Stepper Motor Control with a nuLogic pcStep Board

Abstract

This Tech Note describes how to use a nuLogic pcStep board installed in a GEN-III to control up to 3 axes of stepper motors. A MotionBASIC® program called PCSTEPMB.BAS is used to perform the board level communications necessary to operate the pcStep board. This program is available from ORMEC upon request.

Description

The nuLogic pcStep board is a 3 axis stepper controller. Commands are sent from the GEN-III to the card to set values for acceleration, velocity etc. Some of these commands are described in this Tech Note. For a full list of pcStep commands and syntax refer to the pcStep User Manual provided by nuLogic.

The program PCSTEPMB.BAS provides routines for communicating with the pcStep board. The routine STEP.INIT must be executed after powerup before any of the communication routines can be called. The routines SEND.CMD and RECV.CMD are used to send and receive pcStep commands. To include these routines with your application program, use the MotionPRO directive ‘!INCLUDE PCSTEPMB.BAS at the end of your program.

All pcStep commands and data are presented in hexadecimal form. In MotionBASIC® this is represented by an &h prefix. For example the number &h20 represents hex 20 (decimal 32).

Setting PC-Step Board Address

The pcStep Board base address must be configured to match the address specified in the PCSTEPMB.BAS program module. ORMEC recommends using &h3E0 as other addresses may conflict with the GEN3 hardware. See the pcStep board’s documentation on how to set the base I/O address.

Sending commands

The routine SEND.CMD is used to send a command to the pcStep board. Before calling this routine, the following variables must be set properly:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WR.AXIS</td>
<td>The axis to send the command to (1-3)</td>
</tr>
<tr>
<td>WR.CMD</td>
<td>The command to send (&amp;h41-&amp;h6a)</td>
</tr>
<tr>
<td>WR.DATA&amp;</td>
<td>The data for the command</td>
</tr>
</tbody>
</table>

For example, the command to set the velocity is &h4f. To set the velocity on axis 1 to 1000 steps/sec (&h3e8), the following MotionBASIC® statements are needed:

```motionbasic
wr.axis=1
wr.cmd=&h4f
wr.data=&h3e8
send.cmd
```
The command to start the motion is &h61. Since there is no data for this command, the variable WR.DATA& does not need to be set:

```plaintext
wr.axis=1
wr.cmd=&h61
send.cmd
```

Receiving commands

The routine RECV.CMD is used to receive commands from the pcStep board. Before calling this routine, the variable NUM.PACKS must be set to the number of command packets you wish to receive. A command packet consists of the axis that the command came from, the command and the data. This information is placed in the arrays RD.AXIS(), RD.CMD() and RD.DATA&(). The first command received is placed in element zero of these arrays, the second in element one etc.

If fewer command packets are available to be received than is specified in the variable NUM.PACKS, NUM.PACKS will be changed to reflect how many were actually read. If no command packets are ready to be received, an error 103 will be generated. If you want to avoid generating the error 103, the routine DATA.PENDING can be used to determine if the pcStep board has any data for the GEN-III to receive. DATA.PENDING will set the variable RETVAL to 1 if data is available, zero otherwise.

The following MotionBASIC® code will check to see if a command packet is ready to be received and, if so, receive and display it:

```plaintext
data.pending
IF retval=1 THEN
   num.packs=1
   recv.cmd
   PRINT "Axis =";rd.axis(0)
   PRINT "Command =";rd.cmd(0)
   PRINT "Data =";rd.data&(0)
ELSE
   PRINT "No data pending"
ENDIF
```

Error Codes

The following error codes can be generated by the routines SEND.CMD and RECV.CMD:

- 100 Illegal command ID
- 101 Communications error, ready to receive timeout
- 102 Command error on current packet
- 103 No data available in the return data buffer
- 104 Communications error, incomplete returned data packet
- 105 Board failure
- 108 Command error occurred prior to current packet
- 109 Command error, unable to clear command error bit
- 111 Communications error, returned packet ID is corrupted
- 115 Communications error, unable to flush the data buffer
Program Listings

The following program initiates an index and displays actual position while the motor is in motion:

```
test:
  step.init
  wr.axis=1
  wr.cmd=&h64 :wr.data=&hff :send.cmd 'Relative pos mode
  wr.cmd=&h5a :wr.data=&hffffff :send.cmd 'Closed loop mode
  wr.cmd=&h57 :wr.data=&h1f407d0& :send.cmd 'Steps & lines/rev
  wr.cmd=&h4b :wr.data=&h30 :send.cmd 'Enable limits
  wr.cmd=&h4f :wr.data=&h1110 :send.cmd 'Steps/sec
  wr.cmd=&h62 :wr.data=&h100 :send.cmd 'Accel
  wr.cmd=&h4e :wr.data=&h7d0 :send.cmd 'Target pos
  wr.cmd=&h61 :send.cmd 'Start motion

  CLS
  wr.cmd=&h49 :send.cmd 'Read running/stopped status
  num.packs=1 :recv.cmd

  WHILE rd.data&(0) AND &h80 'While running
    wr.cmd=&h53 :send.cmd 'Read position
    num.packs=1 :recv.cmd
    PRINT @1,1,0;"Actual position:";rd.data&(0) 'Display position
  WEND
RETURN

'!INCLUDE pcstepmb.bas
```