OS CDD

for

LCD\_LED\_UART

Prepared by: Norhan Nassar

Mar 27, 2020

**Contents**

[**1.** **Global Variables** 1](#_Toc36227208)

[**2.** **Configurations** 1](#_Toc36227209)

[**3.** **OS APIs** 2](#_Toc36227210)

# **Global Variables**

**N/A**

# **Configurations**

# MAXTASKNUM: Tasks number in scheduler.

# TICKTIMEUS: Tick time in micro second.

# Task struct: for each task you must create its structure that contains of

* Runnable:

Pointer to Task runnable function.

* Periodicity:

In micro sec.

* First delay:

first delay to start count from in micro sec.

1. **OS APIs**

|  |  |  |  |
| --- | --- | --- | --- |
| **Component Name** | OS | | |
| **API Name** | void Sched\_SystickCbf(void) | | |
| **Return type** | It’s an enum of Error\_Status, it returns OK or NOT\_OK   |  |  | | --- | --- | | OK | 0 | | NOT\_OK | 1 | | | |
| **Input signal** | N/A | Output signal | N/A |
| **Description** | The functionality of this API is to set the call back function of systick when tick  has been arrived. | | |
| **Type (Public/Private)** | Public | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Component Name** | OS | | |
| **API Name** | STD\_ERROR Sched\_init(void) | | |
| **Return type** | It’s an enum of Error\_Status, it returns OK or NOT\_OK   |  |  | | --- | --- | | OK | 0 | | NOT\_OK | 1 | | | |
| **Input signal** | N/A | Output signal | N/A |
| **Description** | The functionality of this API is to init systick, set time to systick according to  tick time in configuration file and set call back function for systick | | |
| **Type (Public/Private)** | Public | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Component Name** | OS | | |
| **API Name** | STD\_ERROR Sched\_CreatTask(task\_t\* task,u32 taskPriority) | | |
| **Return type** | It’s an enum of Error\_Status, it returns OK or NOT\_OK   |  |  | | --- | --- | | OK | 0 | | NOT\_OK | 1 | | | |
| **Input signal** | task:   * Type: pointer to task struct * Description: It takes * Task runnable. * Task periodicity. * Task firstDelayTicks   taskPriority:   * Type: pointer to task struct * Description: it will return error if it exceeds the array of tasks limit and return error -> if TICKTIMEUS in cfg file is 0 | Output signal | N/A |
| **Description** | The functionality of this API is to create task and set its priority. | | |
| **Type (Public/Private)** | Public | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Component Name** | OS | | |
| **API Name** | STD\_ERROR Task\_Suspend(taskRunnable\_t taskRunnable) | | |
| **Return type** | It’s an enum of Error\_Status, it returns OK or NOT\_OK   |  |  | | --- | --- | | OK | 0 | | NOT\_OK | 1 | | | |
| **Input signal** | taskRunnable:   * Type: Pointer to function * Description: pointer to task that you want to suspend it. | Output signal | N/A |
| **Description** | The functionality of this API is to suspend the task. | | |
| **Type (Public/Private)** | Public | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Component Name** | OS | | |
| **API Name** | STD\_ERROR Task\_Resume(taskRunnable\_t taskRunnable) | | |
| **Return type** | It’s an enum of Error\_Status, it returns OK or NOT\_OK   |  |  | | --- | --- | | OK | 0 | | NOT\_OK | 1 | | | |
| **Input signal** | taskRunnable:   * Type: Pointer to function * Description: pointer to task that you want to resume it. | Output signal | N/A |
| **Description** | The functionality of this API is to resume the task. | | |
| **Type (Public/Private)** | Public | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Component Name** | OS | | |
| **API Name** | STD\_ERROR Task\_Pause(taskRunnable\_t taskRunnable, u8 Period) | | |
| **Return type** | It’s an enum of Error\_Status, it returns OK or NOT\_OK   |  |  | | --- | --- | | OK | 0 | | NOT\_OK | 1 | | | |
| **Input signal** | taskRunnable:   * Type: Pointer to function * Description: pointer to task that you want to resume it.   Period:   * Type: u8 * Description: Period that you want to pause the task while. | Output signal | N/A |
| **Description** | * The functionality of this API is to pause the task. | | |
| **Type (Public/Private)** | Public | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Component Name** | OS | | |
| **API Name** | STD\_ERROR Task\_Delete(taskRunnable\_t taskRunnable) | | |
| **Return type** | It’s an enum of Error\_Status, it returns OK or NOT\_OK   |  |  | | --- | --- | | OK | 0 | | NOT\_OK | 1 | | | |
| **Input signal** | taskRunnable:   * Type: Pointer to function * Description: pointer to task that you want to resume it. | Output signal | N/A |
| **Description** | The functionality of this API is to delete the task. | | |
| **Type (Public/Private)** | Public | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Component Name** | OS | | |
| **API Name** | void Sched\_Runnable(void) | | |
| **Return type** | It’s an enum of Error\_Status, it returns OK or NOT\_OK   |  |  | | --- | --- | | OK | 0 | | NOT\_OK | 1 | | | |
| **Input signal** | N/A | Output signal | N/A |
| **Description** | The functionality of this API is to set the runnable of the scheduler. | | |
| **Type (Public/Private)** | Public | | |