RAMPGEN function for smoothing process variable changes

RAMPGEN S61

DESCRIPTION

The RAMPGEN Special Function is intended for smoothing out rapid changes. It provides an output whose rate of change is limited to preset or programmed values. Additionally, it can provide rounding at the beginning and end of any ramped output.

As one of the major applications for the RAMPGEN function is for variable speed drive systems, the terminology used in this leaflet is largely based on this type of application. However, the RAMPGEN function finds uses in many other applications, particularly where bumpless transfer is required between different operating modes in process control.

FEATURES

- All variable changes are smooth and jerk-free due to rate limits and rounding.
- Independent rates for acceleration (variable moving away from zero) and deceleration (variable moving towards zero).
- Ramp output normally follows input variable, but subject to overriding logic inputs for Hold, Raise, Lower or Suicide
- Provision for input (e.g. representing torque limitation) to reduce output rate of change.
- Output limits.
- Rated Output (e.g. for use in feed-forward controls)

SPECIFICATIONS

Control Language

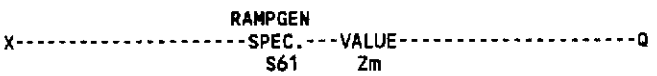


Table 1 Program Rung Data

Sym	Location	Use	Range
X	Rung input	Input to RAMPGEN from rung	- 32768 to +32767
Q	Rung output	Output from RAMPGEN	- 32768 to +32767 (but subject to Q_{POS} and Q_{NEG} limits)
Zm	Value location	Start of parameter list (13 locations)	Any write enabled table usable by special functions

Quantity of data table locations:

13

Equation for rung output:

Q = X when input X steady, with no logic inputs.

Q tends to X when X changes, subject to limit and logic constraints as described in the applications notes below.

Logic inputs:

SUICIDE input forces Q to zero instantly.

LOWER input causes Q to ramp towards zero while applied (independently of X). When LOWER input removed, Q ramps back up to match X.

RAISE input causes Q to ramp away from Zero while applied (independently of X). When

RAISE input removed, Q ramps back down to match X.

HOLD input causes Q to cease ramping. Where rounding is used the ramp will round off and the output Q will remain constant at whatever value it has reached. While HOLD is applied the Input X is ignored.

Logic priority:

SUICIDE overrides HOLD, LOWER and RAISE.

HOLD overrides LOWER and RAISE.

LOWER overrides RAISE

Table 2 Value Data			
Symbol	Location	Use	Range
F	Zm	Fault code and flags. Set by RAMPGEN	See Table 3
MODE	Zm+1	Logic control bits set by user	See Table 4
T_{ACC}	Zm+2	Acceleration time set by user	0 to the greater of Q_{POS} and Q_{NEG}. Corresponds to ramp times of 0 to 320.00s in steps of 0.01s for an output change from 0 to 32000
T_{DEC}	Zm+3	Deceleration time set by user.	0 to the greater of Q_{POS} and Q_{NEG}. Corresponds to ramp times of 0 to 320.00s in steps of 0.01s for an output change from 32000 to 0
T_C	Zm+4	Rounding time constant set by user.	0 to +32000 Corresponds to time constants of 0 to 320.00s in steps of 0.01s

Table 2 (continued)			
Symbol	Location	Use	Range
T	Zm+5	Time of last execution. Set by RAMPGEN	0 to 65535 (msecs)
Q_{POS}	Zm+6	Positive output limit set by user	0 to +32000
Q_{NEG}	Zm+7	Negative output limit set by user	0 to -32000
I	Zm+8	Current limi input set by user using separate rungs, or else defaults to 0 (zero)	-32768 to +32767 Normally zero. Positive value reduces accel when Q pos. Negative value reduces accel when Q neg.
Q_{n-1}	Zm+9	Previous value of output (unrounded). Stored by RAMPGEN. Used in next scan.	You may overwrite this value, e.g.for bumpless transfer.
Q_{n-1}	Zm+10	Previous value of output (rounded). Stored by RAMPGEN. Used in next scan.	You may overwrite this value, e.g.for bumpless transfer.
Q'	Zm+11	Rate of rounded output Stored by RAMPGEN. Used in next scan.	Can be used as output for feedforward. Q'=500(Q-Zm+10)/t where t = time since last execution (ms).
Q_r	Zm+12	Rate remainder Stored by RAMPGEN. Used on next scan.	Meaningless to the user. This value should not be made use of as it may vary with the implementation.

Table 3 Fault conditions and Flags			
Condition	Result	Code in Zm	
		Value	Bits Set
Programming Errors			
No VALUE after Spec	Q = 0	-	-
VALUE given number instead of address	Q = 0	-	-
VALUE address Zm in write-protected memory	Q = 0	-	-
VALUE address Zm in table not usable by Special Functions	Q = 0	-	-
Data Errors			
No rounding $T > T_c$	As normal	3	Zm.0=1 Zm.1=1
Invalid acceleration time ACC <= 0 or > 32000	Q = 0	5	Zm.0=1 Zm.2=1
Invalid deceleration time DEC <= 0 or > 32000	Q = 0	9	Zm.0=1 Zm.3=1
Invalid rounding time constant $T_c < 0$ or > 32000	Q = 0	17	Zm.0=1 Zm.4=1
Invalid positive limit $Q_{POS} < 0$ or > 32000	Q = 0	65	Zm.0=1 Zm.6=1
Invalid negative limit $Q_{NEG} > 0$ or < -32000	Q = 0	129	Zm.0=1 Zm.7=1
Status			
Rate limit overflow. Rate output Q' limited at -32767 or 32767	As normal	33	Zm.0=1 Zm.5=1
Output limited at positive limit	$Q=Q_{POS}$	+20481	Zm.12=1 Zm.14=1 Zm.1 =1
Output limited at negative limit	$Q=Q_{NEG}$	-28671	Zm.12=1 Zm.15=1 Zm.1 =1

Table 4 Mode Word Bit Settings				
Sym	Location	Use	State	Effect
S	Zm+1.0	Suicide logic input. set by user, either preset to suicide off, or else controlled by separate rung.	0 1	Suicide on; Q reduces to zero. Suicide off; normal operation.
H	Zm+1.1	Hold logic input. Set by user, either preset to Hold off, or else controlled by separate rung.	0 1	Hold on; ramp rate reduces to zero, after which Q stays constant. Hold off; normal operation.
R	Zm+1.2	Raise logic input. Set by user, either preset to raise off, or else controlled by separate rung	0 1	Raise off; normal operation. Q ramps away from zero.
L	Zm+1.3	Lower logic input. Set by user, either preset to lower off, or else controlled by separate rung	0 1	Lower off; normal operation. Q ramps towards zero.

APPLICATION NOTES

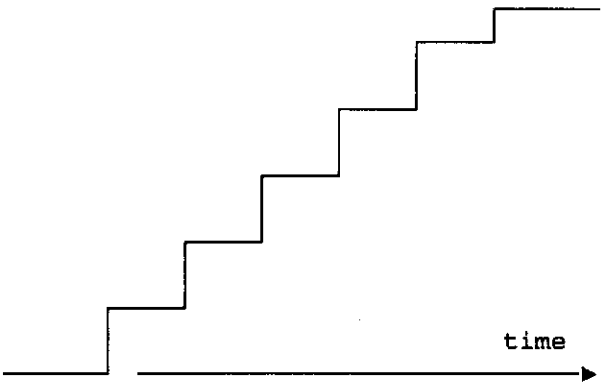
Ramp Rate

Although the RAMPGEN Special Function has facilities for rounding off the start and end of a ramp, to begin with we shall ignore them and consider a ramp without rounding.

While the rung input to a RAMPGEN function does not change, the rung output remains equal to the input; that is a RAMPGEN function has unity gain. The rung output from the RAMPGEN function will also equal the rung input if the rung input changes slowly.

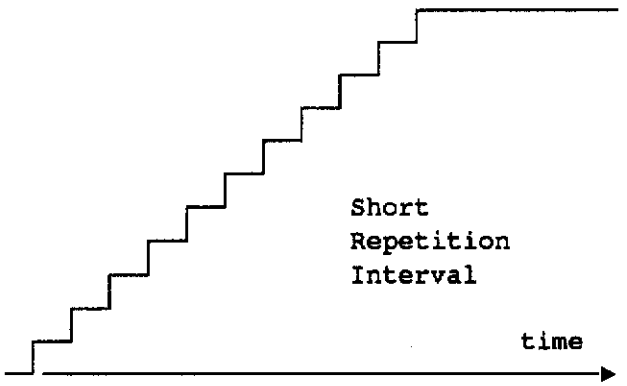
Where there is a much more rapid change of the rung input the output is rate limited. Assuming that the time interval between executions is constant, the

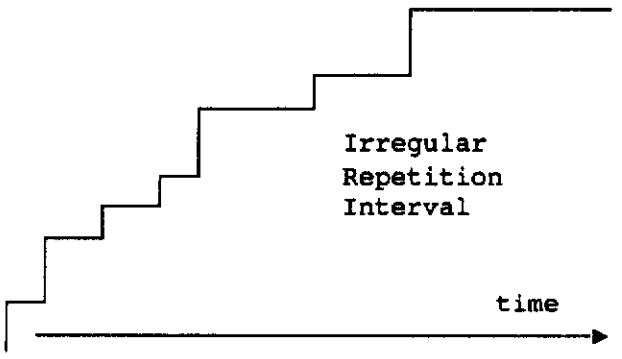
output increases or decreases by a fixed number on each subsequent execution, until the rung output is again equal to the rung input. Consequently, the rung output when plotted as graph is a staircase (see below)



Note...The last step of the ramp will often be smaller than the other steps so that, finally, the rung output will be exactly equal to the rung input.

To make the slope of the staircase constant, regardless of the program repetition interval, RAMPGEN automatically adjusts the step size if the repetition interval ever changes. This is made clear by the diagram opposite.





For use with analog control, the steps in the staircase should be as small as possible to provide the smoothest possible ramp. For this reason, RAMPGEN functions should be operated on every program execution. Do not place a RAMPGEN function in a BLOCK which is operated less frequently unless you have a very slow ramp rate. Also, a RAMPGEN function should never be left without being operated for more than 65 seconds unless it is re-initialized.

The RAMPGEN function has two table values, Zm+2 and ZM+3, for setting the ramp times. The first, referred to as the 'Acceleration time' value, sets the ramp time when the rung output is moving away from zero. The second, referred to as the 'deceleration time' value, sets the ramp time when the rung output is moving towards zero.

The values for Zm+2 and Zm+3 must be:-

the time it would take the output of RAMPGEN to move from 0 to full output

the time it would take the output of RAMPGEN to move from full output to 0.

The full output values can be set independently in positive and negative directions, by the values given to Zm+6 and Zm+7 (referred to by the mnemonics Q_{POS} and Q_{NEG}). If these values have different magnitudes when set, then the value of the largest magnitude is taken as full output when calculating the ramp rates from the ramp times.

Initialization

For a given ramp rate, RAMPGEN adjusts the step size in the staircase to keep the rate constant, regardless of the interval between executions. To do this, RAMPGEN uses location Zm+5 to store the time of its last execution. The next time RAMPGEN executes, it reads the time in the Controller and subtracts the value in Zm+5 from it to find the interval since its last execution.

This technique makes RAMPGEN easy to program. However, it does mean that:-

RAMPGEN must be initialized

If RAMPGEN is not operated within 65 seconds it needs to be re-initialized.

First Execution

When the program first runs Zm+5 may contain zero or a time value stored by RAMPGEN when the Controller was last powered up. Therefore, when RAMPGEN operates, it may calculate a spurious value for the elapsed time since its last execution, and from this set an extremely large step size.

Consequently, changes to the RAMPGEN output should be suppressed during its first operation by using the HOLD logic input. When this is done, RAMPGEN reads the time but its output remains unchanged. For subsequent program cycles the HOLD logic input can be removed.

Re-initialization

If a RAMPGEN Special Function exists in a block that only executes for certain phases of plant operation, it is possible that the RAMPGEN function may not be operated for periods of more than 65 seconds.

In such a case a spurious value of RAMPGEN output could be caused, as the time read from the Controller could have been 'once round the clock'.

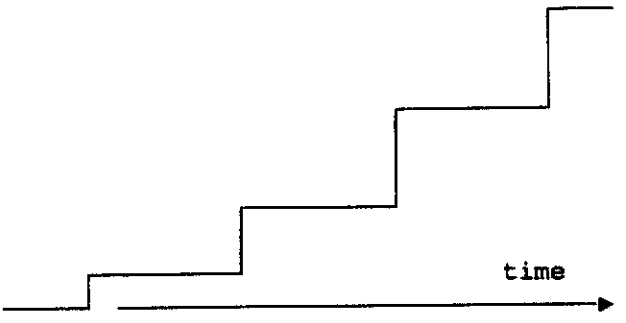
To prevent this, initialization should be included in the block so that the RAMPGEN function operates for the first time with its output held, and for any subsequent operations released.

Rounding at the Start of a Ramp

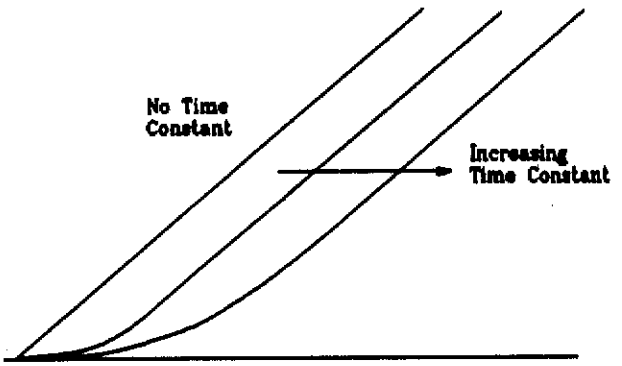
For a simple ramp, the output from RAMPGEN is a staircase, where the step size (give or take one bit) is the same for every program execution so long as the repetition interval is constant.

For a ramp which has rounding at the start, the steps must begin very small and increase in height until the desired ramp rate is attained (see opposite).

The RAMPGEN function provides for this by exponential smoothing, this is equivalent to feeding the output of a linear ramp through a time constant. At the start of a ramp the step size is initially small but increases until the full step size required for acceleration or deceleration is reached (as set by Zm+2 and Zm+3), after which the stepsize remains constant. Similarly, when the end of a ramp is reached the step size reduces over a number of program executions to zero.



To set the required amount of rounding, a time constant value should be entered into table location Zm+4. The diagram below illustrates the effect of altering the rounding time constant.



Mode Word

Various modes of operation of the RAMPGEN function can be controlled by setting bits in the 'Mode Word' Zm+1, as given below.

SUICIDE

For normal running, set the Suicide input Zm+1.0 to ON (i.e. set to 1).

If Zm+1.0 is set to OFF (i.e set to 0), the RAMPGEN output immediately switches to zero as a step, without any ramping or rounding. In practical applications, this facility is normally reserved for abnormal conditions, e.g. emergency stopping.

The suicide input overrides all other logic inputs.

HOLD

For normal running, set the Hold input Zm+1.1, to ON (i.e. set to 1).

If Zm+1.1 is set to OFF (i.e. set to 0), the RAMPGEN output freezes. It will remain at whatever value it has reached until such time as Zm+1.1 is switched on again, regardless of any subsequent change of the rung input,

Hold overrides all other logic inputs except Suicide.

RAISE

For normal running, set the Raise input Zm+1.2, to OFF (i.e. set to 0).

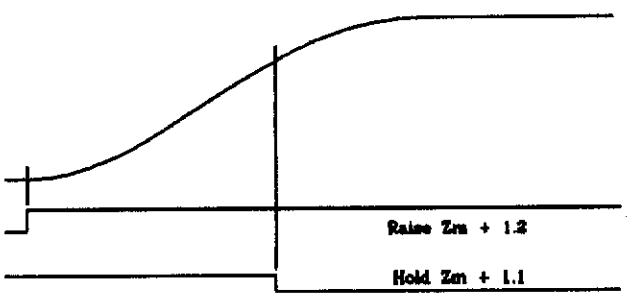
If Zm+1.2 is set to ON (i.e. set to 1), the RAMPGEN output will increase (i.e.

move away from zero) at the acceleration rate limit ACC, set by Zm+2. It will carry on increasing as long as Zm+1.2 is ON, or until the output reaches its limit. When Zm+1.2 is set to OFF the RAMPGEN output will decrease until it once more becomes equal to the rung input of the RAMPGEN function

If the rung output from the RAMPGEN function is positive at the time that Zm+1.2 is set to ON, the RAMPGEN output will move in a positive going direction.

Rounding occurs at the start of ramping when Raise is applied. Rounding also occurs if RAMPGEN reaches its output limit.

In a practical application the Hold input would normally be applied at the end of a Raise and end-of-ramp rounding would take place (see below).



Raise is overridden by all other logic inputs.

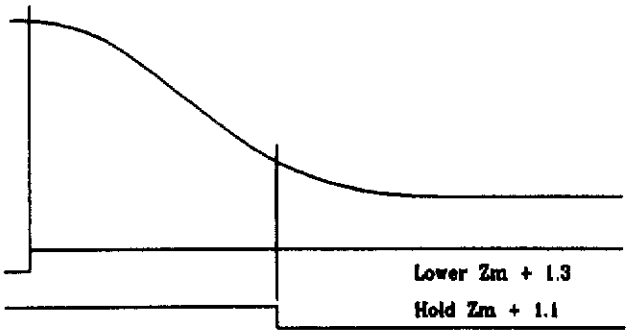
LOWER

For normal running set the Lower input Zm+1.3 to OFF (i.e. set to 0).

If Zm+1.3 is set ON (i.e. set to 1), the RAMPGEN output will decrease (i.e. move towards zero) at the deceleration rate limit DEC set by Zm+3. The output will carry on decreasing so long as Zm+1.3 is ON, or until the output reaches zero. If it reaches Zero it stops, and does not continue ramping in the opposite direction. When Zm+1.3 is set to OFF the RAMPGEN output will increase until it becomes equal to the rung input of the RAMPGEN function.

Rounding occurs at the start of ramping when Lower is applied. Rounding also occurs if the RAMPGEN output reaches zero.

In a practical application the Hold input would normally be applied at the end of Lower and end-of-ramp rounding would take place (see below).



Lower overrides all other logic signal inputs except Hold and Suicide.

Common Setpoints

Two of the RAMPGEN special function features are designed for use where a RAMPGEN function provides a common setpoint to a number of subsystems. A typical example is where a RAMPGEN function provides the master speed set-point for the speed controlled drive systems, on a multi-stand rolling mill.

Rate output

For such an application the speed of response of all drive systems may not be identical. If the speed setpoint comes from a RAMPGEN function the true speed of each drive will also ramp up, but, depending on the response of the particular drive will lag behind the setpoint by a certain time.

To compensate for different time lags a certain amount of rate-of-change can be added to modifiy individual setpoints. To cater for this the RAMPGEN function provides an output rate-of-change, which is stored in table location Zm+11. The following example shows how the rate-of-change value can be used within a program.

```
| INPUT  RAMPGEN                                    MSTRS/P|
+-<AND>---SPEC.---VALUE-----------------------------<OUT>+
| W100     S61     W200                               W250|
|
| RATE   LINCON          MSTRS/P                     S/P/1|
+-<AND>---SPEC.---VALUE---<ADD>---------------------<OUT>+
| W211     S11     W300   W250                        B10 |
|
| RATE   LINCON          MSTRS/P                     S/P/2|
+-<AND>---SPEC.---VALUE---<ADD>---------------------<OUT>+
| W211     S11     W310   W250                        B11 |
```

In the previous example the output rate-of-change from the RAMPGEN function would be stored in location; Zm+11 = W(200+11) = W211. This value is scaled by a suitable factor in each of the LINCON Special Functions and then added to the master setpoint (MSTRS/P) to form the individual setpoints (S/P/1 and S/P/2).

Note...The value held in Zm+11 equals; the current ramp step size, divided by the time (in msec's) since the last execution, multiplied by 500.

Current Limit Output

For maximum throughput in a rolling mill the mill needs to be accelerated up to the designed operating speed as quickly as possible, therefore, the ramp rate for the common setpoint needs to be as fast as possible.

However, as the ramp rate setting is wound up, one of the drives will hit its current or torque limit (Which particular drive this will be depends on the rolling profile set for the mill). The drive will then fail to follow the setpoint, lagging increasingly behind it, and the material being rolled will either break or throw a loop.

To provide for this condition the RAMPGEN function offers a 'Current Limit'. This function takes the form of a value placed in table location Zm+8 that allows the ramp rate to be increased or decreased. However, the ramp rate cannot be made faster than the acceleration or deceleration rates set by Zm+2 and Zm+3.

To decrease the ramp rate during acceleration Zm+8 must be set to:-

positive while the RAMPGEN output is positive.

negative while the RAMPGEN output is negative.

To use this facility, an input signal must be fed to the GEM 80 Controller from each drive to indicate when a particular drive is about to go into limit. These signals must be analog or numeric inputs, not ON-OFF logic inputs.

For normal running, arrange for the signals to be at zero and proportional to the amount by which they exceed a threshold level. The amount by which the signal exceeds the threshold level can be used to determine how much the ramp rate must be reduced by. Obviously the threshold level needs to be set slightly below the current or torque limit setting of the drive.

Next, it is necessary to 'OR' the signals from the various drives, so that the largest signal is used to slow down the RAMPGEN function. (i.e. the one coming from the drive which is operating nearest to the current limit).

Note...When signals are used as described above to slow down the RAMPGEN function, then a closed loop system is constructed. That is, when a signal slows down the RAMPGEN function it will affect the accelerating current through the motors, which in turn reduces the signal slowing down the RAMPGEN function. If the closed loop system is stable it should settle down to a balanced condition, however, if the gain round the loop is too high the RAMPGEN rate could fluctuate about the desired ramp rate value. Therefore, the design of the closed loop system must be checked. The gain round the loop can be reduced if the threshold level is dropped (i.e. if a bigger change of current is used to give the same change in ramp rate). Stability can also be improved by starting the end-of-ramp rounding further away from the end of the ramp, thus making the equivalent time constant longer.

With a closed loop system make sure that signal polarities are correct for all conditions (acceleration, deceleration, positive output and negative output), otherwise positive feedback will occur.

Check also that Zm+8 contains zero (apart from when the RAMPGEN rate is to be modified).

Note...An input to Zm+8 can cause the RAMPGEN output to be driven up or down beyond the level set by the rung input. However, the Zm+8 input cannot make the ramp rate exceed the values set by Zm+2 and Zm+3.

Bumpless Transfer

In many process control systems there is a need to change from AUTO to MANUAL control without any bump. This can be done by setting up the value of the previous unrounded output (table location Zm+9) prior to change over. The RAMPGEN output will then ramp up or down as required from this setting and the user will avoid any step change in the signal.

If the RAMPGEN function needs to be preset rapidly (e.g. in a single program cycle), Zm+10 should also be preset as well as Zm+9. Both table locations should be set to contain the same value.

Choice of Data Tables

The following factors determine which data tables should be used for the Zm values:

1. Whether the table is write protected.

   The table used for the value data must not be write-protected. If it were, RAMPGEN would be unable to store the values of Q_{n-1}, Q' or R, which are required for the next calculation on the next scan.

2. Whether the table contents are cleared on power up.

   It may be necessary to retain the values of Q_{n-1}, Q' and R while the Controller is powered down for some applications, whereas for other applications this may be unnecessary.

3. Whether the table contents are accessible by video.

   If it is neccesary to display any of the values Zm to Zm+12 (numerically or graphically) in a video format the table used must be accessible to video.

Worked Example

Requirements

Unramped input	:	Analog input C0.
	:	Range -32768 to +32767.
Required ramped output	:	Analog output D32
	:	Range limited to ± 32000. (corresponding to ± 10V)
Required acceleration time	:	Range to be limited to 10 to 30 seconds
	:	Set in seconds using thumbwheel input A3.
Required deceleration time	:	10 seconds (preset)
Required rounding	:	0.5 seconds time constant.

Logic Inputs :-

STOP pushbutton	:	A1.0 (normally ON, press for OFF)
RUN pushbutton	:	A1.12 (normally OFF, press for ON)
HOLD pushbutton	:	A1.13 (normally OFF, press for ON)
RAISE pushbutton	:	A1.14 (normally OFF, press for ON)
LOWER pushbutton	:	A1.15 (normally OFF, press for ON)

Note...RAISE and LOWER to be effective only after HOLD has been pressed.

Data preset by User

Table	Value	Remarks
LIMIT function in Rung 3		
W98	10	Low limit for acceleration time.
W99	30	High limit for acceleration time.
RAMPGEN function in Rung 8		
W101.0	ON	Suicide logic (permanently released).
W103	1000	Deceleration time in 0.01s units.
W104	50	Rounding time constant in 0.01s units.
W106	+32000	Positive limit.
W107	-32000	Negative limit.
W108	0	Current limit input (not used)

Rung Comments

Rung 1 is a conventional run/stop rung.

Rung 2 selects Hold as long as Run is not selected. Also; Hold is selected via the initialize contact INIT, on the first program execution.

Rung 3 takes the thumbwheel switch input giving the acceleration time in seconds, converts it from BCD, and limits the number to within the range 10 to 30.

Rung 4 sets T_{ACC} by multiplying the acceleration time by 100 to turn it into 0.01s units.

Rung 5 sets the Hold logic output to ON (for normal running) via the INV instruction. The Hold logic output will get set to OFF (to hold the RAMPGEN output) when the Hold push button is operated (from Rung 2) while neither the Raise or Lower pushbutton is pressed.

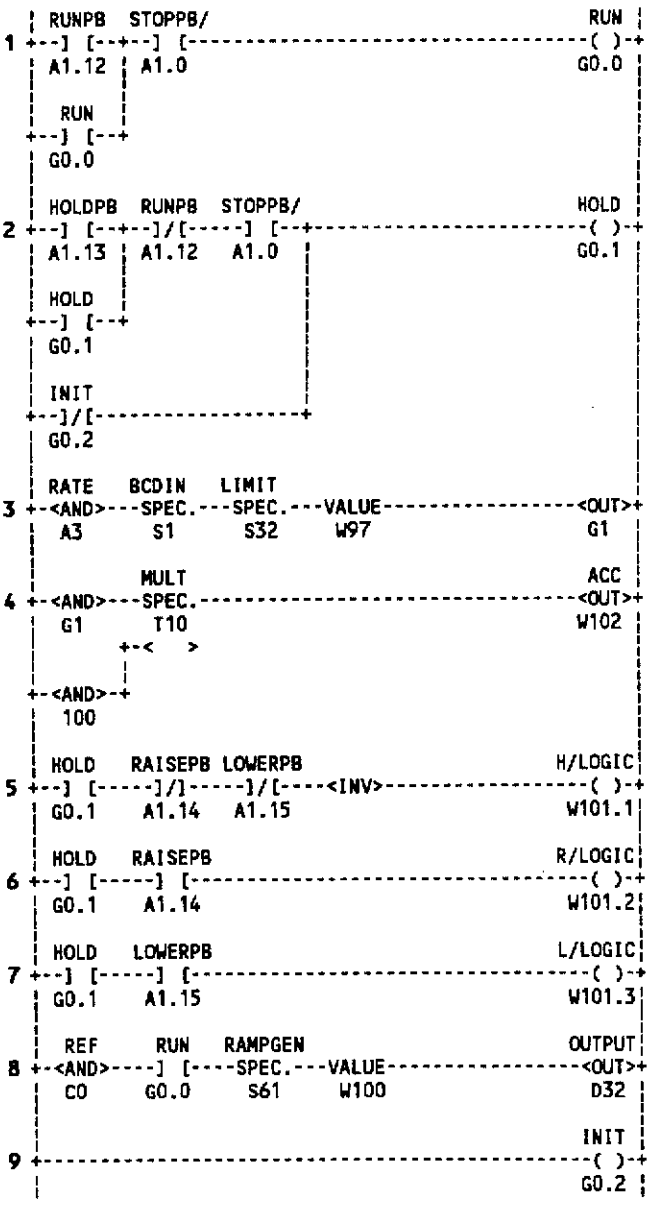
Rung 6 selects Raise or Rung 7 selects Lower, as long as Hold is also selected via Rung 2.

Note...Immediately that the Raise or Lower push buttons are released, the RAMPGEN output will be held via Rung 5.

Rung 8 is the actual RAMPGEN control rung. When the Controller is first run, there is no movement from the rung output throghout the time of the first time reading period because the function is held via Rung 2 and Rung 5.

Rung 9 energizes the INIT coil G0.2 so that on the next and subsequent executions RAMPGEN is allowed to run normally.

Program Rungs



INDUSTRIAL CONTROLS

Kidsgrove, Stoke-on-Trent, ST7 1TW, England
Tel: (0782) 783511 Fax: (0782) 776329 Telex: 36293/4 CICKID G
Registered in England No. 2259095

T1314 Issue 1