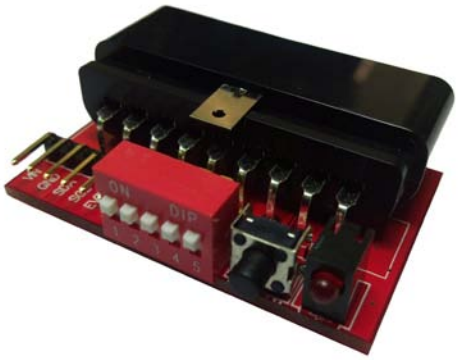
Innovati’s Gamepad Ps

Ps2 Gamepad Control Module

Version: V1.0



Product overview:

Innovati’s GamepadPs module provides simple settings and position obtaining commands with 12 buttons, enabling the user to plan his/her desired operating modes. By connecting cmdBUS and BASIC Commander, you can use simple commands to establish communication with the PS2 gamepad to obtain the button information and create dedicated application commands.

Applications:

- Connect a robot and set up the buttons for advanced and movement control purposes.
- Operate various test tools and machines.
- Control a variety of remote control cars and aircraft when used with the wireless PS2 gamepad.
- Control a variety of application kits by Innovati, Inc.

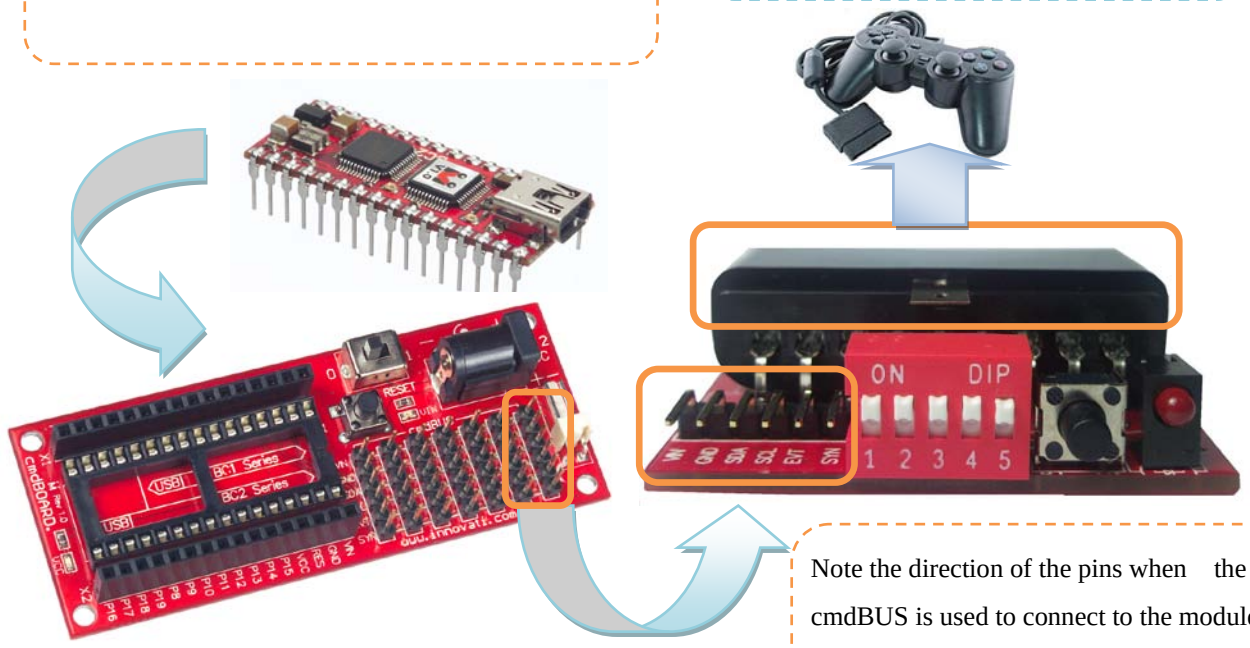
Features:

- It is easy to set. Various applications can be implemented with the dedicated commands simply by connecting cmdBUS to the BASIC Commander.
- The sticks can be set for analogue return and 4-way or 8-way stick position return.
- The origin of the stick can be freely set to a variable between 0 and 10% to avoid jitter.
- The D-pad can be set for 4-way or 8-way stick position return.
- There are 12 function buttons that can be controlled separately or together.
- Calibration is provided with a calibration button. Operation can be interrupted at any time to perform calibration on the stick.
- Customizable button functions, including the time at which the button continuous trigger starts or the continuous trigger rate, can be set via commands.
- You can enable the lock feature for the analogue stick to avoid accidental press.
- Customizable gamepad vibration strength and duration.

Connection: Flip ID switch to the number to be set and connect the cmdBUS to the corresponding pins on the BASIC Commander. You can perform operations via the BASIC Commander after a PS2 gamepad is connected.

It is easier to connect the module if you place the BASIC Commander on the board that provides the cmdBUS pins.

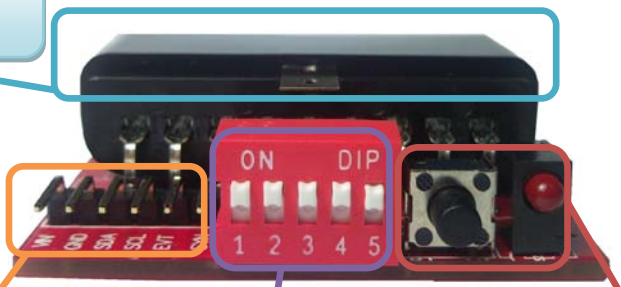
Connect the optional PS2 gamepad



Note the direction of the pins when the cmdBUS is used to connect to the module.

Product specifications :

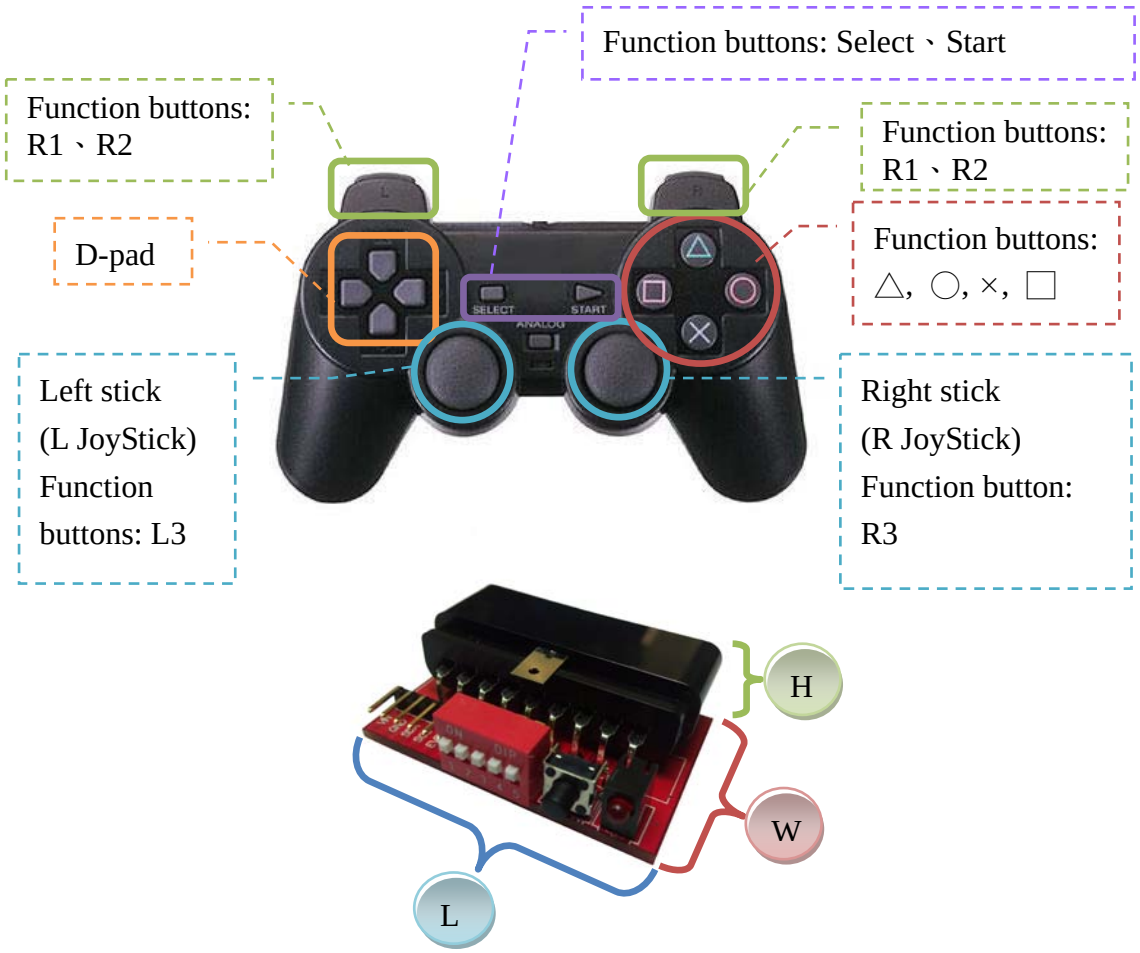
Connect the optional PS2 gamepad



These are cmdBUS pins which are connected to the cmdBUS and BASIC Commander. Note their direction from the left to right are: Vin, Gand, SDA, SCL, EVT, SYN.

The module number switch where the module number is set in binary format. Switch 1 indicates high level. A switch flipped up means 1 while a switch flipped down means 0. The set number shown in the figure is 31.

This is the calibration button that starts calibration. Pressing and holding it for 3 to 5 seconds starts calibration. Note that all commands are invalid during calibration. This is the calibration indicator. When calibration starts, it lights up continuously and goes off after calibration is complete. If it blinks, calibration failed.



L * W * H: 47 * 31* 16 (㎜)

Operating notes:

Module operating temperature -40 ℃~ 123.8 ℃
Module storage temperature -40 ℃~125℃
The module is suitable for the use with the genuine PS2 gamepad. The use of the aftermarket PS2 gamepad is not guaranteed.
How to perform calibration:

1: Press and hold the calibration button for a specific period of time (or via software) to enter Calibration mode. The calibration indicator lights up continuously.
2: Push the stick you want to calibrate up all the way and turn it full two turns to get maximum and minimum values of the XY axes.
3: Finally, center the stick and wait for 3 second to make the stick establish the center of the XY axes.
4: Press the function buttons (△, ○, ×, □) to finish calibration. The calibration indicator goes off.

※ If the calibration indicator blinks, calibration failed. Perform calibration again.

If you accidentally enter Calibration mode, pressing the calibration button again exits this mode.

Command Table:

The following command table shows various commands specifically designed to control the GamepadPs module where the names and parameters of the required commands are shown in bold type or in bold and italic type. Do not change the text in bold type and fill in the appropriate parameters to replace the text in bold and italic type. Note that text is not case-sensitive for innoBASIC Workshop.

Before running GamepadPs commands, define the corresponding parameters and number at the beginning of your program. For example:

Peripheral *ModuleName* As GamepadPs @ *ModuleID*

Command format	Command function
Related gamepad calibration commands	
LStickCalibration( )	Start Calibration mode for the **left** stick. After this command is executed, the stick enters Calibration mode and the calibration LED continuously lights up. At this time, push it up all the way and turn it full two turns to get maximum and minimum values of the XY axes. Center the stick and wait for 3 second to make the stick establish the center of the XY axes. Finally press the function buttons to exit. The LED goes off and the calibration is complete. ※ If the calibration LED blinks, calibration failed. Function buttons: △, ○, ╳, □
RStickCalibration( )	Start Calibration mode for the **right** stick. After this command is executed, the stick enters Calibration mode and the calibration LED continuously lights up. At this time, push it up all the way and turn it full two turns to get maximum and minimum values of the XY axes. Center the stick and wait for 3 second to make the stick establish the center of the XY axes. Finally press the function buttons to exit. The LED goes off and the calibration is complete. ※ If the calibration LED blinks, calibration failed. Function buttons: △, ○, ╳, □
StickCalibration( )	Simultaneously start calibration mode for the left and right sticks. After this command executed, the sticks enter

	Calibration mode and the calibration LED continuously lights up. At this time, push them up all the way and turn them full two turns to get maximum and minimum values of the XY axes. Center the sticks and wait for 3 second to make the sticks establish the center of the XY axes. Finally press the function buttons to exit. The LED goes off and the calibration is complete. ※ If the calibration LED blinks, calibration failed. Function buttons: △, ○, ╳, □
SetCalibrationLX(*LxMin,LxCen,LxMax*)	Set the calibration value of the X axis of the **left stick.** Three Byte parameters are required which are: ***LxMin,*** which indicates the minimum stick calibration value; ***LxCen,*** which indicates the center point; and ***LxMax,*** which indicates the maximum stick calibration value. ※ Note the setting sequence during the manual settings. Enter an integer value between 0~255.
SetCalibrationLY(*LyMin,LyCen,LyMax*)	Set the calibration value of the Y axis of the **left stick.** Three Byte parameters are required which are: ***LyMin,*** which indicates the minimum stick calibration value; ***LyCen,*** which indicates the center point; and ***LyMax,*** which indicates the maximum stick calibration value. ※ Note the setting sequence during the manual settings. Enter an integer value between 0~255.
SetCalibrationRX(*RxMin,RxCen,RxMax*)	Set the calibration value of the X axis of the **right stick.** Three Byte parameters are required which are: ***RxMin,*** which indicates the minimum stick calibration value; ***RxCen,*** which indicates the center point; and ***RxMax,*** which indicates the maximum stick calibration value. ※ Note the setting sequence during the manual settings. Enter an integer value between 0~255.
SetCalibrationRY(*RyMin,RyCen,RyMax*)	Set the calibration value of the Y axis of the **right stick.** Three Byte parameters are required

	which are: ***RyMin,*** which indicates the minimum stick calibration value; ***RyCen,*** which indicates the center point; and ***RyMax,*** which indicates the maximum stick calibration value. ※ Note the setting sequence during the manual settings. Enter an integer value between 0~255.
GetCalibrationLX(*LxMin,LxCen,LxMax*)	Get the calibration value of the X axis of the **left stick.** The minimum value is stored in ***LxMin***, the center point is stored in ***LxCen*** and the maximum value is stored in ***LxMax***. The return value is an integer value between 0~255.
GetCalibrationLY(*LyMin,LyCen,LyMax*)	Get the calibration value of the Y axis of the **left stick.** The minimum value is stored in ***LyMin***, the center point is stored in ***LyCen***, and the maximum value is stored in ***LyMax***. The return value is an integer value between 0~255.
GetCalibrationRX(*RxMin,RxCen,RxMax*)	Get the calibration value of the X axis of the **right stick.** The minimum value is stored in ***RxMin***, the center point is stored in ***RxCen***, and the maximum value is stored in ***RxMax***. The return value is an integer value between 0~255.
GetCalibrationRY(*RyMin,RyCen,RyMax*)	Get the calibration value of the Y axis of the **right stick.** The minimum value is stored in ***RyMin***, the center point is stored in ***RyCen***, and the maximum value is stored in ***RyMax***. The return value is an integer value between 0~255.
Related setting commands	
RestoreSettings( )	Running this command restores the settings to the factory defaults as the following shows**:** · The range of all calibration values is set to: Min=0, Cen=128, Max=255 · The range of the center point of the stick is set to: 5 % · The limit range value of the stick is set to: 80 % · Turn off the rapid fire feature. · Set the resolution of the stick to: 128 · Turn off all events · Turn off the vibration feature.

SetLStickDeadZone(*DZx,DZy*)	Set the range of the center point of the **left stick**. The range of the central zone of the stick is set by ***DZx*** and ***DZy,*** which define the central zone of the XY axes. The input is an integer value between 0~10 in percentage. When the stick is moved within the set zone, it is determined that it is at the center point.
SetRStickDeadZone(*DZx,Dzy*)	Set the range of the center point of the **right stick**. The range of the central zone of the stick is set by ***DZx*** and ***DZy,*** which define the central zone of the XY axes. The input is an integer value between 0~10 in percentage. When the stick is moved within the set zone, it is determined that it is at the center point.
GetLStickDeadZone(*DZx,DZy*)	Get the setting of the central range of the **left stick**. The settings of the XY axes are stored in ***DZx and DZy*** respectively. The return value is an integer between 0~10 in percentage.
GetRStickDeadZone(*DZx,Dzy*)	Get the setting of the central range of the **right stick**. The settings of the XY axes are stored in ***DZx and DZy*** respectively. The return value is an integer between 0~10 in percentage.
SetLStickSaturation(*SATx,SATy*)	Set the limit range value of the **left stick**. ***SATx*** and ***SATy*** are used to set the limit range value of the XY axes. The input is an integer value between 60~100 in percentage. After the command is executed, only the maximum value or minimum value will be returned, regardless whether positive or negative. For the set scale value, only the division calculation is performed between the maximum value and minimum value.
SetRStickSaturation(*SATx,SATy*)	Set the limit range value of the **right stick**. ***SATx*** and ***SATy*** are used to set the limit range value of the XY axes. The input is an integer value between 60~100 in percentage. After the command is executed, only the

	maximum value or minimum value will be returned, regardless whether positive or negative. For the set scale value, only the division calculation is performed between the maximum value and minimum value.
GetLStickSaturation(*SATx,SATy*)	Get the limit range value of the **left stick**. The settings of the XY axes are stored in ***SATx and SATy*** respectively. The return value is an integer between 60~100 in percentage.
GetRStickSaturation(*SATx,SATy*)	Get the limit range value of the **right stick**. The settings of the XY axes are stored in ***SATx and SATy*** respectively. The return value is an integer between 60~100 in percentage.
SetLStickRes(*RESx,RESy*)	Set the resolution of the **left stick**. ***RESx*** and ***RESy*** are used to set the resolutions of the XY axes respectively for the number of scales to be divided within the recognizable range. Set the scale to an integer between 0~128. As 0 is also counted, setting 128 indicates that 128 scales are divided from 0 to 127 positively or from 0 to -127 negatively. Note that while 0 and 1 can also be input, the XY values gotten will be 0 after setting.
SetRStickRes(*RESx,RESy*)	Set the resolution of the **right stick**. ***RESx*** and ***RESy*** are used to set the resolutions of the XY axes respectively for the number of scales to be divided within the recognizable range. Set the scale to an integer between 0~128. As 0 is also counted, setting 128 indicates that 128 scales are divided from 0 to 127 positively or from 0 to -127 negatively. Note that while 0 and 1 can also be input, the XY values gotten will be 0 after setting.
GetLStickRes(*RESx,RESy*)	Get the resolution setting of the **left stick.** The settings of the XY axes are stored in ***RESx and RESy*** respectively. The return value is an integer between 0~128 in percentage.
GetRStickRes(*RESx,RESy*)	Get the resolution setting of the **right stick.**

<table>
<tr><td></td><td>The settings of the XY axes are stored in RESx and RESy respectively. The return value is an integer between 0~128 in percentage.</td></tr>
<tr><td>SetKeyRepeatFunc(Key_ID)</td><td>Set whether rapid input is enabled or not.
Enable = 1, Disable = 0
<table><tr><td>Key_ID</td><td>Bit</td><td>Corresponding button</td><td>Decimal</td></tr><tr><td></td><td>0</td><td>△</td><td>1</td></tr><tr><td></td><td>1</td><td>○</td><td>2</td></tr><tr><td></td><td>2</td><td>×</td><td>4</td></tr><tr><td></td><td>3</td><td>□</td><td>8</td></tr><tr><td></td><td>4</td><td>L1</td><td>16</td></tr><tr><td></td><td>5</td><td>R1</td><td>32</td></tr><tr><td></td><td>6</td><td>L2</td><td>64</td></tr><tr><td></td><td>7</td><td>R2</td><td>128</td></tr><tr><td></td><td>8</td><td>Select</td><td>256</td></tr><tr><td></td><td>9</td><td>Start</td><td>512</td></tr><tr><td></td><td>10</td><td>L3</td><td>1024</td></tr><tr><td></td><td>11</td><td>R3</td><td>2048</td></tr></table>EX: If you want to enable it for △, ○,
Key_ID can be set to:
&B11 (binary) or 3 (decimal).</td></tr>
<tr><td>GetKeyRepeatFunc(Key_ID)</td><td>Get the information about whether rapid input is enabled or not.
Enable = 1, Disable = 0
<table><tr><td>Key_ID</td><td>Bit</td><td>Corresponding button</td><td>Decimal</td></tr><tr><td></td><td>0</td><td>△</td><td>1</td></tr><tr><td></td><td>1</td><td>○</td><td>2</td></tr><tr><td></td><td>2</td><td>×</td><td>4</td></tr><tr><td></td><td>3</td><td>□</td><td>8</td></tr><tr><td></td><td>4</td><td>L1</td><td>16</td></tr><tr><td></td><td>5</td><td>R1</td><td>32</td></tr><tr><td></td><td>6</td><td>L2</td><td>64</td></tr><tr><td></td><td>7</td><td>R2</td><td>128</td></tr><tr><td></td><td>8</td><td>Select</td><td>256</td></tr><tr><td></td><td>9</td><td>Start</td><td>512</td></tr><tr><td></td><td>10</td><td>L3</td><td>1024</td></tr><tr><td></td><td>11</td><td>R3</td><td>2048</td></tr></table></td></tr>
</table>

SetRepeatTime(*Time*)	Set the amount of time during which rapid input is enabled. ***Time*** is used to configure. You can enter an integer ranging between 0~255 in 10 ms.
GetRepeatTime(*Time*)	Get the information about the amount of time during which rapid input is enabled. The return value is stored in ***Time***. The return value is an integer ranging between 0~255 in 10 ms.
SetRepeatRate(*Rate*)	Set the rate at which the rapid input is performed. ***Rate*** is used to configure. You can enter an integer ranging between 0~255 in 10 ms.
GetRepeatRate(*Rate*)	Get the information about the rate at which the rapid input is performed. The return value is stored in ***Rate***. The return value is an integer ranging between 0~255 in 10 ms.
Related application commands	
GetLXYPos(*POSx,POSy*)	Get the coordinate value of the **left stick**. Return the XY coordinates which are stored in ***POSx and POSy*** respectively. The default is -127~+127.
GetRXYPos(*POSx,POSy*)	Get the coordinate value of the **right stick**. Return the XY coordinates which are stored in ***POSx and POSy*** respectively. The default is -127~+127.
GetL4WayValue(*Dir*)	Four ways are used to indicate direction. Get the position of the left stick. The return value is stored in ***Dir*** and indicates direction. The return values are only the numbers 0~4 which are: **0: Stick at center point 1: Stick to the right→** **2: Stick downward↓ 3: Stick to the left←** **4: Stick upward↑**
GetR4WayValue(*Dir*)	Four ways are used to indicate direction. Get the position of the **right stick**. The return value is stored in ***Dir*** and indicates direction. The return values are only the numbers 0~4 which are:

<table>
<tr><td></td><td>0: Stick at center point 1: Stick to the right→
2: Stick downward↓ 3: Stick to the left←
4: Stick upward↑</td></tr>
<tr><td>GetL8WayValue(Dir)</td><td>Eight ways are used to indicate direction. Get the position of the left stick.
The return value is stored in Dir and indicates direction. The return values are only the numbers 0~8 which are:
0: Stick at center point 1: Stick to the right→
2: Stick to the downright↘3: Stick downward↓
4: Stick to the downleft↙5: Stick to the left←
6: Stick to the upleft↖7: Stick upward↑
8: Stick to the upright↗</td></tr>
<tr><td>GetR8WayValue(Dir)</td><td>Eight ways are used to indicate direction. Get the position of the right stick.
The return value is stored in Dir and indicates direction. The return values are only the numbers 0~8 which are:
0: Stick at center point 1: Stick to the right→
2: Stick to the downright↘ 3: Stick downward↓
4: Stick to the downleft↙ 5: Stick to the left←
6: Stick to the upleft↖ 7: Stick upward↑
8: Stick to the upright↗</td></tr>
<tr><td>Status = GetKeyStatus( )</td><td>The button status gotten is stored in Status.
Enable = 1, Disable = 0
<table>
<tr><td rowspan="13">Status</td><th>Bit</th><th>Corresponding button</th><th>Decimal</th></tr>
<tr><td>0</td><td>△</td><td>1</td></tr>
<tr><td>1</td><td>○</td><td>2</td></tr>
<tr><td>2</td><td>×</td><td>4</td></tr>
<tr><td>3</td><td>□</td><td>8</td></tr>
<tr><td>4</td><td>L1</td><td>16</td></tr>
<tr><td>5</td><td>R1</td><td>32</td></tr>
<tr><td>6</td><td>L2</td><td>64</td></tr>
<tr><td>7</td><td>R2</td><td>128</td></tr>
<tr><td>8</td><td>Select</td><td>256</td></tr>
<tr><td>9</td><td>Start</td><td>512</td></tr>
<tr><td>10</td><td>L3</td><td>1024</td></tr>
<tr><td>11</td><td>R3</td><td>2048</td></tr>
</table>
</td></tr>
</table>

	EX: If Status = 3, it is enabled for △, ○.
GetDir4Way(*Dir*)	Get the D-pad status and return a value to indicate direction. The return value is stored in ***Dir***. The return values are only the numbers 0~4 which are: **0: None 1: Right→ 2: Down↓** **3: Left← 4: Up↑**
GetDir8Way(*Dir*)	Get the D-pad status and return a value to indicate direction. The return value is stored in ***Dir***. The return values are only the numbers 0~8 which are: **0:** None **1:** Right→ **2:** Downright↘ **3:** Down↓ **4:** Downleft↙ **5:** Left← **6:** Upleft↖ **7:** Up↑ **8:** Upright↗
SetAnalog(*Mode*)	Set the status of the analogue sticks. ***Mode*** is used to configure. You can enter a value between 0 and 3 as shown below: 0: Turn off analogue sticks 1: Enable analogue return for analogue sticks 2: Lock analogue sticks and set them to analogue on 3: Lock analogue sticks and set them to analogue off ※ Default: 1 (On) **0 and 1 are Off Mode and Enable Mode respectively. After setting, the buttons on the stick can be used to switch between modes.** **2 and 3 are Lock Modes. After setting, the buttons on the stick cannot be used to switch between modes. Mode 0 and 1 cannot also be used to switch to normal mode. Keep this in mind during use.**
StartVib(*Time,Level*)	Enable the gamepad vibration feature. ***Time*** is used to configure. ***Level*** is used to set the vibration level. ***Time***: An integer ranges between 0~255. 0: Continues vibration until the StopVib command is given. 1 indicates 1 second with an increment of 100 ms when one is added.

	Level: An integer ranges between 0~255. 0: No vibration. The higher the number is, the stronger the vibration is 1~255.
StopVib( )	Stop the gamepad vibration feature.
GetVibStatus(*Status,Time,Level*)	Get the gamepad vibration status. The return values are stored in S***tatus, Time and Level*** respectively. ***Status***: Gamepad vibration status. 0: Vibration disabled. 1: Vibration enabled. ***Time***: Remaining time of vibration. 0: Vibration status is 0 and vibration stops when the StopVib command is given. 1: The remaining time is less than 1 second. 2~255: Remaining 1+(Time-1)*100 ms ***Level***: Set the vibration level ranging between 0~255 之間。
GetAnalog(*Mode*)	Get the setting status of the analogue stick. The return value is stored in ***Mode***. The return value may be 0 or 1, where: 0: Disable, 1: Enable. (Whether it is locked is unknown.)
GetConnect(*Status*)	Get the connection status of the gamepad. It is stored in ***Status***. The return value may be 0 or 1, where: 0: gamepad is not detected, 1: gamepad is properly connected.
Related application event commands	
SetStickRefreshRate(*Rate*)	When the stick is set to continuous refresh, it is the fastest rate at which an EVENT is generated. ***Rate*** ranges between 1~255 in 10ms. The values other than 1~255 are invalid. 0 and 1 indicate 10ms.
GetStickRefreshRate(*Rate*)	Get the fastest rate at which EVENT is generated when the stick is set to continuous refresh. The return value is stored in ***Rate*** ranging between 1~255 in 10ms.
EnableLStickEvent( )	Enable **StickEvent of the left stick**. The **SetStickRefreshRate** command determines

	the generation rate.
DisableLStickEvetn( )	Disable **StickEvent of the left stick**.
EnableRStickEvent( )	Enable **StickEvent of the right stick**. The **SetStickRefreshRate** command determines the generation rate.
DisableRStickEvetn( )	Disable **StickEvent of the right stick**.
EnableL4WayEvent( )	Enable **4WayEvent of the left stick**.
DisableL4WayEvent( )	Disable **4WayEvent of the left stick**.
EnableR4WayEvent( )	Enable **4WayEvent of the right stick**.
DisableR4WayEvent( )	Disable **4WayEvent of the right stick**.
EnableL8WayEvent( )	Enable **8WayEvent of the left stick**.
DisableL8WayEvent( )	Disable **8WayEvent of the left stick**.
EnableR8WayEvent( )	Enable **8WayEvent of the left stick**.
DisableR8WayEvent( )	Disable **8WayEvent of the left stick**.
EnableKeyPressedEvent( )	Enable **KeyPressedEvent**.
DisableKeyPressedEvent( )	Disable **KeyPressedEvent**.
EnableKeyRelesedEvent( )	Enable **KeyRelesedEvent**.
DisableKeyRelesedEvent( )	Disable **KeyRelesedEvent**.
EnableDir4WayEvent( )	Enable **Dir4WayEvetn**.
DisableDir4WayEvent( )	Disable **Dir4WayEvetn**.
EnableDir8WayEvent( )	Enable **Dir8WayEvetn**.
DisableDir8WayEvent( )	Disable **Dir8WayEvetn**.

Application events provided by module:

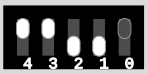
Event	Enable conditions
LStickEvent	The event is generated when the **left stick** starts movement. Return is performed based on the frequency set by **SetStickEvent**.
RStickEvent	The event is generated when the **right stick** starts movement. Return is performed based on the frequency set by **SetStickEvent**.
L4WayEvent	The event is generated when the **left stick changes its direction.** **It is not related to SetStickEvent**.
R4WayEvent	The event is generated when the **right stick changes its direction.** **It is not related to SetStickEvent**.
L8WayEvent	The event is generated when the **left stick changes its direction.** **It is not related to SetStickEvent**.
R8WayEvent	The event is generated when the **right stick changes its direction.** **It is not related to SetStickEvent**.
KeyPressedEvent	It is common to all buttons. When **RepeatKey** is disabled, press any button to generate the event. When **RepeatKey is enabled, press any button and the event is generated based on the time set by RepeatTime and the rate set by RepeatRate**.
KeyReleasedEvent	It is common to all buttons. The event is generated when the action set by KeyRelese is detected.
Dir4WayEvent	The event is generated when the D-pad status changes.
Dir8WayEvent	The event is generated when the D-pad status changes.
CalibrationEndEvent	The event is generated when the calibration ends. Always Enable
ConChangeEvent	The event is generated when it is determined that the gamepad is connected or disconnected. Always Enable

Sample program:

```
Peripheral Ps As GamePadPs @ 31          'Set the module number
Dim b4Dir As Byte                        'Store the direction value gotten
Dim b8WayL,b8WayR As Byte                'Store the direction value of the stick gotten
Dim wStatus As Word                      'Store the button status value gotten

Sub Main()
    Ps.EnableKeyPressedEvent()           'Enable button pressed event
    Ps.EnableKeyReleasedEvent()          'Enable button release event
    Debug "/////  GamePadPs Demo ////" 'Terminal Window shows plan
    Debug CSRXY(1,2),"Direction:"
    Debug CSRXY(1,3),"RStick8Way:"
    Debug CSRXY(1,4),"LStick8Way:"
    Debug CSRXY(1,5),"GetKeyStatus:"

Do
    Ps.GetDir4Way(b4Dir)                 'Get the D-pad status by returning the 4-way directions
    Debug CSRXY(11,2),b4Dir              'Display in Terminal Window (column 11 and row 2)
    Ps.GetR8WayValue(b8WayR)             'Get the right stick status by returning one of the 8-way directions
    Debug CSRXY(12,3),b8WayR             'Display in Terminal Window (column 12 and row 3)
    Ps.GetL8WayValue(b8WayL)             'Get the right left status by returning one of the 8-way directions
    Debug CSRXY(12,4),b8WayL             'Display in Terminal Window (column 12 and row 4)
    Debug CSRXY(15,5),%BIN12 wStatus 'Display Loop in binary format in Terminal Window
                                         '(column 15 and row 5)
Loop
End Sub

Event Ps.KeyPressedEvent()               'Button pressed event
      wStatus = Ps.GetKeyStatus          'Get the current button status and store it in wStatus
End Event

Event Ps.KeyReleasedEvent()              'Button release event
      wStatus = Ps.GetKeyStatus          'Get the current button status and store it in wStatus
End   Event
```

Appendix

Module Number Switch Table:

	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