Help sheet 18

ST Robotics

Chasing coordinates.

The basic code for this is on the last page. Use Acrobat select tool to copy the text into your project.

The principle of operation is as follows:
The DSP can compute a trajectory for a multi-axis move. It works only with motor counts and only with the change in motor counts, not the actual values.

Normally DSP moves (such as DSPMOVE and CONTINUOUS RUN) subtract the current joint coordinates from the target coordinates and send the difference to the DSP. In a route RUN situation the differences from line to line are computed by the CPU in FORTH and sent to the DSP as fast as possible. The DSP stores them in a buffer. If the buffer is full then the CPU program waits for space to be available. If Cartesian coordinates are used (which they usually are) these are converted to joint coordinates on the fly using TRANSFORM

The DSP uses the whole list of values to compute speeds and ramps for each joint. If this can not be computed (see manual) then a “too tight” error is raised.

The second mode of operation in RUN is TIMED. In this each segment (distance from one line to the next) is executed in a fixed time, the value in SEGTIME in mS. The speed is adjusted by the DSP so that the move is completed in that exact time. A long move requires a higher speed. CHASE is related to that.

A new mode in the DSP allows us to check if the buffer is empty. When we send a move to the DSP it immediately uses them and empties the buffer. We can then send the next move while the DSP is working on the last one. The buffer is now not empty and we must wait for it to be empty before we can send the next. Sequence is therefore:
send move n – DSP starts on them, buffer is empty
send move n+1 – DSP buffers them, buffer not empty
DSP completes move n and starts on n+1, buffer is empty
now we can send move n+2
and so on.

So we can never have more than one move in hand. If you are using a PC program to send these coordinates then it will need to know when the controller is ready for the next coordinates.

In the program CHASE on next page
to start with we initialize the DSP and set a starting condition.
Then we enter a loop.
BUFZ is a trap that waits for the buffer to be empty. While in this trap nothing else can be done.
As soon as the buffer is empty BUFZ drops through to
GETCART – this is where you can send the new coordinates to the robot. You have some time because in theory the DSP is still working on the last segment. If you are using a PC supervisor then you need to send a message or a character from GETCART to the PC it to indicate GETCART is ready for the coordinates. The PC then sends the coordinates in some form. GETCART needs to parse the string and get the values in to X Y Z PITCH and W (roll) (5-axes) or X Y Z PITCH YAW and ROLL (6-axes). Or just send X Y and Z if the hand parameters do not change. See later...

As soon as the new coordinates are received NEXTSEG computes the changes in motor counts and sends them to the DSP. The DSP attempts to do this move within the time in SEGTIME. For a short move the speed will be low and for a long move it will be high.

NEXTSEG returns a status which should be zero. If you sent impossible coordinates such that the speed required to do that move within SEGTIME is too great then NEXTSEG returns false and we need to stop everything. The procedure prints the “segment too long” message and aborts with error 31. (that value goes into ERR.) If using a supervisor you can change this to some other string or single character or even an output on the PA port.

A project is provided, CHASE. It may be downloaded from the ST Robotics downloads page as CHASE.ZIP. In this project is a route called LIST1 which is just a list of Cartesian coordinates.

INIT sets workable values for ACCEL and SEGTIME. Change them as necessary.

To test the algorithm a program RJ is provided (random jump). A number is picked from the sequence
CREATE SEQUENCE
1 , 5 , 3 , 9 , 10 , 4 , 2 , 8 , 7 , 3 , 1 , 0 , (0 means stop
so that 1 means get the coordinates from line 1, 5 means get coords from line 5 etc. The numbers are picked at random although a jump from 10 to 1 could result in the too long error.
The result is as on the youtube video at
http://www.strobotics.com/videos/chase.mp4
This shows two clips:
The first is with ACCEL set to 3000 and SEGTIME set to 500mS
As you can see the robot speeds up to complete the longer moves.
Next we changed the SEGTIME to 300mS
As you can see the robot stops part way through because the segment can not be achieved in 300mS, with the appropriate error message. In other words the robot could not make it from line 10 to line 4 in 300mS.

What happens if the move is very small?
The robot moves the short distance in SEGTIME time, so it will move very slowly. Remember also it has to complete each move before it can start on the next. However if the next move is a large one the robot will start accelerating before it finishes the current slow move ready for the faster move.
What happens if the coordinates do not change?
The robot then continues to do zero movement in SEGTIME time. And the segment stored for next move may also be zero so as the next move will not begin until the current one is finished if, for example SEGTIME is 1000, then you may wait up to a second before motion starts again.

```
( CHASE COORDINATES )
: BUFZ ( wait for buffer empty
BEGIN
 24 SPSB SPRB
78 < UNTIL
;
: NEXTSEG
TRANSFORM DROP ( convert to motor values
  DSPCHANS @ 0 DO
       TARGET I IND @ DUP ( IND is just 2* +
       OLDP I IND @
       - SWAP ( leave rel pos on stack, axis 6 on top
       OLDP I IND ! ( store rel pos in OLDP
  LOOP
  1 ( flag for DSP ) TSEG
?STAT
;
: GETCART
( GET THE NEXT COORDINATES TO AIM FOR
;
: CHASE
VSET ( send ACCEL to DSP
CFLAG C1SET ( enable DSPASSUME

( set starting position
DSPCHANS @ 0 DO
     I GLOBALS @ OLDP I IND !
LOOP
( send initial zero move to DSP
MVST ( start new move
     0 0 0 0 0 0 1 TSEG ( first segment zero move
MVRN ( start a run sequence

( now enter loop to collect coordinates and send to DSP
BEGIN ( loop that collects next coordinates
    BUFZ ( wait for buffer empty before sending next coords
    GETCART ( get the next coordinates
    NEXTSEG ( send motor values to DSP
    ( status from NEXTSEG ) 0 > IF ( fail if non zero
        STOP
        BEGIN ?RUN 0= UNTIL ( wait for DSP to stop
        DSPASSUME
        ." Segment too long" 31 ABORT
    THEN
?TERMINAL UNTIL

MVND ( end move sequence
DSPRDY ( wait for DSP to stop
DSPASSUME ( update counts from DSP
;
```

Help sheet 18

ST Robotics

Suggested method for getting coordinates from the supervisor into GETCART using the outer interpreter.

```
: GETCART
( send a starting character to the PC – PC is waiting for this.
2 EMIT ( SEND STX (start of text)
( or send asterisk 42 EMIT
( now enter outer interpreter to get values from PC
OUTER
( PC sees the start char and sends up to 6 values followed by the command
EXIT then return (13):
( in this form:
( 3000 X ! 2000 Y ! 1000 Z ! 900 PITCH ! 0 W ! EXIT or
( 3000 X ! 2000 Y ! 1000 Z ! 900 PITCH ! 0 YAW ! 0 ROLL ! EXIT etc
( or just the ones that change e.g.
( 3000 X ! 2000 Y ! 1000 Z ! EXIT
( now all these values are in the correct variables.
; ( end of definition
```

or using the stack – you must send all 5 values, or all 6 for a 6-axis robot:

```
: GETCART
2 EMIT
OUTER
( just send all 6 values in reverse order e.g.
( 0 0 900 1000 2000 3000 EXIT
( these 5/6 values are on the stack.
#CARTS @ 0 DO
  I 2* X + ! ( pick up from stack
LOOP
;
```

Warning, if you don't send anything the DSP will time out and robot may crash.