SUMMER– 19 EXAMINATION

Subject Name: Embedded Systems **Model Answer Subject Code:** 17658

1

Important Instructions to examiners:

1) The answers should be examined by key words and not as word-to-word as given in themodel answer scheme.
2) The model answer and the answer written by candidate may vary but the examiner may tryto assess the understanding level of the candidate.
3) The language errors such as grammatical, spelling errors should not be given moreImportance (Not applicable for subject English and Communication Skills.
4) While assessing figures, examiner may give credit for principal components indicated in thefigure. The figures drawn by candidate and model answer may vary. The examiner may give credit for anyequivalent figure drawn.
5) Credits may be given step wise for numerical problems. In some cases, the assumed constantvalues may vary and there may be some difference in the candidate's answers and model answer.
6) In case of some questions credit may be given by judgement on part of examiner of relevant answer based on candidate's understanding.
7) For programming language papers, credit may be given to any other program based on equivalent concept.

Q. No.	Sub Q. N.	Answers	Marking Scheme
1	a)	**Attempt any THREE of the following:**	**12- Total Marks**
	(i)	**List ports of 89C51 and alternate pin functions of port 3.**	**4M**
	Ans:	There are 4 ports in 89C51. They are as follows: 1. PORT 0 2. PORT 1 3. PORT 2 4. PORT 3 Alternate functions of Port 3 are,	**(list 1M,pin functions-3M)**

P3 BIT	FUNCTION	PIN
P3.0	RXD	10
P3.1	TXD	11
P3.2	$\overline{INT0}$	12
P3.3	$\overline{INT1}$	13
P3.4	TO	14
P3.5	TI	15
P3.6	$\overline{WR}$	16
P3.7	$\overline{RD}$	17

(ii) **List various software development tools available in IDE .Explain any one in brief.** **4M**

Ans: **Software development tools:** **2M**

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

Explanation : **(Any one tool)2M**

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):** • It supports for defining a processor family and its version • Support a user definable assembler to support a new version or a type of processor. • Provides multiuser environment • Supports conditional and unconditional break points • Provide Debugger.	
	(iii)	**Draw and explain CAN bus protocol.**	**4M**
	Ans:	<u>FRAME TYPES</u> Message transfer is manifested and controlled by four different frame types: A DATA FRAME carries data from a transmitter to the receivers. A REMOTE FRAME is transmitted by a bus unit to request the transmission of the DATA FRAME with the same IDENTIFIER. An ERROR FRAME is transmitted by any unit on detecting a bus error. An OVERLOAD FRAME is used to provide for an extra delay between the preceding and the succeeding DATA or REMOTE FRAMEs. <u>FRAME FORMAT</u>	**Draw-2M, Explain-2M**

4

SOF	ARBITARY	RTR	CONTROL	DATA	CRC	ACK	EOF
X1	X11	X1	X6	X1- 64	X9	X2	X1

OR

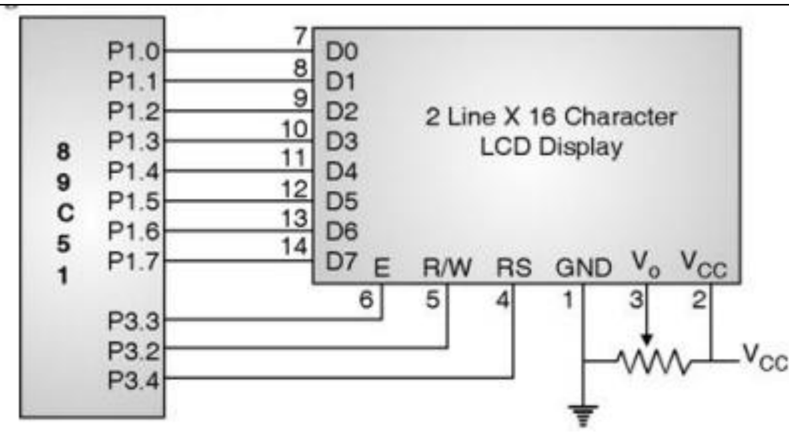
SOF	11-BIT ARBITRATION ID	SRR	IDE	18-BIT ARBITRATION ID	RTR	r0	DLC	0...8 BYTES DATA	CRC	ACK	EOF

- **CAN Frame --** an entire CAN transmission: arbitration ID, data bytes, acknowledge bit, and so on. Frames also are referred to as messages.

.

- **SOF (start-of-frame) bit** – indicates the beginning of a message with a dominant (logic 0) bit.
- **Arbitration ID** – identifies the message and indicates the message's priority. Frames come in two formats -- standard, which uses an 11-bit arbitration ID, and extended, which uses a 29-bit arbitration ID.
- **IDE (identifier extension) bit** – allows differentiation between standard and extended frames.
- **RTR (remote transmission request) bit** – serves to differentiate a remote frame from a data frame. A dominant (logic 0) RTR bit indicates a data frame. A recessive (logic 1) RTR bit indicates a remote frame.
- **DLC (data length code)** – indicates the number of bytes the data field contains.
- **Data Field** – contains 0 to 8 bytes of data.
- **CRC (cyclic redundancy check)** – contains 15-bit cyclic redundancy check code and a recessive delimiter bit. The CRC field is used for error detection.
- **ACK (Acknowledgement) slot** – any CAN controller that correctly receives the message sends an ACK bit at the end of the message. The transmitting node checks for the presence of the ACK bit on the bus and reattempts transmission if no acknowledge is detected. National Instruments Series 2 CAN interfaces have the capability of listen-only mode. Herein, the transmission of an ACK bit by the monitoring hardware is suppressed to prevent it from affecting the behavior of the bus.

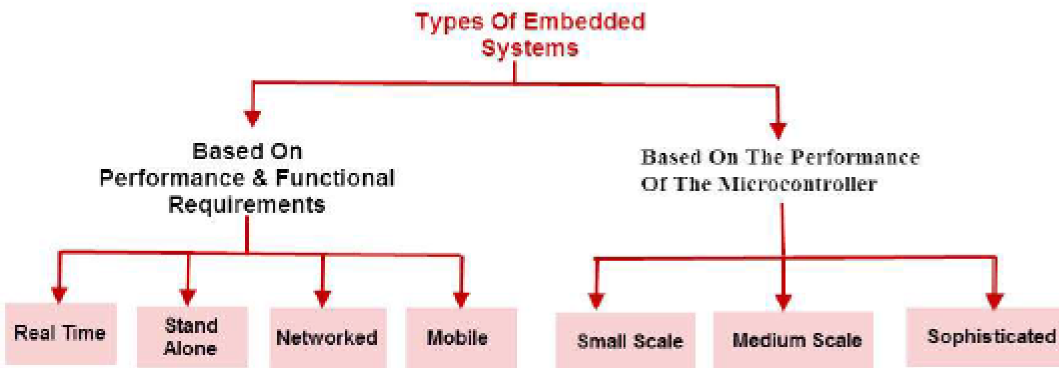
CAN Signal – an individual piece of data contained within the CAN frame data field. You also

		can refer to CAN signals aschannels. Because the data field can contain up to 8 bytes of data, a single CAN frame can contain 0 to 64 individual signals (for 64 channels, they would all be binary) **(EOF) END OF FRAME:** • Each DATA FRAME and REMOTE FRAME is delimited by a flag sequence consisting of 'recessive' bits	
	(iv)	**Draw labeled diagram to interface 16×2 LCD with 89C51.State the function of pins.** **1) RS** **2) R/W** **3) EN**	**4M**
	Ans:	 **Functions:** 1. RS: Register Select RS = 0 →lcdcmd Register RS = 1 → Data Register 2. R/W: 0 → Write , R/W = 1 → Read 3. EN: Enable Used to latch the lcddata present on the lcddata pins A HIGH - LOW signal is required to latch the lcddata. The LCD interprets and executes our lcdcmd at the instant the EN line is brought low. If you never bring EN low, your instruction will never be executed.	**1M-draw, 1M each-functions**

SUMMER– 19 EXAMINATION

Subject Name: Embedded Systems **Model Answer** Subject Code: 17658

6

	b)	**Attempt any ONE of the following:**	**06- Total Marks**
	(i)	**State various types of Embedded system . Explain any one in brief. State any four applications of embedded system.**	**6M**
	Ans:	**Types of embedded system: 2M**  **Explanation : (any two of the following) 2M** **1. Stand Alone Embedded Systems** Stand-alone embedded systems do not require a host system like a computer, it works by itself. It takes the input from the input ports either analog or digital and processes, calculates and converts the data and gives the resulting data through the connected device-Which either controls, drives or displays the connected devices. Examples for the stand alone embedded systems are mp3 players, digital cameras, video game consoles, microwave ovens and temperature measurement systems. **2. Real Time Embedded Systems** A real time embedded system is defined as, a system which gives a required o/p in a particular time. These types of embedded systems follow the time deadlines for completion of a task. Real time embedded systems are classified into two types such as soft and hard real time systems. **3. Networked Embedded Systems** These types of embedded systems are related to a network to access the resources. The	**ES Types:2 M,** **Any two explanation :2 M**

		connected network can be LAN, WAN or the internet. The connection can be any wired or wireless. This type of embedded system is the fastest growing area in embedded system applications. The embedded web server is a type of system wherein all embedded devices are connected to a web server and accessed and controlled by a web browser. Example for the LAN networked embedded system is a home security system wherein all sensors are connected and run on the protocol TCP/IP **4. Mobile Embedded Systems** Mobile embedded systems are used in portable embedded devices like cell phones, mobiles, digital cameras, mp3 players and personal digital assistants, etc. The basic limitation of these devices is the other resources and limitation of memory. **5. Small Scale Embedded Systems** These types of embedded systems are designed with a single 8 or 16-bit microcontroller that may even be activated by a battery. For developing embedded software for small scale embedded systems, the main programming tools are an editor, assembler, cross assembler and integrated development environment (IDE). 6. **Medium Scale Embedded Systems** These types of embedded systems design with a single or 16 or 32 bit microcontroller,RISCs or DSPs. These types of embedded systems have both hardware and softwarecomplexities. For developing embedded software for medium scale embedded systems, themain programming tools are C, C++, and JAVA, Visual C++, and RTOS, debugger, source code engineering tool, simulator and IDE. **7. Sophisticated Embedded Systems** These types of embedded systems have enormous hardware and software complexities, that may need ASIPs, IPs, PLAs, scalable or configurable processors. They are used for cutting edge applications that need hardware and software Co-design and components which have to assemble in the final system. **Applications: (any four of the following)** **1. Applications of Embedded Systems:** Embedded systems are used in different applications like automobiles, telecommunications, smart cards, missiles, satellites, computer networking and digital consumer electronics. **2. Embedded Systems in Automobiles and in telecommunications** 1. Motor and cruise control system 2. Body or Engine safety 3. Entertainment and multimedia in car 4. E-Com and Mobile access 5. Robotics in assembly line 6. Wireless communication 7. Mobile computing and networking **3. Embedded Systems in Smart Cards, Missiles and Satellites** Security systems	**Applications: 2M**

		Telephone and banking Defense and aerospace Communication **4. Embedded Systems in Peripherals & Computer Networking** Displays and Monitors Networking Systems Image Processing Network cards and printers **5. Embedded Systems in Consumer Electronics** Digital Cameras Set top Boxes High Definition TVs DVDs	
	(ii)	**State the scheduling algorithms of RTOS and describe the concept of round robin scheduling.**	**6M**
	Ans:	**<u>State the Scheduling algorithms of RTOS:</u>** 1. First in first out 2. Round-robin algorithm 3. Round robin with priority 4. Shortest job first 5. Non Preemptive multitasking 6. Preemptive multitasking **<u>Round Robin (RR)</u>** • Each process gets a small unit of CPU time (*time quantum*), usually 10-100 milliseconds. After this time has elapsed, the process is preempted and added to the end of the ready queue. • If there are n processes in the ready queue (as a FIFO) and the time quantum is q, then each process gets $1/n$ of the CPU time in chunks of at most q time units at once. No process waits more than $(n-1)q$time units. Example of RR with Time Quantum = 20 <u>Process</u> <u>Burst Time</u> P_1 53	**(State-2M,describe-4M)**

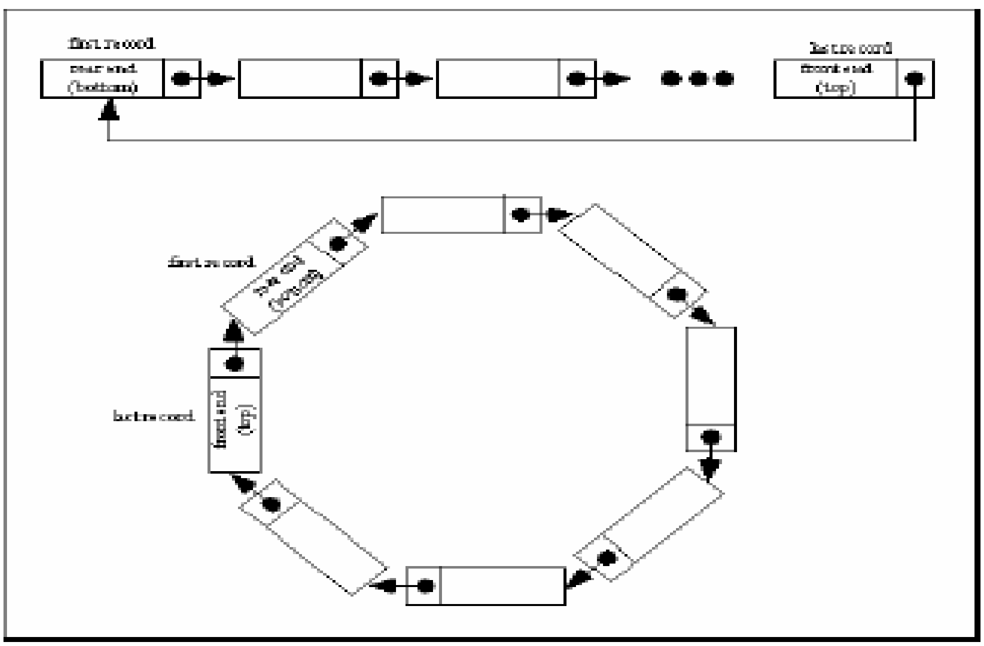
P_2	17
P_3	68
P_4	24

The Gantt chart is:

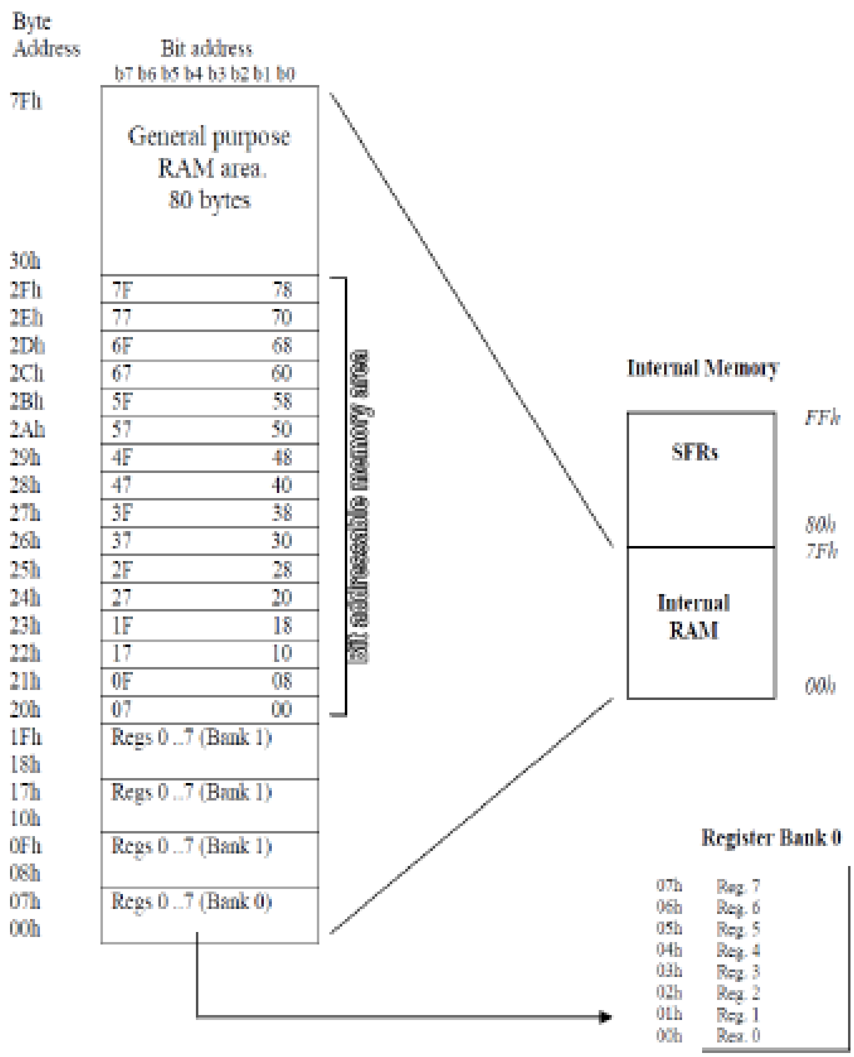
P_1	P_2	P_3	P_4	P_1	P_3	P_4	P_1	P_3	P_3

0 2 3 5 7 9 117 121 134 154 162

- Typically, higher average turn around than SJF, but better *response*



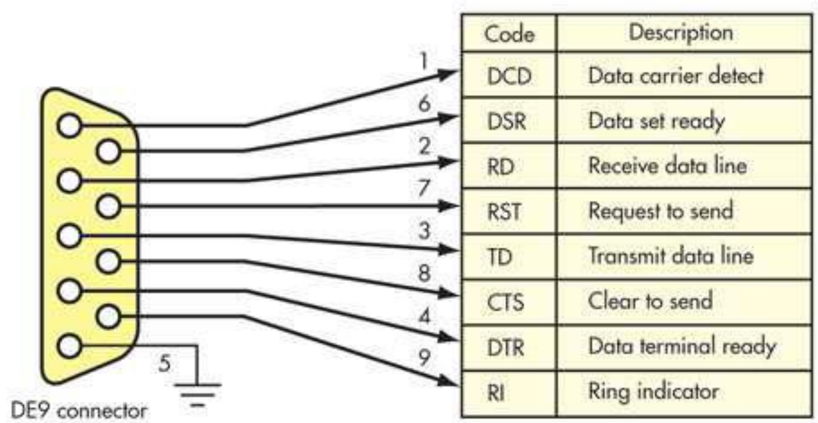
Note: Any other Related example (diagram) can consider

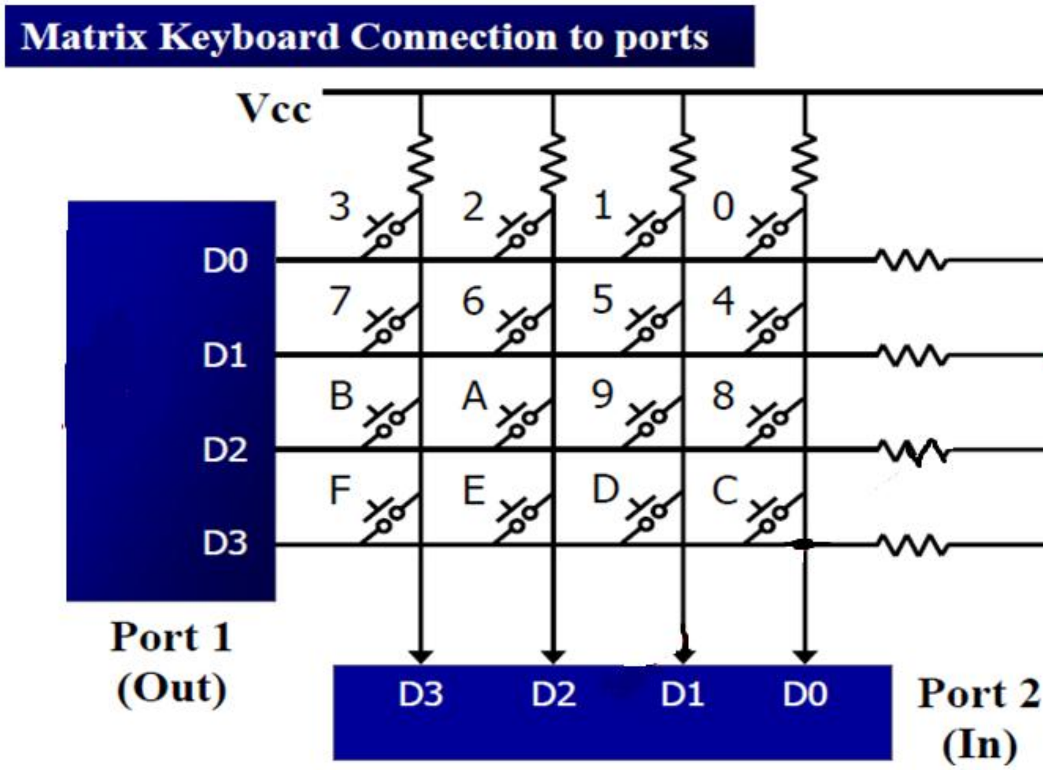
Q. No.	Sub Q. N.	Answers	Marking Scheme
2		**Attempt any FOUR of the following:**	**16- Total Marks**
	a)	**Draw the internal data memory structure of 89C51 and describe register banks.**	**4M**
	Ans:	 Fig.- Internal memory organization • The lowest 32 bytes of RAM are reserved for 4 general register banks. The 8051 has 4	**2M- diagram, 2M- explain**

		selectable banks of 8 addressable 8-bit registers, R0 to R7. • This means that there are essentially 32 available general purpose registers, although only 8 (one bank) can be directly accessed at a time. • Depending on the status of RS1,RS0 in PSW register bank selection is done. By default Bank0 is selected. **RS0, RS1: register bank selects bits** These two bits are used to select one of the four register banks n internal RAM in the table. By writing zeroes and ones to these bits, a group of registers R0- R7 can be used out of four registers banks in internal RAM	
	b)	**Write C program for 89C51 to read data from port P1 and P2.Compare the data and send bigger data on port 3.**	4M
	Ans:	**(any other relevant logic s can be use)**	4M

RS1	RS0	Space in RAM
0	0	Bank 0 (00H- 07H)
0	1	Bank 1 (08H-0FH)
1	0	Bank2 (10H-17H)
1	1	Bank3 (18H-1FH)

```
#include <reg51.h>
void main(void)
{
unsigned char a, b ;
P1=0xFF; //make P1 input port
P2=0xFF; //make P2 input port
P3=0x00; //make P0 output port
while (1)
{
a=P1; //get a byte from P1
b=p2;//get data from p2
if (a>b)
P3=a;
else
P3=b;
}
}
```

c)	**Draw the pin out of RS232 and describe the function of TXD,RXD,DTE and DCE pins.**	**4M**
Ans:	**(2M diagram,2M functions)**  **Function of all pins:** **Pin 1 - Data carrier detect (DCD):** The DCE tells the DTE it is receiving a valid input signal. **Pin 2 - Received data (RD):** This is the actual signal received from the DTE. **Pin 3 -Transmit data (TD):** This is the transmitted signal from the DTE. **Pin 4 -Data terminal ready (DTR):** This line is from the DTE to the DCE indicating readiness to send or receive data. **Pin 5 -Signal ground:** This is the common ground connection for all signals. **Pin 6 -Data set ready (DSR):** The DCE tells the DTE it is connected and ready to receive. **Pin 7 -Request to send (RTS):** This signal from the DTE tells the DCE it is ready to transmit **Pin 8 -Clear to send (CTS):** This line from the DCE tells the DTE it is ready to receive data. **Pin 9 -Ring indicator (RI):** This line was used in older modem connections but isn't used anymore.	
d)	**Draw the interfacing diagram of 4×4 matrix keyboard with 89C51 microcontroller.**	**4M**

	Ans:		**4M**
	e)	**Compare general purpose operating system and RTOS.**	**4M**

Ans:

General OS	RTOS
1. It is used for general universal application	1. It is used for dedicated electronic application
2. There is no task deadline	2. There is a task deadline in RTOS
3. The time response of OS is not deterministic	3. The time response of RTOS is deterministic.
4. Depending upon application we cannot customize the OS	4. Depending upon application, we can customize the RTOS.
5. It does not optimize the memory resources	5. It optimizes the memory resources.
6. It is normally stored in Hard Disk	6. It is normally started in semiconductor memory like EEPROM, Flash EEPROM
7. The application are complied and linked separately from the operating system	7. The applications are usually linked with the RTOS

(Any 4 Points)

(1M each point)

f) State any four design metrics of an embedded system. **4M**

Ans:

ANY 4 POINT S 4 M

1. *Power Dissipation*:For battery operated system this is important feature. Examples are mobile phone or digital camera where if power dissipation is small battery needs to be recharge less frequently.

2. *Unit cost:* the monetary cost of manufacturing each copy of the system, excluding NRE cost.

3. *NRE cost (Non-Recurring Engineering 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").

4. *Size:* the physical space required by the system, often measured in bytes for software, and gates or transistors for hardware.

5. *Memory*: RAM in KB, internal flash memory requirements in MB or GB for running

software and for data storage.

6. *Performance:* the execution time or throughput of the system.Instruction execution time in the system measures performance. Smaller execution time means higher performance. For eg.in mobile phone, voice signals are processed between antenna and speaker in 0.1s shows phone performance.

7. *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.

8. *Flexibility:* the ability to change the functionality of the system without incurring heavy NRE cost. Software is typically considered very flexible. Flexibility in design enables, without significant engineering cost, development of different versions or product or to develop advanced version later on. For example software enhancement by adding extra functions.

9. Reliability: It is measure of how much % you can rely upon the proper functioning of system. Mean Time Between Failure (MTBF) and Mean Time To Repair (MTTR) are used in determining reliability. MTBF gives the frequency of failures in hours/weeks/months.MTTR specifies how long the system is allowed to be out of order following a failure.

10. Maintainability: Deals with support and maintenance to the end user or client in case of technical issues and product failure. A more reliable system means with less maintainability.As reliability of the system increases chances of failure and non-functioning also reduces.

11. *Time-to-market:* The amount of time required to design and manufacture the system to the point the system can be sold to customers. The main contributors are design time, manufacturing time and testing time. There may be multiple players in the embedded industry who develop products of the same category (like mobile phones, portable media players etc.). If you come with new product and time to market is high competitor may take advantage of it with their product.

12. *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. If the prototype is developed faster, the actual estimated development time can be bought down.

13. *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.

14. *Safety:* the probability that the system will not cause harm. It deals with possible damages that can happen to the operators, public and the environment due to breakdown of embedded system, or due to the emission of radioactive or hazardous materials from embedded products. Safety analysis is a must in product engineering to evaluate the anticipated damages and determine best course of action.

15. Operating System : Embedded system should embed a real time operating system (RTOS), which supervises the application software tasks running on the hardware and organizes the accesses to system resources according to priorities and timing constraints of tasks in the system.

Q. No.	Sub Q. N.	Answers	Marking Scheme
3		**Attempt any FOUR of the following:**	**16- Total Marks**
	a)	**Compare between CAN and I2C protocols on the basis of following points:** **(i) Data transfer** **(ii) Number of field** **(iii) Addressing bit** **(iv) Application**	**4M**
	Ans:	(see table below)	**Each point: 1Mark**

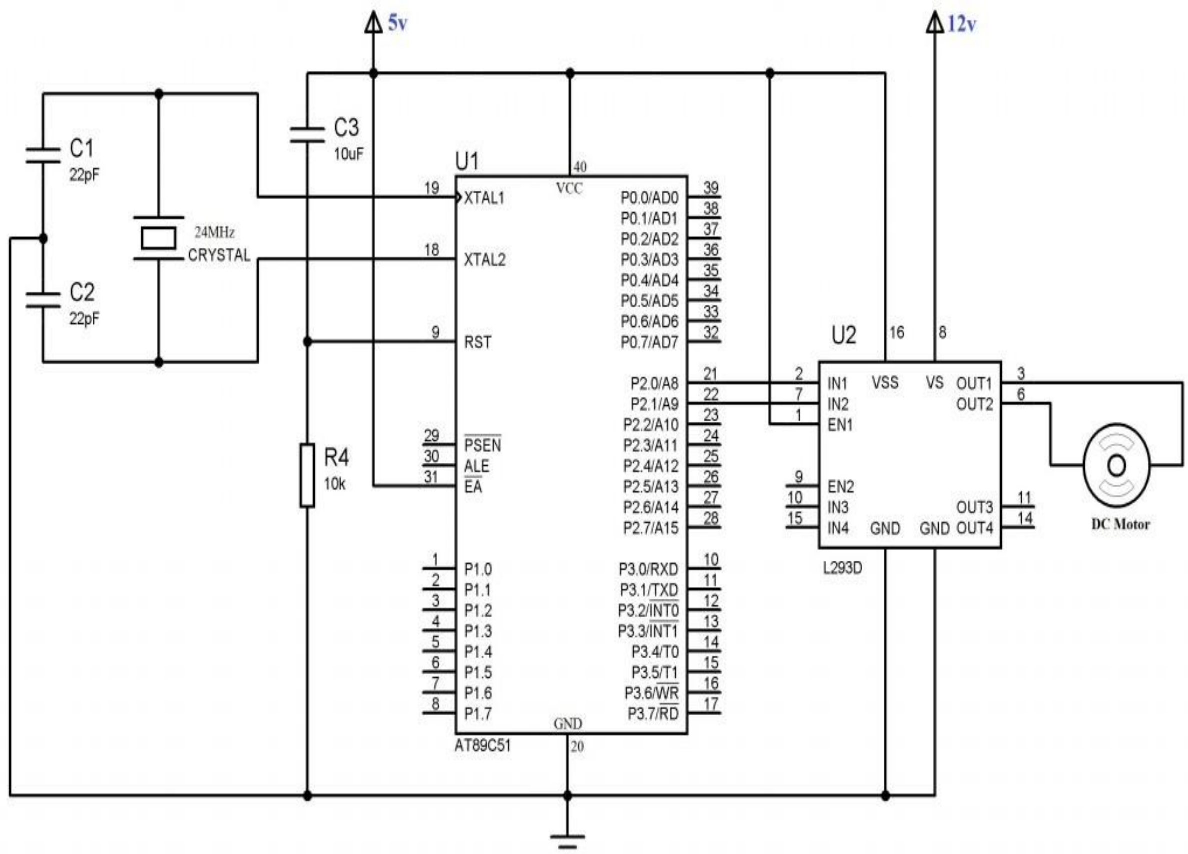
	CAN	I2C
Data transfer	Asynchronous with 250 Kbps up-to 1Mbps.	Synchronous with 3speeds 100Kbps, 400 Kbps and 3.4Mbps
Number of field	08 [including 7 bits of frame end and 3 bits of inter frame gap].	07
Addressing bit	11 bit	7-bit 0r 10 bit address
Application	Copiers, Telescopes, Medical instruments, Elevator	To interface devices like watch dog, Flash and RAM memory,

		controllers, Automobile industry | Real time clock , Microcontrollers	
b)	**If the content of ACC = 0 × 06 and P1 =0× D2. State the result after execution of following statements independently:** **(i) Result = ACC and P1** **(ii) Result = ACC : P1** **(iii) Result = ACC ^ P1** **(iv) Result = ~ P1**	**4M**
Ans:	**I. Result = ACC and P1 = 02H** II. **Result = ACC | P1 = D6H** **III. Result = ACC ^ P1 = D4H** **IV. Result = ~ P1 = 2DH** Note - in bit ii above, consider **:** as OR gate |. If student has answered considering like this or attempted to solve the bit, give appropriate marks.	**Each result:1 Mark**
c)	**State the methods of task synchronization. Describe semaphore with suitable example.**	**4M**
Ans:	The methods of task synchronization are: • Semaphore • Message queue. • Mutual exclusion. • Dead lock. • Mailboxes. • Message Queues. Semaphores: • It is a system of sending message by using flags. Multiple concurrent threads of execution with an application must be able to synchronize their execution & co-ordinate mutually exclusive access to shared resources. • To fulfill this requirement RTOS kernel provides a semaphore object that one or more threads of execution can acquire or release for the purpose of synchronization or mutual exclusion. Semaphore is like a key that allows a test to carry out some operation or to access a 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.	State or List of methods -1M Descript ion-2M Example - 1M

		• A 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. • A kernel supports many different **types of semaphores :**
		• **Binary**: Binary semaphores are used for both mutual exclusion and synchronization purposes. A binary semaphore is used to control sharing a single resource between tasks.
		• **Counting**: it is a semaphore that increments when an IPC is given by a task. It decrements when a waiting task unblocks and starts running.
		• **Mutex or Mutually Exclusion semaphore**: In this a mutex key is used to lock the shared resource, if it is acquired by the task, so that any other task cannot acquire until it is released.
		
		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;
		}	

	d)	**List advantages and disadvantages of embedded system.**	**4M**
	Ans:	Advantages of an embedded systems (any two):- 1. 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. 2. Cost: The streamline make-up of most embedded system allows their parts to be smaller less expensive to produce. 3. Accessibility: If something goes wrong with certain embedded systems they can be too	(2 M :advantages , 2M: disadvantages)

		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. 4. Maintenance: Embedded systems are easier to maintain because the supplied power is embedded in the system and does not required remote maintenance. Disadvantages (any two):- 1. 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. 2. 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. 3. 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". 4. 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.	
	e)	**Draw labeled interfacing diagram to interface DC motor with microcontroller.**	**4M**

	Ans:		**Diagram : 4Marks**

Q. No.	Sub Q. N.	Answers	Marking Scheme
4	**a)**	**Attempt any THREE of the following:**	**12- Total Marks**
	(i)	**List the interrupts of 89C51 microcontroller with their vector locations and order of priority.**	**4M**

Ans:			List:2Marks Vector locations:1Mark Priorities:1Mark

Interrupt	Vector Address	Priority
IE0 / (External interrupt 0 , INT0)	0003H	1
TF0 / (Timer 0 Interrupt)	000BH	2
IE1 / (External interrupt 1 , INT1)	0013H	3
TF1 / (Timer 1 Interrupt)	001BH	4
TI or RI (serial port interrupt)	0023H	5

(ii) State any four features of Bluetooth Technology. **4M**

Ans: **Features of Bluetooth Technology:** *(Any four; Each feature: 1Mark)*

- Short range Radio Frequency at 2.4 GHz
- Point-to-point or point-to-multiple points
- Voice and Data
- Transmit through walls up to 10m
- Supports both synchronous and asynchronous services.
- Bluetooth is IEEE 802.15.1 protocol.
- Bluetooth 1.x supports data rate up to 1Mbps.
- Bluetooth 2.0 enhanced maximum data rate of 3Mbps over 100m.

(iii) Explain the meaning of Deadlock and starvation with reference to embedded system. **4M**

Ans: **Deadlock:** *(Deadlock:2Marks; Starvation:2Marks)*

A deadlock consists of a set of blocked processes, each holding a resource and waiting to acquire a resource held by another process in the set A deadlock, also called as deadly embrace, is a situation in which two threads are each unknowingly waiting for resource held by other.

Example #1

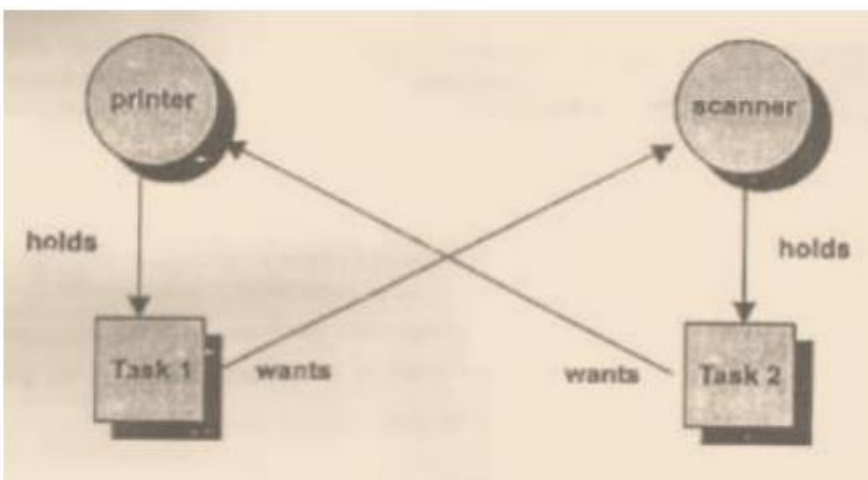
A system has 2 disk drives
P1 and P2 each hold one disk drive and each need the other one.

Example #2

Semaphores A and B, initialized to 1

P0	P1
wait (A);	wait(B)
wait (B);	wait(A)

Example #3



- Task #1 wants the scanner while holding the printer. Task #1 cannot proceed until both the printer and the scanner are in its possession.
- Task #2 wants the printer while holding the scanner. Task #2 cannot continue until it has the printer and the scanner.

Starvation:

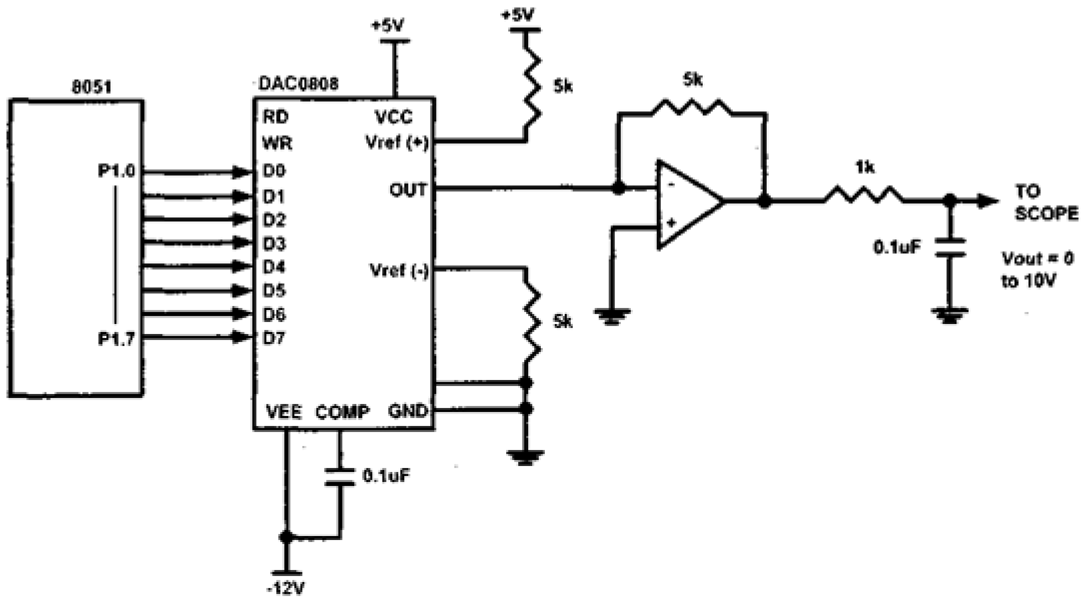
- This occurs in preemptive type of multitasking.
- This happens when a lower priority task waits for a resource, which it shares with higher priority tasks.
- If the higher priority tasks are executing continuously, the lower priority task has to wait indefinitely. This condition is called as starvation.

	(iv)	**State any four specifications of RTOS. Give any four examples of RTOS.**	**4M**
	Ans:	**Specifications of RTOS**: 1. Reliability: The RTOS is reliable, because it is available for all time and normally it does not fail to perform any function/operation. The reliability of system also depends on the hardware board support package and application code. 2. Predictability: In RTOS, the user knows within How much time period the RTOS is going to perform the task i.e. The RTOS has predictability. We can predict, determine how much time takes by RTOS. 3. Performance: The performance of RTOS is very fast so that it can fulfill all timing requirement. 4. Compactness: The RTOS provide compactness. It required less memory space for storage and hence can be used for portable application, like cellphone, ECG machine, etc. 5. Scalability: RTOS can be used in a wide variety of embedded. They must be able to scaleup or scale-down to suit the application. **Examples of RTOS:** • calculators, heart pacemakers • Electric Geyser where water temperature is controlled in real time in the industry the process parameters like temperature, flow, or pressure or status of a device (say a valve open or close) are continuously monitored and instant actions are initiated • Room Air Conditioner which adaptively controls the temperature of a room. • Electric Power System which controls power quality parameters like Frequency, Peak Voltage, Power Factor, e.t.c. • Machine vision guided robotics • Digital audio decoding, transport over a network and encoding using a simple off-the-shelf sound card (VoIP or internet radio applications).	Any four specifications 2M, (Each specification-1/2M) Any four examples 2M, (Each example-1/2M)
	b)	**Attempt any ONE of the following:**	**06- Total**

			Marks
	(i)	**Write 89C51 C language program to generate square wave of 10 KHz on pin P2.7 using timer 0 . Assume crystal frequency as 12 MHz .**	**6M**

		CALCULATIONS:
		Crystal frequency = 12 MHz
		• for 10KHz frequency calculations with 12 MHz
		• The period of the square wave = 1 / 10 KHz = 100µS
		• The high or low portion of the square wave = Time period / 2 = 100µS / 2 = 50µS.
		• Timer clock Frequency is = XTAL / 12 • = 12 MHz / 12 = 1 MHz
		• Timer clock period is = 1/ Timer Frequency = 1 / 1 MHz = 1 µSec
		• Counter = Delay / timer clock period = 50µS / 1 µSec = 50
		• Timer Reload value = Maximum Count – Counter = 65536 – 50 = (65486)d
		• Timer Reload value in HEX =(FFCE) h.
		• TL0 = 0xCE and TH0 = 0xFF.
		
		//C language program to generate square wave over Port Pin P2.7 using timer0
		
		#include<reg 51.h>
		Void T0M1delay (void); //Timer 0, Mode 1(16 bit timer)
		SBIT OUTPUT P2^7; // Initialize Port pin P2.7 as output
		Void main ()
		{
		While (1)
		{
		OUTPUT= ~ OUTPUT; // toggle P2.7
		T0M1delay (); // delay of 50µS
		}
		}
		void T0M1delay () // Timer 0, Mode 1(16 bit timer) - delay of 50µS
		{
		TMOD= 0x01; // Timer 0, Mode 1(16 bit timer)
		TL0 = 0xCE; //Load TL0 = CEH
		TH0 = 0xFF; //Load TH0 = FFH
		TR0 = 1; //Run the timer 0
		while (TF0 = = 0); // Wait for TF0 to overflow
		TR0 = 0; //Stop the timer 0
		TF0 = 0; //Clear TF0
		}
			Calculation 01M program 04M Comments:1Mark

	(ii)	**Draw interfacing diagram of DAC with 89C51 microcontroller. Write a program in " C" language to generate triangular wave.**	**6M**

	Ans:	**Interfacing diagram of DAC with 89C51 microcontroller:**
		
		" C" language to generate triangular wave:
		#include<reg51.h>
		unsigned char d;
		void main(void)
		{
		while(1)
		{
		for(d=0; d<255; d++)
		{
		P1 = d;
		}
		for(d=255; d>0; d--)
		{
		P1 = d;
		}
		}
		}
		(any other relevant logic s can be use)
			Diagram 3 Marks
			Program :3Marks

Q. No.	Sub Q. N.	Answers	Marking Scheme
5.		**Attempt any FOUR of the following:**	**16- Total Marks**
	a)	**Explain JTAG in brief.**	**4M**
	Ans:	**Joint Test Action Group (JTAG) :**Advances in silicon design such as increasing device density and, more recently, BGA packaging has reduced the efficacy of traditional testing methods. In order to overcome these problems, some of the world's leading silicon manufacturers combined to form the Joint Test Action Group. The findings and recommendations of this group were used as the basis for the Institute of Electrical and Electronic Engineers (IEEE) standard 1149.1 : Standard Test Access Port and Boundary Scan Architecture. This standard has retained its link to the group and is commonly know by the acronym JTAG. It was initially devised by electronic engineers for testing printed circuit boards (PCB) using boundary scan. Today it is widely used for IC debug ports. Embedded systems development relies on debuggers communicating with chips with JTAG to perform operations like single stepping & break pointing. An ICE (In Circuit Emulator) uses JTAG as the transport mechanism to access on-chip debug modules inside the target CPU. These modules let software developers debug the software of an embedded system directly at the machine instruction level when needed. **Boundary Scan :** The main advantage offered by utilizing boundary scan technology is the ability to set and read the values on pins without direct physical access.	**Diagram : 1 M, Explanation :3M**

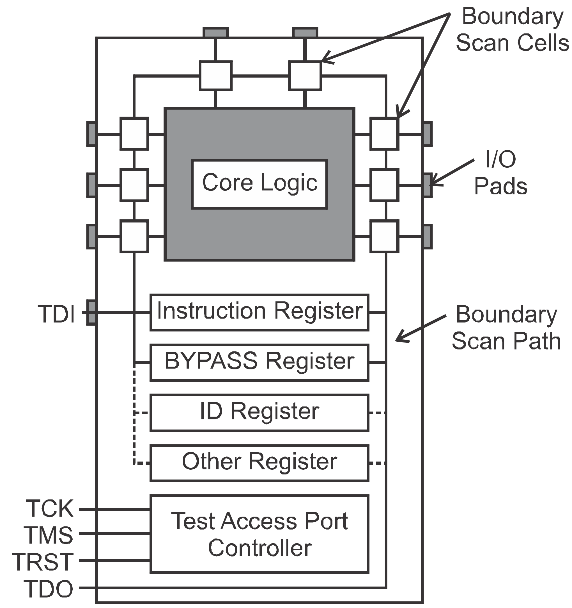


Fig. : Schematic Diagram of a JTAG enabled device

The process of boundary scan can be most easily understood with reference to the schematic diagram shown in figure.

All the signals between the device's core logic and the 'pins' are intercepted by a serial scan path known as the Boundary Scan Register (BSR). In normal operation these boundary scan cells are invisible. However, in test mode the cells can be used to set and/or read values : in external mode these will be the values of the 'pins'; in 'internal' mode these will the values of the core logic.

	b)	**Compare synchronous and asynchronous communication. (any four points)**	**4M**

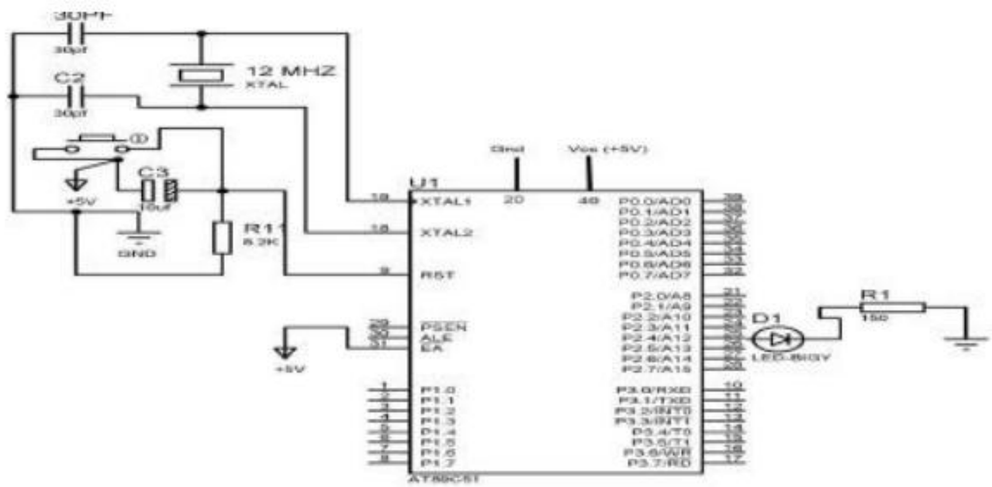
	Ans:		Any four relevant points- 1M each

Sr. No.	Synchronous	Asynchronous
1	Same clock pulse is required at transmitter and receiver	Different clock pulse is required at transmitter and receiver
2	Used to transfer group of Character	Used to transfer one character at a time
3	Synchronous character is required.	Synchronous character Is required.
4	No start and stop signals are Required	Start and stop signals are required.
5	Data transmission rate is greater then or equal to 20Kbps	Data transmission rate is less then or equal to 20 Kbps.
6	It is less reliable	It is more reliable
7	Error checking is not possible	Error checking is possible with parity bit.

	c)	**Draw labeled diagram to interface LED to P2.1 of 89C51. Write a language program to turn on and off this LED after some delay.**	**4M**

Ans:

Diagram : 2M, Program :2 M)



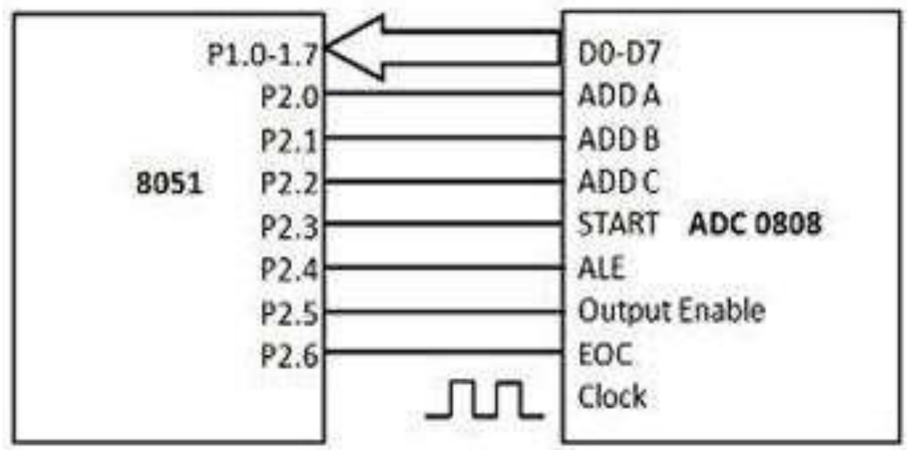
(LED should be connected to P2.1)

'C' program to turn ON and OFF the LED :

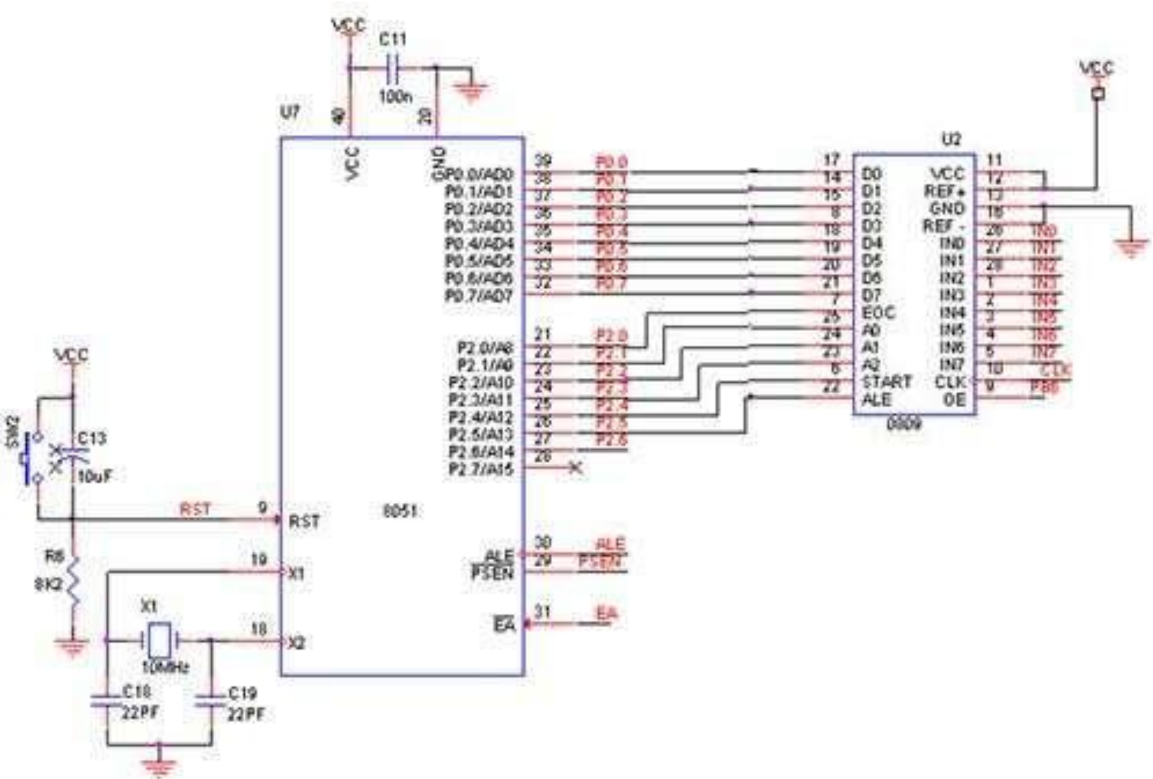
```
#include<reg51.h>
sbit LED=P2^1;                  // Define P1.7 as output LED
void main(void)
{
unsigned int i;                 // Variable declaration
LED=0;                          // Output logic 0 on port pin p1.7
while(1)
{
LED=0;                          // Output logic 0 on port pin p1.7
for(1=0;i<=1000;i++);           // Delay loop
LED=1;                          // Output logic 1 on port pin p1.7
for(1=0;i<=1000;i++);           // Delay loop
```

		} } **(any other relevant logic s can be use)**	
	d)	**Explain inter process communication in brief. State various inter process communication methods.**	**4M**
	Ans:	**Interprocess communication (IPC):** i. Interprocess communication (IPC) is a set of programming interfaces that allow a programmer to coordinate activities among different program processes that can run concurrently in an operating system. ii. This allows a program to handle many user requests at the same time. iii. Since even a single user request may result in multiple processes running in the operating system on the user's behalf, the processes need to communicate with each other. . The IPC interfaces make this possible. . Each IPC method has its own advantages and limitations so it is not unusual for a single program to use all of the IPC methods. **IPC methods:** 1 Pipes -Named pipes and un named pipes. 2 Message queue 3 Semaphores 4 Shared memory 5 Sockets	**Explanation :2M , Methods -2M**

	e)	**Describe the program downloading tool ISP/IAP.**	**4M**
	Ans:	**In system Programming (ISP) :** • Programming is done "within the system" i.e. firmware (IEEE 1394 is wired isochronous high speed serial communication bus) is embedded into the target device without removing it from the target board. • The target device must have an ISP support. No additional hardware is required. • Chips supporting ISP generates the necessary programming signal internally using chip's supply voltage. • Target board is interfaced to utility program running on PC through Serial/Parallel/USB port. • **Serial Protocol for ISP :** JTAG, SPI(serial peripheral interface. **In Application Programming (IAP) :** • IAP is the technique running on the target device for modifying a selected portion of the code memory. • This technique is not used for first time embedding of user written firmware. • It modifies the program code memory under the control of embedded applications. **Examples :** Updating calibration data, look up tables, Boot ROM etc. in code memory **(Note: since the question ask ISP/IAP, so marks must be given if any one is written)**	**4 Mark**
	f)	**Draw the interfacing diagram of ADC with microcontroller.**	**4M**
	Ans:		**Any one diagram 4M**


P1.0-1.7
P2.0
P2.1
8051
P2.2
P2.3
P2.4
P2.5
P2.6
D0-D7
ADD A
ADD B
ADD C
START
ADC 0808
ALE
Output Enable
EOC
Clock

OR

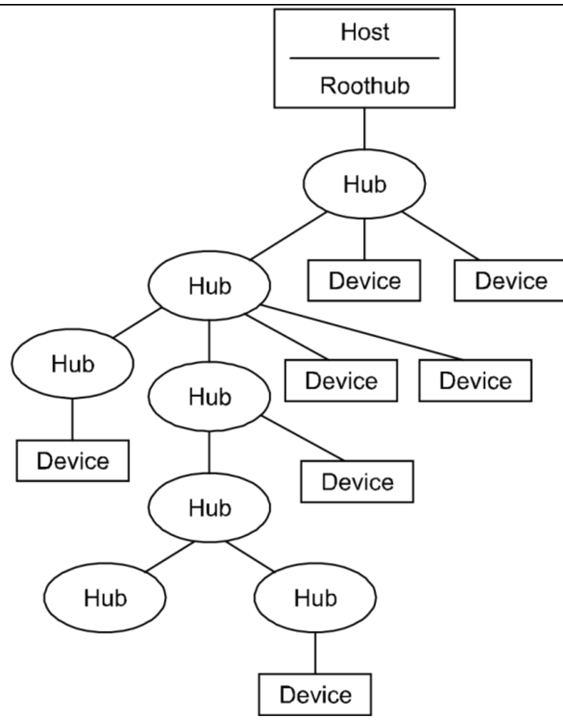

VCC
C11
100n
U7
8051
U2
0809
RST
X1
X2
ALE
PSEN
EA
10MHz
C18
22PF
C19
22PF
C13
10uF
R6
8K2

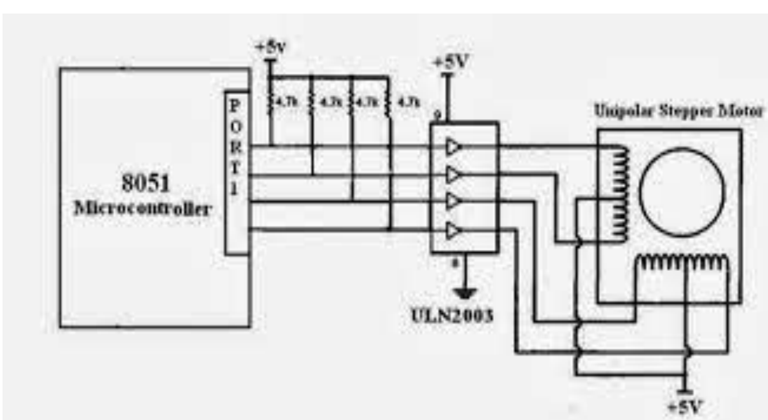
Q. No.	Sub Q. N.	Answers	Marking Scheme
6.		**Attempt any FOUR of the following:**	**16- Total Marks**
	a)	**Compare between assembly language program with an embedded C with reference to following points:** **(i) Execution time** **(ii) Time for loading** **(iii) Hex file size** **(iv) Debugging**	**4M**
	Ans:	(see table below)	**Each point 1 M**

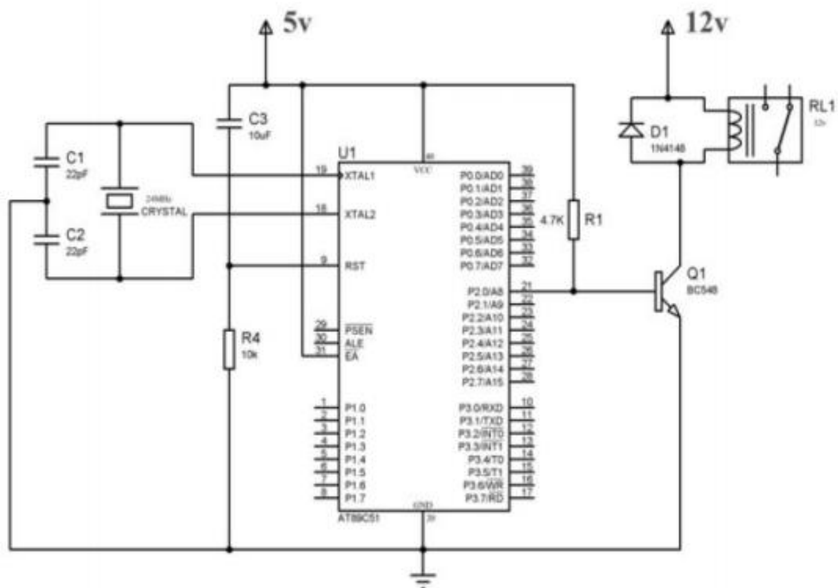
Parameter	ALP	Embedded C
(1) Execution time	Faster (Less execution time required).	Slower (More execution time required).
(2) Time for coding	More time is required for coding.	Less time required for coding and code is efficient.
(3) Hex file size	Less.	More.
(4) Debugging	No easy.	Easy.

Q. No.	Sub Q. N.	Answers	Marking Scheme
	b)	**Draw and explain USB protocol.**	**4M**
	Ans:	**Explanation**: Any computer that you buy today comes with one or more **Universal Serial**	**explanation 2M, Diagram**

		Bus connectors. These USB connectors let you attach mice, printers and other accessories to your computer quickly and easily. The operating system supports USB as well, so the installation of the device drivers is quick and easy, too. Compared to other ways of connecting devices to your computer (including parallel ports, serial ports and special cards that you install inside the computer's case), USB devices are incredibly simple. • The USB is based on a so-called tiered star topology in which there is a single host and up to 127 slave devices. • The host controller is connected to a hub within the PC which allows a number of attachment points (ports). • A further hub can be plugged into each of these attachments and so on. However, there are limitations on this expansion. • A maximum of 127 devices may be connected. This is because the address field in a packet is 7 bits long and the address 0 cannot be used as it has special significance. • A device can be plugged into a hub and that hub can be plugged into another hub and so on. However, the maximum number of tieres permitted is six. • The length of any cable is limited to 5 meters. So, USB is intended as a bus for devices near to PC. For applications requiring distance from the PC, another form of connection is needed such as Ethernet. • Host is the master. So, all communications are initiated by the host. There can be no communication directly between USB devices.	**:2M**



c)	**Draw the interfacing diagram of stepper motor with microcontroller.**	**4M**
Ans:		

	d)	**Draw the interfacing of relay with 89C51 microcontroller. Write C language program to make relay on-off after certain delay**	**4M**
	Ans:	 Program Note : If student has written program to on-off LED continuously, marks can be given	


```
 #include<reg.51.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)
```

		<pre>{
		unsigned int x,y;
		for(x=0; x,itime;x++)
		for(y=0;y<1275;y++);
		}</pre>
		(any other relevant logic s can be use)	

	e)	**Write 89C51 C program to toggle all bits of part P_0 continuously with a 200 millisecond delay.**	**4M**

	Ans:	<pre>#include <reg51.h>
		void delay (unsigned int);
		void main (void)
		{
		while(1) //repeat loop
		{
		P0=0xff; //toggle all bits of port2
		delay (200); //add delay
		P0=0x00; //toggle all bits of port2
		delay (200); //add delay
		}
		}
		void delay (unsigned int i)
		{
		Unsigned int x, y;
		for(x=0; x<i; x++)
		for (y=0; y<1275; y++);
		}</pre>
		(any other relevant logic s can be use)
			4M