



SOLUTIONS CUBED

## RC MICRO PID Velocity Controller Data Sheet

Revision 1  
March 30<sup>th</sup>, 2004

Copyright © 2003 Solutions Cubed

---



---

## Table of Contents

<b>1.</b>	<b>Revision Log</b>	<b>3</b>	
<b>2.</b>	<b>Introduction</b>	<b>4</b>	
2.1	Description	4	
<b>3.</b>	<b>Engineering Specifications</b>	<b>5</b>	
3.1	Absolute Maximum Ratings	5	
3.2	DC Electrical Characteristics	5	
3.3	AC Electrical Characteristics	6	
3.4	Mechanical Dimensions	7	
3.5	Connectivity Overview	8	
3.6	Power Supply Pin Connections	9	
3.7	PID Filter Configuration and Settings	10	
3.8	PID Filter Tuning	12	
3.9	Radio Control (RC) Settings	12	
3.10	Velocity Settings	14	
3.11	Jog Controls	15	
3.12	Virtual Stop Limits	15	
3.13	User Selected Functions	16	
3.14	CHA and CHB Input Circuits	18	
3.15	Two's Complement Numbering System	18	
3.16	RS232 Conversion Circuitry	19	
3.17	VM Capacitor	19	
3.18	Fault Conditions	19	19
<b>4.</b>	<b>Operating Information</b>	<b>20</b>	
4.1	Overview	20	
4.2	Example Schematic	21	
<b>5.</b>	<b>Communication Protocol</b>	<b>22</b>	
5.1	Overview	22	
5.2	Command Set	23	
5.3	USER0, USER1, and STATUS Flags	30	
<b>6.</b>	<b>Quick Start</b>	<b>32</b>	

## List of Figures

Figure 1: Mechanical Dimensions	7
Figure 2: Mechanical Landmark Descriptions	7
Figure 3: RC MICRO PID Velocity Controller Pin Definitions	8
Figure 4: Power Supply Pin Connections	9
Figure 5: PID Programming Window for Solutions Cubed Software	10
Figure 6: PID Period Constants	11
Figure 7: RC Settings Descriptions	13
Figure 8: Graphical Representation of RC Pulse Measurements	13
Figure 9: Velocity Settings Descriptions	14
Figure 10: Virtual Stop Limit Settings Descriptions	15
Figure 11: User Functions Programming Window for Solutions Cubed Software	16
Figure 12: CHA and CHB Input Circuits	18
Figure 13: Two's Complement Examples	18
Figure 14: RS232 <-> TTL Conversion	19
Figure 15: System Block Diagram	20
Figure 16: Example Schematic	21
Figure 17: Velocity Controller Command Set	23
Figure 18: USER0 Bits	30
Figure 19: USER1 Bits	31
Figure 20: STATUS Bits	31

## 1. Revision Log

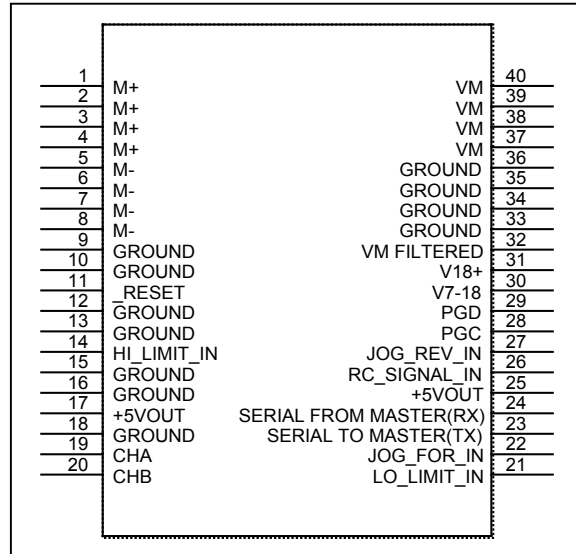
Date	Rev	Description
04-06-04	1	Original implementation

## 2. Introduction

# RC MICRO PID Velocity Controller

## FEATURES

- ◆ Up to 2A continuous current, up to 30VDC brushed motors
- ◆ Responsive motor speed/direction control RC (1-2ms pulse) input signal
- ◆ Programmable PID algorithm settings
- ◆ Built in RC signal test modes
- ◆ Virtual limit switch inputs with speed controlled buffer areas
- ◆ Custom software simplifies programming and testing
- ◆ X4 or X1 quadrature signal decoding
- ◆ Over-current/temperature protection with user enabled auto reset function
- ◆ Last valid signal mode allows controller to operate when RC signal is lost
- ◆ User selectable periodic position storage function



### 2.1 DESCRIPTION

The RC MICRO PID Velocity (RC\_UPID) Controller is a complete PID speed control module. Speed control input is accomplished with a standard Radio Control (RC) signal. When controlling a brushed DC motor equipped with a quadrature encoder, the RC\_UPID Velocity Controller forms a highly versatile, low-cost, speed control module, for low current servo motor control.

The RC\_UPID circuitry includes quadrature decoder circuitry, an on-board voltage regulator system, and an internal H-bridge with over-current, over-temperature, and under-voltage fault detection.

The primary mode of operation is motor speed control based on an RC signal input. There are a variety of configuration settings that the user can access in order to customize the device to a specific RC transmitter/receiver set. In general, a 1.5ms pulse width results in a stopped motor, 2ms would be full speed forward, and 1ms full-speed reverse, other values result in proportional speed control. The user, through a serial interface, can configure the pulse width settings, peak velocity, and a variety of other functions. The RC\_UPID has been optimized for smooth operation at low speeds and is highly responsive to abrupt direction and speed changes.

Additional features include a test mode that allows the user to “insert” a simulated RC signal into the controller via the serial interface, flag indicators that are set when an external RC signal is not present or is outside of the acceptable range of values, and a mode that allows the RC\_UPID to use the last valid signal received when invalid signals are received. Additionally the user may enable a function that stores the current position in non-volatile memory automatically.

The RC\_UPID is based on the hardware used for the UPID, and electrical connections are similar in nature.

The unit measures 2.5” x 1.05” x 0.6”. The package is a 40 pin DIP with 0.9” spacing between rows.

### 3. Engineering Specifications

#### 3.1 Absolute Maximum Ratings

*These are stress ratings only. Stresses above those listed below may cause permanent damage and/or affect device reliability. The operational ratings should be used to determine applicable ranges of operation.*

Storage Temperature	-55°C to +150°C
Operating Temperature	0°C to +150°C
Motor Voltage (VM)	-0.3V to 30.0V
Voltage on logic control pins	-0.3V to +5.5V
Voltage on CHA, CHB, pins	-0.3V to +5.3V
Voltage on VM, M+, M-	35V transient spike
Motor Current Load	10A peak

#### 3.2 DC Electrical Characteristics

At  $T_A = 25^\circ\text{C}$ ,  $V_{\text{MOTOR}} = 24\text{V}$ ,  $I_{\text{LOAD}} = 0.5\text{A}$  FPWM = 19.2kHz

Characteristic	Symbol	Min	Typ	Max	Unit	Notes
Logic Supply Voltage	+5VDC	4.5		5.5	V	
Logic Supply Current Source Capability	I5V		10	25	mA	5VOUT pins can supply a modest amount of current to external circuitry
Supply Current	IQ		20	50	mA	VM = 12V
Motor Voltage	VM	7		30	V	See section 3.6
CHA-B voltage	CHVOLT	-0.3		+5.3	V	CHA-B pins pulled to +5V with 10kΩ resistors
Peak load current	IPK			10	A	
Max current no cooling	INOCOOL		2		A	Tested at 12V 95% duty-cycle
Max current fan cooled (6.2CFM)	ICOOLED		2.75		A	Tested at 12V 95% duty-cycle
Intermittent current fan cooled (6.2CFM)	IINT		5		A	Tested at 12V 95% duty-cycle for 0.5S, 0% for 2S
Low Level Input RX pin	VRXIL			0.5	V	RX pin pulled to +5V with 10kΩ resistor
High Level Input RX pin	VRXIH	2.0			V	RX pin pulled to +5V with 10kΩ resistor
Low Level Input JOG-LIMIT pins	VJOGIL			0.5	V	
High Level Input JOG-LIMIT pin	VJOGIH	2.0			V	

note: "Typ" values are for design guidance only and are not guaranteed

**DC Electrical Characteristics (continued)**At  $T_A = 25^\circ\text{C}$ ,  $V_M = 24\text{V}$ ,  $I_{\text{LOAD}} = 0.5\text{A}$   $F_{\text{PWM}} = 19.2\text{kHz}$ 

Characteristic	Symbol	Min	Typ	Max	Unit	Notes
Low Level Input _RESET pin	VRSTIL			0.5	V	_RESET pin pulled to +5V with 10k $\Omega$ resistor
High Level Input _RESET pin	VRSTIH	2.0			V	_RESET pin pulled to +5V with 10k $\Omega$ resistor
Low Level Input CHA-B pins	VCHIL			0.8	V	CHA-B pins pulled to +5V with 10k $\Omega$ resistors
High Level Input CHA-B pins	VCHIH	3.5			V	CHA-B pins pulled to +5V with 10k $\Omega$ resistors
Low Level Input RC_SIGNAL pin	VRCIL			0.5	V	RC signal input pulled to +5V with 12.5k $\Omega$ res.
High Level Input RC_SIGNAL pin	VRCIH	2.0			V	RC signal input pulled to +5V with 12.5k $\Omega$ res.
Low Level Output TX pin	VTXOL			0.6	V	TX pin pulled to +5V with 10k $\Omega$ resistor
High Level Output TX pin	VTXOH	3.8		4.8	V	TX pin pulled to +5V with 10k $\Omega$ resistor

note: "Typ" values are for design guidance only and are not guaranteed

**3.3 AC Electrical Characteristics**At  $T_A = 25^\circ\text{C}$ ,  $V_M = 24\text{V}$ ,  $I_{\text{LOAD}} = 0.5\text{A}$   $F_{\text{PWM}} = 19.2\text{kHz}$ 

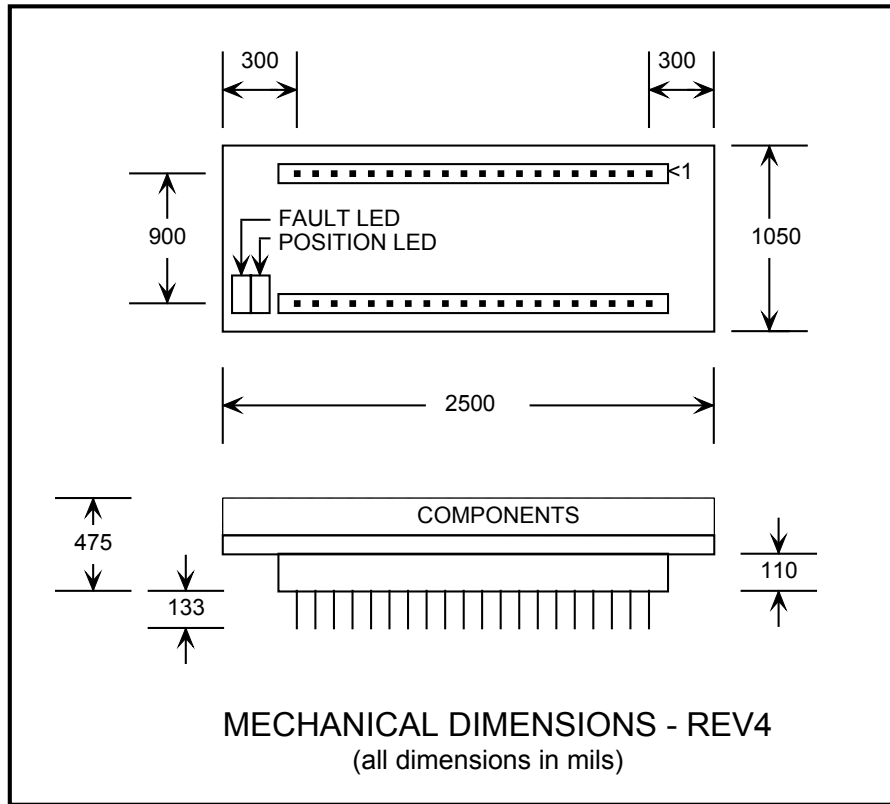
Characteristic	Symbol	Min	Typ	Max	Unit	Notes
Communication bit period	TBIT		104		$\mu\text{S}$	9600bps
Time for a command to be responded to	TTURN	2		40	mS	
Time after power-up before device will communicate	TPWRUP		1000	4000	mS	The onboard microcontroller blinks the P LED 4 times on power up (see section 6)
$V_M$ rise time to ensure good reset	SVM	0.05			V/mS	If this condition is not met then microcontroller may not power up correctly
PWM update rate	PPWM	600	1200	1477	Updates/S	See figure 6
PWM Resolution	RPWM		1000		bits	
PWM frequency	FPWM		19.2		kHz	
Encoder Frequency	FENC		1.5	1.75	MHz	
Thermal shutdown		150	175	200	$^\circ\text{C}$	
Thermal turn-on		120		170	$^\circ\text{C}$	
Short-circuit current limit				20	A	
Over-current shutdown		6	8	10	A	Ratings for 150 $^\circ\text{C}$ , 25 $^\circ\text{C}$ , and -40 $^\circ\text{C}$

note: "Typ" values are for design guidance only and are not guaranteed

**3.4 Mechanical Dimensions**

The following diagram may be used to develop PCB carrying boards or enclosures used to fit the RC\_UPID into custom designs. All electrical pins are spaced at 0.1". Two 20-pin SIP solder-tail sockets spaced 0.9" apart would work well on a carrier board for the RC\_UPID. The Mill-Max 7000 series (Digi-Key PN: ED7020-ND) is one example of this kind of part.

**Figure 1: Mechanical Dimensions**



**Figure 2: Mechanical Landmark Descriptions**

Landmark	Type	Description
P-LED	LED – green	Velocity LED, lit when the velocity read from incremental encoder matches the commanded velocity, also blinks on power up to indicate that the device is operating.
F-LED	LED – red	Was the error indicator LED on the UPID. This input is now used for the jog forward input control and will light when the jog forward input is asserted (pulled to ground).

### 3.5 Connectivity Overview

The DC motor and quadrature encoder connections are made through the 40 pin DIP connections. The individual pin designations are defined below.

**Figure 3: RC\_UPID Pin Definitions**

Pin	Name	Type	Description
1	M+	POWER	The positive lead of the brushed DC motor, or other load
2	M+	POWER	The positive lead of the brushed DC motor, or other load
3	M+	POWER	The positive lead of the brushed DC motor, or other load
4	M+	POWER	The positive lead of the brushed DC motor, or other load
5	M-	POWER	The negative lead of the brushed DC motor, or other load
6	M-	POWER	The negative lead of the brushed DC motor, or other load
7	M-	POWER	The negative lead of the brushed DC motor, or other load
8	M-	POWER	The negative lead of the brushed DC motor, or other load
9	GND	POWER	Ground return
10	GND	POWER	Ground return
11	_RESET	INPUT	Pulling _RESET low performs a hardware reset of the RC_UPID, this pin should be left unconnected if not used
12	GND	POWER	Ground return
13	GND	POWER	Ground return
14	HI_LIMIT_IN	INPUT	High virtual stop limit switch input. This input should be pulled to +5V with a 100K $\Omega$ resistor. When grounded the high stop limit function is enabled.
15	GND	POWER	Ground return
16	GND	POWER	Ground return
17	+5VOUT	POWER	+5VDC output, can supply up to 25mA, should be left unconnected if not needed
18	GROUND	POWER	Ground return
19	CHA	INPUT	Channel A input signal from quadrature incremental encoder, this input is pulled to +5VDC with a 10k $\Omega$ resistor, and is protected with a 5.6V zener diode
20	CHB	INPUT	Channel B input signal from quadrature incremental encoder, this input is pulled to +5VDC with a 10k $\Omega$ resistor, and is protected with a 5.6V zener diode
21	LO_LIMIT_IN	INPUT	Low virtual stop limit input. This input is pulled to +5V with a 10K $\Omega$ resistor on the RC_UPID board. When grounded the low stop limit function is enabled.
22	JOG_FOR_IN	INPUT	Jog forward function input. This input should be pulled to +5V with a 100K $\Omega$ resistor on the RC_UPID board. When grounded the jog forward function is enabled
23	TX	OUTPUT	TTL level, 8N1, serial transmission pin (data to the Master unit)
24	RX	INPUT	TTL level, 8N1, serial reception pin (data from the Master unit)
25	+5VOUT	POWER	+5VDC output, can supply up to 25mA, should be left unconnected if not needed
26	R/C SIGNAL_IN	INPUT	The 1-2ms RC receiver signal should be applied here. This input is pulled to +5V with a weak internal pull-up resistor. The signal may need to be buffered if the R/C receiver puts out a voltage level less than 5V when high.
27	JOG_REV_IN	INPUT	Jog reverse function input. This input is pulled to +5V with a weak internal pull-up resistor on the RC_UPID board. When pulled to ground the jog reverse function is enabled
28	NO_CONNECT	NC	This pin should be left unconnected
29	NO_CONNECT	NC	This pin should be left unconnected
30	V7-18	POWER	Connect this pin to VM_IN if your motor voltage is between +7 and +18VDC
31	V18+	POWER	Connect this pin to VM_IN if your motor voltage is greater than 18VDC
32	VM_IN	POWER	Filtered motor voltage connects to either pin 30 or 31
33	GND	POWER	Ground return
34	GND	POWER	Ground return
35	GND	POWER	Ground return
36	GND	POWER	Ground return
37	VM	POWER	The primary motor supply voltage connects to this pin (30V maximum)
38	VM	POWER	The primary motor supply voltage connects to this pin (30V maximum)
39	VM	POWER	The primary motor supply voltage connects to this pin (30V maximum)
40	VM	POWER	The primary motor supply voltage connects to this pin (30V maximum)

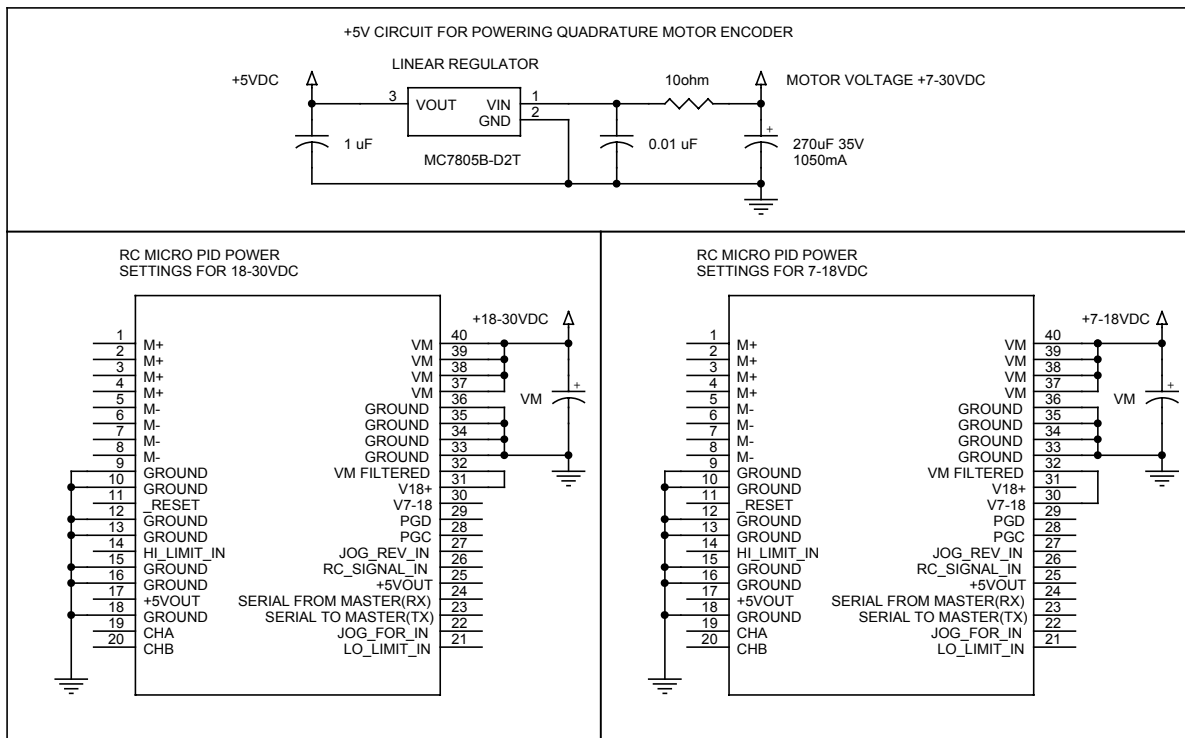
### 3.6 Power Supply Pin Connections

The RC\_UPID carries two linear regulators on board. The motor voltage at pins 37-40 is filtered and limited with a transient voltage suppressor (TVS). The filtered motor voltage is then fed out of the VM\_IN pin (pin 32). To reduce the power dissipation in the linear regulators you must connect the filtered motor voltage to either the V7-18 pin (pin 30) or V18+ pin (pin 31).

For motor voltages of +18-30VDC connect the VM\_IN pin to the V18+ pin (pin 31 to pin 32). For motor voltages of +7-18VDC connect the VM\_IN pin to the V7-18 pin (pin 30 to pin 32).

The +5VOUT pins on the RC\_UPID can provide a limited amount of current to external circuits (about 25mA). In general it cannot provide enough current to power the quadrature encoder attached to your motor. These encoders typically require 50-250mA at 5V. Therefore you should expect to provide a separate +5V supply to your encoder's 5V input. A linear regulator like the one shown below can provide the current required. It is very important to add the VM capacitor (shown as a 270uF capacitor below) to provide instantaneous current to the motor, so current is not drawn from the 0.01uF filter capacitor through the 10Ω resistor.

**Figure 4: Power Supply Pin Connections**

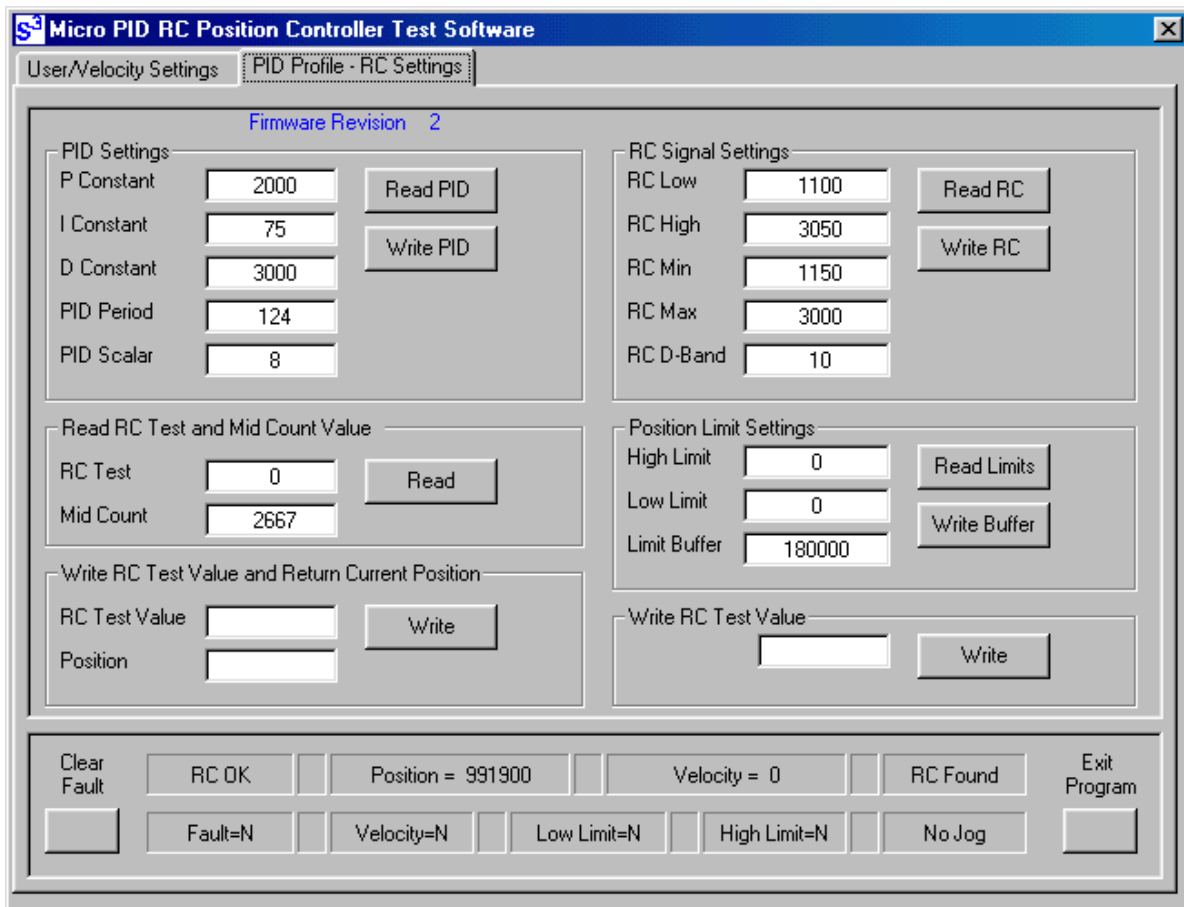


### 3.7 PID Filter Configuration and Settings

The RC\_UPID makes use of a proportional (“P”), integral (“I”), and derivative (“D”), or PID, filtering technique to effect changes in its output PWM signal relative to the error signal generated by the quadrature encoder connected CHA and CHB pins. The user inputs a desired velocity and an error signal is generated based on the difference between the desired velocity and the motor velocity.

Programming the PID settings can be simplified by using the test software provided by Solutions Cubed. The PID filter programming window of the Solutions Cubed software is displayed below.

**Figure 5: PID Programming Window for Solutions Cubed Software**



#### 3.7.1 Proportional Gain (can range from 0 to 32767)

The proportional aspect of the filter is simply the velocity error multiplied by the proportional constant. The “P” portion of the PID is responsible for the gross movement of the motor. Using proportional gain solely will typically result in a control response that undershoots or overshoots the desired velocity. As the motor velocity approaches the desired velocity the error signal becomes too small for the proportional gain to generate a significant PWM output, and therefore continue moving the motor.

**3.7.2 Integral Gain (can range from 0 to 32767)**

The integral aspect of the filter is a continued summation of the velocity error multiplied by the integral constant. The “I” portion of the PID is used to amplify small errors over time and can nudge a motor into the desired velocity. Overly large “I” settings will cause oscillations to occur around the desired velocity. The “I” setting is typically very small.

The integral sum can become “saturated” and swamp out the P and D portions of the PID in instances where the motor velocity cannot match the desired velocity. If the PWM output reaches a maximum of +/-97% duty-cycle then the output of the PID filter is considered “saturated”. When the output is “saturated” the integral portion of the PID filter is skipped until the PWM output is reduced below +/-97%. This prevents “integral wind up” which has the effect of swamping out the “P” portion of the filter. Under normal operation the integral summation value is prevented from exceeding +/-4,095 encoder counts.

The “saturation” limits that occur when the motor speed is maximized can be reduced through the USER0 register by setting the Disable Saturation Limit bit. With this bit set the integral sum continues to build even when the PWM output is full scale, and the error sum is allowed to reach as high as +/-60,000 encoder counts.

**3.7.10 Derivative Gain (can range from 0 to 65535)**

This portion of the PID causes “drag” on the motor during start-up. The derivative gain constant is multiplied by the difference between the current velocity error (desired velocity – motor velocity) and a previous velocity error. In many PID algorithms the last velocity error is used (derivative term = derivative gain X (current error – last error)), but the RC\_UPID slightly modifies this method to increase the effectiveness of the “D” term of the PID algorithm. The actual algorithm used for the “D” term is (derivative term = derivative gain X (error<sub>(N)</sub> – error<sub>(N-15)</sub>)). Adding the “D” term will reduce overshoot and increase rise time with respect to a step function. The “D” term will only impact operation of the motor controller when the motor is in motion.

**3.7.10 PID Period (accepts specific values from 0 to 255)**

The PID Period register is used to specify the number of PID updates that will occur during a 1-second interval. The PID Period register also defines the time-base used to measure a motor’s velocity. The following settings are the only acceptable values for use with the RC\_UPID. If the USER1 Double Count bit is set, then the velocity measurement and PID updates will occur every 2 PID periods. This can be beneficial for motors with low counts-per-revolution encoders (CPR).

**Figure 6: PID Period Constants**

PID Period Constant	PID Period (Seconds)	Updates Per Second	Updates per second – USER0 Double Count bit set	PWM Frequency
100	0.677E-03	1477	739	19.2KHz
108	0.729E-03	1372	686	19.2KHz
116	0.781E-03	1280	640	19.2KHz
124	0.833E-03	1200	600	19.2KHz

**3.7.10 PID Scalar (accepts values from 0 to 31)**

The PID filter generates a 32-bit value that is used as the PWM duty-cycle output. In most cases the PID output is much larger than the +/-1,000 (+/-97% duty-cycle) limit used to generate the motor drive signal. The PID scalar is the number of divide-by-two’s used to scale the PID filter down to a usable level. Changing this value will require that the PID be tuned again. Small values would be used with very low CPR encoders where the error signals generated are never large. In most cases the default value of 8 may be used and the P, I, and D constants may be adjusted for better motion control.

### 3.8 PID Filter Tuning

Tuning the PID filter is typically a process of trial and error. The PID filter tuning process is dependent on your mechanical system, the electrical characteristics of your motor, and its quadrature encoder. Much of the trial and error will be based on the allowable velocity error, overshoot, and response time, required by your mechanical system. PID filter tuning can be greatly simplified with software available at [www.solutions-cubed.com](http://www.solutions-cubed.com), a PC, and the ICON Adapter Board (for connecting a PC to the RC\_UPID).

It is recommended that the 4X Mode be enabled unless you are already using a high-count encoder (1024 CPR). Since this is a velocity control system the error signals can be fairly small and 4X Mode gives the RC\_UPID a little more error signal to work with. You can start tuning the PID by loading a 0 into the I and D constant values (this will remove them from the PID calculation). Leave the PID scalar at 8 and the PID Period to 124 to start with. Increase the P constant until you have gross speed control at higher commanded velocities (control at slow speeds will likely be “choppy”). The P value is typically in the thousands. Then increase the integral (I) constant from 0 to a value in the tens to low hundreds. If the value is too low then slow speed control will be choppy, if set too high then the system will be unstable and will chatter. Tuning the PID values with the RC Test function enabled will eliminate noise from poor radio reception, but will make response to direction changes hard to judge. Since the velocity error is typically small, the derivative (D) constant generally has a limited effect on performance. It will need to be larger than the proportional (P) constant to have the effect of increasing drag on the abrupt changes in commanded velocity. Overly large derivative constants may also cause instability in the control system. In many systems the D constant can be omitted.

### 3.9 Radio Control (RC) Settings

An RC signal, such as those used with remote control cars and airplanes, is the primary control signal used by the RC\_UPID to generate the desired motor velocity. The PID filter then processes the difference between the desired motor velocity and the actual motor velocity (measured from the encoder) to generate a motor drive signal. The RC\_UPID must be loaded with parameters that allow it to determine if an R/C signal is valid.

A typical RC signal is a positive pulse (5V) 1ms-2ms in length and is repeated every 20ms. When not at 5V the RC signal should be 0V. The RC\_UPID will wait for this pulse for up to 53ms. When a pulse is received it is read with a time-base of 0.814us. The time-base is important as it is used to determine the R/C parameters that must be programmed into the RC\_UPID. For example, a 1ms pulse width is counted as  $1000\text{us} / 0.814\text{us} = 1229$ .

1.5ms is generally considered the mid-point of this signal and relates to a commanded velocity of 0 (or no motor movement) while 1ms would relate to full speed reverse and 2ms full speed forward (more on velocity settings later). With the default settings R/C signal pulse widths between 0.5ms (RC Low) and 2.5ms (RC High) are accepted. The commanded (desired) velocity settings are proportional between the RC Min and RC Max settings (default of 1.0ms and 2.0ms respectively). For example, for forward velocities the commanded velocity is determined by this equation.

$$(\text{MaxVelocity}) * \frac{(\text{PulseWidth} - (\text{RCMidPoint} + \text{RCDeadBand}))}{(\text{RCMax} - (\text{RCMidPoint} + \text{RCDeadBand}))} = \text{DesiredVelocity}$$

Example: if your Max Velocity setting was 120 and a 1.75ms ( $1.75\text{ms}/0.814\text{us} = 2150$ ) pulse was received then the desired velocity would be roughly  $\frac{1}{2}$  the maximum velocity. If the RC Dead Band was 0 then the desired velocity would be exactly  $\frac{1}{2}$  the maximum.

$$(120) * \frac{(2150 - (1843 + 20))}{(2458 - (1843 + 20))} = 58 \text{ pulses - per - PID - period}$$

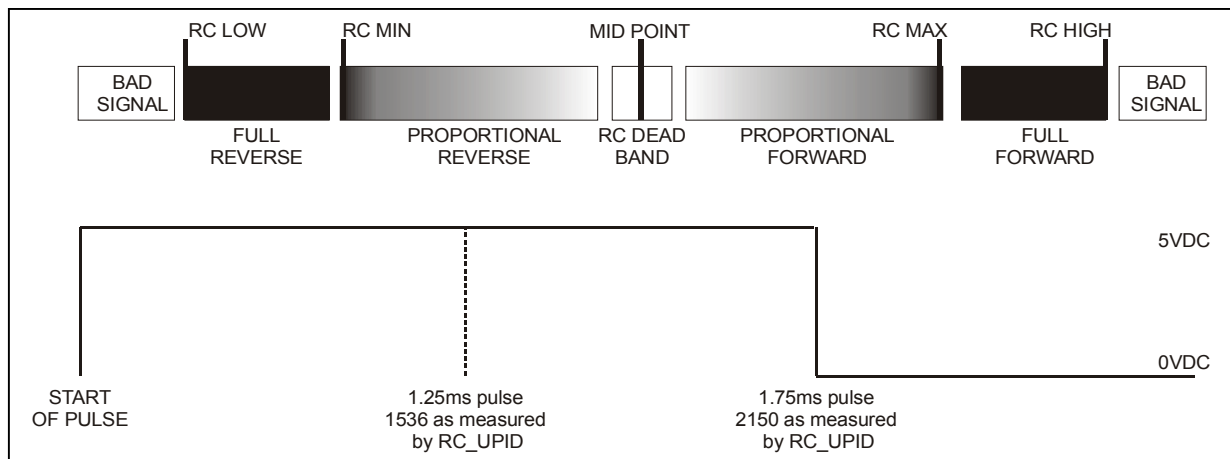
Reverse (negative) velocities are determined in a similar manner.

**Figure 7: RC Settings Descriptions**

Parameter	Description	Default Value
RC Low	Lowest pulse width allowed for RC signal to be considered valid. Signals between RC Low and RC Min are considered to be RC Min in length. Signals shorter than RC Low are considered to be 0 in length.	615 – 0.5ms
RC High	Highest pulse width allowed for RC signal to be considered valid. Signals between RC High and RC Max are considered to be RC Max in length. Signals longer than RC High are considered to be 0 in length.	3073 – 2.5ms
RC Min	Signals between RC Min and (RC Mid Point – RC Dead Band) generate commanded velocity values that are proportional to the Max Velocity setting. Therefore, a pulse width of RC Min would be full reverse (assuming the motor can attain this velocity).	1229 – 1.0ms
RC Max	Signals between RC Max and (RC Mid Point + RC Dead Band) generate commanded velocity values that are proportional to the Max Velocity setting. Therefore, a pulse width of RC Max would be full forward (assuming the motor can attain this velocity).	2458 – 2.0ms
RC Dead Band (RC D-Band)	The RC Dead Band setting denotes a band around the RC Mid Point that results in a commanded velocity of 0. If the RC dead band is set to 0 then RC pulse noise can cause motor movement even when the user is not adjusting the RC signal.	20 – 16us
RC Mid Point	The RC Mid Point is the center point where the RC pulse width. This RC value cannot be programmed and defaults to 1843 (1.5ms). However if the user enables the Auto Calibrate RC Mid Point function then the RC pulse width measured on power up can be auto-loaded as the RC Mid Point	1843 – 1.5ms (see also Auto Calibrate RC Mid Point function)

Figure 8 can help clarify how the RC Settings relate to the R/C signal's pulse width. This diagram shows graphically where two R/C signals fall in relation to the RC Settings. The first is a 1.25ms pulse that relates to ½ speed reverse, and a 1.75ms pulse that relates to ½ speed forward. The lighter the color the lower the commanded velocity.

**Figure 8: Graphical Representation of RC Pulse Measurements**



### 3.10 Velocity Settings

There are two velocity related settings that may be programmed by the user of the RC\_UPID. The first is the Jog Velocity, and it relates to the motor speed when either the Jog Forward or Jog Reverse inputs are asserted. The jog functions are discussed later. The second velocity value, Max Velocity is of greater importance in the operation of the RC\_UPID. The Max Velocity setting is used in conjunction with the RC Settings to determine the desired motor velocity (See section 3.9).

Velocity measurements are calculated as encoder pulses-per-PID-period. Keep in mind that enabling 4X Mode multiplies the actual pulse count by 4, and enabling the Extend PID Period function doubles the period of time that the encoder pulses are accumulated. Both of these functions as well as the PID Period (figure 6) need to be used to determine the actual motor speed as it relates to the velocity reading of the RC\_UPID.

Assuming the encoder is connected to the rear of the motor, prior to any motor gearing, the equation that relates the velocity pulse count to the motor output shaft speed is as follows.

$$\text{MotorShaftFrequency} = \frac{\text{VelocityMeasurement} * \text{PIDUpdatesPerSecond}}{\text{EncoderCPR} * \text{GearRatio}}$$

For example, assuming a velocity count of 40, a PID Period setting of 124 (1200 updates per second), a 512CPR (counts-per-revolution) encoder, and a 5:1 gear ratio, the motor output shaft would rotate at...

$$18.75\text{Hz} = \frac{40 * 1200}{512 * 5}$$

If the 4X Mode were enabled (and the measured velocity is still 40) then the motor shaft would be rotating at...

$$4.69\text{Hz} = \frac{40 / 4 * 1200}{512 * 5}$$

To achieve greatest resolution of velocity control the Max Velocity should be set to a value close to the actual top speed that the motor can achieve. For faster response times the Max Velocity may be set to a value that greatly exceeds the speed the motor can achieve. For example, if through experimentation, you determine that the motor can reach +/-120 encoder counts per PID update period you could set the Max Velocity to 240. This would cause the commanded motor speed to reach the motor's top velocity with an input signal that was 1/2 of full scale.

**Figure 9: Velocity Settings Descriptions**

Parameter	Description	Default Value
Jog Velocity	This is the commanded (desired) motor velocity when the jog forward or jog reverse mode is activated. This number is always positive.	120 encoder counts per PID update period
Max Velocity	The max velocity is used in conjunction with the RC settings and measured RC pulse width to determine the commanded motor velocity. If set above the peak motor speed, response time to RC signal changes is shorter. If set below the peak motor speed then motor speed will be limited to this value. Set the max velocity value to the motors peak speed to get the best resolution for speed control. This number is always positive.	320 encoder counts per PID update period

### 3.11 Jog Controls

Pins 22 (JOG\_FOR\_IN - forward) and 27 (JOG\_REV\_IN - reverse) are used to enable the jog mode of operation. Either forward or reverse jog control become active when the associated input pin is pulled to ground. When jog control is active the motor will slew (either forward or reverse) at the motor speed dictated by the Jog Velocity setting. If unused, these inputs should be tied to +5VDC. Jog control overrides any RC signal received on pin 26 (RC\_SIGNAL\_IN).

If both jog control inputs are pulled to 0V at the same time then the jog forward function takes priority. The jog forward pin (22) must be pulled to +5VDC with an external resistor (100KΩ).

### 3.12 Virtual Stop Limit

The RC\_UPID has virtual stop limits that the user can enable by pulling the associated input to 0V. Pins 21 (LO\_LIMIT\_IN) and 14 (HI\_LIMIT\_IN) are used to set the stop limit positions. The limit position will not be recorded unless **both** of the following are true. First the motor must be stopped, and second the associated stop limit input must be pulled to 0V. If both of these conditions are met then the current motor position will be loaded into EEPROM and motor velocity will be limited when the motor is approaching the limit position. The Limit Buffer setting defines an area around the limit position that causes the desired motor speed to be limited. If the stop limit input is returned to 5V then the position and velocity limiting functions are disabled.

The following example describes how the virtual stop limit function may be used. First, both stop limit inputs must be at +5VDC. If you move the motor to position +100,000 and stop the motor, and then pull the HI\_LIMIT\_IN pin to 0V, +100,000 will be set as the high limit position. Then if you move the motor to position -100,000, stop the motor, and pull the LO\_LIMIT\_IN pin to 0V, the position -100,000 will be used to define the lower limit position. Assume the Limit Buffer is set to 50,000. Now, whenever the motor is moved in the forward direction from +50,000 to +100,000 the maximum motor velocity allowed would decrease proportionately from the Max Velocity (user defined) at +50,000 to 0 at +100,000. This proportional decrease in velocity can give you a soft landing at the high limit position. Similar motor speed limits are placed on the motor when running in reverse towards the lower limit position (between -50,000 and -100,000). Velocity limits are not placed on the motor if it is within the limit position area and moving away from the limit position. In other words, if you've reached the low limit position by moving in reverse, and then move the motor forward the motor speed is not limited (as it is moving away from the stop limit).

The limit positions (high or low) may both be positive position values or negative position values, but the high limit position must be greater than the negative position value. To maintain the stop limits after power is removed from your system you will need to enable the Enable Position Save mode (see section 3.13 User Defined Functions). If Enable Position Save mode is not used then on power up, if pins 21 and/or 14 are at 0V then the limit positions will be set to 0 and the motor will not be allowed to move.

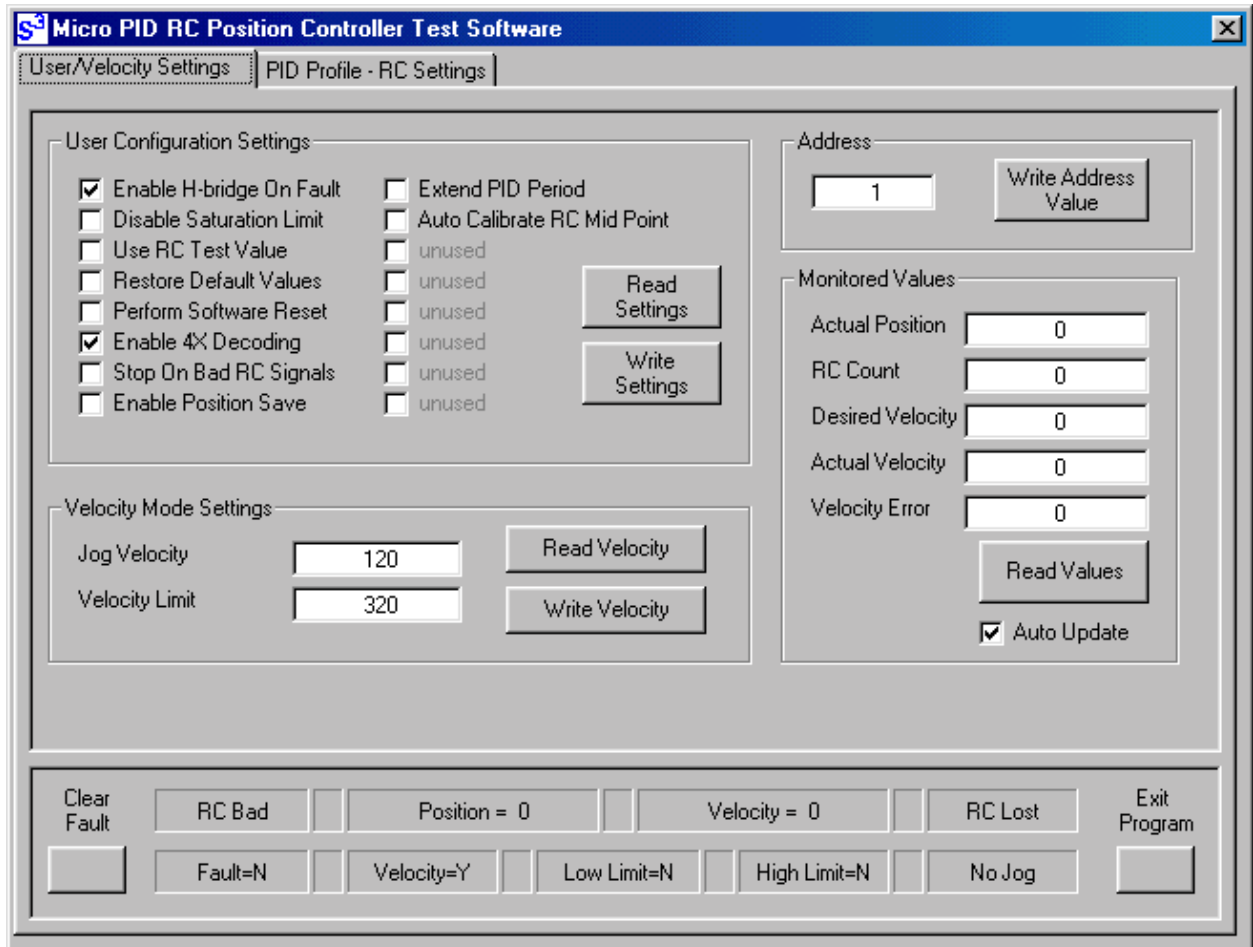
**Figure 10: Virtual Stop Limit Settings Descriptions**

Parameter	Description	Default Value
Buffer Limit	This value is the band around the stop limit positions that causes the motor velocity to be reduced. It is always a positive value.	180000

### 3.13 User Selected Functions

There are a variety of user selectable functions that may be enabled via the serial data interface of the RC\_UPID. A simple method of enabling these functions is with the test software provided at [www.solutions-cubed.com](http://www.solutions-cubed.com), otherwise a custom software solution can be created. Setting/clearing specific bits enable/disable (respectively) the user functions.

Figure 11: User Functions Programming Window for Solutions Cubed Software



#### 3.13.1 Enable H-bridge On Fault – USER0 bit 0

The RC\_UPID H-bridge may enter a fault condition if components of the H-bridge are short-circuited, if the temperature exceeds 150°C, if the motor voltage drops below 6V, or if the motor current exceeds 6-10A (temperature based). If this condition is persistent the RC\_UPID will disable the H-bridge. Under normal conditions the user would have to send a *Clear Fault* command via the serial interface to re-enable the H-bridge. If the user sets the Enable H-bridge On Fault bit of the USER0 registers, then the RC\_UPID will automatically re-enable the H-bridge 25 PID update periods after a fault is registered.

#### 3.13.2 Disable Saturation Limit – USER0 bit 1

When enabled this function removes some of the protection from integral “wind-up” when the motor is already being driven at full speed. More information on this function can be found in section 3.7.2.

**3.13.10 Use RC Test Value – USER0 bit 2**

During testing it may be useful to determine whether motor noise can be attributed to radio signal noise, mechanical shortcomings, or limitations of the RC\_UPID. When the Use RC Test Value function is enabled the RC\_UPID will ignore signals received on the RC\_SIGNAL\_IN pin, and use instead the RC Test Value provided by the *Write RC Test Value* command. This value should be between 0 and 32,767 and represents a measured pulse width. The RC Test Value time-base is the same as the time-base of the other RC settings and equals 0.814us (see section 3.9). If you wanted to insert a test signal of 1.5ms you would send 1843 as the RC Test Value. Inserting RC Test Values and comparing motor smoothness to the control exhibited when the radio is providing the control signal can identify noisy radio systems.

**3.13.4 Restore Default Values – USER0 bit 3**

When this bit is set the user programmable registers (including USER0 and USER1) are reset to their default values. The new values are also written to EEPROM.

**3.13.5 Perform Software Reset – USER0 bit 4**

The microcontroller on board the RC\_UPID will reset itself whenever this bit is set.

**3.13.6 Enable 4X Decoding – USER0 bit 5**

The user may select 4X or 1X decoding of the quadrature encoder signals by setting or clearing a bit in the USER0 register. In 4X mode each set of signals from the quadrature encoder is decoded into 4 pulses. In 1X mode each set of signals from the quadrature encoder is decoded into 1 pulse. Therefore a 500CPR encoder could effectively be used as a 2000CPR encoder if the RC\_UPID was operating in 4X mode. 4X mode may be used to increase resolution in position and velocity control. It can also be used to increase the velocity error signal allowing smaller movement distances to be implemented without using overly large PID constants.

**3.13.7 Stop On Bad RC Signals – USER0 bit 6**

When this bit is set the RC\_UPID will stop the motor whenever pulses at RC\_SIGNAL\_IN (pin 26) are shorter than the RC\_LOW setting, longer than the RC\_HIGH settings, or if a pulse is not received within 53ms. For radios that miss pulses regularly, or experience a lot of noise, this bit should remain cleared. When cleared the last valid RC signal received is used until the next valid signal is received.

**3.13.8 Enable Position Save – USER0 bit 7**

The current motor position can be saved and stored in EEPROM when this function is enabled. When enabled the RC\_UPID will monitor a counter and if the desired velocity remains 0 for 65,000 PID update periods the current motor position will be stored in EEPROM. With a PID update rate setting of 124 this position save will occur after 54 seconds of continuous commands equating to a desired velocity of 0. For noisy radio systems it might be beneficial to turn off the radio and allow the system to set for 1 minute to ensure that no spurious signals cause motor movement (and therefore reset the counter that determines when storage should occur). Any time EEPROM is written to by the RC\_UPID, the time-base used to maintain velocity control is upset. Therefore it is possible that a glitch or jitter could occur when a position save occurs. If this cannot be tolerated in your system it is recommended that you not use this function.

**3.13.9 Extend PID Period – USER1 bit 0**

For systems with lower resolution encoders it may be beneficial to enable this function. When enabled, the RC\_UPID will count velocity pulses for 2 PID update periods prior to calculating the error signal and implementing the PID based control of motor speed. This should allow for lower velocities to be controlled, but at the expense of slower response times.

**3.13.10 Auto Calibrate RC Mid Point– USER1 bit 1**

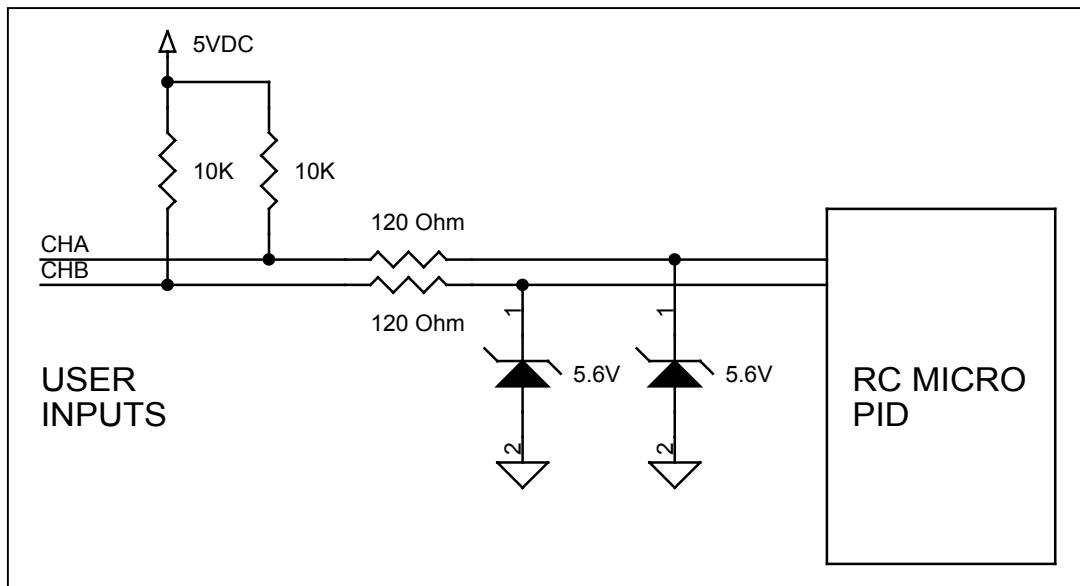
The RC Mid Point (associated with RC signals of 1.5ms in duration) defaults to a value of 1843 (with a time-base of 0.814us/count). The user may not program this value as they do the RC Min, RC Max, etc. However if the user enables this function, then the RC\_UPID will use the signal present on power-up as the new RC Mid Point. The RC\_UPID will wait for 20 consecutive “good” signals, and then take the 21<sup>st</sup> measurement and load it as the RC Mid Point value. All velocity control calculations are then used based on this value. A good RC signal is one that falls within the RC Low and RC High pulse duration boundaries.

Regardless of the actual input signal’s duration, the auto-calibrated mid point will be limited to values less than the RC Max setting minus the RC Dead Band and greater than the RC Min setting plus the RC Dead Band setting.

**3.14 CHA and CHB Input Circuits**

The protection and filtering circuits used by the RC\_UPID could adversely affect some quadrature encoders. The input protection circuits are detailed here for reference.

**Figure 12: CHA and CHB Input Circuits**



**3.15 Two’s Compliment Number System**

The RC\_UPID can accept negative numbers for some settings. In order to generate a two’s compliment negative value, take the binary or hexadecimal representation of the absolute value of the number, compliment it (every 1 becomes a 0 and every 0 becomes a 1) and add 1 to the result.

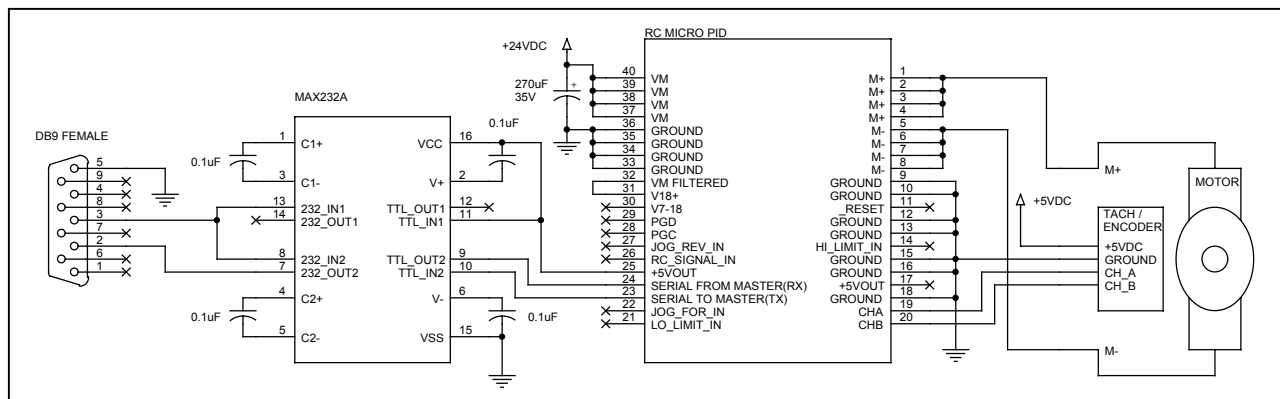
**Figure 13: Two’s Compliment Examples**

Number	Absolute Value	Hexadecimal	Compliment	Two’s Compliment
-1	1	0x00000001	0xFFFFFFF	0xFFFFFFF
-32768	32768	0x00008000	0xFFFF7FFF	0xFFFF8000
-100000	100000	0x000186A0	0xFFFE795F	0xFFFE7960

### 3.16 RS232 Conversion Circuitry

The primary method of programming and interfacing to the RC\_UPID is via a 9.6KBPS serial interface. The RC\_UPID supports TTL level serial communication (0V = logic "0" while +5V = logic "1"). Personal computer serial ports and many other serial devices operate on RS232 electrical specifications. RS232 serial data is transmitted under the same logic principles (therefore the data format does not change) but in order to increase noise immunity the RS232 electrical specification uses voltages from +10V (a logic "0") to -10V (a logic "1"). Therefore in order to send data to the RC\_UPID from a PC additional circuitry converting serial data from RS232 to TTL and back is required. This can be done easily with a single IC such as the MAX232 or HIN232.

Figure 14: RS232 <-> TTL Conversion



Interfacing to the RC\_UPID can be further simplified by downloading free test software at [www.solutions-cubed.com](http://www.solutions-cubed.com)

### 3.17 VM Capacitor

In some applications it may be necessary to provide additional capacitance between the VM and GND motor connections. This is typically true when you have long power supply leads, and your motor is drawing substantial current during stops or starts. If your RC\_UPID suffers from reset conditions, try connecting a large electrolytic capacitor (270uF+) across the VM and GND pins at the screw terminal connector. The capacitor should have a ripple current rating of at least  $\frac{1}{2}$  of the peak load current.

### 3.18 Fault Conditions

The H-bridge on board the RC\_UPID has built in over-current and over-temperature fault conditions. This component has an output that denotes short-circuits, under-voltage, over-current, and over temperature conditions. The H-bridge flag is monitored by the RC\_UPID and is mirrored in the STATUS register.

The fault flag may be cleared with the **Clear Fault** command. Setting Retry Fault bit in the USER0 register to a "1" will cause the controller to clear the fault flags internally after 25 PID update periods have elapsed.

## 4. Operating Information

### 4.1 Overview

The RC\_UPID can provide position and velocity control of a brushed DC motor that is equipped with a 2-channel quadrature encoder. The jog controls, reset, and virtual stop limit inputs should be pulled to +5V with 100K $\Omega$  resistors (to reduce part count some pull-up resistors may be omitted, check figure 3 for input pins that are already pulled high).

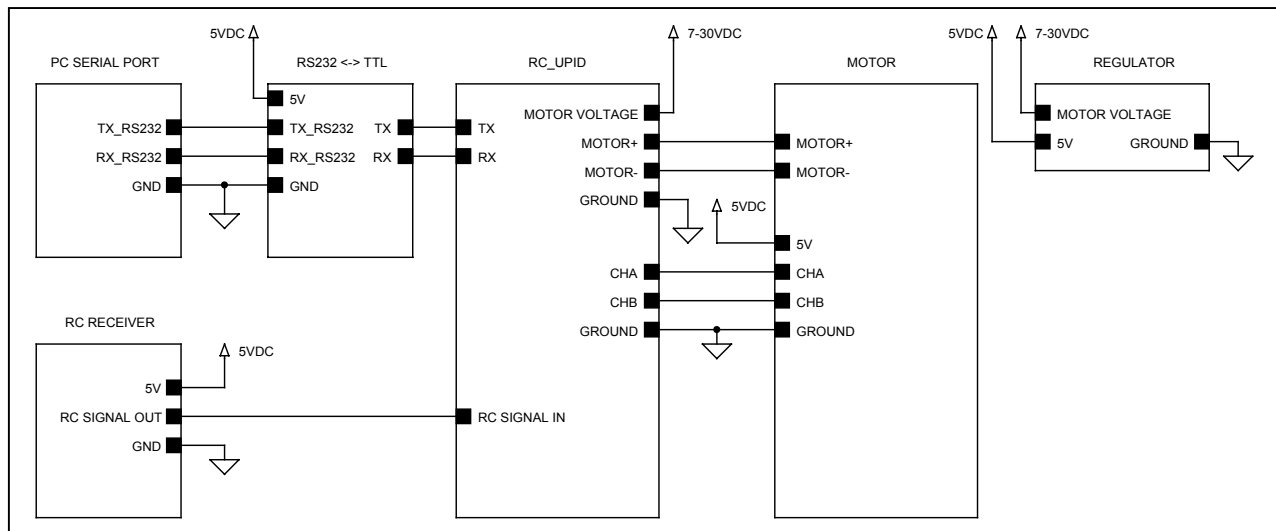
Some RC receivers will not output a solid +5V signal during the high portion of the input pulse. Various discrete logic ICs may be used to convert a low voltage levels to +5V signals. See figure 16 for one example.

Once the RC\_UPID is programmed, and the PID constants are set, a serial interface is unnecessary. But if you can retain one in your design it might be beneficial for troubleshooting or testing at a later date.

A separate 5V output linear regulator should be used to power the motor's quadrature encoder and the RC receiver. This external regulator should be capable of sourcing 250mA. But check your motor and receiver specifications to determine the exact current requirements for your system. The RC\_UPID will operate off of the motor voltage supply.

All power supply grounds should be common in your system.

**Figure 15: System Block Diagram**

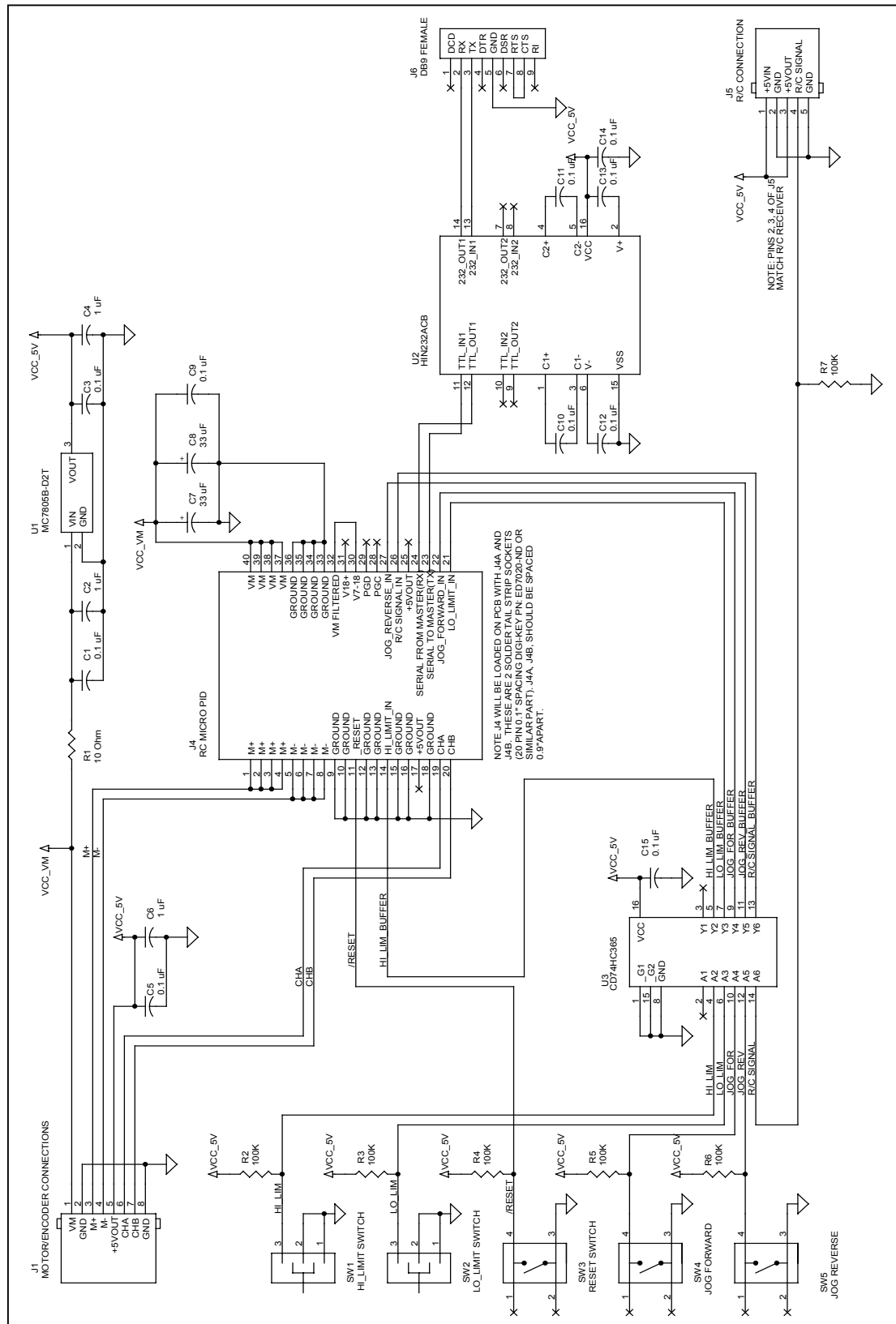


### 4.2 Example Schematic

The following schematic may be used as a reference for designs using the RC\_UPID. The schematic shows all inputs being buffered and all switches pulled up to +5V. It should be noted that not all signals require buffering or pull-ups, but the buffer will provide protection from electrostatic discharge as well as ensure that input signals are at the correct voltage.

A serial interface for programming and testing the RC\_UPID has been included in the example schematic.

Figure 16: Example Schematic



## 5. Communication Protocol

### 5.1 Overview

The RC\_UPID makes use of a serial communication protocol for programming and some of the interface methods. Generally the user will have to utilize the serial interface at least once before placing the RC\_UPID into operation. For instance, the device PID settings would need to be evaluated and modified via the serial interface before the RC\_UPID can be operated as a stand-alone RC signal velocity controller. Accessing the various registers within the RC\_UPID can be simplified by making use of the custom software available at [www.solutions-cubed.com](http://www.solutions-cubed.com).

#### 5.1.1 Format

The serial communication protocol used by the RC\_UPID is TTL level, 8N1, 9.6KBPS, true data. Connecting to a PC serial port will require an RS232 converter (see figure 14). The checksum is a simple modulo 256 summation of all bytes in the message previous to the checksum (the sum of hexadecimal values 0xE5, 0x01, 0x80 is 0x166; the checksum would only use the 0x66 portion of this sum).

Message components greater than 1 byte in length are sent and received least significant byte first. For example, if sending 10,000 as two bytes they would be sent 0x10 first, then 0x27 (both in hexadecimal (10,000 decimal = 0x2710 hexadecimal)).

#### 5.1.2 EEPROM Issues

**The commands below that are marked in bold write to EEPROM when implemented.** It is best to implement these commands while the motor is stopped. Any time that EEPROM is written to the time-base used for velocity control is extended and glitches in velocity control can occur. In addition EEPROM may fail after 100,000 write operations. Any custom control systems should take this into account when implementing commands that write to EEPROM. If the Save Position (USER0 bit 2) function is enabled then the EEPROM will be written to once every minute as long as the motor is not moving. If your system is powered for prolonged periods of time without movement then you should determine the expected life of the EEPROM before enabling the Save Position mode.

#### 5.1.3 Timing Issues

Certain timing issues must be taken into account when implementing this serial data interface. The first critical timing issue is the end of string timer. When the RC\_UPID receives a byte of valid data, it will begin to store each subsequent byte of data that is received within 3 PID update periods (2.5ms at the default setting) of the previous byte. Once the end of string timer has elapsed with no new data arriving, then the data received will be checked for correct format (valid command, correct length, valid checksum). If the data is accepted then a response will be sent and the command will be executed, otherwise the RC\_UPID will take no action. In summary, each byte of data must be sent with less than 2.5ms between bytes, and the response to a valid command will require at least 2.5ms (assuming the default PID update period of 1200Hz is used).

For commands that write to EEPROM, additional time is required for the write operation to take place. The RC\_UPID will return a response after EEPROM has been written to. This may take as long as 40ms.

## 5.2 Command Set

Figure 17: Velocity Controller Command Set

Command	Description
<b>Write RC Values</b>	Writes RC settings to EEPROM, loads registers with new values for immediate use.
Read RC Values	Reads and returns the RC settings from EEPROM.
<b>Write Stop Limit Buffer</b>	Writes the stop limit buffer value to EEPROM and loads registers with new values for immediate use.
Read Stop Limit Data	Reads and returns the low and high stop limit positions and the stop limit buffer values from EEPROM.
<b>Write PID Values</b>	Writes PID constant settings to EEPROM and loads registers for immediate use. Also clear the current sum of integral errors.
Read PID Values	Reads and returns the PID constant settings from EEPROM.
<b>Write Velocity Settings</b>	Writes the jog velocity and max velocity to EEPROM and loads registers with new values for immediate use.
Read Velocity Settings	Reads and returns the current jog velocity and max velocity settings from EEPROM.
<b>Write User Registers</b>	Writes the USER0 and USER1 flag registers to EEPROM.
Read User Registers	Returns the current contents of the USER0, USER1, and STATUS flag registers
<b>Write Address</b>	Write to EEPROM a new address value
Read Position	Reads current position, desired velocity, actual velocity, and raw RC pulse width measurement generated RC_UPID.
Clear Fault Flag	Clears fault condition flag caused by under-voltage, over-temperature, or over-current conditions.
Write RC Test Value	Write an RC test value that can be used instead of an actual RC signal for speed control. The USER0 bit 2 must be set for this value to effect velocity control.
Read RC Test – RC Mid Count Values	Reads the current RC test value and the current RC Mid Point.
Write RC Test Value Return Actual Position Value	Write an RC test value that can be used instead of an actual RC signal for speed control. The USER0 bit 2 must be set for this value to effect velocity control. In addition, this command returns the current motor position value instead of an acknowledge.
Read Firmware Revision	Reads current firmware revision, can be compared to errata sheet document to track known firmware errors

## 5.2.1 Write RC Values

<b>Description:</b> Sets parameters of valid RC signals.			
Component of Command	Range of Values	Number of Bytes	Decimal(Hex) Example
Command Byte	224	1	224(0xE0)
Address	0-255	1	1(0x01)
RC Low Limit Count	0 to 32,767	2	615(0x0267)
RC High Limit Count	0 to 32,767	2	3073(0x0C01)
RC Minimum Count	0 to 32,767	2	1229(0x04CD)
RC Maximum Count	0 to 32,767	2	2458(0x099A)
RC Dead Band Count	0 to 32,767	2	10(0x0A)
Checksum	0-255	1	213(0xD5)
Component of Response	Range of Values	Number of Bytes	Decimal(Hex) Example
ACK	0x06	1	

**Notes:** String would be sent as 0xE0, 0x01, 0x67, 0x02, 0x01, 0x0C, 0xCD, 0x04, 0x9A, 0x09, 0x0A, 0xD5. Commas are just shown for clarity and are not part of the data actually sent. Data is shown as hexadecimal bytes to clarify position of least significant bytes in string.

**5.2.2 Read RC Values**

<b>Description:</b> Reads RC values from EEPROM in RC_UPID.			
<b>Component of Command</b>	<b>Range of Values</b>	<b>Number of Bytes</b>	<b>Decimal(Hex) Example</b>
Command Byte	225	1	225(0xE1)
Address	0-255	1	1(0x01)
Checksum	0-255	1	226(0xE2)
<b>Component of Response</b>	<b>Range of Values</b>	<b>Number of Bytes</b>	<b>Decimal(Hex) Example</b>
Response Byte	160	1	160(0xA0)
Address	0-255	1	1(0x01)
Low Limit Count	0 to 32,767	2	615(0x0267)
High Limit Count	0 to 32,767	2	3073(0x0C01)
Minimum Count	0 to 32,767	2	1229(0x04CD)
Maximum Count	0 to 32,767	2	2458(0x099A)
Dead Band Count	0 to 32,767	2	10(0x0A)
Checksum	0-255	1	149(0x95)
<b>Notes:</b> String would be received as 0xA0, 0x01, 0x67, 0x02, 0x01, 0x0C, 0xCD, 0x04, 0x9A, 0x09, 0x0A, 0x95.			

**5.2.3 Write Stop Limit Buffer**

<b>Description:</b> Writes the 32-bit stop limit buffer to EEPROM. The stop limit buffer is the area around the high or low stop limits that enacts a maximum velocity reduction.			
<b>Component of Command</b>	<b>Range of Values</b>	<b>Number of Bytes</b>	<b>Decimal(Hex) Example</b>
Command Byte	226	1	226(0xE2)
Address	0-255	1	1(0x01)
Limit Position Buffer	1 to +2,147,483,647	4	180,000(0x0002BF20)
Checksum	0-255	1	196(0xC4)
<b>Component of Response</b>	<b>Range of Values</b>	<b>Number of Bytes</b>	<b>Decimal(Hex) Example</b>
ACK	6	1	6 (0x06)
<b>Notes:</b> String would be sent as 0xE2, 0x01, 0x20, 0xBF, 0x02, 0x00, 0xC4.			

**5.2.4 Read Stop Limit Data**

<b>Description:</b> Read stop limit positions and current stop limit buffer value.			
<b>Component of Command</b>	<b>Range of Values</b>	<b>Number of Bytes</b>	<b>Decimal(Hex) Example</b>
Command Byte	227	1	227(0xE3)
Address	0-255	1	1(0x01)
Checksum	0-255	1	228(0xE4)
<b>Component of Response</b>	<b>Range of Values</b>	<b>Number of Bytes</b>	<b>Decimal(Hex) Example</b>
Response Byte	160	1	160(0xA0)
Address	0-255	1	1(0x01)
High Limit Position	-2,147,483,648 to +2,147,483,647	4	500,000(0x0007A120)
Low Limit Position	-2,147,483,648 to +2,147,483,647	4	-500,000(0xFFF85EE0)
Limit Position Buffer	1 to +2,147,483,647	4	180,000(0x0002BF20)
Checksum	0-255	1	127(0x7F)
<b>Notes:</b> String would be received as 0xA0, 0x01, 0x20, 0xA1, 0x07, 0x00, 0xE0, 0x5E, 0xF8, 0xFF, 0x20, 0xBF, 0x02, 0x00, 0x7F.			

**5.2.5 Write PID Values**

<b>Description:</b> Writes PID filter settings to EEPROM.			
<b>Component of Command</b>	<b>Range of Values</b>	<b>Number of Bytes</b>	<b>Decimal(Hex) Example</b>
Command	229	1	229(0xE5)
Address	0 to 255	1	1(0x01)
P Constant	0 to 32,767	2	8000(0x1F40)
I Constant	0 to 32,767	2	35(0x0023)
D Constant	0 to 65,535	2	0(0x0000)
PID Period	0 to 255 (select values only)	1	124(0x7C)
PID Scalar	0 to 31	1	8(0x08)
Checksum	0 to 255	1	236(0xEC)
<b>Component of Response</b>	<b>Range of Values</b>	<b>Number of Bytes</b>	<b>Decimal(Hex) Example</b>
ACK	6	1	6 (0x06)
<b>Notes:</b>			

**5.2.6 Read PID Values**

<b>Description:</b> Read PID filter settings from EEPROM			
<b>Component of Command</b>	<b>Range of Values</b>	<b>Number of Bytes</b>	<b>Decimal(Hex) Example</b>
Command	230	1	230 (0xE6)
Address	0 to 255	1	1 (0x01)
Checksum	0 to 255	1	231 (0xE7)
<b>Component of Response</b>	<b>Range of Values</b>	<b>Number of Bytes</b>	<b>Decimal(Hex) Example</b>
Response Byte	160	1	160(0xA0)
Address	0 to 255	1	1(0x01)
P Constant	0 to 32,767	2	8000(0x1F40)
I Constant	0 to 32,767	2	35(0x0023)
D Constant	0 to 65,535	2	0(0x0000)
PID Period	0 to 255	1	124(0x7C)
PID Scalar	0 to 31	1	8(0x08)
Checksum	0 to 255	1	167(0xA7)
<b>Notes:</b>			

**5.2.7 Write Velocity Settings**

<b>Description:</b> Writes user velocity control parameters to EEPROM.			
<b>Component of Command</b>	<b>Range of Values</b>	<b>Number of Bytes</b>	<b>Decimal(Hex) Example</b>
Command	233	1	233 (0xE9)
Address	0 to 255	1	1 (0x01)
Jog Velocity	1 to 32,767	2	120(0x0078)
Max Velocity	1 to 32,767	2	320(0x0140)
Checksum	0 to 255	1	163(0xA3)
<b>Component of Response</b>	<b>Range of Values</b>	<b>Number of Bytes</b>	<b>Decimal(Hex) Example</b>
ACK	6	1	6 (0x06)
<b>Notes:</b>			

**5.2.8 Read Velocity Settings**

<b>Description:</b> Reads velocity parameters from EEPROM.			
<b>Component of Command</b>	<b>Range of Values</b>	<b>Number of Bytes</b>	<b>Decimal(Hex) Example</b>
Command	234	1	234 (0xEA)
Address	0 to 255	1	1 (0x01)
Checksum	0 to 255	1	235 (0xEB)
<b>Component of Response</b>	<b>Range of Values</b>	<b>Number of Bytes</b>	<b>Decimal(Hex) Example</b>
Response Byte	160	1	160 (0xA0)
Address	0 to 255	1	1 (0x01)
Jog Velocity	1 to 32,767	2	120(0x0078)
Max Velocity	1 to 32,767	2	320(0x0140)
Checksum	0 to 255	1	90(0x5A)

**Notes:****5.2.9 Write User Registers**

<b>Description:</b> Write User Registers is used to define the operating mode of the controller. Various other functions such as software reset, and restoration of default values are controlled via the USER0 and USER1 registers.			
<b>Component of Command</b>	<b>Range of Values</b>	<b>Number of Bytes</b>	<b>Decimal(Hex) Example</b>
Command	235	1	235 (0xEB)
Address	0 to 255	1	1 (0x01)
USER0	0 to 255	1	34 (0x22)
USER1	0 to 255	1	49 (0x31)
Checksum	0 to 255	1	63 (0x3F)
<b>Component of Response</b>	<b>Range of Values</b>	<b>Number of Bytes</b>	<b>Decimal(Hex) Example</b>
ACK	6	1	6 (0x06)

**Notes:** See section 3.13 for USER0 and USER1 bit settings.**5.2.10 Read User Registers**

<b>Description:</b> Reads USER0, USER1, and STATUS registers from RC_UPID.			
<b>Component of Command</b>	<b>Range of Values</b>	<b>Number of Bytes</b>	<b>Decimal(Hex) Example</b>
Command	236	1	236 (0xEC)
Address	0 to 255	1	1 (0x01)
Checksum	0 to 255	1	237 (0xED)
<b>Component of Response</b>	<b>Range of Values</b>	<b>Number of Bytes</b>	<b>Decimal(Hex) Example</b>
Response Byte	160	1	160 (0xA0)
Address	0 to 255	1	1 (0x01)
USER0	0 to 255	1	34 (0x22)
USER1	0 to 255	1	49 (0x31)
STATUS	0 to 255	1	2 (0x02)
Checksum	0 to 255	1	246 (0xF6)

**Notes:** See section 3.13 for USER0 and USER1 bit settings. See section 5.3 for STATUS flag definitions.

**5.2.11 Write Address**

**Description:** The Write Address command stores the new address value EEPROM and replaces the existing device address with the new value. All future serial communication must use the new address value to be considered valid.

Component of Command	Range of Values	Number of Bytes	Decimal(Hex) Example
Command	237	1	237 (0xED)
Address	0 to 255	1	1 (0x01)
New Address	0 to 255	1	2 (0x02)
Checksum	0 to 255	1	240 (0xF0)
Component of Response	Range of Values	Number of Bytes	Decimal(Hex) Example
ACK	6	1	6 (0x06)

**Notes:**

**5.2.12 Read Position**

**Description:** The Read Position command returns data that can be used for data-logging purposes. The data returned includes the current position, desired velocity (as calculated by the RC\_UPID) the actual motor velocity, and the raw RC pulse width as measured by the RC\_UPID. The position and velocity values may be negative and are returned as two's complement numbers.

Component of Command	Range of Values	Number of Bytes	Decimal(Hex) Example
Command	239	1	239 (0xEF)
Address	0 to 255	1	1 (0x01)
Checksum	0 to 255	1	240 (0xF0)
Component of Response	Range of Values	Number of Bytes	Decimal(Hex) Example
Response Byte	160	1	160 (0xA0)
Address	0 to 255	1	1 (0x01)
Actual Position	-2,147,483,648 to +2,147,483,647	4	100,000(0x000186A0)
Raw RC Count	0 to 65535	2	1700(0x06A4)
Desired Velocity	-32,768 to +32,767	2	-65(0xFFBF)
Motor Velocity	-32,768 to +32,767	2	-63(0xFFC1)
Checksum	0 to 255	1	240(0xF0)

**Notes:**

**5.2.13 Clear Fault Flag**

**Description:** This command will clear the STATUS register Fault flag caused by persistent under-voltage, over-current, or over-temperature conditions. Care should be taken not to clear the fault flag if a fault condition still exists.

Component of Command	Range of Values	Number of Bytes	Decimal(Hex) Example
Command	241	1	241 (0xF1)
Address	0 to 255	1	1 (0x01)
Checksum	0 to 255	1	242 (0xF2)
Component of Response	Range of Values	Number of Bytes	Decimal(Hex) Example
ACK	6	1	6 (0x06)

**Notes:** The USER0 Enable H-bridge on Fault setting (enabled by setting bit 0 of the USER0 register with the Write User Registers command) will automatically clear the fault condition after 25 PID periods.

**5.2.14 Write RC Test Value**

**Description:** Load the RC\_UPID with a test value that is used instead of an actual RC signal when the Use Test Value function is enabled (USER0 bit 2 is set).

Component of Command	Range of Values	Number of Bytes	Decimal(Hex) Example
Command	242	1	242 (0xF2)
Address	0 to 255	1	1 (0x01)
RC Test Value	0 to 32,767	2	1700(0x06A4)
Checksum	0 to 255	1	157(0x9D)
Component of Response	Range of Values	Number of Bytes	Decimal(Hex) Example
ACK	6	1	6 (0x06)

**Notes:**

**5.2.15 Read Velocity Settings**

**Description:** Reads the current RC Test Value, and the RC Mid Value. Normally the RC Mid Value is fixed at 1843 counts in duration (1.5ms), but if the Auto Calibrate Mid Point function is enabled (USER1 bit 1 is set) then the RC Mid Value will be equal to the 21<sup>st</sup> consecutive valid signal after power up.

Component of Command	Range of Values	Number of Bytes	Decimal(Hex) Example
Command	243	1	243 (0xF3)
Address	0 to 255	1	1 (0x01)
Checksum	0 to 255	1	244 (0xF4)
Component of Response	Range of Values	Number of Bytes	Decimal(Hex) Example
Response Byte	160	1	160 (0xA0)
Address	0 to 255	1	1 (0x01)
RC Test Value	0 to 32,767	2	1700(0x06A4)
RC Mid Value	0 to 32,767	2	1800(0x0708)
Checksum	0 to 255	1	90(0x5A)

**Notes:**

**5.2.16 Write RC Test Value – Return Position Value**

**Description:** This command writes the RC test value as does the Write RC Test Value command, but returns the position value currently loaded into the RC\_UPID

Component of Command	Range of Values	Number of Bytes	Decimal(Hex) Example
Command	242	1	244 (0xF4)
Address	0 to 255	1	1 (0x01)
RC Test Value	0 to 32,767	2	1700(0x06A4)
Checksum	0 to 255	1	159(0x9F)
Component of Response	Range of Values	Number of Bytes	Decimal(Hex) Example
Response Byte	160	1	160 (0xA0)
Address	0 to 255	1	1 (0x01)
Actual Position	-2,147,483,648 to +2,147,483,647	4	100,000(0x000186A0)
Checksum	0 to 255	1	200 (0xC8)

**Notes:**

**5.2.17 Read Firmware Revision**

**Description:** The current firmware revision can be read with this command and may be used to reference errata documents at [www.solutions-cubed.com](http://www.solutions-cubed.com) to determine if an error exists in your firmware. Errors may or may not exist, and if they do exist may not affect your design.

<b>Component of Command</b>	<b>Range of Values</b>	<b>Number of Bytes</b>	<b>Decimal(Hex) Example</b>
Command	255	1	255 (0xFF)
Address	0 to 255	1	1 (0x01)
Checksum	0 to 255	1	0 (0x00)
<b>Component of Response</b>	<b>Range of Values</b>	<b>Number of Bytes</b>	<b>Decimal(Hex) Example</b>
Response Byte	160	1	160 (0xA0)
Address	0 to 255	1	1 (0x01)
Firmware Revision	0 to 255	1	2 (0x02)
Checksum	0 to 255	1	163 (0xA3)
<b>Notes:</b>			

**5.3 USER0, USER1, and STATUS Flags**

Configuring the RC\_UPID for its various modes of operation is accomplished by writing various flag bits to the USER0 and USER1 registers. Additional information regarding the operation status of the RC\_UPID can be gleaned from the STATUS register. These three registers and their bit assignments are defined here.

**Figure 18: USER0 Bits**

Bit	Flag	Description	R/W
0	RetryFault	When set to "1" the controller will attempt to re-enable the H-bridge 25 PID update periods after a fault condition occurs. This will continue indefinitely.	R/W
1	DisableSaturation	When set to "1" and much of the limiting effects of a saturation condition are reduced. Enabling this function can allow for some integral "wind up" in conditions where the motor is not able to reach commanded velocities.	R/W
2	TestSignal	When set to "1" and the RC Test Value is used to determine commanded velocity instead of an RC signal at the RC_SIGNAL_IN pin.	R/W
3	RestoreDefaults	When set to a "1" the factory default settings for all user accessible registers will be reset including the PID settings, velocity registers, and the user registers. This flag is self-clearing.	R/W
4	SoftwareReset	When set to a "1" a software reset occurs. This flag is self-clearing.	R/W
5	FourXMode	When set to a "1" the quadrature signal from the motor is decoded into 4 pulses for every set of input pulses on the channel A and channel B encoder lines (4X mode). When cleared ("0") each set of pulses from the quadrature signal is decoded into a single output pulse. See section 3.13 for more on 4X_1X mode of operation.	R/W
6	BadSignalStop	When set to "1" the motor will be stopped whenever a bad or lost RC signal occurs. A lost signal occurs whenever a positive RC pulse is not received within 53ms. A bad signal occurs if a received signal is shorter than RC Low or longer than RC High. If this bit is left clear then the last valid signal measurement will be used in the event of lost or bad signals.	R/W
7	SavePosition	When set to a "1" the motor position will be saved to EEPROM if the commanded velocity remains 0 for 65,000 consecutive PID update periods. With the default settings this is roughly 1 minute. This mode should not be enabled for systems that are left powered and not moving for prolonged periods of time. The EEPROM has a lifespan of 1,000,000 write cycles.	R/W

Figure 19: USER1 Bits

Bit	Flag	Description	R/W
0	DoubleCount	When set to a "1" The RC_UPID counts velocity for 2 PID update periods before applying the PID filter to the error signal. This can allow for better control at lower speeds.	R/W
1	AutoCalibrate	When set to a "1" the RC Mid Point setting will be replaced by the 21 <sup>st</sup> consecutive valid signal when the RC_UPID is powered up. Use this function to calibrate the RC_UPID to your RC radios default joystick position.	R/W
2	Unused		
3	Unused		
4	Unused		
5	Unused		
6	Unused		
7	Unused		

Figure 20: STATUS Bits

Bit	Flag	Description	R/W
0	Fault	This bit is set to a "1" when an under-voltage, over-current, or over-temperature fault occurs.	R
1	Velocity	This bit is set to a "1" when the actual velocity is equal to the desired velocity.	R
2	Low Limit	This bit is set to a "1" if the LO_LIMIT_IN pin (21) is asserted.	R
3	High Limit	This bit is set to a "1" if the HI_LIMIT_IN pin (14) is asserted.	R
4	Jog Forward	This bit is set to a "1" if the JOG_FOR_IN pin (22) is asserted.	R
5	Jog Reverse	This bit is set to a "1" if the JOG_REV_IN pin (27) is asserted.	R
6	Bad Signal	This bit is set to a "1" if the signal at the RC_SIGNAL_IN pin (26) is shorter than the duration specified by the RC Low setting or if the signal is longer than the RC High setting.	R
7	Lost Signal	This bit is set to a "1" if no signal has been received within 53ms.	R

### 5.2.1 Write User Registers Example

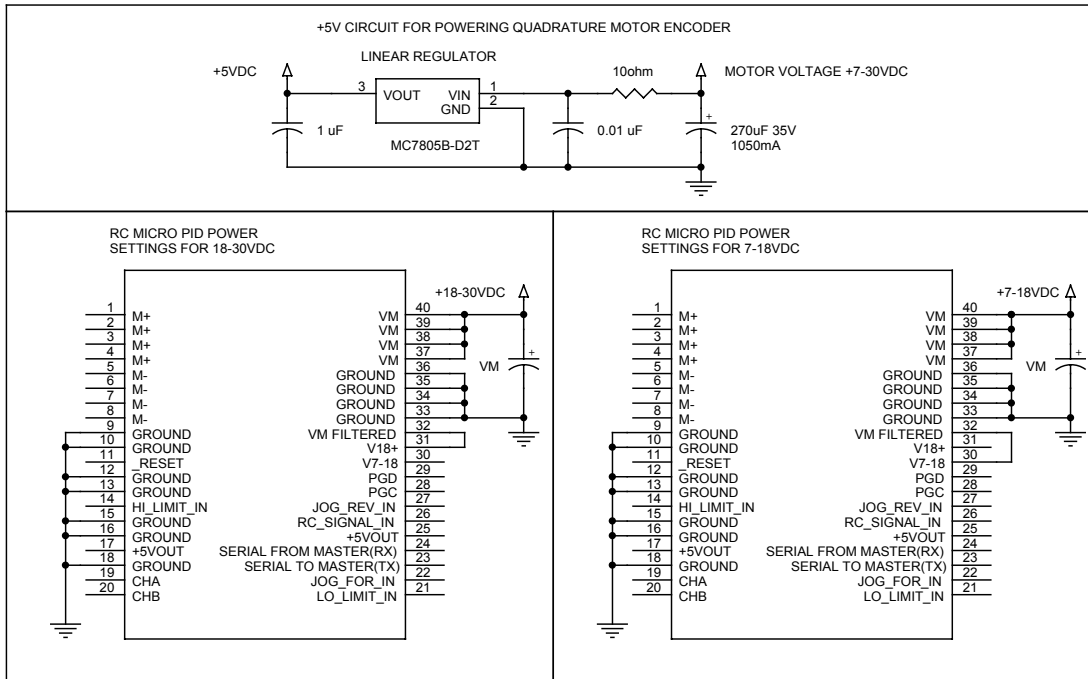
To operate the RC\_UPID with automatic retry on a fault, 4X mode, and with the position save mode enabled, the USER0 register would equal 0xA1 (binary '10100001' ←LSB) and the USER1 register would equal 0x00 (binary '00000000' ←LSB). The total message string sent to a RC\_UPID would look like this...

0xEB, 0x01, 0xA1, 0x00, 0x8D

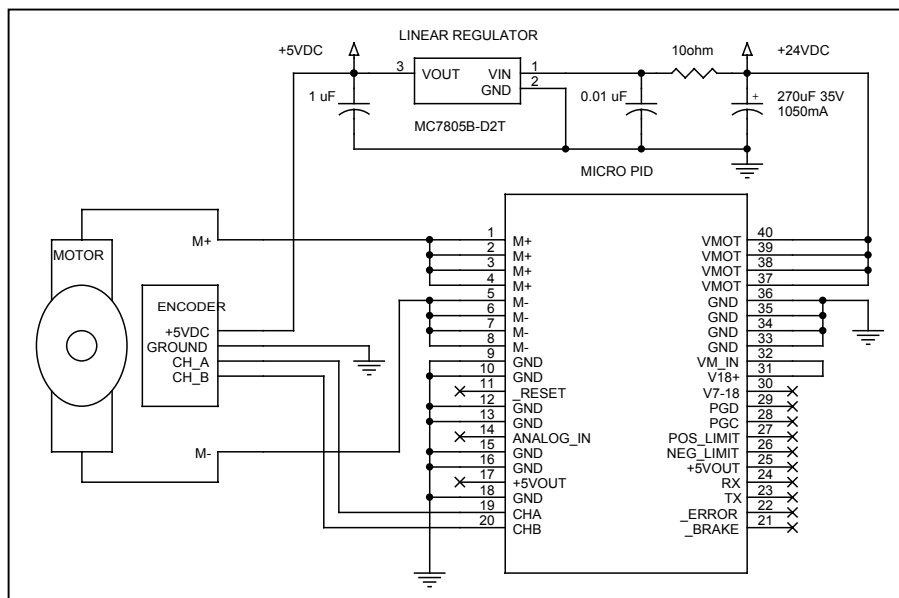
## 6. Quick Start

### 6.1.1 Select the Correct Power Setting

Connect your power supply and pins associated with powering the RC\_UPID as per the diagram below. The 5VOUT pins can provide a small amount of current to external loads. In general the RC MICRO PID 5V supply cannot provide enough current to power a quadrature encoder. An external 5V supply is required to provide power to the quadrature encoder. The diagrams below can be used for reference. The 270uF capacitor below is the VM capacitor.



### 6.1.2 Connect your Motor and Encoder to the RC\_UPID

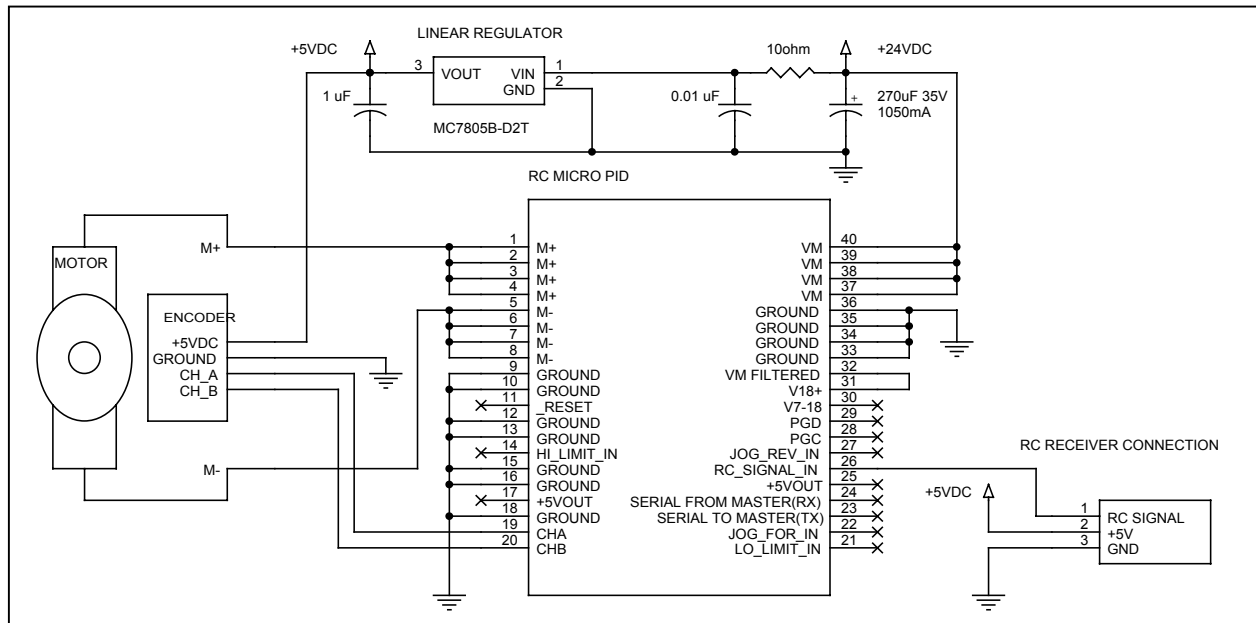




**6.1.5 Connect the RC Radio System**

Turn off power to the system, connect the RC receiver, and then apply power to your system. If your RC receiver outputs a +5V signal then you should be able to implement a direct connection to the RC\_UPID. If not, you may need to provide an op-amp, transistor, or logic buffer circuit that ensures that the RC\_UPID receives a TTL compatible signal. Grounds should be common between the motor power supply and RC receiver. Do not use the 5V outputs available on the RC\_UPID to power the RC receiver or the motor's encoder. This 5V output can only supply a modest amount of current, and is not intended to power additional circuitry.

While not shown here it would be useful to maintain a serial interface for testing and modifying the PID constants when your load is attached. However, once the RC\_UPID has been configured for operation the circuit below is required for motor control with an RC receiver.



**Disclaimer of Liability and Accuracy:** Information provided by Solutions Cubed is believed to be accurate and reliable. However, Solutions Cubed assumes no responsibility for inaccuracies or omissions. Solutions Cubed assumes no responsibility for the use of this information and all use of such information shall be entirely at the user's own risk.

**Life Support Policy:** Solutions Cubed does not authorize any Solutions Cubed product for use in life support devices and/or systems without express written approval from Solutions Cubed.

**Warranty:** Solutions Cubed warrants all Motor Control Modules against defects in materials and workmanship for a period of 90 days. If you discover a defect, we will, at our option, repair or replace your product or refund your purchase price. This warranty does not cover products that have been physically abused or misused in any way.