Topic: The DPID/DPIDE function of AH500 Series PLCs (Examples of temperature control)

Applicable model	AH500 series
Keyword	DPID function, DPIDE function

Table of Contents

1. Preface and Purpose

Preface:
Proportional-integral-derivative controller (PID controller) is widely applied in the field of engineering. It has been presented for nearly sixty years. It is a main technical tool used in industrial control systems because its structure is simple, it is stable and reliable, and it can be easily adjusted.

Purpose:
If users use PID control for the first time, they may not be familiar with the characteristics of the PID control. The document helps users understand the principle and the usage of PID control.

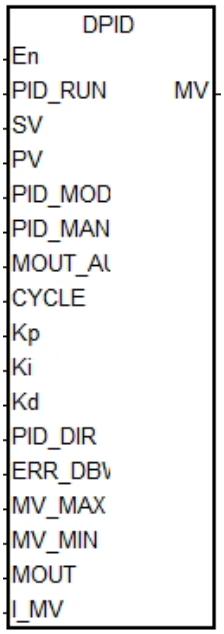
2. Description of DPID

API	Instruction code			Operand	Function
0707	D	PID		**PID_RUN, SV, PV, PID_MODE, PID_MAN, MOUT_AUTO, CYCLE, Kp, Ki, Kd, PID_DIR, ERR_DBW, MV_MAX, MV_MIN, MOUT, I_MV, MV**	PID algorithm

Device	X	Y	M	S	T	C	HC	D	L	SM	SR	E	PR	K	16#	"$"	DF
PID_RUN	●	●	●					●	●	●			●				
SV								●	●				●				
PV								●	●				●				
PID_MODE								●	●				●				
PID_MAN	●	●	●					●	●	●			●				
MOUT_AUTO	●	●	●					●	●	●			●				
CYCLE								●	●				●				
Kp								●	●				●				
Ki								●	●				●				
Kd								●	●				●				
PID_DIR	●	●	●					●	●	●			●				
ERR_DBW								●	●				●				
MV_MAX								●	●				●				
MV_MIN								●	●				●				
MOUT								●	●				●				
I_MV								●	●				●				
MV								●	●				●				

Pulse instruction	16-bit instruction	32-bit instruction (35 steps)
-	-	AH500

Symbol:



PID_RUN	:	Enabling the PID algorithm	Bit
SV	:	Target value (SV)	Double word
PV	:	Present value (PV)	Double word
PID_MODE	:	PID control mode	Double word
PID_MAN	:	PID A/M mode (PID_MAN)	Bit
MOUT_AUTO	:	MOUT automatic change mode	Bit
CYCLE	:	Sampling time (CYCLE)	Double word
Kp	:	Proportional gain (K_p)	Double word
Ki	:	Integral gain (K_i)	Double word
Kd	:	Derivative gain (K_d)	Double word
PID_DIR	:	PID forward/reverse direction (PID_DIR)	Bit
ERR_DBW	:	Error dead bandwidth	Double word
MV_MAX	:	Maximum output value (MV_MAX)	Double word
MV_MIN	:	Minimum output value (MV_MIN)	Double word
MOUT	:	Manual output value (MOUT)	Double word
I_MV	:	Accumulated integral value (I_MV)	Double word
MV	:	Output value (MV)	Double word

Explanation:

1. The instruction is used to implement a PID algorithm. After the sampling time set is reached, the PID algorithm will be implemented. PID stands for Proportional, Integral, Derivative. PID control is widely applied to mechanical equipment, pneumatic equipment, and electronic equipment.
2. The setting of the parameters is described below.

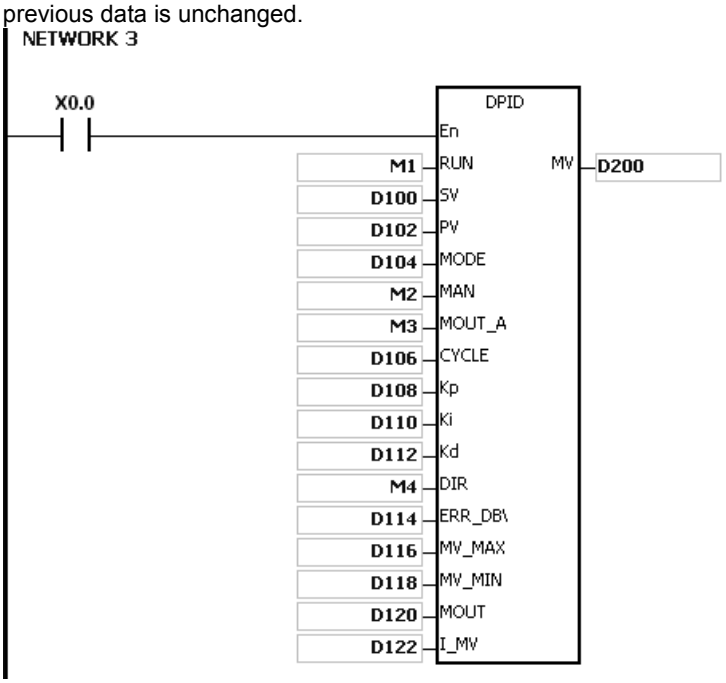
Device number	Data type	Function	Setting range	Description
PID_RUN	BOOL	Enabling the PID algorithm	True: The PID algorithm is implemented. False: The output value (MV) is reset to 0, and the PID algorithm is not implemented.	
SV	REAL	SV	Range of single-precision floating-point values	Target value
PV	REAL	PV	Range of single-precision floating-point values	Present value

Device number	Data type	Function	Setting range	Description
PID_MODE	DWORD/ DINT	PID control mode	0: Automatic control When PID_MAN is turned from true to false, the output value (MV) then is involved in the automatic algorithm. 1: Parameters are tuned automatically. After the tuning of the parameters is complete, the value will be changed to 0 automatically, and the approriate parameters Kp, Ki, and Kd will be calculated. 2: Automatic control When PID_MAN is switched from ON to OFF, the MV involved in the internal algorithm is involved in the automatic algorithm. If the setting value is out of the range, it will be counts as 0.	
PID_MAN	BOOL	PID A/M mode	True: Manual The MV is output according to the MOUT, but it is still in the range of the MV_MIN to the MV_MAX. When PID_MODE is set to 1, the setting is ineffective. False: Automatic The MV is output according to the PID algorithm, and the output value is in the range of MV_MIN to MV_MAX.	
MOUT_AUTO	BOOL	MOUT automatic change mode	True: Automatic The MOUT varies with the MV. False: Normal The MOUT deos not vary with the MV.	
Cycle	DWORD/ DINT	Sampling time (T_S)	1~2,000 (unit: 10 ms)	When the instruction is scanned, the PID algorithm is implmented according to the sampling time, and the MV is refreshed. (Whether the scan time reaches the sampling time is not calculated.) If T_S is less than 1, it will be count as 1. If T_S is greater than 2000, it will be count as 2000. When the instruction PID is used in the time interrupt task, the sampling time is the same as the interval between the time interrupt tasks.
Kp	REAL	Proportional gain (K_p)	Range of positive single-precision floating-point values	It is the magnified proportional value of the error between an SV and a PV. If the magnified proportional value of the error is less than 0, the K_p will be count as 0.
Ki	REAL	Integral gain (K_i)	Range of positive single-precision floating-point values	It is the integral gain (K_i). If the integral gain is less than 0, the K_i will be count as 0.

Device number	Data type	Function		Setting range	Description
Kd	REAL	Derivative gain (K_d)		Range of positive single-precision floating-point values	It is the derivative gain (K_d). If the derivative gain is less than 0, the K_d will be count as 0.
PID_DIR	BOOL	PID forward/reverse direction		True: Reverse action (E=SV-PV) OFF: Forward action (E=PV-SV)	
ERR_DBW	REAL	Error dead bandwidth: Range within which an error (E) is count as 0		Range of single-precision floating-point values	An error (E) is the difference between an SV and a PV. If **ERR_DBW** is set to K0, the function will not be enabled. For example, after **S_3**+5 is set to 5 or -5, an E will be count as 0 if it is between -5 and 5.
MV_MAX	REAL	Maximum output value		Range of single-precision floating-point values	Example: After **MV_MAX** is set to 1,000, an MV will be 1,000 if it exceeds 1,000. **MV_MAX** has to be greater than **MV_MIN**, otherwise the maximum output value set and the minimum output value set will be interchanged.
MV_MIN	REAL	Minimum output value		Range of single-precision floating-point values	Example: After **MV_MIN** is set to -1,000, an MV will be -1,000 if it is less than than -1,000.
MOUT	REAL	Manual output value		It is used with the PID_MAN mode. Users set the MV directly.	
I_MV (It occupies six consecutive 32-bit devices.)	REAL	I_MV	Accumulated integral value temporarily stored	Range of single-precision floating-point values	An accumulated integral value is usually for reference. Users can still clear or modify it according to their needs. When the MV is greater than the MV_MAX, or when the MV is less than MV_MIN, the accumulated integral value in I_MV is unchanged.
		I_MV+1	Previous PV temporarily stored	The previous PV is usually for reference. Users can still clear or modify it according to their needs.	
		I_MV+2 ~ I_MV+5	For system use only		
MV	REAL	MV	The MV is in the range of the MV_MIN to the MV_MAX.		

Example:

1. Before the instruction PID is executed, the setting of the parameters should be complete.
2. When X0.0 is ON, the instruction is executed. When M1 is ON, the PID algorithm is implemented. When M1 is OFF, the MV is 0, and the MV is stored in D200. When X0.0 is switched OFF, the instruction is not executed, and the previous data is unchanged.



Additional remark:

1. The instruction can be used several times, but the registers specified by **I_MV~I_MV**+5 can not be the same.
2. **I_MV** occupies 12 registers. **I_MV** used in the instruction PID in the example above occupies D122~D133.
3. The instruction PID can only be used in the cyclic task and the time interrupt task. When the instruction PID is used in the time interrupt task, the sampling time is the same as the interval between the time interrupt tasks.
4. When the instruction is scanned, the PID algorithm is implmented according to the sampling time, and the MV is directly refreshed. Whether the scan time reaches the sampling time is not calculated automatically.When the instruction is used in the time interrupt, the sampling time is the same as the interval between the time interrupts. The PID algorithm is implemented according to the interval between the time interrupts.
5. Before the PID algorithm is implemented, the present value used in the instruction PID has to be a stable value. When users need the input value in the module to implement the PID algorithm, they have to notice the time it takes for the analog input to be converted into the digital input.

PID algorithm:

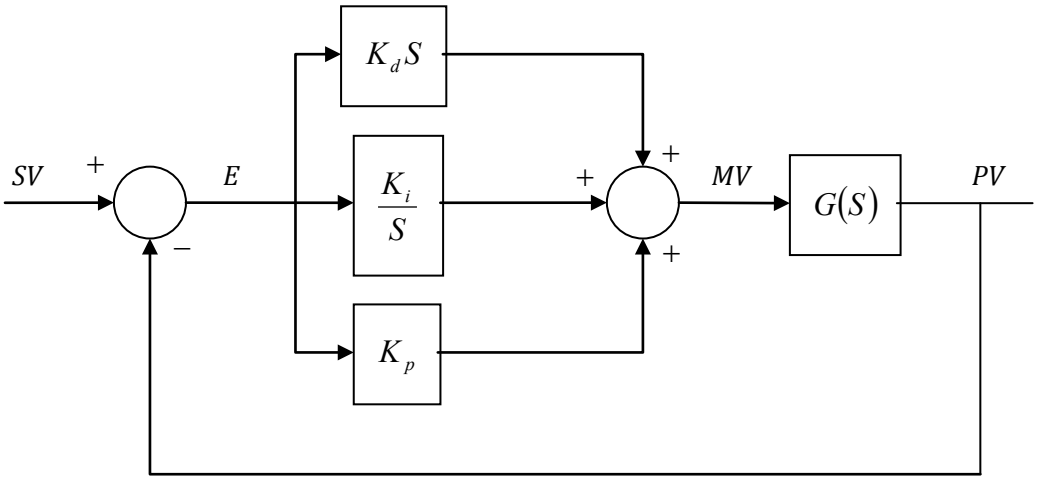
1. When **PID_MODE** is set to 0 or 2, the PID control mode is the automatic control mode.
2. When **PID_MODE** is set to 1, the PID control mode is the automatic tuning mode. After the tuning of the parameter is complete, **PID_MODE** is set to 0. The PID control mode becomes the automatic control mode.
   a) The PID algorithm includes the forward action and the reverse action. Whether the action is the forward one or the reverse one depends on the setting of **PID_DIR**.
   The PID algorithm is as follows.

$$MV = K_P E(t) + K_I \int_0^t E(t)dt + K_D * \frac{dE(t)}{dt}$$

E(t)S represents the derivative value of E(t), and $E(t)\frac{1}{S}$ represents the integral value of $E(t)$.

Action direction	PID algorithm
Forward action	$E(t) = PV(t) - SV(t)$
Reverse action	$E(t) = SV(t) - PV(t)$

- Control diagram:



- Symbols:

MV : Output value

$E(t)$: Error

Forward action $E(t) = PV - SV$

Reverse action $E(t) = SV - PV$

K_P : Proportional gain

PV : Present value

SV : Target value

K_D : Derivative gain

K_I : Integral gain

Suggestion:

1. Owing to the fact that the instruction PID can be used in a lot of controlled environments, users have to select control functions appropriately. For example, if **PID_MODE** is set to 1, the instruction can not be used in a motor control environment, otherwise improper control may occur.
2. When users tune the parameters K_p, K_i, and K_d (**PID_MODE** is set to 0 or 2), they have to tune the K_p first (according to their experiences), and then set set the K_i and the K_d to 0. When the users can handle the control, they can increase the K_i and the K_d,. Please see example four below. When the K_p is 1, it means that the proportional gain is 100%. That is, the error is increased by a factor of one. When the proportional gain is less than 100%, the error is decreased. When th proportional gain is greater than 100%, the error is increased.
3. To prevent the parameters which have been tuned automatically from disappearing after a power cut, it is suggested that users should store the parameters in latching data registers if **PID_MODE** is set to 1. The parameters which have been tuned automatically are not necessarily suitable for every controlled environment. Therefore, the users can modify the parameters which have been tuned automatically. However, it is suggested that users only modify the K_i or the K_d.
4. The action of the instruction depends on many parameters. To prevent improper control from occurring, please do not set parameters randomly.

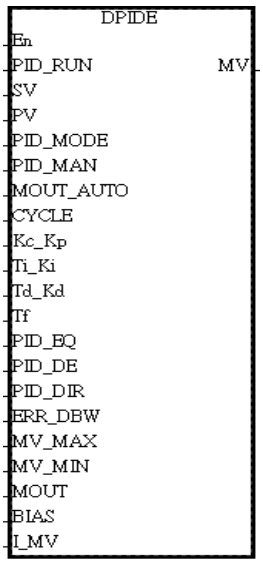
3. Description of DPIDE

API	Instruction code			Operand	Function
0708	D	PIDE		**PID_RUN, SV, PV, PID_MODE, PID_MAN, MOUT_AUTO, CYCLE, Kc_Kp, Ti_Ki, Td_Kd, Tf, PID_EQ, PID_DE, PID_DIR, ERR_DBW, MV_MAX, MV_MIN, MOUT, BIAS, I_MV, MV**	PID algorithm

Device	X	Y	M	S	T	C	HC	D	L	SM	SR	E	PR	K	16#	"$"	DF
PID_RUN	●	●	●					●	●	●			●				
SV								●	●				●				●
PV								●	●				●				●
PID_MODE								●	●				●				
PID_MAN	●	●	●					●	●	●			●				
MOUT_AUTO	●	●	●					●	●	●			●				
CYCLE								●	●				●				
KC_Kp								●	●				●				
Ti_Ki								●	●				●				
Td_Kd								●	●				●				
Tf								●	●				●				
PID_EQ	●	●	●					●	●	●			●				
PID_DE	●	●	●					●	●	●			●				
PID_DIR	●	●	●					●	●	●			●				
ERR_DBW								●	●				●				●
MV_MAX								●	●				●				●
MV_MIN								●	●				●				●
MOUT								●	●				●				
BIAS								●	●				●				●
I_MV								●	●				●				
MV								●	●				●				

Pulse instruction	16-bit instruction	32-bit instruction (43 steps)
-	-	AH500

Symbol:



PID_RUN	:	Enabling the PID algorithm	Bit
SV	:	Target value (SV)	Double word
PV	:	Present value (PV)	Double word
PID_MODE	:	PID control mode	Double word
PID_MAN	:	PID automatic/manual mode	Bit
MOUT_AUTO	:	MOUT automatic change mode	Bit
CYCLE	:	Sampling time (millisecond)	Double word
Kc_Kp	:	Proportional gain	Double word
Ti_Ki	:	Integral gain (second or 1/seccond)	Double word
Td_Kd	:	Derivative gain (second)	Double word
Tf	:	Derivative action time constant (second)	Double word
PID_EQ	:	Selection of a PID formula	Bit
PID_DE	:	Selection of the calculation of the PID derivative error	Bit
PID_DIR	:	PID forward/reverse direction	Bit
ERR_DBW	:	Error dead bandwidth	Double word
MV_MAX	:	Maximum output value	Double word
MV_MIN	:	Minimum output value	Double word
MOUT	:	Manual output value	Double word
BIAS	:	Feedforward output value	Double word
I_MV	:	Accumulated integral value	Double word
MV	:	Output value	Double word

Explanation:

1. The instruction is used to implement an advanced PID algorithm. The PID algorithm is implemented only when the instruction is executed. PID stands for Proportional, Integral, Derivative. PID control is widely applied to mechanical equipment, pneumatic equipment, and electronic equipment.
2. The setting of the parameters is described below.

Device number	Data type	Function	Setting range	Description
PID_RUN	BOOL	Enabling the PID algorithm	True: The PID algorithm is implemented. False: The output value (MV) is reset to 0, and the PID algorithm is not implemented.	

Device number	Data type	Function	Setting range	Description
SV	REAL	SV	Range of single-precision floating-point values	Target value
PV	REAL	PV	Range of single-precision floating-point values	Present value
PID_MODE	DWORD/ DINT	PID control mode	0: Automatic control When PID_MAN is turned from true to false, the output value (MV) then is involved in the automatic algorithm. 1: Parameters are tuned automatically. After the tuning of the parameters is complete, the value will be changed to 0 automatically, and the approriate parameters Kc_Kp, Ti_Ki, Td_Kd, and Tf will be calculated.	
PID_MAN	BOOL	PID A/M mode	True: Manual The MV is output according to the MOUT, but it is still in the range of the MV_MIN to the MV_MAX. When PID_MODE is set to 1, the setting is ineffective. False: Automatic The MV is output according to the PID algorithm, and the output value is in the range of MV_MIN to MV_MAX.	
MOUT_AUTO	BOOL	MOUT automatic change mode	True: Automatic The MOUT varies with the MV. False: Normal The MOUT deos not vary with the MV.	

Device number	Data type	Function	Setting range	Description
CYCLE	DWORD/DINT	Sampling time (T_S)	1~40,000 (unit: ms)	When the instruction is scanned, the PID algorithm is implmented according to the sampling time, and the MV is refreshed. (The CPU module does not automatically judge time and execute the instruciton according to the sampling time.) If CYCLE is less than 1, it will be count as 1. If CYCLE is greater than 40,000, it will be count as 40,000. When the instruction is used in the time interrupt, the CPU module automatically implements the PID algorithm according to the interval between the tiimed interrupts, and the setting of CYCLE does not work.
Kc_Kp	REAL	Proportional gain (K_c or K_p) (The selection of K_c or K_p depends on the setting of the parameter PID_EQ.)	Range of positive single-precision floating-point values	It is a proportional gain. If a proportional gain is less than 0, Kc_Kp will be count as 0. If Kc_Kp is equal to 0 when the independent formula is used, the proportional control is not used.
Ti_Ki	REAL	Integral gain (T_i or K_i) (The selection of T_i or K_i depends on the setting of the parameter PID_EQ.)	Range of positive single-precision floating-point values (Unit: T_i=Second; K_i=1/second)	It is an integral gain. If an integral gain is less than 0, Ti_Ki will be count as 0. If Ti_Ki is equal to 0, the integral control is not used.
Td_Kd	REAL	Derivative gain (T_d or K_d) (The selection of T_d or K_d depends on the setting of the parameter PID_EQ.)	Range of positive single-precision floating-point values (Unit: Second)	It is a derivative gain. If a derivative gain is less than 0, Td_Kd will be count as 0. If Td_Kd is equal to 0, the derivative control is not used.

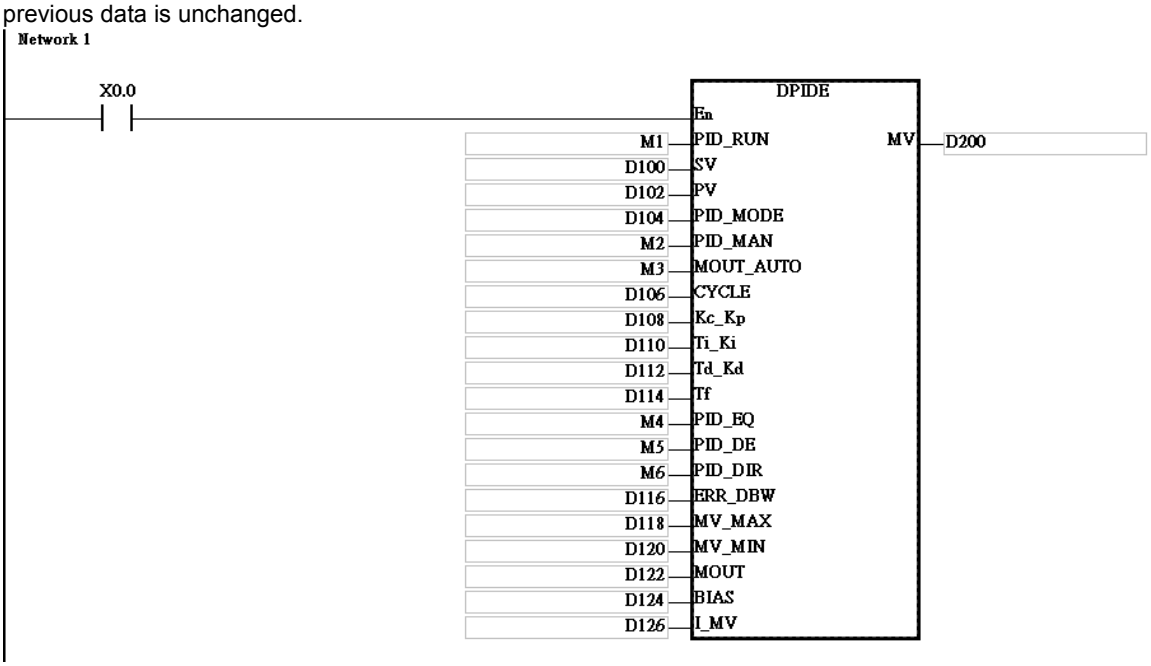
Device number	Data type	Function	Setting range	Description
Tf	REAL	Derivative action time constant (T_f)	Range of positive single-precision floating-point values (Unit: Second)	It the derivative action time constant. If the derivative action time constant is less than 0, Tf will be count as 0. If Tf is equal to 0, the derivative action time control is not used. (Derivative smoothing)
PID_EQ	BOOL	Selection of a PID formula	True: Dependent formula False: Independent Formula	
PID_DE	BOOL	Selection of the calculation of the PID derivative error	True: Using the variations in the PV to calculate the control value of the derivative (Derivative of the PV) False: Using the variations in the error (E) to calculate the control value of the derivative (Derivative of the error)	
PID_DIR	BOOL	PID forward/reverse direction	True: Reverse action (E=SV-PV) False: Forward action (E=PV-SV)	
ERR_DBW	REAL	Error dead bandwidth: Range within which an error (E) is count as 0	Range of single-precision floating-point values	An error (E) is equal to SV−PV or PV-SV. If the setting value is 0, the function will not be enabled, otherwise the CPU module will check whether the present error is less than the absolute value of ERR_DBW, and check whether the present error meets the cross status condition. If the present error is less than the absolute value of ERR_DBW, and meets the cross status condition, the present error will be count as 0, and the PID algorithm will be implemented, otherwise the present error will be brought into the PID algorithm according to the normal processing.

Device number	Data type	Function	Setting range	Description
MV_MAX	REAL	Maximum output value	Range of single-precision floating-point values	Example: After **MV_MAX** is set to 1,000, an MV will be 1,000 if it exceeds 1,000. **MV_MAX** has to be greater than **MV_MIN**, otherwise the maximum output value set and the minimum output value set will be interchanged.
MV_MIN	REAL	Minimum output value	Range of single-precision floating-point values	Example: After **MV_MIN** is set to -1,000, an MV will be -1,000 if it is less than than -1,000. **MV_MIN** has to be lessthan **MV_MAX**, otherwise the maximum output value set and the minimum output value set will be interchanged.
MOUT	REAL	Manual output value	Range of single-precision floating-point values	Mout and PID_MAN are used together. If PID_MAN is set to true, the MV will be output according to the MOUT, but it will be still in the range of the MV_MIN to the MV_MAX.
BIAS	REAL	Feedforward output value	Range of single-precision floating-point values	It is used for the PID feedforward.

Device number	Data type	Function		Setting range	Description
I_MV (It occupies ten consecutive 32-bit devices.)	REAL	I_MV	Accumulated integral value temporarily stored	Range of single-precision floating-point values	An accumulated integral value is usually for reference. Users can still clear or modify it according to their needs. When the MV is greater than the MV_MAX, or when the MV is less than MV_MIN, the accumulated integral value in I_MV is unchanged.
		I_MV+1	Previous error temporarily stored	The system records the previous error.	
		I_MV+2~ I_MV+5	For system use only		
		I_MV+6	The system records the previous PV.		
		I_MV+7~ I_MV+9	For system use only		
MV	REAL	MV	The MV is in the range of the MV_MIN to the MV_MAX.		

Example:

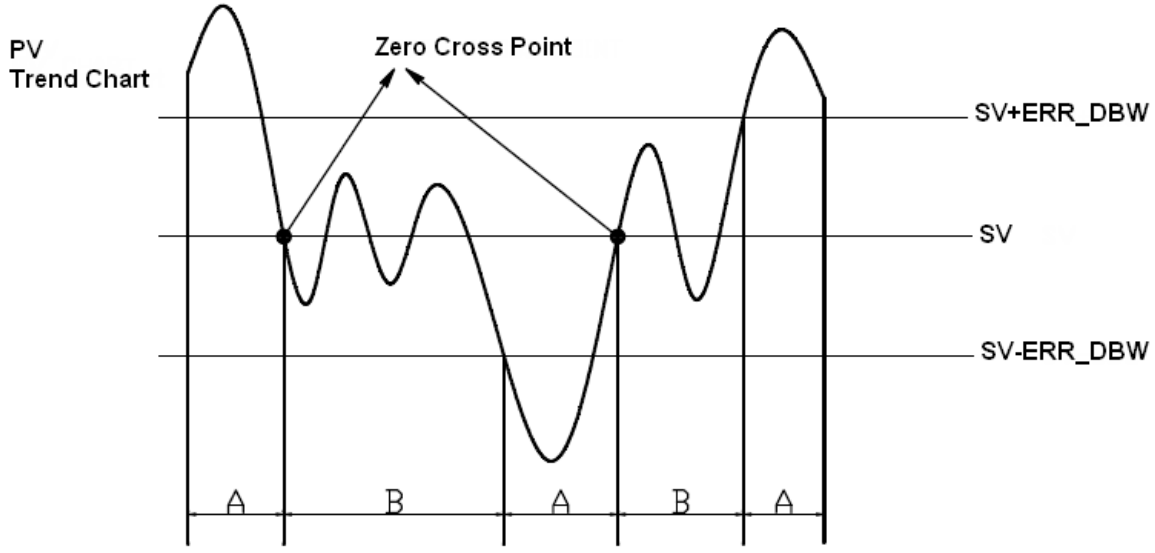
1. Before the instruction DPIDE is executed, the setting of the parameters should be complete.
2. When X0.0 is ON, the instruction is executed. When M1 is ON, the DPIDE algorithm is implemented. When M1 is OFF, the MV is 0, and the MV is stored in D200. When X0.0 is switched OFF, the instruction is not executed, and the previous data is unchanged.



Additional remark:

1. The instruction can be used several times, but the registers specified by **I_MV**~**I_MV**+9 can not be the same.
2. **I_MV** occupies 20 word registers. **I_MV** used in the instruction DPIDE in the example above occupies D126~D145.
3. The instruction DPIDE can only be used in the cyclic task and the time interrupt. When the instruction DPIDE is used in the time interrupt, the sampling time is the same as the interval between the time interrupts.

4. When the instruction DPIDE is scanned, the PID algorithm is implmented according to the sampling time, and the MV is directly refreshed. Whether the scan time reaches the sampling time is not calculated automatically. When the instruction is used in the time interrupt, the sampling time is the same as the interval between the time interrupts. The PID algorithm is implemented according to the interval between the time interrupts.
5. Before the PID algorithm is implemented, the present value used in the instruction DPIDE has to be a stable value. When users need the input value in the module to implement the PID algorithm, they have to notice the time it takes for the analog input to be converted into the digital input.
6. If the PV is in the range indicated by ERR_DBW, the CPU module will bring the error into the PID algorithm until the PV reaches the SV. The cross status condition will not be met until the PV crosses the zero cross point indicated by the SV. If the cross status condition is met, the error will be count as 0 until the PV is out of the range indicated by ERR_DBW. If PID_DE is set to true, the variations in the PV will be used to calculate the control value of the derivative, and the CPU module will count the Delta PV as 0 after the cross status condition is met. (Delta PV=Current PV-Previous PV)
   In the PV trend chart shown below, the CPU module implements the PID algorithm normally in the A sections A. In the B sections, the CPU module counts the error or the Delta PV as 0 when it implements the PID algorithm.



PID algorithms:

1. When **PID_MODE** is set to 0, the PID control mode is the automatic control mode.
   - **Independent formula & derivative of of the E** (**PID_EQ**=False & **PID_DE**=False)

$$CV = K_p E + K_i \int_0^t Edt + K_d \frac{dE}{dt} + BIAS$$

$$E = SV - PV \quad or \quad E = PV - SV$$

- **Independent formula & derivative of the PV** (**PID_EQ**=False & **PID_DE**=Ture)

$$CV = K_p E + K_i \int_0^t Edt - K_d \frac{dPV}{dt} + BIAS$$

$$E = SV - PV$$

$$or$$

$$CV = K_p E + K_i \int_0^t Edt + K_d \frac{dPV}{dt} + BIAS$$

$$E = PV - SV$$

- **Dependent formula & derivative of the E** (**PID_EQ**=True & **PID_DE**=False)

$$CV = K_c \left[E + \frac{1}{T_i} \int_0^t Edt + T_d \frac{dE}{dt} \right] + BIAS$$

$$E = SV - PV \quad or \quad E = PV - SV$$

- **Dependent formula & derivative of the PV** (**PID_EQ**=True & **PID_DE**=True)

$$CV = K_c \left[E + \frac{1}{T_i} \int_0^t Edt - T_d \frac{dPV}{dt} \right] + BIAS$$

$$E = SV - PV$$

$$or$$

$$CV = K_c \left[E + \frac{1}{T_i} \int_0^t Edt + T_d \frac{dPV}{dt} \right] + BIAS$$

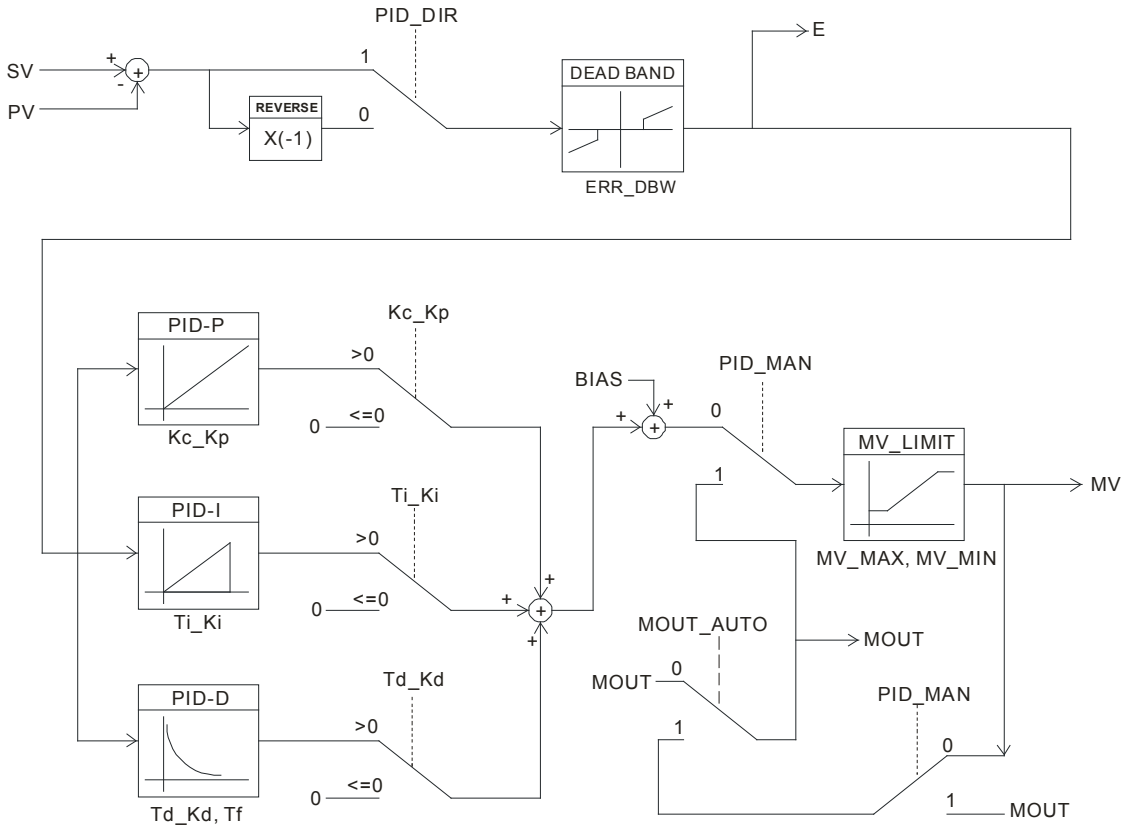
$$E = PV - SV$$

※The CV values in the formulas above are the MV used in DPIDE.

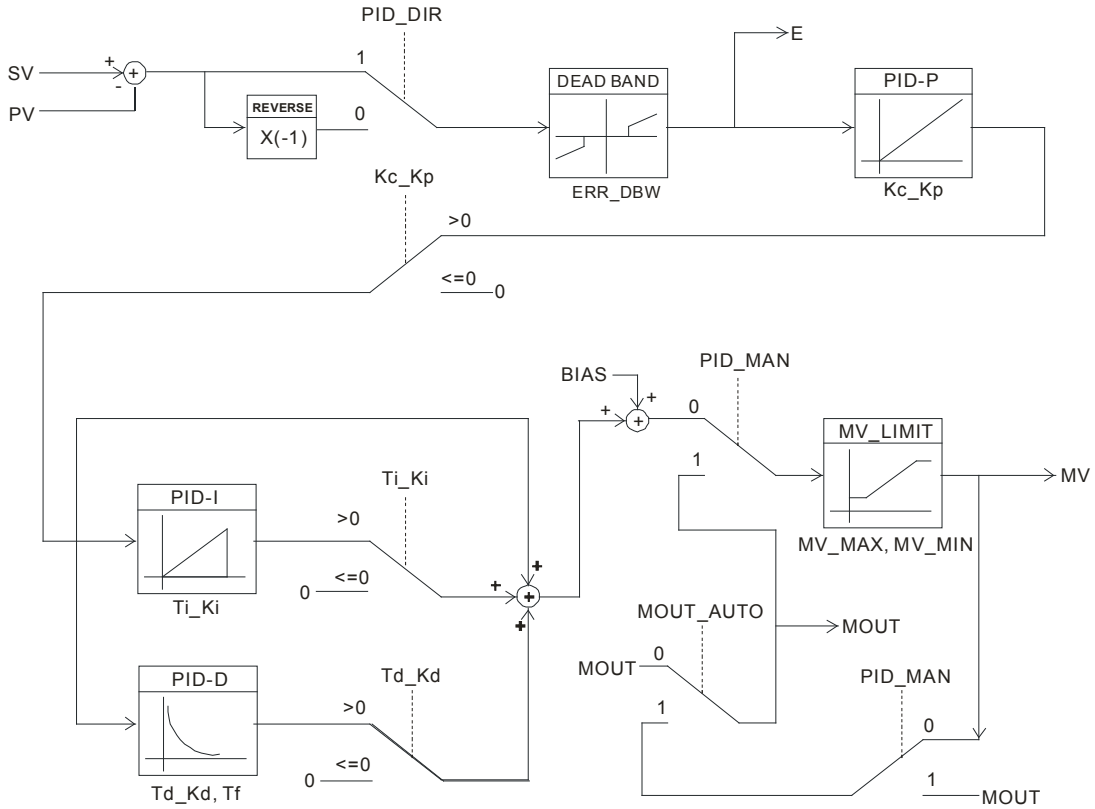
2. When **PID_MODE** is set to 1, the PID control mode is the automatic tuning mode. After the tuning of the parameter is complete, **PID_MODE** is set to 0. The PID control mode becomes the automatic control mode.

PID control diagrams:

PID Block Diagram (Independent)



PID Block Diagram (Dependent)



Suggestion:

a) Owing to the fact that the instruction DPIDE can be used in a lot of controlled environments, users have to select control functions appropriately. For example, the MV switches between the maximum output value and the minimum output value when **PID_MODE** is set to 1. Please do not use DPIDE in the environment controlled by a motor which reacts rapidly, otherwise the violent change of the system resulting from the automatic tuning of the parameters may hurt the staff or damage the system.

b) When users tune the parameters Kc_Kp, Ti_Ki, and Td_Kd (**PID_MODE** is set to 0), they have to tune Kc_Kp first (according to their experiences), and then set Ti_Ki and Td_Kd to 0. When the users can handle the control, they can increase Ti_Ki and Td_Kd. When Kc_Kp is 1, it means that the proportional gain is 100%. That is, the error is increased by a factor of one. When the proportional gain is less than 100%, the error is decreased. When the proportional gain is greater than 100%, the error is increased.

c) To prevent the parameters which have been tuned automatically from disappearing after a power cut, it is suggested that users should store the parameters in latching data registers if **PID_MODE** is set to 1. The parameters which have been tuned automatically are not necessarily suitable for every controlled environment. Therefore, the users can modify the parameters which have been tuned automatically. However, it is suggested that users only modify Ti_Ki or Td_Kd.

d) The action of the instruction depends on many parameters. To prevent improper control from occurring, please do not set parameters randomly.

4. Manually Tuning the Parameters in DPID/DPIDE

Assume that the transfer function of the controlled device G(S) in a control system is the first-order function $G(s)=\frac{b}{s+a}$ (the function for the model of a general motor), the target value (SV) is 1, and the sampling time (TS) is 10 milliseconds. It is suggested that users should follow the steps below.

Step 1: First, set the K_i and the K_d to 0. Next, set the K_p to 5, 10, 20 and 40 successively, and record the target values and the present values. The results are shown in Figure 1.

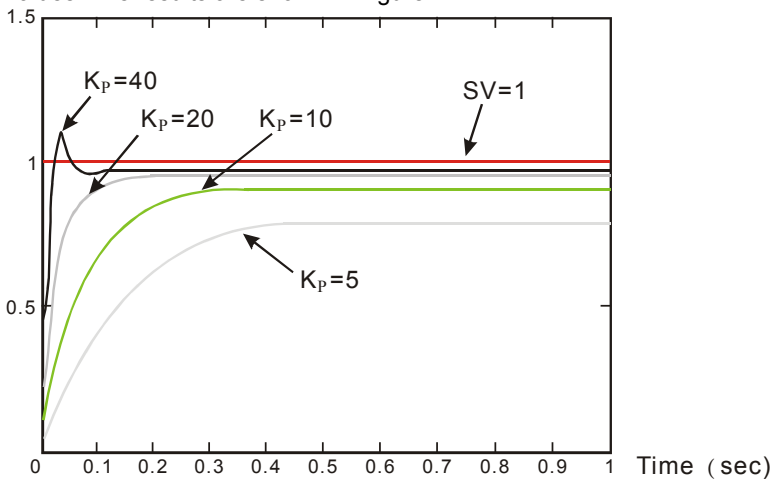


Figure 1

Step 2: When the K_p is 40, there is overreaction. Thus, the K_p is not chosen. When the K_p is 20, the reaction curve of the PV is close to the SV, and there is no overreaction. However, due to the fast start-up, the transient output value (MV) is big. The K_p is not chosen, either. When the K_p is 10, the reaction curve of the PV approaches the SV smoothly. Therefore, the K_p is chosen. When the K_p is 5, the reaction is slow. Thus, the K_p is not chosen.

Step 3: After the K_p is set to 10, increase the K_i. For example, the K_i is set to 1, 2, 4, and 8 successively. The K_i should not be greater than the K_p. Then, increase the K_d. For example, the K_d is set to 0.01, 0.05, 0.1, and 0.2 successively. If the parameter Tf is used, it can be set to $T_d/8$. The K_d should not be greater than ten percent of the K_p. Finally, the relation between the PV and the SV is presented in Figure 2

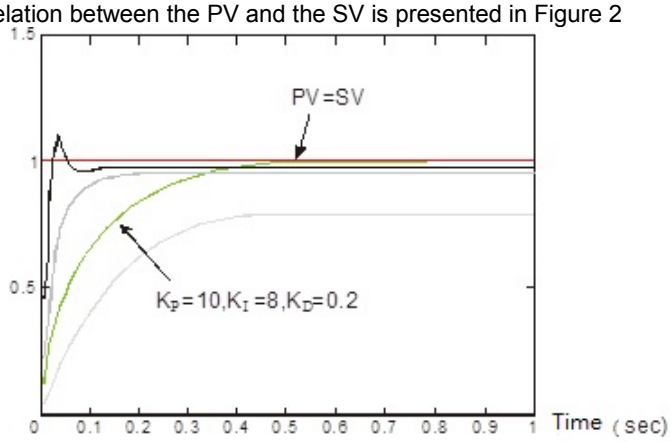


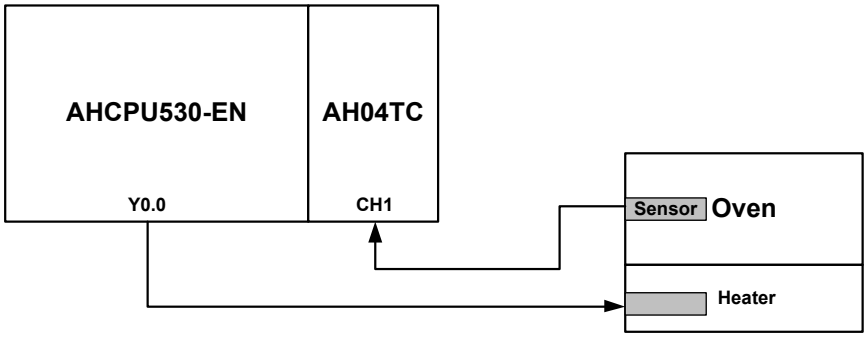
Figure 2

Note: The example is only for reference. Users have to tune the parameters properly according to the practical condition of the control system.

5. Examples

5.1 Example 1: Using a CPU Module to Realize DPID Control (Using a CPU Module to Control a Small Oven)

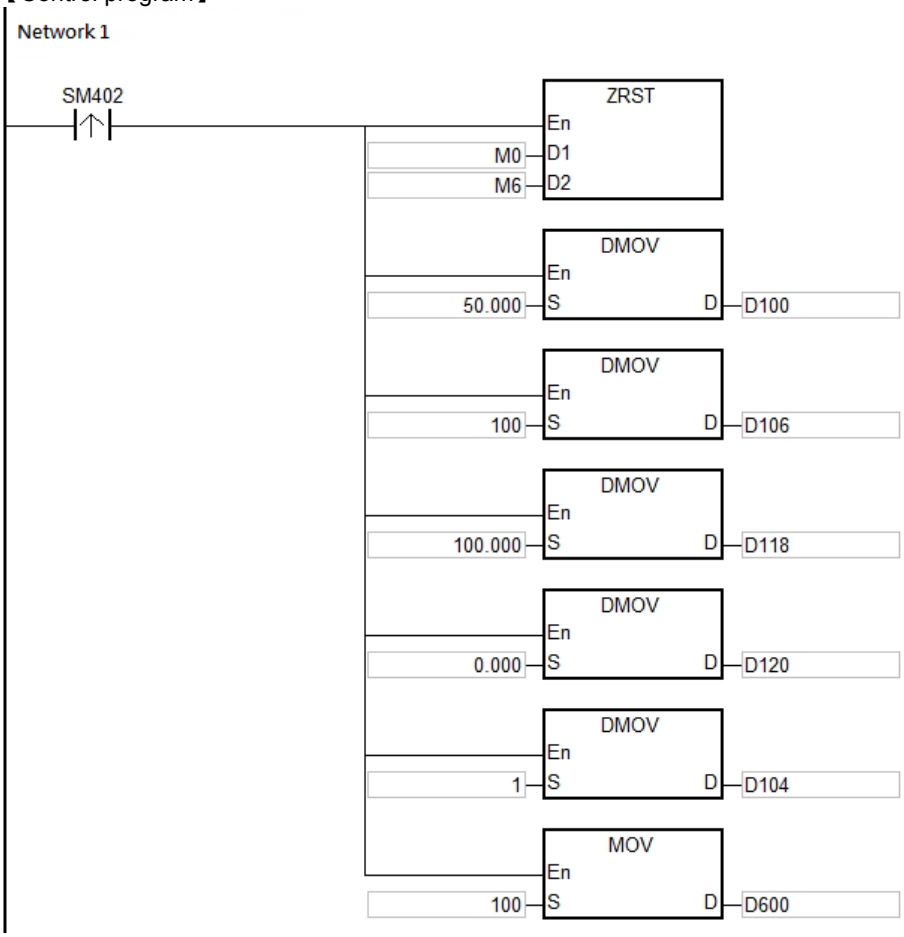
【System structure】

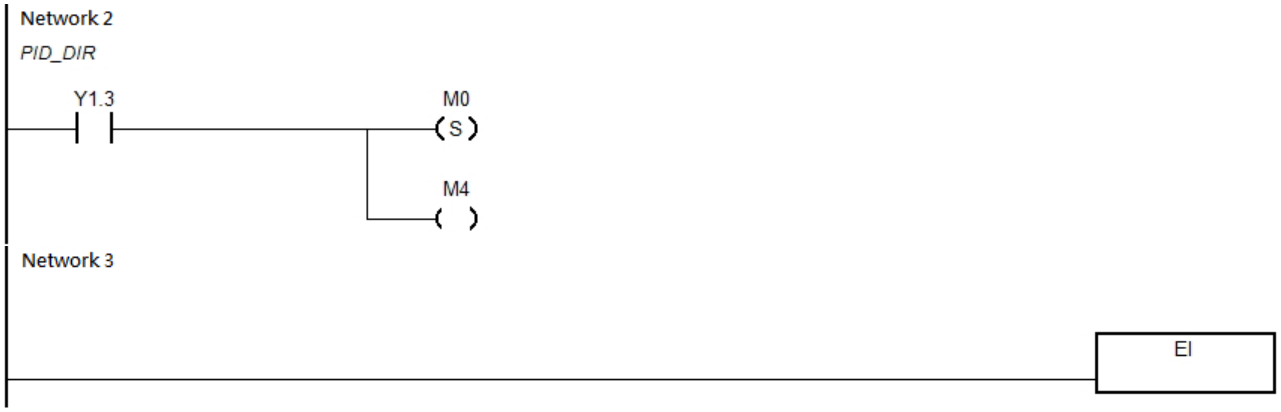


【Control requirement】

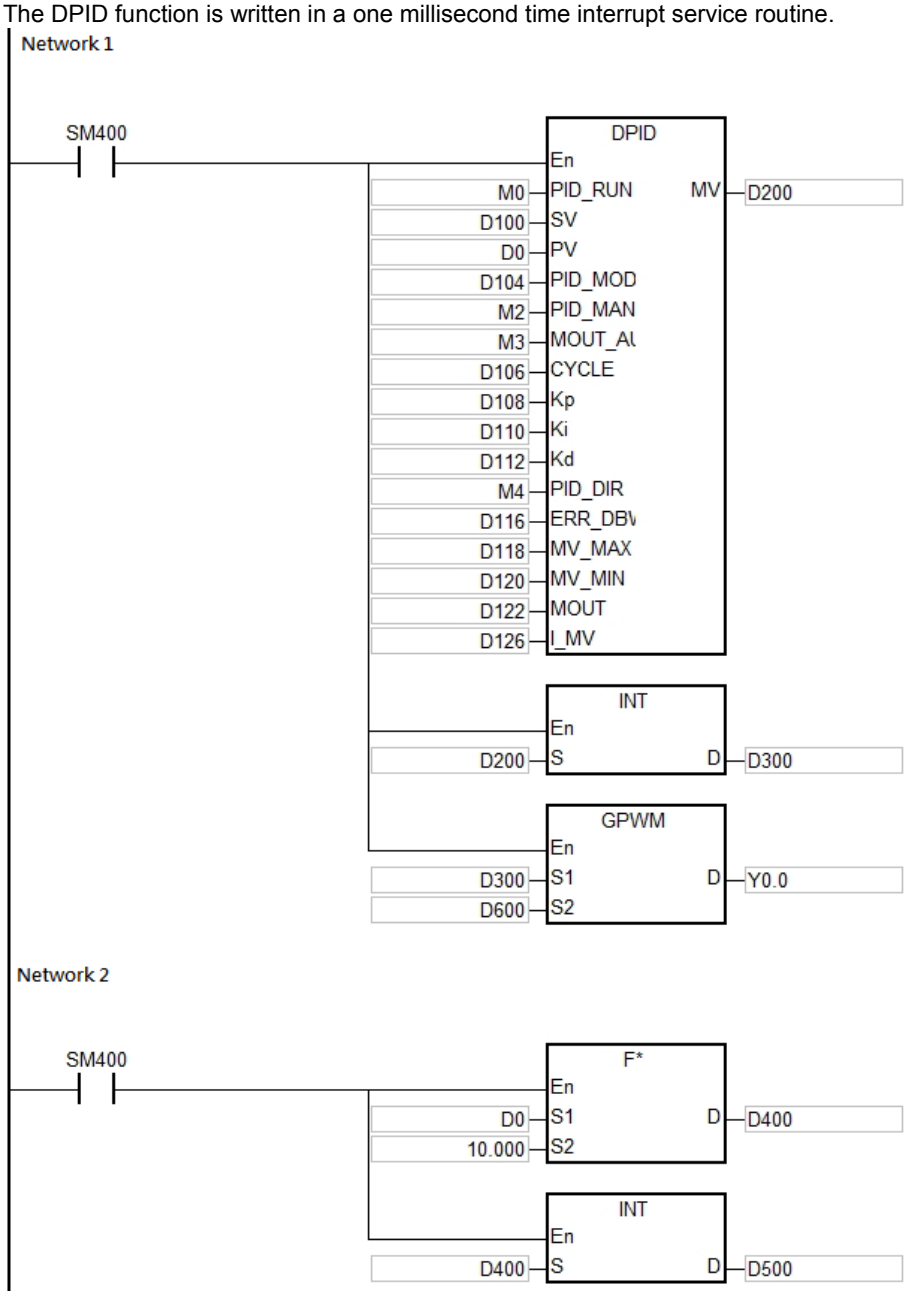
The control environment in this example is a small oven. AHCPU530-EN, AH04TC-5A and a type K thermocouple are used to control the oven. First, DPID parameters are adjusted automatically for temperature control (the value in D104~D105 is 0). After the adjustment of the parameters is complete, the value in D104~D105 will be changed to 1 automatically, and the values of the parameters (Kp, Ki, and Kd) calculated will be used to realize the DPID control of the oven.

【Control program】





The DPID function is written in a one millisecond time interrupt service routine.

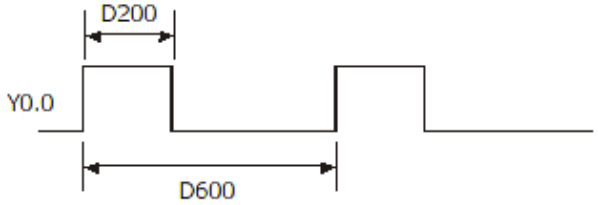


【Device description】

Device	Description
M0	Enabling the control mode
Y0.0	Output device of GPWM
D200~D201	Result of the PID algorithm
D100~D101	Target temperature
D600	Cycle of GPWM
D106~D107	Sampling time of PID
D104~D105	Control mode

【Program description】

- To enable the type K thermocouple connected to AH04TC-5A, users need to set parameters by HWCONFIG in ISPSoft.
- M0 and M1 are enabled. The temperature measurement module AH04TC-5A measures the temperature of the small oven, and sends the temperature to the CPU module. The CPU module adjusts parameters automatically for temperature control (D104~D105=0), and calculates the best values of the PID temperature control parameters. After the adjustment of the parameters is complete, the value in D104~D105 is automatically changed to 1. The parameters automatically calculated (the K_P in D108~D109, the K_I in D110~D111, and the K_D in D112~D113) are used to realize the DPID control of the small oven.
- The function of automatically adjusting parameters for temperature control is used for the PID algorithm. The output result of the PID algorithm (in D200~D201) is used as the input of the instruction GPWM. After GPWM is executed, Y0.0 outputs variable width pulses to control the heater, and the PID control of the small oven is realized.



The experimental result of the initial adjustment of the values of the parameters is shown in Figure 3.

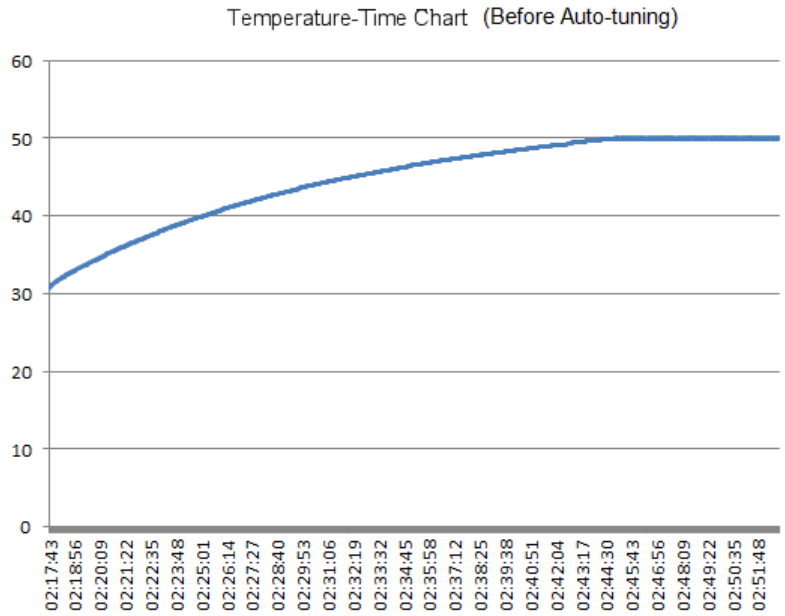


Figure 3

The experimental result of using the values of the parameters after the adjustment for temperature control is shown in Figure 4.

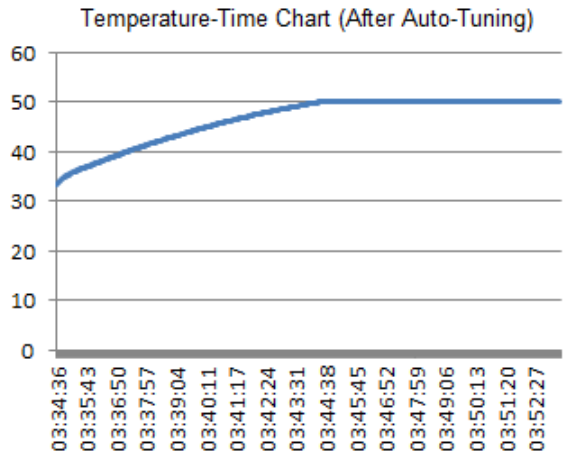
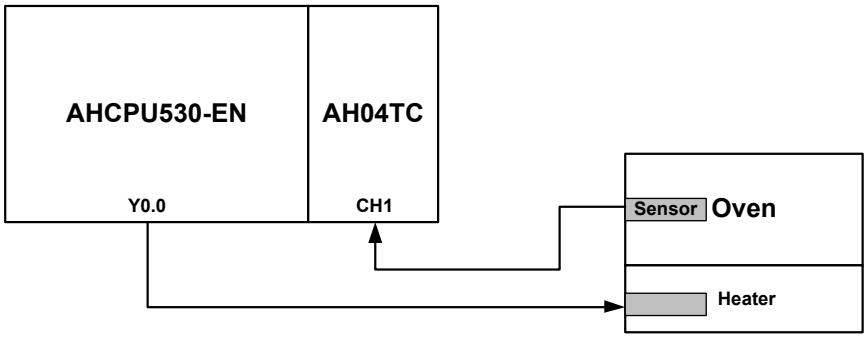


Figure 4

5.2 Example 2: Using a PLC to Realize DPIDE Control (Using a CPU Module to Control a Small Oven)

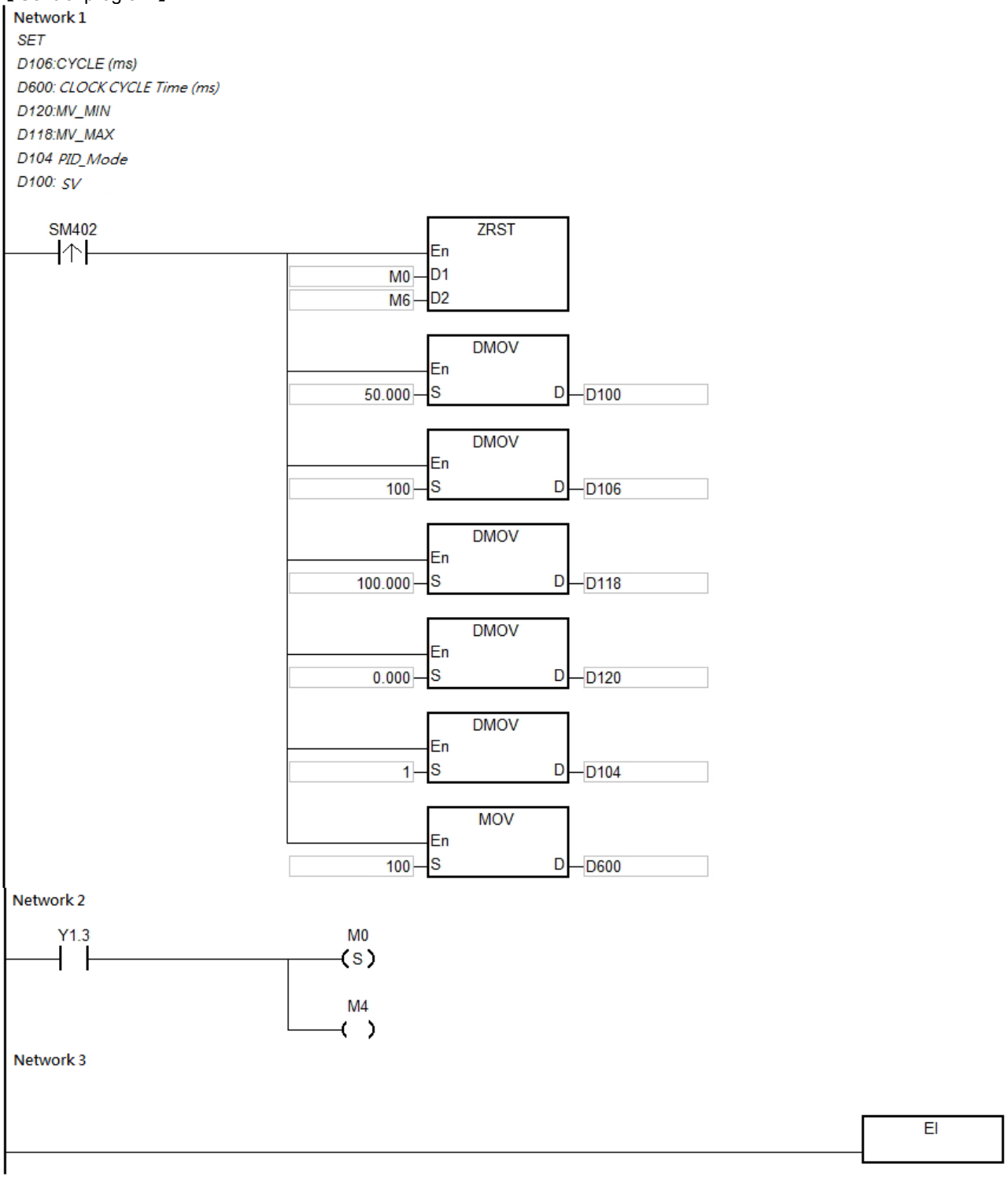
【System structure】



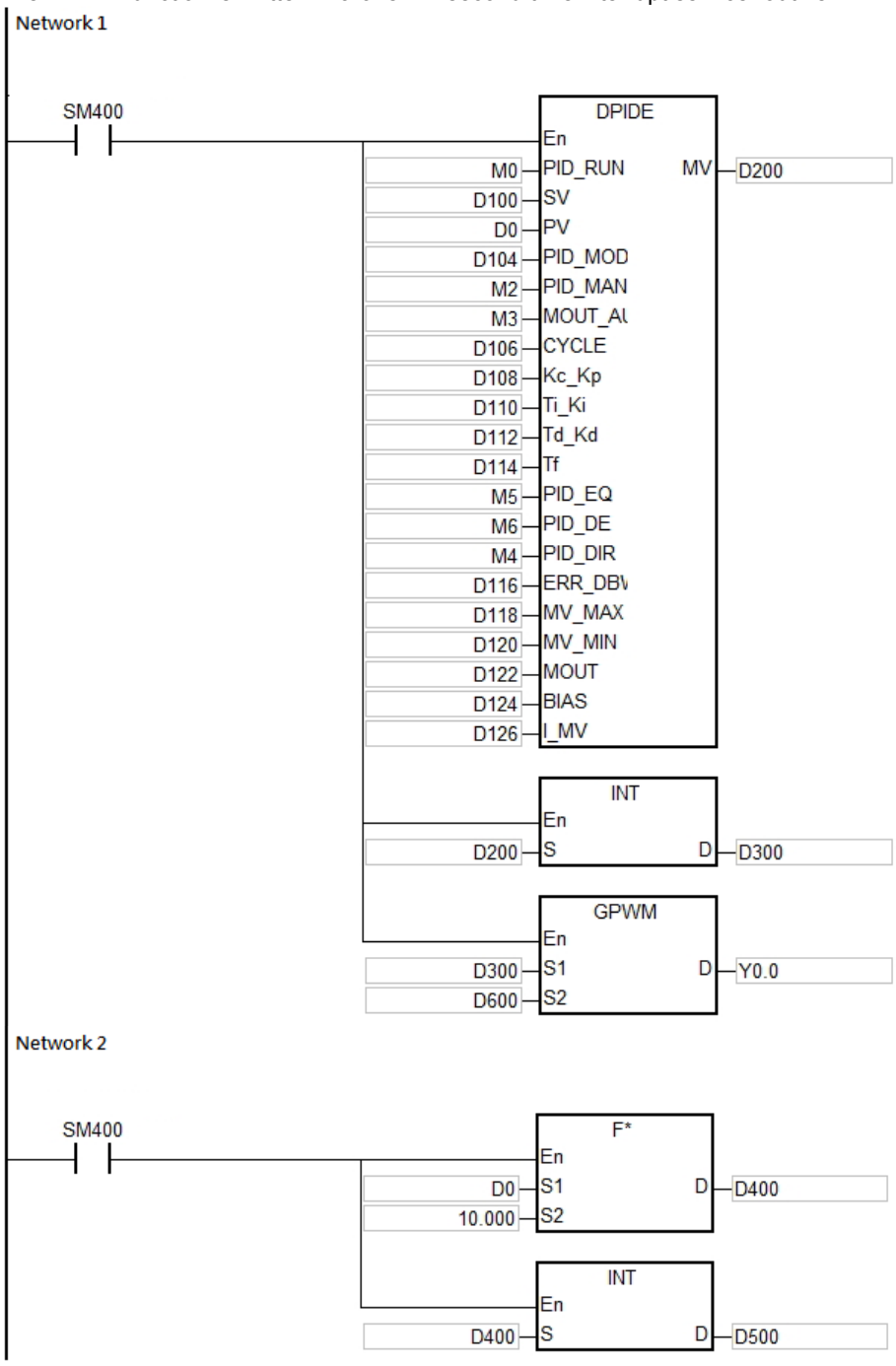
【Control requirement】

The control environment in this example is a small oven. AHCPU530-EN, AH04TC-5A and a type K thermocouple are used to control the oven. First, DPIDE parameters are adjusted automatically for temperature control (the value in D104~D105 is 0). After the adjustment of the parameters is complete, the value in D104~D105 will be changed to 1 automatically, and the values of the parameters (Kp, Ki, Kd, and Tf) calculated will be used to realize the DPIDE control of the oven.

【Control program】



The DPIDE function is written in a one millisecond time interrupt service routine.



【Device description】

Device	Description
M0	Enabling the control mode
Y0.0	Output device of GPWM
D200~D201	Result of the PID algorithm
D100~D101	Target temperature
D600	Cycle of GPWM
D106~D107	Sampling time of PID
D104~D105	Control mode

【Program description】

- To enable the type K thermocouple connected to AH04TC-5A, users need to set parameters by HWCONFIG in ISPSoft.
- After the PID algorithm is implemented, the output result of the PID algorithm (in D200~D201) is used as the input of the instruction GPWM. After GPWM is executed, Y0.0 outputs variable width pulses to control the heater, and the DPIDE control of the small oven is realized.
- When the temperature measured is constant and is 50℃, the best values of the DPIDE temperature control parameters (the K_P in D108~D109, the K_I in D110~D111, the K_D in D112~D113, and the T_f in D114~D115) are calculated automatically. The experimental result of the initial adjustment of the values of the parameters is shown in Figure 5.
- Use ISPSoft to read the values of the DPIDE temperature control parameters (the K_P, K_I, K_D, and T_f) in AH04TC-5A, and write the values to the devices corresponding to the DPIDE temperature control parameters. Set the parameter PID_MODE to 1, that is, write 1 to D104~D105.
- M0 is enabled again by means of the K_P, K_I, K_D, and T_f calculated. The experimental result of using the values of the parameters after the adjustment for temperature control is shown in Figure 6.

The experimental result of the initial adjustment of the values of the parameters is shown in Figure 5.

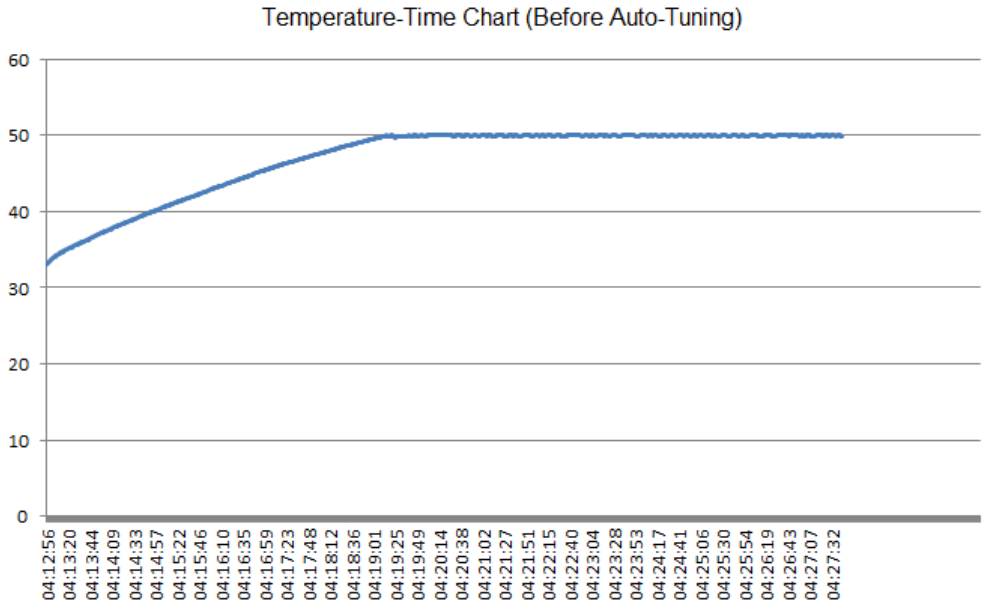


Figure 5

The experimental result of using the values of the parameters after the adjustment for temperature control is shown in Figure 6.

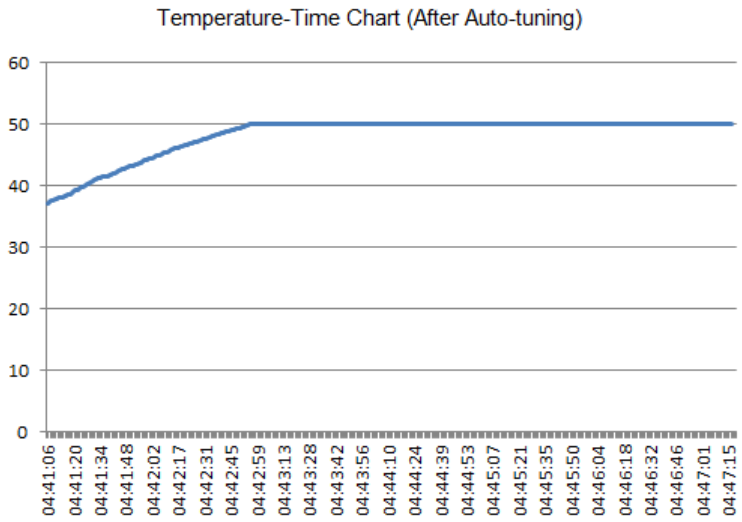
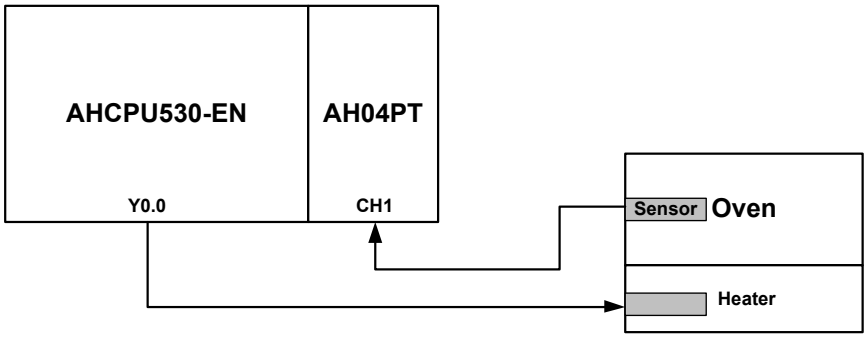


Figure 6

5.3 Example 3: Using a CPU Module to Realize DPID Control (Using a CPU Module to Control a Large Oven)

【System structure】



【Control requirement】
The control environment in this example is a large oven. AHCPU530-EN, AH04PT-5A and a Pt100 sensor are used to control the oven. First, DPID parameters are adjusted automatically for temperature control (the value in D104~D105 is 0). After the adjustment of the parameters is complete, the value in D104~D105 will be changed to 1 automatically, and the values of the parameters (Kp, Ki, and Kd) calculated will be used to realize the DPID control of the oven.

【Control program】
Please refer to the control program in section 5.1 for more information.

【Device description】
Please refer to the device description in section 5.1 for more information.

【Program description】
- To enable the Pt100 sensor connected to AH04PT-5A, users need to set parameters by HWCONFIG in ISPSoft.
- Please refer to the program description in section 5.1 for more information.

The experimental result of the initial adjustment of the values of the parameters is shown in Figure 7.

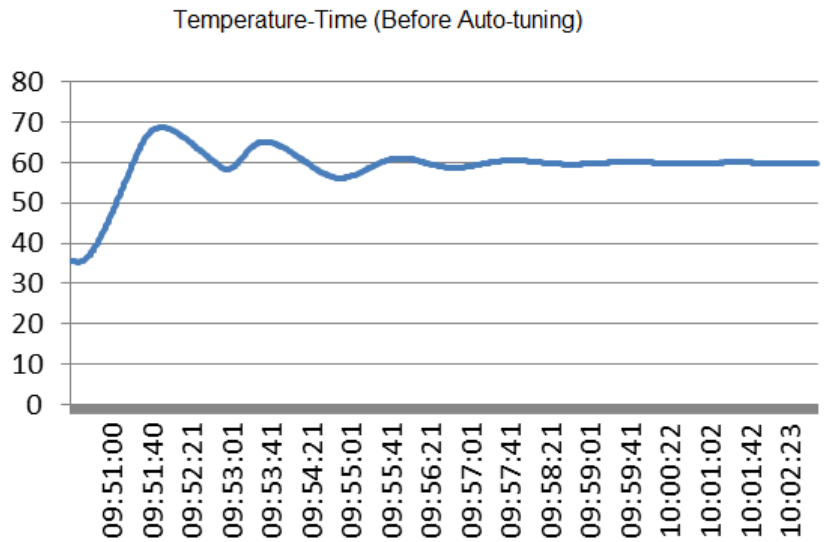


Figure 7

The experimental result of using the values of the parameters after the adjustment for temperature control is shown in Figure 8.

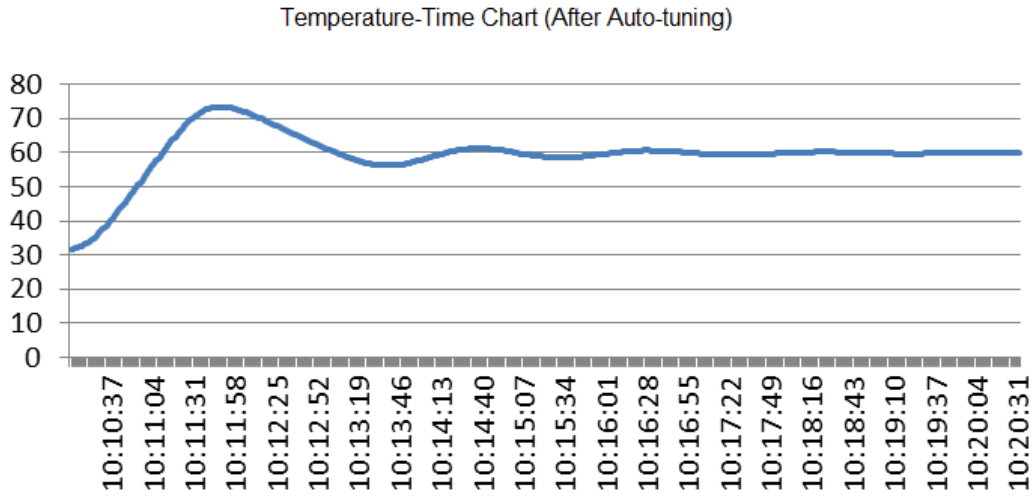
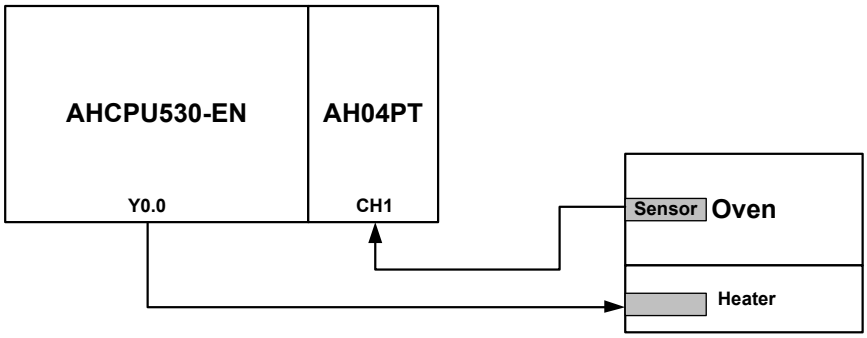


Figure 8

5.4 Example 4: Using a CPU Module to Realize DPIDE Control (Using a CPU Module to Control a Large Oven)

【System structure】



【Control requirement】
The control environment in this example is a large oven. AHCPU530-EN, AH04PT-5A and a Pt100 sensor are used to control the oven. First, DPIDE parameters are adjusted automatically for temperature control (the value in D104~D105 is 0). After the adjustment of the parameters is complete, the value in D104~D105 will be changed to 1 automatically, and the values of the parameters (Kp, Ki, Kd, and Tf) calculated will be used to realize the DPIDE control of the oven.

【Control program】
Please refer to the control program in section 5.2 for more information.

【Device description】
Please refer to the device description in section 5.2 for more information.

【Program description】
- To enable the Pt100 sensor connected to AH04PT-5A, users need to set parameters by HWCONFIG in ISPSoft.
- Please refer to the program description in section 5.2 for more information.

The experimental result of the initial adjustment of the values of the parameters is shown in Figure 9.

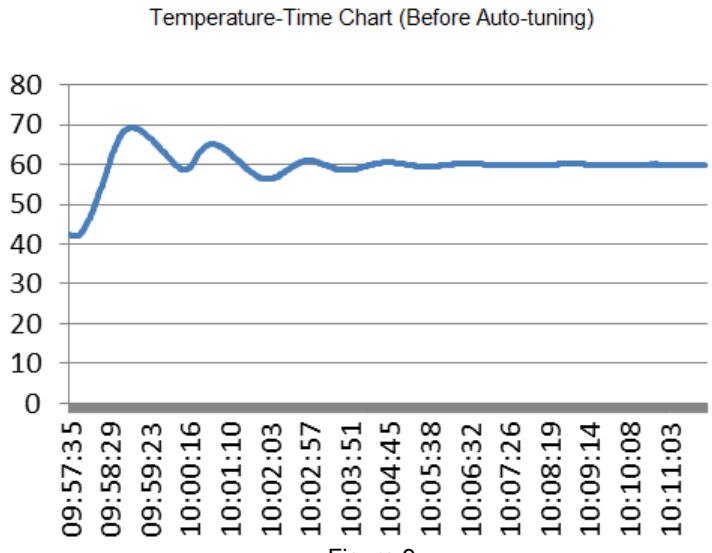


Figure 9

The experimental result of using the values of the parameters after the adjustment for temperature control is shown in Figure 10.

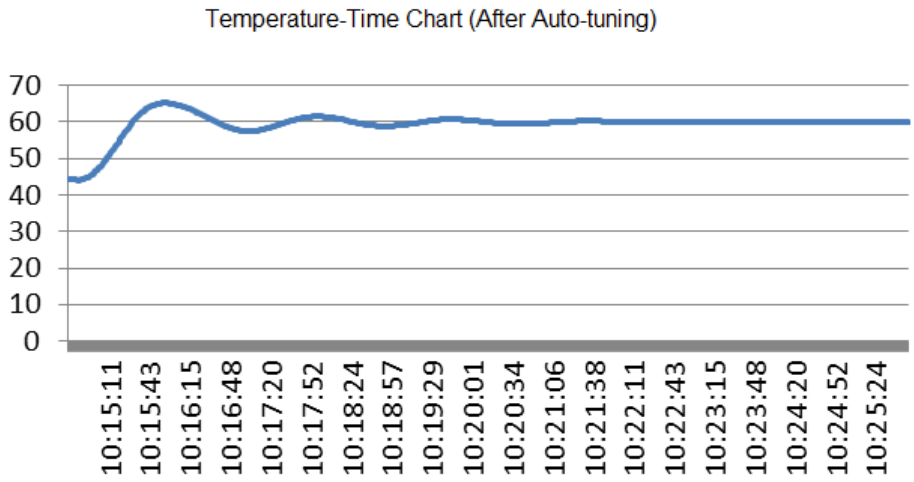
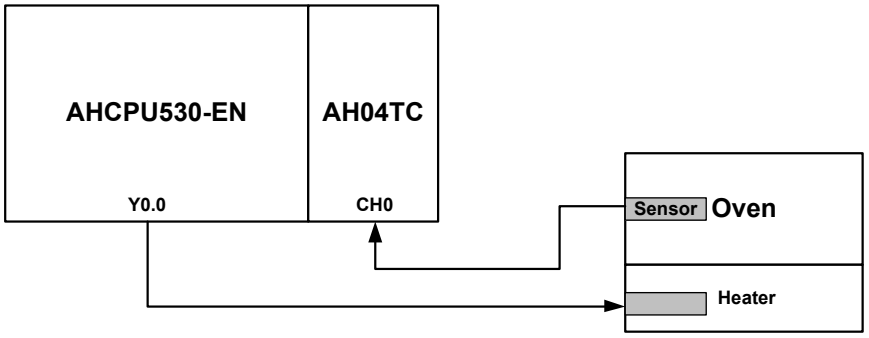


Figure 10

5.5 Example 5: Using AH04TC-5A to Realize DPID Control

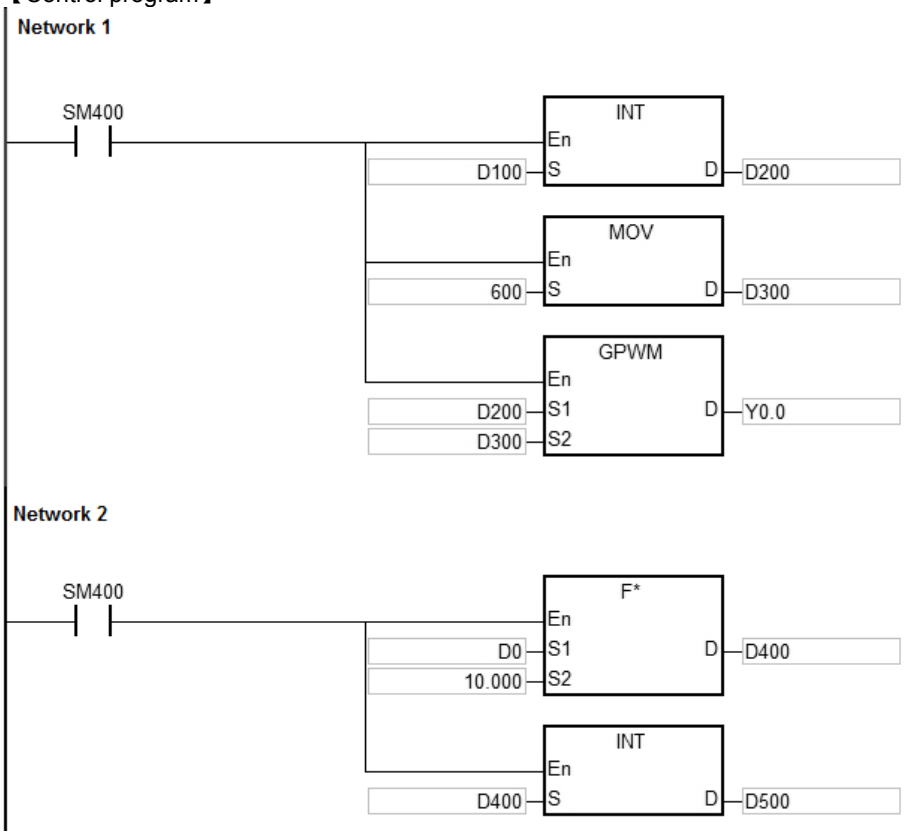
【System structure】



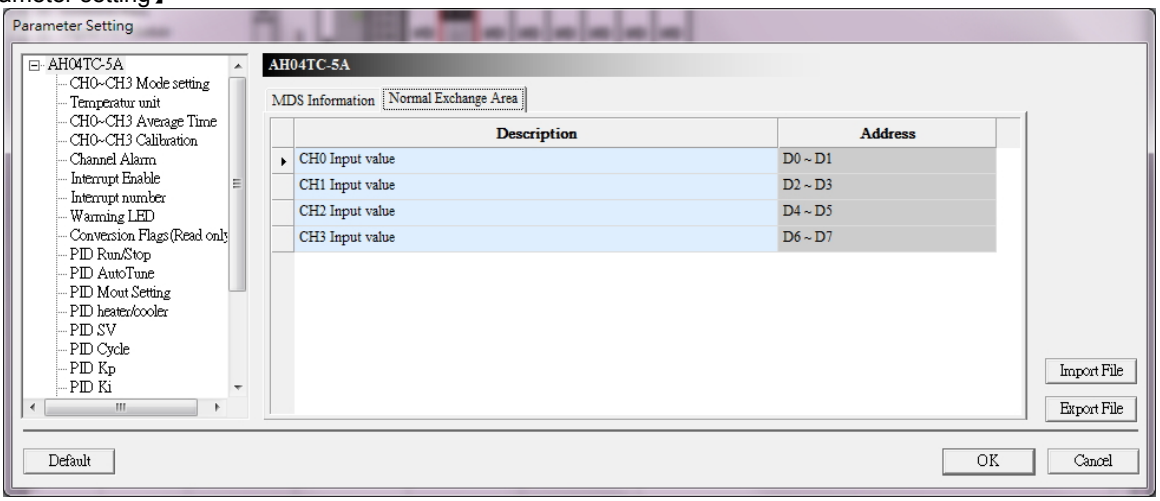
【Control requirement】

The control environment in this example is a small oven. AHCPU530-EN and AH04TC-5A are used to control the oven. First, parameters are adjusted automatically for temperature control. After the adjustment of the parameters is complete, the values of the parameters (Kp, Ki, and Kd) calculated will be used to realize the DPID control of the oven.

【Control program】



【Parameter setting】



1. Click **CH0~CH3 Mode setting** in the **Parameter Setting** window for AH04TC-5A in HWCONFIG, and select **K-Type** from the **Setting Value** drop-down list cell for **CH0 Input mode setting**.
2. Click **PID Run/Stop**, and select the **Run** checkbox in the **Setting Value** cell for **CH0 PID Run/Stop**.
3. Click **PID Auto Tune**, and select the **Enable** checkbox in the **Setting Value** cell for **CH0 PID Auto Tune**.
4. Click **PID SV**, and type “50.0” in the **Setting Value** cell for **CH0 SV**.
5. Click **PID MV_MAX**, and type “2000.0” in the **Setting Value** cell for **CH0 MV_MAX**.
6. Leave the remaining default values unchanged.
7. Click **PID Kp**, and set D110~D111 in the **Address** cell for **CH0 Kp**. Click PID Ki, and set D112~D113 in the **Address** cell for **CH0 Ki**. Click **PID Kd**, and set D114~D115 in the **Address** cell for **CH0 Kd**.

8. Click **PID MV (Read only)**, and set D100~D101 in the **Address** cell for **CH0 MV (Read only)**.

【Device description】

Device	Description
D0~D1	Present value received by channel 0
D100~D101	Output value of PID (MV)
D110~D111	Proportional gain (K_p)
D112~D113	Integral gain (K_i)
D114~D115	Derivative gain (K_d)
D300	Cycle of GPWM

【Usage】

- Users need to set parameters by HWCONFIG in ISPSoft.
- After the PID algorithm is implemented, the output result of the PID algorithm (in D100~D101) is used as the input of the instruction GPWM. After GPWM is executed, Y0.0 outputs variable width pulses to control the heater, and the DPIDE control of the small oven is realized.
- When the temperature measured is constant and is 50℃, the best values of the DPIDE temperature control parameters (the K_P in D110~D111, the K_I in D112~D113, and the K_D in D114~D115) are calculated automatically. The experimental result of the initial adjustment of the values of the parameters is shown in Figure 11.
- Use ISPSoft to read the values of the DPID temperature control parameters (the K_P, K_I, and K_D) in AH04TC-5A, and write the values to the devices corresponding to the DPID temperature control parameters. Unselect the **Enable** checkbox in the **Setting Value** cell for **CH0 PID Auto Tune**.
- The experimental result of using the values of the parameters calculated (the K_P, K_I, and K_D) after the adjustment for temperature control is shown in Figure 12.

The experimental result of the initial automatic adjustment of the values of the parameters is shown in Figure 11.

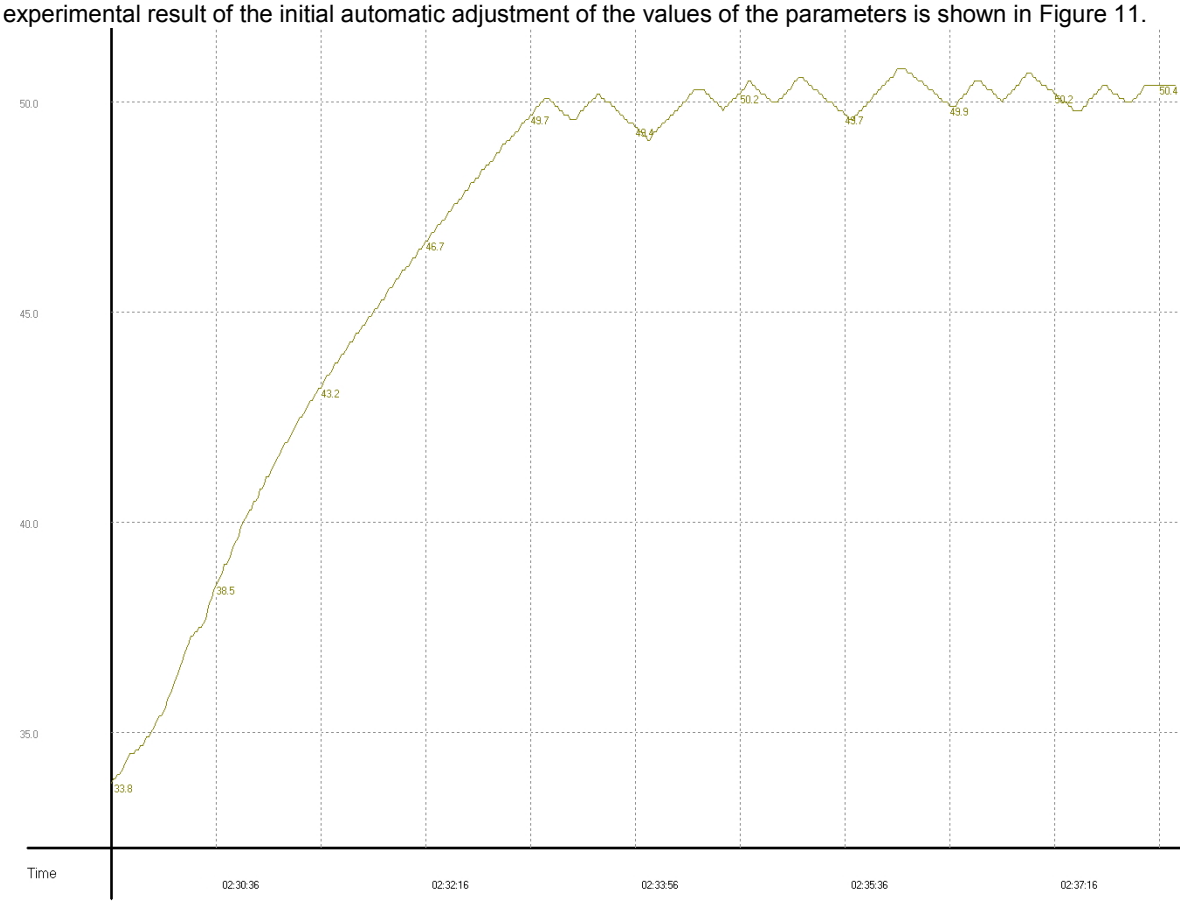


Figure 11

The experimental result of using the values of the parameters (the K_P, K_I, and K_D) after the adjustment for temperature control is shown in Figure 12.

Channel	Starting temperature	Closing temperature	Kp	Ki	Kd
CH0	33.8℃	50.0℃	102.836	6.170	2.57

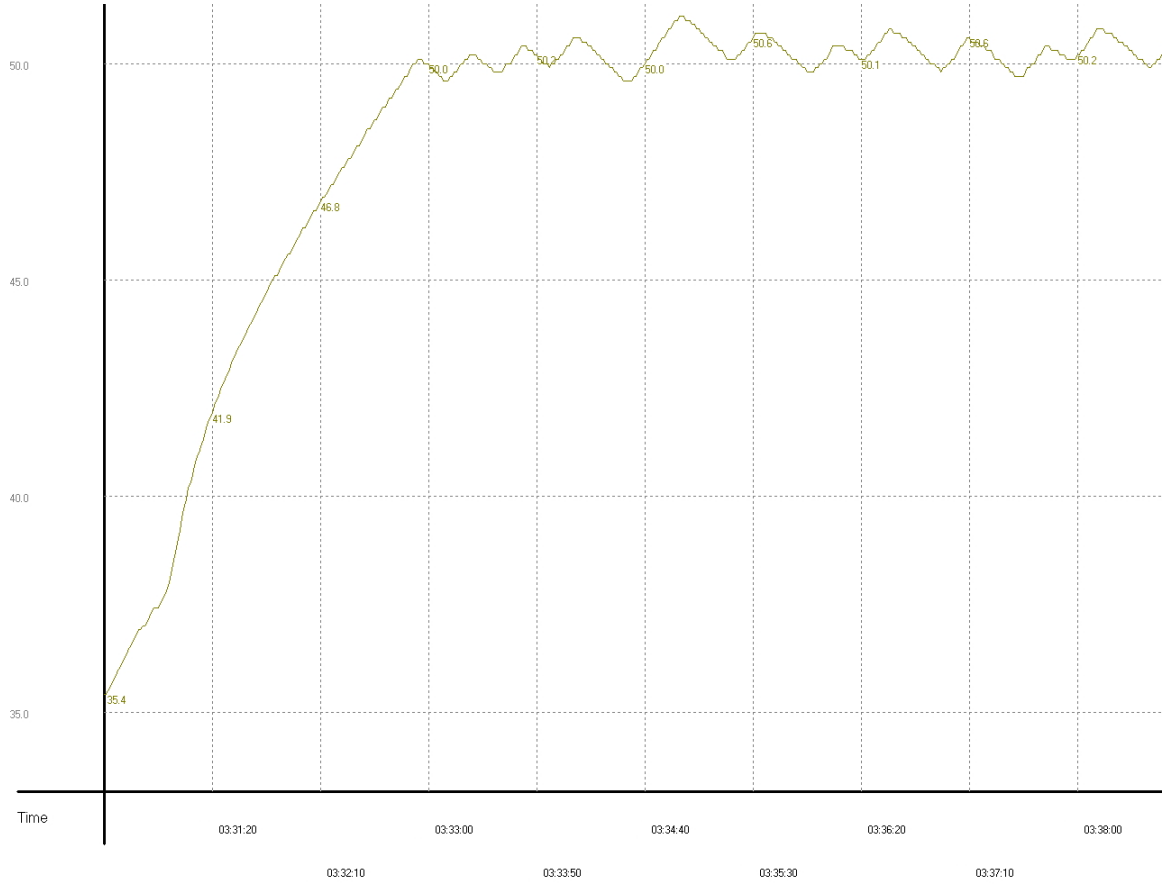


Figure 12