Software Data S33 - RAMP

T314 Issue 9 09/89

Please incorporate the following amendments:-

Page 17. Table 2

Under 'Invalid rounding at start of ramp.

delete 32000

insert 16000

Please retain this Errata Sheet with the document for reference.

Software Data Sheet S33 - RAMP

T314 Issue 9 09/89

Please incorporate the following amendment:-

Page 17, Table 2, Invalid negative limit

QNEG <0 or>-32000

delete < and >
insert > and <

Please retain this Errata Sheet with the document for reference.

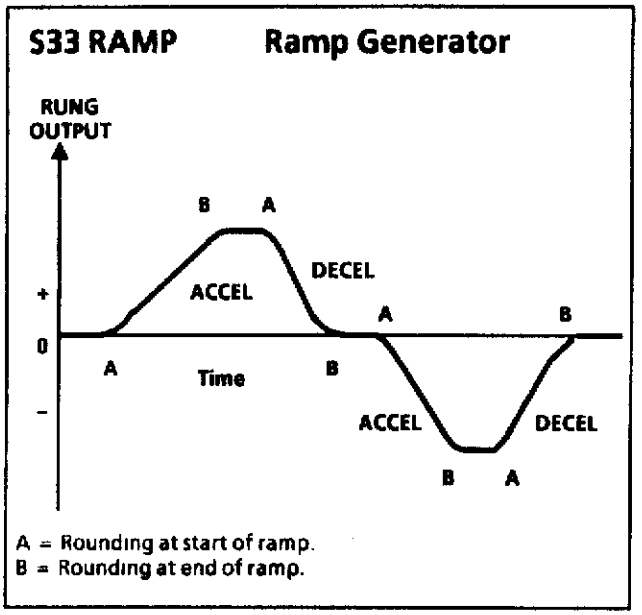
Ramp Generator Functions

RAMP S33

DESCRIPTION

The RAMP function is intended for smoothing out rapid changes of setpoint. 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 changes.

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



FEATURES

- All setpoint changes smooth and jerk-free due to rate limits and rounding.
- Independent rates for acceleration (setpoint moving away from zero) and deceleration (setpoint moving towards zero).
- Ramp output normally follows setpoint input, 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.

  Rate output (e.g. for use in feed-forward anticipatory controls).

SPECIFICATION

Control language:

```
          RAMP
X -- SPEC. -- VALUE -- Q
          S33
```

Quantity of table locations : 15

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 Application Notes overleaf.

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, ramp will round off and the output Q will then 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.

APPLICATION NOTES

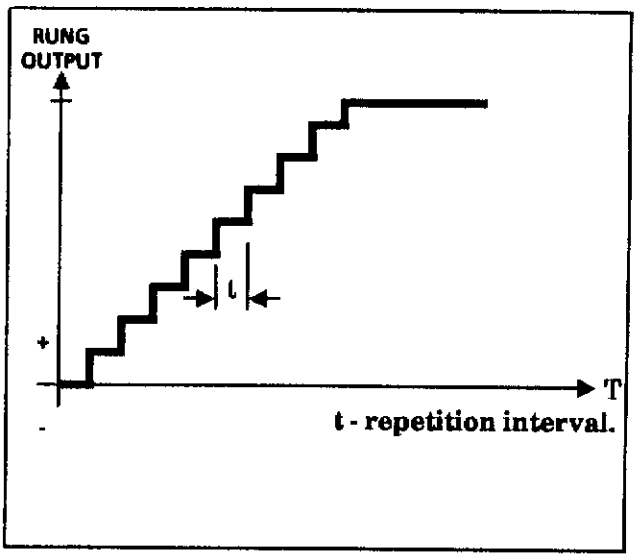
Contents

1. RAMP RATE

Although the RAMP Special Function has facilities for rounding off the start and end of the ramp, we will initially ignore this and consider a ramp with no rounding.

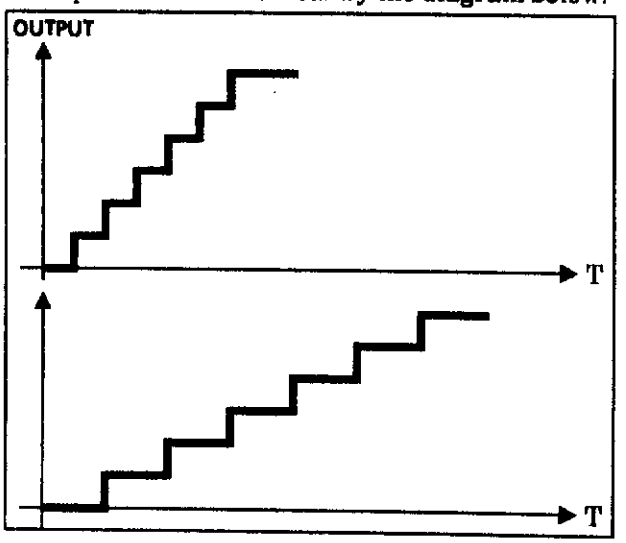
While there is no change in rung input to the RAMP function, the rung output from the RAMP is exactly equal to the rung input; that is, a RAMP function has unity gain. The rung output from the RAMP function also equals the rung input to it if the rung input changes slowly.

When there is a more rapid change to the rung input, the output from the RAMP is rate limited. It increases or decreases by a preset number on each subsequent program execution, until the rung output is again equal to the rung input. Consequently, the rung output when plotted as a graph is a staircase, as shown in the following diagram:



(Note that the last step of the ramp will often be less than the rest of the steps in the ramp, to finally bring the rung output exactly equal to the rung input.)

The slope of the staircase is therefore dependent on the program repetition interval. If you ever alter the program repetition interval, then the ramp rate will change, unless you make a corresponding alteration to the RAMP table values which preset the height of the steps. This is made clear by the diagram below:



For use with analog control, you normally want the steps in the staircase to be as small as possible, to provide the smoothest possible ramp. For this reason, you should normally operate RAMP functions on every program execution. Don't place them in BLOCKs which are operated less frequently unless you require a very slow ramp rate. This will become clear when we work out some example figures for setting up the ramp rates.

Further, for the staircase produced by a RAMP to be regular, the program executions must occur at regular intervals. You should therefore set up your GEM 80 controller with an appropriate value for the preset table value P1. If you set P1 too low, or if you set no value in P1 at all, then the controller will be forced to free run, and ramps will not be of a consistent or regular rate.

The RAMP function has two table values, Zm + 1 and Zm + 2, for setting the ramp rates. The first, referred to as the "Acceleration rate" value, sets the ramp rate when the rung output is moving away from zero. The second, referred to as the "Deceleration rate" value, sets the ramp rate when the rung output is moving towards zero.

To work out these values, you must have the following data available:

- The maximum output which you want from the RAMP.
- How many seconds you want the ramp to take to get there from zero (for acceleration).
- How many seconds you want the ramp to take to get from the maximum output back to zero (for deceleration).
- What the program repetition interval is.

Example:

Maximum output = 32000
Acceleration time from zero to max. output = 30s
Deceleration time from max. output to zero = 10s
Program repetition interval = 20ms

Then, for acceleration:

$$\text{Number of steps in the ramp} = \frac{\text{Acc.time}}{\text{Rep.intvl}} = \frac{30\ \text{s}}{20\text{ms}} = 1500$$

Increase in output per program execution:

$$\text{Step size} = \frac{\text{Max.output}}{\text{No. of steps}} = \frac{32000}{1500} = 21.33$$

Value to be entered in Zm + 1:

ACC = Step size x 50
= 21.33 x 50
= 1066.6
= 1067 to the nearest integer

Similarly, for deceleration:

$$\text{Number of steps in the ramp} = \frac{\text{Dec. time}}{\text{Rep.intvl}} = \frac{10\ \text{s}}{20\text{ms}} = 500$$

Decrease in output per program execution:

$$\text{Step size} = \frac{\text{Max.output}}{\text{No. of steps}} = \frac{32000}{500} = 64.00$$

Value to be entered in Zm + 2:

Dec = Step size x 50
= 64.00 x 50
= 3200

Note the following points:

1. The maximum permitted values of Zm + 1 and Zm + 2 are 16000, and the values must always be positive. Numbers outside this range will cause the RAMP output to be set to zero, and a fault code will be given in table location Zm (See Table 2).

2. Where the step size does not work out to an integer, then it will vary slightly so as to give the correct mean step size. For instance, in the example above for acceleration, steps will be either 21 or 22 but arranged to give the required mean of 21.33.

3. The resolution of most GEM 80 analog output modules is 1 in 2047. For modules with a maximum input of 32752, for instance, a 1-bit change in DAC (digital-to-analog convertor) output would be caused by a number change of 16. If you were using such a module on the output of the RAMP in the above example, then the DAC output would change on each program execution by:

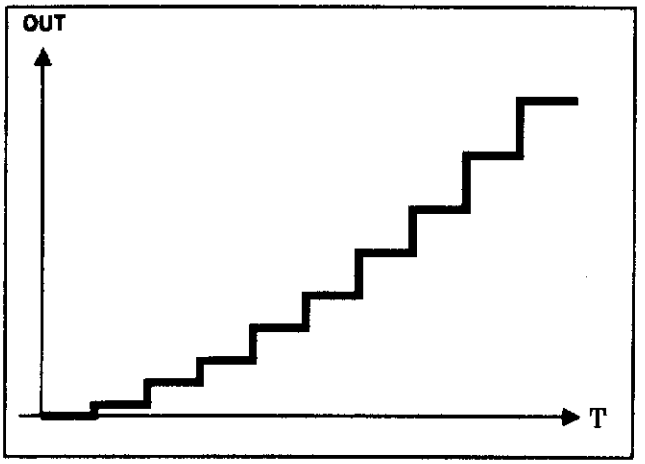
   on acceleration, 21.33/16, i.e. by 1 or 2 steps
   on deceleration, 64/16, i.e. 4 steps.

   The only way of improving the resolution, for the given ramp rate, would be to use a shorter program repetition interval.

2. ROUNDING AT START OF RAMP

For a simple ramp, the output from the RAMP is a staircase where the step size (give or take 1 bit) is the same for every program execution.

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. You can imagine this like the start of an escalator when stationary, as shown in the following diagram:



The RAMP function provides for this is by making the step size increase by a certain amount from what it was on the previous program execution. It carries on doing this until the full step size required for acceleration or deceleration (as we calculated previously) is reached, after which the step size remains constant.

For instance, suppose the increase in step size were made equal to 5. Then we would obtain the following sequence of output values from the RAMP, assuming that its output started from zero:

Program cycle	Step Size	RAMP output
1	5	5
2	10	15
3	15	30
4	20	50
5	25	75
6	30	105
7	35	140

and so on. After a few program cycles, however, the step size would become larger than that required for steady ramping. The rounding will then come to an end, and the step size will thereafter remain constant.

The increase in step size per program cycle is set by the value held in table location Zm+3. The same value is used both at the start of a ramp accelerating in a direction away from zero, and at the start of a ramp decelerating in a direction towards zero. This gives the same sharpness of curvature for both cases.

Where acceleration and deceleration rates are different, the length of the time during which rounding occurs are therefore different, and in proportion to these rates. Thus, for our previous worked example, the deceleration rate was 3 times faster than the acceleration rate, therefore the rounding will last 3 times longer at the start of decelerating than at the start of accelerating. This makes sure that the curvature for both conditions is the same.

Example:

Using the same example as we used for ramp rates, let us now add rounding at the start of the ramp, which is to last for 1 second at the start of deceleration.

(Therefore, since deceleration in this example is 3 times faster than acceleration, rounding will last for 1/3rd of a second on acceleration.)

$$\text{Number of steps during rounding} = \frac{\text{Rounding time}}{\text{Rep.intvl}} = \frac{1\ \text{s}}{20\text{ms}} = 50$$

$$\text{Step size at end of rounding (from previous calculation)} = 64$$

$$\text{Step size increase per cycle} = \frac{\text{Final step size}}{\text{No. of steps}} = \frac{64}{50} = 1.28$$

Value to be entered in Zm+3:

$$KA = \text{Step size increase} \times 50 = 1.28 \times 50 = 64$$

Note the following points:

1. The maximum value permitted for Zm+3 is 16000, and the value must always be positive. Values outside this range will cause the RAMP output to be set to zero, and a fault code will be given in table location Zm (See Table 2).

2. If no value is entered in Zm+3, then by default Zm+3 will contain zero, and the ramp output will not respond to the rung input.

3. If you require no rounding, then the value in Zm+3 must be larger than both the values held in Zm+1 and Zm+2.

3. ROUNDING AT END OF RAMP

Rounding at the end of a ramp, when required, is provided by the RAMP function in a slightly different way to rounding at the start of the ramp. Instead of the step size reducing by an equal amount per program cycle, it tails off exponentially. The response is like the time constant provided by the S37 ANALAG Special Function.

To achieve curvature at the end of a ramp which is similar to that at the start of the ramp, you need to know how far the ramp has got at the start of ramping when it stops rounding. This is given by taking the faster of the acceleration and deceleration ramp rates (if different) for half the rounding time.

Thus, for the previous example:

$$\text{Rounding time} = 1\text{s}$$

$$\text{Deceleration rate} = \frac{\text{Max. output}}{\text{Dec. time}} = \frac{32000}{10\text{s}} = 3200 \text{ per second}$$

When rounding at the start of the ramp finishes, the ramp output will have reached:

$$\text{Output} = 3200 \times \frac{1\text{s}}{2} = 1600$$

To achieve a similar degree of rounding at the end of the ramp, rounding at the end of the ramp should therefore start, in this example, 1600 away from the end of the ramp.

Value to be entered in Zm + 4:

$$KB = \frac{DEC}{e}$$

where DEC = Zm + 2 value calculated previously

e = distance from end of ramp as calculated above.

Hence:

$$KB = \frac{3200}{1600} = 2$$

Note that, on acceleration, the distance away from the end of the ramp when rounding starts will be reduced in proportion to the ramp rate. If, as in the example, the acceleration rate is three times slower than the deceleration rate, then the distance away from the end of the ramp when rounding starts will be one third, or about 533.

Note the following points:

1. The maximum value permitted for Zm + 4 is 50, and the value must always be positive. Values outside this range will cause the RAMP output to be set to zero, and a fault code will be given in table location Zm (See Table 2).

2. If no value is entered in Zm + 4, then by default Zm + 4 will contain zero, and the ramp output will not respond to the rung input.

3. If no rounding is required, then the value in Zm + 4 should be set to the maximum permitted value of 50. This value will still give a very small amount of rounding; you cannot eliminate end-of-ramp rounding entirely.

4. The equivalent time constant during rounding at the end of the ramp is given by:

$$T_c = \frac{2}{KB} \text{ seconds}$$

That is, the closer you make the rounding to the end of the ramp, the shorter the equivalent time constant.

4. QUICK CALCULATION FORMULAE

In the above notes on ramp rates and rounding, we have given the calulations in step-by-step form for each of the values Zm + 1 to Zm + 4, as we think this gives you a better idea of how the RAMP function works. However, these calculations can be reduced to the formulae given below:

Let
- T = program repetition interval
- QMAX = full output
- t_a = acceleration time, zero to QMAX
- t_d = deceleration time, QMAX to zero
- t_r = rounding time

Then:

$Zm+1$: $ACC = QMAX \frac{50\ T}{t_a}$

$Zm+2$: $DEC = QMAX \frac{50\ T}{t_d}$

$Zm+3$: $KA = ACC \frac{T}{t_r}$ or $DEC \frac{T}{t_r}$

whichever is the larger.

$Zm+4$: $KB = \frac{2}{t_r}$

where t_r is in seconds.

(The last formula applies where the distance from the end of the ramp when rounding comes in is equal to the distance from the start of the ramp when rounding ceases.)

5. LOGIC INPUTS

5.1 Suicide

For normal running, you need to set the Suicide input Zm+7 to be ON (i.e. a non-zero input).

If you set Zm+7 OFF (i.e. zero input), the RAMP output switches immediately 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 signal inputs.

5.2 Hold

For normal running, you need to set the Hold input ZM+8 to be ON (i.e. a non-zero input).

If you set ZM+8 OFF (i.e. zero input), the RAMP output freezes. It will remain at whatever value it has reached, regardless of any subsequent change of the rung input to the RAMP, until such time as Zm+8 is again switched ON.

If the RAMP output is ramping at the instant that you set Zm+8 OFF, the output will not freeze instantly, but will be subject to rounding. It will therefore carry on moving by a small number beyond the value it was at when Zm+8 was set OFF. The shape of the rounding is determined by KA, the start-of-ramp rounding constant in Zm+3, and not the end-of-ramp value in Zm+4.

Hold overrides all other logic signal inputs except Suicide.

5.3 Raise

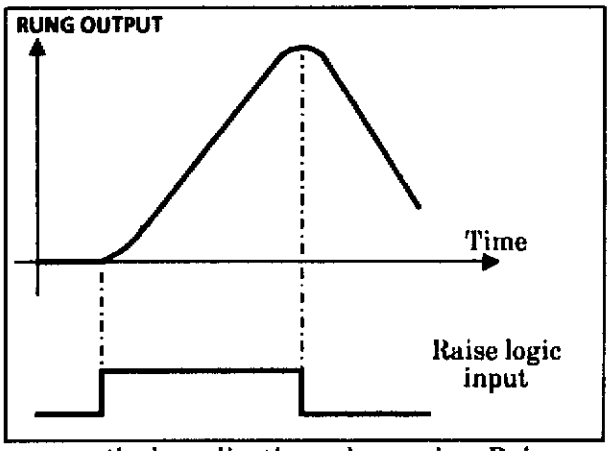
For normal running, you need to set the Raise input Zm+9 to be OFF (i.e. zero input).

If you set Zm+9 ON (i.e. a non-zero input), then the RAMP output will increase (i.e. move away from zero) at the acceleration rate limit ACC set by ZM+1. It will carry on increasing so long as Zm+9 is ON, or until the output reaches limit. When you set Zm+9 OFF again, then the RAMP output will decrease until it again becomes equal to the rung input to the RAMP function.

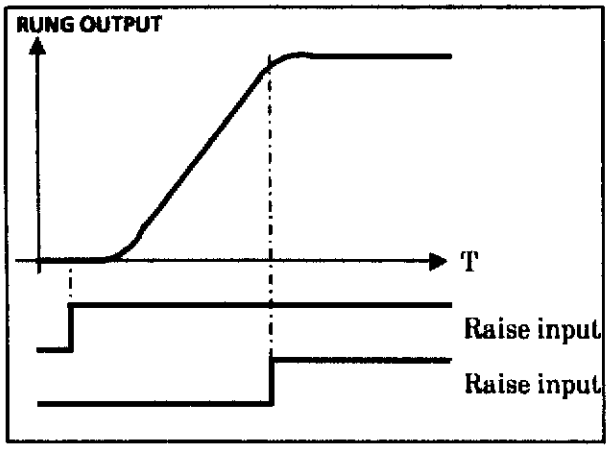
If, at the time that the Raise input is applied by setting Zm+9 ON, the rung output from the RAMP function is positive, then the RAMP output will move in a positive going direction; if the rung output is initially negative, then the output will move in a negative going direction. For the case where the rung output from the RAMP function is exactly zero, the RAMP output will move in a positive going direction.

Rounding occurs at the start of ramping when Raise is applied. If the RAMP output reaches limit, no rounding occurs as limit is reached.

When Raise is removed, the RAMP output does not round off at the end of the ramping action it has just done, but starts rounding from zero slope as on the start of ramp. See the diagram overleaf:



In a practical application when using Raise, you would normally apply the Hold input at the end of a Raise. In this case, you then get end-of-ramp rounding as below:



Raise is overridden by all other logic inputs.

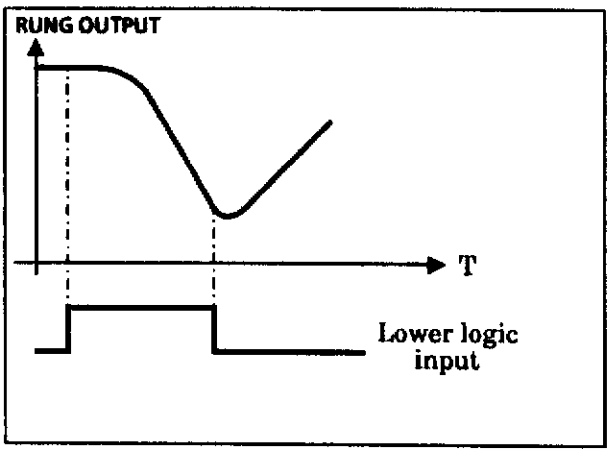
5.4 Lower

For normal running , you need to set the Lower input Zm + 10 to be OFF (i.e. zero input).

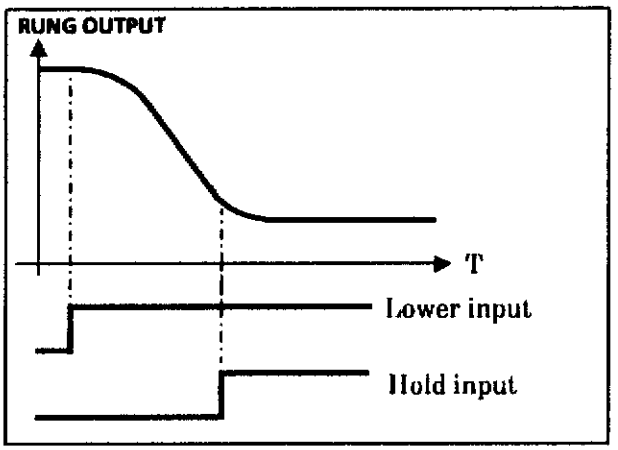
If you set Zm + 10 ON. (i.e. a non-zero input), then the RAMP output will decrease (i.e. move towards zero) at the deceleration rate limit DEC set by ZM + 2. It will carry on decreasing so long as ZM + 10 is ON, or until the output reaches zero. If it reaches zero, it stops there and does not continue ramping in the opposite direction. When you set Zm + 10 OFF again, then the RAMP output will increase until it again becomes equal to the rung input to the RAMP function.

Rounding occurs at the start of ramping when Lower is applied, If the RAMP output reaches zero, no rounding occurs as zero is reached.

When Lower is removed, the RAMP output does not round off at the end of the ramping action it has just done, but starts rounding from zero slope as on the start of ramp. See the diagram below:



In a practical application when using Lower, you would normally apply the Hold input at the end of a Lower. In this case, you then get end-of-ramp rounding as below:



Lower overrides Raise, but is overridden by Hold and Suicide.

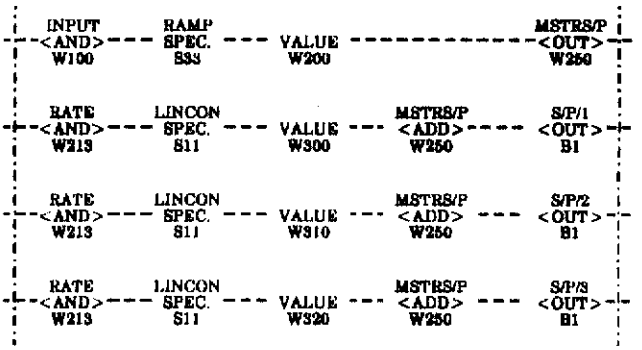
6 COMMON SET POINTS

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

6.1 Rate Output

For such an application, you cannot guarantee that the speed of response of all drive systems will be identical. If the speed setpoint comes from a RAMP, the true speed of each drive will also ramp up, but lagging behind the setpoint by a certain time, dependent on the response of the particular drive.

To compensate for these different time lags, you can modify the individual setpoints by adding to them a certain amount of rate-of-change of setpoint. To cater for this, the RAMP function provides a rate-of-change output, held in table location Zm+13. You might then use rungs in your program such as those shown:



In the above example, the rate output from the RAMP will be in location Zm+13 = W(200+13) = W213. 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, S/P/2 etc.).

Note that the value held in Zm+13 is the current ramp step size multiplied by 50.

6.2 Current Limit Input

For maximum throughput through a rolling mill, you need to accelerate the mill up to the designed operating speed as quickly as possible. You therefore need the ramp rate for the common setpoint to be as fast as possible.

However, as you wind up the ramp rate setting, one of the drives will hit current or torque limit. Which particular drive this is will vary depending on the rolling profile set for the mill. This drive will then fail to follow the setpoint, lagging increasingly behind it. The material being rolled will either break or throw a loop.

To cater for this condition, the RAMP Special Function has "Current Limit" table location Zm+11. If you place a value in this table location, you can increase or decrease the ramp rate. You cannot, however, make the ramp rate faster than the acceleration and decleration rates set by Zm+1 and Zm+2 as the RAMP function will limit it.

To decrease the ramp rate during acceleration, you must set Zm + 11:

- positive while the RAMP output is positive.
- negative while the RAMP output is negative.

To use this facility, you will need an input signal to your GEM 80 controller from each drive, indicating when it is a about to go into limit. These signals must be analog or numeric inputs, not ON-OFF logic inputs. You should arrange for these signals to be zero for normal running, and proportional to the amount by which they exceed a threshold level. You can then use the amount by which the signal exceeds the threshold to determine by how much the ramp rate must be reduced. Obviously, you need to set the threshold slightly below the current or torque limit setting of the drive.

Next, you need to OR-gate the signals from the various drives, so that the signal used to slow down the RAMP is the largest one (i.e. coming from the drive which is operating nearest to current limit).

Note that, when you use signals as described above to slow down the RAMP, you have built yourself a closed loop system. That is, when a signal slows down the RAMP, this will affect the accelerating current through the motors,which will reduce the signal slowing down the RAMP.

If your closed loop system is stable, then it should settle down to a balanced condition. If, however, the gain round the loop is too high, the RAMP rate could oscillate up and down about the required ramp rate. You therefore need to check the design of this loop. You can reduce the gain round the loop if you drop the threshold level (i.e. you use a bigger change in current to give the same change in ramp rate). You can also improve stability by making the end-of-ramp rounding start further away from the end of the ramp, thus making the equivalent time constant longer (see Section 3).

As with any closed loop system, make sure that your signal polarities are correct for all conditions (acceleration, deceleration, positive output and negative output). Otherwise, you will get positive feedback.

Check also that, except when the RAMP rate is to be modified, Zm + 11 contains zero.

Note that an input to ZM + 11 can cause the RAMP output to be driven up or down beyond the level set by the rung input to the RAMP. The Zm + 11 input cannot, however, make the RAMP output exceed the limit levels set by ZM + 5 and Zm + 6, or make the ramp rates exceed the values set by ZM + 1 and Zm + 2.

7. BUMPLESS TRANSFER

In many process control systems, you may require to change from, say, AUTO to MANUAL control without any bump. You can do this with RAMP by setting up the value of the previous output table location Zm + 12 immediately prior to changing over. The RAMP output will then ramp up or down as required from this setting, and you will avoid any step change in signal.

8. CHOICE OF DATA TABLE

The following factors determine the choice of which table to use for the values Zm to Zm + 14:

1. Whether the table is write-protected.

   The table you use for your value data must not be write-protected. If it were, RAMP would not be able to store the values of Q_{n-1}, Q' and R required for the next calculation on the next scan.

2. Whether table contents are cleared on power up.

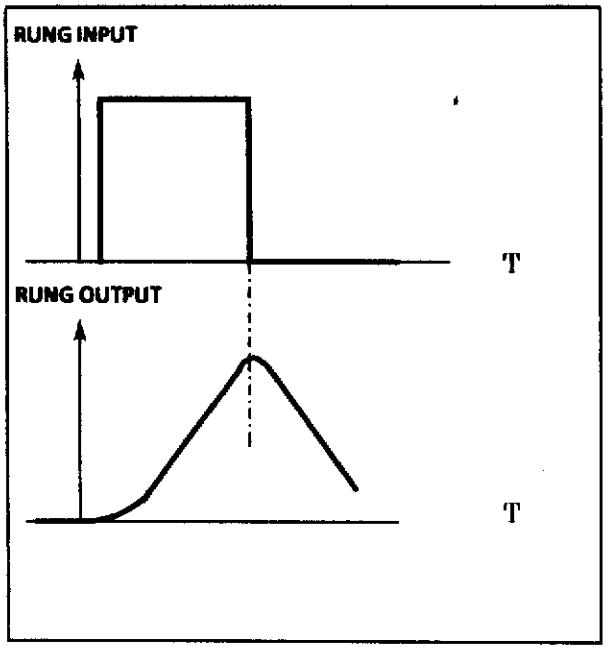
You may want the values of Q_{n-1}, Q' and R retained while the controller is powered down for some applications, whereas, for other applications, this may be unnecessary.

3. Whether table contents are accessible by video. (This only applies to controllers which include video).

Refer to the Technical Manual for the particular model of controller to find the attributes of the various tables included in it.

9. A Note about Rounding

If the rung input to a RAMP Special Function is changed while the function is ramping, and if the change will cause the RAMP output to reverse its direction of ramping, then some rounding will be lost, as shown below:



The RAMP Special Function was designed this way so that the rung output will never overshoot the rung input. However, if your application can stand some overshoot but can't stand loss of rounding, then you can smooth out the corners of the RAMP by using the ANALAG Special Function S37 set to a short time constant (i.e. short compared with the ramp time).

If you are also using the rate output (see Section 6.1), then you should provide an ANALAG Special Function to smooth out this output as well, using identical ANALAG table values. In this way, the rate output will still be the true derivative of the main RAMP output.

If, in addition, you are using the current limit input, then you will need to check the effect of any ANALAG on the stability of any ramp rate reduction closed loop (see Section 6.2).

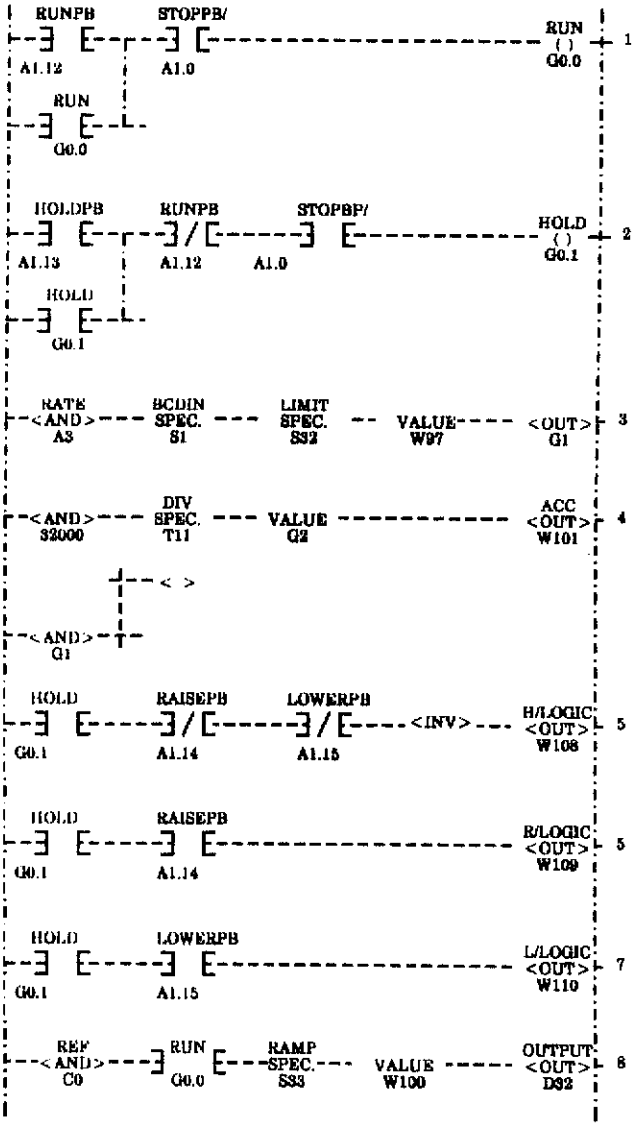
10. WORKED EXAMPLE

10.1 Requirements

Unramped input	:	Analog input C0.
	:	Range -32768 to +32752
Program repetition interval	:	20 ms
Required ramped output	:	Analog output D32.
	:	Range limited to ±3200 (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 at start of ramp	:	1 second from zero to full ramp rate (i.e. 50 scans of program).
Required rounding at end of ramp	:	To begin 1 second from the end of the ramp.
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).
	:	RAISE and LOWER to be effective only after HOLD has been pressed.

10.2 Program rungs



Rung 1 is a conventional run/stop rung.

Rung 2 selects Hold as long as Run is not selected.

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 the acceleration constant ACC by dividing 32000 by the acceleration time (see calculations below).

Rung 5 sets the Hold logic output ON (for normal running) via the INV instruction. The Hold logic output will get set OFF (to hold the RAMP output) when the Hold Pushbutton 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 has also been selected via Rung 2. Note that, immediately either the Raise or Lower Pushbutton is released, the RAMP output will be held via Rung 5.

10.3 Calculations of Value Data

Zm+1: (W101)

$$ACC = QMAX \frac{50\,T}{t_a} = 32000 \times \frac{50 \times 0.02}{t_a} = \frac{32000}{t_a}$$

The above formula is implemented by Rung 4.

Zm+2: (W102)

$$DEC = QMAX \frac{50\,T}{t_d} = 32000 \times \frac{50 \times 0.02}{10} = 3200$$

Zm+3: (W103)

$$KA = ACC \frac{T}{t_r} \text{ or } DEC \frac{T}{t_r}$$

whichever is the larger.

In this instance, ACC is less than or equal to DEC; hence:

$$KA = 3200 \times \frac{0.02}{1} = 64$$

Zm+4: (W104)

$$KB = \frac{2}{t_r}$$

where t_r is in seconds.

$$= \frac{2}{1} = 2$$

10.4 Data preset by User

Table	Value	Remarks
Controller:		
P1	20	20 ms program repetition interval
LIMIT function in Rung 3:		
W98	10	Low limit for acceleration time
W99	30	High limit for acceleration time
RAMP function in Rung 8:		
W102	3200	Constant for deceleration time DEC
W103	64	Rounding at start of ramp KA
W104	2	Rounding at end of ramp KE
W105	+32000	Positive Limit
W106	-32000	Negative Limit
W107	-1	Suicide logic (permanently released)
W111	0	Current limit input (not used)

Table 1 - Program rung and Value data			
Location	Symbol	Use	Range
Zm	F.	Fault code and flags. Set by Ramp.	See Table 2
Zm +1	ACC	Acceleration rate . Set by user.	0 to +16000
Zm +2	DEC	Deceleration rate Set by user.	0 to +16000
Zm +3	KA	Rounding at start of ramping. Set by user.	0 to +16000
Zm +4	KB	Rounding at end of ramping. Set by user.	0 to +50
Zm +5	QPOS	Positive output limit. Set by user.	0 to +32000
Zm +6	QNEG	Negative output limit. Set by user.	0 to -32000
Zm +7	S	Suicide logic input. Set by user, either preset to suicide off, or else controlled by separate rung	0: Suicide on; Q reduces to zero instant-aneously. Any non-zero value: Suicide off (normal operation).
Zm +8	H	Hold logic input. Set by user, either preset to hold off, or else controlled by separate rung.	0: Hold on; ramp rate reduces to zero,after which Q stays constant.
Zm +8 Con'd			Any non-zero value: Hold off (normal operation).
Zm +9	R	Raise logic input. Set by user,either preset to Raise off, or else controlled by separate rung.	0: Raise off (normal operation). Any non-zero value: Q ramps away from zero.
Zm +10	L	Lower logic input. Set by user,either preset to Lower off, or else controlled by separate rung.	0: Lower off (normal operation). Any non-zero value: Q ramps towards zero.
Zm +11	I	Current limit input. Set by user using separate rungs, or else defaults to 0.	±32767 Normally zero. Positive value reduces accel. when Q pos. Negative value reduces accel. when Q neg.

Table 1 - Program rung and Value data			
Location	Symbol	Use	Range
Zm +12	Q_{n-1}	Previous value of output. Stored by RAMP. Used in next scan.	May be over-written by user, e.g. for bumpless transfer.
Zm +13	Q'	Rate output. Stored by RAMP. Used in next scan.	Can be used as output for feedforward.
Zm +14	Q_r	Rate remainder. Stored by RAMP. Used on next scan.	
-	X	Input to RAMP from rung.	± 32767
-	Q	Output from RAMP.	± 32767, but subject to QPOS and QNEG limits.

Table 2 - Fault conditions and flags			
Condition	Code in Zm		
	Result	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			
Invalid accel. rate ACC<0 or > 16000	Q=0	5	Zm.0 = 1 Zm.2 = 1
Invalid decel. rate DEC<0 or > 16000	Q=0	9	Zm.0 = 1 Zm.3 = 1
Invalid rounding at start of ramp. KA < 0 or > 32000	Q=0	17	Zm.0 = 1 Zm.4 = 1
Invalid rounding at end of ramp. KB < 0 or > 50	Q=0	33	Zm.0 = 1 Zm.5 = 1
Invalid positive limit QPOS < 0 or > 32000	Q=0	65	Zm.0 = 1 Zm.6 = 1
Invalid negative limit QNEG < 0 or > -32000	Q=0	129	Zm.0 = 1 Zm.7 = 1
STATUS			
Output limited at positive limit	Q=QPOS	-28672	Zm.12 = 1 Zm.15 = 1
Output limited at negative limit	Q=QNEG	+20480	Zm.12 = 1 Zm.14 = 1

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