**Static Design**

**By: Hassan Mahmoud**

**Schematic (Block Diagram)**

A picture containing text

Description automatically generated

**ECU 1**

**Layered Architecture**

Graphical user interface, application

Description automatically generated

Figure : ECU1 Layered Architecture

**-> MCAL Drivers**

**ECU1 components and APIs**

• DIO Driver:

|  |  |
| --- | --- |
| Name | Pin\_Direction |
| Type | Enum |
| Range | INPUT, OUTPUT |
| Description | Pin direction whether input or output |

|  |  |
| --- | --- |
| Name | Pin\_Value |
| Type | Enum |
| Range | LOW, HIGH |
| Description | Pin Value whether low or high |

|  |  |
| --- | --- |
| Name | PORTS |
| Type | Enum |
| Range | PORTA ---> PORTF |
| Description | Choosing port Number |

|  |  |  |
| --- | --- | --- |
| Name | MDIO\_Init | |
| Arguments | Input | Void |
| Output | N/A |
| Return | OK | Void |
| NOK |
| Description | Initialize DIO peripheral | |
| Type | Synchronous / Non-Reentrant | |

|  |  |  |  |
| --- | --- | --- | --- |
| Name | MDIO\_SetPinDirection | | |
| Arguments | Input | PortNumber | PORTS |
| choose port number | |
| PinNumber | U8 |
| choose pin number | |
| Direction | Pin\_Direction |
| choose pin direction (Input/Output) | |
| Output | N/A | |
| Return | OK | Void | |
| NOK |
| Description | Define the direction of a pin (Input/Output) | | |
| Type | Synchronous / Non-Reentrant | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Name | MDIO\_SetPinValue | | |
| Arguments | Input | PortNumber | PORTS |
| choose port number | |
| PinNumber | U8 |
| choose pin number | |
| Value | Pin\_Value |
| choose pin Value (High/Low) | |
| Output | N/A | |
| Return | OK | Void | |
| NOK |
| Description | Set the logical state of the pin (High/Low) | | |
| Type | Synchronous / Non-Reentrant | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Name | MDIO\_TogglePin | | |
| Arguments | Input | PortNumber | PORTS |
| choose port number | |
| PinNumber | U8 |
| choose pin number | |
| Output | N/A | |
| Return | OK | Void | |
| NOK |
| Description | Toggle the logical state of a pin High to Low and vice versa | | |
| Type | Synchronous / Non-Reentrant | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Name | MDIO\_ReadPin | | |
| Arguments | Input | PortNumber | PORTS |
| choose port number | |
| PinNumber | U8 |
| choose pin number | |
| Output | N/A | |
| Return | OK | - | Pin\_Value |
| NOK | Void | |
| Description | Read the logical state of a pin High or Low | | |
| Type | Synchronous / Non-Reentrant | | |

• ADC Driver:

|  |  |
| --- | --- |
| Name | ADC\_ID |
| Type | Enum |
| Range | ADC0, ADC1 |
| Description | Choose ADC to work |

|  |  |
| --- | --- |
| Name | ADC\_Channel |
| Type | Enum |
| Range | PE3, PE2, PE1, PE0, PD3, PD2, PD1,  PD0, PE5, PE4, PB4, PB5 |
| Description | Choose ADC channel |

|  |  |  |
| --- | --- | --- |
| Name | MADC\_Init | |
| Arguments | Input | Void |
| Output | N/A |
| Return | OK | Void |
| NOK |
| Description | Initialize ADC peripheral | |
| Type | Synchronous / Non-Reentrant | |

|  |  |  |  |
| --- | --- | --- | --- |
| Name | MADC\_Convert | | |
| Arguments | Input | ADC\_n | ADC\_ID |
| Choose ADC peripheral | |
| Cahnnel\_n | ADC\_Channel |
| Choose channel to convert | |
| Output | N/A | |
| Return | OK | - | U32 |
| NOK | Void | |
| Description | Start the conversion of a defined channel and get its reading | | |
| Type | Synchronous / Non-Reentrant | | |

• CAN Driver:

|  |  |
| --- | --- |
| Name | BitRate\_Config\_Struct |
| Type | Struct |
| Elements | uint8 BRP  uint8 SJW  uint8 TSEG1  uint8 TSEG2 |
| Description | Configure bit rate by the user |

|  |  |
| --- | --- |
| Name | CAN\_Config\_Struct |
| Type | Struct |
| Elements | uint8 CANID  uint8 MODE  uint32 BitRate |
| Description | Choose the mode of CAN |

|  |  |
| --- | --- |
| Name | CAN\_MSG\_Object |
| Type | Struct |
| Elements | u32 MsgID  u8 MsgObjectNumber  u8 MsgIDExtension  u8 MsgLen  u32 MsgIDMask |
| Description | Message object structure |

|  |  |  |  |
| --- | --- | --- | --- |
| Name | CAN\_Init | | |
| Arguments | Input | \*ConfigPtr | CAN\_Config\_Struct |
| Input configurations for CAN Bus | |
| \*Bit\_Rate\_Ptr | BitRate\_Config\_Struct |
| Input configurations for Bit Rate | |
| Output | N/A | |
| Return | OK | Void | |
| NOK |
| Description | Responsible for initializing the CAN drive | | |
| Type | Synchronous / Non-Reentrant | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Name | CAN\_Transmit\_SetObject | | |
| Arguments | Input | CAN\_ID | U8 |
| Choose channel | |
| \*MsgObject | CAN\_MSG\_Object |
| Struct that holds the message | |
| Output | N/A | |
| Return | OK | Void | |
| NOK |
| Description | Responsible for configuring transmit message object | | |
| Type | Synchronous / Non-Reentrant | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | CAN\_Transmit\_Object | | | |
| Arguments | Input | CAN\_ID | U8 | |
| Choose channel | | |
| MsgObjNum | U8 | |
| Message object number from 1 to 32 | | |
| \*data\_pt | | U8 |
| Points to data | | |
| Output | N/A | | |
| Return | OK | Void | | |
| NOK |
| Description | Responsible for transmitting message object | | | |
| Type | Synchronous / Non-Reentrant | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Name | CAN\_Receive\_SetObject | | |
| Arguments | Input | CAN\_ID | U8 |
| Choose channel | |
| \*MsgObject | CAN\_MSG\_Object |
| Struct that holds the message | |
| Output | N/A | |
| Return | OK | Void | |
| NOK |
| Description | Responsible for configuring receive message object | | |
| Type | Synchronous / Non-Reentrant | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | CAN\_Receive\_Object | | | |
| Arguments | Input | CAN\_ID | U8 | |
| Choose channel | | |
| MsgObjNum | U8 | |
| Message object number from 1 to 32 | | |
| \*data\_pt | | U8 |
| Points to data | | |
| Output | N/A | | |
| Return | OK | Void | | |
| NOK |
| Description | Responsible for receiving message object | | | |
| Type | Synchronous / Non-Reentrant | | | |

• TIMER Driver:

|  |  |
| --- | --- |
| Name | GPT\_ChannelID |
| Type | Enum |
| Range | Channel\_0 ---> Channel\_11 |
| Description | Choose the GPT channel |

|  |  |
| --- | --- |
| Name | GPT\_Mode |
| Type | Enum |
| Range | GPT\_ONE\_SHOT, GPT\_MODE\_PERIODIC |
| Description | Choose the GPT channel |

|  |  |
| --- | --- |
| Name | GPT\_TIM\_TYPE |
| Type | Enum |
| Range | GPT\_PREDEF\_TIMER\_1US\_16BIT = 4  GPT\_PREDEF\_TIMER\_1US\_24BIT = 4  GPT\_PREDEF\_TIMER\_1US\_32BIT = 0 |
| Description | Choose the GPT channel |

|  |  |
| --- | --- |
| Name | GPT\_ConfigType |
| Type | Struct |
| Elements | Gpt\_ChannelID Channel\_ID  Gpt\_Mode Channel\_Mode  GPT\_TIM\_TYPE Timer\_Type |
| Description | Configuration of Timer |

|  |  |  |  |
| --- | --- | --- | --- |
| Name | MGPT\_Init | | |
| Arguments | Input | ConfigPtr | GPT\_ConfigType |
| Struct that contains configurations | |
| Output | N/A | |
| Return | OK | Void | |
| NOK |
| Description | Initialize GPT peripheral | | |
| Type | Synchronous / Non-Reentrant | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Name | MGPT\_StartTimer | | |
| Arguments | Input | \*ConfigPtr | GPT\_ConfigType |
| Struct contains needed information | |
| Time | U32 |
| Time to count | |
| Output | N/A | |
| Return | OK | Void | |
| NOK |
| Description | Start the timer | | |
| Type | Synchronous / Non-Reentrant | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Name | MGPT\_StopTimer | | |
| Arguments | Input | \*ConfigPtr | GPT\_ConfigType |
| Struct contains needed information | |
| Output | N/A | |
| Return | OK | Void | |
| NOK |
| Description | Stop the timer | | |
| Type | Synchronous / Non-Reentrant | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Name | MGPT\_EnableNotif | | |
| Arguments | Input | \*ConfigPtr | GPT\_ConfigType |
| Struct contains needed information | |
| Output | N/A | |
| Return | OK | Void | |
| NOK |
| Description | Enable notification when the timer finish or start | | |
| Type | Synchronous / Non-Reentrant | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Name | MGPT\_DisableNotif | | |
| Arguments | Input | \*ConfigPtr | GPT\_ConfigType |
| Struct contains needed information | |
| Output | N/A | |
| Return | OK | Void | |
| NOK |
| Description | Disable notification when the timer finish or start | | |
| Type | Synchronous / Non-Reentrant | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Name | MGPT\_HasFinished | | |
| Arguments | Input | \*ConfigPtr | GPT\_ConfigType |
| Struct contains needed information | |
| Output | N/A | |
| Return | OK | Void | |
| NOK |
| Description | Check if the flag is up or not | | |
| Type | Synchronous / Non-Reentrant | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Name | MGPT\_GetTimeElapsed | | |
| Arguments | Input | \*ConfigPtr | GPT\_ConfigType |
| Struct contains needed information | |
| Output | N/A | |
| Return | OK | - | U32 |
| NOK | Void | |
| Description | Get the information of time elapsed from GPT | | |
| Type | Synchronous / Non-Reentrant | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Name | MGPT\_GetTimeRemaining | | |
| Arguments | Input | \*ConfigPtr | GPT\_ConfigType |
| Struct contains needed information | |
| Output | N/A | |
| Return | OK | - | U32 |
| NOK | Void | |
| Description | Get the information of time remaining from GPT | | |
| Type | Synchronous / Non-Reentrant | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Name | MGPT\_ClearFlag | | |
| Arguments | Input | \*ConfigPtr | GPT\_ConfigType |
| Struct contains needed information | |
| Output | N/A | |
| Return | OK | Void | |
| NOK |
| Description | Clear the flag after the finished time | | |
| Type | Synchronous / Non-Reentrant | | |

**-> HAL Drivers**

• Door Driver:

|  |  |  |  |
| --- | --- | --- | --- |
| Name | HDoor\_Init | | |
| Arguments | Input | Door\_ID | U32 |
| Holding door ID | |
| Output | N/A | |
| Return | OK | Void | |
| NOK |
| Description | Initialization the door module | | |
| Type | Synchronous / Non-Reentrant | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Name | HDoor\_GetDoorState | | |
| Arguments | Input | Door\_ID | U32 |
| Holding Door ID | |
| Output | Door is open or door is close | |
| Return | OK | 0 | |
| NOK | 1 | |
| Description | Get the door state | | |
| Type | Synchronous / Non-Reentrant | | |

• Light Driver:

|  |  |  |  |
| --- | --- | --- | --- |
| Name | HLight\_Init | | |
| Arguments | Input | Switch\_ID | U32 |
| Holding Switch ID | |
| Output | N/A | |
| Return | OK | Void | |
| NOK |
| Description | Initialization the Light module | | |
| Type | Synchronous / Non-Reentrant | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Name | HLight\_GetSwitchState | | |
| Arguments | Input | Door\_ID | U32 |
| Holding Light ID | |
| Output | Switch is pressed or switch is not  pressed | |
| Return | OK | 0 | |
| NOK | 1 | |
| Description | Get the switch state | | |
| Type | Synchronous / Non-Reentrant | | |

• Speed Sensor Driver:

|  |  |  |  |
| --- | --- | --- | --- |
| Name | HSpeedSen \_Init | | |
| Arguments | Input | Sensor\_ID | U32 |
| Holding Sensor ID | |
| Output | N/A | |
| Return | OK | Void | |
| NOK |
| Description | Initialization the Sensor module | | |
| Type | Synchronous / Non-Reentrant | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Name | HSpeedSen\_CalcSpeed | | |
| Arguments | Input | Sensor\_ID | U32 |
| Holding Sensor ID | |
| Output | Speed | |
| Return | OK | Void | |
| NOK |
| Description | Calculate speed | | |
| Type | Synchronous / Non-Reentrant | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Name | HSpeedSen\_GetSpeed | | |
| Arguments | Input | Door\_ID | U32 |
| Holding Door ID | |
| Output | Speed | |
| Return | OK | 0 | |
| NOK | 1 | |
| Description | Get the Speed | | |
| Type | Synchronous / Non-Reentrant | | |

**-> Service Layer Drivers**

• Communication Module:

|  |  |  |  |
| --- | --- | --- | --- |
| Name | SendDoorState | | |
| Arguments | Input | Data | U32 |
| Door data | |
| Output | N/A | |
| Return | OK | Void | |
| NOK |
| Description | Send data of door every 10ms | | |
| Type | Synchronous / Non-Reentrant | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Name | SendLightSWState | | |
| Arguments | Input | Data | U32 |
| Light Switch data | |
| Output | N/A | |
| Return | OK | Void | |
| NOK |
| Description | Send data of light switch every 20ms | | |
| Type | Synchronous / Non-Reentrant | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Name | SendSpeed | | |
| Arguments | Input | Data | U32 |
| Holding speed value | |
| Output | N/A | |
| Return | OK | Void | |
| NOK |
| Description | Send data of speed every 5ms | | |
| Type | Synchronous / Non-Reentrant | | |

**ECU 2**

**Layered Architecture**

Graphical user interface

Description automatically generated

Figure ; ECU2 Layered Architecture

**-> MCAL Drivers**

**ECU1 components and APIs**

• DIO Driver:

|  |  |
| --- | --- |
| Name | Pin\_Direction |
| Type | Enum |
| Range | INPUT, OUTPUT |
| Description | Pin direction whether input or output |

|  |  |
| --- | --- |
| Name | Pin\_Value |
| Type | Enum |
| Range | LOW, HIGH |
| Description | Pin Value whether low or high |

|  |  |
| --- | --- |
| Name | PORTS |
| Type | Enum |
| Range | PORTA ---> PORTF |
| Description | Choosing port Number |

|  |  |  |
| --- | --- | --- |
| Name | MDIO\_Init | |
| Arguments | Input | Void |
| Output | N/A |
| Return | OK | Void |
| NOK |
| Description | Initialize DIO peripheral | |
| Type | Synchronous / Non-Reentrant | |

|  |  |  |  |
| --- | --- | --- | --- |
| Name | MDIO\_SetPinDirection | | |
| Arguments | Input | PortNumber | PORTS |
| choose port number | |
| PinNumber | U8 |
| choose pin number | |
| Direction | Pin\_Direction |
| choose pin direction (Input/Output) | |
| Output | N/A | |
| Return | OK | Void | |
| NOK |
| Description | Define the direction of a pin (Input/Output) | | |
| Type | Synchronous / Non-Reentrant | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Name | MDIO\_SetPinValue | | |
| Arguments | Input | PortNumber | PORTS |
| choose port number | |
| PinNumber | U8 |
| choose pin number | |
| Value | Pin\_Value |
| choose pin Value (High/Low) | |
| Output | N/A | |
| Return | OK | Void | |
| NOK |
| Description | Set the logical state of the pin (High/Low) | | |
| Type | Synchronous / Non-Reentrant | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Name | MDIO\_TogglePin | | |
| Arguments | Input | PortNumber | PORTS |
| choose port number | |
| PinNumber | U8 |
| choose pin number | |
| Output | N/A | |
| Return | OK | Void | |
| NOK |
| Description | Toggle the logical state of a pin High to Low and vice versa | | |
| Type | Synchronous / Non-Reentrant | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Name | MDIO\_ReadPin | | |
| Arguments | Input | PortNumber | PORTS |
| choose port number | |
| PinNumber | U8 |
| choose pin number | |
| Output | N/A | |
| Return | OK | - | Pin\_Value |
| NOK | Void | |
| Description | Read the logical state of a pin High or Low | | |
| Type | Synchronous / Non-Reentrant | | |

• ADC Driver:

|  |  |
| --- | --- |
| Name | ADC\_ID |
| Type | Enum |
| Range | ADC0, ADC1 |
| Description | Choose ADC to work |

|  |  |
| --- | --- |
| Name | ADC\_Channel |
| Type | Enum |
| Range | PE3, PE2, PE1, PE0, PD3, PD2, PD1,  PD0, PE5, PE4, PB4, PB5 |
| Description | Choose ADC channel |

|  |  |  |
| --- | --- | --- |
| Name | MADC\_Init | |
| Arguments | Input | Void |
| Output | N/A |
| Return | OK | Void |
| NOK |
| Description | Initialize ADC peripheral | |
| Type | Synchronous / Non-Reentrant | |

|  |  |  |  |
| --- | --- | --- | --- |
| Name | MADC\_Convert | | |
| Arguments | Input | ADC\_n | ADC\_ID |
| Choose ADC peripheral | |
| Cahnnel\_n | ADC\_Channel |
| Choose channel to convert | |
| Output | N/A | |
| Return | OK | - | U32 |
| NOK | Void | |
| Description | Start the conversion of a defined channel and get its reading | | |
| Type | Synchronous / Non-Reentrant | | |

• CAN Driver:

|  |  |
| --- | --- |
| Name | BitRate\_Config\_Struct |
| Type | Struct |
| Elements | uint8 BRP  uint8 SJW  uint8 TSEG1  uint8 TSEG2 |
| Description | Configure bit rate by the user |

|  |  |
| --- | --- |
| Name | CAN\_Config\_Struct |
| Type | Struct |
| Elements | uint8 CANID  uint8 MODE  uint32 BitRate |
| Description | Choose the mode of CAN |

|  |  |
| --- | --- |
| Name | CAN\_MSG\_Object |
| Type | Struct |
| Elements | u32 MsgID  u8 MsgObjectNumber  u8 MsgIDExtension  u8 MsgLen  u32 MsgIDMask |
| Description | Message object structure |

|  |  |  |  |
| --- | --- | --- | --- |
| Name | CAN\_Init | | |
| Arguments | Input | \*ConfigPtr | CAN\_Config\_Struct |
| Input configurations for CAN Bus | |
| \*Bit\_Rate\_Ptr | BitRate\_Config\_Struct |
| Input configurations for Bit Rate | |
| Output | N/A | |
| Return | OK | Void | |
| NOK |
| Description | Responsible for initializing the CAN drive | | |
| Type | Synchronous / Non-Reentrant | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Name | CAN\_Transmit\_SetObject | | |
| Arguments | Input | CAN\_ID | U8 |
| Choose channel | |
| \*MsgObject | CAN\_MSG\_Object |
| Struct that holds the message | |
| Output | N/A | |
| Return | OK | Void | |
| NOK |
| Description | Responsible for configuring transmit message object | | |
| Type | Synchronous / Non-Reentrant | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | CAN\_Transmit\_Object | | | |
| Arguments | Input | CAN\_ID | U8 | |
| Choose channel | | |
| MsgObjNum | U8 | |
| Message object number from 1 to 32 | | |
| \*data\_pt | | U8 |
| Points to data | | |
| Output | N/A | | |
| Return | OK | Void | | |
| NOK |
| Description | Responsible for transmitting message object | | | |
| Type | Synchronous / Non-Reentrant | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Name | CAN\_Receive\_SetObject | | |
| Arguments | Input | CAN\_ID | U8 |
| Choose channel | |
| \*MsgObject | CAN\_MSG\_Object |
| Struct that holds the message | |
| Output | N/A | |
| Return | OK | Void | |
| NOK |
| Description | Responsible for configuring receive message object | | |
| Type | Synchronous / Non-Reentrant | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | CAN\_Receive\_Object | | | |
| Arguments | Input | CAN\_ID | U8 | |
| Choose channel | | |
| MsgObjNum | U8 | |
| Message object number from 1 to 32 | | |
| \*data\_pt | | U8 |
| Points to data | | |
| Output | N/A | | |
| Return | OK | Void | | |
| NOK |
| Description | Responsible for receiving message object | | | |
| Type | Synchronous / Non-Reentrant | | | |

• TIMER Driver:

|  |  |
| --- | --- |
| Name | GPT\_ChannelID |
| Type | Enum |
| Range | Channel\_0 ---> Channel\_11 |
| Description | Choose the GPT channel |

|  |  |
| --- | --- |
| Name | GPT\_Mode |
| Type | Enum |
| Range | GPT\_ONE\_SHOT, GPT\_MODE\_PERIODIC |
| Description | Choose the GPT channel |

|  |  |
| --- | --- |
| Name | GPT\_TIM\_TYPE |
| Type | Enum |
| Range | GPT\_PREDEF\_TIMER\_1US\_16BIT = 4  GPT\_PREDEF\_TIMER\_1US\_24BIT = 4  GPT\_PREDEF\_TIMER\_1US\_32BIT = 0 |
| Description | Choose the GPT channel |

|  |  |
| --- | --- |
| Name | GPT\_ConfigType |
| Type | Struct |
| Elements | Gpt\_ChannelID Channel\_ID  Gpt\_Mode Channel\_Mode  GPT\_TIM\_TYPE Timer\_Type |
| Description | Configuration of Timer |

|  |  |  |  |
| --- | --- | --- | --- |
| Name | MGPT\_Init | | |
| Arguments | Input | ConfigPtr | GPT\_ConfigType |
| Struct that contains configurations | |
| Output | N/A | |
| Return | OK | Void | |
| NOK |
| Description | Initialize GPT peripheral | | |
| Type | Synchronous / Non-Reentrant | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Name | MGPT\_StartTimer | | |
| Arguments | Input | \*ConfigPtr | GPT\_ConfigType |
| Struct contains needed information | |
| Time | U32 |
| Time to count | |
| Output | N/A | |
| Return | OK | Void | |
| NOK |
| Description | Start the timer | | |
| Type | Synchronous / Non-Reentrant | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Name | MGPT\_StopTimer | | |
| Arguments | Input | \*ConfigPtr | GPT\_ConfigType |
| Struct contains needed information | |
| Output | N/A | |
| Return | OK | Void | |
| NOK |
| Description | Stop the timer | | |
| Type | Synchronous / Non-Reentrant | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Name | MGPT\_EnableNotif | | |
| Arguments | Input | \*ConfigPtr | GPT\_ConfigType |
| Struct contains needed information | |
| Output | N/A | |
| Return | OK | Void | |
| NOK |
| Description | Enable notification when the timer finish or start | | |
| Type | Synchronous / Non-Reentrant | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Name | MGPT\_DisableNotif | | |
| Arguments | Input | \*ConfigPtr | GPT\_ConfigType |
| Struct contains needed information | |
| Output | N/A | |
| Return | OK | Void | |
| NOK |
| Description | Disable notification when the timer finish or start | | |
| Type | Synchronous / Non-Reentrant | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Name | MGPT\_HasFinished | | |
| Arguments | Input | \*ConfigPtr | GPT\_ConfigType |
| Struct contains needed information | |
| Output | N/A | |
| Return | OK | Void | |
| NOK |
| Description | Check if the flag is up or not | | |
| Type | Synchronous / Non-Reentrant | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Name | MGPT\_GetTimeElapsed | | |
| Arguments | Input | \*ConfigPtr | GPT\_ConfigType |
| Struct contains needed information | |
| Output | N/A | |
| Return | OK | - | U32 |
| NOK | Void | |
| Description | Get the information of time elapsed from GPT | | |
| Type | Synchronous / Non-Reentrant | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Name | MGPT\_GetTimeRemaining | | |
| Arguments | Input | \*ConfigPtr | GPT\_ConfigType |
| Struct contains needed information | |
| Output | N/A | |
| Return | OK | - | U32 |
| NOK | Void | |
| Description | Get the information of time remaining from GPT | | |
| Type | Synchronous / Non-Reentrant | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Name | MGPT\_ClearFlag | | |
| Arguments | Input | \*ConfigPtr | GPT\_ConfigType |
| Struct contains needed information | |
| Output | N/A | |
| Return | OK | Void | |
| NOK |
| Description | Clear the flag after the finished time | | |
| Type | Synchronous / Non-Reentrant | | |

**-> HAL Drivers**

• Buzzer Driver:

|  |  |  |  |
| --- | --- | --- | --- |
| Name | HBuzzer\_Init | | |
| Arguments | Input | Buzzer \_ID | U32 |
| Holding Buzzer ID | |
| Output | N/A | |
| Return | OK | Void | |
| NOK |
| Description | Initialization the buzzer module | | |
| Type | Synchronous / Non-Reentrant | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Name | HBuzzer \_ON | | |
| Arguments | Input | Buzzer\_ID | U32 |
| Holding Buzzer ID | |
| Output | Turn ON Buzzer | |
| Return | OK | Void | |
| NOK |
| Description | Turn OFF Buzzer | | |
| Type | Synchronous / Non-Reentrant | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Name | HBuzzer \_OFF | | |
| Arguments | Input | Buzzer \_ID | U32 |
| Holding Buzzer ID | |
| Output | Turn OFF Buzzer | |
| Return | OK | Void | |
| NOK |
| Description | Turn OFF Buzzer | | |
| Type | Synchronous / Non-Reentrant | | |

• Lights Driver:

|  |  |  |  |
| --- | --- | --- | --- |
| Name | HLight\_Init | | |
| Arguments | Input | Light\_ID | U32 |
| Holding Light ID | |
| Output | N/A | |
| Return | OK | Void | |
| NOK |
| Description | Initialization the Light module | | |
| Type | Synchronous / Non-Reentrant | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Name | HLight \_ON | | |
| Arguments | Input | Light \_ID | U32 |
| Holding Light ID | |
| Output | Turn ON Light | |
| Return | OK | Void | |
| NOK |
| Description | Turn ON Light | | |
| Type | Synchronous / Non-Reentrant | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Name | HLight \_OFF | | |
| Arguments | Input | Light \_ID | U32 |
| Holding Light ID | |
| Output | Turn OFF Light | |
| Return | OK | Void | |
| NOK |
| Description | Turn OFF Light | | |
| Type | Synchronous / Non-Reentrant | | |

**-> Service Layer Drivers**

• Communication Module:

|  |  |  |  |
| --- | --- | --- | --- |
| Name | ReceiveDoorState | | |
| Arguments | Input | \*Data | Ptr U32 |
| Hold Door data | |
| Output | N/A | |
| Return | OK | Void | |
| NOK |
| Description | Receive data of door every 10ms | | |
| Type | Synchronous / Non-Reentrant | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Name | ReceiveLightSWState | | |
| Arguments | Input | \*Data | Ptr U32 |
| Light Switch data | |
| Output | N/A | |
| Return | OK | Void | |
| NOK |
| Description | Receive data of light switch every 20ms | | |
| Type | Synchronous / Non-Reentrant | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Name | ReceiveSpeed | | |
| Arguments | Input | \*Data | Ptr U32 |
| Holding speed value | |
| Output | N/A | |
| Return | OK | Void | |
| NOK |
| Description | Receive data of speed every 5ms | | |
| Type | Synchronous / Non-Reentrant | | |