1 a) Define Profiling and cycle counting.

b) What is Pipeline Interlock? Explain the following cases with an example using ARM9TDMI pipeline stages.

a. Case with One-cycle interlock caused by load use.

b. Case with branch instruction where processor must flush the pipeline when jumping to a new address.

ANS 1A

The first stage of any optimization process is to identify the critical routines and measure their current performance. o A profiler is a tool that measures the proportion of time or processing cycles spent in each subroutine. You use a profiler to identify the most critical routines. o A cycle counter measures the number of cycles taken by a specific routine. You can measure your success by using a cycle counter to benchmark a given subroutine before and after an optimization.

PG 4 MOD 2 PART B

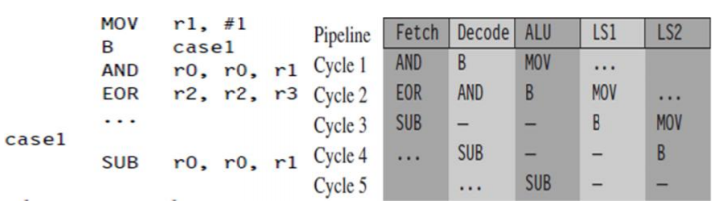
ANS 1B

a)After an instruction has completed the five stages of the pipeline, write the result of the register file. Note that PC points to the address of the instruction being fetched. The ALU is executing the instruction that was originally fetched from the address PC - 8 in parallel with fetching the instruction at address PC. The following example shows how the cycle timings change because an earlier instruction must complete a stage before the current instruction can progress down the pipeline. If an instruction requires the result of a previous instruction that is not available, then the processor stalls. This is called a Pipeline Hazard or Pipeline Interlock. Example1: Case with No Interlock



This instruction pair takes two cycles.

b)Example 4: Why a branch instruction takes three cycles? The processor must flush the pipeline when jumping to a new address



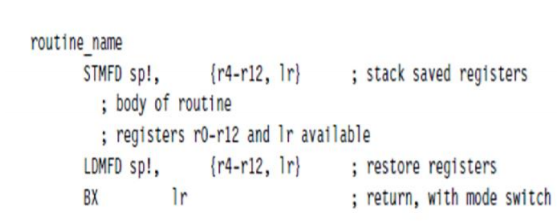
Three executed instructions take a total of five cycles. The MOV instruction executes on the first cycle. In the second cycle, the branch instruction calculates the destination address. This causes the core to flush the pipeline and refill it using this new pc value. The refill takes two cycles. Finally, the SUB instruction executes normally. The pipeline drops the two instructions following the branch when the branch takes place.

2 a) Explain Register Allocation in detail.

b) Explain conditional execution with an example program.

ANS 2A

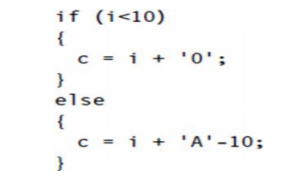
REGISTER ALLOCATION 14 of the 16 visible ARM registers can be used to hold general-purpose data. The other two registers are: stack pointer (r13), and the program counter, (r15). For a function to be ATPCS compliant it must preserve the callee values of registers r4 to r11. ATPCS also specifies that the stack should be eight-byte aligned; therefore we must preserve this alignment if calling subroutines. Use the following template for optimized assembly routines requiring many registers: r12 is also stacked just to keep the stack eight-byte aligned. o Address starts from 0 and ends at 28 for r4-r11 (28 is not multiple of 8) o Address starts from 0 and ends at 32 for r4-r12 (32 is multiple of 8) r12 need not be stacked if your routine doesn’t call other ATPCS routines For ARMv5 and above you can use the preceding template even when being called from Thumb code. For ARMv4T processor, i.e) if the routines are called from Thumb code, then modify the template as follows: P a g e | 11 MICROCONTROLLERS AND EMBEDDED SYSTEMS (18CS44) HEERAH D, ASST. PROF., DEPT.OF CSE implementation. It is assumed that we can read upto 2 characters beyond the end of the input string, which may not be true if the string is right at the end of the available RAM, where reading of the end will cause a data abort. Also, performance can be slower for very short strings because (1) stacking lr causes additional function call overhead and (2) the routine may process upto 2 characters pointlessly, before discovering that they lie beyond the end of the string. This form of scheduling by unrolling is used for time critical parts of an application where you know the data size is large. If the size of the data is also known at compile time we can remove the problem of reading beyond the end of the array. REGISTER ALLOCATION 14 of the 16 visible ARM registers can be used to hold general-purpose data. The other two registers are: stack pointer (r13), and the program counter, (r15). For a function to be ATPCS compliant it must preserve the callee values of registers r4 to r11. ATPCS also specifies that the stack should be eight-byte aligned; therefore we must preserve this alignment if calling subroutines. Use the following template for optimized assembly routines requiring many registers: r12 is also stacked just to keep the stack eight-byte aligned. o Address starts from 0 and ends at 28 for r4-r11 (28 is not multiple of 8) o Address starts from 0 and ends at 32 for r4-r12 (32 is multiple of 8) r12 need not be stacked if your routine doesn’t call other ATPCS routines For ARMv5 and above you can use the preceding template even when being called from Thumb code. For ARMv4T processor, i.e) if the routines are called from Thumb code, then modify the template as follows: P a g e | 11 MICROCONTROLLERS AND EMBEDDED SYSTEMS (18CS44) HEERAH D, ASST. PROF., DEPT.OF CSE implementation. It is assumed that we can read upto 2 characters beyond the end of the input string, which may not be true if the string is right at the end of the available RAM, where reading of the end will cause a data abort. Also, performance can be slower for very short strings because (1) stacking lr causes additional function call overhead and (2) the routine may process upto 2 characters pointlessly, before discovering that they lie beyond the end of the string. This form of scheduling by unrolling is used for time critical parts of an application where you know the data size is large. If the size of the data is also known at compile time we can remove the problem of reading beyond the end of the array. REGISTER ALLOCATION 14 of the 16 visible ARM registers can be used to hold general-purpose data. The other two registers are: stack pointer (r13), and the program counter, (r15). For a function to be ATPCS compliant it must preserve the callee values of registers r4 to r11. ATPCS also specifies that the stack should be eight-byte aligned; therefore we must preserve this alignment if calling subroutines. Use the following template for optimized assembly routines requiring many registers: r12 is also stacked just to keep the stack eight-byte aligned. o Address starts from 0 and ends at 28 for r4-r11 (28 is not multiple of 8) o Address starts from 0 and ends at 32 for r4-r12 (32 is multiple of 8) r12 need not be stacked if your routine doesn’t call other ATPCS routines For ARMv5 and above you can use the preceding template even when being called from Thumb code. For ARMv4T processor, i.e) if the routines are called from Thumb code, then modify the template as follows:



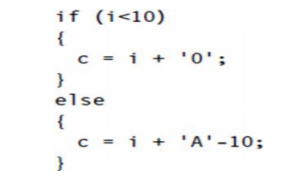
Allocating Variables to Register Members It is best to use alternate names to the registers, rather than explicit register numbers. Benefits of using alternate names: o Allows you to change the allocation of variables to register numbers easily. o Allows using different register names for the same physical register number when their use doesn’t overlap. o Register names increase the clarity and readability of optimized code. For the most part ARM operations are orthogonal with respect to register number. In other words, specific register numbers do not have specific roles. If you swap all occurrences of two registers Ra and Rb in a routine, the function of the routine does not change. However, there are several cases where the physical number of the register is important: Argument registers.The ATPCS convention defines that the first four arguments to a function are placed in registers r0 to r3. Further arguments are placed on the stack. The return value must be placed in r0. Registers used in a load or store multiple. Load and store multiple instructions LDM and STM operate on a list of registers in order of ascending register number. If r0 and r1 appear in the register list, then the processor will always load or store r0 using a lower address than r1 and so on. Load and store double word. The LDRD and STRD instructions introduced in ARMv5E operate on a pair of registers with sequential register numbers, Rd and Rd+1. Furthermore, Rd must be an even register number

ANS 2B

CONDITIONAL EXECUTION The processor core can conditionally execute most ARM instructions. By combining conditional execution and conditional setting of the flags, it is possible to implement simple if statements without any need for branches. This improves efficiency since branches can take many cycles and also reduces code size. Example 1: Converts an unsigned integer 0 ≤ i ≤ 15 to a hexadecimal character c:



We can write this in assembly using conditional execution rather than conditional branches:



The sequence works since the first ADD does not change the condition codes. The second ADD is still conditional on the result of the compare. Section 6.3.1 shows a similar use of conditional execution to convert to lowercase. Conditional execution is even more powerful for cascading conditions

3 a) What is an Embedded system?

b) Explain the main elements of an embedded system with a neat digram.

c) Explain the classification of Embedded systems based on complexity and Performance.

ANS 3A

• An embedded system is an electronic/electro-mechanical system designed to perform a specific function and is a combination of both hardware and firmware (software).

• Every embedded system is unique and the hardware as well as the firmware is highly specialized to the application domain.

ANS 3B

A TYPICAL EMBEDDED SYSTEM

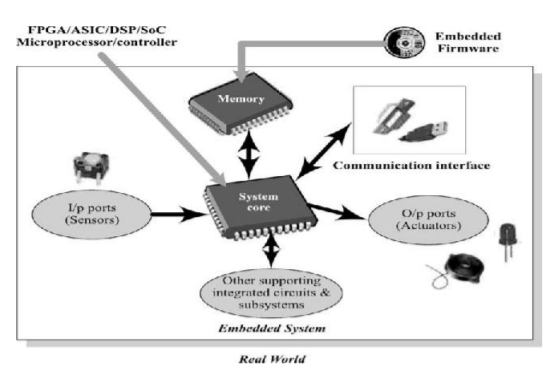


Fig: Elements of an Embedded System

• A typical embedded system contains a single chip controller which acts as the master brain of the system the controller can be a microprocessor or a microcontroller or a field Programmable Gate array device for a digital signa Application Specific User Interface

• These are embedded systems with application-specific user interfaces like Buttons, switches, keypad, lights, bells, display units, etc.

• Mobile phone is an example for this.

• In mobile phone the user interface is provided through the keypad, graphic LCD module, system speaker, vibration alert, etc. A TYPICAL EMBEDDED SYSTEM Fig: Elements of an Embedded System

. • Embedded hardware/software systems are basically designed to regulate a physical variable or to manipulate the state of some devices by sending some control signals to the actuators or devices connected to the output port of the system, in response to the input signals provided by the end users or sensor which are connected to the input ports. Hence an embedded system can be viewed as a reactive system.

• Keyboards push button switches etc., are examples for common user interface input devices whereas LEDs, liquid crystal displays, piezoelectric buzzers etc., examples for common user interface output devices for a typical embedded system. These I/O user interface solely depends on the type of application for which the embedded system is designed.

• The ROM memory of the system is responsible for holding the control algorithm and other important configuration details which is secured from the intruders. The system might also require RAM for performing arithmetic operation or control algorithm execution. In a controller based embedded system, the controller may contain internal memory for storing the control algorithms and it may be EEPROM or flash memory varying from a few kilobytes to megabytes.

ANS 3C

Classification Based on Complexity and Performance:

• Small-Scale Embedded Systems • Medium-Scale Embedded Systems

• Large-Scale Embedded Systems Small-Scale Embedded Systems

• Simple in application needs and the performance requirements are not time critical. E.g.: An electronic toy

• Usually built around low performance and low cost 8-bit or 16-bit microprocessors/microcontrollers.

• May or may not contain an operating system for its functioning. Medium-Scale Embedded Systems

• Slightly complex in hardware and firmware (software) requirements.

• Usually built around medium performance, low cost 16-bit or 32- bit microprocessors/microcontrollers or digital signal processors. Usually contain an embedded operating system (either general purpose or real time operating system)for functioning. Large-Scale Embedded Systems

• Highly complex in hardware and firmware (software) requirements.

• They are employed in mission critical applications demanding high performance.

• Usually built around high performance 32-bit or 64-bit RISC processors/controllers or Reconfigurable System on Chip (RSoC) or multi- core processors and programmable logic devices

• May contain multiple processors/controllers and co-units/hardware accelerators for offloading the processing requirements from the main processor of the system.

• Decoding/encoding of media, cryptographic function implementation, etc. Are examples of processing requirements which can be implemented using a co- processor/hardware accelerator

• Usually contain a high performance real time operating system (RTOS) for task scheduling, prioritization and management.

4 a) Write a short note on PLDs.

b) Differentiate between Embedded and General computing system.

c) Explain the Purpose of Embedded system with respect to Data Processing.

ANS 4A

Programmable Logic Devices:

Programmable Logic Devices:

• Logic devices provide specific functions, including device-to-device interfacing, data communication, signal processing, data display, timing and control operations, and almost every other function a system must perform.

• Logic devices can be classified into two broad categories—fixed and programmable.

• The circuits in a fixed logic device are permanent, they perform one function or set of functions—once manufactured, they cannot be changed.

• Programmable Logic Devices (PLDs) offer customers a wide range of logic capacity, features, speed, and voltage characteristics and these devices can be re- configured to perform any number of functions at any time.

o Designers use inexpensive software tools to quickly develop, simulate, and test their designs. o Then, a design can be quickly programmed into a device, and immediately tested in a live circuit.

• There are no NRE costs and the final design is completed much faster than that of a custom, fixed logic device.

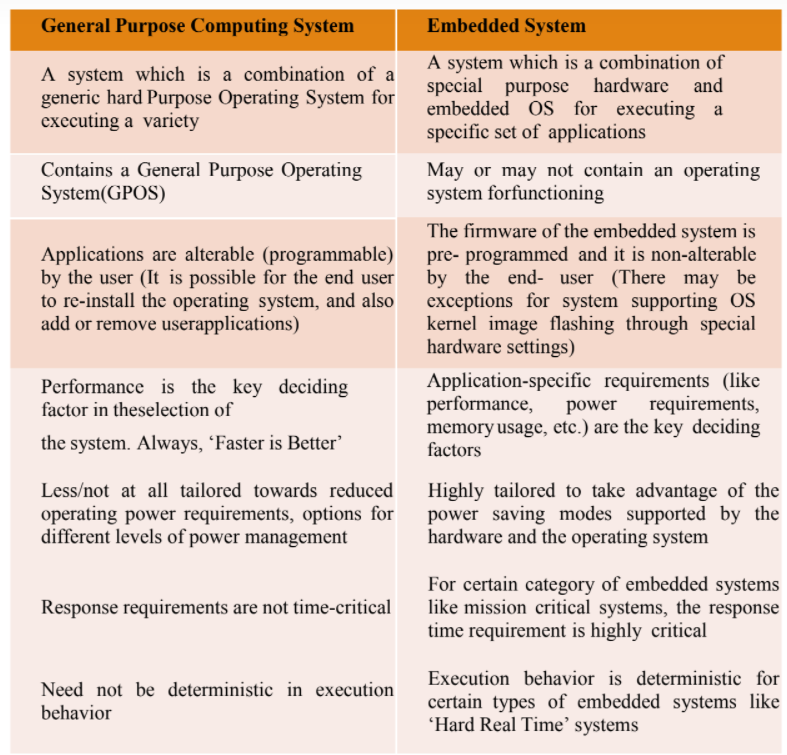
• Another key benefit of using PLDs is that during the design phase customers can change the circuitry as often as they want until the design operates to their satisfaction.

o PLDs are based on re-writable memory technology to change the design, the device is simply reprogrammed.

• Once the design is final, customers can go into immediate production by simply programming as many PLDs as they need with the final software design file.

• The two major types of programmable logic devices are Field Programmable Gate Arrays (FPGAs) and Complex Programmable Logic Devices (CPLDs).

ANS 4B



ANS 4C

Data (Signal) Processing

• The data (voice, image, video, electrical signals and other measurable quantities) collected by embedded systems may be used for various kinds of data processing.

• Embedded systems with signal processing functionalities are employed in applications demanding signal processing like speech coding, synthesis, audio video codec, transmission applications, etc



5 a) Write a short note on Static RAM and Dynamic RAM.

b) Explain the working mechanism of Matrix Keyboard with an example.

ANS 5A

Static RAM (SRAM):

• Static RAM stores data in the form of voltage.

• They are made up of flip-flops.

• Static RAM is the fastest form of RAM available.

o Fast due to its resistive networking and switching capabilities.

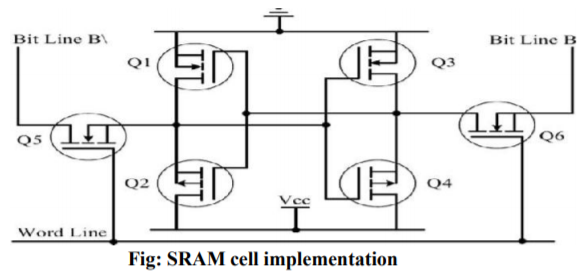


Fig: SRAM cell implementation • In typical implementation, an SRAM cell (bit) is realized using six transistors (or 6 MOSFETs).

• Four of the transistors are used for building the latch (flip-flop) part of the memory cell and two for controlling the access.

• In its simplest representation an SRAM cell can be visualized as shown in the figure below:

• This implementation in its simpler form can be visualised as two cross-coupled inverters with read/write control through transistors.

• The four transistors in the middle form the cross-coupled inverters.

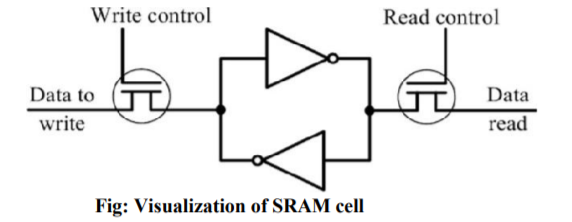


Fig: Visualization of SRAM cell

• The access to the memory cell is controlled by Word Line, which controls the access transistors (MOSFETs) Q5 and Q6.

• The access transistors control the connection to bit lines B & B\.

• In order to write a value to the memory cell, apply the desired value to the bit control lines (For writing 1, make B = 1 and B\ = 0; For writing 0, make B = 0 and B\ = 1) and assert the Word Line (Make Word line high).

• This operation latches the bit written in the flip-flop

• For reading the content of the memory cell, assert both B and bit lines to 1 and set the Word line to 1.

• The major limitations of SRAM are low capacity and high cost.

Dynamic RAM (DRAM):

• Dynamic RAM stores data in the form of charge.

• They are made up of MOS transistor gates.

• Advantages – high density and low cost compared to SRAM.

• Disadvantage – since the information is stored as charge it gets leaked off with time and to prevent this they need to be refreshed periodically.

• Special circuits called DRAM controllers are used for the refreshing operation.

• The refresh operation is done periodically in milliseconds interval.

• The MOSFET acts as the gate for the incoming and outgoing data whereas the capacitor acts as the bit storage unit.

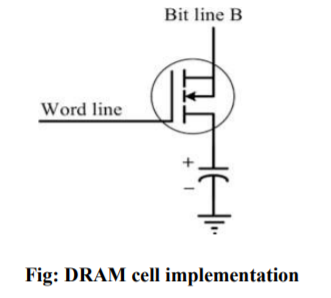


Fig: DRAM cell implementation

ANS 5B

Keyboard

• Keyboard is an input device for user interfacing.

• If the number of keys required is very limited, push button switches can be used and they can be directly interfaced to the port pins for reading.

• Matrix keyboard is an optimum solution for handling large number of key requirements.

• Matrix keyboard greatly reduces the number of interface connections.

• Matrix keyboard connects the keys in a row column fashion

• For example, for interfacing 16 keys, in the direct interfacing technique 16 port pins are required, where as in the matrix keyboard only 4 columns and 4 rows are

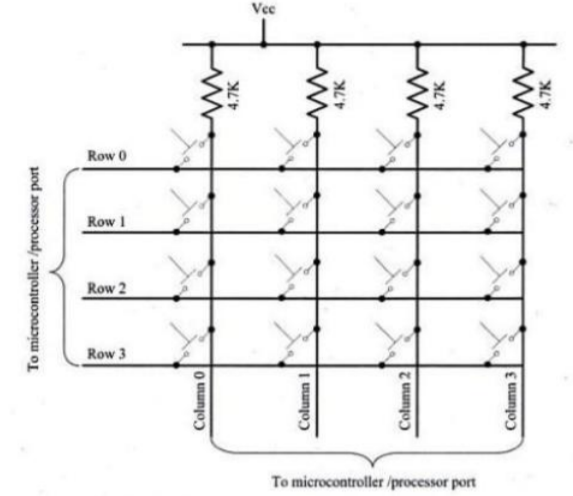


Fig: Matrix keyboard interfacing

• The 16 keys are arranged in a 4\*4 matrix. The following figure illustrates the connection of keys in a matrix keyboard.

• The key press in matrix keyboard is identified with row-column scanning technique where each row of the matrix is pulled low and the columns are read.

• After reading the status of each columns corresponding to a row is pulled high and the next row is pulled low and the status of the column are read.

• When a row is pulled low and if a key connected to the row is pressed, reading the column to which the key is connected will give logic 0.

• Since the keys are mechanical devices, there is a possibility for de-bounce issues, which may give triple key press effect for a single key press.

• The techniques to prevent from this de-bouncing issues is: o Hardware key de-bouncer o Software key de-bouncer: on detection of a key press the key is read again after de-bounce delay. If the key press is genuine one the static key press will remain as “pressed”.

6 a) Explain with neat diagram the classification of ROM memory.

b) Explain the working of 7 segment LED display

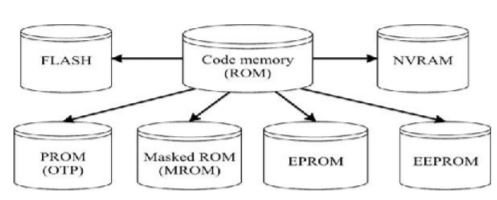
ANS 6A

Program Storage Memory (ROM)

• The program memory or code storage memory of an embedded system stores the program instructions

• The code memory retains its contents even after the power is turned off. It is generally known as non-volatile storage memory.

• It can be classified into different types as shown:



Masked ROM (MROM):

• Masked ROM is a one-time programmable device.

• Masked ROM makes use of the hardwired technology for storing data.

• The device is factory programmed by masking and metallization process at the time of production itself, according to the data provided by the end user.

• Advantage – low cost for high volume production.

• Limitation - inability to modify the device firmware against firmware upgrades. o Since the MROM is permanent in bit storage, it is not possible to alter the bit information.

• Different mechanisms are used for the masking process of the ROM, like o Creation of an enhancement or depletion mode transistor through channel implant.

o By creating the memory cell either using a standard transistor or a high threshold transistor.

• In the high threshold mode, the supply voltage required to turn ON the transistor is above the normal ROM IC operating voltage.

• This ensures that the transistor is always off and the memory cell stores always logic 0. Programmable Read Only Memory (PROM) / (OTP):

• One Time Programmable Memory (OTP) or PROM is not pre-programmed by the manufacturer.

o The end user is responsible for programming these devices.

• This memory has nichrome or polysilicon wires arranged in a matrix. These wires can be functionally viewed as fuses.

o It is programmed by a PROM programmer which selectively burns the fuses according to the bit pattern to be stored.

o Fuses which are not blown/burned represents a logic “1" whereas fuses

ANS 6B

7-Segment Led Display

• The 7-segment LED display is an output device for displaying alpha numeric characters.

• It contains 7 LED segments arranged in a special form used for displaying alpha numeric characters and 1 LED used for representing 'decimal point' in decimal number display.

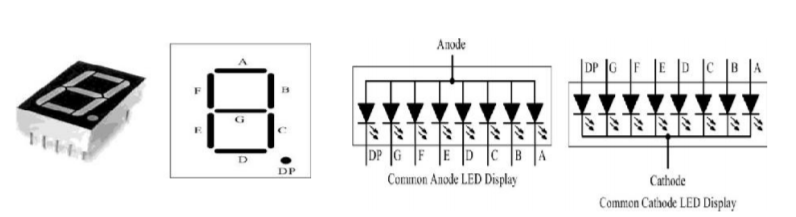
• The LED segments are named A to G and the decimal point LED segment is named as DP.

• The LED segments A to G and DP should be lit accordingly to display numbers and characters.

• The 7-segment LED displays are available in two different configurations, namely; Common Anode and Common Cathode.

• In the common anode configuration, the anodes of the 8 segments are connected commonly whereas in the common cathode configuration, the cathodes of 8 LED segments are connected commonly.

• Figure illustrates the Common Anode and Cathode configurations.



• Based on the configuration of the 7-segment LED unit, the LED segment’s anode or cathode is connected to the port of the processor/controller in the order 'A' segment to the least significant port pin and DP segment to the most significant port pin.

• The current flow through each of the LED segments should be limited to the maximum value supported by the LED display unit. o The typical value is 20mA.

o The current can be limited by connecting a current limiting resistor to the anode or cathode of each segment.

• 7-segment LED display is used in low cost embedded applications like Public telephone call monitoring devices, point of sale terminals, etc.

7 a) Which network topology is followed by USB? Explain the types of data transfer supported by USB?

b) List out some of major applications of Embedded systems.

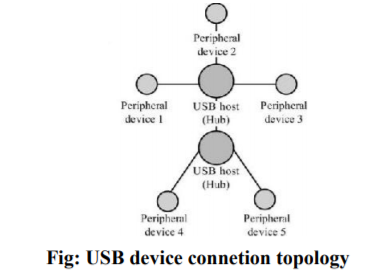
ANS 7A

Universal Serial Bus (USB):

• Universal Serial Bus (USB) is a wired high speed serial bus for data communication.

• The first version of USB (USB 1.0) was released in 1995.

• The USB communication system follows a star topology with a USB host at the centre and one or more USB peripheral devices/USB hosts connected to it.



• A USB host can support connections up to 127, including slave peripheral devices and other USB Figure illustrates the star topology for USB device connection.

• USB transmits data in packet format.

• Each data packet has a standard format.

• USB supports four different types of data transfers:

• Control transfer: Used by USB system software to query, configure and issue commands to the USB device.

• Bulk transfer: Used for sending a block of data to a device.

• Supports error checking and correction.

• Transferring data to a printer is an example for bulk transfer. • Isochronous data transfer: Used for real-time data communication. •

Data is transmitted as streams in real-time.

• Doesn't support error checking and re-transmission of data in case of any transmission loss.

• All streaming devices like audio devices and medical equipment for data collection make use of the isochronous transfer.

• Interrupt transfer: Used for transferring small amount of data

.• The USB communication is a host initiated one.

• Interrupt transfer mechanism makes use of polling technique to see whether the USB device has any data to send.

• The frequency of polling is determined by the USB device and it varies from 1 to 255 milliseconds.

• Devices like Mouse and Keyboard, which transmits fewer amounts of data, uses interrupt transfer.

ANS 7B

1. Consumer electronics: Camcorders, cameras, etc.

2. Household appliances: Television, DVD players, washing machine, refrigerators, microwave oven, etc.

3. Home automation and security systems: Air conditioners, sprinklers, intruder detection alarms, closed circuit television (CCTV) cameras, fire alarms, etc.

4. Automotive industry: Anti-lock braking systems (ABS), engine control, ignition systems, automatic navigation systems, etc.

5. Telecom: Cellular telephones, telephone switches, handset multimedia applications, etc.

6. Computer peripherals: Printers, scanners, fax machines, etc.

7. Computer networking systems: Network routers, switches, hubs, firewalls, etc.

8. Healthcare: Different kinds of scanners, EEG, ECG machines, etc.

9. Measurements & Instrumentation: Digital multimeters, digital CROs, logic analyzers, PLC systems, etc.

10. Banking & Retail: Automated teller machines (ATM) and currency counters, point of sales (POS), etc.

11. Card readers: Barcode, smart card readers, hand held devices, etc.

8 a) Explain the working mechanism of Push Button Switch.

b) Explain the SPI Bus interfacing with a neat diagram.

ANS 8A

• It is an input device.

• Push button switch comes in two configurations, namely 'Push to Make' and 'Push to Break’.

• In the 'Push to Make' configuration, the switch is normally in the open state and it makes a circuit contact when it is pushed or pressed.

• In the 'Push to Break' configuration, the switch is normally in the closed state and it breaks the circuit contact when it is pushed or pressed.

• The push buttons stays in the ‘closed’ (For Push to Make type) or ‘open’ (For Push to Break type) state as long as it is kept in the pushed state and it breaks/makes the circuit connection when it is released.

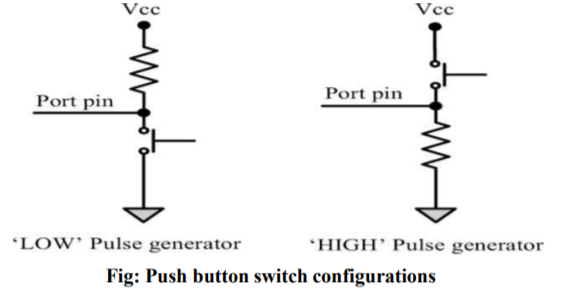
• Push button is used for generating a momentary pulse.

• In embedded applications, push button is generally used as reset and start switch and pulse generator.

• The Push button is normally connected to the port pin of the host processor/controller.

• Depending on the way in which the push button interfaced to the controller, it can generate either a ‘HIGH’ pulse or a 'LOW' pulse.

• Figure illustrates how the push button can be used for generating 'LOW' and 'HIGH' pulses



ANS 8B

• The Serial Peripheral Interface Bus (SPI) is a synchronous bi-directional full duplex four-wire serial interface bus.

• The concept of SPI was introduced by Motorola.

• SPI is a single master multi-slave system.

o There can be more than one master, but only one master device can be active at any given point of time.

• SPI requires four signal lines for communication. They are:

o Master Out Slave In (MOSI) – Signal line carrying the data from master to slave device. It is also known as Slave Input/SIave Data In (SI/SDI)

o Master In Slave Out (MISO) – Signal line carrying the data from slave to master device. It is also known as Slave Output (SO/SDO)

o Serial Clock (SCLK) – Signal line carrying the clock signals

o Slave Select (SS) – Signal line for slave device select. It is an active low signal

• The bus interface diagram shown in the figure illustrates the connection of master and slave devices on the SPI

• The master device is responsible for generating the clock signal.

• It selects the required slave device by asserting the corresponding slave device's slave select signal 'LOW’.

• The data out line (MISO) of all the slave devices when not selected floats at high impedance state.

• The serial data transmission through SPI bus is fully configurable.

o SPI devices contain a certain set of registers for holding these configurations.

o The control register holds the various configuration parameters like master/slave selection for the device, baud rate selection for communication, clock signal control, etc.

o The status register holds the status of various conditions for transmission and reception.

• SPI works on the principle of 'Shift Register’.

o The master and slave devices contain a special shift register for the data to transmit or receive.

o The size of the shift register is device dependent. Normally it is a multiple of 8.

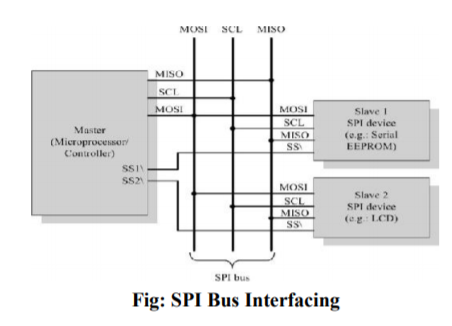
• During transmission from the master to slave, the data in the master's shift register is shifted out to the MOSI pin and it enters the shift register of the slave device through the MOSI pin of the slave device.

• At the same time the shifted out data bit from the slave device's shift register enters the shift register of the master device through MISO pin.

• In summary, the shift registers of 'master' and 'slave' devices form a circular buffer.

• When compared to I2C, SPI bus is most suitable for applications requiring transfer of data in ' streams '.

• The only limitation is SPI doesn't support an acknowledgement mechanism.



9 a) Explain the mechanism of Unipolar and Bipolar phases in Stepper Motor.

b) Define Sensor and Actuator with example

ANS 9A

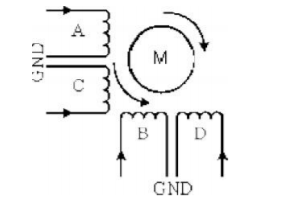
Unipolar:

• A unipolar stepper motor contains two windings per phase. The direction of rotation (clockwise or anticlockwise) of a stepper motor is controlled by changing the direction of current flow.

• Current in one direction flows through one coil and in the opposite direction flows through the other coil. It is easy to shift the direction of rotation by just switching the terminals to which the coils are connected.

• The coils are represented as A, B, C, D. Coil A and C carry current in opposite directions for phase 1. Similarly, coil B and D carry current in opposite direction for phase 2. (only 1 of them will be carrying current at a time.

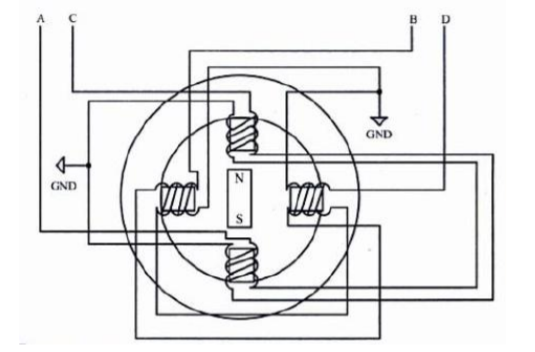
• The figure illustrates the working of a two- phase unipolar stepper motor.



Bipolar:

• A bipolar stepper motor contains single winding per phase. For reversing the motor rotation, the current flow through the windings is reversed dynamically. It requires complex circuitry for current flow reversal.

• The stator winding details for a two phase unipolar stepper motor is shown in the below figure



• The stepper motor can be implemented in different ways by changing the sequence of activation of stator windings.

ANS 9B

SENSORS AND ACTUATORS

• An embedded system is in constant interaction with the real world and the controlling/monitoring functions executed by the embedded system are achieved in accordance with the changes happening to the real world.

• The changes in system environment or variables are detected by the sensors connected to the input port of the embedded system.

• If the embedded system is designed for any controlling purpose, the system will produce some changes in the controlling variable to bring the controlled variable to the desired value.

• It is achieved through an actuator connected to the output port of the embedded system. Sensor:

• A sensor is a transducer device that converts energy from one form to another for any measurement or control purpose.

o E.g.: Temperature sensor, magnetic Hall Effect sensor, humidity sensor, etc.

Actuator:

• An actuator is a form of transducer device (mechanical or electrical) which converts signals to corresponding physical action (motion).

• Actuator acts as an output device.

o E.g.: Stepper motor

10 a) Write a short note on Reset Circuit.

b) What is Embedded Firmware? Explain the various methods for developing Embedded Firmware?

ANS 10A

Reset Circuit:

• The reset circuit is essential to ensure that the device is not operating at a voltage

level where the device is not guaranteed to operate, during system power ON.

• The reset signal brings the internal registers and the different hardware systems of the processor/controller to a known state and starts the firmware execution from the reset vector

o Normally from vector address 0x0000 for conventional processors/controllers.

• The reset signal can be either active high or active low.

• Since the processor operation is synchronized to a clock signal, the reset pulse should be wide enough to give time for the clock oscillator to stabilize before the internal reset state starts.

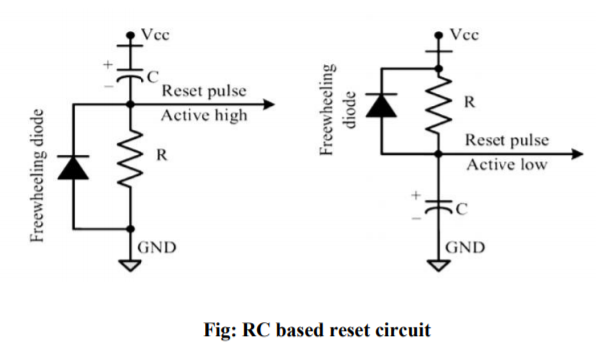
• The reset signal to the processor can be applied at power ON through an external passive reset circuit comprising a Capacitor and Resistor or through a standard Reset IC like MAX810 from Maxim Dallas.

• Select the reset IC based on the type of reset signal and logic level (CMOS/TTL) supported by the processor/controller in use.

• Some microprocessors/controllers contain built-in internal reset circuitry and they don't require external reset circuitry.

• Figure illustrates a resistor capacitor based passive reset circuit for active high and low configurations.

o The reset pulse width can be adjusted by changing the resistance value R and capacitance value C.



ANS 10B

• Embedded firmware refers to the control algorithm (Program instructions) and or the configuration settings that an embedded system developer dumps into the code (Program) memory of the embedded system.

• It is an un-avoidable part of an embedded system. There are various methods available for developing the embedded firmware:

o Write the program in high level languages like Embedded C/C++ using an Integrated Development Environment (IDE).

• The IDE will contain an editor, compiler, linker, debugger, simulator, etc. IDES are different for different family of processors/controllers. For example, Keil µVision 4 IDE is used for all family members of 8051 microcontroller, since it contains the generic 8051 compiler C51.

o Write the program in Assembly language using the instructions supported by your application’s target processor/controller.

• The program written in high level language or assembly code should be converted into a processor understandable machine code before loading it into the program memory.

• The process of converting the program written in either a high level language or processor/controller specific Assembly code to machine readable binary code is called 'HEX File Creation’.

• The methods used for 'HEX File Creation' is different depending on the programming techniques used. o If the program is written in Embedded C/C++ using an IDE, the cross compiler included in the IDE converts it into corresponding processor/controller understandable 'HEX File’.

o If Assembly language based programming technique is used, the utilities supplied by the Processor/controller vendors can be used to convert the source code into 'HEX File’.

o Also third party tools are available, which may be of free of cost, for this conversion.

• For a beginner in the embedded software field, it is strongly recommended to use the high level language based development technique.

o Writing codes in a high level language is easy

o The code written in high level language is highly portable

• The same code can be used to run on different processor/controller with little or less modification.

• The only thing you need to do is re-compile the program with the required processor's IDE, after replacing the include files for that particular processor.

o The programs written in high level languages are not developer dependent.

• Any skilled programmer can trace out the functionalities of the program by just having a look at the program

• It will be much easier if the source code contains necessary comments and documentation lines.

•It is very easy to debug and the overall system development time will be reduced to a greater extent.