DEPARTMENT
OF
ELECTRONICS & COMMUNICATION ENGINEERING

EMBEDDED SYSTEM DESIGN

(Theory Notes)

Autonomous Course

Prepared by

Prof. Abhishek M B

Module – 4 Contents
Interfacing, Peripherals and Communication Protocols: Interfacing General Purpose Microprocessors, Timers, Watchdog Timers, Counting Devices, PWM, LCD, UART, Keypad Controller, Stepper Motor Controller, ADC, Serial Protocols: I2C, CAN, USB, Parallel Protocols: PCI bus, ARM bus Wireless Protocols: IEEE 802.11

Dayananda Sagar College of Engineering

Shavige Malleshwara Hills, Kumaraswamy Layout,
Banashankari, Bangalore-560078, Karnataka
Tel : +91 80 26662226 26661104 Extn : 2731 Fax : +90 80 2666 0789
Web - http://www.dayanandasagar.edu Email : hod-ece@dayanandasagar.edu
(An Autonomous Institute Affiliated to VTU, Approved by AICTE & ISO 9001:2008 Certified)
(Accredited by NBA, National Assessment & Accreditation Council (NAAC) with 'A' grade)

STANDARD SINGLE-PURPOSE PERIPHERALS

❖ TIMERS :-

A timer is an peripheral device that can measure time intervals. Timer can be used to

i) Generat events at specific time or to determine the duration between two external events.

eg: Keeping a traffic light green for a specified duration or communicating bits serially between devices at a specific rate.

ii) To determine duration of two external events.

eg: Computing a cars speed by measuring the time the car takes to pass over two separated sensors in a road.

A timer measures time by counting pulses that occur on an input clock signal having a known period.

❖ COUNTERS :-

A counter counts pulses onsome other input signal.

eg: A counter may be used to count the number of cars that pass over a road sensor ot thr number of people that pass through a turnstile.

❖ **TIMERS & COUNTERS** are combined to measure rates.

eg: Counting the number of times the car wheel rotates in one second, in order to determine a cars speed.

❖ Timer Structure :-

The timer structures are

1) Basic Timer
2) A Timer/Counter
3) A Timer with a terminal count
4) A 16/32-bit Timer
5) A Timer with a prescaler

---

1) Basic Timer :-

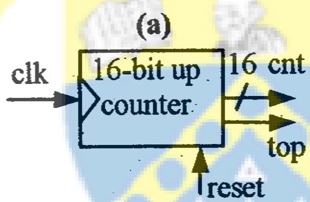


Fig ①: A Basic timer

* The timer has an internal 16 bit upcounter which increments its value on each clockpulse

* The output value cnt represent the number of pulses since the counter was last reset to zero.

* It has additional output top that indicates when the top value of its range has been reached. It is also known as overflow occuring in which case the timer rolls over to zero

* We define a timer range as the maximum time

interval the timer can measure

* Also, a time resolution is the minimum interval it can measure

1) If the given datas are

$f = 100MHz$, $Cnt = 20,000$.

Find resolution and range.

Soln :-

* Resolution $= \frac{1}{f} = \frac{1}{100MHz} = 10nsec$

* Range $= Cnt \times Resolution = 20,000 \times 10nsec = 200\mu sec$

2) If the given data are $f = 100MHz$, counter = 16-bit find resolution and range.

Soln: 16-bit counter will count from 0 to 65,535

$\therefore$ cnt = 65,535

* Resolution $= \frac{1}{f} = \frac{1}{100MHz} = 10nsec$

* Range $= cnt \times Resolution = 65,535 \times 10nsec = 655.35\mu sec$

2) A Timer/Counte :-

* The mode register holds a bit by which the user can set that uses a 2x1 Multiplexer to select the clock input to the internal 16-bit counter

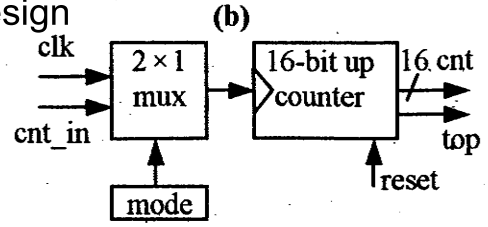


Fig ①: A timer/counter

* The clock input can be the external clock signal, in which case the device acts like a timer.

* The clock input can be the external clock signal, in which case the the device acts like a timer

* The clock input can be the external cnt-in signal, in which case the device acts like a counter, counting the occurrence of pulses on cnt-in.

* Cnt-in would typically be connected to an external sensor, so pulses would occur at indeterminate intervals.

3) A Timer with a terminal count :-

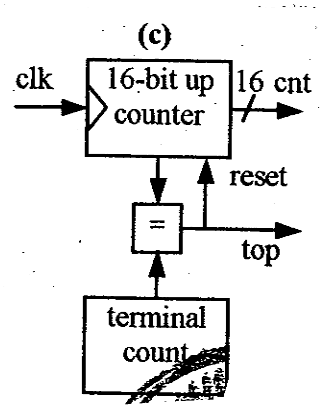


Fig ①: A timer with a terminal count

* A Terminal count register holds a value, which the user sets, indicating the number of clock cycles, in the desired interval & is determined by using simple formulae :

$$\boxed{\text{Number of cycles} = \frac{\text{Desired time interval}}{\text{Clock period}}}$$

For ex, to obtain a duration of 3 microseconds from a clock cycle of 10nsec (f = 100MHz), then

$$\text{No of clock cycles} = \frac{3\times10^{-6}}{10\times10^{-9}} = 300 \text{ cycles.}$$

* The timer structure includes the comparator that asserts the top output (ie top=1) when the terminal count has been reached.

The top output is used to :

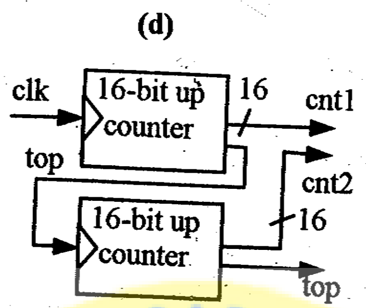
i) Reset the counter to zero and.

ii) To inform the user that the desired time interval has passed.

* To top signal is often connected to an interrupt (Pin). The corresponding ISR would include the actions that must be taken at the specified time interval.

* To improve efficiency, a down counter is used rather than an up counter.

4) A 16/32-bit Timer :-



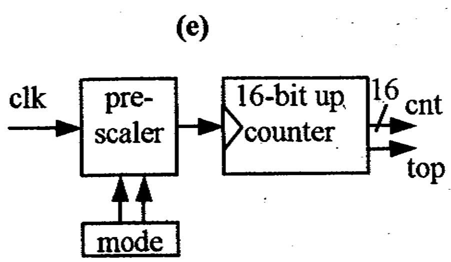
Fig①: A 16/32 bit timer

* Fig-① shows the structure of a timer that can be configured as a 16 or 32 bit timer.

The timer simply uses the top output of its first 16 bit upcounter as the clock input of its second 16 bit counter.

These are known as cascaded counters.

5) A Timer with a prescaler :-



Fig①: A timer with a prescalar

* A prescaler is essentially a configurable clock divider circuit. Depending on the mode bits being input to the prescaler.

* The prescalar o/p signal might be

i) Same as the input signal frequency

ii) 1/2 of the frequency.

iii) 1/4 of the frequency

iv) 1/8 of the frequency.

Mode bits	o/p frequency
0 0	Same as I/p Signal
0 1	1/2 of I/p signal
1 0	1/4 of I/p signal
1 1	1/8 of I/p signal.

Eg :- consider a timer with a resolution of 10 nsec and a range of 65,535. If the prescalar is configured to divide the clock frequency by 8, then calculated timer resolution.

Sol :-

* Resolution = 10 nsec × 8 = 80 nsec

* Range = cnt × Resolution = 65,535 × 80 nsec

$\boxed{\text{Range} = 5.24\ \mu\text{sec}}$

REACTION Timer :-

Reaction Timer is an application that measures the time a person takes to respond to a visual or audio stimulus.

(a)

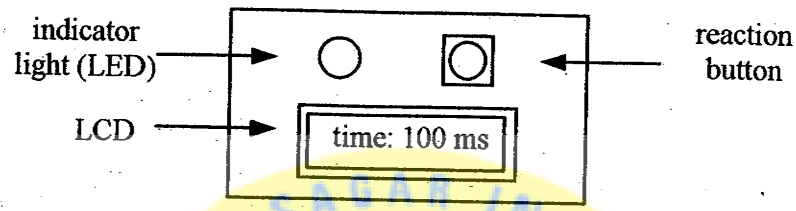


Fig ①: Reaction timer : LED, LCD and button

* The application turns on a LED, then measures the time a person takes to push a button in response, and display this time on an LCD.

  Reaction time is expected to be in the order of seconds and displays reaction time to be miliseco-nd precision.

* In this example, we will use a microcontroller with a built in 16-bit timer. The timer is incremented once every instruction cycle for this microcontroller is 6 clock cycles.

* The Timer doesnot have a prescaler or Terminal count register.

* It has a top signal to indicate overflow also allows us to load an initial value for its internal up counter.

* If the clock frequency is 12 MHz, then

$$period = \frac{1}{f} = \frac{1}{12MHz} = 83.33\,nsec$$

* Each instruction has 6 clock cycles

Resolution = 1 instruction cycle $\times \frac{1}{f}$

Resolution $= 6 \times 83.33\,nsec$

$$\boxed{Resolution = 0.5\mu sec}$$

* The timer is of 16-bit, so maximum value is

$2^{16} = 65,536$

$\therefore$ Range = Resolution $\times$ Maximum value $[0 - 65536 = 65,536]$

$= 0.5\mu sec \times 65,536$

$$\boxed{Range = 32.77\,msec}$$

* We note that this timer's range is smaller than our desired range of several seconds, while its resolution is finer than our required 1msec.

* The initial timer value can be set that the overflow occurs every 1msec. Then we can monitor the top output signal of the timer to activate the code. The code keeps a count of overflows, indicating the number of miliseconds.

* The initial value to be loaded is determined

ie initial value = Maximum value of counter $- \left[\text{Required time} \times \frac{\text{Oscillator freq}}{\text{No of cycles per inst}} \right]$

$= 65,536 - \left[1msec \times \frac{12MHz}{6\ cycles} \right]$

$= 65,536 - 2000$

• Initial value = 63,536

* The Pseudocode describing the reaction timer implementation is written below.

```
/* main. c*/
# define MS_INIT 63535
void main(void)
{
    int count_miliseconds = 0;
    configure timer mode
    set cnt to MS_INIT
    Wait a random amount of time
    turn ON indicator light
    Start timer
    While (user has not pushed reaction button)
    {   if (top)
        { Stop timer
          set cnt to MS_INIT
          Start timer
          reset top
          count_milliseconds ++;
        }
    }
    turn off indicator light
    printf("Time: %i ms", count, miliseconds);
```

FORMULAE

1) Resolution = 1 instruction cycle × $\frac{1}{f}$

2) Range = Resolution × Maximum value (count)

NOTE :-

Resolution = $\frac{1}{f}$

3) Maximum division needed in a prescalar = $\frac{\text{Prescalar measurement}}{\text{Max value of the counter}}$

4) Terminal count = $\frac{\text{Desired Time Interval}}{\text{clock period } (T = \frac{1}{f})}$

---

1) A 16-bit timer operates at a clock frequency of 12MHz. Determine the resolution and range of this timer.

Given :- $f = 12\ MHz$

Timer = 16-bit

Maximum value = $2^{16} - 1 = 65,535$

sol :- * Resolution = $\frac{1}{f} = \frac{1}{12\ MHz}$

$\boxed{\text{Resolution} = 83.33\ nsec}$

* Range = Resolution × Maximum value

= 83.33 nsec × 65,535

$\boxed{\text{Range} = 5.5\ msec}$

2) Determine the range and resolution of a 16-bit timer which operates at a clock frequency of 10Mhz and generate an overflow signal when it reaches FFFF. Calculate the terminal count value for measuring a 3 msec time interval. What is the minimum division needed in a prescaler for measuring 100 msec?.

June-09,6M

Given :-

$f = 10 MHz$

16- bit timer

$\therefore$ Maximum value $= 2^{16} - 1 = 65,535$

Time interval $= 3 msec$

prescalar measurement $= 100 msec$

Sol:- * Resolution $= \frac{1}{f} = \frac{1}{10 MHz}$

$\boxed{\text{Resolution} = 0.1 \mu sec}$

* Range = Resolution × Max value

$= 0.1 \mu sec \times 6$

$\boxed{\text{Range} = 6.5535 msec}$

* Terminal Count $= \frac{\text{Desired time interval}}{\text{period } (T = \frac{1}{f})}$

$= \frac{3 msec}{0.1 \mu se}$

$\boxed{\text{Terminal Count} = 30,000 \text{ cycles.}}$

* Prescalar minimum division $= \frac{\text{Prescaler measurement}}{\text{Max value}}$

$= \frac{100 msec}{65,535}$

$\boxed{\text{Prescaler minimum division} = 1.52 \times 10^{-6}}$

3) Given a 16-bit timer with 20 Mhz,

i) Determine its range and resolution

ii) Calculate the terminal count value needed to measure 1.5 msec interval

iii) If a prescalar is added, what is the minimum division needed to measure an interval of 50 msec. Determine its range and resolution, if the division value is a power of 2.

Jan-08,6M

Given :- $f = 20MHz$, $T = \frac{1}{f} = 50nsec$

Timer is 16 bit $\therefore$ Maximum value $= 2^{16} - 1 = 65,536$

Soln : (a) Resolution $= \frac{1}{f} = \frac{1}{20MHz} = 50\,nsec$

Range = Resolution × Maximum value $= 50nsec \times 65,535$

$\boxed{\text{Range} = 3.27675\ msec}$

(b) Desired time interval $= 1.5\,msec$

Terminal count $= \frac{\text{Desired time interval}}{\text{period } (T = \frac{1}{f})}$

$= \frac{1.5\,msec}{50\,nsec}$

$\boxed{\text{Terminal count} = 30,000}$

(c) Prescaler measurement interval $= 50msec$

prescaler minimum division $= \frac{\text{Prescaler measurement}}{\text{Maximum value}}$

$= \frac{50 msec}{65,535}$

$$\boxed{\text{prescal division minimum} = 0.762\ \mu sec}$$

* Resolution $= \frac{1}{f} = 50 nsec$

Range $= 2^{2 \times 16} *$ Resolution

$$\boxed{\text{Range} = 214.748\ sec}$$

Note :-

* The maximum value of timer $= 2^{16}$
* Determine its range & resolution if the division value is a power of 2.

ie Maximum value of time $= 2^{2 \times 16} = 2^{32}$

WATCHDOG TIMER :-

❖ **What is watchdog timer? Explain ATM timeout using a watchdog timer.**

Jan-11,10M

❖ **With a neat diagram, explain functioning of a watch dog timer. Discuss the usage of watch dog timers. Write a Pseudo code for an ATM machine to demonstrate the usage of watch dog timer.**

Jan-10,10M

A Special type of timer is a watchdog timer, which will reset the system after a predefined timeout.

Watchdog timer reset timer every X time unit, else timer generates a signal indicating that the system failed.

Common use of watchdog timer is to enable an embedded system to restart itself in case of a failure.

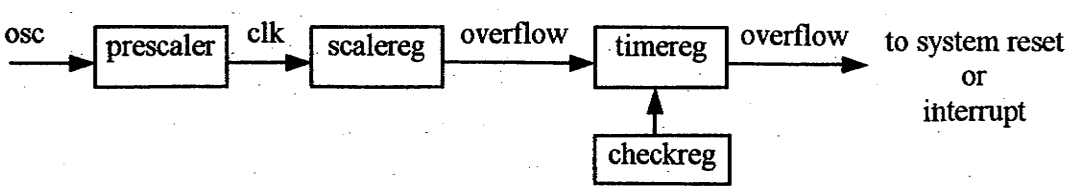
Another common use is to support timeouts in a program while keeping the program structure simple.

Example of ATM timeout using a watchdog timer :-

* In this example, a watchdog timer is used to implement a time out for an automatic teller machine (ATM).

* A normal ATM session involves a user inserting a bank card, typing in a Personal identification number (PIN), and then answering questions about whether to deposit or withdraw money, which account will be involved, how much money will be involved, whether another transaction is desired, and so on.

* We want to design the ATM such that it will terminate the session if at any time the user doesnot press any button for 2 minutes. In this case the ATM will eject the bank card and terminate the session.



Fig(): ATM timeout using a watchdog timer Structure.

* An oscillator signal <u>OSC</u> is connected to prescaler that divides the <u>oscillator frequency by 12</u> (OSC/12) to generate a <u>signal clk</u>

* The signal clock is connected to an 11-bit up counter Scalereg. When scalereg overflows, it rolls over to '0', and its overflow output causes the 16 bit up-counter time reg to increment.

* If timereg overflows, it triggers the System reset or an interrupt. To reset the watchdog Timer, checkreg must be enabled. Then a value can be loaded into timereg.

* When a value is loaded into timereg, the <u>checkreg</u> register is automatically reset. If the checkreg register is not enabled, a value cannot be loaded into timereg. This is to prevent erroneous software from unintentionally resetting the watchdog timer.

* Let us determine what value to load in <u>timereg</u> to achieve a timeout of <u>2 minutes</u>

The osc signal frequency is 12MHz. The <u>timereg</u> is incremented at every 't' seconds, where

$$t = \text{Prescalar} \times \text{Scalereg} \times \frac{1}{\text{osc freq}}$$

$$= 12 \times 2^{11} \times \frac{1}{12 \times 10^6}$$

$$t = 0.002 \text{ sec}$$

$\therefore$ $\boxed{\text{Watchdog timer resolution} = 2 \text{msec}}$

* Since Timereg is a 16-bit register, its range is 0 to 65,535.

∴ Time range = Max count × Resolution

= 65,535 × 2 msec

$\boxed{\text{Timer range} = 131.070 \text{ msec}}$

(Approximately 2.18 minutes)

* To obtain timereg value:

Timereg value = 131,070 − X

X = 131,070 − Timereg value

(WKT timereg value = 2 minutes = 120000 msec)

X = 131070 − 120000

$\boxed{X = 11070}$

* The pseudocode for the main routine and the watchdog reset routine to implement the timeout functionality of the ATM is shown below:

<u>Main pseudo-code:</u>

```
/* main.c */
main ( )
{ wait until card inserted
  call watchdog_reset_routine
  While ( transaction in progress)
  { if ( button pressed)
     { perform corresponding action
       call watchdog_reset_routine
     }
```

```
    /* if watchdog_reset_routine not called every < 2 minutes,
    interrupt_service_routine is called */
  }
}
```

Watchdog timer reset routine :-

```
Watchdog_reset_routine ( )
{  /* Checkreg is set so we can load value into Timereg.
   Zero is loaded into scalereg and 11070 is loaded into
   Timereg */

   checkreg = 1
   scalereg = 1
   timereg  = 11070
}
void interrupt_service_routine ( )
{
   eject card
   reset screen
}
```

❖ **Explain how UART is used for communication highlighting the advantages of UART.**

June-07,6M

UART takes parallel data and transmits serially & UART receives serial data and converts to parallel.

A simple UART may possess

i) **Some configuration registers &**

ii) **Two independently operating processors, one for receiving and the other for transmitting.**

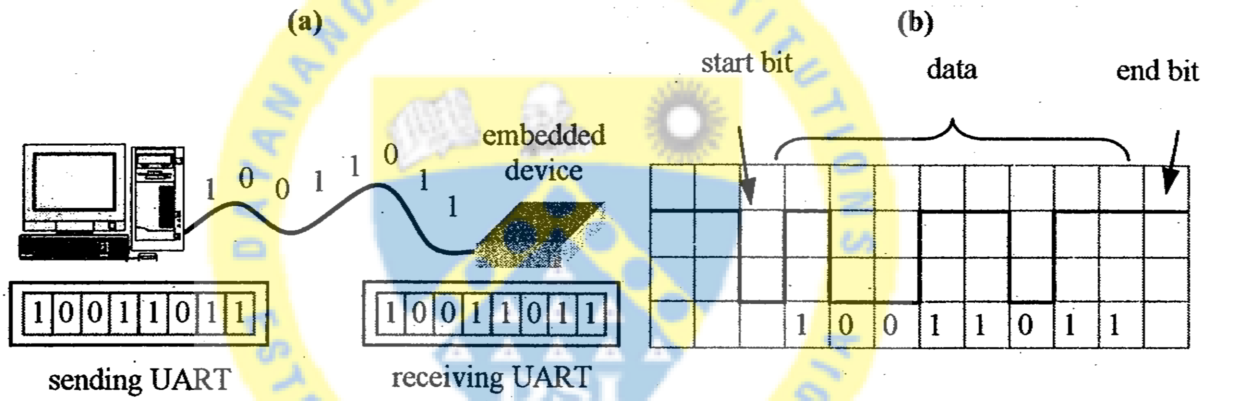


Figure 1 : Serial transmission using UARTs, (a) A PC communicating serially with an embedded device, (b) transmission protocol used by the two UARTs.

* The transmitter may posses a register called a transmit buffer, that holds data to be sent. This register is a shift register, so the data can be transmitted one bit at a time by shifting at the appropriate rate.

* The receiver receives data into a shift register, and then the data can be read in parallel

The receiver is constantly monitoring the receive pin (rx) for a start bit. The start bit is typically signaled by a hight to low transition on the rx pin

* After the start bit has been detected, the receiver starts

sampling Rx pin at predetermined intervals shifting each sampled bit into the receive shift register.

* To determine whether the transmitted data is correct, the transmitter transmits an additional parity bit.

The UART can be configured to check for even parity or no parity at all. once data is received, the UART signals its host processor. The host processor in turn reads the byte out of the receive shift register. The receiver is now ready to receive more data.

Transmitter Operation:-

The host processor (Txing side processor) writes a byte to the transmit buffer of the UART, the transmitter sends a start bit over its transmit pin (tx), signaling the beginning of a Transmission to the remote UART. Then, the transmitter shifts out the data in its transmit buffer over its tx pin at a predetermined rate.

(Txer can also Tx its an additional parity bit)

At this point, the UART processor signals its host processor, indicating that it is ready to send more data if available.

* The transmission protocol used by UART's determines the rate at which bits are sent and received & is called baud rate. The protocol also specifies the number of bits of data and the type of parity sent during each transmission.

* The baud rate determines the speed at which data is exchanged between two serially connected UART's. The

commonly used baud rates are 2400, 4800, 9600 & 19200.

* To use a UART, we must configure its baud rate by writing to the configuration register, and then we must write data to the transmit register and/or read data from the received register.

* To use a UART, we must configure its baud rate by writing to the configuration register, and then we must write data to the transmit register and/or read data from the received register.

For ex, to configure the UART of an 8051 microcontroller we must use the following equation:

$$\boxed{\text{Baud rate} = \left(\frac{2^{smod}}{32}\right) * \frac{\text{osc freq}}{[12 * (256 - TH1)]}}$$

Where,

smod corresponds to 2-bits in a special-function register, osc freq is the frequency of the oscillator, & TH1 is an 8-bit rate register of a build-in timer.

❖ **Determine the values for smod and TH1 to generate a baud t=rate of 9600 for the 8051 baud rate equation, assuming an 11.981MHz oscillator. Remember that smod is a 2 bit and TH1 is 8-bits. There is more than one correct answer.**

Given:

Baud rate = 9,600

TH1 = 8-bit

$f = 11.981 MHz$

smod = 2 bit ie 00, 01, 10, 11.

Soln

$$\text{Baudrate} = \left(\frac{2^{smod}}{32}\right) * \frac{\text{osc freq}}{[12 * (256 - TH1)]}$$

for smod = 2 = 10 :-

$$9600 = \left(\frac{2^2}{32}\right) * \frac{11.981 \times 10^6}{[12 * (256 - TH1)]}$$

$$76,800 = \frac{11.981 \times 10^6}{12 * (256 - TH1)}$$

$$12 \times 256 - 12TH1 = \frac{11.981 \times 10^6}{76800}$$

$$3072 - 12TH1 = \frac{11.981 \times 10^6}{76800}$$

$$12TH1 = 3072 - 156.0026$$

$$12TH1 = 2,915.9974$$

$$TH1 = 242.999$$

$$\boxed{TH1 = 243 = 11110011}$$

for smod = 3 = 11 :-

$$\text{Baud rate} = \left(\frac{2^{smod}}{32}\right) * \frac{\text{osc freq}}{[12 * (256 - TH1)]}$$

$$9600 = \frac{2^3}{32} \times \frac{11.981 \times 10^6}{[12 * (256 - TH1)]}$$

$$\frac{9600 \times 32}{8} = \frac{11.981 \times 10^6}{[12 \times (256 - TH1)]}$$

$$38400 = \frac{11.981 \times 10^6}{(12 \times 256 - 12\,TH1)}$$

$$3072 - 12TH1 = \frac{11.981 \times 10^6}{38400}$$

$$3072 - 12TH1 = 312.0052$$

$$12TH1 = 3072 - 312.0052$$

$$12TH1 = 2759.994$$

$$\boxed{TH1 = 229.99}$$

$$\boxed{TH1 = 230 = 11100110}$$

PULSE WIDTH MODULATION (PWM) :-

❖ **Describe the working of PWM unit with timing diagrams. How can it be used for speed control of DC motor.**

June-11,10M

❖ **With a neat diagram, explain how the pulse width modulator works. What are the considerations in selecting the clock, the prescalar and the counter? Assuming an 8-bit up counter, calculate the count to be loaded in the 'cycle-high' register to get pulses of duty cycle 75%.**

Jan-11,10M June-06,10M

❖ **Describe the working of a PWM unit with a circuit and waveform.**

June-09,6M

❖ **Describe the working of a PWM unit with timing diagrams. How it can be used for speed control of DC motor.**

Jan-08,8M

❖ **Schematically explain how a PWM helps in controlling the speed of DC motor.**

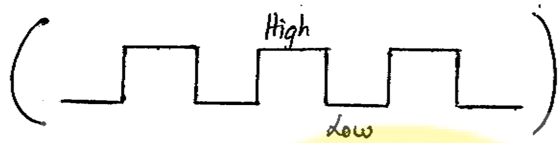
June-07,6M

❖ How PWM can be used for speed control of DC motor? Explain.

Jan-07,8M

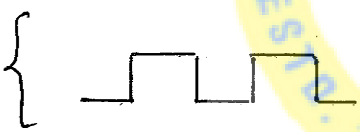
Pulse width Modulator (PWM):-

* A pulse width modulator (PWM) generates an o/p signal that repeatedly switches between high and low values



* We control the duration of the high value and of the low value by indicating the desired period (T) & the desired duty cycle (D).

* Duty cycle is defined as the ratio of ON time to the total period (ON + off) & is expressed in percentage.



$$\Rightarrow \% D = \frac{T_{ON}}{T_{ON} + T_{OFF}} \times 100$$

* There are 3 common use of PWM:

1) To generate a clock-like signal to another device
   ex:- PWM can be used to blink a light at a specific rate.

2) To control the average current or voltage input to a device.
   Ex:- A DC electric motor rotates when its input voltage is set high, with the rotation speed proportional to the input voltage level.

Suppose the revolutions per minute (rpm) equals 10 times the input voltage. To achieve a desired rpm of 125, we

We would need to set the input voltage to 1.25V, where as achieving 250 rpm would require an input voltage of 2.50V

3) To encode control commands in a single signal for use by another device.

ex:- We may control a radio-controlled car by sending pulses of different widths.

A width of 1-msec corresponds to a turn left command, a 4-msec width to turn right, and an 8-msec width to forward.

* The PWM approach makes use of the fact that a DC motor does not come to an immediate stop when its input voltage is lowered to '0', but rather its it coasts.

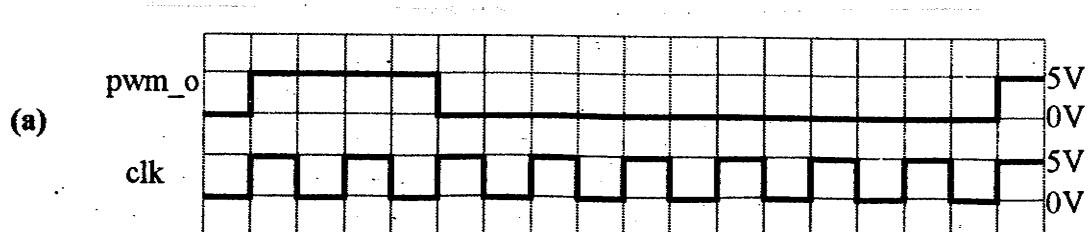
Thus the average input voltage is set to obtain the desired speed.

using a PWM, duty cycle is set to achieve the appropriate average voltage.

* Assuming the PWM's out is 5V when high and 0V when low, then.

* We can obtain an average o/p of 1.25V by setting the duty cycle to 25% ie 5V * 25% = 1.25V. This duty cycle is shown in fig 1ⓐ.

5V × 0.25 = 1.25V.



25% duty cycle – average pwm_o is 1.25V.

Fig① a.

* We can obtain an average o/p of 2.50V by setting the duty cycle to 50% as shown in fig ① b.

$0.5 \times 5V = 2.5V$

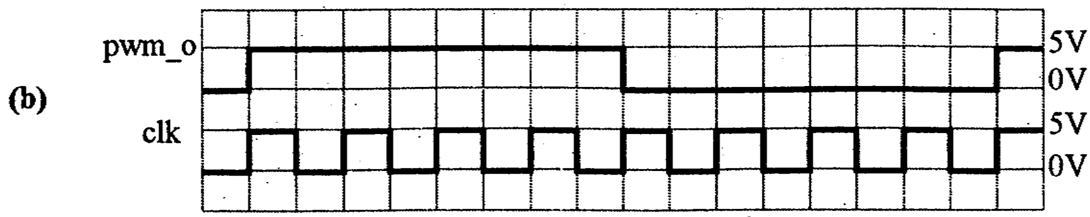


Fig ① b   50% duty cycle – average pwm_o is 2.5V.

* A duty cycle of 75% would result in average o/p of 3.75V as shown in fig 1 ©.

$0.75 \times V = 3.75V$

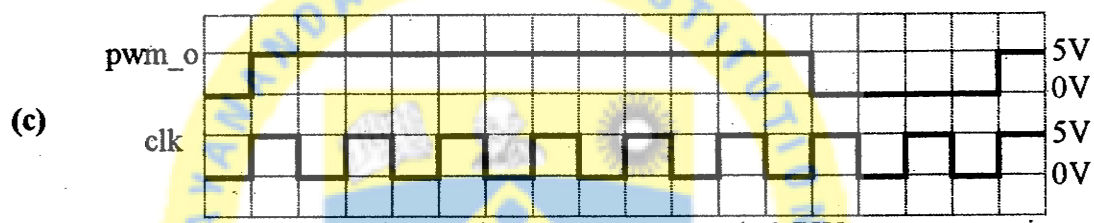


Fig ①.c   75% duty cycle – average pwm_o is 3.75V.

Controlling a DC Motor Using a PWM

* The speed of the DC Motor is proportional to the voltage applied to the motor. We must set the duty cycle of a PWM such that the average o/p voltage equals the desired voltage.

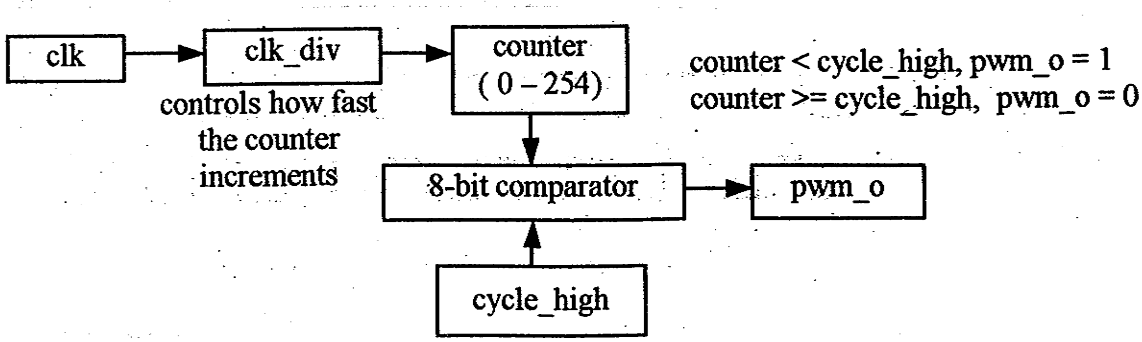


Fig 1 @ : Internal Structure of PWM

* There are two 8 bit registers called clk-div and cycle-high, an 8 bit counter and an 8-bit comparator as shown in fig 1 (a).

* Initially, the value of clk_div is loaded into the register. The clk_div register works as a clock divider. After a specified amount of time has elapsed, a pulse is sent to the counter register. This causes the counter register to increment itself. The comparator then looks at the values in the counter register & the cycle-high register.

* When the counter value < cycle_high, a 1 (+5V) is outputted. When the counter value ≥ cycle_high, a 0 (0V) is outputted.

* When the counter value reaches 254, counter is reset to 0 and the process repeats. Thus clk-div determines the PWM's period, specifying the number of cycles in the period.

* The register cycle_high determines the duty cycle, indicating how many of periods cycle output a 1. If the cycle-high is set to 255 (FFh), the o/p signal is always high resulting in a duty cycle of 100%.

  Illy if the cycle-high is set to 0(00h), the o/p signal is always low resulting in a duty cycle of 0%.

* If the value loaded to the clk_div is too low, the value outputted by the comparator oscillates too quickly. The comparator never outputs zeros long enough for the DC motor to slow down, causing the DC motor to continously run at full speed.

  Setting the value of clk_div to FFh, in this case it works best.

* The relationship between applied voltage & DC motor speed:

input voltage	% of maximum voltage applied	RPM of DC motor
0	0	0
2.5	50	4,600
3.75	75	6,900
5.0	100	9,200

* For the motor to run at 4,600 RPM, 50% duty cycle is needed. The required duty cycle is computed as

$254 \times 0.5 \simeq 127 = 7Fh$.

Thus loading 7Fh into the cycle_high register.

* ||^{ly} to run motor at 6,900RPM, 75% duty cycle is needed. The required duty cycle is computed as

$254 \times 0.75 = 191 = BFh$

Thus loading BFh into the cycle_high register

* The PWM does not provide enough current to run the DC motor. Thus an NPN transistor is used to drive the DC motor as shown below.

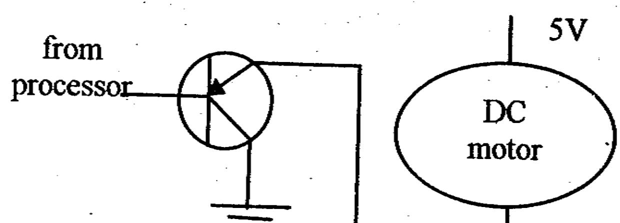


Fig 2@: Connection to DC motor

The pseudo code is :

```
void main(void) {

    /* controls period */
    PWMP = 0xff;
    /* controls duty cycle */
    PWM1 = 0x7f;
    while(1){};
}
```

* The name of the clk_div register is PWMP and cycle_high is PWM1.

STEPPER MOTOR CONTROLLER :-

❖ **Explain how a stepper motor is controlled using driver. Gice relevant hardware and software details.**

Jan-07,8M

* A stepper motor is an electric motor that rotates fixed number of degrees whenever we apply a 'step' signal.

* Stepper motor can rotate 1.8° (Full step) or 0.9° (half step) per step. If the motor rotates 1.8° per step, then to move 360°, the number of steps required is 200° ie (1.8×200 steps = 360°)

* Internally, a stepper motor typically has four coils. To rotate the motor one step, we pass current through one or two of the coils. Thus rotating the motor 360° requires current to the coils in a specified sequence. Applying the sequence is reverse causes reversed rotation.

Stepper motor can be controlled in 2 ways :

1) Using a stepper motor driver
2) Controlling a stepper motor directly

Application of Stepper motor :-

1) Dist drivers
2) Printers
3) Photocopy
4) Fax machines
5) Robots
6) Camcorders
7) VCR's.

Stepper Motor Control using Driver :-

* Controlling a stepper motor requires applying a series of voltages to the four coils of the stepper motor. The coils are energised one or two at a time causing the motor to rotate one step.

* In this example, we are using a 9-volts, 2-phase bipolar stepper motor. The table indicating the input sequence required to rotate the motor. The entire sequence must be applied to get the motor to rotate 7.5 degrees.

To rotate the motor in the opposite direction, we simply apply the sequence in reverse order.

sequence	A	B	A'	B'
1	+	+	-	-
2	-	+	+	-
3	-	-	+	+
4	+	-	-	+
5	+	+	-	-

* We can use an 8051 μc and MC3479P chip to control the stepper motor. We need only worry about setting the direction on the clockwise / counter clock wise pin (CW̄/CCW) and pulsing the clock pin (clk) on the stepper motor driver chip using the 8051 microcontroller.

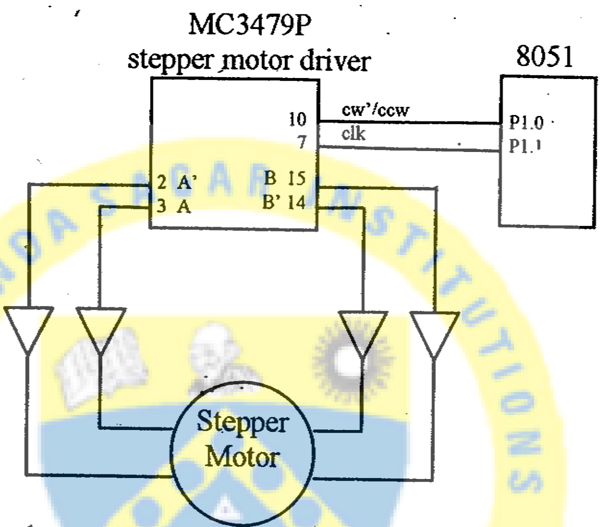


Fig 1 (a): Controlling a stepper motor using a driver

```
/* main.c */
sbit clk = P1^1;
sbit cw = P1^0;
void delay (void)
{
 int i, j;
 for (i=0 ; i<1000; i++)
  for (j=0 ; j<50 ; j++)
   i = i + 0;
}
void main (void)
/* turn the motor forward */
cw = 0;      /* set direction */
```

```
clk=0; /* pulse clock */
delay();
clk=1;

/* turn the motor backwards */
CW=1;       /* set direction */
clk=0;      /* pulse clock */
delay=();
clk=1;
```

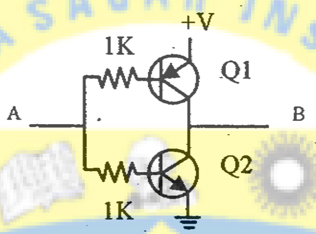


Fig ①b. Buffer

* The o/p pins on the stepper motor driver do not provide enough current to drive the stepper motor. To amplify the current, a buffer is needed and is show in fig 1ⓑ.

* Q_1 is an PNP transistor & Q_2 is an NPN transistor. 'A' is connected to the 8051 microcontroller and 'B' is connected to the stepper motor.

CONTROLLING STEPPER MOTOR DIRECTLY :-

(Without Using DRIVER)

In this example, the stepper motor driver is eliminated. The stepper motor is connected directly to the 8051 μc as shown in fig 1ⓐ.

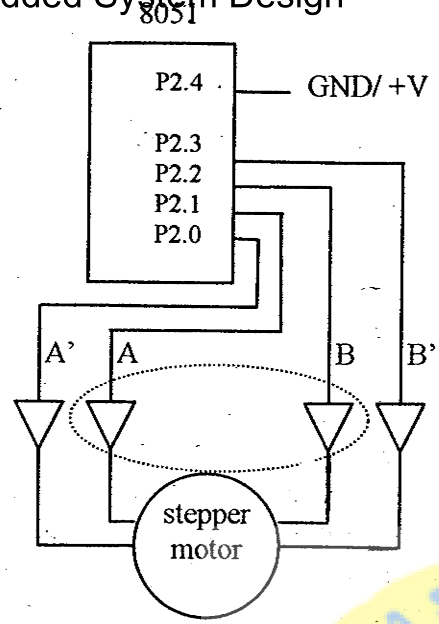


Fig ①a : Controlling a stepper motor directly

* The direction of the stepper motor is controlled manually.

If P2.4 is grounded, the motor rotates counter clockwise, otherwise the motor rotates clockwise

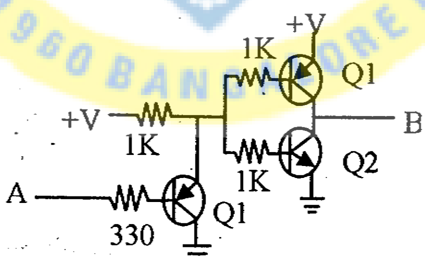


Fig ① b . Buffer.

* The 8051 ports are unable to directly supply the current needed to drive the motor. To amplify the current, a buffer is needed and is shown is fig 1⑥.

* Q_1 are PNP transistor and Q_2 is an NPN transistor. A is connected to the 8051 microcontroller and B is

the stepper motor.

The sample code to run the stepper motor is shown below

```
sbit notA = P2^0;
sbit is A = P2^1;
sbit notB = P2^2;
sbit is B = P2^3;
sbit dir = P2^4;
void delay ( )
{
    int a, b;
    for (a=0; a<5000; a++)
    for (b=0; b<1000; b++);
}
void move (int dir, int steps)
{
    int Y, Z;
    if (dir ==1)
    { for (Y=0; Y<=steps; Y++)
      { for (Z=0; Z<=19; Z+=4)
        { is A = lookup[Z];
          is B = lookup[Z+1];
          not A = lookup[Z+2];
          not B = lookup[Z+3];
          delay( );
        }
      }
    }
    if (dir==0)
    {
      for (Y=0; Y<=step; Y++)
```

```
{
  for (z = 19; z >= 0; z -= 4)
  {
      isA = lookup[z];
      isB = lookup[z-1];
      notA = lookup[z-2];
      notB = lookup[z-3];
      delay();
   }
  }
 }
}

int lookup[20] = {1,1,0,0,0,1,1,0,0,0,
                  1,1,1,0,0,1,1,1,0,0};

void main( )
{
 while (1)
 { /* move forward 15 degrees */
   move (1,2);
  /* move backward 7.5 degrees */
  move (0,1);
 }
}
```

Note :- int lookup[20] = { 1 , 1 , 0 , 0 ,

[z] [z+1] [z+2] [z+3]

0 , 1 , 1 , 0

0 , 0 , 1 , 1

1 , 0 , 0 , 1

1 , 1 , 0 , 0 };

LIQUID CRYSTAL DISPLAY (LCD) :-

* A LCD is a low-cost, low-power device capable of displaying text and images. LCD's are extremely common in embeded systems, Since such system often do not have video monitors like those that come standard with desktop systems.

LCD's can be found in numerous common device like watches, fax, copy machines and calculators.

{ Basic principle :-

The basic principle of one type of LCD, a reflective LCD, works as follows. First, incoming light passes through a polarizing plate. Next, that polarized light encounters liquid crystal material.

If we excite a region of this material, we cause the material molecules to align, which in turn causes the polarized light to pass through the material. otherwise, the light doesnot pass through.

Finally, light that passed through hits a mirror and reflects back, so the excited region appears to light up.

Another type of LCD, an absorption LCD works similarly, but uses a black surface instead of a mirror. The surface below the excited region absorbs light, thus appearing darker than the other regions.

A dot-matrix LCD consists of a matrix of dots that can display alphanumeric characters (letters and digits) as well as other symbols. A common dot-matrix LCD has five columns and eight rows of dots for one character which converts input data into the appropriate electrical signals necessary to excite the appropriate LCD dots.}

* Each type of LCD may be able to display multiple characters. Each characters may be displayed in normal or inverted fashion.

The LCD may permit a character to be blinking or may permit display of a cursor indicating the 'current' character (blinking underscore). Such functionality would be difficult for us to implement using software.

Thus, we use an LCD controller to provide us with a simple interface to an LCD having 8-data inputs (DB_0-DB_7) and one enable input.

* To send a byte to the LCD, we provide a value to the eight inputs and pulse the enable. This byte may be a control word, which instructs the LCD controller to initialize the LCD, clear the display, select the positions of the cursor, brighten the display.

* Alternately, this byte may be a data word, such as an ASCII characters, instructing the LCD to display the character at the currently-selected display position.

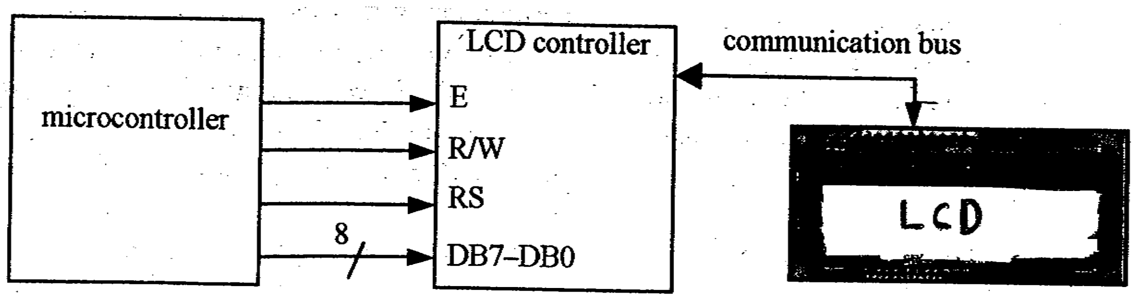


Fig 8 a. LCD initialization Components

Codes	
I/D = 1 cursor moves left	DL = 1 8-bit
I/D = 0 cursor moves right	DL = 0 4-bit
S = 1 with display shift	N = 1 2 rows
S/C = 1 display shift	N = 0 1 row
S/C = 0 cursor movement	F = 1 5 × 10 dots
R/L = 1 shift to right	F = 0 5 × 7 dots
R/L = 0 shift to left	

```
void WriteChar(char c){
    /* indicate data being sent */
    RS = 1;
    /* send data to LCD */
    DATA_BUS = c;
    /* toggle LCD with delay */
    EnableLCD(45);
}
```

* In this example, a microprocessor is connected to an LCD controller, which is turn is connected to an LCD as shown in fig 1ⓐ. The LCD controller receives control words from the microcontroller, it decodes the control words and performs the corresponding action on the LCD
* once the initialization sequence is done, we can send control words or send actual data to be displayed.
* When RS is set to low to indicate that the data sent is control word.
* When RS is high, this indicates that the data sent over the communications bus corresponds to a character that is to be displayed.
* Everytime data is sent, whether it is a control word or data, the enable bit E must be toggled (ie ⎍)
* By using initialization code, the LCD has been set with an 8-bit interface. In addition, the display has been cleared, the cursor is in the home position, and the cursor moves to the right as data is displayed. The LCD is now ready to be written to.
* In order to write data, we set RS = 1. The actual data we want to write is on $DB_0 - DB_7$. The write char functions accepts a character which will be sent to the

LCD controller to display on the LCD.

The Enabled LCD function toggles the enables bit and acts as a delay so that the command can be processed and executed.

KEYPAD CONTROLLERS :-

* A Keypad consist of a set of buttons that may be pressed to provide input to an embeded system. Again, keypads are extremely common in embeded systems, since such systems may lock the keyboard that comes standard with desktop systems.

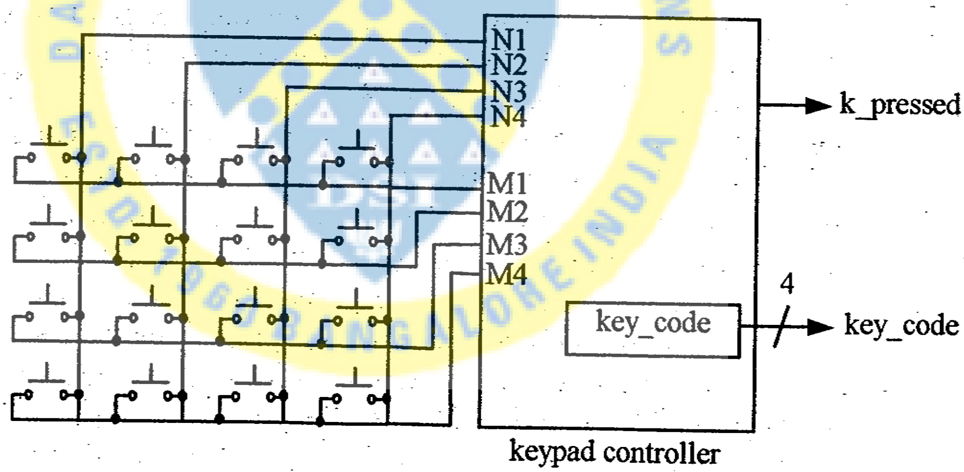


Figure (1) Internal keypad structure, with $N = 4$ and $M = 4$.

Fig (1) shows a simple keypad having buttons arranged in an N-coloumn by M-row grid.

The device has N outputs, each output corresponding to a coloumn, and another 'M' outputs, each output correspon-ds to a row.

When we press a button, one coloumn output and one row output go high, uniquely identifying the pressed

...ction Such a keypad from software, we must scan the coloumn & row outputs. The scanning may be performed by a keypad controller. such a device decodes rather than controls, but we will call it a "controller".

* Fig ① shows the controller, which scans the coloumn and row outputs of the keypad. When the controller detects a button press, it stores a code corresponding to that button into a register, key-code, and sets an output high, k-pressed, indicating that a button has been pressed.

* The software may poll this o/p every 100 milliseconds or so and read the register when the o/p is high. Alternatively, this output can generate an interrupt on our general-purpose processor, eliminating the need for polling.

Analog to digital converter (ADC) :-

June-07,8M

❖ **Highlight the advantages of using data in digital form over its analog form. Explain the working of successive approximation types of analog to digital converter.**

* An analog-to-digital converter (ADC, A/D or A2D) converts an analog signal to a digital signal and a digital-to-analog converter (DAC, D/A or D2A) does the opposite.

* Analog refers to a continously valued signal, such as temperature or speed.

* Digital refers to discretely valued signals, such as integers. and these signals are encoded in binary.

For example consider an analog input signal whose range from 0 to 7.5 volts. We want to represent each possible voltage in this range using a 4 bit binary numbers. The 0000 would be the most obivious encoding for 0V and 1111 for 7.5V. The encodings between 0000 and 1111 would den be evenly distributed to the range between 0 and 7.5V. as shown in fig 1(a).

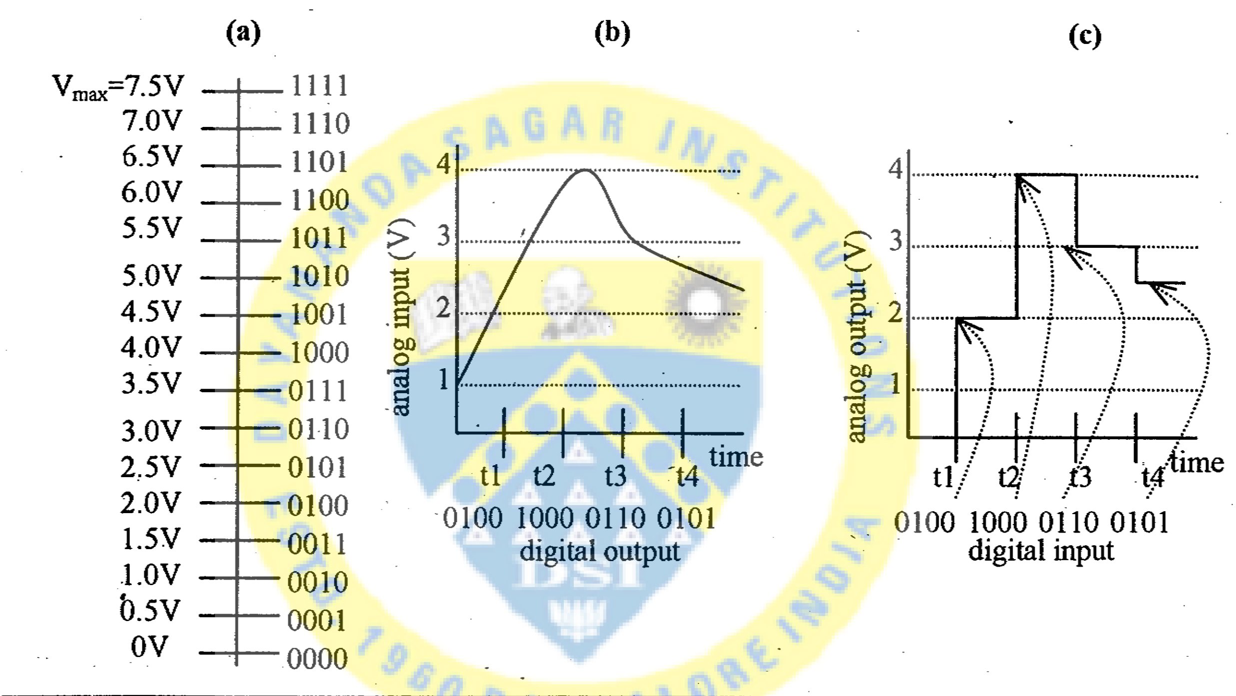


Figure 1 : Conversion: (a) proportionality, (b) analog-to-digital, (c) digital-to-analog.

Fig 1(b) ranging from 1V upto 4V and then down to just over 2V. The digital encoding of this signal, sampled at times t_1, t_2, t_3 and t_4, into four bits.

* We can compute the digital values from the analog values, and vice-versa using the following ratio:

$$\boxed{\frac{e}{V_{max}} = \frac{d}{(2^n - 1)}}$$

Where V_{max} is the maximum voltage that the analog signal can Assume n is the number of bits available for the digital encoding

d is the present digital encoding and

e is the present analog voltage.

For example: Suppose Vmax is 7.5V, Assume analog Voltage e = 3V, WKT it is a 4-bit converter ∴ n = 4

Then $\frac{3V}{7.5V} = \frac{d}{2^4-1}$

$\frac{3 \times 15}{7.5V} = d$

∴ d = 6, or 0110

* The resolution of ADC or DAC is $\frac{V_{max}}{2^n-1}$. This represents the number of volts between "successive digital encoding"

Resolution = $\frac{V_{max}}{(2^n-1)} = \frac{7.5V}{2^4-1} = 0.5V.$

∴ Resolution is 0.5V between successive encoding

1) Given an analog input signal whose voltage ranges from 0 to 15v and an 8-bit digital encoding is used. Calculate the correct encoding for 5v and then trace the successive approximation approach to find the correct encoding.

June-07,10M

Soln 1: WKT the encoding should be

$$\frac{e}{V_{max}} = \frac{d}{2^n - 1}$$

When $V_{max} = V_{max} - V_{min}$

$$\frac{5}{15} = \frac{d}{2^8 - 1}$$

$$\frac{5}{15} \times (2^8 - 1) = d$$

$\boxed{d = 85}$ ← correct encoding.

$$\text{Resolution} = \frac{V_{max}}{2^n - 1} = \frac{15V}{2^8 - 1} = \frac{15}{255}$$

$\boxed{\text{Resolution} = 0.058823V}$

* Applying the successive approximation method we start by finding the halfway point between the maximum and minimum voltages.

Where $V_{max} = 15V$ & $V_{min} = 0V$

$$\therefore e' = \frac{V_{max} + V_{min}}{2} = \frac{15V + 0V}{2} = \overset{(e')}{7.5V}$$

* Since the above voltage is higher than the input voltage (5V). We insert a zero into the highest bit shown below.

∴

0	0	0	0	0	0	0	0

* Now $V_{max} = 7.5V$ and $V_{min} = 0V$

$$\therefore e' = \frac{V_{max} + V_{min}}{2} = \frac{7.5V + 0V}{2} = \overset{(e')}{3.75V}$$

Since $e' < e$ ie $3.75V < 5V$

∴ We insert a one into the next MSB as shown below

0	1	0	0	0	0	0	0

* Now $V_{max} = 7.5V$ & $V_{min} = 3.75V$

$$\therefore e' = \frac{V_{max}+V_{min}}{2} = \frac{7.5V + 3.5V}{2} = 5.63V$$

Since $e' > e$ ie $5.63V > 5V$

∴ We insert a zero into the next MSB as shown below.

0	1	0	0	0	0	0	0

* Now $V_{max} = 5.63V$ & $V_{min} = 3.75V$

$$\therefore e' = \frac{V_{max}+V_{min}}{2} = \frac{5.63V + 3.75V}{2} = 4.69V.$$

Since $e' < e$, we insert a one into next MSB as shown.

0	1	0	1	0	0	0	0

* Now $V_{max} = 5.63V$ & $V_{min} = 4.69V$

$$\therefore e' = \frac{V_{max}+V_{min}}{2} = \frac{5.63+4.69}{2} = 5.16V$$

Since $e' > e$, we insert a zero into next MSB as shown below.

0	1	0	1	0	1	0	0

* Now $V_{max} = 5.16V$ & $V_{min} = 4.93V$

$$\therefore e' = \frac{V_{max}+V_{min}}{2} = \frac{5.16V + 4.93V}{2} = 5.05V$$

we insert a zero into the next MSB Bit as Shown Below.

0	1	0	1	0	1	0	0

* Now $V_{max} = 5.05V$ & $V_{min} = 4.93V$

$$\therefore e^l = \frac{V_{max}+V_{min}}{2} = \frac{5.05V + 4.93V}{2} = 4.99V$$

Since $e^l < e$, we insert a one into LSB as shown below.

0	1	0	1	0	1	0	1

$\therefore$ Resulting value 01010101 = 85d

2) Given an analog input signal whose voltage ranges from 0 to 5v and an 8-bit digital encoding. Calculate the correct encoding for 3.5v and then trace the successive approximation approach to find the correct encoding.

Jan-11,8M

OR

Determine the resolution of an 8-bit ADC with an analog input voltage range of 0 to 5V. Determine the digital encoding for 3.5 volts using a formula and trace the steps using successive approximation technique. Write successive approximation technique. Write the steps for this technique in the form of a table. With necessary columns/informations.

Jan-08,8M

Soln 2: Given $V_{max} = 5V$, $V_{min} = 0V$ $e = 3.5V$, $n = 8$-bit

WKT, The encoding should be

$$\frac{e}{V_{max}} = \frac{d}{(2^n - 1)}$$

$$\frac{3.5V}{5V} = \frac{d}{2^8 - 1}$$

$d = 178.5$

$d \approx 179 \rightarrow$ correct encoding

$179 = 10110011$

Using successive approximation approach.

1) $V_{max} = 5V$, $V_{min} = 0V$ $e = 3.5V$

$$e' = \frac{V_{max} + V_{min}}{2} = \frac{5V - 0V}{2} = 2.5V$$

ii) Now $e' < e$ thus set MSB Bit $\underline{1}0000000$

2) $V_{max} = 5V$ $V_{min} = 2.5V$

$$e' = \frac{V_{max} + V_{min}}{2} = \frac{5V + 2.5V}{2} = 3.75V$$

Now $e' > e$, thus clear the bit $1\underline{0}000000$

3) $V_{max} = 3.75V$ $V_{min} = 2.5V$

$$e' = \frac{3.75V + 2.5V}{2} = 3.125V$$

Now $e' < e$ ie $3.125 < 3.5V$, thus set the bit

$10\underline{1}00000$

4) $V_{max} = 3.75V$, $V_{min} = 3.125V$

$$e' = \frac{3.75 + 3.125V}{2} = 3.437V$$

Now $e' < e$ ie $3.437V < 3.75V$, thus set the bit

$101\underline{1}0000$

5) $V_{max} = 3.75V$, $V_{min} = 3.43\ V$

$$e' = \frac{3.75V + 3.43V}{2} = 3.594V$$

Now $e' > e$, thus clear the bit 1011<u>0</u>000.

6) $V_{max} = 3.594V$, $V_{min} = 3.437V$

$$e' = \frac{3.594 + 3.437}{2} = 3.515V$$

$e' > e$ thus clear the bit 10110<u>0</u>00

7) $V_{max} = 3.515$ $V_{min} = 3.437$

$$e' = \frac{3.515 + 3.437}{2} = 3.476$$

$e' < e$, thus set the bit 101100<u>1</u>0

8) $V_{max} = 3.515$ $V_{min} = 3.476$

$$e' = \frac{3.515 + 3.476}{2} = 3.4955$$

$e' < e$, thus set the bit 1011001<u>1</u>.

$\therefore$ <u>Resulting value 10110011 = 179 d</u>.

❖ **Extend the ratio and resolution equations of analog to digital conversion to any voltage range between Vmin to Vmax rather than 0 to Vmax.**

* Ratio : $\dfrac{(e - V_{min})}{(V_{max} - V_{min})} = \dfrac{d}{(2^n - 1)}$

* Resolution : $\dfrac{(V_{max} - V_{min})}{(2^n - 1)}$

3) The analog input ranges for an 8-bit ADC is -5V to +5V. Determine the resolution of this ADC and also the digital output in binary when the input is 3.5V using formula. Also trace the successive approximation steps for verification. Write it in a tabular form with necessary columns.

June-09,8M

Given :- $V_{min} = -5V$, $V_{max} = +5V$, $n = 8$-bit & $e = 3.5V$

Soln :-

$$\frac{e - V_{min}}{(V_{max} - V_{min})} = \frac{d}{(2^n - 1)}$$

$$\frac{3.5 - (-5V)}{5V - (-5V)} = \frac{d}{(2^8 - 1)}$$

$$\frac{(3.5 + 5V)}{10V} = \frac{d}{(2^8 - 1)}$$

$$216.75 = d$$

$$\therefore \boxed{d = 217} \Rightarrow \boxed{11011001}$$

Using successive approximation approach :-

1) $V_{max} = 5V$, $V_{min} = -5V$, $e = 3.5V$

$$e' = \frac{V_{max} + V_{min}}{2} = \frac{5 - 5}{2} = 0V$$

Now $e' < e$, thus set the bit $\underline{1}000\ 0000$

2) $V_{max} = 5V$, $V_{min} = 0V$

$e' = \frac{5 + 0}{2} = 2.5V$ Now $e' < e$, thus set the bit

$1\underline{1}000000$.

3) $V_{max} = 5V$ $V_{min} = 2.5V$

$e' = \frac{5 + 2.5V}{2} = 3.75V$. Now $e' < e$, thus clear the bit

11 0 00000

4) $V_{max} = 3.75V$, $V_{min} = 2.5V$

$$e' = \frac{3.75+2.5V}{2} = 3.125V.$$

Now e' < e, set the bit 11010000.

5) $V_{max} = 3.75V$, $V_{min} = 3.125V$

$$e' = \frac{3.75V + 3.125V}{2} = 3.4375V.$$

Now e' < e, thus set the bit 1101 1000

6) $V_{max} = 3.75V$, $V_{min} = 3.4375V$

$$e' = \frac{3.75 + 3.4375}{2} = 3.59375V$$

Now e' > e. Thus clear the bit 11011000

7) $V_{max} = 3.59375V$ $V_{min} = 3.4375V$

$$e' = \frac{3.59375 + 3.4375}{2} = 3.515625V.$$

Now e' > e, Thus clear the bit 11011000.

8) $V_{max} = 3.515625V$, $V_{min} = 3.4375V$

$$e' = \frac{3.515625 + 3.4375}{2} = 3.4765625V$$

Now e' < e. Thus set the bit 11011001

∴ Resulting value 11011001 = 217 d.

4) Assume 8-bit encoding of input voltage in the range -5V to +5V. Calculate the encoding for 1.2V and trace the successive approximation approach to find the correct encoding. What is the resolution of the conversion? Extend the ratio and resolution equations to any voltage in the range V_{min} to V_{max}.

Given :- $V_{max} = 5V$, $V_{min} = -5V$, $n = 8$ bit, $e = 1.2V$

Soln :-

$$\frac{e - (V_{min})}{V_{max} - V_{min}} = \frac{d}{2^n - 1}$$

$$\frac{1.2V - (-5V)}{5 - (-5V)} = \frac{d}{2^8 - 1}$$

$$\frac{1.2V + 5V}{10V} = \frac{d}{255}$$

$$d = 158.1$$

$\therefore$ Correct encoding $\boxed{d = 158 = 10011110}$.

1) $V_{max} = 5V$, $V_{min} = -5V$

$$e' = \frac{5-5}{2} = 0V$$

Now $e' < e$. Thus set the bit 10000000.

2) $V_{max} = 5V$ $V_{min} = 0V$

$$e' = \frac{5+0}{2} = 2.5V$$

Now $e' > e$. Thus clear the bit 10000000

3) $V_{max} = 2.5V$, $V_{min} = 0V$

$$e' = \frac{2.5+0}{2} = 1.25V$$

Now $e' > e$. Thus clear the bit 1000 0000

4) $V_{max} = 1.25V$, $V_{min} = 0$

$$e' = \frac{1.25+0}{2} = 0.625V$$

Now $e' < e$. Thus set the bit 10010000.

5) $V_{max} = 1.25V$, $V_{min} = 0.625V$

$$e' = \frac{1.25 + 0.625}{2} = 0.9375V$$

Now $e' < e$. Thus set the bit 10011000

6) $V_{max} = 1.25V$, $V_{min} = 0.9375V$

$$e' = \frac{1.25 + 0.9375V}{2} = 1.09375V$$

Now $e' < e$. Thus set the bit 10011100

7) $V_{max} = 1.25V$, $V_{min} = 1.09375V$

$$e' = \frac{1.25 + 1.09375}{2} = 1.171$$

Now $e' < e$. Thus set the bit 10011110

8) $V_{max} = 1.25V$, $V_{min} = 1.171$

$$e' = \frac{1.25 + 1.771}{2} = 1.2109V.$$

Now $e' > e$. Thus clear the bit 10011110

∴ Resulting value 10011110 = 158 d

5) In successive approximation ADC, calculate the correct encoding of 5V given an analog signal whose voltage ranges from 0 to 5V and an 8-bit digital encoding. Also determine the resolution of this ADC.

Jan-07,8M

Given :- $V_{max} = 5V$, $V_{min} = 0V$ $e = 5V$ $n = 8$-bits.

Soln :-
$$\frac{e}{(V_{max} - V_{min})} = \frac{d}{(2^n - 1)}$$

$$\frac{5}{(5-0)} = \frac{d}{(2^8-1)}$$

$$255 = d$$

$\therefore$ correct encoding $\boxed{d = 255 = 11111111}$

Using successive approximation Approach.

1) $V_{max} = 5V$, $V_{min} = 0V$

$$e' = \frac{V_{max} + V_{min}}{2} = \frac{5+0}{2} = 2.5V$$

Now $e' < e$. Thus, set the bit $\underline{1}000\ 0000$

2) $V_{max} = 5V$, $V_{min} = 2.5V$

$$e' = \frac{5V + 2.5}{2} = 3.75V$$

Now $e' < e$. Thus, set the bit $1\underline{1}00\ 0000$

3) $V_{max} = 5V$, $V_{min} = 3.75V$

$$e' = \frac{5V + 3.75V}{2} = 4.375V$$

Now $e' < e$. Thus, set the bit $11\underline{1}0\ 0000$

4) $V_{max} = 5V$, $V_{min} = 4.375V$

$$e' = \frac{5V + 4.375V}{2} = 4.6875V$$

Now $e' < e$. Thus, set the bit $111\underline{1}\ 0000$

5) $V_{max} = 5V$, $V_{min} = 4.6875V$

$$e' = \frac{5V + 4.6875V}{2} = 4.84375V$$

Now $e' < e$. Thus set the bit $1111\ \underline{1}000$.

6) $V_{max} = 5V$, $V_{min} = 4.84375V$

$$\frac{5V + 4.84375}{2} = 4.921875V$$

Now e<e. Thus, set the bit 11111100

7) $V_{max} = 5V$, V_{min} 4.921875V

$$e' = \frac{5V + 0.921875V}{2} = 4.9609375V$$

Now e'<e. Thus, set the bit 11111110

8) $V_{max} = 5V$, $V_{min} = 4.9609375V$

$$e' = \frac{5V + 4.9609375V}{2} = 4.98046875V$$

Now e'<e. Thus, set the bit 11111111

∴ Resulting value 11111111 = 255d.

---

6) Given an analog output signal whose voltage should range from 0 to 10V and an 8-bit digital encoding provide the encodings for the following desired voltages.

a) 0v   b) 1V   c) 5.33v   d) 10V

e) What is the resolution of our conversion?

Given :- $n=8$, $2^n - 1 = 2^8 - 1 = 255$ $V_{max} = 10V$, $V_{min} = 0V$

a) $e = 0V$,

$$\frac{e}{(V_{max} - V_{min})} = \frac{d}{(2^8 - 1)}$$

$$\frac{0}{10V} = \frac{d}{255}$$

$$\boxed{d = 0 = 00000000}.$$

b) $e = 1V$

$$\frac{1V}{10V} = \frac{d}{255}$$

$d = 25.5$ $\boxed{d = 25 \quad 00011001}$

c) $e = 5.33V$

$$\frac{5.33V}{10} = \frac{d}{255}$$

$$d = 135.9$$

$$\boxed{d = 136 = 10001000}$$

d) $e = 10V$

$$\frac{10}{10} = \frac{d}{255}$$

$$\boxed{d = 255 = 11111111}$$

e) $$\text{Resolution} = \frac{V_{max} - V_{min}}{2^n - 1} = \frac{10V - 0V}{2^8 - 1}$$

$$\boxed{\text{Resolution} = 0.039V}$$