# Design Microcontroller Based- Digital Control System

[1]Pawan Kumar [2]DR.C.RAM SINGLA

[1]Research Scholar, Singhania University, Jhunjhunu, Rajasthan, India

[2]Advisor(R & D)Prof in ECE,Dronacharya College of Engg. & Technology, Gurgaon- 123506, India

***Abstract: The basic principles of designing a digital controller, from the identification of the system to the implementation of a suitable controller algorithm on a microcontroller. Here the classical Ziegler–Nichols PI type This program implements a first-order digital controller module on a PIC16F877 (or equivalent) microcontroller. The microcontroller operates with a 4MHz crystal. The analog input AN0 of the microcontroller is connected to the output sensor of the plant (y). The PORT B output of the microcontroller is connected to an AD7302 type D/A converter. The WR input of the controller is controlled from port pin RC0 of the microcontroller. The sampling interval is 0.1s (100ms) and the timer interrupt service routine is used to obtain the required sampling interval***

----------------------------------------♦------------------------------------

## INTRODUCTION

Control engineering is concerned with controlling a dynamic system or plant. A dynamic system can be a mechanical system, an electrical system, a fluid system, a thermal system, or a combination of two or more types of system. The behaviour of a dynamic system is described by differential equations. Given the model (differential equation), the inputs and the initial conditions, we can easily calculate the system output.

A plant can have one or more inputs and one or more outputs. Generally a plant is a continuous-time system where the inputs and outputs are also continuous in time. For example, an electromagnetic motor is a continuous-time plant whose input (current or voltage) and output (rotation) are also continuous signals. A control engineer manipulates the input variables and shapes the response of a plant in an attempt to influence the output variables such that a required response can be obtained. A plant is an *open-loop* system where inputs are applied to drive the outputs. For example, a voltage is applied to a motor to cause it to rotate. In an open-loop system there is no knowledge of the system output. The motor is expected to rotate when a voltage is applied across its terminals, but we do not know by how much it rotates since there is no knowledge about the output of the system. If the motor shaft is loaded and the motor slows down there is no knowledge about this. A plant may also have disturbances affecting its behaviour and in an open-loop system there is no way to know, or to minimize these disturbances.

output to a known point (e.g. to rotate the motor shaft at a specified rate). This is a single-input, single-output (SISO) system, since there is only one input and also only one output is available.

In general, systems can have multiple inputs and multiple outputs (MIMO). Because of the unknowns in the system model and the effects of external disturbances the open-loop control is not attractive. There is a better way to control the system, and this is by using a sensor to measure the output and then comparing this output with what we would like to see at the system output. The difference between the desired output value and the actual output value is called the *error signal*. The error signal is used to force the system output to a point such that the desired output value and the actual output value are equal. This is termed *closedloop* advantages of closed-loop control is the ability to compensate for disturbances and yield the correct output even in the presence of disturbances. A *controller* (or a *compensator*) is usually employed to read the error signal and drive the plant in such a way that the error tends to zero.

## HARDWARE REQUIREMENTS FOR COMPUTER CONTROL

## GENERAL PURPOSE COMPUTERs

In general, although almost any digital computer can be used for digital control there are some requirements that should be satisfied before a computer is used for such an application. Today, the majority of small and medium scale DDC-type applications are based on microcontrollers which are used as embedded controllers. Applications where user interaction and supervisory control are required are commonly designed around the standard PC hardware. A general purpose computer consists of the following basic building elements:

_ central processing unit (CPU);

_ program memory;

_ data memory;

_ input–output devices.

The CPU is the part which contains the arithmetic and logic unit (ALU), the control unit (CU) and the general purpose registers (GPR). The ALU consists of the logic circuitry necessary to carry out arithmetic and logic operations, for example to add or subtract numbers, to compare numbers and so on. Some ALU units are equipped to carry out multiplication and division and floating point mathematical operations. The CU supervises the operations within the CPU, fetches instructions from the program memory, decodes these instructions and controls the ALU and other parts of the computer so that the required operations can be implemented. The GPR are a set of fast registers which are generally used to carry out fast operations within the CPU. The program memory of a general purpose computer is usually an external unit and attached to the computer via the data bus and the address bus. A bus is a collection of conductors which carry electrical signals. The data bus is a bidirectional bus which carries the data to be sent or received between the CPU and the other parts of the computer. The size of this bus is 8 bits in most microprocessors and microcontrollers. Some microcontrollers have data buses that are 16 or even 32 bits wide. Minicomputers and mainframe computers usually have 64 or even higher data widths. The address bus is a unidirectional bus which is used to address the peripheral

## 8 INTRODUCTION

CU ALU GPR

Program

Memory

Data

Memory

Inputs Outputs

Data Bus

Address Bus

Control Bus

## CPU

devices attached to the computer. For example, when data is to be written to the memory the address of the memory location is sent on the address bus and the actual data byte is sent on the data bus. The program memory is usually a nonvolatile memory, such as electrically programmable read-only memory (EPROM), EEPROM or flash memory. EPROM memory can be programmed using a suitable programmer device. This type of memory has to be erased using an ultraviolet light source before the contents can be changed. EEPROM memory can be programmed and erased by sending electrical signals to the memory. The disadvantage of this memory is that it is usually a slow process to write or read data from an EEPROM memory.

Currently, flash memory is one of the most popular types of nonvolatile memory used. Flash memory is fast and can be erased under program control. The data memory is usually a volatile memory, used to store the user data. RAM type memories are commonly used for this purpose. The size of this memory can vary from several tens of kilobytes to tens of gigabytes.

Minicomputers and larger computers are equipped with auxiliary storage mediums such as hard disks and magnetic tapes. These devices provide bulk storage for programs and data. Magnetic tape is usually used to store the entire contents of a hard disk for backup purposes. Input–output devices are also known as the peripheral devices. Many different types of input devices – scanner, camera, keyboard, microphone and mouse – can be connected to the computer. The output devices can be printers, plotters, speakers, visual display units and so on.

General purpose computers are usually more suited to data processing type applications.

For example, a minicomputer can be used in an office to provide word processing. Similarly, a large computer can be used in a bank to store and manipulate the accounts of thousands of customers.

## MICROCONTROLLERS

A microcontroller is a single-chip computer that is specifically manufactured for embedded computer control applications. These devices are very low-cost and can be used very easily SOFTWARE REQUIREMENTS FOR COMPUTER CONTROL **9**

in digital control applications. Most microcontrollers have the built-in circuits necessary for computer control applications. For example, a microcontroller may have A/D converters so that the external signals can be sampled. They also have parallel input–output ports so that digital data can be read or output from the microcontroller. Some devices have built-in D/A converters and the output of the converter can be used to drive the plant through an actuator (e.g. an amplifier). Microcontrollers may also have built-in timer and interrupt logic. Using the timer or the interrupt facilities, we can program the microcontroller to implement the control algorithm accurately.

Microcontrollers have traditionally been programmed using the assembly language of the target device. As a result, the assembly languages of the microcontrollers manufactured by different firms are totally different and the user has to learn a new language before being able program a new type of device. Nowadays microcontrollers can be programmed using high level languages such as BASIC, PASCAL or C. High-level languages offer several advantages compared to the assembly language:

_ It is easier to develop programs using a high-level language.

_ Program maintenance is much easier if the program is developed using a high-level language.

_ Testing a program developed in a high-level language is much easier.

_ High-level languages are more user-friendly and less prone to making errors.

_ It is easier to document a program developed using a high-level language.

In addition to the above advantages, high-level languages have some disadvantages. For example, the length of the code in memory is usually larger when a high-level language is used, and the programs developed using the assembly language usually run faster than those developed using a high-level language. In this book, PIC microcontrollers are used as digital controllers. The microcontrollers are programmed using the high-level C language.

## SOFTWARE REQUIREMENTS FOR COMPUTER CONTROL

Computer hardware is nowadays very fast, and control computers are generally programmed using a high-level language. The use of the assembly language is reserved for very special and time-critical applications, such as fast, real-time device drivers. C is a popular language used in most computer control applications. It is a powerful language that enables the programmer to perform low-level operations, without the need to use the assembly language.

The software requirements in a control computer can be summarized as follows:

_ the ability to read data from input ports;

_ the ability to send data to output ports;

_ internal data transfer and mathematical operations;

_ timer interrupt facilities for timing the controller algorithm.

All of these requirements can be met by most digital computers, and, as a result, most computers can be used as controllers in digital control systems. The important point is that it is not justified and not cost-effective to use a minicomputer to control the speed of a motor, for example. A microcontroller is much more suitable for this kind of control application. On the other hand, if there are many inputs and many outputs, and if it is required to provide supervisory tasks as well then the use of a minicomputer can easily be justified.

The controller algorithm in a computer is implemented as a program which runs continuously in a loop which is executed at the start of every sampling time. Inside the loop, the desired reference value is read, the actual plant output is also read, and the difference between the desired value and the actual value is calculated. This forms the error signal. The control algorithm is then implemented and the controller output for this sampling instant is calculated.

This output is sent to a D/A converter which generates an analog equivalent of the desired control action. This signal is then fed to an actuator which in turn drives the plant to the desired point.

The operation of the controller algorithm, assuming that the reference input and the plant output are digital signals, is summarized below as a sequence of simple steps:

```
REPEAT FOREVER
When it is time for next sampling instant
_ Read the desired value, $R$
_ Read the actual plant output, $Y$
_ Calculate the error signal, $E = R - Y$
_ Calculate the controller output, $U$
_ Send the controller output to D/A converter
_ Wait for the next sampling instant
END
```

Similarly, if the reference input and the plant output are analog signals, the operation of the controller algorithm can be summarized as:

```
REPEAT FOREVER
When it is time for next sampling instant
_ Read the desired value, $R$, from A/D converter
_ Read the actual plant output, $Y$ , from the A/D converter
_ Calculate the error signal, $E = R - Y$
_ Calculate the controller output, $U$
_ Send the controller output to D/A converter
_ Wait for the next sampling instant
END
```

One of the important features of the above algorithms is that once they have been started they run continuously until some event occurs to stop them or until they are stopped manually by an operator. It is important to make sure that the loop is run continuously and exactly at the same times, i.e. exactly at the sampling instants. This is called synchronization and there are several ways in which synchronization can be achieved in practice, such as:

_ using polling in the control algorithm;

_ using external interrupts for timing;

_ using timer interrupts;

## SOFTWARE REQUIREMENTS FOR COMPUTER CONTROL

_ ballast coding in the control algorithm;

_ using an external real-time clock.

These methods are discussed briefly here.

## POLLING

Polling is the software technique where we keep waiting until a certain event occurs, and only then perform the required actions. This way, we wait for the next sampling time to occur and only then run the controller algorithm. The polling technique is used in DDC applications since the controller cannot do any other operation during the waiting of the next sampling time. The polling technique is described below as a sequence of steps:

```
REPEAT FOREVER
While Not sampling time
WAIT
END
_ Read the desired value, $R$
_ Read the actual plant output, $Y$
_ Calculate the error signal, $E = R - Y$
_ Calculate the controller output, $U$
_ Send the controller output to D/A converter
END
```

## USING EXTERNAL INTERRUPTS FOR TIMING

The controller synchronization task can easily be performed using an external interrupt. Here, the controller algorithm can be written as an interrupt service routine (ISR) which is associated with an external interrupt. The external interrupt will typically be a clock with a period equal to the required sampling time. Thus, the computer will run the interrupt service (i.e. the algorithm) routine at every sampling instant. At the end of the ISR control is returned to the main program where the program either waits for the occurrence of the next interrupt or can perform other tasks (e.g. displaying data on a LCD) until the next external interrupt occurs.

The external interrupt approach provides accurate implementation of the control algorithm as far as the sampling time is concerned. One drawback of this method is that an external clock is required to generate the interrupt pulses.

The external interrupt technique has the advantage that the controller is not waiting and can perform other tasks in between the sampling instants.

The external interrupt technique of synchronization is described below as a sequence of steps:

**MAIN PROGRAM:**

Wait for an external interrupt (or perform some other tasks)

**END**

**INTERRUPT SERVICE ROUTINE (ISR):**

_ Read the desired value, *R*

_ Read the actual plant output, *Y*

_ Calculate the error signal, $E = R - Y$

_ Calculate the controller output, *U*

_ Send the controller output to D/A converter

**RETURN FROM INTERRUPT**

**USING TIMER INTERRUPTS**

Another popularway to perform controller synchronization is to use the timer interrupt available on most microcontrollers. Here, the controller algorithm is written inside the timer interrupt service routine, and the timer is programmed to generate interrupts at regular intervals, equal to the sampling time. At the end of the algorithm control returns to the main program, which either waits for the occurrence of the next interrupt or performs other tasks (e.g. displaying data on an LCD) until the next interrupt occurs.

The timer interrupt approach provides accurate control of the sampling time. Another advantage of this technique is that no external hardware is required since the interrupts are generated by the internal timer of the microcontroller.

The timer interrupt technique of synchronization is described below as a sequence of steps:

**MAIN PROGRAM:**

Wait for a timer interrupt (or perform some other tasks)

**END**

**INTERRUPT SERVICE ROUTINE (ISR):**

_ Read the desired value, *R*

_ Read the actual plant output, *Y*

_ Calculate the error signal, $E = R - Y$

_ Calculate the controller output, *U*

_ Send the controller output to D/A converter

**RETURN FROM INTERRUPT**

**BALLAST CODING**

In this technique the loop timing is made to be independent of any external or internal timing signals. The method involves finding the execution time of each instruction inside the loop and then adding *dummy* code to make the loop execution time equal to the required sampling time.

This method has the advantage that no external or internal hardware is required. But one big disadvantage is that if the code inside the loop is changed, or if the CPU clock rate of the microcontroller is changed, then it will be necessary to readjust the execution timing of the loop.

SOFTWARE REQUIREMENTS FOR COMPUTER CONTROL

The ballast coding technique of synchronization is described below as a sequence of steps. Here, it is assumed that the loop timing needs to be increased and some dummy code is added to the end of the loop to make the loop timing equal to the sampling time:

**DO FOREVER:**

_ Read the desired value, *R*

_ Read the actual plant output, *Y*

```
_ Calculate the error signal, $E = R - Y$
_ Calculate the controller output, $U$
_ Send the controller output to D/A converter
Add dummy code
. . .
. . .
Add dummy code
END
```

## USING AN EXTERNAL REAL-TIME CLOCK

This technique is similar to using an external interrupt to synchronize the control algorithm.

Here, some real-time clock hardware is attached to the microcontroller where the clock is updated at every *tick*; for example, depending on the clock used, 50 ticks will be equal to 1 ms if the tick rate is 20 ms. The real-time clock is then read continuously and checked against the time for the next sample. Immediately on exiting from the wait loop the current value of the time is stored and then the time for the next sample is updated by adding the stored time to the sampling interval. Thus, the interval between the successive runs of the loop is independent of the execution time of the loop. Although the external clock technique gives accurate timing, it has the disadvantage that real-time clock hardware is needed. The external real-time clock technique of synchronization is described below as a sequence of steps. $T$ is the required sampling time in ticks, which is set to $n$ at the beginning of the algorithm. For example, if the clock rate is 50 Ticks per second, then a Tick is equivalent to 20 ms, and if the required sampling time is 100 ms, we should set $T = 5$:

```
$T = n$
Next Sample Time = Ticks + $T$
DO FOREVER:
While Ticks < Next Sample Time
Wait
END
Current Time = Ticks
_ Read the desired value, $R$
_ Read the actual plant output, $Y$
_ Calculate the error signal, $E = R - Y$
_ Calculate the controller output, $U$
```

## REFERENCES

1. "PICmicro Family Tree", PIC16F Seminar Presentation
2. "MOS DATA 1976", General Instrument 1976 Databook
3. "1977 Data Catalog", Micro Electronics from General Instrument Corporation
4. Microchip Technology . "Microchip Technology Delivers Six Billionth PIC Microcontroller".
5. Microchip Product Selector
6. "PIC Paging and PCLATH"
7. *PIC10F200/202/204/206 Data Sheet*. Microchip Technology.
8. "PIC24H Family Overview".
9. 32-bit PIC® MCUs".
10. "MPLAB REAL ICE In-Circuit Emulator Product Overview"
11. Microcontroller Based Applied Digital Control - Dogan Ibrahim