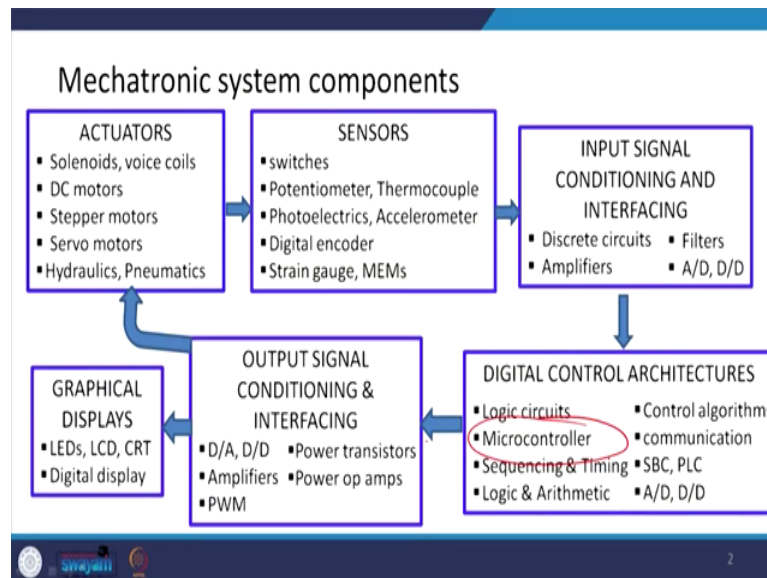


Mechatronics
Prof. Pushparaj Mani Pathak
Department of Mechanical and Industrial Engineering
Indian Institute of Technology, Roorkee

Lecture - 24
Microcontroller

I welcome you all to this NPTEL online certification course on Mechatronics. Today, we are going to talk about Microcontrollers. Here in this schematic of the different mechatronic system components, you can find a microcontroller over here in the digital control architecture.


(Refer Slide Time: 00:49)



(Refer Slide Time: 00:56)

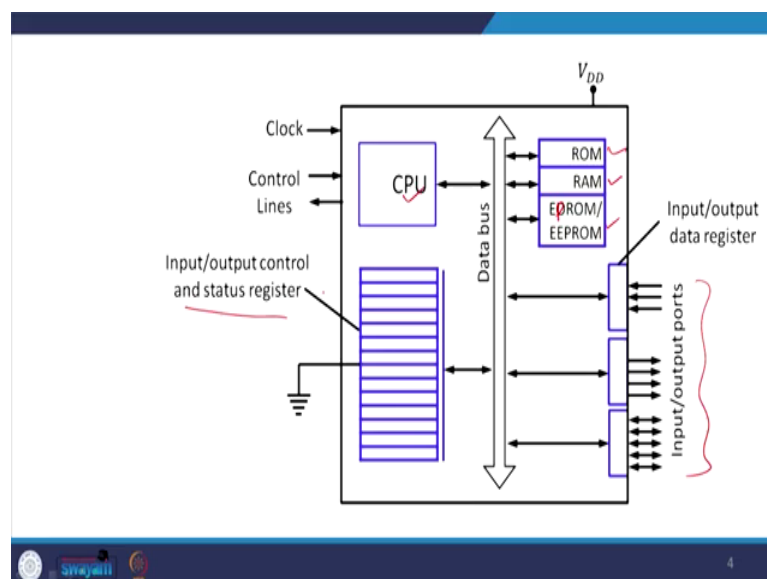
Microcontroller

- Microcontroller is the integration of a microprocessor with memory and input/output interfaces, and other peripherals such as timers, on a single chip.
- Microcontrollers have limited amounts of ROM and RAM and are widely used for embedded control systems.
- A microprocessor system with separate memory and input/output chips is more suited to processing information in a computer system.

The footer of the slide contains a logo on the left, the text "Sivajini" in the center, and the number "3" on the right.

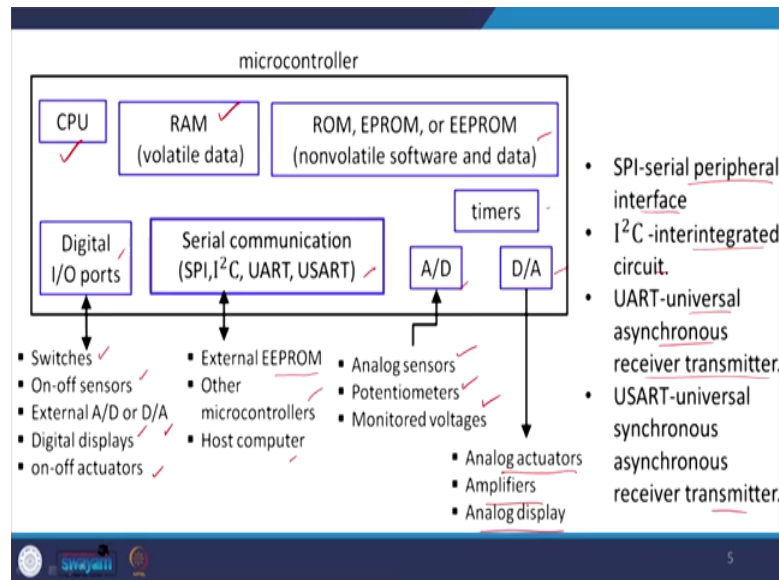
Microcontrollers, as I explained to you in my previous lecture, is an integration of microprocessor with memory and input-output interfaces and other peripherals such as timers on a single chip. The microprocessor has a limited amount of ROM and RAM and is widely used for embedded control systems. A microprocessor system with separate memory and the input-output chip is more suited to processing information in a computer system.

(Refer Slide Time: 01:51)



So, here you can see we have the CPU, we have the ROM, we have the RAM, and we have the EEPROM and EOPROM so, you have input and output data register here, input-output ports are there, and input-output control and status registers are here, and you have the data bus over here.

(Refer Slide Time: 02:16)



So, if we look at further little more detail on the microcontroller here, we can see that we have CPU, RAM, which is volatile data, and ROM, EPROM, and EEPROM. As I defined these memories devices in my last lecture, these are non-volatile software and data. We have digital input-output ports, and then the serial communication units are here, a timer is there and analog to digital as well as the digital to analog converter over here. The digital to analog converter, as you can see that these are the devices that use the analog signals that are the analog actuators, amplifier, and analog display, an analog to digital converter is needed if the input is there from analog sensor, potentiometers, or the monitored voltages.

And the serial communication here is for external EEPROM or other microcontroller host computers, and the digital input-output ports are used for switches, on-off sensors, external analog to the digital or digital to analog converter, digital display as well as on-off actuators.

Here are the various terms used over here, SPI is Serial Peripheral Interface. These are all for serial communication. I²C that is the inter-integrated circuit, UART that is Universal

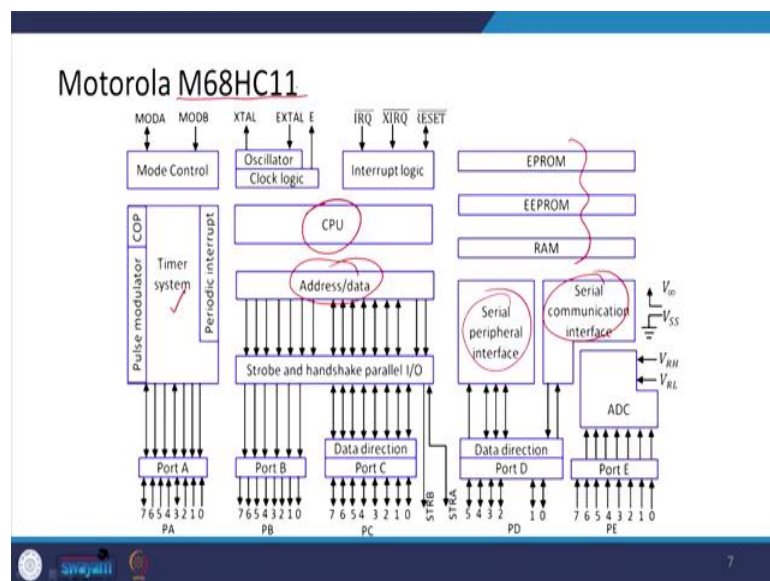
Asynchronous Receiver Transmitter, and USART Universal Synchronous Asynchronous Receiver Transmitter.

(Refer Slide Time: 04:16)

- 8-bit microcontrollers – Motorola 68HC11, Intel 8051 and PIC16C6x/7x .
- 16-bit microcontroller – Motorola 68HC16.
- 32-bit microcontroller – Motorola 68300.

8-bit microcontrollers, here are some examples a Motorola 68HC double 11, Intel 8051, and a PIC16C 6 x/7 x, 16-bit microcontroller are Motorola 68HC16, 32-bit microcontroller is Motorola 68300. So, these are an example of the microcontroller.

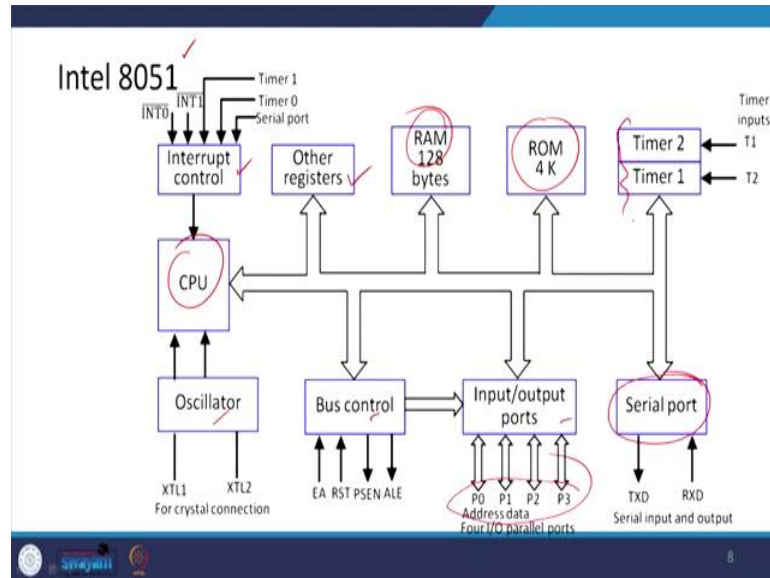
(Refer Slide Time: 04:50)



So, if we look at M16H 68HC11 Motorola microcontroller, so, this is the architecture. So, you have the CPU, you have all the memory devices over here, and you have a time timer

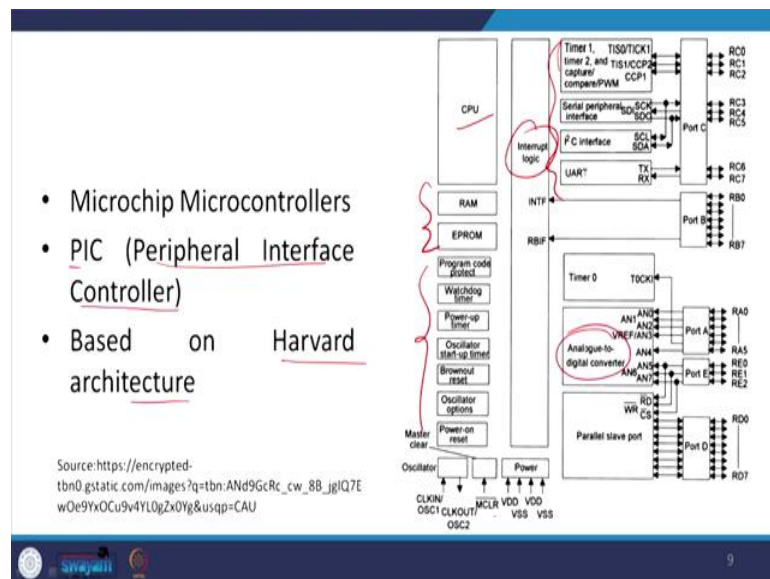
system over here, and you have the serial peripheral interface, serial communication interface all these and here address, and data is there. So, this is the architecture for a Motorola M68HC11.

(Refer Slide Time: 05:35)



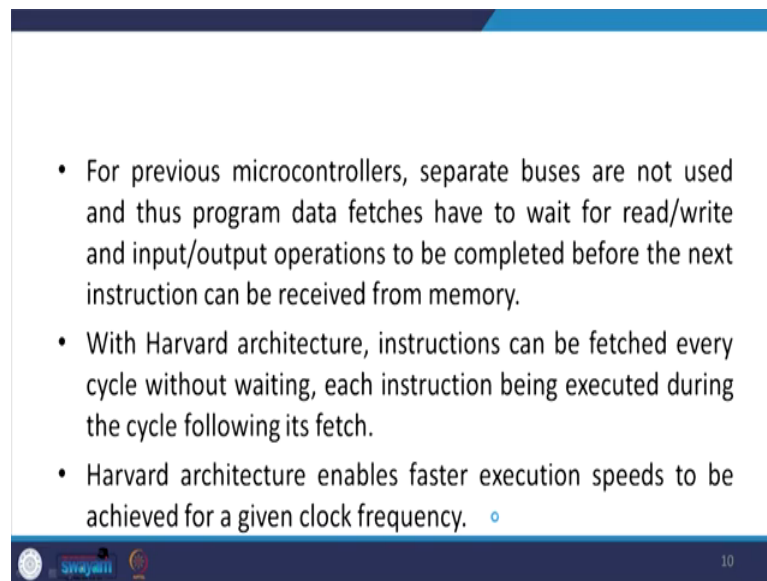
Similarly, we can see the architecture for the Intel 8051 microcontroller. So, again we have the CPU, we have the RAM, we have the ROM, interrupt control is there, a register is there, an oscillator is there, bus control input-output ports are here, as well as the serial ports are there, and here you have the timer.

(Refer Slide Time: 06:01)



One of the very popular microchip microcontrollers is the PIC which is a Peripheral Interface Controller, and this is based on the Harvard architecture, and again here we can see the various components of it. So, we have CPU is there, then you have the memory devices over here, then other program code protector watchdog timer, power-up timer, oscillator start-up timer and all, then what? We have to interrupt logic over there, and here all interfaces are there, which I discussed during the last few slides, the A to D converter, and all are there.

(Refer Slide Time: 07:02)



So, for the previous microcontroller, separate buses are not used, and thus program data features have to wait for read-write and input-output operations to be completed before the next instruction can be received from the memory, but with Harvard architecture, instructions can be fetched every cycle without waiting, and each instruction is executed during the cycle following its fetch. Harvard architecture enables faster execution speed to be achieved for a given clock frequency.

(Refer Slide Time: 07:43)

Atmel AVR microcontrollers and Arduino

- Atmel AVR microcontrollers uses modified Harvard architecture with the program and data stored in separate memory locations.
- Arduino is a small microcontroller board with complementary components which has been designed to facilitate the use of the microcontroller in control projects.
- Arduino board is open source.

11

Atmel AVR microcontroller and Arduino: Let us look at these, the features of these. So, the Atmel AVR microcontroller uses a modified Harvard architecture with the program, and data is stored in separate memory locations. Arduino is a small microcontroller board with complementary components which has been designed to facilitate the use of the microcontroller in control projects. Arduino board is open source.

(Refer Slide Time: 08:16)

PIC 16F84 Block diagram

Source:
<https://ww1.microchip.com/downloads/en/devicedoc/35007b.pdf>

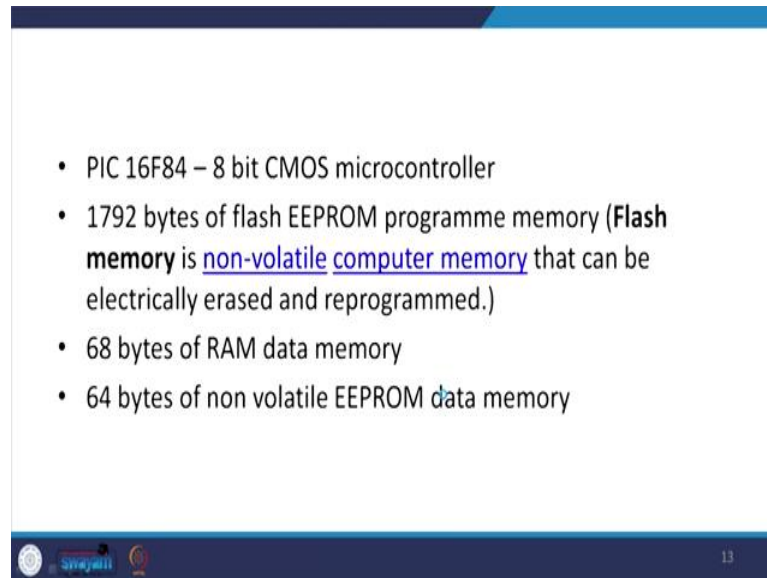
The diagram illustrates the internal architecture of the PIC 16F84 microcontroller. Key components include: a 13-bit Program Counter connected to a 16K x 14-bit FLASH Program Memory; an 8-level Stack (13-bit); 68 x 8-bit RAM File Registers; an 8-bit Instruction Register; an 8-bit Data Bus; an 8-bit ALU; a Watchdog Timer; an Oscillator; a Power-up Reset; a Timer; and various I/O Ports (RA4/TOCKI, RA3/RA0, RB7/RB1, RB0/INT). The diagram also shows connections for OSC2/CLKOUT, OSC1/CLKIN, MCLR, VDD, and VSS.

12

So, this is the block diagram for PIC 16F84 microcontroller. So, again here, we could identify the different components of it over here. So, you have RAM is there, then the

instruction register is there, power-up timer, oscillator start-up timer all timers are there here. Then instruction decode and control timing generation are there. So, here you have the EEPROM data memory is there, flash program memory is over here. So, you have the ALU over here. So, this is how this indicates the block diagram for PIC 16F84 microcontroller.

(Refer Slide Time: 09:19)



This PIC 16F84 is an 8 bit CMOS microcontroller, and it has 1792 bits of flash EEPROM program memory, and this flash memory is non-volatile computer memory that can be electrically erased and reprogrammed. It has 68 bytes of RAM data memory, and 64 bytes are non-volatile EEPROM data memory.

(Refer Slide Time: 09:52)

- PIC16F84 pin-out and required external components
- 18 pin DIP (Dual in line package) IC.
- Bidirectional lines can be individually configured in software as i/p or o/p.

Here we can see the pin-out connections of PIC 16F84 with the required external components and you can see that it is a dual inline package. So, we have the two lines over here, and there is a total 18 number of pins. So, you can see 1 to 9 on one side and 10 to 18 on the other side. So, it is a dual inline package IC, and bidirectional lines can be individually configured in software as input or output pins.

(Refer Slide Time: 10:30)

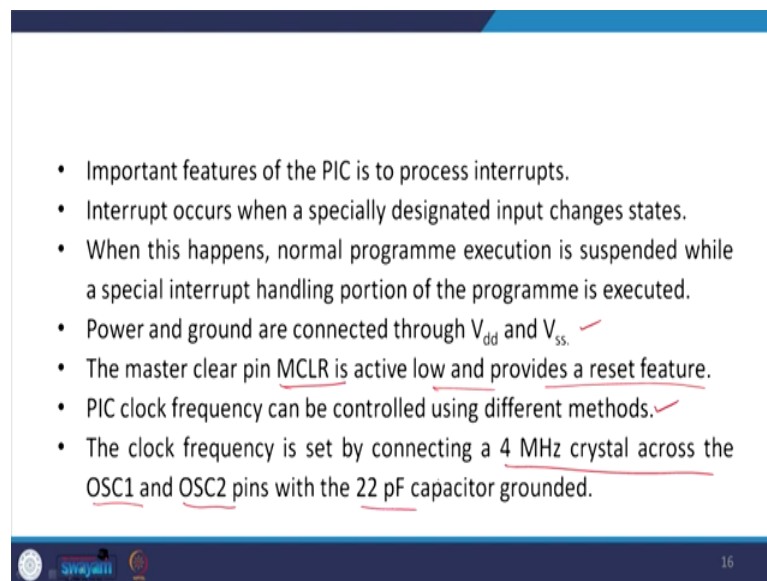
PIC16F84 pin name descriptions

| Pin Identifier | Description |
|-------------------|--|
| RA [0 – 4] | 5 bits of <u>bidirectional</u> I/O (PORTA) |
| RB [0 – 7] | 8 bits of <u>bidirectional</u> I/O (PORTB) |
| V_{ss}, V_{dd} | Power supply ground reference (ss: source) and positive supply (dd: drain) |
| OSC1, OSC2 | Oscillator crystal <u>inputs</u> |
| \overline{MCLR} | Master clear (active low) ✓ |

So, that we can do now here if you look at there is 5 bits of a bi-directional input-output pin which we call as PORT A . So, this is PORT A, RA 01 and 2 or 3 and 4. So, these

form what we call PORT A, and 8-bit bidirectional input-output pins are PORT B. So, here you can see that there is 8 bit 1, 2, 3, 4, 5, 6, 7, 8. This is 8-bit bidirectional input-output pins of port V; V_{ss} and V_{dd} . As you can see here, V_{dd} and V_{cc} are power supply ground references. So, ss represents the source and positive supply. So, this is V_{ss} and V_{dd} , and you have OSC1 and OSC2 that that is the crystal oscillator crystal inputs over here, and there is a master clear port which is active low.

(Refer Slide Time: 12:16)

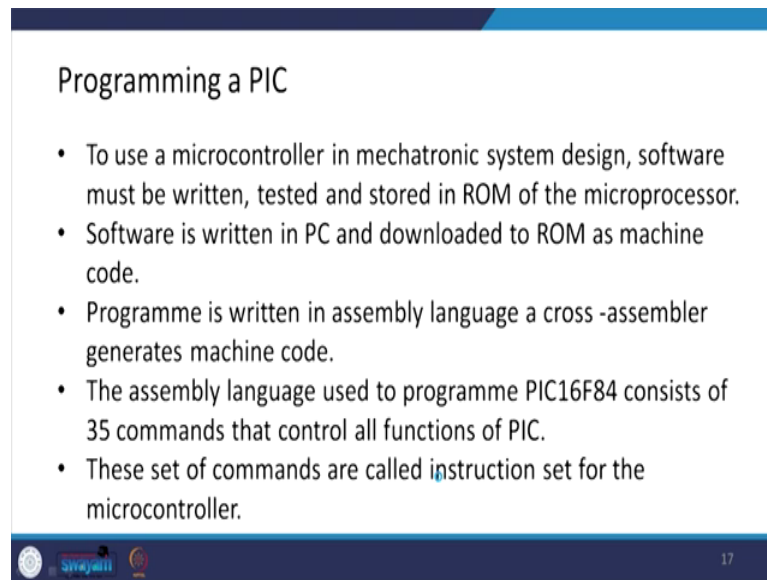


- Important features of the PIC is to process interrupts.
- Interrupt occurs when a specially designated input changes states.
- When this happens, normal programme execution is suspended while a special interrupt handling portion of the programme is executed.
- Power and ground are connected through V_{dd} and V_{ss} . ✓
- The master clear pin MCLR is active low and provides a reset feature.
- PIC clock frequency can be controlled using different methods. ✓
- The clock frequency is set by connecting a 4 MHz crystal across the OSC1 and OSC2 pins with the 22 pF capacitor grounded.

So, this is the pin-out connection for PIC 16F84. An important feature of PIC is to process interrupts that occur when a specially designed input changes states, and when this happens, the normal program execution is suspended while a special interrupter handling portion of the program is executed and the power and grounds are connected through V_{ss} and V_{dd} ports, as I said.

The master clear pin MCLR is active low and provides a reset feature here, and PIC clock frequency can be controlled using different methods. The clock frequency is set by connecting a four mega Hertz crystal across the OSC1 and OSC2 pins with the 22 Pico Farad capacitor grounded.

(Refer Slide Time: 13:07)



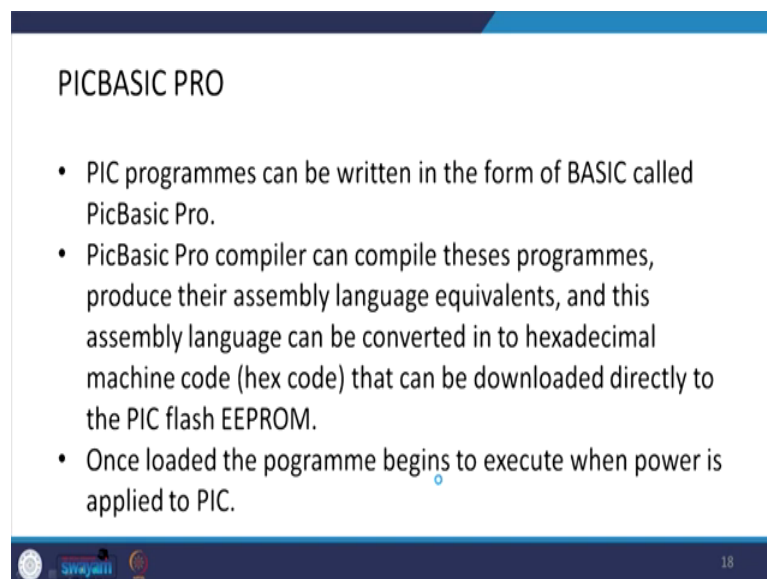
Programming a PIC

- To use a microcontroller in mechatronic system design, software must be written, tested and stored in ROM of the microprocessor.
- Software is written in PC and downloaded to ROM as machine code.
- Programme is written in assembly language a cross -assembler generates machine code.
- The assembly language used to programme PIC16F84 consists of 35 commands that control all functions of PIC.
- These set of commands are called instruction set for the microcontroller.

swajati 17

Now, let us look at the programming of a PIC. To use a microcontroller in mechatronics system design, software must be written, tested, and stored in the ROM of the microprocessor. Software is written in PC and downloaded to the ROM as machine code, and the program is written in assembly language. A cross-assembler generates the machine code. The assembly language used to program PIC16F 84 consists of 35 commands that control all functions of PIC. This set of commands is called an instruction set for the microcontroller.

(Refer Slide Time: 14:04)



PICBASIC PRO

- PIC programmes can be written in the form of BASIC called PicBasic Pro.
- PicBasic Pro compiler can compile theses programmes, produce their assembly language equivalents, and this assembly language can be converted in to hexadecimal machine code (hex code) that can be downloaded directly to the PIC flash EEPROM.
- Once loaded the pogramme begins to execute when power is applied to PIC.

swajati 18

Now, let us look at PicBasic Pro. PIC program can be returned in the form of BASIC, and this is called PicBasic Pro. PicBasic Pro compiler can compile these programs, produce their assembly language equivalents, and this assembly language can be converted into the hexadecimal machine code or hex code that can be downloaded directly to the PIC flash EEPROM, and once loaded, the program begins to execute when the power is applied to a PIC.

(Refer Slide Time: 14:49)

| Math operator or function | Description |
|---------------------------|-------------------------------|
| A + B ✓ | Add A AND B |
| A - B | Subtract B from A |
| A * B | Multiply A and B |
| A / B | Divide A by B |
| A << n | Shift A n bits to the left |
| A >> n | Shift A n bits to the right |
| COS A | Return the cosine of A |
| A MAX B | Return the maximum of A and B |

So, here are the basic math operator or functions with their description A+ B is for adding A AND B, this is for subtracting subtraction of B from A, this is for multiplication, division. This means shift A n bits to the left, and this means shift A n bits to the right, and here return the cosine A, A max B returns the maximum of A and B.

(Refer Slide Time: 15:16)

| Math operator or function | Description |
|---------------------------|--|
| A MIN B | Return the minimum of A and B |
| SIN A | Return the sine of A |
| SQR A | Return the square root of A |
| A & B | Return the bitwise AND of A and B |
| A B | Return the bitwise OR of A and B |
| A ^ B | Return the bitwise Exclusive OR of A and B |
| ~ A | Return the bitwise NOT of A |

Then A min B returns the minimum of A and B, then sin A returns the sine of A, square SQR A returns the square root of A, and A and B return the bitwise AND of A and B, and A | B represents returns the bitwise or of A and B, and this one returns the bitwise exclusive OR of A and B, and ~A returns the bitwise NOT of A.

(Refer Slide Time: 15:58)

PicBasic Pro Logical Comparison Operators

| Operator | Description |
|------------|--------------------------|
| = or == ✓ | equal ✓ |
| <> or != ✓ | not equal |
| < ✓ | less than |
| > | greater than |
| <= | less than or equal to |
| >= ✓ | greater than or equal to |

There is some PicBasic Pro logic comparison operator. So, one '=' or '==' sign represents equal, and this is the representation for not equal. This is for less than, and this is far greater than, and this is less than or equal to, and this is for greater than or equal to.

(Refer Slide Time: 16:24)

| Statement | Description |
|--|---|
| @ assembly statement | Insert one line of assembly language code |
| ADCIN channel, var | Read the on-chip analog to digital converter |
| ASM . . . ENDASM | Insert an assembly language code section consisting of one or more statements |
| BRANCH index. {label' {, label2, ...}} | Computed goto that jumps to a label based on index |
| BRANCHL index. {label1 {, label2, ...}} | Branch to a label that can be outside of the current page of code memory (for PICs with more than 2 k of program ROM) |
| BUTTON pin, down_state, auto_repeat_delay, auto_repeat rate, countdown variable, action_state label CALL assembly_label | Read the state of a pin and perform debounce (by use of a delay) and autorepeat (if used within a loop) |

And then, we can similarly have various statements and their description. For example, you can refer to some of the standard textbooks and the textbook which I will be giving at the end of this lecture in the references to look at some of the descriptions of these statements.

At assembly statement, insert one line of assembly language code ADCIN channel were read on-chip analog to digital converter like this. So, likewise, there are various functions whose description you can read.

(Refer Slide Time: 17:13)

| Statement | Description |
|---|--|
| CALL assembly_label | Call an assembly language subroutine |
| CLEAR | Zero all variables |
| CLEARWDT | Clear the watch-dog timer |
| COUNT pin, period, var | Count the number of pulses occurring on a pin during a period |
| DATA {@ location,} constant1 {, constant2, .} | Define initial contents of the on-chip EEPROM (same as the EEPROM statement) |
| DEBUG item1 {, item2, ...} | Asynchronous serial output to a pin at a fixed baud rate |
| DEBUGIN {timeout, label,} [item1 {item2, }] | Asynchronous serial input from a pin at a fixed baud rate |
| DISABLE | Disable ON INTERRUPT and ON DEBUG processing |
| DISABLE DEBUG | Disable ON DEBUG processing |

CALL assembly label, call an assembly language subroutine; CLEAR zero all variables
 CLEARWDT that clear the watchdog timer, COUNT pin, period, var number of pulses
 occurring on a pin during a period, then DATA define initial content of on-chip EEPROM,
 all these things are there.

(Refer Slide Time: 17:39)

| Statement | Description |
|---|--|
| DISABLE INTERRUPT | Disable ON INTERRUPT processing |
| DTMFOUT pin, {on_ms, off_ms,} {tone1 {, tone2, ..}} | Produce touch tones on a pin |
| {EEPROM {@ location,} constant1 {, constant2, ..}} | Define initial contents of on-chip EEPROM (same as the DATA statement) |
| ENABLE | Enable ON INTERRUPT and ON DEBUG processing |
| ENABLE DEBUG | Enable ON DEBUG processing |
| ENABLE INTERRUPT | Enable ON INTERRUPT processing |
| END | Stop execution and enter low power mode |
| FOR count = start TO end {STEP {-} inc} {body statements} | Repeatedly execute statements as count goes from start to end in fixed increment |

Then similarly, I am just highlighting some of the important ones. So, ENABLE means to enable ON INTERRUPTS and ON DEBUG processing, ENABLE DEBUG to enable ON DEBUG processing, and END means stop execution and enter low power mode. And then we have the for we will look for a loop. So, the count starts TO end a STEP, and then you can have the body statement.

(Refer Slide Time: 18:09)

| Statement | Description |
|--|--|
| FOR count = start TO end {STEP {-} inc} {body statements} | Repeatedly execute statements as count goes from start to end in fixed increment |
| NEXT {count} FREQOUT pin, on_ms, freq1 {, freq2} | Produce up to two frequencies on a pin |
| GOSUB label | Call a PicBasic subroutine at the specified label |
| GOTO label | Continue execution at the specified label |
| HIGH pin | Make pin output high |
| HSERIN {parity_label,} {time_out, label,} [item1 {, item2, ...}] | Hardware asynchronous serial input (if there is a hardware serial port) |
| HSEROUT [item1 {, item2, ...}] | Hardware asynchronous serial output (if there is a hardware serial port) |

Similarly, repeatedly execute a statement as count goes from start to end in a fixed increment, produce up to two frequencies on a pin, then you have the GOTO label continue execution at the specified label, high pin that makes pin output high, then we can have the conditionally jumped to a label.

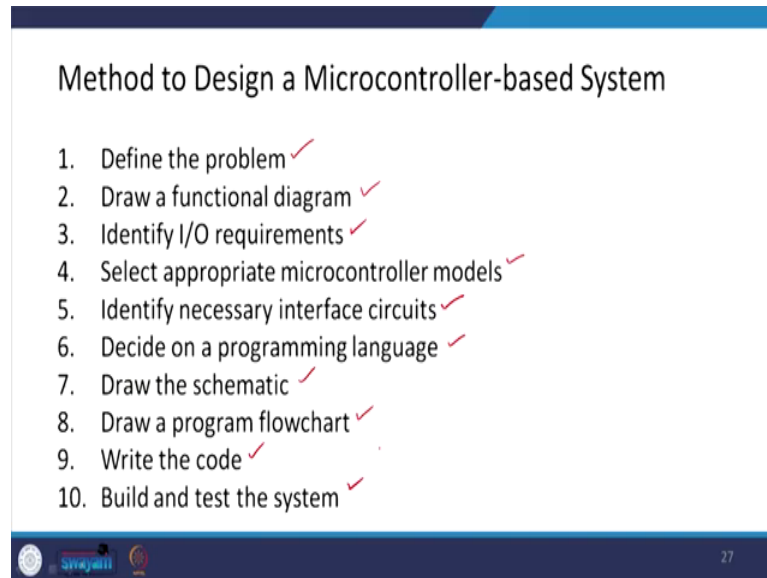
(Refer Slide Time: 18:32)

| Statement | Description |
|--|--|
| IF log_comp THEN label ✓ | Conditionally jump to a label ✓ |
| IF log_comp THEN true_statements ELSE false_statements ENDIF ✓ | Conditional execution of statements ✓ |
| INPUT pin ✓ | Make pin an input ✓ |
| LCDIN {address,} {var1 {, var2, .}} | Read RAM on a liquid crystal display (LCD) |
| LCDOUT item1 {, item2, .} | Display characters on LCD |
| {LET} var = value | Assignment statement (assigns a value to a variable) |
| LOW pin ✓ | Make pin output low ✓ |
| OUTPUT pin | Make pin an o/p ✓ |

So, if log comp THEN labels something like this and conditional execution of a statement that is IF and ELSE type of statement and it terminates with the ENDIF statement, and

INPUT pin, making a pin an input, LOW pin makes pin output, low OUTPUT pin making pin and output and so on, so likewise we can have.

(Refer Slide Time: 19:04)



There are many more commands which I have not discussed over here. As I said, you can refer to some standard textbook or PIC 16F84 catalog to know more number of programs.

Then now, let us see how do we design a microcontroller-based system. I am taking a small example over here. In the last week of this course, I have taken around five examples based on the micro-mechatronic system design examples. There you can find out further details about this. So, the method for the design of a microcontroller-based system starts with this; first of all, you have to define the problem and then draw the functional diagram. After the functional diagram, you have to find out what are your input-output requirements and once input-output requirements are there, select an appropriate microcontroller model, select identify necessary interface circuit, decide on a programming language, draw the schematic, draw a program flowchart, write the code and then build and test the system. So, these ten steps can summarize the design of a microcontroller-based system. Now, a factor is considered while selecting a microcontroller.

(Refer Slide Time: 20:55)

Factors Considered While Selecting a Microcontroller

1. Number of input/output pins ✓
2. Interfaces required ✓
3. Memory requirements ✓
4. The number of interrupts required ✓
5. Processing speed required ✓

So, these factors on which the selection of a microcontroller depends. How many numbers of input-output pins do you need, what are your interface requirement, what is your memory requirement, how many interrupts are needed, and what processing speed is needed. So, based on these, you can select an appropriate microcontroller for a particular application.

(Refer Slide Time: 21:17)

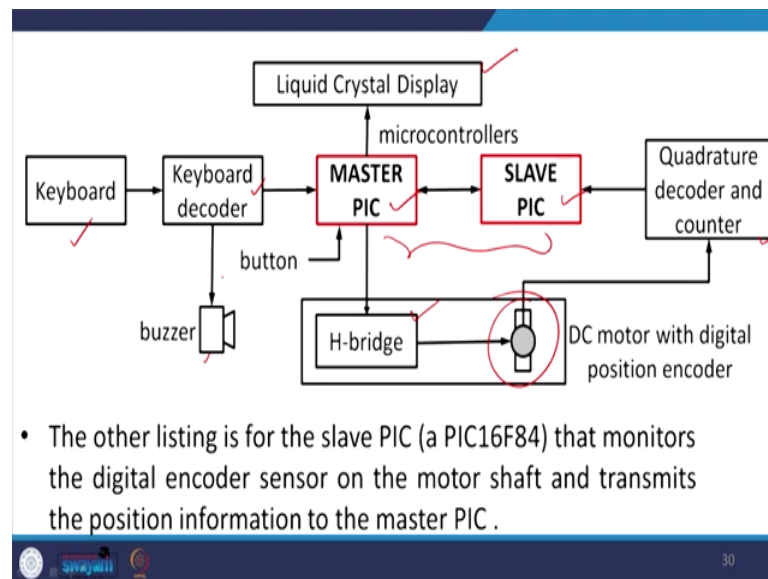
Example: DC motor position and speed controller

- This solution utilizes two PIC microcontrollers. ✓
- The main PIC is called "master" PIC and controls most of the system functions.
- The secondary PIC is called as "slave" PIC, because it provides information to the master PIC upon command.
- There are two software listings. One is for the master PIC (a PIC16F88) that monitors the keypad, provides a menu-driven user interface on the LCD and drives the motor.

Now, let us take an example of the DC motor position and the speed controller. So, in most of our micro-mechatronics system designs, we often use DC motors so, whether it is to make a simple mobile robot or you want to make it a simple manipulator, the DC motors are used. So, I have taken the DC motor position and the speed controller example. Now,

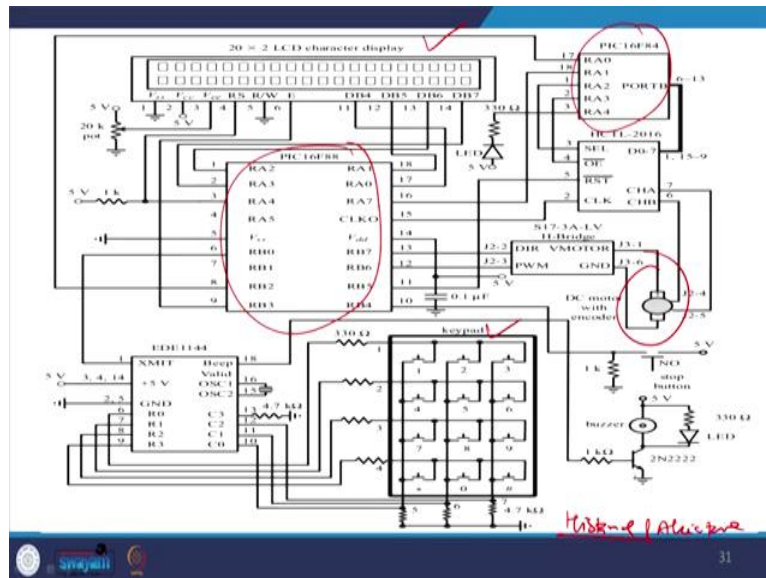
this solution utilizes my PIC microcontroller. The main PIC is called the master PIC and control most of the system functions, and the secondary PIC is called the slave PIC because it provides the information to the master PIC upon the command. There are two software listings; one is for the master PIC that monitored the keypad, provides a menu-driven user interface on the LCD, and drives the motors.

(Refer Slide Time: 22:37)



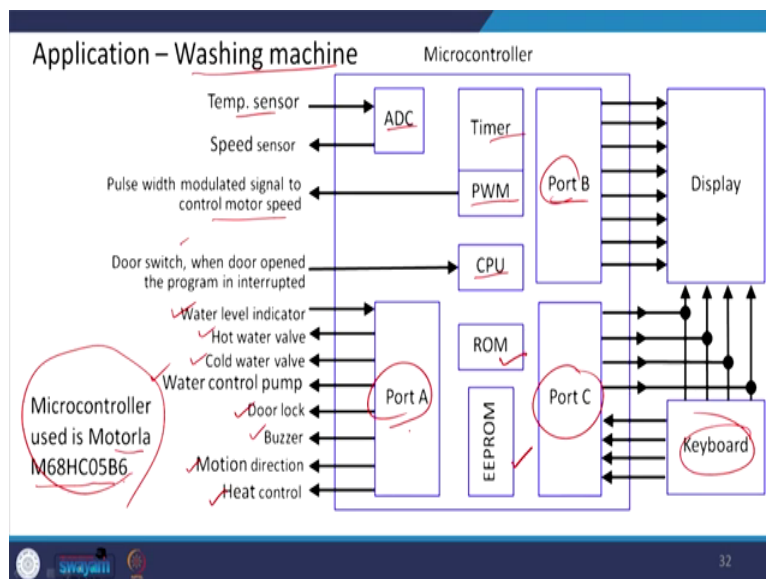
So, here you can see that you have a master PIC. You have a slave PIC. We can have an LCD display over here. So, these are the microcontroller, and I have a DC motor here whose position and speed I want to control. The input for this is through the H bridge, I had talked a lot when I was discussing the actuators, and the output of the motor or the speed of the motor can be taken over from over here. So, a quadrature decoder and counter through this can be there, and then it could be offered to the slave PIC. For providing the input, we have the keyboard, and the keyboard decoder could be there to decode the commands of the keyboard, and you can have a buzzer input. So, this is there. So, the other listing of the slave PIC monitors the digital encoder sensor on the motor shaft and transmits the position information to the master PIC, as I said.

(Refer Slide Time: 23:57)



So, this picture which I have taken from Histan book. So, you can see the connections for this. So, this is your master PIC, and you have the slave PIC over here, you had the display unit over here, here is the keypad here is the motor. So, the other peripheral connections can be made, and for its programming, you, please refer to Alciatore and Histan for reference and the programming part of it.

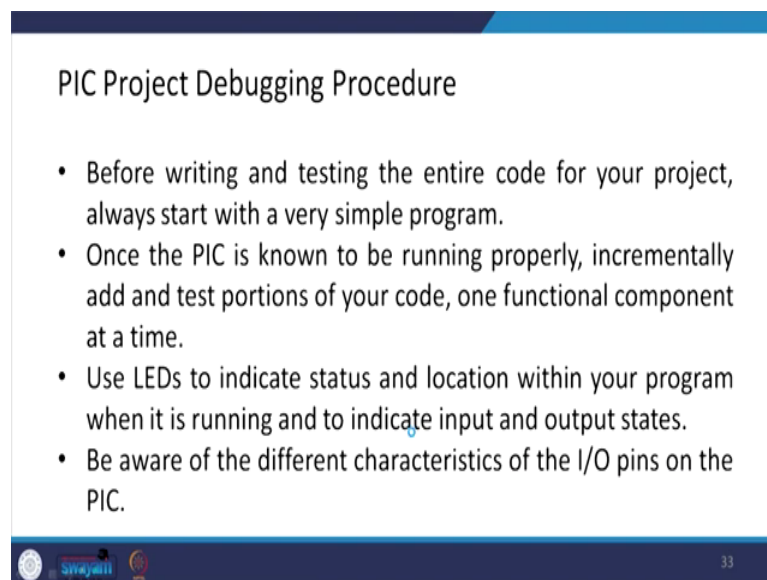
(Refer Slide Time: 24:57)



The next application of microcontrollers we can see in a washing machine that uses a microcontroller used is Motorola M68 HC05B6. So, that is there. So, again here we have

the temperature sensor and speed sensor. They are connected to the analog to digital converter. You have the timer pulse width modulation there to control the motor speed, you have the CPU, you have the memory ROM and EEPROM here, and there is the port A, port B, and port C . So, port B is for the display, port C is for the input that is keyboards and in port A we can have connections for our for heat control motion direction buzzer, door lock, water control pump, cold water valve, hot water valve as well as the water level indicator. The CPU has this door switch. When the door opens, the program is interrupted. So, all these things can this block diagram explains the use of the microcontroller in the washing machine.

(Refer Slide Time: 26:26)

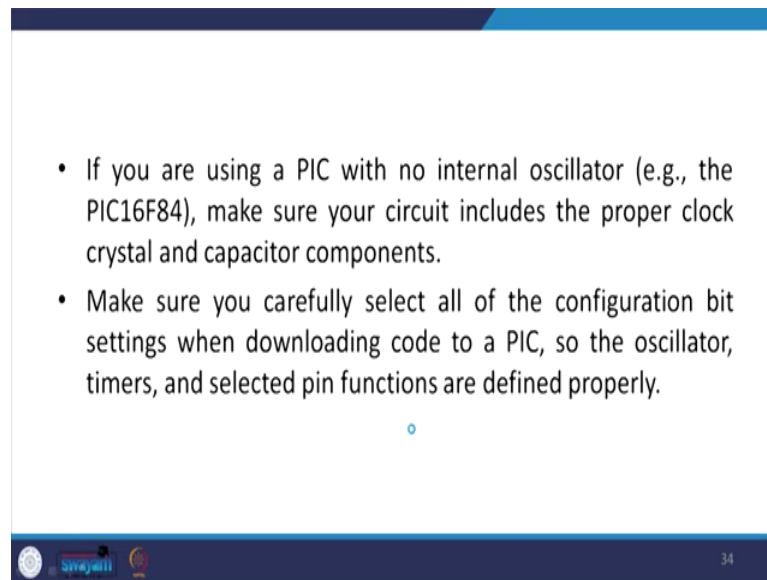


PIC Project Debugging Procedure

- Before writing and testing the entire code for your project, always start with a very simple program.
- Once the PIC is known to be running properly, incrementally add and test portions of your code, one functional component at a time.
- Use LEDs to indicate status and location within your program when it is running and to indicate input and output states.
- Be aware of the different characteristics of the I/O pins on the PIC.

So, for our PIC project above, there could be a systematic debugging procedure for the PIC project so that the mistakes are minimized. So, what we do is that before writing and testing the entire code of your project, always start a very simple program. So, this is always advisable. Once the PIC is known to be running properly, then incrementally add and test portions of your code one functional component at a time. Use LEDs to indicate status and location within your program when it is running and to indicate input and output states. Be aware of the different characteristics of the input-output pins on the PIC.

(Refer Slide Time: 27:21)



Slide 34 contains two bullet points:

- If you are using a PIC with no internal oscillator (e.g., the PIC16F84), make sure your circuit includes the proper clock crystal and capacitor components.
- Make sure you carefully select all of the configuration bit settings when downloading code to a PIC, so the oscillator, timers, and selected pin functions are defined properly.

The slide footer includes the Swayam logo and the number 34.

If you are using a PIC with no internal oscillators, make sure your circuit includes the proper clock crystal and capacitor component. Make sure you carefully select all the configuration bits set when downloading the code or to a PIC. So, the oscillator timer and selected pins functions are defined properly.

(Refer Slide Time: 27:46)



Slide 35 is titled "References" and lists three references:

- W. Bolton, Mechatronics: Electronic Control Systems in Mechanical and Electrical Engineering (6th Edition), Pearson, 2015
- R. Merzouki, A. K. Samantaray, P. M. Pathak, B. Ould Bouamama, Intelligent Mechatronic Systems: Modeling, Control and Diagnosis, ISBN 978-1-4471-4627-8, 2013, Springer, London
- D.G. Alciatore and Michael B. Hiestand, Introduction to Mechatronics, Tata Mc Graw Hill, 2012.

The slide footer includes the Swayam logo and the number 35.

So, these are the references which you can see for the further readings, such as Mechatronics by Bolton and Mechatronics by Alciatore and Hiestand.

Thank you.