CAMBRIDGE
INSTITUTE OF TECHNOLOGY
An Autonomous Institution

Elevium
Learn.Apply.Build
www.elevium.ai

Department of Electronics & Communication Engineering
in Collaboration with Nanochip skills Pvt. Ltd.
III Semester UG Examinations
Open Test Platform based assessment

Course Title: Embedded System Design
Course Code: EC221
Time: 09:00AM - 11:00AM
Max. Marks: 20
Date: 20-11-2024

Name: Vilas Hegde
USN: 1CD23EC182

Q. No:2
Answer:

key objectives: 1. Demonstrate dynamic memory allocation and deallocation:

Use functions like malloc and free from the C standard library for managing dynamic memory in RAM.

Simulate real-time system needs, such as allocating memory for tasks, buffers, or data structures based on system states.

2. Utilize ROM for immutable code:

Store application code, constants, and initialized data in the ROM to optimize RAM usage.

3. Analyze memory utilization: Monitor how much memory is used, wasted, and fragmented during the application's execution.

Log allocation and deallocation events to identify patterns of fragmentation.

4. Compare with static memory allocation:

Implement a static version of the same task and compare its performance and flexibility against the dynamic version.

Development Steps

1. Embedded Setup

Use the Keil uVision IDE or TI Code Composer Studio for coding and debugging.

Program and debug using an in-circuit debugger like JTAG or SWD for TM4C123.

2. System Design

Modules:

Task Scheduler: Handle periodic and aperiodic tasks.

Memory Manager: Implement and simulate malloc and free.

Fragmentation Monitor: Analyze and log memory usage.

Simulated Workload:

Simulate tasks like sensor data processing, communication buffers, and temporary storage.

3. Dynamic Memory Allocation

Implementation:

Allocate memory dynamically for tasks based on priority or need.

Use heap management techniques provided by the microcontroller (e.g., sbrk).

Code Example:

```
#define MEMORY_SIZE 128 *1024  //Size totale en bytes : 128 KB End
#define BLOCK_SIZE 32              /* Minimum block size for memory allocation */
xn typedef struct MemoryBlock {
size_t size;
struct MemoryBlock* next;
It's free.
} MemoryBlock;
 MemoryBlock* freeList = (MemoryBlock*)memory;  /* Initialize the simulated memory region */
void* malloc(size_t size) {
MemoryBlock* current = freeList;
while (current) {
if (current->free && current->size >= size)
current->free = 0;  /* Mark as in use */
return (void*)current + 1;  /RETURN THE MEMORY BLOCK SUCCEEDING THE HEADER/
END
current = current->next;
```

CONCLUSION

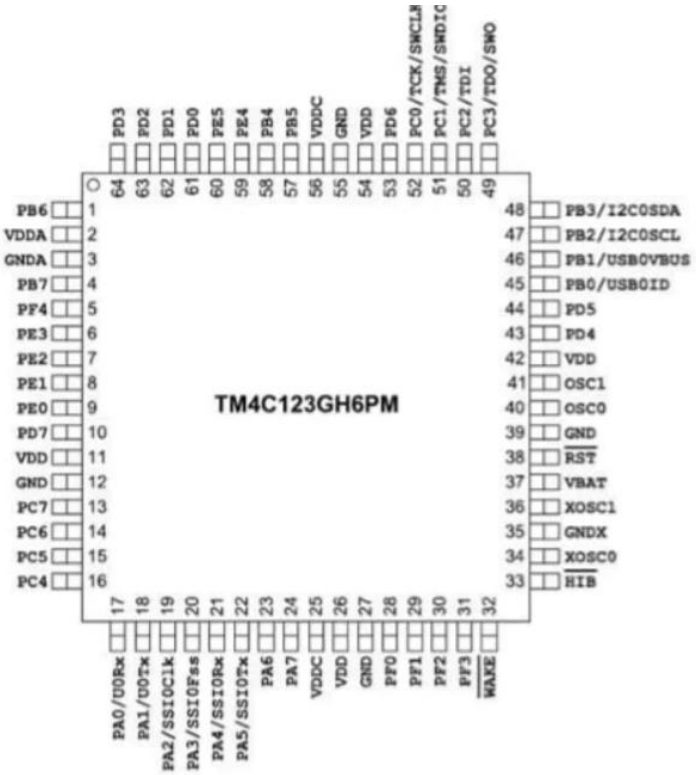
```
return NULL;  // Found no appropriate memory block
END

void free(void* ptr) {

MemoryBlock*
block = (MemoryBlock*)ptr - 1;  // get the header
block->free = 1;

void initMemory() {
free List->size = MEMORY_SIZE - sizeof(MemoryBlock);
freeList->free = 1;
freeList->next = NULL;

static size_t usedMemory = 0;
static size_t totalMemory = MEMORY_SIZE;

void* malloc(size_t size) {.}.

MemoryBlock* current = freeList;
while (current) {
if (current->free && current->size >= size)
CURRENT-FREE = 0;  // Mark as used
usedMemory += size;
return (void*)(current + 1); /* Return memory block after the header;
THE
current = current->next;
FIN
return NULL;  // No good block of memory found
.
void free(void* ptr) {}
```

```
MemoryBlock* block = (MemoryBlock*)ptr - 1;  // Get the header block-
>free = 1


void coalesceFreeBlocks() {


MemoryBlock *current = freeList;
: void reportMemoryUsage() { printf("Total Memory: %zu bytes", totalMemory);
 printf("Used Memory: %zu bytes", usedMemory);
printf("Free Memory: %zu bytes", totalMemory - usedMemory);
```

4) TM4C123 GPIO Ports

Address	7	6	5	4	3	2	1	0	Name
$400F.E608			GPIOF	GPIOE	GPIOD	GPIOC	GPIOB	GPIOA	SYSCTL_I
$4000.43FC	DATA	DATA	DATA	DATA	DATA	DATA	DATA	DATA	GPIO_POI
$4000.4400	DIR	DIR	DIR	DIR	DIR	DIR	DIR	DIR	GPIO_POI
$4000.4420	SEL	SEL	SEL	SEL	SEL	SEL	SEL	SEL	GPIO_POI
$4000.4510	PUE	PUE	PUE	PUE	PUE	PUE	PUE	PUE	GPIO_POI
$4000.451C	DEN	DEN	DEN	DEN	DEN	DEN	DEN	DEN	GPIO_POI
$4000.4524	1	1	1	1	1	1	1	1	GPIO_POI
$4000.4528	0	0	0	0	0	0	0	0	GPIO_POI
$4000.53FC	DATA	DATA	DATA	DATA	DATA	DATA	DATA	DATA	GPIO_POI
$4000.5400	DIR	DIR	DIR	DIR	DIR	DIR	DIR	DIR	GPIO_POI
$4000.5420	SEL	SEL	SEL	SEL	SEL	SEL	SEL	SEL	GPIO_POI
$4000.5510	PUE	PUE	PUE	PUE	PUE	PUE	PUE	PUE	GPIO_POI
$4000.551C	DEN	DEN	DEN	DEN	DEN	DEN	DEN	DEN	GPIO_POI
$4000.5524	1	1	1	1	1	1	1	1	GPIO_POI
$4000.5528	0	0	AMSEL	AMSEL	0	0	0	0	GPIO_POI
$4000.63FC	DATA	DATA	DATA	DATA	JTAG	JTAG	JTAG	JTAG	GPIO_POI
$4000.6400	DIR	DIR	DIR	DIR	JTAG	JTAG	JTAG	JTAG	GPIO_POI

$4000.6420	SEL	SEL	SEL	SEL	JTAG	JTAG	JTAG	JTAG	GPIO_PO
$4000.6510	PUE	PUE	PUE	PUE	JTAG	JTAG	JTAG	JTAG	GPIO_PO
$4000.651C	DEN	DEN	DEN	DEN	JTAG	JTAG	JTAG	JTAG	GPIO_PO
$4000.6524	1	1	1	1	JTAG	JTAG	JTAG	JTAG	GPIO_PO
$4000.6528	AMSEL	AMSEL	AMSEL	AMSEL	JTAG	JTAG	JTAG	JTAG	GPIO_PO
$4000.73FC	DATA	DATA	DATA	DATA	DATA	DATA	DATA	DATA	GPIO_PO
$4000.7400	DIR	DIR	DIR	DIR	DIR	DIR	DIR	DIR	GPIO_PO
$4000.7420	SEL	SEL	SEL	SEL	SEL	SEL	SEL	SEL	GPIO_PO
$4000.7510	PUE	PUE	PUE	PUE	PUE	PUE	PUE	PUE	GPIO_PO
$4000.751C	DEN	DEN	DEN	DEN	DEN	DEN	DEN	DEN	GPIO_PO
$4000.7524	CR	1	1	1	1	1	1	1	GPIO_PO
$4000.7528	0	0	AMSEL	AMSEL	AMSEL	AMSEL	AMSEL	AMSEL	GPIO_PO
$4002.43FC			DATA	DATA	DATA	DATA	DATA	DATA	GPIO_PO
$4002.4400			DIR	DIR	DIR	DIR	DIR	DIR	GPIO_PO
$4002.4420			SEL	SEL	SEL	SEL	SEL	SEL	GPIO_PO
$4002.4510			PUE	PUE	PUE	PUE	PUE	PUE	GPIO_PO
$4002.451C			DEN	DEN	DEN	DEN	DEN	DEN	GPIO_PO
$4002.4524			1	1	1	1	1	1	GPIO_PO
$4002.4528			AMSEL	AMSEL	AMSEL	AMSEL	AMSEL	AMSEL	GPIO_PO
$4002.53FC				DATA	DATA	DATA	DATA		


TM4C123GH6PM
PB6 1
VDDA 2
GNDA 3
PB7 4
PF4 5
PE3 6
PE2 7
PE1 8
PE0 9
PD7 10
VDD 11
GND 12
PC7 13
PC6 14
PC5 15
PC4 16
64 PD3
63 PD2
62 PD1
61 PD0
60 PE5
59 PE4
58 PB4
57 PB5
56 VDDC
55 GND
54 VDD
53 PD6
52 PC0/TCK/SWCLK
51 PC1/TMS/SWDIO
50 PC2/TDI
49 PC3/TDO/SWO
48 PB3/I2C0SDA
47 PB2/I2C0SCL
46 PB1/USB0VBUS
45 PB0/USB0ID
44 PD5
43 PD4
42 VDD
41 OSC1
40 OSC0
39 GND
38 RST
37 VBAT
36 XOSC1
35 GNDX
34 XOSC0
33 HIB
17 PA0/U0Rx
18 PA1/U0Tx
19 PA2/SSI0Clk
20 PA3/SSI0Fss
21 PA4/SSI0Rx
22 PA5/SSI0Tx
23 PA6
24 PA7
25 VDDC
26 VDD
27 GND
28 PF0
29 PF1
30 PF2
31 PF3
32 WAKE

1. TM4C123 Block Diagram