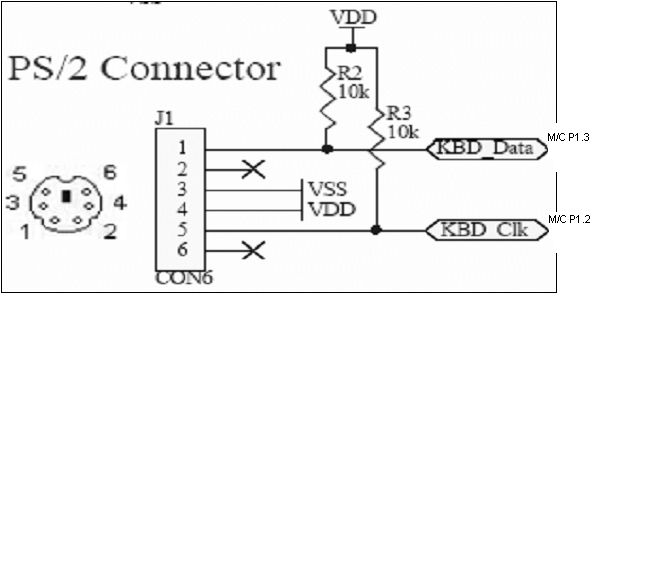
**PS/2 KEYBOARD**

****

**General description**

The PS/2 keyboard was originally an extension of the AT device.  It supported a few additional host-to-keyboard commands and featured a smaller connector.  These were the only differences between the two devices.  However, computer hardware has never been about standards as much as compatibility.  For this reason, any keyboard you buy today will be compatible with PS/2 and AT systems, but it may not fully support all the features of the original devices.

Modern PS/2 (AT) compatible keyboards

* Any number of keys (usually 101 or 104)
* 5-pin or 6-pin connector; adaptor usually included
* Bi-directional serial protocol
* Only scan code set 2 guaranteed.
* Acknowledges all commands; may not act on all of them.

Keyboards consist of a large matrix of keys, all of which are monitored by an on-board processor (called the “keyboard encoder”.)  The specific processor varies from keyboard-to-keyboard but they all basically do the same thing:  Monitor which key(s) is being pressed/released and send the appropriate data to the host.  This processor takes care of all the debouncing and buffers any data in its 16-byte buffer, if needed.  Your motherboard contains a “keyboard controller” that is in charge of decoding all of the data received from the keyboard and informing your software of what’s going on.  All communication between the host and the keyboard uses an IBM protocol.

**Scan Codes**

The keyboard’s processor spends most of its time “scanning”, or monitoring, the matrix of keys.  If it finds that any key is being pressed, released, or held down, the keyboard will send a packet of information known as a “scan code” to your computer.  There is two different types of scan codes: “make codes”and “break codes”.  A make code is sent when a key is pressed or held down.  A break code is sent when a key is released.  Every key is assigned its own unique make code and break code so the host can determine exactly what happened to which key by looking at a single scan code.  The set of make and break codes for every key comprises a “scan code set”.  There are three standard scan code sets, named one, two, and three.  All modern keyboards default to set two.

**Fig3.2.1 Scan Code1**

**Fig3.2.2 Scan Code2**

**Make Codes, Break Codes, and Typematic Repeat**

Whenever a key is pressed, that key’s make code is sent to the computer.  Keep in mind that a make code only represents a key on a keyboard–it does not represent the character printed on that key.  This means that there is no defined relationship between a make code and an ASCII code.  It’s up to the host to translate scan codes to characters or commands.

Although most set two make codes are only one-byte wide, there are a handful of “extended keys” whose make codes are two or four bytes wide.  These make codes can be identified by the fact that their first byte is E0h.

Just as a make code is sent to the computer whenever a key is pressed, a break code is sent whenever a key is released.  In addition to every key having its own unique make code, they all have their own unique break code. Fortunately, however, you won’t always have to use lookup tables to figure out a key’s break code–certain relationships do exist between make codes and break codes.  Most set two break codes are two bytes long where the first byte is F0h and the second byte is the make code for that key.  Break codes for extended keys are usually three bytes long where the first two bytes are E0h, F0h, and the last byte is the last byte of that key’s make code

**Physical Interface**

The physical PS/2 port is one of two styles of connectors:  The 5-pin DIN or the 6-pin mini-DIN.  Both connectors are completely (electrically) similar; the only practical difference between the two is the arrangement of pins.  This means the two types of connectors can easily be changed with simple hard-wired adaptors.  These costs about $6 each or you can make your own by matching the pins on any two connectors.  The DIN standard was created by the German Standardization Organization (Deutsches Institut fuer Norm).  PC keyboards use either a 6-pin mini-DIN or a 5-pin DIN connector.  Keyboards with the 6-pin mini-DIN are often referred to as “PS/2″ keyboards, while those with the 5-pin DIN are called “AT” devices (”XT” keyboards also used the 5-pin DIN, but they are quite old and haven’t been made for many years.)  All modern keyboards built for the PC is either PS/2, AT, or USB.  
The cable connecting the keyboard/mouse to the computer is usually about six feet long and consists of four to six 26 AWG wires surrounded by a thin layer of Mylar foil shielding.  If you need a longer cable, you can buy PS/2 extension cables from most consumer electronics stores.  You should not connect multiple extension cables together.  If you need a 30-foot keyboard cable, buy a 30-foot keyboard cable.  Do not simply connect five 6-foot cables together.  Doing so could result in poor communication between the keyboard/mouse and the host.

As a side note, there is one other type of connector you may run into on keyboards. While most keyboard cables are hard-wired to the keyboard, there are some whose cable is not permanently attached and come as a separate component.  These cables have a DIN connector on one end (the end that connects to the computer) and a SDL (Shielded Data Link) connector on the keyboard end.  A company called “AMP” created SDL.  This connector is somewhat similar to a telephone connector in that it has wires and springs rather than pins, and a clip holds it in place.  The pinouts for each connector are shown below:

**Connector diagram**

|  |  |  |
| --- | --- | --- |
| Male  (Plug) | Female  (Socket) | **5-pin DIN (AT/XT):**  1 – Clock 2 – Data 3 – Not Implemented 4 – Ground 5 – Vcc (+5V) |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Male  (Plug) | Female  (Socket) | | **6-pin Mini-DIN (PS/2):** 1 – Data 2 – Not Implemented 3 – Ground 4 – Vcc (+5V) 5 – Clock 6 – Not Implemented | |
|  | |  | | **6-pin SDL:** A – Not Implemented B – Data C – Ground D – Clock E – Vcc (+5V) F – Not Implemented |
|  |  |  |  |  |

**Fig3.4 Connector Diagram**  
**Electrical Interface**

Vcc/Ground provide power to the keyboard/mouse.  The keyboard or mouse should not draw more than 275 mA from the host and care must be taken to avoid transient surges.  Such surges can be caused by “hot-plugging” a keyboard/mouse                  (ie, connect/disconnect the device while the computer’s power is on.)  Older motherboards had a surface-mounted fuse protecting the keyboard and mouse ports.  When this fuse blew, the motherboard was useless to the consumer, and non-fixable to the average technician.  Most new motherboards use auto-reset “Poly” fuses that go a long way to remedy this problem.

Summary: Power Specifications  
Vcc = +4.5V to +5.5V.  
Max Current = 275 mA.

The Data and Clock lines are both open-collector with pullup resistors to Vcc.  An “open-collector” interface has two possible state: low, or high impedance.  In the “low” state, a transistor pulls the line to ground level.  In the “high impedance” state, the interface acts as an open circuit and doesn’t drive the line low or high. Furthermore, a “pullup” resistor is connected between the bus and Vcc so the bus is pulled high if none of the devices on the bus are actively pulling it low.  The exact value of this resistor isn’t too important (1~10 kOhms); larger resistances result in less power consumption and smaller resistances result in a faster rise time.  A general open-collector interface is shown below:  
**Figure 3.5: General open-collector interface.**

A line is pulled to ground by setting the corresponding pin to output, and writing a “zero” to that port.  The line is set to the “high impedance” state by setting the pin to input.

**Communication**

The PS/2 mouse and keyboard implement a bi-directional synchronous serial protocol.  The bus is “idle” when both lines are high (open-collector).  This is the only state where the keyboard/mouse is allowed begin transmitting data.  The host has ultimate control over the bus and may inhibit communication at any time by pulling the Clock line low.

The device always generates the clock signal.  If the host wants to send data, it must first inhibit communication from the device by pulling Clock low.  The host then pulls Data low and releases Clock.  This is the “Request-to-Send” state and signals the device to start generating clock pulses.

Summary: Bus States  
Data = high, Clock = high:  *Idle state.*  
Data = high, Clock = low:  *Communication Inhibited.*  
Data = low, Clock = high:  *Host Request-to-Send*

All data is transmitted one byte at a time and each byte is sent in a frame consisting of 11-12 bits.  These bits are:

* 1 start bit.  This is always 0.
* 8 data bits, least significant bit first.
* 1 parity bit (odd parity).
* 1 stop bit.  This is always 1.
* 1 acknowledge bit (host-to-device communication only)

The parity bit is set if there is an even number of 1’s in the data bits and reset (0) if there is an odd number of 1’s in the data bits.  The number of 1’s in the data bits plus the parity bit always add up to an odd number (odd parity.)  This is used for error detection.  The keyboard/mouse must check this bit and if incorrect it should respond as if it had received an invalid command.

Data sent from the device to the host is read on the *falling* edge of the clock signal; data sent from the host to the device is read on the *rising* edge*.* The clock frequency must be in the range 10 – 16.7 kHz.  This means clock must be high for 30 – 50 microseconds and low for 30 – 50 microseconds..  If you’re designing a keyboard, mouse, or host emulator, you should modify/sample the Data line in the middle of each cell.  I.e.  15 – 25 microseconds after the appropriate clock transition.  Again, the keyboard/mouse always generates the clock signal, but the host always has ultimate control over communication.

**Communication: Device-to-Host**

The Data and Clock lines are both open collector.  A resistor is connected between each line and +5V, so the idle state of the bus is high. When the keyboard or mouse wants to send information, it first checks the Clock line to make sure it’s at a high logic level.  If it’s not, the host is inhibiting communication and the device must buffer any to-be-sent data until the host releases Clock.  The Clock line must be continuously high for at least 50 microseconds before the device can begin to transmit its data.

As I mentioned in the previous section, the keyboard and mouse use a serial protocol with 11-bit frames.  These bits are:

* 1 start bit.  This is always 0.
* 8 data bits, least significant bit first.
* 1 parity bit (odd parity).
* 1 stop bit.  This is always 1.

The keyboard/mouse writes a bit on the Data line when Clock is high, and it is read by the host when Clock is low.  Figures 2 and 3 illustrate this.

**Fig3.6.1:  Device-to-host communication.**

The clock frequency is 10-16.7 kHz.  The time from the rising edge of a clock pulse to a Data transition must be at least 5 microseconds.  The time from a data transition to the falling edge of a clock pulse must be at least 5 microseconds and no greater than 25 microseconds.

The host may inhibit communication at any time by pulling the Clock line low for at least 100 microseconds.  If a transmission is inhibited before the 11th clock pulse, the device must abort the current transmission and prepare to retransmit the current “chunk” of data when host releases Clock.  A “chunk” of data could be a make code, break code, device ID, mouse movement packet, etc.  For example, if a keyboard is interrupted while sending the second byte of a two-byte break code, it will need to retransmit both bytes of that break code, not just the one that was interrupted.

If the host pulls clock low before the first high-to-low clock transition, or after the falling edge of the last clock pulse, the keyboard/mouse does not need to retransmit any data.  However, if new data is created that needs to be transmitted, it will have to be buffered until the host releases Clock.  Keyboards have a 16-byte buffer for this purpose.  If more than 16 bytes worth of keystrokes occur, further keystrokes will be ignored until there’s room in the buffer.  Mice only store the most current movement packet for transmission.

**Host-to-Device Communication**

The packet is sent a little differently in host-to-device communication…

First of all, the PS/2 device always generates the clock signal.  If the host wants to send data, it must first put the Clock and Data lines in a “Request-to-send” state as follows:

* Inhibit communication by pulling Clock low for at least 100 microseconds.
* Apply “Request-to-send” by pulling Data low, and then release Clock.

The device should check for this state at intervals not to exceed 10 milliseconds.  When the device detects this state, it will begin generating Clock signals and clock in eight data bits and one stop bit.  The host changes the Data line only when the Clock line is low, and the device reads data when Clock is high.  This is opposite of what occurs in device-to-host communication.

After the stop bit is received, the device will acknowledge the received byte by bringing the Data line low and generating one last clock pulse.  If the host does not release the Data line after the 11th clock pulse, the device will continue to generate clock pulses until the Data line is released (the device will then generate an error.)

The host may abort transmission at time before the 11th clock pulse (acknowledge bit) by holding Clock low for at least 100 microseconds.

To make this process a little easier to understand, here’s the steps the host must follow to send data to a PS/2 device:

1)   Bring the Clock line low for at least 100 microseconds.  
2)   Bring the Data line low.  
3)   Release the Clock line.  
4)   Wait for the device to bring the Clock line low.  
5)   Set/reset the Data line to send the first data bit  
6)   Wait for the device to bring Clock high.  
7)   Wait for the device to bring Clock low.  
8)   Repeat steps 5-7 for the other seven data bits and the parity bit  
9)   Release the Data line.  
10) Wait for the device to bring Data low.  
11) Wait for the device to bring Clock low.  
12) Wait for the device to release Data and Clock

Figure 3 shows this graphically and Figure 4 separates the timing to show which signals are generated by the host, and which are generated by the PS/2 device.  Notice the change in timing for the “ack” bit–the data transition occurs when the Clock line is high (rather than when it is low as is the case for the other 11 bits.)

**Fig 3.6.2:  Host-to-Device Communication.**

**Fig3.6.3:  Detailed host-to-device communication.**

Referring to Figure 4, there are two time quantities the host looks for.  (a) is the time it takes the device to begin generating clock pulses after the host initially takes the Clock line low, which must be no greater than 15 ms. (b) is the time it takes for the  packet to be sent, which must be no greater than 2ms.  If either of these time limits is not met, the host should generate an error.  Immediately after the “ack” is received, the host may bring the Clock line low to inhibit communication while it processes data.  If the command sent by the host requires a response, that response must be received no later than 20 ms after the host releases the Clock line.  If this does not happen, the host generates an error.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Keyboard Scan Codes: Set 2 \*All values are in hexadecimal  **101-, 102-, and 104-key keyboards:**   |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | **KEY** | **MAKE** | **BREAK** | ----- | **KEY** | **MAKE** | **BREAK** | —– | **KEY** | **MAKE** | **BREAK** | | A | 1C | F0,1C |  | 9 | 46 | F0,46 |  | [ | 54 | FO,54 | | B | 32 | F0,32 |  | ` | 0E | F0,0E |  | INSERT | E0,70 | E0,F0,70 | | C | 21 | F0,21 |  | - | 4E | F0,4E |  | HOME | E0,6C | E0,F0,6C | | D | 23 | F0,23 |  | = | 55 | FO,55 |  | PG UP | E0,7D | E0,F0,7D | | E | 24 | F0,24 |  | \ | 5D | F0,5D |  | DELETE | E0,71 | E0,F0,71 | | F | 2B | F0,2B |  | BKSP | 66 | F0,66 |  | END | E0,69 | E0,F0,69 | | G | 34 | F0,34 |  | SPACE | 29 | F0,29 |  | PG DN | E0,7A | E0,F0,7A | | H | 33 | F0,33 |  | TAB | 0D | F0,0D |  | U ARROW | E0,75 | E0,F0,75 | | I | 43 | F0,43 |  | CAPS | 58 | F0,58 |  | L ARROW | E0,6B | E0,F0,6B | | J | 3B | F0,3B |  | L SHFT | 12 | FO,12 |  | D ARROW | E0,72 | E0,F0,72 | | K | 42 | F0,42 |  | L CTRL | 14 | FO,14 |  | R ARROW | E0,74 | E0,F0,74 | | L | 4B | F0,4B |  | L GUI | E0,1F | E0,F0,1F |  | NUM | 77 | F0,77 | | M | 3A | F0,3A |  | L ALT | 11 | F0,11 |  | KP / | E0,4A | E0,F0,4A | | N | 31 | F0,31 |  | R SHFT | 59 | F0,59 |  | KP \* | 7C | F0,7C | | O | 44 | F0,44 |  | R CTRL | E0,14 | E0,F0,14 |  | KP - | 7B | F0,7B | | P | 4D | F0,4D |  | R GUI | E0,27 | E0,F0,27 |  | KP + | 79 | F0,79 | | Q | 15 | F0,15 |  | R ALT | E0,11 | E0,F0,11 |  | KP EN | E0,5A | E0,F0,5A | | R | 2D | F0,2D |  | APPS | E0,2F | E0,F0,2F |  | KP . | 71 | F0,71 | | S | 1B | F0,1B |  | ENTER | 5A | F0,5A |  | KP 0 | 70 | F0,70 | | T | 2C | F0,2C |  | ESC | 76 | F0,76 |  | KP 1 | 69 | F0,69 | | U | 3C | F0,3C |  | F1 | 05 | F0,05 |  | KP 2 | 72 | F0,72 | | V | 2A | F0,2A |  | F2 | 06 | F0,06 |  | KP 3 | 7A | F0,7A | | W | 1D | F0,1D |  | F3 | 04 | F0,04 |  | KP 4 | 6B | F0,6B | | X | 22 | F0,22 |  | F4 | 0C | F0,0C |  | KP 5 | 73 | F0,73 | | Y | 35 | F0,35 |  | F5 | 03 | F0,03 |  | KP 6 | 74 | F0,74 | | Z | 1A | F0,1A |  | F6 | 0B | F0,0B |  | KP 7 | 6C | F0,6C | | 0 | 45 | F0,45 |  | F7 | 83 | F0,83 |  | KP 8 | 75 | F0,75 | | 1 | 16 | F0,16 |  | F8 | 0A | F0,0A |  | KP 9 | 7D | F0,7D | | 2 | 1E | F0,1E |  | F9 | 01 | F0,01 |  | ] | 5B | F0,5B | | 3 | 26 | F0,26 |  | F10 | 09 | F0,09 |  | ; | 4C | F0,4C | | 4 | 25 | F0,25 |  | F11 | 78 | F0,78 |  | ' | 52 | F0,52 | | 5 | 2E | F0,2E |  | F12 | 07 | F0,07 |  | , | 41 | F0,41 | | 6 | 36 | F0,36 |  | PRNT SCRN | E0,12, E0,7C | E0,F0, 7C,E0, F0,12 |  | . | 49 | F0,49 | | 7 | 3D | F0,3D |  | SCROLL | 7E | F0,7E |  | / | 4A | F0,4A | | 8 | 3E | F0,3E |  | PAUSE | E1,14,77, E1,F0,14, F0,77 | -NONE- |  |  |  |  |   **Table 3.1 keyboard scan codes**  **ACPI Scan Codes:**   |  |  |  | | --- | --- | --- | | **Key** | **Make Code** | **Break Code** | | Power | E0, 37 | E0, F0, 37 | | Sleep | E0, 3F | E0, F0, 3F | | Wake | E0, 5E | E0, F0, 5E |   **Windows Multimedia Scan Codes:**   |  |  |  | | --- | --- | --- | | **Key** | **Make Code** | **Break Code** | | Next Track | E0, 4D | E0, F0, 4D | | Previous Track | E0, 15 | E0, F0, 15 | | Stop | E0, 3B | E0, F0, 3B | | Play/Pause | E0, 34 | E0, F0, 34 | | Mute | E0, 23 | E0, F0, 23 | | Volume Up | E0, 32 | E0, F0, 32 | | Volume Down | E0, 21 | E0, F0, 21 | | Media Select | E0, 50 | E0, F0, 50 | | E-Mail | E0, 48 | E0, F0, 48 | | Calculator | E0, 2B | E0, F0, 2B | | My Computer | E0, 40 | E0, F0, 40 | | WWW Search | E0, 10 | E0, F0, 10 | | WWW Home |  | E0, F0, 3A | | WWW Back | E0, 38 | E0, F0, 38 | | WWW Forward | E0, 30 | E0, F0, 30 | | WWW Stop | E0, 28 | E0, F0, 28 | | WWW Refresh | E0, 20 | E0, F0, 20 | | WWW Favorites | E0, 18 | E0, F0, 18 | |