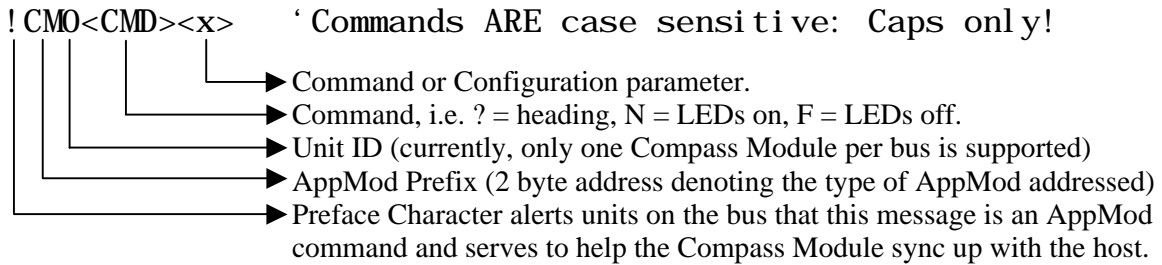


Serial Protocol

The host transmits commands serially to the Compass Module. The command syntax is structured to facilitate the allowed commands and prevent bus contention in the event that more than one AppMod is on the same serial line. The Compass Module uses I/O pin 5 (P5) for the serial interface. Since the Compass Module needs a little time both for power up and in-between messages, it is necessary to have a delay before each serial message. A 'pause 50' command is usually a sufficient delay between messages.



Commands

? Request the Compass Module to report its current heading and status.

```
serout 5, 188+$8000, ["!CMO?"]
```

' Report the current heading and status

After this message is sent, the Compass Module will reply with a single byte. Use the line of code below to receive it.

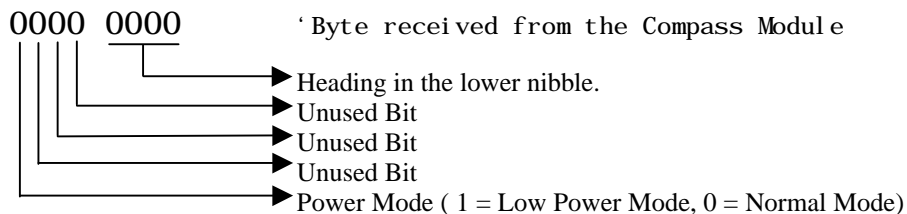
```
serin 5, 188+$8000, [heading]
```

' Receive heading from Compass Module.

The lower nibble (four bits) will contain an value that ranges from 0 – 15 that indicates the current heading. The chart below correlates the nibble values to their respective headings.

Nibble Value	Binary Value	Heading	Nibble Value	Binary Value	Heading
0	0000	N	8	1000	S
1	0001	NNE	9	1001	SSW
2	0010	NE	10	1010	SW
3	0011	ENE	11	1011	WSW
4	0100	E	12	1100	W
5	0101	ESE	13	1101	WNW
6	0110	SE	14	1110	NW
7	0111	SSE	15	1111	NNW

The upper four bits of the reply indicate the status of the unit. The legend below depicts the individual bit meanings.



the Debug window.

```
Heading      var      byte
Start:       pause 200
             serout 5, 188+$8000, ["!CMO?"]
             serin  5, 188+$8000, [Heading]
             debug ?Heading
             goto  Start
```

- D** Set the desired direction according to the parameter sent. The Compass Module will pulse the serial line low when the Compass Module arrives at the desired heading.

```
serout 5, 188+$8000, ["!CMOD", 0]      ' Desired direction is North
```

This command will set the Compass Module's desired direction pointer to North. When the Compass Module's actual direction matches the desired direction, the Compass Module will pulse the serial line low for approximately 125mS. The idea here is that the stamp can set the desired direction in the Compass Module then start the 'Bot (or whatever it is mounted to) turning. While this is happening, the stamp monitors P5 for a low going pulse. The pulse is of sufficient length to allow the stamp to poll it while doing other things, like driving servos, etc. When the stamp detects the pulse, it stops the 'Bot from turning and begins the next move, (whatever that may be).

There are a couple of interesting situations that can arise during the course of operations. Here are the ones that we've foreseen, for your review:

- Setting the desired direction to the direction that the Compass Module is already heading. When this occurs, the Compass Module will wait for 5mS, (5 milliseconds), then pulse the serial line low for 125mS.
- Setting the desired direction to a direction that, for any reason, is not obtainable. Herein lies the problem that the Compass Module will wait forever, looking for a direction that it will never see. Consequently, it will not respond to any further serial commands. The way to back out of this situation is to have the stamp pulse the serial line low for 50mS. When the Compass Module is processing any desired direction command, it also monitors the serial line for a low-going pulse. The low going pulse in this case signals the Compass Module to reset itself (soft-boot). Please note that you should send an 'I' command to reset the module otherwise.

Here's a little program that shows how to use this command with a BoeBot and a BS2:

```
West      con 12                                ' Value 12 from look table.

Start:    pause 200                             ' Wait for system to power up.
          gosub GoWest
          end

GoWest:   serout 5, 188+$8000, ["!CMOD", West]   ' Signal when we're heading due west.
GWyet:    pul sout 12, 700                       ' Rotate the ' Bot
          pul sout 13, 700
          pause 20
          if in5 = 1 then GWyet                  ' Until we are facing west.
          return
```

I Initialize and reset the Compass Module.

```
serout 5, 188+$8000, ["!CMOI"]
```

This command will reset the Compass Module and clear the desired direction command register. The Compass Module behave as though the power was switched off, then on again.

P Put the unit in low-power mode by switching off the LEDs.

```
serout 5, 188+$8000, ["!CMOP"]
```

This command will set the Compass Module to low-power mode. The module will still reply to all commands sent to it but will not illuminate the LEDs. To recover from low-power mode, simply send an 'I' command to initialize the Compass Module.

Sample Program

One thing that would be desirable is, when commanding your 'Bot to turn to a particular direction, it should figure out which way is the best way (shortest way) to rotate to obtain the commanded position. Here's a modified version of a previous example that shows just how easy it is to do with a BS2 and a Compass Module:

```
Heading var byte
West con 12 'Value 12 from table.

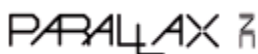
Start: pause 200 'Wait for system to power up.
      gosub GoWest 'Whatever direction its currently facing,
      end 'make it face west, then stop.
GoWest: serout 5, 188+$8000, ["!CMO?"] 'Get initial heading
        serin 5, 188+$8000, [Heading]
        serout 5, 188+$8000, ["!CMOD", West] 'Set desired direction = West
        if Heading - West = 0 then GWend 'If we're not heading west already AND
        if Heading - West > 0 then GWcw 'If we need to rotate counter-clockwise
GWccw: pul sout 12, 800 'Rotate the 'Bot CCW
       pul sout 13, 800
       pause 20
       if in5 = 1 then GWccw 'Until we are facing west.
       goto GWend 'Else if we need to rotate clockwise
GWcw: pul sout 12, 700 'Rotate the 'Bot CW
      pul sout 13, 700
      pause 20
      if in5 = 1 then GWcw 'Until we are facing west.
GWend: return 'Then return (exit)
```

Of course this program could be re-written to optimize for code space, but it was written for maximum clarity instead. Feel free to optimize away if you wish.

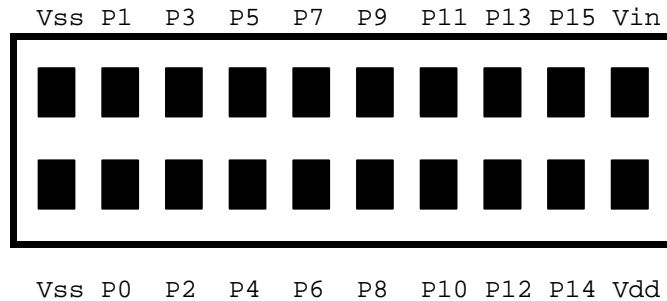
Electrical Specifications

The Compass Module AppMod requires 6 – 12 Volts DC and will draw about 25-50mA depending on how many LEDs are on.

As with all AppMods, the Compass Module AppMod has a special stack-through 2x10 header. This header allows you to connect the Compass Module to either the GrowBot, the BOE-Bot, the Stamp Activity Board, the Super Carrier Board, and of course, other AppMods. We've provided the pin out of the AppMod header below, as viewed from the top.



If you will not be connecting this AppMod to a board with the mating stack-through header, all you need to do is: connect Vss to ground, connect Vin to the positive side of a 6-12 Vdc supply, and connect P5 to the serial host. **Please note: P5 can tolerate a voltage swing from 0 to 5 Vdc only.** Here's a top view of the stack-through header:



Programming Notes

1. Throughout the examples mentioned herein, I/O Pin 5 has been referenced as the serial pin for all serial communications. This is because, with respect to the AppMod header included on various products such as the GrowBot, the Board of Education, the BOE-Bot, and the SuperCarrier Board, I/O Pin 5 has been hardwired to serve this function.
2. The examples mentioned herein assume the use of the BS2-IC. Different commands or parameters may be required if you are using something other than the BS2-IC. Please refer to the AppMod code section on the Parallax CD for code examples for the other stamps.
3. Please note that the **0** in the **CMO?** command is a numeric (zero) and not an alphabet character. Care must be taken since the two characters look alike in many fonts. The zero has no function in this AppMod. Others AppMods may use this character location to specify a module number such that multiple AppMods of the same type may share an I/O line.
4. If the Compass Module is sent a partial or erroneous message, it will eventually time out after about 2.3 seconds and reset itself. If it seems to ignore every message you send it, perhaps it was sent a desired direction command that remains unsatisfied. In this case, you can reset the Compass Module by pulsing its serial input line (P5) low for 50 mS.
5. When using the 'D' command, the Compass Module will pulse the serial data line (P5) low for approximately 125mS. This should allow plenty of time, no matter which stamp you use, for the stamp to detect the pulse.
6. On the other hand, when using multiple 'D' commands, bear in mind that the Compass Module can not receive serial commands while the serial data line is pulsed low. So, you may wish to add a little logic to test that the serial data line is high (5Vdc) before sending the next serial command. Here's how:

Heading	var	byte	
West	con	12	
North	con	0	
Start:	serout 5, 188+\$8000, ["!CMOD", North]		' Request a north vector
Wait1:	pulsout 12, 800		' Rotate the 'Bot
	pulsout 13, 800		
	pause 20		
	if in5 = 1 then Wait1		' Until we're facing north
Wait2:	if in5 = 0 then Wait2		' Wait until pulse is through
	serout 5, 188+\$8000, ["!CMOD", West]		' Then continue with next command
	etc...		

