P3 JPL Manual

Change History

Revision	Date	Description	Originator
AA	2015-11-13	Original Issue	B. Zenowich
AB	2016-01-26	Added some notes for clarification, updated the parameter tables with scaled position info and new homing variables, edited the homing procedure to reflect new method.	B. Zenowich
AC	2016-03-09	Added information about temperature based parameters and safe position.	C. Woodall
AD	2016-06-28	Added information on over current and over temperature faults.	C. Woodall
AE	2016-07-27	Corrected max temperature from 76 °C to 75 °C	C. Woodall
AF	2016-08-31	Added changes to the document to fit JPL control loop and release.	C. Woodall

Introduction
P3 is the third-generation of miniature motor controllers (Pucks) from Barrett Technology. In cooperation with NASA JPL, we have integrated P3 directly into the Maxon EC-max 30272765 and Maxon EC-max 22 283860 actuator which consists of a brushless DC motor. P3 transforms the Maxon motors into a bus-topology actuator requiring just 4 wires (Motor+, GND, CAN_H, CAN_L) and no separate controllers.

How it works: Cascaded Control Loops
P3 provides an industry-standard motion control architecture. The trajectory generator takes a target position and outputs a series of position commands. The position loop uses angular feedback from the integrated encoder to implement a Proportional-Integral (PI) controller which outputs a velocity command. The velocity loop implements a second PI controller which outputs a motor effort. The motor effort applies as a space vector function to yield the proper PWM command to the motor. Each of the controllers has an adjustable bandwidth you can use to fine-tune the motion control characteristics.

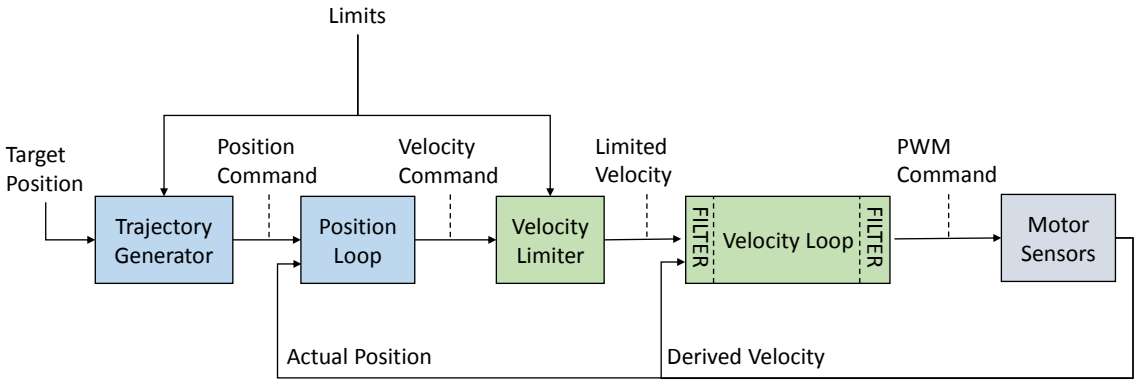

Figure 1: Control Loop Diagram

Note: We are actively developing the P3 firmware at this time.

How to communicate with it
- We support the Barrett CAN Protocol for full configuration and all modes of operation: http://web.barrett.com/support/Puck_Documentation/CAN_Message_Format.pdf
- We support a minimal but growing set of CANopen features at this time: http://web.barrett.com/support/Puck_Documentation/P3_CANopen_revAD.doc

How to control it: Control Modes
There are four distinct control modes that you can choose by setting the MODE property of P3:
1) MODE = 0: Idle. The controller ties the motor phase leads together for a resistive-braking effect. No current can flow through the motor.
2) MODE = 3: Position control. Write to the P property to set the commanded position.
3) MODE = 4: Velocity control. Write to the V property to set the commanded velocity.

4) MODE = 5: Trapezoidal velocity profile control. Write to the P property to set the target position. The controller will generate a trapezoidal velocity profile using ACCEL and MV and begin to output a series of commanded positions to the position controller. (NOTE: This mode is still under development).

About P3 Motor Currents

Motor currents are always 14-bits wide (-8192..8191) corresponding to (-11..11) A_{PK}. Divide A_{PK} by √2 to get A_{RMS}. Commanded motor current is limited to the lower of IPEAK or MT. If the average motor current is above ICONT for more than IPKMS milliseconds, the commanded current will fall back to ICONT until the commanded motor current drops below ICONT. The motor efforts are actually related to desired currents, but the current controller essentially becomes open-loop.

About P3 Angular Velocity

Unlike current, there is no absolute limit for velocity, and the controller does not differentiate between continuous or peak velocities. You are free to set Max Velocity (MV) to any value you want to achieve your desired motion profile, but take care not to exceed the datasheet's specifications. For example, the max velocity of the Maxon MAX EC 27 is 18000 rpm, but the allowable continuous velocity depends on the torque applied to the motor. Please reference the motor datasheets to set these values properly. The controller will obey MV in Position and Velocity modes, not Torque mode.

How to update it: In-System Firmware Updates

We provide a Windows-based firmware updater here:
http://web.barrett.com/support/Puck_Firmware/P3_Update.zip
Follow the directions in that folder's README.txt file to update the P3 firmware over CAN.

Methods for updating firmware are also included in Barrett's wxPuck and wxPuckTester applications, which can be more user-friendly applications for updating firmware.**Motor Control Variables (Barrett CAN Protocol)**

Property	Key	Frac	Units	Notes
T	42	0	A_{PK}*8192/11	Motor current (write=cmd, read=actual)
VERS	0	0	-	Firmware version
SN	2	0	-	Puck serial number
ID	3	0	-	CAN ID
ERROR	4	0	-	Fault status
MODE	8	0	-	Control mode: 0=idle, 2=torque, 3=position, 4=vel
V	44	8	rad/s	Gearhead output velocity (write=cmd, read=actual)
P	48	0	Encoder cts	Motor position (write=cmd, read=actual)
MOFST	61	0	Encoder cts	Raw encoder reading at the start of an electrical cycle
IOFST	62	0	A/D cts	Raw A/D reading of current when MODE = 0
MECH	66	0	Encoder cts	Raw encoder reading (0-4095)
CTS	68	0	Encoder cts	Motor encoder resolution (4096, typ)
IKCOR	93	0	A_{PK}*8192/11	Current to apply during MOFST calibration
SCLCMD	63	0		16-bit command position, scaled between the two soft-stops

SCLFBK	64	0		16-bit position feedback, scaled between the two soft-stops
TENST	83	4	Gear Head Ratio * 16	Gear Head Ratio in Q12.4 units. For example, a gearhead ratio of *300* has *TENST = 300*16 = 4800*

NOTE: All properties listed here are 16-bits wide.

NOTE: Key = Property key value used in the Barrett CAN Protocol message frames, as described in the CAN_Message_Format.pdf document linked above.

NOTE: Frac = Number of bits of fraction defined for each property. For example, property V (velocity) is a 16-bit integer, but the lower 8 bits are interpreted as a fractional value. So an integer value of 64 for V would be interpreted as 64 / 2^8 = 0.25 rad/s.

Motor Control Constants, Example Values for a supported motor(BarrettCAN)

Property	Key	Frac	Units	Value	Conv	Notes
R	109	12	Ω (/ϕ)	0.82	3359	Per-phase or terminal res/2
L	110	12	mH (/ϕ)	0.27	1106	Per-phase or terminal ind/2
J	111	15	$mNms^2$	0.7301	23924	Rotor + gearhead inertia at output
KT	112	12	Nm/A_{RMS} (/ϕ)	1.1/3	1502	Per-phase torque constant
ICONT	113	0	A_{PK}*8192/11	0.76 A_{RMS}	800	A_{RMS} * √2 * 8192/11
IPEAK	114	0	A_{PK}*8192/11	1.7 A_{RMS}	1790	A_{RMS} * √2 * 8192/11
IPKMS	115	0	ms	1000	1000	See NOTE 2, above
IBW	116	0	Hz	300	300	Current control bandwidth R/(2pi * L), typ
VBW	117	0	Hz	25	25	Velocity control bandwidth, IBW/10, typ
PBW	118	8	Hz	0.94	240	Position control bandwidth, VBW/10, typ
POLES	90	0	-	8	8	Rotor magnet count or pole pairs/2
MV	45	8	rad/s	100 RPM	2681	Max velocity at gearhead output (RPM*2pi/60*256)
VBUS	21	0	Volts	24	24	Motor bus voltage
MT	43	0	A_{PK}*8192/11	3.5 A_{RMS}	3685	A_{RMS} * √2 * 8192/11
ACCEL	82	0	rad/s/150 uS	10	10	Only used in position control

NOTE: All properties listed here are 16-bits wide. The current-related properties never exceed 14-bits in practice.

Homing Constants (Barrett CAN Protocol)

Property	Key	Frac	Units	Value	Conv	Notes
MDS	65	0	ms	1000	1000	Auto-home delay, 0 = No auto-homing
IVEL	73	8	rad/s	10 RPM	268	Initialization velocity, + = CW

						homing, - = CCW homing
IHIT	108	0	A_{PK}*8192/11	0.19 A_{RMS}	200	Homing current threshold
IOFF	74	0	Encoder cts	20000	20000	CW hard-stop offset
JOFST	98	0	Encoder cts	40000	40000	CCW hard-stop offset
DP	50	0	16-bit scaled	32768	32768	Default position after homing (0-65535)
E	52	0	Encoder cts	0	0	Single-ended homing offset
DS	60	8	Rad/s	20 RPM	536	Post-homing move velocity

NOTE: Properties with units of “Encoder cts” are 32-bits wide. All others are 16-bits wide.

Homing Procedure
If MDS > 0, auto-homing will begin MDS milliseconds after the motor control firmware begins executing. If MDS = 0, auto-homing is disabled. Set CMD(29) to CMD_HI(13) to initiate homing (or to re-home).

The sign of IVEL determines the initial direction of movement during homing. Dual hard-stop homing requires non-zero values for both IOFF and JOFST (the CW and CCW hard-stop offsets) and E must be set to zero. If you want CW single hard-stop homing, set IOFF, E, and IVEL to positive values. If you want CCW single hard-stop homing, set JOFST and E to positive values and set IVEL to a negative value.

Before starting, the max motor torque is limited to IHIT. The motor then starts moving with a velocity of IVEL until the sensed velocity becomes zero AND the motor current saturates at IHIT. Then the motor reverses direction by setting its velocity to –IVEL.

If this is single-ended homing, then the motor runs until the proper number of encoder counts has passed (IOFF or JOFST, depending on which stop it just hit). The controller then initializes its position value with either 0 or E, depending on the homing direction. Then it moves to its default starting position (DP) using the post-homing move velocity (DS).

If this is double-ended homing, then the motor runs until it hits the second hard-stop (sensed velocity becomes zero AND the motor current saturates at IHIT). Then the motor reverses direction again by restoring its velocity to IVEL. It runs until the proper number of encoder counts has passed (IOFF or JOFST, depending on which stop it just hit). Then it moves to its default starting position (DP) using the post-homing move velocity (DS).

The controller is left in position-control mode after homing so that it is ready to accept a stream of commanded positions.

NOTE: The controller does not verify the operational range during double-ended homing.

NOTE: IOFF and/or JOFST should contain positive values corresponding to the number of encoder counts to traverse after hitting a hard-stop. IOFF = number of encoder counts to back off from the CW stop before setting its CW soft-stop there. JOFST = number of encoder counts to back off from the CCW stop before setting its CCW soft-stop there.

NOTE: Soft-stops are disabled until the homing routine completes.

Additional Properties

Property	Key	Frac	Units	Value	Conv	Notes
THERM	20	8	°C	40 °C	10240	Thermistor temperature in °C (16-bit number in Q8.8 format. Temperature * 2^8)
PTEMP	10	8	°C	70 °C	17920	Peak temperature in °C (16-bit number in Q8.8 format. Temperature * 2^8)
OTEMP	11	0	Bool	1	Fault	Temperature Fault and Fault Reset property.

Errors and Faults

Under the current system there are 2 faults:

1) Over-Temperature Faults
2) Over-Current Faults

These two faults are handled in different ways and have different parameters associated with them.

Over-Current Faults

The overcurrent faults are run on the commanded currents with a low pass filter, with a cutoff of 20Hz. The commanded current is allowed to run from –IPEAK to IPEAK, currents cannot be commanded which are above IPEAK. If the absolute value of the commanded current, I, is greater than ICONT (the continuous current) a counter starts to increment. If I drops below ICONT the timer restarts. Once the counter has been counting for IPKMS ms the currents are thresholded now to ICONT. If the current I drops below ICONT the limit is released and IPKMS must be exceeded again in order to trigger the overcurrent.

The CANOpen error register flag for current is only held while I is being thresholded to ICONT and is not a permanent state.

Over-Temperature Faults

The over temperature fault is controlled by the following parameters: PTEMP, OTEMP and THERM. PTEMP is the peak temperature, OTEMP is a register which stores the error state, and THERM is the current filtered temperature. The THERM temperatures are filtered with a 20Hz cutoff and are mapped from the raw values to temperature using a multiple region linear interpolation which has a max error of 1 °C in the main operational zones and can peak to 2 °C at temperatures below 20 °C.

The over-temperature fault is triggered when THERM is greater than PTEMP for more than 5 seconds (hard-coded). PTEMP is not allowed to be greater than 75 °C. When the over-

temperature fault is generated OTEMP will read back 1. If you write any value to OTEMP it will clear the error. When the fault occurs motor currents are forced to 0.

The CANOpen error register flag is set the whole time the ERROR flag is set until it is cleared, which can only be cleared in CANOpen through a reset.