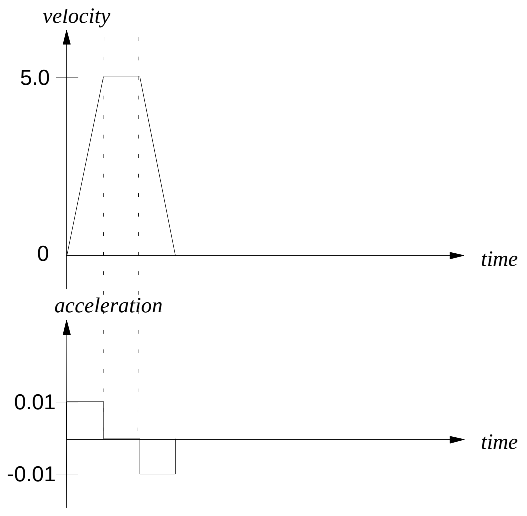
2 How Moves Are Generated

Simple Trapezoidal Move

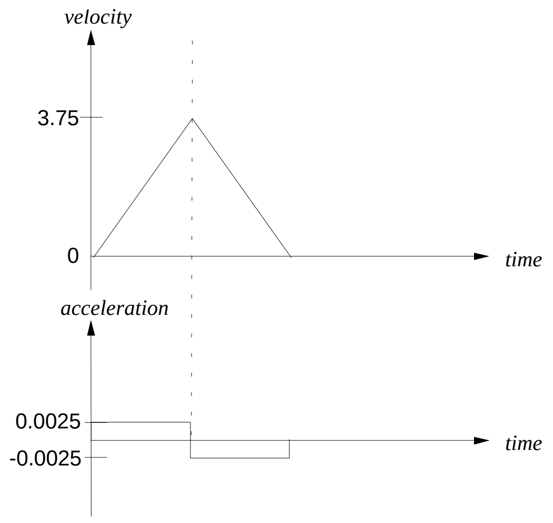
This simple motion program moves motor one from a preset position to a new position with a specified velocity profile characterized by its slew rate and acceleration.



```
trapezoid:

    pos_preset(0x1,0)         ;preset position for axis 1
    axmove(0x1,0.010,5000,5.0);move to position 5000

end
```

Simple Triangular Move

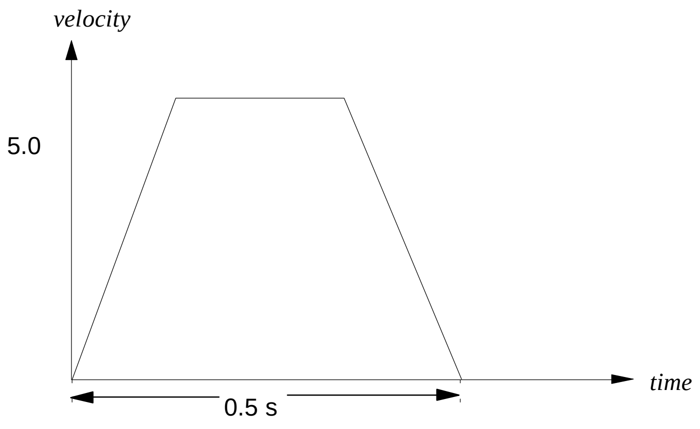
This program moves motor one from an initial position of 0 to a final position of 5,625 counts on a triangular velocity profile. This profile uses an acceleration of 0.0025 counts/(200 µs)^2 and target velocity of 5.0 counts/200µs.



```
triangle:

    pos_preset(0x1,0)
    axmove(0x1,0.0025,5625,5.0)

end
```

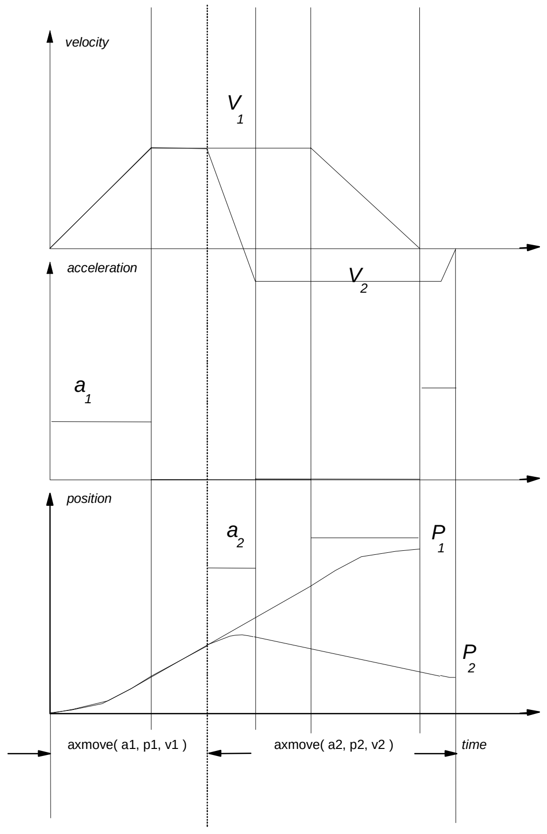
Time Based Trapezoidal Move

This program moves motor one from an initial position of 0 to a final position of 20000 counts in 500 ms (or 2500*200µs) at acceleration = 1 count/(200 µs)^2. Velocity for this move will be automatically calculated by the Mx4.



```
time_trapezoid:

    pos_preset(0x1,0)
    axmove_t (0x1,1,20000,2500)

end
```

Intercepting a Trapezoidal Move

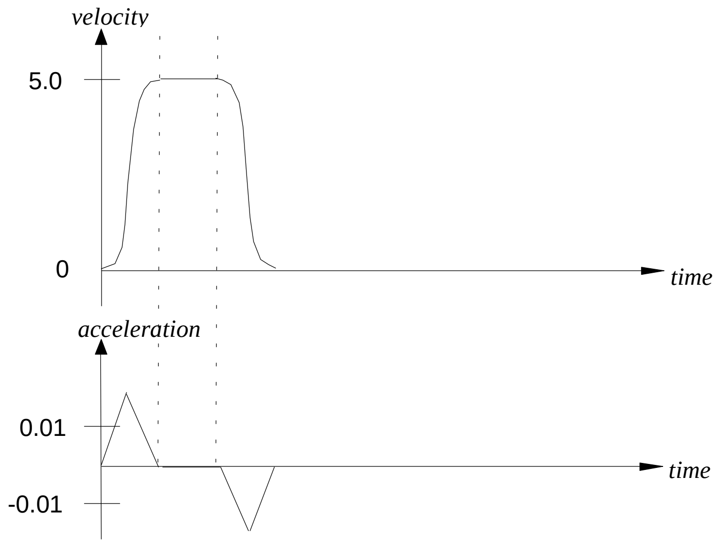


The first move with programmed acceleration (a_1), end position (p_1), and speed (v_1) can be intercepted by another move with an entire different set of arguments at any time.

```
intercept:

    pos_preset(0x1, 0)
    axmove(0x1, 0.01, 50000, 20.0 )   ;move to P1
    wait_until( cpos1 > 25000 )       ;intercept axmove1 at 25000
    axmove(0x1, 0.01, 0, -10.0 )      ;return to origin

end
```

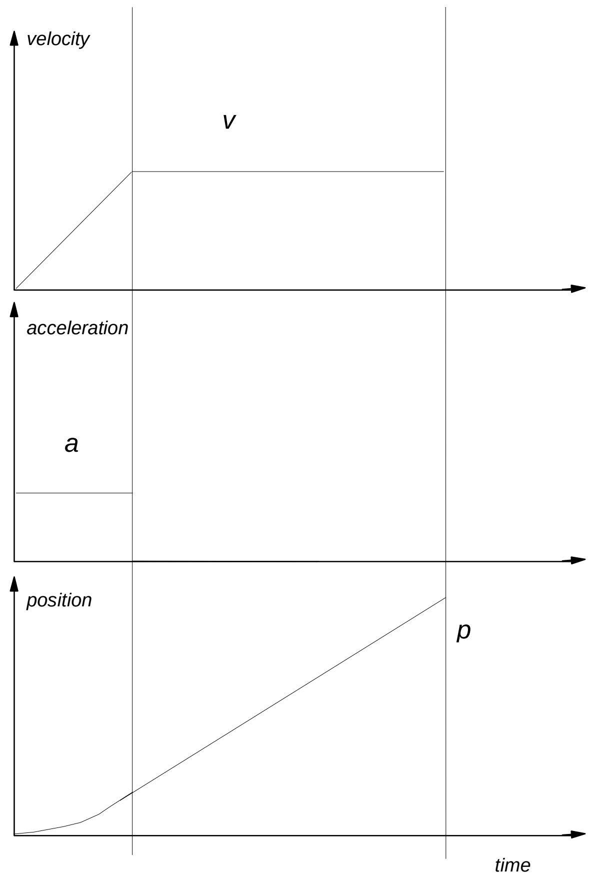
S-Curve Trapezoidal Move

This simple motion program moves motor one from a preset position to a new position with a specified s curve velocity profile characterized by its slew rate and acceleration. Note that the maximum acceleration achieved in the move will be twice that programmed as the acceleration argument.



```
scurve_trapezoid:

    pos_preset(0x1,0)
    axmove_s(0x1,0.010,5000,5.0)

end
```

Trapezoidal Segment Moves

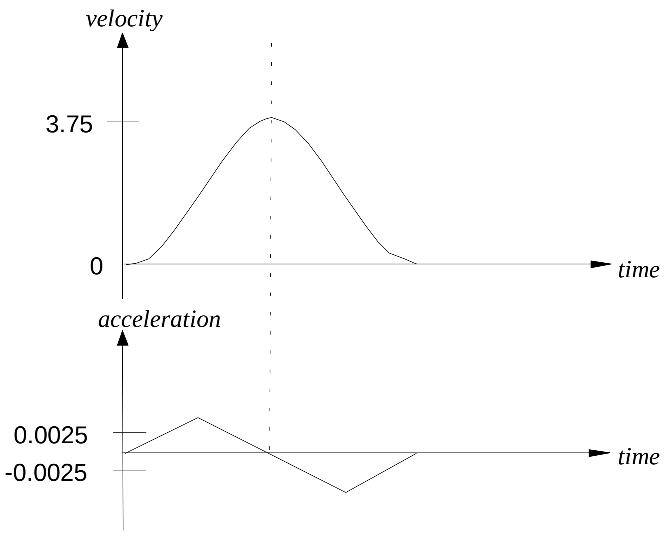


A segment is similar to the trapezoidal move, only its end position will be at its programmed slew velocity. Remember that instruction `segment` must be used in conjunction with other move instructions for blending multiple moves.

```
segment_trapezoid:

    pos_preset(0x1,0)             ;preset position for axis 1
    segment(0x1,0.010,5000,5.0)   ;move to position 5000

end
```

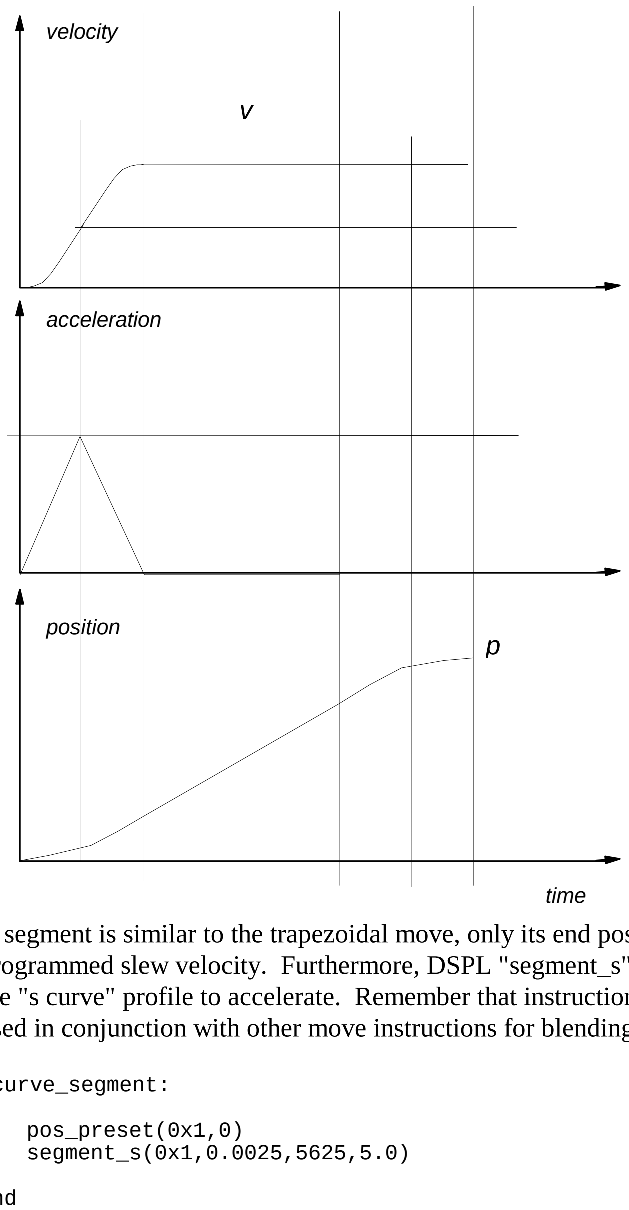
S-Curve Triangular Move

This program moves motor one from an initial position of 0 to a final position of 5,625 counts on a triangular s curve velocity profile. This profile uses an acceleration of 0.0025 counts/(200 µs)^2 and target velocity of 5.0 counts/200µs.



```
scurve_triangle:

    pos_preset(0x1,0)
    axmove_s(0x1,0.0025,5625,5.0)

end
```

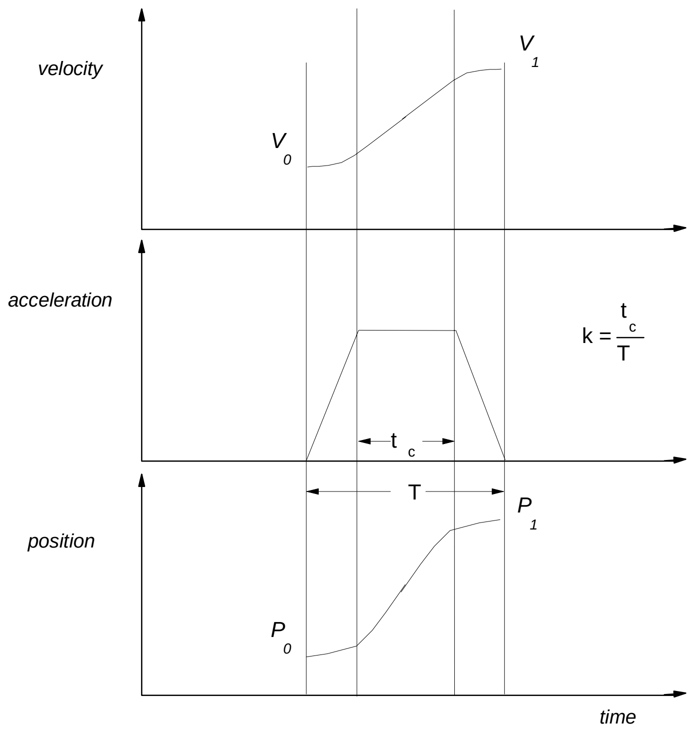
Segment Moves With S-Curve



A segment is similar to the trapezoidal move, only its end position will be at the programmed slew velocity. Furthermore, DSPL "segment_s" instruction uses the "s curve" profile to accelerate. Remember that instruction `segment` must be used in conjunction with other move instructions for blending multiple moves.

```
scurve_segment:

    pos_preset(0x1,0)
    segment_s(0x1,0.0025,5625,5.0)

end
```

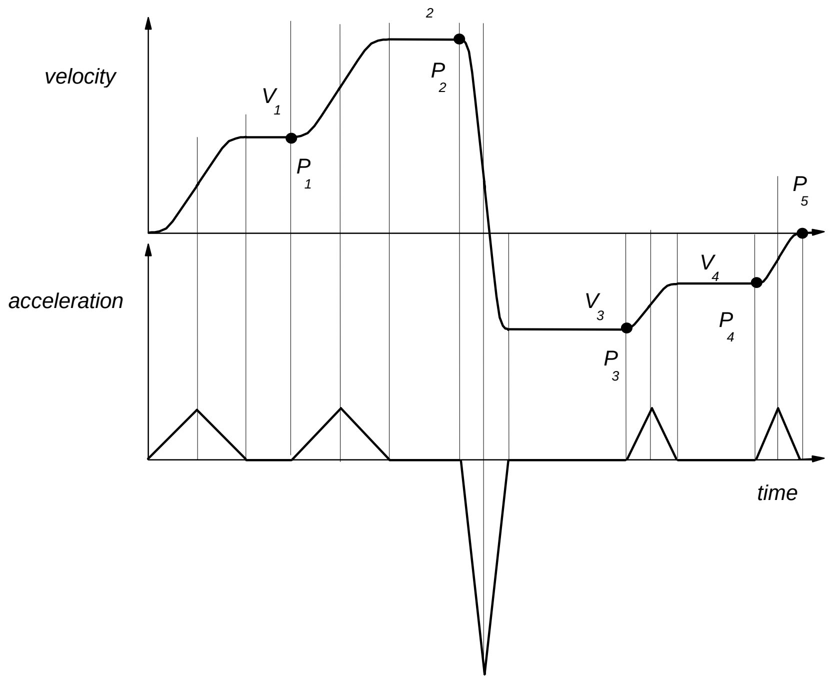
"S" Curve Linear Move



Constrained by a programmed maximum acceleration, the "linear_move_s" command will linearly ramp the speed from its initial value to a new programmed value at a programmed position. The acceleration used with this command is "s" curve and its jerk is determined by: a, k and T. Remember that instruction `linear_move_s` must be used in conjunction with other move instructions for blending multiple moves.

```
linear_smove:

    pos_preset(0x1,0)
    maxacc(0x1, 0.1)
    linear_move_s(0x1,0.01,5000,5.0,7500)

end
```

Blending Moves

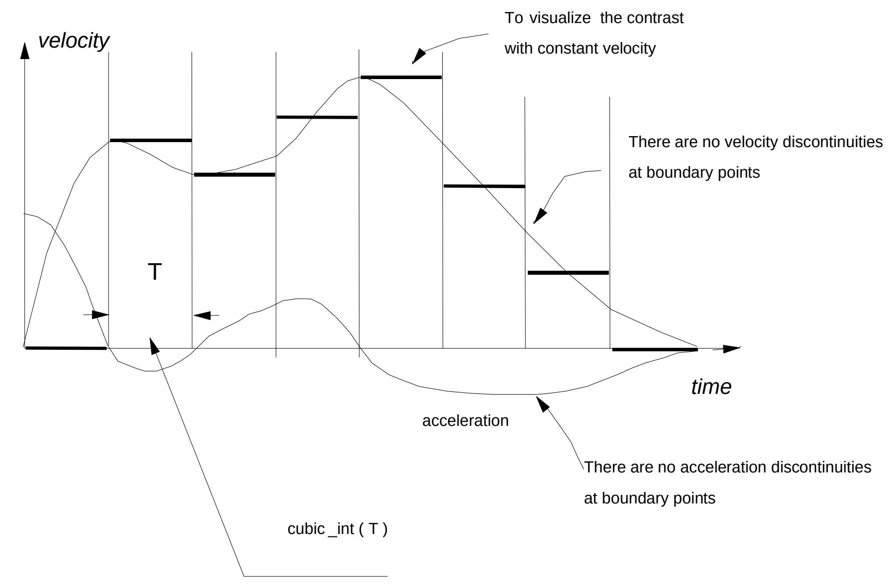


Either one of the DSPL Segment Moves, (segment or segment_s) can be used in blending move applications. Blending allows the migration of one move to another at an exact programmed position without a need to stop the motion.

The following example will generate the velocity profile shown above.

```
plc_program:
       run_m_program( blended_move )
end


blended_move:

    ctrl( 0x1,1000,15000,5160,3140 )  ; set the gains for axis 1
    pos_preset( 0x1, 0)               ; preset axis 1 position

    var1 = 1
    en_motcp(0x1)
    while( var1 < 10 )
        segment_s(0x1, 0.05, 7750,  15 )      ; move to P1

        segment_s(0x1, 0.05,  23250,  30 )    ; move to P2

        segment_s(0x1, 0.15,  15000, -15 )    ; move to P3

        segment_s(0x1, 0.05,  7000, -7.5 )    ; move to P4

        segment_s(0x1, 0.05, 5000, -0.1 )     ; move to P5

        delay(1000)

        axmove(0x1, 0.1, 0, -30)              ; move to origin

        delay(1000)

    wend
end
```

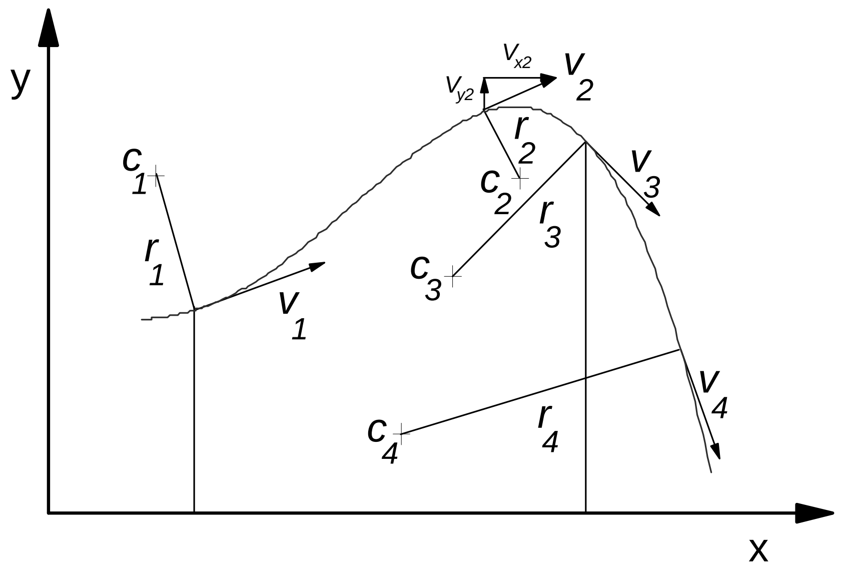
Cubic Spline Moves


To visualize the contrast with constant velocity
velocity
There are no velocity discontinuities at boundary points
T
time
acceleration
There are no acceleration discontinuities at boundary points
cubic _int (T)

Motion control applications requiring fine moves from one point (a point with up to four components for positions and velocities) to another require cubic spline interpolation. Mx4 can run cubic splines either in contouring mode (the host continually updates Mx4's DPR with a new set of points), or in table mode (Mx4's table is pre-loaded with a set of points only once). In table mode the user array can be up to 2,000 points long. Each point specifies the position and velocity for only one motor. The following example illustrates the generation of a cubis spline table and it use.

```
plc_program:

        run_m_program (cubic_spline)

end

cubic_spline:
    ;*****************************************************
    ;
    ;    In this example, 63 xy cubic spline points
    ;    are generated by the DSPL and positioned into
    ;    the Mx4's data memory
    ;
    ;*****************************************************

    ; set the gains
    ctrl (0x3,1000,15000,5160,3140,1000,15000,5160,3140)
    pos_preset(0x3,0,0)       ; preset positions
    maxacc(0x3,1,1)           ; set stop accelrate

    var3 = 0
    while (var3 <=125)        ; generate 63 xy points
       var2 = var3/2          ; on a circle
       var4 = sin(var2)
       var5 = cos(var2)
       var4 = 1000*var4
       var5 = 1000*var5
       table_p(var3) = var4
       var3 = var3 + 1
       table_p(var3) = var5
       var3 = var3 + 1
    wend

    cubic_rate(300)           ;  set time interval
    cubic_scale(0x3,1,0,1,0)  ;  set scale for points
    cubic_int(126,0,0,0x3)    ;  start interpolation

end
```

PVT Hermite Spline



The Hermite Spline or PVT (position, velocity and time) allows the user to specify a set of position, velocity and corresponding time interval for each axis. The velocity for each axis is the resolved component of vector velocity (shown in the figure as Vx2 and Vy2). This vector is tangent to the radius of a local curvature. An example of the use of a PVT spline can be found in the *Mx4* or *Mx4 Octavia User's Guide*.