

Overview:

The Parallax BASIC Stamp 2 (BS2) microcontroller is a widely used embedded control processor. It has the capability of interfacing nicely with the ICON and MINI PID Position Controllers produced by Solutions Cubed. This application note describes some of the basic features of our brushed DC motor position control modules and how they might be controlled with a BS2.

The BS2 code files that are provided for this application note should be compatible with many of the Parallax BASIC Stamp products. In most cases you will simply have to modify the baud rate constants to allow other versions of the BS2 (such as the BS2SX) to work with the code provided.

AN800 implements PID filter programming which is necessary to tune the PID filter for position control. It also implements a step function movement that can be used to tune the PID settings to your particular motor and mechanical system. You will need to implement this application note before going on to AN801 or AN802.

AN801 describes the implementation of Velocity control mode. Motor speed can be controlled with the PID filter by specifying the desired number of encoder pulses every PID update period. Additionally the update rate and acceleration limit can be defined with the position controller's command set.

AN802 defines trapezoidal movement segments that are then used to describe a trapezoidal movement profile. This functionality generates smooth movement from point-to-point while at the same time limiting motor velocity.

More complete descriptions of the various commands, operating modes, and electrical characteristics, of the ICON or MINI PID Position Controllers may be found in the datasheet associated with the product you are using.

Hardware:

This application note was based on connectivity provided by the ICON_BS2 kit available at Parallax or Solutions Cubed. With this kit the BS2 may be interfaced directly to an ICON PID Position Controller or the MINI PID Position Controller. The motor used in testing was a Pittman GM8724S023 (www.pittmannot.com). This motor comes equipped with a 500 counts-per-revolution (CPR) encoder, has a 60.5:1 gear ratio, and operates at 24VDC.

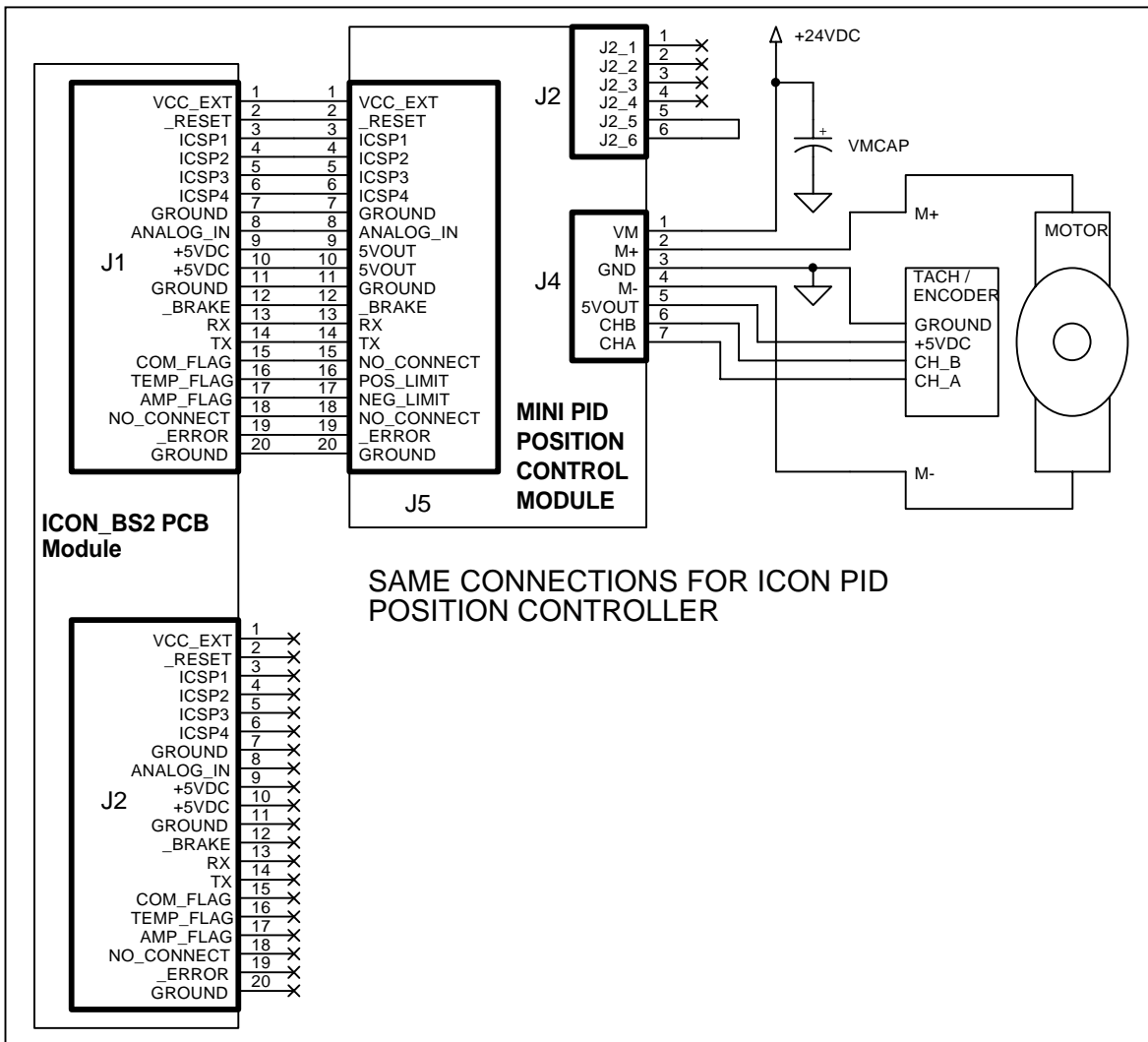
The hardware required for this application note is listed below.

- 1 – Parallax BASIC Stamp 2
- 1 – ICON – BS2 Carrier Board
- 1– ICON PID Position Controller or MINI PID Position Controller
- 1– ICON H-Bridge (not required with MINI PID Position Controller)
- 1 – power supply
- 1– DC motor with a logic level quadrature encoder

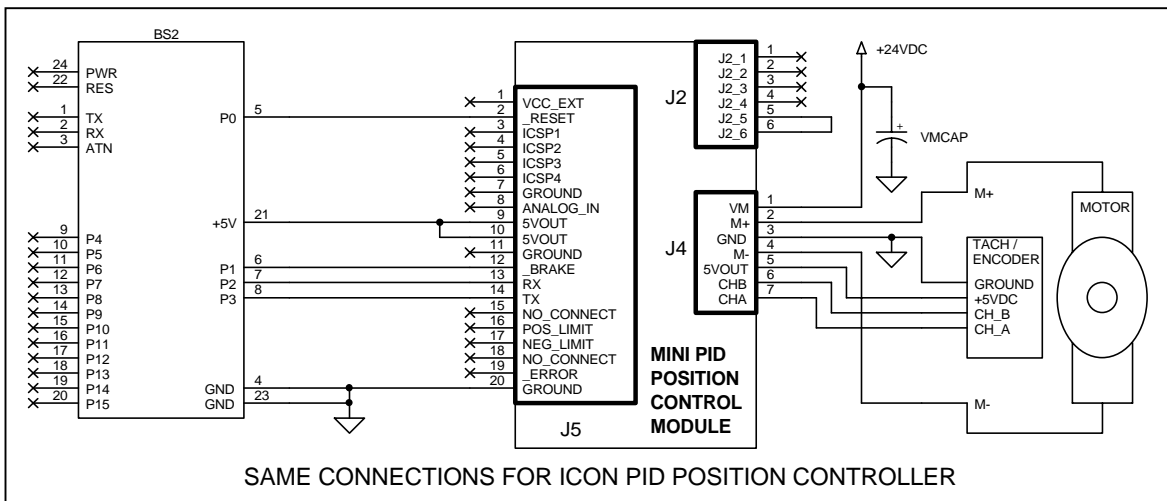
All of these application notes make limited use of the functionality available in these position controllers. You can access more functionality with the ICON Adapter board and the position controller software available at www.solutions-cubed.com.

You should expect to make modifications to some of the settings in these application notes to “fine tune” the position controller to your mechanical system. Additional information on tuning the PID filter can be found in the datasheet of each PID position controller.

Schematic:



Schematic for ICON_BS2 Kit to Position Controller/Motor



Schematic for BS2 to Position Controller/Motor

AN800:

Application note 800 makes use of Direct position control mode. This form of control typically uses the “Write Desired” position command to direct motor movements. In Direct control mode the PID filter settings are used to generate the motor speed and direction signals. The PID algorithm applies the PID settings to an error signal. The error signal is defined as the difference between the actual motor position and the desired motor position. “PID” stands for the three terms that make up the PID filter. They are proportional (P), integral (I), and derivative (D) terms. Additional PID filter settings, such as the error-band, integral clear counter, and PID period constant are also associated with the PID filter. More detailed discussion of these components of the PID filter can be found in the datasheet of the position controller you are using.

In general, the P term is responsible for the major push a motor receives when the error signal is large. If used alone (with $I = 0$ and $D = 0$) the P term will move a motor close to the desired position, but as the error signal gets smaller the P term’s effect on motor movement is limited and eventually has no effect.

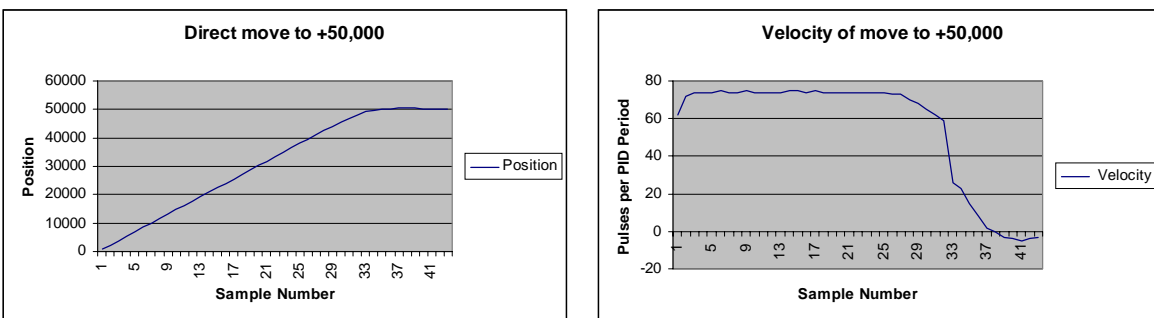
The I term is multiplied by the sum of the error signals over time. Using the I term allows for small motor position corrections. Therefore the I term must be small in relation to the P and D terms. As the error signals build up the I term eventually starts to nudge the motor into the desired position. Overly large I terms cause motor oscillation, and terms which are too small will not push a motor into position when the P term’s effects peters out. The integral clear counter setting in the PID filter is used to clear the summed error signals after the motor is within the user-defined error-band for a number of PID update periods.

Finally, the D term is multiplied by the difference between the current error signal and a previous error signal. This difference is typically very small. To be effective the D term is typically the largest term of the PID set. The D term creates a drag on motor movement that can be used to prevent or minimize overshoot when the actual motor position approaches the desired motor position.

Direct position control mode will typically cause the motor to race to the desired position and then stop based on the PID filter settings.

AN800 commands the motor position to +50,000 and back to +500. The graphs below show the position and velocity readings that occur during the movement. From these graphs it can be seen that the velocity of the movement is high until the motor position approaches the commanded position. The velocity then slows and the move is completed. Overshoot is minimal but can be seen as the velocity goes negative at the end of the move. The negative velocity coincides with the motor being reversed to bring the motor position within the error-band set by the PID filter components.

Many of the subroutines used in the code for AN800 write data to EEPROM. These commands could be placed in a separate program and used to program the position controller settings, freeing up memory and variable space for your actual application.



AN801:

Application note 801 implements Velocity control mode to control motor speed instead of motor position. Velocity control is used to control the number of pulses read from the quadrature encoder during a given PID period. The PID period is defined by the PID period constant, a component of the PID filter setting.

The resolution of the frequency control is limited to the updates-per-second value associated with the PID period constant used (116 in this application). The resolution may be improved by modifying the PID Update Rate or by using 4X-quadrature decoder mode. Both of these functions are described in the datasheet associated with the product you are using.

Response time to motor load changes will be effected by the PID filter setting. It will also be effected by the PID Update Rate and Acceleration Limit set with the "Write-Store Velocity Settings" and "Write Velocity Settings" commands.

Low velocity settings will typically cause the motor to start or stop as its speed ramps up and down past the desired velocity. It is recommended that you use a combination of motor gearing, PID filter settings, velocity settings, and 4X mode to achieve slow motor rotations. The example in AN801 uses a motor with a gear ratio of 60.5:1 to attain motor shaft rotations of 1Hz.

In this example the desired velocity setting is 189. This value refers to 189 pulses received from the quadrature encoder during a PID update period. The PID period constant is 116 and relates to 640 PID updates per second. In order to improve the resolution of the frequency measurements the PID Update Rate (written to the controller as a component of the "Write Velocity Settings" command) is set to 4. This causes the velocity control algorithm to be called every 4 PID periods. Resolution is therefore improved to $640/4 = 160\text{Hz}$. The velocity setting is related to both the motor's gear ratio (60.5:1) and the motor's encoder (500CPR). With the motor used in this application the shaft would rotate once for every 30,250 (60.5×500) pulses received from the encoder. Dividing the desired number of pulses by the frequency resolution gives the desired frequency setting used by the position controller ($30,250/160 = 189$).

The "Write-Store Velocity Settings" command writes velocity control settings to EEPROM. This command should be used to set power-on default settings for motor speed control, but it should not be used while the motor is in motion. Writing to EEPROM takes time and will cause the velocity control to be inaccurate. In order to effect motor speed changes while the motor is moving use the "Write Velocity Settings" command.

Many of the subroutines used in the code for AN801 write data to EEPROM. These commands could be placed in a separate program and used to program the position controller settings, freeing up memory and variable space for your actual application.

AN802:

Application note implements Trapezoidal control mode. Trapezoidal control can be used to minimize abrupt motor speed changes that can occur in Direct control mode. In Trapezoidal control mode the user builds trapezoidal movement-segments, each of which constitutes a single move to a desired position.

Up to 16 different movement-segments can be stored in the position controller's EEPROM. Each movement segment consists of a series of components that are used to ramp a motor's speed up to the desired velocity and then as the motor's position approaches the desired position the motor speed is ramped down. Ideally, a smooth transition from the starting point to the desired position can be achieved.

This application note writes three movement-segments (0,1, and 2) to EEPROM, and then sends the "Run Profile" command to execute these segments.

The components of a movement segment are listed in the table below.

Segment Component	Value Range	Description
Segment number	0-15	index number for segment number
Position	-2,147,483,648 to -2,147,483,647	two's compliment 32 bit number defines the position that the trapezoidal segment will move to
Acceleration	1 to 32,767	the maximum change in motor velocity while increasing motor speed
Velocity	1 to 32,767	The absolute value of the maximum motor velocity
Dwell Time	1 to 16,777,215	number of PID periods allowed for device to hunt for final position after movement exceeds commanded position
PID Number	0 to 2	PID filter settings to be used with this trapezoidal segment

In an application where more than 16 movement-segments are necessary, you could overwrite movement-segment 0 after each move and then use the "Run Profile" command to execute a single movement (in this case use Start Segment = 0, End Segment = 0, and Number of Loops = 1). AN802 writes three movement-segments and executes them with a "Run Profile" command where the Start Segment = 0, End Segment = 2, and the Number of Loops = 2. The components of the "Run Profile" command are listed in the table below.

Description: Run Profile is used while operating in Trapezoidal Control mode. When this command is received each trapezoidal movement segment from the Start Segment through the End Segment is executed (the Start-End segment numbers relate to the "Segment Number" defined in the Write Segment and Read Segment commands). The movement segments can be executed continuously by setting the Number of Loops value to 0, or can be repeated 1 to 255 times by setting the Number of Loops value to a number ranging from 1 to 255. To execute a single trapezoidal segment set the Start Segment and End Segment to the same value.

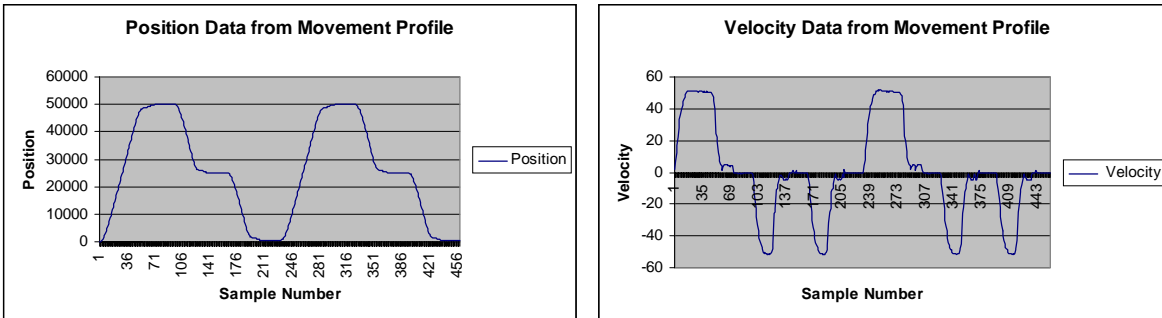
Component of Command	Range of Values	Number of Bytes	Decimal(Hex) Example
Command	226	1	226 (0xE2)
Address	0 to 255	1	1 (0x01)
Start Segment	0 to 15	1	0 (0x00)
End Segment	0 to 15	1	2 (0x02)
Number of Loops	0 to 255	1	2 (0x02)
Checksum	0 to 255	1	231 (0xE7)
Component of Response	Range of Values	Number of Bytes	Decimal(Hex) Example
ACK	6	1	6 (0x06)

Notes: The Number of Loops setting is the number of times the Trapezoidal Profile (Start Segment through End Segment) will be executed. In the example above segments 0,1, and 2, would be executed in that order 2 times. The movement profile would then end. If the Number of Loops setting is 0 then the Start through Stops segments are executed continuously.

AN802 shows the components of each segment in decimal and then breaks them down into hexadecimal byte. This is done to point out the order in which the data is sent. The data may be sent as decimal or may be carried in variables that have their byte-sized components mapped into word-sized variables.

Program and variable space needed to write the movement-segments into EEPROM could be conserved. This would be done writing a separate program to write movement-segment settings to the position controller, and having your application program send commands based on user inputs.

The program listed for AN802 executes three movement-segments via the "Run Profile" command. These segments move to position +50,000, then back to position +25,000, and again back to position +500. The velocity limit is set to 50, and the acceleration limit is 4. The position and velocity curves from this trapezoidal profile are displayed in the graphs below.



Summary:

The BS2 can act as a master unit controlling complex mechanical control systems when used with the ICON or MINI PID Position Controllers. These application notes were kept simple to allow new users to come up to speed with a minimum of effort. Two's complement numbers were avoided to simplify coding, as were variables that made use of the full range of positions allowed by these position controllers. These application notes should act as a springboard for implementing motion control with the BS2.

Many of the commands accessed by these application notes program values into EEPROM on board the position controllers. These values could be programmed into the position controller's with software available at www.solutions-cubed.com and an RS232 to TTL converter (such as the MAX232 from Maxim). This would significantly reduce the overhead required by the BS2 to get the position controller up and running.