THE TMS 5501 MULTIFUNCTION INPUT/OUTPUT CONTROLLER

This is a multifunction peripheral logic device built by Texas Instruments only. It is designed to work with 8080 or 8080A CPUs. The TMS 5501 does not use the 8228 System Controller; it decodes the Data Bus during the SYNC pulse.

The TMS 5501 provides many of the functions provided by the 8255 PPI, 8251 USART, 8253 Programmable Timer/Counter and 8259 Priority Interrupt Control Unit. In each case, the TMS 5501 has simpler logic, with fewer options; but for a very large number of applications, TMS 5501 features will be more than adequate.

Here are the TMS 5501 features provided:

1) **Two external interrupt request lines.**
2) **An 8-bit, parallel input port.**
3) **An 8-bit, parallel output port.**
4) **A single, asynchronous serial I/O channel without handshaking.**
5) **Five programmable timers, each of which times out with an interrupt request after an interval that may range from 64 microseconds to 16.32 milliseconds.**

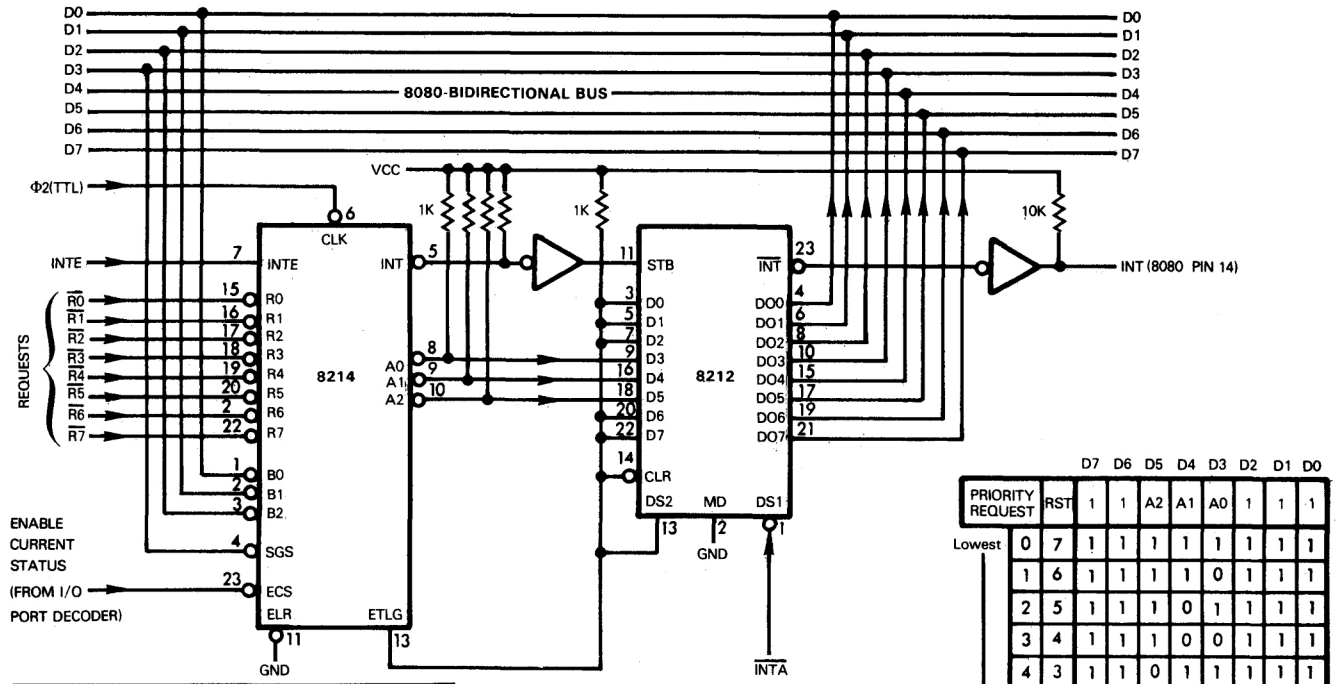
Figure 4-59 illustrates those logic functions in our standard microcomputer system illustration which have been implemented by the TMS 5501.

The TMS 5501 is fabricated using N-channel silicon gate technology and is packaged as a 40-pin DIP.

TMS 5501 DEVICE PINS AND SIGNALS

Figure 4-60 illustrates TMS 5501 device pins and signals. We will begin by summarizing these signals.

There are three data busses. D0 - D7 are the bidirectional Data Bus pins via which data is transferred between the TMS 5501 and the CPU. $\overline{XI0}$ - $\overline{XI7}$ are the pins via which external logic inputs 8-bit parallel data to the TMS 5501. $\overline{XO0}$ - $\overline{XO7}$ are the eight pins via which the TMS 5501 outputs 8-bit parallel data to external logic. Notice that $\overline{XO}$ lines are negative-true whereas XI lines are positive-true. Optionally XI7 may be used for low priority external interrupt requests.



PRIORITY REQUEST	RST	D7	D6	D5	D4	D3	D2	D1	D0
Lowest	0	7	1	1	1	1	1	1	1	1
	1	6	1	1	1	1	0	1	1	1
	2	5	1	1	1	0	1	1	1	1
	3	4	1	1	1	0	0	1	1	1
	4	3	1	1	0	1	1	1	1	1
	5	2	1	1	0	1	0	1	1	1
	6	1	1	1	0	0	1	1	1	1
Highest	7	0	1	1	0	0	0	1	1	1

* RST0 will vector Program Counter to location 0 (zero) and invoke the same routine as "RESET" input to 8080. This could reinitialize the system based on the routine invoked.

(A caution to system programmers.)

Figure 4-58. Generation Of The Restart Instruction Code Following An Interrupt. For An 8080A Microcomputer System

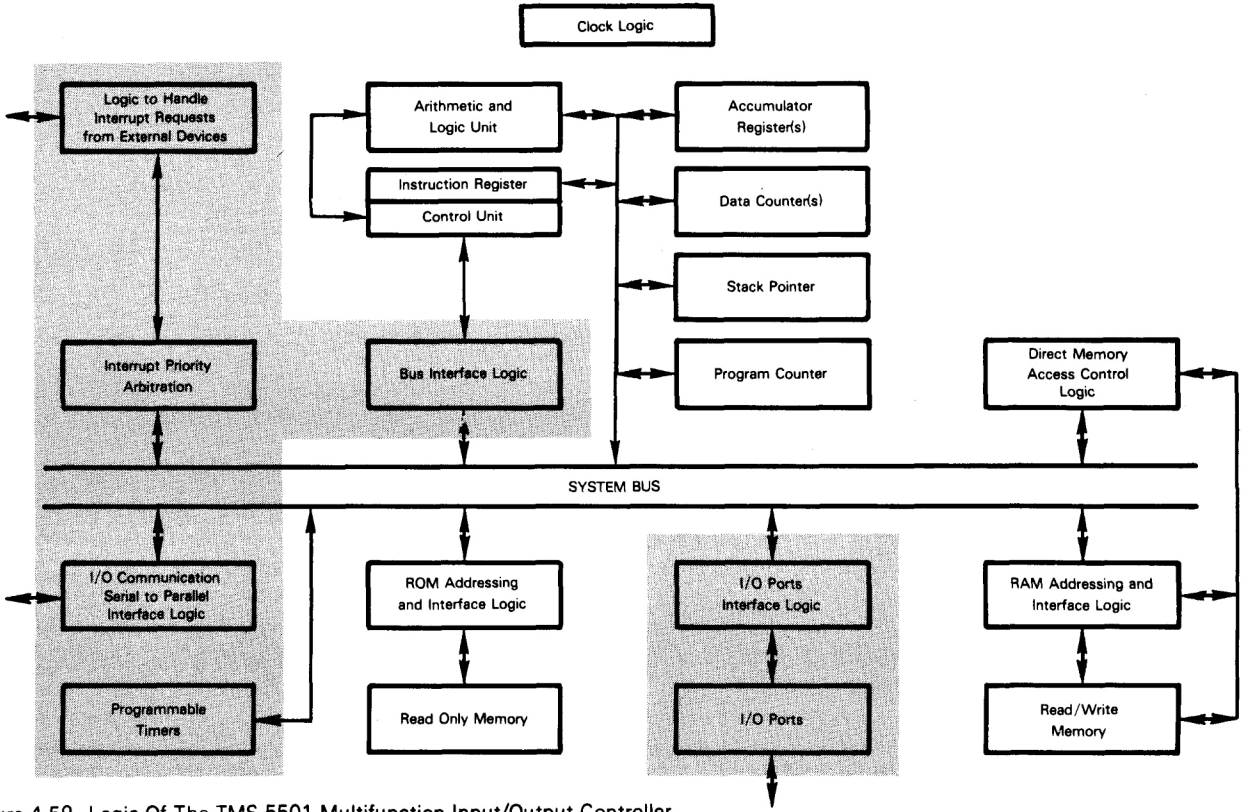
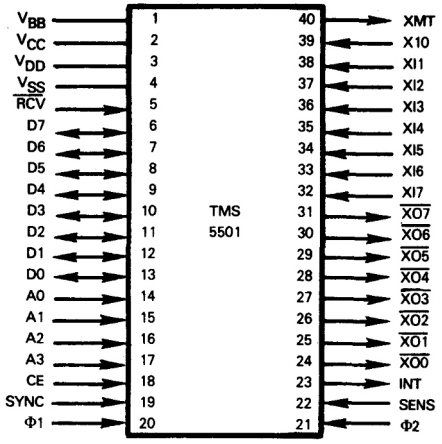


Figure 4-59. Logic Of The TMS 5501 Multifunction Input/Output Controller



PIN NAME	DESCRIPTION	TYPE
D0 - D7	Data Bus to CPU	Bidirectional
XI0 - XI7	Data Bus from external logic	Input
$\overline{XO0}$ - $\overline{XO7}$	Data Bus to external logic	Output
XMT	Transmit serial data line	Output
$\overline{RCV}$	Receive serial data line	Input
SENS	External interrupt sense	Input
INT	Interrupt request	Output
CE	Chip Select	Input
A0 - A3	Address Select	Input
SYNC	Synchronizing signal (SYNC) from 8080A	Input
Φ1, Φ2	Clock inputs, same as to 8080A	Input
V_{BB},V_{CC},V_{DD},V_{SS}	Power supply (-5V, +5V, +12V) and Ground	

Figure 4-60. TMS 5501 Multifunction Input/Output Controller Signals And Pin Assignments

TMS 5501 OUTPUT SIGNAL INVERSION

Do not miss the significance of $\overline{XO}$ negative logic; whatever you write to the TMS 5501 for parallel output will be complemented. $\overline{XO}$ signals are the inverse of the output buffer contents.

Serial I/O data uses the XMT and $\overline{RCV}$ pins. XMT is used to transmit serial data, whereas $\overline{RCV}$ is used to receive serial data. Note that $\overline{RCV}$ is a negative-true signal, whereas XMT is a positive-true signal.

External logic may request interrupt service either via the SENS input or via the XI7 input. A low-to-high transition on either signal constitutes an interrupt request. SENS is always part of external interrupt request logic; XI7 must be programmed for this purpose — in which case the eight XI pins cannot be used to input 8-bit parallel data.

Logic internal to the TMS 5501 may also generate interrupt requests. Whatever the source of the interrupt requests, it is passed on to the CPU via the INT interrupt request signal.

The TMS 5501 is accessed either as 16 I/O ports or 16 memory locations. Addressing logic consists of a chip select (CE) and four address select inputs (A0, A1, A2 and A3).

The TMS 5501 receives the SYNC timing pulse, and this requires special mention. While SYNC is high, the TMS 5501 decodes status off the Data Bus, therefore the 8228 System Controller is not needed.

Additional signals required by the TMS 5501 are the identical two 8080A clock signals Φ1 and Φ2. Slight clock signal variations will confuse serial I/O logic which computes baud rates internally.

TMS 5501 WAIT STATE

A feature of the TMS 5501 which you must note carefully is that it cannot handle Wait states. Any T_W clock periods in a machine cycle will cause the TMS 5501 to malfunction.

There is a further unlikely ramification of the TMS 5501 inability to handle Wait states. **If you are accessing the TMS 5501 as 16 memory locations, then you cannot have a Halt instruction's object code in the memory location immediately preceding the 16 TMS 5501 addresses.** If you do, the Halt instruction will execute, following which the Address Bus will contain the address of the next sequential memory location — which now is a TMS 5501 address. Thus, the TMS 5501 becomes selected. But the TMS 5501 logic cannot cope with a sequence of undefined clock periods, which is exactly what will happen following a Halt instruction's execution. The net effect is that following a Halt, the TMS 5501 receiver buffer loaded flag will be inadvertently cleared.

Always make sure that the memory address directly preceding the 16 addresses assigned to a TMS 5501 remains unused.

TMS 5501 DEVICE ACCESS

Some of the 16 I/O port or memory addresses via which the TMS 5501 device is accessed are equivalent to memory locations, but others are command identifiers. Table 4-16 defines the manner in which addresses are interpreted.

You will find the TMS 5501 far easier to use if you address it as 16 memory locations, because that will give you access to memory referencing instructions.

When creating TMS 5501 select logic, any of the select schemes described earlier in this chapter will do — with one addition. **Include READY as part of the select logic;** if READY is low, a Wait state will follow, and that will cause the TMS 5501 to malfunction. By making READY high a necessary component of device select logic, you can avoid this problem.

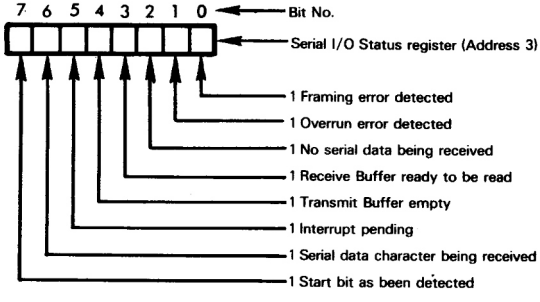
In the following discussion of individual TMS 5501 capabilities, we will use programming examples to show the effectiveness of including the TMS 5501 device within your memory rather than I/O space.

Table 4-16. TMS 5501 Address Interpretations

A3	A2	A1	A0	FUNCTION
0	0	0	0	Read assembled serial input data byte out of Receiver Buffer
0	0	0	1	Read parallel data input via XI0 - XI7
0	0	1	0	Read RST instruction code, as a data byte, when polling interrupt requests
0	0	1	1	Read Status register contents to the CPU
0	1	0	0	Write command code to the TMS 5501
0	1	0	1	Load serial I/O Control register, specifying baud rate and stop bits
0	1	1	0	Write data byte to serial transmit logic
0	1	1	1	Write data byte to parallel output port
1	0	0	0	Write out interrupt mask byte to selectively enable and disable interrupts
1	0	0	1	Write initial count to Interval Timer 1
1	0	1	0	Write initial count to Interval Timer 2
1	0	1	1	Write initial count to Interval Timer 3
1	1	0	0	Write initial count to Interval Timer 4
1	1	0	1	Write initial count to Interval Timer 5
1	1	1	0	No Operation
1	1	1	1	No Operation

TMS 5501 addressable locations 3, 4 and 5 are used for status and controls which generally apply to serial I/O and interrupt processing. We will define how these ports are used now, in advance of our discussion of TMS 5501 serial I/O and interrupt processing capabilities.

Locations 3 and 5 apply to serial I/O logic. Location 3 is a Status register whose bits are interpreted as follows:

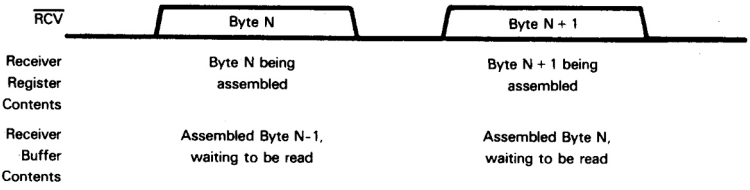


Bits 0 and 1 are standard framing and overrun error indicators.

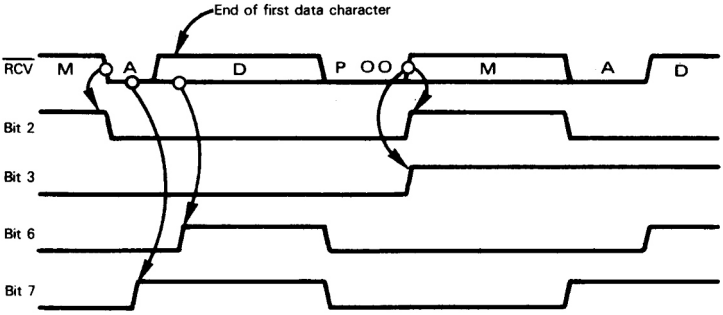
If a framing error is detected, Status register bit 0 will be set to 1 and will remain 1 until assembly of the next complete serial data character has been completed.

If Receiver Buffer contents are not read while the next serial character is being input and assembled, an overrun error will be reported in bit 1 of the Status register. This error indicator will be cleared as soon as the Status register contents are read, or when a reset command is output. Remember, you have the time it takes to receive and assemble one character in which to read the previous character out of the Receiver Buffer. This is because receive logic includes a double buffer. A character is assembled in a

Receiver register; when completely assembled, it is shifted to a Receiver Buffer and the next character is assembled in the Receiver register:



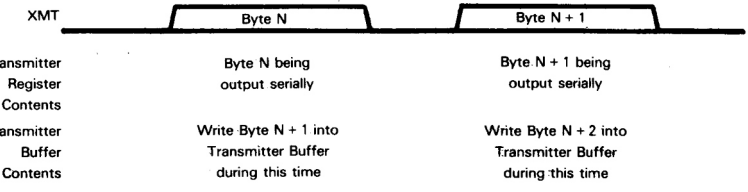
Status bits 2, 3, 6 and 7 monitor the condition of the serial data input signal. During a break, that is, when no valid serial data is being input, status bit 2 will be high. As soon as a start bit has been detected, status bit 2 will be reset low and status bit 7 will be set high. When the first valid data bit is detected, status bit 6 is also set high. When the received character has been assembled in the Receiver Buffer, and may be read by the CPU, status bits 7 and 6 are reset and status bit 3 is set. This may be illustrated as follows:



M Marking
A Start bit
D Data bits
P Parity bit
O Stop bits

Status bit 4 applies to serial transmit logic. As soon as the Transmit Buffer is ready to receive another byte of data, status bit 4 will be set high. It will remain high until new data has been loaded into the Transmit Buffer.

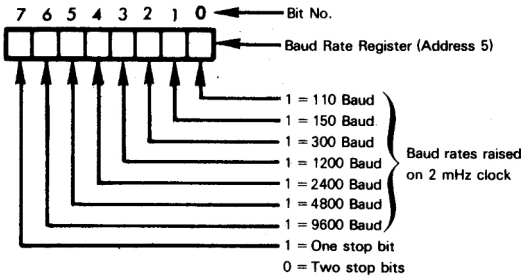
Transmit logic, like receive logic, is double-buffered. A byte of data is held in a Transmitter register while being output serially; meanwhile, the next data byte may be loaded into a Transmitter Buffer. Transmitter Buffer contents are automatically shifted to the Transmitter register when serial output of a data byte is complete. This may be illustrated as follows:



Status bit 4 is high from the instant Transmitter Buffer contents are shifted into the Transmitter register, until a new data byte is written into the Transmitter buffer.

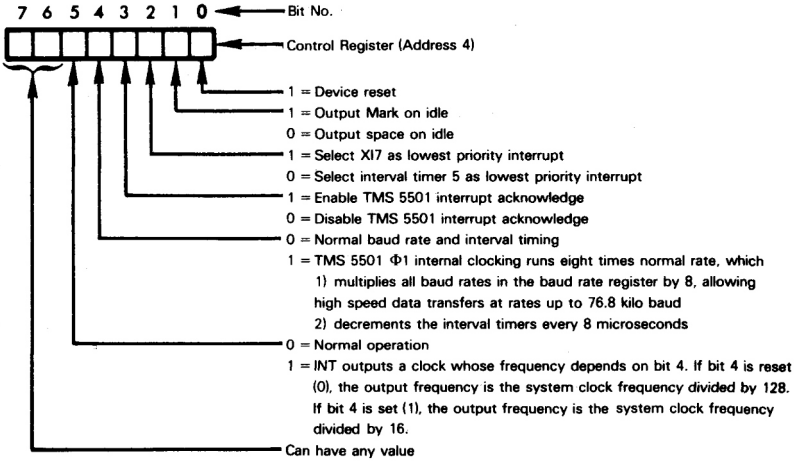
Status bit 5 is set whenever the TMS 5501 has an unacknowledged interrupt request. While this status bit is very important in serial I/O operations, it also may have application elsewhere; this bit therefore may be looked upon as an exception within the Status register, in that it is the only status flag that does not apply strictly to serial I/O operations.

TMS 5501 addressable location 5 is also dedicated to serial I/O. Into this location you must load a control byte which selects baud rate, and the number of stop bits. Register contents will be interpreted as follows:



If more than one of bits 0 through 6 are high, then the highest indicated baud rate will be selected. If no baud rate bit is high, then all serial transmit and receive logic will be inhibited.

TMS 5501 addressable location 4 is a general command register. Its contents will be interpreted as follows:



If your system does not require interrupts from the TMS 5501, you can set bit 5 high to derive a TTL compatible clock from the INT output.

If the TMS 5501 device is reset by outputting 1 to bit 0, then the following events will occur:

TMS 5501 RESET

1) Serial receive logic enters the Hunt mode. Status bits 2, 3, 6 and 7 are all reset; however, reset will not clear the Receive Buffer contents.
2) Serial transmit logic will output a high marking signal. Status bit 4 will be set high indicating that transmit logic is ready to receive another data byte.
3) The interrupt mask register is cleared with the exception of the Transmit Buffer interrupt, which is enabled. (Interrupt levels and interrupt masking are described shortly.)
4) All interval timers are halted.

The Reset has no effect on any of the following:

- Parallel input and output port contents
- Interrupt acknowledge enable
- Interrupt Mask register contents
- Baud rate register contents
- Serial Transmit or Receive Buffer contents

Control command bit 1 determines whether serial transmit logic will mark or space when not transmitting data. A 1 in bit 1 will cause serial transmit logic to mark (output high) while a 0 in bit 1 will cause transmit logic to space (output low).

If Reset conflicts with the break specification, then Reset will override and transmit logic will mark, irrespective of the break bit specification.

The TMS 5501 can receive an interrupt request from one of nine different sources. Using the eight Restart instructions, each interrupt request is assigned one of eight priorities. For this to be possible, two interrupt sources share the lowest priority interrupt level (RST 7); these two sources are an external request arriving via XI7 and the Interval Timer 5 time out interrupt request. **You use bit 2 of the control command to select which requesting source will be active at any time as the lowest priority interrupt.**

Bit 3 of the control command is a master enable/disable for TMS 5501 interrupt logic. If this bit is output as 0, then TMS 5501 interrupt acknowledge logic is disabled — and that effectively disables the entire interrupt processing system. Observe that with interrupt acknowledge logic disabled you can still use polling techniques in lieu of interrupt processing.

Table 4-17. TMS 5501 Interrupt Logic And Priorities

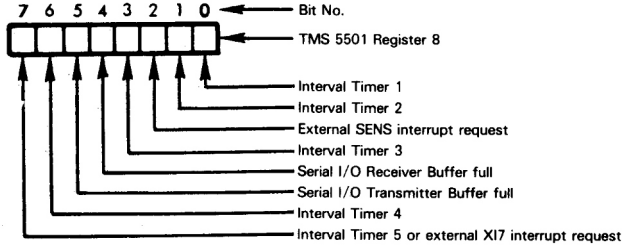
Interrupt and Mask Bit	Data Bus Status			RST Instruction	Interrupting Source
	D5	D4	D3		
0 (highest)	0	0	0	RST 0	Interval Timer 1
1	0	0	1	RST 1	Interval Timer 2
2	0	1	0	RST 2	External SENS interrupt request
3	0	1	1	RST 3	Interval Timer 3
4	1	0	0	RST 4	Serial I/O Receiver Buffer full
5	1	0	1	RST 5	Serial I/O Transmitter Buffer full
6	1	1	0	RST 6	Interval Timer 4
7 (lowest)	1	1	1	RST 7	Interval Timer 5, or external XI7 interrupt request, whichever has been selected by command code

TMS 5501 INTERRUPT HANDLING

The TMS 5501 responds to nine different interrupt requests, with priorities as defined in Table 4-17.

When an interrupt is acknowledged, INT is output high by the TMS 5501. If the TMS 5501 INT output is connected to the 8080A INT input, then the 8080A will acknowledge the interrupt by outputting D1 high at SYNC high. The TMS 5501 responds to this acknowledge by placing an RST instruction's object code on the Data Bus, as required by standard 8080A timing. **This is an utterly standard 8080A interrupt request/acknowledge sequence.**

Interrupts may be selectively disabled by writing a mask to TMS 5501 Register 8; see Table 4-16. A 0 bit will disable an interrupt; mask bits are related to priorities as follows:



Note that TMS 5501 interrupt priorities apply to the request/acknowledge sequence only —which is the standard passive interrupt priority arbitration sequence used in most microcomputer applications. Once an interrupt is acknowledged and is being serviced by an interrupt service routine, it is up to the programmer to disable all interrupts, or selected interrupts, if the interrupt service routine is not itself to get interrupted. If, for example, an interrupt were to be acknowledged at priority 3 (Interval Timer 3), in the normal course of events the 8080A CPU will disable all interrupts upon acknowledging any interrupt. Therefore the Interval Timer 3 interrupt service routine will deny any other interrupt request, whatever its priority, until the Interval Timer 3 service routine completes execution. If the Interval Timer 3 interrupt service routine were to immediately enable all interrupts, then any other interrupt request would be acknowledged, irrespective of priority.

If you want to ensure that only higher priority requests interrupt the Timer 3 service routine, then the Timer 3 service routine must begin by outputting a mask to disable all lower level interrupts at the TMS 5501; then it must enable all interrupts at the CPU. Here is the necessary instruction sequence:

```
MVI     TMS8,07H        ;OUTPUT MASK TO REGISTER 8 OF TMS 5501
EI                      ;ENABLE INTERRUPTS
```

The mask output in this case has the value 07, since mask bits 0, 1 and 2 only must be set to 1, enabling the highest three interrupt priority levels.

TMS 5501 NONSTANDARD FEATURES

Let us now look at the nonstandard features associated with TMS 5501 interrupt handling logic. First of all, so long as there is an unacknowledged interrupt request, Status register bit 5 is set to 1; next the RST instruction object code for the highest level interrupt request is stored in TMS 5501 Register 2. This allows you to bypass normal interrupt processing logic and poll the TMS 5501 instead.

In order to bypass interrupt logic, simply disconnect the TMS 5501 INT output from the 8080A INT input. You can still identify interrupt requests occurring within the TMS 5501 by reading the TMS 5501 Status register. If bit 5 of the Status register is 1, then one or more interrupt requests are active within the TMS 5501. In order to determine which is the highest level active interrupt request, read the contents of TMS 5501 memory location 2. The RST instruction object code corresponding to the highest priority interrupt request will have been assembled in this location. Bits 3, 4 and 5 of the RST instruction object code identify the priority level. Thus you can determine which of the eight priority levels was the highest active interrupt request. Here is a typical polling sequence:

```
;ASSUME THAT THE TMS 5501 ADDRESS SPACE CONSISTS OF 16 MEMORY
;LOCATIONS FROM 8000 THROUGH 800F. TMS5 IS THE SYMBOL ASSIGNED
;TO THE BASE ADDRESS
TMS5    EQU     8000H
        -
        -
        -
;TEST STATUS REGISTER FOR INTERRUPT PENDING
        LDA     TMS5+3      ;LOAD STATUS TO ACCUMULATOR
        ANI     20H         ;ISOLATE BIT 5
        JNZ     TMS5+2      ;IF NOT ZERO, AN INTERRUPT HAS BEEN
                            ;REQUESTED
        -
        -
        -
```

It is worth spending a minute looking at the three-instruction sequence illustrated above. The TMS 5501 Status register contents are loaded into the Accumulator by the LDA instruction. The next instruction isolates bit 5. If bit 5 is 1, then an interrupt has been requested, and the next instruction, a JNZ, branches program execution to a memory location within the TMS 5501 itself. Will that work? Indeed, it will. The label TMS5+2 addresses TMS 5501 Register 2, which contains an RST instruction's object code; this is the object code which would have been output in response to a normal interrupt acknowledge. What the JNZ instruction does is cause this RST instruction's object code to be executed next; and that is precisely the logic sequence which a normal interrupt response would have implemented.

Notice that the very simple method we have illustrated for polling on status only works if the TMS 5501 can be addressed as memory locations rather than I/O ports.

TMS 5501 PARALLEL I/O OPERATIONS

It is very easy to handle simple parallel I/O, without handshaking, using the TMS 5501. This is equivalent to 8255 Mode 0 operation. TMS 5501 address 1 accesses the parallel 8-bit input port, while address 7 accesses a parallel 8-bit output port (see Table 4-16). Assuming that the TMS 5501 is addressed as memory, input and output operations are handled using any memory reference instructions.

A very limited amount of parallel I/O handshaking is available. The SENS interrupt input signal can be used by external logic either to indicate that it has read output data, or to indicate that it has transmitted input data. However, the TMS 5501 device itself has no control signals which can be used to prompt external logic; that is to say, the TMS 5501 has no signal equivalent to the 8255 $\overline{\text{OBF}}$ control. When comparing the parallel I/O capabilities of the TMS 5501 with the 8255, therefore, we conclude that 8255 Mode 0 operations can be duplicated without problems, but neither Mode 1 nor Mode 2 parallel I/O operations with handshaking can be duplicated. Only a primitive level of parallel I/O with handshaking exists within the TMS 5501 and even this exists at the expense of external interrupt logic.

TMS 5501 SERIAL I/O OPERATION

A significant asynchronous, serial I/O capability is provided by the TMS 5501. Synchronous serial I/O is not supported.

There are very significant differences between the implementation of asynchronous serial I/O by the TMS 5501, as compared to the 8251 USART.

The TMS 5501 has separate serial transmit and receive pins (XMT and $\overline{\text{RCV}}$), but it has no accompanying handshaking control signals; instead 5th and 6th priority interrupts identify Receiver Buffer full and Transmit Buffer full, respectively. Bits 2, 3, 6 and 7 of the Status register (addressable location 3) identify the condition of a serial receive data stream.

When using the TMS 5501, you have to continuously read in the contents of the Status register and test the condition of appropriate status bits in order to implement standard serial receive logic; however, in the end you can implement the same serial receive logic as is provided automatically by the 8251 USART. Here is the relationship between the TMS 5501 and the 8251 USART controls:

8251 USART	TMS 5501 EQUIVALENT
TxRDY	Status register bit 4
TxE	None
TxC	Baud Rate register
RxRDY	Status register bit 3
RxC	Baud Rate register
SYNDET	None

Probably the most significant difference between TMS 5501 and 8251 USART control is the fact that TMS 5501 baud rate is programmed by outputting an appropriate Control code, while it is clocked by rate signals input to the 8251 USART. The TMS 5501 advantage is that the TMS 5501 does not need external baud rate clock generation logic; however there must be a very precise synchronization between the TMS 5501 and whatever external logic it is communicating with. Minor timing differences are no problem when using an 8251 USART since a clock signal can accompany the serial data stream. Minor timing differences can be intolerable when using the TMS 5501; a small difference between TMS 5501 baud rate and external clock signals can generate very significant errors.

TMS 5501 INTERVAL TIMERS

The TMS 5501 has five programmable Interval Timers. Each timer can be loaded with an initial count ranging from 01 (lowest) through FF_{16} (highest). Each Timer will decrement one count every 64 microseconds. As soon as a programmable timer counts out to zero, it requests an interrupt. In our discussion of TMS 5501 interrupt logic, we have defined the priority levels assigned to the various Interval Timers. Notice that Interval Timer priorities have been spread across the range of priority levels. By using Interval Timer 1 or 2, you can be sure of precise time intervals, since an interrupt request will be acknowledged with little or no delay. Timers 4 and 5, being the lowest priority, can be used to generate less precise time intervals. It is conceivable that interrupt requests originating at these two timers might have to wait a significant amount of time before being serviced — if there is any degree of interrupt traffic within the microcomputer system.

Loading a 0 value into an Interval Timer causes an immediate interrupt request.

When a nonzero value is loaded into an Interval Timer, it starts to count down immediately. If a new value is loaded into an Interval Timer while it is halfway through counting out, then the new value will be accepted; it will override the previous value and subsequently will be decremented. Therefore the Interval Timers are retriggerable.

Once an Interval Timer counts out, it halts.