Keyboard Encoder Circuits

MICROPROCESSOR MATES WITH MOS/LSI KEYBOARD ENCODER

ABSTRACT

This application note is intended to show how to interface a keyboard to the IMP-16 microprocessor for the purpose of text editing. An example which includes suggested hardware and software is presented to illustrate data inputting from the keyboard to the microprocessor. This example can be used either with the IMP-16 chip set or with the IMP-16C/200 or IMP-16C/300 card.

INTRODUCTION

The MM5740 keyboard encoder interfaced to an IMP-16C card microprocessor provides a very cost-effective means of data entry that takes full advantage of the benefits of MOS/LSI technology. The MM5740 is a complete keyboard interface system capable of providing quad mode* 90 key keyboard encoding in a single integrated circuit. This chip detects a key switch closure and translates it into a coded output while providing all of the necessary functions for modern keyboard system design. Data and control outputs are directly compatible with the TTL logic inputs on the IMP-16C. Characters are read from the keyboard into the read/write memory on the IMP-16C card by means of a program contained in PROM's on the card or in external memory. The characters may be reformatted, edited, converted to binary and processed, transferred to a floppy disk or cassette for more permanent recording, or transmitted to a central computer facility. Typical applications include text editing typewriters, alphanumeric CRT display controllers, remote terminal controllers, data entry and recording systems, operators console in man-machine interactive systems, supervisory or process control systems. Further application information is contained in *AN-80 MOS Keyboard Encoding* and *AN-124 IMP-16 Peripheral Interfacing Simplified. Figure 1* is a functional diagram of a keyboard/IMP-16C interface using the LSI keyboard encoder.

INTERFACE CONSIDERATIONS

The Keyboard

Connecting a physical keyboard to the MM5740 will be covered briefly in the following discussion. A more comprehensive treatment is detailed in AN-80, pgs 3 - 4. For this discussion, reference should be made to *Figure 2* which details the pin connections.

The matrix drive ($X_1 - X_9$) and sense ($Y_1 - Y_{10}$) lines are normally connected to each other via the switch matrix. These lines detect contact closure and sense the key that was depressed. The corresponding character is obtained from a read only memory in the MM5740 which has been mask programmed for the desired code. Nine bits are available for each character. Bits 0 to 7 are generally information bits while bits 8 and 9 may be used for parity or special character control. When a valid key is entered the corresponding 9-bit character is stored internally in latches within the MM5740. After a delay of one bit time (one clock period) the data strobe (pin 13) signal will go high, indicating that data is ready and stored in the output latches. This signal alerts the IMP-16C that the character may now be taken. The function of the data strobe control input (pin 14) is to control the resetting of the data strobe once it has been activated. The output enable (pin 15) serves as the TRI-STATE® control for the code data output lines (B_1 to B_9) and is used to control the resetting of the data strobe output.

To minimize response time, the MM5740 is operated in the pulse data strobe mode. The output enable is tied to ground so that the outputs are always enabled. The data strobe is tied directly to the data strobe control. With this connection, a pulse which is one bit time wide will appear on the data strobe line to indicate available data is present. With a 200 kHz clock, one bit time translates into a 5 μs data strobe pulse.

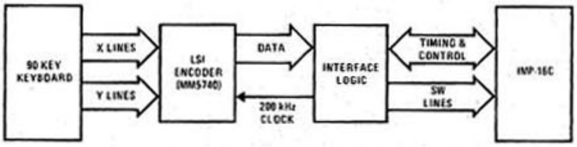
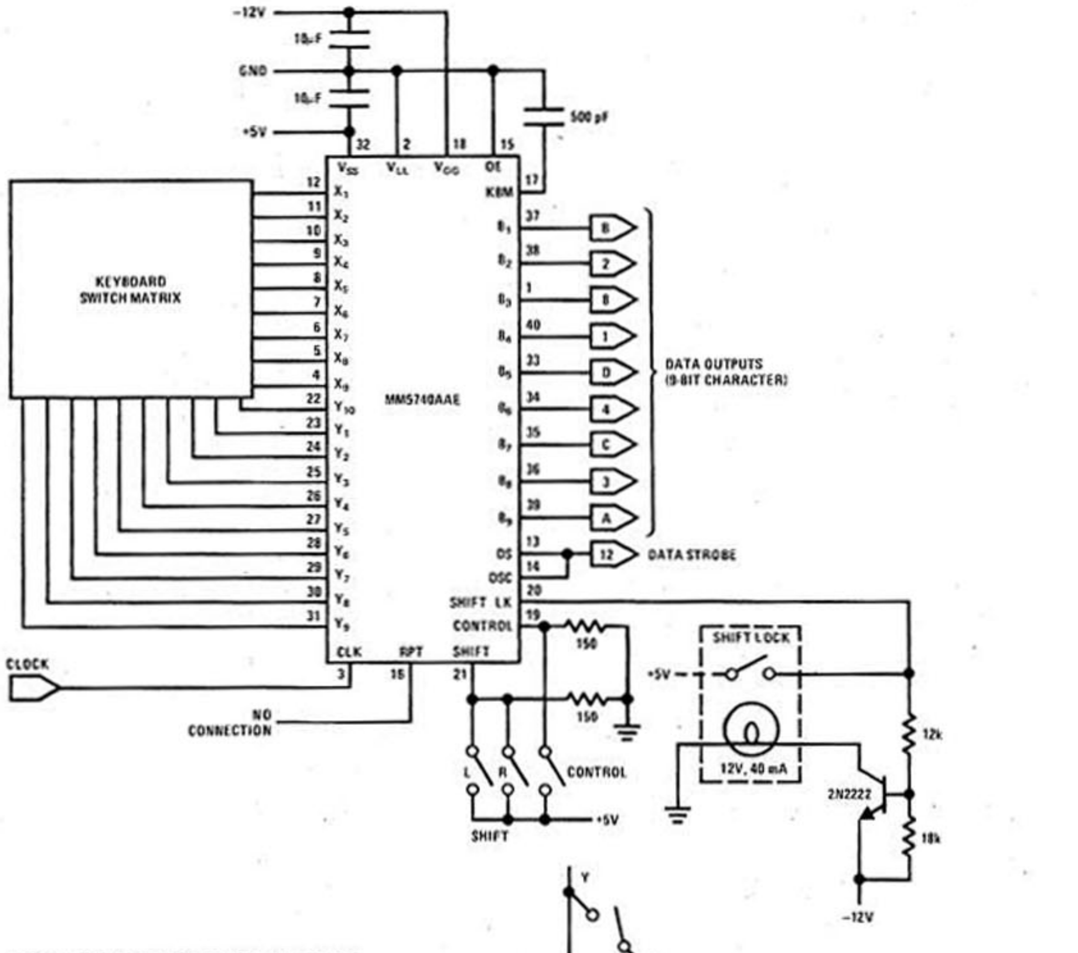


FIGURE 1. Functional Diagram

*Quad mode means the four basic keyboard modes which are; UNSHIFT, SHIFT, CONTROL, SHIFT CONTROL.



Note 1: MM5740AAE denotes inverted ASCII code, even parity, and N key rollover.
Note 2: Clock frequency = 200 kHz.
Note 3: Scan period = 450µs.
Note 4: Key bounce mask time = 4 ms.
Note 5: Data strobe = 5µs pulse.

FIGURE 2. MM5740 Pin Connections

In the following sample interface design the MM5740 chip and several discrete components are mounted on a communications keyboard. A cable from the 40 pin connector on the keyboard to an 8 1/2" x 11" interface board provides the physical communications link to the processor. The interface board has space available for components to implement a cassette and CRT interface for text editing applications. Pages of text could be stored as cassette records, called up by the keyboard and displayed on the CRT. Appropriate keyboard commands could be programmed to edit the page. Lines could be inserted, deleted, copied or moved as required. The finished page could be restored on the cassette. *Figure 3* is a schematic diagram of the keyboard interface logic board.

MM5740–IMP-16C INTERFACE

Three instructions are necessary for the IMP-16C to detect that a character is ready for input and to obtain that character. These instructions are given below:

```
LI    3, X '80    ;DEVICE ADDRESS IN AC3
BOC  13, . + 0    ;WAIT FOR CHARACTER READY
RIN   0           ;INPUT CHARACTER INTO AC0
```

The first instruction sets the peripheral device address of the keybaord (X'80) into accumulator 3 (AC3). This is necessary for proper execution of the RIN instruction (AC3 is added to the sign extended displacement field of the RIN instruction and sent to the peripheral over the ADX lines). The address was chosen so as not to be in conflict with any of the IMP-16P peripherals.

The BOC instruction is essentially a test for keyboard character ready. The data strobe output (DSO) from the keyboard (cable connector pin 12) is stored in a set-reset latch built from cross coupled NAND gates (see *Figure 3*). This is because the DSO pulse width is one clock period or 5.0µs and the processor might not detect DSO in the required time. Refer to *Figure 4* for IMP-16C/MM5740 timing. The complement output of the latch ($\overline{Q}$) is connected to jump condition 13 (JC13). The BOC instruction tests for JC13 and branches to the PC relative address specified in the displacement field if the condition is true. Normally JC13 is true; when a key is pressed DSO goes high which forces $\overline{Q}$ low. The jump condition will then be false and the next instruction executed. This next instruction is a RIN 0 which takes the character from the keyboard encoder (B_1 to B_8) into AC0. Thus, this program is in a one-word BOC loop until a key is pressed.

Execution of the RIN instruction causes:

1. The peripheral device address and order code to be placed on the ADX lines at T4 of microcycle 6 (see *Figure 1-3, IMP-16C Application Manual* Supplement 1, pg. 1 - 3. There are eight timing pulses, T1 to T8, each microcycle. The RIN instruction requires 7 of these microcycles).
2. The RDP (Read Peripheral) flag to be pulsed at T2 of microcycle 7. This is used as a peripheral input gating signal.

The peripheral address and order codes on the ADX lines are set into TTL latches on the IMP-16C during RIN microcycle 6. The ADX lines are sent to all peripherals, but only the one whose address is specified

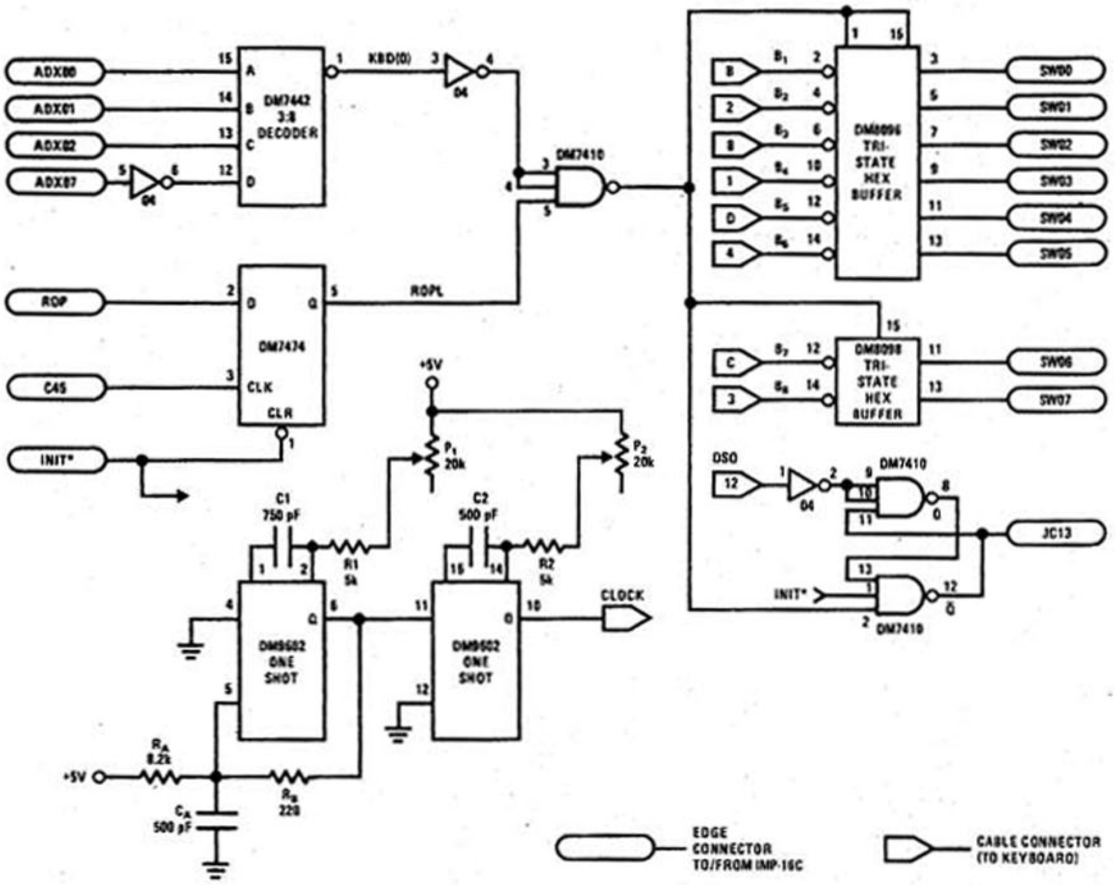


FIGURE 3. Text Editing Keyboard (TEK) Interface Logic for IMP-16C

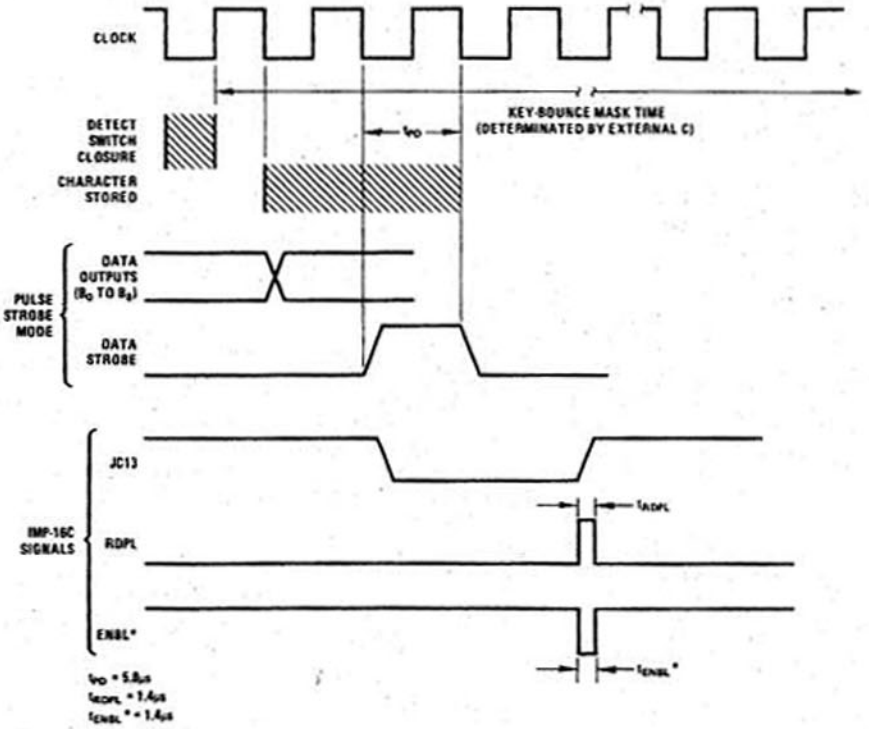


FIGURE 4. MM5740/IMP-C Timing Diagram

will respond. A BCD to binary decoder (DM7442) is used to select one of eight possible order codes. This provides modular expansion capability if new peripherals (keyboards, CRT's, cassettes, printers) are added to the keyboard microprocessor system. The RDP signal is latched (RDPL) on the interface to guarantee that it will be valid at T7 of RIN microcycle 7, when data is taken by the processor. At this time the address and order code is valid and the ENBL signal goes low. This signal enables the TRI-STATE buffers (DM8096, DM8098) which complement the inverted ASCII keyboard data (B_1 to B_8) and place it on the SW bus to the processor. The data is taken by the processor at T7 and transferred into AC0 bits 0 to 7. At this point, one character has been obtained by the processor. The ENBL signal is also used to reset the data strobe latch which makes $\overline{Q}$ high and JC13 true. This reconditions the IMP-16C to be ready for the next character.

The MM5740's clock input (CLK) is provided by a dual one shot (DM9602) connected as an oscillator. A 200 kHz square wave is generated using the logic shown in *Figure 3.*

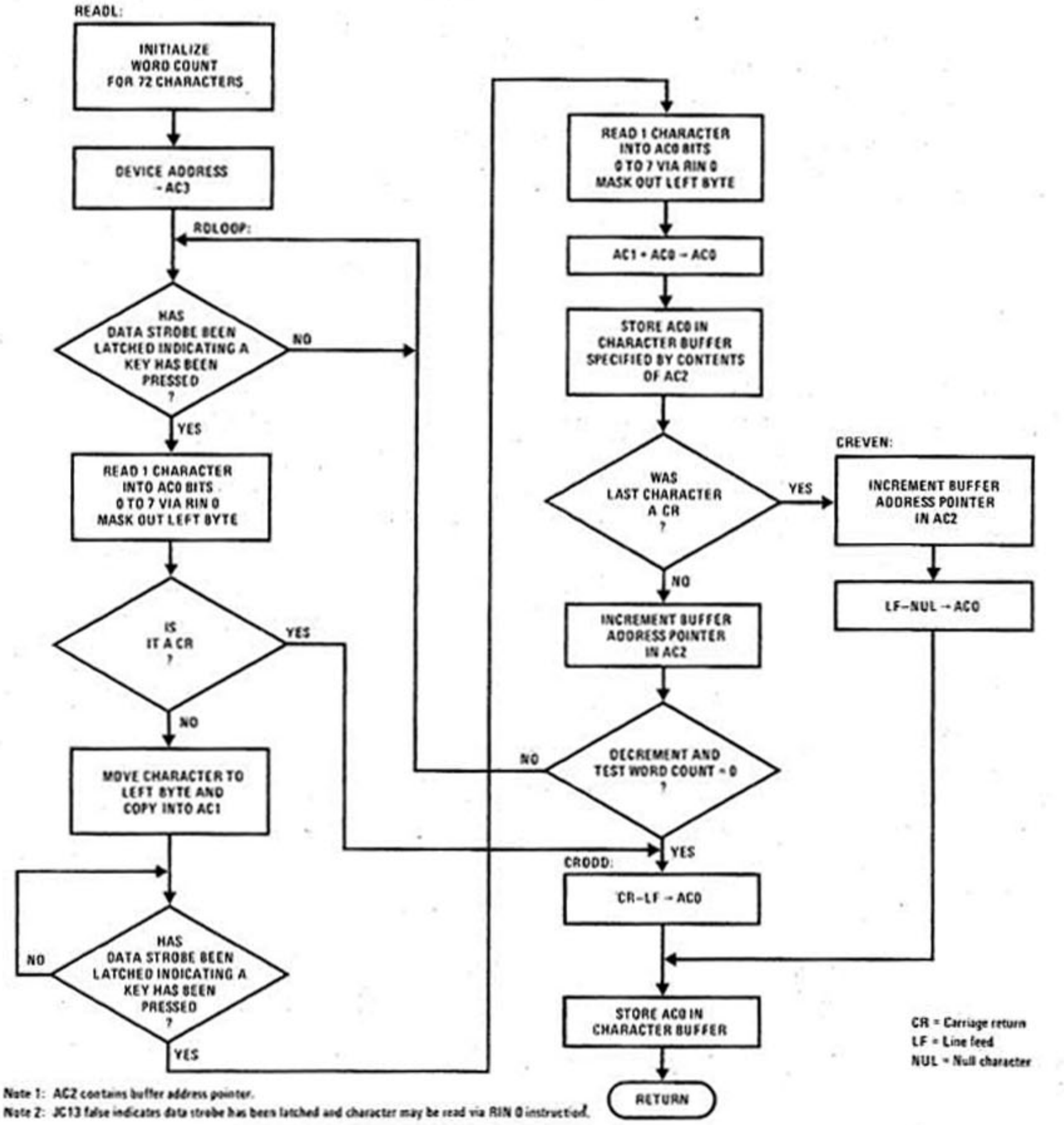
THE PROGRAM

In addition to the three instructions given, a control program is necessary to pack, store and count characters and insert line delimeters—carriage return (CR) and line feed (LF). A flow chart and coding for the program are given in *Figures 5* and *6*.

A line of text is terminated by a CR or when 72 characters have been entered. The CR—LF is inserted and an address pointer is incremented to designate the start of the next line. At this point, the user may request that the last line or entire message be typed on the teletype using the MESG routine in the TTY 16P PROM. Editing functions such as insert, delete, replace, copy, or move lines could be provided if the information was to be output to a CRT, cassette or floppy disc. Although the keyboard encoder (MM5740) used was mask programmed for inverted ASCII code with even parity, any code could be used.

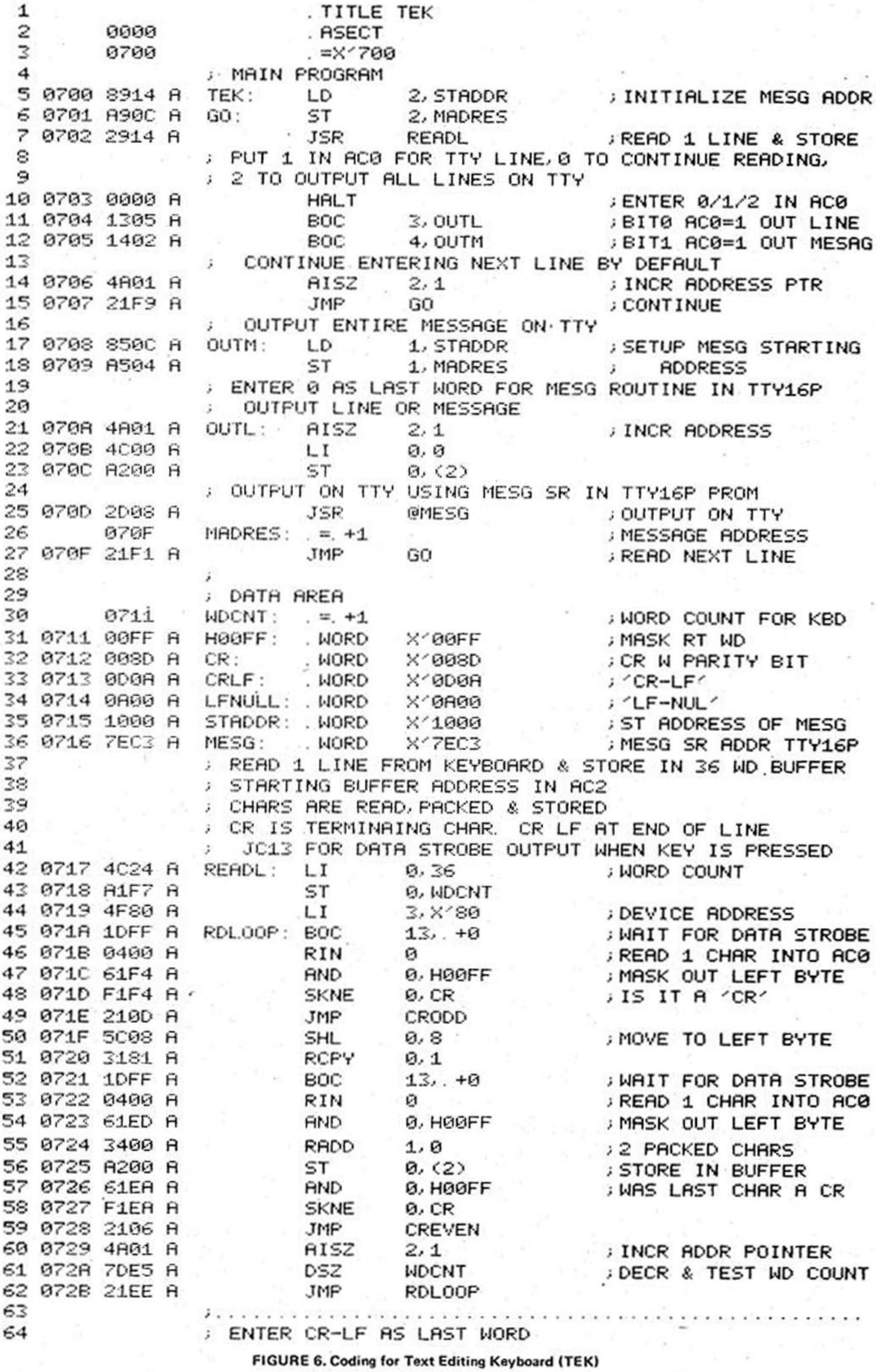
CONCLUSION

The example below demonstrates a keyboard/microprocessor interface taking full advantage of the benefits of LSI technology—small size, increased reliability, fewer interconnections and much more functional capability per unit cost. These advantages may be exploited in a wide range of man-machine or operator interaction systems.


READL:
INITIALIZE WORD COUNT FOR 72 CHARACTERS
DEVICE ADDRESS - AC3
RDLOOP:
HAS DATA STROBE BEEN LATCHED INDICATING A KEY HAS BEEN PRESSED ?
NO
YES
READ 1 CHARACTER INTO AC0 BITS 0 TO 7 VIA RIN 0 MASK OUT LEFT BYTE
IS IT A CR ?
YES
NO
MOVE CHARACTER TO LEFT BYTE AND COPY INTO AC1
HAS DATA STROBE BEEN LATCHED INDICATING A KEY HAS BEEN PRESSED ?
NO
YES
READ 1 CHARACTER INTO AC0 BITS 0 TO 7 VIA RIN 0 MASK OUT LEFT BYTE
AC1 + AC0 → AC0
STORE AC0 IN CHARACTER BUFFER SPECIFIED BY CONTENTS OF AC2
WAS LAST CHARACTER A CR ?
YES
NO
CREVEN:
INCREMENT BUFFER ADDRESS POINTER IN AC2
LF–NUL → AC0
INCREMENT BUFFER ADDRESS POINTER IN AC2
NO
DECREMENT AND TEST WORD COUNT = 0 ?
YES
CRODD:
CR–LF → AC0
STORE AC0 IN CHARACTER BUFFER
RETURN
CR = Carriage return
LF = Line feed
NUL = Null character

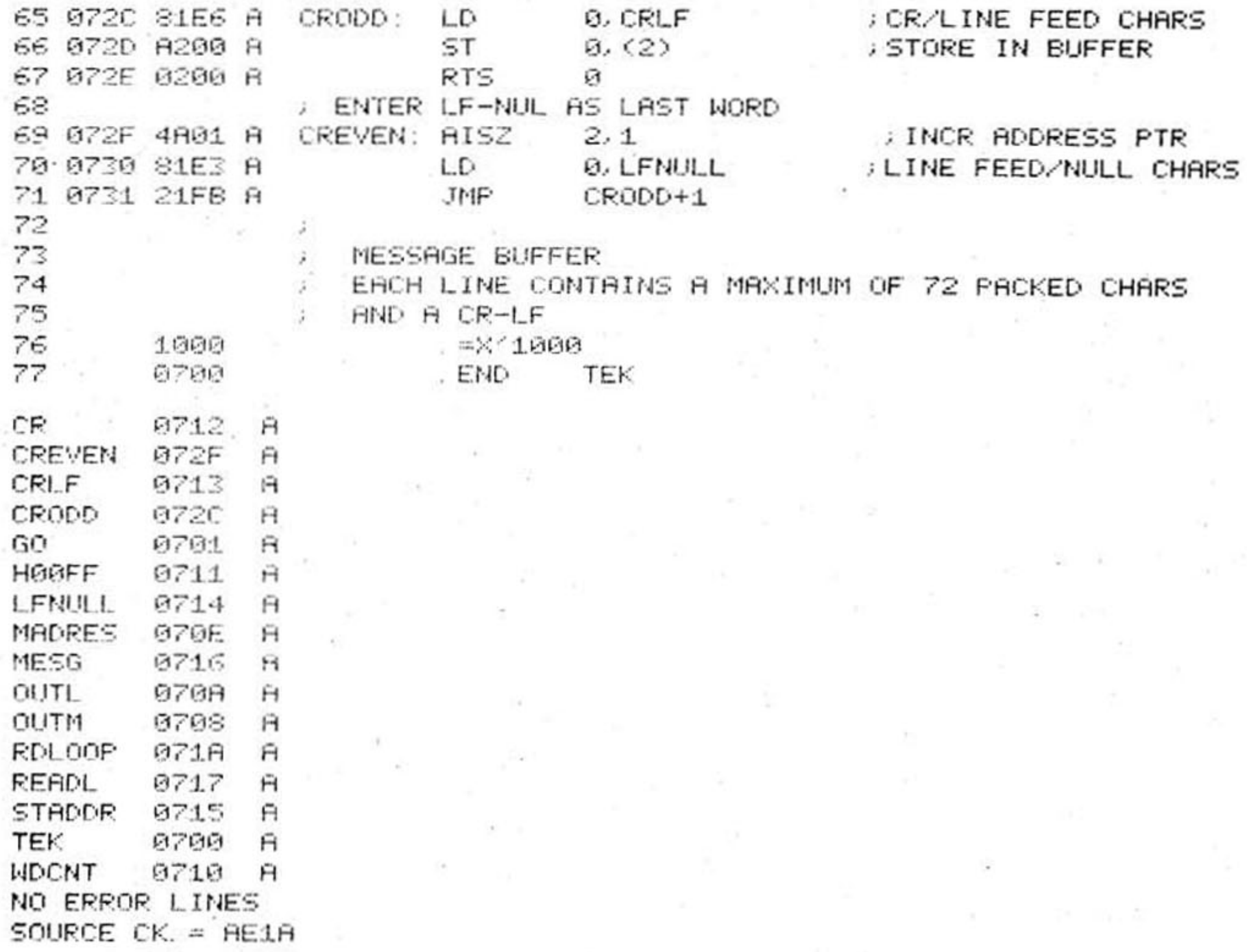
Note 1: AC2 contains buffer address pointer.
Note 2: JC13 false indicates data strobe has been latched and character may be read via RIN 0 instruction.

FIGURE 5. Flowchart of Subroutine (READL) that Reads One Line from the Keyboard



```
                         .TITLE TEK
     0000                .ASECT
     0700                .=X'700
                 ; MAIN PROGRAM
0700 8914 A  TEK:      LD      2,STADDR        ;INITIALIZE MESG ADDR
0701 A90C A  GO:       ST      2,MADRES
0702 2914 A            JSR     READL           ;READ 1 LINE & STORE
                 ; PUT 1 IN AC0 FOR TTY LINE,0 TO CONTINUE READING,
                 ; 2 TO OUTPUT ALL LINES ON TTY
0703 0000 A            HALT                    ;ENTER 0/1/2 IN AC0
0704 1305 A            BOC     3,OUTL          ;BIT0 AC0=1 OUT LINE
0705 1402 A            BOC     4,OUTM          ;BIT1 AC0=1 OUT MESAG
                 ;  CONTINUE ENTERING NEXT LINE BY DEFAULT
0706 4A01 A            AISZ    2,1             ;INCR ADDRESS PTR
0707 21F9 A            JMP     GO              ;CONTINUE
                 ;  OUTPUT ENTIRE MESSAGE ON TTY
0708 850C A  OUTM:     LD      1,STADDR        ;SETUP MESG STARTING
0709 A504 A            ST      1,MADRES        ;    ADDRESS
                 ; ENTER 0 AS LAST WORD FOR MESG ROUTINE IN TTY16P
                 ;  OUTPUT LINE OR MESSAGE
070A 4A01 A  OUTL:     AISZ    2,1             ;INCR ADDRESS
070B 4C00 A            LI      0,0
070C A200 A            ST      0,(2)
                 ; OUTPUT ON TTY USING MESG SR IN TTY16P PROM
070D 2D08 A            JSR     @MESG           ;OUTPUT ON TTY
     070F    MADRES:  .=.+1                    ;MESSAGE ADDRESS
070F 21F1 A            JMP     GO              ;READ NEXT LINE
                 ;
                 ; DATA AREA
     0711    WDCNT:   .=.+1                    ;WORD COUNT FOR KBD
0711 00FF A  H00FF:   .WORD    X'00FF          ;MASK RT WD
0712 008D A  CR:      .WORD    X'008D          ;CR W PARITY BIT
0713 0D0A A  CRLF:    .WORD    X'0D0A          ;'CR-LF'
0714 0A00 A  LFNULL:  .WORD    X'0A00          ;'LF-NUL'
0715 1000 A  STADDR:  .WORD    X'1000          ;ST ADDRESS OF MESG
0716 7EC3 A  MESG:    .WORD    X'7EC3          ;MESG SR ADDR TTY16P
                 ; READ 1 LINE FROM KEYBOARD & STORE IN 36 WD BUFFER
                 ; STARTING BUFFER ADDRESS IN AC2
                 ; CHARS ARE READ,PACKED & STORED
                 ; CR IS TERMINAING CHAR.  CR LF AT END OF LINE
                 ;  JC13 FOR DATA STROBE OUTPUT WHEN KEY IS PRESSED
0717 4C24 A  READL:    LI      0,36            ;WORD COUNT
0718 A1F7 A            ST      0,WDCNT
0719 4F80 A            LI      3,X'80          ;DEVICE ADDRESS
071A 1DFF A  RDLOOP:   BOC     13,.+0          ;WAIT FOR DATA STROBE
071B 0400 A            RIN     0               ;READ 1 CHAR INTO AC0
071C 61F4 A            AND     0,H00FF         ;MASK OUT LEFT BYTE
071D F1F4 A            SKNE    0,CR            ;IS IT A 'CR'
071E 210D A            JMP     CRODD
071F 5C08 A            SHL     0,8             ;MOVE TO LEFT BYTE
0720 3181 A            RCPY    0,1
0721 1DFF A            BOC     13,.+0          ;WAIT FOR DATA STROBE
0722 0400 A            RIN     0               ;READ 1 CHAR INTO AC0
0723 61ED A            AND     0,H00FF         ;MASK OUT LEFT BYTE
0724 3400 A            RADD    1,0             ;2 PACKED CHARS
0725 A200 A            ST      0,(2)           ;STORE IN BUFFER
0726 61EA A            AND     0,H00FF         ;WAS LAST CHAR A CR
0727 F1EA A            SKNE    0,CR
0728 2106 A            JMP     CREVEN
0729 4A01 A            AISZ    2,1             ;INCR ADDR POINTER
072A 7DE5 A            DSZ     WDCNT           ;DECR & TEST WD COUNT
072B 21EE A            JMP     RDLOOP
                 ;.............................................
                 ; ENTER CR-LF AS LAST WORD
```

FIGURE 6. Coding for Text Editing Keyboard (TEK)



```
072C 81E6 A  CRODD:   LD      0,CRLF              ;CR/LINE FEED CHARS
072D A200 A           ST      0,(2)               ;STORE IN BUFFER
072E 0200 A           RTS     0
             ; ENTER LF-NUL AS LAST WORD
072F 4A01 A  CREVEN:  AISZ    2,1                  ;INCR ADDRESS PTR
0730 81E3 A           LD      0,LFNULL            ;LINE FEED/NULL CHARS
0731 21FB A           JMP     CRODD+1
             ;
             ;  MESSAGE BUFFER
             ;  EACH LINE CONTAINS A MAXIMUM OF 72 PACKED CHARS
             ;  AND A CR-LF
     1000             .=X'1000
     0700             .END    TEK

CR       0712  A
CREVEN   072F  A
CRLF     0713  A
CRODD    072C  A
GO       0701  A
H00FF    0711  A
LFNULL   0714  A
MADRES   070E  A
MESG     0716  A
OUTL     070A  A
OUTM     0708  A
RDLOOP   071A  A
READL    0717  A
STADDR   0715  A
TEK      0700  A
WDCNT    0710  A
NO ERROR LINES
SOURCE CK. = AE1A
```

FIGURE 6. Coding for Text Editing Keyboard (TEK) (Continued)

Keyboard Encoder Circuits

MOS ENCODER PLUS PROM YIELD QUICK TURNAROUND KEYBOARD SYSTEMS*

INTRODUCTION

Most modern keyboard designs employ MOS/LSI keyboard encoder IC's to implement all the necessary electronic functions. The key codes specified by the customer are programmed into a read only memory which is an inherent part of the encoder. Although some common encoder formats are available off the shelf, such as ASR33 teletype (MM5740AAE or MM5740AAF), there are many instances where variations of common formats are needed. Since these formats are mask programmed into the keyboard encoder, there is a certain amount of lead time (approximately 12 weeks) before a customer receives his final circuit.

By using a binary coded keyboard encoder in conjunction with a programmable read only memory, customers can build prototype keyboard systems or fill small volume orders in minimum time. This approach keeps all the encoding electronics and timing the same as in the final system, so that a minimum of redesign is necessary to configure the actual final version. This is done when the keyboard encoder with the final mask programmed key codes is received. In addition, the usefulness of being able to reassign key codes quickly in the PROM makes system debugging and alteration an easy task.

The basic configuration for this implementation is shown in the simplified block diagram of *Figure 1.* The key switches and all timing signals are configured in the normal manner. The keyboard encoder chip will emit binary codes for each valid keyswitch closure. These binary outputs are used as addresses for the PROM which is programmed with the desired actual code for each keyswitch. Each key closure is transformed first to an address by the encoder and then to the final code by the PROM. In this manner, a general design is possible, with the only variable being the contents of the PROM which is easily and quickly programmed. When changes are necessary, the PROM may be erased and reprogrammed quickly making it an easy task to finalize design alterations.

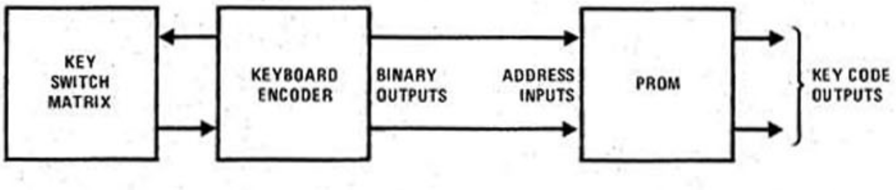
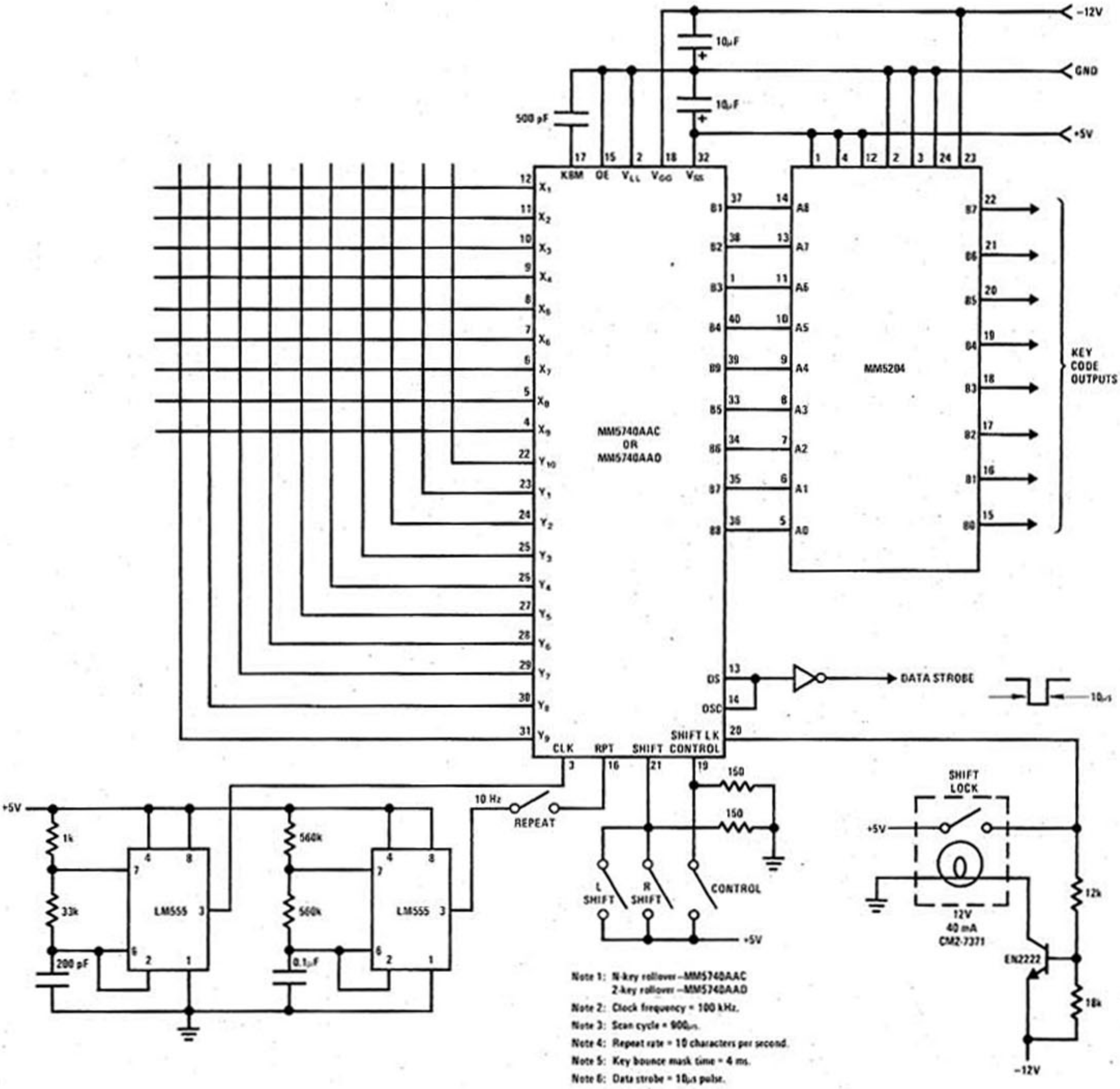


FIGURE 1. Simplified Block Diagram

*REFERENCE: AN-80 MOS Keyboard Encoding by Dom Richiuso

KEYBOARD IMPLEMENTATION

A typical implementation of this approach is shown in *Figure 2.* The encoder employs a dynamic scanning technique to identify key closures. Each keyswitch is defined by a particular X drive line and Y sense line of the encoder. In addition to the basic operation of translating a switch closure to a coded output, the MM5740



Note 1: N-key rollover—MM5740AAC
2-key rollover—MM5740AAD

Note 2: Clock frequency = 100 kHz.

Note 3: Scan cycle = 900μs.

Note 4: Repeat rate = 10 characters per second.

Note 5: Key bounce mask time = 4 ms.

Note 6: Data strobe = 10μs pulse.

FIGURE 2. Typical Keyboard System

provides all the functions necessary for modern keyboard system design. This includes all the logic necessary for key validation, 2-key or N-key rollover, bounce masking, mode selection and strobe generation. Table I illustrates the relationship between keyswitch matrix position, key mode and the binary coded outputs of the MM5740 AAC or AAD encoder. The AAC version provides for N-key rollover while the AAD is a 2-key rollover encoder. Since there are nine X lines, ten Y lines and four modes, 360 nine-bit codes are possible.

In the general application using 90 four mode keys, a 4k PROM (MM5204) should be used. If less than 64 four-mode keys are all that is required, a 2k PROM (MM5203) may be substituted. In this case, the most significant bit (B1) from the encoder is dropped and Table I addresses would go from 0–255. When programming the PROM, it should be noted that the MM5740 uses a bit paired coding system. Any particular key will have 5 common bits (B1, B2, B3, B4, B9) and 4 variable bits (B5, B6, B7, B8) which may change when going from one mode to another. In addition, encoder coding is specified in terms of negative logic so that it may be necessary to complement positive logic PROM contents when ordering encoder masks.

By careful PC board layout, the encoder/PROM prototyping system can utilize the same PC board as the final system with the PROM removed. This can be accomplished by arranging the traces so that it is possible to provide jumpers from the encoder outputs to the PROM outputs. Utilizing this approach allows for a minimum of tooling, parts counts and quick turnaround time for new designs.

TABLE I. Encoder/PROM Mapping

	KEY POSITION		MODE	ADDRESSES (ENCODER OUTPUT)									KEY CODE OUTPUTS (PROM CONTENTS)							
	X	Y		B1	B2	B3	B4	B9	B5	B6	B7	B8	B7	B6	B5	B4	B3	B2	B1	B0
KEY 1	1	1	Unshift	0	0	0	0	0	0	0	0	0	USER DEFINED KEY CODES							
	1	1	Shift	0	0	0	0	0	0	0	0	1								
	1	1	Control	0	0	0	0	0	0	0	1	0								
	1	1	Shift Control	0	0	0	0	0	0	0	1	1								
	1	2	Unshift	0	0	0	0	0	0	1	0	0								
	1	2	Shift	0	0	0	0	0	0	1	0	1								
	1	2	Control	0	0	0	0	0	0	1	1	0								
	1	2	Shift Control	0	0	0	0	0	0	1	1	1								
	.		.					.												
	.		.					.												
	.		.					.												
KEY 90	9	10	Unshift	1	0	1	1	0	0	1	0	0								
	9	10	Shift	1	0	1	1	0	0	1	0	1								
	9	10	Control	1	0	1	1	0	0	1	1	0								
	9	10	Shift Control	1	0	1	1	0	0	1	1	1								

*Encoder outputs are listed in positive true logic notation.

TABLE II. Truth Table MM5740/AAC or MM5740/AAD

MATRIX ADDRESS		COMMON					UNSHIFT				SHIFT				CONTROL				SHIFT CONTROL			
		B1	B2	B3	B4	B9	B5	B6	B7	B8	B5	B6	B7	B8	B5	B6	B7	B8	B5	B6	B7	B8
1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	0	1	1	1	0	0
1	2	1	1	1	1	1	1	0	1	1	1	0	1	0	1	0	0	1	1	0	0	0
1	3	1	1	1	1	1	0	1	1	1	0	1	1	0	0	1	0	1	0	1	0	0
1	4	1	1	1	1	1	0	0	1	1	0	0	1	0	0	0	0	1	0	0	0	0
1	5	1	1	1	1	0	1	1	1	1	1	1	1	0	1	1	0	1	1	1	0	0
1	6	1	1	1	1	0	1	0	1	1	1	0	1	0	1	0	0	1	1	0	0	0
1	7	1	1	1	1	0	0	1	1	1	0	1	1	0	0	1	0	1	0	1	0	0
1	8	1	1	1	1	0	0	0	1	1	0	0	1	0	0	0	0	1	0	0	0	0
1	9	1	1	1	0	1	1	1	1	1	1	1	1	0	1	1	0	1	1	1	0	0
1	10	1	1	1	0	1	1	0	1	1	1	0	1	0	1	0	0	1	1	0	0	0
2	1	1	1	1	0	1	0	1	1	1	0	1	1	0	0	1	0	1	0	1	0	0
2	2	1	1	1	0	1	0	0	1	1	0	0	1	0	0	0	0	1	0	0	0	0
2	3	1	1	1	0	0	1	1	1	1	1	1	1	0	1	1	0	1	1	1	0	0
2	4	1	1	1	0	0	1	0	1	1	1	0	1	0	1	0	0	1	1	0	0	0
2	5	1	1	1	0	0	0	1	1	1	0	1	1	0	0	1	0	1	0	1	0	0
2	6	1	1	1	0	0	0	0	1	1	0	0	1	0	0	0	0	1	0	0	0	0
2	7	1	1	0	1	1	1	1	1	1	1	1	1	0	1	1	0	1	1	1	0	0
2	8	1	1	0	1	1	1	0	1	1	1	0	1	0	1	0	0	1	1	0	0	0
2	9	1	1	0	1	1	0	1	1	1	0	1	1	0	0	1	0	1	0	1	0	0
2	10	1	1	0	1	1	0	0	1	1	0	0	1	0	0	0	0	1	0	0	0	0
3	1	1	1	0	1	0	1	1	1	1	1	1	1	0	1	1	0	1	1	1	0	0
3	2	1	1	0	1	0	1	0	1	1	1	0	1	0	1	0	0	1	1	0	0	0
3	3	1	1	0	1	0	0	1	1	1	0	1	1	0	0	1	0	1	0	1	0	0
3	4	1	1	0	1	0	0	0	1	1	0	0	1	0	0	0	0	1	0	0	0	0
3	5	1	1	0	0	1	1	1	1	1	1	1	1	0	1	1	0	1	1	1	0	0
3	6	1	1	0	0	1	1	0	1	1	1	0	1	0	1	0	0	1	1	0	0	0
3	7	1	1	0	0	1	0	1	1	1	0	1	1	0	0	1	0	1	0	1	0	0
3	8	1	1	0	0	1	0	0	1	1	0	0	1	0	0	0	0	1	0	0	0	0
3	9	1	1	0	0	0	1	1	1	1	1	1	1	0	1	1	0	1	1	1	0	0
3	10	1	1	0	0	0	1	0	1	1	1	0	1	0	1	0	0	1	1	0	0	0
4	1	1	1	0	0	0	0	1	1	1	0	1	1	0	0	1	0	1	0	1	0	0
4	2	1	1	0	0	0	0	0	1	1	0	0	1	0	0	0	0	1	0	0	0	0
4	3	1	0	1	1	1	1	1	1	1	1	1	1	0	1	1	0	1	1	1	0	0
4	4	1	0	1	1	1	1	0	1	1	1	0	1	0	1	0	0	1	1	0	0	0
4	5	1	0	1	1	1	0	1	1	1	0	1	1	0	0	1	0	1	0	1	0	0
4	6	1	0	1	1	1	0	0	1	1	0	0	1	0	0	0	0	1	0	0	0	0
4	7	1	0	1	1	0	1	1	1	1	1	1	1	0	1	1	0	1	1	1	0	0
4	8	1	0	1	1	0	1	0	1	1	1	0	1	0	1	0	0	1	1	0	0	0
4	9	1	0	1	1	0	0	1	1	1	0	1	1	0	0	1	0	1	0	1	0	0
4	10	1	0	1	1	0	0	0	1	1	0	0	1	0	0	0	0	1	0	0	0	0
5	1	1	0	1	0	1	1	1	1	1	1	1	1	0	1	1	0	1	1	1	0	0
5	2	1	0	1	0	1	1	0	1	1	1	0	1	0	1	0	0	1	1	0	0	0
5	3	1	0	1	0	1	0	1	1	1	0	1	1	0	0	1	0	1	0	1	0	0
5	4	1	0	1	0	1	0	0	1	1	0	0	1	0	0	0	0	1	0	0	0	0
5	5	1	0	1	0	0	1	1	1	1	1	1	1	0	1	1	0	1	1	1	0	0
5	6	1	0	1	0	0	1	0	1	1	1	0	1	0	1	0	0	1	1	0	0	0
5	7	1	0	1	0	0	0	1	1	1	0	1	1	0	0	1	0	1	0	1	0	0
5	8	1	0	1	0	0	0	0	1	1	0	0	1	0	0	0	0	1	0	0	0	0
5	9	1	0	0	1	1	1	1	1	1	1	1	1	0	1	1	0	1	1	1	0	0
5	10	1	0	0	1	1	1	0	1	1	1	0	1	0	1	0	0	1	1	0	0	0
6	1	1	0	0	1	1	0	1	1	1	0	1	1	0	0	1	0	1	0	1	0	0
6	2	1	0	0	1	1	0	0	1	1	0	0	1	0	0	0	0	1	0	0	0	0
6	3	1	0	0	1	0	1	1	1	1	1	1	1	0	1	1	0	1	1	1	0	0
6	4	1	0	0	1	0	1	0	1	1	1	0	1	0	1	0	0	1	1	0	0	0
6	5	1	0	0	1	0	0	1	1	1	0	1	1	0	0	1	0	1	0	1	0	0
6	6	1	0	0	1	0	0	0	1	1	0	0	1	0	0	0	0	1	0	0	0	0
6	7	1	0	0	0	1	1	1	1	1	1	1	1	0	1	1	0	1	1	1	0	0
6	8	1	0	0	0	1	1	0	1	1	1	0	1	0	1	0	0	1	1	0	0	0
6	9	1	0	0	0	1	0	1	1	1	0	1	1	0	0	1	0	1	0	1	0	0
6	10	1	0	0	0	1	0	0	1	1	0	0	1	0	0	0	0	1	0	0	0	0
7	1	1	0	0	0	0	1	1	1	1	1	1	1	0	1	1	0	1	1	1	0	0
7	2	1	0	0	0	0	1	0	1	1	1	0	1	0	1	0	0	1	1	0	0	0
7	3	1	0	0	0	0	0	1	1	1	0	1	1	0	0	1	0	1	0	1	0	0
7	4	1	0	0	0	0	0	0	1	1	0	0	1	0	0	0	0	1	0	0	0	0
7	5	0	1	1	1	1	1	1	1	1	1	1	1	0	1	1	0	1	1	1	0	0
7	6	0	1	1	1	1	1	0	1	1	1	0	1	0	1	0	0	1	1	0	0	0
7	7	0	1	1	1	1	0	1	1	1	0	1	1	0	0	1	0	1	0	1	0	0
7	8	0	1	1	1	1	0	0	1	1	0	0	1	0	0	0	0	1	0	0	0	0
7	9	0	1	1	1	0	1	1	1	1	1	1	1	0	1	1	0	1	1	1	0	0
7	10	0	1	1	1	0	1	0	1	1	1	0	1	0	1	0	0	1	1	0	0	0
8	1	0	1	1	1	0	0	1	1	1	0	1	1	0	0	1	0	1	0	1	0	0
8	2	0	1	1	1	0	0	0	1	1	0	0	1	0	0	0	0	1	0	0	0	0
8	3	0	1	1	0	1	1	1	1	1	1	1	1	0	1	1	0	1	1	1	0	0
8	4	0	1	1	0	1	1	0	1	1	1	0	1	0	1	0	0	1	1	0	0	0
8	5	0	1	1	0	1	0	1	1	1	0	1	1	0	0	1	0	1	0	1	0	0
8	6	0	1	1	0	1	0	0	1	1	0	0	1	0	0	0	0	1	0	0	0	0
8	7	0	1	1	0	0	1	1	1	1	1	1	1	0	1	1	0	1	1	1	0	0
8	8	0	1	1	0	0	1	0	1	1	1	0	1	0	1	0	0	1	1	0	0	0
8	9	0	1	1	0	0	0	1	1	1	0	1	1	0	0	1	0	1	0	1	0	0
8	10	0	1	1	0	0	0	0	1	1	0	0	1	0	0	0	0	1	0	0	0	0
9	1	0	1	0	1	1	1	1	1	1	1	1	1	0	1	1	0	1	1	1	0	0
9	2	0	1	0	1	1	1	0	1	1	1	0	1	0	1	0	0	1	1	0	0	0
9	3	0	1	0	1	1	0	1	1	1	0	1	1	0	0	1	0	1	0	1	0	0
9	4	0	1	0	1	1	0	0	1	1	0	0	1	0	0	0	0	1	0	0	0	0
9	5	0	1	0	1	0	1	1	1	1	1	1	1	0	1	1	0	1	1	1	0	0
9	6	0	1	0	1	0	1	0	1	1	1	0	1	0	1	0	0	1	1	0	0	0
9	7	0	1	0	1	0	0	1	1	1	0	1	1	0	0	1	0	1	0	1	0	0
9	8	0	1	0	1	0	0	0	1	1	0	0	1	0	0	0	0	1	0	0	0	0
9	9	0	1	0	0	1	1	1	1	1	1	1	1	0	1	1	0	1	1	1	0	0
9	10	0	1	0	0	1	1	0	1	1	1	0	1	0	1	0	0	1	1	0	0	0

*NEGATIVE LOGIC NOTATION "1" = –, "0" = +