PID 指令

PID 回路

PID 回路指令运用以回路表中的输入和组态信息，进 行 PID 运算。
使 ENO = 0 的错误条件是：SM1.1 (溢出)，SM4.3 (运行时间)，0006 (间接寻址)
该指令影响下列特殊存储器标志位：SM1.1 (溢出)

输入/输出	操 作 数	数据类型
TBL	VB	BYTE
LOOP	常数 (0 到 7)	BYTE

PID 回路指令 (包含比例、积分、微分回路) 是用来进行 PID 运算。但是，可以进行这种 PID 运算的前提条件是逻辑堆栈栈顶 (TOS) 值必须为 1 。该指令有两个操作数：TABLE 和 LOOP。其中 TABLE 是回路表的起始地址；LOOP 是回路号，可以是 0 到 7 的整数。在程序中最多可以用 8 条 PID 指令。如果两个或两个以上的 PID 指令用了同一个回路号，那么即使这些指令的回路表不同，这些 PID 运算之间也会相互干涉，产生不可预料的结果。.
回路表包含 9 个参数，用来控制和监视 PID 运算。这些参数分别是过程变量当前值 (PVn)，过程变量前值 (PVn-1)，给定值 (SPn)，输出值 (Mn)，增益 (Kc)，采样时间 (Ts)，积分时间 (TI)，微分时间 (TD) 和积分项前值 (MX)。
为了让 PID 运算以预想的采样频率工作，PID 指令必须用在定时发生的中断程序中，或者用在主程序中被定时器所控制以一定频率执行。采样时间必须通过回路表输入到 PID 运算中。

使用 STEP 7-Micro/WIN 32 中的 PID 向导

STEP 7–Micro/WIN 32 提供了 PID 向导指导你定义一个闭环控制过程的 PID 算法。选择菜单命令 Tools>Instruction Wizard，然后从指令向导窗口中选择 PID 指令。

PID 算法

PID 控制器调节输出，保证偏差 (e) 为零，使系统达到稳定状态。偏差 (e) 是给定值 (SP) 和过程变量 (PV) 的差。PID 控制的原理基于下面的算式：输出 M (t) 是比例项、积分项和微分项的函数。其中：

$$M(t) = K_c * e + K_c \int_0^t e\,dt + M_{initial} + K_c * de/dt$$

M(t)	PID 回路的输出，是时间的函数
K_C	PID 回路的增益
e	PID 回路的偏差 (给定值与过程变量之差)
$M_{initial}$	PID 回路输出的初始值

为了能让数字计算机处理这个控制算式，连续算式必须离散化为周期采样偏差算式，才能用来计算输出值。数字计算机处理的算式如下：

$$M_n = K_C * e_n + K_I * \sum_{I}^{n} + M_{initial} + K_D * (e_n - e_{n-1})$$

输出= 比例项 + 积分项 + 微分项

其中：

M_n	在第 n 采样时刻，PID 回路输出的计算值
K_C	PID 回路增益
e_n	在第 n 采样时刻的偏差值
e_{n-1}	在第 n-1 采样时刻的偏差值 (偏差前项)
K_I	积分项的比例常数
$M_{initial}$	PID 回路输出的初值
K_D	微分项的比例常数

从这个公式可以看出，积分项是从第 1 个采样周期到当前采样周期所有误差项的函数，微分项是当前采样和前一次采样的函数，比例项仅是当前采样的函数。在数字计算机中，不保存所有的误差项，其实也不必要。

由于计算机从第一次采样开始，每有一个偏差采样值必须计算一次输出值，只需要保存偏差前值和积分项前值。利用计算机处理的重复性，可以化简以上算式为：

$$M_n = K_C * e_n + K_I * e_n + MX + K_D * (e_n - e_{n-1})$$

输出= 比例项 + 积分项 + 微分项

其中：

M_n	在第 n 采样时刻，PID 回路输出的计算值
K_C	PID 回路增益
e_n	在第 n 采样时刻的偏差值
e_{n-1}	在第 n-1 采样时刻的偏差值 (偏差前项)
K_I	积分项的比例常数
MX	积分项前值
K_D	微分项的比例常数

CPU 实际使用以上简化算式的改进形式计算 PID 输出。这个改进型算式是：

$$M_n = MP_n + MI_n + MD_n$$

输出 = 比例项 + 积分项 + 微分项

其中：

M_n	第 n 采样时刻的计算值
MP_n	第 n 采样时刻的比例项值
Mi_n	第 n 采样时刻的积分项值
MD_n	第 n 采样时刻的微分项值

比例项

比例项 MP 是增益 (K_C) 和偏差 (e) 的乘积。其中 K_C 决定输出对偏差的灵敏度，偏差 (e) 是给定值 (SP) 与过程变量值 (PV) 之差。CPU 执行的求比例项算式是：

$$MP_n = K_C * (SP_n - PV_n)$$

其中：

MP_n	第 n 采样时刻比例项的值
Kc	增益
SP_n	第 n 采样时刻的给定值
PV_n	第 n 采样时刻的过程变量值

积分项

积分项值MI **与偏差和成正比。**CPU **执行的求积分项算式是：**

$$MI_n = Kc * T_S / T_I * (SP_n - PV_n) + MX$$

其中:

Mi_n	**第** n **采样时刻的积分项值**
Kc	**增益**
T_S	**采样时间间隔**
T_I	**积分时间**
SP_n	**第** n **采样时刻的给定值**
PV_n	**第** n **采样时刻的过程变量值**
MX	**第** n-1 **采样时刻的积分项 (积分项前值) (也称积分和或偏置)**

积分和 (MX) **是所有积分项前值之和。在每次计算出** MI_n **之后，都要用** MI_n **去更新** mx**。其中** MI_n **可以被调整或限定 (详见"变量和范围"一节)。**MX **的 初值通常在第一次计算输出以前被设置为** $M_{initial}$ **(初值)。积分项还包括其他几个常数：增益** (K_C)**，采样时间间隔** (T_S) **和积分时间** (T_I)**。其中采样时间是重新计算输出的时间间隔，而积分时间控制积分项在整个输出结果中影响的大小。**

微分项

微分项值MD **与偏差的变化成正比。其计算等式为：**

$$MD_n = K_C * T_D / T_S * ((SP_n - PV_n) - (SP_{n-1} - PV_{n-1}))$$

为了避免给定值变化的微分作用而引起的跳变，假定给定值不变($SP_n=SP_{n-1}$)**。这样，可以用过程变量的变化替代偏差的变化，计算算式可改进为**:

$$MD_n = K_C * T_D / T_S * (SP_n - PV_n - SP_n + PV_{n-1})$$

或

$$MD_n = K_C * T_D / T_S * (PV_{n-1} - PV_n)$$

其中:

MD_n	**第** n **采样时刻的微分项值**
Kc	**回路增益**
Ts	**回路采样时间**
T_D	**微分时间**
SP_n	**第** n **采样时刻的给定值**
SP_{n-1}	**第** n-1 **采样时刻的给定值**
PV_n	**第** n **采样时刻的过程变量值**
PV_{n-1}	**第** n-1 **采样时刻的过程变量值**

为了下一次计算微分项值，必须保存过程变量，而不是偏差。在第一采样时刻，初始化为 PV_{n-1}=PV_n。

回路控制类型的选择

在许多控制系统中，只需要一种或二种回路控制类型。例如只需要比例回路或者比例积分回路。通过设置常量参数，可先选中想要的回路控制类型。
如果不想要积分回路，可以把积分时间设为无穷大。即使没有积分作用，积分项还是不为零，因为有初值 MX**。**
如果不想要微分回路，可以把微分时间置为零。
如果不想要比例回路，但需要积分或积分微分回路，可以把增益设为 0.0**，系统会在计算积分项和微分项时，把增益当作** 1.0 **看待。**

回路输入的转换和标准化

每个 PID **回路有两个输入量，给定值** (SP) **和过程变量** (PV)**。给定值通常是一个固定的值，比如是设定的汽车速度。过程变量是与** PID **回路输出有关，可以衡量输出对控制系统作用的大小。在汽车速度控制系统中，过程变量可以是测速仪的输入** (**衡量车轮转速高低**)**。**
给定值和过程变量都可能是现实世界的值，它们的大小、范围和工程单位都可能不一样。PID **指令在对这些量进行运算以前，必须把他们转换成标准的浮点型实数。**
转换的第一步是把 16 **位整数值转成浮点型实数值。下面的指令序列提供了实现这种转换的方法：**

```
XORD    AC0，AC0        //清空累加器。
MOVW    AIW0，AC0       //把待变换的模拟量存入累加器。
LDW>=   AC0，0          //如果模拟量为正
JMP     0               //则直接转成实数
NOT                     //否则
ORD     16#FFFF0000，AC0  //先对 AC0 中值进行 符号扩展
LBL     0
DTR     AC0，AC0        //把 32 位整数转成实数
```

转换的下一步是把实数值进一步标准化为 0.0~1.0 **之间的实数。下面的算式可以用来标准化给定值或过程变量：**

$$R_{Norm} = (R_{Raw} / Span) + Offset)$$

其中：

R_{Norm} **标准化的实数值**

R_{raw} **没有标准化的实数值或原值**

Offset **单极性为** 0.0**，双极性为** 0.5

Span **值域大小，可能最大值减去可能最小值**
单极性为 32,000 (**典型值**)
双极性为 64,000 (**典型值**)

下面的指令把双极性实数标准化为 0.0~1.0 **之间的实数。通常用在第一步转换之后**:

```
/R      64000.0，AC0    //累加器中的标准化值
+R      0.5，AC0        //加上偏置，使其落在 0.0~1.0 之间
MOVR    AC0，VD100      //标准化的值存入回路表
```

回路输出值转换成刻度整数值

回路输出值一般是控制变量，比如，在汽车速度控制中，可以是油阀开度的设置。同时，输出是 0.0～1.0 **之间的标准化了的实数值，在回路输出驱动模拟输出之前，必须把回路输出转换成相应的** 16 **位整数。这一过程，是给定值或过程变量的标准化转换的反过程。该过程的第一步把回路输出转换成相应的实数值，公式如下：**

$$R_{Scal} = (M_n - Offset) * Span$$

其中：

R_{scal} **回路输出的刻度实数值**

M_n **回路输出的标准化实数值**

Offset **单极性为** 0.0，**双极性为** 0.5

Span **值域大小，可能最大值减去可能最小值**

单极性为 32,000 (**典型值**)

双极性为 64,000 (**典型值**)

这一过程可以用下面的指令序列完成：

```
MOVR    VD108,AC0      //把回路输 出值移入累加器
–R      0.5,AC0        //仅双极性有此句
*R      64000.0,AC0    //在累加器中得到刻度值
```

下一步是把回路输出的刻度转换成 16 **位整数，可通过下面的指令序列来完成：**

```
ROUND   AC0 AC0        //把实数转换为 32 位整数
MOVW    AC0, AQW0      //把 16 位整数写入模拟输出寄存器
```

正作用或反作用回路

如果增益为正，那么该回路为正作用回路。如果增益为负，那么是反作用回路。对于增益为零的积分或微分控制来说，如果指定积分时间、微分时间为正，就是正作用回路；指定为负值，则是反作用回路。

变量和范围

过程变量和给定值是 PID **运算的输入值，因此在回路表中，这些值只能被回路指令读而不能改写。输出变量是由** PID **运算产生的，所以在每一次** PID **运算完成之后，需更新回路表中的输出值，输出值被限定在** 0.0～1.0 **之间。当** PID **指令从手动方式转变到自动方式时，回路表中的输出值可以用来初始化输出值 (有关** PID **指令的方式详见下面的“控制方式” 一节)。**

如果使用积分控制，积分项前值要根据 PID **运算结果更新。这个更新了的值用作下一次** PID **运算的输入，当输出值超过范围 (大于** 1.0 **或小于** 0.0)，**那么积分项前值必须根据下列公式进行调整：**

$MX = 1.0 - (MP_n + MD_n)$ **当计算输出** $M_n > 1.0$

或

$MX = -(MP_n + MD_n)$ **当计算输出** $M_n < 0.0$

其中:

MX 经过调整了的积分和 (积分项前值)

MP_n 第 n 采样时刻的比例项值

MD_n 第 n 采样时刻的微分项值

M_n 第 n 采样时刻的输出值

这样调整积分前值，一旦输出回到范围后，可以提高系统的响应性能。而且积分项前值也要限制在 0.0～1.0 之间，然后在每次 PID 运算结束之后，把积分项前值写入回路表， 以备在下次 PID 运算中使用。

用户可以在执行 PID 指令以前修改回路表中积分项前值。在实际运用中， 这样做的目的是找到由于积分项前值引起的问题。手工调整积分项前值时，必须小心谨慎，还应保证写入的值在 0.0～1.0 之间。

回路表中的给定值与过程变量的差值 (e) 是用于 PID 运算中的差分运算，用户最好不要去修改此值。

控制方式

S7–200 的 PID 回路没有设置控制方式，只要 PID 块有效，就可以执行 PID 运算 。在这种意义上说，PID 运算存在一种"自动"运行方式。当 PID 运算不被执行时， 我们称之为"手动"方式。

同计数器指令相似，PID 指令有一个使能位。当该使能位检测到一个信号的正跳变 (从 0 到 1)，PID 指令执行一系列的动作，使 PID 指令从手动方式无扰动地切换到自动方式。为了达到无扰动切换，在转变到自动控制前，必须用手动方式把当前输出值填入回路表中的 M_n 栏。PID 指令对回路表中的值进行下列动作，以保证当使能位正跳变出现时， 从手动方式无扰动切换到自动方式:

- 置给定值 (SP_n) = 过程变量 (PV_n)
- 置过量变量前值 (PV_{n-1}) = 过程变量现值 (PV_n)
- 置积分项前值 (MX) = 输出值 (M_n)

PID 使能位的默认值是 1，在 CPU 启动或从 STOP 方式转到 RUN 方式时建立。CPU 进入 RUN 方式后首次使 PID 块有效，没有检测到使能位的正跳变，那么就没有无扰动切换的动作。

报警与特殊操作

PID 指令是执行 PID 运算的简单而功能强大的指令。如果其他过程需要对回路变量进行 报警等特殊操作，那么可以用 CPU 支持的基本指令实现这些特殊操作功能。

出错条件

如果指令指定的回路表起始地址以及回路号操作数超出范围，那么在编译期间， CPU 令产生编译错误 (范围错误)，从而编译失败。PID 指令不检查回路表中的值是否在范围之内，所以必须小心操作以保证过程变量和设定值不超界。

PID 指令不检查回路表中的值是否超界，你必须保证过程变量和设定值 (以及偏置和前一次过程变量) 必须在 0.0 到 1.0 之间。

如果 PID 计算的算术运算发生错误，那么特殊存储器标志位 SM1.1 (溢出或非法值) 会被置 1，并且中止 PID 指令的执行。 (要想消除这种错误，单靠改变回路表中的输出值是不够的，正确的方法是在下一次执行 PID 运算之前，改变引起算术运算错误的输入值，而不是更新输出值)。

回路表

36 个字节的回路表的格式如表 9-19 所示。

表 9-19 回路表格式

偏移地址	域	格式	类型	描 述
0	过程变量 (PV_n)	双字 - 实数	输入	过程变量，必须在 0.0~1.0 之间
4	设定值 (SP_n)	双字 - 实数	输入	给定值，必须在 0.0~1.0 之间
8	输出值 (M_n)	双字 - 实数	输入/输出	输出值，必须在 0.0~1.0 之间
12	增益 (K_C)	双字 - 实数	输入	增益是比例常数，可正可负
16	采样时间 (T_S)	双字 - 实数	输入	单位为秒，必须是正数
20	积分时间 (T_I)	双字 - 实数	输入	单位为分钟，必须是正数
24	微分时间 (T_D)	双字 - 实数	输入	单位为分钟，必须是正数
28	积分项前项 (MX)	双字 - 实数	输入/输出	积分项前项，必须在 0.0~1.0 之间
32	过程变量前值 (PV_{n-1})	双字 - 实数	输入/输出	最近一次 PID 运算的过程变量值

PID 指令编程举例

在本例中，有一水箱需要维持一定的水位，该水箱里的水以变化的速度流出。这就需要有一个水泵以不同的速度给水箱供水，以维持水位不变，这样才能使水箱不断水。

本系统的给定值是水箱满水位的 75% 时的水位，过程变量由漂浮在水面的水位测量仪给出。输出值是进水泵的速度，可以从允许最大值的 0% 变到 100%。

给定值可以预先设定后直接输入到回路表中，过程变量值是来自水位表的单极性模拟量，回路输出值也是一个单极性模拟量，用来控制进水泵速度。这两个模拟量的范围是 0.0~1.0，分辨率为 1/32000 (标准化)。

在本系统中，只使用比例和积分控制，其回路增益和时间常数可以通过工程计算初步确定。但还需要进一步调整以达到最优控制效果。初步确定的增益和时间常数为：

K_C 是 0.25

T_S 是 0.1 秒

T_I 是 30 分钟

系统启动时，关闭出水口，用手动控制进水泵速度，使水位达到满水位的 75%。然后打开出水口，同时水泵控制从手动方式切换到自动方式。这种切换由一个输入的数字量控制，描述如下：

I0.0 位控制手动到自动的切换，0 代表手动；1 代表自动。

当工作在手动控制方式下，可以把水泵速度 (0.0~1.0 之间的实数) 写到 VD108 (VD108 是回路表中保存输出的寄存器)。

图 9-28 **是本控制实例的程序。**

LAD	STL
MAIN OB1	
Network 1 SM0.1 — SBR0 EN	Network 1 LD SM0.1 //在首次扫描调用初始化子程序 0 CALL 0
SUBROUTINE 0	

Network 1
SM0.0
MOV_R EN ENO
0.75 — IN OUT — VD104
MOV_R EN ENO
0.25 — IN OUT — VD112
MOV_R EN ENO
0.10 — IN OUT — VD116
MOV_R EN ENO
30.0 — IN OUT — VD120
MOV_R EN ENO
0.0 — IN OUT — VD124
MOV_B EN ENO
100 — IN OUT — SMB34
ATCH EN ENO
0 — INT
10 — EVNT
(ENI)
	Network 1
	LD SM0.0
	MOVR 0.75, VD104 //装入设定值75%
	MOVR 0.25, VD112 //装入回路增益=0.25
	MOVR 0.10, VD116 //装入采样时间=0.1秒
	MOVR 30.0, VD120 //装入积分时间=30分
	MOVR 0.0, VD124 //关闭微分作用
	MOVB 100, SMB34 //设定定时中断0发生的时间
	//间隔为100ms
	//设置定时中断以执行PID计算
	ATCH 0, 10 //允许中断
	ENI //子程序0结束

图 9-28 PID **回路控制实例**

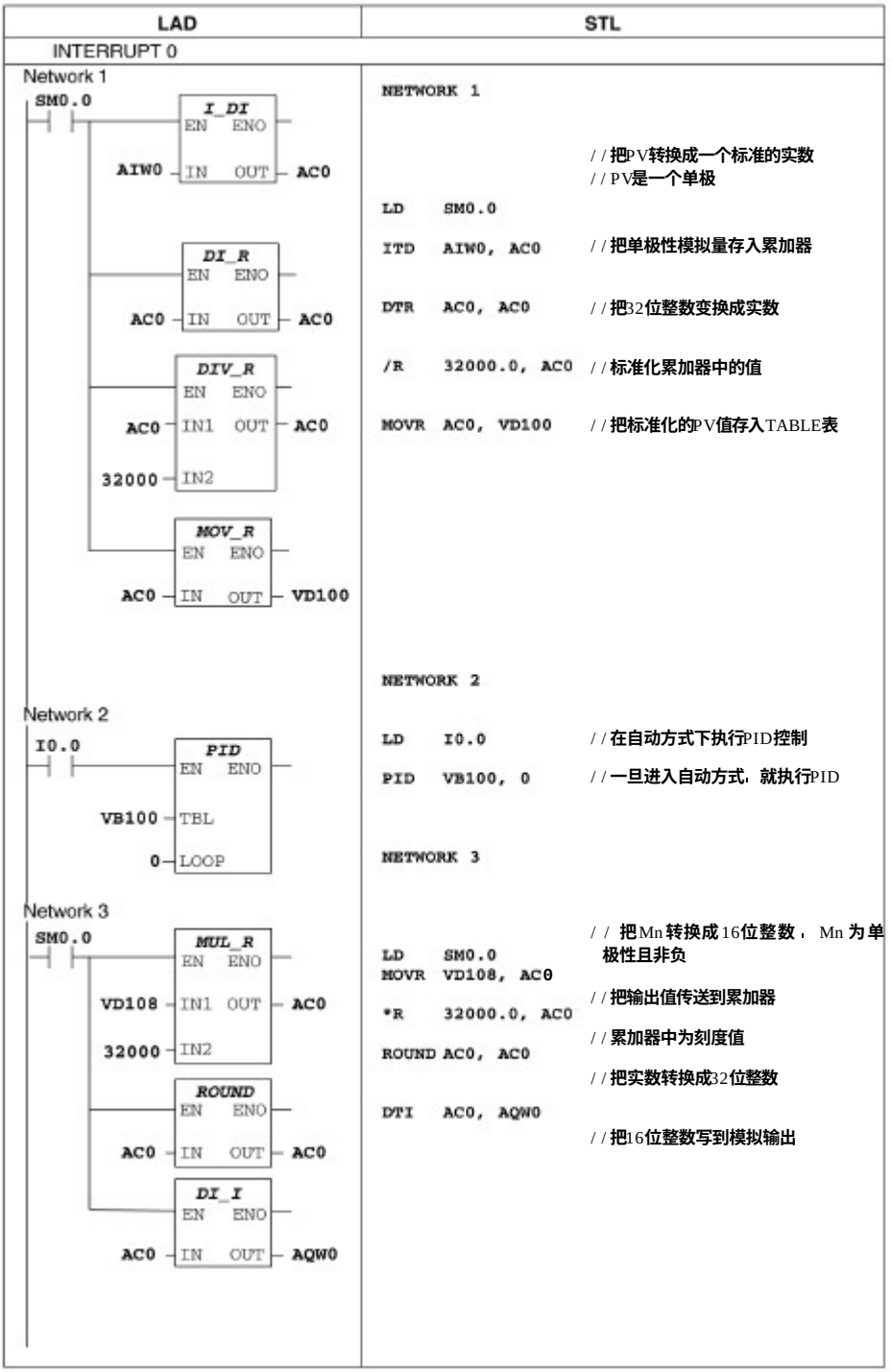

图 9-28 PID **回路控制实例 (续)**

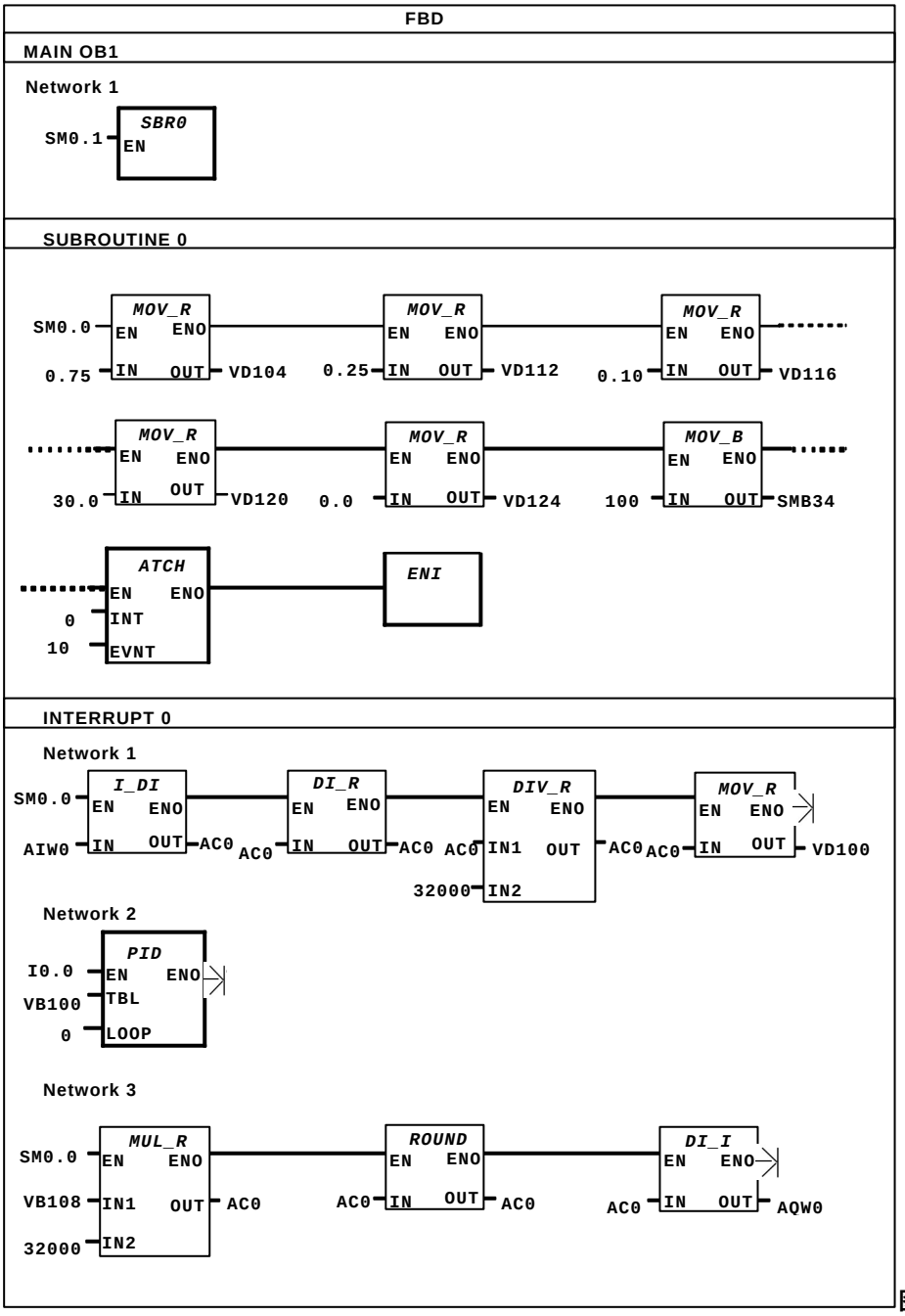

图9-28 PID**回路控制实例 (续)**