Sixth Semester B.E. Degree Examination, Dec.2019/Jan.2020
ARM Microcontroller & Embedded Systems

Time: 3 hrs. Max. Marks: 80

Note: ***Answer FIVE full questions, choosing one full question from each module.***

Module-1

1 a. Explain the architecture of ARM cortex – M3 processor with the help of neat block diagram. **(10 Marks)**

b. List and explain the features of ARM cortex M3 processor. **(06 Marks)**

OR

2 a. Explain the operation modes and privilege levels in cortex M3 processor. **(08 Marks)**

b. Explain two stack model and reset sequence in ARM cortex M3. **(08 Marks)**

Module-2

3 a. Explain the following instruction with examples:
(i) ASR (ii) LSL (iii) ROR (iv) REV **(08 Marks)**

b. Briefly explain bit band operations and memory map of cortex M3. **(08 Marks)**

OR

4 a. Write a note on barrier instruction in cortex M3. **(06 Marks)**

b. With a diagram, explain the organization of CMSiS and its benefits. **(10 Marks)**

Module-3

5 a. Define embedded systems. Explain the 6 purpose of embedded systems with an example for each. **(08 Marks)**

b. Explain the classification of embedded systems based on generation. **(04 Marks)**

c. Mention the application of embedded system with an example for each. **(04 Marks)**

OR

6 a. Explain the different 'on board' communication interfaces in brief. **(08 Marks)**

b. Write a note on: (i) Reset circuit (ii) Watch dog timer. **(08 Marks)**

Module-4

7 a. Explain the different characteristics of embedded system in detail. **(08 Marks)**

b. With a block diagram, mention the components and in the design of a washing machine and also explain its working. **(08 Marks)**

OR

8 a. What is hardware and software co-design? Explain the fundamental design approaches in detail. **(10 Marks)**

b. With FSM model, explain the design and operation of automatic tea/coffee lending machine. **(06 Marks)**

Head of the Department
Dept. of Electronic & Communication Engg.
K.L.S.V.D.I.T. HALIYAL (U.K.)

Scheme & Solution Prepared by
Prof. Rohini Kallur

Module-5

9 a. Define process. Explain in detail the structure, memory organization and state transitions of the process. **(08 Marks)**

b. Explain multi processing, multi tasking and multi programming. **(08 Marks)**

OR

10 a. Explain the simulator and emulator. **(08 Marks)**

b. Write a note on message passing. **(08 Marks)**

Name of the Faculty : Prof. Rohini Kallur

Name of the Institution : KLS VDIT Haliyal

Department : ECE

Subject : Embedded Systems [18EC62]

Semester : VI

I have written the solution for 15EC62 i.e ARM Microcontroller & Embedded Systems as 18EC62 and 17EC62 VTU Examination is not yet conducted and also Model Question paper is not available. 15EC62 matches 100% Syllabus of 18EC62 and 17EC62.

P Kallur 09/07/2021

Subject Teacher :

Prof. Rohini Kallur

Solution of Subject: ARM Microcontroller and Embedded Systems

Subject code: 15EC62 Dec 2019 / Jan 2020

Module 1

1)a) Explain the architecture of ARM Cortex M_3 Processor with the help of neat block diagram. [10 Marks]

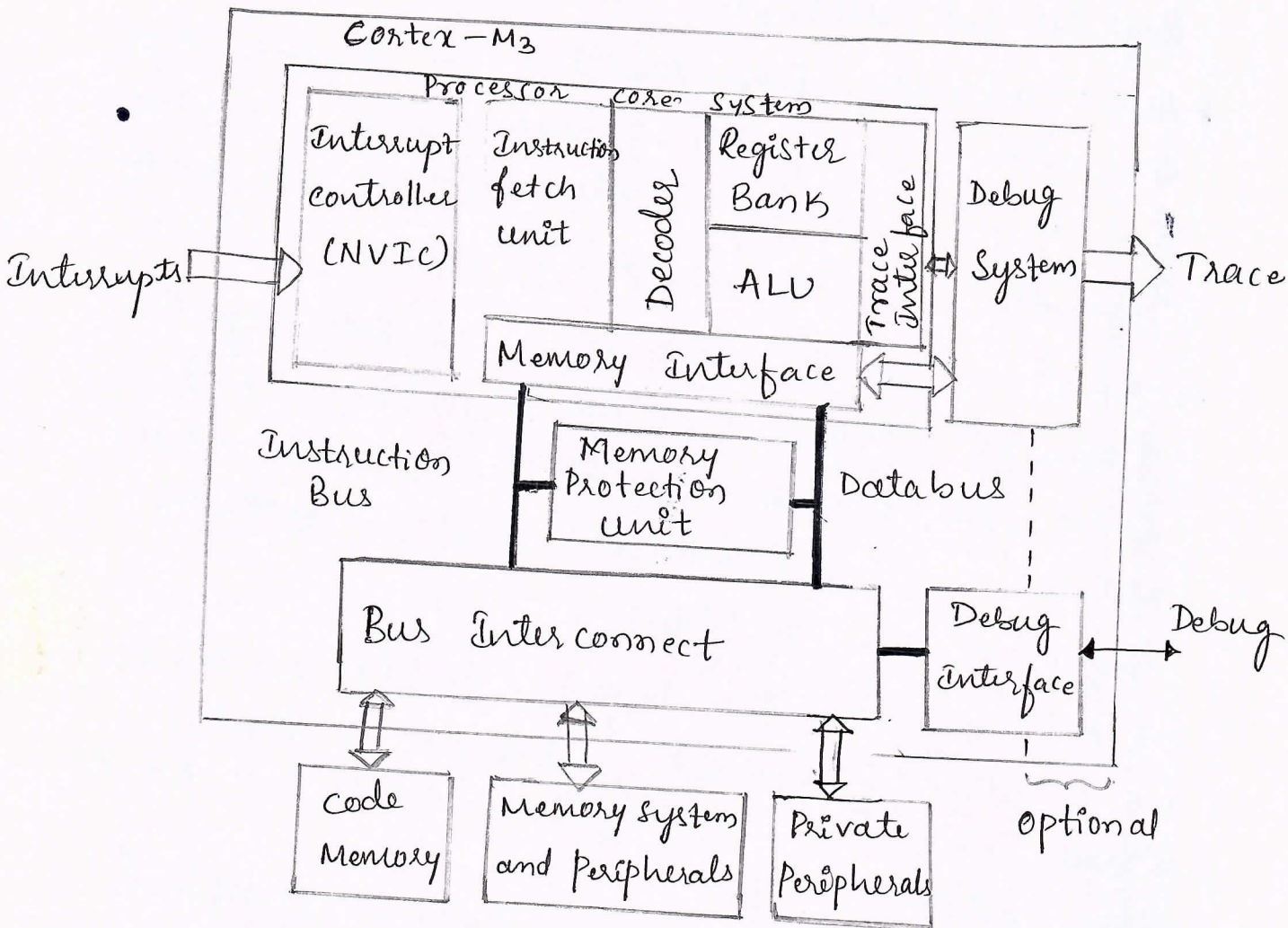


Fig: Architecture of ARM Cortex M_3

→ Cortex M_3 is a 32-bit Micro processor. It has 32-bit data path, a 32-bit register bank and 32-bit memory interfaces.

→ The processor has a Harvard Architecture, which means that it has a seperate instruction bus and data bus.

This allows instructions and accesses to take place at the same time and as a result of this, the performance of the processor increases because data accesses do not affect the instrn Pipeline.

ARM Cortex M_3 has many special features: Optional Memory Protection unit, Internal debugging components, breakpoints and watch points Register set, Interface Trace unit and so on.

→ ARM Cortex M_3 has registers R_0 through R_{15}. in which $R_0 - R_{12}$ are General purpose registers. R_{13} is the stack pointer, R_{14} is the Link Register and R_{15} is the program counter.

→ ARM cortex M_3 also has special registers Such as Program status Registers (PSRs), Interrupt Mask Registers (PRIMASK, FAULT MASK and BASEPRI) and a CONTROL Register of 2 bits.

→ ARM Cortex M_3 has a built in NVIC i.e Nested Vectored Interrupt controller which provides multiple features to handle the interrupts.

→ ARM Cortex M_3 also has a predefined memory map. The total memory of 4 GB is divided into multiple sections and allocated as Code Region, RAM Region, Peripheral and System level region.

→ The processor also has Bus interface, in which we have code memory bus, System bus and Private Peripheral Bus.

→ ARM Cortex M_3 has an Optional MPU. This unit allows access rules to be set up for privileged access and user program access.

→ ARM Cortex M_3 supports Thumb-2 instructio Set. It allows 255 exceptions which includes external interrupts, exceptions and NMI, Reset

→ The Cortex M_3 processor includes a number of debugging features such as program execution controls including halting and stepping, instructio breakpoints, data watchpoints, registers and memor access, profiling and traces.

1) b) List and explain the features of ARM Cortex M_3 Processor. [06 marks]

Ans: The features of ARM Cortex M_3 are

a) High performance

→ Cortex M_3 processor delivers high perfor-mance in microcontroller products:

In this processor many instructions are single cycle. So the performance of the processor will be better. It has seperate data and instruction bus which allows simultaneous data and instruction accesses to be performed.

b) Advanced Interrupt handling features

→ The interrupt features on the Cortex M_3 Processor are easy to use, very flexible, and provide high interrupt processing throughput.

→ It supports up to 240 external interrupt i/ps

c) Low-power Consumption

→ The Cortex M_3 Processor is suitable for low

Power designs because of the low gate count.
→ It has power saving mode support such as Sleeping and Sleep deep mode.

d) System features

The cortex M_3 processor various System features making it suitable for a large no of applications, such as → bit band operation, byte - invariant big endian mode and unalligned data access support.

e) Debug support

ARM Cortex M_3 supports JTAG or Serial wire debug interface. and the coresight debugging solution, processor status or memory contents can be accessed even when the core is running. It has optional ETM for instruction trace and data trace using DWT.

2) a) Explain the Operation modes and privilege levels in Cortex M_3 processor. [08 marks]

Ans:- The ARM Cortex M_3 has two operation modes and two privilege levels. The Operation modes i.e thread mode and handler mode determine whether the processor is running a normal program or running an exception handler like an interrupt handler or System exception handler.

The privilege levels i.e Privileged level and user level provide a mechanism for safegua-rding memory accesses to critical regions as well as providing a basic security model.

	Privileged	User
when running an exception Handler	Handler Mode	
when not running an exception handler	Thread mode	Thread mode

Fig 1: Operation modes & Privilege levels

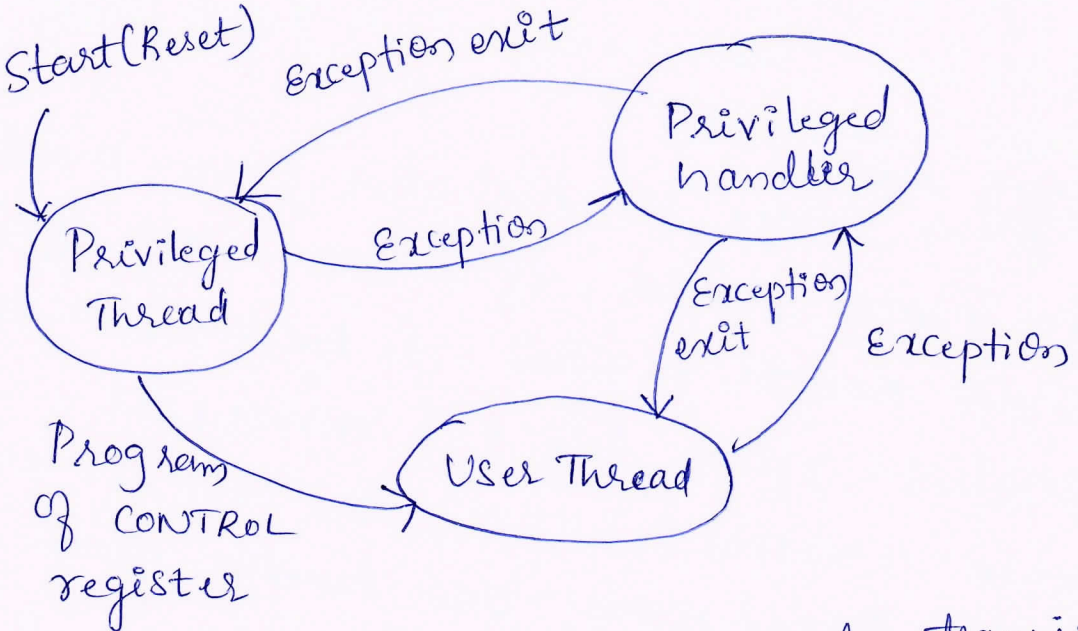


Fig 2: Allowed operation mode transitions.

→ When the processor is running a main program (Thread mode), it can be either in a Privileged State or a user State, but exception handlers can only be in a privileged State.

→ When the processor exits reset, it is in thread mode, with privileged access rights.

→ In the privileged State, a program has access to all memory ranges and can use all supported instructions.

As shown in the figure 2 above, when an exception takes place, the processor will always switch back to the privileged State and return to the previous State when exiting the exception handler. A user program cannot change back

to the privileged state by writing to the control register. It has to go through an exception handler that programs the control register to switch the processor back into the privileged access level when returning to thread mode.

2) b) Explain two stack model and Reset sequence in ARM Cortex M_3. [08 Marks]

Ans:- Two Stack model →

The ARM Cortex M_3 has two stack pointers ① Main stack pointer ② Process Stack pointer

The Stack pointer register to be used is controlled by control register bit 1 [CONTROL[1]].

When CONTROL[1] is 0, the MSP is used for both thread mode and handler mode as shown in figure 1 below.

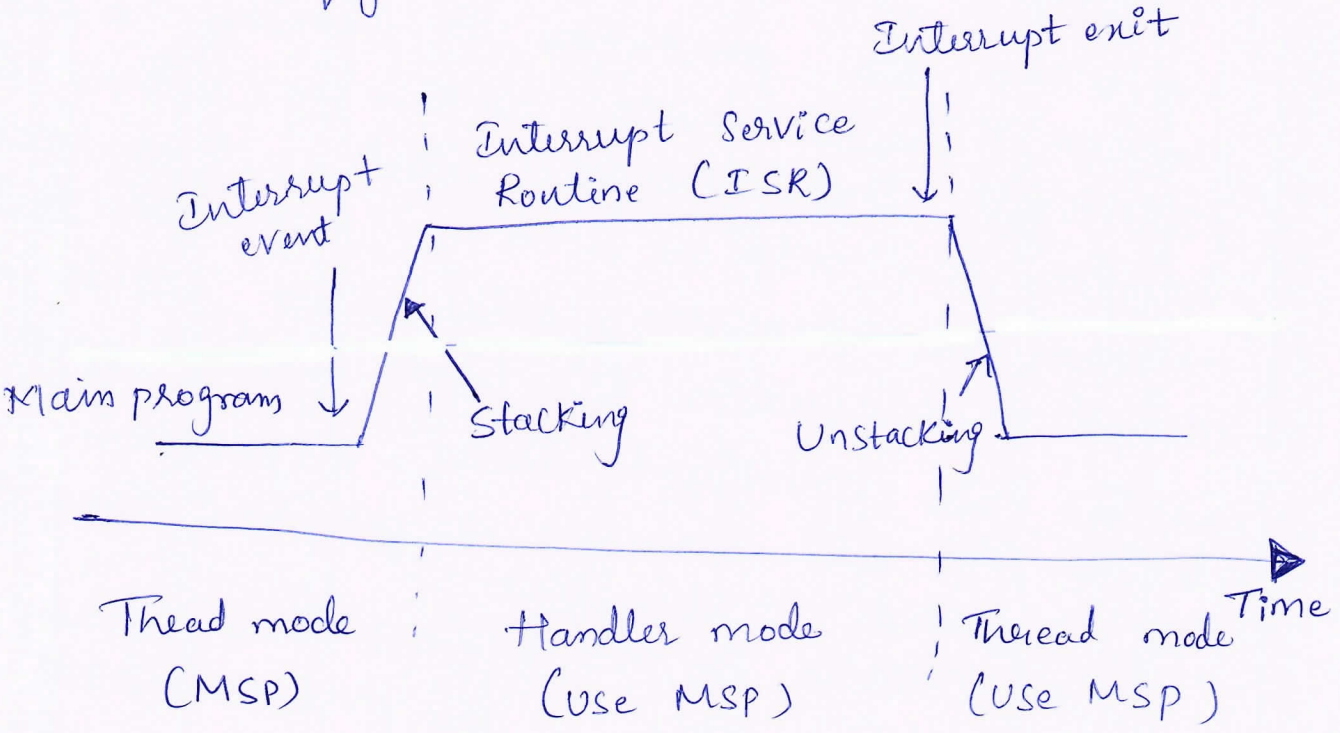


Figure 1: CONTROL[1]=0, Both Thread level and Handler use Main stack

In this arrangement, the main program and the exception handlers share the same stack memory region. This is the default setting after power up.

When the CONTROL[1] is 1, the PSP is used in thread mode as shown in the figure 2 below. In this arrangement, the main program and the exception handler can have seperate stack memory regions.

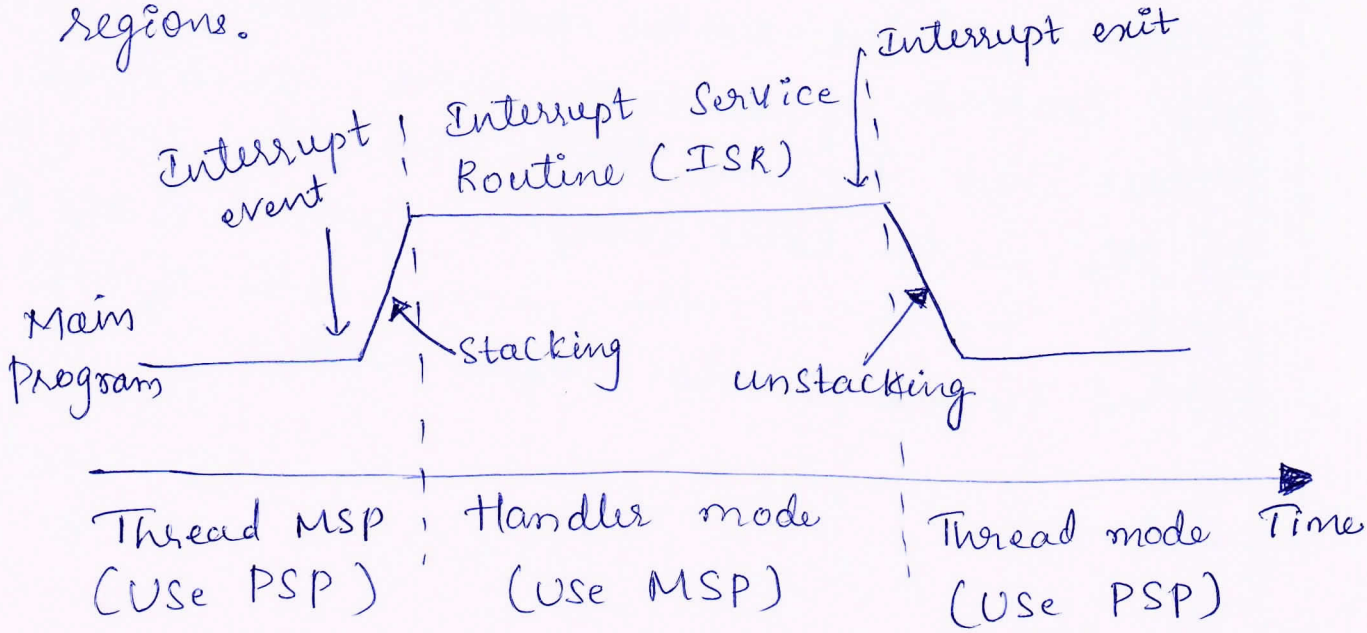


Figure 2: CONTROL[1] = 1; Thread level uses Process Stacks and Handler uses Main Stacks.

Reset Sequence →

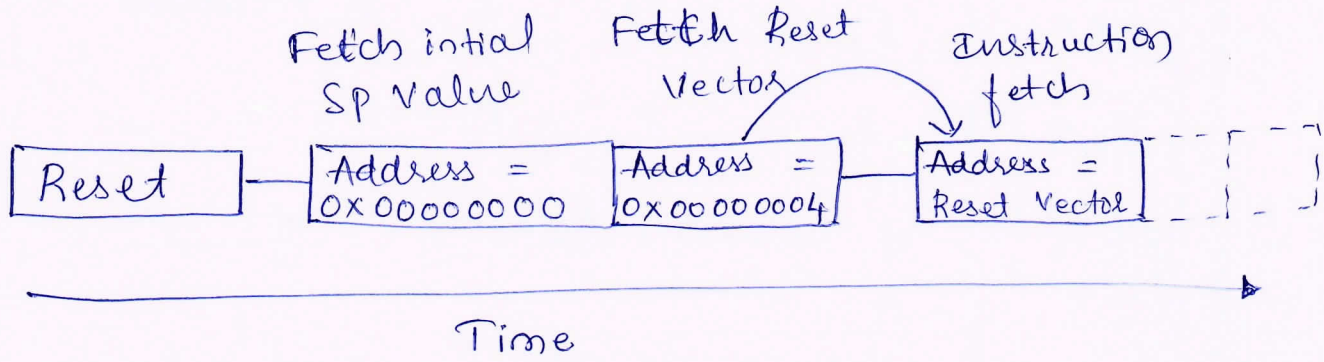


Fig: 3 Reset Sequence

After the processor exits reset, it will read two words from memory as shown in figure 3.

1) Address 0x00000000 : Starting value of R_{13}(SP)
2) Address 0x00000004 : Reset vector, which is the starting address of program execution, LSB should be set to 1 to indicate Thumb State.

In ARM Cortex M_3, the initial value for the MSP is put at the beginning of the memory map, followed by the vector table, which contains vector address values. In addition the contents of the vector table are address values not branch instructions. The first vector in the vector table (exception type 1) is the Reset Vector, which is the second piece of data fetched by the processor after reset.

3) a) Explain the following instruction with examples.
i) ASR ii) LSL iii) ROR iv) REV
[08 marks]

Ans:- i) ASR
It is an Arithmetic Shift Right instruction.
The Syntax of this instruction is

ASR $R_d, R_n, \#immed$; $R_d = R_n >> immed$

ASR R_d, R_n ; $R_d = R_d >> R_n$

ASR.W R_d, R_n, R_m ; $R_d = R_n >> R_m$

From the blocks above we can see that, the contents of the register will be right shifted as many times as defined in the instruction. The LSB of the register will move to the carry flag and MSB will be retained as it is.

Eg: LDR R_0, #0x04

ASR R_1, R_0, #01

Before Execn

R_0 = 0000 0000 0000 0100

After execn

R_1 = 0000 0000 0000 0010 cy = 0

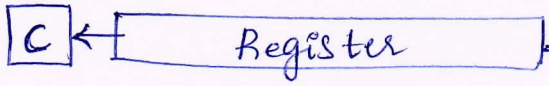
ii) LSL : Logical shift Left

The Syntax of this instruction is

LSL R_d, R_n, #immed ; $R_d = R_n$ << immed

LSL R_d, R_n ; $R_d = R_d$ << R_n

LSL.W R_d, R_n, R_m ; $R_d = R_n$ << R_m



In this instruction, the contents of the register are left shifted by sending the MSB to the carry flag and LSB will get appended with 0's.

Eg: LDR R_0, #0x08

LSR R_1, R_0, #01

Before execution:

R_0 = 0000 0000 0000 1000

After execution:

R_1 = 0000 0000 0001 0000 cy = 0

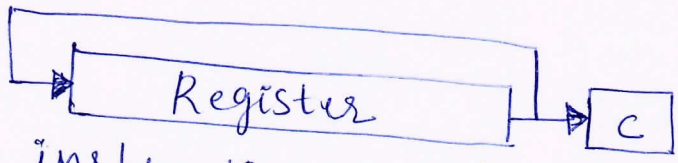
iii) ROR : Rotate Right

The syntax of this instruction is,

ROR R_d, R_n;

ROR.W R_d, R_n, #immed;

ROR.W R_d, R_n, R_m;



This instruction is Rotate Right instruction. In this instruction the source register will get rotated no of times defined in the instruction. The LSB of the content of the register will move to carry flag as well as to the MSB of the register.

Eg: LDR R_1, #0x0A

ROR R_2, R_1, #02

Before execution:

R_1 = 0000 0000 0000 1010

After execution

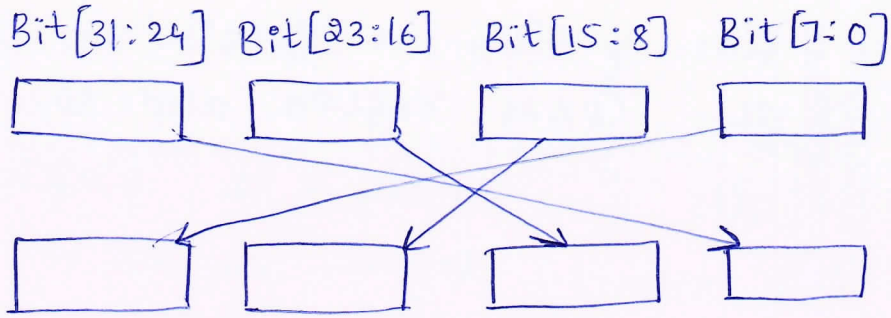
R_2 = 1000 0000 0000 0010 Cy = 1

iv) REV : Reverse bytes in word

Syntax of this instruction is

REV R_d, R_n; R_d = Rev(R_n)

REV.W



In this instruction, the bytes in the word are interchanged and the data will be reversed. Here the MSB will become LSB.

i.e Bit [31:24] will move to Bit [7:0]
Bit [23:16] will move to Bit [15:8]
Bit [15:8] will move to Bit [23:16]
Bit [7:0] will move to Bit [31:24]

Eg: LDR R_0, #12345678

~~R0R~~ REV R_2, R_0

Before execution

R_0 = ~~0001~~ 12 34 56 78

After execution

R_0 12 34 56 78

R_2 78 56 34 12

3)b) Briefly explain bit band operations and memory map of cortex M_3. [08 Marks]

Ans: Bit band operations

Bit band operation support allows a single load/store operation to access to a single data bit. In the ARM cortex M_3, this is supported in two predefined memory regions called bit-band regions.

One of them is located in the first 1 MB of the SRAM region and the other is located in the first 1 MB of the peripheral region. These two memory regions can be accessed like normal memory, but they can also be accessed via a seperate memory region called the bit-band alias. When the bit band alias address is used, each individual bit can be accessed seperately in the LSB of each word-alligned address.

The ARM cortex M_3 uses the following terms for the bit-band memory addresses:

* Bit band region: This is a memory address region that supports bit band operation.
* Bit band alias: Access to the bit-band alias will cause an access to the bit band region.

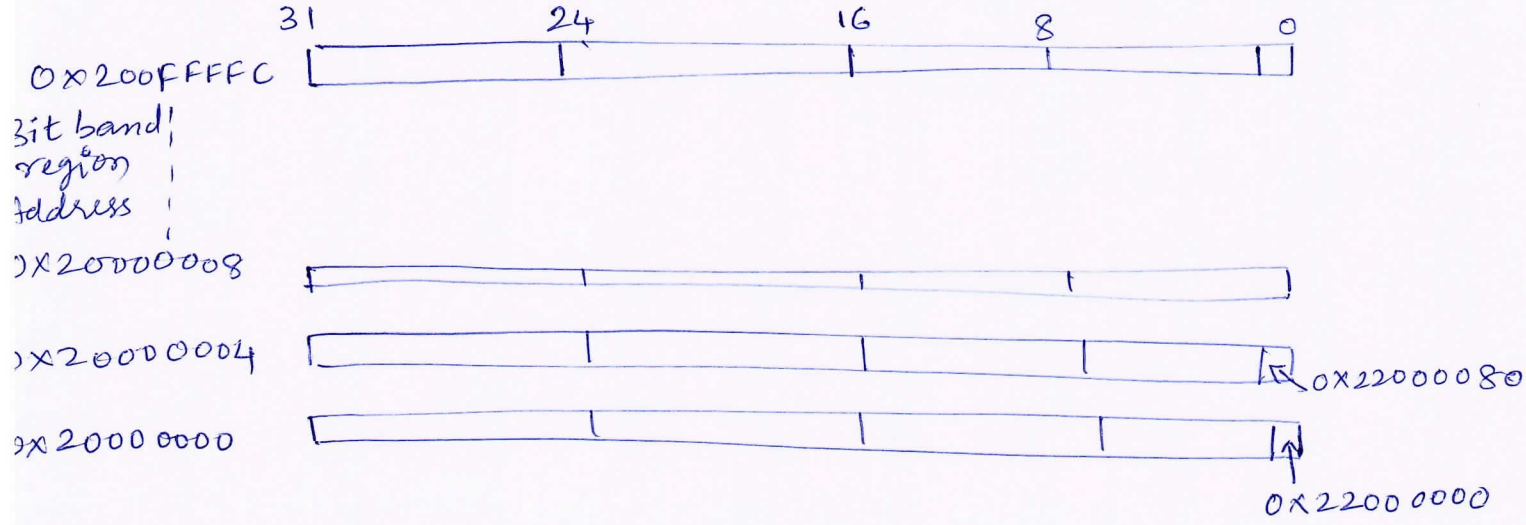


Fig 1: Bit accesses to Bit Band Region via Bit band alias.

Memory Map →

The Cortex - M_3 processor has a fixed memory map as shown in figure 2 below. This makes it easier to port software from one Cortex M_3 product to another.

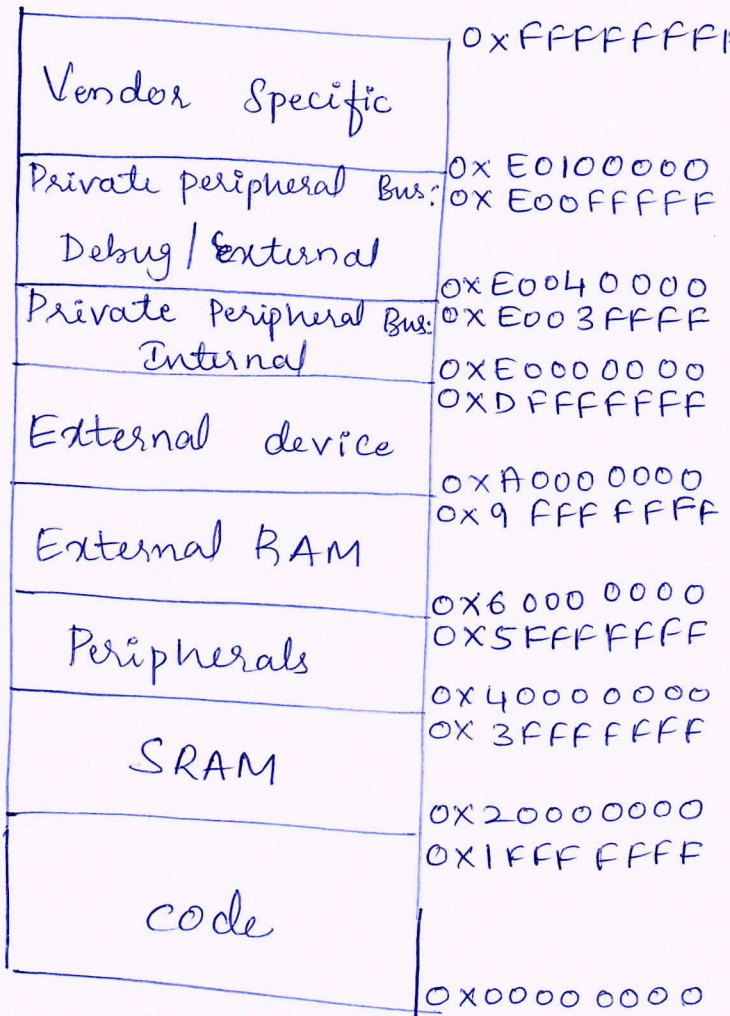


Fig: 1 Cortex M_3 Memory map

The cortex M_3 Processor has a total of 4GB of address space. Program code can be located in the code region, the SRAM region or the external RAM region.

The SRAM memory range is for connecting internal SRAM. Access to this region is carried out via the System interface bus. In this region, a 32-MB range is defined as a bit-band alias. Within the 32-bit band alias memory range each

word address represents a single bit in the 1 MB bit-band region.

Another 0.5GB block of address range is alloca to on-chip peripherals. Similar to the SRAM regi this region supports bit-band alias and is acce via the System bus interface.

Two Slots of 1-GB memory space are allocated for external RAM and external devices. The diffe between the two is that program execution in the external device region is not allowed, and there are some differences with the caching behaviors.

The last 0.5 GB memory is for the System level components, internal peripheral buses, external peripheral bus and Vendor-Specific System peripherals.

4) a) Write a note on barrier instruction in cortex M_3. [06 MARKS]

Ans: The cortex M_3 supports a number of barrier instructions. These instructions are needed as memory systems get more and more complex. The three barrier instructions in the Cortex M_3 are a) DMB b) DSB c) ISB

a) DMB — Data memory barrier; ensures that all memory accesses are completed before new memory access is Committed.

b) DSB — Data Synchronization barrier; ensures that all memory accesses are completed before next instruction is executed.

c) ISB – Instruction Synchronization barrier; flushes the pipeline and ensures that all previc instructions are completed before executing new instructions.

The DSB and ISB instructions can be imp - ortant for self modifying code. The ISB instruct should be used after updating the value of th CONTROL register. DMB is very useful for mult Processor Systems. DMB instructions can be inserted between accesses to the shared memory to ensure that the memory access sequence is exactly the same as expected.

4) b) With a diagram, explain the organization of CMSIS and its benefits. [10 marks]

Ans:- CMSIS Stands for Cortex Microcontroller Software Interface Standard.

→ CMSIS is divided into multiple layers

a) Core peripherals Access layer

b) Middleware access layer

c) Device peripheral access layer

d) Access functions for peripherals

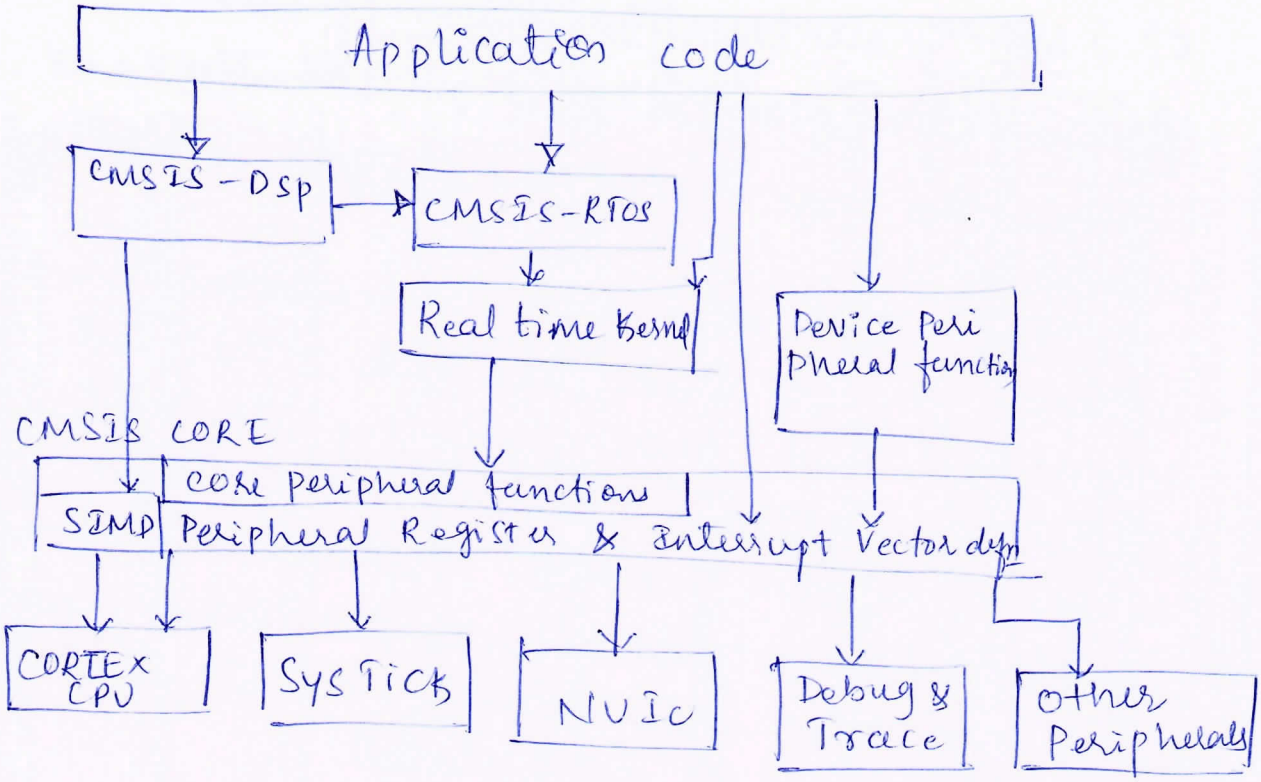


fig1: Structure of CMSIS

Since the CMSIS is incorporated inside the device driver library, there is no special setup requirement for using CMSIS in projects. For each MCU device, the MCU vendor provides a header file, which pulls in additional header files required by the device driver library, including the core peripheral Access layer defined by ARM.

Benefits of CMSIS →

The main advantage is much better software Portability and reusability. It allows software to be quickly ported between Cortex-M3 and other cortex-M processors, reducing time to market.

→ By using the CMSIS, their software products can become compatible with device drivers from multiple micro controllers vendors, including future microcontroller products that are yet to be released.

→ The CMSIS has a small Memory foot print i.e less than 1KB for all core access functions and a few bytes of RAM. It also avoids overlapping of core peripheral driver code when reusing software code from other projects

→ By use of CMSIS, the intrinsic functions are not compiler specific.

5)a) Define embedded Systems. Explain the 6 purpose of embedded Systems with an example for each. [08 marks]

Ans:- Embedded system.

Embedded System is an Electronic / electro mechanical System designed to perform a specific function.

Purpose of Embedded Systems →

Each Embedded System is designed to serve the purpose of any one or a combination of the following.

a) Data collection / Storage / Representation
b) Data Communication
c) Data processing
d) Monitoring
e) Control
f) Application specific user interface

a) Data collection/ Storage / Representation

Embedded System designed for the purpose of data collection performs acquisition of data from the external world. Data collection is usually done for Storage, analysis, manipulation and transmission. The data collected can be either analog or digital.
Eg: Digital camera, which collects data or image, Stores it and represents it on the screen

b) Data communication

Embedded data communication systems are deployed in applications from complex satellite communication to simple home network systems. The transmission of data is achieved either by a wire-in medium or by a wireless medium.
* Wireless modules are — Bluetooth and wi-fi
* wire line modules are - USB, TCP/IP
Eg— Network hubs, routers, switches

c) Data signal processing

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

Eg: A digital hearing aid

d) Monitoring

All Embedded products coming under the medical domain are with monitoring functions

→ ECG is an example of monitoring device

Other examples with monitoring function are digital CRO, digital multimeters and logic analyzers.

e) control

A System with control functionality contains both sensors and actuators. Sensors are connected to the i/p port for capturing the ~~things~~ changes in environmental variable and the actuators connected to the o/p port are controlled according to the changes in the i/p variable.

Eg: Air conditioner System used to control the room temperature to a specified limit is a typical example for control purpose.

f) Application Specific User Interface

Buttons, Switches, Keypad, lights, bells and display unit are Application Specific User interfaces. Mobile Phone is an example of Application Specific User interface.

→ In mobile phone, the user interface is provided through the Keypad, System Speaker, Vibration alert etc.

5.b) Explain the classification of embedded system based on generation. [04 marks]

Ans:- Embedded Systems can be classified based on Generation as given below.

a) First generation Embedded System
b) Second generation Embedded System
c) Third generation Embedded System
d) Fourth generation Embedded System

a) First generation Embedded System

→ These Embedded Systems are built around 8-bit processors such as 8085, Z80. It uses simple circuits with the use of assembly language code, Telephone keypads, Stepper motor control units etc..

b) Second generation Embedded System

These embedded Systems are built around 8/16 bit microprocessors or microcontrollers. These systems are much complex and have powerful instruction set.

Ex- System developed with Data Acquisition Systems, SCADA Systems.

c) Third generation Embedded System

These embedded Systems use 32-bit processors, 16-bit Microcontrollers, DSP Processors evolution, ASIC designs. In this more powerful and complex instruction set is present.

→ Different Vendors came into existence such as Intel, Pentium, Motorola, Zylog. High performance requirements were need of the day.

→ Wide range of applications - Robotics, Media, Industrial Process control, and Networking

d) Fourth Generation Embedded Systems.

These Embedded Systems are built around 64-bit Microprocessors and 32-bit microcontrollers. The concept of System on chip (SOC), Multi core Processors evolved during this generation. These embedded Systems have highly complex and Powerful instruction set. Smart phones and devices, mobile internet devices were evolved.

5) c) Mention the application of embedded system with an example for each. [04 Marks]

Ans:- The application area and the products in the embedded domain are countless. Some of them are listed below.

a) consumer electronics : Washing machine, Cameras
b) Household appliances : Refrigerator
c) Automotive industry : Antilock Braking System, Engine control
d) Home automation & Security Systems : Air conditioner, Sprinkler, fire alarm
e) Telecom : cellular phones, telephone Switches
f) Computer peripherals : Printers, Scanners
g) Computer network System : Network routers, Switches
h) Health care : EEG, ECG machines
i) Banking and retail : ATM, Point of Sales
j) card readers : Bar code, Smart card readers.

6)a) Explain the different 'ON Board' communication Interfaces in brief. [08 marks]

Ans:- ON Board communication interface is also called as Device / Board level communication interface.

The different ON-Board communication Interfaces are

a) Inter Integrated Circuit Bus
b) Serial Peripheral Interface Bus
c) Universal Asynchronous Receiver Transmitter
d) 1-wire Interface
e) Parallel Interface

a) Inter Integrated Circuit bus

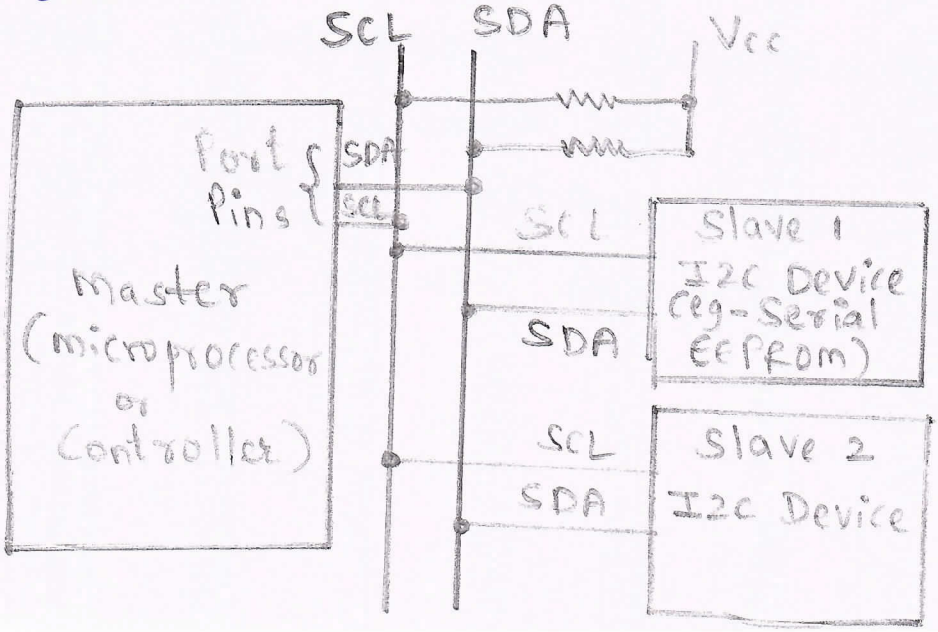
The I2C bus is a synchronous bidirectional half duplex two wire serial interface bus.

The I2C bus comprise of two bus lines

i) Serial clock (SCL line) → which is responsible for generating synchronization clock pulse

ii) Serial data line (SDA line) → which is responsible for transmitting the serial data across devices

I2C bus is a shared bus system to which number of I2C devices can be connected. Devices connected to the I2C bus can act as either master or slave.



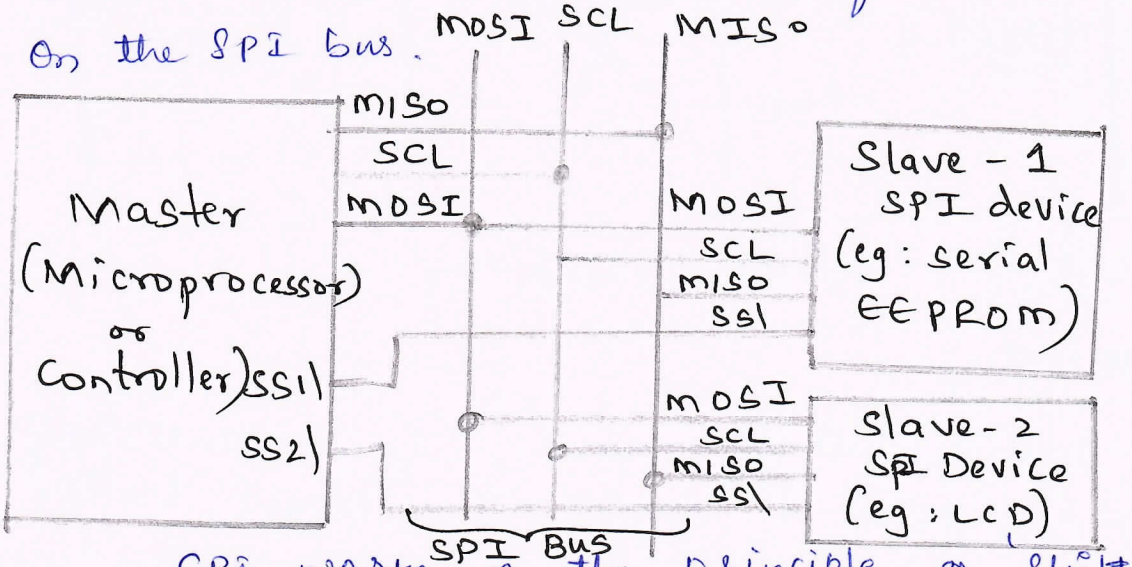
b) Serial peripheral Interface Bus

This is a Synchronous Bi-directional full-duplex four-wire serial interface bus.

SPI requires 4 signals for communication, Namely

a) Master out Slave in
b) Master in Slave out
c) Serial clock
d) Slave Select

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



SPI works on the principle of Shift Register. The master and slave devices contain a special shift register for the data to transmit or receive. 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.

c) UART :-

UART based data transmission is an asynchronous form of serial data transmission. UART based serial data transmission does not require a clock signal to synchronise the transmitting end and receiving end

for transmission. The serial communication settings for both transmitter and receiver should data stream. While sending a byte of data, a start bit is added first and a stop bit is added at the end of the bit stream.

For proper communication, the Transmit line of the sending device should be connected to the Receive line of the receiving device. Figure below illustrates the same

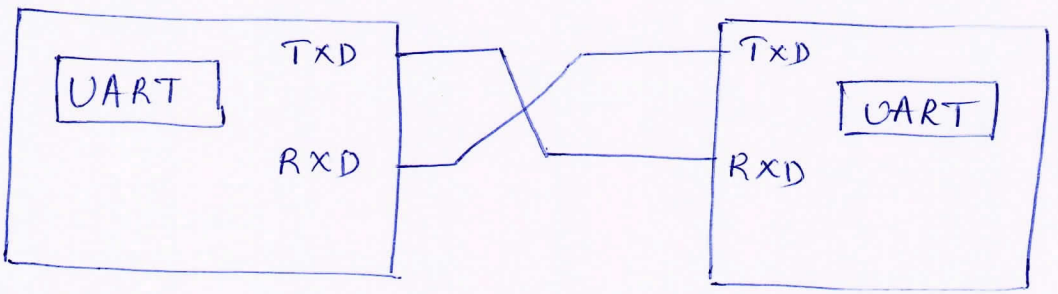


Fig: 3 UART Interfacing

d) 1-wire Interface

1-wire interface is an asynchronous half-duplex communication protocol developed by Maxim Dallas Semiconductor. It makes use of only a single line called DQ for communication and follows the master-slave communication model. One of the key feature of 1-wire bus is that it allows power to be sent along the signal wire as well.

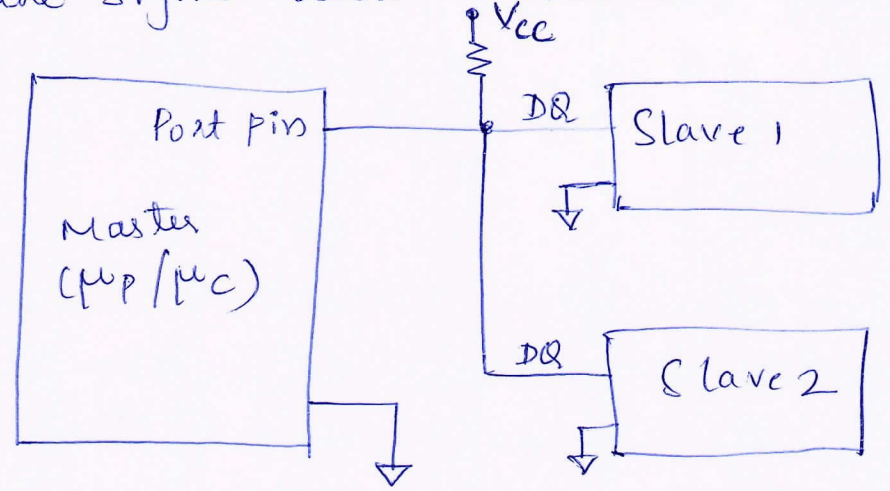


Fig 4: 1-wire Interface Bus

e) Parallel Interface

The host processor/ controller of the embedded System contains a parallel bus and the device which supports parallel bus can directly connect to this bus System.

The communication through the parallel bus is controlled by the control signal interface between the device and the host. The control signals for communication includes Read/write signal and device Select signal. The width of the parallel interface is determined by the data bus width of the host processor.

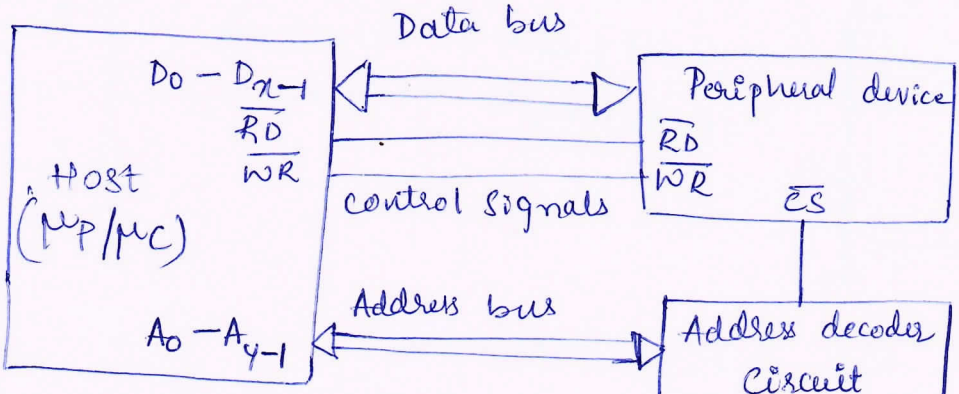


Fig 5: Parallel Interface Bus

6) b) Write a note on (i) Reset circuit (ii) Watch dog timer

Ans: a) Reset Circuit → [08 marks]

A 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.

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

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 Max 810 from Maxim Dallas. Figure 1, below illustrates a resistor capacitor based passive reset circuit for active high and low configurations. The reset pulse width can be adjusted by changing the resistance value R and capacitance value C.

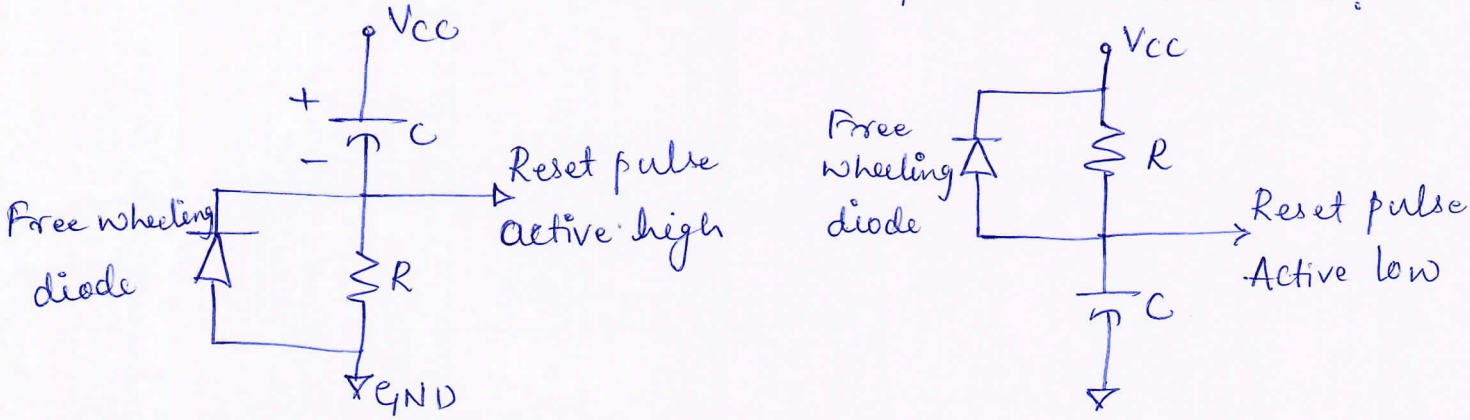


Fig 1: RC Based Reset Circuit

b) Watchdog Timer →

A watchdog timer is a hardware timer for monitoring the firmware execution. Depending on the internal implementation, the watchdog timer increments or decrements a free running counter with each clock pulse and generates a reset signal to reset the processor if the count reaches zero for a down counting watchdog, or the highest count value for an upcounting watchdog. If the watchdog counter is in the enabled state, the firmware can write a zero to it before starting the execution of a piece of code and the watchdog will start counting.

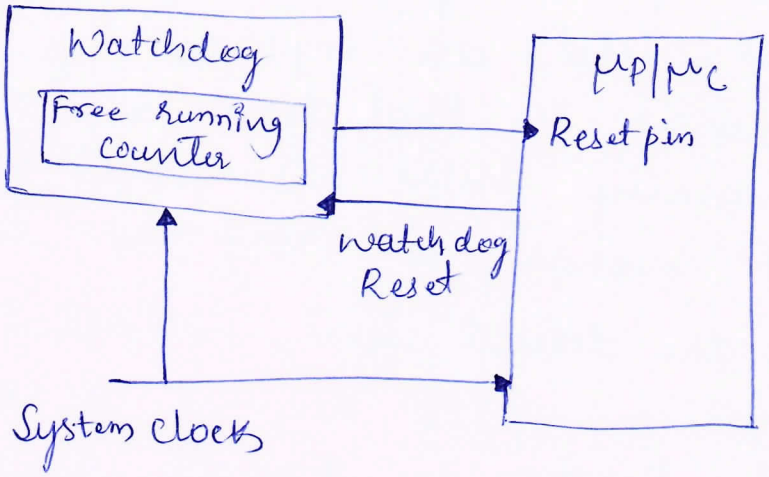


Fig 2: watchdog timer for firmware execution supervision

Figure 2 above illustrates the implementation of an external watchdog timer based microprocessor supervisor circuit for a small scale embedded system.

7) a) Explain the different characteristics of Embedded system in detail. [08 marks]

Ans:- The important characteristics of an Embedded System are

a) Application and domain specific
b) Reactive and Real time
c) operates in Harsh environment
d) Distributed
e) Small size and weight
f) Power concerns

a) Application and domain Specific

Each embedded system is having certain functions to perform and they are developed in such a manner to do the intended functions only. They cannot be used for any other purpose.

b) Reactive and Real time

Embedded Systems are in constant interaction with the real world through sensors and user defined input devices which are connected to

input port of the System. Any changes happening in the Real world are captured by the Sensors or input devices in Real time and the control algorithm running inside the unit reacts in a desired manner to bring the controlled o/p variables to the desired level. Embedded Systems Produce changes in o/p in response to the changes in the input. So they are generally referred as Reactive Systems.

c) Operates in Harsh environment

The environment in which the embedded S/ms deployed may be dusty one or a high temperature Zone or an area Subject to vibrations and shocks. Systems placed in Such areas should be capable to withstand all these adverse conditions.

d) Distributed

The term distributed means that embedded S/ms may be a part of larger Systems. Many numbers of Such distributed embedded Systems form a single large embedded control unit.

e) Small size and weight

Product aesthetics is an important factor in choosing a product. The product aesthetics (Size, weight, shape, Style etc) will be one of the deciding factors to choose a product. In embedded domain ~~also~~ Compactness is a Significant deciding factor. Most of the appln demands Small Size and low weight products.

f). Power Concerns

Power Management is another important factor that needs to be considered in designing embedded systems. Embedded Systems should be designed in such a way as to minimise the heat dissipation by the system. Power Management is a critical constraint in battery operated applicat - on. The more the power consumption the less is th battery life.

7)b) With a block diagram, mention the components and in the design of a washing machine and also explain its working. [08 Marks]

Ans: Washing machine is a typical example of an embedded System providing extensive support in home automation applications.

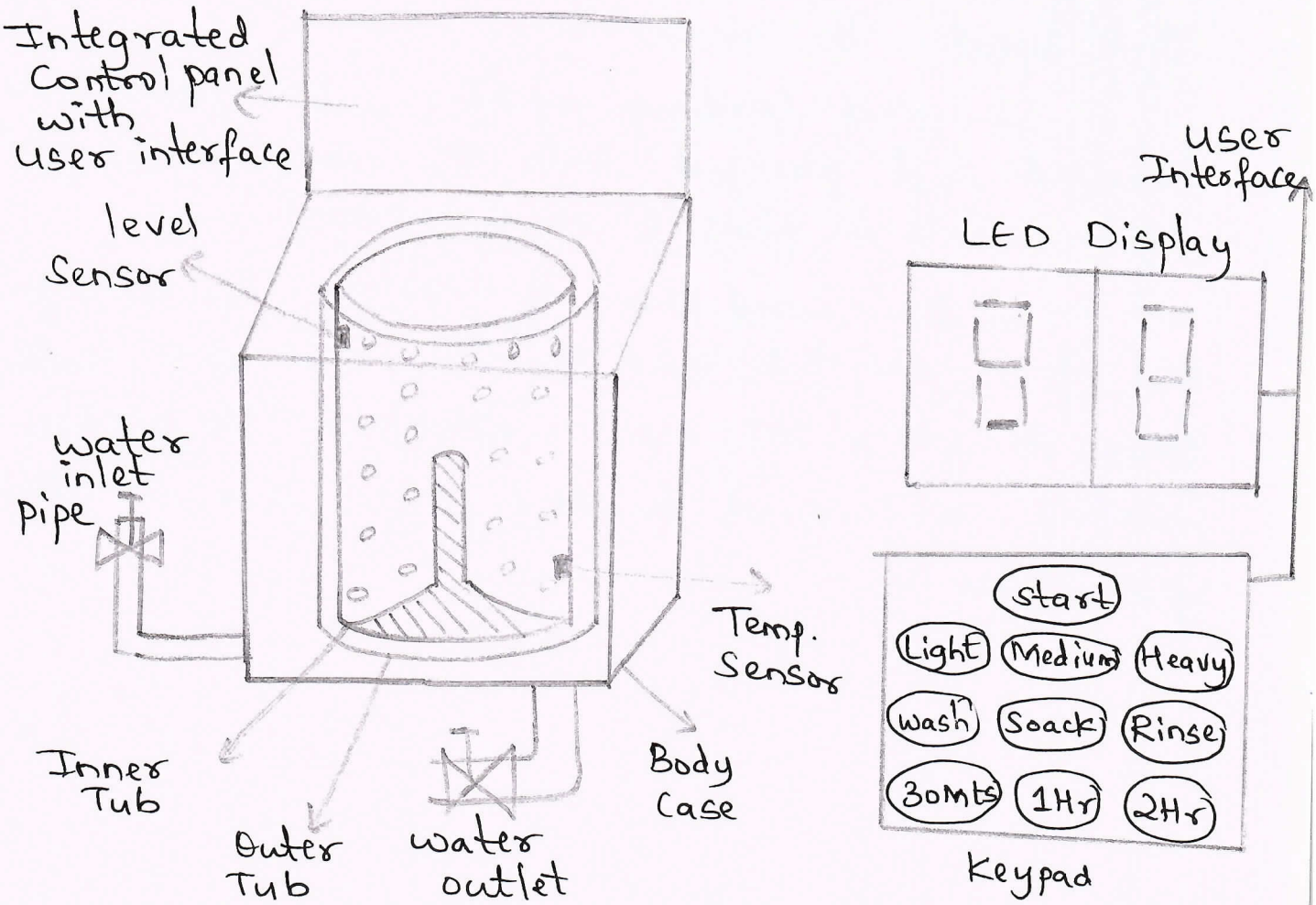


Fig: Washing Machine - Functional Block - diagram

→ The actuator part of the washing machine consists of a motorised agitator, tumble tub, water drawing pump and inlet valve to control the flow of water into the unit.

→ The sensor part consists of the water temperature sensor, level sensor etc.

→ The control part contains a microprocessor/ controller based board with interfaces to the sensors and actuators.

Working →

In top loading models the agitator of the machine twists back and forth and pulls the cloth down to the bottom of the tub. On reaching the bottom of the tub the clothes work their ways up to the top of the tub where the agitator grabs them again and repeats the mechanism.

In front loading machines, the clothes are tumbled and plunged into the water over and over again. This is first phase of washing.

In the second phase of washing, water is pumped out from the tub and the inner tub uses centrifugal force to wring out more water from the clothes by spinning at several hundred RPM. This is called a spin phase.

8)a) What is hardware and software co-design? Explain the fundamental design approaches in detail.

[10 marks]

Ans:- During the co-design process, the product requirements captured from the customer are converted into the system level needs or processing requirements. At this point of time it is not segregated as either hardware or software requirement, instead it is specified as functional requirement. The functions are simulated and verified for their performance & functionality. This is what is called as Hardware software co design

Fundamental design approaches are

a) Selecting the model
b) Selecting the architecture
c) Selecting the language
d) Partitioning system requirement into hardware and software.

a) Selecting the model

In hardware software co-design, models are used for capturing and describing the system characteristics. A model is a formal system consisting of objects and compositional rules.

b) Selecting the architecture

The architecture specifies how a system is going to implement interms of the number and types of different components and the interconnection among them.

Controller architecture, Datapath architecture, CISC, RISC, VLIW are the commonly used architectures.

c) Selecting the language

A programming language captures a computational model and maps it into architecture. A model can be captured using multiple programming languages like C, C++, C#, Java etc. for software implementations and languages like VHDL, System C, Verilog etc for hardware implementations.

d) Partioning System requirements into hardware and Software

From an implementation perspective, it may be possible to implement the system requirements in either hardware or software. It is a tough decision making task to figure out which one to opt. Various hardware-Software trade offs are used for making a decision on the hardware Software partitioning.

3.b) With FSM model, explain the design and operation of Automatic tea/coffee vending machine. [08 Marks]

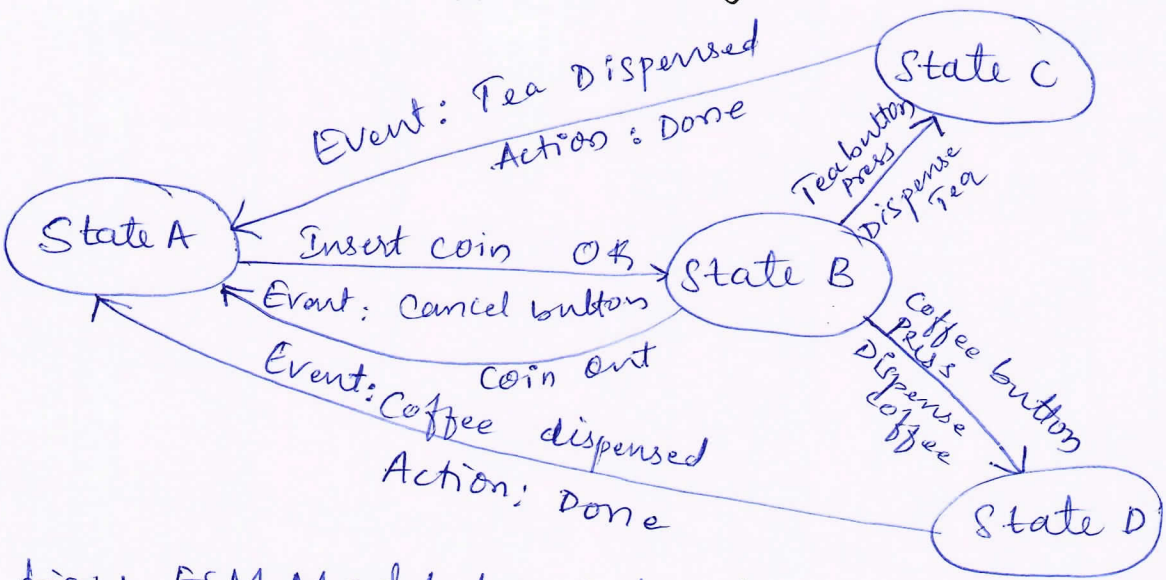


fig: 1 FSM Model for Automatic Tea/coffee Vending machine

The FSM contains four states namely, 'wait for coi 'wait for user input', 'Dispense Tea', 'Dispense coffee' The event 'Insert coin' transitions the state to wait for user input. The System stays in this state until a user input is received from the buttons 'cancel', 'Tea' or 'coffee'. If the event triggered in 'wait state' is cance button press, the coin is pushed out and the state transitions to wait for coin. If the event received in the wait state is either Tea button press or coffee button press, the state changes to dispense Tea and Dispense coffee respectively. Once the coffee or tea vending is over, the respective state transitions back to the wait for coin state.

9)a) Define process. Explain in detail the structure, memory organization and state transitions of the process. [08 marks]

Ans: Process:

A process is a program or part of it, in execution. Process is also known as an instance of a program in execution. A process is sequential in execution.

Structure of a process:

A process mimics a processor in properties and holds a set of registers, process status, a Program counter to point to the next executable instruction of the process, a stack for holding the local variable associated with the process

and the code corresponding to the process.

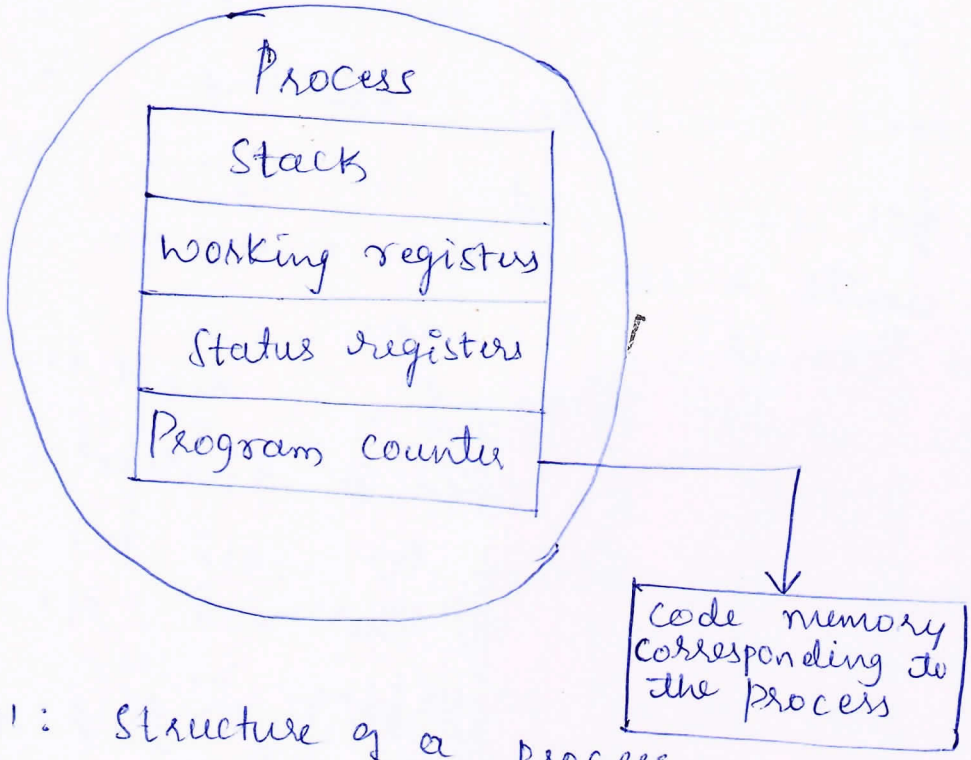


Fig 1: Structure of a process

A process which inherits all the properties of the CPU can be considered as a virtual processor, awaiting its turn to have its properties switched into physical Processor. When the process gets its turn, its registers and the program counter register becomes mapped to the physical registers of the CPU. From the memory Perspective, the memory occupied by the process is segregated into three regions namely, Stack memory, Data memory and code Memory

Process States and State transitions

The State at which a process is being created is referred as 'created State'. The Operating system recognised a process in the created State, but no resources are allocated to the process.

The State where a process is incepted into the memory and awaiting the processor time for execution, is known as Ready State.

At this stage, the process is placed in the Ready list queue maintained by the OS. The State where in the source code maintained by the OS. The State where in the source code instructions corresponding to the process is being executed is called Running State.

Running State is the State at which the proce execution happens. 'Blocked State / wait State' refe to a State where a running Process is temporarily Suspended from execution and does not have immed - ate access to resources. The blocked state might be invoked by various conditions like: the process enters a wait state for an event to occur or waiting for getting access to a shared resource.

A State where the process completes its execution is known as 'completed State'.

The transition of a process from one state to another is known as 'State Transition'.

9) b) Explain multi processing, multi tasking, & multi Programming.

Ans: In the operating System content multiprocessing describes the ability to execute multiple processes Simultaneously. Systems which are capable of perfo - rming multiprocessing are known as multiprocessor Systems. Multiprocessor Systems possess multiple CPUs and can execute multiple process simultaneo - usly.

Multi programming →

The ability of the operating system to have multiple programs in memory, which are ready for execution is referred as multiprogramming

Multitasking

The ability of an operating System to hold multiple processes in memory and switch the Processor (CPU) from executing one process to another process is known as multitasking. Multitasking creates the illusion of multiple tasks executing in parallel. Multitasking involve the switching of CPU from executing one task to another.

There are 3 types of Multitasking

a) **Co-operative Multitasking**

→ This is the most primitive form of multitasking in which task / process gets a chance to execute only when the currently executing task / process voluntarily relinquishes the CPU.

b) **Preemptive Multitasking**

Preemptive multitasking ensures that every task / process gets a chance to execute. When and how much time a process gets is dependent on the implementation of the preemptive Scheduling.

c) Non-Preemptive Multitasking

In non-preemptive multitasking, the process/task, which is currently given the CPU time, is allowed to execute until it terminates or enters the Blocked/wait state, waiting for an I/O or system resource.

10)a) Explain the simulator and emulator. [08 marks]

Ans: **Simulator**

Simulators simulate the target hardware and the firmware execution can be inspected using simulators. The features of the simulator based debugging are listed below.

1) Purely software based
2) Does not require a real target system
3) Very primitive
4) Lacks of Real time behaviour

Advantages

(1) No need for original target board
(2) Simulate I/O peripherals
(3) Simulates Abnormal conditions

Dis advantages

1) Deviation from Real behaviour
2) Lack of Real timeliness

Emulators

Emulator is a self contained hardware device which emulates the target CPU. The emulator hardware contains necessary emulation logic and it is hooked to the debugging application running on the development PC on one end and connects to the target board through some interface on the other end. In summary, the Simulator simulates the target board CPU and the emulator emulates the target board CPU.

→ Emulator will have

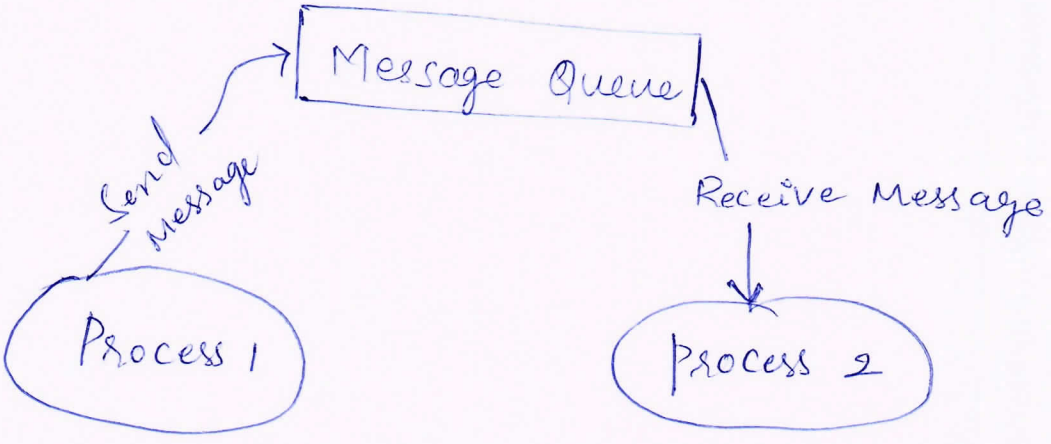
* Emulation Device
* Emulation Memory
* Emulator Control logic
* Device Adapters

10)b) Write a note on Message Passing. [08 marks]

Ans: Message passing is an asynchronous information exchange mechanism used for inter process/Thread communication. The major difference between shared memory and message passing technique is that, through shared memory lots of data can be shared whereas only limited amount of info/data is passed through message passing. Based on message passing operation between the Processes, message passing is classified into,

a) Message Queue →

Usually the process which wants to talk to another process posts the message to a FIFO queue called Message queue, which stores the messages temporarily in a System defined memo Object, to pass it to the desired process.



b) Mail box

Mailbox is an alternate form of 'Message queues' and it is used in certain Real Time os for IPC. The task/thread which wants to send a message to other tasks/threads creates a mailbox for posting the messages. The threads which are interested in receiving the messages posted to the mail box by the mail box Creator thread can subscribe to the mail box.

c) Signalling

Signalling is a primitive way of communication between process / threads. Signals are used for Synchronous notifications where one process or thread fires a Signal, indicating

occurance of a scenario which the other Processes / threads is waiting. Signals are not queued and they do not carry data.

→ whenever a specified signal occurs it is handled in a signal handler associated with the signal.