Keyboard Encoder Circuits

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

MM5740 90-key keyboard encoder

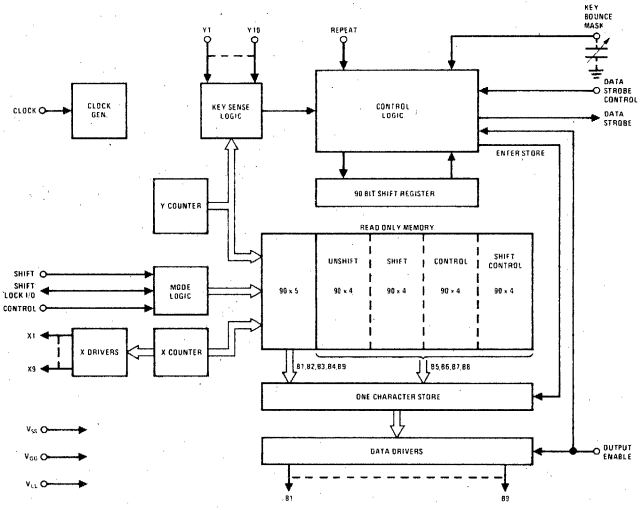
general description

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.

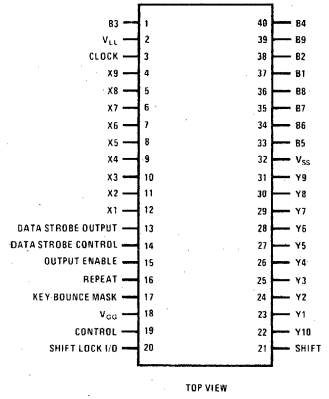
features

- 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



TOP VIEW

Order Number MM5740N
See Package 24

TRI-STATE is a registered trademark of National Semiconductor Corp.

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				3.25	V
Logic Level "0"					V
Logic Level "1"		+0.4			
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 $V_{SS} - V_{LL} \leq 5.25V$. When driving MOS, $V_{SS} - V_{LL} \leq 10.0V$.

Note 3: Trpw min. = $100 \times \frac{1}{f\ clock}$

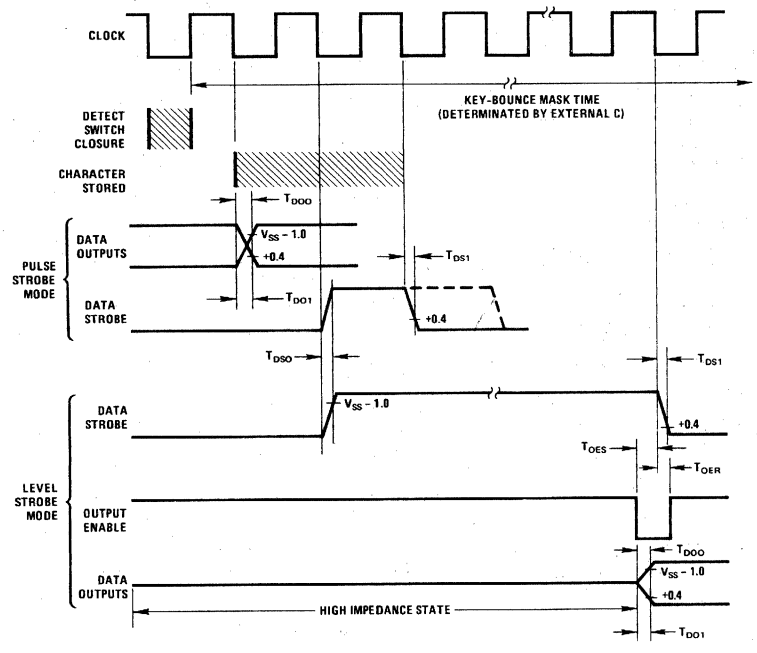
Note 4: If shift and control inputs are derived from a single pole, single throw switch closure to V_{SS}, a 100 OHM resistor returned to V_{LL} (GND) is required on these inputs.

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

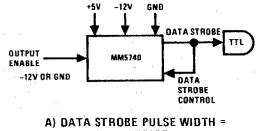
description of pin functions

NAME	PIN NO.	FUNCTION
X1-X9	4-12	These pins are chip outputs which are used to drive the key switch matrix. When activated (at the appropriate scan time) they are driven high.
Y1-Y10	22-31	Pins 22-31 are the Y sense inputs which are connected to the X drive lines via the key switch matrix. They are internally precharged to a low state and are pulled high upon switch closure.
B1-B9	1, 33-40	These are the data outputs which represent the code for each keyswitch. They are TRI-STATE outputs with direct TTL compatibility. When the output enable input (Pin 15) is high, these outputs are in the third state.
Data Strobe Output	13	The function of this pin is to indicate that valid data has been entered by the keyboard and is ready for acceptance. An active data strobe is indicated by a high level. The data strobe may be operated in the pulse or level mode as indicated by the timing diagram.
Data Strobe Control	14	The basic purpose of this input is to provide data strobe output pulse width control. When connected to the data strobe output (Pin 13), the data strobe will exhibit a one bit wide pulse width. The pulse width may be varied by interposing an RC network between the data strobe output and the strobe control input. For level mode of operation the data strobe control input may be tied to V_{SS} or to the data strobe output.
Output Enable	15	This input serves to TRI-STATE the data output (B1-B9) lines. In addition, it controls the return of the data strobe to the idle condition (low state) which is needed in the level strobe mode of operation.
Repeat	16	The repeat input is designed to accept a repeat signal via the repeat key. One data strobe will be issued for each positive interval of the repeat signal. Thus, if a 10 Hz signal is applied to the repeat input via the repeat switch, a 10 character per second data strobe will be issued when a data key and the repeat key are held depressed.
Key-Bounce Mask	17	This pin is intended as a timing node to mask switch key-bounce. The mask time interval is generated by connecting a capacitor to this pin.
Shift	21	When this input is brought to a logic "0" (V_{SS}) level, the encoder will assume the shifted character mode.
Control	19	A logic "0" places the encoder in the control character mode.
Shift Lock I/O	20	This pin is intended to serve as an input when the shift lock key is depressed. It places the encoder in the shift mode. Upon release of the key, the shift mode will be maintained and this pin will serve as an output to drive an indicator. This function is reset by depressing the shift key.
Clock	3	A TTL compatible clock signal is applied to this pin. A bit time is defined as the time from one negative going transition to the succeeding negative going transition of the clock.
V_{SS}	32	+5.0V supply
V_{LL}	2	Ground
V_{GG}	18	-12V supply

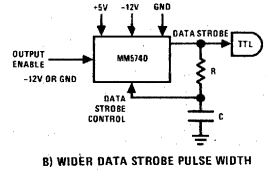
timing diagram



applications information

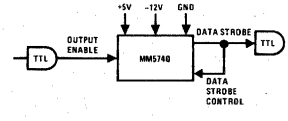


A) DATA STROBE PULSE WIDTH = ONE CLOCK PERIOD



B) WIDER DATA STROBE PULSE WIDTH CONTROLLED BY RC

Pulse Data Strobe Mode



Level Data Strobe Mode

key bounce capacitor values

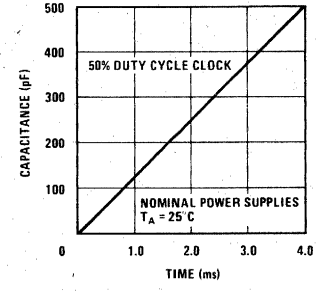


FIGURE 1. Key-Bounce Mask Time

application

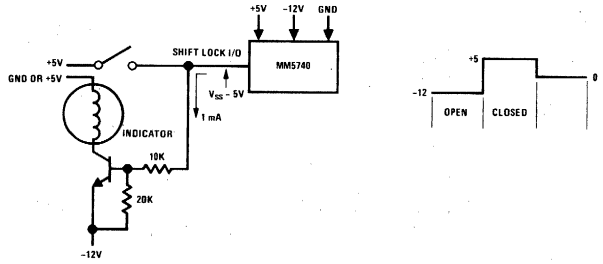
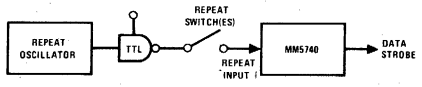
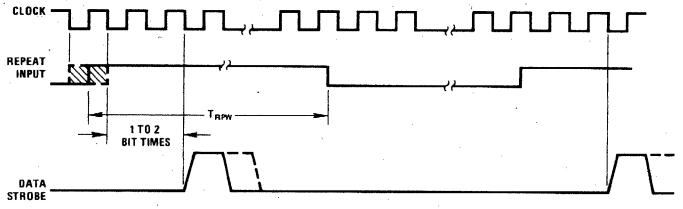


FIGURE 2. Shift Logic I/O Interface

repeat switch function



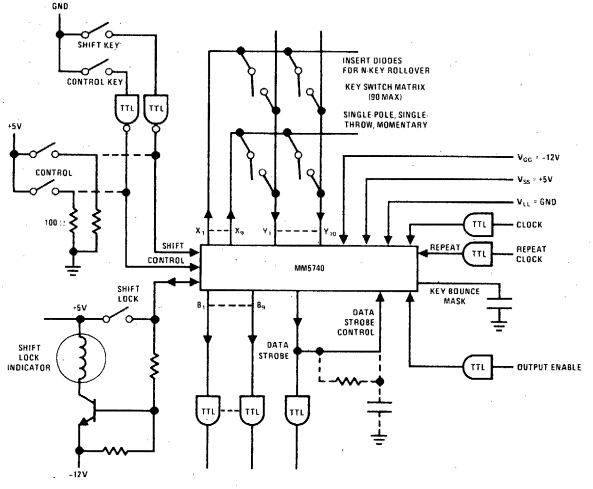
Repeat Switch Connections



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



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.

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	9	9	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	☙	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	I	!	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	O		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		FF	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	$	4	$

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.

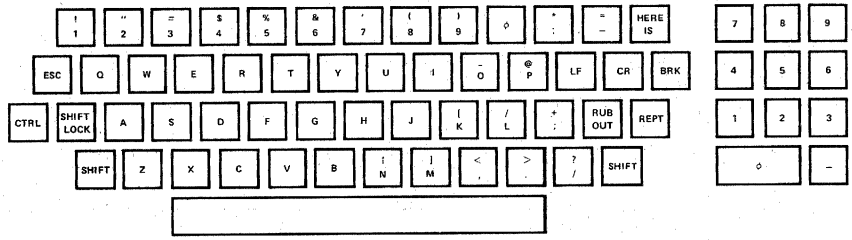
MM5740AAE, MM5740AAF CODE ASSIGNMENT CHARTS (CONTINUED)

MATRIX ADDRESS		COMMON					UNSHIFT				SHIFT				CONTROL				SHIFT CONTROL				CHARACTER			
X	Y	B_1	B_2	B_3	B_4	B_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
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	2	"	2	"
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

Keyboard Encoder Circuits

MICROPROCESSOR MATES WITH MOS/LSI KEYBOARD ENCODER

ABSTRACT

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

INTRODUCTION

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

INTERFACE CONSIDERATIONS

The Keyboard

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

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

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

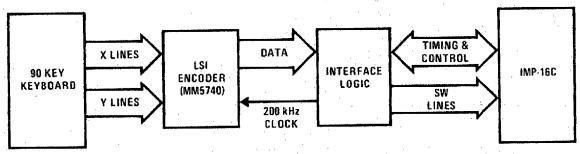
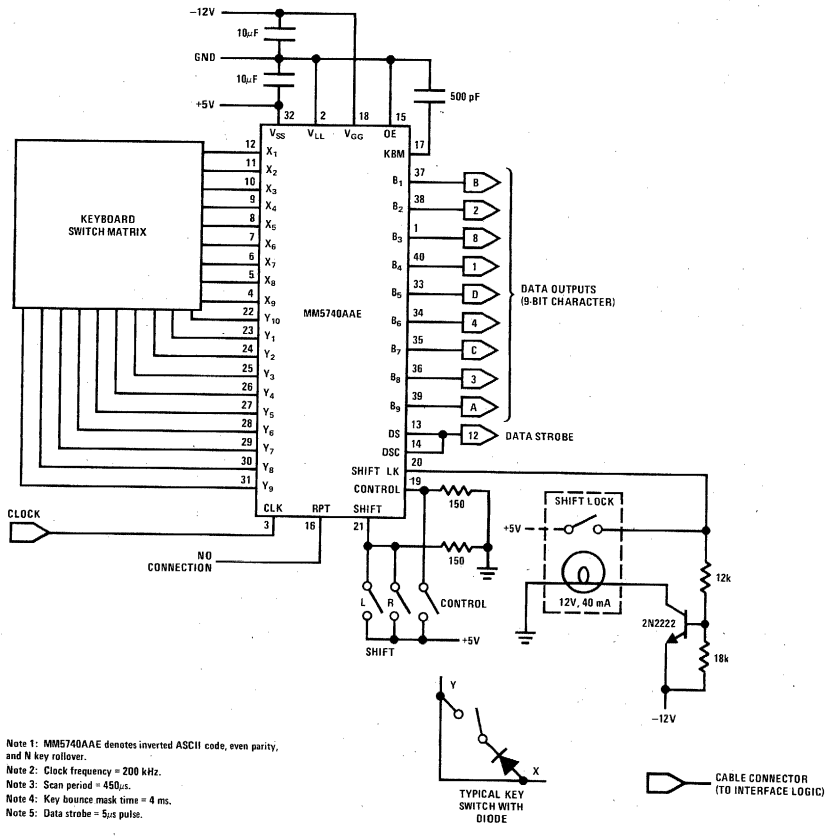


FIGURE 1. Functional Diagram

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



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

FIGURE 2. MM5740 Pin Connections

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

MM5740—IMP-16C INTERFACE

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

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

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

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

Execution of the RIN instruction causes:

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

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

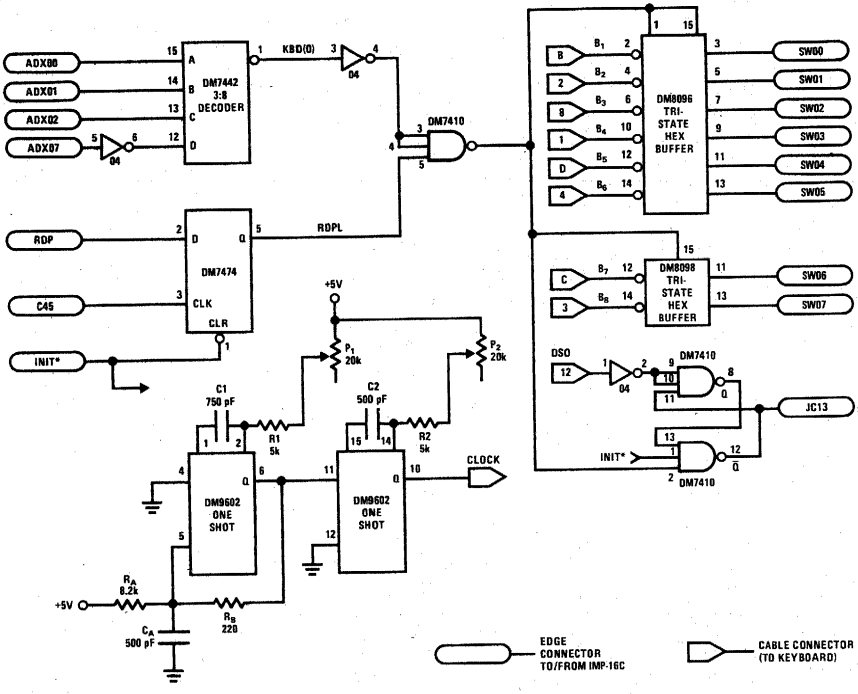


FIGURE 3. Text Editing Keyboard (TEK) Interface Logic for IMP-16C

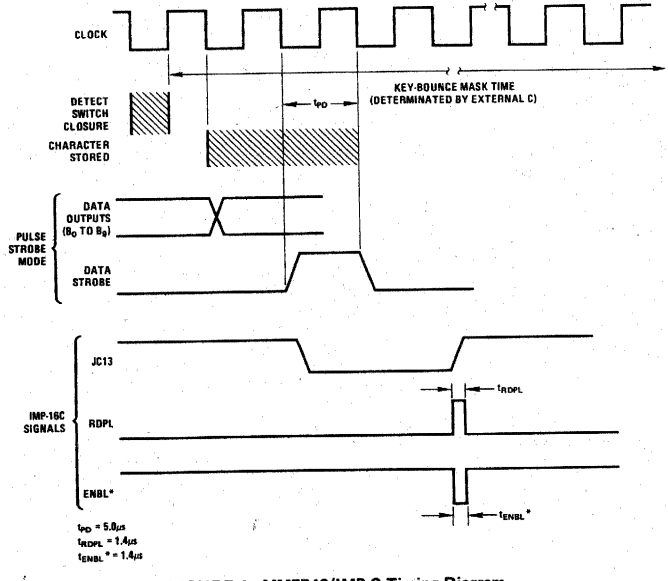


FIGURE 4. MM5740/IMP-C Timing Diagram

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

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

THE PROGRAM

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

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

CONCLUSION

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

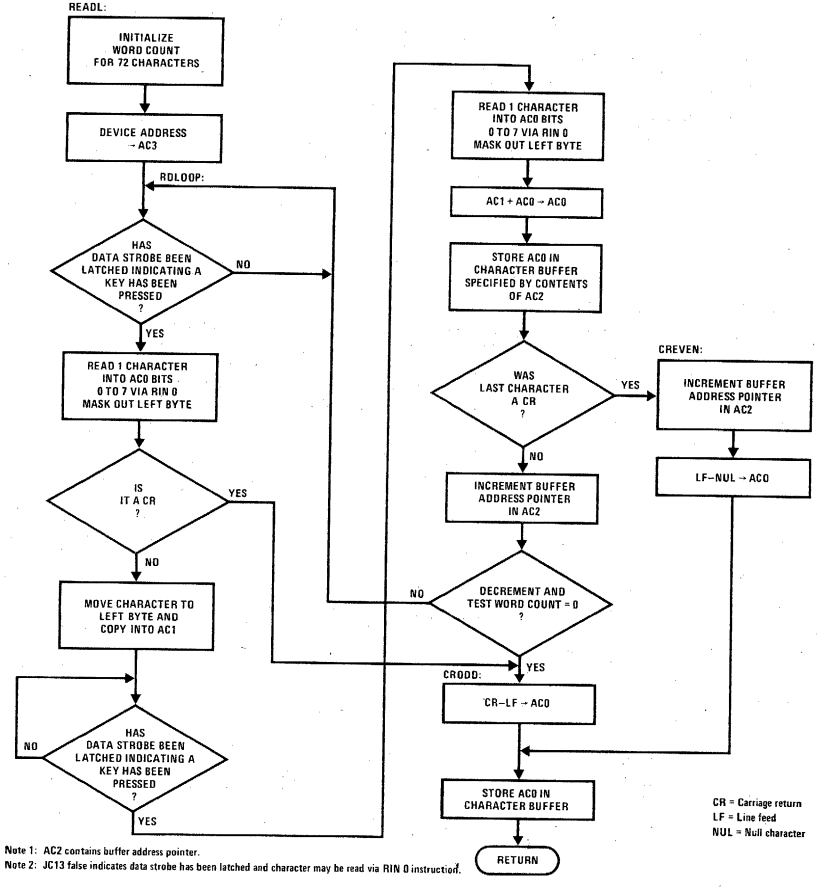


FIGURE 5. Flowchart of Subroutine (READL) that Reads One Line from the Keyboard

```
 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)

```
 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.

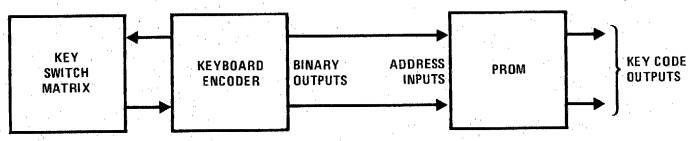


FIGURE 1. Simplified Block Diagram

*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

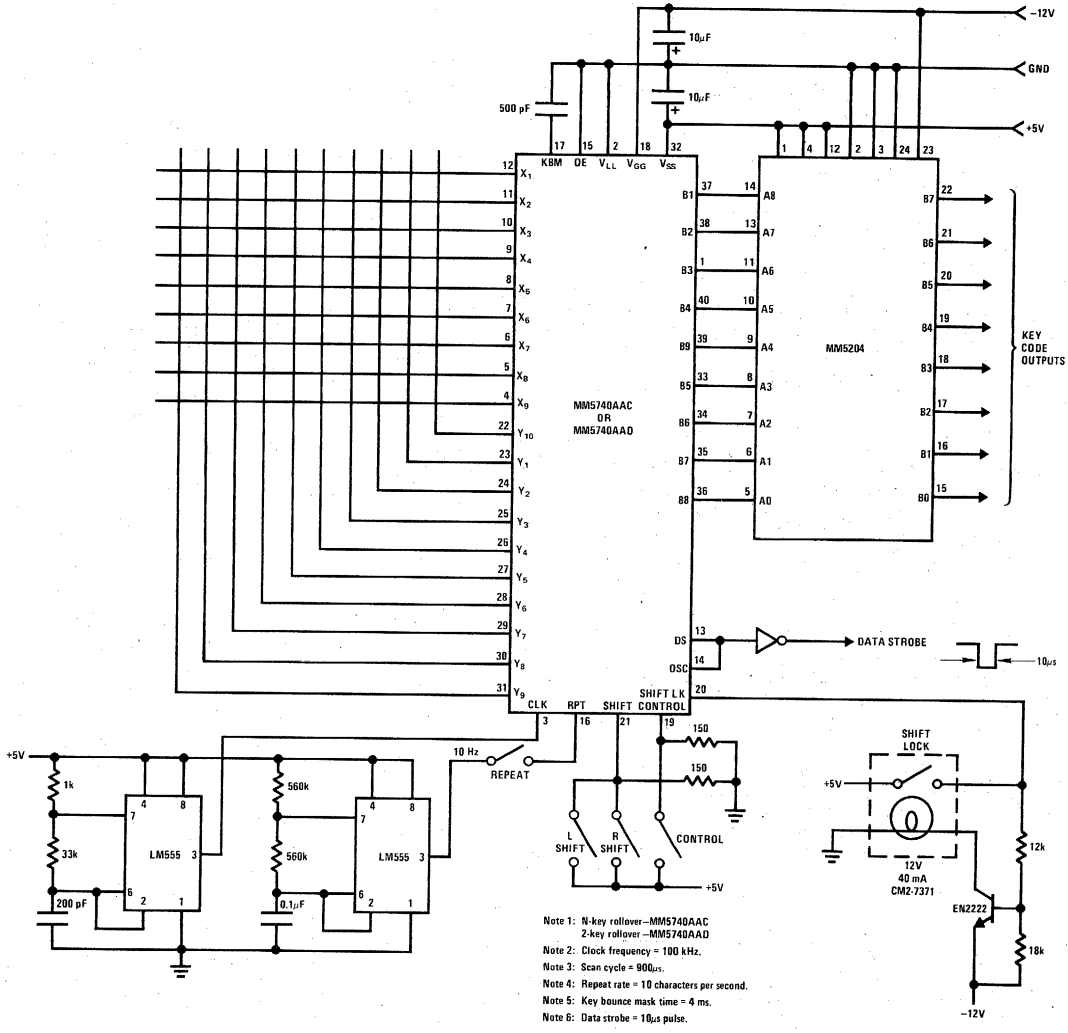


FIGURE 2. Typical Keyboard System

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 modes, 360 nine-bit codes are possible.

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

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

TABLE I. Encoder/PROM Mapping

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

*Encoder outputs are listed in positive true logic notation.

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

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

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