PS2 Controller

Summary

This document provides detailed reference information with respect to the PS/2 Controller peripheral devices.

Core Reference
CR0109 (v1.3) May 26, 2005

The PS/2 Controller provides a bidirectional, synchronous serial interface between a host microcontroller and a PS/2 device (keyboard or mouse).

Important Notice: Supply of this soft core under the terms and conditions of the Altium End-User License Agreement does not convey nor imply any patent rights to the supplied technologies. Users are cautioned that a license may be required for any use covered by such patent rights.

Features

- IBM PS/2 device compatible (keyboard or mouse)
- Bidirectional transmission (synchronous and serial).
- Hardware parity (odd) check on all data received from PS/2 device
- Hardware parity (odd) generation on all data sent to PS/2 device
- Wishbone-compliant (**PS2_W only**)

Available devices

Both Wishbone and non-Wishbone versions of the Controller are available – the PS2_W and the PS2 respectively. These devices can be found in the FPGA Peripherals integrated library `(\Program Files\Altium2004\Library\Fpga\FPGA Peripherals.IntLib)`.

PS2_W – Wishbone-compliant version

Functional Description

Symbol

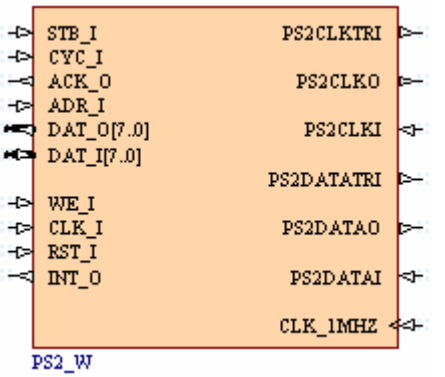

Figure 1. PS/2 Controller Symbol – Wishbone version (PS2_W)

Pin description

Table 1. PS2_W pin description

Name	Type	Polarity/ Bus size	Description
Control Signals			
CLK_I	I	Rise	External system clock
CLK_1MHZ[1]	I	Rise	Reference clock for the generation of timing constants specified within the PS/2 timing specification standard
RST_I	I	High	External system reset
Host Microcontroller Interface Signals			
STB_I	I	High	Strobe signal. When asserted, indicates the start of a valid Wishbone data transfer cycle
CYC_I	I	High	Cycle signal. When asserted, indicates the start of a valid Wishbone cycle

[1] This clock input signal should ideally be 1MHz. It should, at any rate, be faster than the connected PS/2 device. If this clock signal is too low, the Controller may remain waiting too long for data that might not come from the PS/2 device. If it is too high, the Controller may not wait long enough to correctly receive the data sent from the PS/2 device.

Name	Type	Polarity/ Bus size	Description
ACK_O	O	High	Standard Wishbone device acknowledgement signal. When this signal goes high, the Controller (Wishbone Slave) has finished execution of the requested action and the current bus cycle is terminated
ADR_I	I	Level	Address bus, used to select an internal register of the device for writing to/reading from: 0 = Wishbone Control register (WCREG) 1 = Wishbone Data register (WDREG)
DAT_O	O	8	Data to be sent to host microcontroller
DAT_I	I	8	Data received from host microcontroller
WE_I	I	Level	Write enable signal. Used to indicate whether the current local bus cycle is a Read or Write cycle: 0 = Read 1 = Write
INT_O	O	High	Interrupt signal. Used to alert the CPU to the presence of data received from the connected PS/2 device. This signal is asserted for at least 13 periods of *CLK_I* when 1 byte of data has been received from the PS/2 device. (Note: INT_O will not be asserted if the parity of the byte received is not correct)
PS/2 Interface Signals			
PS2CLKTRI	O	Low	Tri-state enable signal for the *PS2CLK* bidirectional buffer
PS2CLKO	O	-	PS2 clock output
PS2CLKI	I	-	PS2 clock input
PS2DATATRI		Low	Tri-state enable signal for the *PS2DATA* bidirectional buffer
PS2DATAO	O	-	PS2 data output (data from the PS2 Controller to the PS/2 device)
PS2DATAI	I	-	PS2 data input (data from the PS/2 device to the PS2 Controller)

Note: To simplify using the bidirectional PSDATA and PSCLK buses, the schematic symbol includes a bus pin for each direction, allowing them to be wired independently. Configuration of bus direction is performed under program control.

Hardware Description

Block Diagram

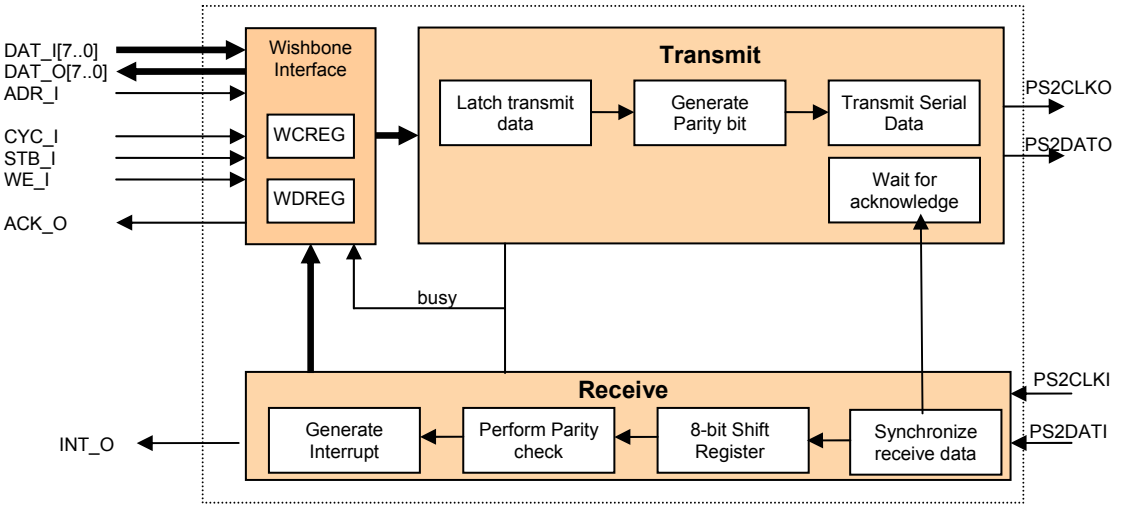

Figure 2. PS2_W Controller block diagram

Internal Wishbone Registers

All Wishbone communication is carried out through two dedicated registers – the Wishbone Control register (WCREG) and Wishbone Data register (WDREG) respectively.

Wishbone Control register (WCREG)

The Wishbone Control register holds the current state of the Controller and is used to control transmission of data to the PS/2 slave device.

Table 2. The WCREG register

MSB LSB

X	X	X	X	X	X	STB	BUSY

Table 3. The WCREG register bit functions

Bit	Symbol	Function
WCREG.7	X	Not used. Read as '0'
WCREG.6	X	Not used. Read as '0'
WCREG.5	X	Not used. Read as '0'
WCREG.4	X	Not used. Read as '0'

Bit	Symbol	Function
WCREG.3	X	Not used. Read as '0'
WCREG.2	X	Not used. Read as '0'
WCREG.1	STB	Strobe flag. This bit is set by the CPU to initiate transmission to the connected PS/2 slave device. This bit is automatically cleared by the Controller when it starts transmission
WCREG.0	BUSY	Busy flag. This bit is set when the PS2_W Controller performs the requested task

Wishbone Data register (WDREG)

When written to, the Wishbone Data register writes data to be transmitted to the slave PS/2 device, into the Controller's transmit buffer.

When read, the register returns whatever data is currently in the Controller's receive buffer. When INT_O is asserted, reading the register returns the byte of data received from the slave PS/2 device.

Table 4. The WDREG register

MSB LSB

DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0

Table 5. The WDREG register bits description

Bit	Symbol	Function
WDREG.7	DB7	Data bit 7
WDREG.6	DB6	Data bit 6
WDREG.5	DB5	Data bit 5
WDREG.4	DB4	Data bit 4
WDREG.3	DB3	Data bit 3
WDREG.2	DB2	Data bit 2
WDREG.1	DB1	Data bit 1
WDREG.0	DB0	Data bit 0

Register reset values

Table 6 shows the values contained in each of the PS2_W's internal Wishbone registers after an external system reset has been received on the RST_I input.

Table 6. Internal Wishbone register reset values

Register	Value after reset
WCREG	00h
WDREG	00h

CPU to PS/2 device - communications overview

The PS2_W Controller provides the Wishbone interface between a microcontroller (host) on the one side and a PS/2 device (keyboard or mouse) on the other. The host microcontroller sends data to and receives data from the PS2_W Controller, through the Controller's internal Wishbone Data register (WDREG).

Note: Switching speeds vary depending on the physical FPGA device used. High speed devices may exhibit undesirable noise effects on any NanoBoard port plug-in connections that are unterminated.

Transmission from CPU to PS/2 device

The CPU sends data to the PS2_W Controller through the Wishbone Data register (WDREG). Once a byte of data is written to this register, the Controller puts it into its transmit buffer.

The PS2_W Controller stores this data until the STB flag (WCREG.1) is set in the Wishbone Control register.

Once the STB flag is set, the PS2_W Controller prepares a single byte of data for transmission. The byte of data is organized into a frame. The Controller also generates a parity bit (odd parity) for the frame — necessary for error checking when a frame of data is received by the PS/2 device.

The frame of data is then transmitted to the connected PS/2 device, over the bidirectional PS2DATA bus (in the schematic design, this data leaves the PS2_W Controller on the PS2DATAO pin). It should be noted that only a single byte of data can be sent at any one time. To send an additional byte of data, the host CPU should read the state of the BUSY flag (WCREG.0) and once it is cleared, write a new byte of data into the Wishbone Data register and set the STB flag in the Wishbone Control register.

See the section 'Transmission from Controller to PS/2 device' for full details of the transmission protocol from PS2_W Controller to PS/2 slave device.

For detailed information with respect to accessing the internal Wishbone registers for write/read operations, see the sections Writing to the Internal Wishbone Registers and Reading from the Internal Wishbone Registers, respectively.

Transmission from PS/2 device to CPU

Data sent from the PS/2 device over the PS2DATA bus arrives at the PS2_W Controller on the PS2DATAI pin. The received data, previously clocked in on the falling edge of the PS2CLK signal, is then synchronized with the clock signal internal to the FPGA device. See the section 'Transmission

from PS/2 device to Controller' for full details of the transmission protocol from PS/2 slave device to PS2_W Controller.

Each received data frame is shifted out through the Receive Shift Register. Parity checking is then performed to ensure the integrity of the data.

If the parity check reveals no errors, the PS2_W Controller then generates an interrupt signal to the CPU – appearing as a high on the INT_O output pin. INT_O stays high for at least 13 periods of the external CLK_I signal, in order for the CPU to 'see' the interrupt. This alerts the CPU to the fact that a byte of data has been received from the PS/2 device.

The CPU then reads the data from the PS2_W Controller's internal Wishbone Data register (WDREG).

The CPU can interrogate the state of the PS2_W Controller at any time by reading the state of the BUSY flag (WCREG.0) in the Controller's internal Wishbone Control register. When this bit is high, the Controller is either transmitting data to, or receiving data from, the PS/2 device.

For detailed information with respect to accessing the internal Wishbone registers for write/read operations, see the sections Writing to the Internal Wishbone Registers and Reading from the Internal Wishbone Registers, respectively

PS2 Protocol

The PS2_W Controller implements a bidirectional protocol for synchronous serial transmission between the host and the PS/2 device (keyboard or mouse).

The PS/2 device sends information to the Controller, for example when a key is pressed on the keyboard, or when the mouse has moved position.

The Controller sends information to the PS/2 device when a specific command — sent from the microcontroller — needs to be executed. When the connected PS/2 device is a keyboard, such a command could be to change the state of an LED (Num Lock, Caps Lock, Scroll Lock).

The clock signal for the Controller-Device interface (PS2CLK) is always generated by the PS/2 device, but is not continuous. Total control over transmission is, however, ultimately in the hands of the microcontroller. The table below illustrates the various states that can be entered, with respect to the Controller-Device interface.

Table 7. PS2 Controller Transmission states

State	PS2DATA	PS2CLK	Note
Idle	High	High	-
Inhibit Transmission	High	Low	Here, the Controller is asking the PS/2 device to generate the clock, in order for the transmitter to transmit serial data out of the shift register.
Host Send Request	Low	High	

Note that the Host Send Request state can only be entered by passing through the Inhibit Transmission state.

Transmission from PS/2 device to Controller

The PS/2 device is free to send data to the Controller when the Controller is in the Idle state. This means that both the PS2CLK and PS2DATA lines are high. The PS/2 device must then wait for half a PS2CLK period before it can start its transmission.

The transmitted data is sent in frames, with each frame carrying a single byte of data. The number of frames sent depends on the number of bytes of data constituting the message to be sent. For example, most scan codes (the codes that represent keys) for a keyboard are a single byte in length, but some can be longer. When sending information to tell the host microcontroller that a key has been released, an extra byte of data needs to be sent. It should be noted that only a single byte of data (therefore a single frame) can be sent to the Controller at any one time. To send additional bytes of data (further frames) the PS/2 device must wait until the Controller returns to the Idle state, before starting each additional transmission.

When sending data from the device to the Controller, the PS2 frame consists of 11 fields. The order of these fields is shown in Table 8.

Table 8. PS2 Frame (from PS/2 device to Controller)

Start	Data 0	Data 1	Data 2	Data 3	Data 4	Data 5	Data 6	Data 7	Parity	Stop

Start - indicates the start of a transmitted frame. This is always a '0'.

Data0-Data7 - the actual data bits. The least significant bit (Data0) is always sent first.

Parity - used for error detection. Odd parity is used. Therefore, if the number of '1's in Data0-7 is even, the Parity bit is set (1). If the number of '1's is odd, the Parity bit is cleared (0).

Stop - indicates the end of the frame currently being transmitted. This is always a '1'

Figure 7 shows the timing for this frame transmission. The PS2_W Controller reads each bit in the frame on the falling edge of the clock signal.

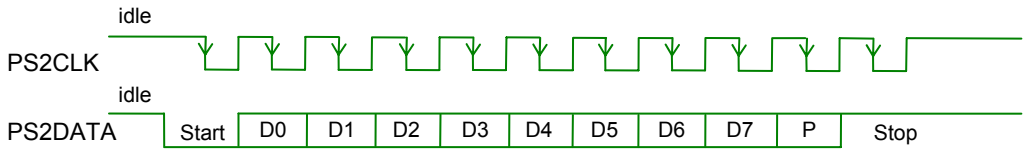

Figure 3. Transmission of data from PS/2 device to Controller

The host microcontroller, as previously mentioned, always has full control over the transmission of data to and from the PS/2 device:

- By monitoring the BUSY flag (WCREG.0) in the Wishbone Control register, it knows the current status of the PS2_W Controller – whether it is sending data to or receiving data from the connected PS/2 device
- The INT_O signal alerts it to any data that has been received by the PS2_W Controller and is therefore ready to be read.

The host can therefore complete reception of data or start a transmission, whenever it deems is appropriate to do so.

The transmission of data from the PS/2 device to the Controller can be inhibited at any time, by the host setting the STB flag (WCREG.1) in the Wishbone Control register. The PS2_W Controller, detecting that the host wants to interrupt, takes the PS2CLK line low for at least one clock period.

When such an event arises, the Controller enters the Inhibit Transmission state. The response of the PS/2 device depends on where in the transmission the inhibit occurs:

- before reception of the Start bit – the PS/2 device has not yet begun transmission of the data. The frame of data will be buffered until the interface re-enters Idle state and the device is free to commence transmission.
- during transmission of a frame (anywhere after the first high-to-low transition of PS2CLK and before the last high-to-low transition) – the PS/2 device aborts transmission and prepares to retransmit the entire message again. The message could be a single byte scan code, in which case the same byte will be retransmitted when the interface re-enters Idle state. If the message being transmitted was composed of several bytes (for example an extended key on a keyboard has been pressed or released), all bytes would be retransmitted, irrespective of how many bytes of the message had already been transmitted.
- After reception of the Stop bit – the PS/2 device has finished transmission of the data (constituent frames) and so retransmission is not necessary. Any new data will be buffered until the interface is in Idle state.

For a PS/2 keyboard, up to 16 bytes of key strokes can be buffered for transmission to the Controller. For a PS/2 mouse, only the current movement packet is stored for transmission.

Transmission from Controller to PS/2 device

For the Host microcontroller to send a command to the PS/2 device via the PS2_W Controller, the STB flag (WCREG.1) in the Wishbone Control register must be set

To effect transmission, the PS2_W Controller must then enter the Host Send Request state. This is achieved by taking the following actions:

- the PS2CLK line is first taken low for at least one clock period (entering Inhibit Transmission state)
- the PS2DATA line is then taken low (providing the Start bit of the frame to be transmitted)
- the PS2CLK line is then released (still holding PS2DATA low).

The PS/2 device regularly checks the data and clock lines for this state and when detected, starts to generate the PS2CLK signal.

When sending data from the Controller to the PS/2 device, the PS2 frame consists of 10 fields. The order of these fields is shown in Table 9.

Table 9. PS2 Frame (from Controller to PS/2 device)

Start	Data 0	Data 1	Data 2	Data 3	Data 4	Data 5	Data 6	Data 7	Parity

Start - indicates the start of a transmitted frame. This is always a '0'.

Data0-Data7 - the actual data bits. The least significant bit (Data0) is always sent first.

Parity - used for error detection. Odd parity is used. Therefore, if the number of '1's in Data0-7 is even, the Parity bit is set (1). If the number of '1's is odd, the Parity bit is cleared (0).

Figure 8 shows the timing for this frame transmission. The PS/2 device reads each bit in the frame on the falling edge of the clock signal.

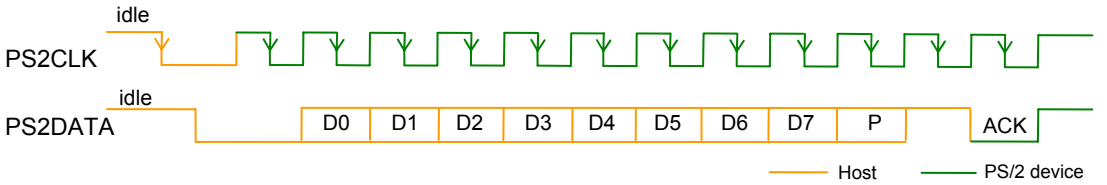

Figure 4. Transmission of data from Controller to PS/2 device

After transmission of the Parity bit, the PS2DATA line is released back to its idle state. The PS/2 device monitors this line and when it has detected no change from the idle state for at least one clock period, it takes the line low again for a single clock period. This tells the Controller that the data has been fully received by the device, i.e. an acknowledgement. Both PS2CLK and PS2DATA lines are released to their idle state. The Controller can initiate another transmission request at this point.

If the PS/2 device does not detect the release of the PSDATA line after the reception of the parity bit, it will continue to provide the PS2CLK signal. Once the PSDATA line is finally released, the device will pull it low and send the command to 'retransmit byte'. On reception of this byte, the PS2_W Controller should then retransmit the previous byte.

The Controller can inhibit transmission of data at any time before the PS/2 device sends the ACK signal, by taking the PS2CLK line low for at least one clock period.

Writing to the Internal Wishbone Registers

The host microcontroller can write to either of the PS2_W's internal Wishbone registers. Selection of a register – either WCREG or WDREG – is achieved by supplying the unique, binary ID address code of the register. This code is sent to the device and appears at the ADR_I input.

Table 10 shows the unique address IDs associated with the registers for the PS2_W.

Table 10. PS2_W Internal Wishbone register unique address IDs.

Register	Unique Register Address ID
WCREG	0
WDREG	1

An internal Wishbone register of the PS2_W device can be written to in the following circumstances. In each case, the write operation occurs on the rising edge of the CLK_I input.

System Reset

After reception of an external system reset (RST_I goes High), both internal Wishbone registers are loaded with the reset value 00h.

Host Microcontroller Write

Data is written from the host microcontroller to an internal Wishbone register, in accordance with the standard Wishbone data transfer handshaking protocol. This data transfer cycle can be summarized as follows:

- The host presents the unique address ID for the register to be written on its ADR_O output and a valid byte of data on its DAT_O output. It then asserts its WE_O signal, to specify a Write cycle
- The PS2_W receives the address ID on its ADR_I input and prepares to receive data into the selected register
- The host asserts its STB_O and CYC_O outputs, indicating that the transfer is to begin. The PS2_W, which monitors its STB_I and CYC_I inputs on each rising edge of the CLK_I signal, reacts to this assertion by latching the byte of data appearing at its DAT_I input into the target register and asserting its ACK_O signal – to indicate to the host that the data has been received
- The host, which monitors its ACK_I input on each rising edge of the CLK_I signal, responds by negating the STB_O and CYC_O signals. At the same time, the PS2_W negates the ACK_O signal and the data transfer cycle is naturally terminated.

Reading from the Internal Wishbone Registers

Data is read from an internal Wishbone register in accordance with the standard Wishbone data transfer handshaking protocol. This data transfer cycle can be summarized as follows:

- The host presents the unique address ID for the register to be read on its ADR_O output. It then negates its WE_O signal to specify a Read cycle
- The PS2_W receives the address ID on its ADR_I input and prepares to transmit data from the selected register
- The host asserts its STB_O and CYC_O outputs, indicating that the transfer is to begin. The PS2_W, which monitors its STB_I and CYC_I inputs on each rising edge of the CLK_I signal, reacts to this assertion by presenting a valid byte of data at its DAT_O output and asserting its ACK_O signal – to indicate to the host that valid data is present
- The host, which monitors its ACK_I input on each rising edge of the CLK_I signal, responds by latching the byte of data appearing at its DAT_I input and negating the STB_O and CYC_O signals. At the same time, the PS2_W negates the ACK_O signal and the data transfer cycle is naturally terminated.

PS2 – Non-Wishbone version

Functional Description

Symbol

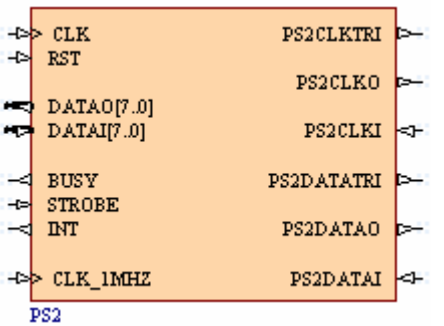

Figure 5. PS/2 Controller Symbol – non-Wishbone version (PS2)

Pin description

Table 11. PS2 pin description

Name	Type	Polarity/ Bus size	Description
Control Signals			
CLK	I	Rising	External (system) clock
CLK_1MHZ[2]	I	Rising	Reference clock for the generation of timing constants specified within the PS/2 timing specification standard.
RST	I	High	External (system) reset
Host Microcontroller Interface Signals			
DATAO	O	8	Output data bus. Used to transfer a byte of data from the PS2 Controller to the host CPU. This data has been received from the PS/2 device.
DATAI	I	8	Input data bus. Used to transfer a byte of data from the host CPU to the PS2 Controller. This data is to be sent to the PS/2 device.

[2] This clock input signal should ideally be 1MHz. It should, at any rate, be faster than the connected PS/2 device. If this clock signal is too low, the Controller may remain waiting too long for data that might not come from the PS/2 device. If it is too high, the Controller may not wait long enough to correctly receive the data sent from the PS/2 device.

Name	Type	Polarity/ Bus size	Description
BUSY	O	High	Active when the PS2 Controller is transmitting data to, or receiving data from, the PS/2 device
STROBE	I	High	This signal is used to initiate a transmission of data from the host CPU to the connected PS/2 device. When asserted, the data on *DATAI* is sent to the PS/2 device. This signal must be asserted for a minimum of one complete period of *CLK*.
INT	O	High	Interrupt signal. Used to alert the CPU to the presence of data received from the connected PS/2 device. This signal is asserted for at least 13 periods of *CLK* when 1 byte of data has been received from the PS/2 device. (Note: INT will not be asserted if the parity of the byte received is not correct)
PS/2 Interface Signals			
PS2CLKTRI	O	Low	Tri-state enable signal for the *PS2CLK* bidirectional buffer
PS2CLKO	O	-	PS2 clock output
PS2CLKI	I	-	PS2 clock input
PS2DATATRI		Low	Tri-state enable signal for the *PS2DATA* bidirectional buffer
PS2DATAO	O	-	PS2 data output (data from the PS2 Controller to the PS/2 device)
PS2DATAI	I	-	PS2 data input (data from the PS/2 device to the PS2 Controller)
STROBE	I	High	This signal is used to initiate a transmission of data from the host CPU to the connected PS/2 device. When asserted, the data on *DATAI* is sent to the PS/2 device. This signal must be asserted for a minimum of one complete period of *CLK*.

Note: To simplify using the bidirectional PSDATA and PSCLK buses, the schematic symbol includes a bus pin for each direction, allowing them to be wired independently. Configuration of bus direction is performed under program control.

Hardware Description

Block Diagram

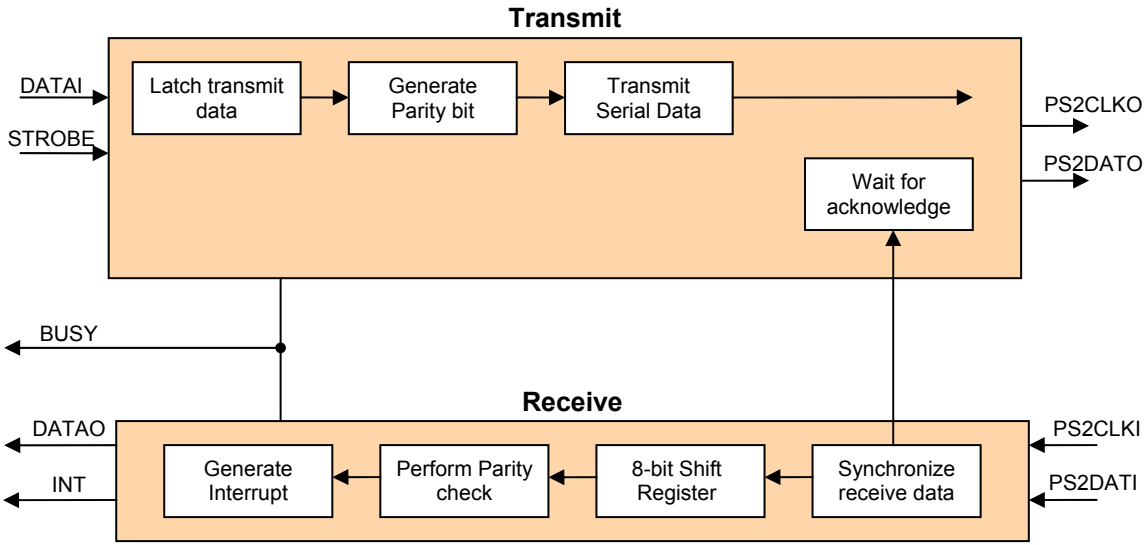

Figure 6. PS2 Controller block diagram

CPU to PS/2 device - communications overview

The PS2 Controller provides the interface between a microcontroller (host) on the one side and a PS/2 device (keyboard or mouse) on the other. The host microcontroller sends data to and receives data from the PS2 Controller, on the DATAI and DATAO buses respectively.

The operational flow can be seen in Figure 6 and is summarized in the following sections.

Note: Switching speeds vary depending on the physical FPGA device used. High speed devices may exhibit undesirable noise effects on any NanoBoard port plug-in connections that are unterminated.

Transmission from CPU to PS/2 device

The CPU sends data to the PS2 Controller on the DATAI bus.

The PS2 Controller stores this data until an active STROBE signal is received from the CPU. The STROBE signal must be high for at least one period of the external system clock signal (on the CLK input).

After receiving a valid STROBE signal, the PS2 Controller prepares a single byte of data for transmission. The byte of data is organized into a frame. The Controller also generates a parity bit (odd parity) for the frame — necessary for error checking when a frame of data is received by the PS/2 device.

The frame of data is then transmitted to the connected PS/2 device, over the bidirectional PS2DATA bus (in the schematic design, this data leaves the PS2 Controller on the PS2DATAO pin). It should be noted that only a single byte of data can be sent at any one time. To send an additional byte of data,

another active STROBE signal must be issued by the host. See the section 'Transmission from Controller to PS/2 device' for full details of the transmission protocol from PS2 Controller to PS/2 device.

Transmission from PS/2 device to CPU

Data sent from the PS/2 device over the PS2DATA bus arrives at the PS2 Controller on the PS2DATAI pin. The received data, previously clocked in on the falling edge of the PS2CLK signal, is then synchronized with the clock signal internal to the FPGA device. See the section 'Transmission from PS/2 device to Controller' for full details of the transmission protocol from PS/2 device to PS2 Controller.

Each received data frame is shifted out through the Receive Shift Register. Parity checking is then performed to ensure the integrity of the data.

If the parity check reveals no errors, the PS2 Controller then generates an interrupt signal to the CPU – appearing as a high on the INT output pin. INT stays high for at least 13 periods of the external CLK signal, in order for the CPU to 'see' the interrupt. This alerts the CPU to the fact that a byte of data has been received from the PS/2 device.

The CPU then reads the data from the PS2 Controller, on the DATAO bus. There is no handshaking between the CPU and PS2 Controller when reading a byte of data. The CPU is running at a far greater speed and so there is no possibility of it missing a byte of data to be read.

The CPU can interrogate the state of the PS2 Controller at any time by reading the state of the Controller’s BUSY output. When this output is high, the Controller is either transmitting data to, or receiving data from, the PS/2 device.

PS2 Protocol

The PS2 Controller implements a bidirectional protocol for synchronous serial transmission between the host and the PS/2 device (keyboard or mouse).

The PS/2 device sends information to the Controller, for example when a key is pressed on the keyboard, or when the mouse has moved position.

The Controller sends information to the PS/2 device when a specific command — sent from the microcontroller — needs to be executed. When the connected PS/2 device is a keyboard, such a command could be to change the state of an LED (Num Lock, Caps Lock, Scroll Lock).

The clock signal for the Controller-Device interface (PS2CLK) is always generated by the PS/2 device, but is not continuous. Total control over transmission is, however, ultimately in the hands of the microcontroller. The table below illustrates the various states that can be entered, with respect to the Controller-Device interface.

Table 12. PS2 Controller Transmission states

State	PS2DATA	PS2CLK	Note
Idle	High	High	-
Inhibit Transmission	High	Low	Here, the Controller is asking the PS/2 device to generate the clock, in order for the transmitter to transmit serial data out of the shift register.
Host Send Request	Low	High	

Note that the Host Send Request state can only be entered by passing through the Inhibit Transmission state.

Transmission from PS/2 device to Controller

The PS/2 device is free to send data to the Controller when the Controller is in the Idle state. This means that both the PS2CLK and PS2DATA lines are high. The PS/2 device must then wait for half a PS2CLK period before it can start its transmission.

The transmitted data is sent in frames, with each frame carrying a single byte of data. The number of frames sent depends on the number of bytes of data constituting the message to be sent. For example, most scan codes (the codes that represent keys) for a keyboard are a single byte in length, but some can be longer. When sending information to tell the host microcontroller that a key has been released, an extra byte of data needs to be sent. It should be noted that only a single byte of data (therefore a single frame) can be sent to the Controller at any one time. To send additional bytes of data (further frames) the PS/2 device must wait until the Controller returns to the Idle state, before starting each additional transmission.

When sending data from the device to the Controller, the PS2 frame consists of 11 fields. The order of these fields is shown in Table 13.

Table 13. PS2 Frame (from PS/2 device to Controller)

Start	Data 0	Data 1	Data 2	Data 3	Data 4	Data 5	Data 6	Data 7	Parity	Stop

Start - indicates the start of a transmitted frame. This is always a '0'.

Data0-Data7 - the actual data bits. The least significant bit (Data0) is always sent first.

Parity - used for error detection. Odd parity is used. Therefore, if the number of '1's in Data0-7 is even, the Parity bit is set (1). If the number of '1's is odd, the Parity bit is cleared (0).

Stop - indicates the end of the frame currently being transmitted. This is always a '1'

Figure 7 shows the timing for this frame transmission. The PS2 Controller reads each bit in the frame on the falling edge of the clock signal.

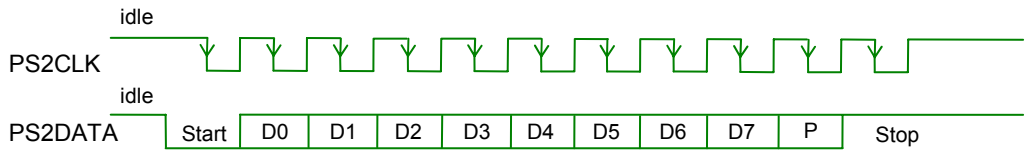

Figure 7. Transmission of data from PS/2 device to Controller

The host microcontroller, as previously mentioned, always has full control over the transmission of data to and from the PS/2 device:

- By monitoring the BUSY output, it knows the current status of the PS2 Controller – whether it is sending data to or receiving data from the connected PS/2 device
- The INT signal alerts it to any data that has been received by the PS2 Controller and is therefore ready to be read.

The host can therefore complete reception of data or start a transmission, whenever it deems is appropriate to do so.

The transmission of data from the PS/2 device to the Controller can be inhibited at any time, by the host taking the STROBE signal high. The PS2 Controller, detecting that the host wants to interrupt, takes the PS2CLK line low for at least one clock period.

When such an event arises, the Controller enters the Inhibit Transmission state. The response of the PS/2 device depends on where in the transmission the inhibit occurs:

- before reception of the Start bit – the PS/2 device has not yet begun transmission of the data. The frame of data will be buffered until the interface re-enters Idle state and the device is free to commence transmission.
- during transmission of a frame (anywhere after the first high-to-low transition of PS2CLK and before the last high-to-low transition) – the PS/2 device aborts transmission and prepares to retransmit the entire message again. The message could be a single byte scan code, in which case the same byte will be retransmitted when the interface re-enters Idle state. If the message being transmitted was composed of several bytes (for example an extended key on a keyboard

has been pressed or released), all bytes would be retransmitted, irrespective of how many bytes of the message had already been transmitted.

- After reception of the Stop bit – the PS/2 device has finished transmission of the data (constituent frames) and so retransmission is not necessary. Any new data will be buffered until the interface is in Idle state.

For a PS/2 keyboard, up to 16 bytes of key strokes can be buffered for transmission to the Controller. For a PS/2 mouse, only the current movement packet is stored for transmission.

Transmission from Controller to PS/2 device

For the Host microcontroller to send a command to the PS/2 device via the PS2 Controller, the Strobe pin must be taken high for at least one period of the external system clock signal (on the CLK input). This enables the PS2 Controller to take data on the DATAI line for transmission.

To effect transmission, the PS2 Controller must then enter the Host Send Request state. This is achieved by taking the following actions:

- the PS2CLK line is first taken low for at least one clock period (entering Inhibit Transmission state)
- the PS2DATA line is then taken low (providing the Start bit of the frame to be transmitted)
- the PS2CLK line is then released (still holding PS2DATA low).

The PS/2 device regularly checks the data and clock lines for this state and when detected, starts to generate the PS2CLK signal.

When sending data from the Controller to the PS/2 device, the PS2 frame consists of 10 fields. The order of these fields is shown in Table 14.

Table 14. PS2 Frame (from Controller to PS/2 device)

Start	Data 0	Data 1	Data 2	Data 3	Data 4	Data 5	Data 6	Data 7	Parity

Start	-	indicates the start of a transmitted frame. This is always a '0'.
Data0-Data7	-	the actual data bits. The least significant bit (Data0) is always sent first.
Parity	-	used for error detection. Odd parity is used. Therefore, if the number of '1's in Data0-7 is even, the Parity bit is set (1). If the number of '1's is odd, the Parity bit is cleared (0).

Figure 8 shows the timing for this frame transmission. The PS/2 device reads each bit in the frame on the falling edge of the clock signal.

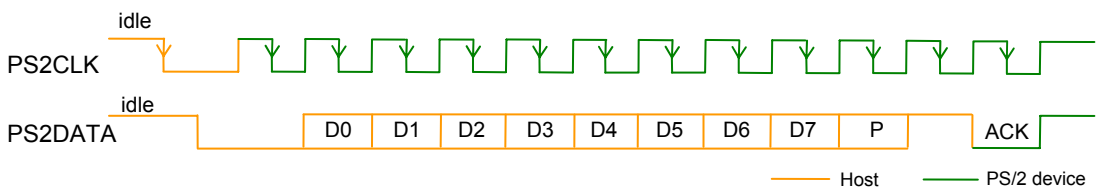

Figure 8. Transmission of data from Controller to PS/2 device

After transmission of the Parity bit, the PS2DATA line is released back to its idle state. The PS/2 device monitors this line and when it has detected no change from the idle state for at least one clock period, it takes the line low again for a single clock period. This tells the Controller that the data has been fully received by the device, i.e. an acknowledgement. Both PS2CLK and PS2DATA lines are released to their idle state. The Controller can initiate another transmission request at this point.

If the PS/2 device does not detect the release of the PSDATA line after the reception of the parity bit, it will continue to provide the PS2CLK signal. Once the PSDATA line is finally released, the device will pull it low and send the command to ‘retransmit byte’. On reception of this byte, the PS2 Controller should then retransmit the previous byte.

The Controller can inhibit transmission of data at any time before the PS/2 device sends the ACK signal, by taking the PS2CLK line low for at least one clock period.

Commands

Commands sent from a Host Microcontroller

Table 15 and Table 16 list some of the common commands that can be sent from a host microcontroller to a connected PS/2 keyboard or PS/2 mouse respectively. The commands are listed in terms of their hexadecimal codes. Note that all of the commands themselves are a single byte in length, but many of the commands require an additional byte of data to be sent by the host when determining specific options.

Table 15. Commands sent from host to PS/2 keyboard

Code	Description and Keyboard Action
EDh	This command is used to turn the keyboard's LED indicators ON or OFF. The command tells the keyboard that the host CPU wishes to change the status of one or more of the LEDs. The keyboard sends an acknowledgement (response code FAh) and then waits for another byte of data from the host, that specifies the state settings for the LEDs as follows: Bit0 – State of Scroll Lock LED Bit1 – State of Num Lock LED Bit2 – State of Caps Lock LED Bit3-7 – Not Used For each indicator, if the corresponding bit is 1, the LED is turned ON. If the bit is 0, the LED is turned OFF.
EEh	This is the echo command. Upon receipt, the keyboard transmits the echo response code (EEh).
F0h	This command is used to select the scan code set to be used for the keyboard. Every key on the keyboard has a scan code associated with it and it is this code that is sent to the host CPU when the key is pressed. The keyboard sends an acknowledgement (response code FAh) and then waits for another byte of data from the host, that specifies the particular scan code set to be used, as follows: 01h – select scan code set 1 02h – select scan code set 2 03h – select scan code set 3. If the byte 00h is sent, the keyboard responds by sending the code for the current scan set that is in use.
F2h	This command is used to read the keyboard ID. Upon receipt, the keyboard responds with the acknowledge response code (FAh), followed by two bytes that constitute the keyboard's ID – 83h and ABh.
F3h	This command is used to set the auto-repeat rate. This is the rate at which the scan code for a depressed key is sent when that key has been held depressed for a length of time exceeding a specified delay. Upon receipt of the command, the keyboard issues the

Code	Description and Keyboard Action
	acknowledge response code and then waits for another byte of data from the host, that specifies the auto-repeat rate and also the delay before auto-repeat comes into effect, as follows: Bit0-4 – specifies the repeat rate (0000 = 30 times per second; 1111 = 2 times per second) Bit5-6 – specifies the delay time (00 = 250ms; 11 = 1s) Bit7 – Not Used. Upon reception of this second byte from the host, the keyboard responds by sending the acknowledgement code again.
F4h	This command is used to enable the keyboard. Upon receipt of the command, the keyboard's output buffer is flushed and keyboard scanning (of the PSDATA and PSCLK lines) is enabled. The keyboard responds by sending the acknowledgement code (FAh).
F5h	This command is used to disable the keyboard. Upon receipt of the command, the keyboard is reset. The keyboard issues the acknowledgement code (FAh) and then keyboard scanning (of the PSDATA and PSCLK lines) is disabled. The keyboard remains in this state, waiting for another command from the host CPU.
FEh	This is the resend command and is used when the host requires the keyboard to retransmit the last byte of data sent. After acknowledging receipt of the command, the keyboard transmits the previously sent byte of data.
FFh	This command is used to reset the keyboard. After acknowledging receipt of the command, the keyboard is reset and subsequently performs its power-on Basic Assurance Test (BAT). Depending on the result of the test, the keyboard will either send the 'Passed' code (AAh) or the 'Failed' code (FCh).

Table 16. Commands sent from host to PS/2 mouse

Code	Description and Mouse Action
E6h	This command is used to set the current scaling from 2:1 to 1:1. After receipt of the command, the mouse replies with the acknowledge code (FAh) and changes the scaling to be 1:1.
E7h	This command is used to set the current scaling from 1:1 to 2:1. After receipt of the command, the mouse replies with the acknowledge code (FAh) and changes the scaling to be 2:1.
E8h	This command is used to set the resolution for the mouse. The mouse sends an acknowledgement (response code FAh) and then waits for another byte of data from the host, that specifies the resolution to be used, as follows: 00h – 1 count per millimeter 01h – 2 counts per millimeter

Code	Description and Mouse Action
	02h – 4 counts per millimeter 03h – 8 counts per millimeter After receipt of this second byte, the mouse sends the acknowledge code again and resets its movement counters
E9h	This command is used to request the current status of the mouse. Upon receipt of the command, the mouse sends the acknowledge response code (FAh), followed by a three byte status packet, structured as follows: Byte1 Bit0 – State of right mouse button (1=pressed; 0=not pressed) Bit1 – State of middle mouse button (1=pressed; 0=not pressed) Bit2 – State of left mouse button (1=pressed; 0=not pressed) Bit3 – Not Used (set to 0) Bit4 – Current scaling in use (1= 2:1; 0= 1:1) Bit5 – Data reporting state (1=enabled; 0=disabled) Bit6 – Current mode (1= Remote mode; 0= Stream mode) Bit7 – Not Used (set to 0) Byte2 – Current resolution in use Byte3 – Current sampling rate in use After sending the status packet, the mouse resets its movement counters.
EAh	This command is used to set the mouse in Stream mode. The mouse responds by sending the acknowledge code (FAh), resetting its movement counters and entering Stream mode. In this mode, a packet of data is sent every time the mouse detects movement, or when one of its buttons has changed state. Data is sent to the host, providing that data reporting is enabled. The frequency of packet transmission is determined by the sampling rate – the default being 100 samples per second.
EBh	This command is used to read sampled data from the mouse whilst it is in Remote mode. The mouse responds by sending the acknowledge response code (FAh) and then sending its current packet of movement data. The movement counters are subsequently reset.
ECh	This command is used to reset Wrap mode. The mouse responds by sending the acknowledge code (FAh), resetting its movement counters and entering the mode that it was in prior to entering Wrap mode (either Stream or Remote).
EEh	This command is used to set the mouse in Wrap mode. The mouse responds by sending the acknowledge code (FAh), resetting its movement counters and entering Wrap mode. In this mode, the mouse echoes all commands directly back to the host, without additional or further response. Two exceptions to this are the commands to reset (FFh) and reset Wrap mode (ECh). In these cases, the mouse responds as per the entries for these commands in this table.

Code	Description and Mouse Action
F0h	This command is used to set the mouse in Remote mode. The mouse responds by sending the acknowledge code (FAh), resetting its movement counters and entering Remote mode. In this mode, the inputs to the mouse (movement and buttons) are still sampled, but no packets of data are sent to the host.
F2h	This command is used to read the mouse ID. Upon receipt, the mouse responds with the acknowledge response code (FAh), followed by the code that represents its ID and distinguishes it as a standard PS/2 mouse - 00h. The movement counters are also reset at this time.
F3h	This command is used to set the sampling rate for the mouse (when monitoring its movement and button inputs). The mouse sends an acknowledgement (response code FAh) and then waits for another byte of data from the host, that specifies the sampling rate to be used, as follows: 0Ah – 10 samples per second 14h – 20 samples per second 28h – 40 samples per second 3Ch – 60 samples per second 50h – 80 samples per second 64h – 100 samples per second C8h – 200 samples per second. After receipt of this second byte, the mouse sends the acknowledge code again and resets its movement counters.
F4h	This command is used to enable data reporting when the mouse is in Stream mode. Upon receipt of the command, the mouse sends the acknowledge response code (FAh) and resets its movement counters. It continues to sample its inputs (movement and buttons) and packets of data are once again sent to the host.
F5h	This command is used to disable data reporting when the mouse is in Stream mode. Upon receipt of the command, the mouse sends the acknowledge response code (FAh) and resets its movement counters. It continues to sample its inputs (movement and buttons) but no packets of data are sent to the host. Stream mode with data reporting disabled is analogous to the mouse being in Remote mode.
F6h	This command is used to load the mouse with default values. Upon receipt of the command, the mouse sends the acknowledge response code (FAh) and then loads the following: • Sampling Rate – 100 samples per second • Resolution – 4 counts per millimeter • Scaling – 1:1 • Data Reporting – Disabled.

Code	Description and Mouse Action
	These default values are also loaded when a reset command is received from the host, or when the mouse is powered-up (as part of its Basic Assurance Test). After loading the values, the mouse resets its movement counters and enters Stream mode.
FEh	This is the resend command and is used when the host requires the mouse to retransmit the last packet of data sent. After acknowledging receipt of the command, the mouse transmits the previously sent packet of data (e.g. movement data, status information, BAT code, ID, etc).
FFh	This command is used to reset the mouse. After acknowledging receipt of the command, the mouse is reset and subsequently performs its power-on Basic Assurance Test (BAT). During this test, default values are loaded for the sampling rate, resolution and scaling, and data reporting is disabled. Depending on the result of the test, the mouse will either send the 'Passed' code (AAh) or the 'Failed' code (FCh). After the result of the power-on test is sent, the mouse sends its ID (00h). The mouse will then enter Stream mode. No movement data packets will be sent however, until the host first sends the command to enable data reporting.

Commands sent from a PS/2 device

Table 17 lists the common commands that can be sent from a PS/2 keyboard to a host microcontroller. The commands are listed in terms of their hexadecimal codes. Note that all of the commands are a single byte in length, with the exception of the keyboard ID.

Table 17. Commands sent from a PS/2 keyboard

Code	Description
00h	Error or output buffer overflow (scan code sets 2 and 3 only)
83ABh	Keyboard ID
AAh	Power-on Basic Assurance Test Passed
FCh	Power-on Basic Assurance Test Failed
EEh	Echo
FAh	Acknowledge.
FEh	Resend. Upon receipt of this code, the PS2 Controller will retransmit the previous byte
FFh	Error or output buffer overflow (scan code set 1 only)

Table 18 lists the common commands that can be sent from a PS/2 mouse to a host microcontroller. The commands are listed in terms of their hexadecimal codes. Note that all of the commands are a single byte in length.

Table 18. Commands sent from a PS/2 mouse

Code	Description
00h	Mouse ID
AAh	Power-on Basic Assurance Test Passed
FCh	Power-on Basic Assurance Test Failed
FAh	Acknowledge
FEh	Resend. Upon receipt of this code, the PS2 Controller will retransmit the previous byte.

Scan Codes

When a key on the keyboard is pressed, a code is sent to the host CPU. With the aid of ASCII look-up tables, the host can determine the function of the pressed key. The transmitted code is called a scan code and is further sub-classed as a 'make' code in the case of a key being pressed.

If a key is held down without being released, the make code for that key will be sent continuously, in accordance with the defined auto-repeat (typematic) rate. It should be noted that if more than one key is pressed and held down, typematic mode only applies to the last key pressed.

When a pressed key is released, an additional scan code is sent to the host to let it know that the key that was pressed has now been released. This additional transmitted code is called a 'break' code.

Most scan codes are a single byte in length, with the exception of some of the extended keys (e.g. SHIFT, CTRL, PAUSE). The extended keys are recognizable by the E0h prefix to their make codes.

The corresponding break code for a key is composed of the prefix byte F0h, followed by the make code for that key. Again, extended keys are the exception to this rule with the F0h byte placed after the E0h byte of the initial make code.

The PAUSE key is an exception to both standard and extended key rulings. Firstly, its make code is 8 bytes in length and starts with E1h and not E0h. Secondly, it has no break code.

The make and break codes for all keys on the PS/2 keyboard constitute the scan code set. There are three scan code sets defined but only scan code set two is recognized fully and used as the default set by all modern PS/2 keyboards.

Table 19 lists all of the keys on a standard PS/2 keyboard, along with their unique scan codes.

Table 19. PS/2 keyboard scan codes (scan code set 2)

Key	Scan Code make (break)	Key	Scan Code make (break)
ESC	76 (F076)	K	42 (F042)
F1	05 (F005)	L	4B (F04B)
F2	06 (F006)	;	4C (F04C)
F3	04 (F004)	‘	52 (F052)

Key	Scan Code make (break)	Key	Scan Code make (break)
F4	0C (F00C)	Enter	5A (F05A)
F5	03 (F003)	Shift (Left)	12 (F012)
F6	0B (F00B)	Z	1A (F01A)
F7	83 (F083)	X	22 (F022)
F8	0A (F00A)	C	21 (F021)
F9	01 (F001)	V	2A (F02A)
F10	09 (F009)	B	32 (F032)
F11	78 (F078)	N	31 (F031)
F12	07 (F007)	M	3A (F03A)
Prt Scr	E012E07C (E0F07CE0F012)	,	41 (F041)
Scroll Lock	7E (F07E)	.	49 (F049)
Pause/Break	E11477E1F014E077 (None)	/	4A (F04A)
`	0E (F00E)	Shift (Right)	59 (F059)
1	16 (F016)	Ctrl (left)	14 (F014)
2	1E (F01E)	Windows (left)	E01F (E0F01F)
3	26 (F026)	Alt (left)	11 (F011)
4	25 (F025)	Spacebar	29 (F029)
5	2E (F02E)	Alt (right)	E011 (E0F011)
6	36 (F036)	Windows (right)	E027 (E0F027)
7	3D (F03D)	Menus	E02F (E0F02F)
8	3E (F03E)	Ctrl (right)	E014 (E0F014)
9	46 (F046)	Insert	E070 (E0F070)
0	45 (F045)	Home	E06C (E0F06C)
-	4E (F04E)	Page Up	E07D (E0F07D)
=	55 (F055)	Delete	E071 (E0F071)
Backspace	66 (F066)	End	E069 (E0F069)
Tab	0D (F00D)	Page Down	E07A (E0F07A)

Key	Scan Code make (break)	Key	Scan Code make (break)
Q	15 (F015)	Up Arrow	E075 (E0F075)
W	1D (F01D)	Left Arrow	E06B (E0F06B)
E	24 (F024)	Down Arrow	E072 (E0F072)
R	2D (F02D)	Right Arrow	E074 (E0F074)
T	2C (F02C)	Num Lock	77 (F077)
Y	35 (F035)	/	E04A (E0F04A)
U	3C (F03C)	*	7C (F07C)
I	43 (F043)	-	7B (F07B)
O	44 (F044)	7	6C (F06C)
P	4D (F04D)	8	75 (F075)
[	54 (F054)	9	7D (F07D)
]	5B (F05B)	+	79 (F079)
\	5D (F05D)	4	6B (F06B)
Caps Lock	58 (F058)	5	73 (F073)
A	1C (F01C)	6	74 (F074)
S	1B (F01B)	1	69 (F069)
D	23 (F023)	2	72 (F072)
F	2B (F02B)	3	7A (F07A)
G	34 (F034)	0	70 (F070)
H	33 (F033)	.	71 (F071)
J	3B (F03B)	Enter	E05A (E0F05A)

Revision History

Date	Version No.	Revision
09-Jan-2004	1.0	New product release
04-Feb-2004	1.01	Addition of CLK_1MHZ signal; new symbol; minor modifications to text
21-Sep-2004	1.1	Addition of Wishbone version of the Controller (PS2_W)
01-Dec-2004	1.2	Schematic symbol updates
26-May-2005	1.3	Updated for Altium Designer SP4