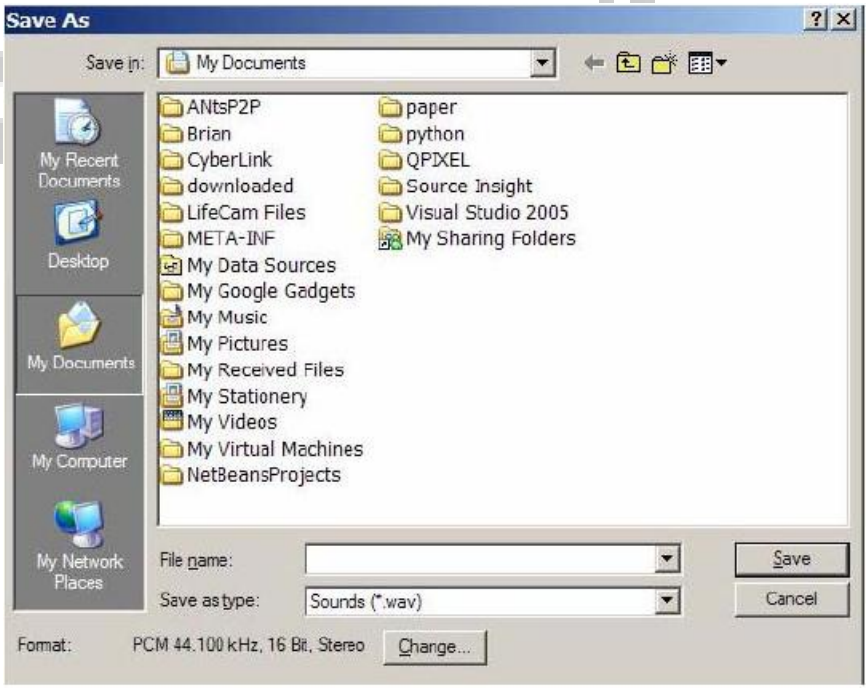
نرم افزار VOM Edit

قالب پیغام های قابل استفاده در این نرم افزار wave **می باشد . برای تبدیل فایل های صوتی مختلف مانند** mp3، wma **و غیره به** wav **به ترتیب زیر عمل کنید.**

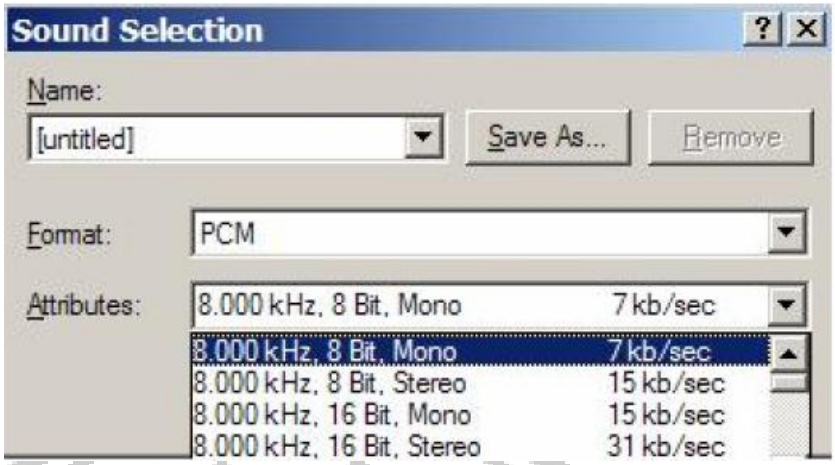
در Windows XP **به روی منوی "start" کلیک کنید ← "program files" ← "Accessories" ← "Entertainment" ← "Sound Recorder"**

از منوی "File" گزینه ی "Open File" را انتخاب نموده و فایل صوتی مورد نظر خود را باز کنید.

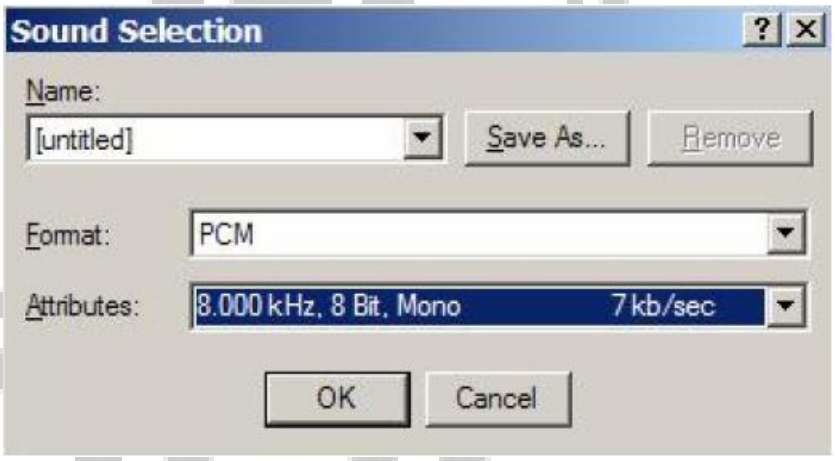
سپس از منوی "File" گزینه ی "Save As" را انتخاب کنید.پنجره ی زیر باز می شود.



در منوی کشویی Save as type قالب wav را انتخاب کنید. سپس به روی Change کلیک کنید، پنجره ی زیر باز می شود.



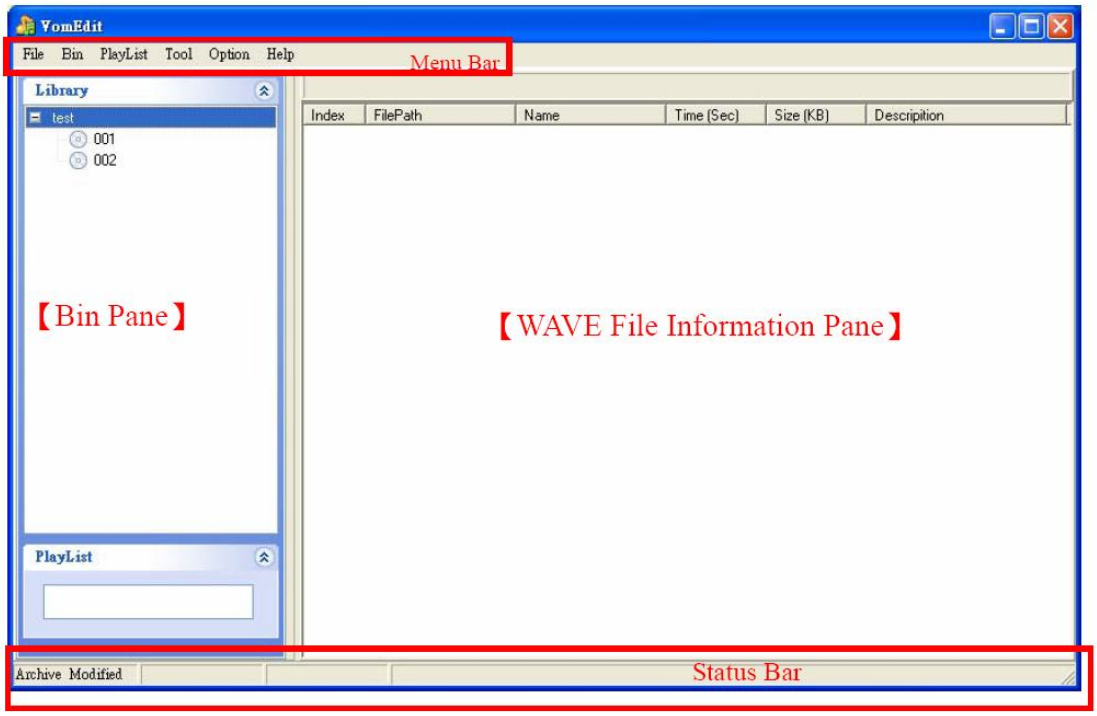
مشخصات را به صورت زیر تنظیم کنید.



OK کرده سپس Save کنید تا پیغام مورد نظر شما با قالب wav ساخته شود.

(در Windows 7 یا در صورت عدم موفقیت در تبدیل فایل مورد نظر به wav از طریق Sound Recorder ، می توانید از نرم افزار Jet Audio یا دیگر تبدیل گر ها استفاده کنید)

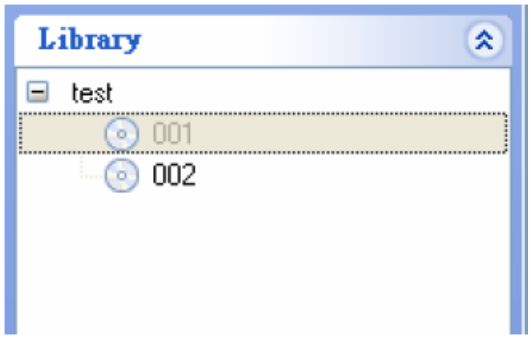
سپس نرم افزار VOM Edit را باز کنید.



1.از منوی File ، گزینه ی Create Archive را انتخاب کرده و یک Archive بسازید که در منوی درختی کنار پنجره ی نرم افزار ظاهر می شود.

2.آرشیو ساخته شده را انتخاب کرده سپس از منوی Bin گزینه ی Create Bin را انتخاب نمایید و

به این ترتیب یک زیر شاخه برای آرشیو خود تهیه نمایید.

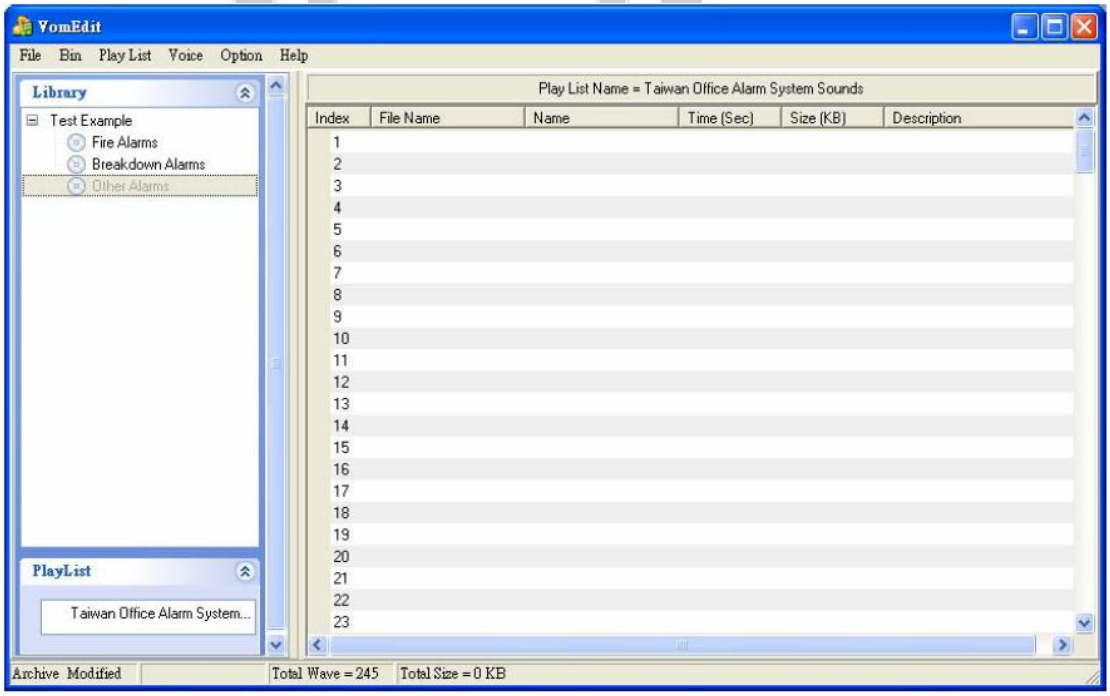


3. Bin **ساخته شده را انتخاب کرده ، سپس از منوی** Voice/Tool **گزینه ی** Add Voice **را انتخاب کرده و فایل های** wav **مورد نظر خود را به** Bin **اضافه کنید.**

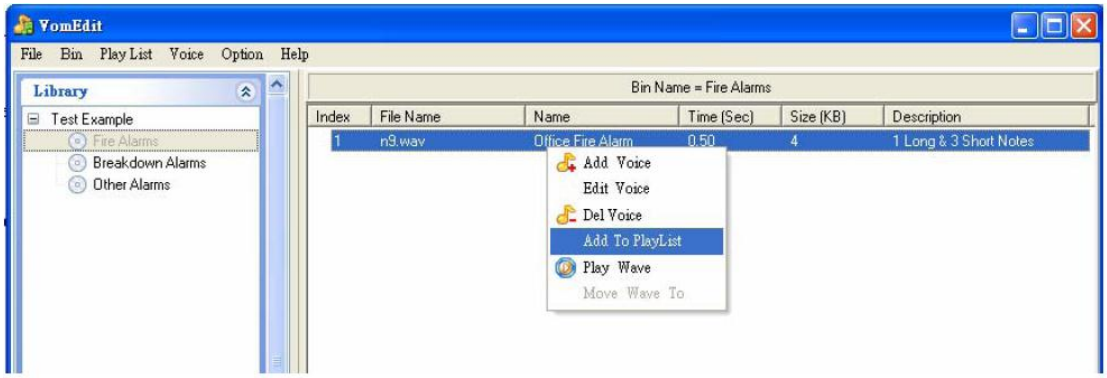
Bin Name = 001

Index	File Name	Name	Time (Sec)	Size (KB)	Description
1	alarm1.wav	Fire alarm	16.31	130	
2	alarm2.wav	Dense Smoke	0.76	6	
3	alarm3.wav	Power Failure	0.86	6	

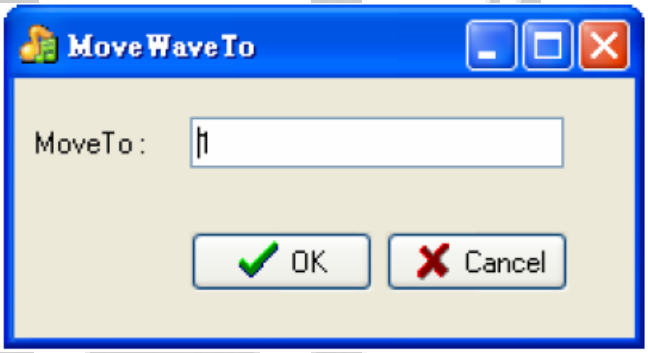
4. از منوی Play List **، گزینه ی** Create Play List **را انتخاب کنید و یک لیست با نام دلخواه خود سازید.**



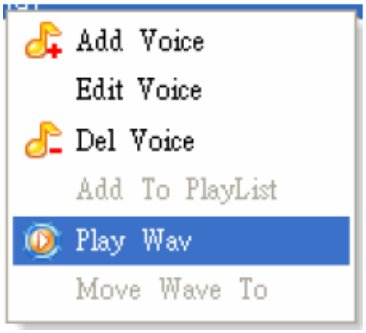
5. به روی فایل های دلخواه **wav** موجود در Bin که قبلا ساخته اید ، کلیک راست کرده و گزینه ی Add To PlayList را انتخاب کنید تا به PlayList اضافه شود.



6. پنجره ای باز می شود که در آن شماره ای که به پیغام مورد نظر در لیست اختصاص داده می شود را تعیین می کنیم.



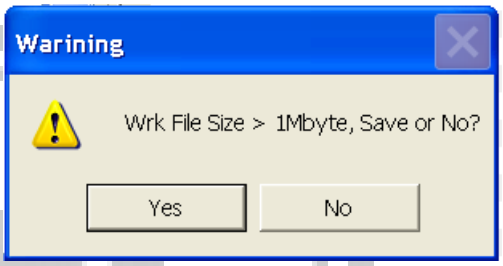
◆ اگر فراموش کرده اید که فایل صوتی wav شما چیست، می توانید به روی آن راست کلیک کرده و گزینه ی Play Wave را انتخاب نمایید تا آن فایل پخش شود.



◆ اگر خواستید موقعیت هر یک از فایل ها در PlayList را تغییر دهید، به روی فایل مورد نظر کلیک راست کرده و گزینه ی Move Wave To را انتخاب کرده و شماره ی مقصد را وارد نمایید.

7. بعد از این که تمام فایل های wav مورد نظر خود را به Play List اضافه کردید، از منوی **PlayList** ، گزینه ی Compile to wrk File را انتخاب نمایید و لیست را با پسوند wrk در مسیر دلخواه خود ذخیره کنید.

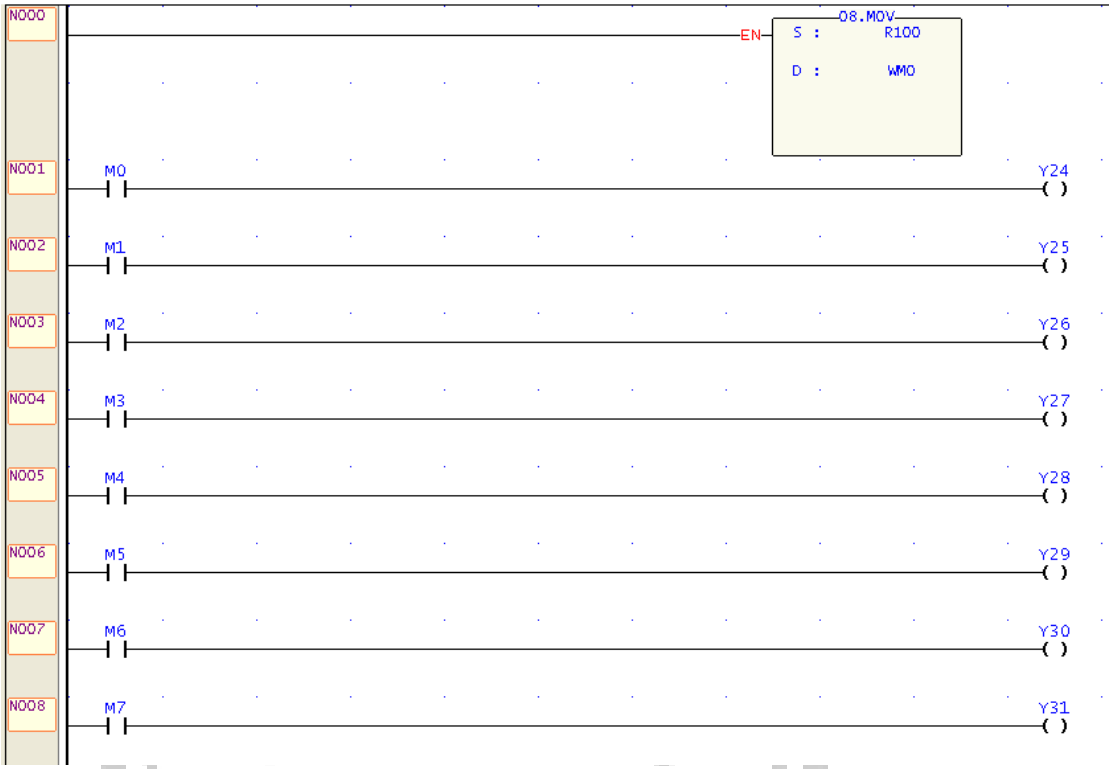
◆ به هنگام Compile ، اگر حجم فایل بیشتر از 1MB بشود، اخطاری داده می شود.



اگر می خواهید محتویات کارت SD به روی حافظه ی داخلی ماژول منتقل شود، محتویات صوتی موجود در کارت باید کمتر از 1MB باشد پس به روی No ، کلیک کنید و از تعداد پیغام های صوتی بکاهید تا کمتر از 1MB شود.

و اگر به تمام پیغام های صوتی خود نیازمندید ، هنگام کار با ماژول VOM ، کارت SD باید به روی آن نصب باشد پس به روی Yes ، کلیک کنید.

مثال برنامه ی PLC :(این مثال برای PLC : FBs-60MA طراحی شده است.)



عدد دستور مورد نظر خود را در R100 بریزید تا دستور اجرا شود.

به عنوان مثال:

اگر عدد 5 را در R100 بریزید ، پیغام پنجم از حافظه ی داخلی یا از روی کارتSD

(در صورت اتصال کارت به روی ماژول) پخش می شود.

اگر عدد 248 را در R100 بریزید ، ولوم صدا به درجه ی دوم کاهش می یابد.

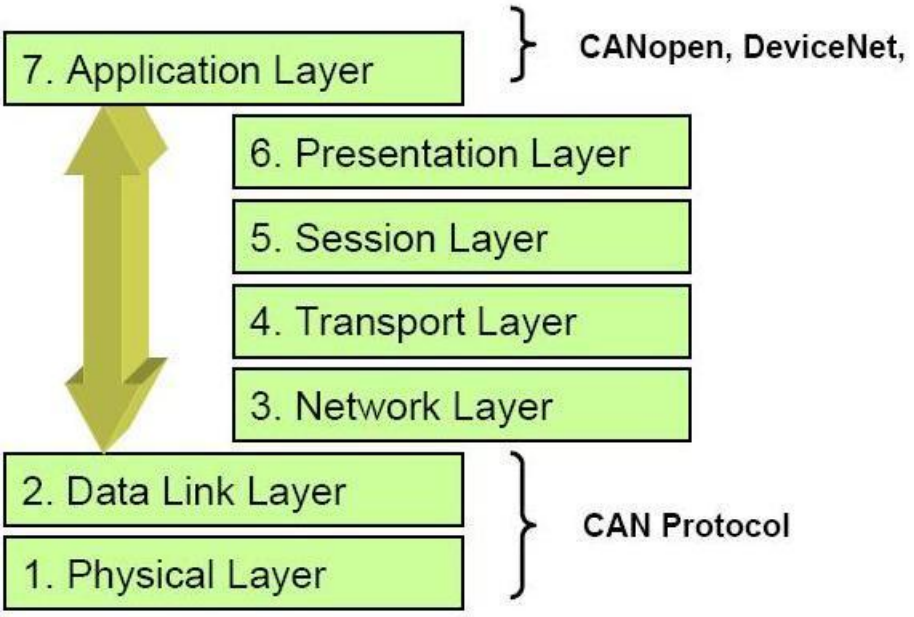
بنام خداوند بخشنده ومهربان

عنوان مدرک :	FATEK PLC FBs CAN
توضیحات :	نحوه راه اندازی کارت FBs-CAN
تعداد صفحه :	15
شماره ویرایش :	1
ویرایش کننده :	ا.رضایی
تاریخ ویرایش :	1393.04.14

✓ پروتوکل CAN (Control Area Network) :

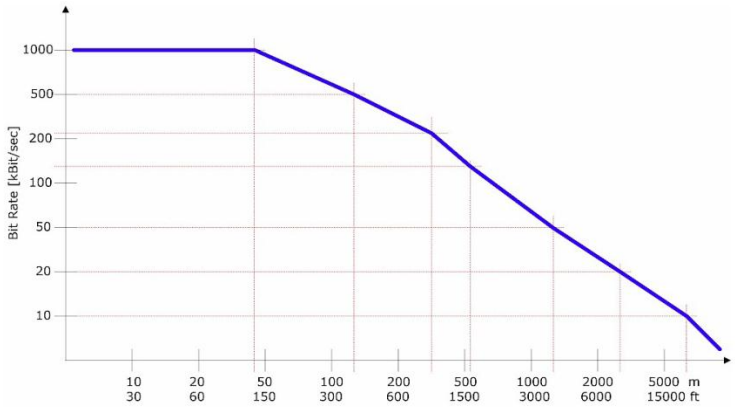
پروتکل ارتباطی CAN، برای اولین بار توسط شرکت بوش برای خودروها طراحی و به بازار عرضه شد و بوسیله خودروسازان اروپایی و سپس امریکایی مورد استفاده قرار گرفت . این پروتوکل در سال 1993 توسط سازمان استاندارد جهانی تحت استاندارد ISO 11898 مورد پذیرش قرار گرفت و پس از آن نیز در اتوماسیون صنعتی ، صنعت خودرو و حمل و نقل جاده ای ، ماشین آلات کشاورزی و راهسازی توسعه یافت. نسخه ای از آن که در اتوماسیون صنعتی توسعه یافته است امروزه به نام CANOpen شناخته می شود و جایگاه ویژه ای دراتوماسیون صنعتی پیدا کرده است .

از دیدگاه مدل OSI و مطابق شکل زیر ، پروتوکل CAN بر اساس لایه های 1و 2 تعریف شده است اما بدلیل محدودیتهای این تعریف در پروتوکل CANOpen لایه هفت نیز اضافه شد .



در حقیقت شبکه CAN ارتباطی سریال و دوسیمه مبتنی بر استاندارد RS-485 است که تا 128حسگر و دستگاه مختلف را به یکدیگر متصل می نماید. این شبکه بر پایه ارسال و دریافت پیامها کارمی کند بدین ترتیب که پیامها توسط حسگرها و ادوات کنترلی در شبکه جاری شده و گیرنده‌ها با دریافت پیام مرتبط با خود عملیات مورد نظر را انجام می‌دهند. کاهش قابل توجه در سیم کشی ، انعطاف پذیری ، قابلیت اطمینان بالا ،اولویت بندی پیامها و امکانات مناسب برای عیب یابی از جمله مزایای این پروتوکل می باشد.

انتقال اطلاعات در این پروتوکل بوسیله یک جفت سیم و بصورت دیفرانسیلی می باشدکه اثر نویز را به حداقل می رساند. طول شبکه تا 1000متر و سرعت انتقال اطلاعات در این پروتوکل می تواند تا 1Mbps (در مسافت 50 متر) باشد .



✓ اصول تبادل اطلاعات در شبکه‌های CAN

در شبکه CAN ، برای ارسال داده‌ها به یک گره یا دستگاه ، آدرس‌دهی مشخصی صورت نمی‌گیرد بلکه محتوای پیام ارسالی به همراه اولویت آن، توسط شناسه‌ای اختصاصی در شبکه مشخص می‌شود. این موضوع هنگامی اهمیت دارد که دستگاه‌های مختلف نیاز به دسترسی همزمان به شبکه داشته باشند. برای ارسال پیام، داده‌ها به همراه شناسه از طریق مبدل‌های CAN برای ارسال آماده شده و به محض آزاد شدن شبکه ، ارسال به تمام گره‌ها انجام می‌شود. دستگاه‌های دیگر (که اکنون در وضعیت گیرنده قرار دارند) پیام را بررسی کرده و در صورتی که به آنها مربوط باشد، آن را می‌پذیرند. افزودن گره‌های جدید به سادگی امکان‌پذیر بوده و نیاز به تغییر سخت‌افزاری چندانی ندارد. CAN پروتوکلی Multi master می باشد به این معنا که هر تجهیزی که به این شبکه متصل شود ، می تواند هم بعنوان فرستنده و هم بعنوان گیرنده اطلاعات کار کند و در زمان های لازم اختیار شبکه را در دست بگیرد مگر اینکه پیامی با اولویت بالاتر از سوی تجهیز دیگری صادر شود .

✓ **ویژگی ها :**

1- پرسرعت تا 1Mbps

2- قابلیت اولویت بندی پیام ها، مناسب برای طراحی سیستم بلادرنگ (پیام با شماره ID پایین تر دارای اولویت بالاتر است)

3- باس 2 سیمه تفاضلی

4- ایمنی بسیار زیاد در برابر نویز

5- قابلیت خطایابی، کنترل و رفع خطا (در مواقع تداخل و...)

6- استاندارد ISO 11898 در لایه Data link و قسمت هایی از لایه فیزیکی

7- برای پیاده سازی، هر Node به موارد زیر نیاز دارد:

❖ **Host processor**

پردازنده تصمیم می گیرد که هر پیام دریافتی چه معنایی دارد و با توجه به پیام چه عملی باید صورت گیرد. همینطور وظیفه تصمیم گیری در مورد پیام هایی که باید ارسال شوند به عهده این قسمت است. کاربر با برنامه نویسی پردازنده، سیستم مورد نظر را کنترل میکند .

8- CAN controller

وظیفه دریافت CAN controller : بیت ها را بصورت سریال از باس دریافت می کند تا یک پیام کامل فراهم شود. سپس پیام به Host processor منتقل می شود.

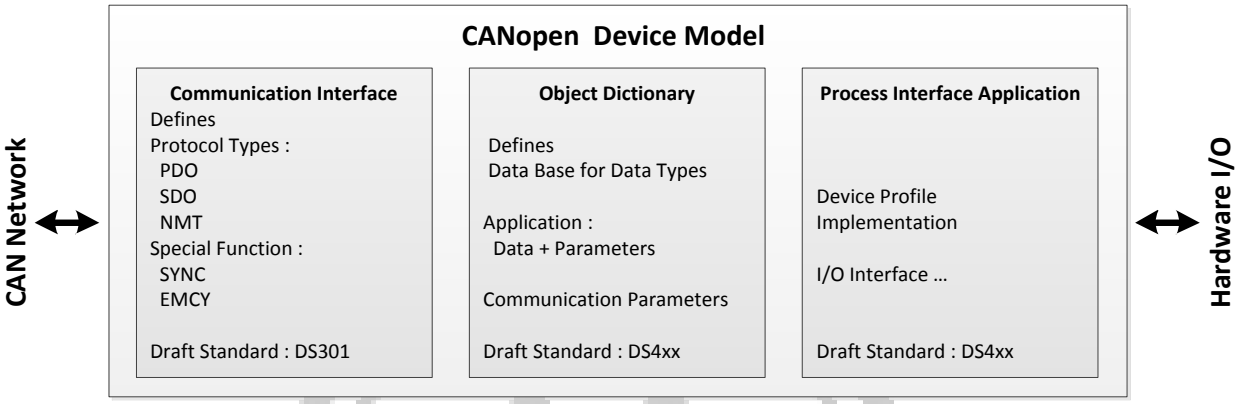
وظیفه ارسال Host processor : بیت های ارسالی را در CAN controller ذخیره می کند تا پیام را بشکل سریال ارسال کند.

❖ Transceiver

وظیفه تبدیل سطوح سیگنال بین باس و CAN controller را به عهده دارد. باس ایجاد شده توسط Transceiver یک باس دو سیمه تفاضلی است. سیگنال های این باس CAN_Hو CAN_L نام دارند.

✓ پروتوکل CANopen

این پروتوکل نسخه ای از CAN است که در اتوماسیون صنعتی توسعه یافته است و از مطابق شکل زیر سه بخش واسط ارتباطی ، Object Dictionary و واسط سخت افزار است.



✓ COB-ID :

هر پیام CANOpen دارای این شناسه است که نشان دهنده شماره Node و نوع پیام می باشد. پیام هایی که از نوع **service** NMT ، SYNC و TIME STAMP می باشند دارای COB-ID ثابت هستند . جدول زیر انواع پیامها و COB-ID آنها را نشان می دهد.

CANopen message types

Message Type	Description	COB-ID
NMT	Network Management (broadcast)	0h
NMT Error Control	Network management error control	701h – 77Fh
BOOT-UP	Boot-Up message	701h – 77Fh
SYNC	Synchronization message (broadcast)	80h
EMERGENCY	Emergency messages	81h - FFh
TIME STAMP	Time stamp (broadcast)	100h
PDO	Process Data Objects	181h - 57Fh
SDO	Service Data Objects	581h – 67Fh

✓ انواع پیامها در CANOpen :

پیامهای PDO و SDO:

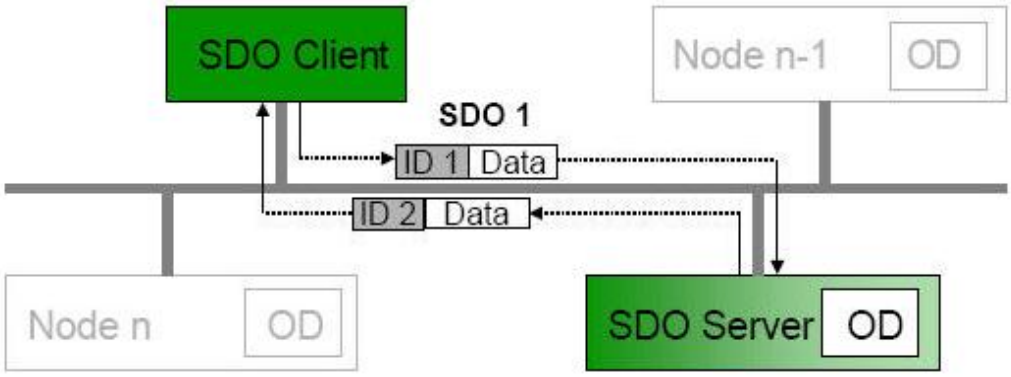
به دو روش می توان پیام های حاوی اطلاعات را از یک Node به Node دیگر فرستاد یا دریافت نمود:

1- SDO(Service Data Object)
2- PDO(Process Data Object)

❖ پیام SDO :

این نوع پیام برای ارسال تنظیمات دستگاه و خواندن یا نوشتن Object Dictionary ها استفاده می شود.

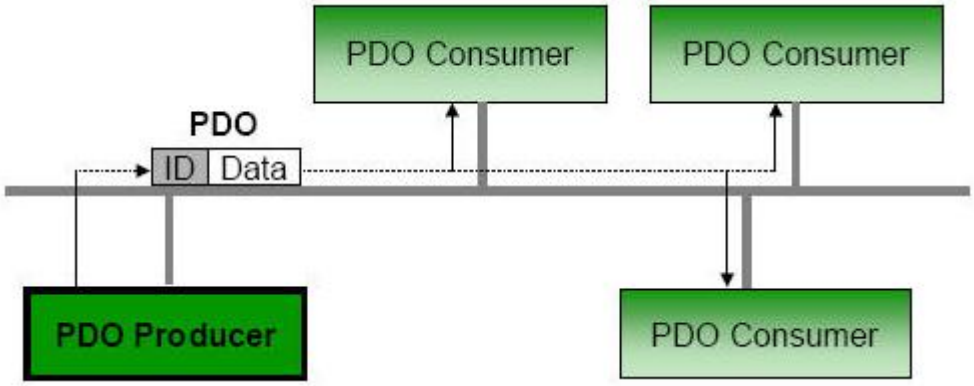
در شکل زیر، چگونگی ارسال و دریافت پیام SDO در شبکه نشان داده شده است:



آدرس رجیسترها در پیام SDO با Index و Subindex مشخص می شوند. رجیستر Index رجیستری 16 بیتی و Subindex رجیستری 8 بیتی می باشد. در شکل زیر فریم SDO مربوط به درخواست اطلاعات آمده است.

❖ **پیام PDO :**

این پیام برای تبادل اطلاعات بصورت Real time استفاده می شود. در فریمهای این پیام می توان 8 بایت را جابجا کرد. این نوع پیام به صورت Producer/Consumer ارسال می شود بدین معنی که پیام فقط توسط یک عضو شبکه ارسال می شود ولی چندین عضو می توانند آن را دریافت نمایند.(شکل زیر)



این پیام به دو دسته RPDO و TPDO تقسیم می شود که در آن RPDO ها برای دریافت اطلاعات از شبکه و TPDO ها برای ارسال اطلاعات به شبکه می باشند. پیام PDO قابلیت ارسال اطلاعات بدون درخواست از Client های دیگر را داراست.

پیامهای PDO در چند حالت فرستاده می شوند :

1- Async : **ارسال بر اساس تغییر مقدار رجیستر یا بطور دوره ایی بعد از گذشت زمانی معلوم**

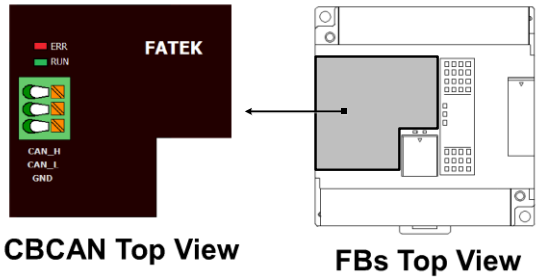
2- sync : **ارسال با تغییر وضعیت ماژولها و پس از دریافت پیغام SYNC**

3- Etime+Sync : **ارسال پس از دریافت پیغام SYNC**

4- Cyclic : **ارسال بطور دوره ایی و بعد از گذشت زمان معلوم**

✓ پیکربندی شبکه CANOpen در FATEK PLC :

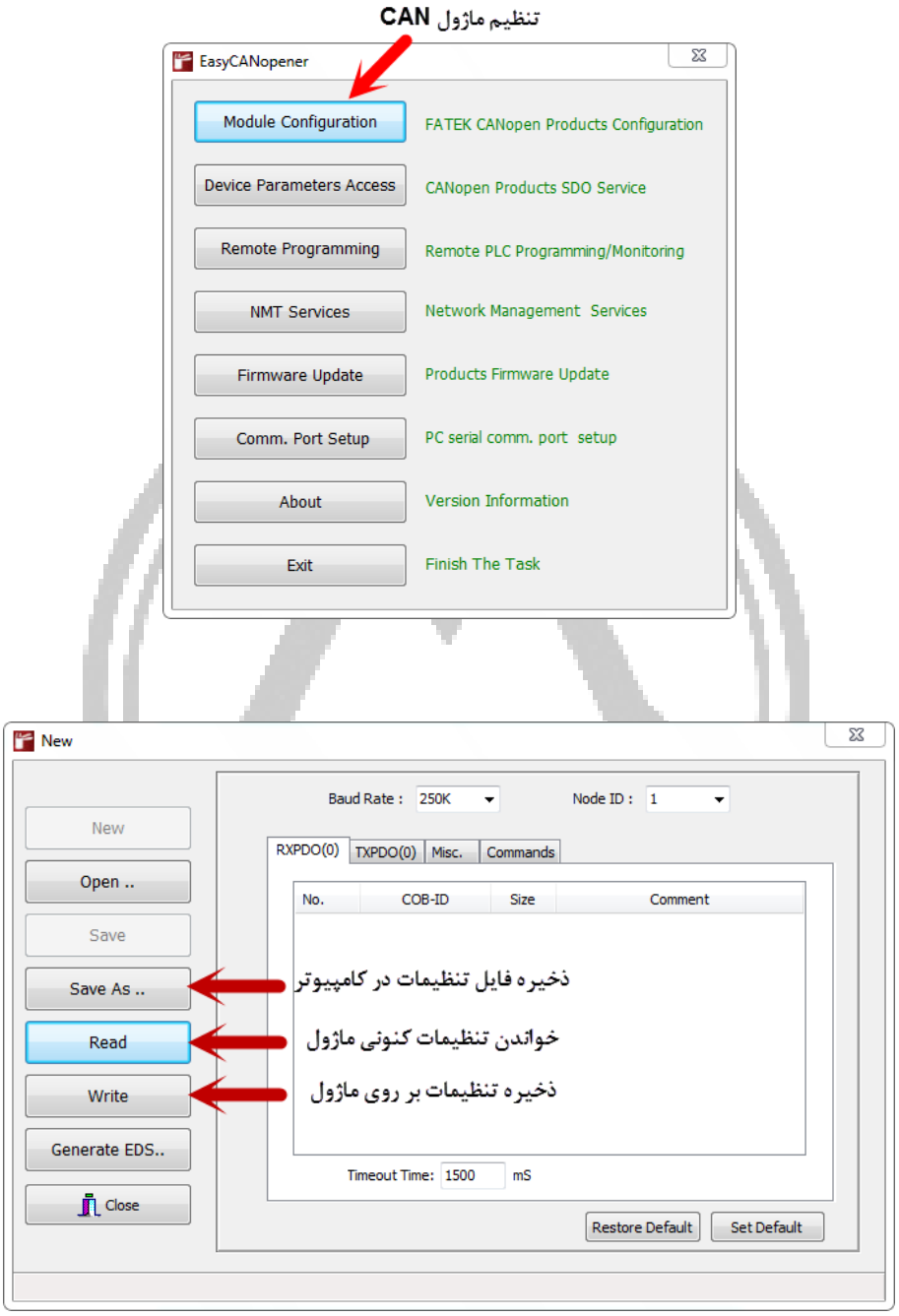
بورد FBs-CBCAN برای اتصال FATEK PLC به شبکه CANOpen می باشد. این ماژول پورتهای **1**و **2** را اشغال نموده و بر روی همه واحدهای اصلی مدلهای سری FBs قابل نصب می باشد. جدول مشخصات و نحوه نصب برروی PLC در زیر آمده است.



پورت ارتباطی **CAN** سری **FBs**

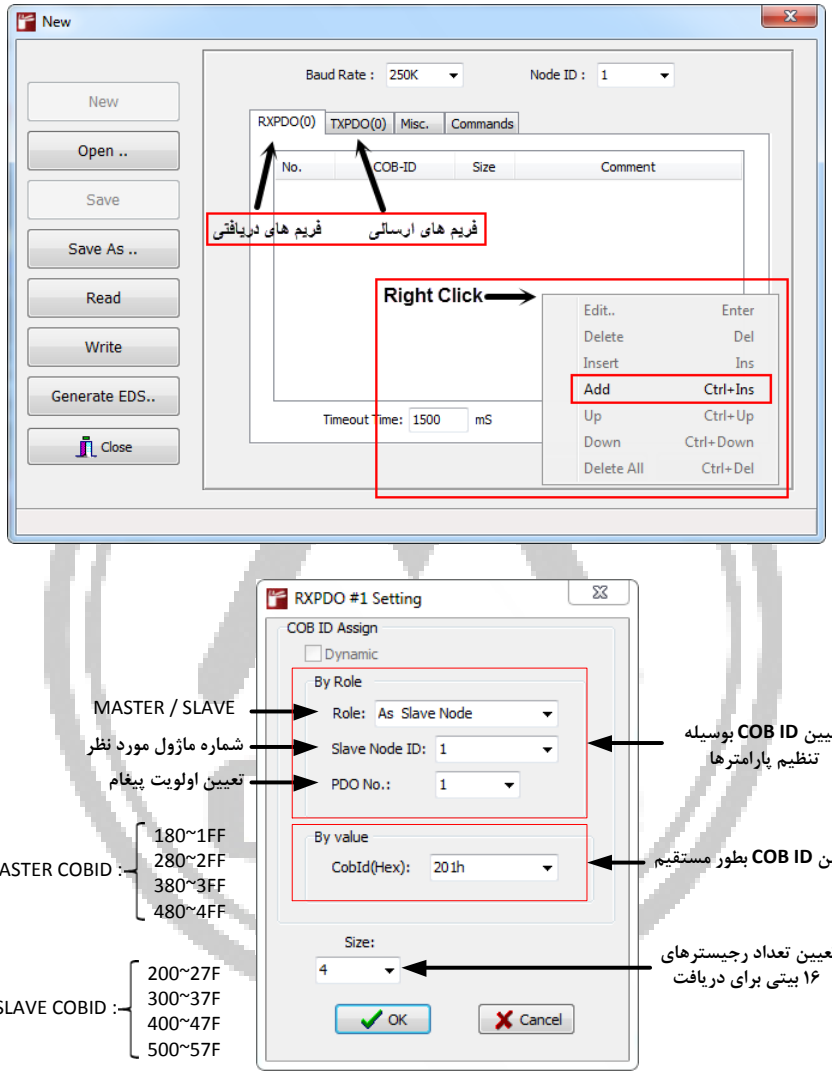
	FBs-CBCAN
Communication standard	CAN 2.0A CANopen
Network topology	3-Phase fieldbus
Communication speed	10K / 20K / 50K / 125K / 250K / 500K / 1Mbps
Maximum number of connection station	127 stations
Method of sending signal	Event or cyclic transmission
Isolation method	Photocouple (signal) isolation, 500VAC, 1 minute
Number of PDO communication	RXPDO-10, TXPDO-10 total up to 80 registers
Number of SDO channels	Client -1, Server-1
Error control	Heartbeat
Wiring mechanism	3-pin spring terminal block
ID setup method	Same as PLC station number or setup by software
Working mode	Master or slave dual modes
Installation position	Expansion slot of main unit

نحوه تنظیم ماژول CAN ، بوسیله نرم افزار Easy can opener می باشد . این نرم افزار از طریق پورتهای سریال مربوط به PLC می تواند ماژول مورد نظر را تنظیم کند.



✓ دریافت پیام ها :

برای دریافت اطلاعات در کل می توان چهار فریم ایجاد کرد که هر فریم دارای یک COB ID و حداکثر 4 رجیستر 16 بیتی می باشد. مقادیر رجیسترهای دریافت شده در رجیسترهای R3655 ~ R3640 ذخیره می شوند.



به دو صورت می توان فریم پیام ها را تشکیل داد :

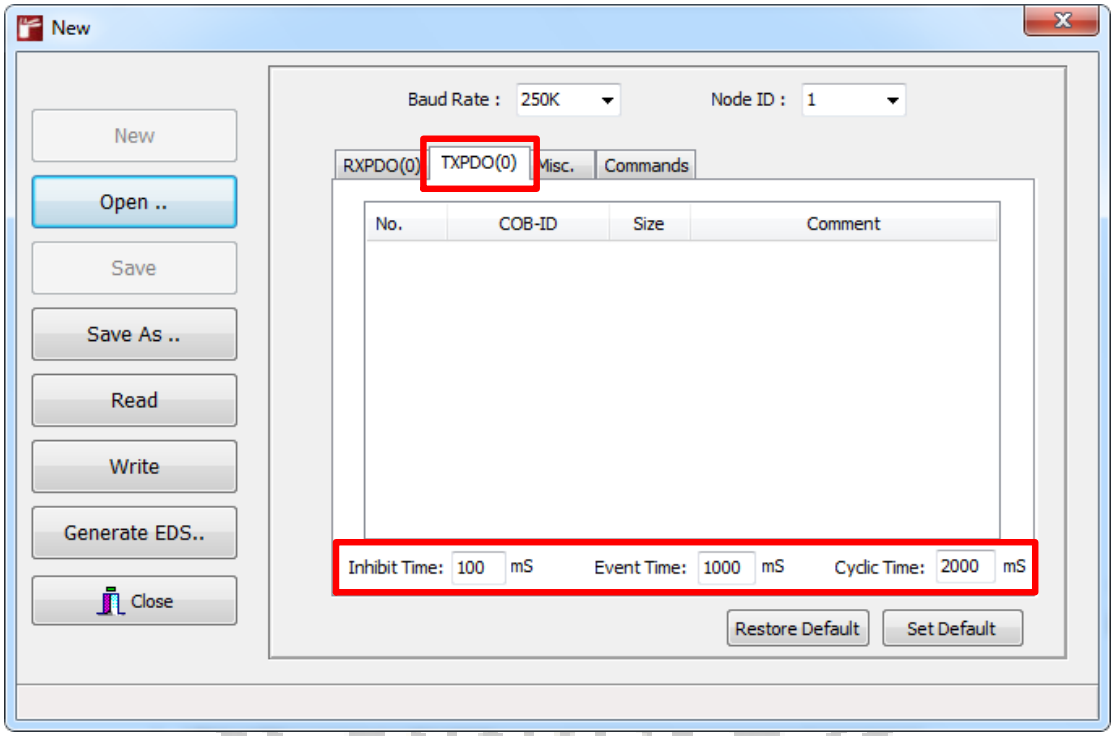
1- By Role : تعیین COB ID با انتخاب چند پارامتر

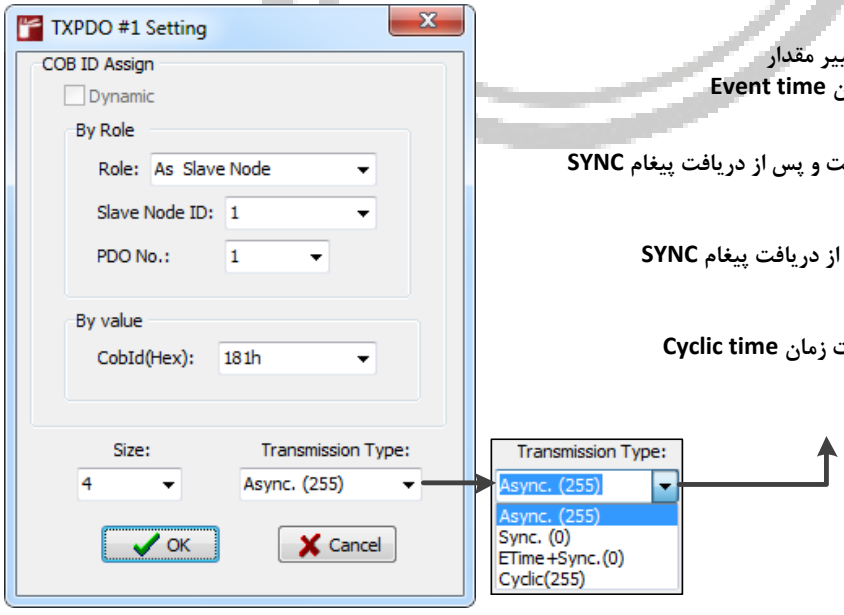
2- By Value : تعیین COB ID بطور مستقیم

اولویتها توسط COB ID تعیین می شوند و می تواند عددی بین 57F ~ 180 هگز باشد. هر چقدر این عدد کوچکتر باشد اولویت بالاتر می باشد. بدین ترتیب اولویت با MASTER و نیز دارای PDO NO کوچکتر می باشد.

✓ ارسال پیام ها :

برای ارسال اطلاعات نیز در کل می توان چهار فریم ایجاد کرد که هر فریم دارای یک COB ID و حداکثر 4 رجیستر 16 بیتی می باشد. این فریم ها مقادیر رجیسترهای R3600 ~ R3615 را ارسال می کنند. در فریم های ارسالی پارامتر Transmission Type باید متناسب با نیاز تنظیم شود .





Async : ارسال بر اساس تغییر مقدار رجیستر یا بعد از گذشت زمان **Event time**

sync : ارسال با تغییر وضعیت و پس از دریافت پیغام **SYNC**

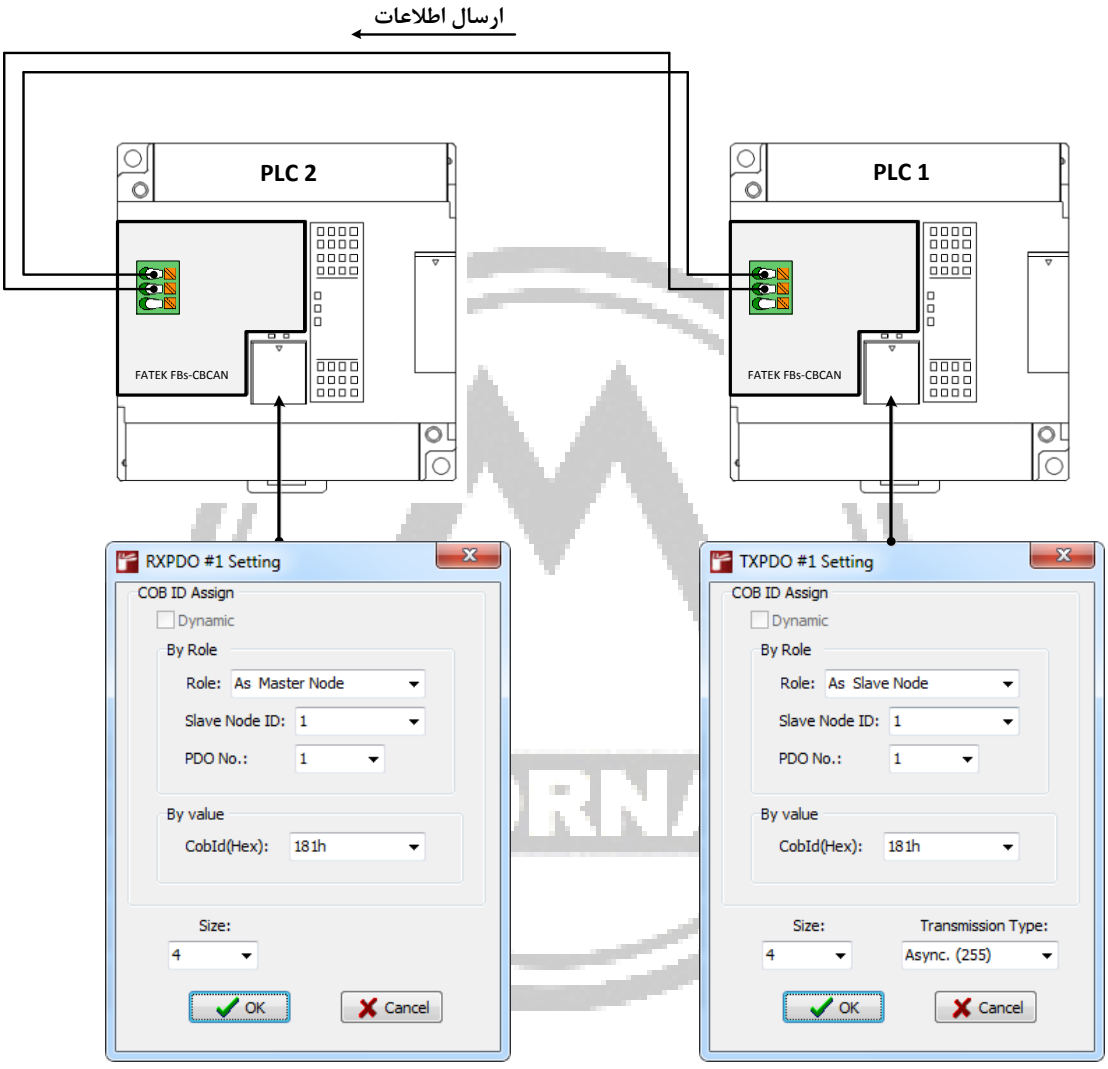
Etime+Sync : ارسال پس از دریافت پیغام **SYNC**

Cyclic : ارسال بعد از گذشت زمان **Cyclic time**

✓ رجیسترهای خاص برای پورت CAN

Sequence	Register	Function	
1	R3680	Module Status	Low Byte: Bit 0 : =0, Normal =1,Stppped when excessive RX error occur while startup. Bit 1: Sync signal time-out, valid only if at least one TXPDO is configured in sync. mode. Bit 2: Reserved Bit 3: =1, CAN Rx error Bit 4: =1, CAN Tx error High Byte: Bit[15:8] CBCAN state : =0, init. =5, OPERATIONAL. =4, STOPPED. =127 PRE-OPERATIONAL
2	R3681	RPDO Status	Each bit represents the receiving status of each RPDO. When Bit =1, it means data update is normal. Bit 0~9 means RPDO 1~10
3	R3682	Heart beat status	Nodes 1-15 ,When Bit1=1, it means the heartbeat in Node 1 is detected.
4	R3683		Node 16-31 ,When Bit1=1, it means the heartbeat in Node 16 is detected.
5	R3684		Node 32-47 ,When Bit1=1, it means the heartbeat in Node 32 is detected.
6	R3685		Node 48-63 ,When Bit1=1, it means the heartbeat in Node 48 is detected.
7	R3686		Node 64-79 ,When Bit1=1, it means the heartbeat in Node 64 is detected.
8	R3687		Node 80-95 ,When Bit1=1, it means the heartbeat in Node 80 is detected.
9	R3688		Node 96-101 ,When Bit1=1, it means the heartbeat in Node 96 is detected.
10	R3689		Node 102-127 ,When Bit1=1, it means the heartbeat in Node 102 is detected.
11	R3690	Time Stamp	Second (0-59)
12	R3691		Minute (0-59)
13	R3692		Hour (0-23)
14	R3693		Day (1-31)
15	R3694		Month (1-12)
16	R3695		Year (2000-2099)
17	R3696	Time Stamp packets receiving indication	The value of this register increment by one when a new Time Stamp packet is received and rollover at 65535.
18	R3697	Ladder software version	The value for ladder software version (Index: 4000H sub-index: 2H) in the object dictionary in decimal. This value will be converted to ASCII characters version, e.g. decimal value *0215 -> ASCII char. '0' '1' '2' '5'*

مثال : می خواهیم از طریق پورت CAN و با استفاده از PDO ، چهار رجیستر 16 بیتی از PLC 1 را به PLC 2 ارسال کنیم.



دریافت فریم و ذخیره در R3640 ~ R3643

ارسال مقادیر رجیسترهای R3600 ~ R3603

مطابق شکل بالا بعد از اتصال پورتهای CAN دو PLC به یکدیگر، با استفاده از نرم افزار CBCAN Configurator و انتخاب گزینه Module Configuration ، تنظیمات هر دو PLC را انجام می دهیم.

در شکل بالا می بینیم که COB-ID هر دو PLC گیرنده و فرستنده با یکدیگر برابر می باشند.

✓ **خواندن یا نوشتن رجیسترهای تجهیزات (Object Dictionary) از طریق سرویس SDO :**

رجیسترهای تجهیزات با پارامترهای Index , Sub-index آدرس دهی می شوند. برای استفاده از سرویس SDO در FATEK PLC رجیسترهای خاص R3700~R3769 استفاده می شود. توضیحات مربوط به این رجیسترها در جدول زیر آمده است.

رجیسترهای PLC	توضیحات
R3700	با قرار دادن عدد 51726 در این رجیستر، پورت CAN شروع به اجرای درخواست می کند.
R3701	نوع فرمان : خواندن مقادیر از رجیسترهای تجهیز دیگر= 5055 نوشتن مقادیر بر رجیسترهای تجهیز دیگر= 5066
R3702	Node ID(station number, 0~127. If =0, this node)
R3703	Object Index (0~65535)
R3704	Object sub-index (0~255)
R3705	طول بایتهای اطلاعات ارسالی یا دریافتی (1~128 بایت)
R3706	اطلاعات شماره 0
R3707	اطلاعات شماره 1
• • •	• • •
R3769	اطلاعات شماره 63

بعد از اجرای سرویس SDO ، رجیسترها مطابق جدول زیر عمل می کنند :

رجیستر PLC	توضیحات	
	خواندن	نوشتن
R3700	این رجیستر کد خطا را نشان می دهد OK : 0 ۱: خطا در کد فرمان ۲: خطا در Node ID ۴: خطا در اجرای SDO	
R3701	بدون تغییر	
R3702		
R3703		
R3704		
R3705	طول اطلاعات خوانده شده	بدون تغییر
R3706	اطلاعات دریافت شده شماره 0	بدون تغییر
R3707	اطلاعات دریافت شده شماره 1	بدون تغییر
. . .	. . .	. . .
R3769	اطلاعات دریافت شده شماره 63	بدون تغییر

چنانچه R3700=4 باشد ، رجیسترهای R3707 , R3706 کد خطا را نشان می دهند

در صورتی که R3700=4 باشد، کد خطا مطابق جدول زیر در رجیسترهای R3707 , R3706 نشان داده می شود.

Error Code Name	*Error Value*	*Description*
ABORT_TIME_OUT	0x05040000L	SDO service T ime out
ABORT_NO_OBJ	0x06020000L	No such object
ABORT_RO	0x06010002L	Attempt to write a read-only object
ABORT_SYS_LENGTH	0x06040047L	Data length exceed system allow
ABORT_NO_SEGEMNT	0x06010000L	Not support segment transfer
ABORT_OBJ_LENGTH	0x06070010L	Not match object length
ABORT_SYNC	0x05040001L	Command specifier not valid
ABORT_TOGGLE_BIT	0x05030000L	Toggle bit not alternated
ABORT_PARM_LENGTH	0x06070012L	Length of service parameter too high
ABORT_WO	0x06010001L	Attempt to read a write-only object
ABORT_READ_LENGTH	0x05040005L	Object length too big to read

توابع فتک

کل توابع فتک

شرح	مشخصه دستور	شماره دستور
تعریف اجرا شدن یا عدم اجرای قسمتهایی از برنامه	MC	0
دستور پایان حلقه کنترلMC	MCE	1
تعریف اجرا شدن یا عدم اجرای قسمتهایی از برنامه	SKP	2
دستور پایان حلقه کنترلSKP	SKPE	3
با لبه بالا رونده ، خروجی به اندازه Scan Time فعال باشد	DIFU	4
با لبه پایین رونده ، خروجی به اندازه Scan Time فعال می شود.	DIFD	5
بیتهای رجیستر D را به تعداد 1 بیت به سمت راست یا چپ شیفت می دهد	BSHF	6
شمارنده 16 یا 32 بیتی در جهت بالا یا پایین	UDCTR	7
کپی کردن محتوای رجیستر مبدا در رجیستر مقصد	MOV	8
ابتدا رجیستر مبدا را اینورس (0ها به 1 و 1ها به 0 تبدیل می شوند) کرده و سپس در رجیستر مقصد کپی می کند	MOV/	9
صفر به یک ویک به صفر تبدیل می شود	TOGG	10
دو رجیستر را با یکدیگر جمع کرده و در رجیستر مقصد قرار می دهد	(+)	11
دو رجیستر را از یکدیگر تفریق کرده و در رجیستر مقصد قرار می دهد	(-)	12
دو رجیستر را با یکدیگر ضرب کرده و در رجیستر مقصد قرار می دهد	(*)	13
دو رجیستر را بر یکدیگر تقسیم کرده و در رجیستر مقصد قرار می دهد	(/)	14
به رجیستر مورد نظر 1 واحد اضافه می کند	(+1)	15
از رجیستر مورد نظر 1 واحد کم می کند	(-1)	16
محتوای دو رجیستر را با یکدیگر مقایسه کرده و مساوی بودن یا کوچکتر یا بزرگتر بودن دو رجیستر را بررسی می کند	CMP	17
بیت های موجود درSa وSb را با هم AND کرده و نتیجه را در D می ریزد.	AND	18
بیت های موجود درSa وSb را با هم OR کرده و نتیجه را در D می ریزد.	OR	19
این دستور برای تبدیل به فرمت BCD استفاده می گردد . اگر داده S در رنج BCD نباشد،"ERR" فعال می شود و اطلاعات قبلی D بدون تغییر باقی می مانند	→BCD	20

21	→BIN	تبدیل عدد بافرمت BCD به عددی با فرمت باینری
22	BREAK	خروج از حلقه FOR-NEXT
23	DIV48	تقسیم 48 بیت بر 48 بیت
24	SUM	مجموع N رجیستر
25	MEAN	میانگین N رجیستر
26	SQRT	مجذور دوم محتوای رجیستر
27	NEG	مقدار D منفی می شود و دوباره در همان رجیستر ریخته می شود.
28	ABS	این تابع،قدر مطلق مقدار ریجیستر D را گرفته و دوباره در D می ریزد.
29	EXT	تبدیل فرمت رجیستر 16بیتی به فرمت رجیستر 32بیتی
30	PID	کنترل بصورت PID (برای کنترل دما ، از فانکشن 86 استفاده شود)
31	CRC	CRC16 checksum calculation
32	ADCNV	تبدیل بازه ورودی آنالوگ از mA[0,20] به بازه mA[4,20]
33	LCNV	تبدیل خطی از بازه [A,B] به بازه [C,D]
34	MLC	Multiple Linear Conversion
35	XOR	عملیات منطقی XOR بین دو رجیستر
36	XNR	عملیات منطقی XNOR بین دو رجیستر
37	ZNCMP	مقایسه محتوای یک رجیستر در دو بازه
38	-	"رزرو"
39	-	"رزرو"
40	BITRD	بیت Nام از رجیستر 16بیتی
41	BITWR	مقداردهی یک بیت در رجیستر 16بیتی
42	BITMV	انتقال یک بیت از یک رجیستر به یک بیت از رجیستر دیگر
43	NBMV	انتقال چهار بیت از یک رجیستر به چهار بیت از رجیستر دیگر
44	BYMV	انتقال یک بایت از رجیستر به یک بایت از رجیستر دیگر
45	XCHG	مقادیر رجیستر Da و Db با هم عوض می شوند.
46	SWAP	بایت بالا و پایین رجیستر را با هم عوض می کند.
47	UNIT	این تابع نیبل های پایین N تعداد از رجیستر ها را به ترتیب در رجیستر D قرار می دهد.

در S یک رجیستر 16 بیتی قرار می گیرد که به ترتیب N تعداد از نیبل های آن به رجیستر D،D+1 و... منتقل می شود.	DIST	48
این تابع، بیت ها ی پایین N تعداد از رجیستر های مشخص شده در S را به ترتیب در رجیستر های D،D+1،... قرار می دهد.	BUNIT	49
این تابع، بایت ها ی مبدا را (N تعداد) به بایت های پایین رجیستر D،D+1، ... منتقل می کند.	BDIST	50
شیفت دادن به چپ	SHFL	51
شیفت دادن به راست	SHFR	52
چرخاندن بیتهای رجیستر به چپ	ROTL	53
چرخاندن بیتهای رجیستر به راست	ROTR	54
تبدیل باینری به کد گری	B →G	55
تبدیل کد گری به باینری	G →B	56
دیکدر	DECOD	57
انکدر	ENCOD	58
تبدیل به عددی برای نمایش در سون سگمنت	→7SG	59
تبدیل به کد اسکی	→ASC	60
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.	→SEC	61
زمان بر حسب ثانیه را بر حسب ساعت،دقیقه و ثانیه تبدیل می کند.	→HMS	62
N تعداد از کدهای ASCII که در S ذخیره شده اند،به معادل هگز خود تبدیل شده و در D ذخیره می شوند.	→HEX	63
N تعداد از نیبل های S را به صورت معادل ASCII در D،D+1،... ذخیره می کند.	→ASC Ⅱ	64
لیبل برای زیربرنامه ها یا اینتراپتها	LBL	65
پرش به لیبل مورد نظر	JMP	66
فراخوانی زیر برنامه مورد نظر	CALL	67
PLC بعد از دیدن این تابع به انتهای تابع CALLی می رود که از طریق آن ، این زیربرنامه فراخوانی شده است	RTS	68
در انتهای روتین INTERRUPT قرار می گیرد.	RTI	69
شروع حلقه FOR	FOR	70

اجرای حلقه بعدی FOR	NEXT	71
"رزرو"	-	72
"رزرو"	-	73
رفرش کردن ورودی و خروجی ها	IMDIO	74
	-	75
این تابع می تواند ورودی 0-9 را توسط ورودی دریافت کن	TKEY	76
4 ورودی اول PLC به 4 خروجی اول PLC طوری متصل می شوند که اتصال هر کدام از آنها یکی از خروجی ها را بدهد.	HKEY	77
این تابع 4 رقم دهدهی را از سوئیچ BCD دستی ،بازخوانی می کند و انها را در D ذخیره می کند.	DSW	78
4 نیبل رجیستر مشخص شده در S ،برای نمایش به SEG-7 سری اول منتقل می شوند	7SGDL	79
این تابع از متد مالتی پلکس برای خواندن N×8 ورودی استفاده می کند که فقط 8 ورودی و N خروجی از PLC را استفاده می کند.	MUXI	80
(motor Pulse output function (for bi-directional drive of step	PLSO	81
پالسی به خروجی OT می فرستد که To میلی ثانیه ON است و پریود آن Tp میلی ثانیه می باشد	PWM	82
این تابع برای به دست آوردن سرعت گردش دستگاه های چرخنده (مانند موتور) بر حسب rpm استفاده می شود	SPD	83
کاراکترهای مختلف را برای نمایش در seg-16 و seg-17 آماده می کند	TDSP	84
"رزرو"	-	85
حلقه کنترل PID دما	TPCTL	86
S0.01 این تابع مانند تایمر ساده است با این تفاوت که این تایمر قابلیت نگه داشتن زمان را دارد.	T.01S	87
S0.1 این تابع مانند تایمر ساده است با این تفاوت که این تایمر قابلیت نگه داشتن زمان را دارد.	T.1S	88
S1 این تابع مانند تایمر ساده است با این تفاوت که این تایمر قابلیت نگه داشتن زمان را دارد.	T1S	89
تنظیم تایمر نگهبان	WDT	90

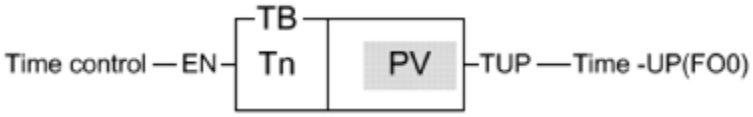
ریست کردن تایمر نگهبان	RSWDT	91
خواندن شمارنده سرعت بالا نرم افزاری	HSCTR	92
بارگذاری شمارنده های سرعت بالا	HSCTW	93
اطلاعاتی را که به صورت ASCII کد شده اند و از S شروع می شوند را به Port1 منتقل می کند	ASCWR	94
تابع شیب برای خروجی دیجیتال به انالوگ	RAMP	95
"رزرو"	-	96
"رزرو"	-	97
Tracking type ramp function for D/A output	RAMP2	98
"رزرو"	-	99
-	توضیح جدول	-
فانکشن 107 برای کپی تعدادی از رجیسترها در رجیسترهای دیگر استفاده می شود.	R →T	100
این تابع بر عکس تابع قبل عمل کرده و محتویات یک رجیستر از جدول مورد نظر را به رجیستر مقصد منتقل می کند.	T →R	101
((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. محتویات آن رجیستری از جدول مبدا که Pr به آن اشاره می کند،به رجیستر معادلش در جدول مقصد منتقل می شود.))	T →T	102
محتویات رجیسترهای جدول مبدا به داخل رجیسترهای معادلشان در جدول مقصد،کپی می شوند	BT_M	103
محتویات رجیسترهای جدول a با محتویات رجیسترهای معادلشان در جدول b ،جا به جا می شوند.	T_SWP	104
جستجو و مقایسه یک ریجیستر با یک جدول و اعلام اینکه با محتوای کدام ریجیستر برابر است	R-T_S	105
این تابع مانند تابع قبل است با این تفاوت که محتوای رجیستر های معادل از دو جدول Ta و Tb با هم مقایسه می شوند.	T-T_C	106
محتوای رجیستر Rs، تمام رجیسترهای جدول Td را کپی می کند.	T_FIL	107
محتوای ریجیسترها به ریجیستر بعدی یا قبلی کپی می شود	T_SHF	108

محتوای رجیسترهای جدول Ts ، یک رجیستر به سمت چپ یا راست می چرخند و نتیجه در Td ذخیره می شود.	T_ROT	109
دستور Queue	QUEUE	110
دستور Stack	STACK	111
مقایسه بین محتوای Rs و یک جفت از رجیسترهای جدول که از Ts شروع می شوند ، صورت می گیرد	BKCMP	112
بر اساس محتوای رجیسترها ، رجیسترها را مرتب می کند	SORT	113
این تابع N تعداد از رجیسترها که از D شروع می شوند را set (1) یا reset 0)) می کند.	Z-WR	114
"رزرو"	-	115
"رزرو"	-	116
"رزرو"	-	117
"رزرو"	-	118
"رزرو"	-	119
اجرای عملیات AND برروی بیتهای چند رجیستر	MAND	120
اجرای عملیات OR برروی بیتهای چند رجیستر	MOR	121
اجرای عملیات XOR برروی بیتهای چند رجیستر	MXOR	122
اجرای عملیات XNOR برروی بیتهای چند رجیستر	MXNR	123
تمام بیت های تعدادی رجیستر را معکوس می کند (0 ها 1 شده و 1 ها 0 می شوند).	MINV	124
بیتهای چند رجیستر را با یکدیگر مقایسه می کند	MCMP	125
خواندن یک بیت از چند رجیستر	MBRD	126
نوشتن بر یکی از بیتهای چند رجیستر	MBWR	127
شیفت دادن بیتهای چند رجیستر همزمان با یکدیگر	MBSHF	128
چرخاندن بیتهای چند رجیستر همزمان با یکدیگر	MBROT	129
شمارش تعداد بیت های 0 یا 1 موجود در چند رجیستر	MBCNT	130
"رزرو"	-	131
"رزرو"	-	132
"رزرو"	-	133

134	-	"رزرو"
135	-	"رزرو"
136	-	"رزرو"
137	-	"رزرو"
138	-	"رزرو"
139	HSPWM	تولید پالس PWM
140	HSPSO	تولید پالس سرعت بالا
141	MPARA	تنظیم پارامترهای پالس های سرعت بالا
142	PSOFF	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 یا در جه، inch و پالس.
144	-	"رزرو"
145	EN	فعال کردن اینتراپت خارجی
146	DIS	اینتراپت را غیر فعال می کند
147	MHSPO	تولید پالس بین چند محور بطور وابسته به یکدیگر
148	MPG	Manual pulse generator for positioning
149	-	"رزرو"
150	M-Bus	اجرای توابع MODBUS
151	CLINK	اجرای پروتوکل FACON
152	-	"رزرو"
153	-	"رزرو"
154	-	"رزرو"
155	-	"رزرو"
156	-	"رزرو"
157	-	"رزرو"
158	-	"رزرو"
159	-	"رزرو"
160	RW-FR	File register access
161	WR-MP	ذخیره رجیسترهای PLC در حافظه بیرونی

خواندن رجیسترهای حافظه بیرونی	RD- MP	162
"رزرو"	-	163
"رزرو"	-	164
"رزرو"	-	165
"رزرو"	-	166
"رزرو"	-	167
"رزرو"	-	168
"رزرو"	-	169
مساوی بودن دو رجیستر	=	170
کوچکتر یا بزرگتر بودن دو رجیستر	>	171
کوچکتر یا بزرگتر بودن دو رجیستر	<	172
نا مساوی بودن دو رجیستر	< >	173
کوچکترمساوی یا بزرگترمساوی بودن دو رجیستر	> =	174
کوچکترمساوی یا بزرگترمساوی بودن دو رجیستر	= <	175
تبدیل اینتیجر به عدد اعشاری	I→F	200
تبدیل عدد اعشاری به عدد اینتیجر	F→I	201
مجموع دو رجیستر با فرمت عدد اعشاری	FADD	202
تفریق دو رجیستر با فرمت عدد اعشاری	FSUB	203
ضرب دو رجیستر با فرمت عدد اعشاری	FMUL	204
تقسیم دو رجیستر با فرمت عدد اعشاری	FDIV	205
مقایسه دو رجیستر با فرمت عدد اعشاری	FCMP	206
مقایسه در یک بازه رجیستر با فرمت عدد اعشاری	FZCP	207
مجذور دوم رجیستر با فرمت عدد اعشاری	FSQR	208
سینوس رجیستر با فرمت عدد اعشاری	FSIN	209
کسینوس رجیستر با فرمت عدد اعشاری	FCOS	210
تانژاند رجیستر با فرمت عدد اعشاری	FTAN	211
منفی کردن رجیستر با فرمت عدد اعشاری	FNEG	212
قدر مطلق رجیستر با فرمت عدد اعشاری	FABS	213

TIMER



Tn : شماره تایمر

TB : واحد شمارش تایمر

PV : بازگذاری مقدار شمارش تایمر

TUP : خروجی تایمر

تعداد تایمرها مجموعاً **256** عدد است.(**T0~T255**) . سه زمان پایه (**TB**) وجود دارد با مقادیر **0.01s , 0.1s , 1s**

تخصیص تایمرهای مختلف به **TB**ها بصورت پیش فرض به صورت زیر می باشد:

T0~T49 : 0.01s

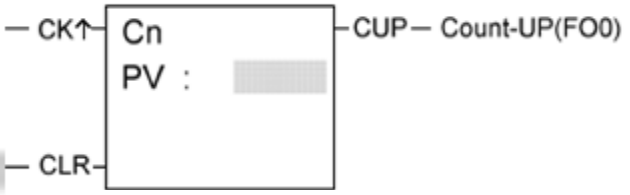
T50~T199 : 0.1s

T200~T255 : 1s

محاسبه زمان تایمر بدین صورت می باشد: زمان تایمر =**TB×PV**

هرگاه پایه" **EN**"فعال شود(**1** شود) تایمر فعال می شود،**CV**از مقدار **0**شروع به افزایش می کند تا زمانی که به مقدار **PV** برسد،آنگاه اگر **M1957=0** مقدار **CV** تا ماکزیمم مقدار **PV**(**32767s**) 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.

COUNTER



در مجموع 200 کانتر 16 بیتی وجود دارد (C0~C199) و 56 کانتر 32 بیتی (C200~C255).

کانتر های C200~C239و C0~C130 کانترهای محافظتی هستند که به هنگام نقص توان یا توقف PLC مقدار CV را حفظ می کنند تا زمانی کهPLC دوباره راه اندازی شود. در کانتر های دیگر ،مقدار CV هنگام توقف PLC ، reset می شود .(0 می شود) . با این تابع ماکزیمم فرکانس شمارش می تواند تا 20Hz افزایش یابد ، برای فرکانسهای بالاتر از کانتر سرعت بالا باید استفاده کرد . با فعال کردن پایه CLR ،پایه های دیگر (CUP و CK) غیر فعال می شوند و هرگاه CLR ،0 باشد کانتر شروع به فعالیت می کند.

SET

D: رجیستر مورد نظر ، هرگاه "EN" فعال شود تمام بیت های رجیستر های مشخص شده ، 1 می شود.

RESET

هرگاه "EN" فعال شود تمام بیت های رجیستر های مشخص شده ، 0 می شود.

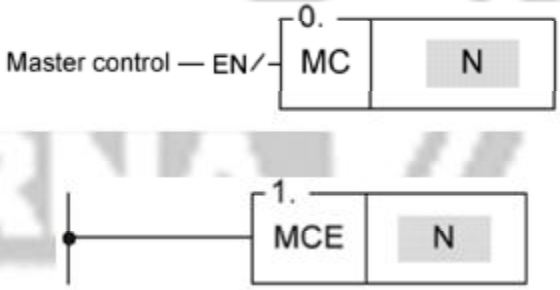
PROGRAM END

هرگاه "EN" از 0 به 1 تغییر کند: این تابع فعال شده ، اسکن برنامه از ابتدا آغاز شده و باقی برنامه ها بعد از تابع END ، دیگر اجرا نمی شوند. وقتی 0="EN" ،این تابع نادیده گرفته شده و برنامه های بعد از این تابع اجرا خواهند شد. به کار بردن END در برنامه اصلی ضرورتی ندارد زیرا CPU ، به صورت خودکار وقتی به انتهای برنامه رسید،به نقطه شروع می رود.

Function 0&1:MC & MCE

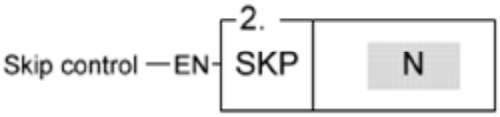
MC:Master Control Loop Start

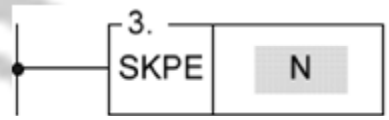
MCE:Master control Loop End



در کل 128 حلقه MC وجود دارد (N=0~127) . هر تابع MC با یک تابع MCE مرتبط است و شماره حلقه آنها N یکسان می باشد . باید توجه کرد که تابع MCE بعد از MC قرار بگیرد. هرگاه "EN" فعال شود، تمام توابع و برنامه هایی که زیر حلقه MC نوشته می شوند اجرا می شود. و چنانچه "EN" غیر فعال شود، تمام توابع و برنامه هایی که زیر حلقه MC نوشته می شوند اجرا نمی شود و در سیکل اسکن تایم برنامه قرار نمی گیرند و تمام مقادیر خروجی ها و تایمر ها ،با غیرفعال شدن MC ،0 می شوند. MCE نیازی به ورودی ندارد چون وابسته به MC است.

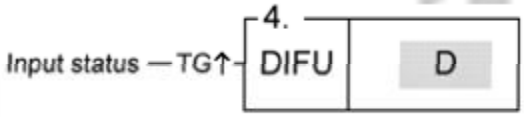
Function 2&3: SKIP START & SKIP END





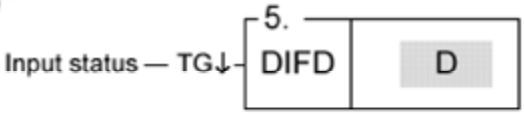
در کل 128 حلقه SKP وجود دارد (N=0~127) . هر تابع SKP با یک تابع SKPE مرتبط است و شماره حلقه آنها N یکسان می باشد . باید توجه کرد که تابع SKPE بعد از SKP قرار بگیرد. هرگاه "EN" غیر فعال شود، تمام توابع و برنامه هایی که زیر حلقه SKP نوشته می شوند اجرا می شود. و چنانچه "EN" فعال شود، تمام توابع و برنامه هایی که زیر حلقه SKP نوشته می شوند اجرا نمی شود و در سیکل اسکن تایم برنامه قرار نمی گیرند و تمام مقادیر خروجی ها و تایمر ها ،با فعال شدن SKP ،0 می شوند. SKPE نیازی به ورودی ندارد چون وابسته به SKP است.

Function 4:DIFFERENTIAL UP



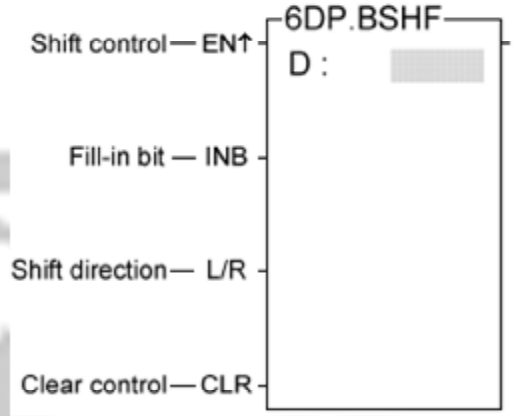
این تابع باعث می شود که هرگاه ورودی TG فعال شد ،خروجی به اندازه Scan Time فعال باشد.

Function 5.DIFFERENTIAL DOWN



وقتی ورودی TG فعال می شود، به محض غیر فعال شدن ،خروجی به اندازه Scan Time فعال می شود.

Function 6.BIT SHIFT

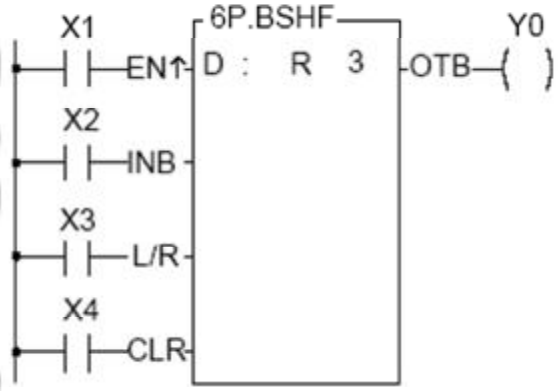


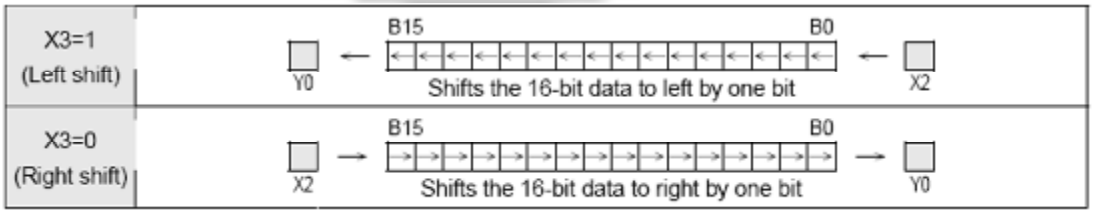
با این تابع می توان رجیستر مورد نظر را **1** بیت به چپ یا راست ، شیفت داد. بدین منظور **CLR** باید **0** باشد.

برای شیفت به راست 0 ="L/R" و برای شیفت به چپ **1="L/R"** باید باشد .

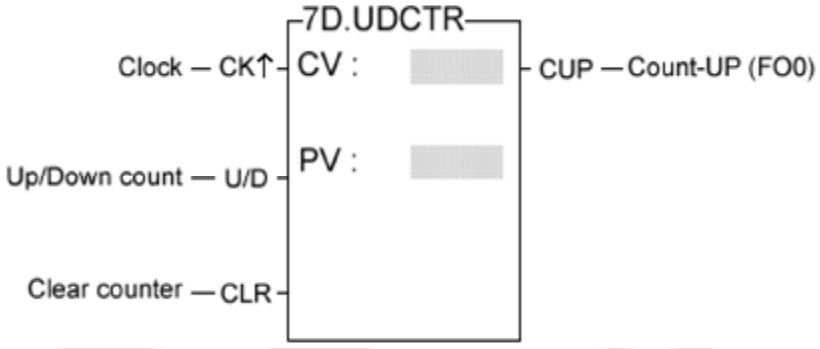
با **"INB"** می توان تعیین نمود که بیت ایجاد شده پس از اعمال شیفت، **0** یا **1** بشود.

مثال:





Function 7.UP/DOWN COUNTER



این کانتر می تواند به صورت دو فاز اعمال شود.

اگر **"U/D"=1** ، با فعال بودن **CK** ، **CV** افزایش می یابد و اگر **"U/D"=0** باشد،کاهش می یابد.

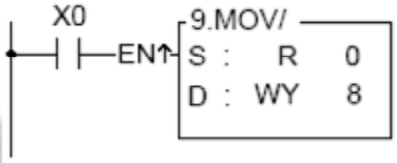
Function 8.MOVE

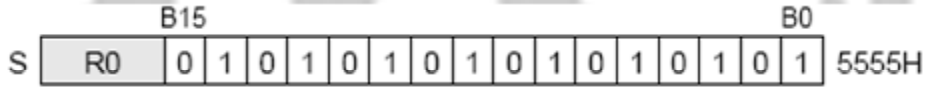
هرگاه **"EN"** فعال شود،اطلاعات موجود در رجیستر **S** به **D** کپی می شود.

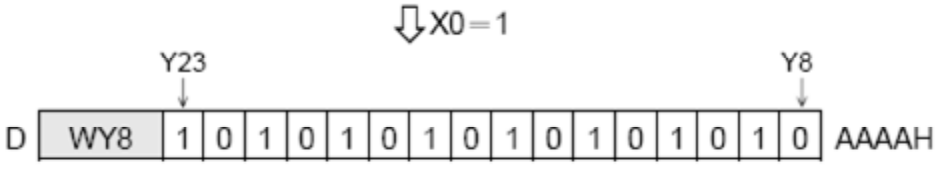
Function 9. MOVE INVERSE

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

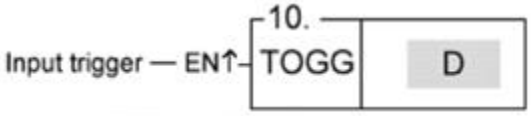
مثال:





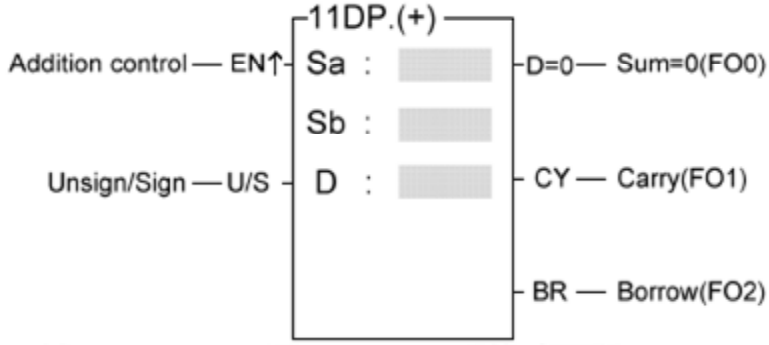


Function 10.TOGGLE SWITCH



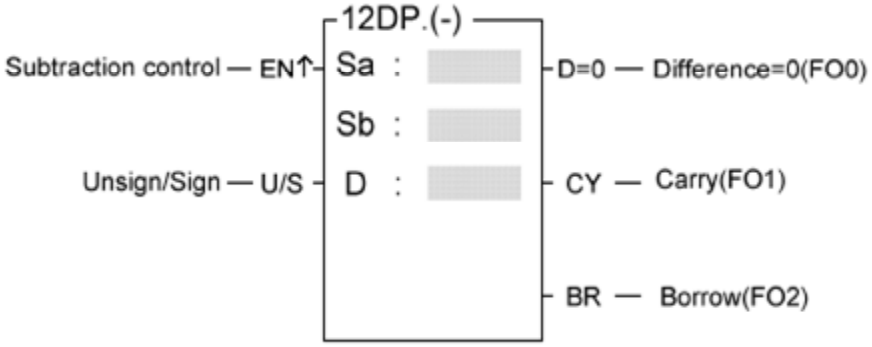
با فعال شدن "EN" ،بیت نوشته شده در D ، اگر 1 باشد 0 و اگر 0 باشد 1 می شود .

Function 11.ADDITION



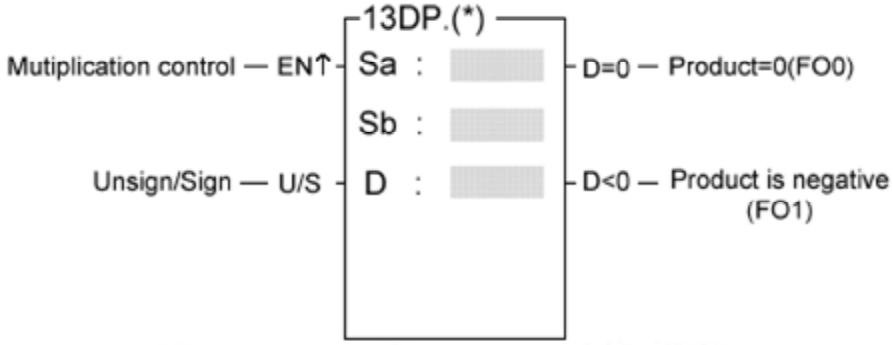
با فعال شدن "EN" ، داده های موجود در Sa و Sb با هم جمع شده و در D ریخته می شود. Sa+Sb=D

Function 12.SUBTRACTION



با فعال شدن "EN" ، داده های موجود در Sa ، منهای Sb شده و نتیجه در D ریخته می شود. Sa-Sb=D

Function 13.MULTIPLICATION

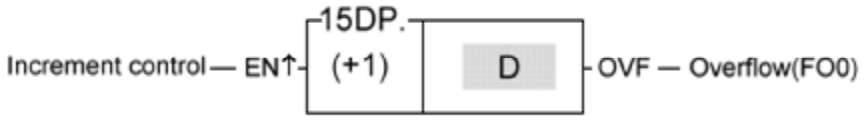


با فعال شدن **"EN"** ، داده های موجود در **Sa**، ضرب در **Sb** شده و نتیجه در **D** ریخته می شود. **Sa×Sb=D**

Function 14.DIVISION

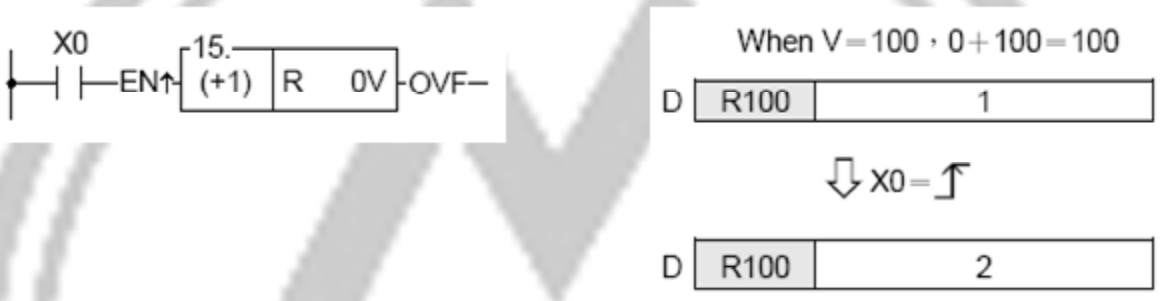
با فعال شدن **"EN"** ، داده های موجود در **Sa**، تقسیم بر **Sb** شده و و نتیجه در **D** ریخته می شود. چنانچه از این فانکشن در حالت **16** بیتی استفاده شود باقیمانده این تقسیم در رجیستر **D+1** قرار خواهد گرفت و چنانچه از این فانکشن در حالت **32** بیتی استفاده شود باقیمانده این تقسیم در رجیستر **D+2** قرار خواهد گرفت . هرگاه **Sb** صفر باشد ، تابع اجرا نشده و **Error** می دهد.

Function 15.INCREMENT

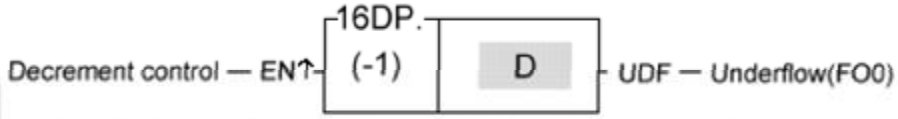


هرگاه "EN" از 0 به 1 تغییر کند ، به مقدار رجیستر D ، "1" واحد اضافه می شود . اگر این افزایش ، باعث از محدوده (range) خارج شدن مقدار D شود، خروجی "OVF" فعال شده و مقدار D منفی می شود.

مثال:



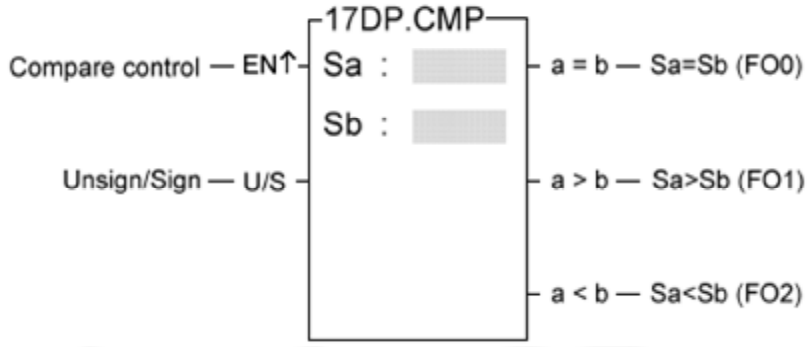
Function 16.DECREMENT



هرگاه "EN" از 0 به 1 تغییر کند، از مقدار رجیستر D ، "1" واحد کم می شود . اگر این کاهش ، باعث از محدوده (range) خارج شدن مقدار D شود، خروجی "UDF" فعال شده و مقدار D پس از رسیدن به آخرین حد عدد منفی ، مثبت می شود.

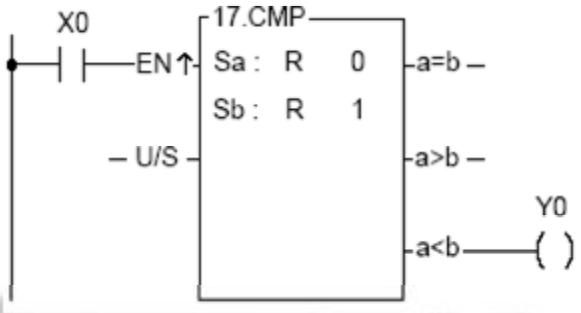
مثال:

Function 17.COMPRARE



هرگاه "EN" از 0 به 1 تغییر کند،مقدار Sa و Sb را مقایسه می کند. اگر با هم برابر باشند ،خروجی "a=b" فعال شده و اگر Sa>Sb باشد، خروجی "a>b" فعال شده و اگر Sa<Sb باشد،خروجی "a<b" فعال می شود.

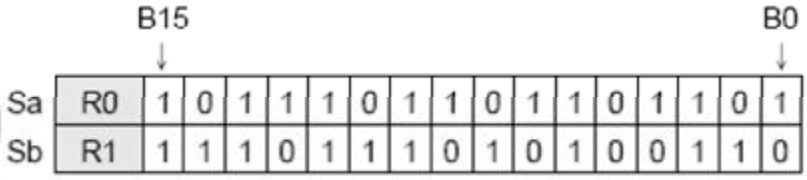
مثال:

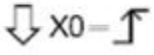


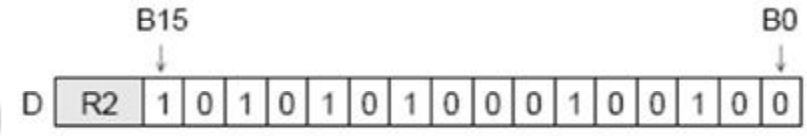
Function 18.LOGICAL AND

هرگاه **"EN"** از **0** به **1** تغییر کند، بیت های موجود در **Sa** و**Sb** را با هم **AND** کرده و نتیجه را در **D** می ریزد. هرگاه تمام بیت های **D** ، صفر شود ، خروجی **"D=0"** فعال می شود .

مثال:



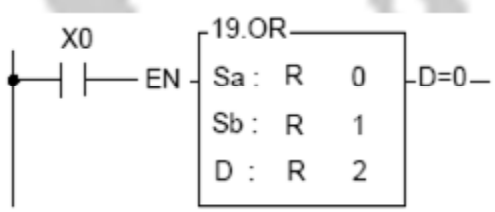


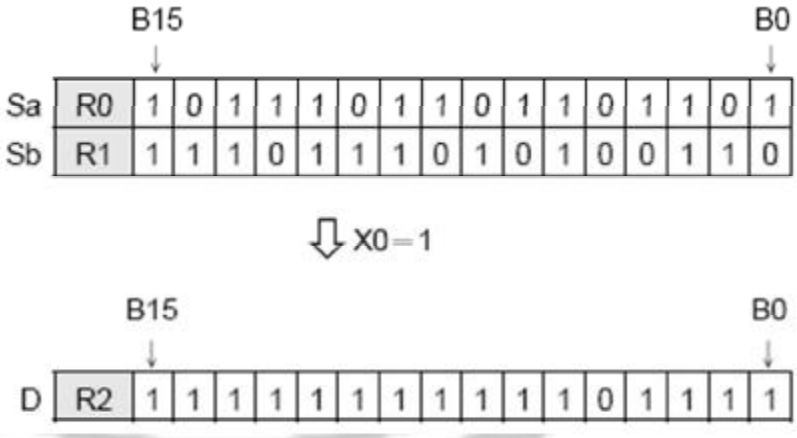


Function 19.LOGICAL OR

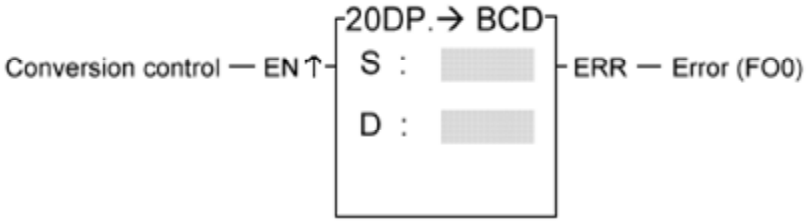
هرگاه "EN" از 0 به 1 تغییر کند، بیت های موجود در Sa وSb را با هم OR کرده و نتیجه را در D می ریزد. هرگاه تمام بیت های D ، صفر شود ، خروجی "D=0" فعال می شود .

مثال:



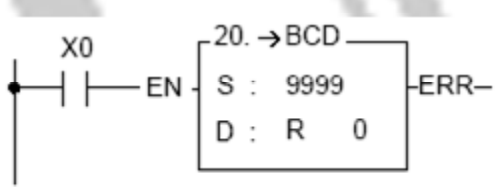


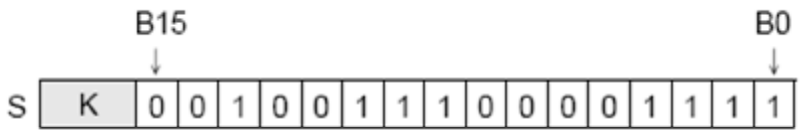
Function 20.BIN TO BCD CONVERSION

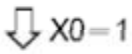


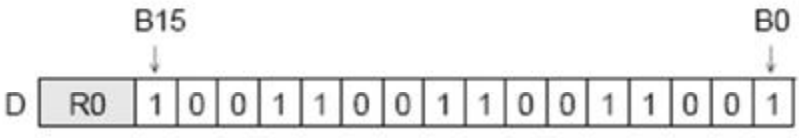
هرگاه "EN" از 0 به 1 تغییر کند، داده های موجود در S را که به صورت کد باینری است، به صورت BCD درآورده و در D می ریزد. اگر داده S در BCD Range نباشد،خروجی "ERR" فعال می شود و اطلاعات قبلی D بدون تغییر باقی می مانند.

مثال:

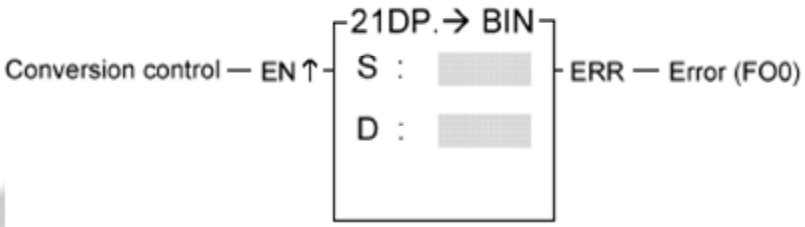






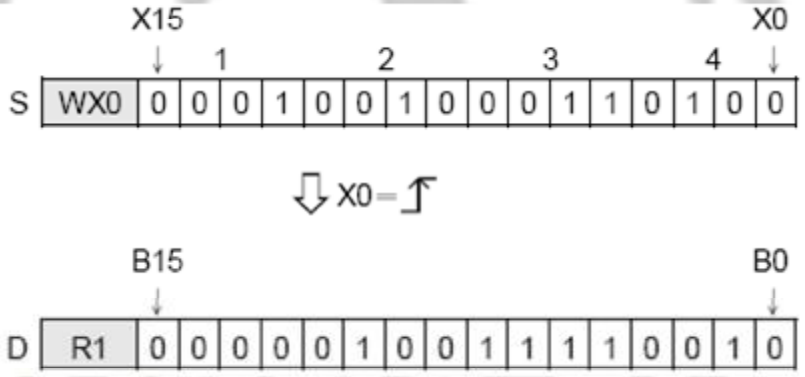


Function 21.BCD TO BIN CONVERSION



برعکس **BIN TO BCD** عمل می کند.

مثال:



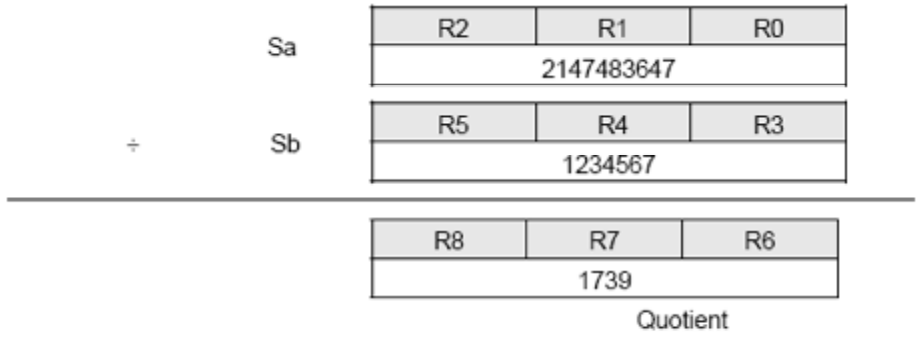
Function 22.BREAK

این تابع باید به همراه حلقه ی **FOR-NEXT (70-71)** استفاده شود. حلقه ی **FOR-NEXT** به طور معمول **N** بار اجرا می شود؛ اما اگر توقف حلقه ضروری بود ، از تابع **BREAK** استفاده می کنیم.

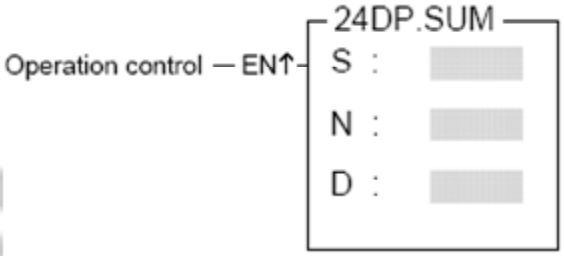
Function 23.48-BIT DIVISON

هرگاه **"EN"** از **0** به **1** تغییر کند، مقدار **48** بیتی موجود در **Sa** را تقسیم بر مقدار **48** بیتی موجود در **Sb** کرده و خارج قسمت را در **D** می ریزد . اگر نتیجه صفر باشد، خروجی **"D=0"** فعال می شود . اگر **Sb** صفر باشد، خروجی **"ERR"** فعال می شود .

مثال:

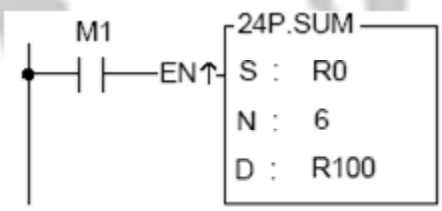


Function 24.SUM



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

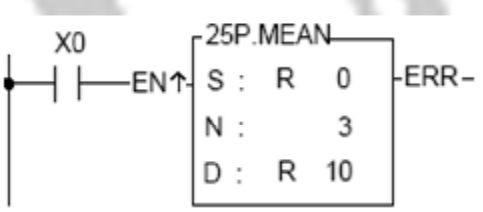
مثال:

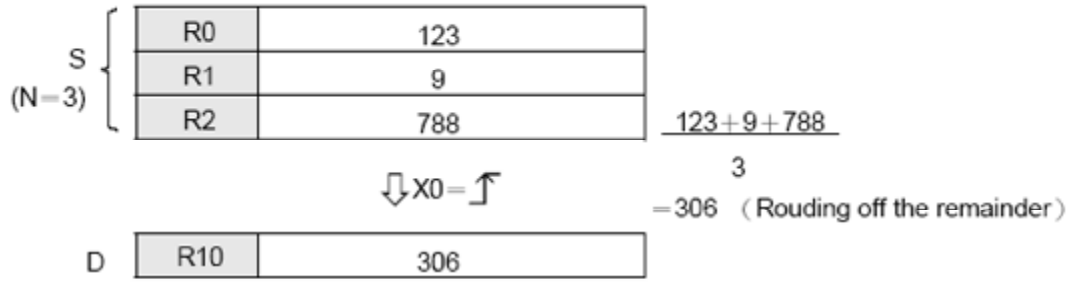


Function 25.MEAN

این تابع برای میانگین گرفتن از مقادیر چند رجیستر متوالی کاربرد دارد. رجیسترشروع در **S** قرار گرفته و تعداد رجیسترها در **N** قرار می گیرد . هرگاه **"EN"** از **0** به **1** 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 **2** تا **256** نباشد ،**"ERR"** فعال شده و تابع اجرا نمی شود.

مثال:



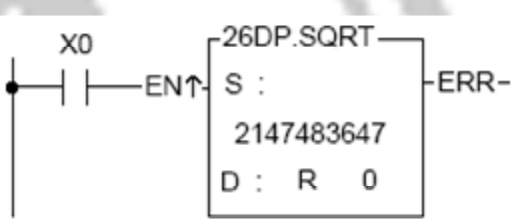


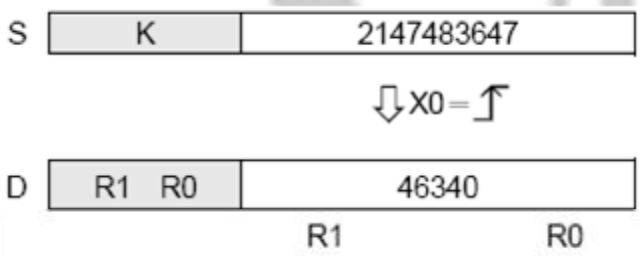
Function 26.SQUARE ROOT

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، خروجی **"ERR"** فعال شده و تابع اجرا نمی شود.

مثال:



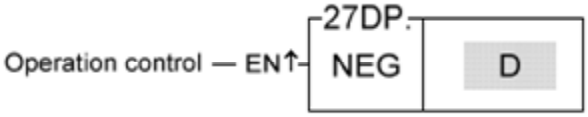


$$\sqrt{2147483647} = 46340.95$$

↑

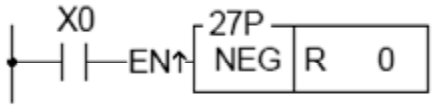
rounding off

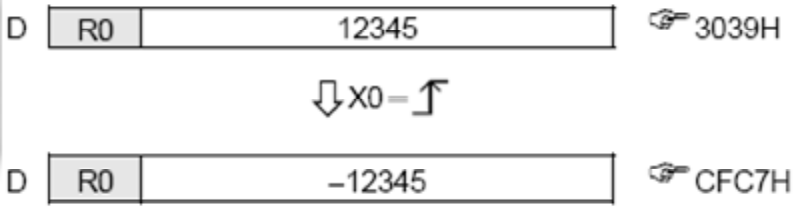
Function 27.NEGATION



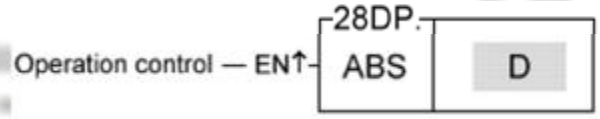
هرگاه "EN" از 0 به 1 تغییر کند، مقدار D منفی شده و دوباره در همان رجیستر ریخته می شود.

مثال:



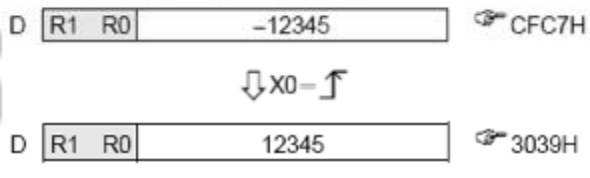


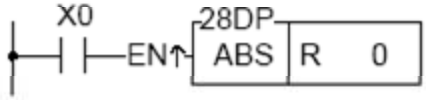
Function 28.ABSOLUTE



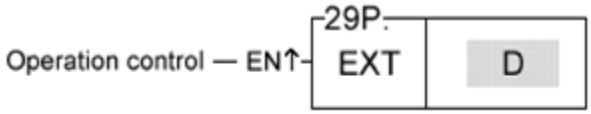
این تابع، قدر مطلق مقدار موجود در D را محاسبه و دوباره در D می ریزد.

مثال:



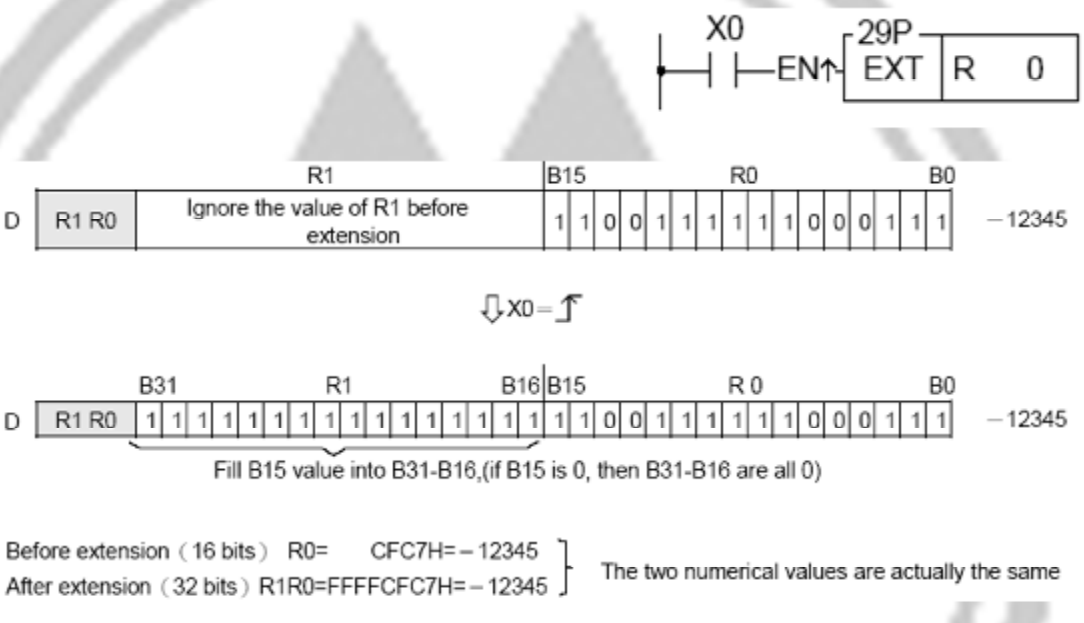


Function 29.SIGN EXTENSION



د ر D یک مقدار 16 بیتی قرار دارد ، با فعال شدن "EN" همین مقدار 32 بیتی می شود. (پس از اجرای این دستور ، رجیستر D+1 نیز. اشغال می شود)

مثال:

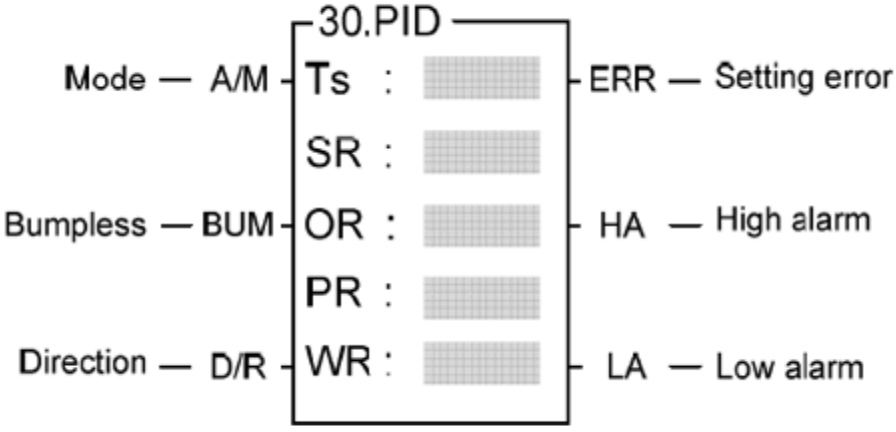


Function 30.PID GENERAL

کنترل PID GENERAL برای کنترل سیستم های عمومی استفاده می شود

ورودی این سیستم از هر ریجیستری می تواند باشد بنابراین از این کنترل می توان به تعداد نامحدود استفاده کرد ولی از این فانکشن فقط می توان یک حلقه PID را تحت کنترل داشت .

Ladder symbol



Range / Operand	HR R0 — R3839	ROR R5000 — R8071	DR D0 — D4095	K
Ts	○	○	○	1～3000
SR	○	○*	○	
OR	○	○*	○	
PR	○	○*	○	
WR	○	○*	○	

Ts: زمان رفرش کردن تابع(از 1 تا 3000 می باشد و هر واحد برابر 0.01 است ، یعنی این تابع را می توان در جاهایی در هر 30 ثانیه فعال کرد)

SR : آدرس شروع مربوط به ریجیسترهای تنظیمات تابع

SR+0	بارگذاری توسط PID
SR+1	Setpoint
SR+2	High Alarm
SR+3	Low Alarm
SR+4	بالاترین مقداری که سنسور می تواند حس کند

SR+5	کمترین مقداری که سنسور می تواند حس کند
SR+6	ریجیستر ورودی حلقه (اگر D4004=0 ورودی تا مقدار 4096(12bit) می باشد و اگر D4004=1 ورودی تا 16383(14bit) می تواند باشد.)
SR+7	مقدار Offset ورودی آنالوگ (برای مثال اگر از سنسور 4 تا 20 میلی آمپر استفاده کنیم ، مقدار این ریجیستر باید 3276 باشد)

OR: آدرس ریجیستر خروجی آنالوگ کنترل PID

PR: پارامترهای کنترل PID

$$Mn = [(D4005/Pb) \times En] + \sum_{0}^{n} [(D4005/Pb) \times Ti \times Ts \times En] - [(D4005/Pb) \times Td \times (PVn - PVn\text{-}1)/Ts] + Bias$$

Mn : Control output at time "n"

Pb : Proportional band (range : 2~5000, unit 0.1%. Kc (gain) =1000/ Pb)

Ti : Intergal time constant (range : 0~9999 corresponds to 0.00~99.99 Repeats/Minute)

Td : Differential time constant (range : 0~9999 corresponds to 0.00~99.99 Minutes)

PVn : Process value at time "n"

PV n-1 : Process value at time "n"

En :Error at time "n" =set value (SP) – process value at time "n" (PVn)

Ts : Interval time of PID calculation (range: 1~3000, unit : 0.01 S)

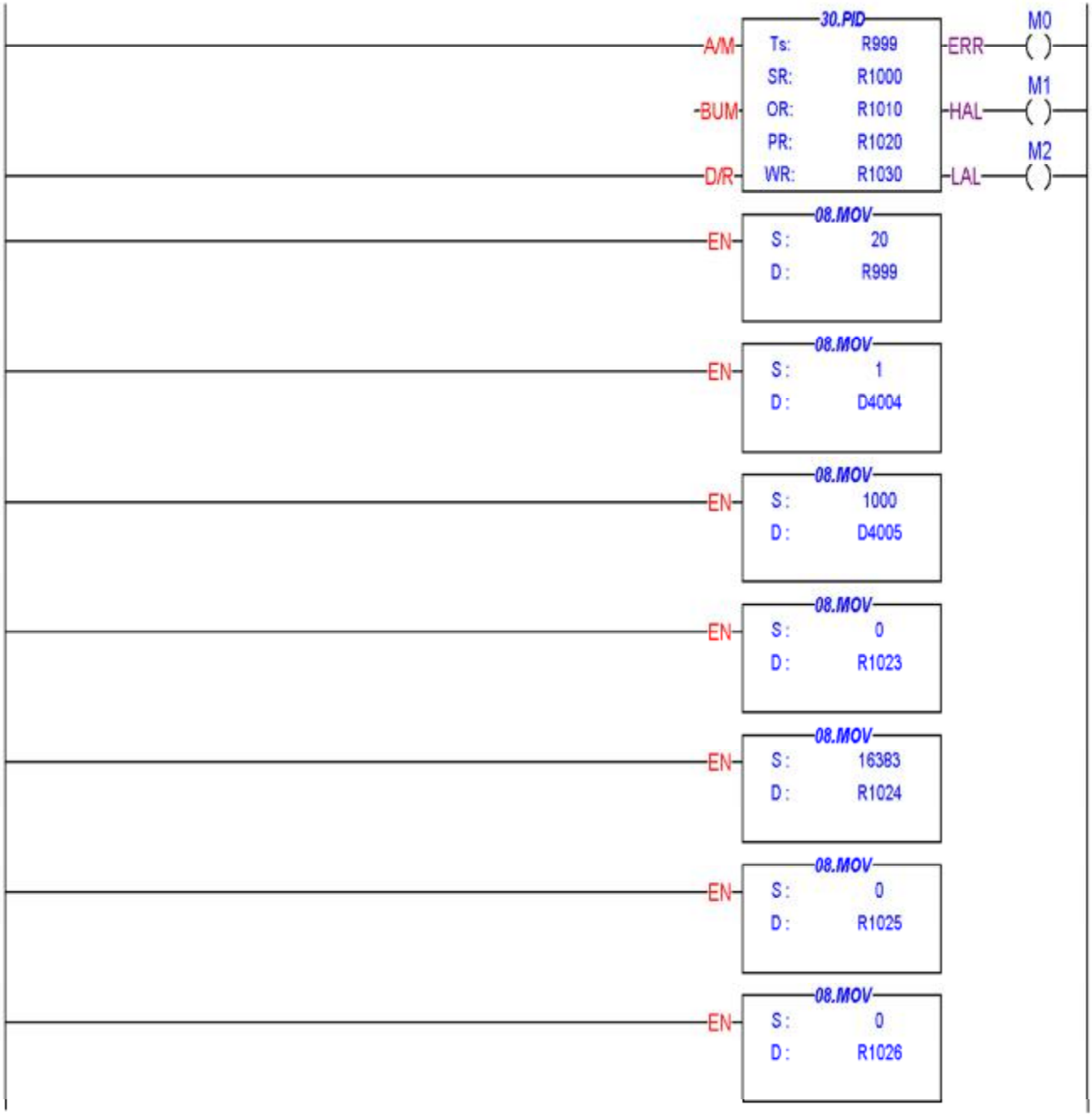
Bias : Control output offset (range: 0~16380)

PR+0	Pb , (2<Pb<5000) (("Gain"Kc = D4005/ Pb (D4005=1000)))
PR+1	Ki 0<Ki<9999 ضریب انتگرال

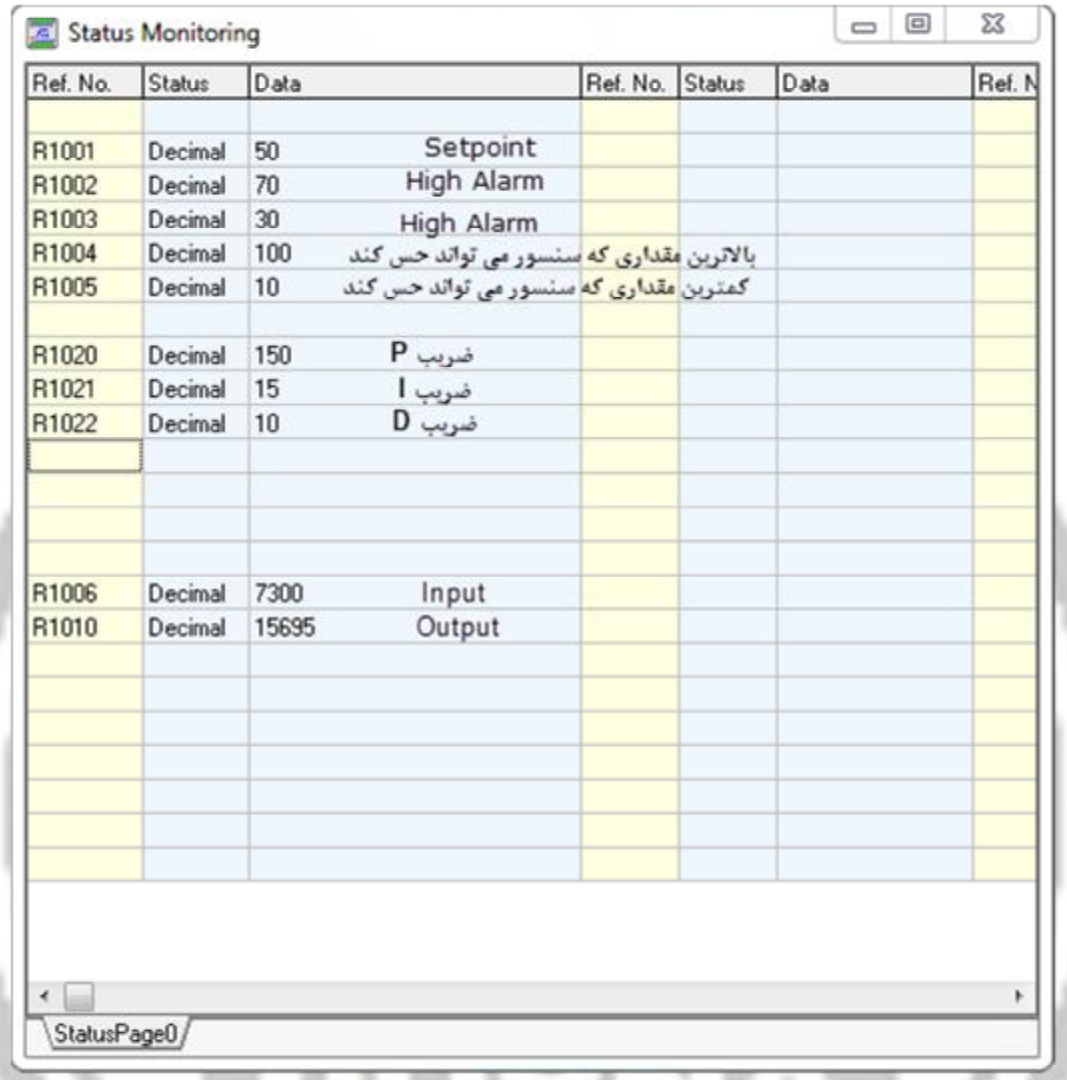
PR+2	ضریب مشتق Td 0<Td<9999
PR+3	Control output offset (range: 0~16380)
PR+4	مصارف خاص ، مقدار 16383 قرار داده شود
PR+5	مصارف خاص ، مقدار 0 قرار داده شود
PR+6	اگر مقدار 0 قرار داده شود بمعنای PID استاندارد و اگر مقدار 1 قرار داده شود بمعنای PI (در مواقعی که سیستم به پایداری نرسد از این گزینه استفاده می شود) می باشد.

WR : شماره رجیستر شروع رجیسترهای داخلی **PID**

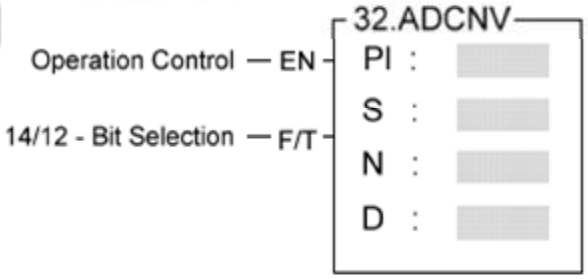
مثال :



پارامترهایی که باید از HMI یا در قسمت تنظیمات به PLC مقدار دهی شوند :



Function 32.ADCNV



این تابع، ورودی آنالوگ بر حسب میلی آمپر را که در محدوده (4~20mA) باشد به عددی در محدوده (0~20mA) تبدیل کرده (0~16383) و به نوعی مانند تبدیل خطی عمل می کند. وقتی ورودی "F/T"، 0 باشد ، عدد تبدیل شده در محدوده (12-bit) 0~ 4095 قرار می گیرد.

و اگر ورودی "F/T"، 1 باشد، عدد تبدیل شده در محدوده (14-bit) 0~16383 قرار می گیرد. وقتی 1="EN" است، تبدیل را از رجیستر S به طول N شروع کرده و نتیجه را در رجیسترهای متوالی با رجیستر شروع D ، به طول N ، ذخیره می کند.

PI=0 : ورودی آنالوگ تک قطبی است (مقدار منفی ندارد)

PI=1 : ورودی آنالوگ دو قطبی است (مقدار منفی دارد)

مثال:

S		D		
R3840	−1229	R500	0	(4 mA)
R3841	409	R501	2047	(12 mA)
R3842	2047	R502	4095	(20 mA)
R3843	−2048	R503	0	(0 mA)
R3844	−2048	R504	0	(0 mA)
R3845	−2048	R505	0	(0 mA)

Function 33.LINEAR CONVERSION

این فانکشن ، عدد ورودی که بین بازه [A,B] می باشد را به عددی خروجی بین بازه [C,D] تبدیل می کند.

در این تابع ، عملگرهای زیر وجود دارند :

Md : (انتخاب حالت کاری (**0** تا **3**)

S : آدرس شروع جدول اطلاعات منبع

Ts : آد رس شروع جدول تبدیل

D : آدرس شروع ذخیره ی نتایج

L : تعداد رجیسترهای تبدیل یافته (1 تا 64)

Md :

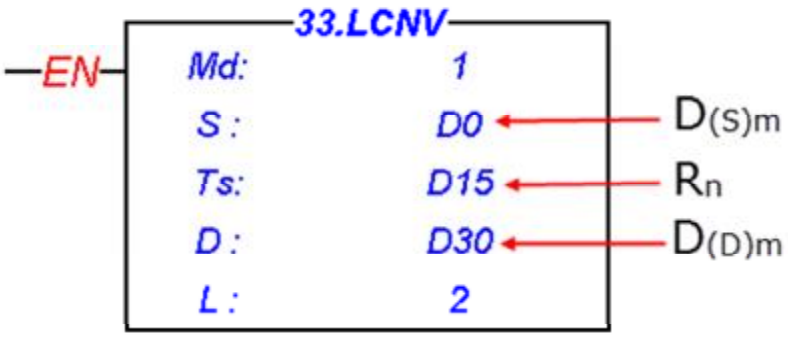
مد صفر : تبدیل عدد ورودی از بازه [A,B] به عدد خروجی بین بازه [C,D]
مد 1 : در این مد می توان چندین تبدیل بازه هابه یکدیگر را با نوشتن فقط یک فانکشن اجرا کرد
مد 2 : تبدیل به تابع خطی درجه 1 D=(A/B)D+C
مد 3 : در این مد می توان چندین تابع خطی درجه 1 را با نوشتن فقط یک فانکشن اجرا کرد

S : محل نوشتن ریجیستر ورودی (وقتی که از مد 1 یا 3 استفاده کنیم چند ریجیستر ورودی یکی بعد از دیگری باید وارد شوند)

Ts : محل وارد کردن رنج بازه ها

D : محل نوشتن ریجیستر خروجی (وقتی که از مد 1 یا 3 استفاده کنیم چند ریجیستر خروجی یکی بعد از دیگری باید وارد شوند)

مثال :



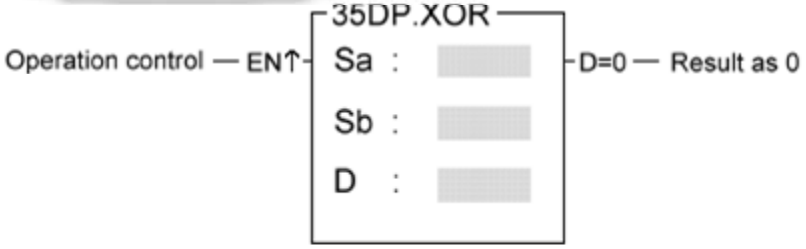
به ازای ورودی های زیر (S)	Ts برای مد صفر	خروجی D
$D_{(S)m}$ →	$[R_n,R_{n+1}] \rightarrow [R_{n+2}, R_{n+3}]$	→ $D_{(D)m}$

به ازای ورودی های زیر (S)	Ts برای مد 1	خروجی D
$D_{(S)m}$ →	$[R_n,R_{n+1}] \rightarrow [R_{n+2}, R_{n+3}]$	→ $D_{(D)m}$
$D_{(S)m+1}$ →	$[R_{n+4},R_{n+5}] \rightarrow [R_{n+6}, R_{n+7}]$	→ $D_{(D)m+1}$
$D_{(S)m+2}$ →	$[R_{n+8},R_{n+9}] \rightarrow [R_{n+10}, R_{n+11}]$	→ $D_{(D)m+2}$

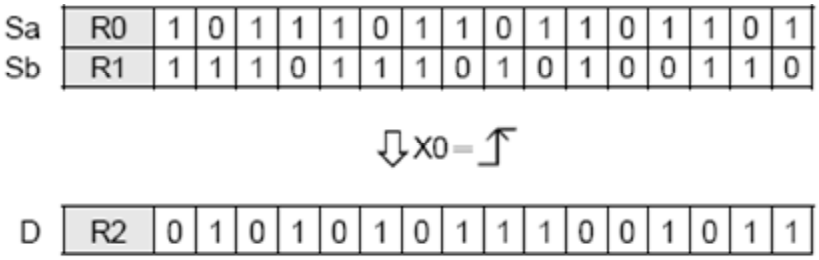
به ازای ورودی های زیر (S)	Ts برای مد 2	خروجی D
$D_{(S)m+1}$ →	$A1=R_n \quad B1=R_{n+1} \quad C1=R_{n+2}$	→ $D_{(D)m+1}=(A1/B1)D+C1$

به ازای ورودی های زیر (S)	Ts برای مد 3	خروجی D
$D_{(S)m}$ →	$A1=R_n \quad B1=R_{n+1} \quad C1=R_{n+2}$	→ $D_{(D)m+1}=(A1/B1)D+C1$
$D_{(S)m+1}$ →	$A2=R_{n+3} \quad B2=R_{n+4} \quad C2=R_{n+5}$	→ $D_{(D)m+2}=(A2/B2)D+C2$
$D_{(S)m+2}$ →	$A3=R_{n+6} \quad B3=R_{n+7} \quad C3=R_{n+8}$	→ $D_{(D)m+3}=(A3/B3)D+C3$

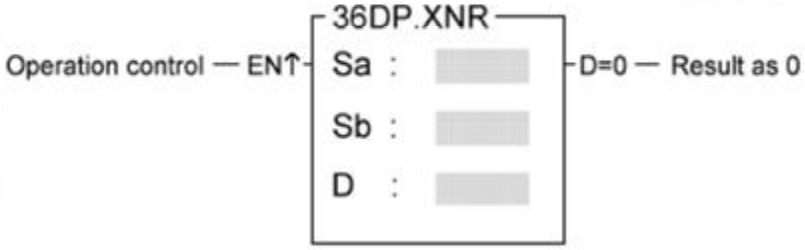
Function 35.EXCLUSIVE OR (XOR)



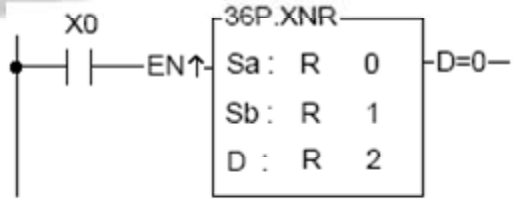
هرگاه "EN" از 0 به 1 تغییر کند،بیت های موجود در Sa و Sb را با هم XOR کرده و نتیجه را در D می ریزد. عملکرد XOR به این صورت است که هرگاه بیت های متناظر Sa و Sb همانند باشند ، بیت متناظر در D ، 0 می شود و هرگاه یکی 1 و آن یکی 0 باشد ، نتیجه در D ، 1 می شود . هرگاه تمام بیت ها ی D ، صفر شود ، خروجی "D=0" فعال می شود .



Function 36.EXCLUSIVE NOR (XNOR)



عملکرد این تابع بر عکس تابع قبل است یعنی دو بیت همانند در Sa و Sb ، متناظر با 1 در D و دو بیت ناهمانند در Sa و Sb ، متناظر با 0 در D می باشد.

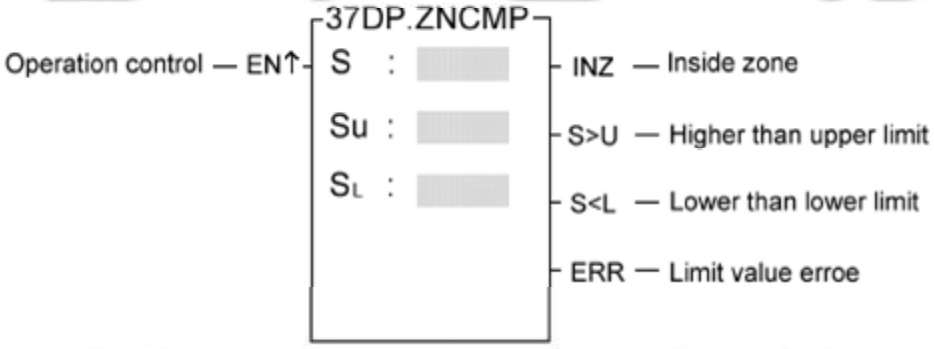


Sa	R0	1	0	1	1	1	0	1	1	0	1	1	0	1	1	0	1
Sb	R1	1	1	1	0	1	1	1	0	1	0	1	0	0	1	1	0

X0

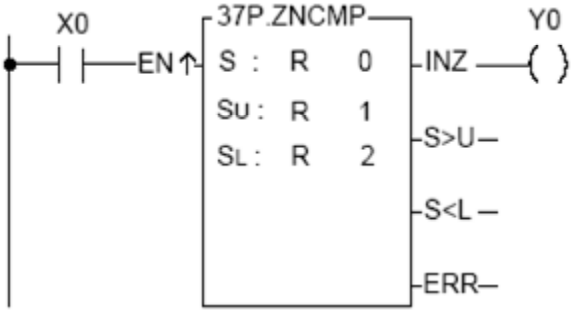
D	R2	1	0	1	0	1	0	1	0	0	0	1	1	0	1	0	0

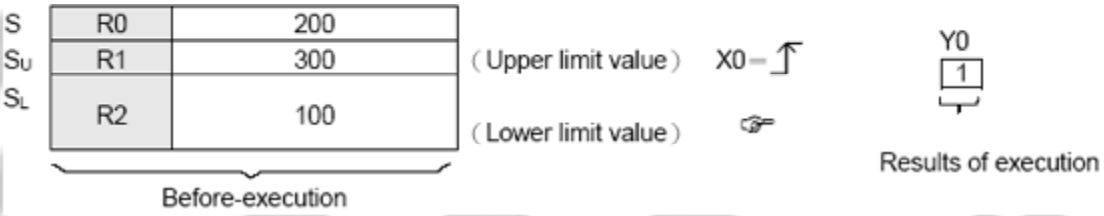
Function 37.ZONE COMPARE



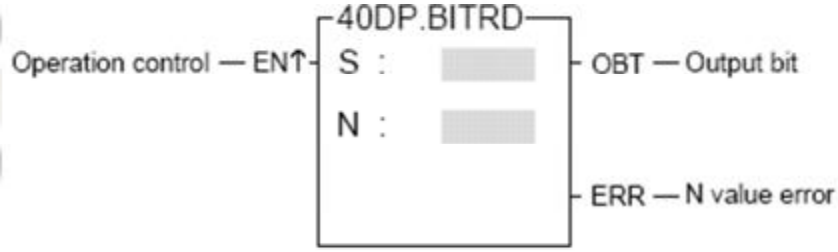
هرگاه "EN" از 0 به 1 تغییر کند، مقدار S با مقدار S_U و S_L مقایسه می شود. اگر بین این دو باشد، خروجی "INZ" فعال می شود . اگر بزرگتر از S_U باشد ، خروجی "S>U" فعال می شود . اگر کوچکتر از S_L باشد ، خروجی "S<L" فعال می شود. اگر $S_L > S_U$ باشد ،ERROR داده خواهد شد.

مثال:





Function 40.BIT READ



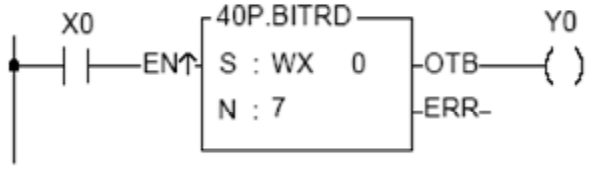
داده های مورد نظر ، در S ریخته می شود. هرگاه "EN" از 0 به 1 تغییر کند: بیت N اُم از داده ی S در خروجی "OBT" قرار می گیرد.

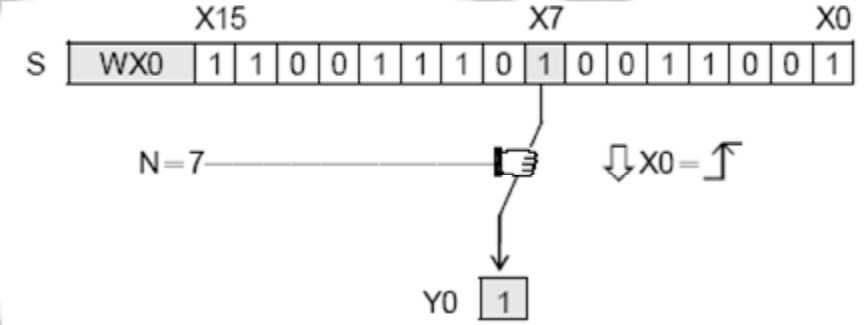
اگر داده ی S ، 16 بیتی باشد : N:0~15

اگر داده ی S ، 32 بیتی باشد : N:0~31

در غیر این صورت error می دهد.

مثال:





Function 41.BIT WRITE

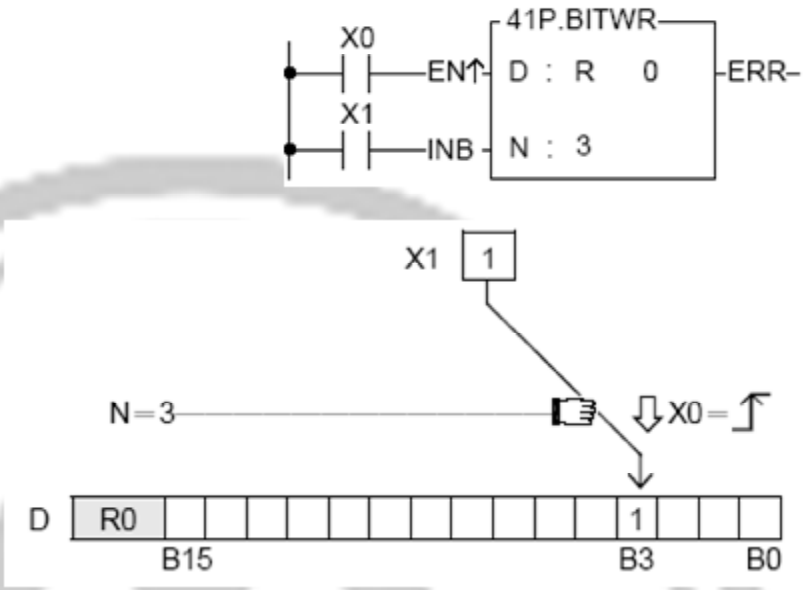
هرگاه "EN" از 0 به 1 تغییر کند: مقدار بیت ورودی "INB" در بیت N اُم رجیستر D ریخته می شود.

اگر عملوند D ، 16 بیتی باشد : N:0~15

اگر عملوند D ، 32 بیتی باشد : N:0~31

در غیر این صورت error می دهد.

مثال:

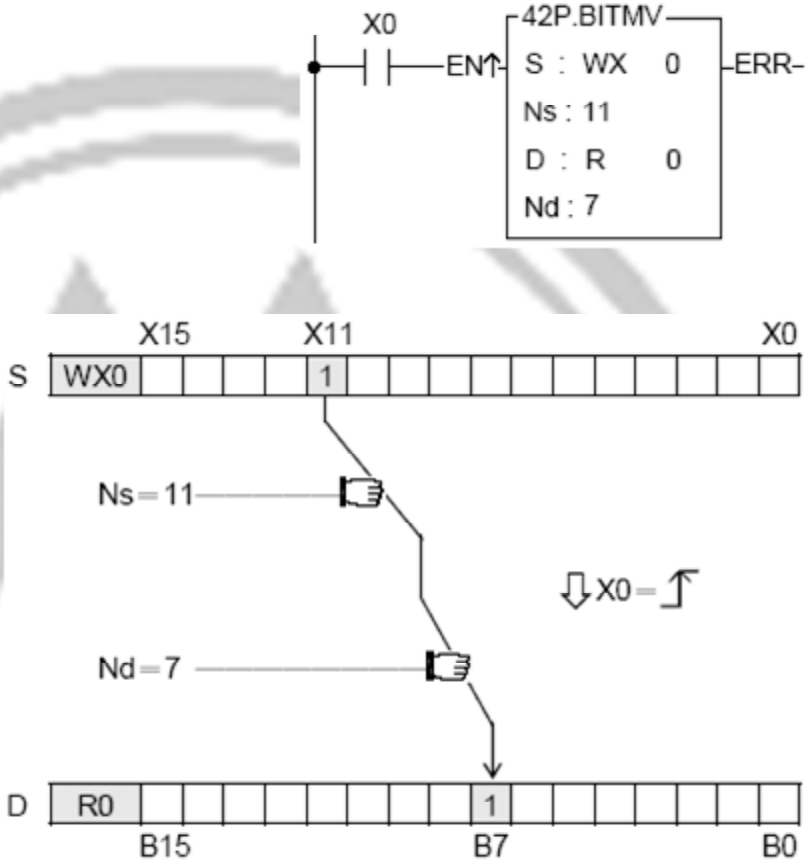


تمام بیت ها غیر از **B3** دست نخورده باقی می مانند.

Function 42.BIT MOVE

هرگاه "**EN**" از **0** به **1** تغییر کند: بیت **Ns** اُم از رجیستر **S** به بیت **Nd** اُم از رجیستر **D** ، منتقل می شود. هرگاه مقادیر **Ns** و **Nd** متناسب با مقادیر **S** و**D** نباشد، **error** فعال می شود.

مثال:

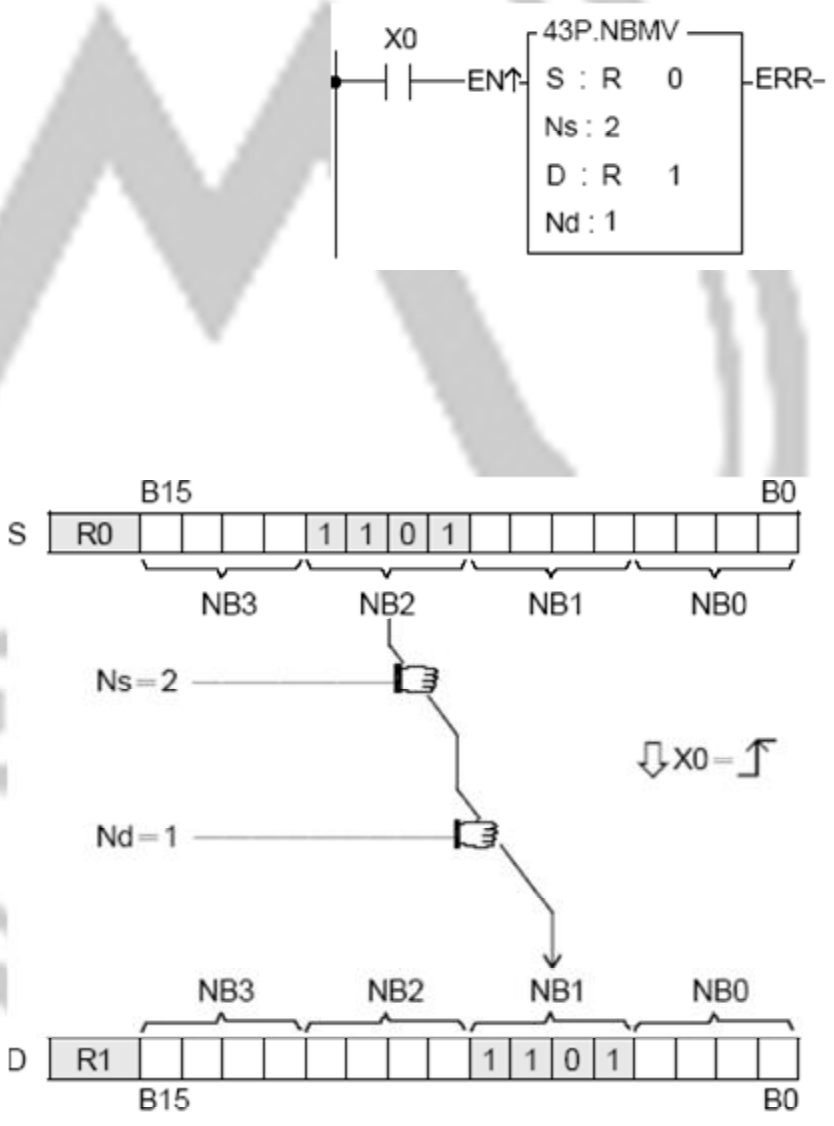


Function 43.NIBBLE MOVE

هرگاه "EN" از 0 به 1 تغییر کند: نیبل Ns ام از رجیستر S به نیبل Nd ام از رجیستر D منتقل می شود .

*نیبل **(Nibble)**: 4 بیت متوالی از یک رجیستر*

مثال:

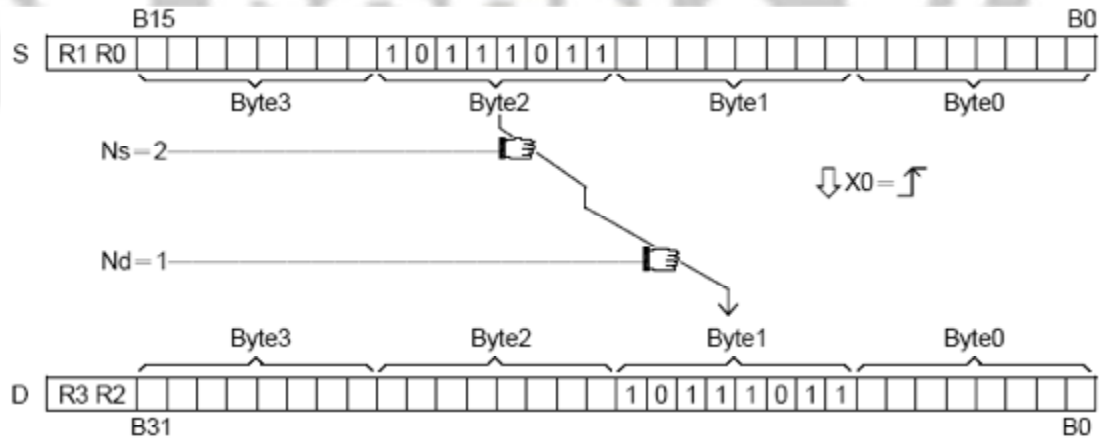


Function 44.BYTE MOVE

هرگاه "EN" از 0 به 1 تغییر کند : بایت Ns اُم از رجیستر S به بایت Nd اُم از رجیستر D منتقل می شود.

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

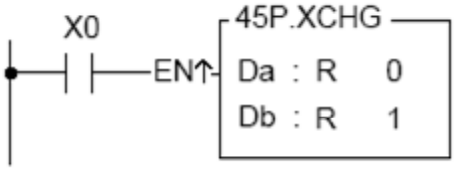
مثال:



Function 45.EXCHANGE

هرگاه "EN" از 0 به 1 تغییر کند: مقادیر رجیستر Da و Db با هم عوض می شوند.

مثال:

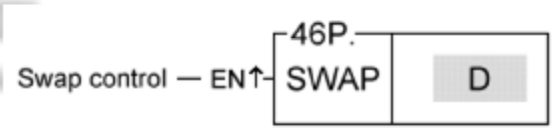


		B15															B0
Da	R0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Db	R1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

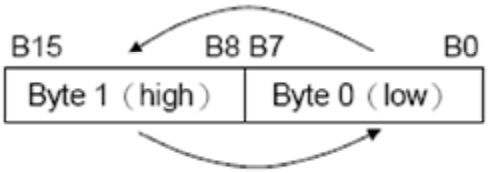
X0 ↓↑

		B15															B0
Da	R0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Db	R1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

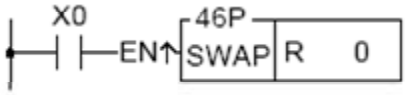
Function 46.BYTE SWAP

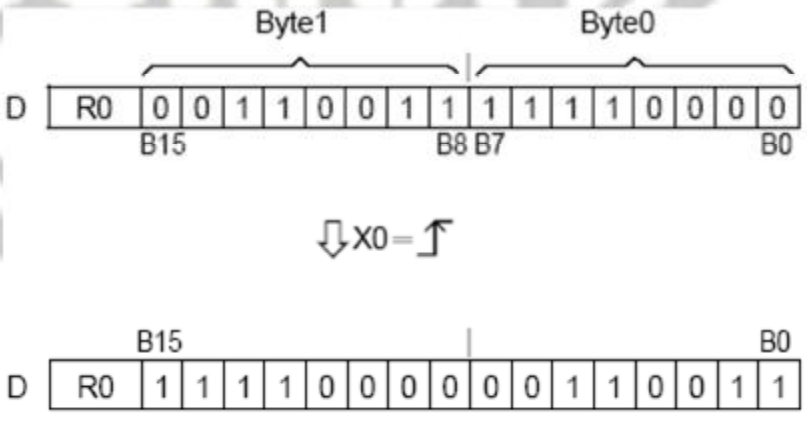


در D یک رجیستر 16 بیتی قرار می گیرد که با فعال شدن "EN"، بایت بالا (High Byte) و بایت پایین (Low Byte) آن با هم عوض می شوند.

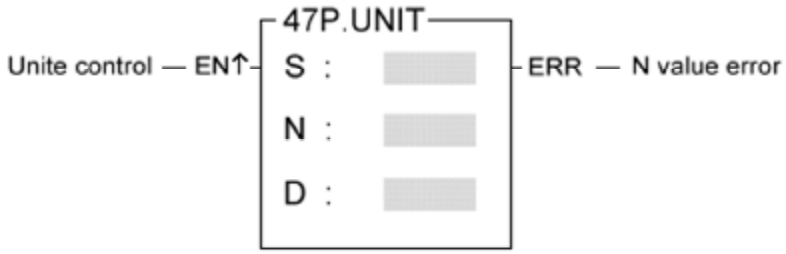


مثال:





Function 47.NIBBLE UNITE

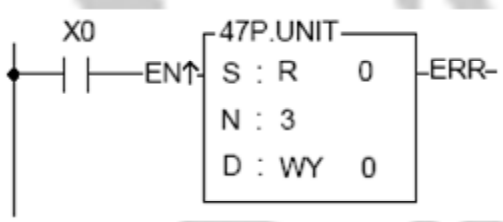


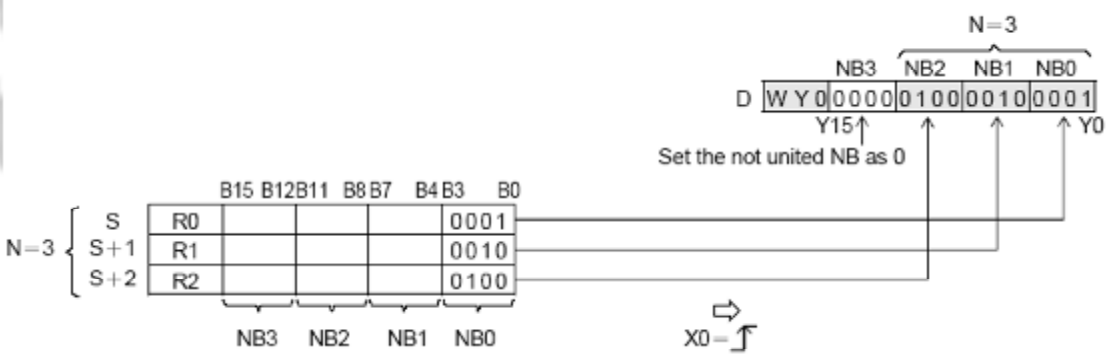
در **S** اولین رجیستر **16** بیتی مورد نظر قرار می گیرد . این تابع نیبل های پایین **N** تعداد از رجیستر ها را به ترتیب در رجیستر **D** قرار می دهد.

***نیبل (Nibble)**: 4 بیت متوالی از یک رجیستر ، (نیبل پایین: 4 بیت کم ارزش رجیستر)*

اگر **N** فرد باشد، باقی رجیستر **D** با **0** پر می شود. مقدار **N** باید **4~1** باشد ، در غیر این صورت **error** داده می شود.

مثال:





Function 48.NIBBLE DISTRIBUTE

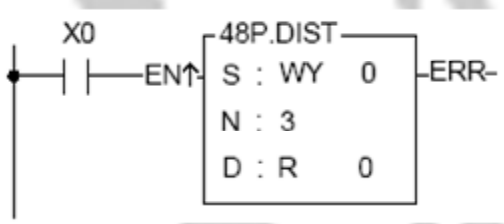
در **S** یک رجیستر**16**بیتی قرار می گیرد که به ترتیب **N** تعداد از نیبل های آن به رجیستر **D+1،D** و... منتقل می شود.

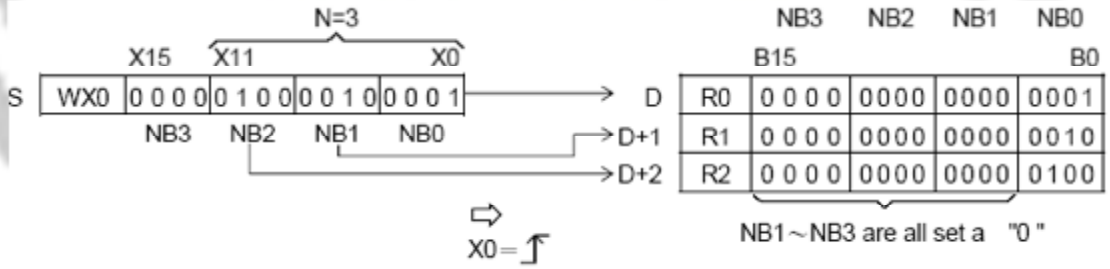
این نیبل ها ،نیبل پایین رجیسترهای **D+1،D** و... را اشغال می کنند و باقی فضای این رجیسترهای با **0** پر می شود.

نیبل ***(Nibble)****: 4 بیت متوالی از یک رجیستر ، (نیبل پایین: 4 بیت کم ارزش رجیستر)*

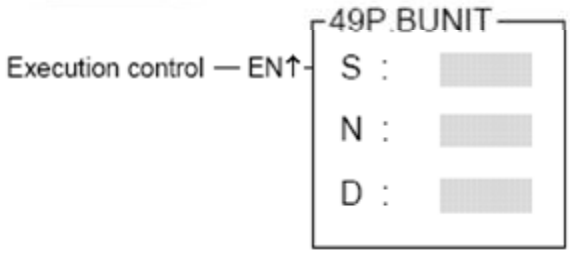
اگر **N:1~4** نباشد ، **error** فعال می شود.

مثال:



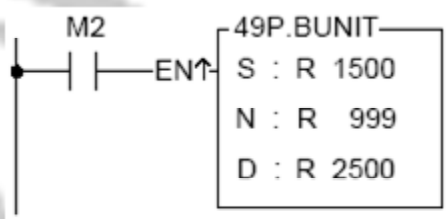


Function 49.BITE UNITE



درS اولین رجیستر از رجیستر های مورد نظر قرار می گیرد. این تابع،بیت ها ی پایین N (Low Byte) تعداد از رجیستر های مشخص شده در S را به ترتیب در رجیستر های D،D+1،... قرار می دهد.

مثال:



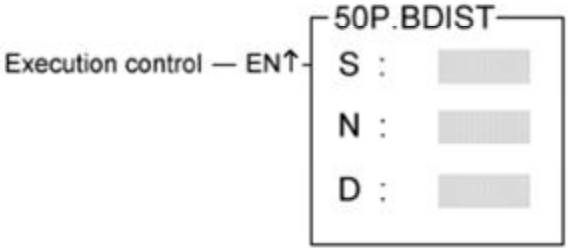
S

	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

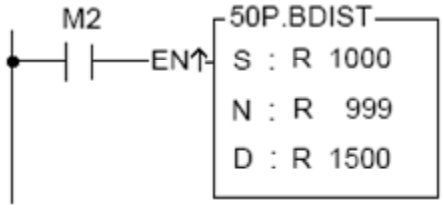
	High Byte	Low Byte
R2500	Byte-0	Byte-1
R2501	Byte-2	Byte-3
R2502	Byte-4	Byte-5
R2503	Byte-6	Byte-7
R2504	Byte-8	Byte-9

Function 50.BYTE DISTRIBUTE



درS اولین رجیستر از رجیستر های مورد نظر قرار می گیرد. این تابع، بایت ها ی مبدا را (N تعداد) به بایت های پایین رجیستر D،D+1،... منتقل می کند.

مثال:



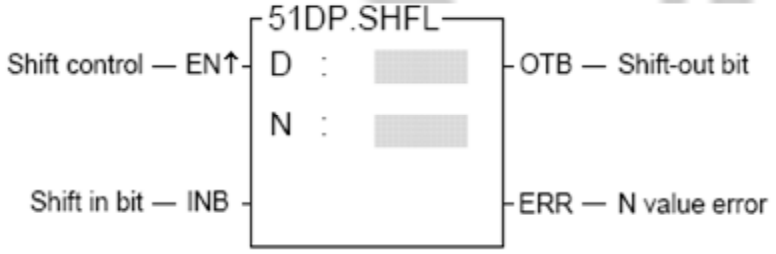
S

	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

Function 51.SHIFT LEFT



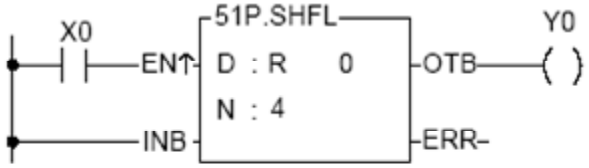
در D رجیستری که قرار است شیفت داده شود قرار می گیرد. N ، تعداد شیفت به چپ را مشخص می کند.جای بیت شیفت داده شده (کم ارزش ترین بیت) را مقدار ورودی "INB" پر می کند و بیت N اُم بالای رجیستر ، پس از شیفت ، به خروجی "OTB" فرستاده می شود .

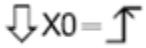
اگر رجیستر مورد نظر 16 بیتی باشد : N:1~16

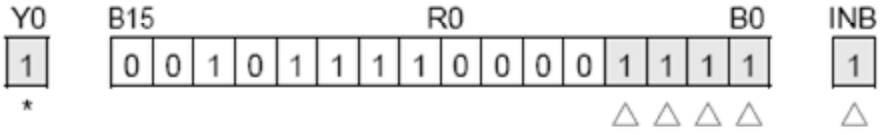
اگر رجیستر مورد نظر 32 بیتی باشد : N:1~32

در غیر این صورت error می دهد.

مثال:







Function 52.SHIFT RIGHT

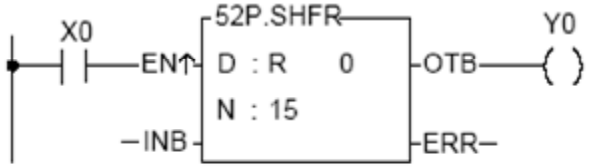
در D رجیستری که قرار است شیفت داده شود قرار می گیرد. N ،تعداد شیفت به راست را مشخص می کند. جای بیت شیفت داده شده (با ارزش ترین بیت) را مقدار ورودی "INB" پر می کند و بیت N اُم پایین رجیستر ، پس از شیفت ، به خروجی "OTB" فرستاده می شود .

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

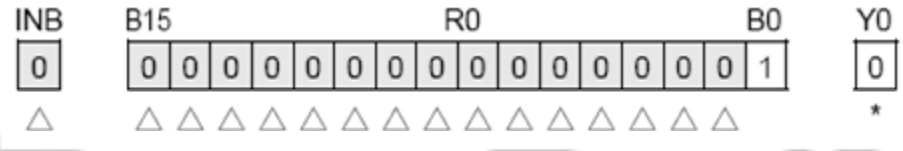
اگر رجیستر مورد نظر 32 بیتی باشد : N:1~32

در غیر این صورت error می دهد.

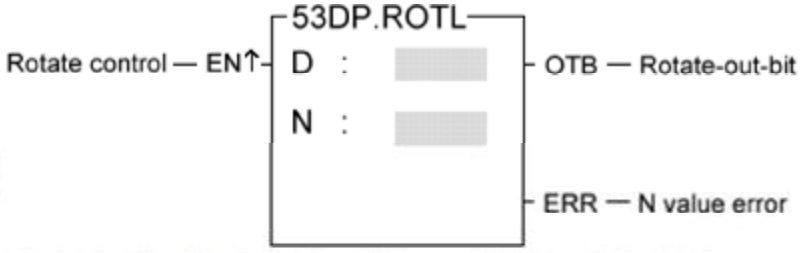
مثال:



⇩X0=⇧



Function 53.ROTATE LEFT



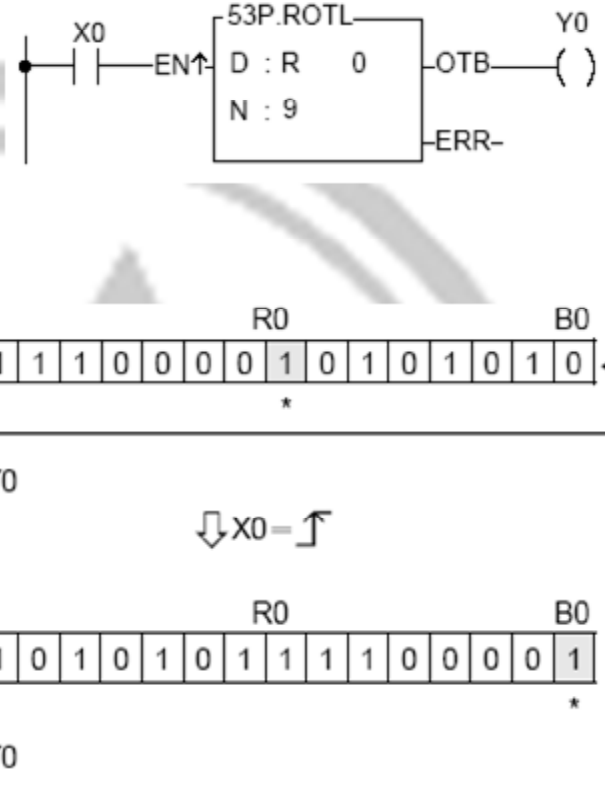
با این تابع یک چرخش به سمت چپ در محل بیت ها انجام می گیرد و چپ ترین بیت (با ارزشترین) به راست ترین بیت (کم ارزشترین) منتقل می شود و یک کپی از آن بیت به خروجی "OTB" نیز می رود.

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

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

در غیر این صورت error می دهد.

مثال:



Function 54.ROTATE RIGHT

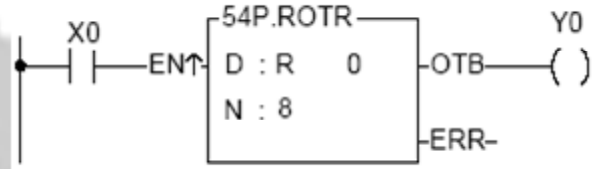
با این تابع یک چرخش به سمت راست در محل بیت ها انجام می گیرد و راست ترین بیت (کم ارزشترین) به چپ ترین بیت (با ارزشترین) منتقل می شود و یک کپی از آن بیت به خروجی "OTB" نیز می رود.

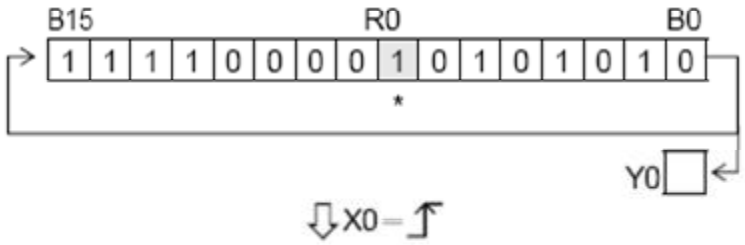
اگر رجیستر مورد نظر 16 بیتی باشد : N:1~16

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

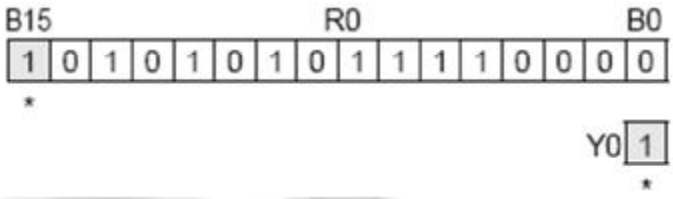
در غیر این صورت error می دهد.

مثال:





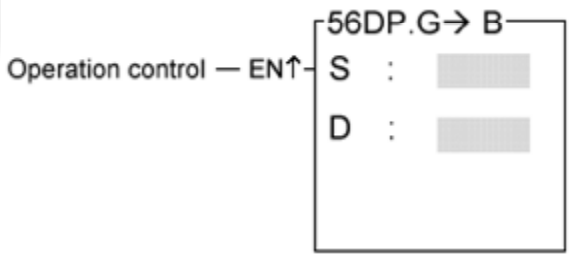
X0 ⇩⇧



Function 55.BINARY TO GRAY

هرگاه "EN" از 0 به 1 تغییر کند: مقدار باینری موجود در رجیستر S به صورت کد گری،کدبندی شده و نتیجه در D ریخته می شود.

Function 56. GRAY TO BINARY



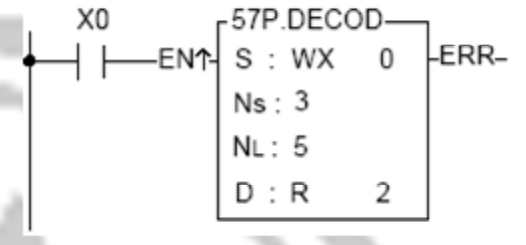
هرگاه "EN" از 0 به 1 تغییر کند: مقدار گری موجود در رجیستر S به صورت باینری،کدبندی شده و نتیجه در D ریخته می شود.

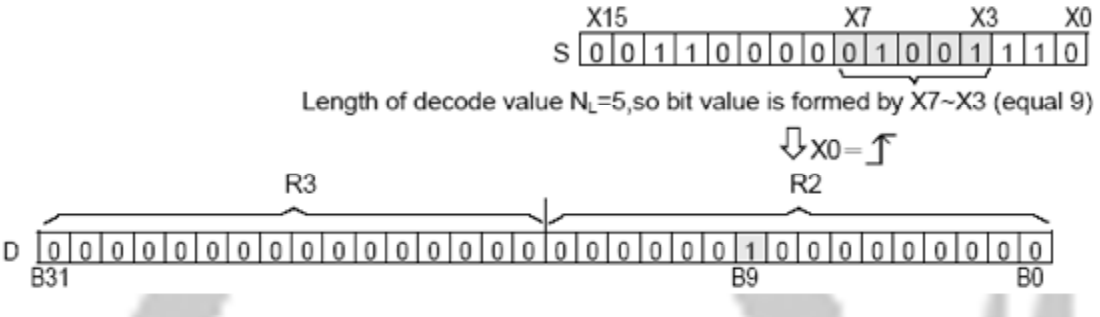
Function 57.DECODE

رجیستری که کدگشایی می شود در S ریخته شده و نتیجه در رجیستر D،D+1،... ریخته می شود. طول رجیستر D : 2^{NL} بیت می شود. مقداری که کدگشایی می شود از بیت Ns اُم رجیستر مبدا (S) آغاز شده و طول آن N_L بیت خواهد بود. معادل دسیمال این مقدار (به عنوان مثال 9)باعث فعال شدن بیت 9اُم(B9) از رجیستر مقصد (D) خواهد

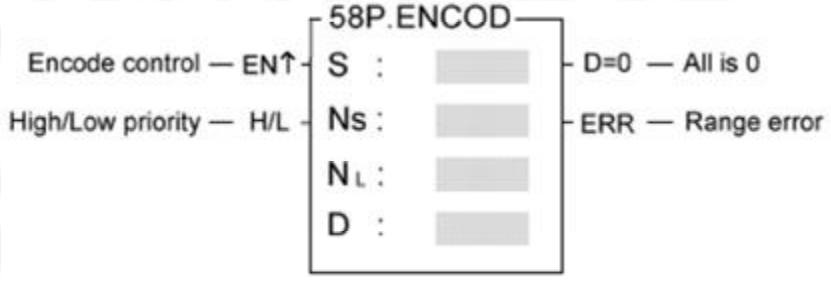
شد. در S یک رجیستر 16 بیتی ریخته می شود پس اگر مقدار Ns یا N_L متناسب با این محدوده نباشد، error می دهد.

مثال:





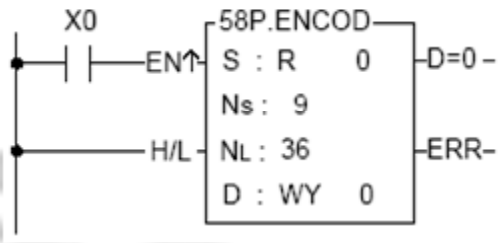
Function 58.ENCODE

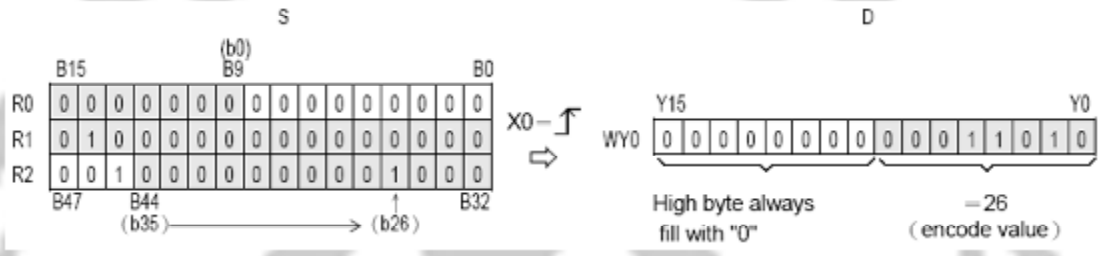


اولین رجیستراز رجیستر های مورد نظر برای کدگذاری در S و مقدار نهایی کدگذاری شده در D ریخته می شود.

مقدار کدگذاری از روی بیت (Ns+1)ام از رجیستر S به طول N_L بیت ، تعیین می شود. هرگاه ورودی "H/L"=1 باشد، کدگذاری با اولویت بالا انجام می شود و هرگاه ورودی "H/L"=0 باشد، کدگذاری با اولویت پایین انجام می شود.

مثال:

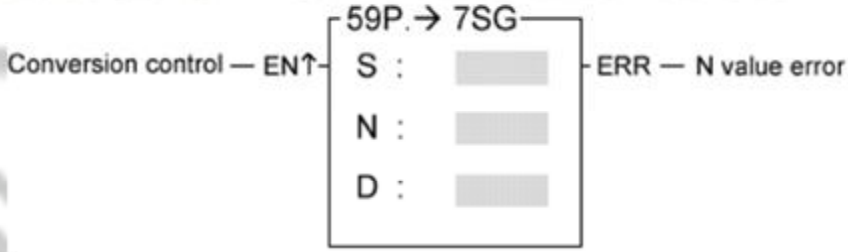




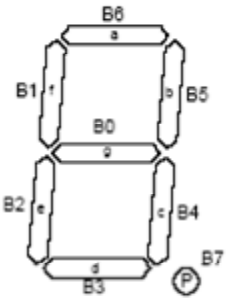
اولین بیت با مقدار 1

برای کدگذاری با اولویت بالا

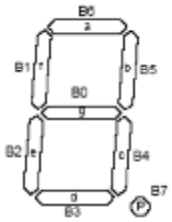
Function 59.7-SEGMENT CONVERSION



در **S** یک رجیستر **16** بیتی قرار می گیرد که هر نیبل(**4** بیت) آن معرف یک عدد هگز(**Hex**) است که از طریق این تبدیل ، به یک عدد **8** بیتی تبدیل می شود(کد **7-seg**) که این عدد در **D** ذخیره می گردد. **N+1** معرف تعداد نیبل هایی است که تبدیل **7-seg** روی آنها انجام می گیرد. رنج موثر **N** ، **0~3** است در غیر این صورت **error** داده خواهد شد . سگمنت های **7-seg** با حروف زیر، علامت گذاری شده اند:

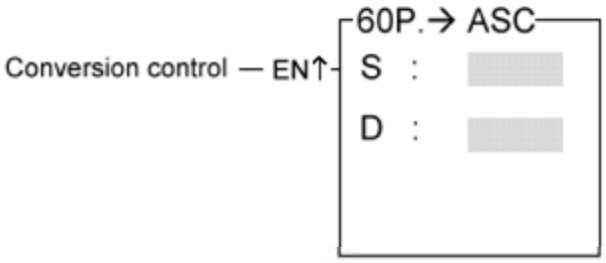


هرگاه بیت **B6** از رجیستر **D**، **1** شود ،معادل نمایش سگمنت **a** از **7-seg** می باشد . هرگاه بیت **B5** از رجیستر **D**، **1** شود ،معادل نمایش سگمنت **b** از **7-seg** می باشد و... ، اگر از **FATEK 7-seg (FBs-7SG)** استفاده می کنید، برای کاربرد نمایش رمزگشایی از طریق **7-seg** ،برای سادگی طراحی می توانید از ترکیب تابع **59** و **84** استفاده کنید.

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	
1	0001		0	0	1	1	0	0	0	0	
2	0010		0	1	1	0	1	1	0	1	
3	0011		0	1	1	1	1	0	0	1	
4	0100		0	0	1	1	0	0	1	1	
5	0101		0	1	0	1	1	0	1	1	
6	0110		0	1	0	1	1	1	1	1	
7	0111		0	1	1	1	0	0	1	0	
8	1000		0	1	1	1	1	1	1	1	
9	1001		0	1	1	1	1	0	1	1	
A	1010		0	1	1	1	0	1	1	1	
B	1011		0	0	0	1	1	1	1	1	
C	1100		0	1	0	0	1	1	1	0	
D	1101		0	0	1	1	1	1	0	1	
E	1110		0	1	0	0	1	1	1	1	
F	1111		0	1	0	0	0	1	1	1	

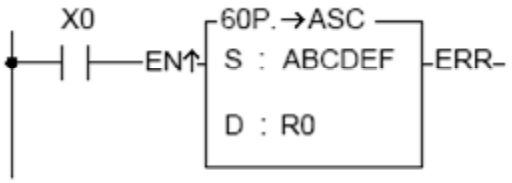
جدول الگوی نمایشی 7-seg

Function 60.ASCII CONVERSION



هرگاه "EN" از 0 به 1 تغییر کند: تبدیل ASCII به روی اعداد وحروف ذخیره شده در S ،انجام می شود و کد ASCII آن حروف یا اعداد در D ذخیره می شود. در S حداکثر 6 رجیستر 16 بیتی ذخیره می شود.(در هر رجیستر ،ASCII×2 قرار می گیرد.) کاربرد این تابع برای دستگاه های نمایشگری است که ورودی را به صورت کد ASCII می پذیرند.

مثال:



S

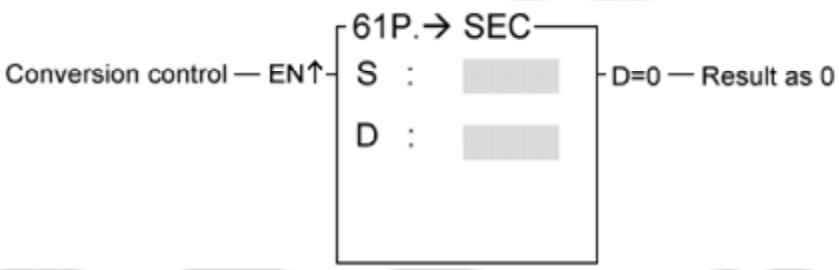
Alphabet
ABCDEF

X0 = ⤴

D

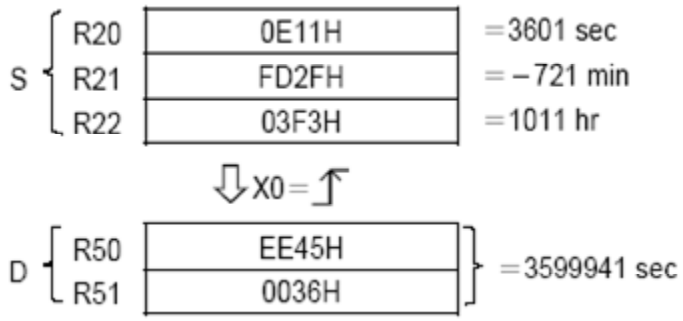
	High Byte	Low Byte
R0	42 (B)	41 (A)
R1	44 (D)	43 (C)
R2	46 (F)	45 (E)

Function 61.H:M:S to SECONDS CONVERSION

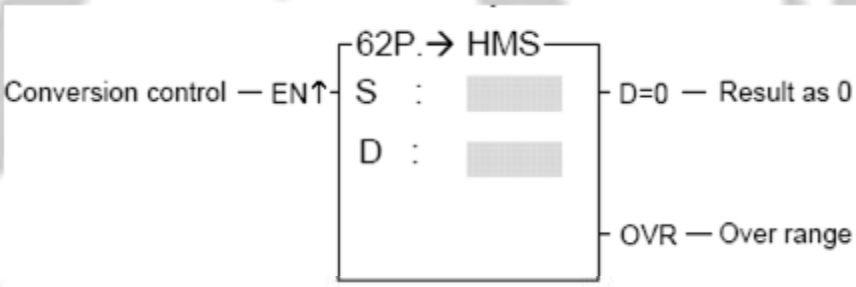


هرگاه "EN" از 0 به 1 تغییر کند: این تابع، مقدار زمانی را که به صورت ساعت ، دقیقه و ثانیه در رجیسترهای S~S+2 ذخیره شده است ، بر حسب ثانیه در رجیستر D ذخیره می کند. اگر نتیجه 0 باشد ،خروجی "D=0" فعال می شود. مقدار ذخیره شده در S بر حسب ثانیه ، S+1 بر حسب دقیقه و S+2 بر حسب ساعت است.

مثال:

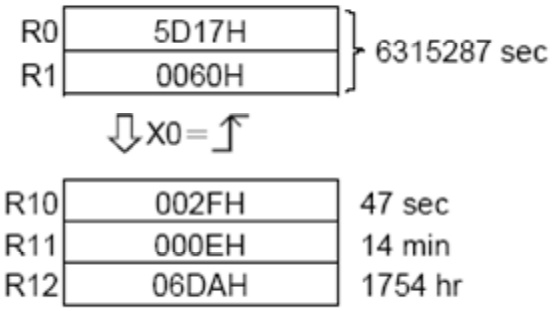


Function 62.SECOND to H:M:S

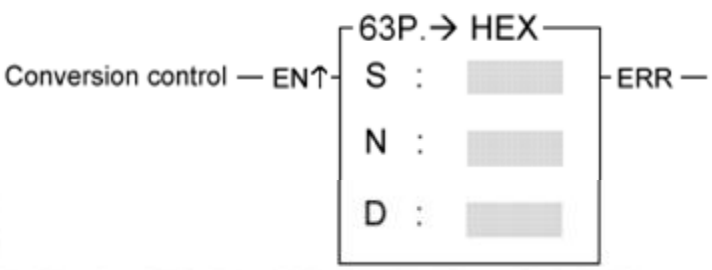


هرگاه "EN" از 0 به 1 تغییر کند: این تابع بر عکس تابع قبل عمل کرده و زمان بر حسب ثانیه را بر حسب ساعت ، دقیقه و ثانیه تبدیل می کند. مقداری که در D ذخیره می شود بر حسب ثانیه ،D+1 بر حسب دقیقه و D+2 بر حسب ساعت است. اگر مقدار موجود در S، 0 باشد ، خروجی "D=0" فعال می شود. مقدار مناسب برای S ، -117968399~117964799 ثانیه می باشد در غیر این صورت خروجی "OVR" (OVER RANGE) فعال می شود.

مثال:

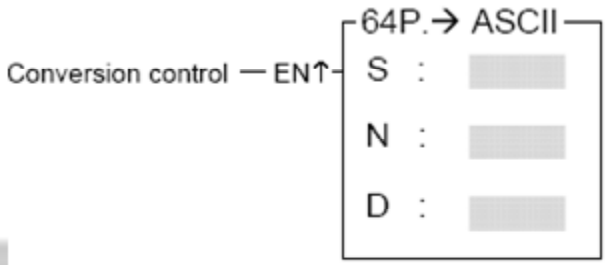


Function 63.ASCII to HEX



هرگاه "EN" از 0 به 1 تغییر کند: به تعداد N 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 آن بتواند کد هگز را پردازش کند . بنابراین اگر مقدار موجود در S ، معادل ASCII کاراکترهای ('0'~'9' و 'A'~'F') نباشد، خروجی "ERR" فعال خواهد شد.

Function 64.HEX toASCII



هرگاه "EN" از 0 به 1 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.

Function 65.LABLE

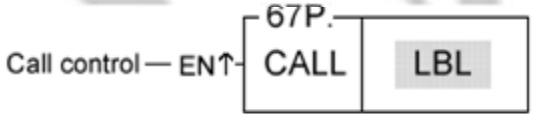
خود این تابع صرفا عملی انجام نمی دهد و به عنوان یک برچسب آدرس ، برای برنامه ها عمل می کند. به عنوان یک نشانگر برای اجرای تابع های CALL،JUMP ، INTERRUPT استفاده می شود.همچنین برای آسان خوانی برنامه نیز می توان به کار برد. در S نام برچسب متشکل از حروف و اعداد نوشته می شود که از 1 تا 6 حرف می تواند داشته باشد. اسامی موجود در جدول زیر کاربری خاص داشته و برای تابع هایINTERRUPT 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.
PSO0I～PSO3I	Labels for the pulse output command finished interrupt service routine.

Function 66. JUMP

هرگاه "EN" از 0 به 1 تغییر کند: برنامه به برچسبی که نام آن در این تابع ،نوشته شده پرش می کند وبرنامه از آنجا ادامه پیدا می کند. این تابع مواقعی کاربرد دارد که قسمتی از برنامه تنها تحت شرایط خاصی اجرا می شود. پرش فقط در برنامه اصلی یا زیربرنامه ها صورت می گیرد و پرش میان برنامه اصلی و زیربرنامه ها انجام نمی گیرد.

Function 67.CALL

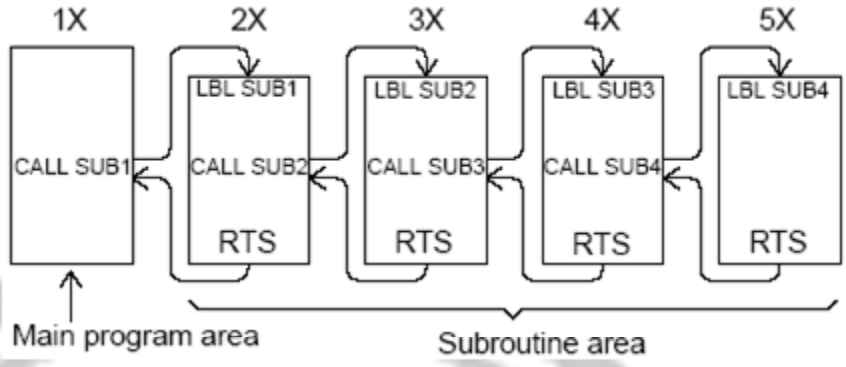


هرگاه "EN" از 0 به 1 تغییر کند:

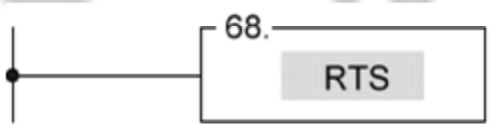
PLC به زیربرنامه ا ی می رود که تابع CALL از طریق LABLE نوشته شده در تابعآن را فراخوانی می کند و به اجرای آن زیربرنامه می پردازد تا زمانی که به تابع RTS بربخورد ،سپس به همانجایی بر می گردد که تابع CALL در ان نوشته شده و از ادامه تابع CALL به اجرای برنامه می پردازد.

تذکر: در انتهای تمام زیربرنامه ها، باید تابع RTS(Return From Subroutine) وجود داشته باشد در غیر این صورت error داده خواهد شد یا CPU خاموش می شود.

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 5 مرحله می تواند انجام بپذیرد.

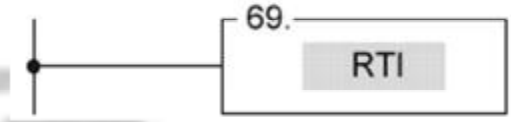


Function 68.RTS



این تابع برای نشان دادن پایان یک زیربرنامه به کار می رود و PLC بعد از دیدن این تابع به انتهای تابع CALLی می رود که از طریق آن ، این زیربرنامه فراخوانی شده است و به اجرای تابع ها بعد از CALL می پردازد.

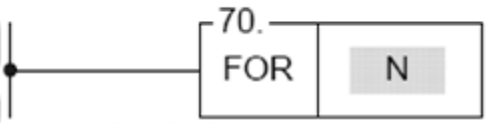
Function 69.RTI



این تابع شبیه تابع RTS است با این فرق که RTS در انتهای زیربرنامه قرار می گیرد در حالی که RTI در انتهای روتین INTERRUPT (وقفه) قرار می گیرد . در مقایسه با CALL که از طریق یک LABLE ، زیربرنامه مورد نظر را اجرا می کند، INTERRUPT مستقیما از طریق سیگنال های سخت افزاری فعال شده و در کار CPU وقفه ایجاد می کند تا به انجام روتین INTERRUPT بپردازد. اگر اینترایتی در حین اجرای روتین یک اینترایت دیگر اتفاق بیافتد روتین

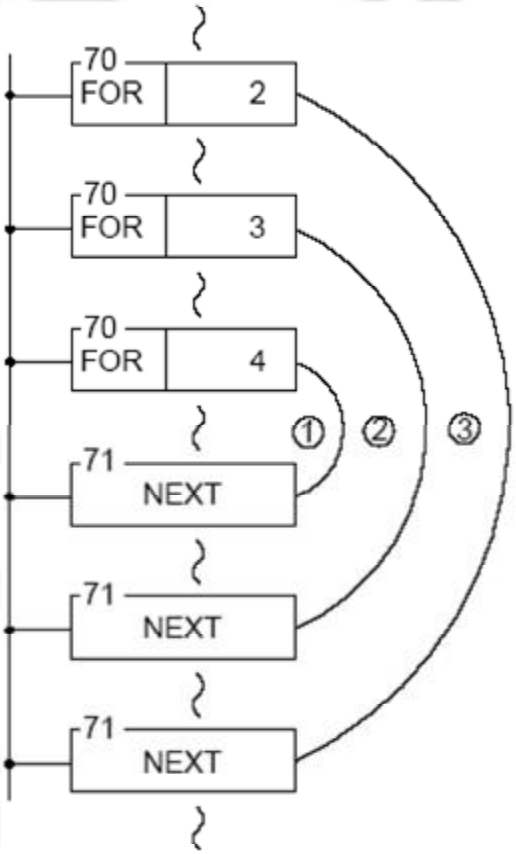
اینتراپتی اجرا خواهد شد که در اولویت بالاتر قرار دارد. توجه داشته باشید که حتما از RTI درانتهای روتین اینتراپت استفاده کنید.

Function 70.FOR

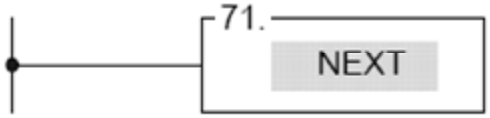


این تابع به همراه تابع NEXT، تشکیل یک حلقه را می دهد که برنامه ی میان FOR وNEXT مربوطه ،N بار اجرا می شود.حلقه های FOR وNEXT دیگری نیز در میان حلقه های FOR و NEXT اولیه می تواند قرار بگیرد.

به مثال توجه کنید:



Function 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.

Function 74.IMMIDIATE I/O

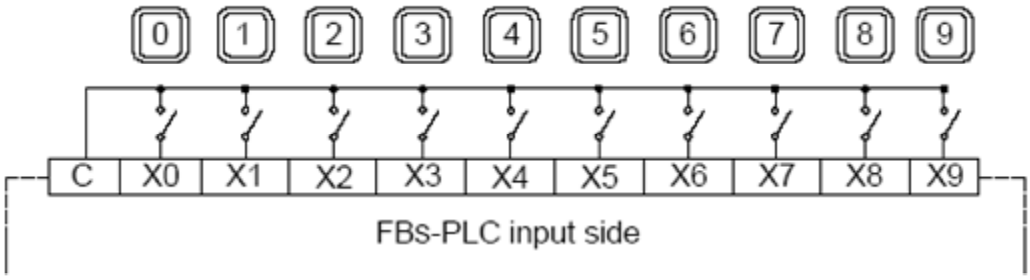
این تابع سیگنال های ورودی و خروجی را فورا ، update می کند. هرگاه "EN" از 0 به 1 تغییر کند: سیگنال های ورودی یا خروجی با آدرس شروع D به تعداد N ،refresh شده و تغییر ایجاد شده در ورودی-خروجی مورد نظر ، بدون نیاز به کامل شدن اسکن برنامه ، در همان لحظه اعمال می شود .

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

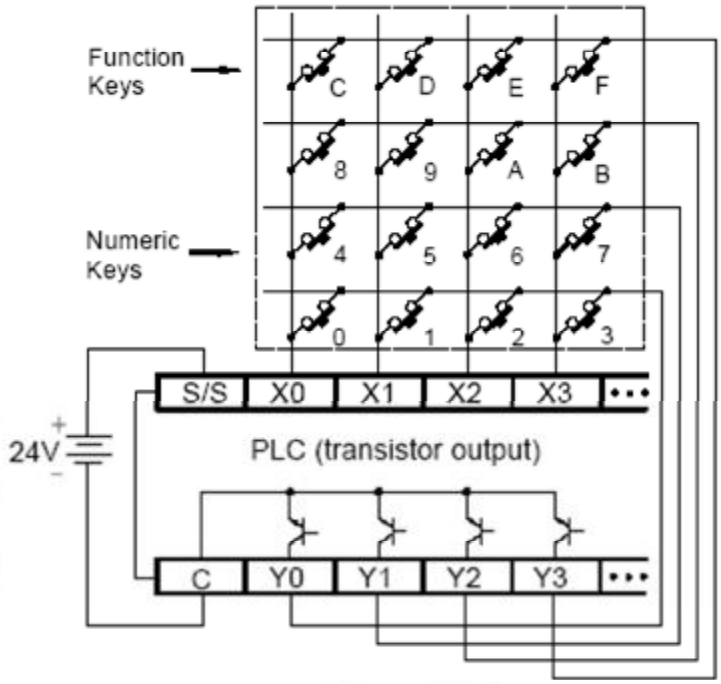
Function 76.DECIMAL-KEY INPUT

این تابع می تواند ورودی 9~0 را توسط کیبورد دریافت کند که این اعداد معادل با IN~IN+9 قرار می گیرند.(IN می تواند از X0 تا X240 باشد) بر اساس ترتیب کلیدهای فشرده شده، یک عدد دهدهی 4 یا 8 رقمی می تواند در رجیستر مشخص شده در D ذخیره شود. اگر رجیستر موجود در D، 16 بیتی باشد ،این عدد دهدهی می تواند 4 رقمی باشد و اگر 32 بیتی باشد ، 8 رقمی خواهد بود. این تابع در صورتی عمل می کند که 1="EN" باشد. هرگاه هریک از کلیدها فشرده شود ، خروجی "KPR" به اندازه ی همان مدت فشرده شدن کلید،1 خواهد ماند. هرگاه هریک از اعداد فشرده شود ، بیت معادل آن عدد که در KL مشخص می شود، 1 خواهد شد و حتی اگر کلید مذکور رها شود بازهم 1 خواهد ماند تا زمانی که عدد دیگری غیر از عدد قبلی فشرده شود؛آنگاه بیت معادل عدد فشرده شده جدید ، در KL، 1 خواهد شد. با فشرده شدن هر عدد ، معادل دهدهی آن در D ذخیره می شود و هرگاه عدد جدیدی فشرده شود ، عدد قبلی به سمت چپ شیفت پیدا می کند.



Function 77.HEX-KEY INPUT

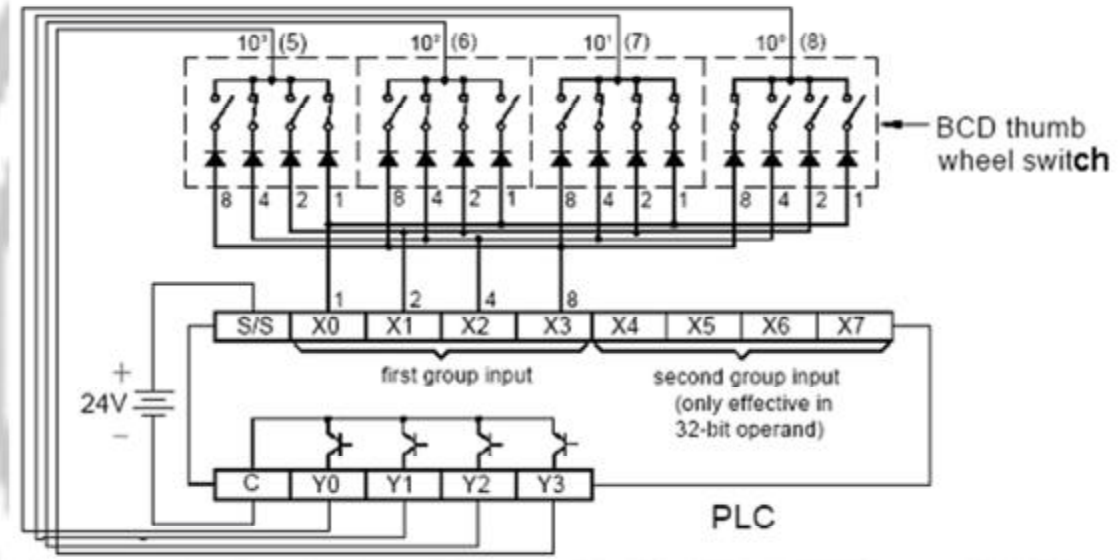
این تابع مشابه تابع قبل است با این تفاوت که علاوه بر اعداد 0~9 ، می تواند 6 ورودی دیگر نیز دریافت کند (A~F) که هرکدام می توانند معادل یک دستورعمل باشند. همچنین اتصال سخت افزاری این تابع با تابع قبل متفاوت است ،4 ورودی تعیین شده PLC به 4 خروجی تعیین شده ، طوری متصل می شوند که اتصال هر کدام از آنها یکی از اعداد را تولید کند.



هرگاه هریک 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 کاربری محاسبات خود تابع است و در جای دیگری نباید از آن استفاده کرد.

78.DIGITAL SWITCH INPUT

هرگاه 1= "EN" بشود، این تابع ، 4 رقم دهدهی را از سوئیچ BCD دستی ، بازخوانی کرده و آنها را در D ذخیره می کند. بیت های هم رقم باید به هم وصل شده و از طریق یک دیود سری بشوند.

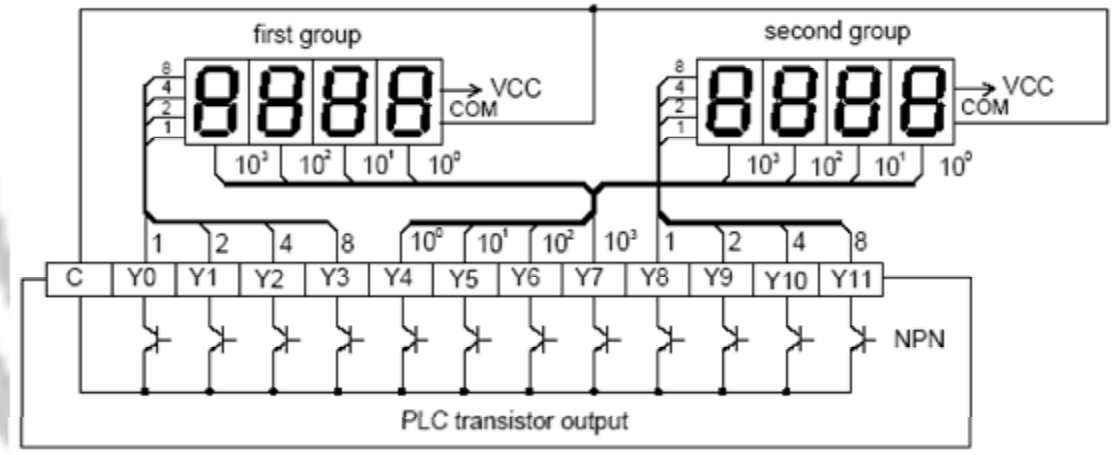


هربار که 4 رقم از سوئیچ ها خوانده شد ، خروجی 1="Dn" می شود. اگر هریک از اعداد خوانده شده در رنج BCD(0~9) نباشد، خروجی "ERR" فعال می شود. PLC مورد استفاده باید دارای خروجی ترانزیستوری باشد.
رجیستر ذخیره شونده در WR برای کاربری محاسبات خود تابع است و در جای دیگری نباید از آن استفاده کرد.

Function 79.7-SEG OUTPUT WITH LATCH

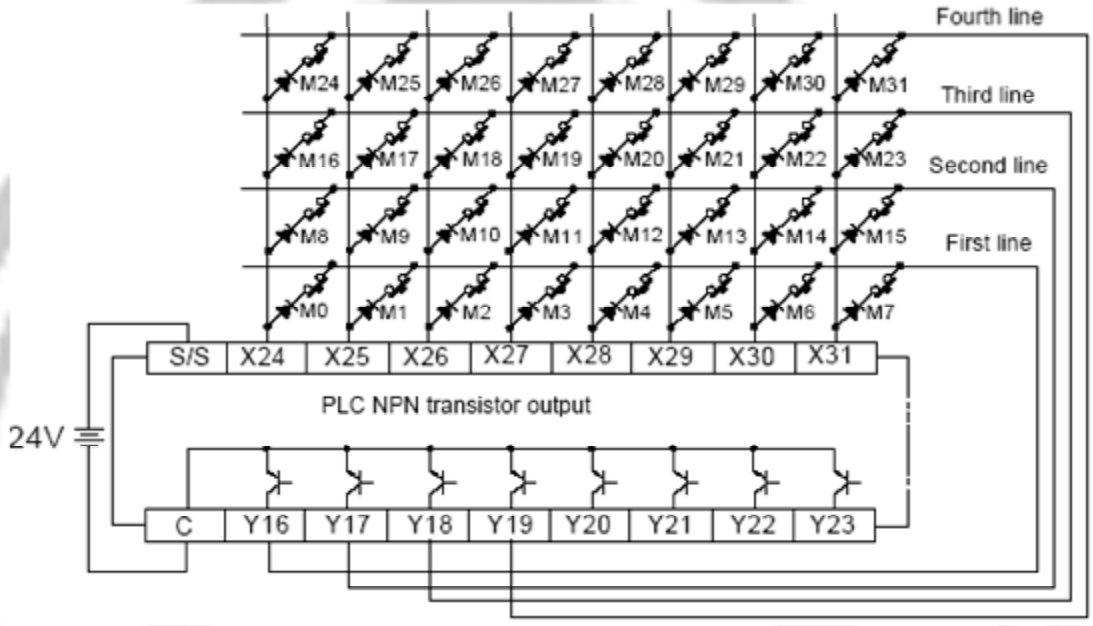
با توجه به شکل بعدی ، هرگاه **1= "EN"**، **4** نیبل رجیستر مشخص شده در **S** ، برای نمایش به **7-SEG**سری اول و **4** نیبل **S+1** ، به **7-SEG** سری دوم منتقل می شوند.

نیبل ***(Nibble)****: 4 بیت متوالی از یک رجیستر* **PLC** مورد استفاده باید دارای خروجی ترانزیستوری باشد.

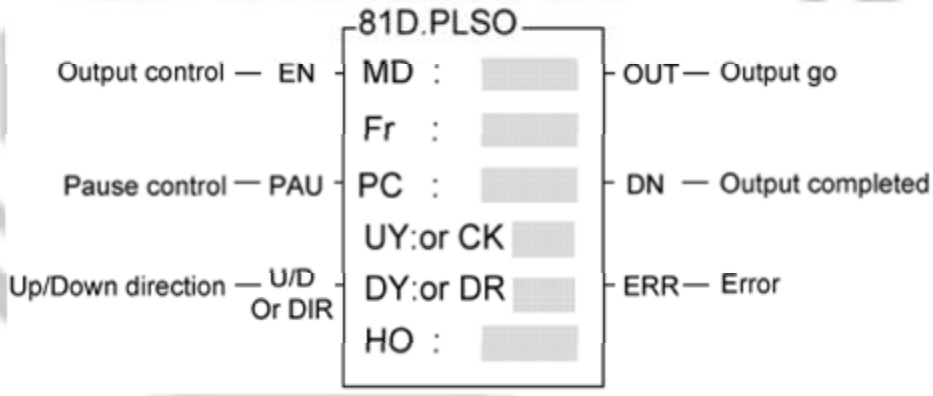


Function 80.MULTIPLEX INPUT

این تابع از متد مالتی پلکس برای خواندن 8×N ورودی استفاده کرده ، 8 ورودی و N خروجی از PLC را اشغال می کند. آدرس شروع 8 ورودی از IN و N خروجی از OT می باشد . در هر اسکن ، یکی از N خروجی ، فعال شده و ردیف مربوط به آن خروجی انتخاب می شود. OT0 مربوط به ردیف اول ،OT1 مربوط به ردیف دوم و ... است.پس از پایان اسکن تمام ردیف ها(Nخط) ،8×N مشخصه خوانده شده در رجیستر D ذخیره شده ،سپس خروجی "DN" یک می شود.(فقط به اندازه یک اسکن ،یک می ماند!)

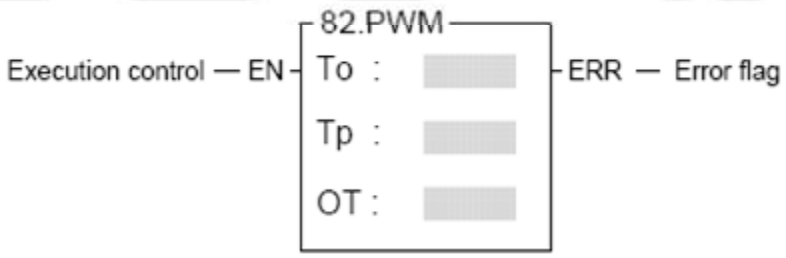


Function 81.PULSE OUTPUT

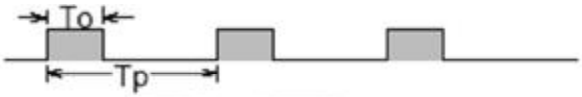


از این تابع برای فرستادن پالس به خروجی استفاده شده و عمده کاربرد آن می تواند راه اندازی استپ موتور با ورودی پالس راست گرد و چپ گرد باشد . از این تابع تنها می توان یکبار در برنامه استفاده نمود و PLC مورد استفاده باید دارای خروجی ترانزیستوری باشد.

Function 82.PULSE WIDTH MODULATION

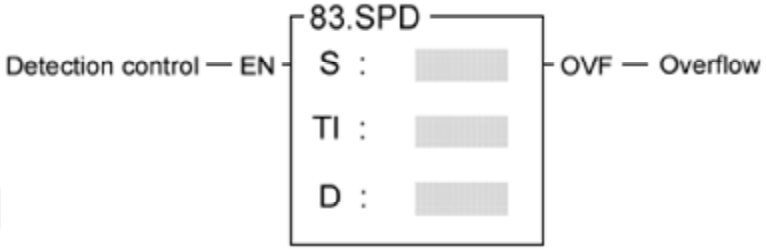


از این تابع برای فرستادن پالس به خروجی با عرض پالس دلخواه استفاده می شود . وقتی 1="EN" است،پالسی به خروجی OT می فرستد که To میلی ثانیه ON است و پریود آن Tp میلی ثانیه می باشد. خروجی مورد استفاده باید ترانزیستوری باشد . حداقل مقدار To، 0 است که در این موقعیت خروجی همیشه به حالت OFF خواهد بود. حداکثر مقدار To، برابر Tp است که در این موقعیت خروجی همیشه به حالت ON خواهد بود.



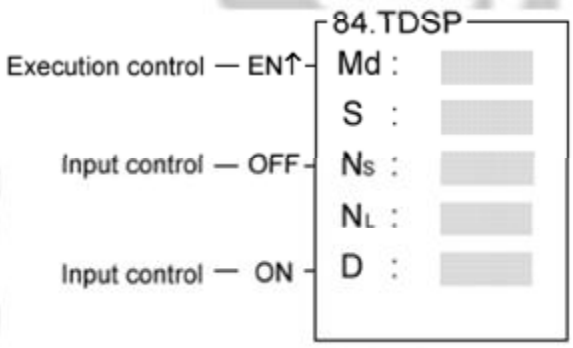
اگر To>Tp بشود ،خروجی "ERR" فعال شده و تابع عمل نخواهد کرد. این تابع می تواند تنها یک بار در برنامه استفاده شود.

Function 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 ورودی های سرعت بالای **PLC** ، به دست آورده 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.

Function 84.16/7-SEG DISPLAY



کاربرد این تابع برای ماژول های **FBs-7SG1** و **FBs-7SG2** بوده و کاراکترهای مختلف را برای نمایش در **16-seg** و **17-seg** آماده می کند. در **S** آدرس شروع کاراکتر هایی که قرار است تبدیل شوند ، ذخیره می شود.

Ns ،مقداری برای اشاره گر تابع است که مشخص می کند کاراکتر ،دقیقاً از کدام بایت شروع می شود.

N_L طول کاراکترمورد نظر را مشخص می کند.

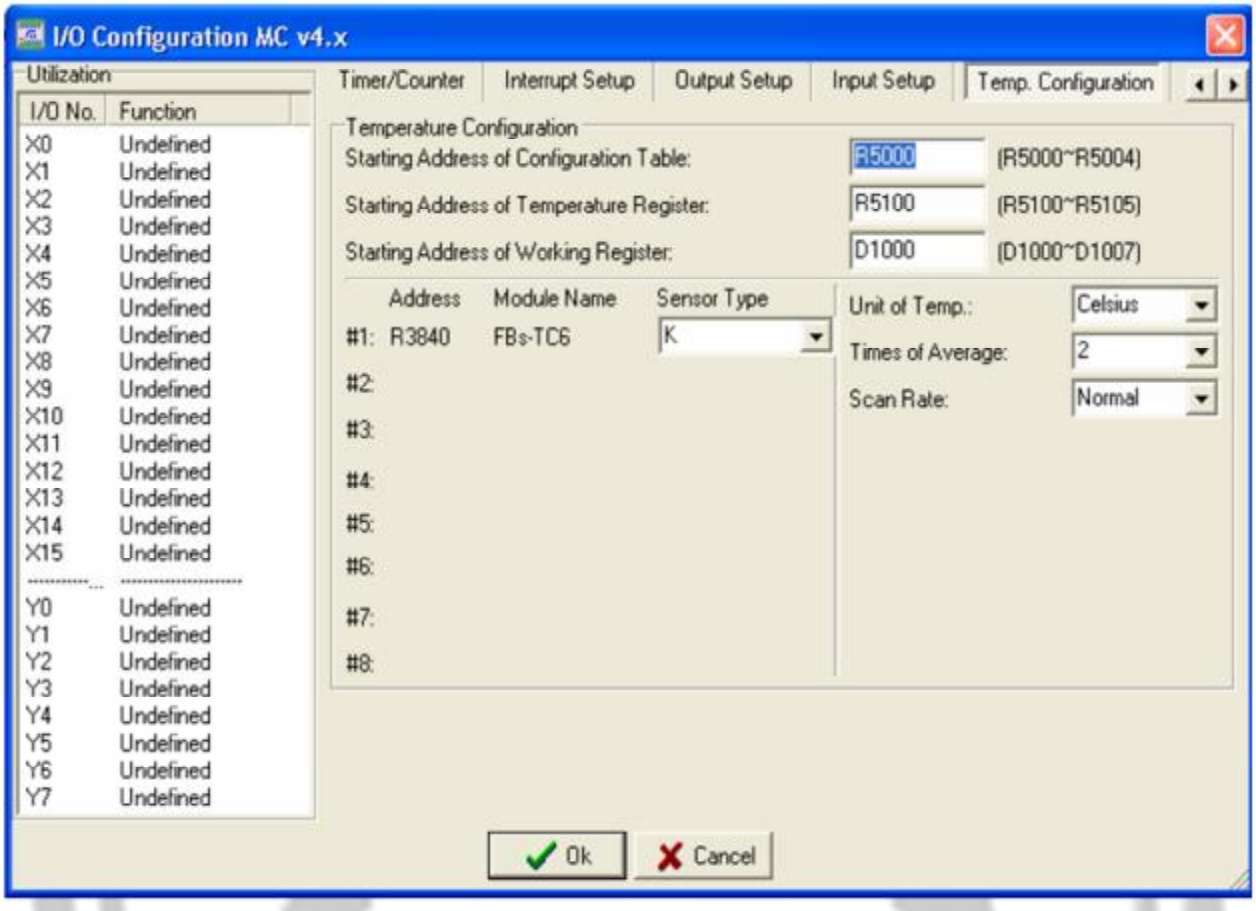
D، آدرس شروع محل ذخیره ی نتایج تبدیل را مشخص می کند.

بعد از اجرای تابع، هر یک کاراکتر (8-bits) از مبدا (S) به الگوی نمایشی 16 بیتی مربوطه تبدیل می شود.

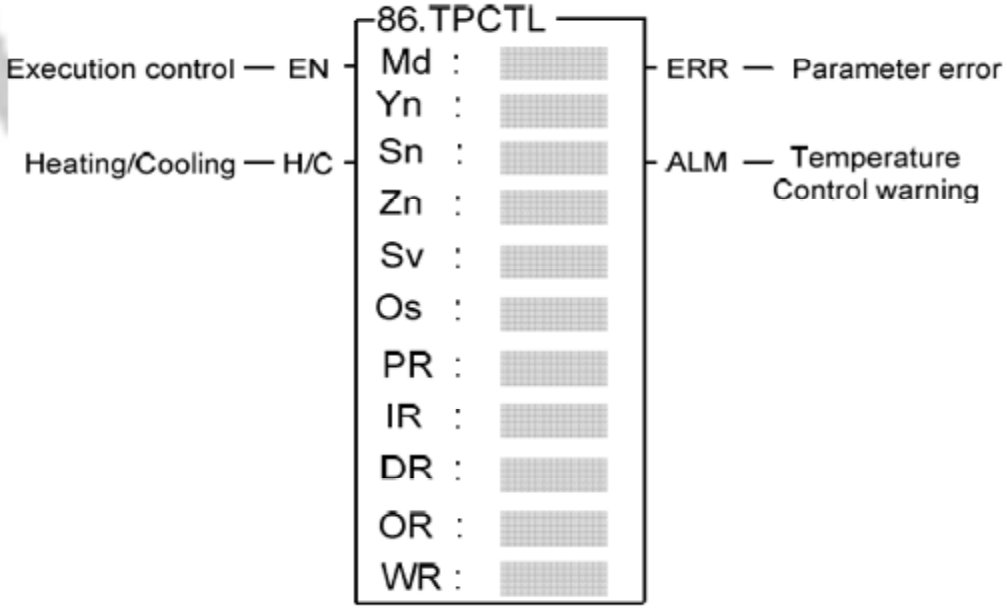
وقتی ورودی های "EN"=1 ،"OFF"=0 ، "ON"=0 و Md=0 باشد،این تابع تبدیل را اجرا می کند.وقتی "OFF"=1 و Md=0، تمام بیت های مربوط به الگوی نمایشی 16-seg صفر خواهند شد.این بدین معنیست که اگر 16-seg در این حالت به PLC متصل شود ،تمام LED های آن خاموش خواهد بود. وقتی "OFF"=1 و Md=1 ، تمام بیت های مربوط به نمایش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

Function 86.TCPTL TEMPERETURE PID

برای استفاده از تابع کنترل دما لازم است که ابتدا یک ماژول دما به PLC وصل شود و ترموکوپل مربوطه نیز به این ماژول متصل گردد و همچنین در جدول I/O Configuration ، مقادیری برای آیتم های موجود در پنجره Temperature Configurationتعیین گردد (که همانطور که در شکل زیر می بینیم با R5000 و R5100 و D1000 مقداردهی شده اند) و در این صورت مقادیر دمای جاری که توسط ترموکوپل خوانده می شود توسط R5100 قابل دستیابی خواهد بود .



Ladder symbol



- اگر H/C صفر باشد , کنترل مانند دستگاه کولر عمل می کند , یعنی اگر مقدار Setpoint کمتر از مقدار دمای کنونی باشد , خروجی فعال می گردد .
- اگر H/C یک باشد , کنترل مانند دستگاه هیتر عمل می کند , یعنی اگر مقدار Setpoint بیشتر از مقدار دمای کنونی باشد , خروجی فعال می گردد .

Md : انتخاب متد PID که شامل دو بخش است و توسط دو عدد صفر و یک مقداردهی می شوند.این دو عدد عبارتند از:

مد صفر : حداقل Overshot , فقط P , I می باشد و D ندارد

مد یک : حلقه PID عمومی

که برای استفاده از حلقه PID لازم است که متد یک انتخاب گردد ،

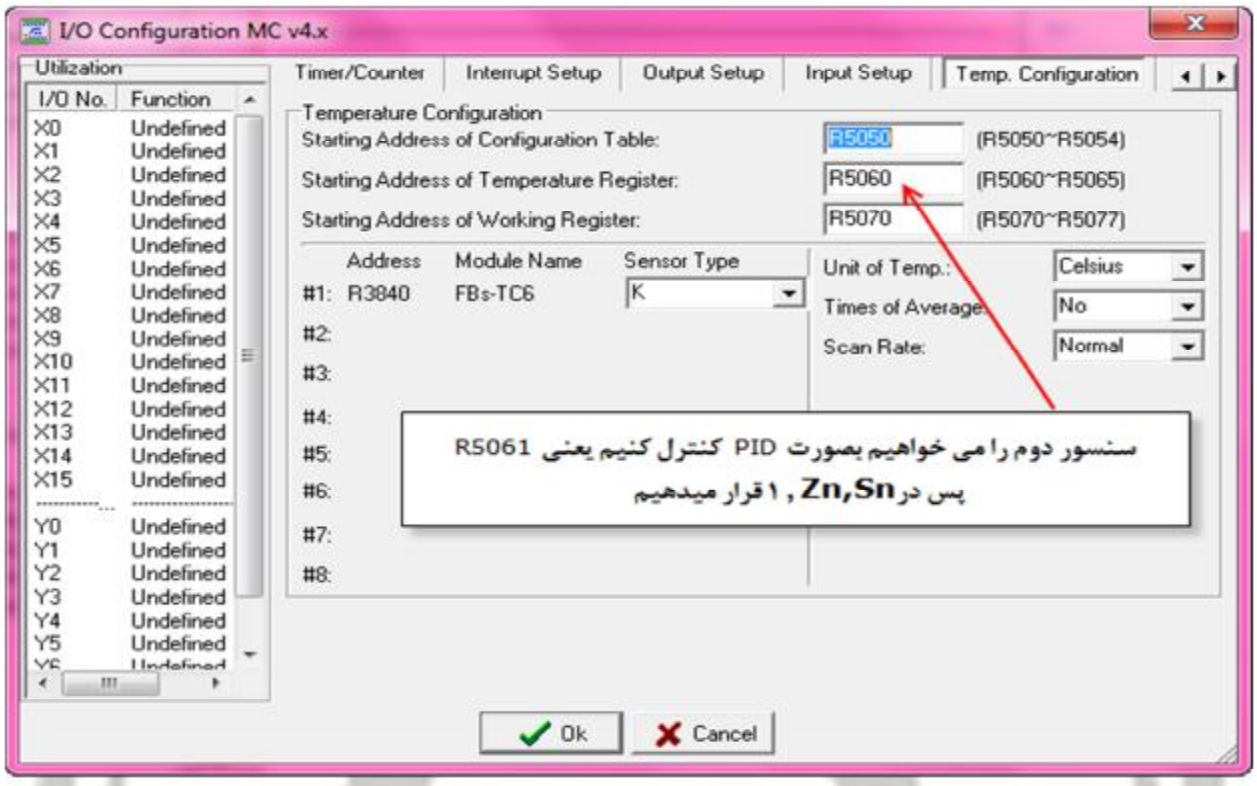
به ازای سنسورهای بعدی, ریجیسترهای بعد از ریجیسترهای نوشته شده در پارامترهای زیر مورد استفاده قرار می گیرند (به تعداد Zn)

با نوشتن یک تابع PID می توان 32 حلقه کنترل PID ایجاد کرد .

Yn : این پارامتر خروجی PID که می تواند فن یا هیتر یا المنت باشد را مشخص می کند .

Sn : شماره شروع سنسور دما که در I/O Configuration تنظیم می شود .

Zn : تعداد سنسورهایی که بعد از Sn می خواهیم با PID کنترل شوند (سنسورهایی که می خواهیم با PID کنترل کنیم باید پشت هم باشند) .



Sv : مقدار Setpoint دمای کنترل.

Os : میزان انحراف مجاز .

به عنوان مثال اگر Setpoint=2000 یعنی دما برروی 200 درجه سانتیگراد تنظیم شده باشد

و OS=50 باشد، میزان انحراف مجاز 50 برابر واحد دما خواهد بود (هر واحد 0.1 درجه را مشخص می کند)

1950 < 2000 < 2050

(195 C) < (200 C) < (205 C)

PR : در این قسمت رجیستر مربوط به مقدار گین برای حلقه PID مشخص می گردد (معمول : 100)

IR : در این قسمت رجیستر مربوط به مقدار ضریب انتگرال برای حلقه PID . مشخص می گردد (معمول : 12)

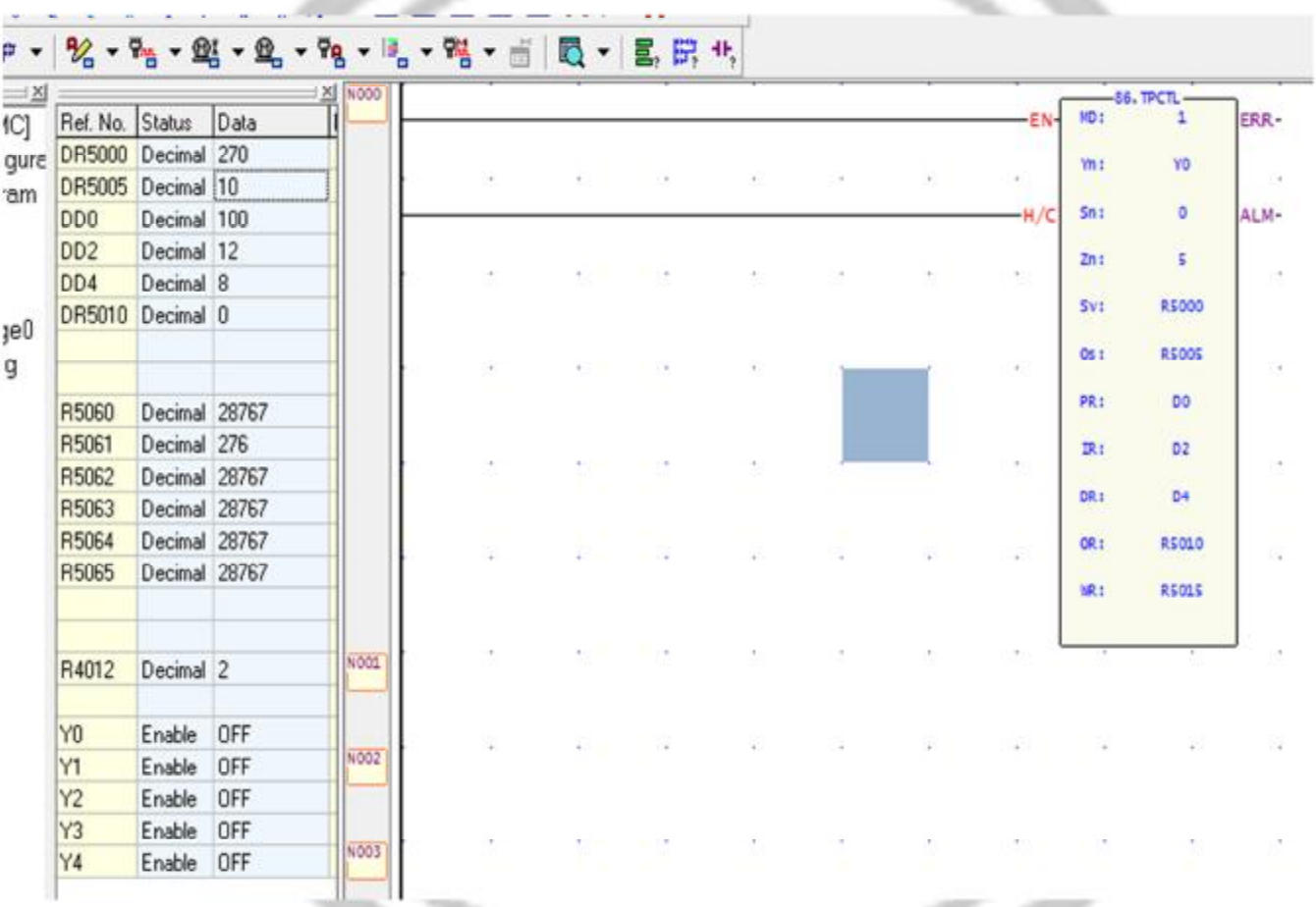
DR : در این قسمت رجیستر مربوط به مقدار ضریب مشتق برای حلقه PID . مشخص می گردد(معمول : 8)

OR : خروجی آنالوگ محاسبات PID . که مقدار آن از صفر تا 16383 تغییر می کند

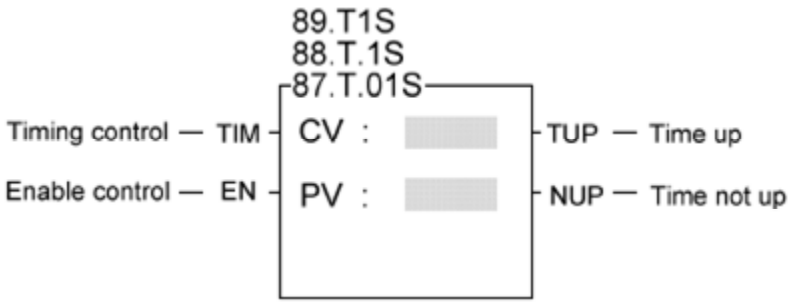
WR : Working Register ها را برای حلقه PID مشخص می کند که 9WORD می باشد.

ریجیستر 32 بیتی R4012 برای فعال کردن سنسور در کنترل PID می باشد به این معنی که به ازای هر بیت یکی از سنسورها فعال می گردد .

بیت صفر سنسور شماره صفر در I/O Configuration و بیت یک برای سنسور شماره 1 و... . چنانچه هر کدام از بیتهای مربوط به ریجیستر 1 شوند, سنسور مربوط به آن در کنترل PID قرار خواهد گرفت .



Function 87.88.89.CUMULATIVE TIMER



این تابع مانند تایمر ساده است با این تفاوت که این تایمر قابلیت نگه داشتن زمان را دارد.

در این تابع وقتی 1-"TIM" ،مانند تایمر ساده عمل می کند،اما اگر 0-"TIM" باشد،مقدار ذخیره شده حفظ می شود و پاک نمی شود.وقتی "TIM" مجدداً 1 شود،محاسبه ی زمان از ادامه ی آخرین باری که نگه داشته شده است ،ادامه پیدا می کند.

اگر تایمر احتیاج به ری ست شدن داشت ،"EN" را 0 کنید.

این تابع دو خروجی دارد :

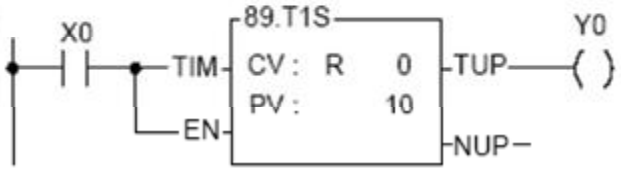
"TUP" که بعد از اتمام محاسبه ی زمان،1 می شود و

"NUP" که معمولاً وقتی "TUP" صفر است،1 می شود.

از ترکیب ورودی ها و خروجی ها برای ایجاد تایمرها با کارایی های مختلف می توانید استفاده کنید.

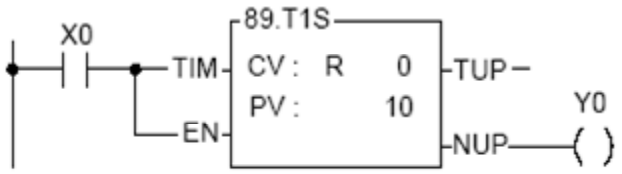
به عنوان مثال:

- وقتی X0 ، ON شود،بعد از 10 ثانیه ، ON،Y0 می شود.

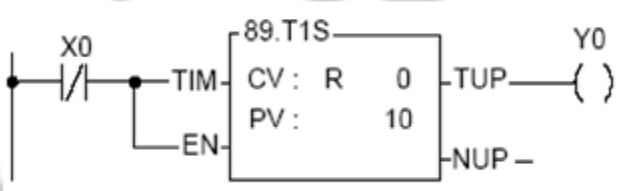


- Y0 به طور عادی ON است،وقتی ON،X0 شود،بعد از 10 ثانیه Y0 ، OFF

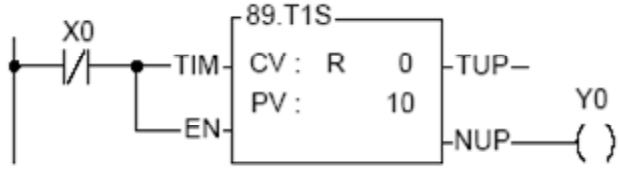
می شود.



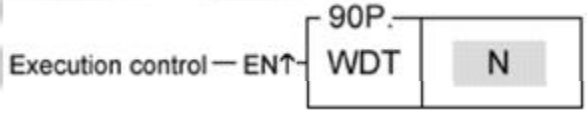
- YO به طور عادی OFF است،وقتی XO، OFF شود،بعد از 10 ثانیه، YO، ON می شود.



- YO به طور عادی ON است،وقتی XO،OFF شود،بعد از 10 ثانیه YO ، OFF می شود.



Function 90.WATCH DOG TIMER

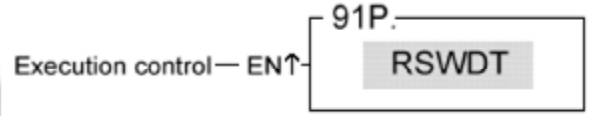


هدف اصلی از طراحی تایمر WDT ، محافظت ویژه ایست که از طرف سیستم اعمال می شود.برای مثال،اگر واحد پردازشگر PLC (CPU) ناگهان آسیب دیده و راهی برای اجرای برنامه یااعمال تغییرات به ورودی ، خروجی ها (I/O refresh)نباشد، بعد از سپری شدن زمان تعیین شده ، WDT به صورت خودکار تمام ورودی ، خروجی ها را خاموش

می کند. در کاربری های خاص ،اگر scan time خیلی طولانی باشد ، ممکن است باعث بروز ناامنی در کارکرد برنامه یا خارج شدن برنامه از کنترل شود که این تابع می تواند به روی scan time ، اعمال محدودیت کند . برای اطمینان از عملکرد صحیح آن ، گزینه Pulse در این تابع باید فعال شود . هرگاه "EN" از 0 به 1 تغییر کند:

WDT شمارش زمان N×10 ms را کرده و اگر در این مدت،ورودی "EN" مجددا اسکن نشود ، پس از زمان تعیین شده ، PLC از حالت Run خارج شده و خروجی ها خاموش می شوند . یک بار که WDT، تنظیم شد ، مقدار آن باقی مانده و نیازی به تنظیم مجدد در هر اسکن نیست .

Function 91.RESET WDT



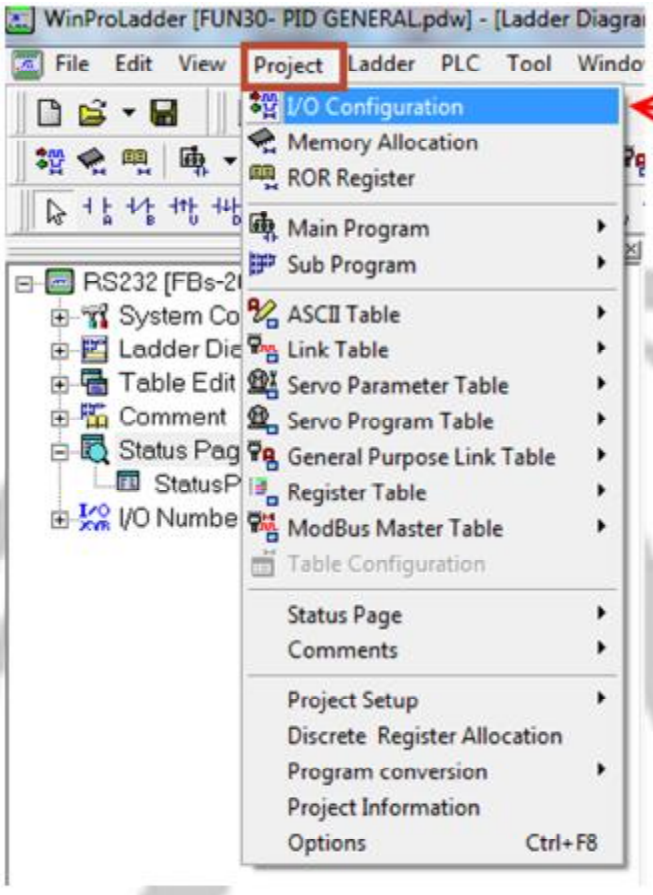
هرگاه "EN"، 1 بشود؛ تایمر WDT ری ست شده و محاسبه زمان را از 0 شروع می کند.

Function 92 & 93 HIGH SPEED COUNTER

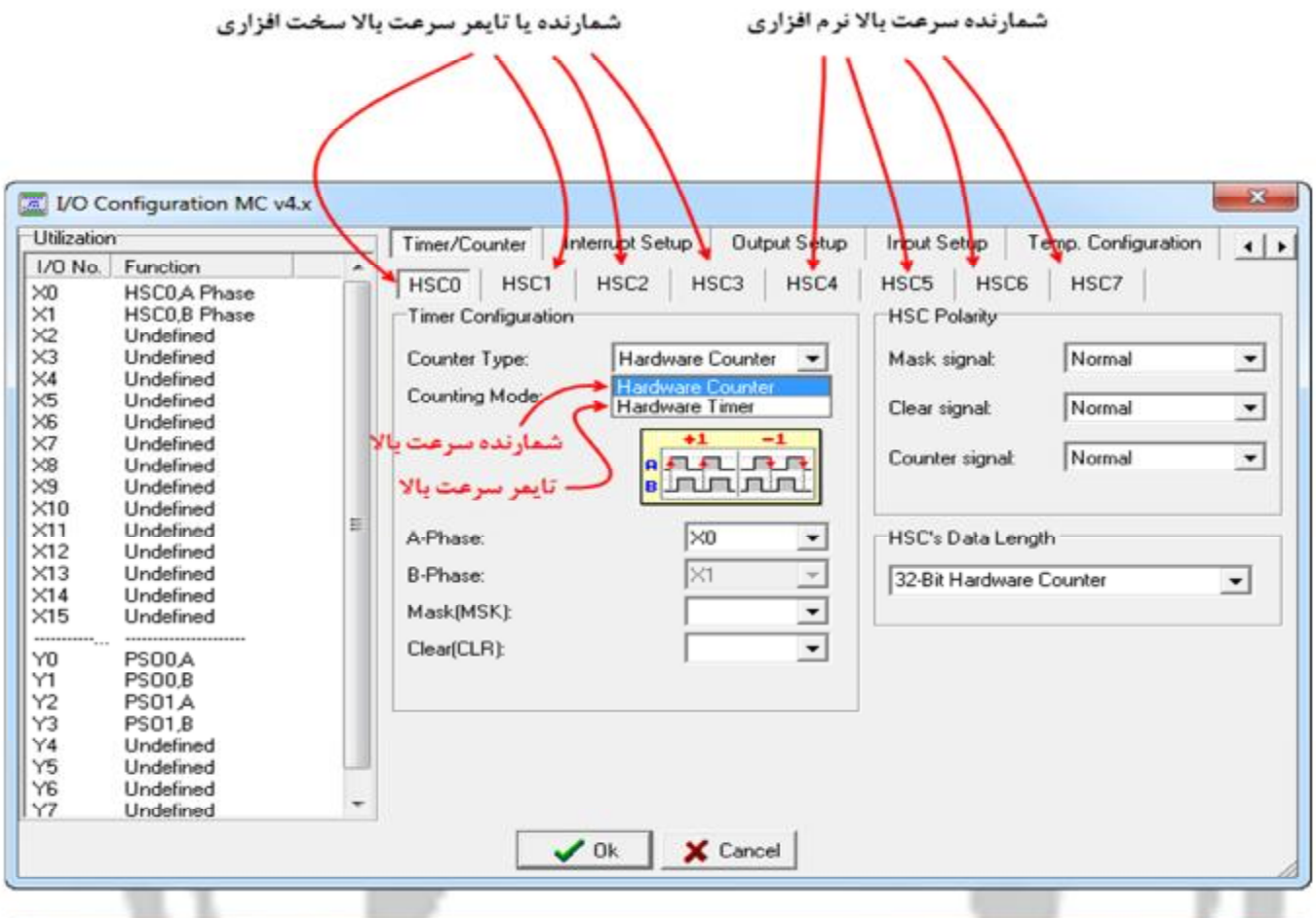
دو نوع شمارنده داریم : سرعت بالا و سرعت پایین (منظور از سرعت فرکانس ورودی می باشد) .

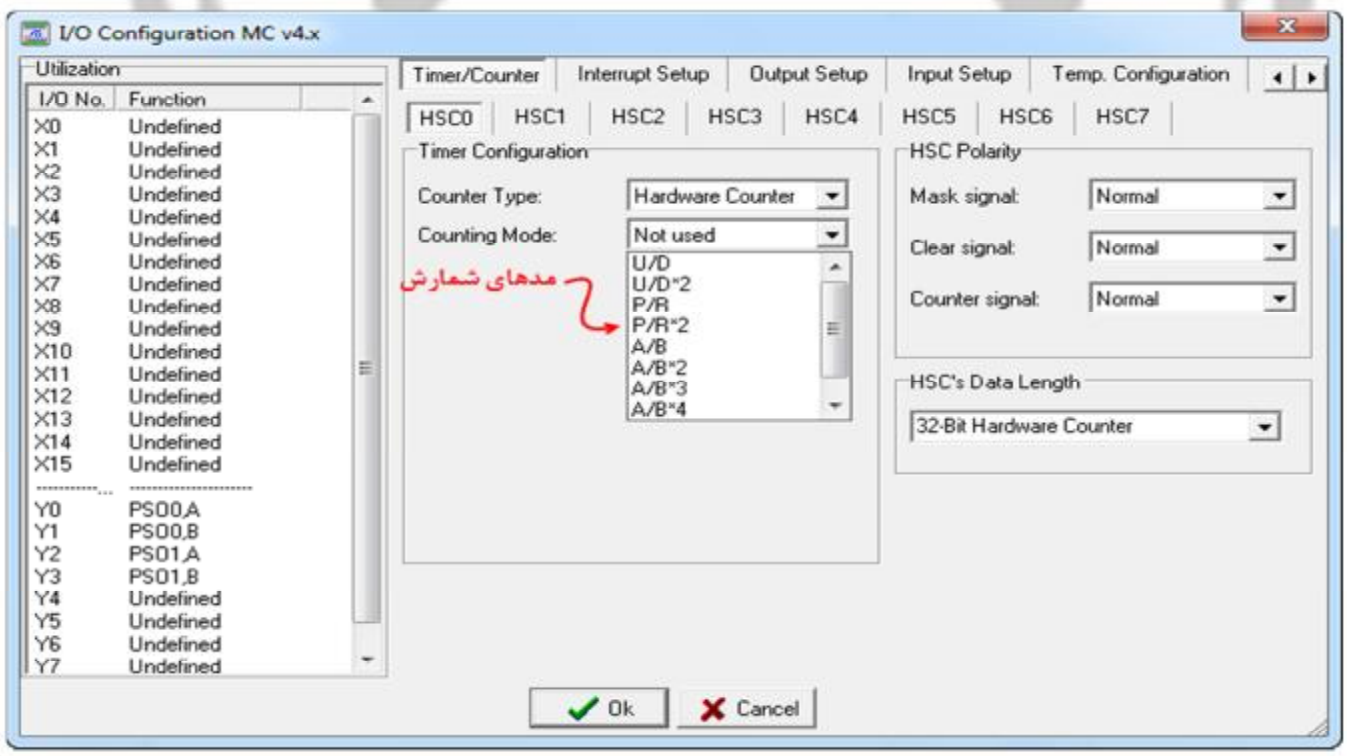
شمارنده های سرعت بالا (در مقیاس کیلوهرتز) نیاز به سخت افزار خاصی دارند تا بتوانند در این فرکانس پالسها را بشمارند ولی شمارنده های سرعت پایین (در مقیاس دهها هرتز) با ورودی های معمولی PLC نیز کار می کنند. به شمارنده های سرعت بالا شمارنده های سرعت بالای سخت افزاری و به شمارنده های سرعت پایین شمارنده های نرم افزاری می گویند.

تنظیم ورودی ها برای حالت شمارنده بودن :



برای تنظیم ورودی های
شمارنده سرعت بالا

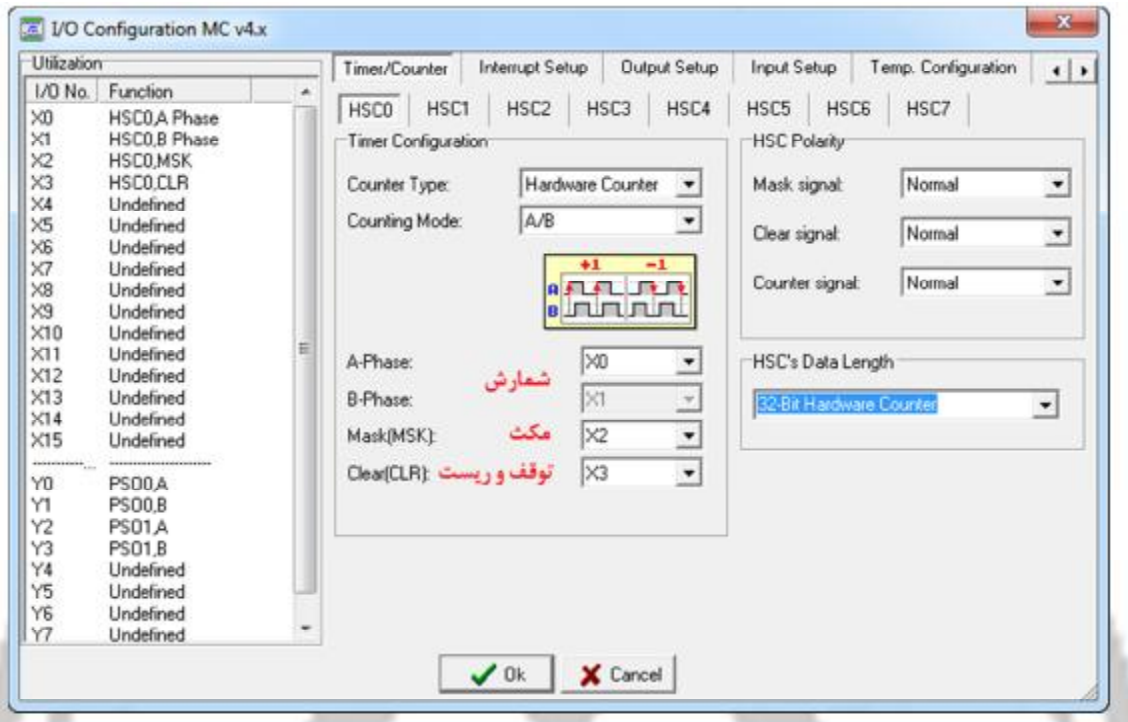

شمارنده یا تایمر سرعت بالا سخت افزاری
شمارنده سرعت بالا نرم افزاری
I/O Configuration MC v4.x
Utilization
I/O No. Function
X0 HSC0,A Phase
X1 HSC0,B Phase
X2 Undefined
X3 Undefined
X4 Undefined
X5 Undefined
X6 Undefined
X7 Undefined
X8 Undefined
X9 Undefined
X10 Undefined
X11 Undefined
X12 Undefined
X13 Undefined
X14 Undefined
X15 Undefined
Y0 PS00,A
Y1 PS00,B
Y2 PS01,A
Y3 PS01,B
Y4 Undefined
Y5 Undefined
Y6 Undefined
Y7 Undefined
Timer/Counter
Interrupt Setup
Output Setup
Input Setup
Temp. Configuration
HSC0
HSC1
HSC2
HSC3
HSC4
HSC5
HSC6
HSC7
Timer Configuration
Counter Type:
Hardware Counter
Counting Mode:
Hardware Counter
Hardware Timer
شمارنده سرعت بالا
تایمر سرعت بالا
+1
-1
A
B
A-Phase:
X0
B-Phase:
X1
Mask(MSK):
Clear(CLR):
HSC Polarity
Mask signal:
Normal
Clear signal:
Normal
Counter signal:
Normal
HSC's Data Length
32-Bit Hardware Counter
Ok
Cancel


I/O Configuration MC v4.x
Utilization
I/O No. Function
X0 Undefined
X1 Undefined
X2 Undefined
X3 Undefined
X4 Undefined
X5 Undefined
X6 Undefined
X7 Undefined
X8 Undefined
X9 Undefined
X10 Undefined
X11 Undefined
X12 Undefined
X13 Undefined
X14 Undefined
X15 Undefined
Y0 PS00,A
Y1 PS00,B
Y2 PS01,A
Y3 PS01,B
Y4 Undefined
Y5 Undefined
Y6 Undefined
Y7 Undefined
Timer/Counter
Interrupt Setup
Output Setup
Input Setup
Temp. Configuration
HSC0
HSC1
HSC2
HSC3
HSC4
HSC5
HSC6
HSC7
Timer Configuration
Counter Type:
Hardware Counter
Counting Mode:
Not used
مدهای شمارش
U/D
U/D*2
P/R
P/R*2
A/B
A/B*2
A/B*3
A/B*4
HSC Polarity
Mask signal:
Normal
Clear signal:
Normal
Counter signal:
Normal
HSC's Data Length
32-Bit Hardware Counter
Ok
Cancel

مدهای کاری هر شمارنده :

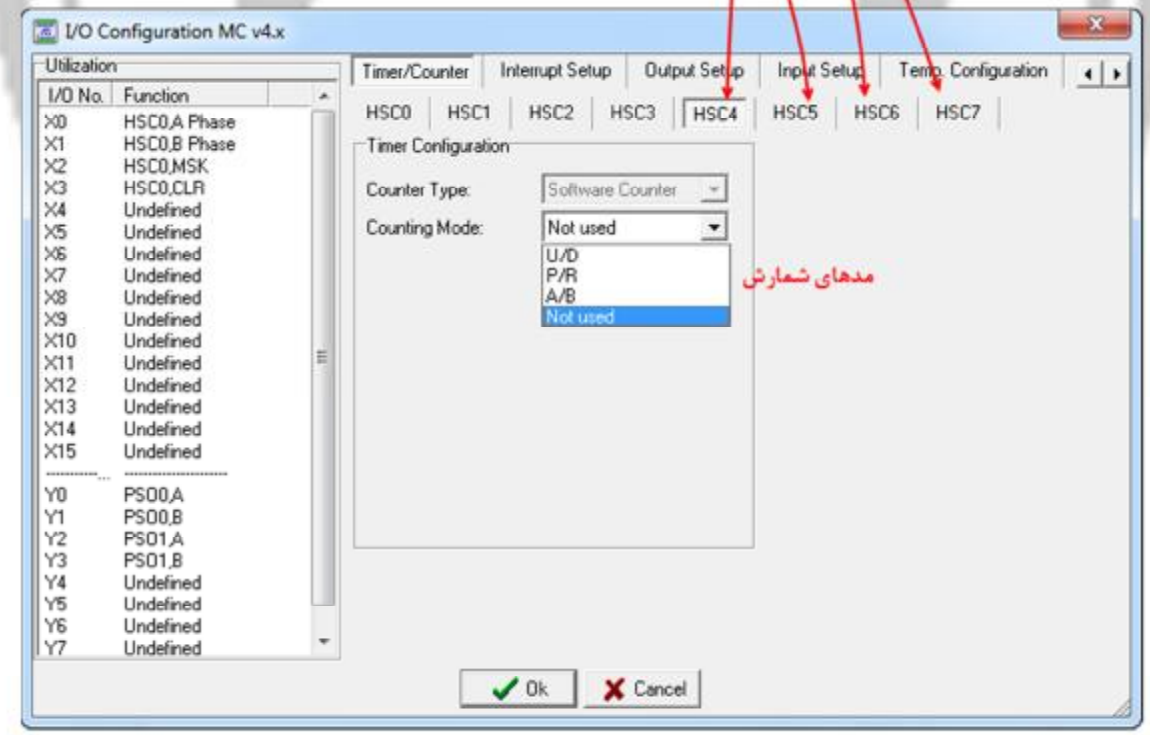
Counting Mode			HHSC (HSC0～HSC3)	SHSC (HSC4～HSC7)	Counting Waveform
					Up Counting（+1） Down Counting（−1）
Up-down pulse	MD 0	U/D	○	○	
	MD 1	U/D×2	○		
Pulse-direction	MD 2	K/R	○	○	
	MD 3	K/R×2	○		
AB phase	MD 4	A/B	○	○	
	MD 5	A/B×2	○		
	MD 6	A/B×3	○		
	MD 7	A/B×4	○		

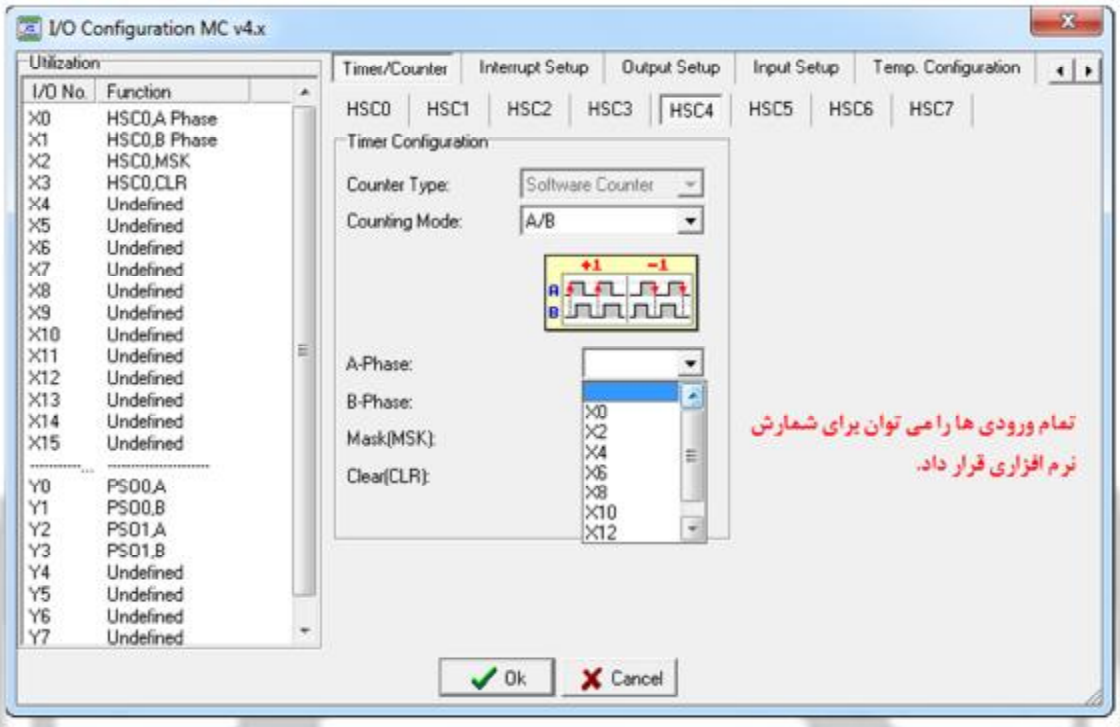
• The up/down arrow (↑ , ↓) on the positive/negative edge in the waveform represents where counting (+1 or −1) occurs.



شمارنده سرعت بالا نرم افزاری

شمارنده های نرم افزاری براساس زمان اسکن برنامه ورودی ها را می شمارد (تا دهها هرتز ، مطابق با زمان اسکن برنامه)





نحوه خواندن شمارنده های سرعت بالا :

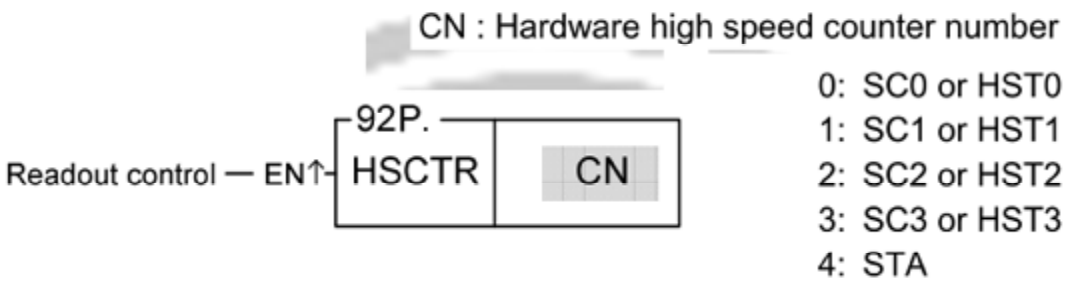
به ازای هر کانال ورودی شمارشگر 2 ریجیستر 32 بیتی اختصاص داده شده است :

CV : Current Value مقدار ی که CPU هم اکنون در حال شمارش است ، دو استفاده از آن می توان داشت :

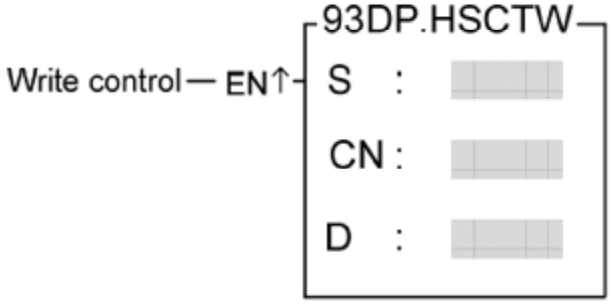
1- **قرار دادن مقدار صفر در ریجیستر شمارش کنونی (ریست کردن شمارشگر) ، توسط فانکشن "HSCTW" 93**
2- **خواندن مقدار کنونی شمارش شده ، توسط فانکشن 92 "HSCTR"**

PV : Preset Value ، وقتی شمارشگر به این مقدار رسید اینتراپت شمارشگر فعال می شود و دستورات اینتراپت برای یک بار اجرا می شود .

فانکشن 92 (HSCTR) : برای کپی کردن مقدار کنونی شمارنده به داخل ریجیستر از پیش تعریف شده برای هر شمارنده



فانکشن 93 (HSCTR) : برای کپی اعداد به حافظه های پیش فرض اختصاص داده شده مربوط به شمارنده ها در CPU می باشند.



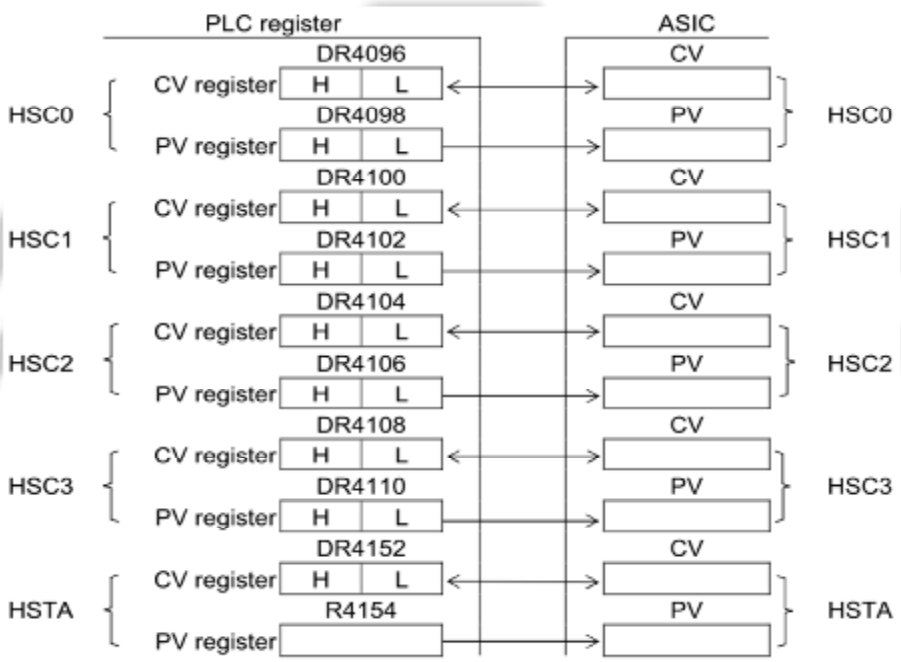
S : The source data for writing

CN : Hardware high speed counter to be written

0: HSC0 or HST1
1: HSC1 or HST2
2: HSC2 or HST3
3: HSC3 or HST4
4: HSTA

D : Write target (0 represents CV, 1 represents PV)

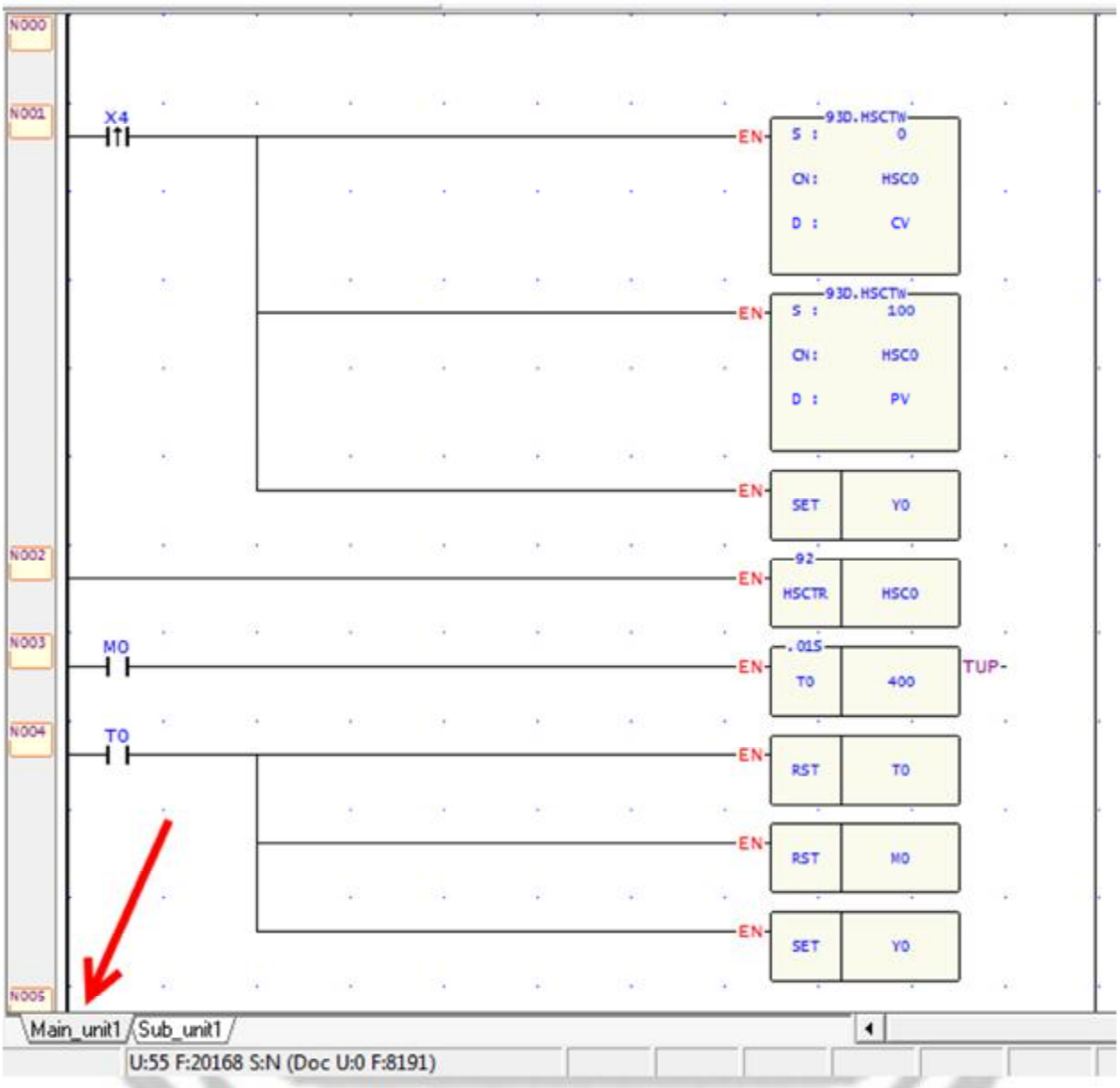
حافظه های پیش فرض اختصاص داده شده مربوط به شمارنده ها :

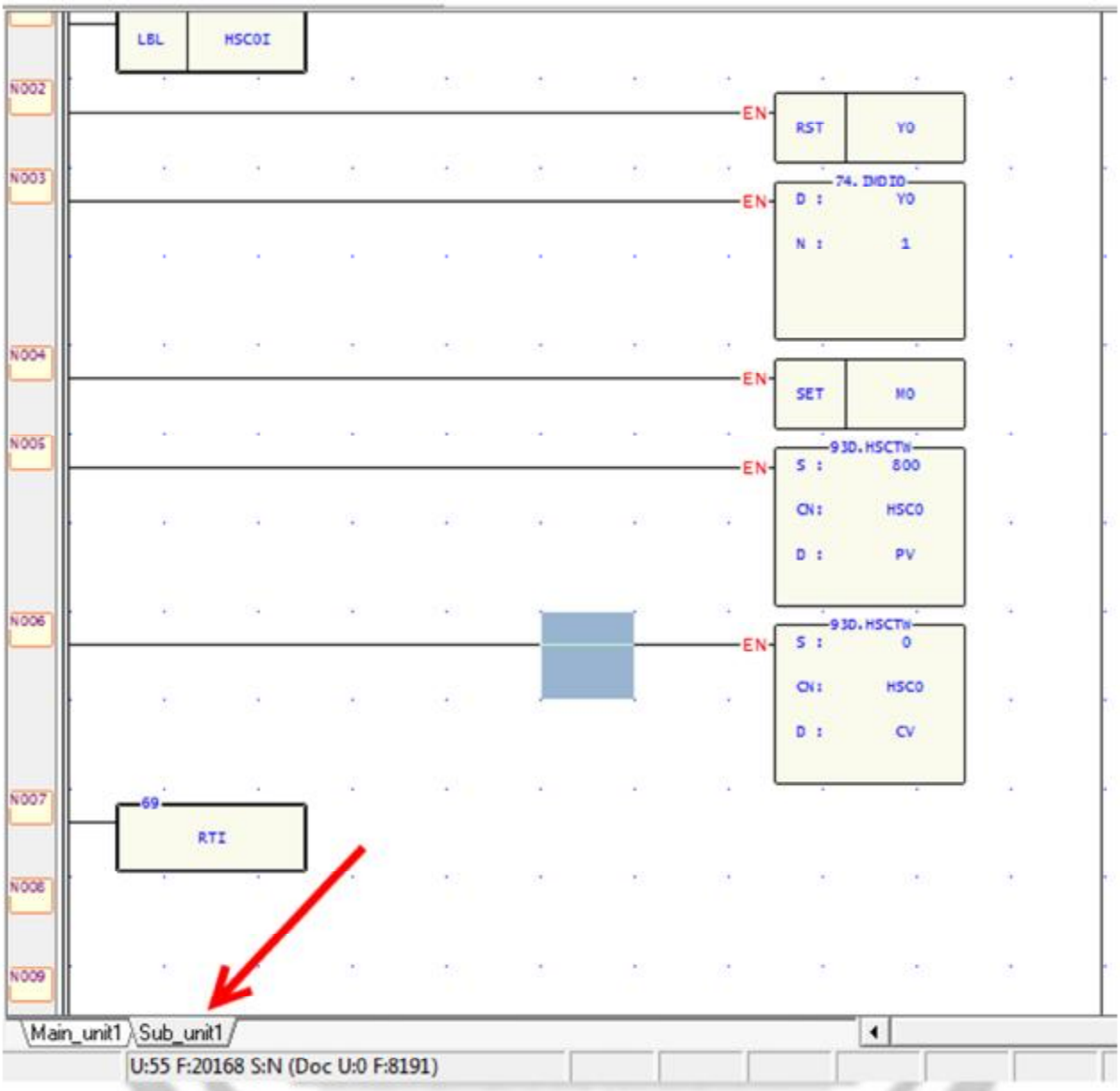


Type / Signal Allowed		MC/MN								MA
		HHSC				SHSC				SHSC
		HSC0	HSC1	HSC2	HSC3	HSC4	HSC5	HSC6	HSC7	HSC4 ~ HSC7
CV Register		DR4096	DR4100	DR4104	DR4108	DR4112	DR4116	DR4120	DR4124	The same as SHSC of MC/MN
PV Register		DR4098	DR4102	DR4106	DR4110	DR4114	DR4118	DR4122	DR4126	The same as SHSC of MC/MN
Counting Input	U,K or A	X0	X4	X8	X12	X0~X15	X0~X15	X0~X15	X0~X15	The same as SHSC of MC/MN
	D,R or B	X1	X5	X9	X13	X0~X15*	X0~X15*	X0~X15*	X0~X15*	The same as SHSC of MC/MN
Control Input	Mask	X2	X6	X10	X14	X0~X15	X0~X15	X0~X15	X0~X15	The same as SHSC of MC/MN
	Clear	X3	X7	X11	X15	X0~X15	X0~X15	X0~X15	X0~X15	The same as SHSC of MC/MN
Software MASK Relay		M1940	M1946	M1976	M1979	M1982	M1984	M1986	M1988	The same as SHSC of MC/MN
Software CLEAR Relay		M1941	M1947	M1977	M1980	Clear the Current Value Register directly				
Software Direction Selection(MD2,3 Only)		M1942	M1948	M1978	M1981	M1983	M1985	M1987	M1989	The same as SHSC of MC/MN
Interrupt Subroutine Label		HSC0I	HSC1I	HSC2I	HSC3I	HSC4I	HSC5I	HSC6I	HSC7I	The same as SHSC of MC/MN

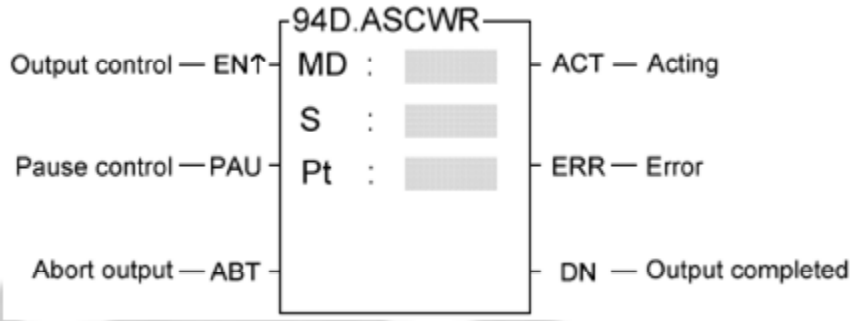
وقفه های مربوط به شمارنده های سرعت بالا : وقتی **CV=PV** شود سیکل برنامه وارد اینترایت مورد نظر می شود

مثال : برنامه ایی که با تحریک ورودی X4 خروجی Y0 فعال شده و موتور متصل به خروجی Y0 برای اولین بار به اندازه 100 پالس می چرخد و سپس 4 ثانیه متوقف شده و سپس در مراحل بعد به اندازه 800 پالس بچرخد و 4 ثانیه متوقف شود.


N000
N001
X4
EN
93D.HSCTW
S :
0
CN:
HSC0
D :
CV
EN
93D.HSCTW
S :
100
CN:
HSC0
D :
PV
EN
SET
Y0
N002
EN
92
HSCTR
HSC0
N003
M0
EN
.01S
T0
400
TUP-
N004
T0
EN
RST
T0
EN
RST
M0
EN
SET
Y0
N005
Main_unit1
Sub_unit1
U:55 F:20168 S:N (Doc U:0 F:8191)



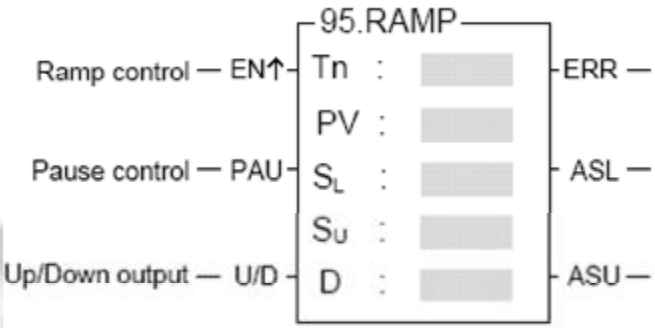
Function 94.ASCII WRITE



از این تابع برای فرستادن اطلاعاتی که به صورت اسکی (ASCII) کد شده اند ، به پورت 1 ، استفاده شده و کاربرد آن در ارتباط با دستگاه هایی است که تنها راه ارتباط با آنها پروتکل اسکی می باشد.

Function 95.RAMP

تابع شیب برای خروجی دیجیتال به انالوگ

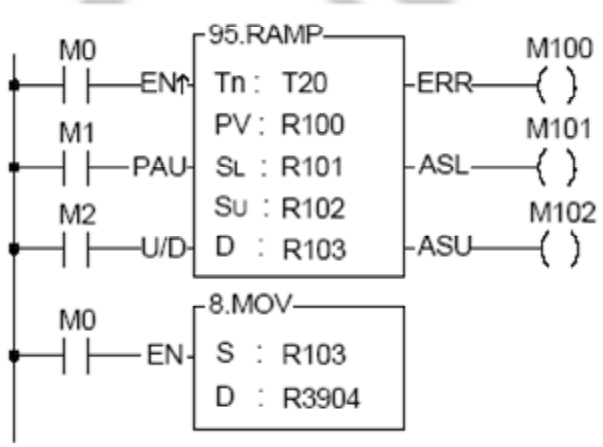


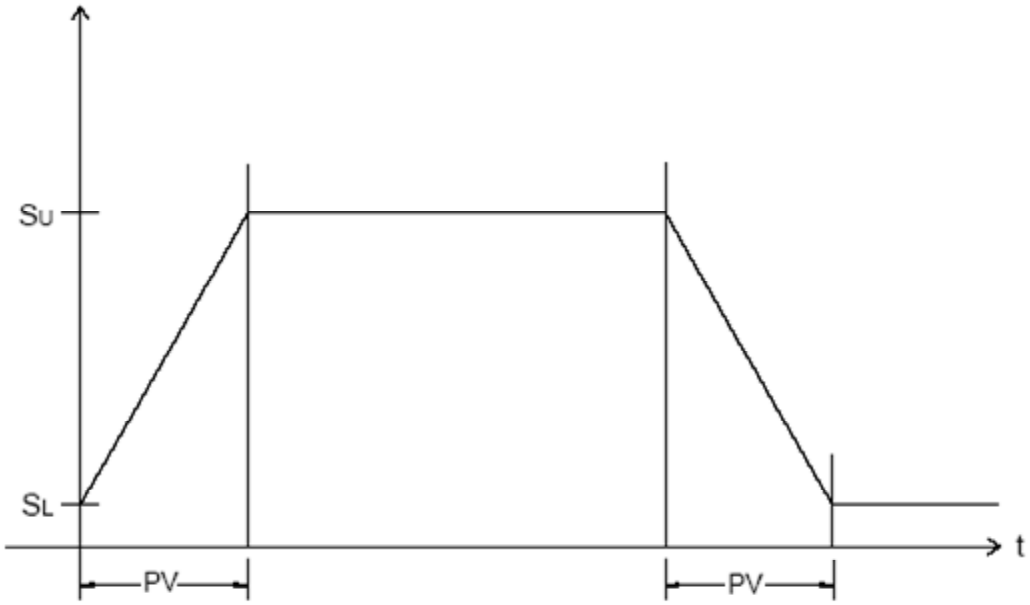
کاربرد این تابع در مواقعی است که نیاز به افزایش یا کاهش مقدار یک رجیستر به ویژه خروجی آنالوگ است که این افزایش یا کاهش با شیبی با ابتدا و انتهای مشخص و در زمان معین صورت می پذیرد . درTn ، تایمری با پایه زمانی 0.01 ثانیه قرار می گیرد که نباید در جای دیگری استفاده شود. PV مقدار پیش تنظیم تایمر Tn است . S_L حد پایین شیب و Su حد بالای شیب را مشخص می کند. هرگاه "EN" از 0 به 1 تغییر کند:

تایمر Tn ،ری ست شده ، سپس اگر ورودی 1="U/D" باشد ،تابع به صورت افزایشی عمل کرده و مقدار S_L در رجیستر D ذخیره شده و هر 0.01s مقدار D به اندازه S_L/ PV – Su افزایش می یابد. وقتی مقدار D به بالاترین حد (Su) رسید، خروجی 1="ASU" می شود. اگر 0="U/D" باشد، مقدار Su در رجیستر D ذخیره شده و هر 0.01s ، مقدار D به اندازه ی S_L/ PV Su کاهش می یابد. وقتی مقدار D به S_L رسید، خروجی 1 "ASL" می شود. پس از

ست شدن ورودی "EN" ، تغییر "U/D" اعمال نخواهد شد . اگر نیاز به توقف عمل شیب دادن بود ،باید ورودی "PAU"=1 بشود.وقتی "PAU"=0 شود ، عمل شیب دادن تا هنگامی که پایان یابد ادامه خواهد یافت. مقدار Su باید بزرگتر از S_L داده شود،در غیر این صورت تابع اجرا نشده و خروجی "ERR"،1 خواهد شد.

مثال:

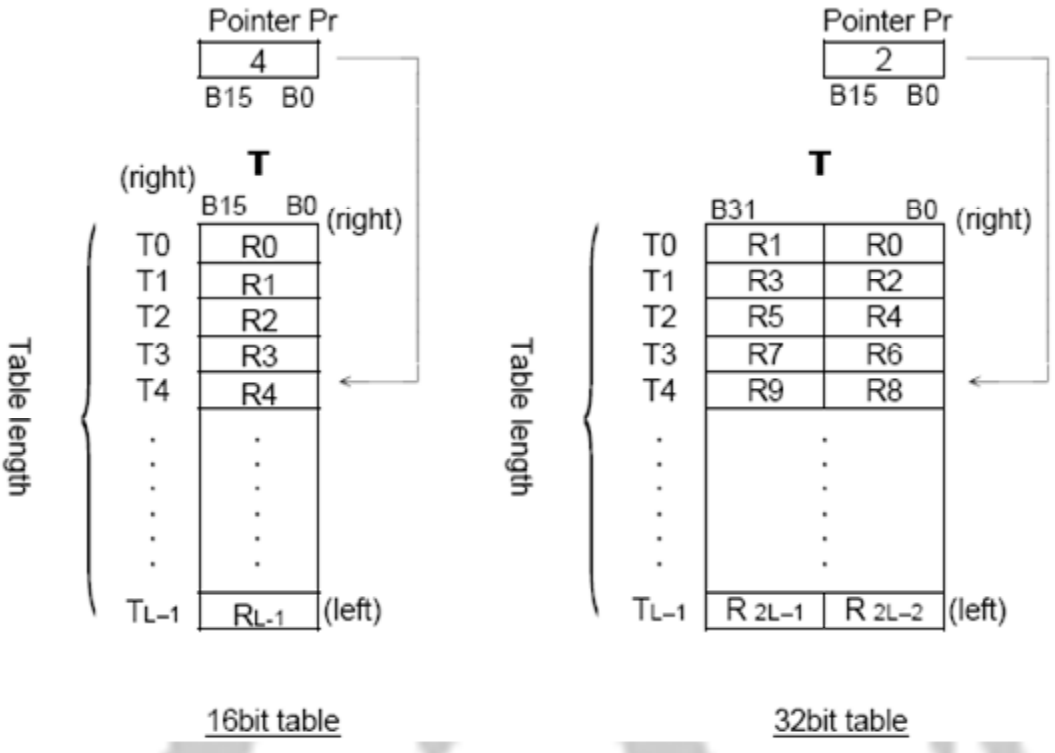




توابع جدولی

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

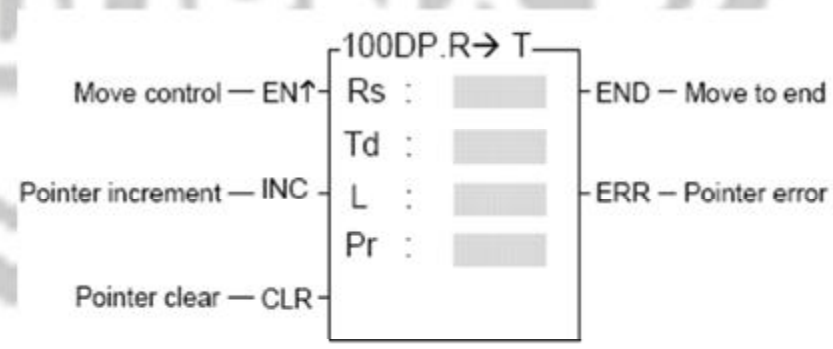
یک جدول شامل دو یا چند رجیستر متوالی است (16 یا 32 بیتی) تعداد رجیستر هایی که تشکیل یک جدول می دهند را طول جدول می نامند.(L) اجرای توابع جدولی بیشتر برای پردازش داده ها استفاده می شود،مانند انتقال ، کپی ، مقایسه ، جستجو و ...بین جدول ها و رجیستر ها یا بین جدول ها. در میان توابع جدولی ،بیشتر توابع از یک اشاره گر (Pointer) استفاده می کنند تا مشخص کنند کدام رجیستر جدول ،هدف اجرا باشد.اشاره گر برای تمام جدول ها، یک رجیستر 16 بیتی بوده و محدوده اشاره آن ،L-1~0 است. هنگام اجرای عملیاتی مانند شیفت به راست یا چپ و چرخش به راست یا چپ ، جهت شیفت به این ترتیب مشخص می شود که اگر جهت به سمت رجیستر بالاتر باشد،چپگرد گفته شده و اگر جهت به سمت رجیستر پایین تر باشد راستگرد گفته می شود.



16bit table

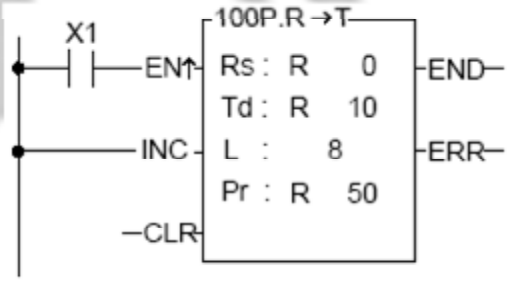
32bit table

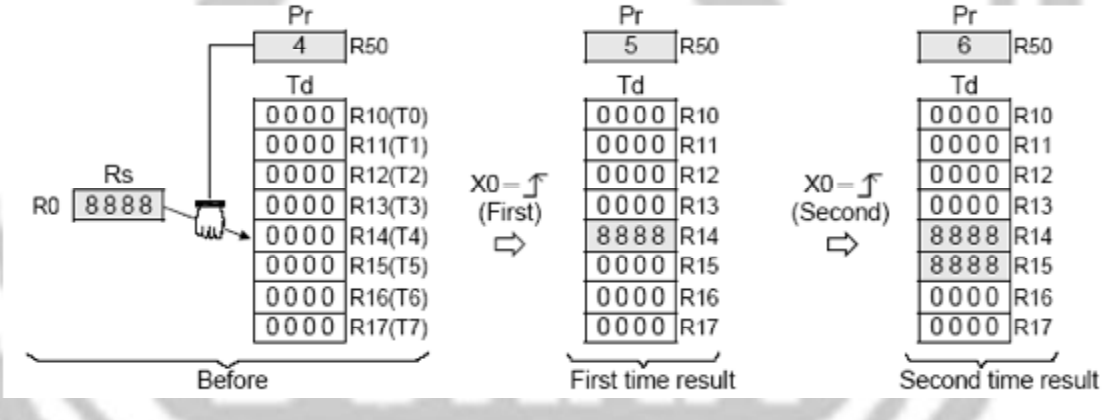
Function 100.REGISTER TO TABLE MOVE



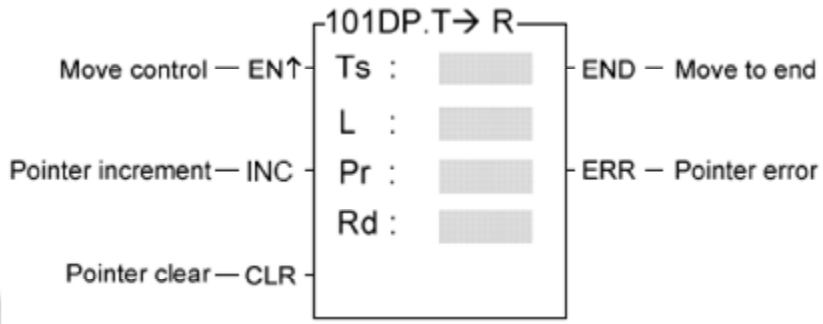
هرگاه "EN" از 0 به 1 تغییر کند: محتویات رجیستر مبدا (Rs) به داخل رجیستری از جدول ریخته می شود که Pointer به آن اشاره می کند. در Td رجیستر شروع جدول قرار می گیرد به طول (L). هرگاه سیگنال ورودی "CLR"-1 بشود ، مقدار اشاره گر(Pr) را ریست می کند. اگر مقدار Pr به L-1 برسد(یعنی به آخرین رجیستر جدول اشاره کند) خروجی "END" ، 1 شده و اجرای این تابع به پایان می رسد. اگر مقدار Pr کمتر از L-1 باشد،مرتب "INC" را چک کرده و اگر"INC" ،1 باشد، مقدار Pr در هر اسکن افزایش می یابد. محدوده موثر Pointer ، 0~L-1 بوده و فراتر از این محدوده ، خروجی "ERR" ،1 شده و این تابع اجرا نمی شود.

مثال:



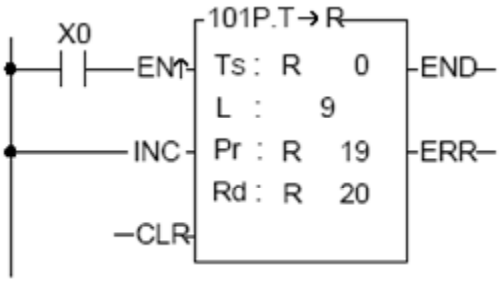


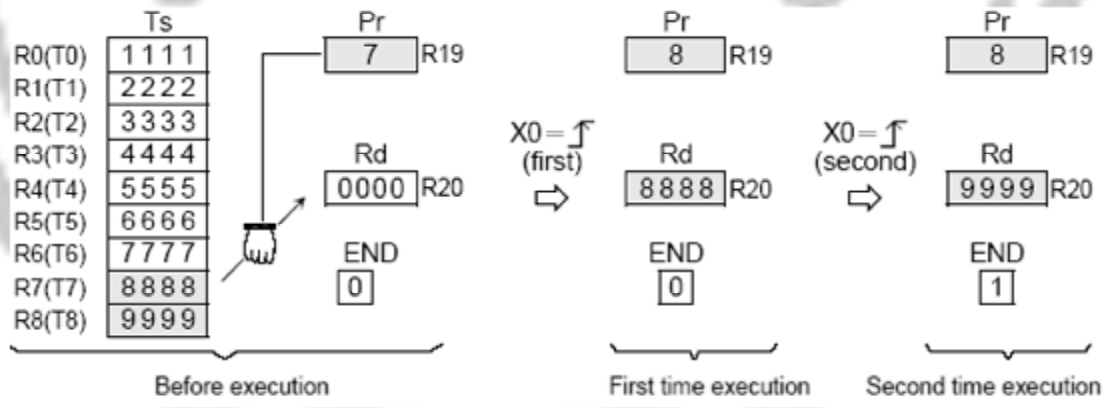
Function 101.TABLE TO REGISTER MOVE



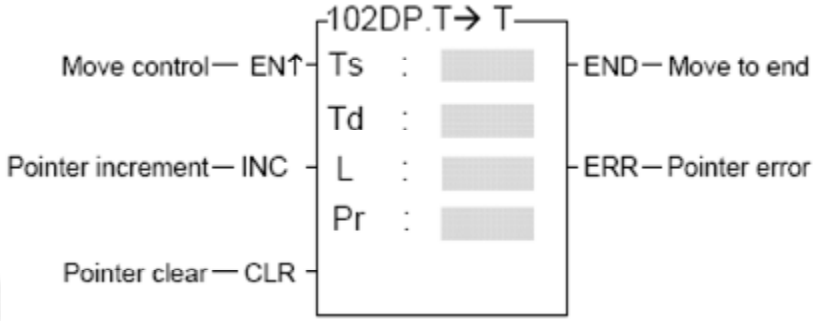
این تابع بر عکس تابع قبل عمل کرده و محتویات یک رجیستر از جدول مورد نظر را به رجیستر مقصد منتقل می کند.

مثال:



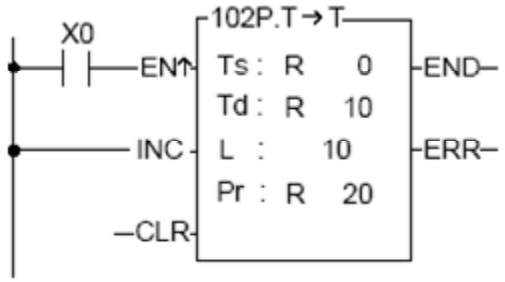


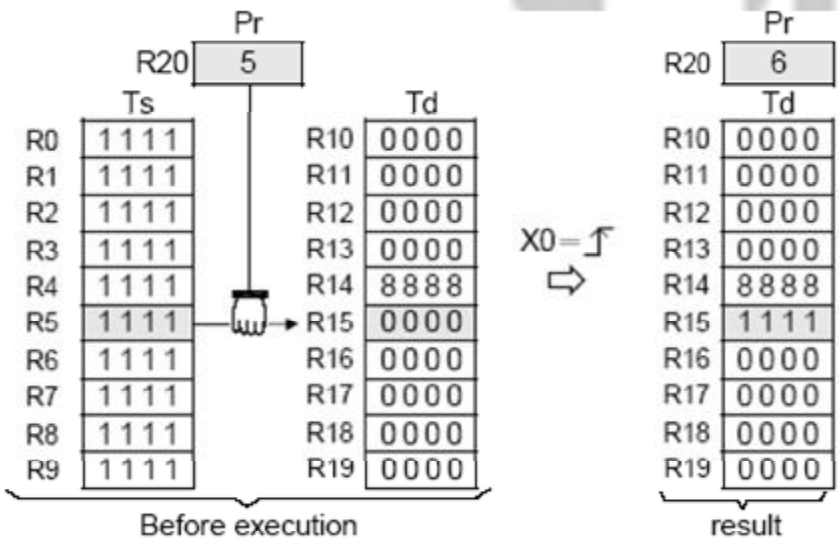
Function 102.TABLE TO TABLE MOVE



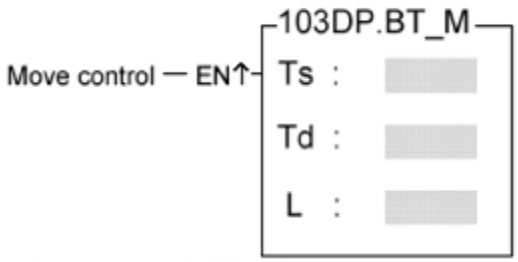
در Ts، رجیستر شروع جدول مبدا قرار می گیرد و در Td، ، رجیستر شروع جدول مقصد قرار می گیرد.

L، طول جدول مقصد و مبدا را مشخص می کند. هرگاه "EN" از 0 به 1 تغییر کند: محتویات آن رجیستری از جدول مبدا که Pr به آن اشاره می کند،به رجیستر معادلش در جدول مقصد منتقل می شود.



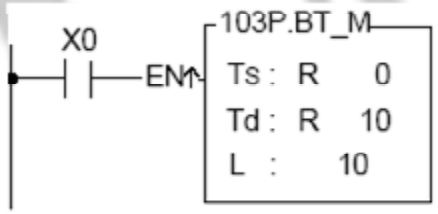


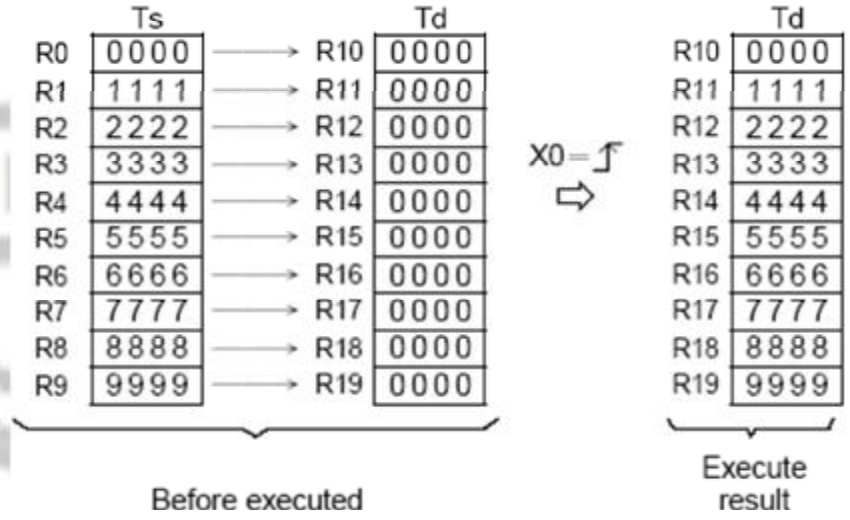
Function 103.BLOCK TABLE MOVE



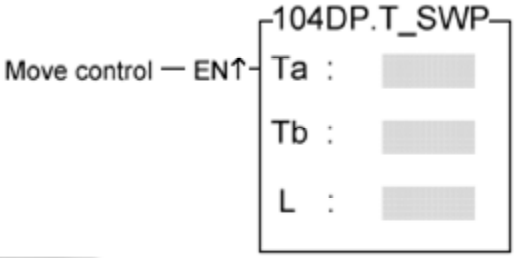
هرگاه "EN" از 0 به 1 تغییر کند: محتویات رجیسترهای جدول مبدا به داخل رجیسترهای معادلشان در جدول مقصد،کپی می شوند. L معرف طول هردو جدول است .

مثال:



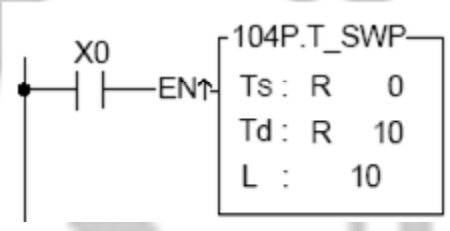


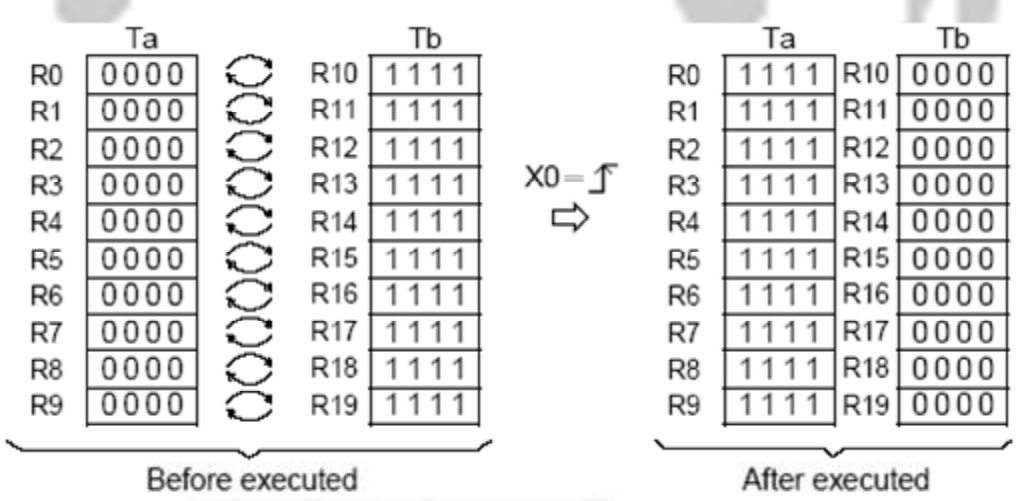
Function 104.BLOCK TABLE SWAP



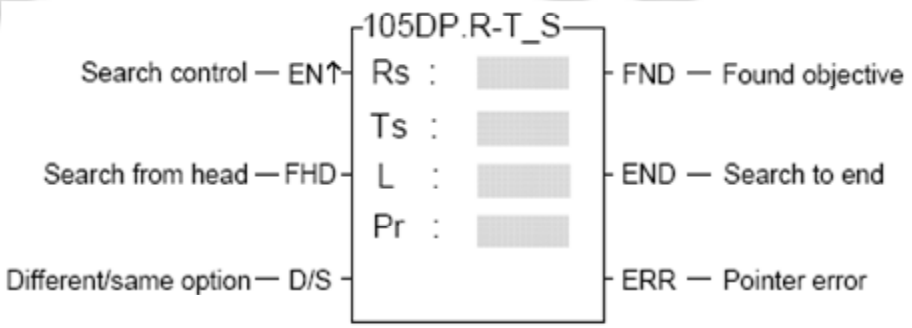
هرگاه "EN" از 0 به 1 تغییر کند: محتویات رجیسترهای جدول a با محتویات رجیسترهای معادلشان در جدول b ،جا به جا می شوند. L معرف طول هردو جدول است .

مثال:



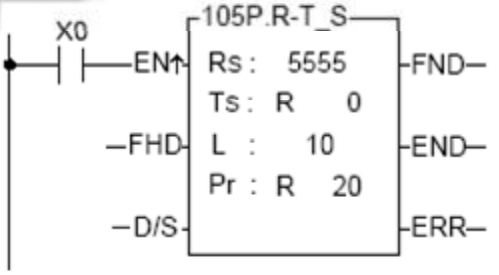


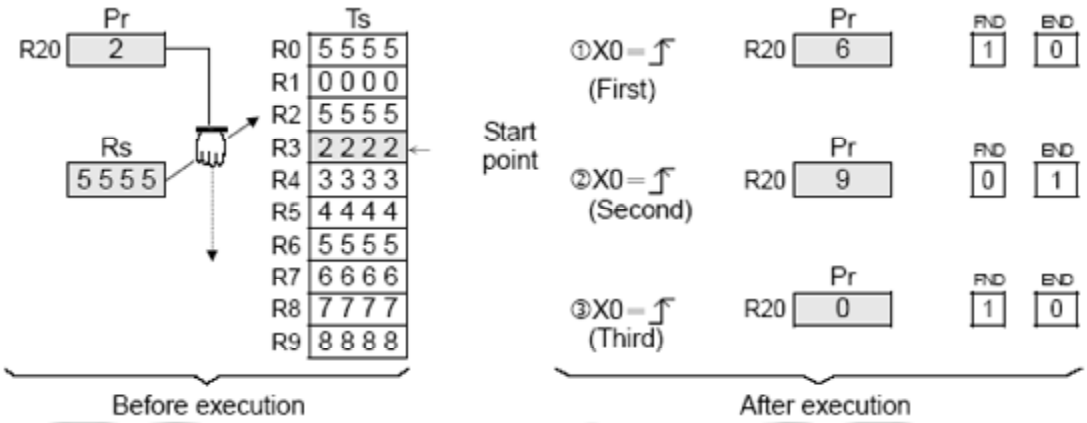
Function 105.REGISTER TO TABLE SEARCH



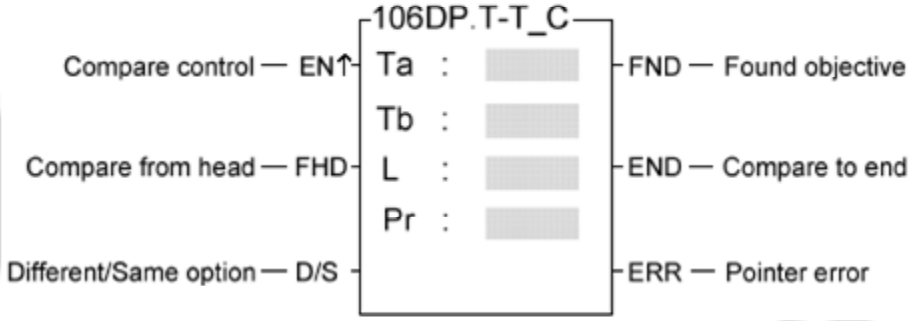
از این تابع برای یافتن مقداری مشخص در دسته ای از رجیسترهای متوالی ، استفاده می شود . هرگاه "EN" از 0 به 1 تغییر کند: وقتی 1="FHD" باشد یا مقدار اشاره گر (Pr) به L-1 رسیده باشد،جستجو از اولین رجیستر جدول Ts آغاز می شود. وقتی 0="FHD" و مقدار Pr کمتر از L-1 باشد،جستجو از ادامه رجیستری که Pr به آن اشاره کند (Pr+1) ، آغاز می شود . وقتی 1="D/S" ،جستجو یرای یافتن اولین رجیستری که محتوای آن با Rs متفاوت باشد صورت می گیرد. وقتی 0="D/S" ،جستجو یرای یافتن اولین رجیستری که محتوای آن با Rs یکسان باشد صورت می گیرد. پس از یافتن اطلاعات مورد نظر ،جستجو متوقف شده ، خروجی "FND" ، 1 شده و مقدار اشاره گر به رجیستر یافته شده ، اشاره می کند . وقتی Pr به L-1 رسید چه محتوای مورد نظر را یافته باشد چه نیافته باشد ،خروجی "END" فعال شده و جستجو متوقف می شود. L معرف طول جدول است .

مثال:

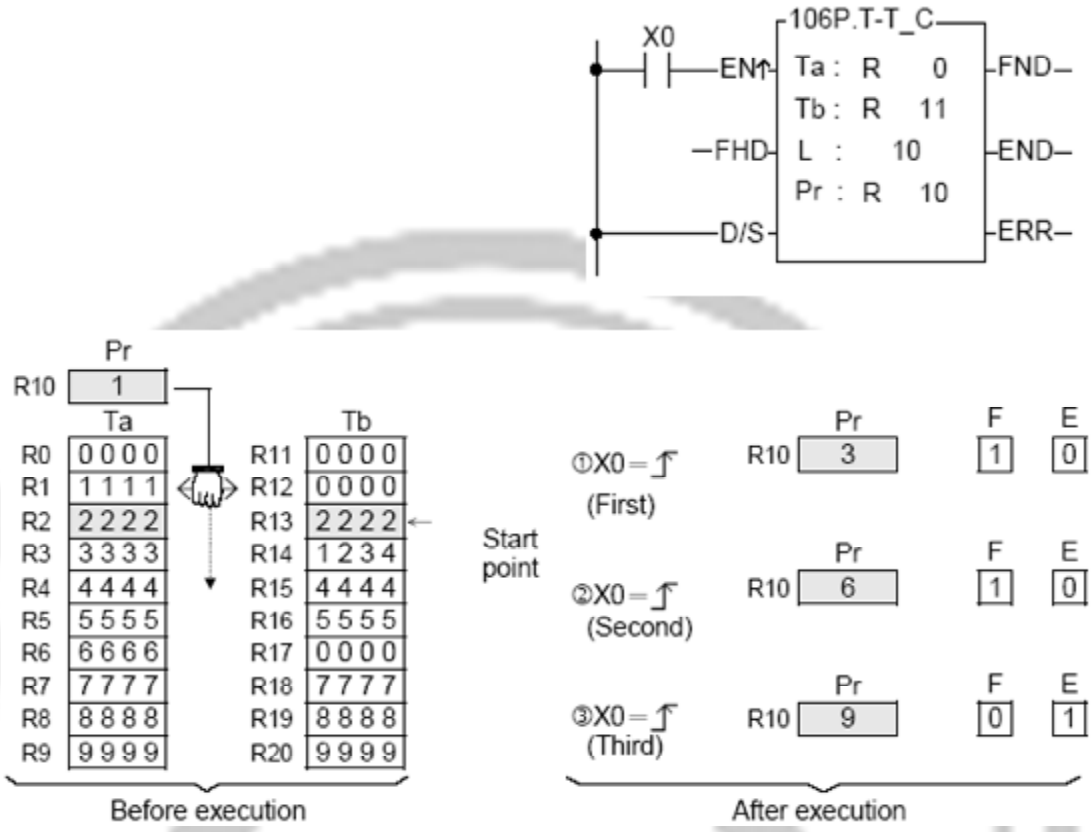




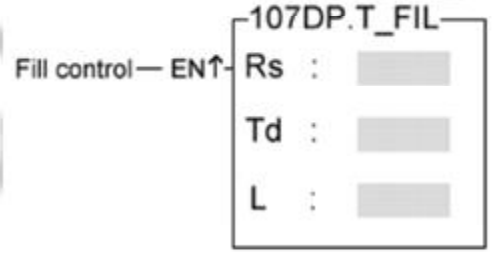
Function 106.TABLE TO TABLE COMPARE



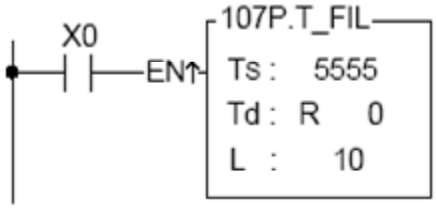
عملکرد این تابع مانند تابع قبل است با این تفاوت که محتوای رجیستر های معادل از دو جدول Ta و Tb با هم مقایسه می شوند. وقتی "D/S"=1 ،جستجو برای یافتن اولین دو رجیستر معادل در دو جدول که محتوای آن ها با هم متفاوت باشد ، صورت می گیرد. وقتی "D/S"=0 ، جستجو برای یافتن اولین دو رجیستر معادل در دو جدول که محتوای آن ها با هم یکسان باشد، صورت می گیرد.

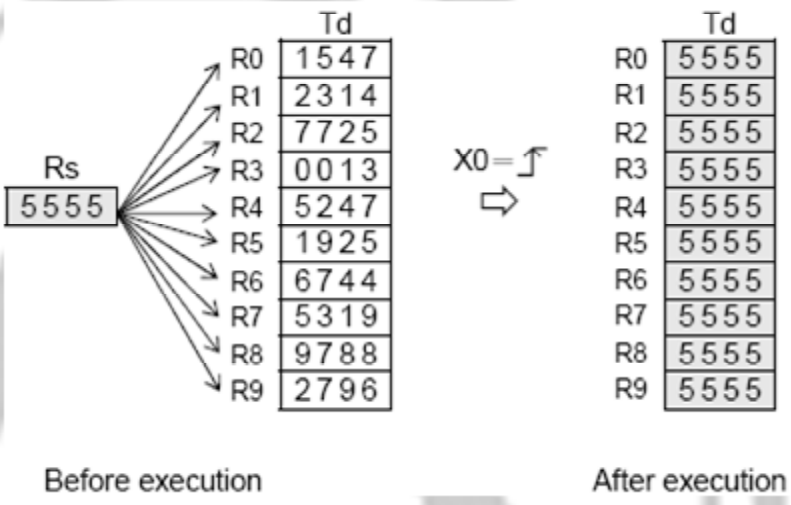


Function 107.TABLE FILL

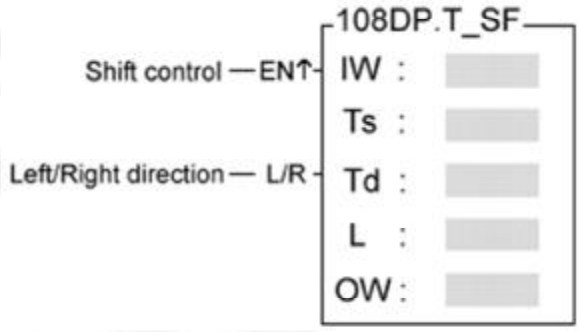


هرگاه "EN" از 0 به 1 تغییر کند: محتوای رجیستر Rs، تمام رجیسترهای جدول Td را پر می کند. کاربرد اصلی این تابع در صفر کردن کل یک جدول یا یکی کردن محتوای تمام رجیستر های یک جدول است. L معرف طول جدول است .



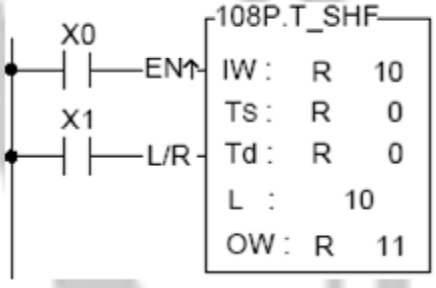


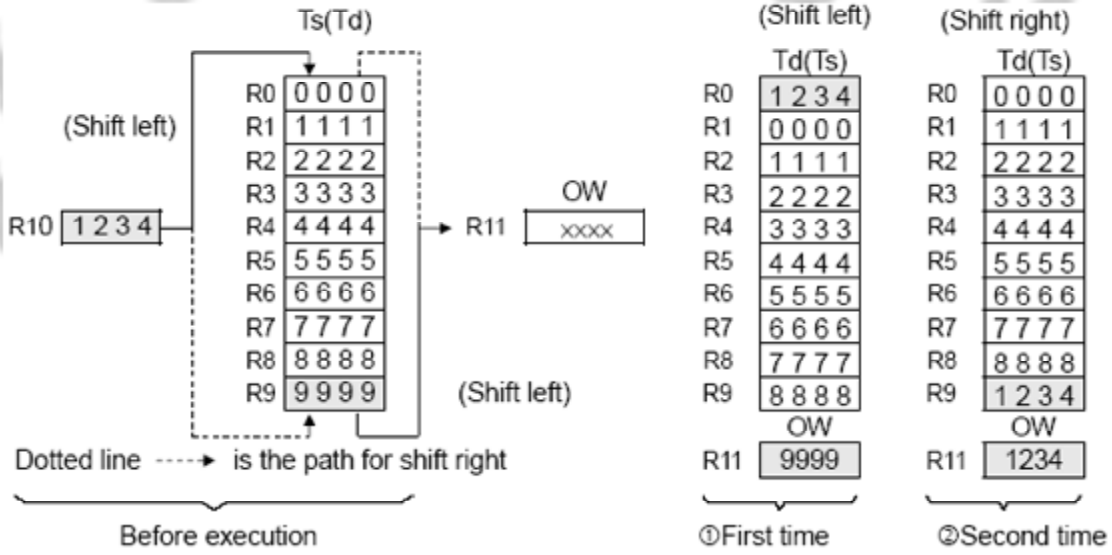
Function 108.TABLE SHIFT



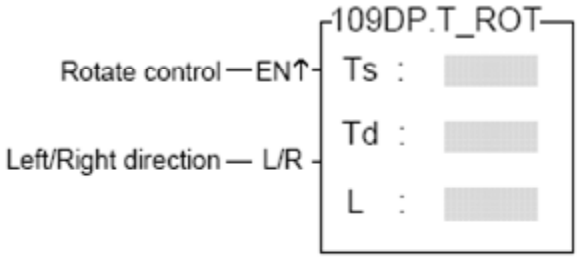
هرگاه "EN" از0 به 1 تغییر کند: محتوای رجیسترهای جدول Ts ، یک رجیستر به سمت چپ یا راست ، شیفت پیدا می کنند.وقتی 1="L/R" شیفت به چپ و وقتی 0="L/R" شیفت به راست صورت می گیرد. فضای خالی ایجاد شده بعد از اعمال شیفت ، توسط رجیستر یا عدد ثابت مشخص شده در IW ، پر می شود. محتوای رجیستری که پس از اعمال شیفت از 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. L معرف طول جدول است .

مثال:



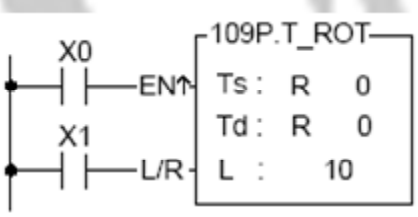


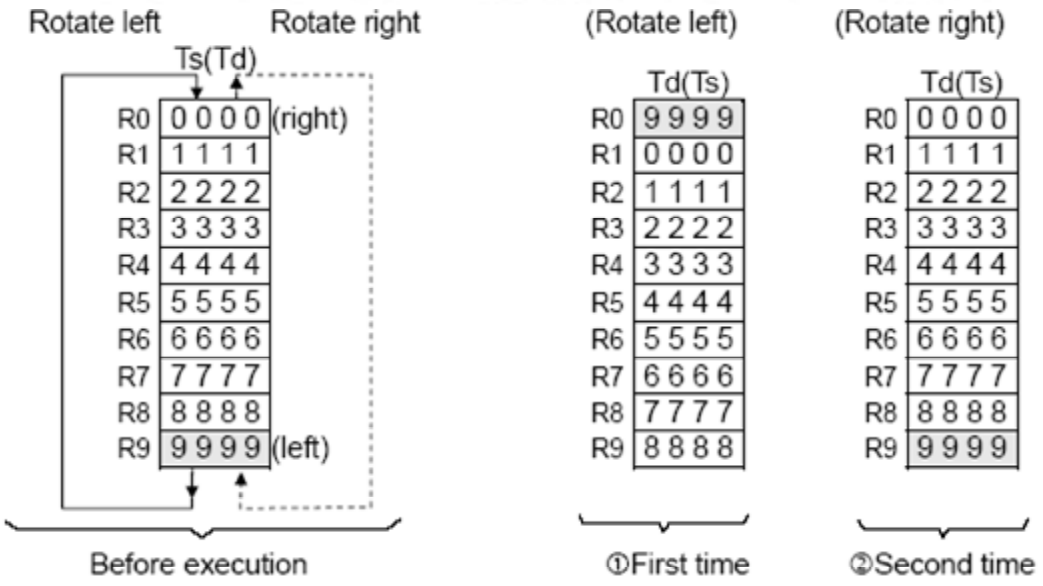
Function 109.TABLE ROTATE



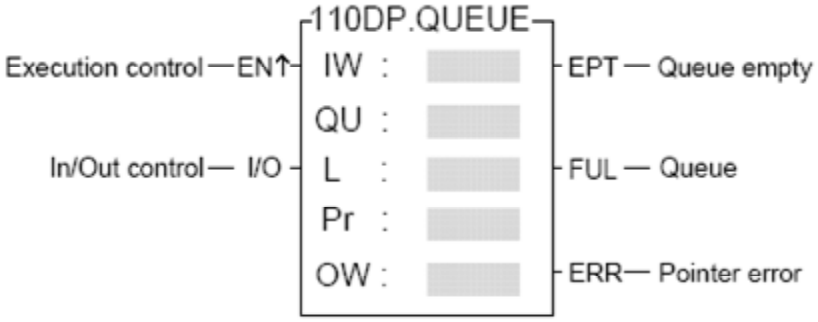
هرگاه "EN" از 0 به 1 تغییر کند: محتوای رجیسترهای جدول Ts ، یک رجیستر به سمت چپ یا راست می چرخند و نتیجه در Td ذخیره می شود. وقتی 1="L/R" ،چرخش به چپ صورت می گیرد . وقتی 0="L/R" ،چرخش به راست صورت می گیرد. L معرف طول جدول است .

مثال:





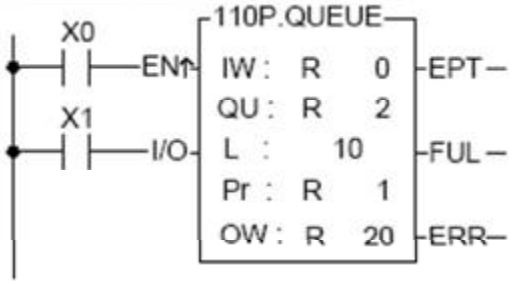
Function 110.QUENE

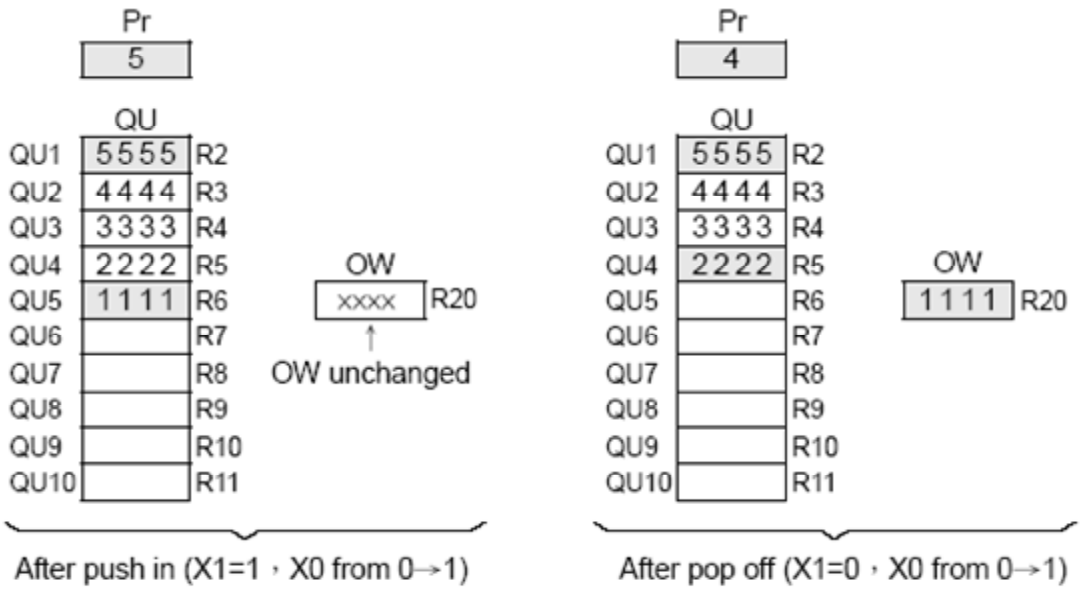


QUEUE نیز نوعی جدول است و همانطور که از نام آن پیداست ، عملکرد آن مانند یک صف می ماند . با کمک این تابع، همان داده ای که ابتدا در **QUEUE** وارد می شود ،(عمل **Push**) ،اولین داده ای خواهد بود که از **QUEUE** خارج می شود.(عمل **Pop**) . شماره رجیسترهای جدول معمولی از **0~L-1** است اما شماره خانه های **QUEUE** ،**1~L** بوده و **Pr=0** برای نمایش خالی بودن **QUEUE**استفاده می شود.

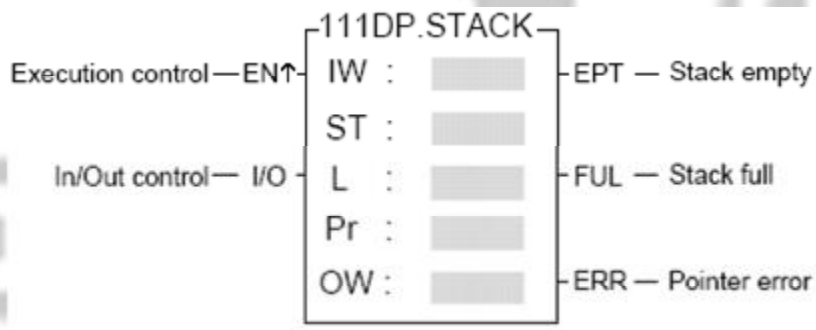
یک **QUEUE** از **L** رجیستر **16** یا **32** بیتی تشکیل شده که از رجیستر مشخص شده در **QU** شروع می شوند. هرگاه ورودی **1="I/O"** ، محتوای **IW** به داخل **QUEUE**، **push** می شود و هرگاه **0="I/O"** ،اولین داده ی درون **QUEUE** (قدیمی ترین) به داخل **OW**، **pop** خواهد شد. محتوای **IW** ،همیشه به داخل اولین رجیستر **QUEUE**، **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 **QUEUE** به **OW** ، **pop** می شود و یکی از مقدار **Pr** کم می شود. اگر **QUEUE** خالی از داده باشد (**Pr=0** باشد)، خروجی **1="EPT"** خواهد شد. اگر **QUEUE** کاملاً پر باشد (**Pr** به رجیستر **L** از **QUEUE** اشاره کند) ،خروجی **1="FUL"** خواهد شد. اگر مقدار **Pr** فراتر از رنج **0~L** داده شود،خروجی **1="ERR"** خواهد شد.

مثال:



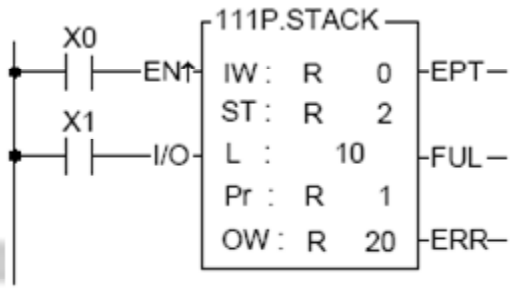


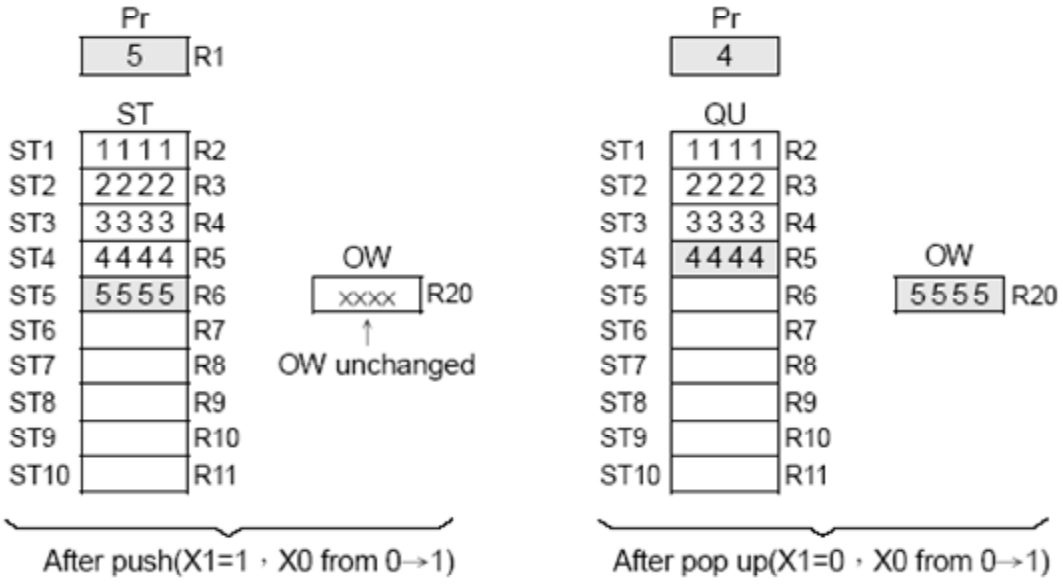
Function 111.STACK



STACK نیزجدولی مانند **QUEUE** است با 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 و همانطور که از نام آن پیداست ، داده ها مانند پشته ای به روی هم انبار شده و در هنگام نیاز ، از رو برداشته می شوند .





Function 112.DRUM

هرگاه "EN" از0 به 1 تغییر کند:

مقایسه بین محتوای Rs و بازه های مشخص شده در جدول Ts صورت می گیرد به عنوان مثال یک بازه شامل Ts0 (حد پایین) و Ts1 (حد بالا)می شود- بعد ازمقایسه بین Rs و اولین بازه ، نتیجه در D0 ذخیره می شود،سپس مقایسه

بین Rs و دومین بازه صورت می گیرد و نتیجه در D1 ذخیره می شود و... . اگر مقدار Rs میان حد بالا و حد پایین یک بازه باشد ،بیت D متناظر با آن بازه ،1 شده، در غیر این صورت 0 خواهد شد.

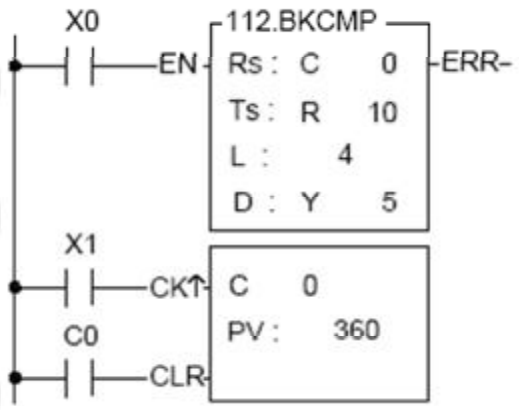
	Upper limit	Lower limit	Compare	Compared value	Result
0	T_{S1}	T_{S0}	⟷	Rs	D_0
1	T_{S3}	T_{S2}	⟷		D_1
≀	≀	≀	≀		≀
L–1	T_{S2L-1}	T_{S2L-2}	⟷		D_{L-1}

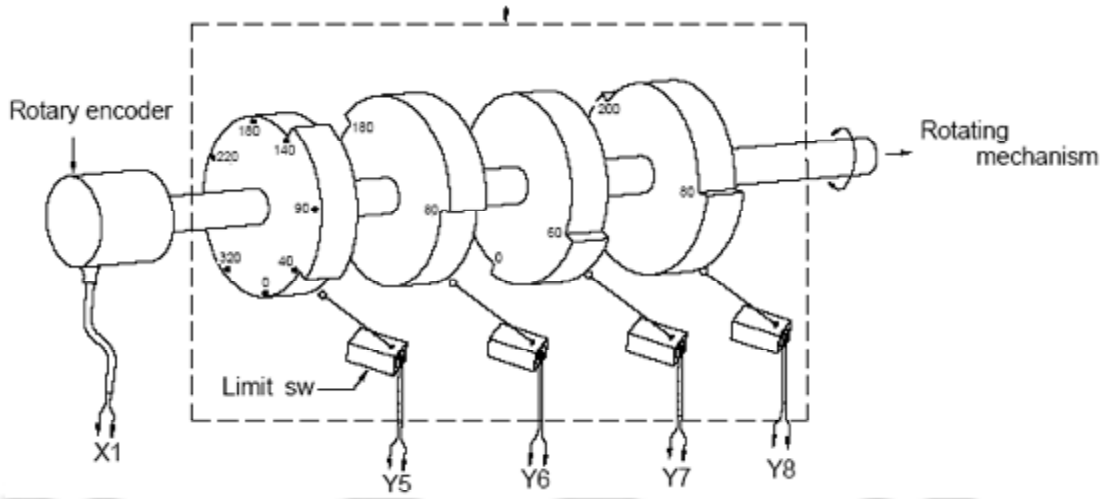
L در این تابع تعداد بازه ها را مشخص می کند. مقایسه تا زمانی ادامه می یابد که Rs با تمام بازه ها مقایسه شود.

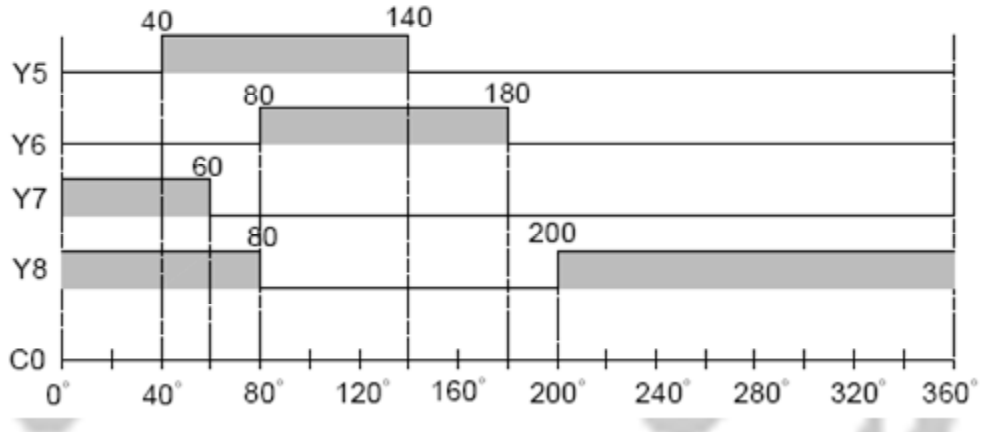
هرگاه بیت M1975=0 باشد، در یک بازه ، حد بالا نمی تواند کوچک تر از حد پایین باشد،در غیر این صورت خروجی "ERR" فعال شده و نتیجه مقایسه برای آن بازه ،0 خواهد شد. هرگاه بیت M1975=1 باشد، ،محدودیتی برای بزرگتر یا کوچکتر بودن حد بالا نسبت به حد پایین وجود نداشته و تنها کافیست مقدار Rs در بازه باشد تا رجیستر معادل آن در D ، 1 شود. این حالت می تواند برای سوئیچ DRUM الکترونیکی چرخنده در محور 360^{o} ،کاربرد داشته باشد.

این تابع در واقع سوئیچی برای محورهای الکترونیکی محسوب می شود که اگر به همراه برنامه INTERRUPT و دستور IMMIDIATE I/O(تابع 74) به کار گرفته شود ، می تواند محور الکترونیکی دقیقی به دست دهد.

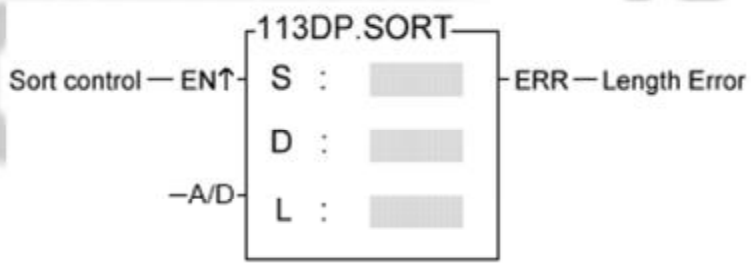
مثال:





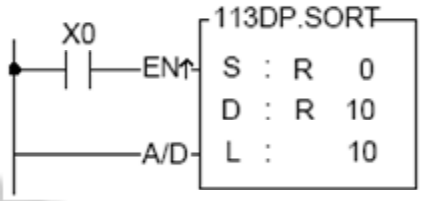


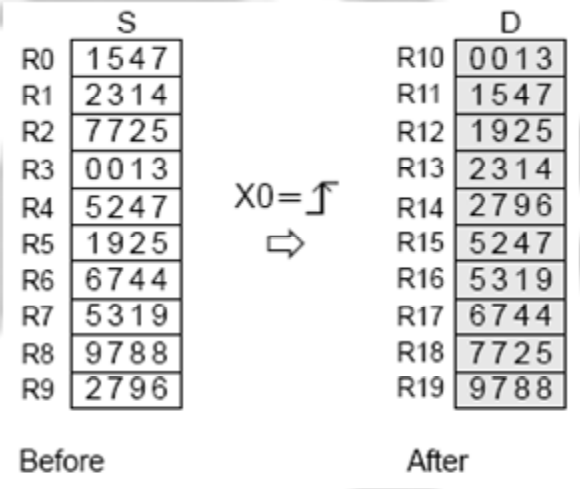
Function 113.DATA SORTING



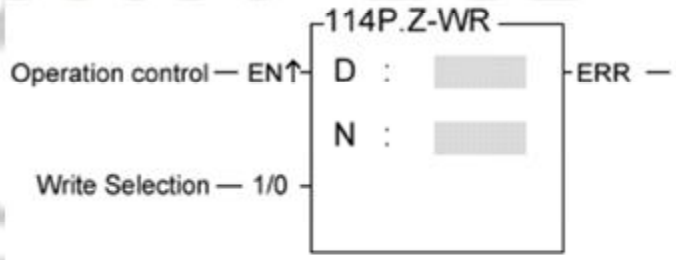
هرگاه "EN" از0 به 1 تغییر کند: این تابع مقادیر رجیستر ها به تعداد L را که از S شروع می شوند ،به صورت افزایشی یا کاهشی مرتب می کند و نتیجه را در D ذخیره می کند. اگر 1 "A/D" رجیسترها به صورت افزایشی مرتب می شوند و اگر 0="A/D" رجیسترها به صورت کاهشی مرتب می شوند . L باید در رنج 127~2 باشد در غیر این صورت خروجی 1="ERR" می شود .

مثال:





Function 114.ZONE WRITE



هرگاه "EN" از0 به 1 تغییر کند: این تابع N تعداد از رجیسترها که از D شروع می شوند را set (1) یا reset (0) می کند. اگر ورودی 1="1/0" باشد،set کرده و اگر 0="1/0" باشد، reset می کند.

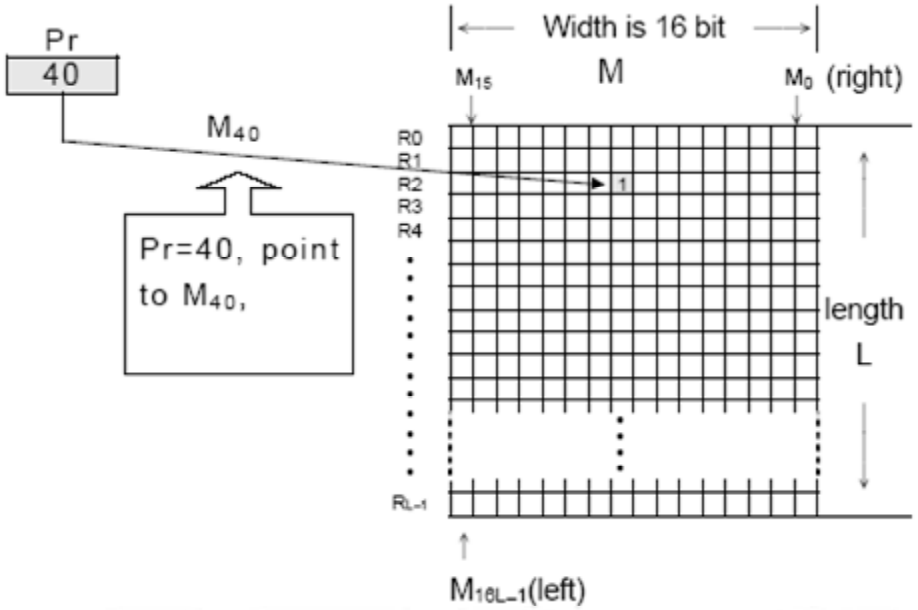
مثال:

در این مثال هرگاه X0=1 ،رجیستر های R0~R9 ،reset شده و 0 می شوند.

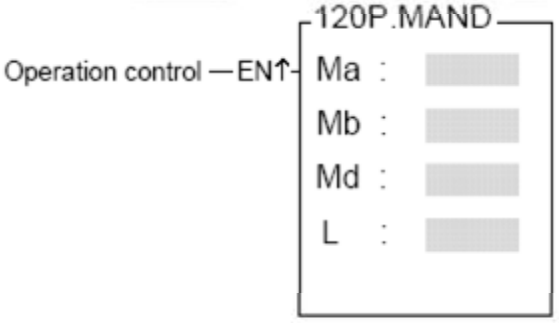
توابع ماتریسی

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			

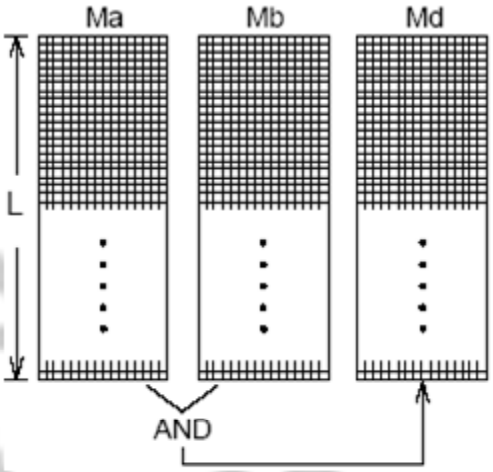
یک ماتریس از 2 یا چند رجیستر پی در پی 16 بیتی تشکیل شده است.به تعداد رجیسترهای تشکیل دهنده ماتریس، طول ماتریس گفته و با (L) نمایش می دهند. پس یک ماتریس 16×L بیت (آرایه) دارد و واحد اصلی اجرای این توابع ،بیت می باشد. توابع ماتریسی عمدتاً برای پردازش گسسته استفاده می شوند مانند انتقال ،کپی، مقایسه ، جستجو و غیره یک آرایه به ماتریس یا ماتریس به ماتریس. در میان این توابع ،بیشتر آنها به یک اشاره گر(pointer) 16 بیتی نیاز دارند تا به یک آرایه خاص در یک ماتریس اشاره کنند. محدوده موثر این اشاره گر ،0~16L-1 است. در اجرای دستوراتی مانند شیفت و چرخش،حرکت به سمت بیت کم ارزش را ،جهت راست و حرکت به سمت بیت با ارزش تر را جهت چپ می دانیم.



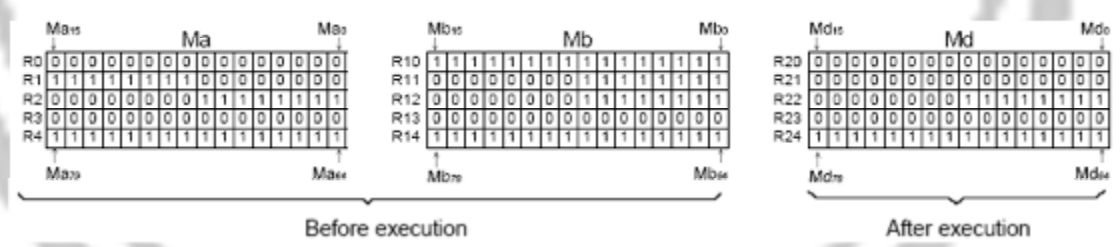
Function 120.MATRIX AND



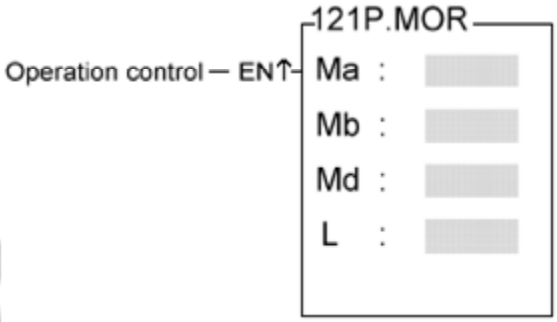
هرگاه "EN" از 0 به 1 تغییر کند: بیت های متناظر در ماتریس Ma و Mb را با یکدیگر AND کرده و نتیجه در Md ریخته می شود.



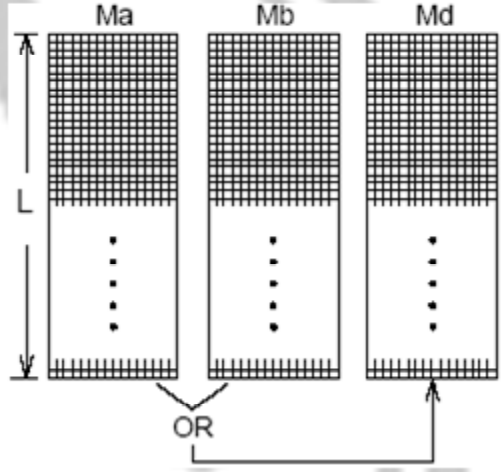
مثال:



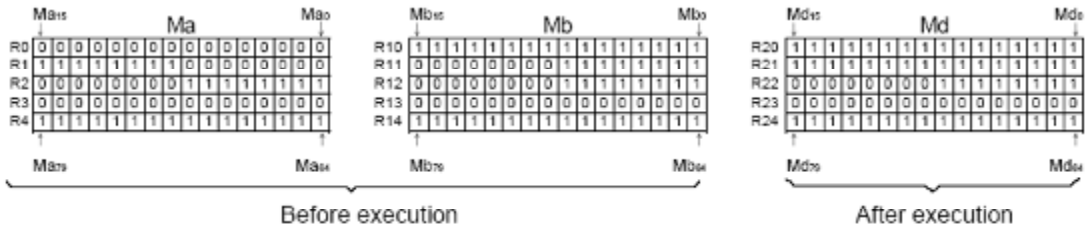
Function 121.MATRIX OR



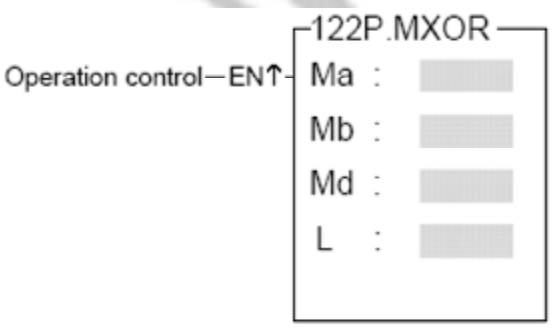
هرگاه "EN" از 0 به 1 تغییر کند: بیت های متناظر در ماتریس Ma و Mb را با یکدیگر OR کرده و نتیجه در Md ریخته می شود.



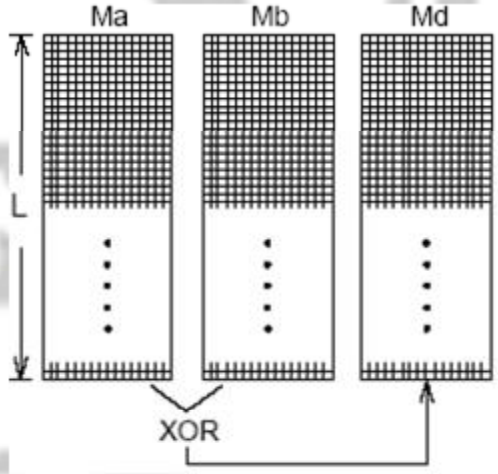
مثال:



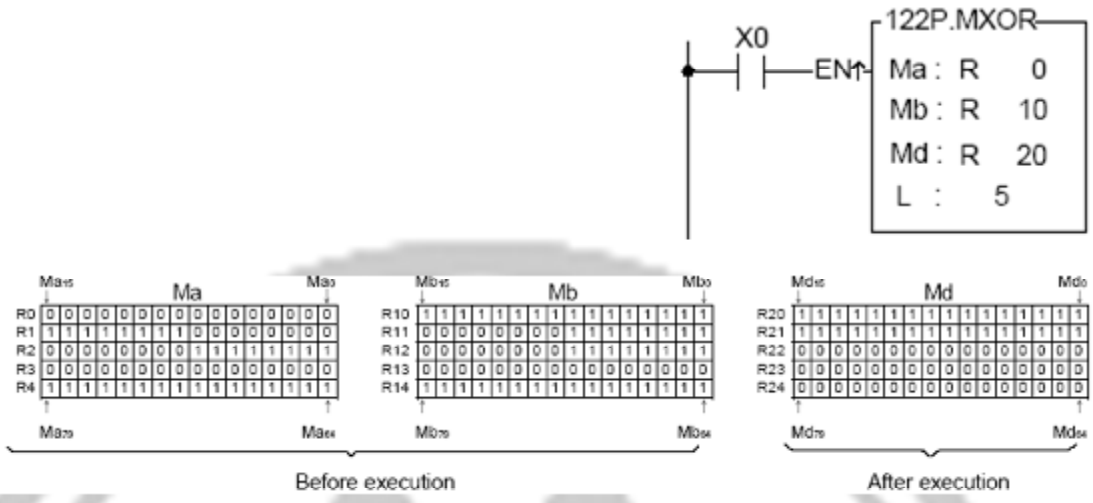
Function 122.MATRIX XOR



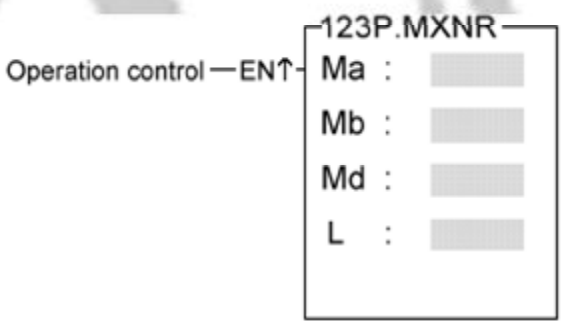
هرگاه "EN" از 0 به 1 تغییر کند: بیت های متناظر در ماتریس Ma و Mb را با یکدیگر XOR کرده و نتیجه در Md ریخته می شود.



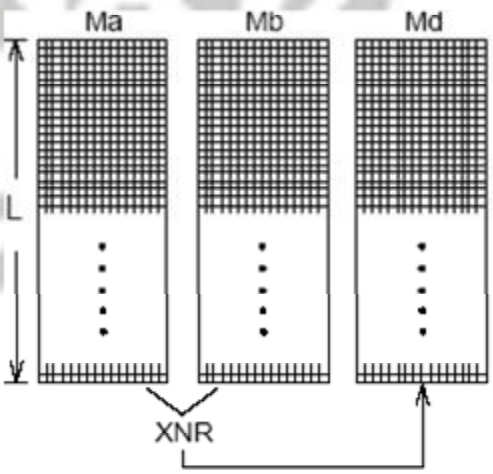
مثال:



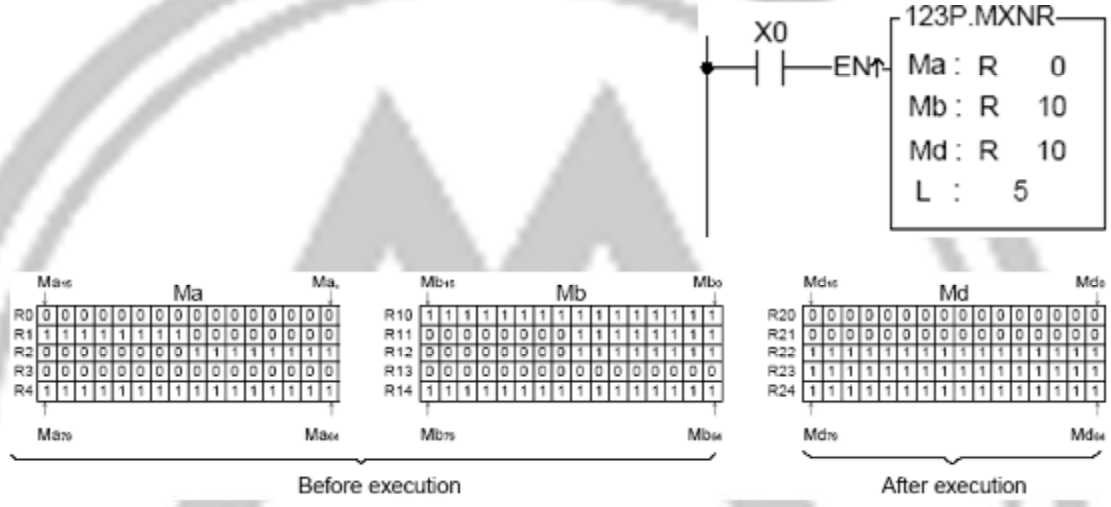
Function 123.MATRIX XNOR



هرگاه "EN" از 0 به 1 تغییر کند: بیت های متناظر در ماتریس Ma و Mb را با یکدیگر XNOR کرده و نتیجه در Md ریخته می شود.

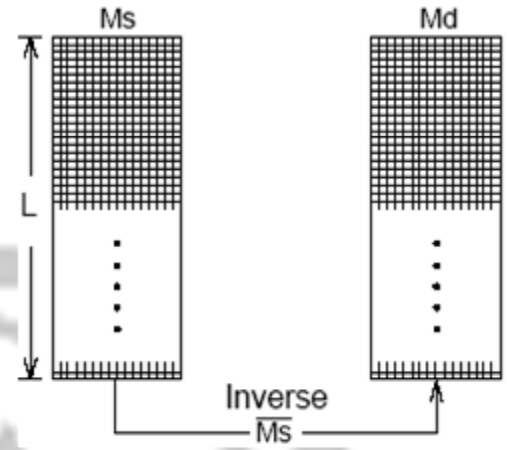


مثال:

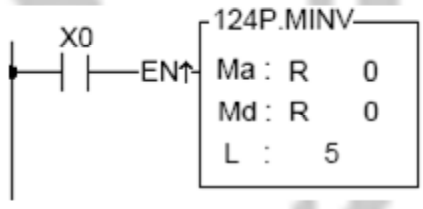


Function 124.MATRIX INVERSE

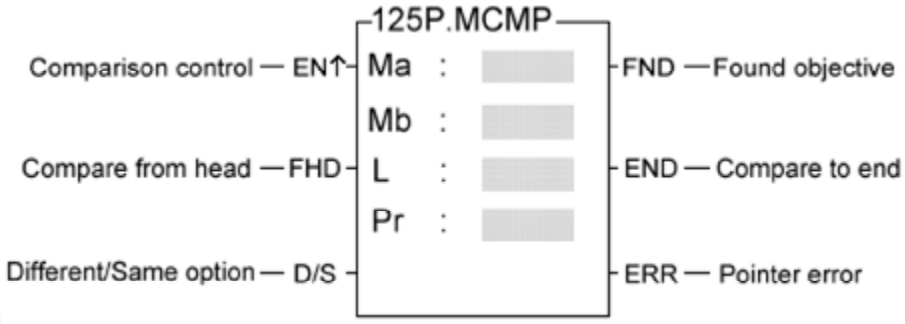
هرگاه "EN" از 0 به 1 تغییر کند: تمام بیت های ماتریس Ms معکوس می شوند(0 ها 1 شده و 1 ها 0 می شوند) و نتیجه در Md ذخیره می شود.



مثال:

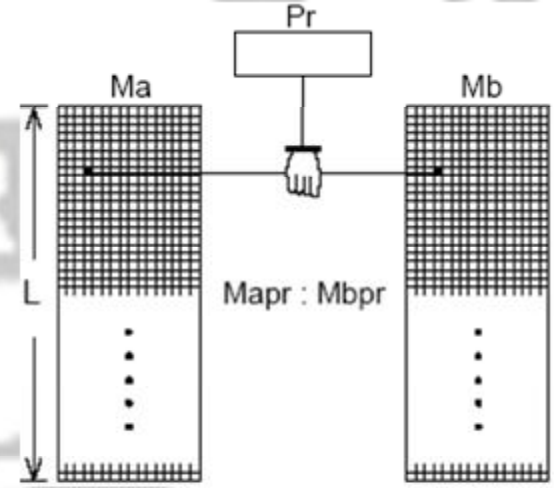


Function 125.MATRIX COMPARE

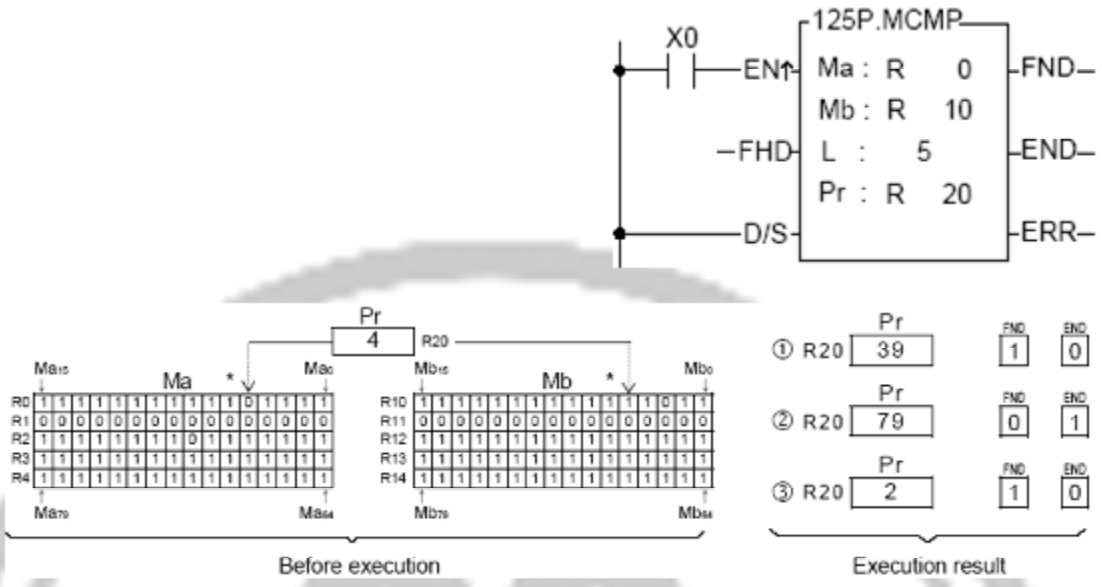


هرگاه "EN" از0 به 1 تغییر کند: مقایسه بین دو بیت متناظر از ماتریس های Ma و Mb آغاز می شود. اگر ورودی 1="FHD" باشد یا مقدار Pr به 16L-1 رسیده باشد،مقایسه از ابتدای ماتریس ها (اولین بیت) ، شروع می شود. وقتی 0="FHD" و مقدار Pr کمتر از 16L-1 باشد،مقایسه از اولین بیت ،بعد از جایی که Pr اشاره می کند (Pr+1)،شروع می شود. وقتی 1="D/S" ،مقایسه برای یافتن اولین جفت بیت متناظری که محتوای آنها متفاوت باشد صورت می گیرد. وقتی 0="D/S" ،مقایسه برای یافتن اولین جفت بیت متناظری که محتوای آنها یکسان باشد صورت می گیرد. بعد از یافتن اطلاعات مورد نظر ،مقایسه متوقف شده ،خروجی "FND"، 1 شده و مقدار اشاره گر به آرایه یافت شده ، اشاره می کند .

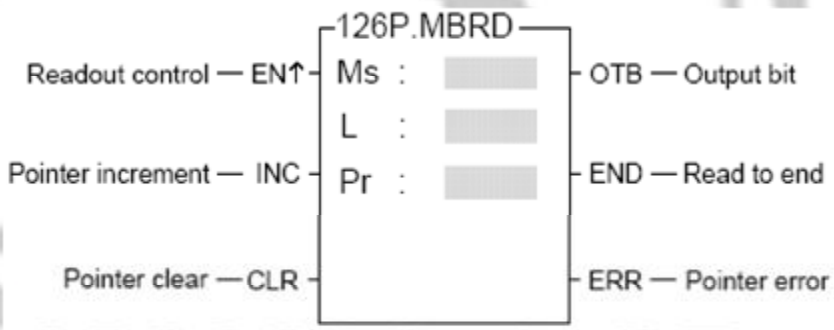
وقتی Pr به 16L-1 رسید،چه بیت های مورد نظر را یافته باشد چه نیافته باشد،خروجی "END" فعال شده و مقایسه متوقف می شود. اگر مقدار اشاره گر خارج از محدوده 16L-1~0 باشد ، خروجی "ERR" فعال می شود.



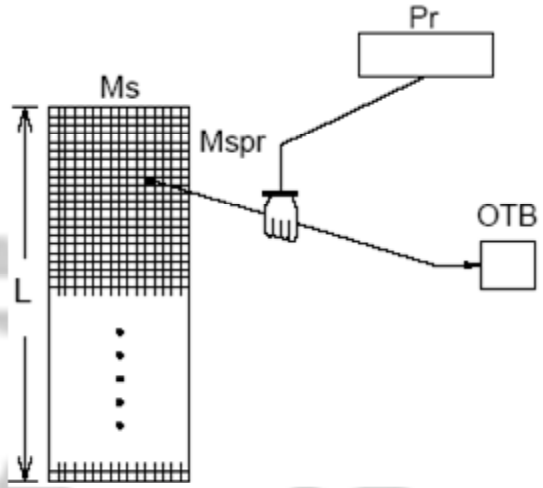
مثال:



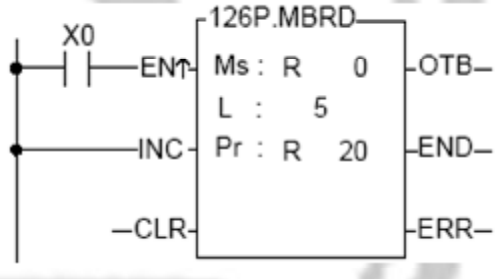
Function 126.MATRIX BIT READ

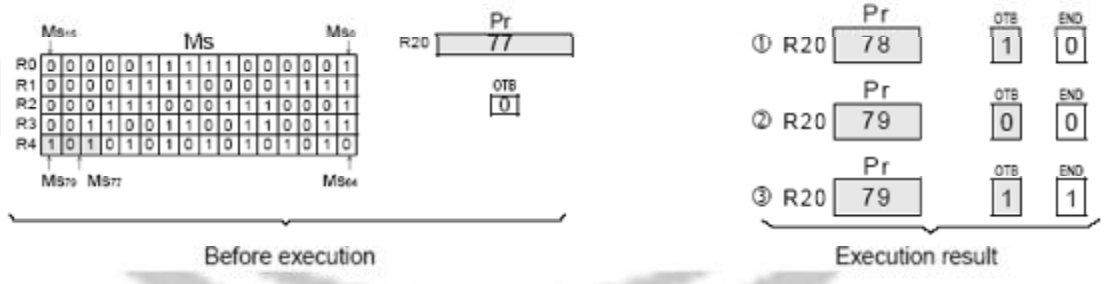


از این تابع برای خواندن مقدار یک بیت مشخص در میان دسته ای از رجیسترها ، استفاده می شود . هرگاه "EN" از 0 به 1 تغییر کند: ورودی "CLR" را چک کرده و اگر 1="CLR"باشد ، مقدار Pr را 0 می کند. مقدار بیتی که Pr به آن اشاره می کند،در خروجی "OTB" ظاهر می شود. اگر مقدار Pr به 16L-1 برسد(آخرین بیت ماتریس) خروجی "END" 1، شده و اجرای این تابع پایان می یابد. اگر مقدار Pr کمتر از 16L-1 باشد،ورودی "INC" را چک می کند که اگر این ورودی 1 باشد ،مقدار Pr افزایش می یابد. محدوده موثر Pr، 16L-1~0 است و فراتر از این محدوده، خروجی 1="ERR" شده و این تابع اجرا نمی شود.

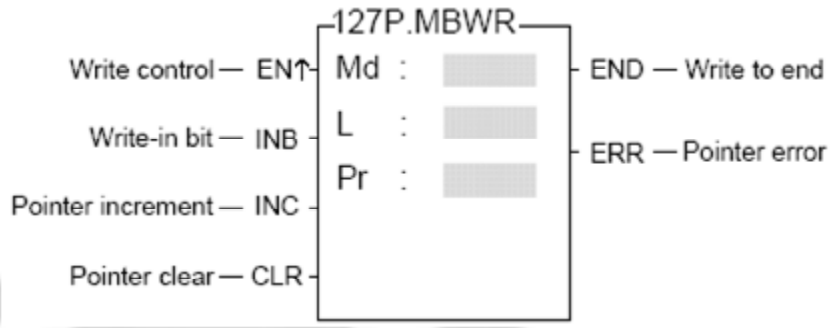


مثال:

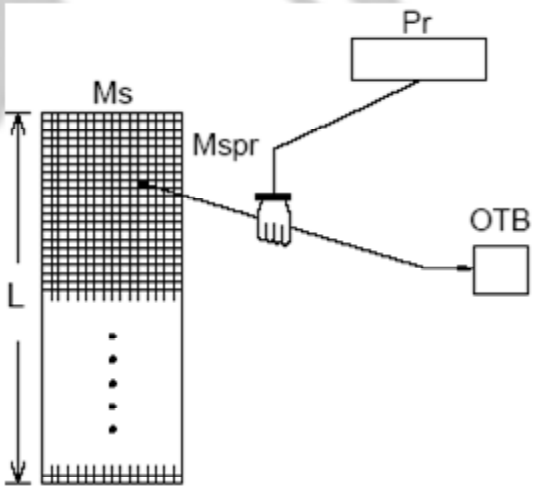




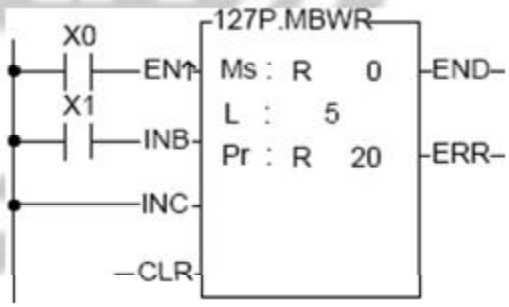
Function 127.MATRIX BIT WRITE

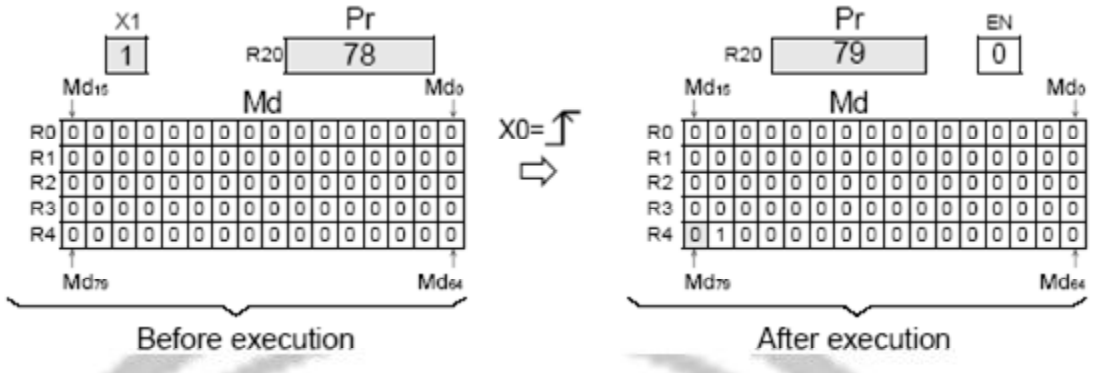


مانند تابع قبل است با این تفاوت که هنگام اجرا ، بیت مشخص شده در ورودی **"INB"** به روی بیتی که **Pr** اشاره می کند ، ریخته می شود.

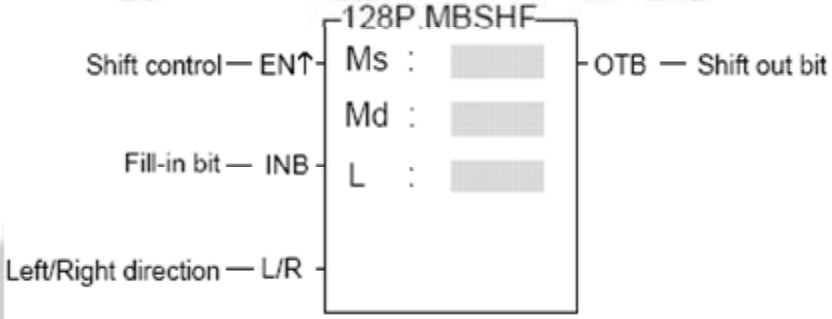


مثال:

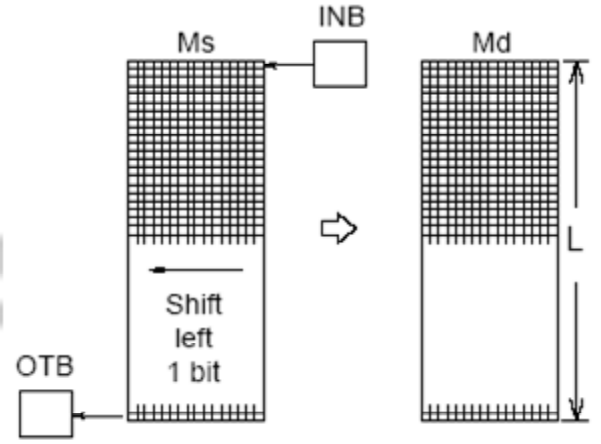


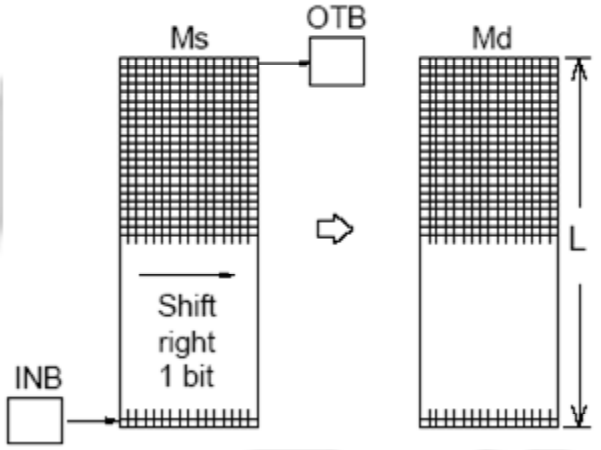


Function 128.MATRIX BIT SHIFT

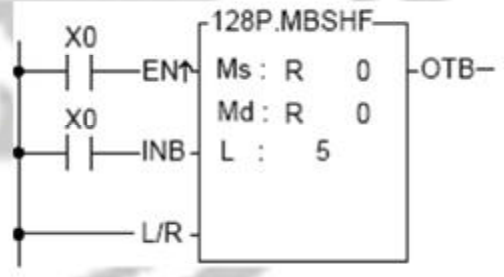


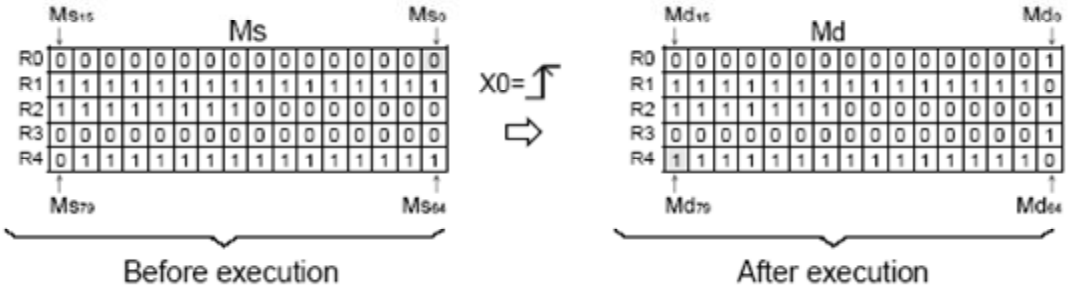
هرگاه "EN" از0 به 1 تغییر کند: بیت های ماتریس Ms، 1 بیت شیفت پیدا کرده و نتیجه در Md ذخیره می شود. اگر ورودی 1="L/R" باشد،شیفت به چپ صورت می گیرد و اگر ورودی 0="L/R" باشد،شیفت به راست صورت می گیرد . بیت خالی ایجاد شده بعد از اعمال شیفت ،توسط ورودی "INB" پر شده و محتوای بیتی که پس از اعمال شیفت از جدول خارج می شود،به خروجی "OTB" منتقل می شود.


INB
Ms
Md
L
Shift
left
1 bit
OTB

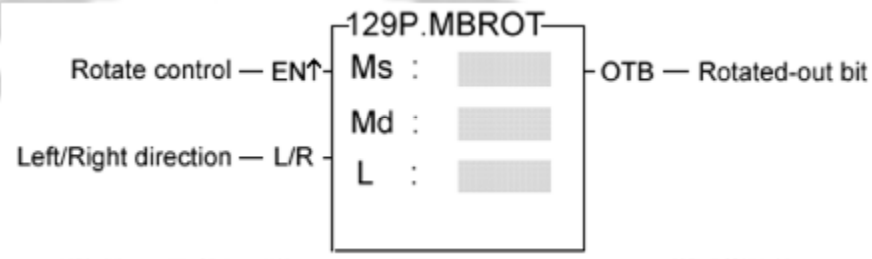

OTB
Ms
Md
L
Shift
right
1 bit
INB

مثال:


128P.MBSHF
X0
EN
Ms : R 0
OTB
Md : R 0
X0
INB
L : 5
L/R

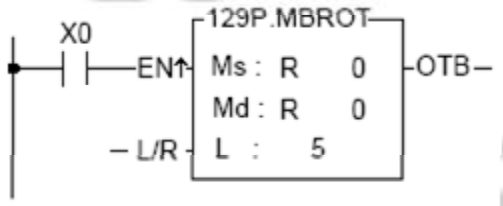

Ms
Md
X0=
Before execution
After execution

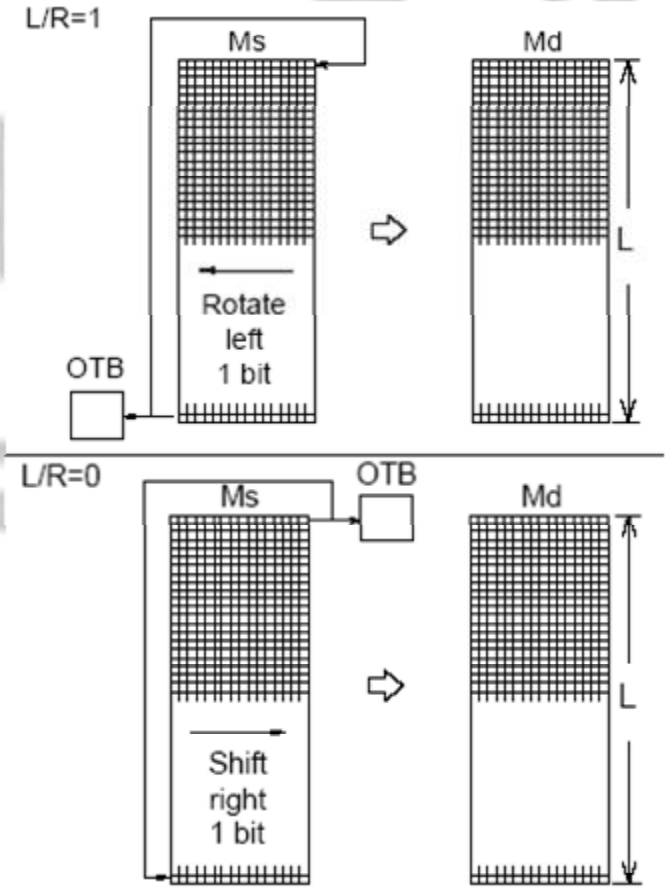
Function 129.MATRIX BIT ROTATE

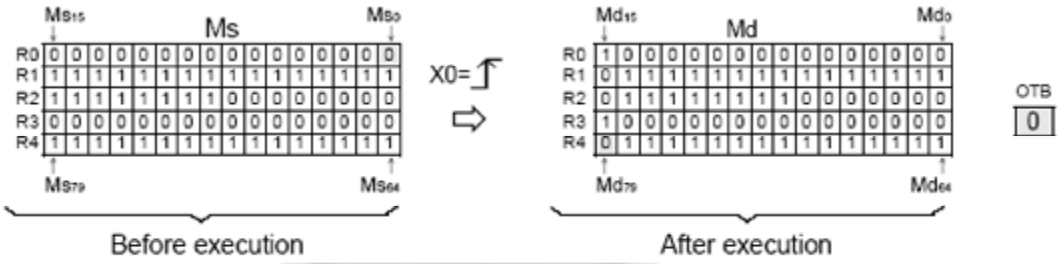


هرگاه "EN" از0 به 1 تغییر کند: بیت های ماتریس Ms ، یک بیت به سمت راست یا چپ می چرخند و نتیجه در ماتریس Md ذخیره می شود. وقتی ورودی 1="L/R"، چرخش به چپ صورت می گیرد. وقتی ورودی 0="L/R"، چرخش به راست صورت می گیرد. یک کپی از بیتی که بر اثر چرخش از یک سر ماتریس خارج شده و در سر دیگر آن قرار می گیرد،به خروجی "OTB" نیز منتقل می شود.

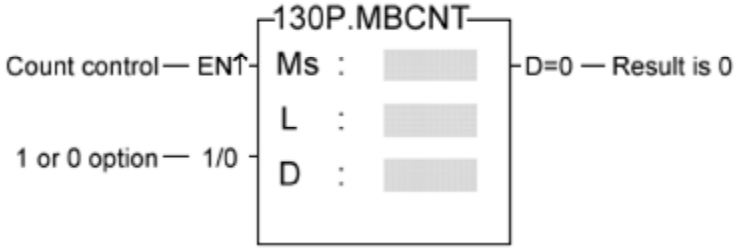
مثال:





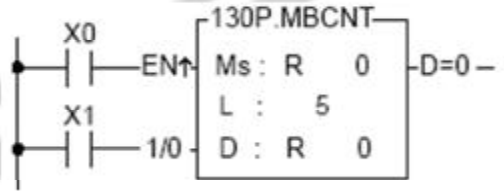


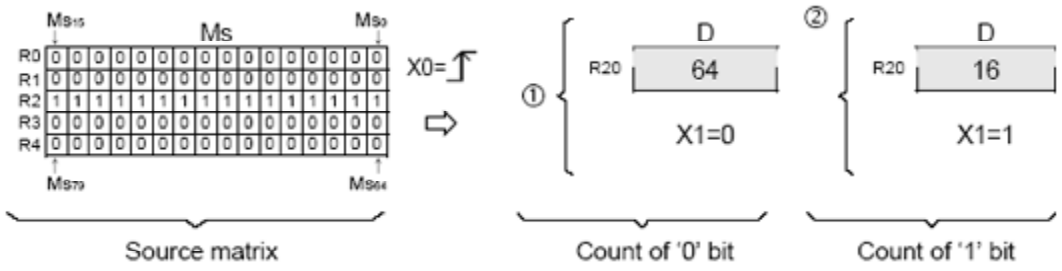
Function 130.MATRIX BIT STATUS COUNT



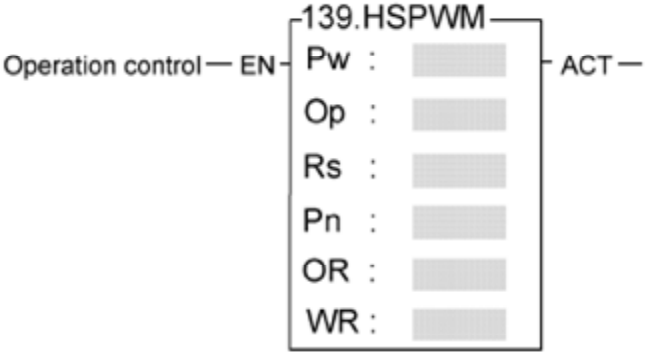
هرگاه "EN" از 0 به 1 تغییر کند: اگر 1="1/0" باشد،تعداد بیت های 1 موجود در ماتریس Ms را شمرده و تعداد در D ذخیره می شود. اگر 0="1/0" باشد،تعداد بیت های 0 موجود در ماتریس Ms را شمرده و تعداد در D ذخیره می شود. اگر هیچ تعداد از آن بیت مورد نظر موجود نباشد، خروجی "D=0" ، 1 خواهد شد.

مثال:





Function 139.HSPWM



این تابع برای فرستادن پالس به خروجی های ترانزیستوری PLC استفاده می شود . وقتی 1="EN" ، این تابع پالسی را به خروجی می فرستد که فرکانس و پریود آن بر اساس فرمول قابل محاسبه است .

Function 140 & 141 . HIGH SPEED PULSE OUTPUT

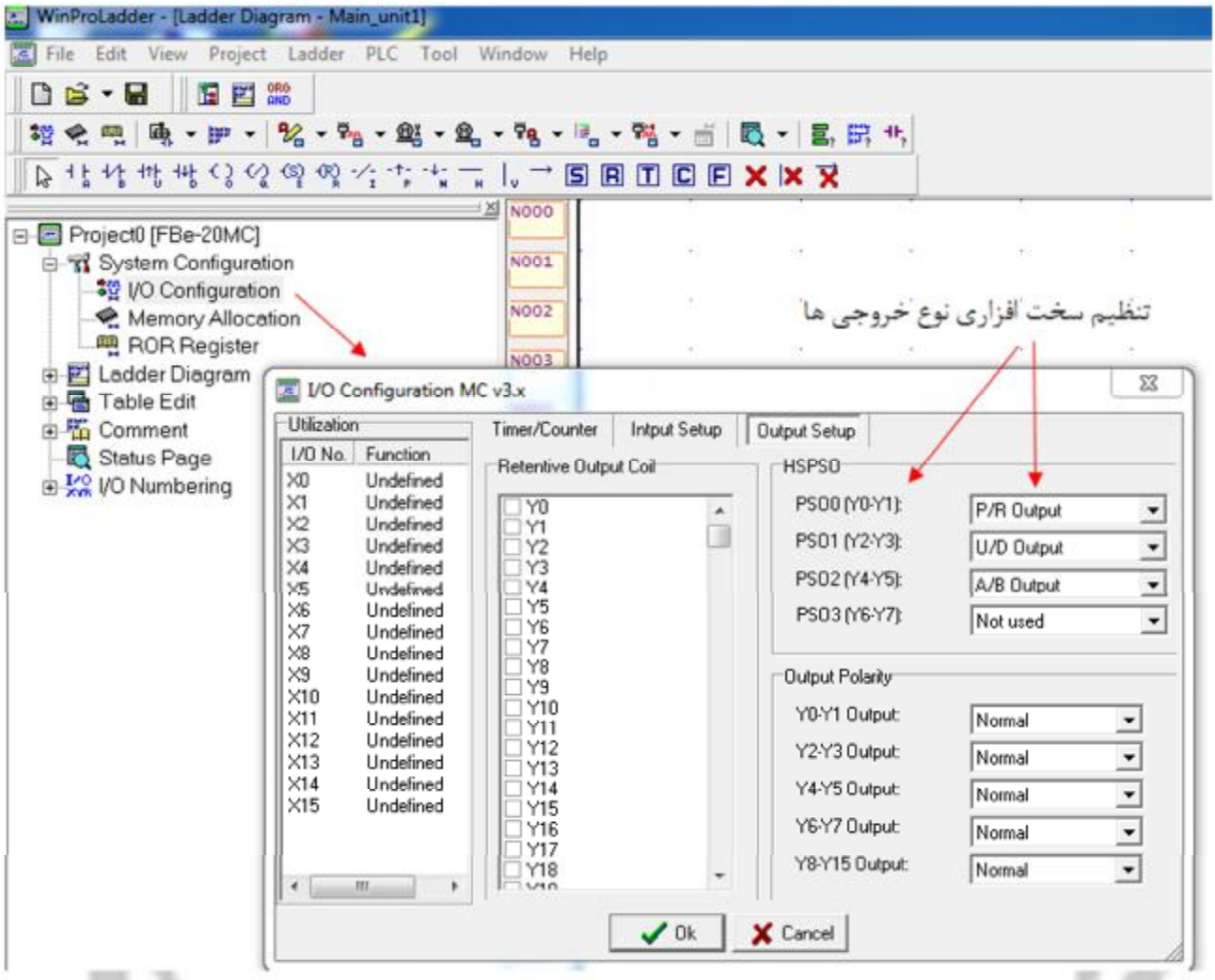
- **جدول زیر حالات مختلف خروجی ها را نشان می دهد**

Axis No.	Exterior output	Output modes			Remark
		U/D output	K/R output	A/B output	
PSO0	Y0 , Y1	Y0=U , Y1=D	Y0=K , Y1=R	Y0=A , Y1=B	Valid for all FBx-xxMCT main unit
PSO1	Y2 , Y3	Y2=U , Y3=D	Y2=K , Y3=R	Y2=A , Y3=B	Not for FBE-20MCT & FBN-19MCT.
PSO2	Y4 , Y5	Y4=U , Y5=D	Y4=K , Y5=R	Y4=A , Y5=B	Only for FBE-40MCT & FBN-36MCT.
PSO3	Y6 , Y7	Y6=U , Y7=D	Y6=K , Y7=R	Y6=A , Y7=B	

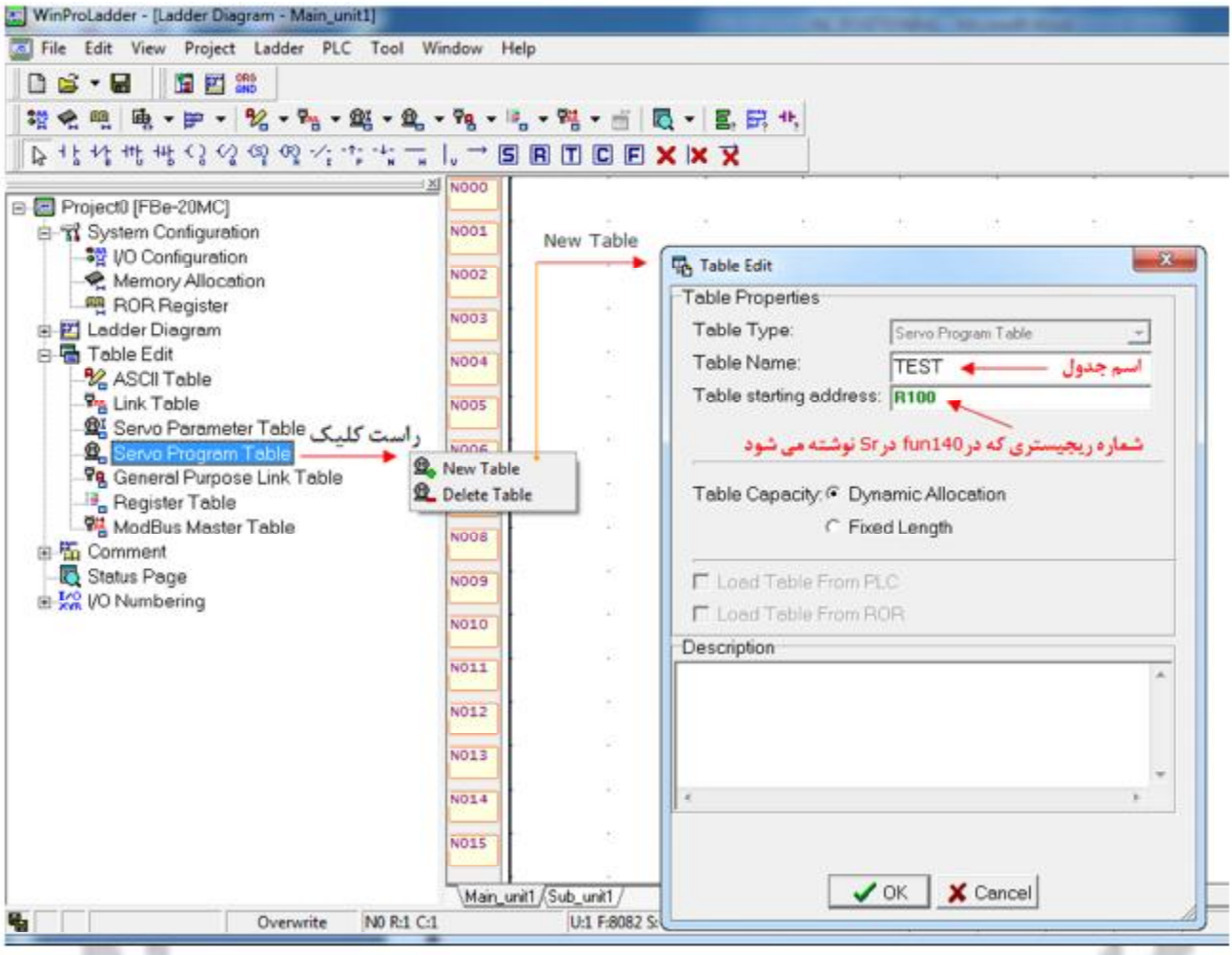
U/D	پالس بالا رونده در خروجی ظاهر میشود(Y0(Y2,Y4,Y6 پالس پایین رونده در خروجی ظاهر میشود(Y1(Y3,Y5,Y7
P/R	پالس درخروجی ظاهر میشود(Y0(Y2,Y4,Y6 جهت چرخش را تعیین می نماید(Y1(Y3,Y5,Y7 ON : شمارش بالا رونده OFF : شمارش پایین رونده
A/B	پالس با فازA در خروجی ظاهر میشود(Y0(Y2,Y4,Y6 پالس با فازB در خروجی ظاهر میشود(Y1(Y3,Y5,Y7

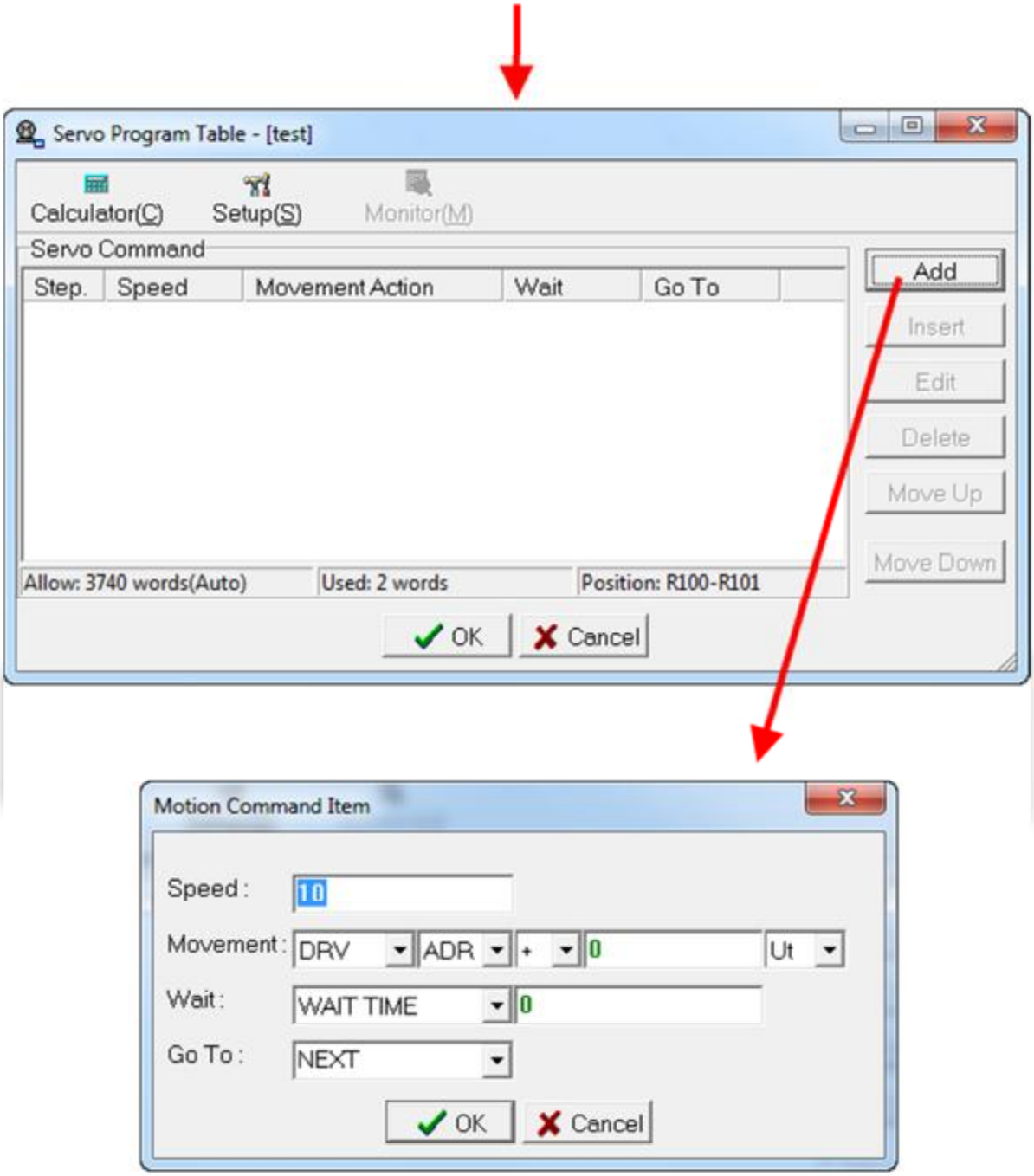
- نحوه تنظیم نمودن خروجیها توسط Winproladder

با انتخاب Output Setup از منوی I/O Configuratuin میتوانید این کار را انجام دهید



- قبل از استفاده از FUN 140 لازم است ابتدا "Servo Program Table" را برنامه ریزی نمایید،این جدول به منظور تعیین پارامترهای مورد نیاز خروجی پالس (ازقبیل : سرعت،فرکانس،وقفه های کاری،جهت چرخش و...) تعبیه گردیده است و FUN 140 پس از فعال شدن پارامترهای ذکر شده را از این جدول می خواند.


WinProLadder - [Ladder Diagram - Main_unit1]
File Edit View Project Ladder PLC Tool Window Help
Project0 [FBe-20MC]
System Configuration
I/O Configuration
Memory Allocation
ROR Register
Ladder Diagram
Table Edit
ASCII Table
Link Table
Servo Parameter Table
Servo Program Table
General Purpose Link Table
Register Table
ModBus Master Table
Comment
Status Page
I/O Numbering
راست کلیک
New Table
Delete Table
New Table
Table Edit
Table Properties
Table Type:
Servo Program Table
Table Name:
TEST
اسم جدول
Table starting address:
R100
شماره ریجیستری که در fun140 در Sr نوشته می شود
Table Capacity:
Dynamic Allocation
Fixed Length
Load Table From PLC
Load Table From ROR
Description
OK
Cancel
Main_unit1
Sub_unit1
Overwrite
N0 R:1 C:1


Servo Program Table - [test]
Calculator(C)
Setup(S)
Monitor(M)
Servo Command
Step.
Speed
Movement Action
Wait
Go To
Add
Insert
Edit
Delete
Move Up
Move Down
Allow: 3740 words(Auto)
Used: 2 words
Position: R100-R101
OK
Cancel
Motion Command Item
Speed :
10
Movement :
DRV
ADR
+
0
Ut
Wait :
WAIT TIME
0
Go To :
NEXT
OK
Cancel

شرح پارامترهای تنظیمی:

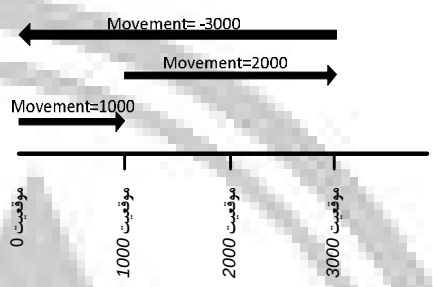
(رجیسترهایی که برای این جدول استفاده می شوند بصورت **32** بیتی می باشند)

پارامتر	شرح
Speed	فرکانس پالس خروجی
DRVC/DRV	** DRVC به منظور اعمال تغییرات سرعت بصورت ترتیبی به کار میرود(8 تغییر سرعت در بیشترین حالت) ** به منظور تغییر سرعت بصورت ترتیبی ، تنها اولین دستور DRVC میتواند مقدار معین (Absolute) را جهت هماهنگی تعیین مکان بکار برد ** جهت چرخش در DRVC تنها توسط + و- تعیین میشود ** جهت چرخش تنها توسط اولین دستور DRVC مشخص میگردد و سایر تغییرات ترتیبی از این جهت پیروی میکنند ** آخرین دستور در این گزینه به منظور تغببرات ترتیبی باید DRV باشد مثال : 001 SPD 10000 * Pulse frequency = 10KHz. DRVC ADR , + , 20000 , Ut * Forward 20000 units. GOTO NEXT 002 SPD 50000 * Pulse frequency =50 KHz DRVC ADR , + , 60000 , Ut * Forward 60000 units. GOTO NEXT 003 SPD 3000 * Pulse frequency = 3KHz. DRV ADR , + , 5000 , Ut * Forward 5000 units. WAIT X0 * Wait until X0 ON to restart from GOTO 1 the first step to execute.
ADR/ABS	ADR/ABS :نحوه جابجایی را تعیین میکند ADR : جابجایی نسبی(در این گزینه میتوانید جهت چپگرد و یا راستگرد بودن خروجی از حالات + و – استفاده نمایید) ABS : جابجایی معین(در این گزینه چپگرد ویا راستگرد بودن خروجی باید اعداد + و – استفاده نمایید بصورت مثال 1500+ ویا 1500-)

- شرح تفاوت میان تعیین مکان نسبی (ADR) و تعیین مکان معین (ABS) :

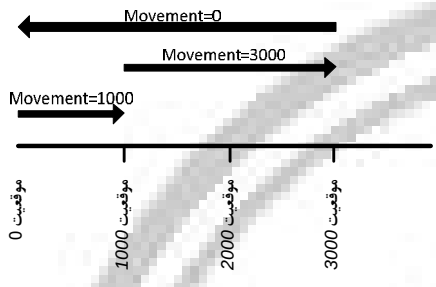
تعریف مد ADR

وقتی بخواهیم تا PLC در هرموقعیتی قرار گیرد باید در رجیستر Movement مقدار حرکت را تعریف کنیم



تعریف مد ABS

وقتی بخواهیم تا PLC در هرموقعیتی قرار گیرد کافیست در رجیستر Movement موقعیت مورد نظر را قرار دهیم



برای تعریف نقطه صفر (رفرنس گرفتن محور) باید عدد صفر را در رجیسترهای رفرنس هر محور انتقال داد

رجیستر رفرس محور ۱ : R4088
رجیستر رفرس محور ۲ : R4090
رجیستر رفرس محور ۳ : R4092
رجیستر رفرس محور ۴ : R4094

+ : حرکت به سمت جلو ویا در جهت عقربه های ساعت **- : حرکت به سمت عقب و یا در خلاف جهت عقربه های ساعت** **‘ ‘ : جهت حرکت توسط مقادیر تعیین میشود(مقادیر مثبت : جلو ،مقادیر منفی : عقب)**	**+ / - / ‘ ‘ :**
این گزینه میتواند از طریق وارد نمودن مقادیر ثابت و یا از طریق رجیسترهای R و D انجام پذیرد(این عمل 2 رجیستر را اشغال مینماید، بطور مثال در صورت انتخاب R0 ، R1 نیز اشغال خواهد شد) **** در صورتیکه مقدار تعیین شده صفر باشد و حالت ADR را انتخاب نموده باشید بدین معناست که حرکت تا بینهایت پالس ادامه میابد** **رنج قابل انتخاب این گزینه:** **99999999 > میزان حرکت >-99999999**	**movement**
Ut/Ps : واحد و یا دقت خروجی را تعیین می نماید **Ut : دقت این گزینه 1 واحد است که توسط پارامترهای FUN140، 0~3 یعنی دقت خروجی بصورت mm,Deg یا Inch خواهد بود** **Ps : دقت خروجی بصورت 1 پالس خواهد بود (پیشنهاد میشود از این گزینه جهت دقت خروجی استفاده گردد)**	**Ut/Ps**

WAIT	هنگامیکه پالس خروجی تعیین شده به اتمام میرسد زمان توقف برای رفتن به خط بعدی برنامه فعال میشود این توقف 5 حالت دارد: 1. زمان توقف که میتواند مقدار ثابتی باشد و یا توسط رجیستر تعیین گردد 2. توسط X0~X255 منتظر می ماند تا ورودی تعیین شده به حالت ON برود 3. توسط Y0~Y255 منتظر می ماند تا خروجی تعیین شده به حالت ON برود 4. توسط M0~M1911 منتظر میماند تا رله کمکی تعیین شده به حالت ON برود 5. توسط S0~S999 منتظر میماند تا رله کمکی تعیین شده به حالت ON برود
ACT	پس از انجام شدن پالسها در خروجی این گزینه فعال شده تا پس اززمان تعیین شده پارامتر GOTO را اجرا نماید
EXT	هنگامیکه خروجی پالس در حال اجرا میباشد، اگر این پارامتر ON شود بلافاصله دستور تعیین شده توسط GOTO اجرا خواهد شد بدون اینکه عمل خروجی در خروجی به پایان رسیده باشد
GOTO	بعد از انجام اعمال EXT,ACT,WAIT این گزینه فعال میگردد جهت مشخص نمودن اجرای دستوربعدی NEXT : دستور خط بعدی اجرا خواهد شد 1~N : شماره خط تعیین شده اجرا خواهد شد R/DXXXX : شماره خط دستور بعدی در این رجیستر میتواند ذخیره شود

- نحوه عملکرد FUN140

Ps : خروجی پالس تعیین شده

0 : Y0,Y1

1 : Y2,Y3

2 : Y4,Y5

3 : Y6,Y7

SR : رجیستر تعیین شده در جدول برنامه ریزی سروو **(Servo Program Table)**

WR : رجیستر مربوط به حالات مختلف کارکرد **FUN 140** که **7** رجیستر را اشغال می نماید (این ریجیسترها برای کارکرد این تابع هستند و در جای دیگر نباید از آنها استفاده کرد)

- **شرح عملکرد**

** هنگام فعال شدن **EN=1** در صورت فعال نبودن **FUN 140** دیگری برروی خروجی مورد نظر **(Ps)** ،دستورات تعیین شده در جدول پارامترهای سروو خط به خط اجرا می شود.

** در صورت غیر فعال شدن **EN=0** بلافاصله پالس خروجی قطع خواهد شد

** در صورت **EN=1** و **PAU=1** خروجی پالس وتوقف خواهد شد و پس از اینکه **PAU=0** خروجی پالس با توجه به جدول پارامترهای سروو ادامه کار خواهد داد

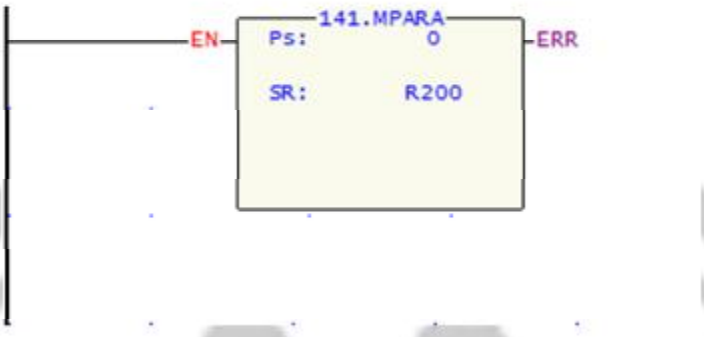
** در صورت **ABT=1** خروجی پالس بلافاصله قطع شده و هنگامیکه **EN=1** گردد ،دستورات جدول پارامترهای سروو از اولین خط شروع به کار می نماید

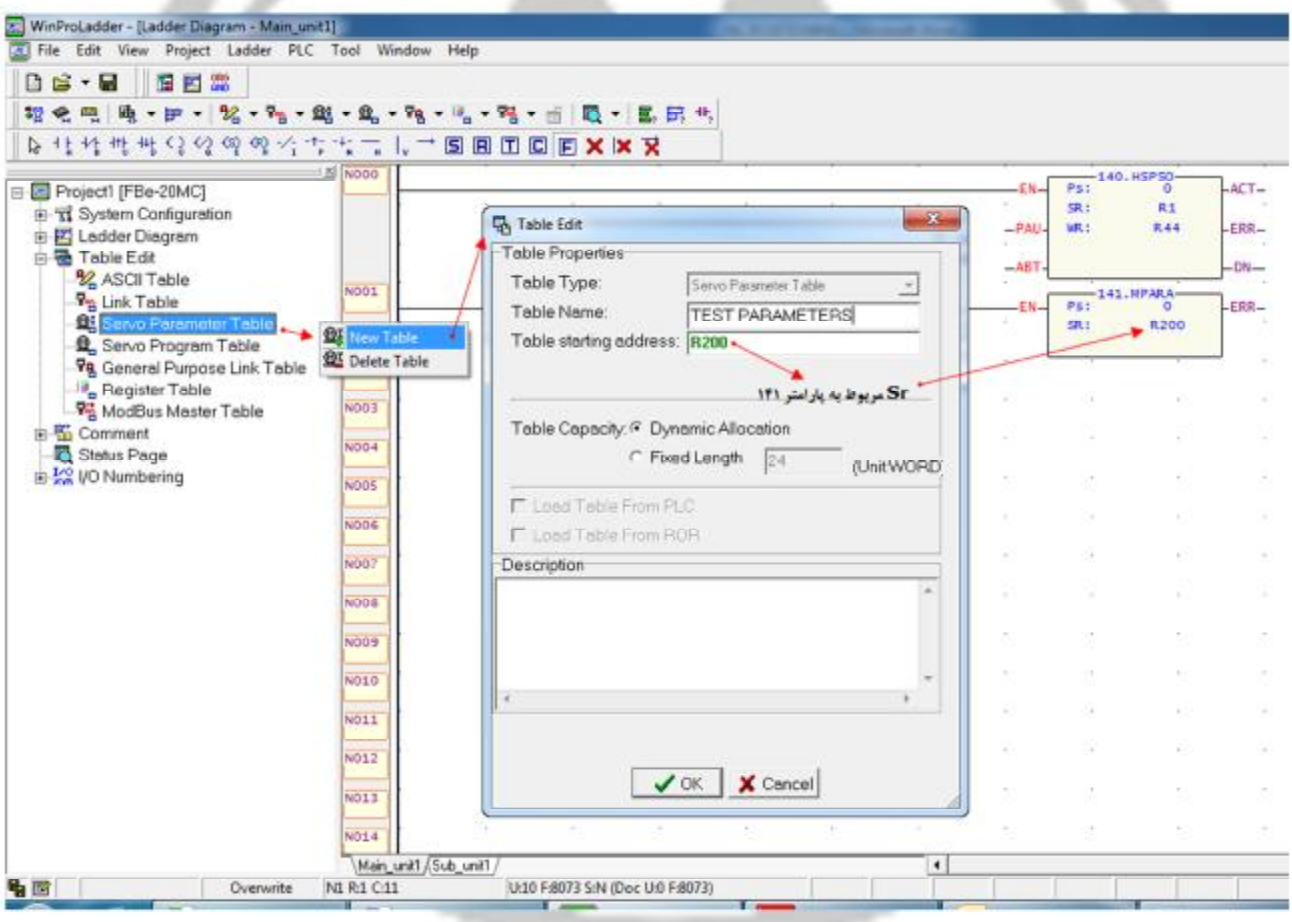
- **حالات کاری خروجی پالس**
  - **تغییرهمزمان مقدار فرکانس خروجی (تغییر فرکانس پالس) در حین اجرای FUN 140**

جهت انجام این کار کافیست عدد 90 را در بایت کم ارزش(Low Byte) R4056 کپی کنید تا بتوانید هنگامیکه خروجی پالس مقادیر را از جدول **SERVO Program** اجرا می نماید ، فرکانس پالس خروجی را بصورت همزمان تغییر دهید.

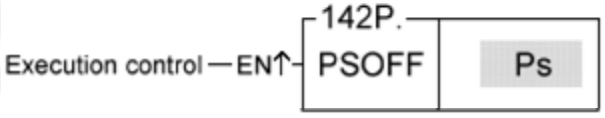
مقدار پیش تنظیم این رجیستر صفر می باشد.

FUN141 فانکشن فعال سازی جدول پارامترهای حرکت سرو :



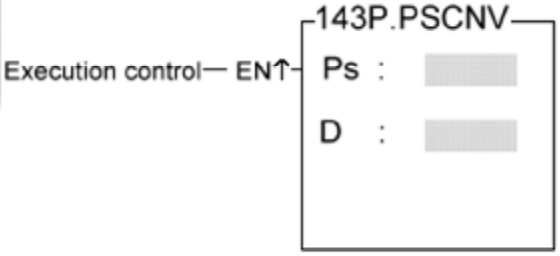


Function 142.STOP PULSE OUTPUT



این تابع فرستادن پالس به خروجی را متوقف می کند.

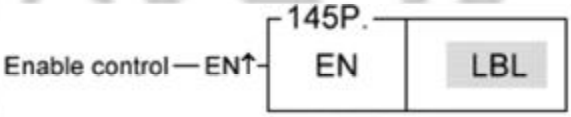
Function 143.PSCNV



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

این تابع تنها هنگام اجرای تابع **140** اجرا می شود.

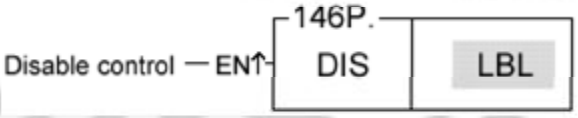
Function 145.ENABLE OF INTERRUPT



هرگاه **"EN"** از **0** به **1** تغییر کند: این تابع برنامه فرعی یا اینتراپتی را فعال می کند که برچسب (**LABLE**) آن در دستور نام برده شده است. **LABLE** اینتراپت ها در جدول زیر آورده شده است.

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

Function 146.DISABLE OF INTERRUPT



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

Function 147.MULTI-AXIS HIGH SPEED PULSE OUTPUT

این تابع برای پشتیبانی از interpolation خطی در کنترل حرکت همزمان چند محور ،استفاده می شود . با برنامه متنی که در جدول Multi-axis Positioning table نوشته می شود مرتبط است . این تابع می تواند تا 4 محور را برای interpolation خطی همزمان، پشتیبانی کند.

Function 150 . MODBUS COMMUNICATION

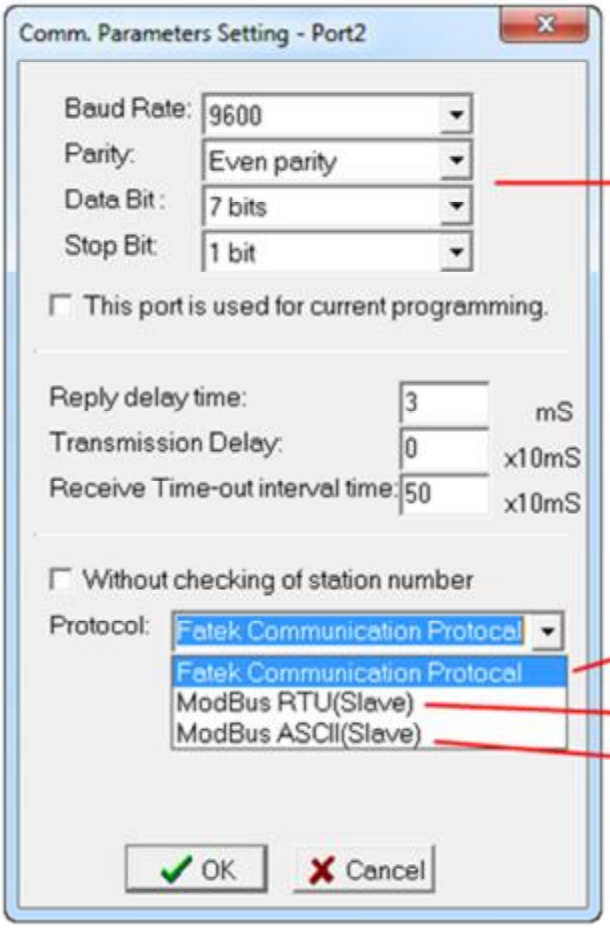
از این فانکشن ها جهت اتصال دو یا چند PLC و یا اتصال PLC به وسایل جانبی از طریق استاندارد MODBUS RTU استفاده میشود.

- **شرح مختصری پیرامون ارتباط PLC با سایر تجهیزات**

1- **FUN150 می تواند از طریق RS485 , RS232 ارتباط برقرار کند.**

2- **Station number و پارامترهای ثابت (Baude rate , Parity , Data bit , Stop bit) در تمام تجهیزاتی که می خواهیم با هم ارتباط دهیم باید با یکدیگر برابر باشند.**

پارامترهای مربوط به FUN 150 ودر صورت لزوم تغییر Time-out و Transaction Delay (جهت وسایلی که سرعت پاسخ آنها پایین است) باید تنظیم شوند.



تنظیمات مربوط به تجهیزات باید باهم برابر باشند

ایجاد ارتباط بین چند **FATEK PLC**
می توان بطورمستقیم از ریجیسترها استفاده کرد

تجهیزاتی که **Modbus RTU** دارند.

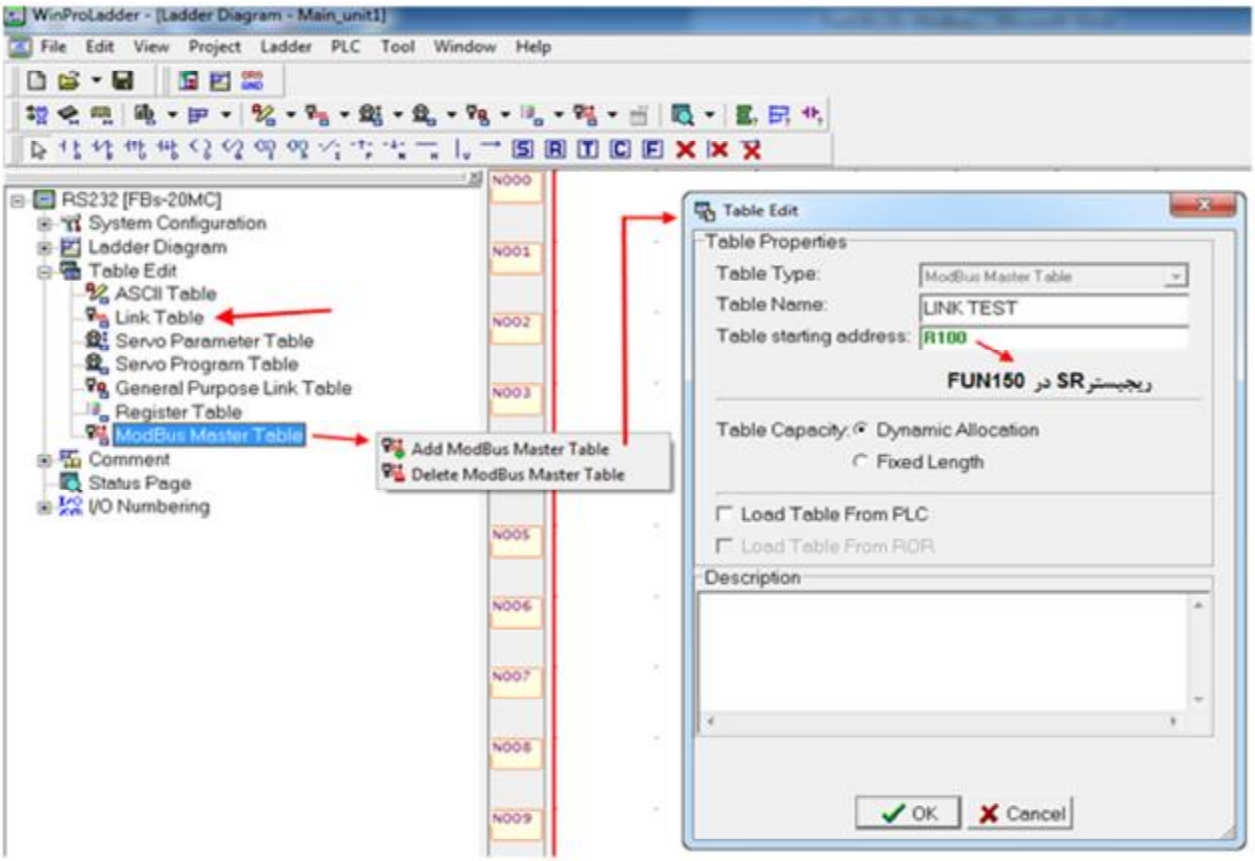
تجهیزاتی که **Modbus ASCII** دارند.

باید از آدرس مودباس ریجیسترها استفاده کرد

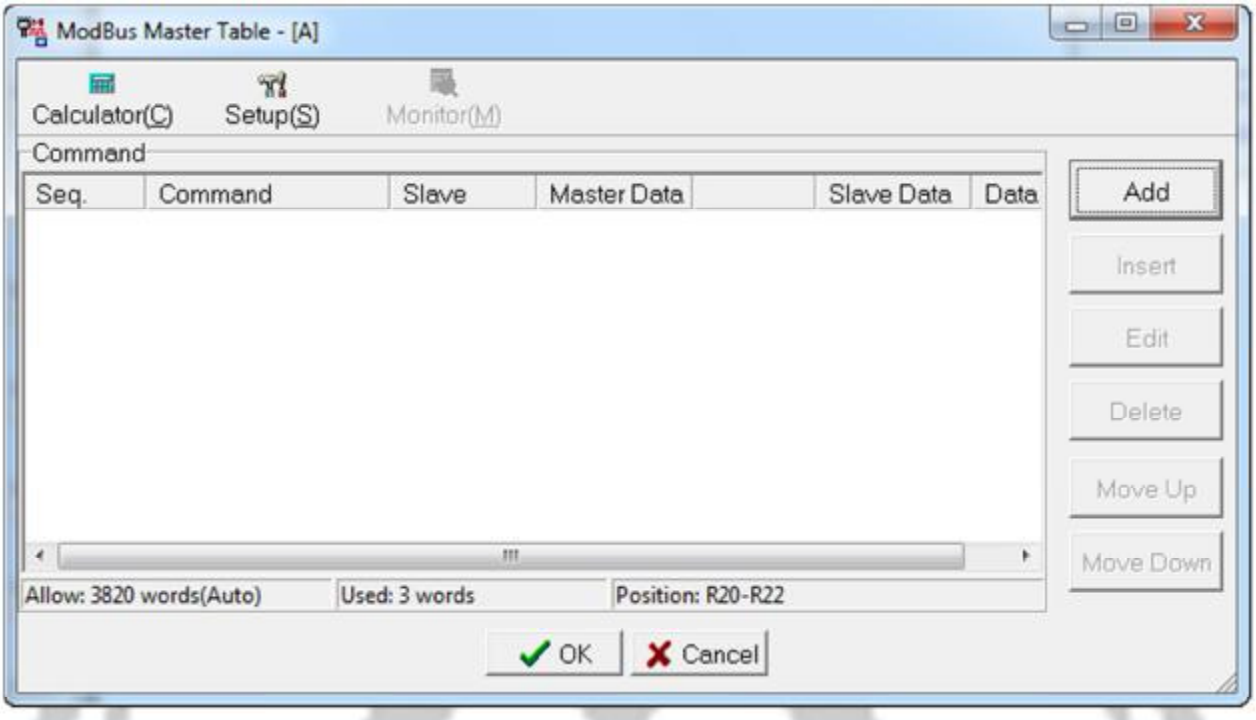
این فانکشن ها باید در برنامه پی ال سی MASTER نوشته شود .

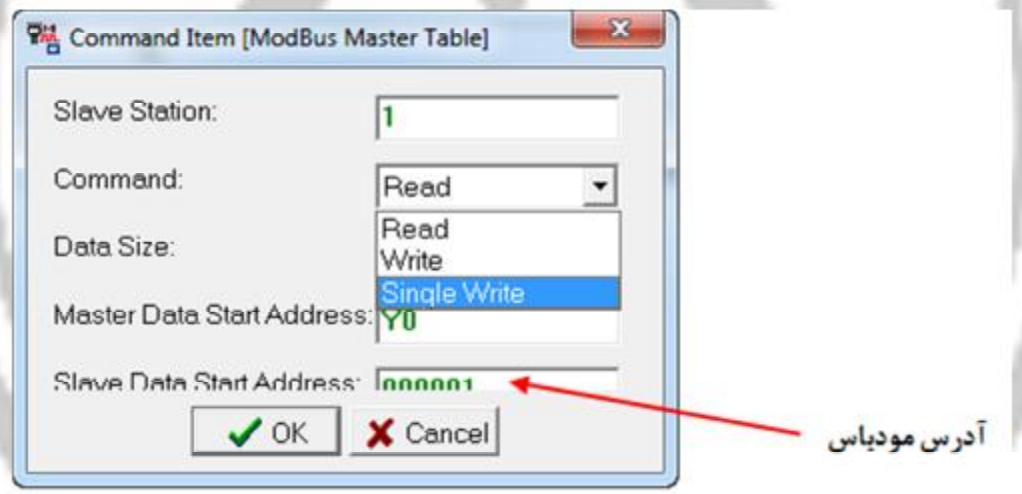
جدول مربوط به ارسال و دریافت اطلاعات برروی پی ال سی MASTER باید تنظیم شود.

جدول MODBUS MASTER TABLE برای فانکشن 150 و استاندارد MODBUS RTU می باشد.



MODBUS MASTER TABLE جدول





Slave Station	شماره Number Station مربوط به تجهیز Slave
Command	فرمان خواندن یا نوشتن از پی ال سی Master به تجهیز Slave می باشد.
Data Lengh	تعداد رجیسترهایی که عمل ارسال و دریافت اطلاعات را انجام میدهند
Master Data Start Address	رجیسترهایی برای ارسال و یا دریافت اطلاعات توسط پی ال سی Master
Slave Data Start Address	رجیسترهایی برای ارسال و یا دریافت اطلاعات توسط پی ال سی Slave

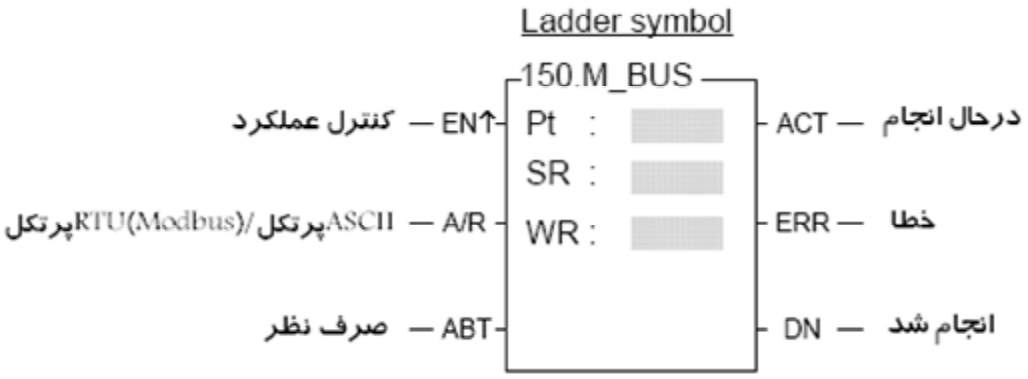
- نوع اطلاعاتی که مابین تجهیزات می تواند ردوبدل شود در جدول زیر مشاهده می نمایید

Data code	Data type	Reference number
0	X (discrete input)	0～255
1	Y (discrete output)	0～255
2	M (internal relay M)	0～1911
3	S (step relay S)	0～999
4	T (timer contact)	0～255
5	C (counter contact)	0～255
6	WX (word of discrete input ,16 bits)	0～240, it must be the multiple of 8.
7	WY (word of discrete output ,16 bits)	0～240, it must be the multiple of 8.
8	WM (word of internal relay,16 bits)	0～1896, it must be the multiple of 8.
9	W S (word of step relay,16 bits)	0～984, it must be the multiple of 8.
10	TR (timer register)	0～255
11	CR (counter register)	0～199
12	R (data register Rxxxx)	0～3839
13	D (data register Dxxxx)	0～4095

- رجیسترهای مربوط به پورتها ارتباطی :

Comm Port / Signals	Port 1	Port 2	Port 3	Port 4
1. Port Ready Indicator	M1960	M1962	M1936	M1938
2. Port Finished Indicator	M1961	M1963	M1937	M1939
3. Port Communication Parameters	R4146	R4158	R4043	R4044
4. TX Delay & RX Time-out Span	R4147	R4159	R4045	R4048

- شرح تنظیمات FUN 150



Pt : شماره پورت ارتباطی PLC با سایر تجهیزات را مشخص می نماید ،1~4

SR : رجیستری را که در جدول تنظیم پارامترها (Modbus Master Table) تعیین نمودید در اینجا باید نوشته شود

WR : رجیستر مربوط به عملکرد FUN 151 که 8 رجیستر را اشغال خواهد نمود

این فانکشن ارتباط مابین 247 ایستگاه جانبی را میتواند برقرار نماید

تنها Master-PLC احتیاج به نوشتن FUN 150 را دارد

اگر A/R=0 باشد FUN 150 بوسیله استاندارد Modbus RTU ارتباط برقرار مینماید و اگر A/R=1 باشد استاندارد ارتباطی Modbus ASCII خواهد بود.

استاندارد ارتباطی Modbus ASCII از OS Version 4.12 به بعد قابل اجرا می باشد

"EN"این فانکشن ها فقط با لبه فعال می شود یعنی با هر لبه یکبار اطلاعات را برروی پورت می فرستد بنابراین ورودی EN این فانکشن را باید با کنتاکت M1920 یا M1921 یا M1922 سری کرد تا دائما اطلاعات برروی پورت فرستاده شود.

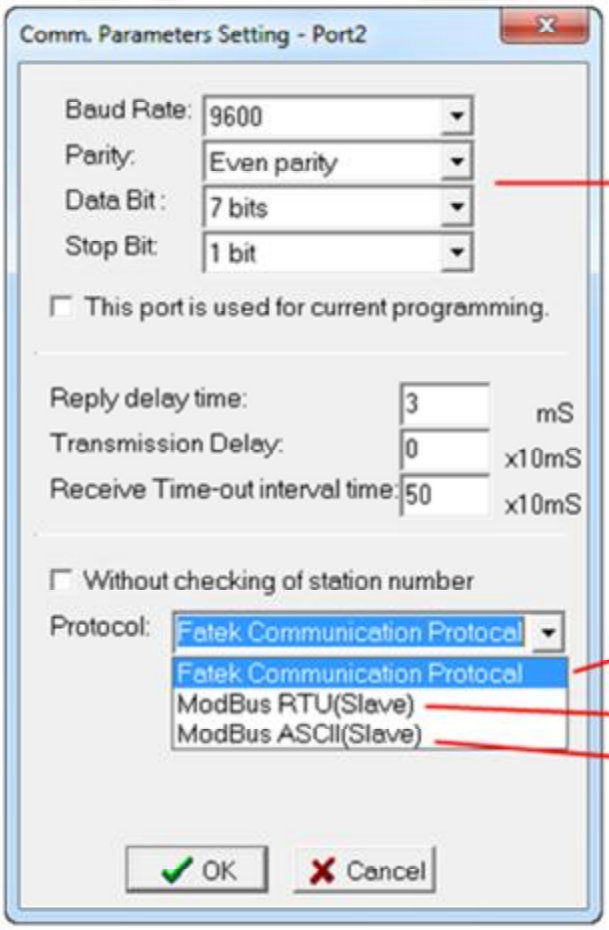
Function 151 . CPU LINK

از این فانکشن جهت اتصال دو یا چند PLC و یا اتصال PLC به وسایل جانبی که استاندارد (FACON) FATEK را ساپورت می کنند استفاده می شود. این فانکشن می تواند اطلاعات ریجیسترشماره n پی ال سی master را در اختیار ریجیستر شماره n پ ال سی های slave بگذارد ، در واقع این فانکشن فقط می تواند برروی پی ال سی های slave بنویسد .

3- FUN151 در بستردرگاههای ETHERNET , RS485 , RS232 می تواند ارتباط برقرار کند

4- پارامترهای ثابت (Baude rate , Parity , Data bit , Stop bit) در تمام تجهیزاتی که می خواهیم با هم ارتباط دهیم باید با یکدیگر برابر باشند.

5- پارامترهای مربوط به FUN 151 ودر صورت لزوم تغییر Time-out و Transaction Delay (جهت وسایلی که سرعت پاسخ آنها پایین است) باید تنظیم شوند.



تنظیمات مربوط به تجهیزات باید باهم برابر باشند

ایجاد ارتباط بین چند FATEK PLC
می توان بطورمستقیم از ریجیسترها استفاده کرد

تجهیزاتی که Modbus RTU دارند.

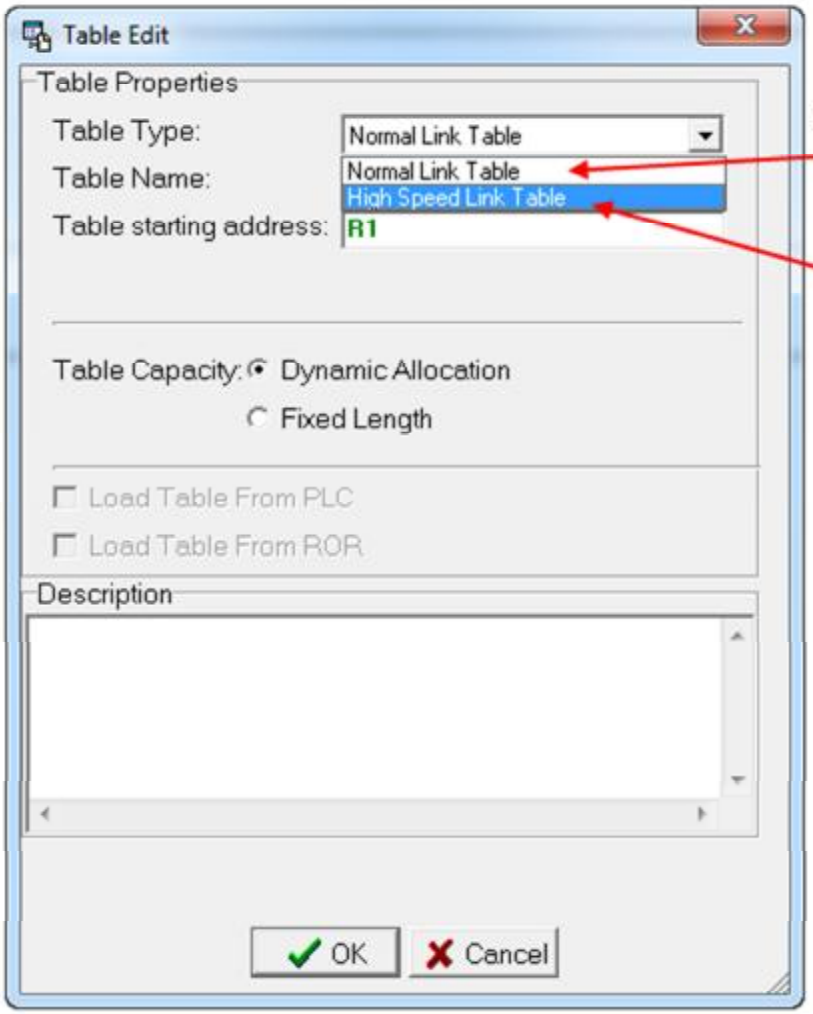
تجهیزاتی که Modbus ASCII دارند.

باید از آدرس مودباس ریجیسترها استفاده کرد

6- این فانکشن باید در برنامه پی ال سی MASTER نوشته شود .

7 جدول مربوط به ارسال و دریافت اطلاعات برروی پی ال سی MASTER باید تنظیم شود.

جدول LINK TABLE برای فانکشن 151 و استاندارد FATEK می باشد.

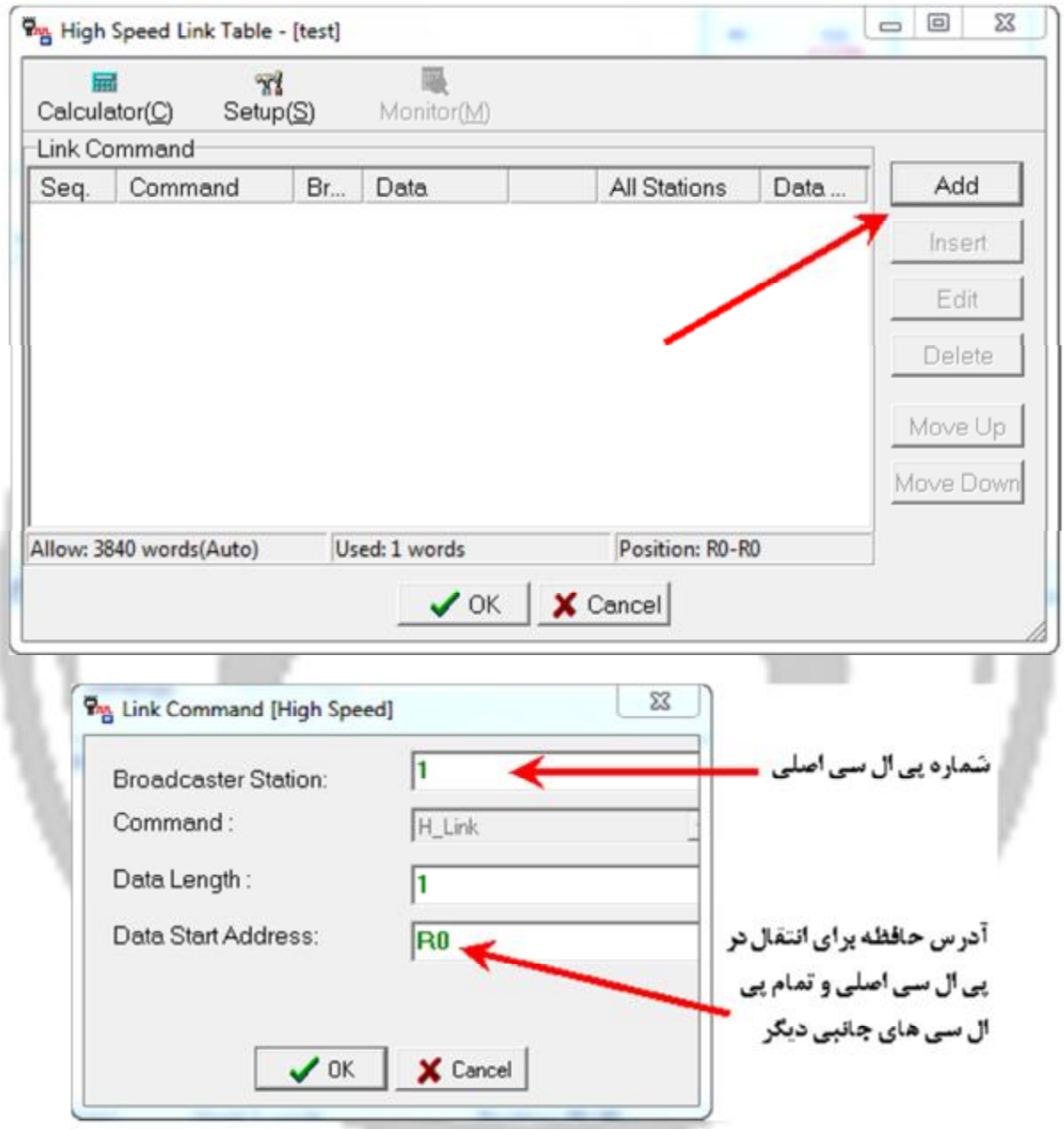


برای خواندن و نوشتن برروی SLAVE

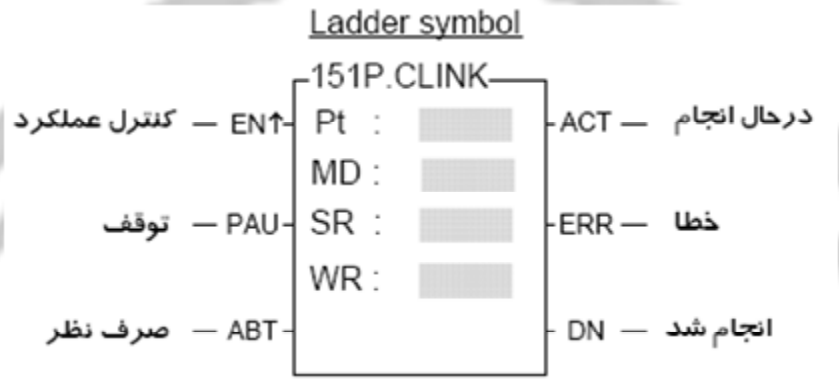
در این حالت می توان در یک لحظه برروی تمام پی ال سی های Slave دیگر نوشت

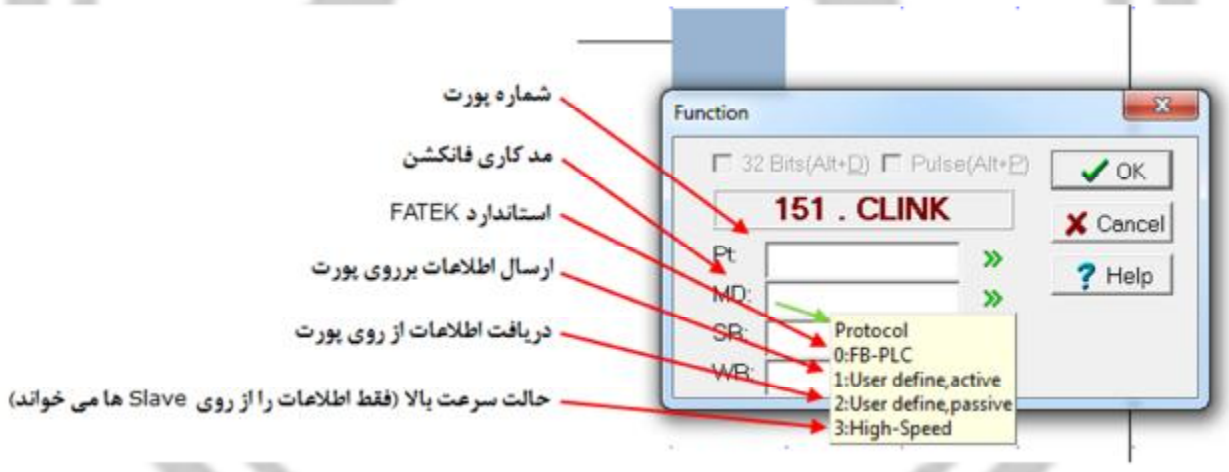
این حالت برای مواردی که می خواهیم اطلاعات پی ال سی اصلی را با سرعت در اشتراک تمام پی ال سی های جانبی بگذاریم استفاده می کنیم

جدول LINK TABLE


High Speed Link Table - [test]
Calculator(C)
Setup(S)
Monitor(M)
Link Command
Seq.
Command
Br...
Data
All Stations
Data ...
Add
Insert
Edit
Delete
Move Up
Move Down
Allow: 3840 words(Auto)
Used: 1 words
Position: R0-R0
OK
Cancel
Link Command [High Speed]
Broadcaster Station:
1
Command :
H_Link
Data Length :
1
Data Start Address:
R0
OK
Cancel
شماره پی ال سی اصلی
آدرس حافظه برای انتقال در
پی ال سی اصلی و تمام پی
ال سی های جانبی دیگر

- شرح عملکرد FUN 151





SR : رجیستری را که در جدول ارتباطات (LINK Table) تعیین نمودید در اینجا باید نوشته شود .

WR : رجیستر مربوط به عملکرد FUN 151 که 8 رجیستر را اشغال خواهد نمود .

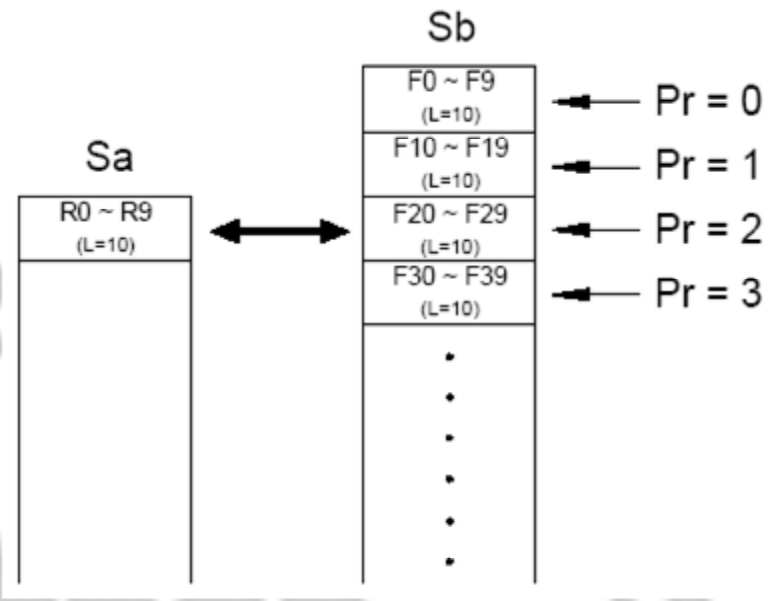
- این ارتباط میتواند تا 254 ایستگاه را پوشش دهد .
- "EN" این فانکشن ها فقط با لبه فعال می شود یعنی با هر لبه یکبار اطلاعات را برروی پورت می فرستد بنابراین ورودی EN این فانکشن را باید با کنتاکت M1920 یا M1921 یا M1922 سری کرد تا دائما اطلاعات برروی پورت فرستاده شود.

- رجیسترهای مربوط به پورتها ارتباطی :

Comm Port / Signals	Port 1	Port 2	Port 3	Port 4
1. Port Ready Indicator	M1960	M1962	M1936	M1938
2. Port Finished Indicator	M1961	M1963	M1937	M1939
3. Port Communication Parameters	R4146	R4158	R4043	R4044
4. TX Delay & RX Time-out Span	R4147	R4159	R4045	R4048

Function 160.READ/WRITE FILE REGISTER

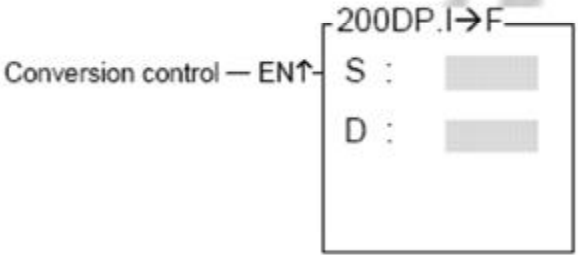
نوعی از رجیستر ها به نام File Registers در حافظه PLC وجود دارند که این تابع،تنها تابعی است که می تواند به پردازش آنها بپردازد. هرگاه "EN" از 0 به 1 تغییر کند: اگر ورودی "R/W"=1 ،محتویات رجیسترهای فایل با آدرس شروع Sb به طول L ،از جایی که Pr اشاره می کند ،به داخل رجیسترهای شروع شونده از Sa ریخته می شود. اگر ورودی "R/W"=0 ، محتویات رجیسترهای شروع شونده از Sa به داخل رجیسترهای فایل با آدرس شروع Sb به طول L ،از جایی که Pr اشاره می کند ،ریخته می شود. با فعال شدن ورودی "INC" ، Pr یکی اضافه خواهد شد. اگر L=0 یا L>511 باشد یا عملیات بخواهد خارج از رنج رجیسترهای فایل (F0~F8191) اجرا شود،خروجی "ERR" فعال خواهد شد.



توابع اعداد اعشاری (Float)

برای کار با اعداد اعشاری باید از فرمت Float استفاده کرد که فضای دو رجیستر را اشغال می کنند (32bits)

Function 200.CONVERSION of INT to FLOAT



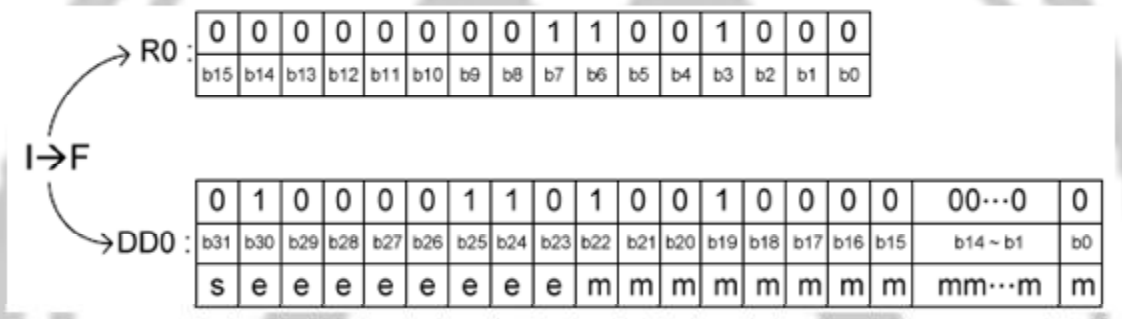
این تابع برای تبدیل فرمت اعداد صحیح به اعشاری استفاده می شود . هرگاه "EN" از 0 به 1 تغییر کند: محتویات رجیستر 16 بیتی (INTEGER) شروع شونده از S درداخل رجیستر 32 بیتی (FLOAT) شروع شونده از D ذخیره می شود.

مثال:

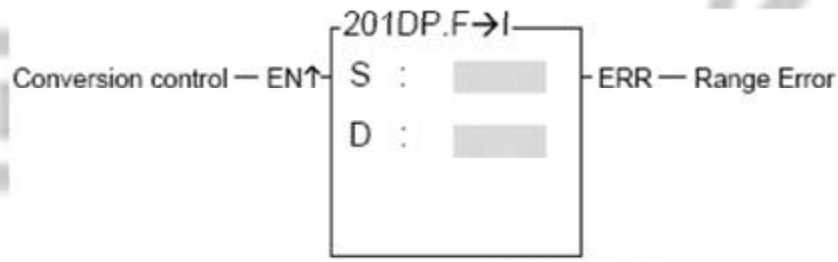
※ R0 = 200 (0000000011001000)

Integer To Floating

DD0 = 43480000H



Function 201. FLOAT to INTEGER



این تابع بر عکس تابع قبل ، فرمت اعداد اعشاری را به عدد صحیح

تبدیل کرده و اعشار آن را از بین می برد . هرگاه "EN" از0 به 1 تغییر کند: محتویات رجیستر 32 بیتی (FLOAT) شروع شونده از S درداخل رجیستر 16 بیتی (INTEGER) شروع شونده از D ذخیره می شود. اگر مقدار مبدا فراتر از محدوده مقصد بوده و قابل تبدیل نباشد ، خروجی"ERR" فعال شده و مقدار D دست نخورده باقی می ماند.

مثال:

※ DR20 = 123.45 →Normalize→ 42F6E666H

Floating To Integer ←

→ D10 = 007BH

DR20:	0	1	0	0	0	0	1	0	1	1	1	1	0	1	1	0	1	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0
	b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
	s	e	e	e	e	e	e	e	e	m	m	m	m	m	m	m	m	m	m	m	m	m	m	m	m	m	m	m	m	m	m	m

F → I

D10:	0	0	0	0	0	0	0	0	0	1	1	1	1	0	1	1
	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0

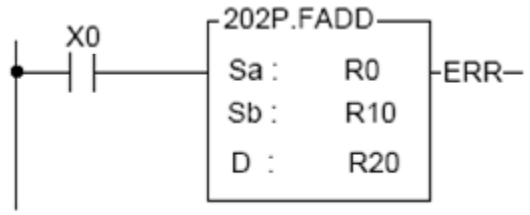
Function 202. FLOATING POINT NUMBER ADDITION

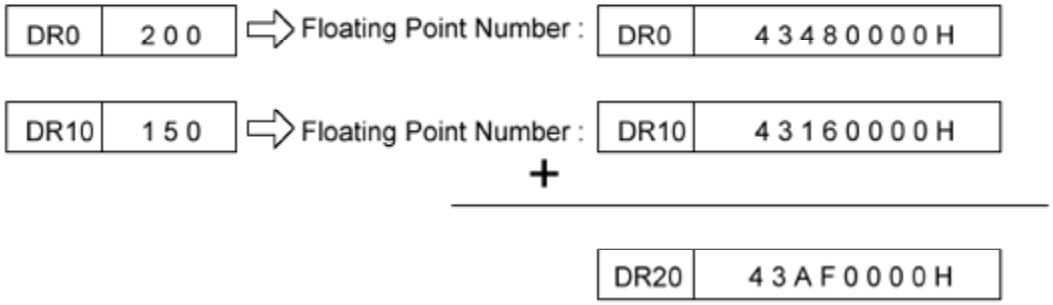
برای جمع اعداد اعشاری استفاده می شود .

هرگاه "EN" از 0 به 1 تغییر کند:

مقدار FLOAT (32 بیتی) Sa با مقدار FLOAT ، Sb جمع شده و نتیجه در D ریخته می شود. اگر مجموع دو عدد فراتر از محدوده یک رجیستر 32 بیتی باشد ($\pm 3.4\times10^{38}$)، خروجی "ERR" فعال می شود.

مثال:

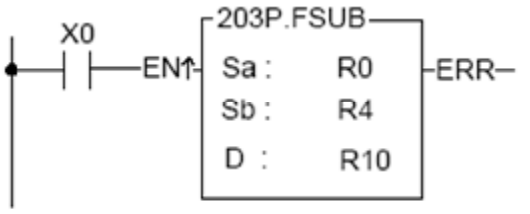


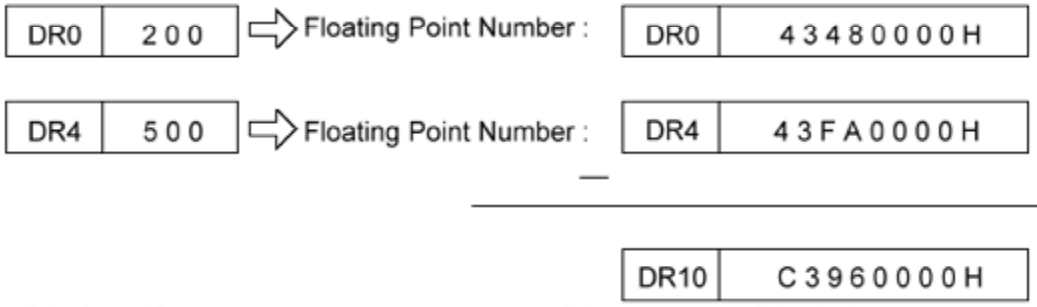


Function 203. FLOATING POINT NUMBER SUBTRACTION

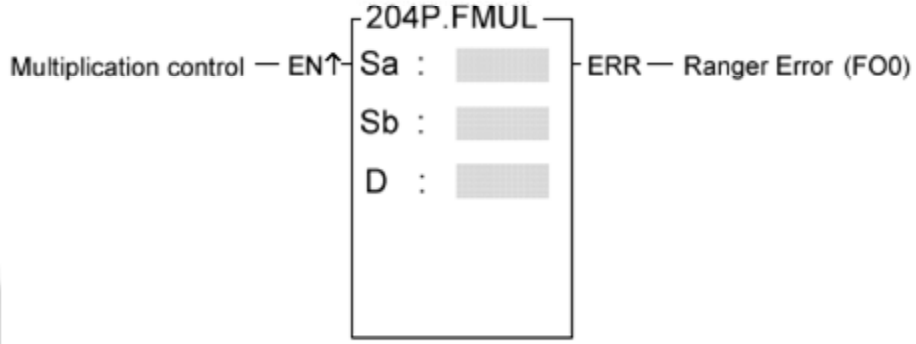
برای تفریق اعداد اعشاری استفاده می شود . هرگاه "EN" از 0 به 1 تغییر کند: مقدار FLOAT (32 بیتی) Sa منهای مقدار FLOAT ، Sb شده و نتیجه در D ریخته می شود. اگر نتیجه تفریق دو عدد فراتر از محدوده یک رجیستر 32 بیتی باشد ($\pm 3.4\times10^{38}$)، خروجی "ERR" فعال می شود.

مثال:



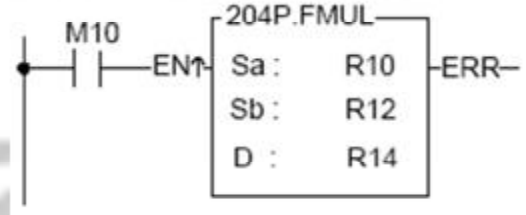


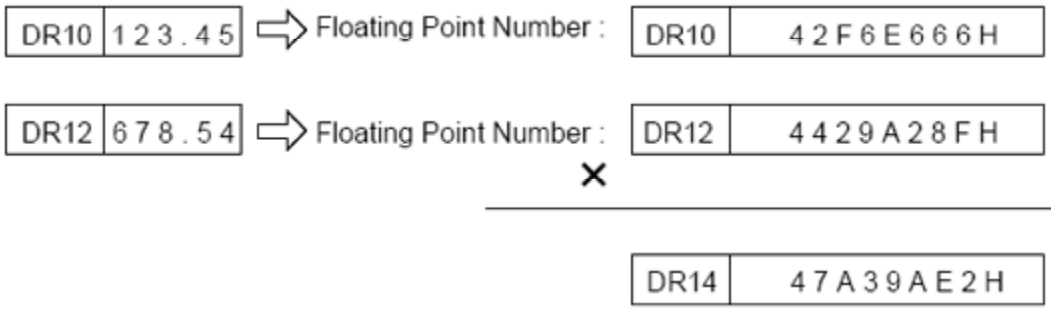
Function 204. FLOATING POINT NUMBER MULTIPLICATION



برای ضرب اعداد اعشاری استفاده می شود . هرگاه "EN" از 0 به 1 تغییر کند: مقدار FLOAT (32 بیتی) Sa ضرب در مقدار FLOAT ، Sb شده و نتیجه در D ریخته می شود. اگر نتیجه ضرب دو عدد فراتر از محدوده یک رجیستر 32 بیتی باشد ($\pm 3.4\times10^{38}$)، خروجی "ERR" فعال می شود.

مثال:

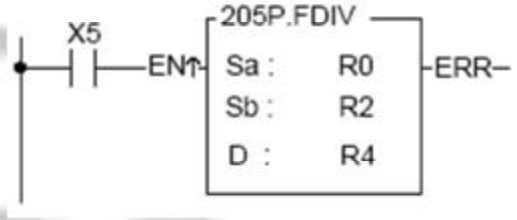


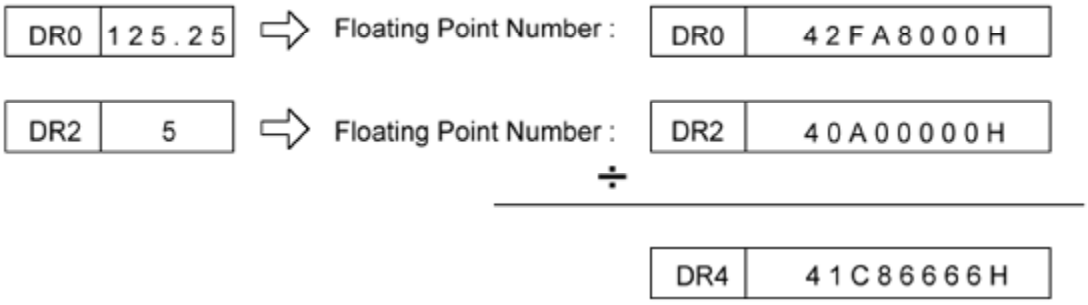


Function 205. FLOATING POINT NUMBER DIVISION

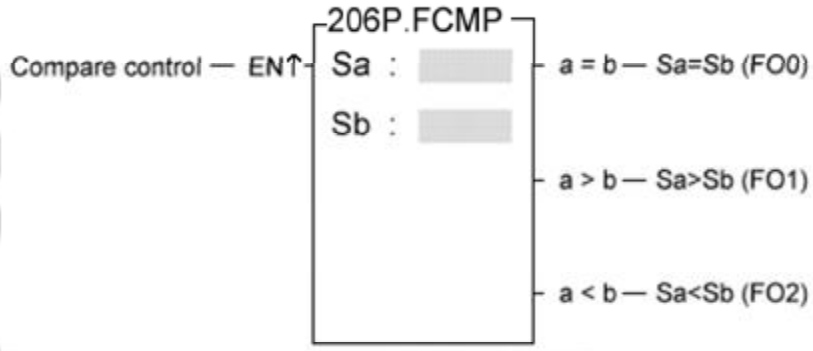
برای تقسیم اعداد اعشاری استفاده می شود . هرگاه "EN" از 0 به 1 تغییر کند: مقدار FLOAT (32 بیتی) Sa را بر مقدار FLOAT ، Sb تقسیم کرده و نتیجه در D ریخته می شود. اگر نتیجه تقسیم دو عدد فراتر از محدوده یک رجیستر 32 بیتی باشد ($\pm 3.4\times10^{38}$)، خروجی "ERR" فعال می شود.

مثال:



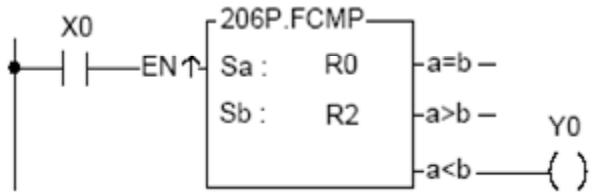


Function 206. FLOATING POINT NUMBER COMPARE



برای مقایسه میان اعداد اعشاری استفاده می شود . هرگاه "EN" از 0 به 1 تغییر کند: مقدار دو رجیستر 32 بیتی Sa و Sb را با هم مقایسه کرده، اگر Sa > Sb : خروجی 1="a > b" می شود. اگر Sa < Sb : خروجی 1="a < b" می شود. اگر Sa = Sb : خروجی 1="a = b" می شود.

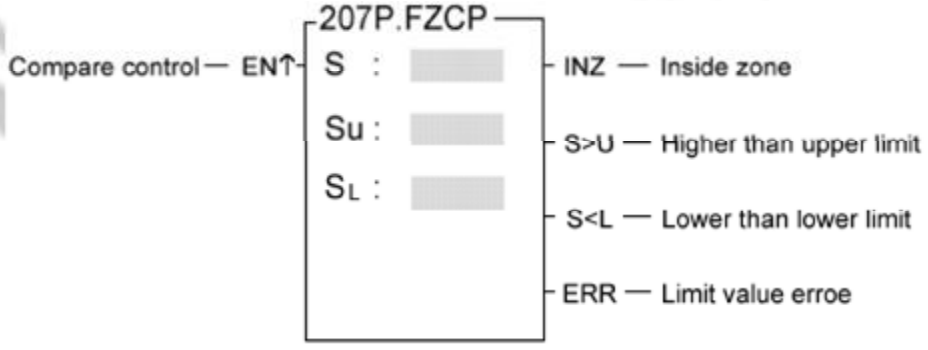
مثال:



DR0	200.1	⇨ Floating Point Number :	DR0	4348199AH

DR2	200.2	⇨ Floating Point Number :	DR2	43483333H

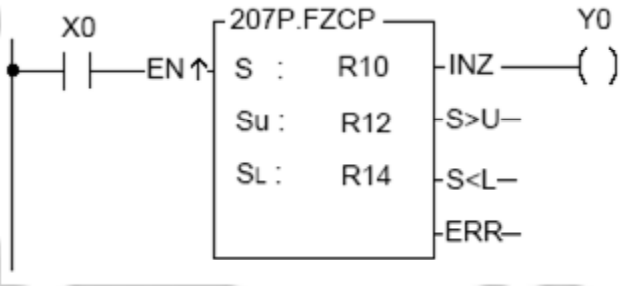
Function 207. FLOATING POINT NUMBER ZONE COMPARE



برای مقایسه ناحیه ای میان اعداد اعشاری استفاده می شود .

هرگاه "EN" از 0 به 1 تغییر کند: مقدار 32 بیتی S را با Su و S_L مقایسه کرده، اگر S > Su : خروجی 1 "S > U" می شود. اگر $S < S_L$: خروجی 1="S < L" می شود. اگر $S_L < S < Su$: خروجی 1="INZ" می شود. اگر $S_L > Su$: خروجی "ERR" داده خواهد شد.

مثال:



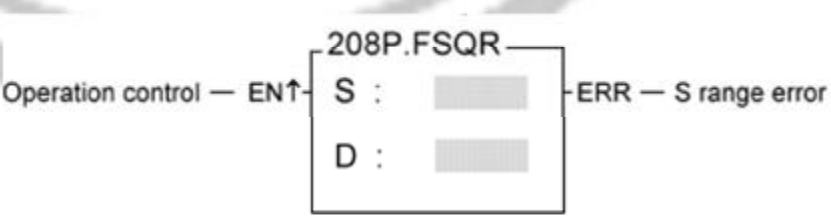
S	DR10	2000.2	⇨ Floating Point Number :	DR10	44FA0666H	
Su	DR12	3000.3	⇨ Floating Point Number :	DR12	453B84CDH	(Upper limit value)
S_L	DR14	1000.1	⇨ Floating Point Number :	DR14	447A0666H	(Lower limit value)

Before-execution

X0 ⎍ → FLOATING ZONE COMPARE → Y0 = 1

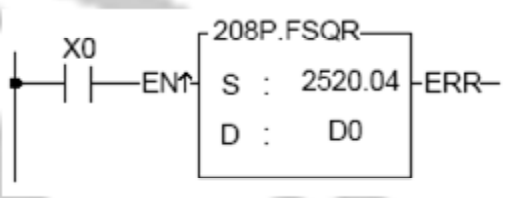
Results of execution

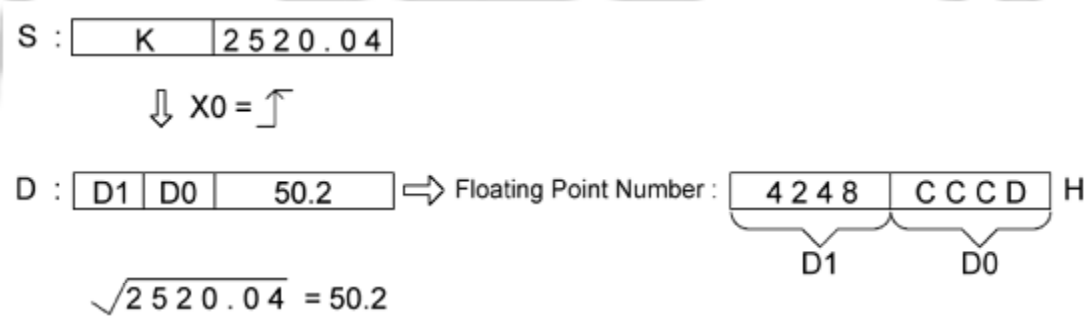
Function 208. FLOATING POINT NUMBER SQUARE ROOT



برای جذر اعداد اعشاری استفاده می شود . هرگاه "EN" از0 به 1 تغییر کند: جذر عدد 32 بیتی S را گرفته و نتیجه را در D می ریزد اگر مقدار S ، منفی باشد، خروجی "ERR" فعال می شود.

مثال:





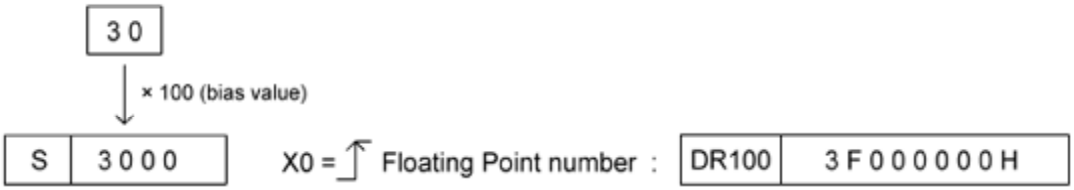
Function 209.SIN INSTRUCTION

تابع مثلثاتی سینوس

هرگاه "EN" از0 به 1 تغییر کند:

از مقدار موجود در رجیستر S ، سینوس گرفته و نتیجه را به صورت 32 بیتی در D و D+1 می ریزد. محدوده S در واحد 0.01 درجه ، 18000+~18000- می باشد که اگر فراتر از این محدوده وارد شود ،خروجی "ERR" فعال می شود.

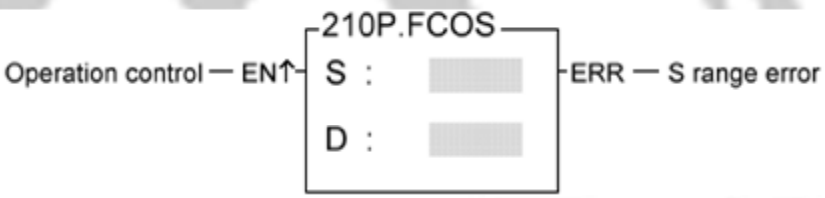
مثال:



SIN(30) = 0.5

Function 210.COS INSTRUCTION

تابع مثلثاتی کسینوس



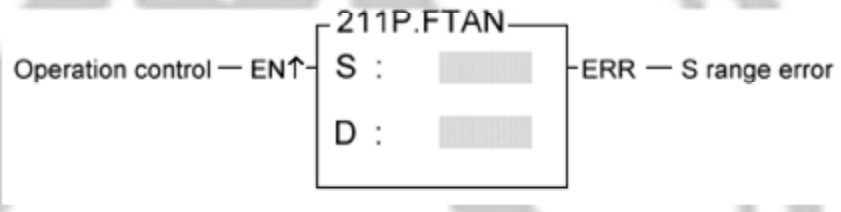
هرگاه "EN" از 0 به 1 تغییر کند: از مقدار موجود در رجیستر S ، کسینوس گرفته و نتیجه را به صورت 32 بیتی در D و D+1 می ریزد. محدوده S در واحد 0.01 درجه ، 18000+~18000- می باشد که اگر فراتر از این محدوده داده شود ،خروجی "ERR" فعال می شود.

مثال:

COS(60) = 0.5

Function 211.TAN INSTRUCTION

تابع مثلثاتی تانژانت

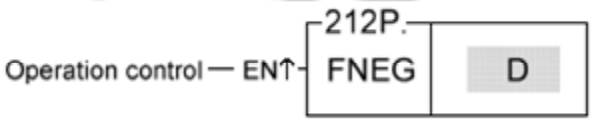


هرگاه "EN" از 0 به 1 تغییر کند: از مقدار موجود در رجیستر S ، تانژانت گرفته و نتیجه را به صورت 32 بیتی در D و D+1 می ریزد. محدوده S در واحد 0.01 درجه ، 18000+~18000- می باشد که اگر فراتر از این محدوده داده شود ، خروجی "ERR" فعال می شود.

مثال:

TAN(45) = 1

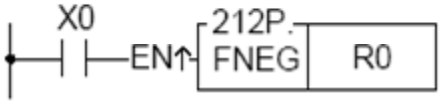
Function 212.CHANGE SIGN OF FLOAT



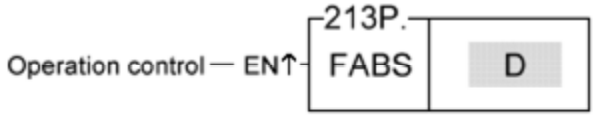
برای تغییر علامت اعداد اعشاری استفاده می شود .

هرگاه "EN" از 0 به 1 تغییر کند: علامت عدد D با فرمت Float را، عکس کرده و نتیجه را در خود D ذخیره می کند.

مثال:

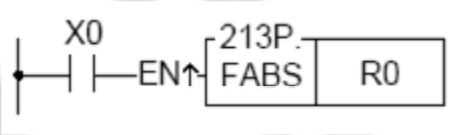


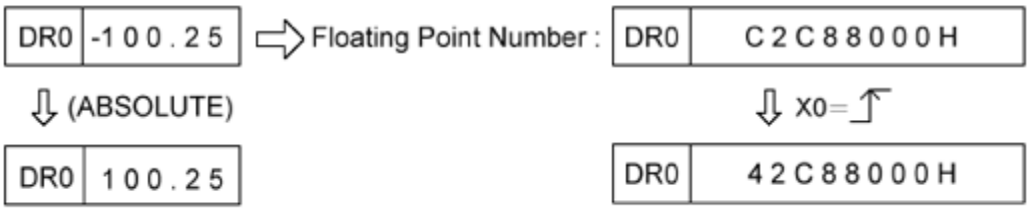
Function 213.FLOAT ABSOLUTE



برای قدر مطلق گرفتن از اعداد اعشاری استفاده می شود . هرگاه "EN" از 0 به 1 تغییر کند: قدر مطلق عدد D با فرمت Float را گرفته و نتیجه را در خود D ذخیره می کند.

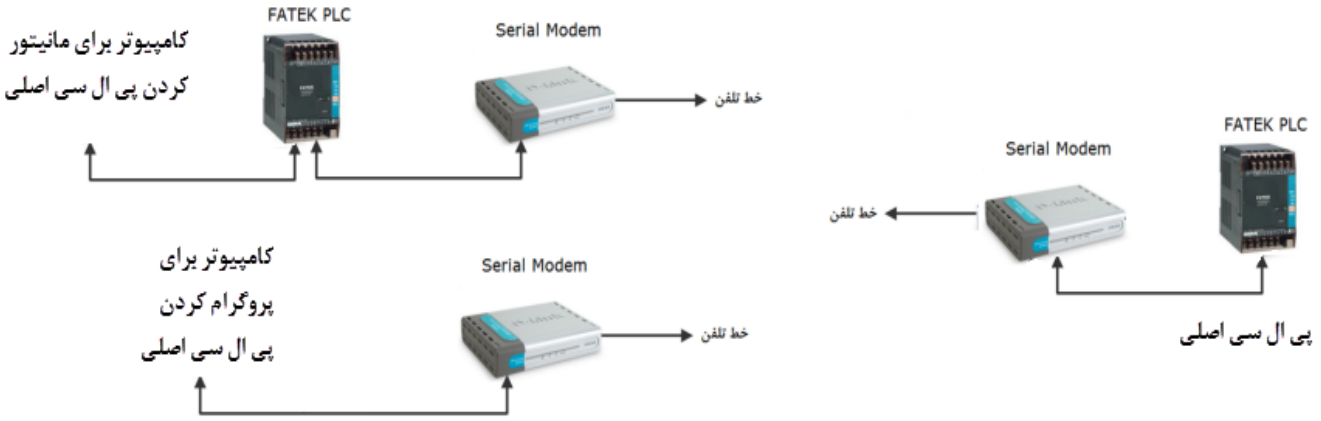
مثال:





بنام خداوند بخشنده و مهربان

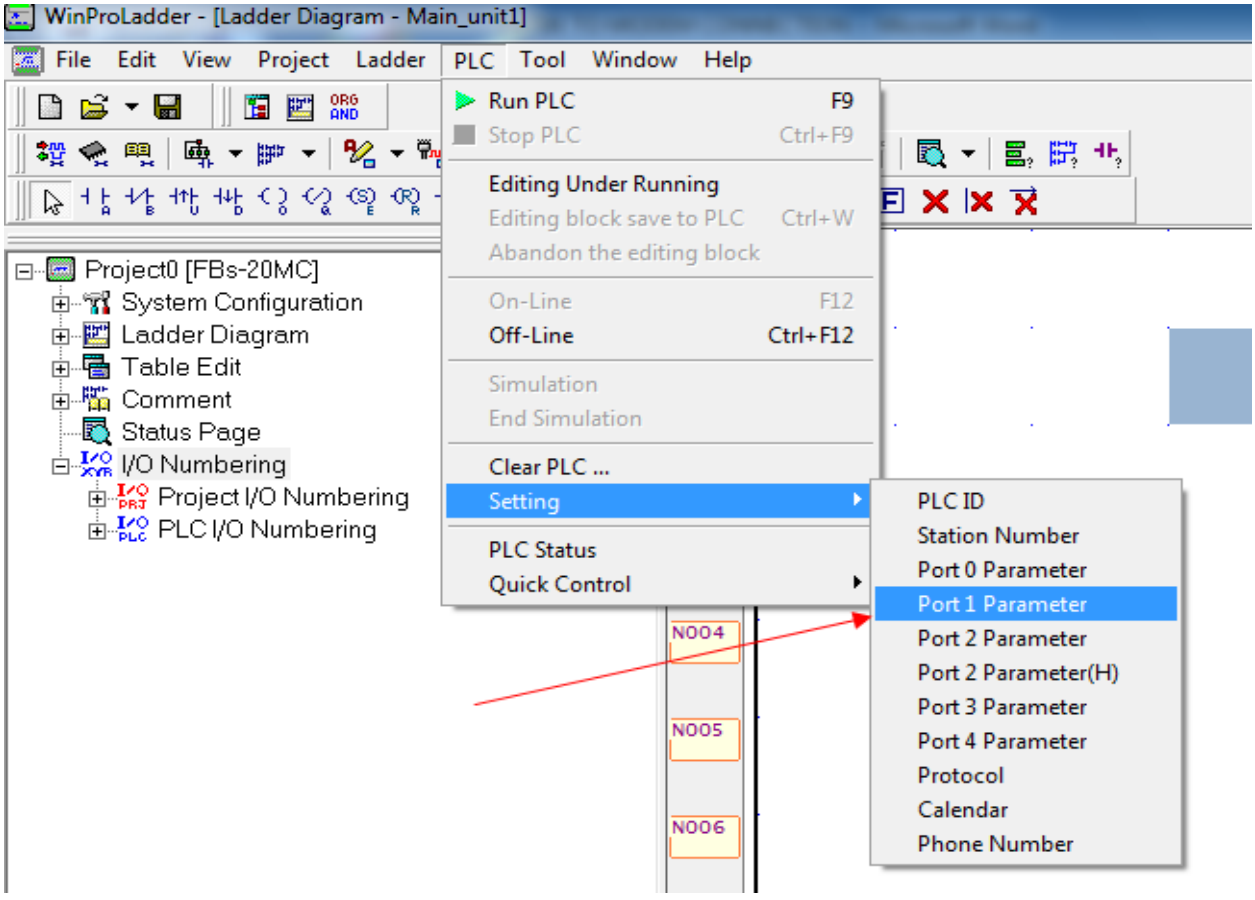
عنوان مدرک :	برنامه نویسی PLC FATEK
توضیحات :	ایجاد ارتباط با FATEK PLC بوسیله خط تلفن
تعداد صفحه :	9
شماره ویرایش :	1
ویرایش کننده :	ا.رضایی
تاریخ ویرایش :	1391.7.27



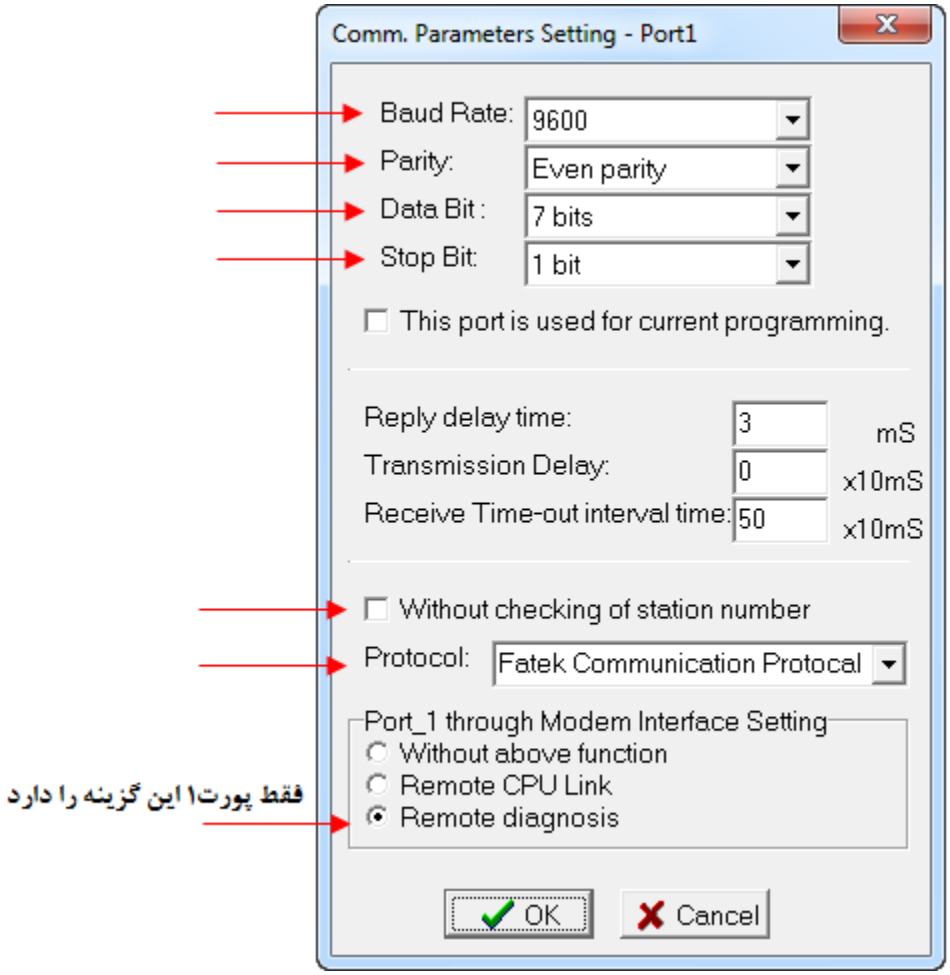
الف) تنظیمات در PLC

1- مودم را به پورت 1 پی ال سی متصل کنید (فقط پورت 1 گزینه Remote diagnosis را دارد).

2- از طریق نرم افزار Winproladder ، به PLC متصل شوید (از طریق RS-232) و در حالی که online هستید به منوی PLC رفته و از فهرست آویزه ای Setting ، پورت 1 را تنظیم کنید و ارتباط خود را با PLC قطع نمائید .



تنظیمات پروتکل پورت 1 :



ب) نحوه ی اتصال به FATEK از طریق مودم :

اکنون می خوا هیم از طریق خط تلفن با مودم ارتباط برقرار نمائیم . می بایست در دو قسمت مجزا تنظیمات انجام پذیرد :

بخش اول) محل قرارگیری FATEK :

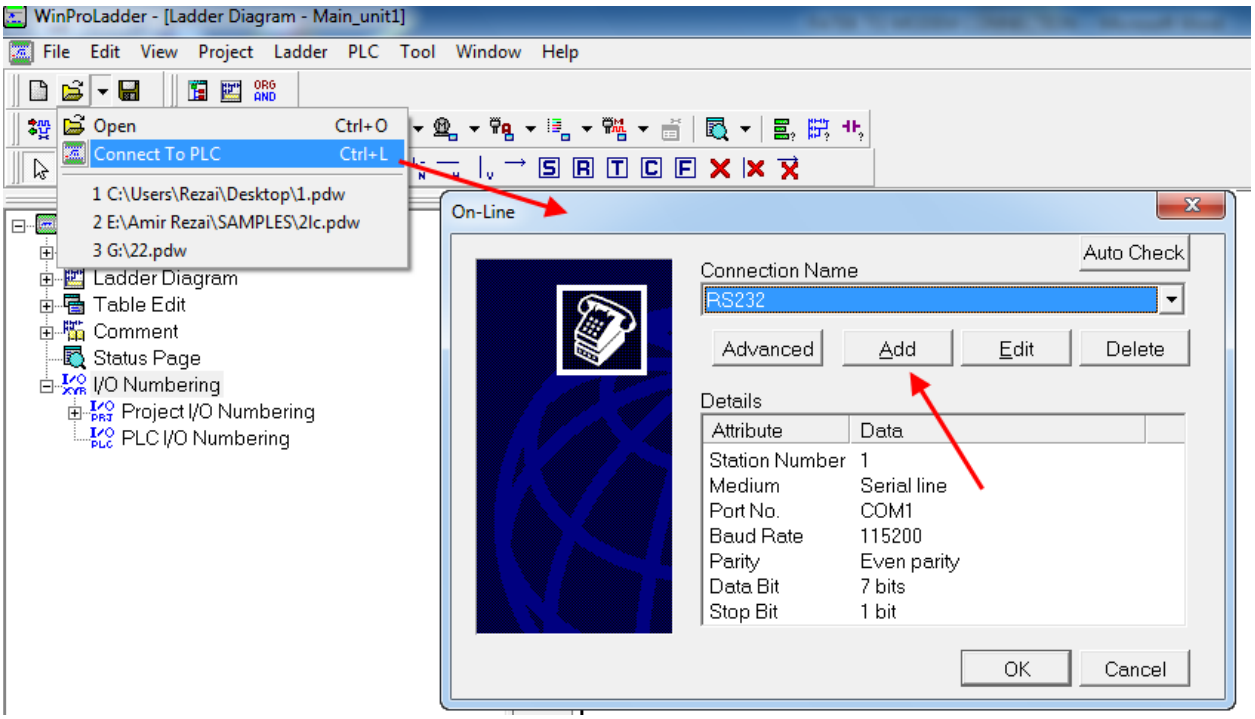
1- کابل ارتباطی بین FATEK و مودم را اتصال دهید (DM106) .

2- یک خط تلفن آزاد – با شماره تلفن مشخص – را به مودم وصل نمائید.

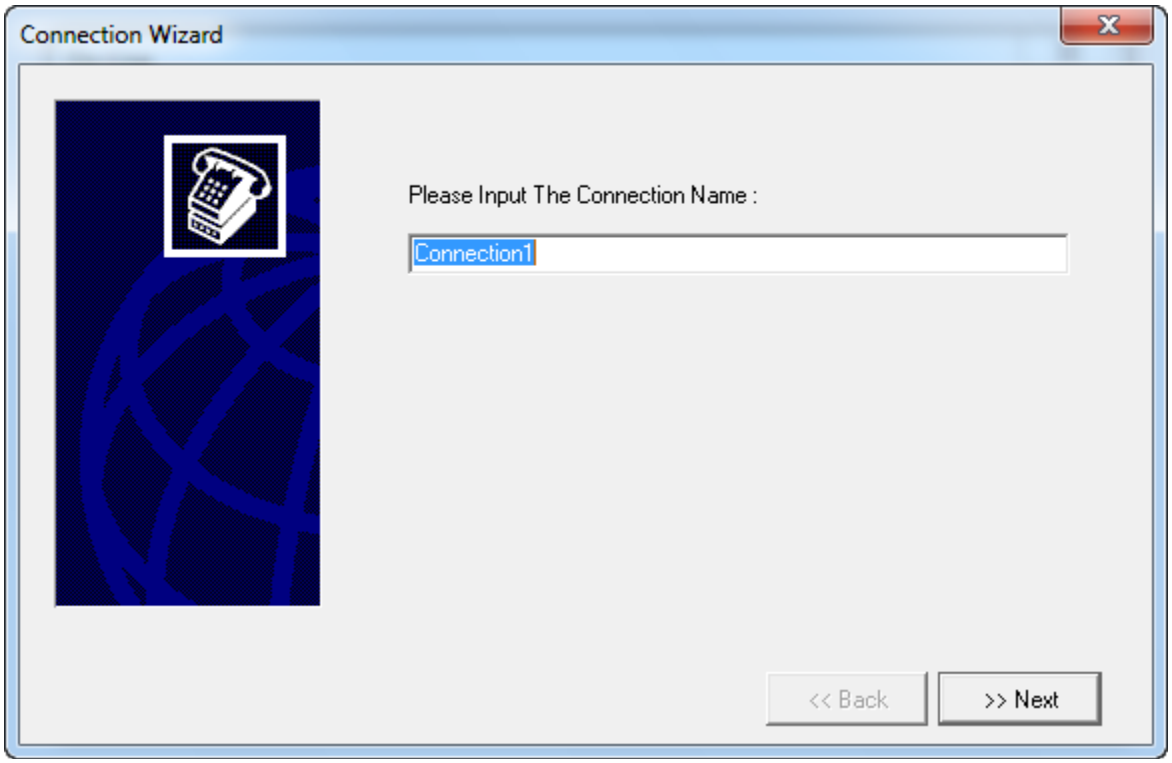
بخش دوم) محل قرارگیری PC :

1- یک خط تلفن آزاد ، به کامپیوتر خود وصل نمائید .

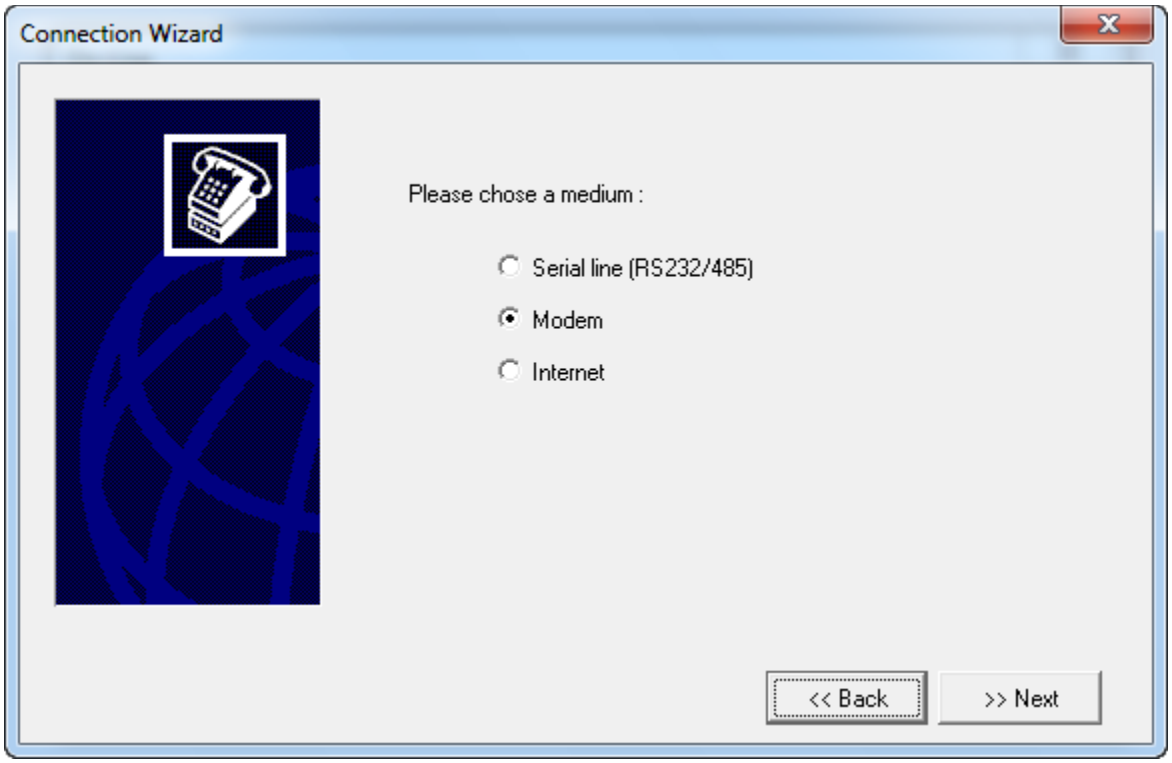
2- نرم افزار Winproladder را باز نموده



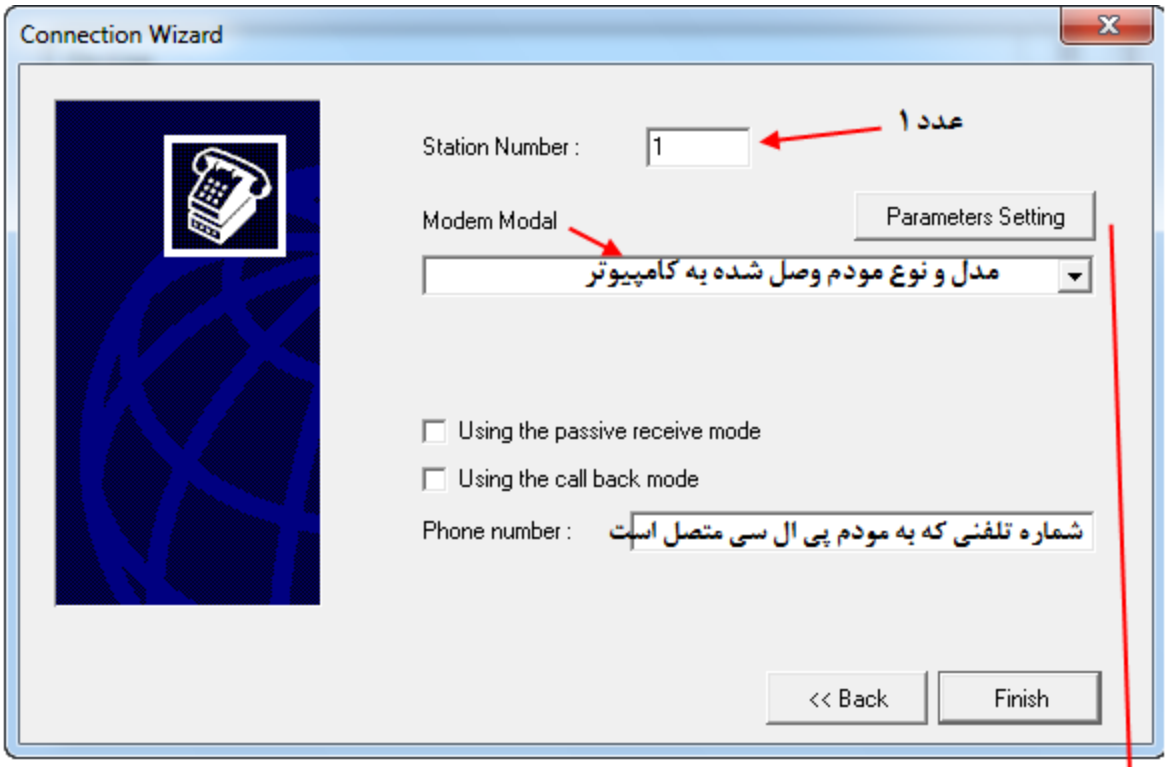
در این پنجره به روی آیکون Add کلیک کنید تا چنین پنجره ای نمایان گردد :

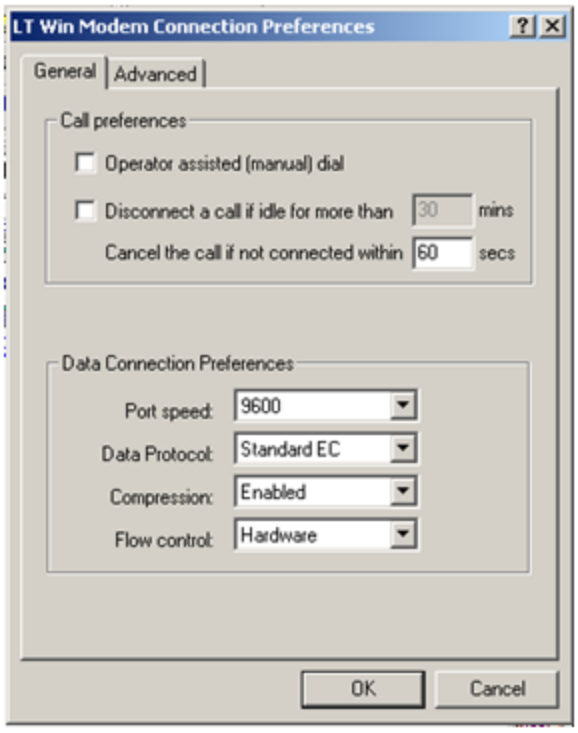


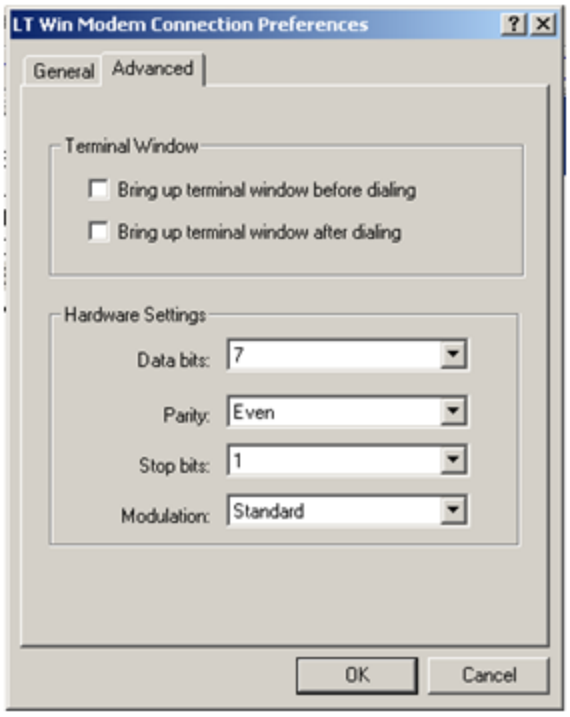
در این پنجره ، نامی برای اتصال خود انتخاب نمائید (به دلخواه) و آیکون Next را فشار دهید تا پنجره زیر باز شود :



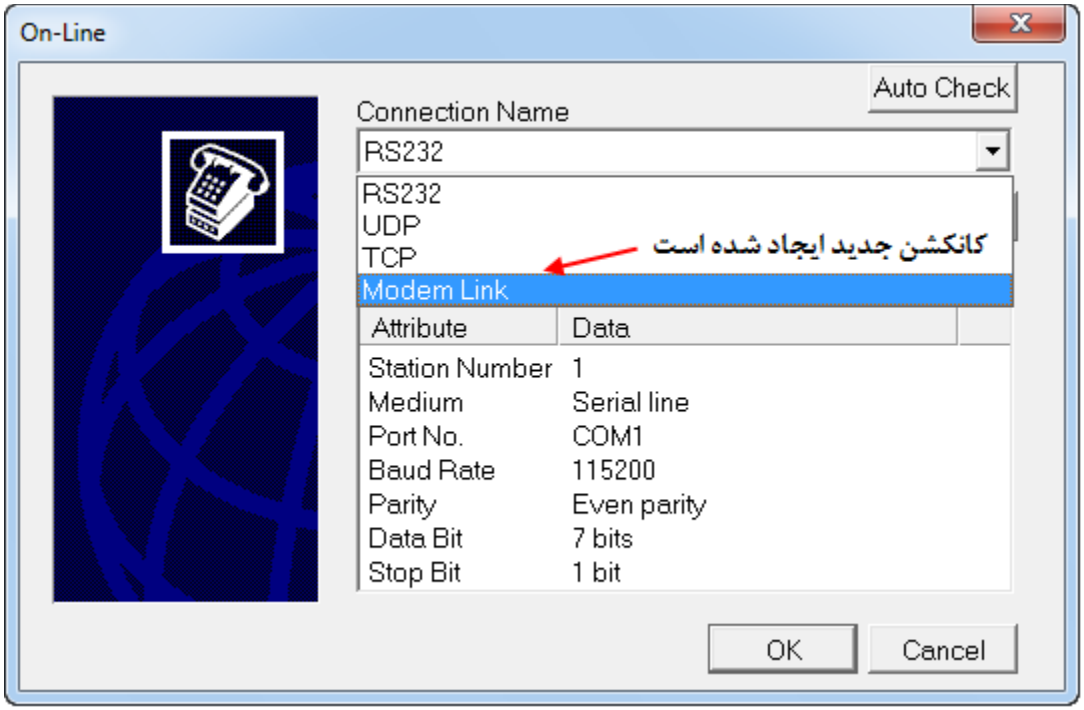
در این پنجره نیز گزینه ی Modem را انتخاب نموده و به بخش بعدی بروید :


Connection Wizard
Station Number :
1
عدد ۱
Modem Modal
Parameters Setting
مدل و نوع مودم وصل شده به کامپیوتر
Using the passive receive mode
Using the call back mode
Phone number :
شماره تلفنی که به مودم پی ال سی متصل است
<< Back
Finish


LT Win Modem Connection Preferences
General
Advanced
Call preferences
Operator assisted (manual) dial
Disconnect a call if idle for more than
30
mins
Cancel the call if not connected within
60
secs
Data Connection Preferences
Port speed:
9600
Data Protocol:
Standard EC
Compression:
Enabled
Flow control:
Hardware
OK
Cancel


LT Win Modem Connection Preferences
General
Advanced
Terminal Window
Bring up terminal window before dialing
Bring up terminal window after dialing
Hardware Settings
Data bits:
7
Parity:
Even
Stop bits:
1
Modulation:
Standard
OK
Cancel

سپس به روی آیکون Finish کلیک نمائید .



ریجیسترهای مربوط به شماره گیری و ایجاد ارتباط

Relay No.	Function	Description
M1959	Modem dialing signal selection	● 0: Dialing by TONE when Port 1 connecting with Modem. ● 1: Dialing by PULSE when Port 1 connecting wit Modem.
M1964	Modem dialing control	● If Port 1 is connected with Modem, when signal 0→1 will dial the phone number, when signal 1→0 will hang-up the phone.
M1965	Dialing success flag	● 1 : Indicating that dialing is successful (when Port 1 is connected with Modem).
M1966	Dialing fail flag	● 1 : Indicating that dialing has failed (when Port 1 is connected with Modem).

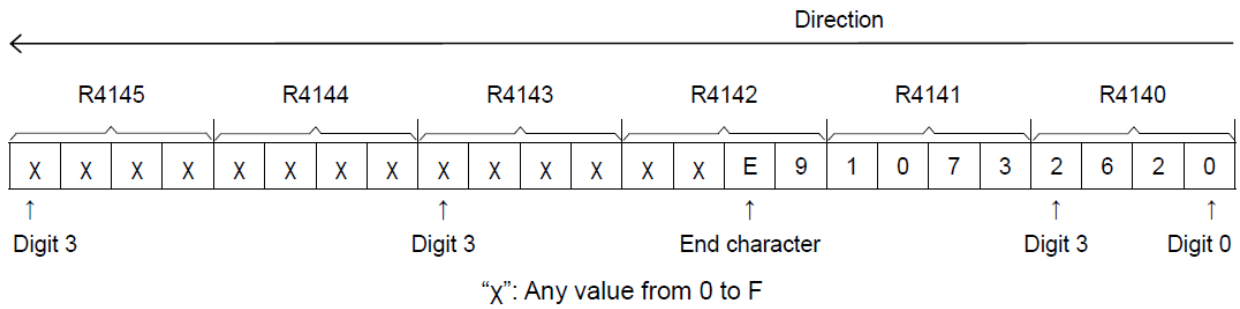
R4140	Telephone Number
R4141	Telephone Number
R4142	Telephone Number
R4143	Telephone Number
R4144	Telephone Number
R4145	Telephone Number

R4163	Modem dialing control setting	● Low Byte of R4163 = 1, Ignore the dialing tone and the busy tone when dialing. = 2, Wait the dialing tone but ignore the busy tone when dialing. = 3, Ignore the dialing tone but detect the busy tone when dialing. = 4, Wait the dialing tone and detect the busy tone when dialing. = Any other value treated as value equal 4.
		● High Byte of R4163 : The Ring count setting for Modem auto answer

R4149	Modem Interface Setting & Port0 without checking of station number for FATEK's external communication protocol	● High Byte of R4149 =55H, Remote-Diagnosis/Remote-CPU-Link by way of Port 1 through Modem connection, it supports user program controlled dial up function =AAH, Remote diagnosis by way of Port 1 through Modem connection , it supports Passive receiving & Active dialing operation mode =Others, without above function ● Low Byte of R4149 : =1, Port 0 without checking of station number for FATEK's external communication protocol (communicating with MMI/SCADA) ≠1, Others, Port 0 network for data acquisition.checks station number, it allows multi-drop

R4149 high byte = AAH means that Port 1 is set up to Modem-specific interface. Though CPU uses FATEK
"Standard Communication Driver" or "ModBus Communication Driver" to control the communication transaction of Port 1,
connection must be made via Modem.

Only the phone number that is stored in the Modem phone number register in the following format will be identified as effective by the PLC host. The phone number must be written hex-decimally. Only 0~9 and "E" are meaningful in the hexadecimal digits. "A" stands for dialing delay and is usually used for international calls or extensions of an automatic switchboard. (a "A" is about 2 seconds). "B" stands for "#" (for B.B.Call), and "C" stands form "*". Among the effective digits, 0~9 is used for phone numbers, while "E" stands for the end of a phone number. Since each register has 4 hexadecimal digits, R4140~R4145 have 24 hexadecimal digits and maximum 23 digits, the end character "E" not counted, can be stored in R4140~R4145. Phone numbers are stored in order from digit 0 of R4140 to digit 3 of R4145. For example, the phone number 02-6237019 is stored in the following order:



"X": Any value from 0 to F

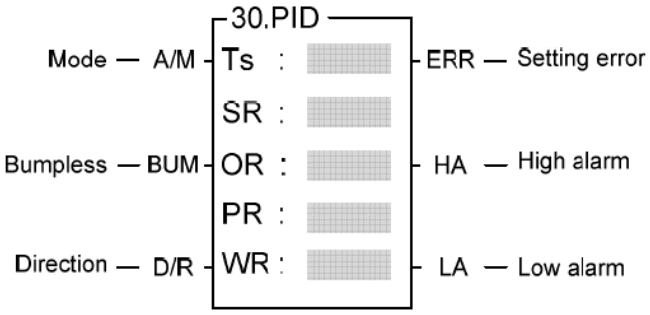
بنام خداوند بخشنده ومهربان

عنوان مدرک :	برنامه نویسی PLC FATEK
توضیحات :	FUN.30 PID GENERAL
تعداد صفحه :	6
شماره ویرایش :	2
ویرایش کننده :	ا.رضایی
تاریخ ویرایش :	1392.8.20

کنترل PID GENERAL برای کنترل سیستم های عمومی استفاده می شود

ورودی این سیستم از هر ریجیستری می تواند باشد بنابراین از این کنترل می توان به تعداد نامحدود استفاده کرد ولی از این فانکشن فقط می توان یک حلقه PID را تحت کنترل داشت .

Ladder symbol



Range / Ope-rand	HR	ROR	DR	K
	R0 | R3839	R5000 | R8071	D0 | D4095	
Ts	○	○	○	1~3000
SR	○	○*	○	
OR	○	○*	○	
PR	○	○*	○	
WR	○	○*	○	

- **اگر D/R صفر باشد , کنترل مانند دستگاه کولر عمل می کند , یعنی اگر مقدار Setpoint کمتر از مقدار دمای کنونی باشد , خروجی فعال می گردد .**
- **اگرD/R یک باشد , کنترل مانند دستگاه هیتر عمل می کند , یعنی اگر مقدار Setpoint بیشتر از مقدار دمای کنونی باشد , خروجی فعال می گردد .**

Ts : PID Operation time interval
SR : Starting register of process control parameter table comprised by 8 consecutive registers.
OR : PID output register
PR : Starting register of the process parameter table comprised by 7 consecutive registers.
WR : Starting register of working variable for PID internal operation. It requires 7 registers and can't be re-used in other part of the ladder program.

Ts: زمان رفرش کردن تابع(از ۱ تا ۳۰۰۰ می باشد و هر واحد برابر ۰.۰۱ است ، یعنی این تابع را می توان در جاههایی در هر ۳۰ ثانیه فعال کرد)

SR : آدرس شروع مربوط به ریجیسترهای تنظیمات تابع

SR+0	**بارگذاری توسط PID**
SR+1	Setpoint
SR+2	High Alarm
SR+3	Low Alarm
SR+4	**بالاترین مقداری که سنسور می تواند حس کند**
SR+5	**کمترین مقداری که سنسور می تواند حس کند**
SR+6	**ریجیستر ورودی حلقه (اگر D4004=0 باشد ورودی بصورت (12bit) و اگر D4004=1 باشد ورودی بصورت (14bit) مورد استفاده قرار می گیرد.)**
SR+7	**مقدار Offset ورودی آنالوگ (برای مثال اگر از سنسور ۴ تا ۲۰ میلی آمپر استفاده کنیم ، مقدار این ریجیستر باید ۳۲۷۶ باشد)**

OR: آدرس ریجیستر خروجی آنالوگ کنترل PID

PR: پارامترهای کنترل PID

$$Mn = [(D4005/Pb) \times En] + \sum_{0}^{n} [(D4005/Pb) \times Ti \times Ts \times En] - [(D4005/Pb) \times Td \times (PVn - PVn\text{-}1)/Ts] + Bias$$

Mn : Control output at time "n"

Pb : Proportional band (range : 2~5000, unit 0.1%. Kc (gain) =1000/ Pb)

Ti : Intergal time constant (range : 0~9999 corresponds to 0.00~99.99 Repeats/Minute)

Td : Differential time constant (range : 0~9999 corresponds to 0.00~99.99 Minutes)

PVn : Process value at time "n"

PV n-1 : Process value at time "n"

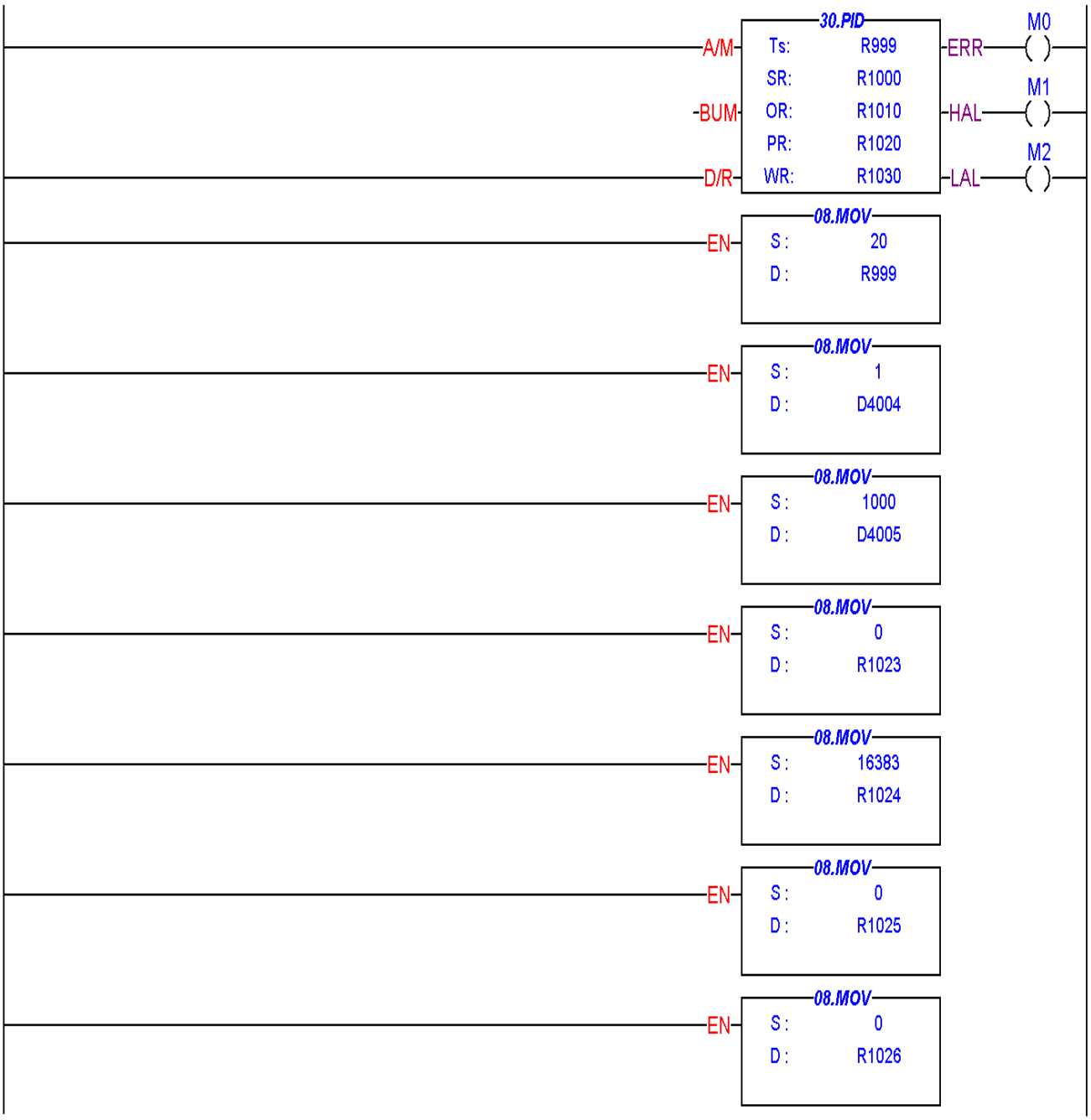
En :Error at time "n" =set value (SP) – process value at time "n" (PVn)

Ts : Interval time of PID calculation (range: 1~3000, unit : 0.01 S)

Bias : Control output offset (range: 0~16380)

PR+0	Pb , (2<Pb<5000) (("Gain"Kc = D4005/ Pb (D4005=1000)))
PR+1	ضریب انتگرال Ki 0<Ki<9999
PR+2	ضریب مشتق Td 0<Td<9999
PR+3	Control output offset (range: 0~16380)
PR+4	مصارف خاص ، مقدار ۱۶۳۸۳ قرار داده شود
PR+5	مصارف خاص ، مقدار ۰ قرار داده شود
PR+6	اگر مقدار ۰ قرار داده شود بمعنای PID استاندارد و اگر مقدار ۱ قرار داده شود بمعنای PI (در مواقعی که سیستم به پایداری نرسد از این گزینه استفاده می شود) می باشد.

مثال :


30.PID
A/M
Ts: R999
SR: R1000
-BUM
OR: R1010
PR: R1020
D/R
WR: R1030
ERR
M0
HAL
M1
LAL
M2
08.MOV
EN
S : 20
D : R999
08.MOV
EN
S : 1
D : D4004
08.MOV
EN
S : 1000
D : D4005
08.MOV
EN
S : 0
D : R1023
08.MOV
EN
S : 16383
D : R1024
08.MOV
EN
S : 0
D : R1025
08.MOV
EN
S : 0
D : R1026

پارامترهایی که باید از HMI یا در قسمت تنظیمات به PLC مقدار دهی شوند :

Status Monitoring

Ref. No.	Status	Data		Ref. No.	Status	Data	Ref. N
R1001	Decimal	50	Setpoint				
R1002	Decimal	70	High Alarm				
R1003	Decimal	30	High Alarm				
R1004	Decimal	100	سنسور می تواند حس کند	بالاترین مقداری که			
R1005	Decimal	10	سنسور می تواند حس کند	کمترین مقداری که			
R1020	Decimal	150	P ضریب				
R1021	Decimal	15	I ضریب				
R1022	Decimal	10	D ضریب				
R1006	Decimal	7300	Input				
R1010	Decimal	15695	Output				

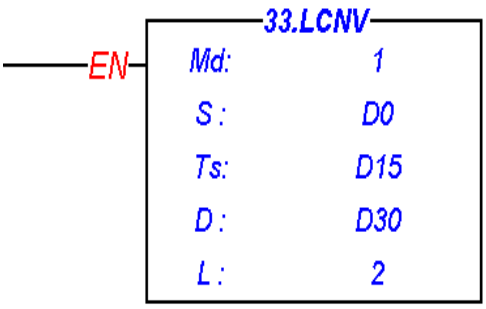
StatusPage0

بنام خداوند بخشنده ومهربان

عنوان مدرک :	برنامه نویسی FATEK PLC
توضیحات :	FUN.33 Linear Conversio
تعداد صفحه :	3
شماره ویرایش :	1
ویرایش کننده :	ا.رضایی
تاریخ ویرایش :	1391.11.16

این فانکشن ، عدد ورودی که بین بازه [A,B] می باشد را به عددی خروجی بین بازه [C,D] تبدیل می کند.

در این تابع ، عملگرهای زیر وجود دارند :



Md : (انتخاب حالت کاری (۰ تا ۳)

S : آدرس شروع جدول اطلاعات منبع

Ts : آد رس شروع جدول تبدیل

D : آدرس شروع ذخیره ی نتایج

L : تعداد رجیسترهای تبدیل یافته (۱ تا ۶۴)

Md :

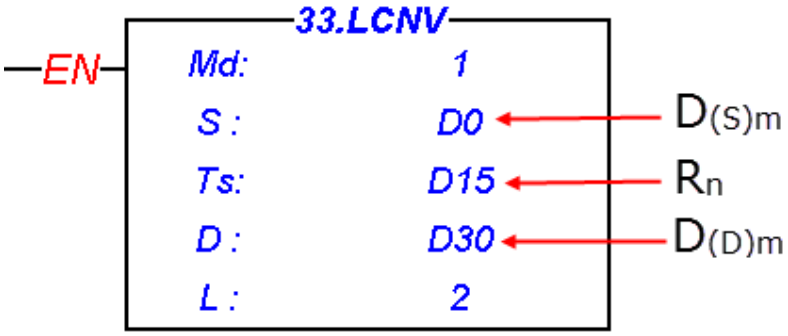
مد صفر : تبدیل عدد ورودی از بازه [A,B] به عدد خروجی بین بازه [C,D]
مد ۱ : در این مد می توان چندین تبدیل بازه هابه یکدیگر را با نوشتن فقط یک فانکشن اجرا کرد
مد ۲ : تبدیل به تابع خطی درجه ۱ D=(A/B)D+C
مد ۳ : در این مد می توان چندین تابع خطی درجه ۱ را با نوشتن فقط یک فانکشن اجرا کرد

S : محل نوشتن ریجیستر ورودی (وقتی که از مد ۱ یا ۳ استفاده کنیم چند ریجیستر ورودی یکی بعد از دیگری باید وارد شوند)

Ts : محل وارد کردن رنج بازه ها

D : محل نوشتن ریجیستر خروجی (وقتی که از مد ۱ یا ۳ استفاده کنیم چند ریجیستر خروجی یکی بعد از دیگری باید وارد شوند)

مثال :



به ازای ورودی های زیر (S)	Ts برای مد صفر	خروجی D
$D_{(S)m}$ →	$[R_n,R_{n+1}]$ → $[R_{n+2}, R_{n+3}]$	→ $D_{(D)m}$

به ازای ورودی های زیر (S)	Ts برای مد 1	خروجی D
$D_{(S)m}$ →	$[R_n,R_{n+1}]$ → $[R_{n+2}, R_{n+3}]$	→ $D_{(D)m}$
$D_{(S)m+1}$ →	$[R_{n+4},R_{n+5}]$ → $[R_{n+6}, R_{n+7}]$	→ $D_{(D)m+1}$
$D_{(S)m+2}$ →	$[R_{n+8},R_{n+9}]$ → $[R_{n+10}, R_{n+11}]$	→ $D_{(D)m+2}$

به ازای ورودی های زیر (S)	Ts برای مد **2**	خروجی D
$D_{(S)m+1}$ →	$A1=R_n$ $B1=R_{n+1}$ $C1=R_{n+2}$	→ $D_{(D)m+1}=(A1/B1)D+C1$

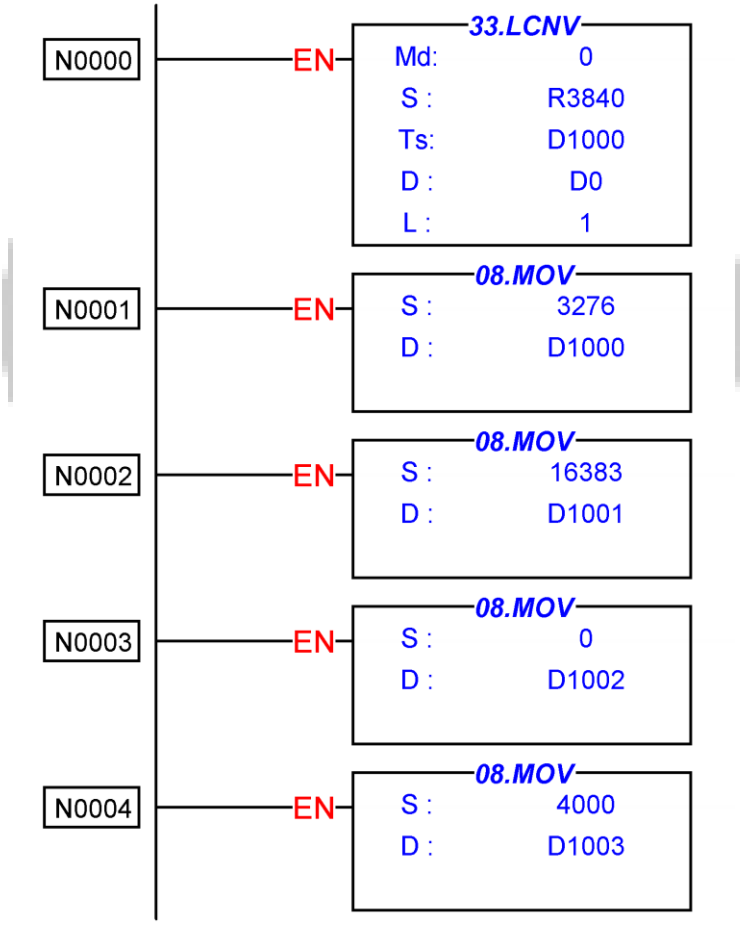
به ازای ورودی های زیر (S) :	Ts برای مد **3**	خروجی D
$D_{(S)m}$ →	$A1=R_n$ $B1=R_{n+1}$ $C1=R_{n+2}$	→ $D_{(D)m+1}=(A1/B1)D+C1$
$D_{(S)m+1}$ →	$A2=R_{n+3}$ $B2=R_{n+4}$ $C2=R_{n+5}$	→ $D_{(D)m+2}=(A2/B2)D+C2$
$D_{(S)m+2}$ →	$A3=R_{n+6}$ $B3=R_{n+7}$ $C3=R_{n+8}$	→ $D_{(D)m+3}=(A3/B3)D+C3$

مثال ۱ :

سنسوری داریم که به ازا ی "صفر بار" خروجی "۴میلی آمپر" و به ازای "۴۰۰۰میلی بار" خروجی "۲۰میلی آمپر" ایجاد می کند ، می خواهیم این سنسور را کالیبره کنیم :

پاسخ : به ازای ۴میلی آمپر در ورودی آنالوگ پی ال سی عدد ۳۲۷۶ نمایش داده می شود و به ازای ۲۰ میلی آمپر عدد ۱۶۳۸۳ نمایش داده می شود پس وقتی فشار بین صفر بار تا ۴۰۰۰ میلی بار تغییر کند ورودی آنالوگ عددی بین ۳۲۷۶ تا ۱۶۳۸۳ می باشد .

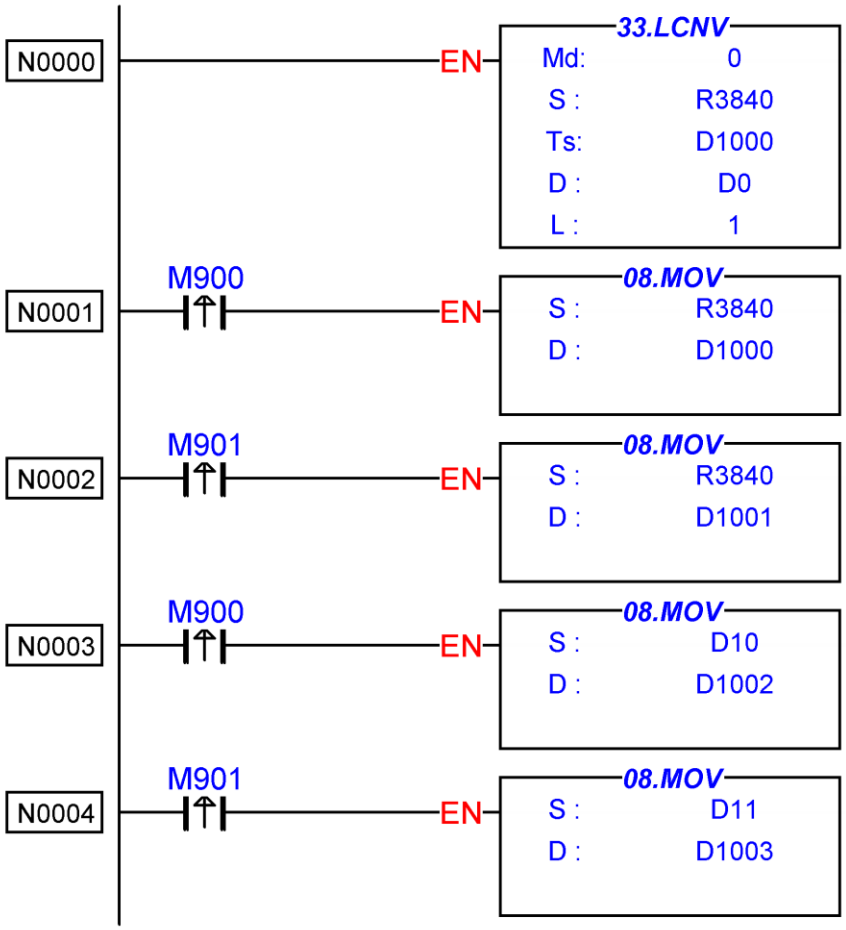
(در برنامه زیر R3840 ورودی آنالوگ و D0 مقدار واقعی فشار می باشد)



مثال ۲ :

لودسلی داریم که می خواهیم آنرا در وزن استاندارد که از روی صفحه HMI وارد کردیم کالیبره کنیم :

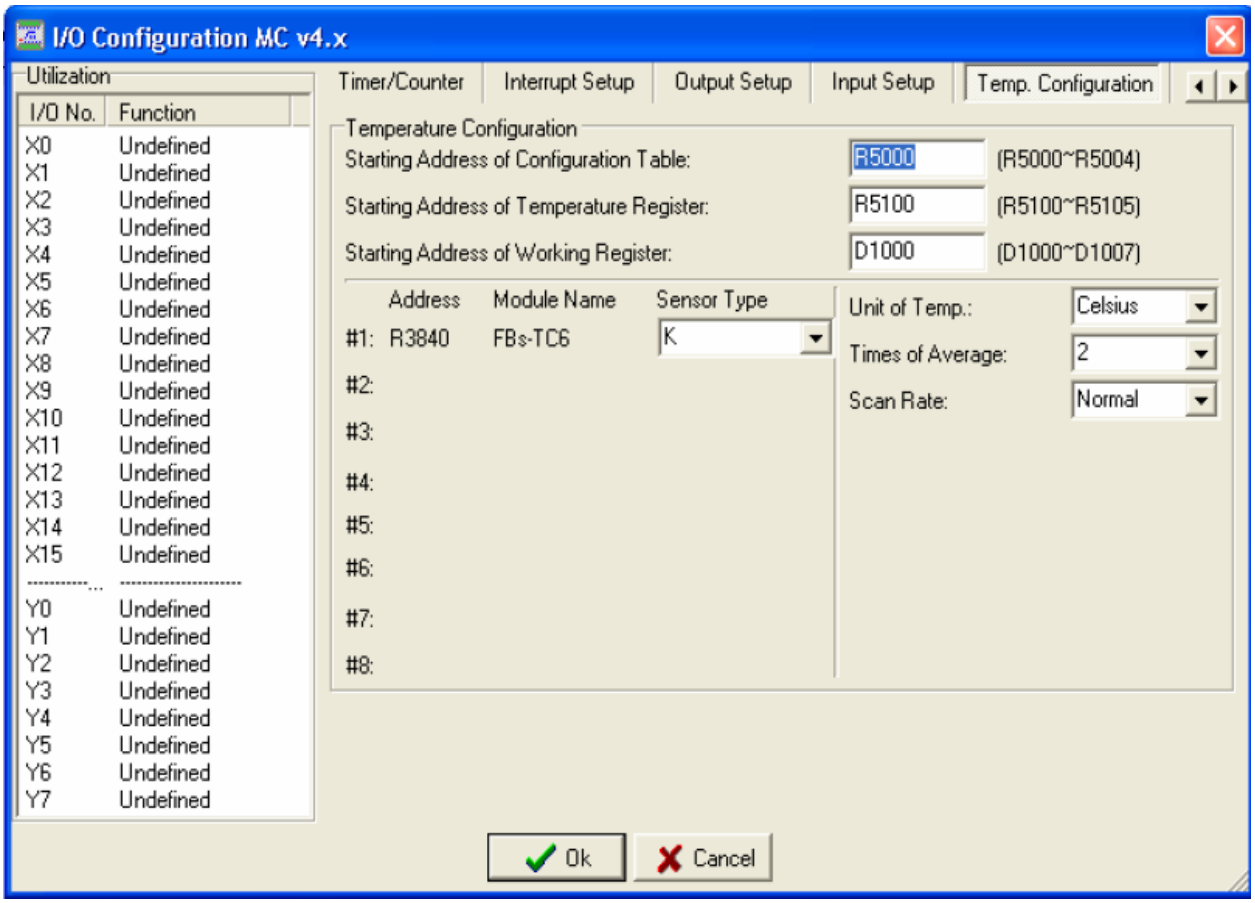
D10	نقطه کالیبره ۱ (مثلا صفر کیلوگرم)	M900	تایید کالیبره حد پایین
D11	نقطه کالیبره ۱ (مثلا ۵۰ کیلوگرم)	M901	تایید کالیبره حد بالا

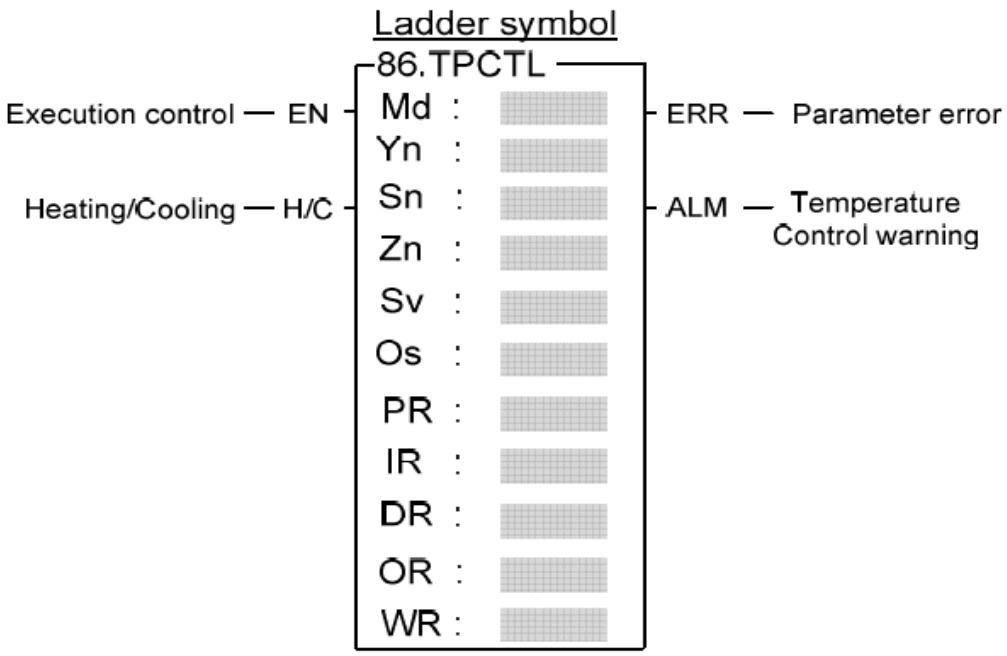


بنام خداوند بخشنده ومهربان

عنوان مدرک :	برنامه نویسی PLC FATEK
توضیحات :	FUN.86 PID TEMPERATURE
تعداد صفحه :	5
شماره ویرایش :	1
ویرایش کننده :	ا.رضایی
تاریخ ویرایش :	1391.6.27

برای استفاده از تابع کنترل دما لازم است که ابتدا یک ماژول دما به PLC وصل شود و ترموکوپل مربوطه نیز به این ماژول متصل گردد و همچنین در جدول I/O Configuration ، مقادیری برای آیتم های موجود در پنجره Temperature Configurationتعیین گردد (که همانطور که در شکل زیر می بینیم با R5000 و R5100 و D1000 مقداردهی شده اند) و در این صورت مقادیر دمای جاری که توسط ترموکوپل خوانده می شود توسط R5100 قابل دستیابی خواهد بود .





- اگر H/C صفر باشد , کنترل مانند دستگاه کولر عمل می کند , یعنی اگر مقدار Setpoint کمتر از مقدار دمای کنونی باشد , خروجی فعال می گردد .
- اگر H/C یک باشد , کنترل مانند دستگاه هیتر عمل می کند , یعنی اگر مقدار Setpoint بیشتر از مقدار دمای کنونی باشد , خروجی فعال می گردد .

Md : انتخاب متد PID که شامل دو بخش است و توسط دو عدد صفر و یک مقداردهی می شوند.این دو عدد عبارتند از :

مد صفر : حداقل Overshot , فقط P , I می باشد و D ندارد

مد یک : حلقه PID عمومی

که برای استفاده از حلقه PID لازم است که متد یک انتخاب گردد ،

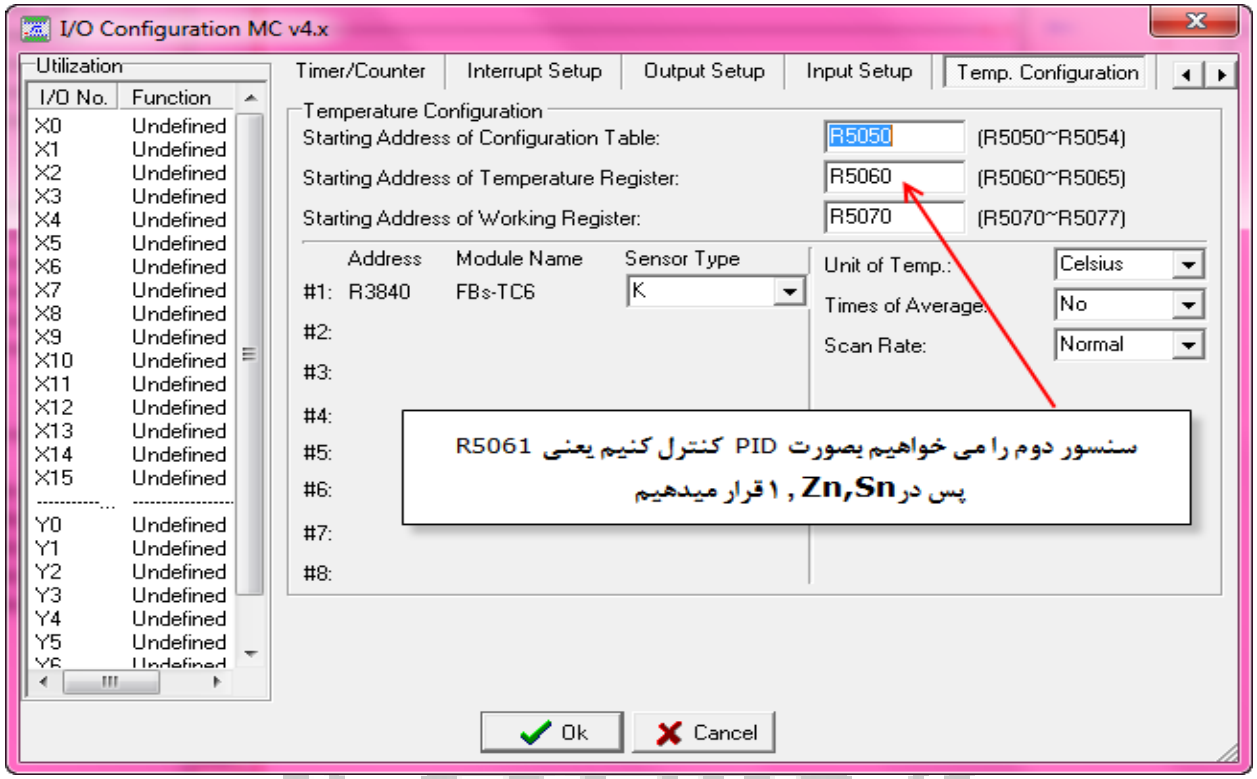
به ازای سنسورهای بعدی, ریجیسترهای بعد از ریجیسترهای نوشته شده در پارامترهای زیر مورد استفاده قرار می گیرند (به تعداد Zn)

با نوشتن یک تابع PID می توان 32 حلقه کنترل PID ایجاد کرد .

Yn : این پارامتر خروجی PID که می تواند فن یا هیتر یا المنت باشد را مشخص می کند .

Sn : شماره شروع سنسور دما که در I/O Configuration تنظیم می شود .

Zn : تعداد سنسورهایی که بعد از Sn می خواهیم با PID کنترل شوند (سنسورهایی که می خواهیم با PID کنترل کنیم باید پشت هم باشند) .



Sv : مقدار Setpoint دمای کنترل.

Os : میزان انحراف مجاز .

به عنوان مثال اگر Setpoint=2000 یعنی دما برروی 200 درجه سانتیگراد تنظیم شده باشد

و OS=50 باشد, میزان انحراف مجاز 50 برابر واحد دما خواهد بود (هر واحد 0.1 درجه را مشخص می کند)

1950 < 2000 < 2050

(195 C) < (200 C) < (205 C)

PR : در این قسمت رجیستر مربوط به مقدار گین برای حلقه PID مشخص می گردد (معمول : 100)

IR : در این قسمت رجیستر مربوط به مقدار ضریب انتگرال برای حلقه PID . مشخص می گردد (معمول : 12)

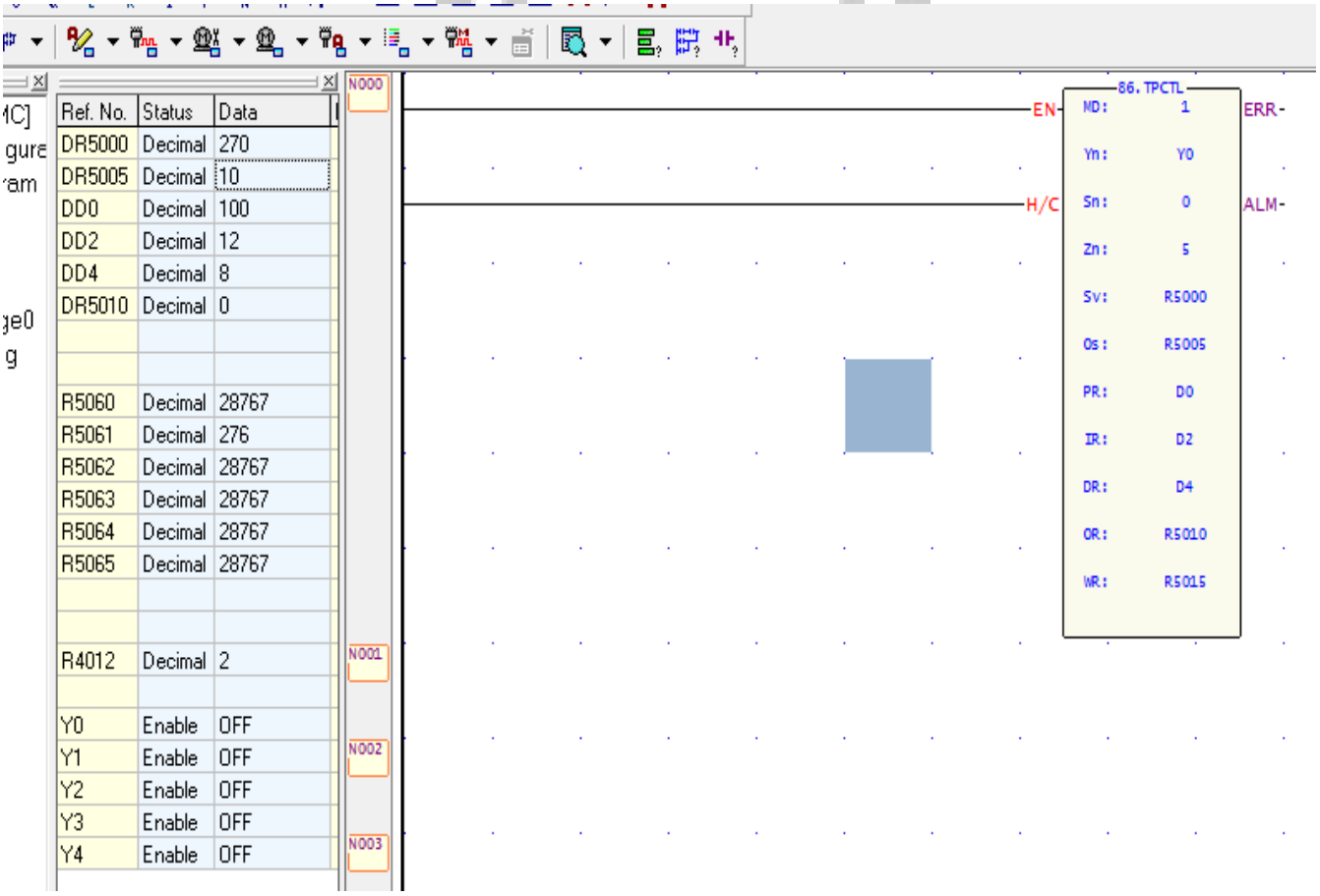
DR : در این قسمت رجیستر مربوط به مقدار ضریب مشتق برای حلقه PID . مشخص می گردد(معمول : 8)

OR : خروجی آنالوگ محاسبات PID . که مقدار آن از صفر تا 16383 تغییر می کند

WR : Working Register ها را برای حلقه PID مشخص می کند که 9WORD می باشد.

ریجیستر 32 بیتی R4012 برای فعال کردن سنسور در کنترل PID می باشد به این معنی که به ازای هر بیت یکی از سنسورها فعال می گردد .

بیت صفر سنسور شماره صفر در I/O Configuration و بیت یک برای سنسور شماره 1 و... , چنانچه هر کدام از بیتهای مربوط به ریجیستر 1 شوند, سنسور مربوط به آن در کنترل PID قرار خواهد گرفت .



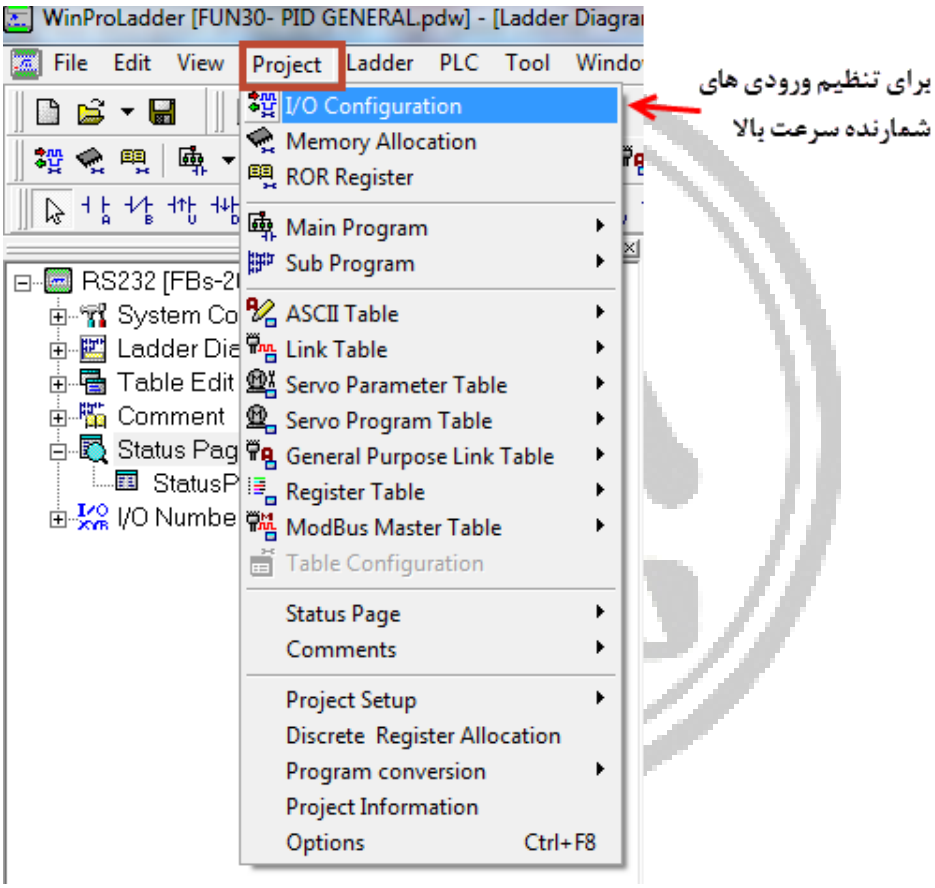
بنام خداوند بخشنده ومهربان

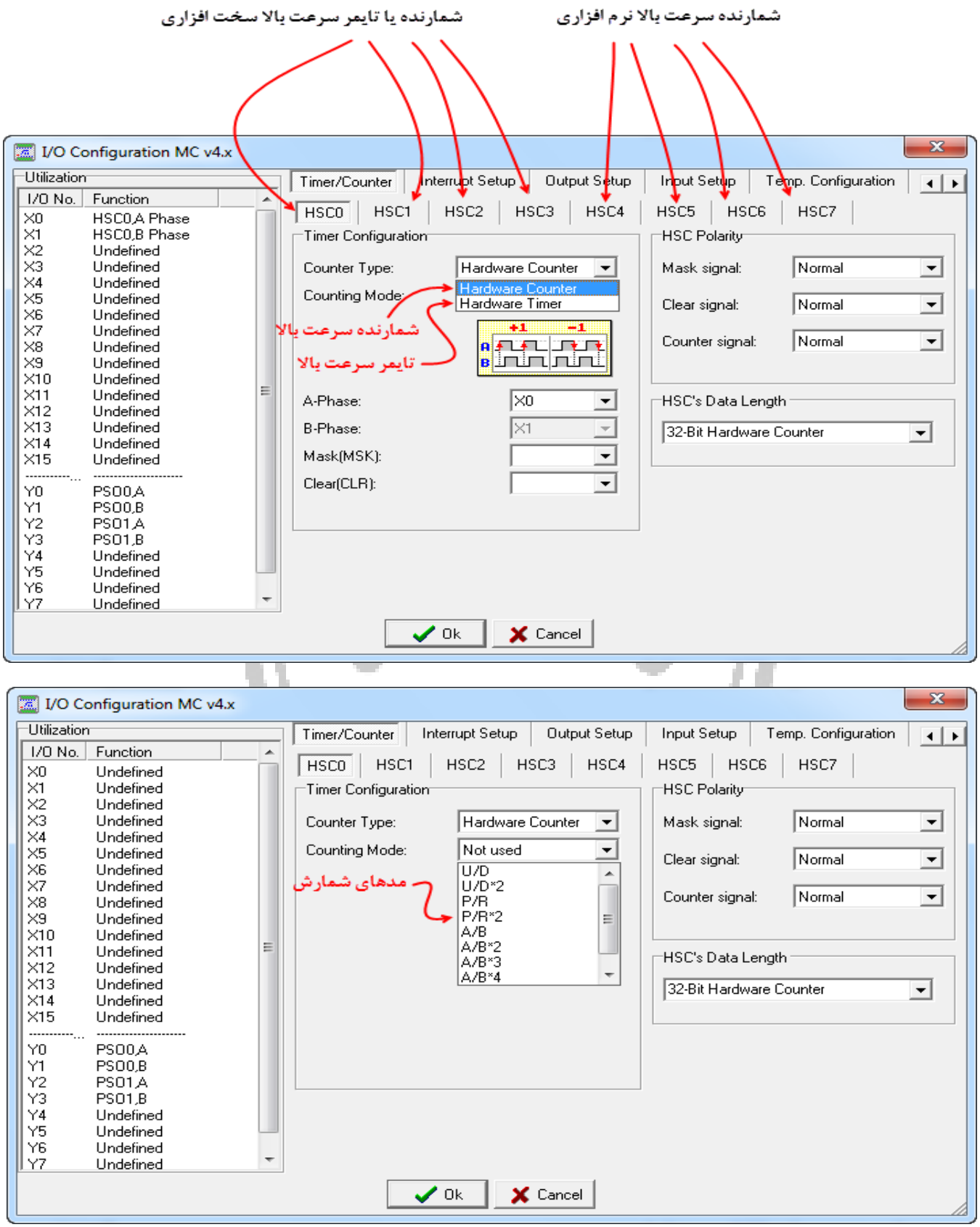
عنوان مدرک :	برنامه نویسی PLC FATEK
توضیحات :	FUN 92&93 High speed counter
تعداد صفحه :	10
شماره ویرایش :	1
ویرایش کننده :	ا.رضایی
تاریخ ویرایش :	1391.7.10

دو نوع شمارنده داریم : سرعت بالا و سرعت پایین (منظور از سرعت فرکانس ورودی می باشد) .

شمارنده های سرعت بالا (در مقیاس کیلوهرتز) نیاز به سخت افزار خاصی دارند تا بتوانند در این فرکانس پالسها را بشمارند ولی شمارنده های سرعت پایین (در مقیاس دهها هرتز) با ورودی های معمولی PLC نیز کار می کنند. به شمارنده های سرعت بالا شمارنده های سرعت بالای سخت افزاری و به شمارنده های سرعت پایین شمارنده های نرم افزاری می گویند.

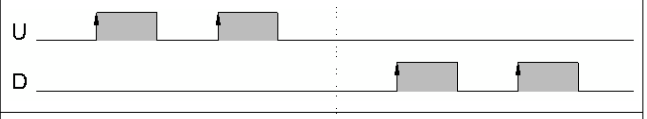
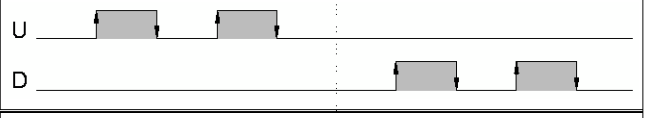
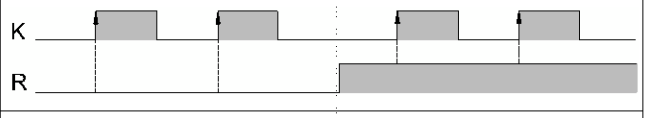
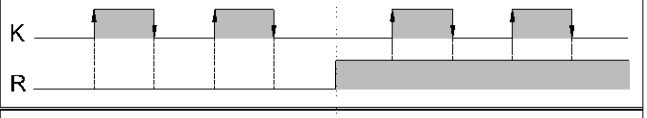
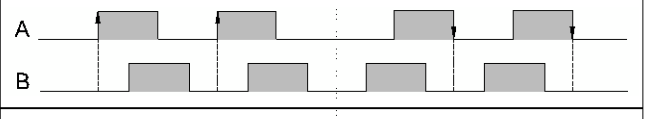
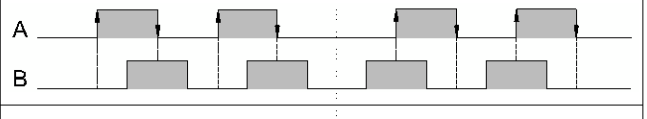
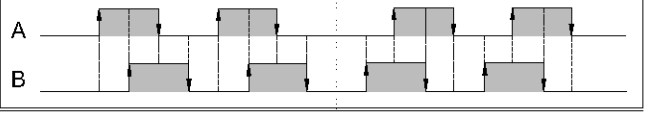
تنظیم ورودی ها برای حالت شمارنده بودن :



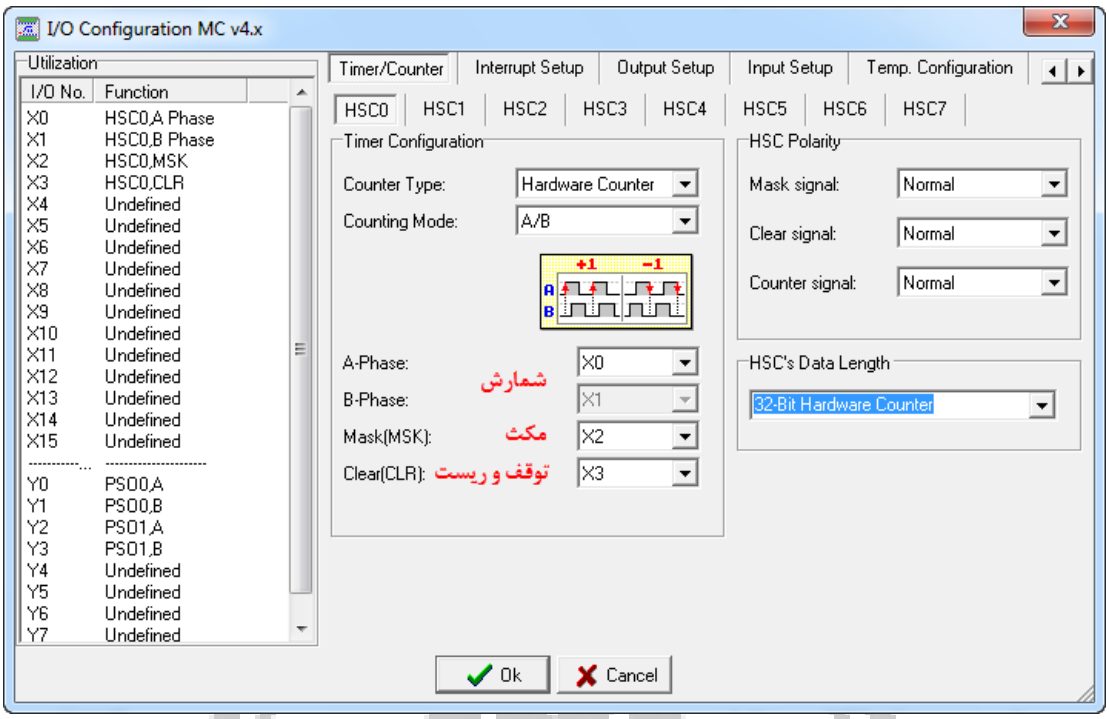

شمارنده یا تایمر سرعت بالا سخت افزاری
شمارنده سرعت بالا نرم افزاری
I/O Configuration MC v4.x
Utilization
I/O No. Function
X0 HSC0,A Phase
X1 HSC0,B Phase
X2 Undefined
X3 Undefined
X4 Undefined
X5 Undefined
X6 Undefined
X7 Undefined
X8 Undefined
X9 Undefined
X10 Undefined
X11 Undefined
X12 Undefined
X13 Undefined
X14 Undefined
X15 Undefined
Y0 PSO0,A
Y1 PSO0,B
Y2 PSO1,A
Y3 PSO1,B
Y4 Undefined
Y5 Undefined
Y6 Undefined
Y7 Undefined
Timer/Counter
Interrupt Setup
Output Setup
Input Setup
Temp. Configuration
HSC0 HSC1 HSC2 HSC3 HSC4 HSC5 HSC6 HSC7
Timer Configuration
Counter Type: Hardware Counter
Counting Mode: Hardware Counter
Hardware Timer
شمارنده سرعت بالا
تایمر سرعت بالا
+1 -1
A
B
A-Phase: X0
B-Phase: X1
Mask(MSK):
Clear(CLR):
HSC Polarity
Mask signal: Normal
Clear signal: Normal
Counter signal: Normal
HSC's Data Length
32-Bit Hardware Counter
Ok
Cancel

I/O Configuration MC v4.x
Utilization
I/O No. Function
X0 Undefined
X1 Undefined
X2 Undefined
X3 Undefined
X4 Undefined
X5 Undefined
X6 Undefined
X7 Undefined
X8 Undefined
X9 Undefined
X10 Undefined
X11 Undefined
X12 Undefined
X13 Undefined
X14 Undefined
X15 Undefined
Y0 PSO0,A
Y1 PSO0,B
Y2 PSO1,A
Y3 PSO1,B
Y4 Undefined
Y5 Undefined
Y6 Undefined
Y7 Undefined
Timer/Counter
Interrupt Setup
Output Setup
Input Setup
Temp. Configuration
HSC0 HSC1 HSC2 HSC3 HSC4 HSC5 HSC6 HSC7
Timer Configuration
Counter Type: Hardware Counter
Counting Mode: Not used
مدهای شمارش
U/D
U/D*2
P/R
P/R*2
A/B
A/B*2
A/B*3
A/B*4
HSC Polarity
Mask signal: Normal
Clear signal: Normal
Counter signal: Normal
HSC's Data Length
32-Bit Hardware Counter
Ok
Cancel

مدهای کاری هر شمارنده :

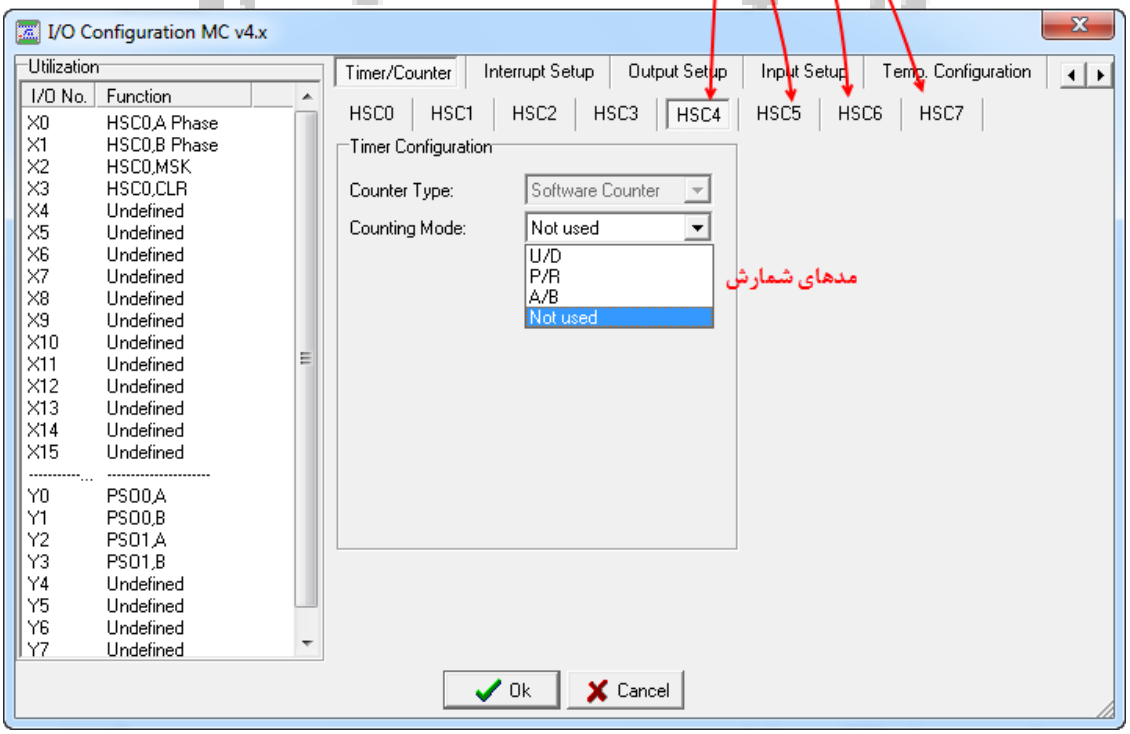
Counting Mode			HHSC (HSC0～HSC3)	SHSC (HSC4～HSC7)	Counting Waveform: Up Counting (+1) / Down Counting (−1)
Up-down pulse	MD 0	U/D	○	○	
	MD 1	U/D×2	○		
Pulse-direction	MD 2	K/R	○	○	
	MD 3	K/R×2	○		
AB phase	MD 4	A/B	○	○	
	MD 5	A/B×2	○		
	MD 6	A/B×3	○		
	MD 7	A/B×4	○		

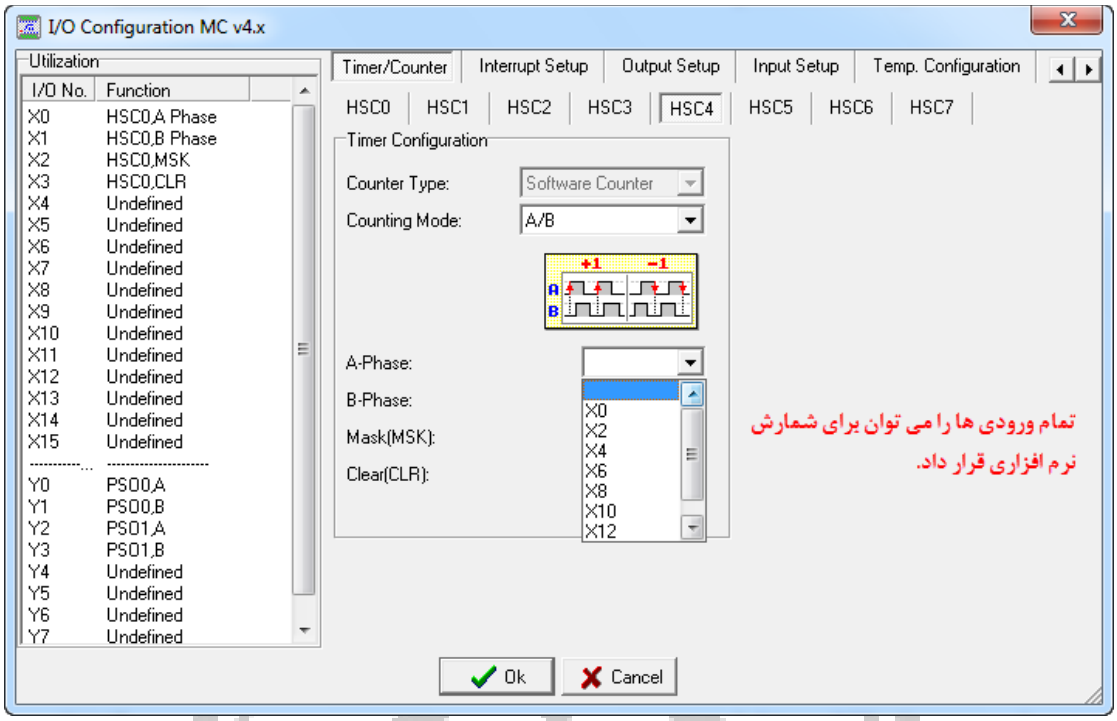
• The up/down arrow (↑ , ↓) on the positive/negative edge in the waveform represents where counting (+1 or –1) occurs.



شمارنده سرعت بالا نرم افزاری

شمارنده های نرم افزاری براساس زمان اسکن برنامه ورودی
ها را می شمارد (تا دهها هرتز ، مطابق با زمان اسکن برنامه)





نحوه خواندن شمارنده های سرعت بالا :

به ازای هر کانال ورودی شمارشگر 2 ریجیستر 32 بیتی اختصاص داده شده است :

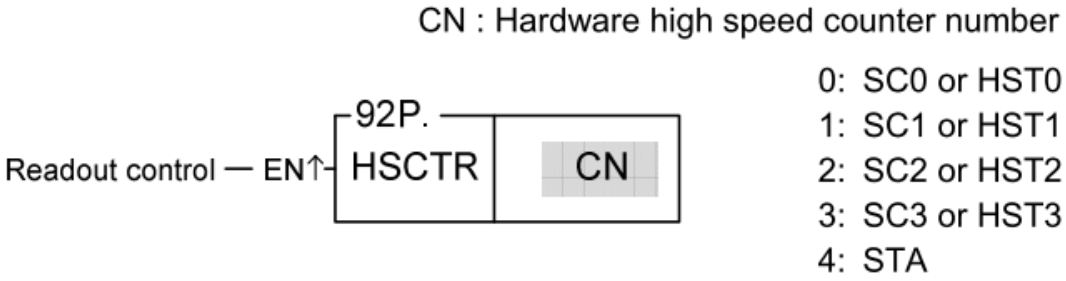
Current Value : CV مقدار ی که CPU هم اکنون در حال شمارش است ، دو استفاده از آن می توان داشت :

۱- قرار دادن مقدار صفر در ریجیستر شمارش کنونی (ریست کردن شمارشگر) ، توسط فانکشن 93 "HSCTW"

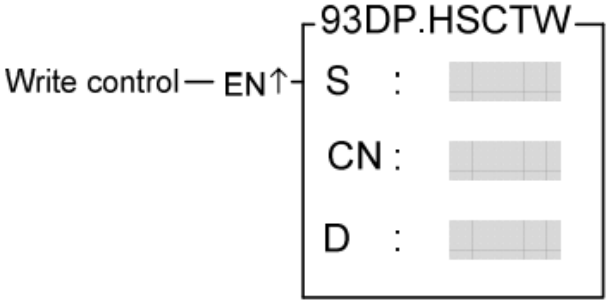
۲- خواندن مقدار کنونی شمارش شده ، توسط فانکشن 92 "HSCTR"

Preset Value : PV ، وقتی شمارشگر به این مقدار رسید اینتراپت شمارشگر فعال می شود و دستورات اینتراپت برای یک بار اجرا می شود .

فانکشن 92 (HSCTR) : برای کپی کردن مقدار کنونی شمارنده به داخل ریجیستر از پیش تعریف شده برای هر شمارنده



فانکشن 93 (HSCTR) : برای کپی اعداد به حافظه های پیش فرض اختصاص داده شده مربوط به شمارنده ها در CPU می باشند.



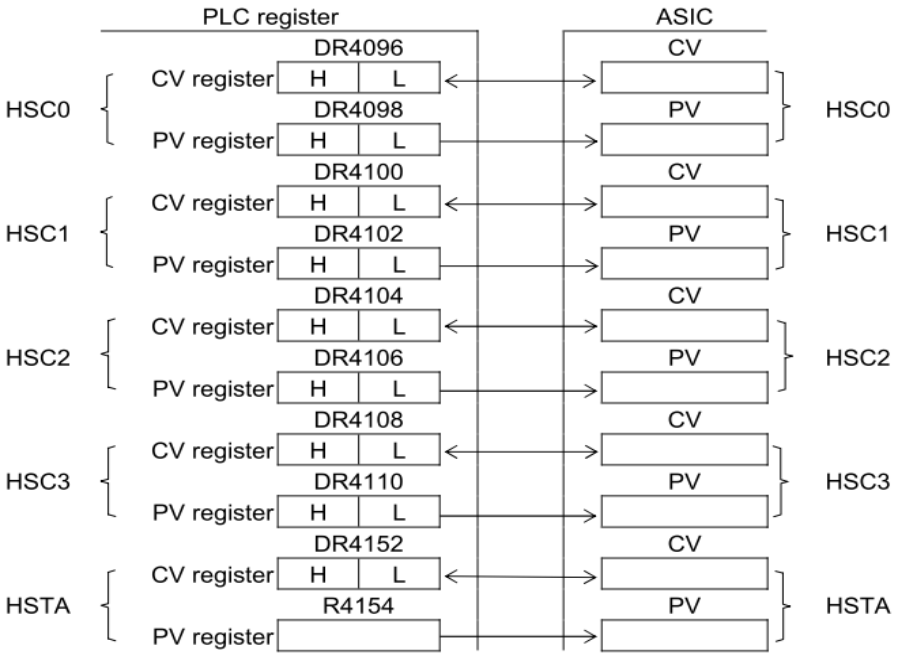
S : The source data for writing

CN : Hardware high speed counter to be written

0: HSC0 or HST1
1: HSC1 or HST2
2: HSC2 or HST3
3: HSC3 or HST4
4: HSTA

D : Write target (0 represents CV, 1 represents PV)

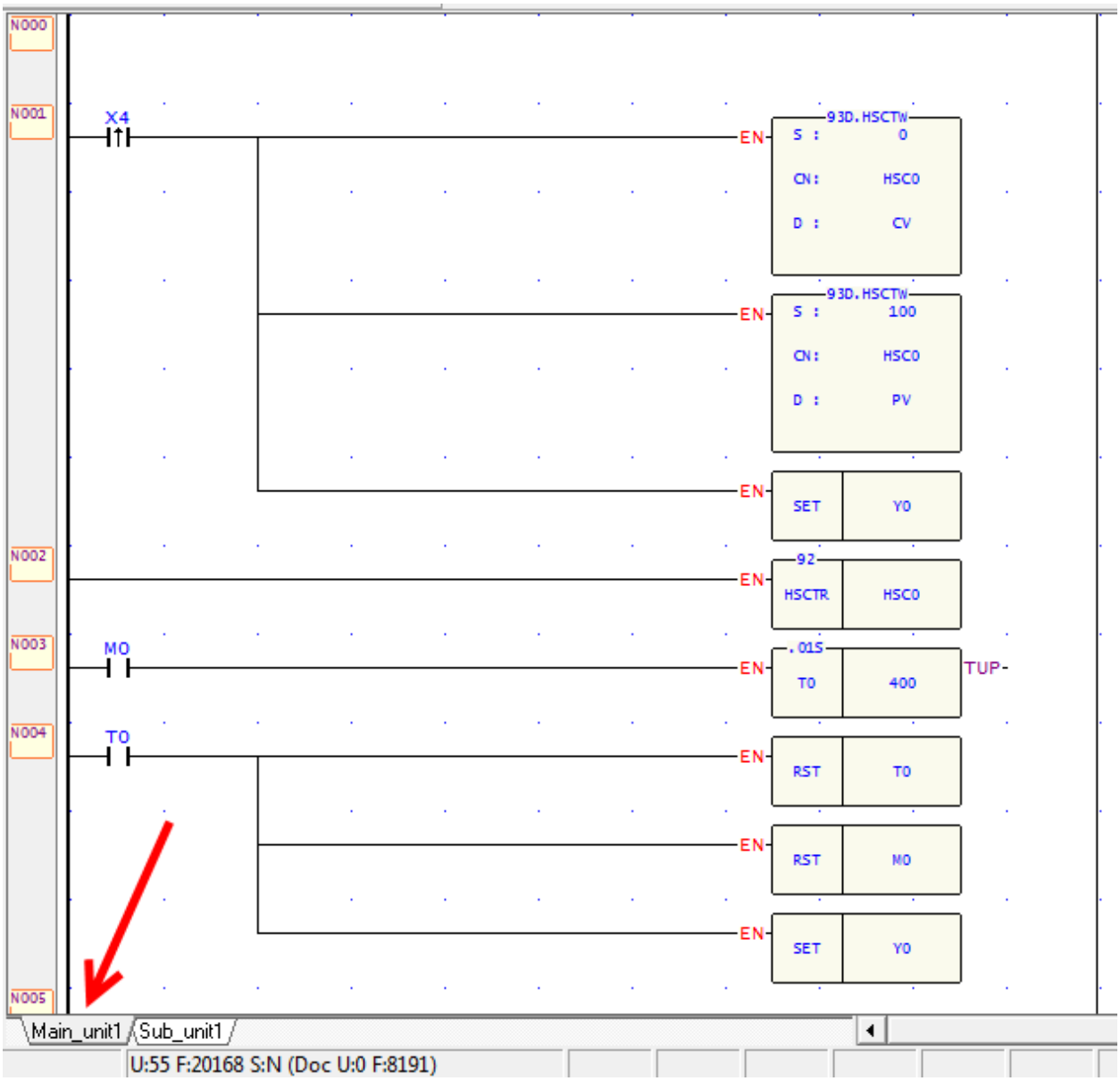
حافظه های پیش فرض اختصاص داده شده مربوط به شمارنده ها :

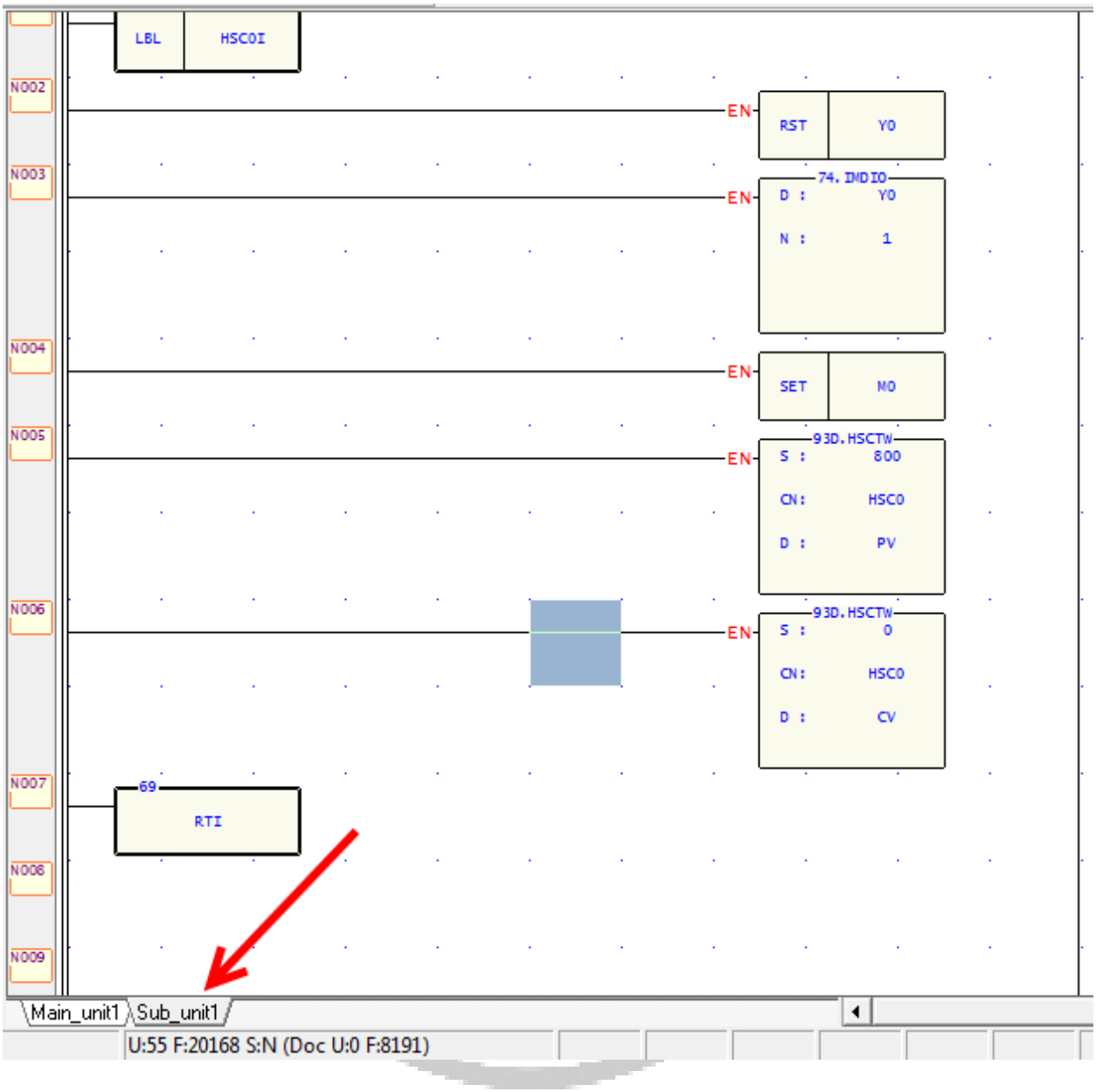


Type / Signal Allowed		MC/MN								MA
		HHSC				SHSC				SHSC
		HSC0	HSC1	HSC2	HSC3	HSC4	HSC5	HSC6	HSC7	HSC4 ~ HSC7
CV Register		DR4096	DR4100	DR4104	DR4108	DR4112	DR4116	DR4120	DR4124	The same as SHSC of MC/MN
PV Register		DR4098	DR4102	DR4106	DR4110	DR4114	DR4118	DR4122	DR4126	The same as SHSC of MC/MN
Counting Input	U,K or A	X0	X4	X8	X12	X0~X15	X0~X15	X0~X15	X0~X15	The same as SHSC of MC/MN
	D,R or B	X1	X5	X9	X13	X0~X15*	X0~X15*	X0~X15*	X0~X15*	The same as SHSC of MC/MN
Control Input	Mask	X2	X6	X10	X14	X0~X15	X0~X15	X0~X15	X0~X15	The same as SHSC of MC/MN
	Clear	X3	X7	X11	X15	X0~X15	X0~X15	X0~X15	X0~X15	The same as SHSC of MC/MN
Software MASK Relay		M1940	M1946	M1976	M1979	M1982	M1984	M1986	M1988	The same as SHSC of MC/MN
Software CLEAR Relay		M1941	M1947	M1977	M1980	Clear the Current Value Register directly				
Software Direction Selection(MD2,3 Only)		M1942	M1948	M1978	M1981	M1983	M1985	M1987	M1989	The same as SHSC of MC/MN
Interrupt Subroutine Label		HSC0I	HSC1I	HSC2I	HSC3I	HSC4I	HSC5I	HSC6I	HSC7I	The same as SHSC of MC/MN

وقفه های مربوط به شمارنده های سرعت بالا : وقتی **CV=PV** شود سیکل برنامه وارد اینتراپت مورد نظر می شود

مثال : برنامه ایی که با تحریک ورودی X4 خروجی Y0 فعال شده و موتور متصل به خروجی Y0 برای اولین بار به اندازه 100 پالس می چرخد و سپس 4 ثانیه متوقف شده و سپس در مراحل بعد به اندازه 800 پالس بچرخد و 4 ثانیه متوقف شود.




LBL
HSC0I
N002
EN
RST
Y0
N003
74. INDIO
D :
Y0
N :
1
N004
SET
M0
N005
93D. HSCTW
S :
800
CN:
HSC0
D :
PV
N006
93D. HSCTW
S :
0
CN:
HSC0
D :
CV
N007
69
RTI
N008
N009
Main_unit1
Sub_unit1
U:55 F:20168 S:N (Doc U:0 F:8191)

بنام خداوند بخشنده ومهربان

عنوان مدرک :	برنامه نویسی PLC FATEK
توضیحات :	FUN140 &141 و تولید پالس برای سیستمهای سرو
تعداد صفحه :	10
شماره ویرایش :	1
ویرایش کننده :	ا.رضایی
تاریخ ویرایش :	1391.6.18

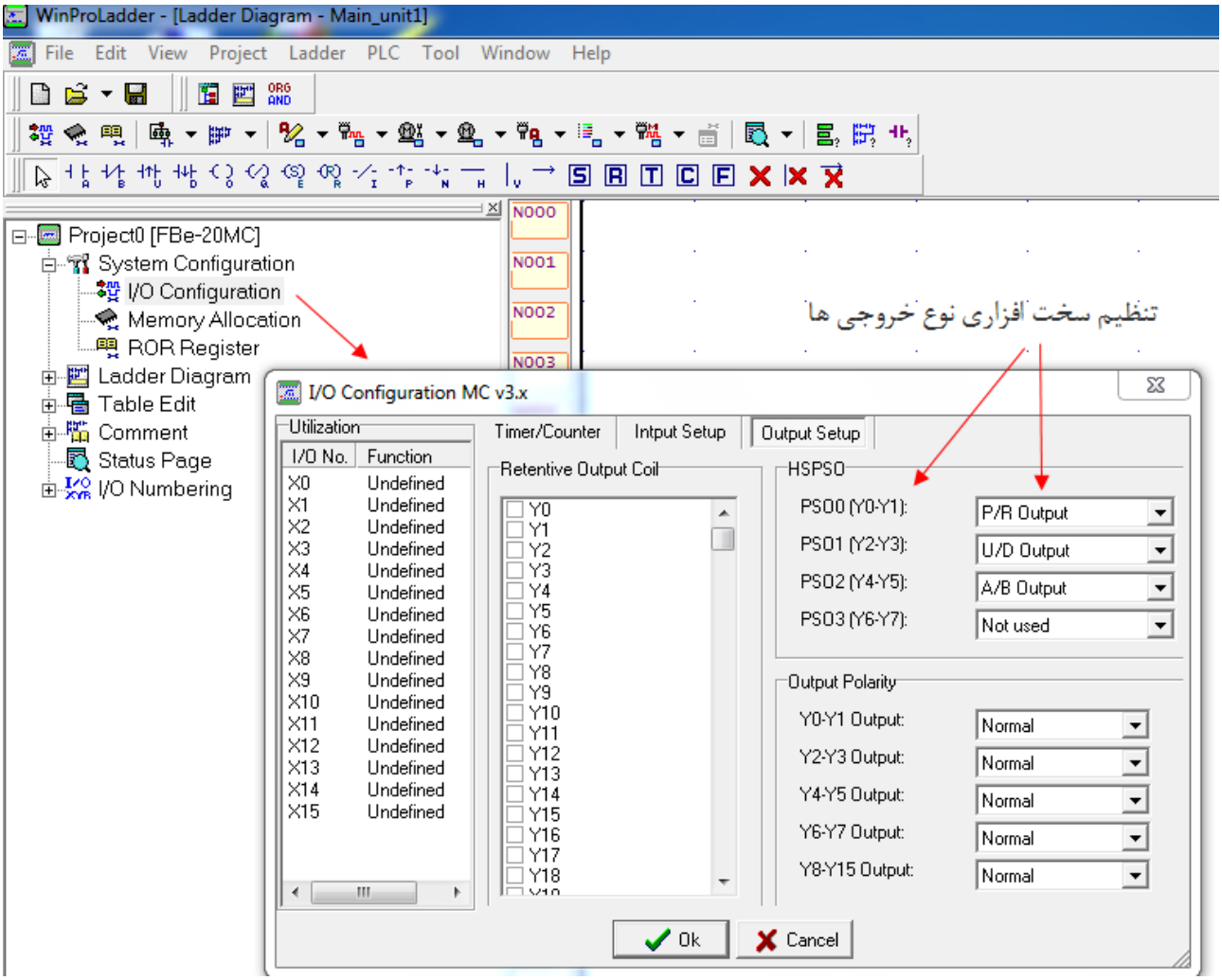
- **جدول زیر حالات مختلف خروجی ها را نشان می دهد**

Axis No.	Exterior output	Output modes			Remark
		U/D output	K/R output	A/B output	
PSO0	Y0 , Y1	Y0=U , Y1=D	Y0=K , Y1=R	Y0=A , Y1=B	Valid for all FBx-xxMCT main unit
PSO1	Y2 , Y3	Y2=U , Y3=D	Y2=K , Y3=R	Y2=A , Y3=B	Not for FBE-20MCT & FBN-19MCT.
PSO2	Y4 , Y5	Y4=U , Y5=D	Y4=K , Y5=R	Y4=A , Y5=B	Only for FBE-40MCT & FBN-36MCT.
PSO3	Y6 , Y7	Y6=U , Y7=D	Y6=K , Y7=R	Y6=A , Y7=B	

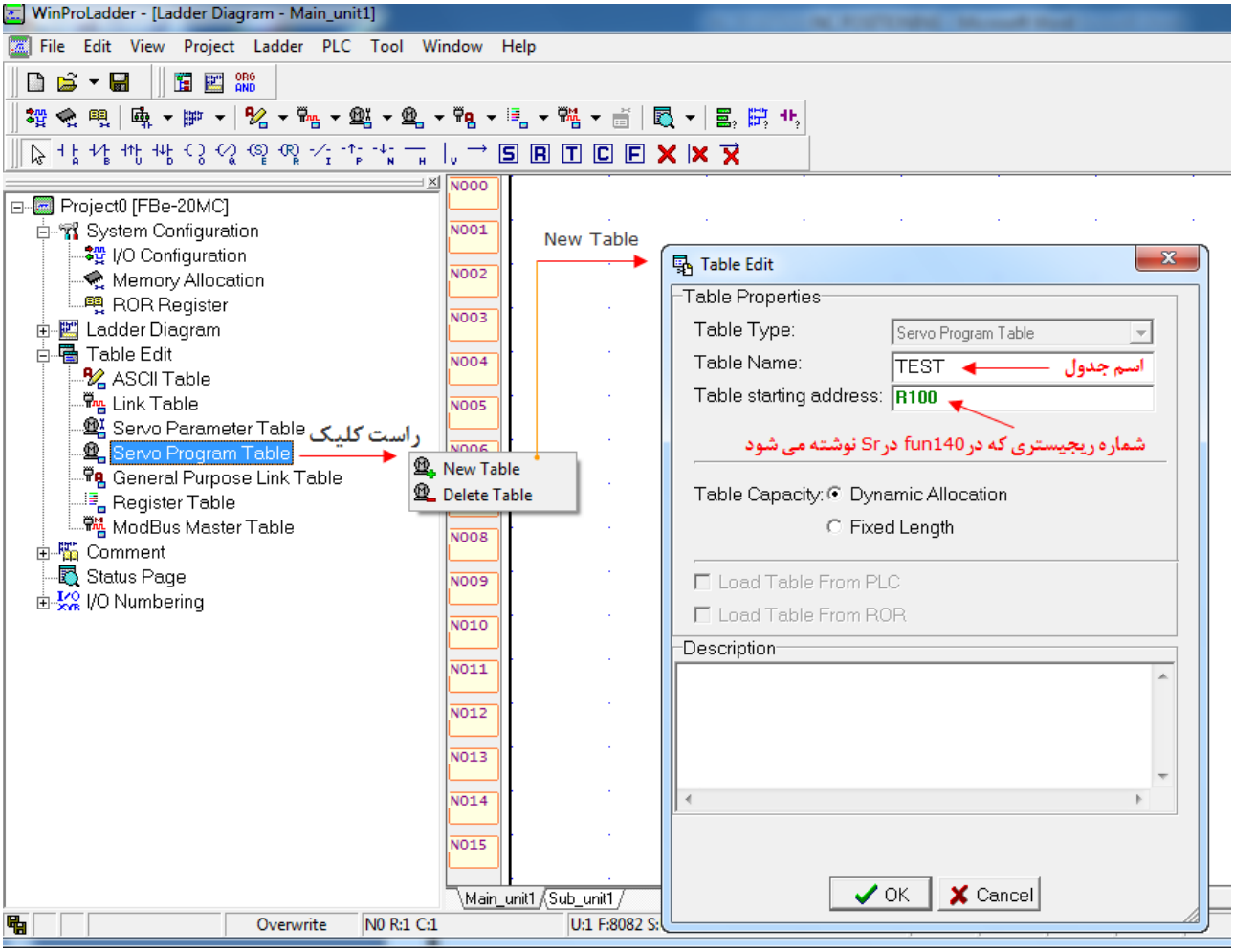
U/D	**پالس بالا رونده در خروجی ظاهر میشود(Y0(Y2,Y4,Y6** **پالس پایین رونده در خروجی ظاهر میشود(Y1(Y3,Y5,Y7**
P/R	**پالس درخروجی ظاهر میشود(Y0(Y2,Y4,Y6** **جهت چرخش را تعیین می نماید(Y1(Y3,Y5,Y7** **ON : شمارش بالا رونده** **OFF : شمارش پایین رونده**
A/B	**پالس با فازA در خروجی ظاهر میشود(Y0(Y2,Y4,Y6** **پالس با فازB در خروجی ظاهر میشود(Y1(Y3,Y5,Y7**

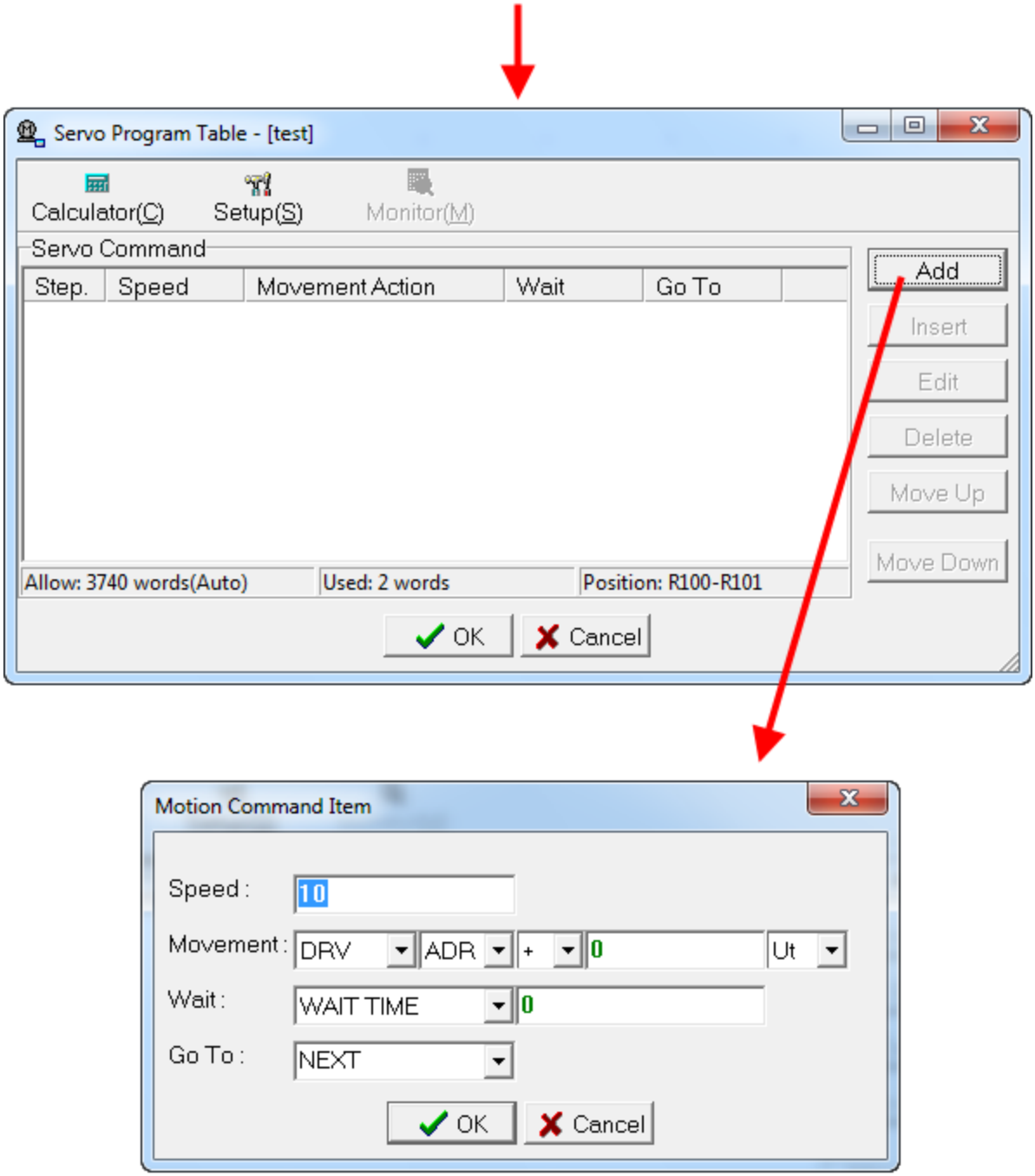
- نحوه تنظیم نمودن خروجیها توسط Winproladder

با انتخاب Output Setup از منوی I/O Configuratuin میتوانید این کار را انجام دهید



- قبل از استفاده از FUN 140 لازم است ابتدا "Servo Program Table" را برنامه ریزی نمایید،این جدول به منظور تعیین پارامترهای مورد نیاز خروجی پالس (ازقبیل : سرعت،فرکانس،وقفه های کاری،جهت چرخش و...)تعبیه گردیده است و FUN 140 پس از فعال شدن پارامترهای ذکر شده را از این جدول می خواند.




Servo Program Table - [test]
Calculator(C)
Setup(S)
Monitor(M)
Servo Command
Step.
Speed
Movement Action
Wait
Go To
Add
Insert
Edit
Delete
Move Up
Move Down
Allow: 3740 words(Auto)
Used: 2 words
Position: R100-R101
OK
Cancel
Motion Command Item
Speed :
10
Movement :
DRV
ADR
+
0
Ut
Wait :
WAIT TIME
0
Go To :
NEXT
OK
Cancel

شرح پارامترهای تنظیمی:

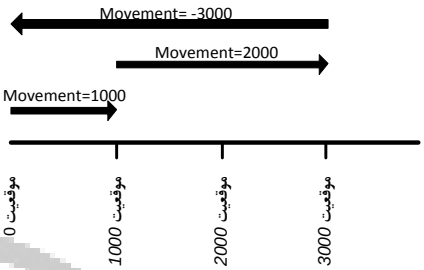
(رجیسترهایی که برای این جدول استفاده می شوند بصورت 32 بیتی می باشند)

پارامتر	شرح
Speed	**فرکانس پالس خروجی**
DRVC/DRV	**** DRVC به منظور اعمال تغییرات سرعت بصورت ترتیبی به کار میرود(8 تغییر سرعت در بیشترین حالت)** **** به منظور تغییر سرعت بصورت ترتیبی ، تنها اولین دستور DRVC میتواند مقدار معین (Absolute) را جهت هماهنگی تعیین مکان بکار برد** **** جهت چرخش در DRVC تنها توسط + و- تعیین میشود** **** جهت چرخش تنها توسط اولین دستور DRVC مشخص میگردد و سایر تغییرات ترتیبی از این جهت پیروی میکنند** **** آخرین دستور در این گزینه به منظور تغببرات ترتیبی باید DRV باشد** **مثال :** 001 SPD 10000 * Pulse frequency = 10KHz. DRVC ADR , + , 20000 , Ut * Forward 20000 units. GOTO NEXT 002 SPD 50000 * Pulse frequency =50 KHz DRVC ADR , + , 60000 , Ut * Forward 60000 units. GOTO NEXT 003 SPD 3000 * Pulse frequency = 3KHz. DRV ADR , + , 5000 , Ut * Forward 5000 units. WAIT X0 * Wait until X0 ON to restart from GOTO 1 the first step to execute.
ADR/ABS	**ADR/ABS :نحوه جابجایی را تعیین میکند** **ADR : جابجایی نسبی(در این گزینه میتوانید جهت چپگرد و یا راستگرد بودن خروجی از حالات + و – استفاده نمایید)** **ABS : جابجایی معین(در این گزینه چپگرد ویا راستگرد بودن خروجی باید اعداد + و – استفاده نمایید بصورت مثال 1500+ ویا 1500-)**

- **شرح تفاوت میان تعیین مکان نسبی (ADR) و تعیین مکان معین (ABS) :**

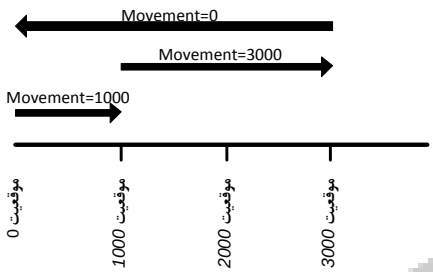
تعریف مد ADR

وقتی بخواهیم تا PLC در هرموقعیتی قرار گیرد باید در رجیستر Movement مقدار حرکت را تعریف کنیم



تعریف مد ABS

وقتی بخواهیم تا PLC در هرموقعیتی قرار گیرد کافیست در رجیستر Movement موقعیت مورد نظر را قرار دهیم



برای تعریف نقطه صفر (رفرنس گرفتن محور) باید عدد صفر را در رجیسترهای رفرنس هر محور انتقال داد

R4088 : رجیستر رفرس محور ۱
R4090 : رجیستر رفرس محور ۲
R4092 : رجیستر رفرس محور ۳
R4094 : رجیستر رفرس محور ۴

+ : حرکت به سمت جلو ویا در جهت عقربه های ساعت **- : حرکت به سمت عقب و یا در خلاف جهت عقربه های ساعت** **‘ ’ : جهت حرکت توسط مقادیر تعیین میشود(مقادیر مثبت : جلو ،مقادیر منفی : عقب)**	+ / - / ‘ ’:
این گزینه میتواند از طریق وارد نمودن مقادیر ثابت و یا از طریق رجیسترهای R و D انجام پذیرد(این عمل 2 رجیستر را اشغال مینماید، بطور مثال در صورت انتخاب R0 ، R1 نیز اشغال خواهد شد) **** در صورتیکه مقدار تعیین شده صفر باشد و حالت ADR را انتخاب نموده باشید بدین معناست که حرکت تا بینهایت پالس ادامه میابد** **رنج قابل انتخاب این گزینه:** **99999999 > میزان حرکت >-99999999**	movement
Ut/Ps : **واحد و یا دقت خروجی را تعیین می نماید** Ut : **دقت این گزینه 1 واحد است که توسط پارامترهای FUN140، 0~3 یعنی دقت خروجی بصورت mm,Deg یا Inch خواهد بود** Ps : **دقت خروجی بصورت 1 پالس خواهد بود (پیشنهاد میشود از این گزینه جهت دقت خروجی استفاده گردد)**	Ut/Ps
هنگامیکه پالس خروجی تعیین شده به اتمام میرسد زمان توقف برای رفتن به خط بعدی برنامه	WAIT

فعال میشود این توقف 5 حالت دارد: 1. زمان توقف که میتواند مقدار ثابتی باشد و یا توسط رجیستر تعیین گردد 2. توسط X0~X255 منتظر می ماند تا ورودی تعیین شده به حالت ON برود 3. توسط Y0~Y255 منتظر می ماند تا خروجی تعیین شده به حالت ON برود 4. توسط M0~M1911 منتظر میماند تا رله کمکی تعیین شده به حالت ON برود 5. توسط S0~S999 منتظر میماند تا رله کمکی تعیین شده به حالت ON برود	
پس از انجام شدن پالسها در خروجی این گزینه فعال شده تا پس اززمان تعیین شده پارامتر GOTO را اجرا نماید	ACT
هنگامیکه خروجی پالس در حال اجرا میباشد، اگر این پارامتر ON شود بلافاصله دستور تعیین شده توسط GOTO اجرا خواهد شد بدون اینکه عمل خروجی در خروجی به پایان رسیده باشد	EXT
بعد از انجام اعمال EXT,ACT,WAIT این گزینه فعال میگردد جهت مشخص نمودن اجرای دستوربعدی NEXT : دستور خط بعدی اجرا خواهد شد 1~N : شماره خط تعیین شده اجرا خواهد شد R/DXXXX : شماره خط دستور بعدی در این رجیستر میتواند ذخیره شود	GOTO

- نحوه عملکرد FUN140

Ps : خروجی پالس تعیین شده

0 : Y0,Y1

1 : Y2,Y3

2 : Y4,Y5

3 : Y6,Y7

SR : رجیستر تعیین شده در جدول برنامه ریزی سروو (Servo Program Table)

WR : رجیستر مربوط به حالات مختلف کارکرد FUN 140 که 7 رجیستر را اشغال می نماید (این ریجیسترها برای کارکرد این تابع هستند و در جای دیگر نباید از آنها استفاده کرد)

- شرح عملکرد

** هنگام فعال شدن EN=1 در صورت فعال نبودن FUN 140 دیگری برروی خروجی مورد نظر (Ps) ،دستورات تعیین شده در جدول پارامترهای سروو خط به خط اجرا می شود.

** در صورت غیر فعال شدن EN=0 بلافاصله پالس خروجی قطع خواهد شد

** در صورت EN=1 و PAU=1 خروجی پالس وتوقف خواهد شد و پس از اینکه PAU=0 خروجی پالس با توجه به جدول پارامترهای سروو ادامه کار خواهد داد

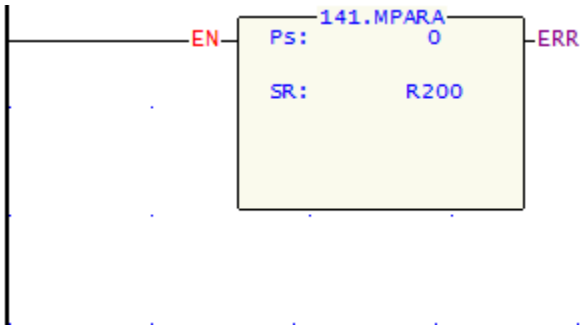
** در صورت ABT=1 خروجی پالس بلافاصله قطع شده و هنگامیکه EN=1 گردد ،دستورات جدول پارامترهای سروو از اولین خط شروع به کار می نماید

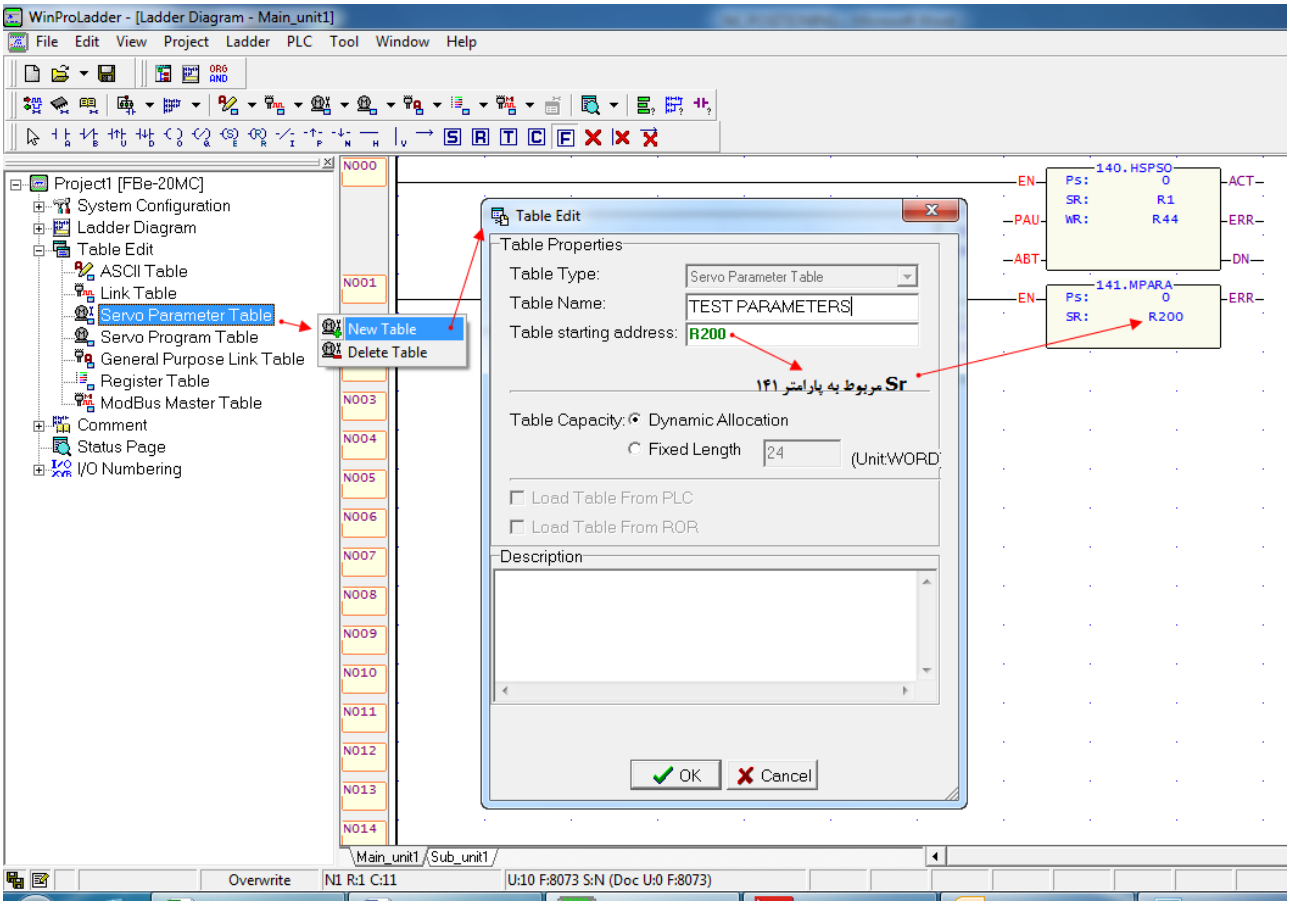
- حالات کاری خروجی پالس
  - تغییرهمزمان مقدار فرکانس خروجی (تغییر فرکانس پالس) در حین اجرای FUN 140

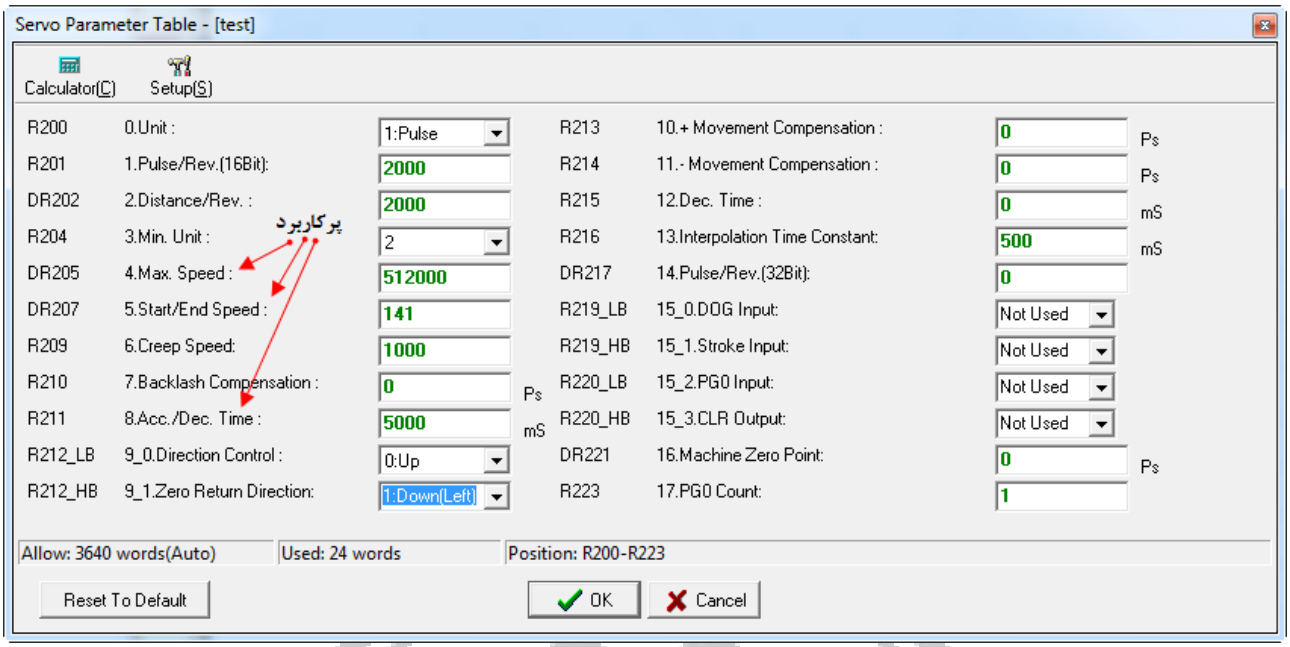
جهت انجام این کار کافیست عدد 90 را در بایت کم ارزش(Low Byte) R4056 کپی کنید تا بتوانید هنگامیکه خروجی پالس مقادیر را از جدول SERVO Program اجرا می نماید ، فرکانس پالس خروجی را بصورت همزمان تغییر دهید.

مقدار پیش تنظیم این رجیستر صفر می باشد.

FUN141 فانکشن فعال سازی جدول پارامترهای حرکت سرو :

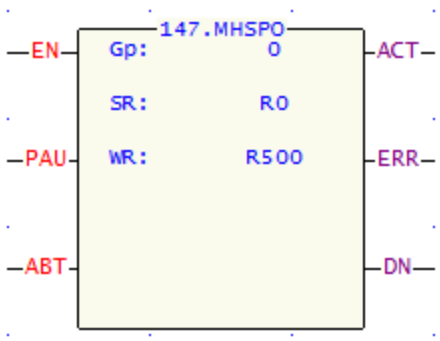





Servo Parameter Table - [test]
Calculator(C)
Setup(S)
پرکاربرد
R200 0.Unit : 1:Pulse
R201 1.Pulse/Rev.(16Bit): 2000
DR202 2.Distance/Rev. : 2000
R204 3.Min. Unit : 2
DR205 4.Max. Speed : 512000
DR207 5.Start/End Speed : 141
R209 6.Creep Speed: 1000
R210 7.Backlash Compensation : 0 Ps
R211 8.Acc./Dec. Time : 5000 mS
R212_LB 9_0.Direction Control : 0:Up
R212_HB 9_1.Zero Return Direction: 1:Down(Left)
R213 10.+ Movement Compensation : 0 Ps
R214 11.- Movement Compensation : 0 Ps
R215 12.Dec. Time : 0 mS
R216 13.Interpolation Time Constant: 500 mS
DR217 14.Pulse/Rev.(32Bit): 0
R219_LB 15_0.DOG Input: Not Used
R219_HB 15_1.Stroke Input: Not Used
R220_LB 15_2.PG0 Input: Not Used
R220_HB 15_3.CLR Output: Not Used
DR221 16.Machine Zero Point: 0 Ps
R223 17.PG0 Count: 1
Allow: 3640 words(Auto)
Used: 24 words
Position: R200-R223
Reset To Default
OK
Cancel

بنام خداوند بخشنده ومهربان

عنوان مدرک :	برنامه نویسی PLC FATEK
توضیحات :	FUN147 تولید پالس برای حرکت چند محور
تعداد صفحه :	12
شماره ویرایش :	0
ویرایش کننده :	ا.رضایی
تاریخ ویرایش :	1391.6.27



عملکرد این تابع به گونه ای است که اگر چند سرو موتور با حرکتهای متفاوت داشته باشیم , سرعت سروموتورها را به گونه ایی تنظیم می نماید که سروموتورها در یک لحظه به حرکت در آمده و در یک لحظه متوقف شوند.

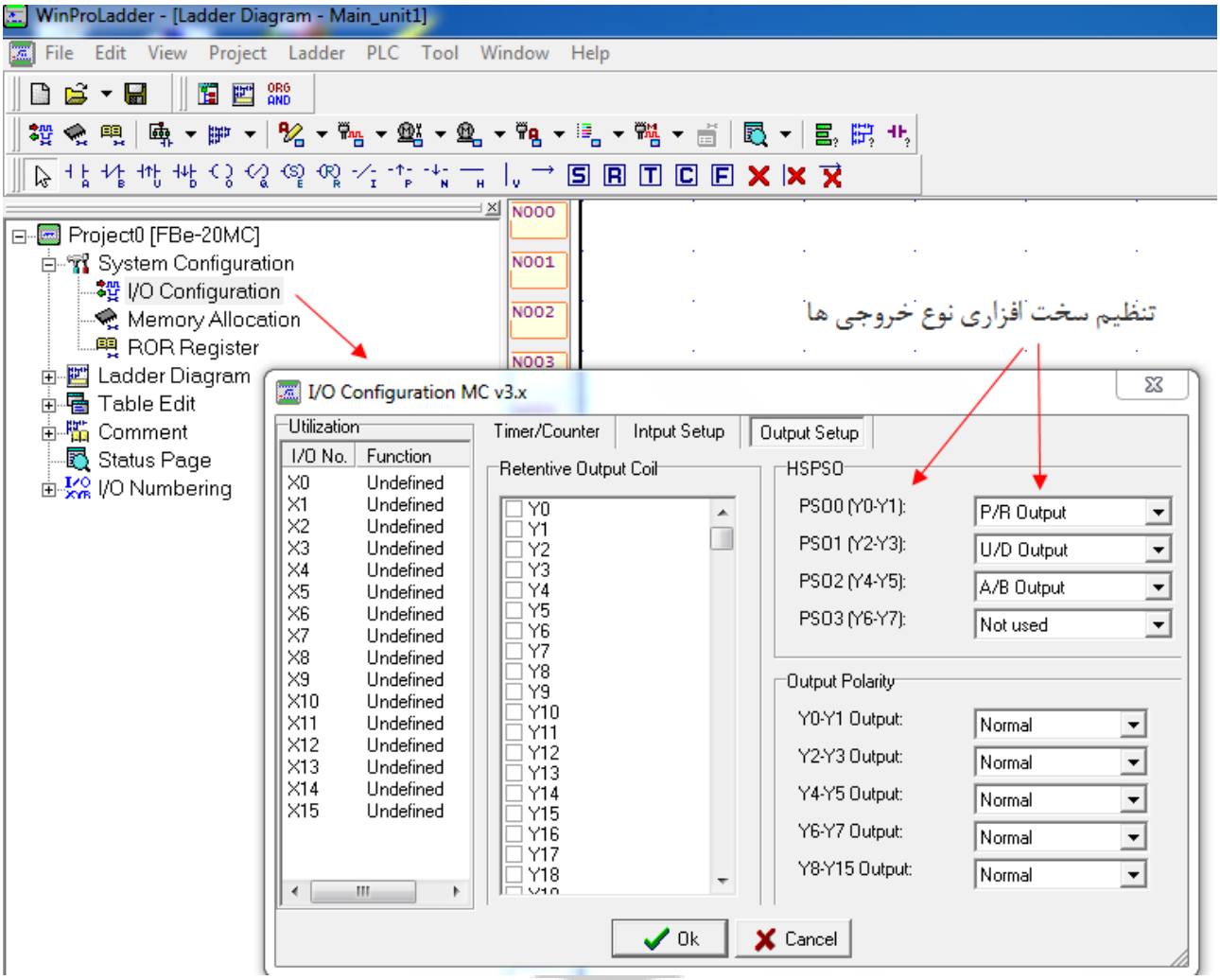
هرنقطه از مکان را یک استپ (step) می نامیم. هر استپ 15 رجیستر برای کد کردن در اختیار دارد.

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

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

- نحوه تنظیم نمودن خروجیها توسط Winproladder

با انتخاب Output Setup از منوی I/O Configuratuin میتوانید این کار را انجام دهید



ارتباطات برای کنترل موقعیت

M1991	ON : پالس را کند کرده سپس قطع می کند. OFF : فوراً پالس را قطع می کند.
M1992	ON: Ps0 آماده است. Ps0 :OFF فعال است.
M1993	Ps1 :ON آماده است. Ps1 :OFF فعال است.
M1994	Ps2 :ON آماده است. Ps2 :OFF فعال است.
M1995	Ps3 :ON آماده است. Ps3:OFF فعال است.
M1934	Gp0 :ON آخرین step را تمام کرد.
M1935	Gp1 :ON آخرین step را تمام کرد.
M2000	چند محور همزمان عمل می کنند.
DR4068	سرعت بردار Gp0
DR4070	سرعت بردار Gp1
D4060	Gp0 error code
D4061	Gp1 error code
D4062	شماره استپی که به Gp0 مربوط است (شماره استپی که کامل شده است)
D4063	شماره استپی که به Gp1 مربوط است (شماره استپی که کامل شده است)

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

در SR ، رجیستر مربوط به شماره شروع جدول مربوطه می باشد.

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

WR+0	**استپ اجرا شده یا متوقف شده**
WR+1	Working flag
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	**توسط سیستم کنترل می شود**

WR+0 : هنگام اجرای این تابع ، محتویات این رجیستر ،استپی را که اجرا می شود (شماره سطر جدول مربوطه) نمایش می دهد(N~1) .اگر تابع در حال اجرا نباشد ،محتویات این رجیستر ،استپی را که در آن جا متوقف شده است نشان می دهد.وقتی EN فعال شود ،استپ بعدی اجرا می شود.(اگر استپ جاری ،آخرین استپ باشد،اجرا از اولین استپ شروع می شود)

WR+1 : B0~B7 (بیت 0 ~بیت7) مجموع استپ ها

B8=ON ، خروجی نگه داشته شده است (paused)

B9=ON ، منتظر شرایط انتقال

B10=0N ،خروجی بی انتها

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

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.

ریجیسترهای مربوط به خطاهای دستورات پالس	
R4060	Error code of PSO 0
R4061	Error code of PSO 1
R4062	Error code of PSO 2
R4063	Error code of PSO 3

-	توضیحات	کد خطا
کدهای خطایی که ممکن است در هنگام اجرای تابع 141 به وجود بیایند	بدون خطا	0
	ارور پارامتر0	1
	ارور پارامتر1	2
	ارور پارامتر2	3
	ارور پارامتر3	4
	ارور پارامتر4	5
	ارور پارامتر5	6
	ارور پارامتر6	7
	ارور پارامتر7	8
	ارور پارامتر8	9
	ارور پارامتر9	10
	ارور پارامتر10	11
	ارور پارامتر11	12
	ارور پارامتر12	13
	ارور پارامتر13	14
	ارور پارامتر14	15

کد خطا	توضیحات	-
30	Error of variable address for speed setting	کدهای خطایی که ممکن است در هنگام اجرای توابع 140 و 147 به وجود بیایند
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 خطا، آخرین کد خطا را حفظ می کند.

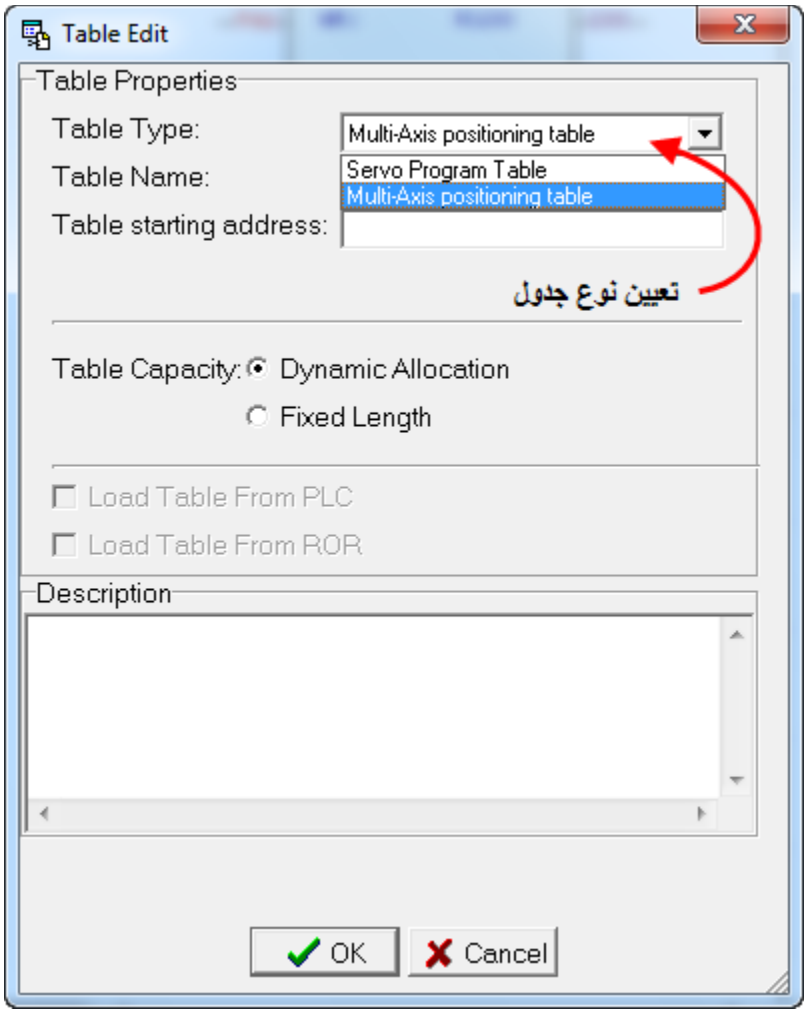
تنظیم جدول برنامه Servo توسط WinProladder

برای استفاده از تابع 147 باید ابتدا جدول مربوط به تولید پالس خروجی ایجاد شود.

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.



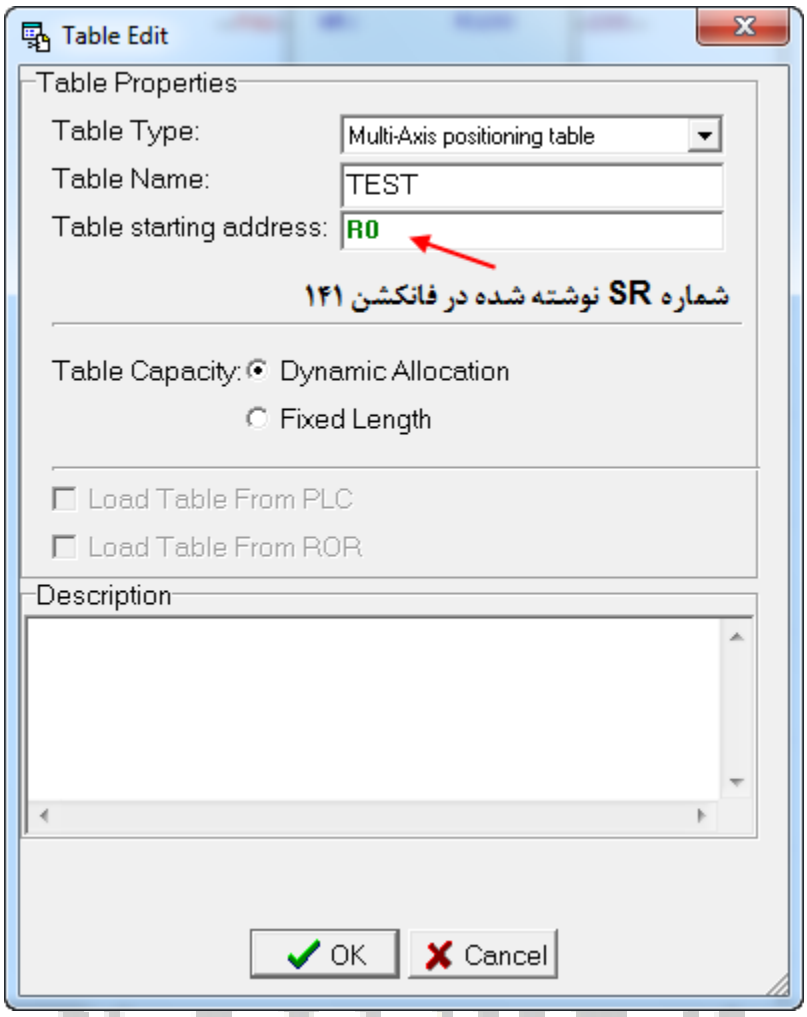
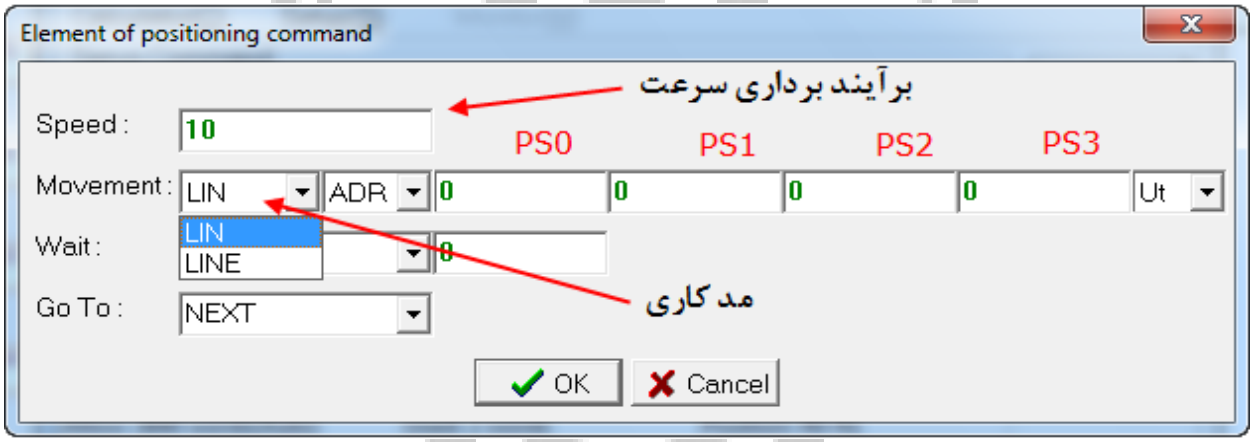
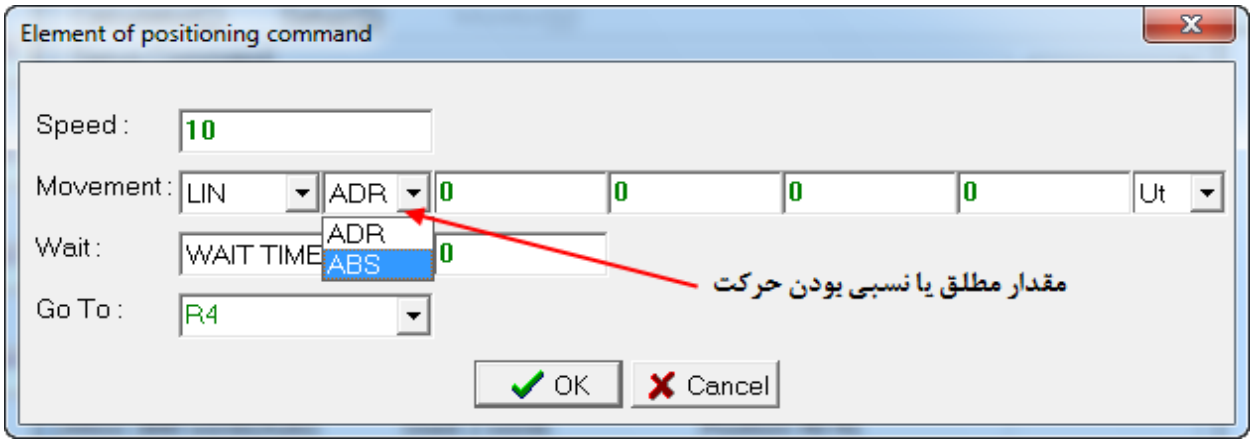
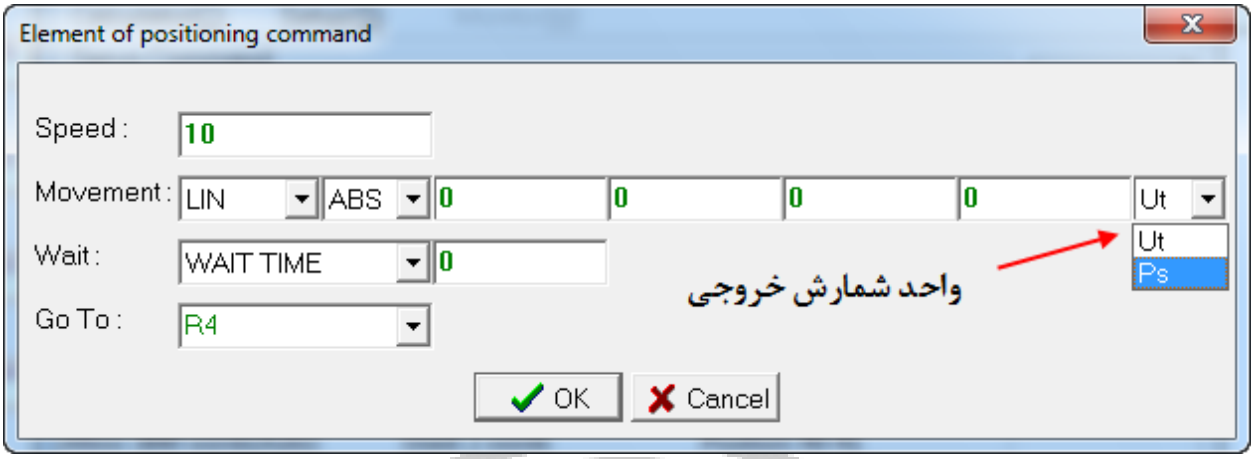


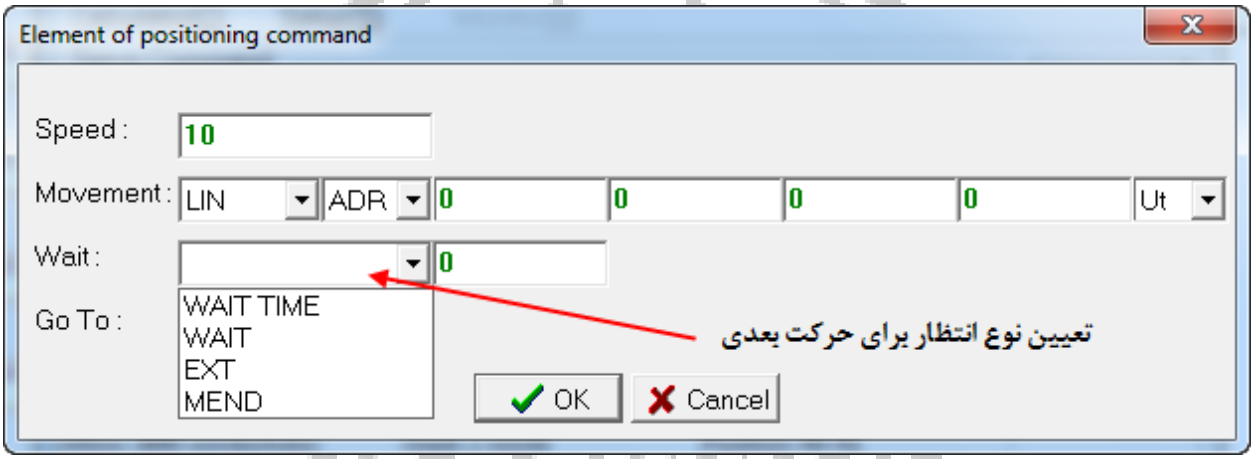
Table Edit
Table Properties
Table Type:
Multi-Axis positioning table
Table Name:
TEST
Table starting address:
R0
شماره SR نوشته شده در فانکشن ۱۴۱
Table Capacity:
Dynamic Allocation
Fixed Length
Load Table From PLC
Load Table From ROR
Description
OK
Cancel

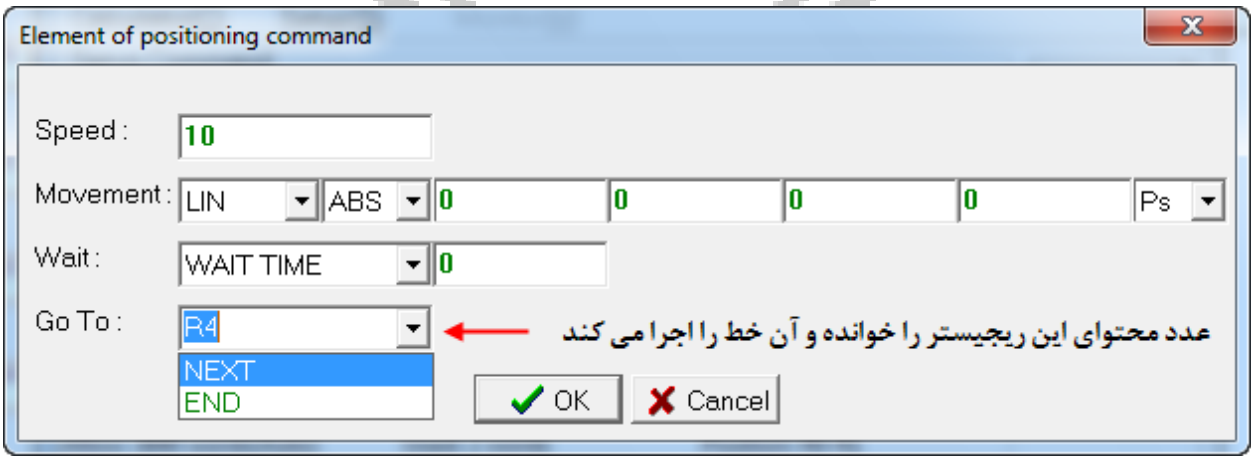

Multi-Axis positioning table - [TEST]
Calculator(C)
Setup(S)
Monitor(M)
Servo Command
Step.
Speed
Movement Action
Wait
Go To
Add
Insert
Edit
Delete
Move Up
Move Down
برای نوشتن برنامه های جدول
Allow: 3840 words(Auto)
Used: 2 words
Position: R0-R1
OK
Cancel


Element of positioning command
برآیند برداری سرعت
Speed :
10
PS0
PS1
PS2
PS3
Movement :
LIN
ADR
0
0
0
0
Ut
LIN
LINE
Wait :
0
مد کاری
Go To :
NEXT
OK
Cancel


Element of positioning command
Speed :
10
Movement :
LIN
ADR
0
0
0
0
Ut
ADR
ABS
Wait :
WAIT TIME
0
مقدار مطلق یا نسبی بودن حرکت
Go To :
R4
OK
Cancel


Element of positioning command
Speed : 10
Movement : LIN ABS 0 0 0 0 Ut
Ut
Ps
Wait : WAIT TIME 0
Go To : R4
واحد شمارش خروجی
OK
Cancel


Element of positioning command
Speed : 10
Movement : LIN ADR 0 0 0 0 Ut
Wait :
Go To :
WAIT TIME
WAIT
EXT
MEND
تعیین نوع انتظار برای حرکت بعدی
OK
Cancel


Element of positioning command
Speed : 10
Movement : LIN ABS 0 0 0 0 Ps
Wait : WAIT TIME 0
Go To : R4
NEXT
END
عدد محتوای این ریجیستر را خوانده و آن خط را اجرا می کند
OK
Cancel

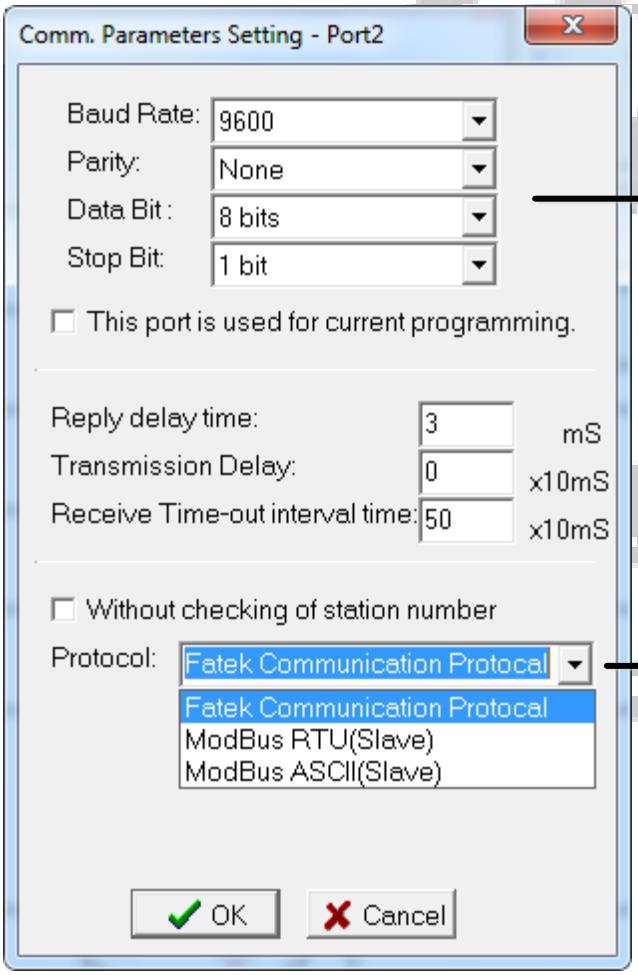
پارامتر	توضیحات
SPEED	سرعت بردار را برای Interpolation خطی تنظیم می کند.(برآیند برداری سرعت محورها) پالس یا m/s بودن را می توان در Servo Parameter Table تعیین نمود (پیش فرض = در حالت فرکانس) عملوند می تواند عدد ثابت یا یک رجیستر باشد که اگر رجیستری باشد به 2 رجیستر، فضا احتیاج دارد.
MOVWMENT	وقتی مختصات یک محور 0 است یا جای خالی (در دستورات متنی)گذاشته شود ،یعنی حرکتی در راستای آن محور نخواهد بود. حداکثر اندازه حرکت باید بین 99999999 ± پالس باشد. عملوند ششم:این عملوند واحد حرکت را مشخص می کند. (Ut/Ps) (می توان در Servo Parameter Table تعیین نمود) Ut : اعداد نوشته شده معادل یک واحد تعیین شده خواهد بود. Ps: اعداد نوشته شده معادل یک پالس خواهد بود. LINE برای interpolation خطی در حرکت بی انتها استفاده می شود. در این دستور، رابطه بین محورها در خروجی، طوری است که بقیه محورها از محوری که بیشترین مختصات را دارد تبعیت خواهند کرد. به عنوان مثال:در حالتی که عملوند ششم Ps است و مختصات محورها Ps0=1000, Ps1=500 ,Ps2=300 Ps3=0 باشد، بدین معنی است که اگر محور Ps0 مقدار 1000 پالس می فرستد، سپس Ps1 و Ps2 به ترتیب 500 و 300 پالس خواهند فرستاد (Ps3 کار نمی کند چون مقدارش 0 است) این دستور این نسبت ها را در دادن پالس به خروجی ادامه می دهد تا زمانی که تابع 147 متوقف شود.
WAIT	پس از اتمام خروجی پالس،این دستور باعث انتظار سیستم برای زمان معین می شود سپس به مرحله ی معین شده می رود. این دستور 5 مدل عملوند دارد: Time: (زمان پایه 0.01 ثانیه)وقتی زمان تعیین شده به پایان رسید،مرحله ای را اجرا می کند که توسط GO TO مشخص شده است. اگر از پارامترهای (X0~X255 S0~S999 M0~M1911 Y0~Y255) تا زمانی صبر می کند که مشخصه مورد نظر ON شود،سپس مرحله ای را اجرا می کند که توسط GO TO مشخص شده است. EXT : اگر مشخصه مورد نظر این دستور (X0~X255 S0~S999 M0~M1911 Y0~Y255) در حین پالس دهی به خروجی، ON شود،فوراً مرحله ای را که توسط GO TO مشخص شده انجام می دهد. اگر تا آخر پالس دهی ON نشود،این دستور مانند WAIT عمل می کند.
GO TO	پس از پایان یافتن دستورات WAIT،ACT و EXT، مرحله ای را اجرا می کند که GOTO به آن اشاره می کند. NEXT :بدین معنی است که مرحله ی بعد اجرا شود. N~1 : مرحله ای را انجام می دهد که LABLE آن عدد مورد نظر باشد. اگر از ریجیسترهای R,D استفاده کنیم مرحله ای اجرا می شود که در این ریجیسترها نوشته شده است.
MEND	پایان

بنام خداوند بخشنده ومهربان

عنوان مدرک :	برنامه نویسی PLC FATEK
توضیحات :	Fun150 Modbus RTU Standard
تعداد صفحه :	7
شماره ویرایش :	2
ویرایش کننده :	ا.رضایی
تاریخ ویرایش :	1393.1.31

از این فانکشن جهت اتصال دو یا چند PLC و یا اتصال PLC به وسایل جانبی از طریق استاندارد MODBUS RTU/ASCII استفاده میشود.

- FUN150 می تواند از طریق RS232, RS485 ارتباط برقرار کند. این فانکشن در برنامه پی ال سی MASTER نوشته می شود.
- Station number و پارامترهای ثابت (Baud rate , Parity , Data bits , Stop bit) در تمام تجهیزاتی که می خواهیم با هم ارتباط دهیم باید با یکدیگر برابر باشند. این فانکشن ارتباط مابین 247 ایستگاه جانبی را میتواند برقرار نماید. پارامترهای مربوط به FUN 150 ودر صورت لزوم تغییر Time-out و Transaction Delay تنظیم شوند. برای تنظیم پورت باید از مسیر PLC>>SETTING در حالت ONLINE می توان پورت را تنظیم کرد.



پارامترهای ارتباطی پورت
این پارامترها باید در تمام تجهیزاتی که می خواهیم به هم ارتباط دهیم ، یکسان باشند

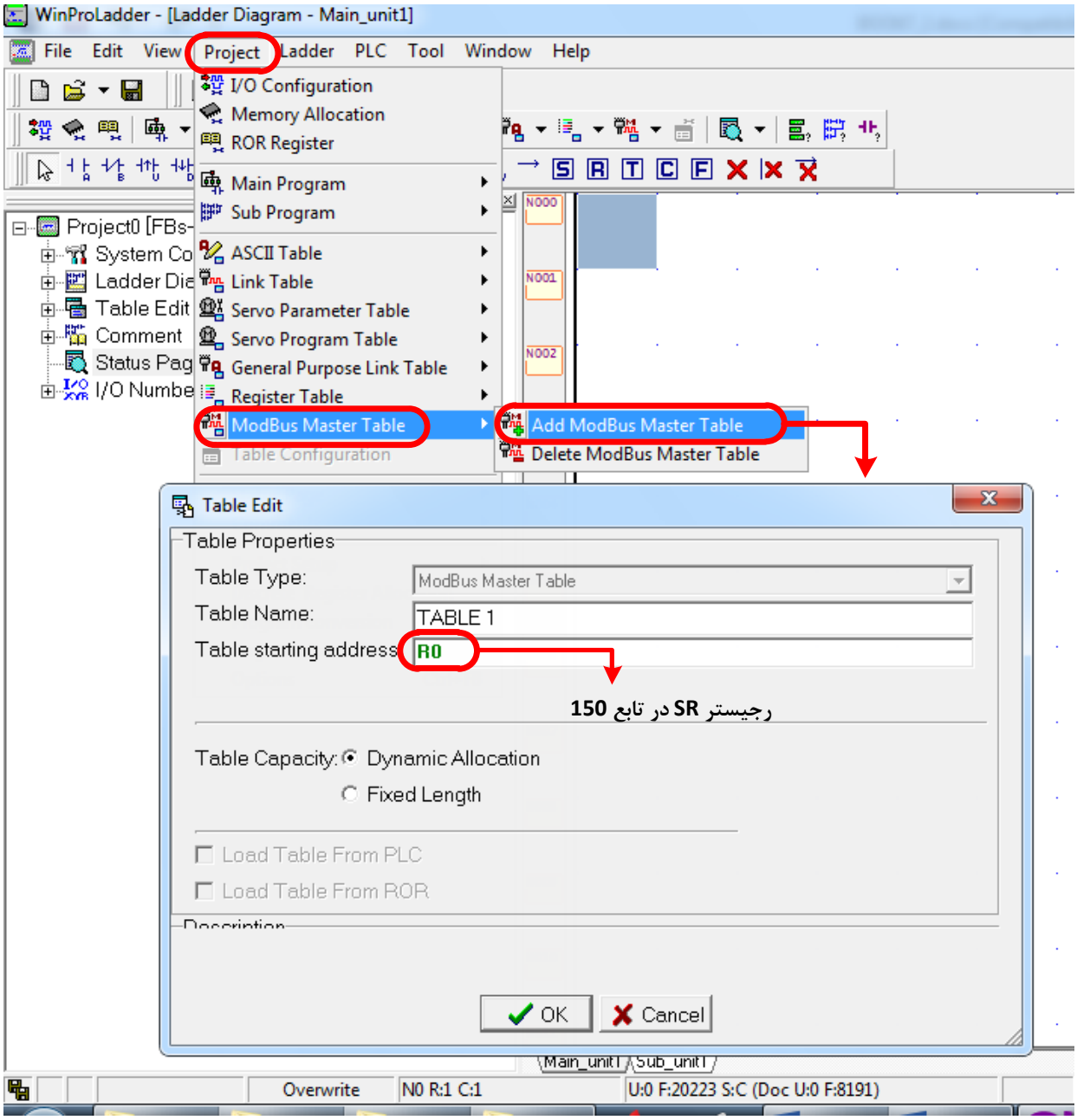
چنانچه این PLC در حالت SLAVE باشد ، باید پروتکل شبکه به CPU اعلام شود.

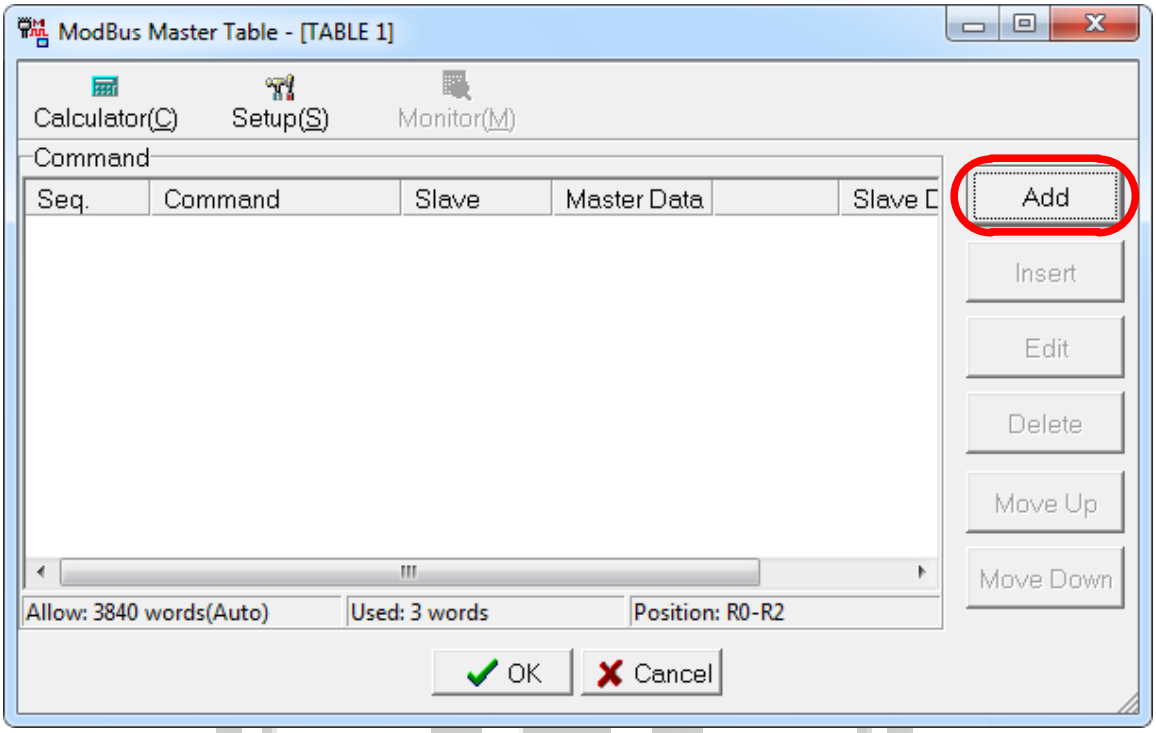
چنانچه PLC در حالت MASTER عمل می کند، این گزینه اهمیتی ندارد.

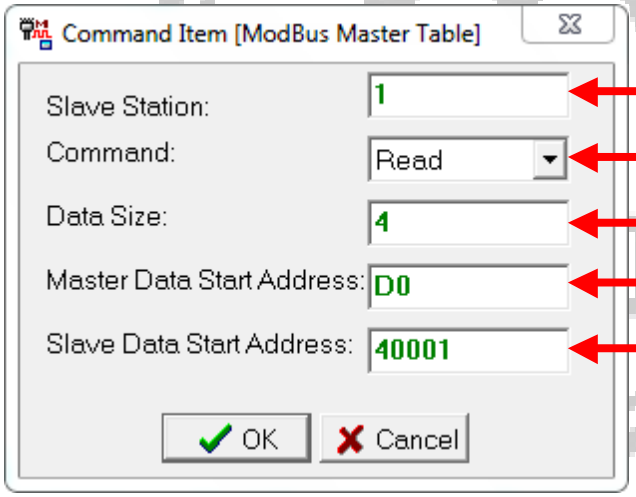
- **بعد از تنظیم پورت عدد مربوط به آن در رجیسترهای خاص هر پورت قرار داده می شود و چنانچه می خواهیم که تنظیمات پورت در پروژه ذخیره شود باید در برنامه با استفاده از تابع 8 (MOVE) این اعداد را در رجیسترها قرار دهیم.**
- **رجیسترهای مربوط به پورتها ارتباطی :**

R4050	رجیستر مربوط به تنظیمات پارامترهای پورت 0	فقط Baud Rate قابل تغییر می باشد
R4146	رجیستر مربوط به تنظیمات پارامترهای پورت 1	تنظیمات : Baud rate , Parity , Data bits , Stop bit
R4158	رجیستر مربوط به تنظیمات پارامترهای پورت 2	تنظیمات : Baud rate , Parity , Data bits , Stop bit
R4161	رجیستر مربوط به تنظیمات پارامترهای پورت 2 (High Speed CPU Link)	● تنظیمات : Baud rate , Parity , Stop bit ● Data bits = 8-bit ● Baud Rate ≥ 38400 bps
R4043	رجیستر مربوط به تنظیمات پارامترهای پورت 3	تنظیمات : Baud rate , Parity , Data bits , Stop bit
R4044	رجیستر مربوط به تنظیمات پارامترهای پورت 4	تنظیمات : Baud rate , Parity , Data bits , Stop bit
R4047	رجیستر مربوط به پروتوکل های ارتباطی پورتهای 1~4	عملکرد PLC در مد SLAVE با پروتوکل FACON یا MODBUS

❖ برای استفاده از تابع 150 ابتدا باید جدول مربوط به آنرا تشکیل داد. جدول مربوط به ارسال و دریافت اطلاعات برروی پی ال سی MASTER تنظیم می شود. نام جدول برای فانکشن 150 و استاندارد مدباس، MODBUS MASTER TABLE می باشد.







شماره **Station number** مربوط به **Slave** مورد نظر

نوع فریم (خواندن یا نوشتن)

تعداد رجیسترها در هر فریم

آدرس شروع رجیستر در **MASTER**

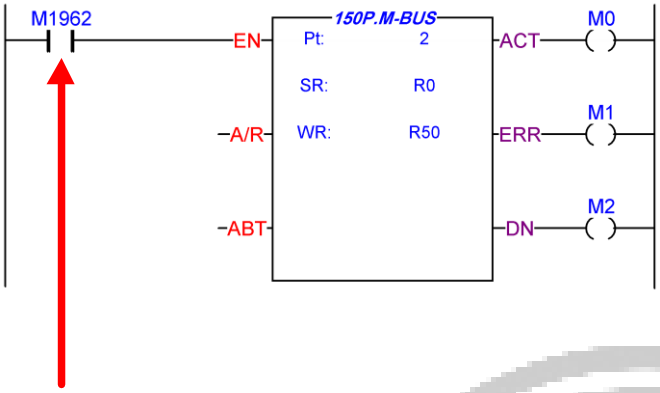
آدرس شروع رجیستر در **SLAVE**

Slave Station	شماره Number Station مربوط به تجهیز Slave
Command	فرمان خواندن یا نوشتن از پی ال سی Master به تجهیز Slave می باشد. (فرمان Single Write ، برای خواندن فقط **1** رجیستر از Slave می باشد)
Data Lengh	تعداد رجیسترهایی که در هر فریم فرستاده یا دریافت می شوند .
Master Data Start Address	رجیسترهای اختصاص یافته برای ارسال و یا دریافت اطلاعات توسط MASTER PLC
Slave Data Start Address	رجیسترهایی برای ارسال و یا دریافت اطلاعات توسط پی ال سی Slave

❖ **آدرس رجیسترهای FATEK PLC در پروتوکل مدباس در جدول زیر مشاهده می نمایید.**

Modbus (5-code)	Modbus (6-code)	Facon	Description
00001～00256	000001～000256	Y0～Y255	(Discrete Output)
01001～01256	001001～001256	X0～X255	(Discrete Input)
02001～04002	002001～004002	M0～M2001	(Discrete M Relay)
06001～07000	006001～007000	S0～S999	(Discrete S Relay)
09001～09256	009001～009256	T0～T255	(Status of T0～T255)
09501～09756	009501～009756	C0～C255	(Status of C0～C255)
40001～44168	400001～404168	R0～R4167	(Holding Register)
45001～45999	405001～405999	R5000～R5998	(Holding Register or ROR)
46001～48999	406001～408999	D0～D2998	(Data Register)
49001～49256	409001～409256	T0～T255	(Current Value of T0～T255)
49501～49700	409501～409700	C0～C199	(Current Value of C0～C199, 16-bit)
49701～49812	409701～409812	C200～C255	(Current Value of C200～C255, 32-bit)

❖ شرح تنظیمات FUN 150



ACT : در حال فرستادن فریم
ERR : خطا در دریافت یا ارسال
DN : موفقیت در ارسال یا دریافت بدون خطا

Pt : شماره پورت مربوطه
SR : آدرس رجیستر شروع جدول
WR : تابع ۱۵۰ برای اجرا، نیاز به تعدادی رجیستر دارد

M1921 : هر ثانیه ۱۰ فریم می فرستد
M1922 : هر ثانیه ۱ فریم می فرستد

M1960 : برای پورت ۱
M1962 : برای پورت ۲
M1936 : برای پورت ۳
M1938 : برای پورت ۴

} بعد از خالی شدن پورت، بلافاصله فریم فرستاده می شود.

Pt : شماره پورت ارتباطی PLC با سایر تجهیزات را مشخص می نماید ،1~4

SR : رجیستر جدول (TABLE STARTING ADDRESS)

WR : رجیستر مربوط به عملکرد FUN 150

"EN"این فانکشن فقط با لبه فعال می شود یعنی با هر لبه یکبار اطلاعات را برروی پورت می فرستد بنابراین ورودی EN این فانکشن را باید با کنتاکت M1921 یا M1922 یا M1962 و ... سری کرد . در هنگام تست و شروع کار بهتر است ابتدا از M1922 استفاده کرده تا بیتها ACT, ERR, DN قابل بررسی باشند و بعد از اطمینان از ارتباط می توان سرعت تبادل اطلاعات را بالاتر برد.

اگر A/R=0 باشد FUN 150 بوسیله استاندارد Modbus RTU ارتباط برقرار مینماید و اگر A/R=1 باشد استاندارد ارتباطی Modbus ASCII خواهد بود.

استاندارد ارتباطی Modbus ASCII از OS Version 4.12 به بعد قابل اجرا می باشد

بنام خداوند بخشنده ومهربان

عنوان مدرک :	برنامه نویسی PLC FATEK
توضیحات :	FUN151 FATEK CPU LINK NETWORK
تعداد صفحه :	6
شماره ویرایش :	2
ویرایش کننده :	ا.رضایی
تاریخ ویرایش :	1393.2.6

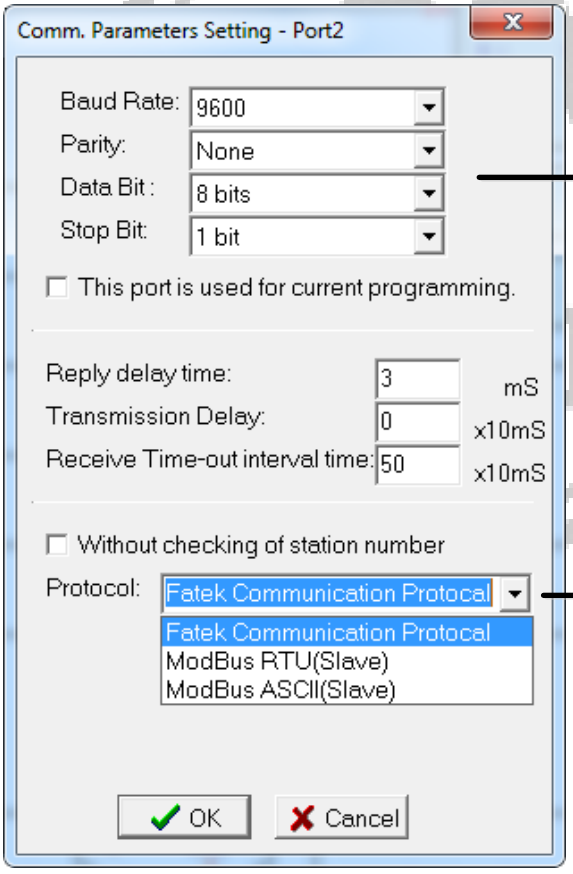
از این فانکشن در چند حالت می توان استفاده کرد:

1- جهت اتصال دو یا چند PLC و یا اتصال PLC به وسایل جانبی که استاندارد FATEK (FACON) را پشتیبانی می کنند

2- خواندن فریم های سریال از روی پورت

3- ارسال فریم سریال به پورت

- FUN151 در بستردرگاههای ETHERNET , RS485 , RS232 ارتباط برقرار می کند
- پارامترهای ثابت (Baud rate , Parity , Data bit , Stop bit) در تمام تجهیزاتی که می خواهیم با هم ارتباط دهیم باید با یکدیگر برابر باشند.
- پارامترهای Time-out و Transaction Delay در صورت لزوم باید تنظیم شوند.



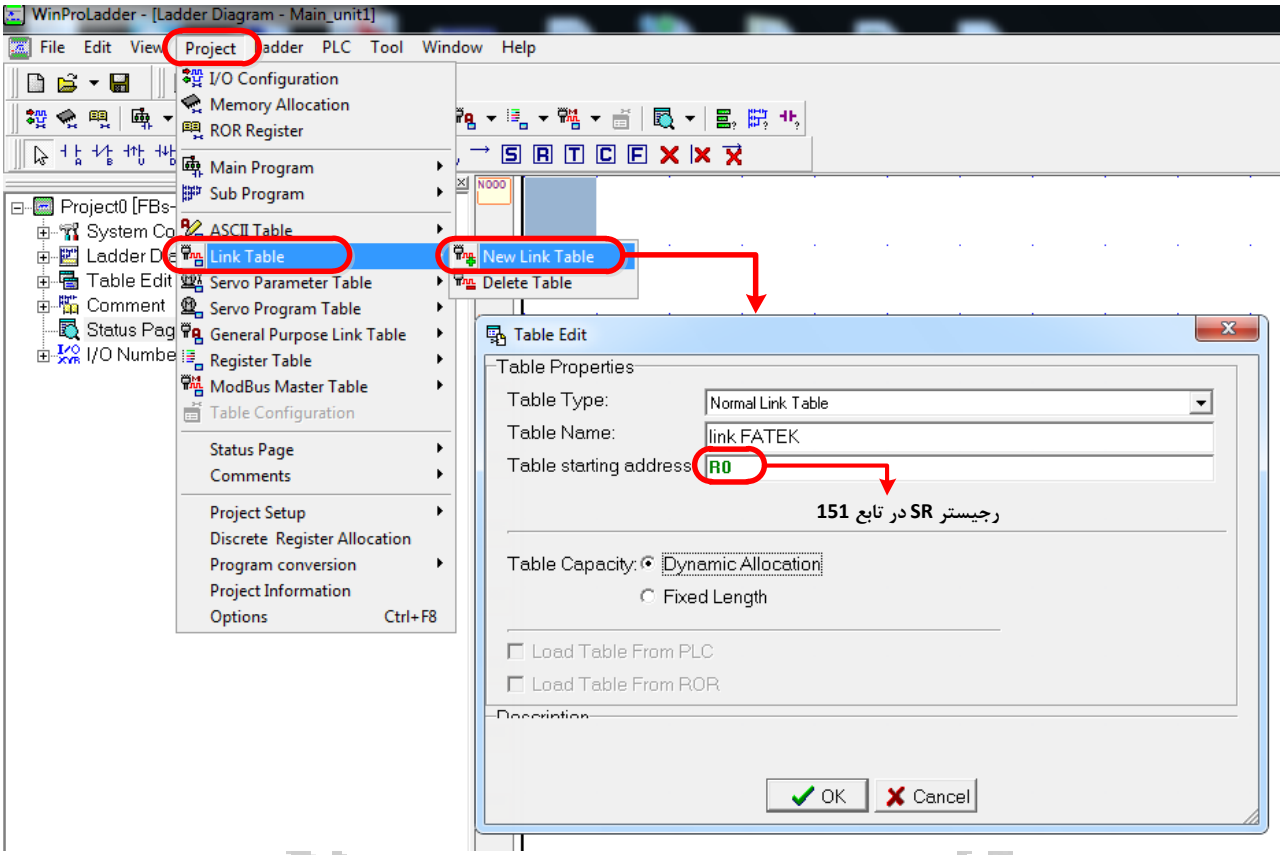
پارامترهای ارتباطی پورت
این پارامترها باید در تمام تجهیزاتی که می خواهیم به هم ارتباط دهیم ، یکسان باشند

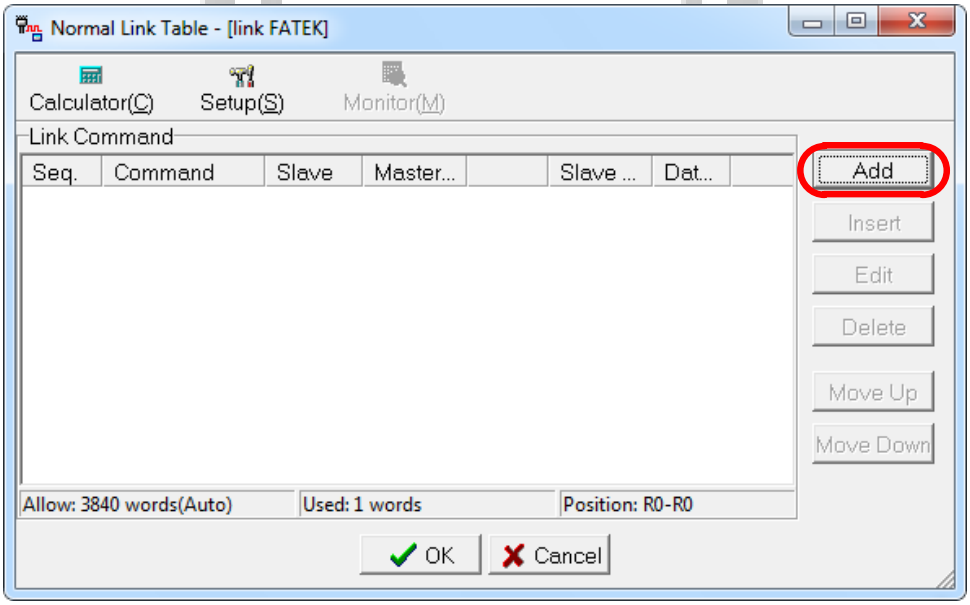
چنانچه این **PLC** در حالت **SLAVE** باشد ، باید پروتکل شبکه به **CPU** اعلام شود.

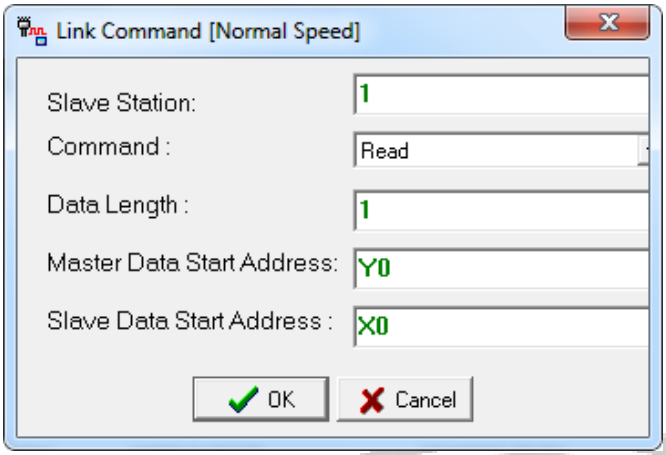
چنانچه **PLC** در حالت **MASTER** عمل می کند، این گزینه اهمیتی ندارد.

- این فانکشن و جدول مربوطه در برنامه پی ال سی MASTER نوشته می شوند .

جدول LINK TABLE برای فانکشن 151 و استاندارد FATEK می باشد.







از **Slave** شماره ۱ ، **X0** را خوانده و در **Y0** مربوط به **MASTER** قرار می دهد

- شرح عملکرد FUN 151

Ladder symbol

Pt : شماره پورت برای ارتباط

MD : مد کاری ، این تابع در چند حالت می تواند کار کند که عبارتند از :

1- **MD0**: برای ارتباط بین تجهیزات با پروتوکل **FACON**

2- **MD1**: ارسال اطلاعات سریال به پورت

3- **MD2**: دریافت و سپس ارسال اطلاعات سریال از پورت

4- **MD3**: ارتباط سرعت بالا بین تجهیزات **FATEK**

SR : در مدهای **0** و **3** ، رجیستری را که در جدول ارتباطات (LINK Table) تعیین شده ، در اینجا باید نوشته شود .

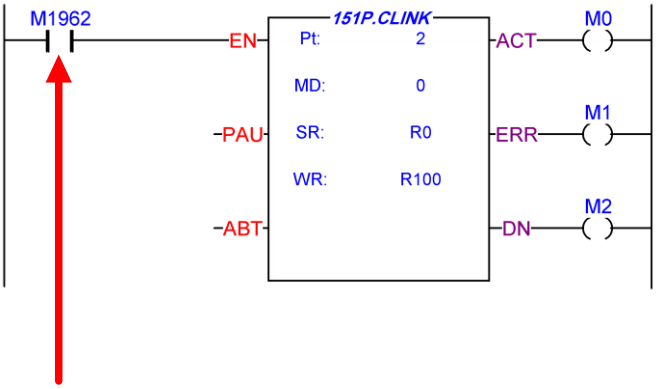
WR : رجیستر مربوط به عملکرد FUN 151 که در مدهای **0** و **3** ، 8 رجیستر را اشغال خواهد نمود .

در تابع 151 در مدهای 1 و 2 بدلیل اینکه پروتوکل خاصی ندارند و فریم باید توسط برنامه نویس تشکیل شود جدولی به SR نمی توان اختصاص داد و باید در رجیسترهای SR و WR مقادیر مورد نیاز را وارد کرد.

نحوه مقداردهی به SR و WR در مدهای 1 و 2 :

SR+0	=00H	دریافت فریم هایی که دارای ابتدا و انتهای مشخص هستند
	=01H	دریافت و سپس ارسال فریم هایی که دارای ابتدا و انتهای مشخص هستند
	=80H	دریافت فریم بدون بررسی ابتدا و انتها
	=81H	دریافت و سپس ارسال فریم بدون بررسی ابتدا و انتها
SR+1	High Byte	بایت مشخص کننده شروع فریم
	Low Byte	بایت مشخص کننده پایان فریم
SR+2	طول پیغام ارسالی (MAX : 511)	
SR+4 . . .	بایتهایی که می خواهیم از طریق پورت به تجهیزات دیگر بفرستیم (8 بیت کم ارزش فرستاده می شوند)	

WR+0	Low Byte	0
	High Byte	عدد 0 یعنی "نرمال"
WR+1 ~ WR+7	رجیسترهای کاری تابع 151	
WR+8	تعداد بایتهای دریافتی	
WR+9 . . .	بایتهای دریافتی	



ACT : در حال فرستادن فریم
ERR : خطا در دریافت یا ارسال
DN : موفقیت در ارسال یا دریافت بدون خطا

Pt : شماره پورت مربوطه
MD : مد کاری
SR : آدرس رجیستر شروع جدول
WR : تابع ۱۵۱ برای اجرا، نیاز به تعدادی رجیستر دارد

M1921 : هر ثانیه ۱۰ فریم می فرستد
M1922 : هر ثانیه ۱ فریم می فرستد

M1960 : برای پورت ۱
M1962 : برای پورت ۲
M1936 : برای پورت ۳
M1938 : برای پورت ۴

} بعد از خالی شدن پورت، بلافاصله فریم فرستاده می شود.

- این ارتباط میتواند تا 254 ایستگاه را پوشش دهد .
- "EN"این فانکشن فقط با لبه فعال می شود یعنی با هر لبه یکبار اطلاعات را برروی پورت می فرستد بنابراین ورودی EN این فانکشن را باید با کنتاکت M1921 یا M1922 یا M1962 و ... سری کرد . در هنگام تست و شروع کار بهتر است ابتدا از M1922 استفاده کرده تا بیتها ACT, ERR, DN قابل بررسی باشند و بعد از اطمینان از ارتباط می توان سرعت تبادل اطلاعات را بالاتر برد.
- رجیسترهای مربوط به پورتهای ارتباطی :

Signals \ Comm Port	Port 1	Port 2	Port 3	Port 4
1. Port Ready Indicator	M1960	M1962	M1936	M1938
2. Port Finished Indicator	M1961	M1963	M1937	M1939
3. Port Communication Parameters	R4146	R4158	R4043	R4044
4. TX Delay & RX Time-out Span	R4147	R4159	R4045	R4048

بنام خداوند بخشنده ومهربان

عنوان مدرک :	برنامه نویسی PLC FATEK
توضیحات :	FUN-161 & 162 MEMORY PACK
تعداد صفحه :	12
شماره ویرایش :	1
ویرایش کننده :	ا.رضایی
تاریخ ویرایش :	1391.7.19

دستورالعمل استفاده از Memory Pack :

در Main Unit مربوط به PLC های سری FBs امکان نوشتن و خواندن برنامه و همچنین خواندن و نوشتن دیتا بر روی Memory امکان پذیر می باشد .

FBs-Pack با نام Rom Pack تولید شده است و دارای ظرفیت 64 k می باشد .

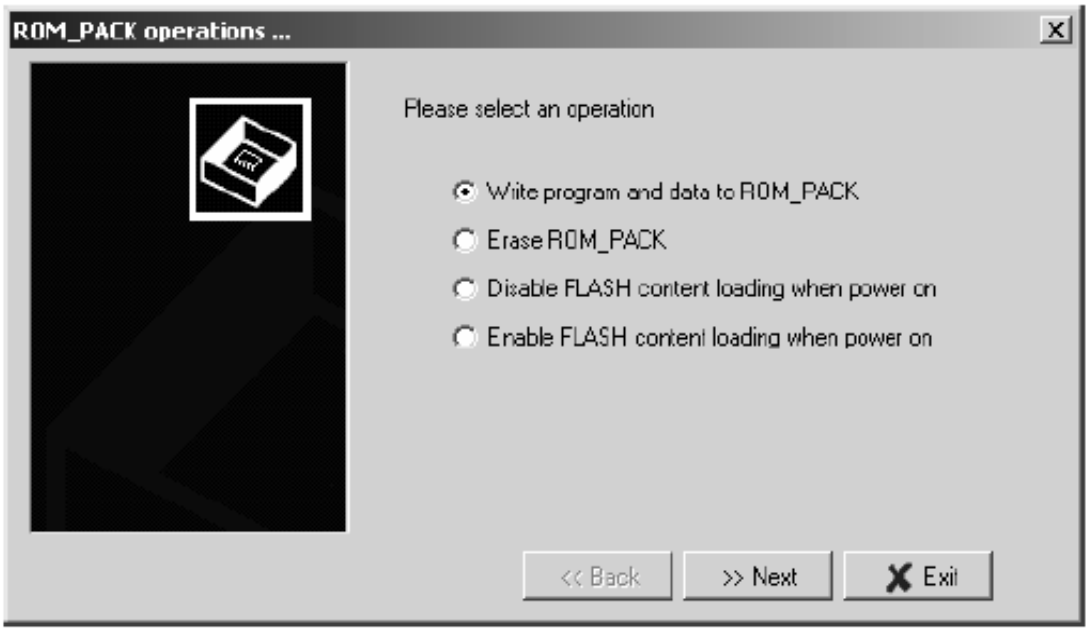
لازم به ذکر است تنظیمات DIP Switch ، امکان وضعیت حفاظت نشده در هنگام نوشتن دیتا بر روی آن (Off) و حفاظت شده (On) برای اجتناب از تداخل نوشتن ، بر روی Memory را فراهم می آورد .

متد زیر جهت استفاده از این نوع حافظه تنظیم شده است .

1.1 نوشتن برنامه و رجیسترهای دیتا در FBS-Pack توسط نرم افزار Winproladder :

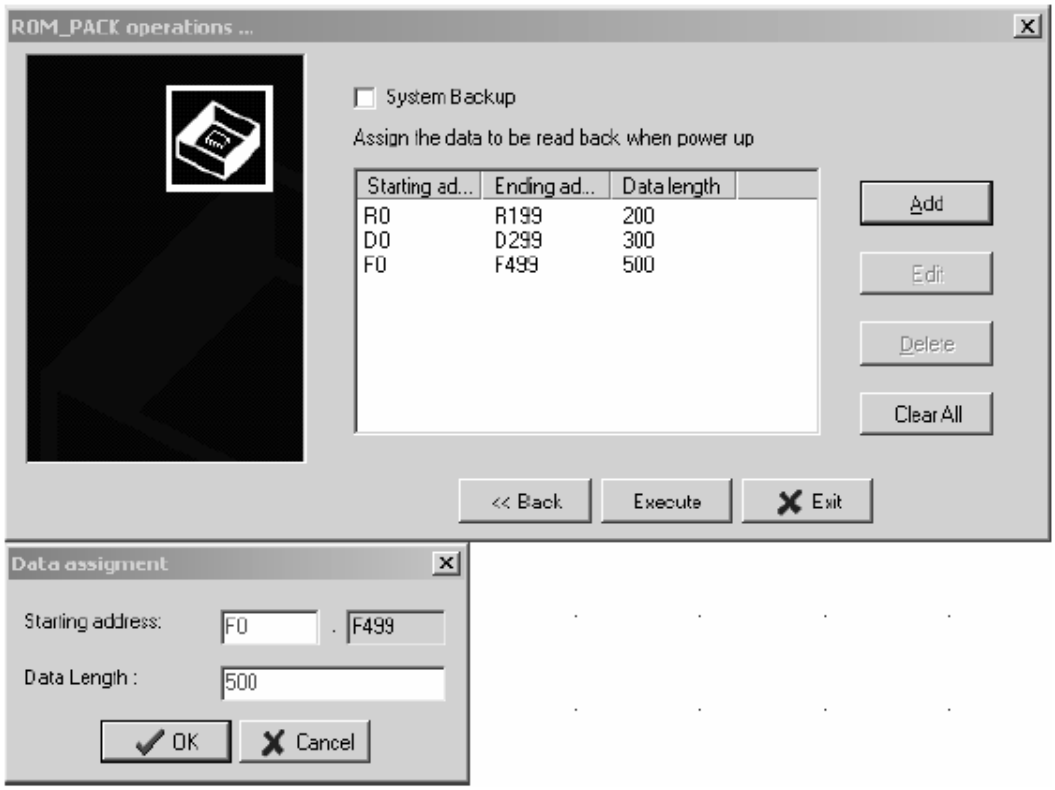
گزینه Memory Pack – Operation را از قسمت Tools انتخاب کنید .

در صورت انتخاب ، منویی مطابق شکل زیرظاهر خواهد شد .



Write Program and data to Pack :

گزینه اول یعنی Write Program and data to Pack **را انتخاب کنید سپس بعد از کلیک دکمه** Next **پنجره زیر ظاهر خواهد شد .**



کاربر در این قسمت می تواند رنج و بازه رجیسترهایی که می خواهد از Memory **خوانده شود و در** PLC **نوشته شود را تعیین نماید و اگر شما از دیتا رجیسترها** Back up **نمی خواهید** Execute **را فشار دهید و** Start **نمایید .**

زمان اجرای دستور شما بستگی به مقدار برنامه و مقادیر رجیسترهای ذخیره شده ، دارد . در هنگام ذخیره سازی ، سیستم پیام " Under Programming , Please Wait... " را ظاهر خواهد نمود و همینطور وقتی دیتا با موفقیت ارسال گردد پیام " Rom Pack Write Ok " ظاهر می شود و همچنین اگر در حین اجرا Fail **رخ دهد پیام " Rom Pack Write Error " ظاهر می شود .**

- – در این مجموعه اجازه دسته بندی 4 گروه رجیستر و یا Back up از سیستم جهت اداره Memory ، داده شده است .
- آیتم System Back up به مفهوم ذخیره تمامی داده های برنامه (اعم از PLC ID و PLC Station و ..) می باشد .

Erase Rom –Pack (پاک کردن Rom-Pack):

با بکاربردن این فانکشن کاربر می تواند برنامه ذخیره شده و یا دیتای موجود در Memory را پاک کند . با اجرای این عمل پیام ″ ... Under Erase Please wait " ظاهر می شود و این پیام تا ظاهر شدن پیام ″ Rom Pack Ok ″ باقی می ماند . اگر در Memory و یا اجرای دستور اشکالی پیش بیاید پیام ″ Rom Pack Erase Error ″ ظاهر می شود .

Disable Flash Content Loading Power On (غیر فعال کردن محتوای Flash هنگام روشن شدن) :

کاربر می تواند با کمک این فانکشن وارد Test Run Mode شود (با فشار دادن Next وارد این مد می شود) .

اگر کاربر به یک Rom Pack جدید احتیاج داشته باشد ، ابتدا این آیتم را انتخاب می کند . (انتخاب این آیتم به منظور جلوگیری از نوشته شدن برنامه Ladder جدید توسط Rom Pack بعد از Power On در داخل PLC می باشد .

Enable Flash Content Loading Power On (فعال کردن محتوای Flash هنگام روشن شدن) :

با فشار دادن Next کاربری نرمال آغاز خواهد شد یعنی بعد از هر بار Power On ، برنامه Ladder ذخیره شده و همچنین دیتاهای ذخیره شده در Rom به PLC منتقل می شود و PLC وارد مد Run می شود (بدون توجه به اینکه قبل از آن PLC ، Run یا Stop بوده است) .

این محصول برای تولید انبوه ماشین آلات و همچنین برای تعمیرات و نگهداری بهتر گزینه خوبی است .

قابل توجه کاربران محترم :

تنظیمات نوشته شده در بخش زیر ، در نحوه استفاده از Rom در کاربری های عادی قابل استفاده نمی باشد و کاربردهای خاص دارد ، ولی مطالعه آن برای درک بهتر توصیه می گردد .

توجه : R4052 رجیستر اختصاصی برای عملکرد Rom Pack میباشد .

Register	Content value	Function
R4052	5530 H	**مد تست و تغییر برای برنامه PLC ، در صورتی که مجهز به Rom باشد .** **دو نوع حلفظه در واحد اصلی PLC برای ذخیره برنامه و رجیسترهای دیتا وجود دارد یکی Rom است که با باطری Back Up کار می کند و این تجهیزات برای همه PLC های این مجموعه استاندارد می باشد و برنامه و Data ها در اینجا اجرا می گردد و دیگری یک Option به نام Rom Pack است که برنامه و دیتاها به صورت مستقیم در آن اجرا نمی گردند .** **در مد تست و تغییر ، برنامه اصلی در داخل Ram که توسط باطری Back up پشتیبانی می شود وجود دارد و این برنامه در هنگام Power Up از Rom به Ram (یعنی از Rom به PLC) منتقل نمی شود . توصیه می شود بعد از اینکه تغییرات برنامه تمام شد و برنامه تست گردید این برنامه و رجیسترهای دیتا به Rom منتقل شود زیرا این یک راه بهتر برای Save طولانی مدت دیتا در داخل PLC و همچنین تعمیرات و نگهداری راحتتر ماشین یا برای تولید انبوه ماشین آلات است .** **در طول تغییر و تست برنامه اگر کاربر بخواهد برنامه را برگرداند ، کافی است که R4052 را صفر کرده و خاموش و روشن نماید (PLC) در این صورت برنامه و رجیسترها از Rom به داخل Ram مربوط به PLC ریخته می شود و اجرا می گردد . (Main Unit به حالت قبل از تغییر بازمی گردد) .**
	----------	----------
	اگر سایر مقادیر را داشته باشد	**عملکرد نرمال و مد نوشتن :** **اگر PLC با Rom مجهز شود ، بعد از هر بار روشن و خاموش شدن برنامه و دیتا از Rom به PLC منتقل می شود و PLC وارد مد Run می شود (و البته صرفنظر از اینکه در چه مدی بوده است Run یا Stop)**

برخی از کاربردها احتیاج دارند که با رجیسترهای انتخاب شده فقط یکبار Initialize شوند و سپس این مقادیر دررجیسترهای Retentive باقی می ماند در این حالت مطابق زیر این رجیستر را تغییر می دهیم .

Register	Content value	Function
R4046	5530 H ---------- اگر سایر مقادیر را داشته باشد	مقادیر دیتا رجیسترهای PLC ، بعد از هر باررروشن و خاموش شدن با مقادیر Rom مقداردهی نخواهند شد . ---------- مقادیر رجیسترهای PLC ، بعد از هربار روشن شدن با مقادیر Rom مقداردهی خواهند شد .

- اگر فقط یکبار Initialize با مقادیر رجیسترهای ذخیره شده در Rom احتیاج باشد ، لازم است که کاربر مقدار 5530H را درداخل رجیستر R4046 مقداردهی نماید .

- کاربر می تواند در هر دو مد PLC (Run و Stop) فرمان پاک کردن Rom و یا نوشتن برنامه و رجیسترهای انتخاب شده را در Rom ، را مطابق جدول زیر ، بدهد .

Register	5550 H	فرمان پاک کردن Rom را می دهد
	5551 H	وضعیت ، حافظه پاک شده است را نشان می دهد
	5552 H	وضعیت ، تغییرات برای پاک کردن را به ما نشان می دهد
	5553 H	وضعیت کامل شدن فرمان Clear را به ما نشان می دهد
	5554 H	به ما می گوید که خطایی در پاک شدن Memory رخ داده است
	5560 H	فرمان نوشتن برنامه و رجیسترهای انتخاب شده را به Rom می دهد
	5562 H	وضعیت نوشته شدن Ladder را به ما نشان می دهد
	5563 H	وضعیت نوشته شدن Register ها را به ما نشان می دهد
	5566 H	وضعیت تغییرات در Ladder را به ما نشان می دهد
	5567 H	وضعیت تغییرات در رجیسترها را به ما نشان می دهد
	5569 H	وضعیت تغییرات در Special Register ها را به ما نشان می دهد
	556A H	وضعیت نوشتن کامل شده است را به ما نشان می دهد
	556B H	وضعیت وجود اشکال در نوشتن Ladder را به ما نشان می دهد
	556C H	وضعیت وجود اشکال در نوشتن Register ها را به ما نشان می دهد

تعیین و بازبینی رجیسترهای ذخیره شده در PLC :

این قسمت شامل انتخاب رجیسترهایی است که می توانند در Rom ذخیره شوند و یا خوانده شوند (در هر بار روشن شدن PLC) مقادیر متغییر یا فیکس می توانند در Rom (برای این نوع کاربرد به طور مناسب و هر وقت که باطری سیستم خالی شود) ذخیره شوند .

رجیسترهای ویژه R4030 تا R4039 برای تعیین هر کدام از گروههای رجیستری (کاربردهایی که در بالا شرح داده شد) که در Rom نوشته می شوند ، استفاده می شوند.

برای این کار لازم است که ابتدا تعیین کنید و سپس فرمان Write بدهید .

Register	Content value	Function
R4030	A66A H --- اگر سایر مقادیر را داشته باشد	این فلگی است که تعیین می کند که چه رجیسترهایی برای نوشتن یا خواندن (مطابق تنظیمات رجیسترهای R4031 تا R4039) انتخاب شده اند (رجیسترهای ماندگار Retentive این فانکشن را Support می کنند) . --- هیچ نوع رجیستری جهت خواندن یا نوشتن در Rom وجود ندارد

Register	Content value	Function
R4031	1 تا 4	تعداد گروههای رجیستری را که قرار است نوشته یا خوانده شود (در Rom) را تعیین می کند .

Register	Content value	Function
R4030	A66A H --- اگر سایر مقادیر را داشته باشد	طول دیتا در گروه رجیستری 0 طول 1 تا 3840 برای رجیسترهای R0 تا R3840 طول 1 تا 3072 برای رجیسترهای R5000 تا R8071 طول 1 تا 4096 برای رجیسترهای D0 تا D4095 طول 1 تا 166 برای رجیسترهای R4165 تا R4000 هنگامی که طول 7FF7 تعیین گردد ، این برای سیستم به معنی این

است که از همه سیستم اعم از PLC ID ,PLC Station یک Back up گرفته شود . اگر طول غیر مجاز باشد یا خارج از محدوده باشد عمل نخواهد کرد .		

توجه کنید که برای رجیسترهای R4033 و R4034 همانند رجیسترهای قبلی عمل می شود و به صورت جفتی استفاده می گردند .

نکته دیگر اینکه برای خواندن و یا نوشتن بر روی Rom می توانید همان کارهای بالا را توسط فانکشن های 162 و 163 انجام دهید .

F-161 نوشتن Data بر روی Data Pack :

S : آدرس شروع دیتایی که می خواهیم از PLC بر روی Rom نوشته شود .

Bk : شماره بلوک دیتا (بر روی Pack)

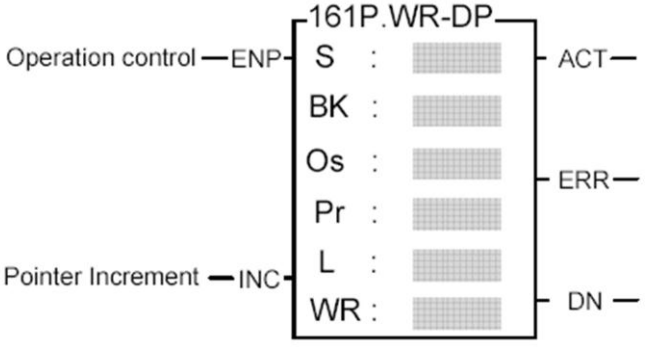
Os : میزان Offset که بر روی بلوک دیتا اعمال می شود

Pr : آدرس Pointer و اشاره گر (آدرس مکان جایی که بروی بلوک Rom برای نوشتن به آن اشاره می شود)

L : تعداد (طول) رجیسترهایی که می خواهیم ذخیره شوند .

WR : آدرس شروع رجیسترهایی که برای انجام فانکشن توسط نرم افزار WinPro اشغال می گردد

Ladder symbol

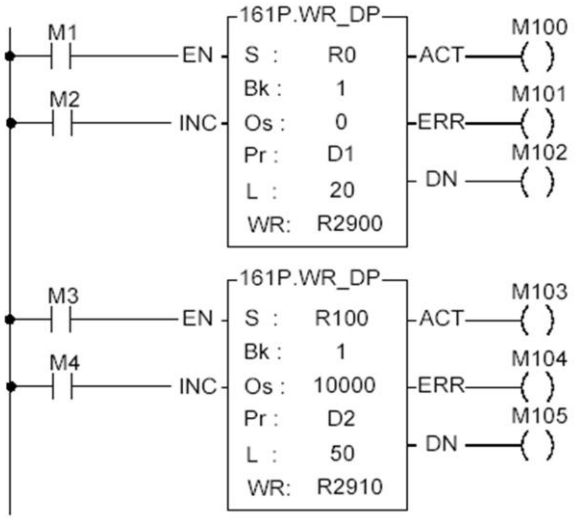


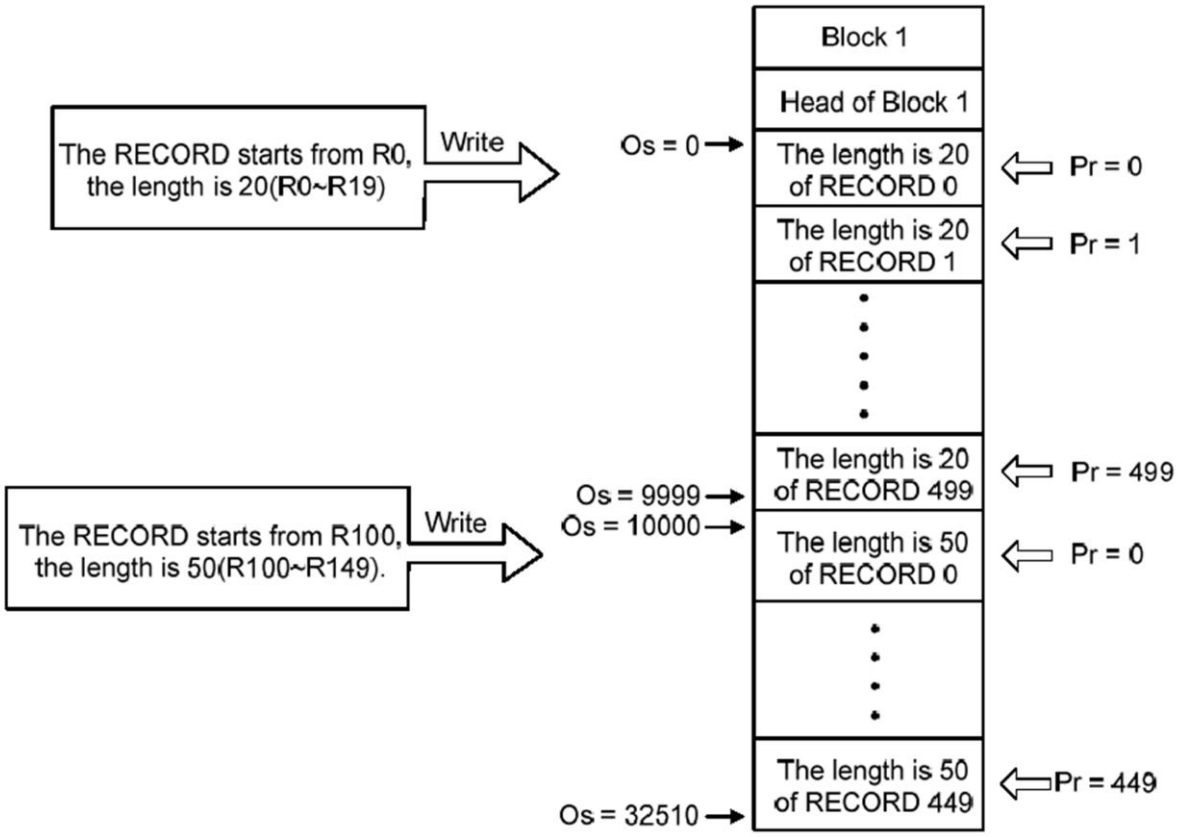
توجه : مقدار S ممکن است با اشاره گرهای V و Z و P0 تا P9 برای آدرس دهی غیر مستقیم استفاده گردد .

هدف از استفاده Rom Pack ، ذخیره طولانی مدت دیتا و Ladder است و می تواند به عنوان جایی که پارامترهای ماشین را در داخل آن ذخیره می کنیم استفاده شود .

هنگامی که پایه En از 0 به 1 تغییر وضعیت می دهد عملیات نوشتن دیتا از جایی که آدرس آن در S مشخص شده آغاز می شود ، Bk شماره بلوکی که عملیات ذخیره سازی در آن انجام می شود را مشخص می کند ، Os میزان Offset در بلوک مشخص می کند ، Pr اشاره می کند به مکان ذخیره سازی و L طول دیتای نوشته شده را مشخص می کند .

به مثال زیر توجه فرمائید :




Block 1
Head of Block 1
The RECORD starts from R0, the length is 20(R0~R19)
Write
Os = 0
The length is 20 of RECORD 0
Pr = 0
The length is 20 of RECORD 1
Pr = 1
The length is 20 of RECORD 499
Pr = 499
Os = 9999
Os = 10000
The RECORD starts from R100, the length is 50(R100~R149).
Write
The length is 50 of RECORD 0
Pr = 0
The length is 50 of RECORD 449
Pr = 449
Os = 32510

- تا زمانی که Inc=1 باشد ، محتوی اشاره گر یک عدد زیاد می شود (در عملیات بعدی اجرای تابع)
- اگر مقدار L برابر صفر باشد و یا از مقدار 128 بیشتر شود ، Err یک خواهد شد و عملیات نوشتن دیتا انجام نمی پذیرد .
- برای اجرای تابع لازم است PLC محاسبات لازم را انجام دهد و دیتایی که لازم است ذخیره شود را جهت مقداردهی صحیح اسکن کند و تغییرات لازم را اعمال نماید ، در طول اجرای Function خروجی Act برابر 1 می شود و هنگامی که عملیات اجرا شد و کامل گردید (البته بدون Error) خروجی Dn برابر 1 می شود و هنگامی که Error وجود داشته باشد خروجیِ Error برابر 1 می شود .

Rom می تواند جهت ذخیره سازی Ladder یا دیتا یا هردو نظام بندی شود ولی ذکر این نکته لازم است که Ladder می تواند فقط در بلوک 0 ذخیره شود ولی Data می تواند در هر دو ذخیره شود و هر بلوک نیز 32K ظرفیت دارد .

F162 خواندن Data از روی Data Pack

Bk : شماره بلوک را تعیین می کند

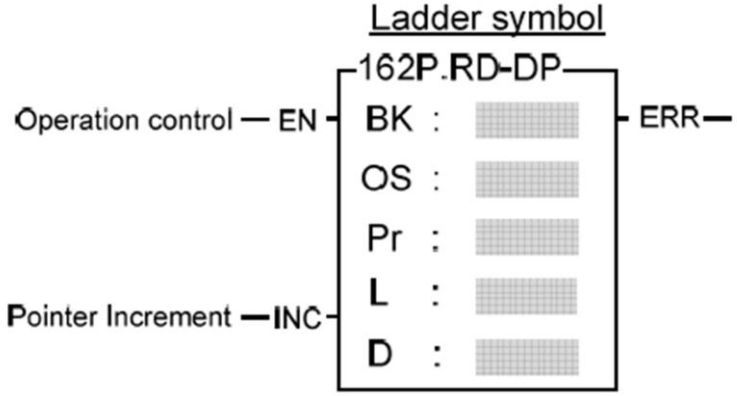
Os : Offset بلوک دیتا (افستی که مطابق تابع به آن آدرس دهی اضافه شده است و از آنجا دیتا خوانده می شود)

Pr : آدرس مربوط به اشاره گر (اشاره به دیتای Rom می کند)

L : طول دیتای خوانده شده

D : آدرس شروع ذخیره سازی مقادیر خوانده شده

Ladder symbol

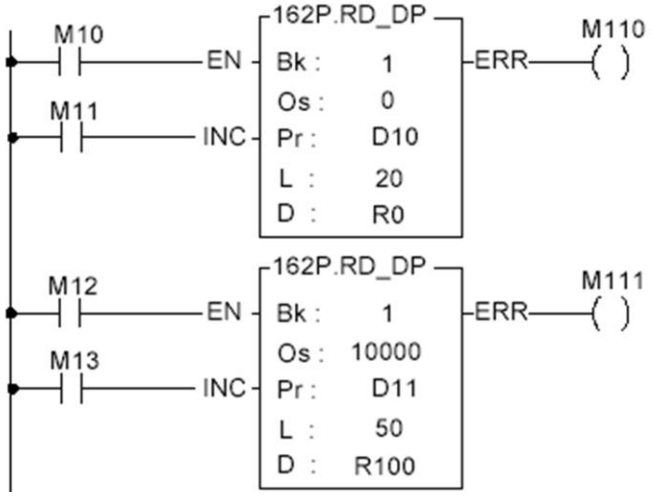


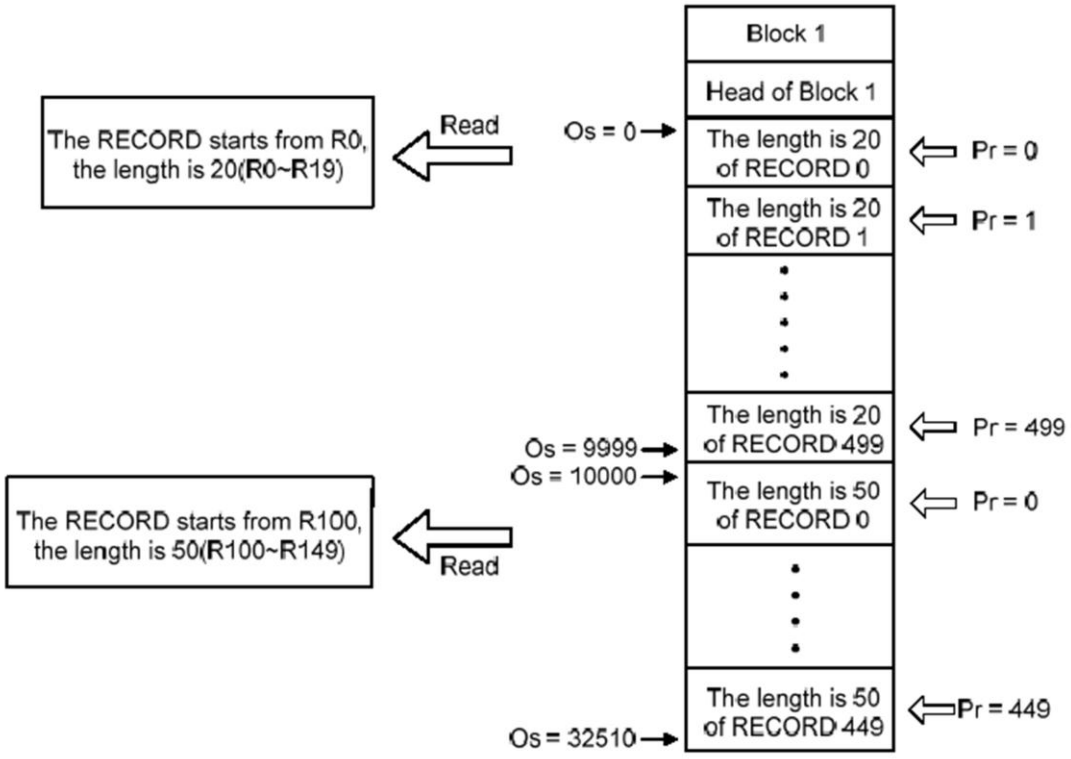
در حقیقت با تابع قبلی ما مقادیر دیتا را در داخل Memory Pack می نوشتیم و با این تابع آن مقادیر (مقادیری که برای راه اندازی و کارکرد ماشین از Memory Pack خوانده می شود) را می خوانیم .

عملکرد این تابع همانند تابع قبلی است .

پایه Error در صورتی یک می شود که یا Memory Pack خالی باشد یا فرمت دیتای داخل آن صحیح نباشد .

به مثال زیر توجه فرمائید .





بنام خداوند بخشنده ومهربان

عنوان مدرک :	نرم افزار FATEK PLC
توضیحات :	وقفه ها
تعداد صفحه :	5
شماره ویرایش :	1
ویرایش کننده :	ا.رضایی
تاریخ ویرایش :	1393.04.08

وقفه های PLC به دو دسته تقسیم می شوند:

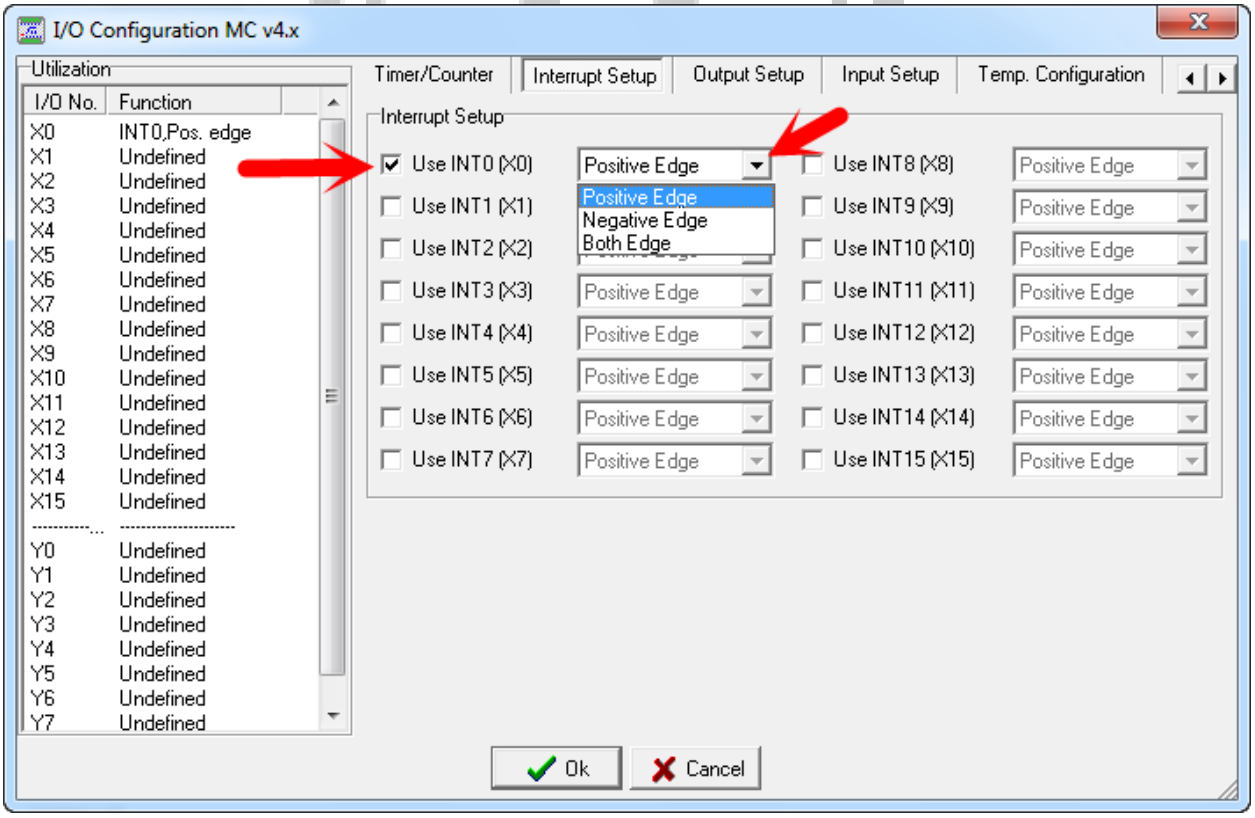
1- وقفه های سخت افزاری

2- وقفه های نـــــرم افزاری

وقفه های سخت افزاری :

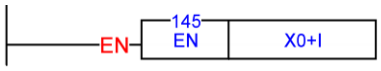
ورودی های دیجیتال X0~X15 می توانند این وقفه ها را ایجاد کنند. وقتی ورودی های دیجیتال فعال شوند (با لبه بالا رونده یا پایین رونده یا هر دو) اسکن برنامه به خطی از برنامه که با لیبل X0-I~X0-15 , X0+I~X15+I معرفی می شود می رود و برنامه وقفه نوشته شده را یکبار اجرا می کنند. در واقع لیبل های X0-I~X0-15 , X0+I~X15+I برای این منظور در برنامه رزرو شده اند و برنامه نویس نمی تواند از این لیبل ها در موارد دیگر استفاده کند.

در گزینه I/O Configuration از منوی project در صفحه Interrupt setup باید ورودی دیجیتال را تنظیم کرد.

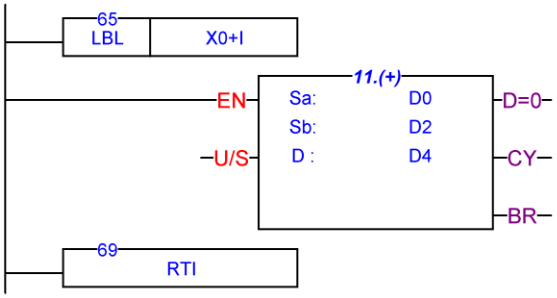


در این نوع اینتراپت باید با تابع **145 (ENABLE OF INTERRUPT)** اینتراپت مورد نظر فعال شود.

تابع شماره **145** در برنامه **Main** باید نوشته شود:



نوشتن لیبل و برگشت از وقفه در **SUB** :



2- وقفه های نـــــرم افزاری :

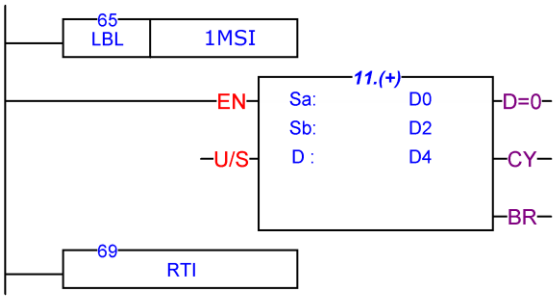
وقفه های زمانی ، وقفه شمارنده های سخت افزاری سرعت بالا، وقفه های مربوط به توابع تولید پالس از این جمله هستند.

وقفه های زمانی : چنانچه بخواهیم برنامه ای بصورت دوره ای با بازه های زمانی ثابت اجرا شود می توانیم از این اینتراپتها استفاده کنیم.

نحوه اجرای وقفه زمانی : کافیست لیبل مربوط به وقفه زمانی (جدول زیر) را در صفحه SUB نوشته و در انتهای برنامه از فانکشن شماره 69 (Return from interrupt routine) استفاده کنیم.

Label name	Description
1MSI	One interrupt every 1mS
2MSI	One interrupt every 2mS
3MSI	One interrupt every 3mS
4MSI	One interrupt every 4mS
5MSI	One interrupt every 5mS
10MSI	One interrupt every 10mS
50MSI	One interrupt every 50mS
100MSI	One interrupt every 100mS

نوشتن لیبل و برگشت از وقفه در SUB :



وقفه های مربوط به شمارنده سرعت بالا (HSC) ، وقتی فراخوانی می شوند که تعداد پالس شمارش شده با مقدارپالس مورد نظر برابر شود (توضیحات مربوط به این وقفه در بخش توابع (شماره های 92و93) آمده است)

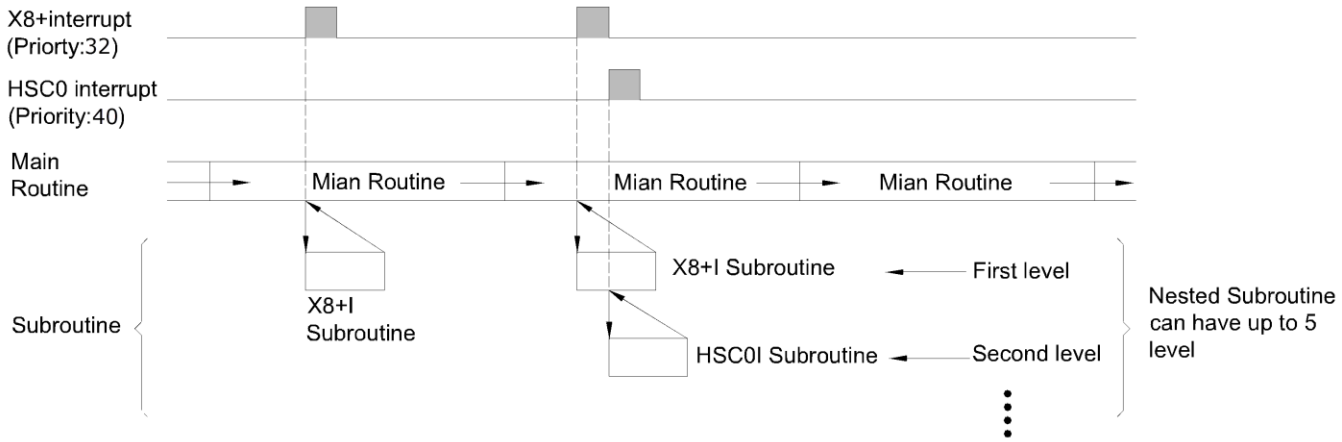
وقفه های توابع تولید پالس (PSO) ، وقتی فراخوانی می شود که در خروجی تعداد پالس مورد نظر ایجاد شده باشد .

اولویت

در جدول زیر در ستون Priority اولویت اجرای هر وقفه آمده است. چنانچه دو وقفه با یکدیگر فراخوانده شوند (یا در حین اجرای یک وقفه ، وقفه دیگری نیز فراخوانی شود) ، وقفه ایی که دارای اولویت بالاتر است اجرا می شود.

عدد Priority بالاتر به معنای اولویت بالاتر می باشد.

Priority	Interrupt Label	Priority	Interrupt Label	Priority	Interrupt Label
1	X15−I	18	X7+I	35	PSO1I
2	X15+I	19	X6−I	36	PSO0I
3	X14−I	20	X6+I	37	HSC3I/HST3I
4	X14+I	21	X5−I	38	HSC2I/HST2I
5	X13−I	22	X5+I	39	HSC1I/HST1I
6	X13+I	23	X4−I	40	HSC0I/HST0I
7	X12−I	24	X4+I	41	100MSI (100MS)
8	X12+I	25	X3−I	42	50MSI (50MS)
9	X11−I	26	X3+I	43	10MSI (10MS)
10	X11+I	27	X2−I	44	5MSI (5MS)
11	X10−I	28	X2+I	45	4MSI (4MS)
12	X10+I	29	X1−I	46	3MSI (3MS)
13	X9−I	30	X1+I	47	2MSI (2MS)
14	X9+I	31	X0−I	48	1MSI (1MS)
15	X8−I	32	X0+I	49	HSTAI (ATMRI)
16	X8+I	33	PSO3I		
17	X7−I	34	PSO2I		



بنام خداوند بخشنده ومهربان

عنوان مدرک :	FATEK HMI
توضیحات :	برنامه نویسی PLC برای دستگاه های پرکن و ماژول لودسل
تعداد صفحه :	10
شماره ویرایش :	1
ویرایش کننده :	ا.رضایی
تاریخ ویرایش :	1393.09.19

نحوه اتصال لودسلها در سیستم های توزین که بیشتر از یک لودسل دارند :

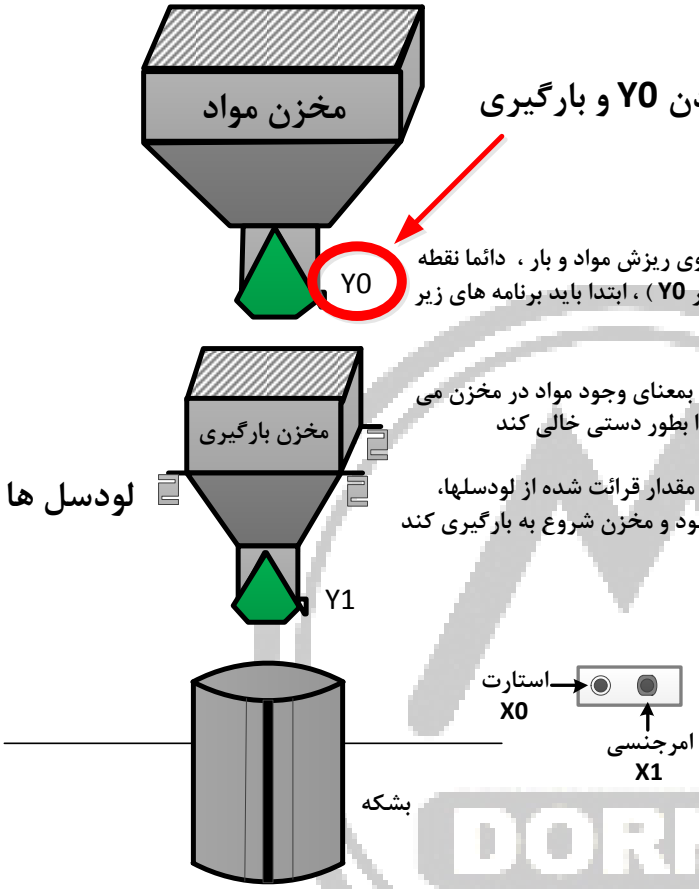
برای مثال اگر 4 لودسل به یک مخزن متصل شده اند سیم های -Sig , +Sig , -Exc , +Exc هر چهار لودسل را به یکدیگر متصل کنید و در واقع هر چهار لودسل با یکدیگر موازی می شوند.

بهترین راه اتصال سیم های لودسل ها لحیم کردن آنها به یکدیگر است ولی اگر می خواهید از جانکشن باکس استفاده کنید حتما از مدلهای خوب با اتصالات خوب استفاده کنید چون در بعضی موارد مشاهده شده که استفاده از جانکشن باکس های نامناسب دقت بارگیری را کم می کنند.

تنظیم لودسل ها از لحاظ مکانیکی :

برای مثال اگر وزنه ۵ کیلویی را در هر نقطه ایی از مخزن قرار دهید، وزن اندازه گیری شده توسط لودسل ها در هر نقطه، باید ۵کیلو باشد.

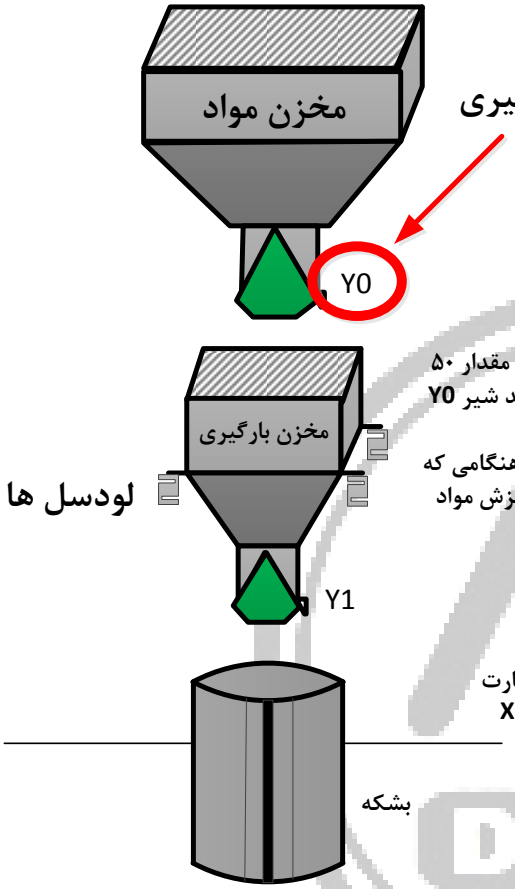
نحوه روشن کردن Y0 و بارگیری



چون لودسلها بدلیل قرارگرفتن در فشارهای مکانیکی و نیروی ریزش مواد و بار ، دائما نقطه صفر خود را از دست می دهند.قبل از بارگیری (بازکردن شیر **Y0**) ، ابتدا باید برنامه های زیر در **PLC** نوشته شود.

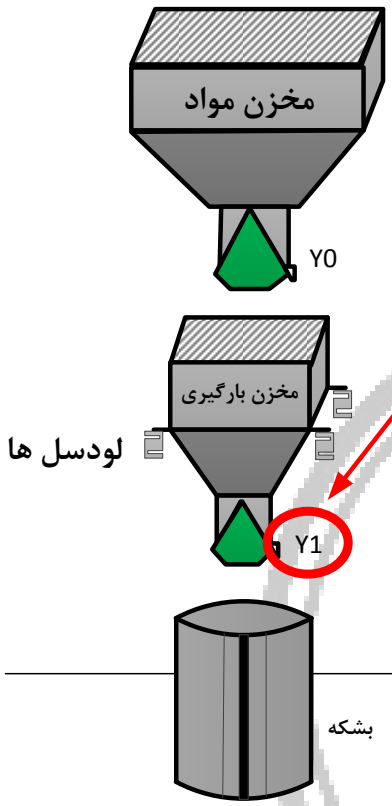
۱- اگر مقدار قرائت شده از لودسل ها از حد مجاز بیشتر بود بمعنای وجود مواد در مخزن می باشد در این حالت نباید شیر باز شود و اپراتور باید مخزن را بطور دستی خالی کند

۲- اگر مقدار قرائت شده از لودسل ها از حد مجاز کمتر بود مقدار قرائت شده از لودسلها، باید نقطه صفر درنظر گرفته شود. و پس از آن شیر **Y0** باز شود و مخزن شروع به بارگیری کند



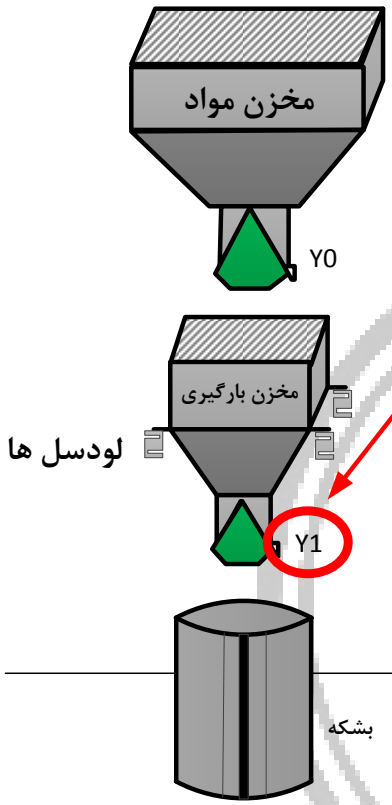
در نظر بگیرید که وزن بارگیری، ۵۰ کیلوگرم می باشد. باید قبل از اینکه لودسلها مقدار ۵۰ کیلوگرم را نشان دهند شیر **Y0** خاموش شود. برای مثال در ۴۹کیلو و ۹۰۰گرم باید شیر **Y0** خاموش شود .

در این مثال مقدار مواد از لحظه ایی که شیر **Y0** شروع به بسته شدن می کند تا هنگامی که شیر بطور کامل بسته شد ۱۰۰ گرم در نظر گرفته شده است. به این مقدار ، وزن ریزش مواد گفته می شود.



نحوه روشن کردن Y1 و تخلیه مواد از مخزن به بشکه

وقتی شیر **Y0** خاموش شد ، لودسلها همچنان در حال لرزش و تکانهای مکانیکی در اثر ریزش مواد هستند، برای اینکه مقدار وزن بارگیری شده را بدانیم باید اندکی صبر کنیم تا لرزش ها متوقف شده و لودسل به پایداری برسد
ذاتا لودسلها نمی توانند سریعا وزن را نشان دهند و نیاز به اندکی تاخیر در روند وزن کشی دارند.



نحوه خاموش کردن Y1 و اتمام تخلیه مواد از مخزن به بشکه

وقتی مقدار وزن مخزن از حدی کمتر شد بمعنای خالی شدن مخزن می باشد .
در این هنگام می توان شیر **Y1** را خاموش کرد .

برای بارگیری مجدد بعد از بسته شدن شیر **Y1** باید اندکی صبر کرد تا ضربه ایی که شیر **Y1** به مخزن وارد کرده که در نتیجه آن لرزش مخزن و نوسان لودسلها می باشد به پایان برسد و لودسلها در حالت پایدار وزن مخزن خالی را نشان دهند.

برنامه PLC در نرم افزار Winproladder

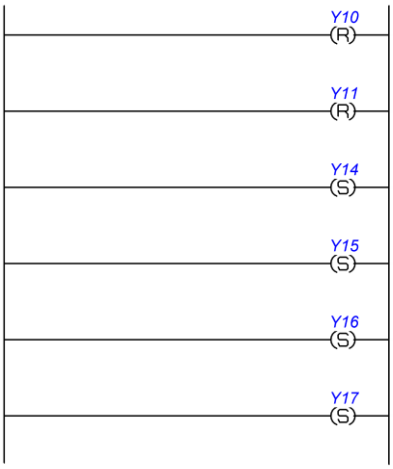
ابتدا ماژول لودسل را تنظیم می کنیم. تنظیمات ماژول لودسل با **Set** یا **Reset** کردن خروجی های دیجیتال که به هر کارت اختصاص می یابد انجام می شود.

FBs-1LC

Signal	Name	Function Description	
Y_s+1,Y_s+0	SPAN	00	0~10mV(2mV/V)
		01	0~25mV(5mV/V)
		10	0~50mV(10mV/V)
		11	0~100mV(20mV/V)
Y_s+3,Y_s+2	RESERVED	Reserved	
Y_s+5,Y_s+4	CONVSERSION RATE	00	5Hz
		01	10Hz
		10	20Hz
		11	25Hz
Y_s+7,Y_s+6	AVERAGE COUNT	00	No Average
		01	2 Samples
		10	4 Samples
		11	8 Samples

برای مثال اگر ماژول لودسل به پی ال سی **FBs-24MA** وصل شده باشد، شروع خروجی های ماژول لودسل از **Y10** شروع می شود. با توجه به جدول زیر خروجی های (**Y10** و **Y11** برای تنظیم **Span**) ، (**Y12** و **Y13** رزرو) ، (**Y14** و **Y15** برای تنظیم سرعت اسکن لودسل) ، (**Y16** و **Y17** برای تنظیم تعداد میانگین گیری) استفاده می شوند.

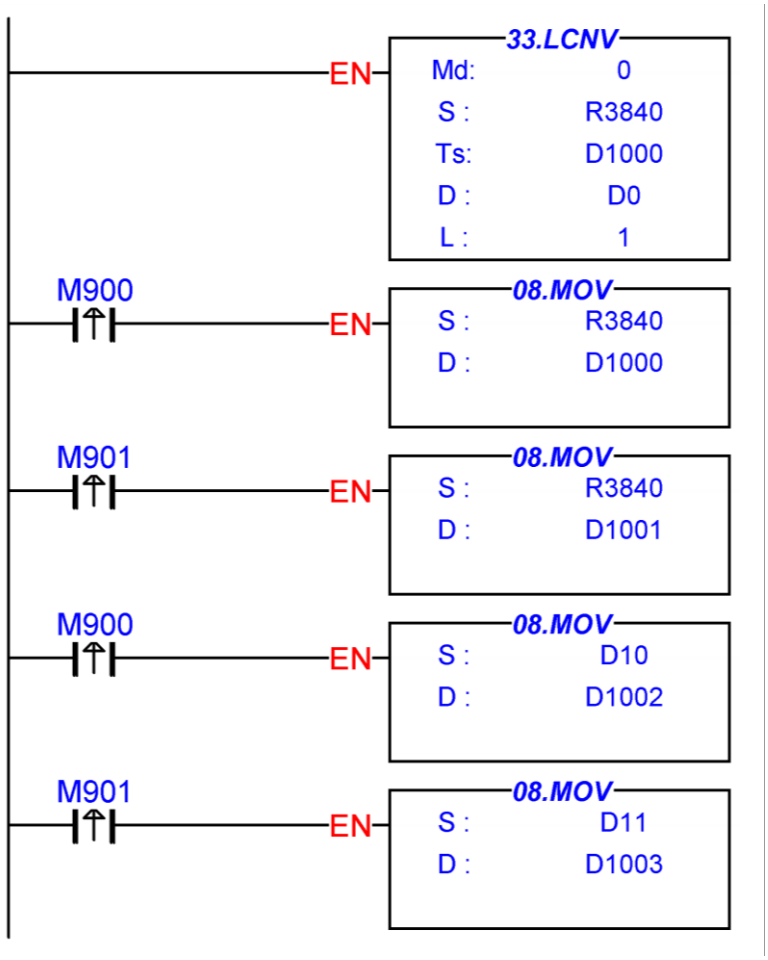
مطابق برنامه زیر ابتدا سرعت اسکن را **25** بار در ثانیه و تعداد میانگیری را **8** نمونه قرار می دهیم و چنانچه این مقادیر برای این مجموعه مناسب نبود آنها را تغییر می دهیم تا بهترین دقت گرفته شود.



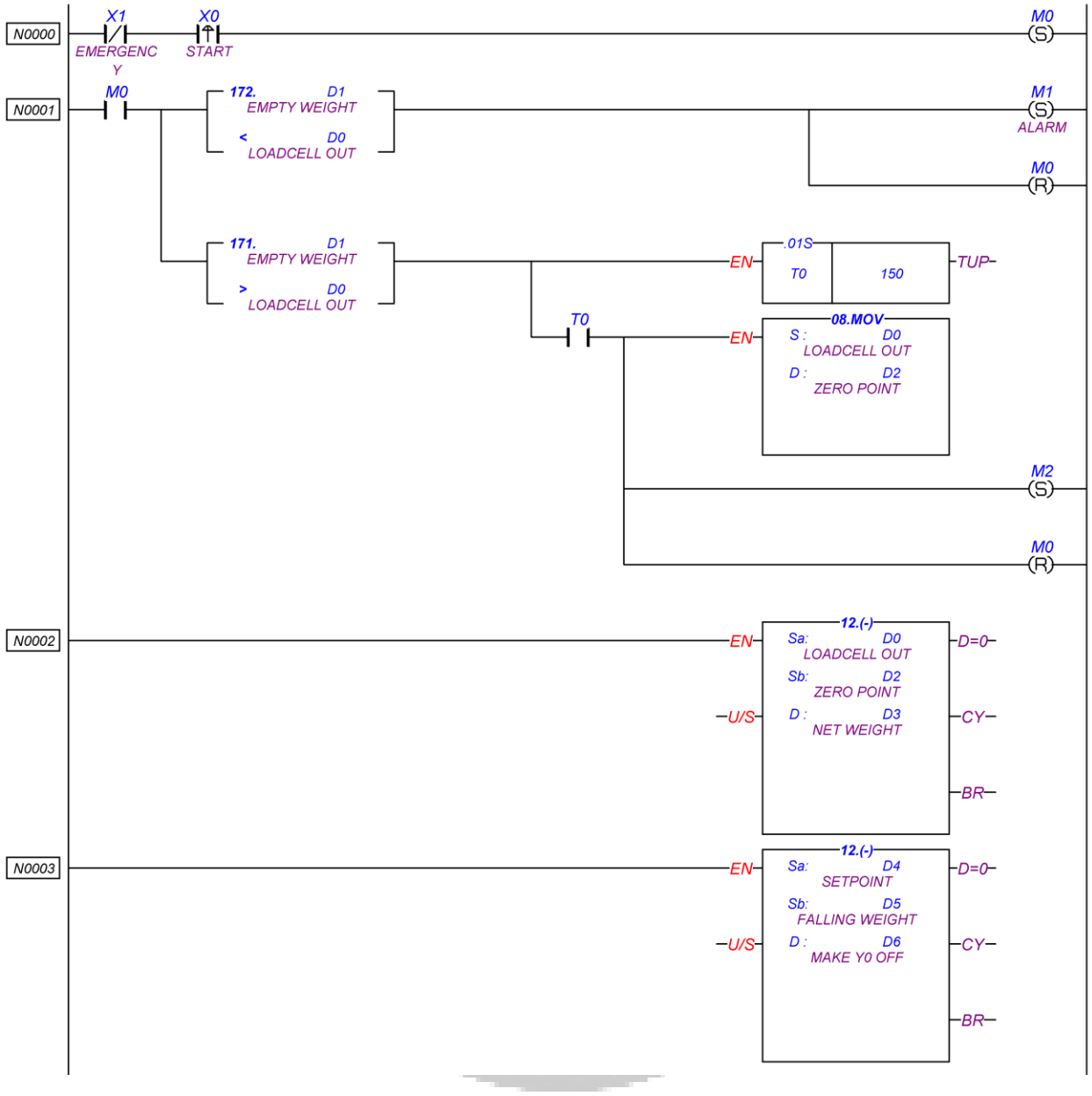
سپس مقدار آنالوگ لودسل که مقداری بین **32000+** و **32000-** می باشد را با استفاده از تابع **33** کالیبره می کنیم و نتیجه این تابع را در رجیستر **D0** قرار می دهیم .

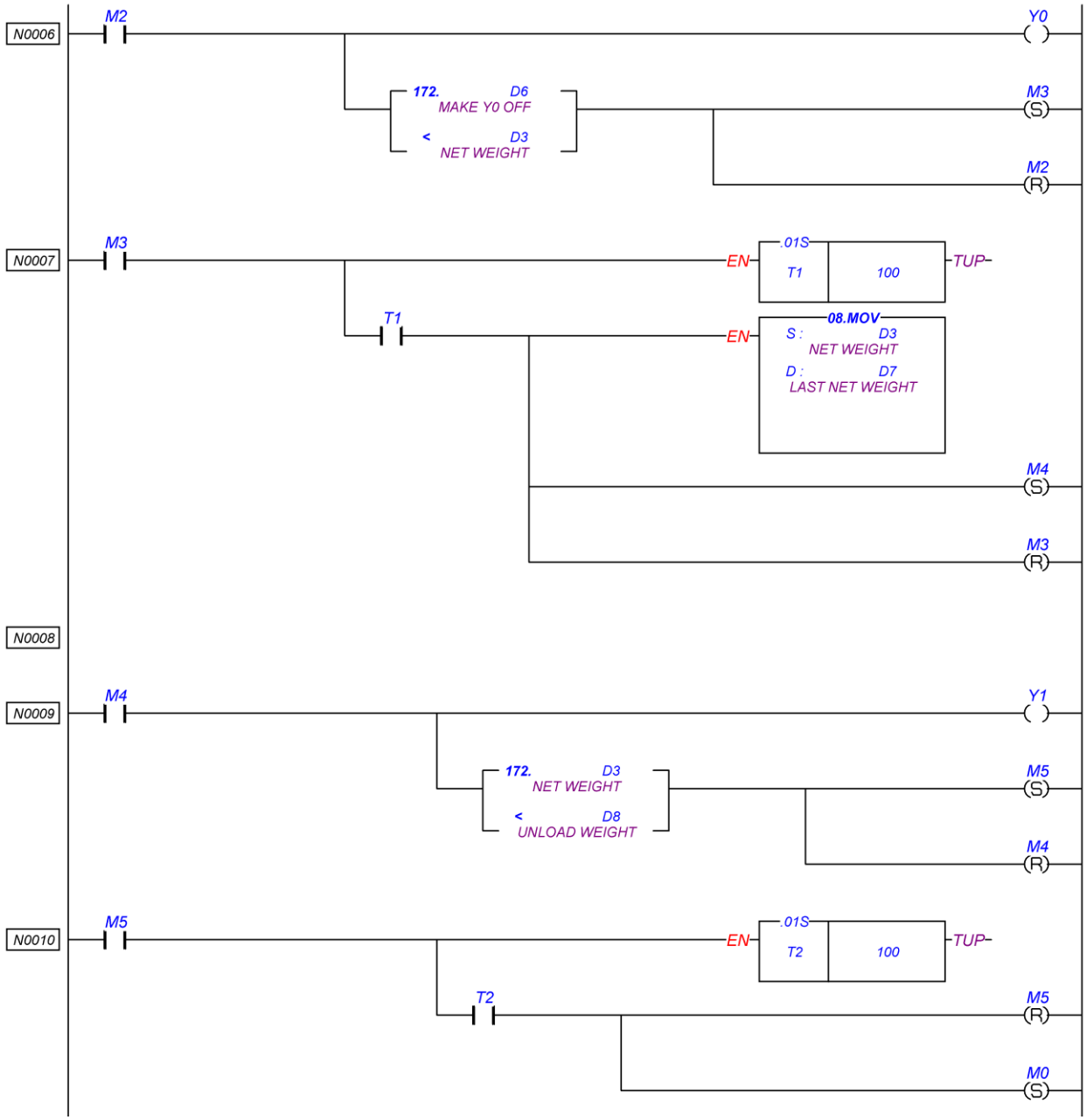
می خواهیم لودسل را در وزن استاندارد که از روی صفحه **HMI** وارد کردیم کالیبره کنیم :

D10	نقطه کالیبره ۱ (مثلا صفر کیلوگرم)	M900	تایید کالیبره حد پایین
D11	نقطه کالیبره ۱ (مثلا ۵۰ کیلوگرم)	M901	تایید کالیبره حد بالا



در ادامه برنامه **PLC** را مطابق توضیحات گفته شده می نویسیم :


N0000
X1
EMERGENCY
X0
START
M0
N0001
M0
172. D1
EMPTY WEIGHT
< D0
LOADCELL OUT
M1
ALARM
M0
171. D1
EMPTY WEIGHT
> D0
LOADCELL OUT
EN
.01S
T0
150
TUP
T0
EN
08.MOV
S : D0
LOADCELL OUT
D : D2
ZERO POINT
M2
M0
N0002
EN
12.(-)
Sa: D0
LOADCELL OUT
Sb: D2
ZERO POINT
U/S
D : D3
NET WEIGHT
D=0
CY
BR
N0003
EN
12.(-)
Sa: D4
SETPOINT
Sb: D5
FALLING WEIGHT
U/S
D : D6
MAKE Y0 OFF
D=0
CY
BR

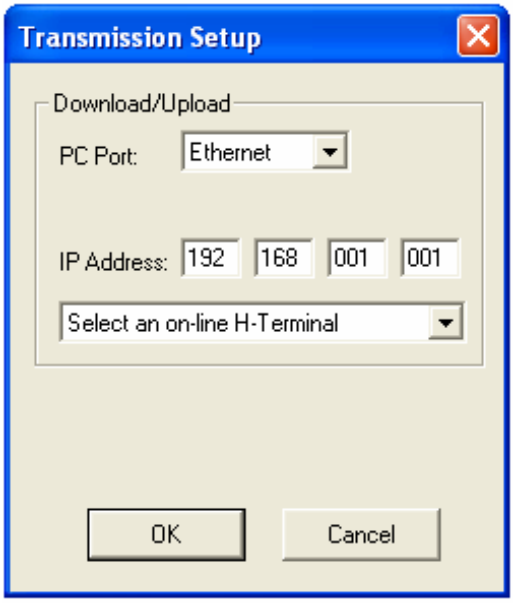

N0006
M2
Y0
172.
D6
MAKE Y0 OFF
<
D3
NET WEIGHT
M3
M2
N0007
M3
.01S
EN
T1
100
TUP
T1
08.MOV
EN
S :
D3
NET WEIGHT
D :
D7
LAST NET WEIGHT
M4
M3
N0008
N0009
M4
Y1
172.
D3
NET WEIGHT
<
D8
UNLOAD WEIGHT
M5
M4
N0010
M5
.01S
EN
T2
100
TUP
T2
M5
M0

اتصال PC به HMI (ارسال برنامه به HMI) از طریق پورت Ethernet

1- ابتدا برای HMI یک IP تعریف می کنیم .(در قسمت Configuration) و همچنین Upload Download Copy Port را Ethernet انتخاب می کنیم .

2- HMI را با یک کابل Cross به PC متصل می کنیم .

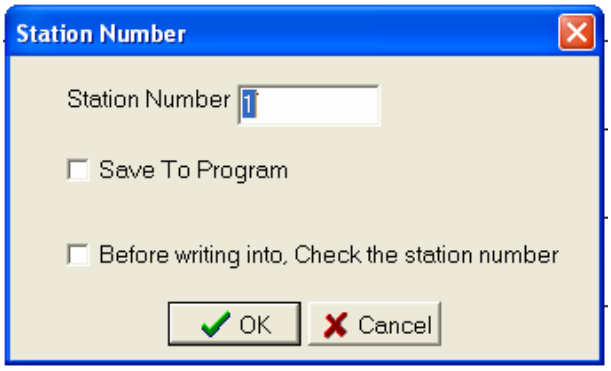
3- در قسمت Transmission Setup از منوی Option نرم افزار مربوطه (H-Designer) مطابق شکل زیر PC Port را Ethernet انتخاب کرده و در قسمت IP نیز IP تنظیم شده در Configuration را قرار می دهیم .

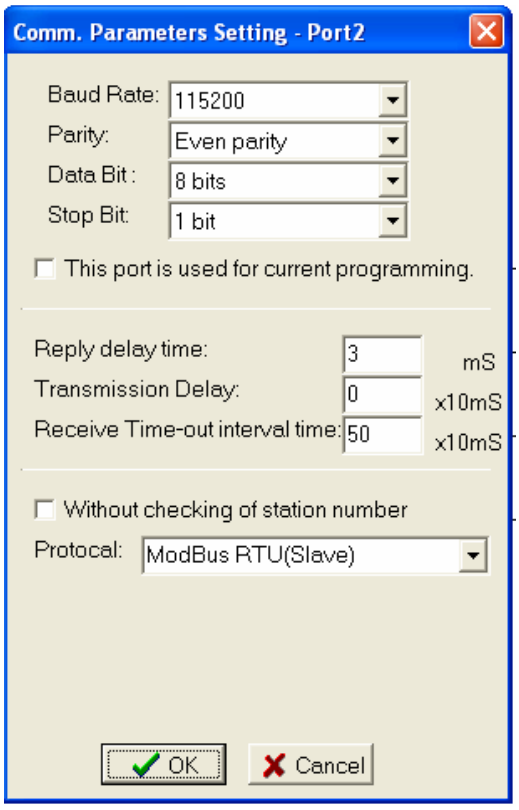


4 – برنامه را کامپایل کرده و DownLoad FirmWare And Application را زده و برنامه را Down Load می کنیم .

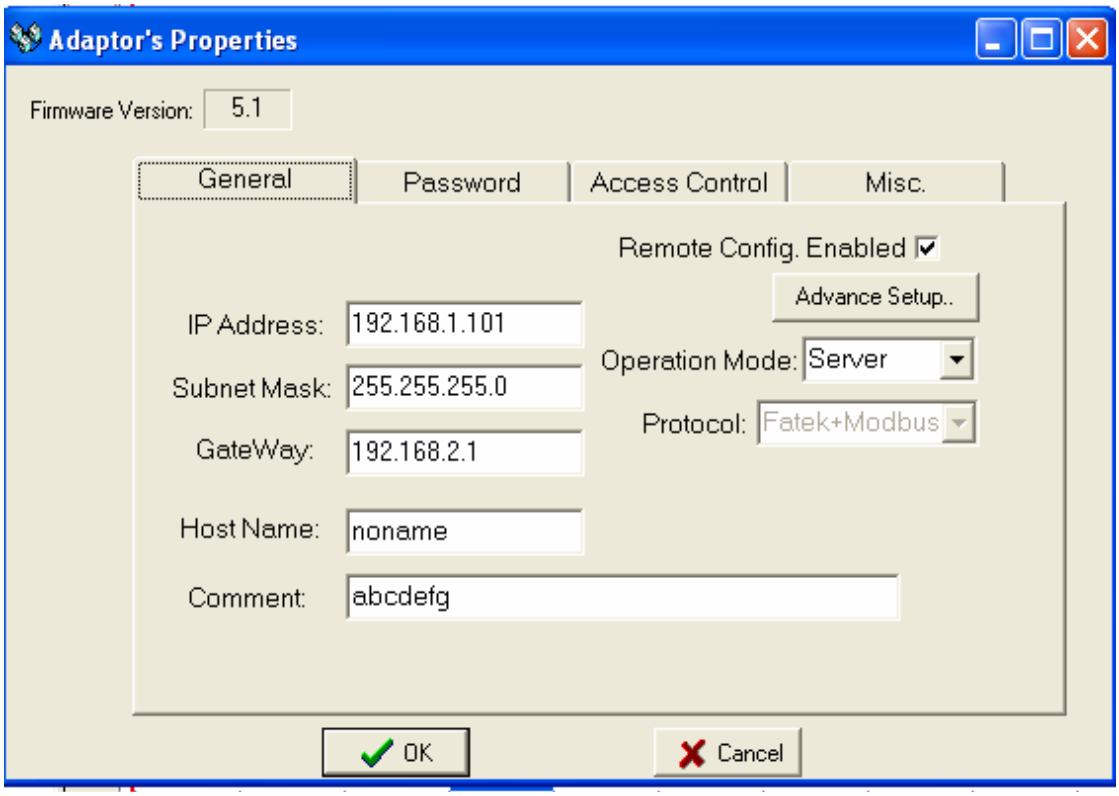
اتصال PLC به HMI از طریق پورت Ethernet

1- ابتدا به PLC مورد نظر وصل می شویم (توسط کابل پروگرمینگ) و مطابق شکل زیر در منوی PLC و در قسمت Setting یک Station Number برای PLC مورد نظر انتخاب کرده و در قسمت Port 2 Parameter نیز BuadRate را بر روی عدد 115200، Parity را Even ، Data Bit را 8 و Stop Bit را بر روی 1 تنظیم می کنیم و همچنین Protocol را Modbud RTU (Slave) انتخاب می کنیم .

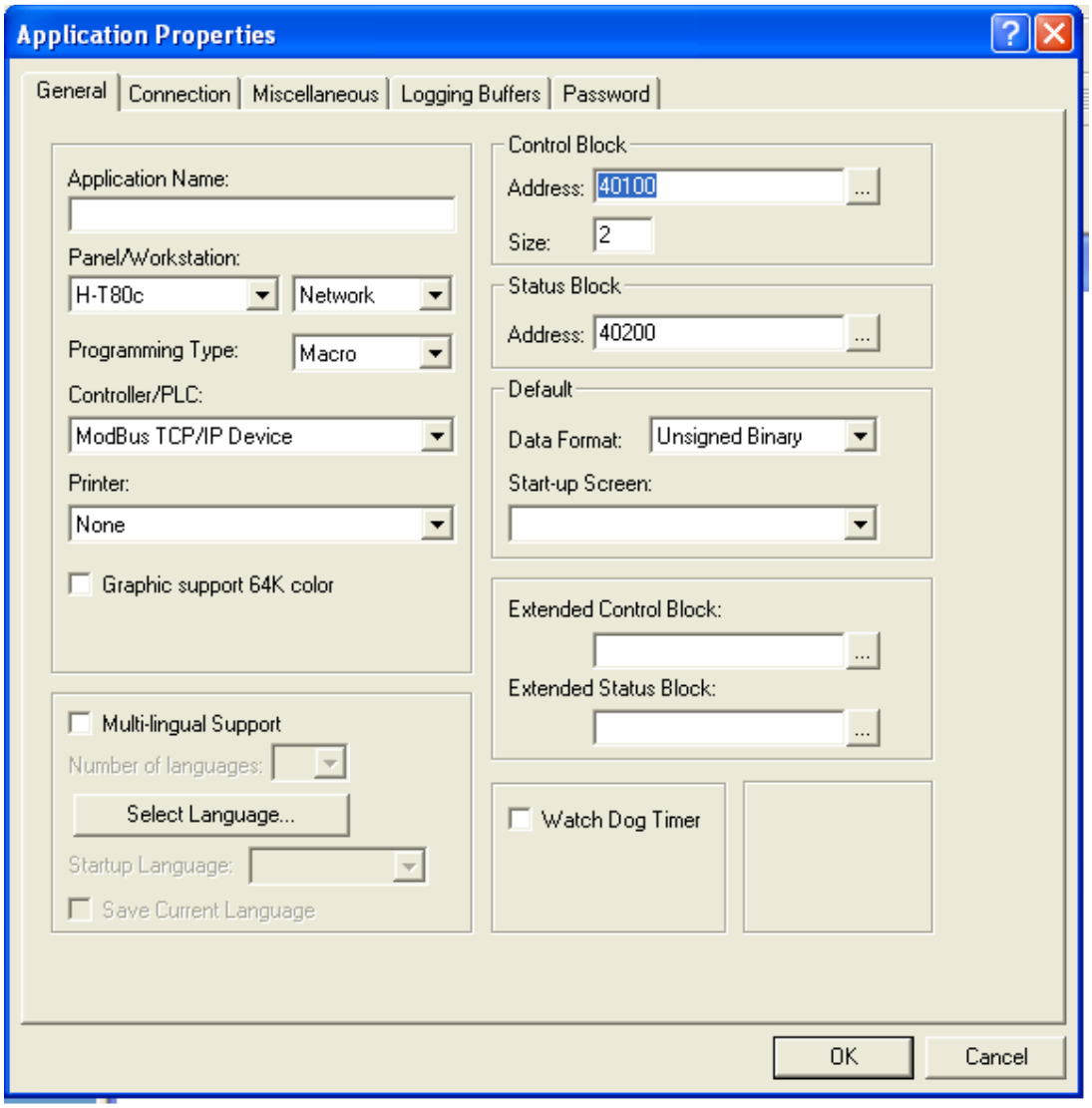




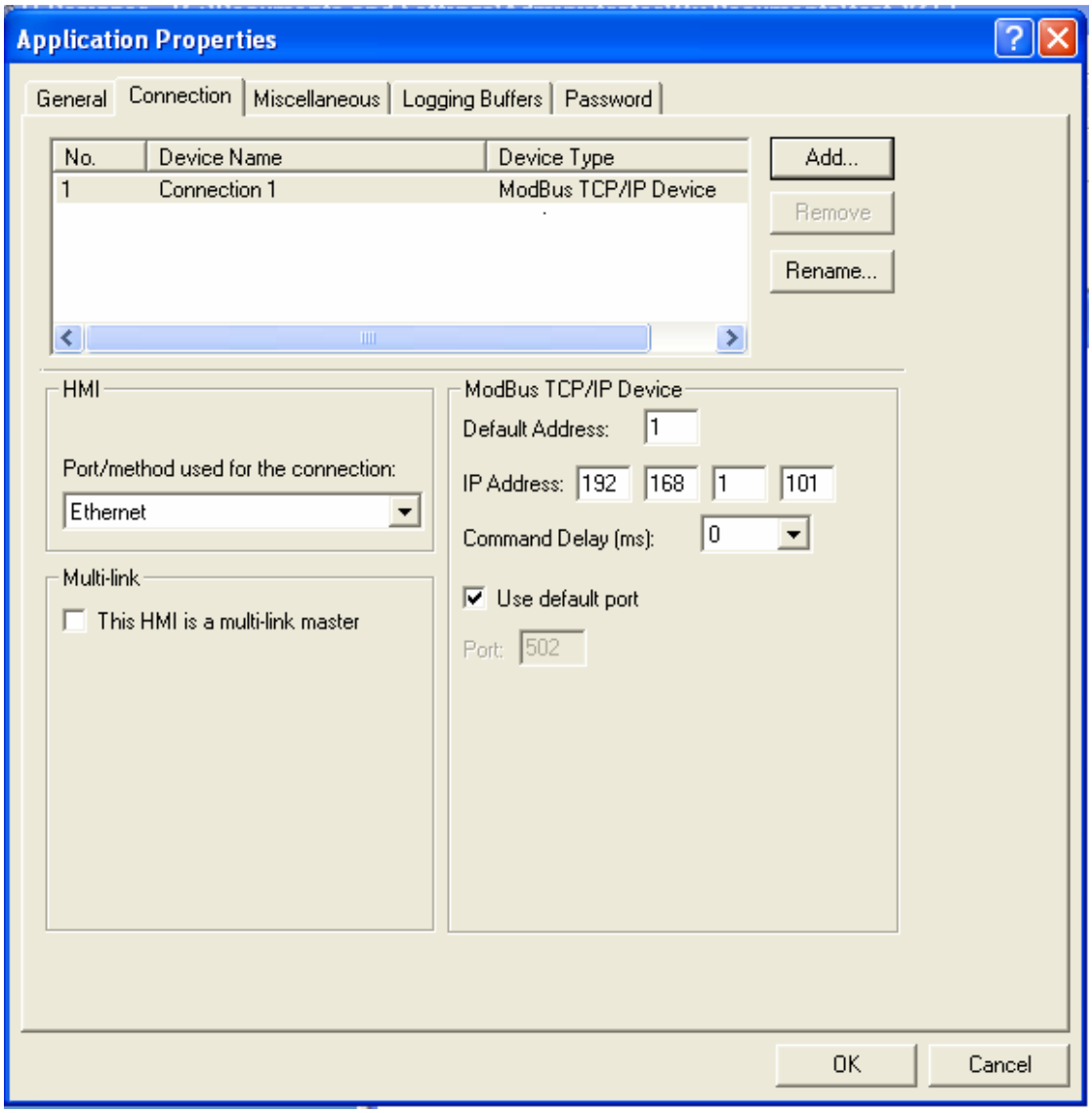
2- در مرحله 2 ابتدا نرم افزار Fatek Ethernet Configuration را باز میکنیم و توسط کابل Ethernet معمولی به کارت CBE متصل می شویم و یکبار Scan میکنیم و بعد از شناسایی ماژول یک بار بر روی آن کلیک می کنیم و در پنجره Adaptor Properties در قسمت IP Address برای ماژول یک IP انتخاب میکنیم و همچنین در پنجره Operation Mode ، مد Server را برای برقراری ارتباط انتخاب می کنیم (در حقیقت HMI به عنوان Client سیستم عمل می کند).



همانطور که در شکل 1 می بینیم ابتدا کنترلر PLC را در قسمت Workstation Properties ، ازنوع Modbus TCP/IP Device انتخاب می کنیم .



سپس در قسمت Connection و در قسمت پورت HMI ، پورت Ethernet را برای برقراری ارتباط با PLC انتخاب می کنیم (همانند شکل زیر) و همچنین در منوی سمت راست و در قسمت Default Address ، Station Number مربوط به PLC را قرار می دهیم (Station Number تعیین شده در مرحله 1) و در قسمت IP Address نیز IP مربوط به PLC را که در مرحله 2 شرح داده شده است را قرار می دهیم .



توجه فرمایید که روش آدرس دهی در این حالت از قوانین Modbus RTU تبعیت می کند.

در انتها برنامه را Compile کرده و در داخل HMI می ریزیم و برای برقراری Connection نیز از کابل Ethernet معمولی استفاده می کنیم .

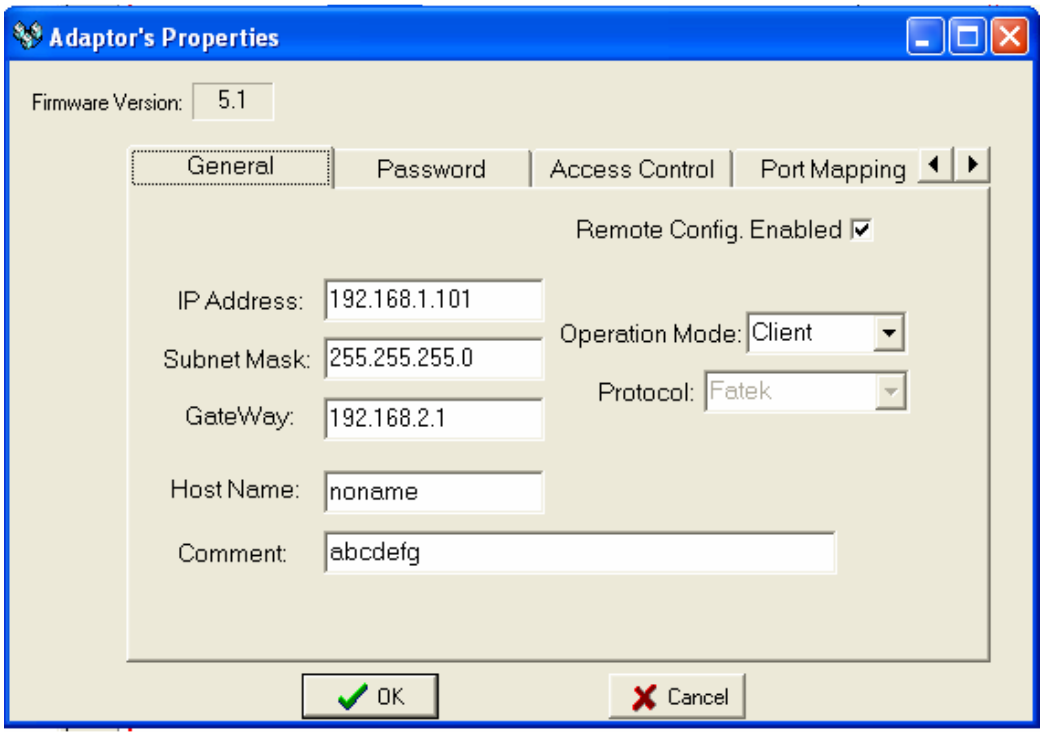
بسمه تعالی

1- در ابتدا به PLC که می خواهیم به عنوان Client یا Master سیستم تعریف شود متصل می شویم و عددی را برای Station Number این PLC انتخاب می کنیم و همینطور سرعت پورت 2 را بر روی عدد 115200 تنظیم می کنیم .

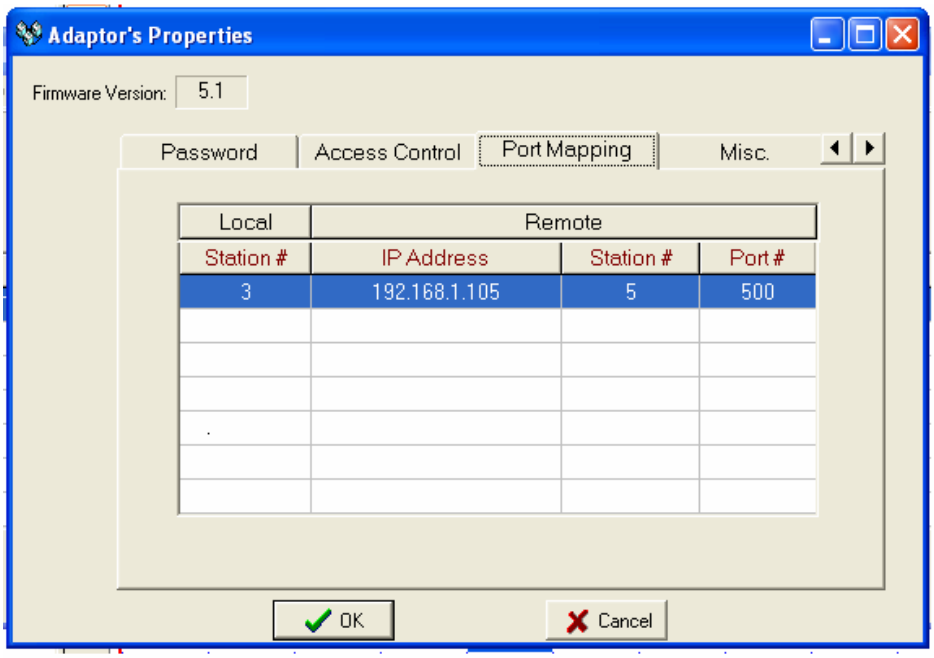
2- سپس به PLC که می خواهیم به عنوان Server یا Slave سیستم تعریف شود متصل می شویم و عددی را برای Station Number این PLC انتخاب می کنیم و همینطور سرعت پورت 4 را (به این دلیل که در این مثال CM25E را به این PLC و ماژول CBE را به Client وصل کرده ایم) نیز بر روی عدد 115200 تنظیم می کنیم .

3- ماژول CBE را که بر روی Client نصب شده است را با کابل Ethernet (این کابل را کابل شماره 1 نامگذاری می کنیم) به PC متصل می کنیم (قبل از آن Firewall مربوط به Windows را خاموش کرده و یک IP نیز برای کامپیوتر تعریف می کنیم)

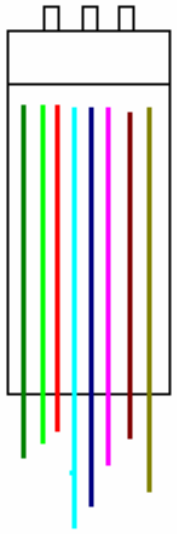
4 - مطابق شکل زیر بعد از زدن دکمه Scan در Ethernet Adapter Configuration یک ماژول شناسایی می شود که با کلیک کردن بر روی آن ماژول پنجره زیر ظاهر شده که در این پنجره یک IP برای PLC که به عنوان Client تعریف شده است (مثلا 192.168.1.101) در نظر می گیریم .



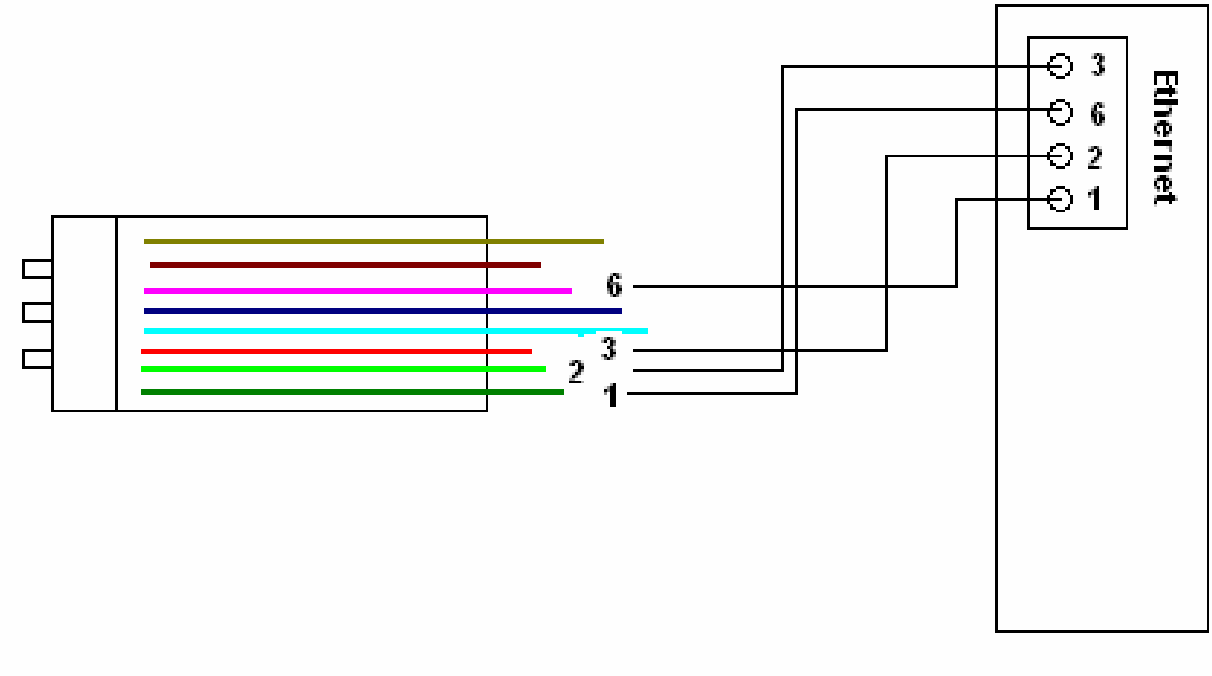
5- مطابق شکل زیر در قسمت Port Mapping و در قسمت Remote یک IP Address و یک Station Number برای PLC که می خواهیم به عنوان Server شبکه تعیین شود ، قرار می دهیم و همچنین در قسمت Local نیز Station Number مربوط به PLC که به عنوان Client در شبکه تعیین شده است را وارد می کنیم .
نکته : Station Num تنظیم شده در قسمت Remote می بایست با Station Number مربوط به PLC ، Slave سیستم که Station Number آن در مرحله 2 تنظیم شده است یکی باشد .



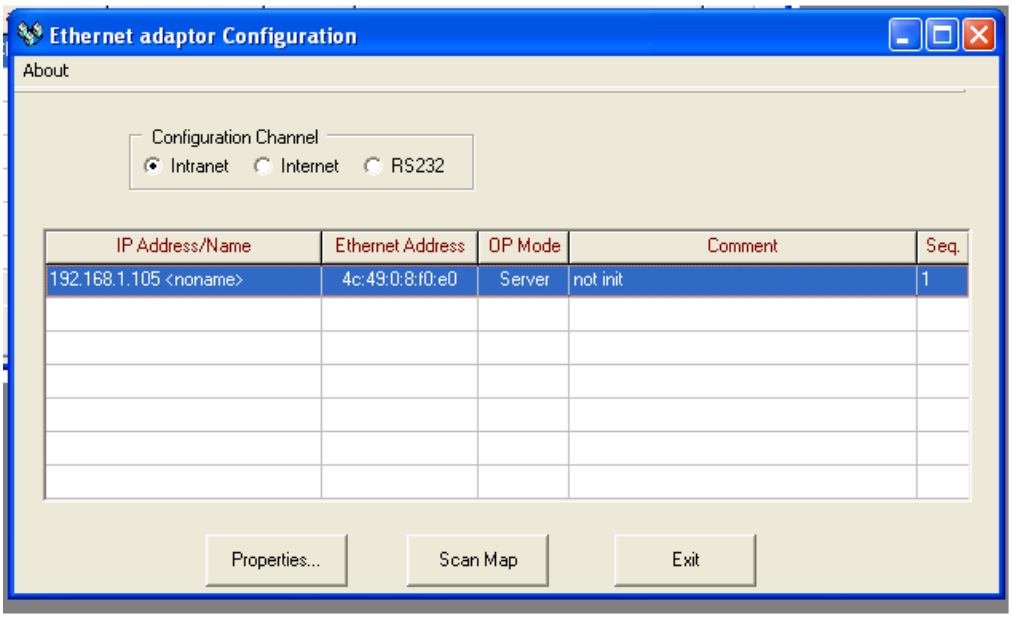
کابلی مطابق با شکل زیرآماده می کنیم .
همانطور که در شکل زیر مشاهده می کنیم ، اگر سوکت کابل Ethernet را به گونه ای در دست بگیریم که زائده آن به سمت پایین قرار بگیرد سیم های بکار رفته در این سوکت از سمت چپ به ترتیب شماره بندی خواهد شد (به این ترتیب سیم سبز رنگ شماره 1 ، سیم سبز روشن شماره 2 ،سیم قرمز شماره 3 و ...)



با ترکیب بالا :
- سیم شماره 1 سوکت به پایه شماره 6 سوکت Ethernet ماژول CM25E وصل خواهد شد .
- سیم شماره 2 سوکت به پایه شماره 3 سوکت Ethernet ماژول CM25E وصل خواهد شد .
- سیم شماره 3 سوکت به پایه شماره 2 سوکت Ethernet ماژول CM25E وصل خواهد شد .
- سیم شماره 6 سوکت به پایه شماره 1 سوکت Ethernet ماژول CM25E وصل خواهد شد .



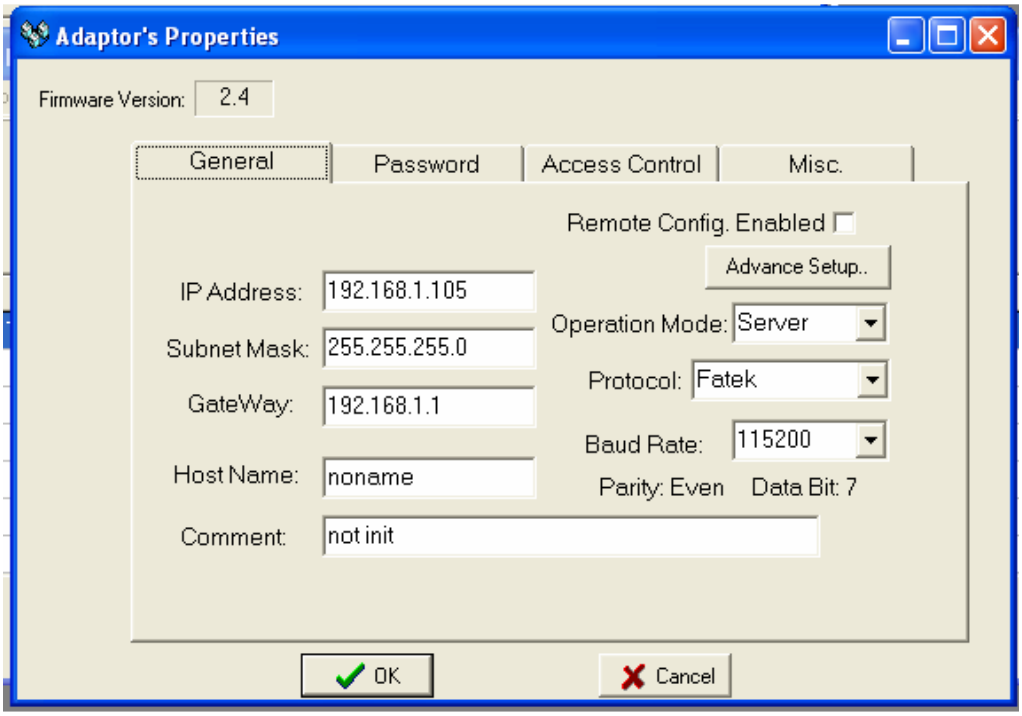
4- کابل را ساخته و به ماژول CM25E متصل می کنیم و سر دیگر آن را در داخل سوکت PC یا Laptop قرار می دهیم و همانند مرحله 2 نرم افزار Ethernet Adapter Configuration را باز کرده و یک بار Scan می کنیم و به این ترتیب به ماژول CM25E متصل می شویم (همانند شکل زیر)



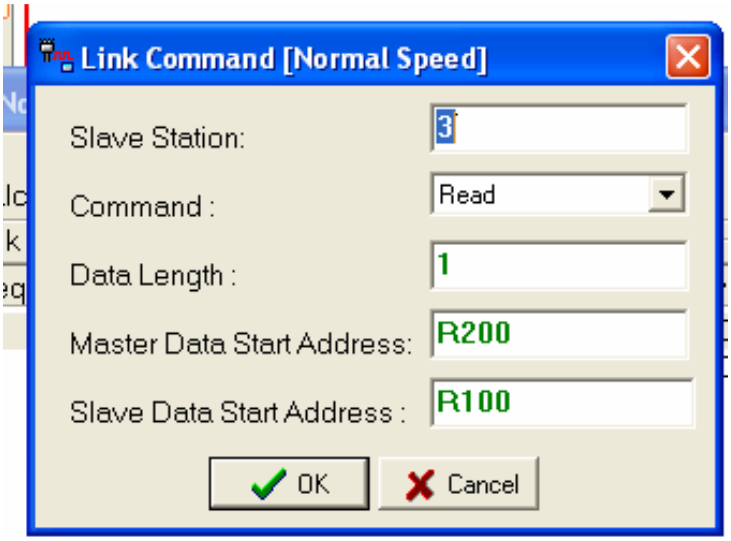
5- با کلیک بر روی IP شناسایی شده پنجره زیر ظاهر خواهد شد و کافی است در این پنجره و در قسمت IP Address ، IP مربوط به Server سیستم را که در مرحله 5 ودر قسمت Remote تنظیم شده است را وارد نماییم .

نکات قابل توجه در این مرحله عبارتند از :

- Operation Mode می بایست در مد Server قرار گیرد
- Baud Rate می بایست بر روی 115200 قرار گیرد
- در این مد پنجره Port Mapping وجود ندارد



6- تنظیمات Port های Ethernet به پایان رسید حال کافی است توسط نرم افزار WinProladder با PLC که به عنوان Client تعریف شده است ارتباط برقرار کرده و در آن محیط یک جدول Link Table تعریف نمود (همانند شکل زیر) که در این جدول و در قسمت Slave Station ، Station تنظیم شده در مرحله 5 و در قسمت Local را قرار می دهیم .



بعد از تنظیم جدول کافی است که جدول مربوطه را با فانکشن F151 بر روی Bus ارسال نماییم (همانند شکل زیر) و با فعال کردن تابع فوق و ارتباط دو ماژول CBE و CM25E توسط کابل 2 تبادل اطلاعات آغاز خواهد شد .