Real-Time Systems
(0630581)

Lecture (5)

Computer Software Requirements for Real-Time Applications

Prof. Kasim M. Al-Aubidy
Computer Engineering Department
Philadelphia University
Summer Semester, 2011

Lecture Outline:

- Microcomputer programming.
- Programming techniques.
- Software requirements for real-time systems
- Program development process.

Microcontroller Programming:

Microcontroller Programming:

- MCs 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.
- MCs can be programmed using high level languages such as BASIC, and C.

HLL Advantages:

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.

HLL Disadvantages:

- The length of the code in memory is usually larger when a high-level language is used.
- The programs developed using the assembly language usually run faster than those developed
- using a high-level language.

- 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 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. 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;
  - ballast coding in the control algorithm;
  - using an external real-time clock.
- These methods are discussed briefly through lecture.

Programming Techniques:

1. Using 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

2. Using External Interrupts for Timing:

- The controller synchronization task can easily be performed using an external interrupt.
- The controller algorithm can be written as an interrupt service routine (ISR).
- The external interrupt will typically be a clock with a period equal to the required sampling time. Thus, the computer will run the ISR 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

3. Using Timer Interrupts:

- Another popular way to perform controller synchronization is to use the timer interrupt available on most microcontrollers.
- The controller algorithm is written inside the timer ISR, 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 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

4. Using 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 MC is changed, then it will be necessary to readjust the execution timing of the loop.
- 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

5. Using an External Real-Time Clock:

- This technique is similar to using an external interrupt to synchronize the control algorithm. Here, some RT 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 s if the tick rate is 20 ms. The RT 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 RT clock hardware is needed.
- The external RT 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

Send the controller output to D/A converter

Next Sample Time=Current Time + T

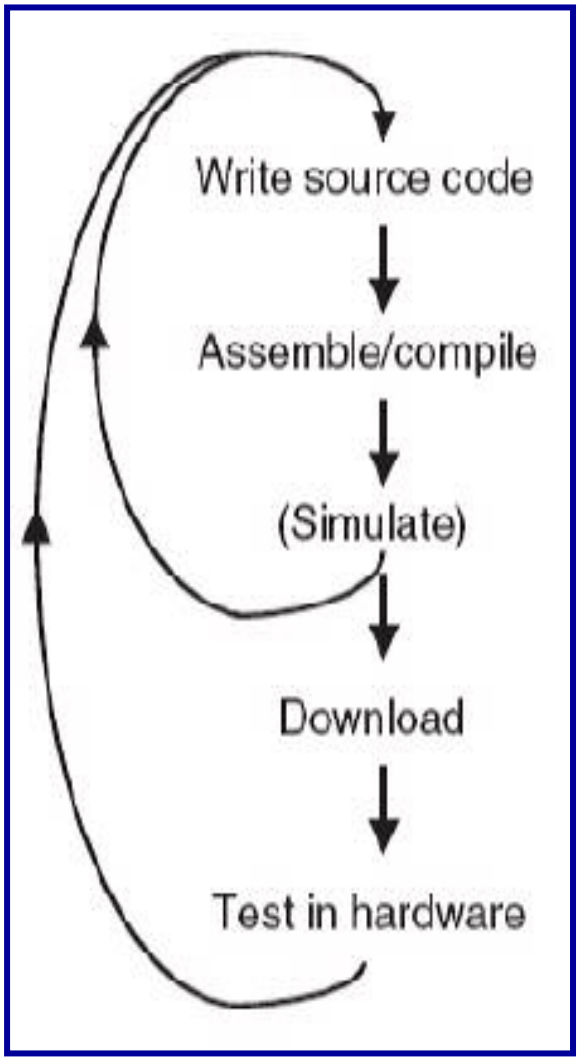
End

Software Requirements for Real-Time Systems:

- 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 real-time systems 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.

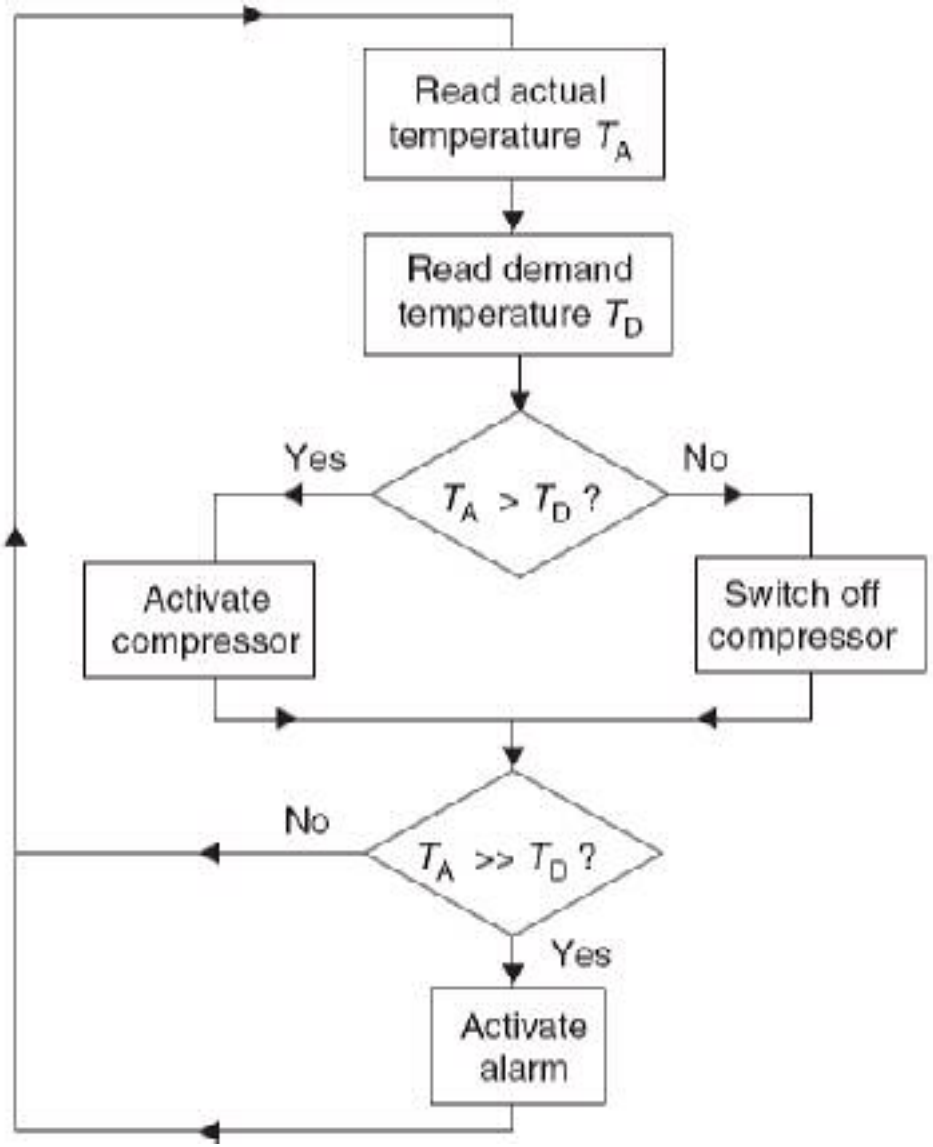
Program Development Process:

- The programmer writes the program, called the source code, in Assembler language.
- This is then assembled by the Cross-Assembler running on the host computer. The designer may choose to test the program by simulation. This is likely to lead to program.
- When satisfied with the program, the developer will then download it to the program memory of the microcontroller itself, using either a stand-alone ‘programmer’ linked to the host computer or a programming facility designed into the embedded system itself.
- The designer will then test the program running in the actual hardware. Again, this may lead to changes being required in the source code.
- To develop a simple project, a selection of different software tools is beneficial. These are usually bundled together into what is called an Integrated Development Environment (IDE), such as PROTUS and MPLAB.



Example:

Software design using flowcharts.



Example:

Software design using State diagrams.