

Module 3: Programmable Logic Devices (PLDs)

Lecture 1

Introduction to Micro-processors and Micro-controllers

1. Introduction

Programmable Logic Devices (PLD) are programmable systems and are generally used in manufacturing automation to perform different control functions, according to the programs written in its memory, using low level languages of commands. There are following three types of PLDs are being employed in mechatronics systems.

1. Microprocessor

It is a digital integrated circuit which carries out necessary digital functions to process the information obtained from measurement system.

2. Microcomputer

It uses microprocessor as its central processing unit and contains all functions of a computer.

3. Programmable Logic Controller (PLC)

It is used to control the operations of electro-mechanical devices especially in tough and hazardous industrial environments.

A typical programmable machine has basic three components as shown in Figure 3.1.1:

1. Processor, which processes the information collected from measurement system and takes logical decisions based on the information. Then it sends this information to actuators or output devices.
2. Memory, it stores
 - a. the input data collected from sensors
 - b. the programs to process the information and to take necessary decisions or actions. Program is a set of instructions written for the processor to perform a task. A group of programs is called software.

3. Input/output devices: these are used to communicate with the outside world/operator.

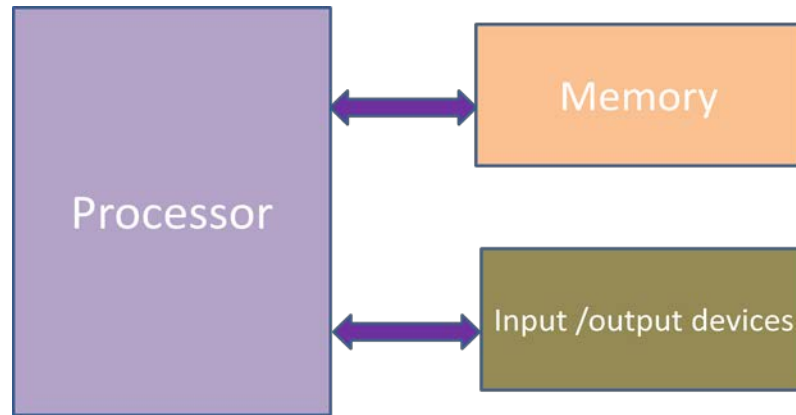


Figure 3.1.1 Components of a programmable logic device

2. Microprocessor

It is a multi-purpose, programmable device that reads binary instructions from a storage device called memory, processes the data according to the instructions, and then provides results as output. In common practice it is also known as CPU (central processing unit). CPU can be referred as complete computational engine on a single chip. First Microcontroller, Intel 4004 was launched in 1971. It was able to process just 4 bits. It started a new era in electronics engineering. Microprocessor chip was one of the important inventions of the 20th century. Table 3.1.1 shows the history of micro-processors.

Table 3.1.1 History of Micro-Processors

Name	Date	No. of Transistors	Width of smallest wire on chip	Clock Speed	Data Width (In Bits)	Millions of Instructions per second(MIPS)
8080	1974	6000	6	2MHz	8	0.64
8088	1979	29000	3	5 MHz	16	0.33
80286	1982	134000	1.5	6MHz	16	1
80386	1985	275000	1.5	16	32	5
80486	1989	1200000	1	25	32	20
Pentium	1993	3100000	0.8	60	32	100
Pentium II	1997	7500000	0.35	233	32	300
Pentium III	1999	9500000	0.25	450	32	510
Pentium 4	2000	42000000	0.18	1.5 GHz	32	1700
Pentium 4P	2004	125000000	0.09	3.6 GHz	32	7000

Applications of microprocessors are classified primarily in two categories:

1. Reprogrammable Systems : Micro computers
2. Embedded Systems : photocopying machine, Digital camera

Microprocessor works or operates in binary digits i.e. 0 and 1, bits. These bits are nothing but electrical voltages in the machine, generally 0 - low voltage level, and 1 - high voltage level. A group of bits form a 'word'. In general, the word length is about 8 bits. This is called as a 'byte'. A word with a length of 4 bits is called as a 'Nibble'

Microprocessor processes the ‘commands in binary form’ to accomplish a task. These are called as ‘instructions’. Instructions are generally entered through input devices and can be stored in a storage device called *memory*.

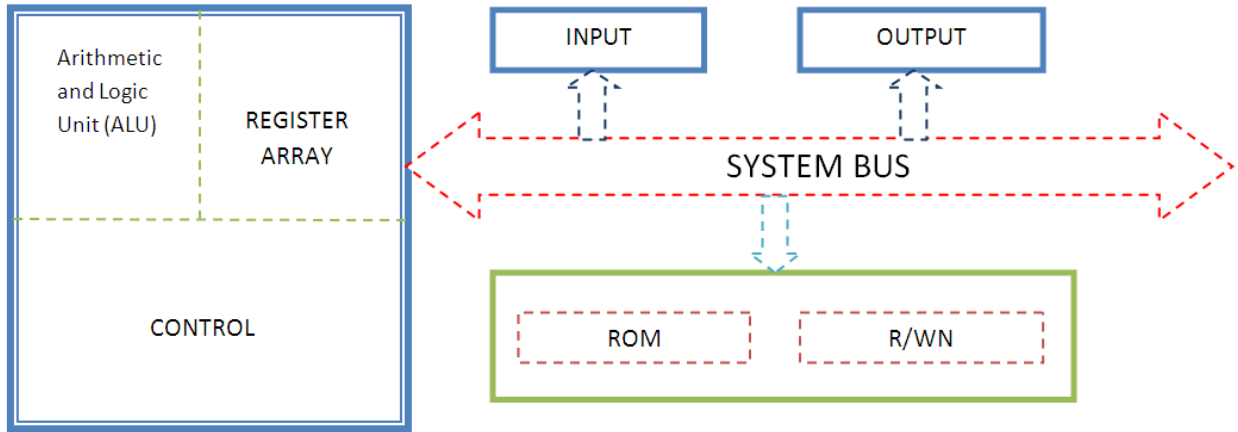


Figure 3.1.2 Schematic of configuration of a micro processor

Figure 3.1.2 and 3.1.3 show the configuration and basic blocks of a microprocessor. The functions of each element are as follows.

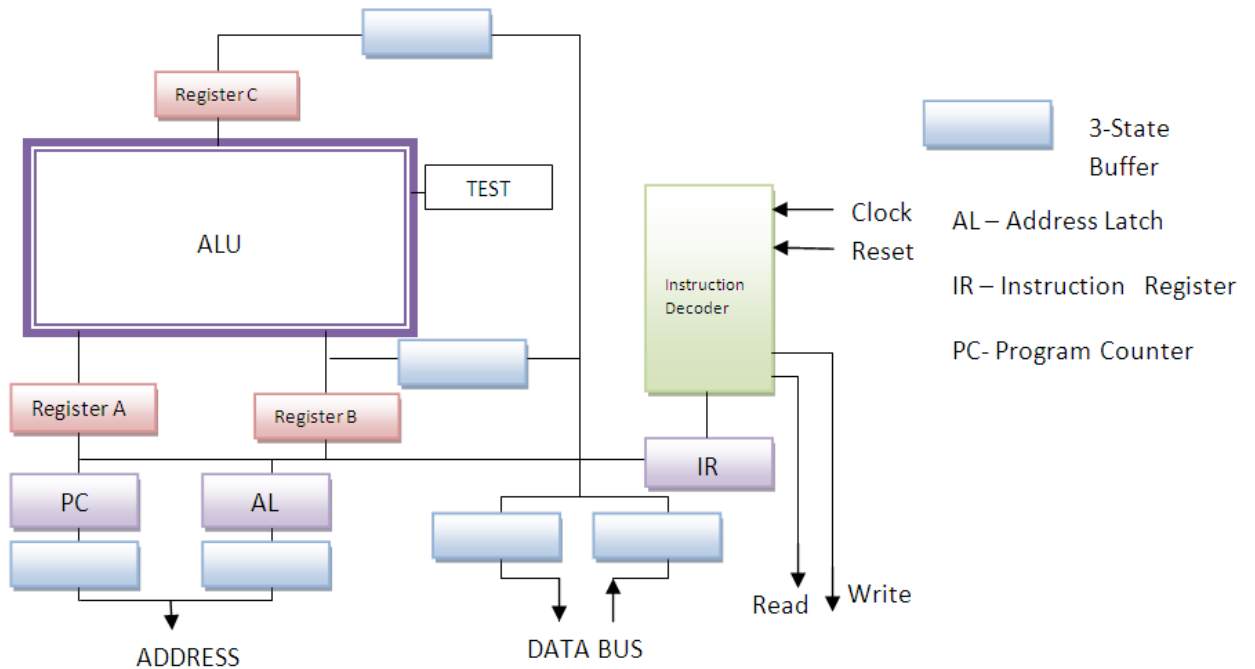


Figure 3.1.3 Working of a microprocessor

1. **ALU:** ALU stands for Arithmetical Logical Unit. As name indicates it has two parts:
 - a. Arithmetical unit which is responsible for mathematical operations like addition, subtraction, multiplication and division,
 - b. Logical unit which is dedicated to take logical decisions like greater than, less than, equal to, not equal to etc. (Basically AND/OR/NOT Operations)
2. **Register Array:** Registers are small storage devices that are available to CPU or processors. They act as temporary storage for processing of intermediate data by mathematical or logical operations.
3. **Control:** This part of CPU is dedicated to coordinate data flow and signal flow through various types of buses i.e. Data Bus, Control Bus, and Address Bus etc. It directs data flow between CPU and storage and I/O devices.
4. **Memory:** There are two different types of memory segments being used by the CPU. First is the ROM which stands for Read Only Memory while other is R/W which stands for Read and Write Memory or Random Access Memory (RAM).
 - a. **ROM:** From this memory unit, CPU can only read the stored data. No writing operations can be done in this part of memory. Thus it is used to store the programs that need no alteration or changes like Monitor Program or Keyboard driver etc.
 - b. **R/W:** As name indicates it is opposite to ROM and used for both reading and writing operations. In general User's program and instruction are stored in this segment of memory unit.
5. **Input Devices:** Input devices are used to enter input data to microprocessor from Keyboard or from ADC which receives data from sensors/signal conditioning systems.
6. **Output Devices:** These devices display the results/conclusions coming out from ALUs either in soft copy (Monitor) or in Hard Copy (Printer).

2.1 Functions of microprocessor

Various functions of microprocessor are as follows:

- Microprocessor performs a variety of logical and mathematical operations using its ALU.
- It controls data flow in a system and hence can transfer data from one location to another based on the instructions given to it.
- A microprocessor can take necessary decisions and jump to a new set of instructions based on those decisions.

2.2 Elements of microprocessor

A simple microprocessor consists of following basic elements (see Figure 3.1.3):

- Data Bus: Through data bus, the data flow between
 - a. various storage units
 - b. ALU and memory units
- Address Bus: It controls the flow of memory addresses between ALU and memory unit.
- RD (read) and WR (write) lines set or obtain the addressed locations in the memory.
- Clock line transfers the clock pulse sequence to the processor.
- Reset Line is used to restart execution and reset the processor to zero.
- Address Latch is a register which stores the addresses in the memory.
- Program Counter: It is a register which can increment its value by 1 and keeps the record of number of instructions executed. It can be set to zero when instructed.
- Test Register: It is a register which stores intermediate or in-process data of ALU operations. For example it is required to hold the 'carry' while ALU is performing 'addition' operation. It also stores the data which can be accessed by Instruction decoder to make any decision.
- 3-State Buffers: These are tri-state buffers. A tri-state buffer can go to a third state in addition to the states of 1 and 0.
- The instruction register and instruction decoder are responsible for controlling the operations of all other components of a microprocessor.

There are following control lines present in a microprocessor, which are used to communicate instructions and data with the instruction decoder.

- Instruct the A register to latch the value currently on the data bus.
- Instruct the B register to latch the value currently on the data bus.
- Instruct the C register to latch the value currently output by the ALU.
- Instruct the program counter register to latch the value currently on the data bus.
- Instruct the address register to latch the value currently on the data bus.
- Instruct the instruction register to latch the value currently on the data bus.
- Instruct the program counter to increment.

- Instruct the program counter to reset to zero.
- Activate any of the six tri-state buffers (six separate lines).
- Instruct the ALU what operation to perform.
- Instruct the test register to latch the ALU's test bits.
- Activate the RD line.
- Activate the WR line

3. *Microcomputer*

Microcomputer is a microprocessor based system. It is a data processing system that employs a microprocessor as its central unit. Based on the input it takes decisions. These decisions are further used to control a system or to actuate an action or operation.

3.1 Microprocessor based programmable controller

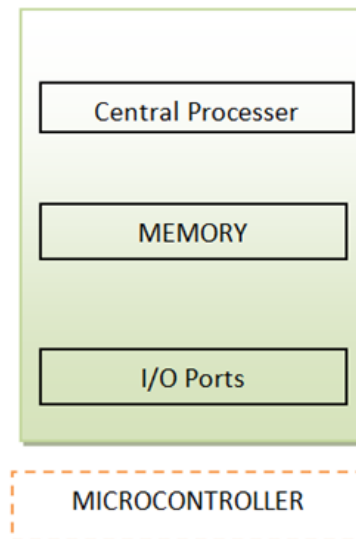


Figure 3.1.4 Schematic of microcontroller.

It is a microprocessor-based system. It implements the functions of a computer and a controller on a single chip. Generally microcontroller is programmed for one specific application and it is dedicated to a specific control function.

Microcontrollers find applications in automobiles, aircraft, medical electronics and home appliances. They are small in size and can be embedded in an electromechanical system without taking up much space. Thus we can have a system with its functions completely designed into a chip. However microcontrollers have very little user programmable memory. Various types of microcontroller chips available in market are: Motorola 68HC11, Zilog Z8 and Intel MCS51 and 96 series.

Module 3: Programmable Logic Devices (PLDs)

Lecture 2

Introduction to microprocessor programming

In this lecture we will study the various number systems, programming languages, and internal architecture of the basic microprocessor, 8085.

1. Number System

Number system is a way of representing the value of any number with respect to a base value. Number System can be classified on the basis of its “base”. Each number has a unique representation in a number system. Different number systems have different representation of the same number. In general Binary, Octal, Decimal and Hexadecimal Number systems are used in microprocessor programming. Table 3.2.1 shows different numbering systems and their details.

Table 3.2.1 Numbering systems

Number System	Base	Allowable Digits/Characters	Examples
Binary	2	0,1	(11001010001010) ₂
Octal	8	0,1,2,3,4,5,6,7	(5671235246214) ₈
Decimal	10	0,1,2,3,4,5,6,7,8,9	(9823654178523) ₁₀
Hexadecimal	16	0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F	(A852F6DB) ₁₆

1.1 Number representation

1.1.1 Conversion of any number system to decimal number system:

Let B be the base of number system and $A_n, A_{n-1}, \dots, A_1, A_0$ be the digits of given number. Then to convert it into decimal equivalent we can use the following formula:

$$N = A_n \cdot B^n + A_{n-1} \cdot B^{n-1} + \dots + A_1 \cdot B + A_0 \cdot B^0 \tag{3.2.1}$$

Example: what is the decimal equivalent of $(11101011)_2$?

➔ Here, we have taken $A_n = 1, A_{n-1} = 1, \dots, A_{n-3} = 0$, while $n=8$ and $B = 2$.

Then the decimal equivalent is $(235)_{10}$.


1.1.2 Decimal number system to any number system:

Any number in decimal system can be changed to any other number system by continuously dividing it by base of the required number system and then writing remainders after each step in reverse order.

Let us take an example of converting a decimal number 235 to its binary equivalent. Following table shows the conversion process as stated above.

Table 3.2.2 Binary representation of $(235)_{10}$

2	235	1
2	117	1
2	58	0
2	29	1
2	14	0
2	7	1
2	3	1
2	1	1



→ Hence Binary equivalent of $(235)_{10}$ is $(11101011)_2$.

1.1.3 Hexadecimal system:

This system is quite extensively used in microprocessor programming. It facilitates much shorter representation of number in comparison with that obtained by using the binary number system. Hexadecimal system has a base of 16 and it is easy to write and remember the numbers and alphabets viz. 0 to 9 and A to F. Table 3.2.3 shows numerals and alphabets used in hexadecimal system for representation of a number.

Table 3.2.3 Numerals and alphabets used in hexadecimal system

Decimal	Binary	Hexadecimal
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

Example: Let us convert the number $(235)_{10}$ to hexadecimal equivalent. Table 3.2.4 shows the conversion of this decimal number.

Table 3.2.4 Decimal to hexadecimal conversion

<u>Number</u>	<u>Division</u>	<u>Remainder</u>	<u>Hexadecimal equivalent of remainder as per table 3.2.3</u>
<u>235</u>	<u>235 divided by 16 = 14 (full)</u>	<u>11</u>	<u>B</u>
<u>14</u>	<u>14 divided by 16 = Cannot divide</u>	<u>14</u>	<u>E</u>

Then by arranging the hexadecimals in reverse order i.e. **(EB)₁₆**. Thus $(235)_{10} = (EB)_{16}$.

1.1.4 Binary coded decimal (BCD)

BCD code expresses each digit of a decimal system by its nibble equivalent. It uses 4 bit binary strings to represent the digits 0 to 9. Figure 3.2.1 shows the representation of number 523 as 010100100011 using BCD system. Due its longer representation scheme, it is now rarely used in micro-electronics programming.

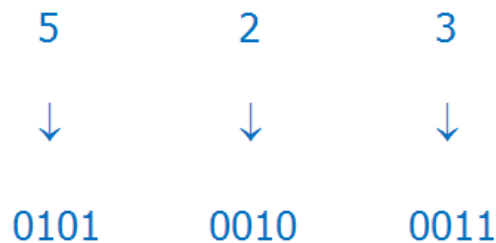


Figure 3.2.1 BCD representation system

Example: $(235)_{10}$ can be represented by using PCD as 001000110101.

2. Low Level Programming Language

Microprocessors recognize the binary numbers and they operate in binary numbers. Each microprocessor has its own binary words, meanings and languages. Each machine has its own set of instructions based on the design of its CPU. Binary language is called as *machine language*. English-like words are used to represent the binary instructions of a machine. This is called as *assembly language* programs. The general purpose languages such as BASIC, FORTRAN, C, etc. are called as *high-level languages*. The *machine language* and *assembly language* are however specific to a microprocessor, therefore these are termed as low-level languages.

2.1 Assembly language

In assembly language, a word length is of about eight bits. It is called as a *byte*. Assembly language can have 256 combinations of bytes. Thus the language has 256 words. There can be a various patterns of bytes. With the help of electronic logic gates, these patterns give a specific meaning to each combination of bytes. These are called as an '*instruction*'. Instructions are made up of one word or several words. The set of instructions designed into the machine makes up the *machine language* that is specific to each microprocessor based system viz. micro-computer.

Thus we can say,

'Machine language is the binary medium of communication through a designed set of instructions specific to each computer.'

'Assembly level language is a medium of communication with a computer in which programs are written in mnemonics. An assembly language is specific to a given computer.'

2.2 Assembly language programming

Assembly language programming is generally written by using hexadecimal codes. Programs can be written by using special keyboard equipped with using hex keys. Programs also have instructions to translate these keys into their equivalent binary patterns. The data and instructions are stored in prescribed locations in memory. An operation code is written which accomplishes the intended task(s). These tasks are carried out on 'operand(s)' by the operation code. 8085 is a typical general purpose microprocessor and has 8-bit word length. Now, let us learn its architecture, working and programming.

3. Internal Architecture of 8085 Microprocessor

3.1 Register Array

8085 Microprocessor consists of six registers, one accumulator and a flag register. The typical architecture is shown in figure 3.2.2. There are six general-purpose registers B, C, D, E, H, and L, each having capacity to store 8 bit data. They are combined as BC, DE, HL to perform 16 bit operations. In addition to this Register array, two 16 bit registers viz. stack register and program counter are provided. As discussed in the earlier lecture, the ‘program counter’ is employed to sequence the execution of instructions. It always points to the memory address from which the next byte is to be fetched. Stack Pointer points to the memory location in R/W (Read and/or write) memory. It is also termed as a ‘stack’.

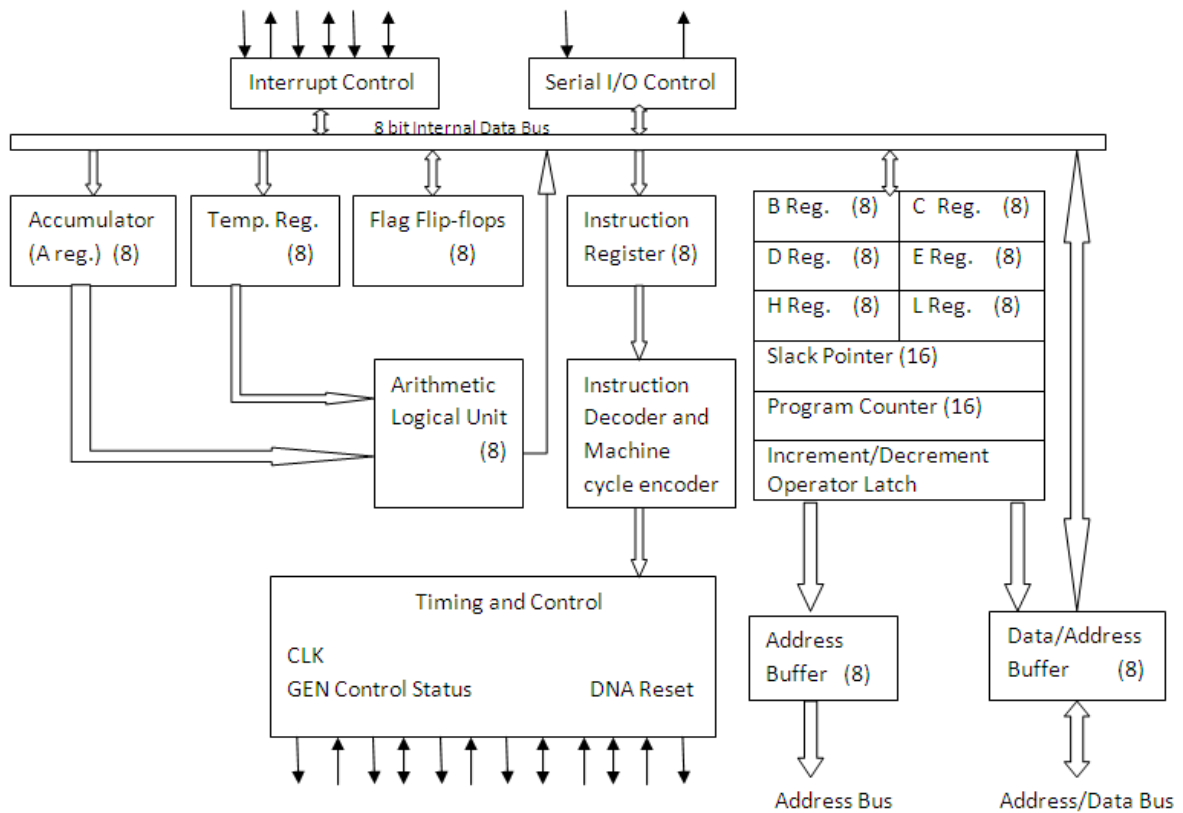


Figure 3.2.2 Architecture of 8085 microprocessor

3.2 Accumulator

The accumulator is 8-bit register (can store 8 bit data). It is a part of arithmetic/logic unit (ALU). In general, after performing logical or arithmetical operations, result is stored in accumulator. Accumulator is also identified as Register A.

3.3 Flags

ALU of 8085 have five flip flops whose states (set/reset) are determined by the result data of other registers and accumulator. They are called as Zero, Carry, Sign, Parity and Auxiliary-Carry flags.

- Zero Flag (Z): When an arithmetic operation results in *zero*, the flip-flop called the Zero flag - which is set to one.
- Carry flag (CY): After an addition of two numbers, if the sum in the accumulator is larger than eight bits, then the flip-flop uses to indicate a *carry* called the Carry flag – which is set to one.
- S-Sign (S): It is set to 1, if bit D7 of the result = 1; otherwise reset. D7 is the first digit of a binary number.

D7	D6	D5	D4	D3	D2	D1	D0
S	Z		AC		P		CY

- P-Parity (P): If the result has an even number of 1s, the flag is set to 1; for an odd number of 1s the flag is reset.
- AC-Auxiliary Carry (AC): In an arithmetic operation, when a carry is generated by digit D3 and passed to digit D4, the AC flag is set. Generally this flag is used internally for Binary Coded Decimals (BCD).

Figure 3.2.2 shows a 8-bit flag register, adjacent to the accumulator. It is not used as a register. Out of eight bit-positions, five positions are used to store the outputs of five flip-flops. These flags play an important role in decision-making process of the microprocessor.

3.4 Instruction Register/Decoder

Before execution of an instruction, it is sent to the Instruction Register. Instruction register stores current instruction of any program. Decoder takes the instruction from memory, decodes it and then passes it to the next stage.

3.5 Memory Address Register

Memory Address Register (MAR) holds the address of next instruction to be executed.

3.6 Control Generator

In microprocessor, the Control Generator generates a signal that executes the operations in accordance to the decoded instructions. In fact it creates a signal (information) which have details about connections between different blocks of the microprocessor so that data reaches to the respective place.

3.7 Register Selector

Register selector is basically a logical controller which directs switching between different registers of microprocessor.

3.8 General Purpose Registers

Microprocessor has few extra registers which can be used to store additional data during a program.

4. *Programming in 8085*

As mentioned in above section, a simple and very effective substitution to binary codes could be use of standard English words to complete any task. For example addition of two numbers can be represented by ADD. Such codes are referred as *mnemonic codes* and that language is called *assembly language*. Most of the early processers including 8085, are programmed using *mnemonics*. However, assembly language codes should be converted into binary one so that microprocessor can identify the instructions given to it. This operation is done by Assembler. In assembly language, instructions are composed of two segments which are as follows:

1. Operation (Op) Code: It depends on which operation is to be performed. For example for OR operation, we have Op Code “OR”.
2. Operands: Operand is the object on which the required operation is to be done. Generally operations are done on data stored in registers.

4.1 Classification of Instructions:

- Data Transfer
- Arithmetic
- Logical
- Program Control

4.1.1 Data Transfer

1. Load: It reads content from specified memory location and copies it to specified register location in CPU.
2. Store: It copies content of a specified register into specified memory location.

4.1.2 Arithmetic:

1. Add: It adds contents of a specified memory location to the data in some register.
2. Decrement: It subtracts 1 from contents of specified location.
3. Compare: It tells whether contents of a register are greater than, less than or same as content of specified memory location.

4.1.3 Logical:

1. AND: Instruction carries out Logical AND operation with the contents of specified memory location and data in some register. Numbers are *ANDed* bit by bit.
2. OR: Instruction carries out Logical OR operation with the contents of specified memory location and data in some register. Numbers are *ORed* bit by bit.
3. Logical Shift: Logical shift instruction involves moving a pattern of bits in the register one place to left or right by moving a zero in the end of number.

4.1.4 Program Control:

1. Jump: This instruction changes the sequence in which program steps are carried out. Normally program counter causes the program to be carried out sequentially in strict numerical sequence. However, JUMP causes program counter to some other specified location in the program.
2. Branch: This is a conditional instruction which might 'branch' if 'zero' results or 'branch' if 'plus' results of an operation. Branch also followed if right conditions occur in the decision making process.
3. Halt: This instruction stops all further microprocessor activity.

4.2 Example: Though the basic concept remains same, however the Op codes may be different for various microprocessors. A program example for 8085 is as follows.

- | | |
|--|-------|
| 1. Add a 8-bit number 16F to Accumulator
16F | ADI |
| 2. Add contents of Register D to Accumulator
D | ADD |
| 3. Subtract a 8-bit number 32H from Accumulator
32H | SUI |
| 4. Subtract contents of Register B from Accumulator | SUB B |
| 5. Increment the contents of Register D by 1 | INR D |
| 6. Decrement the contents of Register C by 1 | DCR C |
| 7. Load a 8-bit number 15H in register D
D, 15H | MVI |
| 8. Copy from Register B to Register C
C,B | MOV |

4.3 Sample Program:

Write the instructions to load two hexadecimal numbers 36B and 419 in the registers C and D respectively. Add the numbers, and store the result in memory location B244D.

Mnemonics Code:

```
MVI C, 36B      // Load Register C with 36B
MVI D, 419      // Load Register D with 419
ADD D           // Add two bytes and save sum in C
STA B244D       // Store the sum in memory location B244D
HLT             // Terminate
```

Module 3: Programmable Logic Devices (PLDs)

Lecture 3

Programmable Logic Controllers

1. Introduction

Any computer having input and output interfaces can be used to control external devices. However most of the computers are not industrially hardened. Input / Output devices of general-purpose microcomputers are not engineered to handle line-voltages and currents above transistor-transistor logic (TTL) levels. Also they are not designed to with-stand the temperature, humidity, and vibration on shop floors. These drawbacks of a general purpose computer have been rectified by developing a Programmable Logic Controller (PLC) with built-in isolation into their inputs and outputs.

“The programmable logic controller is defined as a digital electronic device that uses a programmable memory to store instructions and to implement functions such as logic, sequencing, timing, counting and arithmetic words to control machines and processes.”

PLCs are generally used for incorporating automation in open loop systems where processes are to be performed in a sequential manner. PLCs are used for automation of assembly lines in industries. They are generally designed for multiple input multiple output (MIMO) systems. In PLCs, instructions are saved in nonvolatile memory. Some of the advantages of PLCs are:

- Cost effective
- Flexibility and ability to use similar system for other processes
- Programming interface is easier in comparison to other processers
- Resistant to impact and vibration
- Resistant towards electrical and mechanical noise
- Ability to work at high temperatures

Now let us study the structure and functioning of a PLC.

2. Programmable Logic Controller: Structure and Functioning

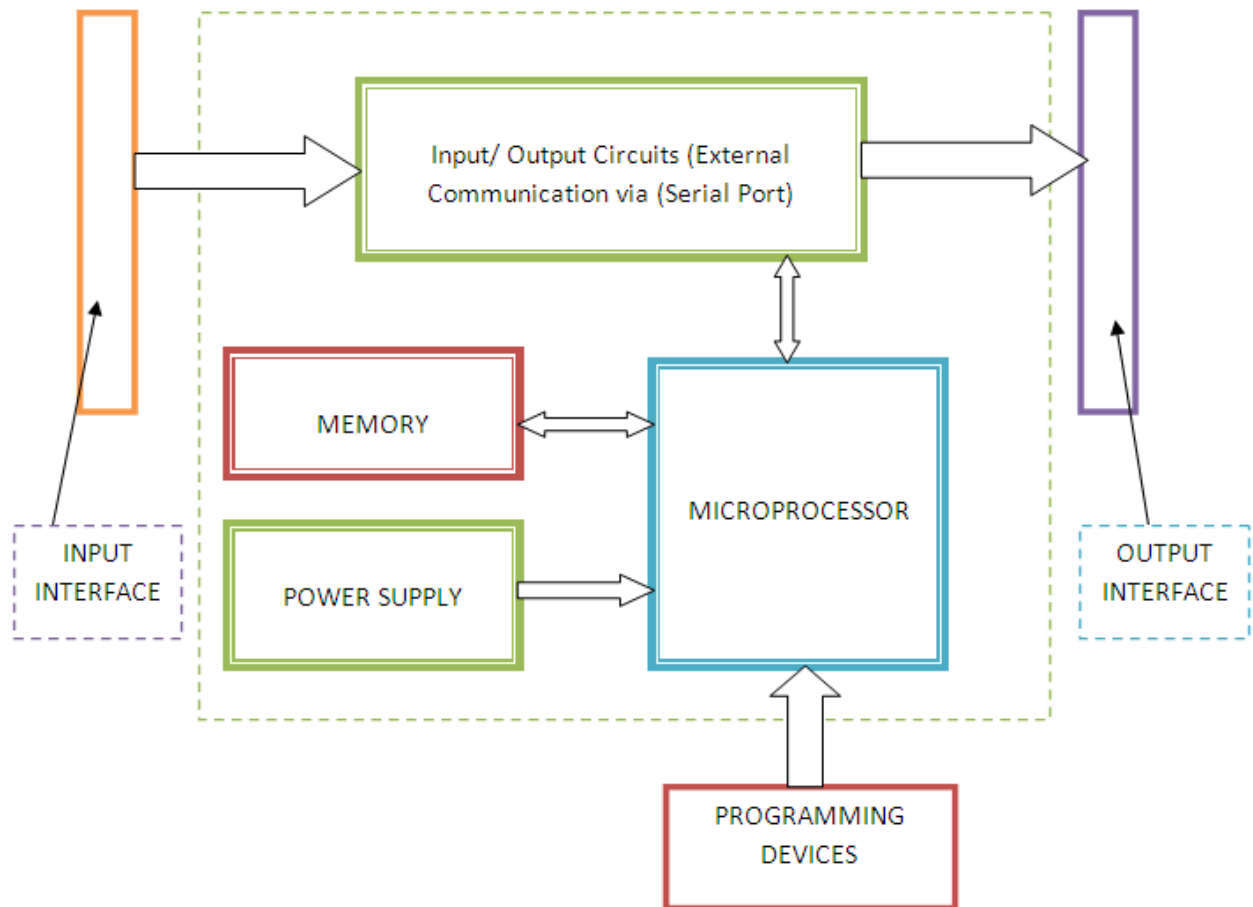


Figure 3.3.1 Block diagram of a PLC

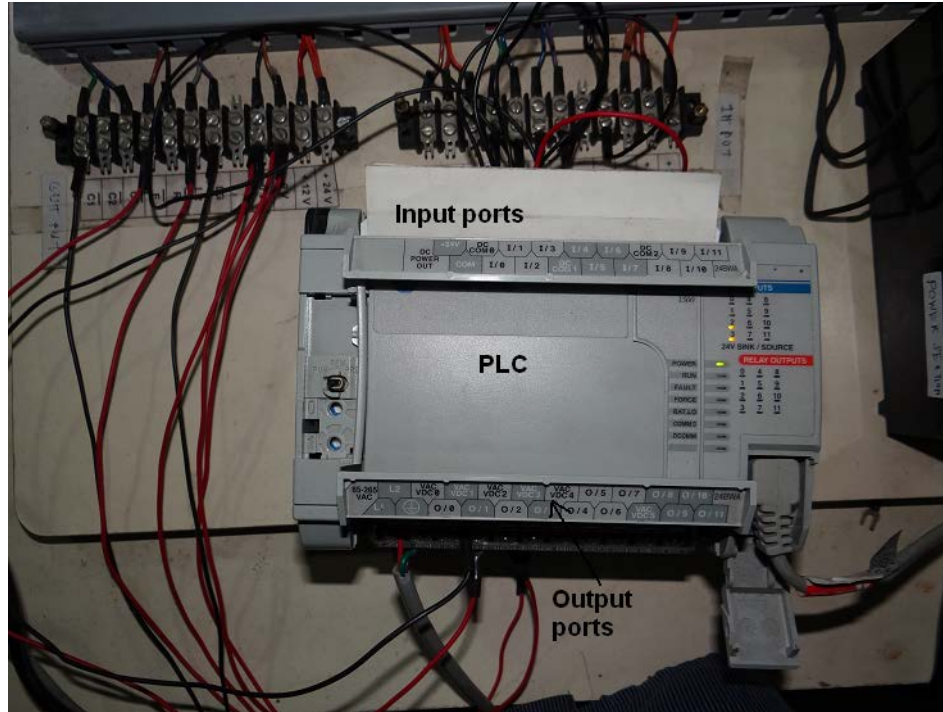


Figure 3.3.2 An Industrial PLC

Figure 3.3.1 shows the basic elements of a PLC. It is basically a microprocessor based control system. Microprocessor communicates with the outside world with input/output devices via a circuitry. This circuitry protects the microprocessor and other elements of PLC from the high voltages and currents coming to the PLC. Microprocessor does its basic functions of taking decisions according to the instructions written in the programs which are stored in the memory. PLC scans a set of sensor inputs rapidly and repeatedly. Then it evaluates their logic relationships to defined outputs according to a logic program. At last it sets the outputs according to the programmed logic. Figure 3.3.2 shows an industrial PLC with input and output ports.

3. *Programming a PLC:*

PLCs are programmed through concept of ladder logic. In general there exists a graphical user interface (GUI) to program a PLC that makes it different from other processors. Ladder logic comprises of two columns. Left column shows input devices like switches, sensors while in output column is at right side which shows actuators like cylinders, motors.

Meanings of symbols used in PLC Program:

[[

This instruction is called as “examine on” or “normally opened” as input functions or storage bits. If the corresponding memory bit is a “1” then the respective ‘rung’ will continuously be executed and the corresponding outputs will be energized. Rung is one of the multiple horizontal programming lines in a ladder logic diagram.

NOTE: Other factors may also affect rung simultaneously.

If the corresponding bit is “0” then the rung will not be executed continuously and outputs will be de-energized. If this instruction is used as input bit, its status should be according to the status of the real world input devices connected to the input table by identical addresses.

Addressing Sample: I: 3/1

This indicates address of a sample. I indicates input image table, 3 indicates slot no. 3 of input port and 1 indicates bit three of 3rd slot of input port.

[/

This instruction is called “examine off” or “normally closed” as input functions or storage bits. If the corresponding memory bit is a “1” then the respective ‘rung’ will continuously be executed and the corresponding outputs will be energized.

NOTE: Other factors may also affect this rung simultaneously.

If the corresponding bit is “0” this instruction will not allow rung continuously and outputs will be energized. If used as input bit, its status should correspond to the status of the real world input devices tied to the input table by identical addresses.

OTE → () →

This is called as ‘output energize’. This instruction sets the specified bit when rung continuity is achieved. Under normal operating conditions, if the set bit corresponds to an output device, output device will be energized when rung goes true.

Addressing Example O:3/1

- O -- output image table
- 3 -- slot three
- 1 -- bit one of slot three

OTL → -(L)-

This is called as ‘output latch’. This instruction functions similar to output energize except that once a bit is set with OTL, it is latched on. Once an OTL bit has been set ON (1 on the memory) it will remain ON even if the rung condition goes false. The bit must be reset.

(U)

This is called as ‘output unlatch’. This is used to unlatch (reset) a latched bit. Its address must be same as latched one.

Timer

This is also called as “TON”. Figure 3.3.3 shows the schematic of a Timer. It is used to turn an output ON or OFF after the timer has been ON for preset time interval. This output instruction begins timing when rung goes true. It waits the specified amount of time (As specified in Preset), keeps track of accumulated intervals which have occurred (ACCUM), and sets DN (Done) bit when ACCUM time equals preset time.

As long as rung condition remains true, Timer adjusts its accumulated value to each evaluation until it reaches the preset value. The accumulated value is reset when rung condition go false, regardless of whether timer has timed out. ” TIME BASE” is an amount of time after which accumulator increases its value by 1.

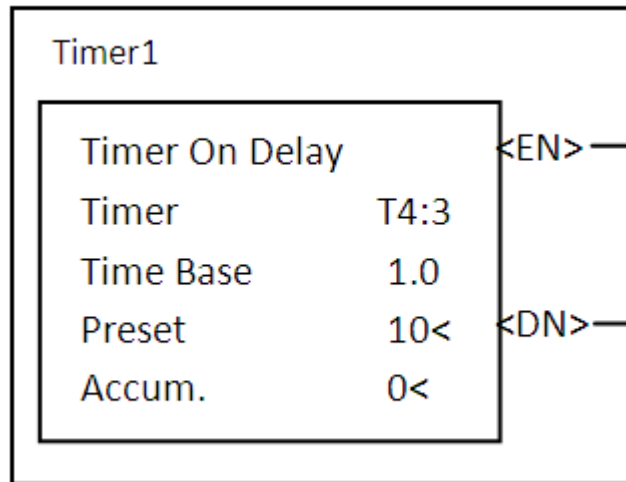


Figure 3.3.3 Schematic of a Timer

4. Case study

In this segment, we will see how PLCs are incorporated to control various activities in an industry. In this illustration we have a conveyor belt run with two motors at its ends, three different stations to perform various activities like painting of vehicle body or fitting of any component in chassis etc along with two switches to run conveyer. Figure 3.3.4 shows the photograph of a conveyor belt system. The PLC is of “Bull 1764 Micrologix 1500 LSP Series C” which can be controlled by a Graphical User Interface “RS Logic 500 Starter”.

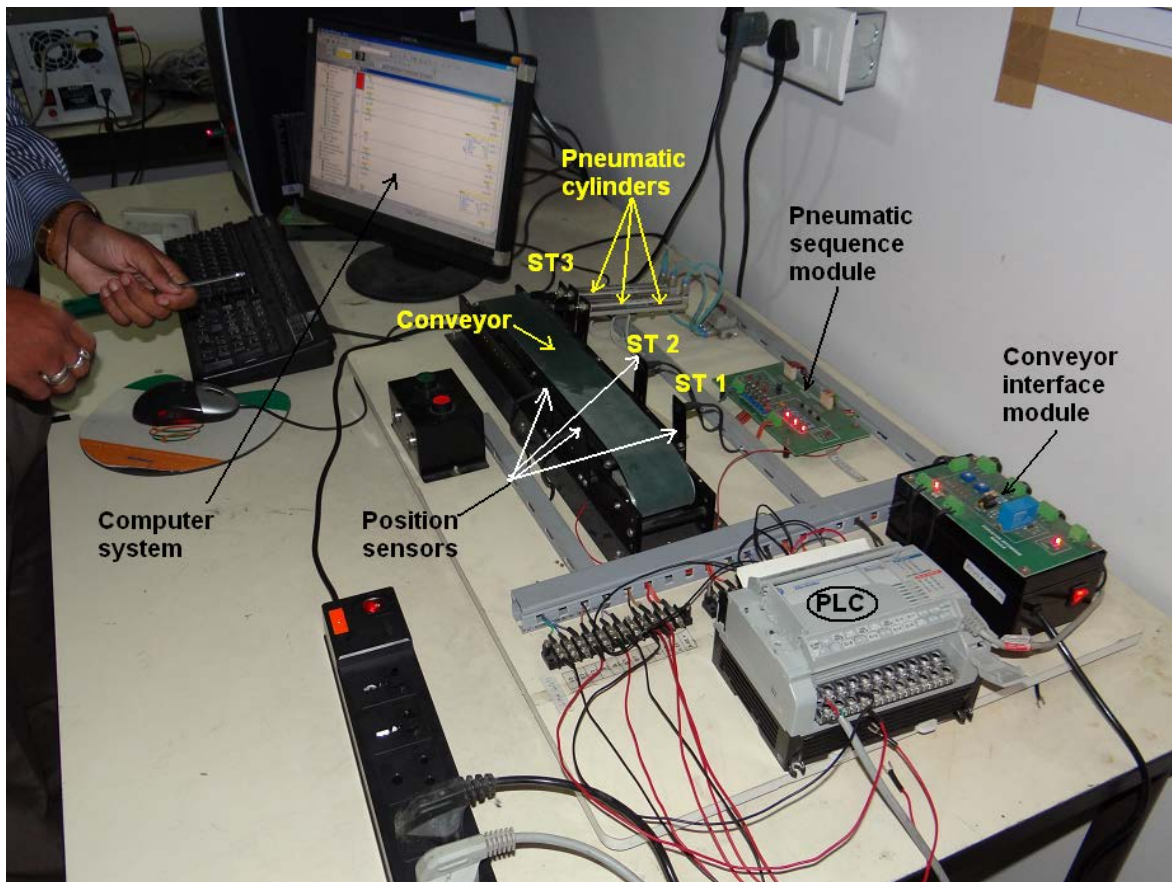


Figure 3.3.4 PLC controlled conveyor belt system

To run the conveyer belt with the help of switches

As discussed in earlier sections, PLCs are controlled through Ladder Logic. In input section of the ladder, name of the input device must be mentioned on the top of symbol, followed by primary input port. Secondary input port is mentioned just below symbol. In similar way, output symbol should be mentioned with name and output ports as shown in figure 3.3.5.

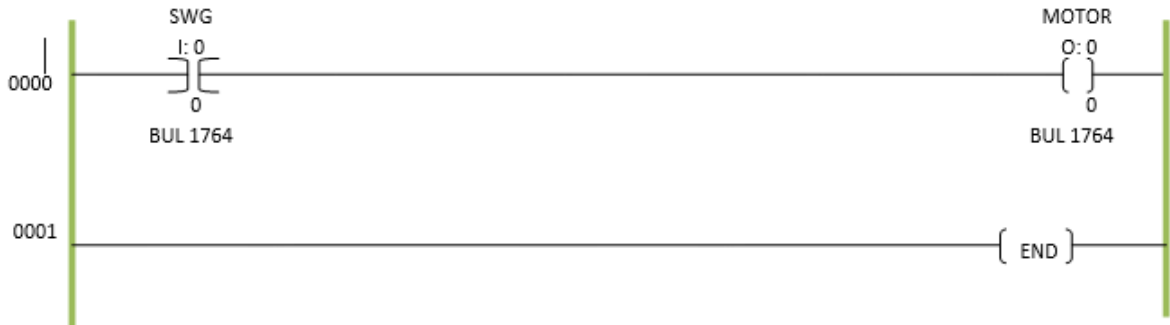


Figure 3.3.5 A rung to run the conveyer belt.

To control movement of pneumatic devices in an industry with PLCs

Figure 3.3.6 shows a program code to control the motion of pneumatic cylinder with a switch.

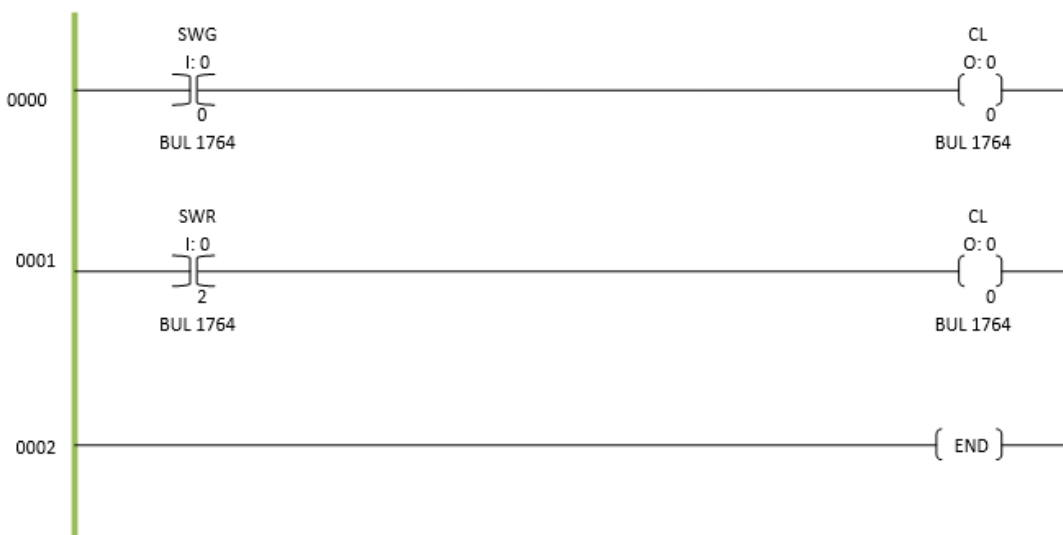


Figure 3.3.6 Program code to actuate a pneumatic cylinder

To control the operation of sensors

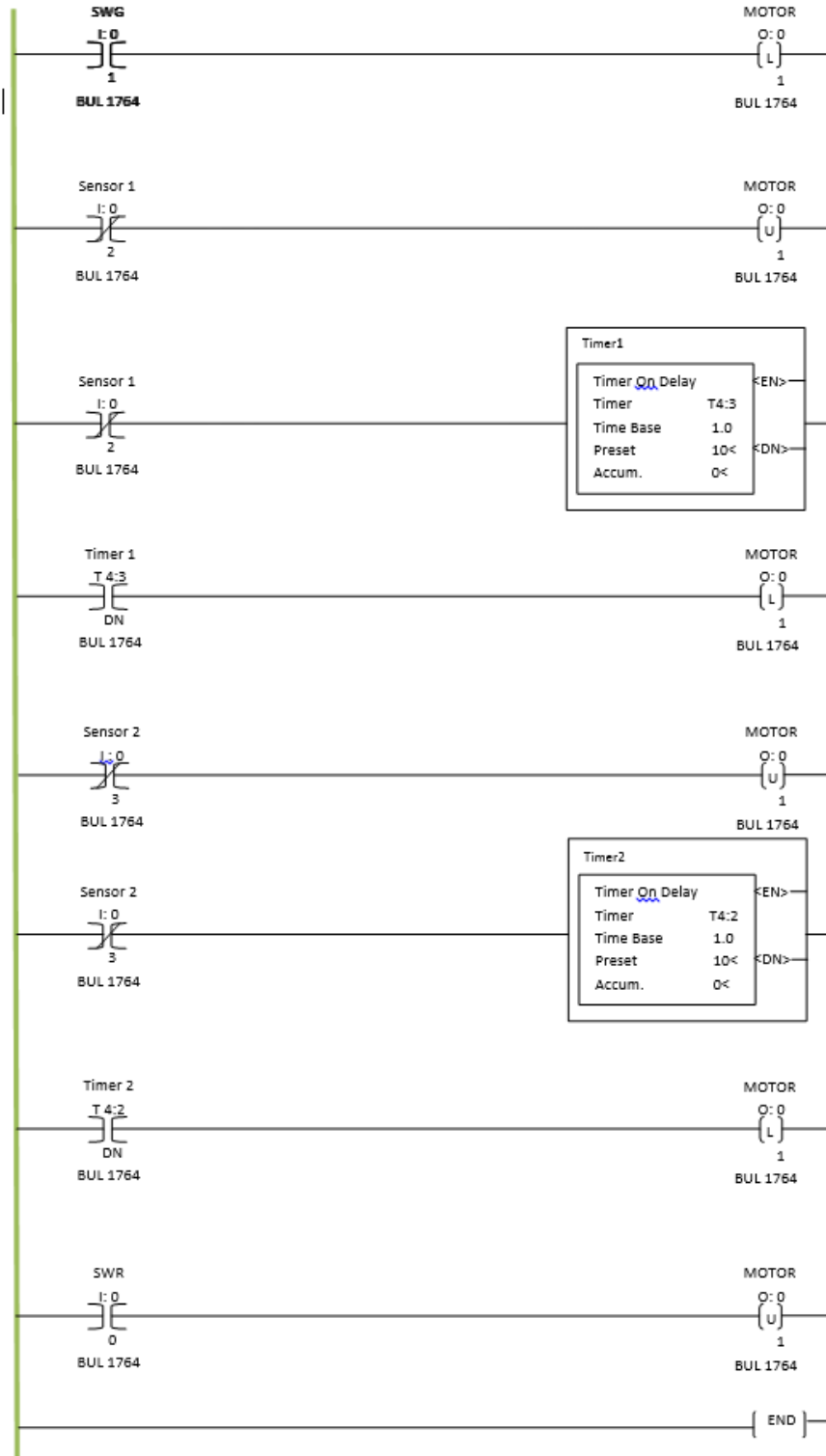


Figure 3.3.7 Program code to make use of sensors and actuators

In industrial applications, it is required to use various sensors to control the operations of systems and processes using PLCs. Figure 3.3.7 shows a typical program to operate an electric motor and a pneumatic cylinder with the help of some sensors such as pneumatic proximity switch.

To control a mechatronics system we need to combine various mechanical and electrical input and output devices and to operate them in a sequential manner. Consider a prototype of industrial assembly line with 3 stations as shown in Figure 3.3.4.

At first station ST1, the sensor identifies an object (finished product) on the conveyor belt and sends a signal to the controller. The controller processes this information and actuates the electric motor to run the conveyor belt.

Second Station ST2: It is allotted for the inspection of the finished product or object. At ST2, the conveyor belt stops. In case any fault is diagnosed by the inspection system, the product will be taken away by the pneumatic actuators placed at Station 3, ST3.

Module 3: Programmable Logic Devices (PLDs)

Lecture 4

PID controllers

1. Introduction

In mechatronics, generally the objectives are to automate a process or to control parameters of system. Control systems for manufacturing systems can be categorized into two types. First is the sequential control where all the operations are carried out in a sequence to automate the mechanical system(s). Automated vehicle assembly line is an example of such control system. Such operations are controlled by Programmable Logic Controller (PLCs) which we have already studied in previous lecture.

In the other type of the control system, precise control over output of system is to be obtained. Therefore a continuous monitoring of such system is essential. For example there is a necessity to continuously monitor and control the fuel tank used in a Boiler based power plant. This type of control is also called modulating control. Feedback systems and Proportional-integral-derivative (PID) controllers are employed in these systems.

In general a closed loop system has several input variables and several output variables. However one or two dominant input variables are considered in designing the control system. Output variables are measured by using suitable transducer system and the feedback is sent to the controller for comparison. A block diagram of closed loop system is shown figure 3.4.1.

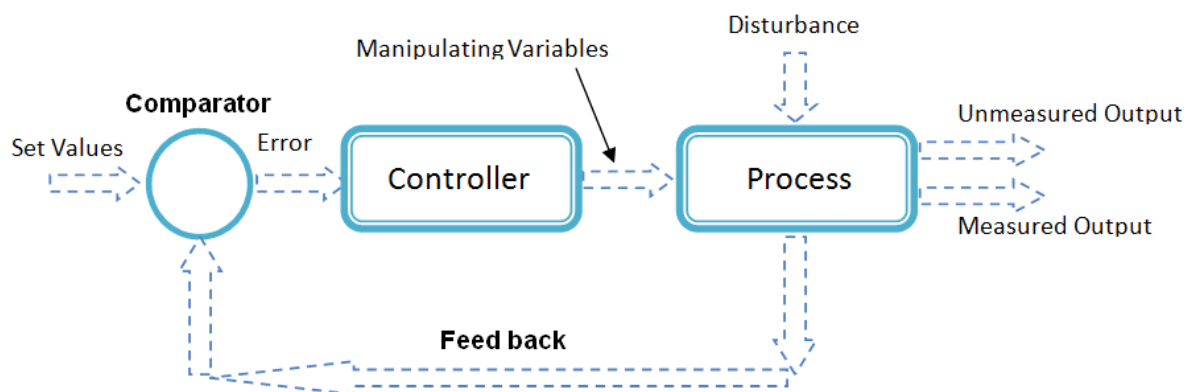


Figure 3.4.1 A closed loop control system

There are various types of closed loop control systems being used in mechatronics. These are listed as follows.

1. Single Input Single Output (SISO)
2. Multiple Input Single Output (MISO)
3. Multiple Input Multiple Output (MIMO)
4. Single Input Multiple output (SIMO)

2. PID Controller for SISO systems

PID controller is commonly used for SISO systems. Figure 3.4.2 shows the basic blocks of a SISO system. It has single input and single output. It has a controller which controls the operation of a process based on the feedback received.

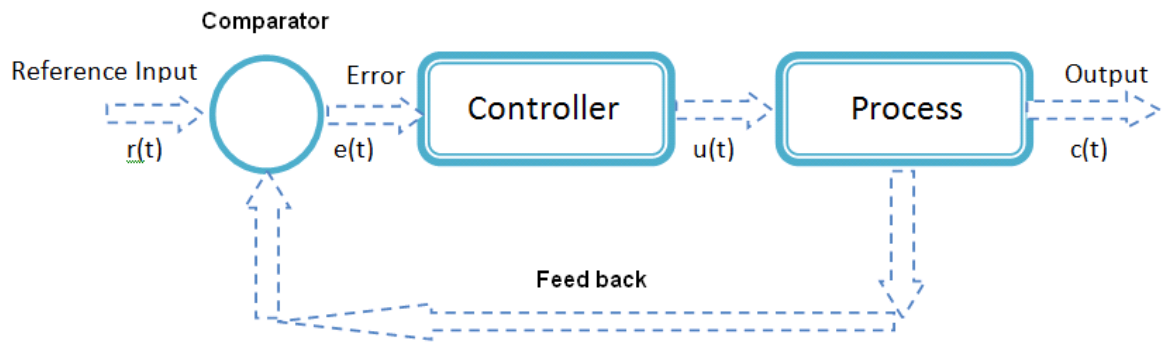


Figure 3.4.2 A SISO system.

For a PID controller, the output can be expressed in terms of input as follows:

$$u(t) = K_p \left[e(t) + \tau_d \frac{d e(t)}{dt} + \frac{1}{\tau_i} \int_0^t e(\tau) d\tau \right] \quad (3.4.1)$$

And the transfer function of PID controller can be written as,

$$C(s) = K_p \left[1 + \tau_d s + \frac{1}{\tau_i s} \right] \quad (3.4.2)$$

Where $K_p \rightarrow$ Proportional Gain

$\tau_d \rightarrow$ Derivative Time

$\tau_i \rightarrow$ Integral Time

PID controller consists of Proportional, Integrator and Differentiator Controllers which can be understood by considering a first order system SISO whose transfer function can be written as,

$$P(s) = \frac{K}{1+\tau s} \quad (3.4.3)$$

Now let us study the Proportional, Integrator and Differentiator Controllers one by one and then adding them together as PID controller.

2.1 Proportional Controller (P – Controller)

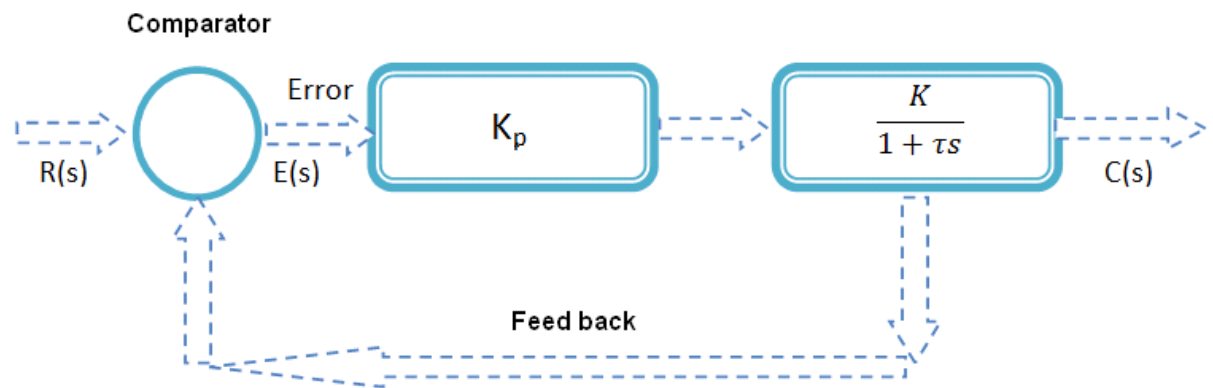


Figure 3.4.3 Proportional controller

The proportional controller gives an output value that is proportional to the error value with a gain value of K_p . The proportional response can be adjusted by multiplying the error by a constant K_p , called the proportional gain constant (Eq. 3.4.1). Figure 3.4.3 shows the schematic of a proportional controller for a closed loop control system, the transfer function can be written as,

$$\frac{C(s)}{R(s)} = \frac{\frac{KK_p}{1+\tau s}}{1+\frac{K}{1+\tau s}} \quad (3.4.4)$$

$$\frac{C(s)}{R(s)} = \frac{KK_p}{1+KK_p} \cdot \frac{1}{1+\tau'.s} \quad (3.4.5)$$

Where $\tau' = \frac{\tau}{1+KK_p}$

Thus unit step response for Proportional Controller will be,

$$c(t) = \frac{KK_P}{1+KK_P} (1 - e^{-\frac{st}{\tau'}}) \quad (3.4.6)$$

Effect of adding Proportional Controller in the system:

On adding a proportional controller in system, Time response of system improves by a factor of $\frac{1}{1+KK_P}$. Also on adding proportional controller, steady state offset arises between desired response and output response as shown in figure 3.4.4.

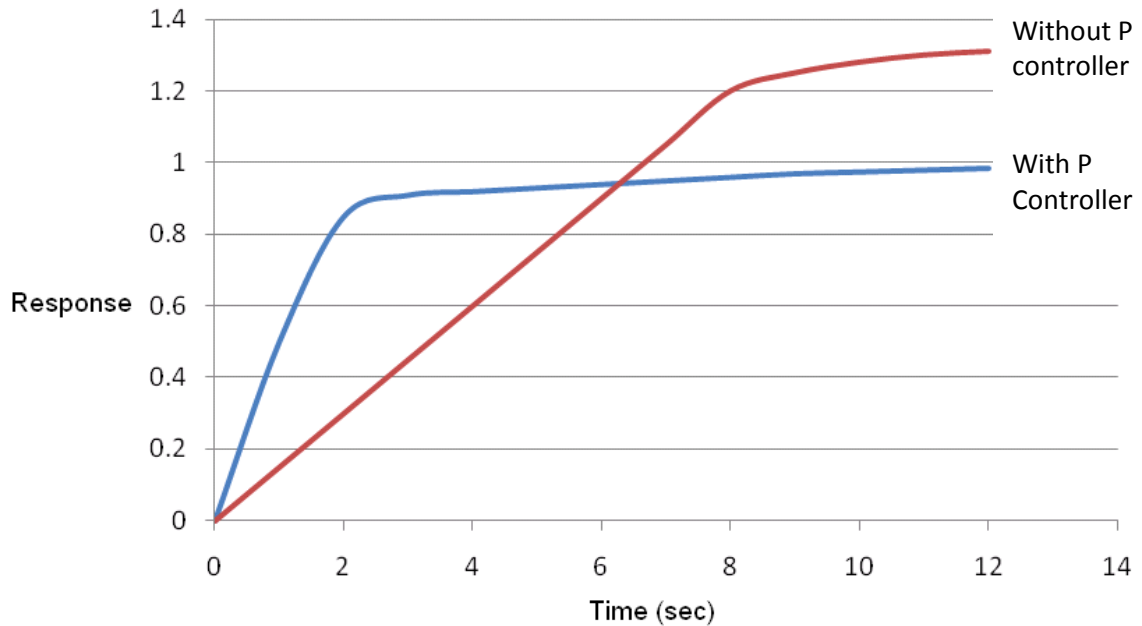


Figure 3.4.4 System response with and without P-controller

2.2 Integral Controller

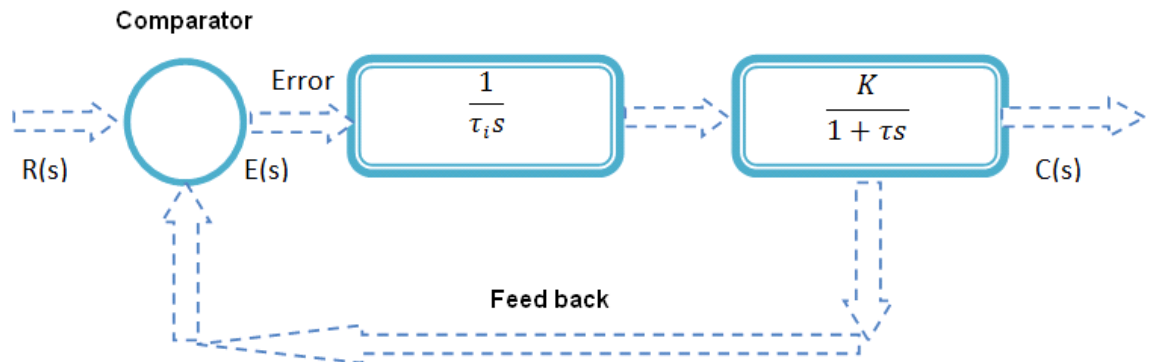


Figure 3.4.5 A I-Controller

In an integral controller, the manipulation equals the integral of the error over time, multiplied by a gain K_I (Eq. 3.4.1). Figure 3.4.5 shows the block diagram of Integral controller employed for a SISO system. The closed loop transfer function can be written as,

$$\frac{C(s)}{R(s)} = \frac{\frac{K}{\tau_i s(1+\tau s)}}{1 + \frac{K}{\tau_i s(1+\tau s)}} \quad (3.4.7)$$

$$\frac{C(s)}{R(s)} = \frac{K}{K + \tau_i s + \tau \tau_i s^2} \quad (3.4.8)$$

For a Step input, $R(s) = 1/s$, we get the steady state error as,

$$e(s) = \frac{\tau_i s(1+\tau s)}{\tau_i s(1+\tau s) + K} \cdot \frac{1}{s} \quad (3.4.9)$$

$$e_{ss} = \lim_{s \rightarrow 0} s \cdot e(s) = 0 \quad (3.4.10)$$

Effect of adding Integral Controller in system:

The step response of this closed loop system with integral action is shown in figure 3.4.6. The integral term enhances the movement of the process towards desired point. It also eliminates the residual steady-state error that produces with a pure proportional controller. From the transfer function, it is observed that use of integral controller leads to increasing order of closed loop system which may cause instability, slow and oscillatory response. However the system has a major advantage that integral controller produces zero steady state error.

The drawbacks of integral controller can be rectified if we use Proportional controller along with Integral one.

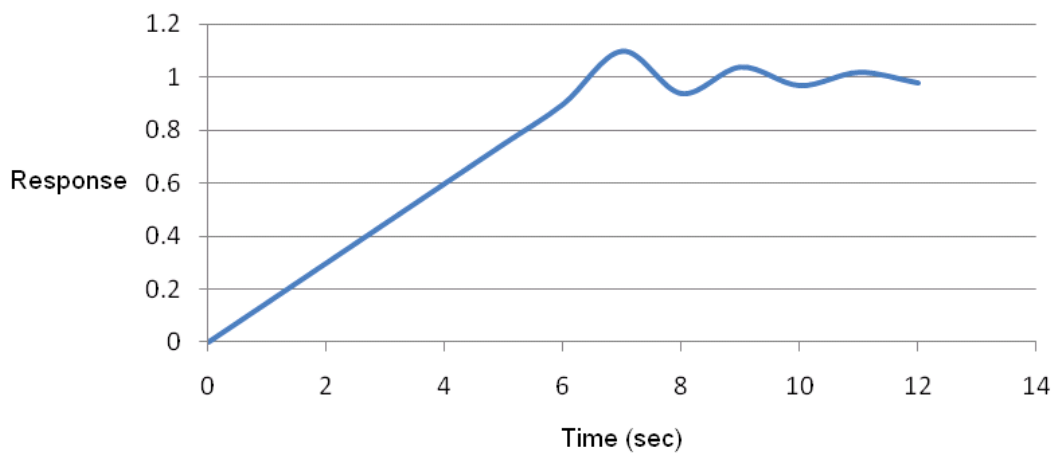


Figure 3.4.6 Response using a Integral controller

2.3 Differential control

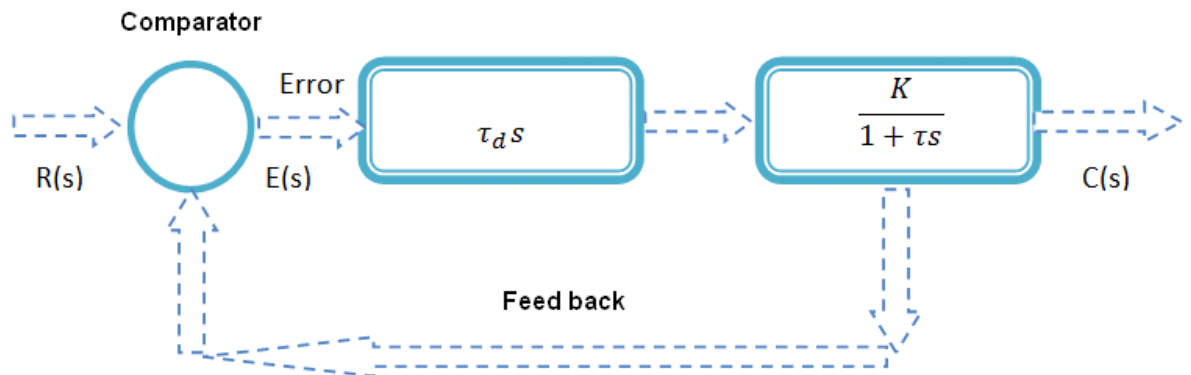


Figure 3.4.7 Block diagram of Differential controller

A derivative controller uses the derivative of the error instead of the integral. Figure 3.4.7 shows the building blocks of a differential controller. In this closed loop system, the transfer function can be written as,

$$\frac{C(s)}{R(s)} = \frac{\frac{\tau_d s K}{(1+\tau s)}}{1 + \frac{\tau_d s K}{(1+\tau s)}} \quad (3.4.11)$$

$$\frac{C(s)}{R(s)} = \frac{\tau_d s K}{1 + \tau s + \tau_d s K} \quad (3.4.12)$$

For Step input $R(s) = 1/s$, the steady state error would be,

$$e(s) = \frac{\tau_d s K}{1 + \tau s + \tau_d s K} \frac{1}{s} \quad (3.4.13)$$

Effect of adding Differential Controller in system:

Derivative controller improves stability of the system and it also improves settling time. Derivative of the error can be calculated by determining slope of the error over time and multiplying this term with derivative gain τ_d . Figure 3.4.8 shows the response of a system to unit step Input. It shows an initial jump in the response. This is due to the effect of the derivative controller. Here derivative gain is very high which results in high settling time.

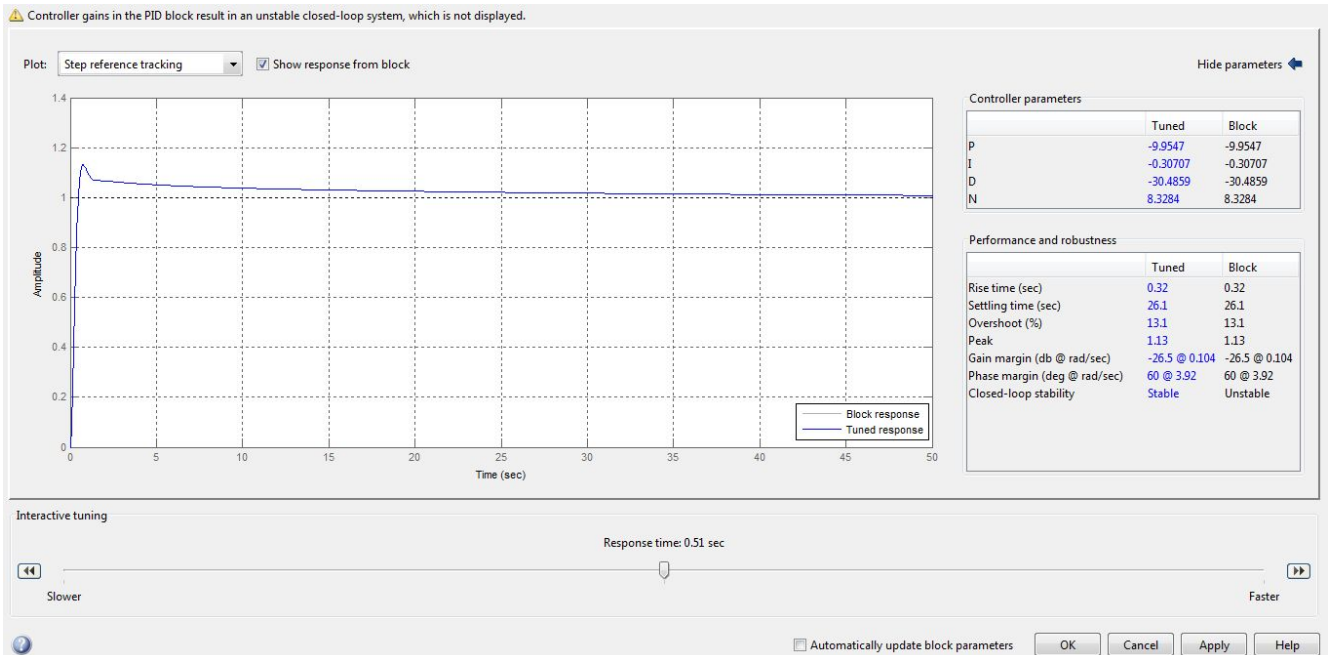


Figure 3.4.8 Response using a Differential controller

After discussing various components of controller following conclusions can be drawn.

1. Proportional Controller improves system response time. It provides high proportional gain which results into very low rise time and thus improves the response system.
2. Integral Controller makes the system steady with error approaches zero. But Integral controller may increase instability of a system and may cause oscillations. However in Proportional system provides very low value of Integral gain resulting in very low amount of oscillations.
3. Derivative controller improves system settling time and also improves stability.

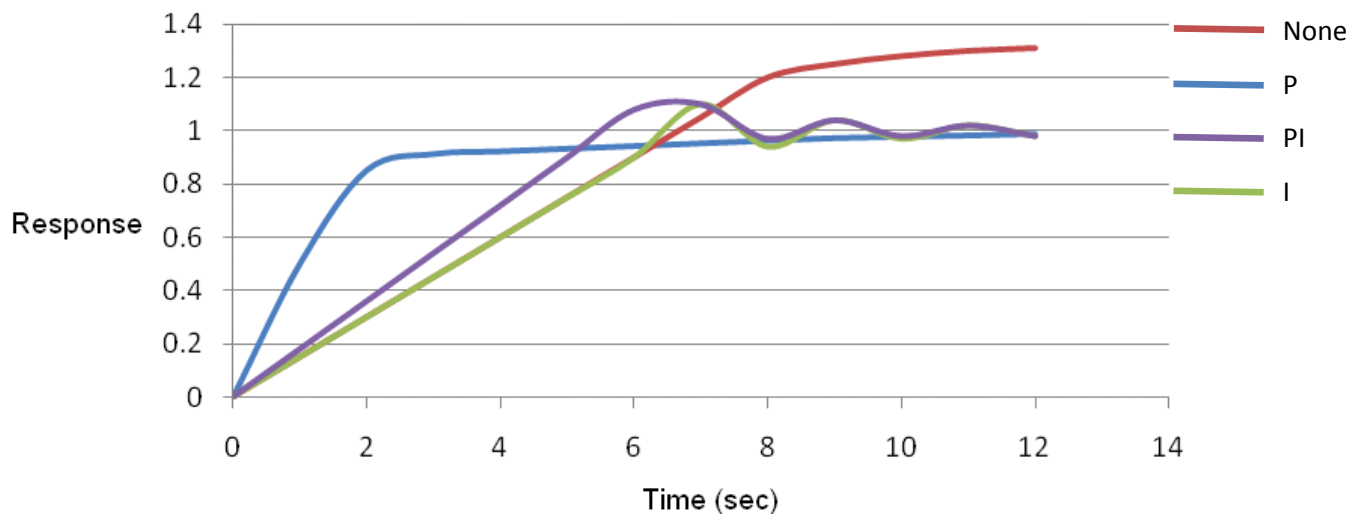


Figure 3.4.9 Comparison of different control systems.

A comparison of systems with no controller, only proportional controller, only Integral controller and both proportional and integral controller can be seen in Figure 3.4.9. It can be seen that the response curve produced by PI controller is better in comparison with that obtained by only P, only I and without any controller. PI controller has the advantages of both the P as well as I controllers. Therefore in general, it is recommended not to employ integral and derivative controllers on their own. They are always to be used in conjunction with a proportional controller.

3. Types of PID controller:

3.1 Parallel PID Controller

Figure 3.4.10 shows the parallel configuration of PID controller. In general, this type of PID is preferred over series one because in Parallel PID, output of one controller does not affect the output of other. This allows freedom to control parameters independently.

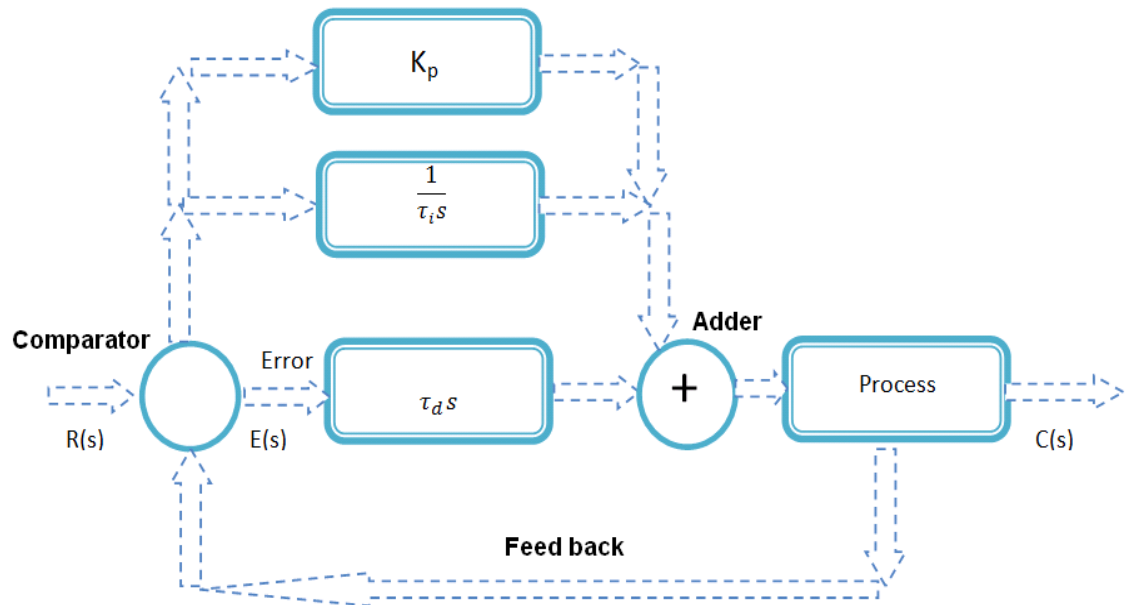


Figure 3.4.10 Configuration of PID controller

3.2 Series PID Controller

Figure 3.4.11 shows the typical configuration of series PID controller. In this type of controller, the output of a controller affects the output obtained from the other. Therefore the order of controller must be taken into account during designing such configuration.

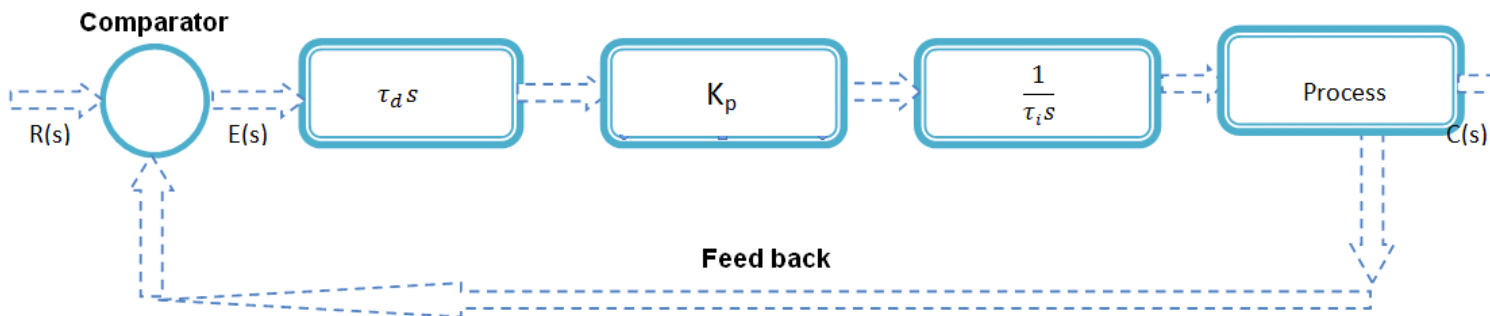


Figure 3.4.11 Series PID controller

Detail analysis and study of these controllers is out of the scope of the present course. However readers are advised to read following books for further study:

- a) PID controllers 2nd ed., by Karl J. Astrom and Tore Hagglund, Research Triangle Park, ISA 1995.
- b) Structure and synthesis of PID controllers, by Aniruddha Dutta, Ming-Tzu Ho and Shankar P. Bhattacharyya, Springer-Verlag, Londoan 2000.

PID controllers have wide variety of applications manufacturing industry. Some of them are listed as follows.

1. PID control is used in automatic car steering when it is integrated with Fuzzy Logic
2. In movement detection system of modern seismometer
3. In water/oil level monitoring in tanks
4. Head positioning of a disk drive
5. Automated inspection and quality control
6. Manufacturing process control: CNC machine tools
7. Chemical process control: flow control, temperature control
8. Automatic control of material handling equipments
9. Automatic packaging and dispatch
10. To ensure safety during manufacturing operations