APPLEBOX

Keyswitches and Keyboards at the Apple - Part 2

Basics about keyboard decoders and how they work

After we treated the topics of the keyswitches themselves very close in the first part - the topic of the matrix was only spotted with very few lines and the decoders had been left for this page.

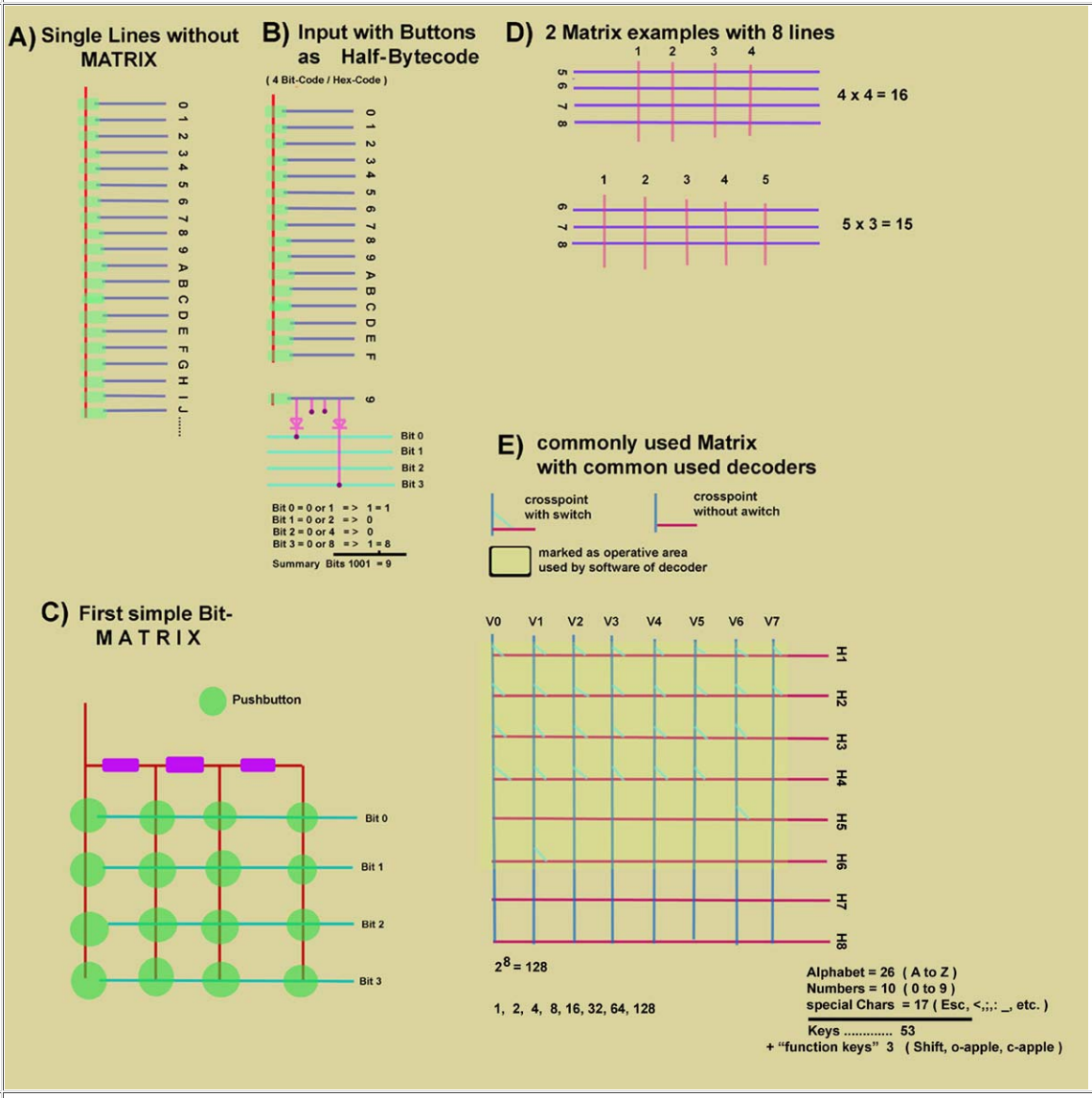
So before we treat the decoders we should switchback to the matrix again and view the function of this "trick". Basically we have at the Apple II and II+ a keyboard with 57 keys and at the Apple IIe -model there have only 5 keys added :

the left shiftkey - for uppercase
the right shiftkey - also for uppercase So this one does the same as the switch above (except in very rare special cases like in some games)
the Capslock-key which is in fact a "real switch" that remains in one position open or close until it´s pressed again....
the open-apple-key .... - in modern keyboards it´s comparable to the "Alt"-key
the closed-apple-key..... - it´s comparable to the modern "AltGr"-key .

In technical view only the capslock-key is a real switch, while all other keyswitches act in fact like "push-buttons" that only give contact of pressed and immediatly open contact again in the moment they are released.

So if you would like to determine which key is pressed without a matrix this would reqire 1 common line along to every switch with for example +5Volt and 57 lines for the apple II or 62 lines at the apple IIe ! this won´t be very smart and it would waste a lot of wiring and if you like to decide to make a PCB it would become very complicated to handle the task by avoiding that one line hits another line.....

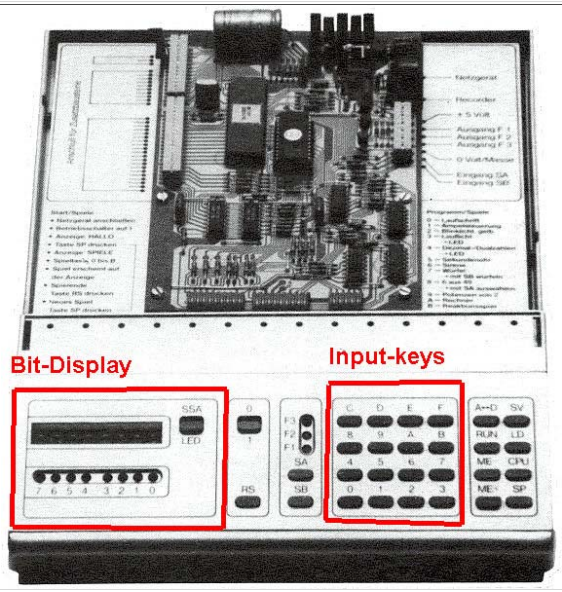
So the big trick to solve the problem is the matrix !
It´s something like a chessboard where you can determine each field by a "row" and a "column" number.
lets view below some scetches of matrix-models:



Section A) of the picture above explains the last part of the text above showing the one powerline and each keybutton with a single own line without using a matrix. In very early somputersystens and testing- or microprocessorlearningsystems instead of using keybuttons the input was also sometimes done by entering directly the bytes (8 Bit per Byte) with small Dualinlineswitches and a Button representing something like an "Enter"-button. So each of the single bits was represented by one switch and all 8 switches together were set for entering one byte. The enter-command was given by a button that gave an impulse to the board-logic and the byte was pushed into a flipflop register where the CPU pickup the input and moved it to memory. To be very exact the Byte was entered with two strikes because each Hex-number only represented half of the Byte and the lower hexnumber represented the bits 0 to 3 and the second Hexnumber represented the Bits from 4 to 7 ! Thats also the reason that programs in magazines noted in pure machine code are always written each command with 2 hexnumbers ((i.e. OE 10 07 A0 A6 .... and so on.....)

The next step was to enter the bytes with two strikes of a Button and that was immediatly translated to a Byte as input. The decoding was executed with the help of a diodematrix which caused the byte to be set by each buttonstrike. In both of this cases we talk about so called "hardware-decoded-keyboard", because the functions are wired with diodes and could not be changed (schemata B) and C)).

Such systems were very timeeating because all inputs and outputs were handled directly with machine-code in hex-code (based to the basic of the number 16 and these were represented with characters from 0 to 9 and A to F). Such systems didn´t even have a Editor or a symbolic assemblerprogram ! the next picture after this textbloxk shows such a system with 6502 CPU and the Display was given also in Hex-code by hexadecimal LED-Display.



Now lets take a look at Section D) 2 pictures before. It displays 2 examples of a matrix..... the first splits the wires in a matrix 4 x 4 and you might be able to mount 16 pushbuttons..... the other matrix is 5 x 3 and it provides us only with the possibility to mount 15 pushbuttone and not displayed in the picture but explained by simple arithmetics would be a matrix of 2 x 6 and the matrix would only offer 12 points for mounting pushbuttons..... so with 8 wires representing 8 Bits a matrix of 4 x 4 offers best resolution..... probably you might waist some time thinking about why this phenomenon happens ???
A hint: it´s got to do something with a greek man called Pytagoras and a thing he discovered.....

But now back to the picture before and the section E) ..... matrix-models used with decoderchips in the 70´s to 80´s use 8 x 8 lines resulting in 64 possible crossingpoints where a keybutton can be mounted..... and with a special trick there can be up to 255 key-codes generated....
"Stop !" you might say loud ... "how can that be that 64 buttons can resolve up to 255 items (or how can you make 255 siuations with only 64 buttons)? "

well lets examine the keys at the computer itself to discover the trick..... the very first time the trick was performed was when advancing from the Apple II to the II+ or II europlus. The key we talk about was the shift-key ! All keys had before at the older models only one code issued and one sign displayed and that old models only had uppercase characters ! By adding the ahift-key the computer became the ability to do the same as the kmost typewriting-machines by displaying upper case as well as lowercase.....
so that one key doubled the amount of displayed and representable keys and characters.... it was done by just adding the amount of 64 bit to the true value of every key ! This was done by the encoderchip the detected each keypress in the matrix and treated some keys with very special "permissions"
to be pressed "together with another key"! Teh monet we switched in the discusion from the basic 64 keys possible to be decoded with diodes and wires to instead be decoded by a matrix and a decoder chip is the very moment were we switched from "hardware-coded-keyboards" to "software-coded-keyboards". This happened in the years about 1979 to 1980 when a complete familly of microproscessors were brought up to the market known as the 65XX-family and the 8051-family. These chips had some special features in those days (now that nothing realy upraising at all ...) but in those days it was amazing that these new chips did not onbly contain a CPU but also some small amount of ROM and RAm inside and a lot of lines outside as special "communication"-lines.... this permitted the chip to be progrqammed with a small program to act / react to incidents happening at the communication-lines and the lines were used to act as a matrix and the program permitted special treatment / reaction to the hit of special selected connections....
- and violá ! the encoder-chip was born....

the rest is rather simple ... after the schift.key was "invented" soon at apple the "open-apple" and "closed apple" key were invented at the apple computers while in the Commodore-world and at IBM the Ctrl-key and Alt-key instead took place. This enabled the programmers to make a giant leap in programming special functions to ebable software to do a lot of things with double keystroke instead of long lasting commandchains...
you could call it a kind of "instant-driving a program with special-key-commands" .... at the Apple the program Appleworks was one of the very first programs with such enhancements .... that was the birth of the so called "command-key-charts" often printed on a carton and covered with plastic put on top of the keyboard untill the user knew the commands by mind without the charts....

the "losers" in those days were the guys that just copied the disk and forgot to also xerox the commandcharts because they never used the programs full features and power.... and the biggest losers were those guys that were to lazy to inspect the help-files on the disk too.... because in the most help-files there was also some documentation on the "special-key-commands" .... so a lot of games never had been played with joy but rather more stoking around the keyboard to find out how to just even get the game running and how to move around at all.....

if you ever get in such a situation the firstaid is allways to hunt for documentation and for the help-files ... it will save a lot of painfull struggles at the keyboard....

but now back to the topic itself .... another common thing in those days was the fact that a lot of computers sold in foreign countries had been equipped with a Char-ROM that contained a "national" (say better foreign "local") Character-set. This was a nice thing ... but there was one big disadvantage :
A lot of companies were to lazy to translate their programs developed for the U.S. Market to another language - so a very large amount of programs requested to run with U.S. Character-set..... so one of the most common modifications at the Apple computers was replacing the Character ROM with a larger Chip and a wire leading to a switch to permit switching between the two Character-sets : the local one and the U.S. version..... nearly the half of all Applecomputers (including the original ones and the clones) had such a switch somewhere - you would only have within the USA a chance to find computers without such a switch.... It´s a often performed mistake to think this to be a task related to the decoder chip .... It´s NOT ! This is only related to the Character ROM !

But back to the encoderchips: with upgrowing needs (in other words: with growing amount of sold computers) the chip-companies like AMI, Intel National semiconductor and others started to make own special decoderchips and "public" decoderchips.....

So whats the difference ? The public decoder-chips could be bought by any customer at any electronic-shop or it could be at least ordered from the catalog and these chips had a documentation within the catalogs of the chip-suppliers and a related "datasheet" that uncovered the decoding itself by long lists of bytes being emitted when striking a defined key ... the so called "truth-tables".... this kind of decoders where used also by companies that built different keyboards for different computers and different computer models.....

The opposite to this were the so calles "custom-chips" - this special decoders were made on the demand of companies making a large amount of same keyboards.... the went up to the chip-company and asked them to make a series of some say 10.000 chips with a decoder program only known by the company itself and the code and the documantation was given back to the keyboard-making-company and these chips were never sold in public or documented in public.... this is true for quite a lot of keyboards from cherry and a lot of keyboards made in Taiwan in the late ´78 to ´85 made for clones....
most of this kind of chips have stange marks like TK-10 or something like that and usually they don´t even have a real company Logo.

But you should not condemn them all together.... some of these keyboards were sold for a lot of bucks and the programmers in those days loved some of them... just for example the so callesd "Apple keyboard commander" : these keyboards had a special feature - a "function"-key that permitted "rapid" programing ... instead of typing the entire command "for " you just stroke once while pressing the "function"-key the f-key and the entire sequence of letters apeared at the prompt and the cursor was positioned right one empty space after "for "... just waiting for you to only enter the linenumber and then to continue entering program-code... each letter when used with the function-key was representing an entire command.... and that saved a lot of typing at the times of the Apple II+ ! Several keyboards from clones had that feature too..... in most cases you could identify these kind of keyboards by the fact that the caps on the keys also had on the front side written very small the commands related to the very key.....

The next point with this topic of decoder chips is the following: at hardwired keyboards the so called "debouncing" of the keys was eralized by wired logic omn the board that was related to the so called flip-flop-chips and the "bus-transcievers"/"latches" ... every mechanical key produces while neing pressed a series of so called "micro"-contact-hits and that generates a number of keystrokes issued within parts of a second.....

But of course the user only normaly wants to get one "a"-letter when striking the key and not a series of 6 to 8 "a"´s ! So every keyboard circuitary contains some kind of timing circuit to eliminate the additional amaount of letters. When the decoder-chips came into the market this kind of timing was integrated into the decoder chips -usualy realized with a so called R-C combination with a capacitor and one resistor that together set the time of delay before a new keystroke will be accepted by the chip..... the resisitor loads up the capasitor and when the capacitor was loaded to a defined limit the circuit opens the gate for new entered strikes.... this is for example a common problem with old keyboards.... if the ceramic-capacitor at the decoderchip is damaged this causes a series of letters issued to the computer if one single stroke was issued..... at the IIe its the one with the marking C70 and the resistor R32 close to the notched side of the decoderchip KR 3600-pro at the right front (towards the keyboard) of the mainboard.

And finally we can now finish the roundup by talking about the communication between the decoder chip and the computer itself.....
ther must be some communication between both .... the chip must give a signal to the computer that a kaystroke happened and the computer must give some kind of notice to the chip after it picked up the byte from the chip and tell the chip that the task has been performed.....
so in normal case there is a signal line like "Strobe" that indicates the keystroke and the line to instruct the chip that the computer picked up the information like control. The correct used terms change from company to company so using the correct term usually afforde to read the datasheet of the chip.

Now following will be the datasheets of the most common decoderchips and at the beginning sometimes some short remarks by me.Thereafter is every time a link to permit downloading the datasheet.

The MM5740 was used in the datanetic keyboard and is now also searched for the Mimeo or other replications. The chip was also used in some very early Apple II´s.

MM5740

Keyboard Encoder Circuits

For additional application information, see AN-128 and AN-139 at the end of this section.

MM5740 90-key keyboard encoder

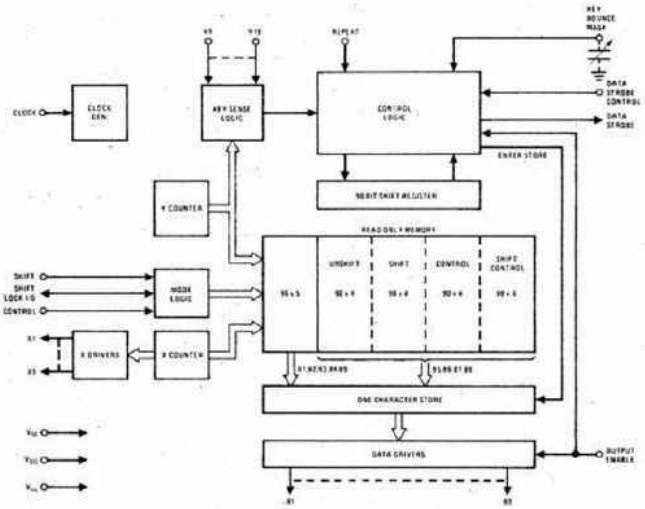
general description

The MM5740 MOS/LSI keyboard encoder is a complete keyboard interface system capable of encoding 90 single pole single throw switch closures into a usable 9-bit code. It is organized as a bit paired system and is capable of N key or two key rollover. The MM5740 is fabricated with silicon gate technology and provides for direct TTL/DTL compatibility on Data and Strobe outputs without the use of any special interface components.

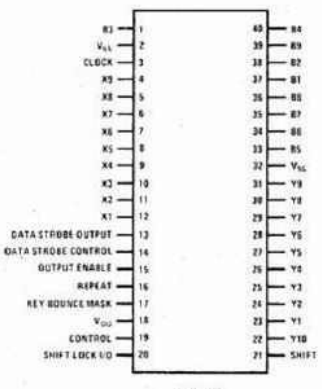
features

- TRI-STATE® data outputs directly compatible with TTL/DTL or MOS logic
- Function inputs directly compatible with TTL/DTL logic
- Only one TTL level clock required
- N key/two key rollover (mask programmable)
- 90 key-quad mode capability
- One character data storage
- Repeat function (selectable)
- Shift lock with indicator capability
- Key bounce masking by single external capacitor
- Level or pulse data strobe output
- Data strobe pulse width control

block and connection diagrams



Dual-In-Line Package



Order Number MM5740N
See Package 24

absolute maximum ratings

Data and Clock Input Voltages and Supply Voltages with Respect to V_{SS}	+0.3V to −20V
Power Dissipation	600 mW at T_A = +25°C
Operating Temperature	−25°C to +70°C ambient
Storage Temperature	−65°C to +160°C
Lead Temperature (Soldering, 10 seconds)	300°C

electrical characteristics (Note 1,5)

PARAMETER	CONDITIONS	MIN	TYP	MAX	UNITS
Clock Repetition Rate		10		200	kHz
Clock Pulse Width	Rep. Rate = 200 kHz	2.4		2.6	μs
	Rep. Rate = 10 kHz	20		80	μs
Clock Amplitude					
Logic Level "0"				3.25	V
Logic Level "1"		+0.4			V
Clock Transition Times					
Risetime	Rep. Rate = 200 kHz			100	ns
Falltime	Rep. Rate = 200 kHz			100	ns
Clock Input Capacitance			5.0		pF
Data Input Levels, Y1 thru Y10					
Logic Level "0"				V_{SS} − 1.5	V
Logic Level "1"		−4.5			V
Logic Level "0"				3.25	V
Logic Level "1"		+0.4			V
Data Strobe Control					
Logic Level "0"				+3.5	V
Logic Level "1"		+0.4			V
Data Output Levels, X1 thru X9					
Logic Level "0"	When Connected to Y1 thru Y10			V_{SS} − 0.75	V
Logic Level "1"	via Switch Matrix, (C_L = 75 pF)	−4.5			V
B1 thru B9 and Data Strobe					
Logic Level "0"	I = 100μA (Note 2)			V_{SS} − 1.0	V
Logic Level "1"	I = 1.6 mA (Note 2)	+0.4			V
Shift Lock Voltage Open	Before Closure		V_{GG} − 2.0		V
Shift Lock Voltage Closed	Switch Closed		V_{SS}		V
Shift Lock Voltage Locked	After Release, (I = 1.0 mA) (Figure 2)		V_{SS} − 5.0	V_{SS} − 8.0	V
Transition Times					
Data Strobe (T_{DS1})	C_L = 100 pF, I = 1.6 mA			2.5	μs
Data Strobe (T_{DS0})	C_L = 100 pF, I = 100μA			1.0	μs
Data Output Levels					
(T_{DO1})	C_L = 100 pF, I = 1.6 mA			2.5	μs
(T_{DO0})	C_L = 100 pF, I = 100μA			1.0	μs
Output Enable Setup Time (T_{OES})		2.5			μs
Output Enable Release Time (T_{OER})		2.5			μs
Repeat Input Pulse Width (T_{RPW})	(Note 3)				
	f_{CLOCK} = 10 kHz	10			ms
	f_{CLOCK} = 200 kHz	0.5			ms
Power Supply Current	I_{GG}, I_{SS}		20	35	mA

Note 1: These specifications apply for V_{SS} = +5.0 VDC ±5%, V_{GG} = −12.0 VDC ±5%, V_{LL} = GND and T_A = 0°C to +70°C.

Note 2: When outputs B1 thru B9 and Data Strobe are driving TTL/DTL $V_{SS} - V_{LL} < 5.25V$. When driving MOS, $V_{SS} - V_{LL} \leq 10.0V$.

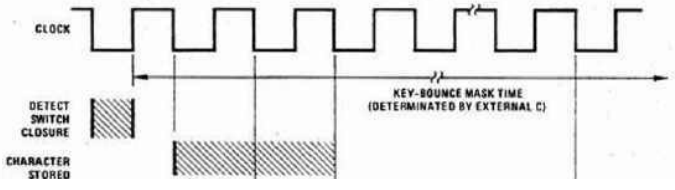
Note 3: $T_{RPW\ min.} = 100 \times \frac{1}{f_{clock}}$

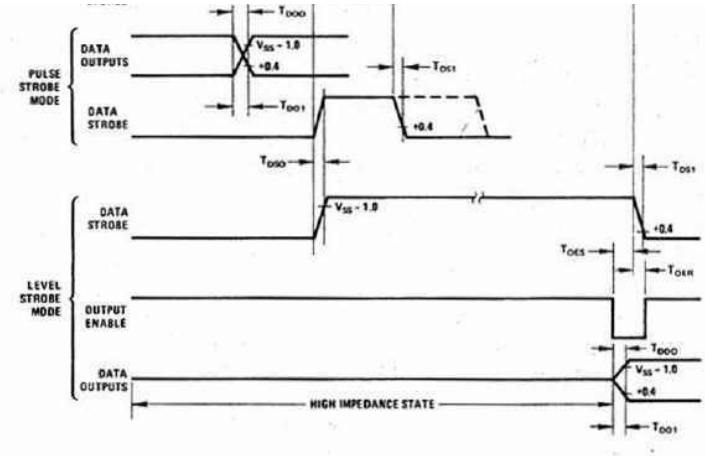
Note 4: If shift and control inputs are derived from a single pole, single throw switch closure to V_{SS}, a 100 OHM resistor returned to V_{LL} (GND) is required on these inputs.

Note 5: The following inputs have internal pull-up resistors to V_{SS}: clock, output enable, repeat, shift, control.

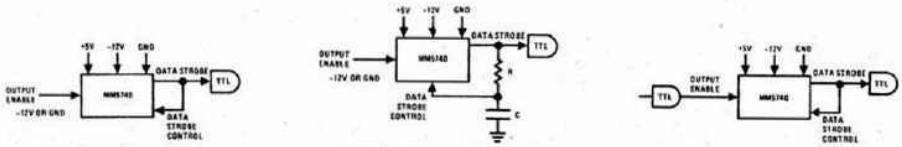
10

timing diagram





applications information



Pulse Data Strobe Mode

Level Data Strobe Mode

key bounce capacitor values

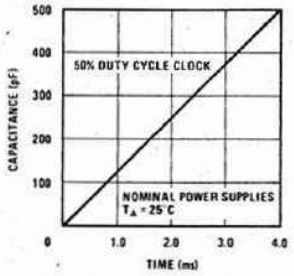


FIGURE 1. Key-Bounce Mask Time

10

MM5740

application

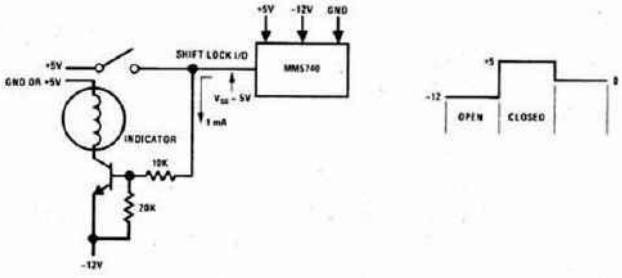
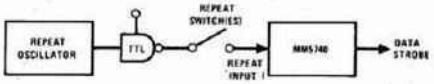
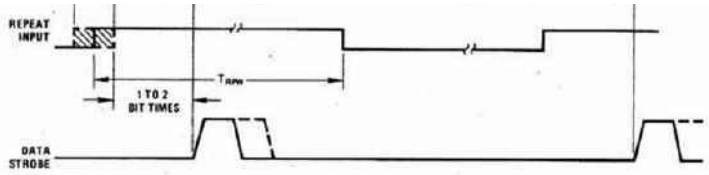


FIGURE 2. Shift Logic I/O Interface

repeat switch function



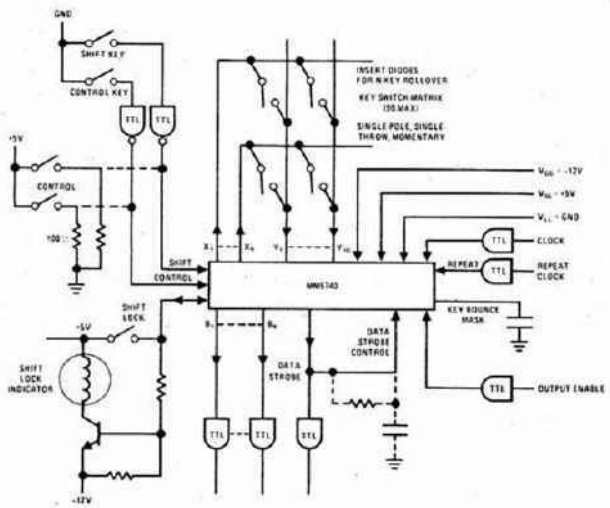
Repeat Switch Connections



Note: Both Repeat Switch and a Data Key must be depressed to enable repeat function. For N-Key Rollover, the data outputs will represent the current valid data key (N Key Roll during Repeat).

Repeat Function

typical applications



MM5740

CODE ASSIGNMENT CHART

Customer: _____

Date: _____

MATRIX ADDRESS		COMMON					UNSHIFT				SHIFT				CONTROL				SHIFT CONTROL				CHARACTER			
X	Y	B_1	B_2	B_3	B_4	B_9	B_5	B_6	B_7	B_8	B_5	B_6	B_7	B_8	B_5	B_6	B_7	B_8	B_5	B_6	B_7	B_8	US	S	C	SC
(Note 3)	1																									
	2																									
	3																									
	4																									
	5																									
	6																									
	7																									
	8																									
	9																									
	10																									
	1																									
	2																									
	3																									
	4																									
	5																									
	6																									
	7																									
	8																									
	9																									
	10																									
	1																									
	2																									
	3																									
	4																									
	5																									
	6																									
	7																									
	8																									
	9																									
	10																									

☐ N-Key Rollover
☐ 2 Key Rollover
Page ☐ of 3 (Note 1)
Note: Use B8 if parity bit is desired

Note 1: 3 code assignment charts are required for each keyboard encoder pattern. Fill in a "1" or "0" in each output box (B_1 thru B_9). Indicate page number.
Note 2: The matrix is 9 "X" locations by 10 "Y" locations.
Note 3: Write in 10 one's, 10 two's, etc. in successive X address locations up to 9. This will fill 3 charts. The first page will have address matrix location 1,1; 1,2: 1,3... 1,10; 2,1; 2,2... 2,10; 3,1, etc. up to 3,10. Page 2 has 4,1 to 6,10. Page 3 has 7,1 to 9,10.
Note 4: A contact closure at the address matrix location will cause the appropriate bit pattern to appear at the output in negative true logic. V_{OH} = "0"; V_{OL} = "1."
Note 5: See application note AN-80 for coding example.

10

MM5740

MM5740AAE, MM5740AAF CODE ASSIGNMENT CHARTS

MATRIX ADDRESS		COMMON					UNSHIFT				SHIFT				CONTROL				SHIFT CONTROL				CHARACTER			
X	Y	B_1	B_2	B_3	B_4	B_9	B_5	B_6	B_7	B_8	B_5	B_6	B_7	B_8	B_5	B_6	B_7	B_8	B_5	B_6	B_7	B_8	US	S	C	SC
1	1	0	0	0	1	0	1	1	0	1	1	1	0	1	1	1	0	1	1	1	0	1	8	8	8	8
1	2	0	0	1	0	0	1	1	0	1	1	1	0	1	1	1	0	1	1	1	0	1	4	4	4	4
1	3	1	0	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	5	5	5	5
1	4	1	0	0	0	0	1	1	0	1	1	1	0	1	1	1	0	1	1	1	0	1	1	1	1	1
1	5	0	1	0	0	0	1	1	0	1	1	1	0	1	1	1	0	1	1	1	0	1	2	2	2	2
1	6	1	1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	3	3	3	3
1	7	0	0	0	0	1	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	@	@	@	@
1	8	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	6	6	6	6
1	9	1	0	0	1	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	0	0	9	9
1	10	1	1	1	0	0	1	1	0	1	1	1	0	1	1	1	0	1	1	1	0	1	7	7	7	7
2	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	FF	FF	FF	FF
2	2	1	0	1	1	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	CR	CR	CR	CR
2	3	0	0	1	1	0	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	FS	FS	FS	FS
2	4	1	0	1	1	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	GS	GS	GS	GS
2	5	1	1	0	1	0	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	VT	VT	VT	VT
2	6	0	1	1	1	0	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	SO	SO	SO	SO
2	7	0	0	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	SP	SP	SP	SP
2	8	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	HT	HT	HT	HT
2	9	0	0	0	1	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	BS	BS	BS	BS
2	10	1	0	1	1	1	0	1	0	0	1	1	0	1	0	1	0	0	1	1	0	1				
3	1	0	0	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	`	`	`	`
3	2	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	LF	LF	LF	LF
3	3	0	0	0	0	0	1	0	1	0	0	0	1	1	1	0	0	1	0	0	0	0	P	P	DLE	NUL
3	4	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	DEL	DEL	DEL	DEL
3	5	1	1	0	1	0	1	1	0	1	0	1	0	0	1	1	0	1	0	1	0	0	+	+	-	-
3	6	0	1	1	1	1	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	-	-	-	-
3	7	1	1	1	1	0	0	1	0	1	1	1	0	0	0	1	0	1	1	1	0	0	?	?		?
3	8	0	0	0	0	0	1	0	1	0	1	0	1	0	1	0	0	1	1	0	0	1	P	P	DLE	DLE
3	9	1	1	1	1	0	0	0	1	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	SI	SI
3	10	0	1	0	1	1	1	1	0	0	0	1	0	1	1	1	0	0	0	1	0	1	-	"	-	"

MATRIX ADDRESS		COMMON					UNSHIFT				SHIFT				CONTROL				SHIFT CONTROL				CHARACTER			
X	Y	B_1	B_2	B_3	B_4	B_9	B_5	B_6	B_7	B_8	B_5	B_6	B_7	B_8	B_5	B_6	B_7	B_8	B_5	B_6	B_7	B_8	US	S	C	SC
4	1	1	0	0	1	0	1	1	0	0	0	1	0	1	1	1	0	0	0	1	0	1	9	)	9	)
4	2	1	0	0	1	0	0	0	1	1	0	0	1	1	0	0	0	0	0	0	0	0	)	)	HT	HT
4	3	1	1	1	1	1	0	0	1	1	1	0	1	0	0	0	0	0	1	0	0	1	0		SI	US
4	4	1	1	0	1	0	0	0	1	0	1	0	1	1	0	0	0	1	1	0	0	0	K	[	VT	ESC
4	5	0	0	1	1	0	0	0	1	1	1	0	1	0	0	0	0	0	1	0	0	1	L	\	FS	FS
4	6	0	0	1	1	0	0	1	0	1	1	1	0	0	0	1	0	1	1	1	0	0	<	<	<	<
4	7	0	1	1	1	1	0	1	0	0	1	1	0	1	0	1	0	0	1	1	0	1	-	-	-	-
4	8	0	0	1	1	0	0	0	1	1	0	0	1	1	0	0	0	0	0	0	0	0	L	L	FF	FF
4	9	1	1	0	1	0	0	0	1	0	0	0	1	0	0	0	0	1	0	0	0	1	K	K	VT	VT
4	10	0	0	0	1	0	1	1	0	1	0	1	0	0	1	1	0	1	0	1	0	0	8	(	8	(
5	1	0	1	1	0	0	1	1	0	0	0	1	0	1	1	1	0	0	0	1	0	1	G	&	G	&
5	2	1	0	1	0	0	1	0	1	0	1	0	1	0	1	0	0	1	1	0	0	1	U	U	NAK	NAK
5	3	1	0	0	1	0	1	0	1	0	1	0	1	0	1	0	0	1	1	0	0	1	Y	Y	EM	EM
5	4	0	1	0	1	0	0	0	1	1	0	0	1	1	0	0	0	0	0	0	0	0	J	J	LF	LF
5	5	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	0	1	0	0	0	1	H	H	BS	BS
5	6	1	0	1	1	0	0	0	1	0	1	0	1	1	0	0	0	1	1	0	0	0	M	J	CR	GS
5	7	0	1	1	1	1	0	0	1	0	1	0	1	1	0	0	0	1	1	0	0	0	N	^	SO	RS
5	8	1	0	1	1	0	0	0	1	0	0	0	1	0	0	0	0	1	0	0	0	1	M	M	CR	CR
5	9	0	1	1	1	0	0	0	1	0	0	0	1	0	0	0	0	1	0	0	0	1	N	N	SO	SO
5	10	1	1	1	0	0	1	1	0	1	0	1	0	0	1	1	0	1	0	1	0	0	7	'	7	'
6	1	1	0	1	0	0	1	1	0	0	0	1	0	1	1	1	0	0	0	1	0	1	5	%	5	%
6	2	0	1	0	0	0	1	0	1	1	1	0	1	1	1	0	0	0	1	0	0	0	R	R	DC2	DC2
6	3	0	0	1	0	0	1	0	1	1	1	0	1	1	1	0	0	0	1	0	0	0	T	T	DC4	DC4
6	4	0	1	1	0	0	0	0	1	1	0	0	1	1	0	0	0	0	0	0	0	0	F	F	ACK	ACK
6	5	1	1	1	0	0	0	0	1	0	0	0	1	0	0	0	0	1	0	0	0	1	G	G	BEL	BEL
6	6	0	1	1	0	0	1	0	1	0	1	0	1	0	1	0	0	0	1	0	0	0	V	V	SYN	SYN
6	7	0	1	0	0	0	0	0	1	0	0	0	1	0	0	0	0	1	0	0	0	1	B	B	STX	STX
6	8	0	0	0	1	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	CAN	CAN	CAN	CAN
6	9	1	0	0	1	0	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	EM	EM	EM	EM

6	10	0	0	1	0	0	1	1	0	1	0	1	0	0	1	1	0	1	0	1	0	0	4	S	4	S

Negative True Logic

$B_1 - B_7$ = ASCII Code
B_8 = Even parity (on $B_1, B_2, B_3, B_4, B_5, B_6, B_7, B_8$)
B_9 = Selective Repeat Bit

Note: Use B_8 if parity bit is desired.

MM5740AAE, MM5740AAF CODE ASSIGNMENT CHARTS (CONTINUED)

MATRIX ADDRESS		COMMON					UNSHIFT				SHIFT				CONTROL				SHIFT CONTROL				CHARACTER			
X	Y	B_1	B_2	B_3	B_4	B_5	B_5	B_6	B_7	B_8	B_5	B_6	B_7	B_8	B_5	B_6	B_7	B_8	B_5	B_6	B_7	B_8	US	S	C	SC
7	1	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	DC2	DC2	DC2	DC2
7	2	1	0	1	0	0	0	0	1	1	0	0	1	1	0	0	0	0	0	0	0	0	E	E	ENG	ENG
7	3	1	1	0	0	0	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	DC3	DC3	DC3	DC3
7	4	0	0	1	0	0	0	0	1	0	0	0	1	0	0	0	0	1	0	0	0	1	D	D	EOT	EOT
7	5	0	0	1	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	DC4	DC4	DC4	DC4
7	6	1	1	0	0	0	0	0	1	1	0	0	1	1	0	0	0	0	0	0	0	0	C	C	ETX	ETX
7	7	1	0	1	0	0	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	NAK	NAK	NAK	NAK
7	8	0	1	1	0	0	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	SYN	SYN	SYN	SYN
7	9	1	1	1	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	ETB	ETB	ETB	ETB
7	10	1	1	0	0	0	1	1	0	0	0	1	0	1	1	1	0	0	0	1	0	1	3	#	3	#
8	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	ENQ	ENQ	ENQ	ENQ
8	2	1	1	1	0	0	1	0	1	1	1	0	1	1	1	0	0	0	1	0	0	0	W	W	ETB	ETB
8	3	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	ACK	ACK	ACK	ACK
8	4	1	1	0	0	0	1	0	1	0	1	0	1	0	1	0	0	1	1	0	0	1	S	S	DC3	DC3
8	5	1	1	1	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	BEL	BEL	BEL	BEL
8	6	0	0	0	1	0	1	0	1	1	1	0	1	1	1	0	0	0	1	0	0	0	X	X	CAN	CAN
8	7	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	SI	SI	SI	SI
8	8	0	0	0	0	0	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	DLE	DLE	DLE	DLE
8	9	1	0	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	DC1	DC1	DC1	DC1
8	10	0	1	0	0	0	1	1	0	1	0	1	0	0	1	1	0	1	0	1	0	0	?	-	?	-
9	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	NUL	NUL	NUL	NUL
9	2	1	0	0	0	0	1	0	1	1	1	0	1	1	1	0	0	0	1	0	0	0	Q	Q	DC1	DC1
9	3	1	1	0	1	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	ESC	ESC	ESC	ESC
9	4	1	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	1	0	0	0	1	A	A	SOH	SOH
9	5	1	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	SOH	SOH	SOH	SOH
9	6	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	0	1	1	0	0	1	Z	Z	SUB	SUB
9	7	0	1	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	STX	STX	STX	STX
9	8	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	ETX	ETX	ETX	ETX
9	9	0	0	1	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	EOT	EOT	EOT	EOT
9	10	1	0	0	0	0	1	1	0	1	0	1	0	0	1	1	0	1	0	1	0	0				

Negative True Logic

$B_1 - B_7$ = ASCII Code
B_8 = Even parity (on $B_1, B_2, B_3, B_4, B_5, B_6, B_7, B_8$)
B_9 = Selective Repeat Bit

Note: Use B_8 if parity bit is desired.

ASR

ASR 33
MM5740AAE (N-KEY ROLLOVER)
MM5740AAF (2-KEY ROLLOVER)

Typical Keyboard Arrangement

10

the MM5740 Datasheet as PDF-file

The MM5745 chip wasn´t used within the apple computers, but you might find the chip used in keyboards from thirdparty manufacturers...

MM5745, MM5746 78-key keyboard encoder

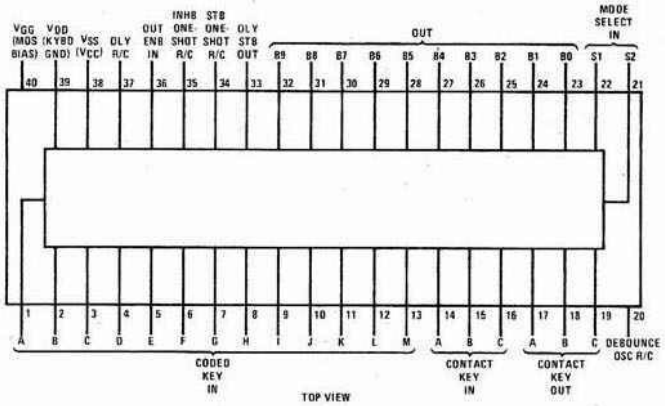
general description

The MM5745, MM5746 MOS/LSI keyboard encoder is a complete keyboard interface system capable of encoding 78 double-pole single-throw switches (hall-effect, capacitive, or contact) into a 10-bit code. Full quad-mode operation allows 4 independent 10-bit codes per switch. Debounce circuits for contact keys are provided for 3 function switches. The MM5745, MM5746 is fabricated with low threshold metal gate P-channel enhancement devices and ion-implanted resistors and provides for direct TTL/DTL compatibility on Data and Strobe outputs without the use of any special interface components.

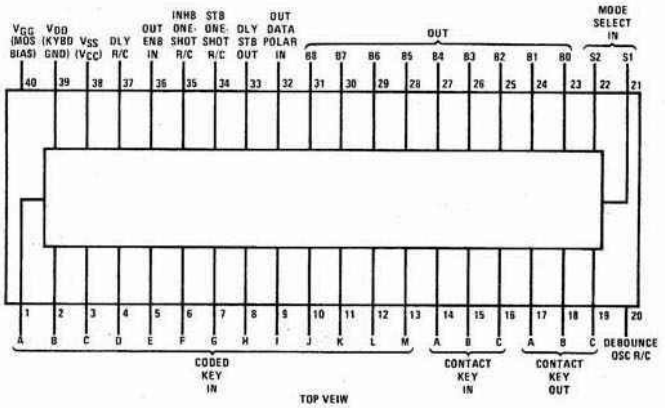
features

- 78- key quad-mode capability
- N-key/2-key rollover
- 1 character data storage
- Level or pulse data strobe output
- Data strobe pulse width control
- Key bounce delay control
- Function key debounce circuits
- Data and Strobe outputs directly compatible with TTL/DTL or MOS logic

connection diagrams (Dual-In-Line Packages)



Order Number MM5745N
See Package 24



Order Number MM5746N
See Package 24

absolute maximum ratings

Voltage at Any Pin Except Outputs	V_{SS} + 0.3V to V_{SS} − 25V
Voltage at Any Output Pin	V_{SS} + 0.3V to V_{SS} − 20V
Power Dissipation	700 mW at T_A = 25°C
Operating Temperature	−25°C to +70°C ambient
Storage Temperature	−65°C to +160°C
Lead Temperature (Soldering, 10 seconds)	300°C

electrical characteristics (Note 1)

PARAMETER		CONDITIONS	MIN	TYP	MAX	UNITS
V_{IH}	High Level Input Voltage	With Respect to V_{SS}			−1.5	V
V_{IL}	Low Level Input Voltage	With Respect to V_{DD}			0.8	V
V_{OH}	High Level Output Voltage	With Respect to V_{SS}			−1.8	V

V_{OL}	Low Level Output Voltage	With Respect to V_{DD}, I_{OL} = 1.6 mA			0.4	V
I_{IL}	Low Level Input Current (Logic)	V_{SS} = 5.25V, V_{IN} = 0.4V (Not Including MOS Inputs), (Note 2)			−1.6	mA
t_r	10–90% Output Rise Time	C_L = 50 pF			1	μs
t_f	90–10% Output Fall Time	C_L = 50 pF			1	μs
t_d	Delay Time Input to Output	Delay Capacitor = 0, R_L = 200Ω			20	μs
t_s	Delay from Strobe to Data Output		0.5			μs
D_{td}	Delay R/C Time Delay	±25% Variation Max per Given Set of R and C	40		80	μs
		R–Useful Range	200		680	kΩ
		C–Useful Range at Min R	0.001		0.002	μFd
I_{td}	Inhibit One-Shot Time Delay	±25% Variation Max per Given Set of R and C	1		30	ms
		R–Useful Range	200		680	kΩ
		C–Useful Range at Min R	0.025		0.75	μFd
S_{td}	Strobe One-Shot Time Delay	±25% Variation Max per Given Set of R and C Typ	40		80	μs
		R–Useful Range	200		680	kΩ
		C–Useful Range at Min R	0.001		0.002	μFd
B_{td}	Debounce Oscillator	±25% Variation Max per Given Set of R and C	1		7	ms
		R–Useful Range	200		680	kΩ
		C–Useful Range at Min R	0.025		0.175	μFd
I_{SS}	Supply Current	V_{SS} = 5.25V			100	mA
I_{GG}	Bias Current	V_{GG} = −18V			5	mA

Note 1: V_{SS} = 5V ±5%, V_{DD} = Gnd, V_{SS} = −12V to −18V and T_A = 0°C to +70°C.
Note 2: The following inputs have internal pull-up resistors to V_{SS}: Output Enable, Output Data Polarity.

functional description

A block diagram of the MM5745 and MM5746 keyboard encoders is shown in *Figure 1*. Connection diagrams for these devices are shown on the previous page. The following discussions are based on *Figure 1*.

Coded Key Inputs

Thirteen MOS type coded key inputs, designated A–M can be coded in an M of N format. These codes must be specified with each reprogramming of the coding mask. A maximum of 78 input codes may be specified. Typically, coding takes the form of 2 out of 13 inputs.

Contact Key Inputs

Three MOS type contact key inputs designated A, B and C can be used to debounce contact type switches.

functional description (Continued)

Mode Select Inputs

Two mode inputs, designated S1 and S2, are used to select any 1 of the 4 output coding modes. The binary number selections to represent a given output code mode must be specified with each reprogramming of the coding mask.

Output Data Polarity Input (MM5746 Only)

The Output Data Polarity Input, when switched from one state to the other, causes a reversal of the output data polarity. When open, the input is held high, logical "1", by an internal pull-up resistor, and the data comes through non-inverted from the output ROM.

Output Enable Input

The Output Enable Input enables the output storage latches to accept new output data and allows an output strobe to be generated. When the input is open, an internal pull-up resistor holds the input high, logical "1", and enables the output. When held low, logical "0", the output and strobe are disabled.

Debounce Oscillator R/C Input

The Debounce Oscillator R/C Input is a timing input that can eliminate closing or opening contact bounce durations of between 1 to 2 clock periods. Depending upon the length of bounce and R/C values chosen, the output will be delayed from the inputs from 1 to 14 ms. The resistor connects to V_{GG} and the Capacitor connects to V_{SS}.

Strobe One-Shot R/C Input

The Strobe One-Shot R/C Input is a timing input used to adjust the width of the delayed output strobe. The strobe width has a ±25% variation for a given set of R and C. The pulse width range can be varied between 1 μs and 10 ms. The resistor and capacitor timing elements are connected as stated for the Debounce Oscillator R/C input.

Inhibit One-Shot R/C Input

The Inhibit One-Shot R/C Input is a timing input used to disable the Encoder Chip outputs for a period of time after new data has appeared at the outputs and a strobe issued. The inhibit time is necessary to allow the Coded Key inputs to settle out after a keyswitch is depressed. The time slot is adjustable from 1–10 ms ±25%. The recovery time is less than 100 μs. The resistor and capacitor timing elements are connected as stated for the Debounce Oscillator R/C Input.

Delay R/C Input

The Delay R/C Input is a timing input used to determine that valid data is present at the Coded Key Inputs. Valid data must be present continuously for some period of time adjustable between 40 and 80 μs ±25% before the data is accepted as valid data. The resistor and capacitor timing elements are connected as stated for the Debounce Oscillator R/C Input.

Contact Key Outputs

Three contact key outputs designated A–C provide bounce-free non-inverted outputs corresponding to their respective inputs.

Data Outputs

Ten Data Output lines designated B0–B9 are provided. The specific output code related to a given input code and mode must be specified with each reprogramming of the coding mask.

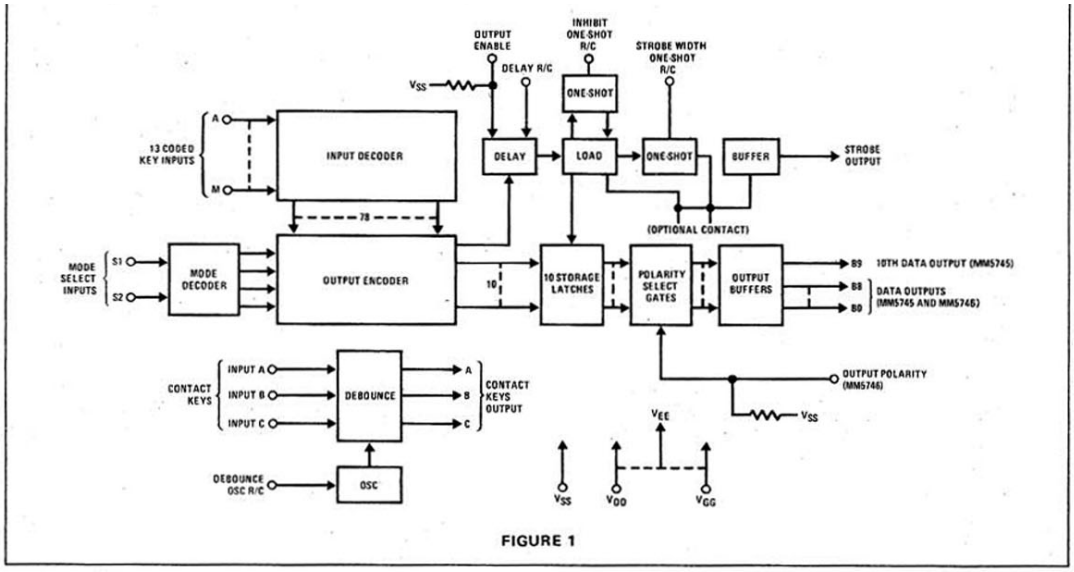


FIGURE 1

10-12

functional description (Continued)

Strobe Output

The Strobe Output is used to indicate that new data has just been placed on the Data Output lines.

Data Transfer

Input data, typically in a 2 out of 13 format, is introduced by depressing a keyswitch. The data passes through the input buffers, input inverters, and is decoded into single line codes if the data is valid. There are a maximum of 78 single line codes and these are coded into 41-bit output words. The 41st bit is used to enable the delay R/C timer. Valid input data must be present continuously for typically 60 μs before it is accepted as valid input data and the proper output codes and strobe are generated.

The status of the mode select inputs determines which of the 4 10-bit output codes are selected (first 40 bits). The mode select lines are programmable in binary format and therefore are decoded into single line codes. The output encode in reality has 82 input lines (78 input codes and 4 modes). When a valid input code is present and the mode is selected, the proper 10-bit word is steered through the Mode "OR" Gates and to the inputs of the storage latches. When the proper delay interval has elapsed, the load logic loads the new data into the storage latches.

Both polarities of the 10 data bits are fed to the Polarity Select Gates where the output Data Polarity Input selects the desired polarity output. The selected 10 data bits output the chip through the Output Buffers.

Logic Sequence

The Logic Sequence is not initiated until the successful completion of the delay timing cycle. At the completion of the delay cycle, 3 things happen almost simultaneously. First, a load signal of approximately 2 μs is fed to the storage latches to accept new data. Second, the Strobe Pulse, typically 60 μs wide, is generated. This pulse will not go true until at least 1/2 μs after the data is present at the outputs. Third, the inhibit timing cycle is initiated within 2 μs after the load and strobe inputs are generated and locks out the load and strobe inputs for the duration of the inhibit timing cycle. This insures that only one strobe is generated and no data is changed during the inhibit cycle.

If the input data disappears less than 1/2 μs after the completion of the delay cycle, it is possible that erroneous logic sequencing can take place. The symptoms are new data, but no strobe or no new data, but a strobe is generated.

If the output enable input is held false, no logic sequencing can take place and the chip remains locked up with the existing data statically available at the outputs and no strobes can be generated.

A programming option is available wherein a level strobe can be specified instead of the delayed strobe as described above. In this option, the level strobe goes true at the end of the delay cycle as does the delayed strobe, but is remains true as long as a valid data input signal is present. It is not affected by the inhibit timing cycle. The level strobe responds to the data input lines and is inhibited only by the Output Enable going false.

Debounce Circuits

The debounce circuits utilize a pulse train clock oscillator and shift registers. The input must remain in one state for 2 consecutive clock pulses before it will change the output to that state. The outputs follow the input, in that they are non-inverting.

OPTIONS

The following options are customer specified. (For format information, see Programming Format section).

Input Code

The input code M out of N (typically 2 out of 13) must be specified for each reprogramming of the coding mask.

Mode Select

The Mode Select lines bit pattern must be specified for each mode for each reprogramming of the coding mask. Each mode must be specified whether used or not.

Output Code

The Output Code must be specified for each input code and mode as above.

Strobe

The Delayed Strobe is automatically selected unless the option for the level strobe is selected.

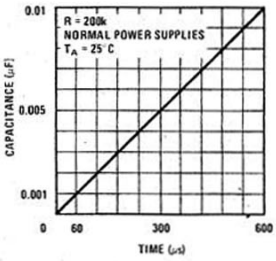
Input Resistors

Each of the 13 inputs and the 2 mode select inputs may have internal resistors (4.5 kΩ ±30%) connected to V_{SS}, V_{DD} or left open.

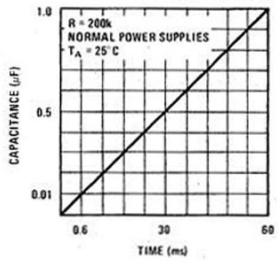
10-13

typical performance characteristics

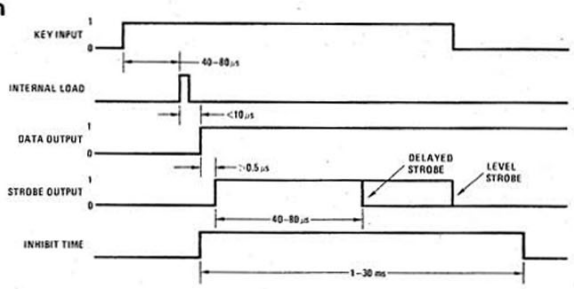
Delay Capacitor Values



Delay Capacitor Values



timing diagram



MM5745, MM5746

10

10-15

the MM5745 datasheet as PDF-file

The AY-5-3600 was NOT used in the Apple IIe. The correct chip is below of this one ! This chip is similar but NOT the same to other chips that just have other extensions.

GENERAL INSTRUMENT **AY-5-3600**

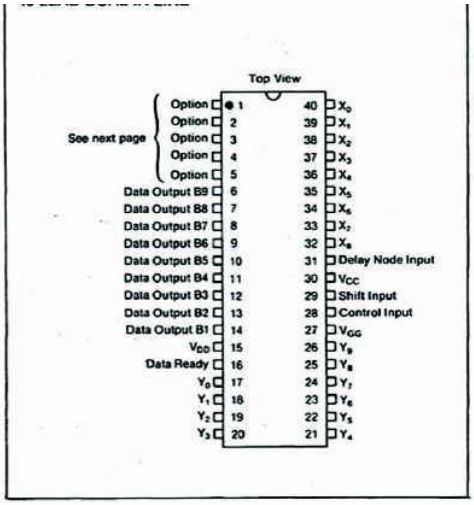
Keyboard Encoder

FEATURES

- One integrated circuit required for complete keyboard assembly

PIN CONFIGURATION
40 LEAD DUAL-IN-LINE

- N key rollover or lock out operation
- Quad mode operation
- Lock out/rollover selection under external control (option)
- Self-contained or slave oscillator circuit
- 10 output data bits available
- Outputs directly compatible with TTL/DTL or MOS logic arrays
- Output data buffer register included
- Output enable provided (option)
- External data complement control provided (option)
- Pulse or level data ready output signal provided (option)
- "Any Key Down" output provided (option)
- Externally controlled delay network provided to eliminate the effect of contact bounce
- Programmable coding with a single mask change
- Static charge protection on all input and output terminals
- Entire circuit protected by a layer of glass passivation



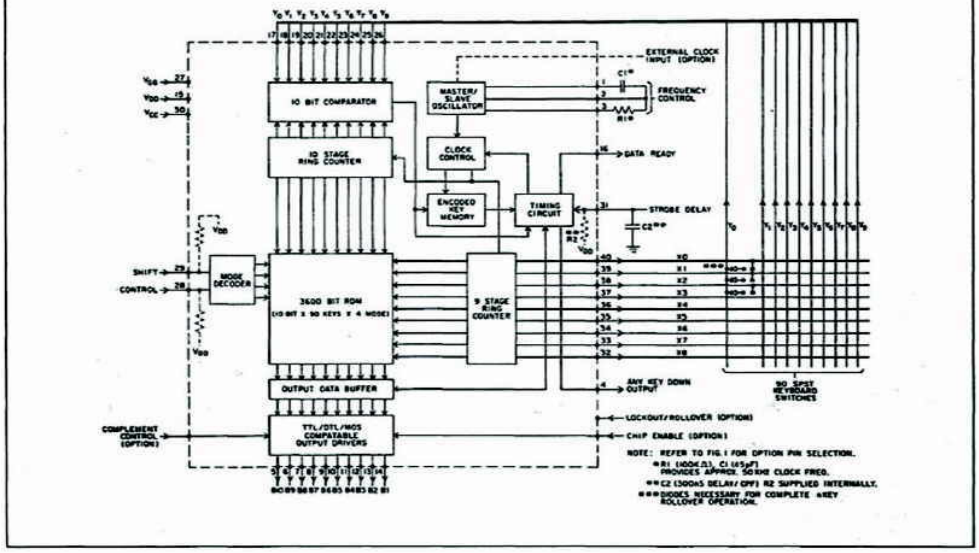
ROM

DESCRIPTION

The General Instrument AY-5-3600 is a Keyboard Encoder containing a 3600 bit Read Only Memory and all the logic necessary to encode single pole single throw keyboard closures into a usable 10 bit code. Data, Any Key Down and Data Ready outputs are directly compatible with TTL/DTL or MOS logic arrays without the need for any special interface components.

The AY-5-3600 is fabricated with MTNS technology and contains 5000 P channel enhancement mode transistors on a single monolithic chip.

BLOCK DIAGRAM



AY-5-3600

CUSTOM CODING INFORMATION

The custom coding information for General Instrument's AY-5-3600 Keyboard Encoder ROM should be transmitted to General Instrument in the form of 80 column punched cards. Each ROM pattern requires 92 cards (1 title card, 1 circuit option card and 90 ROM pattern cards). (See Note 1)

If it is not possible to supply punched cards, then the Truth Table should be completed (See Note 1). However, there would be a substantial savings in both the coding charge and turn-around time if punched cards were used. Upon receipt of the punched cards or the Truth Table, General Instrument will prepare a computer-generated Truth Table which will be returned to the user for verification.

NOTE 1: Card and Truth Table format available upon request.

PIN OPTIONS

Pins 6-40 of the AY-5-3600 are permanently assigned. The functions assigned to pins 1-5 depend on which functional options are selected from the following:

ROM

External Clock
—requires one package pin to input an external clock source.
Internal Oscillator
—requires three package pins interconnected with an external RC network to develop the clock required.
Lockout/Rollover (LO/RO)
—requires one package pin to externally select N-Key Lockout or N-Key Rollover. LO = +5V, RO = GND.
Complement Control (CC)
—requires one package pin to externally control the logic state of the data bits (B1-B10) and, if required, the Data Ready output.
Chip Enable (CE)
—requires one package pin to control the data bits (B1-B10) and, if required, the Data Ready and Any Key Output.
Any Key Output (AKO)
—requires one package pin to indicate a key depression.
Output Data Bit 10 (B10)
—requires one package pin when ten data bits are required to encode each key.

Select the pin options desired:
External Clock + 4 of the following functions
OR
Internal Oscillator + 2 of the following functions
LO/RO, CC, CE, AKO, BIO

The following chart lists the pin assignments according to the functions selected above:

PIN 1	PIN 2	PIN 3	PIN 4	PIN 5
External Clock	LO/RO	CC	CE	AKO
External Clock	LO/RO	CC	CE	BIO
External Clock	LO/RO	CC	AKO	BIO
External Clock	LO/RO	CE	AKO	BIO
External Clock	CC	CE	AKO	BIO

Internal Oscillator	LO/RO	CC
	LO/RO	CE
	LO/RO	AKO
	LO/RO	BIO
	CC	CE
	CC	AKO
	CC	BIO
	CE	AKO
	CE	BIO
	AKO	BIO

ELECTRICAL CHARACTERISTICS

Maximum Ratings*

V_{DD} and V_{GG} (with respect to V_{CC}) . . . . . −20V to +0.3V
Logic input voltages (with respect to V_{CC}) . . . . . −20V to +0.3V
Storage Temperature . . . . . −65°C to +150°C
Operating Temperature Range. . . . . 0°C to +70°C

*Exceeding these ratings could cause permanent damage. Functional operation of this device at these conditions is not implied—operating ranges are specified below.

Standard Conditions (unless otherwise noted)

V_{CC} = +5 Volts ±0.5 Volts
V_{GG} = −12 Volts ±1.0 Volts, V_{DD} = GND
(V_{CC} = Substrate Voltage)
Operating Temperature (T_A)= 0°C to +70°C

ELECTRICAL CHARACTERISTICS

Characteristics	Sym	Min	Typ**	Max	Units	Conditions
Clock Frequency	f	10	50	100	kHz	See Block diagram footnote* for typical R-C values
External Clock Width		7	—	—	μs	
Clock Input	V_{I0}	V_{GG}	—	.15	V	
	V_{I1}	V_{CC} −1.4	—	V_{CC} +0.3	V	
Data Input (Shift, Control, Complement Control, Lockout/Rollover, Chip Enable & External Clock)						
Logic "0" Level	V_{I0}	V_{GG}	—	+0.75	V	
Logic "1" Level	V_{I1}	V_{CC} −1.1	—	V_{CC}+0.3	V	
Shift & Control Input Current	I_{NSC}	75	95	120	μA	V_I = +5V
X Output (X_0-X_9)						
Logic "1" Output Current	I_{XI}	40	170	400	μA	V_{OUT} = V_{CC} (See Note 2)
		600	1300	2500	μA	V_{OUT} = V_{CC}−1.3V
		900	1600	3500	μA	V_{OUT} = V_{CC}−2.0V
		1500	3800	6000	μA	V_{OUT} = V_{CC}−5V
		3000	6000	10000	μA	V_{OUT} = V_{CC}−10V
Logic "0" Output Current	I_{XO}	8	15	50	μA	V_{OUT} = V_{CC}
		6	11	35	μA	V_{OUT} = V_{CC}−1.3V
		5	10	30	μA	V_{OUT} = V_{CC}−2.0V
		2	5	15	μA	V_{OUT} = V_{CC}−5V
		—	0.5	5	μA	V_{OUT} = V_{CC}−10V
Y Input (Y_0-Y_9)						
Trip Level	V_Y	V_{CC}−5	V_{CC}−3	V_{CC}−2	V	Y Input Going Positive (See Note 2)
Hysteresis	ΔV_Y	0.5	0.9	1.4	V	(See Note 1)
Selected Y Input Current	I_{YS}	18	36	100	μA	V_{IN} = V_{CC}
		14	28	90	μA	V_{IN} = V_{CC}−1.3V
		13	25	80	μA	V_{IN} = V_{CC}−2.0V
		6	12	60	μA	V_{IN} = V_{CC}−5V
		—	1	30	μA	V_{IN} = V_{CC}−10V
Unselected Y Input Current	I_{YU}	9	18	50	μA	V_{IN} = V_{CC}
		7	14	45	μA	V_{IN} = V_{CC}−1.3V
		6	13	40	μA	V_{IN} = V_{CC}−2.0V
		3	6	30	μA	V_{IN} = V_{CC}−5V
		—	0.5	15	μA	V_{IN} = V_{CC}−10V
Input Capacitance	C_{IN}	—	3	10	pF	at 0V (All Inputs)
X-Y Precharge Characteristics	φP	1500	3500	5000	μA	V = V_{CC}
		200	600	1500	μA	V = V_{CC}−5 (See Note 2)
Switch Characteristics						
Minimum Switch Closure	—	—	—	—	—	See Timing Diagram
Contact Closure Resistance	Z_{CC}	—	—	300	Ω	
	Z_{CO}	1 × 10⁷	—	—	Ω	
Strobe Delay						
Trip Level (Pin 31)	V_{SD}	V_{CC}−4	V_{CC}−3	V_{CC}−2	V	
Hysteresis	V_{SD}	0.5	0.9	1.4	V	(See Note 1)
Quiescent Voltage (Pin 31)		−3	−5	−9	V	With Internal Switched Resistor
Data Output (B1-B10), Any Key Down Output, Data Ready						
Logic "0"	—	—	—	.55	V	I_{OL} = .25mA
	—	—	—	0.8	V	I_{OL} = 1.6mA
Logic "1"	—	V_{CC} −1.3	—	—	V	I_{OH} = .95mA
Power						
I_{CC}	—	—	8	13	mA	V_{CC} = +5V
I_{GG}	—	—	8	13	mA	V_{GG} = −12V

**Typical values are at +25°C and nominal voltages.

NOTE
1. Hysteresis is defined as the amount of return required to unlatch an input.
2. Precharge of X outputs and Y inputs occurs during each scanned clock cycle.

ROM

OPERATION

The AY-5-3600 contains (see Block Diagram) a 3600 bit ROM, 9-stage and 10-stage ring counters, a 10 bit comparator, timing circuitry, a 90 bit memory to store the location of encoded keys for n key rollover operation, an externally controllable delay network.for eliminating the effect of contact bounce, an output data buffer, and TTL/DTL/MOS compatible output drivers.

The ROM portion of the chip is a 360 by 10 bit memory arranged into four 90-word by 10-bit groups. The appropriate levels on the Shift and Control Inputs selects one of the four 90-word groups; the 90-individual word locations are addressed by the two ring counters. Thus, the ROM address is formed by combining the Shift and Control Inputs with the two ring counters.

The external outputs of the 9-stage ring counter and the external inputs to the 10-bit comparator are wired to the keyboard to form an X-Y matrix with the 90-keyboard switches as the crosspoints. In the standby condition, when no key is depressed, the two ring counters are clocked and sequentially address the ROM, thereby scanning the key switches for key closures.

When a key is depressed, a single path is completed between one output of the 9-stage ring counter (X0 thru X8) and one input of the 10-bit comparator (Y_0-Y_9). After a number of clock cycles, a condition will occur where a level on the selected path to the comparator matches a level on the corresponding comparator input from the 10-stage ring counter.

N KEY ROLLOVER

— When a match occurs, and the key has not been encoded, the switch bounce delay network is enabled. If the key is still depressed at the end of the selected delay time, the code for the depressed key is transferred to the output data buffer, the data ready signal appears, a one is stored in the encoded key memory and the scan sequence is resumed. If a match occurs at another key location, the sequence is repeated thus encoding the next key. If the match occurs for an already encoded key, the match is not recognized. The code of the last key encoded remains in the output data buffer.

N KEY LOCKOUT

— When a match occurs, the delay network is enabled. If the key is still depressed at the end of the selected delay time, the code for the depressed key is transferred to the output data buffer, the data ready signal appears and the remaining keys are locked out by halting the scan sequence. The scan sequence is resumed upon key release. The output data buffer stores the code of the last key encoded.

SPECIAL PATTERNS

— Since the selected coding of each key and all the options are defined during the manufacture of the chip, the coding and options can be changed to fit any particular application of the keyboard. Up to 360 codes of up to 10 bits can be programmed into the AY-5-3600 ROM covering most popular codes such as ASCII, EBCDIC, Selectric, etc., as well as many specialized codes. The ASCII code in conjunction with internal oscillator, 10 data outputs and any key down output, is available as a standard pattern (See Figure 2).

ROM

TIMING DIAGRAM

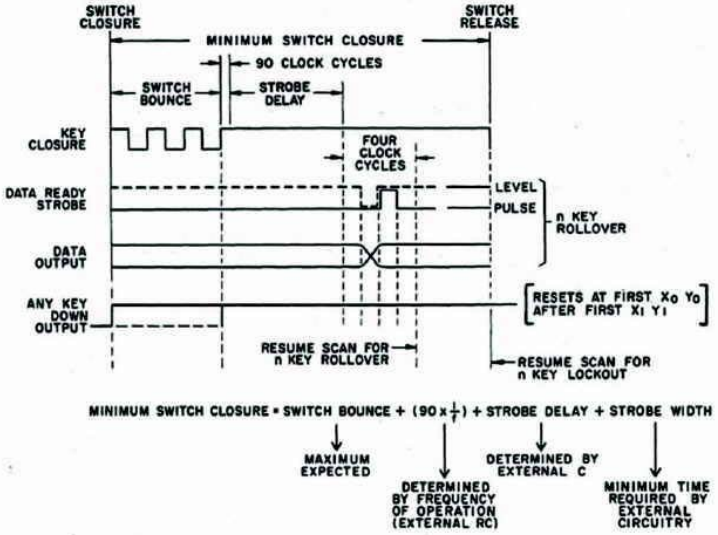


Fig.1

SYMBOL	MODE N	MODE S	MODE C	MODE SC
Φ		X1 Y2, X9 Y8		
A		X0 Y2		X1 Y2
B		X5 Y3		X2 Y2
C		X2 Y3		X3 Y2
D		X2 Y2		X4 Y2
E		X2 Y1		X5 Y2
F		X3 Y2		X6 Y2
G		X8 Y2		X7 Y2
H	X0 Y5	X0 Y5, X5 Y2	X0 Y5	X0 Y5
I		X7 Y1		X0 Y6
J		X6 Y2		X5 Y6
K		X7 Y2		X3 Y6
L	X2 Y6	X2 Y5, X8 Y2	X2 Y6	X2 Y6
M		X7 Y3		X2 Y5
N		X6 Y3		X4 Y5
O		X8 Y1		
P		X6 Y5		X0 Y2, X3 Y2
Q		X0 Y1		X1 Y3
R		X3 Y1		X2 Y3
S		X1 Y2		X4 Y3
T		X4 Y1		X5 Y3
U		X0 Y1		X6 Y3
V		X4 Y3		X7 Y3
W		X1 Y1		X5 Y5
X		X1 Y3		X8 Y2
Y		X5 Y1		X5 Y6
Z		X0 Y3		X5 Y5
a	X0 Y2		X0 Y2	
b	X5 Y3		X5 Y3	
c	X2 Y3		X2 Y3	
d	X2 Y2		X2 Y2	
e	X2 Y1		X2 Y1	
f	X3 Y2		X3 Y2	
g	X4 Y2		X4 Y2	
h	X5 Y2		X5 Y2	
i	X7 Y1		X7 Y1	
j	X6 Y2		X5 Y2	
k	X7 Y2, X2 Y5		X7 Y2	
l	X8 Y2		X8 Y2	
m	X7 Y3, X1 Y6		X7 Y3	

SYMBOL	MODE N	MODE S	MODE C	MODE SC
SOH			X5 Y8	X5 Y8, X0 Y8
STX			X1 Y8	X4 Y8, X1 Y8
ETX	X4 Y4	X4 Y4	X4 Y4	X4 Y4, X5 Y8
EOT				X4 Y1
ENQ				X3 Y1
ACK			X2 Y8	X7 Y1, X2 Y8
BEL			X3 Y8	X6 Y1, X3 Y8
BS				X3 Y4
HT	X0 Y4	X0 Y4	X0 Y4, X8 Y8	X8 Y8
LF	X7 Y6	X7 Y6	X7 Y6	
VT	X3 Y7	X3 Y7	X3 Y7	X3 Y7
FF	X7 Y8		X7 Y8	X7 Y8
CR	X3 Y5	X3 Y5	X3 Y5, X1 Y6	X1 Y6
SO	X0 Y7		X0 Y7, X1 Y8	X0 Y7, X1 Y8
SI	X1 Y7	X1 Y7	X1 Y7	X1 Y7
DLE				X3 Y1
DC1				X3 Y1
DC2				X6 Y7
DC3				X2 Y1
DC4				X3 Y0
NAK				X2 Y0
SYN				X5 Y4
ETB				X1 Y0
CAN	X3 Y4		X3 Y4	
EM				X8 Y0
SUB				X0 Y0
ESC				X7 Y0
FS				X1 Y4
GS				X7 Y6
RS	X1 Y6	X1 Y4	X1 Y4	
US	X2 Y7	X2 Y7	X2 Y7	X2 Y7
SP	X3 Y3, X4 Y8	X4 Y8, X3 Y3	X4 Y8, X3 Y3	X4 Y8, X3 Y3
!	X5 Y8	X5 Y8, X2 Y8	X5 Y8	X5 Y8
"	X3 Y8	X3 Y8, X7 Y5, X1 Y8	X3 Y8	X3 Y8, X7 Y5
#	X6 Y8	X6 Y8, X2 Y0	X6 Y8	X6 Y8
$	X2 Y5	X2 Y5, X2 Y8	X2 Y5	X2 Y5
%	X1 Y5	X1 Y5, X4 Y0	X1 Y5	X1 Y5
&	X8 Y8	X6 Y0, X6 Y8, X2 Y8	X6 Y8	X6 Y8
'	X7 Y5	X3 Y8	X7 Y5	X7 Y4
(	X7 Y8	X7 Y4, X3 Y4, X8 Y8	X7 Y8	X7 Y8

ROM

h	X6 Y2, X1 Y8		X6 Y2		(	X4 Y8	X4 Y8, X6 Y7, X8 Y8	X4 Y8	X4 Y8
a	X8 Y1		X6 Y1		*	X5 Y8	X5 Y8, X7 Y8, X5 Y4	X5 Y8	X5 Y8
p	X6 Y6, X2 Y8		X6 Y5		+	X2 Y5	X2 Y5, X5 Y6, X7 Y7	X2 Y5	X2 Y5, X7 Y7
q	X8 Y1		X0 Y1		,	X8 Y3	X8 Y3	X8 Y3	X8 Y2
r	X3 Y1		X3 Y1		—	X2 Y4	X2 Y4, X8 Y7	X2 Y6	X8 Y7
s	X1 Y2		X3 Y2		.	X6 Y4	X8 Y4	X8 Y4	X8 Y4
t	X4 Y1		X4 Y1		/	X2 Y6		X7 Y4	
u	X5 Y1		X6 Y1		0	X8 Y7, X8 Y8	X8 Y8	X6 Y7, X8 Y8	X8 Y8
v	X4 Y3		X4 Y3		1	X0 Y2, X0 Y3		X0 Y0	
w	X1 Y1		X1 Y1		2	X1 Y2, X1 Y3		X1 Y0	
x	X1 Y3		X1 Y2		3	X2 Y6		X2 Y0	
y	X5 Y1		X5 Y1		4	X3 Y2		X3 Y0	
z	X8 Y3		X0 Y3		5	X4 Y0		X4 Y0	
{		X8 Y5, X2 Y3		X4 Y5, X8 Y6	6	X5 Y2, X2 Y8		X5 Y0	
\				X1 Y1	7	X8 Y5, X3 Y8		X0 Y0	
}	X8 Y6	X1 Y5	X8 Y5	X8 Y1	8	X7 Y5		X7 Y0	
A		X1 Y8		X2 Y4	9	X8 Y2, X8 Y3		X8 Y0	
	X4 Y7, X8 Y7		X4 Y7, X8 Y7	X4 Y7	:	X5 Y4	X8 Y5	X5 Y4	X8 Y5
[	X3 Y6	X3 Y5	X3 Y6		;	X8 Y5, X5 Y6		X8 Y5, X5 Y6	
(	X4 Y5	X8 Y5	X4 Y5		<	X8 Y3	X7 Y8, X8 Y5, X0 Y8	X6 Y5	
~				X6 Y4	=	X8 Y4, X7 Y7	X7 Y7, X6 Y4, X4 Y7	X6 Y4	
DEL			X2 Y8	X2 Y8	>	X5 Y5	X5 Y5, X5 Y8, X0 Y7	X5 Y5	
NULL	X5 Y7	X5 Y7	X5 Y7, X0 Y8	X5 Y7, X0 Y8	?	X4 Y6	X4 Y8, X7 Y6	X4 Y6	

Note 1. Bits 1 to 6 and bit 8 of the AY-5-3600 correspond to bits 1 to 7 of ASC II.

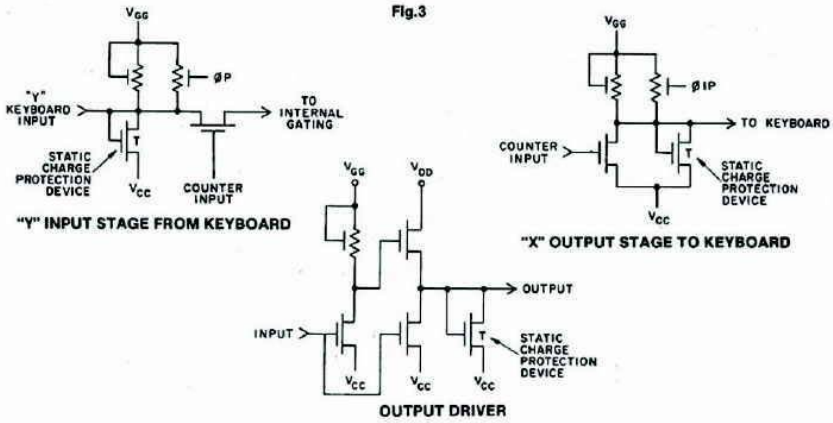
Note 2. Codes 0000011 and 0011111 are not present in the standard AY-5-3600 pattern.

Fig.2 STANDARD AY-5-3600 CODE ASSIGNMENTS ASCII CODE

OPTIONS PROVIDED WITH STANDARD ENCODER

- Device Marking: AY-5-3600
- Internal Oscillator on Pin Nos. 1, 2, 3
- Any Key Output on Pin No. 4
- Any Key Output True (Logic 1) During Key Depression
- Output Data Bit B10 on Pin No. 5
- N-Key Rollover Only
- True Outputs Only
- Pulse Data Ready Signal
- Internal Resistor to V_{DD} on Shift/Control Pin
- Plastic Package

Fig.3



NOTE: Output driver capable of driving one TTL load with no external resistor.
Capable of driving two TTL loads using an external 6.8KΩ resistor to V_{DD}.

TYPICAL CHARACTERISTIC CURVES

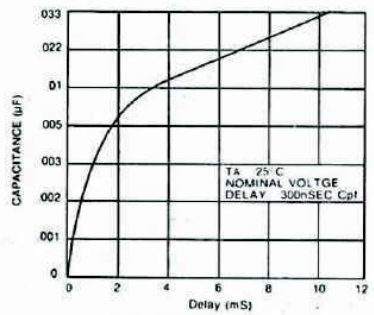


Fig.4 STROBE DELAY vs. C_1

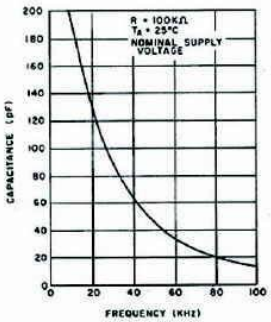
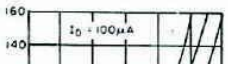


Fig.5 OSCILLATOR FREQUENCY vs. C_2



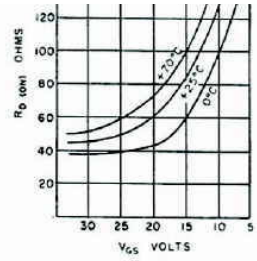


Fig.6 TYPICAL OUTPUT ON RESISTANCE (R_{DON}) vs. GATE BIAS VOLTAGE (V_{GS})

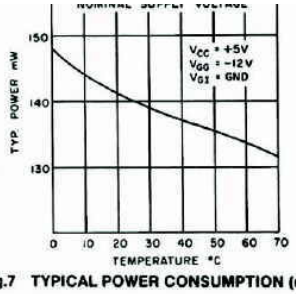


Fig.7 TYPICAL POWER CONSUMPTION (mW)

3-28

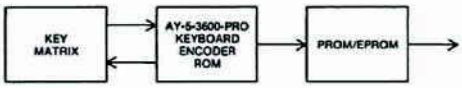
the AY-5-3600 chip datasheet

The AY-5-3600 PRO was the chip that was really used in the Apple IIe series....

GENERAL INSTRUMENT **AY-5-3600-PRO**

Keyboard Encoder and PROM/EPROM Application

The AY-5-3600-PRO is pre-programmed during manufacture to provide specific yet simple binary coded outputs thus allowing the purchase of off-the-shelf devices (distributors, etc.). To enhance the device flexibility, the binary outputs have been organized to provide direct interface with a PROM/EPROM.



The PROM (Programmable Read Only Memory) permits the programming of the required output code in the factory or the field within minutes, thus making it extremely suitable for small quantity, fast turnaround keyboard requirements. The EPROM (Erasable Programmable Read Only Memory) is ideally suited for prototyping, where patterns are quite variable, allowing the EPROM to be erased and reprogrammed repeatedly. Similar advantages are realized in the field where pattern changes are necessary in order to respond to redefined requirements or to subtle system peculiarities not previously encountered.

Technical Description

The AY-5-3600-PRO is a binary coded MOS-LSI device programmed to furnish 360 unique 9-bit codes (90 keys × 4 modes × 9 bits). Option selections include such popular functions as Internal Oscillator, Lockout/Rollover and an Any Key Down output. For further, more explicit device characteristics refer to the preceding pages. The internal oscillator is a self contained (on-chip) circuit option which eliminates the need for any external clock source. For applications necessitating an external clock source the internal oscillator input pins may be utilized to function in the slave mode of operation. Lockout or Rollover is selectable via an input pin, thus allowing the versatility required on various keyboard applications. The Any Key Down output performs the function of a gating signal by acknowledging both a key depression and release, making it a convenient signal for use in a repeat application.

For ease of translation, each key is assigned an X-Y coordinate and, in turn, each X-Y coordinate has been identified with a specific yet simple binary coded output. Two formats are described: the first for application with a 64 key 4 mode keyboard and the second for a 90 key 4 mode keyboard.

The 64 key 4 mode application as illustrated in Fig. 8 utilized keyboard encoder addresses X0 Y0 thru X6 Y3. A unique combination of one input (Y) and one output (X) is assigned to each key, for a total coverage of 64 keys. Binary coded outputs B2-B9 have been arranged to provide the necessary 8-bit address inputs to the PROM/EPROM, with B2 and B3 representing the variable mode identification and B4-B9 each specific key closure.

When a key is depressed a path is completed between one X line and one Y line thus addressing that specific X-Y ROM coordinate in the AY-5-3600-PRO. The 8-bit binary code for that X-Y location (ref. Truth Table page 14-15) is transferred into a one character 8-bit output latch (B2-B9) thus providing the appropriate 8-bit address to the 256 x 8 PROM/EPROM.

Expansion to a 90 key 4 mode operation (see Fig. 9) is identical to the 64 key 4 mode except: the 90 key 4 mode version utilizes the full complement of addresses X0 Y0 thru X8 Y9 (90 keys). The 8-bit binary code (B2-B9) previously produced to address the 256 x 8 PROM/EPROM is now expanded to a 9-bit binary code (B1-B9) for addressing to a 512 x 8 PROM/EPROM. With expansion to a 90 key 4 mode application outputs B1-B3 now serve as the variable mode identification.

The interface to a PROM/EPROM enables the custom programming of the required output data in the PROM/EPROM to directly coincide to the specific address inputs from the AY-5-3600-PRO. Any PROM whether it be bipolar, ultraviolet erasable or electrically alterable, may be employed to provide a wide variety of "off-the-shelf" keyboards. Once the keyboard assembly has gone beyond the prototyping stage, and assuming the quantity/cost permit, the PROM/EPROM data can be converted to the standard AY-5-3600 data format (ref. AY-5-3600 Custom Coding Information sheet) and produced in production quantities. This eliminates the PROM/EPROM expense while assuring the absence of undefined coding changes.

ROM

Summary of Important Features

- Ability to deliver complete keyboard assemblies within days without sacrificing the features offered in the AY-5-3600 Keyboard Encoder
- Ability to buy off-the-shelf devices (distributor, etc.)
- Ability to verify the specific pattern format using a PROM/EPROM prior to a 'custom' encoder commitment

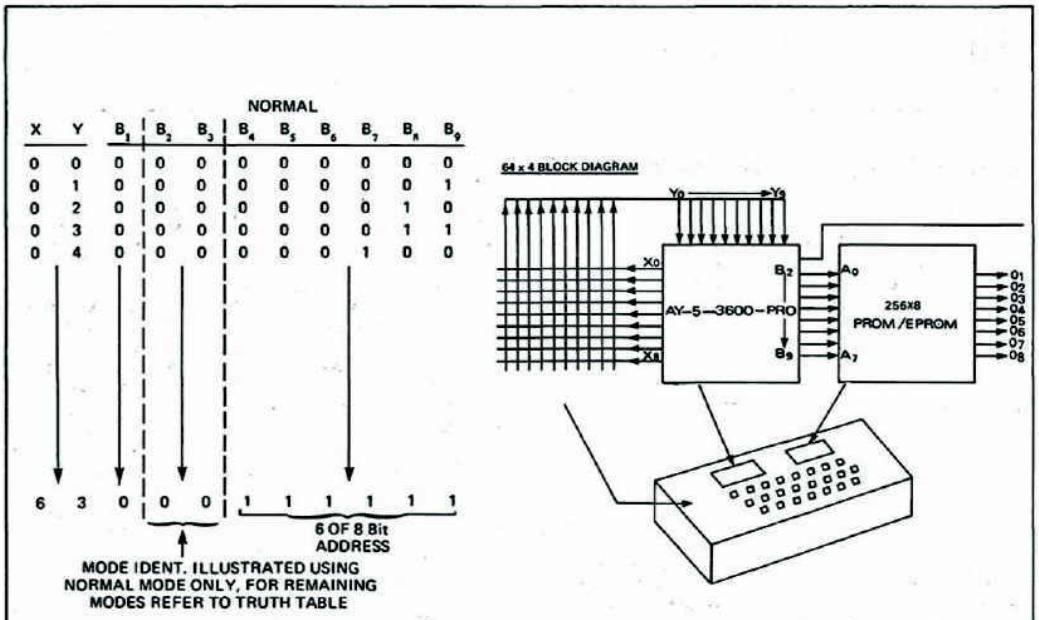


Fig.8 64 KEY 4 MODE KEYBOARD APPLICATION

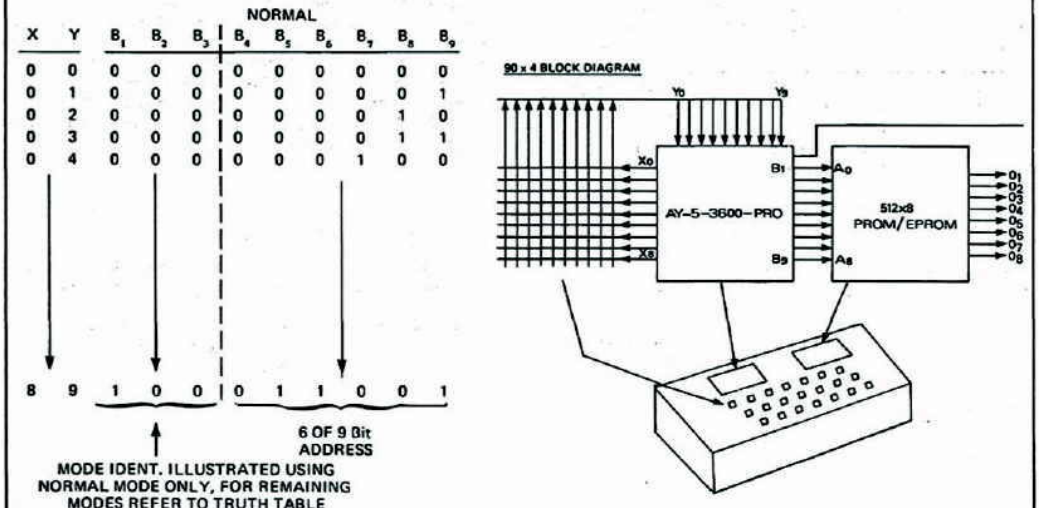


Fig.9 90 KEY 4 MODE KEYBOARD APPLICATION

OPTIONS

- Device Marking: AY-5-3600-PRO
- Internal Oscillator on Pin Nos. 1, 2, 3
- Lockout/Rollover on Pin No. 4
  Internal Resistor to V_{DD} on Lockout/Rollover Pin
- True Outputs Only
- Any Key Output on Pin No. 5.
  Any Key Output True (Logic 1) During Key Depression
- Pulse Data Ready Signal
- Plastic Package
- Internal Resistor to V_{DD} on Shift/Control Pin

XY	NORMAL	SHIFT	CONTROL	SHFT/CTR	XY	NORMAL	SHIFT	CONTROL	SHFT/CTR
0	000000000	001000000	010000000	011000000	45	000101101	001101101	010101101	011101101
1	000000001	001000001	010000001	011000001	46	000101110	001101110	010101110	011101110
2	000000010	001000010	010000010	011000010	47	000101111	001101111	010101111	011101111
3	000000011	001000011	010000011	011000011	48	000110000	001110000	010110000	011110000

4	000000100	001000100	010000100	011000100	49	000110001	001110001	010110001	011110001
5	000000101	001000101	010000101	011000101	50	000110010	001110010	010110010	011110010
6	000000110	001000110	010000110	011000110	51	000110011	001110011	010110011	011110011
7	000000111	001000111	010000111	011000111	52	000110100	001110100	010110100	011110100
8	000001000	001001000	010001000	011001000	53	000110101	001110101	010110101	011110101
9	000001001	001001001	010001001	011001001	54	000110110	001110110	010110110	011110110
10	000001010	001001010	010001010	011001010	55	000110111	001110111	010110111	011110111
11	000001011	001001011	010001011	011001011	56	000111000	001111000	010111000	011111000
12	000001100	001001100	010001100	011001100	57	000111001	001111001	010111001	011111001
13	000001101	001001101	010001101	011001101	58	000111010	001111010	010111010	011111010
14	000001110	001001110	010001110	011001110	59	000111011	001111011	010111011	011111011
15	000001111	001001111	010001111	011001111	60	000111100	001111100	010111100	011111100
16	000010000	001010000	010010000	011010000	61	000111101	001111101	010111101	011111101
17	000010001	001010001	010010001	011010001	62	000111110	001111110	010111110	011111110
18	000010010	001010010	010010010	011010010	63	000111111	001111111	010111111	011111111
19	000010011	001010011	010010011	011010011	64	100000000	101000000	110000000	111000000
20	000010100	001010100	010010100	011010100	65	100000001	101000001	110000001	111000001
21	000010101	001010101	010010101	011010101	66	100000010	101000010	110000010	111000010
22	000010110	001010110	010010110	011010110	67	100000011	101000011	110000011	111000011
23	000010111	001010111	010010111	011010111	68	100000100	101000100	110000100	111000100
24	000011000	001011000	010011000	011011000	69	100000101	101000101	110000101	111000101
25	000011001	001011001	010011001	011011001	70	100000110	101000110	110000110	111000110
26	000011010	001011010	010011010	011011010	71	100000111	101000111	110000111	111000111
27	000011011	001011011	010011011	011011011	72	100001000	101001000	110001000	111001000
28	000011100	001011100	010011100	011011100	73	100001001	101001001	110001001	111001001
29	000011101	001011101	010011101	011011101	74	100001010	101001010	110001010	111001010
30	000011110	001011110	010011110	011011110	75	100001011	101001011	110001011	111001011
31	000011111	001011111	010011111	011011111	76	100001100	101001100	110001100	111001100
32	000100000	001100000	010100000	011100000	77	100001101	101001101	110001101	111001101
33	000100001	001100001	010100001	011100001	78	100001110	101001110	110001110	111001110
34	000100010	001100010	010100010	011100010	79	100001111	101001111	110001111	111001111
35	000100011	001100011	010100011	011100011	80	100010000	101010000	110010000	111010000
36	000100100	001100100	010100100	011100100	81	100010001	101010001	110010001	111010001
37	000100101	001100101	010100101	011100101	82	100010010	101010010	110010010	111010010
38	000100110	001100110	010100110	011100110	83	100010011	101010011	110010011	111010011
39	000100111	001100111	010100111	011100111	84	100010100	101010100	110010100	111010100
40	000101000	001101000	010101000	011101000	85	100010101	101010101	110010101	111010101
41	000101001	001101001	010101001	011101001	86	100010110	101010110	110010110	111010110
42	000101010	001101010	010101010	011101010	87	100010111	101010111	110010111	111010111
43	000101011	001101011	010101011	011101011	88	100011000	101011000	110011000	111011000
44	000101100	001101100	010101100	011101100	89	100011001	101011001	110011001	111011001

ROM

3-31

the AY-5-3600-PRO decoderchip

The KR3600 decoderchip sound quite similar to AY-5-3600 but it isn´t similar at all ! But it was acommon used chip in third-party keyboards.

STANDARD MICROSYSTEMS CORPORATION

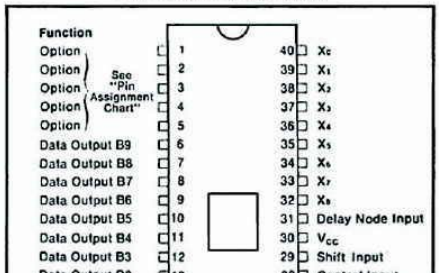
KR3600-XX
KR3600-ST
KR3600-STD
KR3600-PRO

Keyboard Encoder Read Only Memory

FEATURES

- Data output directly compatible with TTL
- N Key rollover or lockout operation
- Quad mode
- Lockout/rollover selection externally selected as option
- On chip-master/slave oscillator
- All 10 output bits available
- Fully buffered data outputs
- Output enable provided as option
- Data compliment control provided as option
- Pulse or level data ready output signal provided as an option
- Any key down output provided as an option
- Contact bounce circuit provided to eliminate contact bounce
- Static charge protection on all input/outputs
- Pin for Pin replacement for GI AY-5-3600

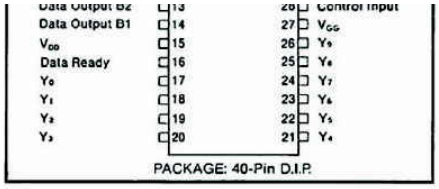
PIN CONFIGURATION



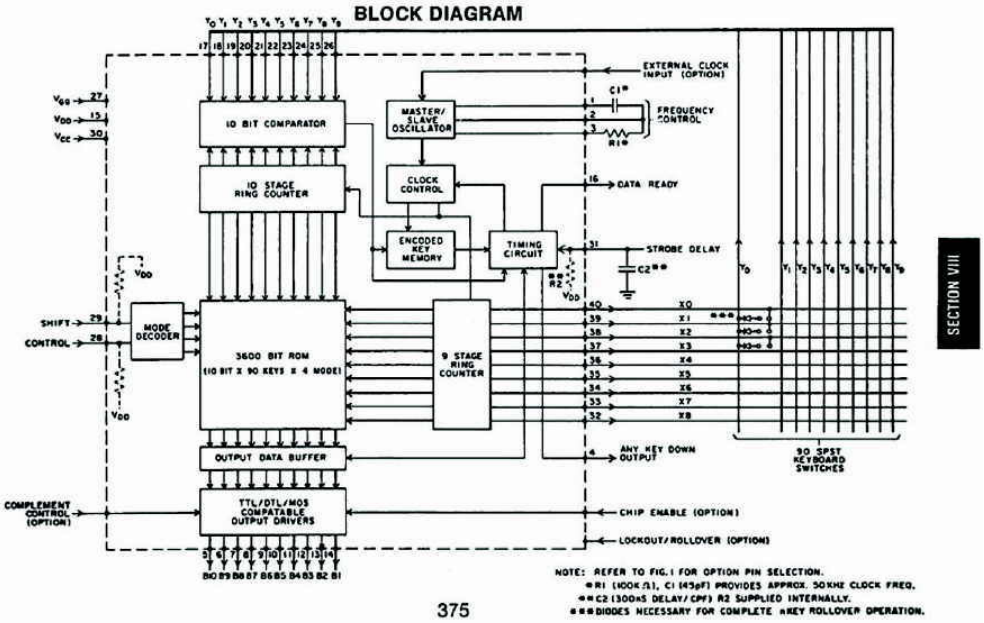
GENERAL DESCRIPTION

The SMC Microsystems KR3600-XX is a Keyboard Encoder containing a 3600 bit read only memory and all the logic necessary to encode single pole single throw keyboard closures into a 10 bit code.

The KR3600-XX is fabricated with a low voltage p channel technology and contains the equivalent of 5000 transistors on a monolithic chip in a 40 lead dip ceramic package.



BLOCK DIAGRAM



NOTE: REFER TO FIG. 1 FOR OPTION PIN SELECTION.
* R1 (100K Ω), C1 (43pF) PROVIDES APPROX. 50KHz CLOCK FREQ.
** C2 (300nS DELAY/ CPF) R2 SUPPLIED INTERNALLY.
*** DIODES NECESSARY FOR COMPLETE n KEY ROLLOVER OPERATION.

SECTION VIII

DESCRIPTION OF OPERATION

The KR3600 contains a 3600 bit ROM, 9-stage and 10-stage ring counters, a 10 bit comparator, timing circuitry, a 90 bit memory to store the location of encoded keys for n key rollover operation, an externally controllable delay network for eliminating the effect of contact bounce, an output data buffer, and TTL/DTL/MOS compatible output drivers.

The ROM portion of the chip is a 360 by 10 bit memory arranged into four 90-word by 10-bit groups. The appropriate levels on the Shift and Control Inputs selects one of the four 90-word groups; the 90-individual word locations are addressed by the two ring counters. Thus, the ROM address is formed by combining the Shift and Control Inputs with the two ring counters.

The external outputs of the 9-stage ring counter and the external inputs to the 10-bit comparator are wired to the keyboard to form an X-Y matrix with the 90-keyboard switches as the crosspoints. In the standby conditions, when no key is depressed, the two ring counters are clocked and sequentially address the ROM, thereby scanning the key switches for key closures.

When a key is depressed, a single path is completed between one output of the 9-stage ring counter (X0 thru X8) and one input of the 10-bit comparator (Y_0-Y_9). After a number of clock cycles, a condition will occur where a level on the selected path to the comparator matches a level on the corresponding comparator input from the 10-stage ring counter.

N KEY ROLLOVER — When a match occurs, and the key has not been encoded, the switch bounce delay network is enabled. If the key is still depressed at the end of the selected delay time, the code for the depressed key is transferred to the output data buffer, the data ready signal appears, a one is stored in the encoded key memory and the scan sequence is resumed. If a match occurs at another key location, the sequence is repeated thus encoding the next key. If the match occurs for an already encoded key, the match is not recognized. The code of the last key encoded remains in the output data buffer.

N KEY LOCKOUT — When a match occurs, the delay network is enabled. If the key is still depressed at the end of the selected delay time, the code for the depressed key is transferred to the output data buffer, the data ready signal appears and the remaining keys are locked out by halting the scan sequence. The scan sequence is resumed upon key release. The output data buffer stores the code of the last key encoded.

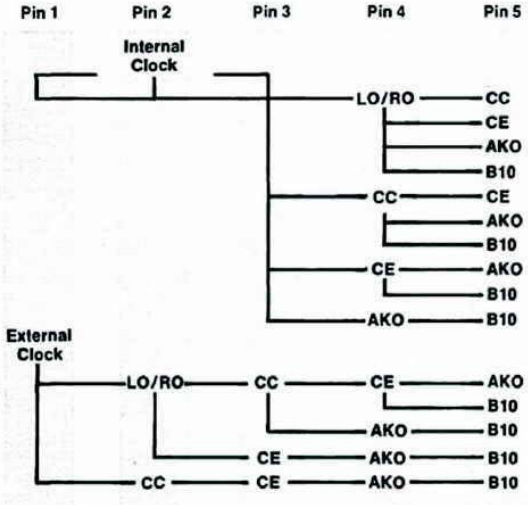
SPECIAL PATTERNS — Since the selected coding of each key and all the options are defined during the manufacture of the chip, the coding and options can be changed to fit any particular application of the keyboard. Up to 360 codes of up to 10 bits can be programmed into the KR3600 ROM covering most popular codes such as ASCII, EBCDIC, Selectric, etc., as well as many specialized codes.

CUSTOM CODING INFORMATION

The custom coding information for SMC's 3600 Bit Keyboard Encoder ROM should be transmitted to SMC. The Truth Table should be completed on the format supplied.

LEGEND

CC = Complement Control
AKO = Any Key Down Output
B10 = B10 (Data) Output
LO/RO = Lockout/Rollover
CE = Chip Enable
Internal Clock = Self Contained Oscillator
External Clock = External Frequency Source



Pin 1	Pin 2	Pin 3	Pin 4	Pin 5

OPTION SELECTION/PIN ASSIGNMENT
FIGURE 1

MAXIMUM GUARANTEED RATINGS*

Operating Temperature Range	0°C to +70°C
Storage Temperature Range	−55°C to +150°C
Lead Temperature (soldering, 10 sec.)	+325°C
Positive Voltage on any Pin, V_{CC}	+0.3 V
Negative Voltage on any Pin, V_{CC}	−25 V

*Stresses above those listed may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other condition above those indicated in the operational sections of this specification is not implied.

ELECTRICAL CHARACTERISTICS
(T_A = 0°C to 70°C, V_{CC} = +5V ±5%, V_{GG} = −12V ±1.0V, V_{DD} = GND, unless otherwise noted)

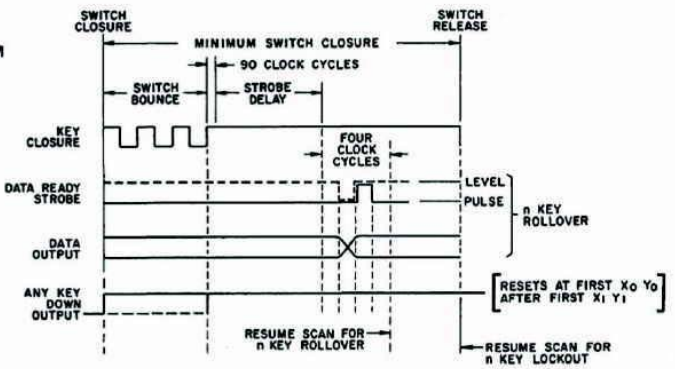
Characteristics	Min	Typ**	Max	Units	Conditions
Clock Frequency	10	50	100	KHz	See Block diagram footnote* for typical R-C values
External Clock Width	7	—	—	μs	
Data & Clock Input (Shift, Control, Compliment Control, Lockout/Rollover, Chip Enable & External Clock)					
Logic "0" Level	V_{GG}	—	+0.8	V	
Logic "1" Level	V_{CC}−1.5	—	V_{CC}+0.3	V	
Shift & Control Input Current	75	150	220	μA	V_{IN} = +5V
X Output (X_0-X_8)					
Logic "1" Output Current	40	250	500	μA	V_{OUT} = V_{CC} (See Note 2)
	600	1300	4000	μA	V_{OUT} = V_{CC}−1.3V
	900	2000	6500	μA	V_{OUT} = V_{CC}−2.0V
	1500	2000	14,000	μA	V_{OUT} = V_{CC}−5V
	3000	10,000	23,000	μA	V_{OUT} = V_{CC}−10V
Logic "0" Output Current	8	30	60	μA	V_{OUT} = V_{CC}
	6	25	50	μA	V_{OUT} = V_{CC}−1.3V
	5	20	45	μA	V_{OUT} = V_{CC}−2.0V
	2	10	30	μA	V_{OUT} = V_{CC}−5V
	—	0.5	5	μA	V_{OUT} = V_{CC}−10V
Y Input (Y_0-Y_9)					
Trip Level	V_{CC}−5	V_{CC}−3	V_{CC}−2	V	Y Input Going Positive (See Note 2)
Hysteresis	0.5	0.9	1.4	V	(See Note 1)
Selected Y Input Current	18	100	170	μA	V_{IN} = V_{CC}
	14	80	150	μA	V_{IN} = V_{CC}−1.3V
	13	50	130	μA	V_{IN} = V_{CC}−2.0V
	5	40	110	μA	V_{IN} = V_{CC}−4.0V
Unselected Y Input Current	9	40	80	μA	V_{IN} = V_{CC}
	7	30	70	μA	V_{IN} = V_{CC}−1.3V
	6	25	60	μA	V_{IN} = V_{CC}−2.0V
	3	15	40	μA	V_{IN} = V_{CC}−5V
	—	0.5	20	μA	V_{IN} = V_{CC}−10V
Input Capacitance	—	3	10	pF	at 0V (All Inputs)
Switch Characteristics					
Minimum Switch Closure	—	—	—	—	See Timing Diagram
Contact Closure Resistance	—	—	300	Ω	Z_{CC}
	1 x 10^7	—	—	Ω	Z_{CO}
Strobe Delay					
Trip Level (Pin 31)	V_{CC}−4	V_{CC}−3	V_{CC}−2	V	
Hysteresis	0.5	0.9	1.4	V	(See Note 1)
Quiescent Voltage (Pin 31)	−3	−5	−9	V	With Internal Switched Resistor
Data Output (B1-B10), Any Key Down Output, Data Ready					
Logic "0"	—	—	0.4	V	I_{OL} = 1.6m A
Logic "1"	V_{CC}−1	—	—	V	I_{OH} = 1.0m A
	V_{CC}−2	—	—	V	I_{OH} = 2.2m A
Power					
I_{CC}	—	12	25	mA	V_{CC} = +5V
I_{GG}	—	12	25	mA	V_{GG} = −12V

**Typical values are at +25°C and nominal voltages.

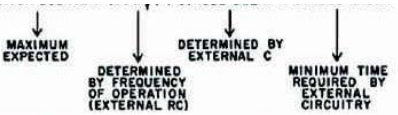
NOTE
1. Hysteresis is defined as the amount of return required to unlatch an input.
2. Precharge of X outputs and Y inputs occurs during each scanned clock cycle.

SECTION VIII

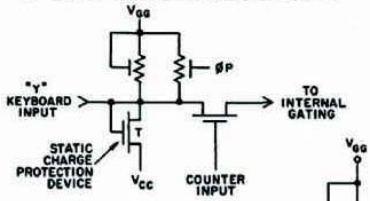
TIMING DIAGRAM



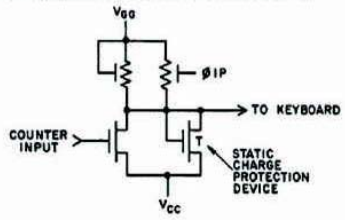
MINIMUM SWITCH CLOSURE = SWITCH BOUNCE + (90 x $\frac{1}{f_c}$) + STROBE DELAY + STROBE WIDTH



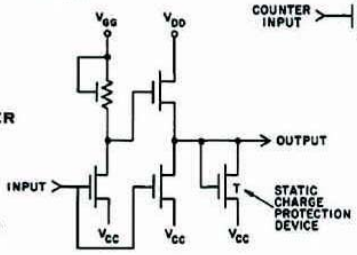
"Y" INPUT STAGE FROM KEYBOARD



"X" OUTPUT STAGE TO KEYBOARD

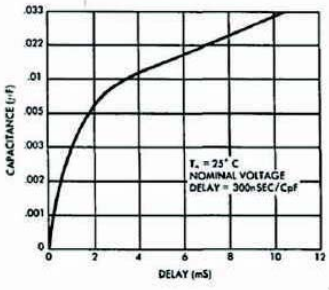


OUTPUT DRIVER

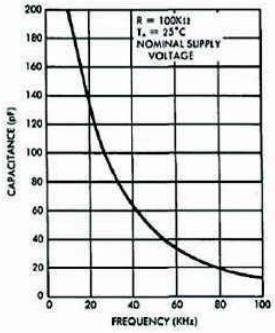


NOTE: Output driver capable of driving one TTL load with no external resistor. Capable of driving two TTL loads using an external 6.8KΩ resistor to V_{GG}

STROBE DELAY vs. C_2



OSCILLATOR FREQUENCY vs. C_1



KR3600-STD

XY	Normal B-12345678910	Shift B-12345678910	Control B-12345678910	Shift Control B-12345678910
00	1 1000111001	< 0011111001	1 1000111011	SUB 0101100001
01	q 1000110101	Q 1000100101	q 1000111111	DLE 0000100001
02	a 1000010101	A 1000000101	a 1000011111	@ 0000000101
03	z 0101110101	Z 0101100101	z 0101111111	P 0000100101
04	HT 1001000001	HT 1001000001	HT 1001000001	I 1001000101
05	H 0001000101	H 0001000101	H 0001000101	H 0001000111
06	+ 1101011001	+ 1101011001	+ 1101011001	+ 1101011011
07	SO 0111001001	> 0111111001	SO 0111000001	SO 0111000011
08	p 0000110101	@ 0000000101	NUL 0000000001	NUL 0000000001
09	1 1000111001	! 1000011001	SOH 1000000001	SOH 1000000001
10	2 0100111001	@ 0000000101	2 0100111011	ETB 1110100001
11	w 1110110101	W 1110100101	w 1110111111	^ 0011100101
12	s 1100110101	S 1100100101	s 1100111111	A 1000000101
13	x 0001110101	X 0001100101	x 0001111111	Q 1000100101
14	RS 0111100001	RS 0111100001	RS 0111100001	FS 0011100001
15	% 1010011001	% 1010011001	% 1010011001	% 1010011011
16	m 1011010101	] 1011100101	CR 1011000001	CR 1011000001
17	SI 1111000001	SI 1111000001	SI 1111000001	SI 1111000011
18	n 0111010101	^ 0111100101	SO 0111000001	SO 0111000001
19	2 0100111001	" 0100011001	STX 0100000001	STX 0100000001
20	3 1100111001	# 1100011001	3 1100111011	NAK 1010100001
21	e 1010010101	E 1010000101	e 1010011111	DC3 1100100001
22	d 0010010101	D 0010000101	d 0010011111	B 0100000101
23	c 1100010101	C 1100000101	c 1100011111	R 0100100101
24	— 1111100100	— 1111100100	— 1111100100	^ 0111100100
25	$ 0010011001	$ 0010011001	$ 0010011001	$ 0010011011
26	L 0011000101	L 0011000101	L 0011000101	L 0011000111
27	US 1111100001	US 1111100001	US 1111100001	US 1111100011
28	6 0110111001	& 0110011001	ACK 0110000001	ACK 0110000001
29	k 1101010101	[1101100101	DEL 1111111101	DEL 1111111101
30	4 0010111001	$ 0010011001	4 0010111011	DC4 0010100001
31	r 0100110101	R 0100100101	r 0100111111	ENQ 1010000001
32	f 0110010101	F 0110000101	f 0110011111	C 1100000101
33	SP 0000011000	SP 0000011000	SP 0000011000	SP 0000011000
34	CAN 0001101000	(0001011000	CAN 0001100000	BS 0001000000
35	CR 1011000001	CR 1011000001	CR 1011000001	M 1011000101
36	[1101111101	[1101111101	[1101111111	K 1101000101
37	VT 1101000000	VT 1101000000	VT 1101000000	VT 1101000010
38	7 1110111001	' 1110011001	BEL 1110000001	BEL 1110000001
39	" 0100011001	" 0100011001	" 0100011001	" 0100011011
40	5 1010111001	% 1010011001	5 1010111011	STX 0100000001
41	t 0010110101	T 0010100101	t 0010111111	EOT 0010000001
42	g 1110010101	g 1110000101	G 1110011111	D 0010000101
43	v 0110110101	V 0110100101	v 0110111111	S 1100100101
44	ETX 1100000001	ETX 1100000001	ETX 1100000001	ETX 1100000001
45	] 1011111101	] 1011111101	] 1011111111	N 0111000101
46	? 1111111001	? 1111111001	? 1111111011	[1101100101
47	— 1011011001	— 1011111001	— 1011011001	— 1011011011
48	) 1001011001	) 1001011001	) 1001011001	) 1001011011
49	SP 0000011001	SP 0000011001	SP 0000011001	SP 0000011011
50	6 0110111001	> 0111111001	6 0110111011	SOH 1000000001
51	y 1001110101	Y 1001100101	y 1001111111	DC1 1000100001
52	h 0001010101	H 0001000101	h 0001011111	E 1010000101
53	b 0100010101	B 0100000101	b 0100011111	T 0010100101
54	: 0101111001	* 0101011001	: 0101111011	SYN 0110100001
55	> 0111111001	> 0111111001	> 0111111011	Z 0101100101
56	; 1101111001	+ 1101011001	; 1101111011	Y 1001100101
57	NUL 0000000001	NUL 0000000001	NUL 0000000001	NUL 0000000001
58	* 0101011001	* 0101011001	* 0101011001	* 0101011011
59	! 1000011001	! 1000011001	! 1000011001	! 1000011011
60	7 1110111001	& 0110011001	7 1110111011	ETX 1100000001
61	u 1010110101	U 1010100101	u 1010111111	BEL 1110000001
62	j 0101010101	J 0101000101	j 0101011111	F 0110000101
63	n 0111010101	N 0111000101	n 0111011111	U 1010100101
64	= 1011111000	= 1011111000	= 1011111010	~ 0111111100
65	< 0011111001	< 0011111001	< 0011111011	W 1110100101
66	p 0000110101	P 0000100101	p 0000111111	J 0101000101
67	0 0000111001	) 1001011001	0 0000111011	DC2 0100100001
68	& 0110011001	& 0110011001	& 0110011001	& 0110011011
69	# 1100011001	# 1100011001	# 1100011001	# 1100011011
70	8 0001111001	* 0101011001	8 0001111011	ESC 1101100001
71	i 1001010101	I 1001000101	i 1001011111	ACK 0110000001

72	k 1101010101	K 1101000101	k 1101011111	G 1110000101
73	m 1011010101	M 1011000101	m 1011011111	V 0110100101
74	/ 1111011001	? 1111111001	/ 1111011001	' 1110011001
75	' 1110011001	" 0100011001	' 1110011001	" 0100011001
76	LF 0101000000	LF 0101000000	LF 0101000000	GS 1011100000
77	= 1011111001	+ 1101011001	= 1011111001	+ 1101011001
78	FF 0011001001	< 0011111001	FF 0011000001	FF 0011000011
79	(0001011001	(0001011001	(0001011001	(0001011011
80	9 1001111001	(0001011001	9 1001111011	EM 1001100001
81	o 1111010101	O 1111000101	o 1111011111	] 1011100101
82	I 0011010101	L 0011000101	I 0011011111	X 0001100101
83	. 0011011001	. 0011011001	. 0011011001	. 0011011011
84	. 0111011001	. 0111011001	. 0111011001	. 0111011011
85	; 1101111001	: 0101111001	; 1101111001	: 0101111001
86	] 1011100101	[1101100101	] 1011100101	[1101100101
87	- 1011011001	_ 1111100101	- 1011011001	_ 1111100101
88	0 0000111001	0 0000111001	0 0000111001	0 0000111001
89	9 1001111001	) 1001011001	HT 1001000001	HT 1001000001

Options:
Internal oscillator (pins 1, 2, 3)
Any key down (pin 4) positive output
N key rollover only
Pulse data ready signal
Internal resistor to Vpp on shift and control pins
KR3600-STD outputs provides ASCII bits 1-6 on B1-B6, and bit 7 on B8

KR 3600-ST

XY	Normal B-123456789	Shift B-123456789	Control B-123456789	Shift/Control B-123456789
00	\ 000001101	~ 011111101	NUL 000000001	RS 011110001
01	= 101111010	+ 110101001	GS 101110001	VT 110100010
02	DC3 110010010	DC3 110010010	DC3 110010010	DC3 110010010
03	— 101101001	— 111110101	CR 101100010	US 111110010
04	BS 000100010	BS 000100010	BS 000100010	BS 000100010
05	0 000011001	0 000011001	0 000011001	0 000011001
06	• 011101001	• 011101001	• 011101001	• 011101001
07	000000000	000000000	000000000	000000000
08	000000000	000000000	000000000	000000000
09	000000000	000000000	000000000	000000000
10	/ 111101010	? 111111001	ST 111100001	US 111110010
11	• 011101001	> 011111010	SO 011100010	RS 011110001
12	? 001101010	< 001111001	FF 001100001	FS 001110010
13	m 101101110	M 101100101	CR 101100010	CR 101100010
14	n 011101110	N 011100101	SO 011100010	SO 011100010
15	b 010001110	B 010000101	STX 010000010	STX 010000010
16	v 011011110	V 011010101	SYN 011010010	SYN 011010010
17	c 110001101	C 110000110	ETX 110000001	ETX 110000001
18	x 000111101	X 000110110	CAN 000110001	CAN 000110001
19	z 010111110	Z 010110101	SUB 010110010	SUB 010110010
20	LF 010100001	LF 010100001	LF 010100001	LF 010100001
21	\ 001110101	: 001111110	FS 001110010	FS 001110010
22	DEL 111111110	DEL 111111110	DEL 111111110	DEL 111111110
23	[110110110	] 101110110	ESC 110110001	GS 101110001
24	7 111011010	7 111011010	7 111011010	7 111011010
25	8 000111010	8 000111010	8 000111010	8 000111010
26	9 100111001	9 100111001	9 100111001	9 100111001
27	000000000	000000000	000000000	000000000
28	000000000	000000000	000000000	000000000
29	000000000	000000000	000000000	000000000
30	; 110111010	: 010111001	ESC 110110001	SUB 010110010
31	i 001101101	L 001100110	FF 001100001	FF 001100001
32	k 110101110	K 110100101	VT 110100010	VT 110100010
33	j 010101101	J 010100110	LF 010100001	LF 010100001
34	h 000101110	H 000100101	BS 000100010	BS 000100010
35	g 111001110	G 111000101	BEL 111000010	BEL 111000010
36	f 011001101	F 011000110	ACK 011000001	ACK 011000001
37	d 001001110	D 001000101	EOT 001000010	EOT 001000010
38	s 110011110	S 110010101	DC3 110010010	DC3 110010010
39	a 100001110	A 100000101	SOH 100000010	SOH 100000010
40	000000000	000000000	000000000	000000000
41	[110111101	[101111101	ESC 110110001	GS 101110001
42	GR 101100010	GR 101100010	GR 101100010	GR 101100010
43	' 111001001	" 010001001	BEL 111000010	STX 010000010
44	4 001011010	4 001011010	4 001011010	4 001011010
45	5 101011001	5 101011001	5 101011001	5 101011001
46	6 011011001	6 011011001	6 011011001	6 011011001
47	000000000	000000000	000000000	000000000
48	000000000	000000000	000000000	000000000
49	000000000	000000000	000000000	000000000
50	p 000011110	P 000010101	DEL 000010010	DEL 000010010
51	o 111101101	O 111100110	SI 111100001	SI 111100001
52	i 100101101	I 100100110	HT 100100001	HT 100100001
53	u 101011110	U 101010101	NAK 101010010	NAK 101010010
54	y 100111110	Y 100110101	EM 100110010	EM 100110010
55	t 001011101	T 001010110	DC4 001010001	DC4 001010001
56	r 010011101	R 010010110	DC2 010010001	DC2 010010001
57	e 101001101	E 101000110	ENQ 101000001	ENQ 101000001
58	w 111011101	W 111010110	ETB 111010001	ETB 111010001
59	q 100011101	Q 100010110	DC1 100010001	DC1 100010001
60	000000000	000000000	000000000	000000000
61	000000000	000000000	000000000	000000000
62	DC2 010010001	DC2 010010001	DC2 010010001	DC2 010010001
63	000000000	000000000	000000000	000000000
64	1 100011010	1 100011010	1 100011010	1 100011010
65	2 010011010	2 010011010	2 010011010	2 010011010
66	3 110011001	3 110011001	3 110011001	3 110011001
67	000000000	000000000	000000000	000000000
68	000000000	000000000	000000000	000000000
69	000000000	000000000	000000000	000000000
70	0 000011001	) 100101010	DLE 000010010	HT 100100001
71	9 100111001	(000101001	EM 100110010	BS 000100010
72	8 000111010	* 010101010	CAN 000110001	LF 010100001
73	7 111011010	& 011001010	ETB 111010001	ACK 011000001
74	6 011011001	A 011110110	SYN 011010010	RS 011110001
75	5 101011001	% 101001010	NAK 101010010	ENQ 101000001
76	4 001011010	$ 001001001	DC4 001010001	EOT 001000010
77	3 110011001	# 110001010	DC3 110010010	ETX 110000001
78	2 010011010	@ 000000110	DC2 010010001	NUL 000000001
79	1 100011010	! 100001001	DC1 100010001	SOH 100000010
80	000000000	000000000	000000000	000000000
81	000000000	000000000	000000000	000000000
82	000000000	000000000	000000000	000000000
83	000000000	000000000	000000000	000000000
84	000000000	000000000	000000000	000000000
85	SP 000001010	SP 000001010	NUL 000000001	NUL 000000001
86	000000000	000000000	000000000	000000000
87	DC1 100010001	DC1 100010001	DC1 100010001	DC1 100010001
88	HT 100100001	HT 100100001	HT 100100001	HT 100100001
89	ESC 110110001	ESC 110110001	ESC 110110001	ESC 110110001

Options: Pin 1, 2, 3—Internal oscillator
Pin 4—Lockout (logic 1), rollover (logic 0)
Pin 5—Any key down output
All outputs complemented
Level data ready

KR 3600-PRO

XY	Normal	Shift	Control	Shift/Control
00	000000000	001000000	010000000	011000000

01	000000001	001000001	010000001	011000001
02	000000010	001000010	010000010	011000010
03	000000011	001000011	010000011	011000011
04	000000100	001000100	010000100	011000100
05	000000101	001000101	010000101	011000101
06	000000110	001000110	010000110	011000110
07	000000111	001000111	010000111	011000111
08	000001000	001001000	010001000	011001000
09	000001001	001001001	010001001	011001001
10	000001010	001001010	010001010	011001010
11	000001011	001001011	010001011	011001011
12	000001100	001001100	010001100	011001100
13	000001101	001001101	010001101	011001101
14	000001110	001001110	010001110	011001110
15	000001111	001001111	010001111	011001111
16	000010000	001010000	010010000	011010000
17	000010001	001010001	010010001	011010001
18	000010010	001010010	010010010	011010010
19	000010011	001010011	010010011	011010011
20	000010100	001010100	010010100	011010100
21	000010101	001010101	010010101	011010101
22	000010110	001010110	010010110	011010110
23	000010111	001010111	010010111	011010111
24	000011000	001011000	010011000	011011000
25	000011001	001011001	010011001	011011001
26	000011010	001011010	010011010	011011010
27	000011011	001011011	010011011	011011011
28	000011100	001011100	010011100	011011100
29	000011101	001011101	010011101	011011101
30	000011110	001011110	010011110	011011110
31	000011111	001011111	010011111	011011111
32	000100000	001100000	010100000	011100000
33	000100001	001100001	010100001	011100001
34	000100010	001100010	010100010	011100010
35	000100011	001100011	010100011	011100011
36	000100100	001100100	010100100	011100100
37	000100101	001100101	010100101	011100101
38	000100110	001100110	010100110	011100110
39	000100111	001100111	010100111	011100111
40	000101000	001101000	010101000	011101000
41	000101001	001101001	010101001	011101001
42	000101010	001101010	010101010	011101010
43	000101011	001101011	010101011	011101011
44	000101100	001101100	010101100	011101100
45	000101101	001101101	010101101	011101101
46	000101110	001101110	010101110	011101110
47	000101111	001101111	010101111	011101111
48	000110000	001110000	010110000	011110000
49	000110001	001110001	010110001	011110001
50	000110010	001110010	010110010	011110010
51	000110011	001110011	010110011	011110011
52	000110100	001110100	010110100	011110100
53	000110101	001110101	010110101	011110101
54	000110110	001110110	010110110	011110110
55	000110111	001110111	010110111	011110111
56	000111000	001111000	010111000	011111000
57	000111001	001111001	010111001	011111001
58	000111010	001111010	010111010	011111010
59	000111011	001111011	010111011	011111011
60	000111100	001111100	010111100	011111100
61	000111101	001111101	010111101	011111101
62	000111110	001111110	010111110	011111110
63	000111111	001111111	010111111	011111111
64	100000000	101000000	110000000	111000000
65	100000001	101000001	110000001	111000001
66	100000010	101000010	110000010	111000010
67	100000011	101000011	110000011	111000011
68	100000100	101000100	110000100	111000100
69	100000101	101000101	110000101	111000101
70	100000110	101000110	110000110	111000110
71	100000111	101000111	110000111	111000111
72	100001000	101001000	110001000	111001000
73	100001001	101001001	110001001	111001001
74	100001010	101001010	110001010	111001010
75	100001011	101001011	110001011	111001011
76	100001100	101001100	110001100	111001100
77	100001101	101001101	110001101	111001101
78	100001110	101001110	110001110	111001110
79	100001111	101001111	110001111	111001111
80	100010000	101010000	110010000	111010000
81	100010001	101010001	110010001	111010001
82	100010010	101010010	110010010	111010010
83	100010011	101010011	110010011	111010011
84	100010100	101010100	110010100	111010100
85	100010101	101010101	110010101	111010101
86	100010110	101010110	110010110	111010110
87	100010111	101010111	110010111	111010111
88	100011000	101011000	110011000	111011000
89	100011001	101011001	110011001	111011001

Options:
Internal oscillator (pins 1, 2, 3)
Lockout/rollover (pin 4), with internal resistor to V00
Lockout is logic 1

Any key down (pin 5), positive output
Pulse data ready
Internal resistor to V00 on shift & control pins

DESCRIPTION

The KR 3600 PRO is a MOS/LSI device intended to simplify the interface of a microprocessor to a keyboard matrix. Like the other KR 3600 parts, the KR 3600 PRO contains all of the logic to de-bounce and encode keyswitch closures, while providing either a 2-key or N-key rollover.

The output of the KR 3600 PRO is a simple binary code which may be converted to a standard information code by a PROM or directly by a microprocessor. This permits a user maximum flexibility of key layout with simple field programming.

The code in the KR 3600 is shown in Table I. The format is simple: output bits 9, 8, 7, 6, 5, 4 and 1 are a binary sequence. The count starts at X0, Y0 and increments through X0Y1, X0Y2...X8Y9. Bit 9 is the LSB; bit 1 is the MSB.

Bits 2 and 3 indicate the mode as follows:

Bit 2	Bit 3	
0	0	Normal
0	1	Shift
1	0	Control
1	1	Shift Control

For maximum ease of use and flexibility, an internal scanning oscillator is used, with pin selection of N-key lockout (also known as 2-key rollover) and N-key rollover. An "any-key-down" output is provided for such uses as repeat oscillator keying.

Figure 1 shows a PROM-encoded 64 key, 4 mode application, using a 256x8 PROM, and Figure 2 a full90 key, 4 mode application, utilizing a 512x8 PROM.

If N-key rollover operation is desired, it is recommended that a diode be inserted in series with each switch as shown. This prevents "phantom" key closures from resulting if three or more keys are depressed simultaneously.

FIGURE 1
KR 3600 PRO TYPICAL APPLICATION
64 KEY, 4 MODE

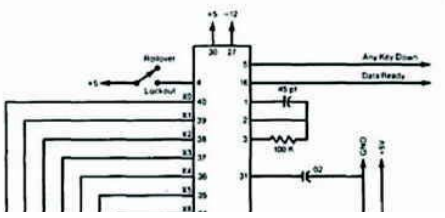
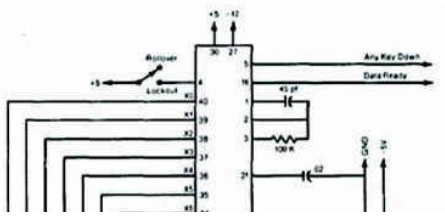
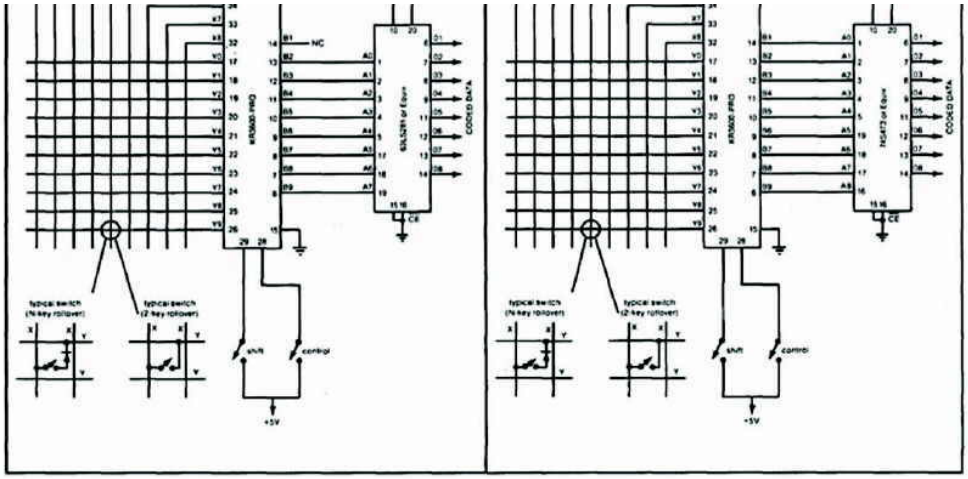


FIGURE 2
KR 3600 PRO TYPICAL APPLICATION
90 KEY, 4 MODE





the KR3600 decoderchip datasheet

And here is a basic application note about keyboard decoderchips for those who demand for more information to the topic.....

AN-128

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.

10

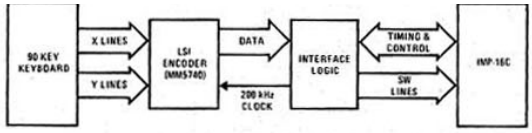


FIGURE 1. Functional Diagram

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

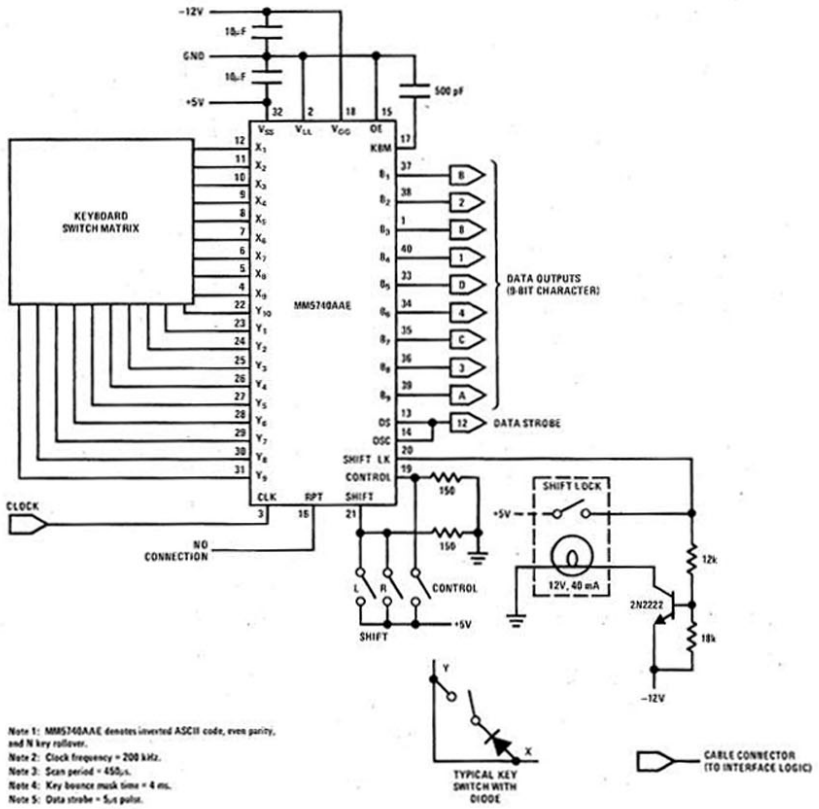


FIGURE 2. MM5740 Pin Connections

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

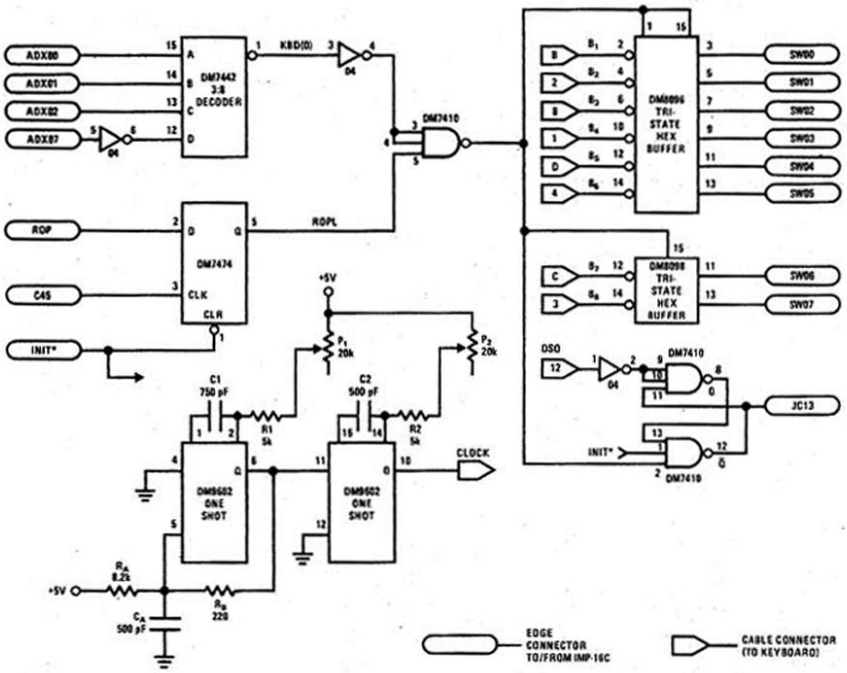


FIGURE 3. Text Editing Keyboard (TEK) Interface Logic for IMP-16C

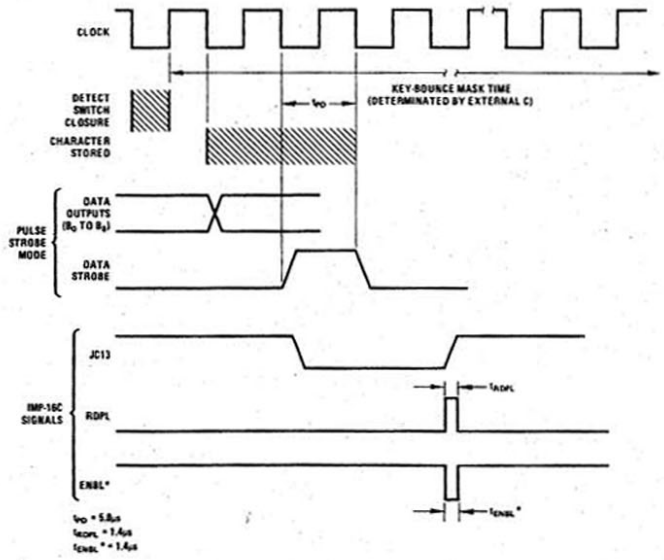


FIGURE 4. MM5740/IMP-C Timing Diagram

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.

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.

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

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.

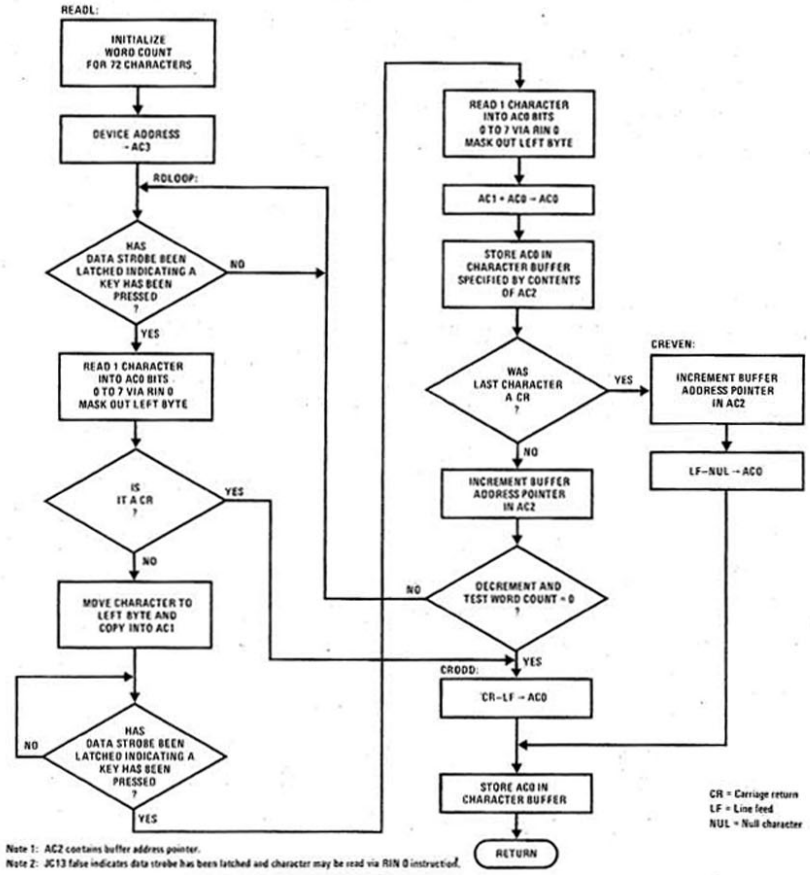


FIGURE 5. Flowchart of Subroutine (READL) that Reads One Line from the Keyboard

AN-128

```
 1                          .TITLE TEK
 2          0000            .ASECT
 3          0700            .=X'700
 4                    ; MAIN PROGRAM
 5  0700 8914 A     TEK:    LD      2,STADDR        ;INITIALIZE MESG ADDR
 6  0701 A90C A     GO:     ST      2,MADRES
 7  0702 2914 A             JSR     READL           ;READ 1 LINE & STORE
 8                    ; PUT 1 IN AC0 FOR TTY LINE, 0 TO CONTINUE READING,
 9                    ; 2 TO OUTPUT ALL LINES ON TTY
10  0703 0000 A             HALT                    ;ENTER 0/1/2 IN AC0
11  0704 1305 A             BOC     3,OUTL          ;BIT0 AC0=1 OUT LINE
12  0705 1402 A             BOC     4,OUTM          ;BIT1 AC0=1 OUT MESAG
13                    ;   CONTINUE ENTERING NEXT LINE BY DEFAULT
14  0706 4A01 A             AISZ    2,1             ;INCR ADDRESS PTR
15  0707 21F9 A             JMP     GO              ;CONTINUE
16                    ;   OUTPUT ENTIRE MESSAGE ON TTY
17  0708 850C A     OUTM:   LD      1,STADDR        ;SETUP MESG STARTING
18  0709 A504 A             ST      1,MADRES        ;    ADDRESS
19                    ; ENTER 0 AS LAST WORD FOR MESG ROUTINE IN TTY16P
20                    ;   OUTPUT LINE OR MESSAGE
21  070A 4A01 A     OUTL:   AISZ    2,1             ;INCR ADDRESS
22  070B 4C00 A             LI      0,0
23  070C A200 A             ST      0,(2)
24                    ; OUTPUT ON TTY USING MESG SR IN TTY16P PROM
25  070D 2D08 A             JSR     @MESG           ;OUTPUT ON TTY
26          070F    MADRES: .=.+1                   ;MESSAGE ADDRESS
27  070F 21F1 A             JMP     GO              ;READ NEXT LINE
28                    ;
29                    ; DATA AREA
30          0711    WDCNT:  .=.+1                   ;WORD COUNT FOR KBD
31  0711 00FF A     H00FF:  .WORD   X'00FF          ;MASK RT WD
32  0712 008D A     CR:     .WORD   X'008D          ;CR W PARITY BIT
33  0713 0D0A A     CRLF:   .WORD   X'0D0A          ;'CR-LF'
```

```
34 0714 0A00 A  LFNULL: .WORD   X'0A00          ;'LF-NUL'
35 0715 1000 A  STADDR: .WORD   X'1000          ;ST ADDRESS OF MESG
36 0716 7EC3 A  MESG:   .WORD   X'7EC3          ;MESG SR ADDR TTY16P
37              ; READ 1 LINE FROM KEYBOARD & STORE IN 36 WD BUFFER
38              ; STARTING BUFFER ADDRESS IN AC2
39              ; CHARS ARE READ, PACKED & STORED
40              ; CR IS TERMINAING CHAR.  CR LF AT END OF LINE
41              ;  JC13 FOR DATA STROBE OUTPUT WHEN KEY IS PRESSED
42 0717 4C24 A  READL:  LI      0,36            ;WORD COUNT
43 0718 A1F7 A          ST      0,WDCNT
44 0719 4F80 A          LI      3,X'80          ;DEVICE ADDRESS
45 071A 1DFF A  RDLOOP: BOC     13,.+0          ;WAIT FOR DATA STROBE
46 071B 0400 A          RIN     0               ;READ 1 CHAR INTO AC0
47 071C 61F4 A          AND     0,H00FF         ;MASK OUT LEFT BYTE
48 071D F1F4 A          SKNE    0,CR            ;IS IT A 'CR'
49 071E 210D A          JMP     CRODD
50 071F 5C08 A          SHL     0,8             ;MOVE TO LEFT BYTE
51 0720 3181 A          RCPY    0,1
52 0721 1DFF A          BOC     13,.+0          ;WAIT FOR DATA STROBE
53 0722 0400 A          RIN     0               ;READ 1 CHAR INTO AC0
54 0723 61ED A          AND     0,H00FF         ;MASK OUT LEFT BYTE
55 0724 3400 A          RADD    1,0             ;2 PACKED CHARS
56 0725 A200 A          ST      0,(2)           ;STORE IN BUFFER
57 0726 61EA A          AND     0,H00FF         ;WAS LAST CHAR A CR
58 0727 F1EA A          SKNE    0,CR
59 0728 2106 A          JMP     CREVEN
60 0729 4A01 A          AISZ    2,1             ;INCR ADDR POINTER
61 072A 7DE5 A          DSZ     WDCNT           ;DECR & TEST WD COUNT
62 072B 21EE A          JMP     RDLOOP
63              ;.....................................................
64              ; ENTER CR-LF AS LAST WORD
```

FIGURE 6. Coding for Text Editing Keyboard (TEK)

10

AN-128

```
65 072C 81E6 A  CRODD:  LD      0,CRLF          ;CR/LINE FEED CHARS
66 072D A200 A          ST      0,(2)           ;STORE IN BUFFER
67 072E 0200 A          RTS     0
68              ; ENTER LF-NUL AS LAST WORD
69 072F 4A01 A  CREVEN: AISZ    2,1              ;INCR ADDRESS PTR
70 0730 81E3 A          LD      0,LFNULL        ;LINE FEED/NULL CHARS
71 0731 21FB A          JMP     CRODD+1
72              ;
73              ;  MESSAGE BUFFER
74              ;  EACH LINE CONTAINS A MAXIMUM OF 72 PACKED CHARS
75              ;  AND A CR-LF
76      1000             =X'1000
77      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.

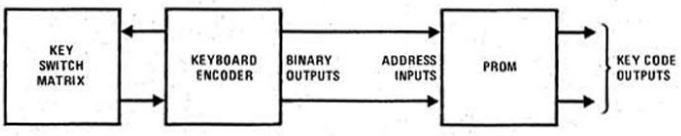


FIGURE 1. Simplified Block Diagram

*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

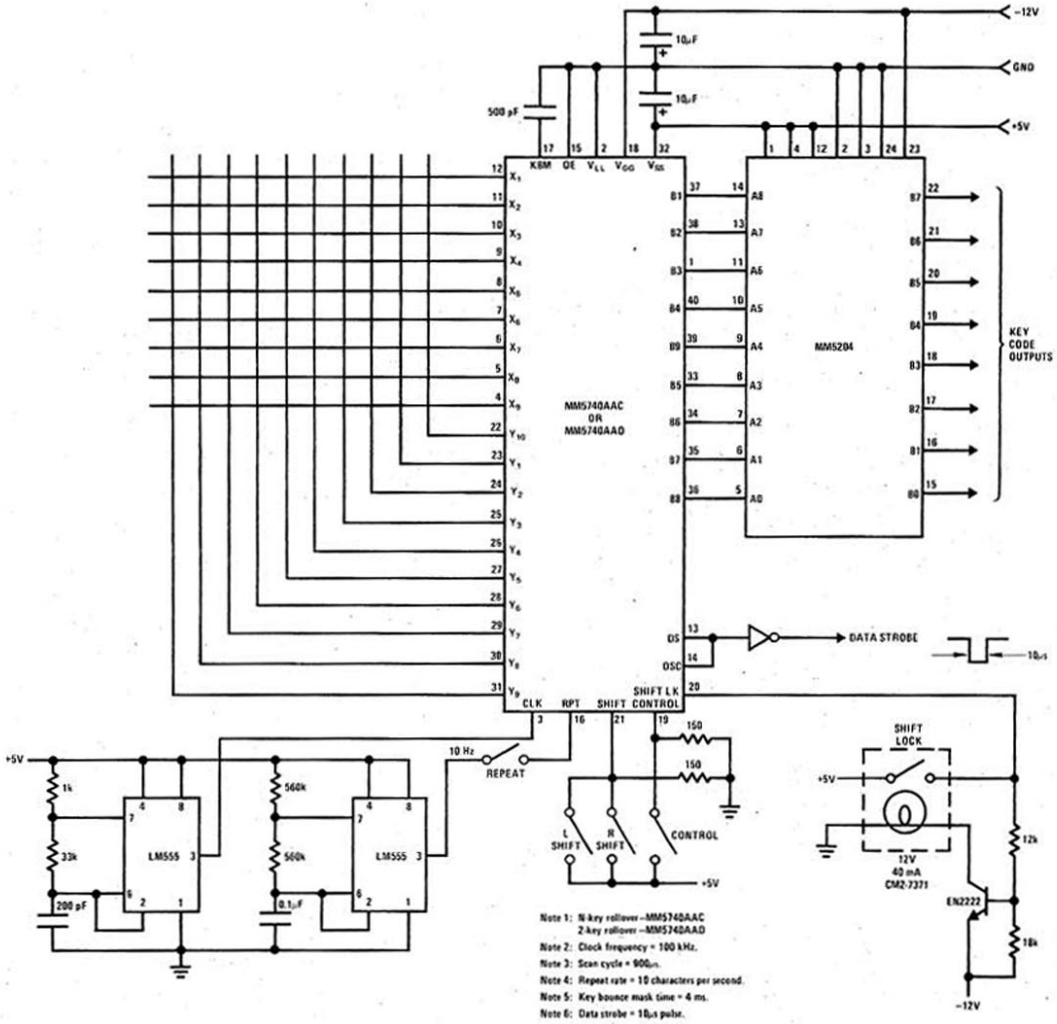


FIGURE 2. Typical Keyboard System

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 mode

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.

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

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								
	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	USER DEFINED KEY CODES							
	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.

10

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	0	1	1	1	0	1	1	1	0	1	1	0	1	0	0	1	1	0	0
5	3	1	0	0	1	1	0	1	1	1	1	0	1	1	0	1	0	0	1	0	0	0
5	4	1	0	0	1	1	0	0	1	1	1	0	0	1	0	1	0	0	1	0	0	0
5	5	1	0	1	0	0	1	1	1	1	1	1	1	1	0	1	0	0	1	1	0	0
5	6	1	0	1	0	0	1	0	1	1	1	1	1	0	0	1	0	0	1	1	0	0
5	7	1	0	1	0	0	0	1	1	1	1	0	1	1	0	1	0	0	1	0	0	0
5	8	1	0	1	0	0	0	0	1	1	1	0	0	1	0	1	0	0	1	0	0	0
5	9	1	0	0	1	1	1	1	1	1	1	1	1	1	0	1	0	0	1	1	0	0
5	10	1	0	0	1	1	1	0	1	1	1	1	0	1	0	1	0	0	1	1	0	0
6	1	1	0	0	1	1	1	0	1	1	1	0	1	1	0	1	0	0	1	0	0	0
6	2	1	0	0	1	1	0	0	1	1	1	0	0	1	0	1	0	0	1	0	0	0
6	3	1	0	0	1	1	0	1	1	1	1	0	1	1	0	1	0	0	1	1	0	0
6	4	1	0	0	1	1	0	0	1	1	1	0	0	1	0	1	0	0	1	1	0	0
6	5	1	0	0	1	1	0	0	1	1	1	0	1	1	0	1	0	0	1	0	0	0
6	6	1	0	0	1	1	0	0	1	1	1	0	0	1	0	1	0	0	1	0	0	0
6	7	1	0	0	0	0	1	1	1	1	1	1	1	1	0	1	0	0	1	1	0	0
6	8	1	0	0	0	0	1	0	1	1	1	1	0	1	0	1	0	0	1	1	0	0
6	9	1	0	0	0	0	1	0	1	1	1	0	1	1	0	1	0	0	1	0	0	0
6	10	1	0	0	0	0	1	0	1	1	1	0	0	1	0	1	0	0	1	0	0	0
7	1	1	0	0	0	0	0	1	1	1	1	1	1	1	0	1	0	0	1	1	0	0
7	2	1	0	0	0	0	0	0	1	1	1	1	0	1	0	1	0	0	1	1	0	0
7	3	1	0	0	0	0	0	1	1	1	1	0	1	1	0	1	0	0	1	0	0	0
7	4	1	0	0	0	0	0	0	1	1	1	0	0	1	0	1	0	0	1	0	0	0
7	5	0	1	1	1	1	1	1	1	1	1	1	1	1	0	1	0	0	1	1	0	0
7	6	0	1	1	1	1	1	0	1	1	1	1	0	1	0	1	0	0	1	1	0	0
7	7	0	1	1	1	1	0	1	1	1	1	0	1	1	0	1	0	0	1	0	0	0
7	8	0	1	1	1	1	0	0	1	1	1	0	0	1	0	1	0	0	1	0	0	0
7	9	0	1	1	1	1	0	1	1	1	1	1	1	1	0	1	0	0	1	1	0	0
7	10	0	1	1	1	1	0	0	1	1	1	1	0	1	0	1	0	0	1	1	0	0
8	1	0	1	1	1	0	0	1	1	1	1	0	1	1	0	1	0	0	1	0	0	0
8	2	0	1	1	1	0	0	0	1	1	1	0	0	1	0	1	0	0	1	0	0	0
8	3	0	1	1	0	1	1	1	1	1	1	1	1	1	0	1	0	0	1	1	0	0
8	4	0	1	1	0	1	1	0	1	1	1	1	0	1	0	1	0	0	1	1	0	0
8	5	0	1	1	0	1	0	1	1	1	1	0	1	1	0	1	0	0	1	0	0	0
8	6	0	1	1	0	1	0	0	1	1	1	0	0	1	0	1	0	0	1	0	0	0
8	7	0	1	1	0	0	1	1	1	1	1	1	1	1	0	1	0	0	1	1	0	0
8	8	0	1	1	0	0	1	0	1	1	1	1	0	1	0	1	0	0	1	1	0	0
8	9	0	1	1	0	0	0	1	1	1	1	0	1	1	0	1	0	0	1	0	0	0
8	10	0	1	1	0	0	0	0	1	1	1	0	0	1	0	1	0	0	1	0	0	0
9	1	0	1	0	1	1	1	1	1	1	1	1	1	1	0	1	0	0	1	1	0	0
9	2	0	1	0	1	1	1	0	1	1	1	1	0	1	0	1	0	0	1	1	0	0
9	3	0	1	0	1	1	0	1	1	1	1	0	1	1	0	1	0	0	1	0	0	0
9	4	0	1	0	1	1	0	0	1	1	1	0	0	1	0	1	0	0	1	0	0	0
9	5	0	1	0	1	0	1	1	1	1	1	1	1	1	0	1	0	0	1	1	0	0
9	6	0	1	0	1	0	1	0	1	1	1	1	0	1	0	1	0	0	1	1	0	0
9	7	0	1	0	1	0	0	1	1	1	1	0	1	1	0	1	0	0	1	0	0	0
9	8	0	1	0	1	0	0	0	1	1	1	0	0	1	0	1	0	0	1	0	0	0
9	9	0	1	0	0	1	1	1	1	1	1	1	1	1	0	1	0	0	1	1	0	0
9	10	0	1	0	0	1	1	0	1	1	1	1	0	1	0	1	0	0	1	1	0	0

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

10-30

Application Notes about keyboard decoder chips

some other keyboard encoderchips will be added soon.....

back to the keyswitch and keyboardpage Part 1

back to the downloadpage