El teclado AT-PS/2: Interfaz y funciones.

Version 1.2, 15 Noviembre de 2002

Alejandro D. L. R. ([-Ali-])
bulkman@eresmas.com

Dedicado a mi pequeña えっちゃん,
que me apoya incondicionalmente
desde tan larga distancia. ちゅううう.

28 de Octubre de 2002

Nota del autor

Las informaciones recogidas en este documento han sido obtenidas de terceros, en su mayoría incompletas en muchos aspectos o incluso contradictorias. Me he tomado la libertad de adaptar todo lo que he visto conveniente y añadir algo de mi propio saber para obtener un texto lo más coherente posible y libre de contradicciones en lo posible a la par que veraz. He de decir que la mayoría de las fuentes estaban en ingles y que yo, al menos, no he sido capaz de encontrar un documento que verse sobre el teclado AT en español. Esto y la necesidad de comprender el protocolo AT me han animado a realizar este documento. ¿Necesidad? Si, mi afición por los emuladores de maquinas recreativas y en particular el conocido MAME, me llevo hasta los encoders de teclado. Pensé si era posible construir un encoder de teclado totalmente funcional aunque más asequible en precio que los que hay en el mercado Lo primero era obtener información del teclado AT, información que no poseía. Después de investigar y ver la cantidad de información poco precisa, incompleta o contradictoria, me decidí a recopilar todo lo posible y crear un texto único que abarcara todo lo posible.

Desconozco si los textos de los que obtuve información poseen derechos de autor o similar. Mi único objetivo es obtener sin animo de lucro, un documento en español, útil y didáctico sobre el teclado AT y su protocolo. Si alguien piensa que se vulneran sus derechos que me lo haga saber para realizar los cambios pertinentes y por supuesto, si alguna persona encontrara algún error de cualquier tipo, que no dude en hacérmelo saber para subsanarlo tan pronto como sea posible en beneficio de todos. Soy un perfeccionista ☹

"Toda la información contenida en este articulo se proporciona sin ninguna garantía explícita o implícita. No garantizo la exactitud de la información ni el uso que se haga de ella: esta sólo debe ser usada para propósitos educacionales."

Gracias a Carlengue de #pic en hispano que me animo a meterme en este lio ☺

Espero que os sirva de ayuda a todos.

Alejandro D. Luna

Email: **bulkman@eresmas.com**

Índice

1. Introducción

En el siguiente articulo presentaremos una descripción del teclado, su evolución histórica, su interfaz eléctrica, el protocolo usado por el teclado para la comunicación con un host, códigos de teclado, inicialización, compatibilidad entre los teclados, etc. Pero centrándonos en el teclado en si, no como se programa el teclado usando el sistema operativo.

1.1. Historia del teclado

Desde la invención de la plataforma IBM PC, el teclado ha servido como dispositivo primario de entrada de datos al ordenador y actualmente ya forma parte de la arquitectura estándar de los PC, al incorporarlos estos en su interface.

Hay muchos teclados diferentes, no sólo en lo que se refiere a los fabricantes de los mismos sino también en lo referente a la localización de las teclas. Los hay suecos, franceses, ingleses, alemanes y claro está españoles. Pero en el ámbito del PC sólo existen tres teclados estandarizados en cuanto a lo que se refiere a la cantidad de teclas y los códigos scan[1] que estas producen. Los símbolos grabados en cada tecla dependen de la versión especifica del país, pero el orden fundamental de las teclas, números, caracteres especiales, teclas de función y de cursor se mantienen siempre.

Durante todo este tiempo, la plataforma del IBM PC ha soportado tres tipos de teclado: el teclado XT, el teclado AT y el teclado MF-II.

Al comienzo de su desarrollo, la plataforma IBM soportaba el teclado **PC/XT** (1981) que disponía de 83 teclas diferentes. Problemas de diseño hacían poco ergonómico a este teclado. Por supuesto, los teclados XT no son compatibles con plataformas de PC posteriores, como la AT y PS/2. Ya que la plataforma XT esta obsoleta no la describiremos en detalle en este documento:

- 83 teclas.
- Conector 5-pin DIN.
- Protocolo serie unidireccional.
- Trama: 2 bit de start, 8 bits de datos, 1 bit make/break y 1 bit de stop. El bit make/break se usa para saber si la tecla esta pulsada/no-pulsada
- Usa el scan code set 1.
- No acepta comandos del host.
- Reset hardware usando las líneas del conector.

[1] Ver punto 6 “Reconocimiento de teclas”

Teclado XT 83 teclas

Los errores de diseño se eliminaron con la introducción del teclado **PC/AT** (1984), que poseía 84 teclas: las teclas [Enter] y [Shift] se hicieron más grandes con lo que permitían un acceso más fácil, pero como el tamaño del teclado no había cambiado otras teclas se hicieron más pequeñas para compensar el aumento de aquellas ([Bloq Despl] y [+] en el teclado numérico). Y se añadió la tecla [Sys] –que más tarde se rebautizó como [Pet Sys]-.

- 84 teclas.
- Conector 5-pin DIN.
- Protocolo serie bidireccional sincrono.
- Trama: 1 bit de start, 8 de datos, 1 de paridad y 1 de stop (1 ack opcional).
- La tecla pulsada/no-pulsada se conoce mediante el scan code enviado.
- Usa el scan code set 2.
- Acepta hasta 8 comandos del host.
- Reset software a través de un comando software del host.

Teclado AT 84 teclas

Como vemos, estos teclados ya no son compatibles con los sistemas XT y viceversa.

Hubo un tiempo que los teclados traían un interruptor en la parte de atrás que nos permitía seleccionar el modo de funcionamiento (AT o XT). En estos teclados, si se usaba el modo XT desde el interruptor, el protocolo bidireccional era eliminado y el teclado funcionaba exactamente como un teclado XT original aun siendo un teclado AT.

En 1987 IBM desarrolló el teclado **MF-II** (MultiFunción II o teclado extendido) a partir del AT y por tanto compatible con el estándar AT pero no con el XT. El número romano en su nombre se lo debe a su precursor, el teclado MF-I para PC/XT, aunque no era idéntico al estándar de teclado PC/XT y sólo se utilizo un breve periodo de tiempo.

Las características del MF-II hacen que sea actualmente el estándar mas difundido:

- Usa el mismo interfaz de teclado que el AT con 101 teclas.
- Añade un bloque teclas adicionales con la teclas de cursor, de forma que el teclado numérico se pudiera usar permanentemente para introducir números.
- Las teclas de función se desplazaron.
- Se añadieron dos teclas de función adicionales [F11] y [F12] por compatibilidad con los teclados de Mainframes y otros ordenadores grandes de IBM.
- Las teclas [Alt] se movieron al lado de la [Barra de Espacios] para alcanzarlas mejor. Además se añadió la tecla [AltGr] cuya activación es equivalente a la pulsación simultanea de [Ctrl] y [Alt]
- Se añadieron tres LED que indican el estado de las teclas [Num Lock], [Caps] y [Scroll Lock].
- Soporta un tercer scan code set (set 3). Por defecto usa el scan code set 2.
- Permite 17 comandos desde el host

Hay dos versiones diferentes del teclado MF-II, que solo se diferencian en la añadidura de una tecla: la versión US con 101 teclas y la versión europea con 102 teclas. Esta tecla fue colocada a la derecha de la tecla [Shift], lo que motivo que [Shift] fuera dividida en su tamaño y vuelve a ser tan pequeña como lo era en el viejo teclado PC/XT. Para colmo, esta nueva tecla tampoco aporta gran cosa al teclado del usuario europeo pues su función se puede obtener con otras teclas. Hoy en día vemos teclados con mas de 102 teclas, pero todos usan el mismo protocolo AT para la comunicación y solo se diferencian en los nuevos códigos añadidos para las nuevas teclas.

Teclado AT 101 teclas US

Teclado AT 102 teclas EU

Como indicamos, estos teclados (AT, MF-II) ya no compatibles con el estándar XT pues el mayor numero de teclas, el control de los LED, etc. motivó adaptar protocolo del teclado y ampliar el número de comandos para soportar las mejoras. Por ejemplo, frente a los 9 bits de datos que usaba el XT pues no controlaba la paridad, ahora se usan 10 bits pues si se controla la paridad, etc.

Los teclados **PS/2** son básicamente iguales a los MF-II. Sus únicas diferencias son el conector que usan (mini-DIN de 6 pin) mas pequeño que el AT y soportan nuevos comandos. Pero la comunicación se realiza usando el protocolo AT (incluso los ratones PS/2 usan el mismo protocolo de comunicación).

Actualmente la denominación de teclado AT o teclado PS/2 solo hace referencia al tipo de conector. Es difícil saber que comandos u opciones soporta un determinado teclado. Por ejemplo, un teclado con conector estilo PS/2 puede soportar solo 7 comandos, parcialmente otros y el resto simplemente confirmarlos sin hacer nada. Pero podemos encontrar otros teclados con conector estilo AT que soporten cada comando del teclado original PS/2 (incluso mas). Por ello **es importante tratar los teclados modernos como compatibles**, no como estándares.

Un teclado compatible AT-PS/2 tiene las características siguientes:

- Cualquier número de teclas (normalmente más de 101).
- Conector de 5-pin o 6-pin.
- Protocolo bidireccional sincrono.
- Sólo se garantiza soporte para el scan code set 2.
- Confirma (ACK) todos los comandos, aunque puede que no haga nada.

Este documento se centra en el teclado MF-II y su protocolo AT ya que con su descripción cubrimos la totalidad de teclados actuales que usan el estándar AT (PS/2 y compatibles). En los últimos años empiezan a aparecer teclados USB, pero su interfaz es bastante más compleja que la AT y se sale fuera del ámbito de este documento.

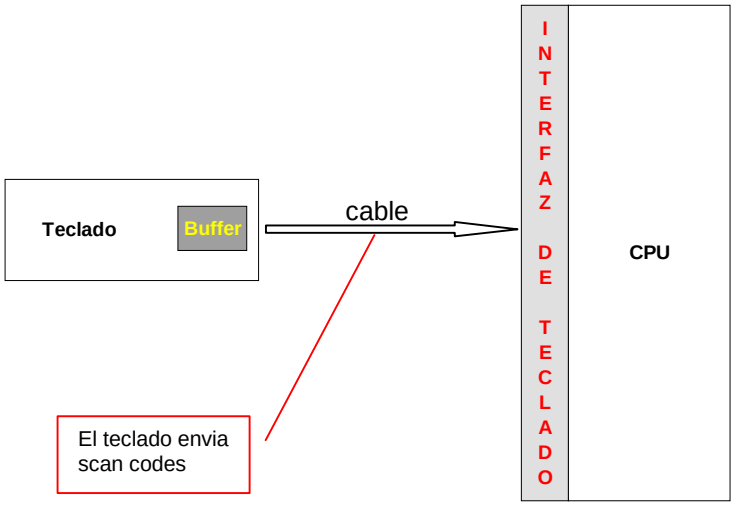
1.2. El subsistema del teclado.

En todos los teclados podemos distinguir dos partes:

- El teclado con su cable
- Una interfaz de teclado que une el teclado con el ordenador.

El teclado esta continuamente rastreando una matriz de teclas en espera de una tecla presionada o soltada por el usuario. Cuando ocurre tal evento, el teclado asigna un único byte (o secuencia de bytes) llamado scan code a dicha pulsación y lo transmite hacia el PC por el cable. La interfaz de teclado del PC recibe cada scan code y después de comprobar la paridad de los datos transmitidos, solicita una retransmisión si ocurrió un error o pasa el dato hacia el microprocesador del PC. Si el microprocesador está ocupado cuando se genera el scan code, el interfaz de teclado indicará al teclado que el procesador esta ocupado. El teclado entonces esperará hasta que el procesador le dé la señal de que puede

transmitir. Mientras el procesador esta ocupado, el teclado puede seguir generando scan codes que almacena en un buffer interno en el mismo teclado (no confundir con el buffer del host).



Una vez el teclado envía un scan code y el interfaz de teclado comprueba que es valido, este scan code es convertido a un código interno del host, después de ser pasado al procesador del host: cuando se recibe un scan code valido, la circuiteria del interfaz de teclado genera una interrupción en el procesador del host. Si el procesador es capaz de atender la interrupción, ejecutará la rutina de interrupción. Es dentro de esta rutina de interrupción donde los scan codes son convertidos al conjunto de caracteres interno de la CPU (ASCII, etc..).

Como hemos indicado antes, el subsistema del teclado esta compuesto por dos partes, el teclado y la interfaz del teclado:

a) El teclado

Realiza las siguientes funciones:

- Captura las pulsaciones del usuario sobre la matriz de teclas.
- Codifica las pulsaciones en scan codes (ver punto 9).
- Transmite los scan codes a través de su cable hacia el interface del teclado en el ordenador.

Para implementar estas funciones, el teclado IBM fue diseñado sobre un único microcontrolador (MCU). Los teclados que soportaban el IBM XT, estaban diseñados usando el MCU 8048 de Intel. Los teclados AT y MF II, usan otros microcontroladores.

b) El interfaz de teclado

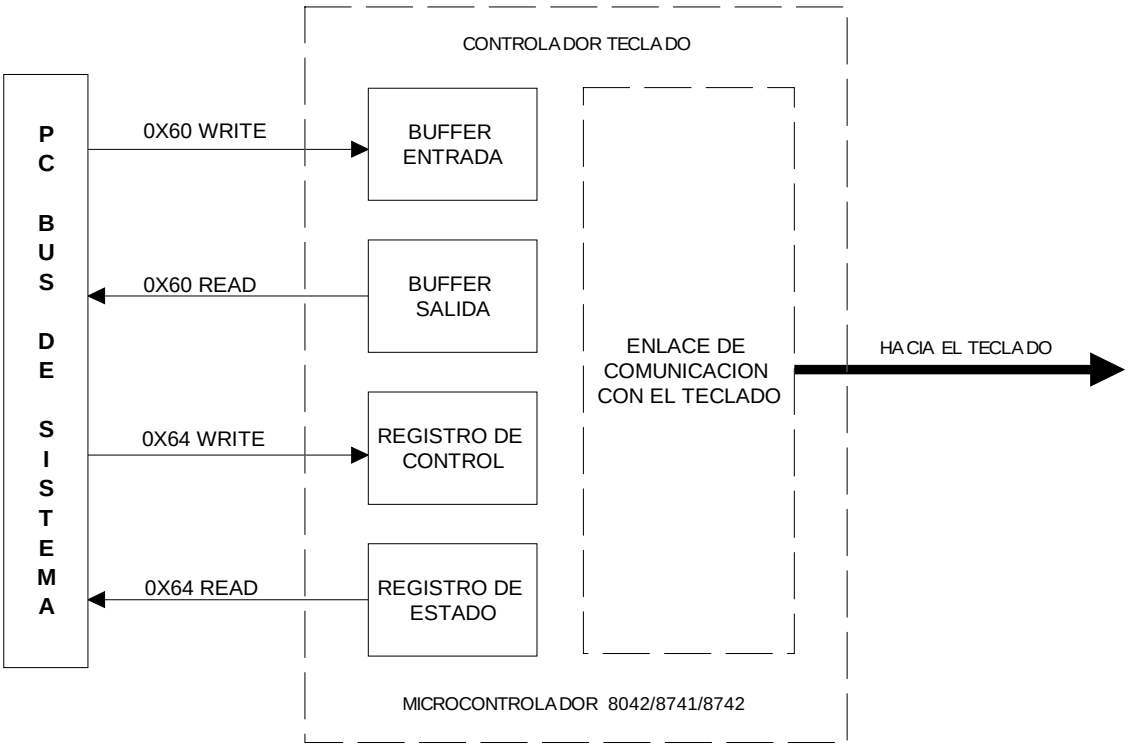
Realiza las siguientes funciones:

- Suministra energía al teclado.
- Trasmite comandos del host al teclado.
- Recibe las respuestas del teclado ante los comandos enviados.
- Recibe scan codes desde el teclado.
- Proporciona una interfaz para el bus de sistema del ordenador.

El diseño del interfaz de teclado integra todas estas funciones en un único microcontrolador que sirve como controlador del interface. Las primeras interfaces de teclado AT se diseñaron usando el microcontrolador Intel 8042. En los nuevos AT y PS/2, se usan los Intel 8641 y 8742.

A su vez, el interfaz del teclado puede ser dividida en dos partes:

- Enlace de comunicación con teclado
- Interface del bus de sistema del PC



El enlace de comunicación del teclado no solo transmite y recibe datos desde el teclado, sino que también comprueba los datos recibidos para detectar errores de transmisión y controla el flujo de datos desde el teclado hacia el host.

El interface del bus de sistema del PC es el punto donde el microprocesador del PC interacciona con el teclado. El host configura y monitoriza el teclado a través del interface, enviando comandos de teclado directamente hacia el teclado o escribiendo comandos del controlador de teclado en la interfaz del controlador.

El interfaz de teclado consiste en un buffer de entrada, un buffer de salida y los registros de control y estado del controlador de teclado. Los buffers de entrada y salida están mapeados en la dirección 0x60 en el espacio de entrada/salida (I/O) del PC. Al buffer de entrada se accede mediante escrituras en la dirección 0x60, mientras que las lecturas a la dirección 0x60 acceden al bus de salida.

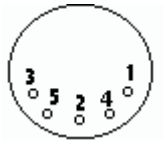
Los registros de control y estado del controlador de teclado están mapeados en la dirección 0x64 del espacio I/O del PC. Al registro de estado se accede leyendo en la dirección 0x64, mientras que al registro de control se accede escribiendo en esa misma dirección. El host lee las respuestas del teclado a los comandos del host y los scan codes, desde el buffer de salida. El host envía comandos al controlador de teclado, escribiendo en el registro de control. Para los comandos del controlador que requieren datos adicionales además del byte del comando, el host escribe los datos necesitados en el buffer de entrada. El host monitoriza la transmisión y recepción de datos del interface de teclado, mediante la lectura del registro de estado del controlador de teclado.

2. Características mecánicas y eléctricas

2.1. Características mecánicas

El teclado y el interfaz de teclado están físicamente conectados a través del cable del teclado. Este cable esta formado por un grupo de 5 cables con una malla de protección contra interferencias (shield) unido a un conector circular de 5 o 6 pine de tipo DIN por un extremo, mientras que el otro esta directamente conectado a la circuiteria interna del teclado.

Hay dos tipos de conectores: el de 5-pin DIN usado en el estándar AT y el de 6-pin mini-DIN que usado en el estándar PS/2:

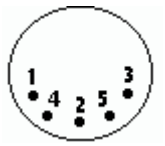


DIN hembra

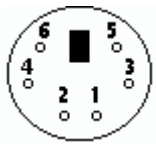
1- Clock
2- Data
3- N/C
4- GND
5- VCC

mini-DIN macho

1 – Data
2 – N/C
3 – GND
4 – VCC
5 – Clock
6 – N/C



DIN macho



mini-DIN hembra

El chasis metálico va conectado a la malla de proteccion del cable o un su defecto a GND.

2.2. Características eléctricas

De las anteriores señales, las señales vcc y gnd se usan para alimentación del teclado y las suministra el interfaz de teclado en el host. La intensidad que el interfaz es capaz de suministrar al teclado varía de una placa base a otra, pero debe ser suficiente para alimentar un teclado normal (del orden de cientos de mA; un valor típico sería 300 mA). Las señales clock y data son usadas para la comunicación entre host y teclado en colector abierto[2].

Línea	Descripción.
VCC	Alimentación del teclado. **+5 V**.
GND	Masa del teclado. **0 V**.
CLOCK	Línea de reloj (CLK, CTS). **TTL**.**Colector abierto**.
DATA	Línea de datos (RxD,TxD,RTS). **TTL**. **Colector abierto**.
N/C	No Conectado. Reset en algunos teclados viejos

[2] En colector abierto (“open drain”) sólo tenemos dos posibles estados: “0” lógico o desconectado. No tenemos “1” lógico por lo que hay que obtenerlo mediante circuiteria externa.

Nota histórica:

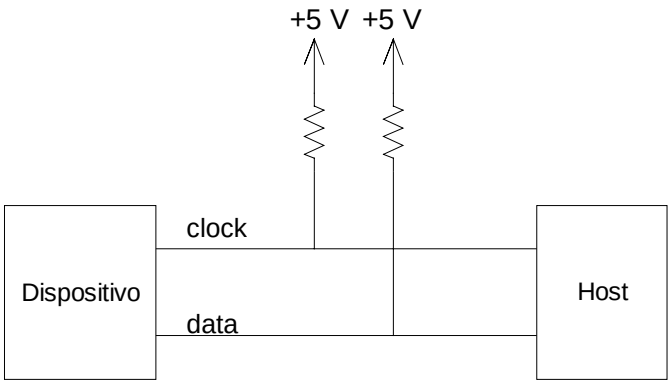
> *La documentación AT moderna indica que el pin 3 es "reservado", por lo que el teclado debe proveer su propio Reset a petición del host. Pero en el teclado AT **original** de IBM, el pin 3 era una línea de Reset real y los teclados de IBM lo necesitaban en esa época (los teclados AT originales no funcionarán en algunos viejos clónicos a causa de esto). Por eso en los teclados AT con DIN, el pin 3 no esta conectado.*

Todos los niveles eléctricos son TTL:

CLOCK / DATA	MIN.	MAX.
V_{out} low	0	0,7 V
V_{out} high	2,4 V	5,25 V

Ya que la comunicación se realiza sobre las líneas clock y data que son de colector abierto, necesitamos resistencias de polarización o de pull-up para obtener niveles lógicos altos. Para tener valores lógicos altos con una alimentación de +5V, los valores típicos son de 5-10 kΩ para las resistencias de pull-up. Si queremos poner una línea a nivel alto (+5 V) sólo hay que dejarla libre –poniendo a alta impedancia la salida de esa línea en el dispositivo/host-; esto hara que se ponga a nivel alto a través de las resistencias de pull-up. Si queremos poner una línea a nivel bajo (0 V) debemos forzarla a ese nivel[3] colocando la salida del dispositivo/host a 0 V.

Esquema típico de una configuración a colector abierto con resistencias pull-up:



En este caso cuando el dispositivo deja la línea clock libre (alta impedancia) esta se pone a un nivel alto (+5 V) debido a la circulación de corriente electica a traves de la resistencia de pull-up.

[3] El protocolo AT usa lógica positiva; es decir, un nivel alto se corresponde con un "1" lógico (+5 V), y un nivel bajo se corresponde con un "0" lógico (0 V o GND).

3. Interface y comunicación

3.1. Descripción general

Los primeros teclados diseñados para el IBM PC/XT permitían únicamente la transmisión unidireccional asincrona de scan codes desde el teclado hacia el host. El host poseía un mínimo control sobre el teclado (la señal reset –pin 3- formaba parte de la interfaz del teclado).

Las mejoras de los teclados AT requerían un mayor control del host sobre la configuración y funcionamiento del teclado: esto motivo un rediseño del teclado, del interfaz de teclado y del desarrollo de un protocolo para manejar el envío de datos desde el teclado al host. Este **protocolo bidireccional sincrono** define un conjunto de especificaciones para las señales de datos y reloj para las transferencias desde el teclado al host y otro conjunto distinto para las transferencias desde el host al teclado. Es decir, el comportamiento del protocolo difiere según el sentido de la transferencia.

Además el protocolo define un conjunto de comandos que el host puede enviar al teclado para controlar su estado o cambiar su configuración. Este conjunto dota al host de comandos para hacer un reset del teclado, habilitar o deshabilitar (bloquear) el teclado, en el caso de algunos teclados, cambiar el scan code set activo del teclado.
El protocolo también define el conjunto de respuestas que el teclado debería transmitir como respuesta a un comando del host.

El protocolo también da prioridad a las transferencias del host hacia el teclado sobre las transferencias del teclado hacia el host. En cualquier caso, si el teclado esta trasmitiendo un scan code o una respuesta al host y el host desea enviar un comando al teclado, el teclado liberará el control de las líneas de clock y data para permitir que el host tome el control. El host transmitirá el comando al teclado. Luego el teclado responderá al host si es necesario, y una vez acabada la respuesta, volverá a retransmitir el dato que fue interrumpido.

El protocolo define que en la comunicación **siempre es el teclado el responsable de generar la señal de reloj (clock) en cualquier caso** tanto si el teclado envía datos, como si los recibe. Si el host quiere enviar datos debe solicitar al teclado que genere la señal de reloj, como veremos mas tarde.

La máxima frecuencia de funcionamiento[4] es de 33 kHz, aunque normalmente los teclados conectados al host operan entre 10-20 kHz. La frecuencia de trabajo la decide el teclado y tiene un **duty-cycle del 50%**. Es recomendable usar frecuencias entorno a 15 kHz, lo que significa que el reloj debería estar a nivel alto durante unos 30 microsegundos, y a nivel bajo durante otros 30 microsegundos (16,6 kHz).

[4] Este punto no esta comprobado. Es un resumen de diversos documentos contrastados .

La comunicación se realiza una trama de 11 (o 12) bits:

1 bit START (siempre 0)
8 bits de datos, LSB primero
1 bit paridad impar
1 bit STOP (siempre 1)
1 bit ACK (siempre 0) sólo enviado si es el host quien manda datos al teclado

El bit de paridad usa paridad impar. En paridad impar el número de 1 de los datos mas el bit de paridad debe ser impar, o lo que es lo mismo, el bit de paridad es 1 si el número de unos en los 8 bits de datos es par, y es 0 si el número de unos es impar. Su uso es para detección de errores.

El bit ACK (ACKnolegdment) es usado para confirmar la recepción de los datos (no confundir con el comando ACK que veremos mas adelante). Este bit solo es usado cuando el host envía datos al teclado, no cuando el teclado envía datos al host: **el teclado no necesita confirmar de los datos que envía**.

Durante las transmisiones, el bus puede estar en uno de los siguientes tres estados:

- Libre (idle) : Si las líneas clock y data están a nivel alto, no hay actividad en el bus.
- Inhibido (inhibit) : Si el host pone la línea clock a nivel bajo, indica que quiere enviar datos al teclado e inhibe el bus.
- Solicitud de envío (Request-To-Send o RTS): Si el host pone la línea data a nivel bajo y clock a nivel alto, indica que quiere enviar un comando u otra información al teclado.
  En algunos textos se habla también de contención de línea o "line contention", cuando la inhibición que se produce una vez que el teclado ha empezado a transmitir, únicamente para hacer distinción de la inhibición antes de iniciar la transmisión ya que el comportamiento del teclado es ligeramente diferente.

3.2. Envío de datos hacia el host

Cuando el teclado desea enviar datos al host, debe esperar que el bus este libre (clock=1 y data=1). Mientras esta condición no ocurre el teclado sigue procesando las pulsaciones y las almacena en un buffer interno.

Para ello el teclado comprueba periódicamente si el bus esta inhibido o hay una solicitud de envío (RTS):

- Si el bus esta inhibido (clock=0) el teclado espera a que este libre, procesando y almacenando en su buffer interno las pulsaciones de teclas realizadas en ese tiempo.

- Si el bus esta en RTS (clock=0, data=1) el teclado almacena las pulsaciones de teclas en el buffer, pasa a recibir los datos del host y a procesarlos.

Una vez que el teclado detecta la condición de bus libre, debe garantizar que espera al menos 50 µs desde que acaba el estado de inhibición hasta que pone clock a nivel bajo para comenzar el bit de start (el teclado puede esperar mas si quiere; la liberación de la línea clock por parte del host **no es un comando** para que el teclado empiece a transmitir, sino una señal de que tiene permiso para transmitir tan pronto como pueda hacerlo).

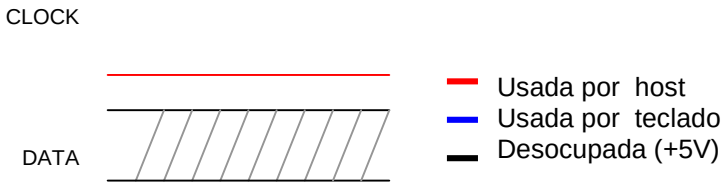


Fig. 1: bus inhibido

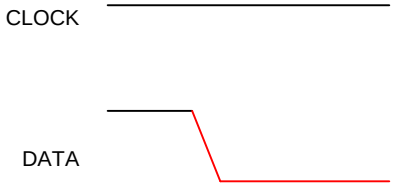


Fig 2: bus en RTS

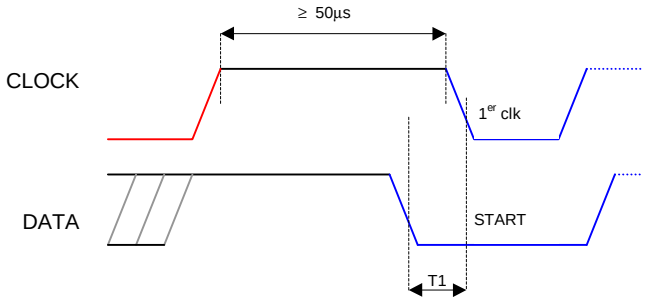
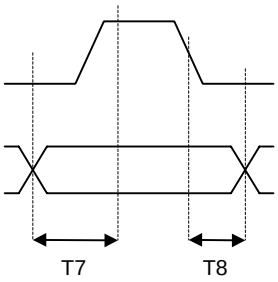


Fig 3: bus libre y teclado comienza transmisión

El teclado transmite 1 byte de datos por la línea de datos, usando 11 pulsos de reloj, en el siguiente orden:

1º bit de start (0)
2º 8 bits de datos **comenzando por el LSB**
3º bit de paridad impar
4º bit de stop (1).

El host muestrea los datos en el flanco de bajada del reloj, por lo que el teclado cambia los datos en el nivel bajo del reloj. Los datos deben ser validos al menos 5 µs (t7) antes del flanco de subida del reloj y 5 µs (t8) después del flanco de bajada del reloj.



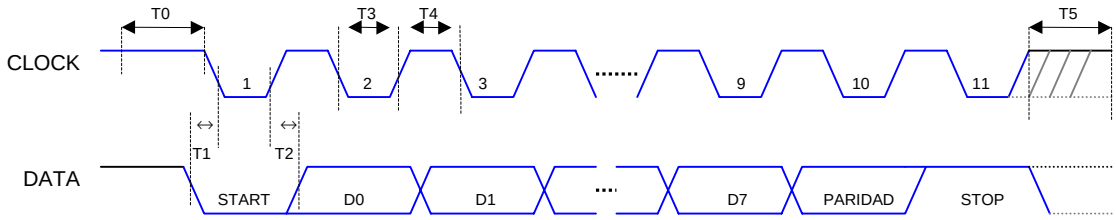
Durante la transmisión el teclado debe comprobar si la línea clock sigue libre (clock=1) al menos cada 60 µs, ya que el host puede inhibir la transmisión en cualquier momento colocando clock a nivel bajo durante al menos 60 µs (esta inhibición una vez comenzada la transmisión es denominada en algunos textos como "line contention" o contención de línea).

Si se produce una inhibición una vez iniciada la transmisión y antes de llegar al flanco de subida del 10º pulso de reloj (bit de paridad) la transmisión del byte se cancela, el teclado deja libre el bus (clock=1 y data=1) y reenviara el byte interrumpido tan pronto como pueda y el bus este libre de nuevo.

Si la inhibición ocurre después del flanco de subida del 10º pulso de reloj (bit de stop) la transmisión **se da por completa** y el host debe aceptar los datos.

Una vez enviado el bit de stop el teclado libera el bus. El host puede inhibir el bus (extendiendo el 11º pulso de reloj a un nivel bajo) entre 0-50 µs (t5). Esto es una señal para el teclado de que el host esta ocupado y no es capaz de aceptar otra transmisión del teclado. El host liberara la línea una vez que ha procesado el dato transmitido y esta preparado para aceptar otra transmisión. Durante este tiempo el teclado debe esperar.

Diagrama de tiempos:



PARAMETRO DE TIEMPO		MIN/MAX
T0	Tiempo de espera mínimo desde que bus esta libre hasta que el teclado empieza a generar clk.	50/- µs
T1	Tiempo desde de una transición de data hasta el flanco de bajada de clk	5/25 µs
T2	Tiempo desde el flanco de subida de clk hasta una transición de data	5/T4-5 µs
T3	Tiempo inactivo de clk	30/50 µs
T4	Tiempo activo de clk	30/50 µs
T5	Tiempo después del clk 11 que el teclado debe esperar en previsión de que el host pueda inhibir el bus	0/50 µs

Ejemplo de transmisión:

1) Esperar clock = 1
2) Retraso de 50μs
3) Clock todavía está a 1?
   No (bus inhibido) → Ir a 1
4) Data = 1?
   No (hay petición de envío –RTS-) → Abortar transmisión, leer byte del host y procesarlo.
5) Enviar bit start (0)
6) Enviar 8 bits datos
7) Enviar bit paridad

   } Después de enviar cada bit, comprobar si clock = 0 (contencion de línea) Si clock = 0 abortar.

8) Enviar bit stop (1)
9) Libera bus
10) Retraso de 50μs → Durante este tiempo el host puede inhibir

Como el host puede inhibir el bus durante la transmisión hay que detectar esta condición. En la practica se puede realizar la comprobación cada vez que enviamos un bit de datos (salvo stop). Observando los tiempos, vemos que un pulso de reloj puede durar un mínimo de 30+30 μs hasta un máximo de 50+50 μs; es decir, entre 60-100 μs y el host mantiene la línea a nivel bajo al menos durante 100 μs, con lo que en el peor caso -si usamos un periodo de reloj alto de 50+50 μs- no habrá problema en detectar la condición después de cada pulso de reloj.
Si estamos enviado el bit de stop (pulso de reloj 11), aun cuando ocurra la inhibición se da transmisión por correcta.
En este ejemplo, el teclado siempre espera el máximo tiempo (50 μs) al final de una transmisión para dar tiempo al host para inhibir el bus si lo necesita.

La secuencia para enviar un bit de datos seria:

a) set/reset data
b) retraso 15μs
c) pone clock = 0
d) retraso 30 μs
e) pone clock = 1
e) restraso15 μs

Al entrar en la rutina el reloj ya estaba a nivel alto por la anterior llamada; cambiamos el dato y esperamos 15 μs, luego ponemos clock a 0 durante 30 μs, y luego volvemos a ponerlo a 1. De esta forma, clk está a nivel alto 15, a bajo 30 μs, y luego a alto 15 μs (se reparte la duración del nivel alto entre el final del anterior pulso y el comienzo del siguiente, obteniendo como resultado 15+15 = 30 μs). El resultado es un pulso de reloj de 30 μs a nivel alto y otros 30 μs a nivel bajo.

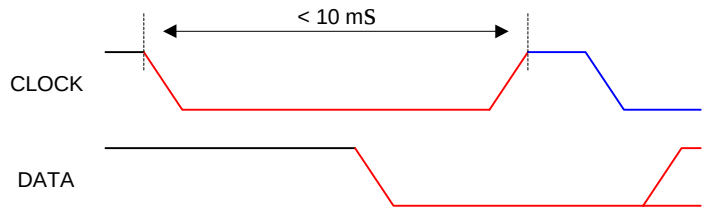
3.3. Recepción de datos desde el host

Si el host quiere enviar datos, primero comprueba si el teclado esta transmitiendo. Si es así y no ha llegado al 10º pulso de reloj, el host puede abortar la transmisión. En caso contrario, el host debe aceptar los datos.

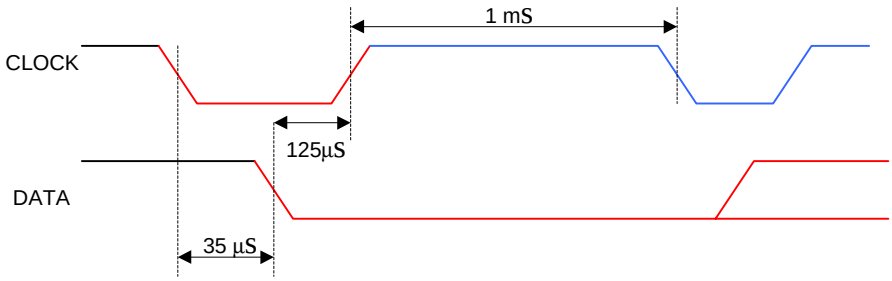
Una vez que el host tiene datos para enviar, realiza la siguiente secuencia:

1) Pone la línea clock a nivel bajo al menos durante 60 µs (otras referencias hablan de 100 µs)
2) Activa "request-to-send", poniendo la línea data a nivel bajo.
3) Cuando el host esta preparado para empezar a enviar datos, libera la línea clock

Con esto el host evita que el teclado envíe datos al mismo tiempo que él intenta enviar datos al teclado. El teclado debe comprobar la situación de RTS en intervalos que no excedan de 10 ms; es decir, el tiempo desde que el teclado detecta que hay un RTS y clock pasa a nivel alto hasta el teclado comienza a generar la señal de reloj no debe ser mayor de 10 ms. Cuando la línea clock se ponga a nivel alto, el teclado comenzará a generar la señal de reloj y a capturar los datos.
Hay que destacar que el host no envía un bit de start, simplemente se limita a poner la línea data a nivel bajo antes de dejar libre la línea clock. Aunque funciona como un bit de start, no es un bit de start.



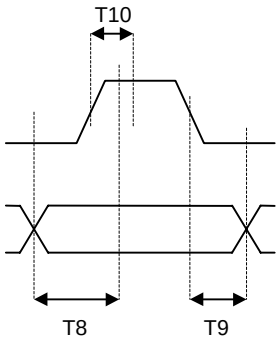
Ejemplo: Secuencia típica de inicio de una transmisión de datos



El host inicia una transmisión poniendo a nivel bajo la línea clock. Aproximadamente 35 µs después, el host pone a nivel bajo la línea data. Esta secuencia de pasos, indica al

teclado que el host quiere transmitir un dato. La línea clock es liberada por el host aproximadamente 125 μs después del flanco de bajada de la señal data (se cumple el mínimo de 60 μs exigido por la condición de RTS). El teclado pone la línea clock a nivel alto en ese instante. Aproximadamente 1 ms después del flanco de subida del reloj, comienza la transferencia de datos. Durante este tiempo el host mantiene la línea de datos a nivel bajo, comenzando la transmisión cuando el teclado pone a nivel bajo la señal de reloj. Esto sirve como bit de start. Vemos que se cumplen los tiempos mínimos del protocolo.

El host cambia los datos en el nivel bajo del pulso reloj. Los datos deben estar estables al menos 1 μs (t8 = -1μs) antes del nivel alto del reloj y estables hasta el flanco de bajada del reloj (t9 ≥ 0 μs); es decir, el dato debe estar estable al menos 1 μs antes del nivel alto del pulso de reloj y permanecer estable hasta el flanco de bajada. El teclado captura el dato tan pronto como el reloj este a nivel alto, normalmente entre 5-25 μs después del flanco de subida de cada pulso de reloj (en la practica esta captura se podría realizar también en el flanco de subida si el host ya tiene estable el dato).



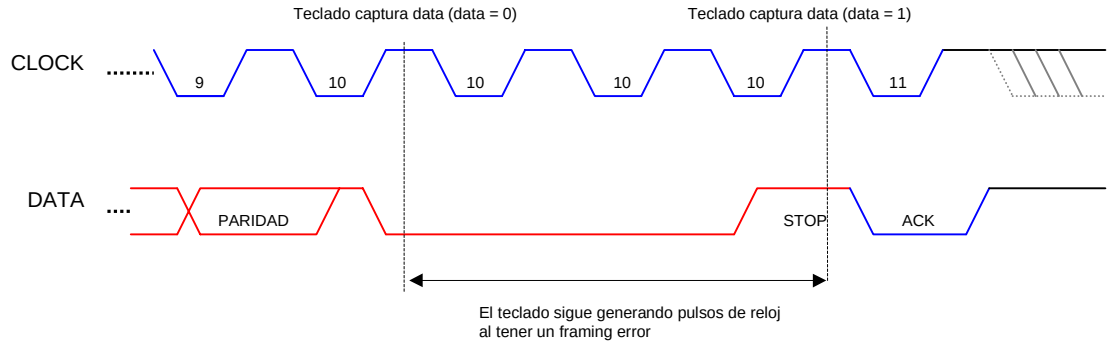
T10 : Tiempo desde una transición de CLK de bajo a alto hasta que el teclado captura el dato, usado para cronometrar cuando el teclado muestrea la línea data.

Una vez recibidos los 8 bits de datos y la paridad -pulso 10º de reloj-, el host pone la línea data a nivel alto (bit stop) dejando libre la línea clock y espera el siguiente pulso de reloj (el 10º) del teclado. El teclado comprobará entonces si hay un nivel alto en la línea de datos (bit stop):

- Si data=1, entonces el teclado confirma la recepción poniendo la línea data a nivel bajo (bit ACK) y da un pulso mas de reloj (el 11º).

- Si data=0 después del pulso 10º de reloj, ocurrió un error de trama o "framing error". El teclado continuará generando pulsos de reloj hasta que la línea de datos este a nivel alto (llega bit stop). Luego envía el bit de ACK para acabar la transmisión. Mas tarde enviará un **comando** Resend para indicar el error ocurrido.

Situación de error de trama:

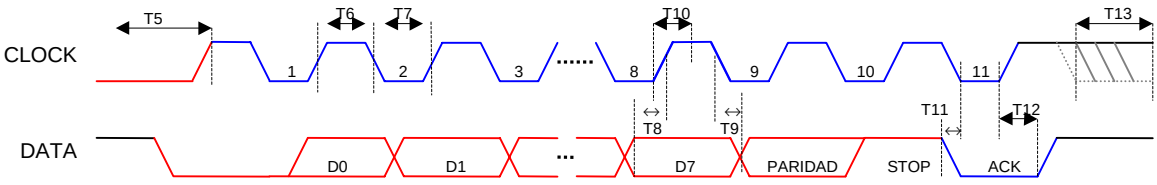


Una vez el teclado ha puesto el bit ACK en la línea, el teclado libera la línea de data después de liberar la línea clock.

Después que el teclado ha enviado el bit de ACK el host puede inhibir el bus entre 0-50 µs mientras procesa los dato, indicando al teclado que no transmita inmediatamente.

Inmediatamente después de recibir un byte el teclado realiza una comprobación de la paridad del dato recibido. Si hay un error de paridad o el dato recibido no es reconocido como un comando valido, el teclado solicitará una retransmisión del byte enviando un comando Resend (0xFE) al host.

Diagrama de tiempos completo:



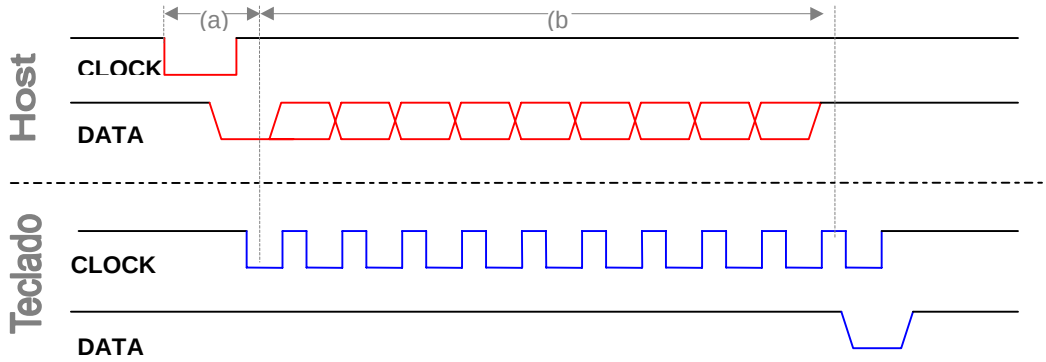
PARAMETRO DE TIEMPO		MIN/MAX
T5	Tiempo en el que el teclado comprueba que el host quiere enviar (data=0 y clock libre). El teclado debe comprobar al menos cada T5 esta condición.	10/- ms
T6	Tiempo inactivo de clk	30/50 µs
T7	Tiempo activo de clk	30/50 µs
T8*	Tiempo en que los datos deben ser estables antes del nivel alto de clk	-1/- µs
T9	Tiempo en que los datos deben ser estables después del flanco de bajada de clk	0/- µs
T10	Tiempo desde una transición de CLK de bajo a alto hasta que se muestrea el dato, usado para cronometrar cuando el teclado muestrea data	5/T7-5 µs
T11	Tiempo desde la transición del bit de control ACK hasta el flanco de subida de clk	5/- µs
T12	Tiempo desde el flanco de bajada de clk hasta la transición del bit de control ACK	5/- µs
T13	Tiempo después del clk 11 que el host puede inhibir el bus para asegurar que el teclado no empieza otra transmisión y poder procesar el dato actual	0/50 µs

*T8 indica que el dato debe ser valido antes de 1 µs después del flanco de subida de clk.

Nota 1: No se muestran los tiempos en las transiciones de datos antes y después de los flancos clk (al contrario cuando el teclado enviaba donde teníamos T1=5 µs y T2=5 µs mínimo). Para enviar el bit de control si hay que respetarlos (T11 y T12) pues el bit de control ACK lo ***envía el teclado*** *como confirmación de la recepción y por tanto esta sujeto a los tiempos de un envío del teclado.*

Nota 2: El tiempo de T10 en otras referencias es de 30/50 pero no tiene mucho sentido pues si por ejemplo, si T10=30 µs y T7=30 µs, esto quiere decir que debemos leer el dato justo cuando empieza el flanco de bajada del reloj, lo cual a todas luces no parece muy adecuado aunque para otros valores si tenga sentido. Debido a la falta de información al respecto, optamos por una solución de un ejemplo comercial, donde se captura el dato entre 5/25 µs después del flanco de subida, lo que se puede traducir en T10 = 5/T7-5 µs.

Otras consideraciones de tiempo:



(a) El tiempo que tarda el teclado en comenzar a generar la señal de reloj después que el host ha puesto a nivel bajo la señal de reloj, no debe ser mayor que 10 ms (15 ms según otras fuentes). Es decir, el teclado debe empezar la comunicación antes de 10 (o 15 ms)

(b) El tiempo que tarda en enviarse un byte completo no debe ser mayor que 2 ms.

Si algunos de estos limites no se cumplen, el host genera un error y no tomará en cuenta el dato recibido, solicitando su reenvío con un comando Resend.

Inmediatamente después que el host ha enviado el dato y lo ha recibido el teclado, el host puede poner la línea de clock a nivel bajo para inhibir la comunicación. Si el comando enviado por el host necesita de una respuesta, **dicha respuesta debe ser enviada por el teclado antes de 20 ms** (25 ms según otras fuentes) desde que el host libera la línea de clock Si la respuesta no se recibe en este tiempo, no es valida o tiene un error de paridad, el host genera un error (solicita reenvío con comando Resend).

Ejemplo de recepción del teclado:

a) Comprobar si data = 0 al menos cada 10 ms
b) Si data ha sido puesta a 0 por el host, leer un byte desde el host:
   1- Espera que clock = 1
   2- Todavía data = 0?
      No → un error ocurrió y aborta la recepción.
   3- Lee 8 bits de datos
   4- Lee 1 bit de paridad
   5- Lee 1 bit de stop
   6- Todavía data = 0?
      Si →sigue generando pulsos de reloj hasta que data = 1, entonces genera un error
   7- Escribe bit de ACK
   8- Comprueba bit de paridad
      Si hay un error en la paridad genera un error (Resend)
   9- Retraso 45 μs para dar tiempo al host para inhibir la siguiente transmisión (junto con los 5 μs de la escritura –ver abajo- tenemos 50 μs)

La lectura de un bit seria:

a) retraso 15 μs
b) pone clock a 0
c) retraso 30 μs
d) pone clock a 1
e) lee línea data

Y la escritura del bit de control:

a) retraso 10 μs
b) pone data a 0
c) retraso 5 μs
d) pone clock a 0
e) retraso 30 μs
f) pone clock a 1
g) retraso 5 μs
h) pone data a 1

4. Encendido e inicialización del teclado: Power-On y Reset

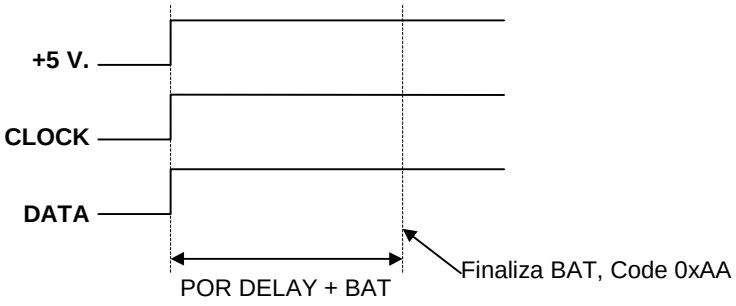
El teclado genera una señal de "Power-On-Reset" (POR) cuando recibe alimentación eléctrica. La duración del POR es de un mínimo de 150 ms y un máximo de 2000 ms desde que se aplica la alimentación al teclado por primera vez (en algunas referencias se habla entre 300-1000 ms).

Una vez que ocurre el POR el teclado realiza una comprobación y una calibración de sus sistemas –RAM y ROM- denominada "Basic Assurance Test" (BAT), que dura un mínimo de 300 ms y un máximo de 500 ms (tiempo adicional al del POR). Al comenzar el BAT el teclado habilita sus tres LED, los apaga una vez completado el BAT y envía un código al host indicando si hubo o no error:

- Si no hubo error en el BAT, envía código de finalización con éxito 0xAA seguido del ID 0x00 y el teclado comienza el escaneo de las teclas.
- Si fallo algo, envía código de error 0xFC[5] e ID 0x00 y el teclado queda en espera de comandos.

Este proceso es el mismo que ocurre cuando llega un comando de Reset (0xFF) desde el host, pero al ser un reset software no hay etapa de POR y solo realiza el BAT.

De esta forma, los códigos de finalización son enviados entre 450 ms y 2500 ms después del POR+BAT ([150-2000] + [300-500] ms), y entre 300 ms y 500 ms después de que un comando Reset es recibido (en algunas referencias se habla de un POR+BAT de [300-1000] ms).



Muchos teclados ignoran sus líneas clock y data hasta mientras se realiza el BAT y solo las atienden una vez que se ha enviado el código de finalización. Por lo tanto una condición de inhibición del bus (clock=0) puede que no evite que el teclado envíe el código de finalización del BAT.

[5] Otras fuentes indican que se puede enviar también 0xFD, pero no lo detallan. En cualquier caso, el código 0xFC parece ser lo normal.

Hemos visto que cuando el teclado AT es inicializado por primera vez envía un 0xAA si ha pasado el test o 0xFC/0xFD si hubo un fallo. Pero antes de hacer esto, envía un flujo continuo de 0xAA con la paridad incorrecta: una vez que el ordenador envía un 0xFE para indicar que hay un error de paridad, el teclado para el envío de los AA erróneos y envía un 0xAA correcto o 0xFC/0xFD según el resultado del BAT.
Este mecanismo fue usado como un parche rápido en el firmware del teclado para cronometrar los reset perdidos (el teclado siempre acaba el reset antes que el ordenador se inicialice, luego el ordenador podría perder algún AA/FD/FC mientras se está inicializando y de esta forma no pierde nada). Este mecanismo no se usa si se recibe un comando Reset, pues el ordenador ya esta inicializado.

Oficialmente hasta que el host no recibe el código 0xAA, 0x00 no debe intentar comunicarse con el teclado. Como hemos dicho muchos teclados ignoran el estado del bus, así que una petición de RTS podría ser ignorada mientras el teclado realiza el BAT, pero otros dispositivos con protocolo AT sí podrían atender el bus mientras realizan el BAT. En principio es aconsejable seguir el proceso visto.

Una vez realizado el BAT, el teclado pone por defecto los siguientes valores:

- Typematic delay: 500 ms.
- Typematic rate: 10.9 cps.
- *Scan code set: 2.
- *Set all keys typematic/make/break.

*Estos valores por defecto pueden variar en algunos teclados o estar incorporados en su hardware.

5. El buffer de teclado

El teclado posee un buffer interno en forma de cola (FIFO) de 17 bytes donde se guardan los scan codes de las teclas pulsadas hasta que el host este preparado para recibirlos. De estos 17 bytes, los 16 primeros están reservados para los scan codes y el 17º se usa para la condición de desbordamiento (buffer overrun).

Si se intenta meter más de 16 bytes en el buffer ocurre un error de desbordamiento y se inserta en el byte 17 un código de overrun (0x00 ó 0xFF según el scan code set usado)[6] y se ignoran todas las teclas pulsadas mas tarde. El código es enviado al host una vez que este alcanza el final del buffer.

Una vez ocurre el desbordamiento, si se siguen presionando teclas sin que el buffer se vacíe, los scan codes de esas pulsaciones se pierden.

Cuando el teclado pueda enviar datos, los bytes del buffer se envían como en una operación normal.

[6] 0x00 para scan code set 2 y 3, 0xFF para scan code set 1. Ver punto 7, “comandos “

Si una pulsación genera una secuencia de múltiples bytes (por ejemplo las teclas extendidas, como [Pause], [Scroll Lock], etc.), la secuencia entera debe caber en el espacio disponible del buffer, en otro caso, la pulsación es descartada y ocurre un error de desbordamiento.

Los códigos de respuesta (ACK) no ocupan posiciones del buffer. Esto es importante, pues si el teclado confirma un envío del host enviándole un ACK y luego el host solicita el reenvío (comando Resend) del último dato, el teclado debe enviar el último ACK no el ultimo byte del buffer.

Nota: No confundir este buffer interno del teclado con el buffer que el controlador del teclado(8042) tiene en el host. Si algún dispositivo hiciera las veces de host, no podría enviar el comando de buffer-overrun pues está reservado para el teclado. En su lugar solo podrá inhibir la línea para evitar que el teclado siga llenado su buffer y así darle tiempo para procesar los datos recibidos.

6. Reconocimiento de teclas: scan codes.

Antes de hablar de los comandos, necesitamos conocer algunos aspectos del funcionamiento del teclado para entender como los comandos influyen en ellos. Veremos como el host puede reconocer que tecla ha sido pulsada, cuando ha sido soltada, cuando tiene que repetir una tecla si esta se mantiene pulsada un tiempo, etc.

Debemos recordar que el objeto de nuestro estudio es el teclado AT pues el que se usa actualmente y no tiene sentido hablar de otras plataformas. Esto se evidencia cuando queremos usar un scan code set del teclado XT pero en un teclado AT, es decir, que el teclado AT emule uno XT.

6.1. Scan codes

El procesador del teclado esta la mayor parte del tiempo rastreando la matriz de teclas. Si halla una tecla presionada, que se deja de presionar o se que mantiene presionada un tiempo, el teclado detecta que tecla es y la codifica en un código de rastreo (scan code) que es enviado al host.

Un scan code sólo representa una tecla en un teclado, no representa el carácter impreso sobre la tecla; es decir, no hay una relación entre el “scan code” y el código ASCII de la tecla que representa. Es la CPU del host la encargada de convertir los “scan codes” al código ASCII correspondiente, no el teclado –como vimos en la introducción del texto -.

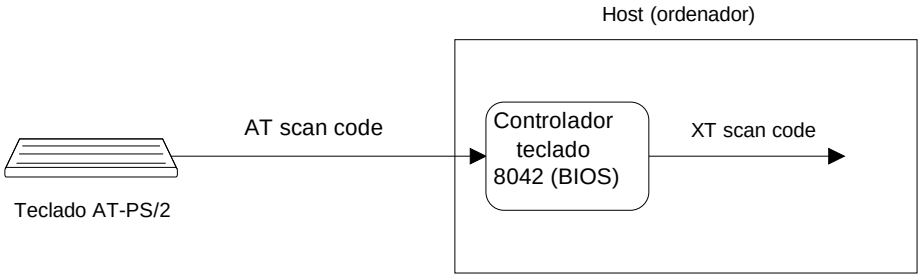
El conjunto de todos los scan codes de un teclado se conoce por “scan code set” (conjunto de códigos de rastreo). Hay tres “scan code sets” o modos estandarizados:

- Scan code set 1 o XT: soportado por los teclados XT originales
- Scan code set 2 o AT: soportado por todos los teclados modernos AT y compatibles
- Scan code set 3 o MF-II: opcional. Soportado por algunos teclados PS/2 modernos.

Todos los teclados modernos usan por defecto el scan code set 2. Algunos teclados PS/2 modernos soportan los tres sets, pero no necesariamente soportan los conjuntos 1 y 3 y solo garantizan con el 2.

Por desgracia este tema de los scan code sets y como identificar el tipo de teclado es un poco confuso por la falta de información exacta.

Nosotros solo veremos los scan codes que envía el hardware del teclado: estos scan codes no tienen porque ser los mismos que un programa en el host pueda leer, pues por razones de compatibilidad con software realizado para sistemas viejos, el controlador de teclado del host puede convertir los scan codes AT recibidos desde el teclado, a scan codes XT antes de dárselos a la CPU. Todavía este mecanismo de conversión de la BIOS persiste en algunos sistemas, por lo que el código leído desde el buffer del teclado del host no tiene porque ser necesariamente el mismo código que el teclado transmitió.



6.2. Make codes, break codes.

Los scan code set están formado por dos tipos de códigos: make code y break code.

Con estos make/break codes el host es capaz de saber cuando se pulsa o se suelta una tecla: cuando se pulsa una tecla el teclado envía el make code de dicha tecla y cuando la tecla es soltada envía el break code.
Normalmente el make code coincide con el scan code de la tecla de forma que cada tecla tiene un make code único y un break code único[7] asociados a ella.

En general los scan codes están formados por un único byte, pero también existen teclas especiales (“extended keys”) que requieren un tratamiento especial. Estas teclas suelen ser las teclas de función o de control, como [Ctrl], [F1] a [10], [Shift], etc. para así distinguirlas de las teclas normales. Los scan codes asociados a este tipo de teclas dependerá del scan code set activo en cada momento, por ejemplo:

[7] La tecla “Pause/Break” no tiene un break code en los scan code sets 1 y 2. Si la tecla es pulsada se envía su make code, y si se suelta no se envía nada.

- En los scan codes sets 1 y 2: las teclas normales tienen un make code propio que coincide con su scan code, mientras que las teclas extendidas son identificadas por su primer byte que siempre es E0h.
- En el scan code set 3: **todas** las teclas tienen un make code propio que coincide con su scan code por lo que cada tecla extendida también tiene un make code único.

Aunque hay una gran cantidad de scan codes, salvo excepciones, hay ciertas relaciones entre los make y break codes que evitan que usemos intensivamente tablas lookup para hallarlos.

Por ejemplo en el scan code set 2, en general se cumplen las siguientes reglas:

- El make code de una tecla coincide con su scan code.
- El make code de una tecla especial, siempre empieza por E0 seguido por el scan code de la tecla.
- Los break codes de las teclas están formados por 2 bytes, el primero siempre es F0 y el segundo el scan code de esa tecla.
  Por esto, las teclas normales suelen tener 2 bytes (F0,scan code), y las teclas extendidas 3 bytes (F0,E0,scan code).

Ejemplo:

Tecla	Make Code (set 2)	Break Code (set 2)
A	1C	F0 1C
7	3D	F0 3D
Kp 7	6C	F0 6C
F2	06	F0 06
LSHIFT	12	F0 12
←	E0 75	F0 E0 75
RCTRL	E0 14	F0 E0 14
Pause	E1 14 77 E1 F0 14 F0 77	nada

Tecla	Make Code (set 3)	Break Code (set 3)
A	1C	F0 1C
7	3D	F0 3D
Kp 7	6C	F0 6C
F2	0F	F0 0F
LSHIFT	12	F0 12
←	61	F0 61
RCTRL	58	F0 58
Pause	62	F0 62

Observamos ligeras diferencias entre un set y otro (azul) y como los brak code llevan un prefijo F0 (rojo). Mas adelante profundizaremos en las particularidades de cada scan code set.

Veamos la secuencia de scan codes que el teclado enviaría al host al presionar la tecla [A] mayúscula usando el set 2:

1. Pulsar [LShift], por ejemplo.
2. Pulsar [a].
3. Soltar [a].
4. Soltar [LShift].

Los scan codes generados serian: make code del [LShift] (0x12), make code de [a] (0x1C), break code de [a] (0xF0, 0x1C) y break code de [LShift] (0xF0, 0x12). Estos scan codes serán metidos en el buffer del teclado y enviados al host en la siguiente secuencia: 0x12, 0xF0, 0x1C, 0xF0, 0x1C, 0xF0, 0x12.

En general: si se pulsa una tecla, el teclado envía el make code de la tecla. Si se pulsa una segunda tecla mientras la primera sigue pulsada, se envía el make code de la segunda. Si la segunda tecla se deja de pulsar antes que soltar la primera, la primera tecla es desactivada. Para reactivar la primera tecla, antes debe de dejarse de pulsar.
Si dos o más teclas son presionadas simultáneamente, todas son validas y se envían todos los make codes (no se generan errores).

En capítulos más adelante se muestran los scan code set y las diferencias existen en el funcionamiento de los make y break codes. Recordemos que todos los teclados AT soportan el set 2, - pero no necesariamente los sets 1 y 3- aunque actualmente la mayoría de teclados comerciales soportan tres modos de operación y la elección de un modo u otro depende de la asignación de códigos de rastreo que deseemos.

6.3. Retardo y ratio de repetición (Typematic delay/rate)

Si mantenemos pulsada una tecla, el teclado envía el scan code correspondiente repetidamente hasta que se deje de pulsar. Aquí hay dos parámetros importantes: el factor de repetición (typematic rate) y el factor de retraso (delay rate).

- Delay rate: tiempo que pasa desde que se envía el **primer** scan code, hasta que se comienza a enviar el segundo mientras se mantiene la tecla pulsada. Sirve para detectar cuando la tecla se pulsa un tiempo suficiente para considerar necesario repetir el scan code.
- Typematic rate: es el numero de scan codes (medidos en caracteres por segundo o cps) que serán enviados pasado el retraso.

El delay oscila entre 250 ms y 1000 ms, y el rate entre 2.0 cps hasta 30.0 cps. Estos parámetros se pueden modificar con el comando “Set Typematic Rate/Delay” (0xF3).

Por ejemplo, supongamos que pulsó la tecla [a]. En ese instante aparece una “a” en la pantalla. Si mantenemos pulsada la tecla, al cabo de un tiempo el teclado decidirá que es

necesario repetir esa tecla y volverá a enviar una segunda letra “a”: este tiempo es el delay. Y a partir de aquí se seguirán enviando “a” a la velocidad marcada por el rate (no hay ya retraso entre una tecla y otra pues el tiempo lo marca la tasa de cps especificada).

Con excepción de la tecla [Pause], todas las teclas que se mantienen pulsadas durante ese tiempo, se repetirán. El make code es repetido hasta que la tecla ese deja de pulsar. En el caso de que exista mas de una tecla presionada, únicamente la ultima tecla presionada será repetida. La repetición se para si la última tecla se deja de pulsar –aunque las otras estén presionadas- o si otras teclas han sido presionadas en ese tiempo.

7. Comandos

Hemos visto que el teclado es capaz de enviar scan codes (make/break) de las teclas pulsadas. El host al mismo tiempo puede enviar al teclado comandos para indicarle alguna operación y el teclado puede responder a los comandos que el host envía. Muchos de estos comandos a su vez necesitan de argumentos adicionales para seleccionar alguna determinada opción.

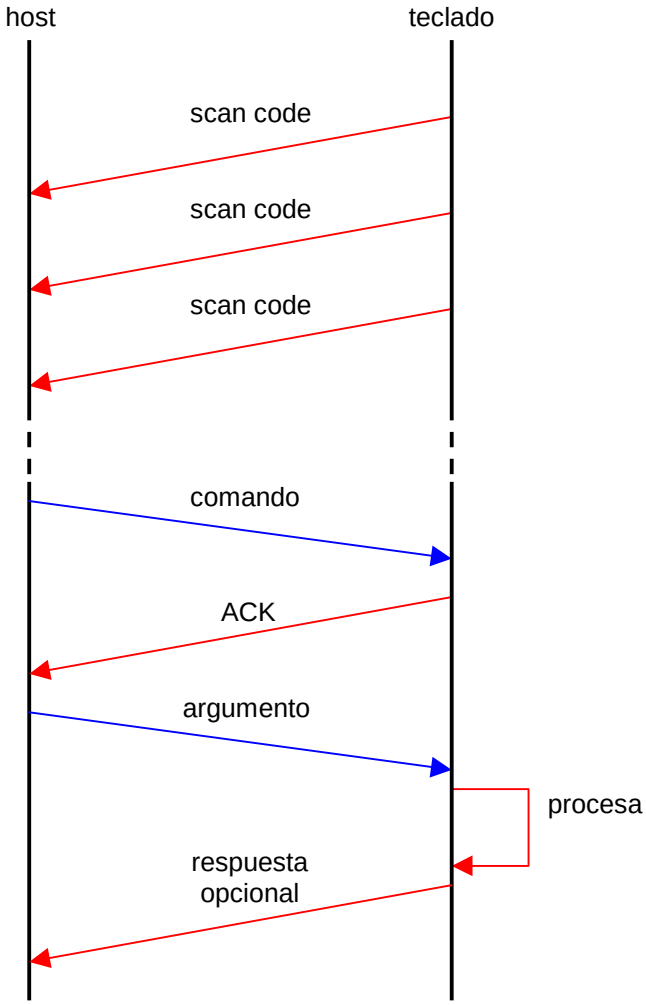
7.1. Comandos del host

Comando	Valor Hex
Set/Reset Status Indicators	ED
Echo	EE
Invalid Command	EF
Select Scan Code Set*	F0
Invalid Command	F1
Read ID*	F2
Set Typematic Rate/Delay	F3
Enable (clear buffer)	F4
Default Disable	F5
Set Default	F6
Set All Keys - Typematic* - Make/Break* - Make* - Typematic/Make/Break*	F7 F8 F9 FA
Set Key Type - Typematic* - Make/Break* - Make*	FB FC FD
Resend	FE
Reset	FF

* Originariamente estos comandos sólo están disponibles en teclados MF-II o PS/2. Los teclados que no los soporten envían la señal de reconocimiento ACK y no realizan ninguna acción.

Los comandos pueden ser enviados al teclado en cualquier momento y el **teclado debe responder en 20 ms como máximo** (algunas referencias indican 25 ms) excepto cuando realiza un test de comprobación (BAT) o ejecuta un comando de Reset.
En general cuando el teclado recibe un comando contesta con un comando de confirmación de recepción (comando ACK) y a continuación se prepara para recibir bytes desde el host con los argumentos del comando, procesándolos. El host sólo envía un comando, espera una respuesta ACK –hay excepciones- y luego envía los argumentos del comando si este los necesita.

Ejemplo de comunicación



7.1.1. Set/Reset Status Indicators (Hex ED)

Los tres LED indicadores de estado son accesibles para el host a través de este comando. El teclado activa o desactiva los LED cuando recibe la secuencia del comando correcta. La secuencia del comando comienza cuando el host envía al teclado el comando 0xED: el teclado contesta con un comando ACK, interrumpe el rastreo de teclas y espera un byte de argumento que envía el host. El byte de argumento tiene la siguiente forma:

7							0
0	0	0	0	0	CL	NL	SL

SL = Scroll Lock LED (0-off, 1-on)
NL = Num Lock LED (0-off, 1-on)
CL = Caps Lock LED (0-off, 1-on)

Los valores se pueden activar o desactivar en cualquier combinación. Una vez actualizado el estado de los LED, si el teclado estaba desbloqueado (“Enabled”) continua el rastreo de teclas.

Si en lugar del byte de argumento se recibe otro comando, la ejecución del comando “Set/Reset Status Indicators” se interrumpe: no se realiza ningún cambio en los LED, el nuevo comando es procesado y continua el rastreo de teclas (no se bloquea en esta ocasión).

Inmediatamente después del Power-On-Reset los LED se ponen por defecto al estado de desactivado (Off). Si se reciben los comandos “Default Disable (0xF5)” o “Set Default (0xF6)”, los LED permanecen en el mismo estado que estaban antes de recibir dichos comandos.

7.1.2. Echo (Hex EE)

Si el teclado recibe este comando responde con otro echo (0xEE) y, si estaba habilitado, continua el rastreo de teclas. El echo es una ayuda al diagnóstico.

7.1.3. Invalid Command (Hex EF, F1)

Estos comandos no están soportados. Si el teclado recibe alguno, no envía un comando de ACK sino un comando Resend y continua el rastreo de teclas.

7.1.4. Select Scan Code Set (hex F0)

Este comando fuerza al teclado a seleccionar uno de los tres posibles scan code sets. Al recibir el teclado este comando, responde con un ACK, limpia el buffer y typematic key (si hay alguna activa). Luego el host envía un byte de argumento. El teclado responde con un ACK y selecciona uno de los scan codes set según el valor del byte de argumento: 0x01, scan code set 1; 0x02, scan code set 2; 0x03, scan code set3. Un valor de argumento de 0x00 provoca que el teclado responda con un ACK y envíe un byte al host indicando qué scan code set esta usando en ese instante.

Después de establecer el nuevo scan code set, el teclado continúa con el rastreo de teclas donde lo interrumpió.

7.1.5. Read ID (Hex F2)

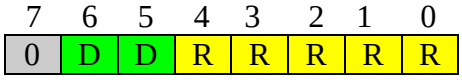
Este comando solicita información de identificación (ID) del teclado. El teclado responde con un ACK, interrumpe el rastreo de teclas y envía al host 2 bytes de identificación (0x83AB): primero el byte LSB 0xAB y segundo el byte MSB 0x83. El segundo byte debe ser enviado tras al primero no mas tarde de 500 μs desde el final del envío del primer byte. Después de enviar el segundo byte, el teclado continua el rastreo de teclas.

7.1.6. Set Typematic Rate/Delay (Hex F3)

Este comando cambia la factor de repetición (typematic rate) y el factor de retardo (delay rate). Al recibir el comando, el teclado responde con un ACK, interrumpe el rastreo de teclas y espera a que el host envíe el byte de argumento con los valores de rate y delay. El teclado responde con otro ACK al byte de argumento, cambia el rate y delay a los valores indicados por el host y continua el rastreo (si estaba previamente habilitado).

El calculo del rate y del delay se realiza con tablas ya prefijadas, aunque sus valores no son aleatorios, sino que obedecen a unas formulas matemáticas a partir de los bits del byte de argumento, siempre con una tolerancia del ± 20%:

El bit 7 siempre esta a 0. Los bits 6 y 5 forman un número D que indica el delay medido en ms. Los bits 4, 3, 2, 1 y 0 se usan para calcular el rate medido en «caracteres por segundo (cps)»: los bits 4 y 3 forman un numero B, y los bits 2,1,0 otro número A:



$$delay: \ (DD+1)*250 \pm 20\% \ \ ms$$

$$rate: \frac{1}{(8+A)*(2^{B})*0.00417} \pm 20\% \ \ cps$$

Rate

Bit 4-0 hex	Typematic Rate	Bit 4-0 hex	Typematic Rate
00	30.0	10	7.5
01	26.7	11	6.7
02	24.0	12	6.0
03	21.8	13	5.5
04	20.0	14	5.0
05	18.5	15	4.6
06	17.1	16	4.3
07	16.0	17	4.0
08	15.0	18	3.7
09	13.3	19	3.3
0A	12.0	1A	3.0
0B	10.9	1B	2.7
0C	10.0	1C	2.5
0D	9.2	1D	2.3
0E	8.6	1E	2.1
0F	8.0	1F	2.0

Delay

Bit 5-6 bin	Delay (ms)
00	250
01	500
10	750
11	1000

Por defecto, el sistema establece un parámetro 2B, es decir:

Typematic rate = 10.9 cps ± 20%
Typematic delay = 500 ms ± 20%

Si el teclado en lugar de recibir el parámetro con valores de rate/delay recibe otro comando (bit 7 a 1), dejará inalterados los valores de typematic, procesará dicho comando, y **permanecerá bloqueado** hasta que se le habilite el teclado (comando Enable 0xF4).

7.1.7. Enable (Hex F4)

Este comando **reanuda el funcionamiento bloqueado** por un comando anterior. Al recibir el teclado este comando, responde con un ACK, vacía el buffer de salida, borra la última typematic key y continua el rastreo de teclas.

Es usado para **desbloquear el teclado**.

7.1.8. Set Default (Hex F6)

Este comando inicializa el teclado a los valores por defecto que tiene durante el Power-On-Reset: el teclado responde con un ACK, borra el buffer de salida, pone por defecto el tipo de tecla o “keys type” (solo si se usa el scan code set 3) y el typematic rate/delay, borra la ultima typematic key y continua el rastreo de teclas si no estaba inhibido.

Es **similar a un *reset en caliente***.

7.1.9. Default Disable (Hex F5)

Realiza las mismas funciones que el anterior pero **no** continua el rastreo de teclas, dejando el teclado bloqueado y la espera de comandos. No tiene influencia en los LED.

Es similar a un **bloqueo implícito del teclado**.

7.1.10. Set All Keys (Hex F7, F8, F9, FA)

Estos comandos indican al teclado que cambie el comportamiento de **todas** las teclas según la siguiente lista:

Valor hex	Comando
F7	Set All Keys – Typematic
F8	Set All Keys – Make/Break
F9	Set All Keys – Make
FA	Set All Keys – Typematic/Make/Break

- Typematic: al pulsar la tecla se envía el make code y se permite repetición. No se envían break codes.
- Make/Break: al pulsar la tecla se envía el make code, y al soltarla el break code. No se permite repetición.
- Make: al pulsar la tecla sólo envía el make code. No envía break codes ni repeticiones.
- Typematic/Make/Break: se envían make codes y break codes. Se permite repetición.

El teclado responde con un ACK, limpia su buffer de salida, cambia todas las teclas al tipo indicado y continua el rastreo de teclas (si estaba previamente habilitado).

Estos comandos pueden ser enviados usando cualquier scan code set, pero **únicamente afectan al scan code set 3**.

7.1.11. Set Key Type (Hex FB, FC, FD)

Estos comandos indican al teclado que cambie el comportamiento de teclas **individuales** con el mismo significado que en el comando “Set All Keys Types”:

Valor hex	Comando
FB	Set All Keys – Typematic
FC	Set All Keys – Make/Break
FD	Set All Keys – Make

El teclado responde con un ACK, limpia su buffer de salida, y se prepara para recibir la identificación de la tecla. El host identifica cada tecla por su valor de scan code definido en el scan code set 3. Unicamente los valores del scan code set 3 son validos para identificar la tecla. Una vez recibido la identificación de tecla, el teclado cambia el tipo de la tecla al indicado anteriormente por el comando y continua el rastreo de teclas (si estaba previamente habilitado).

Estos comandos pueden ser enviados usando cualquier scan code set, pero **únicamente afectan al scan code set 3**.

7.1.12. Resend (Hex FE)

El host envía este comando si detecta un fallo en la transmisión del teclado. Este comando sólo puede ser enviado después de una transmisión del teclado y antes de habilitar la comunicación para la siguiente recepción. El teclado responde enviando de nuevo el dato anterior (a menos que el comando anterior fuera un Resend, en cuyo caso el teclado envía el ultimo byte antes del comando Resend, o dicho de otra forma, si ya era un 0xFEh, el último dato que envió que no fuera 0xFEh).

Si el último byte enviado por el teclado era un ACK sin ningún byte de argumento adicional, el teclado contestará al Resend con un ACK.

7.1.13. Reset (Hex FF)

El host envía el comando Reset para iniciar un reset software y forzar al teclado a realizar una comprobación interna. El teclado responde con un comando ACK y se asegura que el host se da por enterado del ACK antes de ejecutar el Reset. El host indica que acepta el ACK colocando las líneas clock y data a nivel alto un mínimo de 500 ms. El teclado permanece deshabilitado desde que recibe el comando de Reset hasta que el host acepta el ACK o envía otro comando que anula al anterior.
Una vez aceptado el ACK el teclado es reinicializado (restablece los valores por defecto para Typematic rate/delay y limpia su buffer de salida) y luego ejecuta de nuevo el BAT. Después de devolver el código de finalización (0xAA), el teclado activa por defecto el scan code set 2.

7.2. Comandos del teclado

Comando	Valor Hex
Key Detection Error/Overrun	00 (set 2 y 3)
Keyboard ID	83 AB
BAT Completion Code	AA
BAT Fairule Code	FC/FD
Echo	EE
Acknowledge (ACK)	FA
Resend	FE
Key Detection Error/Overrun	FF (set 1)
Relase Code	F0
Scan/Relase Code	E0 (set 2)

7.2.1. Key Detection Error (Hex 00 o FF)

El teclado envía un error de detección de tecla si las condiciones del teclado hacen imposible identificar qué tecla es pulsada. Si el teclado esta usando el scan code set 1, el código enviado es 0xFF. Para los sets 2 y 3, el código es 0x00.

7.2.2. Overrun (Hex 00 o FF)

Si la capacidad del buffer interno del teclado se sobrepasa, el ultimo scan code detectado es sustituido por un código de desbordamiento (overrun). El código es enviado al host cuando se llega al final de la cola del buffer. Si el teclado esta usando el scan code set 1, el código enviado es 0xFF. Para los sets 2 y 3, el código es 0x00.

Si el host intenta leer el teclado directamente estando el buffer vacío, accederá a la posición 17ª del mismo encontrándose el código de desbordamiento.

7.2.3 Keyboard ID (Hex 83 AB)

El ID del teclado consiste en dos bytes, 0x83AB. Si el teclado recibe un comando de lectura de ID desde el host, el teclado responde con un ACK, interrumpe el rastreo de teclas y envía los dos bytes de ID comenzando por el LSB (0xAB), y a lo sumo 500 μs después el MSB (0x83). Finalizado el envío el teclado continua el rastreo.

7.2.4. BAT Completion Code (Hex AA)

Si el teclado realiza el BAT con éxito envía el código de finalización 0xAA. Cualquier otro código enviado indica un fallo en el teclado.

7.2.5. BAT Fairule Code (Hex FC/FD)

Si ocurre un error durante el BAT el teclado envía el código 0xFC u otro, interrumpe el rastreo de teclas y espera una respuesta del host o un comando Reset.
Este código también es enviado si como respuesta a una petición de Resend, llega otra entrada errónea[8].

Fallo en el diagnóstico (0xFD): El teclado periódicamente se autocomprueba y envía este código si detecta algún fallo. Si el fallo sucede durante el BAT, dejará de rastrear las teclas en espera de un comando del host; en cualquier otro momento continuará rastreando las teclas.

[8] Ver “Confirmación de comandos”.

7.2.6. Echo (Hex EE)

El teclado envía este código en respuesta a otro comando Echo del host.

7.2.7. Acknowledgment o ACK (Hex FA)

El teclado envía un comando de confirmación (ACK) si le llega cualquier entrada valida (comando o byte de argumento) que no sean los comandos Echo o Resend.

Dependiendo del comando que se confirma, el byte de ACK puede ir seguido de bytes de argumento para completar la respuesta.

Si el host envía un comando o un argumento, debe esperar la respuesta completa del teclado antes de poder enviar otro byte. Aun así, si el teclado es interrumpido por el host mientras está enviando un ACK, descarta el ACK y pasa a recibir y contestar al nuevo comando que envía el host.

Para mas información ver el punto 8 “confirmación de comandos”.

7.2.8. Resend (Hex FE)

El teclado responde con un comando Resend si recibe cualquier entrada incorrecta (comando o argumento no valido, error de paridad o error de framing). Si el host no envía nada al teclado, no se requiere ninguna respuesta.

7.2.9. Relase Code (Hex F0)

El teclado envía 0xF0 si una tecla se deja de pulsar, y seguidamente el scan code de la tecla.

7.2.10. Scan/Relase Code (Hex E0)

Solo aplicable si esta activo el set 2. El teclado envía un código 0xE0 si se pulsa o se suelta una tecla extendida, seguido del scan code de la tecla presionada/soltada

8. Confirmación de comandos

El teclado envía un comando de confirmación (ACK) si le llega cualquier entrada valida (comando o byte de opción) que no sean los comandos Echo o Resend. Dependiendo del comando que se confirma el byte de ACK puede ir seguido de otros para completar la respuesta.

Si el host envía un comando o un argumento, debe esperar la respuesta completa del teclado antes de poder enviar otro byte. Aun así, si el teclado es interrumpido por el host mientras esta enviando un ACK, descarta el ACK y pasa a recibir y contestar al nuevo comando que envía el host.

El teclado responderá antes de 20 ms (25 ms según otras referencias) a menos que el host inhiba la comunicación. Además si la respuesta necesita de varios bytes, los bytes de la respuesta no podrán estar separados mas de 20 ms (el comando Reset es una excepción, pues el ACK y el byte de acabado 0xAA están separados hasta 500 ms debido al retraso producido por la calibración).

Inmediatamente después que el teclado recibe un dato, comprueba si es incorrecto (comando/argumento no valido, error de paridad, error de framing). Si no es correcto, solicitará al host que lo reenvíe enviando comando Resend (0xFE) en lugar de un ACK. Si el dato que envía el host también es incorrecto, el teclado enviara un comando de error (0xFC).

Si el host recibe un comando Resend, debería volver retransmitir el comando entero, no solo el byte de argumento.

Nota: En muchos PC, el puerto PS/2 provee de una repuesta Resend (0xFE) prefabricada si el teclado no envía una respuesta después de un tiempo o si no responde a la señal de "Request-To-Send". Así que esta respuesta aparente desde el teclado, puede indicar que ha sido desconectado.

9. Ejemplo de inicialización

Lo siguiente es un ejemplo de la comunicación entre un ordenador y un teclado cuando se arranca el sistema. Aparentemente los tres primeros comandos fueron enviados por el controlador del teclado, el siguiente (que habilita el LED de Num Lock) por la BIOS, y el resto son enviados por el sistema operativo.

Estos resultados pueden ser distintos de un ordenador a otro pero dan una idea de qué sucede al arrancar el sistema.

```
 Teclado: AA   Auto-test en proceso; AA con paridad incorrecta
    Host: FE   Resend
 Teclado: AA   Auto-test en proceso; AA con paridad incorrecta
    Host: FE   Resend
```

```
 Teclado: AA   Auto-test acabó;Inic. controlador teclado; AA
correcta
    Host: ED   Indicadores de estado On/Off
 Teclado: FA   ACK
    Host: 00   Apaga todos los LEDs
 Teclado: FA   ACK
    Host: F2   Lee ID
 Teclado: FA   ACK
 Teclado: AB   Primer byte de ID
    Host: ED   Indicadores de estado On/Off; inicialización de BIOS
 Teclado: FA   ACK
    Host: 02   Enciende LED Num Lock
 Teclado: FA   ACK
    Host: F3   Cambia Typematic Rate/Delay ;inicializacion de S.O.
 Teclado: FA   ACK
    Host: 20   500 ms / 30.0 cps
 Teclado: FA   ACK
    Host: F4   Habilitado (Enable)
 Teclado: FA   ACK
    Host: F3   Cambia Typematic Rate/delay
 Teclado: FA   ACK
    Host: 00   250 ms / 30.0 cps
 Teclado: FA   ACK
```

10. Tablas de scan codes.

Hay tres modos de funcionamiento en los teclados actuales que elegiremos según la asignación de scan codes que deseemos para cada tecla:

Modo 1 (XT): En este modo el teclado proporciona scan codes XT.

- El make code de una tecla coincide con su scan code.
- El break code de una tecla es el make code pero con el bit 7 activo. Es decir, el break code se puede obtener con un OR de 0x80 y el make code (make OR 0x80).
- El teclado maneja los cursores del teclado numérico -no existían en el teclado XT- simulando la pulsación o liberación de una tecla [Shift] (dependiendo de si [Shift] o [Num Lock] están presionadas) y envía los códigos del teclado numérico.

Modo 2 (AT): El modo 2 funciona como el modo 1 con variaciones.

- El make code de una tecla coincide con su scan code.
- El break code es el scan code pero con un prefijo F0.
- Las teclas extendidas que no existían en el XT, siempre van prefijadas por E0 seguido del scan code.
  Ejemplo: para la tecla [←] su make code es E0, 69, y su break code E0, F0, 69

Modo 3 (MF II):

- Todas las teclas poseen un **único** scan code (no se distingue entre teclas normales y extendidas; la mayoría de scan codes coinciden con el scan code del set 2).
- El make code coincide con el scan code.
- El break code funciona como el modo 2: scan code con un prefijo F0.

Por último, el scan code 0x00 no aparece en ningún scan set.

10.1. Estándares de teclados.

Vamos a ver a continuación la asignación de los scan codes a cada tecla en particular, centraremos en los teclados AT estándar con al menos 101 teclas.

Como hemos comentado anteriormente, la asignación de los scan codes no tiene porque coincidir con el símbolo grabado en la tecla. Esto se pone de manifiesto si cambiamos el lenguaje del teclado: así un teclado en español tiene una tecla [º] -al lado de la tecla [1]-, pero un teclado ingles mostrará el símbolo [`] aunque ambas teclas estén situadas en la misma posición del teclado y producen el mismo scan code. Por ello en lugar de mostrar los scan codes asociados a cada símbolo impreso, vamos a mostrar los scan codes que produce cada tecla física, lo que nos permitirá no depender del lenguaje usado por el teclado. Para ello usamos una numeración de las teclas. Además esta numeración del

teclado es útil si tenemos problemas con teclados estropeados, pues pueden ayudarnos a conocer cuando la BIOS señala una tecla estropeada.

El formato de las tablas es el siguiente:

1ª col: numero asociado a la tecla.
2ª col: símbolo en teclado US asociado a la tecla y entre ( ) el símbolo europeo si es diferente.
3ª col: make code según el set
4ª col: break code según el set
5ª col: el estado por defecto de la tecla (igual para los tres sets):

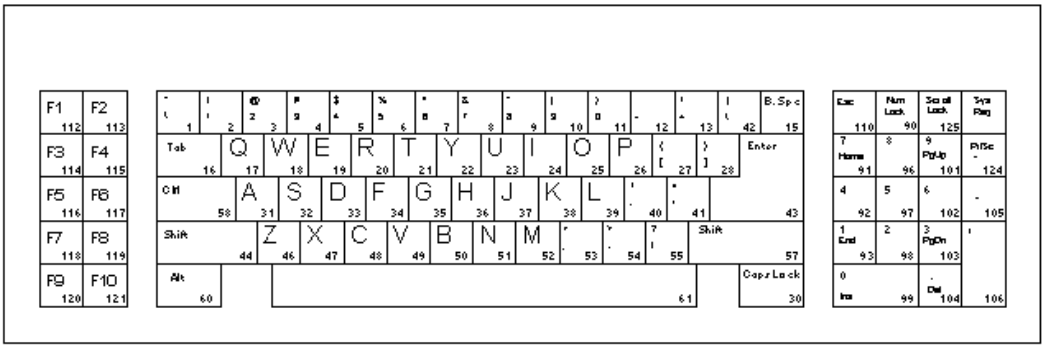
M = sólo make codes.
T = Typematic (repetitiva y make codes).
MB = Make/Break (make/break codes. No repetitiva).

Se incluyen notas sobre combinaciones especiales de las teclas que hay que tener en cuenta pues algunas no siguen la norma general de ese scan set.

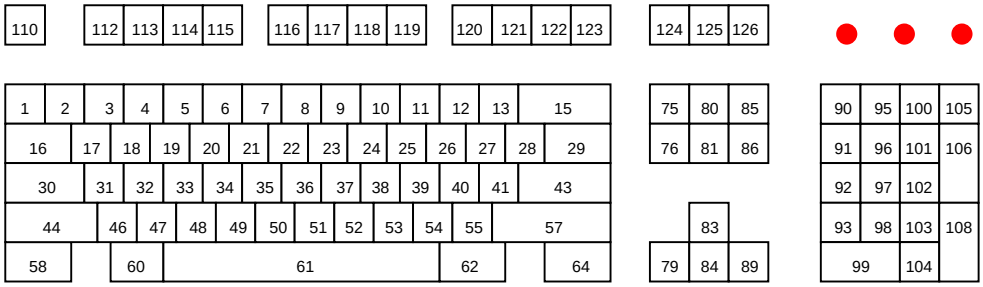
Ya que los teclados modernos pueden tener mas teclas extendidas no estándar como por ejemplo [Wake Up], [Sleep], [Power], teclas para control multimedia o acceso a páginas web y email, mostraremos estas teclas y sus scan codes asociados por separado, pues sus posiciones en el teclado suelen variar mucho al no estar estandarizadas.

Mostramos a continuación unos esquemas de la disposición de las teclas en los distintos modelos de teclado que resultan de nuestro interés. Incluimos el teclado de 84 teclas XT solo por motivos históricos pues, aunque ya no se usan y son complicados de encontrar, los estándares posteriores de teclados heredaron su asignación de codigos scan.

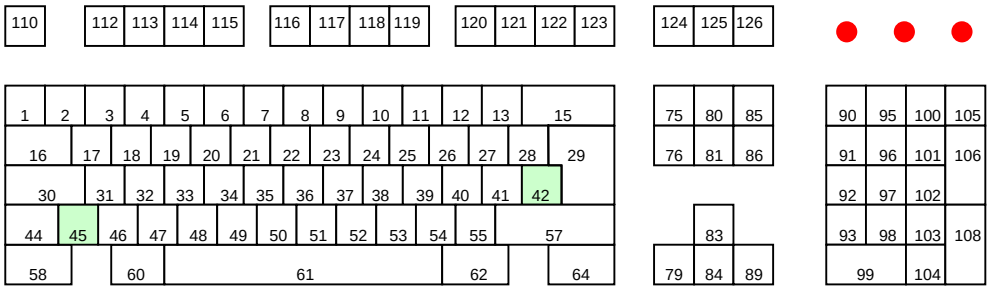
- **84 teclas (US)**



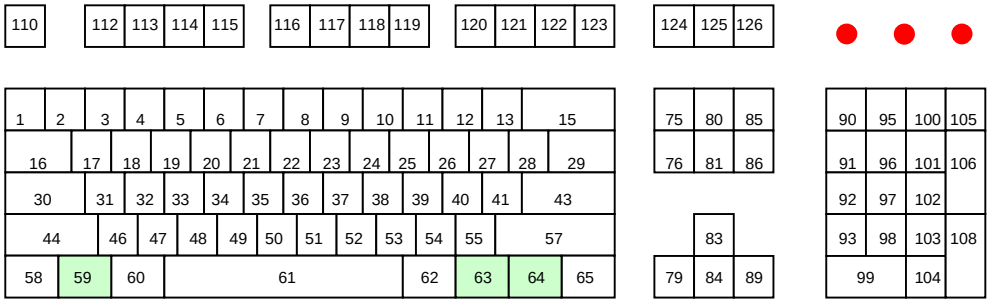
- **101 teclas (US)**



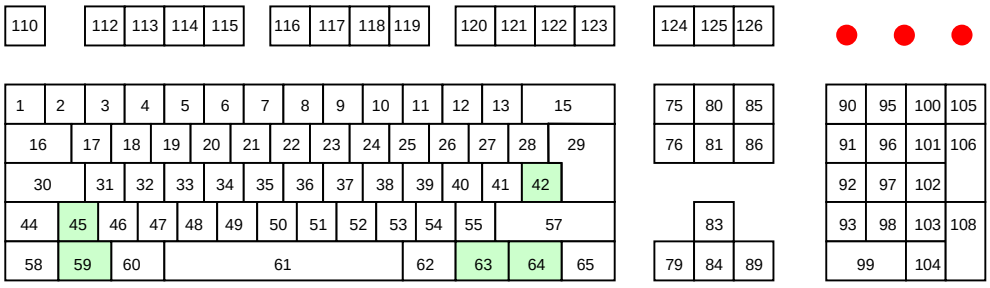
- **102 teclas (Europa)**



- **104 teclas (US)**



- **105 teclas (Europa)**



10.2. Scan code set 1

Número tecla	Tecla US(EU)	Make code	Break code	Key type def.
1	` (º)	29	A9	T
2	1	02	82	T
3	2	03	83	T
4	3	04	84	T
5	4	05	85	T
6	5	06	86	T
7	6	07	87	T
8	7	08	88	T
9	8	09	89	T
10	9	0A	8A	T
11	0	0B	8B	T
12	- (‘)	0C	8C	T
13	= (¡)	0D	8D	T
15	Del	0E	8E	T
16	Tab	0F	8F	T
17	q	10	90	T
18	w	11	91	T
19	e	12	92	T
20	r	13	93	T
21	t	14	94	T
22	y	15	95	T
23	u	16	96	T
24	i	17	97	T
25	o	18	98	T
26	p	19	99	T
27	[	1A	9A	T
28	]	1B	9B	T
29*	\ (})	2B	AB	T
30	Caps	3A	BA	MB
31	a	1E	9E	T
32	s	1F	9F	T
33	d	20	A0	T
34	f	21	A1	T
35	g	22	A2	T
36	h	23	A3	T
37	j	24	A4	T
38	k	25	A5	T
39	l	26	A6	T
40	; (ñ)	27	A7	T
41	´	28	A8	T
42**	\ (})	2B	AB	T
43	Enter	1C	9C	T
44	LShift	2A	AA	MB
45**	<> (\ )	56	D6	T
46	z	2C	AC	T
47	x	2D	AD	T
48	c	2E	AE	T
49	v	2F	AF	T
50	b	30	B0	T
51	n	31	B1	T
52	m	32	B2	T
53	,	33	B3	T
54	.	34	B4	T

55	/ (-)	35	B5	T
57	RShift	36	B6	MB
58	LCtrl	1D	9D	MB
59	LGui	E0, 5B	E0, DB	MB
60	LAlt	38	B8	MB
61	Space	39	B9	T
62	RAlt	E0, 38	E0, B8	M
63	RGui	E0, 5C	E0, DC	MB
64	RCtrl	E0, 1D	E0, 9D	M
65	Apps	E0, 5D	E0, DD	MB
75 nota 1,2,3	Ins	E0, 52	E0, D2	M
76 nota 1,2,3	Delete	E0, 53	E0, D3	T
79 nota 1,2,3	←	E0, 4B	E0, CB	T
80 nota 1,2,3	Home	E0, 47	E0, C9	M
81 nota 1,2,3	End	E0, 4F	E0, CF	M
83 nota 1,2,3	↑	E0, 48	E0, C8	T
84 nota 1,2,3	↓	E0, 50	E0, D0	T
85 nota 1,2,3	Pg Up	E0, 49	E0, C9	M
86 nota 1,2,3	Pg Dn	E0, 51	E0, D1	M
89 nota 1,2,3	→	E0, 4D	E0, CD	T
90	Num Lock	45	C5	M
91	Kp 7	47	C7	M
92	Kp 4	4B	CB	M
93	Kp 1	4F	CF	M
95 nota 6	Kp /	E0, 35	E0, B5	M
96	Kp 8	48	C8	M
97	Kp 5	4C	CC	M
98	Kp 2	50	D0	M
99	Kp 0	52	D2	M
100	Kp *	37	B7	M
101	Kp 9	49	C9	M
102	Kp 6	4D	CD	M
103	Kp 3	51	D1	M
104	Kp .	53	D3	M
105	Kp -	4A	CA	M
106	Kp +	4E	CE	T
108	Kp Enter	E0, 1C	E0, 9C	M
110	Esc	01	81	M
112	F1	3B	BB	M
113	F2	3C	BC	M
114	F3	3D	BD	M
115	F4	3E	BE	M
116	F5	3F	BF	M
117	F6	40	C0	M
118	F7	41	C1	M
119	F8	42	C2	M
120	F9	43	C3	M
121	F10	44	C4	M
122	F11	57	D7	M
123	F12	58	D8	M
124 nota 4	Prt Scr	E0, 2A, E0, 37	E0, B7, E0, AA	M
125	Scroll	46	C6	M
126 nota 5	Pause	E1, 1D, 45, E1, 9D, C5	nada	M

* Sólo en teclados con 101 teclas – US (y otros) ** Sólo en teclados con 102 teclas – Europa (y otros)

Scan codes de ACPI

Tecla	Make Code	Break Code
Power	E0, 5E	E0, DE
Sleep	E0, 5F	E0, DF
Wake	E0, 63	E0, E3

Scan codes de teclas Multimedia de Windows

Tecla	Make Code	Break Code
Next Track	E0, 19	E0, 99
Previous Track	E0, 10	E0, 90
Stop	E0, 24	E0, A4
Play/Pause	E0, 22	E0, A2
Mute	E0, 20	E0, A0
Volume Up	E0, 30	E0, B0
Volume Down	E0, 2E	E0, AE
Media Select	E0, 6D	E0, ED
E-Mail	E0, 6C	E0, EC
Calculator	E0, 21	E0, A1
My Computer	E0, 6B	E0, EB
WWW Search	E0, 65	E0, E5
WWW Home	E0, 32	E0, B2
WWW Back	E0, 6A	E0, EA
WWW Forward	E0, 69	E0, E9
WWW Stop	E0, 68	E0, E8
WWW Refresh	E0, 67	E0, E7
WWW Favorites	E0, 66	E0, E6

Nota 1: Si Num Lock esta activado:

Make: E0, 2A, mm, mm /
Break: bb, bb, E0, AA

Nota 2: Si Num Lock esta desactivado y se pulsa [LSHIFT]:

Make: E0, AA, mm, mm
Break: bb, bb, E0, 2A

Nota 3: Si Num Lock esta desactivado y se pulsa [RSHIFT]:

Make: E0, B6, mm, mm
Break: bb, bb, E0, 36

En cualquier caso, si se pulsan ambas teclas SHIFT a la vez se transmiten los scan codes conjuntamente en el orden (pendiente de verificación):

Make: E0, AA, E0, B6, mm, mm
Break: bb, bb, E0, 36, E0, 2A

Esto se resume en la siguiente tabla:

Combinación	Num Lock OFF	Num Lock ON
Make/break base	mm, mm / bb, bb	E0, 2A, mm, mm / bb bb, E0, AA
[LShift]	E0, AA, mm, mm / bb, bb, E0, 2A	mm, mm / bb bb
[RShift]	E0, B6, mm, mm / bb, bb, E0, 36	mm, mm / bb bb
[LCtrl]	mm, mm / bb bb	mm, mm / bb bb
[RCtrl]	mm, mm / bb bb	mm, mm / bb bb
[LAlt]	mm, mm / bb bb	mm, mm / bb bb
[RAltT]	mm, mm / bb bb	mm, mm / bb bb
[LShift]+[RShift]	E0, AA, E0, B6, mm, mm / bb, bb, E0, 36, E0, 2A	E0, AA, E0, B6, mm, mm / bb, bb, E0, 36, E0, 2A

Nota 4: Cuando se presionan las teclas [Ctrl] o [Shift] + [Prt Scr]:

Make code: E0, 37
Break code: E0, B7

Cuando se presionan las teclas [Alt]+[PRT Scr]:

Make code: 54
Break code: D4

Nota 5: Cuando se presionan las teclas [Ctrl]+[Pause]:

Make code: E0, 46, E0, C6
Break code: nada

La tecla [Pause] **no** es Typematic (repetitiva). Todos los scan codes asociados ocurren al pulsar la tecla, es decir, [Pause] es de tipo Make siempre

Nota 6: Cuando se presina [LShift]:

Make code: E0, AA, mm, mm
Break code: bb, bb, E0, 2A

Cuando se presiona [RShift]:

Make code: E0, B6, mm, mm
Break code: bb, bb, E0, 36

Si [LShift] y [RShift] están pulsadas a la vez, se envían ambos conjuntos de scan codes junto con el de la tecla:

Make code: E0, AA, E0, B6, mm, mm
Break code: bb, bb, E0, 36, E0, 2A

Ejemplos de secuencias generadas:

- Pulsamos [a] (1E) un tiempo, luego pulsamos [9] (0A). Soltamos [a] (9E) y mantenemos apretado [9] hasta que la liberamos (8A):

  1E, 1E, 1E, 1E, 1E, 1E, 0A, 0A, 0A, 0A, 0A, 9E, 0A, 0A, 0A, 0A, 8A

- Pulsamos [a] (1E) un tiempo, luego pulsamos [9] (0A). Soltamos [9] (8A) y mantenemos [A] apretada hasta que la liberamos (9E):

  1E, 1E, 1E, 1E, 1E, 1E, 0A, 8A, 9E

  En este caso, [a] mientras se pulsa es repetitiva. Cuando pulsamos [9], la nueva tecla repetitiva es [9] (por eso no sigue repitiendo [a] -1E- después de liberar [9] y solo vemos el 9E al liberar A).

- Con Num Lock desactivado:

  Pulsamos [LShift] un tiempo, luego [↑]. Soltamos [↑] y luego [LShift]:

  2A, 2A, 2A, E0, AA, E0, 48 | E0, C8, E0, 2A, AA

  Pulsamos [RShift] un tiempo, luego [↑]. Soltamos [↑] y luego [RShift]:

  36, 36, 36, E0, B6, E0, 48 | E0, C8, E0, 36, B6

- Pulsamos [LCtrl], luego [Pause] y soltamos [LCtrl]:

  1D, 1D, 1D, E0, 46, E0, C6, 9D

  Pulsamos [RCtrl], luego [Pause] y soltamos [RCtrl]:

  E0, 1D, E0, 46, E0, C6, E0, 9D

Pulsamos [LShift] y [/] del teclado numérico:

2A, 2A, 2A, E0, AA, E0, 35, E0, B5, E0, 2A, AA

Pulsamos [RShift]: y [/] del teclado numérico:

36, 36, 36, E0, B6, E0, 35, E0, B5, E0, 36, B6

Pulsamos [LShift] + [RShift] y [/] del teclado numérico en es orden. Soltamos primero [/], luego [LShift], y luego [Rshift]:

2A, 2A, 36, 36, E0, AA, E0, B6, E0, 35, E0, B5, E0, 36, E0, 2A, AA, B6

10.3. Scan code set 2

Número tecla	Tecla US(EU)	Make code	Break code	Key type def.
1	` (º)	0E	F0, 0E	T
2	1	16	F0, 16	T
3	2	1E	F0, 1E	T
4	3	26	F0, 26	T
5	4	25	F0, 25	T
6	5	2E	F0, 2E	T
7	6	36	F0, 36	T
8	7	3D	F0, 3D	T
9	8	3E	F0, 3E	T
10	9	46	F0, 46	T
11	0	45	F0, 45	T
12	- (‘)	4E	F0, 4E	T
13	= (¡)	55	F0, 55	T
15	Del	66	F0, 66	T
16	Tab	0D	F0, 0D	T
17	q	15	F0, 15	T
18	w	1D	F0, 1D	T
19	e	24	F0, 24	T
20	r	2D	F0, 2D	T
21	t	2C	F0, 2C	T
22	y	35	F0, 35	T
23	u	3C	F0, 3C	T
24	i	43	F0, 43	T
25	o	44	F0, 44	T
26	p	4D	F0, 4D	T
27	[	54	F0, 54	T
28	]	5B	F0, 5B	T
29*	\ (})	5D	F0, 5D	T
30	Caps	58	F0, 58	MB
31	a	1C	F0, 1C	T
32	s	1B	F0, 1B	T
33	d	23	F0, 23	T
34	f	2B	F0, 2B	T
35	g	34	F0, 34	T
36	h	33	F0, 33	T
37	j	3B	F0, 3B	T
38	k	42	F0, 42	T
39	l	4B	F0, 4B	T
40	; (ñ)	4C	F0, 4C	T
41	´	52	F0, 52	T
42**	\ (})	5D	F0, 5D	T
43	Enter	5A	F0, 5A	T
44	LShift	12	F0, 12	MB
45**	<> (\ )	61	F0, 61	T
46	z	1A	F0, 1A	T
47	x	22	F0, 22	T
48	c	21	F0, 21	T
49	v	2A	F0, 2A	T
50	b	32	F0, 32	T
51	n	31	F0, 31	T
52	m	3A	F0, 3A	T
53	,	41	F0, 41	T
54	.	49	F0, 49	T

55	/ (-)	4A	F0, 4A	T
57	RShift	59	F0, 59	MB
58	LCtrl	14	F0, 14	MB
59	LGui	E0, 1F	E0, F0, 1F	MB
60	LAlt	11	F0, 11	MB
61	Space	29	F0, 29	T
62	RAlt	E0, 11	E0, F0, 11	M
63	RGui	E0, 27	E0, F0, 27	MB
64	RCtrl	E0, 14	E0, F0, 14	M
65	Apps	E0, 2F	E0, F0, 2F	MB
75 nota 1,2,3	Ins	E0, 70	E0, F0, 70	M
76 nota 1,2,3	Delete	E0, 71	E0, F0, 71	T
79 nota 1,2,3	←	E0, 6B	E0, F0, 6B	T
80 nota 1,2,3	Home	E0, 6C	E0, F0, 6C	M
81 nota 1,2,3	End	E0, 69	E0, F0, 69	M
83 nota 1,2,3	↑	E0, 75	E0, F0, 75	T
84 nota 1,2,3	↓	E0, 72	E0, F0, 72	T
85 nota 1,2,3	Pg Up	E0, 7D	E0, F0, 7D	M
86 nota 1,2,3	Pg Dn	E0, 7A	E0, F0, 7A	M
89 nota 1,2,3	→	E0, 74	E0, F0, 74	T
90	Num Lock	77	F0, 77	M
91	Kp 7	6C	F0, 6C	M
92	Kp 4	6B	F0, 6B	M
93	Kp 1	69	F0, 69	M
95 nota 1,2	Kp /	E0, 4A	E0, F0, 4A	M
96	Kp 8	75	F0, 75	M
97	Kp 5	73	F0, 73	M
98	Kp 2	72	F0, 72	M
99	Kp 0	70	F0, 70	M
100	Kp *	7C	F0, 7C	M
101	Kp 9	7D	F0, 7D	M
102	Kp 6	74	F0, 74	M
103	Kp 3	7A	F0, 7A	M
104	Kp .	71	F0, 71	M
105	Kp -	7B	F0, 7B	M
106	Kp +	79	F0, 79	T
108	Kp Enter	E0, 5A	E0, F0, 5A	M
110	Esc	76	F0, 76	M
112	F1	05	F0, 05	M
113	F2	06	F0, 06	M
114	F3	04	F0, 04	M
115	F4	0C	F0, 0C	M
116	F5	03	F0, 03	M
117	F6	0B	F0, 0B	M
118	F7	83	F0, 83	M
119	F8	0A	F0, 0A	M
120	F9	01	F0, 01	M
121	F10	09	F0, 09	M
122	F11	78	F0, 78	M
123	F12	07	F0, 07	M
124 nota 4	Prt Scr	E0, 12, E0, 7C	E0, F0, 7C, E0,F0, 12	M
125	Scroll	7E	F0, 7E	M
126 nota 5	Pause	E1, 14, 77, E1, F0, 14, F0, 77	Nada	M

* Sólo en teclados con 101 teclas – US (y otros) ** Sólo en teclados con 102 teclas – Europa (y otros)

Scan codes ACPI

Tecla	Make Code	Break Code
Power	E0, 37	E0, F0, 37
Sleep	E0, 3F	E0, F0, 3F
Wake	E0, 5E	E0, F0, 5E

Scan codes de teclas Multimedia de Windows

Tecla	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, 3A	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

Nota 1: Cuando se presiona la tecla [LShift]:

Make code: E0, F0, 12, mm, mm
Break code: bb, bb, bb, E0, 12

Nota 2: Cuando se presiona la tecla [RShift]:

Make code: E0, F0, 59, mm, mm
Break code: bb, bb, bb, E0 59

Nota 3: Cuando se presiona la tecla [Num Lock] -On-:

Make code: E0, 12, mm, mm
Break code: bb, bb, bb, E0, F0, 12

Nota 4: Cuando se presionan las teclas [Ctrl]+[Prt Scr]:

Make code: E0, 7C
Break code: E0, F0, 7C

Cuando se presionan las teclas [Alt]+[Prt Scr]:

Make code: 84
Break code: F0, 84

Nota 5: Cuando se presionan las teclas [Ctrl]+[Pause]:

Make code: E0, 7E, E0, F0, 7E
Break code: nada

La tecla [Pause] **no** es Typematic (repetitiva). Todos los scan codes asociados ocurren al pulsar la tecla, es decir, [Pause] es de tipo Make siempre.

'mm mm' – make code de la tecla normal cuando se pulsa.
'bb bb bb' – break code de la tecla normal cuando se pulsa.

10.4. Scan code set 3

Número tecla	Tecla US(EU)	Make code	Break code	Key type def.
1	` (º)	0E	F0, 0E	T
2	1	16	F0, 16	T
3	2	1E	F0, 1E	T
4	3	26	F0, 26	T
5	4	25	F0, 25	T
6	5	2E	F0, 2E	T
7	6	36	F0, 36	T
8	7	3D	F0, 3D	T
9	8	3E	F0, 3E	T
10	9	46	F0, 46	T
11	0	45	F0, 45	T
12	- (‘)	4E	F0, 4E	T
13	= (¡)	55	F0, 55	T
15	Del	66	F0, 66	T
16	Tab	0D	F0, 0D	T
17	q	15	F0, 15	T
18	w	1D	F0, 1D	T
19	e	24	F0, 24	T
20	r	2D	F0, 2D	T
21	t	2C	F0, 2C	T
22	y	35	F0, 35	T
23	u	3C	F0, 3C	T
24	i	43	F0, 43	T
25	o	44	F0, 44	T
26	p	4D	F0, 4D	T
27	[	54	F0, 54	T
28	]	5B	F0, 5B	T
29*	\ (})	5C	F0, 5C	T
30	Caps	14	F0, 14	MB
31	a	1C	F0, 1C	T
32	s	1B	F0, 1B	T
33	d	23	F0, 23	T
34	f	2B	F0, 2B	T
35	g	34	F0, 34	T
36	h	33	F0, 33	T
37	j	3B	F0, 3B	T
38	k	42	F0, 42	T
39	l	4B	F0, 4B	T
40	; (ñ)	4C	F0, 4C	T
41	´	52	F0, 52	T
42**	\ (})	5C	F0, 5C	T
43	Enter	5A	F0, 5A	T
44	LShift	12	F0, 12	MB
45**	<> (\ )	13	F0, 13	T
46	z	1A	F0, 1A	T
47	x	22	F0, 22	T
48	c	21	F0, 21	T
49	v	2A	F0, 2A	T
50	b	32	F0, 32	T
51	n	31	F0, 31	T
52	m	3A	F0, 3A	T
53	,	41	F0, 41	T
54	.	49	F0, 49	T

55	/ (-)	4A	F0, 4A	T
57	RShift	59	F0, 59	MB
58	LCtrl	11	F0, 11	MB
59	LGui	8B	F0, 8B	MB
60	LAlt	19	F0, 19	MB
61	Space	29	F0, 29	T
62	RAlt	39	F0, 39	M
63	RGui	8C	F0, 8C	MB
64	RCtrl	58	F0, 58	M
65	Apps	8D	F0, 8D	MB
75	Ins	67	F0, 67	M
76	Delete	64	F0, 64	T
79	←	61	F0, 61	T
80	Home	6E	F0, 6E	M
81	End	65	F0, 65	M
83	↑	63	F0, 63	T
84	↓	60	F0, 60	T
85	Pg Up	6F	F0, 6F	M
86	Pg Dn	6D	F0, 6D	M
89	→	6A	F0, 6A	T
90	Num Lock	76	F0, 76	M
91	Kp 7	6C	F0, 6C	M
92	Kp 4	6B	F0, 6B	M
93	Kp 1	69	F0, 69	M
95	Kp /	77	F0, 77	M
96	Kp 8	75	F0, 75	M
97	Kp 5	73	F0, 73	M
98	Kp 2	72	F0, 72	M
99	Kp 0	70	F0, 70	M
100	Kp *	7E	F0, 7E	M
101	Kp 9	7D	F0, 7D	M
102	Kp 6	74	F0, 74	M
103	Kp 3	7A	F0, 7A	M
104	Kp .	71	F0, 71	M
105	Kp -	84	F0, 84	M
106	Kp +	7C	F0, 7C	T
108	Kp Enter	79	F0, 79	M
110	Esc	08	F0, 08	M
112	F1	07	F0, 07	M
113	F2	0F	F0, 0F	M
114	F3	17	F0, 17	M
115	F4	1F	F0, 1F	M
116	F5	27	F0, 27	M
117	F6	2F	F0, 2F	M
118	F7	37	F0, 37	M
119	F8	3F	F0, 3F	M
120	F9	47	F0, 47	M
121	F10	4F	F0, 4F	M
122	F11	56	F0, 56	M
123	F12	5E	F0, 5E	M
124	Prt Scr	57	F0, 57	M
125	Scroll	5F	F0, 5F	M
126	Pause	62	F0, 62	M

* Sólo en teclados con 101 teclas – US (y otros) ** Sólo en teclados con 102 teclas – Europa (y otros)

*** En gris las teclas que han cambiado respecto el set 2

11. Bibliografía

- Aplication note AN1723 de Motorola
- Synaptcis Touchpad Interfacing Guide, 2º Ed.
- http://www.beyondlogic.org/
- http://panda.cs.ndsu.nodak.edu/~achapwes/PICmicro/keyboard/scancodes.html
- http://panda.cs.ndsu.nodak.edu/~achapwes/PICmicro/PS2/ps2.htm - Protocol used by AT and PS/2 keyboards.
- http://www.repairfaq.org – PC Keyboard FAQ