AGILE SYSTEMS
575 Kumpf Drive
Waterloo, ON
Canada N2V 1K3

T [519] 886-2000
F [519] 886-2075
E info@agile-systems.com

Technical Brief

Brief Number: **TB-002-11-06**
Reference: PVT, PT trajectory
Application: **MAX3000 and microMAX Products**
Revision: 1
Date: Nov 09, 2006

Overview This technical brief outlines the PVT, and PT trajectory mode

Example See Two_Axis_PVT_Example.zip on the MAXCLib motion Library CD

Specification NA

Image NA

Description

PTP, PVT and PT trajectory modes

There are three methods of specifying a motion profile for the MAX controller.

- PTP – P(oint) t(o) p(oint) motion, where the user specifies the position endpoint, velocity, acceleration and jerk. The MAX uses this information to calculate the trajectory.
- PVT mode - P(osition)V(elocity)T(ime) where the relative position setpoints and velocity setpoints are specified at defined times for full control of the motion profile.
- PT – P(osition), T(ime) same as the PVT mode, however the user only specifies relative position setpoints at defined times and the MAX controller cacluate the appropriate veolocity setpoints.

The MAX controller supports move commands that specify an incremental position to which to move.

In cases where the path of multiple axes needs to be coordinated, or where moves are generated on-the-fly, the MAX controller supports the ability to specify that an axis be at position X at time T, and optionally, to be at position X and moving at velocity V at time T. This specifying of a motion profile using position, velocity, versus time is called the PVT interface.

The support for PVT is extremely flexible and capable, allowing it to be used in a variety of machines (tightly coordinated, loosely coordinated) with a variety of hosts (OS, MIPS) on a variety of networks (fast, time-stamped, slow, asynchronous). To use the PVT feature in your system, you will need to know some fundamentals of Agile's PVT. You should also

AGILE SYSTEMS
575 Kumpf Drive
Waterloo, ON
Canada N2V 1K3
T [519] 886-2000
F [519] 886-2075
E info@agile-systems.com

understand Agile Systems Inc.'s implementation of PVT so you can calculate the information you will need to supply the controller.

Agile's PVT Fundamentals

The PVT (Position, Velocity, Time) interface on the MAX provides the ability for users to specify complex motions on multiple coordinated axes with few points and generate these motions as the motion is proceeding. It also provides the capability to load a motion and repeat this motion many times. Figure 5 describes the PVT functionality.

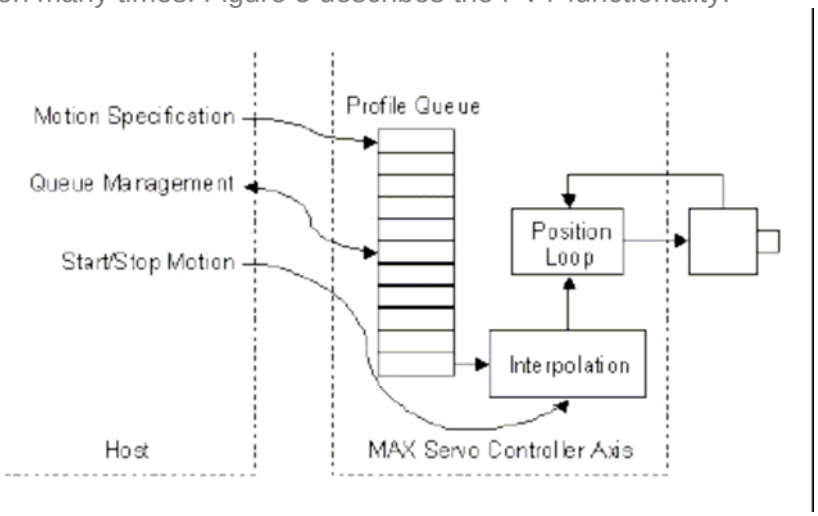

Figure 5. PVT functionality on microMAX.

The MAX Servo controller provides PVT capability through a simple command interface. Points are specified to a profile queue in PV pairs. The profile queue provides in excess of 667 packets to be stored on the controller. The are the related PVT/PT registers abd MAXClib commands.

PVT/PT Registers

- Profile_Arm
- Profile L*ower*_Threshold
- Profile_Upper_Threshold

PVT/PT MAXCLib functions

- ProfileDeltaTimeRegister()
- ProfileDeltaTimeValue()
- ProfileEnd()
- ProfileFlush()
- ProfilePositionRegister()
- ProfilePositionValue()
- ProfileSetRegisterRegister()

AGILE SYSTEMS
575 Kumpf Drive
Waterloo, ON
Canada N2V 1K3
T [519] 886-2000
F [519] 886-2075
E info@agile-systems.com

- ProfileSetRegisterValue()
- ProfileSplinePositionRegister()
- ProfileSplinePositionValue()
- ProfileStartArmed()
- ProfileVelocityRegister()
- ProfileVelocityValue()

Note: For more details on the following Registers, please refer to the RegisterReference.chm file in DPWin.

Note: Fore more details on the following MAXClib functions please refer to the CommandReference.chm file in DPWin.

The queue can contain a motion profile to be executed multiple times. These PV points are spaced apart in regular, adjustable intervals. Between these points, the interpolator generates points using 3rd order interpolation.

The minimum Delta Time is 2ms.

AGILE SYSTEMS
575 Kumpf Drive
Waterloo, ON
Canada N2V 1K3
T [519] 886-2000
F [519] 886-2075
E info@agile-systems.com

PVT Mathematics

Interpolation

The MAX controller interpolates intermediate PVT points from the user PVT points using 3rd order interpolation mathematics. The general problem is to specify a motion profile that is reproduced smoothly by the motion controller.

The user provides desired PVT points (position and velocity) at periodic intervals. Given these pairs pairs, the MAX controller calculates coefficients to allow smooth interpolation of position, velocity, and acceleration at much shorter time intervals than the desired PVT pairs.

For example:

Many accurate motion profiles can be described by PVT pairs generated every 2 to 50ms by the host computer (application software), but to achieve smooth operation, The MAX controller interpolates position, velocity, and acceleration setpoints every 250us. The interpolation math is as follows.

At the beginning of an interpolation interval, the time is t0, and the position-velocity (PV) set point is (P0, V0). At the end of the interpolation interval, the time is t0 + T, and the PV set point is (PT, VT).

Within this interval, the interpolated position is:
$P(t) = a(t – t0)3 + b(t – t0)2 + c(t – t0) + d$

Velocity is the derivative of position:
$V(t) = 3a(t – t0)2 + 2b(t – t0) + c$

Acceleration is the derivative of velocity:
$A(t) = 6a(t – t0) + 2b$

MAX calculates the coefficients a, b, c, d for each interval by solving the four boundary condition equations:

1. $P(t0) = P0$, Þ $d = P0$
2. $V(t0) = V0$, Þ $c = V0$
3. $P(t0 + T) = PT$, Þ $PT = aT3 + bT2 + cT + d$
4. $V(t0 + T) = VT$, Þ $PT = 3aT2 + 2bT + c$

Note that this approach gives us 3rd order interpolation of position, 2nd order interpolation of velocity, and 1st order interpolation of acceleration for points inside the segment.

If we have a series of PV pairs with poorly calculated V setpoints, we can generate large discontinuities between acceleration setpoints moving from one segment to the next. In other words, bad setpoints in gives bad motion out.

As an example, consider the following data set where data points are space 1ms apart and V set point data is approximately 10-20% higher than the ideal number. The PV

AGILE SYSTEMS
575 Kumpf Drive
Waterloo, ON
Canada N2V 1K3
T [519] 886-2000
F [519] 886-2075
E info@agile-systems.com

interpolation algorithm will generate a solution for this, however intuitively we see that we are asking for a motion that is on average faster than that needed to do the job.

	PV Pair		
T (ms)	P set	V set	V(T) - Ideal
10	500	169	140
11	656	198	168
12	842	227	197
13	1056	256	227
14	1300	283	255
15	1570	309	282
16	1867	333	308
17	2189	355	332
18	2534	376	354
19	2901	395	375

The PV coefficient calculator will generate a, b, c, d coefficients to give 3rd order interpolation which meets these requirements. On a machine, the resultant motion is not smooth; rather it suffers from a large vibration at the frequency 1/T!

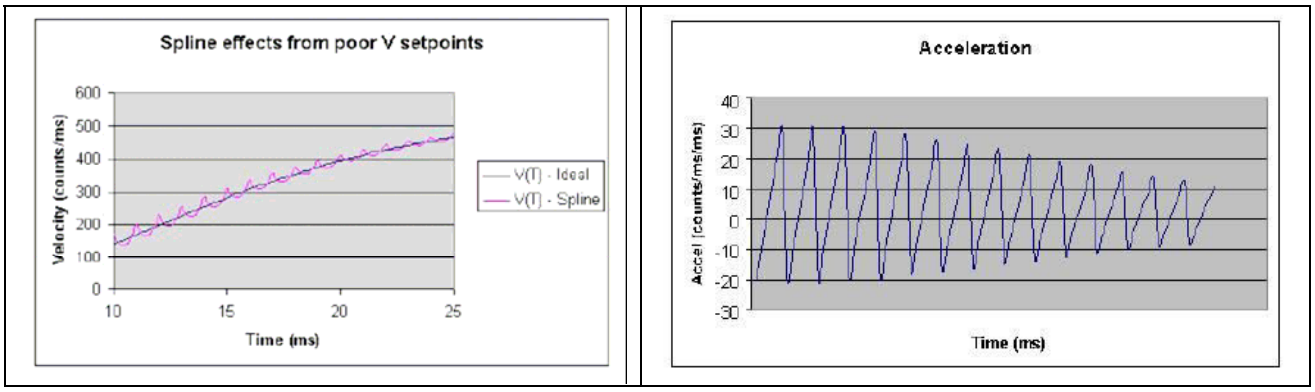

Figure 7. Motion artifacts introduced by poorly generated velocity setpoints

The point here is that poorly generated velocity setpoints can create unwanted artifacts in the motion profile. Ideally, velocity setpoints are generated so that the piecewise linear

AGILE SYSTEMS
575 Kumpf Drive
Waterloo, ON
Canada N2V 1K3
T [519] 886-2000
F [519] 886-2075
E info@agile-systems.com

acceleration segments are continuous at segment boundaries (in other words, there is no jerk at boundaries).

Note: If we achieve this condition, the current setpoints (magnitude) to the servomotors vary linearly between PV pairs and do not change abruptly from segment to segment resulting in smooth motion. This type of artifact can also happen during very slow velocity moves due to quantization effects in the position and velocity setpoints.

The MAX controller uses fractional representation for position and velocity PVT points during spline operations. If the application software generates slow velocity profiles where 1 or 2 bits of position information is important, motion smoothness may be improved by adding fractional position information to position setpoints.

AGILE SYSTEMS
575 Kumpf Drive
Waterloo, ON
Canada N2V 1K3
T [519] 886-2000
F [519] 886-2075
E info@agile-systems.com

Effective Use of PVT

Overview

An effective method of using PVT is discussed. For effective use, the number of points on the controller must be kept to a minimum for increased responsiveness while eliminating the risk of starving the PVT queue on the controller.

The PVT Queue

To understand how we will use the PVT features to optimize the PVT performance, read the background on PVT. The queue structure is outlined briefly in figure 8.

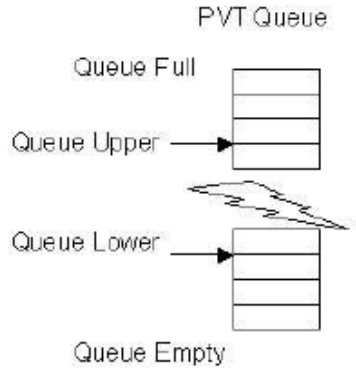

Figure 8. PVT Queue structure

The PVT Queue can hold 667 packets of information. Packets do not correspond directly to PVT points however.

MaxCLib and PVT

Each of the PVT commands in the MAXCLib motion library returns an integer that contains the status of the PVT queue. In particular, the status of the Queue Upper Threshold (QUT) and Queue Lower Threshold (QLT) pointers are contained in the return argument.

It is possible to use these pointers in a couple of ways.

1. If the user requires tight control of the trajectory and needs to make changes on the “Fly”, the host application can be written for “real time” performance. The host application can calculate and send part of the trajectory to the controller and then continuously queried the PVT queues.

   When the PVT queues are at the minimum threshold the host application can send more PVT points. The minimum queue size is 3 points and the minimum time between PVT points is 2ms, therefore the controller needs a minimum of 6 ms of trajectory information to operating correctly.

   This method of controlling the PVT queue may not be desirable for a couple reasons:

AGILE SYSTEMS
575 Kumpf Drive
Waterloo, ON
Canada N2V 1K3
T [519] 886-2000
F [519] 886-2075
E info@agile-systems.com

a. The amount of network traffic for continuous polling of the query is high. In these situations, it is possible to have the controller send the queue status to the host PC to minimize the amount of network traffic.

b. To operate at a 2ms of PVT, may require the user to use a real time operating system for the host PC. However 2 ms PVT point and 3 points are the minimum specification, applications using a PVT time of 100ms and 100 points have also produced excellent trajectory results.

2. If the user does not require “real time” control of the trajectory, the trajectory can be pre-calculated on the host PC. The user can load the trajectory on the controller and have the host PC or a standalone application execute at any time.

The Queue Threshold Zones

With the two threshold pointers, the PVT queue can be divided into 3 zones. Each zone is differentiated by its probability of queue starvation. The first zone is indicated when the number of points is below the QLT (in red). In this zone, points should be added to the PVT queue until the number of points is greater than the QLT.

In this zone, the number of points is critically low. If the PVT thread loses context while the queue is in this state, there is no guarantee that we will get context back before the queue runs out of points. The second zone is indicated when the number of points is above the QLT, but below the QUT (in green). In this zone, it is not fatal if we lose context at any time.

This is because the number of points in the queue multiplied by DeltaT is longer than the maximum process time. The PVT process will be given context again before the queue underflows. However, note that when the PVT process is given context, the state of the queue could be in zone 1.
The third zone is indicated when the number of points in the queue exceeds both the QLT and QUT (in orange). In this zone, we have more than a desirable number of points. While there is no risk of queue underflow, the condition is not desirable from a dynamic motion perspective.

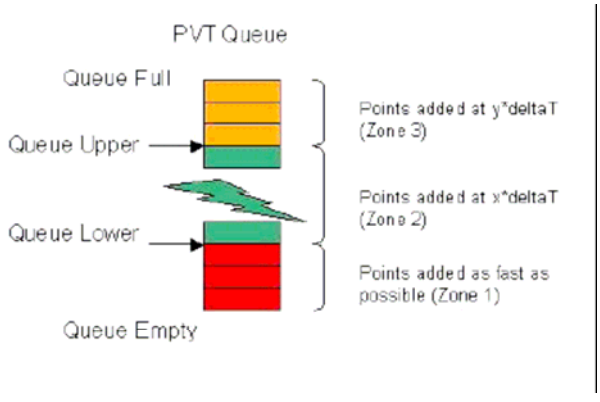

Figure 9. PVT Queue threshold zones

The Process Time

AGILE SYSTEMS
575 Kumpf Drive
Waterloo, ON
Canada N2V 1K3
T [519] 886-2000
F [519] 886-2075
E info@agile-systems.com

Now that we have established three zones of 'criticalness' for the PVT queue, a method for controlling the process time must be established. In some cases, the host operating system controls the minimum process time. In these cases we must use a time that is longer than the minimum process time of the operating system.

Using the Sleep timer or other similar method, we can loosely control the amount of time between each successive context of the PVT process. The method assumes a reasonable amount of confidence in the process timing mechanism. For instance, if the known maximum context switch of the operating system is 1ms and the Sleep timer is set for 100ms, then it can be assumed that the longest amount of time the PVT process will sleep for is 101ms.

The Sleep timer is configured each time the thread should sleep. Here the PVT process should implement the x and y multipliers shown in the previous figure.

- If the queue is in zone 1, points will be added as quickly as possible. No timer configuration is necessary.
- If the queue is in zone 2, points will be added at the rate of x multiplied by deltaT. If x were 1, then in a perfect system points would be added to the end of the queue as fast as they are taken out. Selection of x is discussed in the next section.
- If the queue is in zone 3, points will be added at the rate of y multiplied by deltaT. A y greater than 1 ensures that points will enter the queue at slower rate than they are taken out. The net result is a shrinking queue.

Note that through this discussion it has been assumed that transmission of the points is not a significant factor in the rate points can be added to the queue.

Each packet in a PVT command is 10 bytes. Keep an eye on the bit rate of the transmission medium when deciding the QLT and QUT for the system.
Determining x, y, deltaT and QLT

There are no rules or equations for determining x and y. These parameters are greatly affected by the timer accuracy. If the timer is very reliable (as in a RTOS), x can be slightly greater than 1. This has the effect of keeping the number of points just above the QLT. If timer accuracy is suspect, then x should be slightly less than 1. This has the effect of raising the number of quiescent points in the queue to just below the QUT. Select y such that the amount of time will not cause the queue to be starved. In fact, y should be selected such that after waiting y*deltaT, the number of points in the queue is between QLT and QUT.

The equation to relate deltaT, QLT, and process time is:

QLT = Process Time / deltaT.

Note that deltaT must be greater than or equal to 2ms.

Transmission Rate

This section will outline significant factors in communication speed. In systems where the queue size will be minimized, transmission delays can be significant. It is difficult to apply

AGILE SYSTEMS
575 Kumpf Drive
Waterloo, ON
Canada N2V 1K3
T [519] 886-2000
F [519] 886-2075
E info@agile-systems.com

numbers to host delays, as they are highly dependant on host capabilities. The known delays are presented.

When sending a packet to the MAX controller, it is important to understand Host and transmission timing and delays. There is a total of 4 significant delays when transmitting or receiving data from a host. There is 1 delay on the controller itself.

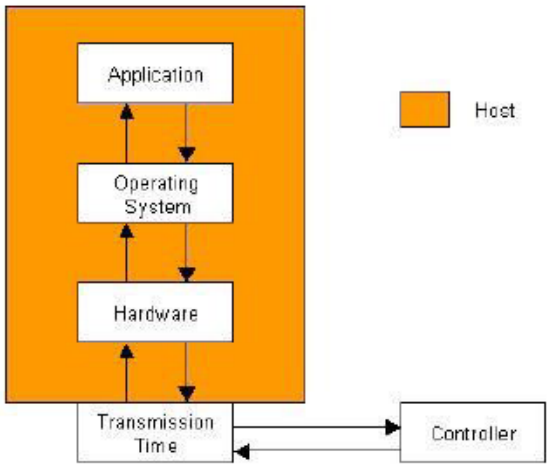

Figure 10. Transmission path with delays.

Figure 10 shows transmission path with delays. Each arrow represents a transmission path and each box represents a delay. Of the 3 host delays, Operating System and Application delays are highly dependant on operating system and host processing capacity. Controller delays are 100us in the worst case, and Transmission Time can be calculated using the transmission rate and the number of bytes per packet (10 bytes for RS232, RS422, RS485). For every packet, a response must be received. Therefore the minimum amount of time required between transmits is:

100us + 2 x (Transmission Time + Hardware + Operating System + Application).

The round trip requirement exists for multi-drop networks as well. For instance, if two controllers are being synchronized with PVT where packets are sent to each node in an alternating fashion, the minimum time between packets to the same node is:

2 x (100us + 2(Transmission Time + Hardware + Operating System + Application))

AGILE SYSTEMS
575 Kumpf Drive
Waterloo, ON
Canada N2V 1K3
T [519] 886-2000
F [519] 886-2075
E info@agile-systems.com

PVT Implementation

There are three main parts to the PVT structure.

1. Specifying a trajectory
2. Sending the points to the profile queue
3. Queue management.

The user creates a motion trajectory on the host computer and sends the points to the MAX Controller. The points on the trajectory must be the same time interval apart, unless another PVT() command is issued.

PVT Pseudo-Code Example

Overview: The following pseudo code example outlines how to use the PVT structure.

Note: For more details on the following Registers, please refer to the RegisterReference.chm file in DPWin.

Note: Fore more details on the following MAXClib functions please refer to the CommandReference.chm file in DPWin.

1. Setup the PVT Registers:
    - Set lower queue threshold.
        - Register Name: Profile_Lower_Threshold
    - Set upper queue threshold
        - Register Name: Profile_Upper_Threshold
    - Set the profile arm to enable axis queue for profile generation
        - Register Name: Profile_Arm_Clear
    - Set the profile queue and reset the profile pointer
        - MAXCLib Function: ProfileFlush()

2. Set PVT delta time value for profile motion segments and send PV points to the specified profile queue
    - Set the delta time
        - MAXCLib Function: ProfileDeltaTimeValue()
    - Set a Start position value
        - MAXCLib Function: ProfilePositionValue()

AGILE SYSTEMS
575 Kumpf Drive
Waterloo, ON
Canada N2V 1K3
T [519] 886-2000
F [519] 886-2075
E info@agile-systems.com

- Set a Start velocity value
    - MAXCLib Function: ProfileVelocityValue()

3. Start the PVT mechanism and added position and velocity points when polled or notified by the lower and upper queue thresholds (Queue Management)
    - Start the PVT mechanism
        - MAXCLib Function: ProfileStartArmed()
    - Set a position value
        - MAXCLib Function: ProfilePositionValue()
    - Set a velocity value
        - MAXCLib Function: ProfileVelocityValue()
    - Check the upper and lower threshold queue values
        - Register Name: Profile_Lower_Threshold
        - Register Name: Profile_Upper_Threshold

4. Stop the profile segment algorithm from processing any more profile points:
    - Send PVT stop
        - MAXCLib Function: ProfileEnd()

AGILE SYSTEMS
575 Kumpf Drive
Waterloo, ON
Canada N2V 1K3
T [519] 886-2000
F [519] 886-2075
E info@agile-systems.com

PT Implementation

Overview: The following pseudo code example outlines how to use the PT structure.

Note: For more details on the following Registers, please refer to the RegisterReference.chm file in DPWin.

Note: Fore more details on the following MAXClib functions please refer to the CommandReference.chm file in DPWin.

PT Pseudo-Code Example:

1. Setup the PT Registers:
    - Clear the PVT Queue Buffer
        - MAXCLib Function: ProfileFlush()
    - Set lower queue threshold.
        - Register Name: A_Profile_Lower_Threshold_A1
    - Set upper queue threshold
        - Register Name: A_Profile_Upper_Threshold_A1
    - Set the profile arm to enable axis queue for profile generation
        - Register Name: Profile_Arm_Clear
    - Set the profile queue and reset the profile pointer
        - MAXCLib Function: ProfileFlush()

2. Set PT delta time value for profile motion segments and send PT points to the specified profile queue
    - Set the delta time
        - MAXCLib Function: ProfileDeltaTimeValue()
    - Set a Start position value
        - MAXCLib Function: ProfileSplinePositionValue()

3. Start the PVT mechanism and added position and velocity points when polled or notified by the lower and upper queue thresholds (Queue Management)
    - Start the PVT mechanism

AGILE SYSTEMS
575 Kumpf Drive
Waterloo, ON
Canada N2V 1K3
T [519] 886-2000
F [519] 886-2075
E info@agile-systems.com

- MAXCLib Function: ProfileStartArmed()

- Set position value
  - MAXCLib Functiion: ProfileSplinePositionValue()

- Check the upper and lower threshold queue values
  - Register Name: Profile_Lower_Threshold
  - Register Name: Profile_Upper_Threshold

4. Stop the profile segment algorithm from processing any more profile points:

- Send PVT stop
  - MAXCLib Function: ProfileEnd()