# Moving Robot API design documentation

## DIO Module

| Function name | Dio_InitPortPin | | |
|---|---|---|---|
| Arguments | Input | DIO_port | Uint8 |
| | | Port number / symbolic name | |
| | | DIO_Pin | Uint8 |
| | | Pin number / symbolic name | |
| | | DIO_Direction | Uint8 |
| | | Define port pin direction. Available modes:<br>• Input<br>• Output push pull | |
| | Output | | |
| | Input/ Output | | |
| Return | E_OK | 1 | |
| | E_NOT_OK | 0 | |
| Description | Responsible for initializing a port pin direction. Must be specified before read/write access on a pin. | | |

| Function name | Dio_Read | | |
|---|---|---|---|
| Arguments | Input | DIO_Port | Uint8 |
| | | Port number / symbolic name | |
| | | DIO_Pin | Uint8 |
| | | Pin number / symbolic name | |
| | Output | DIO_LEVEL | Uint8* |
| | | Pointer to Physical level of the specified pin. | |

| | | |
|---|---|---|
| | | Available values: STD_ON / STD_OFF |
| | Input/ Output | |
| Return | E_OK | 1 |
| | E_NOT_OK | 0 |
| Description | Responsible for reading the physical current value of a hardware port pin. It should be able to read the value of the pin whether it's input or output without affecting its current state.<br>If the pin is uninitialized, the function should return an error and not do anything.<br>For the output parameters it must check for a null pointer exception before proceeding. | |

| Function name | Dio_Write | | |
|---|---|---|---|
| Arguments | Input | DIO_Port | Uint8 |
| | | Port number / symbolic name | |
| | | DIO_Pin | Uint8 |
| | | Pin number / symbolic name | |
| | | DIO_LEVEL | Uint8 |
| | | Physical level to write on the specified pin.<br>Available values: STD_ON / STD_OFF | |
| | Output | | |
| | Input / Output | | |
| Return | E_OK | 1 | |
| | E_NOT_OK | 0 | |
| Description | Responsible for writing the physical value of a hardware port pin. It should be able to write the value of the pin if it's output without.<br>If the pin is uninitialized, the function should return an error and not do anything. | | |

## Timer Module

<table>
<tr><td>Function name</td><td colspan="3">TMR_init</td></tr>
<tr><td rowspan="13">Arguments</td><td rowspan="11">Input</td><td>TMR_Config</td><td>structure</td></tr>
<tr><td colspan="2">Pointer to structure address holding timer configuration parameters.<br><br>Structure members must include:</td></tr>
<tr><td>TMR_Channel</td><td>Uint8</td></tr>
<tr><td colspan="2">Available values are the available number of channels of timers.</td></tr>
<tr><td>TMR_Operation_Mode</td><td>Uint8</td></tr>
<tr><td colspan="2">Define the operation mode: Normal/CC ..etc.</td></tr>
<tr><td>TMR_CLK</td><td>Uint8</td></tr>
<tr><td colspan="2">Define the clock source and pre-scalar<br>Some of the valid values: Clk, Clk/8 , External</td></tr>
<tr><td colspan="2">Structure members may include:</td></tr>
<tr><td>TMR_IRQ_EN</td><td>Uint8</td></tr>
<tr><td colspan="2">Configuration to enable or disable interrupts on timer over Flow or other interrupt flags depending on the channel configuration.</td></tr>
<tr><td>Output</td><td colspan="2"></td></tr>
<tr><td>Input / Output</td><td colspan="2"></td></tr>
<tr><td rowspan="2">Return</td><td>E_OK</td><td colspan="2">1</td></tr>
<tr><td>E_NOT_OK</td><td colspan="2">0</td></tr>
<tr><td>Description</td><td colspan="3">Function responsible for initializing a timer channel according to configuration parameters.<br>This function isn’t supposed to handle PWM mode.<br>For the input parameter TMR_Config a null pointer check must be performed. If a nullptr detected, it should return an error and do nothing.<br>If the TMR_Channel isn’t valid, the function should return an error and do nothing.<br>User must initialize the channel using this function before performing start/stop operations.</td></tr>
</table>

| Function name | TMR_Start | | |
|---|---|---|---|
| Arguments | Input | TMR_Channel | Uint8 |
| | | Channel number for the timer to start. | |
| | | TMR_InitalValue | uint16 |
| | | Initial value to store in the timer register | |
| | Output | | |
| | Input / Output | | |
| Return | E_OK | 1 | |
| | E_NOT_OK | 0 | |
| Description | Function responsible for starting the work of the timer specified with a starting value TMR_InitalValue in the timer/counter register. | | |

| Function name | TMR_Stop | | |
|---|---|---|---|
| Arguments | Input | TMR_Channel | uint8 |
| | | Channel number for the timer to stop. | |
| | Output | | |
| | Input / Output | | |
| Return | E_OK | 1 | |
| | E_NOT_OK | 0 | |
| Description | Function responsible for stop the operation of the timer channel specified by the parameter TMR_Channel. | | |

| Function name | TMR_GetStatus | | |
|---|---|---|---|
| Arguments | Input | TMR_Channel | Uint8 |
| | | Channel number for the timer to start. | |
| | Output | TMR_status | uint8* |
| | | Pointer to where to store the timer status. | |

| | | |
|---|---|---|
| | | A value 1 should be stored if overflow occurred and 0 otherwise. |
| | Input / Output | |
| Return | E_OK | 1 |
| | E_NOT_OK | 0 |
| Description | Function responsible for starting the work of the timer specified with a starting value TMR_InitalValue in the timer/counter register. | |

## PWM module

<table>
<tr><td>Function name</td><td colspan="3">PWM_Init</td></tr>
<tr><td rowspan="8">Arguments</td><td rowspan="6">Input</td><td>PWM_Config</td><td>structure</td></tr>
<tr><td colspan="2">Pointer to structure address holding pwm configuration parameters.<br>Structure members must include:</td></tr>
<tr><td>TMR_Channel</td><td>0,1,2</td></tr>
<tr><td>TMR_Operation_Mode</td><td>PWM/Fast PWM</td></tr>
<tr><td>TMR_CLK</td><td>Clk<br>Clk/8<br>.<br>.<br>External</td></tr>
<tr><td colspan="2"></td></tr>
<tr><td>Output</td><td colspan="2"></td></tr>
<tr><td>Input / Output</td><td colspan="2"></td></tr>
<tr><td rowspan="2">Return</td><td>E_OK</td><td colspan="2">1</td></tr>
<tr><td>E_NOT_OK</td><td colspan="2">0</td></tr>
<tr><td>Description</td><td colspan="3">Function responsible for initializing a timer channel for PWM operation according to configuration parameters.<br>This function isn’t supposed to handle other timer modes.<br>For the input parameter PWM_Config a null pointer check must be performed. If a nullptr detected, it should return an error and do nothing.<br>If the TMR_Channel isn’t valid for PWM operation, the function should return an error and do nothing.</td></tr>
</table>

| | |
|---|---|
| | User must initialize the channel using this function before performing start/stop operations. |

| | | | |
|---|---|---|---|
| Function name | PWM_Start | | |
| Arguments | Input | TMR_Channel | Uint8 |
| | | Channel number for the pwm to start. | |
| | Output | | |
| | Input / Output | | |
| Return | E_OK | 1 | |
| | E_NOT_OK | 0 | |
| Description | Function responsible for starting the work of the pwm specified.<br>If the TMR_Channel passed isn't available for PWM operation, the function should return an error and do nothing. | | |

| | | | |
|---|---|---|---|
| Function name | PWM_Stop | | |
| Arguments | Input | TMR_Channel | Uint8 |
| | | Channel number for the pwm to stop. | |
| | Output | | |
| | Input / Output | | |
| Return | E_OK | 1 | |
| | E_NOT_OK | 0 | |
| Description | Function responsible for stopping the work of the pwm specified.<br>If the TMR_Channel passed isn't available for PWM operation, the function should return an error and do nothing. | | |

## LCD Module

| | | |
|---|---|---|
| Function name | LCD_Init | |
| Arguments | Input | |

| | | |
|---|---|---|
| | Output | |
| | Input / Output | |
| Return | E_OK | 1 |
| | E_NOT_OK | 0 |
| Description | Function responsible for initializing LCD module as well as clearing the screen and initializing the cursor.<br><br>Pins for:<br>LCD_Data_Pins, LCD_EN_Pin, LCD_RS_Pin, and LCD_RW_Pin must be predefined before function call. | |

| | | | |
|---|---|---|---|
| Function name | LCD_Display | | |
| Arguments | Input | LCD_String_Disp | Uint8* |
| | | Pointer to the start of the string to display | |
| | | LCD_String_Length | Uint8 |
| | | Length of the LCD_String_Disp | |
| | Output | | |
| | Input / Output | | |
| Return | E_OK | 1 | |
| | E_NOT_OK | 0 | |
| Description | Function responsible for displaying a string on the screen.<br>It should display the text from the current cursor location.<br>It's not responsible for setting the cursor or clearing the screen | | |

| | | | |
|---|---|---|---|
| Function name | LCD_SetCursor | | |
| Arguments | Input | LCD_Cursor_row | Uint8 |
| | | Row to set the cursor | |
| | | LCD_Cursor_col | Uint8 |
| | | Column to set the cursor | |
| | Output | | |
| | Input / Output | | |

| | | |
|---|---|---|
| Return | E_OK | 1 |
| | E_NOT_OK | 0 |
| Description | Function responsible for setting the cursor location. | |

| | | |
|---|---|---|
| Function name | LCD_ClearDisplay | |
| Arguments | Input | |
| | Output | |
| | Input / Output | |
| Return | E_OK | 1 |
| | E_NOT_OK | 0 |
| Description | Function responsible for clearing the display and setting the cursor at the beginning. | |

## Motor Module

| | | | |
|---|---|---|---|
| Function name | MOTOR_Init | | |
| Arguments | Input | MOTOR_Config | structure |
| | | Pointer to structure address holding motor configuration parameters.<br>Structure **members** must include: | |
| | | MOTOR_ID | Identifier to reference the motor with functions start/stop |
| | | MOTOR_Dir1_Pin | DIO port pin for direction |
| | | MOTOR_Dir2_Pin | DIO port pin for direction |
| | | MOTOR_Speed_Pin | DIO port pin for speed |

| | | | |
|---|---|---|---|
| | | MOTOR_PWM_CH | Channel of PWM to assign to the motor |
| | Output | | |
| | Input / Output | | |
| Return | E_OK | 1 | |
| | E_NOT_OK | 0 | |
| Description | Function responsible for initializing one motor instance and specify its DIO pins and their required configuration.<br>For the input parameter MOTOR_Config a null pointer check must be performed. If a nullptr detected, it should return an error and do nothing | | |

| | | | |
|---|---|---|---|
| Function name | MOTOR_Start | | |
| Arguments | Input | MOTOR_ID | Uint8 |
| | | Motor identifier to select which motor to start | |
| | | MOTOR_Speed | Uint8 |
| | | Define the PWM signal where 0 means motor stopping and (2^n)-1 means max speed. And n is the number of bits of the PWM channel. | |
| | Output | | |
| | Input / Output | | |
| Return | E_OK | 1 | |
| | E_NOT_OK | 0 | |
| Description | Function responsible for starting the work of the specified motor according to the specified speed.<br>If the passed MOTOR_ID argument is not valid, the function should return an error and do nothing. | | |

| | | | |
|---|---|---|---|
| Function name | MOTOR_Stop | | |
| Arguments | Input | MOTOR_ID | Uint8 |
| | | Motor identifier to select which motor to start | |
| | Output | | |

| | | |
|---|---|---|
| | Input / Output | |
| Return | E_OK | 1 |
| | E_NOT_OK | 0 |
| Description | Function responsible for stopping the work of the specified motor.<br>If the passed MOTOR_ID argument is not valid, the function should return an error and do nothing. | |

## ROBOT CONTROL Module

| | | |
|---|---|---|
| Function name | ROBOT_Init | |
| Arguments | Input | |
| | Output | |
| | Input / Output | |
| Return | E_OK | 1 |
| | E_NOT_OK | 0 |
| Description | Function responsible for initializing Robot module.<br>Pins for:<br>Motors and LCD must be predefined before function call. | |

| | | |
|---|---|---|
| Function name | ROBOT_Start | |
| Arguments | Input | |
| | Output | |
| | Input / Output | |
| Return | E_OK | 1 |
| | E_NOT_OK | 0 |
| Description | Function responsible for starting the work of the robot and its associated modules (LCD , Motors , Timers , PWM). | |

| Function name | ROBOT_Stop | |
|---|---|---|
| Arguments | Input | |
| | Output | |
| | Input / Output | |
| Return | E_OK | 1 |
| | E_NOT_OK | 0 |
| Description | Function responsible for stopping the work of the robot and its associated modules (LCD , Motors , Timers , PWM). | |

| Function name | ROBOT_MoveUpdate | |
|---|---|---|
| Arguments | Input | |
| | Output | |
| | Input / Output | |
| Return | E_OK | 1 |
| | E_NOT_OK | 0 |
| Description | Function responsible for updating the work of the robot.<br>It's a periodic function and should be constantly called to control the robot state from Power to running to stop mode. | |