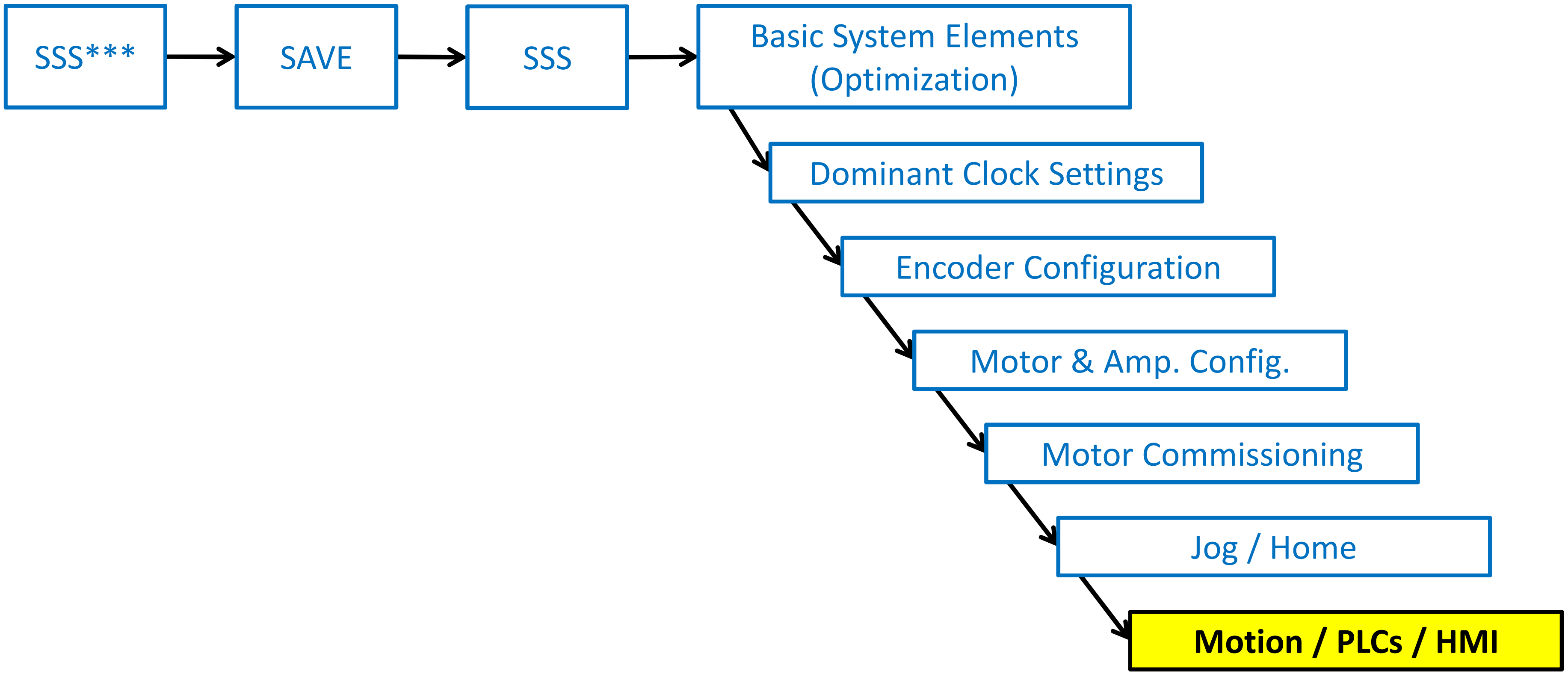
OMRON | DELTA TAU

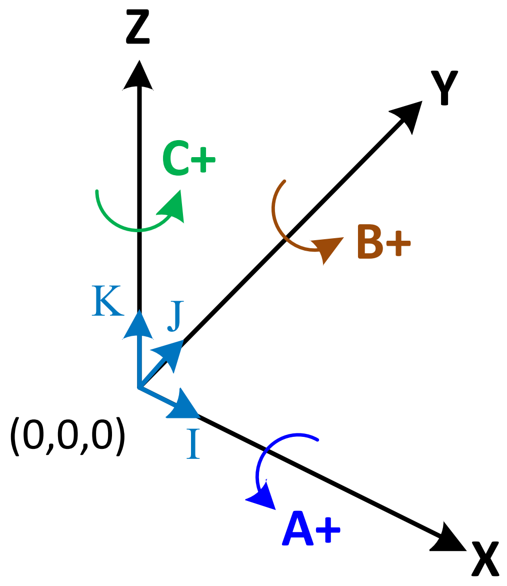
Coordinate Systems
&
Motion Programs

System Configuration



Coordinate System Overview

- **Main structure for providing multi-axis coordination**
  - Difficult, often impossible to perform with Jog moves
- **Axes to start, stop, or change speed at the same time should be in the same coordinate system**
- **Axes to act independently should be in separate coordinate system**
- **Key concept**
  - Mapping of motors (actuators) to axes (programming) coordinates
  - Two methods for assigning a motor to an axis
    - Axis definition
      - Typically one to one mapping
      - Optional rotation and translation (offset)
    - Kinematics
      - Typically for non-linear mapping
      - Greatly useful for non-Cartesian geometries like robots



Coordinate System Axes

- **There are 128 (0 .. 127) coordinate systems available**
  - **Sys.MaxCoords** specifies maximum number (for optimization)
  - Most users will not employ CS0 as a real CS

- **Each coordinate system can carry up to 32 axes**
  - **4** sets of **Cartesian** axes
    - Permit 3D space rotation and translation
    - Only [X Y Z] and [XX YY ZZ] permit
      - Circular interpolation in 3D
      - Tool radius compensation
  - **2** sets of **Rotary** axes
    - Permit rollover, **Coord[].PosRollOver[]**
  - Other axes
    - General purpose use, e.g. force / auto-focus loop

AXES NAMES			
A	**Z**	HH	SS
B	**AA**	LL	TT
C	**BB**	MM	**UU**
U	**CC**	NN	**VV**
V	DD	OO	**WW**
W	EE	PP	**XX**
X	FF	QQ	**YY**
Y	GG	RR	**ZZ**

Axis Definition

&1#1->819200X + 128000

Coordinate system number

Motor number

Optional offset in motor units

Motor units per axis units
(= 1 if PosSf & Pos2Sf ≠1)

Possible definitions

- One to one
  - &1#1->X
- Linear combination(s)
  - &1#1->X+2Y-10
- Kinematics
  - &1#1->I

Note

Must assign null definition first before moving motor from one coordinate system to another

Useful commands

- Undefine all
  - Clear axes definitions of all coordinate systems
  - Automatically assigns active motors to null in CS0
- Undefine
  - Clear axes definitions of currently addressed CS
- Null definition, e.g. &1#1->0
  - Detaches motor from any axis
  - To remove from the coordinate system completely
    - Assign to CS0 or another CS

Axes Definitions

- A motor assigned to an axis, in a coordinate system, cannot be simultaneously assigned to another axis

- A motor assigned to a coordinate system cannot be simultaneously assigned to another coordinate system

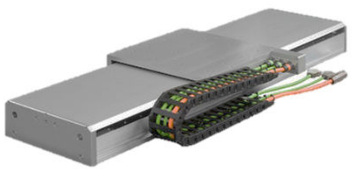
Legal Assignments

```
&1#1->X
&1#2->X
```

Motors #1 and #2 act as X-axis in the same coordinate system.

```
&1#1->X
&2#2->X
```

Motors #1 and #2 act as X-axis in different coordinate systems.



Bad Practices

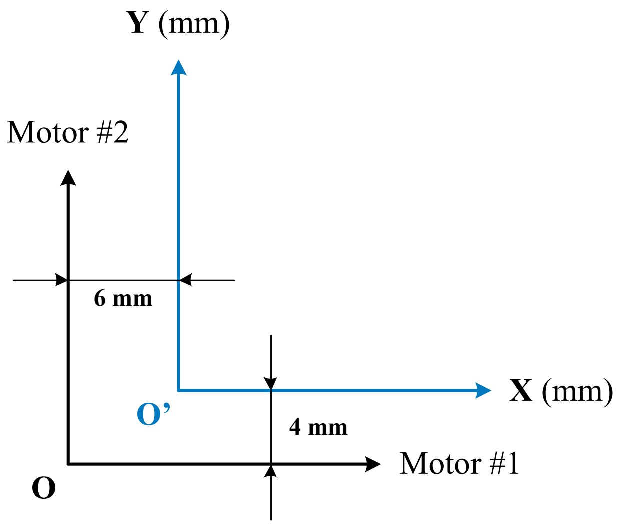
```
&1#1->X
&1#1->Y
```

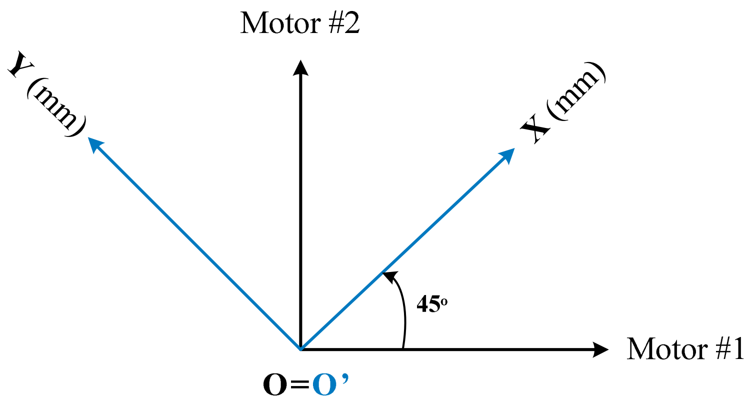
The second statement will cancel the first one (it is re-assigned).

```
&1#1->X
&2#1->X
```

The second statement will be Rejected (it is already assigned, tied to another CS).

Translation & Rotation Example





&1#1->X + 6

&1#2->Y + 4

&1#1->0.707X – 0.707Y

&1#2->0.707Y + 0.707X

Note

It is possible to perform the same (and more) transformations using transformation matrices

Motion Program Overview

- **Sequenced and coordinated multi-axis trajectory execution**
  - Motion programs execute line-by-line.
  - Moves on the same line will finish at the same time.

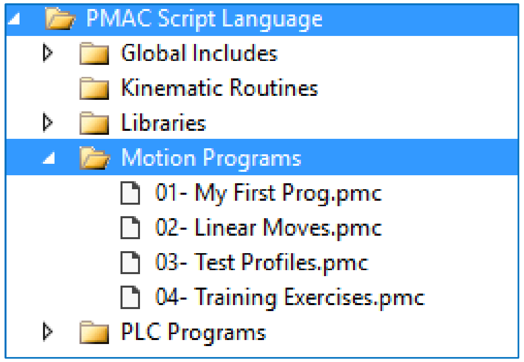
- **Written with standard flow control syntax and functions**
  - Can synchronize I/Os with axes moves
  - Can perform mathematical, logical, and I/O related operations
    - IF/ELSE Statements, WHILE loops, SWITCH statements
  - Subprogram calls (with arguments) and jumps; CALL, GOSUB, and GOTO commands

- **Constraints**
  - Must run in a coordinate system
  - A motion program does not (necessarily) belong to a coordinate system
    - It can be run in up to 128 different coordinate systems <u>simultaneously</u>
  - Size is limited by PMAC memory (~1GB)

Motion Program Structure

- **A motion program starts with an OPEN PROG (name or number} and ends with a CLOSE**
- **You can either give the motion programs a name or a non-negative integer number**
  - When given a name, the IDE will automatically assign them numbers, starting with 100000
    - By the order that they are downloaded
  - Program number 0 is reserved to the rotary buffer
  - Best to stick to either numbering or naming convention (recommended for clarity)

```
OPEN PROG ExamplePROG
ABS RAPID X 0 Y 0
DWELL 0
LINEAR TA 190 TS 10 TD 190 F 1
X 0.500000 Y 1.000000
X 1.000000 Y 0.250000
X-1.000000 Y-0.000250
DWELL 0
CLOSE
```



Position Modes

- **Absolute ABS**
  - Endpoint programmed with respect to the “zero” origin of the coordinate system

- **Incremental INC**
  - Endpoint programmed with respect to the present axis position

Caution

If not explicitly specified, a new motion program command will use the last programmed position mode

Move Modes: Rapid

- **RAPID**
  - Sequenced Jog moves, trapezoidal velocity-vs-time profile
  - For fast, point-to-point moves
  - No blending with other move modes
  - Can interrupt new endpoint every servo cycle (e.g. send continuously)
  - Motor control elements
    - Motor[].JogTa, Motor[].JogTs, Motor[].JogSpeed
    - Motor[].MaxSpeed
    - Motor[*x*].RapidSpeedSel
      - =1 (default) MaxSpeed governs speed
      - =0 JogSpeed governs speed
  - CS control element Coord[].RapidVelCtrl
    - = 0 (default) Each axis finishes the move separately with its own top speed
    - = 1 All axes finish the move at the same time

Move Modes: Linear

- **LINEAR**
  - Straight line path in Cartesian coordinates, trapezoidal velocity-vs-time profile
  - Point-to-point moves, most common motion program move mode
  - Can blend with LINEAR, CIRCLE, or SPLINE moves
  - Cannot interrupt until end of move
    - Hint: use small moves to break into new one(s)
  - Move parameters
    - **Coord[].Ta, TA**, acceleration time in msec
    - **Coord[].Ts, TS**, acceleration s-curve time in msec
    - **Coord[].Td, TD**, (final) deceleration time in msec
    - **Coord[].Tm, TM**, move time in msec

      OR

      Coord[].Tm , F, vector feedrate
      - Axis units per Coord[].FeedTime
      - Typically per seconds, FeedTime = 1000 msec

- **If Coord[].SegMovetime = 0, linear moves are constrained by**
  - Motor[].MaxSpeed
    - Maximum program velocity [axis units/ms]
  - Motor[].InvAMax
    - Maximum program acceleration [ms^2/axis units]
  - Motor[].InvDMax
    - Maximum program deceleration [ms^2/axis units]
  - Motor[].InvJMax
    - Maximum program jerk [ms^3/axis units]

- **Coord[].SegMovetime ≠ 0**
  - Circular interpolation
  - Kinematics
  - Multi-block lookahead

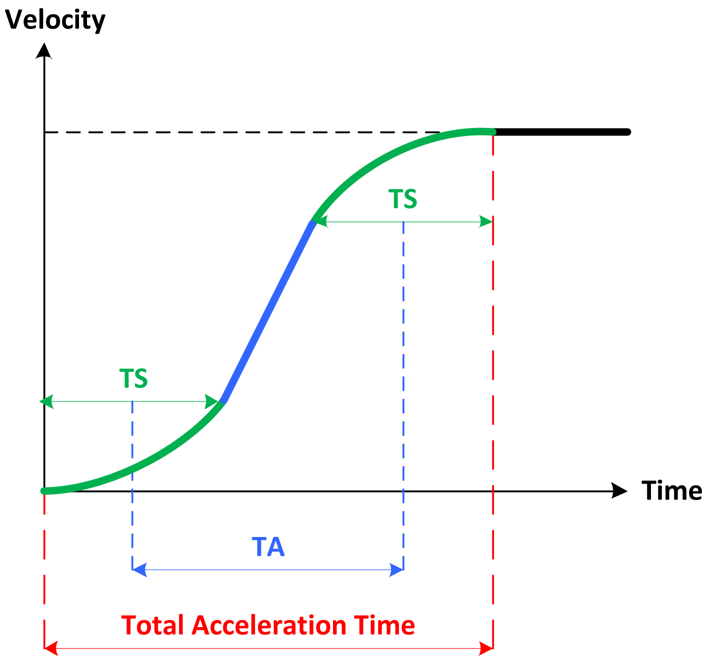
Move Modes: Linear Profile

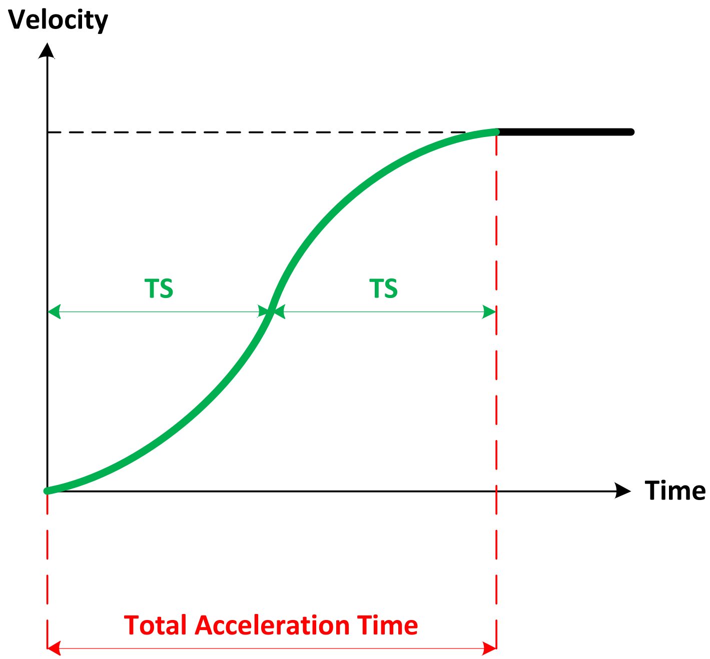
➢ Linear move acceleration rules

- If TA >= TS
  - Total Accel. Time = ½TS + TA + ½TS
    = TA + TS
- If TA < TS
  - Total Accel. Time (extended) to = 2 * TS

Note

Same as Jog acceleration time rules. Also, same for deceleration TD





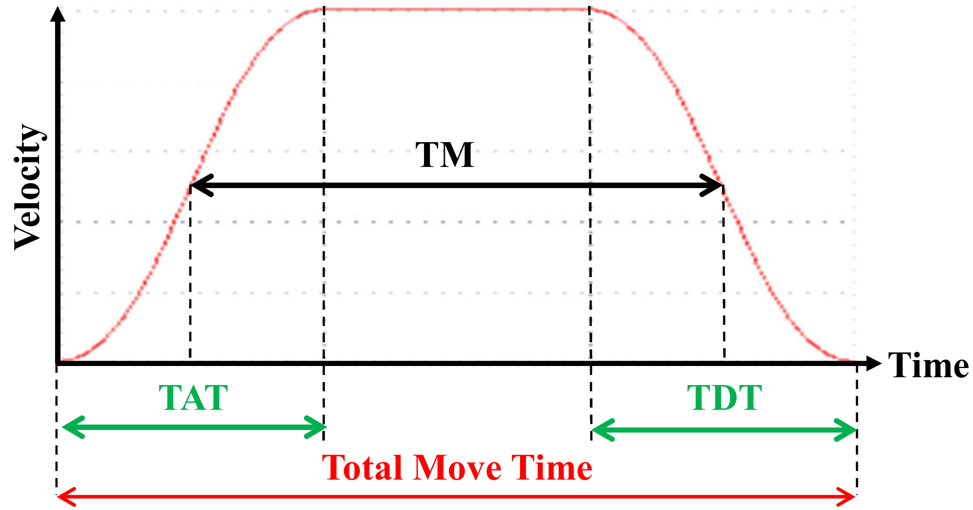
Move Modes: Linear Profile

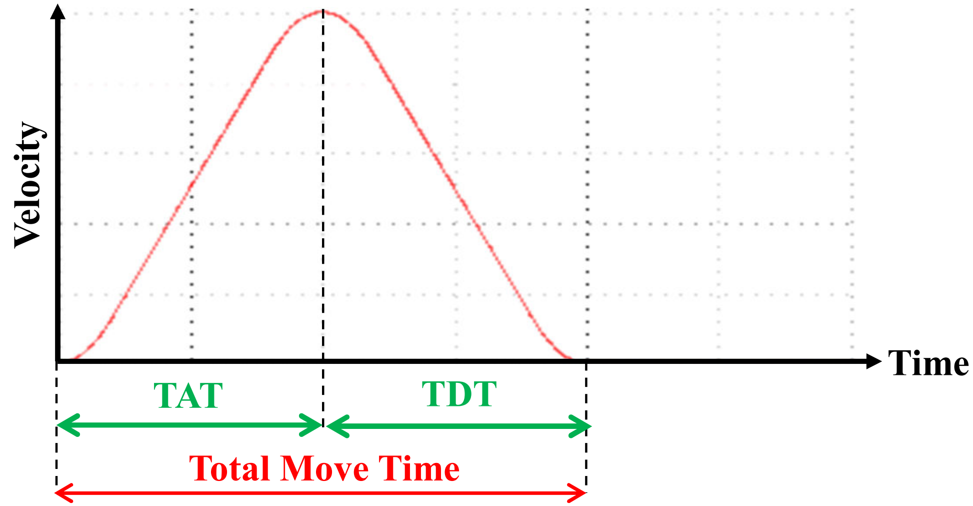
- **Linear move profile w/ TM**
  - TM used for time-specified moves
  - If TM >= Total Acceleration Time TAT
    - Total Move Time = TM + TAT
  - If TM < Total Acceleration Time TAT
    - Total Move Time (extended) to = 2 * TAT

- **Linear move profile w/ F**
  - Vector feedrate, F is used for speed-specified moves
  - Total move time = $\frac{\text{Distance}}{\text{F}}$ + TAT

Assuming TAT = TDT for simplicity

Note



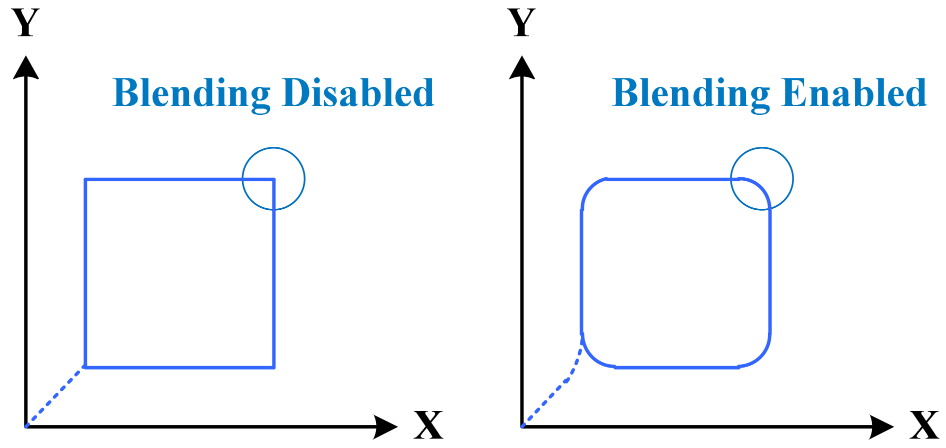


Move Blending

- **No intervening stop between moves**
  - Enabled by default

- **Blending can be performed between**
  - Linear, circular, and spline moves

- **Blending is disabled if any of the following is true**
  - Moves are separated by a DWELL
  - Coord[].NoBlend
    - = 1, linear – circle blending disabled
    - = 2, spline – spline blending disabled
    - = 3, all blending disabled
  - More than (Coord[].GoBack + 1) jumps without a move in a WHILE or GOTO loop



- **Blending control elements**
  - Coord[x].CornerBlendBp, Coord[x].CornerDwellBp
    - Between linear and circular
  - Coord[x].CornerAccel, Coord[x].CornerRadius
  - Coord[x].InPosTimeout

Move Blending Example

- **For 2 simple linear moves**
  - For simplicity, no s-curve acceleration time

- **With Blending Off**
  - Total Move Time =

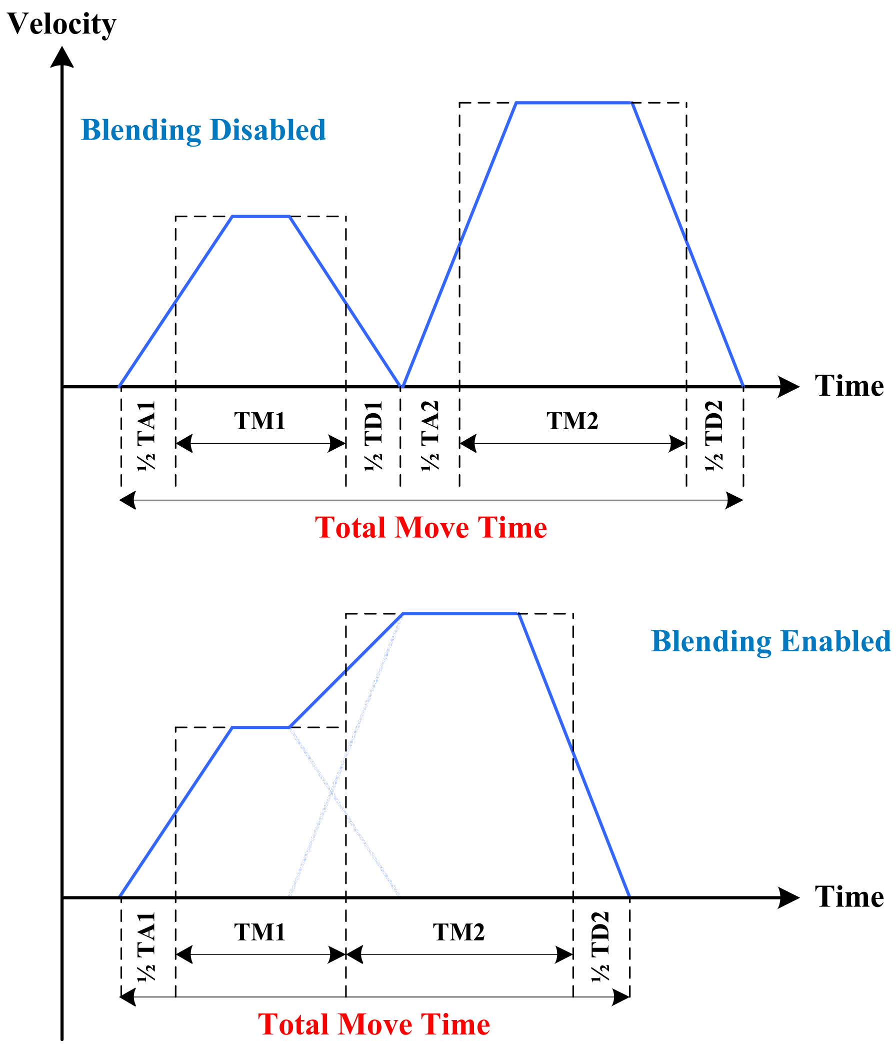
  ½ TA1 + TM1 + ½ TD1 + ½ TA2 + TM2 + ½ TD2

- **With Blending On**
  - Total Move Time =

  ½ TA1 + TM1 + ½ TD1 + ½ TA2 + TM2 + ½ TD2

Note

If s-curve was also used in this example, then ½ of Total Accel or Decel times are used instead of just TA and TD



Dwell vs. Delay

- **DWELL {data}**
  - Maintain present position for specified time in msec
  - A dwell starts exactly at the end of the deceleration
  - A dwell command stops all “lookahead” pre-calculations
    - Disables blending
  - The dwell time is always the specified time. It is not affected by the time base feedrate override %

- **DELAY {data}**
  - Maintain present position for specified time in msec
    - Equivalent to a zero-distance move
  - A delay starts at the middle of the deceleration
  - A delay command does not stop any “lookahead” calculations
    - Allows blending
  - The delay time varies with the time base feedrate override %
  - If delay time < Total Acceleration Time then
    - Delay time is extended to TAT

Note

Delay must be used for a slave in external time base or kinematics tracking to prevent discontinuities in the calculation

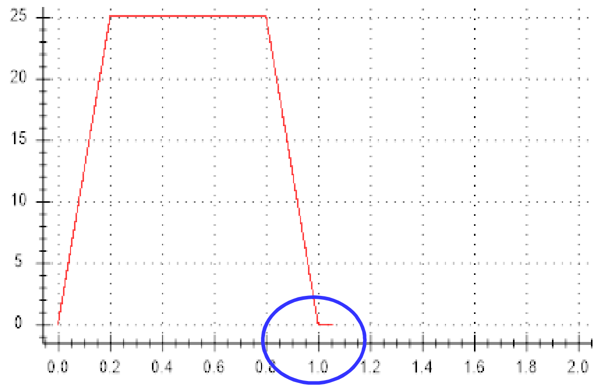
Dwell vs. Delay

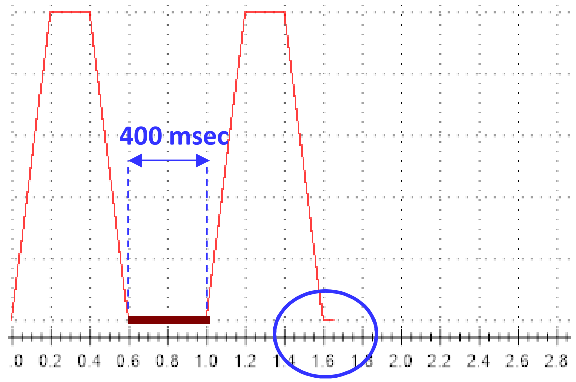
```
LINEAR TA200 TS0 TD200 TM400
INC X 10
INC X 10
```

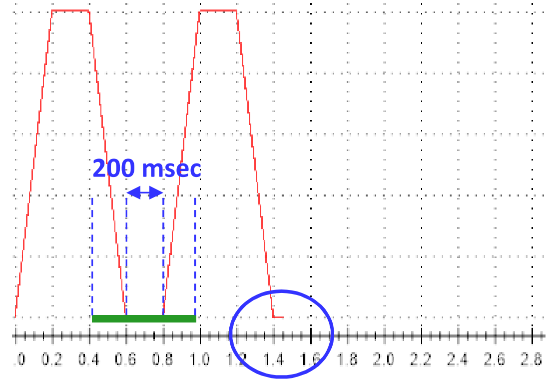
```
LINEAR TA200 TS0 TD200 TM400
INC X 10
DWELL 400
INC X 10
```

```
LINEAR TA200 TS0 TD200 TM400
INC X 10
DELAY 400
INC X 10
```

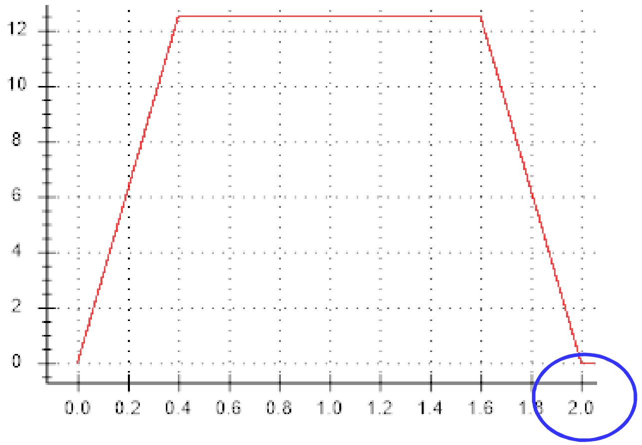
&1 %100

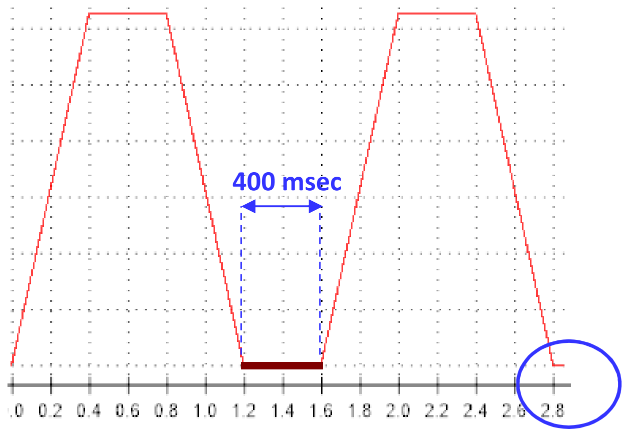


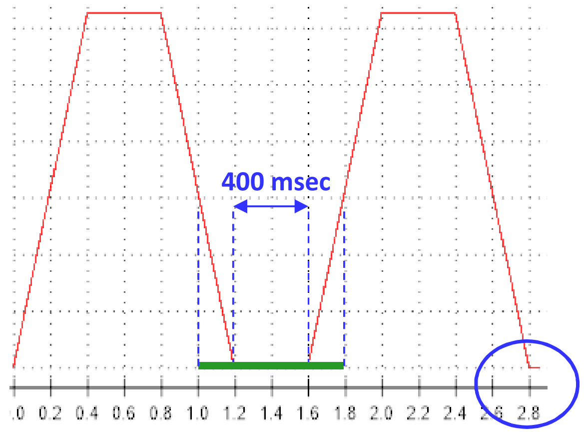




&1 %50







Motion Program Execution

To execute a motion program

- Online terminal (e.g. execute program 5, or ExamplePROG in CS 1)
  - &1 B5R
  - &1 B ExamplePROG R

  Or
  - &1 START 5
  - &1 START ExamplePROG
- Program buffer, e.g. from PLC (same example)
  - START 1: 5
  - START 1: ExamplePROG

Note

If the program is assigned a name, space is required. But, in general, PMAC's parser is not space sensitive

To view the contents of a (downloaded) motion program

- LIST PROG 5
- LIST PROG ExamplePROG

One line motion program online Execution

- **CPX** command processor 1-line motion program execution
- Useful for troubleshooting and point-point teaching
- E.g. **&1 CPX ABS LINEAR F 10 X 0**

Caution

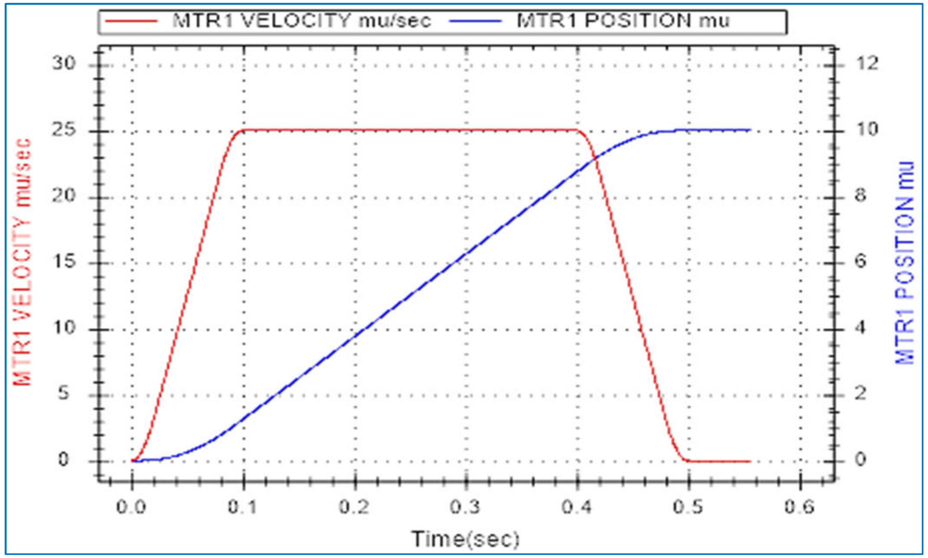
The CPX command, if not explicitly specified on the same line, uses the last programmed move mode, acceleration, feedrate or TM

Note

Make sure motor program velocity, acceleration, and jerk limits are properly set before executing a motion program

Motion Program Exercise 1

- **Motor #1 assigned to the X-axis, and configured in mm**
- **PROG 1; Requirements:**
  - Rapid X move to startpoint (0) @ 100 mm/sec
  - Wait until X-axis is in-position (to within ±5 um)
  - Linear X move to 10 mm in exactly 500 msec
    - **100 msec total accel/decel (same) time**
      - Using TA, TS, and TD of choice
    - **500 msec total move time** programmed using TM
  - Wait until X is in-position (to ±5 um)
  - Use DWELLs as needed
    - As long as they do not affect the move requirements
  - Gather data in-program for the 0 – 10 mm move
    - Velocity, and position versus time
    - Display data to your boss (same as shown plot)

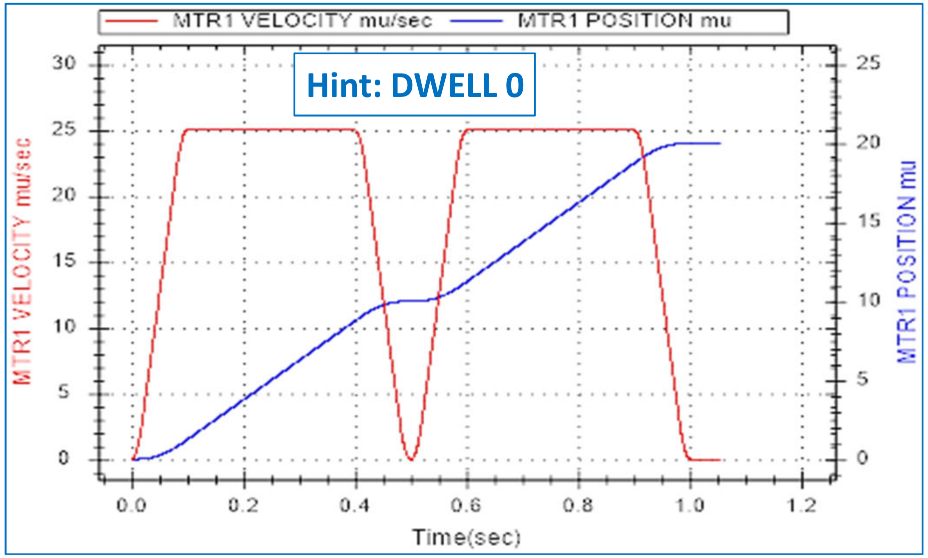


Here is some help
But incorrect solution!

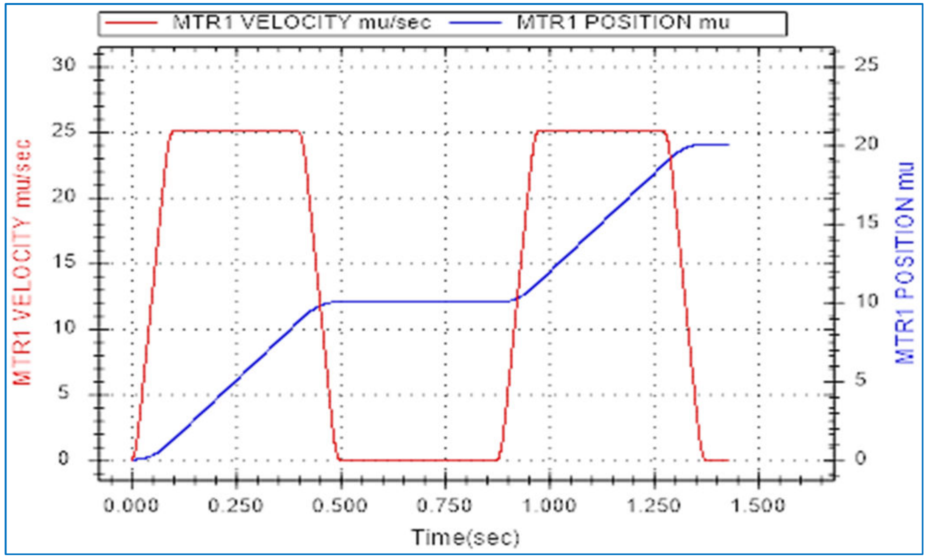
```
OPEN PROG 1
…
ABS RAPID X 0
…
Gather.Enable = 2
…
ABS LINEAR TA 160 TS 40 TD 160 TM 800 X 10
…
DWELL 0
Gather.Enable = 0
CLOSE
```

Motion Program Exercise 2

- **PROG 2; Requirements:**
  - Same X-axis, and requirements as PROG 1
    - Hint: copy program and rename
  - 2 consecutive (non-blended) 10 mm moves instead of 1
    - Same accel/decel profile(s) as PROG 1
    - Can use ABS or INC move mode (your choice)
    - **Total move time for both moves** must be exactly **1 second**
  - Replicate shown plot



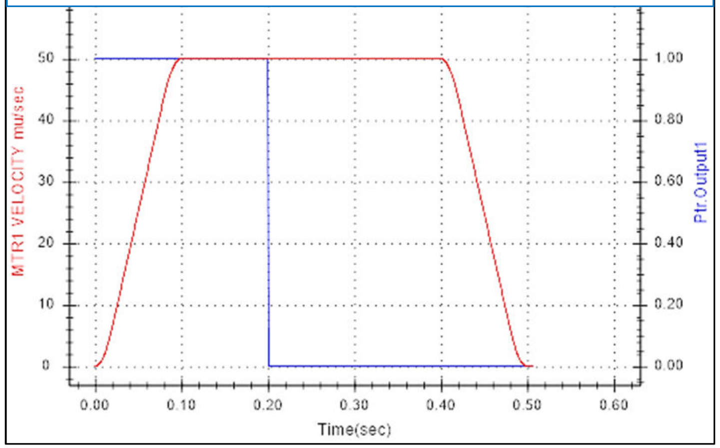
- **Quiz / practice**
  - What is the total move time (from 0 to 20 mm)? If:
    - A 475 msec DWELL is added between the 2 moves
    - A 475 msec DELAY is added between the 2 moves
    - The two moves were blended without any DELAY or DWELL



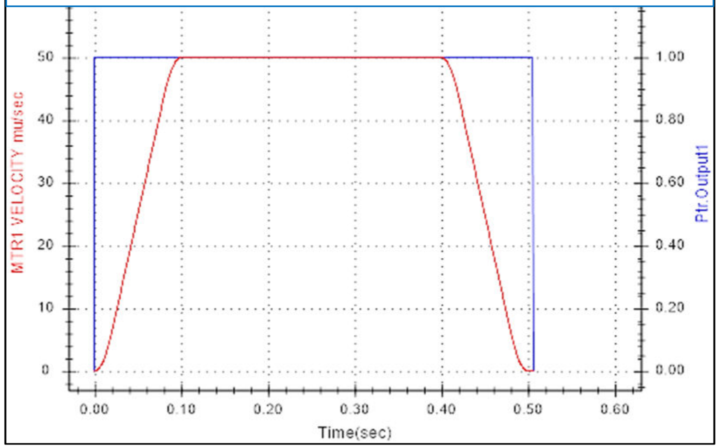
Motion Program Pre-calculation

- **With blending, PMAC automatically pre-computes moves ahead of time**
  - Rapid: **No** pre-calculation
  - Linear and circular blending: **1 move** ahead pre-calculation
  - Non-segmented linear and spline: **2 moves** ahead pre-calculation
  - Segmented lookahead: Controlled by **Coord[x].LHDistance**, unit in segments
  - 2D tool radius compensation: Controlled by **Coord[x].CCDistance**, unit in program moves
- **Implications on variable or I/O assignments**
  - Pre-calculation disallows variable or I/O synchronization with motion
    - Quick fix: synchronous == assignment

Output1 =1 at end of blended move



Output1 == 1 at end of blended move

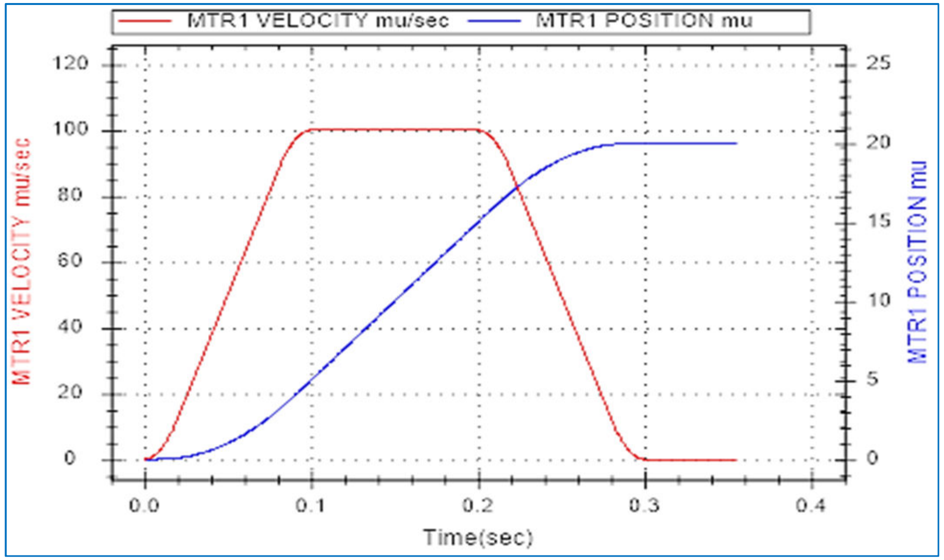


Motion Program Exercise 3

- **PROG 3; Requirements:**
  - Same X-axis, and requirements as PROG 2
  - We chose to blend the two moves
  - This time, we would like to program the move(s) at various speeds in mm/sec
  - This parameter must be a global variable called Prog3FR
    - Set equal to the speed observed in PROG 2 to obtain the same result

Hint

```
ABS LINEAR TA 80 TS 20 TD 80 F(Prog3FR) X 10
…
```



- **Quiz / practice**
  - What is the total move time if Prog3Fr is set to the maximum axis speed of 250 mm/sec
  - Why couldn't X reach the programmed feedrate (speed)?

Vector Feedrate F

- **Allows tool-tip (multi-axis/motors) speed control**
  - Regardless of the direction or move size
- **Underlying Actuator geometry can be Cartesian or non-Cartesian (kinematics)**
- **To calculate axes velocities w/ F, Power PMAC:**
  - Computes vector distance using Pythagorean theorem
  - Computes vector move time
  - Computes individual axis velocity

$$\text{Distance} = \sqrt{X^2 + Y^2 + Z^2}$$

$$\text{Move Time} = \frac{\text{Distance}}{F}$$

$$V_x = \frac{X}{\text{Move Time}} \qquad V_y = \frac{Y}{\text{Move Time}} \qquad V_z = \frac{Z}{\text{Move Time}}$$

- **Feedrate Commands**
  - FRAX(X,Y,Z) or Coord[].FRAxes = $1C0
    - Primary feedrate-axis
  - FRAX2(XX,YY,ZZ) or Coord[].FR2Axes = $D0000000
    - Secondary feedrate-axis
  - Coord[].AltFeedRate
    - Non-feedrate axis feedrate

Vector Feedrate F

```
FRAX(X,Y)
X 3 Y 4 Z 12 C 90 F 10
```

$VD = \sqrt{3^2 + 4^2} = 5$

$VMT = \frac{5}{10} = 0.5$

$V_x = \frac{3}{0.5} = 6$

$V_y = \frac{4}{0.5} = 8$

$V_z = \frac{12}{0.5} = 24$

$V_c = \frac{90}{0.5} = 180$

```
FRAX(X,Y,Z)
X 3 Y 4 Z 12 C 90 F 10
```

$VD = \sqrt{3^2 + 4^2 + 12^2} = 13$

$VMT = \frac{13}{10} = 1.3$

$V_x = \frac{3}{1.3} = 2.31$

$V_y = \frac{4}{1.3} = 3.08$

$V_z = \frac{12}{1.3} = 9.23$

$V_c = \frac{90}{1.3} = 69.2$

```
Coord[].AltFeedrate = 2
FRAX(X,Y,Z)
C 90 F 10
```

$VD = \sqrt{0^2 + 0^2 + 0^2} = 0$

$VMT = \frac{0}{10} = 0$

$V_x = 0$

$V_y = 0$

$V_z = 0$

$V_c = \frac{90}{2} = 45$

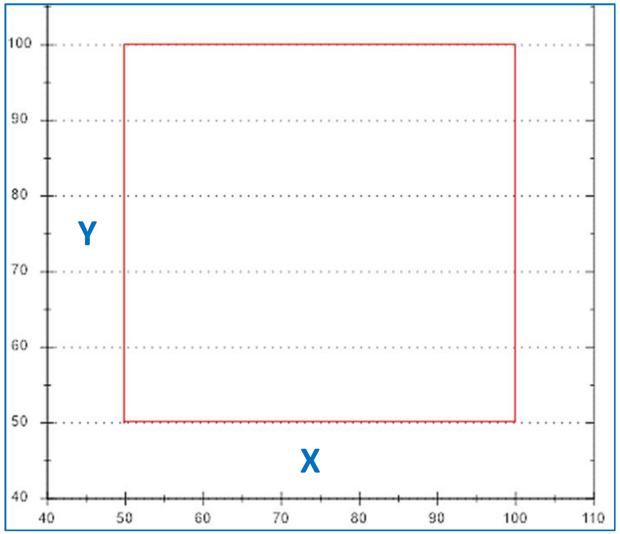
Non Vector Move Time larger, so it is used:

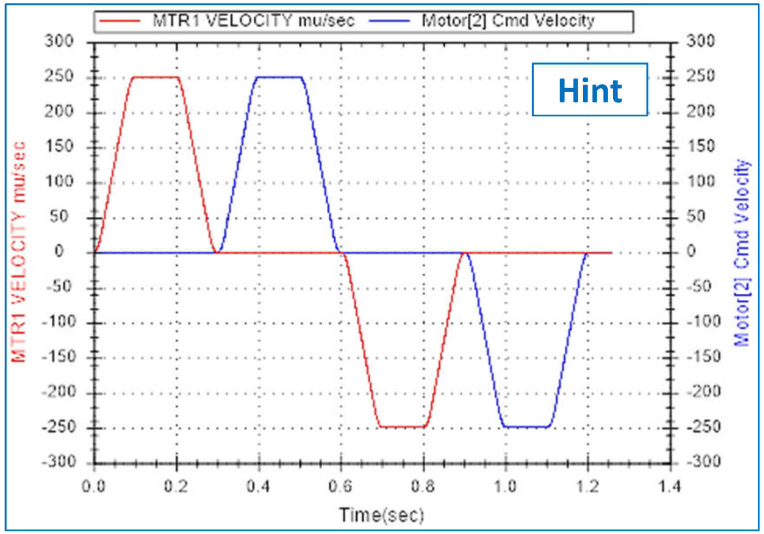
$$\text{Non Vector Move Time} = \frac{90}{\text{Coord[].AltFeedrate}}$$

Motion Program Exercise 4

- **Motor #1 assigned to the X-axis, and configured in mm**
- **Motor #2 assigned to the Y-axis, and configured in mm**

- **PROG 4; Requirements**
  - Rapid X and Y move (at the same time) to start point (50,50) @ 250 mm/sec
  - Wait until both X and Y are in-position (to ±5 um)
  - Draw a 50 mm square @ 250 mm/sec (replicating the plots below)





Move Modes: Spline

- **SPLINE**
  - Useful for moving large masses smoothly
    - Alleviates mechanical vibration and resonances
    - Improves settling time
  - Non rational cubic B spline interpolator
  - Non-segmented connected parabolic velocity-vs-time profile
  - Guaranteed to be continuous in position, velocity, and acceleration, even at move boundaries
  - Can blend with SPLINE moves only
  - Cannot interrupt until end of move
    - Hint: use small moves to break into new one(s)
  - Will not reach in-between programmed points
    - May not be suitable for precise path following

- **Not Constrained by**
  - Motor[].MaxSpeed
  - Motor[].InvAMax
  - Motor[].InvDMax
  - Motor[].InvJMax

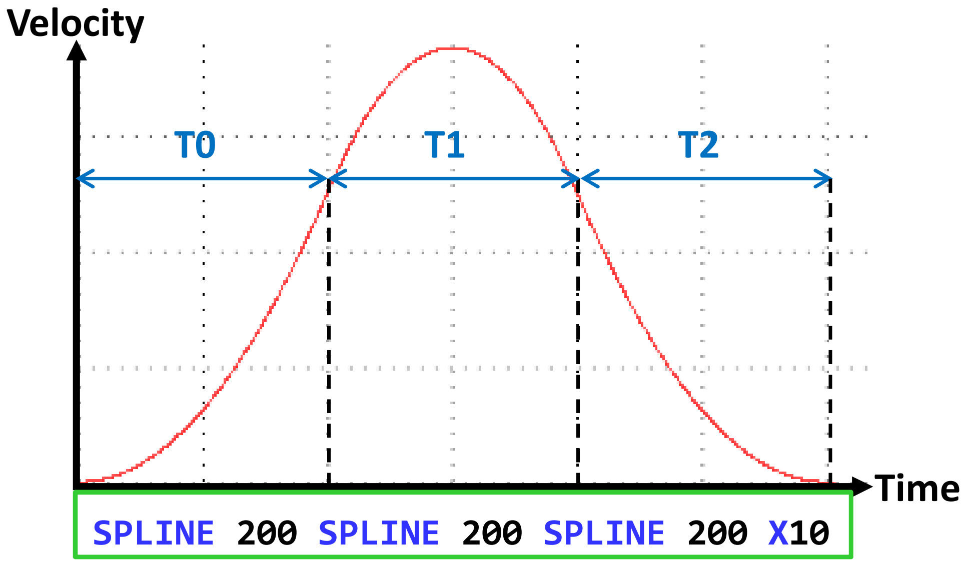
Caution

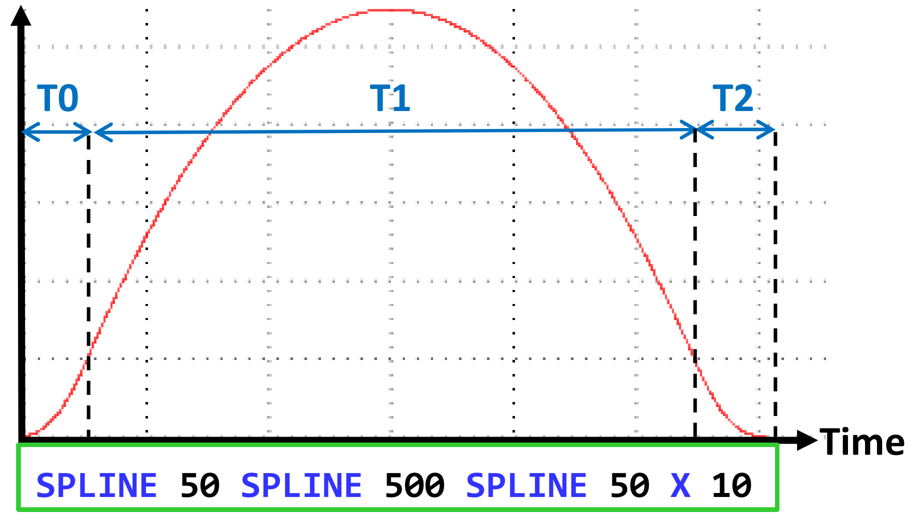
A SPLINE move will try to achieve the programmed move time at "whatever" necessary (computed) velocity and acceleration

Move Modes: Spline Profile

- **Spline move parameters**
  - Requires 3 time segments to complete 1 spline
  - Can be uniform or non-uniform
  - Can be specified in 3 ways:
    - SPLINE T0
      - T2 & T1 are automatically set = T0
    - SPLINE T0 SPLINE T2
      - T1 is automatically set = T0
    - SPLINE T0 SPLINE T1 SPLINE T2
  - T0, T1, and T2 are in milliseconds. The incoming values are stored respectively in Coord[].T0Spline, Coord[].T1Spline, and Coord[].T2Spline

- **Practice**
  - Try the shown moves using CPX commands!
  - **&1 CPX INC SPLINE 200 SPLINE 200 SPLINE 200 X10**





Move Modes: Circle

- **CIRCLE**
  - Used for circular type motion
    - Circles or arcs
  - Segmented velocity-vs-time profile
    - Coordinate system must be in segmentation mode
    - Coord[].SegMoveTime > 0
      - Typically = 4 or 8 * Sys.ServoPeriod
  - Can blend with LINEAR, CIRCLE moves

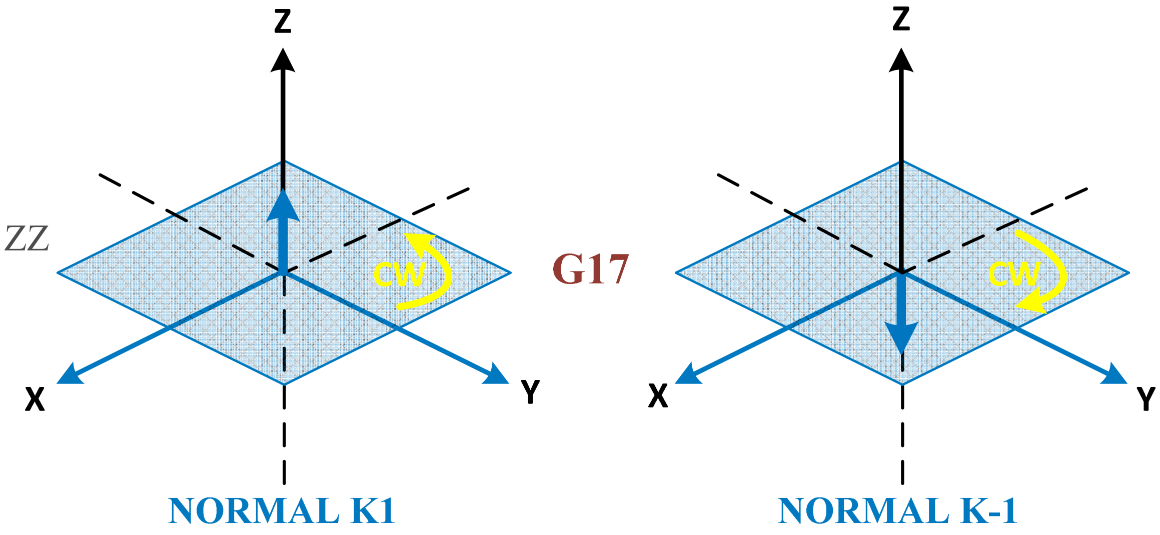
Caution

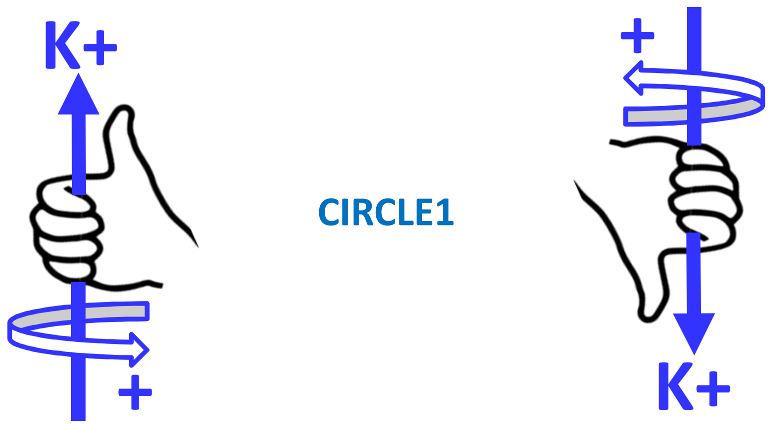
Circle moves are not limited by MaxSpeed, InvAMax, InvDMax, and InvJMax in segmentation mode “without lookahead”

- **Because SegMoveTime > 0, Not Constrained by**
  - Motor[].MaxSpeed
  - Motor[].InvAMax
  - Motor[].InvDMax
  - Motor[].InvJMax
- **Other constraints**
  - Coord[].MinArcLen
    - Minimum arc length
  - Coord[].MaxCirAccel
    - Circle acceleration limit
  - Coord[].RadiusErrorLimit
    - Automatic spiraling limit

Move Modes: Circle

- **Circle command parameters**
  - NORMAL vector
    - I, J, and K normal vectors for X, Y, and Z
    - II, JJ, and KK normal vectors for XX, YY, and ZZ
    - A combination produces 3D circular motion
  - CIRCLE orientation
    - CIRCLE1: X/Y/Z CW
    - CIRCLE2: X/Y/Z CCW
    - CIRCLE3: XX/YY/ZZ CW
    - CIRCLE4: XX/YY/ZZ CCW
  - Comply with G-code standards





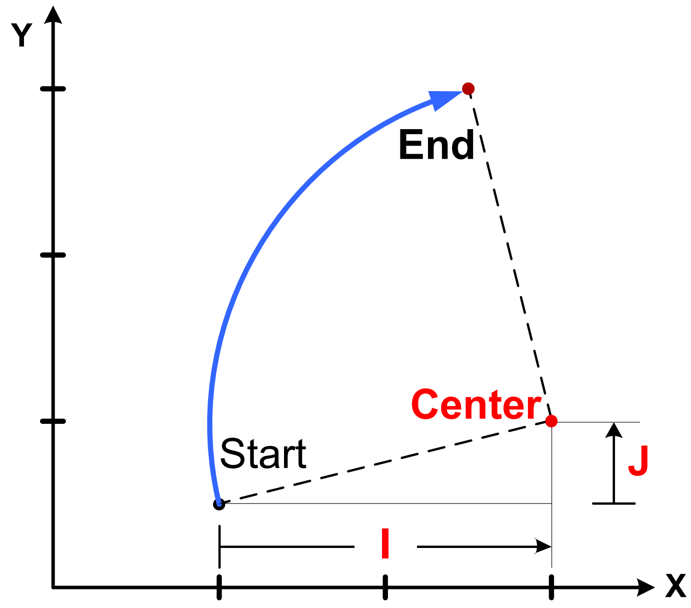
Note Same rules for ZX (G18), and YZ (G19) planes

Move Modes: Circle Command

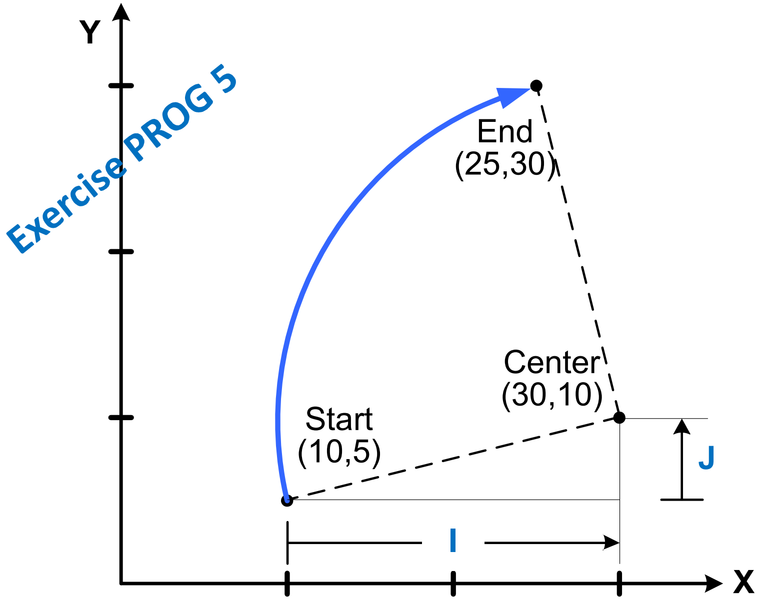
INC(I, J, K)	ABS	NORMAL K 1	CIRCLE 2	X 25 Y 30	F 100	I 20 J 5
Centerpoint coords INC or ABS?	Endpoint coords INC or ABS?	Normal vector. Which plane?	Move direction CW or CCW?	Endpoint coordinates	Move speed F or TM	Centerpoint vector Coords I J K or R

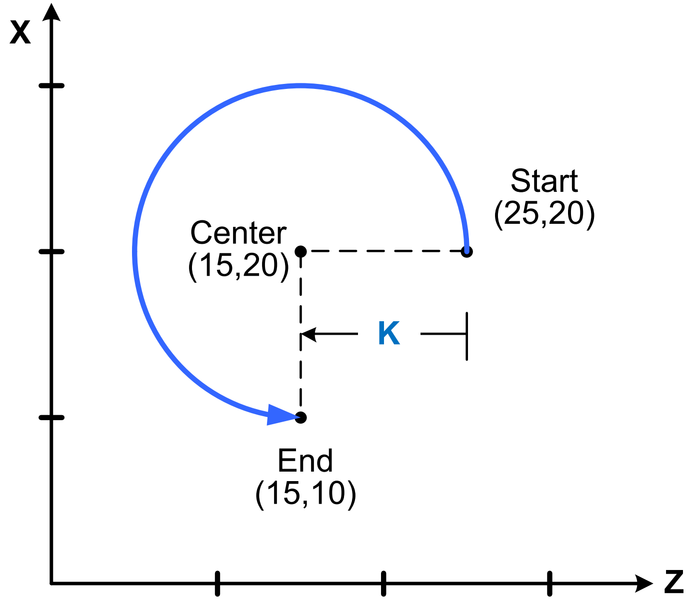
- **Arc** command specification
  - Endpoint (X, Y, Z) and center point vector coordinates (I, I, K)

  OR
  - Endpoint (X, Y, Z) and radius R
- **Full circle** command specification
  - Endpoint (X, Y, Z) and center point vector coordinates (I, I, K)
  - No full circle with R command



Move Modes: Circle Examples






```
INC(I,J,K)
ABS NORMAL K1 CIRCLE2 X 25 Y 30 F 50 I 20 J 5
```

OR

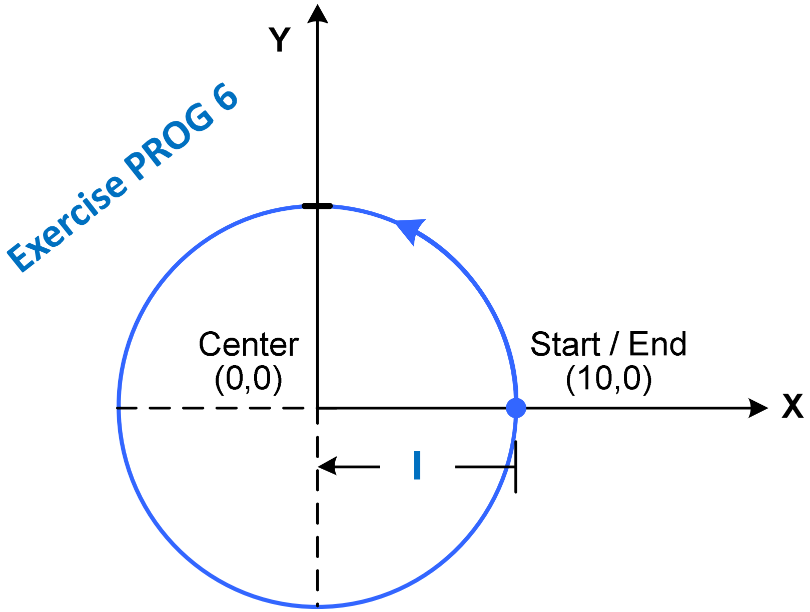
```
ABS(I,J,K)
ABS NORMAL K1 CIRCLE2 X 25 Y 30 F 50 I 30 J 10
```

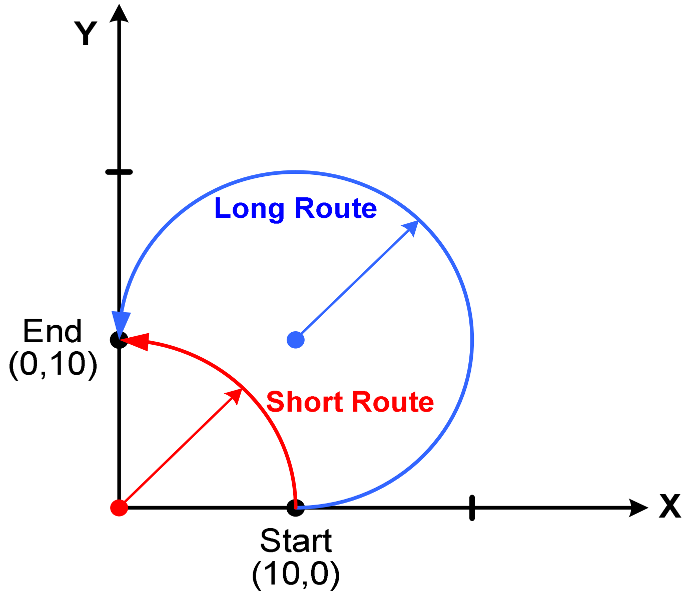
```
INC(I,J,K)
ABS NORMAL J1 CIRCLE1 Z 15 X 10 TM 500 K-10 I0
```

OR

```
ABS(I,J,K)
INC NORMAL J1 CIRCLE1 Z-10 X-10 TM 500 K 15 I 20
```

Move Modes: Circle Examples






```
INC(I,J,K)
ABS NORMAL K1 CIRCLE1 X 10 Y 0 F 100 I-10 J 0
```

OR

```
ABS(I,J,K)
INC NORMAL J1 CIRCLE1 X 0 Y 0 TM 500 I 0 J 0
```

```
ABS NORMAL K1 CIRCLE1 X 0 Y 10 F 50 R 10
```

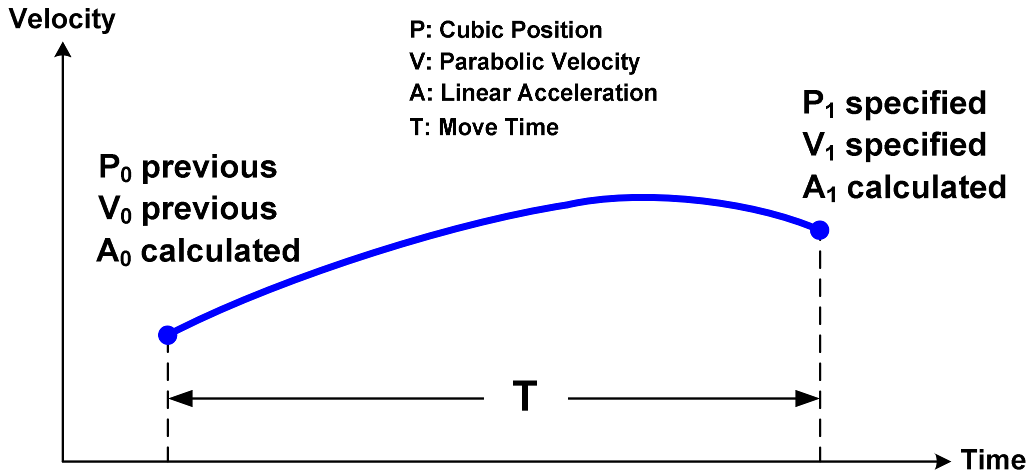
```
ABS NORMAL K1 CIRCLE1 X 0 Y 10 F 50 R-10
```

Move Modes: PVT

- **PVT (Position Velocity Time)**
  - Used for custom (tightly controlled) path generation
    - E.g. Robots handling delicate material (collision avoidance)
    - Increase throughput by avoiding dwells (linear moves) in limited workspaces
  - Specified position and velocity at the move boundaries
    - Hermite-spline path, parabolic velocity-vs-time profile
  - Can blend with LINEAR, and CIRCLE moves
  - Passes exactly through programmed points

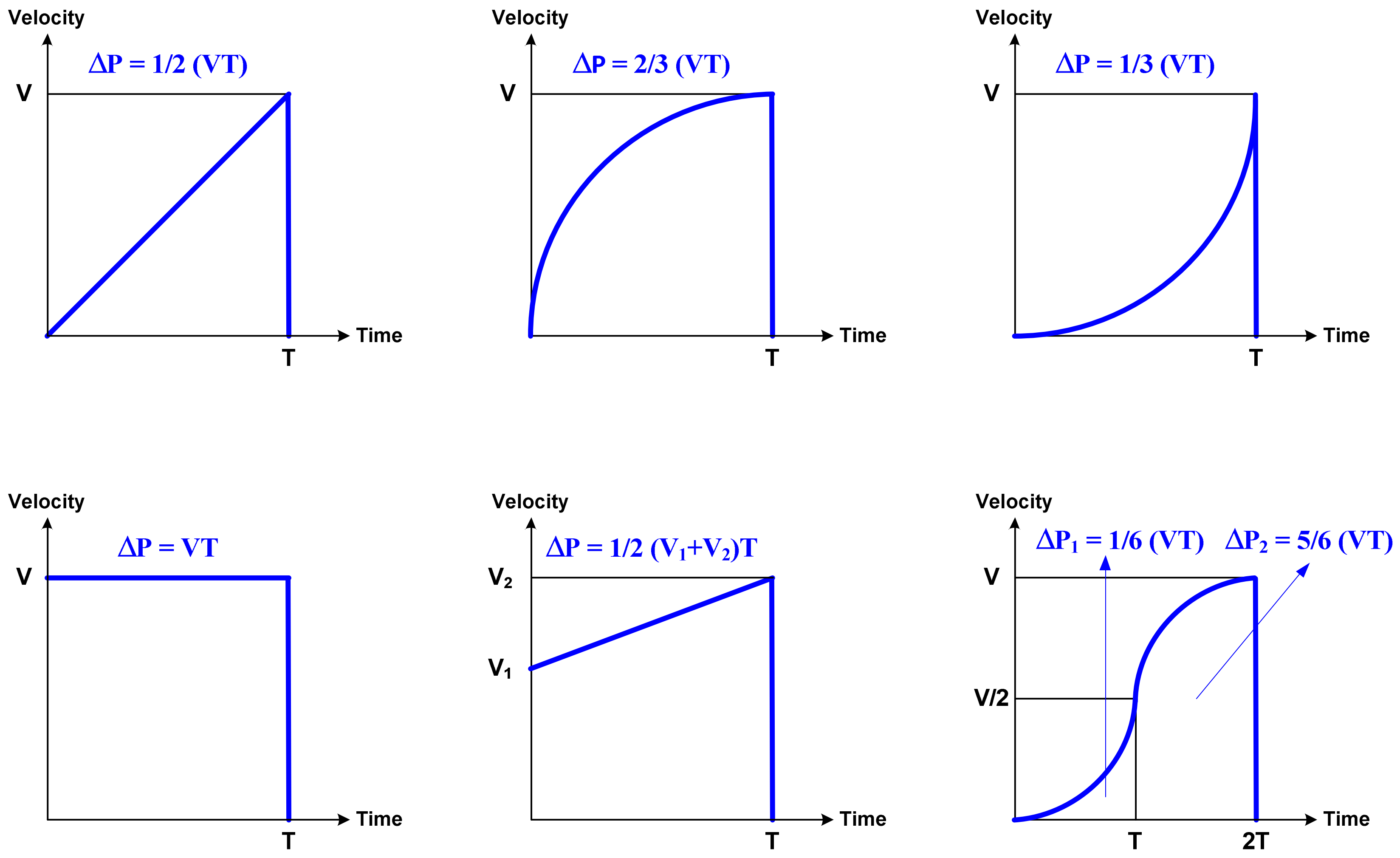
- **Not Constrained by (without lookahead)**
  - Motor[].MaxSpeed
  - Motor[].InvAMax
  - Motor[].InvDMax
  - Motor[].InvJMax

Caution PVT moves are not limited by MaxSpeed, InvAMax, InvDMax, and InvJMax in segmentation mode “without lookahead”



Note With PVT, it is the user’s responsibility to make sure that the programmed points produce graceful transition(s), continuous accelerations

Move Modes: Common PVT Shapes



Move Modes: PVT Example

- **PVT command parameters**
  - PVT **Time [msec]**

    X **position [mu] : velocity [mu/sec]**

```
INC
PVT 800
X 133.333:250
PVT 400
X 100.000:250
PVT 400
X 96.667:225
PVT 800
X 140.000:125
PVT 2000
X 83.333:0
```