# Compass Module AppMod

## General Description


Don't know where to take your project next? The Compass Module can give your application direction. Typically, most people are relegated to spending and arm and a leg to buy a GPS unit and then invest a ton of time writing a software to interface to it. Thanks to the folks at the Dinsmore Instrument Company, this is no longer difficult nor expensive. Their new little compass sensor provides a low cost, direct interface, direction sensor that is perfect for many applications. Additionally, we added a microprocessor to fleshout a system and make the module as useful and as easy to use as it can be.

## Features

- **Intuitively depicts eight directions with just four LEDs.**
- **Serial interface 'auto-bauds' to any standard baudrate from 2400 to 38.4K.**
- **Direction feedback scheme signals back when a particular heading is reached.**
- **Serial selectable low-power mode for battery powered applications.**
- **Serial selectable operational mode makes it easy and intuitive to read.**

## How it Works

The Dinsmore Compass Sensor is an electro-mechanical device. As a result, it is sensitive to shock and the relative angle that it operates at. The manufacturer of the module has dampened the mechanical movement of the compass to give it a 'liquid filled' behavior. Roughly, it has a 3.5 second response time for a 90° rotation. The sensor needs to operate parallel to the ground, plus or minus 5°. Considering all this, the sensor is perfect for a small, indoor robot, but may not be suitable for an RC airplane-based application.

The PIC® chip on board handles the serial communication, reads the compass sensor, and controls the LEDs. Unlike a standard mechanical compass that has no PIC chip, the Compass Module AppMod is easy to read and intuitive to use in that when the 'W' LED is on, the heading is west. A northeast heading is represented by both the 'N' and 'E' LEDs on. However, on a standard compass, when the needle points to the west your actual heading is east (this is because the needle of a standard compass always points north).

The serial interface provides a convenient, easy to use interface that uses only one I/O line. Some serial commands execute almost instantaneously, but others require some time to execute before the module is ready to receive another command. So, please pay careful attention to the examples included herein.

* PIC Is a registered trademark of Microchip Technologies

## Operational Modes

The Compass Module has three user selectable operational modes: Stand Alone Mode, Low Power Mode, and Compass Emulation Mode. All modes are run-time selectable via serial commands. All of the serial commands are discussed in detail in the Commands section of the document.

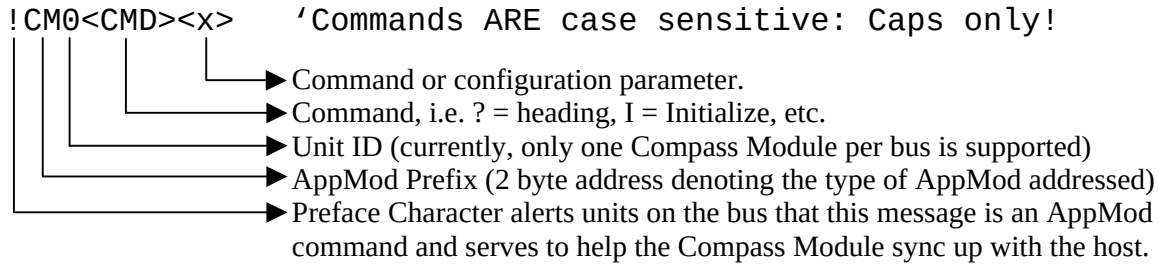
The Compass Module powers up in **Stand Alone Mode.** In Stand Alone Mode, the LEDs will update in realtime (as you move the module about) but will not respond to serial commands. Once you pulse the serial data line P5 low, the Compass Module will switch to Serial Mode and respond to subsequent valid serial commands. To revert back to Stand Alone Mode, you can either send an 'A' command, or cycle power.

In **Low Power Mode**, the LEDs will not illuminate although the Compass Module will still respond to serial commands. The Compass Module powers up to Normal Power Mode (LEDs enabled). If Low Power Mode is desired, the user may enable this mode by sending a 'C' command. Current consumption of the Compass Module is detailed in the Electrical Specifications section of this document.

In **Compass Emulation Mode**, the reported heading and the indication on the LEDs will behave just as a standard mechanical compass. In other words, the needle in the standard compass will simply point north always. In this case, it is up to the user to infer the actual heading. The default mode of the Compass Module is to indicate the actual heading, not behave as a simple compass. If Compass Emulation mode is desired, the user may enable this mode by sending the 'C' command.

## 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 processing time both for power up and between some messages, it may be necessary to have a delay before each serial message. A 'pause 1' command provides an adequate delay.



## Switching to Serial Mode

First, you need to switch the Compass Module to Serial Mode. Here are three different ways to switch from Stand Alone mode to Serial Mode. Any one of these Methods will do the job. Just place one of them in the intialization area of your code ( it should execute once ).

Method I

```
high 5                  'Must initialize the line high
pulsout 5,500           'Low going pulse for 1 mSec.
input 5                 'Return the line to high-z mode.
```

Method II

```
low 5                   'Bring the line low
pause 1                 'for 1 mSec.
input 5                 'Return the line to high-z mode.
```

Method III

```
serout 5,84+$8000,[0]  'Use 9600 (or slower) baudrate
                       'Sending a 0 pulses the serial line low.
```

After the Compass Module is switched to Serial Mode, serial commands may be issued. Here is a list of the allowed commands and a description of what they do. Please refer to the Sample Programs section for working examples of how to use these commands to do neat things.

## Commands

**A** Switch the Compass Module from Serial Mode to Stand Alone Mode.

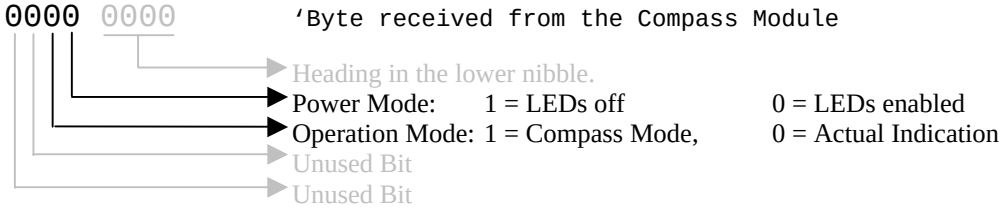
```
serout 5,188+$8000,["!CM0A"]
```

This command will switch the Compass Module to Stand Alone Mode. The LEDs will update in realtime, but the module will not respond to serial commands.

**C** Configure the Compass Module mode control bits.

```
serout 5,188+$8000,["!CM0C",%11]
```

With this command, the Compass Module's configuration may be altered under serial control by setting and/or clearing the mode control bits. Currently, only the Power Mode and the Operational Mode bits are implemented. The legend below shows the relative location of the configuration bits.




**I** Initialize the Compass Module.

```
serout 5,188+$8000,["!CM0I"]
```

This command will perform a soft reset on the Compass Module. All internal registers will be cleared, all LEDs will be illuminated, and the Compass Module will remain in Serial Mode and Normal Power Mode.

## Commands (continued)

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

```
serout 5,32+$8000,["!CM0?"]          ' Report the current heading and status
```

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

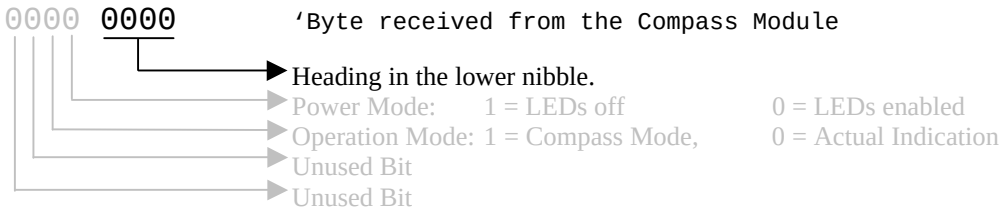
```
serin  5,32+$8000,[heading]          ' Receive heading from Compass Module.
```

The lower nibble (four bits) will contain an value that ranges from 0 – 7 that indicates the current heading. The chart below correlates the returned value to the respective heading.

| Lower Nibble | Binary Value | Heading |
|---|---|---|
| 0 | 0000 | N |
| 1 | 0001 | NE |
| 2 | 0010 | E |
| 3 | 0011 | SE |
| 4 | 0100 | S |
| 5 | 0101 | SW |
| 6 | 0110 | W |
| 7 | 0111 | NW |

**Table 1**

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




Here's a mini sample program for the BS2 that will switch the Compass Module to serial mode, query the Compass Module, and echo the reply to the Debug window about four times per second.

```
'{$STAMP BS2}                              'Stamp type declaration

Heading        var    byte

               pause 200                   'Wait for system to power up.
               serout 5,188+$8000,[0]      'Switch the to Serial Mode
Start:         serout 5,32+$8000,["!CM0?"] 'Send the Query command
               serin  5,32+$8000,[Heading] 'Receive the reply
               debug ?Heading              'Send the reply to the debug screen
               pause 200                   'Repeat roughly four times per second
               goto Start
```

## Commands (continued)

**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. The parameter is a number, 0 through 7 inclusive (from Table 1), that corresponds to the desired direction.

```
serout 5,188+$8000,["!CM0D",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 pointer, the Compass Module will pulse the serial line low for approximately 25 mS. 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 delayed by at least 750 uS and is of sufficient length to allow the stamp to poll it while doing other things, like driving servos, checking whisker inputs, 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 750 uS then pulse the serial line low for 25 mS.
- 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 100uS. 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 6                                 'Value West = 6 from Table 1.

Start:  pause 200                             'Wait for system to power up.
        serout 5,188+$8000,[0]                'Switch to Serial Mode
        debug cr,"Turning to the west…"
        gosub GoWest
        debug cr,"We are now heading west!"
        end

GoWest: serout 5,32+$8000,["!CM0D",West]      'Signal when we're heading due west.
GWyet:  pause 20                              'Wait pulse servo delay
        if in5 = 0 then GWdone                'If not there yet,
        pulsout 12,725                        '  rotate the 'Bot slowly
        pulsout 13,725                        '  and check for pulse again.
        goto GWyet                            'Else
GWdone: return                                '  return to main
```

## Sample Programs

The first type of program that we should examine is one that puts the 'C' command through its paces.

```
Init:   pause 200                           'Wait for system to power up.
        serout 5,188+$8000,[0]              'Switch to Serial Mode
Start:  serout 5,32+$8000,["!CM0C",%01]     'Disable Compass Mode,  enable Low Power mode
        pause 1000
        serout 5,32+$8000,["!CM0C",%10]     ' Enable Compass Mode, disable Low Power Mode
        pause 1000
        serout 5,32+$8000,["!CM0C",%11]     ' Enable Compass Mode,  enable Low Power Mode
        pause 1000
        serout 5,32+$8000,["!CM0C",%00]     'Disable Compass Mode, disable Low Power Mode
        pause 1000                          'note that this is the same as an 'I' command
        goto Start                          'also, there will be no visable indications
```

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:

```
Speed          var     word
Act            var     byte
Des            var     byte
CW             var     byte
CCW            var     byte
Ctr            var     byte
MidL           con     744
MidR           con     746

Init:   pause 200
        serout 5,188+$8000,[0]        'Initialize
        Des = 4                       'Set desired direction
ChkDir: serout 5,32+$8000,["!CM0?"]   'Get current direction
        serin  5,32+$8000,500,Init,[Act]
        if Des = Act then ChkDir              'If Act <> Des
        CW  = Act                     ' then we need to figure out
        CCW = Act                     ' which is the best (shortest)
        Ctr = 0                       ' direction to rotate.  This
Rloop:  CW  =  (CW+1)&7                       ' is done using two counters:
        CCW = (CCW-1)&7                       ' one that counts clockwise
        Ctr = Ctr + 1                 ' while the other counts in
        Speed  = -18                  ' the counterclockwise direction.
        if CCW = Des then Turn        ' Based on which one reached the
        Speed  =  15                  ' desired direction first, the
        if  CW = Des then Turn        ' speed (direction of rotation)
        goto Rloop                    ' is decided and the 'Bot turns
Turn:   pulsout 13,MidL+Speed*Ctr             ' until the desired direction is
        pulsout 12,MidR+Speed*Ctr             ' reached.
        pause 10
        goto ChkDir
```

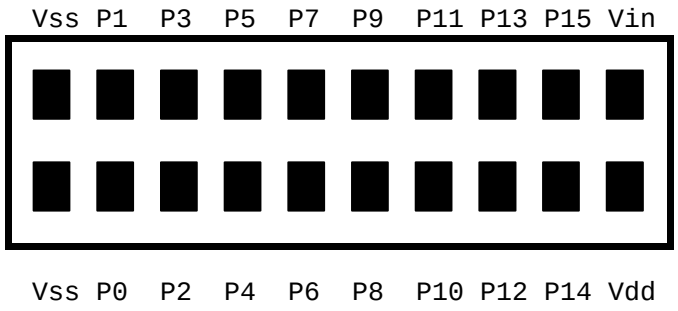
Please note that the relative direction of the Compass Module may be 180 degrees out of phase with the actual bearing of the foundation upon which it may be mounted. For BoEBot users, this will be the case and you will need to re-assign the numbers that correlate to the directions such that North would be four instead of zero, etc.

## Electrical Specifications

The Compass Module AppMod requires 6 – 24 Volts DC and will draw between 30 and 72mA 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 6-24 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 `CM0?` command is a numeric (ascii zero, 30h) and not an alphabet character. Care must be taken since the two characters look alike in many fonts. The zero has no function except for a placeholder 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 ~1 mS.

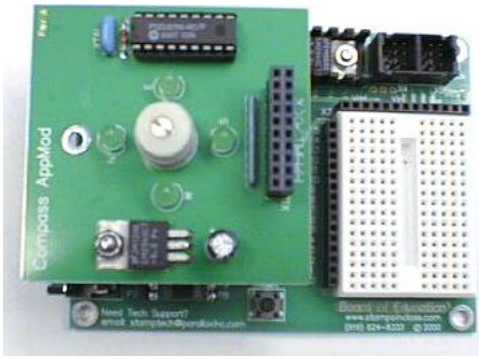
5. When using the 'D' command, the Compass Module will pulse the serial data line (P5) low for approximately 750uS. This should allow plenty of time, no matter which stamp you use, for the stamp to detect the pulse.

6. On the other hand, when using multiple 'D' commands, bear in mind that the Compass Module cannot receive serial commands while the serial data line is pulsed low. So, you may wish to add a little code to test that the D command reply has completed (serial data line is high at5Vdc) before sending the next serial command.

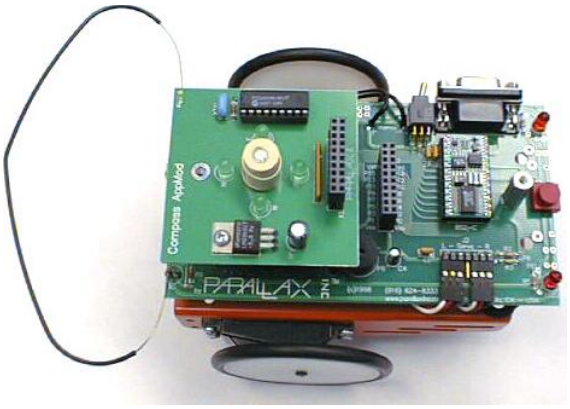
## Connection and Orientation

The Compass Module AppMod can be used with any base PC board that sports an AppMod header. Different base boards orient their AppMod headers differently. In general, you need to locate and match pin 1 to pin 1. If there is no label as such, you may look at the shape of the pad on the PC board. The square pad is pin 1. Here's a couple of photos that depict how to plug the Compass Module AppMod into the various base boards that offer AppMod headers.

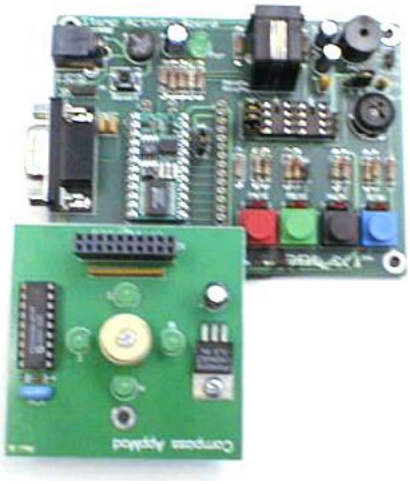
### Board of Education/SuperCarrier




### The GrowBot




### BASIC Stamp Activity Board



### DeA Training 'Bot