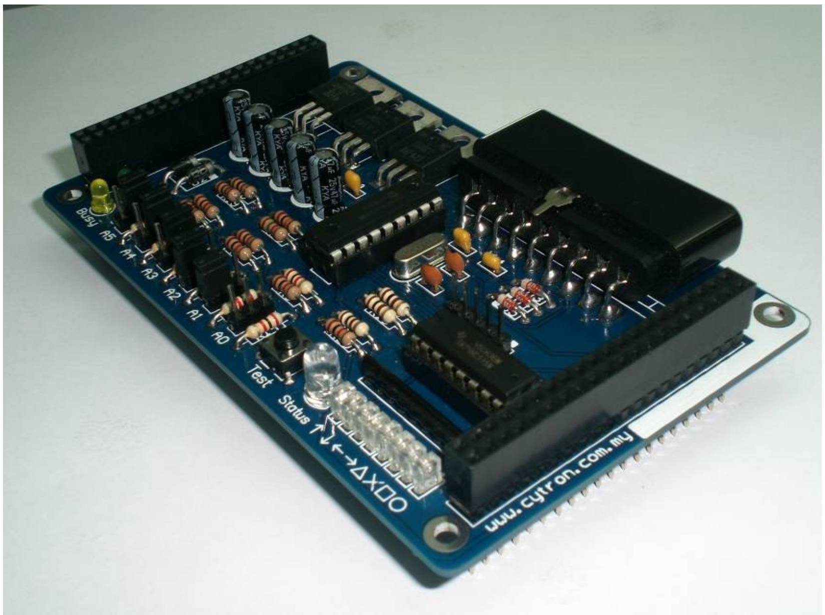
IFC-PS01
Interface Free Controller
Play Station 2 Card



Card Library Functions for Visual C# Express and Visual Basic Express

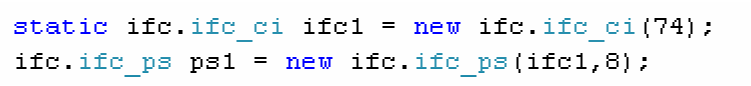
V1.0

Apr 2009

Information contained in this publication regarding device applications and the like is intended through suggestion only and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. No representation or warranty is given and no liability is assumed by Cytron Technologies Incorporated with respect to the accuracy or use of such information or infringement of patents or other intellectual property rights arising from such use or otherwise. Use of Cytron Technologist's products as critical components in life support systems is not authorized except with express written approval by Cytron Technologies. No licenses are conveyed, implicitly or otherwise, under any intellectual property rights.

Function Prototype for Play Station 2 card (PS01)

This document explains the function prototype for controlling IFC-PS01 using PC through IFC-CI00. User may also use ‘object browser’ under Microsoft Visual C# to view the summary, parameter and return value description of IFC-PS01 function prototype. User need to add reference ‘ifc_ci.dll’ and ‘ifc_ps.dll’ for IFC-CI00 and IFC-PS01 card in order to control/communicate IFC-PS01 using PC. Please note that before user start the programming, user need to initialize the ‘ifc.ifc_ci’ and ‘ifc.ifc_ps’ in order to use the functions to control IFC-PS01. Example of creating a ‘ifc.ifc_ci’ class called ‘ifc1’ and ‘ifc.ifc_ps’ class called ps1:



```
static ifc.ifc_ci ifc1 = new ifc.ifc_ci(74);
ifc.ifc_ps ps1 = new ifc.ifc_ps(ifc1,8);
```

For ‘ifc.ifc_ci’ class, user need to specified the COM Port that is connected to IFC-CI00 and for ‘ifc.ifc_ps’ class, user need to specified the IFC-CI00 in use and also the address for IFC-PS01. Please make sure that the address must be unique and different with other IFC card in the IFC system.

Function Prototype	Example	Summary	Parameter Description	Return Value
ifc_ps(ifc.ifc_ci *ifc_ci*, int *address*)	**ifc.ifc_ps**(ifc1, 8)	Initializes a new instance of the ifc.ifc_ps class using the specified ifc.ifc_ci and address for IFC-PS01.	*ifc_ci*: ifc.ifc_ci in use. *address*: Address for IFC-PS01, in range of 0 to 63. (int)	
ifc_ps(ifc.ifc_ci *ifc_ci*, byte *address*)	**ifc.ifc_ps**(ifc1, 8)	Initializes a new instance of the ifc.ifc_ps class using the specified ifc.ifc_ci and address for IFC-PS01.	*ifc_ci*: ifc.ifc_ci in use. *address*: Address for IFC-PS01, in range of 0 to 63. (byte)	
ps_stat()	**ps1.ps_stat**()	To check the connectivity of PS2 controller to IFC-PS01		Return true if PS2 controller is connected to IFC-PS01. (bool)

void ps_vibrate(int *motor*, **int** *vibrate*)	**ps1.ps_vibrate(1, 1)**	To control the vibrator motors on PS2 controller.	*motor*: Vibrator motors on PS2 controller in range of 1 to 2. (int) *vibrate*: For Motor 1, the valid value for vibrate is 1 to turn ON and 0 to turn OFF. For Motor 2, the valid value will be in range of 0 to 255 which represent the percentage of the vibration. (int)	
void ps_vibrate(int *motor*, **bool** *vibrate*)	**ps1.ps_vibrate(1, true)**	To control the vibrator motors on PS2 controller.	*motor*: Vibrator motors on PS2 controller in range of 1 to 2. (int) *vibrate*: True to turn ON and false to turn OFF the selected vibrator motor. (bool)	
void ps_vibrate(byte *motor*, **byte** *vibrate*)	**ps1.ps_vibrate(2, 255)**	To control the vibrator motors on PS2 controller.	*motor*: Vibrator motors on PS2 controller in range of 1 to 2. (byte) *vibrate*: For Motor 1, the valid value for vibrate is 1 to turn ON and 0 to turn OFF. For Motor 2, the valid value will be in range of 0 to 255 which represent the percentage of the vibration. (byte)	
void ps_vibrate(byte *motor*, **bool** *vibrate*)	**ps1.ps_vibrate(2, true)**	To control the vibrator motors on PS2 controller.	*motor*: Vibrator motors on PS2 controller in range of 1 to 2. (byte) *vibrate*: True to turn ON and false to turn OFF the selected vibrator motor. (bool)	
ps_joy		To read the analog value of joystick on PS2 controller.		
ps_sw		To read the status of push buttons on PS2 controller.		

Table 1 Function Prototype for Play Station (PS01)

Table 2 is function prototype for sub function of ‘**ps_joy**’. These functions are to read the analog value of joystick on PS2 controller.

Function Prototype	Example	Summary	Return Value
joy_ld()	**ps1.ps_joy.joy_ld**()	Left joystick in down direction.	Return the joystick information in down direction. The value is 0 when the joystick is at center or upper, and the value change to 100 when the joystick is being moved down. (byte)
joy_ll()	**ps1.ps_joy.joy_ll**()	Left joystick in left direction.	Return the joystick information in left direction. The value is 0 when the joystick is at center or right, and the value change to 100 when the joystick is being moved to left. (byte)
joy_lr()	**ps1.ps_joy.joy_lr**()	Left joystick in right direction.	Return the joystick information in right direction. The value is 0 when the joystick is at center or left, and the value change to 100 when the joystick is being moved to right. (byte)
joy_lu()	**ps1.ps_joy.joy_lu**()	Left joystick in up direction.	Return the joystick information in up direction. The value is 0 when the joystick is at center or lower, and the value change to 100 when the joystick is being moved up. (byte)
joy_lx()	**ps1.ps_joy.joy_lx**()	Left joystick in x-axis.	Return the joystick information in x-axis. The value is around 128 when the joystick is at center, 0 when the joystick is at left side, and 255 when the joystick is at the right side. (byte)
joy_ly()	**ps1.ps_joy. joy_ly**()	Left joystick in y-axis.	Return the joystick information in y-axis. The value is around 128 when the joystick is at center, 0 when the joystick is at upper side, and 255 when the joystick is at the lower side. (byte)
joy_rd()	**ps1.ps_joy. joy_rd**()	Right joystick in down direction.	Return the joystick information in down direction. The value is 0 when the joystick is at center or upper, and the value change to 100 when the joystick is being moved down. (byte)

joy_rl()	**ps1.ps_joy. joy_rl**()	Right joystick in left direction.	Return the joystick information in left direction. The value is 0 when the joystick is at center or right, and the value change to 100 when the joystick is being moved to left. (byte)
joy_rr()	**ps1.ps_joy. joy_rr**()	Right joystick in right direction.	Return the joystick information in right direction. The value is 0 when the joystick is at center or left, and the value change to 100 when the joystick is being moved to right. (byte)
joy_ru()	**ps1.ps_joy. joy_ru**()	Right joystick in up direction.	Return the joystick information in up direction. The value is 0 when the joystick is at center or lower, and the value change to 100 when the joystick is being moved up. (byte)
joy_rx()	**ps1.ps_joy. joy_rx**()	Right joystick in x-axis.	Return the joystick information in x-axis. The value is around 128 when the joystick is at center, 0 when the joystick is at left side, and 255 when the joystick is at the right side. (byte)
joy_ry()	**ps1.ps_joy. joy_ry**()	Right joystick in y-axis.	Return the joystick information in y-axis. The value is around 128 when the joystick is at center, 0 when the joystick is at upper side, and 255 when the joystick is at the lower side. (byte)

Table 2

Table 2 is function prototype for sub function of '**ps_sw**'. These functions are to read the status of push buttons on PS2 controller.

Function Prototype	Example	Summary	Return Value
circle()	**ps1.joy_sw.circle**()	Circle button on PS2 controller.	Return true if the button is pressed. (bool)
cross()	**ps1.joy_sw.cross**()	Cross button on PS2 controller.	Return true if the button is pressed. (bool)
down()	**ps1.joy_sw.down**()	Down button on PS2 controller.	Return true if the button is pressed. (bool)
joyl()	**ps1.joy_sw.joyl**()	Joystick Left button on PS2 controller.	Return true if the button is pressed. (bool)
joyr()	**ps1.joy_sw.joyr**()	Joystick Right button on PS2 controller.	Return true if the button is pressed. (bool)
l1()	**ps1.joy_sw.l1**()	L1 button on PS2 controller.	Return true if the button is pressed. (bool)
l2()	**ps1.joy_sw.l2**()	L2 button on PS2 controller.	Return true if the button is pressed. (bool)
left()	**ps1.joy_sw.left**()	Left button on PS2 controller.	Return true if the button is pressed. (bool)
r1()	**ps1.joy_sw.r1**()	R1 button on PS2 controller.	Return true if the button is pressed. (bool)
r2()	**ps1.joy_sw.r2**()	R2 button on PS2 controller.	Return true if the button is pressed. (bool)
right()	**ps1.joy_sw.right**()	Right button on PS2 controller.	Return true if the button is pressed. (bool)
select()	**ps1.joy_sw.select**()	Select button on PS2 controller.	Return true if the button is pressed. (bool)
square()	**ps1.joy_sw.square**()	Square button on PS2 controller.	Return true if the button is pressed. (bool)
start()	**ps1.joy_sw.start**()	Start button on PS2 controller.	Return true if the button is pressed. (bool)
triangle()	**ps1.joy_sw.triangle**()	Triangle button on PS2 controller.	Return true if the button is pressed. (bool)
up()	**ps1.joy_sw. up**()	Up button on PS2 controller.	Return true if the button is pressed. (bool)

Table 3

Prepared by
Cytron Technologies Sdn. Bhd.
19, Jalan Kebudayaan 1A,
Taman Universiti,
81300 Skudai,
Johor, Malaysia.

Tel: +607-521 3178
Fax: +607-521 1861

URL: www.cytron.com.my
Email: support@cytron.com.my
sales@cytron.com.my