▶Trio Motion Technology◀

Trio Motion Technology Ltd.
Shannon Way,
Tewkesbury
Glos. GL20 8ND U.K.

Tel: 01684 292333
Fax: 01684 297929
Email: apps@triomotion.com
Web: www.triomotion.com

Application Note

Doc No.: TN20-16
Version: 1.0
Date: 24 February 1999
Subject: Process Buffers in Motion Coordinator series 2.

Moves and Movetype.

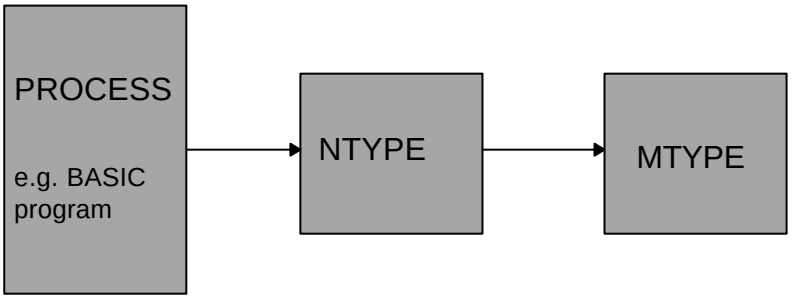
When the BASIC issues a move instruction to the motion system, a MOVETYPE is set in the MTYPE buffer for the axis being acted upon. The MTYPE is actually a code number where 0 = IDLE (or no move in progress) and, for example, 4 = MOVECIRC. For a list of the codes check section 8 of the manual.

So, if a MOVEABS(n) AXIS(2) command is issued, the value 2 appears in the axis 2 MTYPE buffer. When the move is completed, the MTYPE value goes back to 0.

Buffered Moves

If a second move is requested for an axis when a move is already in progress, the move is loaded into a move buffer, called the NTYPE, and held in readiness for the present move to finish.

The instant the move in the MTYPE register is finished, the buffered move is pulled forward and starts in the next servo period.



When both MTYPE and NTYPE registers have a move loaded and another move command is issued, this is sent to the PROCESS BUFFER for that particular process in the multi tasking system. Once the process buffer is has a move in it, no more moves can be buffered so if a process issues another move, it will wait until the move in progress has finished so that the buffered moves can all shuffle forward and leave room in the process buffer.

If the process is a BASIC program, this results in the program waiting at the line containing the move instruction. When a move instruction is entered at the command line on terminal channel 0, this results in the terminal "locking up" until the process buffer becomes free and the move command can thus go on into the NTYPE buffer.

Effect on Program Flow

When 2 or more axes are being controlled by one BASIC program, it is possible for the moves on one axis to hold up the progress of the other axes. For instance if axis 1 has a series of long moves and axis 0 a series of short moves, the following program would cause axis 0 to wait for axis 1, even though this is not obvious at first sight.

```
FOR n=1 TO 50
      MOVE(VR(n)) AXIS(0)
      MOVE(VR(100+n) AXIS(1)
NEXT n
```

The sequence would be as follows:

Command	AXIS 0	AXIS 1
MOVE(VR(1)) AXIS(0)	MOVE(VR(1)) in MTYPE	
MOVE(VR(101)) AXIS(1)		MOVE(VR(101)) in MTYPE (say this is a long move)
MOVE(VR(2)) AXIS(0)	MOVE(VR(2)) in NTYPE	
MOVE(VR(102)) AXIS(1)		MOVE(VR(102)) in NTYPE
MOVE(VR(3)) AXIS(0)	MOVE(VR(3)) in PROCESS BUFFER.	
MOVE(VR(103)) AXIS(1)		waits at this line for MOVE(VR(1)) to finish
MOVE(VR(1)) finishes:	MOVE(VR(2)) in MTYPE MOVE(VR(3)) in NTYPE	MOVE(VR(103)) in PROCESS BUFFER.
MOVE(VR(4)) AXIS(0)		waits at this line for MOVE(VR(101)) to finish
MOVE(VR(2)) finishes:	MOVE(VR(3)) in MTYPE IDLE in NTYPE	still waiting for MOVE(VR(101)) to finish.
MOVE(VR(3)) finishes:	IDLE in MTYPE IDLE in NTYPE	still waiting.
AXIS(0) has now stopped!		
MOVE(VR(101)) finishes:	MOVE(VR(4)) in MTYPE	
MOVE(VR(104)) AXIS(1)		etc…..

Solution

To avoid the moves being processed from holding up the BASIC program, one can test the NTYPE parameter for zero before loading a new move. For example:

```
FOR n=1 TO 50
    IF NTYPE AXIS(0)=0 THEN
        MOVE(VR(n)) AXIS(0)
    ENDIF

    IF NTYPE AXIS(1)=0 THEN
        MOVE(VR(100+n)) AXIS(1)
    ENDIF

NEXT n
```