# به نام خدا

# دستورکار کنترل صنعتی

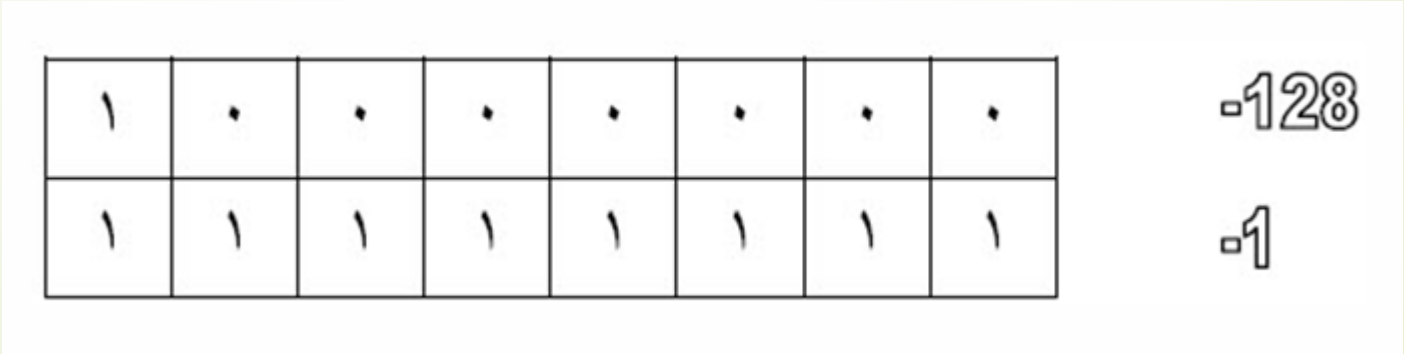
(فصل پنجم)

دکتر فلاح

# ساختمان داده در S5W

**داده نوع short integer :**

**نحوه نوشتن آن در برنامه به فرم SINT می باشد. این نوع داده یک بایت از فضای حافظه را می گیرد و اعداد علامت دار را پشتیبانی می کند، که بیت آخر آن بیت علامت است. اگر بیت آخر ۱ باشد ، یعنی عدد منفی است و اگر صفر باشد یعنی عدد مثبت است و رنج تغییرات آن ۱۲۸- تا ۱۲۷ خواهد بود . اعداد منفی به صورت متمم ۲ ذخیره می شوند . در شکل (۵-۱) اعداد نمایش داده شده به صورت متمم ۲ هستند.**




**شکل (۵-۱)**

# ساختمان داده در S5W

**داده نوع unsigned short integer :**

**نحوه نوشتن آن در برنامه به صورت USINT : این نوع داده یک بایت فضا را اشغال می کند و از اعداد مثبت ، ۰ تا ۲۵۵ پشتیبانی می کند. دقت شود همه عملیات های محاسباتی و منطقی فقط برروی متغیرهای هم نوع صورت می گیرد. حتی عددی که به صورت بیتی از ورودی خوانده ی شود ، به عنوان عدد در نظر گرفته نمی شود. بلکه از نوع بایت است در این نوع از متغیرها بیت ها ارزش مکانی ندارند.**

**داده از نوع integer :**

**نحوه نوشتن آن در برنامه به صورت INT می باشد و دو بایت از فضای حافظه را به متغیر اختصاص داده و دارای علامت بوده و رنج آن از ۳۲۷۶۸- تا ۳۲۷۶۷ می باشد.**

# ساختمان داده در S5W

داده از نوع **unsigned integer** :

این داده به صورت **UINT** مورد استفاده قرار می گیرد و دوبایت از حافظه را اشغال کرده و بی علامت است و رنج آن از ۰ تا ۶۵۵۳۵ می باشد.

داده ازنوع **double integer** :

این داده در برنامه به صورت **DINT** نوشته شده و ۴ بایت از حافظه را اشغال کرده و یک عدد علامت دار بوده و رنجی از ۲^۳۱- تا ۱-(۲^۳۱) دارد.

داده نوع **unsigned double integer** :

این داده به صورت **UDINT** نوشته شده و بی علامت بوده ، ۴ بایت از حافظه را اشغال کرده و رنج آن نیز از ۰ تا ۱- (۲^۳۲) می باشد

# ساختمان داده در S5W

**داده ازنوع long integer :**

**این داده به صورت LINT نوشته می شود، ۸ بایت از حافظه را اشغال کرده و یک داده علامت دار بوده و رنج آن از ۲^۶۳- تا ۱-(۲^۶۳) می باشد. البته همه plc ها از آن پشتیبانی نمی کنند.**

**داده ازنوع unsinged long integer :**

**این داده به صورت ULINT نوشته می شود، از ۸ بایت تشکیل شده و بی علامت است و رنج آن از ۰ تا ۲^۶۴ می باشد، این نوع داده نیز در در برخی PLC ها پشتیبانی نمی شود.**

**داده از نوع Time :**

**متغیری که داده آن از نوع Time است می تواند زمان را ذخیره کند. که در plc های مختلف فضاهای مختلفی به آن اختصاص داده اند.**

# ساختمان داده در S5W

در plc زیمنس این فضا ۲ بایت است در واقع یک شمارنده داریم که با یک پالس یک میلی ثانیه می شمارد، مثلا ۲ دقیقه و ۵ ثانیه را به صورت T#2M5S می دهد، البته در کاربردهای حجیم تر که به بایت می رسد می توان میزان را به ساعت و روز افزایش داد.

داده نوع BOOL :

این نوع داده یک داده منطقی بوده و تنها یک بیت از فضای حافظه را به خود اختصاص می دهد.

داده نوع Byte :

این نوع داده یک بایت از فضای حافظه را به خود اختصاص داده (بیت های داده از نوع بایت دارای ارزش مکانی نیستند). اگر بخواهیم مقدار عددی ذخیره شده در آن را بخوانیم باید آن را به صورت USINT تبدیل نماییم و بلعکس اگر بخواهیم عملیات AND یا OR انجام دهیم حتما باید آن را به بایت تبدیل کنیم.

# ساختمان داده در S5W

**داده نوع WORD :**

**این نوع داده دو بایت از فضای حافظه را به خود اختصاص می دهند و بیت ها دارای ارزش مکانی نیستند.**

**داده نوع double word :**

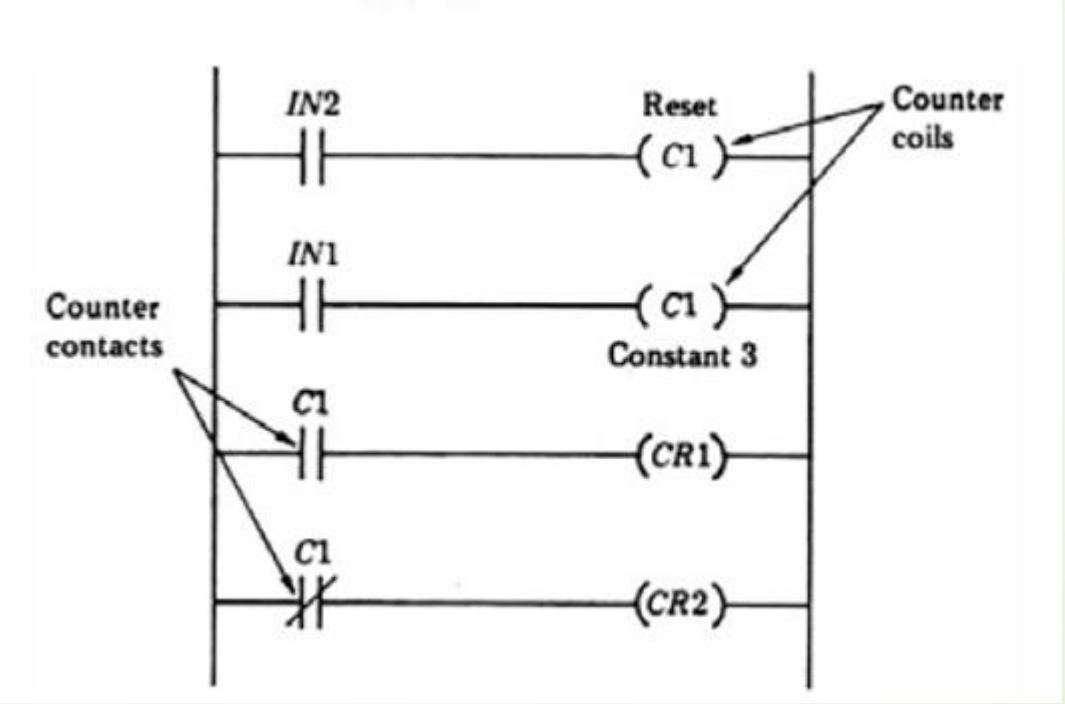
**این نوع داده ها ۴ بایت از فضای حافظه را در برگرفته و بیت های آن دارای ارزش مکانی نیستند. و به صورت DWORD نوشته می شوند.**

**داده نوع LONG WORD :**

**به صورت LWORD نوشته شده و ۸ بایت از فضای حافظه را در بر گرفته و بیت های آن ارزش مکانی ندارند.**

# شمارنده ها

برای شمارش تعداد مشخص از تماس(کنتاکت) صورت گرفته استفاده می شوند. شکل (۵-۲) نمودار نرده ای نحوه استفاده از شمارنده ها را نشان می دهد.




شکل (۵-۲)

# شمارنده ها

در شمارنده ها دو ورودی مستقل موجود می باشند ، یکی شمارنده را صفر کرده یعنی IN2 و تعداد کنتاکت های ورودی یعنی IN1 توسط شمارنده شمرده می شود.

علاوه بر آن مقدار منطقی خروجی شمارنده C1 زمانی روشن می گردد، که تعداد کنتاکت های تحقق یافته بزرگتر یا مساوی مقدار پارامتر شمارنده که در این مثال ۳ است شود. لذا خروجی CR1 با روشن وخاموش شدن حداقل ۳ بار ورودی IN1 روشن می شود.

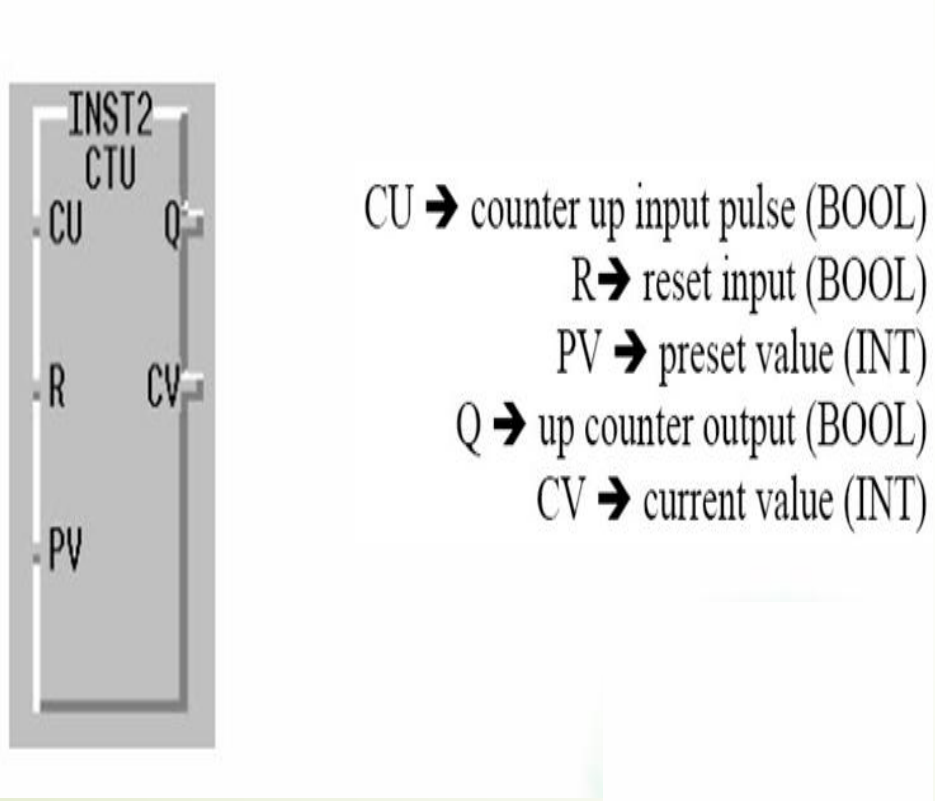
انواع شمارنده ها:

1. شمارنده صعودی
2. شمارنده نزولی
3. شمارنده حلقوی

# شمارنده ها

## شمارنده صعودی (CTU)

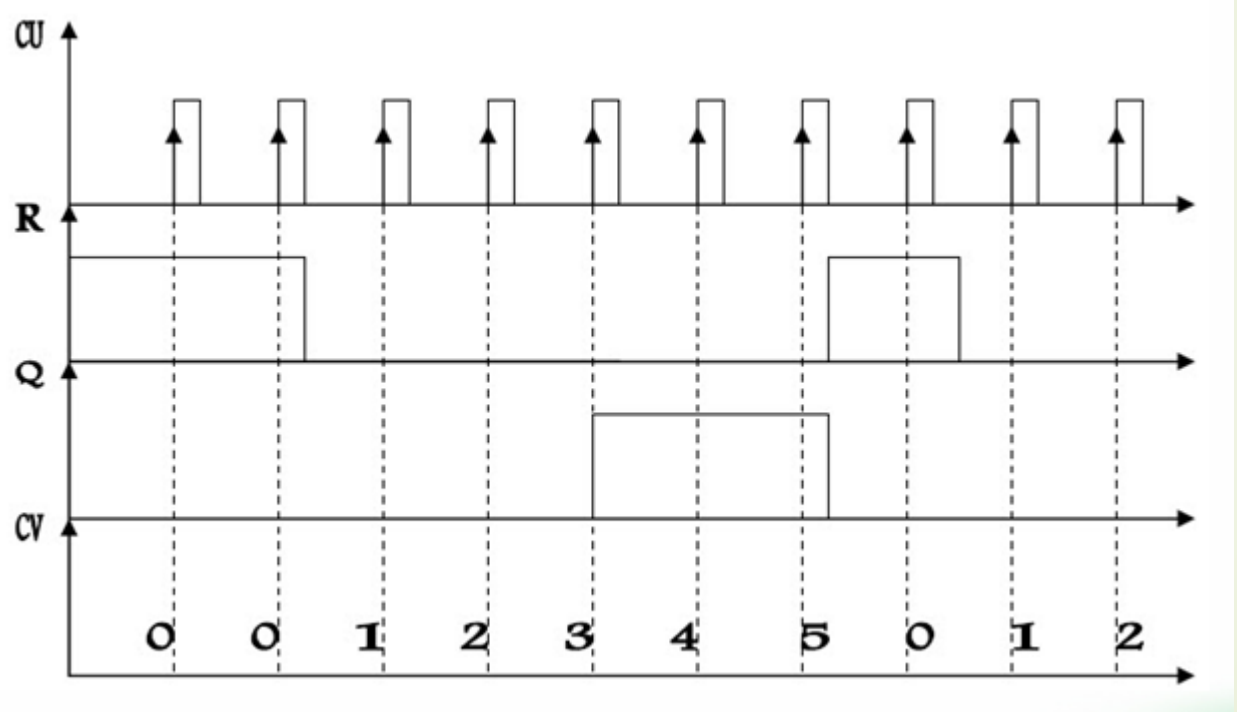
شکل زیر طرح کلی یک شمارنده صعودی نشان می دهد.




شکل (۵-۳)

اگر **R** برابر ۱ باشد. آنگاه **CV** برابر صفر و اگر **CV** بزرگتر مساوی **PV** باشد آنگاه **Q** برابر ۱ می شود، **CV** می تواند منفی باشد اگر **PV** منفی باشد. در این شرایط خروجی **Q** همواره برابر ۱ است چون شرط برقرار است. این شمارنده در **PLC** امکان شمارش تا ۲۰۰ بار در ثانیه را به ما می دهد . برای شمارش بیشتر باید از شمارنده های سخت افزاری استفاده کرد.اگر **PV** را برابر ۳ در نظر بگیریم دیاگرام آن به شکل (۵-۴)خواهد بود.

# شمارنده ها

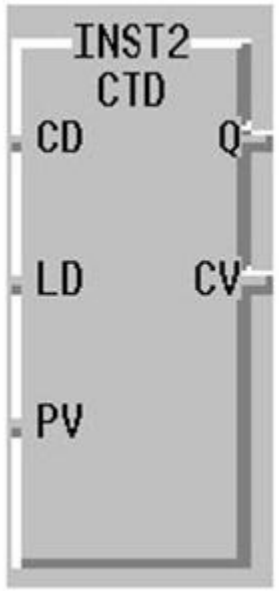



**شکل (۵-۴)**

# شمارنده ها

**شمارنده نزولی(CTD)**

**شکل زیر شمای کلی یک شمارنده نزولی است.**

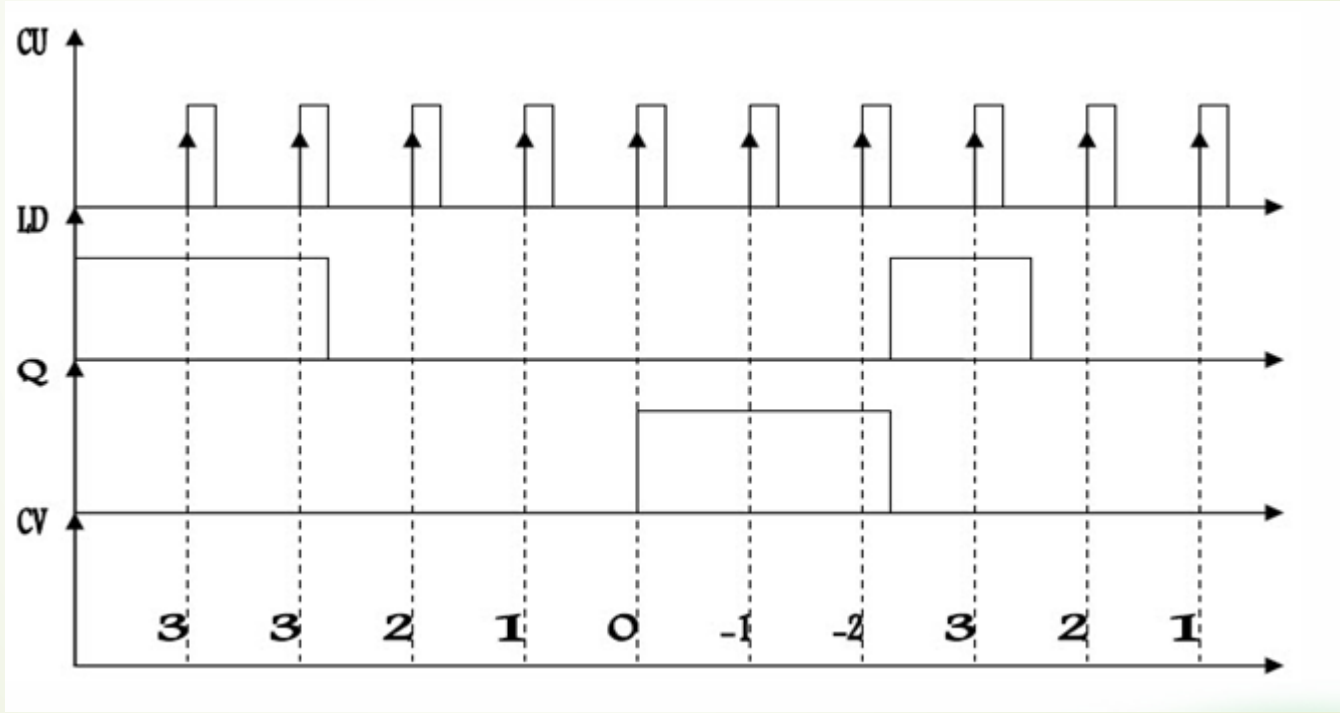



CD ➔ counter Down input pulse (BOOL)
LD➔ Load Perset Value (BOOL)
PV ➔ preset value (INT)
Q ➔ up counter output (BOOL)
CV ➔ current value (INT)

**شکل (۵-۵)**

# شمارنده ها

**اگر LD برابر با یک باشد آنگاه CV برابر PV و اگر CV<=0 آنگاه Q برابر ۱ می شود. دیاگرام تایمینگ با فرض PV برابر ۳ به شکل زیر می باشد.**




**شکل (۵-۶)**

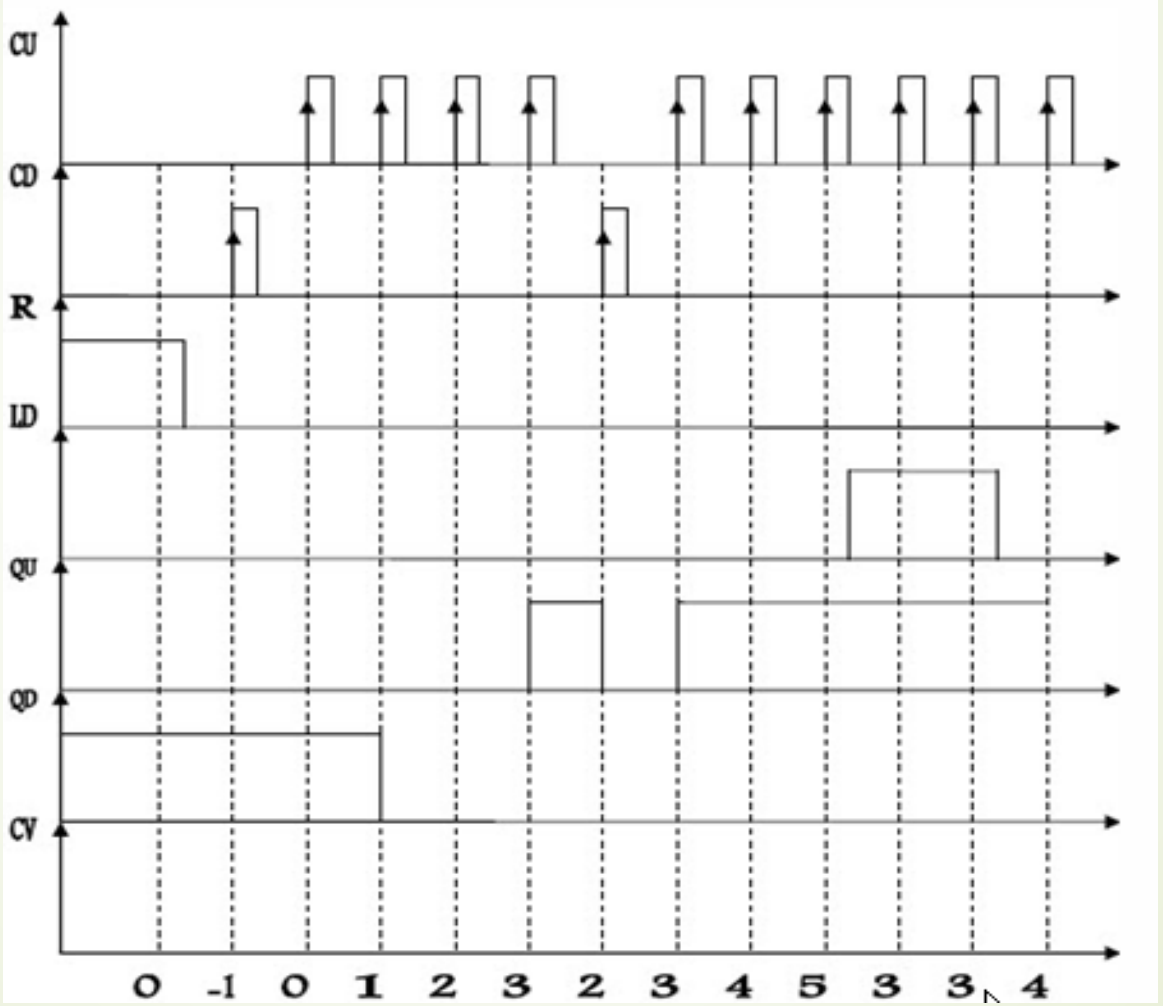
# شمارنده ها

## شمارنده صعودی نزولی(CTUD)

بلوک دیاگرام آن به شکل (۵-۷) می باشد. اگر R=1 باشد ، آنگاه CV=0 واگر LD=1 ، آنگاه CV=PV و اگرPV<=CV آنگاه QV=1 و اگر CV<=0 آنگاه QD=1 می شود. این مطالب به ترتیب گفته شده دارای الویت است. اگر R و LD با هم یک شوند اول R پذیرفته می شود و سپس LD دیاگرام تایمینگ آن با فرض PV=3 به شکل (۵-۸) خواهد شد.


شکل (۵-۷)

# شمارنده ها

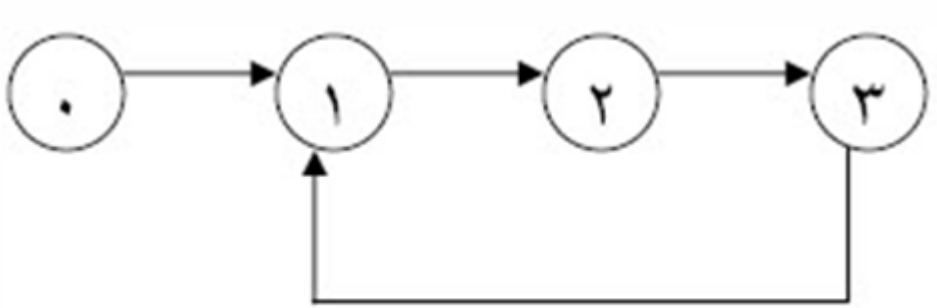



شکل (۵-۸)

# شمارنده ها

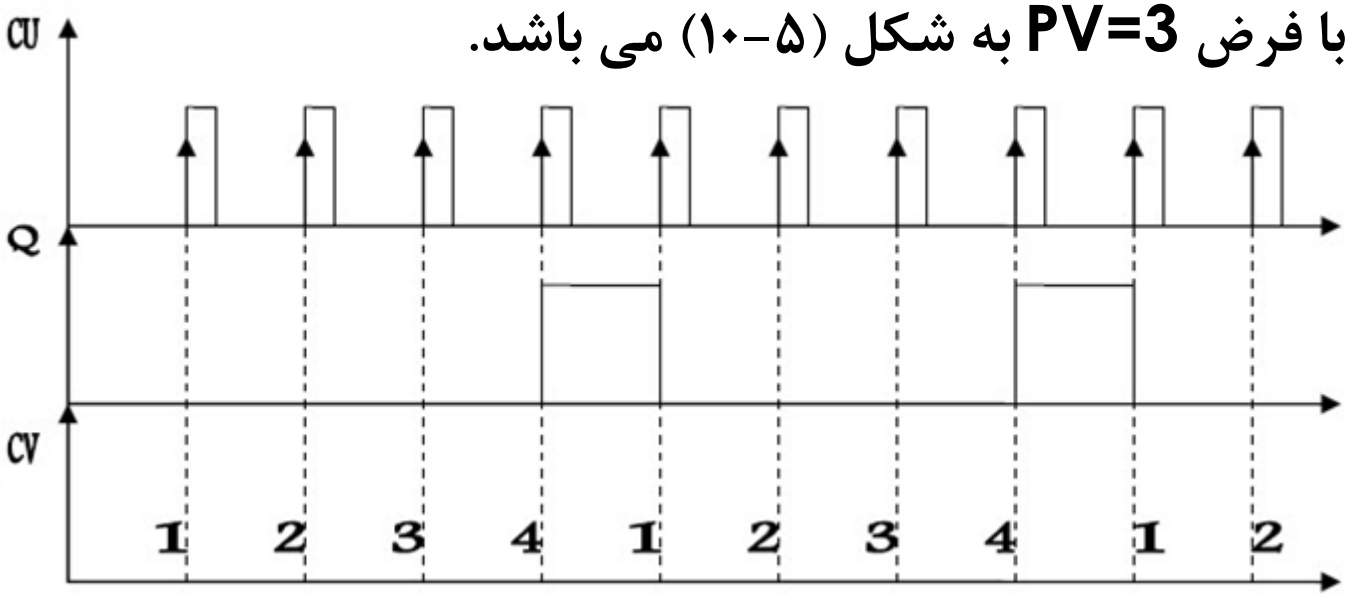
**شمارنده حلقوی یا RC**

**در شکل (۵-۹) یک شمارنده حلقوی نمایش داده شده است.**




**شکل (۵-۹)**

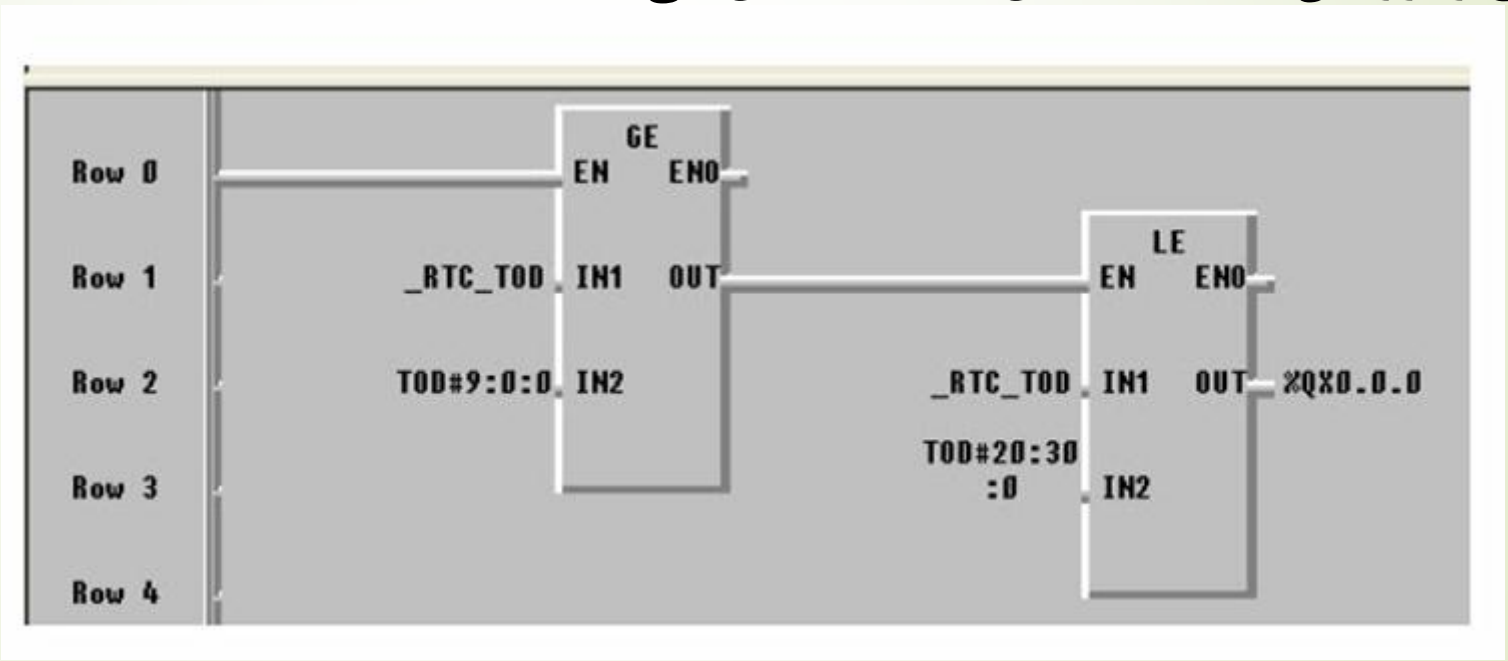
**دیاگرام تایمینگ آن با فرض PV=3 به شکل (۵-۱۰) می باشد.**




**شکل (۵-۱۰)**

# RTC

یک آرایه است که زمان دقیق را در خود حفظ می کند. یکی از خانه های آن به نام Time of date به صورت RTC-TOD قابل آدرس دهی می باشد مثلا اگر بخواهیم برنامه ای بنویسیم که همه روزه از ساعت ۹ الی ۲۰.۳۰ دقیقه شب موتوری را روشن کند. به شکل (۵-۱۱) عمل می کند.



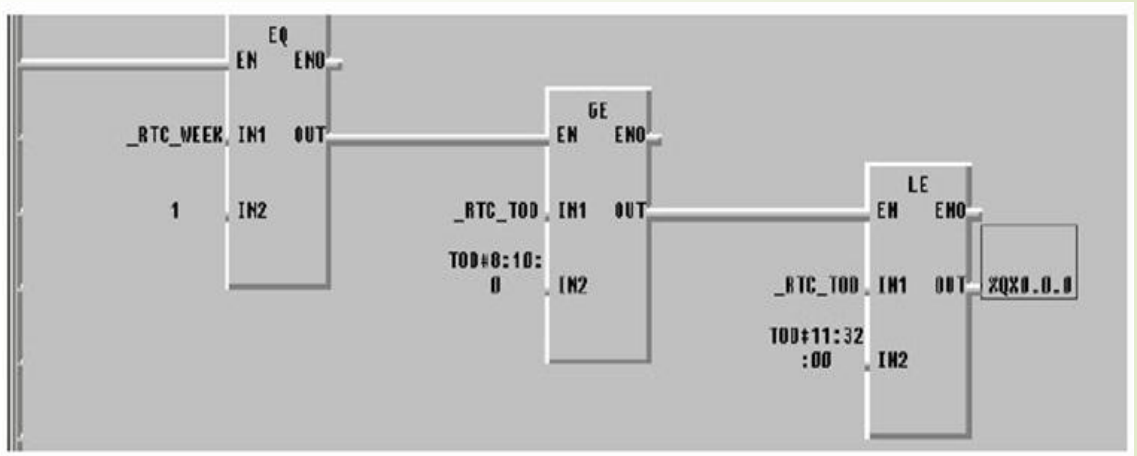

شکل (۵-۱۱)

# RTC

متغیری که روزها ی هفته را نگه می دارد . به **RTC-Week** معروف است که از جدول زیر پیروی می کند.

جدول (۵-۱)

| روزهای هفته | شناسه عددی |
|---|---|
| دوشنبه | ۰ |
| سه شنبه | ۱ |
| چهارشنبه | ۲ |
| پنجشنبه | ۳ |
| جمعه | ۴ |
| شنبه | ۵ |
| یکشنبه | ۶ |




شکل (۵-۱۲)

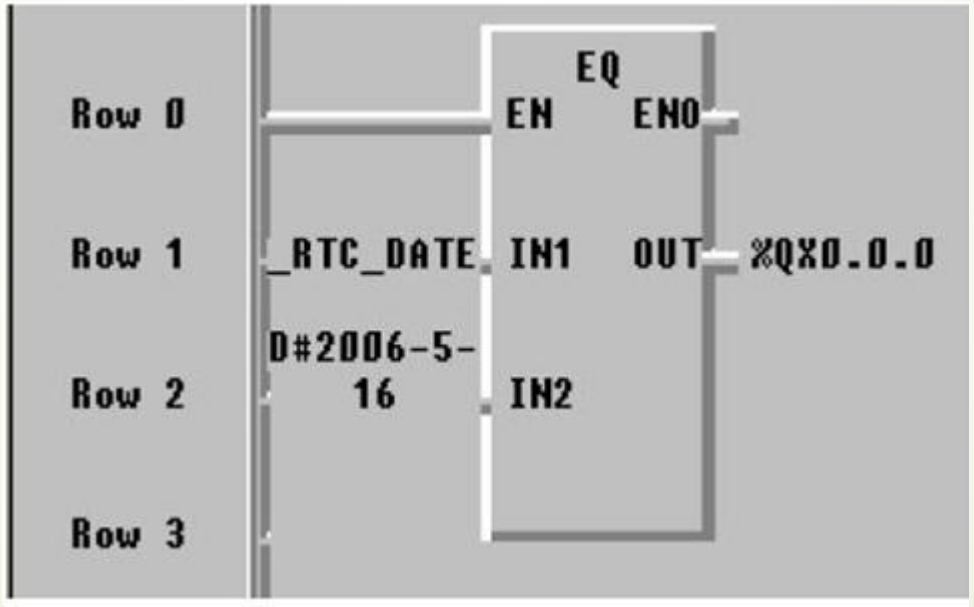
مثلا اگر بخواهیم برنامه ای بنویسیم که فقط روزهای سه شنبه از ۸:۱۰ تا۱۱:۳۲ دقیقه موتوری را روشن کند به این صورت عمل می کنیم، که در شکل (۵-۱۲)آمده است .

# RTC

**متغیر RTC-DATE:**

**تاریخ به مانند سال ، ماه ، روز را در خودنگه می دارد.**

**مثلا اگر برنامه ای بخواهیم بنویسیم که فقط در یک روز خاص ، موتوری را ON کند به شکل زیر عمل می کنیم.**




**شکل (۵-۱۳)**

# RTC

تمامی این متغیرها در آرایه ای به نام RTC-TIME[ ] به مانند جدول زیر وجود دارد.

جدول (۵-۲)

| _RTC_TIME[ ] | [0] | [1] | [2] | [3] | [4] | [5] | [6] | [7] |
|---|---|---|---|---|---|---|---|---|
| توضیحات | سال از ۰۰ تا ۹۹ | ماه از ۱ تا ۱۲ | روزهای ماه از ۱ تا ۳۱ | ساعت از ۰۰ تا ۲۳ | دقیقه از ۰۰ تا ۵۹ | ثانیه از ۰۰ تا ۵۹ | روزهای هفته از ۰ تا ۶ | قرن از ۲۰ تا ۲۱ |

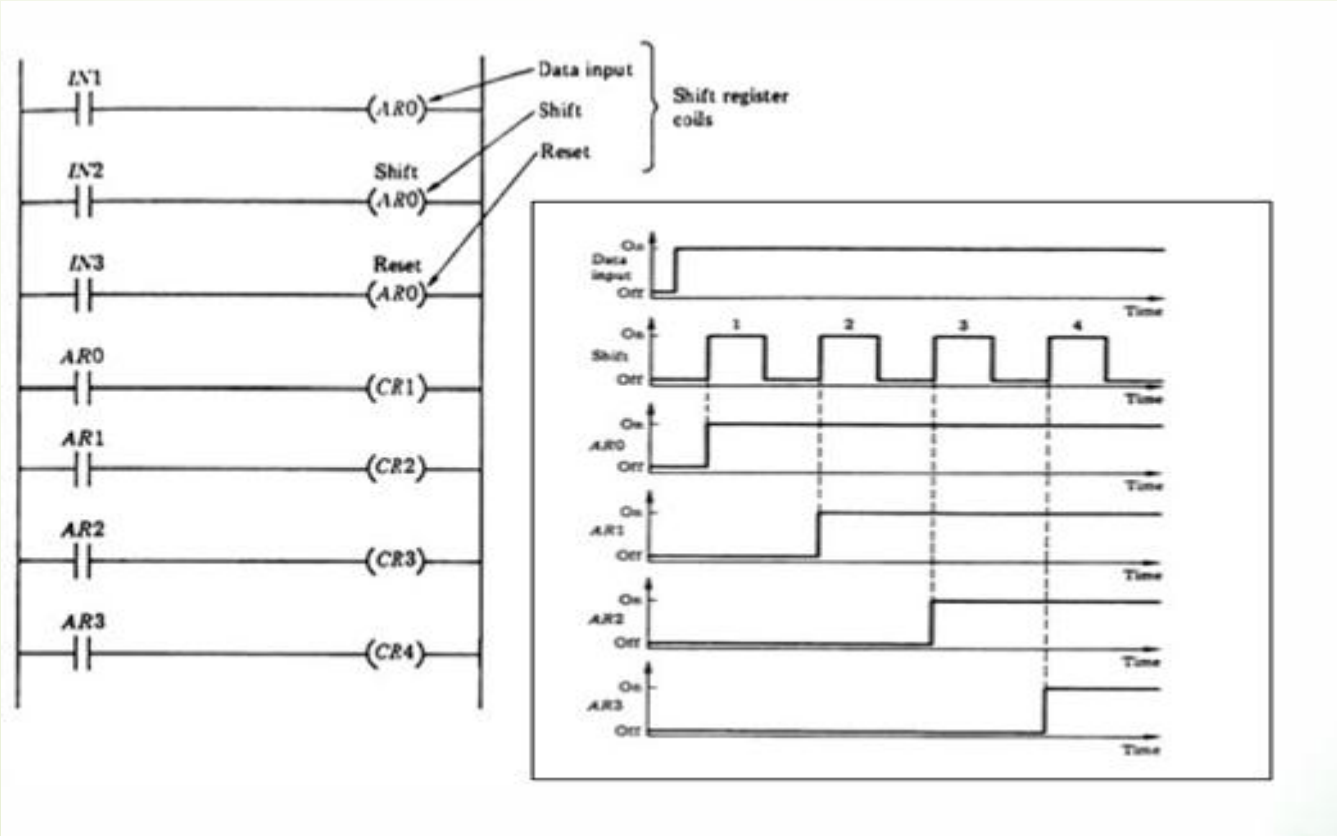
دو متغیر زمان و تاریخ به صورت D و TOD می باشند.

به عنوان مثال: TOD#12:31:20 یعنی ۱۲ ساعت ، ۳۱ دقیقه و ۲۰ ثانیه

D#2016-5-16 یعنی روز ۱۶ اهم از ماه پنجم در سال ۲۰۱۶

# شیفت رجیسترها

تعدادی رله های غیر واقعی که به صورت حافظه در کنار یکدیگر قرار گرفته اند تا یک رجیستر را تشکیل دهند. که انواع ۴ ، ۸ ، ۱۶ بیتی آن ها وجود دارند. طبق شکل(۵-۱۴) یک رجیستر ۴ بیتی را در نظر بگیرید.



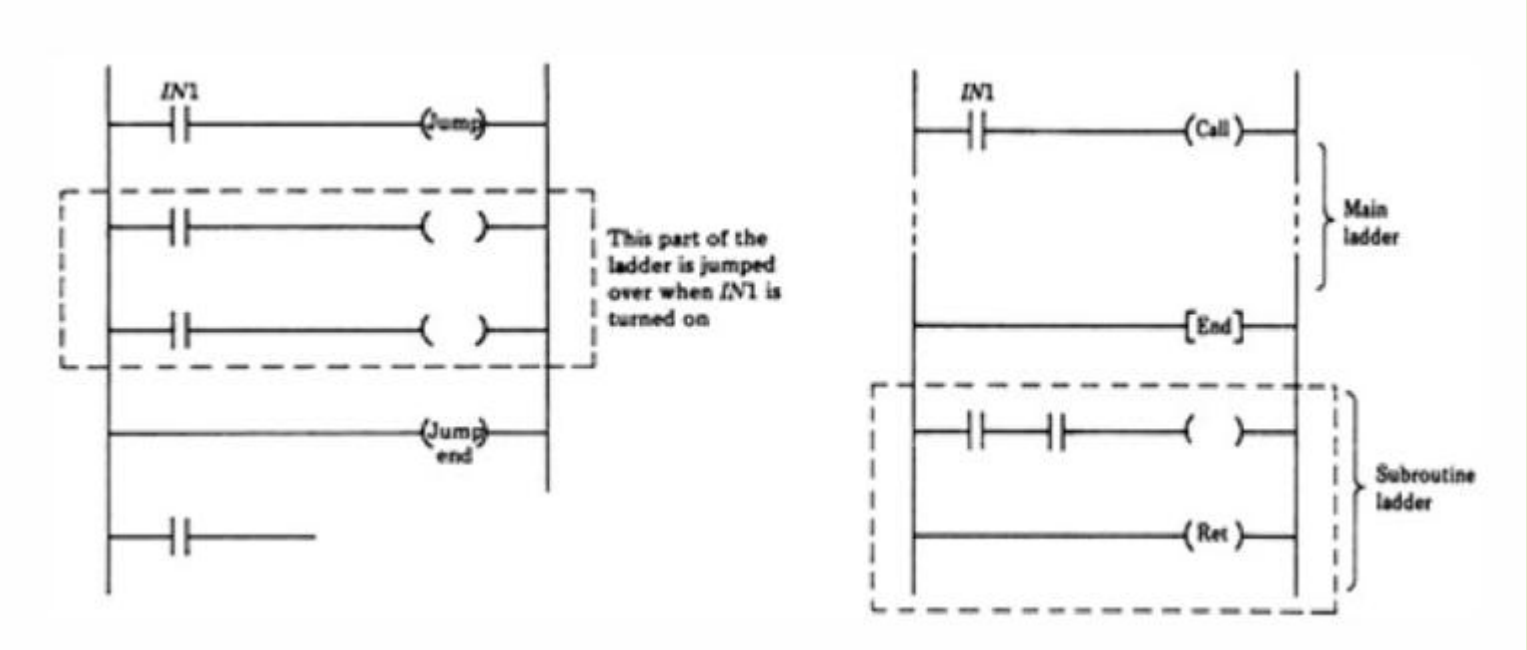

شکل (۵-۱۴)

# شیفت رجیسترها

که از حافظه های AR0 تا AR3 تشکیل شده است.با تحریک این رجیستر ها می توان داده های آن را یک بیت شیفت داد. اگر به مدار نردبانی در شکل(۵-۱۴)دقت کنید. می بینیم که از ۳ ورودی تشکیل شده است. که ورودی اول داده را به رجیستر وارد می کند. این داده بروی رجیستر اول AR0 قرار می گیرد، ورودی دوم دستور شیفت را ارسال نموده و ورودی آخر برای صفر کردن مقدار رجیستر است، با روشن شدن ورودی IN1 مقدار AR0 یک می گردد با هربار اتصال ورودی IN2 مقادیر رجیستر شیفت پیدا کرده و با اتصال ورودی IN3 مقادیر تمامی بیت های رجیستر صفر می گردد ، اگر با دیاگرام زمانی پیش برویم. نحوه ی روشن شدن بیت های A0 تا A3 و به تبع آن روشن شده بیت های CR1 تا CR4 با زمان بندی نمایش داده شده در این شکل روشن خواهد شد.

# زیر برنامه ها و یا پرش در برنامه

مطابق با شکل (۵-۱۵) می توان از یک برنامه پرش نمود و یا یک زیر برنامه را فراخوانی کرد.




شکل (۵-۱۵)

# فانکشن بلاک ها در استاندارد IEC 61131-3

یک فانکشن یک برنامه کوچک منطقی است که با اجرای آن یک کار انجام می شود
ویژگی های فانکشن ها عبارت است از :

1. فانکشن ها فاقد حافظه هستند ، باید برای پایه خروجی آن ها متغیری تعریف کرد تا داده در آن ذخیره شود. برعکس تایمرها .
2. نوع داده ورودی خروجی باید از یک نوع باشد.
3. هر فانکشن می تواند چند ورودی داشته باشد ولی همواره دارای یک خروجی است.
4. هر فانکشن دارای یک پایه توان کننده ورودی است.

(البته باید توجه به این نکته شود که پایه حساس به لبه است یا سطح)

5. هرگاه EN=1 آنگاه فانکشن اجرا می شود.

# فانکشن بلاک ها در استاندارد IEC 61131-3

6. هر فانکشن دارای یک فعال ساز خروجی نیز می باشد هرکاه **EN** برابر ۱ باشد ، در صورتی که خطایی در فانکشن رخ ندهد . **ENO** نیز برابر ۱ خواهد شد.

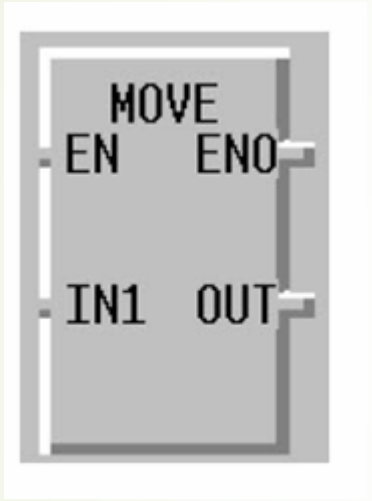
**ویژگی های فانکشن بلاک ها:**

1. فاکشن بلاک ها دارای حافظه هستند.
2. نوع داده ورودی و خروجی در فاکشن بلاک ها می توانند متفاوت باشد.
3. یک فانکشن بلاک می تواند شامل تعدادی ورودی و تعدادی خروجی باشد.
4. هر فانکشن بلاک دارای یک پایه توان کننده ورودی است ، هرگاه مقدار این پایه از صفر به یک تغییر کند ، برنامه فانکشن بلاک اجرا می شود.

# فانکشن بلاک ها در استاندارد IEC 61131-3

**فانکشن MOVE :**

**یک فانکشن موو به شکل (۵-۱۶) می باشد.**




**شکل (۵-۱۶)**

**فانکشن ADD : تعداد ورودی ها از ۲ تا ۸ می باشد.(جمع)**

**فانکشن DIV : که تعداد ورودی های آن ۲ می باشد.(تقسیم)**

**فانکشن MUL : تعداد ورودی های آن ۲ تا ۸ است.(ضرب)**

**فانکشن SUB : تعداد ورودی های آن ۲ می باشد.(تفریق)**

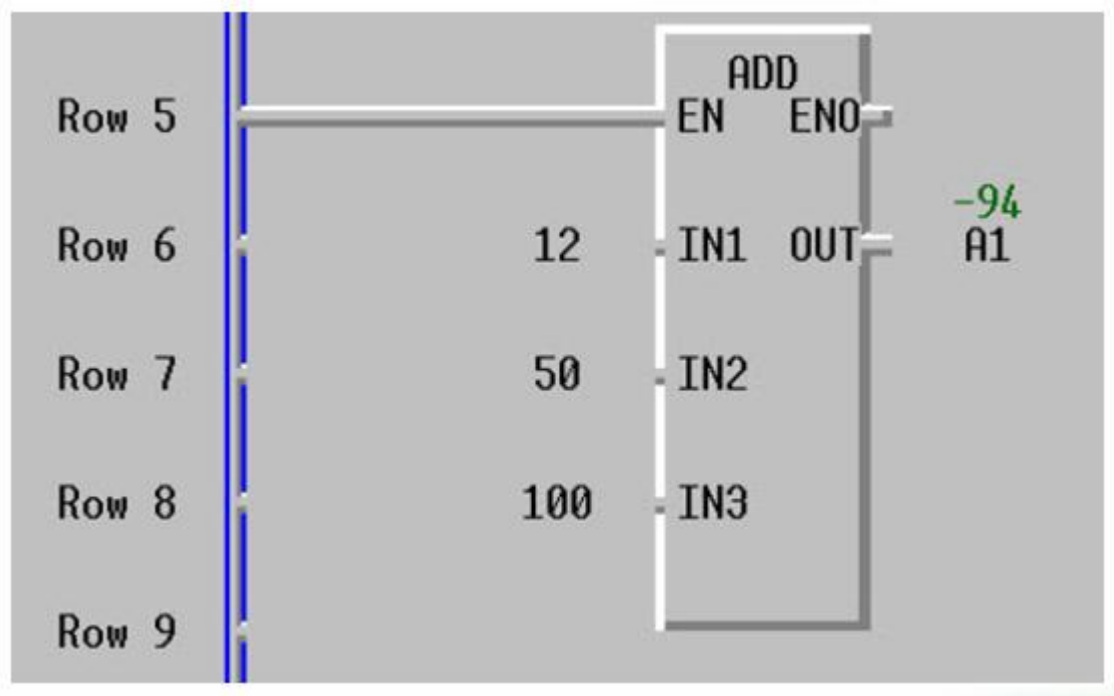
**فانکشن ABS : تعداد ورودی های آن یک می باشد.(قدر مطلق)**

**فانکشن MOD : تعداد ورودی های آن ۲ می باشد.(باقیمانده تقسیم)**

# فانکشن بلاک ها در استاندارد IEC 61131-3

البته برای عملیات های پیچیده تر به پردازنده های قویتری نیاز مندیم.

مثال) برنامه ای بنویسید که ۳ مقدار ۱۲ ، ۵۰ و ۱۰۰ را با هم جمع کند.

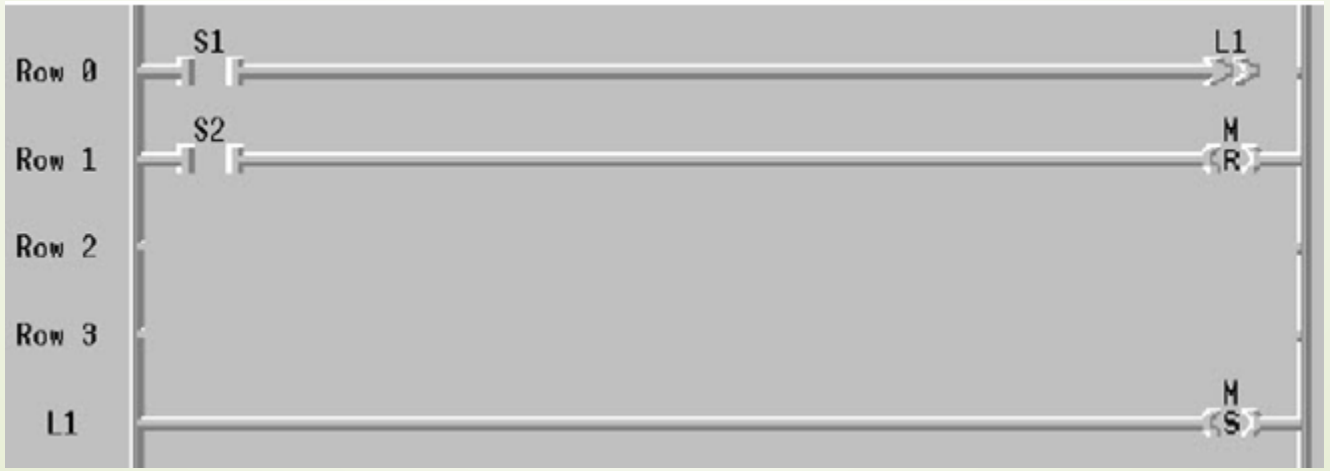



شکل (۵-۱۷)

# فانکشن بلاک ها در استاندارد IEC 61131-3

البته همانطور که میبینید جواب نهایی ۹۴- شد ، که باید۱۶۲ باشد. علت آن این است که متغیر خروجی یعنی A1 از نوع SINT تعریف شده که اعداد بین ۱۲۸- تا ۱۲۷ را ذخیره کرد.(باید توجه شود تمامی عملیات های ریاضی فانکشن هستند.)

دستور Jump :یک پرش بی قید و شرط است. اگر RLO یعنی نتیجه عمل منطقی سمت چپ، یک باشد، به بر چسب مشخص شده پرش می کند و خطوط برنامه بین سطح فعلی و برچسب را اجرا نمی کند.به عنوان مثال اگر در شکل (۵-۱۸) S1 برابر ۱ باشد خطوط ۱ تا ۳ اجرا نمی شوند.

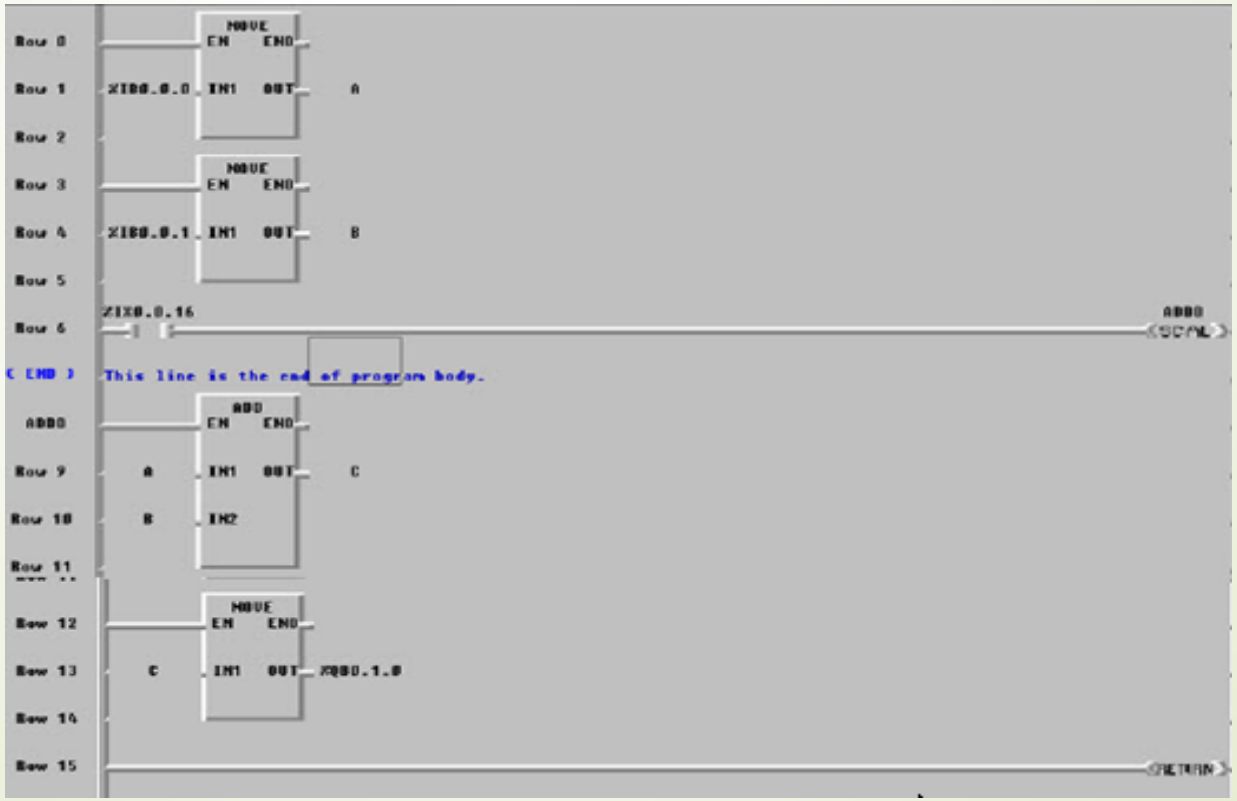



شکل (۵-۱۸)

# فانکشن بلاک ها در استاندارد IEC 61131-3

## دستور SUBROTIN CALL (SCAL)

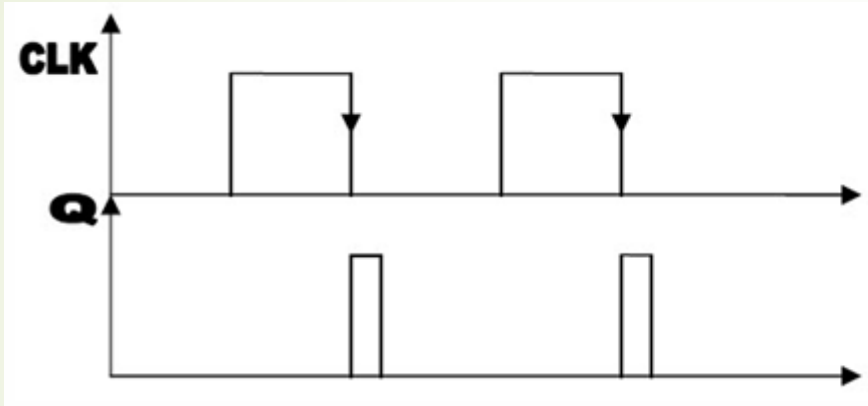
این دستور برنامه ای است که جزو برنامه اصلی نیست و مانند یک تابع فراخوانی می شود ، مثلا اگر بخواهیم برنامه ای بنویسیم که دو عدد را گرفته و زمانی که کلید فشرده شد با فراخوانی یک SCAL آنها را جمع کند . از شکل زیر تبعیت می کنیم.

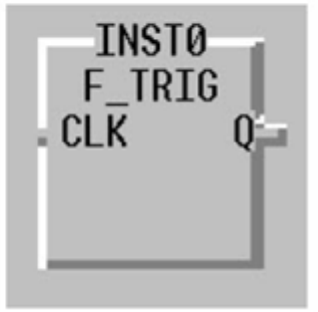



شکل (۵-۱۹)

# فانکشن بلاک ها در استاندارد IEC 61131-3

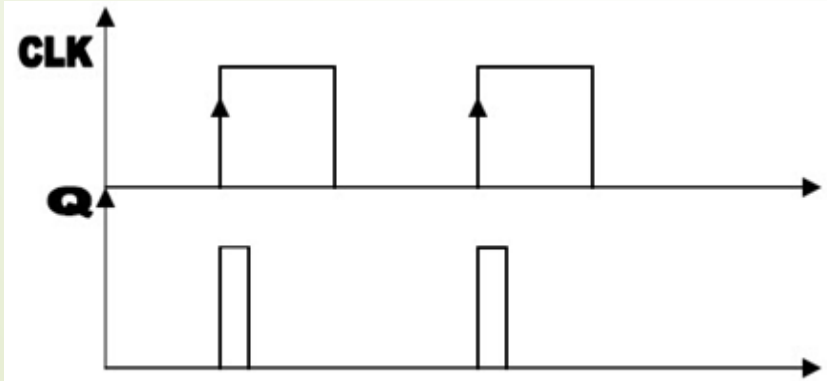
**F-TRIG** : آشکار ساز لبه پایین رونده




شکل (۵-۲۰)

**R-TRIG** : آشکار سازه لبه بالا رونده

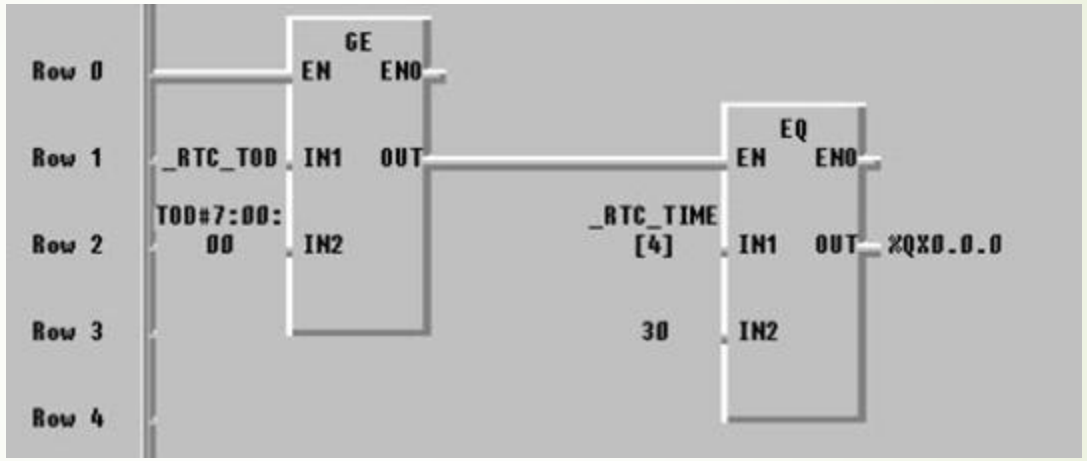




شکل (۵-۲۱)

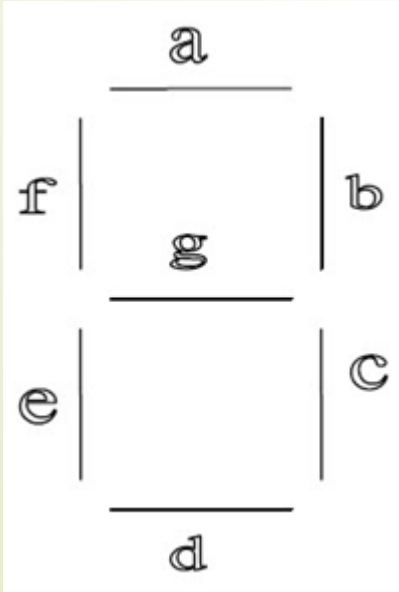
# فانکشن بلاک ها در استاندارد IEC 61131-3

**مثال) برنامه ای بنویسید که هر دقیقه در تمام ساعات مثلا ۷:۳۰ ، ۸:۳۰ و ... به جز ساعت ۱۲ شب و ۷ صبح موتوری را روشن کند.**




**شکل (۵-۲۲)**

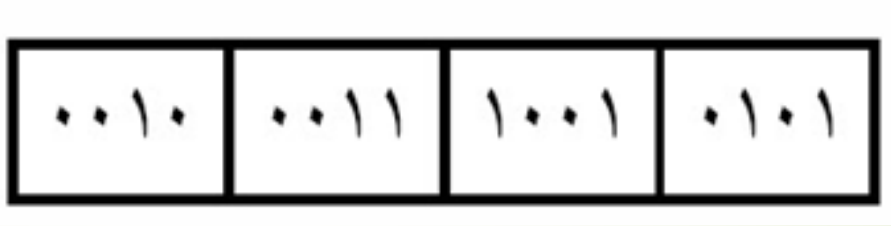
**درایو سون سگمنت:**




**شکل (۵-۲۳)**

# فانکشن بلاک ها در استاندارد IEC 61131-3

به این منظور همواره عددی که قرار است نمایش داده شود را درون یک متغیر از نوع WORD قرار دارد . سپس با استفاده از فانکشن بلاک SEG می توان کد عدد ذخیره شده در متغیر ورودی را بدست آورد. مثلا A=16#1385 به معنای آن است که ورودی A در مبنای ۱۶ بوده و نحوه ذخیره در آن به صورت زیر است:




شکل (۵-۲۴)

فانکشن SINT-TO-BCD:

این فانکشن بای تبدیل SINT به BCD استفاده می شود.

فانکشن DWORD-TO-WORD :

16#5B4E6F6D ⟶ 16#6F6D

# فانکشن بلاک ها در استاندارد IEC 61131-3

این فانکشن بیت کم ارزش **DWORD** را در متغیر **WORD** قرار می دهد.

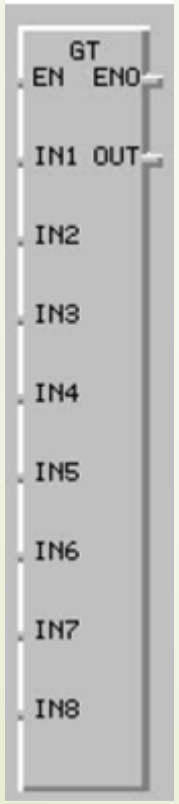
**DWORD-WORD** : **DWORD** را به دوقسمت تقسیم کرده و هر قسمت را داخل یک **WORD** قرار می دهد.

عملگرهای مقایسه کننده:

$$IN1 > IN2 > IN3 > IN4 > IN5 > IN6 > IN7 > IN8$$

در زبان **LAD** و سایر زبان ها عملگر مقاسه ای به شرح ذیل است.

**GT** : بزرگتر از ، **GE** : بزرگتر یا مساوی از ، **LT** : کوچکتر از، **LE** : کوچکتر یا مساوی از ، **EQ** : مساوی و **NE** : نامساوی (تمامی موارد مذکور فانکشن است.) اگر شکل (۵-۲۵) برقرار باشدو **NE** برابر ۱ باشد. آنگاه خروجی **OUT** برابر ۱ و در غیر این صورت برابر صفر است.




شکل (۵-۲۵)