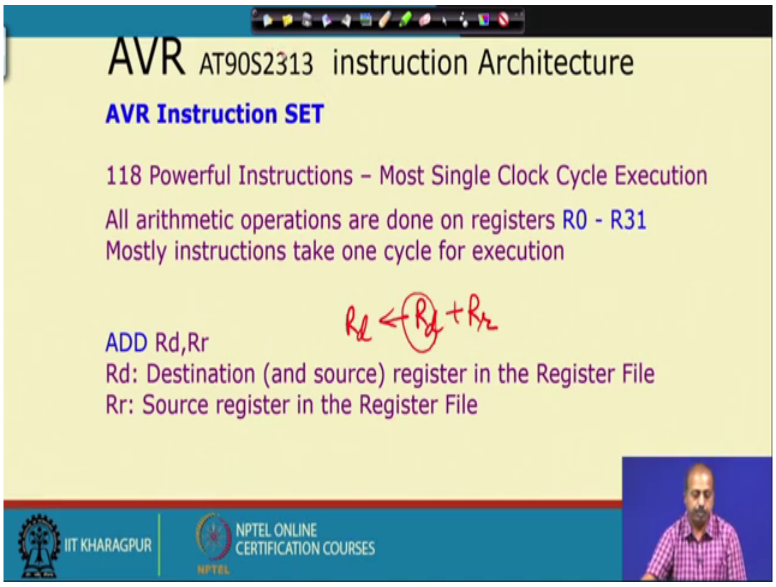
Microprocessors and Microcontrollers
Prof. Santanu Chattopadhyay
Department of E & EC Engineering
Indian Institute of Technology, Kharagpur

Lecture - 51
AVR

So, the ADD instruction in this is a AVR processor so it may be like add R d comma R r, where R d is the destination and source in the register file, and R r is a source register in the register file. So, it will be the operation that will happen is that R d we will get R d plus R r.

(Refer Slide Time: 00:22)



AVR AT90S2313 instruction Architecture

AVR Instruction SET

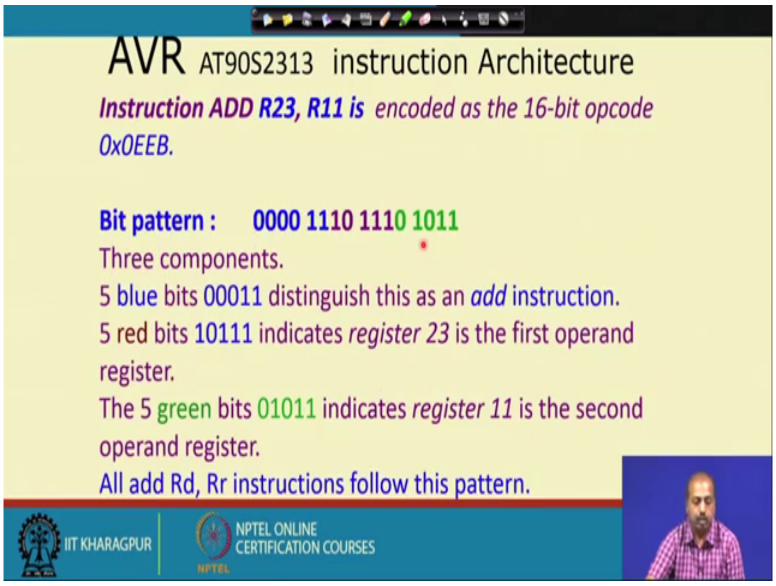
118 Powerful Instructions – Most Single Clock Cycle Execution

All arithmetic operations are done on registers R0 - R31
Mostly instructions take one cycle for execution

ADD Rd,Rr $R_d \leftarrow R_d + R_r$
Rd: Destination (and source) register in the Register File
Rr: Source register in the Register File

So, this is the destination and source and the second register is the other source. So, this way this add instruction may be executed.

(Refer Slide Time: 00:49)


AVR AT90S2313 instruction Architecture

Instruction ADD R23, R11 is encoded as the 16-bit opcode 0x0EEB.

Bit pattern : **0000 1110 1110 1011**

Three components.

5 blue bits 00011 distinguish this as an *add* instruction.

5 red bits 10111 indicates *register 23* is the first operand register.

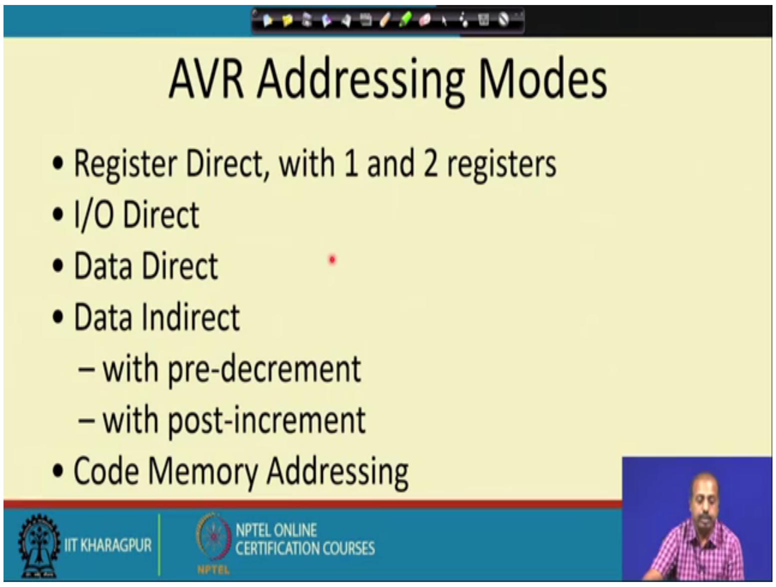
The 5 green bits 01011 indicates *register 11* is the second operand register.

All add Rd, Rr instructions follow this pattern.

So, for example, this add R 23 comma R 11. So, this will be given as a 16 bit code 0 0EEB and the bit pattern coding is like this. So, out of that this 5 bits, so 0 0 0 1 1 they will correspond to add instruction then next 5 red bits, so this 1 0 1 1 1. So, they will give the register number 23 as the first operand and then this 0 1 0 1 1, so that is 11. So, that is the second; second operand of the register.

So, this all R d, R r instruction will follow this pattern. So, this has got a coding like this. So, this 0 is redundant. So, this 0 is extra nothing is there. So, that is taken as 0.

(Refer Slide Time: 01:42)


AVR Addressing Modes

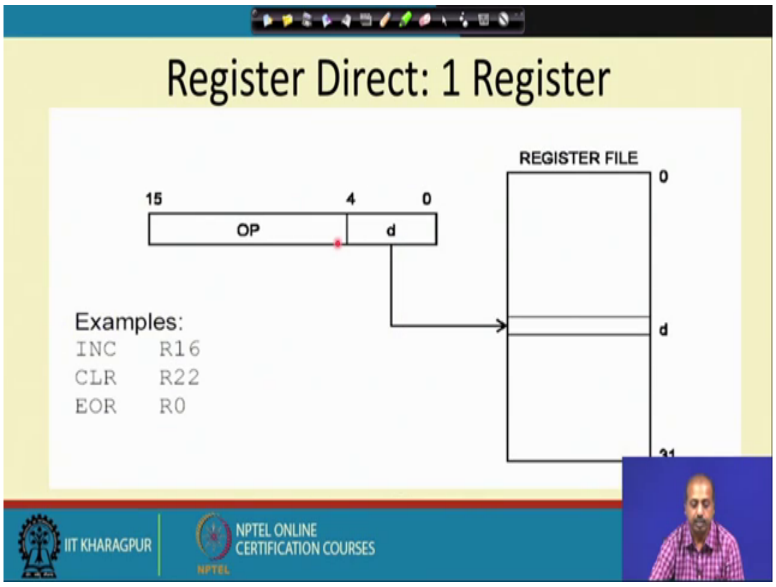
- Register Direct, with 1 and 2 registers
- I/O Direct
- Data Direct
- Data Indirect
  - with pre-decrement
  - with post-increment
- Code Memory Addressing

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES

So, all instructions will be coded like this. So, if we are looking into addressing modes. So, then there are different modes like you have register direct with 1 and 2 registers. So, we can directly specify your operand as a register or you can have IO direct.

So, you can have some from the input output port addresses the data direct data indirect with pre and data indirect can also be with pre decrement post increment type of thing and code memory addressing. So, we will see these addressing modes.

(Refer Slide Time: 02:12)



Register Direct: 1 Register

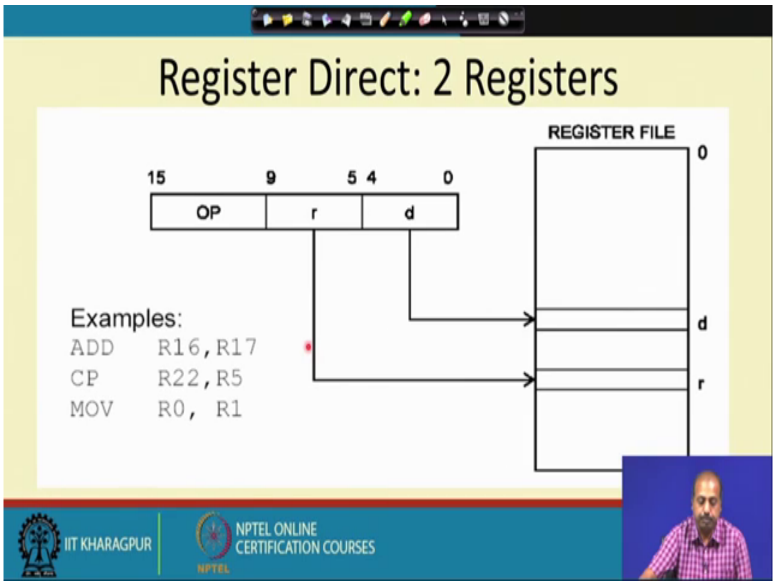
Examples:

```
INC   R16
CLR   R22
EOR   R0
```

Register direct, one register like say increment R16, so this instruction, this will increment the R16 register. So, there are 16 bit instructions; in the register file so R16 is given the number the corresponding index is d. So, the opcode part will be there and that this d the this 4 bits will identify the register on which you want to do the operation. So, this is d. So, in the current register bank register 16 will be incremented or say clear 22, so based on that 22. So, it will be the destination will be come.

So, this d is the number. So, for this case this d will be 16, for the second instruction d will be 22, for the third instruction d will be 0, so it will be like that. But 4 bit coding will be sufficient to differentiate between the 16 different registers that we have.

(Refer Slide Time: 03:10)



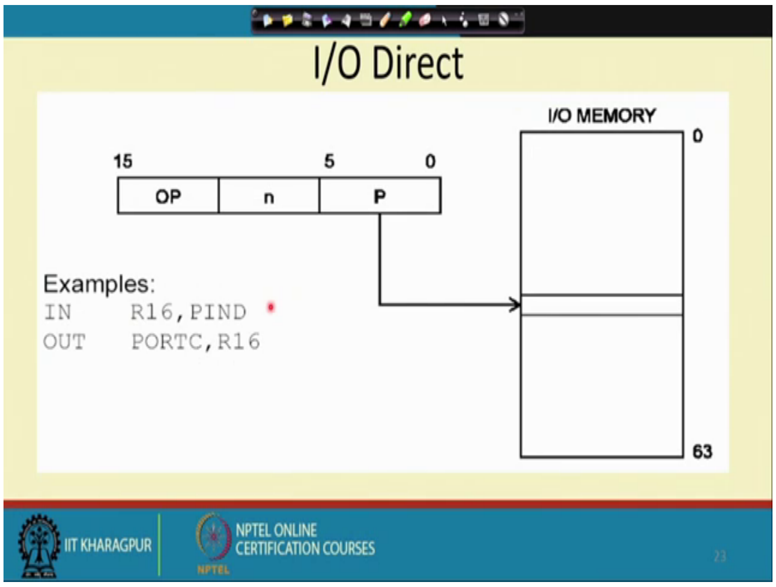
Register Direct: 2 Registers

Examples:

```
ADD   R16,R17
CP    R22,R5
MOV   R0, R1
```

Now, you can have register direct with two registers like this add instruction add R 6 comma R 17 here we will have this register direct and this R r and R d. So, both may be there and they can be used for again we can have two registers mentioned here and the indices stored as part of the instruction. So, this is also a 16 bit coding. So, register direct with two registers.

(Refer Slide Time: 03:40)



I/O Direct

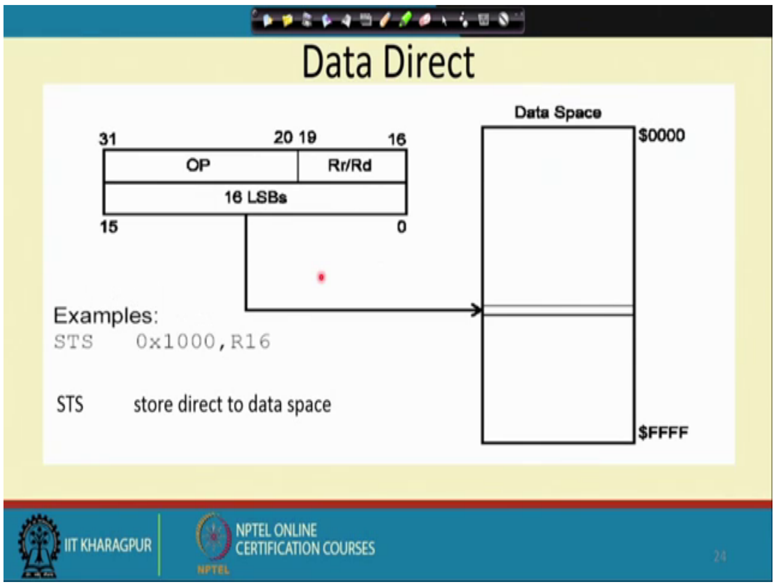
Examples:

```
IN    R16,PIND
OUT   PORTC,R16
```

IO indirect, you can say that it is in R16 comma PIND. So, PIND it will identify some port number. So, this will be so this P will be identifying. So, this is the opcode part which is say in; then say R16, so this is the register. Now, this is the register number. So, this will be coming to this n and this IO memory. So, this PIND, so this will as there are 64 IO pins as we have said. So, it is the it is a 64 IO locations that we have seen. So, we have it is from the 60 whatever be the port number we have identified.

So, from that port the value will be put into this corresponding register. So, from this particular port the value will come to the register R16. Similarly out PORTC R16. So, it will output this R16 value onto PORTC. So, again that PORTC will have some address and accordingly it will be selected.

(Refer Slide Time: 04:46)



Data Direct

Examples:

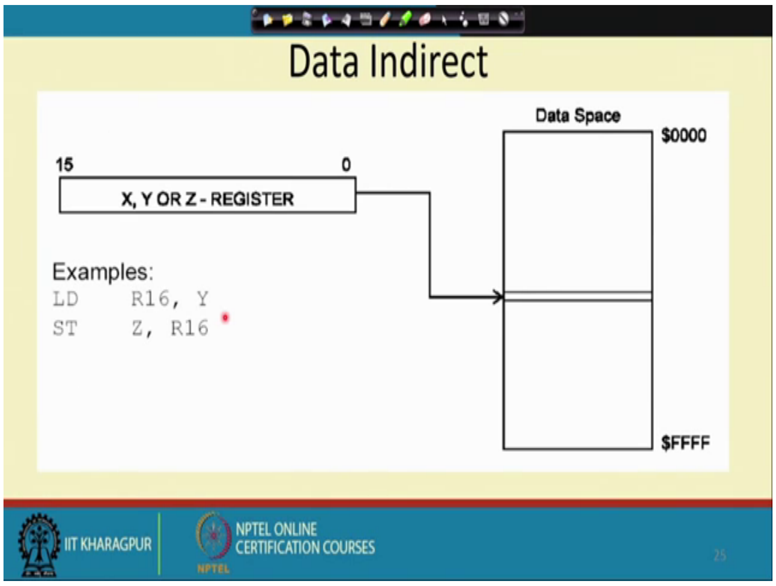
```
STS    0x1000,R16
```

STS   store direct to data space

Then data direct, you can say like the STS instruction. So, stored data store direct to data space. So, here, we have got this opcode which is that STS and this register which register will be acting as the source and that value will be going to this destination.

So, this LS this address is a 16 bit address and the 16 bit address will be will be used that will be part of the instruction and this R16 value will be copied onto that particular address. So, this is a 2 byte sorry this is a 32 bit instruction. So, 0 to 32. So, so previously we had all 16 bit instructions. So, this is 32 bit instruction.

(Refer Slide Time: 05:36)



Data Indirect

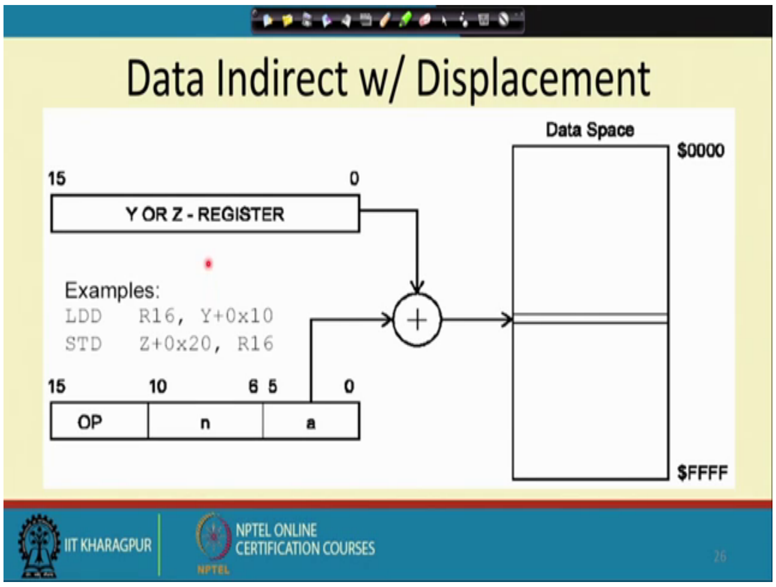
Examples:

```
LD    R16, Y
ST    Z, R16
```

Then we have got data indirect. So, you can there are 3 registers as we said X Y and Z and these registers can be used for indirect addressing. So, you can say like load R16 comma Y. So, wherever Y is pointing to Y register is pointing to, so that locations content will be copied onto R16 register or stored Z comma R16.

So, Z is another pointer. So, whatever be the value of R16. So, that will be stored into the memory location pointed to by the Z register. So, we have got this X Y or Z register working like that.

(Refer Slide Time: 06:15)



Data Indirect w/ Displacement

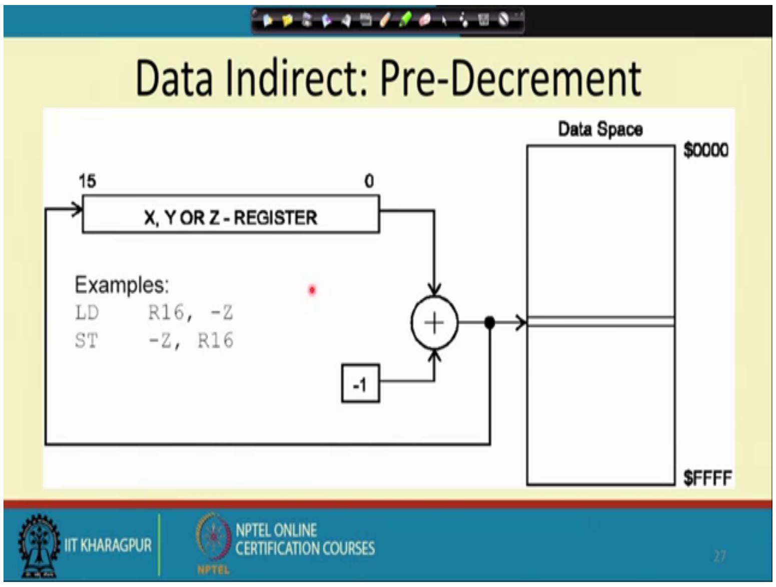
Examples:

```
LDD   R16, Y+0x10
STD   Z+0x20, R16
```

You also have data indirect with displacement. So, it is a load with displacement LDD. So, this is R16 Y plus some constant value ten. So, Y register with that this 10 value which is coded here will be added and that will act as the offset where the value should be written. So, if this location content is a 1000, so with that this 0 will be this 10 will be added. So, it will become 1010 and then that location content will be updated.

Similarly, we can have STD where we have first we mentioned this indirect register plus the offset which is say 0 x 2 0 and that value will be, so there will be storing the value of register R16. So, this way we can have this data indirect with displacement.

(Refer Slide Time: 07:15)


Data Indirect: Pre-Decrement

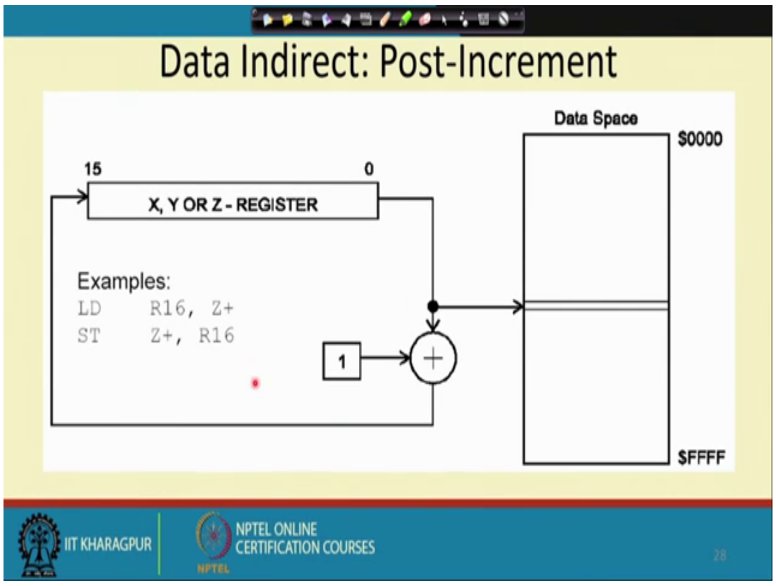
Examples:

```
LD    R16, -Z
ST    -Z, R16
```

There is another mode data indirect with pre decrement. So, data indirect with pre decrement here it is LD there is no displacement. So, there is that LDD part is missing D is not there only LD R16. So, it is, in R16 will be loading the value pointed from the memory location pointed to by the Z register, but before that Z will be decremented by 1. So, it is pre decrement.

So, Z value will be decremented and then that address will be used to access the data memory. So, this is minus Z or you can have this pre decrement ST minus Z R16. So, you can decrement before accessing the location and first R16 value will be stored in the memory location pointed to by Z minus 1 and simultaneously so Z is decremented by 1.

(Refer Slide Time: 08:12)



Data Indirect: Post-Increment

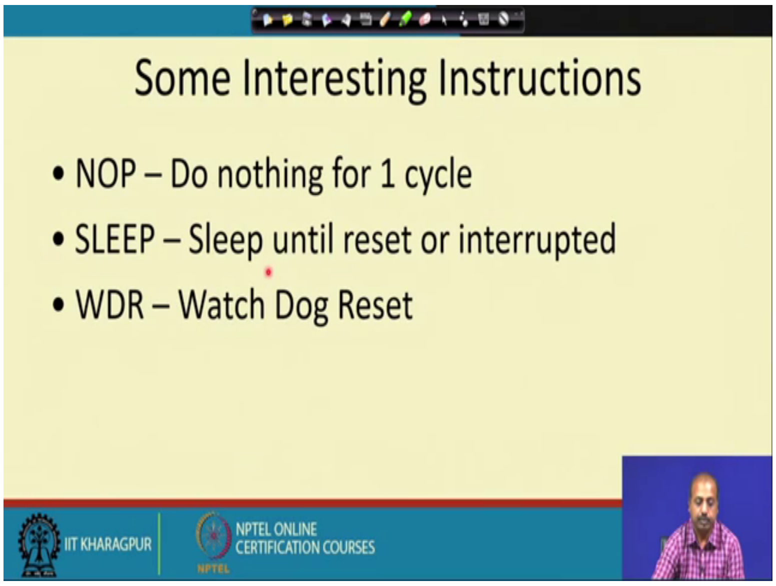
Examples:

```
LD    R16, Z+
ST    Z+, R16
```

So, we can have post increment also. So, it is Z plus, so this Z register value will be incremented, but the previous value of Z register will be used to access the memory after that this Z, Z register content will be incremented by 1. So, this is the post increment mode and we can have similarly for storing also we can have post increment ST Z plus R16. So, that way we can have this value stored in the memory location pointed to by Z register after that this Z register content will be updated.

So, we can have this data indirect with post increment. So, we have pre decrement and post increment. We do not have post decrement or pre increment. So, those modes are not there, but pre decrement and post increment these are the two modes that we have with data indirect mode.

(Refer Slide Time: 09:09)



Some Interesting Instructions

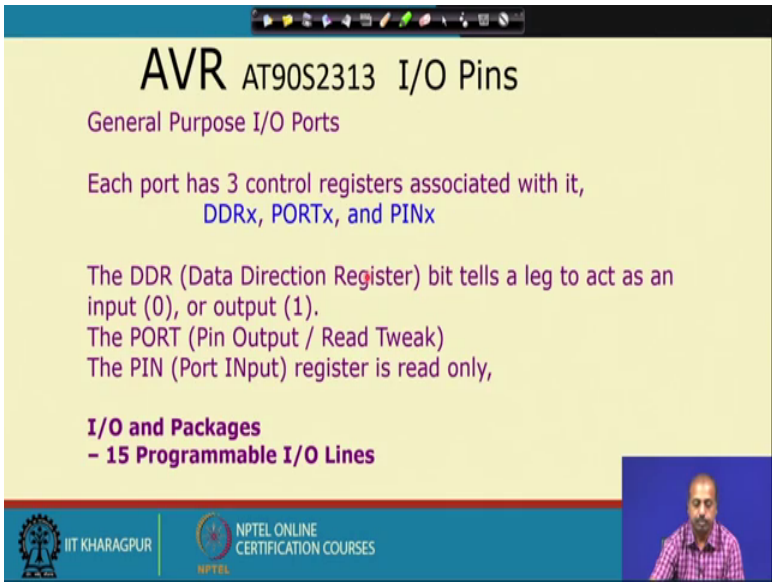
- NOP – Do nothing for 1 cycle
- SLEEP – Sleep until reset or interrupted
- WDR – Watch Dog Reset

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES

Some other interesting instructions like we have got this NOP, NOP we do nothing for one cycle then sleep. So, sleep until reset or interrupted. So, sleep instruction is there then watch dog reset.

So, as we are discussing while talking about that pic microcontroller. So, we have got watch dog timer. So, here also we have got watch dog and there is an instruction by which you can reset that watch dog timer. So, you can say like WDR watch dog reset that can be used.

(Refer Slide Time: 09:39)



AVR AT90S2313 I/O Pins

General Purpose I/O Ports

Each port has 3 control registers associated with it,
DDRx, PORTx, and PINx

The DDR (Data Direction Register) bit tells a leg to act as an input (0), or output (1).
The PORT (Pin Output / Read Tweak)
The PIN (Port INput) register is read only,

I/O and Packages
– 15 Programmable I/O Lines

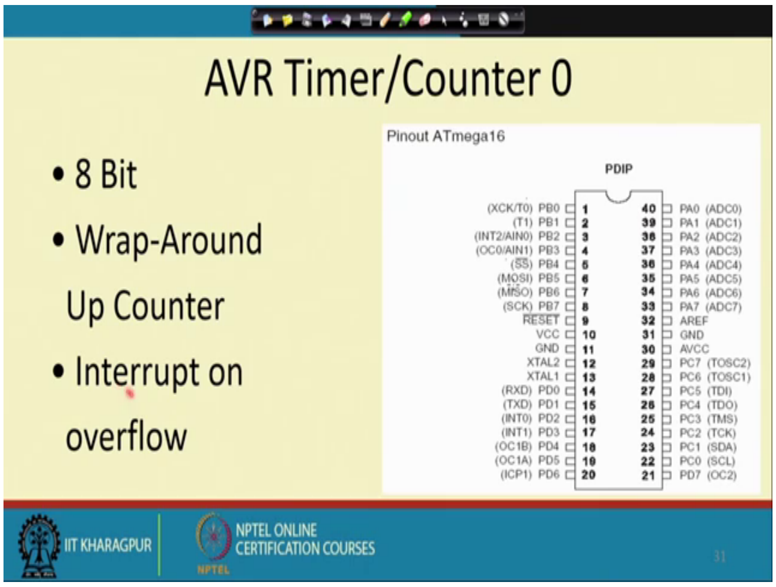
IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES

So, for the IO pins, these are general purpose IO ports and each port has 3 control registers associated with it DDRx, PORTx and PINx. So, the DDR register is the data direction register. So, it will tell whether a pin will act as an input pin or an output pin. So, it is if it is 0 then it will be an out input pin and if the bit is 1 in the DDR register then it will act as output pin.

Then the port, it is pin will act as output or it will be a read operation and then read tweak. So, we will come to that and there is a pin. So, there is a pin can be the port identified; port input. So, this register is read only ok. So, that way we can have different configurations.

Then there are IO pins and packages. So, 15 programmable IO lines are there. So, that can be used for input output operation.

(Refer Slide Time: 10:45)



AVR Timer/Counter 0

- 8 Bit
- Wrap-Around Up Counter
- Interrupt on overflow

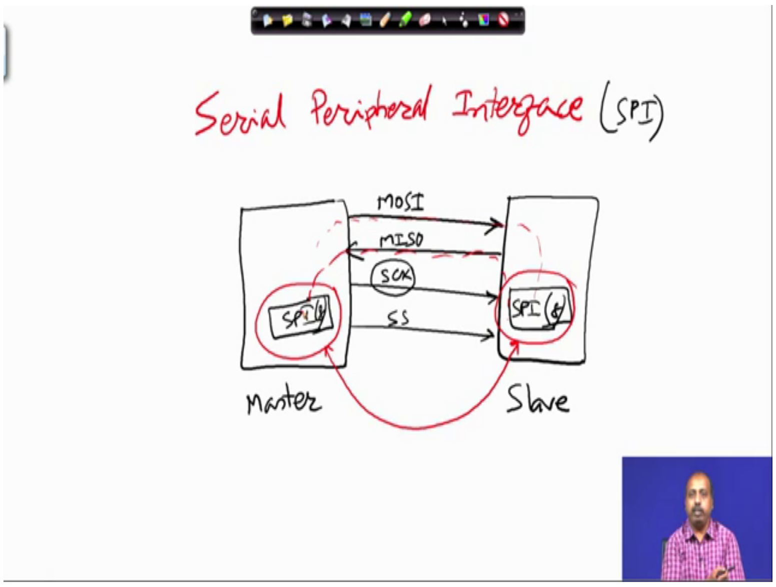
Pinout ATmega16
PDIP
(XCK/T0) PB0 1
(T1) PB1 2
(INT2/AIN0) PB2 3
(OC0/AIN1) PB3 4
(SS) PB4 5
(MOSI) PB5 6
(MISO) PB6 7
(SCK) PB7 8
RESET 9
VCC 10
GND 11
XTAL2 12
XTAL1 13
(RXD) PD0 14
(TXD) PD1 15
(INT0) PD2 16
(INT1) PD3 17
(OC1B) PD4 18
(OC1A) PD5 19
(ICP1) PD6 20
40 PA0 (ADC0)
39 PA1 (ADC1)
38 PA2 (ADC2)
37 PA3 (ADC3)
36 PA4 (ADC4)
35 PA5 (ADC5)
34 PA6 (ADC6)
33 PA7 (ADC7)
32 AREF
31 GND
30 AVCC
29 PC7 (TOSC2)
28 PC6 (TOSC1)
27 PC5 (TDI)
26 PC4 (TDO)
25 PC3 (TMS)
24 PC2 (TCK)
23 PC1 (SDA)
22 PC0 (SCL)
21 PD7 (OC2)
IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES

As far as timers are concerned, it has got 8 bit timer and these timers are wrap around up counter they can be configured as wrap around up counter. So, after the overflow, it will be coming down to 0 and from there it will be going and there will be an interrupt controller. So, there is an interrupt on the overflow like if they when the timer or the counter overflows then this interrupt will be generated and it can be told the processor can be told that an interrupt has occurred.

So, this diagram, this is one pin diagram for this at mega 16 processor. So, you see that you can have a number of ports like PA 0 to PA 7 as ports then PB 0 to PB 7 as ports. So, there are 4 such ports PA, PB, PC and PD and so there are timers like T0 and T1 there are two timers then we have got analog inputs 0 1, so for this analog input.

Then there are some special lines that you can see here is the MOSI that master out slaving and MOSI, MISO, then this SS bar and SCK. So, these are used for say some special communication serial communication which is known as SPI or serial peripheral interface, which is like this.

(Refer Slide Time: 12:15)



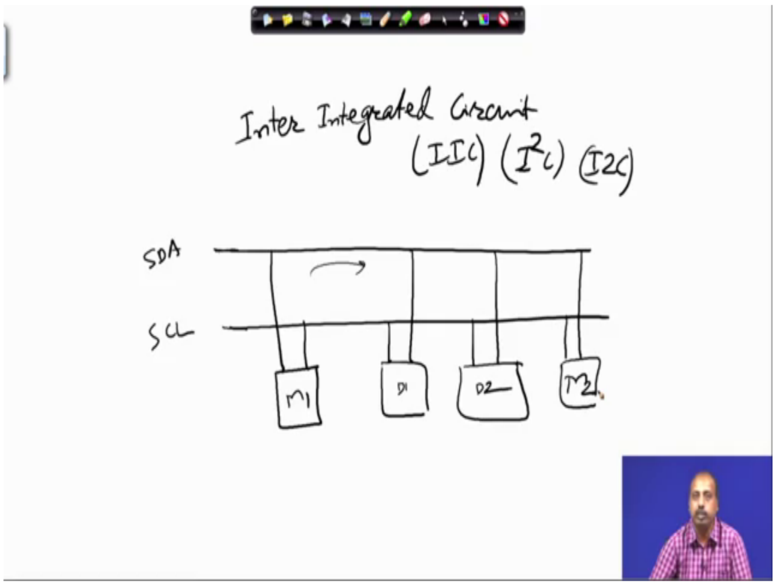
So, this serial peripheral interface or SPI, it is a very simple type of interface where between the processor and device. So, there is a master and there is a slave ok. So, this is the master and this is the slave and the data transfer is very simple. Both the master and slave they have got one register this is they are called SPI register. So, this is a suppose this is the SPI register in the master. So, this is an 8 bit register and slave also has got one SPI register which is 8 bit, fine.

Now, there is a pin which is master out slave in or MOSI master out slave in there is another pin master in slave out master in slave out. So, from the master side we can say it like. So, you can see it like this. So, then there is another pin which is the slave clock. So, this is slave clock and there is another pin which is slave select. So, this is the configuration. Now, when the data transfer will take place. So, this is a synchronous transmission and this whole thing is controlled by this slave clock. So, the master will be giving the clock and the transfer will be based on that and the transfer that happens is basically an exchange. So, this transfer that takes place is an exchange of these two register values. So, there is no direction or transmission. So, it is always bi directional. So, this SPI value will be going through this MOSI line to this one and this value will be coming through this MISO line to this one.

So, that way there is an exchange between the two content; So, this is a very simple type of protocol and many of the embedded systems they use this type of protocol because the interfacing devices that we are going to connect, they are going to be very simple. And they do not support say very complex communication protocol and so, this serial peripheral interface or SPI, this is used, so a very simple protocol ok.

So, you can always support in a microcontroller this type of protocol, but in atmega series you see you have got this thing directly implemented. So, you have got this slave select line then MOSI, MISO and this slave clock. So, they are useful for doing this for SPI type of interface another interesting interface that we have here is via this SDA and SCL line. So, this is the this is these are for another type of interface which is known as I square C interface ok.

(Refer Slide Time: 15:30)



So, they are for this I square C interface inter integrated circuits inter integrated circuit they have got the name IIC or I square C or I 2 C there are various names ok. So, here what happens is that we have got two buses one is called this SDA and another is called SCL, and then we have got these devices that will hang from this buses. So, every device has got one connection to SDL and one connection to SCL, it will have like this.

Now, when the communication takes place, this master, so if this is a master and these are two devices then this master will put this device address onto the bus and the device which will be which will have that address. So, this will be sensing that value and then it will be the next byte that will follow. So, this is a serial transmission; so, next byte that will follow so that will be taken by that device.

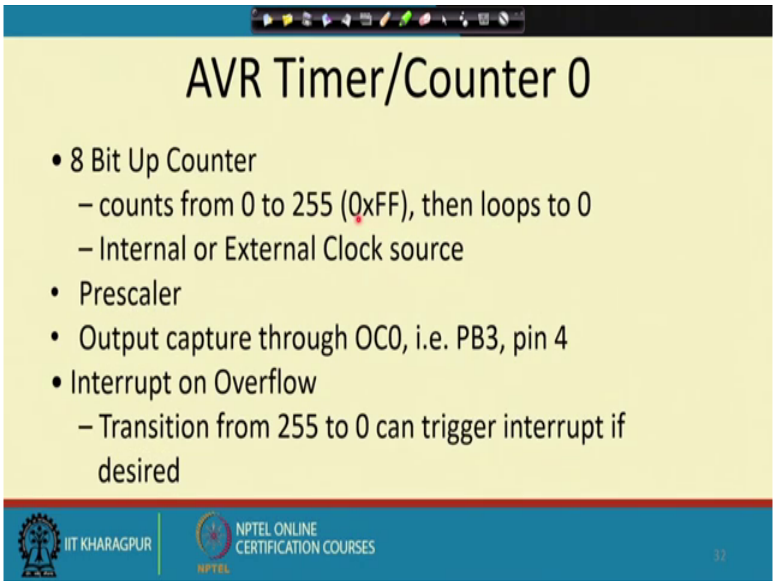
So, this is a two wire interface where we have got only two lines SDA and SCL and which is serial transmission this is serial transmission, but we can have multiple bus masters and multiple bus slaves that we can connect to this system. So, this be master 1, this be master 2. And there is a protocol by which it will detect that there may there is a collision or something like that.

So, there are various such things are there various protocols are there in I square C, but still it is much much simpler compared to full fledged communication protocol where I will have number of devices hanging from buses there will be bus arbiter and all those. So, those things are not there. So, this SPI and I square C they are two very simple interfacing standards and in atmega series of processor so you will find them ok. So, they are useful, so for connecting some simple device to the system.

So, the many of these devices like say real time clock module then, this ADC modules, then this small chip memory chips, so they are having this SPI or I square C interfaces ok.

So, that way the design becomes simpler for those say those process those say chips or if you are designing a sensor maybe we do not bother much to have a very good interface with that sensor and that sensor may not have say for example, USB interface or R S 2 32 C interface like that. So, it can simply support one of these interfaces and this that device will be able to talk to this atmega series processor.

(Refer Slide Time: 18:28)



AVR Timer/Counter 0

- 8 Bit Up Counter
  - counts from 0 to 255 (0xFF), then loops to 0
  - Internal or External Clock source
- Prescaler
- Output capture through OC0, i.e. PB3, pin 4
- Interrupt on Overflow
  - Transition from 255 to 0 can trigger interrupt if desired

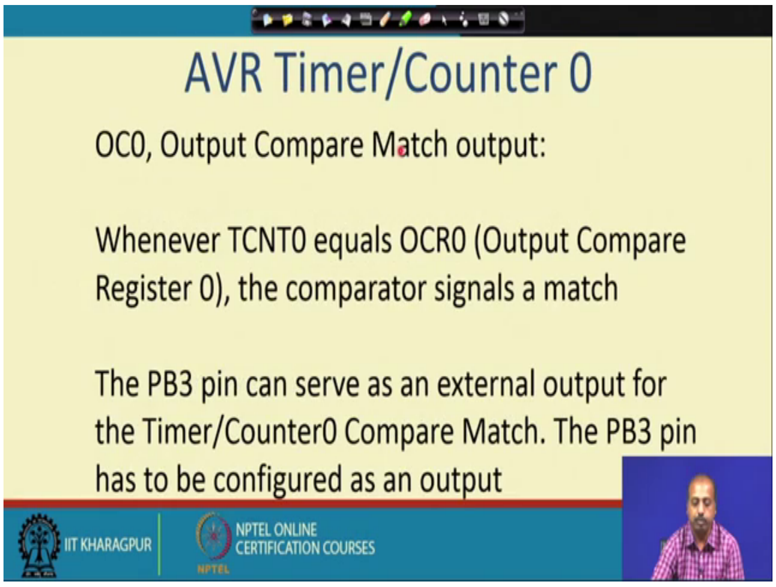
IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES

So, we have got this timer counter 0 it is 8 bit up counter it counts from 0 to 255 and then loops down to loops back to 0 the clock source can be internal or external. So, when it is a clock source. So, it is a timer when it to the clock source is external.

So, this is a counter there can be a prescalar. So, you can initialize this timer value to some initial value. So, that way it can start counting from there then output is captured through OC0 that is PB 3 pin 4. So, if there is an output overflow. So, that will be captured by that it will also generate an interrupt on overflow. So, whenever it makes a transition from 255 to 0 it can it can you can make it to trigger an interrupt ok.

So, you can make it to trigger an interrupt. Just like in 8051 we have seen that the timers can be programmed so that the interrupt occurs it here. So, here also the same thing you can make the interrupt to occur.

(Refer Slide Time: 19:26)


AVR Timer/Counter 0

OC0, Output Compare Match output:

Whenever TCNT0 equals OCR0 (Output Compare Register 0), the comparator signals a match

The PB3 pin can serve as an external output for the Timer/Counter0 Compare Match. The PB3 pin has to be configured as an output

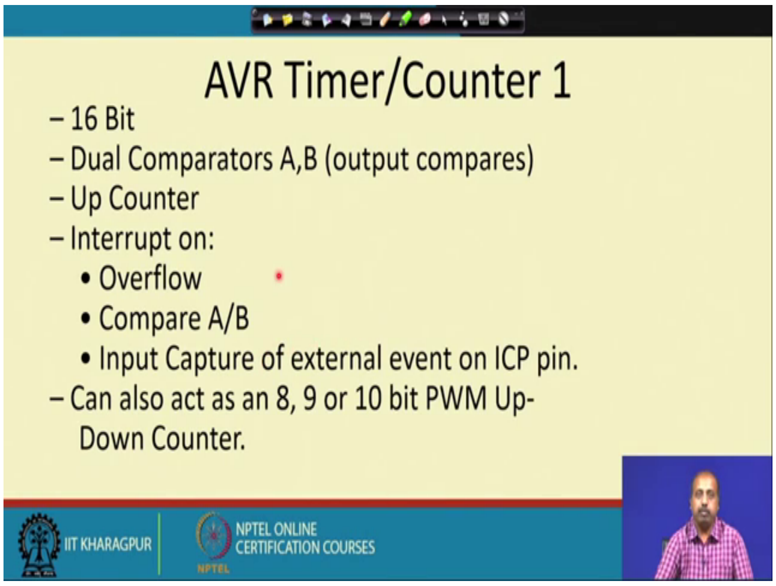
IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES

So, OCO is the output compare. So, this is a output compare match output. So, it is when the output matches whenever this TCNT0 will match with OCR0. So, there is the output compare register 0, so there will be this value will be shift there and whenever this count value becomes equal to this OCR0 register value output compare register value then this output compare match will become enable.

So, this can be used to solve as an external output for the timer counter compare match. So, internally this is the timer is doing something and if you want to detect some particular point of occurrence you can put the desired time values in the OCR register and then you can observe this OCO this OCO pin ok.

So, with the OCO pin say whenever this value the timer value will match with that OCR0 register then this counter timer counter 0 compare match. So, that output will be made high. So, to the external world you will know that there is a match in the current value. So, we can use it for triggering many other operation maybe in some other devices. So, we can trigger some operation. So, it can be used for that purpose.

(Refer Slide Time: 20:52)



AVR Timer/Counter 1

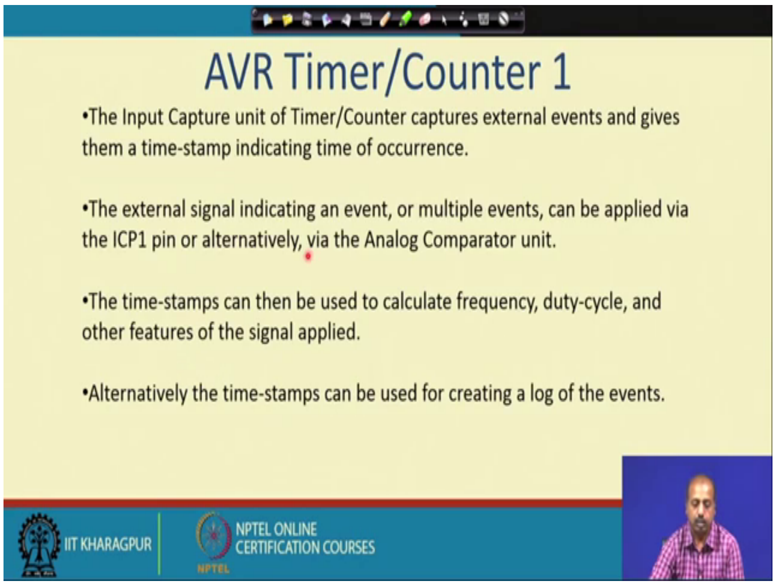
- 16 Bit
- Dual Comparators A,B (output compares)
- Up Counter
- Interrupt on:
  - Overflow
  - Compare A/B
  - Input Capture of external event on ICP pin.
- Can also act as an 8, 9 or 10 bit PWM Up-Down Counter.

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES

Then we have got another timer which is AVR counter timer counter one is a 16 bit timer. So, there are dual comparators A and B. So, for output comparison it is a up counter it provides an over for interruption overflow it compared with A B. So, if the comparison matches then also an interrupt will be generated and this input capture of external event on ICP pin.

So, there is an ICP pin, if there is some external value that we that for you can want to match. So, you can external event can be taken through the ICP pin when it is working as a counter mode. Can also; you can you can also be used to act as 8 9 or 10 bit pulse width modulation applications. So, if you want to modulate some pulse width for transmission. So, you can use it in up down counter mode in 8 9 or 10 bit pulse width modulation.

(Refer Slide Time: 21:48)



AVR Timer/Counter 1

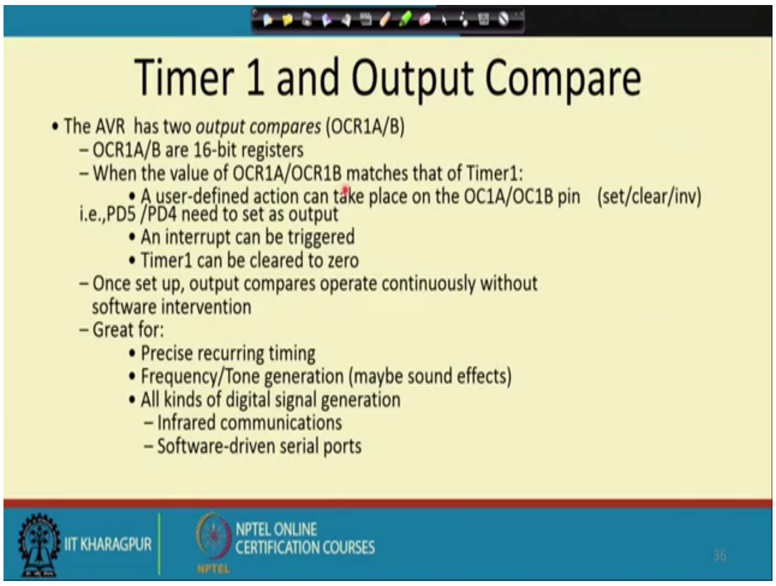
- The Input Capture unit of Timer/Counter captures external events and gives them a time-stamp indicating time of occurrence.
- The external signal indicating an event, or multiple events, can be applied via the ICP1 pin or alternatively, via the Analog Comparator unit.
- The time-stamps can then be used to calculate frequency, duty-cycle, and other features of the signal applied.
- Alternatively the time-stamps can be used for creating a log of the events.

The input capture unit of timer or counter it will capture external events and give them a time stamp indicating the time of occurrence. So, this is one important facility that it has. So, you can get the timestamp of occurrences of the events the external signal indicating an event or multiple events can be applied via this ICP 1 pin or via the analog comparator unit. So, that way you can give these external signals to the AVR microcontrollers.

The timestamps that we have got, it can be we can use it for calculating frequency duty cycle or other features of the signal. So, it is like the some signal is coming so we are trying to capture some feature of the signal. So, at different, as the values of the signals are changing, we captured the values of the signal and from the timer counter module. So, we can get the corresponding timestamps. So, later on we can use it for getting the frequency or some other features of the of the signal so that can be used for some other application some for signal processing application.

Alternatively, if they can also be used for creating a time logs, like say if you are if there is a telephone call, so how long the call has taken place. So, you can start when the call is starting. So, you can you can have the timestamp stored and then when the call is ending you can again get the timestamp. So, internally this timer is running so you can get the timestamps from there and you can get an information about the duration of the call. So, this way this time stamping feature can be utilized in different applications.

(Refer Slide Time: 23:28)



Timer 1 and Output Compare

- The AVR has two *output compares* (OCR1A/B)
  - OCR1A/B are 16-bit registers
  - When the value of OCR1A/OCR1B matches that of Timer1:
    - A user-defined action can take place on the OC1A/OC1B pin (set/clear/inv) i.e.,PD5 /PD4 need to set as output
    - An interrupt can be triggered
    - Timer1 can be cleared to zero
  - Once set up, output compares operate continuously without software intervention
  - Great for:
    - Precise recurring timing
    - Frequency/Tone generation (maybe sound effects)
    - All kinds of digital signal generation
      - Infrared communications
      - Software-driven serial ports

IIT KHARAGPUR NPTEL ONLINE CERTIFICATION COURSES

Then in timer 1, it has got two output compares OCR1 A and B. So, they are 16 bit registers ok. So, when the value of this OCR1 A or 1 B matches with timer one. So, user defined action can take place on the OCA OC1A or 1B pin. Like we can set or reset or invert the pin. So, you see this is a very interesting thing, like if whenever this matching will occur, so you can program the system.

So, that that p particular pin will get set or reset may be this is connected to some LED, that LED will glow that that match has occurred something like that. And interrupt can also be triggered and timer 1 can be cleared to 0. So, either you can, in other processors what was happening is that for in for example, in 8051. So, timer starts and when the timer overflows then it gives an interrupt and that way.

But in between time values, if you want to get some stamping like if you want to get some notification in between that is not going to happen. So, you have to stop the timer with that time value then only you can get that notification, but in this case it is more flexible because I can have this timer timers programmed in such a fashion that when that time value reaches a particular value set in the OCR register.

So, we can initiate some action, like setting clearing bits or setting an interrupt or stopping the timer by resetting it to 0. Once setup the output will compare the output compares operate continually without software intervention. So, you just have to tell that I need to compare this. So, once that is done. So, it will be continually doing.

So, with precise recurring timing, this is this can be used for recurring timing say when the event is occurring again. So, you can find out frequency or tone generation for sound effects at some particular frequency you want to generate some tones ok. So, for different I may have a number of sound as different number of different sounds to be produced and each of them will occur at some different periodic boundaries ok.

So, that way I can do that by controlling this output capture part and then the digital signal generation like infrared communication then software driven serial ports. So, though wherever we need some sort of periodic thing to occur continually like in 8051 we had had that mode 2 where it was a auto reload type of operation, but here it is not required. So, here it is you can do this OCR and this output a timer 1 for doing this operation.