Web Site: www.parallax.com
Forums: forums.parallax.com
Sales: sales@parallax.com
Technical: support@parallax.com

Office: (916) 624-8333
Fax: (916) 624-8003
Sales: (888) 512-1024
Tech Support: (888) 997-8267

Eddie Control Board Firmware Command Set

The Eddie Control Board (below, left) is a complete robot controller and sensor-interface solution. It can be programmed with custom firmware to accommodate a wide range of applications, or can be used with Parallax's ready-to-go Eddie Control Board firmware, designed for the Eddie Robot Platform (below, right).

This firmware provides an easy-to-use serial command interface to control and manage all of the on-board peripheral electronics such as motor drivers, digital I/O, and analog to digital converter (ADC) channels.

The following document provides a command set summary followed by a detailed description and example for each command, for the firmware distributed as EddieFirmware-v1.1.zip. This file is available along with other Eddie related downloads, at www.parallax.com/eddie.

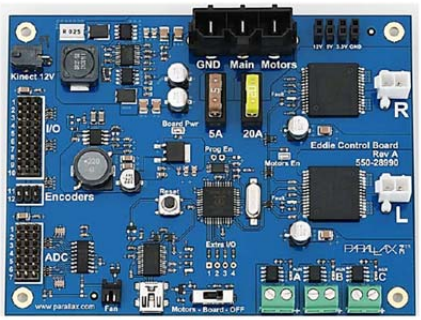

Table of Contents

Command Set Summary

This table provides a summary of available commands and associated parameters. For detailed information, see the Command Set Detail section. Note: click on the command mnemonic to jump to that section.

Cmd	Input Parameters	Return Parameters	Values	Description
Interface				
HWVER		<***version***>	***version*** = 0..FFFF	Get hardware version
VER		<***version***>	***version*** = 0..FFFF	Get firmware version
VERB	<***mode***>		***mode*** = 0(off), 1(on)	Set verbose mode
BLINK	<***pin***><***rate***>		***pin*** = 0..1F ***rate*** = 0..FFFF	Toggle pin at a specified rate in increments of 0.1Hz
I/O Control				
IN	<***bitmask***>		***bitmask*** = 0..7FFFF	Set GPIO pins in bitmask to inputs
OUT	<***bitmask***>		***bitmask*** = 0..7FFFF	Set GPIO pins in bitmask to outputs
LOW	<***bitmask***>		***bitmask*** = 0..7FFFF	Set GPIO pins in bitmask to low (only applies to output pins)
HIGH	<***bitmask***>		***bitmask*** = 0..7FFFF	Set GPIO pins in bitmask to high (only applies to output pins)
INS		<***bitmask***>	***bitmask*** = 0..7FFFF	Get GPIO pins currently set as inputs
OUTS		<***bitmask***>	***bitmask*** = 0..7FFFF	Get GPIO pins currently set as outputs
LOWS		<***bitmask***>	***bitmask*** = 0..7FFFF	Get GPIO pins currently set as low
HIGHS		<***bitmask***>	***bitmask*** = 0..7FFFF	Get GPIO pins currently set as high
READ		<***bitmask***>	***bitmask*** = 0..7FFFF	Get current state (high/low) of all GPIO pins
Sensor Interfacing				
SPNG	<***bitmask***>		***bitmask*** = 0..FFFF	Set pins in bitmask to act as PING))) sensor pins
SGP	<***bitmask***>		***bitmask*** = 0..7FFFF	Set pins in bitmask to act as GPIO pins
PING		<***value1***>[<***value2***>... <***valueN***>]	***value*** = 0,12..B54	Get PING))) sensor sonar measurements (one 12-bit value per sensor)
ADC		<***value1***>... <***value8***>	***value*** = 0..FFF	Get all ADC values (12-bit values)
Motor Control				
GO	<***left***><***right***>		***left***/***right*** = 80..7F	Set motor power (signed byte)
GOSPD	<***left***><***right***>		***left***/***right*** = 8000..7FFF	Set motor speed (signed word)
STOP	<***dist***>		***dist*** = 0..FFFF	Slow to a stop over specified distance
TRVL	< ***dist*** >< ***speed*** >		***dist*** = 8000..7FFF ***speed*** = 0..7FFF	Travel a specified distance in a straight line, ramping up to a maximum specified speed
TRVL	<***angle***><***speed***>		***angle*** = 8000..7FFF ***speed*** = 0..7FFF	Rotate in place by a specified angle, ramping up to a maximum specified speed
ACC	<***rate***>		***rate*** = 0..FFFF	Set rate of acceleration/deceleration
SPD		<***left***><***right***>	***left***/***right*** = 8000..7FFF	Get the current average speed (positions per second) for both wheels
HEAD		<***angle***>	***angle*** = 0..168 (decimal 0..359)	Get the current heading (in degrees) relative to start
DIST		<***left***><***right***>	***left***/***right*** = 80000000.. 7FFFFFFF	Get the position of each wheel (signed 32-bit value) relative to start
RST				Reset the distance and heading values to 0

Communication Protocol

From an application developer's standpoint, it is useful to understand the general structure of commands and the requirements of the communication protocol. When interfacing with the Eddie Control Board using the firmware discussed in this document, ensure the following format and settings are followed.

COM Settings

The Eddie Control Board communicates over USB; and when connected to a PC, the board enumerates as a serial COM port. Configure the COM port to use these settings:

- 115.2 kBaud
- 8-bit character size
- 1 stop bit
- No parity

General Command Form

All commands adhere to the same general format which is shown below:

Input:	*<cmd>* **[** **<WS>** *<param1>* ... **<WS>** *<paramN>* **]** **<CR>**
Response (Success):	**[** *<param1>* ... **<WS>** *<paramN>* **]** **<CR>**
Response (Failure):	**ERROR[** **<SP>** **-** **<SP>** *<verbose_reason>* **]** **<CR>**

...where:

- Sections inside braces **[ ]** are for one or more optional parameters (as required by the command or mode).
- *<cmd>* is the command mnemonic.
- *<param1>* and *<paramN>* are any parameters required by the command. Numbers are always entered as hex values and are two's complement whenever the value is signed.
- **<WS>** is one or more white-space characters. Only space (ASCII 32) or tab (ASCII 9) characters are accepted as whitespace.
- **<CR>** is a single carriage-return character (ASCII 13).
- **<SP>** is a single space character (ASCII 32).
- *<verbose_reason>* is the optional error message displayed when verbose mode is enabled (see VERB command).

The range of allowed graphical characters is from ASCII 32 up to ASCII 126 (inclusive). Any non-graphical characters outside this range with the exception of carriage return (ASCII 13) and tab (ASCII 9) are treated as invalid characters and are ignored (will not be added to the buffer).

Up to 254 characters may be entered to form a valid command (including the terminating carriage return character). Any characters entered beyond this limit will be ignored and mark the command as invalid. The command handler will only process and respond to a command after a carriage return character is received.

Command Set Detail

HWVER

Get hardware version.

Syntax	Input	HWVER
	Response	<***version***>
Parameters:	***version***	The hardware version is displayed as four hex characters.
Example: The hardware version command returns a hexadecimal value of 2 which represents version 2.		**HWVER** 0002

VER

Get firmware version.

Syntax	Input	VER
	Response	<***version***>
Parameters:	***version***	The firmware version is displayed as four hex characters.
Example: The firmware version command returns a hexadecimal value of A which represents version 10.		**VER** 000A

VERB

Set verbose mode.

Syntax	Input	VERB <***mode***>
	Response	
Parameters:	***mode***	0 = Verbose mode off (default); 1 = verbose mode is on.
Details:		When verbose mode is on, command input errors will return descriptive text in addition to the standard command acknowledgement. When verbose mode is off, no descriptive text is returned.
Example: With verbose mode off (default) an invalid entry does not return a text error message. After verbose mode is turned on, an error message is returned for an invalid input.		**akdj** ERROR **VERB 1** **akdj** ERROR – Invalid Command

BLINK

Toggle the specified pin at a specified rate (in increments of 0.1Hz).

Syntax	Input	BLINK <***pin***> <***rate***>
	Response	
Parameters:	***pin***	***An 8-bit hex value specifying a*** pin number on the Propeller, from 0 to 18.
	rate	***A 16-bit hex value*** specifying the blink frequency for the specified pin. This value is in units of 0.1Hz. A rate of 0 disables blinking functionality.
Details:		The blink command can be used to automatically toggle an output pin connected to an LED or light strip. This can be useful for debugging and when controlling lights or accessories from the AUX ports. Only affects pins currently set as an output. See OUT
Example: Toggle pin P16 (16 = hex 10) which connects to AUX A port. Makes the output LED toggle at a rate of 5 Hz (50 = hex 32).		**`BLINK 10 32`**

IN

Set GPIO pins in bitmask to be inputs.

Syntax	Input	IN <***bitmask***>
	Response	
Parameters:	***bitmask***	The bitmask is a 32-bit hex value. A bit value of '1' will set the corresponding GPIO pin to be an input. A bit value of '0' has no effect on the pin mode. By default all GPIO pins are inputs. Only affects pins currently set as GPIO pins. See SGP
Example: Configure pins P0, P2, P6-9, and P12-17 as inputs.		**`IN 0003F3C5`**

OUT

Set GPIO pins in bitmask to be outputs.

Syntax	Input	OUT <***bitmask***>
	Response	
Parameters:	***bitmask***	The bitmask is a 32-bit hex value. A bit value of '1' will set the corresponding GPIO to be an output. A bit value of '0' has no effect on the pin mode. Only affects pins currently set as GPIO pins. See SGP
Example: Configure pins P1, P3-5, P10-11 and P18 as outputs.		**`OUT 00040C3A`**

LOW

Set GPIO pins in bitmask to drive low (when configured as an output).

Syntax	Input	LOW <***bitmask***>
	Response	
Parameters:	***bitmask***	The bitmask is a 32-bit hex value. A bit value of '1' will set the corresponding GPIO pin to drive low (when configured as an output). A bit value of '0' has no effect on the pin drive state. By default all GPIO pins are low. Only affects pins currently set as GPIO pins. See SGP
Example: Drive pins P5-P7, P10-12, and P14-P18 low, when configured as outputs.		**LOW 0007DCE0**

HIGH

Set GPIO pins in bitmask to drive high (when configured as an output).

Syntax	Input	HIGH <***bitmask***>
	Response	
Parameters:	***bitmask***	The bitmask is a 32-bit hex value. A bit value of '1' will set the corresponding GPIO pin to drive high (when configured as an output). A bit value of '0' has no effect on the pin drive state. Only affects pins currently set as GPIO pins. See SGP
Example: Drive pins P0-P4, P8-P9 and P14-P15 high, when configured as outputs.		**HIGH 0000C31F**

INS

Get which GPIO pins are set as inputs.

Syntax	Input	INS
	Response	<***bitmask***>
Parameters:	***bitmask***	The bitmask is a 32-bit hex value. A bit value of '1' is returned when the corresponding GPIO pin is configured as an input pin. A bit value of '0' means that the pin is either set as an output, or the pin is not configured as a GPIO pin.
Example: Pins P0, P2, P6-9, P12-15 and P17 are currently configured as inputs.		**INS** **0002F3C5**

OUTS

Get which GPIO pins are set as outputs.

Syntax	Input	OUTS
	Response	<***bitmask***>
Parameters:	***bitmask***	The bitmask is a 32-bit hex value. A bit value of '1' is returned when the corresponding GPIO pin is configured as an output pin. A bit value of '0' means that the pin is either set as an input, or the pin is not configured as a GPIO pin.
Example: Pins P1, P3 -5, P10-11 and P18 are currently as configured as outputs.		**OUTS** **00040C3A**

LOWS

Get which GPIO pins are set to drive output low.

Syntax	Input	LOWS
	Response	<***bitmask***>
Parameters:	***bitmask***	The bitmask is a 32-bit hex value. A bit value of '1' is returned when the corresponding GPIO pin is configured to drive output low. A bit value of '0' means that the pin is either set to drive the output high, or it is not configured as a GPIO pin.
Example: Pins P5-P7, P10- P12, and P14-P18 are currently set to drive low, when configured as outputs.		**LOWS** **0007DCE0**

HIGHS

Get which GPIO pins are set to drive output high.

Syntax	Input	HIGHS
	Response	<***bitmask***>
Parameters:	***bitmask***	The bitmask is a 32-bit hex value. A bit value of '1' is returned when the corresponding GPIO pin is configured to drive output high. A bit value of '0' means that the pin is either set to drive output low, or it is not configured as a GPIO pin.
Example: Pins P0-P4, P8-P9 and P14-P15 are currently set to drive high, when configured as outputs.		**HIGHS** **0000C31F**

READ

Get the logical state of all input pins.

Syntax	Input	READ
	Response	<***bitmask***>
Parameters:	***bitmask***	The bitmask is a 32-bit hex value. A bit value of '1' is returned when the corresponding GPIO pin is configured as an input and its logical state is high. A bit value of '0' means that the logical state on the pin is either low or not configured as a GPIO pin.
Example: Pins P0-P1 and P4-P8 are GPIO pins in a high state.		**READ** **000001F3**

SPNG

Set pins in bitmask to act as PING))) Ultrasonic Sensor pins.

Syntax	Input	SPNG <***bitmask***>
	Response	
Parameters:	***bitmask***	The bitmask is a 32-bit hex value. A bit value of '1' will configure the corresponding pin to be read as a PING))) sensor instead of a GPIO pin. A bit value of '0' has no effect on the configuration state of the pin.
Details:		If a pin is configured as a PING))) sensor pin, when the PING command is issued, the controller will send out a pulse to start the measurement, then measure the pulse returned from the PING))) sensor to calculate the distance value. Only affects pins 0 through 15. At power on, the default bitmask is hex "3".
Example: Configure pins P2-P9 to be read as PING))) sensors.		**SPNG 000003FC**

SGP

Set pins in bitmask to act as GPIO pins.

Syntax	Input	SGP <***bitmask***>
	Response	
Parameters:	***bitmask***	The bitmask is a 32-bit hex value. A bit value of '1' will configure the corresponding pin to act as a GPIO pin. A bit value of '0' has no effect on the configuration state of the pin.
Details:		Pins configured as GPIO pins can be controlled and read using the I/O control commands (such as IN, OUT, LOW, HIGH, READ, etc.). Only affects pins 0 through 18.
Example: Configure pins P6-8, P14, P17 as GPIO pins.		**SGP 000241C0**

PING

Read PING))) sensors.

Syntax	Input	PING
	Response	<***value1***>[<***value2***>...<***valueN***>]
Parameters:	***valueN***	The values returned for each sensor are 12-bit hex values. One measurement is returned for each pin configured as a PING))) sensor pin.
Details:		The PING command will only initiate a measurement on pins that have been configured as PING))) sensor pins (see SPNG command). The response includes a measurement for each active sensor, from lowest pin number to highest, with a range in hex of 12 to B54. There are 470.2 units per mm or 11942.75 units per inch. Non-functioning sensors, or sensors with no objects within its range, will respond with a null.
Example: The distance measurements for the 10 previously configured PING))) sensor pins.		**PING** 133 3C9 564 0F9 29B 0F0 31A 566 1E0 A97

ADC

Get all Analog to Digital Converter (ADC) values.

Syntax	Input	ADC
	Response	<***value1***>...<***value8***>
Parameters:	***valueN***	The values returned for each ADC channel are 12-bit hex values. The possible range is 0 to FFF, which corresponds to 0.00V up to 5.00V respectively. One measurement is returned for each of the eight ADC channels. Note that the highest ADC channel (channel 8) is connected through an on-board voltage divider to the input supply voltage to the board so it may be used to monitor the battery voltage.
Details:		It is important to note that the Eddie Control Board currently uses a 10-bit ADC so the accuracy of the two least significant bits is not guaranteed.
Example: The ADC measurements values for all eight ADC channels.		**ADC** 9C7 11E E4E 5AB 20F 97B 767 058

GO

Set motor power (left and right).

Syntax	Input	GO <***left***> <***right***>
	Response	
Parameters:	***left***/***right***	The left and right power levels are entered as signed (two's complement) 8-bit hex values. The range of valid values is 81 (full reverse) to 7F (full forward). A value entered of 80 will be clipped to 81 to maintain symmetry between positive and negative drive power levels.
Details:		This command sets the motor output PWM ratio, which effectively corresponds to the motor drive power. The range of values (in decimal terms) is -127 to +127, and corresponds to -100% to +100% duty cycle ratio respectively. Note that two's complement signed 8-bit values typically range from -128 to +127; so to maintain symmetry in the effective drive power, a value of -128 is clipped to be -127.
Example: Set the left motor power level to a value of 54 and set the right motor power level to a value of -68.		**GO 36 BC**

GOSPD

Set motor speed (left and right).

Syntax	Input	GOSPD <***left***> <***right***>
	Response	
Parameters:	***left***/***right***	The left and right speeds have units of positions per second and are entered as signed (two's complement) 16-bit hex values. The range of allowed values is from 8000 to 7FFF.
Details:		This command sets the drive speed in positions per second. Because it uses encoder/position feedback for each wheel, the controller can automatically regulate drive power to each motor in order to maintain the true desired speed. When setting the desired drive speed, keep in mind that the motors have physical limitations for maximum output power and top speed, so for this command to operate properly and maintain consistent speed, values should be chosen which will not exceed the motors' capabilities. When transitioning from one set speed to another, the controller will transition gradually according to the rate of acceleration set by the ACC command.
Example: Set both motors to drive forward at 47 positions per second.		**GOSPD 2F 2F**

STOP

Slow to a stop over a specified distance

Syntax	Input	STOP <***distance***>
	Response	
Parameters:	***distance***	Stopping distance, in positions, entered as a 16-bit hex value. The range of allowed values is 0 to FFFF.
Details:	This command will cause the robot to slow to a stop over the specified distance. A value of zero will immediately stop the robot. At high speeds, low values may cause the robot to pass the position while processing the command, causing it to overshoot and reverse to travel back to the desired position. At low speeds the robot may stop before the desired distance.	
Example: Slow to a stop, over 10 positions.		**`STOP 0A`**

TRVL

Drive forward by a specified distance (in positions)

Syntax	Input	TRVL < ***distance*** > <***speed***>
	Response	
Parameters:	***distance***	Distance of travel, in positions, entered as a signed (two's complement) 16-bit hex value. The range of allowed values is 8000 to 7FFF.
	speed	Speed, in positions per second, entered as a 16-bit hex value. The range of allowed values is 0 to FFFF
Details:	This command will cause the robot to travel along a straight line for the specified distance, ramping the wheels up to the maximum speed specified in this command. The rate of acceleration can be specified using the ACC command. The accuracy of the distance and straightness of travel is affected by the resolution of the wheel encoders, and by any slippage between the tires and the floor surface.	
Example: Travel in a straight line for 419 positions. Ramp the speed up to a maximum of 37 positions/second.		**`TRVL 1A3 25`**

TURN

Rotate in place by a specified angle (in degrees)

Syntax	Input	TURN <***angle***> <***speed***>
	Response	
Parameters:	***angle***	Angle of rotation (in degrees) is entered as a signed (two's complement) 16-bit hex value. The range of allowed values is 8000 to 7FFF. Negative values perform counterclockwise rotation.
	speed	Speed (in positions per second) is entered as a 16-bit hex value. The range of allowed values is 0 to FFFF
Details:		This command will cause the robot to rotate in place the specified number of degrees, ramping each wheel up to the maximum speed specified in this command. The rate of acceleration can be specified using the ACC command. The angle is used to calculate the number of positions to travel for each wheel. The accuracy of the angle rotated is affected by the resolution of the wheel encoders, and by any slippage between the tires and the floor surface.

Example:	
Rotate the robot in place, counterclockwise, by an angle of 271 degrees. And ramp the speed up to a maximum of 75 positions/second.	**`TURN FEF1 4B`**

ACC

Set the rate of acceleration or deceleration.

Syntax	Input	ACC <***rate***>
	Response	
Parameters:	***rate***	The rate of acceleration (in positions per second per second) is entered as a 16-bit hex value. The range of allowed values is 0 to FFFF.
Details:		The value entered for the rate of acceleration is applied whenever the controller is transitioning from one desired speed to another. The exception to this is the STOP command where the rate of deceleration is instead dictated by the stopping distance, and the GO command where the motors are driven by a power level instead of at a controlled speed.

Example:	
Set the rate of acceleration to 256 positions per second per second.	**`ACC 100`**

SPD

Get the current average speed for each wheel.

Syntax	Input	SPD
	Response	<***left***> <***right***>
Parameters:	***left***/***right***	The current average speed for each wheel (in positions per second) is returned as a signed (two's complement) 16-bit hex value.
Details:	Speeds are sampled over the previous half second.	
Example: The average speed for the left wheel is 181 positions per second and for the right wheel is41 positions per second.		**SPD** B5 29

HEAD

Get the current heading relative to start.

Syntax	Input	HEAD
	Response	<***heading***>
Parameters:	***heading***	The current heading (in degrees) is returned as a 12-bit hex value. The valid range of values is 0 to 360 degrees (or 0 to 168 in hex).
Details:	The current heading will be maintained as a relative angle from the initial heading, or from whenever the last RST (reset distance and heading) command was issued. The accuracy of the heading is determined by the resolution of the wheel encoders, and subject to slippage between the tires and the floor surface.	
Example: The current heading is 244 degrees.		**HEAD** **0F4**

DIST

Get the distance of each wheel from the start or reset position.

Syntax	Input	DIST
	Response	<***left***> <***right***>
Parameters:	***left***/***right***	The left and right accumulated distance values (as number of positions) are returned as signed (two's complement) 32-bit hex values.
Details:		The controller keeps track of how far (in positions) from the origin the wheels have traveled. Driving forward increases the distance value, and driving reverse decreases the value. The distance for each wheel can be reset by issuing the RST (reset distance and heading) command. The accuracy of the distance is determined by the resolution of the wheel encoders, and is subject to slippage between the tires and the floor surface.

Example:	
The left wheel has travelled 1,351,014,186 and the right wheel has travelled 13,534,095 positions from the origin.	**DIST** 5086D72A 00CE838F

RST

Reset distance and heading.

Syntax	Input	RST
	Response	
Details:		This command will reset the distance values for both wheels and the heading angle back to zero.

Example:	
Reset the distance values for both wheels and the heading to zero.	**RST**

Revision History

Version 1.0

Initial document release.

Version 1.1

Corrected GOSPD description in Command Set Summary

Fixed VERB command name in General Command Form section

Corrected example usage of HWVER command

Corrected description of the TRVL command

Clarified BLINK, IN, LOW, READ, and HEAD parameters.

Enhanced details descriptions for VERB, BLINK, SPNG, SGP, PING, HEAD, and DIST commands

Corrected or clarified Example descriptions in IN, OUT, LOW, INS, OUTS, LOWS, HIGHS, SGP, PING, ADC, GO, GOSPD, TURN, ACC, SPD, HEAD, DIST, and RST commands.