MF2 Compatible Keyboard with COP8 Microcontrollers

National Semiconductor
Application Note 734
Volker Soffel
February 1991

ABSTRACT

This application note describes the implementation of an IBM MF2 compatible keyboard with National Semiconductor's COP888CL or COP943C/COP880CL microcontrollers. Two different solutions have been developed. One solution, suitable for laptop/notebook keyboards is based on the COP888CL with special power saving techniques. The other for most price competitive standard desktop keyboards is based on the COP943C/COP880C microcontrollers. The same principles can be applied to all types of keyboards or data input devices.

FEATURES

- Single chip solution
- Low cost R/C or ceramic oscillator optional
- LED direct drive capability
- I/Os with software programmable on chip pull-ups
- Current saving M2CMOS technology
- Multi-input wakeup and HALT mode for further power consumption reduction (COP888CL only)
- Software key rollover
- Schmitt triggers on keyboard data and clock lines

INTRODUCTION

The expression MF2 keyboard stands for multi-functional keyboard version 2. This type of keyboard was first developed and defined by IBM for use with all types of PC (XT, AT, PS/2). In the meantime it has become an industry standard and today nearly all PCs have an MF2 compatible keyboard. As the name suggests, this keyboard features all operation modes which are necessary to stay compatible with the older XT and AT type keyboards. In the following chapters the features and functions of an MF2 keyboard as well as their implementation with a COP8 microcontroller are described.

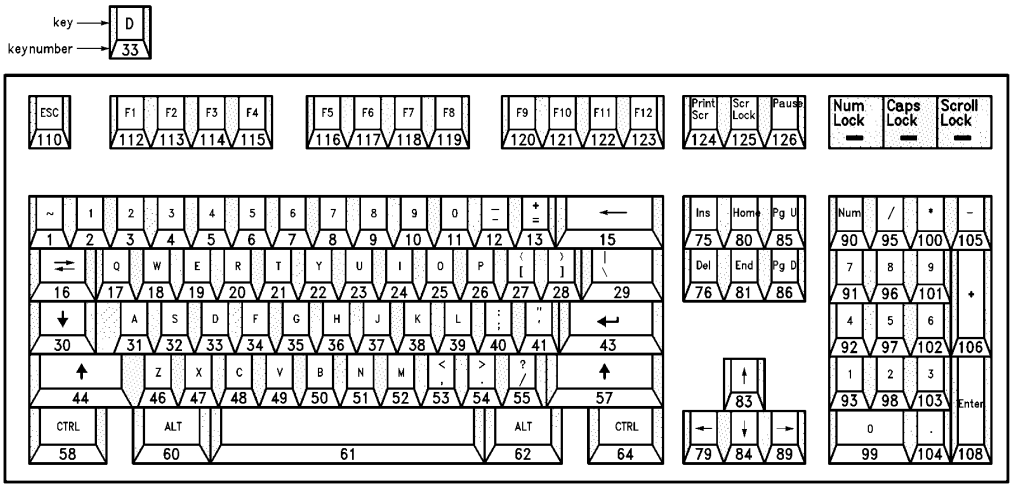
MF2 KEYBOARD KEY-LAYOUT

Figure 1 shows the key layout of the U.S. version of an MF2 keyboard. Its outer appearance is characterized by 101 keys (102 for some countries), a separate cursor and numeric key pad, and 12 function keys in the upper row. The keyboard sends a ''make'' code if a key is depressed and a ''break'' code if the key is released. These make and break codes are independent of any country-specific keyboard layouts, which means they are independent of the symbols printed on the keys. These codes are solely determined by the physical position of a key on the keyboard. The physical position of a key on an MF2 keyboard is defined by its assigned key number, which is shown in *Figure 1.*

HARDWARE

Laptop/Notebook Keyboard With COP888CL

Figure 2 shows the schematics of an MF2 keyboard with a COP888CL microcontroller. The G, C and L ports of the COP888CL are software programmable I/Os and can be programmed either as TRI-STATE® inputs, inputs with weak pull-up, push-pull output low, or push-pull output high.

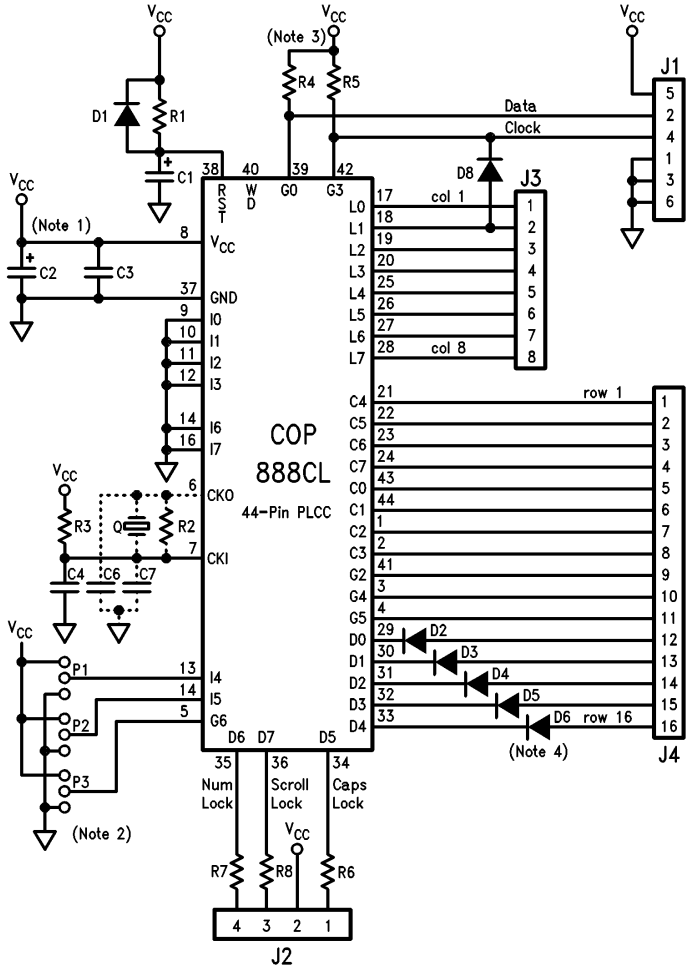


TL/DD/11091–10

FIGURE 1. MF2 Keyboard U.S. Layout

TRI-STATE® is a registered trademark of National Semiconductor Corporation.

RRD-B30M75/Printed in U. S. A.



TL/DD/11091–11

Note 1: C2 (47 μF level off capacitor) can be removed when the power supply ripple $<$ $\pm$10%, 0.5 V/ms.

Note 2: Jumper P1: Mode select: 0 = XT-mode, 1 = AT-mode. Jumper P2, P3: not used.

Note 3: Care must be taken if there are pullups in the computer system that clock/data line current $<$ 3 mA.

Note 4: Diodes D2–D6 should be removed if keyboard has hardware keyrollover (diodes in matrix).

FIGURE 2. MF-2 Keyboard Schematics with a 44-Pin COP888CL

The keyboard is organized as an 8 input by 16 output matrix. The COP888CL's L port is configured as a weak pull-up input port, thus allowing the use of the multi-input wakeup feature. Most of the time the chip is in the current saving HALT mode (Idd $\leq$ 10 μA). Any keystroke or a data transmission from the computer will create a high to low transition on one of the L lines, which wakes up the μC from HALT mode. After returning from the HALT mode, the keyboard is scanned in order to detect which key is pressed and the appropriate key code is sent to the computer. This event-driven keyboard scanning results in lowest possible current consumption as HALT mode is even entered between successive single keystrokes. The diodes in the D-lines of the key matrix prevent a high current from being drawn. When two keys in the same column are pressed, two outputs could be potentially connected together: one of the D output lines, which is high and the polled line, which is pulled low. In this case, excessive current would be drawn without the protection diodes. These diodes can be omitted if the keyboard already has decoupling diodes in its matrix (hardware key rollover). All other matrix lines source current in the μA range and there is no need for current limiting diodes.

The G0 and G3 pins are used for the keyboard data and clock lines. The pull-ups on these lines ensure a defined logic ''1'' level. The keyboard interface on the computer side uses open collector drivers and the G0, G3 pins of the COP888CL are configured as TRI-STATE (Hi−Z) inputs when a ''1'' is written to the data or clock line. To output a logic ''0'' the μC pulls the data or clock line low (push-pull low output). A maximum current of 3 mA can be sunk into the data and clock pins. Schmitt triggers on the data and clock line inputs reduce the risk of errors in the data received by the keyboard.

The microcontroller provides the option of using a low cost R/C oscillator with frequency variation tight enough to fulfill the requirements for a keyboard, in addition to the option of using a crystal or a ceramic clock.

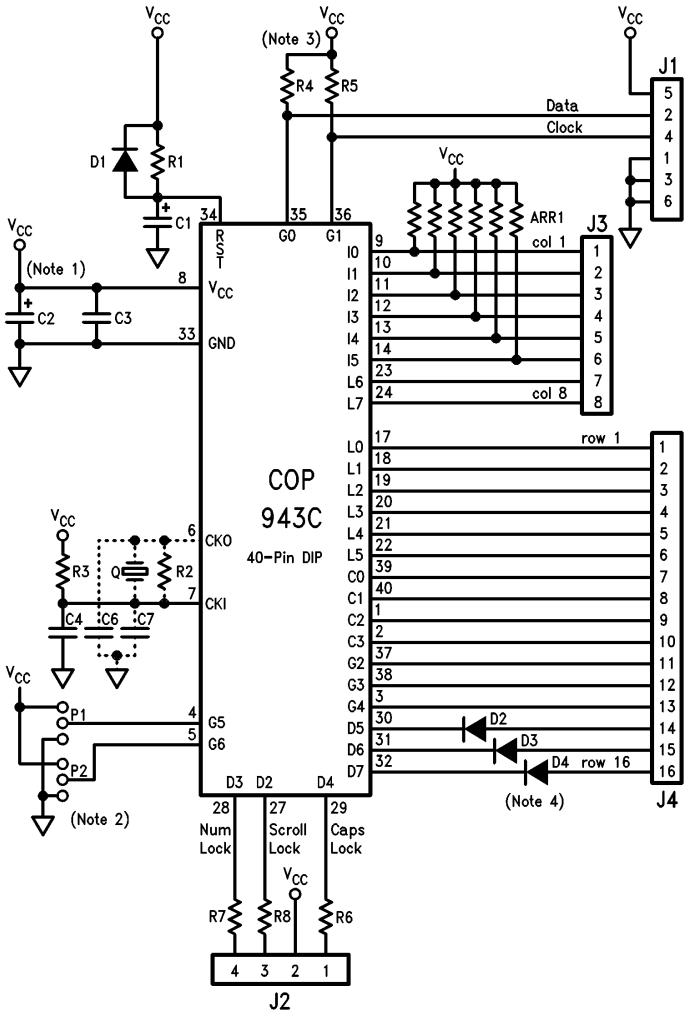
The XT or AT/PS-2 operation mode can be selected via a hardware switch. Additional inputs for customer specific settings are available.

The three LEDs of an MF2 keyboard are driven directly by three of the COP888CL's high sink D-lines (max. 15 mA for each pin), thus eliminating the need for additional LED drivers or transistors.

The keyboard logic generates a Power-On Reset (POR) signal when the power is first applied to the keyboard. After POR the keyboard performs the Basic Assurance Test (BAT). The BAT consists of a keyboard controller self-test. During the BAT, any activity on the data and clock lines is ignored. The 3 keyboard LEDs are turned on at the beginning and turned off at the end of the BAT. Upon satisfactory completion of the BAT, the keyboard sends the BAT completion code (hex AA) to the computer and keyboard scanning begins. Any code other than hex AA is interpreted by the computer as a BAT error.

Desktop Keyboard with COP943C or COP880C

Figure 3 shows the schematic for an MF2 keyboard with the COP943C/COP880C. The only difference compared to COP888CL solution is that the COP943C/COP880C microcontrollers do not have the multi-input wakeup feature, which allows an event driven keyboard scanning. The key matrix is therefore continuously scanned in a loop. With the COP943C/COP880C solution a part of the I port is used as the key matrix input. The I port is a TRI-STATE (Hi−Z) input port (requires external pull-ups).



TL/DD/11091–12

Note 1: C2 (47 μF level off capacitor) can be removed when the power supply ripple $<$ $\pm$10%, 0.5 V/ms.

Note 2: Jumper P1: Mode select: 0 = XT-mode, 1 = AT-mode. Jumper P2: P3: not used.

Note 3: Care must be taken if there are pullups in the computer system that clock/data line current $<$ 3 mA.

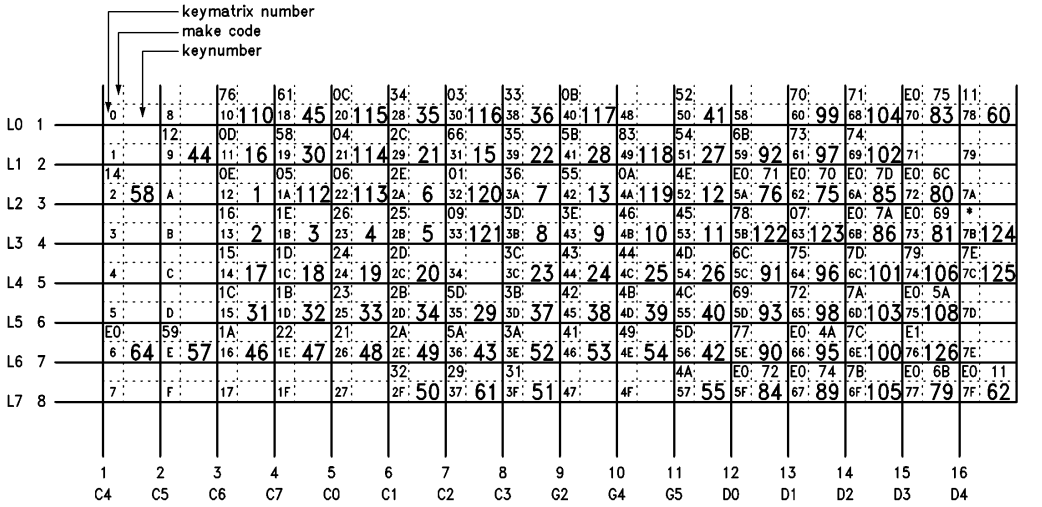
Note 4: Diodes D2–D4 should be removed if keyboard has hardware keyrollover (diodes in matrix).

FIGURE 3. MF-2 Keyboard Schematic with a 40-Pin COP943C/COP880C

Key Matrix Organization

Figure 4 shows an example of what an MF2 keyswitch matrix could look like. Each key position in the matrix is marked with its key number.

For example: Key number ''58'' is located at the key matrix position number ''2'' and has the AT-set make code ''14 Hex''. Looking at *Figure 1,* one can see that key number ''58'' belongs to the left ''CNTRL'' key. Note that the ''SHIFT'', ''CNTRL'' and ''ALT'' keys are located in their own matrix lines, separate from all other keys. The reasons for that will be explained in the chapter ''Software Key Rollover''.



TL/DD/11091–13

FIGURE 4. Keyboard Matrix COP888CL AT Code Set

Code Sets

The MF2 keyboard supports 3 different sets of make and break codes. Code set 1 is used for XT/PC and PS/2−30 compatible computers. Code set 2 is used for AT and all other PS/2 models compatible computers and code set 3 is used for workstations and terminal emulations on the PC. The country specific keyboard driver on the PC side converts the ''key position'' codes from the keyboard into the ASCII codes that correspond to the characters printed on the keycaps (as long as the right driver is installed on the PC). Appendix 1 gives a complete overview of the key numbers and their make and break codes for all 3 code sets. The symbols of the U.S. keyboard layout are only listed for reference and are different for other country layouts. The break code for code set 1 is equal to the make code with the most significant bit set. The make codes preceded with a ''F0 Hex'' code give the break codes of code sets 2 and 3.

KEYBOARD SOFTWARE

The software of the keyboard microcontroller can be subdivided into the following five main tasks:

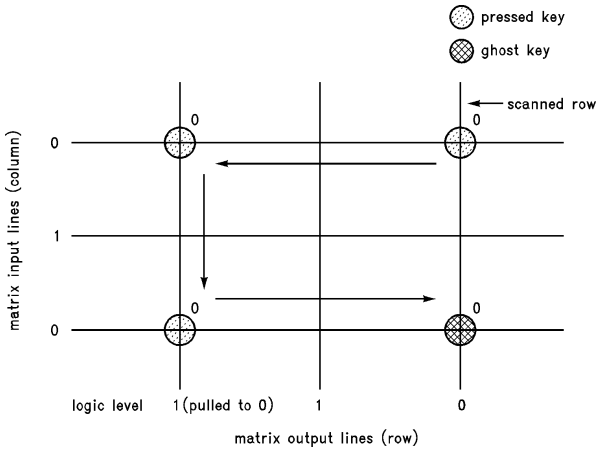
- key detection
- software key rollover
- key decoding and encoding
- keycode transmission
- keyboard command set

Key Detection

Key detection is done by scanning the keyboard matrix in the following way. Sequentially each of the 16 matrix output lines are pulled low, while all the others are high. The 8 matrix input lines are read and the 8-bit input value is compared with the result of the previous scanning of the same matrix output line (a history of the previous scan is kept in the μC's RAM). Thus the keyboard microcontroller's key detection routine detects any key change in that matrix output line (key pressed or released) since the previous scan. It is important to recognize released keys, as the MF2 keyboard not only sends a key's ''make'' code when the key is pressed, but also a key's ''break'' code when the key is released. Key debouncing is performed by software by making sure that the time between two scans is bigger than the key bounce time (typically 8 ms).

Software Key Rollover

Software key rollover means that no decoupoing diodes are used in the key switch matrix. However, the keyboard action is still N key rollover in nature. That is, if N keys are depressd in some sequence and held down, the make code of these keys is transmitted in that sequence. However, if three keys from three corners of a rectangle in the key switching matrix are depressed, a ''ghost'' key (a key which is not really pressed) would be created (see *Figure 5*). To prevent this, a special algorithm, which checks for such special key combinations, has been implemented into the keyboard software. If a ''ghost'' key has been detected the keyboard outputs the ''key detection error code'' and the N key rollover reverts to a 2 key rollover. To ensure that all 3-key combinations used on a PC (e.g., CNTRL + ALT + DEL) are still possible, keyboard manufacturers using this method organize the key switch matrix accordingly (an example is given in *Figure 4*).



TL/DD/11091−14

FIGURE 5. Software Key Rollover

```
;       SOFTWARE KEY ROLLOVER
;
;LENGTHC: COUNTER  FOR NO.  OF BYTES (15 FOR A 16 BY  8  MATRIX)
;         WHICH  HAVE  TO BE COMPARED WITH THE  ACTUAL SCANNED
;         BYTE.
;LASTSCN: RAM LOCATION WHICH CONTAINS THE RESULT OF THE ACTUAL
;         SCANNED LINE
;
;PNTSCAN: RAM  LOCATION  WHICH CONTAINS A POINTER  TO  THE  RAM
;         CELL IN THE  SCAN HISTORY TABLE THAT STORES THE RESULT
;         OF  THE PREVIOUS SCAN FOR THE ACTUAL  SCANNED  MATRIX
;         LINE
;SCNLOT:  START ADRESS OF THE RAM SCAN HISTORY TABLE (16 BYTES)
;MATLEN:  MATRIX LENGTH (IN THIS CASE MATLEN=16dec)
;BITC  :  SHIFT COUNTER FOR BYTE SHIFT
;TYPSAV:  RAM ADRESS OF TYPEMATIC RATE SAVE REGISTER
;TYPST :  RAM ADRESS FOR TYPEMATIC RATE VALUE
;STATUS:  RAM ADRESS OF GENERAL STATUS FLAG REGISTER
;STAT2 :  RAM ADRESS OF GENERAL STATUS FLAG REGISTER 2
;TYPCO1:  RAM ADRESS OF REGISTER THAT CONTAINS TYPEMATIC KEY
;         MAKE CODE
;SCNCNT:  SCAN COUNTER FOR 16 MATRIX LINES
;
;
;
        .LOCAL
KEYROL:

        LD      LENGTHC,#00F    ;LOAD TABLE LENGTH COUNTER
        LD      X,#LASTSCN      ;POINT TO RAM LOCATION WHERE
                                ;RESULT OF PREVIOUS SCAN IS
                                ;STORED
        LD      A,PNTSCAN       ;LOAD POINTER TO ACTUAL SCAN
                                ;LINE
        INC     A
        X       A,B             ;POINT TO THE NEXT SCAN LINE
$NEXTB:
        IFBNE   #((SCNLOT+MATLEN)&00F) ;IF END OF HISTORY SCANTABLE
                                       ;IN RAM NOT REACHED
        JP      $1                     ;THEN OK
        LD      B,#SCNLOT       ;ELSE POINT TO BEGINNING OF TABLE
$1:
        LD      A,[X]           ;COMPARE NEW SCANNED MATRIX LINE
        OR      A,[B]           ;WITH ALL OTHER PREVIOUS SCANNED
                                ;BYTES IN TABLE
        IFEQ    A,#0FF          ;IF NO KEYS PRESSED IN
                                ;SAME INPUT LINE
        JP      $INCB           ;THEN COMPARE WITH NEXT BYTE
                                ;IN SCAN TABLE
                                ;ELSE LOOK IF MORE THAN
                                ;TWO KEYS ARE PRESSED
                                ;IN ONE OF THE TWO
                                ;COMPARED BYTES
        LD      A,[X]           ;LOAD 1ST OF COMP.BYTES
```

TL/DD/11091–1

```
        LD      BITC,#08        ;LOAD BIT COUNTER
$ZERO1:
        RRC     A
        IFNC                    ;IF 1 KEY PRESSED
        JP      $ZERO3          ;THEN TEST IF 2ND
                                ;KEY IS PRESSED
        DRSZ    BITC            ;IF NOT ALL BITS CHECKED
        JP      $ZERO1          ;THEN CONTINUE CHECK
        JP      $INCB

$ZERO2:
        RRC     A
        IFNC                    ;IF 2ND KEY PRESSED
        JP      $ENDLP          ;THEN ERROR: "GHOST KEY"
$ZERO3: DRSZ    BITC            ;IF NOT ALL BITS CHECKED
        JP      $ZERO2          ;THEN CONTINUE CHECK
$INCB:
        LD      A,[B+]          ;INC B
        DRSZ    LENGTHC         ;IF NEW SCANNED MATRIX LINE
                                ;NOT COMPARED WITH ALL OTHER
                                ;BYTES IN TABLE
        JP      $NEXTB          ;THEN COMP. WITH NEXT
                                ;BYTE IN TABLE
        SC                      ;IF ALL COMPARED, SET NO ERROR
                                ;FLAG
$ENDLP:
        LD      B,#STAT2        ;POINT TO STATUS FLAG REGISTER
        IFNC                    ;ERROR DURING THIS SCAN?
        JP      $ERROR          ;YES, DO ERROR PROCEDURE
        IFBIT   ERR2,[B]        ;ERROR DURING PREVIOUS SCANS,
                                ;BUT NO ERROR DURING THIS
                                ;SCAN?
        JP      $RESTORE        ;YES, RESTORE TYPEMATIC RATE
        RET

$RESTORE:
        RBIT    ERR2,[B]
        JSR     TSTOP           ;STOP TYPEMATIC TIMER
        LD      A,TYPSAV        ;LOAD SAVED TYPEMATIC VALUE
        X       A,TYPST         ;RESTORE OLD TYPEMATIC VALUE
        RET                     ;NO ERROR DURING THIS SCAN:
                                ;RETURN

$ERROR:
        IFBIT   ERR2,[B]        ;IF ERROR OCURRED ALREADY
                                ;DURING PREVIOUS SCAN
        JP      $ERREND         ;THEN DO NOTHING
        SBIT    ERR2,[B]        ;ELSE SET PREVIOUS ERROR FLAG
        LD      B,#TYPST        ;POINT TO TYPEMATIC VALUE
                                ;REGISTER
        LD      A,[B]
        X       A,TYPSAV        ;SAVE TYPEMATIC RATE/DELAY
        LD      [B],#07F        ;SET TYPEMATIC TO 1s DELAY,
                                ;2 CHARACTERS/s FOR ERROR CODE
```

TL/DD/11091–2

```
                                ;REPETITION
        LD      A,#000          ;IF SET2,3 ERROR CODE 00
        LD      B,#STATUS       ;POINT TO STATUS FLAG REGISTER
        IFBIT   SET1,[B]
        LD      A,#0FF          ;ELSE ERROR CODE FF
        X       A,TYPCO1        ;PUT IN TYPEMATIC BUFFER
        JSR     TSTART          ;INIT & START TYPEMATIC TIMER
$ERREND:
        LD      A,SCNCNT        ;INCREMENT SCAN COUNTER
        INCA
        X       A,SCNCNT
        RETSK                   ;RET AND SKIP FOR ROLLOVER ERROR
        .LOCAL
        .END
```

TL/DD/11091–3

Key Decoding and Encoding

After detection of a key change (pressing or releasing a key), the software first has to determine the physical location of the key in the key matrix. This decoding process is done by calculating an internal key number out of the key matrix column and row position of the changed key. At the same time, it is determined if the key has been pressed or released. A pressed or released key is then signaled by setting or resetting a ''key down'' flag in RAM. The internal key number and the ''key down'' status flag are the input parameters to the key encoding procedure. The internal key number is used to get the ''make'' code for the key out of a ROM look-up table, which has been matched to the physical matrix organization of the keyboard. If the ''key down'' flag is reset (key is released) the software calculates the key ''break'' code out of the previously fetched key ''make'' code. In this way, each pressed or released key is encoded with its appropriate ''make'' or ''break'' code, which is then written to the keyboard controllers 16 byte output buffer (FIFO) until the computer interface is ready to receive it. Before writing to the FIFO the software checks whether there is still enough capacity to store the key code.

Key Repetition

All keys are typematic (repetitive) by default. That means when a key is pressed and held down, the μC continues to send the ''make'' code for that key until it is released. When two or more keys are held down, only the code for the last key pressed is repeated. Typematic operation will stop when this key is released, even if other keys are still held down.

The default values for typematic operation are:

delay time = 500 ms
repetition rate = 10 characters/second,

where the delay time is the time which is inserted before a character is repeated for the first time.

Operating Protocol

There are two different transmission protocols for an MF2 keyboard: the AT transmission protocol and the XT transmission protocol. Data transmission to and from the keyboard is synchronous serial, the data format for the XT mode is:

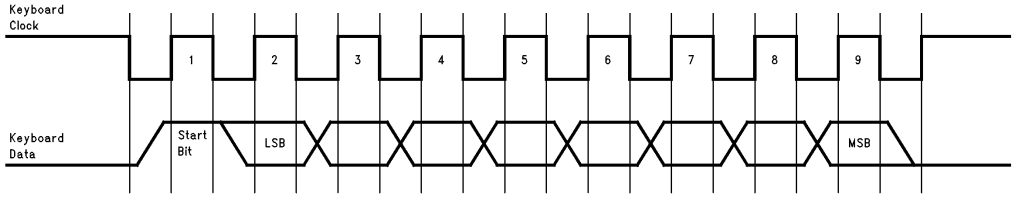
9 bits in length
1 start bit (high)
8 data bits (LSB first)

The data format for AT and PS/2 modes is:

11 bits in length
1 start bit (low)
8 data bits (LSB first)
1 parity bit (odd)
1 stop bit (high)

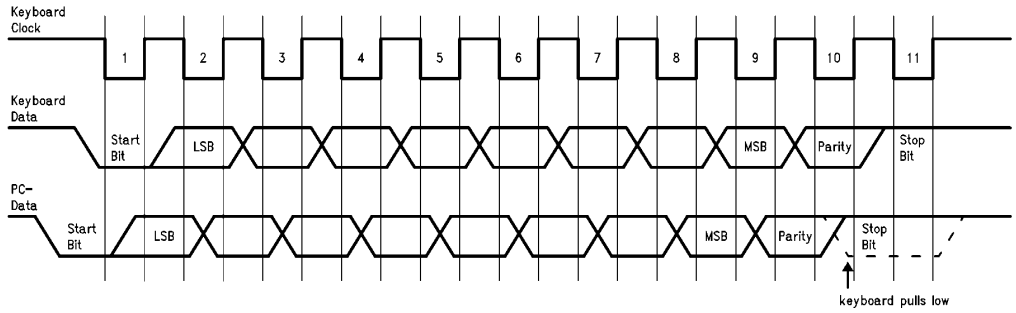
If no data is transmitted, both data and clock lines are in the high state. The clock signal is always provided by the keyboard. *Figure 6* shows the XT and the AT protocol timings.

XT-Protocol



TL/DD/11091–15

AT-Protocol



TL/DD/11091–16

FIGURE 6. XT and AT Protocol Timings

Keyboard Data Transmission in XT Format

At the falling edge of the clock, the start bit (high) is shifted out, followed by the 8 data bits (least significant bit first). Data is valid on the rising edge of the clock and changes after the falling edge of the clock.

Keyboard Data Transmission in AT Format

Before sending data, the keyboard monitors the clock and data lines. If the clock line is low, then the keyboard is disabled by the computer and no data is transmitted. The microcontroller continues to scan the keyboard and stores key data in its output buffer. If the data line is low, while the clock line is high, the computer requests to send and the keyboard goes into receive mode. The keyboard is only allowed to transmit data when both data and clock lines are high.

The keyboard pulls the data line low (start bit) and starts the clock. The 8 data bits (least significant bit first) are shifted out, followed by the parity (odd) and stop bit (high). Data is valid after the falling edge of the clock and changes after the rising edge of the clock. If no data is transmitted both data and clock lines are high. If the computer pulls the clock line low for at least 60 μs before the 10th bit is transmitted, the keyboard stops transmission and stores the aborted data in its output buffer.

```
;  SENDBY:   SEND BYTE TO COMPUTER
;INPUT PARAMETER:
;BYTSEN: RAM LOCATION CONTAINING THE
;        BYTE TO BE TRANSMITTED
;OUTPUT:
;        DATSEN FLAG IN STATUS REGISTER
;        1=BYTE SENT,0=BYTE NOT SENT
;PARCNT: PARITY COUNTER REGISTER
;BITC  : DATA LENGTH COUNTER FOR TRANSMISSION LOOP
;
;CLOCK HIGH TIME (=CLOCK LOW TIME) = 40us
;AT 3.58MHz CLOCK (INSTR. CYCLE = 2.79us)
;
;DATA REGISTER OF PORT G DATA AND CLOCK LINES IS
;PRESET WITH "0"

        .LOCAL

SendBy:
        LD      B,#STATUS       ;POINT TO  STATUS FLAG  REGISTER
        RBIT    DatSen,[B]      ;RESET "BYTE SEND" FLAG
        LD      A,BytSen        ;LOAD BYTE TO SEND
        LD      BITC,#009       ;DATA LENGTH
        IFBIT   PCXT,[B]        ;IF XT MODE
        JMP     PCMode          ;THEN JUMP TO XT
                                ;SEND ROUTINE
                                ;ELSE SEND AT PROTOCOL
$ATSEND:
        LD      PARCNT,#10      ;LOAD PARITY COUNTER
        LD      B,#PORTGP       ;POINT TO GPORT INPUT
                                ;REGISTER
WAITS:
```

TL/DD/11091–4

```
        IFBIT   ClockL,[B]      ;IF CLOCKLINE HIGH
        JP      $ClocOK         ;THEN OK
        JP      WAITS           ;ELSE KEYBOARD DISABLED:
                                ;WAIT

$ClocOK:
        IFBIT   DataLn,[B]      ;IF DATALINE IS HIGH
        JP      $DataOK         ;THEN OK
        RET                     ;ELSE PC SENDS DATA:
                                ;RETURN (GOTO RECEIVE)


$DataOK:
        LD      B,#PORTGC       ;POINT TO PORT G CONFIGURATION
                                ;REGISTER
        RC                      ;STARTBIT = 0
$SendBt:
        RBIT    ClockL,[B]      ;SET CLOCKLINE HIGH (TRI-STATE)
        IFBIT   ClockL,PORTGP   ;IF PC DOES NOT PULL CLOCKL LOW
        JP      $ClockH         ;THEN OK
        RBIT    DataLn,[B]      ;ELSE SET DATA LINE BACK TO HIGH
        RET                     ;STOP TO SEND

$ClockH:
        IFC                     ;IF BIT TO TRANSMIT = "1"
        JP      $DATHI          ;THEN DATALINE HIGH
        SBIT    DataLn,[B]      ;ELSE DATALINE LOW
        JP      $CLKLOW         ;SET CLOCKLINE LOW
$DATHI:
        RBIT    DataLn,[B]      ;SET DATALINE HIGH (TRI-STATE)

$CLKLOW:
        SBIT    ClockL,[B]      ;SET CLOCKLINE LOW
        IFC                     ;IF BIT=1
        DRSZ    PARCNT          ;THEN DECR. PARITY COUNTER
        RRC     A               ;SHIFT NEXT BIT INTO CARRY
        NOP
        DRSZ    BITC            ;IF NOT ALL BITS SENT
        JP      $SendBt         ;THEN TRANSMIT NEXT BIT
$PARITY:                        ;SEND PARITY BIT
        NOP                     ;DELAY
        NOP
        NOP
        RBIT    ClockL,[B]      ;SET CLOCKLINE HIGH
        IFBIT   00,PARCNT       ;IF NUMBER OF "1" = ODD
        JP      $DLOW           ;THEN PARITY = 0
        RBIT    DataLn,[B]      ;ELSE PARITY = 1
        JP      $CLKL2
$DLOW:
        SBIT    DataLn,[B]      ;SET DATALINE LOW
        NOP
$CLKL2:
        NOP                     ;DELAY
        NOP
```

TL/DD/11091–5

```
        NOP
        SBIT    ClockL,[B]      ;SET CLOCKLINE LOW
        JSR     DEL12           ;INSERT DELAY  12 INSTR. CYCLES

        RBIT    ClockL,[B]      ;SET CLOCKLINE HIGH

                                ;TRANSMIT STOP BIT
        RBIT    DataLn,[B]      ;SET DATA LINE HIGH (STOP BIT)
        JSR     DEL11           ;INSERT DELAY 11 INSTR. CYCLES
        SBIT    Clockl,[B]      ;SET CLOCKLINE LOW
        JSR     DEL12           ;INSERT DELAY 12 INSTR. CYCLES
$ENDSB:
        RBIT    ClockL,[B]      ;SET CLOCKLINE HIGH
        RBIT    DATALN,[B]      ;DATA HIGH (XT MODE)
        LD      B,#STATUS       ;POINT TO STATUS FLAG REG.
        SBIT    DatSen,[B]      ;SET DATA SENT FLAG

        LD      A,BYTSEN
        IFEQ    A,#0FE          ;IF SENT BYTE = RESEND
                                ;COMMAND
        RET                     ;THEN DON'T SAVE
        X       A,SENBYT        ;ELSE SAVE LAST SENT BYTE
                                ;IN SENBYT IN CASE PC ASKS
                                ;KEYBOARD TO RESEND
        RET

;XT TRANSMISSION PROTOCOL
PCMode:
        IFBIT   CLOCKL,PORTGP   ;CLOCKLINE HIGH?
        JP      $PCSND          ;YES,START TO SEND
        JMP     POWRUP          ;ELSE RESET
$PCSND:
        LD      B,#PORTGC
        SBIT    DATALN,[B]      ;DATA LINE LOW BEFORE
                                ;START TO SEND
        SC                      ;START BIT = 1
$PCSEND:
        SBIT    ClockL,[B]      ;CLOCKLINE LOW
        IFC                     ;IF BIT TO SEND=1
        JP      $DATH2          ;THEN SET DATALINE HIGH
        SBIT    DataLn,[B]      ;ELSE SET DATALINE LOW
        NOP                     ;DELAY
        NOP
        NOP
        NOP
        NOP
        NOP
        JP      $CLKHI
$DATH2:
        RBIT    DataLn,[B]      ;SET DATALINE HIGH
        IFBIT   DATALN,PORTGP   ;IF DATALINE HIGH
        JP      $CLKHI          ;THEN OK
                                ;ELSE KEYBOARD DISABLED
        RBIT    CLOCKL,[B]      ;CLOCKLINE HIGH
```

TL/DD/11091–6

```
        RET                     ;STOP TO SEND
$CLKHI:
        RBIT    ClockL,[B]      ;SET CLOCKLINE HIGH
        RRC     A               ;SHIFT NEXT BIT TO TRANSMIT
                                ;INTO CARRY
        NOP                     ;DELAY
        NOP
        NOP
        NOP
$PCOK:
        DRSZ    BITC            ;IF NOT ALL BITS SENDED
        JP      $PCSEND         ;THEN CONTINUE
        SBIT    CLOCKL,[B]      ;ELSE CLOCKLINE LOW
        SBIT    DATALN,[B]      ;DATA LOW
        JSR     DELAYD          ;10 INSTR. CYCLES DELAY
        JP      $ENDSB

DEL12:  NOP
DEL11:  NOP
DELAYD: RET
        .LOCAL
        .END
```

TL/DD/11091–7

Keyboard Receives Data

The keyboard can only receive data from the computer in AT-PS/2 mode. The computer pulls the data line low (start bit) after which the keyboard starts to shift out 11 clock pulses within 15 ms. Transmission has to be completed within 2 ms. Data from the computer changes after the falling edge of the clock and is valid before the rising edge of the clock. After the start bit, 8 data bits (least significant bit first), followed by the parity bit (odd) and the stop bit (high) are shifted out by the computer with the clock signal provided by the keyboard. The keyboard pulls the stop bit low in order to acknowledge the receipt of the data. If a transmission error occurred (parity error or similar) the keyboard issues the ''RESEND'' command to the PC.

```
;
;  RECDAT:   RECEIVE DATA COMMING FROM PC
;
;RETURN, IF PARITY ERROR
;
;RETURN SKIP , IF BYTE WAS RECEIVED
;WITHOUT ERROR
;
;BTRECV: RAM LOCATION CONTAINING THE
;        RECEIVED BYTE
;
;
;BITC  : RECEIVE LOOP COUNTER REGISTER
;PARCNT: PARITY COUNTER REGISTER
;

RecDat:

        CLRA
        LD      B,#PORTGC       ;B POINT TO PORT G
                                ;CONFIGURATION
        LD      X,#BTRECV       ;X POINT TO RECEIVED BYTE
                                ;RAM CELL
        LD      PARCNT,#10      ;LOAD PARITY COUNTER
        LD      BITC,#009       ;LOAD RECEIVE LOOP COUNTER
                                ;(8 DATABITS + 1 PARITY BIT)
        RC                      ;START BIT= "0"
RdByte:

        SBIT    ClockL,[B]      ;SET CLOCKLINE LOW
                                ;(CLOCK IN START BIT)
        IFC                     ;IF "1"-BIT RECEIVED
        DRSZ    PARCNT          ;THEN DECR. PARITY COUNTER
        RRC     A               ;SHIFT CARRY TO BIT 7 OF ACCU
        X       A,[X]           ;STORE RECEIVED BYTE
        LD      A,[X]           ;RESTORE AS  LONG AS NOT
                                ;FULL BYTE RECEIVED


        RBIT    ClockL,[B]      ;SET CLOCKLINE HIGH
;READ IN RECEIVED BIT
        RC                      ;RECEIVED BIT= "0"
        IFBIT   DataLn,PORTGP   ;IF DATALINE = "1"
        SC                      ;THEN RECEIVED BIT= "1"
        DRSZ    BITC            ;9 BITS RECEIVED?
        JP      RdByte          ;NO,LOOP

;CLOCK LOW PULSE AFTER PARITY HAS BEEN RECEIVED
        SBIT    ClockL,[B]      ;SET CLOCKLINE LOW
        JSR     DELAYD          ;INSERT 10 INSTR. CYCLES DELAY
        RBIT    ClockL,[B]      ;SET CLOCKLINE HIGH
;PC SENDS STOP BIT
        SBIT    DataLn,[B]      ;PULL STOP BIT LOW
```

TL/DD/11091–8

```
                                ;TO ACKNOWLEDGE RECEIPT OF BYTE
        JSR     DELAYD          ;INSERT DELAY
;CLOCK LOW PULSE (CLOCK ACKNOWLEDGE FOR PC)
        SBIT    ClockL,[B]      ;SET CLOCKLINE LOW
        JSR     DELAYD          ;INSERT DELAY
        RBIT    ClockL,[B]      ;SET CLOCKLINE HIGH

        RBIT    DataLn,[B]      ;RETURN DATA TO HIGH

;PARITY CHECK
        IFBIT   00,PARCNT       ;IF NO. OF RECEIVED DATA "1"=ODD
        JP      PAR0            ;THEN PARITY BIT MUST BE "0"
ParOne:                         ;ELSE PARITY BIT MUST BE "1"
        IFC                     ;IF RECEIVED PARITY BIT=1
        RETSK                   ;THEN OK,RETURN SKIP
        JP      PARERR          ;ELSE PARITY ERROR
PAR0:
        IFNC                    ;IF RECEIVED PARITY BIT =0
        RETSK                   ;THEN OK,RETURN SKIP
                                ;ELSE PARITY ERROR

ParErr: LD      BytSen,#0FE     ;LOAD "RESEND" CODE
        JSR     SByWPo          ;SEND RESEND CODE TO PC
        RET                     ;ERROR,RETURN
        .END
```

TL/DD/11091–9

Commands from the Computer

The following table shows the commands and their hexadecimal values the computer may send to the keyboard. Only AT-PS/2 compatible computers can send commands to the keyboard and the keyboard can only receive the commands when operated in the AT-mode.

The commands can be sent to the keyboard at any time. The keyboard responds within 20 ms to any valid transmission with ACK (FA Hex), except for the ECHO command where the keyboard responds with EE Hex, the RESEND command and the reserved commands.

Command	Hex Value
Set/Reset Mode Indicators	ED
Echo	EE
Reserved	EF
Select Alternate Code Set	F0
Reserved	F1
Read Keyboard ID	F2
Set Typematic Rate/Delay	F3
Enable	F4
Default Disable	F5
Set Default	F6
Set All Keys	
Typematic/No Break	F7
Make/Break/No Typematic	F8
Make/No Typematic	F9
Typem./Make/Br.	FA
Set Key Type	
Typematic/No Break	FB
Make/Break/No Typematic	FC
Make/No Typematic	FD
Resend	FE
Reset	FF

In the XT mode the keyboard only accepts the RESET command, which is assumed when the computer pulls the clock line low for at least 10 ms.

Commands to the Computer

The following table shows the commands and their hexadecimal values the keyboard may send to the system.

Command	Hex Value
Key Detection Error/ Buffer Overrun	00 (Code Sets 2 and 3)
Keyboard ID	83AB
BAT Completion Code	AA
BAT Failure Code	FC
Echo	EE
Acknowledge	FA
Resend	FE
Key Detection Error/ Buffer Overrun	FF (Code Set 1)

SUMMARY

When using National Semiconductor's microcontroller to implement the functions of an MF2 keyboard, very few external components are necessary. *Figure 2* shows the complete schematic of an MF2 keyboard based on the COP888CL. The implementation of software key rollover eliminates the need for decoupling diodes in the 16 by 8 key matrix. LED direct drive capability of the COP8 and a RC oscillator with tolerances tight enough to meet the requirements for a keyboard further reduce component count and price. Schmitt triggers on the ports used for the keyboards data and clock lines add additional security against transmission errors. Where low power consumption is the most important design factor (e.g., laptop or notebook computers) the COP8's M2CMOS technology and the multi-input wakeup feature offer a remarkable improvement over the NMOS controllers used in most of today's existing solutions.

National Semiconductor offers three chips tailored for the needs of a keyboard designer. Starting with the most price competitive 2.5k ROM device COP943C, an upgrade path is provided with the COP880C to 4k ROM. Both devices are intended for the use in standard MF2 desktop keyboards. The COP888CL is ideally suited for notebook or laptop keyboards, as it has special power saving features. The complete software for an MF2 keyboard as well as complete demo keyboards and keyboard evaluation boards for the COP888CL and COP943C/COP880C microcontrollers are available. Contact National Semiconductor's μC marketing or applications for further information.

APPENDIX I. KEY NUMBERS AND THEIR CORRESPONDING MAKE/BREAK CODES FOR ALL THREE CODE SETS

Key Position and Symbol		Table I (XT and PS/2 30)		Table II (AT and PS/2 50, 60, 80)		Table III (Terminal MODE)	
		Make	Break	Make	Break	Code	Type
01	~	29	A9	0E	F0–0E	0E	Typematic
02	! 1	02	82	16	F0–16	16	Typematic
03	@ 2	03	83	1E	F0–1E	1E	Typematic
04	# 3	04	84	26	F0–26	26	Typematic
05	$ 4	05	85	25	F0–25	25	Typematic
06	% 5	06	86	2E	F0–2E	2E	Typematic
07	∧ 6	07	87	36	F0–36	36	Typematic
08	& 7	08	88	3D	F0–3D	3D	Typematic
09	* 8	09	89	3E	F0–3E	3E	Typematic
10	(9	0A	8A	46	F0–46	46	Typematic
11	) 0	0B	8B	45	F0–45	45	Typematic
12	__ -	0C	8C	4E	F0–4E	4E	Typematic
13	+ =	0D	8D	55	F0–55	55	Typematic
15	B.S. ←	0E	8E	66	F0–66	66	Typematic
16	TAB	0F	8F	0D	F0–0D	0D	Typematic
17	Q	10	90	15	F0–15	15	Typematic
18	W	11	91	1D	F0–1D	1D	Typematic
19	E	12	92	24	F0–24	24	Typematic
20	R	13	93	2D	F0–2D	2D	Typematic
21	T	14	94	2C	F0–2C	2C	Typematic
22	Y	15	95	35	F0–35	35	Typematic
23	U	16	96	3C	F0–3C	3C	Typematic
24	I	17	97	43	F0–43	43	Typematic
25	O	18	98	44	F0–44	44	Typematic
26	P	19	99	4D	F0–4D	4D	Typematic
27	{ [	1A	9A	54	F0–54	54	Typematic
28	}]	1B	9B	5B	F0–5B	5B	Typematic
29*	| \	2B	AB	5D	F0–5D	5C	Typematic
30	Caps Lk	3A	BA	58	F0–58	14	Make/Break
31	A	1E	9E	1C	F0–1C	1C	Typematic
32	S	1F	9F	1B	F0–1B	1B	Typematic
33	D	20	A0	23	F0–23	23	Typematic
34	F	21	A1	2B	F0–2B	2B	Typematic
35	G	22	A2	34	F0–34	34	Typematic

Key Position and Symbol		Table I (XT and PS/2 30)		Table II (AT and PS/2 50, 60, 80)		Table III (Terminal MODE)	
		Make	Break	Make	Break	Code	Type
36	H	23	A3	33	F0–33	33	Typematic
37	J	24	A4	3B	F0–3B	3B	Typematic
38	K	25	A5	42	F0–42	42	Typematic
39	L	26	A6	4B	F0–4B	4B	Typematic
40	: ;	27	A7	4C	F0–4C	4C	Typematic
41	“ ’	28	A8	52	F0–52	52	Typematic
42**	| \	2B	AB	5D	F0–5D	53	Typematic
43	Enter (L)	1C	9C	5A	F0–5A	5A	Typematic
44	Shift (L)	2A	AA	12	F0–12	12	Typematic
45**	Macro	56	D6	61	F0–61	13	Typematic
46	Z	2C	AC	1A	F0–1A	1A	Typematic
47	X	2D	AD	22	F0–22	22	Typematic
48	C	2E	AE	21	F0–21	21	Typematic
49	V	2F	AF	2A	F0–2A	2A	Typematic
50	B	30	B0	32	F0–32	32	Typematic
51	N	31	B1	31	F0–31	31	Typematic
52	M	32	B2	3A	F0–3A	3A	Typematic
53	< ,	33	B3	41	F0–41	41	Typematic
54	> .	34	B4	49	F0–49	49	Typematic
55	? /	35	B5	4A	F0–4A	4A	Typematic
57	Shift (R)	36	B6	59	F0–59	59	Make/Break
58	Ctrl (L)	1D	9D	14	F0–14	11	Make/Break
60	Alt (L)	38	B8	11	F0–11	19	Make/Break
61	Space	39	B9	29	F0–29	29	Typematic
62	Alt (R)	E0–38	E0–B8	E0–11	E0–F0–11	39	Make
64	Ctrl (R)	E0–1D	E0–9D	E0–14	E0–F0–14	58	Make
90	Num Lk	45	C5	77	F0–77	76	Make
91	7 Home	47	C7	6C	F0–6C	6C	Make
92	4 ←	4B	CB	6B	F0–6B	6B	Make
93	1 End	4F	CF	69	F0–69	69	Make
96	8 ↑	48	C8	75	F0–75	75	Make
97	5	4C	CC	73	F0–73	73	Make
98	2 ↓	50	D0	72	F0–72	72	Make
99	0 Ins	52	D2	70	F0–70	70	Make
100	*	37	B7	7C	F0–7C	7E	Make

*101-Keyboard only

**102-Keyboard only

Key Position and Symbol		Table I (XT and PS/2 30)		Table II (AT and PS/2 50, 60, 80)		Table III (Terminal MODE)	
		Make	Break	Make	Break	Code	Type
101	9 Pg UP	49	C9	7D	F0–7D	7D	Make
102	6 →	4D	CD	74	F0–74	74	Make
103	3 Pg DN	51	D1	7A	F0–7A	7A	Make
104	Del	53	D3	71	F0–71	71	Make
105	-	4A	CA	7B	F0–7B	84	Make
106	+	4E	CE	79	F0–79	7C	Make
108	Enter	E0–1C	E0–9C	E0–5A	E0–F0–5A	79	Typematic
110	Esc	01	81	76	F0–76	08	Make
112	F1	3B	BB	05	F0–05	07	Make
113	F2	3C	BC	06	F0–06	0F	Make
114	F3	3D	BD	04	F0–04	17	Make
115	F4	3E	BE	0C	F0–0C	1F	Make
116	F5	3F	BF	03	F0–03	27	Make
117	F6	40	C0	0B	F0–0B	2F	Make
118	F7	41	C1	83	F0–83	37	Make
119	F8	42	C2	0A	F0–0A	3F	Make
120	F9	43	C3	01	F0–01	47	Make
121	F10	44	C4	09	F0–09	4F	Make
122	F11	57	D7	78	F0–78	56	Make
123	F12	58	D8	07	F0–07	5E	Make
125	Scr Lk	46	C6	7E	F0–7E	5F	Make

Key Position and Symbol		Cursor Pad <NUM Lock Off/Shift Off> or <NUM Lock On/Shift On>				Table III (Terminal Mode)	
		Table I (XT and PS/2 30)		Table II (AT and PS/2 50, 60, 80)			
		Make	Break	Make	Break	Code	Type
75	Insert	E0–52	E0–D2	E0–70	E0–F0–70	67	Make
76	Delete	E0–53	E0–D3	E0–71	E0–F0–71	64	Typematic
79	←	E0–4B	E0–CB	E0–6B	E0–F0–6B	61	Typematic
80	Home	E0–47	E0–C7	E0–6C	E0–F0–6C	6E	Make
81	End	E0–4F	E0–CF	E0–69	E0–F0–69	65	Make
83	↑	E0–48	E0–C8	E0–75	E0–F0–75	63	Typematic
84	↓	E0–50	E0–D0	E0–72	E0–F0–72	60	Typematic
85	PG UP	E0–49	E0–C9	E0–7D	E0–F0–7D	6F	Make
86	PG DN	E0–51	E0–D1	E0–7A	E0–F0–7A	6D	Make
89	→	E0–4D	E0–CD	E0–74	E0–F0–74	6A	Typematic

***. Cursor Pad Key—<NUM Lock On/Shift Off>**

Table I: Make Code = = E0–2A–Make Code
Break Code = = Break Code–E0–AA

Table II: Make Code = = E0–12–Make Code
Break Code = = Break Code E0–F0–12

***. Cursor Pad Key—<NUM Lock Off/Shift On>**

Table I: Make Code = E0–AA–Make Code
Break Code = Break Code–E0–2A

Table II: Make Code = E0–F0–12–Make Code
Break Code = Break Code E0–12

Key Code of "Pause", "PRTSC" and "/" Keys

TABLE I. XT and PS/2 30

Key Position and Symbols		Make	Break
126	Pause	E1–1D–45–E1–9D–C5	No Break Code (Make Only)
	Ctrl-"Pause"	E0–46–E0–C6	No Break Code (Make Only)
124	Print Screen	E0–2A–E0–37	E0–B7–E0–AA
	Shift-"PRTSC"	E0–37	E0–B7
	Ctrl-"PRTSC"	E0–37	E0–B7
	Alt-"PRTSC"	54	D4
95	/	E0–35	E0–B5
	Shift-"/"	E0–AA–E0–35	E0–B5–E0–2A

TABLE II. AT and PS/2 50, 60, 80

Key Position and Symbols		Make	Break
126	Pause	E1–14–77–E1–F0–14–F0–77	No Break Code (Make Only)
	Ctrl-"Pause"	E0–7E–E0–F0–7E	No Break Code (Make Only)
124	Print Screen	E0–12–E0–7C	E0–F0–7C–E0–F0–12
	Shift-"PRTSC"	E0–7C	E0–F0–7C
	Ctrl-"PRTSC"	E0–7C	E0–F0–7C
	Alt-"PRTSC"	84	F0–84
95	/	E0–4A	E0–F0–4A
	Shift-"/"	E0–F0–12–E0–4A	E0–F0–4A–E0–12

TABLE III. Terminal Mode

Key Position and Symbols		Code	Type
126	Pause	62	Make
124	Print Screen	57	Make
95	/	77	Make

APPENDIX II. REFERENCES

1. IBM Technical Reference Manuals XT, AT and PS/2
2. Chicony, Chicony Keyboards General Specification, 1988
3. C′ T Magazin fuer Computertechnik, No. 6, 1988, pages 148ff. No. 7, 1988, pages 178ff. Martin Gerdes, "Knoepfchen, Knoepfchen"

National Semiconductor Corporation
2900 Semiconductor Drive
P.O. Box 58090
Santa Clara, CA 95052-8090
Tel: 1(800) 272-9959
TWX: (910) 339-9240

National Semiconductor GmbH
Livry-Gargan-Str. 10
D-82256 Fürstenfeldbruck
Germany
Tel: (81-41) 35-0
Telex: 527649
Fax: (81-41) 35-1

National Semiconductor Japan Ltd.
Sumitomo Chemical
Engineering Center
Bldg. 7F
1-7-1, Nakase, Mihama-Ku
Chiba-City,
Ciba Prefecture 261
Tel: (043) 299-2300
Fax: (043) 299-2500

National Semiconductor Hong Kong Ltd.
13th Floor, Straight Block,
Ocean Centre, 5 Canton Rd.
Tsimshatsui, Kowloon
Hong Kong
Tel: (852) 2737-1600
Fax: (852) 2736-9960

National Semiconductores Do Brazil Ltda.
Rue Deputado Lacorda Franco
120-3A
Sao Paulo-SP
Brazil 05418-000
Tel: (55-11) 212-5066
Telex: 391-1131931 NSBR BR
Fax: (55-11) 212-1181

National Semiconductor (Australia) Pty, Ltd.
Building 16
Business Park Drive
Monash Business Park
Nottinghill, Melbourne
Victoria 3168 Australia
Tel: (3) 558-9999
Fax: (3) 558-9998