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

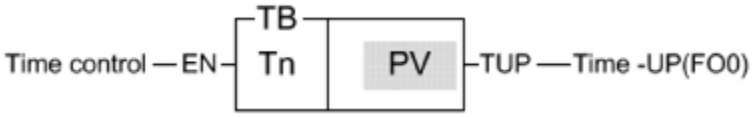
ریست کردن تایمر نگهبان	RSWDT	91
خواندن شمارنده سرعت بالا نرم افزاری	HSCTR	92
بارگذاری شمارنده های سرعت بالا	HSCTW	93
اطلاعاتی را که به صورت ASCII کد شده اند و از S شروع می شوند را به Port1 منتقل می کند	ASCWR	94
تابع شیب برای خروجی دیجیتال به انالوگ	RAMP	95
"رزرو"	-	96
"رزرو"	-	97
Tracking type ramp function 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

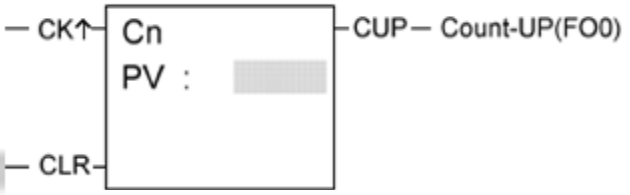
T50~T199 : 0.1s

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.

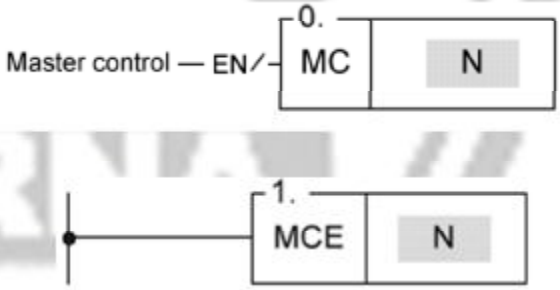
PROGRAM END

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

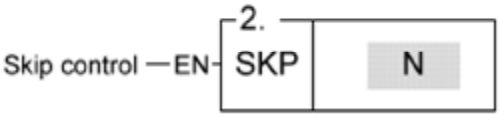
MC:Master Control Loop Start

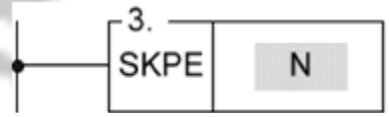
MCE:Master control Loop End



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.

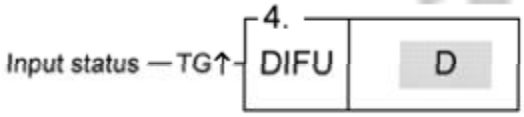
Function 2&3: SKIP START & SKIP END





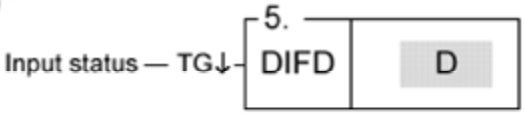
در کل 128 حلقه SKP وجود دارد (N=0~127) . هر تابع SKP با یک تابع SKPE مرتبط است و شماره حلقه آنها N یکسان می باشد . باید توجه کرد که تابع SKPE بعد از SKP قرار بگیرد. هرگاه "EN" غیر فعال شود، تمام توابع و برنامه هایی که زیر حلقه SKP نوشته می شوند اجرا می شود. و چنانچه "EN" فعال شود، تمام توابع و برنامه 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



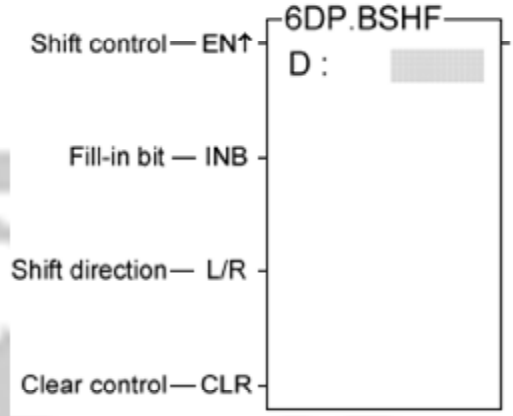
این تابع باعث می شود که هرگاه ورودی TG فعال شد ،خروجی به اندازه Scan Time فعال باشد.

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

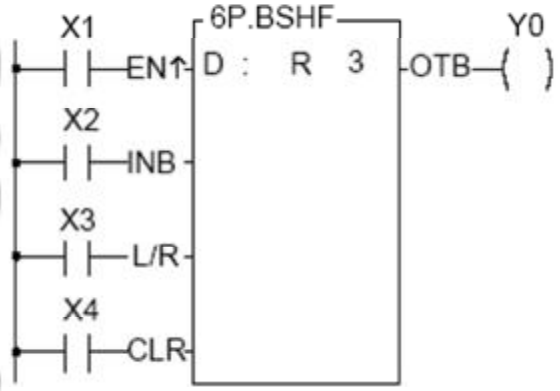


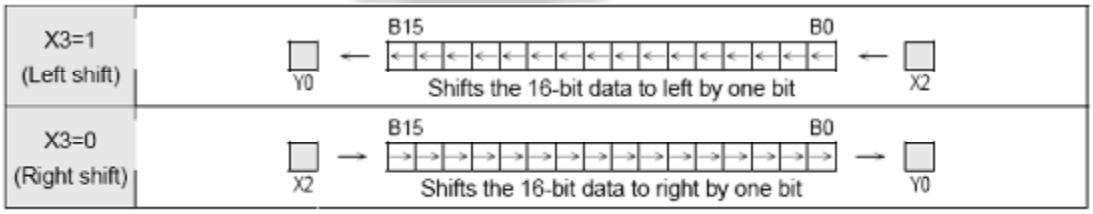
با این تابع می توان رجیستر مورد نظر را **1** بیت به چپ یا راست ، شیفت داد. بدین منظور **CLR** باید **0** باشد.

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

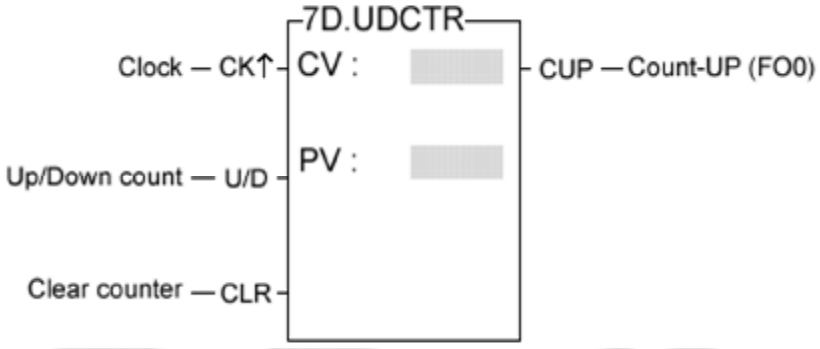
با **"INB"** می توان تعیین نمود که بیت ایجاد شده پس از اعمال شیفت، **0** یا **1** بشود.

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.

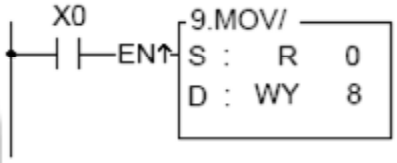
Function 8.MOVE

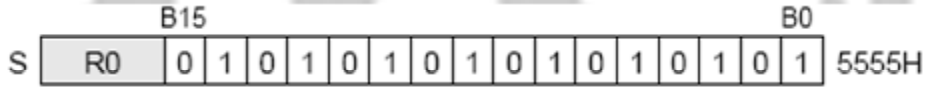
هرگاه **"EN"** فعال شود،اطلاعات موجود در رجیستر **S** به **D** کپی می شود.

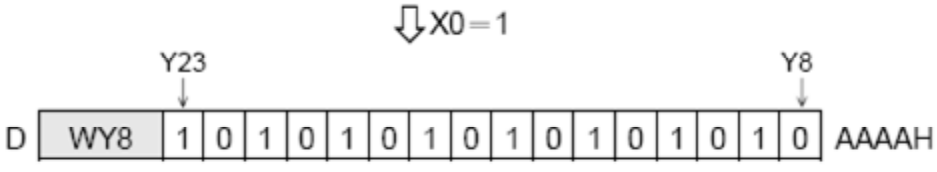
Function 9. MOVE INVERSE

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.

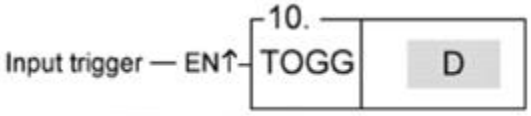
مثال:





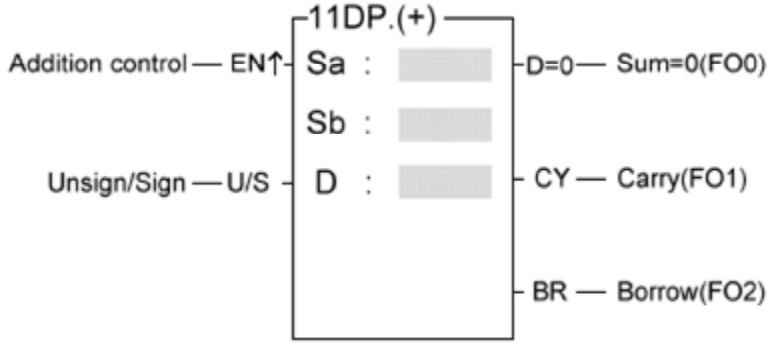


Function 10.TOGGLE SWITCH



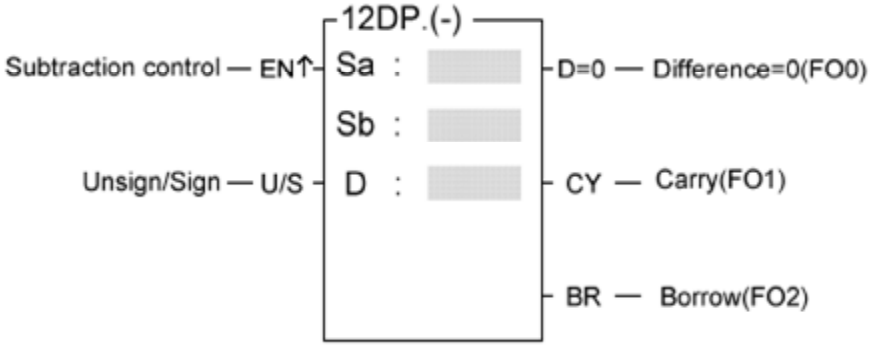
با فعال شدن "EN" ،بیت نوشته شده در D ، اگر 1 باشد 0 و اگر 0 باشد 1 می شود .

Function 11.ADDITION



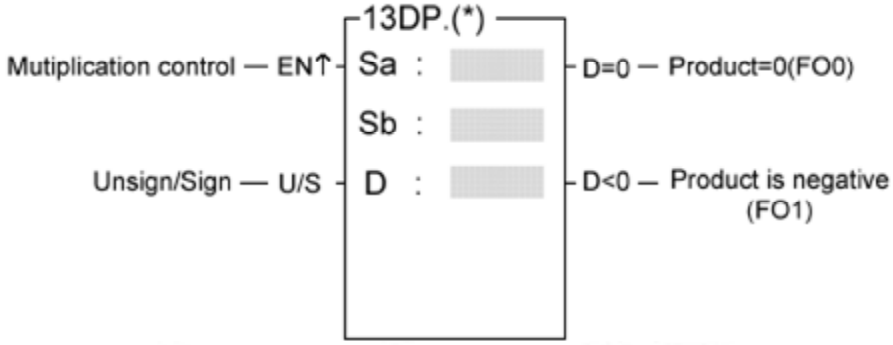
با فعال شدن "EN" ، داده های موجود در Sa و Sb با هم جمع شده و در D ریخته می شود. Sa+Sb=D

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

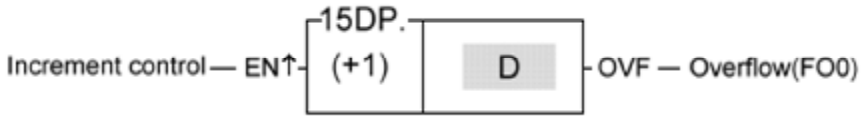


با فعال شدن **"EN"** ، داده های موجود در **Sa**، ضرب در **Sb** شده و نتیجه در **D** ریخته می شود. **Sa×Sb=D**

Function 14.DIVISION

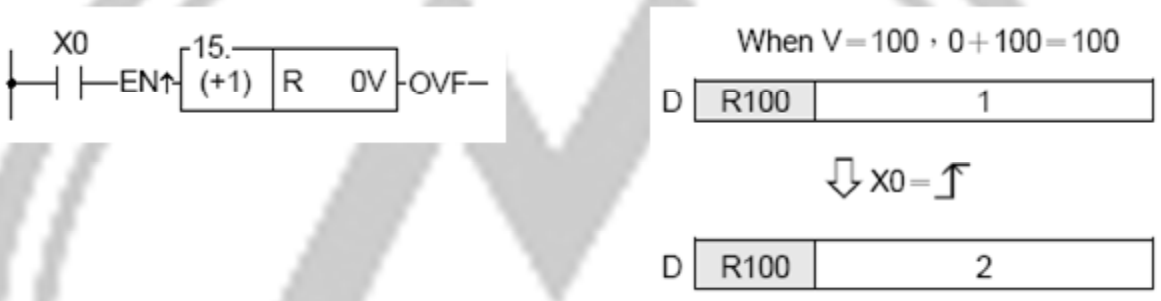
با فعال شدن **"EN"** ، داده های موجود در **Sa**، تقسیم بر **Sb** شده و و نتیجه در **D** ریخته می شود. چنانچه از این فانکشن در حالت **16** بیتی استفاده شود باقیمانده این تقسیم در رجیستر **D+1** قرار خواهد گرفت و چنانچه از این فانکشن در حالت **32** بیتی استفاده شود باقیمانده این تقسیم در رجیستر **D+2**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

Function 15.INCREMENT

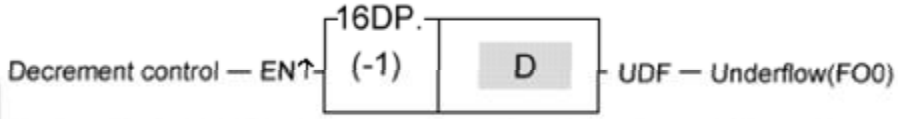


هرگاه "EN" از 0 به 1 تغییر کند ، به مقدار رجیستر D ، "1" واحد اضافه می شود . اگر این افزایش ، باعث از محدوده (range) خارج شدن مقدار D شود، خروجی "OVF" فعال شده و مقدار D منفی می شود.

مثال:



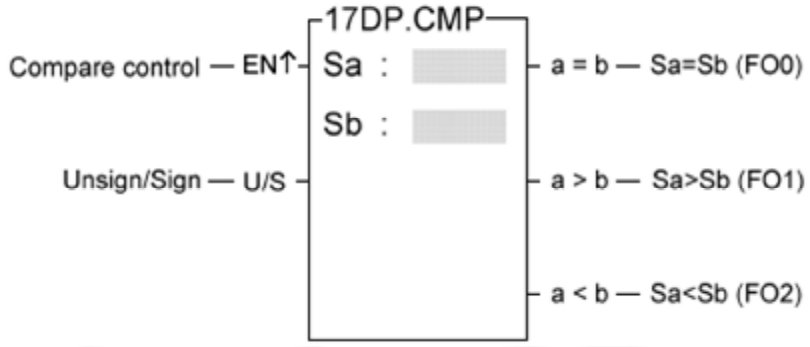
Function 16.DECREMENT



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.

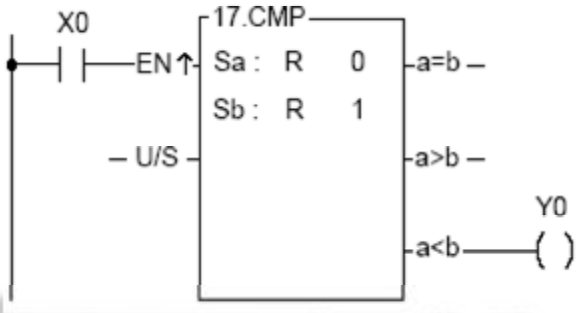
مثال:

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.

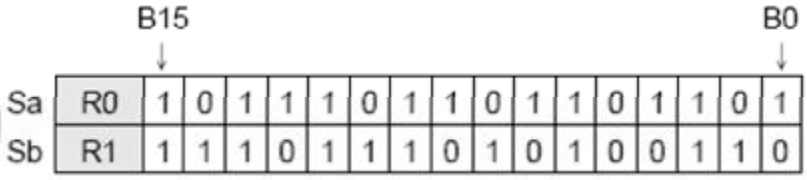
مثال:

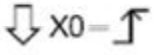


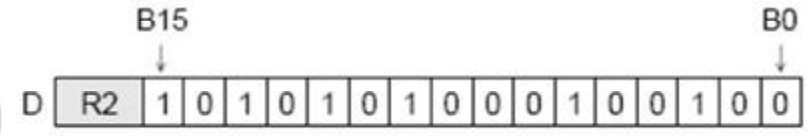
Function 18.LOGICAL AND

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:



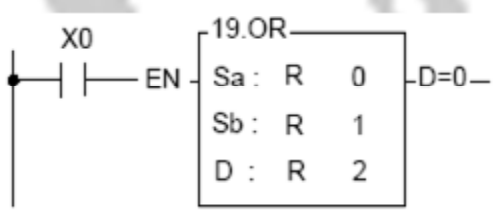


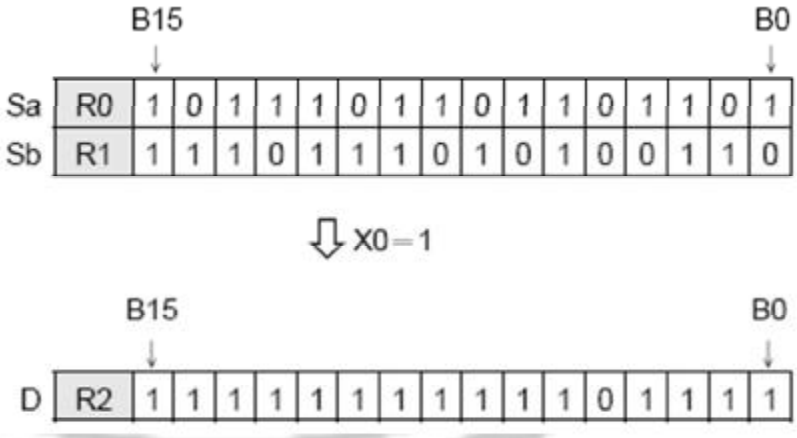


Function 19.LOGICAL OR

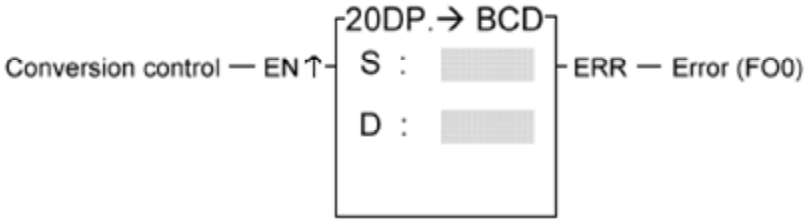
هرگاه "EN" از 0 به 1 تغییر کند، بیت های موجود در Sa وSb را با هم OR کرده و نتیجه را در D می ریزد. هرگاه تمام بیت های D ، صفر شود ، خروجی "D=0" فعال می شود .

مثال:



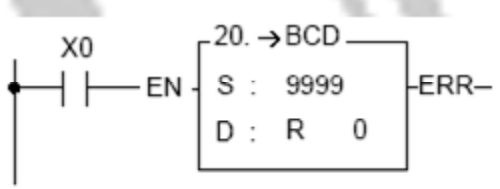


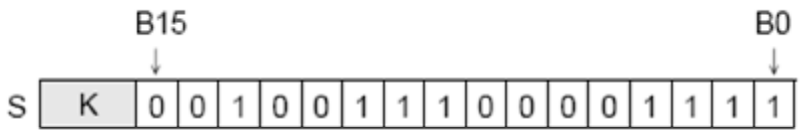
Function 20.BIN TO BCD CONVERSION

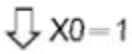


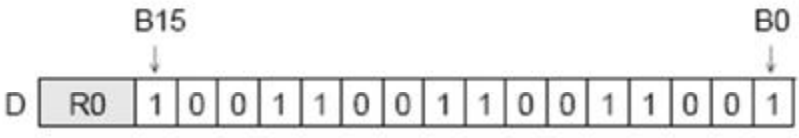
هرگاه "EN" از 0 به 1 تغییر کند، داده های موجود در S را که به صورت کد باینری است، به صورت BCD درآورده و در D می ریزد. اگر داده S در BCD Range نباشد،خروجی "ERR" فعال می شود و اطلاعات قبلی D بدون تغییر باقی می مانند.

مثال:

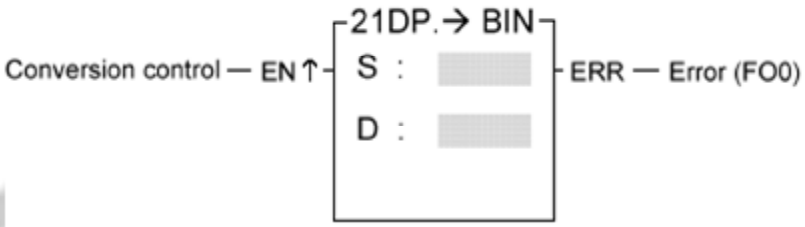






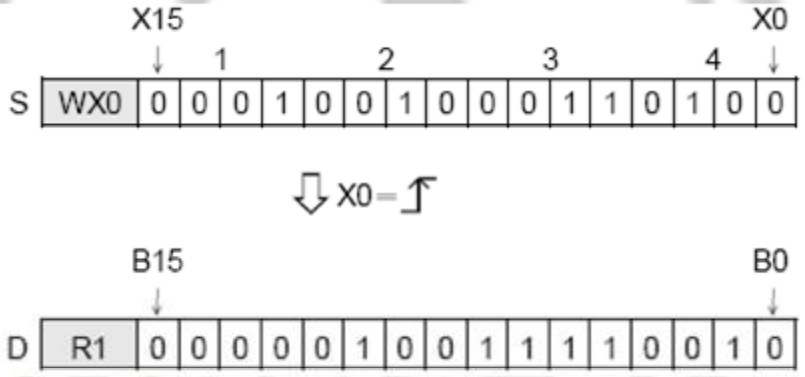


Function 21.BCD TO BIN CONVERSION



برعکس **BIN TO BCD** عمل می کند.

مثال:



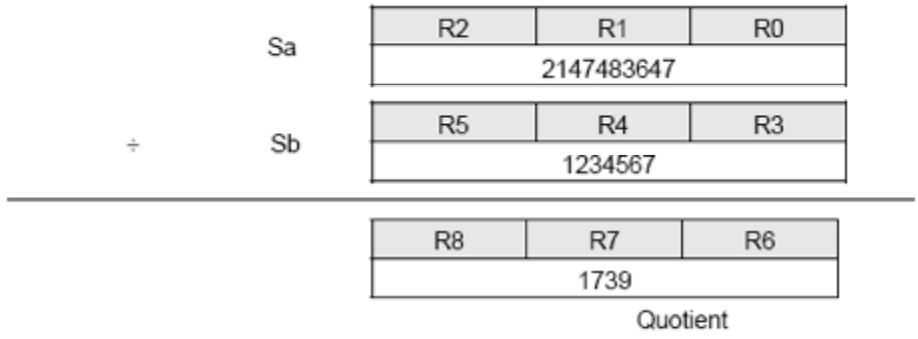
Function 22.BREAK

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.

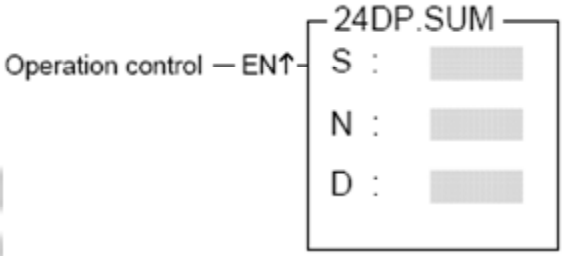
Function 23.48-BIT DIVISON

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:

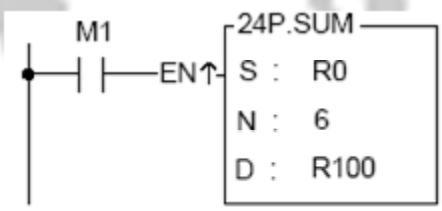


Function 24.SUM



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.

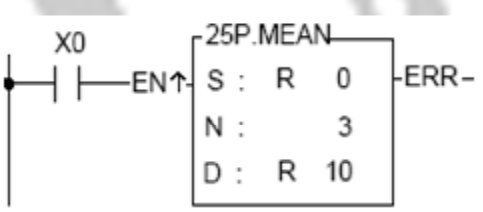
مثال:

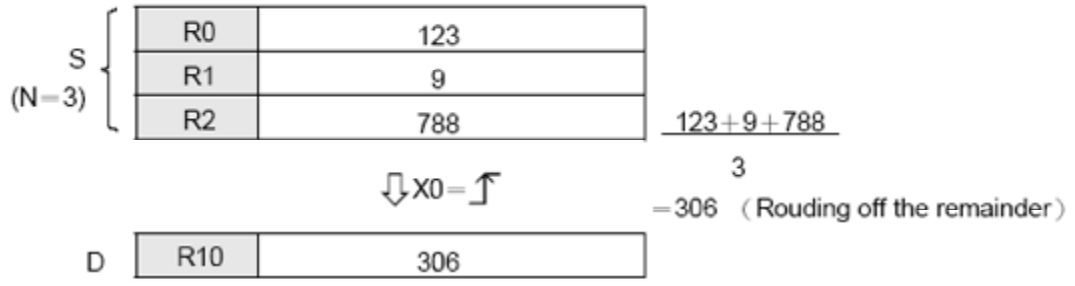


Function 25.MEAN

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:



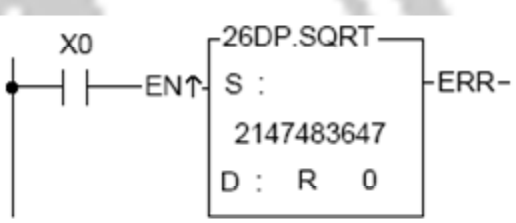


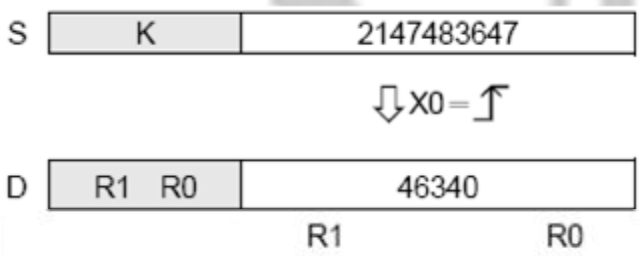
Function 26.SQUARE ROOT

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:



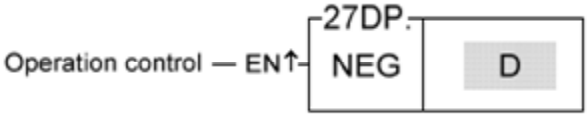


$$\sqrt{2147483647} = 46340.95$$

↑

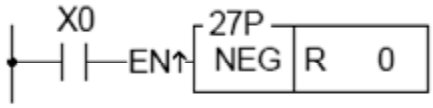
rounding off

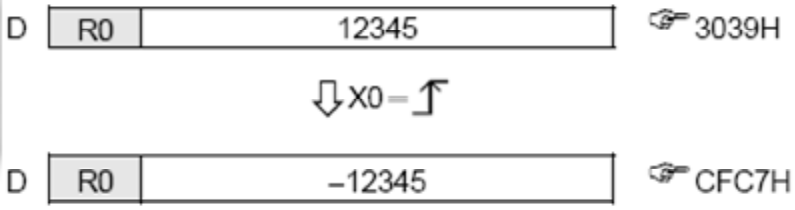
Function 27.NEGATION



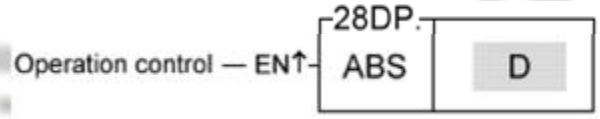
هرگاه "EN" از 0 به 1 تغییر کند، مقدار D منفی شده و دوباره در همان رجیستر ریخته می شود.

مثال:



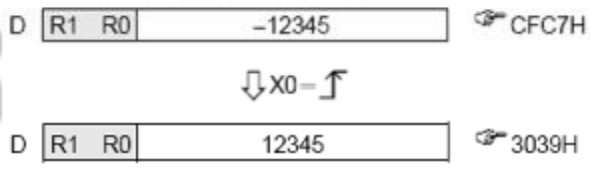


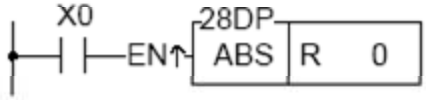
Function 28.ABSOLUTE



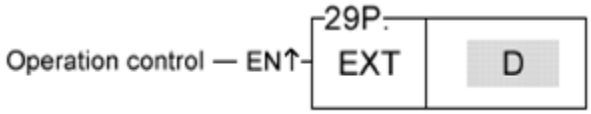
این تابع، قدر مطلق مقدار موجود در D را محاسبه و دوباره در D می ریزد.

مثال:



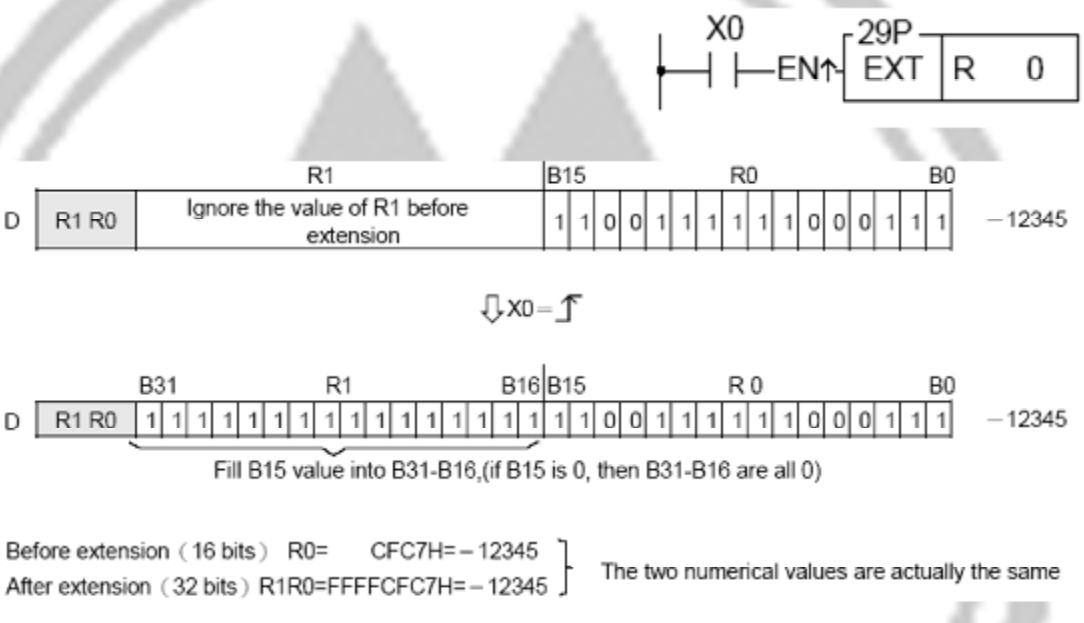


Function 29.SIGN EXTENSION



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:

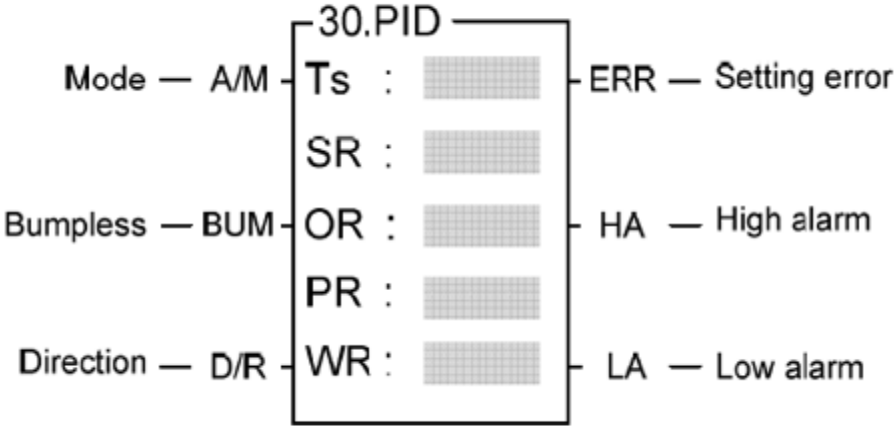


Function 30.PID GENERAL

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)

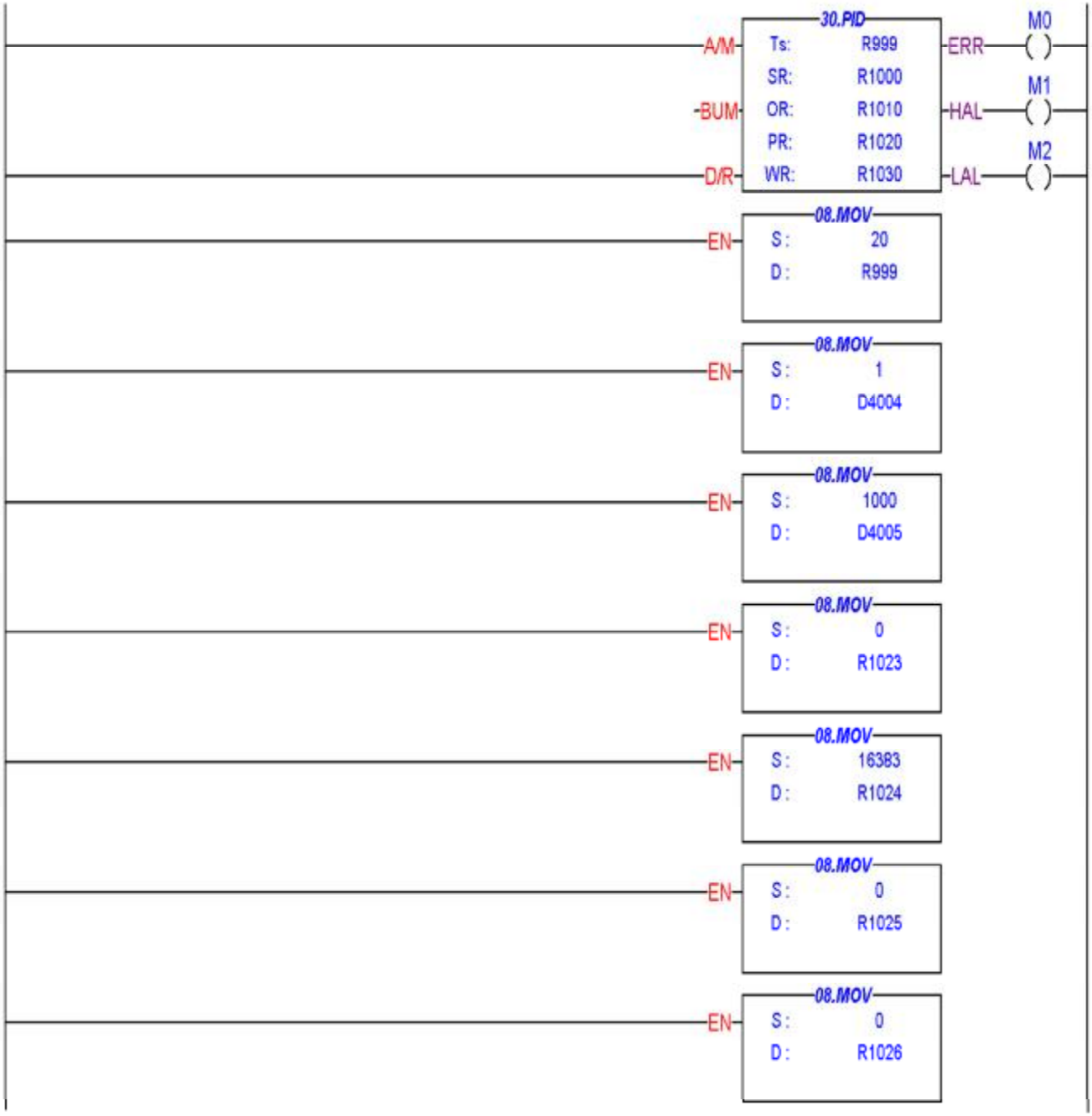
Bias : Control output offset (range: 0~16380)

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

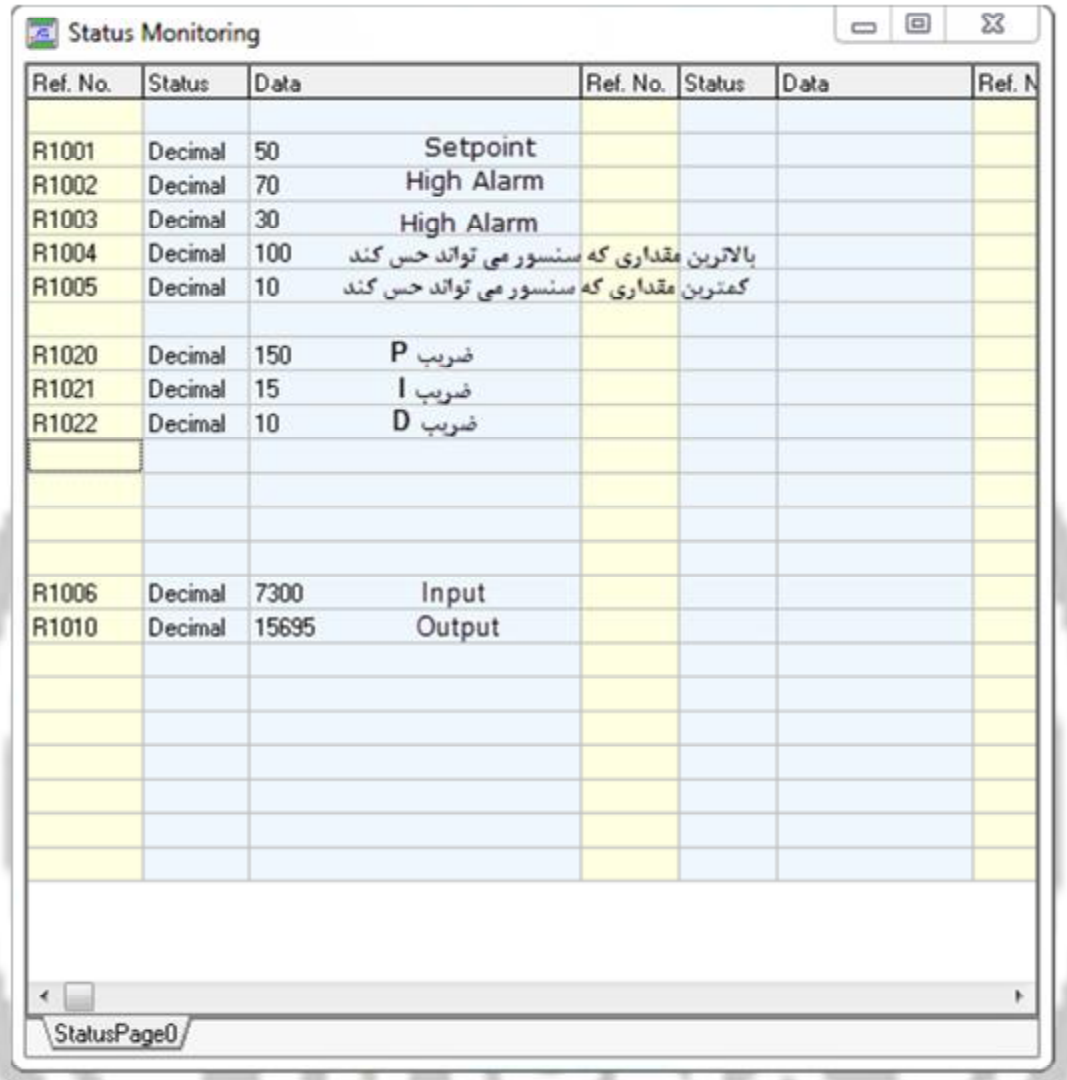
PR+2	ضریب مشتق Td 0<Td<9999
PR+3	Control output offset (range: 0~16380)
PR+4	مصارف خاص ، مقدار 16383 قرار داده شود
PR+5	مصارف خاص ، مقدار 0 قرار داده شود
PR+6	اگر مقدار 0 قرار داده شود بمعنای PID استاندارد و اگر مقدار 1 قرار داده شود بمعنای PI (در مواقعی که سیستم به پایداری نرسد از این گزینه استفاده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

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

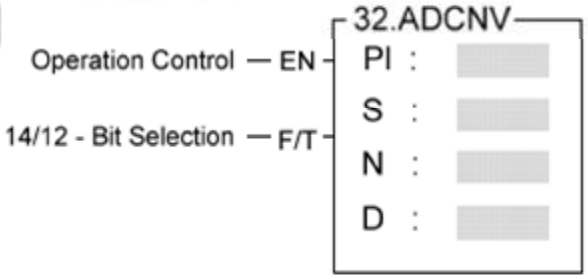
مثال :



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 :

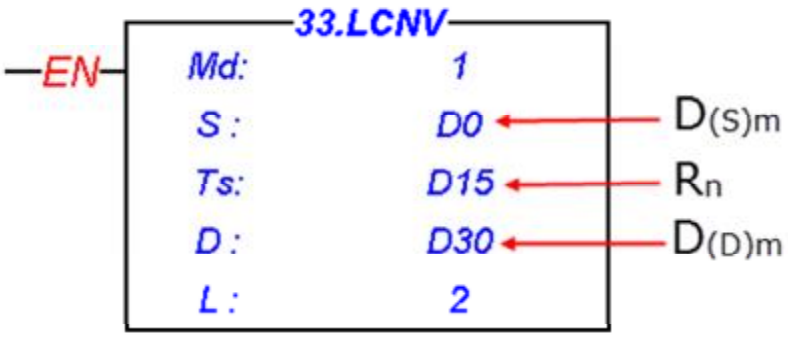
مد صفر : تبدیل عدد ورودی از بازه [A,B] به عدد خروجی بین بازه [C,D]
مد 1 : در این مد می توان چندین تبدیل بازه هابه یکدیگر را با نوشتن فقط یک فانکشن اجرا کرد
مد 2 : تبدیل به تابع خطی درجه 1 D=(A/B)D+C
مد 3 : در این مد می توان چندین تابع خطی درجه 1 را با نوشتن فقط یک فانکشن اجرا کرد

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 :



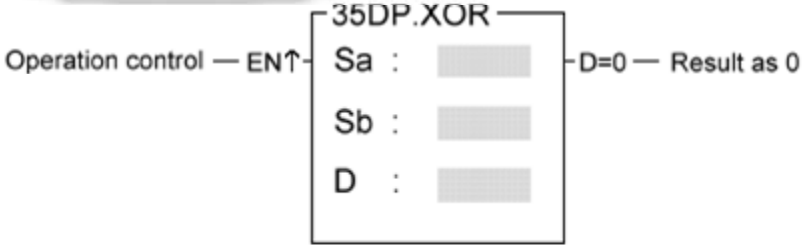
به ازای ورودی های زیر (S)	Ts برای مد صفر	خروجی D
$D_{(S)m}$ →	$[R_n,R_{n+1}] \rightarrow [R_{n+2}, R_{n+3}]$	→ $D_{(D)m}$

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

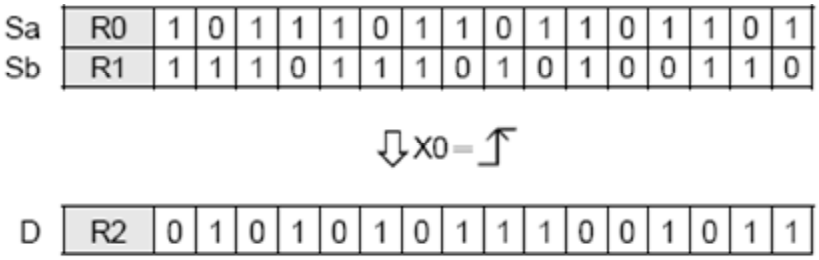
به ازای ورودی های زیر (S)	Ts برای مد 2	خروجی D
$D_{(S)m+1}$ →	$A1=R_n \quad B1=R_{n+1} \quad C1=R_{n+2}$	→ $D_{(D)m+1}=(A1/B1)D+C1$

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

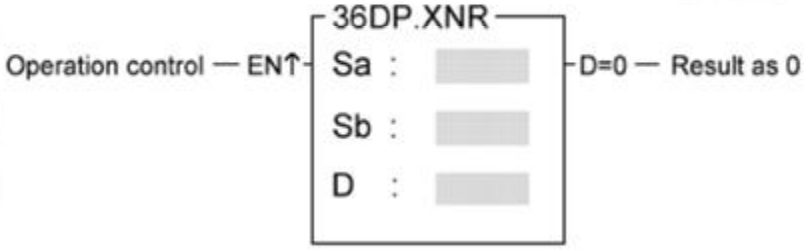
Function 35.EXCLUSIVE OR (XOR)



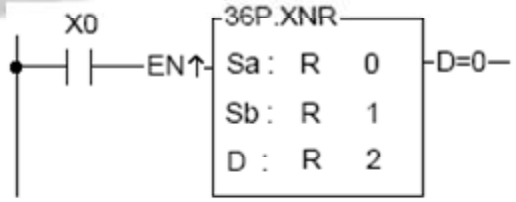
هرگاه "EN" از 0 به 1 تغییر کند،بیت های موجود در Sa و Sb را با هم XOR کرده و نتیجه را در D می ریزد. عملکرد XOR به این صورت است که هرگاه بیت های متناظر Sa و Sb همانند باشند ، بیت متناظر در D ، 0 می شود و هرگاه یکی 1 و آن یکی 0 باشد ، نتیجه در D ، 1 می شود . هرگاه تمام بیت ها ی D ، صفر شود ، خروجی "D=0"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

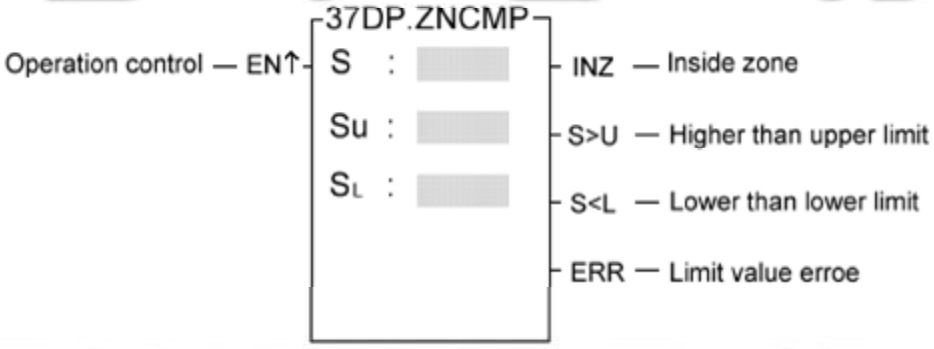


Sa	R0	1	0	1	1	1	0	1	1	0	1	1	0	1	1	0	1
Sb	R1	1	1	1	0	1	1	1	0	1	0	1	0	0	1	1	0

X0

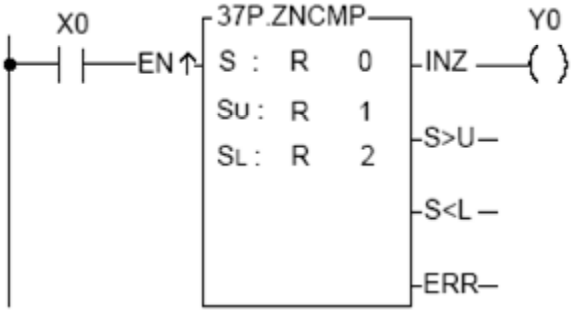
D	R2	1	0	1	0	1	0	1	0	0	0	1	1	0	1	0	0

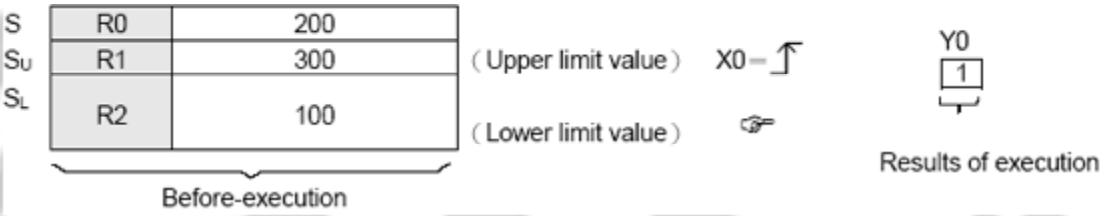
Function 37.ZONE COMPARE



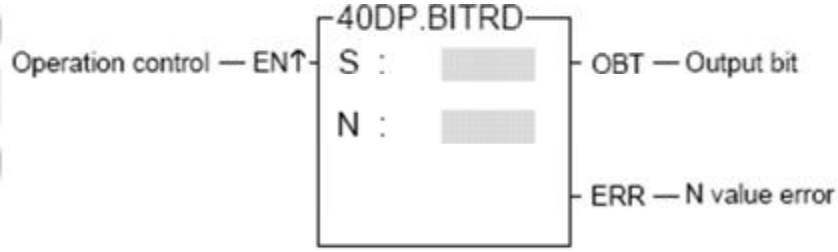
هرگاه "EN" از 0 به 1 تغییر کند، مقدار S با مقدار S_U و S_L مقایسه می شود. اگر بین این دو باشد، خروجی "INZ" فعال می شود . اگر بزرگتر از S_U باشد ، خروجی "S>U" فعال می شود . اگر کوچکتر از S_L باشد ، خروجی "S<L" فعال می شود. اگر $S_L > S_U$ باشد ،ERROR داده خواهد شد.

مثال:





Function 40.BIT READ



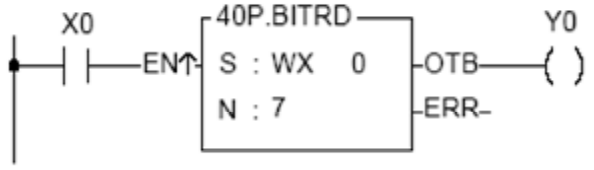
داده های مورد نظر ، در S ریخته می شود. هرگاه "EN" از 0 به 1 تغییر کند: بیت N اُم از داده ی S در خروجی "OBT" قرار می گیرد.

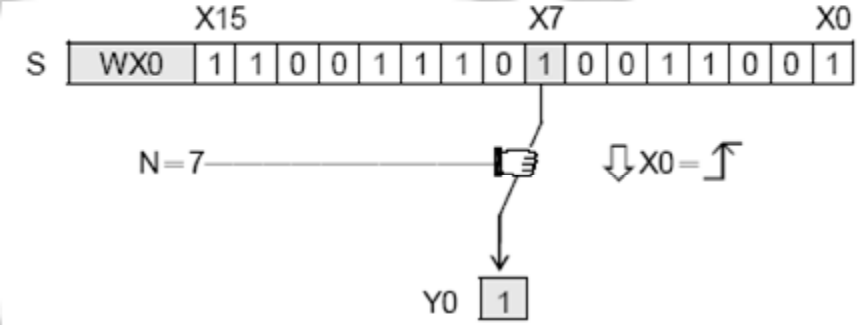
اگر داده ی S ، 16 بیتی باشد : N:0~15

اگر داده ی S ، 32 بیتی باشد : N:0~31

در غیر این صورت error می دهد.

مثال:





Function 41.BIT WRITE

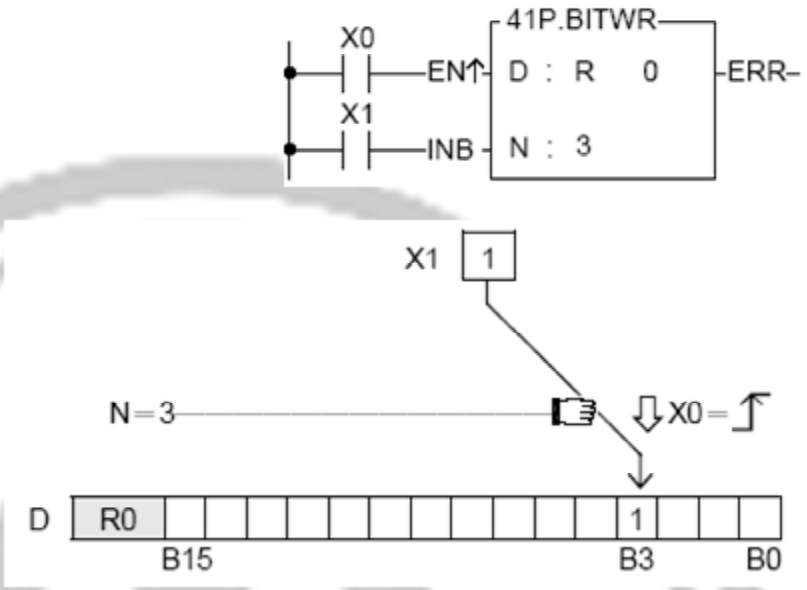
هرگاه "EN" از 0 به 1 تغییر کند: مقدار بیت ورودی "INB" در بیت N اُم رجیستر D ریخته می شود.

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:

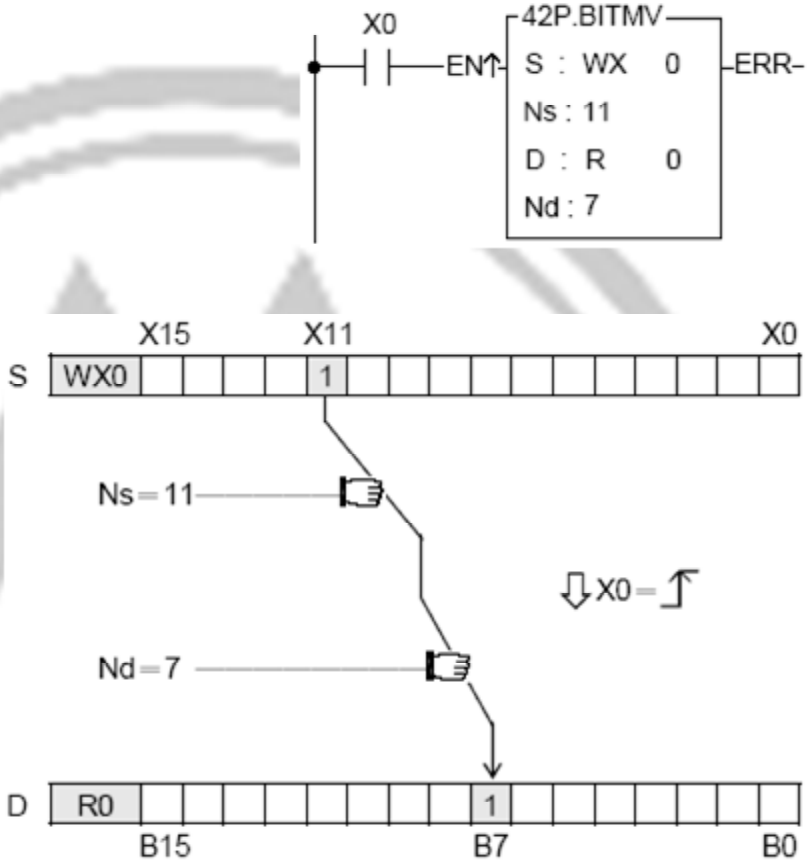


تمام بیت ها غیر از **B3** دست نخورده باقی می مانند.

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:

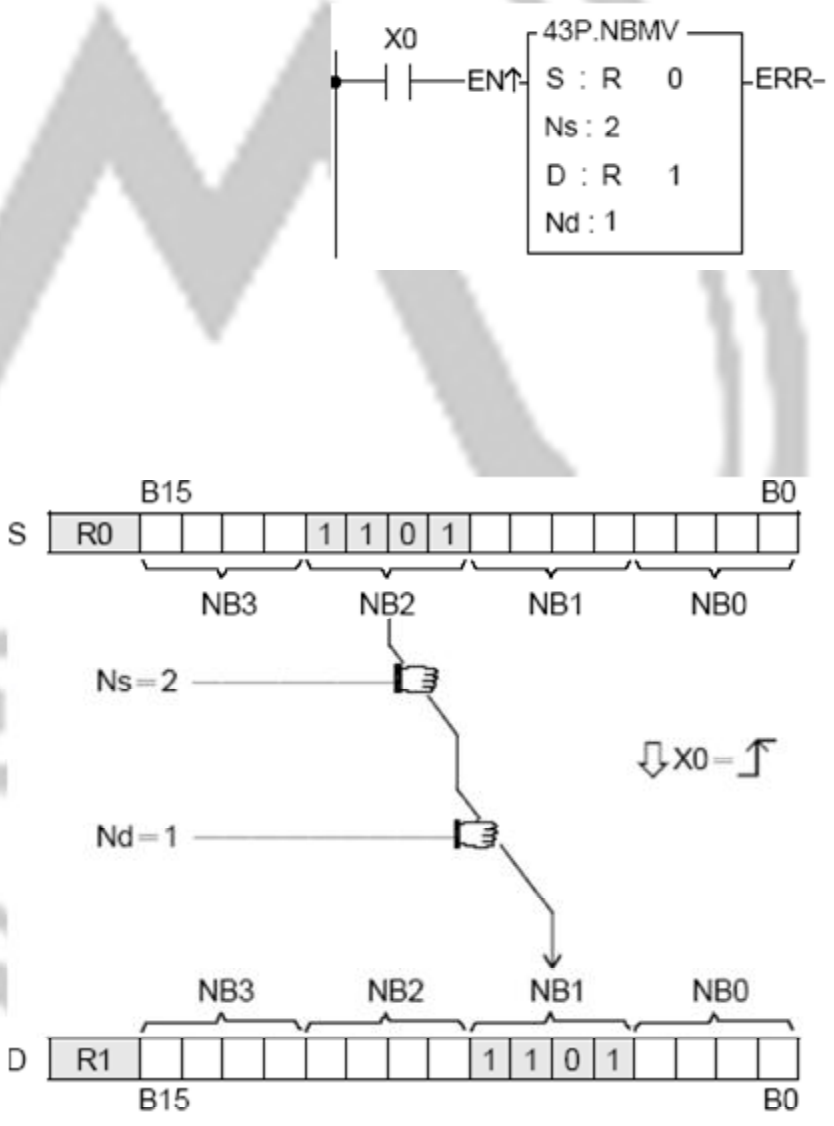


Function 43.NIBBLE MOVE

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:

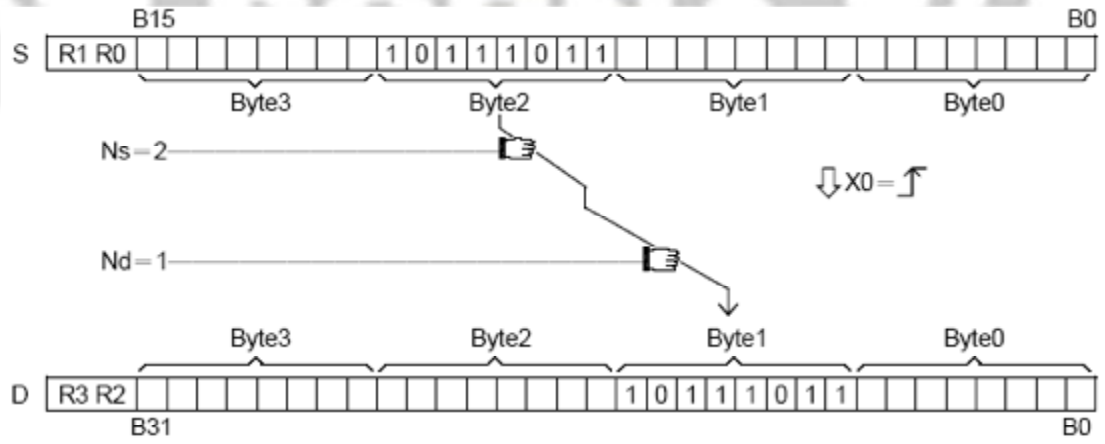


Function 44.BYTE MOVE

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.

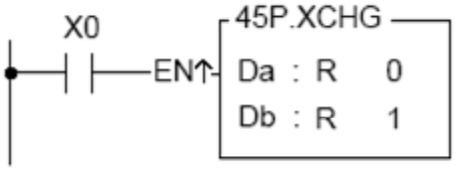
مثال:



Function 45.EXCHANGE

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

مثال:

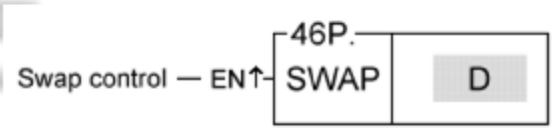


		B15															B0
Da	R0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Db	R1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

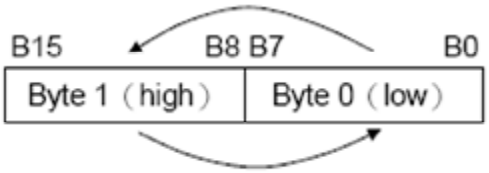
X0 ↓↑

		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

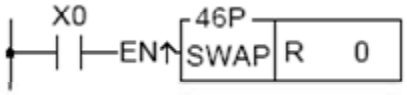
Function 46.BYTE SWAP

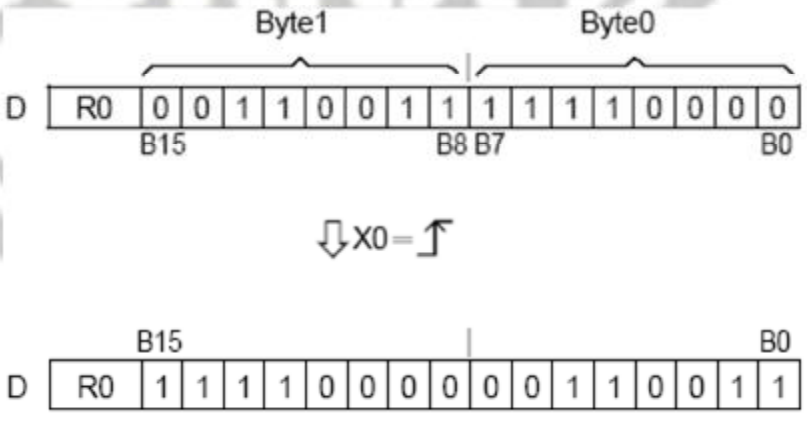


در D یک رجیستر 16 بیتی قرار می گیرد که با فعال شدن "EN"، بایت بالا (High Byte) و بایت پایین (Low Byte) آن با هم عوض می شوند.

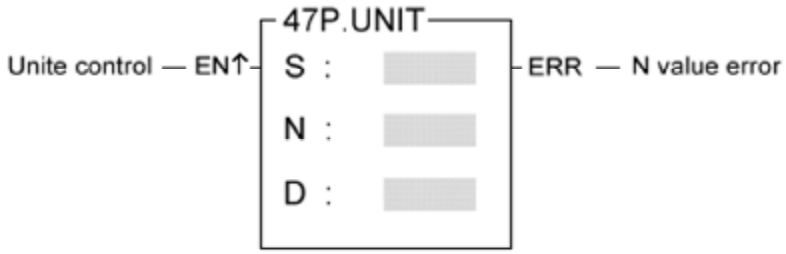


مثال:





Function 47.NIBBLE UNITE

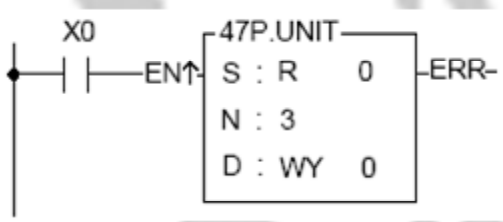


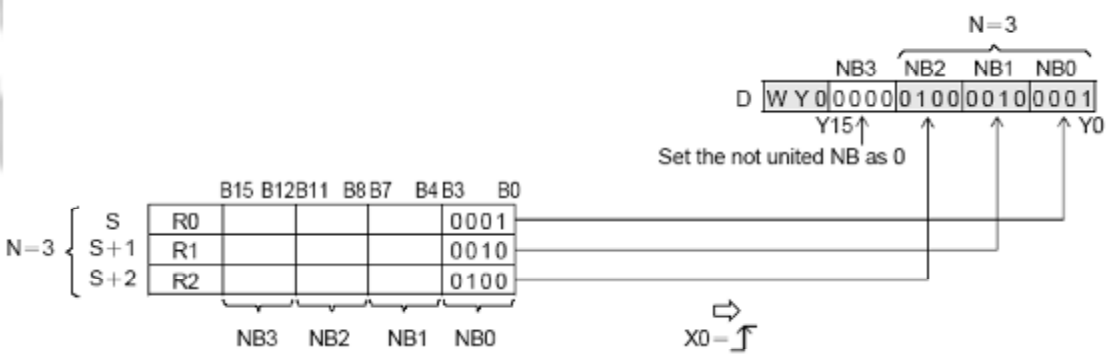
در **S** اولین رجیستر **16** بیتی مورد نظر قرار می گیرد . این تابع نیبل های پایین **N** تعداد از رجیستر ها را به ترتیب در رجیستر **D** قرار می دهد.

***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

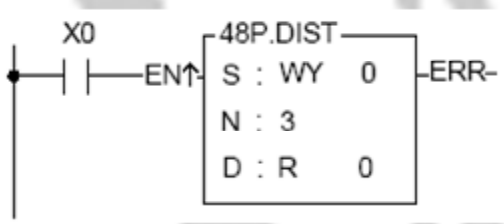
در **S** یک رجیستر**16**بیتی قرار می گیرد که به ترتیب **N** تعداد از نیبل های آن به رجیستر **D+1،D** و... منتقل می شود.

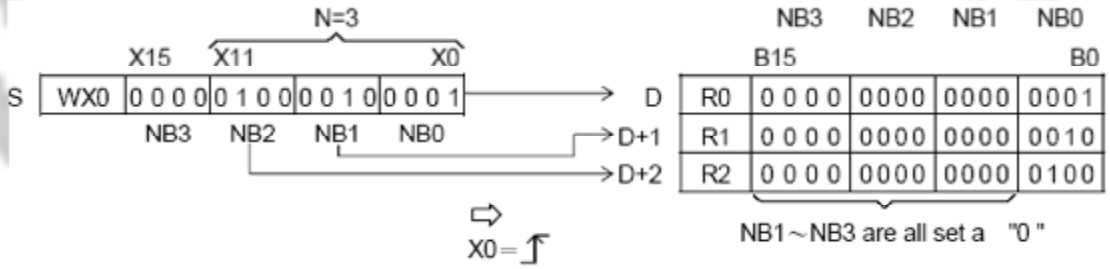
این نیبل ها ،نیبل پایین رجیسترهای **D+1،D** و... را اشغال می کنند و باقی فضای این رجیسترهای با **0** پر می شود.

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

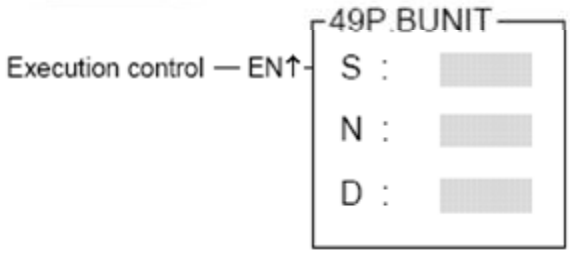
اگر **N:1~4** نباشد ، **error** فعال می شود.

مثال:



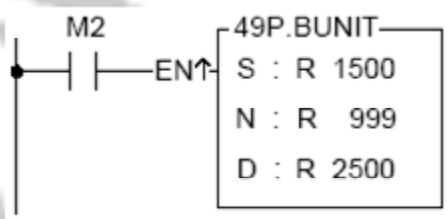


Function 49.BITE UNITE



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:



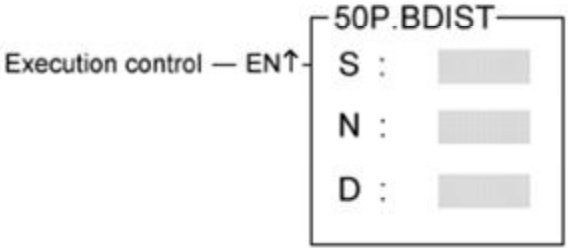
S

	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

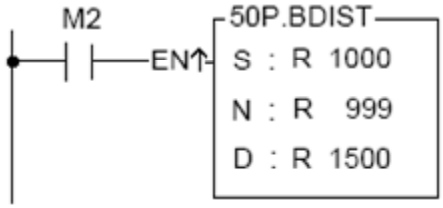
	High Byte	Low Byte
R2500	Byte-0	Byte-1
R2501	Byte-2	Byte-3
R2502	Byte-4	Byte-5
R2503	Byte-6	Byte-7
R2504	Byte-8	Byte-9

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:



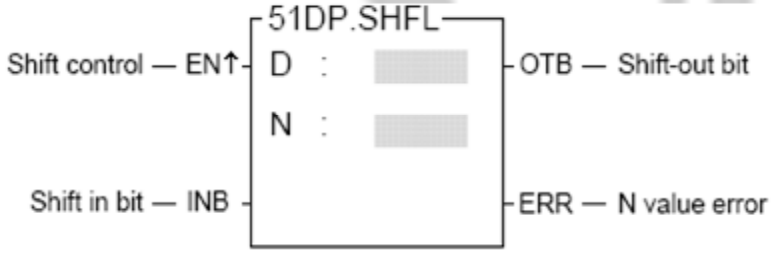
S

	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



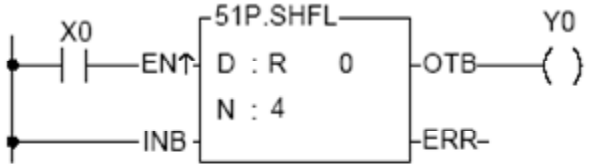
در D رجیستری که قرار است شیفت داده شود قرار می گیرد. N ، تعداد شیفت به چپ را مشخص می کند.جای بیت شیفت داده شده (کم ارزش ترین بیت) را مقدار ورودی "INB" پر می کند و بیت N اُم بالای رجیستر ، پس از شیفت ، به خروجی "OTB" فرستاده می شود .

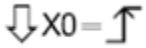
اگر رجیستر مورد نظر 16 بیتی باشد : N:1~16

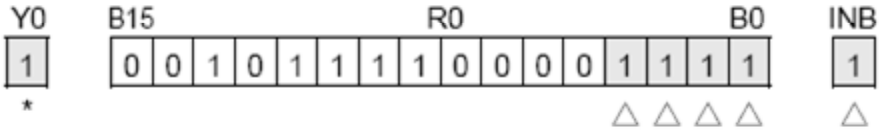
اگر رجیستر مورد نظر 32 بیتی باشد : N:1~32

در غیر این صورت error می دهد.

مثال:







Function 52.SHIFT RIGHT

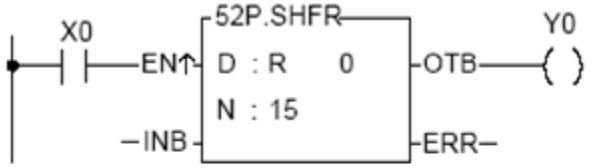
در D رجیستری که قرار است شیفت داده شود قرار می گیرد. N ،تعداد شیفت به راست را مشخص می کند. جای بیت شیفت داده شده (با ارزش ترین بیت) را مقدار ورودی "INB" پر می کند و بیت N اُم پایین رجیستر ، پس از شیفت ، به خروجی "OTB" فرستاده می شود .

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

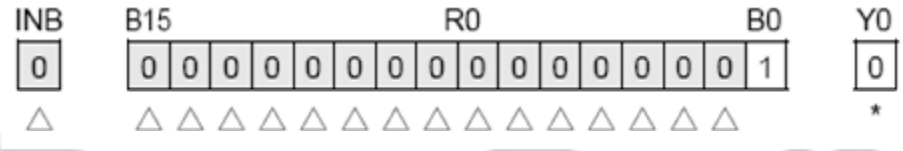
اگر رجیستر مورد نظر 32 بیتی باشد : N:1~32

در غیر این صورت error می دهد.

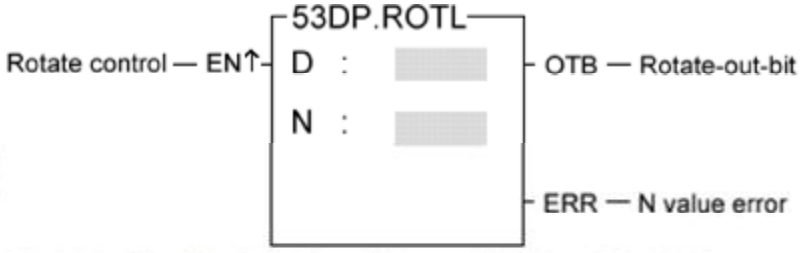
مثال:



⇩X0=⇧



Function 53.ROTATE LEFT



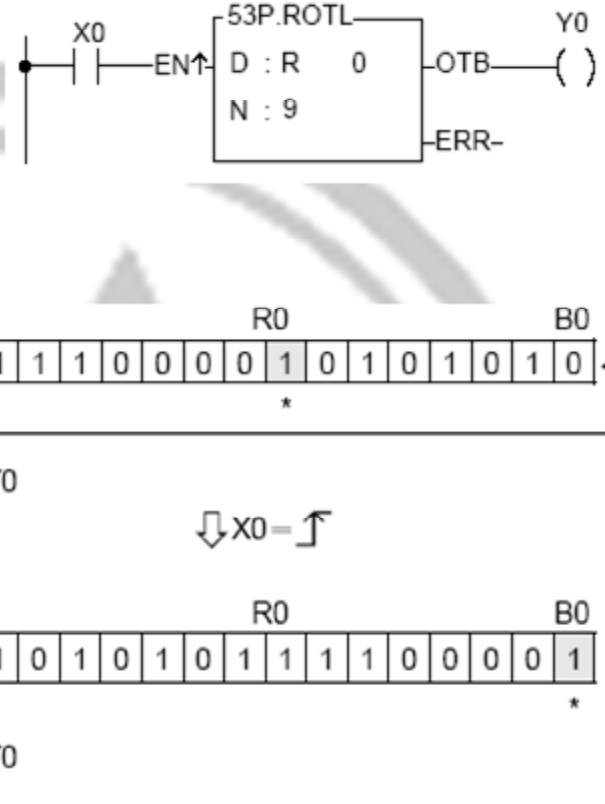
با این تابع یک چرخش به سمت چپ در محل بیت ها انجام می گیرد و چپ ترین بیت (با ارزشترین) به راست ترین بیت (کم ارزشترین) منتقل می شود و یک کپی از آن بیت به خروجی "OTB" نیز می رود.

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

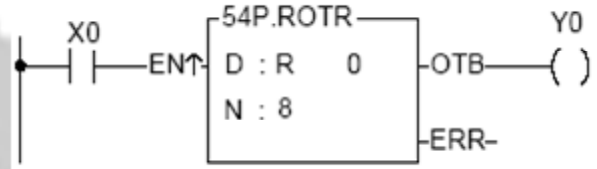
با این تابع یک چرخش به سمت راست در محل بیت ها انجام می گیرد و راست ترین بیت (کم ارزشترین) به چپ ترین بیت (با ارزشترین) منتقل می شود و یک کپی از آن بیت به خروجی "OTB" نیز می رود.

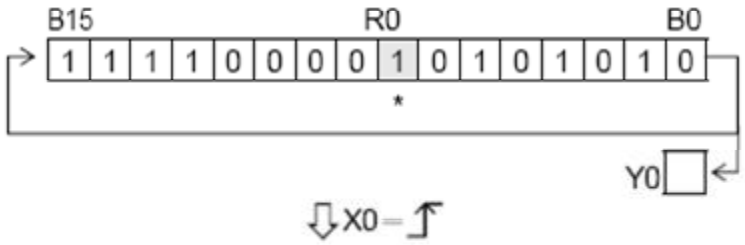
اگر رجیستر مورد نظر 16 بیتی باشد : N:1~16

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

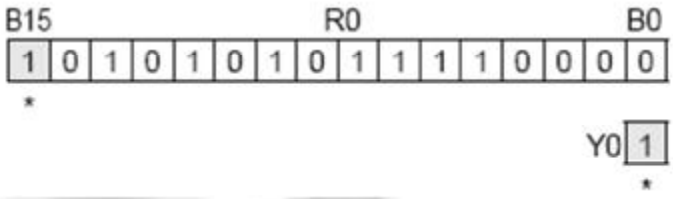
در غیر این صورت error می دهد.

مثال:





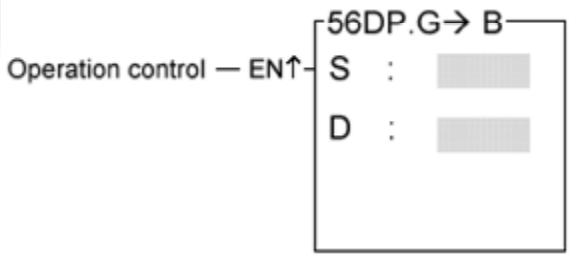
X0 ⇩⇧



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



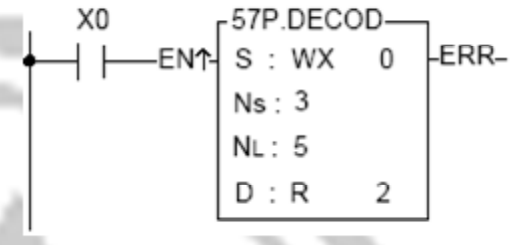
هرگاه "EN" از 0 به 1 تغییر کند: مقدار گری موجود در رجیستر S به صورت باینری،کدبندی شده و نتیجه در D ریخته می شود.

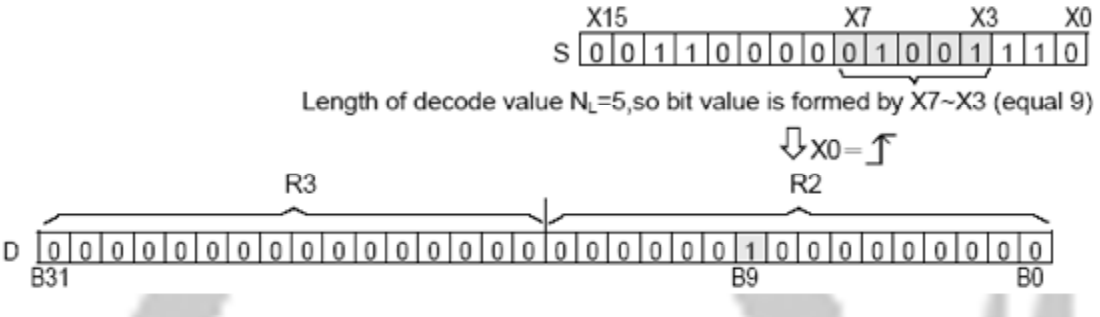
Function 57.DECODE

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

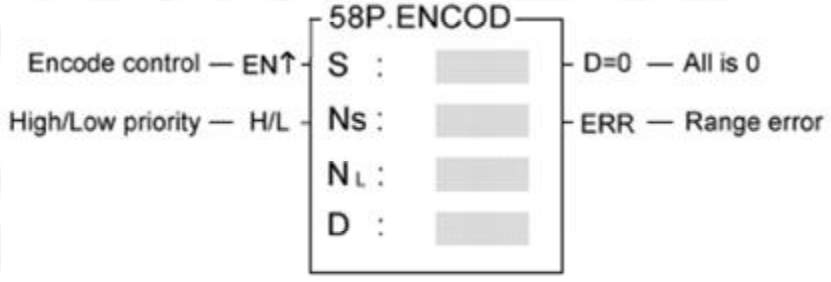
شد. در S یک رجیستر 16 بیتی ریخته می شود پس اگر مقدار Ns یا N_L متناسب با این محدوده نباشد، error می دهد.

مثال:





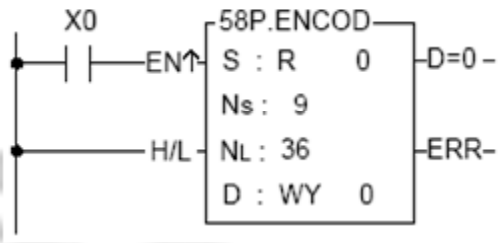
Function 58.ENCODE

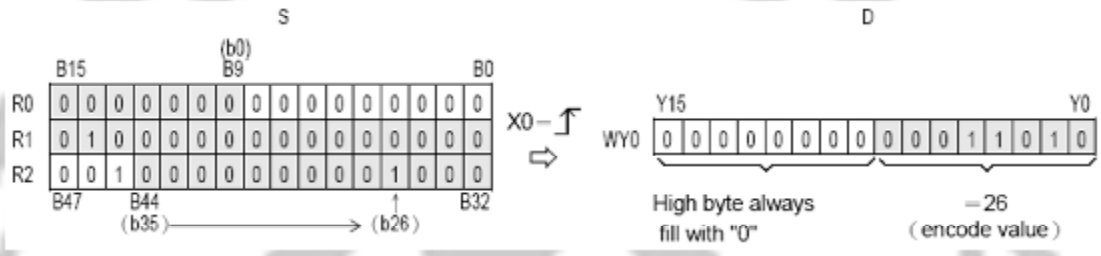


اولین رجیستراز رجیستر های مورد نظر برای کدگذاری در S و مقدار نهایی کدگذاری شده در D ریخته می شود.

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:

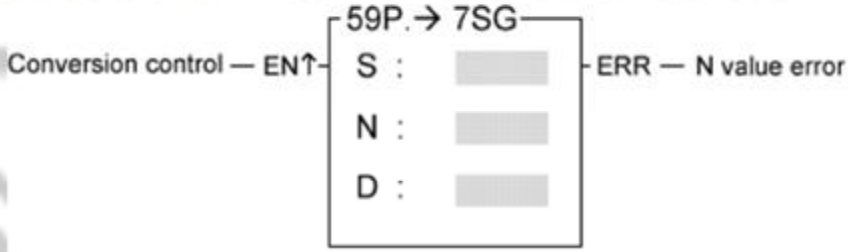




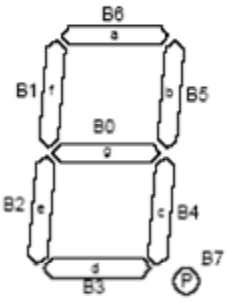
اولین بیت با مقدار 1

برای کدگذاری با اولویت بالا

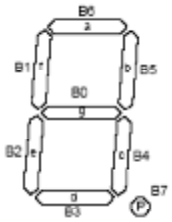
Function 59.7-SEGMENT CONVERSION



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:

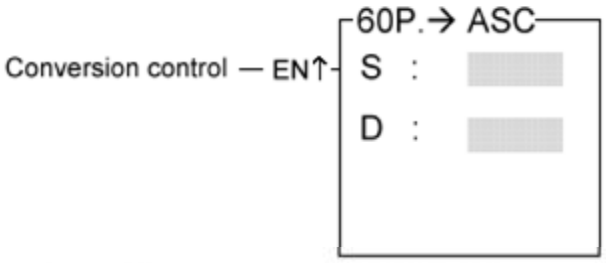


هرگاه بیت **B6** از رجیستر **D**، **1** شود ،معادل نمایش سگمنت **a** از **7-seg** می باشد . هرگاه بیت **B5** از رجیستر **D**، **1** شود ،معادل نمایش سگمنت **b** از **7-seg** می باشد و... ، اگر از **FATEK 7-seg (FBs-7SG)** استفاده می کنید، برای کاربرد نمایش رمزگشایی از طریق **7-seg** ،برای سادگی طراحی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

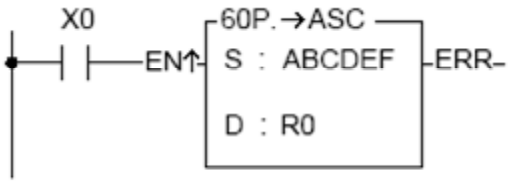
جدول الگوی نمایشی 7-seg

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

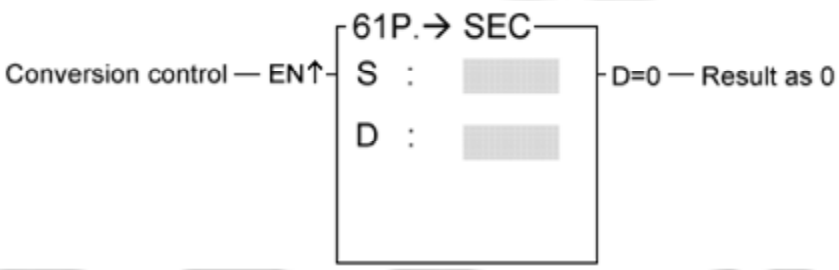
Alphabet
ABCDEF

X0 = ⤴

D

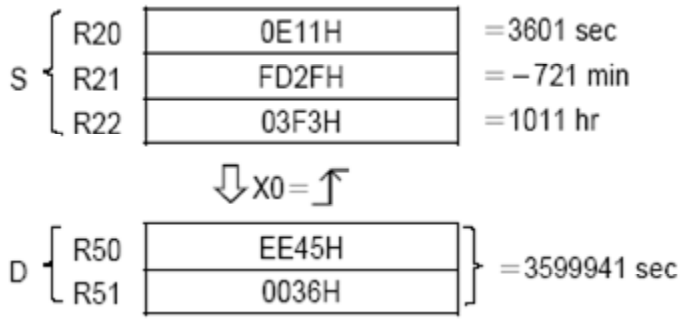
	High Byte	Low Byte
R0	42 (B)	41 (A)
R1	44 (D)	43 (C)
R2	46 (F)	45 (E)

Function 61.H:M:S to SECONDS CONVERSION

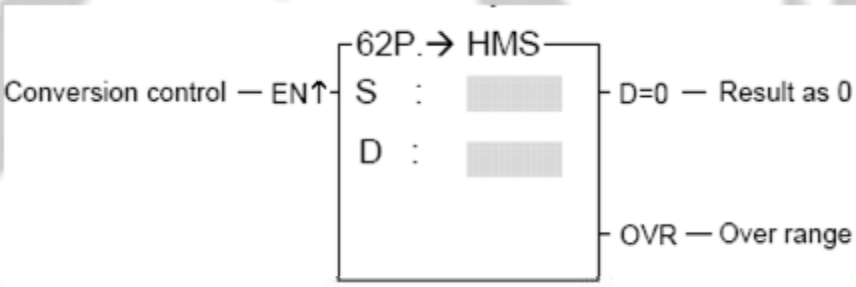


هرگاه "EN" از 0 به 1 تغییر کند: این تابع، مقدار زمانی را که به صورت ساعت ، دقیقه و ثانیه در رجیسترهای S~S+2 ذخیره شده است ، بر حسب ثانیه در رجیستر D ذخیره می کند. اگر نتیجه 0 باشد ،خروجی "D=0" فعال می شود. مقدار ذخیره شده در S بر حسب ثانیه ، S+1 بر حسب دقیقه و S+2 بر حسب ساعت است.

مثال:

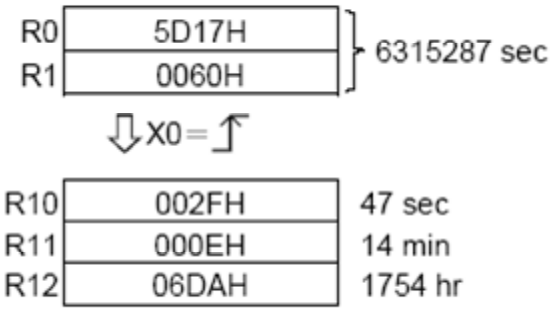


Function 62.SECOND to H:M:S

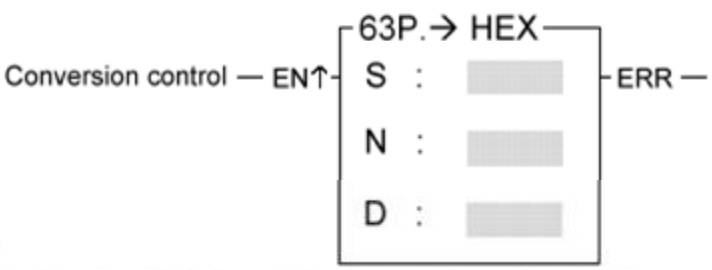


هرگاه "EN" از 0 به 1 تغییر کند: این تابع بر عکس تابع قبل عمل کرده و زمان بر حسب ثانیه را بر حسب ساعت ، دقیقه و ثانیه تبدیل می کند. مقداری که در D ذخیره می شود بر حسب ثانیه ،D+1 بر حسب دقیقه و D+2 بر حسب ساعت است. اگر مقدار موجود در S، 0 باشد ، خروجی "D=0" فعال می شود. مقدار مناسب برای S ، -117968399~117964799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

مثال:

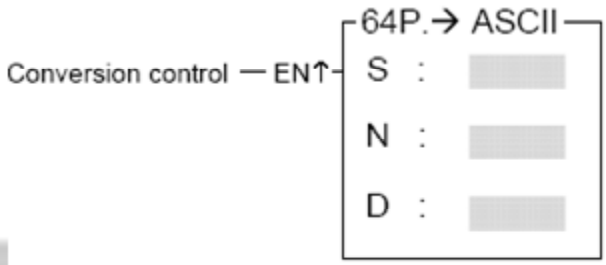


Function 63.ASCII to HEX



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

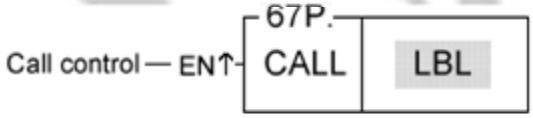
خود این تابع صرفا عملی انجام نمی دهد و به عنوان یک برچسب آدرس ، برای برنامه ها عمل می کند. به عنوان یک نشانگر برای اجرای تابع های CALL،JUMP ، INTERRUPT استفاده می شود.همچنین برای آسان خوانی برنامه نیز می توان به کار برد. در S نام برچسب متشکل از حروف و اعداد نوشته می شود که از 1 تا 6 حرف می تواند داشته 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

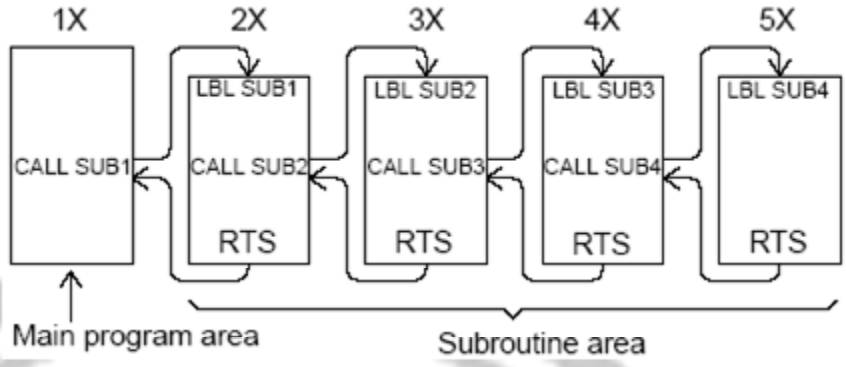


هرگاه "EN" از 0 به 1 تغییر کند:

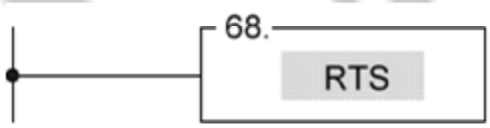
PLC به زیربرنامه ا ی می رود که تابع CALL از طریق LABLE نوشته شده در تابعآن را فراخوانی می کند و به اجرای آن زیربرنامه می پردازد تا زمانی که به تابع RTS بربخورد ،سپس به همانجایی بر می گردد که تابع CALL در ان نوشته شده و از ادامه تابع CALL به اجرای برنامه می پردازد.

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.

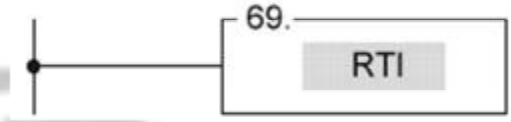


Function 68.RTS



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.

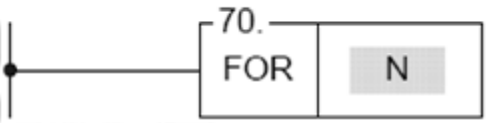
Function 69.RTI



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

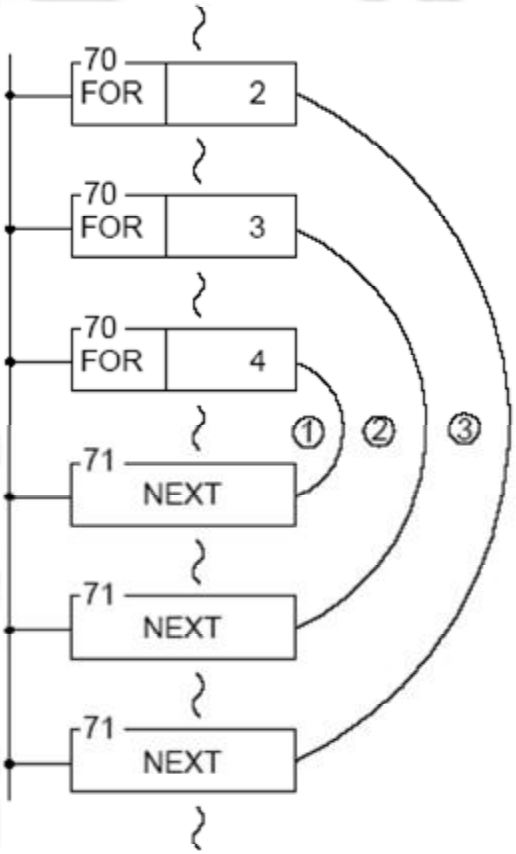
اینتراپتی اجرا خواهد شد که در اولویت بالاتر قرار دارد. توجه داشته باشید که حتما از RTI درانتهای روتین اینتراپت استفاده کنید.

Function 70.FOR

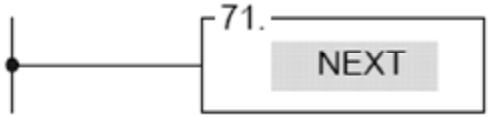


این تابع به همراه تابع NEXT، تشکیل یک حلقه را می دهد که برنامه ی میان FOR وNEXT مربوطه ،N بار اجرا می شود.حلقه های FOR وNEXT دیگری نیز در میان حلقه های FOR و NEXT اولیه می تواند قرار بگیرد.

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

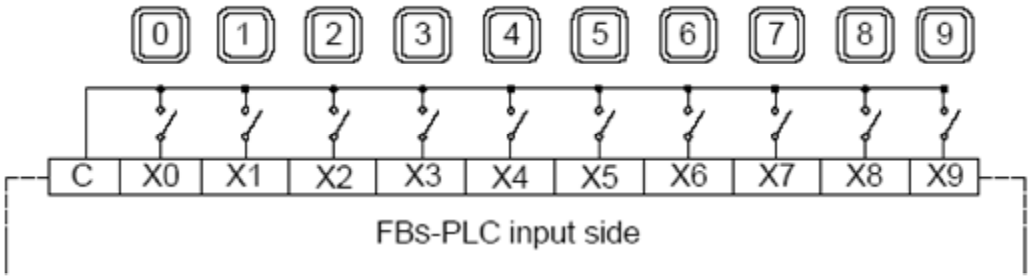
این تابع سیگنال های ورودی و خروجی را فورا ، update می کند. هرگاه "EN" از 0 به 1 تغییر کند: سیگنال های ورودی یا خروجی با آدرس شروع D به تعداد N ،refresh شده و تغییر ایجاد شده در ورودی-خروجی مورد نظر ، بدون نیاز به کامل شدن اسکن برنامه ، در همان لحظه اعمال می شود .

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

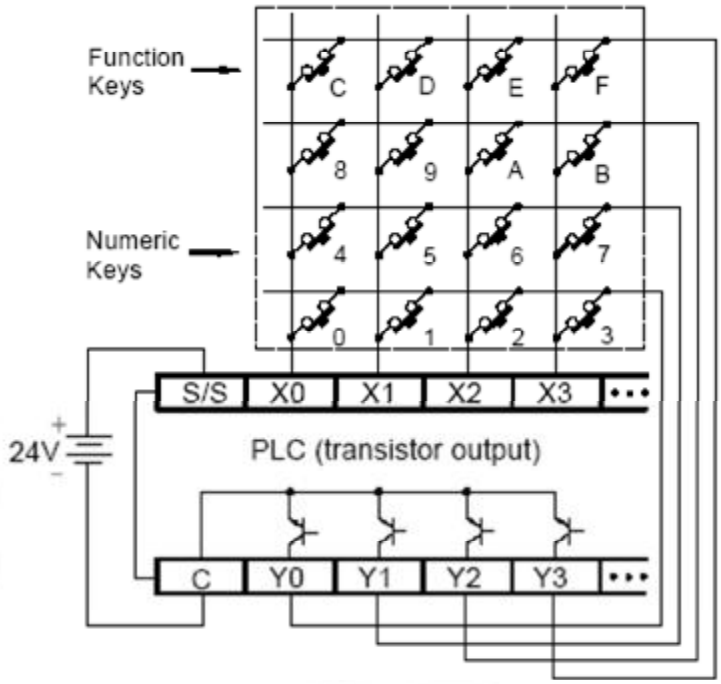
Function 76.DECIMAL-KEY INPUT

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

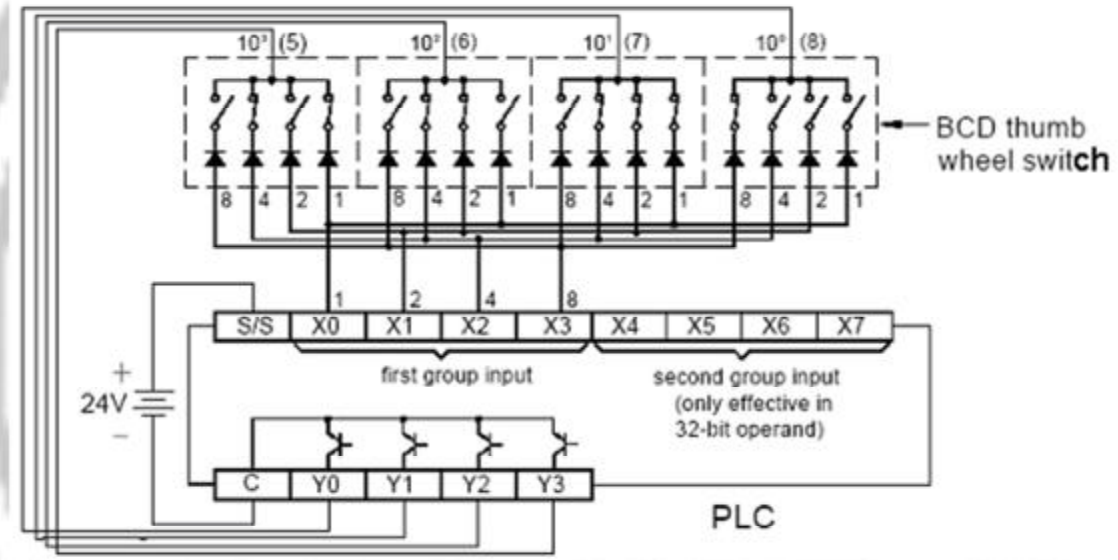
این تابع مشابه تابع قبل است با این تفاوت که علاوه بر اعداد 0~9 ، می تواند 6 ورودی دیگر نیز دریافت کند (A~F) که هرکدام می توانند معادل یک دستورعمل باشند. همچنین اتصال سخت افزاری این تابع با تابع قبل متفاوت است ،4 ورودی تعیین شده PLC به 4 خروجی تعیین شده ، طوری متصل می شوند که اتصال هر کدام از آنها یکی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

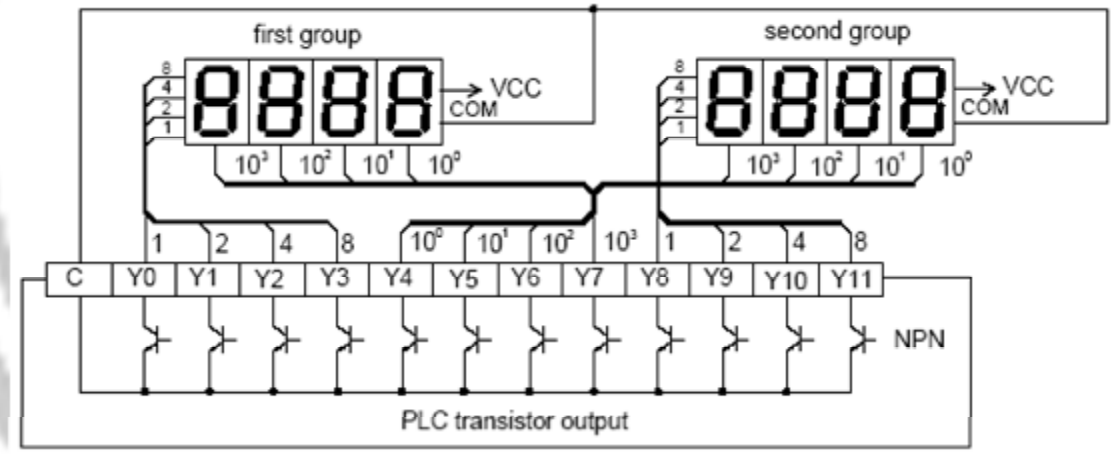


هربار که 4 رقم از سوئیچ ها خوانده شد ، خروجی 1="Dn" می شود. اگر هریک از اعداد خوانده شده در رنج BCD(0~9) نباشد، خروجی "ERR" فعال می شود. PLC مورد استفاده باید دارای خروجی ترانزیستوری باشد.
رجیستر ذخیره شونده در WR برای کاربری محاسبات خود تابع است و در جای دیگری نباید از آن استفاده کرد.

Function 79.7-SEG OUTPUT WITH LATCH

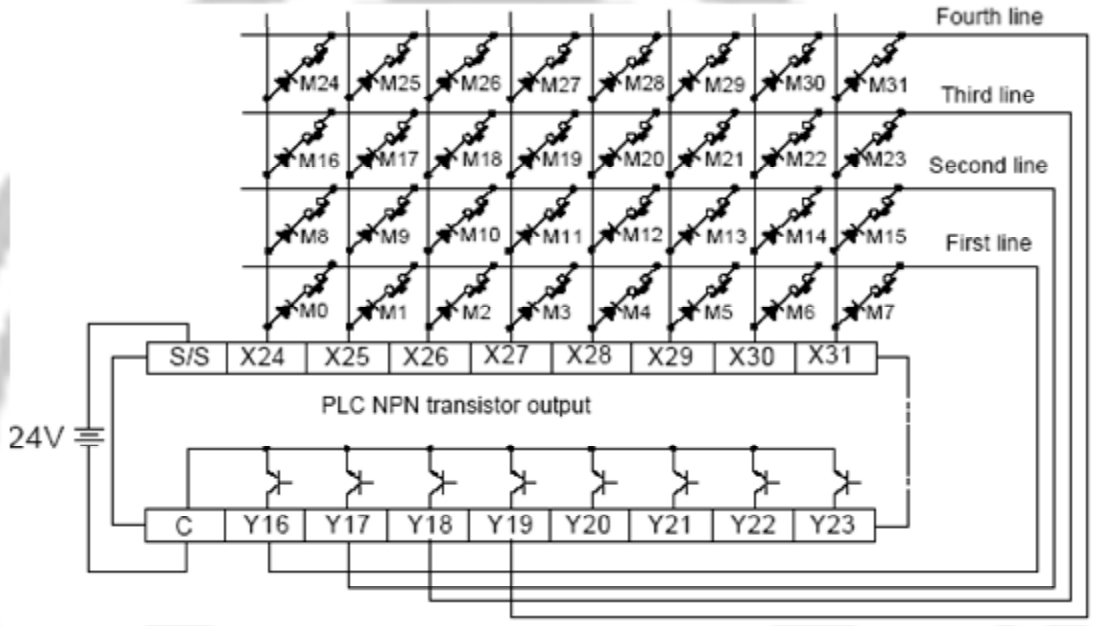
با توجه به شکل بعدی ، هرگاه **1= "EN"**، **4** نیبل رجیستر مشخص شده در **S** ، برای نمایش به **7-SEG**سری اول و **4** نیبل **S+1** ، به **7-SEG** سری دوم منتقل می شوند.

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

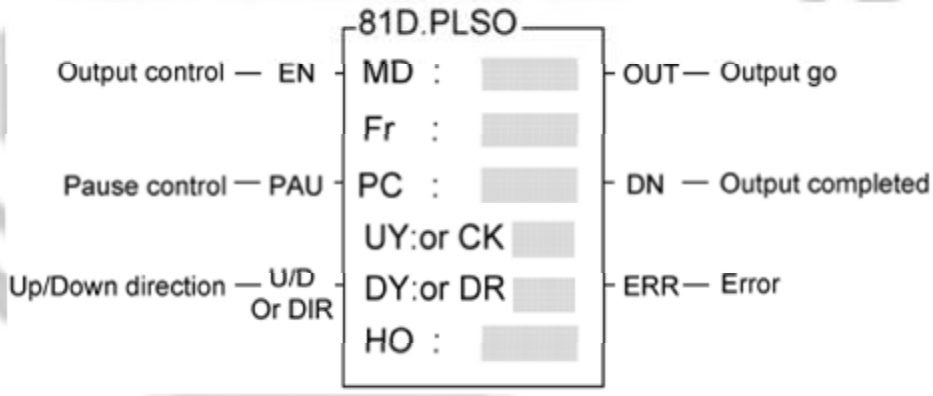


Function 80.MULTIPLEX INPUT

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!)

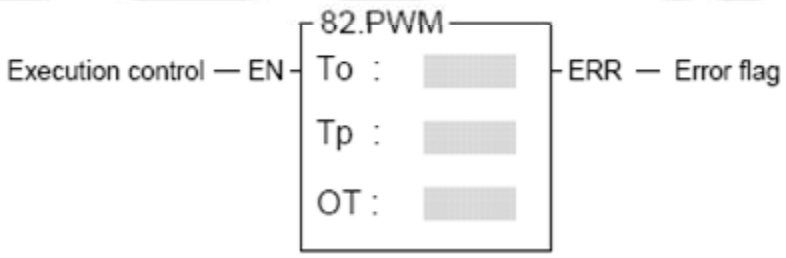


Function 81.PULSE OUTPUT

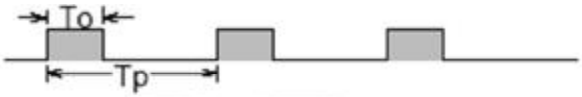


از این تابع برای فرستادن پالس به خروجی استفاده شده و عمده کاربرد آن می تواند راه اندازی استپ موتور با ورودی پالس راست گرد و چپ گرد باشد . از این تابع تنها می توان یکبار در برنامه استفاده نمود و PLC مورد استفاده باید دارای خروجی ترانزیستوری باشد.

Function 82.PULSE WIDTH MODULATION

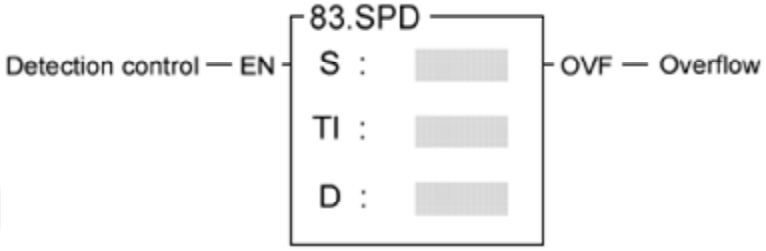


از این تابع برای فرستادن پالس به خروجی با عرض پالس دلخواه استفاده می شود . وقتی 1="EN" است،پالسی به خروجی OT می فرستد که To میلی ثانیه ON است و پریود آن Tp میلی ثانیه می باشد. خروجی مورد استفاده باید ترانزیستوری باشد . حداقل مقدار To، 0 است که در این موقعیت خروجی همیشه به حالت OFF خواهد بود. حداکثر مقدار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



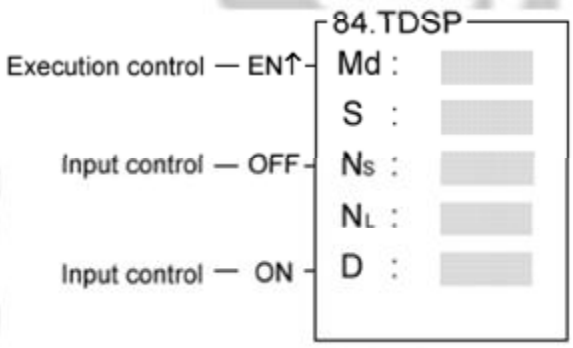
اگر To>Tp بشود ،خروجی "ERR" فعال شده و تابع عمل نخواهد کرد. این تابع می تواند تنها یک بار در برنامه استفاده شود.

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.

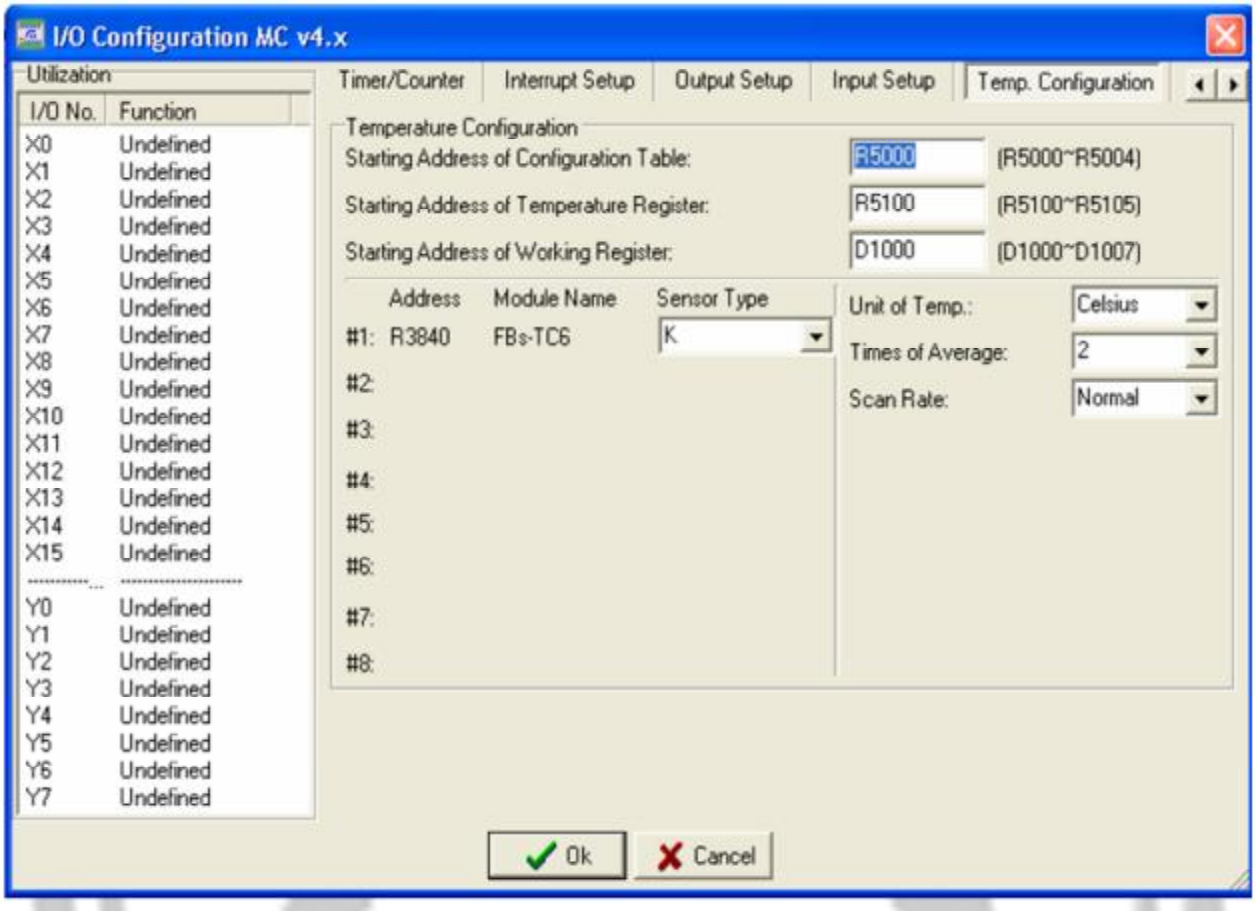
D، آدرس شروع محل ذخیره ی نتایج تبدیل را مشخص می کند.

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

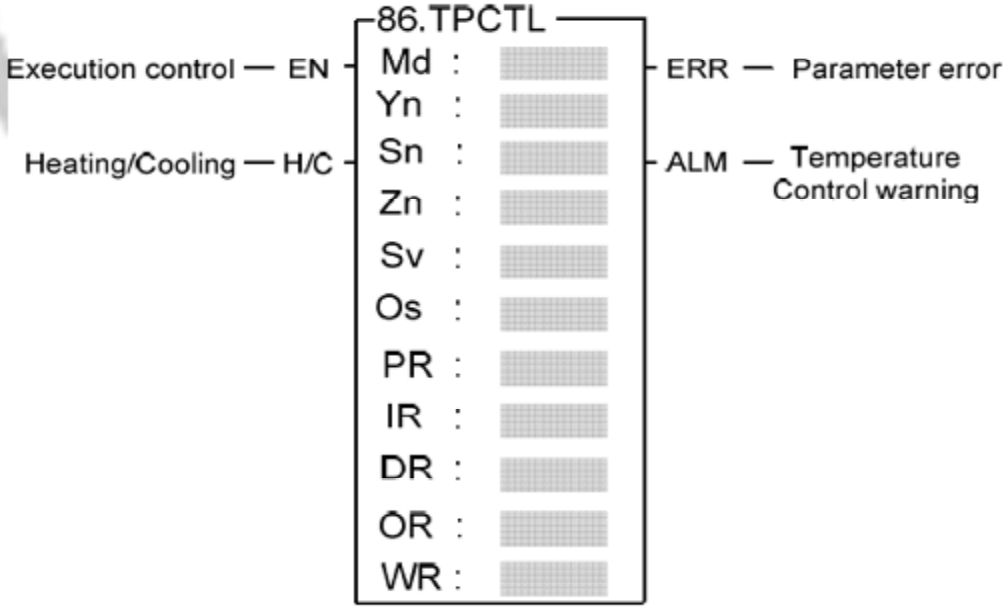
وقتی ورودی های "EN"=1 ،"OFF"=0 ، "ON"=0 و Md=0 باشد،این تابع تبدیل را اجرا می کند.وقتی "OFF"=1 و Md=0، تمام بیت های مربوط به الگوی نمایشی 16-seg صفر خواهند شد.این بدین معنیست که اگر 16-seg در این حالت به PLC متصل شود ،تمام LED های آن خاموش خواهد بود. وقتی "OFF"=1 و Md=1 ، تمام بیت های مربوط به نمایش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

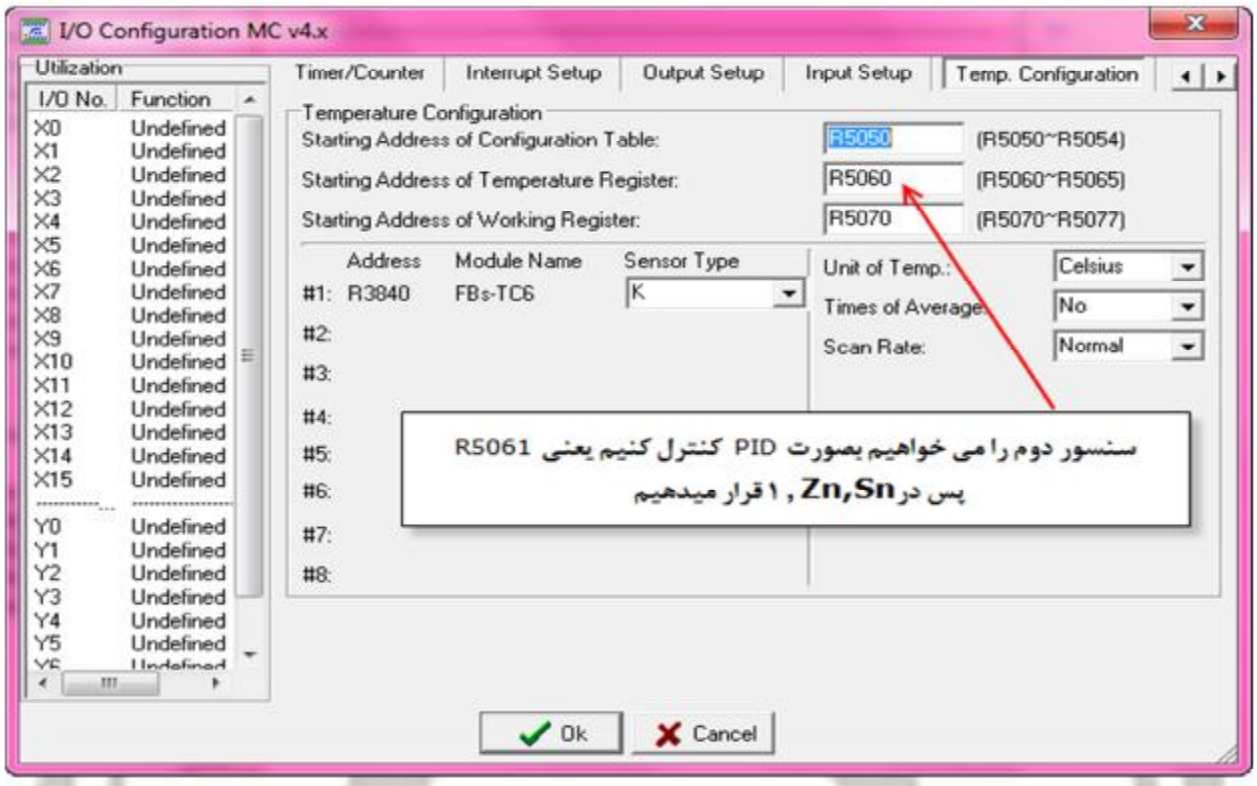
به ازای سنسورهای بعدی, ریجیسترهای بعد از ریجیسترهای نوشته شده در پارامترهای زیر مورد استفاده قرار می گیرند (به تعداد Zn)

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)

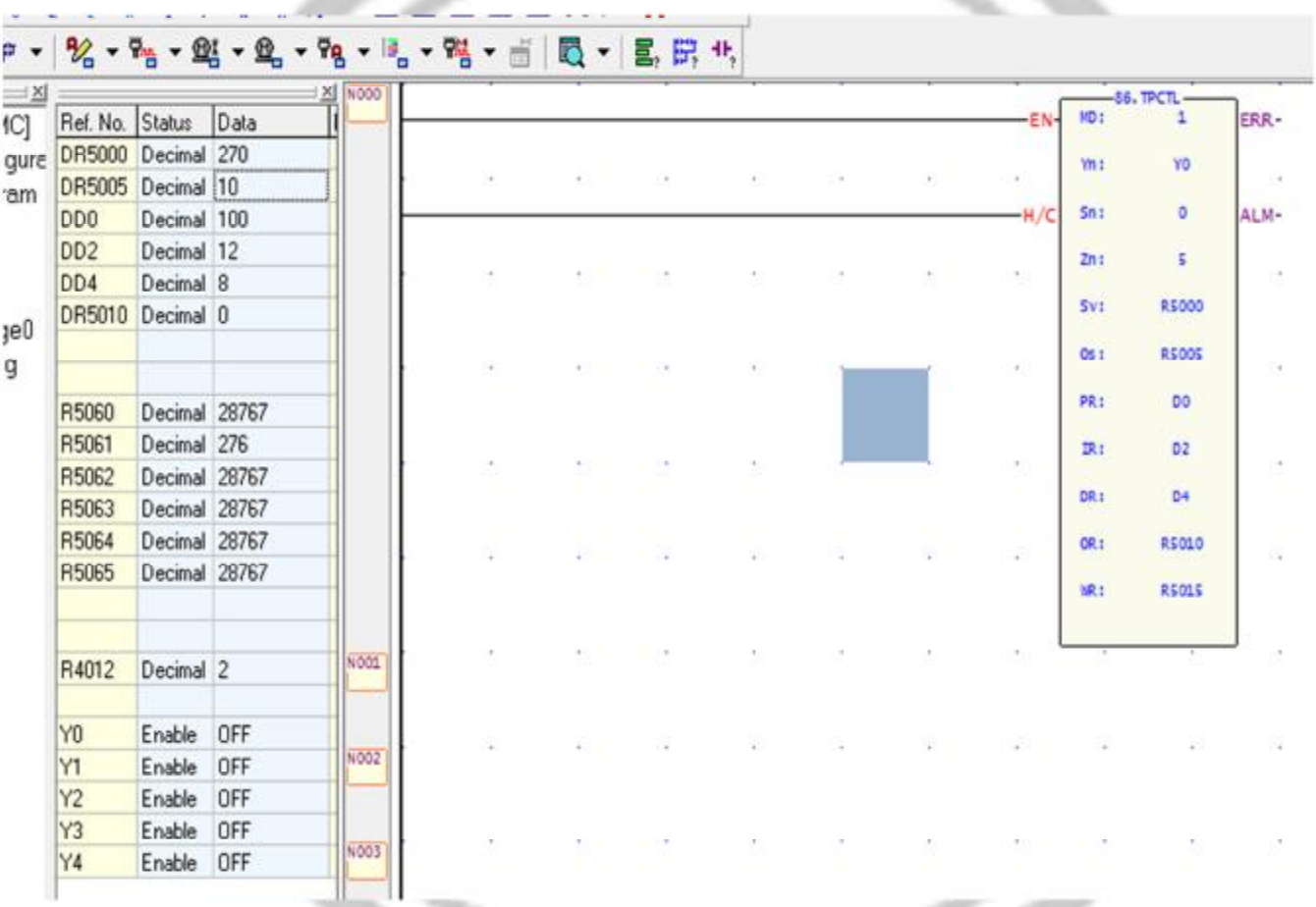
DR : در این قسمت رجیستر مربوط به مقدار ضریب مشتق برای حلقه PID . مشخص می گردد(معمول : 8)

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

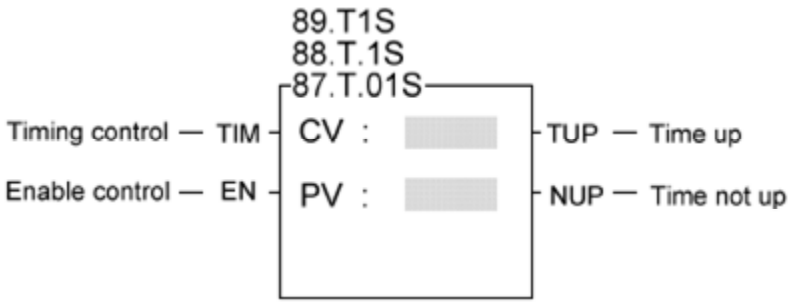
WR : Working Register ها را برای حلقه PID مشخص می کند که 9WORD می باشد.

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 :

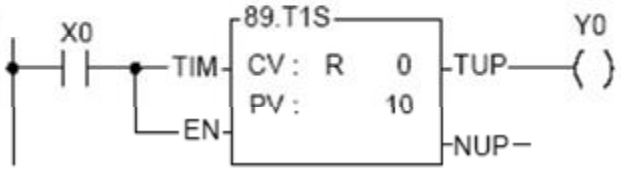
"TUP" که بعد از اتمام محاسبه ی زمان،1 می شود و

"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.

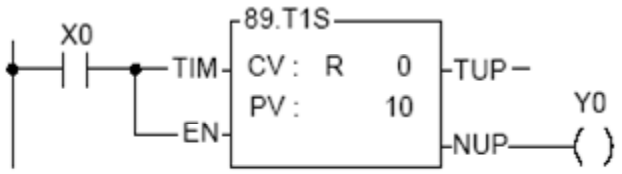
به عنوان مثال:

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

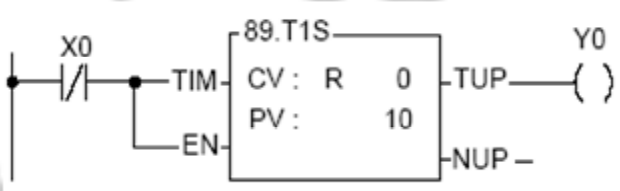


- Y0 به طور عادی ON است،وقتی ON،X0 شود،بعد از 10 ثانیه Y0 ، OFF

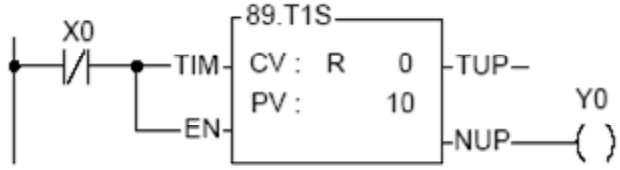
می شود.



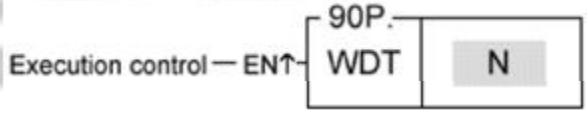
- YO به طور عادی OFF است،وقتی XO، OFF شود،بعد از 10 ثانیه، YO، ON می شود.



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



Function 90.WATCH DOG TIMER

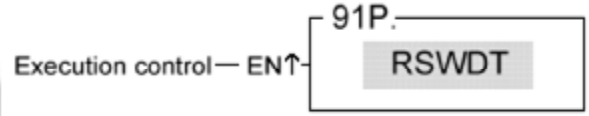


هدف اصلی از طراحی تایمر WDT ، محافظت ویژه ایست که از طرف سیستم اعمال می شود.برای مثال،اگر واحد پردازشگر PLC (CPU) ناگهان آسیب دیده و راهی برای اجرای برنامه یااعمال تغییرات به ورودی ، خروجی ها (I/O refresh)نباشد، بعد از سپری شدن زمان تعیین شده ، WDT به صورت خودکار تمام ورودی ، خروجی ها را خاموش

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



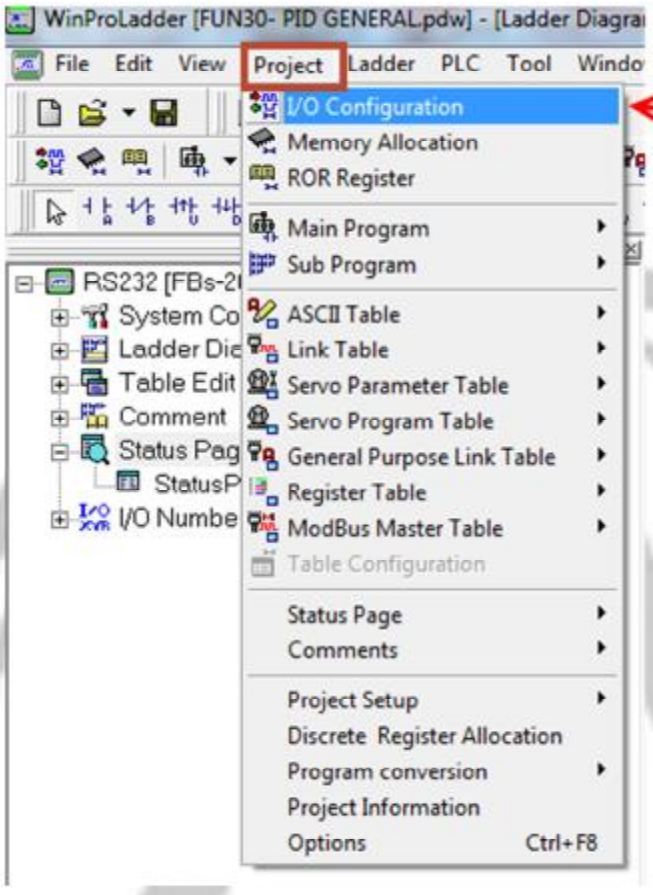
هرگاه "EN"، 1 بشود؛ تایمر WDT ری ست شده و محاسبه زمان را از 0 شروع می کند.

Function 92 & 93 HIGH SPEED COUNTER

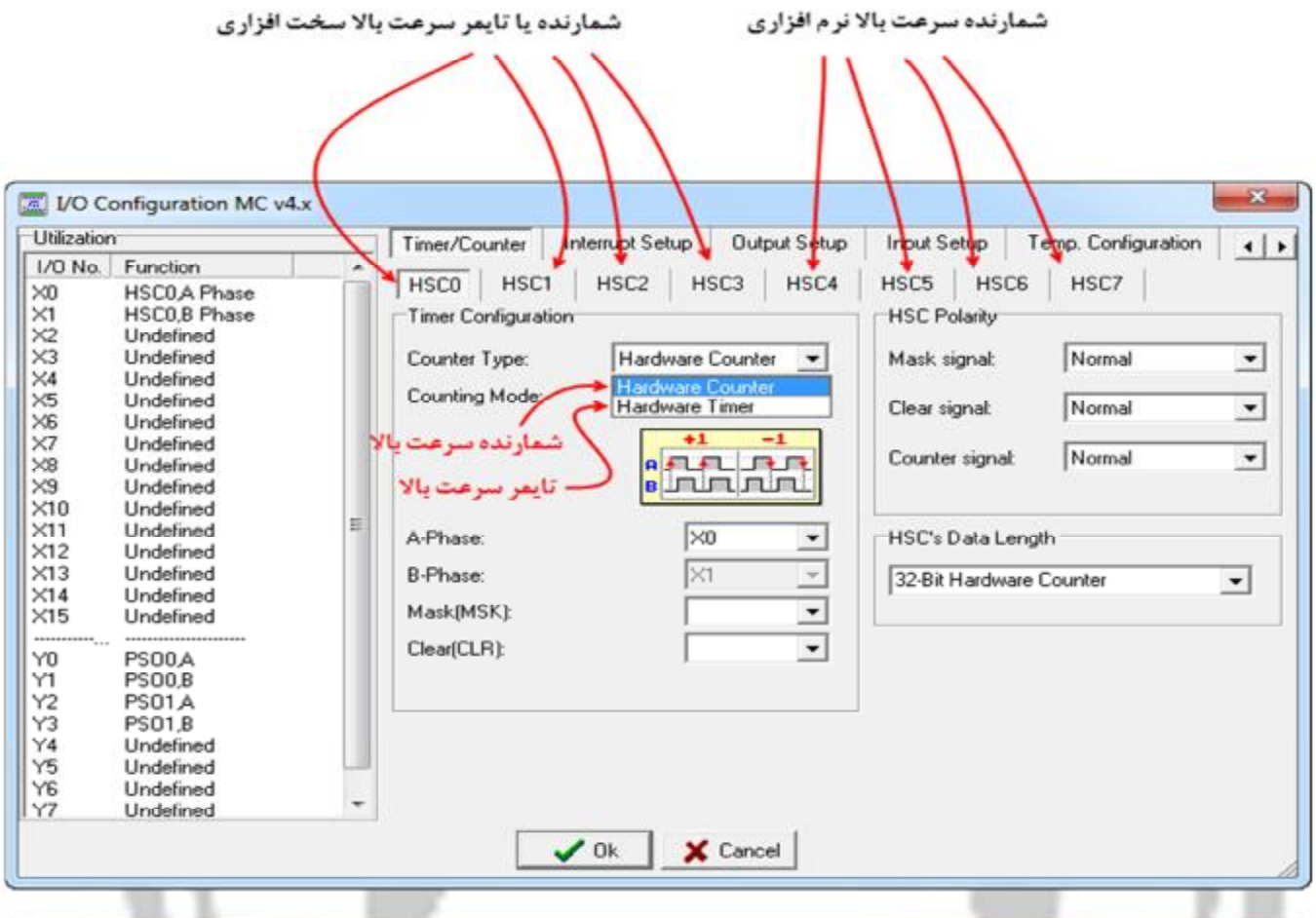
دو نوع شمارنده داریم : سرعت بالا و سرعت پایین (منظور از سرعت فرکانس ورودی می باشد) .

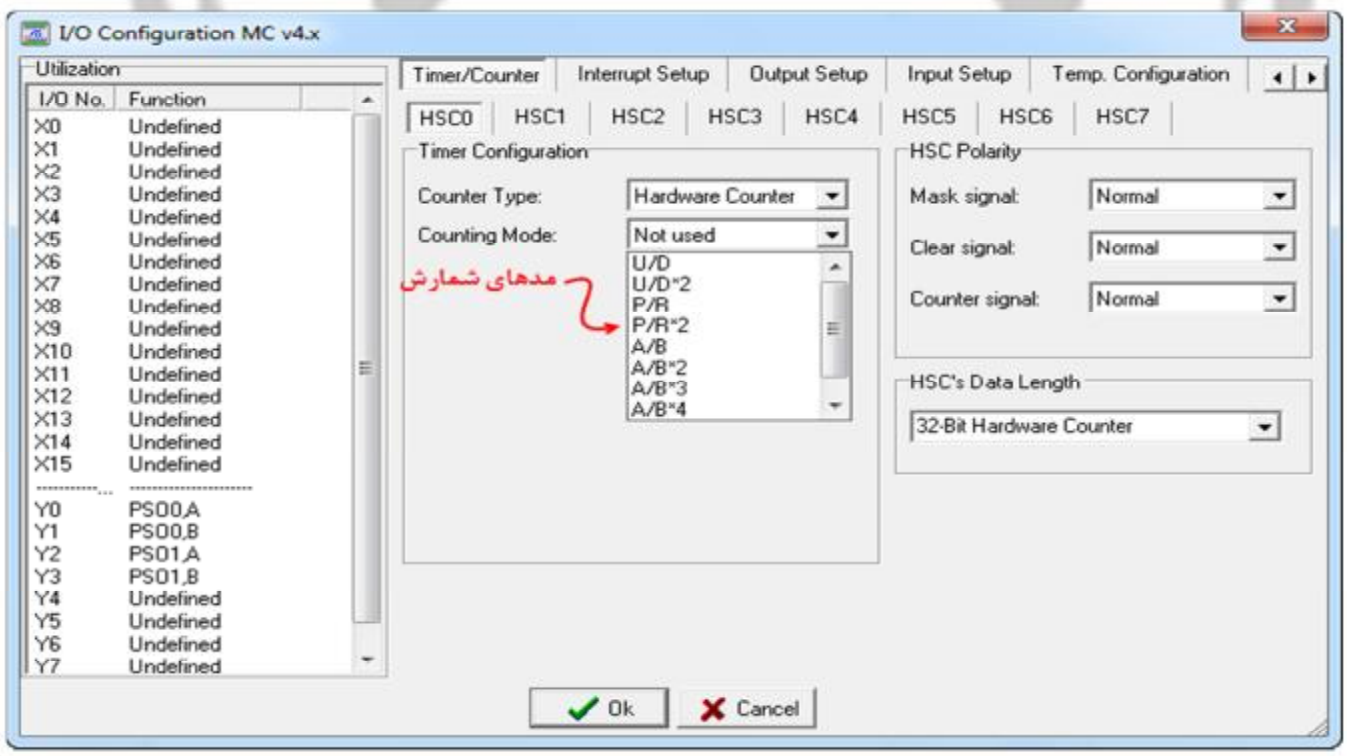
شمارنده های سرعت بالا (در مقیاس کیلوهرتز) نیاز به سخت افزار خاصی دارند تا بتوانند در این فرکانس پالسها را بشمارند ولی شمارنده های سرعت پایین (در مقیاس دهها هرتز) با ورودی های معمولی PLC نیز کار می کنند. به شمارنده های سرعت بالا شمارنده های سرعت بالای سخت افزاری و به شمارنده های سرعت پایین شمارنده های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

تنظیم ورودی ها برای حالت شمارنده بودن :



برای تنظیم ورودی های
شمارنده سرعت بالا

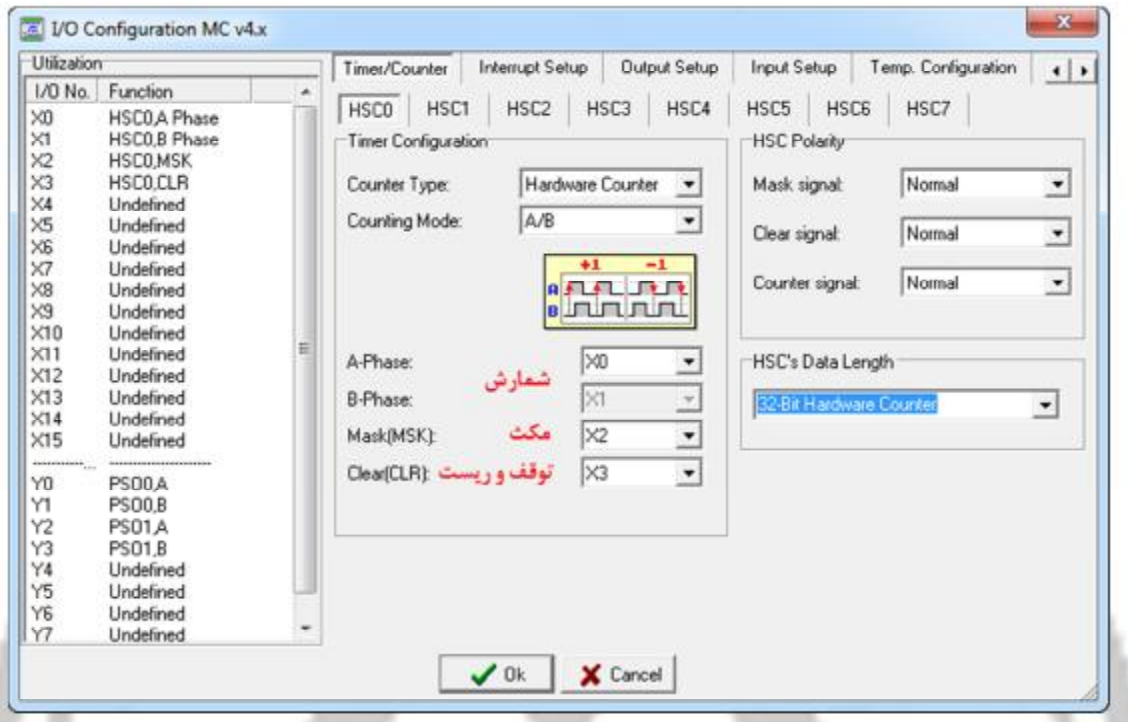

شمارنده یا تایمر سرعت بالا سخت افزاری
شمارنده سرعت بالا نرم افزاری
I/O Configuration MC v4.x
Utilization
I/O No. Function
X0 HSC0,A Phase
X1 HSC0,B Phase
X2 Undefined
X3 Undefined
X4 Undefined
X5 Undefined
X6 Undefined
X7 Undefined
X8 Undefined
X9 Undefined
X10 Undefined
X11 Undefined
X12 Undefined
X13 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 :

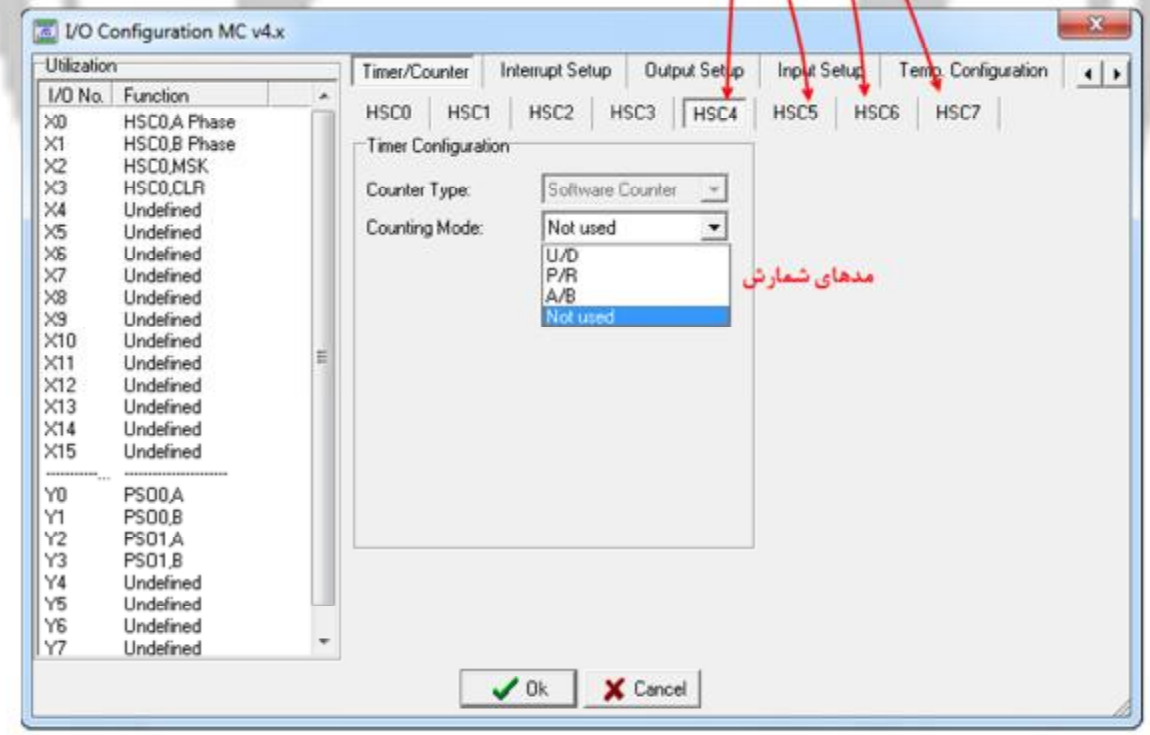
Counting Mode			HHSC (HSC0～HSC3)	SHSC (HSC4～HSC7)	Counting Waveform
					Up Counting（+1） Down Counting（−1）
Up-down pulse	MD 0	U/D	○	○	
	MD 1	U/D×2	○		
Pulse-direction	MD 2	K/R	○	○	
	MD 3	K/R×2	○		
AB phase	MD 4	A/B	○	○	
	MD 5	A/B×2	○		
	MD 6	A/B×3	○		
	MD 7	A/B×4	○		

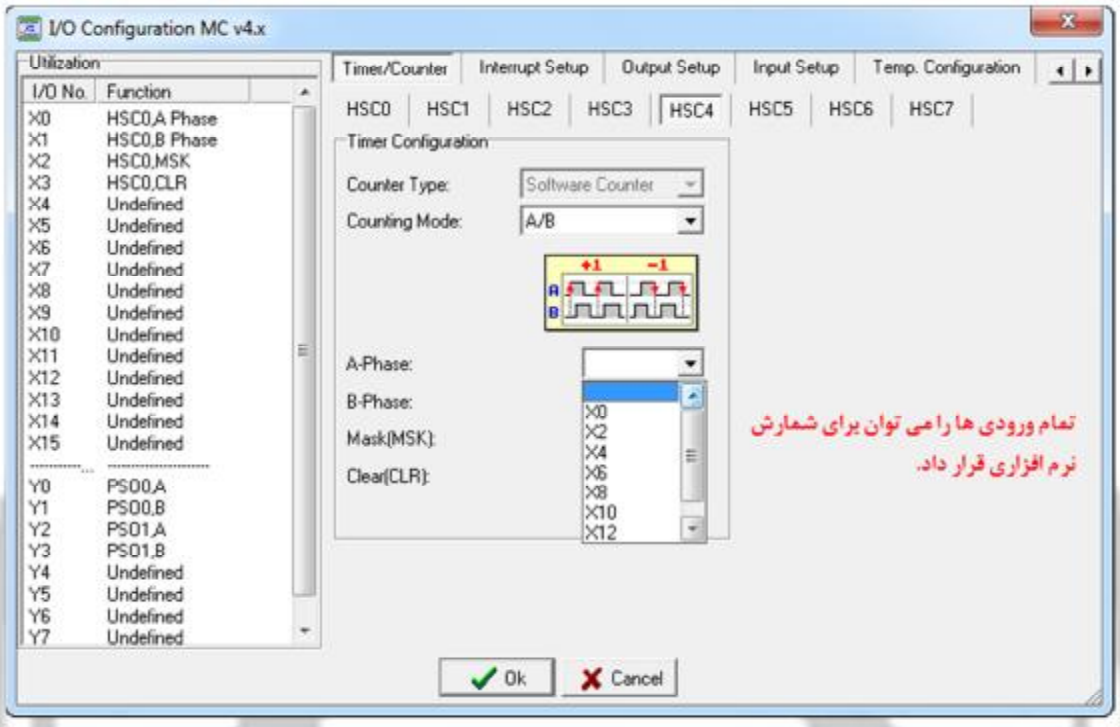
• The up/down arrow (↑ , ↓) on the positive/negative edge in the waveform represents where counting (+1 or −1) occurs.



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 :

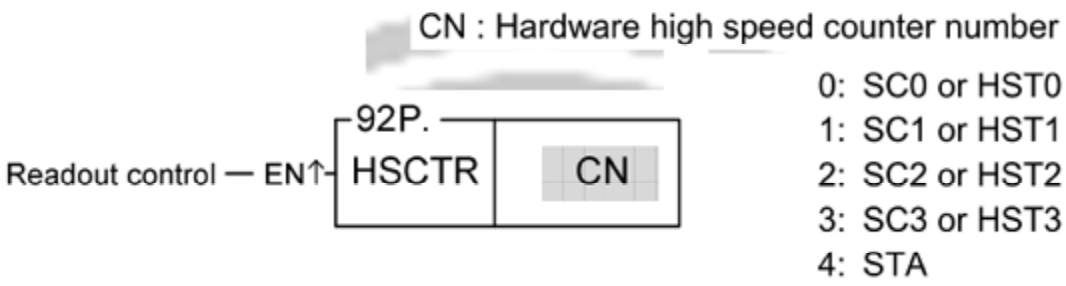
به ازای هر کانال ورودی شمارشگر 2 ریجیستر 32 بیتی اختصاص داده شده است :

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

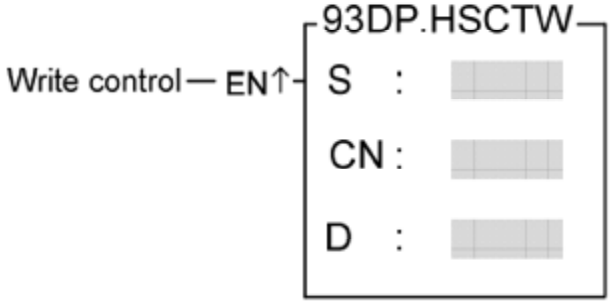
1- **قرار دادن مقدار صفر در ریجیستر شمارش کنونی (ریست کردن شمارشگر) ، توسط فانکشن "HSCTW" 93**
2- **خواندن مقدار کنونی شمارش شده ، توسط فانکشن 92 "HSCTR"**

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.



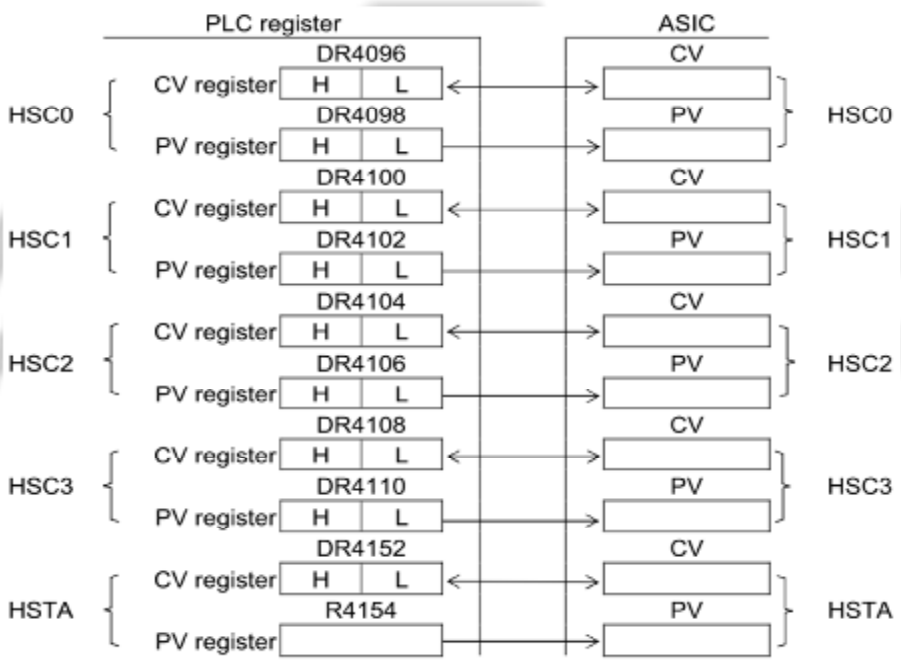
S : The source data for writing

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)

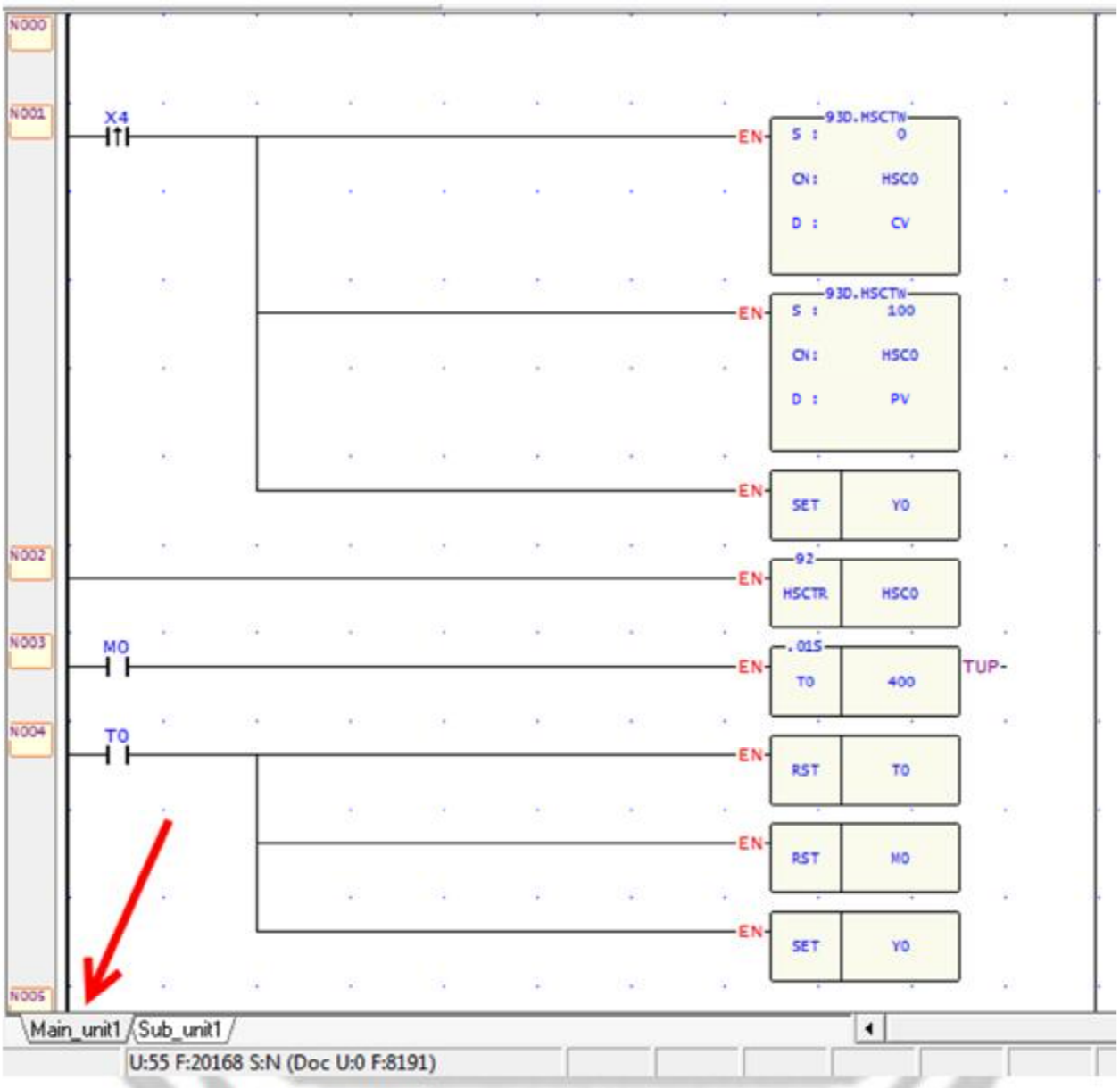
حافظه های پیش فرض اختصاص داده شده مربوط به شمارنده ها :

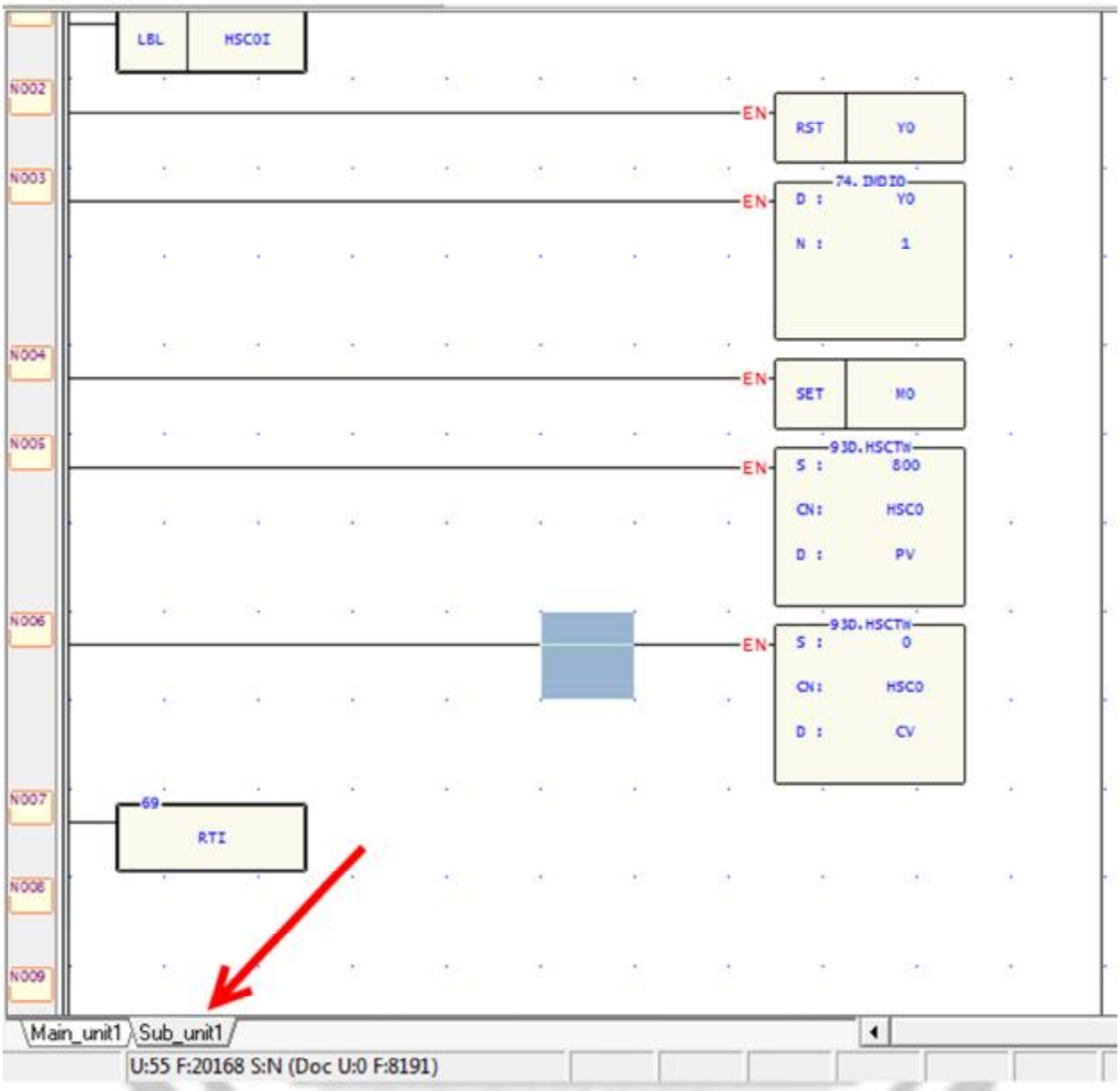


Type / Signal Allowed		MC/MN								MA
		HHSC				SHSC				SHSC
		HSC0	HSC1	HSC2	HSC3	HSC4	HSC5	HSC6	HSC7	HSC4 ~ HSC7
CV Register		DR4096	DR4100	DR4104	DR4108	DR4112	DR4116	DR4120	DR4124	The same as SHSC of MC/MN
PV Register		DR4098	DR4102	DR4106	DR4110	DR4114	DR4118	DR4122	DR4126	The same as SHSC of MC/MN
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

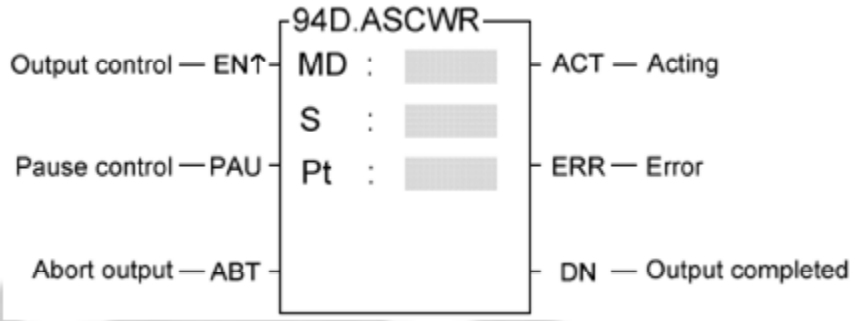
وقفه های مربوط به شمارنده های سرعت بالا : وقتی **CV=PV** شود سیکل برنامه وارد اینترایت مورد نظر می شود

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)



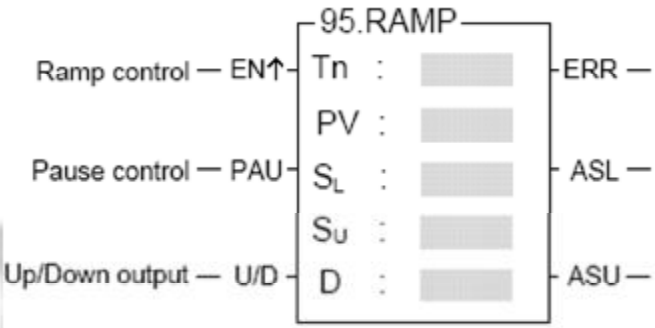
Function 94.ASCII WRITE



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

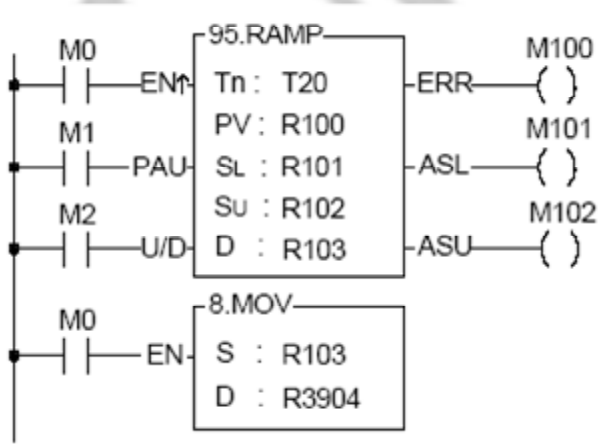


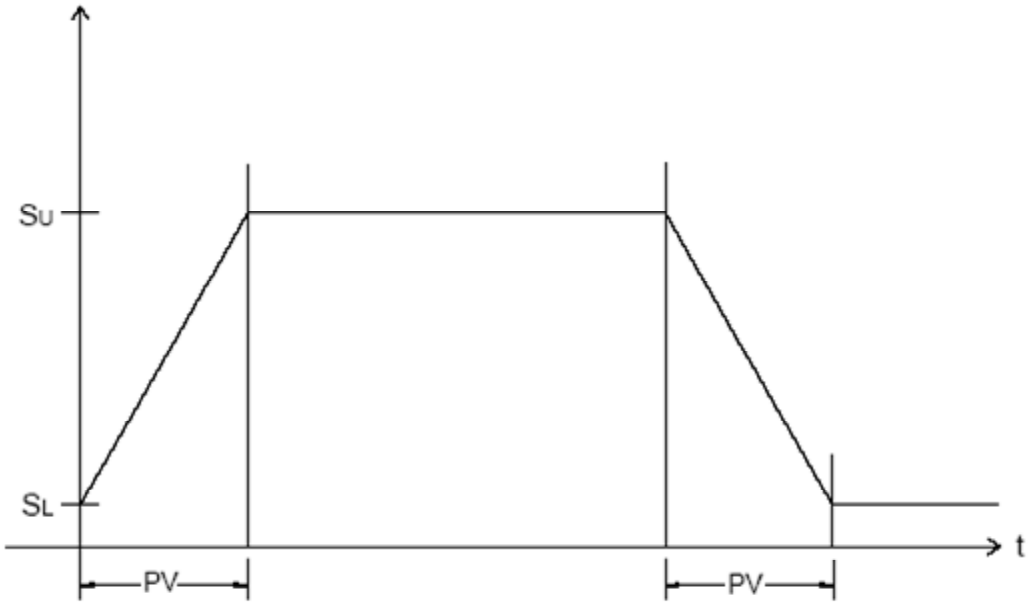
کاربرد این تابع در مواقعی است که نیاز به افزایش یا کاهش مقدار یک رجیستر به ویژه خروجی آنالوگ است که این افزایش یا کاهش با شیبی با ابتدا و انتهای مشخص و در زمان معین صورت می پذیرد . درTn ، تایمری با پایه زمانی 0.01 ثانیه قرار می گیرد که نباید در جای دیگری استفاده شود. PV مقدار پیش تنظیم تایمر Tn است . 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:

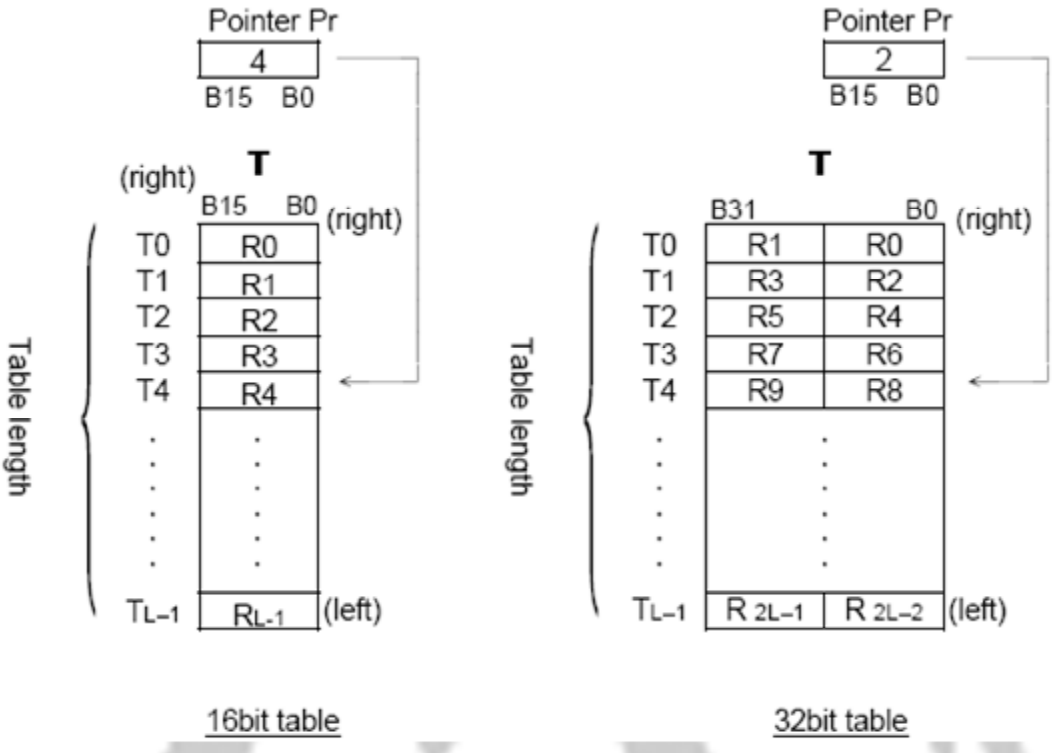




توابع جدولی

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

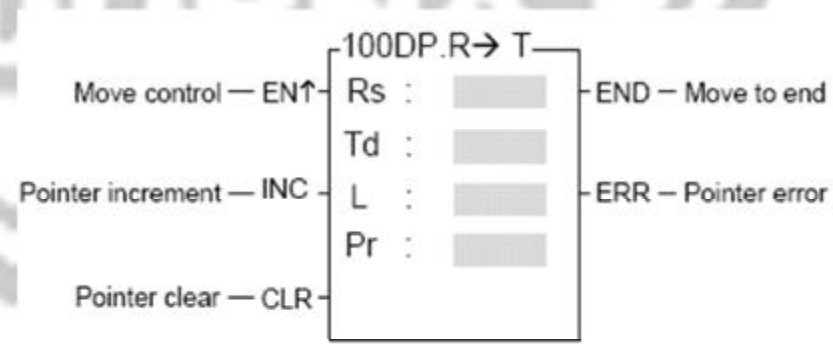
یک جدول شامل دو یا چند رجیستر متوالی است (16 یا 32 بیتی) تعداد رجیستر هایی که تشکیل یک جدول می دهند را طول جدول می نامند.(L) اجرای توابع جدولی بیشتر برای پردازش داده ها استفاده می شود،مانند انتقال ، کپی ، مقایسه ، جستجو و ...بین جدول ها و رجیستر ها یا بین جدول ها. در میان توابع جدولی ،بیشتر توابع از 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

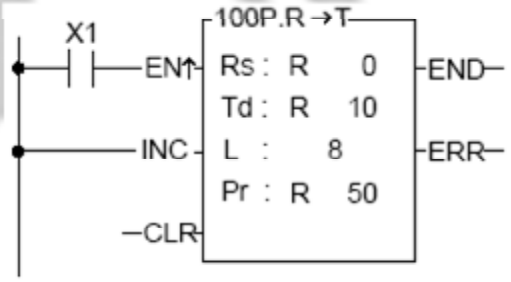
32bit table

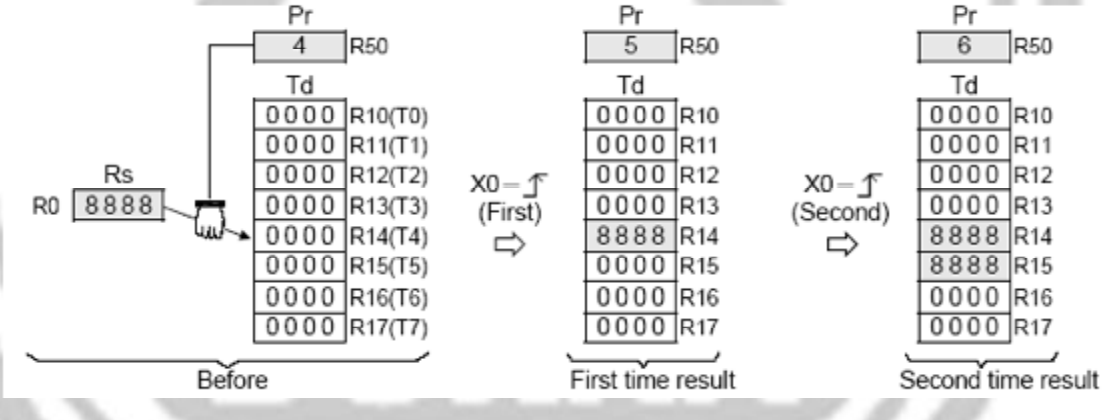
Function 100.REGISTER TO TABLE MOVE



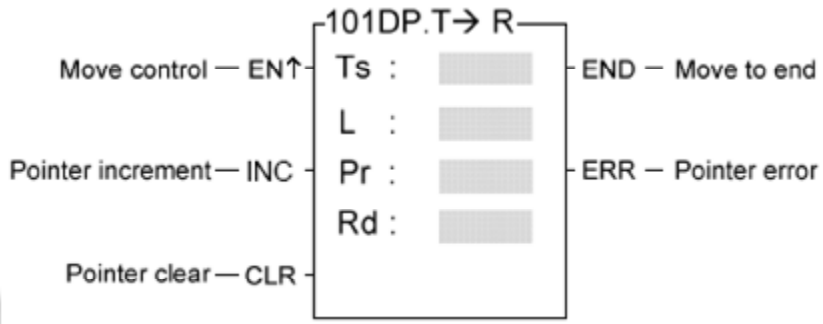
هرگاه "EN" از 0 به 1 تغییر کند: محتویات رجیستر مبدا (Rs) به داخل رجیستری از جدول ریخته می شود که Pointer به آن اشاره می کند. در Td رجیستر شروع جدول قرار می گیرد به طول (L). هرگاه سیگنال ورودی "CLR"-1 بشود ، مقدار اشاره گر(Pr) را ریست می کند. اگر مقدار Pr به L-1 برسد(یعنی به آخرین رجیستر جدول اشاره کند) 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:



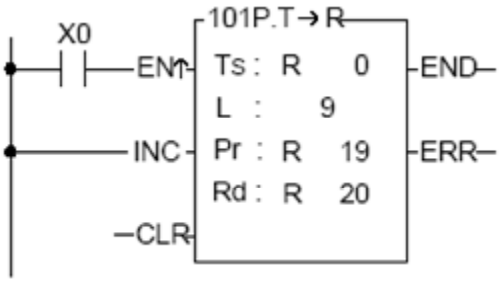


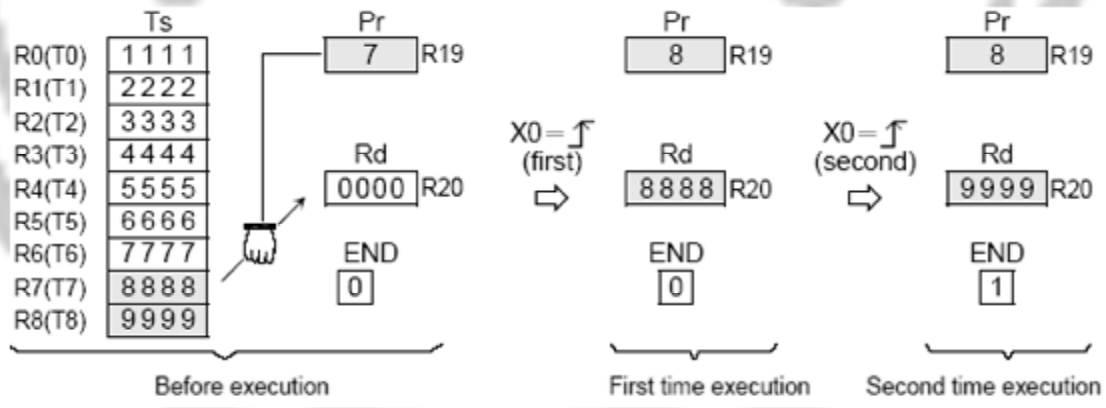
Function 101.TABLE TO REGISTER MOVE



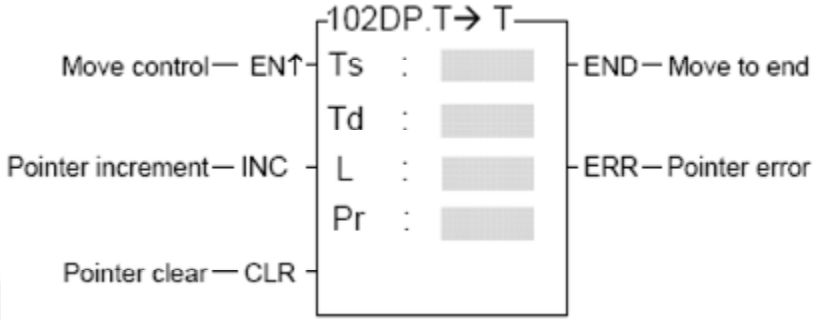
این تابع بر عکس تابع قبل عمل کرده و محتویات یک رجیستر از جدول مورد نظر را به رجیستر مقصد منتقل می کند.

مثال:



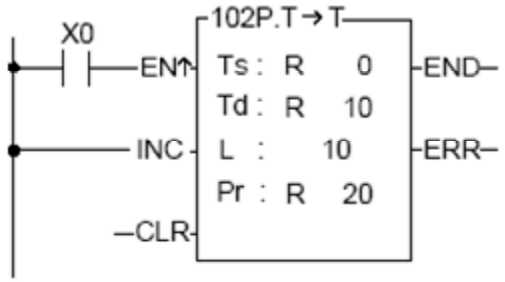


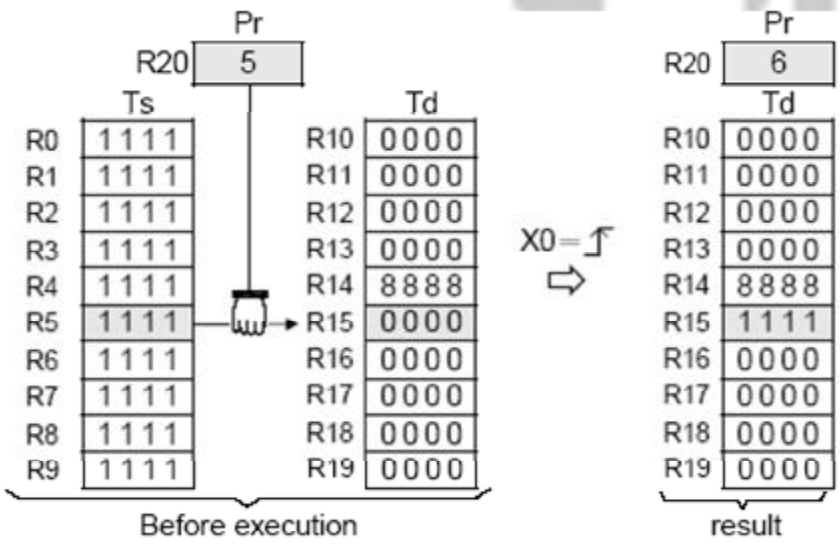
Function 102.TABLE TO TABLE MOVE



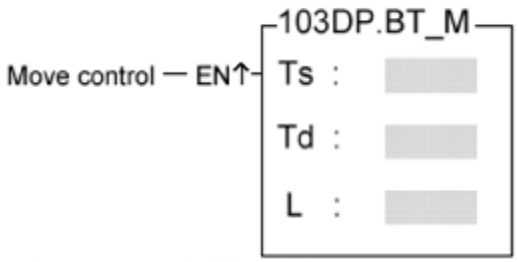
در Ts، رجیستر شروع جدول مبدا قرار می گیرد و در Td، ، رجیستر شروع جدول مقصد قرار می گیرد.

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.



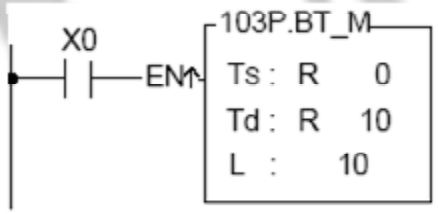


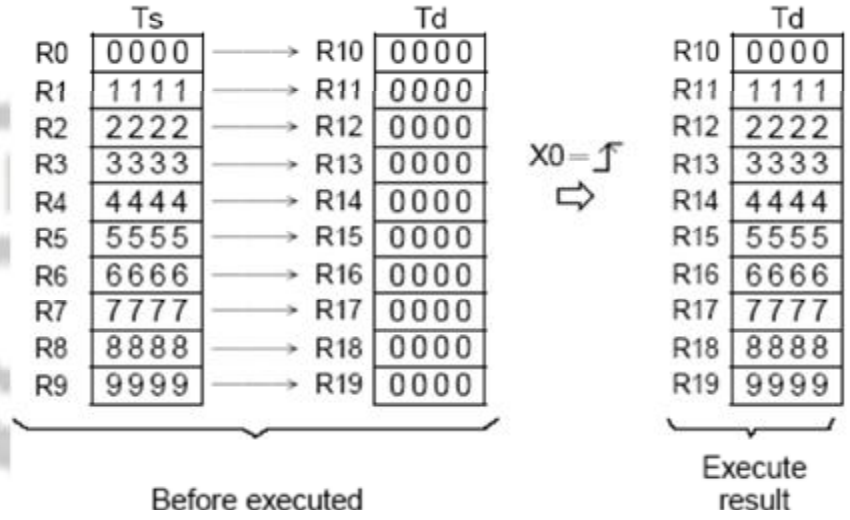
Function 103.BLOCK TABLE MOVE



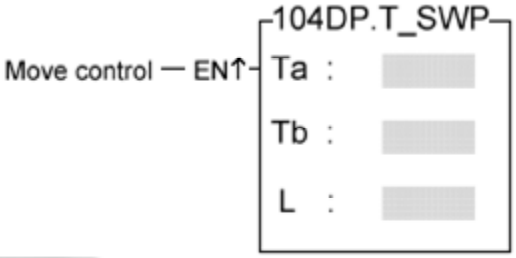
هرگاه "EN" از 0 به 1 تغییر کند: محتویات رجیسترهای جدول مبدا به داخل رجیسترهای معادلشان در جدول مقصد،کپی می شوند. L معرف طول هردو جدول است .

مثال:



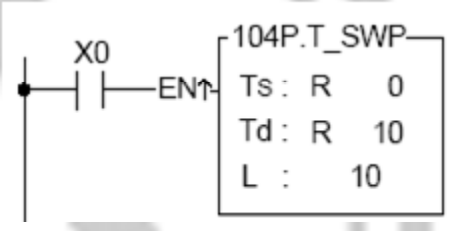


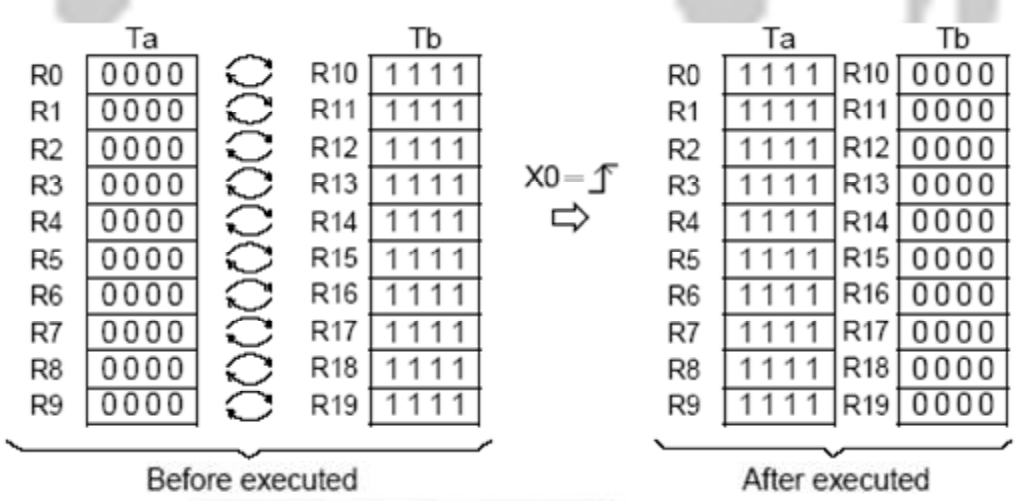
Function 104.BLOCK TABLE SWAP



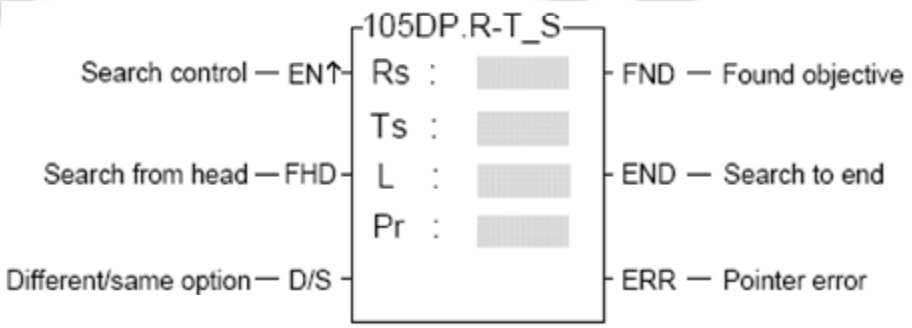
هرگاه "EN" از 0 به 1 تغییر کند: محتویات رجیسترهای جدول a با محتویات رجیسترهای معادلشان در جدول b ،جا به جا می شوند. L معرف طول هردو جدول است .

مثال:



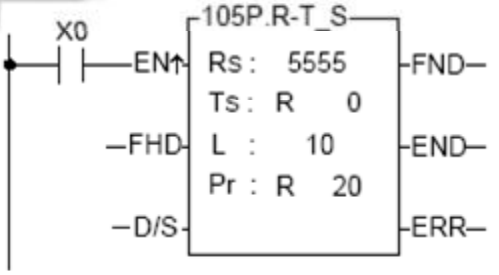


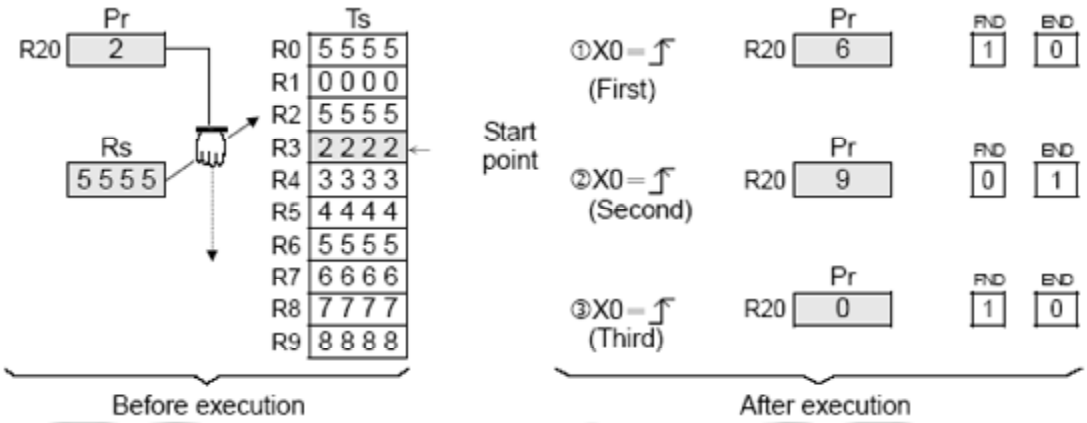
Function 105.REGISTER TO TABLE SEARCH



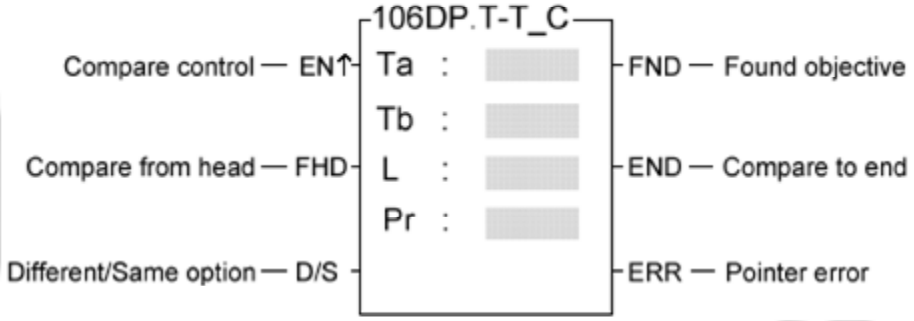
از این تابع برای یافتن مقداری مشخص در دسته ای از رجیسترهای متوالی ، استفاده می شود . هرگاه "EN" از 0 به 1 تغییر کند: وقتی 1="FHD" باشد یا مقدار اشاره گر (Pr) به L-1 رسیده باشد،جستجو از اولین رجیستر جدول Ts آغاز می شود. وقتی 0="FHD" و مقدار Pr کمتر از L-1 باشد،جستجو از ادامه رجیستری که Pr به آن اشاره 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:

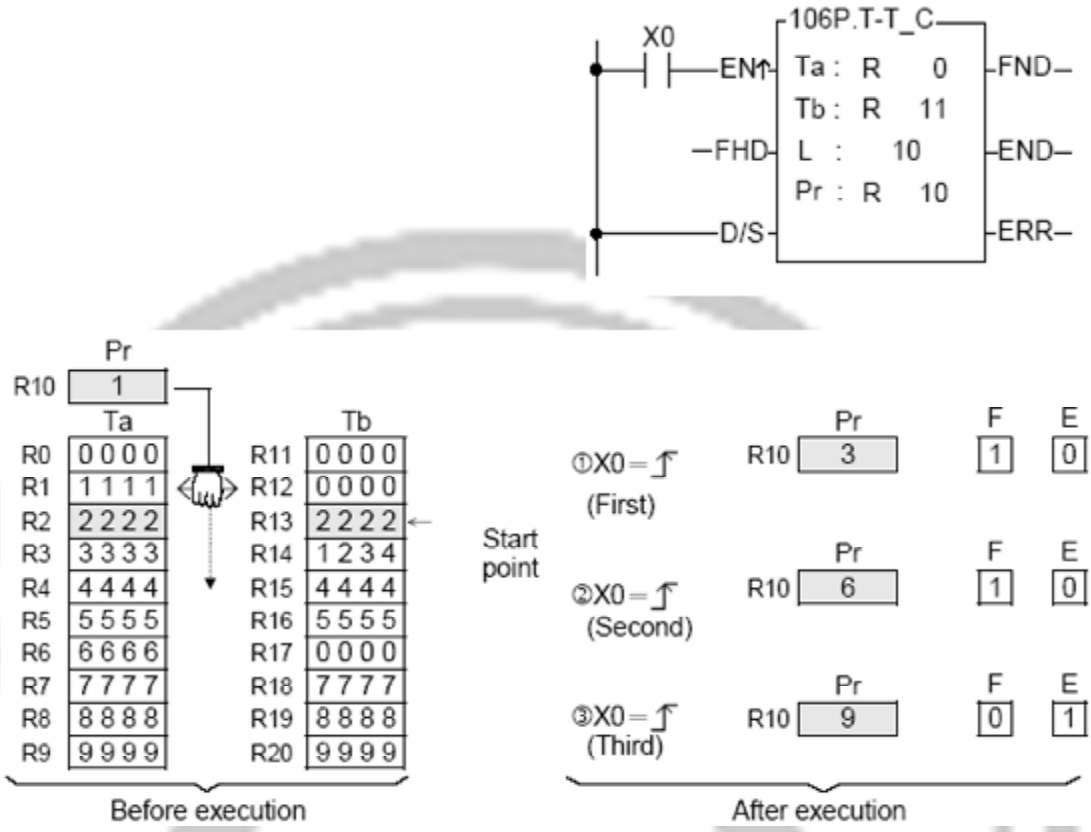




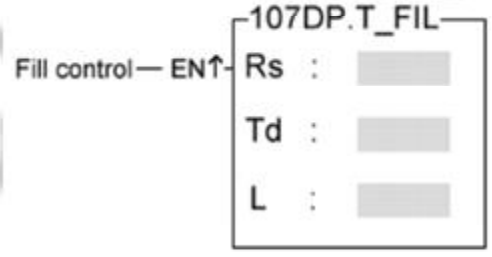
Function 106.TABLE TO TABLE COMPARE



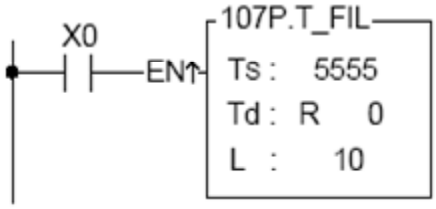
عملکرد این تابع مانند تابع قبل است با این تفاوت که محتوای رجیستر های معادل از دو جدول Ta و Tb با هم مقایسه می شوند. وقتی "D/S"=1 ،جستجو برای یافتن اولین دو رجیستر معادل در دو جدول که محتوای آن ها با هم متفاوت باشد ، صورت می گیرد. وقتی "D/S"=0 ، جستجو برای یافتن اولین دو رجیستر معادل در دو جدول که محتوای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

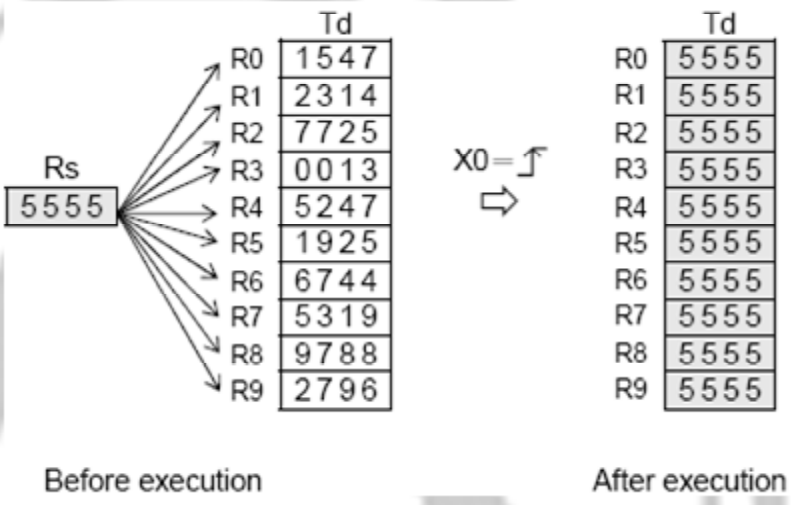


Function 107.TABLE FILL

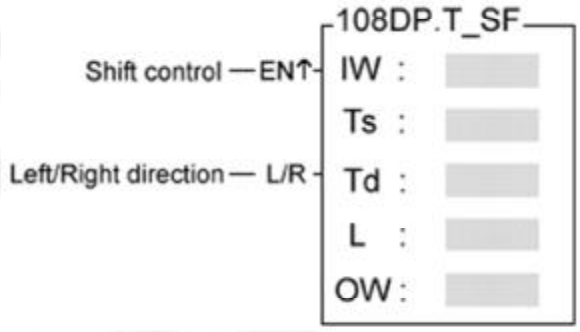


هرگاه "EN" از 0 به 1 تغییر کند: محتوای رجیستر Rs، تمام رجیسترهای جدول Td را پر می کند. کاربرد اصلی این تابع در صفر کردن کل یک جدول یا یکی کردن محتوای تمام رجیستر های یک جدول است. L معرف طول جدول است .



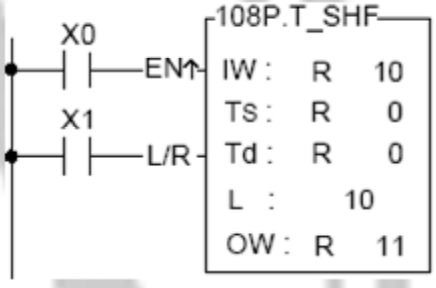


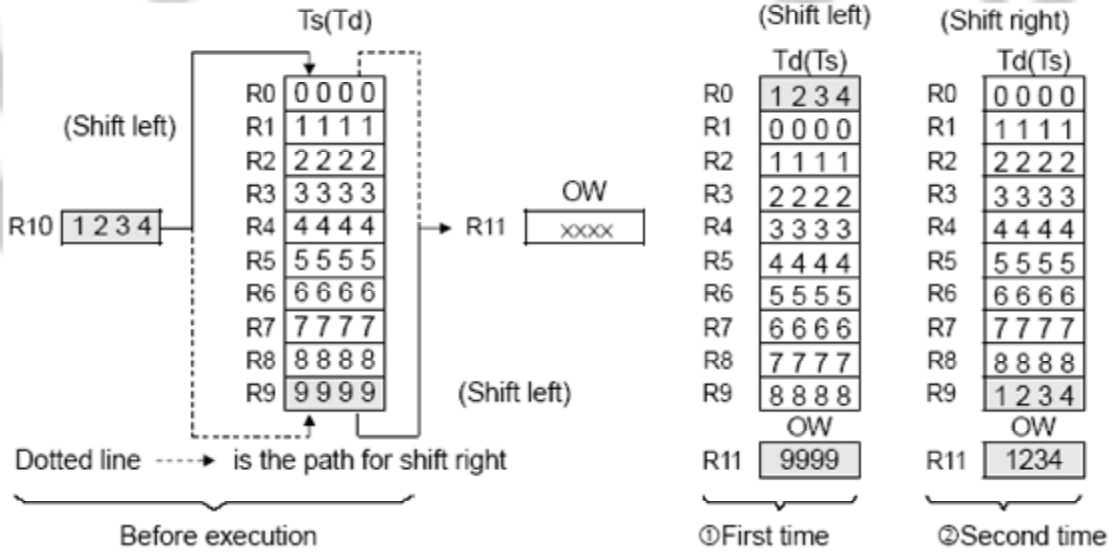
Function 108.TABLE SHIFT



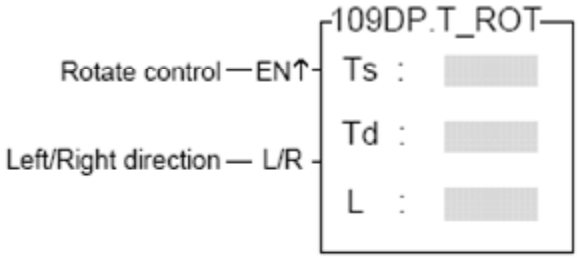
هرگاه "EN" از0 به 1 تغییر کند: محتوای رجیسترهای جدول Ts ، یک رجیستر به سمت چپ یا راست ، شیفت پیدا می کنند.وقتی 1="L/R" شیفت به چپ و وقتی 0="L/R" شیفت به راست صورت می گیرد. فضای خالی ایجاد شده بعد از اعمال شیفت ، توسط رجیستر یا عدد ثابت مشخص شده در IW ، پر می شود. محتوای رجیستری که پس از اعمال شیفت از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



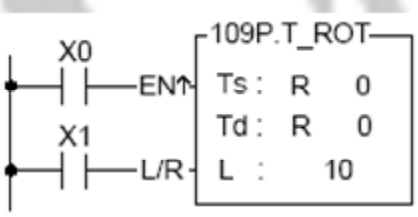


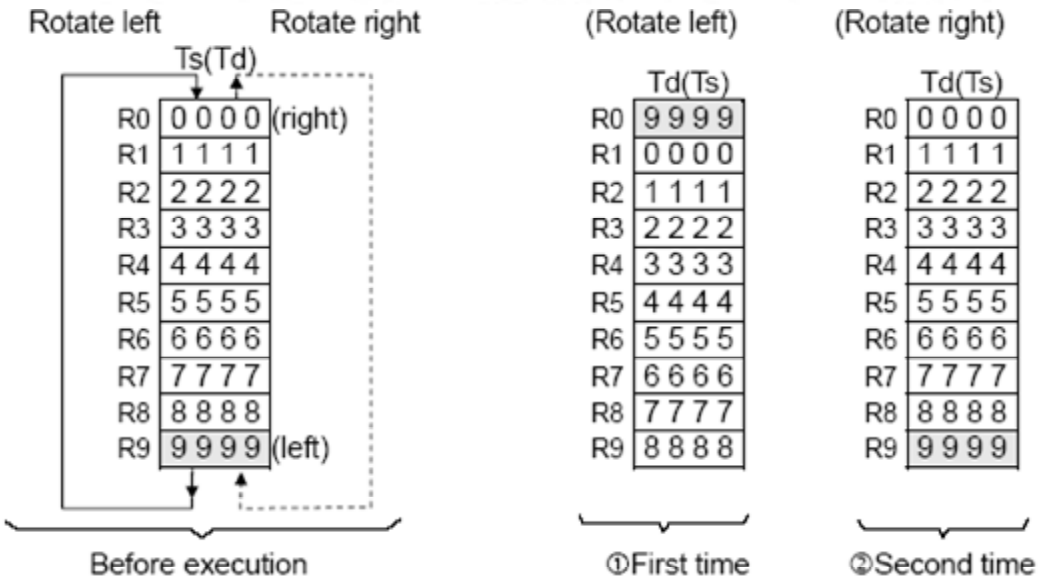
Function 109.TABLE ROTATE



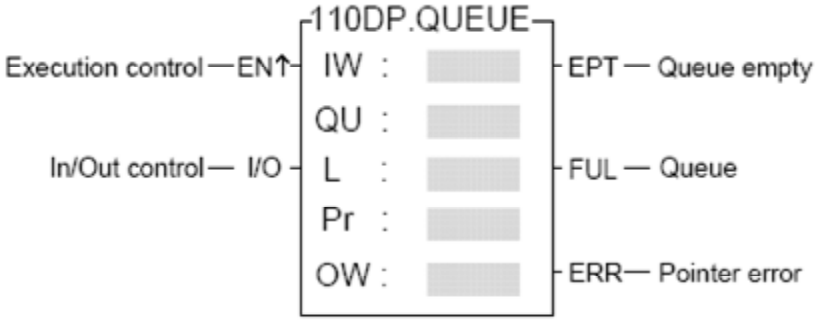
هرگاه "EN" از 0 به 1 تغییر کند: محتوای رجیسترهای جدول Ts ، یک رجیستر به سمت چپ یا راست می چرخند و نتیجه در Td ذخیره می شود. وقتی 1="L/R" ،چرخش به چپ صورت می گیرد . وقتی 0="L/R" ،چرخش به راست صورت می گیرد. L معرف طول جدول است .

مثال:





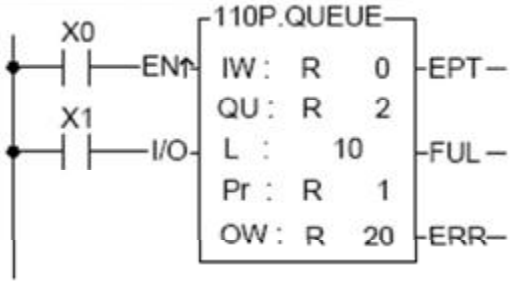
Function 110.QUENE

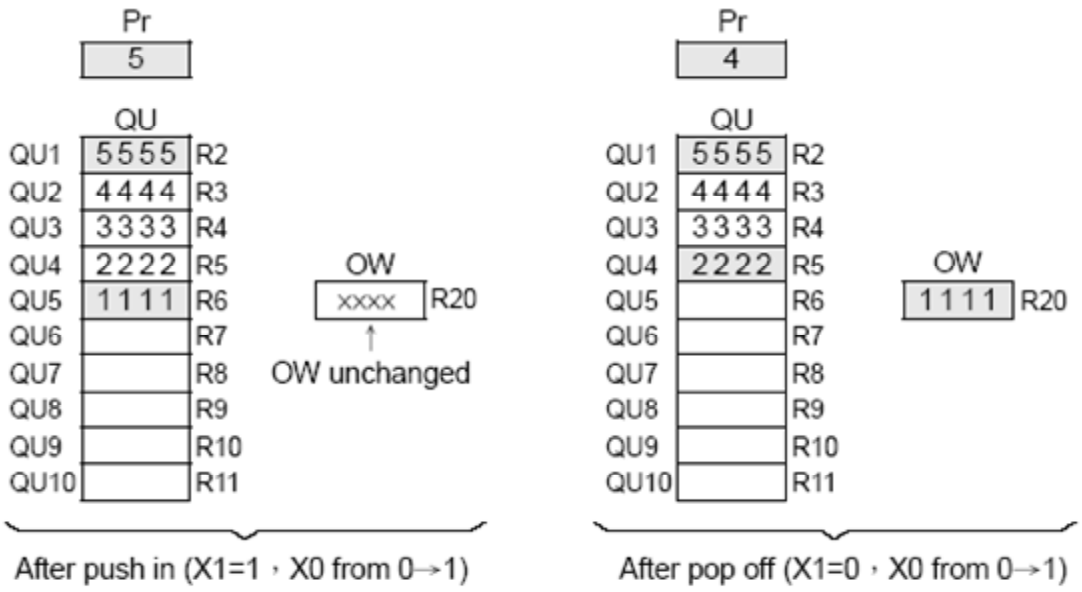


QUEUE نیز نوعی جدول است و همانطور که از نام آن پیداست ، عملکرد آن مانند یک صف می ماند . با کمک این تابع، همان داده ای که ابتدا در **QUEUE** وارد می شود ،(عمل **Push**) ،اولین داده ای خواهد بود که از **QUEUE** خارج می شود.(عمل **Pop**) . شماره رجیسترهای جدول معمولی از **0~L-1** است اما شماره خانه های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

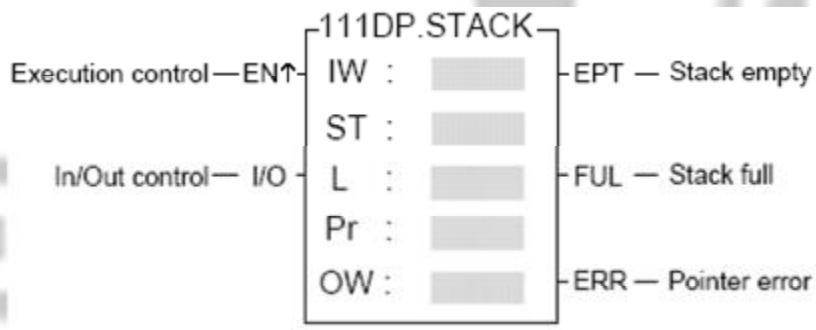
یک **QUEUE** از **L** رجیستر **16** یا **32** بیتی تشکیل شده که از رجیستر مشخص شده در **QU** شروع می شوند. هرگاه ورودی **1="I/O"** ، محتوای **IW** به داخل **QUEUE**، **push** می شود و هرگاه **0="I/O"** ،اولین داده ی درون **QUEUE** (قدیمی ترین) به داخل **OW**، **pop** خواهد شد. محتوای **IW** ،همیشه به 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:



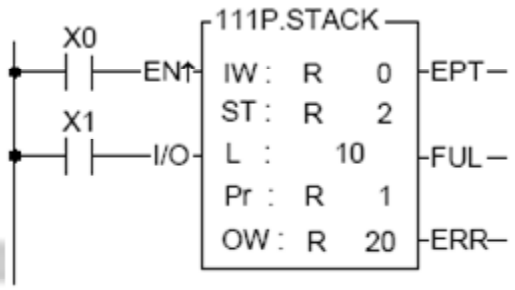


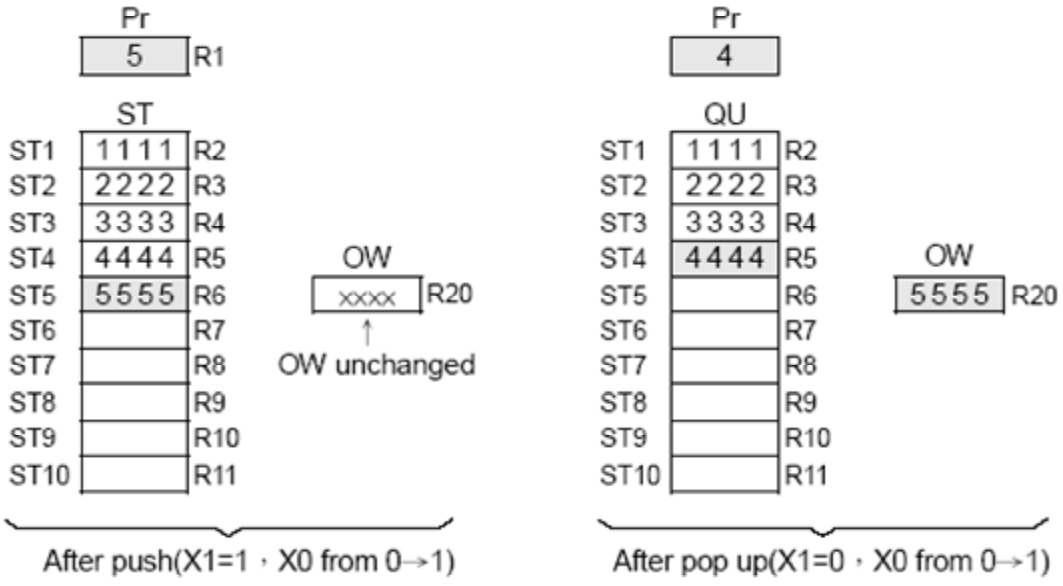
Function 111.STACK



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.

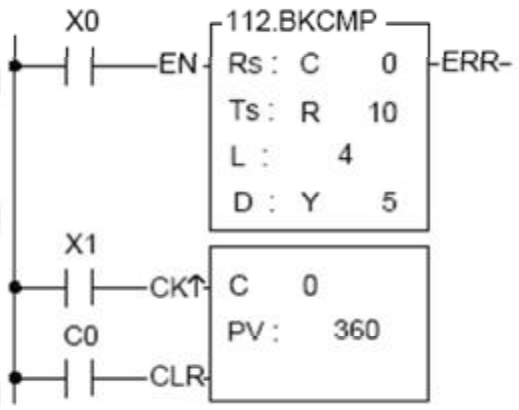
	Upper limit	Lower limit	Compare	Compared value	Result
0	T_{S1}	T_{S0}	⟷	Rs	D_0
1	T_{S3}	T_{S2}	⟷		D_1
≀	≀	≀	≀		≀
L–1	T_{S2L-1}	T_{S2L-2}	⟷		D_{L-1}

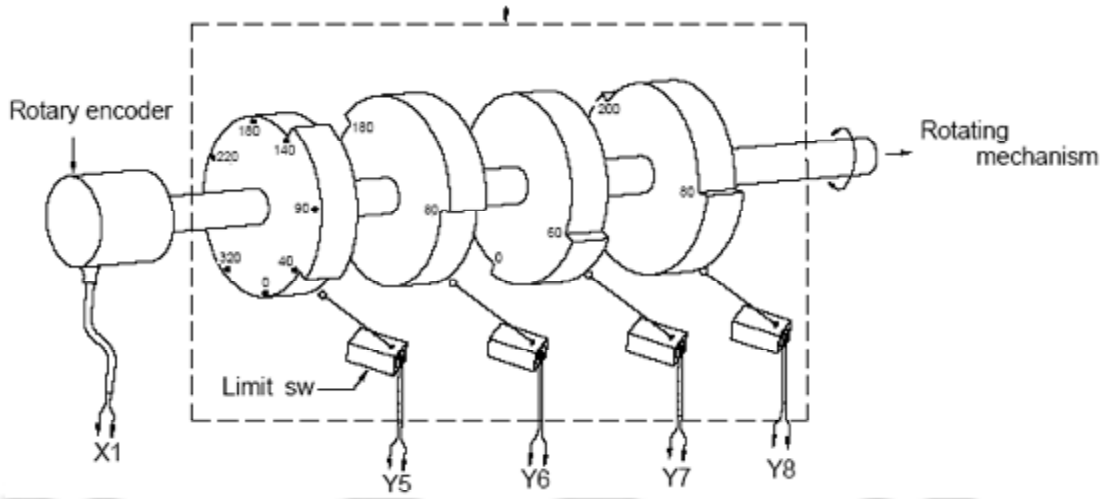
L در این تابع تعداد بازه ها را مشخص می کند. مقایسه تا زمانی ادامه می یابد که Rs با تمام بازه ها مقایسه شود.

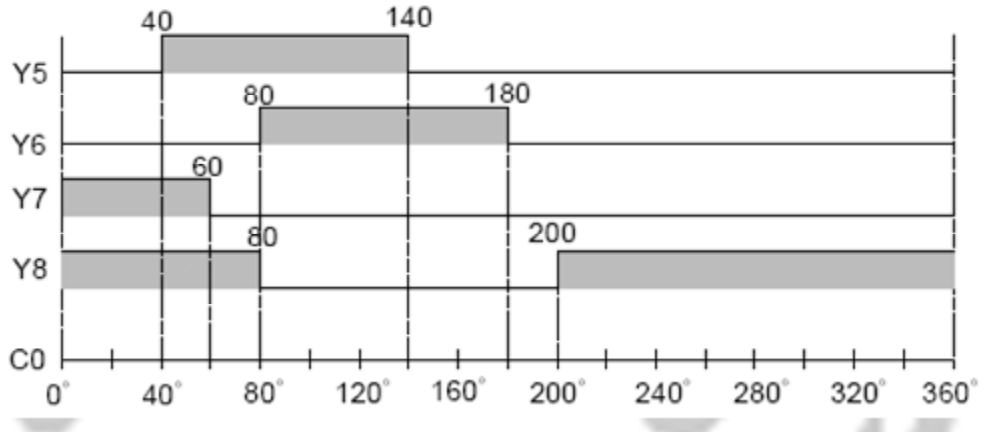
هرگاه بیت M1975=0 باشد، در یک بازه ، حد بالا نمی تواند کوچک تر از حد پایین باشد،در غیر این صورت خروجی "ERR" فعال شده و نتیجه مقایسه برای آن بازه ،0 خواهد شد. هرگاه بیت M1975=1 باشد، ،محدودیتی برای بزرگتر یا کوچکتر بودن حد بالا نسبت به حد پایین وجود نداشته و تنها کافیست مقدار Rs در بازه باشد تا رجیستر 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.

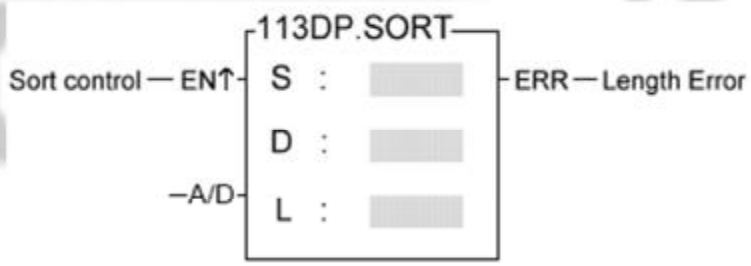
مثال:





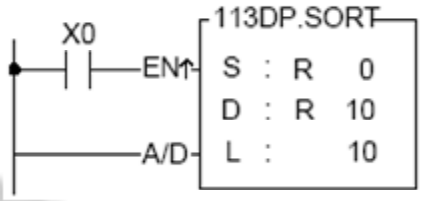


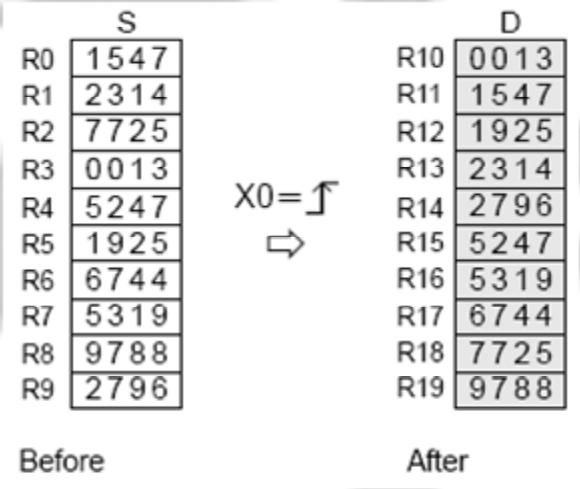
Function 113.DATA SORTING



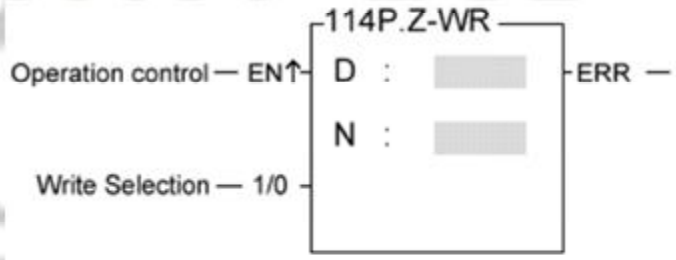
هرگاه "EN" از0 به 1 تغییر کند: این تابع مقادیر رجیستر ها به تعداد L را که از S شروع می شوند ،به صورت افزایشی یا کاهشی مرتب می کند و نتیجه را در D ذخیره می کند. اگر 1 "A/D" رجیسترها به صورت افزایشی مرتب می شوند و اگر 0="A/D" رجیسترها به صورت کاهشی مرتب می شوند . L باید در رنج 127~2 باشد در غیر این صورت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

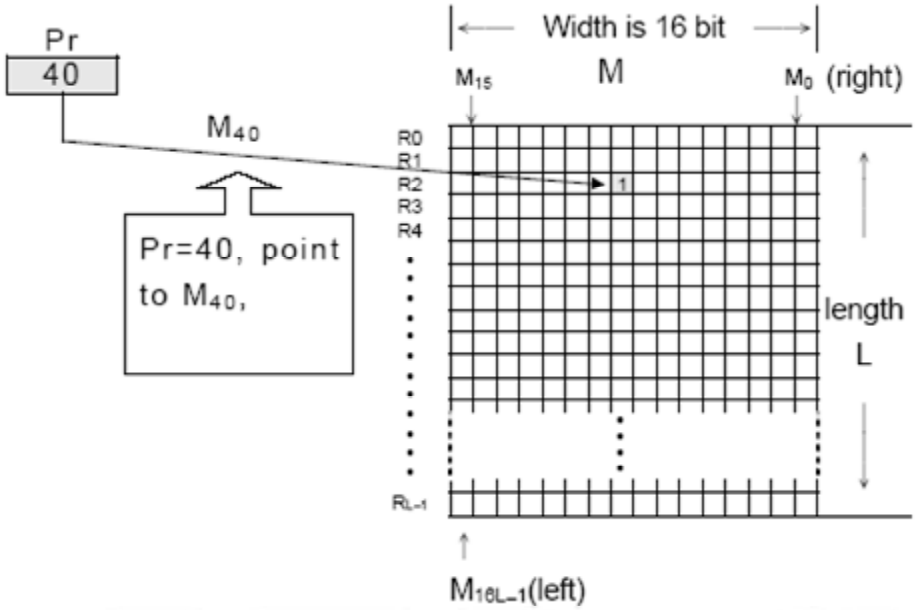
مثال:

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

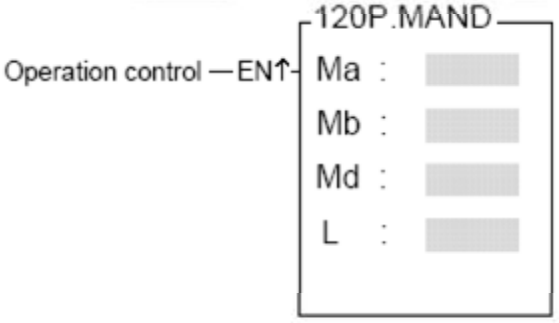
توابع ماتریسی

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			

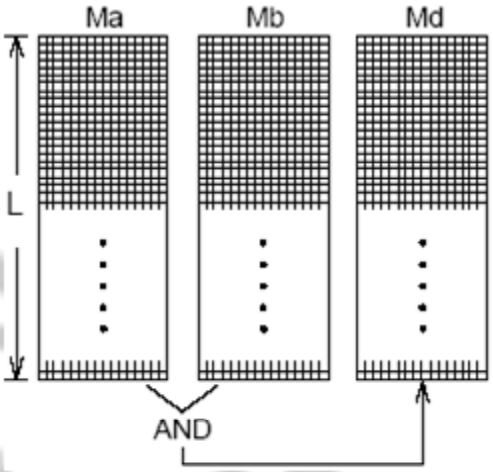
یک ماتریس از 2 یا چند رجیستر پی در پی 16 بیتی تشکیل شده است.به تعداد رجیسترهای تشکیل دهنده ماتریس، طول ماتریس گفته و با (L) نمایش می دهند. پس یک ماتریس 16×L بیت (آرایه) دارد و واحد اصلی اجرای این توابع ،بیت می باشد. توابع ماتریسی عمدتاً برای پردازش گسسته استفاده می شوند مانند انتقال ،کپی، مقایسه ، جستجو 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.



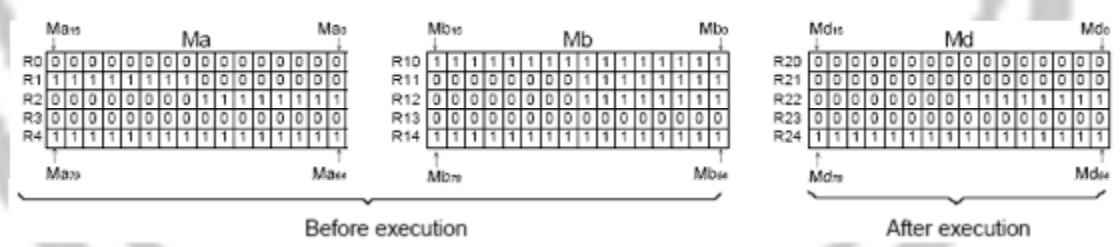
Function 120.MATRIX AND



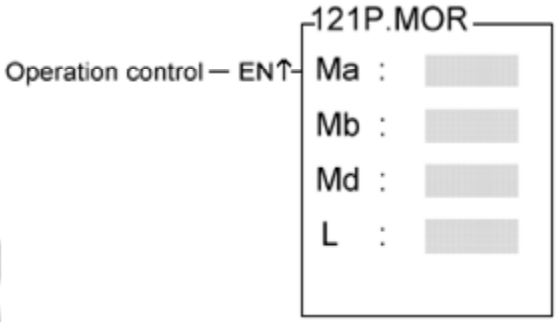
هرگاه "EN" از 0 به 1 تغییر کند: بیت های متناظر در ماتریس Ma و Mb را با یکدیگر AND کرده و نتیجه در Md ریخته می شود.



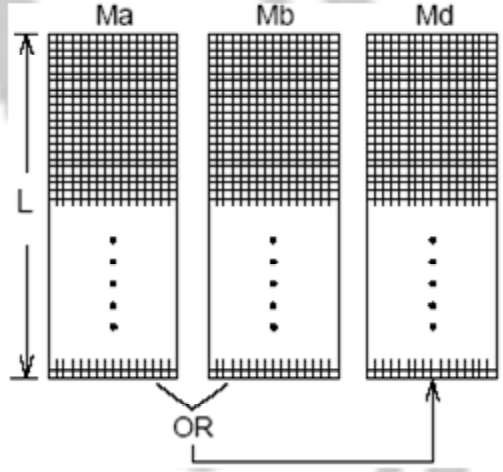
مثال:



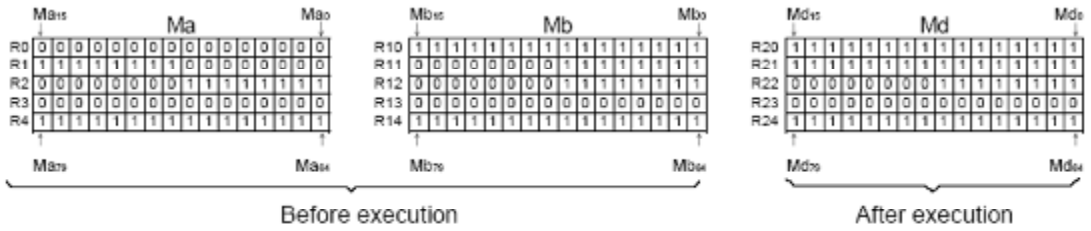
Function 121.MATRIX OR



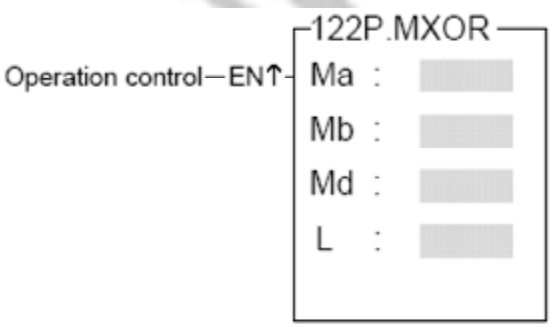
هرگاه "EN" از 0 به 1 تغییر کند: بیت های متناظر در ماتریس Ma و Mb را با یکدیگر OR کرده و نتیجه در Md ریخته می شود.



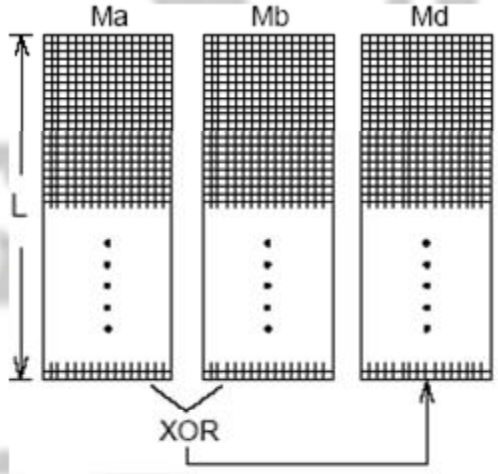
مثال:



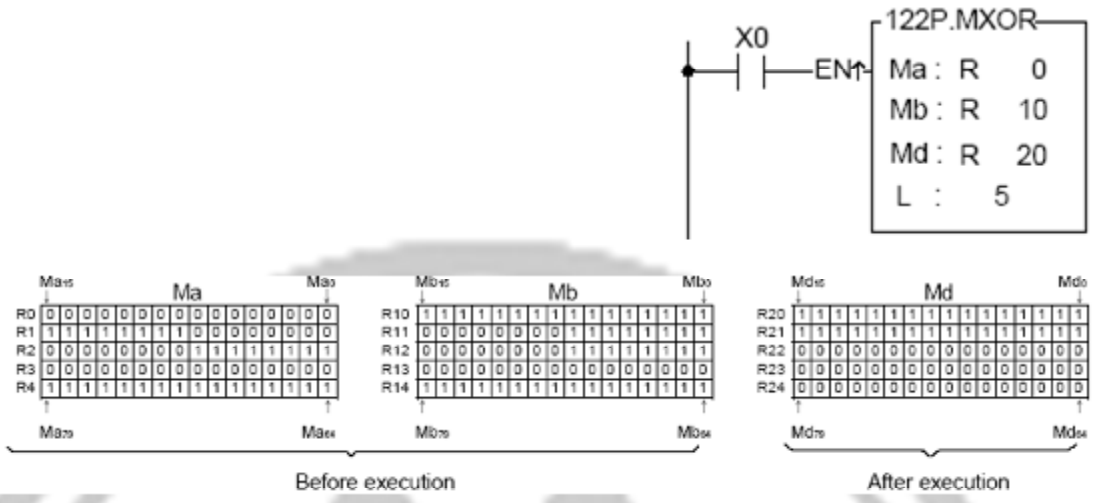
Function 122.MATRIX XOR



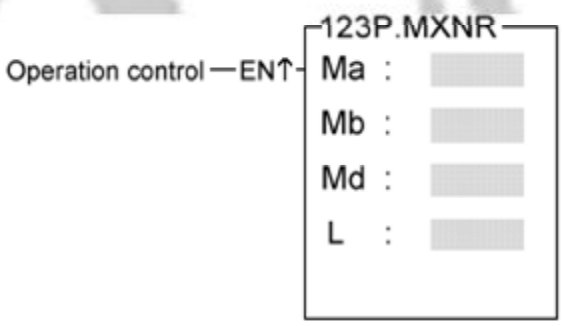
هرگاه "EN" از 0 به 1 تغییر کند: بیت های متناظر در ماتریس Ma و Mb را با یکدیگر XOR کرده و نتیجه در Md ریخته می شود.



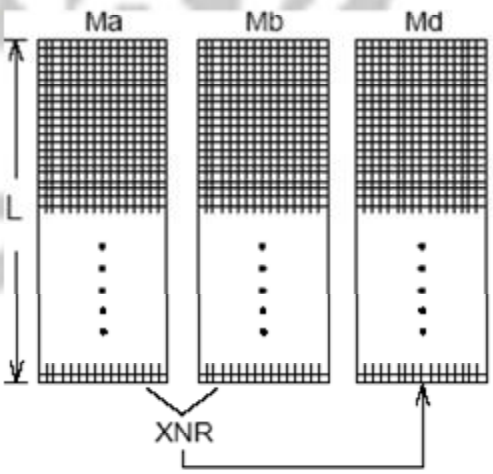
مثال:



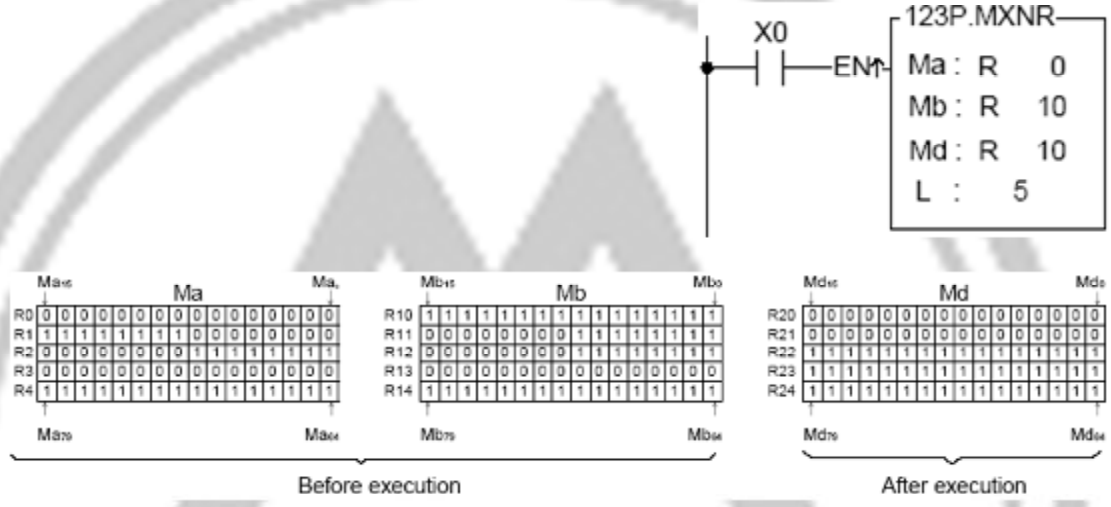
Function 123.MATRIX XNOR



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.

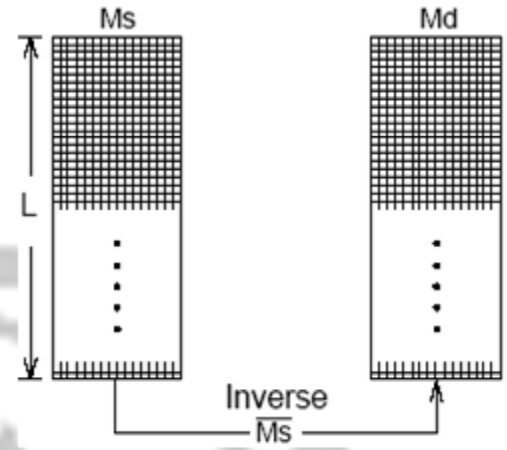


مثال:

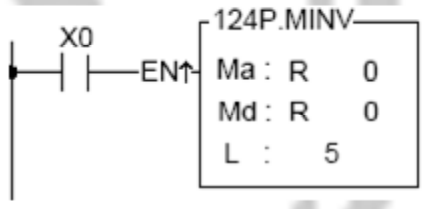


Function 124.MATRIX INVERSE

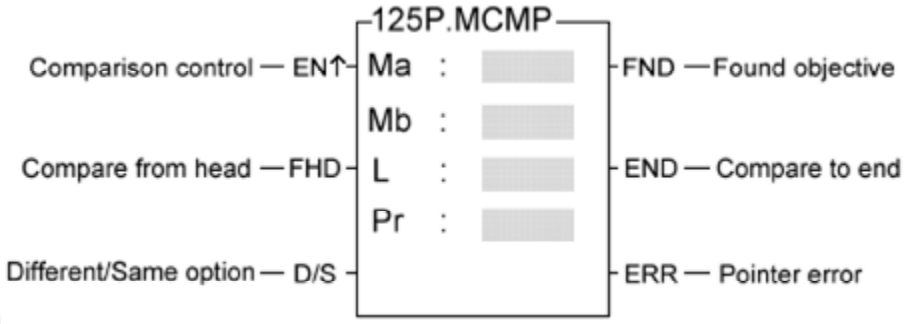
هرگاه "EN" از 0 به 1 تغییر کند: تمام بیت های ماتریس Ms معکوس می شوند(0 ها 1 شده و 1 ها 0 می شوند) و نتیجه در Md ذخیره می شود.



مثال:

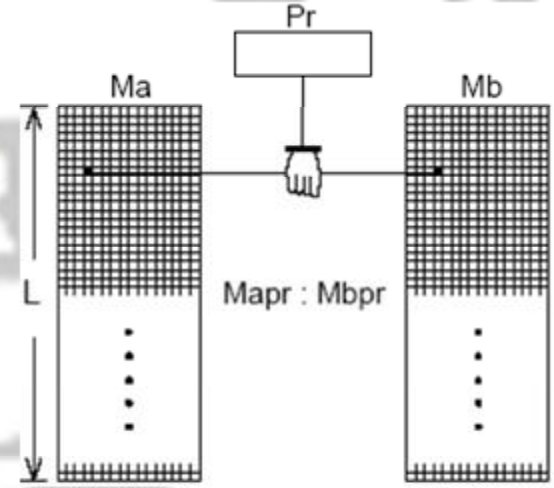


Function 125.MATRIX COMPARE

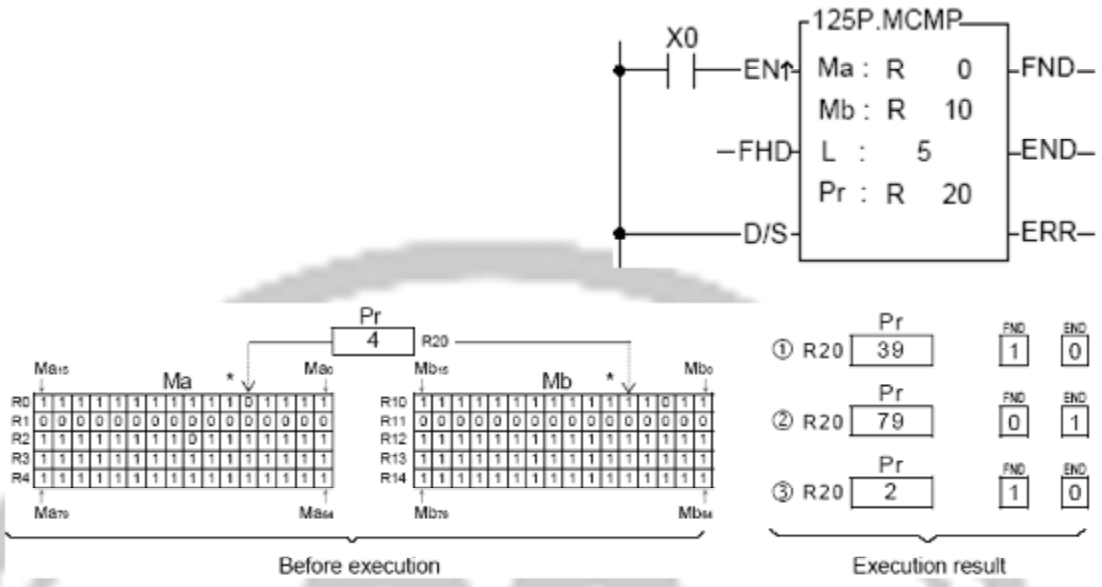


هرگاه "EN" از0 به 1 تغییر کند: مقایسه بین دو بیت متناظر از ماتریس های Ma و Mb آغاز می شود. اگر ورودی 1="FHD" باشد یا مقدار Pr به 16L-1 رسیده باشد،مقایسه از ابتدای ماتریس ها (اولین بیت) ، شروع می شود. وقتی 0="FHD" و مقدار Pr کمتر از 16L-1 باشد،مقایسه از اولین بیت ،بعد از جایی که Pr اشاره می کند (Pr+1)،شروع 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 .

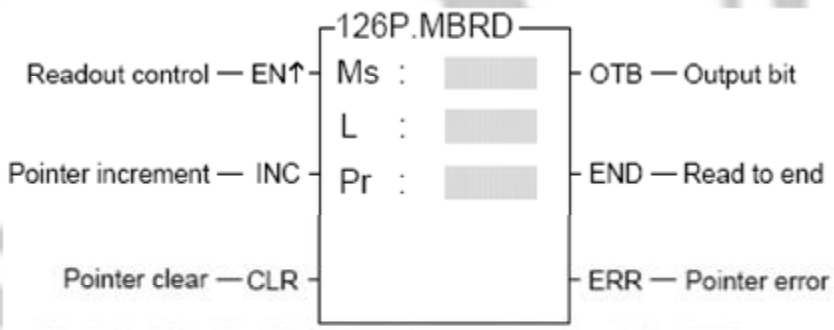
وقتی Pr به 16L-1 رسید،چه بیت های مورد نظر را یافته باشد چه نیافته باشد،خروجی "END" فعال شده و مقایسه متوقف می شود. اگر مقدار اشاره گر خارج از محدوده 16L-1~0 باشد ، خروجی "ERR" فعال می شود.



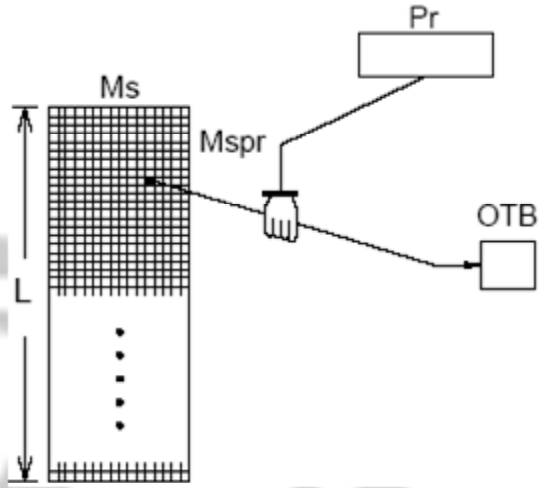
مثال:



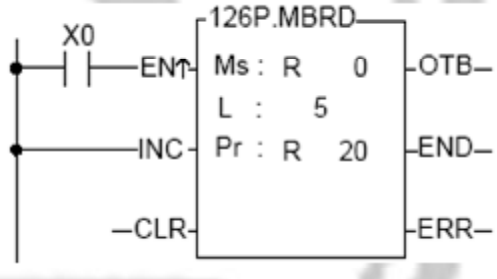
Function 126.MATRIX BIT READ

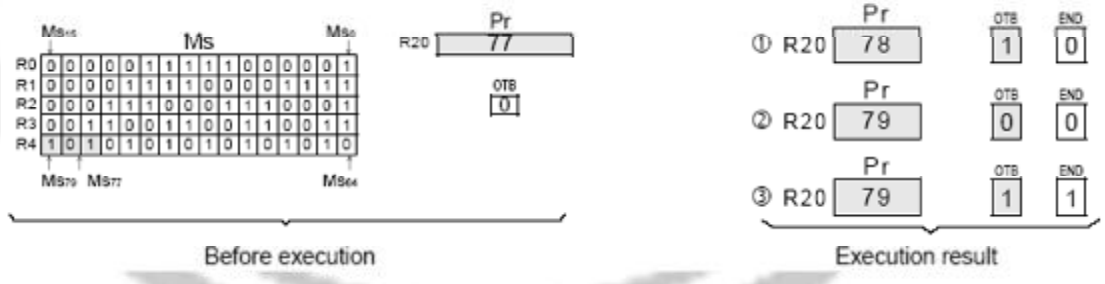


از این تابع برای خواندن مقدار یک بیت مشخص در میان دسته ای از رجیسترها ، استفاده می شود . هرگاه "EN" از 0 به 1 تغییر کند: ورودی "CLR" را چک کرده و اگر 1="CLR"باشد ، مقدار Pr را 0 می کند. مقدار بیتی که Pr به آن اشاره می کند،در خروجی "OTB" ظاهر می شود. اگر مقدار Pr به 16L-1 برسد(آخرین بیت ماتریس) خروجی "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.

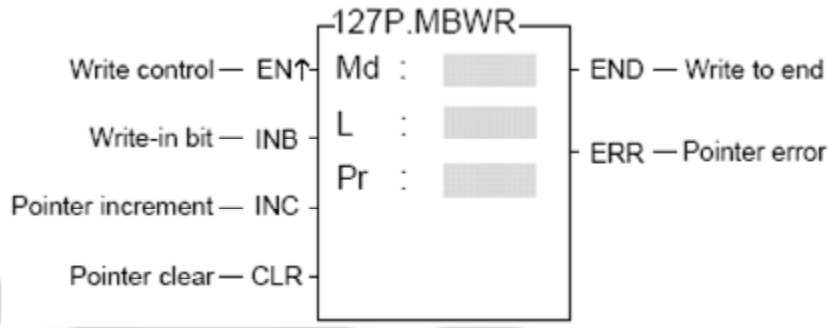


مثال:

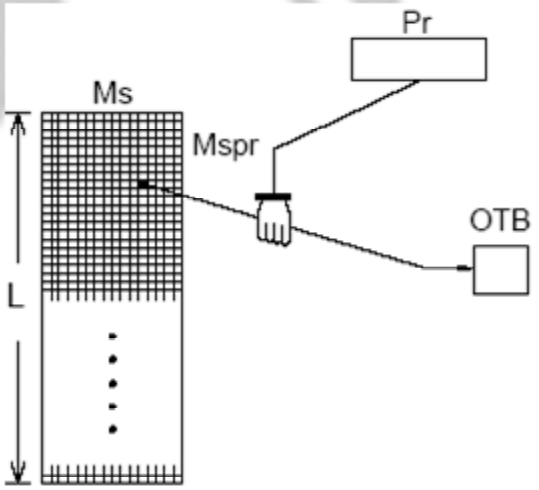




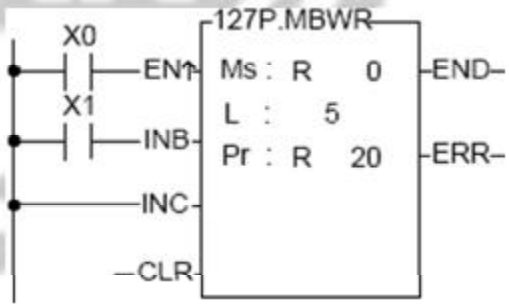
Function 127.MATRIX BIT WRITE

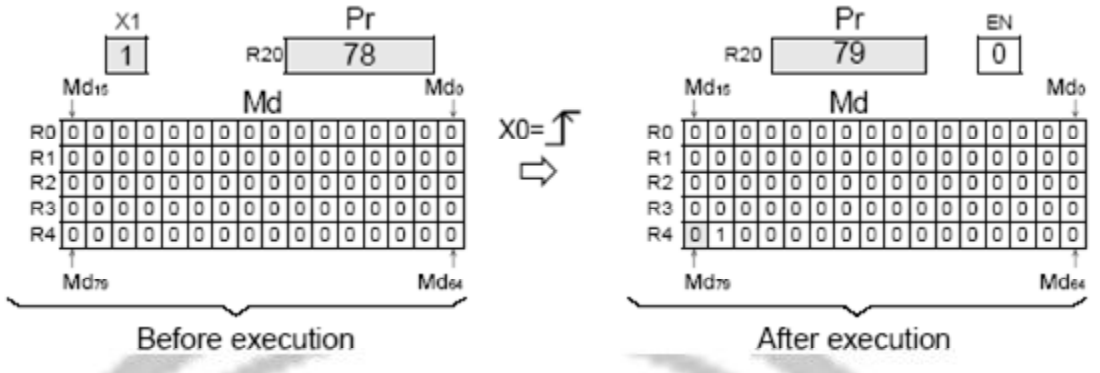


مانند تابع قبل است با این تفاوت که هنگام اجرا ، بیت مشخص شده در ورودی **"INB"** به روی بیتی که **Pr** اشاره می کند ، ریخته می شود.

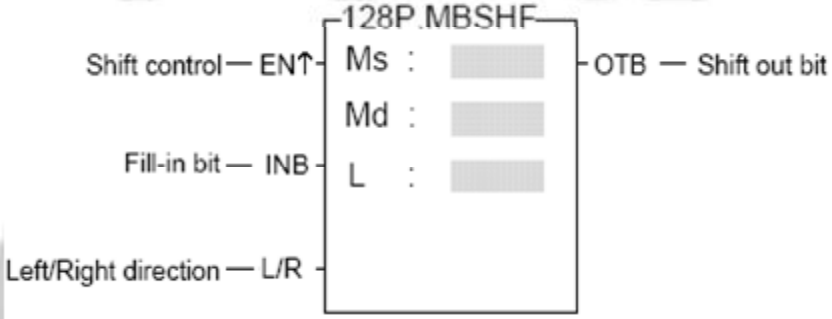


مثال:

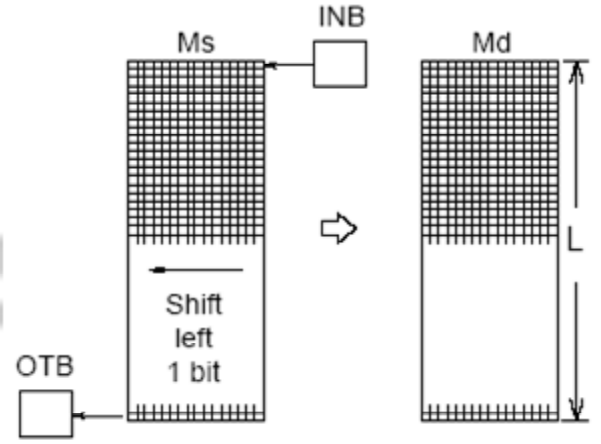


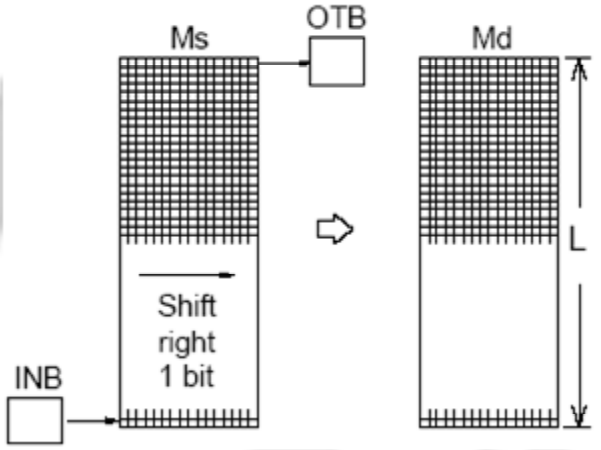


Function 128.MATRIX BIT SHIFT

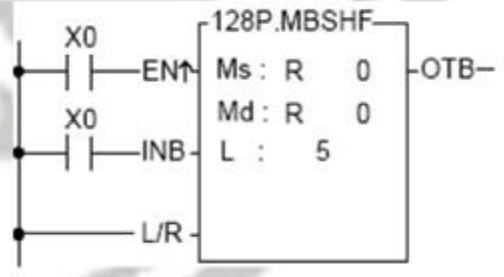


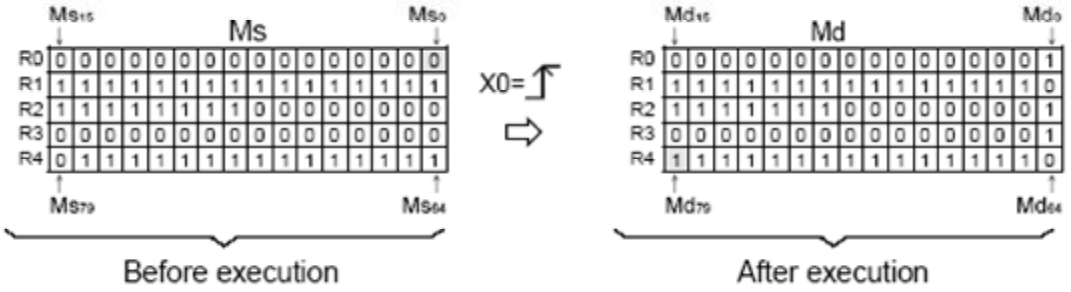
هرگاه "EN" از0 به 1 تغییر کند: بیت های ماتریس Ms، 1 بیت شیفت پیدا کرده و نتیجه در Md ذخیره می شود. اگر ورودی 1="L/R" باشد،شیفت به چپ صورت می گیرد و اگر ورودی 0="L/R" باشد،شیفت به راست صورت می گیرد . بیت خالی ایجاد شده بعد از اعمال شیفت ،توسط ورودی "INB" پر شده و محتوای بیتی که پس از اعمال شیفت از جدول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


INB
Ms
Md
L
Shift
left
1 bit
OTB

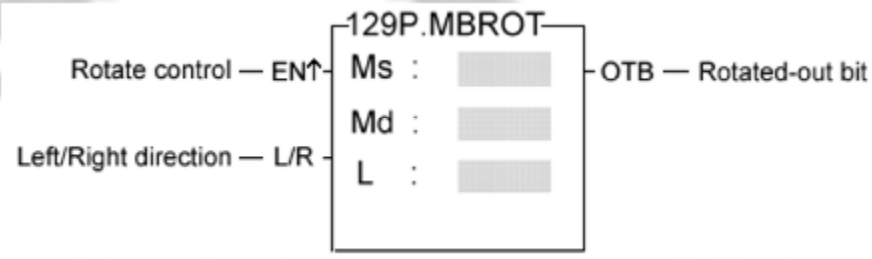

OTB
Ms
Md
L
Shift
right
1 bit
INB

مثال:


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

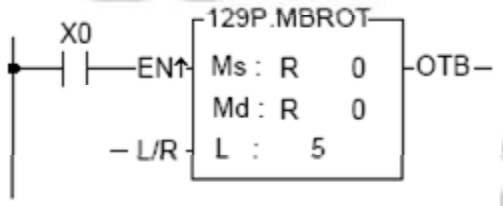

Ms
Md
X0=
Before execution
After execution

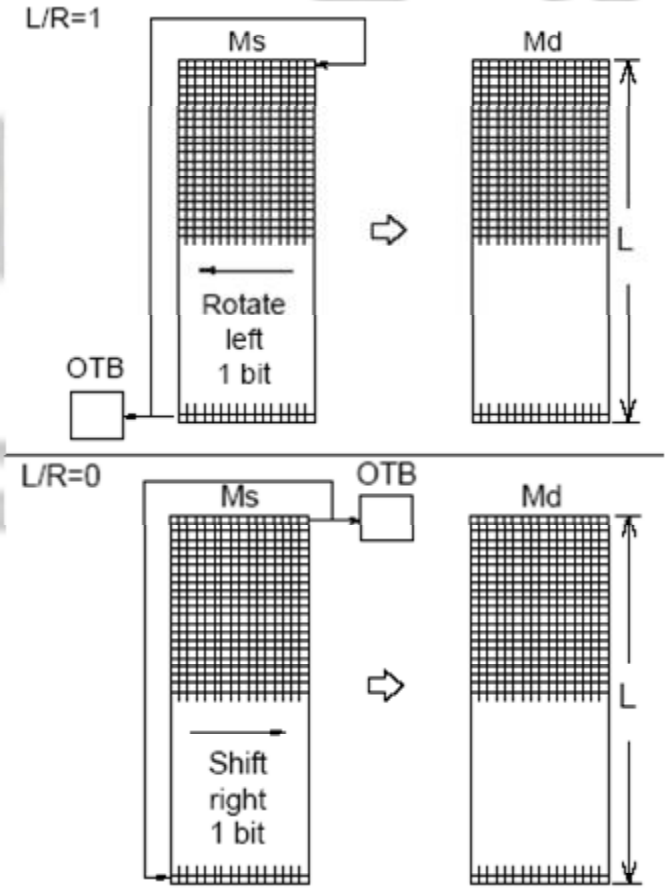
Function 129.MATRIX BIT ROTATE

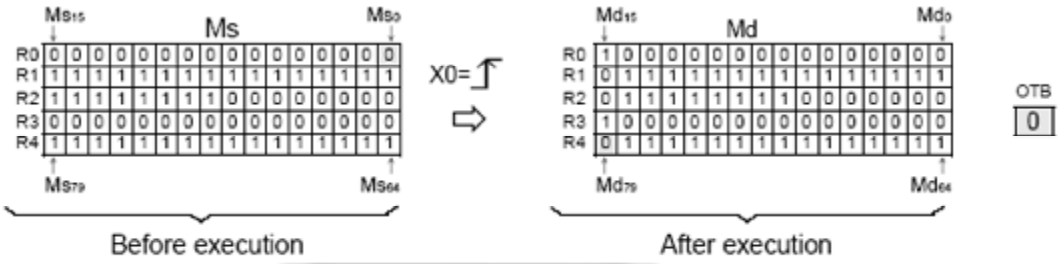


هرگاه "EN" از0 به 1 تغییر کند: بیت های ماتریس Ms ، یک بیت به سمت راست یا چپ می چرخند و نتیجه در ماتریس Md ذخیره می شود. وقتی ورودی 1="L/R"، چرخش به چپ صورت می گیرد. وقتی ورودی 0="L/R"، چرخش به راست صورت می گیرد. یک کپی از بیتی که بر اثر چرخش از یک سر ماتریس خارج شده و در سر دیگر آن قرار می گیرد،به خروجی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

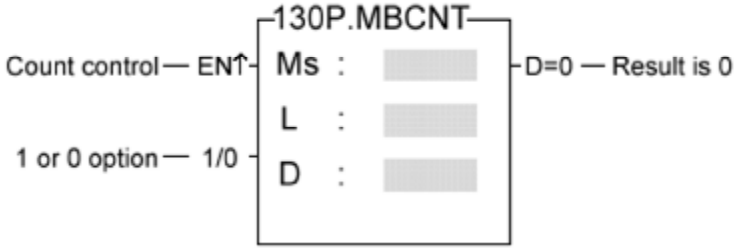
مثال:





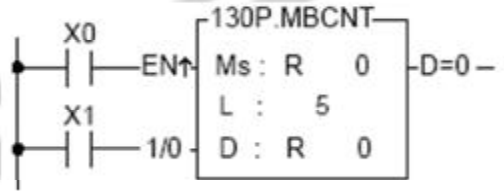


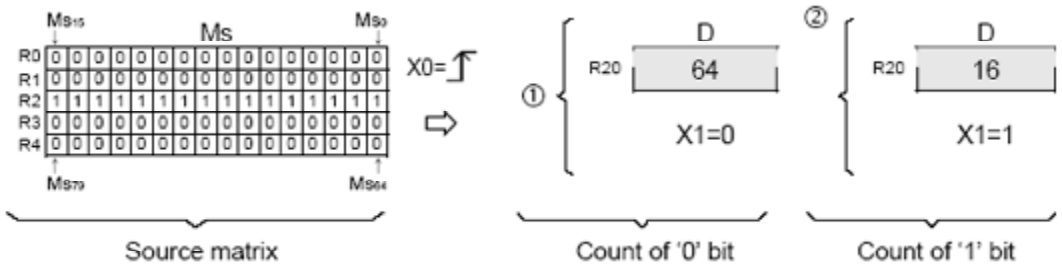
Function 130.MATRIX BIT STATUS COUNT



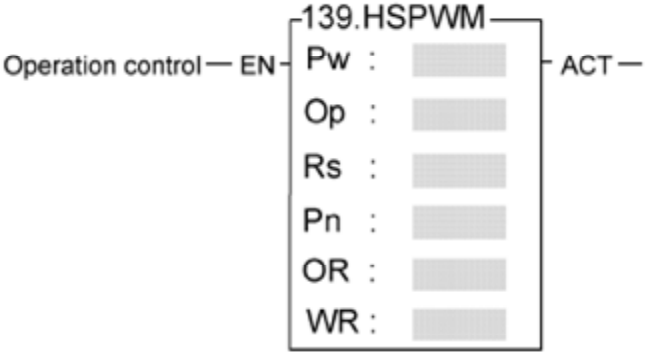
هرگاه "EN" از 0 به 1 تغییر کند: اگر 1="1/0" باشد،تعداد بیت های 1 موجود در ماتریس Ms را شمرده و تعداد در D ذخیره می شود. اگر 0="1/0" باشد،تعداد بیت های 0 موجود در ماتریس Ms را شمرده و تعداد در D ذخیره می شود. اگر هیچ تعداد از آن بیت مورد نظر موجود نباشد، خروجی "D=0" ، 1 خواهد شد.

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

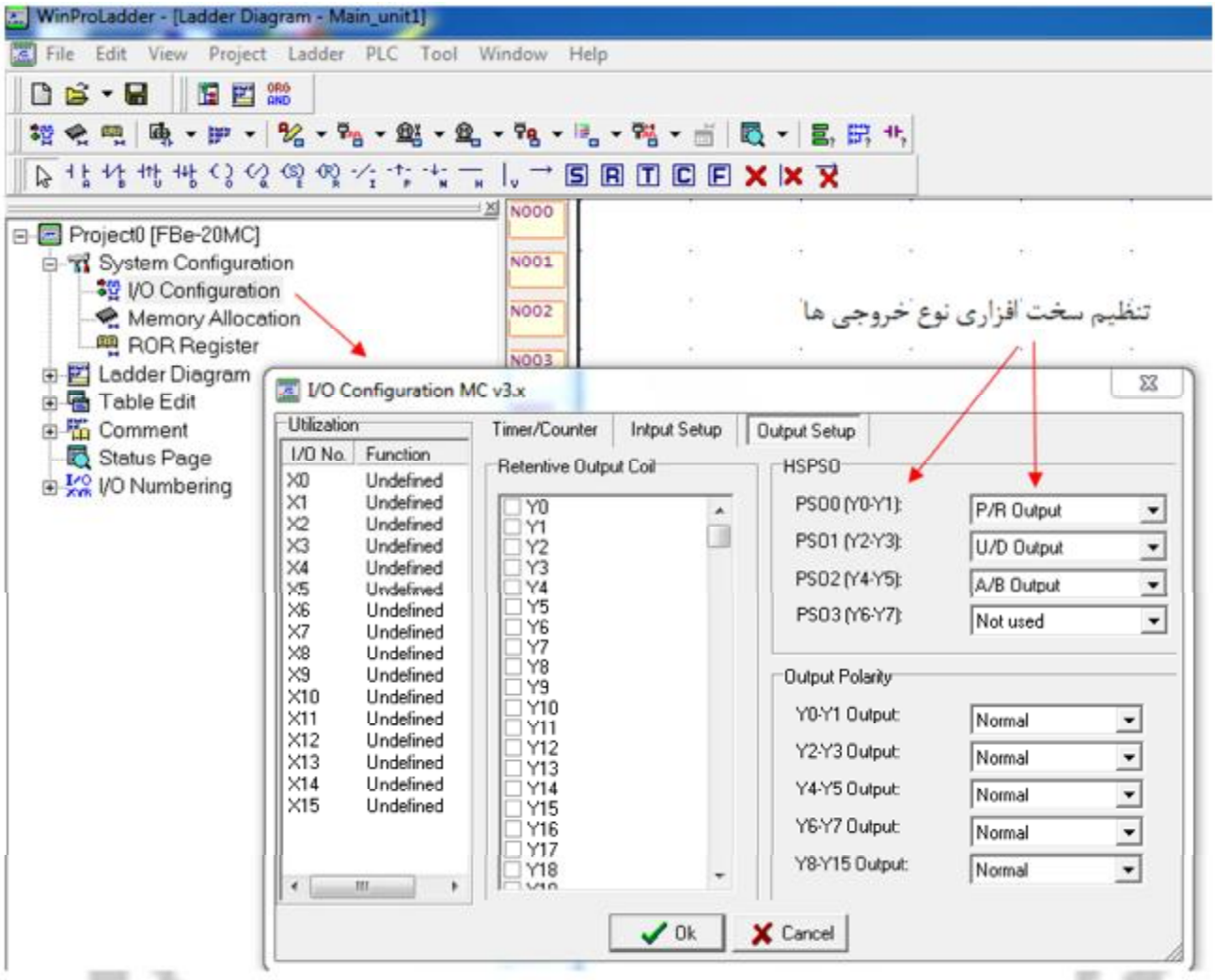
- **جدول زیر حالات مختلف خروجی ها را نشان می دهد**

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	

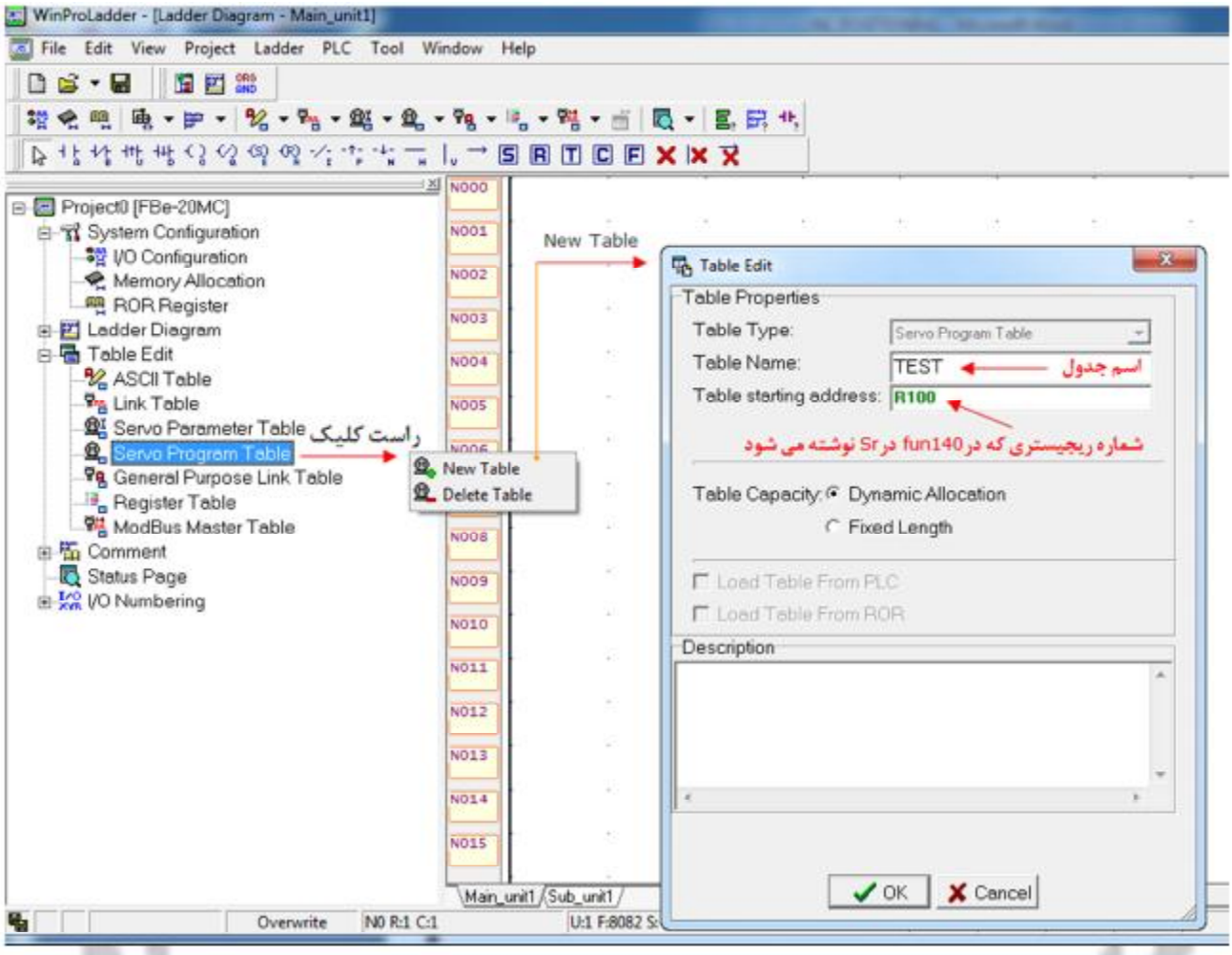
U/D	پالس بالا رونده در خروجی ظاهر میشود(Y0(Y2,Y4,Y6 پالس پایین رونده در خروجی ظاهر میشود(Y1(Y3,Y5,Y7
P/R	پالس درخروجی ظاهر میشود(Y0(Y2,Y4,Y6 جهت چرخش را تعیین می نماید(Y1(Y3,Y5,Y7 ON : شمارش بالا رونده OFF : شمارش پایین رونده
A/B	پالس با فازA در خروجی ظاهر میشود(Y0(Y2,Y4,Y6 پالس با فازB در خروجی ظاهر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

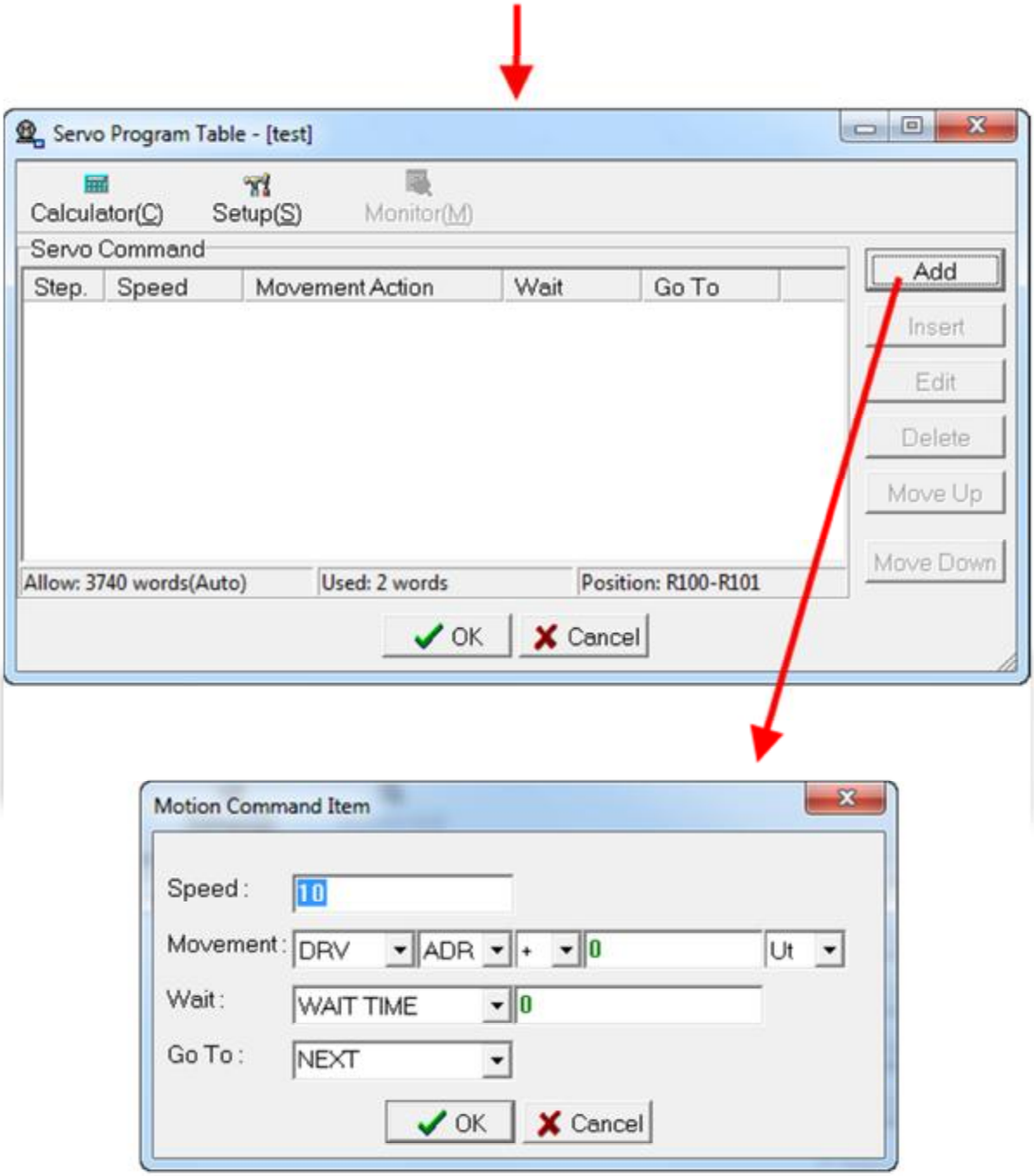
- نحوه تنظیم نمودن خروجیها توسط Winproladder

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:

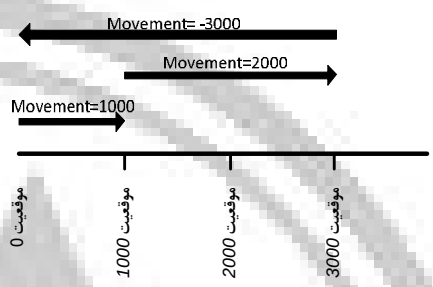
(رجیسترهایی که برای این جدول استفاده می شوند بصورت **32** بیتی می باشند)

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) :

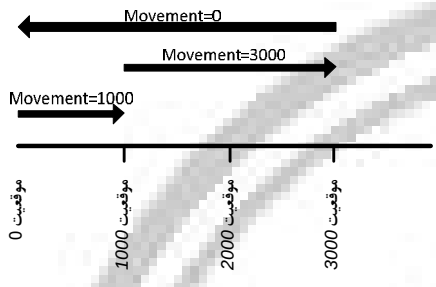
تعریف مد ADR

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

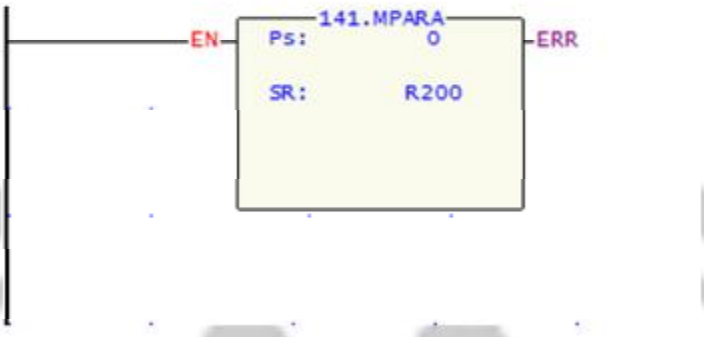
** در صورت **ABT=1** خروجی پالس بلافاصله قطع شده و هنگامیکه **EN=1** گردد ،دستورات جدول پارامترهای سروو از اولین خط شروع به کار می نماید

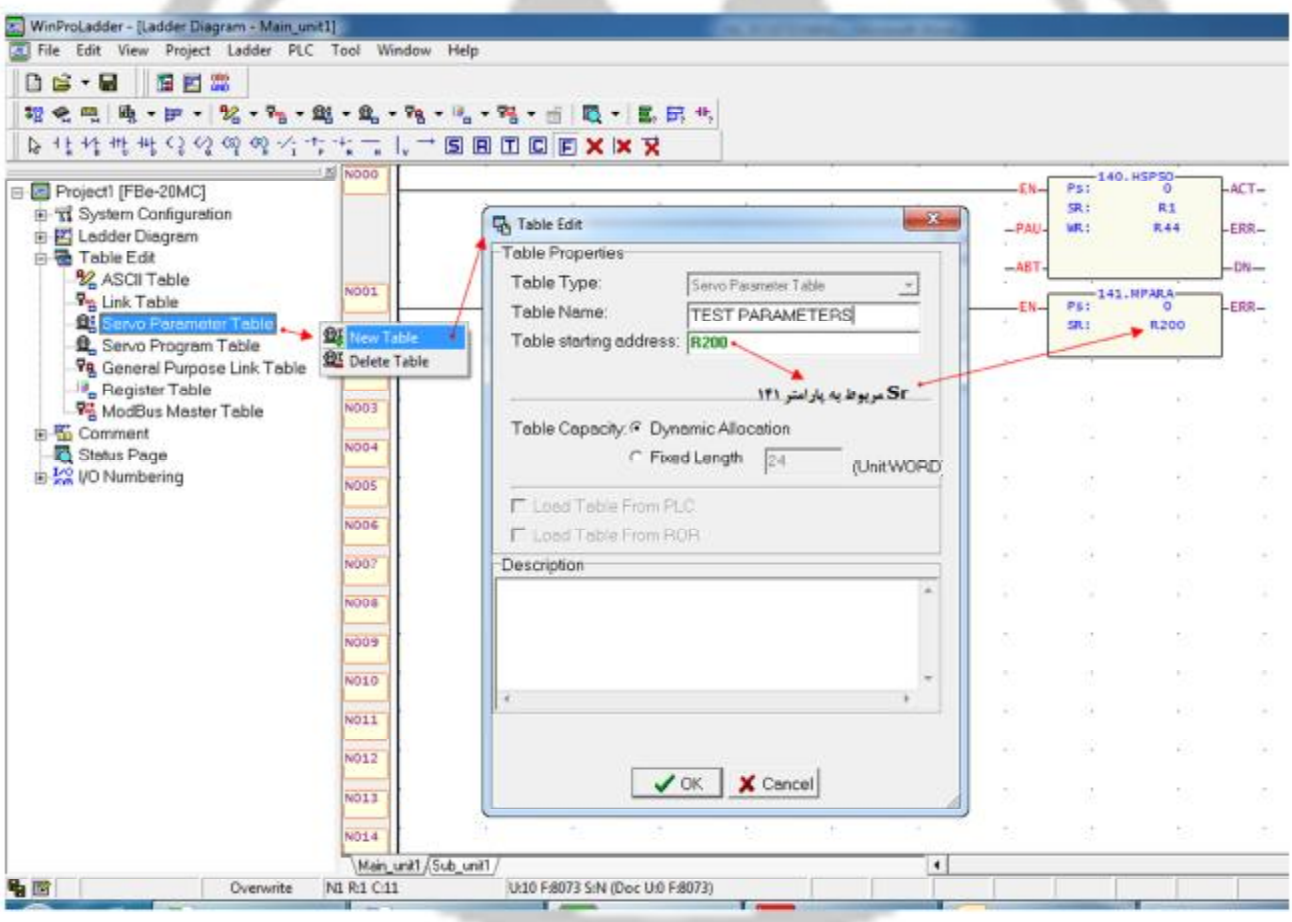
- **حالات کاری خروجی پالس**
  - **تغییرهمزمان مقدار فرکانس خروجی (تغییر فرکانس پالس) در حین اجرای FUN 140**

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.

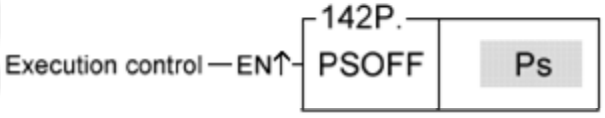
مقدار پیش تنظیم این رجیستر صفر می باشد.

FUN141 فانکشن فعال سازی جدول پارامترهای حرکت سرو :



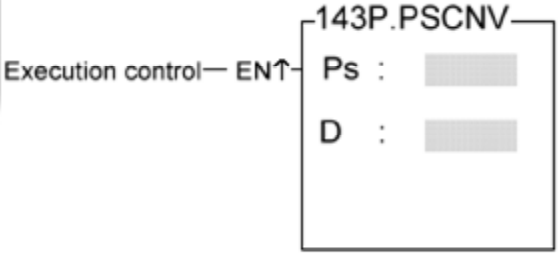


Function 142.STOP PULSE OUTPUT



این تابع فرستادن پالس به خروجی را متوقف می کند.

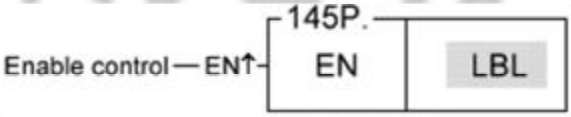
Function 143.PSCNV



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.

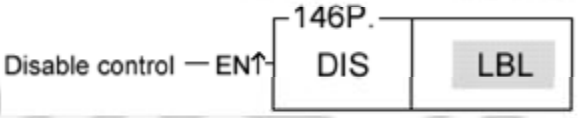
Function 145.ENABLE OF INTERRUPT



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

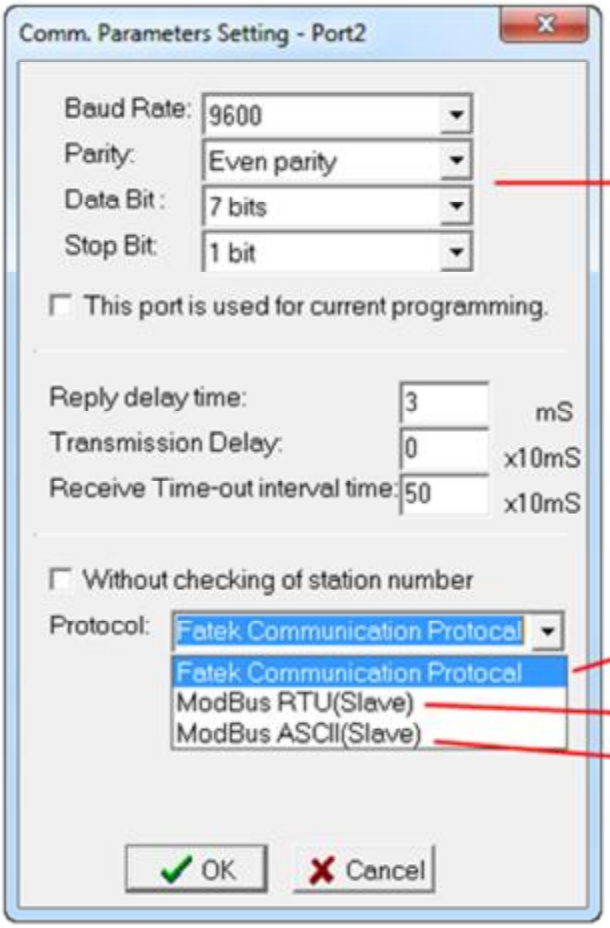
از این فانکشن ها جهت اتصال دو یا چند PLC و یا اتصال PLC به وسایل جانبی از طریق استاندارد MODBUS RTU استفاده میشود.

- **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.

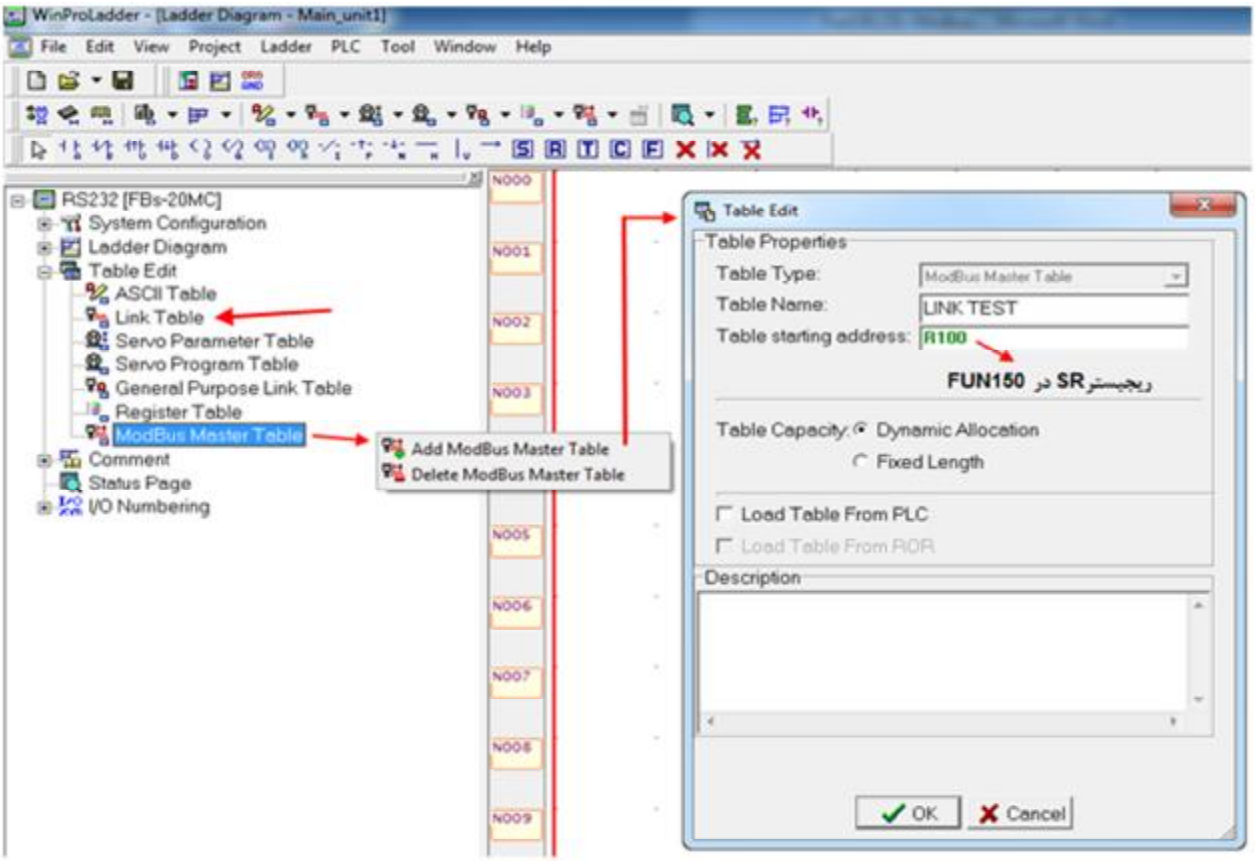
تجهیزاتی که **Modbus ASCII** دارند.

باید از آدرس مودباس ریجیسترها استفاده کرد

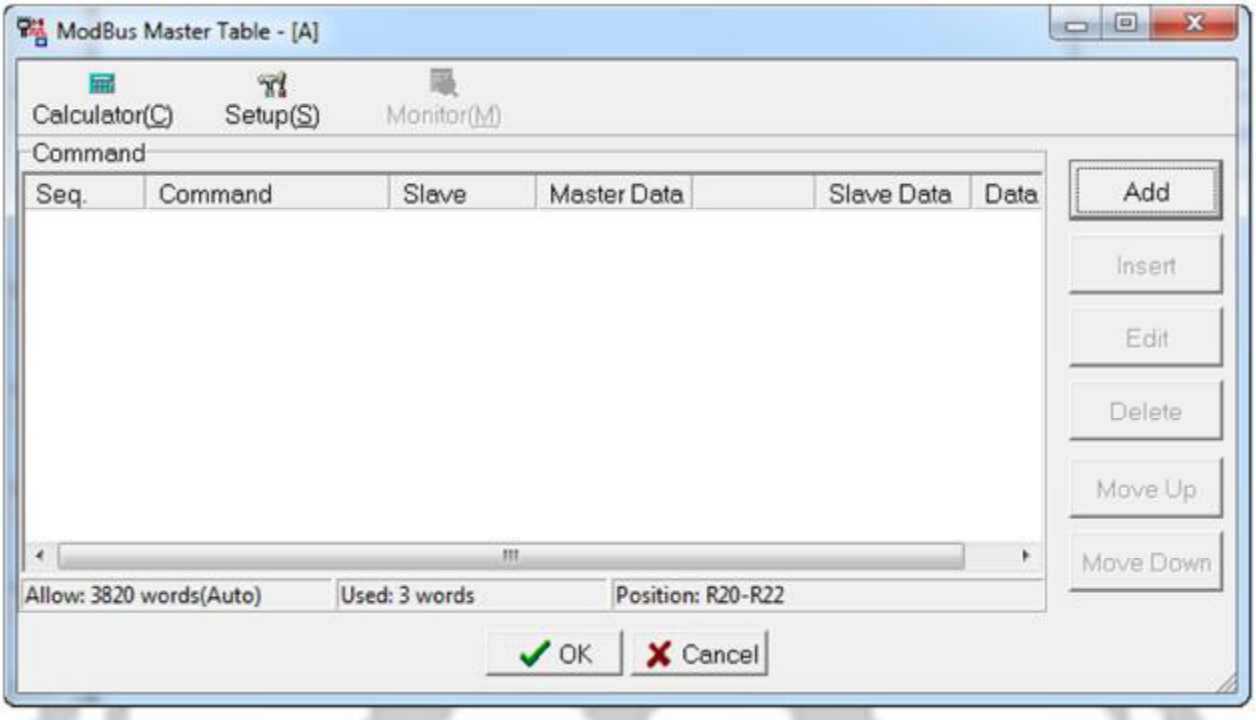
این فانکشن ها باید در برنامه پی ال سی MASTER نوشته شود .

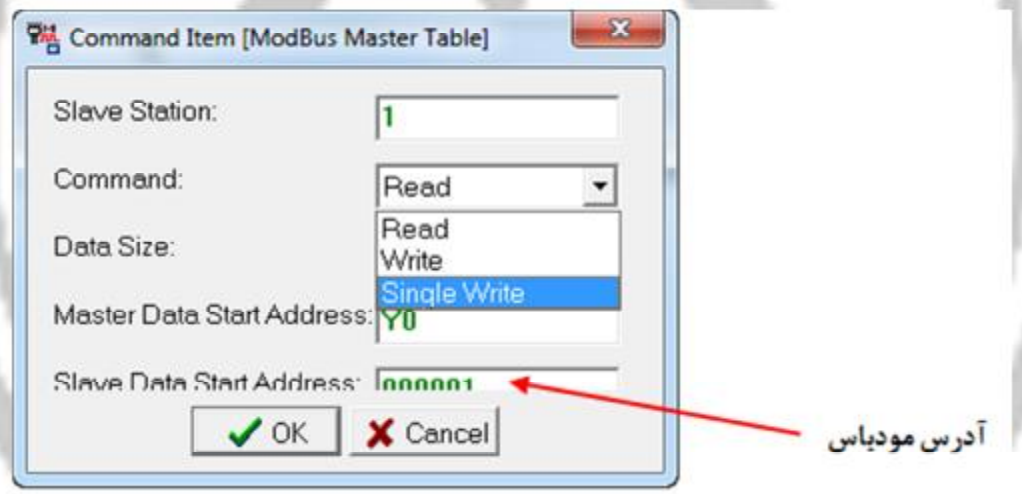
جدول مربوط به ارسال و دریافت اطلاعات برروی پی ال سی MASTER باید تنظیم شود.

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

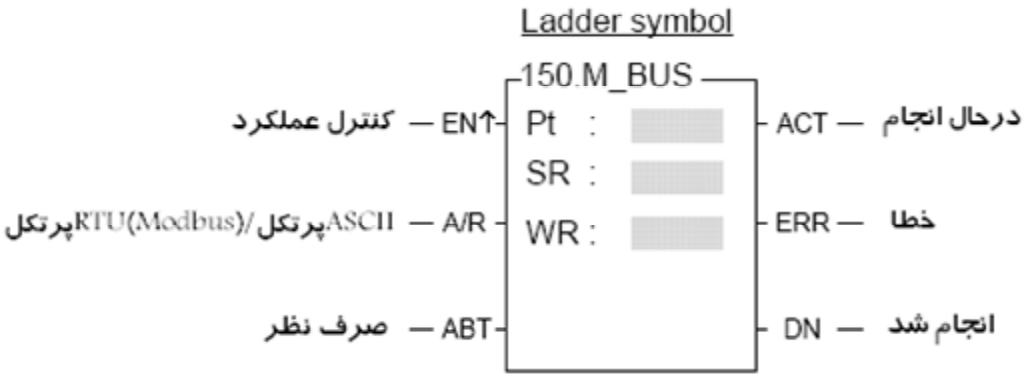
- نوع اطلاعاتی که مابین تجهیزات می تواند ردوبدل شود در جدول زیر مشاهده می نمایید

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.

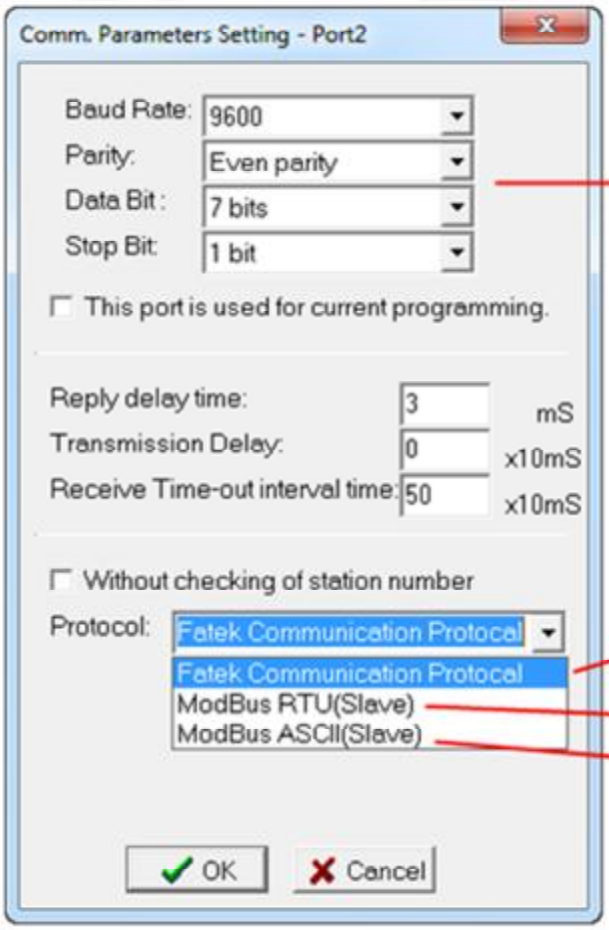
Function 151 . CPU LINK

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.

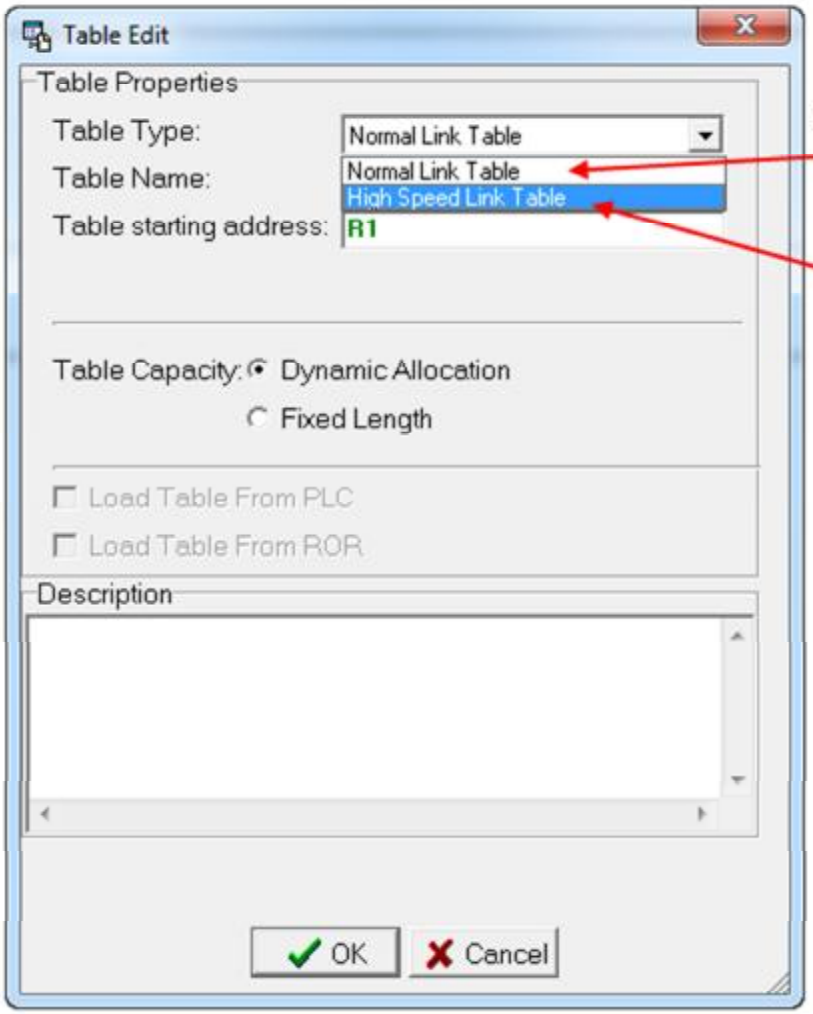
تجهیزاتی که Modbus ASCII دارند.

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.

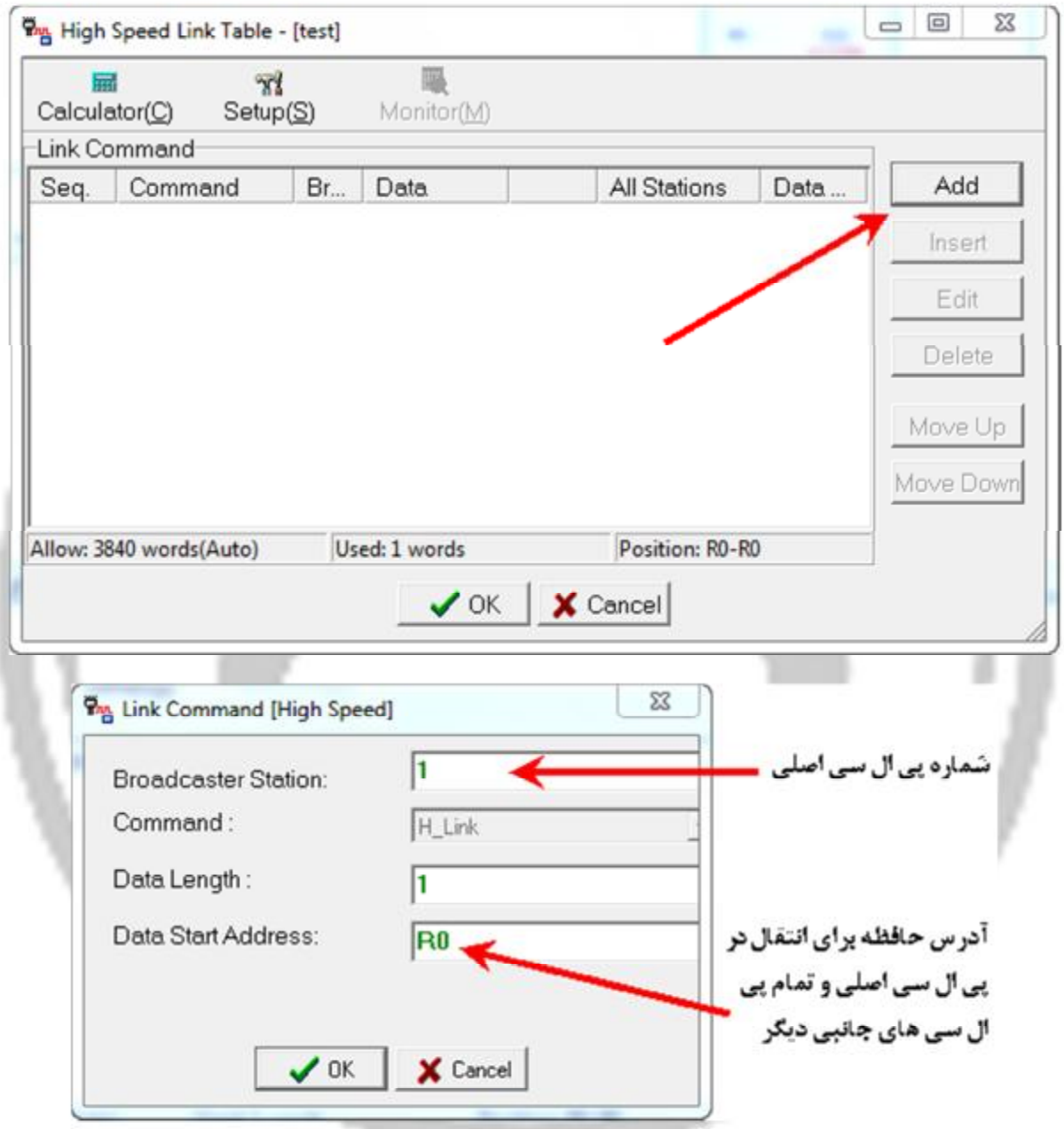


برای خواندن و نوشتن برروی SLAVE

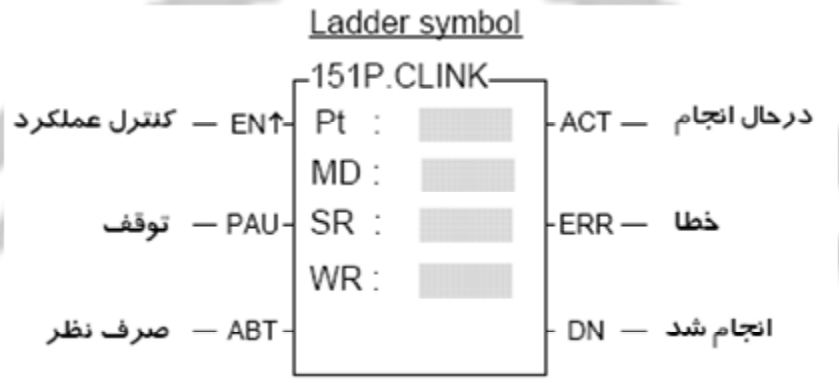
در این حالت می توان در یک لحظه برروی تمام پی ال سی های Slave دیگر نوشت

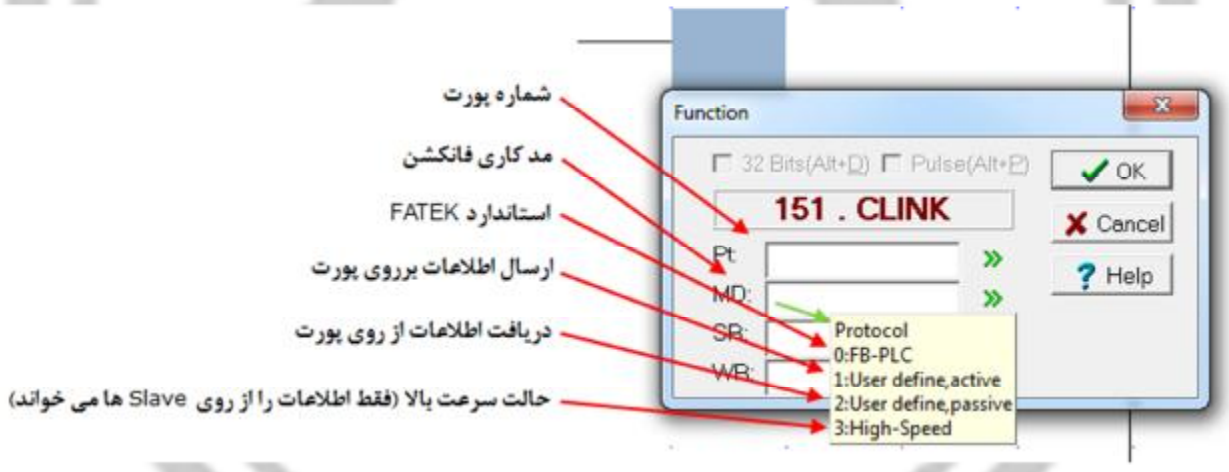
این حالت برای مواردی که می خواهیم اطلاعات پی ال سی اصلی را با سرعت در اشتراک تمام پی ال سی های جانبی بگذاریم استفاده می کنیم

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 .

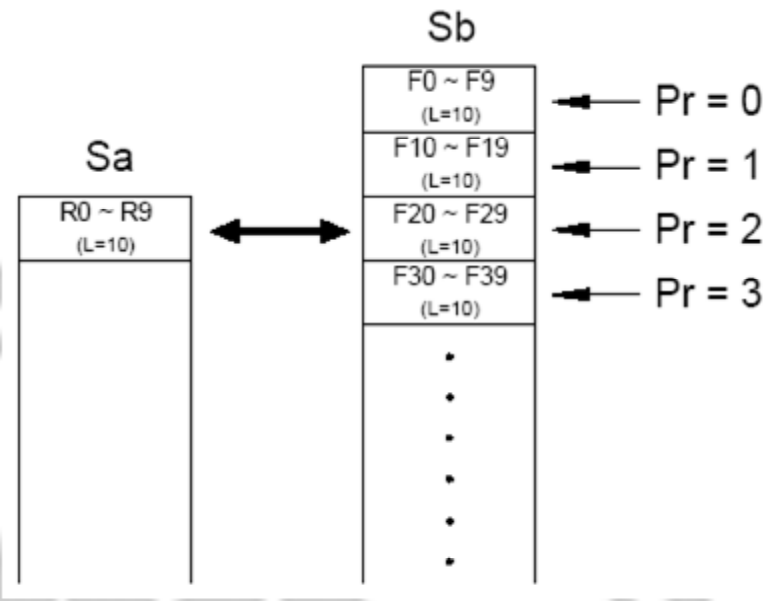
- این ارتباط میتواند تا 254 ایستگاه را پوشش دهد .
- "EN" این فانکشن ها فقط با لبه فعال می شود یعنی با هر لبه یکبار اطلاعات را برروی پورت می فرستد بنابراین ورودی EN این فانکشن را باید با کنتاکت M1920 یا M1921 یا M1922 سری کرد تا دائما اطلاعات برروی پورت فرستاده شود.

- 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

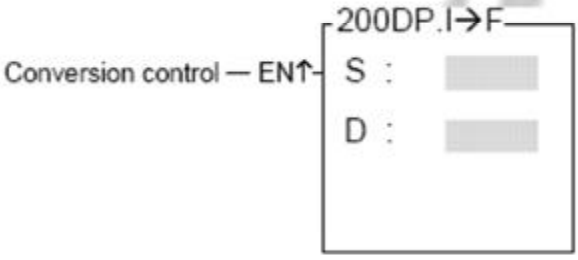
نوعی از رجیستر ها به نام File Registers در حافظه PLC وجود دارند که این تابع،تنها تابعی است که می تواند به پردازش آنها بپردازد. هرگاه "EN" از 0 به 1 تغییر کند: اگر ورودی "R/W"=1 ،محتویات رجیسترهای فایل با آدرس شروع Sb به طول L ،از جایی که Pr اشاره می کند ،به داخل رجیسترهای شروع شونده از Sa ریخته می شود. 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



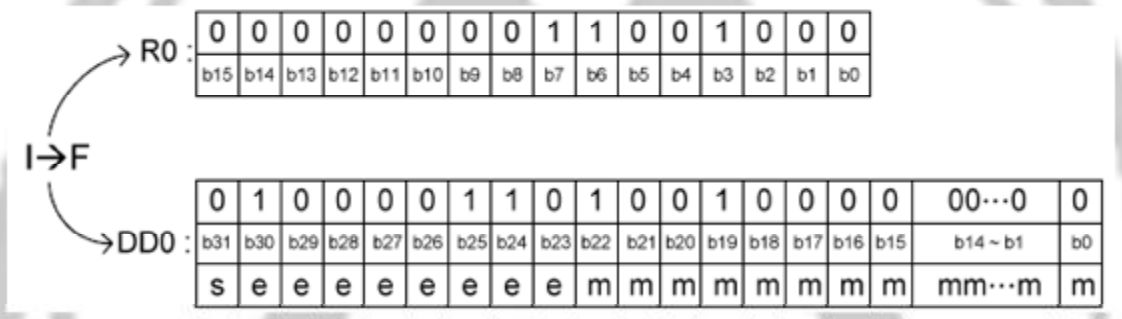
این تابع برای تبدیل فرمت اعداد صحیح به اعشاری استفاده می شود . هرگاه "EN" از 0 به 1 تغییر کند: محتویات رجیستر 16 بیتی (INTEGER) شروع شونده از S درداخل رجیستر 32 بیتی (FLOAT) شروع شونده از D ذخیره می شود.

مثال:

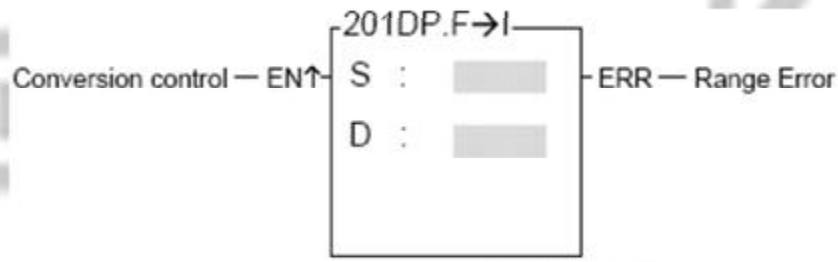
※ R0 = 200 (0000000011001000)

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

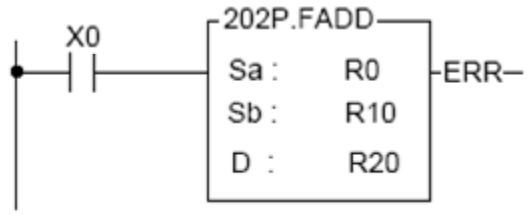
Function 202. FLOATING POINT NUMBER ADDITION

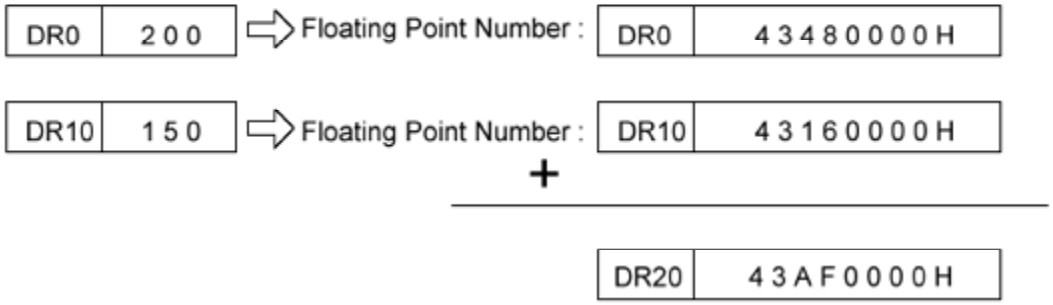
برای جمع اعداد اعشاری استفاده می شود .

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:

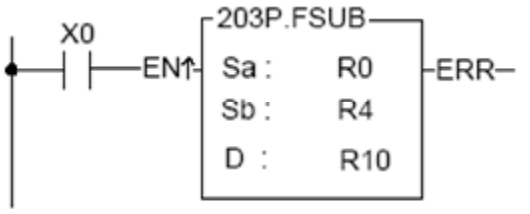


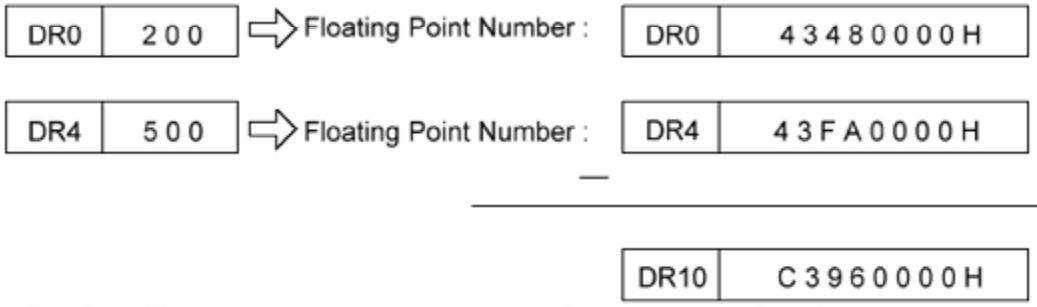


Function 203. FLOATING POINT NUMBER SUBTRACTION

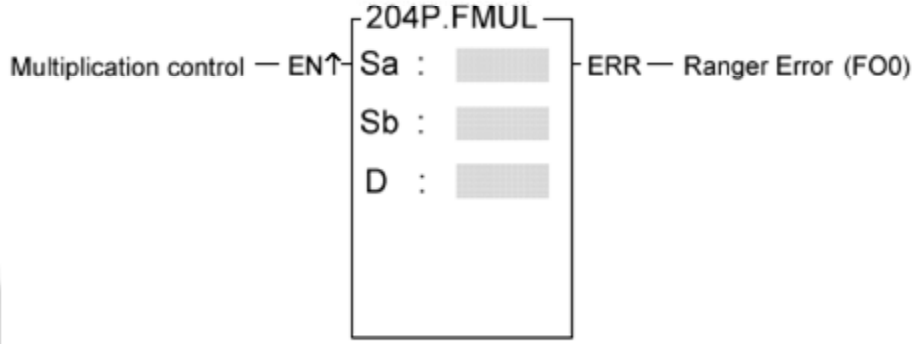
برای تفریق اعداد اعشاری استفاده می شود . هرگاه "EN" از 0 به 1 تغییر کند: مقدار FLOAT (32 بیتی) Sa منهای مقدار FLOAT ، Sb شده و نتیجه در D ریخته می شود. اگر نتیجه تفریق دو عدد فراتر از محدوده یک رجیستر 32 بیتی باشد ($\pm 3.4\times10^{38}$)، خروجی "ERR" فعال می شود.

مثال:



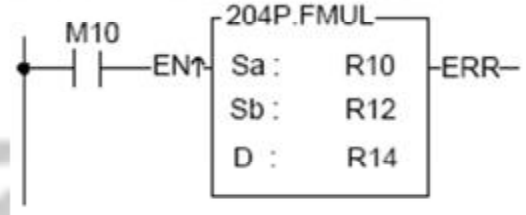


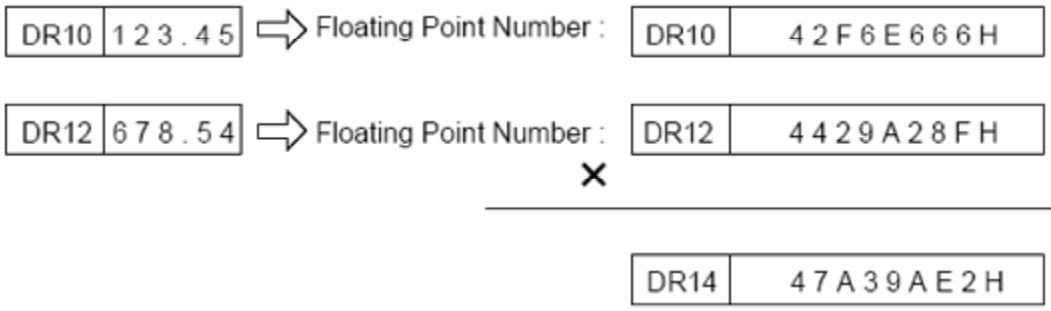
Function 204. FLOATING POINT NUMBER MULTIPLICATION



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:

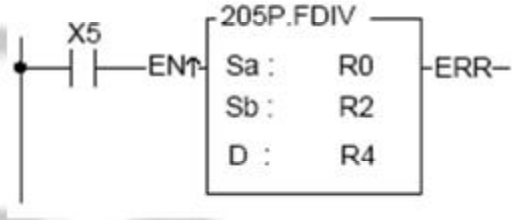


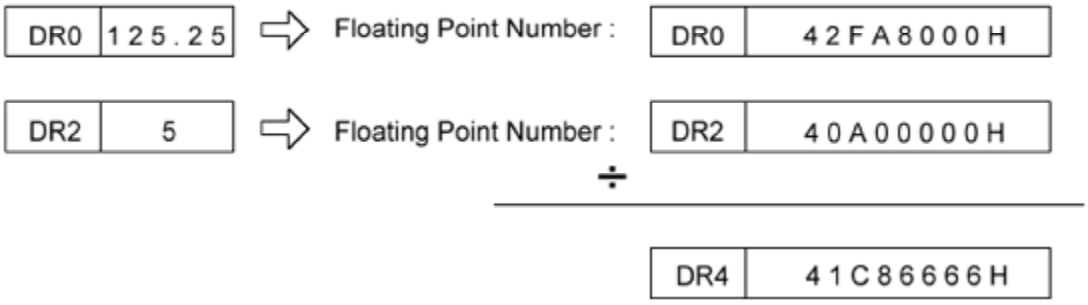


Function 205. FLOATING POINT NUMBER DIVISION

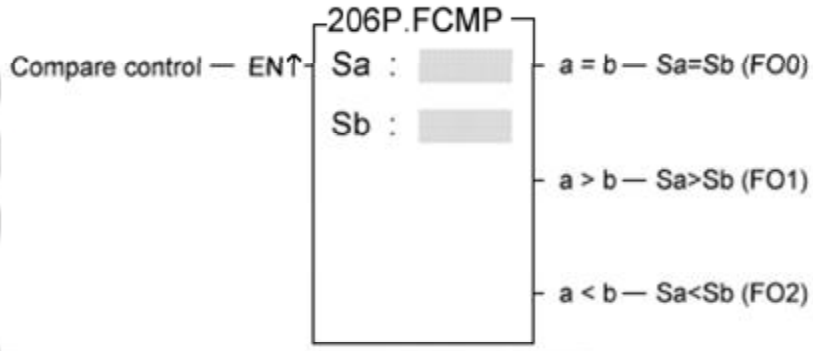
برای تقسیم اعداد اعشاری استفاده می شود . هرگاه "EN" از 0 به 1 تغییر کند: مقدار FLOAT (32 بیتی) Sa را بر مقدار FLOAT ، Sb تقسیم کرده و نتیجه در D ریخته می شود. اگر نتیجه تقسیم دو عدد فراتر از محدوده یک رجیستر 32 بیتی باشد ($\pm 3.4\times10^{38}$)، خروجی "ERR" فعال می شود.

مثال:



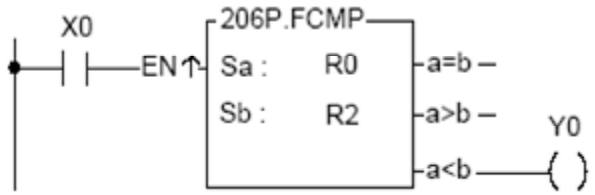


Function 206. FLOATING POINT NUMBER COMPARE



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.

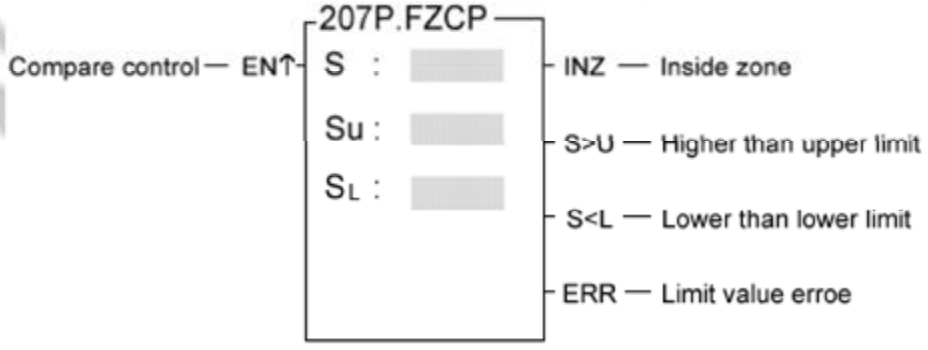
مثال:



DR0	200.1	⇨ Floating Point Number :	DR0	4348199AH

DR2	200.2	⇨ Floating Point Number :	DR2	43483333H

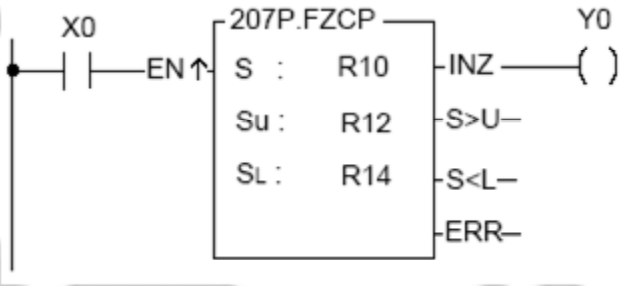
Function 207. FLOATING POINT NUMBER ZONE COMPARE



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:



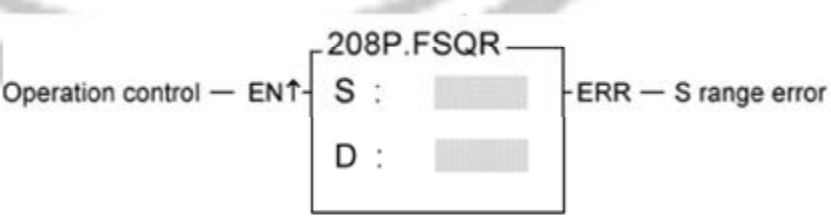
S	DR10	2000.2	⇨ Floating Point Number :	DR10	44FA0666H	
Su	DR12	3000.3	⇨ Floating Point Number :	DR12	453B84CDH	(Upper limit value)
S_L	DR14	1000.1	⇨ Floating Point Number :	DR14	447A0666H	(Lower limit value)

Before-execution

X0 ⎍ → FLOATING ZONE COMPARE → Y0 = 1

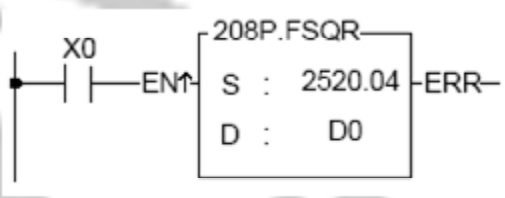
Results of execution

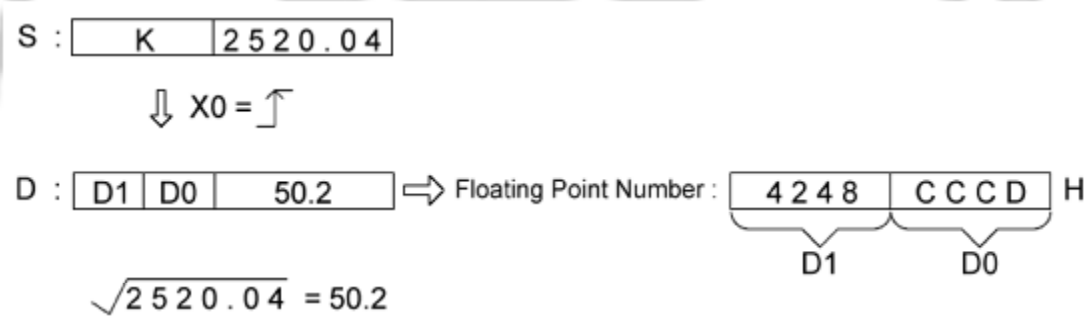
Function 208. FLOATING POINT NUMBER SQUARE ROOT



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:





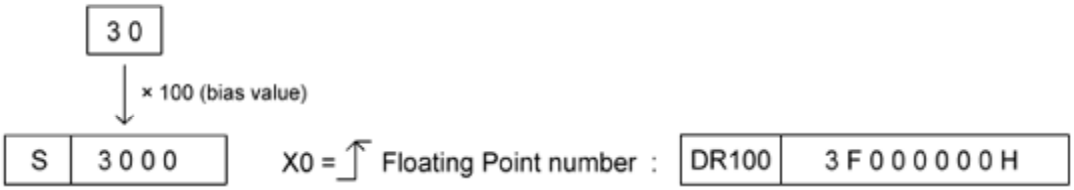
Function 209.SIN INSTRUCTION

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.

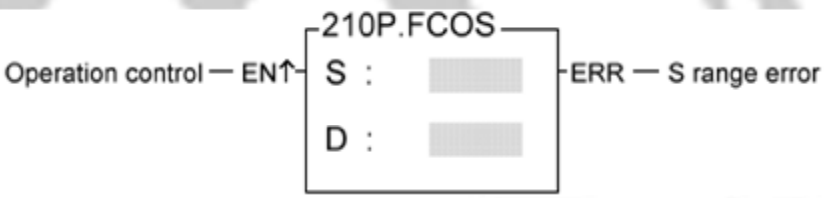
مثال:



SIN(30) = 0.5

Function 210.COS INSTRUCTION

تابع مثلثاتی کسینوس



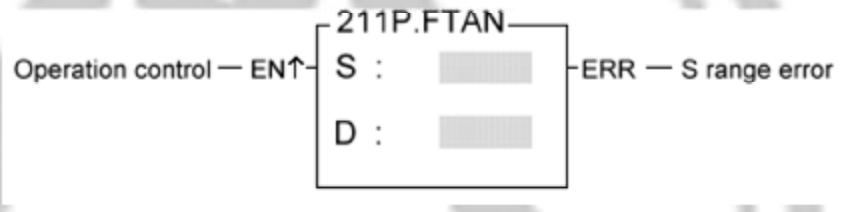
هرگاه "EN" از 0 به 1 تغییر کند: از مقدار موجود در رجیستر S ، کسینوس گرفته و نتیجه را به صورت 32 بیتی در D و D+1 می ریزد. محدوده S در واحد 0.01 درجه ، 18000+~18000- می باشد که اگر فراتر از این محدوده داده شود ،خروجی "ERR" فعال می شود.

مثال:

COS(60) = 0.5

Function 211.TAN INSTRUCTION

تابع مثلثاتی تانژانت

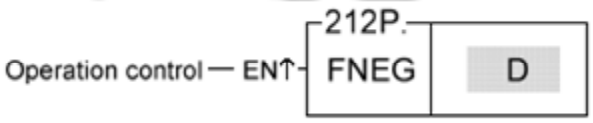


هرگاه "EN" از 0 به 1 تغییر کند: از مقدار موجود در رجیستر S ، تانژانت گرفته و نتیجه را به صورت 32 بیتی در D و D+1 می ریزد. محدوده S در واحد 0.01 درجه ، 18000+~18000- می باشد که اگر فراتر از این محدوده داده شود ، خروجی "ERR" فعال می شود.

مثال:

TAN(45) = 1

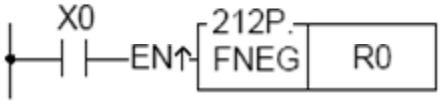
Function 212.CHANGE SIGN OF FLOAT



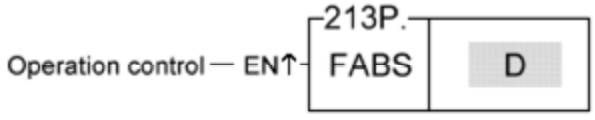
برای تغییر علامت اعداد اعشاری استفاده می شود .

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: