Abstract

MotionBASIC®'s REPEAT queues provide a powerful tool for implementing complex sequences of repeated motion. This Tech Note shows you how you can keep track of which motion command in a REPEAT queue is currently executing.

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:

```plaintext
REPEAT GEAR AXIS FOR DISTANCE1 IN 200 AFTER ASEN@ 'move 0
REPEAT GEAR AXIS FOR 0 IN 1000 'move 1
REPEAT GEAR AXIS FOR -DISTANCE2 IN 100 'move 2
REPEAT GEAR AXIS FOR 0 IN 1000 'move 3
REPEAT GEAR AXIS FOR DISTANCE3 IN 100 'move 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 have a machine sensor wired to a DIO point which is configured to generate an event. If this event occurs during either move 0, 2 or 4 you want to stop the machine immediately and generate a fault. How can your program tell which motion is executing when the DIO event occurs?

Solution

DSP.CTR@ is an ORMEC variable that counts how many motion statements the axis DSP processor has completed since the variable was last reset. The rest of this Tech Note shows you how to use it to solve this requirement.

Implementation

First you need to configure a DIO EVENT to call a routine in your program whenever the sensor is detected.

```plaintext
' configure the dio event
IO.MODE@(1)="R"
ON EVENT DIO@(1) GOSUB EVENT.HDLR
EVENT DIO@(1) ON
EVENT ON
```

DSP.CTR@ starts at zero and increments each time a motion statement is executed by the DSP. It has a maximum value of 32767 after which it resets back to zero on execution of the next motion statement. To avoid the rollover giving erroneous results you must detect that it has occurred and correct for it. The following section shows how to handle the correction.

Before creating the REPEAT queue, you must preset some variables.
'preset some control variables
CORR = 0 'roll over correction factor
Q.LENGTH = 5 'number of moves going in the queue
DELTA = 32768 MOD Q.LENGTH 'roll over correction adder
DSP.CTR@(AXIS) = 0
OLD.DSP.CTR = 0

'now build your motion queue
REPEAT GEAR AXIS FOR DISTANCE1 IN 200 AFTER ASEN@ 'move 0
REPEAT GEAR AXIS FOR 0 IN 1000 'move 1
REPEAT GEAR AXIS FOR -DISTANCE2 IN 100 'move 2
REPEAT GEAR AXIS FOR 0 IN 1000 'move 3
REPEAT GEAR AXIS FOR DISTANCE3 IN 100 'move 4

Then, any time your program needs to know which motion is currently executing, call the following subroutine:

UPD.PTR:
TMP = DSP.CTR@(AXIS) 'store the current value
IF TMP < OLD.TMP THEN 'have we rolled over?
    CORR = (CORR + DELTA) MOD Q.LENGTH 'if so, new correction factor
ENDIF
OLD.DSP.CTR = TMP 'remember the last DSP.CTR@
Q.PTR = (TMP + CORR) MOD Q.LENGTH 'calculate the queue pointer
RETURN

While the first motion is executing, Q.PTR will equal 0, while the second is executing it will equal 1, and so on.

To maintain validity, UPD.CTR: must be executed at least once every 32768 motion statements. You could do this by adding GOSUB UPD.CTR statements at appropriate places in your program or by using a Timed Interrupt as described in Tech Note #5.

When the machine sensor is tripped, a DIO event occurs and program execution branches to the event handler routine.

EVENT.HDLR:
    UPD.PTR 'see which move is currently executing
    IF (Q.PTR=0 OR Q.PTR=2 OR Q.PTR=4) THEN FAULT@=ON
RETURN

Generating a FAULT@ will drop out the NO FAULT relay which should be tied in to your ESTOP circuitry thereby stopping all motion. You could alternately set a flag in the event routine which is checked by your main program in order to generate the fault.

Further Information

For further information refer to the MotionBASIC® Hypertext Manual under REPEAT and DSP.CTR@.