Operating Systems Design for Embedded Environments

**Practice 4 3.0 OS Task Context Switch**

|  |  |
| --- | --- |
| **Project:** | **OS Context Switch** |
| **Team:** | **Team 4** |
| **Date:** | **16/05/2014** |
| **Comments:** | **This is just an initial draft, can be changed if needed.** |

**Table of Contents**

[Operating Systems Design for Embedded Environments 1](#_Toc388111840)

[1. SW Conceptual design 3](#_Toc388111841)

[2 Function Description and Dynamic Behavior 6](#_Toc388111842)

[*2.1* Function *void DisableAllInterrupts (void)* 6](#_Toc388111843)

[*2.2* Function *void EnableAllInterrupts (void)* 6](#_Toc388111844)

[*2.3* Function *Mem\_ReturnType Mem\_Alloc(Mem\_SizeType)* 6](#_Toc388111845)

[*2.4* Function *void Task\_200ms (void)* 6](#_Toc388111846)

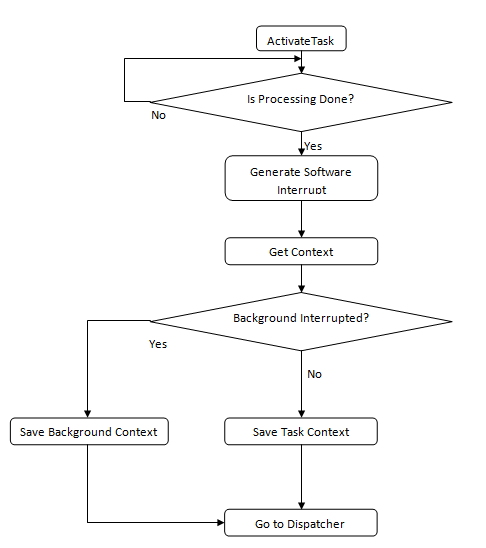
[*2.5* Function *void Task\_100ms (void)* 7](#_Toc388111847)

[2.6 Context Switch Assembler 7](#_Toc388111848)

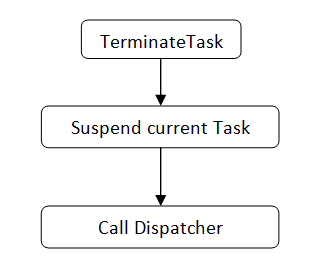
[*2.7* Scheduler task callback functions 7](#_Toc388111849)

# SW Conceptual design

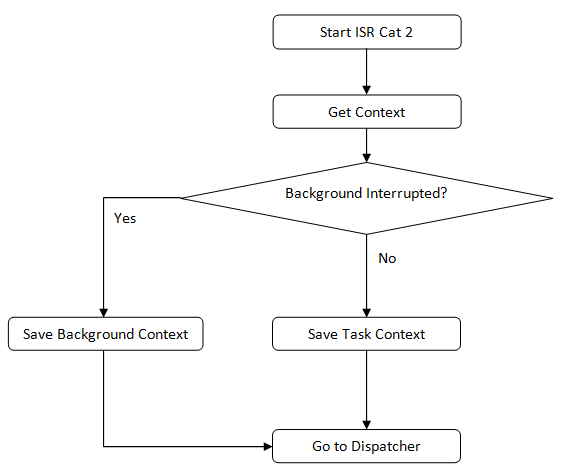
Dispatcher called after an ActivateTask routine



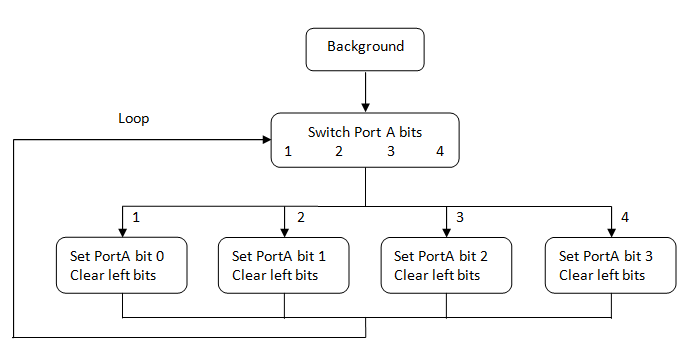
Dispatcher called after TerminateTask



Switch Context in an ISR category 2 and Dispatcher call at the end



Rotabit



# Function Description and Dynamic Behavior

## Function void DisableAllInterrupts (void)

|  |  |  |
| --- | --- | --- |
| **Description** | *void DisableAllInterrupts ( void )* | |
| **Return Value** | *Void* | |
| **Precondition** | *Within the critical section, no API service calls are allowed.*  *How the system interrupts are disabled will differ between implementations and between microcontrollers* | |
| **Parameters** | *None* |  |
| **Post Conditions** | *System service shall support nested calls in order to avoid any enabling interrupt when not yet required.* | |

**Dynamic Behavior**

API service saves the current state of all interrupts, disables all interrupts that are enabled, and identifies the beginning of a critical section.

## Function void EnableAllInterrupts (void)

|  |  |
| --- | --- |
| **Description** | *void EnableAllInterrupts ( void )* |
| **Return Value** | *void* |
| **Precondition** | *Within the critical section, no API service calls are allowed. If DisableAllInterrupts () was not previously called, the action taken is undefined by the specification.* |
| **Post condition** | *System service shall support nested calls in order to avoid any enabling interrupt when not yet required.* |
| **Error Conditions** |  |

**Dynamic Behavior**

API service enables all interrupts that were enabled prior to the previous call to DisableAllInterrupts and identifies the end of a critical section.

## Function Mem\_ReturnType Mem\_Alloc(Mem\_SizeType)

|  |  |
| --- | --- |
| **Description** | *Mem\_ReturnType Mem\_Alloc(Mem\_SizeType)* |
| **Return Value** | *Mem\_ReturnType - Returns a pointer to the allocated memory, or NULL if the request fails* |
| **Precondition** | *None* |
| **Parameters** | *Mem\_SizeType - Size of memory in bytes to allocate* |
| **Error Conditions** |  |

**Dynamic Behavior**

Dynamic memory allocation function

## Function void Task\_200ms (void)

|  |  |
| --- | --- |
| **Description** | *void Task\_200ms (void)* |
| **Return Value** | *void* |
| **Precondition** | *None* |
| **Post condition** | *None* |

**Dynamic Behavior**

Timer event periodic task callback function 200ms.

## Function void Task\_100ms (void)

|  |  |
| --- | --- |
| **Description** | *void Task\_100ms (void)* |
| **Return Value** | *None* |
| **Parameters** | *None* |
| **Post condition** | *None* |
| **Error Conditions** |  |

**Dynamic Behavior**

Timer event periodic task callback function 100ms.

## Context Switch Assembler

\_\_asm

{

*PULD ; (CCR) Pull stack into the CPU Register D*

*STD CCR\_ContextSaving\_u16 ; Store the CPU Register D value in fixed memory*

*PULD ; (D || BA) Pull the stack into the CPU Register D*

*STD D\_ContextSaving\_u16 ; Store the CPU Register D value in fixed memory*

*PULD ; (IX) Pull the stack into the CPU Register D*

*STD IY\_ContextSaving\_u16 ; Store the CPU Register D value in fixed memory*

*PULD ; (IY) Pull the stack into the CPU Register D*

*STD IX\_ContextSaving\_u16 ; Store the CPU Register D value in fixed memory*

*PULD ; (PC) Pull the stack into the CPU Register D*

*STD PC\_ContextSaving\_u16 ; Store the CPU Register D value in fixed memory*

*PULA ; (P\_PAGE) Pull the stack into the CPU Register D*

*STAA PPAGE\_ContextSaving\_u8 ; Store the CPU Register A value in fixed memory*

*STS SP\_ContextSaving\_u16 ; (SP) Store Stack Pointer in fixed memory*

}

## Scheduler task callback functions

|  |  |
| --- | --- |
| **Task** | **Description & Period** |
| *(void) SchM\_Tsk\_1ms (void);* | *Callback function for 1ms* |
| *(void) SchM\_Tsk\_4ms (void);* | *Callback function for 4ms* |
| *(void) SchM\_Tsk\_8ms (void);* | *Callback function for 8ms* |
| *(void) SchM\_Tsk\_16ms (void);* | *Callback function for 16ms* |
| *(void) SchM\_Tsk\_32ms (void);* | *Callback function for 32ms* |
| *(void) SchM\_Tsk\_64ms (void);* | *Callback function for 64ms* |
| *void Task\_200ms (void)* | *Callback function for 200ms* |
| *void Task\_100ms (void)* | *Callback function for 100ms* |

**Dynamic Behavior**

Callback functions for periods requested.