Innovati Joystick 2A Module
2-Dimensional Joystick and Joystick Button

Version: V1.1

Product Overview: Innovati's Joystick 2A module provides simple setup and position reading functions. Users can easily customize Joystick 2A module to fit their own needs. By connecting the Joystick 2A module to Innovati's Basic Commander through cmdBUS, users can run various customized functions and achieve intuitive control of devices such as robots, robotic arms, etc. Joystick 2A provides two kinds of coordinate systems: rectangular and polar. Depending on their applications, users may switch to any one of the two coordinate systems. In addition to x- and y-axis movement control, Joystick 2A also has one button for complex applications. Please use "**Joystick2A**" as the module object name in program.

Applications:

- Robotic arm control: controls the rotation angle of the robotic arm with a polar coordinate output.
- Remote control of cars and aircraft with the use of wireless output module.
- The operation of various testing equipment
- Motor control: controls motor acceleration with the use of a motor module and motor fixed-speed cruising with the use of buttons.
- Control of all Innovati's application packages

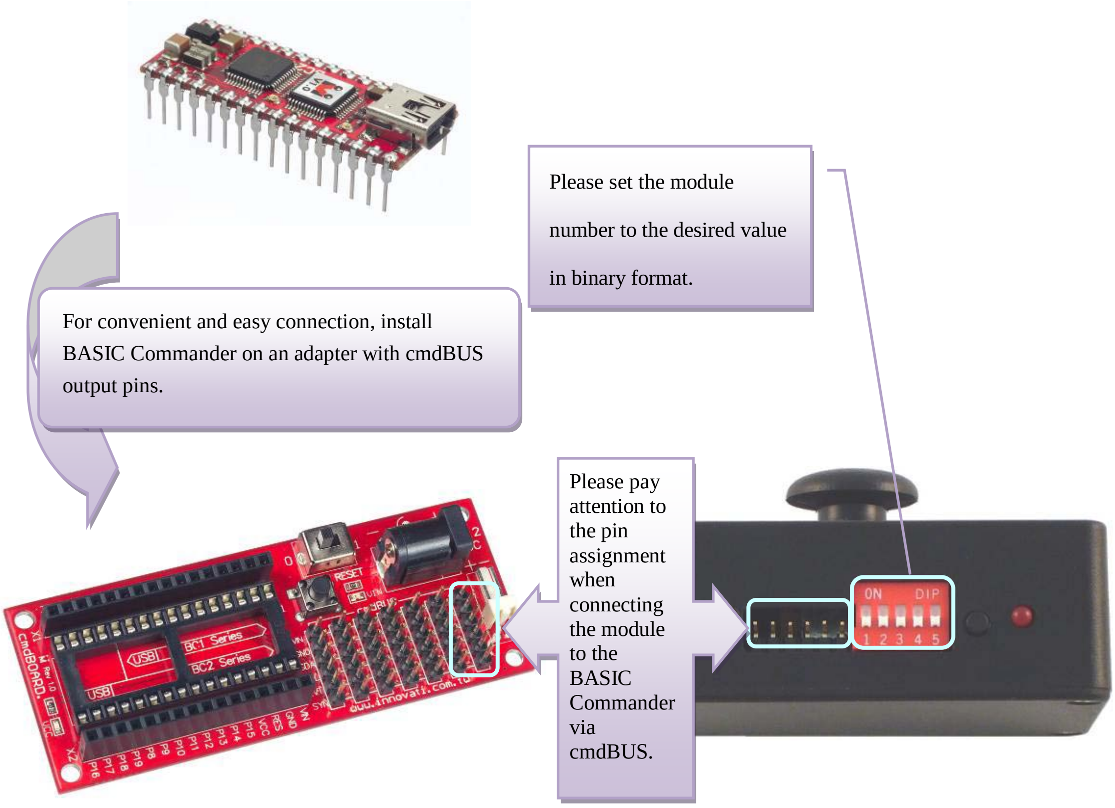
Product Features:

- Easy setup: To use the dedicated instructions for various applications, all that is needed is to connect Joystick 2A module to BASIC commander through cmdBUS.
- 2-dimensional movement.
- Two return coordinate systems: Joystick 2A provides return values in two coordinate systems: rectangular and polar. Users can choose either coordinate

system at any time or use both simultaneously.

- 4-way and 8-way joystick positions: for fast and intuitive control of various basic control applications
- Large origin range: 0~10% customized origin range, preventing joystick bouncing.
- Return value limit: Users can set limits for all return coordinates and thus limit the joystick operational range.
- Customized resolution: The return coordinate can have at most 128 levels and the polar angle has at most 360 levels. Depending on actual applications, the two resolutions may be set individually.
- The joystick includes a joystick button that can serve as a button; pushing down the stick activates the button function. The customized button functions, including the activation time of keystroke repetition and repetition rates, can be specified through simple instructions.
- Easy Calibration: Joystick 2A module comes with a calibration button. The calibration of the joystick can be done at any time during the operation.

Connection: To access Joystick 2A through the BASIC Commander, set the ID switches to the desired number settings and connect cmdBUS to the proper pins on the BASIC Commander.



Please set the module number to the desired value in binary format.

For convenient and easy connection, install BASIC Commander on an adapter with cmdBUS output pins.

Please pay attention to the pin assignment when connecting the module to the BASIC Commander via cmdBUS.

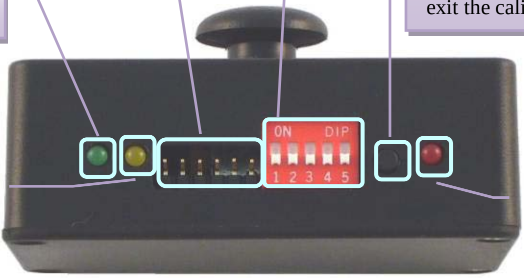
Product Specifications:

Module number switches: Sets the module ID in binary form. The leftmost switch No. 1 is the most significant bit of the module number. The top position is 1 while the bottom position 0. The module number shown in the figure is 0.

cmdBUS pins: Please use cmdBUS to connect the BASIC Commander. Please mind the pin assignment when connecting. These pins are, from left to right, Vin, GND, SDA, SCL, EVT, and SYN.

Calibration activation button: Pressing down and holding this button for 5 seconds activates the calibration mode. Please note that all instructions are ineffective during calibration. Upon completion of calibration, please press the button on the top of the stick to save the calibration results and exit the calibration mode. To exit without saving, please push again the calibration button. The module will exit the calibration mode immediately.

EVT light: The switched on green LED represents an event sent by the module.



SDA communication light: The switched on yellow LED represents the module sending data requests to

Calibration activation light: The flickering red LED represents the module in the calibration mode.

Figure 1: Pin assignment and switch description

After entering the calibration mode (when the red LED starts to blink), please push the stick all the way up and move the stick against the edges for 2 turns to get the maximum and minimum displacements of the x- and y-axes. Finally, move the stick back to the center and let it stand for 3 seconds so that the joystick can record the center of the x- and y-axes. Push down the stick to exit the calibration mode.

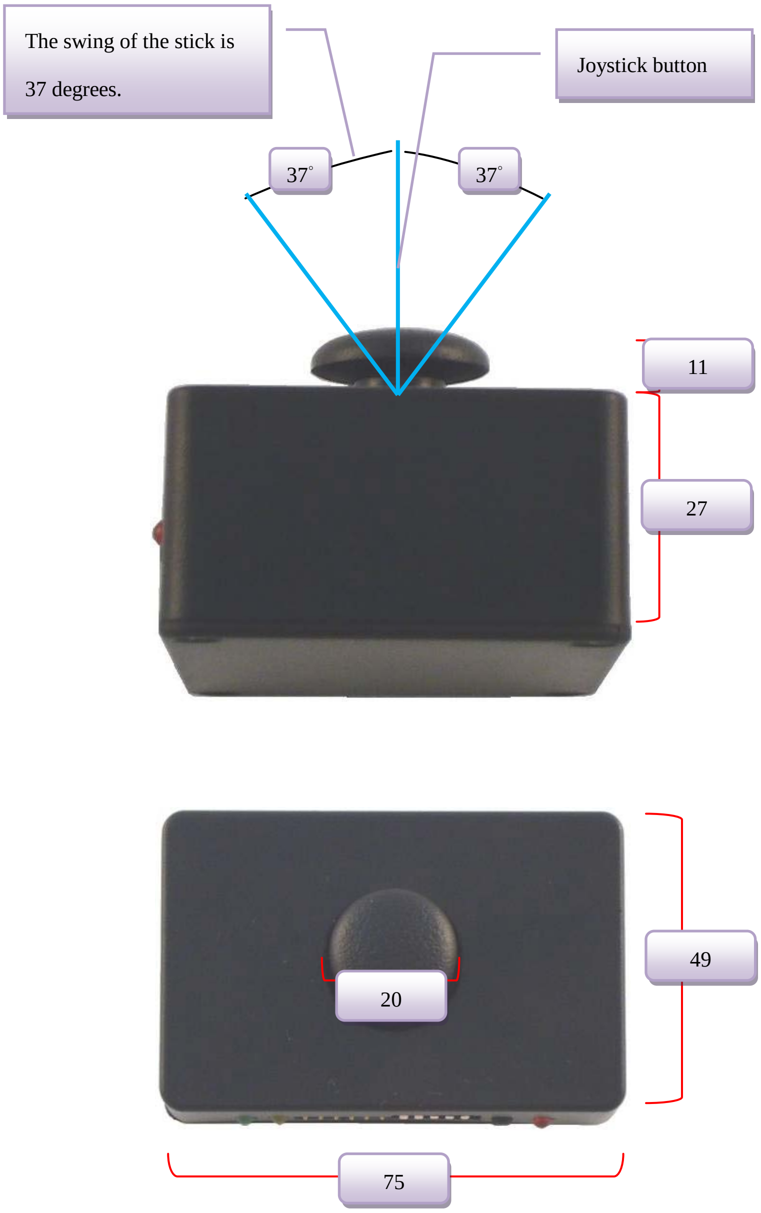
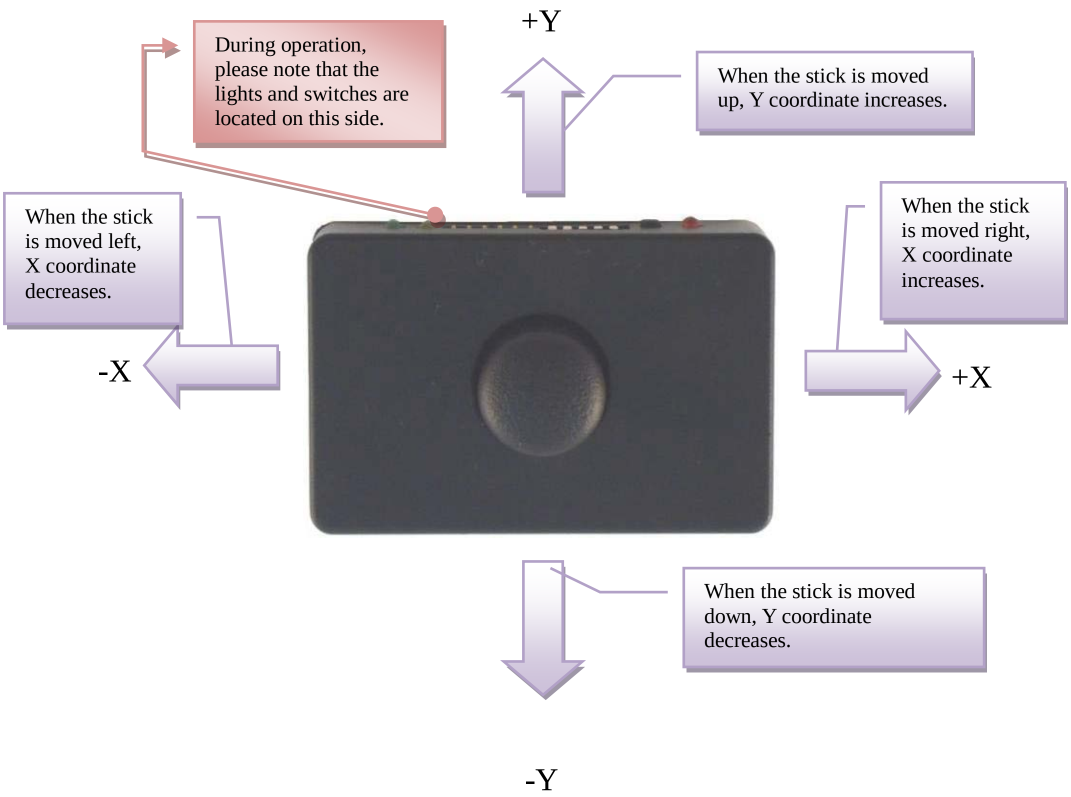


Figure 2: Joystick specifications (Unit: mm)



During operation, please note that the lights and switches are located on this side.

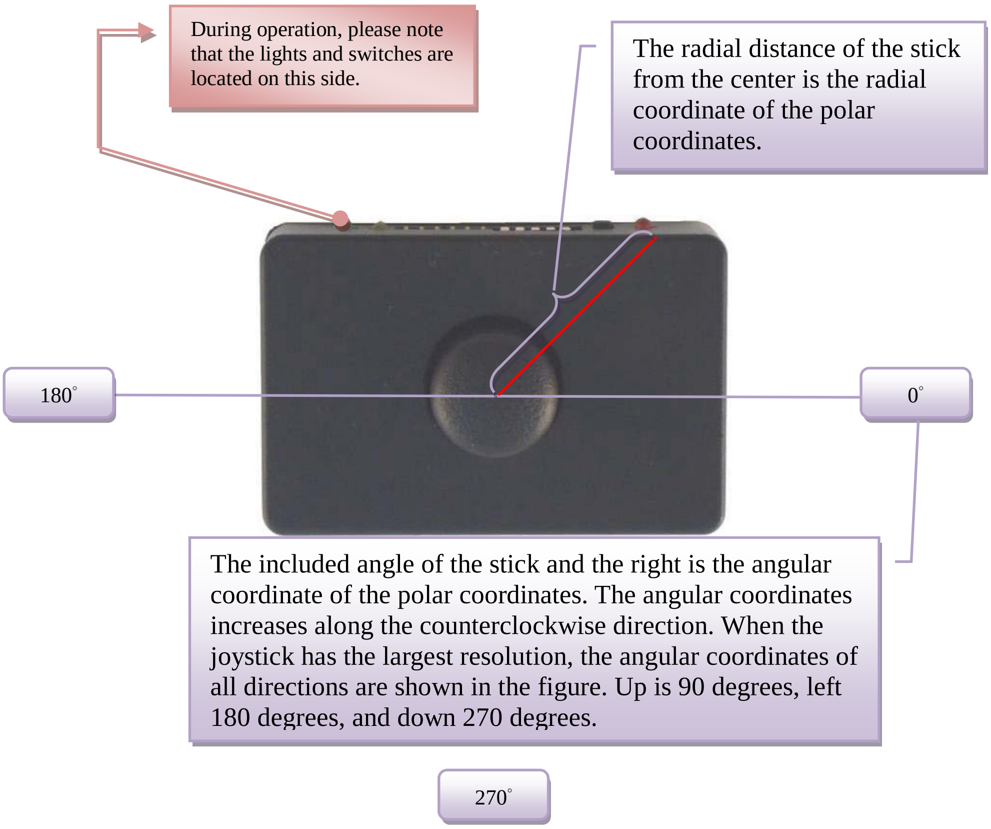
When the stick is moved up, Y coordinate increases.

When the stick is moved left, X coordinate decreases.

When the stick is moved right, X coordinate increases.

When the stick is moved down, Y coordinate decreases.

Figure 3: Joystick movements and the rectangular coordinate system



During operation, please note that the lights and switches are located on this side.

The radial distance of the stick from the center is the radial coordinate of the polar coordinates.

The included angle of the stick and the right is the angular coordinate of the polar coordinates. The angular coordinates increases along the counterclockwise direction. When the joystick has the largest resolution, the angular coordinates of all directions are shown in the figure. Up is 90 degrees, left 180 degrees, and down 270 degrees.

Figure 4: Joystick movements and the polar coordinate system

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{IN}	Conditions				
I_{IN}	Operating Current	7.5	—	—	8	—	mA

Table 1: Rated operating current (at 25℃)

Absolute Maximum Ratings:

Operating temperature: 0°C~ 70°C

Storage temperature: 0°C ~ 70°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 joystick calibration	
GetCalibrationX(*Xmin*, *Xcen*, *Xmax*)	Read the joystick x-axis calibration values. The three return values are digitized voltage values. ***Xmin*** is the minimal x coordinate, ***Xcen*** is the center x coordinate, and ***Xmax*** is the maximal x coordinate. ***Xmin, Xcen*** and ***Xmax*** ranging from 0 to 65535.
GetCalibrationY(*Ymin*, *Ycen*, *Ymax*)	Read the joystick y-axis calibration values. The three return values are digitized voltage values. ***Ymin*** is the minimal y coordinate, ***Ycen*** is the center x coordinate, and ***Ymax*** is the maximal y coordinate. ***Ymin, Ycen*** and ***Ymax*** ranging from 0 to 65535.
SetCalibrationX(*Xmin*, *Xcen*,	Set the x-axis joystick calibration values. Three arguments

Xmax**)**	are required: ***Xmin***, the minimum joystick calibration value, ***Xcen***, the joystick center calibration value, and ***Xmax*** the maximum calibration value. When manually calibrating the joystick, please pay attention to the input order: please input the minimum value in ***Xmin*** and the maximum value in ***Xmax***. ***Xmin, Xcen*** and ***Xmax*** ranging from 0 to 65535.
SetCalibrationY(*Ymin*, *Ycen*, *Ymax*)	Set the y-axis joystick calibration values. Three arguments are required: ***Ymin***, the minimum joystick calibration value, ***Ycen***, the joystick center calibration value, and ***Ymax*** the maximum calibration value. When manually calibrating the joystick, please pay attention to the input order: please input the minimum value in ***Ymin*** and the maximum value in ***Ymax***. ***Ymin, Ycen*** and ***Ymax*** ranging from 0 to 65535.
StartCalibration()	Activate the joystick calibration mode. After executing this function, the joystick enters the calibration mode. At this moment, please push the stick all the way up and move the stick against the edges for 2 turns to get the maximum and minimum displacements of the x- and y-axes. Finally, move the stick back to the center and let it stand for 3 seconds so that the joystick can record the center of the x- and y-axes. Push down the stick to exit the calibration mode.
Commands for reading the joystick coordinates	
GetXY(*X*, *Y*)	Read the current x and y coordinates of the joystick in the rectangular coordinate system. ***X*** is the x coordinate, and ***Y*** the y coordinate. Both have a default range from -127~127. **SetXYRes**() changes the ***X*** and ***Y*** ranges.
GetPolarBinaryRadian(*Radius*, *Angle*)	Read the integer polar coordinate of the joystick. The return value ***Radius*** is the radial coordinate, and ***Angle*** the angular coordinate with the positive x-axis being zero degrees. ***Angle*** increases along the counterclockwise direction. By default, ***Radius*** ranges from 0 to 127 and **SetRadiusRes**() changes its range. By default, ***Angle*** ranges from 0 to 127 and **SetRadianRes**() changes its range
GetPolarRadian(*Radius*, *Radian*)	Read the floating-point polar coordinate of the joystick. The first return value, ***Radius***, is the floating-point radial coordinate, ranging from 0 to 1. The second return value ***Radian*** is the floating-point angular coordinate in unit of radian, ranging from 0~6.37.

***Dir* = Get4WayStatus()**	Read the 4-way joystick position. The return value ***Dir*** is a directional value, ranging from 0~4. They represent: 0 ➔ The joystick is in the center 1 ➔ The joystick is on the right. 2 ➔ The joystick is at the bottom 3 ➔ The joystick is on the left. 4 ➔ The joystick is on the top.
***Dir* = Get8WayStatus()**	Read the 8-way joystick position. The return value ***Dir*** is a directional value, ranging from 0~8. They represent: 0 ➔ The joystick is in the center 1 ➔ The joystick is on the right. 2 ➔ The joystick is on the bottom right. 3 ➔ The joystick is at the bottom. 4 ➔ The joystick is on the bottom left. 5 ➔ The joystick is on the left. 6 ➔ The joystick is on the top left. 7 ➔ The joystick is on the top. 8 ➔ The joystick is on the top right.
Commands for reading joystick range settings	
GetStickDeadZone(*DZx*, *DZy*)	Read the origin dead zones of the x-axis and the y-axis. The return values, ***DZx*** and ***DZy***, are integers, ranging from 0 to 10 in unit of %. The system treats as the origin those return rectangular coordinates that fall within these ranges.
GetRadiusDeadZone(*DZr*)	Read the origin dead zone of the radial axis. The return value, ***DZr***, is one integer, ranging from 0 to 10 in unit of %. The system treats as the origin those return radial polar coordinates that fall within this range.
GetXYSaturation(*SATx*, *SATy*)	Read the saturation limits of the x- and y- axes. The return values, ***SATx*** and ***SATy***, are integers, ranging from 60 to 100 in unit of %. The system returns the maximum or minimum values when the current rectangular coordinates, positive or negative, exceed these limits. The system only calculates division values with the full scale being the saturation limit.
GetRadiusSaturation(*SATr*)	Read the saturation limit of the radial axis. The return value, ***SATr***, is one integer, ranging from 60 to 100 in unit of %. The system returns the maximum or minimum values when the current radial polar coordinate, positive or negative,

	exceeds this limit. The system only calculates division values with the full scale being the saturation limit.
GetXYRes(*RESx*, *RESy*)	Read the resolution settings of the x- and y-axes. The return values, ***RESx*** and ***RESy***, are integers, ranging from 0 to 128. Within the distinguishable ranges, the system quantizes the rectangular coordinates by the scale of the resolution settings. Because the index scale includes zero, a resolution of 128 means the positive scale is divided into 128 divisions from 0 to 127. Likewise, the negative scale is also divided into 128 divisions, from 0 to -127.
GetRadiusRes(*RESr*)	Read the resolution setting of the radial axis. The return value, ***RESr*** is one integer, ranging from 0 to 128. Within the distinguishable ranges, the system quantizes the radial polar coordinate by the scale of the resolution setting. Because the index scale includes zero, a resolution of 128 means the positive scale is divided into 128 divisions from 0 to 127.
GetRadianRes(*RESa*)	Read the resolution setting of the polar angle. The return value, ***RESa*** is one integer, ranging from 0 to 360. The system quantizes the polar angle by the scale of the resolution setting. Because the index scale includes zero, a resolution of 360 means the scale is divided into 360 divisions from 0 to 359.
Commands for customizing joystick ranges	
RestoreSettings()	Restore all joystick settings to the default values.
SetStickDeadZone(*DZx*, *DZy*)	Set the origin dead zones of the x-axis and the y-axis. The two arguments, ***DZx and DZy, are*** the x-axis and the y-axis settings, respectively. Both range from 0 to 10 in unit of %. The system treats as the origin those rectangular coordinates of joystick positions that fall within these ranges.
SetRadiusDeadZone(*DZr*)	Set the origin dead zone of the radial axis in the polar coordinate system. The argument ***DZr,*** is one integer, ranging from 0 to 10 in unit of %. The system treats as the origin those radial coordinates of joystick positions that fall within this range.
SetXYSaturation(*SATx*, *SATy*)	Set the saturation limits of the x- and y- axes in the rectangular coordinate systems. The two arguments, ***SATx*** and ***SATy***, are integers, ranging from 60 to 100 in unit of %.

	After executing the function, the system returns the maximum or minimum values when the current rectangular coordinates, positive or negative, exceed these limits. The system only calculates division values with the full scale being the saturation limit.
SetRadiusSaturation(*SATr*)	Set the saturation limit of the radial axis in the polar coordinate system. The argument, ***SATr***, is one integer, ranging from 60 to 100 in unit of %. After executing the function, the system returns the maximum or minimum values when the current radial polar coordinate, positive or negative, exceeds this limit. The system only calculates division values with the full scale being the saturation limit.
SetXYRes(*RESx*, *RESy*)	Set the resolutions of the x- and y-axes. The arguments, ***RESx*** and ***RESy***, are integers, ranging from 0 to 128. Within the distinguishable ranges, the system quantizes the rectangular coordinates by the scale of the resolution settings. Because the index scale includes zero, a resolution of 128 means the positive scale is divided into 128 divisions from 0 to 127. Likewise, the negative scale is also divided into 128 divisions, from 0 to -127. Please not that, although the resolution can be 0 or 1, the return x and y coordinates are both zero with 0 or 1 resolution settings.
SetRadiusRes(*RESr*)	Set the resolution setting of the radial axis. The argument, ***RESr***, is one integer, ranging from 0 to 128. Within the distinguishable ranges, the system quantizes the z coordinate by the scale of the resolution setting. Because the index scale includes zero, a resolution of 128 means the scale is divided into 128 divisions from 0 to 127. Please not that, although the resolution can be 0 or 1, the return radial coordinate is zero with 0 or 1 resolution settings.
SetRadianRes(*RESa*)	Set the resolution setting of the polar angle. The argument, ***RESa***, is one integer, ranging from 0 to 360. The system quantizes the polar angle by the scale of the resolution setting. Because the index scale includes zero, a resolution of 360 means the scale is divided into 360 divisions from 0 to 359. Please not that, although the resolution can be 0 or 1, the return polar angle is zero with 0 or 1 resolution settings.

Commands for button operations	
***Sta* = GetButtonStatus()**	Read the button status and saves it in ***Sta***. The return values are described below: When the keystroke repetition is disabled: 0: The button is pressed. 1: The button is not pressed. When the keystroke repetition is enabled: 0: No new button event is detected. 1: The button is just pressed down or the button is pressed and held for a time equal to or longer than Repeat Time. The value will be set as 1 repeatedly after an interval of Repeat Rate. This value remains 1 until **GetButtonStatus** is executed.
ButtonRepeatFunc(*Rep*)	Enable and disables the automatic keystroke repetition of the joystick button. The argument, ***Rep***, is one- integer with input values of 0 and 1 only. 0: Disable the automatic keystroke repetition. Only one **ButtonEvent** is activated when the joystick button is pressed. The return joystick button status remains 1 (pressed) when the joystick button is pressed and held. 1: Enable the automatic keystroke repetition. When the joystick button is pushed down, **ButtonEvents** are activated repeatedly according to repeat time and repeat rate until the button is released.
SetRepeatTime(*Time*)	Set the activation interval of automatic keystroke repetition. This value ranges from 0 to 255, in unit of 10ms. After setting, the system returns a **ButtonEvent** when the button is pushed and held longer than this setting and starts repeatedly returning **ButtonEvents** according to Repeat Rate. Please note that **ButtonRepeatFunc**() has to be executed first to enable the automatic keystroke repetition. Besides, zero ***Time*** does not disable the automatic keystroke repetition. Instead, the system immediately starts return **ButtonEvents** repeatedly according to repeat rate.
GetRepeatTime(*Time*)	Read the activation interval of automatic keystroke repetition. This return value ranges from 0 to 255.
SetRepeatRate(*Rate*)	Set the repetition interval of automatic keystroke repetition. This value ranges from 0 to 255, in unit of 10ms. After

	setting, the system returns **ButtonEvents** repeatedly every interval of repeat rate when the button is pushed and held longer than repeat time. Please note that **ButtonRepeatFunc**() has to be executed first to enable the automatic keystroke repetition. With zero repeat rate, the system does not repeatedly return **ButtonEvents** after repeat time.
GetRepeatRate(*Rate*)	Read the repetition interval of automatic keystroke repetition. This return value ranges from 0 to 255.
Commands for enabling and disabling events	
EnableStickEvent()	Enable **StickEvent**
DisableStickEvent()	Disable **StickEvent**
Enable4WayEvent()	Enable **Chang4WayEvent**
Disable4WayEvent()	Disable **Chang4WayEvent**
Enable8WayEvent()	Enable **Chang8WayEvent**
Disable8WayEvent()	Disable **Chang8WayEvent**
EnableButtonEvent()	Enable **ButtonEvent**
DisableButtonEvent()	Disable **ButtonEvent**
EnableCalEndEvent()	Enable **CalEndEvent**
DisableCalEndEvent()	Disable **CalEndEvent**
SetEventRefreshRate(*Rate*)	Set the event refresh rate. This value ranges from 0 to 255 in unit of 10ms. After setting, two successive events will be activated in the intervals set by this function. Please note that zero refresh rates are equivalent to the refresh rate of 1.
GetEventRefreshRate(*Rate*)	Read the event refresh rate. This return value ranges from 0 to 255.

Event	Activating Conditions
StickEvent	This event is activated when joystick movement is detected.
Change4WayEvent	This event is activated when the 4-way value of the joystick changes.
Change8WayEvent	This event is activated when the 8-way value of the joystick changes.
ButtonEvent	When the automatic keystroke repetition is disabled: This event is activated when the joystick button is pushed. When the automatic keystroke repetition is enabled: This event is activated when the joystick button is pressed after an internal of Repeat Time and, every interval of Repeat Rate.
CalEndEvent	This event is activated when the joystick calibration completes.

Exaple Program:

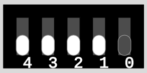
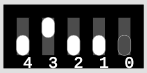
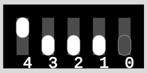
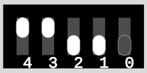
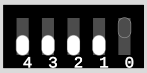
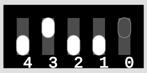
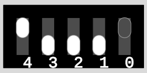
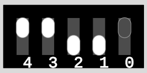
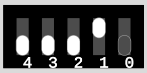
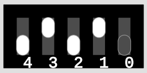
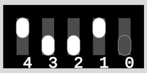
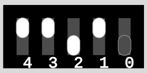
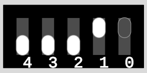
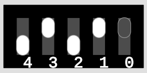
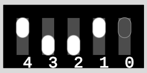
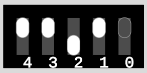
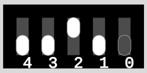
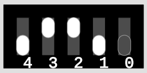
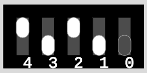
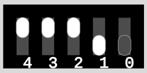
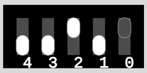
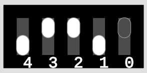
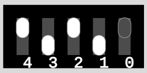
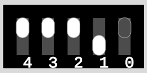
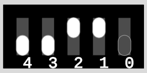
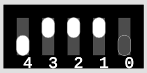
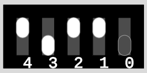
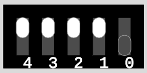
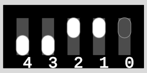
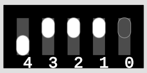
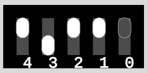
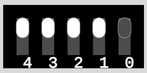
(Detects joystick movements, and returns its x and y coordinates when movements are detected.)

```
Peripheral myJoy As Joystick2A @ 0  ‘       Set the module to be operated as 0.

Sub Main()
        myJoy.SetStickDeadZone(2, 2)         ‘       Set the x- and y-axis origin dead zones.
        myJoy.SetXYSaturation(80, 80)        ‘       Set the x- and y-axis saturation limits.
        myJoy.SetXYRes(128, 128)             ‘       Set the x- and y-axis resolutions.

        myJoy.SetEventRefreshRate(1)         ‘       Set the event refresh rate.
        myJoy.EnableStickEvent()             ‘       Enable StickEvent.

        Do
        Loop
End Sub

Event myJoy.StickEvent()
        Dim sX, sY As Short

        myJoy.GetXY(sX, sY)
        Debug CSRXY(1, 1),  “X:  “, %DEC4R sX,  “, Y:  “, %DEC4R sY, CR
End Event
```

Appendix

1. Known problem:
   - At version 1.0, command “RestoreSettings” will only change the values in RAM. After turn off the power and turn on again, the settings will become the last set values.

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