

Microprocessors and Microcontrollers
Prof. Santanu Chattopadhyay
Department of Electrical & Electronics and Communication Engineering
Indian Institute of Technology, Kharagpur

Lecture – 35
8051 Microcontroller (Contd.)

So, in this example that we have considering.

(Refer Slide Time: 00:20)

Example Interrupt Service Routine

Pin 3.3 (INT1) is connected to a pulse generator. Write a program in which the falling edge of the pulse will send a high to P1.3, connected to a LED.

```

org 0000h ←
ljmp main
; ISR for hardware interrupt INT1
org 0013h
setb p1.3 ←
mov r3, #255 ←
back: djnz r3, back
      clr P1.3
      reti
; Main program for initialization
org 30h ←
main: setb tcon.2 ; make INT1 edge triggered
      mov ie, #0000000B ; enable int1
here: sjmp here

```

The diagram illustrates the hardware setup where a pulse generator is connected to pin P1.3 of the 8051 microcontroller. This pin is also connected to an LED. A memory map on the right shows the instruction LJMPL main at address 0000h and the start of the interrupt service routine (ISR) at address 0013h. The ISR code is shown as a sequence of instructions: setb p1.3, followed by a loop (djnz r3, back) that counts down from 255, and finally clearing P1.3 and returning (reti). The main program starts at address 30h, where it configures the timer control register (tcon.2) for edge triggering and enables the interrupt (ie) by setting bit 0 of the interrupt enable register (ie).

So, this 8051 port 3.3 this pulse generator is connected and on 1.3 line, we have got the LED. Now so, this so at this o r g 0000h, so you know that this is the instruction this is an assembler directory that tells the assembler that this assembly process should put the code from this address onwards.

So, at the location 000 h; so, if you look into this memory map; so, what we do? At this at this position this is the address 0000 there we are putting the instruction LJMP main. So, this is the instruction LJMP main that is put here. So, your main is located; so, your main program will start from 30 h onwards. So, this is the 30 h this point onwards the program will be loaded. So, there I have got this 8 bit contour; so, the so, this is there. So, this is the set B instruction is there.

So, when the processor will start it will find this LJMP and it will come to this 30 h point and from that point it will start executing. So, apart from that we have said that this connected to the external interrupt INT1 ok.

(Refer Slide Time: 01:52)

Interrupt Vectors

Each interrupt has a **specific** place in **code** memory where program execution (interrupt service routine) begins.

Reset:	0000h
External Interrupt 0:	0003h
Timer 0 overflow:	000Bh
External Interrupt 1:	0013h
Timer 1 overflow:	001Bh
Serial :	0023h
Timer 2 overflow (8052+)	002bh

Note: that there are only 8 memory locations between vectors.

Now in INT1; so, you need to find out what is the interrupt address of this INT1. So, INT1 is 0013; so, this is the vector address. So, I have to put the corresponding ISR in the you have to put the corresponding ISR in the vector address 1 3; so, this is done. So, this is my interrupt service routine in the in the in the interrupt service routine what I will do? I will be I will be glowing this LED on I will be glowing this LED on and accordingly I will be turning it on I will wait for some delay then I will be turning the LED.

So, whenever this interrupt will occur from this source; whenever this interrupt will occur from this source I will glow this i will put a 1 here. So, the LED is turned on wait for 255 cycles by this is small delay look software delay look is there which will be putting a delay; then I will turn off the delay and then it will return. So, how this whole thing is done? Is that you see we are. So, in the main program what we are doing we are making this set b tcon point tcon dot 2. So, tcon dot 2 it will make this INT1 edge triggered. So, tcon register; so, we had got that interrupt types that. So, that type is set to 2; type is set to s triggering by setting that b to 1.

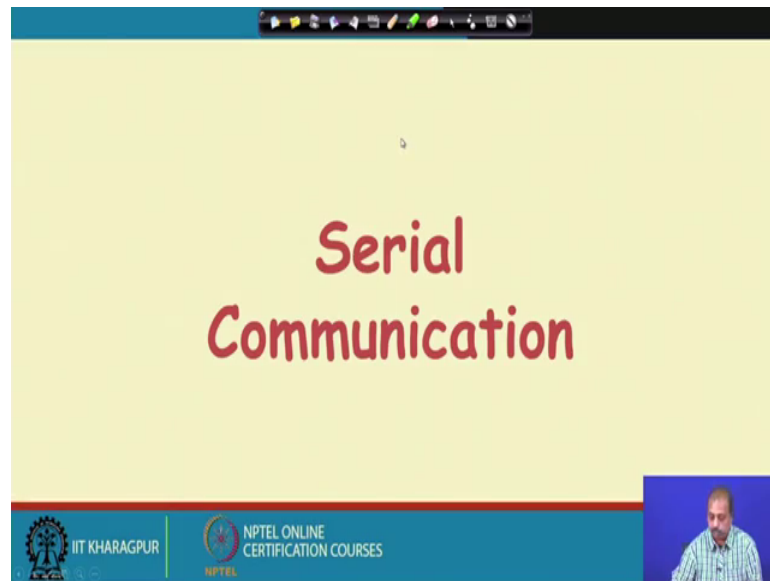
And this interrupt enable register; so, we are enabling the interrupt slight e a b set to 1. So, this part is setting that e a bit. So, this is setting of that e a bit and this is the enabling of the external interrupt. So, rest of the interrupts are disabled; so, in the main program we just select the type of this interrupt 1 and we select and we enable the interrupt 1. And then where the main program it waits in an infinite loop sjmp because main program has got nothing more to do.

Whereas, now the timer has been enabled; so, from the outside world this pulses will be generated and whenever this edge triggering will occur low to high transition will occur. So, this will be the processor will get an interrupt; on getting this interrupt this ISR will be this since it is an INT1. So, processor will automatically jumped to the address 0013.

So, so 0013 is somewhere here; so, this is the address 0013. So, there actually I will have all this instruction the set B P P 3 P 1.3. So, this instruction will be there; so, this will be starting from this point. So, there it will be executing this as a result this LED will be turned on, it will wait for some delay here and then the LED will be turned off and then it will be returning from interrupt.

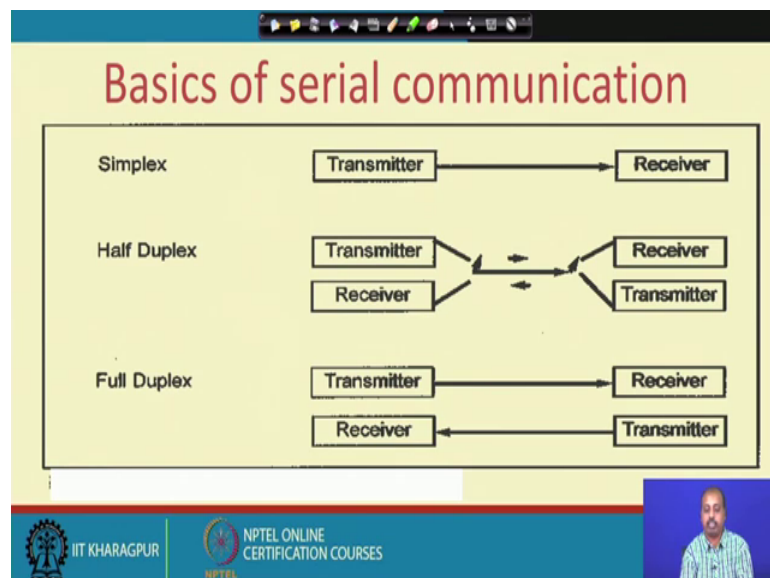
Again; so, it will be returning to this point only it is the main program will program will continue waiting here. Then after sometimes again the next falling edge will come and again the system will get another interrupt and when it gets another interrupt again the same the interrupt service routine will be invoked. So, it will come to this point it will continue from that point onwards. So, this say this the interrupt service routines the internal external interrupt service routines can be interrupt sources can be integrated with the system and we can have the service routines for them for getting various jobs done ok.

(Refer Slide Time: 05:27)



So, next we look into the serial communication. So, in case of 8085 we have seen the serial communication. So, in case of 8051 also serial communication is possible and we will see that how this is done like there is a there are some dedicated pins for that purpose as well; so, how to do the serial communication in 8051.

(Refer Slide Time: 05:51)



So just to recapitulate; so, we have got The basic serial communication modes like simplex, half duplex and full duplex in simplex mode the transmission is in one direction from transmitter to receiver in case of half duplex. So, this transmission can be in both

the direction, but at one point of time transmission is in one direction only; so, when it is connected to. So, once; so, that there is shown by this switches.

Once this connection is established; so, it will the transmitter to receiver. After sometime the direction will be reversed and now it will be transmitter to receiver. So, it can be both way, but at one point of time only one direction the information will flow. On other hand in case of full duplex we have separate transmission and receiving time lines. So, transmission can take place in both the directions simultaneously; so, that is the full duplex type of communication.

(Refer Slide Time: 06:48)

Start and stop bits

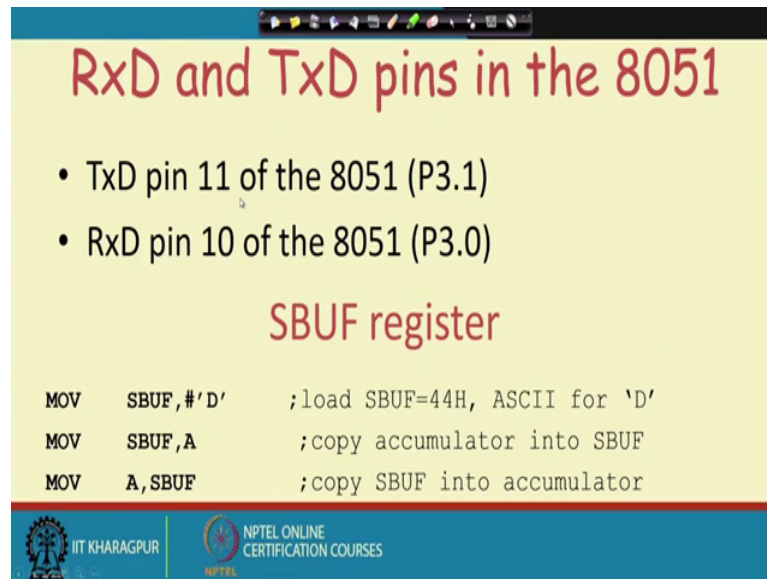
- When there is no transfer the signal is high
- Transmission begins with a start (low) bit
- LSB first
- Finally 1 stop bit (high)
- Data transfer rate (baud rate) is stated in bps
- bps: bit per second

The diagram illustrates the bit sequence: space (high), stop bit (high), data bits d7 (0), d6 (1), d5 (0), d4 (0), d3 (0), d2 (0), d1 (0), d0 (1), start bit (low), and mark (high). Arrows indicate the direction of transmission, and labels show that d7 goes out last and d0 goes out first.

So, when there is any serial communications they follow these type of protocol when no communication is taking place. So, there is we have the signal line is high and that we have got when the transmission starts the first we send a low bit. So, that is called the that is the low bit is there that is LSB and then after that the bits are transmitted.

So, in this diagram the start bit is the low bit and that is the start of the transmission and then we are sending the 8 bit of data d 0 to d 7 and then one or more stop bits may be set. And we have some bits per second the bps setting because this is mostly asynchronous transmission. So, we need to set the bit rate ok; so, at what rate the transmission will take place that bit rate has to be set.

(Refer Slide Time: 07:44)



RxD and TxD pins in the 8051

- TxD pin 11 of the 8051 (P3.1)
- RxD pin 10 of the 8051 (P3.0)

SBUF register

```
MOV    SBUF,#'D'    ;load SBUF=44H, ASCII for 'D'  
MOV    SBUF,A       ;copy accumulator into SBUF  
MOV    A,SBUF       ;copy SBUF into accumulator
```

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

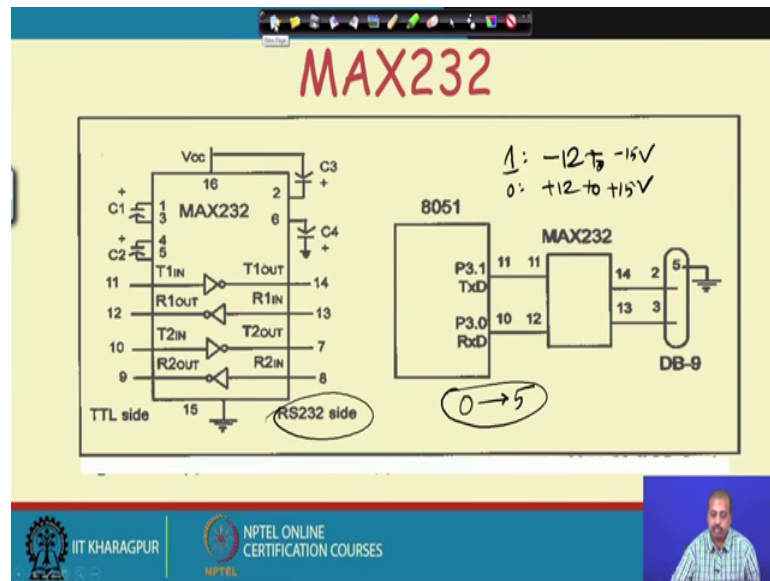
In case of 8051; we have got two dedicated pins receive data and transmit data R x D and T x D. So, these are the two pins in 8051 that we have.

So, they are again multiplex ok. So, unlike a 8085 where we have got this S I D and S O D in which dedicated for this serial communication; here it is not. So, here this pin 3.0 is dedicated is multiplexed with R x D and pin 3.1 is multiplexed with T x D and for transmission. And there is a special register called SBUF register; so, SBUF is the serial buffer you can say. And for transmitting any data you have to put the content into the SBUF register; as soon as you can you put content to the SBUF register the serial communication starts.

So, you can have an instruction MOV SBUF comma hash D hash this character D. So, this is SBUF will be loaded with a SBUF value of d which is 44 h or you can say like MOV SBUF comma A. So, accumulators content accumulator content will be copied onto SBUF and whatever be the value A; so, that will be transmitted.

You can get the copy of this SBUF into A so, that way. So, we can read the content of SBUF buffer in to the buffer register into the A register you would see what it is something is been received and something has been; so, you can do that.

(Refer Slide Time: 09:21)



Now, there is another problem like in case of 8051; so, this is the TTL chip. So, what happens is that this receive and transmit lines that we have. So, that is pin 3.1 and 3.0 they will give you the TTL level output. So, TTL level output means; so, this signal values are in the range of 0 to 5 volt.

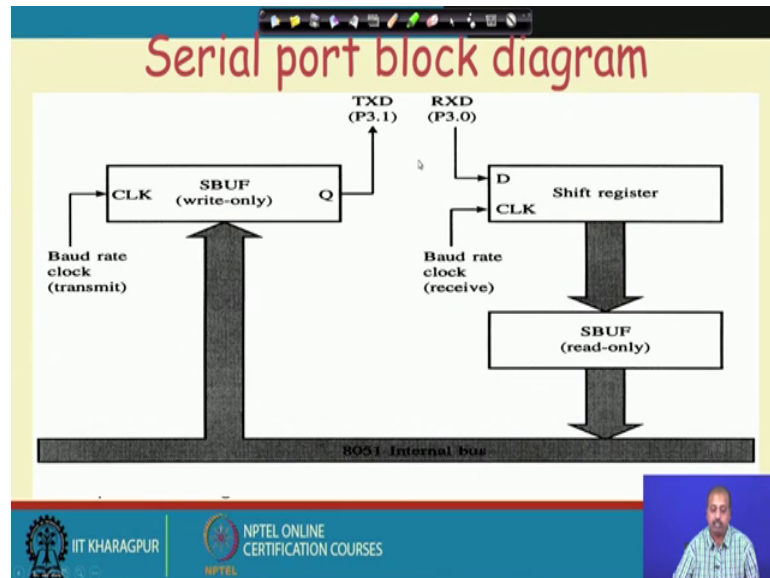
So, it is a there is a some definite high and low level. So, the accordingly the signal value cannot be the low cannot be lower than 0 and high cannot be more than 5. On the other hand this serial communication protocol likes is RS232 C type of protocol. So, they have got different voltage levels for logic high the voltage level is minus 12 volt to minus 15 volt minus 12 to minus 15 volt and for logic 0 it is a plus 12 to plus 15 volt.

So, you see that this 8051 chip it will not be able to provide this type of voltage levels. So, normally what is done is that we have got a for serial communication interface we have got another chip which is known as MAX232. So, this one this is this left side is TTL compatible on the right side is RS232 C compatible. So this 8051 is connected to this MAX232 pin. So, pin number 11 of this of this 8051 is connected to a pin number 11 of MAX232 then pin number 10 of 8051 connected to pin number 12 of MAX232.

Similarly, on the output type you can have some connector to which this is RS232 site. So, whatever device you are going to connect; so, we can design a suitable connector and

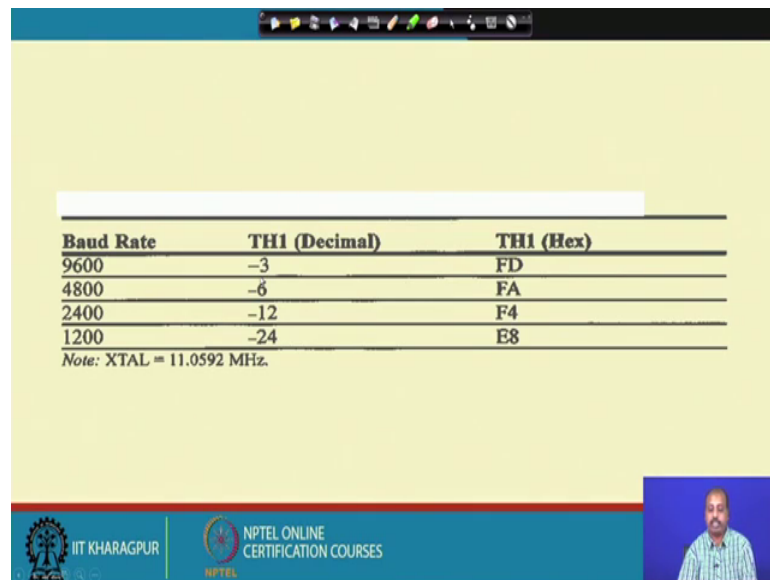
do the connections there then the signal levels will be compatible. Anyway so, once we have done this thing; so, we can go for the serial communication.

(Refer Slide Time: 11:23)



So, this for the serial communication whenever you write some data onto this SBUF register it will be transmitted via this TXD pin. Similarly whenever it receives some data; so, it is there in the shift register and from the shift register it comes to the SBUF register; so the shift register is a not directly accessible. So, this SBUF is accessible by in the program mode. So, you can read the content and get it into the accumulated. So, you have to set the baud rate for this transmission and the baud rate for receiving. So, thus baud rates are to be set for ensuring proper communication through the serial port.

(Refer Slide Time: 11:08)



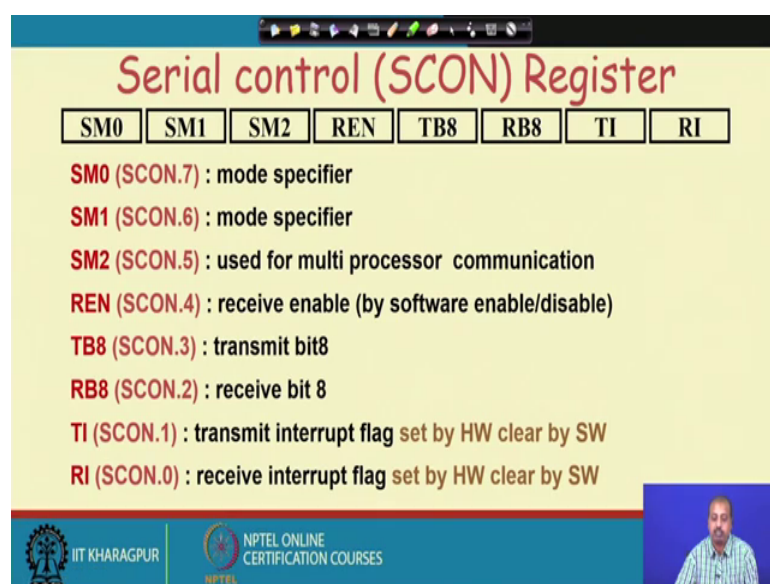
Baud Rate	TH1 (Decimal)	TH1 (Hex)
9600	-3	FD
4800	-6	FA
2400	-12	F4
1200	-24	E8

Note: XTAL = 11.0592 MHz.

Now, for the setting of baud rate there are some values ok. So, me timer in this timer 1 has to be used for this baud rate setting. And this if you want to set a baud rate of 9600 then this TH1 should be set to minus 3. If you are trying to set this baud rate to 4800 then this TH1 should be set to minus 1; then minus 12 and minus 24.

So, these assuming that the crystal frequency is 11.0592 megahertz. So, these are the various values of TH1 register that should be set for this for getting this baud rate.

(Refer Slide Time: 12:49)



Serial control (SCON) Register

SM0	SM1	SM2	REN	TB8	RB8	TI	RI
-----	-----	-----	-----	-----	-----	----	----

- SM0 (SCON.7) :** mode specifier
- SM1 (SCON.6) :** mode specifier
- SM2 (SCON.5) :** used for multi processor communication
- REN (SCON.4) :** receive enable (by software enable/disable)
- TB8 (SCON.3) :** transmit bit8
- RB8 (SCON.2) :** receive bit 8
- TI (SCON.1) :** transmit interrupt flag set by HW clear by SW
- RI (SCON.0) :** receive interrupt flag set by HW clear by SW

Now, for serial communication; so, there is a register for serial control register or SCON register in this SM0, SM1, SM2; so, they are the mode specifier. So, this and this REN is the receive enable TB 8 is for transmit bit 8 and RB8 is the receive bit 8; TI is the transmit interrupt flag. So, it will be set by hardware and RI by software.

So, when this TI interrupt occurs; so, then; that means, some transmission has taken place and when this RI interrupt occurs then this RI interrupts receive interrupt has taken place. But physically TI and RI also they are sending a single interrupt to the processor and the process in the program you need to check whether the TI bit is set or RI bit is set. Accordingly you can understand whether when some data has been transmitted or some data has been received; so, we will see that.

(Refer Slide Time: 13:50)

SM0	SM1	MODE	operation	transmit rate
0	0	0	shift register	fixed (xtal/12)
0	1	1	8 bit UART	variable (timer1)
1	0	2	9 bit UART	fixed (xtal/32 or xtal/64)
1	1	3	9 bit UART	variable (timer1)

So, this SM0 SM1; so, this is a if it is 0 0. So, it is the shift register mode and then the transmitter rate is fixed. So, it is crystal divided by 12 that is that 11.0592 mega hertz that divided by 12 that is the rate at which the transmission will take place. If you are using the mode 0 1; so, it is a 9 bit UART and then this that I will be now the transmission rate can be variable the different baud rates that you are talking about here.

So, different baud rate can be selected if you are operating in the time in the mode 1 by setting this SM0 SM1 to 1. So, if you are operating then you can set this mode to timer 1. You can use the timer 1 for setting the baud rate if; you set it up as 1, 0 then it is mode 2

in mode 2 we have that 9 bit timer. So, this 9 bit UART; so, there I can have say one extra bit there and then we say.

So, that about the transmission baud rate is fixed; so, this is crystal by 32 or crystal by 64. So, that way this transmission rate is fixed and mode 1 1; so, it is 9 bit UART with a variable type variable baud rate. So, you can have this timer to set the to set the transmission rate.

(Refer Slide Time: 15:13)

The slide is titled "Mode of operation" in red text. It contains two main sections: "Mode 0 :" and "Application".

- Mode 0 :**
 - Serial **data** enters and exits through **RxD**
 - **TxD** outputs the shift **clock**.
 - 8 bits are transmitted/received(LSB first)
 - The baud rate is fixed a 1/12 the oscillator frequency.
- Application**
 - Port expansion

The diagram shows a block labeled "8051" with two pins: "TXD" and "RXD". The "TXD" pin is connected to a "clk" input of a "Shift register". The "RXD" pin is connected to a "data" input of the "Shift register". The "Shift register" has eight output lines, each with an upward-pointing arrow.

At the bottom of the slide, there are logos for "IIT KHARAGPUR" and "NPTEL ONLINE CERTIFICATION COURSES". A small video inset of a man is visible in the bottom right corner.

So, mode 0 is the simplest one; so, let us try to understand. So, the serial data enters and exits through receives data; so, serials. So, the transmit data will output the shift clock and so, this is the thing. So, you have got this transmit data and receive data; so, this; so, this is data is used for both transmission and receive.



So, 8 bit is a transmitted this LSB will be received transmitted first baud rate is fixed as 1 by 12 of the oscillator frequency. So, this is typically used port expansion; so, you have got to so, many extra bits and control to. So, you can you can use this individual bits for different port as different port bits ok.

(Refer Slide Time: 16:00)

Mode of operation

- Mode 1
 - Ten bits are transmitted (through TxD) or received (through RxD)
 - A start bit (0), 8 data bits (LSB first), and a stop bit (1)
 - On receive, the stop bit goes into RB8 in SCON
 - the baud rate is determined by the Timer 1 overflow rate.
 - Timer1 clock is $1/32$ machine cycle ($MC=1/12$ XTAL)
 - Timer clock can be programmed as $1/16$ of machine cycle
 - Transmission is initiated by any instruction that uses SBUF as a destination register.

```
graph LR; XTAL[11.0592 MHz XTAL oscillator] --> Div12[+ 12]; Div12 -- "Machine cycle freq. 921.6 kHz" --> Div32[+ 32 by UART]; Div32 -- "28800 Hz" --> Timer1[To timer 1 to set the baud rate];
```

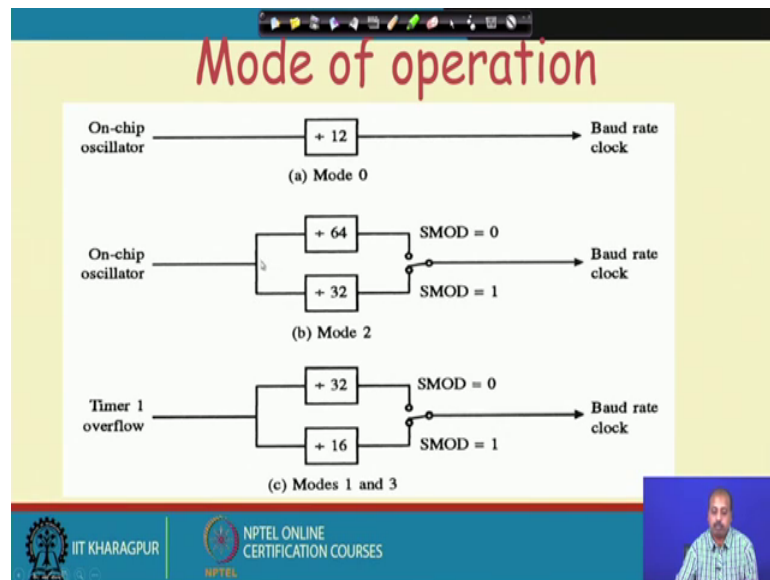
Then mode 1; so, here 10 bits are transmitted through T x D or this or received through a R x D. So, 10 bits out of this 10 bits one is the start bit 8 bits of data and 1 stop bit. So, this is the 10 bit that will be transmitted and on this is the stop bit goes into RB8 RB8 register when the baud rate is determined by the timer 1 overflow rate; so, if you if the timer 1 clock is 1 by 32 of the machine cycle.

So, machine cycle is 1 by 12 of the piston; so, you can find like what is the what is the what is the value like because the machine cycle pistol is 11.0592 that divided by 12 and that is again divided by 32 for feeding the UART clock. So, that way you can set this timer 1 value; so, this will go to the timer 2. So, this timer 2 is now getting a clock of 3 into 28800 hertz.

So, accordingly you can set the timer value that for getting that delay. So, timer clock can be programmed as one 16th of the machine cycles. So, that way previously it was one twelfth of the machine cycle; now it becomes one 16th of the machine cycle. So, if you follow this formula then you can get the delays and whenever any instruction uses SBUF as its destination register transmission will start.

So, UART clock is controlled by this timer 1 and for setting the timer 1 you can understand that this crystal is a crystal clock is machine cycle is divided by 16 and that way. So, machine cycle is one twelfth of the piston clock and this piston clock is divided by sorry the machine cycle is divided by 16 so, that will give you the delay for the timer.

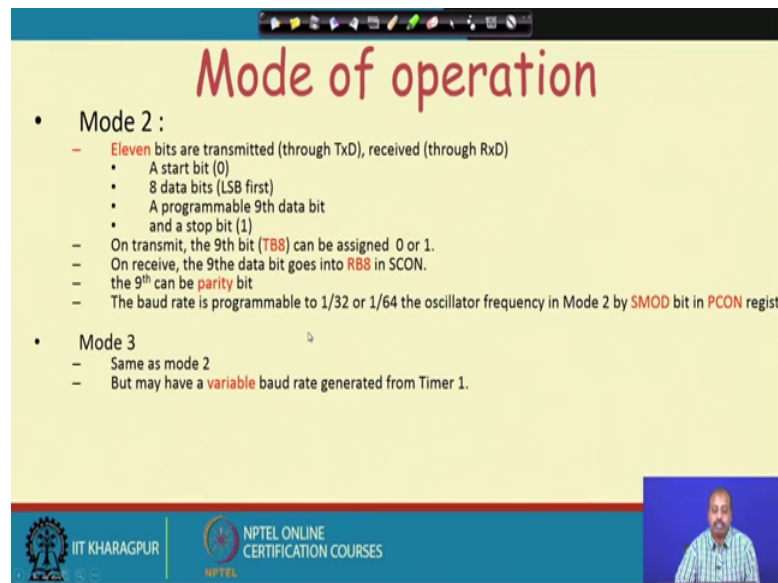
(Refer Slide Time: 17:52)



So, in mode 0 the on-chip oscillator is divided by 12; so, that is the baud rate and clock. So, this is if it is in mode 2; so, you have got two SMOD is equal 0 SMOD equal to 1. So, if SMOD equal to 0 it is divided by 64 that is fixed if it is in SMOD equal to 1; so, it is divided by 32. So, these baud rate is also fixed in MODs 1 3; so, this. So, this transmission rate is controlled by this timer 1 overflow; so, that is why this MODs are useful.

So, if it is if it is connected if it is SMOD equal to 0 then the timer overflow will be divided by 32 if it is SMOD is equal to 1, then the timer 1 will be you divided by divided 16 that overflow will be. So, that is determined in the baud rate this way we can have different baud rate settings and different transmissions of serial transmissions of 8051.

(Refer Slide Time: 18:54)



The slide is titled "Mode of operation" in a large, red, serif font. Below the title, there are two main bullet points. The first is "Mode 2:", followed by a list of sub-points: "Eleven bits are transmitted (through TxD), received (through RxD)", "A start bit (0)", "8 data bits (LSB first)", "A programmable 9th data bit", and "and a stop bit (1)". Further sub-points for Mode 2 include: "On transmit, the 9th bit (TB8) can be assigned 0 or 1.", "On receive, the 9th data bit goes into RB8 in SCON.", "the 9th can be parity bit", and "The baud rate is programmable to 1/32 or 1/64 the oscillator frequency in Mode 2 by SMOD bit in PCON register". The second main bullet point is "Mode 3", with sub-points: "Same as mode 2" and "But may have a variable baud rate generated from Timer 1." The slide has a yellow background and a blue footer containing logos for IIT KHARAGPUR and NPTEL ONLINE CERTIFICATION COURSES. A small video inset of a man is visible in the bottom right corner.

In mode 2 11 bits are transmitted through transmitter. So, TXD or received through R x D and out of that 11 bit ones is the start bit 8 bit of data one programmable ninth data bit. So, programmable ninth data bits; so, you can use it of parity or something like that and one stop bit. So, one transmit this 9 bit the TB 8 can be assigned 0 or 1 on this is the 9 bit will be RB8 register RB8 of the SMOD. So, it can be the parity bit so,; so, we can compute.

So, this bit is available in the; so, rest of the 8 bits will go to the SBUF register, but you can get the ninth bit in the TB8 or RB8 of this SCON register. So, you can check for the parity by that and again the baud rate is programmable by this SMOD and there is a PCON register is also there will come to that which will control this thing.

Mode 3 same as mode 2, but may have variable baud rate generated by timer 1. So, that way it is the mode 3 operation.

(Refer Slide Time: 20:07)

What is SMOD

- ❑ Bit 7 of PCON register
- ❑ If SMOD=1 **double** baud rate
- ❑ PCON is not bit addressable
- ❑ How to set SMOD

```
Mov a, pcon
Setb acc.7
Mov pcon, a
```

MSB							LSB
SMOD	—	—	—	GF1	GF0	PD	IDL

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, this SMOD is the bit 7 of the PCON register. So, PCON is the power control register. So, this is the out of the bit 7 is if SMOD equal to 1 then double baud rate is used. So, whatever if you if you looking to this diagram like here; so, we said that SMOD equal to 0; it is 64 and SMOD equal to 1, it is 32 similarly here SMOD equal to 0 is 32 and SMOD equal to 1 is the clock is divided by 16.

So, clock is clock frequency will be higher if you make a SMOD equal to 1. So, that is controlled by the PCON registers bit numbers 7. So, PCON is not bit addressable, but; so, to set it. So, you have to you can do it like this first we get this PCON register value onto a register, then we set bit this accumulator seventh bit and then we move this accumulator in to PCON register. So, that can be done; so, this bit will get modified this bit is now getting set for that purpose.

(Refer Slide Time: 21:15)

The slide displays the Power Control Register (PCON) bit field. The bits are arranged from MSB (Most Significant Bit) on the left to LSB (Least Significant Bit) on the right. The bit field is represented as follows:

MSB							LSB								
PCON.7	SMOD	PCON.6	—	PCON.5	—	PCON.4	—	PCON.3	GF1	PCON.2	GF0	PCON.1	PD	PCON.0	IDL

Below the bit field, a table provides the function for each bit:

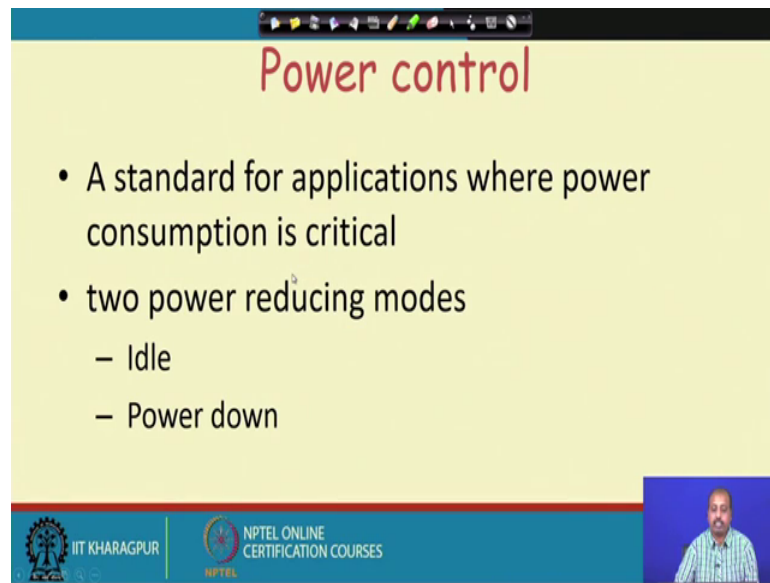
BIT	SYMBOL	FUNCTION
PCON.7	SMOD	Double Baud rate bit. When set to a 1 and Timer 1 is used to generate baud rate, and the Serial Port is used in modes 1, 2, or 3.
PCON.6	—	Reserved.
PCON.5	—	Reserved.
PCON.4	—	Reserved.
PCON.3	GF1	General-purpose flag bit.
PCON.2	GF0	General-purpose flag bit.
PCON.1	PD	Power-Down bit. Setting this bit activates power-down operation.
PCON.0	IDL	Idle mode bit. Setting this bit activate idle mode operation.

The slide also features the IIT KHARAGPUR logo and NPTEL ONLINE CERTIFICATION COURSES logo at the bottom left, and a small video inset of a presenter at the bottom right.

This PCON register is a power is the power control register; so it has got this SMOD bit. So, apart from that it has got this; so, this 3 bits are reserved. So, then we have got this GF 0, GF 1 and PD and IDL; so, GF 1 is general purpose flag bit GF 0 is general purpose flag bit. So, they can be used by the programmer then this P D PCON dot 1 is the power down bit. So, if you set this P D to 1; so, this will activate the power down operation and if it I if PCON 0 is the IDL bit. So, if you set this IDL bit to 1.

So, this the idle thus; so, it will may activate the idle mode of the operation. So, from idle mode operations; so, it will be it will be taking some interrupts should be reached the processor to the two to activate it on the other this power down mode. So, if it is power down may be it can it can advice many devices to go to power down mode and all that. So,me control signals are activated by which it will tell external world that it is moving to power down mode.

(Refer Slide Time: 22:23)



Power control

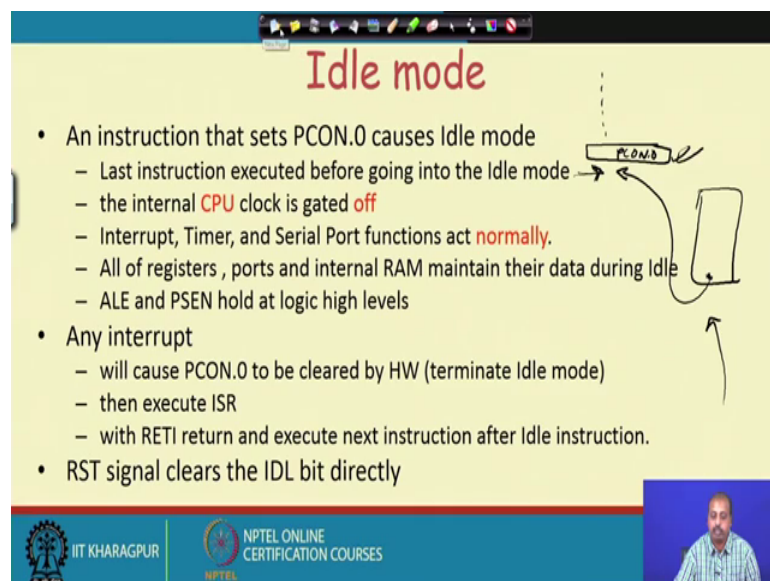
- A standard for applications where power consumption is critical
- two power reducing modes
 - Idle
 - Power down

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, power control standard for applications where power consumption is critical, so this is there are. So, there are many standard for this power control like ACPI, we have got a control standard for this power control. So, similarly we have got for any processor that you have now. So, if you look into you will find now some power control mechanism has been provided.

So, this 8051 also has got this power control feature. So, this has got one idle mode and a power down mode.

(Refer Slide Time: 23:00)



Idle mode

- An instruction that sets PCON.0 causes Idle mode
 - Last instruction executed before going into the Idle mode
 - the internal CPU clock is gated off
 - Interrupt, Timer, and Serial Port functions act normally.
 - All of registers, ports and internal RAM maintain their data during Idle
 - ALE and PSEN hold at logic high levels
- Any interrupt
 - will cause PCON.0 to be cleared by HW (terminate Idle mode)
 - then execute ISR
 - with RETI return and execute next instruction after Idle instruction.
- RST signal clears the IDL bit directly

PCON.0

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

In idle mode; so, an instructions that sets this PCON dot 0 it will go into idle mode and last instruction executed before going in to idle mode. So, the internal CPU also this is the; so, that will be last instruction. So, that will be after that nothing more will happen the CPU clock is gated off.

So, CPU will not get the clock as a result the internal operation will stop interrupt timer and serial port functions act normally. So, these operations will continue, but only the CPU will be not doing any further instruction processing. So, it will be idle, but the interrupt timer and serial port they will operate; all of registers ports and internal RAM maintain their data during this idle period.

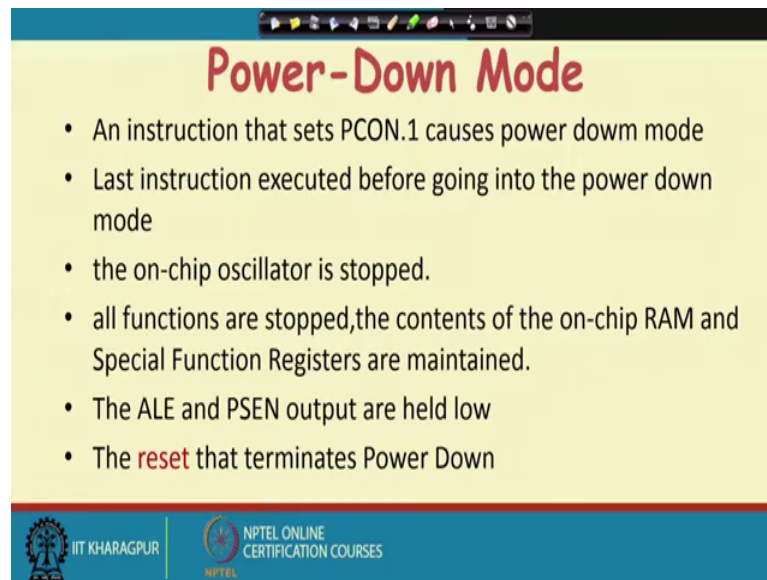
So, the internal values will be stored, but will be held, but the processor will not do any other operation. ALE and PSEN will hold at logic high level since it is the processor is not fetching any further instruction. So, this ALE is also high and PSEN bar line is also high to mean that no activity is taking place.

Now, to come out of the idle mode; so, it has to use some interrupt and interrupt will cause this PCONs dot 0 to be cleared by hardware. So, it will terminate the idle mode and then it will go to the interrupt service routine. So, that is very good like if some event has occurred in the outside world. So, it will be it will be turning on the processors operation. So, it will go to the ISR and from that points onwards whatever the processes was doing previously; so, it will continue with that.

So, the instructions just after the idle instruction; so, they are on returned from the interrupt. So, it will be occurring that way; so, there as it is as it is said here suppose this is a piece of code and this instruction is sitting this PCON dot 0. So, it is setting that PCON dot 0; now after that, so it is can come back to this point, but at this point the processor has become idle. So, if the processor has become idle after sometime from interrupt has occurred and when the interrupt has occurred then this is the interrupt service routine so, processor will execute this interrupt service routine.

And at the end there is a return the data instructional and the data instruction will take it to the instruction just after setting of this PCON thing. So, this way it can come back to this point like it is coming to this after the interrupt is over. So, it can come to this point and the idle mode is over. So, this way interrupt can take the processor out of this idle mode and do the operation.

(Refer Slide Time: 25:55)



Power-Down Mode

- An instruction that sets PCON.1 causes power down mode
- Last instruction executed before going into the power down mode
- the on-chip oscillator is stopped.
- all functions are stopped, the contents of the on-chip RAM and Special Function Registers are maintained.
- The ALE and PSEN output are held low
- The **reset** that terminates Power Down

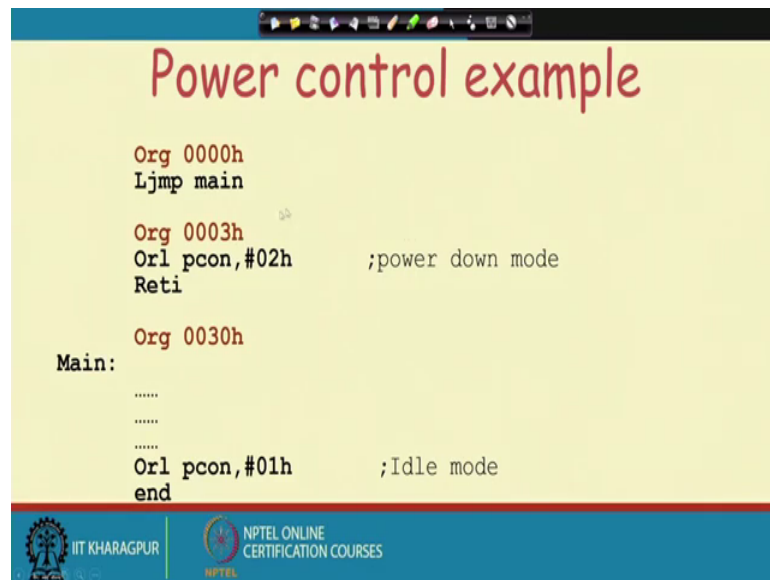
IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

On the other hand this power down mode; so, an instruction that sets an PCON dot 1 it causes the power down mode. And the same thing that is a last instruction is executed before going into power down mode the on chip oscillated will be stopped and all functions are stopped contents of 1 GB RAM and SSFRs will be maintained ok.

So, ALE PSEN bar will be held low only reset can terminate the power down mode. So, in unlike this, but unlike this the idle mode where the interrupt can turn off this power down mode; this idle mode in case of power down mode only the reset operation can be they can reset and reset the this entire power down mode. And it can take it take the processor out of this operation out of this power down mode. So, these are very useful when you are when you are talking about this power handling part.

So, these are very much useful for designing systems where we have to when we are going to save the battery like most of the cases microcontroller they are used for embedded application where the battery life is important. So, we can use this processor for doing this things; we can say that we can put the power down we cannot required the operations we can put it in power down mode and later on or in the idle mode and whenever the this service is required we can turn on the device or turn on the services; so, this is a power control example.

(Refer Slide Time: 27:23)



```
Power control example

Org 0000h
Ljmp main

Org 0003h
Orl pcon,#02h ;power down mode
Reti

Org 0030h
Main:
.....
.....
.....
Orl pcon,#01h ;Idle mode
end
```

So, this is Ljmp main; so, this is main the routine. So, you see that is this is the PCON so, PCON register we are ending with 02. So, this will put it in the power down mode and so, this is the idle mode. So, it is you can use it can come back to this when the interrupt occurs. So, it can come back to this point and on reset. So, it will be this power down will be cancelled and this is the idle mode. So, when the idle mode some interrupt will occur for example, here when the interrupt occurs. So, it will be doing that.