فصل پنجم

شیوه‌های کنترل فرآیند

۵-۱- کنترل فرآیندها

در کنترل فرآیندها معمولاً با دو نوع برنامه مواجه هستیم:

۵-۱-۱- برنامه‌های ترکیبی (Combination Logic)

در این‌گونه برنامه‌ها شرایط موجود در مرحلهٔ فعلی، حرکت به سوی مرحلهٔ بعد را مشخص می‌نماید و هیچ‌گونه اطلاعاتی در مورد انجام مرحلهٔ بعد وجود ندارد، مانند پروسهٔ کنترلی راکتورها، بویلرها و ... . در حقیقت در برنامه‌های ترکیبی، شرایط فعلی موجود، اجرای مرحلهٔ بعد را تعریف می‌نمایند.

۵-۱-۲- برنامه‌های ترتیبی (Sequential Logic)

در این برنامه‌ها انجام یک مرحله، مشروط به انجام مرحله یا مراحل قبلی است یا به طور خلاصه در این برنامه‌ها انجام مراحل به ترتیب خاصی صورت می‌گیرد. مثلاً در سیستم آبکاری فلزات، ابتدا باید زنگ زدایی انجام شود و پس از اتمام این مرحله، مرحلهٔ بعدی انجام گیرد.

در برنامه‌های ترتیبی آغاز مرحله بعدی به دو صورت زیر مشخص می‌گردد.

۱- اتمام یک مرحله آغاز مرحلهٔ بعد را تعریف می‌کند که به این برنامه‌ها Process Dependent گویند.

۲- اتمام زمان انجام یک مرحله، آغاز مرحلهٔ بعد را مشخص می‌کند. به این برنامه‌ها Time - Controlled گویند.

در اکثر پروسه‌های صنعتی تلفیقی از دو برنامهٔ ترکیبی و ترتیبی در کنترل فرآیندها مورد استفاده قرار می‌گیرد.

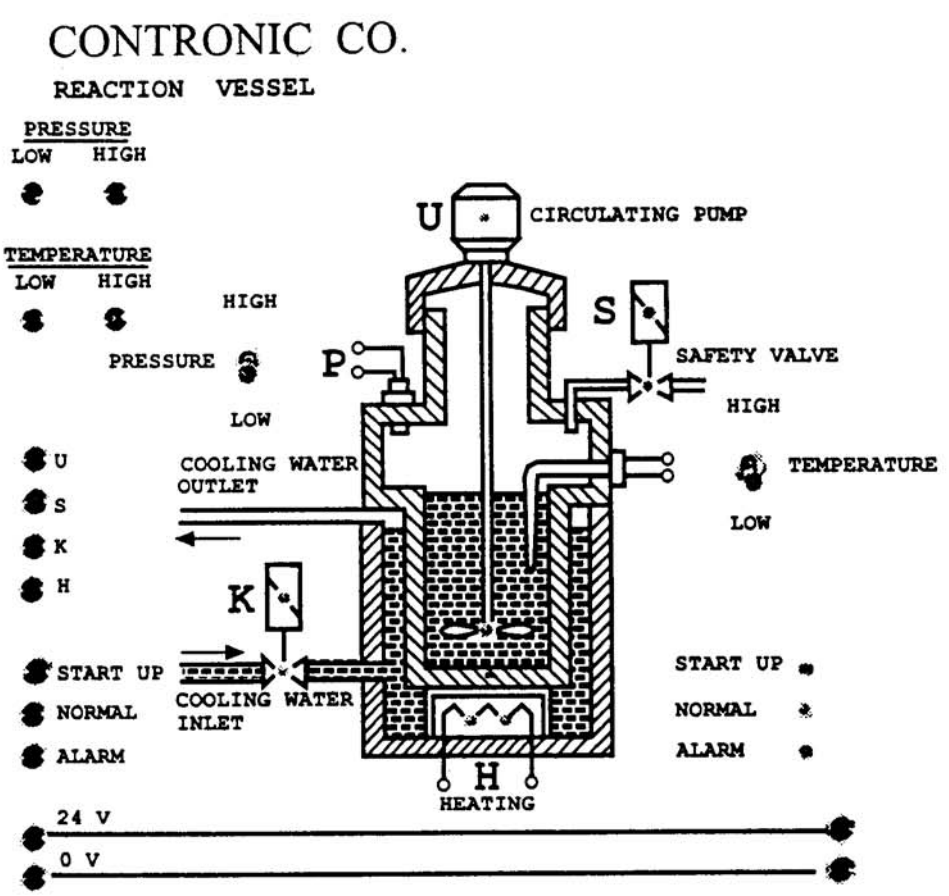
برای روشن شدن مطلب به ذکر یک مثال کاملاً عملی و کاربردی در مورد برنامه‌های ترکیبی می‌پردازیم.

مثال ۵-۱ : در کنترل فرآیندهایی نظیر راکتورها، مخازن تحت فشار کوره‌ها و ... رعایت مسائل ایمنی، صدور علائم خطر و انجام مانورهای مناسب به صورت اتوماتیک علاوه بر عملکرد نرمال سیستم کاملاً ضروری است. شبیه ساز "Reaction Vessel" برای آشنایی با چنین فرآیندهایی طراحی شده است. همان‌گونه که در شکل ۵-۱ دیده می‌شود راکتورهای مواد شیمیایی علاوه بر همزن معمولاً دارای فشارسنج و دماسنج برای اندازه‌گیری فشار و درجه حرارت داخل مخزن، همچنین سیستم گرم کننده و خنک کننده جهت کنترل درجه حرارت و شیر اطمینان جهت کنترل فشار و شرایط ایمنی می‌باشند. علاوه بر موارد مذکور سیستم‌های هشدار دهنده‌ای در راکتورها جهت اعلام نقص‌های احتمالی تعبیه شده‌اند. این سیستم‌ها چگونگی وضعیت عملکرد راکتورها را نشان می‌دهند مانند LEDهای نشان دهنده، آژیر یا ... .

در این مثال قصد داریم برنامهٔ کنترل راکتور را به گونه‌ای بنویسیم که:

۱- چنانچه درجه حرارت یا فشار داخلی سیستم پائین باشد و یا به عبارت دیگر آلارم‌های TEMPERATURE LOW یا PRESSURE LOW وجود داشته باشد سیستم گرم کننده (HEATING) و همزن شروع به کار نموده، LED مربوط به حالت START UP روشن شود.

۲- در صورتی که شرایط نرمال باشد یعنی فشار و درجه حرارت نه بالا و نه پائین باشد یا به عبارت دیگر هیچ یک از آلارم‌های PRESSURE LOW ، PRESSURE HIGH ، TEMPERATURE LOW ، TEMPERATURE HIGH وجود نداشته باشد، تنها LED مربوط به حالت NORMAL روشن شود.



شکل ۵-۱ : طرح ساده‌ای از شبیه‌ساز Reaction Vessel

۳- در صورتی که فشار داخلی سیستم بالا باشد و یا آلارم PRESSURE HIGH وجود داشته باشد شیر اطمینان یا SAFETY VALVE باز شده، LED مربوط به حالت ALARM نیز چشمک بزند.

۴- در صورتی که درجه حرارت داخلی سیستم بالا باشد و یا آلارم TEMPERATURE HIGH وجود داشته باشد فرمان باز شدن ورودی آب خنک کننده (COOLING WATER) صادر شده، LED مربوط به حالت ALARM چشمک بزند و همزن نیز شروع به کار کند.

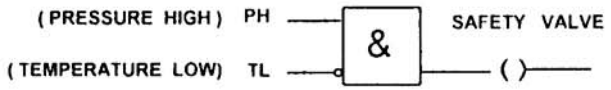
جهت نوشتن این برنامه تمام مراحل ذکر شده در روش برنامه‌نویسی فصل قبل را مرحله به مرحله دنبال می‌کنیم.

۱- **تعریف پروژه**: همانگونه که ذکر شد پروژهٔ کنترل راکتور در ۳ حالت START UP ، NORMAL و ALARM که هر یک معرف وجود شرایط خاصی هستند تعریف شده است.

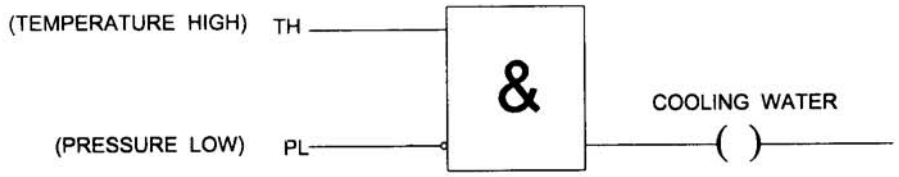
۲- **رسم فلوچارت برنامه**: برای رسم فلوچارت، در نظر گرفتن شرایط ایمنی و اعمال آنها در فلوچارت الزامی است. در رسم فلوچارت، ورودی‌ها و خروجی‌ها بر اساس اسامی آنها در سیستم نامگذاری شده‌اند. به عنوان مثال ورودی TH معرف حالت TEMPERATURE HIGH می‌باشد. به دلیل این که در رسم فلوچارت برنامه ملزم به در نظر گرفتن شرایط ایمنی هستیم لذا مرحلهٔ پنجم برنامه‌نویسی راکه همان بررسی شرایط ایمنی است در این قسمت توضیح خواهیم داد. اکنون شرایط ایمنی را با ذکر چند سؤال بررسی می‌کنیم.

سؤال اول: شیر اطمینان در چه شرایطی عمل می‌کند؟

افرادی که محیط‌های صنعتی و فرآیندهای کوچک و بزرگ را تجربه نموده‌اند به خوبی می‌دانند که شیر اطمینان در حالتی که فشار داخلی بالا بوده، درجه حرارت داخلی سیستم نیز پائین نباشد عمل می‌کند. زیرا در حالتی که فشار داخلی سیستم بالا می‌رود درجه حرارت نیز به نوبهٔ خود افزایش می‌یابد و در صورتی که در این حالت درجه حرارت داخلی سیستم پائین باشد باید به صحت عملکرد سنسورهای درجه حرارت شک نمود. بنابراین شرایط لازم جهت عملکرد شیر اطمینان را می‌توان به صورت زیر نشان داد:



سیستم آب خنک کننده نیز هنگامی شروع به کار می‌کند یا به عبارت دیگر فرمان باز شدن شیر آب خنک کننده هنگامی صادر می‌شود که درجه حرارت داخل سیستم بالا بوده، فشار داخلی نیز پائین نباشد.



شاید در ذهن خوانندگان این سؤال مطرح شود که ارتباط بین شروع به کار سیستم آب خنک‌کننده و پائین نبودن فشار داخل سیستم چیست؟ و یا به عبارت دیگر، آیا وجود شرط درجه حرارت بالا به تنهایی برای شروع به کار سیستم مذکور کافی است؟

در پاسخ به این سؤال باید گفت که بالا بودن درجه حرارت داخل سیستم همراه با بالا رفتن فشار داخلی آن می‌باشد و در صورت داشتن درجه حرارت بالا و فشار پائین احتمال معیوب بودن یکی از دو سنسور فشار و دما وجود دارد.

توجه داشته باشید که در دو مورد فوق یعنی باز شدن شیر اطمینان و به کار افتادن سیستم خنک‌کننده شرایط ایمنی نظیر احتمال معیوب بودن سنسورهای فشار و درجه حرارت نیز در نظر گرفته شده است.

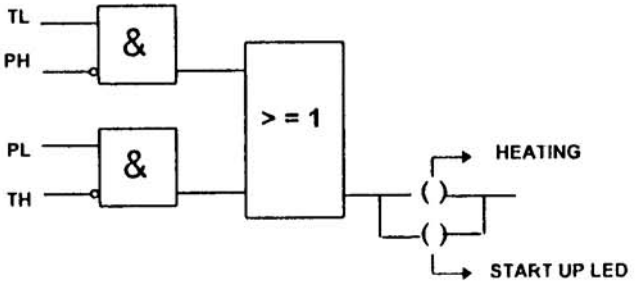
سؤال دوم: شرایط لازم جهت به کار افتادن سیستم گرم کننده (HEATING) چیست؟

سیستم گرم کننده در صورت برقراری لااقل یکی از دو شرط زیر به کار می‌افتد.

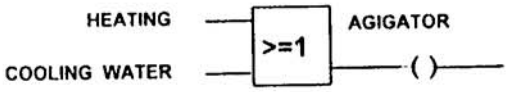
۱- در صورتی که درجه حرارت پائین بوده، فشار بالا نباشد. زیرا داشتن درجه حرارت پائین به همراه فشار بالا دو شرط متضاد و متناقض یکدیگرند که وجود یکی، عدم وجود دیگری را ثابت می‌کند. بنابراین در این حالت نیز احتمال معیوب بودن سنسورها در نظر گرفته شده است.

۲- در حالتی که فشار پائین بوده، درجه حرارت بالا نباشد. در این حالت نیز نظیر حالت قبلی احتمال معیوب بودن سنسورها در نظر گرفته شده است، زیرا در صورتی که فشار پائین باشد به طور منطقی نباید انتظار بالا بودن درجه حرارت را داشته باشیم. پس در صورتی که فشار پائین و درجه حرارت بالا باشد احتمالاً یکی از سنسورها معیوب است.

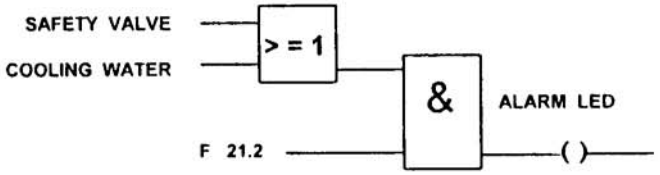
شرایط لازم جهت به کار افتادن سیستم گرم کننده در فلوچارت زیر نشان داده شده است. از آنجایی که به همراه روشن شدن سیستم گرم کننده لازم است که LED مربوط به حالت START UP نیزروشن شود فرمان روشن‌شدن این LED درفلوچارت زیر در نظر گرفته شده است.



روشن شدن همزن الکتریکی مستلزم روشن بودن و یا در حال کار بودن سیستم گرم کننده یا سیستم خنک کننده می‌باشد زیرا در حالتی که سیستم گرم کننده در حال کار باشد روشن بودن همزن الکتریکی جهت یکسان سازی درجه حرارت مواد داخل سیستم الزامی است. در مورد سیستم خنک‌کننده نیز همین استدلال را می‌توان بیان نمود.



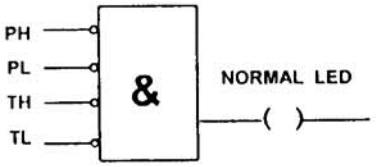
شرایط لازم برای چشمک زدن LED مربوط به حالت ALARM در فلوچارت زیر آمده است.



همانگونه که در فلوچارت فوق دیده می‌شود LED مربوط به حالت ALARM هنگامی چشمک می‌زند که شیر اطمینان باز شده باشد یا اینکه سیستم خنک کننده در حال کار باشد. F 21.2 سرعت چشمک زدن LED مربوطه را معین می‌کند. (در بلوک OB 1 در مورد این فلگ و چگونگی تعیین سرعت چشمک زدن به تفصیل سخن خواهیم گفت.)

سؤال سوم: شرایط لازم برای وضعیت NORMAL چیست؟

- در وضعیت NORMAL نباید فشار و درجه حرارت بالا یا پائین باشد به عبارت دیگر دما و فشار باید در محدودهٔ قابل قبول و تعریف شده برای سیستم وجود داشته باشد. در فلوچارت زیر این شرایط نشان داده شده‌اند.



همان‌گونه که قبلاً نیز ذکر شد در تمامی حالات فوق احتمال معیوب بودن سنسورهای درجه حرارت و فشار در نظر گرفته شده است.

از آنجایی که سیستم راکتورهای شیمیایی واقعاً حساس بوده و کنترل این‌گونه سیستم‌ها از اهمیت بالایی برخوردار است در برنامه‌نویسی بایستی احتمال معیوب بودن سنسورها نیز در نظر گرفته شود و برنامه به شیوه‌ای نوشته شود که در صورت معیوب بودن حتی یک سنسور، هیچ عملی انجام نشود.

۳- تهیه لیستی از ابزار مورد نیاز: جهت وارد نمودن اطلاعات در مورد ورودی‌ها و خروجی‌ها از عملوندهای زیر استفاده می‌کنیم:

ورودی‌ها:
- PH : I 21.0
- TL : I 21.1
- TH : I 21.2
- PL : I 21.3

خروجی‌ها:
- SAFETY VALVE : Q 14.0
- COOLINGWATER : Q 14.1
- HEATING : : Q 14.2
- START UP LED : Q 14.3
- AGIGATOR : Q 14.4
- ALARM LED : Q 14.5
- NORMAL LED : Q 14.6

۴- نوشتن برنامه به یکی از سه روش برنامه‌نویسی: جهت نوشتن این برنامه از روش CSF استفاده شده است.

به دلیل اینکه کنترل راکتور شامل چندین مرحله است برنامهٔ نوشته شده در بلوک PB 20 شامل ۶ قسمت (SEGMENT) می‌باشد. برای اجرای این برنامه نیاز به تعریف بلوک OB 1 داریم و برای تعیین سرعت چشمک زدن LED هشدار دهنده از بلوک FB 100 که در مثال ۴-۲۰ برنامه‌نویسی شد استفاده می‌کنیم.

```
 OB 1

SEGMENT  1          0000
0000       :JU  FB 100
0001 NAME  :BLINK
0002 TIM   :     T     8
0003 DATA  :     KT 050.1
0004       :JU  PB  20
0005       :BE
```

```
 FB 100

SEGMENT  1          0000
NAME :BLINK
DECL :TIM        I/Q/D/B/T/C: T
DECL :DATA       I/Q/D/B/T/C: D  KM/KH/KY/KS
                                /KF/KT/KC/KG: KT
000B       :AN   =TIM
000C       :LW   =DATA
000D       :SEC  =TIM
000E       :L    =TIM
000F       :T    FW  20
0010       :BE
```

```
 PB 20

SEGMENT  1          0000
              +---+
 I 21.0    ---! & !     +------+
 I 21.1    --O!   !--+-! =    ! Q 14.0
              +---+     +------+
```

```
SEGMENT  2          0004
          +---+
 I 21.2  ---! & !    +------+
 I 21.3  --O!   !--+-! =    ! Q 14.1
          +---+    +------+

SEGMENT  3          0008
          +---+
 I 21.1  ---! & !     +---+
 I 21.0  --O!   !-----!>=1!
          +---+      !   !
          +---+      !   !
 I 21.3  ---! & !    !   !    +------+
 I 21.2  --O!   !----!   !--+-! =    ! Q 14.2
          +---+     +---+  ! +------+
                           ! +------+
                          +-! =    ! Q 14.3
                            +------+

SEGMENT  4          0010
          +---+
 Q 14.2  ---!>=1!    +------+
 Q 14.1  ---!   !--+-! =    ! Q 14.4
          +---+    +------+

SEGMENT  5          0014
          +---+
 Q 14.0  ---!>=1!     +---+
 Q 14.1  ---!   !-----! & !
          +---+      !   !    +------+
           F 21.2  ---!   !--+-! =    ! Q 14.5
                      +---+    +------+

SEGMENT  6          001B
          +---+
 I 21.0  --O! & !
 I 21.1  --O!   !
 I 21.2  --O!   !    +------+
 I 21.3  --O!   !--+-! =    ! Q 14.6
          +---+    +------+:BE
```

روند اجرای برنامه به صورت زیر است:

سطر اول در OB 1 دستور پرش به برنامه بلوک FB 100 را صادر می‌کند. سپس در همین بلوک نام برنامه و شمارهٔ تایمری که در برنامه‌های دیگر استفاده نشده است از کاربر خواسته می‌شود. در قسمت DATA، کاربر، زمان تایمر و همچنین تولرانس و دقت چشمک زدن را مشخص می‌کند. همان‌گونه که می‌دانید ضریب زمانی ۱ سبب می‌شود که کم ارزش‌ترین بیت FW 20 یعنی F 21.0 با سرعت 0.1 ثانیه و F 21.1 با دو برابر سرعت بیت قبلی خود یعنی F 21.0 چشمک بزند و به همین ترتیب الی آخر. در این مثال از F 21.2 جهت تعیین سرعت چشمک زدن LED مربوط به حالت ALARM استفاده شده است یعنی سرعت چشمک زدن این LED، ۰/۴ ثانیه است. در سطر پنجم فرمان پرش به PB 20 صادر شده است. از این پس پردازنده دستورات موجود در PB 20 را سطر به سطر اجرا نموده، به محض رسیدن به انتهای بلوک به OB 1 باز می‌گردد و دستور BE موجود در OB 1 اتمام اجرای برنامه را اعلام می‌کند.

جهت ادامهٔ بحث نیاز به تعریف یک دستور جدید داریم.

۵-۲- دستور DO

این دستور یکی از دستورات تکمیلی است که تنها در FBها قابل استفاده است. این دستور را می‌توان در مورد DWها و FWها اعمال نمود.

اکنون فرض کنید قصد داریم در یک بلوک DB که شامل ۱۰۰ کلمه (DW) می‌باشد دستورات زیررا اجراکنیم. به عبارت دیگر محتویات تمام کلمه‌های موجود در این DB را با استفاده از دستور L بارگذاری کنیم.

```
: L  DW  0
: L  DW  1
: L  DW  2
     ⋮
: L  DW  99
```

جهت جلوگیری از طولانی شدن برنامه و همچنین صرفه‌جویی در زمان از دستور DO استفاده می‌کنیم. بدین ترتیب که آدرس اولین DW را خوانده، سپس با آدرس دهی‌های پیاپی اطلاعات موجود در این DWها را بارگذاری می‌کنیم. به این نوع آدرس دهی، آدرس‌دهی غیرمستقیم (Indirect Addressing) گویند.

مثال ۵-۲: قصد داریم در DB 30 و در کلمه‌های DW 15 الی DW 45 عدد صفر ($0000_{(H)}$) را بارگذاری کنیم. در صورتی که از دستور L و T استفاده کنیم برنامهٔ مذکور به این صورت خواهد بود:

```
 FB 5

SEGMENT  1              0000
NAME :ZERO

0005        :C    DB  30
0006        :L    KF +0
0008        :T    DW  15
0009        :T    DW  16
000A        :T    DW  17
000B        :      :
000C        :      :
000D        :T    DW  44
000E        :T    DW  45
000F        :BE
```

ملاحظه می‌کنید که اجرای این برنامه در مورد بارگذاری ۳۰ کلمه تا چه اندازه طولانی شده است. واضح است که برای مقادیر بالاتر، برنامه طولانی‌تر خواهد شد.

حال با استفاده از دستور DO این عمل را انجام می‌دهیم. این برنامه در FB 6 به صورت زیر نوشته شده است.

```
 FB 6

SEGMENT  1              0000
NAME :DO

0005        :C    DB  30
0007        :L    KF +15
0008 M001   :T    FW  30
000A        :L    KF +0
000B        :DO   FW  30
000C        :T    DW   0
000D        :L    FW  30
000E        :I         1
000F        :T    FW  30
0011        :L    KF +45
0012        :<=F
0013        :JC   =M001
0014        :BE
```

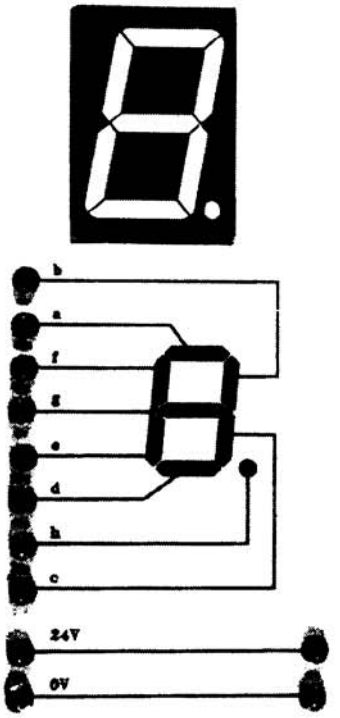
در سطر اول این برنامه از دستور C DB 30 به منظور صدا زدن و معتبر نمودن DB 30 استفاده شده است و این بدان معنی است که از این پس تمام اعمال L و T بر روی DWهای موجود در این DB انجام می‌گیرد. در سطر دوم و سوم با استفاده از دو دستور L و T عدد دهدهی ۱۵ در FW 30 قرار می‌گیرد. عدد ۱۵ آدرس شروع کلمه‌هایی است که قرار است بر روی آنها عملیاتی انجام گیرد. در سطر چهارم عدد دهدهی ۰ در انبارک بارگذاری می‌شود. در دستور DO FW 30، محتوای FW 30 (یا هر FW دیگری که بعد از دستور DO قرار می‌گیرد) آدرس کلمه‌ای از DB خواهد بود که باید اطلاعات به آنجا ارسال شوند. در دستور بعدی یعنی T DW 0 محتویات کلمهٔ شمارهٔ ۱۵ به کلمهٔ شماره ۰ انتقال می‌یابد. در دستور L FW 30 و I 1 به عدد ثابت ۱۵، یک واحد اضافه شده، حاصل مجدداً به FW 30 انتقال می‌یابد. سپس عدد حاصل با عدد ۴۵ مقایسه می‌شود و در صورتی که از ۴۵ کوچکتر و یا با آن مساوی باشد اجرای برنامه از سطری که با برچسب M001 مشخص شده، دنبال می‌گردد. اجرای این برنامه همچنان ادامه می‌یابد تا اینکه محتوای FB 30 که آدرس کلمه‌ای از DB است از ۴۵ تجاوز نماید در این حالت اجرای برنامه پایان می‌یابد و مجدداً از ابتدای برنامه، سیکل مذکور تکرار می‌شود.

همانگونه که ملاحظه نمودید با استفاده از دستور DO آدرس خطها یا کلمه‌های موجود در DB را تغییر داده، سپس دستورات L و T را بر روی آنها انجام می‌دهیم. در این قسمت به ذکر یک مثال کاملاً کاربردی و عملی در مورد چگونگی استفاده از دستور DO می‌پردازیم.

مثال ۵-۳: اکثر خوانندگان با LEDهای نورانی که هفت قسمتی بوده و Seven Segment نامیده می‌شوند آشنایی دارند. شکل ۵-۲ را که در واقع شبیه‌ساز طراحی شده برای برنامه‌نویسی این مثال می‌باشد در نظر بگیرید.

در این مثال قصد داریم برنامه‌ای بنویسیم که با استفاده از آن، اعداد ۰ تا ۹ با فاصلهٔ زمانی ۰/۸ ثانیه بر روی Seven Segment به نمایش در آیند. در گوشهٔ سمت راست Seven Segment یک نقطه (Point) نیز وجود دارد که با نمایش هر عدد، این نقطه باید روشن باشد.

با اندکی دقت در می‌یابیم که برای نوشتن این برنامه کافی است اعداد ۰ تا ۹ را به معادل هگزادسیمال تبدیل نموده، آنها را با فاصلهٔ زمانی ۰/۸ ثانیه بر روی صفحهٔ نمایش هفت قسمتی بفرستیم. واضح است که معادل هگز این اعداد باید در یک بلوک DB تعریف شوند.



شکل ۵-۲ : شمای یک نمایش دهندهٔ هفت قسمتی (7 Segment)

از آنجایی که برای روشن شدن هر یک از قسمتهای Seven Segment باید آنها را به یکی از خروجی‌های PLC نسبت دهیم معادل با هر قسمت، یکی از خروجی‌های 14 QB را که در ادامه نشان داده شده است در نظر می‌گیریم.

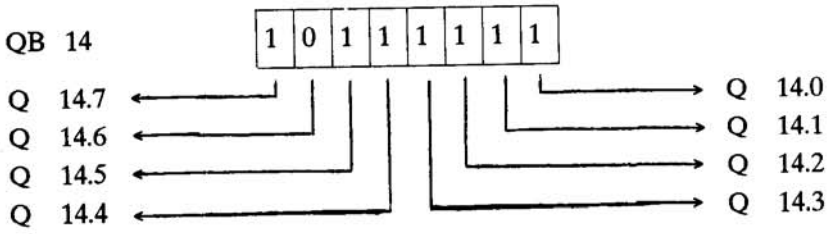
a ≡ Q 14.0	e ≡ Q 14.4
b ≡ Q 14.1	f ≡ Q 14.5
c ≡ Q 14.2	g ≡ Q 14.6
d ≡ Q 14.3	h ≡ Q 14.7

نسبت دادن هر یک از Segmentها به یک بیت در بایت خروجی ۱۴ بدین معنی است که به عنوان مثال برای روشن شدن قسمت a در Seven Segment باید خروجی Q 14.0 فعال شود و

برای روشن شدن Point کافی است که "۱" = Q 14.7 و ... . اکنون معادل هگز هر یک از اعداد ۰ تا ۹ را به دست می‌آوریم. برای مثال عدد ۰ را در نظر می‌گیریم. این عدد در این نمایش دهنده به صورت .0 به نمایش در می‌آید. برای نمایش این عدد کافی است که بیت‌های خروجی QB 14 به صورت زیر باشند:

a ≡ Q 14.0 = "۱"		e ≡ Q 14.4 = "۱"	
b ≡ Q 14.1 = "۱"		f ≡ Q 14.5 = "۱"	
c ≡ Q 14.2 = "۱"		g ≡ Q 14.6 = "۰"	
d ≡ Q 14.3 = "۱"		h ≡ Q 14.7 = "۱"	

پس برای نمایش .0 بر روی Seven Segment کافی است که QB 14 به صورت زیر باشد.

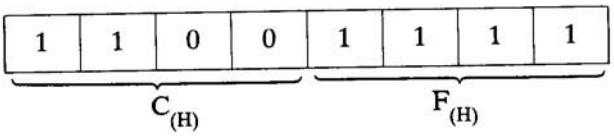


سپس مقدار موجود در این کلمهٔ خروجی یعنی $۱۰۱۱۱۱۱۱_{(۲)}$ را به مبنای ۱۶ تبدیل می‌کنیم.

$۱۰۱۱\ ۱۱۱۱_{(۲)} = BF_{(H)}$

بنابراین برای نمایش .0 کافی است که عدد $BF_{(H)}$ را در DW 0 قرار داده، سپس آن را به خروجی بفرستیم. برای نمایش .3 باید ارزش بیت‌های QB 14 به صورت زیر باشد:

a ≡ Q 14.0 = "۱"		e ≡ Q 14.4 = "۰"	
b ≡ Q 14.1 = "۱"		f ≡ Q 14.5 = "۰"	
c ≡ Q 14.2 = "۱"		g ≡ Q 14.6 = "۱"	
d ≡ Q 14.3 = "۱"		h ≡ Q 14.7 = "۱"	



به عبارت دیگر می‌توان مقدار CF_H را در DW 3 قرار داده، سپس آن را به خروجی فرستاد.
یافتن معادل هگز سایر اعداد را به عنوان تمرین به خواننده واگذار می‌کنیم.
مقادیر معادل هگز نمایش اعداد .0 تا .9 در ادامه آمده است.

0. ≡ $BF_{(H)}$	5. ≡ $ED_{(H)}$
1. ≡ $86_{(H)}$	6. ≡ $FD_{(H)}$
2. ≡ $DB_{(H)}$	7. ≡ $87_{(H)}$
3. ≡ $CF_{(H)}$	8. ≡ $FF_{(H)}$
4. ≡ $E6_{(H)}$	9. ≡ $EF_{(H)}$

برای تکمیل برنامه احتیاج به یک شمارنده داریم به طوری که هر ۰/۸ ثانیه یک بار عدد نمایش داده شده را یک واحد افزایش دهد. در مثال قبل دیدیم که با استفاده از دستور DO، آدرس‌ها را با دستور I 1، یک واحد افزایش دادیم. در اینجا مجاز به استفاده از دستور I 1 نیستیم زیرا سرعت انتقال اطلاعات به خروجی و تغییر وضعیت صفحهٔ نمایش هفت قسمتی به قدری سریع است که چشم انسان قدرت تشخیص و تمییز اعداد نمایش داده شده را از یکدیگر ندارد، چراکه عمل انتقال اطلاعات و تغییر وضعیت صفحهٔ نمایش با سرعت Cycle Time و در حدود چند میلی ثانیه انجام می‌گیرد. برای رفع این مشکل از برنامهٔ چشمک‌زن نوشته شده در FB 100 استفاده می‌کنیم.

همان‌گونه که قبلاً توضیح داده شد برنامهٔ FB 100 یکی از پرکاربردترین برنامه‌ها است که ابتدا شمارهٔ تایمری که در برنامه‌های دیگر استفاده نشده، سپس سرعت انتقال اطلاعات یا تولرانس تایمر را از کاربر سؤال می‌کند و استفاده‌کننده می‌تواند با انتخاب هر یک از بیت‌های FW 20 تعریف شده در FB 100 سرعت چشمک زدن را تغییر دهد. بنابراین برای نوشتن این برنامه ابتدا یک DB ایجاد نموده، معادل هگز تمام اعداد ۰ تا ۹ را در آن قرار می‌دهیم. سپس با تعریف یک FB مثلاً FB 35، DB ایجاد شده را معتبر می‌نماییم و با استفاده از دستور DO و آدرس‌دهی غیرمستقیم، DWهای موجود را به خروجی می‌فرستیم. در OB 1 که ساختار کلی برنامه را مشخص می‌کند دستور پرش به FB 35 را صادر و سپس با فراخوانی FB 100 سرعت چشمک زدن را مشخص می‌کنیم. برنامهٔ نوشته شده در ادامه آورده شده است.

(در اینجا از نوشتن FB 100 خودداری شده است زیرا همان‌طور که گفته شد بلوک‌های FB را تنها یک بار نوشته، در صورت نیاز، با فراخوانی آنها مقادیر پارامتر را وارد می‌کنیم.)

```
 OB 1

SEGMENT  1              0000
0000       :JU  FB  35
0001 NAME  :SEGMENT
0002       :JU  FB 100
0003 NAME  :BLINK
0004 TIM   :    T    8
0005 DATA  :    KT 050.1
0006       :BE


 FB 35

SEGMENT  1              0000
NAME :SEGMENT

0005       :C   DB  31
0006       :A   F   21.3
0007       :CU  C    5
0008       :L   C    5
0009       :T   FW  30
000A       :DO  FW  30
000B       :L   DW   0
000C       :T   QB  14
000D       :L   FW  30
000E       :L   KF +10
0010       :>=F
0011       :R   C    5
0012       :BE


DB31

    0:      KH = 00BF;
    1:      KH = 0086;
    2:      KH = 00DB;
    3:      KH = 00CF;
    4:      KH = 00E6;
    5:      KH = 00ED;
    6:      KH = 00FD;
    7:      KH = 0087;
    8:      KH = 00FF;
    9:      KH = 00EF;
   10:      KH = 0000;
```

ممکن است در ذهن خواننده این سؤال مطرح شود که دلیل استفاده از KH 00 در سطر یازدهم DB 31 چیست؟

پاسخ به این سؤال بسیار ساده است. در صورتی که DW 10 که همان KH 00 است در سطر آخر DB 31 گنجانده نشود آخرین عدد نمایش داده شده، با سرعت سیکل زمانی برنامه تغییر می‌کند. در این حالت قادر به مشاهدهٔ این عدد نخواهیم بود، اما با مقایسهٔ DW 10 برای فاصلهٔ زمانی یک سیکل می‌توان آخرین عدد را که 9 می‌باشد به مدت ۰/۸ ثانیه مشاهده نمود.

۵-۳- ارسال پیام‌های خطا بر روی صفحهٔ نمایش

افرادی که با سیستم‌های کنترل مونیتورینگ و پیشرفته آشنایی دارند به خوبی می‌دانند که در سیستم‌های کنترل امروزی پیامهای خطا و هشدار بر روی صفحهٔ مونیتور به نمایش در می‌آیند. این پیامها به حالت چشمک‌زن و یا ثابت بر روی صفحهٔ مونیتور ظاهر می‌شوند. همان‌گونه که در فصول گذشته ذکر شد پیامهای خطا و هشدار در بلوک DB نوشته شده است و به هنگام ایجاد هر گونه اشکال با آدرس‌دهی مناسب می‌توان با معتبر نمودن DB مذکور و بارگذاری DWهای مربوطه پیام دلخواه و متناسب با مشکل ایجاد شده در سیستم را در نقاط مختلف صفحهٔ نمایش ظاهر ساخت. در مثال زیر نحوهٔ ارسال اطلاعات و پیامها به طور مفصل آمده است.

مثال ۵-۴: فرض کنید که در یک سیستم کنترلی ۴ پیام و یا هشدار وجود دارد (خوانندگان می‌دانند که در سیستم‌های کنترلی عظیم ممکن است هزاران پیام بر روی صفحهٔ مونیتور ظاهر شود در این مثال برای جلوگیری از طولانی شدن برنامه تنها ۴ پیام برای سیستم در نظر گرفته شده است.) این پیامها در ۴ سطر از DB 32 نوشته شده‌اند. همان‌طور که قبلاً اشاره شد پیامها را با فرمت KS نمایش می‌دهیم. از آنجایی که پیامهای قابل نمایش بر روی صفحهٔ مونیتور در PLC زیمنس نباید از ۱۲ کلمه (۱۲ کلمه = ۲۴ بایت) تجاوز نماید بنابراین در ستون اول DB 32 که معرف شماره بایت‌های اشغال شده توسط پیامهاست مضاربی از عدد ۱۲ دیده می‌شود. در پیامها فاصلهٔ بین دو کلمه (Blank Space) نیز یک کاراکتر محسوب می‌شود. به دلیل اینکه هر کاراکتر مطابق کد ASCII ، ۸ بیت یا یک بایت از حافظه را اشغال می‌کند پس می‌توان از ۱۲ کلمه (۲۴ بایت) یا به عبارت دیگر ۲۴ کاراکتر در هر سطر استفاده نمود. در DB 32 سطرهای ۱ الی ۴ به پیامها اختصاص

داده شده است. برای مقاصد بعدی برنامه در سطرهای بعد، از کاراکترهای 0 استفاده شده است. در DB 32 چگونگی آرایش کاراکترها و پیامها را در این بلوک مشاهده می‌کنید.

```
DB32

  0:      KS =' tanks levels are stable';
 12:      KS =' motor no-6 is defective';
 24:      KS ='fan is out of order now ';
 36:      KS ='temperature is too high ';
 48:      KS ='000000000000000000000000';
 60:      KS ='000000000000000000000000.'
```

اکنون بلوک DB 33 را در نظر بگیرید که تمام سطرهای آن با کاراکترهای 0 پر شده‌اند.

```
DB33

  0:      KS ='000000000000000000000000';
 12:      KS ='000000000000000000000000';
 24:      KS ='000000000000000000000000';
 36:      KS ='000000000000000000000000';
 48:      KS ='000000000000000000000000';
 60:      KS ='000000000000000000000000';
```

اکنون ۴ ورودی را که به عنوان مثال سیگنال‌های ارسال شده از سنسورهای اندازه‌گیری فشار، درجه حرارت، لرزش و ... هستند در نظر بگیرید. در صورت فعال شدن هر ورودی یکی از پیامها (پیام متناظر با هر سنسور) از DB 32 به DB 33 انتقال می‌یابد.

نحوهٔ انتقال به گونه‌ای است که هر سطر پیام از DB 32 به سطر متناظر خود در DB 33 منتقل می‌شود.به عنوان مثال در صورت فعال شدن ورودی ناشی از سیگنال سنسور درجه حرارت، پیام "temperature is too high" از سطر چهارم DB 32 به سطر چهارم DB 33 انتقال می‌یابد. این پیام در DB 33 به جای کاراکترهای 0 قرار می‌گیرد. در حالت کنترل مونیتورینگ می‌توان تغییرات DB 33 را بر روی صفحهٔ نمایش مشاهده نمود و در صورت ارسال پیام و یا هشدار به رفع اشکال

سیستم پرداخت. در این مثال قصد داریم تا برنامه‌ای بنویسیم که این عمل یعنی انتقال اطلاعات پیام و علائم هشداردهنده را از DB 32 به DB 33 انجام دهد. البته انتقال و ارسال پیامها منوط به فعال شدن ورودی متناظر با هر پیام است.

با اندکی دقت در متن مثال در می‌یابیم که با استفاده از دستور DO می‌توان این برنامه را به راحتی نوشت. در حقیقت، فعال شدن هر ورودی می‌تواند مشخص کنندهٔ آدرس شروع بایت‌های اشغال شده توسط پیام متناظر با آن ورودی باشد. از آنجایی که استفاده از دستور DO تنها در FB مجاز است بنابراین برنامهٔ اصلی را در یک بلوک FB می‌نویسیم.

```
 FB 35

SEGMENT  1            0000
NAME :MESSAGE1

0005        :A    I    16.0      سنسور مربوط به پیام اول
0006        :JC   =M001
0007        :A    I    16.1      سنسور مربوط به پیام دوم
0008        :JC   =M002
0009        :A    I    16.2      سنسور مربوط به پیام سوم
000A        :JC   =M003
000B        :A    I    16.3      سنسور مربوط به پیام چهارم
000C        :JC   =M004
000D        :JU   =M005
000E        :BEU
000F M005   :C    DB   32        DB اولی که حاوی پیام‌هاست.
0010        :DO   FW   30        آدرس‌دهی غیرمستقیم
0011        :L    DW    0
0012        :C    DB   33        DB دومی که حاوی کاراکترهای 0 است.
0013        :DO   FW   40        آدرس‌دهی غیرمستقیم
0014        :T    DW    0
0015        :L    FW   30
0016        :I          1
0017        :T    FW   30
0018        :L    FW   40
0019        :I          1
001A        :T    FW   40
001B        :L    FW   30
001C        :L    KF  +12        انتهای ۱۲ کلمهٔ اول
001E        :!=F
001F        :JC   =M001
```

```
0020        :L    FW  30
0021        :L    KF +24        انتهای ۱۲ کلمهٔ دوم
0023        :!=F
0024        :JC   =M002
0025        :L    FW  30
0026        :L    KF +36        انتهای ۱۲ کلمهٔ سوم
0028        :!=F
0029        :JC   =M003
002A        :L    FW  30
002B        :L    KF +48        انتهای ۱۲ کلمهٔ چهارم
002D        :!=F
002E        :JC   =M004
002F        :BEU
0030 M001   :L    KF +0         آدرس شروع پیام اول
0032        :T    FW  30
0033        :T    FW  40
0034        :BEU
0035 M002   :L    KF +12        آدرس شروع پیام دوم
0037        :T    FW  30
0038        :T    FW  40
0039        :BEU
003A M003   :L    KF +24        آدرس شروع پیام سوم
003C        :T    FW  30
003D        :T    FW  40
003E        :BEU
003F M004   :L    KF +36        آدرس شروع پیام چهارم
0041        :T    FW  30
0042        :T    FW  40
0043        :BE
```

باید توجه داشت که در صورت انتخاب آدرس شروع هر پیام نیاز به برنامه‌ای است که از آدرس شروع، ۲۴ کاراکتر را دریافت و به DB 33 ارسال نماید. قسمتی از برنامه که با برچسب M005 مشخص شده است هر بار که آدرس شروع پیام را دریافت کند از آن آدرسی را با ۱۲ کلمه فراخوانده، انتهای هر ۱۲ کلمه را نیز با استفاده از مقایسه‌گرهای برنامه محاسبه و مشخص می‌کند. در ضمن اگر پیام در صفحهٔ مونیتور سیستم کنترل مونیتورینگ فقط یک بار به مونیتور فرستاده شود به دلیل سرعت پردازنده قابل مشاهده نخواهد بود. بنابراین می‌بایست عمل دریافت و ارسال اطلاعات از DBها تا رفع عیب و اشکال به صورت مداوم و پیوسته بر روی مونیتور ارسال شود.

به دلیل اینکه قسمت خاصی از صفحهٔ مونیتور به نمایش پیامها و علائم هشدار دهنده اختصاص داده شده است و در هر قسمت از مونیتور نمی‌توان پیامهای خطا را به نمایش در آورد. می‌خواهیم با ایجاد تغییراتی در برنامه با فعال و غیرفعال شدن هر سنسور پیغام متناظر با آن سنسور فقط به سطر اول از DB 33 منتقل شود. این برنامه در FB 36 به صورت زیر نوشته شده است.

```
 FB 36

SEGMENT  1           0000
NAME :MESSAGE1

0005       :A    I   16.0
0006       :JC   =M001
0007       :A    I   16.1
0008       :JC   =M002
0009       :A    I   16.2
000A       :JC   =M003
000B       :A    I   16.3
000C       :AN   I   16.0
000D       :AN   I   16.1
000E       :AN   I   16.2
000F       :JC   =M004
0010       :JU   =M005
0011       :BEU
0012 M005  :C    DB  32
0013       :DO   FW  30
0014       :L    DW   0
0015       :C    DB  33
0016       :DO   FW  40
0017       :T    DW   0
0018       :L    FW  30
0019       :I         1
001A       :T    FW  30
001B       :L    FW  40
001C       :I         1
001D       :T    FW  40
001E       :L    FW  30
001F       :L    KF +12
0021       :!=F
0022       :JC   =M001
0023       :L    FW  30
0024       :L    KF +24
```

```
0026        :!=F
0027        :JC   =M002
0028        :L    FW  30
0029        :L    KF +36
002B        :!=F
002C        :JC   =M003
002D        :L    FW  30
002E        :L    KF +48
0030        :!=F
0031        :JC   =M004
0032        :BEU
0033 M001   :L    KF +0
0035        :T    FW  30
0036        :T    FW  40
0037        :BEU
0038 M002   :L    KF +12
003A        :T    FW  30
003B        :L    KF +0
003D        :T    FW  40
003E        :BEU
003F M003   :L    KF +24
0041        :T    FW  30
0042        :L    KF +0
0044        :T    FW  40
0045        :BEU
0046 M004   :L    KF +36
0048        :T    FW  30
0049        :L    KF +0
004B        :T    FW  40
004C        :BE
```

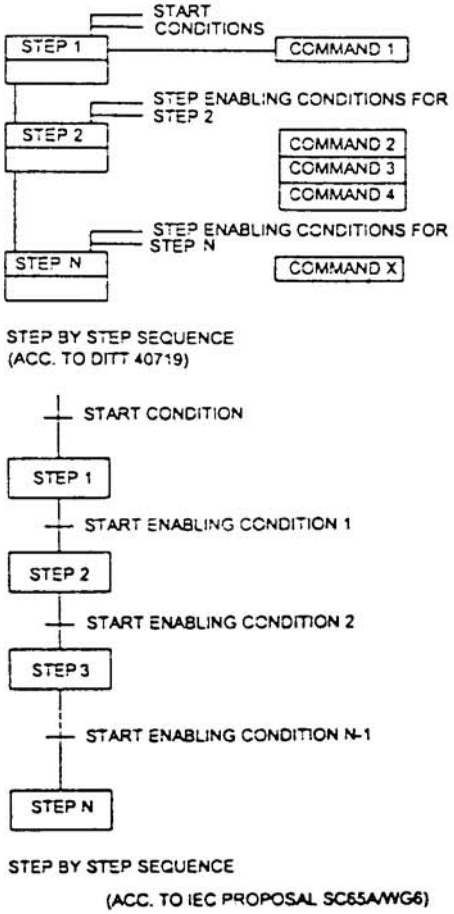
آدرس شروع سطر اول در DB دوم

در صورتی که برنامهٔ بلوک FB 36 را اجرا کنیم خواهیم دید که هنگام فعال و سپس غیرفعال شدن ورودی‌های سنسورها پیام خطای متناظر با هر ورودی بر روی سطر اول DB 33 منتقل می‌شود.

چنانچه در این برنامه بخواهیم در صورت عدم وجود خطا هیچ پیغامی بر روی صفحهٔ مونیتور ظاهر نشود می‌توان سطر پنجم از DB 32 را با کاراکترهای Blank Space پر نمود و با ایجاد تغییرات اندکی در برنامه، اطلاعات این سطر یعنی جای خالی را به سطر اول از DB 33 منتقل کرد. نوشتن این برنامه به عنوان تمرین به خواننده واگذار می‌شود.

۵-۴- ساختار برنامه‌های ترتیبی

برخلاف برنامه‌های ترکیبی که شرایط فعلی برنامه، اجرای مرحلهٔ بعدی را مشخص می‌نماید در برنامه‌های ترتیبی اجرای هر مرحله منوط به انجام زمان اجرای مرحلهٔ قبل و یا اتمام انجام پروسهٔ قبلی است. مشخصهٔ این برنامه‌ها، مرحله به مرحله‌ای بودن آن و برقراری شرایط خاصی برای اجرای مراحل است. در این‌گونه برنامه‌ها، عملیات کنترل پروسه شامل مراحلی جداگانه است که اجرای هر مرحله وابسته به برقراری شرایط تعریف شده‌ای در برنامه می‌باشد. هر مرحله دارای فرمانهای کنترل وشرایط فعال‌سازی‌است. درشکل ۵-۳ اجرای‌مرحله به‌مرحلهٔ یک‌برنامهٔ‌ترتیبی نشان‌داده‌شده است.



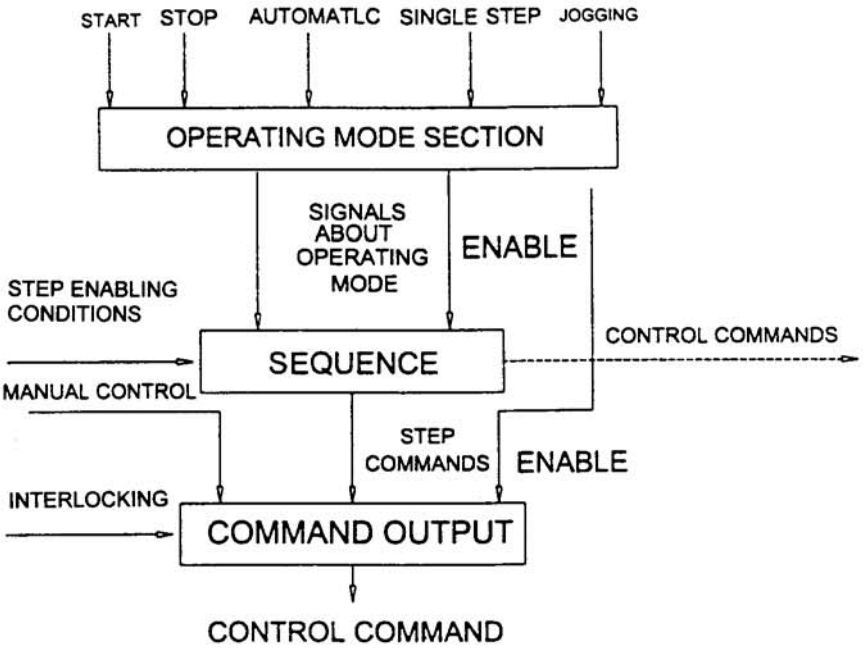
شکل ۵-۳ : اجرای مرحله به مرحلهٔ برنامهٔ ترتیبی

شرایط فعال‌سازی هر مرحله اجازهٔ ورود به مرحلهٔ بعدی و اجرای آن را صادر می‌نماید. فرمانهای هر مرحله شامل دستورات کنترلی برای واحدهای داخلی و خارجی است. به عنوان مثال ست نمودن فلگ‌ها، شروع نمودن زمان شمارش، سوئیچ نمودن المان‌ها و عناصر کنترلی و ...

به طور کلی هر برنامهٔ ترتیبی شامل سه بخش زیر است:

- انتخاب نحوهٔ اجرای مرحله (MODE SECTION)
- اجرای مرحله (SEQUENCE)
- بخش خروجی فرمانها (COMMAND OUTPUT)

در شکل ۵-۴ این قسمتها نشان داده شده است.

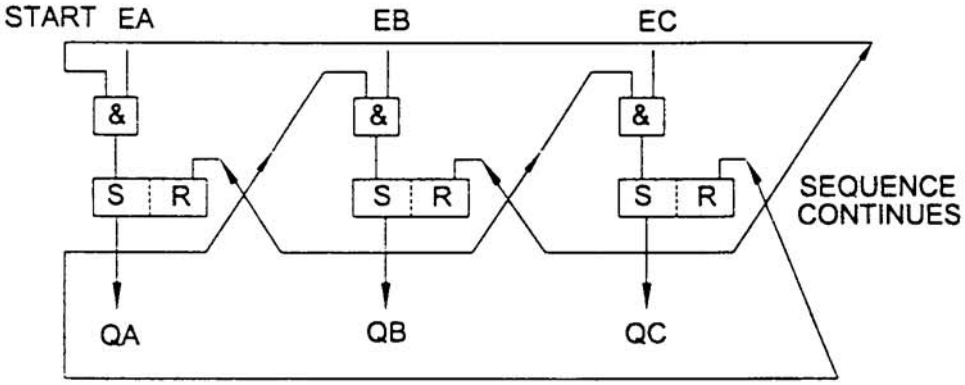


شکل ۵-۴ : ساختار یک کنترل‌کنندهٔ ترتیبی

- MODE SECTION : در این قسمت، پارامترهای اولیهٔ پیش‌نیاز (PRESET) جهت عملکرد مدهای مختلف اجرایی برنامه، پردازش می‌شوند. نتیجهٔ این قسمت به صورت سیگنال‌هایی به قسمت SEQUENCE و فرمانهای خروجی یا COMMAND OUTPUT

ارسال می‌گردد. این مرحله شامل فرمانها یا انتخاب‌های START ، STOP ،AUTOMATIC ، SINGLE STEP و JOGGING می‌باشد.

- در بخش SEQUENCE مراحل مختلف برنامه یکی پس از دیگری و مرحله به مرحله و البته منوط به برقراری شرایط فعال شدن هر مرحله، اجرا می‌شود. هر مرحله یا STEP را می‌توان معادل یا متناظر با قسمتی از برنامهٔ کنترل دانست که برای شروع و گذار از این مرحله نیاز به فرمانهای ست و ری‌ست دارد. این فرمانهای ست و ری‌ست ممکن است از فلیپ‌فلاپ، شمارنده، تایمر و یا ... صادر شوند. شکل زیر گویای مطلب عنوان شده است.



شکل ۵-۵ : نمونهٔ یک برنامهٔ ترتیبی با سه مرحله و چگونگی ست و ری ست شدن هر یک از مراحل

همانگونه که در شکل فوق ملاحظه می‌کنید از خروجی فلیپ‌فلاپ‌ها فرمانهایی صادر می‌شود که هر یک، فرمان آغاز مرحلهٔ بعد و پایان مرحلهٔ قبل است. اجرای مرحلهٔ فعلی در صورتی ادامه می‌یابد که شرایط فعال سازی آن (ENABLE) برقرار باشد. اگرچه فرمانهای کنترلی (CONTROL COMMAND) معمولاً از طریق قسمت خروجی فرمان به المان‌ها و عناصر کنترلی فرستاده می‌شوند اما این فرمانها می‌توانند مطابق شکل ۵-۵ مستقیماً در هر مرحله صادر شوند. برخلاف برنامه‌های ترکیبی که در برنامه‌نویسی آنها مجاز به استفاده از دستورات ست و ری‌ست (R و S) نبودیم در برنامه‌های ترتیبی اجرای مراحل توسط دستورات S و R به پیش می‌رود. این دستورات در هر مرحله و بنا به شرایط خاص هر مرحله ممکن است توسط یک

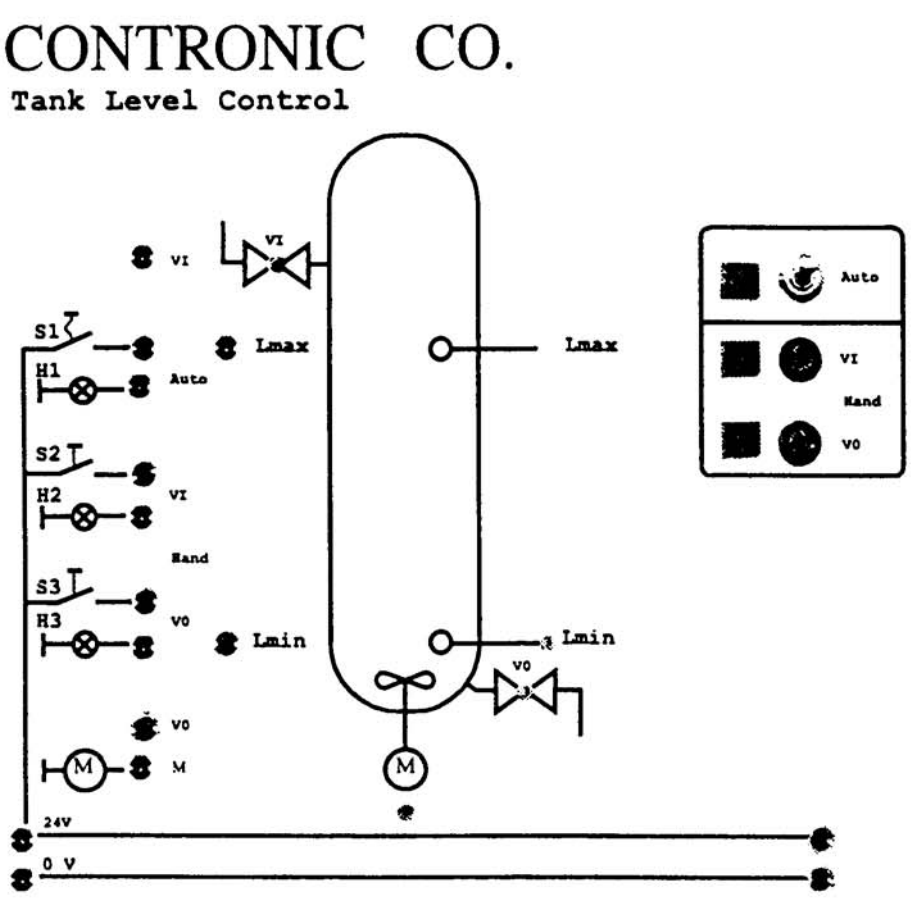
فلیپ‌فلاپ، تایمر یا شمارنده صادر شده باشد. همان‌گونه که دیده می‌شود برای ست‌شدن مرحلهٔ A باید دو سیگنال ENABLE (که در این مرحله همان سیگنال START است) و سیگنال گذار (TRANSITION) یا اتمام مرحلهٔ آخر یعنی C ، با یکدیگر AND شوند. پس از ست‌شدن مرحلهٔ A و ارسال فرمان Q_A در خروجی فلیپ‌فلاپ، این خروجی، سیگنال ENABLE مرحلهٔ بعدی یعنی B را فعال می‌سازد. برای ست‌شدن مرحلهٔ B نیز دو سیگنال ENABLE B و سیگنال اتمام مرحلهٔ A با یکدیگر AND شده، پس از ست‌شدن مرحلهٔ B و ارسال فرمان Q_B در خروجی، این فرمان، مرحلهٔ قبل یعنی مرحلهٔ A را ری‌ست می‌نماید و سیگنال ENABLE مرحلهٔ بعدی یعنی C را فعال می‌کند. برای ست‌شدن مرحلهٔ C نیز حاصل ترکیب عطفی دو سیگنال ENABLE C و سیگنال اتمام مرحلهٔ B لازم است. پس از ست‌شدن این مرحله، Q_C ، اجرای مرحلهٔ قبل یعنی B را ری‌ست نموده، سیگنال ENABLE مرحلهٔ بعد یعنی مرحلهٔ A را صادر می‌کند.

در بخش COMMAND OUTPUT فرمانهای هر مرحله به صورت منطقی با سیگنال‌های حاصل از بخش MODE SECTION و همچنین با سیگنال‌های START و STOP در ارتباط هستند. خروجی این قسمت فرمانهای ارسال شده به سوی عناصر کنترلی می‌باشد.

حال که با برنامه‌های ترتیبی و اجزای تشکیل‌دهندهٔ آنها آشنا شدیم به ذکر یک مثال می‌پردازیم. اگر به خاطر داشته باشید در فصل چهارم در مورد نحوهٔ روبرو شدن با یک پروژهٔ صنعتی به بحث پرداختیم و در نظر گرفتن شرایط ایمنی را به عنوان یکی از مهمترین مواردی که در برنامه‌نویسی باید مد نظر قرار داشت عنوان نموده، برای فهم بیشتر مطالب، مثال کنترل سطح مایع داخل مخزن را بررسی نمودیم. اکنون قصد داریم تا در ادامه، برنامهٔ کنترلی این مخزن را بنویسیم.

مثال ۵-۵: در بسیاری از پروسه‌های صنعتی عملکرد اتوماتیک و دستی سیستم تحت کنترل همراه با نیازمندیهای اساسی پروسه باید در برنامهٔ کنترل در نظر گرفته شود. شبیه‌ساز کنترل سطح مایع مخزن را که به منظور درک روند برنامه‌نویسی این‌گونه پروسه‌ها طراحی شده است در نظر بگیرید.

همان‌گونه که ملاحظه می‌شود علاوه بر شیرهای ورودی و خروجی (VI و VO) و سنسورهای نشان‌دهندهٔ سطح مایع (L min و L max) و موتور الکتریکی همزن (M) موارد دیگری نیز در نظر گرفته شده است. در این پروسه دو حالت کنترلی Auto و Hand وجود دارد که برای هر یک کلید جداگانه‌ای تعبیه شده است.



شکل ۵-۶ : شبیه‌ساز کنترل سطح مایع داخل مخزن

در صورتی که کلید Auto فعال باشد، سیستم به صورت خودکار مراحل زیر را انجام می‌دهد ولی در صورتی که کلید Hand فعال باشد کنترل به صورت دستی (Manual) انجام می‌شود. در حالت دستی دو کلید VI و VO وجود دارد که با فشار دادن هر یک، مرحلهٔ مربوط به همان کلید انجام می‌شود. مثلاً با فشار دادن کلید VO مراحل مربوط به باز شدن شیر خروجی دنبال می‌شود.

دراین پروسه در صورت خالی بودن مخزن و یا به عبارتی روشن شدن LED مربوط به L min، مواد شیمیایی از طریق شیر ورودی VI وارد مخزن شده، شیر ورودی تا زمان پر شدن مخزن یا فعال شدن LED مربوط به L max باز باقی می‌ماند. به محض پر شدن مخزن و روشن شدن LED

مربوط به L max شیر ورودی VI بسته و جهت تسریع واکنش، موتور الکتریکی همزن روشن می‌شود. مدت زمان روشن بودن موتور بستگی به نوع واکنش دارد. برای پروسه مذکور، این مدت را کوتاه و برابر ۵ ثانیه در نظر می‌گیریم . پس از خاموش شدن موتور همزن، شیر خروجی VO باز و مواد تخلیه می‌شوند. پس از تخلیه شدن، LED مربوط به L min روشن و سیکل مذکور مجدداً انجام می‌شود.

مواردی که ذکر شد عملکرد سیستم در حالت Auto است. قصد داریم در این حالت LED مربوط به حالت Auto نیز روشن باشد و هر زمان که کلید Auto غیرفعال شود LED مربوطه خاموش شده، کنترل به صورت دستی انجام گیرد.

در حالت کنترل دستی، برنامهٔ کنترلی به گونه‌ای نوشته شود که:

اگر کلید VI را فشار دهیم LED مربوطه روشن و شیر ورودی VI نیز باز شود و در صورتی که کلید VO را فشار دهیم باز هم LED مربوطه روشن و شیر خروجی VO نیز باز شود و در هر دو حالت فوق در صورت پر یا خالی شدن مخزن اجرای مرحلهٔ مربوطه متوقف گردد. در این حالت باید برنامه به گونه‌ای نوشته شود که وقتی هر یک از دو کلید VI و VO را رها نمودیم اجرای پروسه متوقف شود.

در مورد این پروسه روند برنامه‌نویسی را دنبال می‌کنیم.

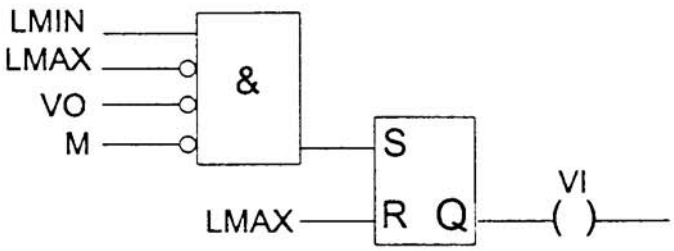
۱- **تعریف پروژه:** کنترل این پروسه به دو صورت AUTO و HAND به صورتی که تشریح شد انجام می‌گیرد.

۲- **رسم فلوچارت برنامه:** در رسم فلوچارت برنامه سعی شده تا در مورد ورودی‌ها و خروجی‌ها از اسامی متناظر آنها استفاده شود.

همان‌گونه که گفته شد کاربرد دستورات S و R در برنامه‌های ترتیبی بسیار زیاد است به طوری که تقریباً در فلوچارت هر مرحله، این دستورات نیز مشاهده می‌شوند. این دستورات ممکن است مربوط به فلیپ‌فلاپ، تایمر یا شمارنده باشد. در ادامه، فلوچارت هر مرحله در حالت AUTO به طور جداگانه آمده است. در رسم فلوچارت‌ها شرایط ایمنی نیز در نظر گرفته شده‌اند.

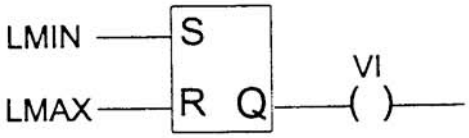
از خوانندگان تقاضا می‌شود به توضیحات ارائه شده در مورد هر فلوچارت دقت کافی مبذول دارند زیرا این روند برنامه‌نویسی کاملاً کلاسه شده بوده، نوشتن برنامهٔ پروژه‌های عظیم نیز با

به‌کارگیری این روش، بسیار ساده خواهد بود.



در فلوچارت مرحلهٔ اول که در انتها، فرمان باز شدن شیر ورودی VI را صادر می‌کند از یک فلیپ‌فلاپ SR استفاده شده است که ورودی S آن حاصل ترکیب عطفی سیگنال‌های L min ، $\overline{L\ max}$ ، $\overline{VO}$ و $\bar{M}$ بوده، ورودی R آن سیگنال L max باشد. حال به بررسی این فلوچارت می‌پردازیم.

در متن مسأله عنوان شده بود که هرگاه سطح مایع درون مخزن به L min برسد فرمان باز شدن و هنگامی که سطح مایع به L max برسد فرمان بسته شدن شیر ورودی صادر می‌شود. در صورتی که فرمان باز و بسته شدن شیر ورودی را با ست و ری‌ست شدن یک فلیپ‌فلاپ نمایش دهیم باید فلوچارت زیر را داشته باشیم:



اما در فلوچارت رسم شده برای این مرحله دیدیم که ورودی ست این فلیپ‌فلاپ حاصل ترکیب عطفی چندین سیگنال می‌باشد. برای روشن شدن مطلب و دلیل استفاده از چنین ترکیبی در ورودی S ، فرض کنید که به عنوان مثال در حالتی هستیم که سطح مایع داخل مخزن به L min رسیده و فرمان باز شدن شیر ورودی VI صادر شده باشد. حال اگر در این زمان شیر خروجی VO نیز باز باشد واضح است که تمام مواد داخل شده از شیر ورودی VI از طریق شیر خروجی VO بدون مخلوط

شدن به وسیلهٔ همزن الکتریکی خارج شده، عملاً در این پروسه از همزن استفاده نمی‌شود. در ضمن از آنجایی که شیر خروجی باز است هیچ‌گاه مخزن پر نمی‌شود و سیگنال ناشی از L max همیشه در حالت "۰" باقی مانده، خروجی فلیپ‌فلاپ را ری‌ست نمی‌کند. به عبارت دیگر شیر ورودی برای همیشه باز باقی خواهد ماند. بنابراین در ورودی S فلیپ‌فلاپ علاوه بر L min باید از نقیض VO نیز استفاده نمود و این بدان معنی است که برای باز شدن شیر ورودی لازم است که سطح مایع داخل مخزن به L min رسیده، علاوه بر آن شیر خروجی VO نیز بسته باشد. دلیل استفاده از سیگنال‌های دیگر نیز در ادامه آمده است:

$\overline{\text{L max}}$: واضح است که چنانچه سیگنال L min ظاهر شود و یا به عبارت دیگر "۱" = L min باشد باید برای سیگنال L max مقدار "۰" را داشته باشیم و در صورتی که L min و L max هر دو در یک زمان مقدار "۱" داشته باشند احتمال معیوب بودن یکی از سنسورها وجود دارد. بنابراین در حالتی که "۱" = L min است، "۰" = L max خواهد بود و در نتیجه "۱" = $\overline{\text{L max}}$ می‌باشد. دلیل استفاده از $\overline{\text{L max}}$ در نظر گرفتن شرایط ایمنی در این پروژه است و در صورت پر بودن مخزن نیز شیر ورودی بسته خواهد شد.

$\bar{M}$: از آنجایی که در مدت زمان پر شدن مخزن بایستی موتور الکتریکی همزن خاموش باشد از سیگنال $\bar{M}$ در ورودی S همراه با سیگنال‌های دیگر استفاده شده است. در حالتی که موتور خاموش است "۰" = M بوده، نقیض M دارای ارزش "۱" می‌باشد.

بنابراین خروجی فلیپ‌فلاپ نشان داده شده در این فلوچارت هنگامی ست می‌شود که:

اولاً "۱" = L min بوده، یا به عبارت دیگر سطح مایع داخل مخزن به کمترین مقدار خود برسد.

ثانیاً "۰" = L max (یا "۱" = $\overline{\text{L max}}$) باشد زیرا در حالتی که سطح مخزن به پائین‌ترین مقدار خود رسیده باشد داشتن مقدار "۱" در L max کاملاً غیرمنطقی است.

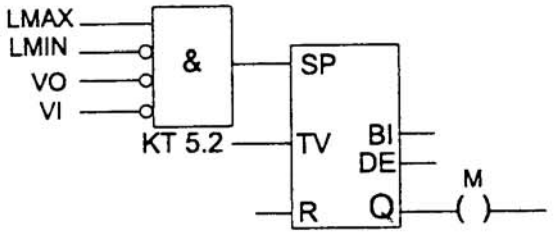
ثالثاً "۰" = VO (یا "۱" = $\overline{\text{VO}}$) باشد. یعنی برای ست شدن خروجی فلیپ‌فلاپ یا به عبارت دیگر برای باز شدن شیر ورودی VI باید شیر خروجی VO بسته باشد.

رابعاً "۰" = M (یا "۱" = $\bar{M}$) باشد. یعنی برای باز شدن شیر ورودی می‌بایست موتور الکتریکی همزن خاموش باشد. پس در ورودی S فلیپ‌فلاپ از ترکیب عطفی سیگنال L min و نقیض سیگنال‌های L max، VO و M استفاده می‌کنیم. حال ورودی R این فلیپ‌فلاپ را بررسی می‌کنیم.

در متن مسأله ذکر شد که اگر سطح مایع داخل مخزن به L max برسد شیر ورودی بسته خواهد شد. پس ورودی R تنها سیگنال L max می‌باشد.

سؤال: دلیل استفاده از فلیپ فلاپ SR در این مرحله چیست یا به عبارت دیگر در صورتی که از فلیپ‌فلاپ RS استفاده شود چه اشکال یا تغییری در برنامه بوجود می‌آید؟

همان‌گونه که در فصول گذشته عنوان شد در فلیپ‌فلاپ‌ها همواره از نظر اجرایی ارجحیت با ورودی دوم است، یعنی در فلیپ‌فلاپ SR به دلیل نزدیکتر بودن ورودی R به انتهای برنامه، این ورودی ارجح خواهد بود. بنابراین سیگنال L max را در ورودی R یک فلیپ‌فلاپ SR قرار می‌دهیم تا نسبت به ورودی S ارجح باشد، زیرا روند اجرای برنامه به گونه‌ای است که ابتدا شیر ورودی باز و فرمان ست صادر می‌شود و پس از رسیدن سطح مایع به L max شیر ورودی بسته و یا خروجی فلیپ‌فلاپ ری‌ست می‌شود. اکنون فلوچارت مرحلهٔ دوم یعنی روشن شدن موتور الکتریکی همزن را بررسی می‌کنیم.



همان‌گونه که ملاحظه می‌کنید جهت ست شدن تایمر استفاده شده در این مرحله از حاصل ترکیب عطفی سیگنال‌های L max ، $\overline{\text{L min}}$ ، $\overline{\text{VO}}$ و $\overline{\text{VI}}$ استفاده شده است. این بدان معنی است که برای روشن ماندن موتور الکتریکی همزن برای مدت زمان خواسته شده بایستی تمامی شرایط زیر برقرار باشند:

۱- "۱" = L max، یا به عبارت دیگر سطح مایع داخل مخزن در بالاترین حد ممکن خود باشد.

۲- "۰" = L min (یا "۱" = $\overline{\text{L min}}$)، واضح است که چنانچه "۱" = L max باشد L min نمی‌تواند مقدار "۱" داشته باشد. (در نظر گرفتن شرایط ایمنی)

۳- "۰" = VO (یا "۱" = $\overline{\text{VO}}$)، یعنی شیر خروجی بسته باشد.

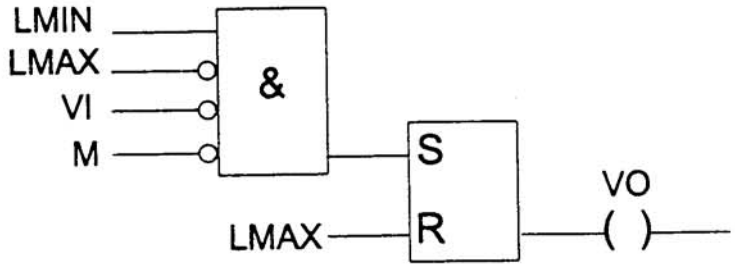
۴- "۰" = VI (یا "۱" = $\overline{VI}$)، یعنی شیر ورودی بسته باشد.

در این فلوچارت برای روشن ماندن موتور در مدت زمان خواسته شده در متن مسأله (۵ ثانیه) از یک تایمر استفاده شده است حال ممکن است در ذهن خواننده این سؤال مطرح شود که:

سؤال: برای انجام این مرحله از پروژه مجاز به استفاده از چه نوع تایمری هستیم؟

در این پروسه به دلیل وابسته بودن خروجی به ورودی باید از تایمر SP استفاده نمائیم. زیرا جهت ست شدن تایمر باید تمامی شرایط مذکور برقرار باشد و در صورت قطع شدن یکی از شرایط باید خروجی تایمر غیرفعال شود، و از آنجایی که در تایمر SP فعال بودن خروجی وابسته به فعال بودن ورودی S می‌باشد از این نوع تایمر استفاده نموده‌ایم. در این تایمر برای پارامتر TV از KT 5.2 استفاده شده است. برای دستیابی به دقت بیشتر و تولرانس یا خطای کمتر می‌توان از KT 50.1 نیز استفاده نمود.

همان‌گونه که در فلوچارت دیده می‌شود از ورودی R و خروجی‌های BI و DE تایمر، استفاده نشده است و تنها خروجی تایمر، جهت فرمان دادن به موتور الکتریکی همزن به کار برده شده است. دلیل عدم استفاده از خروجی‌های BI و DE را می‌توان عدم نیاز به آنها و دلیل عدم استفاده از ورودی R را می‌توان استفاده از تایمر SP دانست. زیرا در تایمر SP وضعیت خروجی کاملاً وابسته به ورودی S است و در این پروسه ورودی S حاصل ترکیب عطفی شرایط و سیگنال‌هایی است که عدم برقراری حداقل یکی از آنها موجب ری‌ست شدن تایمر مذکور شده، خروجی "۰" خواهد شد. بنابراین عدم برقراری حداقل یکی از شرایط باعث ری‌ست شدن خروجی خواهد شد. در اینجا نیازی به تعریف ورودی R جهت ری‌ست کردن خروجی Q نداریم. حال فلوچارت مرحلهٔ سوم یعنی باز شدن شیر خروجی VO را رسم می‌کنیم.



در این مرحله جهت فرمان باز شدن شیر خروجی VO از یک فلیپ‌فلاپ SR استفاده شده است که ورودی S آن حاصل ترکیب عطفی L max و $\overline{L\ min}$ ، $\overline{VI}$ و $\bar{M}$ بوده، ورودی R آن L min می‌باشد. این بدان معنی است که برای باز شدن شیر خروجی باید تمامی شرایط زیر برقرار باشند:

۱- سیگنال "۱" = L max باشد یعنی سطح مایع داخل مخزن در بالاترین مقدار ممکن خود قرار گیرد.

۲- سیگنال "۰" = L min باشد (یا "۱" = $\overline{L\ min}$)، این شرط به دلیل در نظر گرفتن شرایط ایمنی در مورد معیوب بودن سنسورهاست.

۳- سیگنال "۰" = VI باشد (یا "۱" = $\overline{VI}$)، به عبارت دیگر شیر ورودی بسته باشد.

۴- سیگنال "۰" = M باشد (یا "۱" = $\bar{M}$)، به عبارتی موتور الکتریکی همزن خاموش باشد.

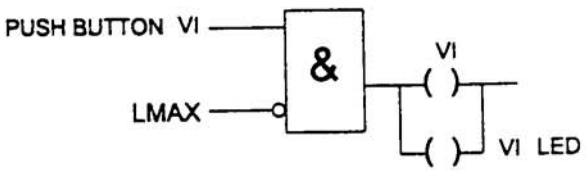
واضح است که برای ری‌ست یا بسته شدن شیر خروجی کافی است سیگنال "۱" = L min بوده، و یا به عبارت دیگر سطح مایع به پائین‌ترین مقدار خود رسیده باشد.

سه مرحلهٔ مذکور مربوط به حالتی است که سیستم به صورت اتوماتیک (AUTO) کنترل می‌شود. حال مراحل مربوط به حالت HAND را بررسی می‌کنیم. همان‌گونه که گفته شد در این حالت می‌خواهیم پروسه به گونه‌ای عمل نماید که:

- چنانچه پوش باتن (Push Button) مربوط به VI دستی را فشار دهیم لامپ یا LED مربوطه روشن و شیر ورودی نیز باز شود.

- چنانچه پوش باتن مربوط به VO دستی را فشار دهیم LED مربوطه روشن و شیر خروجی نیز باز شود.

توجه داشته باشید که در نظر گرفتن شرایط ایمنی در هر دو حالت AUTO و HAND الزامی است. در حالت HAND و برای مرحلهٔ VI دستی، فلوچارت زیر را در نظر بگیرید.



در این مرحله از ترکیب عطفی دو سیگنال پوش باتن VI دستی و $\overline{L\ max}$ استفاده شده است. در اینجا برخلاف حالت AUTO از فلیپ‌فلاپ و یا در حالت کلی از ست و ری‌ست استفاده نشده است. دلیل عدم استفاده از ست و ری‌ست پس از توضیح روند پروسه کاملاً روشن خواهد شد. روند اجرای این مرجله بدین صورت است که:

در صورت فشار دادن و نگه داشتن پوش باتن VI دستی و پائین‌تر بودن سطح مخزن از بالاترین مقدار ممکن خود، شیر ورودی باز و LED مربوطه نیز روشن خواهد شد. در صورتی که دست را از روی پوش باتن برداریم و یا در حالتی که مخزن پر شده و یا به عبارت دیگر "۱" = L max شود ("۰" = $\overline{L\ max}$) حاصل ترکیب عطفی این دو سیگنال "۰" و شیر ورودی بسته می‌شود. سپس LED مربوط به VI دستی نیز خاموش خواهد شد. در اینجا علت عدم استفاده از S و R مشخص می‌گردد. اگر به خاطر داشته باشید در فصل سوم تفاوت بین دو دستور ست (S) و هم‌ارزی (=) را بیان کردیم. دو برنامهٔ زیر را در نظر بگیرید.

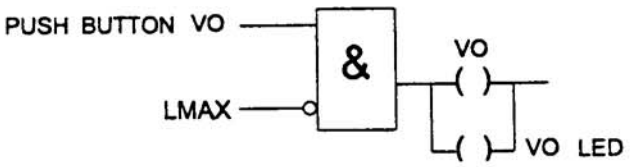
(۱)
```
:A  I  0.0
:S  Q  14.0
```

(۲)
```
:A  I  0.0
:=  Q  14.0
```

در برنامهٔ (۱) فعال شدن (ست شدن) خروجی Q 14.0 تنها مستلزم وجود یک لبهٔ بالارونده در ورودی I 0.0 است و در صورتی که I 0.0 در مدت کوتاهی فعال و سپس غیرفعال شود خروجی Q 14.0 در حالت فعال باقی می‌ماند و برای ری‌ست شدن احتیاج به یک ورودی R دارد.

در برنامهٔ (۲)، خروجی کاملاً وابسته به ورودی است و در صورتی که ورودی I 0.0 فعال باشد خروجی Q 14.0 نیزفعال است ودرصورتی که "۰" = I 0.0 باشد خروجی نیز غیرفعال خواهد بود.

در مرحلهٔ بعدی حالت HAND، فلوچارت پروسهٔ خالی شدن مخزن را بررسی می‌کنیم. مطابق فلوچارت زیر، شیر خروجی VO هنگامی باز خواهد شد که پوش باتن VO در حالت فعال نگه داشته شده، سطح مخزن نیز به L min نرسیده باشد.



۳- تهیهٔ لیستی از ابزار مورد نیاز: ورودی‌ها، خروجی‌ها و به طور کلی عملوندهای مورد استفاده برای اجرای این پروسه به شرح زیر می‌باشد:

ورودی‌ها	L_{min}	: I 21.0	
	L_{max}	: I 21.1	
	کلید AUTO	: I 21.2	
	کلید دستی VI	: I 21.3	
	کلیددستی VO	: I 21.4	
خروجی‌ها	VO	: Q 14.0	
	M	: Q 14.1	
	VI	: Q 14.2	
	AUTO LED	: Q 14.3	
	VI LED دستی	: Q 14.4	
	VO LED دستی	: Q 14.5	

۴- نوشتن برنامه به یکی از سه روش برنامه‌نویسی: در این پروسه، برنامهٔ اجرای حالت AUTO را در یک بلوک PB و اجرای حالت HAND را در PB دیگر قرار می‌دهیم. اگر به خاطر داشته باشید در فصل سوم در یکی از مثالهای اولیه مسأله‌ای به صورت زیر مطرح شد که:

می‌خواهیم برنامه‌ای بنویسیم که با فعال شدن یک کلید، بلوک خاصی به اجرا در آید و در صورت غیرفعال بودن همان کلید، بلوک دیگری اجرا گردد. در انتهای برنامه خاطرنشان کردیم که این کلید می‌تواند کلید مشخص کنندهٔ حالت AUTO و HAND (MANUAL) باشد.

در این پروژه نیز کلید AUTO یا ورودی I 21.2 را به عنوان کلید مشخص کنندهٔ حالت اجرای برنامه در یکی از دو وضعیت AUTO یا HAND تعریف می‌کنیم. اگر برنامهٔ حالت AUTO را در PB 20 و حالت HAND را در PB 21 بنویسیم در OB 1 که ساختار کلی اجرای برنامه را نشان می‌دهد خواهیم داشت:

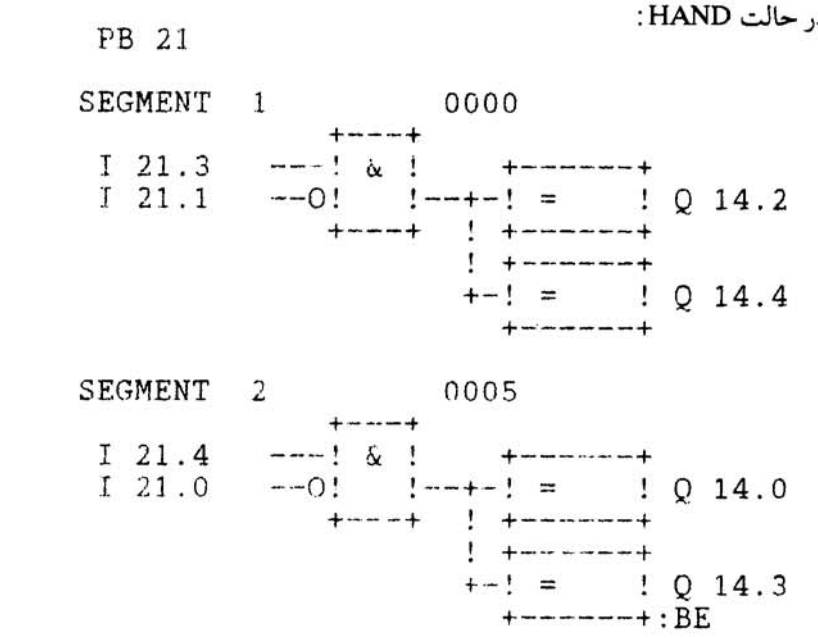
```
OB 1

SEGMENT  1            0000
0000      :A    I     21.2
0001      :=    Q     14.3
0002      :JC   PB    20
0003      :AN   I     21.2
0004      :JC   PB    21
0005      :BE
```

بلوک‌های PB 20 و PB 21 نیز به صورت زیر نوشته می‌شوند.

بلوک برنامه در حالت AUTO:

```
 PB 20

SEGMENT  1          0000
            +---+
 I 21.0   ---! & !
 I 21.1   --O!   !   Q 14.2
 Q 14.0   --O!   !    +------+
 Q 14.1   --O!   !----!S     !
            +---+    !      !   +------+
           I 21.1   --!R    Q!-+-! =    ! Q 14.2
                      +------+   +------+

SEGMENT  2          000A
            +---+
 I 21.1   ---! & !
 I 21.0   --O!   !   T 2
 Q 14.0   --O!   !    +------+
 Q 14.2   --O!   !----!1_-_  !
            +---+    !       !
          KT 005.2 --!TV  BI!-
                     !    DE!-
                     !       !   +------+
                   --!R    Q!-+-! =    ! Q 14.1
                     +------+   +------+

SEGMENT  3          0017
            +---+
 I 21.1   ---! & !
 I 21.0   --O!   !   Q 14.0
 Q 14.2   --O!   !    +------+
 Q 14.1   --O!   !----!S     !
            +---+    !      !   +------+
           I 21.0   --!R    Q!-+-! =    ! Q 14.0
                      +------+   +------+:BE
```

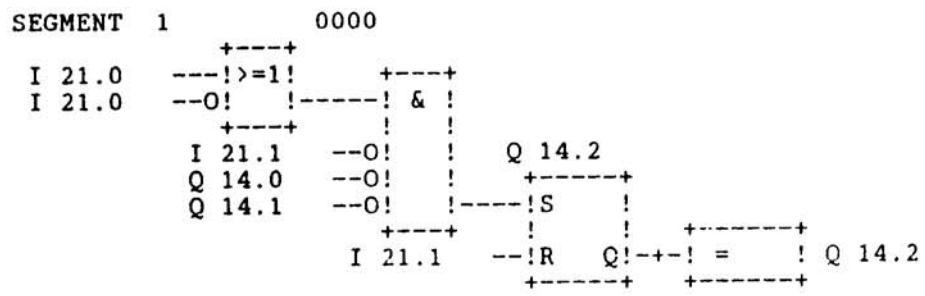
بلوک برنامه در حالت HAND:



۵- **بررسی شرایط ایمنی**: در بخش رسم فلوچارت، شرایط ایمنی نیز در نظر گرفته شده است. در این قسمت قصد داریم که با ایجاد تغییراتی در برنامه، پروسه را به نحوی کنترل نمائیم که به عنوان مثال:

۱- در حالت AUTO و در صورت قطع برق و وصل مجدد آن اجرای پروسه از حالت پر شدن مخزن شروع شود.

برای انجام این عمل کافی است با ایجاد تغییرات اندکی در PB 20 (بلوک برنامهٔ حالت AUTO) باعث شویم که همواره "۱" = L min باشد. برای این عمل می‌توان ترکیب فصلی I 21.0 و $\overline{\text{I 21.0}}$ (Lmin و $\overline{\text{L min}}$) را در ورودی S فلیپ‌فلاپ قرار داد.

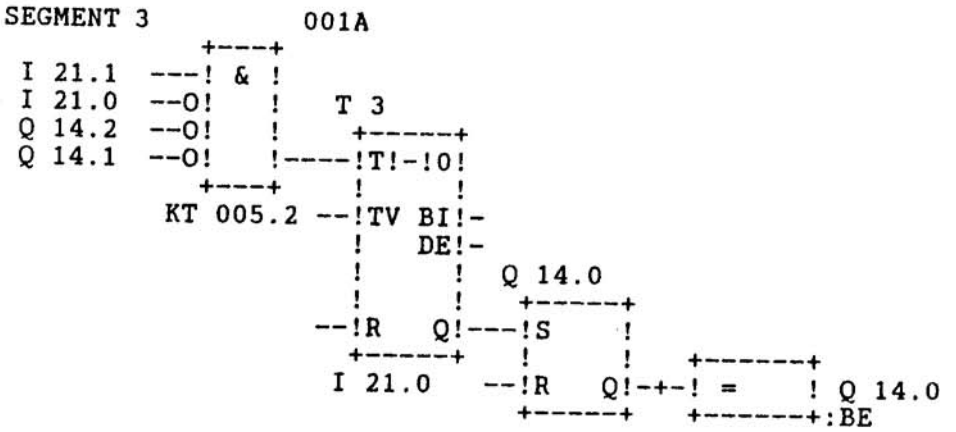


۲- فرض کنید که پروسه در حالت AUTO و در مرحلهٔ روشن بودن موتور الکتریکی همزن است. قصد داریم با ایجاد تغییراتی در برنامه باعث شویم که در صورت خاموش شدن موتور الکتریکی همزن، انجام پروسه به مدت ۵ ثانیه متوقف شود تا مواد شیمیایی داخل مخزن از تلاطم باز ایستد و پس از گذشت این زمان، شیر خروجی VO باز شود.

خوانندگان توجه داشته باشند که معمولاً در نظر گرفتن این مدت زمان تأخیر بین دو مرحله از یک پروسه الزامی است، زیرا در صورتی که بلافاصله پس از خاموش شدن همزن الکتریکی، شیر خروجی باز شود مواد شیمیایی متلاطم به هنگام خروج از لولهٔ خروجی به اطراف پاشیده می‌شود. بنابراین در طراحی سیستم و برنامه‌نویسی، در نظر گرفتن چنین مواردی موجب اجرای مطلوب‌تر پروسه خواهد شد.

بـه ایـن مـدت زمـان تأخیر کـه در یک پـروسه و بـین دو مـرحـله در نـظر گـرفته مـی‌شود Waiting Time می‌گویند. این زمان تأخیر یکـی از پـرکاربردترین مـوارد در پـروسه‌های کـنترلی محسوب می‌شود.

بنابراین برای داشتن چـنین حـالتی کـافی است در حـالت AUTO (در بـلوک PB 20) و در SEGMENT 3 از یک تایمر SD استفاده کنیم. دلیل استفاده از تایمر SD، نیاز به تأخیر (Delay) در این پروسه است.



در ادامه، برنامهٔ کلی با ایجاد دو تغییر یاد شده به روش STL آمده است.

بلوک برنامه در حالت AUTO:

```
 PB 20

SEGMENT  1              0000
0000      :A(
0001      :O    I     21.0
0002      :ON   I     21.0
0003      :)
0004      :AN   I     21.1
0005      :AN   Q     14.0
0006      :AN   Q     14.1
0007      :S    Q     14.2
0008      :A    I     21.1
0009      :R    Q     14.2
000A      :A    Q     14.2
000B      :=    Q     14.2
000C      :***


SEGMENT  2              000D
000D      :A    I     21.1
000E      :AN   I     21.0
000F      :AN   Q     14.0
0010      :AN   Q     14.2
0011      :L    KT  005.2
0013      :SP   T      2
0014      :NOP  0
0015      :NOP  0
0016      :NOP  0
0017      :A    T      2
0018      :=    Q     14.1
0019      :***
```

```
SEGMENT  3              001A
001A      :A    I     21.1
001B      :AN   I     21.0
001C      :AN   Q     14.2
001D      :AN   Q     14.1
001E      :L    KT  005.2
0020      :SD   T      3
0021      :NOP  0
0022      :NOP  0
0023      :NOP  0
0024      :A    T      3
0025      :S    Q     14.0
0026      :A    I     21.0
0027      :R    Q     14.0
0028      :A    Q     14.0
0029      :=    Q     14.0
002A      :BE
```

بلوک برنامه در حالت HAND:

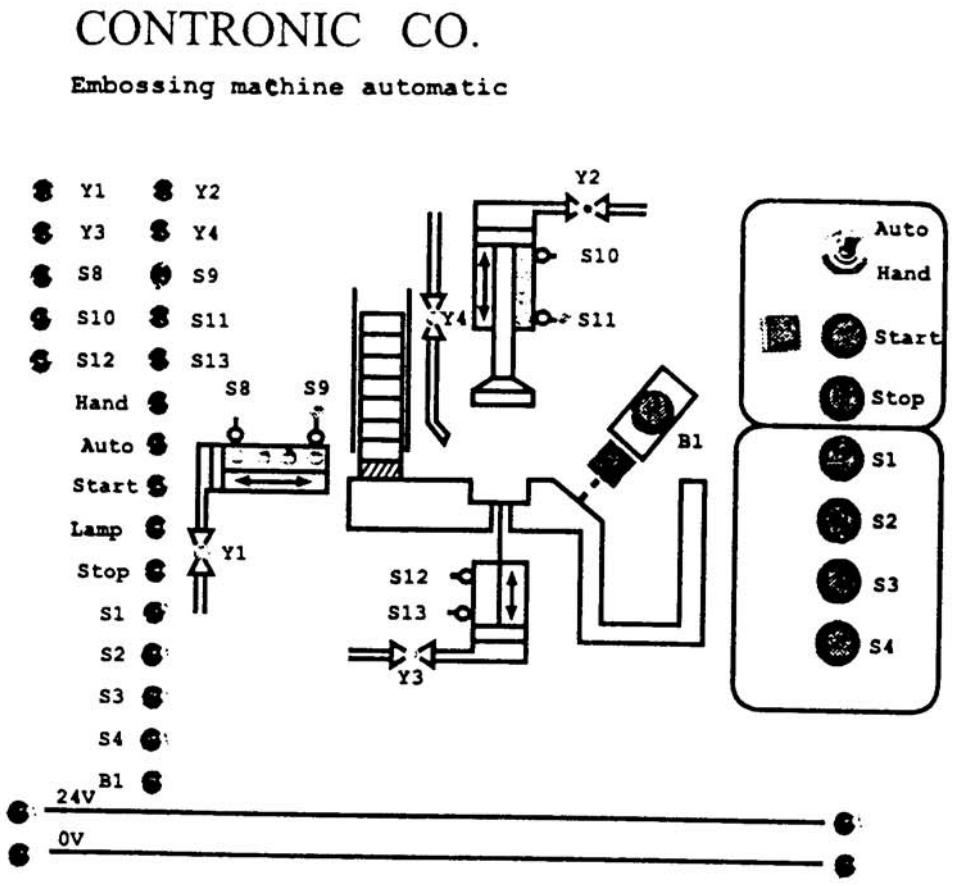
```
 PB 21

SEGMENT  1              0000
0000      :A    I     21.3
0001      :AN   I     21.1
0002      :=    Q     14.2
0003      :=    Q     14.4
0004      :***

SEGMENT  2              0005
0005      :A    I     21.4
0006      :AN   I     21.0
0007      :=    Q     14.0
0008      :=    Q     14.3
0009      :BE
```

اکنون به منظور درک بهتر برنامه‌نویسی فرآیندهای ترتیبی به توضیح عملکرد شبیه‌ساز دیگری می‌پردازیم:

مثال ۵-۶ : برای کنترل فرآیندهای ترتیبی (Sequential) ، هنگامی که چند عمل یکی پس از دیگری و تحت شرایط خاصی صورت می‌گیرند ساختار منطق کنترلی از سازمان مشخصی پیروی

می‌کند. برای کسب تجربه و برای کنترل چنین فرآیندهایی شبیه‌ساز ماشین پرس طراحی گردیده است.

شیرهای هیدرولیک، لیمیت سوئیچ‌ها، لامپ‌ها (LED) ، سوئیچ‌های فرمان متعدد، سلکتورهای انتخاب حالت دستی، اتوماتیک و تک مرحله‌ای برای شبیه‌سازی چنین فرآیندهایی ایده‌آل است. فرآیندهای پیچیده‌تر در حقیقت تکرار متناسب همان مراحلی است که برنامهٔ آن برای این شبیه‌ساز طراحی شده است. در شکل ۵-۷ این شبیه‌ساز را به همراه تمام عناصر کنترلی موجود ملاحظه می‌کنید.



شکل ۵-۷ : شبیه‌ساز ماشین پرس و عناصر کنترلی آن

۵-۵- نقطهٔ شروع یا حالت اولیه (Initial State)

در برنامه‌های ترتیبی لازم است که کنترل فرآیند از نقطه‌ای خاص شروع شود. مسلماً آغاز عملیات کنترل در این نقطه شرایط خاص خود را دارد. به این شرایط کنترلی که در برنامه‌های ترتیبی آغازکنندهٔ اجرای پروسه می‌باشند حالت اولیه یا Initial State گویند.

در این پروسه، حالت اولیه به گونه‌ای است که جهت آغاز اجرای پروسه باید تمام سیلندرها عقب بوده، هیچ قطعه‌ای در خط تولید وجود نداشته باشد. در صورت برقراری دو شرط مذکور و فشرده‌شدن کلید START، شیر ورودی روغن Y1 مربوط به سیلندر هیدرولیک اول باز خواهد شد. روند اجرای پروسه به شرح زیر است:

قطعات ورق خام در نردبان عمودی موجود در شکل بر روی یکدیگر قرار گرفته‌اند. با باز شدن شیر Y1 سیلندر شمارهٔ ۱ به جلو آمده، این حرکت رو به جلو توسط سنسورهایی که در این شبیه‌ساز با LED نمایش داده شده‌اند حس می‌شود. با جلو آمدن بازوی سیلندر شمارهٔ ۱، تنها یک قطعه ورق درون قالب قرار می‌گیرد که قرارگرفتن قطعه درون قالب توسط سنسور S9 حس می‌شود. (در حالتی که سیلندر حرکت کرده و سنسور S9 تحریک گردد قطعهٔ مورد نظر درون قالب قرار می‌گیرد.) در این مرحله شیر ورودی روغن سیلندر شمارهٔ ۲ یعنی Y2 باز و Y1 بسته می‌شود. سیلندر شمارهٔ ۲ به صورت عمودی پائین آمده، به محض تحریک نمودن سنسور S11، به آخرین حد مجاز حرکت خود می‌رسد. در اینجا قطعهٔ موجود در قالب کاملاً به شکل مورد نظر در آمده است. با تحریک شدن سنسور S11 شیر ورودی Y2 بسته و شیر ورودی Y3 (مربوط به بازوی هیدرولیک شمارهٔ ۳) باز خواهد شد. این بازو، قطعهٔ شکل داده شده را از قالب بیرون می‌اندازد. حرکت این بازو نیز به صورت عمودی و از پائین به بالا می‌باشد. به محض تحریک شدن سنسور S12 شیر ورودی Y3 بسته و Y4 باز می‌شود. با باز شدن Y4 ، هوای با فشار زیاد قطعهٔ موجود در خط را به محلی که جهت جمع‌آوری قطعات آماده درنظر گرفته‌شده است پرتاب می‌کند. عبور قطعه وافتادن آن در محل مذکور توسط سنسور B1 حس می‌شود. پس‌ازاین مرحله قطعهٔ بعدی وارد خط شده، سیکل مجدداً تکرار می‌شود.

نکاتی که در نوشتن برنامه جهت کنترل این پروسه بایستی مدنظر قرار گیرند عبارتند از:

۱- در صورت وجود قطعه‌ای در خط تولید، مجاز به واردکردن قطعهٔ دیگری نیستیم. به عبارت دیگر در هر زمان باید تنها یک قطعه در خط قرار داشته باشد.

۲- در صورت قطع برق و وصل مجدد آن، اجرای پروسه از همان مرحله‌ای آغاز شود که سیستم قبل از قطع برق در آن حالت قرار داشته است و یا در حالتی که نیاز به توقف خط و اجرای پروسه بوده، به عبارتی کلید STOP توسط کاربر استفاده شده باشد پس از شروع به کار مجدد، اجرای پروسه از همان مرحلهٔ قبل از توقف (STOP) آغاز گردد.

در این پروسه نیز همانند مثالهای قبلی دو حالت کنترل AUTO و HAND وجود دارد. در صورت انتخاب حالت کنترلی HAND می‌توان توسط کلیدهای S1 الی S4 که در شکل نشان داده شده‌اند اجرای مرحله به مرحله نیز داشته باشیم. در اجرای مرحله به مرحله، فشار هر کلید موجب به اجرا در آمدن مرحلهٔ متناظر با شمارهٔ کلید خواهد شد. همانگونه که عنوان شد قصد داریم در صورت توقف برنامه در هر دو حالت AUTO و HAND اجرای برنامه از مرحلهٔ قبل از توقف برنامه (STOP) آغاز شود. برای نوشتن این برنامه از تعدادی فلگ و فلیپ‌فلاپ استفاده خواهیم کرد. اگر به یاد داشته باشید در فصل دوم بیان شد که می‌توان برخی شرایط و حالات را در فلگ قرار داد و در صورت نیاز می‌توانیم اطلاعات را با فراخوانی فلگ‌ها از حافظه بخوانیم.

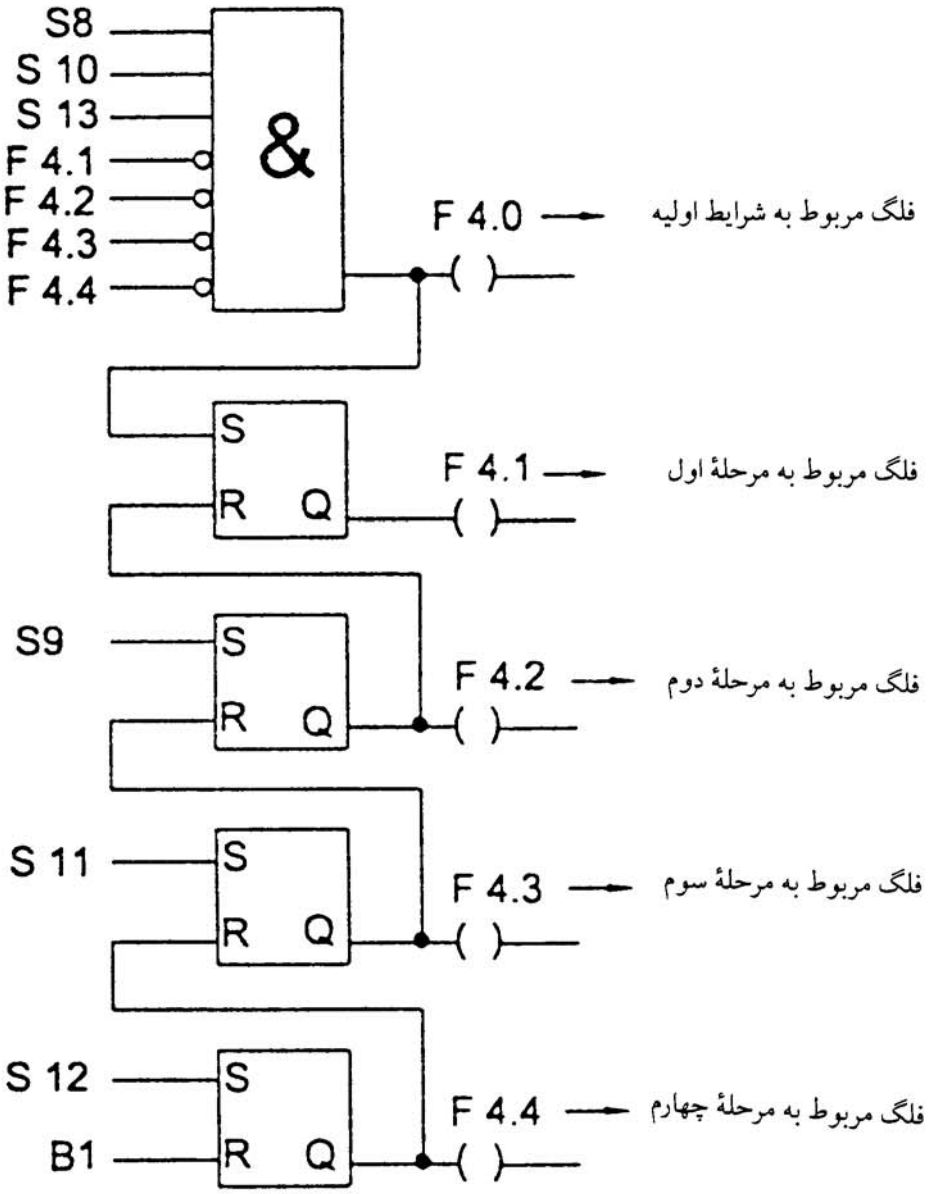
توجه داشته باشید که اطلاعات و شرایط مذکور بایستی در فلگ‌های پایدار (Retentive) قرار گیرند زیرا همانگونه که در فصل دوم گفته شد فلگ‌های پایدار پس از قطع منبع تغذیه اطلاعات خود را حفظ می‌کنند و از آنجایی که در برخی پروسه‌های حساس مانند این پروسه، اجرای برنامه پس از قطع جریان برق باید از همان مرحلهٔ قبل از قطع منبع تغذیه دنبال شود و نیاز به اطلاعات و آگاهی از شرایط موجود در آن مرحله می‌باشد، این شرایط و حالات را باید در فلگ‌های پایدار قرار داد. در PLC مورد بحث ما ۲۵۶ فلگ بایت وجود دارد (FY 0 - FY 255) که نیمه اول فلگ‌ها یعنی (FY 0 - FY 127) پایدار و بقیه ناپایدار می‌باشند. بنابراین از بیت‌های FW 4 (FY 4 , FY 5) جهت ذخیرهٔ شرایط مذکور استفاده شده است.

در مورد این پروسه نیز روند برنامه‌نویسی را به روش کلاسیک دنبال خواهیم نمود.

۱- **تعریف پروژه:** همانطور که ذکر شد این پروسه در دو حالت AUTO و HAND انجام‌پذیر است که توضیحات مربوط به چگونگی اجرای هر حالت ارائه شد.

۲- **رسم فلوچارت:** برای رسم فلوچارت ابتدا مرحلهٔ اولیه یا Initial State را بررسی می‌کنیم. شرایط اولیه به گونه‌ای است که تمام سیلندرها عقب بوده، قطعه‌ای در خط قرار نداشته باشد. عقب

بودن سیلندرها توسط سنسورهای S8 ، S10 و S13 حس می‌شود و عدم وجود قطعه در خط را می‌توان با فراخوانی اطلاعات موجود در فلگ‌های هر مرحله دریافت. در ادامه فلوچارت تمام مراحل رسم شده است.

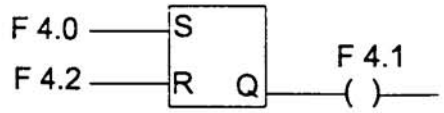


حال به توضیح هر یک از مراحل فوق می‌پردازیم:

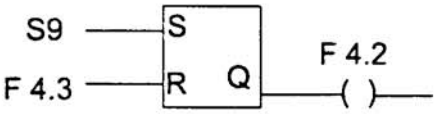
حالت اولیه یا F 4.0 : همان‌گونه که در متن مسأله ملاحظه نمودید حالت Initial State شامل شرایطی است که تمامی سیلندرها عقب بوده، هیچ قطعه‌ای درون خط نباشد. عقب بودن سیلندرها توسط سنسورهای S8 ، S10 و S13 حس می‌شوند. حال عدم وجود قطعه در هر یک از مراحل را بررسی می‌کنیم.

حالتی را در نظر بگیرید که قطعه‌ای در مرحلهٔ سوم وجود داشته باشد پس فلیپ‌فلاپ مربوط به این مرحله ست بوده و داریم: "۱" = F 4.3 . بنابراین عدم وجود قطعه در این مرحله را می‌توان با نقیض F 4.3 نشان داد. زیرا در صورتی که قطعه‌ای در این مرحله نباشد "۰" = F 4.3 بوده، نقیض این فلگ مقدار "۱" را خواهد داشت. بنابراین عدم وجود قطعه در مراحل چهارگانه را می‌توان با حاصل ترکیب عطفی نقیض فلگ‌های مربوط به هر چهار مرحله نشان داد. پس F 4.0 که معرف حالت اولیه یا Initial State می‌باشد هنگامی "۱" خواهد شد که حاصل ترکیب عطفی نشان داده شده در فلوچارت زیر "۱" باشد.

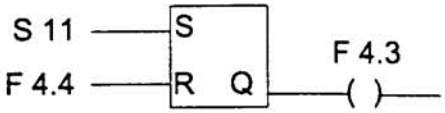
مرحلهٔ اول یا F 4.1 : همان‌گونه که در فلوچارت کلی ملاحظه می‌شود جهت فعال و غیرفعال نمودن این مرحله از یک فلیپ‌فلاپ SR استفاده شده است. در متن مسأله دیدیم که شرط انجام این مرحله وجود شرایط اولیه یا Initial State می‌باشد. بنابراین برای ست نمودن فلگ F 4.1 که معرف انجام مرحلهٔ اول است از F 4.0 استفاده شده است. جهت ری‌ست نمودن این فلگ، فلگ F 4.2 یعنی انجام مرحلهٔ بعدی به کار برده شده است، زیرا همان‌طور که قبلاً گفته شد در برنامه‌های ترتیبی انجام هر مرحله، مرحلهٔ قبلی را ملغی (ری‌ست) می‌کند. در اینجا نیز به محض فعال شدن F 4.2 یعنی آغاز مرحلهٔ دوم، F 4.1 ری‌ست خواهد شد و این بدان معنی است که با آغاز مرحلهٔ دوم، مرحلهٔ اول نیز پایان یافته است. در زیر فلوچارت مربوط به این مرحله جداگانه رسم شده است.



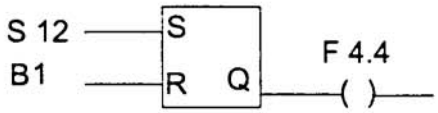
مرحله دوم F 4.2: همان‌طور که در فلوچارت زیر مشاهده می‌شود شرط اجرای این مرحله تحریک سنسور S9 می‌باشد. به محض اینکه بازوی هیدرولیک شمارهٔ ۱ به منتهی‌الیه سمت راست خود یعنی سنسور S9 برسد F 4.2 ست می‌شود. ورودی R فلیپ‌فلاپ به کار برده شده در این مرحله، F 4.3 یعنی اجرای مرحلهٔ سوم خواهد بود.



مرحله سوم F 4.3: شرط ورود به این مرحله تحریک سنسور S11 می‌باشد. در صورتی که بازوی هیدرولیک شمارهٔ ۲ به پایین‌ترین نقطهٔ ممکن خود برسد یا به عبارت دیگر سنسور S11 تحریک گردد فلیپ‌فلاپ مربوط به این مرحله ست می‌شود و خواهیم داشت "۱" = F 4.3. ورودی R این فلیپ‌فلاپ F 4.4 یعنی فلگ مربوط به مرحلهٔ بعدی است، بدین ترتیب که در صورت فعال شدن این فلگ ، فلگ مرحلهٔ قبل یعنی F 4.3 ری‌ست می‌شود.



مرحلهٔ چهارم F 4.4: همان‌طور که در فلوچارت زیر مشاهده می‌کنید شرط ورود به این مرحله تحریک شدن سنسور S12 می‌باشد. این بدان معنی است که بازوی هیدرولیک شمارهٔ ۳ به بالاترین حد مجاز خود رسیده است. ورودی ری‌ست فلیپ‌فلاپ این مرحله سنسور B_1 می‌باشد. عبور قطعه از این مرحله فلگ F 4.4 را ری‌ست می‌کند و پس از این مرحله، سیکل مجدداً تکرار می‌شود.

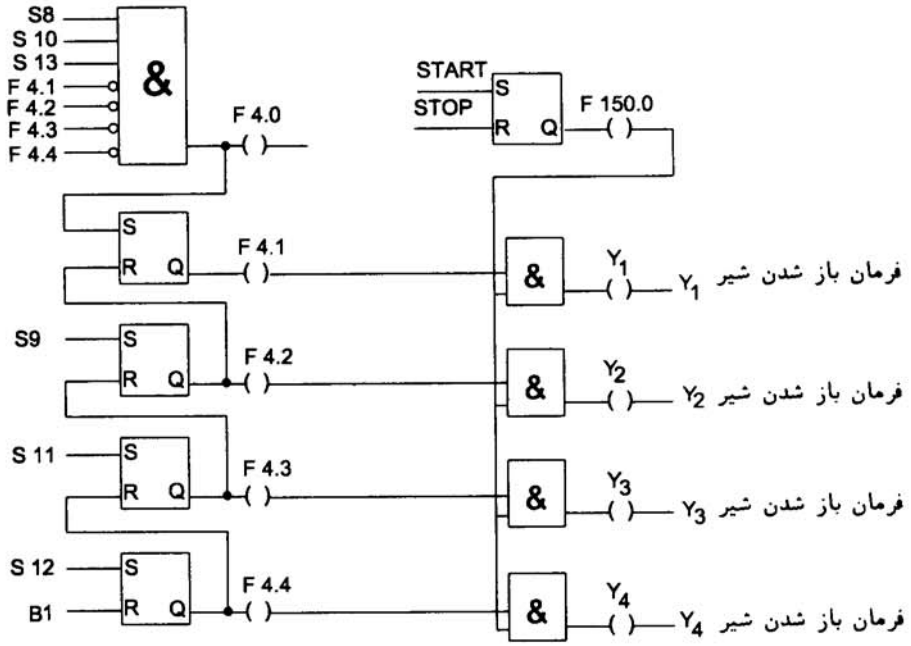


حال چگونگی ارتباط کلیدهای START و STOP را با برنامهٔ نوشته شده (فلوچارت رسم شده) بررسی می‌کنیم:

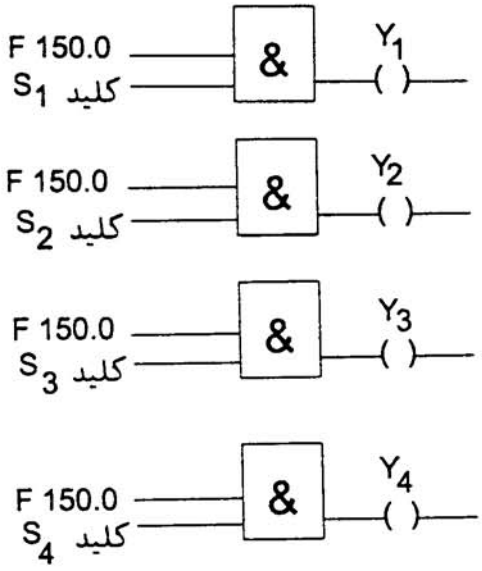
در صورتی که به خاطر داشته باشید در قسمت قبل بیان شد که به دلیل نیاز به اطلاعات در مورد شرایط و حالات سیستم پس از قطع جریان برق از بیت‌های فلگ پایدار استفاده نمودیم ولی در مورد START و STOP باید از یک فلگ ناپایدار استفاده نماییم. دلیل این امر کاملاً روشن است، زیرا به عنوان مثال، در صورت توقف سیستم (فشرده شدن کلید STOP) مجدداً نیاز به راه‌اندازی سیستم (فشرده شدن کلید START) داریم و باید از فلگ‌های ناپایدار که در هنگام قطع جریان برق اطلاعات خود را از دست می‌دهند استفاده کنیم.

حال که محل ذخیرهٔ اطلاعات مربوط به START و STOP مشخص گردید این سؤال مطرح است که این اطلاعات (به عنوان مثال F 150.0) چگونه با سایر فلگ‌ها ارتباط دارد؟

کاملاً واضح است که در مورد این قسمت نیز باید از یک فلیپ‌فلاپ SR استفاده کنیم که ورودی S آن کلید START و ورودی R آن کلید STOP باشد. خروجی این فلیپ‌فلاپ یعنی F 150.0 باید با تمامی فلگ‌های مربوط به مراحل چهارگانه، AND شود، زیرا تنها در صورت روشن بودن و در حال کار بودن سیستم (F 150.0 = "۱" یا START = "۱") مراحل چهارگانه قابل اجرا هستند. در فلوچارت زیر این مطلب به روشنی دیده می‌شود.



تا این قسمت از مثال در مورد حالت AUTO بحث نمودیم. اکنون حالت HAND را بررسی می‌کنیم. همان‌طور که قبلاً ذکر شد در این حالت چهار کلید S1، ...، S4 برای اجرای هر مرحله وجود دارد. در ادامه، فلوچارت این حالت رسم شده است.



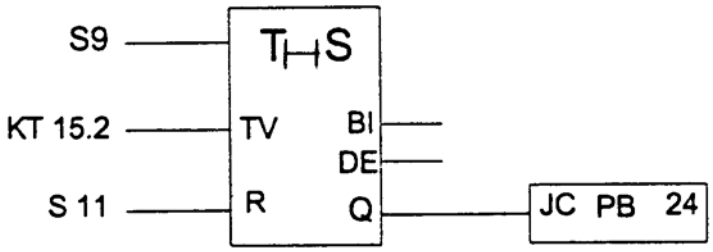
اکنون فرض کنید که قصد داریم در برنامهٔ حالت AUTO تغییری ایجاد کنیم به این صورت که اگر به عنوان مثال در مرحلهٔ دوم یعنی پایین آمدن پرس هیدرولیک شمارهٔ ۲ بنا به دلایلی مثلاً قطع شدن لولهٔ متصل به تانک حاوی روغن مشکلی بوجود آید اجرای این مرحله با تأخیر اندکی انجام شود این تأخیر در این مرحله ۱۵ ثانیه در نظر گرفته می‌شود و با ایجاد تغییری دیگر در برنامه به PLC فرمان می‌دهیم که در صورت عدم فعال شدن خروجی در طول این مدت (عدم رفع اشکال مذکور) اجرای برنامه متوقف شود. به این زمان تأخیر که معمولاً جهت مسائل ایمنی در نظر گرفته می‌شود Monitoring Time می‌گویند.

در برخی از پروسه‌های می‌توان برای هر مرحله و یا حتی برای کل پروسه Monitoring Time و Waiting Time تعریف نمود. مثلاً برای تعریف کل پروسه می‌توان از F 4.0 به عنوان شرط شروع (ست) و از تحریک سنسور B1 به عنوان شرط اتمام انجام پروسه (ری‌ست) استفاده نمود. در صورتی که برای هر مرحله از پروسه هم Waiting Time و هم Monitoring Time در نظر گرفته شده باشد رابطهٔ زیر را خواهیم داشت.

$$\left(\text{Monitoring Time}_{\text{هر مرحله}}\right) = \left(\text{Waiting Time}_{\text{هر مرحله}}\right) + \left(\text{Tolerance}_{\text{هر مرحله}}\right) + \left(\text{زمان اجرای}_{\text{هر مرحله}}\right)$$

یا (زمان اجرای مرحله + ٪ + TW = TM)

برای ایجاد تأخیر مذکور لازم است در برنامهٔ حالت AUTO بخش دیگری اضافه و در آن از تایمر SS جهت ایجاد این تأخیر استفاده کنیم.



و در PB 24 فرمان STP که PLC را به حالت STOP می‌برد استفاده شده است.

```
PB 24

SEGMENT  1              0000
0000       :STP
0001       :BE
```

۳- تهیهٔ لیستی از ابزار مورد نیاز: در این پروسه ۱۵ ورودی ۵ خروجی و ۱ تایمر و تعدادی فلگ خواهیم داشت که در ادامه آورده شده‌اند.

ورودی‌ها:

- I 22.0 : کلید STOP
- I 22.1 : کلید AUTO
- I 22.2 : کلید HAND
- I 22.3 : کلید S1
- I 22.4 : کلید S2
- I 22.5 : کلید S3
- I 22.6 : کلید S4
- I 21.0 : سنسور S8
- I 21.1 : سنسور S10
- I 21.2 : سنسور S13
- I 21.3 : سنسور S9
- I 21.4 : سنسور S11
- I 21.5 : سنسور S12
- I 21.6 : سنسور B1
- I 21.7 : کلید START

خروجی‌ها:

- Q 14.0 : فرمان باز شدن شیر Y_1 (حرکت بازوی هیدرولیک ۱)
- Q 14.1 : فرمان باز شدن شیر Y_2 (حرکت بازوی هیدرولیک ۲)
- Q 14.2 : فرمان باز شدن شیر Y_3 (حرکت بازوی هیدرولیک ۳)
- Q 14.3 : فرمان باز شدن شیر Y_4 (حرکت بازوی هیدرولیک ۴)
- Q 14.4 : LED مربوط به حالت START

فلگ‌ها:

- (پایدار) F 4.0 : فلگ مربوط به حالت Initial State
- (پایدار) F 4.1 : فلگ مربوط به مرحلهٔ اول
- (پایدار) F 4.2 : فلگ مربوط به مرحلهٔ دوم
- (پایدار) F 4.3 : فلگ مربوط به مرحلهٔ سوم
- (پایدار) F 4.4 : فلگ مربوط به مرحلهٔ چهارم
- (پایدار) F 20.0 : مربوط به حالت AUTO / HAND
- (ناپایدار) F 150.0 : مربوط به حالت START / STOP

T_4 : برای در نظر گرفتن Monitoring Time

۴- نوشتن برنامه به یکی از سه روش برنامه‌نویسی: در ادامه، تمام بلوکها آورده شده‌اند.

```
OB 1

SEGMENT  1              0000
0000       :A    T      4
0001       :JC   PB   24
0002       :A    I    21.7
0003       :S    F   150.0
0004       :A    I    22.0
0005       :R    F   150.0
0006       :A    F   150.0
0007       :=    Q    14.4
0008       :A    I    22.1
0009       :S    F    20.0
000A       :A    I    22.2
000B       :R    F    20.0
000C       :A    F    20.0
000D       :JC   PB   22
000E       :AN   F    20.0
000F       :JC   PB   23
0010       :BE
```

بلوک برنامهٔ حالت HAND:

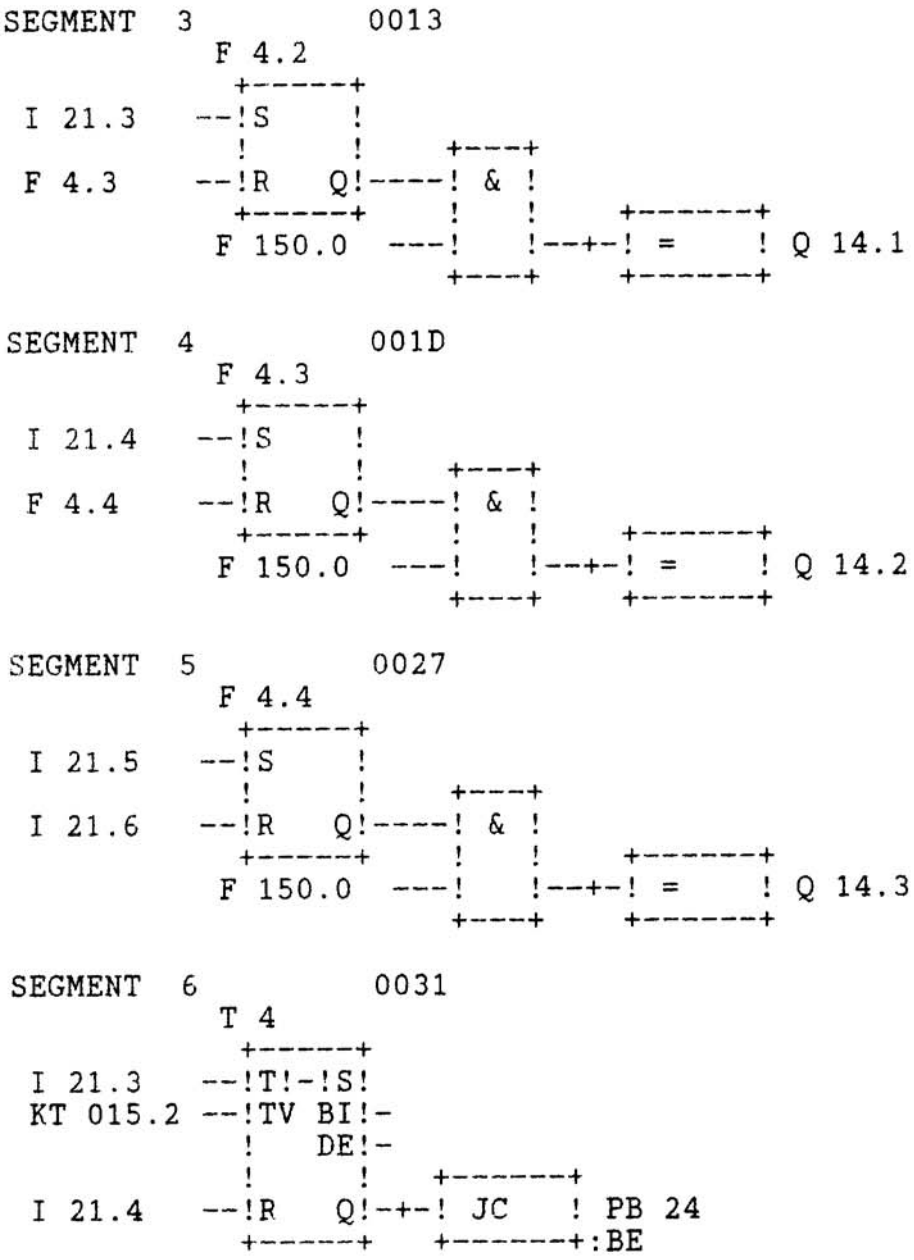
```
 PB 23

SEGMENT  1              0000
                +---+
 F 150.0  ---! & !     +-------+
 I 22.3   ---!   !--+-! =     ! Q 14.0
                +---+  +-------+

SEGMENT  2              0004
                +---+
 F 150.0  ---! & !     +-------+
 I 22.4   ---!   !--+-! =     ! Q 14.1
                +---+  +-------+
```

```
SEGMENT  3          0008
            +---+
 F 150.0  ---! & !     +------+
 I 22.5   ---!   !--+-! =    ! Q 14.2
            +---+     +------+

SEGMENT  4          000C
            +---+
 F 150.0  ---! & !     +------+
 I 22.5   ---!   !--+-! =    ! Q 14.2
            +---+     +------+

SEGMENT  5          0010
            +---+
 F 150.0  ---! & !     +------+
 I 22.6   ---!   !--+-! =    ! Q 14.3
            +---+     +------+:BE
```

بلوک برنامهٔ حالت AUTO:

```
 PB 22

SEGMENT  1          0000
            +---+
 I 21.0   ---! & !
 I 21.1   ---!   !
 I 21.2   ---!   !
 F 4.1    --O!   !
 F 4.2    --O!   !
 F 4.3    --O!   !     +------+
 F 4.4    --O!   !--+-! =    ! F 4.0
            +---+     +------+

SEGMENT  2          0009
           F 4.1
            +-----+
 F 4.0    --!S    !
           !      !    +---+
 F 4.2    --!R   Q!----! & !
            +-----+    !   !    +------+
           F 150.0  ---!   !--+-! =    ! Q 14.0
                      +---+    +------+
```

ادامهٔ بلوک PB 22:



در این قسمت قصد داریم برای درک بهتر و همچنین آشنایی با تکنیک‌های برنامه‌نویسی در PLC ، بلوک برنامهٔ حالت AUTO را تنها با استفاده از یک شمارنده بازنویسی کنیم. البته در این قسمت از فلگ استفاده نمی‌شود و شمارندهٔ مذکور باید همانند فلگ‌های استفاده شده در بلوک PB 22 از نوع پایدار انتخاب شود.

```
PB 25

SEGMENT  1                0000
0000      :A    I    21.0
0001      :A    I    21.1
0002      :A    I    21.2
0003      :A    I    21.6
0004      :R    C     5
0005      :O    I    21.3
0006      :O    I    21.4
0007      :O    I    21.5
0008      :CU   C     5
0009      :L    C     5
000A      :L    KF  +0
000C      :!=F
000D      :A    F   150.0
000E      :=    Q    14.0
000F      :L    C     5
0010      :L    KF  +1
0012      :!=F
0013      :A    F   150.0
0014      :=    Q    14.1
0015      :L    C     5
0016      :L    KF  +2
0018      :!=F
0019      :A    F   150.0
001A      :=    Q    14.2
001B      :L    KF  +3
001D      :L    C     5
001E      :!=F
001F      :A    F   150.0
0020      :=    Q    14.3
0021      :BE
```

با ایجاد تغییرات اندکی در بلوک حالت AUTO باید در OB 1 نیز تغییرات را به صورت زیر اعمال کنیم:

```
 OB 1

SEGMENT  1            0000
0000       :A    I    21.7
0001       :S    F   150.0
0002       :A    I    22.0
0003       :R    F   150.0
0004       :A    I    22.1
0005       :S    F    20.0
0006       :A    I    22.2
0007       :R    F    20.0
0008       :A    F    20.0
0009       :JC   PB   25
000A       :AN   F    20.0
000B       :JC   PB   23
000C       :BE
```

حال قصد آن داریم تا با ایجاد تغییری در برنامه، پروسه به صورت مرحله به مرحله (SINGLE STEP) انجام شود. در این برنامه می‌خواهیم روند اجرای برنامه به گونه‌ای باشد که با فشار دادن کلید (۱ + STEP) مربوط به هر مرحله تنها پروسهٔ همان مرحله انجام شود. این برنامه در PB 32 به صورت زیر آمده است. در این برنامه کلیدی اضافه می‌کنیم که در صورت فعال شدن آن، مرحلهٔ پس از مرحلهٔ فعلی انجام گیرد. به این کلید، (۱ + STEP) می‌گوئیم. این کلید با I 16.1 نشان داده شده است.

```
 PB 32

SEGMENT  1            0000
                  +---+
 I 21.0     ---! & !
 I 21.1     ---!   !
 I 21.2     ---!   !
 F 4.1      --O!   !
 F 4.2      --O!   !
 F 4.3      --O!   !     +-------+
 F 4.4      --O!   !--+-! =     ! F 4.0
                  +---+     +-------+
```

ادامهٔ بلوک PB 32:

```
SEGMENT  2          0009
            +---+    F 4.1
 F 4.0    ---! & !    +-----+
 I 16.1   ---!   !----!S    !
            +---+    !     !    +---+
           F 4.2     --!R   Q!----! & !
                       +-----+    !   !    +------+
                      F 150.0  ---!   !--+-! =    ! Q 14.0
                                 +---+    +------+

SEGMENT  3          0014
            +---+    F 4.2
 I 21.3   ---! & !    +-----+
 I 16.1   ---!   !----!S    !
            +---+    !     !    +---+
           F 4.3     --!R   Q!----! & !
                       +-----+    !   !    +------+
                      F 150.0  ---!   !--+-! =    ! Q 14.1
                                 +---+    +------+

SEGMENT  4          001F
            +---+    F 4.3
 I 21.4   ---! & !    +-----+
 I 16.1   ---!   !----!S    !
            +---+    !     !    +---+
           F 4.4     --!R   Q!----! & !
                       +-----+    !   !    +------+
                      F 150.0  ---!   !--+-! =    ! Q 14.2
                                 +---+    +------+

SEGMENT  5          002A
            +---+    F 4.4
 I 21.5   ---! & !    +-----+
 I 16.1   ---!   !----!S    !
            +---+    !     !    +---+
           I 21.6    --!R   Q!----! & !
                       +-----+    !   !    +------+
                      F 150.0  ---!   !--+-! =    ! Q 14.3
                                 +---+    +------+

SEGMENT  6          0035
           T 4
            +------+
 I 21.3   --!T!-!S!
 KT 015.2 --!TV BI!-
            !   DE!-
            !     !   +------+
 I 21.4   --!R   Q!-+-! JC   ! PB 24
            +-----+   +------+:BE
```