Dual Motor Quad Servo Controller – BM011

OPEN SOURCE HARDWARE MODULE

hardware made easy

Solutions Cubed

designservices@solutions-cubed.com
phone – 530.891.8045
256 East First Street
Chico, CA 95928

Contact Solutions Cubed, LLC for your custom designs:

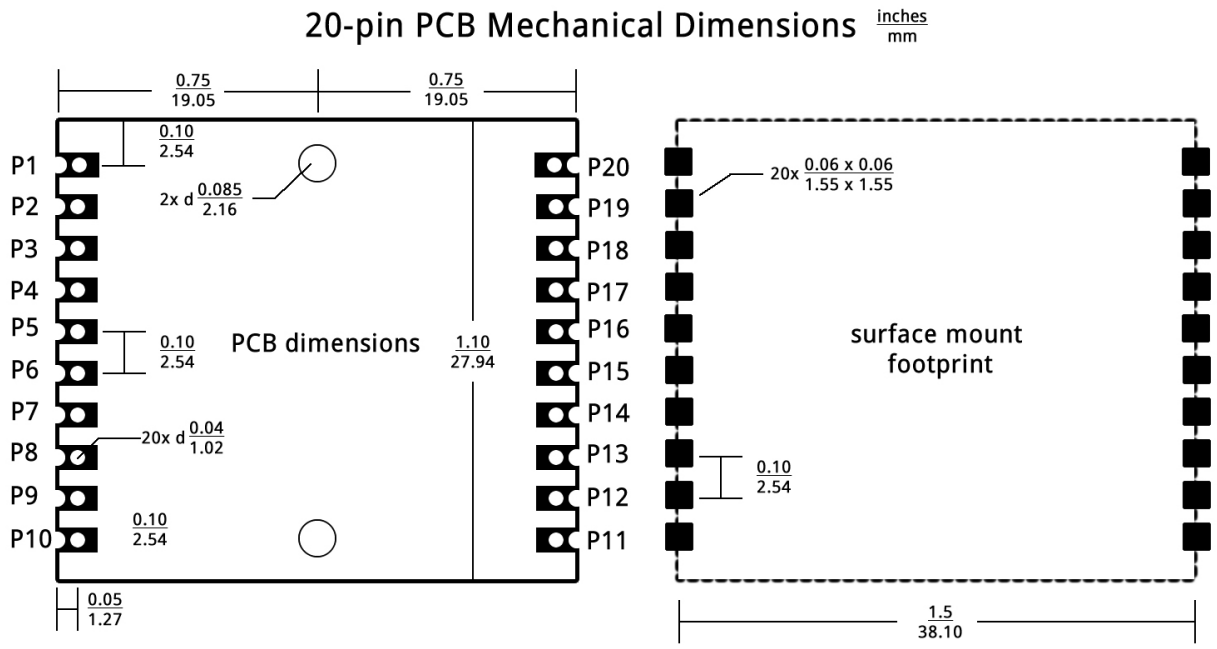
Solutions Cubed is an innovative electronic design firm. We have created successful designs for a myriad of industries including mass produced consumer products, deep-sea robotic components, and encrypted encoders for the banking industry. We love meeting new customers and are interested in hearing about your design needs.

Product Description:

This module can be used to control DC motors, generate pulses to control servos, and read servo pulses from radio receivers. A logic-level serial communication interface is the primary means of controlling the module.

- Simple serial data interface at 9600BPS for control
- 3.3V or 5.0V operation
- Control speed and direction of two DC motors, 6-24V, 1-2A continuous, 5A peak
- Over temperature, over current, under voltage protections
- Generate four 0.5-2.5mS pulses in 10uS increments
- Read four 0.5-2.5mS pulses in 10uS increments
- Special modes directly convert input pulses to motor speed and direction control
- Open-source firmware and hardware so you can modify the design as you like

Dimensions:



Specifications:

Characteristic	Min	Typ	Max	Unit	Notes
VDD Operating voltage	3		5.25	V	VDD pins
Operating current		7		mA	No motors/servos attached
VMOT Motor Voltage	6		24	V	Operating near the maximum voltage may require external protection from voltage spikes
Continuous Motor Current		1		A	At 90% duty cycle, 12V
Peak Motor Current		5	8	A	
Motor drive PWM frequency			7.8	KHz	
Servo pulse output/input resolution		10		uS	
Servo pulse output range	0		2550	uS	
Servo pulse output accuracy		+/-20		uS	
Servo pulse input measurement range	450		2550	uS	register is loaded with a 0 to indicate the pulse width is outside of this range
Servo pulse input accuracy		+/-20		uS	
Servo pulse output period	12	24	510	mS	this is user adjustable but defaults to 24mS
Operating temperature	-40		+85	°C	

Pin Functions and Notes

#	Name	Maximum Voltage	Notes
1	DOUT	VDD	has series 270Ω resistor on board
2	DIN	VDD	has series 270Ω resistor on board
3	S_OUT1	N/A	has series 270Ω resistor on board
4	S_OUT2	N/A	has series 270Ω resistor on board
5	S_OUT3	N/A	has series 270Ω resistor on board
6	S_OUT4	N/A	has series 270Ω resistor on board
7	S_IN1	VDD	has series 270Ω resistor on board
8	S_IN2	VDD	has series 270Ω resistor on board
9	S_IN3	VDD	has series 270Ω resistor on board
10	S_IN4	VDD	has series 270Ω resistor on board
11	VDD	5.5V	Microcontroller and servo power supply, provide external filtering if servos create substantial noise.
12	VDD	5.5V	Microcontroller and servo power supply, provide external filtering if servos create substantial noise.
13	GND	0V	Ground return for motor and logic supplies
14	GND	0V	Ground return for motor and logic supplies
15	M2-	VMOT	motor control, connects to negative motor lead
16	M2+	VMOT	motor control, connects to positive motor lead
17	M1-	VMOT	motor control, connects to negative motor lead
18	M1+	VMOT	motor control, connects to positive motor lead
19	VMOT	24V	Operating near the maximum voltage may require external protection from voltage spikes
20	VMOT	24V	Operating near the maximum voltage may require external protection from voltage spikes

Programming Header J1

#	Name	Maximum Voltage	Notes
1	MCLR	VDD	Reset line for in-circuit programmer
2	5V	VDD	connects to VDD
3	GND	N/A	ground return
4	PGD	VDD	In-circuit programming data line
5	PGC	VDD	In-circuit programming clock line

User Notes/Tips

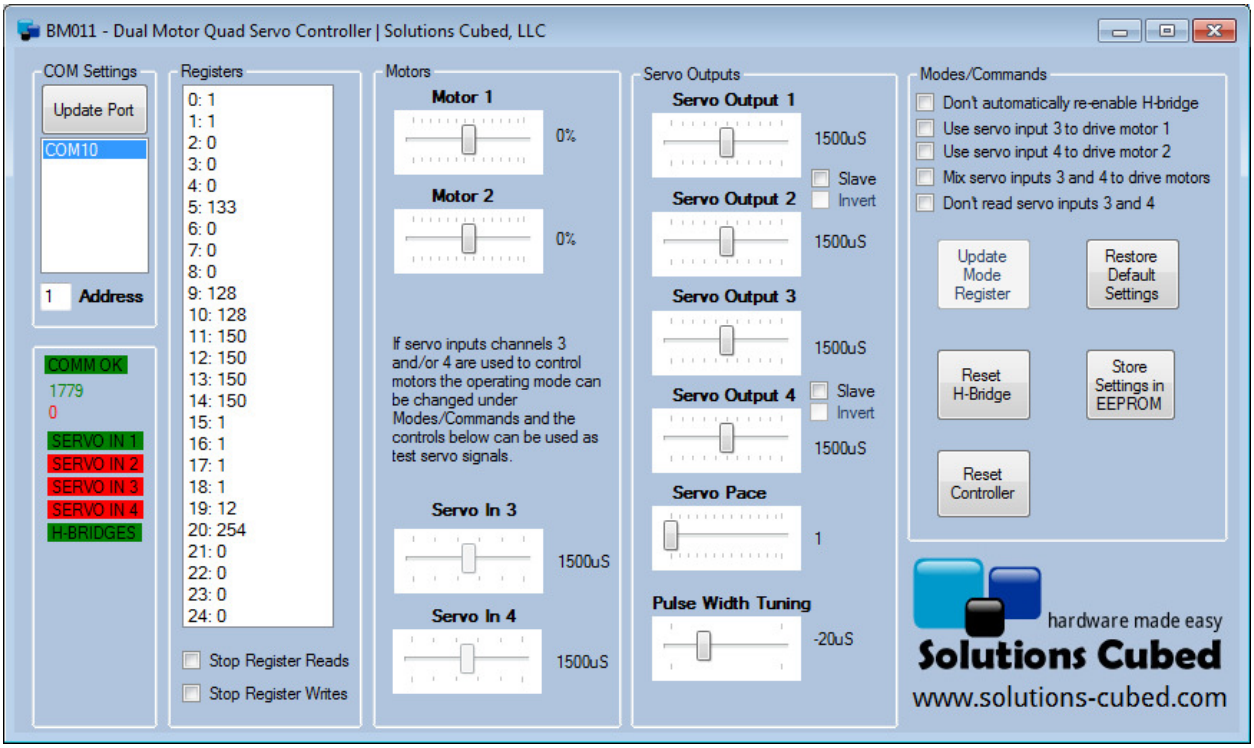
1. RC servos can draw large amounts of current. If you are driving heavy loads with your servos it is best to wire the servo power lines directly into your battery. See application schematics for examples.

2. The PIC16F1829 firmware is open-source and may be programmed using a free development environment and C compiler from Microchip. We used MPLAB X and XC8. The module may be debugged and programmed with the low cost devices such as Microchip's PICKit3. This will connect to J1 on the module by using our CS005 Programming Adapter kit available at www.solutions-cubed.com

3. Skid steering is accomplished with two DC motors forming an axel. Typically a 3rd roller wheel is attached for stability. Running one motor forward and the other reversed causes the chassis to move in one direction. A skid steering mode exists that allows you to use servo input channels S_IN3 and S_IN4 as inputs to a mixer algorithm. The output is the speed and direction control of the DC motors. This makes the creation of a skid steering system simple. This operating mode is enabled by setting a bit in the MODE register.

S_IN3 controls speed					
S_IN3	S_IN4	Motor 1	Motor 2	Duty Cycle Motor1	Duty Cycle Motor2
100	150	0	254	-100%	100%
110	150	25	229	-80%	80%
120	150	51	203	-60%	60%
130	150	76	178	-40%	40%
140	150	102	152	-20%	20%
150	150	127	127	0%	0%
160	150	152	102	20%	-20%
170	150	178	76	40%	-40%
180	150	203	51	60%	-60%
190	150	229	25	80%	-80%
200	150	254	0	100%	-100%
S_IN4 controls turning					
S_IN3	S_IN4	Motor 1	Motor 2	Duty Cycle Motor1	Duty Cycle Motor2
150	100	254	254	100%	100%
150	110	229	229	80%	80%
150	120	203	203	60%	60%
150	130	178	178	40%	40%
150	140	152	152	20%	20%
150	150	127	127	0%	0%
150	160	102	102	-20%	-20%
150	170	76	76	-40%	-40%
150	180	51	51	-60%	-60%
150	190	25	25	-80%	-80%
150	200	0	0	-100%	-100%

4. Test Software is available to test the serial interface and various control settings of this module. The test software source code is available on our web site.



Communication Protocol

Hardware Usage:

The asynchronous serial communication takes place at 9600BPS using eight data bits, no parity, and one stop bit (9600, n, 8, 1). Data is sent LSB first. There is no hardware flow control. Serial data is received by the module on its DIN pin. It sends data on its DOUT pin.

Communication Sequence Overview:

The serial protocol is a master/slave interface. This module is the slave device and unless otherwise noted does not initiate communication.

There are four types of communication packets. They are Read Packets, Write Packet, ACK Packets, and Reply Packets. The master always initiates a communication sequence with a command and a slave always terminates the communications sequence with a reply or ACK.

Packet Byte	Representation
Operation	Read = 209, Write = 210
Address	default is 1
Length	3 to 30
Packet Data	Message specific
Checksum	sum of all characters in packet proceeding the checksum

Figure 5.2: Packet information

Operation: There are two operations: read or write. A read operation allows the master to specify which registers to read from the module. A write operation allows the master to write to specific registers.

Address: The default device address is 1. This value may be changed using a write operation.

Length: This byte tells the receiver how many bytes are *following* the length byte. This includes the checksum.

Packet Data: The specific data that is necessary for a complete message.

Checksum: A simple sum of *all* of the bytes in the packet before the checksum. For example, if the packet data is 209, 1, 3, 0, 1, the checksum would be 214.

Since the checksum is a single byte value only the remainder is sent for values greater than 255. If the sum of a packet's bytes is 257, the checksum would be 1 (257/256 = 1 with a remainder of 1). If the sum is 512 the checksum is 0 (512/256 = 2 with a remainder of 0).

In most cases when you are summing values in a byte sized variable any overflow is lost, and the remainder is all that is left. This is the checksum.

Read Operation
A read operation allows the master to read specific registers from the module.

Operation Byte	Device Address	Length	Start Index	Number of Registers	Checksum

Figure: Form of a read command

The "Start Index" is the register index value that the read should begin from. For example, if you want to start your read at register 4, then the Start Index is 4. The "Number of Registers" is number of registers to read, including the register identified by the Start Index. The example below shows a read command of register 4. A reply packet will be returned when the read operation is accepted.

Master Sends: 209, 1, 3, 4, 1, 218
Slave returns: reply packet

Write Operation
A write operation allows the master to write specific registers to the module.

Operation Byte	Device Address	Length	Start Index	Data1	Data N	Checksum

Figure: Form of a write command

The "Start Index" identifies the first register that will be written to. The "Data1" register is the data to write to the Start Index register. With the Write command it is possible to write to more than one register at a time. The data will be written to sequential register starting at the Start Index. In the figure above the "Data N" byte will be written to the "Start Index" +1 register. The example below shows a write of 150 to register 11 and 151 to register 12.

Master Sends: 210, 1, 4, 11, 150, 151, 15
Slave returns: ACK packet

You'll notice that the sum of the bytes in the packet is 527. Since the checksum is a byte sized value only the overflow is sent (527 = hexadecimal 0x20F, therefore the lower byte is sent, 0x0F, or decimal 15.

Reply Packet
The module sends Reply Packets to the Master in response to a Read Operation. Each Reply Packet will begin with the slave address (default to "1"). Next is a length byte, followed by the message data. The last byte in the packet is the checksum.

Device Address	Length	Data1	DataN	Checksum

Figure: Reply packet representation

The "Data" in the reply packet is the register data that the master requested using a read command.

ACK Packet
The module always sends an ACK packet to the Master in response to a Write Operation. Each ACK Packet is a single byte.

ACK
6

Figure: ACK packet representation

The module sends the ACK packet *after* the requested write is completed. If the master does not receive an ACK after an appropriate period of time, it may assume that the write operation did not work.

Error Detection:
Error detection is accomplished by inspection of the received data and making sure that the data was received in a timely and appropriate fashion. Inspection of the data packets will be performed by…

- verifying that all elements of the packet are present
- making sure that the message is the correct length
- verifying the checksum
- verifying that the message is supported by the Slave
- testing all values with limited range
- inter-character time

The inter-character time is the time between successive characters (bytes) in the same packet. The maximum time allowed is 2ms.

Serial Communication Examples:
Example 1 - Setting Motor1 (register 9) to full speed reverse (1) and Motor2 (register 10) to full speed forward (254).

Master Sends: 210, 1, 4, 9, 1, 254, 223
Slave returns: 6

Example 2 - Setting Servo_Out1, Servo_Out2 (registers 11, 12) to 1mS (100), and Servo_Out3, Servo_Out4 (registers 13, 14) to 1.5mS (150).

Master Sends: 210, 1, 6, 11, 100, 100, 150, 150, 216
Slave returns: 6

Example 3 – Reading Servo_In1 (register 5) through Servo_In4 whose contents are 1mS, 2mS, 1.25mS and 1.75mS respectively.

Master Sends: 209, 1, 3, 5, 4, 222
Slave returns: 1, 5, 100, 200, 125, 175, 94

Register Definitions:

This module is controlled by reading from and writing to internal registers via the previously described serial interface. The registers are defined below.

Index	Register	Range	Default Value	Description
0	Firmware	1-255	NA	Firmware revision loaded into the module.
1	Address	0-255	1	The serial address this unit will respond to.
2	Command	0-255	0	Commands specific to the module may be implemented by writing to this register. See the detailed register definition.
3	Mode	0-255	0	Operating modes specific to the module may be implemented by writing to this register. See the detailed register definition.
4	Indicator	0-255	0	Indicators specific to this module may be read from this register. See the detailed register definition.
5	Servo_In1	45-255*	NA	The length of the pulse read at the associated pin in 10uS increments. The module will read pulses from 450-255uS. * Pulses outside the defined range result in an error code of 0 being loaded into this register.
6	Servo_In2	45-255*	NA	see above
7	Servo_In3	45-255*	NA	see above
8	Servo_In4	45-255*	NA	see above
9	Motor1	1-254	127	Writing to this register controls motor speed and direction. 127 is stopped or 0% duty cycle. 1 is full speed reverse or -99%, full forward is 254 or 99%. Speed ramping can be achieved by incrementally increasing or decreasing the value you are sending.
10	Motor2	1-254	127	see above
11	Servo_Out1	50-250	150	Controls the output pulse width in 10uS intervals. Most servos will accept values from 0.5-2.5mS. The pulse outputs are generated in sequence, Servo_Out1 then Servo_Out2, etc. They are generated every 2mS* Servo_Period.
12	Servo_Out2	50-250	150	see above
13	Servo_Out3	50-250	150	see above
14	Servo_Out4	50-250	150	see above

Register Definitions (continued):

Index	Register	Range	Default Value	Description
15	Servo_ Pace1	1-255	10	You can slow down the rate of change of the associated Servo_Out1 signal by modifying this value. This value sets the allowed change in output per Servo_Period. The default of value 10 allows you to change output pulse width 100uS every servo period. So changing the Servo_Out1 register from 100 to 200 would allow the pulse width to change from 1mS to 2mS in 240mS (Servo_Out1 change / Servo_Pace1 x Servo_Period). Setting the Servo_Pace1 value to 1 would allow you to change the output pulse from 1mS to 2mS in 2.4S ((2mS-1mS)/0.01mS x 24mS).
16	Servo_ Pace2	1-255	10	see above
17	Servo_ Pace3	1-255	10	see above
18	Servo_ Pace4	1-255	10	see above
19	Servo_ Period	6-255	12	This register establishes the period of time between the start of each output pulse. It is in 2mS intervals. Setting Servo_Period to 10 has each servo output pulse occur every 20mS. Some servos motors behave poorly if this value is shorted or extended significantly. Most servos expect a signal between 20-25mS. However, you can extend the period somewhat if you want to slow down the movement of you servos beyond what the Servo_Pacex registers allow.
20	Servo_ Adjust	0-255	254	Used to fine tune servo output pulse width. 254 subtracts 20uS from the pulse width commanded which typically accounts for interrupt and other programmatic delays.
21	Unused1	0-255	0	
22	Unused2	0-255	0	
23	Unused3	0-255	0	
24	Programmed	0-255	0	This register is used to determine if EEPROM is blank (first time power up)

Detailed Register Descriptions:

Command Register (index = 2): Writing values to the Command register implements specific commands. Once these commands are executed the register value is returned to 0.

Command	Value	Description
Restore	1	Restores internal registers to their default settings and stores those values in EEPROM so they become the power on default values.
Store	2	Stores the current register values in EEPROM so they become the power on default values.
Reset Controller	3	Forces the module's microcontroller into a watchdog timer reset.
Reset H-Bridge	4	Briefly disables and re-enables the motor drive circuitry to clear any fault conditions. This command also clears the H-bride fault status flag (bit 0 of the Indicator register).

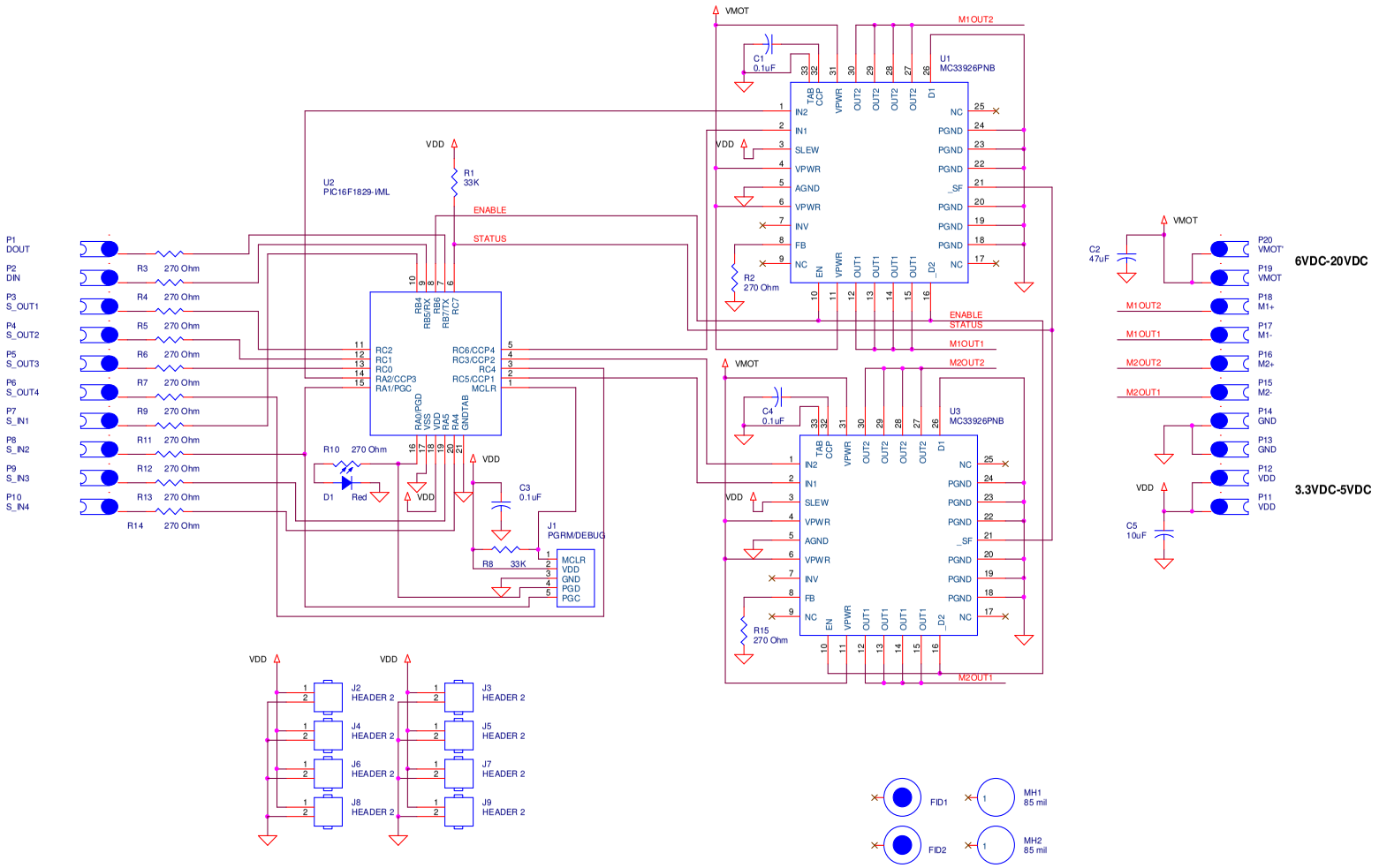
Mode Register (index = 3): Setting specific bits in the Mode register causes the module to implement special operating modes. A bit is set by writing a "1" to it. Multiple special modes can be implemented at the same time, although not all modes work together.

Mode	Bit	Description
Disable Auto Reset	0	Setting this bit prevents the controller from automatically re-enabling the H-bridges if they shut down due to a fault condition. With this bit set the H-bridges will have to be reset using the Reset H-Bridge command (see Command register).
Servo_In3 Controls Motor 1	1	Setting this bit causes the controller to use the servo input signal at Servo_In3 as the drive signal for Motor1. Servo signals from 1-2mS will be used with 1mS equal to full speed reverse, 1.5mS stopped, and 2mS = full speed forward.
Servo_In4 Controls Motor 2	2	as above but Servo_In4 controls Motor2
Mix Servos for Motor Control	3	Servo signal inputs 3 and 4 are mixed to create drive signals for both motors. This allows you to create a skid steering robot with Servo_In3 controlling speed (forward and reverse) and Servo_In4 controlling direction (left and right).
Disable Servo Inputs	4	Setting this disables the code that reads pulses at the Servo_Inx pins. This allows you control the contents of the Servo_Inx registers through the serial interface and may be useful in testing special operating modes (above).

Indicator Register (index = 4): The Status register contains bit values that indicate important conditions in the module.

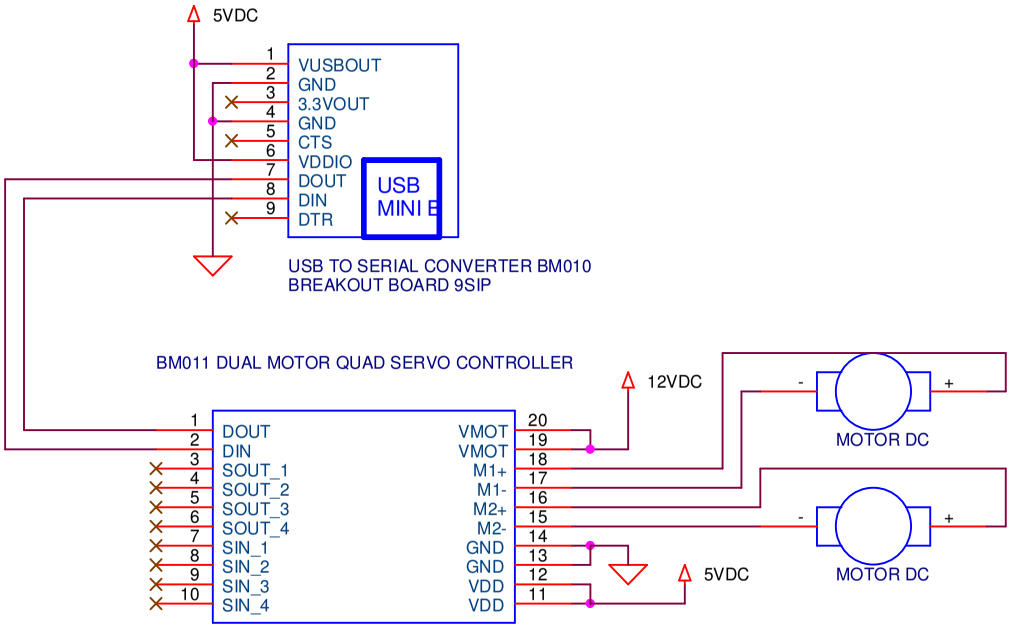
Indicator	Bit	Description
H-Bridge Fault	0	Set when one or both H-bridges have experienced a fault condition. This bit can only be cleared by resetting the H-bridges via the Command register (see Reset H-bridge command). This bit being set may not indicate a fault condition is present. Under normal operation the module will attempt to reset the H-bridges if a fault occurs.
	1	unused
	2	unused
	3	unused
	4	unused
	5	unused
	6	unused
	7	unused

Schematics:

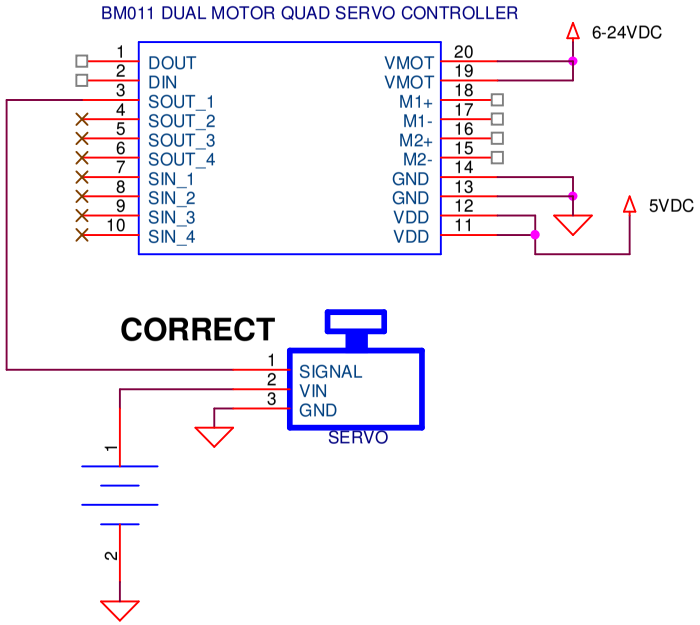

VMOT
M1OUT2
U1
MC33926PNB
C1
0.1uF
VDD
R1
33K
U2
PIC16F1829-IML
ENABLE
STATUS
R2
270 Ohm
M1OUT1
M2OUT2
M2OUT1
U3
MC33926PNB
C4
0.1uF
R15
270 Ohm
P1
DOUT
P2
DIN
P3
S_OUT1
P4
S_OUT2
P5
S_OUT3
P6
S_OUT4
P7
S_IN1
P8
S_IN2
P9
S_IN3
P10
S_IN4
R3 270 Ohm
R4 270 Ohm
R5 270 Ohm
R6 270 Ohm
R7 270 Ohm
R9 270 Ohm
R11 270 Ohm
R12 270 Ohm
R13 270 Ohm
R14 270 Ohm
R10 270 Ohm
D1
Red
C3
0.1uF
R8 33K
J1
PGRM/DEBUG
MCLR
VDD
GND
PGD
PGC
C2
47uF
6VDC-20VDC
C5
10uF
3.3VDC-5VDC
J2 HEADER 2
J3 HEADER 2
J4 HEADER 2
J5 HEADER 2
J6 HEADER 2
J7 HEADER 2
J8 HEADER 2
J9 HEADER 2
FID1
FID2
MH1
85 mil
MH2
85 mil

Application Schematics:

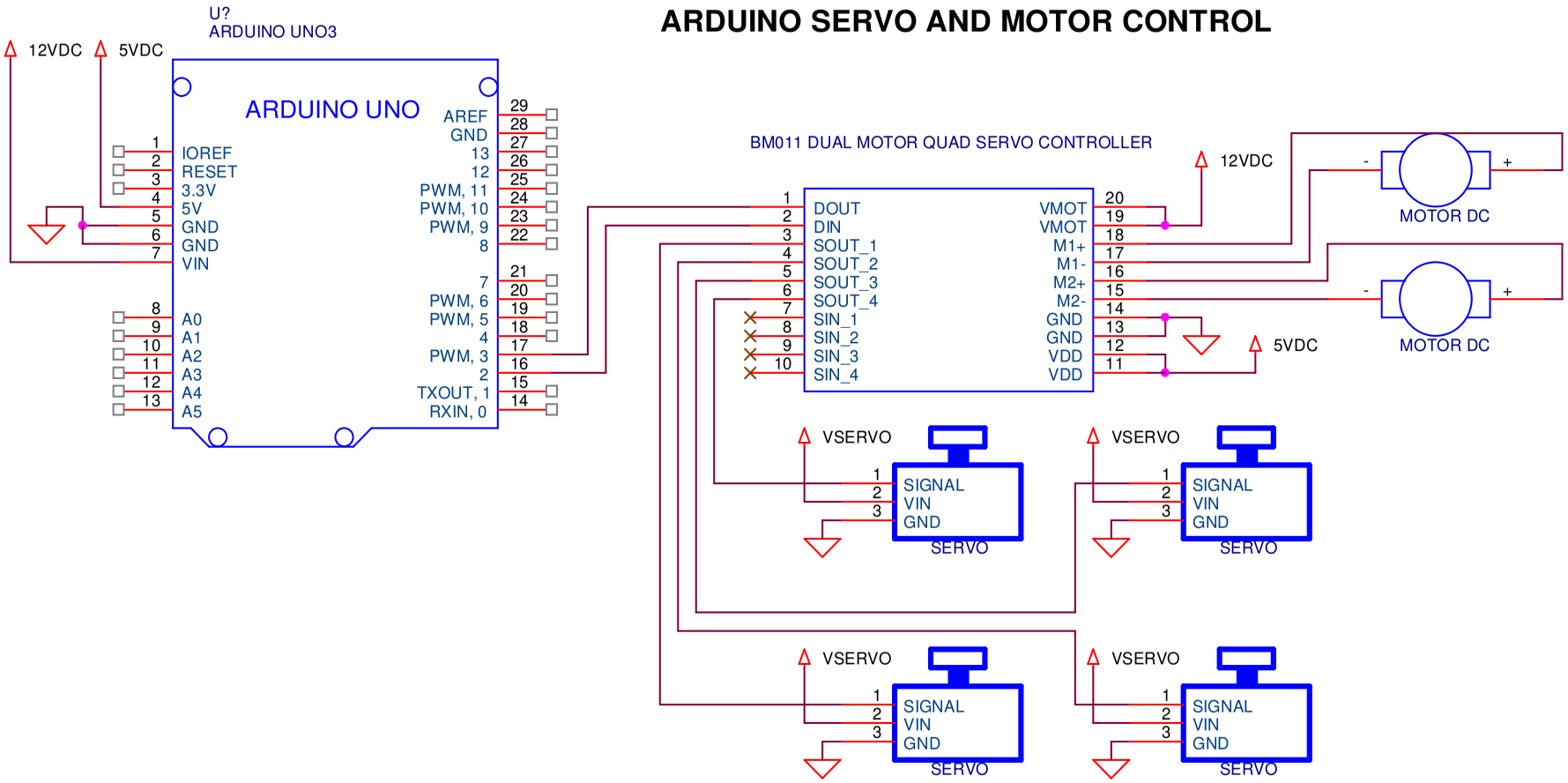
DUAL MOTOR CONTROL OVER USB INTERFACE



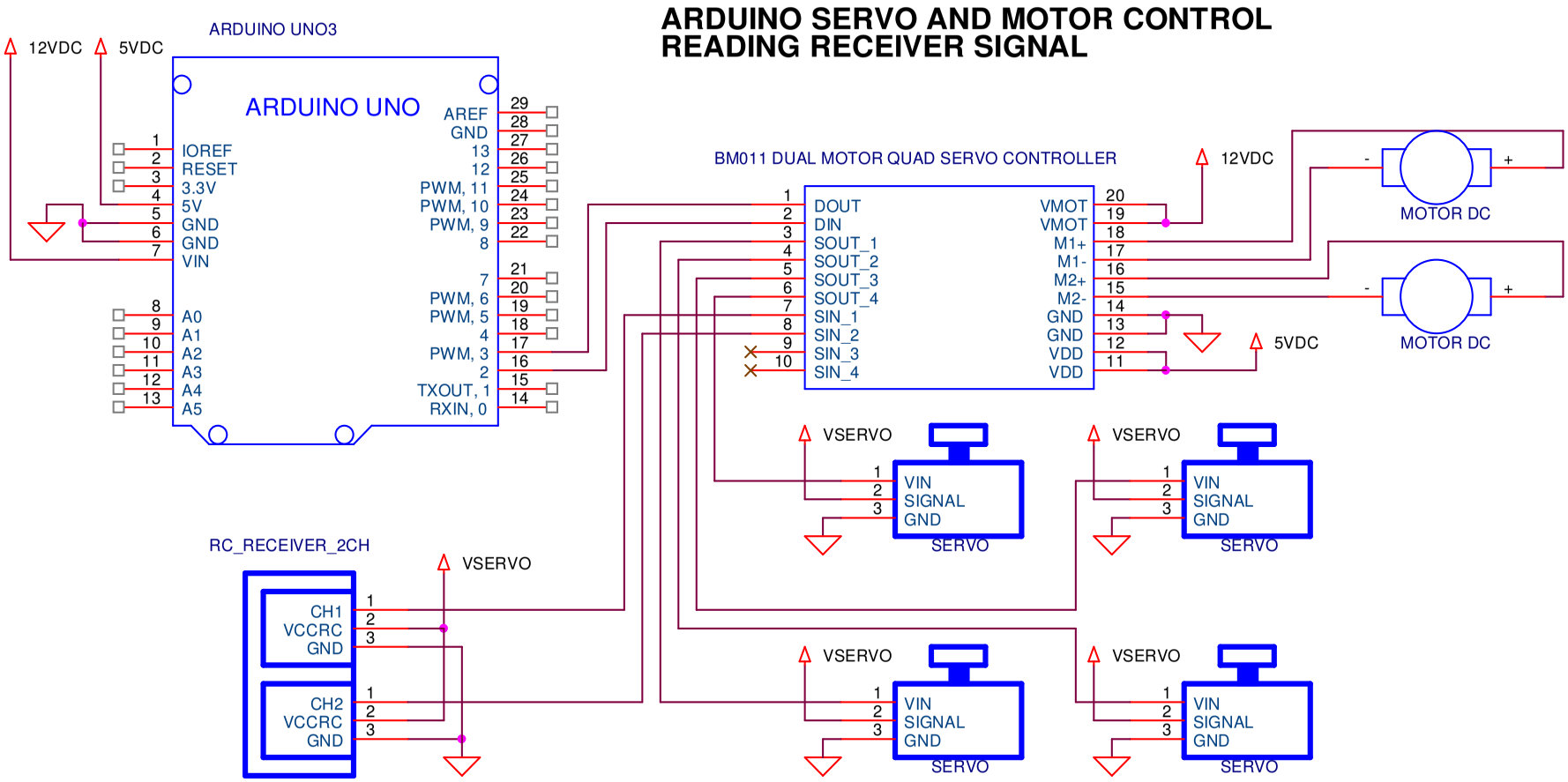
IF SERVOS DRAW HEAVY LOAD WIRE THEIR POWER DIRECTLY TO THEIR SUPPLY



Application Schematics:



Application Schematics:



Application Schematics: