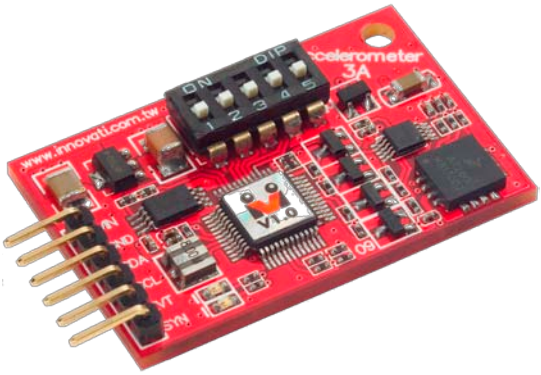
Innovati’s Accelerometer 3A

Three-Axis Acceleration Sensing Module

Version: V1.1



Product Overview: Innovati’s Accelerometer 3A module is a user-friendly, high-precision three-axis acceleration sensing module. With the connection through the cmdBUS to the BASIC Commander, the sensed acceleration values in three axes or the angle between the acceleration orientation and the axes can be retrieved by using a simple command. Furthermore, the software calibration can be performed so as to improve the adaptability to many conditions.

Application:

- Measurement of the static tilt angle for controlling the vehicle to keep balance.
- Measurement of the dynamic acceleration for sensing the magnitude of the force and direction.
- Measurement of the static acceleration for sensing the direction of the gravitational force.

Product Features:

- Digitally measure acceleration values in three axes.
- Provide four precision levels (1.5g, 2g, 4g, and 6g) which can be selected through the software at any time to meet the different measurement demands.
- Measure the angle between the acceleration and the axes in units of degrees.
- Provide the calculated values of the 2-D resultant force and its corresponding angle and arbitrarily select the two axes to be measured.
- Provide the calculated values of the 3-D resultant force and its corresponding angle.
- Provide the notification for the applied force. After the target applied force to be detected is configured, the module can automatically generate the notification event. The user can set the values of the target forces in the X, Y, and Z axes, or the 2-D resultant force separately. Up to 8 notification events for each force value are available for the user to flexibly configure the module.
- Provide the notification for the deviation angle. After the base angle to be compared and the deviation angle in units of degrees are configured, the module will automatically generate the notification event when the measured angle exceeds the deviation angle. The user can configure the 2-D resultant force in any two of the axes and the deviation angle for the notification.
- Provide memory space for storing 256 angle values such as the current angle, or any angle as the base angle for configuring the deviation angle.
- The precision of the measured angle value can be up to 1 degree.
- The detectable range of the acceleration is up to ±6 g.
- The module can automatically store the maximum acceleration value in each axis. The user can quickly read or recalculate the values through the command.
- Five sensing frequencies or refresh frequencies (100 Hz, 50 Hz, 25 Hz, 10 Hz, and 1 Hz)

can be selected and the user can change the refresh frequencies any time through the command.

- Provide notification events for refreshing of sensing values. After the notification is configured through the command, the module will automatically generate the notification event each time the sensing value is refreshed.
- Provide commands for configuring the calibration value. The user can change the calibration value at any time so as to obtain the returned value which meets the demands.
- The user can read the digital voltage value in each axis directly through the command.

Connection: Directly put the ID switch on the required number, and then connect the cmdBUS to the corresponding pins on the BASIC Commander so that the user can perform the required operations through the BASIC Commander.

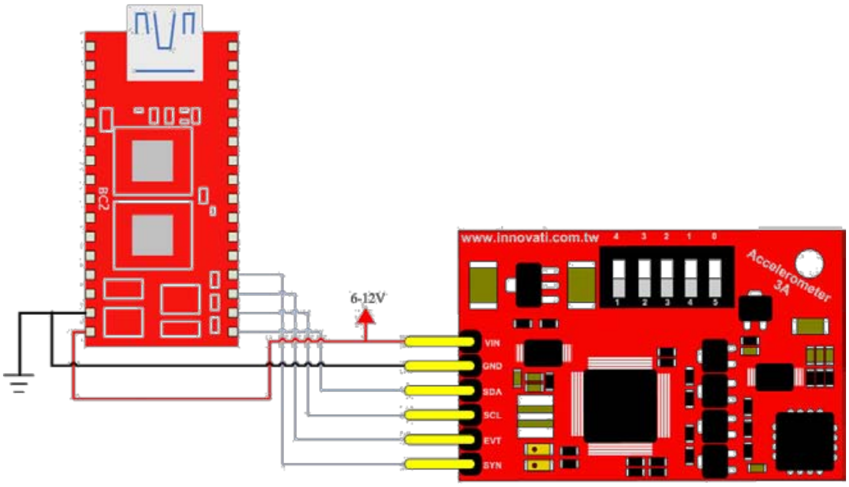
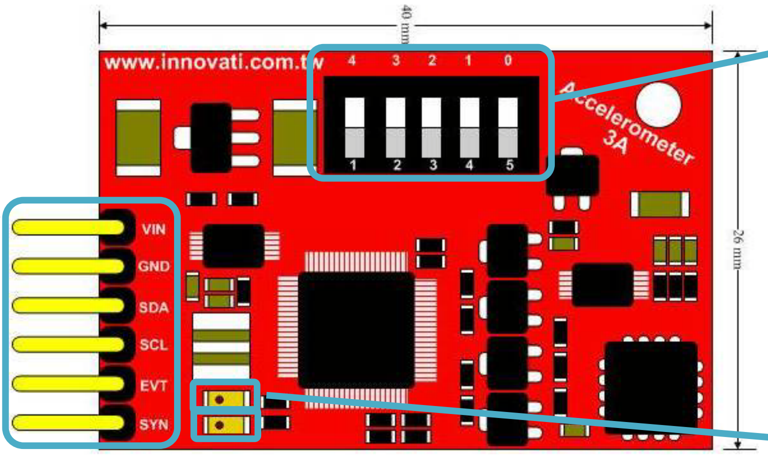


Figure 1　Connection with the BASIC Commander

Product Specifications:

Pins for cmdBUS: Connect these pins to the corresponding pins on the BASIC Commander for controlling the Accelerometer 3A module through the BASIC Commander. While connecting, please notice the pin assignment. Connect Vin to the Vin on the BASIC Commander. Incorrect pin connection may cause damage to the module.)

Module ID Setting Switch: The module ID of the Accelerometer 3A module can be configured with the binary digits from the right to the left. This ID number allows the BASIC Commander to determine the module to be controlled during the operation (Please refer to Appendix 2).



From the top to the bottom:
Green Event Indicator: The blinking light indicates that the module is transmitting an event.
Orange Command Indicator: The blinking light indicates the module and the SBC are receiving data.

Figure 2　Description of pins and switches on the module

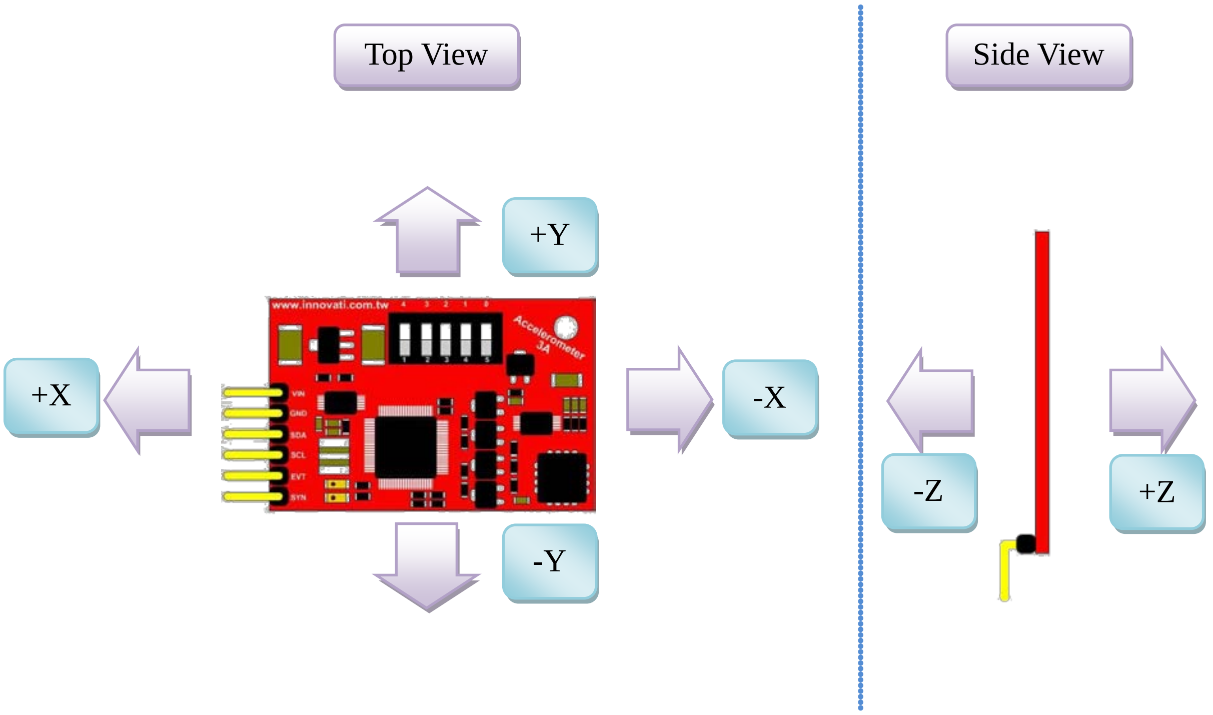


Figure 3 Variation of the dynamic acceleration values in each axis

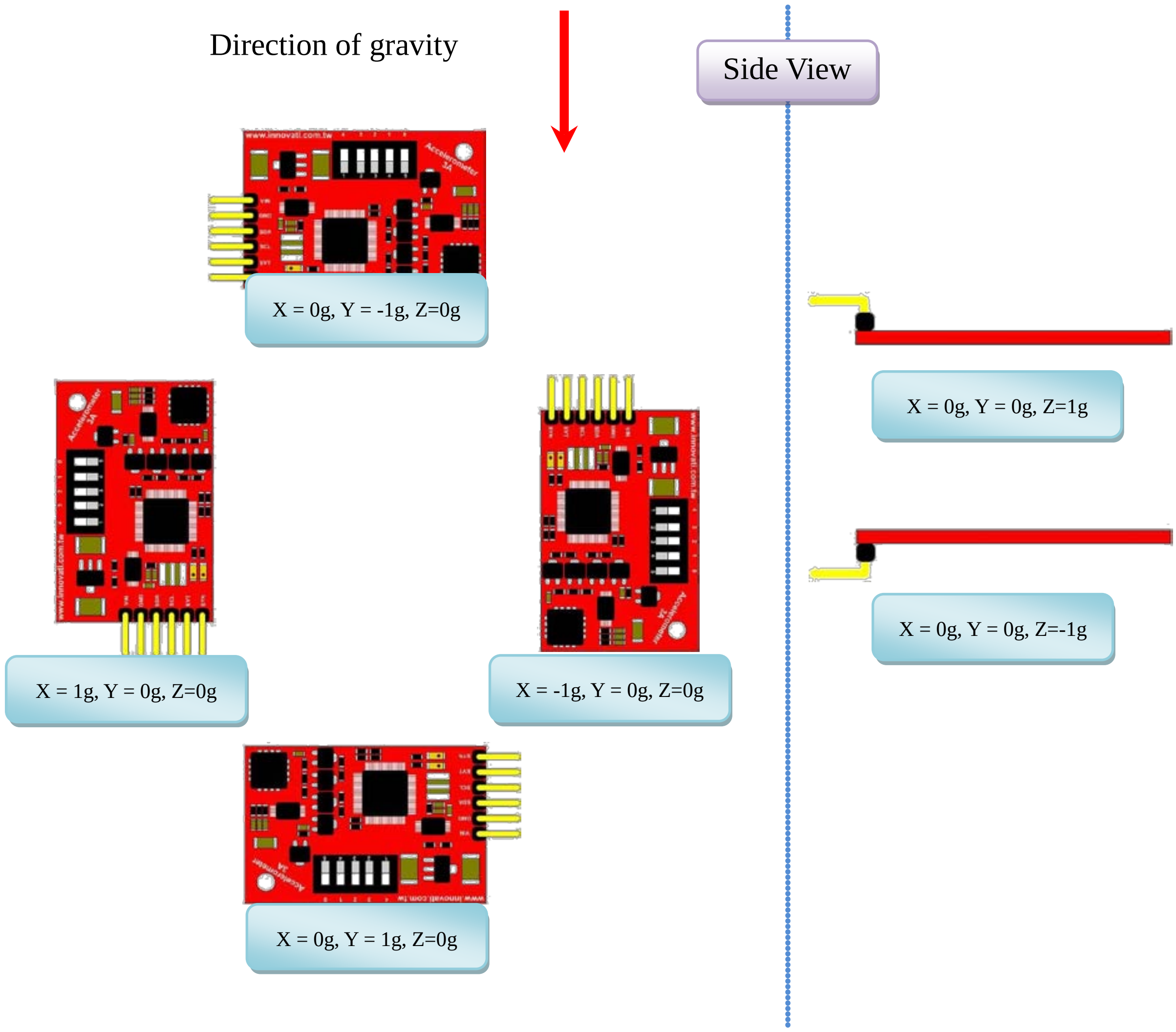


Figure 4 Display static acceleration value in each axis

Precautions for Operations:

- Please place the module as level as possible so as to obtain a better measurement value.

Operating Temperature of the Module: 0 ℃~ 70 ℃
Storage Temperature of the Module: -40 ℃~125℃

List of Commands:

The following list shows various commands dedicated to controlling the Accelerometer 3A module. The command name and parameters which should be input are shown in bold or bold-italic typefaces. The words in bold typeface should not be changed while being input. The words in bold-italic typefaces can be filled with parameters in properly defined format by the user. Please note that the words in uppercase or lowercase are regarded as the same word while entering the program in the innoBASIC Workshop.

Before executing the commands for Accelerometer 3A, please define the corresponding parameters and the module ID at the beginning of the program, for example:

Peripheral *ModuleName* As Accelerometer3A @ *ModuleID*

Command Format	Command Function
Commands for Sensing Acceleration and Angle	
GetXForce(*ForceX*)	Get the force values in X, Y, and Z axes; the value in the X axis is stored in ***ForceX***; the value in the Y axis is stored in ***ForceY***; the value in the Z axis is stored in ***ForceZ***. The retrieved value is an integer in the range of -32768~32767. The unit will be different depending on the mode. Please refer to **SetMode** command for the unit corresponding to each mode setting.
GetYForce(*ForceY*)	
GetZForce(*ForceZ*)	
GetXYZForce(*ForceX, ForceY, ForceZ*)	
GetForce2D(*Force, Angle*)	Get the resultant force value on the 2-D plane and store it in ***Force***. The unit will be different depending on the mode. Please refer to **SetMode** command for the unit corresponding to each mode setting. Meanwhile, the angle between the force and the dominant axis is retrieved and stored in ***Angle*** in units of degrees. The dominant axis setting can be configured by using **SetAxis2D** command. The retrieved value of ***Force*** is an integer in the range of 0~65535. The retrieved value of ***Angle*** is an integer in the range of 0~359.
GetAngle2D(*Angle*)	Get the angle between the resultant force and the dominant axis on the 2-D plane and store it in ***Angle*** in units of degrees. The dominant axis setting can be changed by using **SetAxis2D** command. The retrieved value of ***Angle*** is an integer in the range of 0~359.
GetForce3D(*Force, Angle1, Angle2*)	Get the resultant force value in the 3-D space and store it in ***Force***. The unit will be different depending on the mode. Please refer to **SetMode** command for the unit corresponding to each mode setting.

	Meanwhile, the angle between the component force on the XY plane and the X axis is retrieved and stored in ***Angle1*** in units of degrees. The angle between the force and the Z axis is stored in ***Angle2*** in units of degrees. The retrieved value of ***Force*** is an integer in the range of 0~65535. The retrieved value of ***Angle1*** is an integer in the range of 0~359. The retrieved value of ***Angle2*** is an integer in the range of 0~179.
GetAngle3D(*Angle1*, *Angle2*)	Get the angle between the component force on the XY plane of the resultant force in the 3-D space and the X axis and store it in ***Angle1*** in units of degrees. Get the angle between the force and Z axis and store it in ***Angle2*** in units of degrees. The retrieved value of ***Angle1*** is an integer in the range of 0~359. The retrieved value of ***Angle2*** is an integer in the range of 0~179.
GetXADVal(*Value*)	Get the digitized voltage value of the applied force in each axis (X, Y, and Z axes) and store it in ***Value***. The retrieved value of ***Value*** is an integer in the range of 0~65535.
GetYADVal(*Value*)	
GetZADVal(*Value*)	
Commands for sensing and configuring the 2-D deviation angle	
SaveCurrAngle2D(*Number*)	Store the currently measured 2-D angle in the location specified by ***Number***. The value of ***Number*** can be an integer in the range of 0~255.
SaveAngle2D(*Number*, *Angle*)	Store the input value of ***Angle*** in the location specified by ***Number***. The input value of ***Number*** can be any integer in the range of 0~255. The input value of ***Angle*** can be any integer in the range of 0~359.
LoadAngle2D(*Number*, *Angle*)	Read the 2-D angle value stored in the location specified by ***Number*** and store it in ***Angle***. The value of ***Number*** can be any integer in the range of 0~255. The retrieved value of ***Angle*** is an integer in the range of 0~359.
GetDevAngle2D(*Number*, *Angle*)	Get the deviation angle from the configured base direction. This command will set the angle value stored in the location specified by ***Number*** as the base direction. The input value of ***Number*** should be an integer in the range of 0~255. Then the deviation angle between the currently measured 2-D angle and the base direction will be retrieved and stored in ***Angle*** in units of degrees. If the currently measured direction is within 180 degrees counterclockwise relative to the base direction, the retrieved value is positive. If the currently measured direction is within 179 degrees clockwise relative to the base direction,

	the retrieved value is negative. The retrieved value of ***Angle*** is an integer in the range of -179~180.
SetDevAngleLimit2D(*Angle*)	Set the limit for determining the 2-D deviation angle in units of degrees. The input value of ***Angle*** can be an integer in the range of 0~179. The default value is 5.
GetDevAngleLimit2D(*Angle*)	Get the limit for determining the 2-D deviation angle in units of degrees and store it in ***Angle***. The retrieved value is an integer in the range of 0~179.
SetDevAngleNum2D(*Number*)	Store the 2-D angle value in the location specified by ***Number*** as the base direction for detecting the 2-D deviation angle. The input value of ***Number*** is an integer in the range of 0~255.
GetDevAngleNum2D(*Number*)	Get the location for storing the configured base direction for determining the 2-D deviation angle and store it in ***Number***. The retrieved value of ***Number*** is an integer in the range of 0~255.
EnableDevAngleLimitEvent2D()	Enable the notification event for exceeding the limit of the 2-D deviation angle.
DisableDevAngleLimitEvent2D()	Disable the notification event for exceeding the limit of the 2-D deviation angle.
***Status*=GetDevAngleLimitStatsu2D()**	Get the status of whether the current 2-D angle exceeds the limit of the 2-D deviation angle. If the retrieved value of ***Status*** is 1, it means that the currently detected direction exceeds the limit of the 2-D deviation angle. If the retrieved value of ***Status*** is 0, it means that the detected direction is within the limit of the 2-D deviation angle.
Commands for configuring the notifications for applied forces	
SetXForceLimit(*Number, Limit*)	The input value of ***Number*** is used to specify the parameter group to be configured, which can be any integer in the range of 0~7. The input value of ***Limit*** is used for configuring the limit of the applied force for the notification, which can be any integer in the range of 0~65535. Once the absolute value of the applied force in the corresponding axis (X, Y, or Z axes) exceeds the value, it will be determined as exceeding the limit setting.
SetYForceLimit(*Number, Limit*)	
SetZForceLimit(*Number, Limit*)	
GetXForceLimit(*Number, Limit*)	Use the input value of ***Number*** to get the limit of the applied force in the corresponding axis (X, Y, or Z axes) for the notification event, which can be any integer in the range of 0~7. The configured limit value is stored in ***Limit*** and the retrieved value of ***Limit*** will be an integer in the range of 0~65535.
GetYForceLimit(*Number, Limit*)	
GetZForceLimit(*Number, Limit*)	
***Status*=GetXForceLimitStatus()**	Get the status of the configuration of the force limit in each axis (X, Y, or Z axes) for each group and store it
***Status*=GetYForceLimitStatus()**	

***Status*=GetZForceLimitStatus()**	in ***Status***; each bit is corresponding to the setting of each group, for example: ***Status*** = 1 ➔ It means that the setting of Group 0 reaches the predefined limit of the applied force but Groups 1~7 do not reach the limit of the applied force. ***Status*** = 2 ➔ It means that the setting of Group 1 reaches the predefined limit of the applied force but Groups 2~7 do not reach the limit of the applied force. ***Status*** = 3 ➔ It means that settings of Groups 0 and 1 have reached the limit of applied force.
EnableXForceLimitEvent()	Enable the notification event for the limit of the applied force in the corresponding axis (X, Y, or Z axes).
EnableYForceLimitEvent()	
EnableZForceLimitEvent()	
DisableXForceLimitEvent()	Disable the notification event for the limit of the applied force in the corresponding axis (X, Y, or Z axes).
DisableYForceLimitEvent()	
DisableZForceLimitEvent()	
SetForceLimit2D(*Number, Limit*)	The input value of ***Number*** is used to specify the parameter group to be configured, which can be any integer in the range of 0~7. The input value of ***Limit*** is used for configuring the limit of the applied force for the notification, which can be any integer in the range of 0~65535. Once the absolute value of the applied force on the 2-D plane exceeds the value, it will be determined as exceeding the limit setting.
GetForceLimit2D(*Number, Limit*)	Use the input value of ***Number*** to get the limit of the applied force on the 2-D plane for the notification event, which can be any integer in the range of 0~7. The configured limit value is stored in ***Limit*** and the retrieved value of ***Limit*** will be an integer in the range of 0~65535.
***Status*=GetForceLimit2DStatus()**	Get the status of the configuration of the force limit on the 2-D plane for each group and store it in ***Status***; each bit is corresponding to the setting of each group, for example: ***Status*** = 1 ➔ It means that the setting of Group 0 reaches the predefined limit of the applied force but Groups 1~7 do not reach the limit of the applied force. ***Status*** = 2 ➔ It means that the setting of Group 1 reaches the predefined limit of the applied force but Groups 2~7 do not reach the limit of the applied force. ***Status*** = 3 ➔ It means that settings of Groups 0 and 1 have reached the limit of applied force.
EnableForceLimit2DEvent()	Enable the notification event for the limit of the applied force on the 2-D plane.
DisableForceLimit2DEvent()	Disable the notification event for the limit of the

	applied force on the 2-D plane.
SetForceLimit3D(*Number, Limit*)	The input value of ***Number*** is used to specify the parameter group to be configured, which can be any integer in the range of 0~7. The input value of ***Limit*** is used for configuring the limit of the applied force for the notification, which can be any integer in the range of 0~65535. Once the absolute value of the applied force in the 3-D space exceeds the value, it will be determined as exceeding the limit setting.
GetForceLimit3D(*Number, Limit*)	Use the input value of ***Number*** to get the limit of the applied force in the 3-D space for the notification event, which can be any integer in the range of 0~7. The configured limit value is stored in ***Limit*** and the retrieved value of ***Limit*** will be an integer in the range of 0~65535.
***Status*=GetForceLimit3DStatus()**	Get the status of the configuration of the force limit in the 3-D space for each group and store it in ***Status***; each bit is corresponding to the setting of each group, for example: ***Status*** = **1** ➔ It means that the setting of Group 0 reaches the predefined limit of the applied force but Groups 1~7 do not reach the limit of the applied force. ***Status*** = 2 ➔ It means that the setting of Group 1 reaches the predefined limit of the applied force but Groups 2~7 do not reach the limit of the applied force. ***Status*** = 3 ➔ It means that settings of Groups 0 and 1 have reached the limit of applied force.
EnableForceLimit3DEvent()	Enable the notification event for the limit of the applied force in the 3-D space.
DisableForceLimit3DEvent()	Disable the notification event for the limit of the applied force in the 3-D space.
Commands for retrieving and clearing the maximum applied force	
GetMaxXForce(*Force*)	Get the maximum applied force that has ever occurred in the corresponding axis (X, Y, or Z axes) and store it in ***Force***. The retrieved value of ***Force*** will be an integer in the range of -32768~32767.
GetMaxYForce(*Force*)	
GetMaxZForce(*Force*)	
GetMaxForce2D(*Force, Angle*)	Get the maximum applied force that has ever occurred on the 2-D plane and store it in ***Force***. Meanwhile, get the 2-D angle and store it in ***Angle***. The retrieved value of ***Force*** will be an integer in the range of -32768~32767. The retrieved value of ***Angle*** will be an integer in the range of 0~359.
GetMaxForce3D(*Force, Angle1, Angle2*)	Get the maximum applied force that has ever occurred in the 3-D space and store it in ***Force***. Meanwhile, get the angle between the component force on the XY

	plane and the X axis and store it in ***Angle1***. Get the angle between the force and the Z axis and store it in ***Angle2***. The retrieved value of ***Force*** will be an integer in the range of -32768~32767. The retrieved value of ***Angle1*** will be an integer in the range of 0~359. The retrieved value of ***Angle2*** will be an integer in the range of 0~179.
ClearMaxXForce()	Clear the record of the maximum applied force in the corresponding axis (X, Y, or Z axes).
ClearMaxYForce()	
ClearMaxZForce()	
ClearMaxForce2D()	Clear the record of the maximum applied force on the 2-D plane.
ClearMaxForce3D()	Clear the record of the maximum applied force in the 3-D space.
Commands for Various Settings	
SetMode(*Mode*)	Set the sensitivity for the acceleration sensing according to the value of ***Mode***. The default value is 0. The input value of ***Mode*** can be 0~3: ***Mode*** = 0 ➔ When the measured Force = 800, it is normalized to 1g. This is suitable for measuring the acceleration value within ±1.5 g. ***Mode*** = 1 ➔ When the measured Force = 600, it is normalized to 1g. This is suitable for measuring the acceleration value within ±2 g. ***Mode*** = 2 ➔ When the measured Force = 400, it is normalized to 1g. This is suitable for measuring the acceleration value within ±4 g. ***Mode*** = 3 ➔ When the measured Force = 300, it is normalized to 1g. This is suitable for measuring the acceleration value within ±6 g.
GetMode(*Mode*)	Get the configured sensitivity value. The retrieved value of ***Mode*** will be an integer in the range of 0~3.
SetRefreshFreq(*Rate*)	Set the refresh frequency of the measured values according to the value of ***Rate***. The default value is 1. Five input values of ***Rate*** are configurable: ***Rate*** = 0 ➔ Refresh the angle value every 10 ms (100 Hz) ***Rate*** = 1 ➔ Refresh the angle value every 20 ms (50 Hz) ***Rate*** = 2 ➔ Refresh the angle value every 40 ms (25 Hz) ***Rate*** = 3 ➔ Refresh the angle value every 100 ms (10 Hz) ***Rate*** = 4 ➔ Refresh the angle value every 1000 ms (1 Hz)

GetRefreshFreq(*Rate*)	Get the refresh frequency for the measured value and store it in ***Rate***. The retrieved value of ***Rate*** will be an integer in the range of 0~4. Each value corresponds to the refresh frequency as defined in **SetRefreshFreq**().
***Status*=GetRefreshStatus()**	Read the refresh status. When the retrieved value of ***Status*** is 1, it means that the measured value has been updated and the built-in status value will be set as 0. It will be set as 1 only when the module has refreshed the measured value.
EnableRefreshEvent()	Enable the alarm event for notifying the update of measurement value.
DisableRefreshEvent()	Disable the alarm event for notifying the update of measurement value.
SetAxis2D(*Type*)	The value of ***Type*** is used for specifying the axis for determination on the 2-D plane: The input value of ***Type*** can be in the range of 0~5 which corresponds to the following: 0➔ The X axis is used as the 0-degree axis and the Y axis is used as the 90-degree axis 1➔ The Y axis is used as the 0-degree axis and the X axis is used as the 90-degree axis 2➔ The X axis is used as the 0-degree axis and the Z axis is used as the 90-degree axis 3➔ The Z axis is used as the 0-degree axis and the X axis is used as the 90-degree axis 4➔ The Y axis is used as the 0-degree axis and the Z axis is used as the 90-degree axis 5➔ The Z axis is used as the 0-degree axis and the Y axis is used as the 90-degree axis
GetAxis2D(*Type*)	Get the type for the determination on the 2-D plane and store it in ***Type***
ABConvert(*Angle, Binary*)	Convert the input angle value from the 360-degree scale to the 256-division scale as a ***Binary*** output. The input value of ***Angle*** can be any integer in the range of -32768~32767. The retrieved value of ***Binary*** will be an integer in the range of -23302~23301.
BAConvert(*Binary, Angle*)	Convert the input value of ***Binary*** from the 256-division scale into the 360-degree scale and store the retrieved angle value in ***Angle***. The input value of ***Binary*** can be any integer in the range of -23302~23301. The retrieved value of ***Angle*** will be an integer in the range of -32768~32767.
SaveCalVal(*Mode, X0G, X1G, X-1G, Y0G, Y1G, Y-1G, Z0G, Z1G, Z-1G*)	According to the value of ***Mode***, the calibration value for each mode can be stored. The input value of ***Mode*** should be an integer in the range of 0~3. The input

	values of ***X0G***, ***X1G***, and ***X-1G*** should be the voltage values obtained by using the command **GetXADVal** when the applied forces of 0g, 1g and -1g are sensed in the X axis, respectively. The input values of ***Y0G***, ***Y1G***, and ***Y-1G*** should be the voltage values obtained by using the command **GetYADVal** when the applied forces of 0g, 1g and -1g are sensed in the Y axis, respectively. The input values of ***Z0G***, ***Z1G*** , and ***Z-1G*** should be the voltage values obtained by using the command **GetZADVal** when the applied forces of 0g, 1g and -1g are sensed in the Z axis, respectively. The input values other than ***Mode*** should be integers in the range of 0~65535. ※
LoadCalVal(*Mode, X0G, X1G, X-1G, Y0G, Y1G, Y-1G, Z0G, Z1G, Z-1G*)	According to the value of ***Mode***, the calibration value for each mode can be retrieved. The input value of ***Mode*** should be an integer in the range of 0~3. The retrieved values of ***X0G***, ***X1G***, and ***X-1G*** will be the calibration voltage values for the applied forces of 0g, 1g and -1g in the X axis, respectively. The retrieved values of ***Y0G***, ***Y1G***, and ***Y-1G*** will be the calibration voltage values for the applied forces of 0g, 1g and -1g in the Y axis, respectively. The retrieved values of ***Z0G***, ***Z1G*** , and ***Z-1G*** will be the calibration voltage values for the applied forces of 0g, 1g and -1g in the Z axis, respectively. The retrieved values will be integers in the range of 0~65535.
RestoreCalVal()	Restore the default calibration values to replace the current calibration values

※While configuring the calibration values, please measure the digital voltage values under three conditions (0g, 1g, and -1g) in each axis according to Figure 4. Because the voltage value may have perturbations, it is recommended to get 10 readings and take the average of the readings except the maximum and the minimum readings. During the measurement, please fix the module properly to prevent numerical error due to vibration.

Application Events Provided by the Module:

Event	Activation Condition
XForceLimitEvent	After the command **EnableXForceLimitEvent**() is executed, this event will be activated when the sensed applied force in the X axis exceeds the preset limit.
YForceLimitEvent	After the command **EnableYForceLimitEvent**() is executed, this event will be activated when the sensed applied force in the Y axis exceeds the preset limit.
ZForceLimitEvent	After the command **EnableZForceLimitEvent**() is executed, this event will be activated when the sensed applied force in the Z axis exceeds the preset limit.

ForceLimitEvent2D	After the command **EnableForceLimit2DEvent**() is executed, this event will be activated when the sensed applied force on the 2-D plane exceeds the preset limit.
ForceLimitEvent3D	After the command **EnableForceLimit3DEvent**() is executed, this event will be activated when the sensed applied force in the 3-D space exceeds the preset limit.
DevAngleLimitEvent2D	After the command **EnableDevAngleLimitEvent2D**() is executed, this event will be activated when the difference between the sensed direction and the preset base direction exceeds the preset angle value by using the command **SetDevAngleLimit2D**().The base direction can be configured by using the command **SetDevAngleNum2D**().
ForceRefershEvent	After the command **EnableRefreshEvent()** is activated, once the measurement value is updated, the corresponding event will be activated.

Demonstration Program:

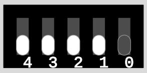
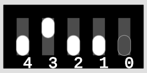
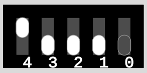
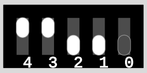
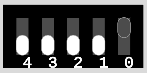
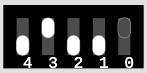
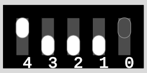
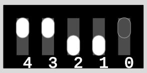
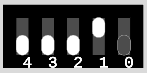
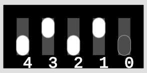
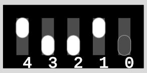
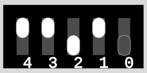
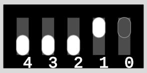
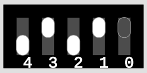
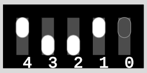
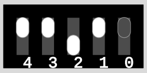
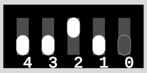
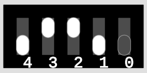
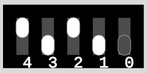
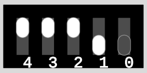
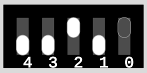
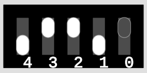
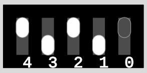
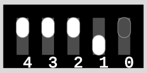
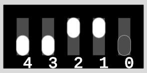
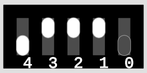
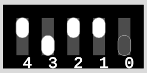
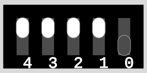
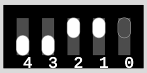
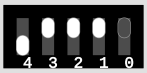
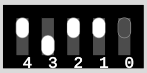
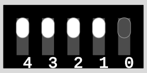
```
Peripheral myG As Accelerometer3A @ 0              '    Set the module ID as 0

Dim g_bStatus As Byte                              '    Store the retrieved status values
Dim g_wADx, g_wADy, g_wADz As Word                 '    Store the voltage values in the X, Y, and Z axes
Dim g_iFx, g_iFy, g_iFz As Integer                 '    Store the force values in the X, Y, and Z axes
Dim g_wF2D As Word                                 '    Store the force values on the 2-D plane
Dim g_wAngle As Word                               '    Store the angle of the force on the 2-D plane
Dim g_wF3D as Word                                 '    Store the force value in the 3-D space
Dim g_wAngle1 as Word                              '    Store the angle between the force and the XY plane in
                                                   '    the 3-D space
Dim g_bAngle2 as Byte                              '    Store the angle between the force and the Z axis in the
                                                   '    the 3-D space
Dim g_bLimit as Byte                               '    Store the sensing values for the event

Sub Main()
    Dim i as Byte                                  '    Store the loop count parameter

    Pause 1000
    myG.SetMode(0)                                 '    Set the sensitivity mode as 0 for measuring the
                                                   '    acceleration value within +/- 1.5g
    myG.SetRefreshFreq(3)                          '    Set the refresh frequency as 10 times per second
    myG.SetAxis2D(0)                               '    Set the XY plane as the 2-D plane
    Debug CLS
    Pause 1000

'   The For Loop is executed to obtain the voltage values in each axis 10 times
    For i=1 To 10
'       DoLoop is used to ensure obtaining the updated values
        Do
```

```
            g_bStatus = myG.GetRefreshStatus()              '   Get the refresh status
        Loop Until g_bStatus=1

        myG.GetXADVal(g_wADx)                               '   Get the voltage value corresponding to the acceleration
                                                            '   in the X axis
        myG.GetYADVal(g_wADy)                               '   Get the voltage value corresponding to the acceleration
                                                            '   in the Y axis
        myG.GetZADVal(g_wADz)                               '   Get the voltage value corresponding to the acceleration
                                                            '   in the Z axis
        Debug "X: ", g_wADx, ", Y: ", g_wADy, ", Z: ", g_wADz, CR
    Next

    myG.GetXYZForce(g_iFx, g_iFy, g_iFz)                    '   Get the acceleration values in the X, Y, and Z axes
    myG.GetForce2D(g_wF2D, g_wAngle)                        '   Get the force value and angel on the XY plane
    myG.GetForce3D(g_wF3D, g_wAngle1, g_bAngle2)            '   Get the force value and angel on the XY plane

    Debug "Force X: ", g_iFx, ", Y: ", g_iFy, ", Z: ", g_iFz, CR
    Debug "2D Force: ", g_wF2D, ", Angle: ", g_wAngle, CR
    Debug "3D Force: ", g_wF3D, ", Angle1: ", g_wAngle1, ", Angle2: ", g_bAngle2, CR

    myG.SetForceLimit2D(0, 800)                             '   Set the limit of the applied force on the 2-D plane as
                                                            '   800(1g)
    g_bLimit = 0
    myG.EnableForceLimit2DEvent()                           '   Enable the notification event for the limit of the applied
                                                            '   force on the 2-D plane.

'   Execute the DoLoop till the detected 2-D applied force exceeds the limit (1g)
    Do
    Loop Until g_bLimit=1

    Debug "Finish"
End Sub

Event myG.ForceLimitEvent2D()
    myG.GetForce2D(g_wF2D, g_wAngle)                        '   Get the force value and angel on the XY plane
    Debug "Event Force: ", g_wF2D, CR
    g_bLimit = 1
End Event
```

Appendix

1. Known problems:
    - v1.0: When the direction setting is configured with a value other than the XY plane, the command **DevAngleLimitEvent2D** will still use the XY plane as the base directions for generating the event.

2. List of the Configuration of the Module ID Switch:

	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