Changing Motions in REPEAT Queues

Abstract

MotionBASIC®’s REPEAT queues provide a powerful tool for implementing complex sequences of repeated motion. This Tech Note shows you how you can replace entries in a motion queue without interrupting the current motion sequence.

Description

A REPEAT queue is a list of motion statements that have been passed on to the axis DSP processor to be executed independent of your MotionBASIC® program. For example:

```
distance=200    'set the distance
REPEAT GEAR axis FOR distance IN 200 AFTER ASEN@ 'motion 0
REPEAT GEAR axis FOR 0 IN 1000  'motion 1
REPEAT GEAR axis FOR -distance/2 IN 100 'motion 2
REPEAT GEAR axis FOR 0 IN 1000  'motion 3
REPEAT GEAR axis FOR -distance/2 IN 100 'motion 4
```

The axis DSP processor will execute these five motion statements, one after the other. When the last statement has been executed, it will go back and start at the beginning of the queue again. Since this is done by the axis DSP processor, your MotionBASIC® program is free to go on and work on other tasks.

Suppose however, you need to change the value of the distance variable without interrupting the sequence. Further, suppose your process requires you to change it after motion 1 is completed so that for one cycle, motion 0 uses the old distance value and motions 2 and 4 use the new distance value. Subsequent, repeated motions 0, 2 and 4 use the new distance value. How can you accomplish this?

Solution

Either GEAR FOR 0 IN 0 or MOVE FOR 0 IN 0 allow you to abort a repeated motion sequence at the end of the current queue. The rest of this Tech Note shows you how to use one of these statements to solve the stated requirement.

Implementation

```
WAIT UNTIL GEAR.RATIO@(axis)>0       'wait for queue to start
GEAR axis FOR 0 IN 0 'abort current queue after completion of motion 4
GEAR axis FOR distance IN 200 AFTER ASEN@ 'old nonrepeat motion 0
distance=100              'change the distance
GEAR axis FOR 0 IN 1000  'new nonrepeat motion 1
GEAR axis FOR -distance/2 IN 100 'new nonrepeat motion 2
GEAR axis FOR 0 IN 1000  'new nonrepeat motion 3
GEAR axis FOR -distance/2 IN 100 'new nonrepeat motion 4
REPEAT GEAR axis FOR distance IN 200 AFTER ASEN@ 'new repeat motion 0
REPEAT GEAR axis FOR 0 IN 1000  'new repeat motion 1
REPEAT GEAR axis FOR -distance/2 IN 100 'new repeat motion 2
REPEAT GEAR axis FOR 0 IN 1000  'new repeat motion 3
REPEAT GEAR axis FOR -distance/2 IN 100 'new repeat motion 4
```

First, wait until the queue has started executing the first motion. We want to do this because if we abort the queue near the very end of motion 4 we may not rebuild the queue fast enough to avoid a small amount of time where no gear is occurring. In this example we can detect this...
state by looking at the variable GEAR.RATIO@. Motion 0 is the only positive motion and therefore the only motion for which GEAR.RATIO@ will be positive and non-zero. Another way to detect which motion is currently executing is via the variable DSP.CTR@ as described in Tech Note #6.

Next, use the GEAR FOR 0 IN 0 command to abort the current queue when it next completes motion 4. Then start rebuilding the queue with non-repeated GEARs changing the value of distance before the appropriate GEAR statement.

Once you have completely rebuilt the queue using non-repeat GEAR statements, rebuild it again using REPEAT GEARs. This approach keeps motion 4 at the end of the REPEAT queue just as it was in the original queue. By keeping the queue in the same order as the original, you can use the same code to change distance as many times as you want.

If you don’t need to keep motion 4 at the end of the queue, you can use fewer statements to effect the change. In this example, you start using REPEAT GEAR statements as soon as you have changed the value of distance. But now, motion 1 ends up last in the queue.

```plaintext
WAIT UNTIL GEAR.RATIO@(axis)>0 'wait for queue to start
GEAR axis FOR 0 IN 0 'abort current queue after completion of motion 4
GEAR axis FOR distance IN 200 AFTER ASEN@ 'old nonrepeat motion 0
GEAR axis FOR 0 IN 1000 'old nonrepeat motion 1
distance=100 'change the distance
REPEAT GEAR axis FOR -distance/2 IN 100 'new repeat motion 2
REPEAT GEAR axis FOR 0 IN 1000 'new repeat motion 3
REPEAT GEAR axis FOR -distance/2 IN 100 'new repeat motion 4
REPEAT GEAR axis FOR distance IN 200 AFTER ASEN@ 'new repeat motion 0
REPEAT GEAR axis FOR 0 IN 1000 'new repeat motion 1
```

It is very easy to change the distance after the last motion in the queue, simply issue all new repeat commands with the new distance value immediately after the GEAR FOR 0 IN 0 that terminates the current queue. If you can, it is always best to build your original queue in a way so you can make any changes after the last motion statement.

```plaintext
WAIT UNTIL GEAR.RATIO@(axis)>0 'wait for queue to start
distance=100 'change the distance
GEAR axis FOR 0 in 0 'abort at end of queue
REPEAT GEAR axis FOR distance IN 200 AFTER ASEN@ 'new repeat motion 0
REPEAT GEAR axis FOR 0 IN 1000 'new repeat motion 1
REPEAT GEAR axis FOR -distance/2 IN 100 'new repeat motion 2
REPEAT GEAR axis FOR 0 IN 1000 'new repeat motion 3
REPEAT GEAR axis FOR -distance/2 IN 100 'new repeat motion 4
```

Further Information

For further information refer to the MotionBASIC® Hypertext Manual under MOTION QUEUE and REPEAT.
Customer Support Engineering Tech Note #10

Analog Voltage Proportional to Pacer Velocity

Abstract

This Tech Note shows a way to generate an analog voltage proportional to the velocity of a pacer encoder.

Description

Applications that use a pacer encoder to track material passing through a machine often need to generate an analog voltage proportional to the pacer encoder’s speed. This signal could be used for display purposes or to control other parts of the machine that need to match the pacer encoder speed. This “Tech Note” shows a way to generate such an output with minimal programming.

Solution

Configure the pacer encoder axis as though it is connected to a velocity mode servo drive. and enable the axis in mode 14.

The DSP Velocity Observer converts changes in encoder position to a speed value. Since the "axis" is not being commanded to move, the DSP processor interprets this speed value as a velocity error. The DSP servo algorithms multiply this velocity error by a proportional gain and convert it into an analog voltage.

Implementation

Use MotionPRO’s GEN-III Configuration utility to select the proper encoder and set up the axis for velocity mode. Using the "Motor/Load Parameters" form, position the cursor on the "Motor Model Number" field, then press the enter key. Select the appropriate encoder from the list. Set Drive type to "Velocity" and set "Maximum Motor Speed" to the maximum speed that the encoder will turn. For an EDR-25S/A1500 encoder, the form should look like Figure 1.

![Motor/Load Parameters Form]

```
Axis 1 of Axis Set {1,2 }

Motor Model Number: EDR-25S/A1500  Feedback: INCREMENTAL
Position Transducer Resolution: 6000 cnts/rev  Drive Type: VELOCITY
Motor Torque Peak: 1.00 in-lbs  Rated: 1.00 in-lbs
Drive Input at Max Speed: 10.00 volts
Maximum Motor Speed: 3000 RPM
Motor Inertia: 0.000470 in-lb-sec²
Load Inertia at Motor Shaft: 0.000000 in-lb-sec²

Velocity Gain: 300.00 RPM/volt
Total Machine Inertia: 0.000470 in-lb-sec²
Maximum Motor Acceleration: 338 rev/sec²
```

Figure 1

TN010a -1- March 17, 1993