Title of the paper is “Microcontroller Architecture and Programming”.

Name of the module is “Serial data I/O and Interrupts”.

Outline of the module:

1. Serial data Input /Output
2. Interrupts

Learning Outcomes of this module:

At the end of the module, learner will be able to:

- Describe the various operating modes of the UART, and associated control registers.
- List the types of interrupts, the interrupt program addresses, and the interrupt control registers.

1. Serial Data Input /Output

Computers, must be able to communicate with other computers. One cost-effective way to communicate, is to send and receive data-bits serially. The 8051 has a serial data communication circuit, that uses:

I. Serial port buffer – register,(SBUF) to **hold** data,

II. Serial Port Control Register (SCON), to controls data communication,

III. POWER CONTROL register (PCON), to controls data rates, and

IV. Two pins, RECEIVE DATA (RXD i.e. Port 3.0) and TRANSMIT DATA (TXD i.e. Port 3.1), which connects to the serial data network.

Serial Port Buffer Register (SBUF), which is a SFR, is physically two registers. One is “write” only, and is used to hold data, to be transmitted out of the 8051 via TXD i.e. TRANSMIT DATA pin of the 8051. The other is “read” only, and used to holds the received data, from external sources via RXD i.e. RECEIVE DATA pin of 8051.

Both the registers mutually use the address **99H.**

There are four programmable modes for serial data communication that are chosen by setting the SM0 & SM1 bits in SERIAL PORT CONTROL register (SCON).

7	6	5	4	3	2	1	0
SM0	SM1	SM2	REN	TB8	RBS	TI	RI

The Serial Port Control (SCON) Special Function Register

Data Transmission

Transmission of serial data bits, begins anytime, the data is “written” to, SERIAL PORT BUFFER register (SBUF). TRANSMIT INTERRUPT flag (**TI),** of SERIAL PORT CONTROL REGISTER (SCON) is set to bit “ 1” when the data has been transmitted and signifies, that SERIAL PORT BUFFER register (SBUF) is empty for transmission purposes, and that another data byte, can be sent.

If the program fails, to wait for the TRANSMIT INTERRUPT flag (**TI)** of SERIAL PORT CONTROL Register (SCON) and send the data to, SERIAL PORT BUFFER register (SBUF) while a previous data byte,

is in the process of being transmitted, the results will be unpredictable. A polite term for this action, is "garbage out".

Data Reception

Reception of serial data, will begin, ***if*** the RECEIVE ENABLE bit (REN) in SERIAL PORT CONTROL register (SCON) is set to “1” . In addition, RECEIVER INTERRUPT flag (RI) in SCON register is set after data has been received.

When, RI flag is made = to “0”, prevents the reception of new data until the program has dealt with the old data and thereafter reset RI flag is set to “1” for new reception.

RI flag, must have been reset by the program before the last bit is received; or the incoming data will be lost. Incoming data is not transferred to, SERIAL PORT BUFFER register (SBUF) until the last, data bit has been received, so that the previous transmission can be read, from SERIAL PORT BUFFER register (SBUF) while new data is being received.

Serial Data Transmission Modes

The 8051 designers, have included four modes of serial data transmission, that enable data communication to be done in a variety of ways and a multiples of baud rates. Modes are selected by the programmer by setting the mode bits, SM0 and SM1 in SERIAL PORT CONTROL register (SCON)

SM0	SM1	Mode	Description
0	0	0	Shift register, baud - f/12
0	1	1	8-bit UART, baud variable
1	0	2	9-bit UART, baud - f/32 or 1/64
1	1	3	9-bit UART: baud - variable

I. Serial Data Mode 0

Setting bits, SM0 and SM1 in SCON register to 00 binary, configures SERIAL PORT BUFFER register (SBUF) to receive or transmit eight data bits using pin RXD (Receive Data) for both receive & Transmit function. Pin TXD (TRANSMIT-DATA) is connected to the internal shift frequency pulse source, to supply shift pulses to external circuits. The shift frequency, or baud rate, is fixed at 1/12 of the oscillator frequency.

While transmitting, data is shifted out of pin RXD sequentially. The data changes, on the falling edge of the clock pulse. The system designer must design the external circuitry, that receives this transmitted data to receive, the data reliably based on this timing.

Received data, comes in on, pin RXD and should be synchronized with the shift clock pulse.

Data is shifted, in to SERIAL PORT BUFFER register (SBUF) on, the rising edge of the shift clock.

Mode 0 is *not* intended, for data communication, between computers, but as a high-speed, serial data-collection method using discreet logic to achieve high data rates. The baud rate used in mode, 0 will be much higher, than standard rate.

For Example, for a 6 megahertz crystal, the shift rate will be 500 kilohertz.

ii. Serial Data Mode 1- Standard UART

When SM0 and SM1 are set to 01b, SERIAL BUFFER register (SBUF), becomes a 10-bit full-duplex, receiver/transmitter, that may receive and transmit data at the same time. Pin RXD receives all data, and pin TXD transmits all data.

Figure 2 shows the format data word

1	2	3	4	5	6	7	8	9	10
Start bit	**LSB 0**	**1**	**2**	**3**	**4**	**5**	**6**	**MSB 7**	**Stop bit**

Transmitted data is sent, as a start bit as first bit, than, eight- data bits with, least significant bit, (LSB) being the first i.e. It becomes the second bit, then the rest of the bits and a stop bit as the 10^{th} bit. Interrupt flag (TI) in SCON register is set once all ten bits have been sent.

Each bit interval is the inverse of the baud rate frequency, and each bit is maintained high or low, over that interval.

Received data is obtained in the same order; reception is triggered by the falling edge of the start bit and continues, **till** the stop bit is true (0 level). Of the original ten bits (received), the start bit is discarded; the eight data bits, go to SBUF, and the stop bit is saved in bit RB8 of SCON register. RI bit of SCON register is set to 1, indicating a new data byte has been received.

Mode 1, 2 & 3 will be covered, in the later modules.

2 . Interrupts

A computer program, has only two ways, to "know" the conditions, that exist in internal, and external circuits, of 8051.

i) One method uses software instructions, that jump to subroutines on the states of flags and port pins.

ii) The second method responds to hardware signals, called interrupts, that force the program to call a subroutine.

Software techniques, use up, the processor time that could be devoted to other tasks; interrupts, take processor time only when action by the program is needed. As most of the applications of microcontrollers involve responding to events/ situations quickly enough to control the environment that generates the events (generically termed as, real-time programming).

To give an example: If there is a fire, then fire sensor will have to activate the microcontroller and this then, will do the necessary action like, putting the alarm, sending the SMS to the owner, fire service agencies etc.}. Interrupts, are often the only way in which real-time programming can be done successfully.

Interrupts may be generated, by internal chip operations or may be provided, by external sources. Any such interrupt, can cause, the 8051 to perform a hardware call to an interrupt-handling subroutine that is located, at a predetermined address, in program memory. (This is set by the 8051 designer. and not the programmer)

SIX interrupts are provided in the 8051.

- Three of these are generated, automatically by internal operations: Timer flag 0, Timer flag 1, and the serial port interrupt (RI or TI).
- Two interrupts, are triggered by external signals, provided by circuitry, that is connected to pins INT0 and INT1 (port pins P3.2 and P3.3) as shown in fig.
- Reset

All interrupt functions are under the control of the program. The programmer is able to alter “control bits” in the Interrupt Enable register (IE), the Interrupt Priority register (IP), and the Timer Control register (TCON). {in SFR}. The program can block all or any

combination of the interrupts from acting on the program by suitably setting or clearing bits in these registers.

After the interrupt has been handled, by the interrupt subroutine, which is placed by the programmer, at the interrupt location in program memory, the interrupted program must resume operation at the instruction where the interrupt took place. Program resumption, is done by storing the interrupted, PROGRAM COUNTER (PC) address on the stack, in RAM before changing the PC to the interrupt address in ROM. The PC address will be restored from the stack, after an **RETI** (Return Interrupt) instruction is executed at the end of the interrupt subroutine.

Timer Flag Interrupt

When a timer/counter overflows, the corresponding Timer flag, TF0 or TF1, is set to “1”. The flag is cleared to “0” , when the resulting interrupt generates a program call to the appropriate timer subroutine in memory.

Example: Suppose every after every 1 sec, the timer T0 overflows. This will set TF0 to “1” and interrupt handler, directs to serve the routine to give a beep sound. Thus a beep will be generated every after 1 sec.

Serial Port Interrupt

- If a data byte is received, an interrupt bit, RI, is set to 1 in the SCON register.
- When a data byte has been transmitted, an interrupt bit, TI, is set in SCON register.

The program that handles serial data communication must reset RI or TI to 0 to enable the next data communication operation.

External Interrupts

Pins INT0 and INT1 are used by external circuitry. Inputs on these pins can set the Interrupt flags IE0 and IE1 in the TCON register to "1" by two different methods.

i) The IEX flags may be set when the INTX pin signal reaches a low level, or OR

ii) The flags may be set when a high-to-low transition takes place on the (INT0 OR INT1) pin.

Bits IT0 (interrupt 0) and IT1 in TCON – TIMER CONTROL Register, is used to program these two method.

Flags (IE1 & IE0) will be reset when an interrupt is accepted by the processor and the interrupt -subroutine is, accessed.

It is the responsibility of the system designer and programmer to reset any-level, generated external interrupts, after they are serviced by the program. The external circuit must remove the interrupt signal,

(i.e. Voltages or pulses) before an RETI is executed. Failure to remove, will result in an immediate interrupt after RETI, from the same source.

Reset

A reset can be considered, to be the ultimate interrupt, because the program may not block the action of the voltage, on the RST pin. This type of interrupt is often called, *non-maskable,* because no combination of bits in any register can stop, or mask, the reset action. Unlike other interrupts, the PC- programme counter is not stored for later program, resumption; a reset is an absolute command to jump to program address 0000H and commence running from start.

Whenever a high level, is applied to the RST pin, the 8051 enters a reset condition. After the RST pin is brought low, most of the internal registers will have the default values.

The books referred for this module are shown on the display.

Thank you very much.