T.Y. Diploma : Sem. VI
[ET/EJ/EN/EX]

Embedded Systems

Time: 3 Hrs.] Prelim Question Paper Solution **[Marks : 100**

Q.1 Attempt any FIVE of the following : [20]

Q.1(a) Compare assembly language and embedded C language. [4]

(A)

Parameter	Assembly	C
Ease of writing Program	Complex	Easy
Time for coding	More	Less
Time for execution	Less	More
Debugging	Tedious	Simple
Updating	Tedious	Simple
Hex file size	Small	More
Memory required	Less	More
Compatibility	Changes with MC	Portable

Q.1 (b) Describe the CAN protocol with neat diagram. [4]

(A) CAN is a serial bus for interconnecting a central network.

- It is used in automobiles where number of devices and controllers are located and distributed in it e.g.: breaks, engine, power, lamp, temperature controller, AC, gate, dash board, display, cruise control, etc.
- It is also used in medical and industrial plant.
- CAN bus is a standard bus in distributed network.
- CAN is implemented with the help of microcontroller.
- It is bidirectional serial line which is operated at 1 mbps.
- It employs a twisted pair connection of 120Ω line impedance at each controller port.
- Length of pair is 40 meter.
- CAN serial line is pulled to logic 1 (+4.5V to 12.3V) through pull up register.
- In ideal state it's in logic 1 also called as recessive state.
- Each node has buffer gate between i/p and CAN serial line.
- A node gets i/p at any instant from the line after sensing that instant line.
- It is pulled down to logic 0 called as dominant state.
- Each node has a current driver between o/p pin and serial line node sends the data bit as a data frame and data frame start through logic 1.
- CAN has a field for bus arbitration.
- Control bits for address and data length, data bits, CRC checkers, acknowledgement and ending bits.
- A CAN bus line available between CAN controller and host node.
- The bus arbitration methods are CSMA (Carrier Sense Multiple Access) and AMP (Arbitration on Message Priority).

Serial IO CAN bus
Serial bus controller for CAN in a
Microcontroller

Serial CAN bus

CAN Device controller Processor of system B	CAN Device controller Processor of system C	CAN Device controller Processor of system D	CAN Device controller Processor of system E

Q.1 (c) List the Parallel Communication Protocol and describe any one. [4]

(A)
1. Industry Standard Architecture (ISA)
2. Peripheral Component Interface (PCI)
3. PCI – X

- PCI: PCI stands for Peripheral Component Interconnect/Interface bus.
- It is popular for higher bandwidth processor independent which can function as peripheral bus.
- It is introduced by Intel in 90's.
- It is 32 bit local bus and extended up to 64 bit by processor it requires.
- It has high speed I/O subsystem performance.
- The PCI is designed to meet economically the i/o requirement of modern system.
- It supports ten i/o devices and provides 3 types of synchronous parallel interface.
- It has two versions:
  32 bit (33 MHz)
  64 bit (66 MHz)
- The data transfer rate for synchronous is 132 mbps and for asynchronous it is 528 mbps.
- The PCI driver can access hardware automatically or by programmer can assign address.
- The automatic detection and assignment of addresses of various devices simplifies the addition and removal of the system peripheral.
- PCI is designed to support variety of microprocessor best configuration including single and multi processing system.

OR

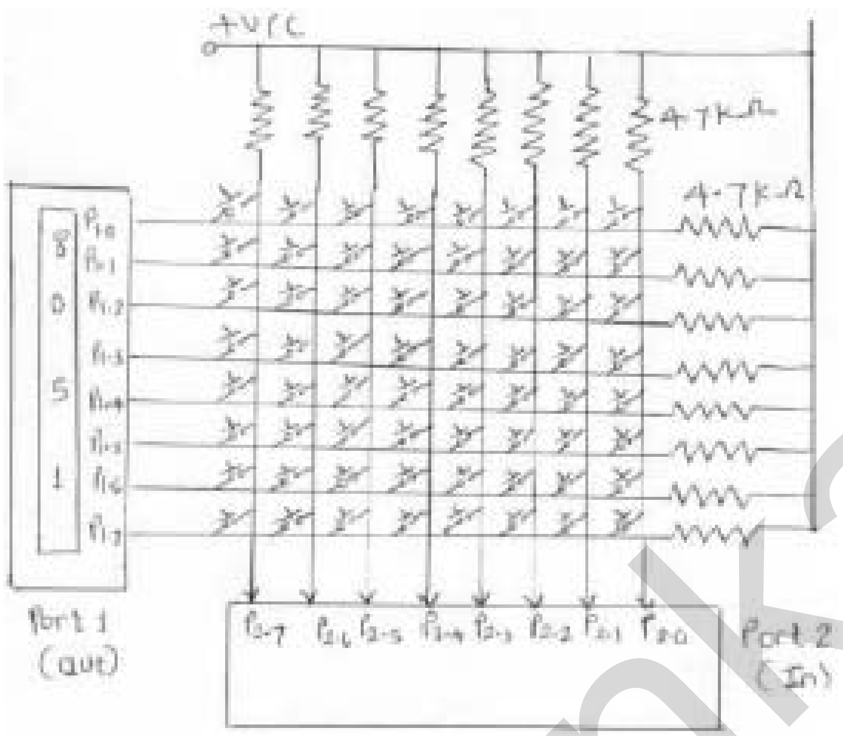
PCI-X
- It is an extension of PCI bus and supports 64 bit, 100MHz transfer.
- PCI-X is revised to double the maximum clock speed to improve the data exchange transfer between processor and peripherals.
- The data exchange rate is 1.06 gbps.

Q.1 (d) Differentiate RTOS with desktop operating system. (Any four points). [4]

(A)

RTOS	Desktop OS
Predictable schedule.	Unpredictable but efficient schedule.
More deterministic.	Less deterministic.
Ability to scale up and down to meeting application needs (scalable).	Less scalability.
Fast context switching.	Higher context switch latency.
Uses priority preemptive scheduling.	Uses round robin way of scheduling.
Support for diskless embedded systems by allowing executables to boot and run from RAM or RAM.	Has to be booted from disk.
Reduced memory requirements.	More memory requirements.
Faster performance.	Slower.

Q.1 (e) Draw the interfacing of 8 × 8 matrix keyboard with 89C51 microcontroller. **[4]**
(A)



Q.1 (f) State any 4 features of IDE and ICE. **[4]**
(A) Features of IDE :

1. Facility for defining a processor family as well as defining its version.
2. Includes source code engineering tools that incorporate the editor, compiler for C and embedded C++.
3. It has a source code engineering tools such as Assembler, linker, locator, logic analyser, stethoscope, and help to user
4. Optimizes the use of memory
5. It provides the multiuser environment
6. Design process division in to a number of sub parts.
7. Simulation of hardware such as emulator, peripherals and I/O devices on a host PC.
8. Supports break points
9. Debugging facility
10. Verification of performance of the target system.
11. Includes a logic analyzer for up to 256 or 512 transactions on the address and data buses after triggering.

Features of ICE :

1. It is used to debug a target system without using actual processor.
2. It allows the program to be tested on actual hardware system.
3. It provides the facilities like break points or single stepping.
4. ICE connected between the host computer and board to be tested.

Q.1 (g) Describe program down loading tool ISP / IAP. **[4]**
(A) In-System Programming (ISP) :

- In-System Programming (ISP) is a method where a programmable device is programmed after the device is soldered in target hardware.
- ISP reduces the extra handling step required in the manufacturing process to program the devices on an external programmer before placing them on target hardware.
- ISP supports the functions of some programmable logic devices, microcontrollers, and other programmable electronic devices to be programmed during installed in a complete system.

- The advantage of ISP is that the manufacturers of embedded devices can integrate programming and testing into a production stage instead of a separate programming stage before assembling the system.
- Basically, the device supporting ISP have an internal circuitry to generate required programming voltage from the system supply voltage, and communicate with the programmer using a serial protocol.

In-Application Programming (IAP) :
- In-Application Programming (IAP) is a method where a programmable device is programmed after the device is soldered in target hardware while the application code is running.
- Implementation of applications with IAP is possible because embedded system can be re-programmed remotely without a service personal to be actually present.
- Basically, IAP is used with external Flash memory which a separate component mounted on the circuit board.
- Bit in some case, the additional program memory available from which the microcontroller can execute code, while the Flash memory is re-programmed.
- IAP is only possible if the microcontroller has on-chip Flash memory.

Q.2 Attempt any FOUR of the following : [16]
Q.2(a) State any four design metrics of an embedded system. [4]
(A) Processor power
- Selection of the processor is based on the amount of processing power to get the job done and the register width required.
- 8 bit, 16 bit, 32 bit and 64 bit microcontrollers are provided.
- Processing power is different for different microcontrollers.
- High clock, speed and addressing capable microcontrollers are available.
- Very powerful DSPS are available for real time analysis of audio and video signals.

Memory
- Designer has to make an estimate of the memory requirement and must make provision for expansion.
- In a system, there are different types of memories : RAM, ROM, EPROM, PROM, etc.
- Secondary storage devices like HDD can be embedded into the system like mobile.
- Flash memory can be used instead of secondary memory. Hence, we can load NT in embedded system. E.g. Embedded Linux OS can be loaded into wristwatches.

Operating system
- In desktop, the selection of O.S. is limited.
- In embedded system, a variety of operating systems are available which can be ported into the embedded system.
- It is categorized as follows : Embedded OS, real time OS and mobile OS. These operating systems occupy less area in memory than desktop.
- For real time applications, we should use real time OS.
- We can develop our own OS kernel.
- We can use open source OS like Linux. This OS is free and can be customized.

Reliability
- Embedded system often reside in machines that are expected to run continuously for years without errors and in some cases recover by themselves, if an error occurs.
- So, the software is usually developed and tested more carefully than that for personal computers and unreliable moving parts such as disk drives, switches or buttons are avoided.

Unit cost
- The monetary cost of manufacturing each copy of the system, excluding NRE cost.

NRE cost

- The monetary cost of designing the system. Once the system is designed, any number of units can be manufactured without incurring any additional design cost (hence the term "non-recurring").

Size

- The physical space required by the system, often measured in bytes for software, and gates or transistors for hardware.

Performance

- The execution time or throughput of the system.

Power

- The amount of power consumed by the system, which determines the lifetime of a battery, or the cooling requirements of the IC, since more power means more heat.

Flexibility

- The ability to change the functionality of the system without incurring heavy NRE cost. Software is typically considered very flexible.

Time to market

- The amount of time required to design and manufacture the system to the point the system can be sold to customers.

Time to prototype

- The amount of time to build a working version of the system, which may be bigger or more expensive than the final system implementation, but can be used to verify the system's usefulness and correctness and to refine the system's functionality.

Correctness

- Our confidence that we have implemented the system's functionality correctly. We can check the functionality throughout the process of designing the system and we can insert test circuitry to check that manufacturing was correct.

Safety

- The probability that the system will not cause harm.

Q.2(b) Interface 4×4 matrix key pad to 8051 and write the C language program to send the key code on any port. **[4]**

(A)

```
# include <stdio.h>
# include <reg51.h>
# define ROW P1
# define COL P3
void delay (unsigned int);
void main ( )
{
   unsigned char keypad[4][4] = {'0' , '1' , '2' , '3' , '4' , '5' , '6', '7' , '8' , '9', 'A' , 'B', 'C' , 'D' , 'E' , 'F' } ;
unsigned char colloc , rowloc ;
COL = 0×FF;

while(1)
{
do
         {
ROW = 0×00 ;
colloc = COL ;
colloc&= 0×0F ;
         }
while (colloc!= 0×0F) ;
         do
         {
         do
              {
              delay (20);
              colloc = COL ;
colloc&= 0×0F ;
              }
         while (colloc == 0×0F) ;
         delay (20);
         colloc = COL ;
colloc&= 0×0F ;
         }
         while (colloc = = 0x0F) ;

while(1)
         {
ROW = 0×FE ;
                    colloc = COL ;
colloc&= 0×0F ;
                    if (colloc!= 0×0F)
                    {
                    rowloc = 0 ;
                    break ;
                    }
ROW = 0×FD ;
                    colloc = COL ;
colloc&= 0x0F ;
                    if (colloc!= 0x0F)
                    {
                    rowloc = 1 ;
                    break ;
                    }
   ROW = 0×FB ;
                    colloc = COL ;
colloc&= 0×0F ;
                    if (colloc!= 0×0F)
                    {
                    rowloc = 2 ;
                    break ;
                    }
           ROW = 0×F7 ;
                    colloc = COL ;
colloc&= 0×0F ;
                    rowloc = 3 ;
                    break ;
         }

if (colloc = = 0×0E)
         P0 = keypad [rowloc][0] ;
```

```
            else if (colloc = = 0x0D)
            P0 = keypad [rowloc][1] ;
            else if (colloc = = 0x0B)
            P0 = keypad [rowloc][2] ;
            else
            P0 = keypad [rowloc][3] ;
        }
    }

    void delay (unsigned int i)
    {
    unsignedint x , y;
    for(x=0; x< i ; x++)s
    for(y=0; y<1275; y++);
    }
```

Q.2(c) Define multitasking. Describe the process of multitasking with suitable example. [4]

(A) The ability of operating system to hold the multiple tasks (processes) in memory and allot CPU for each task by task switching is known as multitasking. Multitasking involves scheduling of CPU time among various tasks as per scheduling algorithm.

The co- operative multitasking, pre emptive multitasking and non pre emptive multitasking has the following scheduling algorithms

1. First come first served /FIFO
2. Last come First served/LIFO
3. Shortest Job First Scheduling
4. Priority based Scheduling
5. Round Robin scheduling.

Non pre emptive : First come First served :
This algorithm allocates the CPU time to task based on the order in which they enter the 'Ready' queue. The first entered task gets the service first. The draw back of this method is that it supports monopoly of the task. A task does not need any input/output operation, continues it's execution until it finishes it's task. If the tasks contain any input output operation, the CPU is relinquished by the task. In general FCFS favours CPU intensive usage tasks and input/output intensive usage task.

Q.2(d) Find the contents of port after execution of the following code. [4]

(i) P2 = ~ 0 × FF

(ii) ACC = ACC & 0 × 00

(A) (i) P2 = ~ 0 × FF
P2 = 0 × 00

(ii) ACC = 0 × 00
ACC = 0 × 00

Q.2(e) How the assembly language instruction is used in C language Program. [4]

(A) Example of writing assembly language instruction in c:

```
#include<reg51.h>
 void main()
{

   int a = 3, b = 3, c;
```

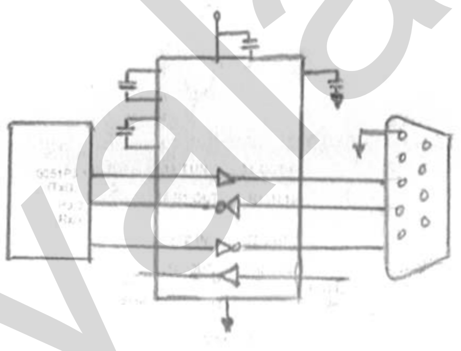
```
  asm {
     mov ax,a
     mov bx,a
     add ax,bx
     mov c,ax
  }

}
```

It is used by using asm in { } brackets Or writing asm before each instruction as shown below

```
#include<reg51.h>
 void main()
{

  int a = 3, b = 3, c;

asm mov ax,a
asm mov bx,a
asm add ax,bx
asm mov c,ax

}
```

Q.2(f) Draw and describe the RS 232 interface with 8051 using Max232C. **[4]**

(A)



The above figure shows the interface between microcontroller and RS 232. It uses max 232 line driver/voltage converters for RS 232 signal to TTL voltage level acceptable to 8051. Max 232 works on single power supply i.e. +5V. It has two sets of line driver. It uses 4 capacitors ranging from 1 microfarad to 220 microfarad.

Q.3 Attempt any FOUR of the following : **[16]**

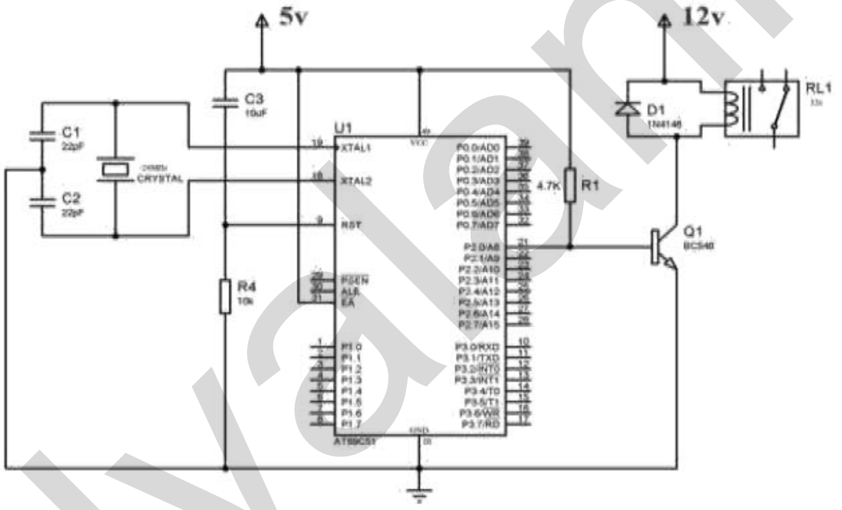
Q.3(a) Write 89C51 'C' Program to transfer the message "MSBTE" serially at 4800 baud rate continuously. Use 8 bit data and 1 stop bit. **[4]**

(A)

```
#include <stdio.h>
#include <reg51.h>
void sertext (unsigned char) ;
void main( )
  {
    TMOD = 0x20 ;
    SCON = 0x50 ;
    TH1=0xFA ;
    TR1 = 1 ;
sertext ('M') ;
sertext ('S') ;
sertext ('B') ;
sertext ('T') ;
sertext ('E') ;
   while (1)
    {
    }
  }

void sertext (unsigned char x)
  {
    SBUF = x ;
    while (TI = = 0) ;
     {
       TI = 0 ;
     }
  }
```

Q.3(b) Draw the interfacing of relay with 89C51 microcontroller. Write C language program to make relay ON/OFF after certain delay. **[4]**

(A)



Program

```
#include <reg51.h>
sbit relay =P2 0;
void ms_delay (unsigned int);               //delay function
void main(void)
{
    P2=0;        //initialize port
    while(1)     //loop forever
    {
        relay=1;                //relay is on
        ms_delay(200);          //delay
        relay=0;                //relay is off
        ms_delay(200);          //delay
    }
}
    void ms_delay (unsigned int itime)
    {
    unsigned int x,y;
    for(x=0; x<itime; x++)
    for(y=0; y<1275; y++)
    }
```

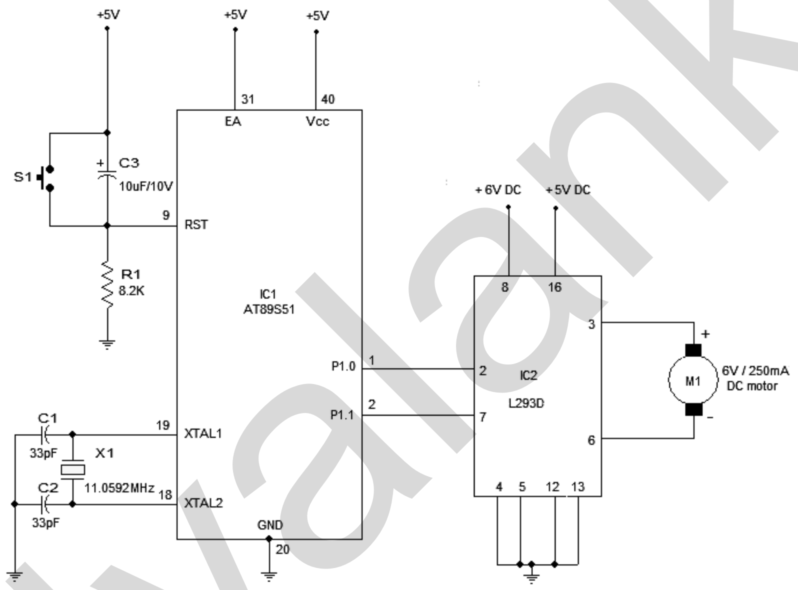
Q.3(c) Write 89C51 'C' Program to toggle bit of P1.5 continuously with a 250 ms. Use the timer interrupt delay. **[4]**

(A) Consider the mode '1' of the timer. The maximum delay generated in mode '1' is 71.7ms.Therefor to generate the delay of 250ms the timer is operated for 3.48 = 3 times.

```
# include <stdio.h>
# include<reg51.h>
sbit wave = P1^5;
void main ( )
{
While(1)
{
    unsigned intx,y;
    for (x=0;x<3;x++)
{
Wave = 0;
TMOD = 0×00;
TH1 = 0×00;
TL1 = 0×00;
TR1 = 1;
While (TF1= =0)
TR1 = 0;
TF1 = 0;
}
for (y = 0; y <3; y++)
{
Wave = 1;
TMOD = 0×00;
TH1 = 0×00;
TL1 = 0×00;
TR1 = 1;
While (TF1 = = 0)
TR1 = 0;
TF1 = 0;
}
}
}
```

Q.3(d) Interface the DC motor to microcontroller and write the C program to drive the motor. **[4]**

(A)
```
#include<regx51.h>
void my_delay(void);
sbit mpin_1 = P1^0;
sbit mpin_2 = P1^1;
void main()
{
do
{
mpin_1 = 1;
mpin_2 = 0; //Rotates Motor Anti-Clockwise
my_delay();
mpin_1 = 1;
mpin_2 = 1; //Stops Motor
my_delay();
mpin_1 = 0;
mpin_2 = 1; //Rotates Motor Clockwise
my_delay();
mpin_1 = 0;
mpin_2 = 0; //Stops Motor
my_delay();
} while(1);
}
void my_delay()
{
unsigned char i,j,k;
for(i=0;i<6;i++)
for(j=0;j<255;j++)
for(k=0;k<255;k++);
}
```



Q.3 (e) Define the semaphore and deadlock. **[4]**

(A) Semaphore :

- RTOS kernels provide a semaphore to ensure the integrity of data during sharing the common data.
- Resource count and a wait queue is associated with semaphore where the resource count shows availability of resource and the wait queue manages the tasks waiting for resources from the semaphore.
- A semaphore functions as a manager which decides whether a task has the access to the resource.
- When task acquires the semaphore, it receives an access to the resource.
- The number of times the semaphore can be acquired by the task is determined by the resource count of a semaphore.
- The resource count is decremented by one, when a task acquires the semaphore and its resource count is incremented by one when a task releases the semaphore.

Deadlock :

- A deadlock is a situation in which two or more tasks or processes waiting for each other to release a resource, or more than two processes are waiting for resources in a circular chain and neither request can be satisfied.

- Deadlock is a common problem in multiprocessing and multitasking where numbers of tasks or processes share a specific type of mutually exclusive resource.
- Now a day, the computers or embedded systems designed for the time-sharing or real time are equipped with a hardware lock which provides exclusive access to processes, forcing serialized access.
- There is no general solution to avoid deadlocks.

Q.3(f) Write C language program to rotate stepper motor by 90 degree clockwise. Assume step angle is 1.8 degree and 4 step sequence. [4]

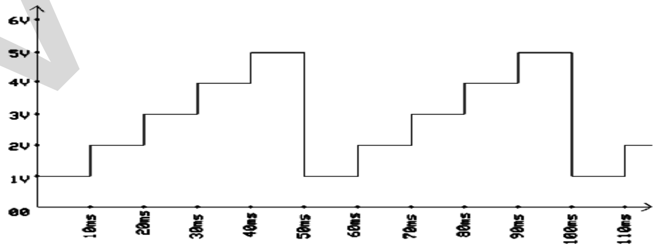
(A) Step angle=1.8 degree
Total steps required=90/1.8=50 Four step sequence so
Count=50/4=12.5=approximately 13

```
#include<regx51.h>
  void delay();
  void main()
  {
  int x;
  for (x=0;x,13;x++)
  {
  P0=0x33;
  delay(10);
  P0=0x66;
  delay(10);
  P0=0xcc
  delay(10);
  P0=0x99;
  delay(10);
  }
  }
  void delay(unsigned int t)
  {
  unsigned int x,y;
  for(x=0;x<=t;x++)
  for(y=0;y<=1275;y++);
  }
```

Q.4 Attempt any THREE of the following : [12]

Q.4(a) Write 'C' language program to generate triangular waveform continuously using DAC0808. The digital input is applied through Port P1. Also draw interfacing diagram. [4]

(A)



Given from wave form
5V = FF, 4V = CC, 3V = 99, 2V = 66, 1V = 33, 0V= 00.
For dely of 10ms Timer 1 mode 1 count = DBA6

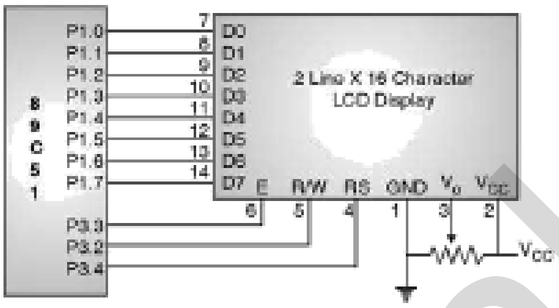
```
#include <stdio.h>
#include <reg51.h>
```

```
sbitcs = P3^3 ;
sbitwr = P3^4 ;
sbitxfer = P3^5 ;
void delay ()
void main( )
{

cs = 0 ;
while (1)
          {

wr = 1 ;
xfer = 1 ;
      P1 = 0×00 ;
xfer = 0 ;
wr = 0 ;
Delay( )
wr = 1 ;
xfer = 1 ;
     P1 = 0×33 ;
xfer = 0 ;
wr = 0 ;
Delay( )
wr = 1 ;
xfer = 1 ;
      P1 = 0×66 ;
xfer = 0 ;
wr = 0 ;
Delay( )
wr = 1 ;
xfer = 1 ;
      P1 = 0×99 ;
xfer = 0 ;
wr = 0 ;
Delay( )
wr = 1 ;
xfer = 1 ;
      P1 = 0×CC ;
xfer = 0 ;
wr = 0 ;
Delay( )

wr = 1 ;
xfer = 1 ;
      P1 = 0×FF;
xfer = 0 ;
wr = 0 ;
Delay( )

}
          }

Void delay ()
{
```

```
TMOD = 0×10;
TH1 = 0×DB;
TL1 = 0×A6;
TR1;
While (TF1==0)
TR1 = 0;
TF1 = 0;
}
```

Q.4(b) Draw the interfacing of LCD display to 89C51 microcontroller and describe the function of RS and RW pins. **[4]**

(A)

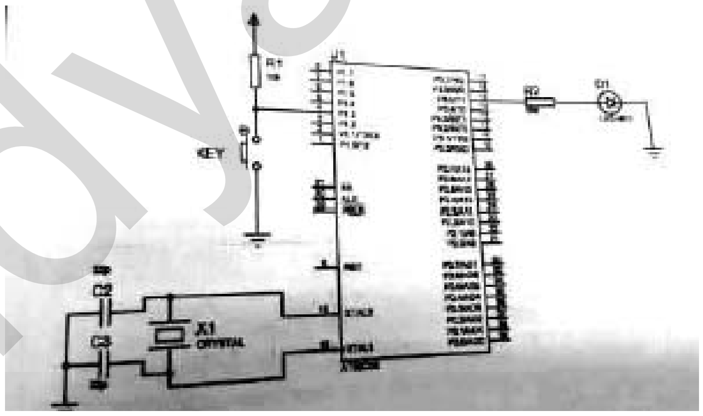


(1) **RS:** RS is the register select pin used to write display data to the LCD (characters), this pin has to be high when writing the data to the LCD. During the initializing sequence and other commands this pin should low.

(2) **R/W:** Reading and writing data to the LCD for reading the data R/W pin should be high (R/W=1) to write the data to LCD R/W pin should be low (R/W=0).

Q.4(c) Draw the interfacing of key and LED to 89C51 microcontroller pins P1.0 and P2.0 respectively. Write C language program to read the status of key and display it on LED. (Key open = LED OFF and key closed = LED ON) **[4]**

(A)



Program

```
#include<reg51.h>
sbit=sw P1^3;           //make P1.3 as input
sbit led=P3^5;          //make P1.3 as output
void main (void)
{
sw=1;
led=0;
while(1)                //repeat loop
{
if(sw= =0)              //if switch is closed on the led else off the led
led=1;
else
led=0;
}                         //end of while
}                         //end of main
```

Q.4(d) Write C language program to generate a square wave of 2 KHz frequency on P1.1 pin by using timer 0 and mode 1. Assume XTAL frequency is 11.0592 MHz. **[4]**

(A) Crystal frequency= 11.0592 MHz
I/P clock = 11.0592 X 106 = 11.0592MHz
1/12 x 11.0592Mhz = 921.6 Khz
Tin = 1.085μ sec
For 2 kHz square wave
Fout = 2 KHz Tout = 1/2 X 103
Tout = 500μ sec
Consider half of it = Tout = 250μ sec
N = Tout / Tin = 250/1.085= 230
65536-230= (65306p10) =FF1A

Program

```
#include<reg51.h>
void delay(void);
sbit p=P1^5;
void main (void)
{
while (1)
{
    p=~p;
    delay();
}
}
void delay()
{
TMOD=0X01;       //set timer 0 in mode 1 i.e. 16 bit number
TL0=0X1AH;       //load TL register with LSB of count
TH0=0XFFH ;      //Load TH register with MSB of count
TR0=1            //start timer 0
While(TF0==0)    //wait until timer rolls over
TR0=0;           //Stop timer 0
TF0=0;           //Clear timer flag 0
}
```

Q.4(e) List the scheduling algorithm or RTOS. Describe any one scheduling algorithm in brief. **[4]**

(A) (i) First in first out
(ii) Round-robin algorithm
(iii) Round robin with priority
(iv) Shortest job first
(v) Non Preemptive multitasking
(vi) Preemptive multitasking

(i) First in first out
In first in first out scheduling algorithm the task which are ready to run are kept in queue and the cpu serves the task on first in first served basis. this scheduling algorithm is shown in fig. is very simple to implement but not well suited for most applications because it is difficult to

estimate the amount of time a task has to wait for being executed . However this is good algorithm for an embedded system has to perform few small tasks all with small execution time. If there is no time critically and the number of tasks is small, this algorithm can be implemented.

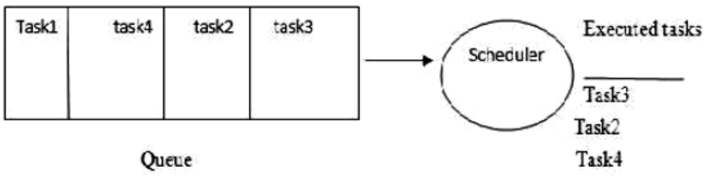


Fig Scheduling algorithm : FIFO

Q.4(f) Write C language program to read P1 and store the one's complement of P1 to P2. **[4]**

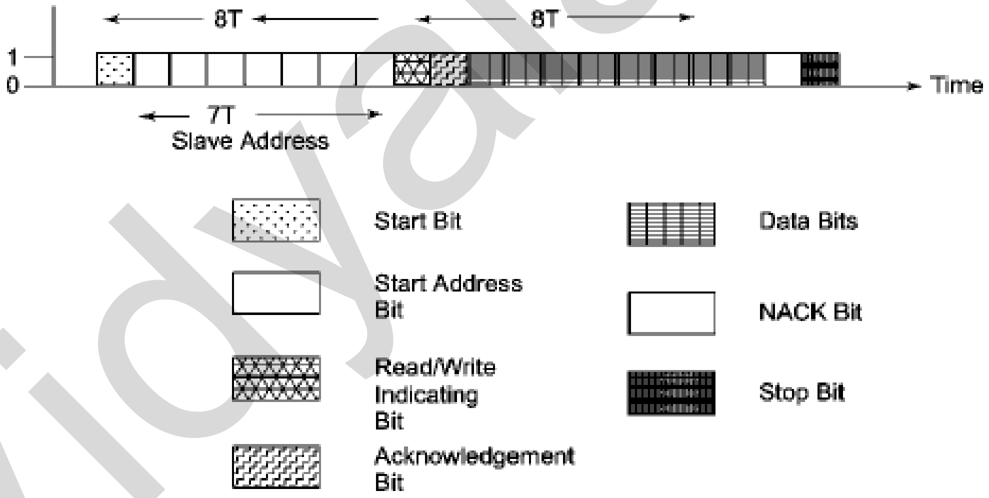
(A)

```
#include<regx51.h>
void main(void)
{
unsigned char a,b,c;
P1=0xFFH;                //set port 0 as an input port
P2=0x00FH;               //set port 1 as an output port
a=P1;                    //read number from port 1
a=~a;                    //logical not operator
P2=a;                    //store one's complement to port 2
```

Q.5 Attempt any FOUR of the following : **[16]**

Q.5(a) Draw the frame format of I^2C and explain each field in brief. **[4]**

(A)



- First field of 1 bit – START bit = one.
- Second field of 7 bits – address field. It defines the slave address, which is being sent the data frame(of many bytes) by the master.
- Third field of 1 control bit – defines whether a read or write cycle is in progress.
- Fourth field of 1 control bit – defines whether is the present data is an acknowledgment (from slave).
- Fifth field of 8 bits – I2C device data byte (transmitted by either master or slave).
- Sixth field of 1 bit – bit NACK (negative acknowledgment) from the receiver (when master is receiving). If active the acknowledgment is expected from the slave.
- Seventh field of 1 bit – stop bit send by master.

Q.5(b) List any four software development tools used in an embedded system and state the function of each. **[4]**

(A) Software development tools:

- Compiler
- Cross assembler
- Cross compiler
- Locators
- Loaders
- Simulators
- Debugger
- Integrated development environment (IDE)

Explanation :

Compiler

It is a computer program that transforms the source code written in a programming or source language into another computer language i.e. target language i.e. binary code known as object code.

Cross assembler:

It is useful to convert object codes for microcontrollers or processor to other codes for another microcontrollers or processor and vice versa.

Cross compiler:

It is used to create executable code other than one on which the compiler is run. They are used to generate executable for embedded systems or multiple platforms.

Linker/Locator:

- It is used for relocation process.
- It is done during compilation also it can be done at run time by a relocating loader.
- It is a program that takes one or more objects generated by compiler and combines them into a single executable program.

Simulators

- A simulator is the s/w that simulates an h/w unit like emulator, peripheral, network and I/O devices on a PC.
- It defines a processor or processing device as well as various versions for the target system.
- Monitors the detailed information of as source code part with labels and symbols during the execution for each single step.
- Provides the detailed information of the status of memory RAM and simulated ports, simulated peripheral devices of the defined target system.

Integrated Development Environment (IDE):

An IDE is a software application that provides comprehensive facilities to computer programmers for software development.

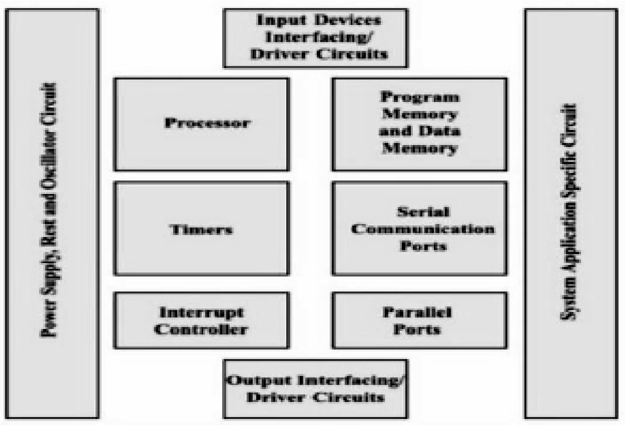
An IDE consists of:

- A source code editor.
- A compiler and or interpreter.
- Build automation tools.
- A debugger.

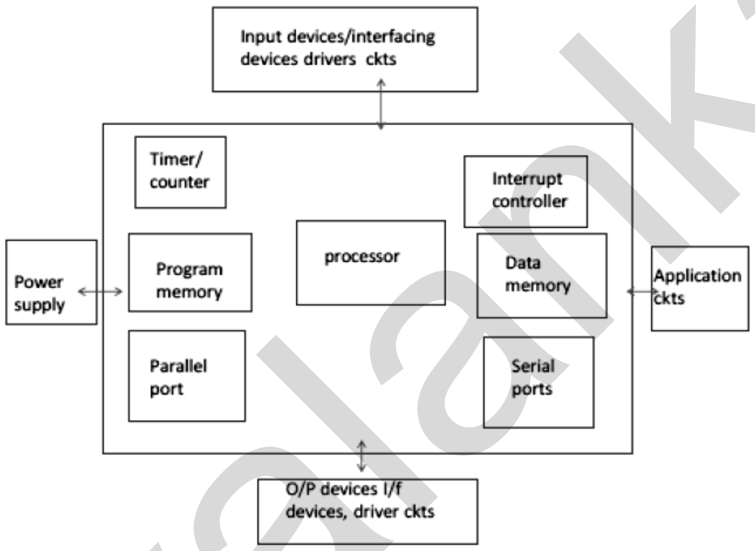
IDE is dedicated to a specific programming language, allowing a feature set that most closely matches the programming paradigms [model] of the language. IDE's typically present a single program in which all development is done. This program typically provides many features for authoring, modifying, compiling, deploying, and debugging software.

Q.5(c) Draw block diagram of embedded system and describe any four hardware units of embedded system. **[4]**

(A)


Input Devices Interfacing/ Driver Circuits
Power Supply, Rest and Oscillator Circuit
Processor
Program Memory and Data Memory
Timers
Serial Communication Ports
Interrupt Controller
Parallel Ports
System Application Specific Circuit
Output Interfacing/ Driver Circuits

OR



Processor:

The processor is the heart of embedded system. The selection of processor is based on the following consideration :

- Instruction set.
- Maximum bits of operation on single arithmetic and logical operation.
- Speed.
- Algorithms processing and capability.
- Types of processor (microprocessor, microcontroller, digital signal processor, application specific processor, general purpose processor).

Power source:

Internal power supply is must. Es require from power up to power down to start time task. Also it can run continuously that is stay "On' system consumes total power hence efficient real time programming by using proper 'wait' and 'stop' instruction or disable some unit which are not in use can save or limit power consumption.

Clock / oscillator Circuits

The clock ckt is used for CPU, system timers, and CPU machine cycles clock controls the time for executing an instruction. Clock oscillator may be internal or external .It should be highly stable.

Real time clock(RTC)

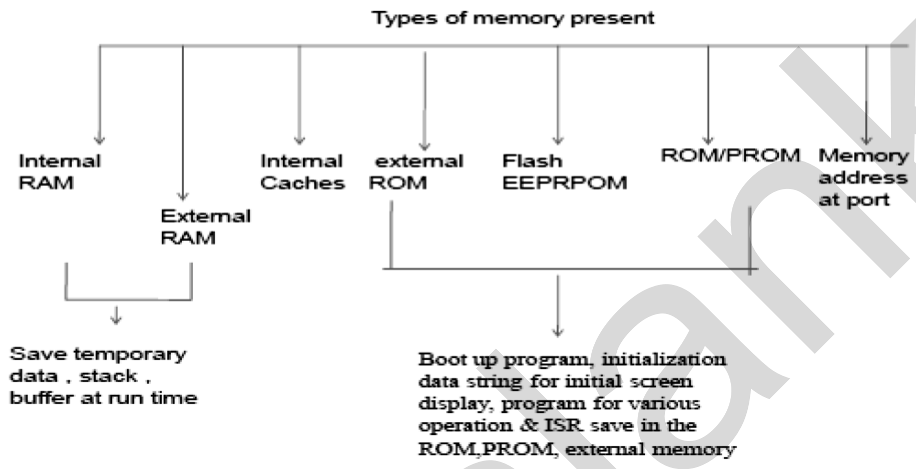
It require to maintain scheduling various tasks and for real time programming RTC also use for driving timers, counters needs in the system.

Reset Circuits and power on reset:
Reset process starts executing various instruction from the starting address. The address is set by the processor in the program counter. The reset step resent and runs the program in the following way.

- System program that execute from beginning.
- System boot up program.
- System initialization program.

Peripherals:
The peripheral devices are provided on the embedded system boards for an easy integration. Typical devices include serial port, parallel port, network port, keyboard and mouse ports, a memory unit port and monitor port. Some specialized embedded systems also have other ports such as CAN-bus.

Memories:



Types of memory present
Internal RAM
External RAM
Internal Caches
external ROM
Flash EEPRPOM
ROM/PROM
Memory address at port
Save temporary data , stack , buffer at run time
Boot up program, initialization data string for initial screen display, program for various operation & ISR save in the ROM,PROM, external memory

Q.5(d) Find the contents of port after execution of following code: **[4]**
(i) P2 = 0 × 7 4 >> 3; **(ii) P3 = 0 × 0 4 | 0 × 6 8;**

(A) **(i) P2 = 0 × 7 4 >> 3**
This indicates shift 74 by right 3 times.
74H = 0111 0100
After shifting by 3 times towards right we get
0000 1110 = 0X0E

(ii) P3 = 0 × 0 4 | 0 × 6 8

```
04   0000 0100
                  OR ing
68   0110 1000
--------------
     0110 1100
```

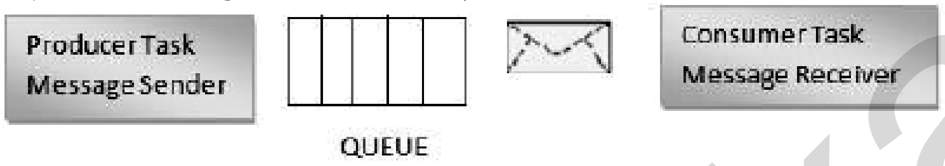
P3=0X6C

Q.5(e) Explain inter-task communication with reference to RTOS. **[4]**
(A) **Inter task communication:**
Inter task communication involves sharing of data among tasks through sharing of memory space, transmission of data, etc.. Some of the mechanisms available for executing Inter Task communications are :
(i) Message queue
(ii) Pipes
(iii) Remote procedure call
Inter task communications is executed using following mechanisms.

Message queues :
A message queue is an object used for inter task communication through which task send or receive messages placed in a shared memory.

The queue may follow:
(i) First In First Out (FIFO)
(ii) Last In First Out (LIFO) or
(iii) Priority (PRI) sequence.

Usually, a message queue comprises of an associated queue control block (QCB), name, unique ID, memory buffers, queue length, maximum message length and one or more task waiting lists. A message queue with a length of 1 is commonly known as a mailbox.



Pipes
A pipe is an object that provide simple communication channel used for unstructured data exchange among tasks. A pipe does not store multiple messages but stream of bytes. Also, data flow from a pipe cannot be prioritized.

Remote procedure call (RPC)
It permits distributed computing where task can invoke the execution of another task on a remote computer.

Q.5(f) Define embedded system. List any two advantages and disadvantages of embedded system. [4]

(A) Definition:
An Embedded system is a combination of computer hardware and software. As with any electronic system, this system requires a hardware platform and that is built with a microprocessor or microcontroller. The Embedded system hardware includes elements like user interface, Input/output interfaces, display and memory, etc. Generally, an embedded system comprises power supply, processor, memory, timers, serial communication ports and system application specific circuits.

Advantages (any two):
- **Design and Efficiency:**
  The central processing core in embedded system is generally less complicated, making it easier to design. The limited function required of embedded system allows them to design to most efficiently perform their function.
- **Cost:**
  The streamline make-up of most embedded system allows their parts to be smaller less expensive to produce.
- **Accessibility:**
  If something goes wrong with certain embedded systems they can be too inaccessible to repair. This problem is addressed in the design stage, so by programming an embedded system. So that it will not affect related system negatively when malfunctioning.
- **Maintenance:**
  Embedded systems are easier to maintain because the supplied power is embedded in the system and does not required remote maintenance.
- **Redundancies:**
  Embedded system does not involve the redundant programming.

Disadvantages (any two):

- **Difficult to change configurations and features:**
  Once an embedded system is deployed (or finalized), it will be difficult to change its configuration - both its hardware and software. Remote update of software is possible provided the capability is included. Hence, proper requirement analysis is a must before deployment. Hardware configuration change will be much more trickier which may require existing boards be completely replaced. I have seen this happen and it is not pretty.
- **Issue of scalability**
  Because it is difficult to change configuration, an embedded system cannot be easily scaled up as demand/scope changes. Said so, embedded systems can be designed to scale up for example using expansion ports or networking etc. This means it must be decided before hand during design phase for scale up provisions.
- **Limitation of hardware**
  With a limited memory or computing capability in most embedded systems, there is always a limitation (or an upper limit) on our software design (upgrade). Be always aware of "Memory" and "Speed".
- **Applied for a specific purpose**
  By definition, embedded systems are constrained in their objectives. If it is decided to "rehash" an existing embedded system for a completely different purpose, it will normally result in significant change(s) in either or both its hardware or/and software.

Q.6 Attempt any FOUR of the following : [16]

Q.6(a) List four features of each of the following. [4]

(i) Bluetooth (ii) Zighee

(A) (i) Bluetooth features

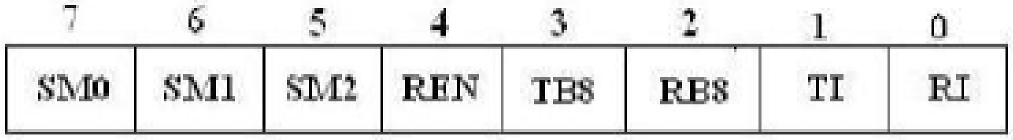
- IEEE Standard 802.15.1.
- Frequency (GHz) 2.4.
- Maximum raw bit rate (Mbps) 1-3.
- Typical data throughput (Mbps) 0.7-2.1
- Maximum (Outdoor) Range (Meters) 10 (class 2), 100 (class 1)
- Relative Power Consumption Medium.
- Example Battery Life in Days.
- Network Size 7.
- Cost Low

(ii) Zighee

- IEEE Standard 802.15.4
- Frequency (GHz) 0.868, 0.915, 2.4.
- Maximum raw bit rate (Mbps) 0.250.
- Typical data throughput (Mbps) 0.2.
- Maximum (Outdoor) Range (Meters) 10-100.
- Relative Power Consumption Very low.
- Example Battery Life – Months to years.
- Network Size 64,000+

Q.6(b) Draw the format of SCON register and explain all the bits. [4]

(A) SCON Register format:



7	6	5	4	3	2	1	0
SM0	SM1	SM2	REN	TB8	RB8	TI	RI

Function:
SM0, SM1 bits in SCON are used to define the type of the serial communication, baud rate and framing.

SM0	SM1	Mode	Function
0	0	0	Shift register, baud rate = f/12, **Synchronous** serial communication mode
0	1	1	8-bit UART; baud = variable , asynchronous serial communication mode
1	0	2	9-bit UART; baud = f/32 or f/64, asynchronous serial communication mode
1	1	3	9-bit UART; baud = variable, asynchronous serial communication mode

SM2 - bit is used for multiprocessor communication. Set or cleared by the program to enable multiprocessor communications in mode 2 and 3. When set to 1 an interrupt is generated if bit 9 of the received data is a 1; no interrupt is generated if bit 9 is 0.
REN – Receiver enable bit. To accept reception of data this bit must be 1;
TB8 – Transmitted bit 8.
RB8 – Received bit 8.
TI – Transmit interrupt flag. This will be enabled when all bits in the transmitted buffer is shifted out.
RI – Receive interrupt flag. This will be enabled when a character is received in the receiver buffer.

Q.6(c) State any four data types used in Embedded C, with their value range. [4]
(A)

Type	Bits	Bytes	Range
char	8	1	– 128 to + 127
unsigned char	8	1	0 to 255
enum	16	2	– 32,768 to + 32,767
short	16	2	– 32,768 to + 32,767
unsigned short	16	2	0 to 65,535
int	16	2	– 32,768 to + 32,767
unsigned int	16	2	0 to 65,535
long	32	4	– 2,147,483,648 to + 2,147,483,647
unsigned long	32	4	0 to 4,294,697,295

The Keil or SPJ C compiler provides several new data types to support a microcontroller and embedded systems applications:

Type	Bits	Bytes	Range
bit	1	0	0 to 1
sbit	1	0	0 to 1
sfr	8	1	0 to 255
sf16	16	2	0 to 65,535

Q.6(d) State the methods of task synchronization. Describe Semaphore with suitable example. [4]

(A) The methods of task synchronization are:

- Semaphore
- Message queue
- Mutual exclusion
- Dead lock
- Mailboxes
- Message Queues

Semaphore:

A semaphore is a single variable that can be incremented or decremented between zero and some specified maximum value. The value of the semaphore can communicate state information. A mail box flag is an example of a semaphore. The flag can be raised to indicate a latter is waiting in the mailbox. A semaphore is a means of protecting a resource/data shared between threads. It is a token based mechanism for controlling when a thread can have access to the resource/data. Usually a semaphore handle will be able to be received from the system by name/id.

Semaphores are used for two purposes :

(i) Process Synchronization.

(ii) Critical Section problem / Mutual Exclusion.

Example of using semaphores for Synchronization:

Assume two concurrent process P1 and P2 having statements S1 and S2. We want in any case S1 should execute first. this can be achieved easily by initialize Sem=0;

```
In process P1
{
    // Execute whatever you want to do
    // before executing P2
    S1;
signal(Sem);
}
in process P2
{
wait(Sem);
    S2;
}
```

Q.6(e) Differentiate between CAN with I2C protocols with respective to [4]

(i) Data transfer rate **(ii) Number of fields**

(iii) Addressing bit **(iv) Application**

(A)

Parameter	CAN	12C
Data Transfer Rate	2Mbps	100 kbps, 400kbps
Number of field	8	7
Address bit	11 (or 12)	7
Application	Automobile	Interconnection of IC, Serial bus

Q.6(f) What do you mean by starvation? [4]

(A) Starvation is the problem caused in multitasking system. When a process does not get the required resources for a long time then it is referred as starvation.

The priority based on scheduling algorithms executes the processes. Always the higher priority task art of the ready queue get the change to be executed.

The priority can be assigned while creation of the task or SJF (shortest job first) assign the highest priority to the task which has minimum execution time.

If there are two tasks in the ready queue, A and B the task with highest priority (say B) is executed first and A is waiting for B to be completed. Now if in between some other task having higher priority than task A is ready in queue, then task A will not be executed and if any other high priority task is dependent on the results obtained by task A, then the higher priority task will not be executed as A is not executed.

Thus the task a is not getting the required resource for its execution for the long time. This is called as starvation.

Thus starvation is said to occur when a task with a lower priority is not getting a change to executed if more and more processes with higher priorities enter the ready queue before the process with lower priority state is executed.

If can be effectively solved by dynamically raising the priority of the low priority task which is under starvation this technique is called as “aging”.

❑ ❑ ❑ ❑ ❑