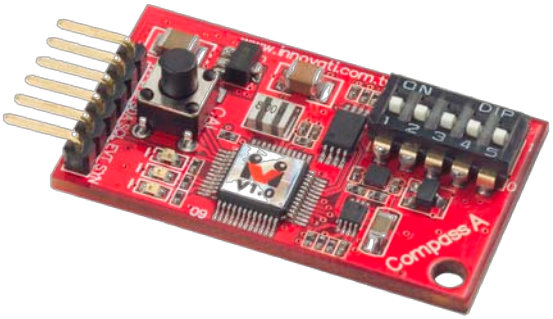
Innovati Compass A Module

Version: V1.0



Product Overview: Innovati's Compass A Module is an easy-to-use, high precision electronic compass. Accessed via cmdBUS and BASIC Commander, Compass A provides users, through simple instructions, with the current directional angle with respect to North and the magnetic field intensity. Besides, Compass A can be calibrated at any place, at any time, making itself capable of all kinds of applications. Please use "**CompassA**" as the module object name in program.

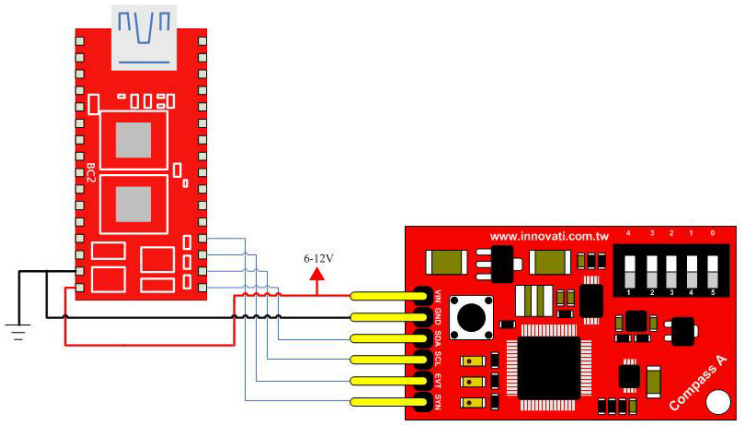
Applications:

- Designs that obtain digitized directional angle with respect to North with electronic compasses.
- Carriers that moves in a fixed direction. Compass A's deviation angle detection enables carriers to move in a fixed direction.
- Applications related to magnetic field intensity measurement.

Product Features:

- Directional angles with respect to North in unit of degrees.
- 3-axis magnetic field measurement.
- Special deviation angle function: Compass A gives users directional angles with respect to certain directions, not limited to North.
- Deviation reminders: Compass A sends out reminders when the current direction is out of the preset directional angles.
- Automatic refresh of current directions: Compass A has 5 refresh rates and users can switch to a proper refresh rate at any time.
- Easy calibration: Users can activate the calibration mode through software or hardware buttons at any time. Scheduled automatic calibrations and calibration completion reminders are also available.
- Up to 256 angle storage memories: Compass A saves up to 256 current angles, arbitrary angles as the base angles of the deviation angle function.
- High directional angle precision: up to degrees.
- Maximum detectable magnetic intensity up to ±300μT.
- A minimum resolution of 0.6μT in the x and y axes and 1.2 μT in the z axis.
- Up to 20 measurements per second.

Connection: To access Compass A through BASIC Commander, set the ID switches to the desired number settings, and connect the cmdBUS to the proper pins on the BASIC Commander.

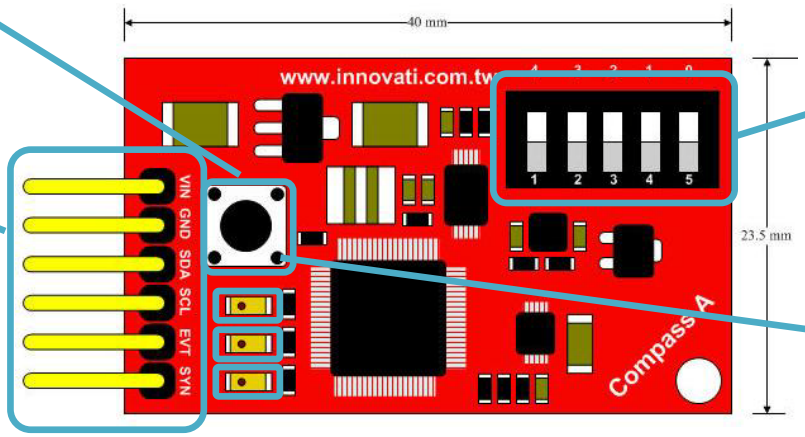


Product Specifications:

cmdBUS pins: To access the Compass A module through the BASIC Commander, connect these pins to the corresponding pins on the BASIC Commander. (When connecting Compass A, please pay attention to pin assignments. For example, the Vin on the Compass A module should be connected to the Vin on the BASIC Commander. Incorrect pin connection may damage both modules.)

Module ID Switches: These switches determine Compass A's module ID in binary format, from right to left. Module IDs enable BASIC Commander to distinguish those modules under its control. (Please refer to Appendix 2.)

Calibration button: To enter the calibration mode, push and hold the button for two seconds when the calibration LED is off. ~~when the calibration LED is not flickering.~~ After pushing the button, the calibration LED starts to blink and the module is in calibration mode. To exit the calibration mode, press again the button while the calibration LED is flickering. After pressing the button, the calibration LED ~~stops blinking~~turns off.



The three LEDs from top to bottom are:
Calibration LED: flickers when the module is calibrating itself;
Event LED: flickers when the module is transmitting events;
Instruction LED: flickers when the module is communicating with SBC.

Figure 1: Pin assignment and switch description.

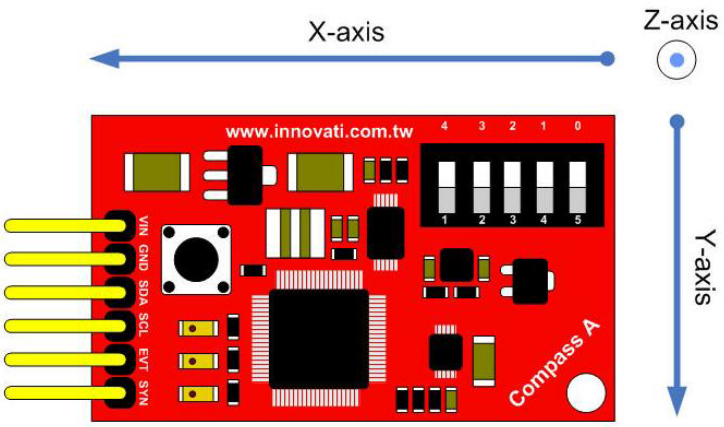


Figure 2: Axis definition

During calibration, please keep the module horizontal and rotate it perpendicular to the z-axis (as shown in the figure). Note that the calibration LED is flickering during calibration. Please do not rotate the module too fast so that the module can determine the limits in all axes, and rotate the module more than 360 degrees. The module can be rotated either clockwise or counterclockwise.

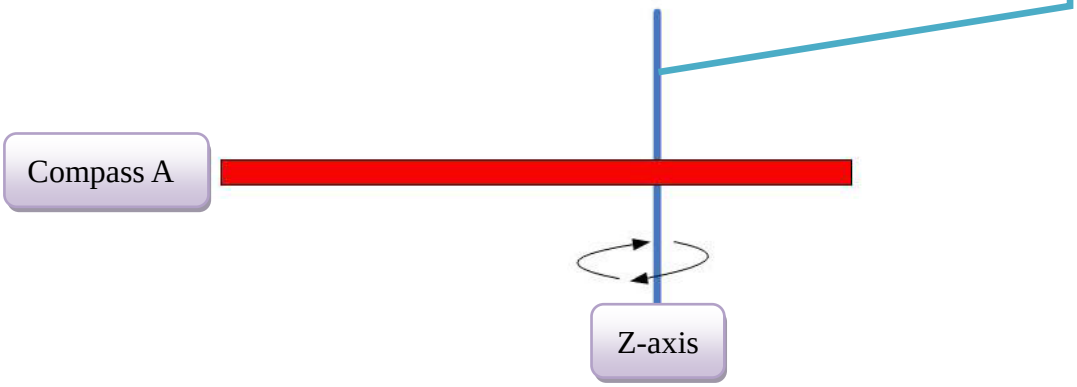


Figure 3: Calibration

Precautions:

- Magnetic field intensity differs by occasion due to various equipment interferences. It is recommended to calibrate Compass A module before use.

- Compass A's zero readings may not be the direction of actual magnetic North when it is near strong magnetic fields. Instead, the direction with zero readings could be the direction of that magnetic field.

- To get more accurate measurements, place the module as horizontally as possible.

Absolute Maximum Ratings:

Operating temperature: 0 °C ~ 70 °C

Storage temperature: -40 °C~125°C

Commands and Events:

The following tables list all the unique commands and events provided with the Servo Runner A Module. Note that essential words in the commands will be written in **bold** type and ***italics*** in bold type. The bold type word must be written exactly as shown, whereas the italic bold type words must be replaced with the user values. Note that the innoBASIC language is case-insensitive.

Command Format	Function of the Command
Commands for measurement of directional angles and magnetic field intensity	
GetXField(*FieldX*)	Read the difference between the central magnetic field intensity and the axial magnetic field intensities of the x-axis and y-axis. The magnetic field of the x-axis is stored in ***FieldX***, while that of the y-axis ***FieldY***. The return value ranges from -32768 ~ 32767.
GetYField(*FieldY*)	
GetXYField(*FieldX, FieldY*)	
GetAngle(*Angle*)	Read the directional angle of the magnetic North clockwise with respect to the x-axis in unit of degrees. The angle is stored in ***Angle*** which ranges from 0~359.
GetField(*Field, Angle*)	Read the directional angle of the magnetic North clockwise with respect to the x-axis in unit of degrees. The angle is stored in ***Angle*** which ranges from 0~359 while the measured magnetic intensity is stored in ***Field***, ranging from 0~65535.
GetHxHyHz(*Hx, Hy, Hz*)	Return the magnetic intensities of the x, y, and z-axes and stores them in ***Hx***, ***Hy*** and ***Hz***, respectively. Each return value ranges from -32768 to 32767. The sign of the return value represents the direction of the magnetic field.
Commands for measurement and setting of deviation angles.	

SaveCurrAngle (*Number*)	Save the currently measured angle to the position specified by ***Number***, which ranges from 0 to 255.
SaveAngle(*Number, Angle*)	Save the value of ***Angle*** to the position specified by ***Number***. ***Number*** ranges from 0 to 255 while ***Angle*** 0 to 359.
LoadAngle(*Number, Angle*)	Read the value stored in the position specified by ***Number*** and saves it in ***Angle***. ***Number*** ranges from 0 to 255 while the return value of ***Angle*** is between 0 and 359.
GetDevAngle(*Number, Angle*)	Read the deviation angles of the current direction with respect to the preset base. The function uses the directional angle saved in the memory block specified by ***Number*** as the base direction. It returns the deviation angle of the current measurement with respect to the base direction in unit of degree and saves the value in ***Angle***. When the current measurement lies within 180 degrees counterclockwise with respect to the base direction, this function returns a positive value. Likewise, this function returns a negative value when the current measurement lies within 179 degrees clockwise with respect to the base direction. The return value of ***Angle*** ranges from 180 to -179.
SetDevAngleLimit(*Angle*)	Set the limit of deviation angle in unit of degrees. ***Angle*** ranges from 0 ~ 179 and its default value is 5.
GetDevAngleLimit(*Angle*)	Return the current deviation angle limit setting in unit of degrees and saves the value in ***Angle***.
SetDevAngleNum~~ber~~(*Number*)	Set the value stored in the memory position specified by ***Number*** to be the base direction. ***Number*** ranges from 0 to 255.
GetDevAngleNum~~ber~~(*Number*)	Return the memory position where the current base angle is stored and saves this value in ***Number***. The return value ranges from 0 to 255.
EnableDevAngleLimitEvent()	Enable the deviation angle limit event.
DisableDevAngleLimitEvent()	Disable the deviation angle limit event.
***Status* = GetDevAngleLimitStatsu()**	Check if the current directional angle exceeds the deviation angle limit. When the current direction angle exceeds the limit, this function returns 1 in ***Status*** and otherwise 0.
Commands related to refresh and calibrating Compass A module.	
SetRefreshFreq(*Rate*)	Set the refresh rate of the directional angle measurement by the value specified in ***Rate***. 5 refresh rates are available: ***Rate*** = 0 -> Refresh the angle measurement every 50 ms

	(20Hz) ***Rate*** = 1 -> Refresh the angle measurement every 100 ms (10Hz) ***Rate*** = 2 -> Refresh the angle measurement every 250 ms (4Hz) ***Rate*** = 3 -> Refresh the angle measurement every 500 ms (2Hz) ***Rate*** = 4 -> Refresh the angle measurement every 1000 ms (1Hz)
GetRefreshFreq(*Rate*)	Return the angle measurement refresh rate setting. The return value of ***Rate*** ranges from 0 to 4. The values are defined in the same way as **SetRefreshFreq**().
Status **= GetRefreshStatus()**	Check the refresh status. When the angle measurement is refreshed, it returns 1 in ***Status***. After checking the status, the system sets the status back to zero. It will be set to 1 again after the angle measurement is refreshed.
EnableRefreshEvent()	Enable the angle measurement refresh reminder event
DisableRefreshEvent()	Disable the angle measurement refresh reminder event.
ABConvert(*Angle, Binary*)	Convert the input ***Angle*** into a binary output on a scale of 360 degrees equal to 256 and saves the output value in ***Binary***. ***Angle*** ranges from 0 ~ 65535 and the return value of ***Binary*** from 0 ~ 65535.
BAConvert(*Binary, Angle*)	Convert the input ***Binary*** into an angular value on a scale of 256 equal to 360 degrees and saves the return value in ***Angle***. ***Binary*** ranges from 0 ~ 65535 and the return value of ***Angle*** from 0 ~ 65535.
Calibration(*Time*)	Set the module calibration duration by the input value of ***Time***. 5 different calibration duration are available: ***Time*** = 0 -> Keep calibrating the module until the button is pressed. ***Time*** = 1 -> Calibrate the module for 10 seconds. ***Time*** = 2 -> Calibrate the module for 20 seconds. ***Time*** = 3 -> Calibrate the module for 30 seconds. ***Time*** = 4 -> Calibrate the module for 60 seconds.
GetCalValue(*Hx, Hy*)	Save the calibration values of the x- and y-axes in ***Hx*** and ***Hy***, respectively. The return values of ***Hx*** and ***Hy*** range from -32768 to 32767.
SaveDefaultCalValue()	Save current calibration values to the default ones.
LoadDefaultCalValue()	Load the default calibration values to the current ones.

Event provided by the module:

Event	Activation Condition
FieldRefreshEvent	After **EnableRefreshEvent**() is executed, this event is activated when the module refresh the current angle measurement. The refresh time varies by the setting of **SetRreshFreq**().
DevAngleLimitEvent	After **EnableDevAngleLimitEvent**() is executed, this event is activated when the deviation of the current directional angle measurement from the base direction exceeds the setting of **SetDevAngleLimit**(). **SetDevAngleNumber**() sets the base direction.
CalEndEvent	This event is activated when Compass A calibration completes.

Example Program:

```
Peripheral myCompass As CompassA @ 0          '   Set the module to be operated as 0.

Dim g_bCalEndEvent As Byte                    '   Store the variable for determining the completeness of calibration.
Dim g_iFX As Integer                          '   Save the x-axis magnetic field intensity.
Dim g_iFY As Integer                          '   Save the y-axis magnetic field intensity.
Dim g_wAngle As Word                          '   Save the directional angle measurement.
Dim g_iDevAngle As Integer                    '   Save the deviation angle.

Sub Main()                                    '   Main program
    g_bCalEndEvent = 0                        '   Clear the variable for determining the completeness of calibration.
    Debug CLS
    Debug "Module calibration starts. Please horizontally rotate the module at least 360 degrees. ", CR
    myCompass.Calibration(2)                  '   Set the calibration duration to be 20 seconds.

    Do
    Loop Until g_bCalEndEvent=1               '   Exit after the calibration completion event is detected.

    Debug "Calibration completes.", CR

    myCompass.SetRefreshFreq(4)               '   Set the refresh frequency.
    myCompass.SaveAngle(0, 0)                 '   Save zero degrees in the 0th angle storage block.
    myCompass.SetDevAngleLimit(45)            '   Set the deviation angle limit to be 45 degrees.
    myCompass.SetDevAngleNum(0)               '   Set the angle stored in the 0th storage block to be the base direction.

    myCompass.EnableRefreshEvent()            '   Enable the measurement refresh event.
    myCompass.EnableDevAngleLimitEvent()      '   Enable the deviation angle limit event.

    Do                                        '   Infinite loop
    Loop
```

```
End Sub

Event myCompass.FieldRefreshEvent()                ' Refresh event
    myCompass.GetXYField(g_iFX, g_iFY)             ' Read the x- and y-axis magnetic field intensities
    myCompass.GetAngle(g_wAngle)                   ' Read the current directional angle with respect to the north
    Debug CSRXY(1, 5), "Current directional angle: ", %DEC3 g_wAngle, CR
    Debug CSRXY(1, 6), "The x-axis magnetic field intensity: ", %DEC6 g_iFX, CR
    Debug CSRXY(1, 7), "The y-axis magnetic field intensity: ", %DEC6 g_iFY, CR
End Event

Event myCompass.DevAngleLimitEvent()               ' Deviation angle limit exceeding event
    myCompass.GetDevAngle(0, g_iDevAngle)          ' Read the deviation angle
    Debug CSRXY(1, 10), "The deviation angle: ", %DEC4 g_iDevAngle, CR
End Event

Event myCompass.CalEndEvent()                      ' Calibration completion event
    g_bCalEndEvent = 1
End Event
```

Appendix

1. Known problems:

2. Table for the module numbers and the switches:

	0		8		16		24
	1		9		17		25
	2		10		18		26
	3		11		19		27
	4		12		20		28
	5		13		21		29
	6		14		22		30
	7		15		23		31