Motion Programs

Coordinating synchronized motion of axes

What is a Motion Program?

- Intended for commanding moves to motors (i.e. coordinating synchronous motion)
- Must run in a coordinate system
- Power PMAC can hold as many motion programs as memory permits
- A motion program can be run in multiple coordinate systems simultaneously (up to 128)
- Can call subprograms and optionally pass arguments thereto
- Can perform mathematical, logical, and I/O related operations like PLCs (PLCs are general-purpose programs and are described in another training section)
- Calculations are sequenced and synchronized to move execution
- Uses the same flow control logic syntax as a PLC

Procedure for Making the Program

Step 1: Define coordinate system with axis definitions

Step 2: Create opening and closing brackets of the program

Step 3: Select the move mode (**Linear**, **Circle**, **Spline**, **Rapid**, or **PVT**)

Step 4: Select absolute (**abs**) or incremental (**inc**) position programming modes

Step 5: Configure appropriate speed, acceleration, and time settings

Step 6: Program the moves

Step 7: Download the motion program

Step 8: Execute the program from the Terminal Window with **&*m* B*n* R**, where ***m*** is the coordinate system number you defined in Step 1, and *n* is the motion program number you defined in Step 2. Make sure the motors in that C.S. are in closed-loop mode first. If calling from a PLC, can use the **start *m*:*n*** command to start program ***n*** in C.S. ***m***.

Outline of a Motion Program

```
// Step 1: Define Coordinate System (C.S.) and Axis Definitions
undefine all
&1            // Select C.S. #1
#1->1000X     // Assign motor 1 to the X axis w/ 1000 counts per user unit
#2->500Y      // Assign motor 2 to the Y axis w/ 500 counts per user unit

// Step 2: Create opening bracket of motion program
open prog 1   // Opening bracket, defining this as Program 1

// Step 3: Define Move Mode
linear        // Linear move mode

// Step 4: Define Position Programming Mode
abs           // Absolute position programming mode

// Step 5: Define Speed, Acceleration, and Move Time Parameters
TA 125        // 125 ms acceleration time
TS 35         // 35 ms S-Curve time
TM 1000       // 1000 ms move time before deceleration
              // Total move time is TM + TA = 1125 ms
              // Note: Can also use feedrate (F) rather than TM

// Step 6: Program the Moves
X 10 Y 20     // Move X to 10 user units, move Y to 20 user units
close         // Closing bracket
```

Power PMAC Script

All that remains are steps 7 and 8, which are just to download the program and then type **#1J/#2J/ &1 B1 R** in the Terminal Window.

Step 2: Open and Closing Brackets

- **All motion programs must have an opening statement and a closing statement, e.g.:**

```
open prog 1
// Program contents
close
```

Power PMAC Script

- **In Power PMAC, you have the choice of either numbering your motion program with integers (e.g. 1, 2, 3) like above, or with names:**

```
open prog MainProg
// Program contents
close
```

Power PMAC Script

The IDE automatically assigns an internal number corresponding to this named program, starting at 100000. You can use it anywhere when starting (with the **start** or **b** command), calling this program (with the **call** command), or listing the program's contents (with the **list prog** command).

Step 3: Move Mode

- **linear:** Linear interpolated blended moves. Trapezoidal velocity-vs-time profiles. Straight-line path in Cartesian coordinates.
- **pvt**: Moves with specified endpoint position and velocity and specified move time. Uses Hermite-spline path for parabolic velocity-vs-time profiles.
- **circle**: Move in a circular motion with specified center or radius, and end point. Sinusoidal velocity-vs-time profiles.
- **spline**: Move using cubic B-spline interpolator for parabolic velocity-vs-time profiles.
- **rapid**: Move using a PMAC-sequenced jog move. Trapezoidal velocity-vs-time profiles.

Step 4: Position Mode

- **abs**
  Use absolute positioning (i.e. relative to origin of coordinate system)

- **inc**
  Use incremental positioning (i.e. relative to the most recent commanded position)

Step 5: Move Parameters

For Linear and Circle moves, specify:

Acceleration time (**TA**) in ms; optionally specify (final) decel. time (**TD**) in ms
S-Curve time (**TS**) in ms
Move time (**TM**) in ms, or feedrate [user units/**Coord[*x*].FeedTime**] (**F**) if feedrate axis (selected by FRAX command)

For Spline mode, specify:

spline*{data0}* sets all 3 section times to ***{data0}***

spline*{data0}*spline*{data1}* sets section time T0 to ***{data0}***, times T1 & T2 to ***{data1}***

spline*{data0}*spline*{data1}*spline*{data2}* sets T0 to ***{data0}***, T1 to ***{data1}*** , T2 to ***{data2}***

For PVT moves, specify:

Position, velocity, and time (see PVT Mode section of the training)

For Rapid moves, specify:

Motor[*x*].JogTa: if >= 0, Accel. Time [msec]; if < 0, inverse accel. rate [$msec^2$/ct]
Motor[*x*].JogTs: if >=0, S-Curve Time [msec]; if < 0, inverse jerk rate [$msec^3$/ct]
Motor[*x*].RapidSpeedSel : Jog Speed [cts/msec]
=0 (default): **Motor[*x*].MaxSpeed** governs speed
=1: **Motor[*x*].JogSpeed** governs speed

Step 5: Move Parameters

Example: Setting Up a Linear Move

```
linear        // Select linear move mode
abs           // Selects absolute position programming mode

TA 125        // 125 ms acceleration time
TS 35         // 35 ms S-Curve time
TM 1000       // 1000 ms move time from start of move to onset of deceleration
```

Power PMAC Script

Move Parameters Explained

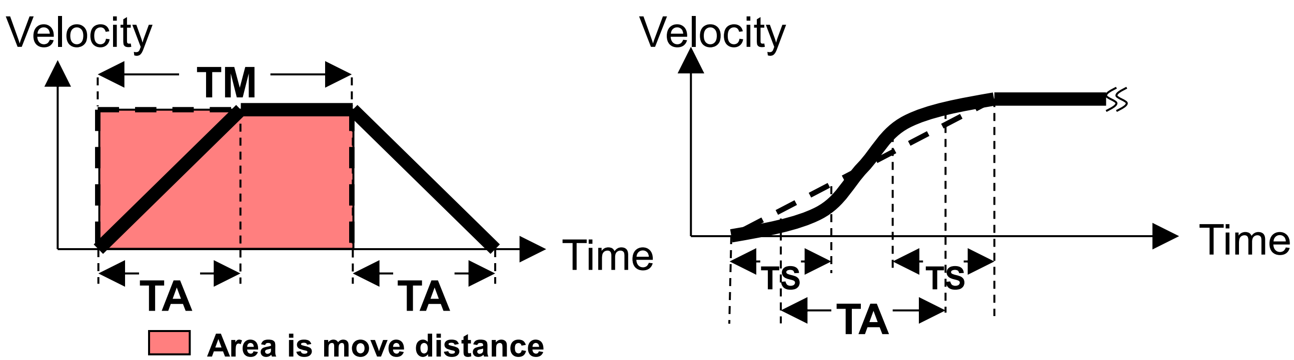
TA: Part of the commanded acceleration time between blended moves (**Linear** and **Circle** mode), and from and to a stop for these moves.

TS: Specifies the time, at both the beginning and end of the total acceleration time, in **Linear** and **Circle** mode blended moves that is spent in S-curve acceleration.

Total acceleration time is **TAT = TA + TS**, in general.

TM: Establishes the time to be taken by subsequent **Linear** and **Circle** mode moves between onset of acceleration and onset of deceleration.

F: Sets the commanded velocity for upcoming **Linear** and **Circle** mode blended moves [(user length units)/**Coord[*x*].FeedTime**]



The effect of each of these commands on each Move Mode will be described more in detail in each subsequent Move Mode section of the training.

Note

Feedrate Command

- **Commands**

  F {*velocity*} // Feedrate (top speed during a move) definition

  Frax(Axes) // Vector feedrate axes definition

- **F{*velocity*} specifies velocity for feedrate axes (tool tip)**

  Velocity unit: (User distance unit / User time unit)

  User distance unit: Defined in Axes Definition

  User time unit: Defined in **Coord[*x*].FeedTime**, C.S. ***x*** feedrate time unit, msec

When using **F**, **TM** is dictated by the following formula:

$$TM = \frac{\text{Total Distance}}{F} - TAT$$, where TAT is the total accel. time.

When defining **TM** instead of **F**, the top speed F_{max} for the move is given as:

$$F_{max} = \frac{\text{Distance at Constant Velocity}}{TM}$$

Here, F_{max} is computed a bit differently than when specifying **F**, since it uses just the constant velocity distance, not total distance, of the move.

Example:

If user distance unit is in inches, and **Coord[*x*].FeedTime**=1000 (default), then **F 5** means setting tool tip move speed as 5 inches/sec

Note

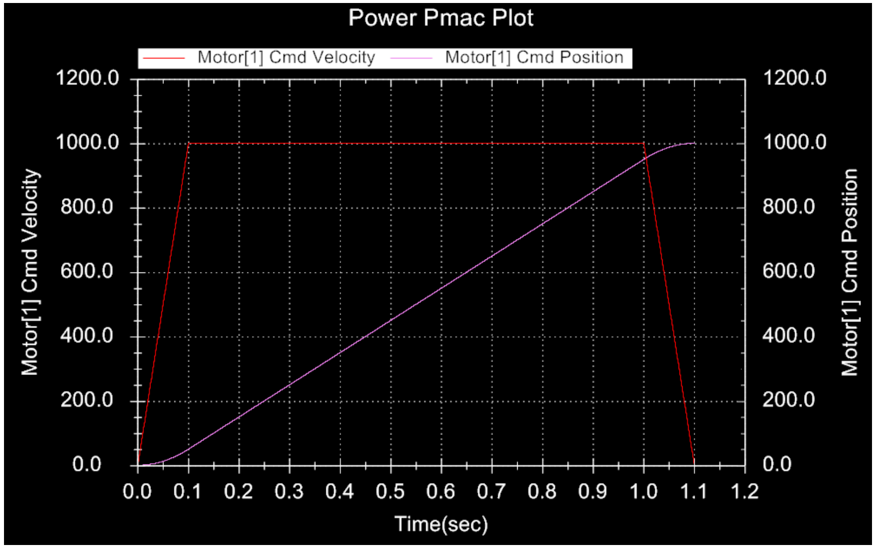
Move time of a move statement can be defined by `F` or `TM`. Either one will reset previous move time definition.

F vs. TM

```
open prog UsingFProg
linear inc
ta 100 ts 0 F 1
dwell 0 Gather.Enable=2 dwell 0
x 1
dwell 0 Gather.Enable = 0 dwell 0
close
```

Power PMAC Script

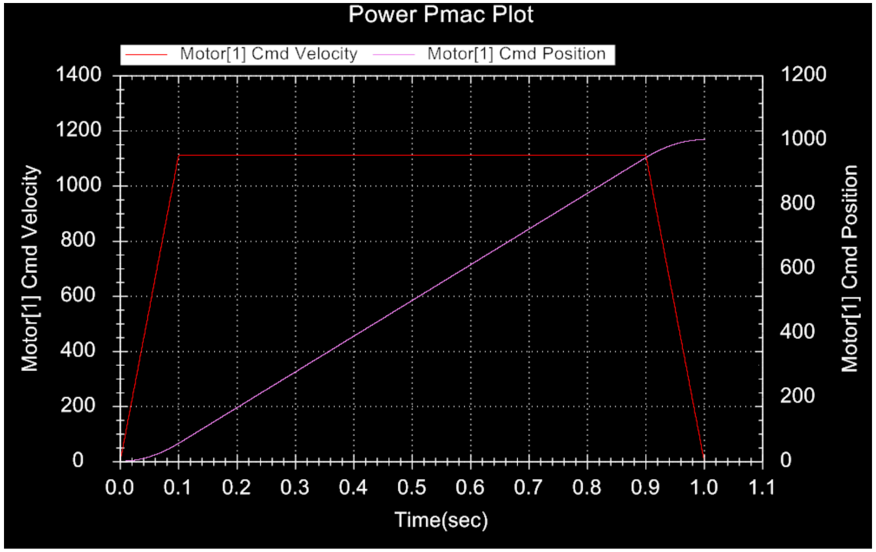
Top speed is indicated by F: 1 unit/sec




```
open prog UsingTMProg
linear inc
ta 100 ts 0 tm 900
dwell 0 Gather.Enable=2 dwell 0
x 1
dwell 0 Gather.Enable = 0 dwell 0
close
```

Power PMAC Script

Top speed is indicated by the distance and move time settings: $1.1\overline{1}$ unit/sec



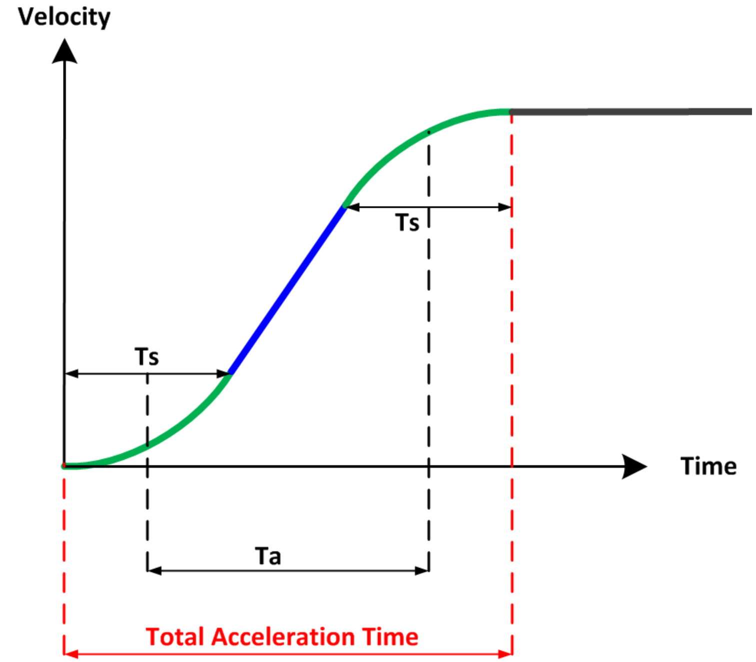
Move Time Commands

If Ta >= Ts
Total Accel Time = Ta + Ts

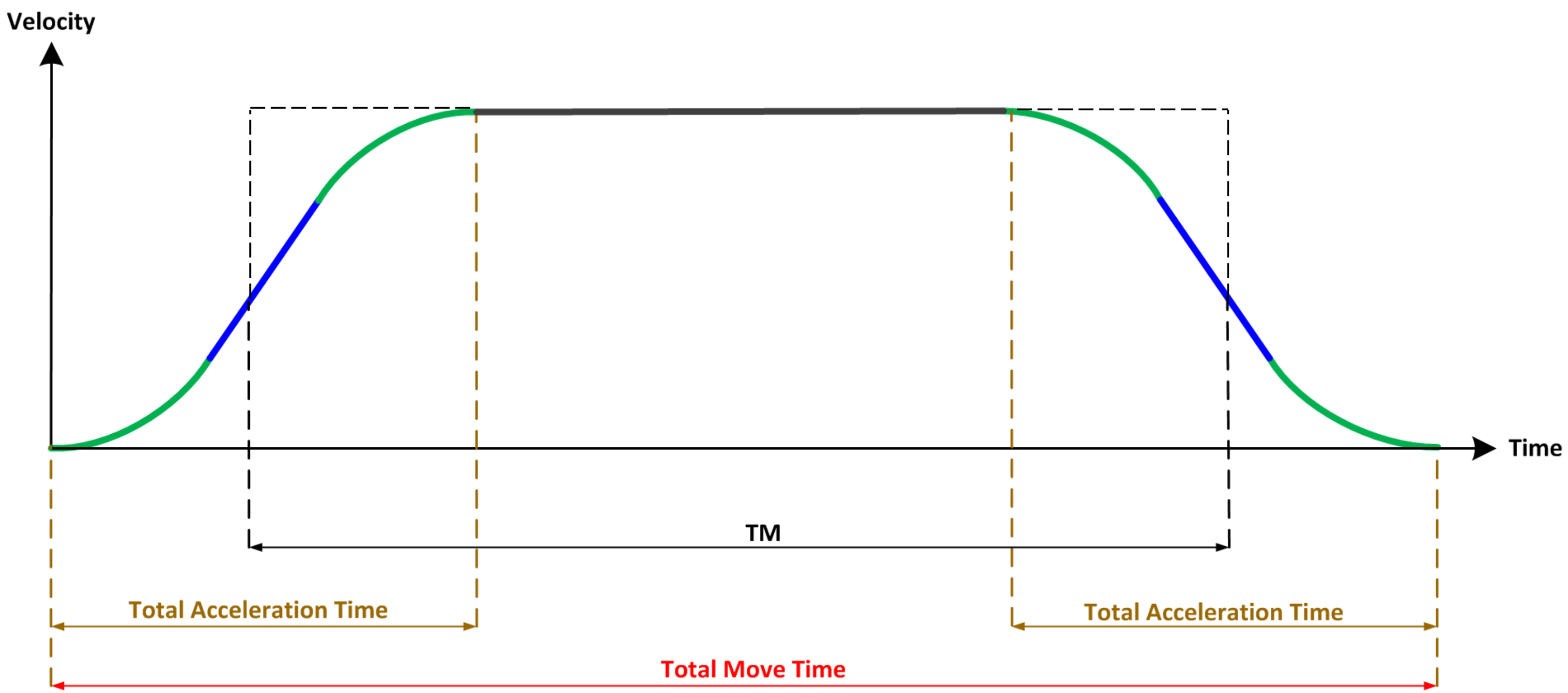
If Ta < Ts
Total Accel Time = 2 * Ts

If Td >= Ts
Total Decel Time = Td + Ts

If Td < Ts
Total Decel Time = 2 * Ts



Move Time Commands



If TM >= Total Accel Time (TAT)
Total Move Time = TM + TAT

If TM < Total Accel Time
Total Move Time = 2 * TAT

Feedrate Command

- **frax(*Axes*) specifies which axes are in feedrate calculation**
  - When multiple axes are involved in a move, such as a tool tip in an XYZ Cartesian coordinate system, the distance calculation needs to be specified as a vector length for the move time calculation
  - Any non-feedrate axis move statement(s) **on the same line** as the feedrate axes' move statement(s) will complete in the same amount of time

Example:

frax(X,Y,Z) (default) means distance is calculated from Axes X, Y, and Z

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

Vector Feedrate Axes Example

```
inc                 // Incremental Move
frax (X,Y)          // Feedrate Axes [X,Y]
X3 Y4 F10           // Move distance X=3 Y=4, with speed 10 unit/sec
```

Power PMAC Script

Velocity calculation

$$\text{Distance} = \sqrt{3^2 + 4^2} = 5;\ \text{Move Time} = \frac{5}{10} = 0.5 \text{ sec}$$

$$V_x = \frac{3}{0.5} = 6 \text{ unit/sec};\ V_y = \frac{4}{0.5} = 8 \text{ unit/sec}$$

```
inc                 // Incremental Move
frax (X,Y)          // Feedrate Axes [X,Y]
X3 Y4 Z12 F10       // Move distance X=3 Y=4, with speed 10 unit/sec,and
                    // Z=12
```

Power PMAC Script

Velocity calculation

$$\text{Distance} = \sqrt{3^2 + 4^2} = 5;\ \text{Move Time} = \frac{5}{10} = 0.5 \text{ sec}$$

$$V_x = \frac{3}{0.5} = 6 \text{ unit/sec};\ V_y = \frac{4}{0.5} = 8 \text{ unit/sec};\ V_z = \frac{12}{0.5} = 24 \text{ unit/sec}$$

```
inc                 // Incremental Move
frax (X,Y,Z)        // Feedrate Axes [X,Y,Z]
X3 Y4 Z12 F10    // Move distance X=3 Y=4 Z=12, with speed 10 unit/sec
```

Power PMAC Script

Velocity calculation

$$\text{Distance} = \sqrt{3^2 + 4^2 + 12^2} = 13;\ \text{Move Time} = \frac{13}{10} = 1.3 \text{ sec}$$

$$V_x = \frac{3}{1.3} = 2.31 \text{ unit/sec};\ V_y = \frac{4}{1.3} = 3.08 \text{ unit/sec};\ V_z = \frac{12}{1.3} = 9.23 \text{ unit/sec}$$

Step 6: Programming the Move

- **There are 32 axis names which can be used per Coordinate System:**
  A, B, C, X, Y, Z, U, V, W (I, J, K, and N not permitted)
  AA, BB, CC, …, XX, YY, ZZ (except II, JJ, and KK)

- **To command an axis to move, just write the axis letter and then either a numeric literal immediately thereafter or a parentheses with a numerical statement therein; e.g.:**

```
X 10 // Move the X-axis 10 user units
Y(Sin(MyGlobalVar)) // Move the Y-axis Sin(MyGlobalVar) user units
```

Power PMAC Script

- **Can command several moves simultaneously by writing them on the same line; e.g.:**

```
U30 V40 W10 // Command U to move 30 user units, V 40 units, and W 10 units
```

Power PMAC Script

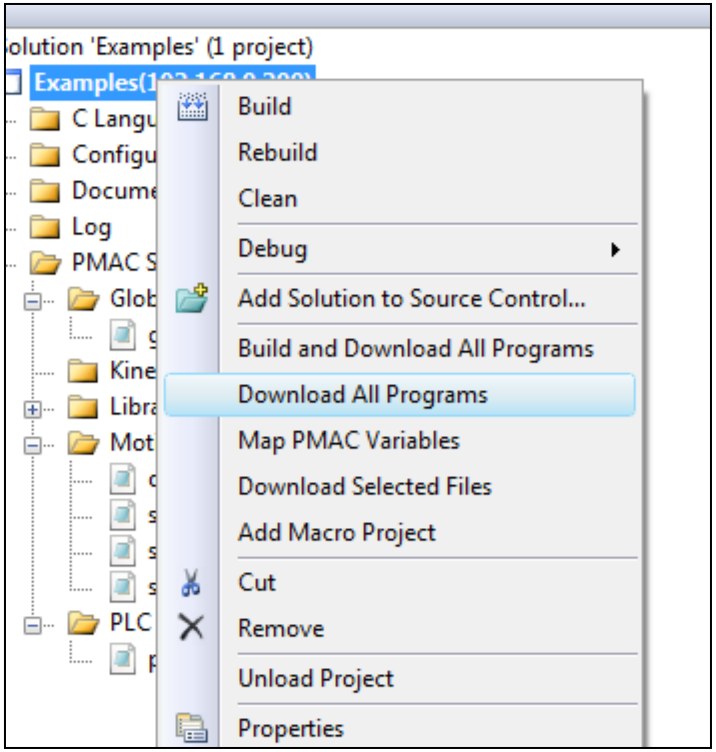
- **Omit parentheses for numeric literals; this is more computationally efficient than using parentheses. Parentheses are only required for computations, not for numeric literals.**

Note

This example assumes that MyGlobalVar is a previously defined variable.

Steps 7 - 8: Downloading and Running

- **In the Power PMAC IDE, right-click on your Project name and then click "Build and Download" to download the program to PMAC:**



- **Finally, type &*m* Bn R in the Terminal Window to start program *n* in coordinate system *m*. Note that all motors in the coordinate system must be in closed-loop mode before running the program**

  One can easily put a motor into closed loop mode from the Terminal Window with **#*x*J**/, where ***x*** is the motor number.

delay

- **delay*{data}***
  - Waits the duration ***{data}*** in milliseconds
  - If delay comes after a blended move, the TA deceleration time from the move occurs within the delay time, not before it
  - If the specified delay time is less than the acceleration time currently in force (**TA** or 2***TS**), the entire delay will occur during the acceleration, effectively not occurring at all
  - The actual time for delay does vary with a changing time base (current % value, from whatever source)
  - PMAC precomputes upcoming moves (and the lines preceding them) during a delay

- **Example:**

```
delay 1000 // Delay 1000 msec before continuing motion program
MyGlobalVar=35
delay(MyGlobalVar+45) // Delay 80 msec before continuing program
```

Power PMAC Script

Note

The Delay command will not cause loss of synchronicity with the master signal when using external time base.
This example assumes that MyGlobalVar is a previously defined variable.

dwell

- **dwell{*data*}**
  - Waits the duration ***{data}*** in milliseconds
  - If the previous servo command was a blended move, there will be a **TA** time deceleration to a stop before the dwell time starts
  - **dwell** is not sensitive to a varying time base – it always operates in real time (as defined by **Sys.ServoPeriod**)
  - Power PMAC does not precompute upcoming moves (and the program lines before them) during the **dwell**; it waits until after it is done to start further calculations upon the next servo cycle

- **Example:**

```
dwell 1000 // Dwell 1000 msec before continuing motion program
MyGlobalVar=10
dwell(MyGlobalVar*5) // Dwell 50 msec before continuing program
```

Power PMAC Script

Caution

Use of any Dwell command, even a Dwell 0, while in external time base will cause a loss of synchronicity with the master signal. This example assumes that MyGlobalVar is a previously defined variable.

while

- **while(*condition*){*contents*}**
  - Performs {***contents***} until ***condition*** goes false
  - Logical condition syntax is C-like
  - Leave {***contents***} blank to wait without performing additional actions
  - If {***contents***} occupies only a single statement, its surrounding brackets ({ and }) may be omitted

- **Example:**

```
while(Input1 == 0) {} // Pause here until Machine Input 1 goes high
while(Input2 == 1)
{
		Counter++; // Increment Counter while Input2 is 1
}
```

Power PMAC Script

Note

Waiting in an empty loop will not cause loss of synchronicity with a master signal.
This example assumes that Input1, Input2, and Counter are previously defined variables.

if

- **if(*condition*){*contents1*} else {*contents2*}**
  - Performs {***contents1***} if ***condition*** is true; otherwise, performs {***contents2***}.
  - **else** clause is optional.
  - Logical condition syntax is C-like.
  - If {***contents1***} or {***contents2***} occupy only a single statement, their surrounding brackets ({ and }) may be omitted.

- **Example:**

```
if(Input1 == 0) // If Machine Input 1 is low
{
		Output1 = 0; // Set Output 1 low
} else
{
		Output1 = 1; // Set Output 1 high
}
```

Power PMAC Script

Note

The above example assumes that Input1 and Output1 are previously defined variables.

switch

- **switch(*Variable*){*contents*}**
  - Compares ***Variable*** to a number of distinct, integer (ONLY) states and takes actions for each value. Syntax is C-like.
  - If ***Variable*** matches one of the states listed, that branch of code is executed.
  - **break** prevents code execution from passing to subsequent states; omit **break** if the program should continue to subsequent branches.
  - The **default** branch of code (see below) executes if ***Variable*** does not match any specified states.
- **Example:**

```
switch(MachineState)
{
		case 0:
				// action1
				break;

		case 1:
				// action2
				break;

		default:
				// action3
				break;
}
```

Power PMAC Script

This example assumes that MachineState is a previously defined variable.

Note

Jump-Back Rule

PMAC will not blend through subsequent moves if it encounters a number of jumps back greater than (Coord[*x*].GoBack + 1) (by default, 2 jumps back):

```
global MyVar(3);
open prog jb1
MyVar(0)=1;
while (MyVar(0)<11){
  MyVar(1)=0;
   while (MyVar(1)<360){
    MyVar(2)=10+MyVar(0)*Cosd(MyVar(1));
    X(MyVar(2));
    MyVar(1)++;
   }
  MyVar(0)++;
}
close
```

Power PMAC Script

Blending stops each time inner loop is exited: two ending braces (}) encountered before next move.

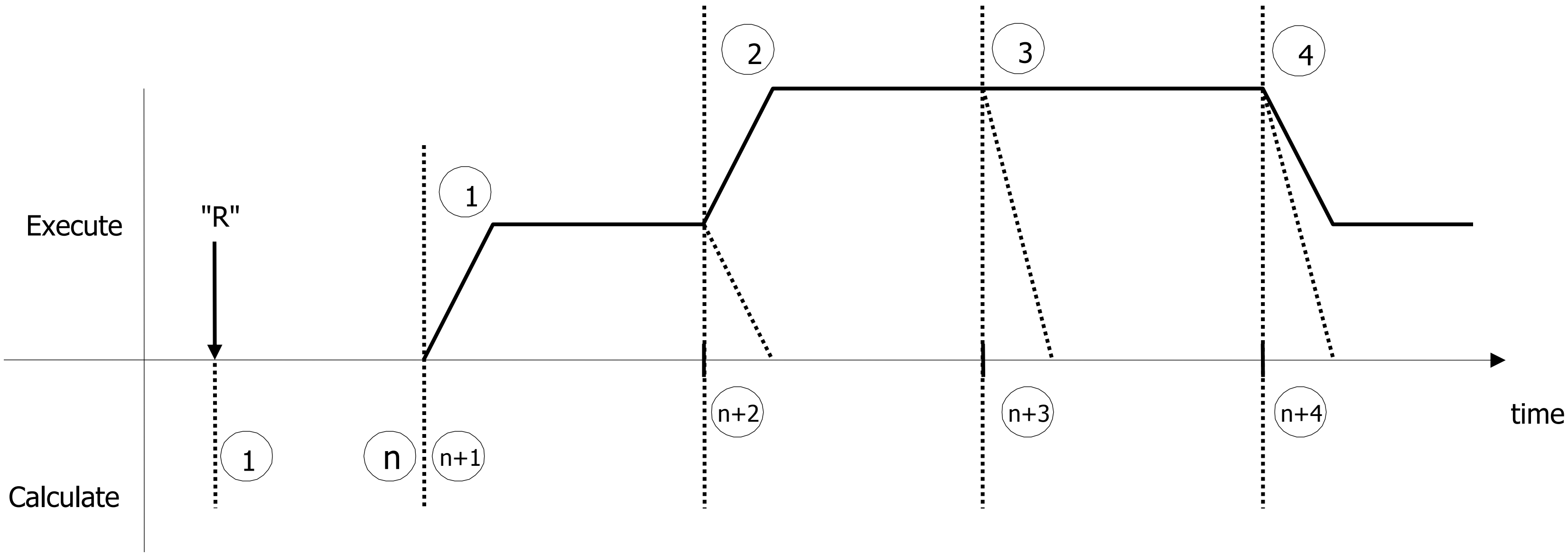
```
open prog jb2
MyVar(0)=1;
while (MyVar(0)<11){
            MyVar(1)=0;
            while (MyVar(1)<359){

            MyVar(2)=10+MyVar(0)*Cosd(MyVar(1))
            X(MyVar(2))
            MyVar(2)++;
  }
 MyVar(2)=10+MyVar(0)*Cosd(MyVar(1))
 X(MyVar(2))
 MyVar(0)++;
}
close
```

Power PMAC Script

Putting a move between the two ending braces (}), or setting **Coord[*x*].GoBack** > 0, makes blending continuous throughout entire example.

Motion Program Precalculation

Power PMAC plans “n” moves ahead for Blended Moves



How big is “n”?

- 0 for Rapid moves, Dwell between moves, or if **Coord[*x*].NoBlend** = 1 in non-Rapid modes
- 1 if segmentation is on (**Coord[*x*].SegMoveTime** > 0) and **Coord[x].LHDistance** = 0
- 1 for basic blending without acceleration control and **Coord[*x*].SegMoveTime** = 0
- 2 if segmentation is off (**Coord[*x*].SegMoveTime** = 0) and acceleration limits enabled
- As large as necessary when using Special Lookahead to keep Lookahead buffer full
- Enough moves for intersection/interference-check calculations when using 2D cutter comp.

Synchronous Variable Assignment

- **Because of how PMAC performs Lookahead for numerical calculations that are not necessarily related to moves, normal variable assignments may be executed before the user expects**
- **To force variable assignments to occur at the beginning of the next move, use a synchronous variable**
- **Just like a normal variable assignment, but with == rather than = in the assignment expression**
- **Can be used for `global`, `csglobal`, or `ptr` variables (except self-assigned `ptr` variables)**
- **Number of assignments limited by Coord[*x*].SyncOps (8192 by default)**
- **Example:**

```
ptr Output1->GateIo[0].DataReg[3].0.1   // Pointer to 1st I/O Card, Output 1
ptr Output2->GateIo[0].DataReg[3].1.1   // Pointer to 1st I/O Card, Output 2
global MyGlobal
open prog 3
linear abs TA300 TM1500 TS150 // Define motion parameters
Output1==1   // Machine output 1 will go high just as the X 30 move starts
X30          // Move X-axis to 30 user units
Output1==0   // Machine output 1 will go low just as the Y 40 move starts
Output2==1   // Machine output 2 will go high just as the Y 40 move starts
MyGlobal==10 // Set a global variable synchronously
Y40
Output2==0   // Machine output 2 will go low as the program finishes
dwell 0      // This dwell 0 is necessary to force Output2==0 to occur
             // (dwell 0 acts like a sequenced move here, forcing Output2==0 to occur)
close
```

Power PMAC Script

Simple Move Example

```
/************* Setup and Definitions **************/
undefine all
&1              // Coordinate System 1
#1->1000x       // Assign motor 1 to the X-axis - 1 program unit
                // of X is 1000 encoder counts of motor #1

/************* Motion Program Text *****************/
open prog 1               // Open buffer for program entry, Program #1
linear                    // Blended linear interpolation move mode
abs                       // Absolute mode - moves specified by position
TA500                     // Set 1/2 sec (500 msec) acceleration time
TS0                       // Set no S-curve acceleration time
F5                        // Set feedrate (speed) of 5 units/sec
X10                       // Move X-axis to position 10
dwell 500                 // Stay in position for 1/2 sec (500 msec)
X0                        // Move X-axis to position 0
close                     // Close buffer - end of program
```

Power PMAC Script

To run this program, type this in the Terminal Window:

```
#1J/ &1 B1 R // Close loop, C.S. #1, point to Beginning of Program 1, Run
```

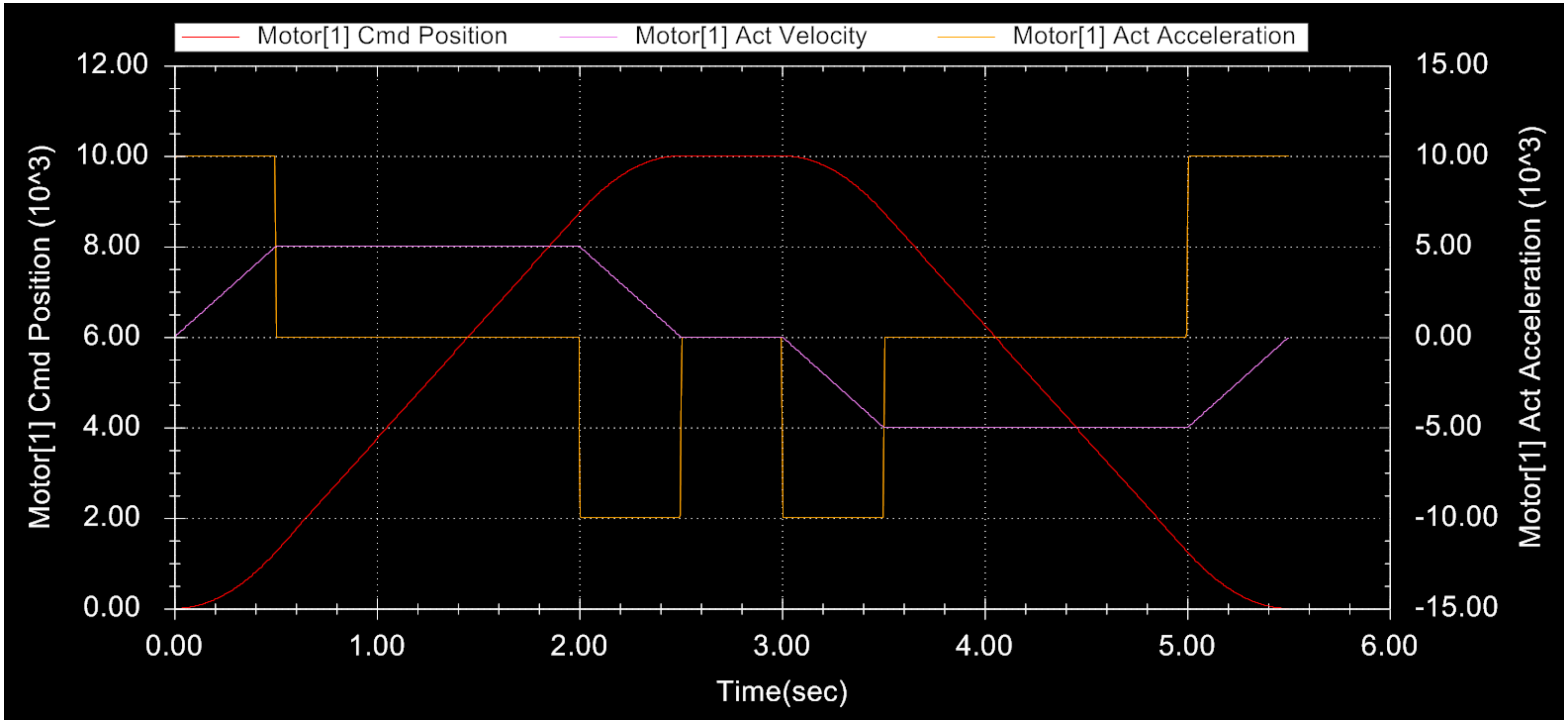
Power PMAC Script

To run from a PLC program:

```
jog/ 1 start 1:1; // Close loop, C.S. #1, point to Beginning of Program 1, Run
```

Power PMAC Script

Simple Move Example



A More Complex Move Example

```
//*************** Setup and Definitions ****************//
undefine all
&2              // Coordinate system 2
#2->1000X       // 1 unit of X is 1000 counts of motor 2
//****************** Motion Program Text ****************//
open prog 2 // Open buffer for entry
local ctr;
linear          // Blended linear interpolation move mode
inc             // Incremental mode - moves specified by distance
TA500           // 1/2 sec (500 msec) acceleration time
TS250           // 1/4 sec in each half of S-curve
ctr = 0;        // Initialize a loop counter variable
while (ctr<3){ // Loop until condition is false (3 times)
  X10           // Move X-axis 10 units (= 10,000 cts) positive
  dwell 500     // Hold position for 1/2 sec
  X-10          // Move X-axis back 10 units negative
  dwell 500     // Hold position for 1/2 sec
  ctr++;        // Increment loop counter
}               // End of loop
close           // Close buffer - end of program
```

Power PMAC Script

To run this program, type this in the Terminal Window:

```
#2J/ &2 B2 R // Close loop, C.S. 2, point to Beginning of Program 2, Run
```

Power PMAC Script

A More Complex Move Example