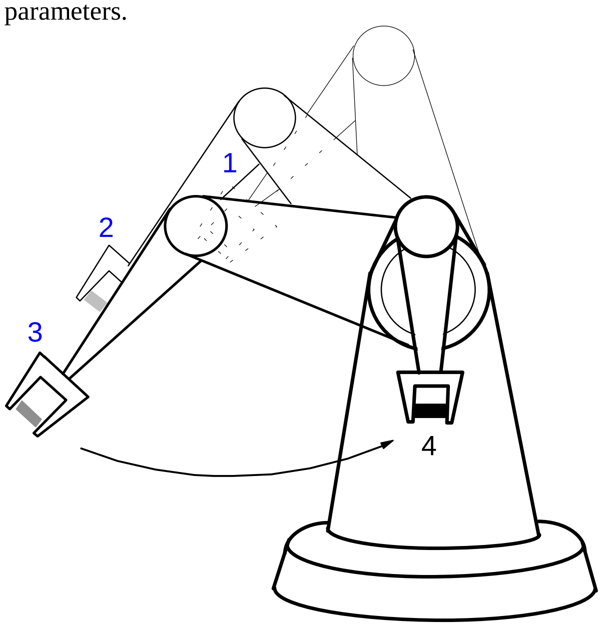
3 Time Based Motion Programs

In the following application a series of moves for multiple joints are to be completed within the specified times: `t1,t2,...` respectively. This means that all motors must reach their intermediate target positions (`posx, posy, posz and posw`) simultaneously. The DSPL instruction `AXMOVE_T` is ideal for this application. It is important to note that a real time execution of `AXMOVE_T(or AXMOVE)`with its new move parameter(s) will intercept the one in progress. There are two ways to supply a DSPL program with target positions (and/or other move parameters). The first method allows the host to update move parameters using real time command `CHANGE_VAR`. This command is provided with the Mx4 C++ /Visual Basic 32-bit DLL. In the second method the DSPL retrieves the move parameters from its own table memory. Alternatively, the DSPL can use its own floating point math for real time computation of move parameters.



1) Host updates the target positions to reach in a specified time

In this case host updates the target points. The communication protocol between DSPL and host programs is as follows. First, the DSPL resets `flag = 0` to let host program know it can update target points. Host uses command `CHANGE_VAR` to update the target points. Upon the completion of variable update, host sets the `flag = 1` to let DSPL program know update is finished. The DSPL uses the recently updated variables as arguments for `AXMOVE_T` command and resets the `flag = 0` to let the host program know that once again host is allowed to update target points.

```
#define    accx   var2
#define    posx   var3
#define    t      var4
#define    accy   var5
#define    posy   var6
#define    accz   var7
#define    posz   var8
#define    accw   var9
#define    posw   var10
#define    flag   var11

#include “init_mx4.hll”

plc_program

    run_m_program(move_arm)

plc_end

move_arm:
    call(init_mx4)         ;initialize the gains
    t = 200                ;set time to 200*200µsec = 40 ms
    flag = 0               ;tell the host it can update motion
                           ; parameters
    wait_until(flag == 1)  ;wait until host finished updating
                           ; parameters

    while (var1 == 1)
        axmove_t(0xf, accx, posx, t, accy, posy, t, accz, posz, t,
                       accw, posw, t)

        flag = 0           ;tell host it can change move parameters

        wait_until(cpos 1 == posx);wait until move is finished
        wait_until(flag == 1)  ;host sets flag upon updating motion
                               ; parameters
    wend
end
```

2) DSPL calculates / retrieves the target positions to reach in a specified time

In this case, the target points are retrieved from the Mx4 table memory. The subroutine, `get_points` performs this data retrieval. The variable `size` holds the number of prestored target points. To download target position to the Mx4 table memory, you may use the `download position` facility provided with Mx4pro v4.

```
#define        size   var1
#define        accx   var2
#define        posx   var3
#define           t   var4
#define        accy   var5
#define        posy   var6
#define        accz   var7
#define        posz   var8
#define        accw   var9
#define        posw   var10
#define        flag   var11

#include “mx4_init.hll”

plc_program
    run_m_program(move_arm)
plc_end

move_arm:
    t = 200                   ;set time to 200*200µsec = 40 ms
    accx = 1
    accy = 1
    accz = 1
    accw = 1

    size = 500                ;the total number of moves

    call(get_points)
    while (size >= 1)

        axmove_t(0xf, accx, posx, t, accy, posy, t, accz, posz, t,
                      accw, posw, t)
        targetx = posx
        call(get_points)

        wait_until(cpos 1 == targetx) ;wait until move is finished
        var1 = var1 - 1
    wend
end
```

```
get_points:
    posx = table_p(index)  ;retrieve one set of 32-bit target points
    index = index + 2
    posy = table_p(index)
    index = index + 2
    posz = table_p(index)
    index = index + 2
    posw = table_p(index)
    ret()
end
```