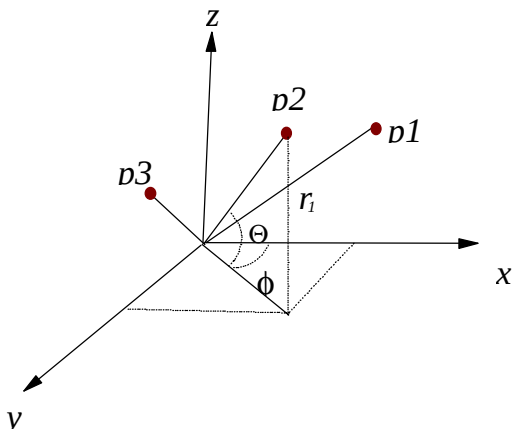
10 Moves in Polar Coordinates

This application describes the DSPL programming for moves in polar co-ordinate.



The application program moves a three-axis motion system from *p*1 to *p*2 and *p*3 in the polar coordinate. The three points, *p*1, *p*2 and *p*3 are characterized by their *r*, Θ and ϕ as follows:

*p*1: r_1, Θ_1 and ϕ_1
*p*2: r_2, Θ_2 and ϕ_2
*p*3: r_3, Θ_3 and ϕ_3

The following illustrates `"main.hll"` that performs the required moves. This program uses external routines contained in programs `"coordinate_xfer.hll"` and `"get_a_point.hll"`.

Polar Coordinate Move, ‘main.hll’

```
#define  x      var20
#define  y      var21
#define  z      var22

#define  teta   var23
#define  phi    var24
#define  r      var25
#define  index  var26

#include “coordinate_xfer.hll”
#include “get_a_point.hll”

plc_program:

  run_m_program (move_in_polar_coordinate)

end_plc

move_in_polar_coordinate:

  var1 = 1
  while (var1 == 1)

    call (get_a_new_point)      ;get a point provided by either
    call (polar2cartesian)      ;the Mx4(case 1) or the host(case 2)

  wend
end
```

Point Retrieving Subroutine, ‘get_a_point.hll’

Case 1: All points are computed and stored in Mx4 by the Mx4’s own DSPL

```
get_a_new_point:

    ;*************************************************************
    ;*
    ;*  this routine is useful if end points are computed
    ;*  by the Mx4 and stored in the Mx4 table.
    ;*
    ;*************************************************************

    r = table_p(index) ;pick r, teta and phi
    index = index + 1
    teta  = table_p(index)
    index = index + 1
    phi = table_p (index)
    index = index + 1

    ret()
end
```

Case 2: All points are provided to the Mx4 in real time by the host

```
get_a_new_point:

    ;*************************************************************
    ;*
    ;*  this routine is useful if end points are provided
    ;*  by the Mx4 and stored in the Mx4 table.
    ;*
    ;*************************************************************

    r = var30    ;host uses instruction change_var to update points
    teta = var31 ;to update points characterized by:r,teta and phi
    phi  = var32

    ret()

end
```

Polar to Cartesian Xformation, ‘coordinate_xfer.hll’

```
polar2cartesian:

    ;*********************************************************
    ;*
    ;*  this routine transfers polar to Cartesian
    ;*  coordinate.  And executes a trapezoidal move
    ;*  to reach the target point within a specified time
    ;*
    ;*********************************************************

    x = r * cos (phi) * cos (teta)
    y = r * sin (phi) * cos (teta)
    z = r * sin (teta)

    axmove_t(0x7, x_accel, x, y_accel, y, time, z, z_accel, time)

    ret()
end
```