THE HARDWARE INTERFACE

- The idea behind the UART is to relieve both the programmer and the processor of the toil associated with **Asynchronous Serial I/O.**

- To receive and send data, the program simply reads and writes bytes to the UART, which appears to the processor as one or more ordinary memory locations or I/O ports.

Polling versus Interrupts

Polling

- There are two distinct disadvantages with this technique.

- The first problem is the processing power that is wasted by a computer that continually scans the status byte.

- If this computer is dedicated only to handling specific communications tasks at hand then this is not a problem.

- However, computers are often performing several tasks at the same time, especially with the introduction of co-resident programs and multitasking operating systems.

- The second problem with the polling approach is that a new character may be lost while the first is being processed.

- This depends on two factors: the speed at which characters are being received and the speed at which they are being processed.

Hardware Interrupts

- This technique transfers to the peripheral device itself the responsibility of notifying the CPU when an I/O operation is required.

- This notification takes the form of an interrupt request whereby the I/O device actively solicits the CPU's attention.

- When the CPU completes its current instruction, it decides if it will acknowledge the request for service.

- If so, it issues an interrupt acknowledgment, then turns program control over to a section of code, designed to handle the device's needs - the interrupt service routine or the interrupt handler.

- Once the ISR terminates, the CPU returns to the main program where execution resumes as if the interruption had never occurred.

Generating Interrupts

- The UART then must generate an interrupt under a variety of conditions:

- Transmitter Interrupts. An interrupt is generated when the transmitter buffer becomes empty - i.e., can accept another byte for serialization.

- Receiver Interrupts. An interrupt is generated when an assembled byte has been placed in the receiver's FIFO.

- **RS-232 Interrupts**: an interrupt can be generated by a change of state of any of the RS-232 input lines - CTS, DSR, DCD, RI.

- **Interrupt on Receiver Error or BREAK**: Generates an interrupt for any irregular Serialization condition: BREAK, parity, overrun, or framing.

- The UART architecture must have an Interrupt Enable register, which contains a bit to enable interrupts for each of the above conditions.

- It is frequently necessary to disable and re-enable UART interrupts entirely at various points in program execution.

- After the UART generates an interrupt, it will not generate another one until the first one has been "cleared".

- The most common way of clearing the interrupt is to service its cause. For example, if the interrupt was generated by RxRDY, reading the available character clears the interrupt.

UART Hardware: The National 8250/16550

- The 16550 has replaced the 8250. It has a faster I/O bus but both are identical functionally.

- The 8250 requires three basic interfaces:
    - The system I/O bus
    - The clock
    - RS-232 I/O

- The 8250 is connected to the low-order 8 bits of the CPU's data bus by means of the data lines D0 - D7. This is the path in and out of the UART.

- Read and write operations are differentiated by the data input and output strobe lines DISTR and DOSTR.

- The 8250 comprises several internal registers, all which are individually addressable by means of three register select inputs A0-A2.

- Transmitting a byte is a three step operation:

    1. CPU places the outbound data byte on the 8-bit data lines D0-D7.

    2. The register number of the Transmitter Buffer register is placed on the register select inputs A0-A2.

    3. The logic on the data strobe lines DISTR and DOSTR moves the byte from D0-D7 into the transmitter's buffer.

    - The 8250 moves the byte from the Buffer register to the transmitter's shift register when the latter is empty.

- Assuming that an inbound byte has been received and is waiting in the receiver buffer register, the step are as follows:

    1. The Receiver Buffer register's number is placed on register select inputs A0-A2.

    2. A read operation is performed by the logic on the data strobe lines DISTR and DOSTR.

    3. The byte moves from the receive buffer to D0-D7 where it is captured by the CPU.

- The final point of interface with the I/O bus is the interrupt line (INTRPT).

- This line is set to TRUE (logical 1) whenever a condition exists for which the 8250 is programmed to generate and interrupt.

8250 Clock and Timing

- The 8250's reference clock is injected at the XTAL1 input, where it passes through a user-programmable divider circuit to produce a master data clock.

- This signal is 16 times higher than the desired baud rate, thus fixing the clocking factor at 16.

- This is done to prevent false start bit detection.

8250 Internal Architecture

- A program exerts control over the 8250 by reading and writing ten registers.

- Except for the Interrupt Identification register, which is read-only, data written to a register can be read back.

Register Addressing on the 8250

- The 8250 has to address 10 registers using only 3 physical register select lines.

- In order to extend the addressing the following technique is used: when bit 7 of the Data Format register is 1, registers 0 and 1 become the low and high-order bytes of the Baud Rate Divisor Latch.

Registers of the UART

- Like the CPU chip, the UART contains registers, or internal memory locations. There are three types of registers:
    - Buffer registers, which receive commands from the CPU.
    - Status registers, which are used to inform the CPU of what is going on in the UART.
    - Buffer registers, which hold characters pending transmission or processing.

- The way registers are accessed depends on the architecture of the computer in which the 8250 is installed.

- Values to be placed in the registers are sent to an appropriate I/O address by means of and OUT (Assembly language) command.

- Registers to be read are accessed by means of an IN instruction accompanied by the appropriate address.

8250 Register Usage Summary

- The numbers preceding each register name are the offsets to be added to the base address of the COM port when accessing the registers.

0: Receiver Buffer Register

- After a stream of bits on the serial input line (SIN) has been assembled into a byte, reading this register fetches the byte.
- When bit 7 of the Data Format register is TRUE, this register becomes the LSB Baud Rate Divisor Latch.

0: Transmitter Buffer Register

- Writing a byte to register 0 results in its serialization and transmission at the serial output line (SOUT) in the current data format and the baud rate.
- When bit 7 of the Data Format register is TRUE, this register becomes the LSB Baud Rate Divisor Latch.

1: Interrupt Enable register

- It is possible to instruct the 8250 to generate an interrupt signal whenever certain events occur.
- This register is used to tell the 8250 (and hence the CPU) which events should cause interrupts.
- This register is used to enable four types of interrupts:
    - Bit 0 : Data available for reading from the Receiver Buffer register.
    - Bit 1 : Transmitter Holding register empty (TBE). The 8250 can accept another byte for transmission.
    - Bit 2 : Receiver Line status. A result of parity error, overrun error, framing error or BREAK.
    - Bit 3 : Modem status. Generate an input when any of the RS-232 inputs changes states.
    - Bits 4-7 : Always zero.

2: Interrupt Identification register

- Provides information about the current status of the pending interrupts.
- Bit 0 is set to TRUE if there is no pending interrupt.
- If bit 0 is FALSE (logical 0), bits 1 and 2 indicate which interrupt is pending as follows:

Bit 2	Bit 1	Interrupt pending
1	1	Line status (error or BREAK)
1	0	Received data available
0	1	Transmitter Buffer Empty
0	0	Modem status (RS-232)

- Each interrupt has a priority (top to bottom above): while an interrupt is pending, interrupts with equal or lower priority are locked out.

3: Line Control register (Data Format)

- Used to set the communications parameters. The meanings of each bit in the register are as follows:

Bit 0 : Word length: least significant bit

Bit 1 : Word length: most significant bit

Bit 2 : Stop bits

Bit 3 : Parity enable

Bit 4 : Parity select

Bit 5 : Parity one. Makes the parity bit a logical zero if TRUE. If FALSE, the parity bit becomes a logical one.

Bit 6 : BREAK

Bit 7 : Divisor Latch Access Bit (DLAB). Used to extend the number of registers that can be addressed with 3 lines.

4: Modem Control (RS-232 Output Control)

- Controls the handshaking signals sent out from the UART. The functions of the bits are as follows:

  Bit 0 : Data Terminal Ready (DTR)

  Bit 1 : Request To Send (RTS)

  Bit 2 : GPO1. User defined output 1.

  Bit 3 : GPO2. User defined output 2.

  Bit 4 : Test mode loop-back.

5: Line Status Register (Serialization status)

- Used to obtain information concerning the receipt (assembly) and transmission (serialization) of data.
- The conditions reported by bits 1 - 4 generate an interrupt when bit 2 is set in the Interrupt Enable Register.
- The bit functions are as follows:

  Bit 0 : Data Ready. An incoming character has been received and placed in the receiver buffer.

  Bit 1 : Overrun error. A character has been received before the last one was removed for processing.

  Bit 2 : Parity error. Incorrect parity on an inbound character.

  Bit 3 : Framing Error. A received character does not have a valid stop bit.

  Bit 4 : Break Interrupt. The received data input line is zero for more than the maximum character length.

  Bit 5 : Transmitter Buffer Empty. UART is ready to receive a new character to transmit.

  Bit 6 : Transmitter shift register empty. No bytes in the transmitter buffer or in the transmitter's shift register.

  Bit 7 : Always zero.

6: Modem Status register (MODEM status)

- Provides about the status of the handshaking lines. A TRUE in any of these bits means that its input has changed since the last read.

- The bit functions are as follows:

  Bit 0 : Delta CTS. Clear to send line has changed.

  Bit 1 : Delta DSR. Data set ready line has changed.

  Bit 2 : Delta RI. Ring Indicator has changed from on to off.

  Bit 3 : Delta DCD. Received line Data Carrier detect has changed.

  Bit 4 : CTS. Clear to send input is high (OK).

  Bit 5 : DSR. Data set ready input is high (OK).

  Bit 6 : RI. Ring indicator is high.

  Bit 7 : DCD. Received carrier detect is high.

7: Scratch Pad.

- This register is unused.

8: LSB Baud Rate Divisor Latch
9: MSB Baud Rate Divisor Latch

- The 8250's reference clock is divided by the 16-bit integer contained in the LSB and MSB Divisor Latch registers.

- The resulting frequency is the master data clock that drives the transmitter logic and optionally the receiver logic.

- The master data clock is then again divided by 16 to produce the baud clock, which controls the speed at which data is received and transmitted.

- The divisor for any baud rate can therefore be calculated by the formula:

$$Divisor = \frac{\text{Reference Clock Frequency}}{16 * \text{Desired Baud Rate}}$$

- The table given shows the divisors (in hex) for the most popular baud rates. Note that the UART uses a crystal frequency of 1.8432 MHz.

Interrupts on the 8250

- The 8250's sole response t an interrupt is to assert its INTRPT line to inform the CPU that an interrupt has occurred.

- The CPU has to determine the address of the interrupt handler (not the 8250) and turns control over to it.

- When the interrupt handler receives an interrupt it initially only knows that the 8250 caused the interrupt, nothing else.

- It first examines the interrupt identification register to see what caused the interrupt.

- The contents of this register are converted to the address of the appropriate subfunction to service the pending interrupt.

- For example, the IIR might indicate Data Ready, meaning that a character has been received, in which case it is read from the receiver buffer register.

- The interrupt could also indicate that the transmitter holding register is empty, in which case the next character to be sent, if any, is placed in the transmitter holding register.

16550 UART

- This device is functionally compatible with both the 8250 and 16450 UARTS. It has some major enhancements such 16-byte transmit and receive FIFO queues.

- It is capable of receiving 15 extra characters before an overrun error occurs. This also implies that a communications program can now send 16 bytes at a time instead of 1.

- The Interrupt Identification Register provides information about the interrupt.
  - Bit 3 : Set to 0 when the FIFO is disabled. It is set to 1 to indicate timeout generated transmit or receive interrupt.
  - Bit 6, 7 : Set to 0 when the FIFO is enabled, 1 otherwise.

  - The FIFO Control register is accessed at offset 2. This is register controls the transmit and receive FIFOs built into the 16550.

  - Using this register you can enable the FIFOs, clear them, and set the FIFO interrupt-trigger level.

    Bit 0 : Set to 1 to enable both the transmit and receive FIFOs.

    Bit 1 : Set to 1 to clear all the bytes from the receiver FIFO.

    Bit 2 : Set to 1 to clear all the bytes from the transmit FIFO.

    Bit 3 : Set to 1 to support DMA .

    Bit 4, 5 : Unused.

    Bit 6, 7 : FIFO Trigger Level.

Bit 7	Bit 6	Trigger Level
0	0	1
0	1	4
1	0	8
1	1	14