MOS KEYBOARD ENCODING

INTRODUCTION

The use of MOS large scale integration has made it possible to provide, in a single integrated circuit, all of the components necessary for economic implementation of the keyboard encoding function. The MM5740 is a complete keyboard interface system capable of providing quad mode 90 key keyboard encoding. In addition to the basic problem of translating a switch closure to a coded output, the MM5740 provides all of the functions necessary for modern keyboard system design. The salient features include programmable N-key or 2-key rollover, TRI-STATE® parallel data outputs with a one character memory and externally controllable pulse or level data strobe modes. Added conveniences are on-chip clock generation from a single TTL input, external RC control of strobe pulse width and capacitor control of key bounce mask time, single pin shift lock/shift indicator capability and repeat key function.

THEORY OF OPERATION

In order to better understand the following discussion, reference should be made to Figure 1 which shows a functional block diagram of the MM5740 as well as the pin connection diagram.

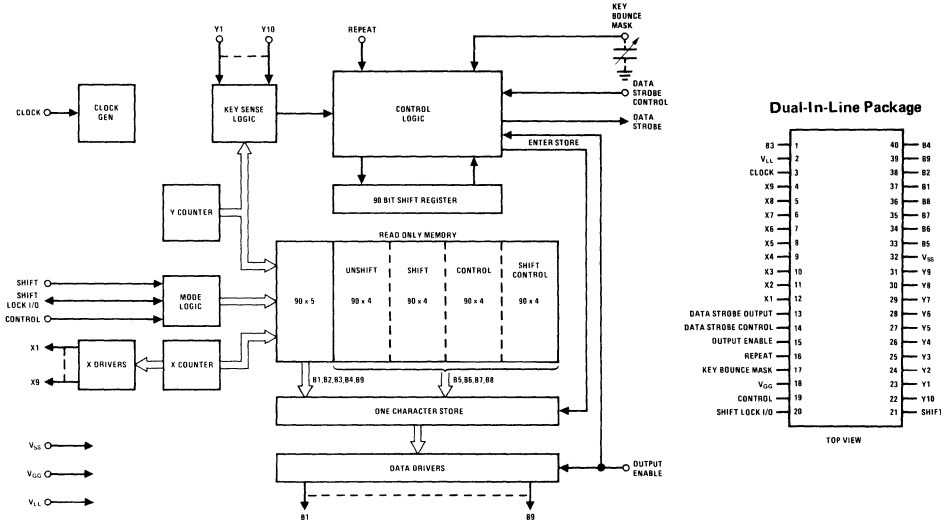


FIGURE 1. Block and Connection Diagrams

A dynamic scanning technique is utilized to detect and pinpoint keyswitch activity on the keyboard. In this method, two ring counters are employed. A 9-bit X counter monitors the rows of the switch matrix while the columns are monitored by a 10-bit Y counter. Since there are 9 x 10 = 90 possible counter states, ninety data switches may be defined by their X-Y coordinates.

Every scan time (90-bit times), all keys are interrogated in turn to determine their status. Each particular key is interrogated every 90-bits or once each keyboard scan cycle. This is implemented by employing the X counter as drivers to drive each row of the switch matrix in turn. The Y counter enables the sensing of each column of the switch matrix. Therefore, if a switch is closed it will be detected when its row (X coordinate) is driven and its column (Y coordinate) is sensed. This corresponds to the particular X-Y time (one of 90 time slots in the scan) defining the closed switch. A key detect pulse at the X-Y time in question is issued and this is used to define the status of that particular key.

The above discussion has shown us how to detect a key closure and pinpoint which key switch it was. We must now take this information and provide a coded bit pattern for that key. In addition to scanning the key switch matrix, the X and Y counters are used to scan the contents of a Read Only Memory which contains the bit pattern for each key switch. For example, when switch X_1, Y_1 is being interrogated (at X_1-Y_1 time), the counters are addressing the memory location in the ROM which contains the code for switch X_1Y_1. If the key is closed, a key detect pulse will be issued. At this time, the ROM contents are strobed into latches which then hold the code pattern for key X_1, Y_1. It should be noted that the scanning of the switch matrix and ROM occur simultaneously and repetitively. The clocks are never stopped, instead information is obtained dynamically.

For each keyswitch it is possible to have four modes. These are determined by the status of the SHIFT and CONTROL keys which are inputs to the MM5740. There are four possible code patterns for each key as determined by the mode switches. These code patterns are related in that, for any key, in any of its modes, only four of its 9-bits may vary. All code patterns are programmed at the time of manufacture using a single mask variation.

Rollover

The MM5740 keyboard encoder chip is capable of either N-key or 2-key rollover. The customers' preference is programmed on the same mask used to define the code pattern for the keyswitches. N-key rollover is defined as the ability to issue a code each time a new key is depressed regardless of the status of all other keys. This means that each time a new key is depressed, a code for that key would be issued even if other keys are still depressed. This type of rollover is ideal for fast typing applications, since the typist does not have to train herself to release the currently depressed key before she depresses a new one. Even in slower typing applications, burst typing of typical trigrams such as "THE" and "ERE" can be a problem if N-key rollover is not employed.

In 2-key rollover, a code is issued at the time of depressing a new key, provided all other keys are released. If a key is depressed a code will be issued and the keyboard will be ignored until that switch has been released. If two depressions are performed in turn the first will be recognized and the second code will be issued when the first key is released.

N-key rollover is accomplished by remembering the status of each key on the previous scan cycle. This is achieved by using a 90-bit shift register whose input is the key detect signal. If the input and output of the shift register are monitored, it is possible to compare the status of each key on the past scan (output of shift register) with its status on the current scan (input of shift register). A valid key closure is present only if the input of the shift register is true and its output false. A release is present when the input is false and the output is true. The other two cases (00 and 11) correspond to no change in status; either released or depressed. The ROM word is strobed into the output latches only when a valid key closure is detected.

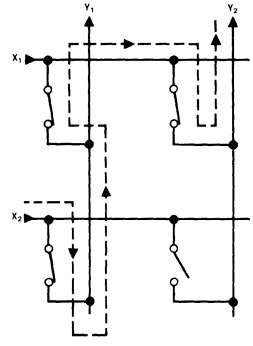
Once a previously closed key is released it then becomes possible to accept it as a valid closure if it is depressed again. For 2-key rollover, the same criterion is used to determine a valid key closure. However, after a key has been depressed no further entries will be acted upon until that key has been released. It should be also noted that the depression of the SHIFT and/or CONTROL keys after the depression of a data key will be ignored. The mode control keys must be depressed before depressing the data keys.

Masking the Bounce

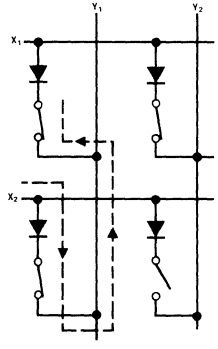
Most keyswitches, primarily mechanical or reed, will exhibit a bounce characteristic upon both depression and release. In normal operation, many keyboard scans could occur during the time of switch bounce. It is then necessary to make sure that this bounce is not recognized as multiple depressions and/or releases. As soon as the scanner detects a change in key status, an internal time delay is activated which instructs the encoder to ignore keyboard activity. At the end of the time out (usually several milliseconds), when switch bounce has subsided, the encoder resumes interrogating the switch matrix. The time delay is continuously variable with an external capacitance connected to pin 17 of the MM5740. Typical time delays are illustrated in Figure 1 of the data sheet. The continuously variable bounce masking allows the MM5740 to be used with a wide range of commercial key switches.

The Phantom Key

In keyboard system design, a frequent question arises about the use of diodes in the keyswitch matrix. The need for diodes in series with each keyswitch stems from the fact that, under certain conditions, an undepressed key will appear as one which is depressed. Consequently an undesired additional character will be transmitted. Since an open keyswitch was detected, this keyswitch is often termed the phantom key. In order to better understand this problem, reference should be made to Figure 2. In this diagram, phantom switch X_2Y_2 will be detected since there exists a closed path from the X_2 drive line to the Y_2 sense line which does not pass through the open switch X_2Y_2. By placing diodes in series with each keyswitch, this problem is eliminated since the sneak path has been blocked.



(a) NO DIODES



(b) DIODES BLOCK SNEAK PATH

FIGURE 2. Phantom Key Illustration

For a phantom key situation to occur it is necessary that at least 3 switches be closed and that they are located at the corners of some matrix rectangle. In addition, if a 2-key rollover encoder is used no diodes are required since the keyboard will be ignored if one key remains depressed.

For an N-key rollover keyboard, if it can be certain that no more than 2 keys will remain closed, no diodes are required. If this condition cannot be guaranteed it is possible, by clever key station layout, that when 3 keys are down they do not form the corners of a matrix rectangle. When none of the above restrictions can be met and true N-key rollover is required, it becomes necessary to insert diodes in series with the key switches. For uniformity it is recommended that a diode be used with each switch. However, diodes are not required along the diagonal coordinates of the switch matrix and may be omitted if desired. It should be noted that this discussion pertains to switches which form a contact between the matrix coordinate lines. These include physical contact types such as mechanical or reed. For solid state or capacitive switches where there is isolation between an X-Y coordinate, this problem may not apply.

Input-Output Interfacing

In order to better see how the MM5740 can be connected to external components, an understanding of its input/output capabilities is necessary. All functional control inputs as well as the Data output lines (B_1-B_9) and the Data Strobe output are directly TTL compatible. The functional control inputs are clock, output enable, repeat, shift and control. These input pins may be driven from a standard TTL logic gate which also drives 10 other TTL loads. Thus the worst case input levels are specified at +0.4 and +2.4V. The code data output and data strobe lines can directly drive one standard TTL load and 100 pF at 200 kHz. If the clock frequency is reduced the capacitive load may be increased proportionally. The matrix drive (X_1-X_9) and sense (Y_1-Y_{10}) lines are normally connected to each other via the switch matrix. Typical waveforms for these signals are shown in Figure 3.

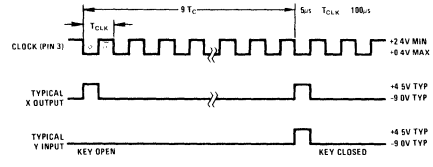


FIGURE 3. Matrix Signal Waveforms

Both the X and Y lines are unconditionally precharged toward the V_{GG} supply (-12V) when the clock is high. At the appropriate bit times of the scan cycle, each X line will go high during the time that the clock is low. For an open switch the Y line will remain negative. When a switch is closed the X line pulse will be transmitted to the Y input causing it to go high. The Y line will re-establish itself to the idle (negative level) condition during the next high level of the clock.

The X lines are capable of driving 75 picofarads at a 200 kHz clock rate. If the clock rate is reduced the load capacitance may be increased proportionally. Figure 4 is a schematic of the X output and Y input circuit of the encoder as it is normally connected to a matrix switch. The Y input has a high impedance device (.5 - 1MΩ) to the V_{GG} supply. This device is intended to support the idle

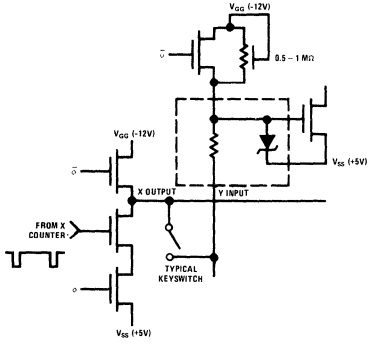


FIGURE 4. X Output – Y Input Schematic

level during low frequency operation. If capacitive or solid state switches are used it may be necessary to buffer the X and Y pins of the MM5740 from the keyswitch matrix. For this purpose reference should be made to Figure 5 which shows the typical X and Y line voltage-current characteristic. The right hand portion of the curve describes the X line active high characteristic while the left hand portion illustrates the X and Y characteristic when trying to charge toward the V_{GG} supply. These curves are useful for designing buffer circuitry driven by the X outputs or intended to drive the Y inputs of the MM5740.

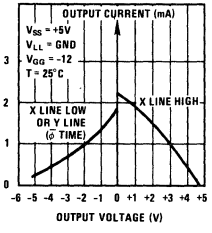


FIGURE 5. Typical X & Y Line Voltage – Current Characteristics

Shift Lock Operation

In many keyboards it is desirable to operate the keyboard in the shift mode for an indefinite period of time. This would occur, for instance, if it was desired to type a letter in upper case. A keyboard encoded with the MM5740 provides this function electronically, eliminating costly mechanically interlocked switches.

When the shift lock key is depressed, the keyboard assumes the shift mode. If the shift lock switch is then released, the keyboard remains locked in the shift mode. In addition, the shift lock pin (MM5740 pin 20) then operates as an output driver and supplies current to drive a shift lock indicator. This indicator signals the operator that the keyboard is in the shift lock mode. When it is desired to terminate this mode of operation, the shift key is momentarily depressed extinguishing the indicator light and placing the encoder in the unshifted mode. It should be noted that the shift lock keyswitch need only be a momentary single pole-single throw normally open switch. It is especially convenient to use a momentary lighted push button keyswitch for this application. Figure 6 illustrates a typical circuit diagram for implementing the shift lock function.

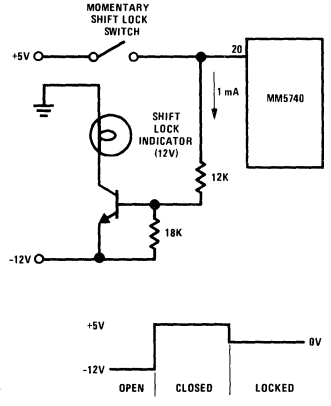


FIGURE 6. Shift Lock I/O Interface

Repeat Function

A useful feature contained in many keyboard systems is the repeat character key. By depressing this key and a data key the data character will be repeated. In addition to repeated character entry this function is useful for such things as underlining, back spacing and cursor control.

An incoming repeat clock (TTL level) at the desired rate is accepted by the MM5740. This clock is internally synchronized to the keyboard clock and is used to pulse the data strobe signal at the repeat rate. The pulse width of the data strobe is variable and its determination will be discussed shortly. This pulse occurs at the scan bit position occupied by the character key in question. Since the repeat clock is asychronous with the encoder clock, it is necessary that the repeat pulse width be at least as wide as one scan time (90 encoder clock periods). For most applications this presents no problem since the encoder clock rate will be several orders of magnitude faster than the repeat rate.

Program Unused Code Bits For Special Functions

In most cases only seven bits are required to generate the character codes. This leaves bits eight and nine available to provide other keyboard functions with a minimum of extra hardware. If parity is desired bit 8 is programmed to supply this function. For applications where special character control is required it is possible to use bits 8 and 9 to satisfy this need. For example, in a terminal application the display control keys are sometimes treated as non-transmit characters. In such situations, the ninth bit could be programmed to serve as a strobe inhibit. With this arrangement, the B_9 output (pin 39) is gated with the Data Strobe to prevent transmission for the appropriate characters.

Another frequent application occurs when it is required that only certain characters on the keyboard be allowed to repeat. This may be accomplished by programming bit 9 of the character code as a repeat inhibit bit. When gated with the repeat oscillator, a selective repeat input is supplied to the encoder via the repeat key. Since bit 9 is common in all modes, the repeat will be inhibited or enabled for every mode of the key in question. If it is desired to selectively repeat the same key in different modes, bit 8 may be used as the inhibit bit since this bit is programmable in each of the four possible keyboard modes.

In situations where it is desired to class characters in sets which are to be processed differently, some code bits may be programmed to provide a process recognition code. For example, by programming bits 8 and 9 as process recognition bits, keyboard characters may be divided into four classes for separate processing. When a character is entered, the processor or keyboard controller will examine the process code and channel the information to the appropriate processing logic or execute a predetermined routine according to character class.

It should be emphasized that any of the code bits may be programmed to provide additional features as explained above and that bits 8 and 9 are the most frequently available extra bits when using a standard seven bit code. Figure 7 illustrates a typical arrangement for implementing the transmit inhibit and selective repeat functions.

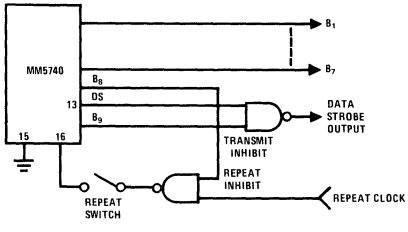


FIGURE 7. Typical Transmit Inhibit and Selective Repeat Arrangement

Data Strobe, Data Strobe Control and Output Enable

In order for the keyboard to effectively communicate with the system to which it is connected, it must supply a signal which alerts that system when data has been entered. This requirement is met by the Data Strobe output (pin 13). Once a valid character has been entered by the keyboard and is stored in the output latches of the MM5740, a Data Strobe is issued. The Data Strobe signal may take on several formats which will be discussed shortly.

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) is also an encoder input and it serves two purposes. Firstly, it serves as the TRI-STATE control for the code data output lines (B_1-B_9). Secondly, it is used to control the resetting of the Data Strobe Output.

In order to better understand the interrelationship of these functions reference should be made to Figure 8 which illustrates the pertinent logic.

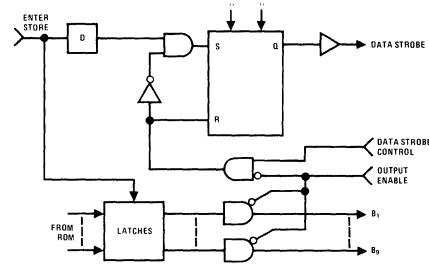


FIGURE 8. Data Strobe, Data Strobe Control and Output Enable Logic

When a valid key is entered an Enter Store signal is generated which enables the output latches to accept coded data from the ROM. The Enter Store signal, after being delayed one bit time, forms the set term of an RS flip flop. This flip flop inputs at $\overline{\phi}$ time (clock high) and outputs at ϕ time when the clock is low. The Data Strobe will go high when the flip flop is set, indicating that data is ready and stored in the output latches. The Data Strobe Control and Output Enable form the reset term for the Data Strobe flip flop. When the Data Strobe control is high and the Output Enable is low, the Data Strobe output will be reset. In addition, when the output enable is high the data output lines are placed in the high impedance state.

Now that the basic logic relationships are understood we will discuss typical applications concerning these functions. For a pulse mode data strobe, the output enable is grounded (always enabled) and 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. If a pulse width greater than 1 bit time is required,

it may easily be achieved by inserting an RC network between pin 13 and pin 14 as shown in Figure 9. The pulse width is approximately 0.6 RC. This feature eliminates the need for a one-shot circuit often found in many keyboard systems. By allowing for a resistor and capacitor on the printed circuit board, a variable pulse width can be achieved without affecting board layout. If a one bit wide strobe is desired, a wire link can replace the space taken by the resistor and the capacitor is left out.

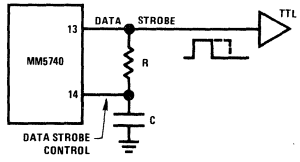


FIGURE 9. Variable Width Pulse Data Strobe

In many applications, it is more convenient to have a level strobe mode of operation. For example, in bus structured systems where there are many keyboards and/or other peripherals connected to the main processing system a controlled level strobe is desired. Such a configuration is illustrated in Figure 10. In this situation many peripherals share a TRI-STATE bus. When a peripheral such as the

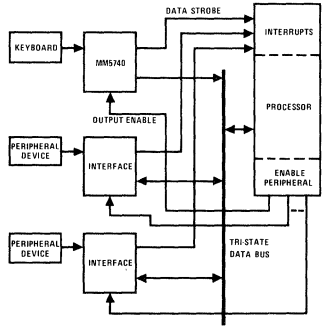


FIGURE 10. Keyboard Use in a Bus-Structured Interrupt System

keyboard has data available, the Data Strobe goes high. This signal interrupts the processor. At the appropriate time, the processor will respond to this interrupt by setting a flag. This flag can be used to enter a read peripheral mode and send an enable signal to the keyboard. When the output enable of the MM5740 goes low the Data Outputs are presented to the TRI-STATE bus for acceptance by the processor. On the next negative edge of the clock, the data strobe will be automatically reset. At this point the keyboard and processor have completed a data exchange and conditions are such that additional data may be entered. Of course the time periods of concern in such a design are dictated by the keyboard activity rate and the processor organization.

In most situations, the processor will readily accept data presented by the keyboard at normal keyboard rates. If conditions are such that the processor will be busy for periods of time longer than the time between keyboard entries, then a buffer memory will be needed to store those entries. These considerations pertain to total system design and will vary in each particular situation.

For the purpose of further illustrating the use of the data strobe, data strobe control and output enable, let us consider the particular application where it is desired to poll several keyboards in a system. We will consider that the keyboards can be either localized or at remote locations and that a scanning interrogation technique will be used.

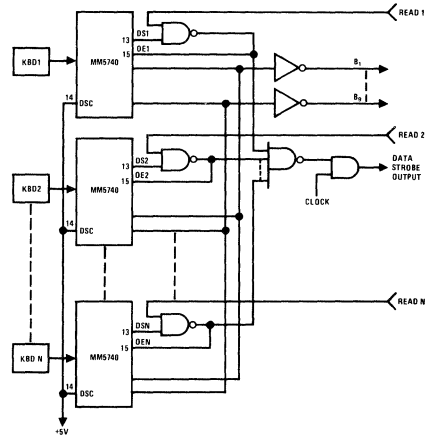


FIGURE 11. Polling System for Localized Keyboards

In Figure 11 such a polling set up is shown for localized keyboards and the associated timing diagram is shown in Figure 12. Since the keyboards are local, the wiring capacitance can be reasonable enough to make use of the TRI-STATE feature of the MM5740 and allow all encoders to share a common data bus. When a character is entered from any keyboard, the corresponding data strobe line (pin 13) will go high. When the keyboard is read, the output enable line will go low, taking the data lines out of the high impedance state. One bit time later, the data strobe will be automatically reset which in turn will drive the output enable high again. This removes that keyboard from the data bus and allows the remaining keyboards to be interrogated. The data strobes for each keyboard are combined and gated with the clock to form a single data strobe output to the receiving system. When the receiving system sees the data strobe pulse, the output data is stable and ready for acceptance.

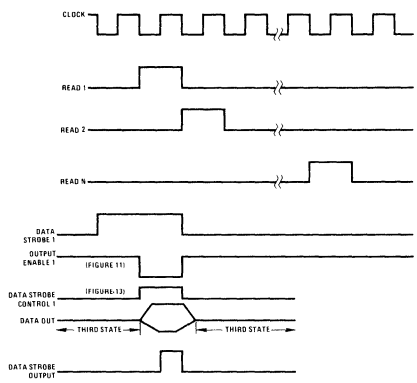


FIGURE 12. Timing Diagram for Keyboard Polling System

When this application applies to remote keyboards or a heavily loaded TTL bus, the keyboard encoders must be buffered. This may be accomplished as shown in Figure 13 by using TTL TRI-STATE devices between the encoders and the common bus. In this technique, the keyboard encoders are always enabled by grounding pin 15. When the data strobe goes high and the keyboard is interrogated, the data strobe control line will go high. This takes the data output gates out of the high impedance state. On the next positive clock the data will be stable and ready for acceptance by the receiving system. In addition, the fact that the data strobe control went high will reset the data strobe on the next negative edge of the clock.

Use Two Encoders for 180 Key Capability

Some special applications may require an N-key rollover encoder capable of handling a keyboard with greater than 90 keyswitches. Such situations are easily handled by using two MM5740 encoders to provide a capability of up to 180 keyswitches. Figure 14 illustrates how such a system is configured and Figure 15 shows typical timing waveforms. It should be noted that such a configuration is achieved with a minimum of peripheral circuitry.

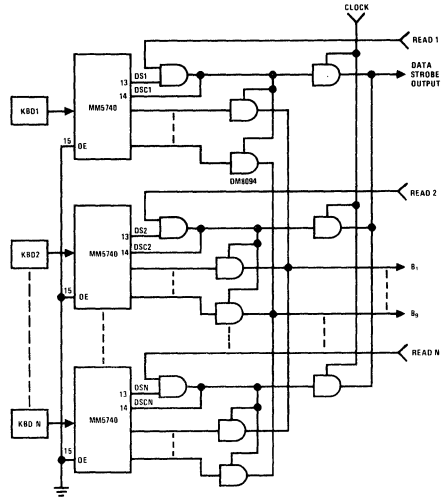


FIGURE 13. Polling System for Remote Keyboards

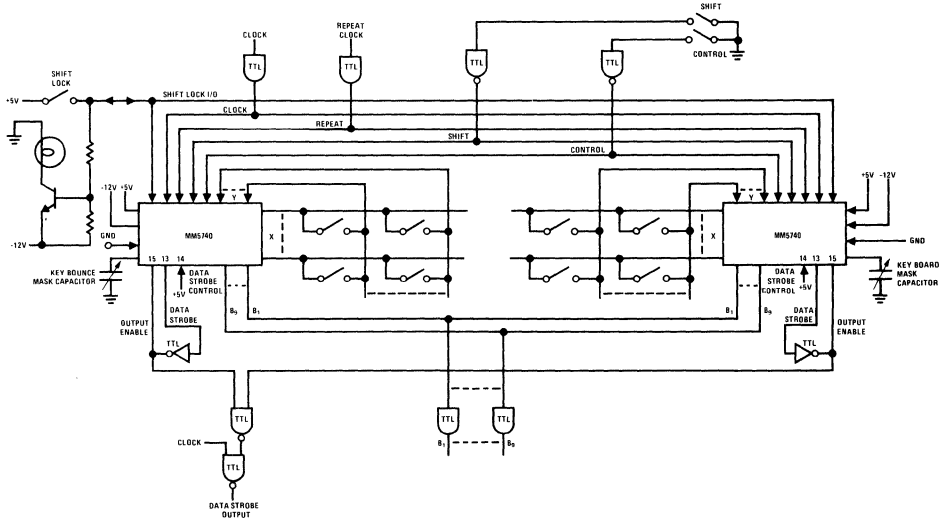


FIGURE 14. Two Encoders Give 180 Key, N-Key Rollover Operation

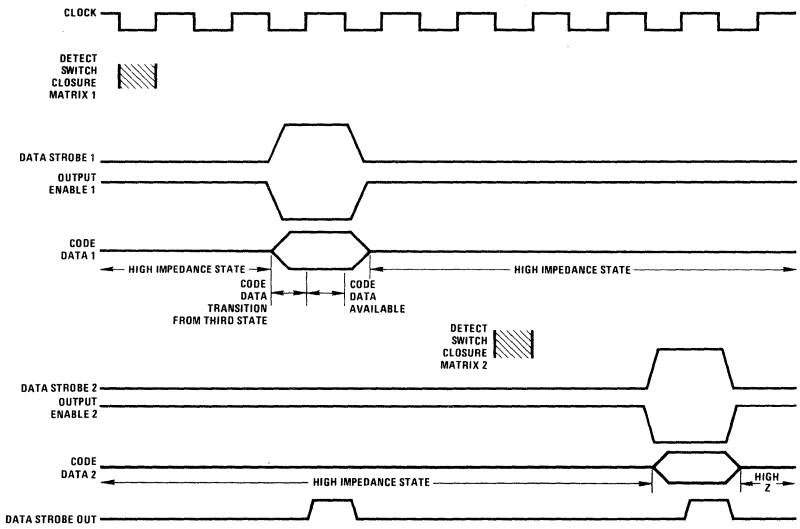


FIGURE 15. Typical Timing Diagram – 2 Encoder Expansion Operations

The keyswitches (up to 180) are wired in two X-Y matrices; one set for each MM5740 encoder. Both encoders operate asynchronously with respect to each other; at any time the scan position in one matrix may not be the same scan position in the second matrix. The essential point to remember is that no two keys may be depressed at "exactly" the same time and still result in a valid code output. This assumption is quite valid in that there is always some finite time period between the depression of a key and the depression of a succeeding key, even though that time may be as low as 5-10 ms. Anyone using a keyboard as an input device to his system must know what the maximum keyboard activity rate will be for his particular application. This information tells the system designer how much keybounce he can tolerate, what the encoder frequency should be, the type of rollover to specify, the width of the data strobe, whether his system is fast enough to accept asynchronous data from the keyboard without a buffer store and many other vital design specifications.

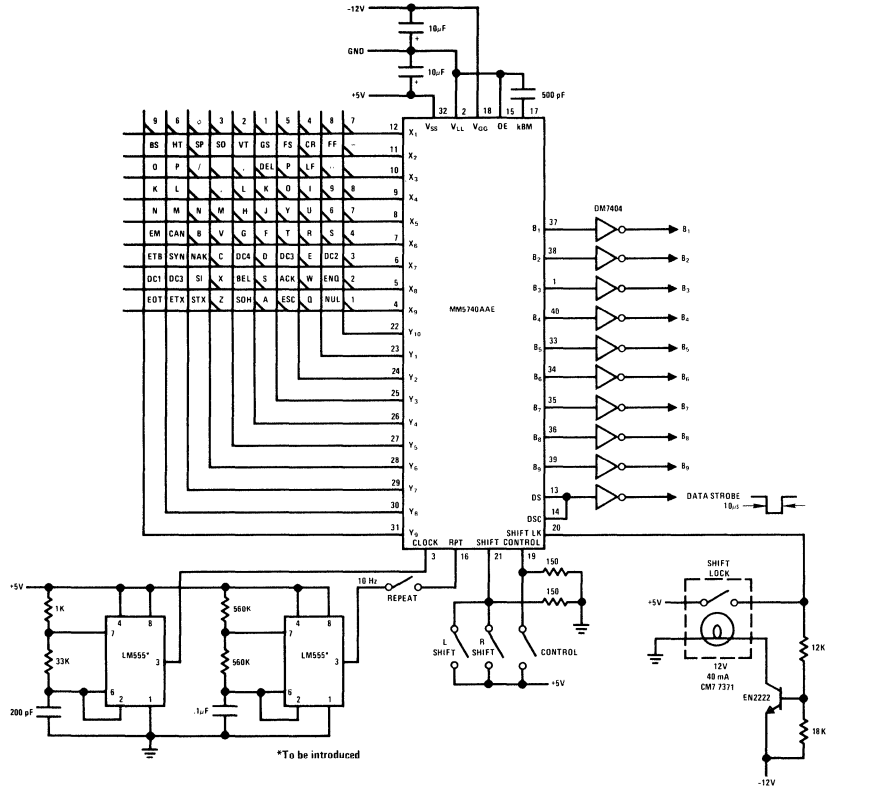
The above discussion relates to the application shown in Figure 14 in the following way. With no activity on the keyboard the data outputs of both encoders are in the third or high impedance state. A valid key closure in one matrix activates the data strobe and code data outputs of the corresponding encoder removing it from the high impedance state. The second encoder remains in the high impedance state. Resetting of the data strobe is automatically accomplished via the output enable pin and results in a one bit time wide data strobe output pulse. Code data is valid during the time that the output enable is low and the clock is high. After the first data strobe is reset both encoders are in the idle condition. If a valid closure is now detected in the second switch matrix, the second encoder performs exactly like the first one did a short time earlier. During this activity, the first encoder is in the high impedance state. The data strobe outputs from each encoder are combined in a 7400 gate to provide one data strobe output line for the keyboard system.

Since both key switch matrices are physically part of the same keyboard, two keys will not be activated at exactly the same time. Therefore, each encoder, operating asychronously with respect to the other, perform as separate entities. However, the receiving system which communicates with the keyboard encoders cannot distinguish that there are two encoders and two sub-matrices. Externally this arrangement appears as a single encoder with up to 180 key capability.

Complete ASR 33 Keyboard System

By using the MM5740AAE or MM5740AAF, typical ASR 33 type communications keyboards may be configured. These parts are available as standard off the shelf units. While both versions contain the same ROM encoding (all 90 key positions), the AAE provides N-key rollover and the AAF is programmed for 2-key rollover. It should be noted that many variations of ASR 33 keyboards may be configured by connecting to the appropriate matrix coordinates.

One common keyboard system is shown in Figure 16. The associated keyboard layout is shown in Figure 17 while the key codes are shown in Figure 18. It is apparent that a minimum of external hardware is needed to configure a total keyboard. The ease with which the MM5740 is able to interface with inexpensive TTL circuitry eliminates the need for many external resistors and level translators. It should also be noted that MM5740 TTL inputs may be supplied from standard TTL logic without any restrictions on the TTL-TTL loading. For example, the MM5740 clock input may be supplied from a standard TTL gate which also drives 10 other TTL loads. By incorporating any of the techniques discussed previously many variations of this basic keyboard system may be implemented with a minimum of additional hardware.



Note 1: Matrix coordinates not shown with switch are not used for this particular configuration. These coordinates, however, are programmed for the characters shown and may be used if desired.

Note 2: N-key rollover – MM5740AAE, 2-key rollover – MM5470AAF.

Note 3: Clock frequency = 100 kHz

Note 4: Scan cycle = 900μs

Note 5: Repeat rate = 10 characters per second

Note 6: Key bounce mask time = 4.0 ms

Note 7: Data strobe = 10μs pulse

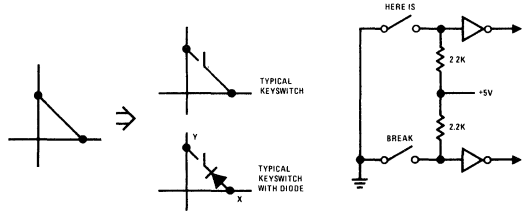
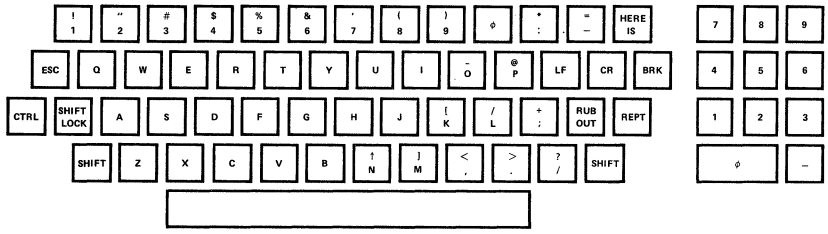


FIGURE 16. ASR-33 Keyboard



ASR 33
MM5740AAE (N-key rollover)
MM5740AAF (2-key rollover)

FIGURE 17. Typical Keyboard Arrangement

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	ϕ	ϕ	ϕ	ϕ
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		*	.	*

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.

FIGURE 18. Code Assignment Chart

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	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	6	&	6	&
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		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	1	1	0	0	1	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.

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	ENQ	ENQ
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	1	'	1	'

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.

FIGURE 18. Code Assignment Chart (cont'd)

CONCLUSION

The MM5740 keyboard encoder offers a wide versatility of features not found in conventional MSI designs. In addition, performance specifications of the MM5740 far exceed those of other MOS encoders currently available. The wide range of clock frequency, low power dissipation, variable bounce masking, ease of interface, rollover capability and its many other notable features make the MM5740 a logical choice for your keyboard encoding function. This encoder has been designed not only as a powerful part of any keyboard system but also with the intention of minimizing the amount of additional keyboard electronics and simplifying the interaction of the keyboard with other system components with which it must communicate. As a result, the MM5740 should go a long way toward improving the cost-performance basis of keyboard oriented systems.