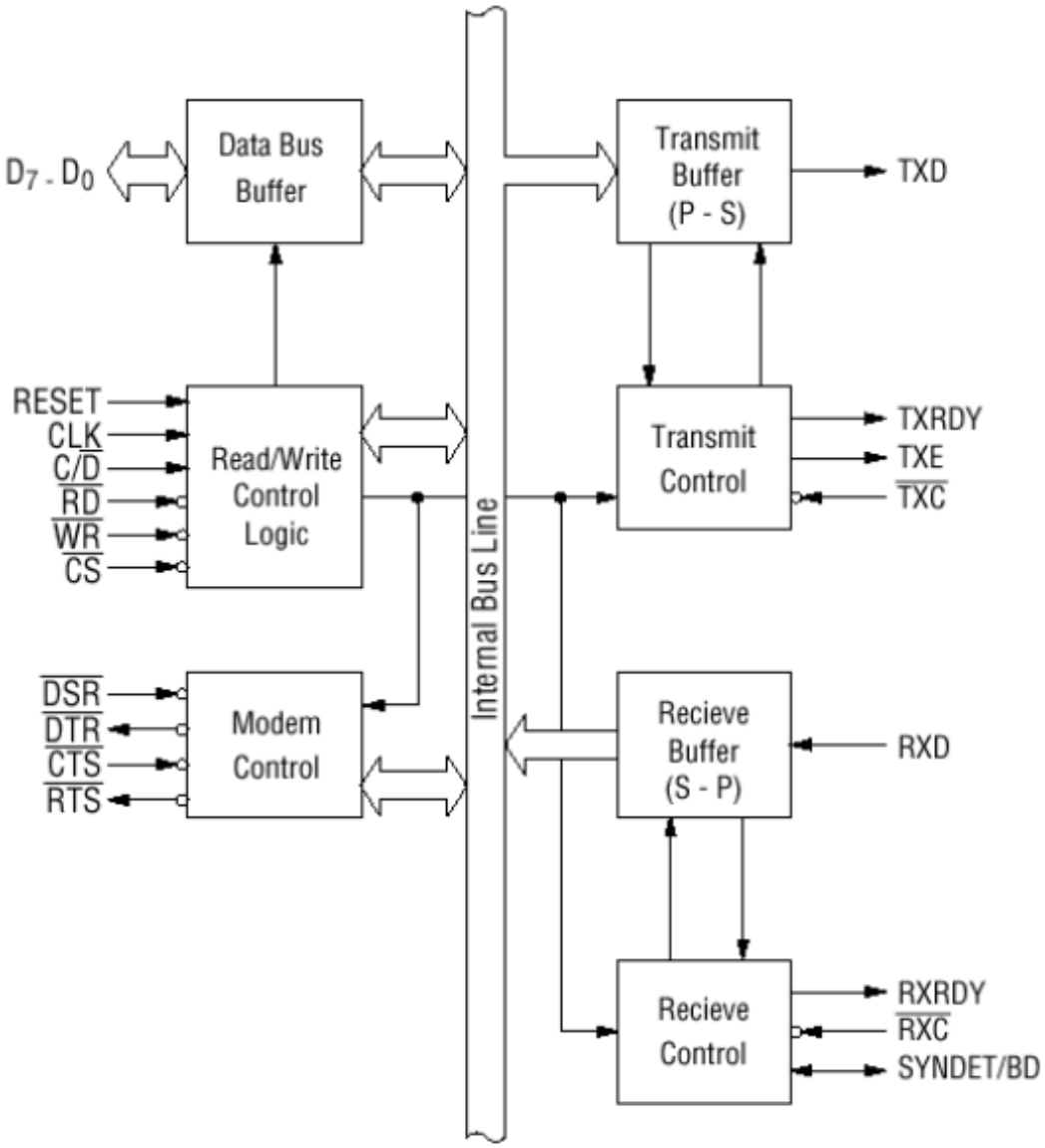
Programmable Communication Interface 8251 USART

8251 is a USART (Universal Synchronous Asynchronous Receiver Transmitter) for serial data communication. As a peripheral device of a microcomputer system, the 8251 receives parallel data from the CPU and transmits serial data after conversion. This device also receives serial data from the outside and transmits parallel data to the CPU after conversion.



Block diagram of the 8251 USART (Universal Synchronous Asynchronous Receiver Transmitter)

The 8251 functional configuration is programed by software. Operation between the 8251 and a CPU is executed by program control. Table 1 shows the operation between a CPU and the device.

$\overline{CS}$	C/$\overline{D}$	$\overline{RD}$	$\overline{WR}$	
1	×	×	×	Data Bus 3-State
0	×	1	1	Data Bus 3-State
0	1	0	1	Status → CPU
0	1	1	0	Control Word ← CPU
0	0	0	1	Data → CPU
0	0	1	0	Data ← CPU

Table 1 Operation between a CPU and 8251

Control Words

There are two types of control word.

1. Mode instruction (setting of function)

2. Command (setting of operation)

1) Mode Instruction

Mode instruction is used for setting the function of the 8251. Mode instruction will be in "wait for write" at either internal reset or external reset. That is, the writing of a control word after resetting will be recognized as a "mode instruction."

Items set by mode instruction are as follows:

- Synchronous/asynchronous mode
- Stop bit length (asynchronous mode)
- Character length

• Parity bit

• Baud rate factor (asynchronous mode)

• Internal/external synchronization (synchronous mode)

• Number of synchronous characters (Synchronous mode)

The bit configuration of mode instruction is shown in Figures 2 and 3. In the case of synchronous mode, it is necessary to write one-or two byte sync characters. If sync characters were written, a function will be set because the writing of sync characters constitutes part of mode instruction.

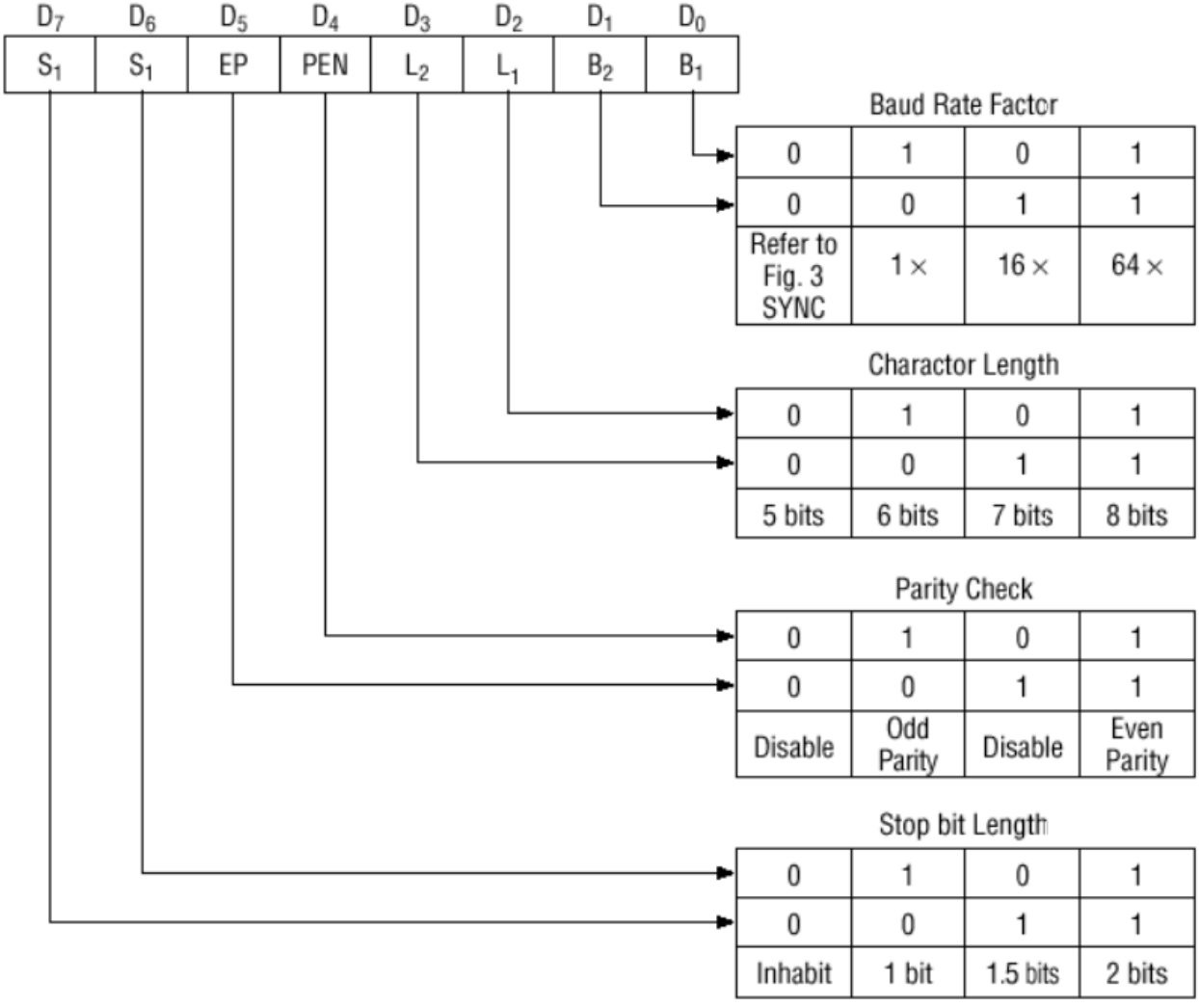


Fig. 2 Bit Configuration of Mode Instruction (Asynchronous)

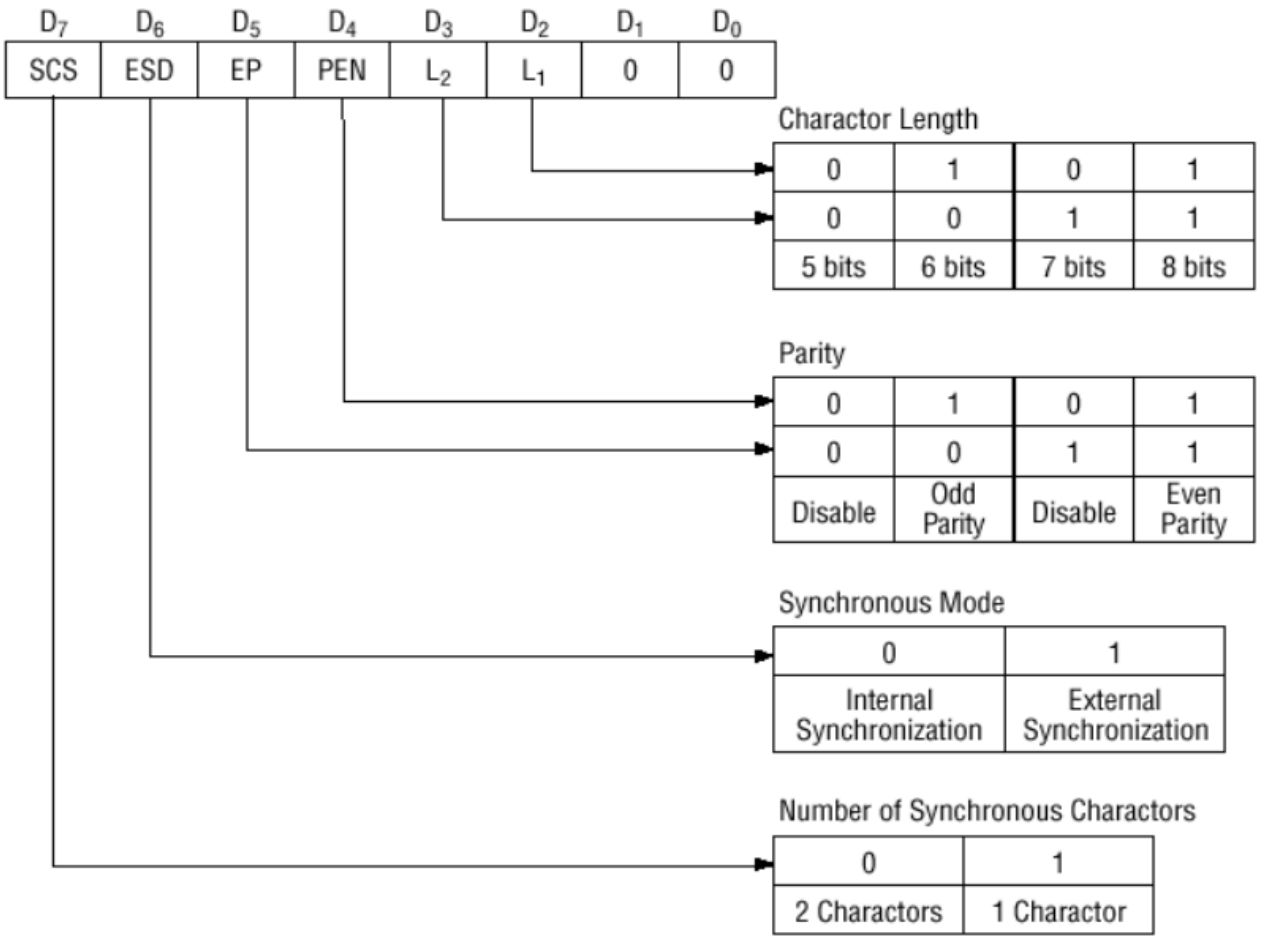


Fig. 3 Bit Configuration of Mode Instruction (Synchronous)

2) Command

Command is used for setting the operation of the 8251. It is possible to write a command whenever necessary after writing a mode instruction and sync characters.

Items to be set by command are as follows:

- Transmit Enable/Disable
- Receive Enable/Disable
- DTR, RTS Output of data.
- Resetting of error flag.

- Sending to break characters
- Internal resetting
- Hunt mode (synchronous mode)

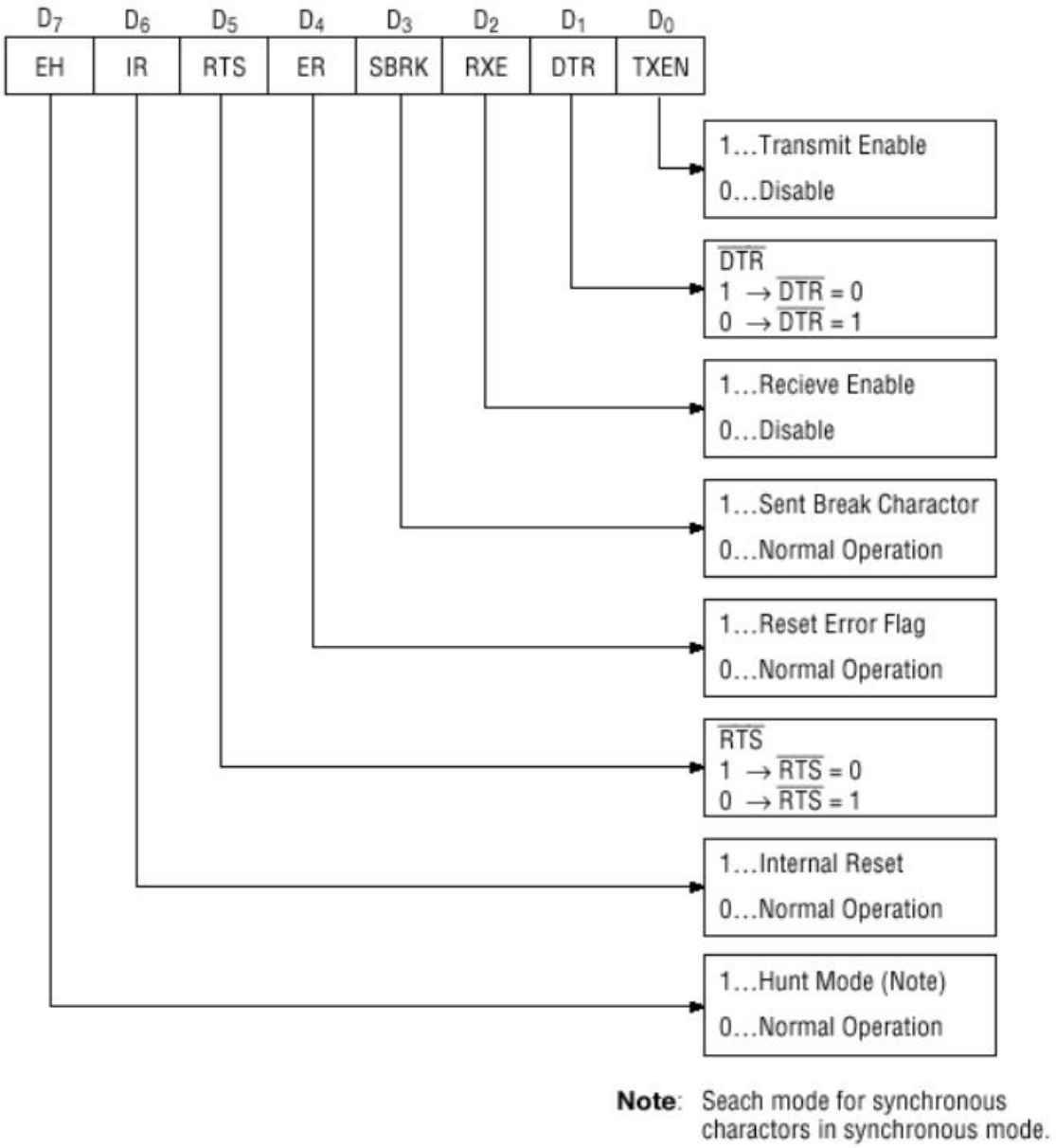


Fig. 4 Bit Configuration of Command

Status Word

It is possible to see the internal status of the 8251 by reading a status word. The bit configuration of status word is shown in Fig. 5.

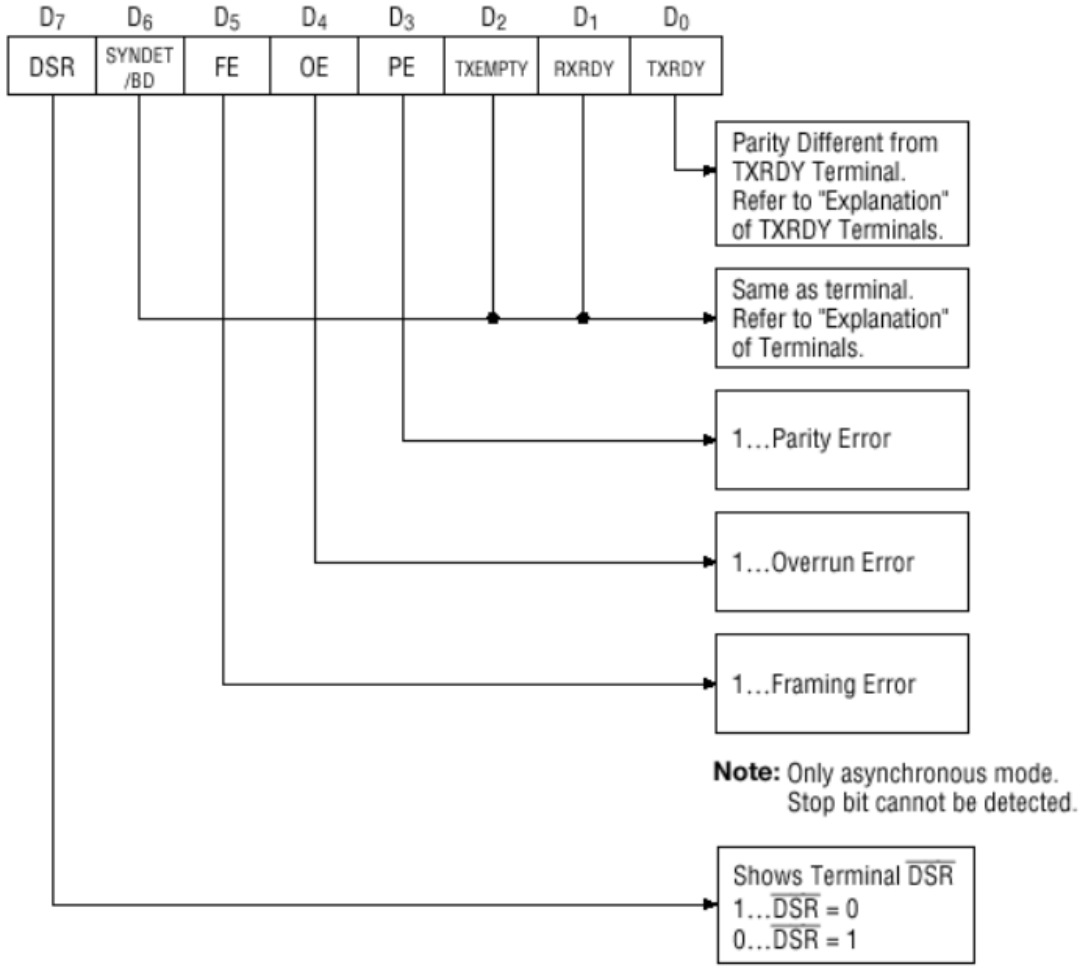


Fig. 5 Bit Configuration of Status Word

Pin Description

D 0 to D 7 (I/O terminal)

This is bidirectional data bus which receive control words and transmits data from the CPU and sends status words and received data to CPU.

RESET (Input terminal)

A "High" on this input forces the 8251 into "reset status." The device waits for the writing of "mode instruction." The min. reset width is six clock inputs during the operating status of CLK.

CLK (Input terminal)

CLK signal is used to generate internal device timing. CLK signal is independent of RXC or TXC. However, the frequency of CLK must be greater than 30 times the RXC and TXC at Synchronous mode and Asynchronous "x1" mode, and must be greater than 5 times at Asynchronous "x16" and "x64" mode.

WR (Input terminal)

This is the "active low" input terminal which receives a signal for writing transmit data and control words from the CPU into the 8251.

RD (Input terminal)

This is the "active low" input terminal which receives a signal for reading receive data and status words from the 8251.

C/D (Input terminal)

This is an input terminal which receives a signal for selecting data or command words and status words when the 8251 is accessed by the CPU. If C/D = low, data will be accessed. If C/D = high, command word or status word will be accessed.

CS (Input terminal)

This is the "active low" input terminal which selects the 8251 at low level when the CPU accesses. Note: The device won't be in "standby status"; only setting CS = High.

TXD (output terminal)

This is an output terminal for transmitting data from which serial-converted data is sent out. The device is in "mark status" (high level) after resetting or during a status when transmit is disabled. It is also possible to set the device in "break status" (low level) by a command.

TXRDY (output terminal)

This is an output terminal which indicates that the 8251is ready to accept a transmitted data character. But the terminal is always at low level if CTS = high or the device was set in "TX disable status" by a command. Note: TXRDY status word indicates that transmit data character is receivable, regardless of CTS or

command. If the CPU writes a data character, TXRDY will be reset by the leading edge or WR signal.

TXEMPTY (Output terminal)

This is an output terminal which indicates that the 8251 has transmitted all the characters and had no data character. In "synchronous mode," the terminal is at high level, if transmit data characters are no longer remaining and sync characters are automatically transmitted. If the CPU writes a data character, TXEMPTY will be reset by the leading edge of WR signal. Note : As the transmitter is disabled by setting CTS "High" or command, data written before disable will be sent out. Then TXD and TXEMPTY will be "High". Even if a data is written after disable, that data is not sent out and TXE will be "High".After the transmitter is enabled, it sent out. (Refer to Timing Chart of Transmitter Control and Flag Timing)

TXC (Input terminal)

This is a clock input signal which determines the transfer speed of transmitted data. In "synchronous mode," the baud rate will be the same as the frequency of TXC. In "asynchronous mode", it is possible to select the baud rate factor by mode instruction. It can be 1, 1/16 or 1/64 the TXC. The falling edge of TXC sifts the serial data out of the 8251.

RXD (input terminal)

This is a terminal which receives serial data.

RXRDY (Output terminal)

This is a terminal which indicates that the 8251 contains a character that is ready to READ. If the CPU reads a data character, RXRDY will be reset by the leading edge of RD signal. Unless the CPU reads a data character before the next one is received completely, the preceding data will be lost. In such a case, an overrun error flag status word will be set.

RXC (Input terminal)

This is a clock input signal which determines the transfer speed of received data. In "synchronous mode," the baud rate is the same as the frequency of RXC. In "asynchronous mode," it is possible to select the baud rate factor by mode instruction. It can be 1, 1/16, 1/64 the RXC.

SYNDET/BD (Input or output terminal)

This is a terminal whose function changes according to mode. In "internal synchronous mode." this terminal is at high level, if sync characters are received and synchronized. If a status word is read, the terminal will be reset. In "external synchronous mode, "this is an input terminal. A "High" on this input forces the 8251 to start receiving data characters.

In "asynchronous mode," this is an output terminal which generates "high level"output upon the detection of a "break" character if receiver data contains a "low-level" space between the stop bits of two continuous characters. The terminal will be reset, if RXD is at high level. After Reset is active, the terminal will be output at low level.

DSR (Input terminal)

This is an input port for MODEM interface. The input status of the terminal can be recognized by the CPU reading status words.

DTR (Output terminal)

This is an output port for MODEM interface. It is possible to set the status of DTR by a command.

CTS (Input terminal)

This is an input terminal for MODEM interface which is used for controlling a transmit circuit. The terminal controls data transmission if the device is set in "TX Enable" status by a command. Data is transmitable if the terminal is at low level.

RTS (Output terminal)

This is an output port for MODEM interface. It is possible to set the status RTS by a command.

8527 DMA Controller

The i8527 controller has four independent channels each of which contains an address register and a counter. The counter decrements as each byte transfer occur, and forces termination of the DMA operation after the last transfer. The controller increments the address registers after each operation, so that successive data transfers are made at contiguous ascending addresses. The arbiter resolves conflicts among the channels for access to memory. Two methods have been used in this

chip to make the chip useful in a variety of different applications. In one mode the channels have a fixed priority and conflicts are resolved according to the priority, for example, Channel 0 has highest priority and Channel 3 lowest. The second mode is a rotating priority scheme in which priority rankings are the four cycle shifts of 0-1-2-3, when a channel is granted access to the bus the priority ranking shifts cyclically to place the channel in the lowest priority position for the next arbitration cycle.

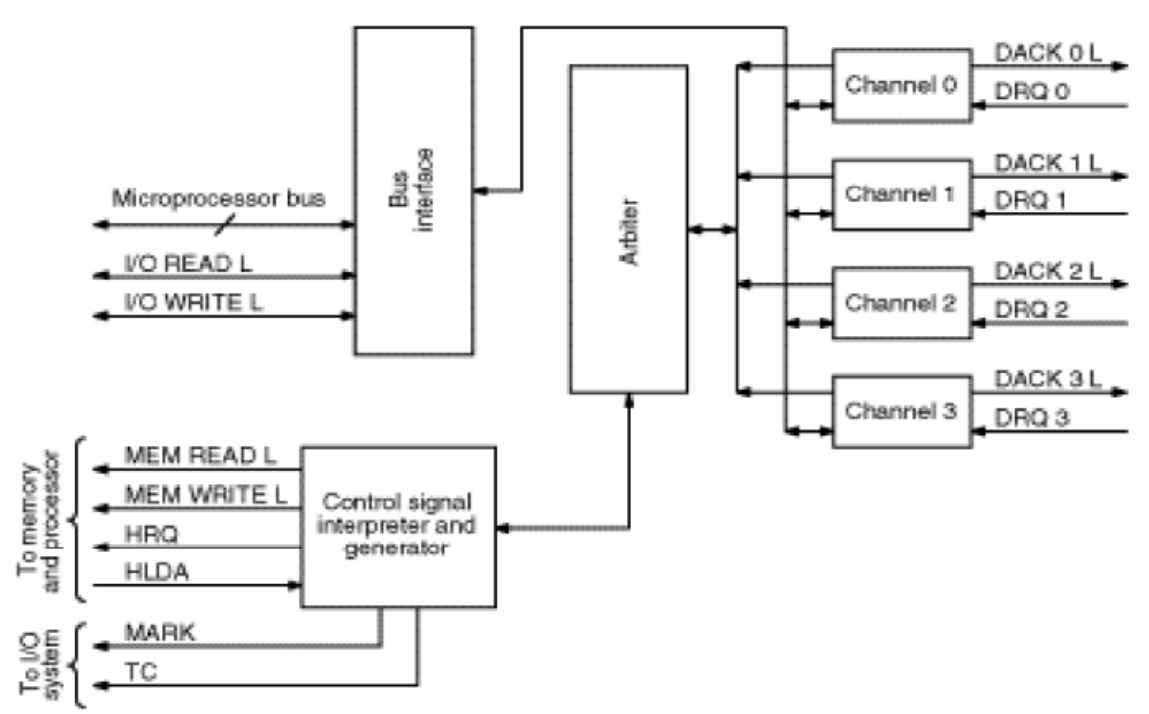


Figure 5-4: Structure of the i8527 DMA controller

The chip has four signals associated with the READ and WRITE operation. MEM READ L and MEM WRITE L are signals produced by DMA controller to exercise memory. The two signals I/O READ L and I/O WRITE L are bidirectional, they are inputs from the microprocessor when the microprocessor sends commands to the 8257 and reads back the 8257 status. During the I/O operation these signals are output from the 8257 and are functionally opposite to the memory signals. The 8257 takes control of the bus by exercising HALT (HRQ) and receives back the "go-ahead" signal on HALT ACKNOWLEDGE (HLDA).

Two signals produced by the DMA controller can be used by the I/O port to assist in controlling the transfer process. One signal TC--terminal count--is asserted during the last cycle of a DMA block. This can be used to describe a DMA mode on an I/O port or to reset the port's internal state to indicate the end of a transfer. The second--MARK--is inserted when the remaining count on a channel became a multiple of 128--providing a convenient timing signal for an external device.

Interrupts 8086

The meaning of 'interrupts' is to break the sequence of operation. While the CPU is executing a program, an 'interrupt' breaks the normal sequence of execution of instructions, diverts its execution to some other program called ***Interrupt Service*** *Routine* (TSR). After executing TSR, the control is transferred back again to the main program which was being executed at the time of interruption.

Whenever a number of devices interrupt a CPU at a lime, and if the processor is able to handle them properly, it is said to have ***multiple interrupt processing capability.***

Need for Interrupt: Interrupts are particularly useful when interfacing I/O devices that provide or require data at relatively low data transfer rate.

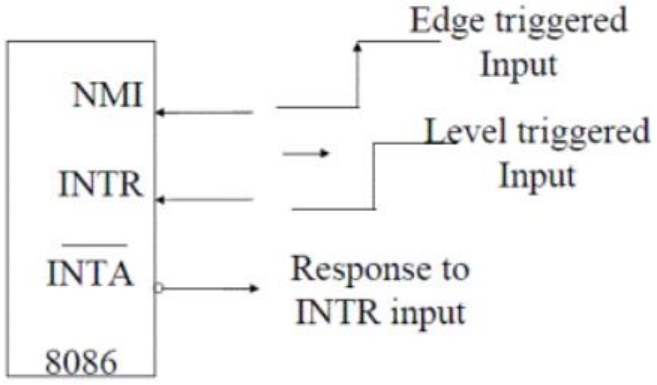
Sources of Interrupts in 8086: There are two pins for Interrupts in 8086. These are:

- *Hardware Interrupts (External Interrupts) – INTR, NMI*
- *Software Interrupts (Internal Interrupts and Instructions) – INT n instructions*

(i) Hardware Interrupts (External Interrupts)-

The Intel microprocessors 8086 **support hardware interrupts** through:

- Two pins that allow interrupt requests, -INTR and NMI
- However one pin that acknowledges, INTR is INTA.



INTR and NMI

1. **INTR** is a maskable hardware interrupt. The interrupt can be enabled/disabled using STI/CLI instructions or using more complicated method of updating the Interrupt Flag (IF).
2. The 1NTR, further, is of 256 types. The INTR types may be from **00 to FFH** (or 00 to *255).* If more than one type of INTR interrupt occurs at a time, then an external chip called programmable interrupt controller is required to handle them. The same is the case for INTR interrupt input of 8085.
    - When an interrupt occurs, the processor stores FLAGS register into stack, disables further interrupts, fetches from the bus one byte representing interrupt type, and jumps to interrupt processing routine address of which is stored in location
        4 * <interrupt type>

    Interrupt processing routine should return with the IRET instruction.

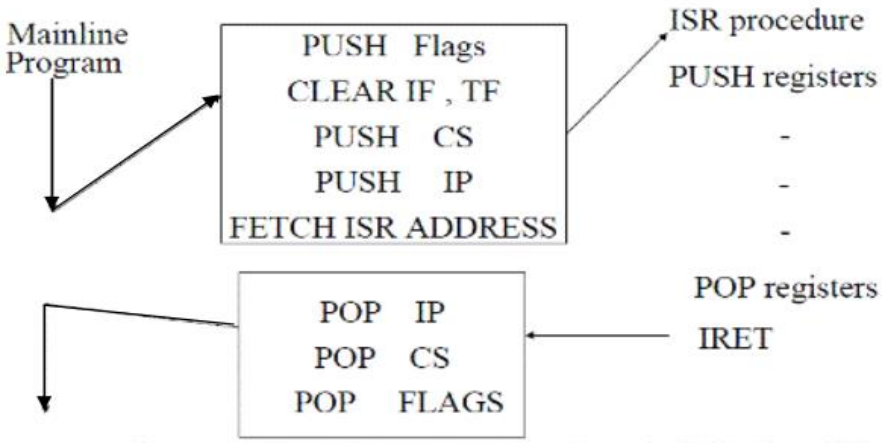
3. **NMI** is a non-maskable interrupt which means that any interrupt request at NMI input cannot be masked or disabled by any means.
    - This Interrupt is processed in the same way as the INTR interrupt.
    - Interrupt type of the NMI is 2, i.e. the address of the NMI processing routine is stored in location 0008h.
    - This interrupt has higher priority than the maskable interrupt.
        Ex: NMI, INTR.

(ii) Software Interrupts (Internal Interrupts and Instructions)-

Software interrupts can be caused by:

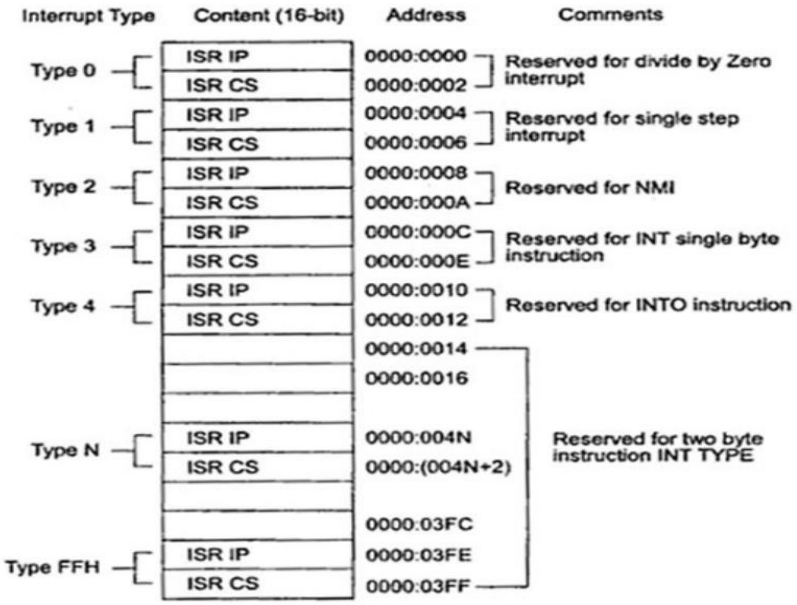
- INT instruction - breakpoint interrupt. This is a type 3 interrupt.
- INT <interrupt number> instruction - any one interrupt from available 256 interrupts.
- INTO instruction - interrupt on overflow
- Single-step interrupt - generated if the TF flag is set. This is a type 1 interrupt. When the CPU processes this interrupt it clears TF flag before calling the interrupt processing routine.
- Processor exceptions: Divide Error (Type 0), Unused Opcode (type 6) and Escape opcode (type 7).
- Software interrupt processing is the same as for the hardware interrupts.
- - **Ex: INT n** (Software Instructions)
- **Control is provided** through:
    - IF and TF flag bits
    - IRET and IRETD

Action taken when Interrupts **ocurres**



7. It decrements SP by 2 and pushes the flag register on the stack.
8. Disables INTR by clearing the IF.
9. It resets the TF in the flag Register.
10. It decrements SP by 2 and pushes CS on the stack.
11. It decrements SP by 2 and pushes IP on the stack.
12. Fetch the ISR address from the interrupt vector table.
13. After executing ISR (i.e., when IRET invoked) then IP, CS and Flag register content is popped so SP will be decremented gradually.

Interrupt Vector Table



- Every external and internal interrupt is assigned with a type (N), that is either implicit (in case of NMI, TRAP and divide by zero) or specified in the instruction INT N (in case of internal interrupts).
- In case of external interrupts, the type is passed to the processor by an external hardware like programmable interrupt controller.
- The 8086 supports a total of 256 types of the interrupts. i.e. from 00 to FFH. Each interrupt requires 4 bytes. i.e. two bytes each for IP and CS of its TSR. Thus a total of 1024 bytes are required for 256 interrupt types, hence the interrupt vector table starts at location 0000:0000 and ends at 0000:03FFH.
- The interrupt vector table contains the IP and CS of all the interrupt types stored sequentially from address 0000:0000 to 0000:03FF H.
- The interrupt type N is multiplied by 4 and the hexadecimal multiplication obtained gives the offset address in the zeroth code segment at which the IP and CS addresses of the interrupt service routine (ISR) are stored.

  $IP = (4 \times n)_H$
  $CS = (4 \times n)_H + 2$; where n-type of interrupt
- The execution automatically starts from the new CS: IP.

Interrupt Type

Type 0 – Type 4 → Intel predefined
Type 5 – Type 31 →Reserved
Type 32 – Type 255 → User defined Maskable Interrupt

Functions associated with Type 0 – Type 4 → Intel predefined

INT Number	Physical Address
INT 00	00000
INT 01	00004
INT 02	00008
:	:
:	:
INT FF	003FC

Type 0 (divide error)

- It is invoked by the microprocessor whenever there is an attempt to divide a number by zero.
- ISR is responsible for displaying the message "Divide Error" on the screen.
- IP:00000, CS:00002

Type 1 (Trap or Single step)

For single stepping the trap flag must be 1.

- After execution of each instruction, 8086 automatically jumps to 00004H to fetch 4 bytes for CS: IP of the ISR.
- The job of ISR is to dump the registers on to the screen

Type 2 (Non maskable Interrupt)

- Whenever NMI pin of the 8086 is activated by a high signal (5v), the CPU Jumps to physical memory location 00008 to fetch CS: IP of the ISR associated with NMI.

Type 3 (Break point)

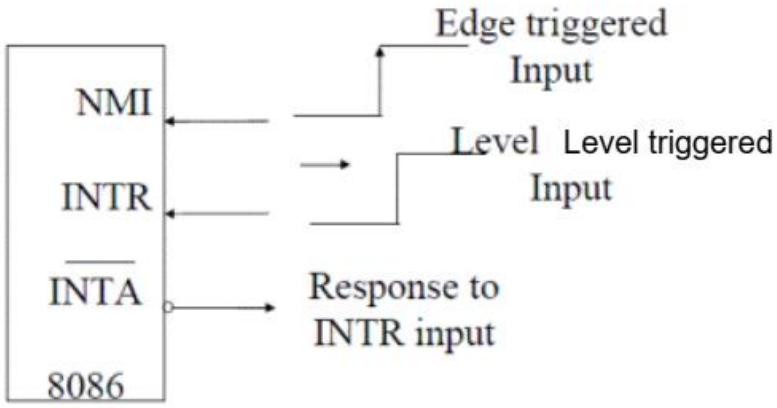
- A break point is used to examine the CPU and memory after the execution of a group of Instructions.
- It is one byte instruction whereas other instructions of the form "INT 3" are 2 byte instructions.

Type 4 (Signed number overflow)

- There is an instruction associated with this INT 0 (interrupt on overflow).
- If INT 0 is placed after a signed number arithmetic as IMUL or ADD the CPU will activate Type 4 if 0F = 1.
- In case where 0F = 0 , the INT 0 is not executed but is bypassed and acts as a NOP

Performance of Hardware Interrupts

- NMI : Non maskable interrupts - TYPE 2 Interrupt
- INTR : Interrupt request - Between 20H and FFH



Interrupt Priority Structure

Interrupt	Priority
Divide Error, INT(n),INTO	Highest
NMI	↓
INTR	↓
Single Step	Lowest