

Swati Kadivar

Homework3


Q1

1. Plated or Non plated holes if no specifications provided holes will be plated)
2. Non plated or plated slots/edges and cutout's
3. Various laminates including high temp (tg) FR4, Rogers, Polimide, and Aluminum Clad
- 4. Full range of finished board thicknesses
5. Different solder mask colors (default green)
6. Different silkscreen (legend) colors (default white)
7. Trace/Space down to 4/4 mils (8/8 typical)
8. UL Markings (Also 94V O upon request)
9. Lead Free logo markings on boards (upon request)
10. Controlled Dielectric and/or Controlled Impedance
11. Inner layer Cu. Wt. ½ oz. 3 oz.
12. Countersinks/Counterbores
13. Fiducials and/or Tooling Holes (no additional charge if requested)
14. Hole Tolerance +/- 0.003" (if requested)

2.

MicroSelectionRev3pt0 (1).xlsx - Excel		Maitreyee Rao			
File		Home		Insert	
Clipboard		Font		Alignment	
A2		Freescal			
1	Manufactu	Part Numb	Family	Core	#Cores
2	Freescal	MKL25212	Kinetis L	ARM M0+	32
3	Freescal	MKL26212	Kinetis L	ARM M0+	32
4	Freescal	MKL11DX12	Kinetis	ARM M4	32
5	Freescal	MK20DX12	Kinetis	ARM M4	32
6	TI	LM3S162	Stellaris		
7	TI	TM4C123	Tiva		
8	TI	TMS320C1	C5000		
9	Freescal	MK20DX12	Kinetis		
10	Freescal	MK20DX21	Kinetis		
11	TI	LM3S279	Stellaris		
12	TI	LM3S5B9	Stellaris		
13	TI	TMS320VC	C5000		
14	TI	LM3S979	Stellaris		
15	Freescal	MCIMX2861	MX28		
16	TI	AM1707C	Sitara		
17	Freescal	DSP8567	DSP56		
18	Freescal	MCIMX2571	MX257		
19	TI	LM3S9B9	Stellaris		
20	TI	TMS320C1	C6000		
21	TI	TMS320C1	C6000		
22	TI	TMS320C1	Delphin		
23	TI	LM3S9B9	Stellaris		
24	Freescal	DSP56371	DSP56		

1. Processor: TI : AM5K2E04

 AM5K2E04 ACTIVE	
Product details	Technical documentation
Design & development	Ordering & quality
Parameters	Features
Description	
Arm CPU	4 Arm Cortex-A15
Arm MHz (Max.)	1250, 1400
Co-processor(s)	Network co-processor
CPU	32-bit
Protocols	Ethernet
Ethernet MAC	8-Port 1Gb Switch
PCIe	4 PCIe Gen 2
Hardware accelerators	Packet Accelerator, Security Accelerator
Features	Networking
Operating system	Linux, RTOS
Rating	Catalog
Operating temperature range (C)	-40 to 100, 0 to 85

- **Peripherals**
 - Two PCIe Gen2 Controllers with Support for
 - Two Lanes per Controller
 - Supports Up to 5 GBaud
 - One HyperLink
 - Supports Connections to Other KeyStone Architecture Devices Providing Resource Scalability
 - Supports Up to 50 GBaud
 - 10-Gigabit Ethernet (10-GbE) Switch Subsystem
 - Two SGMII/XFI Ports with Wire Rate Switching and MACSEC Support
 - IEEE1588 v2 (with Annex D/E/F) Support
 - One 72-Bit DDR3/DDR3L Interface with Speeds Up to 1600 MT/s in DDR3 Mode
 - EMIF16 Interface
 - Two USB 2.0/3.0 Controllers
 - USIM Interface
 - Two UART Interfaces
 - Three I²C Interfaces
 - 32 GPIO Pins
 - Three SPI Interfaces
 - One TSIP
 - Support 1024 DS0s
 - Support 2 Lanes at 32.768/16.384.192 Mbps Per Lane

Part number ↓↑	Buy	TI.com inventory ↓↑	Qty Price (USD) ↓↑
AM5K2E04XABD25 ✓ ACTIVE	Not available	Not available	1ku \$73.920
AM5K2E04XABD4 ✓ ACTIVE	Not available	Not available	1ku \$88.704
AM5K2E04XABDA25 ✓ ACTIVE	Not available	Not available	1ku \$93.878
AM5K2E04XABDA4 ✓ ACTIVE	Out of stock	Out of stock 🔔 Notify me when available	1ku \$108.662 ▼


This processor has almost all the features as required, but price is ~70\$+ for all of AM5K2E04 family processors as they have multiple PCIe ports, higher end Ethernet switch subsystem and DDR3 support too.

1. Under \$50 - **NO**
2. DDR3 Memory Interface - **YES**
3. NAND Memory Interface - **YES**
4. 3 PCIe ports - **YES**

5. 3 Gigabit Ethernet ports - **YES**
6. 2 USB or ULPI ports - **YES**
7. 4 UARTs – **2 UARTs**
8. SPI and I2C port - **YES**
9. SD Card or MMC port - **YES**
10. Wake on LAN capability – **NO**

So if we can effort little higher price for given requirements then this processor is best fit. But if price is something we can not give up on then below processor is better choice at little less features matching with given requirements.

2. TI: TMS320DM8127

 TMS320DM8127 ✓ ACTIVE	
Product details	<u>Technical documentation</u>
Arm CPU	1 Arm Cortex-A8
Arm MHz (Max.)	1000
CPU	32-bit
Display type	2 LCD
Ethernet MAC	2-port 10/100/1000
PCIe	1 PCIe Gen 2
Hardware accelerators	Face Detect
Rating	Catalog
Operating temperature range (C)	-40 to 90, 0 to 90

Part number ↓↑	Buy	TI.com inventory ↓↑	Qty Price (USD) ↓↑
TMS320DM8127SCYE0 ✓ ACTIVE	<input type="text" value="Enter quantity"/> <input type="button" value="Add to cart"/>	790	1ku \$36.910 ▼

This processor is available at lower price but it has just 1 PCIe Gen2 available. So if we increase PCIe ports then price will go above 50\$.

<ul style="list-style-type: none"> • General-Purpose Memory Controller (GPMC) <ul style="list-style-type: none"> – 8- or 16-Bit Multiplexed Address and Data Bus – 512MB of Address Space Divided Among up to 8 Chip Selects – Glueless Interface to NOR Flash, NAND Flash (BCH/Hamming Error Code Detection), SRAM and Pseudo-SRAM – Error Locator Module (ELM) Outside of GPMC to Provide Up to 16-Bit or 512-Byte Hardware ECC for NAND – Flexible Asynchronous Protocol Control for Interface to FPGA, CPLD, ASICs, and so Forth 	<ul style="list-style-type: none"> • Dual USB 2.0 Ports With Integrated PHYs <ul style="list-style-type: none"> – USB2.0 High- and Full-Speed Clients – USB2.0 High-, Full-, and Low-Speed Hosts, or OTG – Supports End Points 0–15 • One PCI Express 2.0 Port With Integrated PHY <ul style="list-style-type: none"> – Single Port With One Lane at 5.0 GT/s – Configurable as Root Complex or Endpoint • Eight 32-Bit General-Purpose Timers (Timer1–8) • One System Watchdog Timer (WDT0) • Six Configurable UART/IrDA/CIR Modules
<ul style="list-style-type: none"> – UART0 With Modem Control Signals – Supports up to 3.6864 Mbps UART0/1/2 – Supports up to 12 Mbps UART3/4/5 – SIR, MIR, FIR (4.0 MBAUD), and CIR • Four Serial Peripheral Interfaces (SPIs) (up to 48 MHz) <ul style="list-style-type: none"> – Each With Four Chip Selects • Three MMC/SD/SDIO Serial Interfaces (up to 48 MHz) <ul style="list-style-type: none"> – Three Supporting up to 1-, 4-, or 8-Bit Modes • Four Inter-Integrated Circuit (I²C Bus) Ports 	<ul style="list-style-type: none"> – Independent QDMA Channels • Dual Port Ethernet (10/100/1000 Mbps) With Optional Switch <ul style="list-style-type: none"> – IEEE 802.3 Compliant (3.3-V I/O Only) – MII/RMII/GMII/RGMII Media Independent Interfaces
<ul style="list-style-type: none"> • Dual Port Gigabit Ethernet MACs (10/100/1000 Mbps) [Ethernet Switch] with MII/RMII/GMII/RGMII and MDIO interface supporting IEEE 1588 Time-Stamping, AVB, and Industrial Ethernet Protocols • Two USB ports with integrated 2.0 PHY • PCIe x1 GEN2 Compliant interface • Two 10-serializer McASP audio serial ports (with DIT mode) • Four quad-serializer McASP audio serial ports (with DIT mode) • One McBSP multichannel buffered serial port • Six UARTs with IrDA and CIR support • Four SPI serial interfaces • Three MMC/SD/SDIO serial interfaces • Four I²C master and slave interfaces • Parallel Camera Interface (CAM) • Up to 128 General-Purpose I/Os (GPIOs) • Eight 32-bit general-purpose timers • System watchdog timer • Dual DDR2, and DDR3 SDRAM interfaces • Flexible 8- or 16-bit asynchronous memory interface • Two Controller Area Network (DCAN) modules • Spin Lock • Mailbox 	

1. Under \$50 - **YES**
2. DDR3 Memory Interface -**YES**
3. NAND Memory Interface - **YES**
4. 3 PCIe ports – **1 PCIe port**
5. 3 Gigabit Ethernet ports - **YES**
6. 2 USB or ULPI ports - **YES**
7. 4 UARTs – **6 UARTs**
8. SPI and I2C port - **YES**

9. SD Card or MMC port - **YES**
 10. Wake on LAN capability - **NO**

3.

A. FreeRTOS : Its available at no cost, available as open source and x86 is supported

OS and RTOS Products											
1-Jul-2021											
RTOS	RTOS Source	Real Time	Audio Class	Pricing Info - RTOS + TCP/IP + USB Host +	Memory Requirements	Supports MMU	Response Time in usec	MultiCore Support	MISRA complie	Supports Intel	S
41 FreeRTOS	Open	Yes	Free + HCC		Yes				Yes	IA32 flat memory model, any x86 running in Real mode only	

B. Windows 10 IoT : Core is free; Core services are \$90 + \$0.30/device/month and It has storage of 2GB which should be there for a security camera to store the video recordings

C. MicroC/OS-III

MicroC/OS-III	Apache 2.0	open source	embedded	active	ARM7-9-11/Cortex-M1-3-4-A8/9, AVR, HC11/12/S12, ColdFire, Blackfin, MicroBlaze, NIOS, 8051, x86, Win32, H8S, M16C, M32C, MIPS, 68000, PIC24/dsPIC33/PIC32, MSP430, PowerPC, SH, StarCore, Renesas RX100-200-600-700, RL; STM32, ...	micrium.com/rtos/kernels
Milos	GNU GPL				Cortex-M3	www.milos.it
miosix	GNU GPL				stm32, efm32 e LPC2000	www.miosix.org

https://en.wikipedia.org/wiki/Comparison_of_real-time_operating_systems

RTOS	RTOS Source	Real Time	Audio Class
13 9 uC/OSIII	Micrium (Acquired by Silicon Labs)	Yes	For single product use: OS - \$6750, uC/USB-Host - \$8500, uC/UDP-IP - \$8000, uC/FTP - \$2250

Automotive application :

<https://www.researchgate.net/publication/251931744> The design and realization of vehicle real-time operating system based on UCOS-II

D.MQX– It is free, and opensource, and supports Freescale MCUs

OS and RTOS Products					
1-Jul-2021					
	RTOS	RTOS Source	Real Tim	Pricing Info - RTOS + TCP/IP + USB Host + Audio Class	Supports Freescale
21	MQX	Freescale	Yes	Free + HCC	All Vybrid, Coldfire, Kinetis K1x, K2X, K30_100, K40_100, and K6x; PowerQuicc I & II, PowerPC, PX, MPC5xx, MPC52xx

E. RIOT:

This OS is opensource, free(Low cost) and supports basic MCUs which are perfect for IOT applications like MSP430(TI), ARM + FREESCALE, supports Atmel.

Also it has RISC-V arch support with Microsemi SoCs

Also this OS is developer friendly which makes it easy to develop various iot applications and bug fixing, upgrade also easier.

<https://www.riot-os.org/>

F. Android :

Android is available at no cost, opensource, easy to upgrade and bug fix which makes it perfect for layman users. Also it supports many interfaces and so easy to port to new system/sale.

G. Windows 10 / MacOS – this OS are having high speed and multithreading environment support which allows us to switch between multiple tasks and work on them at the same time. Also it has huge memory access support so can be connected to external memory(hard drives).

4.

Let Y = Buy, D = Build

YC = Buy Costs, DC = Build Costs

YI = Buy Income, DI = Build Income

YPR = Buy Profit Rate,

DPR = Build Profit Rate

UC = unit cost

GP/U = Gross Profit/Unit

FC = Fixed Cost Rate,

T = time in months since start.

TP = Time in months since production

V = Volume in U/month 200per/month

In this case,

YC = FC * T +FDA*T= \$500k*T + \$200K*T for 6 months; then FC*T.

YPR = GP/U * V = 10000 * 200/month = 2000k/month;

YI = YPR*YTP – YC = 2000k*(T-6) – 500k * T - 200k*6; for T>6; @ T=24 YI = 22.8M

Cost savings/month = (YUC – DUC)*V = (2500-400)*200 = 4200/month

Breakeven for Build: Find T so that

DI >= YI DI = DPR*DTP – FC*T –NRE -FDA = V*(GP/U + (YUC-DUC))*(T-24) –FC*T - NRE -FDA

2000k*(T-6) – 500k * T - 200k*6 = 200*(10000 +2100)*(T-24) –500k*T –2400k

2000k*T – 12000k -500k * T – 1200k = 2420k *T – 58080k – 500k*T – 2400k

1500k*T – 13200k = 1920k*T – 60480k

420k*T = 47280k

T = 112.5 months

a. The profit made in the first 2 years, assuming the COTS solution. __\$22.8M__

**b. The costs savings/month for the design and build solution versus the COTS solution.
____\$4200/month_____**

c. The number of months until breakeven for the build solution

__112.5~113months__ or 24+89__

d. Circle what your company should do **BUY BUILD**

if build time is more than 2 years then it is better to buy a machine. Here in our case, build time is 24+89 > way more than 2 years. So I choose BUY option.

// missing header files – std libraries + math library for PI

Int a, b, c, d; **// a user should minimize the use of global variable as much as possible in embedded software design**

int main()

{

char x='Y';

while(x='Y') **// should check with '=='**

{

//...

cout<<"Continue? (Y/N)"; **// std libraries are not added so cout, cin will throw error**

cin>>x;

}

menu(x, a); **// menu function is defined after it's use and not declared previously!**

a=b; /* assignment **// missing closing comment which comments next line**

c=d; /* of both pairs */

if(0 < a < 5) c=b;

}

// return type of main is int but the statement is missing

_interrupt double compute_area(double radius) **// interrupt service routines never take either argument nor returns anything – this ISR definition won't work**

{

double area = PI * radius * radius; **// PI is not defined anywhere – user has to define own such constants at top of the code – or include math library**

printf("\nArea = %.2f", area);

return area;

}

void menu() **// Function definition + declaration should be present before it's use**

// no arguments here but calling the function with argument

{

//...

}

6.

We have 22 bit address line and 16x data bus(2 bytes).

Total : $2 * 2^{22} = 2 * 4,194,304$ bytes ~ 8Mb of memory can be accessed with 22-bit address line


S29AL008J55BFIR20

Manufacturer Cypress Semiconductor Corp

Manufacturer Product Number S29AL008J55BFIR20

Digi-Key Part Number 428-4185-ND

Product Attributes

TYPE	DESCRIPTION	SELECT	<input type="checkbox"/>
Category	Integrated Circuits (ICs) Memory	<input type="radio"/>	<input checked="" type="radio"/>
Mfr	Cypress Semiconductor Corp	<input type="checkbox"/>	
Series	AL-J	<input type="checkbox"/>	
Package	Tray 	<input type="checkbox"/>	
Part Status	Active	<input type="checkbox"/>	
Memory Type	Non-Volatile	<input type="checkbox"/>	
Memory Format	FLASH	<input type="checkbox"/>	
Technology	FLASH - NOR	<input type="checkbox"/>	
Memory Size	8Mb (1M x 8, 512K x 16)	<input type="checkbox"/>	
Memory Interface	Parallel	<input type="checkbox"/>	
Write Cycle Time - Word, Page	55ns	<input type="checkbox"/>	
Access Time	55 ns	<input type="checkbox"/>	
Voltage - Supply	3V ~ 3.6V	<input type="checkbox"/>	
Operating Temperature	-40°C ~ 85°C (TA)	<input type="checkbox"/>	
Mounting Type	Surface Mount	<input type="checkbox"/>	
Package / Case	48-VFBGA	<input type="checkbox"/>	
Supplier Device Package	48-FBGA (8.15x6.15)	<input type="checkbox"/>	
Base Product Number	S29AL008	<input type="checkbox"/>	

https://www.digikey.com/en/products/detail/cypress-semiconductor-corp/S29AL008J55BFIR20/3862778?utm_adgroup=Integrated%20Circuits%20%28ICs%29&utm_source=google&utm_medium=cpc&utm_campaign=Shopping_Supplier_Cypress%20Semiconductor%20Corp_0428_Co-

3.6

A sequential circuit, or more formally, a *finite-state machine*, is how we ultimately transform the behavior expressed in the state diagram into a hardware and/or software implementation.

Such circuits form the basis for the sophisticated computation and control algorithms that one finds at the core of most modern digital systems. Finite State Machines can be used to model problems in many fields, including mathematics, artificial intelligence, games or linguistics. A Finite State Machine, or FSM, is a computation model that can be used to simulate sequential logic, or, in other words, to represent and control execution flow.

3.30

Four fundamental parameters should be considered when designing or selecting a clock system or time base. For the basic clock, these parameters are:

- **Frequency and frequency range:**
- **Rise times and fall times:**
- **Precision and Stability :**

Because ripple counters are asynchronous, one should never decode any of the state variable combinations to generate a specific frequency. Building a higher frequency from a lower one is done using a phase locked loop (PLL). Crystal-based sources are generally the best solution for stable and accurate timing signals. When using such devices, one can start either with the basic crystal and then design the analog electronics necessary to implement the desired oscillator or buy a prepackaged oscillator. If the application demands greater accuracy and stability than are available with standard devices.

3.18

Timer design:

Below is the code for Arduino PWM pin programmed for motor:

```
int motorPin = 9;
int button1 =2;
int button2= 3;
```

```

int val_motor= 0;

void setup(){
    pinMode(motorPin, OUTPUT);
    pinMode(button1, INPUT);
    pinMode(button2, INPUT);

}

void loop() {
    if(button1 && val_motor!=255)
    {
        val_motor +=5;
        analogWrite(motorPin, val_motor);
        delay(30);
    }
    else if (button2 && val_motor!=0)
    {
        val_motor -=5;
        analogWrite(motorPin, val_motor);
        delay(30);
    }
}

```

Frequency selection:

Taking **8 – bit timer1** This timer counts from 0 - 255
 So it will roll after 255 timer ticks

This pin9 of Arduino has frequency of **490Hz:**

Here, I designed a system in which **duty cycle is controlled by button inputs**. On each button1 press, duty cycle will be increased by 5% and button2 press will decrease duty cycle by 5% as shown in above loop example.

11.15

Child processes and, consequently, their threads share the same firmware memory area. As a result, two different threads can be executing the same function at the same time. Functions using *only* local variables are inherently *reentrant*. That is, they can be simultaneously called and executed in two or more contexts.

Local variables are copied to the stack and each invocation will get new copies. On the other hand, functions that use global variables, variables local to the process, variables passed by reference, or shared resources are not reentrant. One must be particularly careful to ensure that all accesses to any common resources are coordinated. When designing the application, one must make certain that one thread cannot corrupt the values of the variables in a second. Any shared functions must be designed to be reentrant.

11.23

Below are the characteristics which are different between real time and non real time systems:

1. Time Constraints:

Time constraints related with real-time systems simply is time interval allotted for the response of the ongoing program. The task should be completed within this time interval. Real-time system is responsible for the completion of all tasks within their time intervals. Non real time systems do not have any deadlines or any time constraints associated with them.

2. Correctness:

Real-time systems are required to produce correct result within the given time interval. If the result is not obtained within the given time interval then also result is not considered correct. Non real time systems do not have any such hard requirement for correct result within given time.

3. Embedded:

In embedded systems, combination of hardware and software designed for a specific purpose. Real-time systems collect the data from the environment and passes to other components of the system for processing. Non real time systems can be embedded or non embedded systems.

4. Safety:

Safety is necessary for any system but real-time systems are expected to provide critical safety. Real-time systems are supposed to perform for a long time without failures. It also recovers very soon when failure occurs in the system and it does not cause any harm to the data and information. For non real time systems, safety is not a critical factor to consider, for example display of a toy or street light bulb.

5. Concurrency:

Real-time systems are concurrent that means it can respond to a several number of processes at a time. There are several different tasks going on within the system and it responds accordingly to every task in short intervals. This makes the real-time systems concurrent systems.

6. Distributed:

In various real-time systems, all the components of the systems are connected in a distributed way. The real-time systems are connected in such a way that different components are at different geographical locations. Thus all the operations of real-time systems are operated in distributed ways. Non real time systems do not have as such requirements on connections.

7. Stability:

Even when the load is very heavy, real-time systems respond in the time constraint i.e. real-time systems does not delay the result of tasks even when there are several task going on a same time. Stability in real-time systems is critical factor.

Ref: <https://www.geeksforgeeks.org/characteristics-of-real-time-systems/>

11.31

Below are the steps followed when interrupt has occurred:

1. The value of flag register is pushed into the stack. It means that first the value of SP (Stack Pointer) is decremented by 2 then the value of flag register is pushed to the memory address of stack segment.
 2. The value of starting memory address of CS (Code Segment) is pushed into the stack.
 3. The value of IP (Instruction Pointer) is pushed into the stack.
 4. IP is loaded from word location (Interrupt type) * 04.
 5. CS is loaded from the next word location.
 6. Interrupt and Trap flag are reset to 0.
-

11.10

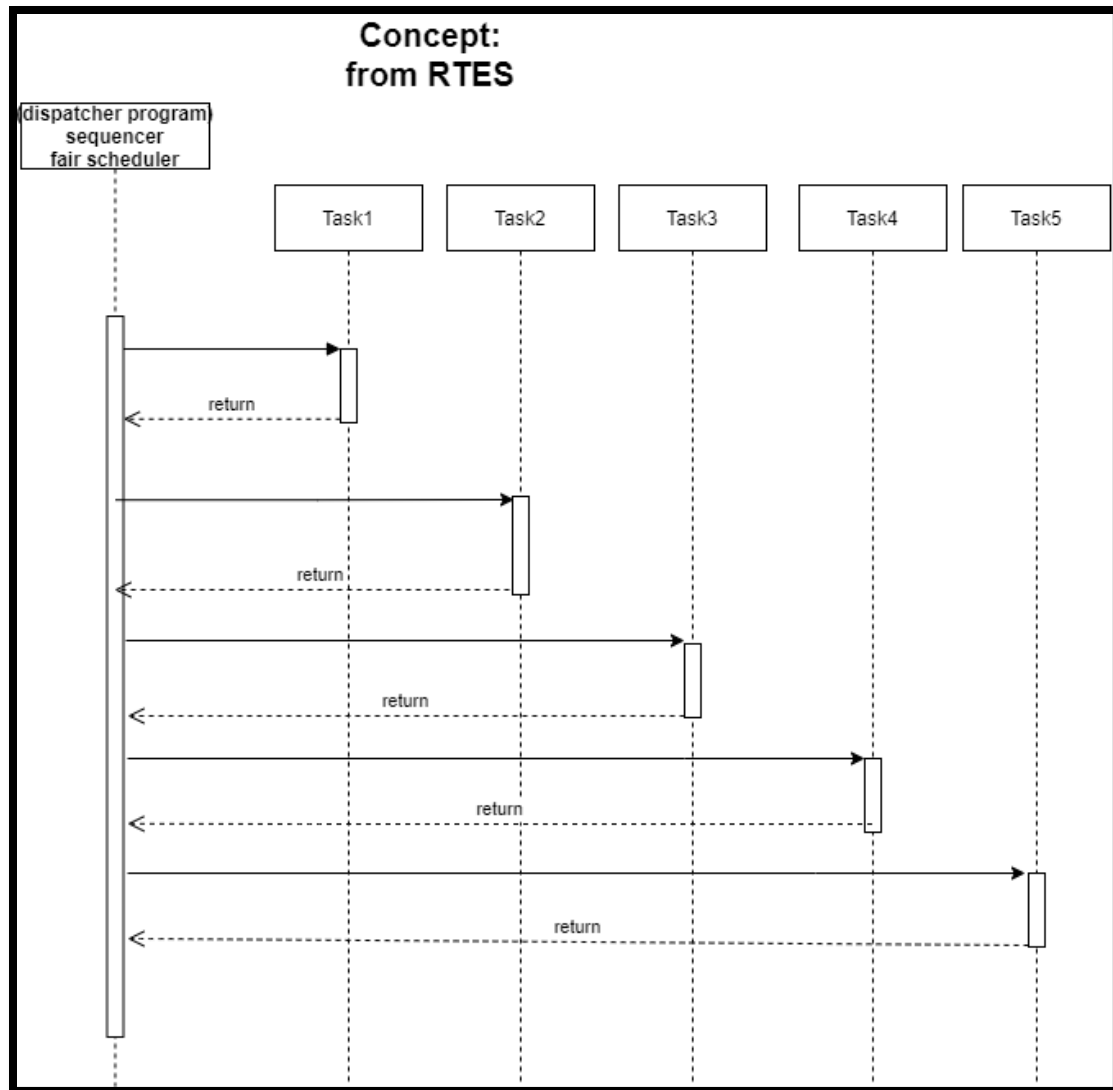
The systems in which the foreground tasks are those initiated by interrupt or by a real-time constraint that must be met, will be assigned the higher priority levels in the system – are best suited for foreground model.

In contrast, system with tasks which are noninterrupt driven and are assigned the lower priorities, Once started, the task will typically run to completion; however, it can be interrupted or preempted by any foreground task at any time – this type of systems can be modeled with background model.

Often separate ready queues will be maintained for the two types of tasks.

So to conclude, foreground modelling is good for real time systems while background modelling is good for nonreal time systems. The background tasks should include all those that do not have tight time constraints.

11.4



I have scheduled given tasks with **complete fair scheduler-Round Robin** as we don't know the priorities of the tasks. So to minimize the average waiting time, it will be better to give each task one time slice in sequence. If I schedule them with some other policy then when P(25) is executing, at that time other services may behave like they are never getting a CPU (starving for CPU time). So complete fair scheduler – Round robin policy will be perfect to schedule given tasks.

12.7

In a real-time context, a task that can be determined to always meet its timeliness constraints is said to be *schedulable*. A task that can be guaranteed to always meet all deadlines is said to be *deterministically schedulable*. Such a situation occurs when an event's worst case response time is less than or equal to the task's deadline. When all tasks can be scheduled, the overall system can be scheduled.

12.8

In addition to satisfying time constraints, a goal in formulating a task schedule is to keep the CPU as busy as possible, ideally close to 100%, but with some margin for additional tasks. Such a metric is referred to as *CPU utilization*.

Actual CPU utilization varies depending on the amount and type of managed computing tasks. Certain tasks require heavy CPU time, while others require less because of non-CPU resource requirements. CPU utilization may be used to gauge system performance. For example, a heavy load with only a few running programs may indicate insufficient CPU power support, or running programs hidden by the system monitor - a high indicator of viruses and/or malware.

12.11

Embedded system has the following tasks (exec. times, periods): P1(4,16), P2(3,8), P3(2,7).

- a. CPU Utilization = $(4/16 + 3/8 + 2/7) * 100 = (0.625) * 100 = \mathbf{62.5\%}$
- b. $n = 3 ; 3(2^{1/3}) - 1 = 0.77975 \Rightarrow \mathbf{77.97\%}$
here, CPU Utilization = 62.5% < 77.97%
so **YES, set of tasks can be scheduled using a rate-monotonic schedule**
- c. **UML sequence diagram:**

