CEGELEC
INDUSTRIAL CONTROLS

PIDSIM
Function for Closed Loop Control

DESCRIPTION

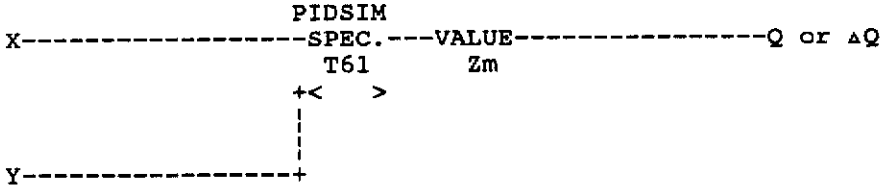
The PIDSIM special function is intended for the control of closed loop systems. It incorporates greatly improved facilities and provides better control than the existing Special Functions PIDABS and PIDINC.

FEATURES

- Independently programmable; proportional, integral and derivative gains.
- Integral or derivative gain settings, and rate limit settings, are independent of the execution interval.
- Derivative terms can be made to operate on the error or on the measured value.
- Independent positive and negative output limits.
- Mode selection for greater flexibility.
- Control input to suicide the output (without any rate limit action).
- Choice of INCREMENTAL or ABSOLUTE outputs.
- Deadband facility enables small errors to be treated as zero.

SPECIFICATION

Control Language:



Older GEM80 programming units will not automatically recognize the mnemonic PIDSIM, but it can be added as a user defined mnemonic to any System Programmer.

Quantity of table locations: 20

Basic equation; Absolute output:

$$Q_n = \frac{P}{100}E_n + \frac{D}{100t}(X_n - X_{n-1}) + \frac{It}{100}E_n + Q_I + Offset$$

Basic equation; Incremental output:

$$\Delta Q_n = Q_n - Q_{n-1}$$

i.e.

$$\Delta Q_n = \frac{P}{100}E_n + \frac{D}{100t}(X_n - X_{n-1}) + \frac{It}{100}E_n + Q_I - Q_{n-1}$$

Where:

Q_n = output on scan n
Q_{n-1} = output on scan n-1
Q_I = integral component of output on scan n-1
E = error (setpoint - measured value)
X = either error or measured value
P = proportional gain
D = differential gain
I = integral gain
t = time since last execution (secs)

Table 1 Program Rung Data

Sym	Location	Use	Range
X	Rung input	Upper branch; used for Setpoint (Desired Value)	±32767
Y	Rung input	Lower branch; used for Measured Values	±32767
Q	Rung output	Control output	±32767 Subject to upper and lower limits and rate limits if in absolute output mode
Zm	Value location	Start of parameter list	Any write-enable table usable by special functions

Table 2 Value Data

Sym	Location	Use	Range
F	Zm	Fault code	See Table 3
Mode *	Zm + 1	Bit selection of modes	See application notes
P *	Zm + 2	Proportional gain	0-32767. Corresponds to gains of 0 to 327.67 in steps of 0.01
I *	Zm + 3	Integral gain	0-32767. Corresponds to gains of 0 to 327.67/secs in steps of 0.01/secs
D *	Zm + 4	Derivative gain	0-32767. corresponds to gains of 0 to 327.67/secs in steps of 0.01/secs
DBAND *	Zm + 5	Deadband value. If error lies within ± of this value, it is set to zero	0-32767. See application notes, Section 4
LOLIM *	Zm + 6	Lower output limit	-32768 to +32767
HILIM *	Zm + 7	Upper output limit	-32768 to +32767
L_1 *	Zm + 8	Rate limit away from zero. Only applies when in absolute output mode	0-32767. corresponds to rates of 0 to 327.67 bits/ms in store of 0.01 bits/ms
L_2 *	Zm + 9	Rate limit towards zero. Only applies when in absolute output mode	0-32767. Corresponds to rates of 0 to 327.67 bits/ms in steps of 0.01 bits/ms
OFFSET *	Zm + 10	Offset on output	-32768 to +32767
I_2 *	Zm + 11	Post integral gain	0-10000. Corresponds to gains of 0 to 1.0000 in steps of 0.001
T	Zm + 12	Time when PIDSIM last executed	0-65535 in msec units. Displayed as -32768 to +32767 when monitoring
E_{n-1}	Zm + 13	Error on last scan (scan n-1)	-32768 to +32767
Q_{n-1}	Zm + 14	Output on last scan (scan n-1)	-32768 to +32767 for incremental O/P. Subject to O/P limits for absolute O/P.
MV_{n-1}	Zm + 15	MV on last scan (scan n-1)	-32768 to +32767

Table 2 **Value Data (Continued)**

Sym	Location	Use	Range
R_1	Zm +16 Zm +17	Integration remainder (2 words). Least significant word. Most significant word.	±2147483647 Set to zero if Zm + 3 = 0. Set to zero if Zm + 3 = 02
Q_1	Zm + 18 Zm + 19	Integral component of output on last scan (scan n-1) (2 words). Least significant word Most significant word	±2147483647. Subject to integral limit Set to zero if Zm + 3 = 0 Set to zero if Zm + 3 + 0

Table 3 **Fault Conditions and Flags**

Bit Set	Value	Condition	Result
0	@1	Rate limit L_1 exceeded	(t in ms)
1	@2	Rate limit L_2 exceeded	$Q_n = Q_{n+1} + \frac{L_2 t}{100}$ (t in ms)
2	@4	Integral component of O/P limited at HILIM.	$Q_n = Q_p + Q_D$ + HILIM
3	@8	Integral component OF O/P limited at LOLIM	$Q_n = Q_p + Q_D$ + LOLIM
4	@10	O/P limited by LOLIM	Q_n = LOLIM
5	@20	O/P limited by HILIM	Q_n = HILIM
6	@40	Post integral gain $1_2 < = 0$	$I_2 = 0$
7	@80	Post integral gain $I_2 < =$ 10000	I_2 = 10000
9	@200	General parameter error	$Q_n = Q_{n-1}$
10	@400	Incremental O/P overflow	Indication only

APPLICATION NOTES

PRINCIPLES OF CLOSED LOOP CONTROL

Given that the PIDSIM Special Function is going to be used, we naturally assume that the user has some knowledge of Closed Loop Control theory. If this is not the case, we strongly recommend that the user purchases the manual 'Closed Loop Control using GEM80 Programmable Controllers' or else refers to a standard textbook on the subject.

The following practical points should not be ignored:

Accuracy

Steady state accuracy of a closed loop is the sum of the accuracies of the following:

A Transducer converting the measured variables into an electrical signal.

Any signal conditioning equipment scaling and/or isolating this signal.

Analog to digital conversion of the measured value.

Any scale factor or linearization calculations on the measured value.

Note: Inaccuracies in the control signal to the plant, do not enter into the system accuracy calculations. However, the resolution of the control signal (the smallest change that can be made) can be important. This topic is dealt with later.

Response

Depends on:

Plant characteristics.

The response of the measuring transducer.

The placement of the measuring transducer.

Analog and digital filtering.

For instance, if a temperature transducer is placed a long distance away from the heat source, any change of temperature due to heat input may not be sensed by the transducer for several seconds, and a fast control loop becomes impossible.

INCREMENTAL/ABSOLUTE OUTPUT MODES

The ABSOLUTE OUTPUT mode must be used where the control signal is roughly proportional to the setpoint under steady state conditions.

The INCREMENTAL OUTPUT mode must be used where the control signal to the plant always has the same value under steady state conditions, regardless of the setpoint value. This will be referred to as the 'home value' of the control signal. With this type of system the control signal moves away from its home value when the setpoint is changed, but settles back to its home value when steady conditions have been achieved.

Example of a flow control loop

Consider the closed loop control of fluid flow but with valves having different types of actuators.

Example 1

A valve actuator input in the range of 4-20mA, where:

4mA = valve fully closed
20mA = valve fully open

For a larger flow, a larger control signal is needed, i.e. the control signal must increase as the flow setpoint is increased. Therefore, for this type of valve actuator, PIDSIM must be used in ABSOLUTE OUTPUT mode.

Example 2

A valve actuator input in the range of 4-20mA, where:

4mA = valve closing at maximum rate
20mA = valve opening at maximum rate
12mA = valve stationary (stays put)

Under steady state conditions the valve must remain stationary regardless of the flow required, but when the flow changes valve movement is called for. Once the change is complete the valve must stay put at its new opening. For this reason, the control signal must always return to its 'home value' (12mA) regardless of the flow setpoint.

For most applications of INCREMENTAL OUTPUT mode the 'home' output value will be zero, e.g. where the plant signal swings between −10V and +10V. The example given above makes it clear that this is not always so and is not a requirement for PIDSIM. INCREMENTAL OUTPUT mode is most often used for driving motor-powered actuators without built-in positioning servos, which require zero input for any steady state position. It may also be used where the plant itself provides integral action.

SIGNAL LEVELS

Normally, the control signal and the measured value will be analog signals, however, within a GEM80 Controller both signals will be treated as numbers.

When programming a GEM80 Controller, a user may be tempted to convert the signals into meaningful numbers for monitoring: e.g. converting a thermocouple signal into a number equal to the actual temperature in °C or °F. Such conversions may reduce control loop resolution and cause a 'dither' type of fluctuation.

For example, if a temperature in the range of 0° to 100°C; is converted into a number in the range of 0 to 100; the system will only be able to set the temperature in increments of 1°C. If instead the temperature is converted into a number in the range of 0 to 10,000, then it can be controlled in increments of 0.01°C.

Similarly, programming a valve actuator with a three term control function; which provides a control signal with an output number range of 0 to 100; will only allow the valve to be set to discrete positions 1 per cent apart.

It may well be that to maintain the measured value equal to the setpoint, the valve should be at a position halfway between two steps, in which case, the Controller will cause the valve to dither up and down by 1 per cent as it tries to obtain the correct mean setting.

Therefore, the numbers used for the setpoint, measured value and control signal, should be sufficient to give the resolution required.

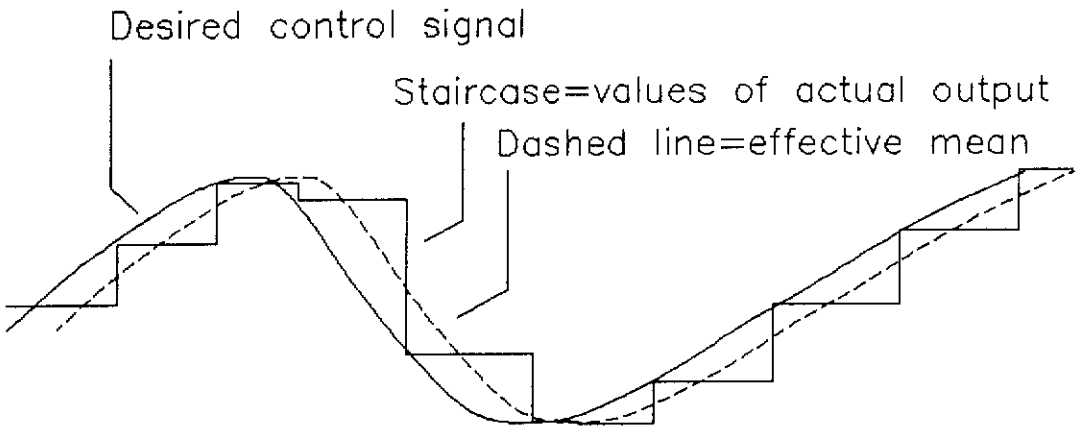
Where 'real values' need to be provided for monitoring purposes, simple multitudes should be used (i.e. 10 or 100 times). If this is not feasible, separate rungs for monitoring should be included in the program.

PROGRAM REPETITION INTERVAL

To achieve the fastest response, the GEM80 program could be arranged so that PIDSIM updates the value of the plant control signal on every cycle through the program.

However, updating the control signal more frequently than necessary is undesirable, as PIDSIM involves several calculations inside it and consequently takes longer to perform than most other functions.

Because of this, we recommend that the sections of program containing PIDSIM be included within 'BLOCKs' initiated at regular intervals, where the time interval is such as to make only a marginal difference to the closed loop response. The effect on the control signal being updated at the repetition interval, is shown below.



In effect, the control signal is delayed by half the repetition interval.

For further details on execution times, refer to the Technical Manual of your GEM80 Controller.

Time dependant BLOCK example

In the program example below, coil G50.0 energizes after a delay of 2 seconds. However, on the next program execution, the G50.0 contact in the same rung resets the DELAY function and de-energizes the coil; the operation then repeats.
Consequently, the normally-open contact G50.0, in the next rung, causes the BLOCK containing PIDSIM to be operated at 2 second intervals.

As the BLOCK repetition interval depends on the VALUE set for the DELAY function, the BLOCKs initiation is unaffected if there is any subsequent alteration of the configuration data for the program cycle time.

Program rungs

```
|                                                          |
+--]/[---DELAY---VALUE--------------------------( )--+
| G50.0   G51      20                             G50.0|
|                                                       |
+--] [-----------------------------------------OBEY--+
  G50.0                                          BLOCK
                                                  *
**************************************************
|                                                       |
| SETPT  PIDSIM                                CTRLSIG|
+-<AND>---SPEC.---VALUE-------------------------<OUT>+
|  G52     T61     W100                          D32  |
|       +-<   >                                       |
| M.V.  |                                             |
+-<AND>-+                                             |
|  G53                                                |
|                                                       |
+ ***************** END OF BLOCK ******************* +
```

Programs containing several 3-term controls

With a system containing several three term controls, in order not to slow down the program cycle time too much, different control loop BLOCKs should be initiated on different cycles through the program. This can easily be achieved by using the SEQR (sequencer) special function. If all the control loops are to operate at the same repetition interval this is straightforward, but if control loops are required to operate at different intervals, we recommend the use of a timing diagram.

Example:

There are 3 loops, two can be operated at 4-second repetition intervals, the remaining loop must be operated at a 2-second interval.

One way of doing this would be to arrange the operation of the loops as follows:-

Time (seconds)	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1st loop	X				X				X				X		
2nd loop			X				X				X				X
3rd loop		X		X		X		X		X		X		X	

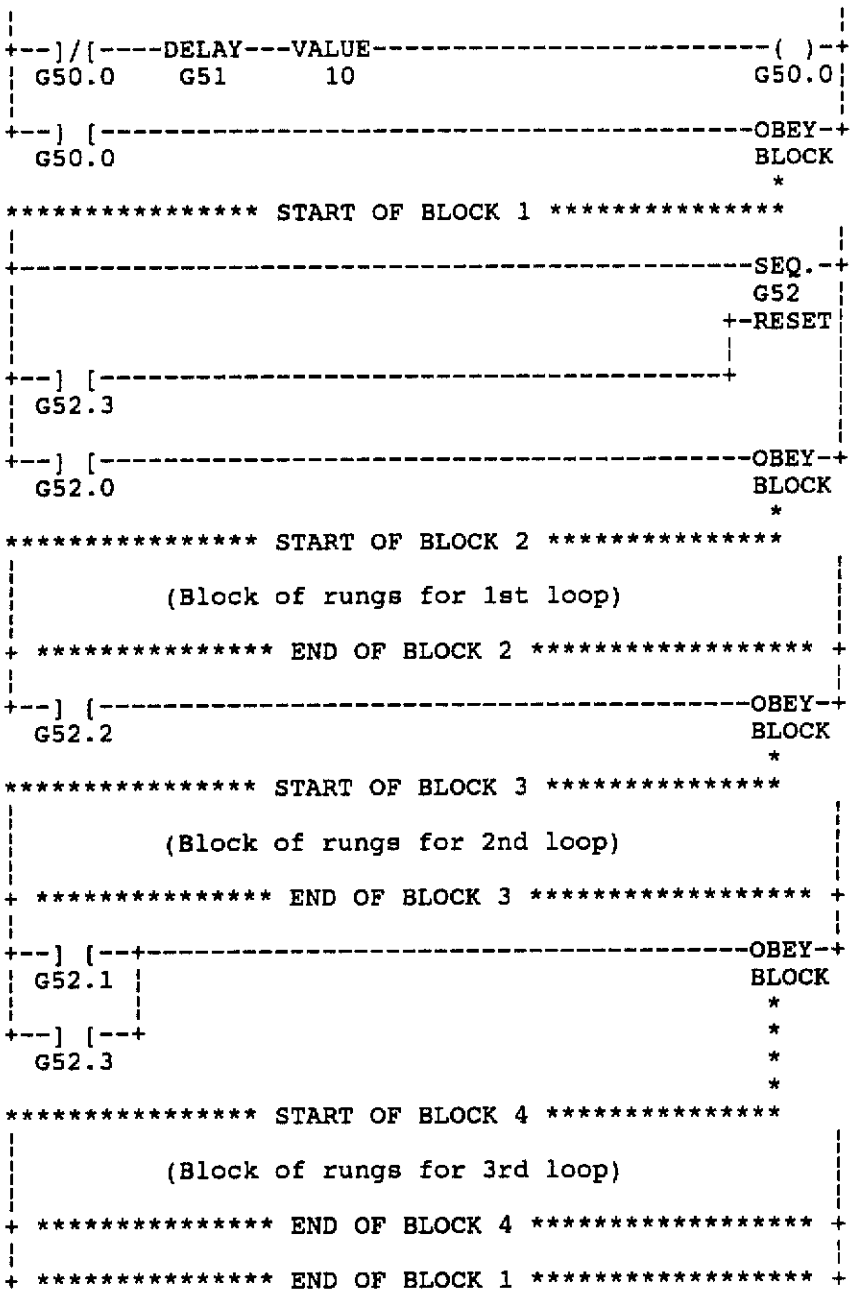
The timing diagram has been arranged so that a control loop operates every second and the cycle of operation repeats every 4 seconds. Consequently, a rung with a 1-second delay is needed to operate into a sequencer with 4 steps, after which, the sequencer must be reset so that the cycle of operations can be repeated. A suitable ladder diagram is shown opposite. This is not the only method of

programming a set of loops but it is probably the most orderly.

Note: It is essential that each loop operates at a regular interval. The alternative timing diagram shown below does not observe this rule and is therefore incorrect.

Time (seconds)	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1st loop	X				X				X			X			
2nd loop		X				X				X				X	
3rd loop			X	X			X	X			X	X			

Program rungs



In the second timing diagram, instead of operating at regular 2-second intervals the 3rd loop operates at intervals of 1 and 3 seconds, this will give a somewhat unpredictable performance.

There are other methods of initiating loops at regular intervals, for example, using timing markers E0.0 or E0.1. However, problems may be encountered (depending on the program cycle time) when trying to count timing markers. The DELAY method described above works regardless of the program cycle time.

Note: The same techniques for time dependant blocks can be used for other types of function besides three term control. When a timing diagram timing is constructed, a user may find it useful to include other functions operated at regular intervals, to ensure that no one program scan takes longer to complete than any other program scan.

DETERMINATION OF CONTROLLER SETTINGS

When PIDSIM is incorporated into a program, values must be entered to set the proportional gain, integral time and derivative time. These values can be determined by one of the following four methods:

Measurement

Running the plant open loop and measuring its response to a small step change of input. From this response find the settings from the standard formulae.

Calculation

Using the plants design information to calculate its transfer functions and then using standard analytical techniques, such as Bode or Nyquist diagrams, to find the required Controller settings.

Knowledge

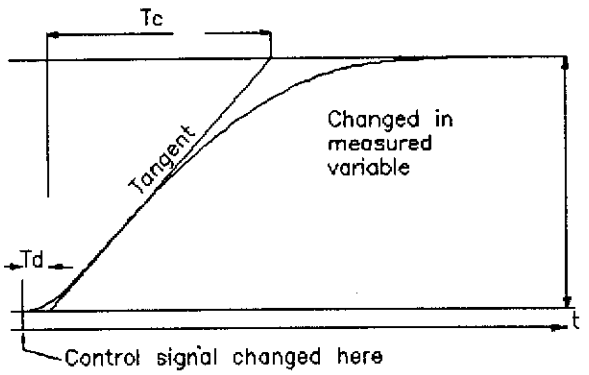
Using similar settings to those of an existing plant for initial trials.

Cut-and-Try

Starting from a low gain and winding the gains up until the response starts to become osillatory.

Measurement Method

The response of the measured value to a small step change of control signal can be plotted or recorded as a graph, like the example shown opposite.



Note: When conducting such a test, a control signal should be used which is large enough to make the plant operate in the middle of its range prior to applying the step change. For example, with a valve actuator, the valve should already be open and not starting from its closed position when you apply the step change of control signal. This will avoid the non-linearities often found in plant characteristics at the extremes of the control range.

From this graph three values should be measured as follows:

1. The gain of the plant:

$$A = \frac{\text{change in measured value}}{\text{change in control signal}}$$

2. The Effective deadtime:

   This is obtained by drawing a tangent (see above), to a point where it cuts the initial value of the measured value.

3. The Effective time constant:

This is obtained by extending the tangent to the point where it cuts the final value of the measured value and finding the time difference between the two tangent ends.

T_c = Effective time constant

The three values are used in determining the 3-term controller settings and repetition interval.

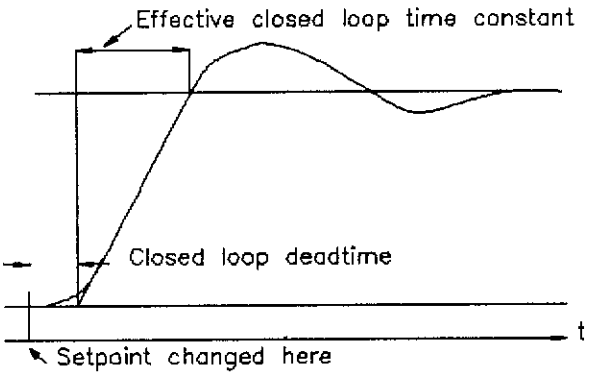
Note: If a plant uses the INCREMENTAL OUTPUT mode, its output continues to change until the control signal is set to its home value, it will not therefore produce a response like the one shown. To conduct a test, change the control signal for a small time (e.g. 1 second) and then bring it back to its home value. By denoting the length of time that the control signal was applied for as T_1 the formula for the gain of the plant then becomes:

$$A = \frac{\text{change in measured value}}{\text{change in control signal}} \times \frac{1}{T_1}$$

If the measured deadtime T_d is close to the test time T_1 the test should be redone with a larger change of control signal for a shorter test time.

Closed Loop Response

If the plant is controlled by a closed loop with a three term control function and a very short update time, then the fastest response that can be obtained for a small step change in setpoint is shown in Figure 3.



The following approximations apply:-

The deadtime of the closed loop response is about the same as that of the open loop response.

The effective time constant of the closed loop response is about the same as the deadtime.

No improvement in the speed of response can be obtained by further tuning of the 3 term control function. Instead, the response becomes more unstable or oscillatory.

Proportional, Integral and Derivative Gains

By measuring the values of plant gain A, plant deadtime T_d and plant time constant T_c, settings can be determined from the following rules of thumb:

Repetition interval

$$T = \frac{T_d}{2}$$

In many cases, even with PIDSIM placed in a BLOCK executed at intervals of several seconds, T_d is so long that this condition is easily met. However, if T_d is fairly short and T is of the same order as $T_d/2$, then the sampling delay should be allowed for by treating the effective plant deadtime as:

Effective plant deadtime

$$T_d' = T_d + (\frac{T}{2})$$

In addition, there will always be a delay of one program execution between the time when the Controller reads the setpoint and measured value inputs, and the time when it sets the control signal output to the plant. For most process control applications this time will be negligible, but for very fast loops, the execution time T_e should be added on, thereby making the effective plant deadtime into:

Effective plant deadtime

$$T_d' = T_d + (\frac{T}{2}) + T_e$$

Accordingly, for a plant with a 10-second time delay; operated by PIDSIM within a BLOCK operated at 1 second intervals and a program cycle time of 20 ms, the effective deadtime would be 10.52 seconds.

Note...The extra 0.02 seconds is negligible.

Rules of thumb:-

For 2-term PI control

Proportional gain $= \frac{1}{A} \times \frac{T_c}{T_d'} \times 0.9$

Integral time $= \frac{T_d'}{0.3}$

For 3-term PID control:

Proportional gain $= \frac{1}{A} \times \frac{T_c}{T_d'} \times 1.2$

Integral time $T_1 = \frac{T_d'}{0.5}$

Derivative time $T_D = T_d' \times 0.5$

(Note: proportional band % $= \frac{100}{\text{proportional gain}}$)

Calculation Method

If enough is known about a plants design to calculate its transfer functions, it may be possible to analyse the closed loop system using Bode or Nyquist methods.

When compared with a pure analog system, the only extra factor that must be considered is the time delay effect of the program repetition interval. Using a digital programmable Controller would give an additional time delay (transport lag) of T/2 + T_s which would not be present in an analog system.

Where a closed loop system with a bandwidth of 1 radian/second is called for (and if T_s is negligible), choose T = $^1/_2$ to give the extra 14° phase shift required at the bandwidth frequency. Shortening the repetition interval reduces the phase shift proportionally. Alternatively, if PIDSIM is executed on every program execution so that T is equal to T_s then T_s should be made equal to $^1/_6$ to get the same 14°phase shift.

From the analysis, it should be possible to determine the values for proportional time, integral time (and derivative time if needed) and the program and BLOCK repetition intervals, to get the type of response required.

Knowledge Method

Where a GEM80 Controller operates a similar plant, the values of its settings can be directly copied.

Where a similar plant comes under analog control, or where an analog control loop is being replaced with a GEM80 system, the same settings can be used for the proportional gain, integral gain and derivative times. However, a suitable BLOCK repetition interval still needs to be established.

Rules of thumb:

For 2-term PI control

$$T = 0.15T_I$$

where T_I is the integral time

For 3-term PID control

$$T = 0.25T_I$$

where T_I is the integral time

Note...These formulae assume that the integral action has been wound up, to give the quickest reset action to a setpoint change which gives a satisfactory response. If this is not so, the repetition interval may have to be made somewhat shorter than the implied intervals given by the rules of thumb.

Cut and Try Method

This is the least methodical method. It is not recommended for users who lack experience in setting up closed loop controls, instead, we recommend that they use the measurement method.

An experienced user may attempt to short cut open-loop testing of the plant and enter first-try table values directly. How these table values relate to proportional gain, integral time and derivative time, is described in a following section.

Response not fast enough

Sometimes, the response from a closed loop system controlled by PIDSIM is not fast enough for the application.

The closed loop response depends on the system deadtime. Normally this is determined by the plant, but in some cases, could be caused by the GEM80 program cycle time. The only way to speed up the response is to reduce the deadtime. Any improvement in response time will be proportional to the reduction in deadtime.

Where the plant deadtime is predominant, the deadtime can only be reduced by modifying the plant itself. Measuring transducers may have to be physically moved so that they respond to changes earlier. In other cases, the system may have to be redesigned so that a single loop is replaced by two loops in cascade.

TABLE VALUES

The values that must be set up for PIDSIM are related to the values obtained for proportional gain, integral time and derivative time.

Table value P = proportional gain in units of 0.01 (e.g. the value required is 5000 for a proportional gain of 50).

Table value I = P/T_I (e.g. the value required is 50 for an integral time of 100 seconds with P = 5000).

Table value D = P/T_D (e.g. 10000 for a derivative time of 2 seconds with P = 5000).

WORKED EXAMPLES

Measurement Method

Temperature control: An output from a GEM80 Controller feeds a valve in a steam heating pipe. The measured value comes from an RTD transducer.

Both the measured value and the control signal to the plant are scaled, using the LINCON Special Function S11, to be within the range of 0 to 32000.

A test is conducted by running the valve with a control signal value of 12000 which is then increased, as a step change, to 13500. The corresponding change in the measured value is from 12375 to 12975. The response shows an effective deadtime of 12s and an effective time constant of 140s.

Therefore:

$$A = \frac{12975 - 12375}{13500 - 12000} = \frac{600}{1500} = 0.400$$

$$T_d = 12 \text{ s}$$

$$T_c = 140 \text{ s}$$

Rule of thumb for repetition interval:

$$T = \frac{T_d}{2} = 6\text{s}$$

Let T = 5 secs. The program execution time is liable to be much shorter than this and can be ignored. Therefore:

$$T_d' = T_d + \frac{T}{2}$$

$$= 12 + \left(\frac{5}{2}\right) = 14.5s$$

Using 3-term control we can now pick out the PID terms:

Proportional gain

$$= \frac{1}{A} \times \frac{T_c}{T_d'} \times 1.2$$

$$= \frac{1}{0.400} \times \frac{140}{14.5} \times 1.2$$

$$= 28.97$$

Integral time T_I

$$= \frac{T_d'}{0.5}$$

$$= \frac{14.5}{0.5} = 29s$$

Derivative time T_D

$$= T_d' \times 0.5$$

$$= 14.5 \times 0.5 = 7.25s$$

Lastly, the required table values are calculated:

P = proportional gain in units of 0.01

$$= 2897$$

$$I = \frac{P}{T_I}$$

$$= \frac{2897}{29} = 99.896$$

= 100 to the nearest integer

$$D = P\ T_D$$

$$= 2897 \times 7.25 = 21001.3$$

= 21001 to the nearest integer

Calculation Method

No examples are given here, as worked examples and analytical methods for the design of closed loop systems can be found in various text-books.

Cut-and-Try Method

Firstly, estimate the repetition interval (bearing in mind that this should not be more than half of the plant deadtime), for example, 1 sec.

To begin with there is no requirement for integral or derivative action, so leave the table values for I and D set to zero.

Initially, to achieve a low proportional gain, set P to a low arbitrary value. If the control signal and measured value have been scaled using LINCON so that most of the permissible number range is utilized, it is unlikely that P will need to be less than 100 for the first try.

Begin to increase P in stages, this will speed up the plant response to setpoint changes. Eventually, an optimum setting will be reached, beyond which the plant response becomes more oscillatory.

The measured value will always be found to be somewhat less than the setpoint. Some integral action is required that resets the measured value so that it equals the setpoint. However, integral terms tend to have a destabilizing influence and the P term must first be reduced to 90 per cent of the optimum value.

For example:

If the optimum setting for P is found to be 3520 then P would be set to:

$$3520 \times 0.9 = 3168$$

The value of I can now be increased. Each new trial setting for I must be no greater than twice that of the previous setting.

This is all that needs to be done for 2-term PI control, however, a check should be done to see if the repetition intervals initial setting of 1 second can be made any longer.

For example:

If the optimum value of I is 98, then by using the P and I settings the integral time for PIDSIM will be:

$$T_I = \frac{\textit{Table value for P}}{\textit{Table value for I}}$$

$$= \frac{3168}{98} = 32.3s$$

For 2-term control with a well tuned Controller, it should be possible to use a repetition interval approaching: $0.15T_I$ or $0.15 \times 32.3 = 4.85s$.

The repetition interval can be increased, by altering the value which defines the delay time of the BLOCK containing PIDSIM. Altering the repetition interval should make no difference to the response of PIDSIM or its control loop.

Provided that a derivative term would not introduce to much noise, a 2-term control can be converted into a 3-term control. When D has been wound up to its optimum setting, it should then be possible to increase P by approximately 30 per cent and I by about 100 per cent. Therefore, with the D term added, the P and I terms can be set to :

$$P = 3168 \times 1.3 = 4118$$

$$I = 98 \times 2 = 196$$

The P and I values must not be set until after the D term has been added, otherwise an oscillatory action or an unstable system may be produced. Unlike reset action, derivative action tends to make a control much more stable.

INITIALIZATION

PIDSIM automatically adjusts the size of the integral and derivative terms by taking into account the time interval since its last execution. To do this, PIDSIM uses location Zm+12 to store the time of its last execution. To find the interval since its last execution, PIDSIM reads the time from the Controller on its next execution and subtracts the value in Zm+12 from it.

The time since PIDSIM's last execution is used in calculations for the output and calculations for rate limits.

This technique makes PIDSIM easy to program as nothing needs to be known about program cycle times. However, it does mean that:

PIDSIM must be initialized.

If PIDSIM is not operated within 65 seconds, it must be re-initialized.

First execution

When the program is first run, Zm+12 may contain zero or a time value stored by PIDSIM from when the Controller was last powered up. Therefore, when PIDSIM operates, it could calculate a spurious value for the elapsed time since its last execution and set extremely large gains.

Consequently, PIDSIM's output should be suppressed for the first operation. This can be done by setting the Evaluate/Not Evaluate bit (bit Zm+1.3) in the mode word to OFF. On account of this, PIDSIM will read the time but its output will remain unchanged. For subsequent program cycles, bit Zm+1.3 should be set to ON.

Re-initialization

If the PIDSIM Special Function is contained in a BLOCK which only executes for certain phases of plant operation, it may not have been operated for the past 65 seconds. Because of this, a spurious output value could be caused, as the time read from the Controller could have been 'once round the clock', or more.

In this case, initialization should be included in the BLOCK, so that the PIDSIM function operates for the first time with its output not evaluated but held and is only subsequently released.

LIMITS

Limits in ABSOLUTE OUTPUT Mode

A closed loop system only remains in closed loop mode as long as the plant responds to the demands of the control signal.

For example, suppose signal levels have been chosen to give an output from a PIDSIM function, within the range of 0 to 10000, and that the PIDSIM output has been scaled using a LINCON function and fed to an analog output module to give a 4 to 20 mA output to the plant.

When the setpoint is changed the PIDSIM output could go up to 15000, however, it is unlikely that the analog output module would provide 30 mA, and even if it could, it may have no effect on a valve actuator once the valve is set fully open, as any further increase in signal would make no difference.

Without limits, any large change in setpoint could cause undesirable overshoots in the measured value. This is due to the length of time it takes for the output of PIDSIM to come back down to its working range of values. Until this happens, the plant control signal remains at its maximum value or above and the measured value is driven too high.

For this reason, limits should be set according to the following rules:-

> Limits should be set for the 3-term control function itself and not for subsequent functions operating on the control signal.
>
> Limits should be chosen so that no following hardware or software is saturated, i.e. hardware may be driven flat out but no harder, and software program functions must not be driven to maximum or limit number values.

In ABSOLUTE OUTPUT mode rate limits should be set, so that the control signals rate of change is no faster than that at which the plant can follow. The table values L_1 and L_2 determine the control signals rate of change in bits/s.

Integral Component Limits

The integral component of PIDSIM's output is subject to the same range limits (Zm+6, Zm+7) as the output itself.

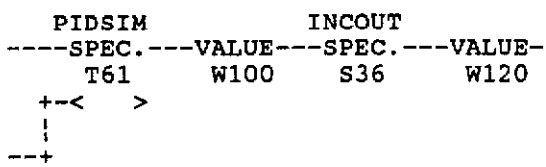
If PIDSIM is operated in the multiply mode, Q_I will be limited to Zm+6 or Zm+7, divided by $I_2/10000$.

Limits in INCREMENTAL OUTPUT Mode

The INCREMENTAL OUTPUT is normally used where its 'home' output value is zero. When the 'home' output value is not zero, an offset can be selected by using the OFFSET table location in PIDSIM. In INCREMENTAL OUTPUT mode HILIM and LOLIM are ignored and PIDSIM's output is subject to the normal number limits, i.e. ±32767.

With many actuators, if the setpoint to the control loop is changed or if there is a disturbance affecting the measured value, the system may come to a steady state condition, where PIDSIM (set for INCREMENTAL OUTPUT) gives an output that is to small to overcome friction. This will cause the actuator to stop short with the loss of proper reset action. Furthermore, the actuator may overheat if left standing with a steady current flowing through it.

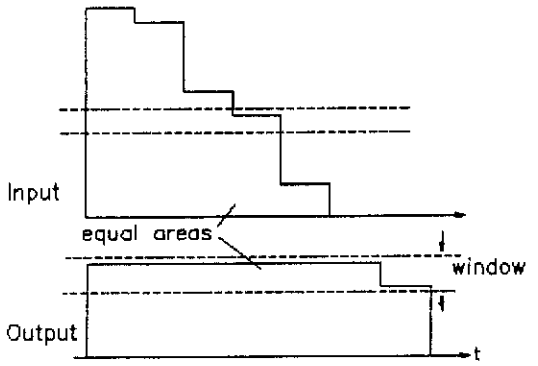
To overcome this problem, add the Special Function INCOUT to the ladder diagram; following PIDSIM (see below).



Characteristics of INCOUT

The INCOUT function will only give outputs which lie between the upper and lower limits of windows symmetrical about zero. The upper limits operate as normal limits, the lower limits provide a deadzone. The deadzone should be set so that small signals which the actuator cannot handle, are blocked.

INCOUT's chief property is an output whose integral is equal to that of its PIDSIM input (subject to the output being within the limit window). This is illustrated in the diagram below.



Note: If the input area had been fractionally less, then the last output value – on the last program scan shown – might have fallen below the lower limit window. In which case the actual output on this scan would have been zero and the value not output would have been held over until PIDSIM provided some further input to INCOUT.

A check can be made using the difference equation for INCREMENTAL OUTPUT, that with a step change of setpoint, PIDSIM would normally give a large positive output on the first program repetition interval, followed by a large negative output on the second interval. INCOUT allows these two spikes to be used in the plant output without them being chopped off by limits or disregarded by the actuator.

MISCELLANEOUS APPLICATION NOTES

PIDSIM can be set to operate in ERROR DERIVATIVE mode or in MV DERIVATIVE mode by setting bit zero of mode word Zm+1 (see Table 4). If not set, PIDSIM defaults to MV DERIVATIVE mode.

PIDSIM calculates its derivative term from rate of change of error in ERROR DERIVATIVE mode; and from rate of change of measured value in MV DERIVATIVE mode.

Compared with the ERROR DERIVATIVE mode, use of the MV DERIVATIVE mode reduces the size of transient output changes when there's a rapid change of setpoint. However, ERROR DERIVATIVE mode may be more appropriate for the inner loop of cascade controls.

In ERROR DERIVATIVE mode, a genuine step change of setpoint cause a transient output for one execution only. If the DERIVATIVE gain is of reasonable size this spike will be clipped by the appropriate output limits and the derivative action will be largely lost. To prevent this occurring, step setpoint changes should be avoided. To do this, use RAMPGEN to smooth out the changes or rate limits of a previous function providing the setpoint. Even though the setpoint input signal changes slowly this may give you a faster output response.

MULTIPLY/NOT MULTIPLY Modes

PIDSIM can be set to operate in MULTIPLY mode, or in NOT MULTIPLY mode by setting bit 2 of mode word Zm+1. If not set, then PIDSIM defaults to NOT MULTIPLY mode.

If MULTIPLY MODE is selected, PIDSIM makes use of the integral gain I_2. If MULTIPLY MODE is not selected, any value for I_2 is ignored.

I_2's effect on the error is exactly the same as that of the normal integral gain I, however, I_2 also multiplies the accumulated error Q_I . The basic equation when MULTIPLY mode is selected, is modified as follows:

$$\left[Q_n = \frac{P}{100} E_n + \frac{D}{100t}(X_n - X_{n-1}) + \left(\frac{It}{100} E_n + Q_I\right)\left(\frac{I_2}{10000}\right) + Offset \right]$$

I_2 can be used in MULTIPLY mode when the rung output Q_n is to vary with an external variable (i.e. heat input with flow). Instead of waiting for the closed loop to compensate for the disturbance, the heat input Q_n can be altered immediately by making I_2 proportional to flow. To maintain stability, the value of I will probably have to be modified and the output OFFSET set correctly if the heat input signal has an elevated zero.

I_2 can also be used as an additional multiplier for the integral term when very long integral times are required. In the equation (since I_2 is divided by 10,000) setting I_2 to 100 makes the integral gain become 1 per cent of its value in NOT MULTIPLY mode, thereby allowing integral times to be set 100 times longer.

Note: I_2 is restricted to the range 0-10,000. Any value larger than 10,000 will be treated as 10,000.

Output Offset

For a control using absolute output mode, if the output to the plant has an elevated zero (i.e. 4mA with a 20mA signal span), the output can initially be set to this value with the offset value in Zm+10. Although an offset is not needed on the output; on a control that includes integral action, the loop will start up quicker from switch on with OFFSET. Furthermore, if any control loop adaption is required, the integral term will only have to reset the measured value equal to the setpoint and can be modified by varying I_2.

Suicide

To operate PIDSIM normally; set suicide to off by setting bit 4 of the mode word equal to 1. If this bit is subsequently set to zero the output from PIDSIM will fall to zero instantly (i.e. without any rate limit action). Suicide operates this way for both ABSOLUTE OUTPUT mode and INCREMENTAL OUTPUT mode.

Operation of the deadband facility

It may be found that a control loop 'hunts' if a small error persists due to external effects. This is more likely on integral control. However, due to quantization errors, it may also occur (even without integral action) on the DAC output and on the ADC of the measured value. To prevent this, PIDSIM includes a deadband facility for the setting of a deadband value, DBAND, which operates on the error signal as follows:

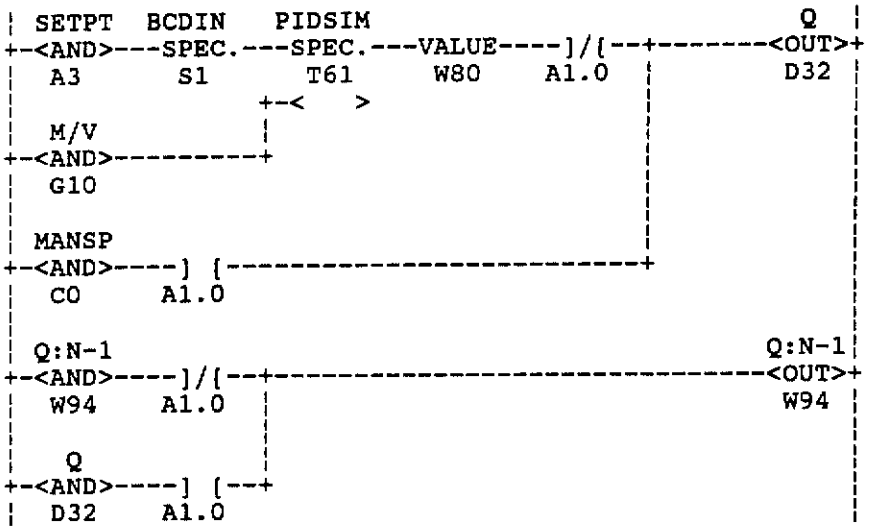
If: |error| <DBAND then: error = 0
If: error>DBAND then: error = error−DBAND
If: error<−DBAND then: error = error+DBAND

Bumpless changeover

With closed loop controls using INCREMENTAL OUTPUT mode, there is no problem changing over from automatic closed loop control to manual control. Under steady state conditions PIDSIM will give zero output with any offset required, if the home value of the control signal being provide by a following LINCON Special Function is not zero.

However, with closed loop controls using the ABSOLUTE OUTPUT mode, the PIDSIM output must (under steady state conditions) be of the appropriate size and generally not zero. Therefore, when a change over from manual to automatic control becomes necessary, the PIDSIM output must be preloaded with a value which equals that of the manual control output. This can be done by arranging for PIDSIM's previous output table location to track the manual control output.

This will make sure that a changeover from manual (A1.0 closed) to auto (A1.0 open) is bumpless, but not a changeover from auto to manual. If a bumpless changeover from auto to manual is required, then the manual setpoint MANSP could be fed in via a RAMPGEN Special Function and the RAMPGEN output could be preset by using a rung similar to the second rung shown in the example below.

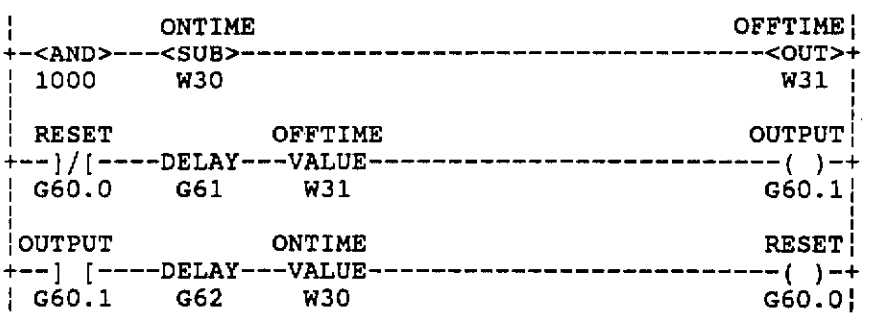


Note: Table location W94 is the 15th parameter in PIDSIM's parameter list, which begins at table location (W80).

Contactor Outputs

In some applications, it may be necessary for the output of a closed loop control to switch contactors on and off with a variable on-to-off time ratio, i.e. when controlling electric heating elements.

For such applications, PIDSIM must be used in the ABSOLUTE OUTPUT mode, with its output feeding a variable mark-space generator. The following example shows a possible realization. In this section of program, the PIDSIM output is taken into table location W30 and has been arranged to lie within the range of 0 to 1000.



The first rung is designed so that there is a constant cycle time of 100 seconds (i.e. 1000 units of 0.1 secs). The PIDSIM output into W30 sets the on-time, subtracting the on-time from 1000 gives the required off-time. The second rung (with G60.0 initially de-energized) provides an output after a delay equal to the off-time, the output then initiates the on-time delay in the third rung and the output from G60.1 remains on until this delay has timed out. When the on-time delay has timed out, G60.0 resets the off-time delay in the second rung and a de-energizing G60.1 resets the on-time delay in the third rung. The sequence then repeats.

Such a section of program should be operated on every program scan, it should not be put into a BLOCK which operates at a slower repetition interval so that the DELAYs operate correctly.

Reverse Acting Controls

With certain types of control loop, the controller output may need to be reduced with increasing error. To provide this type of control the S5 Special Function NEGATE should follow PIDSIM. For start up purposes the initial output value can be set by means of the OFFSET value in Zm+10.

EXERCISES

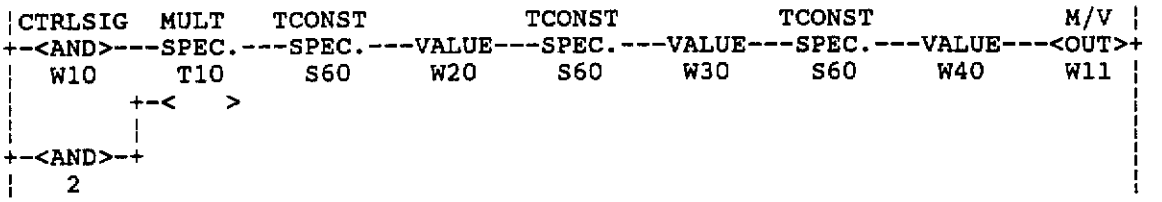
With a GEM80 Controller and an 8902 simulation unit, PIDSIM can be tried out using the following simple exercises, in which the plant is simulated within the GEM80 program itself.

In both exercises, the setpoint is taken from the thumbwheel switches (A3) and the measured value is displayed on the LCD numeric display (B2).

Absolute Output Exercise

Set the program repetition interval to 100ms by making P1 = 100.

Simulate the plant with the following rung (see below):

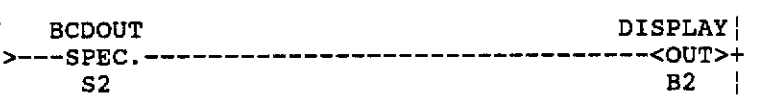


Set up data for the TCONST functions as follows:-

W21 = 100 W31 = 100 W41 = 2000
W23 = −1 W33 = −1 W43 = −1

Explanation: The rung represents the plant as a gain of 2 (via the MULT) and as three time constants in cascade. The first two are time constants of 1 second and the other a time constant of 20 seconds. When several time constants are put in cascade the larger ones should always be put last in the rung and the smaller ones first; see Software Data Sheet S60 for the Special Function TCONST.

Insert the following rung to display the measured value on the LCD:



This rung uses BCDOUT to convert the output of the previous rung into a suitable form to drive the display.

Before putting a closed loop control round the 'plant', get the feel of its response by running it open loop. To do this, insert the following rung, and directly drive the plant control signal W10 using the thumbwheel switches.

```
| INPUT  BCDIN                                    CTRLSIG|
+-<AND>---SPEC.-----------------------------------<OUT>+
|   A3     S1                                      W10  |
```

For a large change of input signal (e.g. if the thumbwheel switches are altered from 0000 to 1000) it will be found that the number displayed changes slowly for the first few seconds then speeds up to its maximum rate of change, after which, the rate of change falls off, and it takes a long time to move to the last figure. Since the plant has a gain of 2 the final figure should be twice the figure set on the thumbwheel (as long as the former is less than 9999, which is the largest number that the LCD can display).

If the measurement method is attempted, a plot of the plant response will have to be made to find the deadtime and the effective time constant. One way of doing this is to route the measured value output from W11, to an analog output (e.g. D32) to which a pen recorder can be connected. Another way, is to write a section of program that copies the contents of W11 into a sequence of table locations every 5 seconds. Subsequently, the table locations can be read off as a data list on the programmer screen and plotted out as a graph.

If the cut-and-try method is used, insert the following rungs and delete the rung used for driving the plant control signal 'W10' from the thumbwheel input A3.

```
|                                                    |
+--]/[----DELAY---VALUE-------------------------( )-+
| G0.0     G1       5                           G0.0  |
|                                                    |
+--] [-----------------------------------------OBEY-+
  G0.0                                           BLOCK
                                                  *
**************************************************
|                                                    |
| INPUT   BCDIN  PIDSIM                       CTRLSIG|
+-<AND>---SPEC.---SPEC.---VALUE-----------------<OUT>+
|  A3      S1      T61      W100                  W10  |
|                +-<   >                             |
|  M/V           |                                    |
+-<AND>---------+                                    |
|                                                    |
|                                                    |
+ ***************** END OF BLOCK ****************** +
|                                                    |
|                                                    |
+--------------------------------------------( )-+
|                                              W101.3|
```

In the preceding ladder diagram, the first pair of rungs arrange for the following BLOCK to be executed at 0.5 second intervals. The working rung, containing the 3-term control signal PIDSIM, forms the setpoint from the thumbwheel input A3 and then converts it to decimal form with the Special Function BCDIN. The measured value M/V is fed into PIDSIM's other input.

The last rung sets PIDSIM to evaluate, but before this rung is reached, PIDSIM will operate once in non-evaluate mode, where it will read the time from the Controllers millisecond clock but will not change its output.

The PIDSIM data tables should now be set up as given in Table 2:

W101 = @10 = bit 4 set ON
W102 = P; try 100 initially
W103 = I; set to 0 initially
W104 = D; set to 0 initially
W105 = Deadband = 0
W106 = LOLIM = −32000
W107 = HILIM = 32000
W108 = Rate limit away from zero = 32000
W109 = Rate limit towards zero = 32000
W110 = Output offset = 0
W111 = Post integral gain = 0 (since multiply mode not used)

Setting bit 4 of mode word W101 to ON, takes suicide off. Table values W106 to W109 set the limits and rate limits to large values so that they do not interfere with the system. Initially, W105, W110 and W111 are all set to zero. W112 to W118 contain intermediate values used by or calculated by PIDSIM and should not be written to.

Try changing the setpoint on A3 and noting the plant response at B2. Try altering the values of P, I and D in W101 to W103. There will be little difficulty in making the system unstable if the values are made to high.

If the analysis method is chosen use whichever method is most familiar (i.e. Bode, Nyquist etc...). In the example below a bode diagram is used. First of all, note that the transfer function of the plant is:

$$\frac{2}{(1+20p)(1+p)}\ 2$$

for which the amplitude and phase shift plots are as shown below.

If the Controller gain is set to 10 the phase shift at crossover is approximately 177°, which is far to close to 180°. Putting in a derivative term, with a derivative time of 1 second, reduces this to approximately 132° (see dotted line). The repetition rate which the 3 term control operates at has to be chosen. If it is twice per second, this will mean operating PIDSIM at 1/2 second intervals which gives an effective time delay of 14° phase shift at crossover (see dashed line) but makes no difference to the amplitude plot. The phase shift at crossover is then 146°. Putting in an integral time of 4 seconds (see chain line) introduces another 14° of phase shift at crossover, making a total of 160°. The system is then stable, although somewhat wobbly, with only a 20° phase margin. For these values the corresponding table values will be:

W102 = P in units of 0.01 = 1000 for a gain of ten

W103 = I = P/T_I = 1000/4 = 250

W104 = D = $P.T_D$ = 1000 x 1 = 1000

and the 1/2 second interval can be set, as in the cut-and-try example, with a BLOCK executed every 1/2 second.

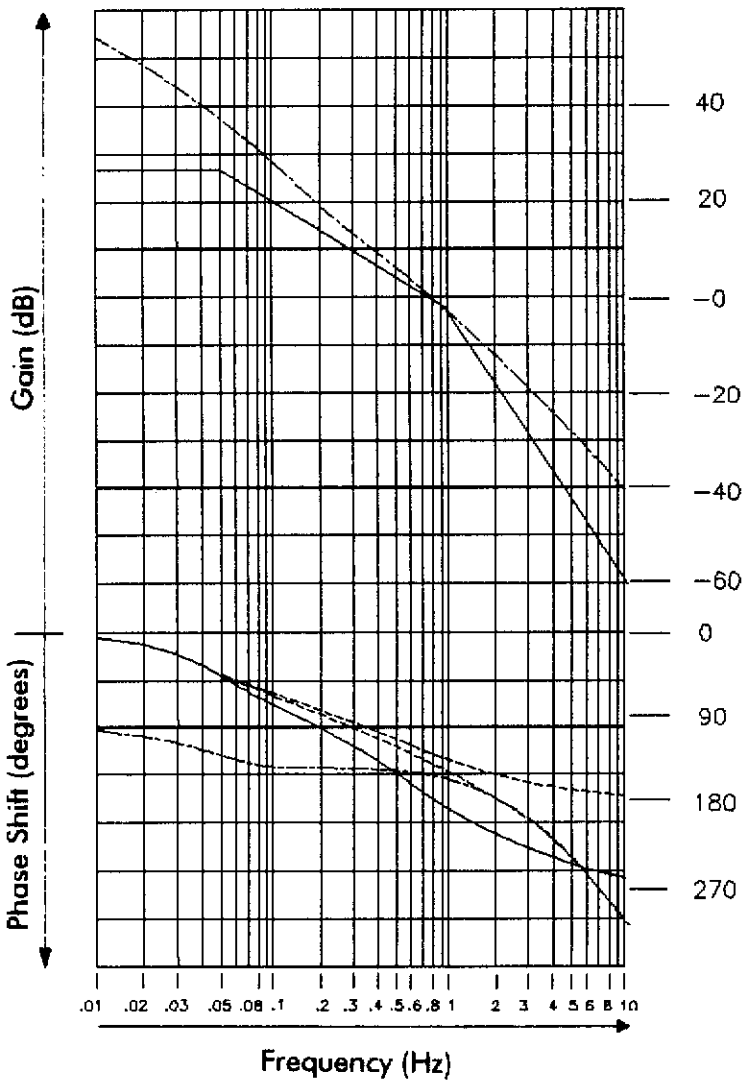
Incremental Output Exercise

Simulate the plant with the same rung as before, but insert an ADD; with location W11, before the final element OUT to W11; i.e:

```
|CTRLSIG  MULT   TCONST          TCONST          TCONST            M/V     M/V |
+-<AND>---SPEC.---SPEC.---VALUE---SPEC.---VALUE---SPEC.---VALUE---<AND>---<OUT>+
|  W10     T10     S60     W20     S60     W30     S60     W40     W11     W11 |
|       +-<   >                                                               |
|       |                                                                     |
|    +-<AND>-+                                                                |
|      2                                                                      |
```

Set PIDSIM to INCREMENTAL OUTPUT mode by making W101.1 = 1, then try running it similar to the ABSOLUTE OUTPUT mode exercise. The W-tables W106 to W109 will be ignored in INCREMENTAL OUTPUT mode.

Explanation: The additional ADD in the plant simulation rung means that, under steady state conditions the control signal in W10 must be zero. For this condition, the rung adds nothing to W11, so that W11 stays put at whatever value it has reached with no control signal input. This is the sort of plant characteristic that the INCREMENTAL OUTPUT mode is intended to control, where, after a change of setpoint, the control signal always comes back to its 'home' value (in this case zero).



Status Indications

If a fault is detected, then:

- bit 15 of the MODE WORD (zm+1) is set ON

- a specific bit of the FAULT WORD (Zm) is set ON

(In this context, a 'fault' means anything that prevents PIDSIM evaluating its output normally. For instance, the term 'fault' includes the output going into limit which is likely to happen during normal operation. Therefore, the term 'fault' does not necessarily mean an operational fault condition)

Bit 15 of the MODE WORD (Zm+1.15) is set to OFF at the beginning of each PIDSIM execution. If bit 15 is OFF on exit from PIDSIM then no faults were detected on that scan.

Table 3 gives the meanings for particular bits that are ON and the outputs that PIDSIM will give under fault conditions. If bit 9 in the fault word is ON, then at least one of the input data table values is outside the allowed range given in Table 2. In this case, PIDSIM defaults to NOT EVALUATE mode and the rung output remains at the output of the previous scan (Zm+14). The time since the last execution continues to be calculated.

Single Cycle Operation

For test purposes, PIDSIM may be operated in single cycle mode so that output changes can be observed one execution at a time. When a Controller containing PIDSIM is set to single cycle mode, the time is artificially incremented by 100 msecs each operation, rather than operating in real time mode.

CEGELEC
INDUSTRIAL CONTROLS

Kidsgrove, Stoke-on-Trent, ST7 1TW, England
Tel: (0782) 783511 Fax (0782) 776329

 Publication No. T1313 Issue 1

ALSTOM Power Conversion

France
9, rue Ampère
91345 Massy Cedex
Sales Tel: +33 (0) 8 25 02 11 02
Support Tel (International):
+33 (0) 3 84 55 33 33
Support Tel. (National):
08 25 02 11 02

Germany
Culemeyerstraße 1
D-12277 Berlin
Sales Tel: +49 (0) 30 74 96 27 27
Support Tel (International):
+49 (0) 69 66 99 831
Support Tel (National):
01 80 3 23 45 72

UK
Boughton Road, Rugby
Warwickshire, CV21 1BU
Sales Tel: +44 (0) 1788 563625
Support Tel: +44 (0) 1788 547490

USA
610 Epsilon Drive
Pittsburgh, PA 15238
Sales Tel: +1 412 967 0765
Support Tel: +1 800 800 5290

ALSTOM