FATEK PLC توابع

فهرست

تایمر----------6

کانتر----------7

SET----------8

RESET----------8

0&1.MC & MCE----------8

2&3. SKIP START & SKIP END----------9

4.DIFFERENTIAL UP----------9

5.DIFFERENTIAL DOWN----------9

6.BIT SHIFT----------10

7.UP/DOWN COUNTER----------11

8.MOVE----------13

9. MOVE INVERSE----------14

10.TOGGLE SWITCH----------14

11.ADDITION----------15

12.SUBTRACTION----------15

13.MULTIPLICATION----------16

14.Division----------16

15.INCREMENT----------17

16 .DECREMENT----------18

17.COMPARE----------19

18.LOGICAL AND----------20

19.LOGICAL OR----------21

20.BIN TO BCD CONVERSION----------22

21.BCD TO BIN CONVERSION----------23

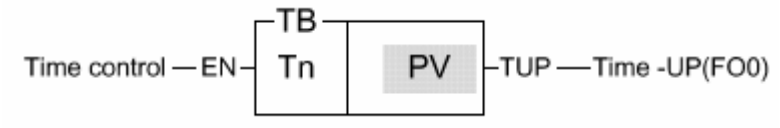
24----------22.BREAK
24----------23.48-BIT DIVISON
25----------24.SUM
26----------25.MEAN
27----------26.SQUARE ROOT
28----------27.NEGATION
28----------28.ABSOLUTE
29----------29.SIGN EXTENSION
29----------31.CRC16
30----------32.ADCNV
31----------35.EXCLUSIVE OR (XOR)
32----------36.EXCLUSIVE NOR (XNOR)
33----------37.ZONE COMPARE
34----------40.BIT READ
35----------41.BIR WRITE
36----------42.BIT MOVE
37----------43.NIBBLE MOVE
38----------44.BYTE MOVE
39----------45.EXCHANGE
40----------46.BYTE SWAP
41----------47.NIBBLE UNITE
42----------48.NIBBLE DISTRIBUTE
43----------49.BITE UNITE
44----------50.BYTE DISTRIBUTE
45----------51.SHIFT LEFT
46----------52.SHIFT RIGHT
47----------53.ROTATE LEFT
48----------54.ROTATE RIGHT
49----------55.BINARY TO GRAY

49----------56. GRAY TO BINARY
50----------57.DECODE
51----------58.ENCODE
52----------59.7-SEGMENT CONVERSION
54----------60.ASCII CONVERSION
55----------61.H:M:S to SECONDS CONVERSION
56----------62.SECOND to H:M:S
57----------63.ASCII to HEX
57----------64.HEX toASCII
58----------PROGRAM END
58----------65.LABLE
59----------66.JUMP
59----------67.CALL
60----------68.RTS
60----------69.RTI
61----------70.FOR
62----------71.LOOP END
62----------74.IMMIDIATE I/O
63----------76.DECIMAL-KEY INPUT
64----------77.HEX-KEY INPUT
65----------78.DIGITAL SWITCH INPUT
66----------79.7-SEG OUTPUT WITH LATCH
67----------80.MULTIPLEX INPUT
68----------81.PULSE OUTPUT
70----------82.PULSE WIDTH MODULATION
71----------83.SPEED DETECTION
73----------84.16/7-SEG DISPLAY
75----------87.88.89.CUMULATIVE TIMER
77----------90.WATCH DOG TIMER

77----------91.RESET WDT
78----------92.HSCTR
79----------93.HSCTW
80----------94.ASCII WRITE
81----------95.RAMP
83----------توابع جدولی
84----------100.REGISTER TO TABLE MOVE
86----------101.TABLE TO REGISTER MOVE
87----------102.TABLE TO TABLE MOVE
88----------103.BLOCK TABLE MOVE
89----------104.BLOCK TABLE SWAP
90----------105.REGISTER TO TABLE SEARCH
92----------106.TABLE TO TABLE COMPARE
93----------107.TABLE FILL
94----------108.TABLE SHIFT
96----------109.TABLE ROTATE
97----------110.QUENE
99----------111.STACK
100----------112.DRUM
102----------113.DATA SORTING
103----------114.ZONE WRITE
104----------توابع ماتریسی
105----------120.MATRIX AND
106----------121.MATRIX OR
107----------122.MATRIX XOR
108----------123.MATRIX XNOR
109----------124.MATRIX INVERSE
110----------125.MATRIX COMPARE
112----------126.MATRIX BIT READ

114----------127.MATRIX BIT WRITE
115----------128.MATRIX BIT SHIFT
117----------129.MATRIX BIT ROTATE
119----------130.MATRIX BIT STATUS COUNT
120----------139.HSPWM
121----------142.STOP PULSE OUTPUT
121----------143.PSCNV
122----------145.ENABLE OF INTERRUPT
123----------146.DISABLE OF INTERRUPT
124----------147.MULTI-AXIS HIGH SPEED PULSE OUTPUT
133----------160.READ/WRITE FILE REGISTER
134----------200.CONVERSION of INT to FLOAT
135----------201. FLOAT to INTEGER
136----------202. FLOATING POINT NUMBER ADDITION
137----------203. FLOATING POINT NUMBER SUBTRACTION
138----------204. FLOATING POINT NUMBER MULTIPLICATION
139----------205. FLOATING POINT NUMBER DIVISION
140----------206. FLOATING POINT NUMBER COMPARE
141----------207. FLOATING POINT NUMBER ZONE COMPARE
142---------- 208. FLOATING POINT NUMBER SQUARE ROOT
143----------209.SIN INSTRUCTION
144----------210.COS INSTRUCTION
145----------211.TAN INSTRUCTION
146----------212.CHANGE SIGN OF FLOAT
147----------213.FLOAT ABSOLUTE

TIMER



Tn: Timer Number

PV: Preset value of the timer

TB:Time Base

TUP :Time Up

CV:Current Value

تعداد تایمرها مجموعاً 256 تا است.(T0~T255)

سه زمان پایه (TB) وجود دارد با مقادیر0.01s,0.1s,1s

تخصیص تایمرهای مختلف به TBها به صورت زیر می باشد:

T0~T49 : 0.01s

T50~T199 : 0.1s

T200~T255 : 1s

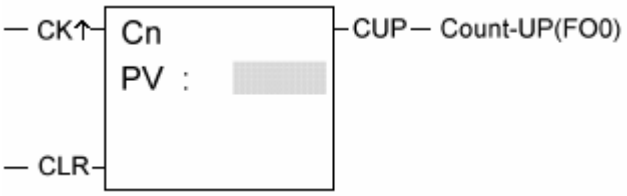
محاسبه زمان تایمر بدین صورت می باشد:

زمان تایمر =PV×TB

هرگاه پایه" EN"فعال شود(1 شود) تایمر فعال می شود،CVاز مقدار 0شروع 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.

مقدار PV بعد از اجرای برنامه دیگر قابل تغییر نمی باشد.

COUNTER



CN:Counter Number

CLR:Clear Control

CK:Clock

CUP:Count UP

در مجموع 200 کانتر 16 بیتی وجود دارد (C0~C199) و

56 کانتر 32 بیتی (C200~C255)

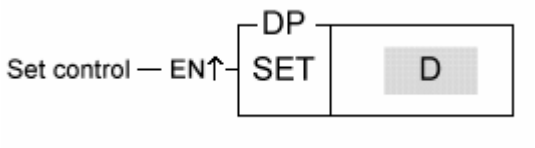
کانتر های C200~C239و C0~C130 کانترهای محافظتی هستند که به هنگام نقص توان یا توقف PLC مقدار CV را حفظ می کنند تا زمانی کهPLC دوباره راه اندازی شود.

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 استفاده کنید.

با فعال کردن پایه CLR ،پایه های دیگر (CUPو 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.

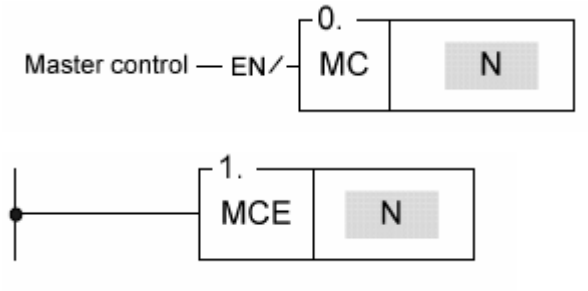
RESET

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

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 یکسان می باشد (N)

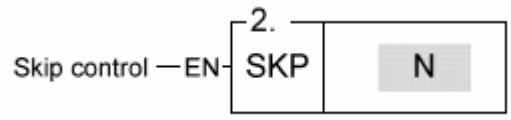
باید توجه کرد که تابع MCE بعد از MC قرار بگیرد.

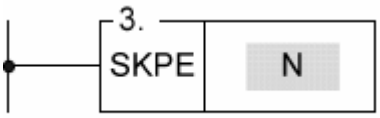
هرگاه "EN" فعال شود،تابع MC اجرا نمی شود.

تمام مقادیر خروجی ها و تایمر ها ،با فعال شدن MC، 0 می شوند و تابع ها دیگر اجرا نخواهند شد.

MCE نیازی به ورودی ندارد چون وابسته به MC است.

2&3. SKIP START & SKIP END



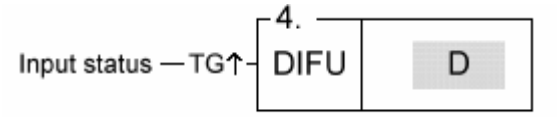


128 حلقه ی SKP وجود دارد(N=0~127)

هر تابع SKP با یک تابع SKPE مرتبط است و تابع SKPE باید بعد از SKP قرار بگیرد .

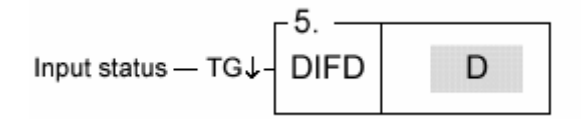
وقتی EN ،0شود ، SKP اجرا نخواهد شد.

4.DIFFERENTIAL UP



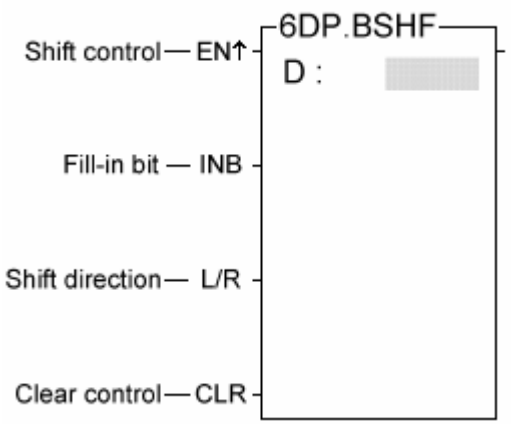
این تابع باعث می شود که هرگاه ورودی TG فعال شد ،خروجی به اندازه Scan Time فعال باشد.

5.DIFFERENTIAL DOWN



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

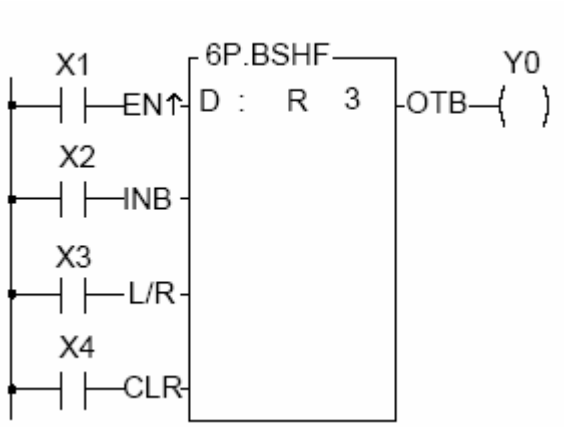
6.BIT SHIFT

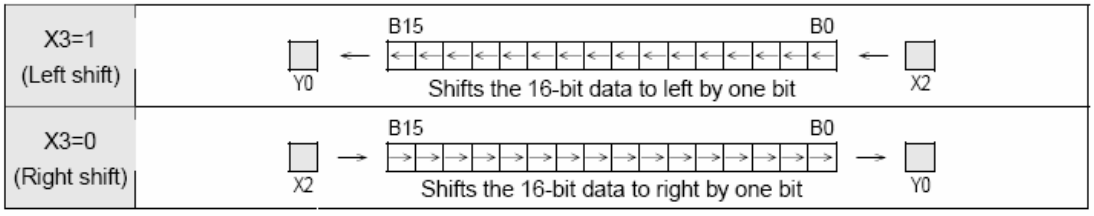


با این تابع می توان رجیستر مورد نظر را 1 بیت به چپ یا راست ،شیفت داد.بدین منظور CLR باید 0 باشد.برای شیفت به راست "L/R"=0 و برای شیفت به چپ "L/R"=1 شود.

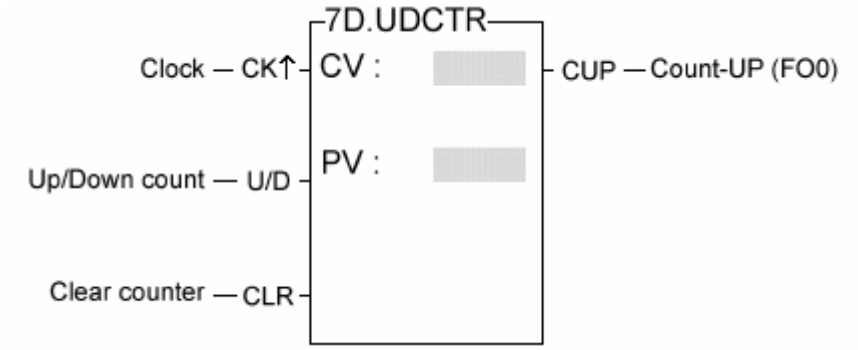
با "INB" می توان تعیین نمود که بیت ایجاد شده پس از اعمال شیفت،0 یا 1 بشود.

مثال:





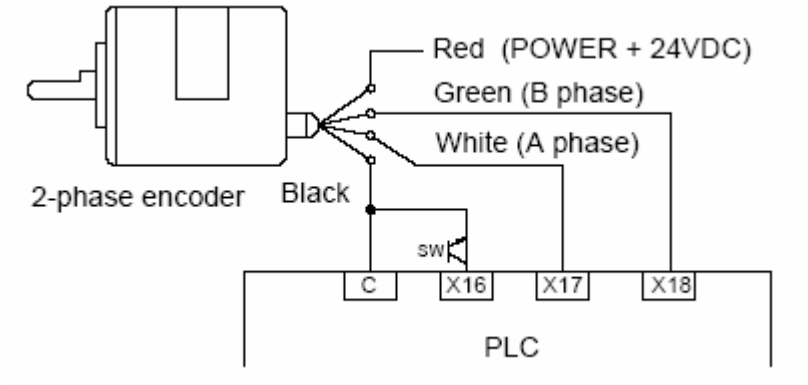
7.UP/DOWN COUNTER

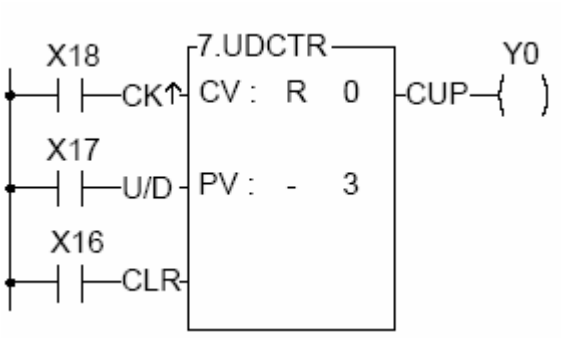


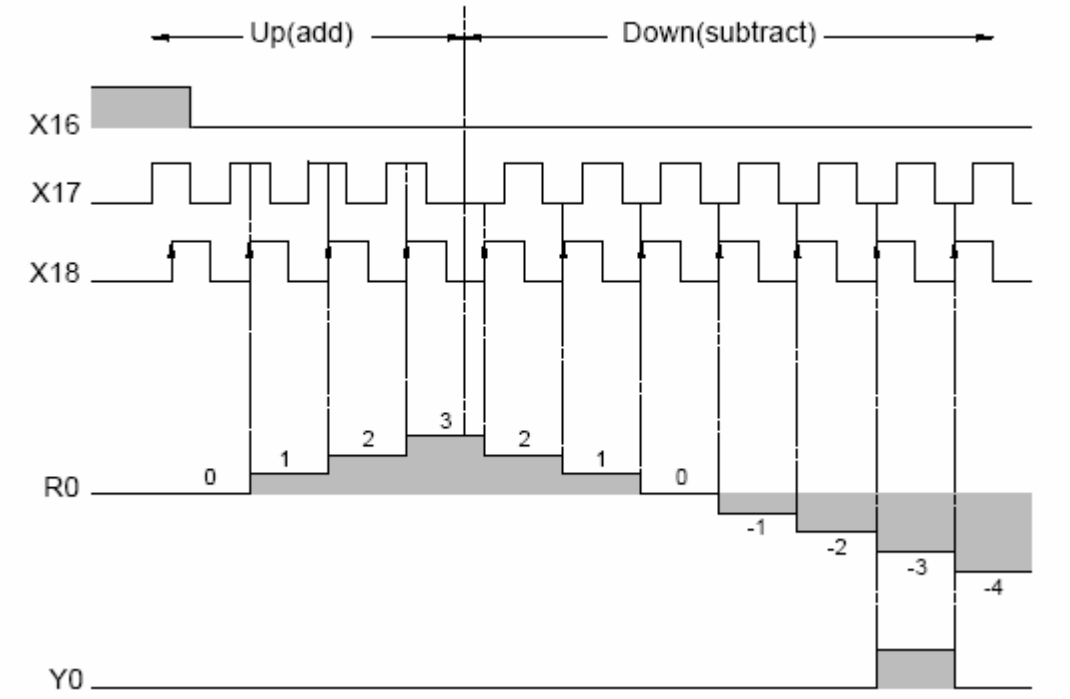
این کانتر می تواند به صورت دو فاز اعمال شود.

برای شروع به شمارشCLR باید 0باشد،اگر "U/D"=1 ،با فعال بودن CK ،CV افزایش می یابد و اگر "U/D"=0 باشد،کاهش می یابد.

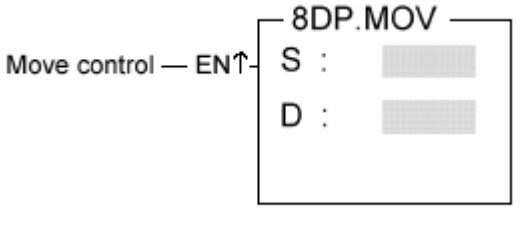
مثال:




X18
7.UDCTR
Y0
CK↑
CV : R 0
CUP
X17
U/D
PV : - 3
X16
CLR

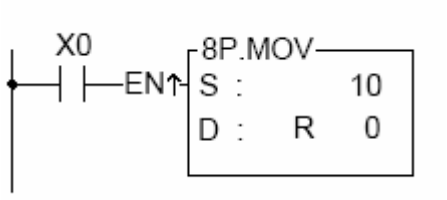

Up(add)
Down(subtract)
X16
X17
X18
R0
Y0
0
1
2
3
2
1
0
-1
-2
-3
-4

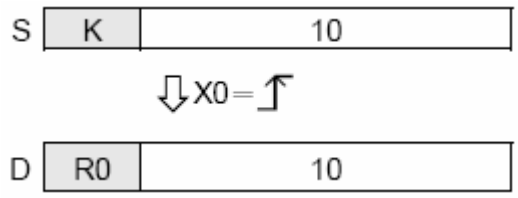
8.MOVE



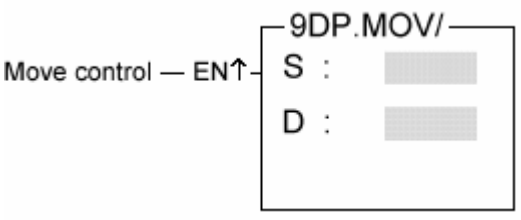
هرگاه "EN" فعال شود،اطلاعات موجود در رجیستر S به D انتقال داده می شود.

مثال:



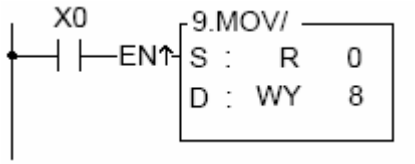


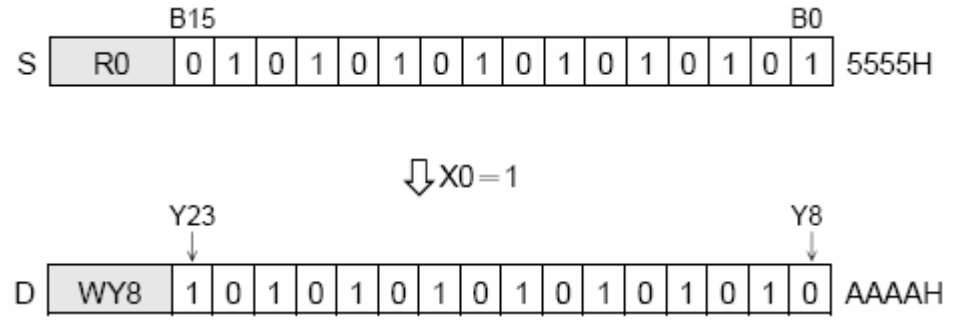
9. MOVE INVERSE



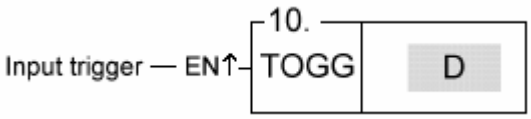
با فعال شدن "EN"، اطلاعات موجود در Sعکس شده (0ها به 1 و 1ها به 0 تبدیل می شوند) وبه D منتقل می شود.

مثال:



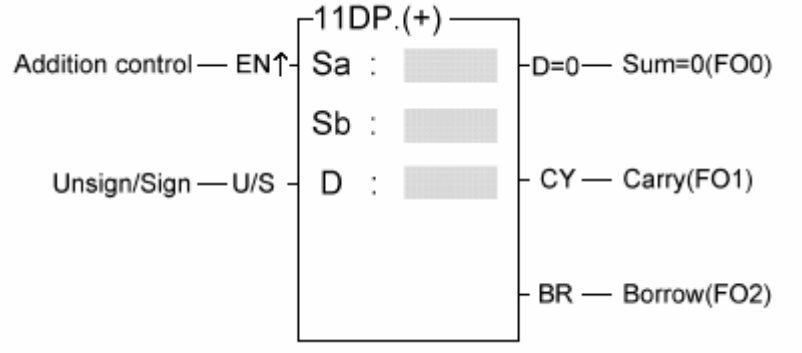


10.TOGGLE SWITCH



با فعال شدن "EN"، 0های موجود در رجیستر D، 1 شده و 1 ها 0 می شوند.

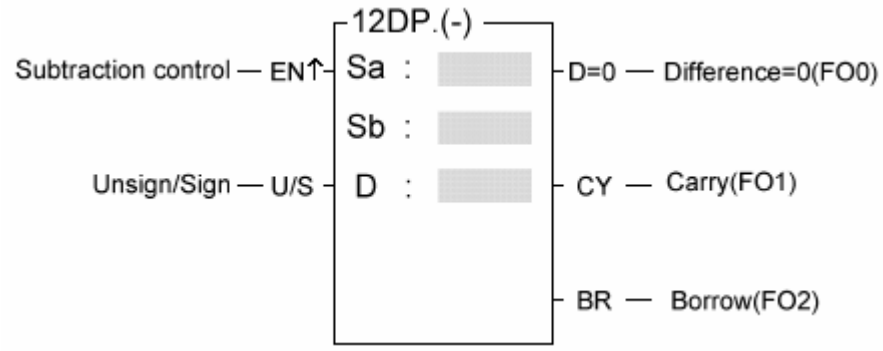
11.ADDITION



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

Sa+Sb=D

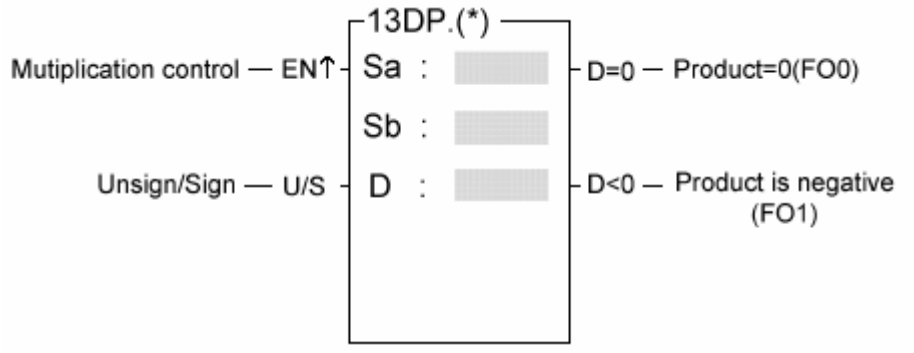
12.SUBTRACTION



با فعال شدن "EN" ،داده های موجود در Sa، منهای Sbشده و نتیجه در D ریخته می شود.

Sa-Sb=D

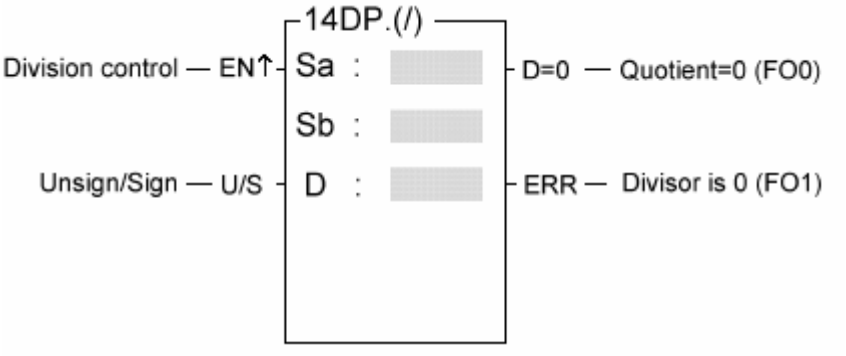
13.MULTIPLICATION



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

Sa×Sb=D

14.Division



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

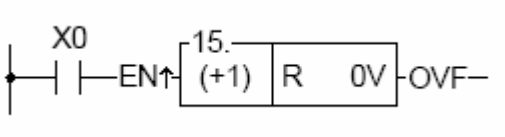
هرگاهSb صفر باشد،تابع اجرا نمی شود و Errorمی دهد.

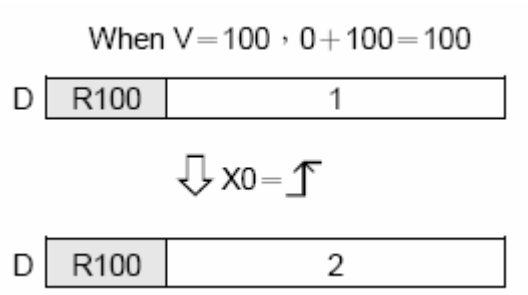
15.INCREMENT

هرگاه "EN" از 0 به 1 تغییر کند،به مقدار رجیستر D ،1+ اضافه می شود .

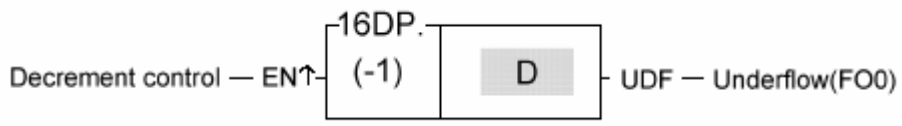
اگر این افزایش ،باعث از رنج (range) خارج شدن مقدار D شود،"OVF" فعال می شود و مقدار D منفی می شود.

مثال:



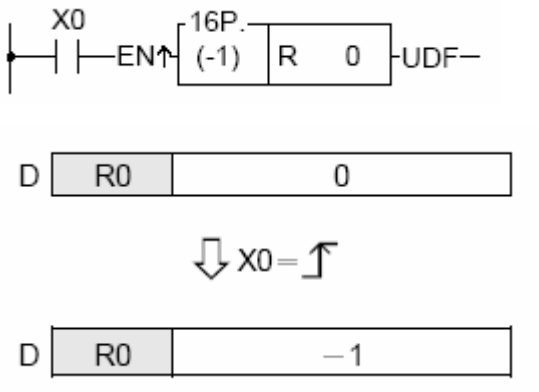


16.DECREMENT

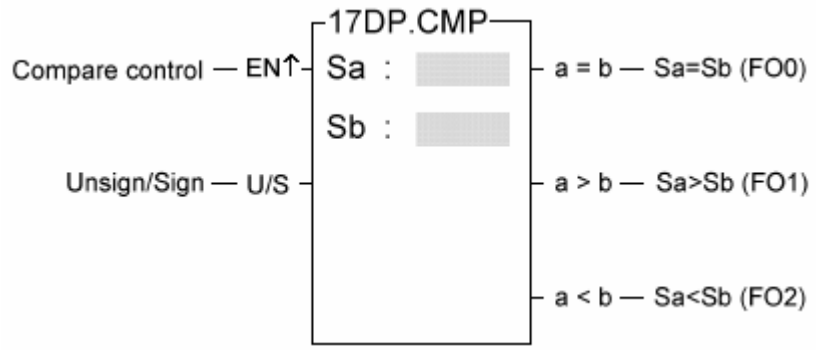


هرگاه "EN" از 0 به 1 تغییر کند،از مقدار رجیستر D ،1 کم می شود.(1- می شود)
اگر این کاهش باعث از رنج (range) خارج شدن مقدار D شود،"UDF" فعال می شود و مقدار D مثبت می شود.

مثال:



17.COMPARE



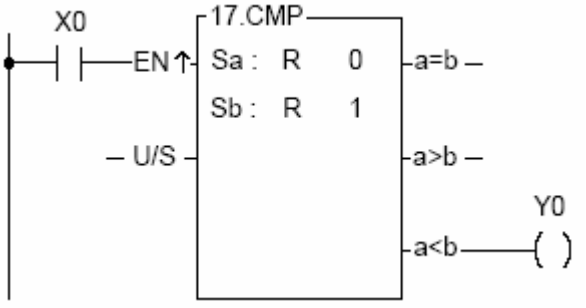
هرگاه "EN" از 0 به 1 تغییر کند،مقدار Sa و Sb را مقایسه می کند.

اگر با هم برابر باشند ،خروجی "a=b" فعال شده

اگر Sa>Sb باشد، خروجی "a>b" فعال شده

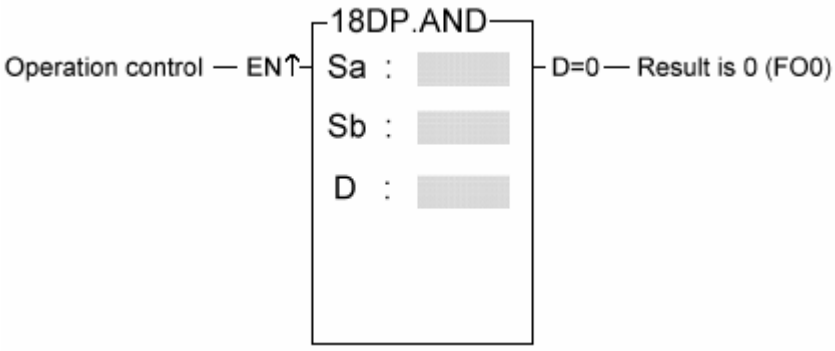
و اگر Sa<Sb باشد،خروجی "a<b" فعال می شود.

مثال:



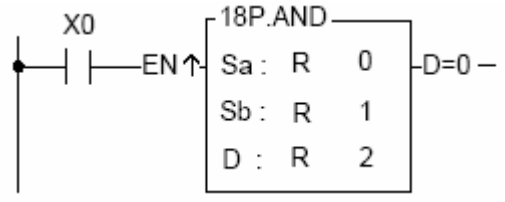
اگر می خواهید نتایجی مانند $<>,\leqq,\geqq$ به دست آورید،ابتدا نتایج <،> و = را به رله فرستاده سپس آنها را از رله گرفته و با یکدیگر ترکیب کنید.

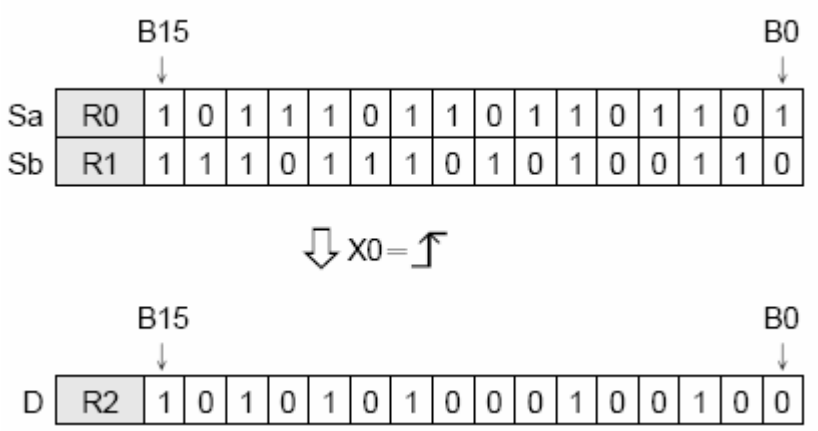
18.LOGICAL AND



هرگاه "EN" از 0 به 1 تغییر کند،بیت های موجود درSa وSb را با هم AND کرده و نتیجه را در D می ریزد.

مثال:

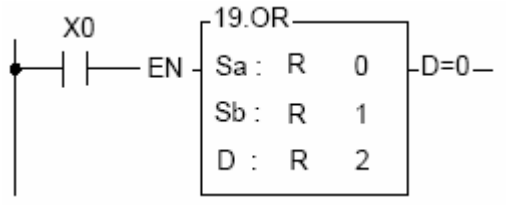


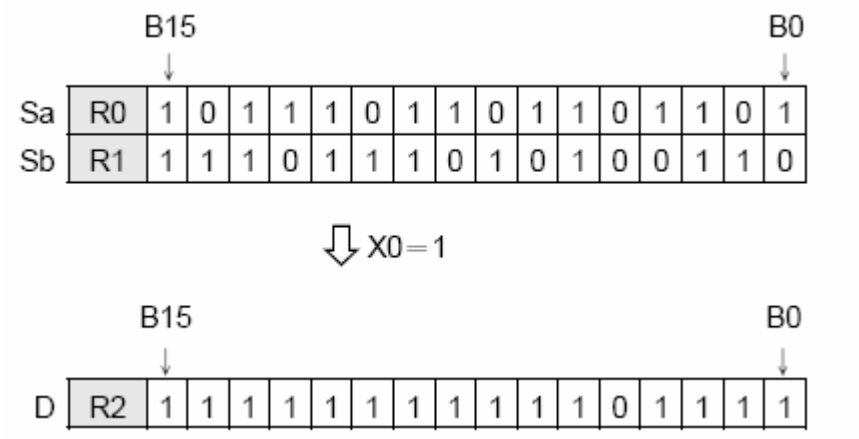


19.LOGICAL OR

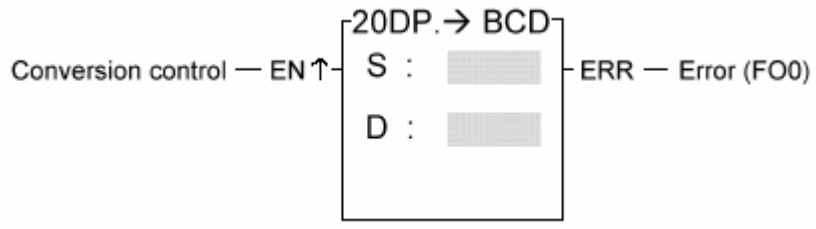
هرگاه "EN" از 0 به 1 تغییر کند،بیت های موجود درSa وSb را با هم OR کرده و نتیجه را در D می ریزد.

مثال:





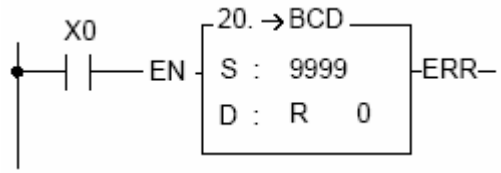
20.BIN TO BCD CONVERSION

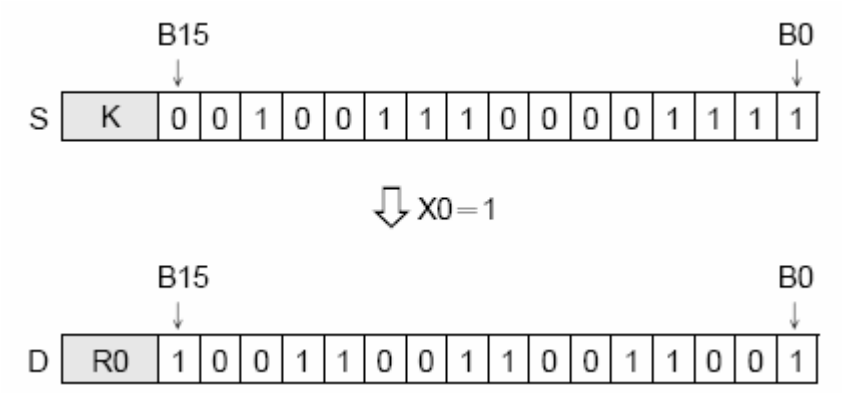


هرگاه "EN" از 0 به 1 تغییر کند،داده های موجود در S را که به صورت باینری است،به صورت BCD درآورده و در D می ریزد.

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

مثال:

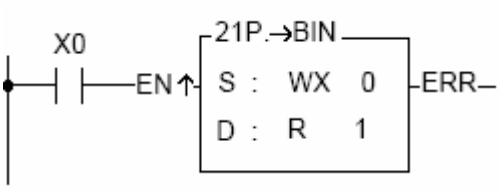


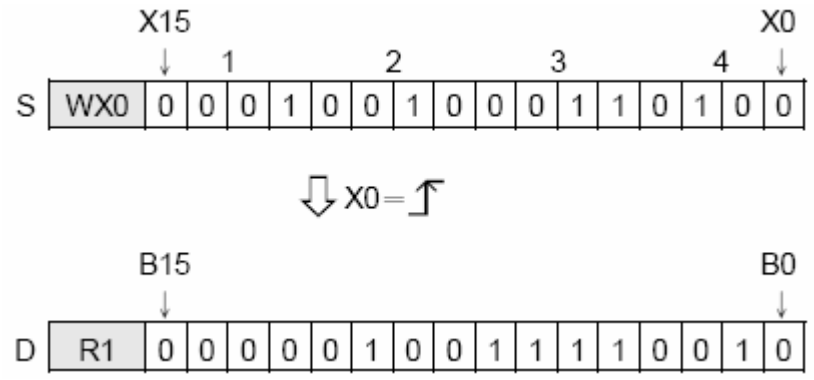


21.BCD TO BIN CONVERSION

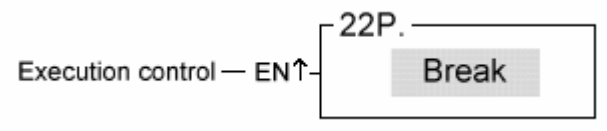
برعکس BIN TO BCD عمل می کند.

مثال:



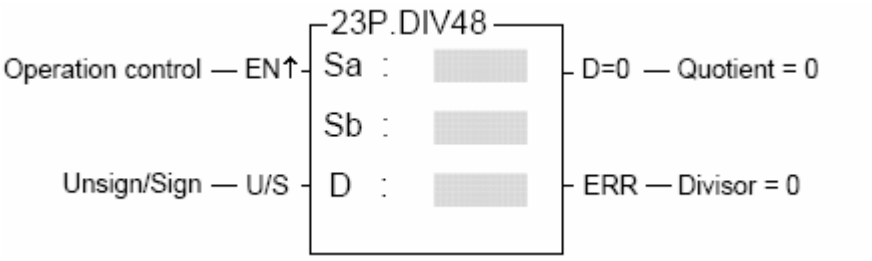


22.BREAK



این تابع باید به همراه حلقه ی FOR-NEXT 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.

23.48-BIT DIVISON

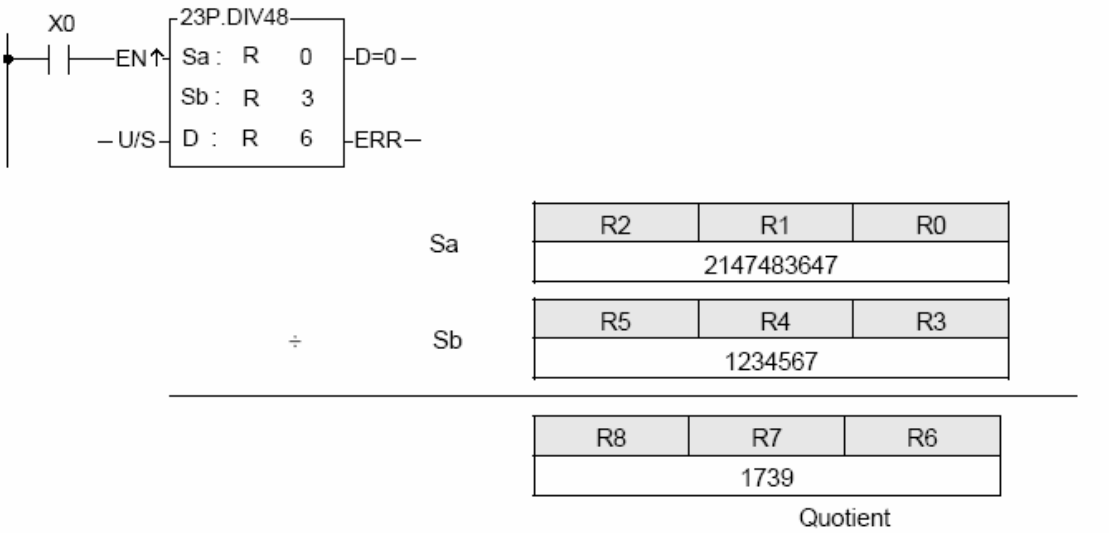


هرگاه "EN" از 0 به 1 تغییر کند،مقدار 48 بیتی موجود در Sa را تقسیم بر مقدار 48 بیتی موجود درSb کرده و خارج قسمت را در D می ریزد.

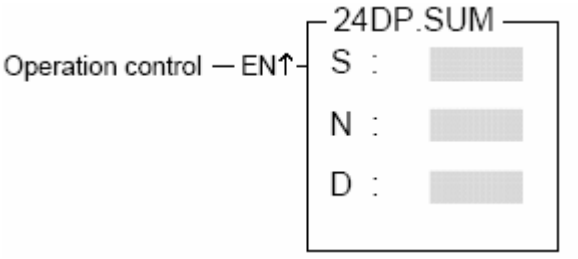
اگر نتیجه صفر باشد"D=0" فعال می شود.

اگر Sb صفر باشد"ERR" فعال می شود.

مثال:



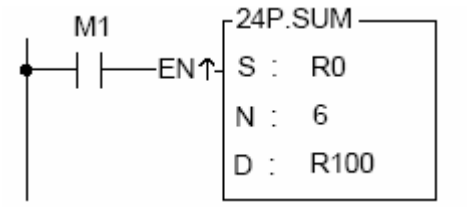
24.SUM

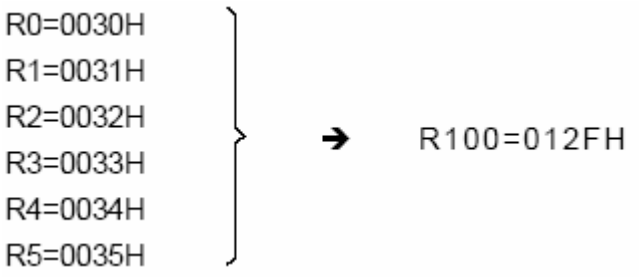


با این تابع می توان مجموع چند رجیستر متوالی را به یک رجیستر دیگر منتقل کرد.

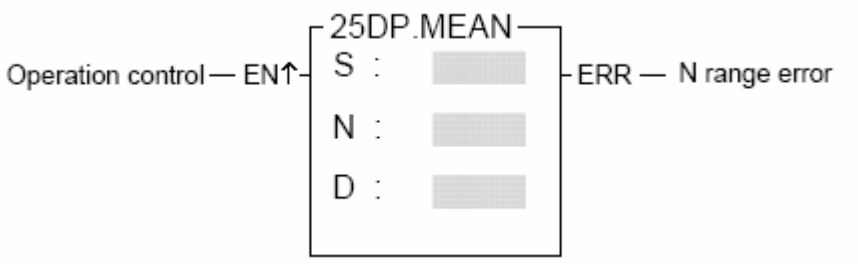
در S اولین رجیستر قرار داده می شود،در N تعداد رجیسترهای متوالی تعیین می شود و مجموع این N تعداد رجیستر در D ریخته می شود.

مثال:





25.MEAN



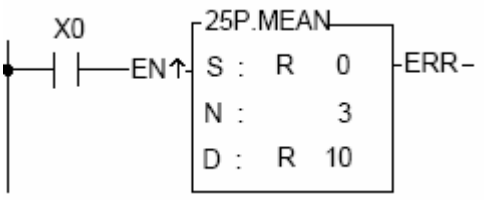
این تابع برای میانگین گرفتن از مقادیر چند رجیستر متوالی کاربرد دارد.

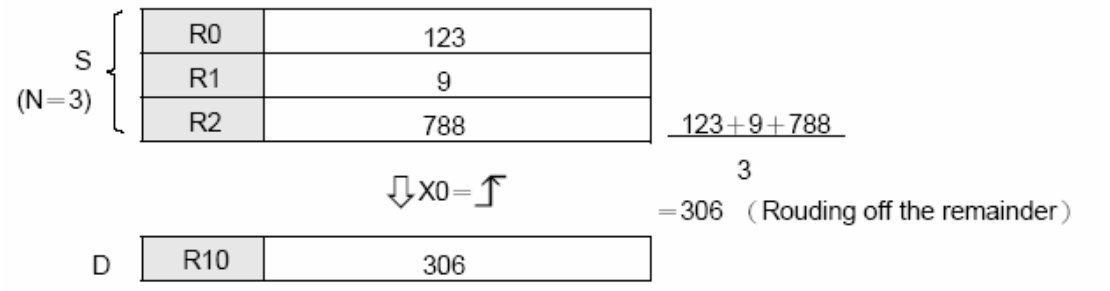
رجیسترشروع در S قرار می گیرد،تعداد رجیسترها در N.

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.

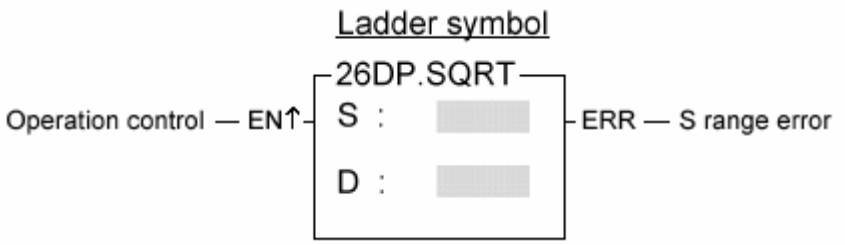
اگر مقدار N بین 2 تا 256 نباشد ،"ERR" فعال شده و تابع اجرا نمی شود.

مثال:





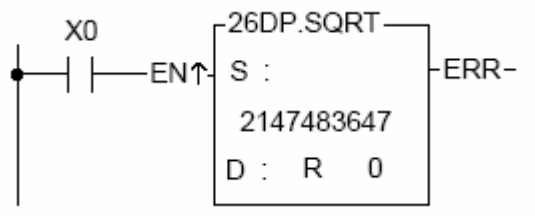
26.SQUARE ROOT

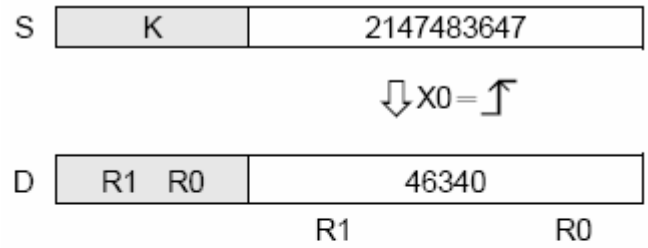


این تابع،جذر عدد موجود در S را گرفته و نتیجه را بدون در نظر گرفتن اعشار آن در D می ریزد.

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

مثال:



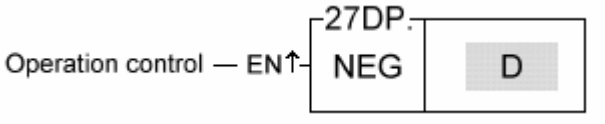


$$\sqrt{2147483647} = 46340.95$$

↑

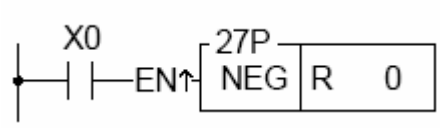
rounding off

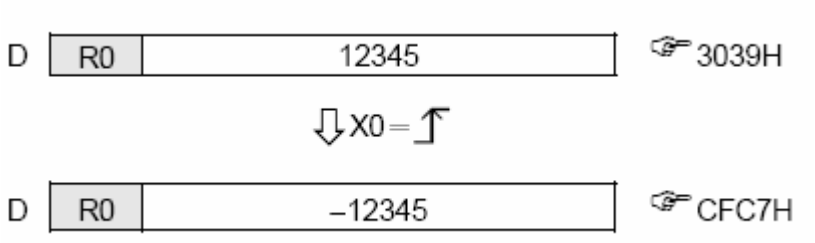
27.NEGATION



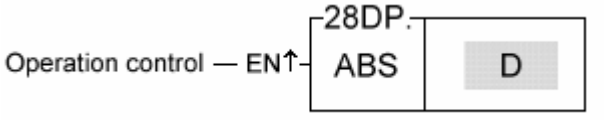
هرگاه "EN" از 0 به 1 تغییر کند،مقدار D منفی می شود و دوباره در همان رجیستر ریخته می شود.

مثال:



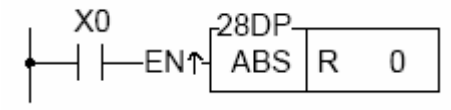


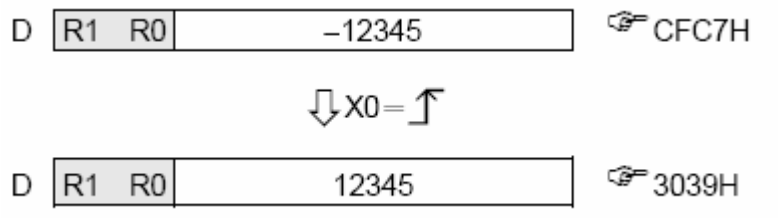
28.ABSOLUTE



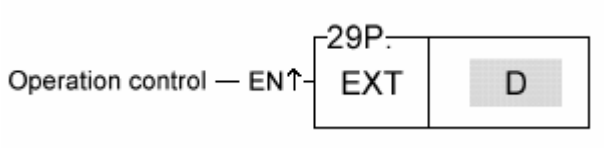
این تابع،قدر مطلق مقدار موجود د ر D را دوباره در D می ریزد.

مثال:



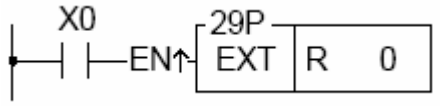


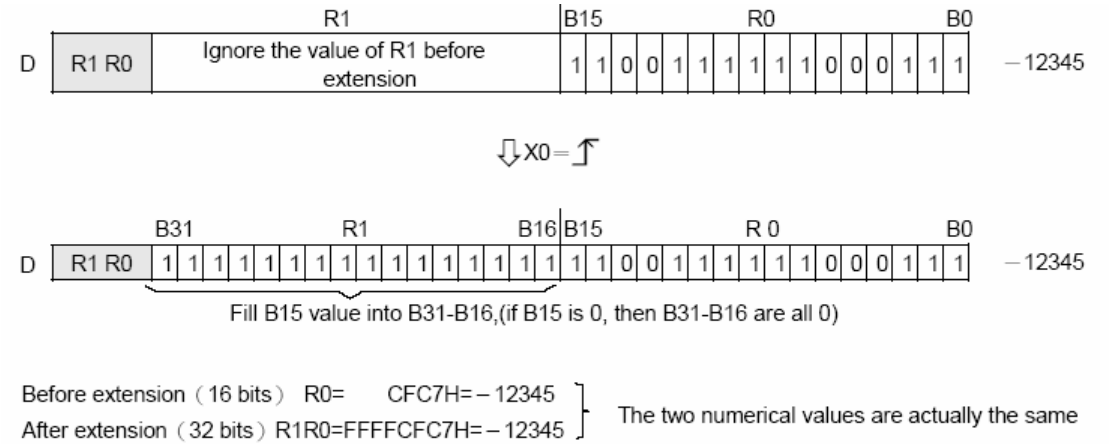
29.SIGN EXTENSION



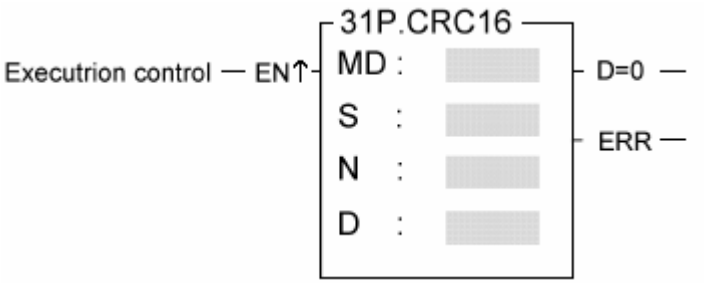
د ر D یک مقدار 16 بیتی قرار دارد ،با فعال شدن "EN" همین مقدار 32 بیتی می شود.(برای این کار از رجیستر D+1 استفاده می شود)

مثال:





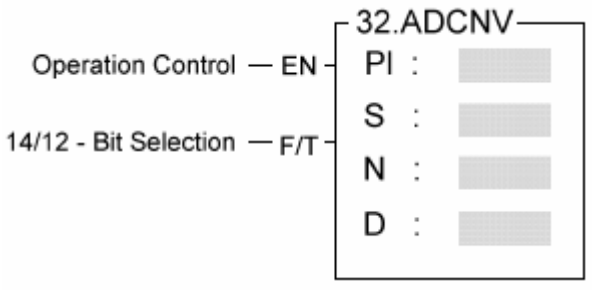
31.CRC16



این تابع، هرگاه "EN" از 0 به 1 تغییر کند،شروع به محاسبه می کند.آدرس شروع محاسبات در S قرار می گیرد ،تعداد رجیسترهایی که محاسبه بر روی آنها صورت می گیرد با N بیان می شود و نتیجه در رجیستر معادل D و D+1 ریخته می شود.

کاربرد این تابع بر روی اطلاعات پیام های دریافتی است که نتیجه چک کردن آنها باید 0 بشود، در غیر این صورت "ERR" فعال خواهد شد.

32.ADCNV



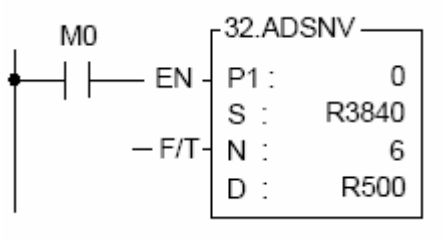
این تابع،ورودی آنالوگ بر حسب میلی آمپر را در رنج (0~20mA) می خواند و ان را به عددی تبدیل می کند که برای کار PLC مناسب تر است.

وقتی ورودی "F/T"، 0 باشد ،عدد تبدیل شده در رنج (12-bit) 0~4095 قرار می گیرد.

و اگر ورودی "F/T"، 1 باشد، عدد تبدیل شده در رنج(14-bit) 0~16383 قرار می گیرد.

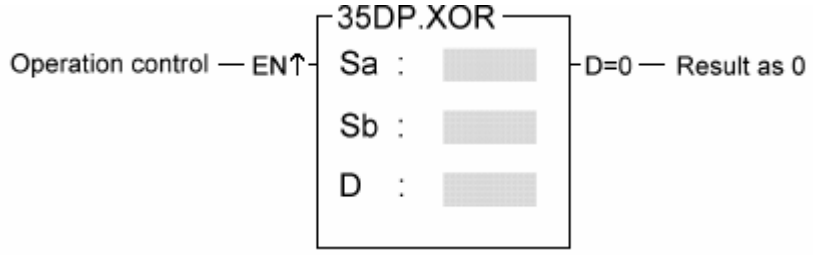
وقتی "EN"=1 است، تبدیل را از S به طول N شروع کرده و نتایج را در D ذخیره می کند.

مثال:



S			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)

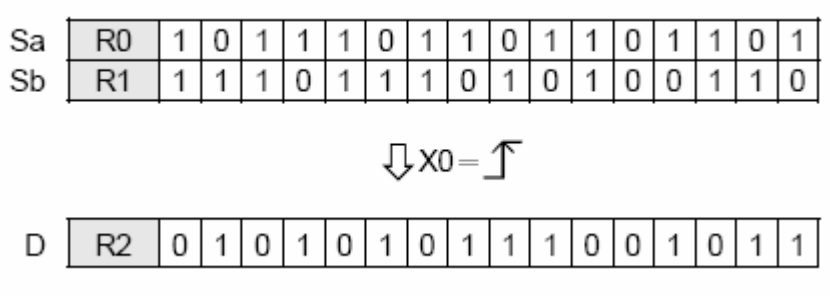
35.EXCLUSIVE OR (XOR)



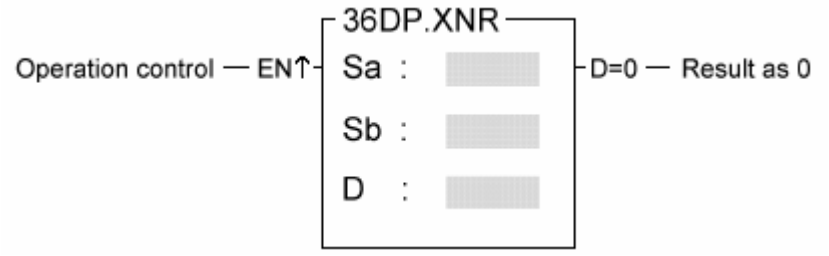
هرگاه "EN" از 0 به 1 تغییر کند،بیت های موجود در Sa و Sb را با هم XOR کرده و نتیجه را در D می ریزد.

عملکرد به این صورت است که هرگاه بیت های متناظر Sa و Sb همانند بودند ،بیت متناظر در D ،0 می شود و هرگاه یکی 1 و آن یکی 0 بود ،نتیجه در D ،1 می شود.

هرگاه تمام بیت ها یD ،صفر شود ، "D=0" فعال می شود.



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 نا همانند 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

37.ZONE COMPARE

هرگاه "EN" از 0 به 1 تغییر کند،مقدار S با مقدار S_U و S_L مقایسه می شود.

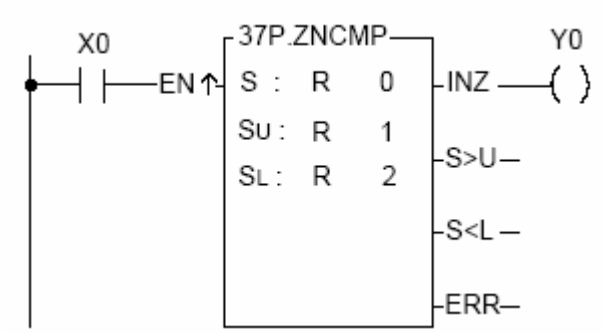
اگر بین این دو باشد،"INZ" فعال می شود

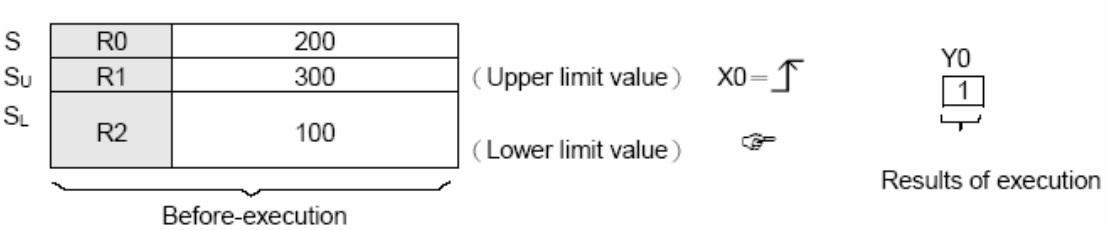
اگر بزرگتر از S_U باشد ،"S>U" فعال می شود

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

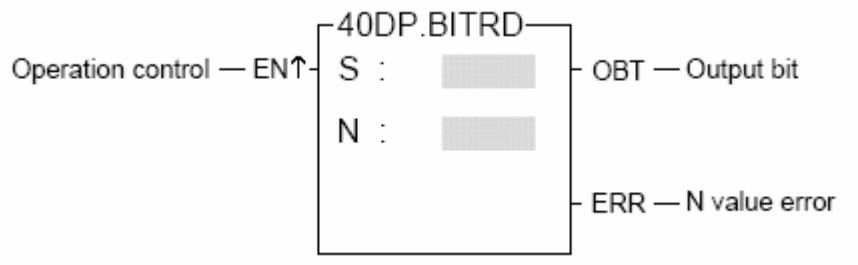
اگر $S_U < S_L$ باشد ،ERROR داده خواهد شد.

مثال:





40.BIT READ



داده های منبع در S ریخته می شود.

هرگاه "EN" از 0 به 1 تغییر کند:

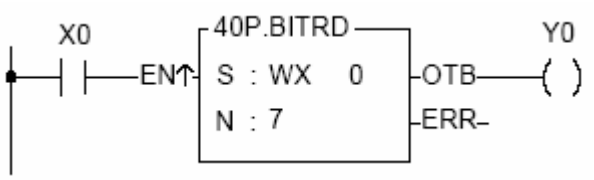
Nاُمین بیت داده ی S در خروجی "OBT" قرار می گیرد.

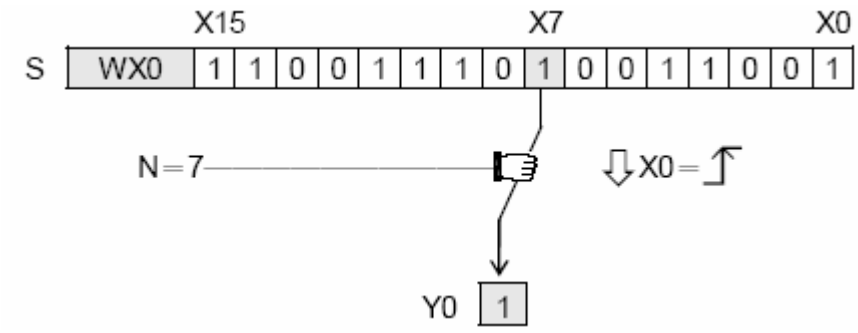
اگر داده ی S ، 16 بیتی باشد => N:0~15

اگر داده ی S ، 32 بیتی باشد => N:0~31

در غیر این صورت errorمی دهد.

مثال:





41.BIR WRITE

هرگاه "EN" از 0 به 1 تغییر کند:

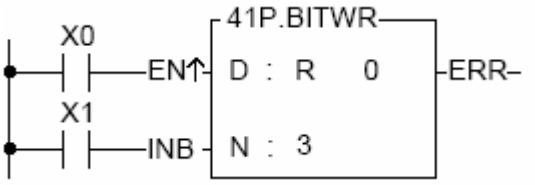
مقدار "INB" در Nامین بیت رجیستر D ریخته می شود.

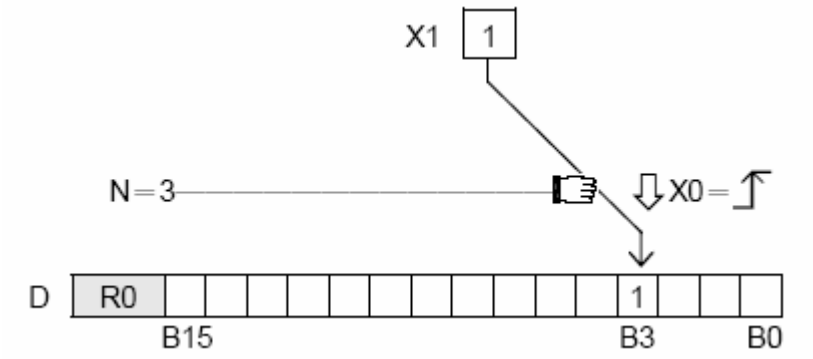
اگر عملوند D، 16 بیتی باشد => N:0~15

اگر عملوند D ، 32 بیتی باشد => N:0~31

در غیر این صورت errorمی دهد.

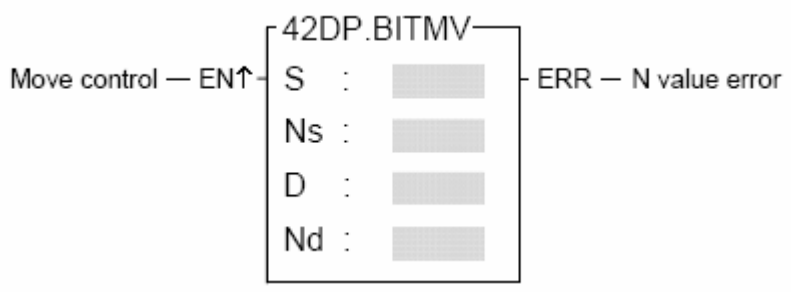
مثال:





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

42.BIT MOVE



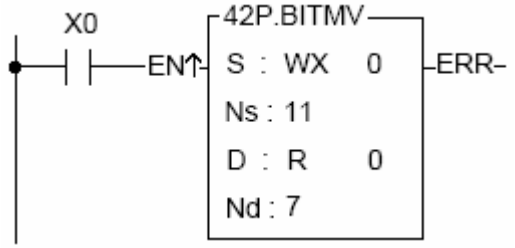
هرگاه "EN" از 0 به 1 تغییر کند:

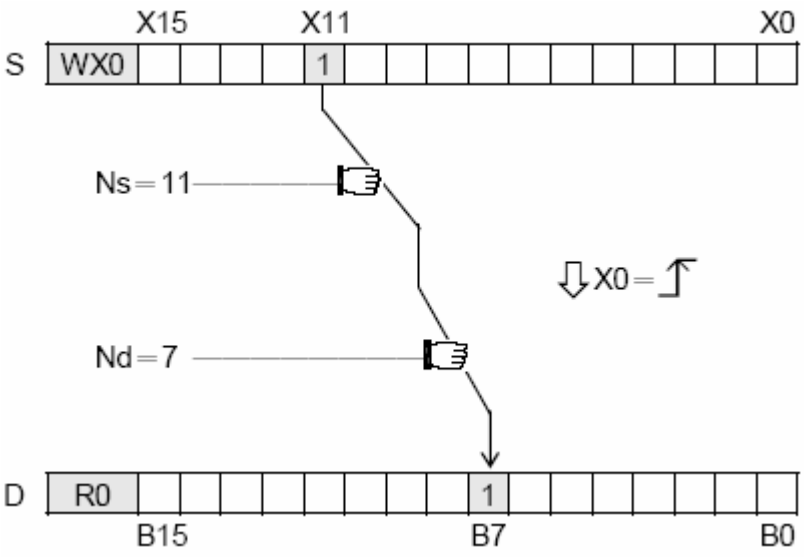
Nsأمین بیت رجیستر S به

Ndأمین بیت رجیستر D ریخته می شود.

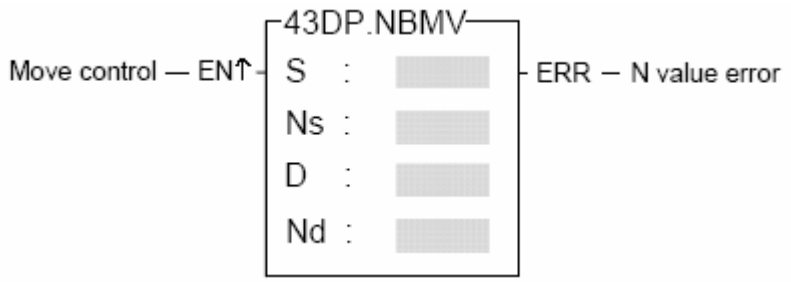
هرگاه مقادیر Ns و Nd متناسب با مقادیر S و D نباشد، error فعال می شود.

مثال:





43.NIBBLE MOVE

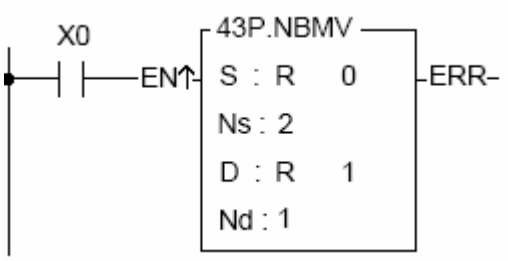


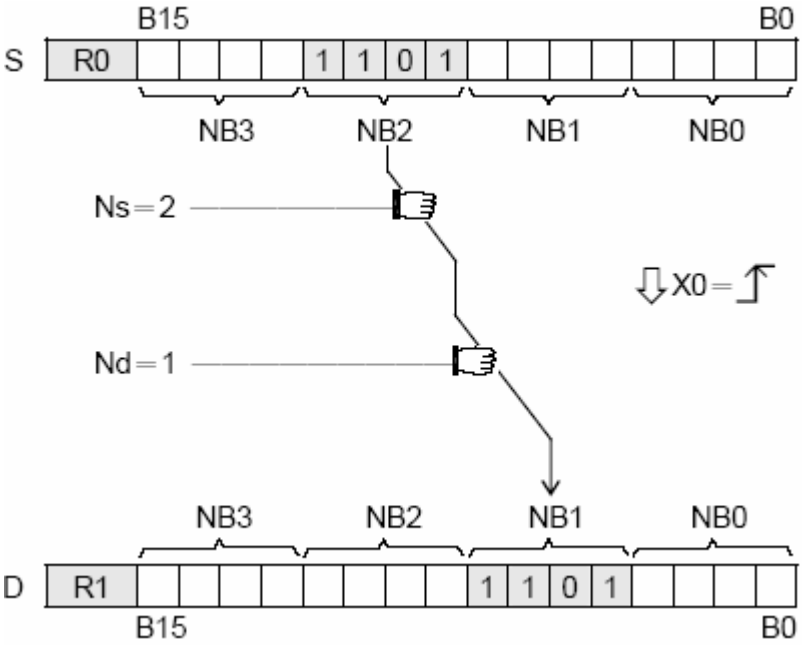
هرگاه "EN" از 0 به 1 تغییر کند:

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

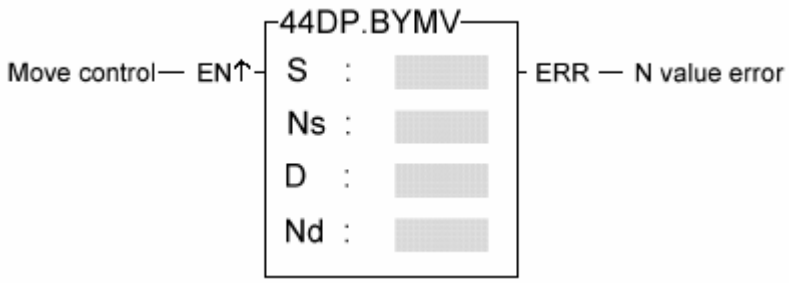
Ndأمین نیبل از رجیستر D منتقل می کند.

مثال:





44.BYTE MOVE



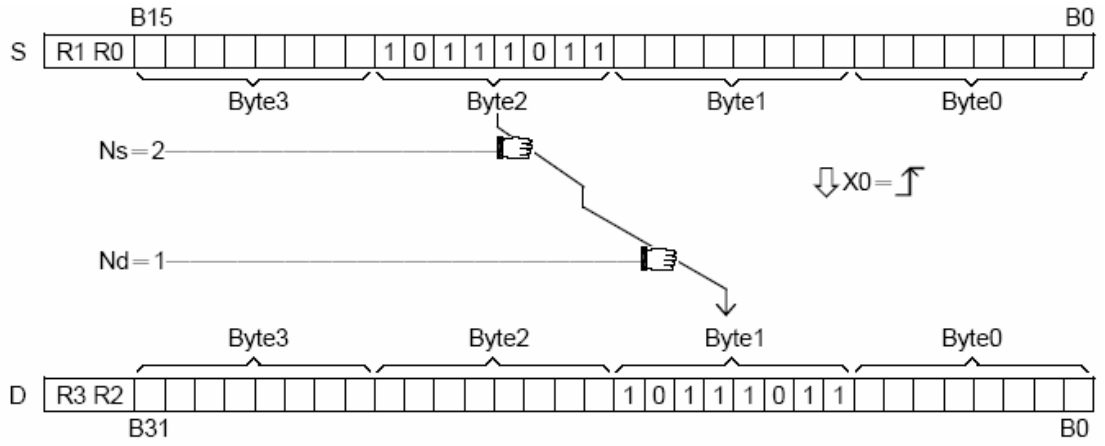
هرگاه "EN" از 0 به 1 تغییر کند:

Nsأمین با یت رجیستر S به

Ndأمین با یت رجیستر D ریخته می شود.

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

مثال:

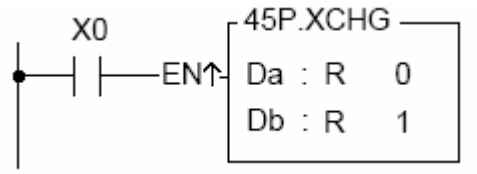


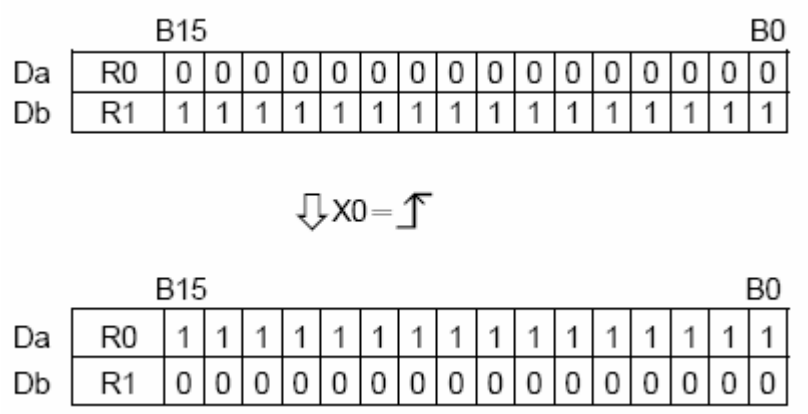
45.EXCHANGE

هرگاه "EN" از 0 به 1 تغییر کند:

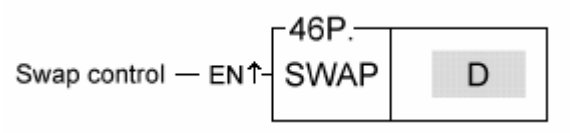
مقادیر رجیستر Da و Db با هم عوض می شوند.

مثال:

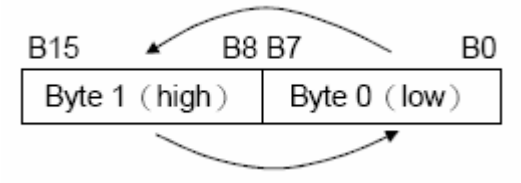




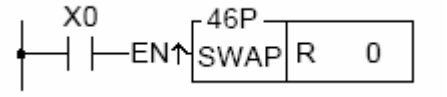
46.BYTE SWAP

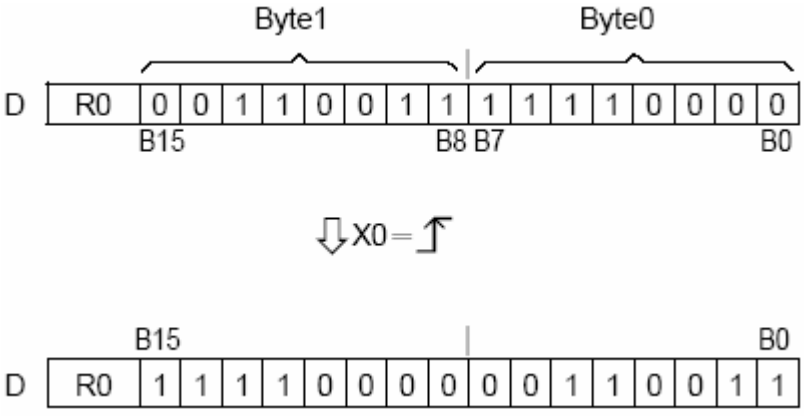


در D یک رجیستر 16 بیتی قرار می گیرد که با فعال شدن "EN"، بایت بالا و پایین آن با هم عوض می شوند.

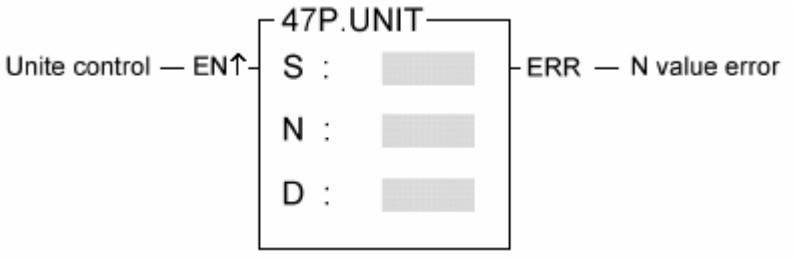


مثال:





47.NIBBLE UNITE



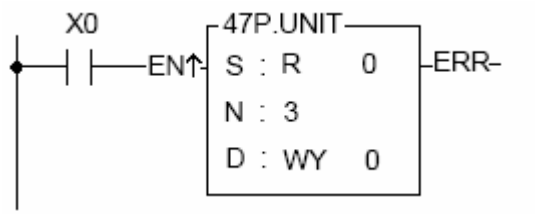
در S اولین رجیستر 16 بیتی مورد نظر قرار می گیرد .

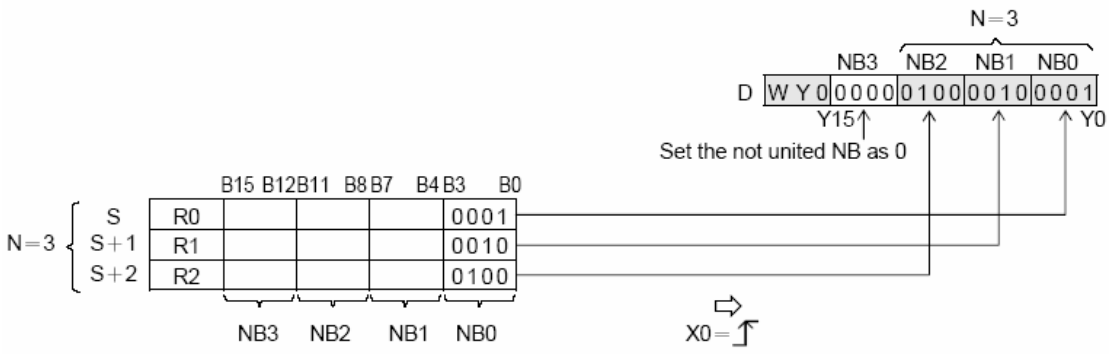
این تابع نیبل های پایین N تعداد از رجیستر ها را به ترتیب در رجیستر D قرار می دهد.

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:





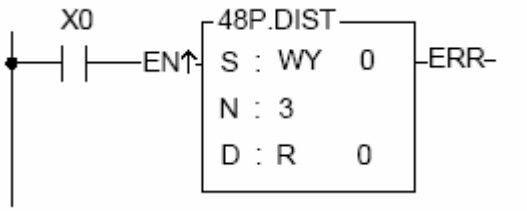
48.NIBBLE DISTRIBUTE

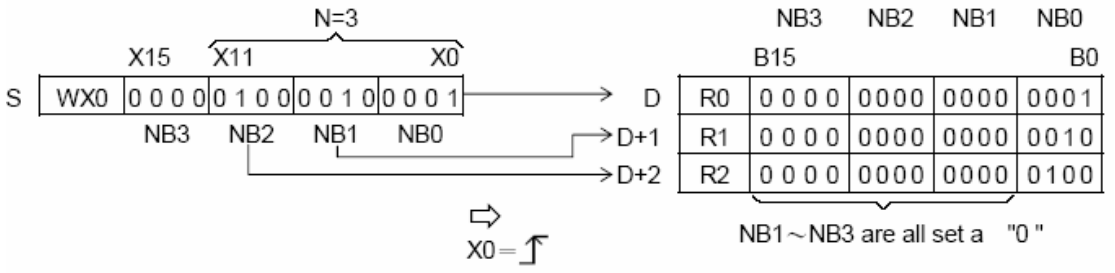
در S یک رجیستر 16 بیتی قرار می گیرد که به ترتیب N تعداد از نیبل های آن به رجیستر D+1،D و... منتقل می شود.

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

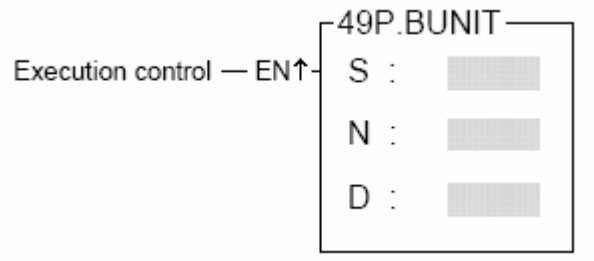
اگر N:1~4 نباشد ، error فعال می شود.

مثال:





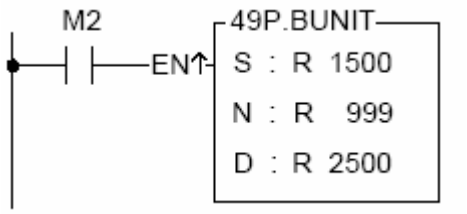
49.BITE UNITE



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

این تابع،بیت ها ی پایین N تعداد از رجیستر های مشخص شده در S را به ترتیب در رجیستر های D،D+1،... قرار می دهد.

مثال:



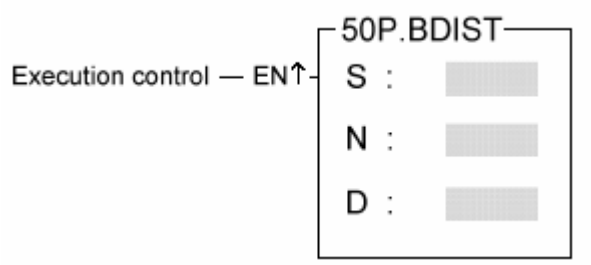
S

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

D

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

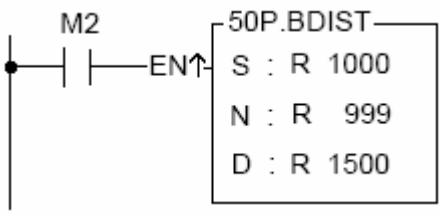
50.BYTE DISTRIBUTE



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

این تابع،با یت ها ی مبدا را (N تعداد) به با یت های پایین رجیستر D،D+1،... منتقل می کند.

مثال:



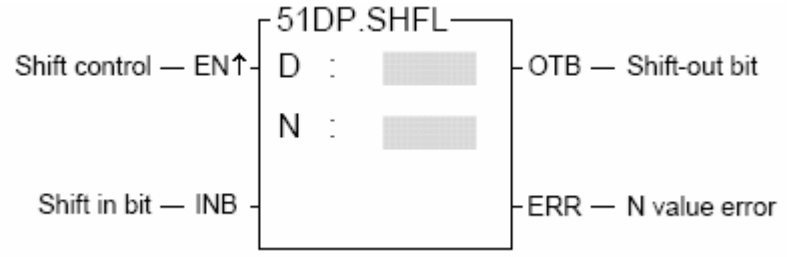
S

	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

51.SHIFT LEFT



در D رجیستری که قرار است شیفت داده شود قرار می گیرد.

N ،تعداد شیفت به چپ را مشخص می کند.

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

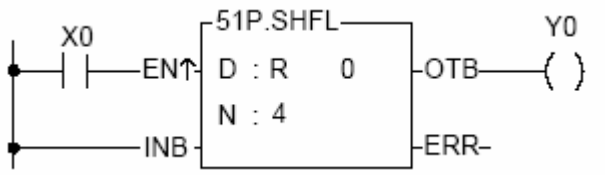
Nأمین بیت بالای رجیستر به خروجی "OTB" می رود.

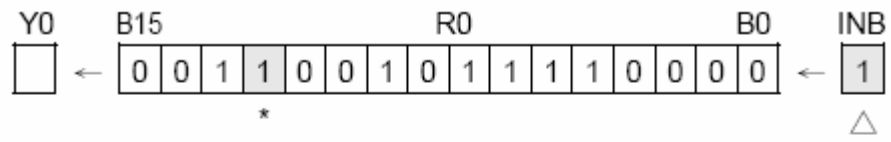
اگر رجیستر مورد نظر 16 بیتی باشد => N:1~16

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

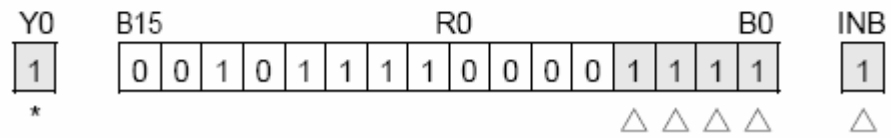
در غیر این صورت errorمی دهد.

مثال:

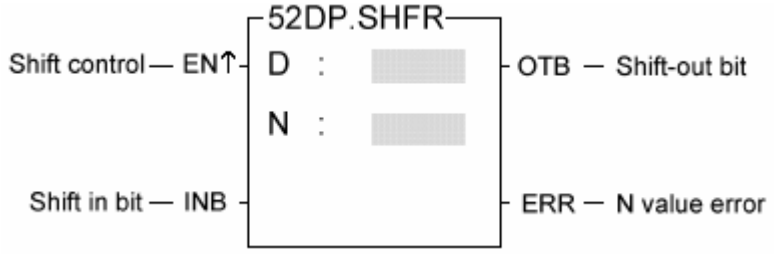




X0 = ↑



52.SHIFT RIGHT



در D رجیستری که قرار است شیفت داده شود قرار می گیرد.

N ،تعداد شیفت به راست را مشخص می کند.

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

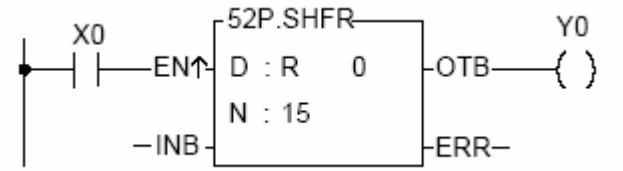
Nامین بیت پایین رجیستر به خروجی "OTB" می رود.

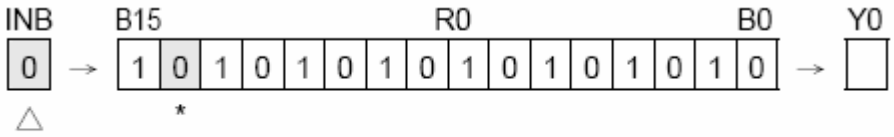
اگر رجیستر مورد نظر 16 بیتی باشد => N:1~16

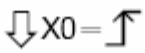
اگر رجیستر مورد نظر 32 بیتی باشد => N:1~32

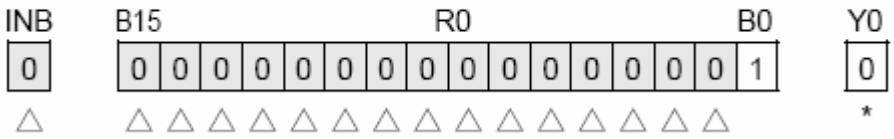
در غیر این صورت errorمی دهد.

مثال:

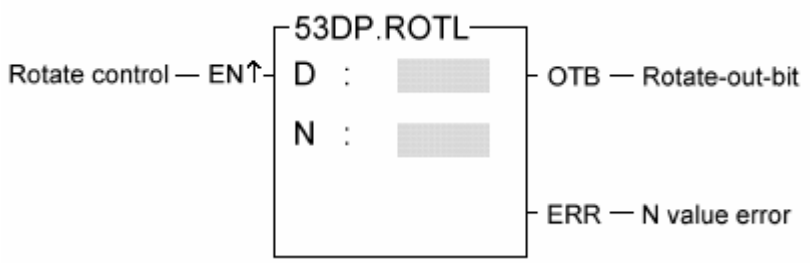








53.ROTATE LEFT



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

چپ ترین بیت به راست ترین بیت منتقل می شود و

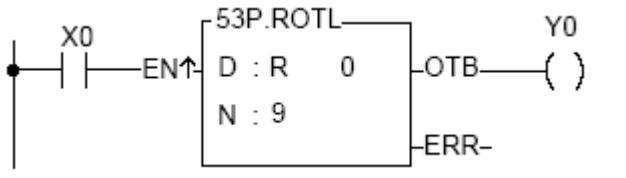
یک کپی از آن به خروجی "OTB" می رود.

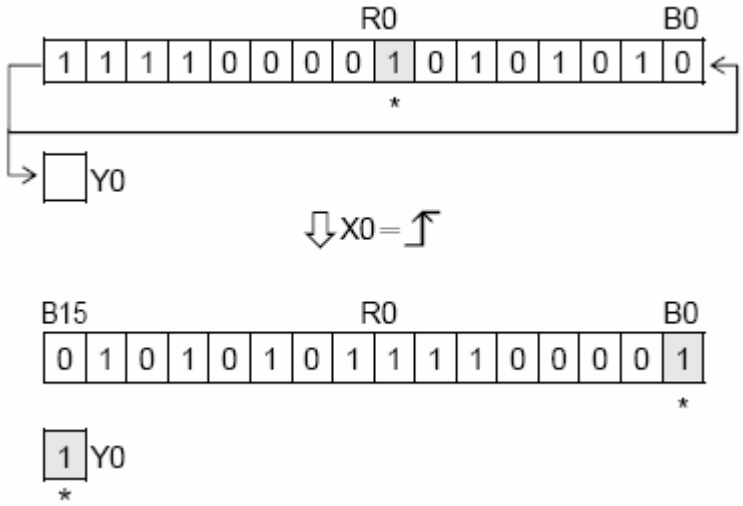
اگر رجیستر مورد نظر 16 بیتی باشد => N:1~16

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

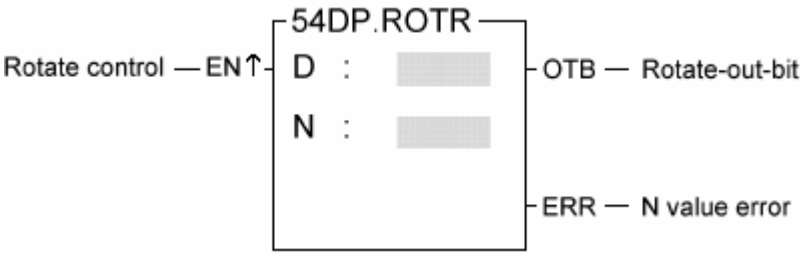
در غیر این صورت errorمی دهد.

مثال:



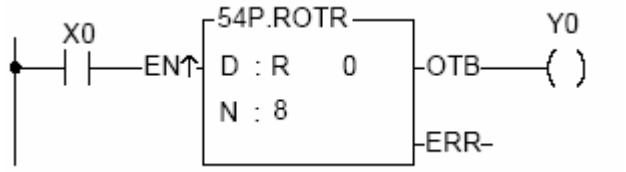


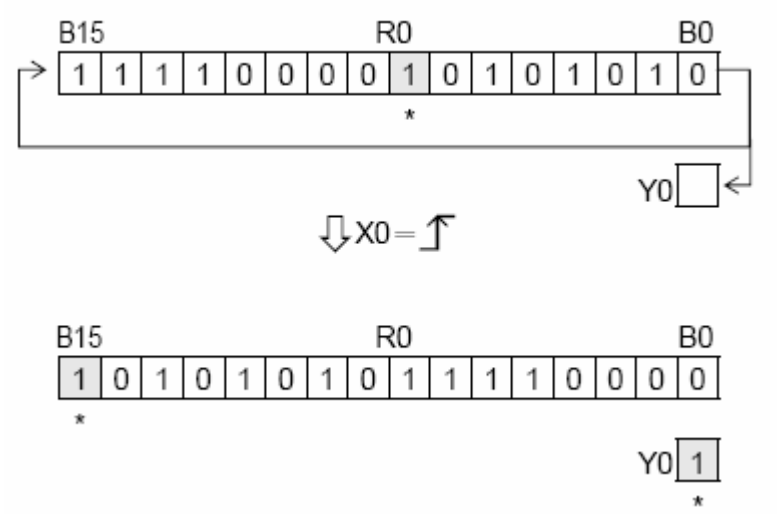
54.ROTATE RIGHT



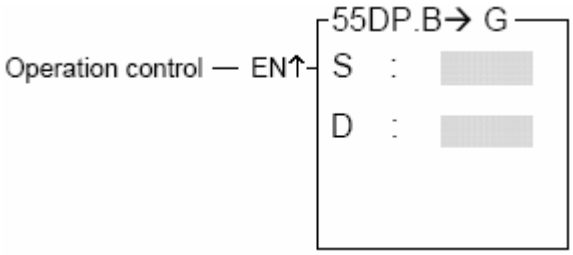
با این تابع یک چرخش به سمت راست در محل بیت ها انجام می گیردو
راست ترین بیت به چپ ترین بیت منتقل می شود و
یک کپی از آن به خروجی "OTB" می رود.
اگر رجیستر مورد نظر 16 بیتی باشد => N:1~16
اگر رجیستر مورد نظر 32 بیتی باشد => N:1~32
در غیر این صورت errorمی دهد.

مثال:





55.BINARY TO GRAY



هرگاه "EN" از 0 به 1 تغییر کند:

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

56. GRAY TO BINARY

هرگاه "EN" از 0 به 1 تغییر کند:

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

57.DECODE

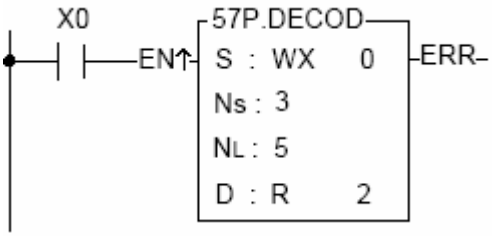
رجیستری که کدگشایی می شود در S ریخته می شود و نتیجه در رجیستر D،D+1،...
می شود.

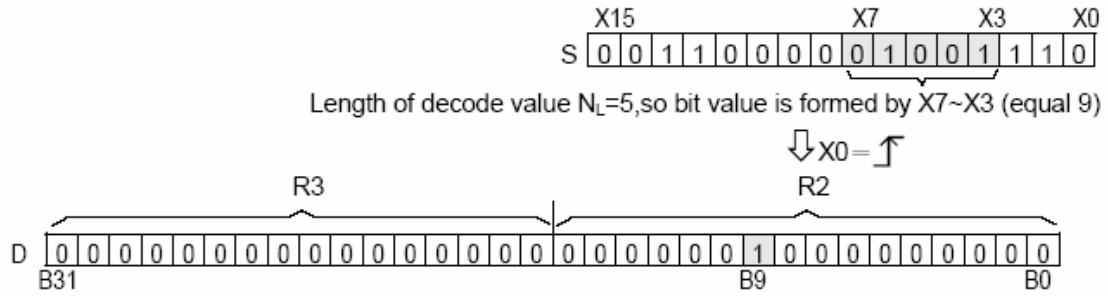
طول رجیستر D : 2^{NL} بیت می شود.

مقداری که کدگشایی می شود از Nsامین بیت 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 9ام(B9) از رجیستر مقصد (D) خواهد شد.

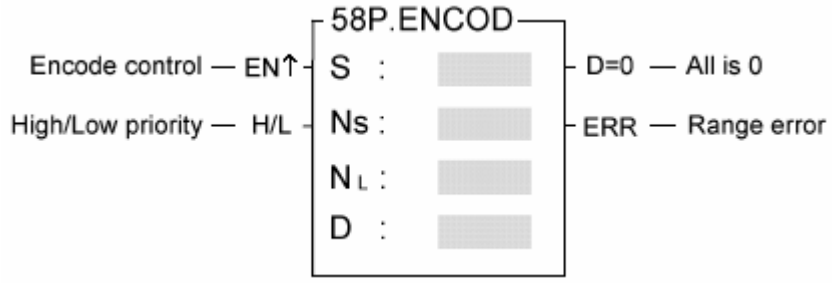
در S یک رجیستر 16 بیتی ریخته می شود پس اگر مقدار Ns یا N_L متناسب با این رنج نباشد، error داده خواهد شد.

مثال:





58.ENCODE



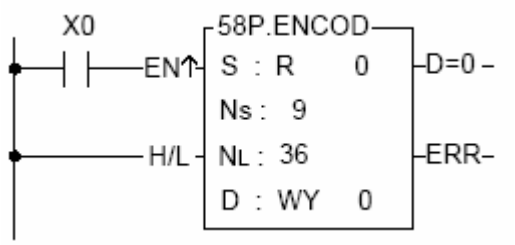
اولین رجیستراز رجیستر های مورد نظر برای کدگذاری در S ریخته می شود و مقدار نهایی کدگذاری شده در D .

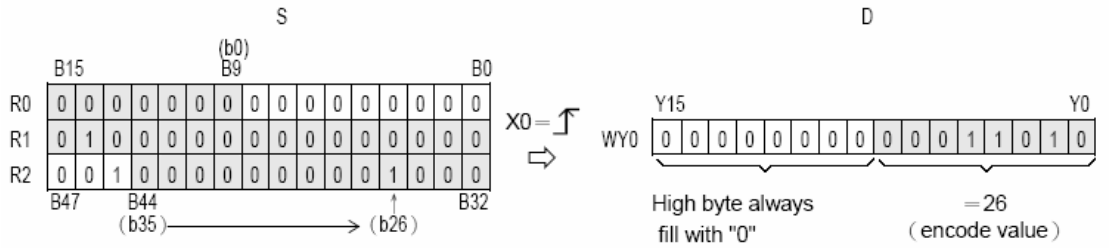
مقدار کدگذاری از روی (Ns+1)اُمین بیت رجیستر S به طول N_L بیت ،تعیین می شود.

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

هرگاه "H/L"=0 باشد، کدگذاری با اولویت پایین انجام می شود.

مثال:

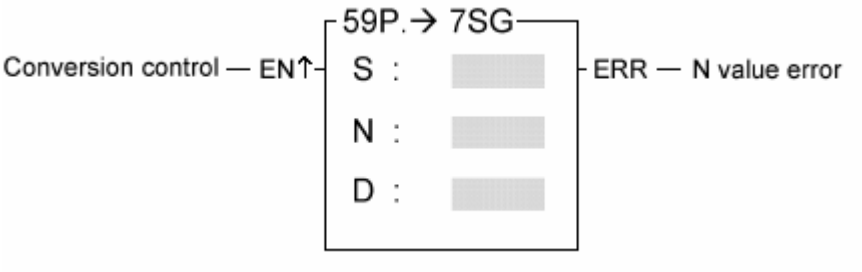




اولین بیت با مقدار1

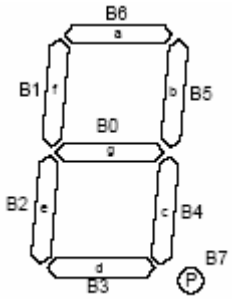
برای کدگذاری با اولویت بالا

59.7-SEGMENT CONVERSION



در S یک رجیستر 16 بیتی قرار می گیرد که هر نیبل(4 بیت) آن معرف یک عدد هگز است که از طریق این تبدیل ،به یک عدد 8 بیتی تبدیل می شود(کد 7-seg) که این عدد در D ذخیره می شود.

سگمنت های 7-seg با حروف زیر،علامت گذاری شده اند:



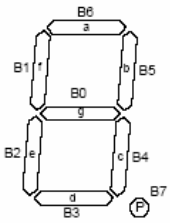
هرگاه بیت B6 از رجیستر D ،1 شود ،معادل نمایش سگمنت a از 7-seg می باشد

هرگاه بیت B5 از رجیستر D ،1 شود ،معادل نمایش سگمنت b از 7-seg می باشد و...

تعداد نیبل هایی که تبدیل 7-seg روی آنها انجام می گیرد N+1 است.

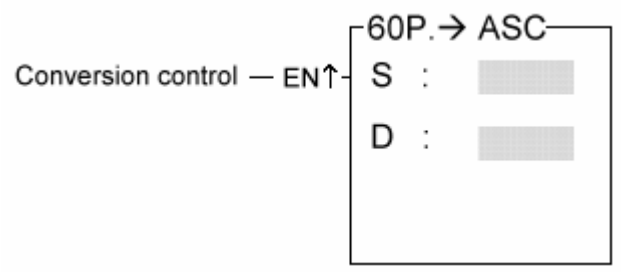
رنج موثر N ،0~3 است در غیر این صورت error داده خواهد شد.

اگر از 7-seg FATEK (FB-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

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

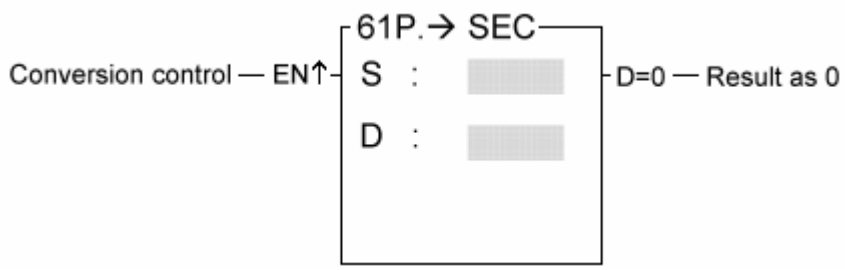
Alphabet
ABCDEF

X0 =

D

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

61.H:M:S to SECONDS CONVERSION



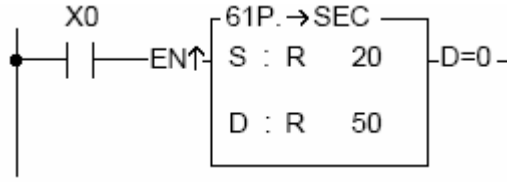
هرگاه "EN" از 0 به 1 تغییر کند:

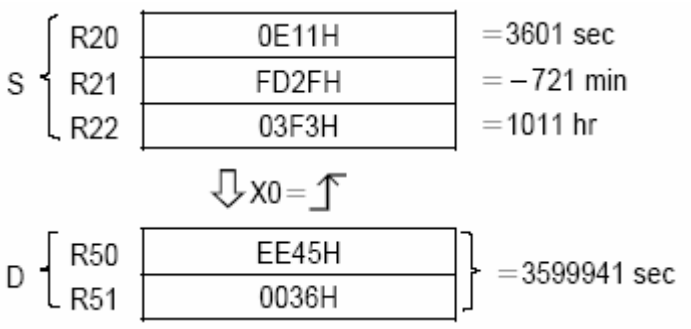
این تابع زمانی را که به صورت ساعت ،دقیقه و ثانیه در رجیسترهای S~S+2 ذخیره شده است را بر حسب ثانیه در رجیستر D ذخیره می کند.

اگر نتیجه 0 باشد ،"D=0" فعال می شود.

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

مثال:





62.SECOND to H:M:S

هرگاه "EN" از 0 به 1 تغییر کند:

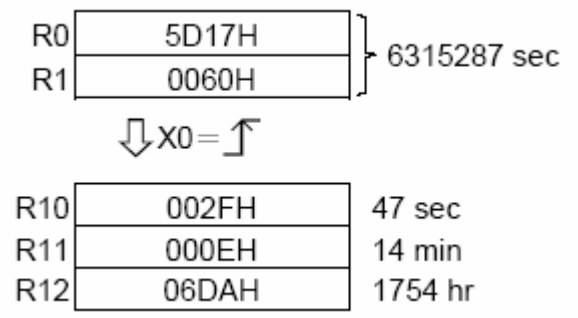
این تابع بر عکس تابع قبل عمل می کند و زمان بر حسب ثانیه را بر حسب ساعت،دقیقه و ثانیه تبدیل می کند.

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

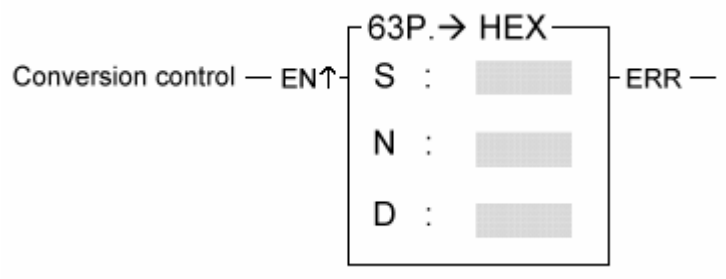
اگر مقدار موجود در S، 0 باشد ،"D=0" فعال می شود.

مقدار مناسب برای S، 117964799~117968399- ثانیه می باشد در غیر این صورت "OVF" (OVER RANGE) فعال می شود.

مثال:



63.ASCII to HEX

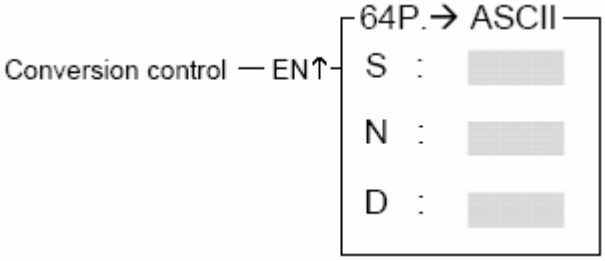


هرگاه "EN" از 0 به 1 تغییر کند:

N تعداد از کدهای ASCII که در S ذخیره شده اند،به معادل هگز خود تبدیل شده و در D ذخیره می شوند.

هدف اصلی این تابع تبدیل کد ASCII کاراکترهای (’0’~’9’ و ’A’~’F’) به معادل هگز آنها می باشد و برای دستگاه هایی کاربرد دارد کهCPU می تواند کد هگز را پردازش کند بنابراین اگر مقدار موجود در S ،معادل ASCII کاراکترهای (’0’~’9’ و ’A’~’F’) نباشد، "ERR" فعال خواهد شد.

64.HEX 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،D+1،... ذخیره می کند.

کاربرد این تابع در این است که اعداد هگز پردازش شده توسط PLC را به کد ASCII برای ارتباط از طریق پورت ها تبدیل کند.

PROGRAM END

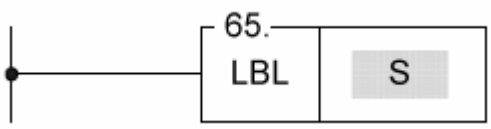
هرگاه "EN" از 0 به 1 تغییر کند:

این تابع فعال شده و به ابتدای برنامه می رود.

تمام برنامه ها بعد از تابعEND دیگر اجرا نمی شوند.وقتی "EN"=0 ،این تابع نادیده گرفته می شود و برنامه های بعد از این تابع اجرا نخواهند شد.

به کار بردن END در برنامه اصلی ضرورتی ندارد زیرا CPU ،به صورت خودکار وقتی به انتهای برنامه رسید،به نقطه شروع می رود.

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 JUMP، CALL، INTERRUPT استفاده می شود.همچنین برای آسان خوانی برنامه می توان به کار برد.

در S نام برچسب متشکل از حروف و اعداد نوشته می شود که از 1 تا 6 حرف می تواند داشته باشد.

اسامی موجود در جدول زیر برای تابع های INTERRUPTی 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.

66.JUMP

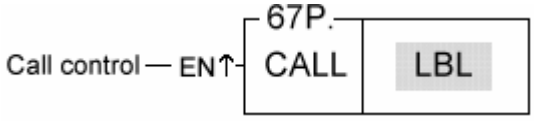
هرگاه "EN" از 0 به 1 تغییر کند:

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.

67.CALL

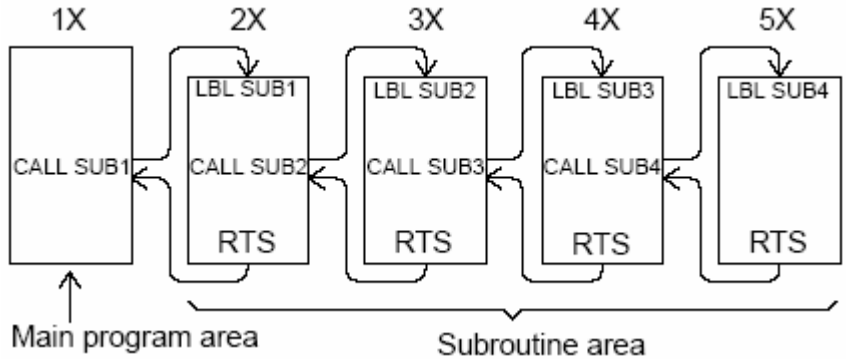


هرگاه "EN" از 0 به 1 تغییر کند:

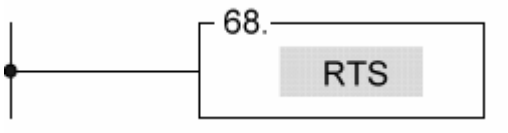
PLC به زیربرنامه ا ی می رود که تابع CALL از طریق LABLE نوشته شده در تابعآن را فراخوانی می کند و به اجرای آن زیربرنامه می پردازد تا زمانی که به تابع RTS بربخورد ،سپس به همانجایی بر می گردد که تابع CALL در ان نوشته شده و از ادامه تابع CALLبه اجرای برنامه می پردازد.

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.

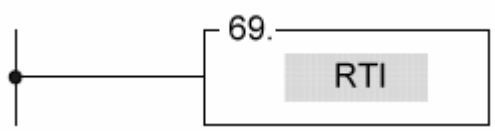


68.RTS



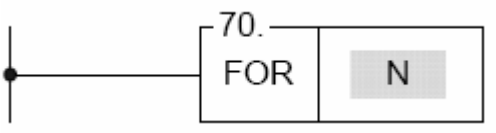
این تابع برای نشان دادن پایان یک زیربرنامه به کار می رود و PLC بعد از دیدن این تابع به انتهای تابع CALLی می رود که از طریق آن ، این زیربرنامه فراخوانی شده است و به اجرای تابع ها بعد از CALL می پردازد.

69.RTI



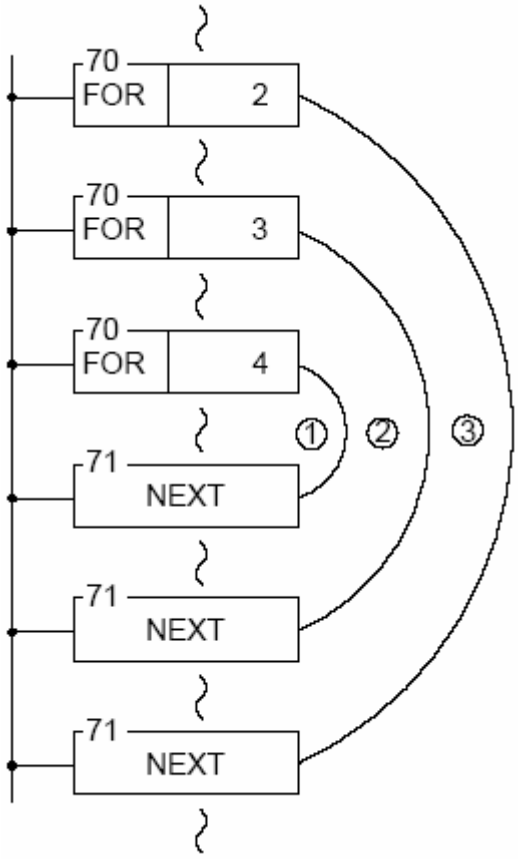
این تابع شبیه تابع RTS است با این فرق که RTS در انتهای اجرای زیربرنامه قرار می گیرد در حالی که RTI در انتهای روتین INTERRUPT قرار می گیرد.
در مقایسه با CALL که از طریق یک LABLE ،زیربرنامه مورد نظر را اجرا می کند، INTERRUPT مستقیما از طریق سیگنال های سخت افزاری فعال شده و در کار CPU وقفه ایجاد می کند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

70.FOR

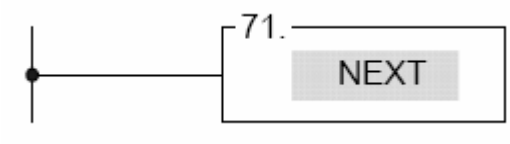


این تابع به همراه تابع NEXT ،تشکیل یک حلقه را می دهد که برنامه ی میان FOR و NEXT مربوطه ،N بار اجرا می شود.حلقه های FOR و NEXT دیگری نیز در میان حلقه های FOR و NEXT اولیه می تواند قرار بگیرد.

به مثال توجه کنید:

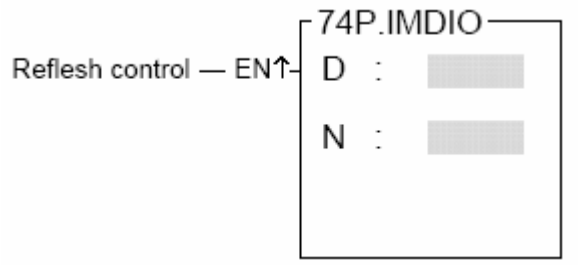


71.LOOP END



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

74.IMMIDIATE I/O



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

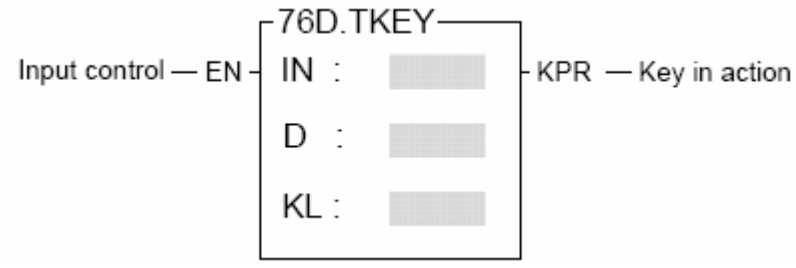
هرگاه "EN" از 0 به 1 تغییر کند:

N سیگنال ورودی یا خروجی ،refresh خواهند شد.

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

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

76.DECIMAL-KEY INPUT



این تابع می تواند ورودی 0~9 را توسط کیبورد دریافت کند که این اعداد معادل با IN~IN+9 قرار می گیرند.بر اساس ترتیب کلیدهای فشرده شده،یک عدد دهدهی 4 یا 8 رقمی می تواند در رجیسترمشخص شده در D ذخیره شود.

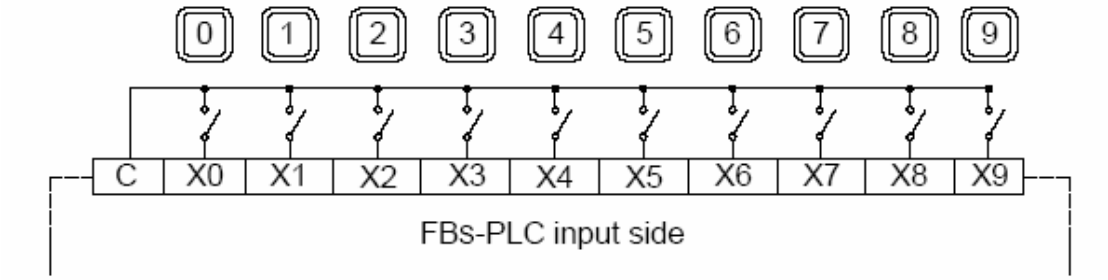
اگر رجیستر موجود در D، 16 بیتی باشد ،این عدد می تواند 4 رقمی باشد و اگر 32بیتی باشد،8 رقمی خواهد بود.

این تابع در صورتی عمل می کند که "EN"=1 باشد.

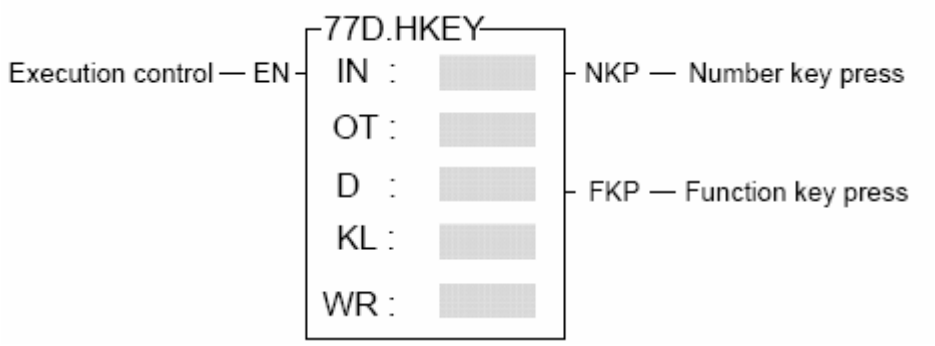
هرگاه هریک از کلیدها فشرده شود ، خروجی "KPR" به اندازه ی همان مدت فشرده شدن کلید،1 خواهد ماند.

هرگاه هریک از اعداد فشرده شود ، رجیستر 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 رجیستر 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.

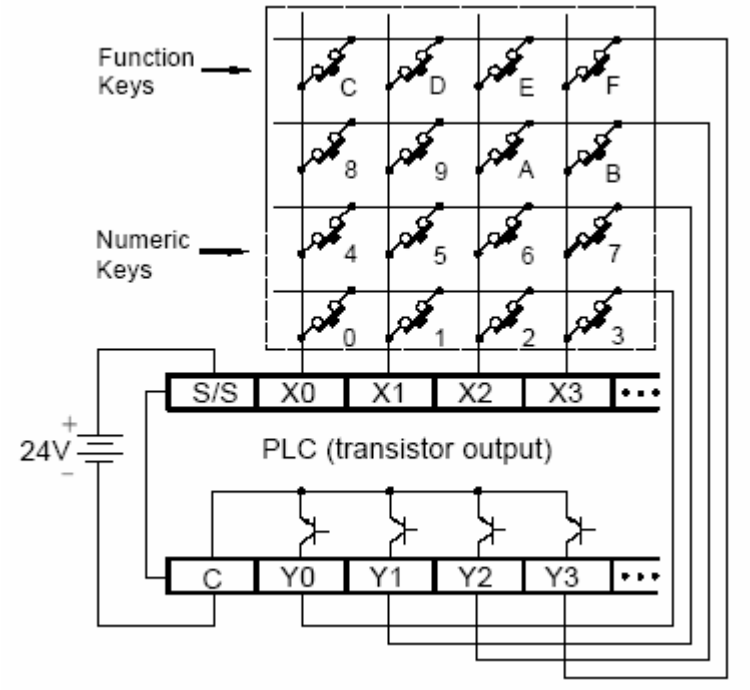


77.HEX-KEY INPUT



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.

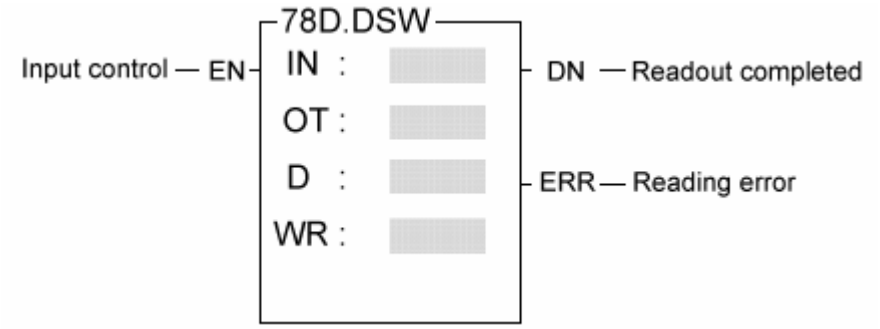
همچنین اتصال سخت افزاری این تابع با تابع قبل متفاوت است ،4 ورودی اول PLC به 4 خروجی اول PLC طوری متصل می شوند که اتصال هر کدام از آنها یکی از خروجی ها را بدهد.



هرگاه هریک از اعداد0~9 فشرده شود ،خروجی "NKP" به اندازه همان مدت فشرده شدن کلید ،1 خواهد ماند و هرگاه یکی از دستورات A~F فشرده شوند،خروجی "FKP" به اندازه همان مدت فشرده شدن کلید ،1 خواهد ماند.

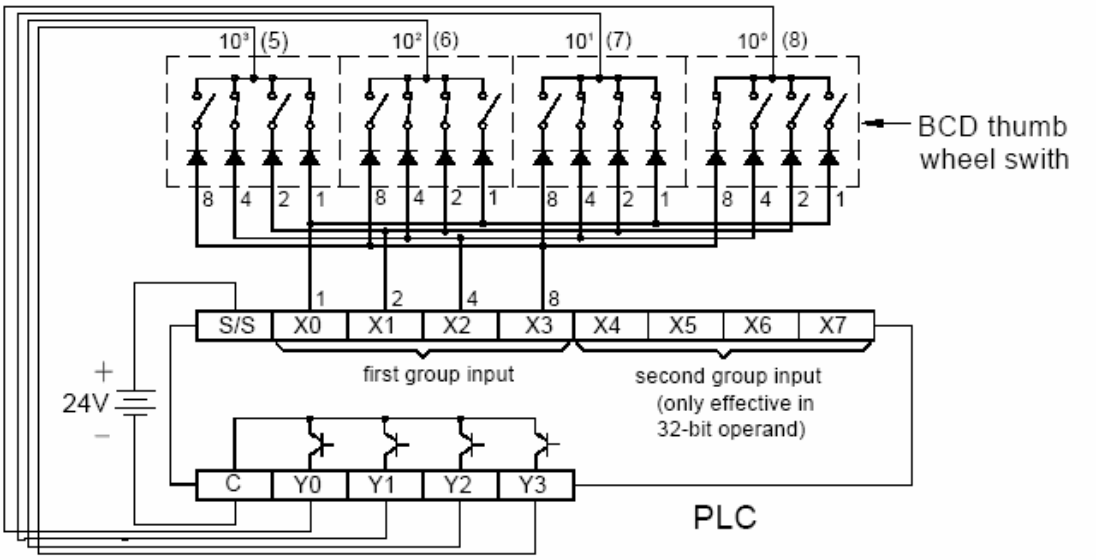
رجیستر ذخیره شونده در WR برای استعمال تابع است و در جای دیگری نباید از آن استفاده کرد.

78.DIGITAL SWITCH INPUT



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

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



هربار که 4 رقم از سوئیچ ها خوانده شد ،1="Dn" می شود.

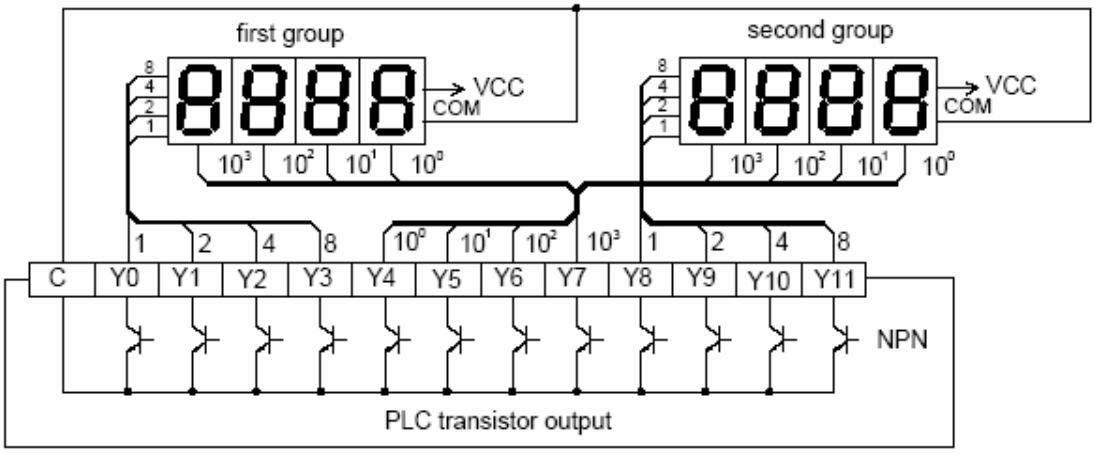
اگر هریک از اعداد خوانده شده در رنج BCD(0~9) نباشد،"ERR" فعال می شود.

PLC مورد استفاده باید دارای خروجی ترانزیستوری باشد.

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

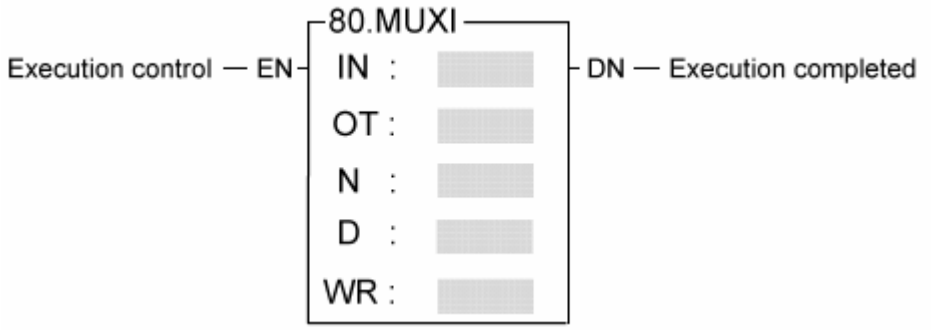
79.7-SEG OUTPUT WITH LATCH

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



PLCهای FACON هر دو نوع خروجی ترانزیستوری NPN و PNP را داراست.

80.MULTIPLEX INPUT



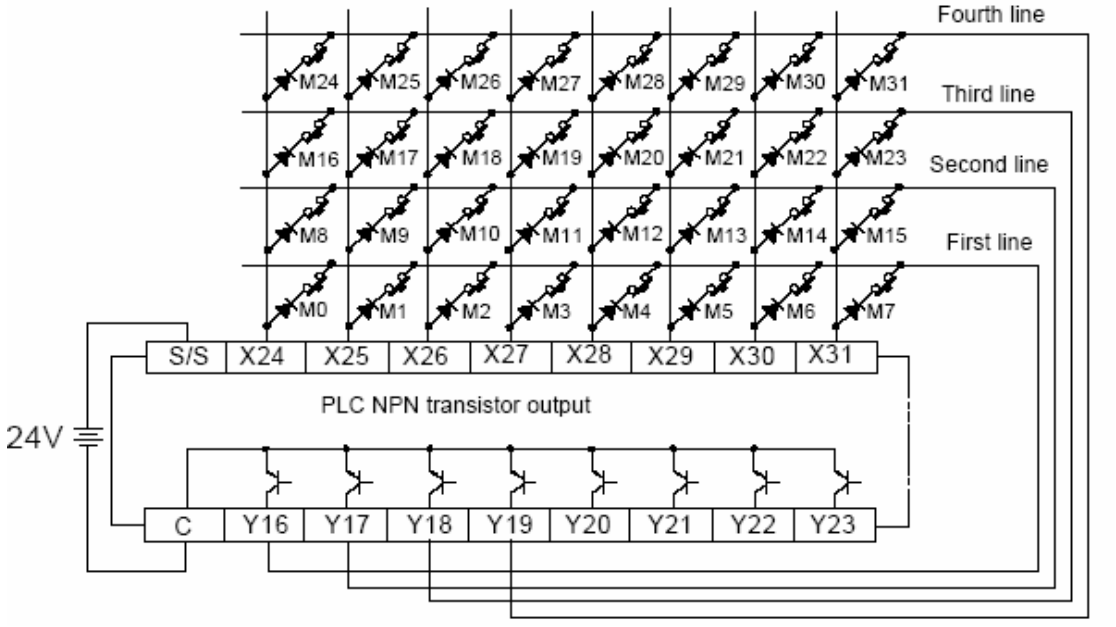
این تابع از متد مالتی پلکس برای خواندن 8×N ورودی استفاده می کند که فقط 8 ورودی و N خروجی از PLC را استفاده می کند.

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

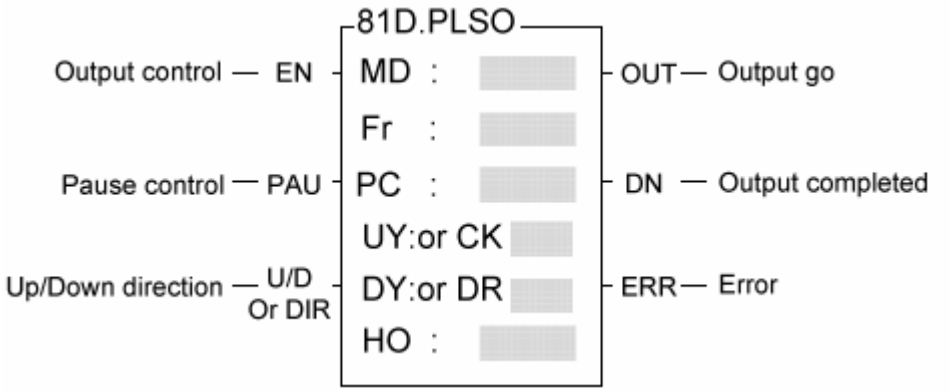
در هر اسکن ،یکی از Nخروجی ،1 می شود و خط مربوط به آن خروجی انتخاب می شود.

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

پس از اسکن تمام خطوط (Nخط) ،8×N مشخصه در رجیستر D ذخیره می شود،سپس "DN" یک می شود.اما فقط به اندازه یک اسکن ،یک می ماند.



81.PULSE OUTPUT



وقتی MD=0 :

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

اگر U/D=1: پالس به UY می رود.

اگر U/D=0: پالس به DY می رود.

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

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

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

در طول انتقال پالس،این تابع به چک کردن مقدار فرکانس (FR) و تعداد پالس های نهایی (PC) می پردازد.بنابراین تا زمانی که فرستادن پالس پایان نیافته،می توانید Fr و PC را تغییر دهید .اما تنظیم جهت پالس ("U/D") تنها هنگام ری ست کردن،قابل اجرا خواهد بود.(وقتی "EN" از 0 به 1 تغییر می کند) و به همان حالت باقی می ماند تا زمانی که انتقال پالس متوقف شود یا ری ست دیگری اتفاق بیافتد.

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

وقتی MD=1 :

پالس خروجی به روی پایه کنترلی "DIR" و رجیستر CK اثر می گذارد.

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

رنج موثر PC برای رجیستر 16 بیتی :0~32767

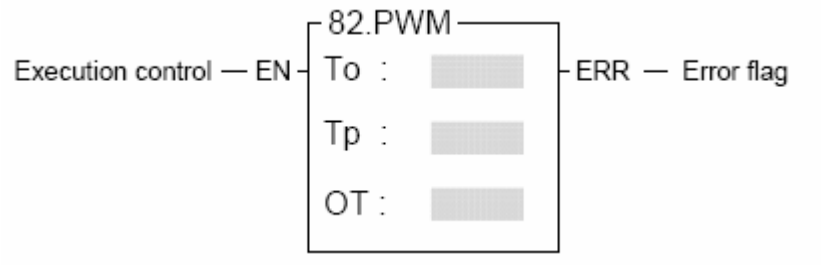
رنج موثر PC برای رجیستر 32 بیتی : 0~2147483647

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

رنج موثر FR: 8~2000

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

82.PULSE WIDTH MODULATION



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

حداقل مقدار To، 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، "ERR" فعال شده و تابع عمل نخواهد کرد.

این تابع تنها یک بار می تواند استفاده شود.

83.SPEED DETECTION

این تابع برای به دست آوردن سرعت گردش دستگاه های چرخنده (مانند موتور) بر حسب rpm استفاده می شود؛ به این صورت که فرکانس سیگنال ورودی را از طریق 8 ورودی سرعت بالای PLC(X0~X7) به دست می آورد و با کمک زمان نمونه برداری (TI) ، تعداد پالس ورودی S را مشخص می کند.

مطلوب است که هنگام استفاده از این تابع ، طراحی به صورتی انجام گیرد که پالس بیشتری در هر چرخش تولید شود تا نتیجه ی بهتری به دست آید،اما مجموع فرکانس تمام سیگنال های آشکار شده باید کمتر از 5KHz باشد،در غیر این صورت WDT(Watch Dog Timer) فعال می شود.[تابع 90]

رجیستر D برای ذخیره نتایج از سه رجیستر 16 بیتی متوالی استفاده می کند(D0~D2)

در D0 نتیجه ی محاسبه ذخیره می شود.

در D1 مقدار شمارش کنونی ذخیره می شود.

در D2 زمان نمونه برداری ذخیره می شود.

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

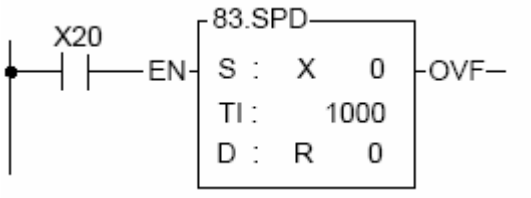
D0 یک رجیستر 16 بیتی است پس حداکثر آن می تواند 32767 باشد ،اگر پریود نمونه برداری خیلی طولانی باشد یا سرعت پالس ورودی خیلی زیاد باشد،مقدار D0 ممکن است از 32767 بیشتر شود در این صورت 1="OVF" شده و عملیات متوقف می شود.

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

سرعت :

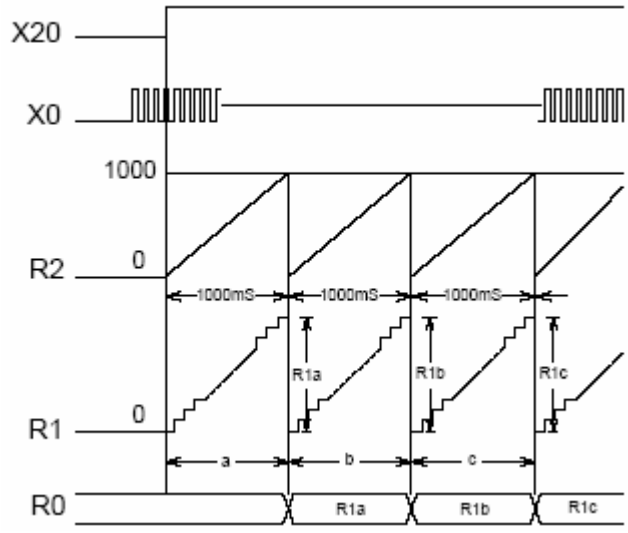
$$N = \frac{(D0) \times 60}{n \times TI} \times 10^3 \quad (rpm)$$

مثال:

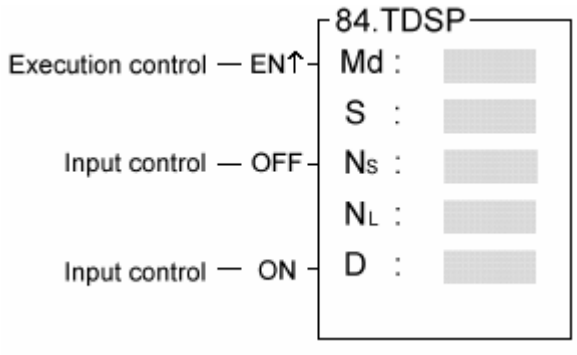


اگر n=60 باشد:

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



84.16/7-SEG DISPLAY



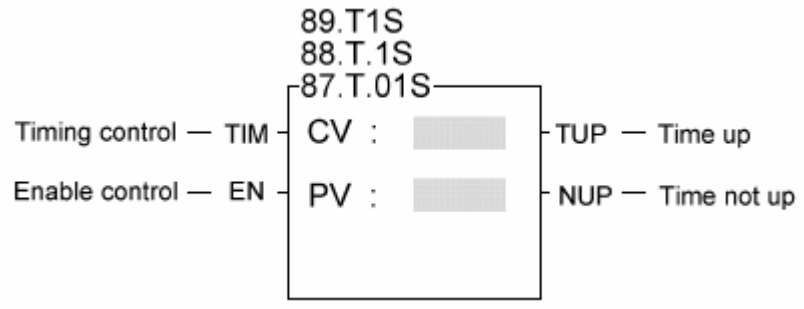
این تابع برای مدل های FBs-7SG1 و FBs-7SG2 کاربرد دارد و می تواند کاراکترهای مختلف را برای نمایش در 16-seg و 17-seg آماده کند.

در S آدرس شروع کاراکتر های مورد نظر برای تبدیل ، ذخیره می شود.
Ns ،مقداری برای pointer است که مشخص می کند کاراکتر ،دقیقاً از کجا شروع می شود.
N_L طول 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) مبدا (S) به الگوی نمایشی 16 بیتی مربوطه تبدیل می شود.

وقتی "EN"=1، "OFF"=0 ، "ON"=0 و Md=0 باشد،این تابع تبدیل را اجرا می کند.وقتی "OFF"=1 و Md=0، تمام بیت های مربوط به الگوی نمایشی 16-seg صفر خواهند شد.این بدین معنیست که اگر 16-seg در این حالت به PLC متصل شود ،تمام LED های آن خاموش خواهد بود.
وقتی "OFF"=1 و Md=1 ، تمام بیت های مربوط به نمایش 7-seg صفر می شود.
وقتی "ON"=1 و Md=0 ، تمام بیت های مربوط به نمایش 16-seg، 1 می شود.
وقتی "ON"=1 و Md=1 ، تمام بیت های مربوط به نمایش 7-seg، 1 می شود.

MSB / LSB	x000	x001	x010	x011	x100	x101
0000						
0001						
0010						
0011						
0100						
0101						
0110						
0111						
1000						
1001						
1010						
1011						
1100						
1101						
1110						
1111						

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 "TIM"=1 ،مانند تایمر ساده عمل می کند،اما اگر "TIM"=0 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 :

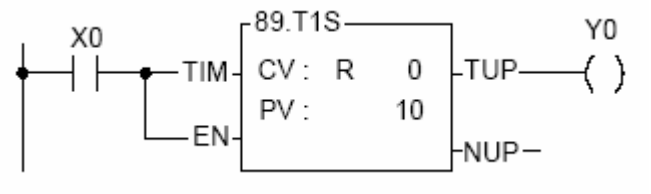
"TUP" که بعد از اتمام محاسبه ی زمان،1 می شود و

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

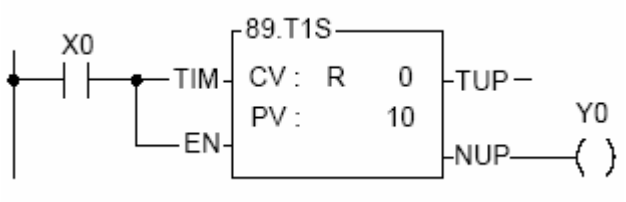
از ترکیب ورودی ها و خروجی ها برای ایجاد تایمرها با کارایی های مختلف می توانید استفاده کنید.

به عنوان مثال:

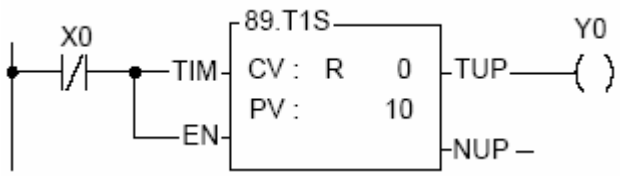
- وقتی X0 ، ON شود،بعد از 10 ثانیه ، ON،YO می شود.



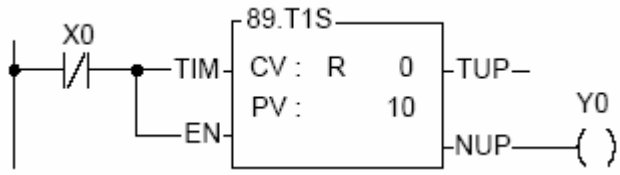
- YO به طور عادی ON است،وقتی XO،ON شود،بعد از 10 ثانیه YO ، OFF می شود.



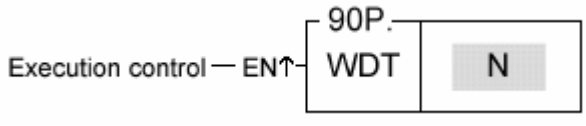
- YO به طور عادی OFF است،وقتی XO، OFF شود،بعد از 10 ثانیه، YO، ON می شود.



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



90.WATCH DOG TIMER



هدف اصلی از طراحی تایمر WDT ، محافظت ویژه ایست که از طرف سیستم اعمال می شود.برای مثال،اگر CPUی PLC ناگهان آسیب ببیند و راهی برای اجرای برنامه یا I/O refresh نباشد،بعد از سپری شدن زمان تعیین شده در WDT، WDT به صورت خودکار تمام I/O ها را خاموش می کند.

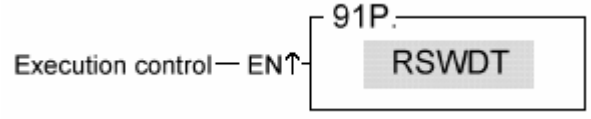
در کاربردهای خاص ،اگر scan time خیلی طولانی باشد ، ممکن است باعث ایجاد مشکلات امنیتی یا عدم پیروی از کنترل شود که این تابع می تواند برای اعمال محدودیت مورد نیاز به روی scan time ، استفاده شود.

هرگاه "EN" از 0 به 1 تغییر کند:

WDT شمارش زمان N×10 ms را آغاز می کند و اگردر این مدت،"EN" تغییر نکند پس از زمان تعیین شده ،PLC خاموش می شود.

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

91.RESET WDT

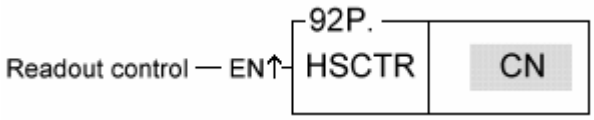


وقتی "EN"، 1 بشود؛

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

92.HSCTR

دسترسی به مقدار کنونی کانتر سرعت بالای سخت افزاری



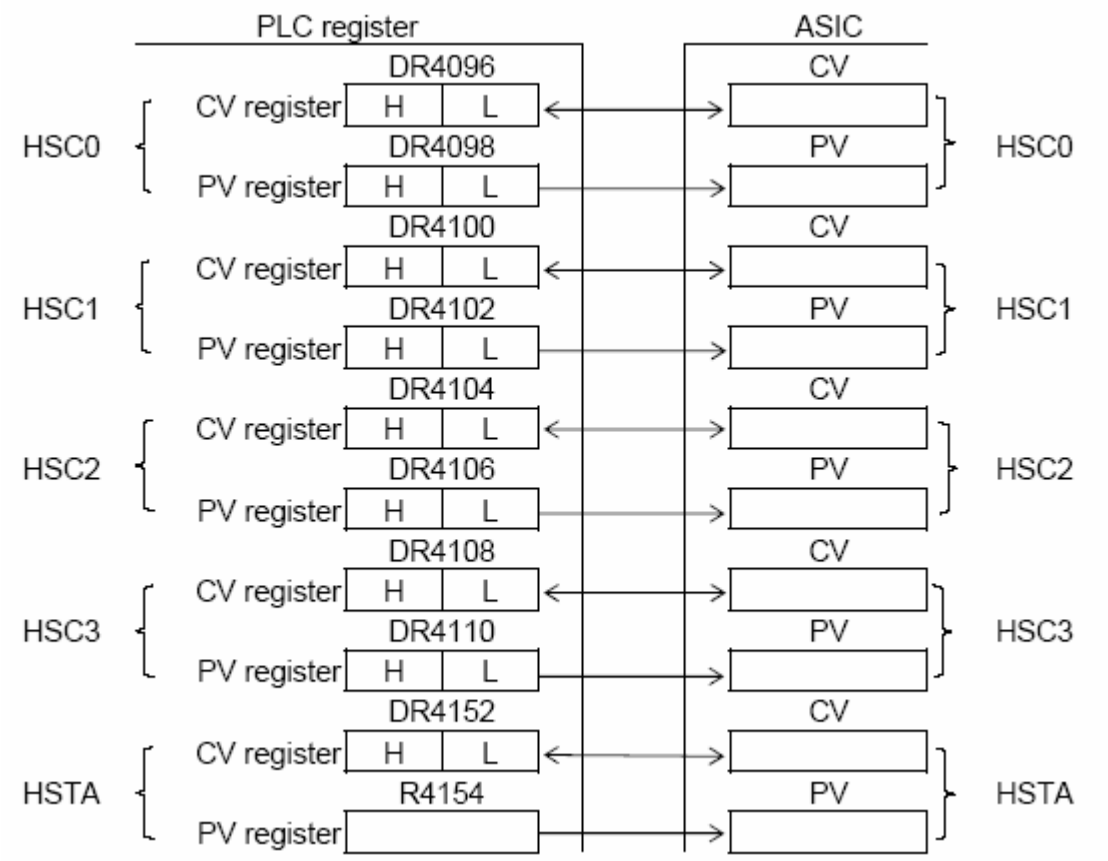
در CN شماره کانتر قرار می گیرد که بیان کننده ی مد های مختلف می باشد مانند پالس کم/زیاد،جهت پالس و فاز AB .

مد 0~3 برای کانترهای سخت افزاری است و می تواند شمارش ،مقایسه و فرستادن اینتراپت بدون مداخله ی CPU را انجام دهد.اما مد 4~7 ،اینتراپت CPU را به کار می گیرند در نتیجه اگر فرکانس شمارش بالا برود،سرعت اسکن PLC ضعیف می شود.

هرگاه "EN" از 0 به 1 تغییر کند:

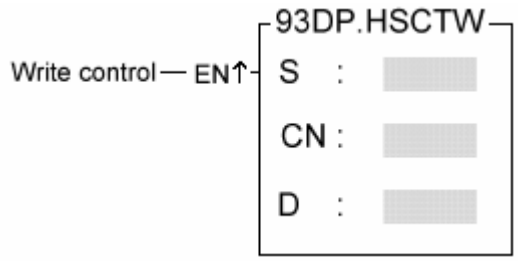
مقدار CV موجود که به صورت سخت افزاری از طریق CN مقدار گذاری شده به داخل رجیستر CV مربوطه ریخته می شود.

نمودار زیر ترتیب CV و PVها در سخت افزارو رجیستر های CV و PV مربوطه آنها را در PLC نشان می دهد.



93.HSCTW

نوشتن مقدار کنونی ومقدار از پیش تعیین کننده کانتر سرعت بالای سخت افزاری



برای دانستن رابطه ی بین CV ، PVهای سخت افزاری و رجیستر های CV و PV مربوطه ی آنها در CPU ،به تابع92 مراجعه کنید.

در CN ،مد مورد نظر را انتخاب کنید.

اگر D را 0 قرار دهید ،تعیین کننده ی CV و

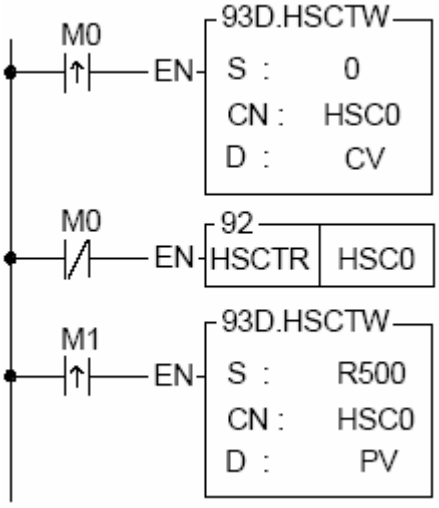
اگر D را 1 قرار دهید،تعیین کننده ی PV می باشد.

هرگاه "EN" از 0 به 1 تغییر کند:

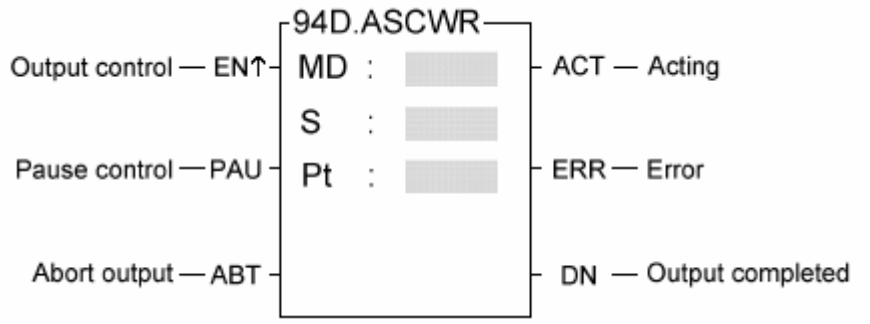
محتویات رجیستر CV یا PV که توسط CN تعیین می شود،به روی CV یا PV سخت افزاری مربوطه نوشته می شود.

یکی از کاربردهای ترکیب این دستور ،اجرای انواع مختلف شمارش دقیق یا کنترل جایگیریهاست.به این صورت که وقتی مقدار CV به PV می رسد،کانتر به سرعت سیگنال اینتراپت می فرستد.از طریق برنامه سرویس اینتراپت می توان کاربردهای فوق را اجرا نمود.

وقتی وقفه ای در منبع تغذیه PLC ایجاد می شود،مقدار سخت افزاری CV به صورت اتوماتیک به داخل رجیسترهای CV نوشته می شود و وقتی توان مجدداً افزایش یافت،این مقادیر به جای خود بازگردانده می شوند. مثال:



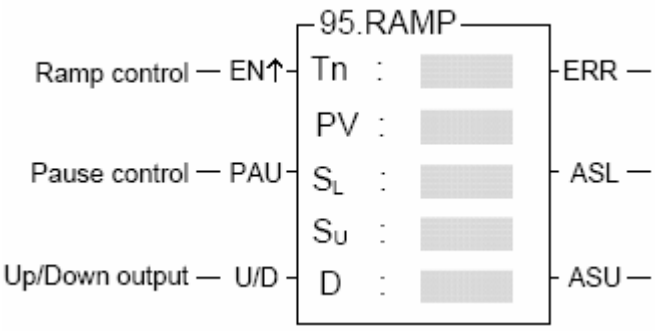
94.ASCII WRITE



هرگاه "EN" از 0 به 1 تغییر کند و MD=0 باشد ،اطلاعاتی را که به صورت ASCII کد شده اند و از S شروع می شوند را به Port1 منتقل می کند تا زمانی که به پایان فایل برسد.
محتویات فایل Sمی تواند از طریق نرم افزار WinProladder پردازش شود.
اطلاعات پردازش شده باید به فرمت ASCII باشد،درغیراین صورت این تابع ،تبادل اطلاعات را متوقف می کند و "ERR" 1 خواهد شد.اگر تمام فایل به صورت صحیح منتقل شود،خروجی"DN" ، 1 می شود.
در طول مدت انتقال اطلاعات ،خروجی "ACT"، 1 خواهد بود.تنها هنگام توقف خروجی، error یا کنسل کردن ،"ACT" به 0 بر می گردد.
این تابع می تواند به صورت پی در پی به کار برده شود اما در یک زمان مشخص ، تنها یکی از آنها اجرا خواهد شد.
وقتی این تابع در حال اجراست ،اگر "PAU"، 1 بشود؛این تابع انتقال اطلاعات را متوقف می کند؛ با برگشتن "PAU" به 0 ،انتقال اطلاعات ادامه می یابد.
وقتی این تابع در حال اجراست ،اگر "ABT"، 1 بشود؛این تابع انتقال اطلاعات را رها می کند ، سپس توانایی اجرای تابع بعدی را خواهد داشت.

95.RAMP

تابع شیب برای خروجی دیجیتال به آنالوگ



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

PV برای تنظیم تایمر شیب می باشد.

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

هرگاه "EN" از 0 به 1 تغییر کند:

ابتدا تایمر Tn ،ری ست شده و 0 می شود.سپس اگر "U/D"=1 مقدار S_L در رجیستر D ذخیره می شود و وقتی رجیستر M1974=0، هر 0.01s مقدار D به اندازه Su - S_L/ PV افزایش می یابد.

وقتی مقدار D به Su رسید،خروجی "ASU"=1 می شود.

اگر "U/D"=0 ،مقدار Su در رجیستر D ذخیره می شود.وقتی M1974=0، هر 0.01s ، مقدار D به اندازه ی Su - S_L/ PV کاهش می یابد.

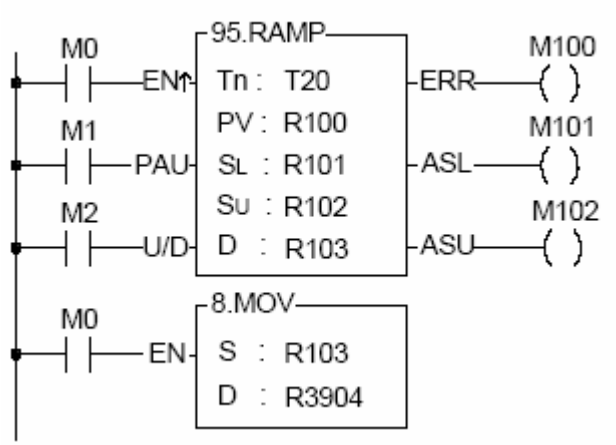
وقتی مقدار D به S_L رسید، خروجی "ASL"=1 می شود.

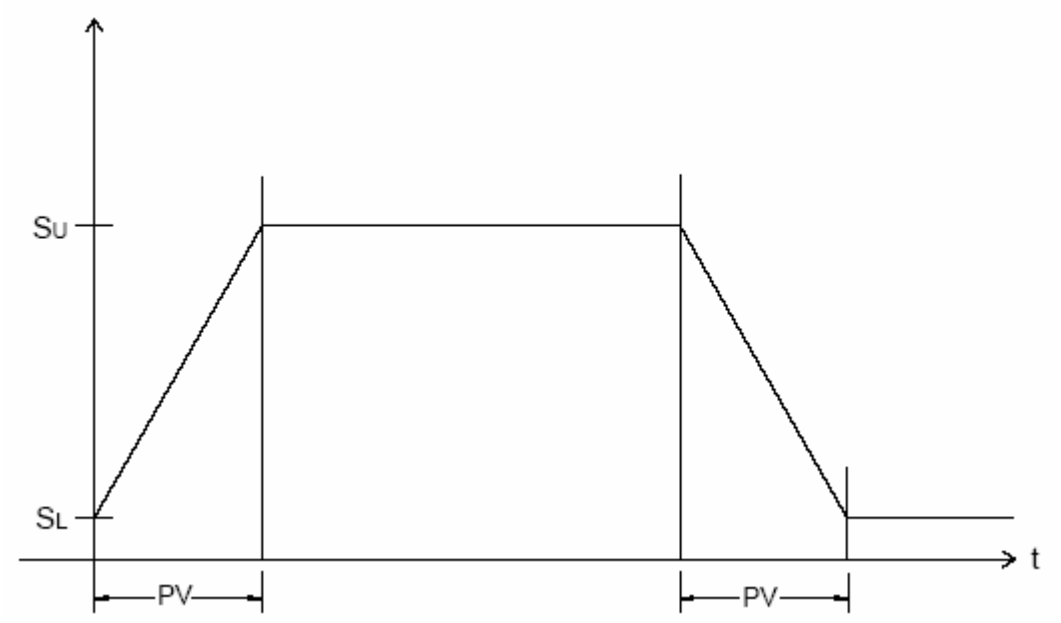
همزمان با تغییر "EN" از 0 به 1 ،جهت شیب "U/D" اندازه گرفته می شود.بعد از اینکه خروجی D شروع به شیب گرفتن کرد،تغییر "U/D" فایده ای ندارد.

اگر نیاز به توقف عمل شیب دادن بود ،باید "PAU"=1 بشود.وقتی "PAU"=0 کنید و هنوز شیب دادن به پایان نرسیده باشد،عمل شیب دادن تا هنگامی که پایان یابد ادامه خواهد یافت.

مقدارSu باید بزرگتر از S_L داده شود،در غیر این صورت تابع اجرا نشده و خروجی"ERR" 1، خواهد شد.

مثال:


95.RAMP
M0
EN
Tn : T20
ERR
M100
M1
PAU
PV : R100
SL : R101
ASL
M101
M2
U/D
SU : R102
D : R103
ASU
M102
8.MOV
M0
EN
S : R103
D : R3904


SU
SL
t
PV
PV

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

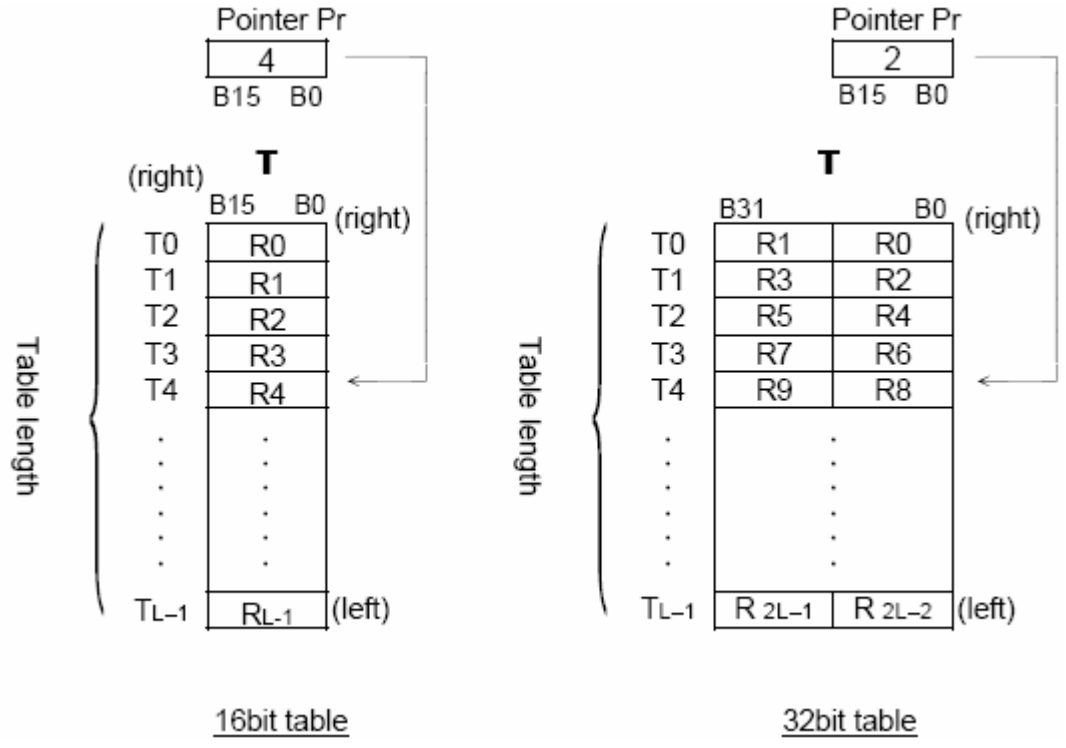
یک جدول شامل دو یا چند رجیستر متوالی است (16 یا 32 بیتی)

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

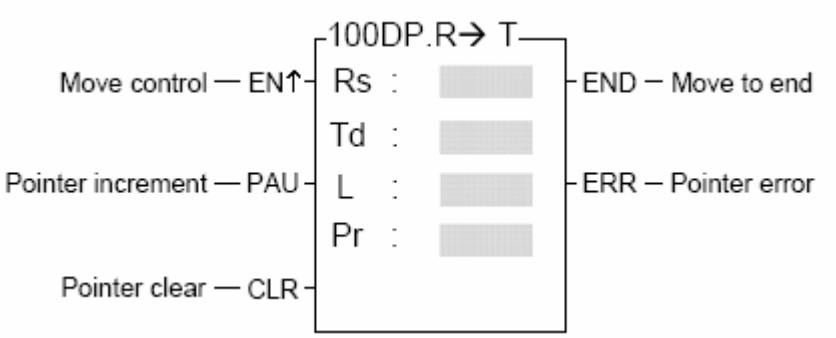
اجرای توابع جدولی بیشتر برای پردازش داده ها استفاده می شود،مانند انتقال ، کپی ، مقایسه ، جستجو و ...بین جدول ها و رجیستر ها یا بین جدول ها.

در میان توابع جدولی ،بیشتر توابع از یک Pointer استفاده می کنند تا مشخص کنند کدام رجیستر جدول ،هدف اجرا باشد.Pointer برای هر دو جدول 16 و 32 بیتی ، یک رجیستر 16 بیتی است.رنج موثر Pointer ،0~L-1 است.

در هنگام عملیات جدولی ،عملیاتی مانند شیفت به راست یا چپ،چرخش به راست یا چپ به این ترتیب مشخص می شود که جهت به سمت رجیستر بالاتر باشد،چپ گفته می شود و اگر جهت به سمت رجیستر پایین تر باشد راست گفته می شود.



100.REGISTER TO TABLE MOVE



هرگاه "EN" از 0 به 1 تغییر کند:

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

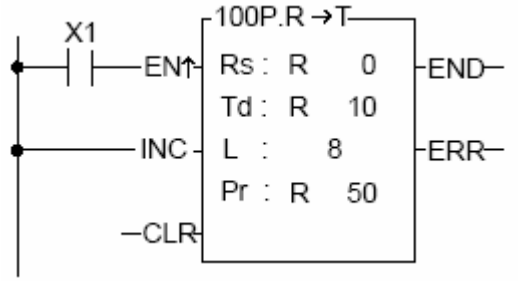
در Td رجیستر شروع جدول قرار می گیرد به طول (L).

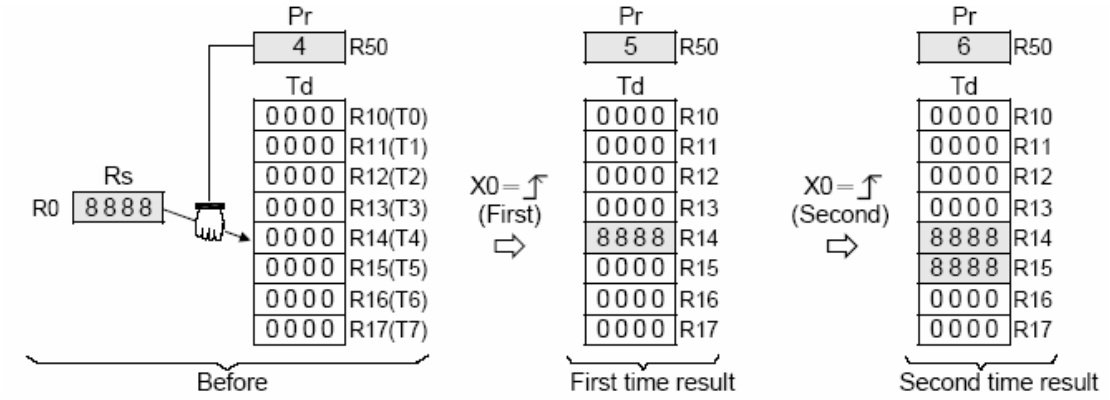
این تابع قبل از اجرا ، ابتدا سیگنال ورودی "CLR" را چک می کند.اگر"CLR"=1 مقدار Pointer (Pr) را چک می کند.اگر مقدار Pr به L-1 رسیده بود(یعنی به آخرین رجیستر جدول اشاره می کرد) خروجی "END" ، 1 می شود و اجرای این تابع پایان می یابد.

اگر مقدار Pr کمتر از L-1 باشد،سپس مجدداً "INC" را چک می کند.اگر"INC" ،1 باشد، مقدار Pr افزایش می یابد."CLR" می تواند بدون اینکه تحت تاثیر ورودی های دیگر باشد،مستقلاً اجرا شود.

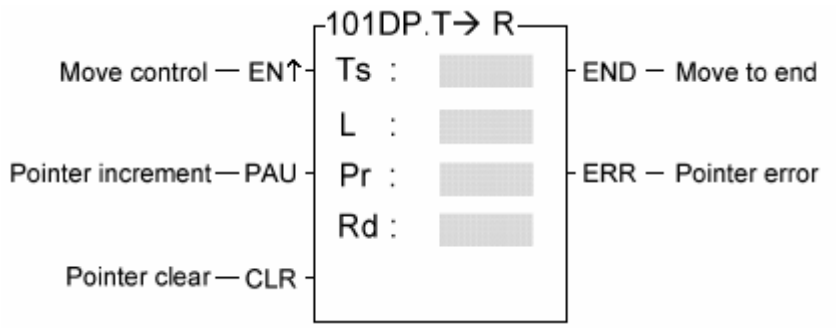
رنج موثر Pointer :0~L-1 است ، فراتر از این رنج "ERR" ،1 خواهد شد و این تابع اجرا نمی شود.

مثال:


X1
100P.R →T
EN
Rs : R 0
END
Td : R 10
INC
L : 8
ERR
Pr : R 50
CLR

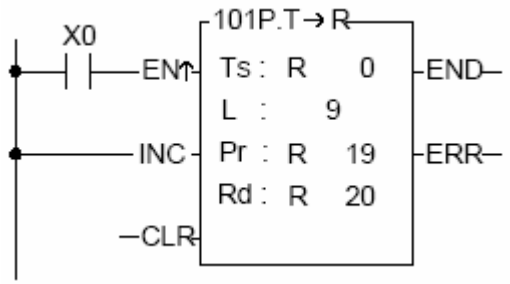

Pr
4
R50
Td
0000 R10(T0)
0000 R11(T1)
0000 R12(T2)
0000 R13(T3)
0000 R14(T4)
0000 R15(T5)
0000 R16(T6)
0000 R17(T7)
Rs
R0 8888
Before
X0 = ↑
(First)
Pr
5
R50
Td
0000 R10
0000 R11
0000 R12
0000 R13
8888 R14
0000 R15
0000 R16
0000 R17
First time result
X0 = ↑
(Second)
Pr
6
R50
Td
0000 R10
0000 R11
0000 R12
0000 R13
8888 R14
8888 R15
0000 R16
0000 R17
Second time result

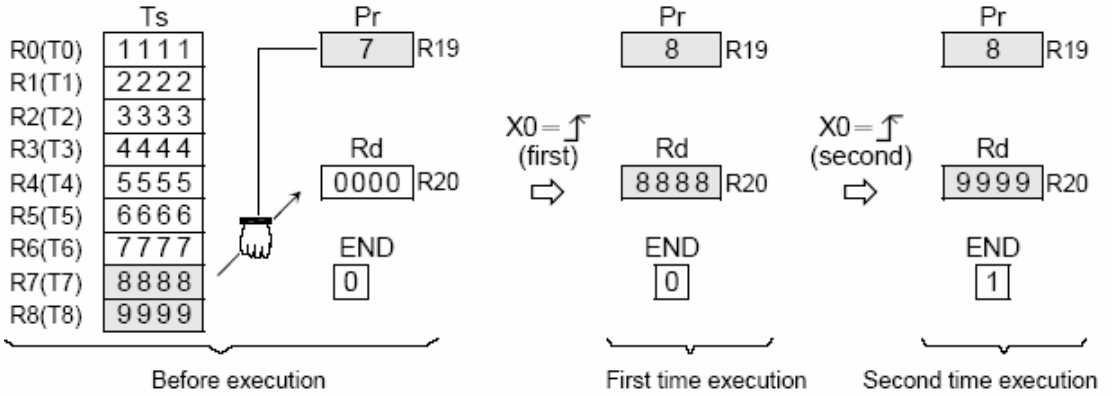
101.TABLE TO REGISTER MOVE



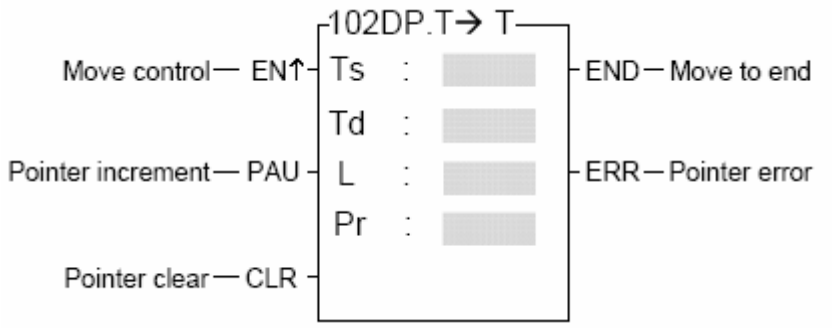
این تابع بر عکس تابع قبل عمل کرده و محتویات یک رجیستر از جدول مورد نظر را به رجیستر مقصد منتقل می کند.

مثال:





102.TABLE TO TABLE MOVE



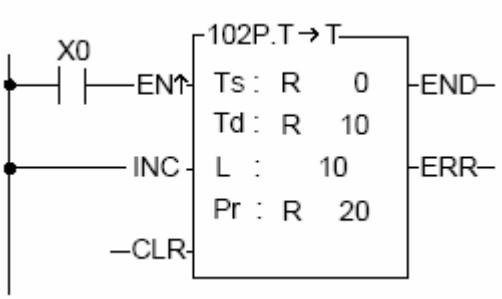
در Ts، رجیستر شروع جدول مبدا قرار می گیرد و

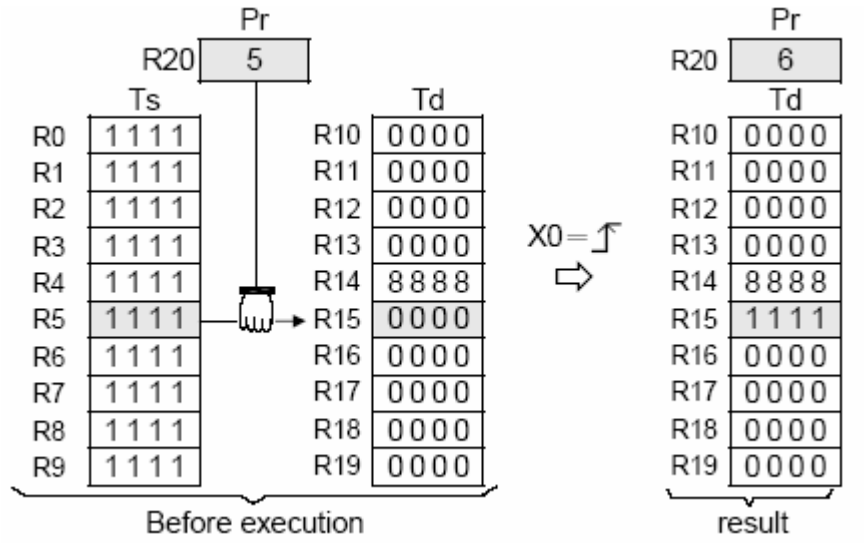
در Td، ، رجیستر شروع جدول مقصد قرار می گیرد.

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

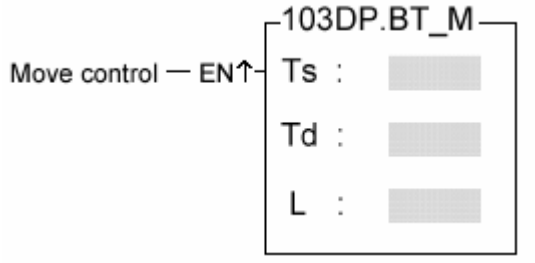
هرگاه "EN" از 0 به 1 تغییر کند:

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





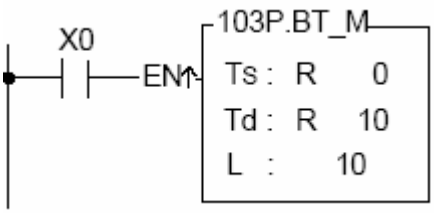
103.BLOCK TABLE MOVE

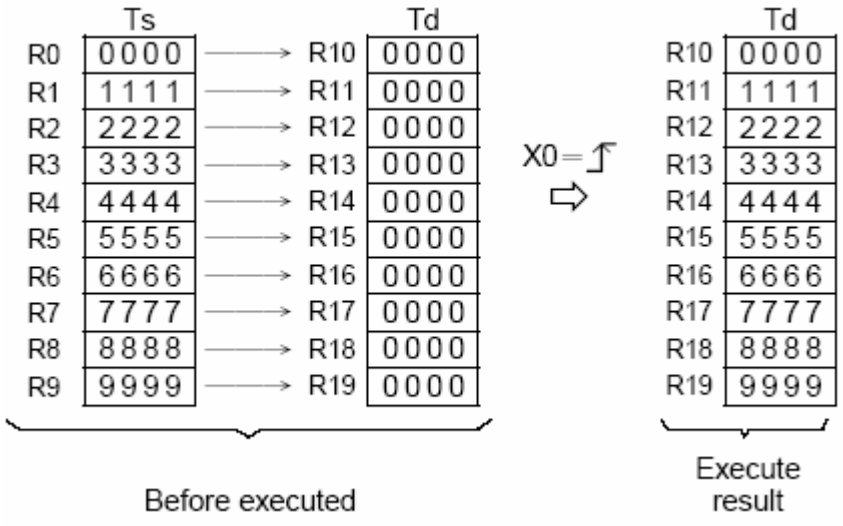


هرگاه "EN" از 0 به 1 تغییر کند:

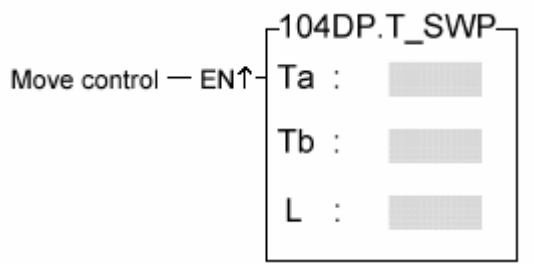
محتویات رجیسترهای جدول مبدا به داخل رجیسترهای معادلشان در جدول مقصد،کپی می شوند.

مثال:





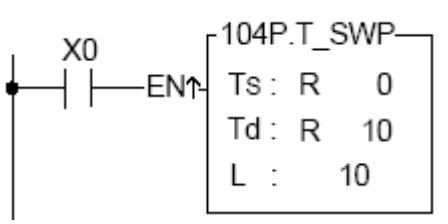
104.BLOCK TABLE SWAP

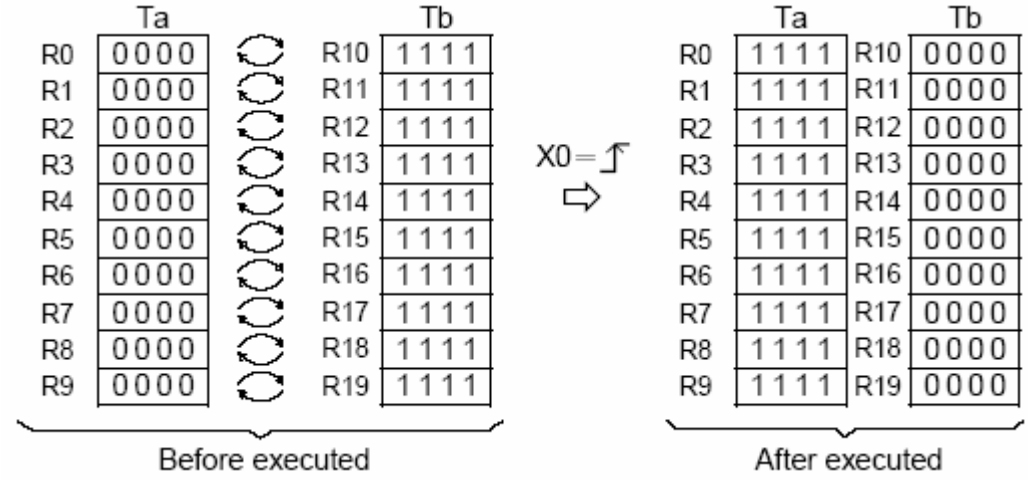


هرگاه "EN" از 0 به 1 تغییر کند:

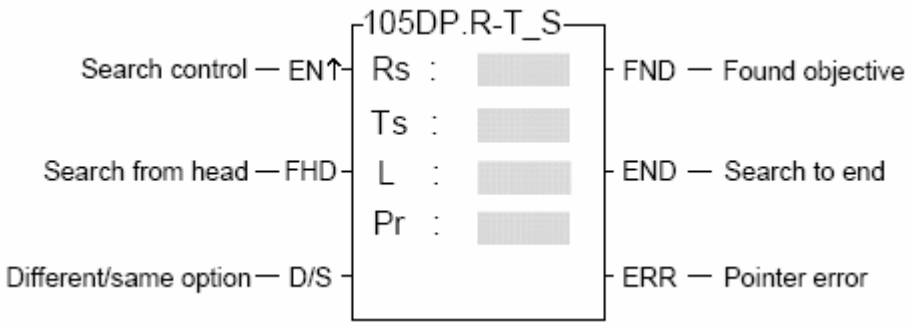
محتویات رجیسترهای جدول a با محتویات رجیسترهای معادلشان در جدول b ،جا به جا می شوند.

مثال:





105.REGISTER TO TABLE SEARCH



هرگاه "EN" از 0 به 1 تغییر کند:

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

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

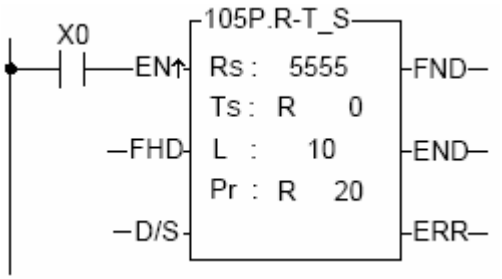
وقتی "D/S"=1 ،جستجو یرای یافتن اولین رجیستری که محتوای آن با Rs متفاوت باشد صورت می گیرد.

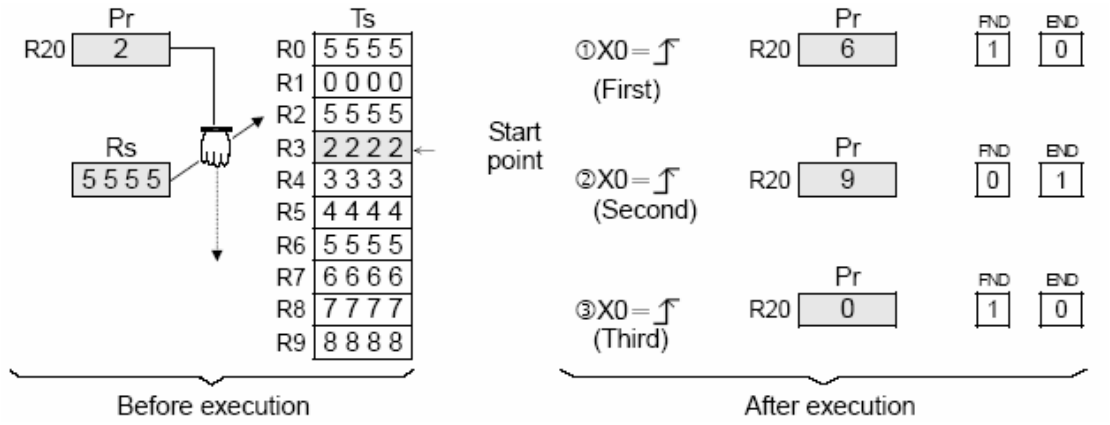
وقتی "D/S"=0 ،جستجو یرای یافتن اولین رجیستری که محتوای آن با Rs یکسان باشد صورت می گیرد.

بعد از یافتن اطلاعات مورد نظر ،جستجو متوقف می شود و خروجی "FND" ، 1 می شود.

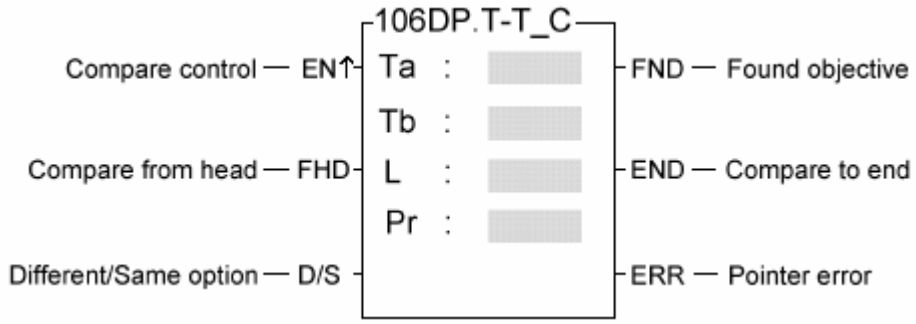
وقتی Pr به L-1 رسید چه محتوای مورد نظر را یافته باشد چه نیافته باشد ،"END" فعال شده و جستجو متوقف می شود.

مثال:

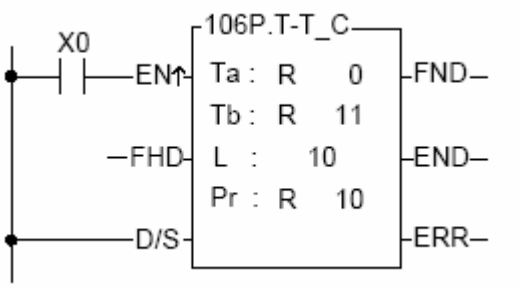

X0
105P.R-T_S
EN
Rs : 5555
Ts : R 0
FHD
L : 10
Pr : R 20
D/S
FND
END
ERR

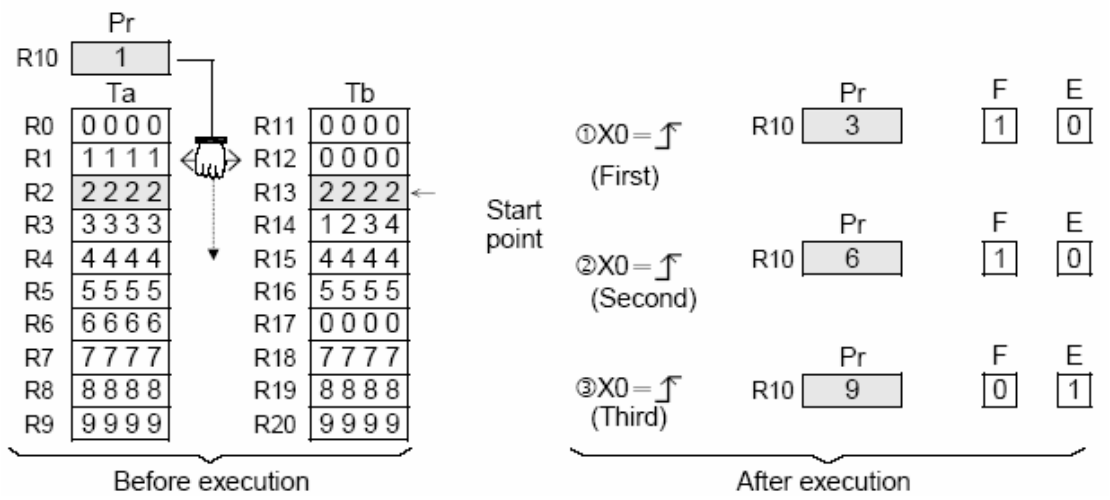

Pr
R20 2
Rs
5555
Ts
R0 5555
R1 0000
R2 5555
R3 2222
R4 3333
R5 4444
R6 5555
R7 6666
R8 7777
R9 8888
Before execution
Start point
①X0 (First)
②X0 (Second)
③X0 (Third)
Pr
R20 6
FND 1
END 0
Pr
R20 9
FND 0
END 1
Pr
R20 0
FND 1
END 0
After execution

106.TABLE TO TABLE COMPARE

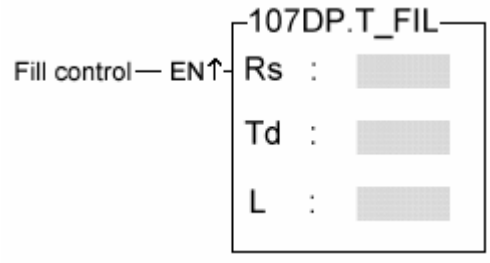


این تابع مانند تابع قبل است با این تفاوت که محتوای رجیستر های معادل از دو جدول Ta و Tb با هم مقایسه می شوند.





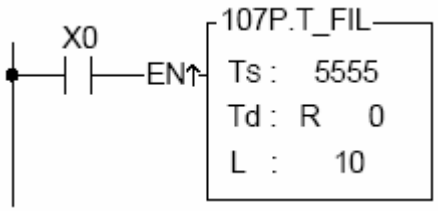
107.TABLE FILL

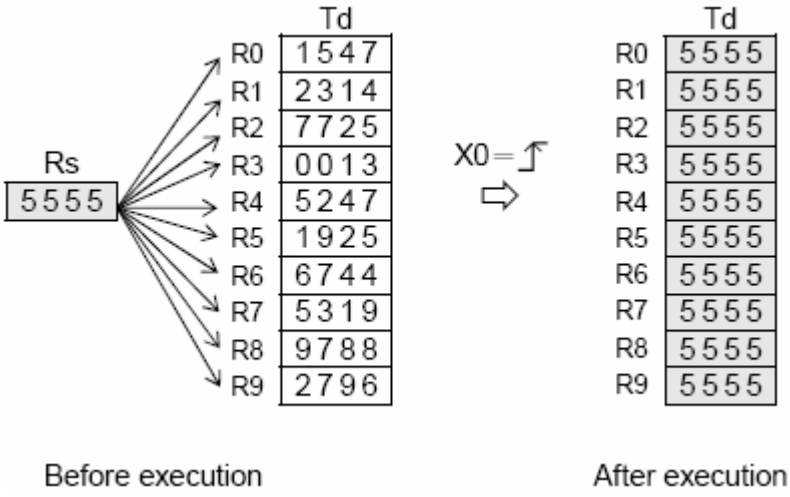


هرگاه "EN" از 0 به 1 تغییر کند:

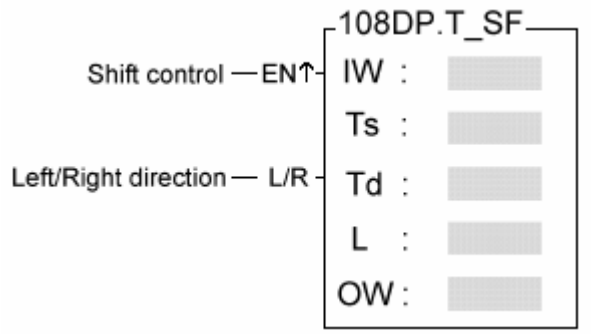
محتوای رجیستر Rs، تمتم رجیسترهای جدول Td را پر می کند.

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





108.TABLE SHIFT



هرگاه "EN" از 0 به 1 تغییر کند:

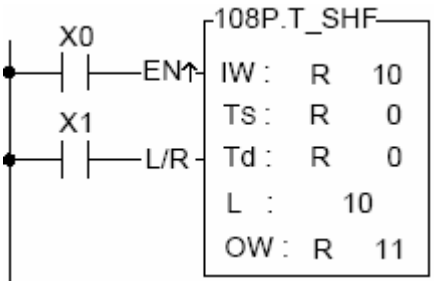
محتوای رجیسترهای جدول Ts ، یک رجیستر به سمت چپ یا راست شیفت پیدا می کند.وقتی "L/R"=1 شیفت به چپ صورت می گیرد و وقتی "L/R"=0 شیفت به راست صورت می گیرد.

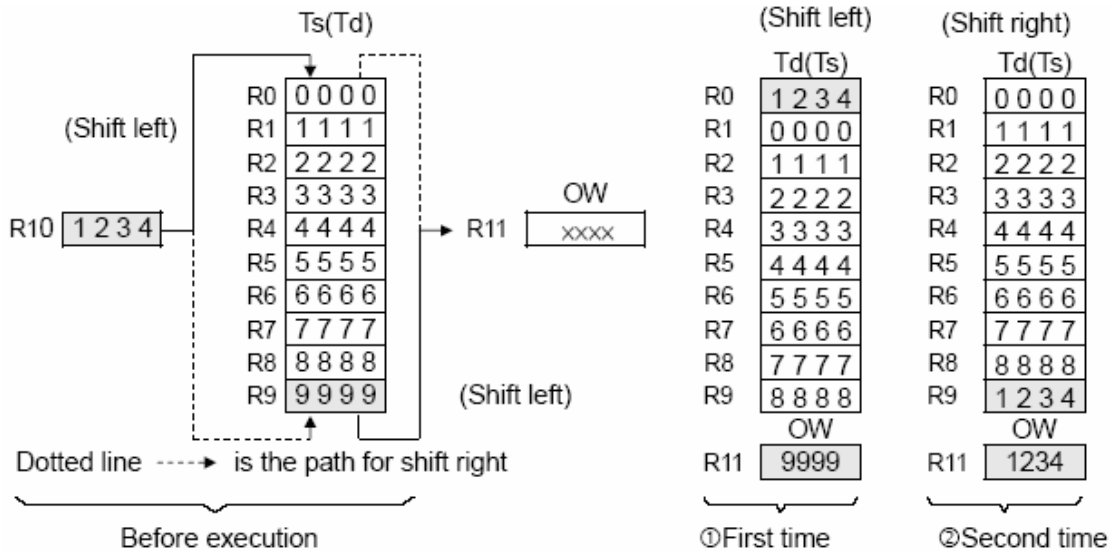
فضای خالی ایجاد شده بعد از اعمال شیفت ،توسط رجیستر یا عدد ثابت مشخص شده در IW ، پر می شود.

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

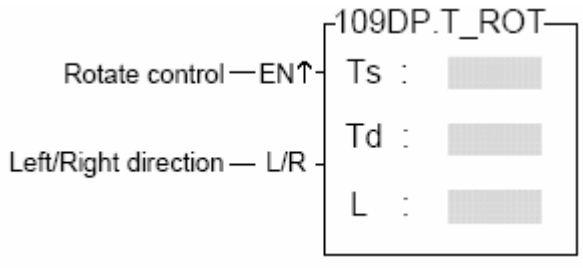
جدول شیفت داده شده می تواند در جدول Td یا خود Ts ذخیره شود.

مثال:


X0
EN
X1
L/R
108P.T_SHF
IW : R 10
Ts : R 0
Td : R 0
L : 10
OW : R 11


Ts(Td)
(Shift left)
R10 1234
R0 0000
R1 1111
R2 2222
R3 3333
R4 4444
R5 5555
R6 6666
R7 7777
R8 8888
R9 9999
R11 xxxx
OW
(Shift left)
Dotted line ----→ is the path for shift right
Before execution
(Shift left)
Td(Ts)
R0 1234
R1 0000
R2 1111
R3 2222
R4 3333
R5 4444
R6 5555
R7 6666
R8 7777
R9 8888
OW
R11 9999
①First time
(Shift right)
Td(Ts)
R0 0000
R1 1111
R2 2222
R3 3333
R4 4444
R5 5555
R6 6666
R7 7777
R8 8888
R9 1234
OW
R11 1234
②Second time

109.TABLE ROTATE



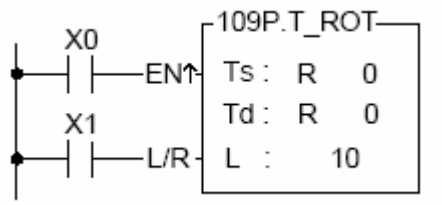
هرگاه "EN" از 0 به 1 تغییر کند:

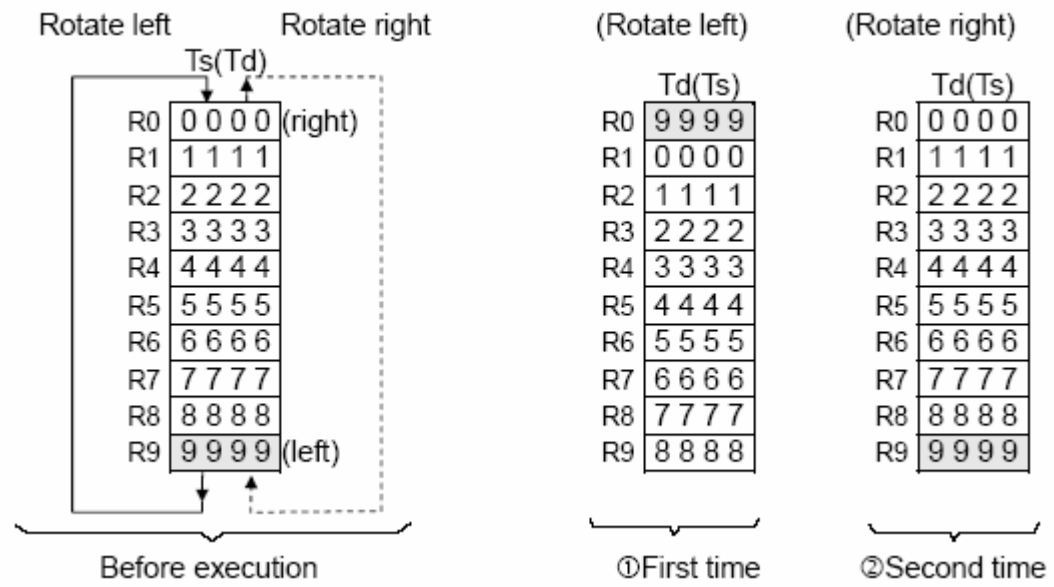
محتوای رجیسترهای جدول Ts ، یک رجیستر به سمت چپ یا راست می چرخند و نتیجه در Td ذخیره می شود.

وقتی "L/R"=1 ،چرخش به چپ صورت می گیرد

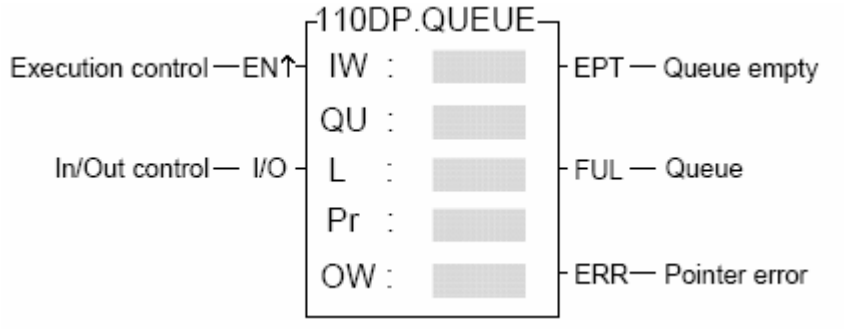
وقتی "L/R"=0 ،چرخش به راست صورت می گیرد.

مثال:





110.QUENE



QUENE نیز نوعی جدول است،با این تفاوت که شماره رجیسترهای جدول معمولی از 0~L-1 است اما شماره جدول های QUENE، 1~L است و Pr=0 برای نمایش خالی بودن QUENE استفاده می شود.

در QUENE :

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

یک QUENE از L رجیستر 16 یا 32 بیتی تشکیل شده که از رجیستر مشخص شده در QU شروع می شوند.

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

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

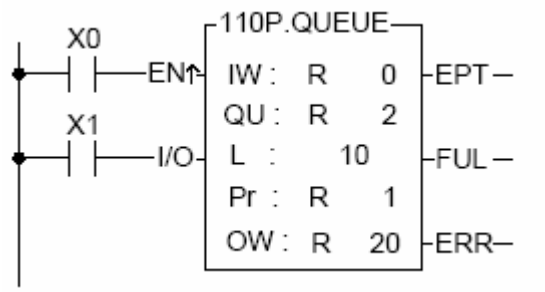
محتوای IW ،همیشه به داخل اولین رجیستر QUENE، push می شود و پس از هر push یکی به مقدارpointer(Pr) اضافه می شود و همیشه هنگام pop کردن ،قدیمی ترین محتوا از داخل QUENE به OW ، pop می شود و یکی از مقدار Pr کم می شود.

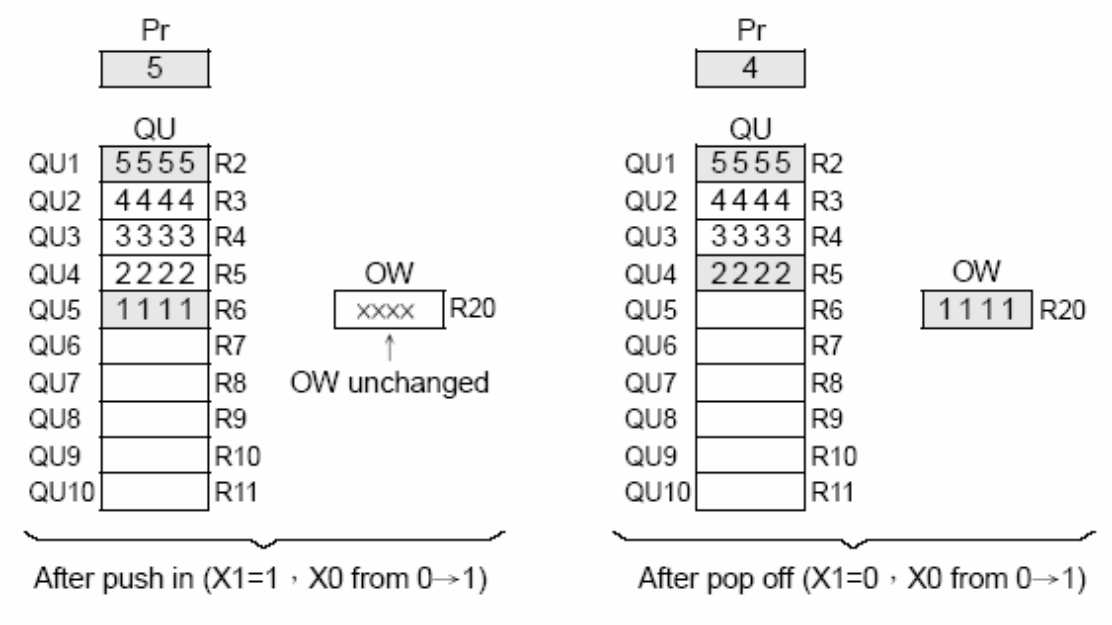
اگر QUENE خالی از داده باشد (Pr=0 باشد)، "EPT"=1 خواهد شد.

اگر QUENE کاملاً پر باشد (Pr به رجیستر L از QUENE اشاره کند) ،"FUL"=1 خواهد شد.

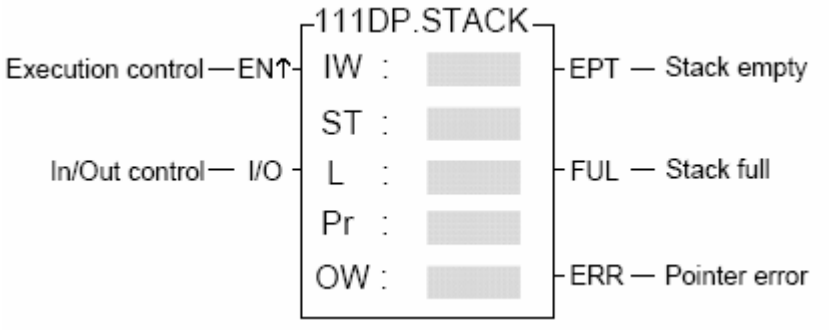
اگر مقدار Pr فراتر از رنج 0~L داده شود، "ERR"=1 خواهد شد.

مثال:


X0
X1
EN
I/O
110P.QUEUE
IW : R 0
QU : R 2
L : 10
Pr : R 1
OW : R 20
EPT
FUL
ERR


Pr
5
QU
QU1 5555 R2
QU2 4444 R3
QU3 3333 R4
QU4 2222 R5
QU5 1111 R6
QU6 R7
QU7 R8
QU8 R9
QU9 R10
QU10 R11
OW
×××× R20
OW unchanged
After push in (X1=1，X0 from 0→1)
Pr
4
QU
QU1 5555 R2
QU2 4444 R3
QU3 3333 R4
QU4 2222 R5
QU5 R6
QU6 R7
QU7 R8
QU8 R9
QU9 R10
QU10 R11
OW
1111 R20
After pop off (X1=0，X0 from 0→1)

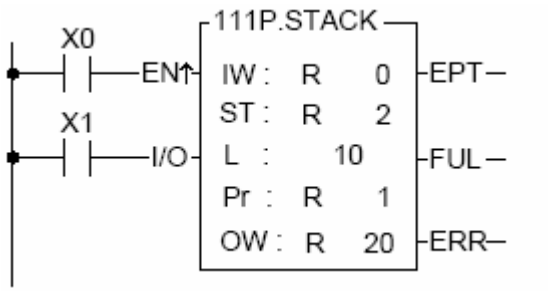
111.STACK

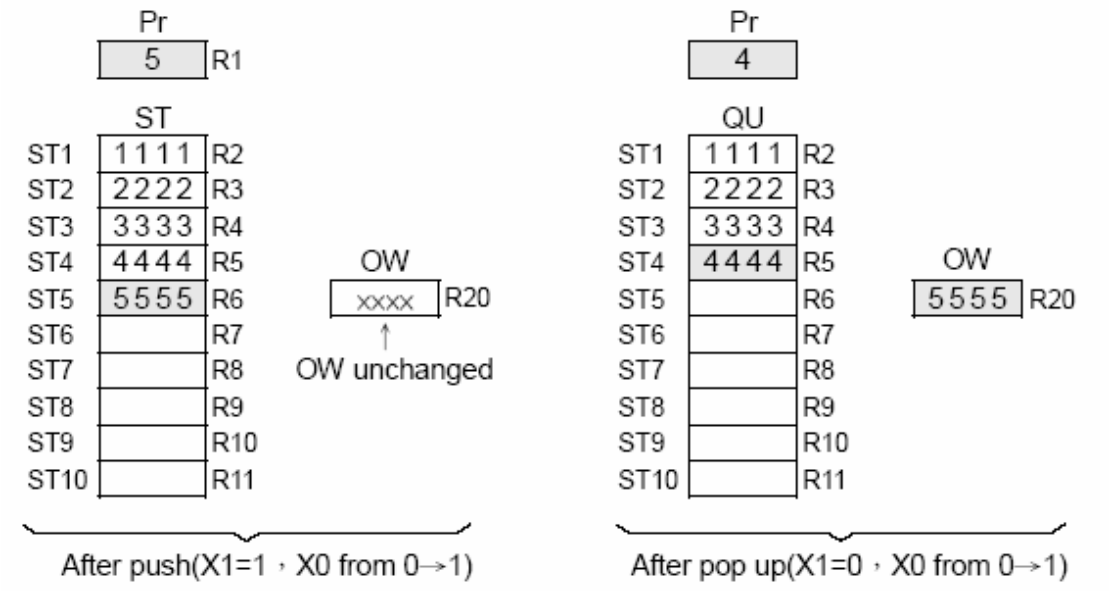


STACK نیزجدولی مانند QUEUE است با یک تفاوت.

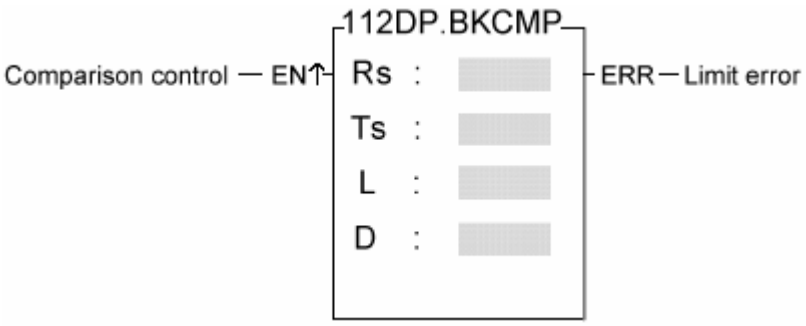
در QUEUE اولین داده ای که push می شود،اولین داده ای خواهد بود که pop می شود

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





112.DRUM



هرگاه "EN" از 0 به 1 تغییر کند:

مقایسه بین محتوای Rs و یک جفت از رجیسترهای جدول که از Ts شروع می شوند ، صورت می گیرد.(به عنوان مثال یک جفت شامل Ts0 که به آن حد پایین و Ts1 که به آن حد بالا می گویند،می شود.) بعد ازمقایسه بین Rs و اولین جفت ، نتیجه در D0 ذخیره می شود،سپس مقایسه بین Rs و دومین جفت صورت می گیرد و نتیجه در D1 ذخیره می شود و...

اگر مقدار Rs د رمحدوده حد بالا و حد پایین یک جفت باشد ،بیت D متناظر با آن جفت 1، می شود در غیر این صورت 0 خواهد بود.

	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 جفت ها را مشخص می کند.

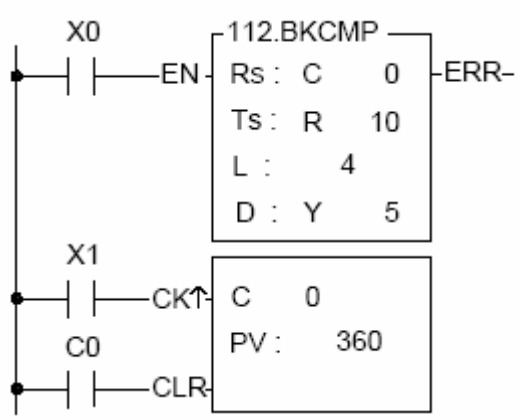
مقایسه تا زمانی ادامه می یابد کهRs با تمام جفت ها مقایسه شود.

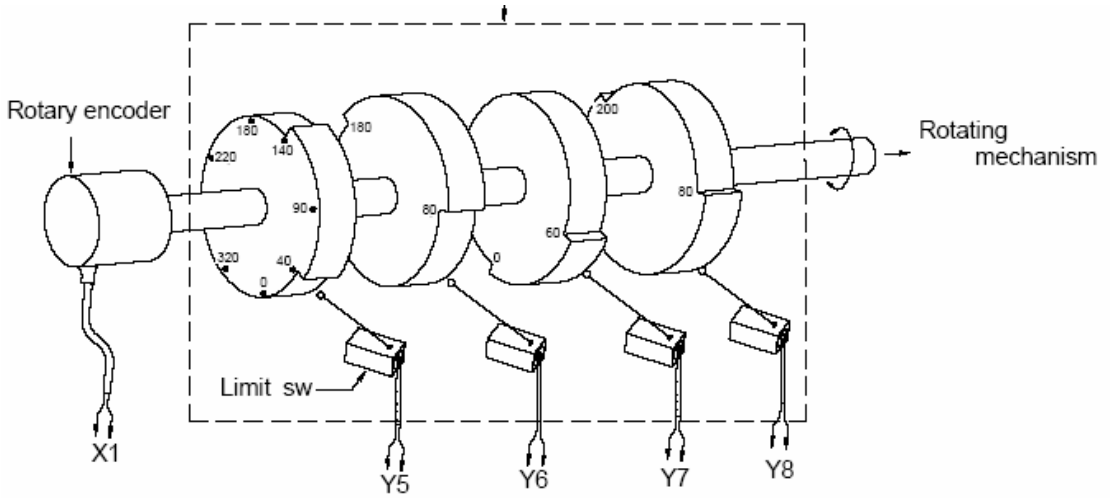
وقتی رجیستر M1975=0 است و در جفتی ،حد بالا کوچک تر از حد پایین باشد، "ERR" فعال می شود و نتیجه مقایسه برای آن جفت ،0 خواهد بود.

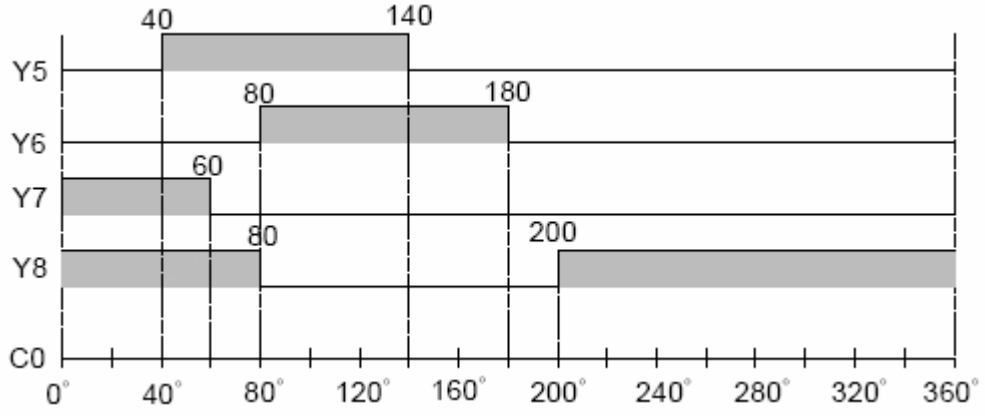
وقتی رجیستر M1975=1 است ،محدودیتی برای بزرگتر یا کوچکتر بودن حد بالا نسبت به حد پایین وجود ندارد.این حالت می تواند برای سوئیچ DRUM الکترونیکی چرخنده 360^o ای ،کاربرد داشته باشد.

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

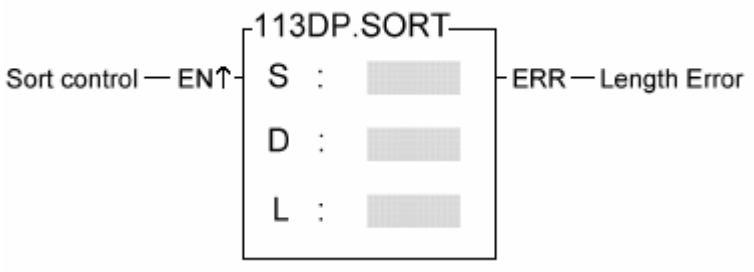
مثال:







113.DATA SORTING



هرگاه "EN" از 0 به 1 تغییر کند:

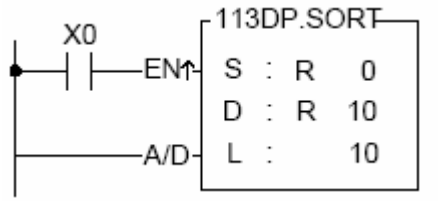
این تابع رجیستر ها به تعداد L را که از S شروع می شوند ،به صورت افزایشی یا کاهشی مرتب می کند و نتیجه را در D ذخیره می کند.

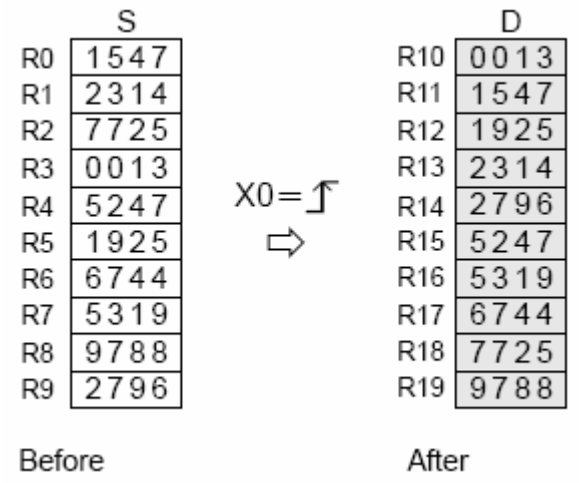
اگر "A/D"=1 رجیسترها به صورت افزایشی مرتب می شوند و

اگر "A/D"=0 رجیسترها به صورت کاهشی مرتب می شوند.

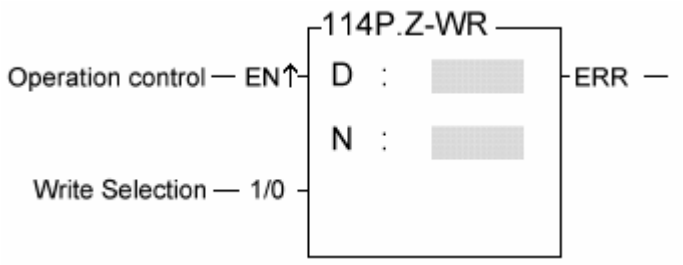
L باید در رنج 2~127 باشد در غیر این صورت "ERR"=1.

مثال:





114.ZONE WRITE



هرگاه "EN" از 0 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 1="1/0" باشد،set می کند 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 می گویند و با (L) نمایش می دهند.

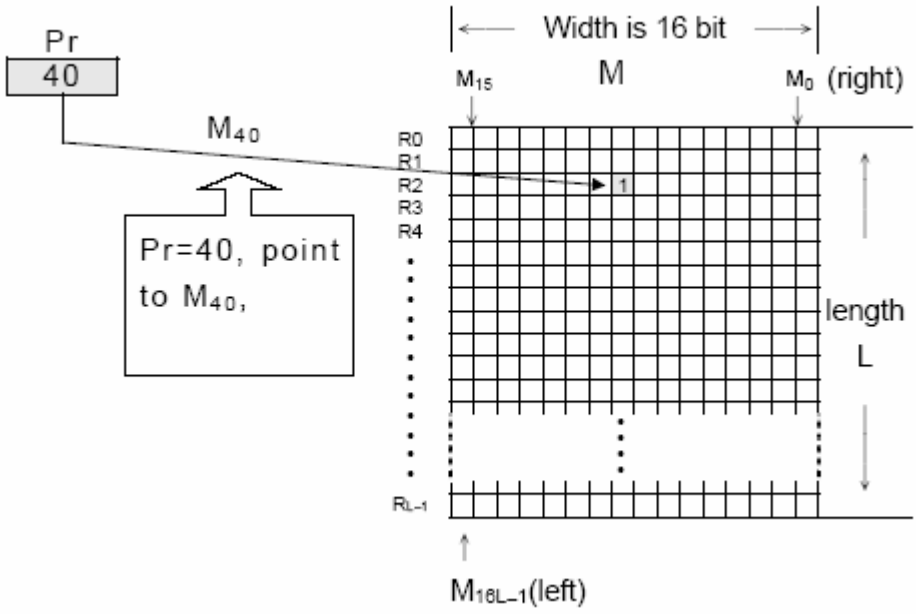
پس یک ماتریس 16×L بیت (آرایه) دارد و واحد اصلی اجرای این توابع ،بیت می باشد.

توابع ماتریسی عمدتاً برای پردازش گسسته صورت می گیرد مانند انتقال ،کپی، مقایسه ، جستجو و غیره یک آرایه به ماتریس یا ماتریس به ماتریس.

در میان این توابع ،بیشتر آنها به یک اشاره گر(pointer) 16 بیتی نیاز دارند تا به یک آرایه خاص در یک ماتریس اشاره کنند.

رنج موثر این pointer، 0~16L-1 است.

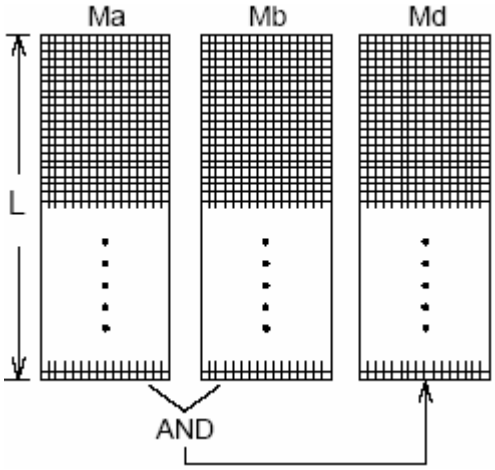
جهت اجرای دستوراتی مانند شیفت و چرخش،حرکت به سمت بیت کم ارزش را جهت راست می دانیم و حرکت به سمت بیت با ارزش تر را جهت چپ می دانیم.



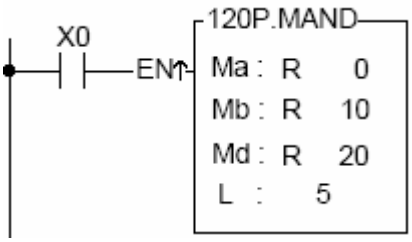
120.MATRIX AND

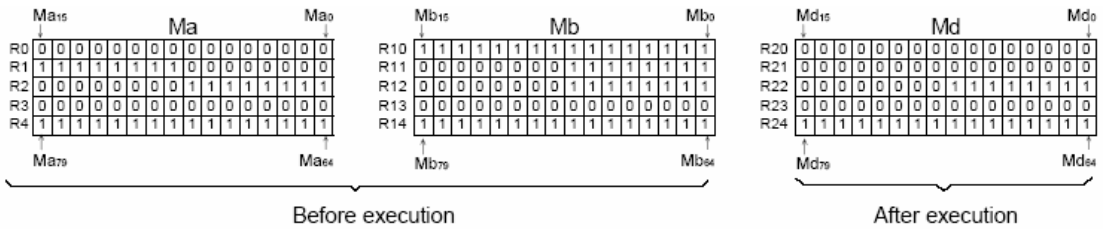
هرگاه "EN" از 0 به 1 تغییر کند:

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

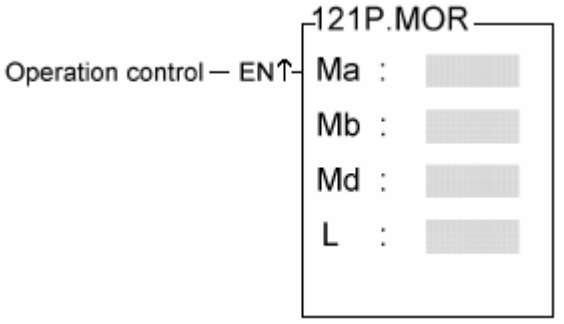


مثال:



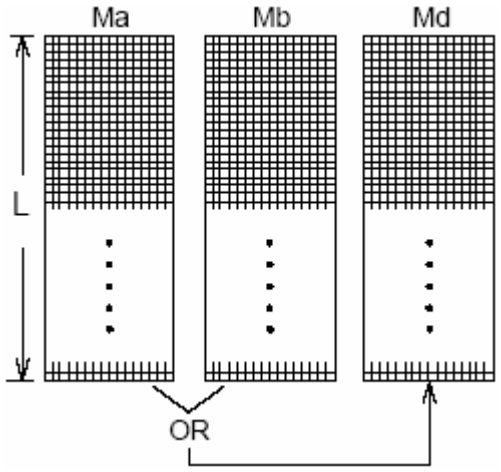


121.MATRIX OR

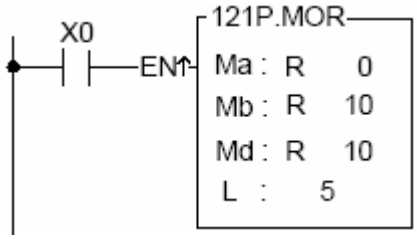


هرگاه "EN" از 0 به 1 تغییر کند:

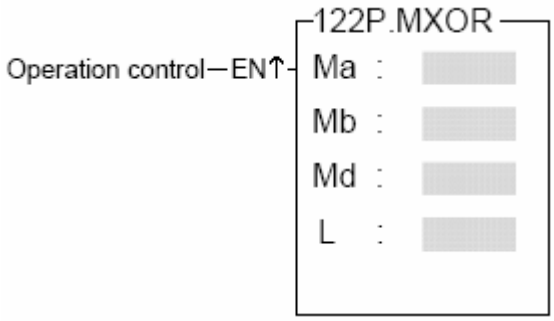
بیت های متناظر در ماتریس Ma و Mb را با یکدیگر OR کرده و نتیجه در Md ریخته می شود.



مثال:

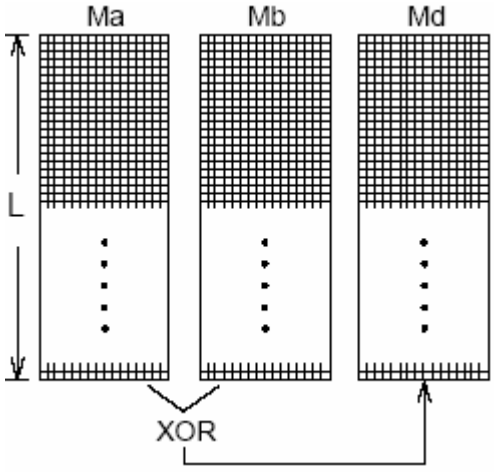


122.MATRIX XOR

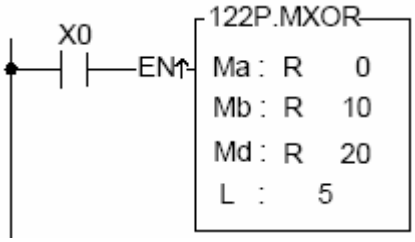


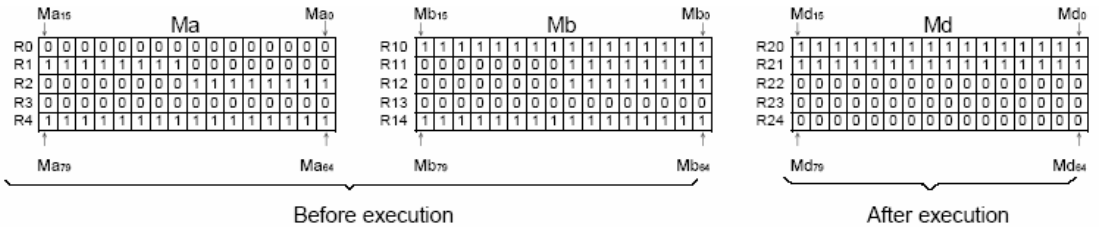
هرگاه "EN" از 0 به 1 تغییر کند:

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

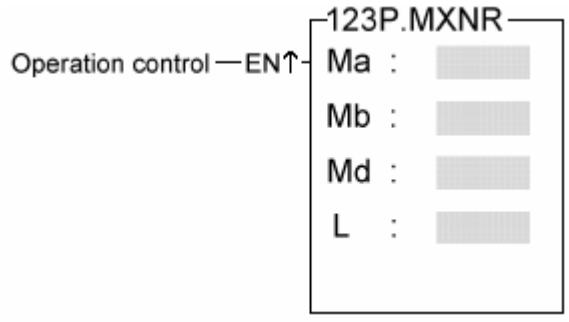


مثال:



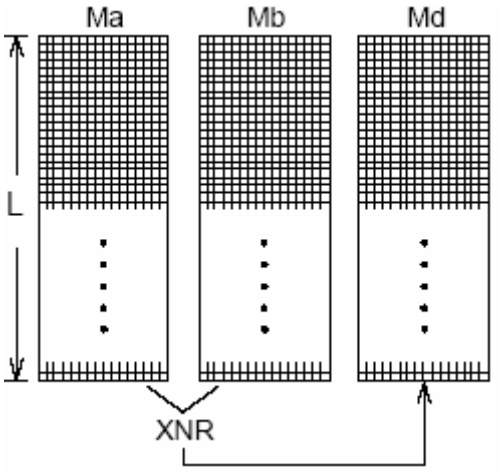


123.MATRIX XNOR

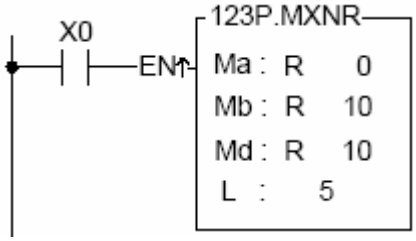


هرگاه "EN" از 0 به 1 تغییر کند:

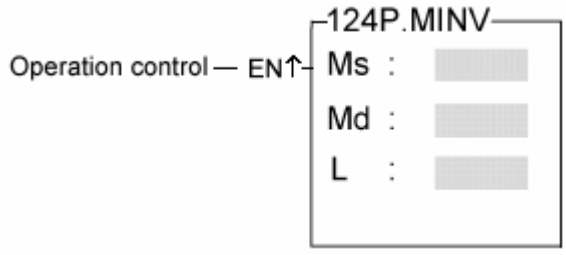
بیت های متناظر در ماتریس Ma و Mb را با یکدیگر XNOR کرده و نتیجه در Md ریخته می شود.



مثال:

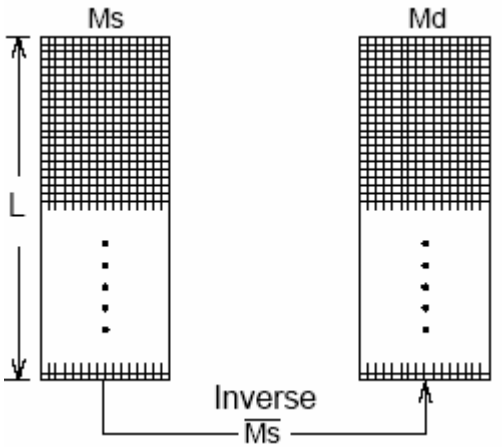


124.MATRIX INVERSE

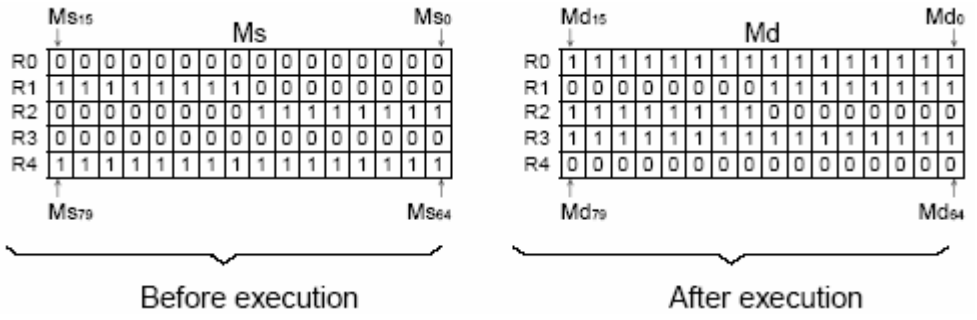


هرگاه "EN" از 0 به 1 تغییر کند:

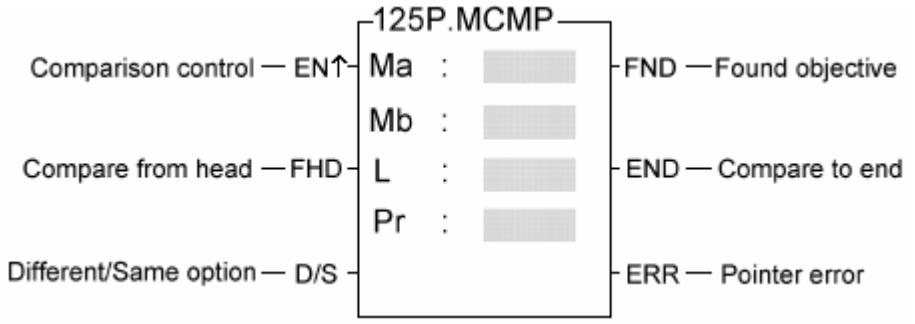
تمام بیت های ماتریس Ms معکوس می شوند(0 ها 1 شده و 1 ها 0 می شوند) و نتیجه در Md ذخیره می شود.



مثال:



125.MATRIX COMPARE



هرگاه "EN" از 0 به 1 تغییر کند:

وقتی "FHD"=1 یا مقدار Pr به 16L-1 رسیده است،مقایسه بین دو بیت متناظر از ماتریس های Ma و Mb ،از آغاز ماتریس ها (اولین بیت) ، شروع می شود.

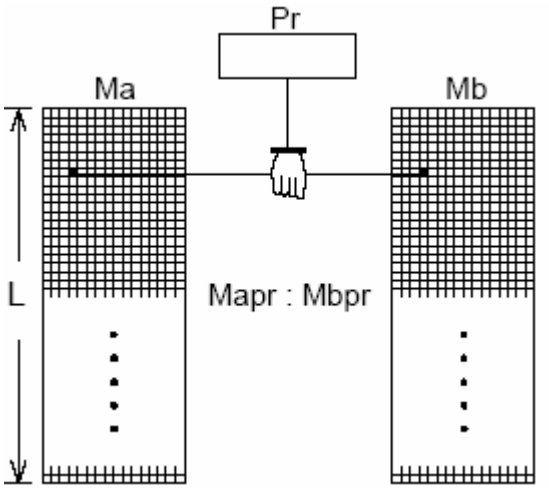
وقتی "FHD"=0 و مقدار Pr کمتر از 16L-1 باشد،مقایسه از اولین بیت ،بعد از جایی که Pr اشاره می کند،شروع می شود.

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

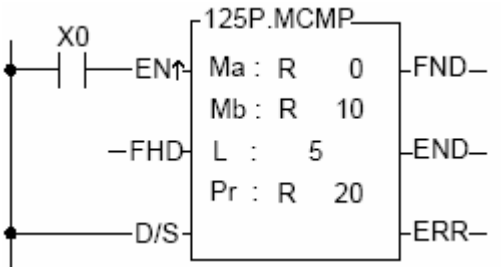
وقتی "D/S"=0 ،مقایسه برای یافتن اولین جفت بیت متناظری که محتوای آنها یکسان باشد صورت می گیرد.

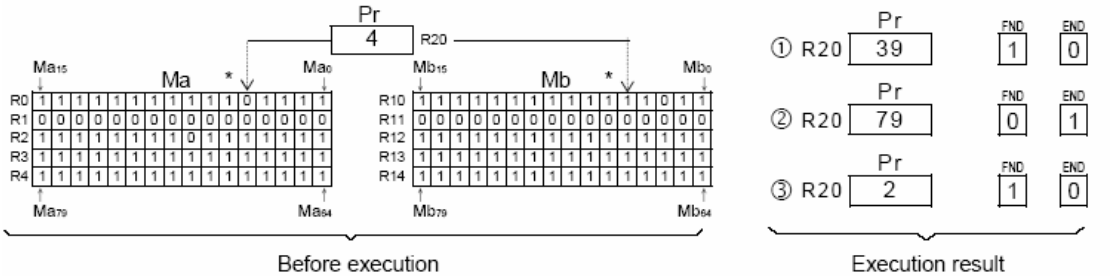
بعد از یافتن اطلاعات مورد نظر ،مقایسه متوقف می شود و خروجی "FND" ،1 می شود.

وقتی Pr به 16L-1 رسید،چه بیت های مورد نظر را یافته باشد چه نیافته باشد،"END" فعال شده و مقایسه متوقف می شود.

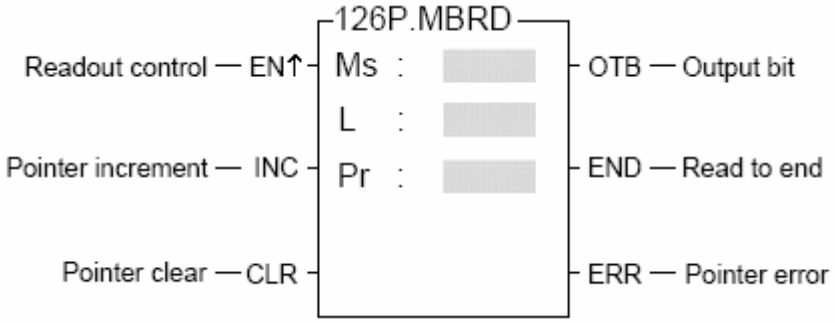


مثال:


X0
125P.MCMP
EN↑
Ma : R 0
Mb : R 10
FHD
L : 5
Pr : R 20
D/S
FND
END
ERR


Pr
4
R20
Ma15
Ma
Ma0
Mb15
Mb
Mb0
R0
R1
R2
R3
R4
R10
R11
R12
R13
R14
Ma79
Ma64
Mb79
Mb64
Before execution
① R20 Pr 39 FND 1 END 0
② R20 Pr 79 FND 0 END 1
③ R20 Pr 2 FND 1 END 0
Execution result

126.MATRIX BIT READ



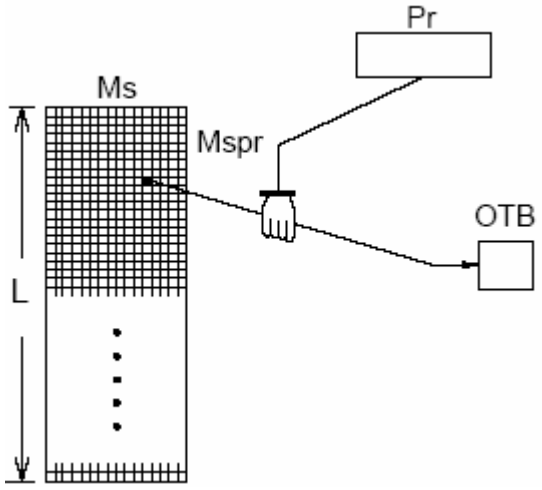
هرگاه "EN" از 0 به 1 تغییر کند:

مقدار بیتی که Pr به آن اشاره می کند،در خروجی "OTB" ظاهر می شود.

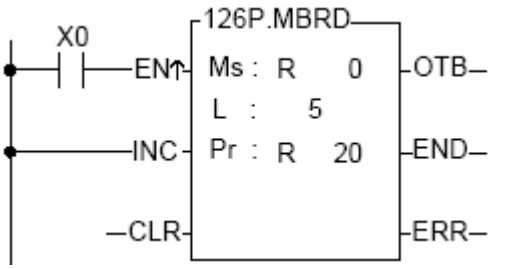
این تابع قبل از اجرا ،ابتدا "CLR" را چک می کند ،اگر "CLR"=1، مقدار Pr را 0 می کند سپس بیت را به خروجی "OTB" می فرستد.بعد از آن دوباره مقدار Pr را چک می کند .اگر مقدار Pr به 16L-1 رسیده بود(یعنی به آخرین بیت ماتریس اشاره می کرد) خروجی "END"، 1 می شود و اجرای این تابع پایان می یابد.

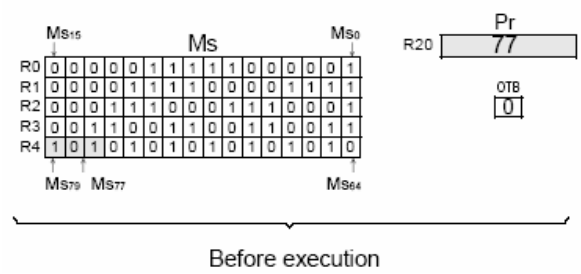
اگر مقدار Pr کمتر از 16L-1 باشد،مجدداً "INC" را چک می کند .اگر "INC"، 1 باشد ، مفدار Pr افزایش می یابد. "CLR" می تواند بدون ایت که تحت تاثیر ورودی های دیگر باشد، مستقلاً اجرا شود.

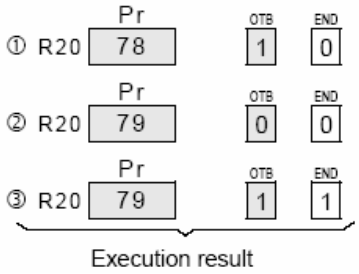
رنج موثر Pr، 16L-1~0 است،فراتر از این رنج "ERR"=1 خواهد شد و این تابع اجرا نمی شود.



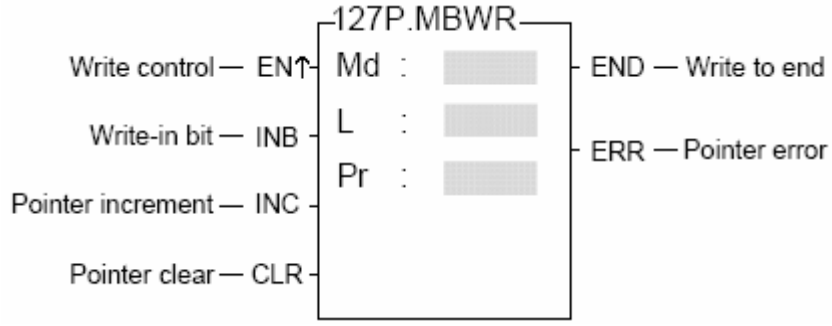
مثال:


X0
126P.MBRD
EN
Ms : R 0
L : 5
Pr : R 20
OTB
INC
END
CLR
ERR

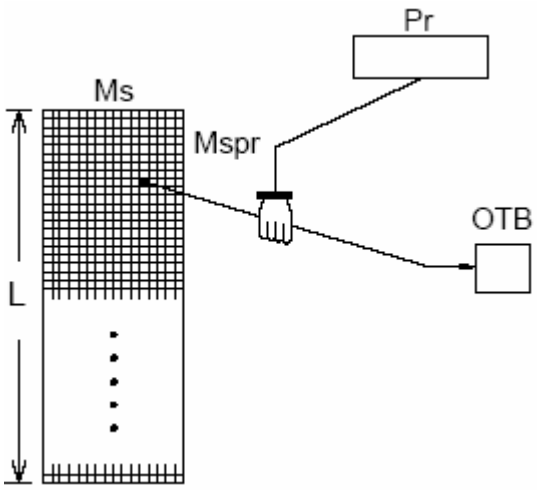

Ms15
Ms
Ms0
R0 0 0 0 0 0 1 1 1 1 1 0 0 0 0 0 1
R1 0 0 0 0 1 1 1 1 1 0 0 0 0 1 1 1 1
R2 0 0 0 1 1 1 0 0 0 1 1 1 0 0 0 1
R3 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1
R4 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0
Ms79 Ms77
Ms64
Pr
R20 77
OTB
0
Before execution


Pr
① R20 78
OTB 1
END 0
Pr
② R20 79
OTB 0
END 0
Pr
③ R20 79
OTB 1
END 1
Execution result

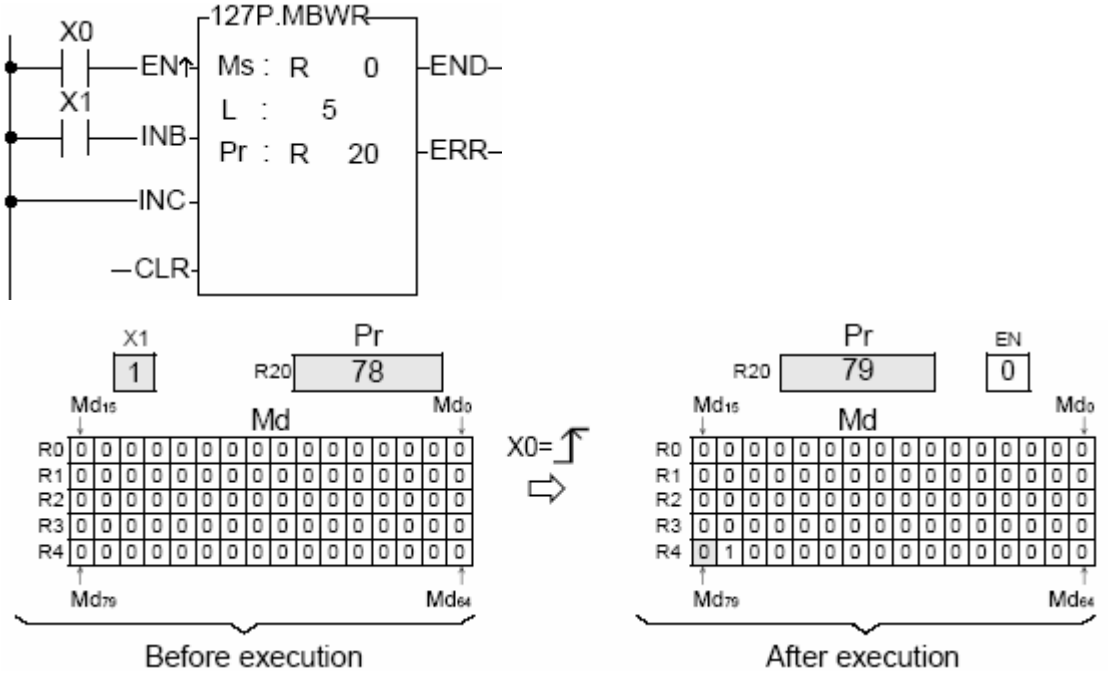
127.MATRIX BIT WRITE



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



مثال:



128.MATRIX BIT SHIFT

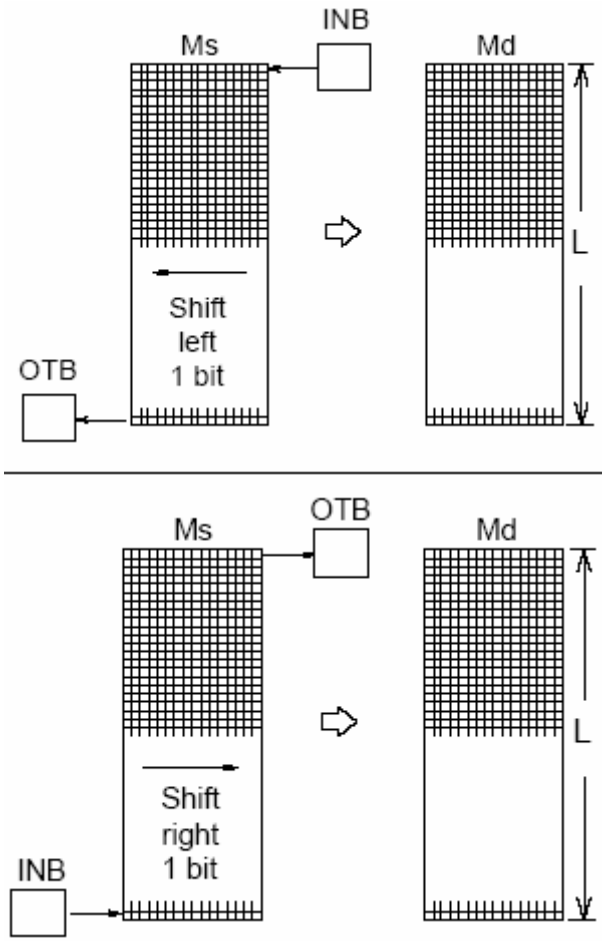
هرگاه "EN" از 0 به 1 تغییر کند:

بیت های ماتریس Ms، 1 بیت شیفت پیدا می کنند و نتیجه در Md ذخیره می شود.

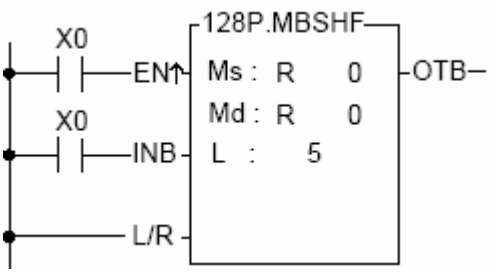
اگر "L/R"=1 باشد،شیفت به چپ صورت می گیرد و

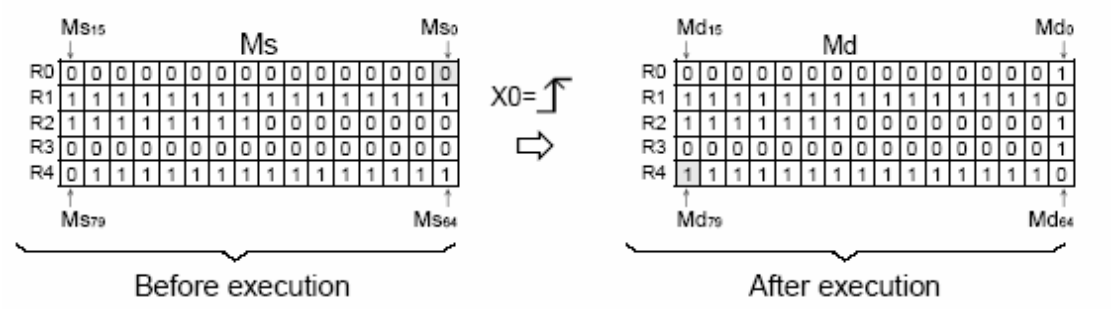
اگر "L/R"=0 باشد،شیفت به راست صورت می گیرد .

بیت خالی ایجاد شده بعد از اعمال شیفت ،توسط "INB" پر می شود،محتوای بیتی که پس از اعمال شیفت از جدول خارج می شود،به خروجی "OTB" منتقل می شود.

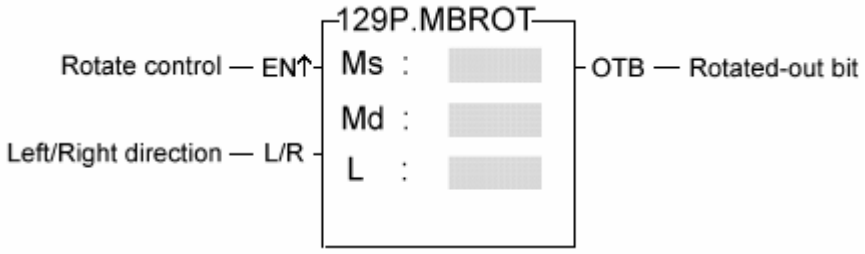


مثال:


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


Ms15
Ms
Ms0
R0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
R1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
R2 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0
R3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
R4 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
Ms79
Ms64
Before execution
X0=
Md15
Md
Md0
R0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
R1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0
R2 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 1
R3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
R4 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0
Md79
Md64
After execution

129.MATRIX BIT ROTATE



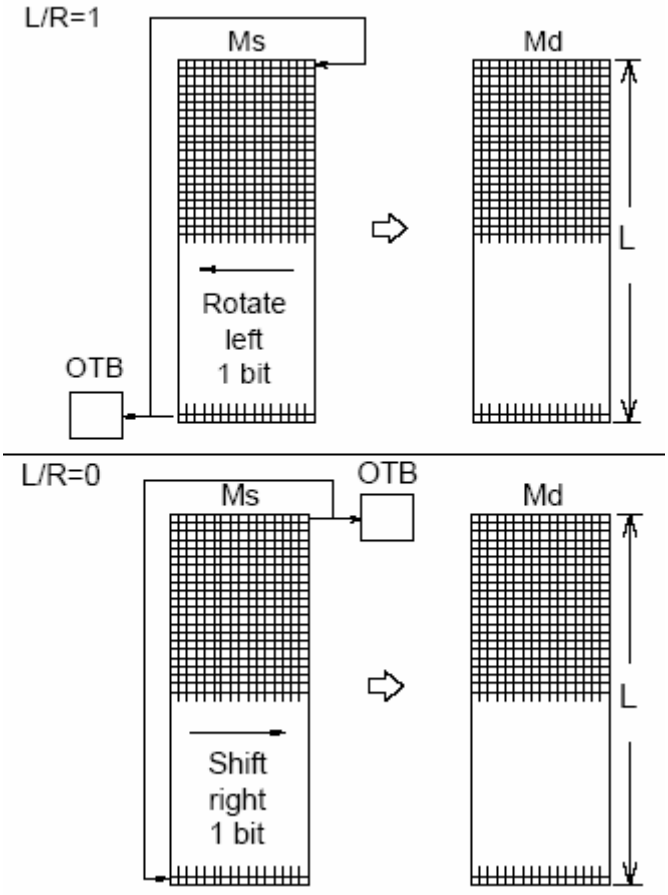
هرگاه "EN" از 0 به 1 تغییر کند:

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

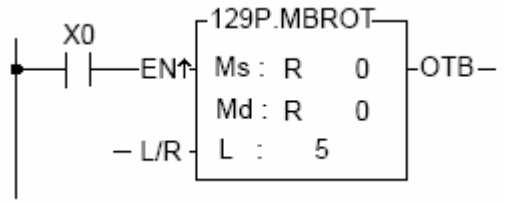
وقتی "L/R"=1، چرخش به چپ صورت می گیرد.

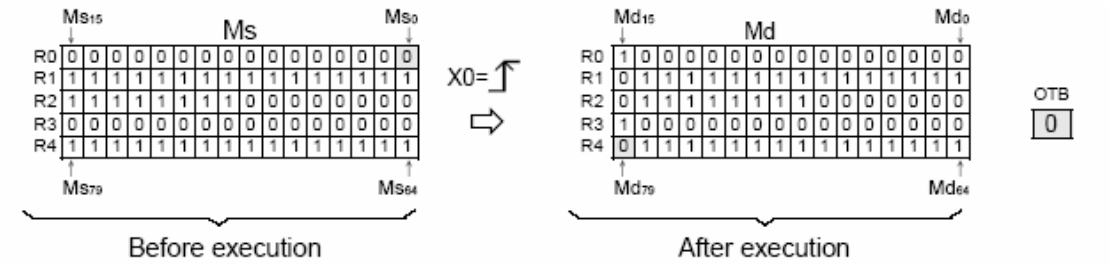
وقتی "L/R"=0، چرخش به راست صورت می گیرد.

یک کپی از بیتی که بر اثر چرخش از یک سر ماتریس خارج شده و در سر دیگر آن قرار می گیرد،به خروجی "OTB" نیز می رود.

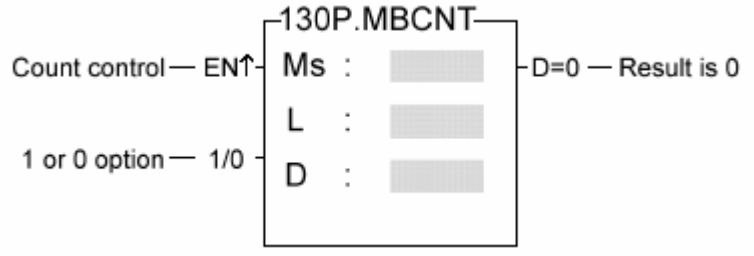


مثال:


X0
129P.MBROT
EN
Ms : R 0
Md : R 0
L/R
L : 5
OTB


Ms
Md
X0=
OTB
0
Before execution
After execution

130.MATRIX BIT STATUS COUNT



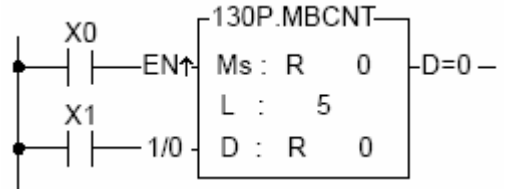
هرگاه "EN" از 0 به 1 تغییر کند:

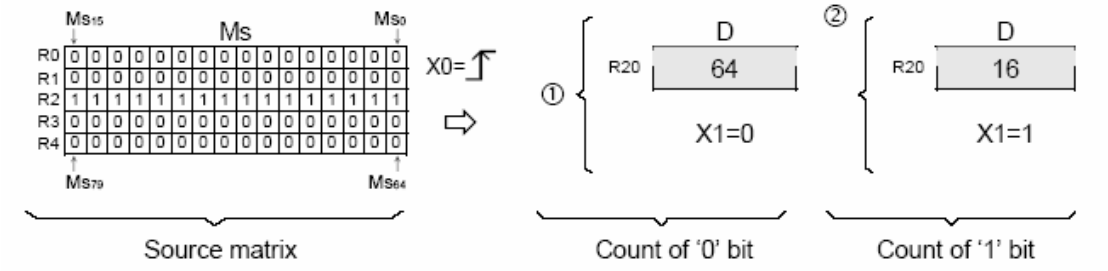
اگر 1="1/0" باشد،تعداد بیت های 1 موجود در ماتریس Ms را شمرده و تعداد در D ذخیره می شود.

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

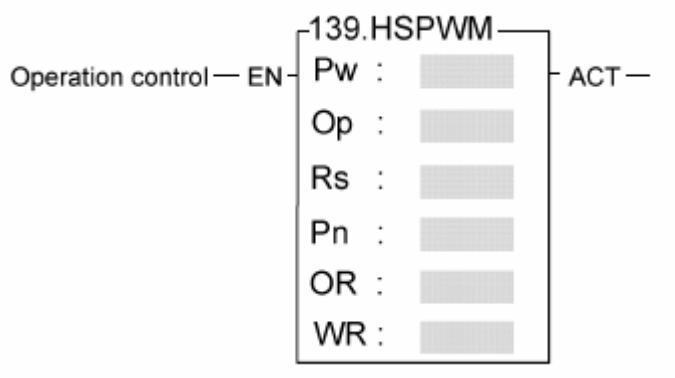
اگر هیچ تعداد از آن بیت مورد نظر موجود نباشد، "D=0" ، 1 خواهد شد.

مثال:





139.HSPWM



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

در PW خروجی مورد نظر انتخاب می شود:

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

در Rs ، رزولوشن یک دوره پالس تعیین می شود:

1=1/1000 , 0=1/100

1) اگر Rs=1/100 :

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

2) اگر Rs=1/1000 :

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

پریود:

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

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

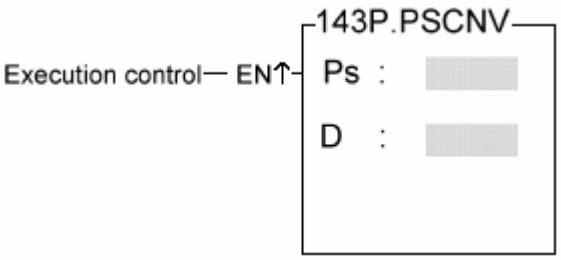
برای Rs=1/100 : OR=0~100

برای Rs=1/1000 : OR=0~1000

142.STOP PULSE OUTPUT

این تابع فرستادن پالس به خروجی را متوقف می کند.

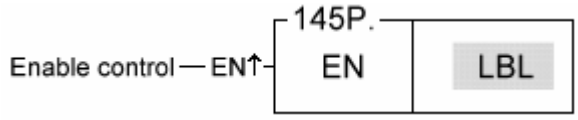
143.PSCNV



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

این تابع فقط وقتی تابع 140 اجرا می شود،می تواند تبدیل را انجام دهد.

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.

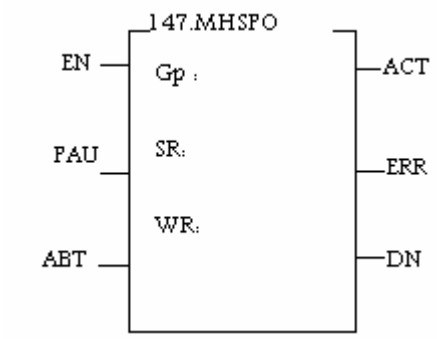
LABLE اینتراپت های مختلف در جدول زیر آورده شده است.

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

146.DISABLE OF INTERRUPT

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

147.MULTI-AXIS HIGH SPEED PULSE OUTPUT



این تابع برای پشتیبانی از interpolation خطی برای کنترل حرکت چند محوره ،استفاده می شود که شامل برنامه حرکتی که توسط برنامه متنی نوشته شده است،می شود.ما هر نقطه از مکان را یک استپ (step) می نامیم. هر استپ 15 رجیستر برای کد کردن در اختیار دارد.

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

یا interpolation خطی دو مجموعه ی دو محوری.

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

خروجی ها برای این تابع باید تنظیم بشوند تا در یکی از مدهای U/D یا A/B خروجی مناسب بدهند در غیر این صورت مانند خروجی های عادی رفتار خواهند کرد.

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

Y1 (Y3,Y5,Y7) و پالس پایین رونده می فرستد.

مد A/B: Y0 (Y2,Y4,Y6) و پالس فاز A می فرستد.

Y1 (Y3,Y5,Y7) و پالس فاز B می فرستد.

ارتباطات برای کنترل موقعیت

M1991	ON: پالس را کند کرده سپس قطع می کند. OFF:فوراً پالس را قطع می کند.
M1992	ON: Ps0 آماده است. OFF: Ps0 فعال است.
M1993	ON: Ps1 آماده است. OFF: Ps1 فعال است.
M1994	ON: Ps2 آماده است. OFF: Ps2 فعال است.
M1995	ON: Ps3 آماده است. Ps3:OFF فعال است.
M1934	ON: Gp0 آخرین step را تمام کرد.
M1935	ON: Gp1 آخرین step را تمام کرد.

M2000 : ON ، چند محور همزمان عمل می کنند.

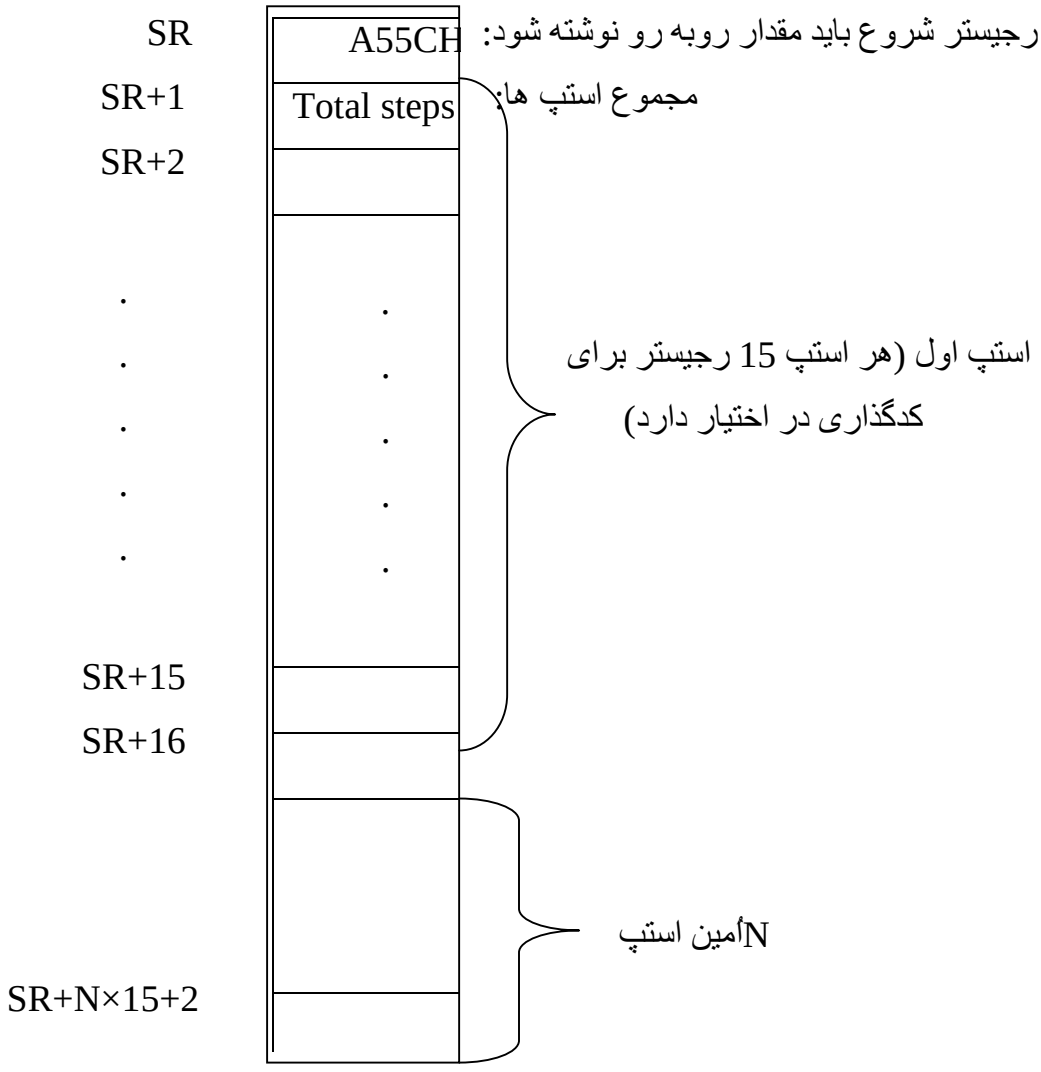
DR4068	سرعت بردار Gp0
DR4070	سرعت بردار Gp1
D4060	Gp0 error code
D4061	Gp1 error code
D4062	شماره استپی که به Gp0 مربوط است (شماره استپی که کامل شده است)
D4063	شماره استپی که به Gp1 مربوط است (شماره استپی که کامل شده است)

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

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

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

در رجیستر شروع باید مقدار روبه رو نوشته شود: A55CH



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

WR+0	استپ اجرا شده یا متوقف شده
WR+1	Working flag
WR+2	توسط سیستم کنترل می شود
WR+3	توسط سیستم کنترل می شود
WR+4	توسط سیستم کنترل می شود
WR+5	توسط سیستم کنترل می شود
WR+6	توسط سیستم کنترل می شود
WR+7	توسط سیستم کنترل می شود
WR+8	توسط سیستم کنترل می شود

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

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

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

B9=ON ، منتظر شرایط انتقال

B10=0N ،خروجی بی انتها

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

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

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

وقتی step کامل می شود "DN" روشن می شود و این وضعیت باقی می ماند.کاربر می تواند از لبه بالا رونده DN برای پاک کردن WR+1استفاده کند.

مشخصه Error		کدError	
R4060(PS0)	0 :	Error فاقد	کدهای اروری که ممکن است در هنگام اجرای تابع 141 به وجود بیایند
R4061(PS1)	1 :	ارور پارامتر0	
R4062(PS2)	2 :	ارور پارامتر1	
R4063(PS3)	3 :	ارور پارامتر2	
R4060(Gp0)	4 :	ارور پارامتر3	
R4061(GP1)	5 :	ارور پارامتر4	
	6 :	ارور پارامتر5	
	7 :	ارور پارامتر6	
	8 :	ارور پارامتر7	
	9 :	ارور پارامتر8	
	10 :	ارور پارامتر9	
	13 :	ارور پارامتر12	
	14 :	ارور پارامتر13	
	15 :	ارور پارامتر14	
	30 :	Error of variable address for speed setting	
	31 :	Error of setting value for speed setting	
	32 :	Error of variable address for stroke setting	
	33 :	Error of setting value for stroke setting	
	34 :	Illegal positioning program	
	35 :	length error of total step	140
	36 :	Over the maximum step	147
	37 :	Limited frequency error	
	38 :	initiate/stop frequency error	
	39 :	Over range of compensation value for movement	
	40 :	Over range of moving stroke	
	41 :	ABS positioning is not allowed within DRVC commands	
	42 :	DRVZ cant follow DRVC	
	50 :	Illegal operation mod of DRVZ	
	51 :	Illegal DOG input number	
	52 :	Illegal PG0 input number	
	53 :	Illegal CLR output number	
	60 :	Illegal linear interpolation command	

توجه : محتوای رجیستر مشخصه ارور، آخرین کد ارور را حفظ می کند.برای اطمینان از این که ارور بیشتری روی نمی دهد،می توانید این رجیستر را 0 کنید.

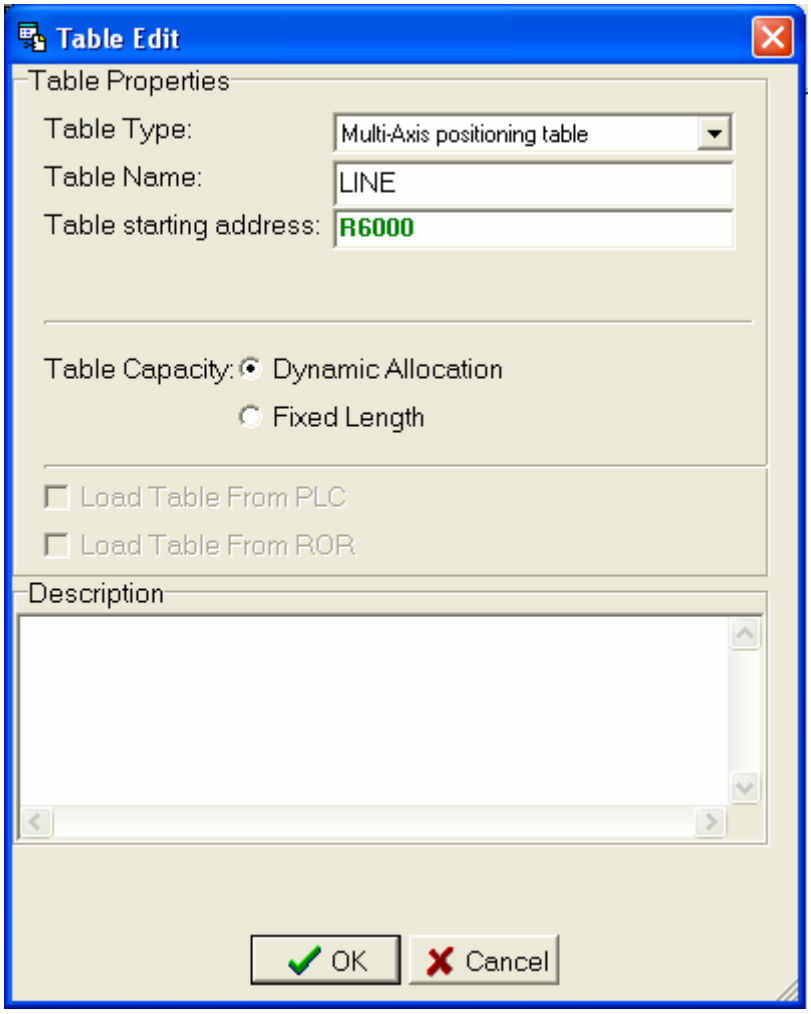
تغییر جدول برنامه Servo توسط WinProladder

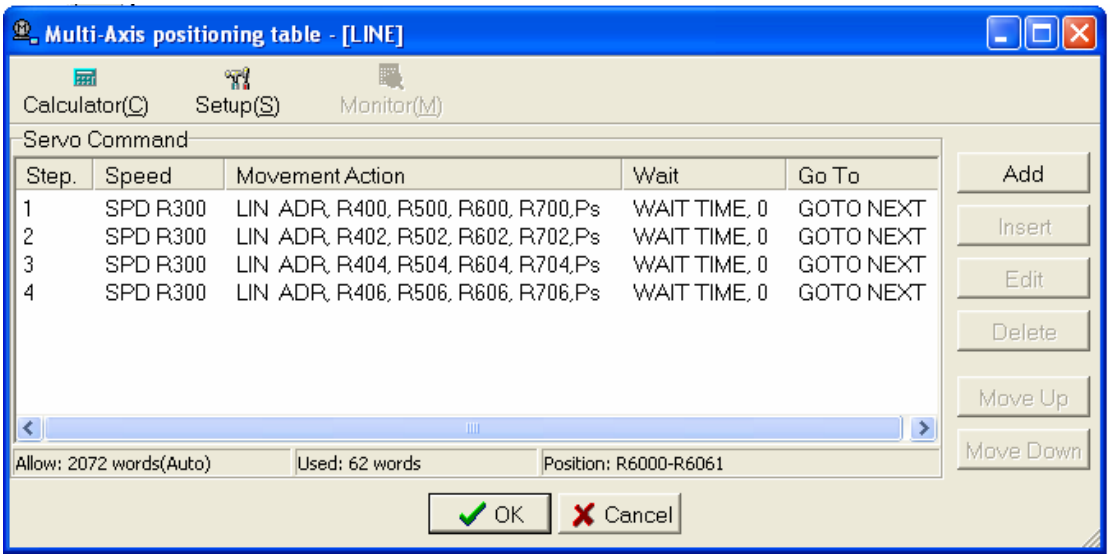
برای استفاده از تابع 147 باید ابتدا تنظیمات زیر صورت گیرد.

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

Project name / Table Edit / Servo Program Table

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





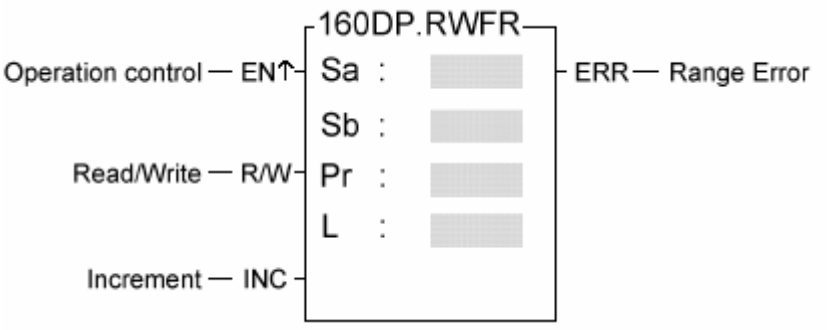
دستورات مکان یابی برای interpolation خطی به صورت زیر هستند:

دستور	رجیستر عملوند	توضیحات
SPD	XXXXXX یا Rxxxx یا Dxxxx	سرعت بردار را برای interpolation خطی تنظیم می کند.سرعت حرکت بر حسب فرکانس یا m/s بیان می شود. (پارامتر-0=1 یا 2،بر حسب فرکانس ، سیستم به صورت پیش فرض در حالت فرکانس است) عملوند می تواند عدد ثابت یا یک رجیستر باشد که اگر رجیستری باشد به 2 رجیستر، فضا احتیاج دارد. فرکانس محور مربوطه برای خروجی از روی مشخصات سرعت بردار،محاسبه خواهد شد. 1≦ رنج فرکانس خروجی ≦ 921600 Hz
LIN	ADR/ABS,X,Y,Z,W,Ut/Ps X: مختصات محور Ps0 Y: مختصات محور Ps1 Z: مختصات محور Ps2 W: مختصات محور Ps3	جهت و اندازه حرکت را مشخص می کند بر حسب پالس یا mm ، درجه ، inch (وقتی پارامتر-0=1 از تابع 141:تنظیمات بر حسب پالس خواهد بود،پیش فرض سیستم پالس است) وقتی ششمین عملوند LIN، Ut باشد،بر اساس مشخصات پارامترهای 1،2،3 از تابع 141،سیستم ، شمارش پالس مربوطه رابه خروجی مورد نظر تبدیل می کند. عملوند اول:ADR یا ABS ADR : حرکت نسبی ABS : حرکت مطلق

		عملوندهای دوم تا پنجم:مختصات محورهای X,Y,Z,W را مشخص می کند. وقتی مختصات یک محور 0 است یا جای خالی (در دستورات متنی)گذاشته می شود و عملوند اول نیز ADR است،یعنی حرکتی در راستای آن محور نخواهد بود. وقتی به جای مختصات یک محور جای خالی (در دستورات متنی)گذاشته می شود و عملوند اول نیز ABS است،یعنی حرکتی در راستای آن محور نخواهد بود. حداکثر اندازه حرکت باید کمتر از 1999999 ± پالس باشد. عملوند ششم:این عملوند واحد حرکت را مشخص می کند. Ut : هر عدد معادل یک واحد خواهد بود(توسط پارامتر 0،3 دستور 141 مشخص می شود) Ps:رزولوشن اجرا شونده،معادل یک پالس خواهد بود.
LINE	ADR/ABS,X,Y,Z,W,Ut/Ps X: مختصات محور Ps0 Y: مختصات محور Ps1 Z: مختصات محور Ps2 W: مختصات محور Ps3	LINE برای interpolation خطی در حرکت بی انتها استفاده می شود. در این دستور ،رابطه بین محورها در خروجی ،طوری است که بقیه محورها از محوری که بیشترین مختصات را دارد تبعیت خواهند کرد. به عنوان مثال:در حالتی که عملوند ششم Ps است و مختصات محورها Ps0=1000,Ps1=500,Ps2=300,Ps3=0 باشد،بدین معنی است که اگر محور Ps0 ، 1000 پالس می فرستد، سپس Ps1 و Ps2 به ترتیب 500 و 300 پالس خواهند فرستاد.(Ps3 کار نمی کند چون مقدارش 0 است) این دستور این نسبت ها را در دادن پالس به خروجی ادامه می دهد تا زمانی که تابع 147 متوقف شود.
WAIT	Time,XXXXX یا Rxxxx یا Dxxxx یا X0~X255 یا Y0~Y255 یا M0~M1911 S0~S999	پس از اتمام خروجی پالس،این دستور باعث انتظار سیستم برای زمان معین می شود سپس به مرحله ی معین شده می رود. این دستور 5 مدل عملوند دارد: Time: (زمان پایه 0.01 ثانیه)وقتی زمان تعیین شده به پایان رسید،مرحله ای را اجرا می کند که توسط GOTO مشخص شده است. X0~X255 / Y0~Y255 / M0~M1911 / S0~S999 تا زمانی صبر می کند که مشخصه مورد نظر ON شود،سپس مرحله ای را اجرا می کند که توسط GOTO مشخص شده است.

EXT	X0~X255 یا Y0~Y255 یا M0~M1911 یا S0~S999	اگر مشخصه مرد نظر این دستور در حین پالس دهی به خروجی،ON شود،فوراً مرحله ای را که توسط GOTO مشخص شده انجام می دهد.اگر تا آخر پالس دهی ON نشود،این دستور مانند WAIT عمل می کند.
GOTO	NEXT یا 1~N یا Rxxxx یا Dxxxx	پس از پایان یافتن دستورات ACT،WAIT و EXT، مرحله ای را اجرا می کند که GOTO به آن اشاره می کند. NEXT :بدین معنی است که مرحله ی بعد اجرا شود. 1~N : مرحله ای را انجام می دهد که LABLE آن عدد مورد نظر باشد. Rxxxx } مرحله ای که اجرا می شود در این Dxxxx } رجیستر ها ذخیره شده است.
MEND		پایان برنامه مکان یابی

160.READ/WRITE FILE REGISTER



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

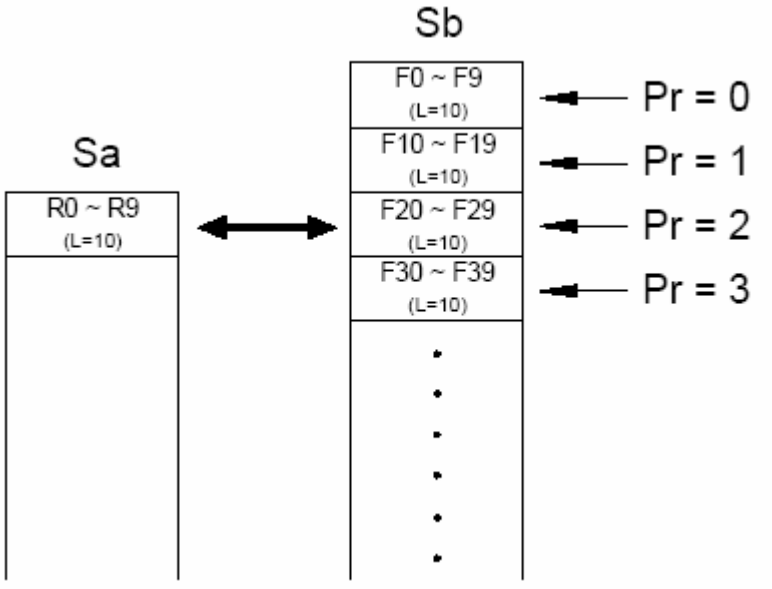
هرگاه "EN" از 0 به 1 تغییر کند:

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

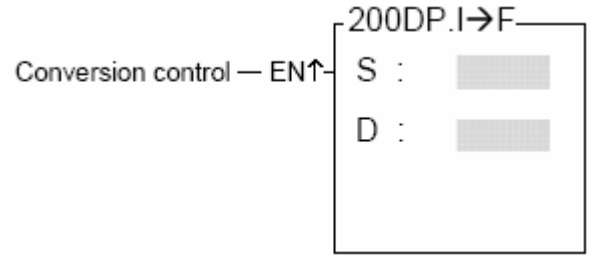
اگر "R/W"=0 ، محتویات رجیسترهای شروع شونده از Sa به داخل رجیسترهای فایل با آدرس پایه Sb به طول L ،از جایی که Pr اشاره می کند ،ریخته می شود.

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

اگر L=0 یا L>511 باشد یا عملیات بخواهد خارج از رنج رجیسترهای فایل (F0~F8191) اجرا شود،"ERR" فعال خواهد شد.



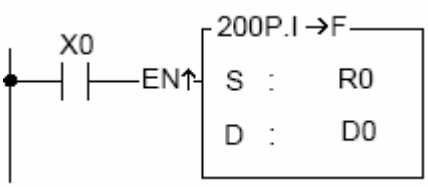
200.CONVERSION of INT to FLOAT

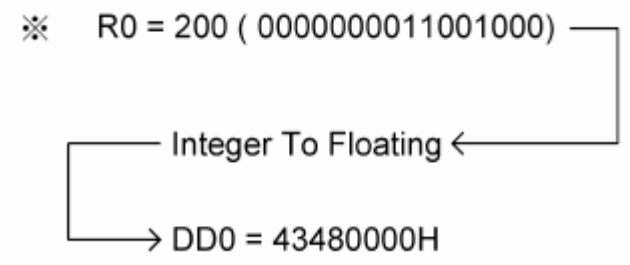


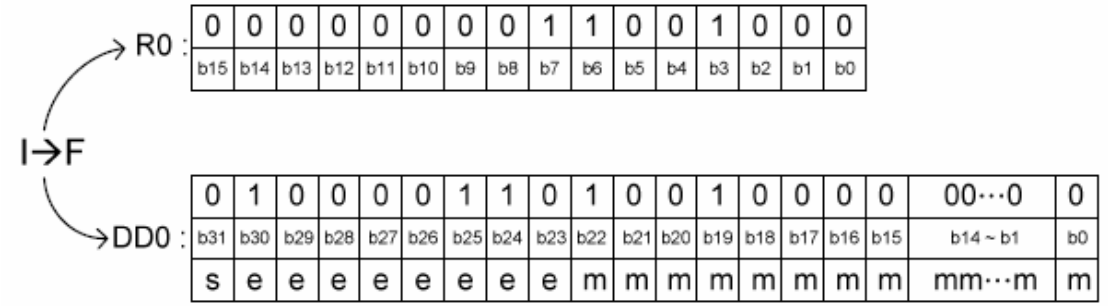
هرگاه "EN" از 0 به 1 تغییر کند:

محتویات رجیستر 16 بیتی (INTEGER) شروع شونده از S درداخل رجیستر 32 بیتی (FLOAT) شروع شونده از D ذخیره می شود.

مثال:







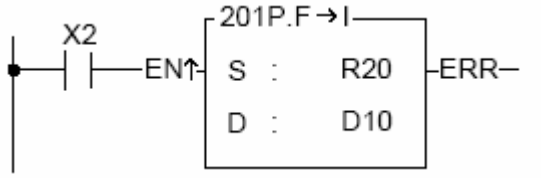
201. FLOAT to INTEGER

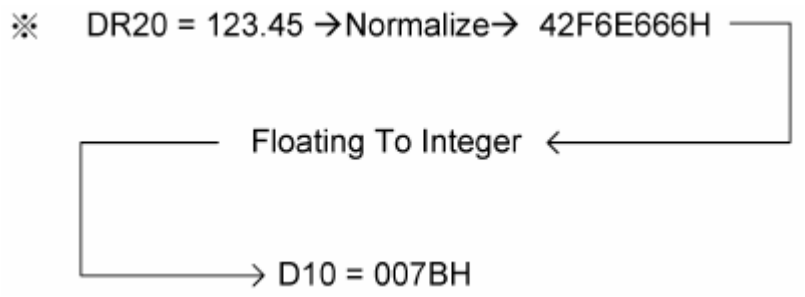
هرگاه "EN" از 0 به 1 تغییر کند:

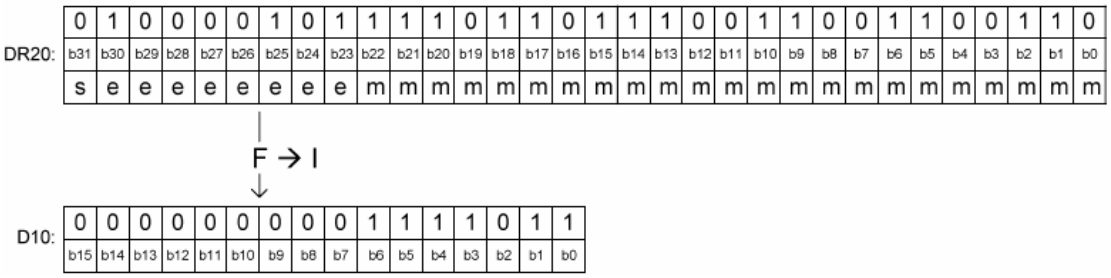
محتویات رجیستر 32 بیتی (FLOAT) شروع شونده از S درداخل رجیستر 16 بیتی (INTEGER) شروع شونده از D ذخیره می شود.

اگر مقدار مبدا فراتر از رنج مقصد باشد و قابل تبدیل شدن نباشد ، "ERR" فعال می شود و مقدار D دست نخورده باقی می ماند.

مثال:







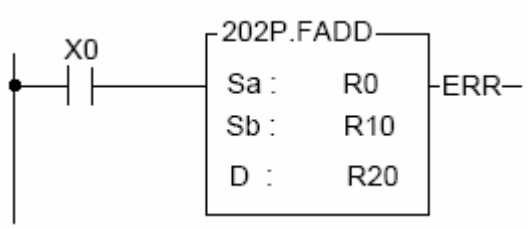
202. FLOATING POINT NUMBER ADDITION

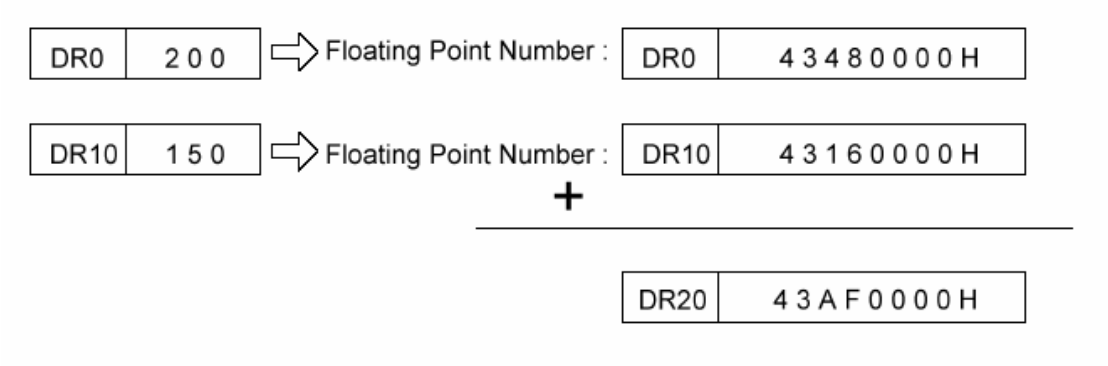
هرگاه "EN" از 0 به 1 تغییر کند:

مقدارFLOAT (32 بیتی) Sa با مقدار FLOAT ، Sb جمع شده و نتیجه در D ریخته می شود.

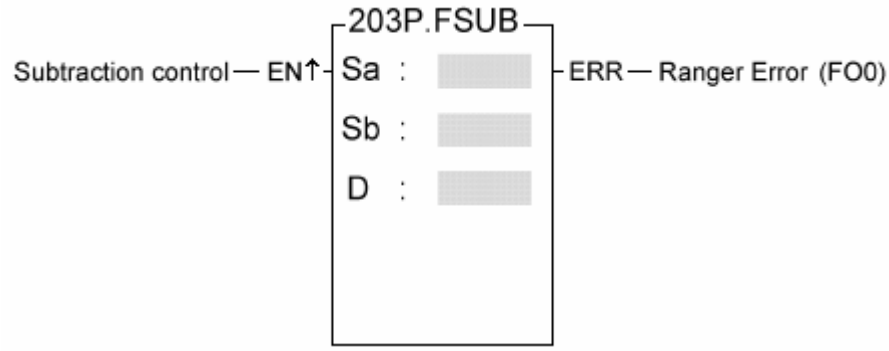
اگر مجموع دو عدد فراتر از رنج یک رجیستر 32 بیتی باشد ($\pm 3.4\times10^{38}$)، "ERR" فعال می شود.

مثال:





203. FLOATING POINT NUMBER SUBTRACTION

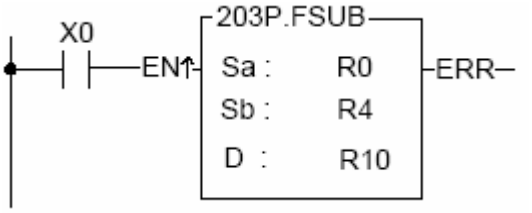


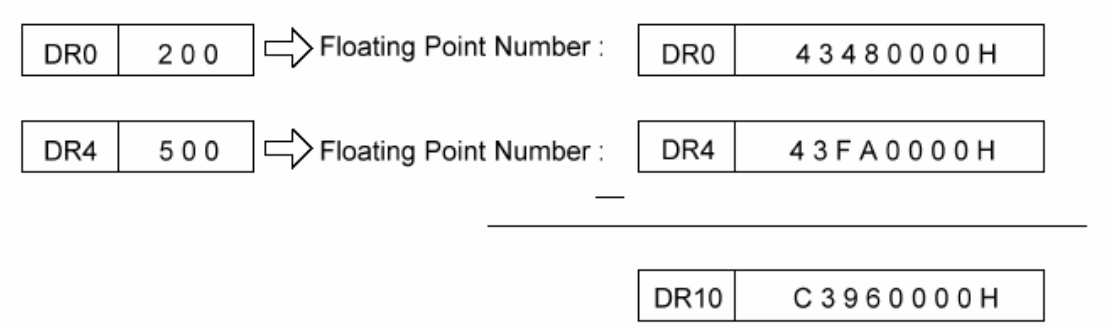
هرگاه "EN" از 0 به 1 تغییر کند:

مقدار FLOAT (32 بیتی) Sa منهای مقدار FLOAT ، Sb شده و نتیجه در D ریخته می شود.

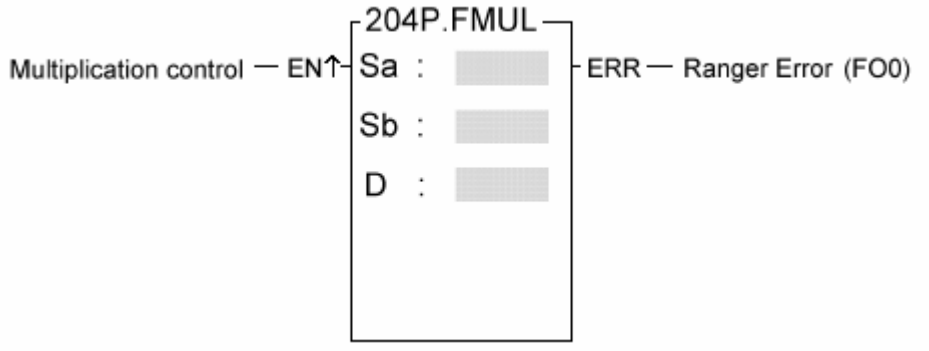
اگر نتیجه تفریق دو عدد فراتر از رنج یک رجیستر 32 بیتی باشد ($\pm 3.4\times10^{38}$)، "ERR" فعال می شود.

مثال:





204. FLOATING POINT NUMBER MULTIPLICATION

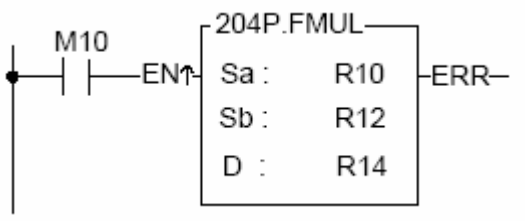


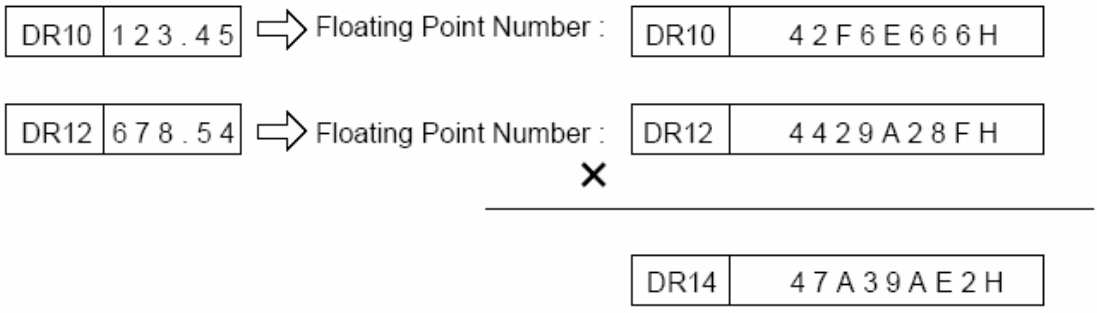
هرگاه "EN" از 0 به 1 تغییر کند:

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 رنج یک رجیستر 32 بیتی باشد ($\pm 3.4\times10^{38}$)، "ERR" فعال می شود.

مثال:





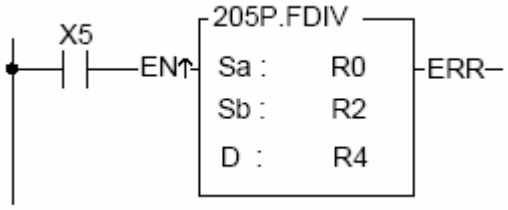
205. FLOATING POINT NUMBER DIVISION

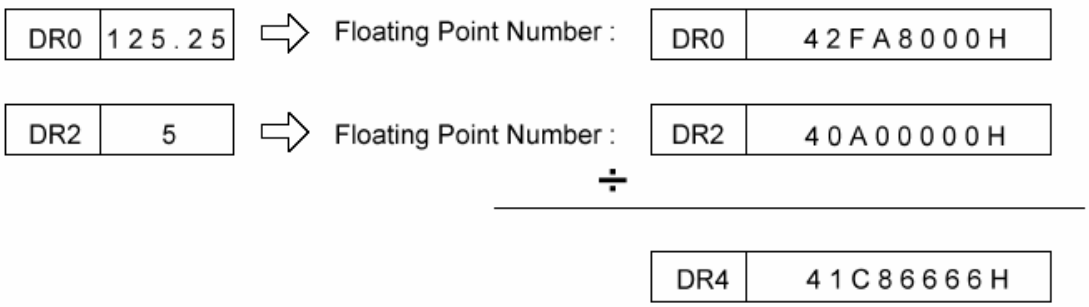
هرگاه "EN" از 0 به 1 تغییر کند:

مقدارFLOAT (32 بیتی) Sa را بر مقدار FLOAT ، Sb کرده و نتیجه در D ریخته می شود.

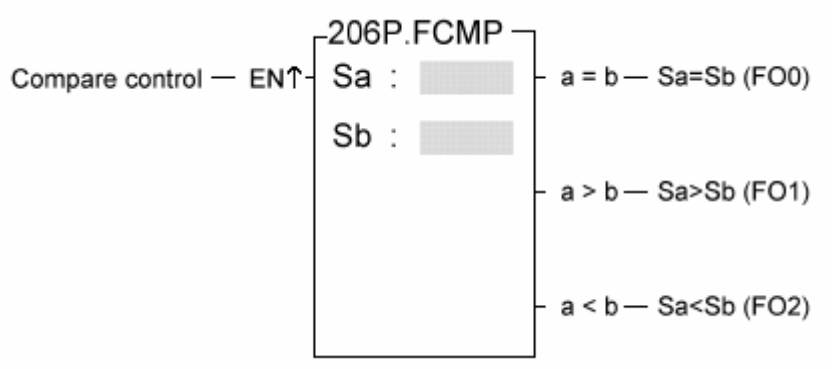
اگر نتیجه تقسیم دو عدد فراتر از رنج یک رجیستر 32 بیتی باشد ($\pm 3.4\times10^{38}$)، "ERR" فعال می شود.

مثال:





206. FLOATING POINT NUMBER COMPARE



هرگاه "EN" از 0 به 1 تغییر کند:

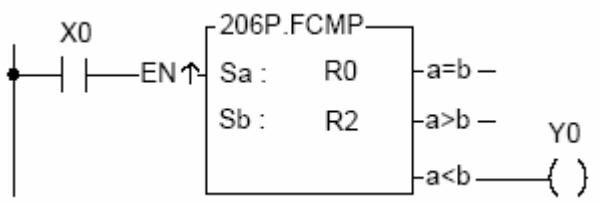
مقدار دو رجیستر 32 بیتی Sa و Sb را با هم مقایسه کرده،

اگر Sa > Sb : "a > b"=1 می شود.

اگر Sa < Sb : "a < b"=1 می شود.

اگر Sa = Sb : "a = b"=1 می شود.

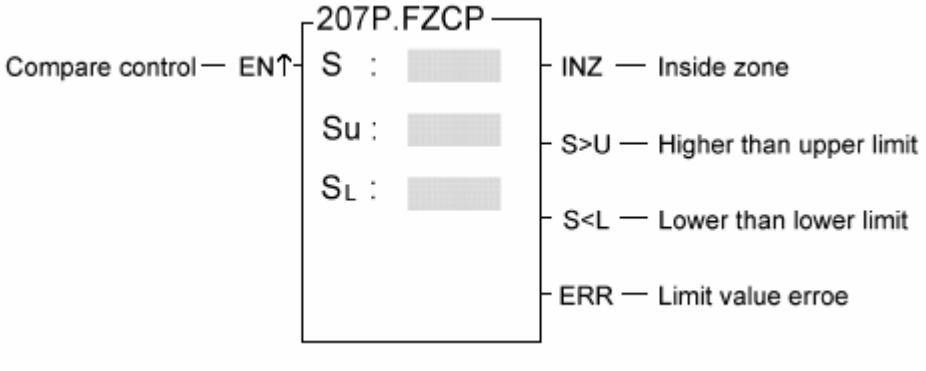
مثال:



DR0	200.1	⇨ Floating Point Number :	DR0	4348199AH

DR2	200.2	⇨ Floating Point Number :	DR2	4348333 3H

207. FLOATING POINT NUMBER ZONE COMPARE



هرگاه "EN" از 0 به 1 تغییر کند:

مقدار 32 بیتی S را با Su و S_L مقایسه کرده،

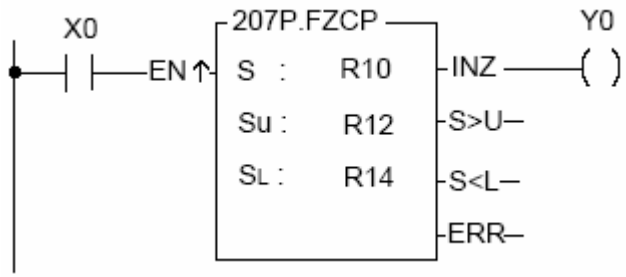
اگر S > Su : "S > U"=1 می شود.

اگر $S < S_L$: "S < L"=1 می شود.

اگر $S_L < S < Su$: "INZ"=1 می شود.

اگر $S_L > Su$: "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)
SL	DR14	1000.1	⇨ Floating Point Number :	DR14	447A0666H	(Lower limit value)

Before-execution

X0 = ↑ → FLOATING ZONE COMPARE → Y0 = 1

Results of execution

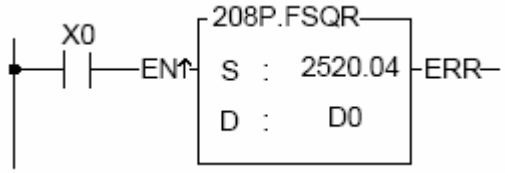
208. FLOATING POINT NUMBER SQUARE ROOT

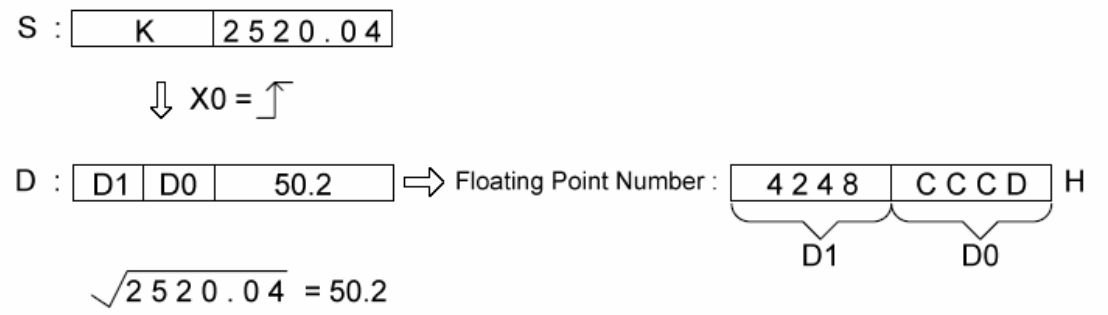
هرگاه "EN" از 0 به 1 تغییر کند:

جذر عدد 32 بیتی S را گرفته و نتیجه را در D می ریزد

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

مثال:





209.SIN INSTRUCTION

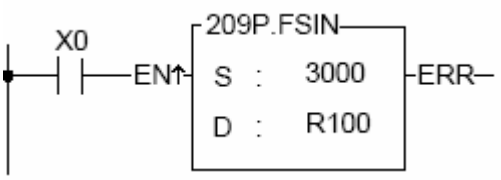
تابع مثلثاتی سینوس

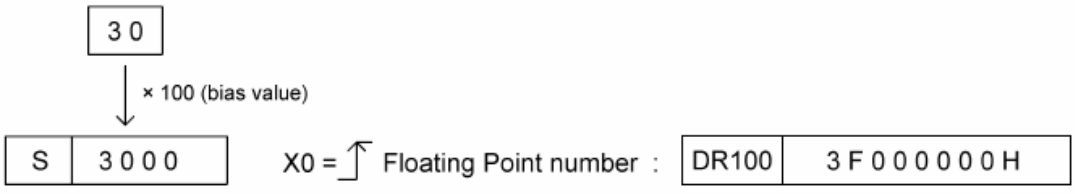
هرگاه "EN" از 0 به 1 تغییر کند:

از مقدار موجود در رجیستر S ، سینوس گرفته و نتیجه را به صورت 32 بیتی در D و D+1 می ریزد.

رنج S در واحد 0.01 درجه ، 18000+~18000- می باشد که اگر فراتر از این رنج داده شود ، "ERR" فعال می شود.

مثال:





SIN(30) = 0.5

210.COS INSTRUCTION

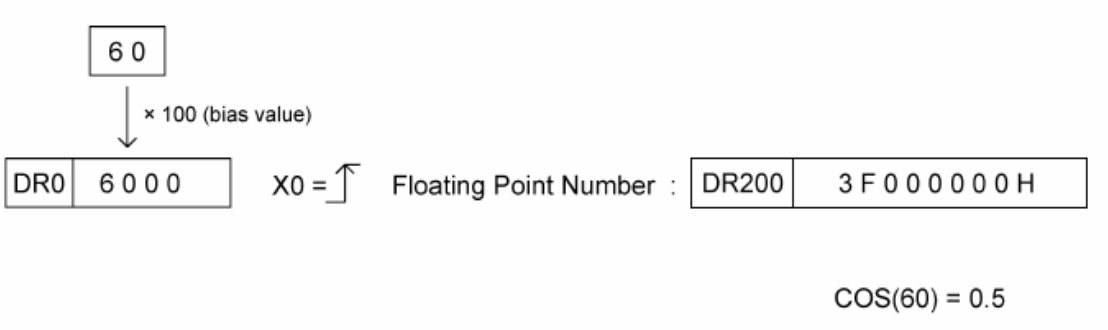
تابع مثلثاتی کسینوس

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.

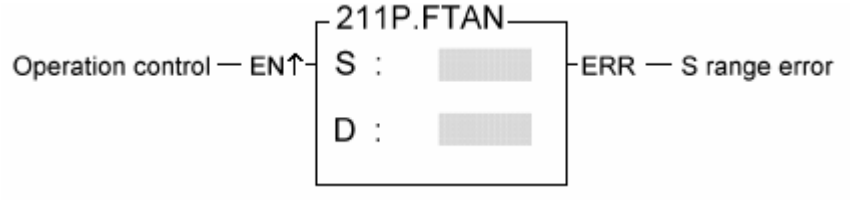
رنج S در واحد 0.01 درجه ، 18000+~18000- می باشد که اگر فراتر از این رنج داده شود ، "ERR" فعال می شود.

مثال:



211.TAN INSTRUCTION

تابع مثلثاتی تانژانت

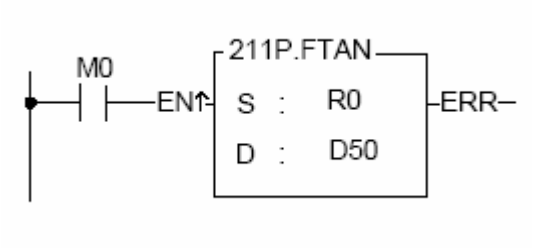


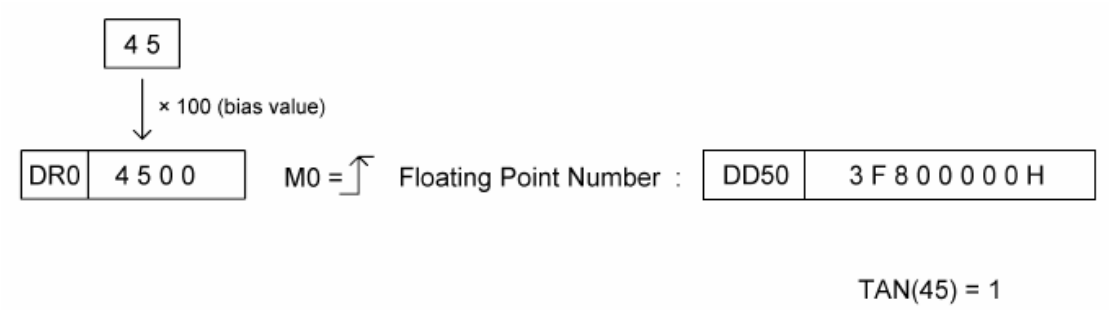
هرگاه "EN" از 0 به 1 تغییر کند:

از مقدار موجود در رجیستر S ، تانژانت گرفته و نتیجه را به صورت 32 بیتی در D و D+1 می ریزد.

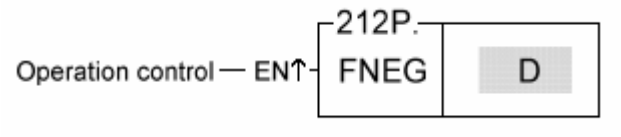
رنج S در واحد 0.01 درجه ، 18000+~18000- می باشد که اگر فراتر از این رنج داده شود ، "ERR" فعال می شود.

مثال:





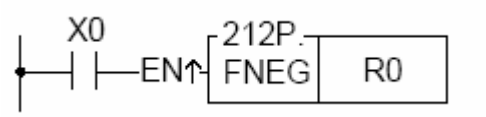
212.CHANGE SIGN OF FLOAT

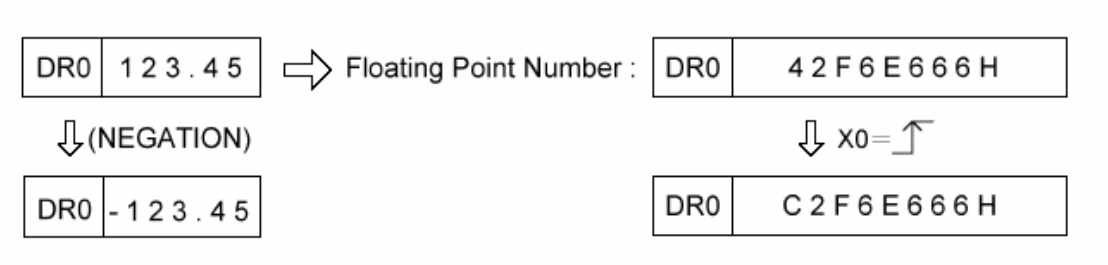


هرگاه "EN" از 0 به 1 تغییر کند:

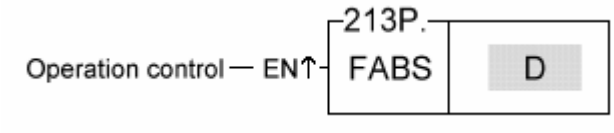
علامت عدد FLOAT(32 بیتی) D را، عکس می کند و نتیجه را در خود D ذخیره می کند.

مثال:





213.FLOAT ABSOLUTE



هرگاه "EN" از 0 به 1 تغییر کند:

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

مثال: