ElectricWaterHeater

0.0.1

Generated by Doxygen 1.8.17

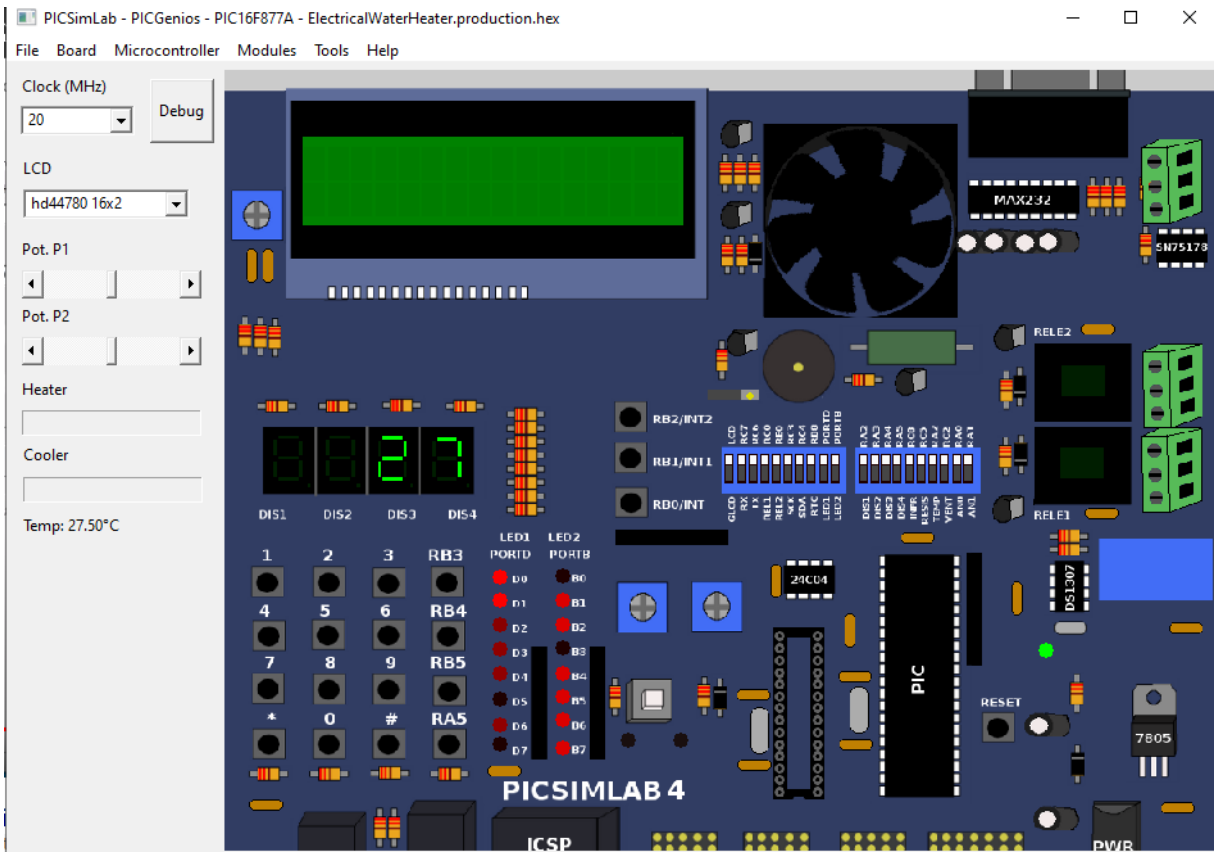
Chapter 1

Electrical Water Heater

this project is implementation of the Swift Act requirement of Electrical Water Heater `link`.

1.1 Demo

1.1.1 click image to check demo video



1.2 Dependencies:

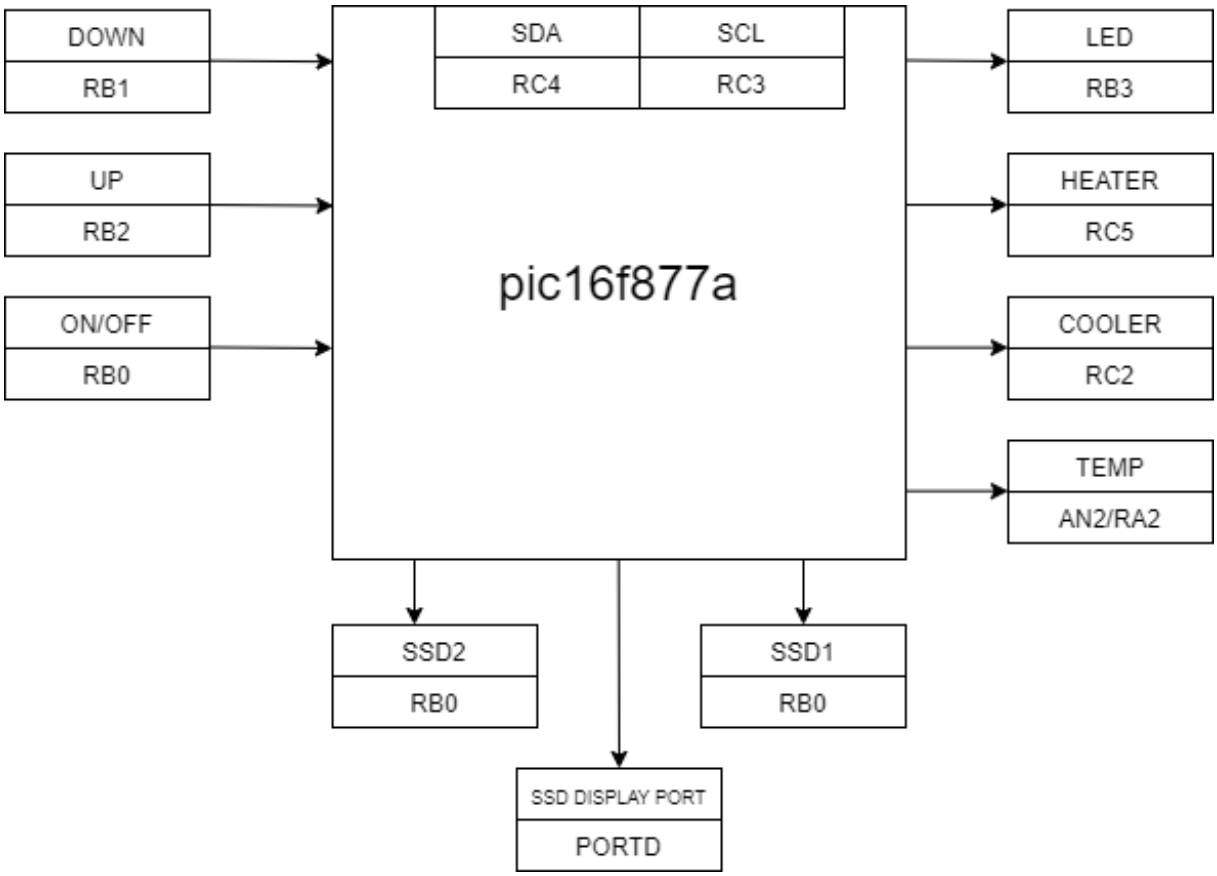
1. PICSimlab simulation program link.
2. project developed using MPLAB X IDE from microchip.
3. xc8 compiler from microchip.

1.2.0.0.1 project documented useing doxygen documentaion in docmentation/html/index.html

1.3 Main Components:

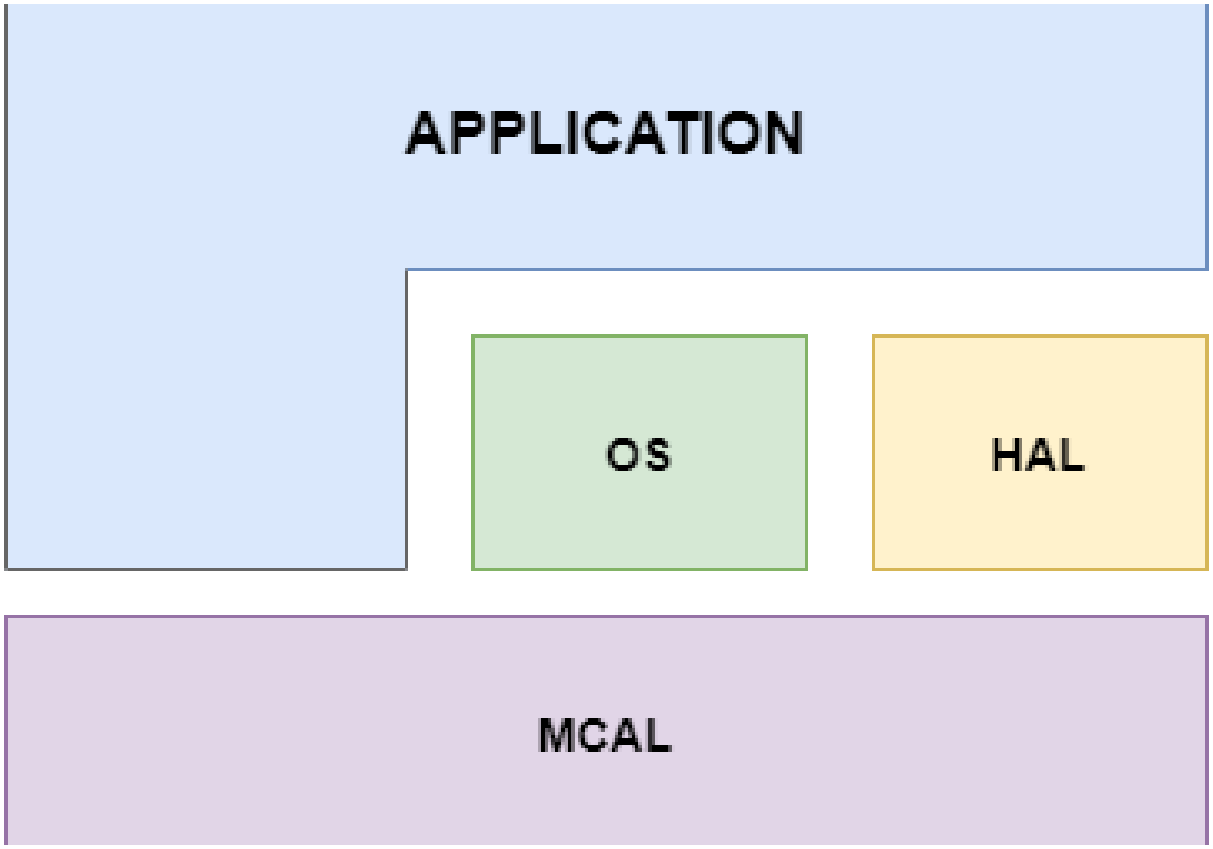
1.3.1 Hardware:

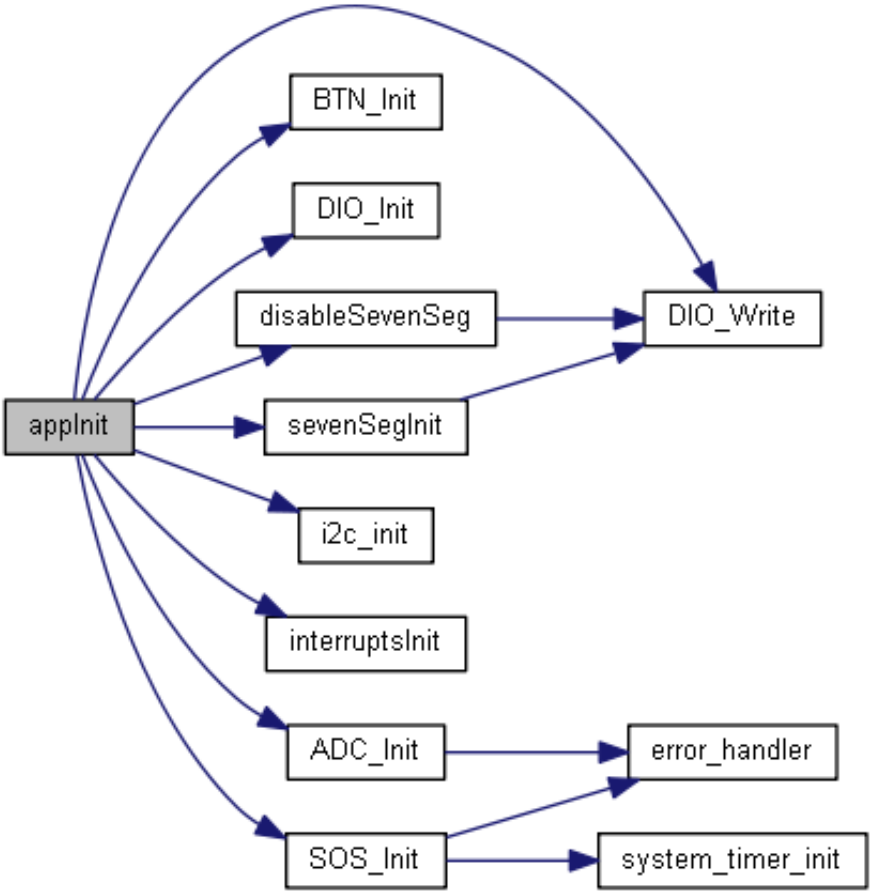
The project emulate Electrical Water Heater on PICSimlab simulation program using PicGenios kit with PIC16↩ F877A.



1.4 Software Components:

1.4.0.1 Project Static Architecutre:



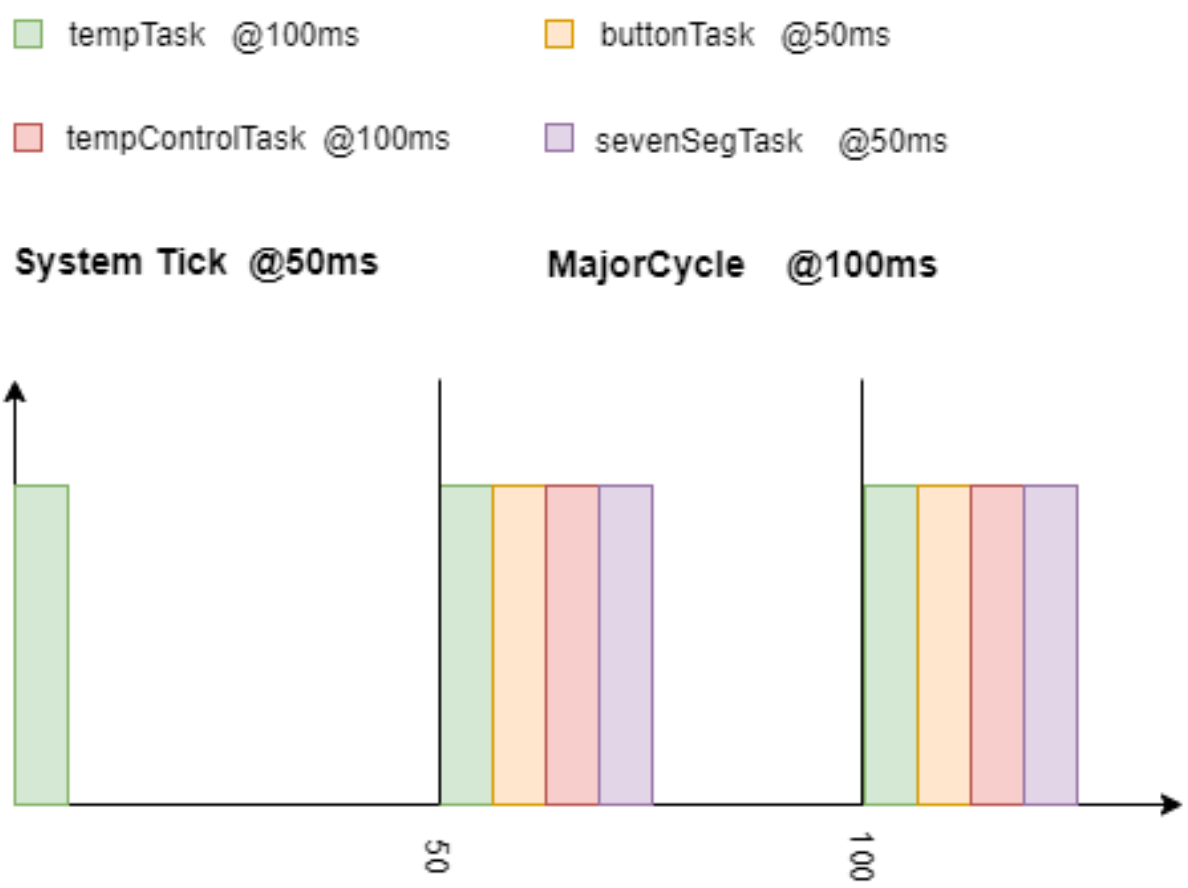


Note: Application calls MCAL directly to inialize the state of LED, Heater and Cooler to off state as there is no HAL driver for thoes peripheral_i saw that those peripheral logic is too simple to make a sperated driver for them_.

1.4.0.2 Project Detailed Design:

- Interrupt
  - interruptsInit() : **isr.c** , **isr.h**
  - __interrupt() : **isr.c**
- ADC
  - ADC_Init() : **adc.c** , **adc.h**
  - ADC_trigger() : **adc.c** , **adc.h**
  - getADC_value**() :** **adc.c** , **adc.h**
- systemTimer
  - start_system_timer() : **timer.c** , **timer.h**
  - system_timer_init() : **timer.c** , **timer.h**
- EXT_EEPROM
  - e2pext_r() : **eeprom_ext.c** , **eeprom_ext.h**
  - e2pext_w() : **eeprom_ext.c** , **eeprom_ext.h**
- ERROR_handle_Module
  - error_handler() : **ErrorHandler.c** , **SystemErrors.h**
- APP
  - Tasks
    - tempControlTask() : **main.c**
    - tempTask() : **main.c**
    - sevenSegTask() : **main.c**
    - buttonTask() : **main.c**
  - Functions
    - main() : **main.c**
    - appInit() : **main.c**
    - OS_startFunction() : **main.c**
    - checkONBtnStatus() : **main.c**
- DIO
  - DIO_Init() : **DIO.c** , **DIO.h**
  - DIO_Read() : **DIO.c** , **DIO.h**
  - DIO_Toggle() : **DIO.c** , **DIO.h**
  - DIO_Write() : **DIO.c** , **DIO.h**
  - disableSevenSeg() : **7seg.c** , **7seg.h**
- I2C
  - i2c_acktst() : **i2c.h**
  - i2c_init() : **i2c.c** , **i2c.h**
  - i2c_rb() : **i2c.c** , **i2c.h**
  - i2c_start() : **i2c.c** , **i2c.h**
  - i2c_stop() : **i2c.c** , **i2c.h**
  - i2c_wb() : **i2c.c** , **i2c.h**
- SSD
  - sevenSegInit() : **7seg.c** , **7seg.h**
  - sevenSegSendChar() : **7seg.c** , **7seg.h**
  - disableSevenSeg() : **7seg.c** , **7seg.h**
- SOS
  - SOS_createTask() : **SOS.c** , **SOS.h**
  - SOS_Deinit() : **SOS.c** , **SOS.h**
  - SOS_deletTask() : **SOS.c** , **SOS.h**
  - SOS_Init() : **SOS.c** , **SOS.h**
  - SOS_run() : **SOS.c** , **SOS.h**
  - SOS_StartProc() : **SOS.c** , **SOS.h**

1.4.0.3 Task-TimeLine:



- Project uses a non-premature operating system with 50ms tick time.
- task periodic time specified in the below graph.
- project has five different tasks {tempTask,ButtonTask,tempControlTask,sevenSegTask} their priority as their listing order with highest priority task [tempTask] and lowest priority is [sevenSegTask].
- the system repeats it's operation every 100ms *Major Cycle*.
- os prehook runs tempTask once to prepare current temp for seven segment display before it's calling.
- operation time of each task is neglectable *no blocking for a considerable amount of time*.

1.4.0.4 Operating System:

This project uses a non premative OS with a periodic task, the os prority feature is turned off as a result task take it's priority from the order of its creation relative to other tasks. The project has five different tasks {tempTask,Button↩ Task,tempControlTask,sevenSegTask} their priority as thier listing order with highest priority task [tempTask] and lowest priority is [sevenSegTask].

1.4.0.5 System Tasks:

1.4.0.5.1 tempTask:

- check if user set a temperature
- if a temp is set then
  1. get the average temp of the last ten readings.
  2. take an action based on the state of the heater and cooler state and the set temp value.
- if the cooler element is on turn on the led if the heater element is on blink led every 1 sec.

1.4.0.5.2 ButtonTask:

- call button manager.
- update the state of each button{up-down-on/off}.
- if the on-off button is pressed switch the system {on/off} respectively and exit task.
- if in normal mode and up or down button pressed to get the last set value from EXT_EEPROM and change mode to TEMP_SET_MOD.
- if in temp set mode and up or down button pressed to increase or decrease the temp to set by 5 depending on which button pressed respectively.
- if neither the up nor down button pressed to save the set temp to EXT_EEPROM and change mode to normal mode.

1.4.0.5.3 tempControlTask:

- Get the latest adc converted value form the ADC chanal connected to temp sensor.
- calculate the current temp in celsius.
- save current temp in *last 10 temp reading* array.

1.4.0.5.4 sevenSegTask:

- if in normal mode display the current temp value.
- if in temp set mode flash led every 1sec and display last set temp
  - Interact with the user if up or down button pressed to change the temp to set by 5 degrees above or below the current temp based on the button pressed.
  - Max temp 75 and Min temp to set is 35.

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

Data Structure Documentation

4.1 BtnConfigType Struct Reference

`#include <btn.h>`

Data Fields

- uint8_t u8_DioGroupId
- ActiveStateType e_BtnActiveState

4.1.1 Detailed Description

Definition at line 8 of file btn.h.

4.1.2 Field Documentation

4.1.2.1 e_BtnActiveState

`ActiveStateType BtnConfigType::e_BtnActiveState`

Definition at line 11 of file btn.h.

4.1.2.2 u8_DioGroupId

`uint8_t BtnConfigType::u8_DioGroupId`

Definition at line 10 of file btn.h.

The documentation for this struct was generated from the following file:

- F:/Carier/Embedded/PIC/ElectricalWaterHeater/HAL/BTN/btn.h

4.2 DioConfigParam_t Struct Reference

`#include <DIO.h>`

Data Fields

- uint8_t Mask
- uint8_t Port
- uint8_t Direction
- uint8_t UsePullUp

4.2.1 Detailed Description

Definition at line 29 of file DIO.h.

4.2.2 Field Documentation

4.2.2.1 Direction

`uint8_t DioConfigParam_t::Direction`

Definition at line 33 of file DIO.h.

4.2.2.2 Mask

`uint8_t DioConfigParam_t::Mask`

Definition at line 31 of file DIO.h.

4.2.2.3 Port

`uint8_t DioConfigParam_t::Port`

Definition at line 32 of file DIO.h.

4.2.2.4 UsePullUp

```
uint8_t DioConfigParam_t::UsePullUp
```

Definition at line 34 of file DIO.h.

The documentation for this struct was generated from the following file:

- F:/Carier/Embedded/PIC/ElectricalWaterHeater/MCAL/DIO/DIO.h

4.3 gstr_ADC_ConfigParam_t Struct Reference

```
#include <adc.h>
```

Data Fields

- uint8_t ADC_CNTRL_0
- uint8_t ADC_CNTRL_1

4.3.1 Detailed Description

Definition at line 31 of file adc.h.

4.3.2 Field Documentation

4.3.2.1 ADC_CNTRL_0

```
uint8_t gstr_ADC_ConfigParam_t::ADC_CNTRL_0
```

Definition at line 33 of file adc.h.

4.3.2.2 ADC_CNTRL_1

```
uint8_t gstr_ADC_ConfigParam_t::ADC_CNTRL_1
```

Definition at line 34 of file adc.h.

The documentation for this struct was generated from the following file:

- F:/Carier/Embedded/PIC/ElectricalWaterHeater/MCAL/ADC/adc.h

4.4 sevenSegConfigStruct Struct Reference

`#include <7seg.h>`

Data Fields

- uint8_t u8_selectorPinGroupId
- uint8_t u8_sevenSegGroupId

4.4.1 Detailed Description

Definition at line 11 of file 7seg.h.

4.4.2 Field Documentation

4.4.2.1 u8_selectorPinGroupId

`uint8_t sevenSegConfigStruct::u8_selectorPinGroupId`

Definition at line 13 of file 7seg.h.

4.4.2.2 u8_sevenSegGroupId

`uint8_t sevenSegConfigStruct::u8_sevenSegGroupId`

Definition at line 14 of file 7seg.h.

The documentation for this struct was generated from the following file:

- F:/Carier/Embedded/PIC/ElectricalWaterHeater/HAL/7SEG/7seg.h

4.5 SOS_cfg_str Struct Reference

`#include <SOS.h>`

Data Fields

- uint8_t u8_timer_ch
- uint8_t u8_tick_reslution

4.5.1 Detailed Description

Definition at line 28 of file SOS.h.

4.5.2 Field Documentation

4.5.2.1 u8_tick_reslution

```
uint8_t SOS_cfg_str::u8_tick_reslution
```

Definition at line 31 of file SOS.h.

4.5.2.2 u8_timer_ch

```
uint8_t SOS_cfg_str::u8_timer_ch
```

Definition at line 30 of file SOS.h.

The documentation for this struct was generated from the following file:

- F:/Carier/Embedded/PIC/ElectricalWaterHeater/ServiceLayer/SOS/SOS.h

4.6 SOS_obj_str Struct Reference

Data Fields

- uint8_t Id
- void(∗ callB_fun)(void)
- uint16_t fire_tick
- uint16_t current_ticks
- uint8_t priority
- uint8_t type

4.6.1 Detailed Description

Definition at line 13 of file SOS.c.

4.6.2 Field Documentation

4.6.2.1 callB_fun

```
void(* SOS_obj_str::callB_fun) (void)
```

Definition at line 17 of file SOS.c.

4.6.2.2 current_ticks

```
uint16_t SOS_obj_str::current_ticks
```

Definition at line 19 of file SOS.c.

4.6.2.3 fire_tick

```
uint16_t SOS_obj_str::fire_tick
```

Definition at line 18 of file SOS.c.

4.6.2.4 Id

```
uint8_t SOS_obj_str::Id
```

Definition at line 16 of file SOS.c.

4.6.2.5 priority

```
uint8_t SOS_obj_str::priority
```

Definition at line 20 of file SOS.c.

4.6.2.6 type

```
uint8_t SOS_obj_str::type
```

Definition at line 21 of file SOS.c.

The documentation for this struct was generated from the following file:

- F:/Carier/Embedded/PIC/ElectricalWaterHeater/ServiceLayer/SOS/SOS.c

Chapter 5

File Documentation

5.1 F:/Carier/Embedded/PIC/ElectricalWaterHeater/Application/main.c File Reference

Enumerations

- enum operationMode_t { TEMP_SET_MODE, NORM_MODE }

Functions

- void tempControlTask ()
- void buttonTask ()
- void sevenSegTask ()
- void tempTask ()
- void appInit ()
- void checkONBtnStatus ()
- void OS_startFunction ()
- void main (void)

5.1.1 Enumeration Type Documentation

5.1.1.1 operationMode_t

```
enum operationMode_t
```

Enumerator

TEMP_SET_MODE	
NORM_MODE	

Definition at line 34 of file main.c.

5.1.2 Function Documentation

5.1.2.1 appInit()

```
void appInit ( )
```

init temp control element to zero

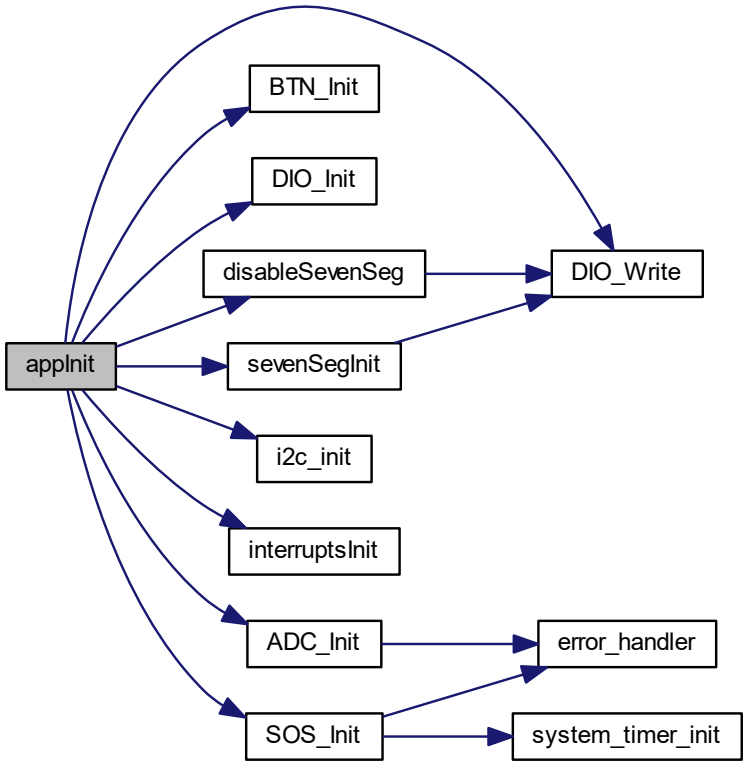
call MCAL functions from application as it is necessary to initialize the heater/cooler and led and there is no module implemented for them case their functionality are very simple and don't need to be structured necessary overhead↩
_

this block of code used to set the temp to 60 in the first run this can be simplified by checking for 0xff pattern but this is work for now

set initial values for global variables

Definition at line 307 of file main.c.

Here is the call graph for this function:



Here is the caller graph for this function:

5.1.2.2 buttonTask()

```
void buttonTask ( )
```

if on/off button pressed rise disable system flag we check on this flag on the main loop and disable the system if the flag is up

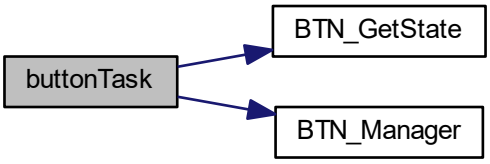
functionality of the button depend on the current system mode {NORM_MODE,TEMP_SET_MODE} in normal mode: if the up or down buttons pressed get temp from the external e2Prom change mode to TEMP_SET_M↩ IDE in TEMP_SET_MODE: if up or down buttons pressed change temp by 5 with max 75 and min 35 degree

if time out counter reached time out counts *button task called every 50ms so we need 50000/50 = 100 enterance befor change the state to normal state*

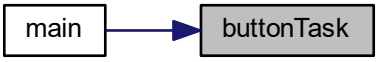
load setted value to e2prom set temp for temp control task fire temp control task flag reset timeout counter change state to normal state

Definition at line 136 of file main.c.

Here is the call graph for this function:



Here is the caller graph for this function:

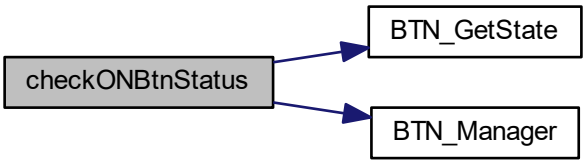


5.1.2.3 checkONBtnStatus()

```
void checkONBtnStatus ( )
```

Definition at line 354 of file main.c.

Here is the call graph for this function:



5.1.2.4 main()

```
void main (
             void  )
```

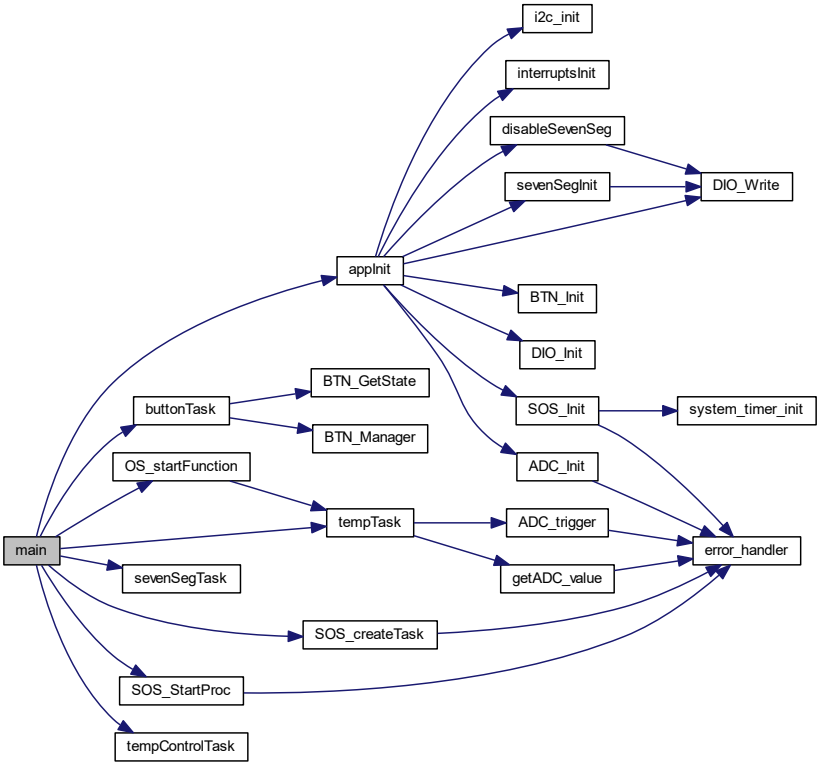
task priority is set by thier order of call not thier priority number as the priority feature is off from the small os.

if the on off btn is pressed and the system is on disable system disable system by call init function of the system

halt system and only check for the on btn

Definition at line 373 of file main.c.

Here is the call graph for this function:

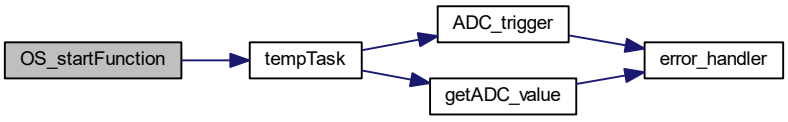


5.1.2.5 OS_startFunction()

```
void OS_startFunction ( )
```

Definition at line 368 of file main.c.

Here is the call graph for this function:



Here is the caller graph for this function:

5.1.2.6 sevenSegTask()

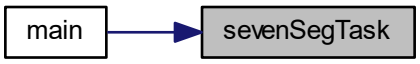
```
void sevenSegTask ( )
```

two modes for lcd {NORMAL MODE , TEMP SET MODE} if NORMAL_MODE: display current temp to the SSD else: blink SSD every 1 sec display the stored temp in EXT_EEPROM update temp based on user input up-down

control the ssd on/off time as 1 sec for each

Definition at line 224 of file main.c.

Here is the caller graph for this function:



5.1.2.7 tempControlTask()

```
void tempControlTask ( )
```

if the desired temp have been set get average temp from last 10 readings control the cooler/heater element based on themp

display the average of the avaliable readins before getting ten readings the average = average of the last x reading where x is <=10

if both heater and cooler off if temp blow critical turn on heater else *temp above critical* turn on cooler else if heater on and temp above critical turn heater off and turn cooler on and turn led on else if cooler on and temp below critical turn cooler off and turn heater on

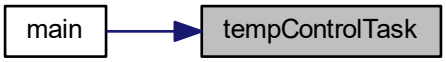
turn on the heater

turn on the cooler

if the heater on toggle the led state every 1000 ms we enter the function every 100ms so the counter count to 10 times befor toggle

Definition at line 46 of file main.c.

Here is the caller graph for this function:



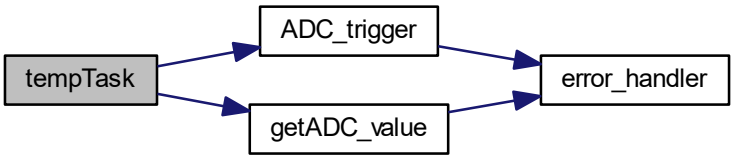
5.1.2.8 tempTask()

```
void tempTask ( )
```

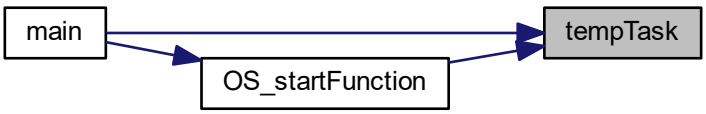
trigger ADC on specific chanal get temp from adc value set adc value in it's pos in temp array

Definition at line 284 of file main.c.

Here is the call graph for this function:



Here is the caller graph for this function:



5.2 latex/latexdocumentation.md File Reference

5.3 F:/Carier/Embedded/PIC/ElectricalWaterHeater/HAL/7SEG/7seg.c File Reference

Functions

- ERROR_STATE sevenSegInit (void)
- ERROR_STATE disableSevenSeg (void)
- ERROR_STATE sevenSegSendChar (const uint8_t u8_value, const uint8_t u8_SSD_selector)

Variables

- const uint8_t gcau8_sevenSegment_valueTable [NUM_OF_DISP_VALUES] ={0x3F,0x06,0x5B,0x4F,0x66,0x6D,0x7D,0x07,0x7

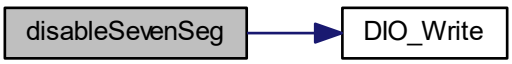
5.3.1 Function Documentation

5.3.1.1 disableSevenSeg()

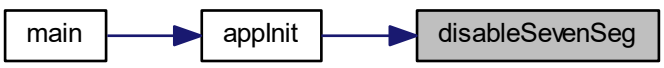
```
ERROR_STATE disableSevenSeg (
            void  )
```

Definition at line 52 of file 7seg.c.

Here is the call graph for this function:



Here is the caller graph for this function:



5.3.1.2 sevenSegInit()

```
ERROR_STATE sevenSegInit (
            void  )
```

Function prototype: void sevenSegInit(void)

Summary: initalize 7 SEG

Description: init 7 segmet based on linking configration paramters

Precondition: No preconditions required

Parameters: void

Returns: ERROR_STATE{OK,NOK}

Example: sevenSegInit()

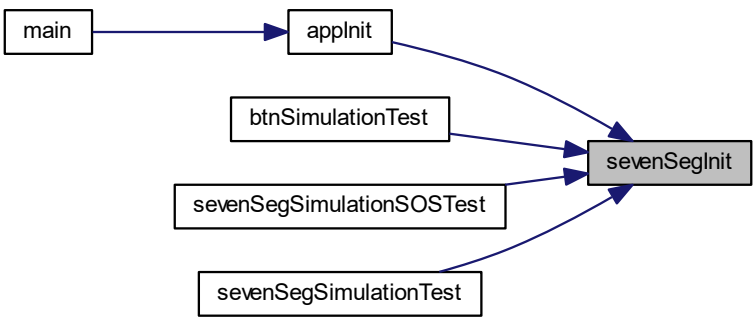
Remarks:SET SEVEN SEG DISPLAY PORT TO 0

MODULE STATE INIT

Definition at line 26 of file 7seg.c.

Here is the call graph for this function:

Here is the caller graph for this function:



5.3.1.3 sevenSegSendChar()

```
ERROR_STATE sevenSegSendChar (
             const uint8_t u8_value,
             const uint8_t u8_SSD_selector )
```

Function prototype: void sevenSegSendChar(const uint8_t value)

Summary: write to 7 SEG

Description:write w digit value to twi 7 segment display

Precondition: 7 segment must be initalized

Parameters: const uint8_t value

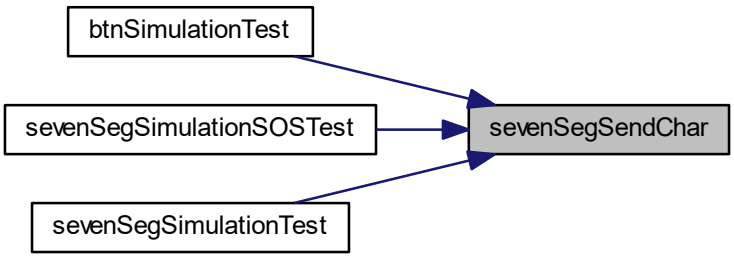
Returns: ERROR_STATE{OK,NOK}

Example: sevenSegSendChar(15)

Remarks:

Definition at line 58 of file 7seg.c.

Here is the caller graph for this function:



5.3.2 Variable Documentation

5.3.2.1 gcau8_sevenSegment_valueTable

```
const uint8_t gcau8_sevenSegment_valueTable[NUM_OF_DISP_VALUES] ={0x3F,0x06,0x5B,0x4F,0x66,0x6D,0x7D,0x07,0x7F
```

Definition at line 24 of file 7seg.c.

5.4 F:/Carier/Embedded/PIC/ElectricalWaterHeater/HAL/7SEG/7seg.h File Reference

Data Structures

- struct sevenSegConfigStruct

Functions

- ERROR_STATE sevenSegInit (void)
- ERROR_STATE sevenSegSendChar (const uint8_t u8_value, const uint8_t u8_SSD_selector)
- ERROR_STATE disableSevenSeg (void)

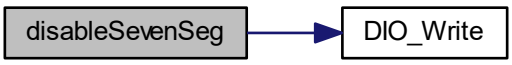
5.4.1 Function Documentation

5.4.1.1 disableSevenSeg()

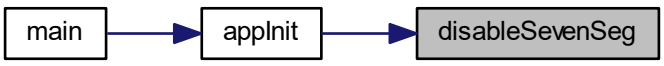
```
ERROR_STATE disableSevenSeg (
            void  )
```

Definition at line 52 of file 7seg.c.

Here is the call graph for this function:



Here is the caller graph for this function:



5.4.1.2 sevenSegInit()

```
ERROR_STATE sevenSegInit (
            void  )
```

Function prototype: void sevenSegInit(void)

Summary: initalize 7 SEG

Description: init 7 segmet based on linking configration paramters

Precondition: No preconditions required

Parameters: void

Returns: ERROR_STATE{OK,NOK}

Example: sevenSegInit()

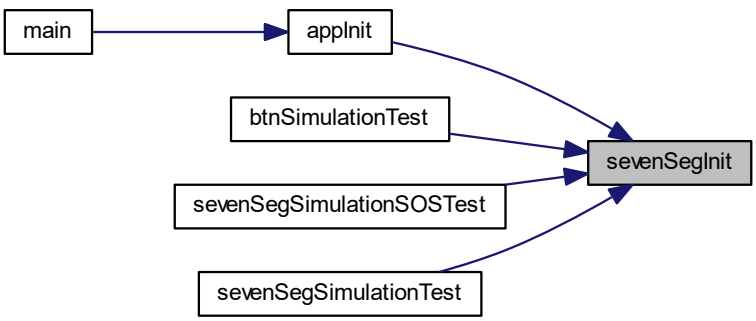
Remarks:SET SEVEN SEG DISPLAY PORT TO 0

MODULE STATE INIT

Definition at line 26 of file 7seg.c.

Here is the call graph for this function:

Here is the caller graph for this function:



5.4.1.3 sevenSegSendChar()

```
ERROR_STATE sevenSegSendChar (
            const uint8_t u8_value,
            const uint8_t u8_SSD_selector )
```

Function prototype: void sevenSegSendChar(const uint8_t value)

Summary: write to 7 SEG

Description:write w digit value to twi 7 segment display

Precondition: 7 segment must be initalized

Parameters: const uint8_t value

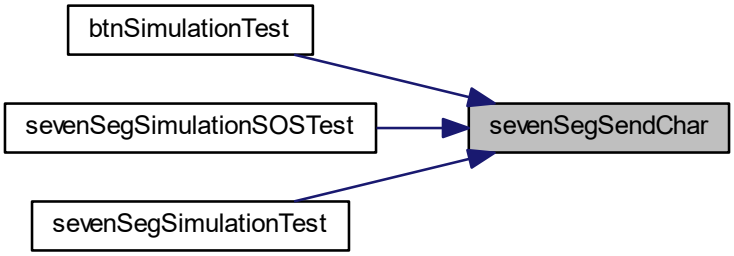
Returns: ERROR_STATE{OK,NOK}

Example: sevenSegSendChar(15)

Remarks:

Definition at line 58 of file 7seg.c.

Here is the caller graph for this function:



5.5 F:/Carier/Embedded/PIC/ElectricalWaterHeater/HAL/7SEG/7seg_Cfg.c File Reference

Variables

- const sevenSegConfigStruct sevenSegConfigParam [SEVEN_SEG_NUM]

5.5.1 Variable Documentation

5.5.1.1 sevenSegConfigParam

```
const sevenSegConfigStruct sevenSegConfigParam[SEVEN_SEG_NUM]
```

Initial value:

```
=
{
    {
        SEVEN_SEG_ONE_SELECTOR,
        SEVEN_SEG_CONGIF
    },
    {
        SEVEN_SEG_TWO_SELECTOR,
        SEVEN_SEG_CONGIF
    }
}
```

Definition at line 11 of file 7seg_Cfg.c.

5.6 F:/Carier/Embedded/PIC/ElectricalWaterHeater/HAL/7SEG/7seg_Cfg.h File Reference

Variables

- const sevenSegConfigStruct sevenSegConfigParam [SEVEN_SEG_NUM]

5.6.1 Variable Documentation

5.6.1.1 sevenSegConfigParam

```
const sevenSegConfigStruct sevenSegConfigParam[SEVEN_SEG_NUM]
```

Definition at line 11 of file 7seg_Cfg.c.

5.7 F:/Carier/Embedded/PIC/ElectricalWaterHeater/HAL/7SEG/unitTest/7seg_test.c File Reference

Functions

- void sevenSegSimulationTest ()
- void sevenSegSimulationSOSTest ()

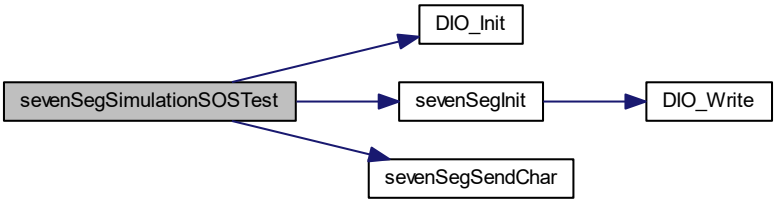
5.7.1 Function Documentation

5.7.1.1 sevenSegSimulationSOSTest()

```
void sevenSegSimulationSOSTest ( )
```

Definition at line 37 of file 7seg_test.c.

Here is the call graph for this function:

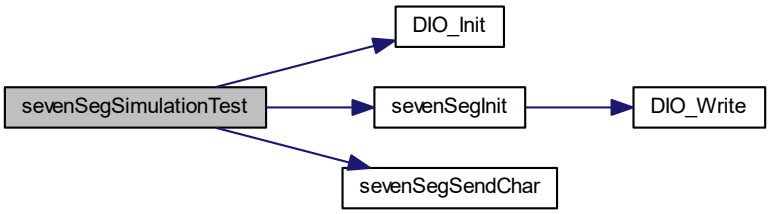


5.7.1.2 sevenSegSimulationTest()

```
void sevenSegSimulationTest ( )
```

Definition at line 10 of file 7seg_test.c.

Here is the call graph for this function:



5.8 F:/Carier/Embedded/PIC/ElectricalWaterHeater/HAL/7SEG/unitTest/7seg_test.h File Reference

Functions

- void sevenSegSimulationTest (void)
- void sevenSegSimulationSOSTest (void)

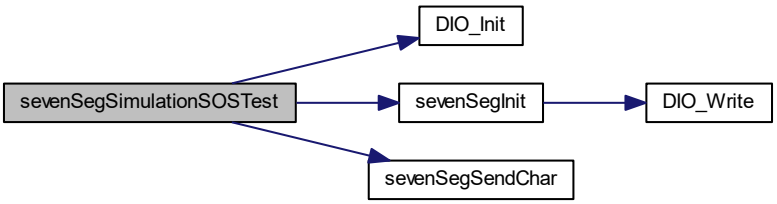
5.8.1 Function Documentation

5.8.1.1 sevenSegSimulationSOSTest()

```
void sevenSegSimulationSOSTest (
            void  )
```

Definition at line 37 of file 7seg_test.c.

Here is the call graph for this function:

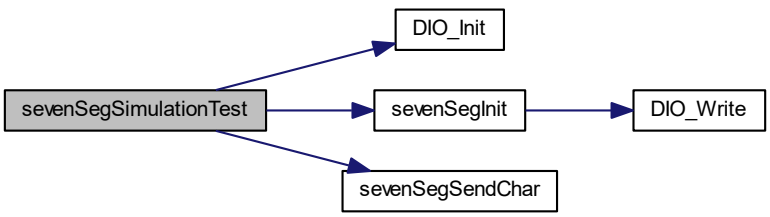


5.8.1.2 sevenSegSimulationTest()

```
void sevenSegSimulationTest (
            void  )
```

Definition at line 10 of file 7seg_test.c.

Here is the call graph for this function:



5.9 F:/Carier/Embedded/PIC/ElectricalWaterHeater/HAL/BTN/btn.c File Reference

Functions

- ERROR_STATE BTN_Init (void)
- ERROR_STATE BTN_GetState (BtnStateType *ep_BtnStatePtr, uint8_t BtnId)
- ERROR_STATE BTN_Manager (void)

5.9.1 Function Documentation

5.9.1.1 BTN_GetState()

```
ERROR_STATE BTN_GetState (
            BtnStateType * ep_BtnStatePtr,
            uint8_t u8_BtnId )
```

Function prototype: ERROR_STATE BTN_GetState(BtnStateType *ep_BtnStatePtr, uint8_t u8_BtnId)

Summary: get specified btn state

Description: return the specified btn as an output paramter pointer

Precondition: Module must be initalized through BTN_Init() function

Parameters: BtnStateType *: pointer to an enum of type BtnStatetype uint8_t : btn group id in the DIO configeration struct

Returns: ERROR_STATE{OK,NOK}

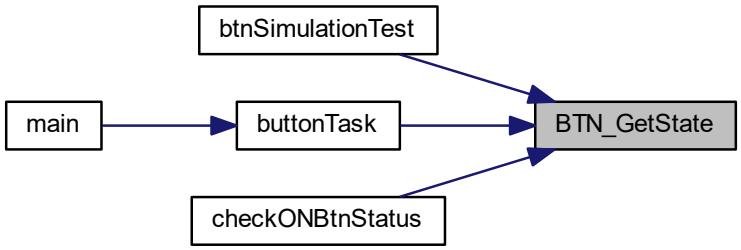
Example: BTN_GetState(&u8_on_off_btn_status,BTN0)

report error

report error

Definition at line 29 of file btn.c.

Here is the caller graph for this function:



5.9.1.2 BTN_Init()

```
ERROR_STATE BTN_Init (
            void  )
```

Function prototype: void BTN_Init(void)

Summary: initalize btns

Description: init configured btns to it's initial state as OFF

Precondition: No preconditions required

Parameters: void

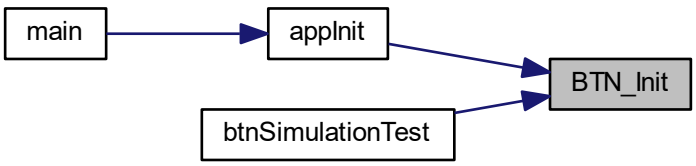
Returns: ERROR_STATE{OK,NOK}

Example: BTN_Init()

Remarks:

Definition at line 7 of file btn.c.

Here is the caller graph for this function:



5.9.1.3 BTN_Manager()

```
ERROR_STATE BTN_Manager (
            void  )
```

Function prototype: ERROR_STATE BTN_Manager(void)

Summary: serves as the btn dispatcher

Description: periodic function that update the state of each btn, recommended call time 50ms

Precondition: Module must be initalized through BTN_Init() function

Parameters: void

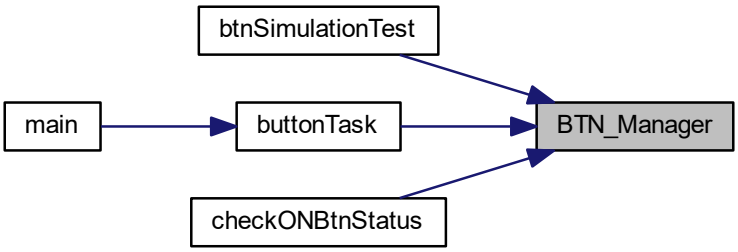
Returns: ERROR_STATE{OK,NOK}

Example: BTN_Manager();

report error

Definition at line 49 of file btn.c.

Here is the caller graph for this function:



5.10 F:/Carier/Embedded/PIC/ElectricalWaterHeater/HAL/BTN/btn.h File Reference

Data Structures

- struct BtnConfigType

Enumerations

- enum BtnStateType { BUT_OFF, BUT_ON, BUT_PRE_PRESSED, BUT_PRE_RELEASED, BUT_PRSSED, BUT_RELEASED }
- enum ActiveStateType { ActiveLow, ActiveHigh }

Functions

- ERROR_STATE BTN_Init (void)
- ERROR_STATE BTN_GetState (BtnStateType *ep_BtnStatePtr, uint8_t u8_BtnId)
- ERROR_STATE BTN_Manager (void)

5.10.1 Enumeration Type Documentation

5.10.1.1 ActiveStateType

```
enum ActiveStateType
```

Enumerator

ActiveLow	
ActiveHigh	

Definition at line 7 of file btn.h.

5.10.1.2 BtnStateType

```
enum BtnStateType
```

Enumerator

BUT_OFF	
BUT_ON	
BUT_PRE_PRESSED	
BUT_PRE_RELEASED	
BUT_PRSSED	
BUT_RELEASED	

Definition at line 6 of file btn.h.

5.10.2 Function Documentation

5.10.2.1 BTN_GetState()

```
ERROR_STATE BTN_GetState (
            BtnStateType * ep_BtnStatePtr,
            uint8_t u8_BtnId )
```

Function prototype: ERROR_STATE BTN_GetState(BtnStateType *ep_BtnStatePtr, uint8_t u8_BtnId)

Summary: get specified btn state

Description: return the specified btn as an output paramter pointer

Precondition: Module must be initalized through BTN_Init() function

Parameters: BtnStateType *: pointer to an enum of type BtnStatetype uint8_t : btn group id in the DIO configeration struct

Returns: ERROR_STATE{OK,NOK}

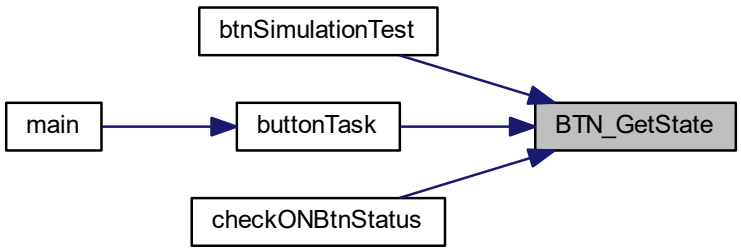
Example: BTN_GetState(&u8_on_off_btn_status,BTN0)

report error

report error

Definition at line 29 of file btn.c.

Here is the caller graph for this function:



5.10.2.2 BTN_Init()

```
ERROR_STATE BTN_Init (
             void  )
```

Function prototype: void BTN_Init(void)

Summary: initalize btns

Description: init configured btns to it's initial state as OFF

Precondition: No preconditions required

Parameters: void

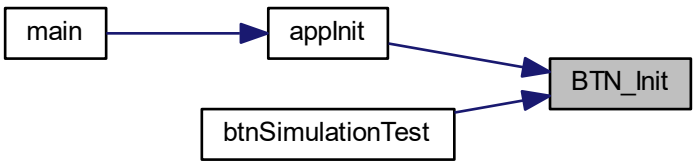
Returns: ERROR_STATE{OK,NOK}

Example: BTN_Init()

Remarks:

Definition at line 7 of file btn.c.

Here is the caller graph for this function:



5.10.2.3 BTN_Manager()

```
ERROR_STATE BTN_Manager (
            void  )
```

Function prototype: ERROR_STATE BTN_Manager(void)

Summary: serves as the btn dispatcher

Description: periodic function that update the state of each btn, recommended call time 50ms

Precondition: Module must be initalized through BTN_Init() function

Parameters: void

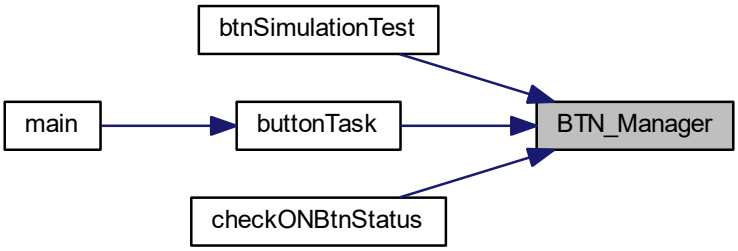
Returns: ERROR_STATE{OK,NOK}

Example: BTN_Manager();

report error

Definition at line 49 of file btn.c.

Here is the caller graph for this function:



5.11 F:/Carier/Embedded/PIC/ElectricalWaterHeater/HAL/BTN/btn_Cfg.c File Reference

Variables

- const BtnConfigType gcae_BUT_ConfigParam [BTN_NUM_OF_BUTTONS]

5.11.1 Variable Documentation

5.11.1.1 gcae_BUT_ConfigParam

```
const BtnConfigType gcae_BUT_ConfigParam[BTN_NUM_OF_BUTTONS]
```

Initial value:

```
=
{
    {
        0x03,
        ActiveLow
    },
    {
        0x04,
        ActiveLow
    },
    {
        0x05,
        ActiveLow
    }
}
```

Definition at line 4 of file btn_Cfg.c.

5.12 F:/Carier/Embedded/PIC/ElectricalWaterHeater/HAL/BTN/btn_Cfg.h File Reference

Variables

- const BtnConfigType gcae_BUT_ConfigParam [BTN_NUM_OF_BUTTONS]

5.12.1 Variable Documentation

5.12.1.1 gcae_BUT_ConfigParam

```
const BtnConfigType gcae_BUT_ConfigParam[BTN_NUM_OF_BUTTONS]
```

Definition at line 4 of file btn_Cfg.c.

5.13 F:/Carier/Embedded/PIC/ElectricalWaterHeater/HAL/BTN/unit test/btn_test.c File Reference

Functions

- void btnSimulationTest (void)

Variables

- BtnStateType u8_on_off_btn_status
- BtnStateType u8_up_btn_status
- BtnStateType u8_down_btn2_status

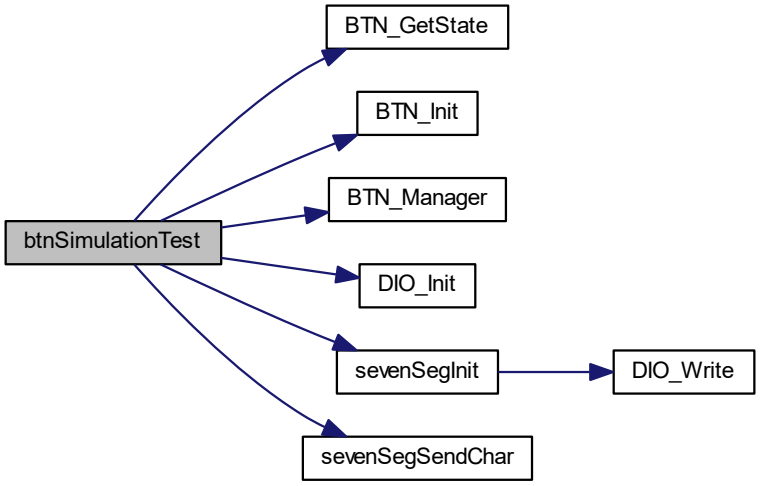
5.13.1 Function Documentation

5.13.1.1 btnSimulationTest()

```
void btnSimulationTest (
            void  )
```

Definition at line 17 of file btn_test.c.

Here is the call graph for this function:



5.13.2 Variable Documentation

5.13.2.1 u8_down_btn2_status

```
BtnStateType u8_down_btn2_status
```

Definition at line 14 of file btn_test.c.

5.13.2.2 u8_on_off_btn_status

```
BtnStateType u8_on_off_btn_status
```

Definition at line 12 of file btn_test.c.

5.13.2.3 u8_up_btn_status

```
BtnStateType u8_up_btn_status
```

Definition at line 13 of file btn_test.c.

5.14 F:/Carier/Embedded/PIC/ElectricalWaterHeater/HAL/BTN/unit test/btn_test.h File Reference

Functions

- void btnSimulationTest (void)

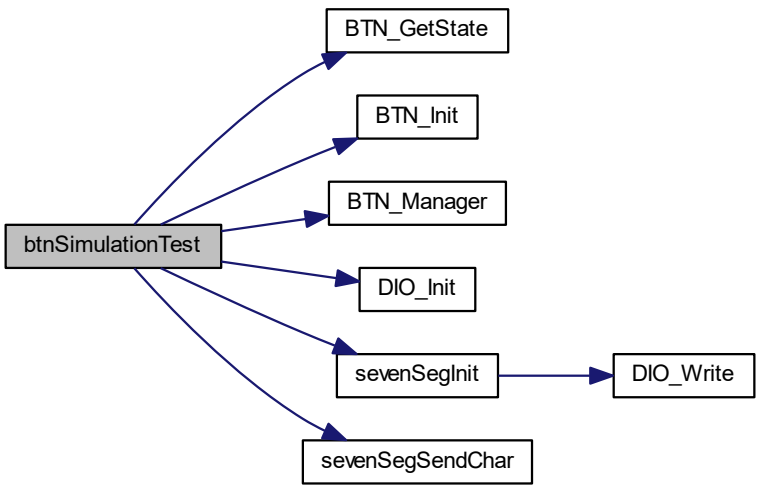
5.14.1 Function Documentation

5.14.1.1 btnSimulationTest()

```
void btnSimulationTest (
            void  )
```

Definition at line 17 of file btn_test.c.

Here is the call graph for this function:



5.15 F:/Carier/Embedded/PIC/ElectricalWaterHeater/HAL/eeprom_ext/eeprom_ext.c File Reference

Functions

- unsigned char e2pext_r (unsigned int addr)
- void e2pext_w (unsigned int addr, unsigned char val)

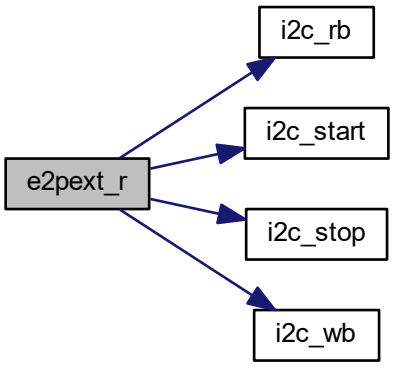
5.15.1 Function Documentation

5.15.1.1 e2pext_r()

```
unsigned char e2pext_r (
            unsigned int addr )
```

Definition at line 29 of file eeprom_ext.c.

Here is the call graph for this function:



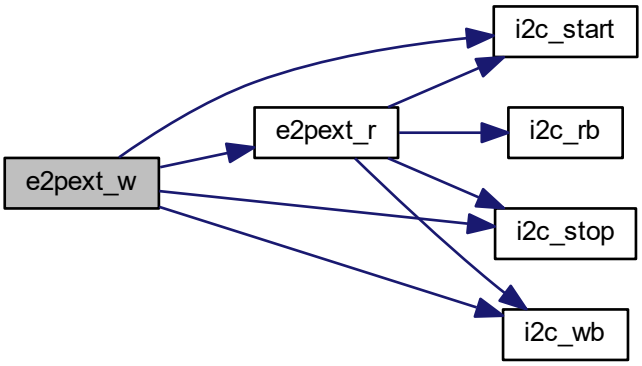
Here is the caller graph for this function:

5.15.1.2 e2pext_w()

```
void e2pext_w (
            unsigned int addr,
            unsigned char val )
```

Definition at line 65 of file eeprom_ext.c.

Here is the call graph for this function:



5.16 F:/Carier/Embedded/PIC/ElectricalWaterHeater/HAL/eeprom_ext/eeprom_ext.h File Reference

Functions

- unsigned char e2pext_r (unsigned int addr)
- void e2pext_w (unsigned int addr, unsigned char val)

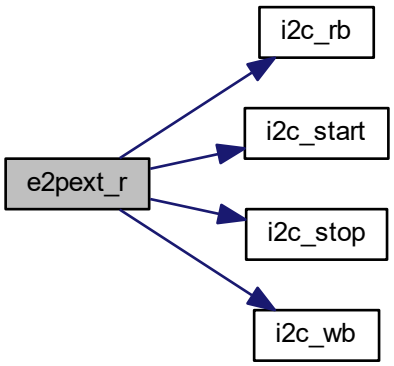
5.16.1 Function Documentation

5.16.1.1 e2pext_r()

```
unsigned char e2pext_r (
            unsigned int addr )
```

Definition at line 29 of file eeprom_ext.c.

Here is the call graph for this function:



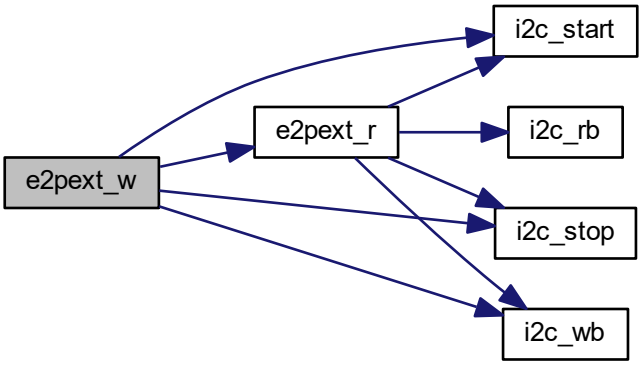
Here is the caller graph for this function:

5.16.1.2 e2pext_w()

```
void e2pext_w (
            unsigned int addr,
            unsigned char val )
```

Definition at line 65 of file eeprom_ext.c.

Here is the call graph for this function:



5.17 F:/Carier/Embedded/PIC/ElectricalWaterHeater/Kit_info/boardExamples.txt File Reference

5.18 F:/Carier/Embedded/PIC/ElectricalWaterHeater/MCAL/ADC/adc.c File Reference

Functions

- ERROR_STATE ADC_Init (void)
- ERROR_STATE ADC_trigger (uint8_t u8_canal)
- ERROR_STATE getADC_value (uint16_t *u16p_adcValue)

Variables

- uint16_t gu8_ADC_ADCValue
- uint8_t gu8_ADC_State =ADC_IDLE_STATE

5.18.1 Function Documentation

5.18.1.1 ADC_Init()

```
ERROR_STATE ADC_Init (
             void  )
```

Description:

This function is used to initialize the adc based on the configuration in adc_cfg module.

PRE-CONDITION: Module must be idle *not initalized befor*, DIO must be intalized

POST-CONDITION:

Returns

ERROR_STATE{OK,NOK}.

Example Example:

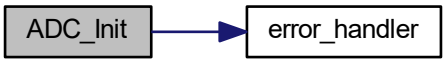
```
ADC_Init();
```

See also

Dio_Init

Definition at line 41 of file adc.c.

Here is the call graph for this function:



Here is the caller graph for this function:

5.18.1.2 ADC_trigger()

```
ERROR_STATE ADC_trigger (
            uint8_t u8_canal )
```

Description:

This function is used to trigger adc on specified chanal

PRE-CONDITION: Module must be initalized, DIO must be intalized

POST-CONDITION: when converstion done gu8_ADC_State upadate with the conversion value

Returns

ERROR_STATE{OK,NOK}.

Example Example:

```
ADC_UpdateValue(2);
```

See also

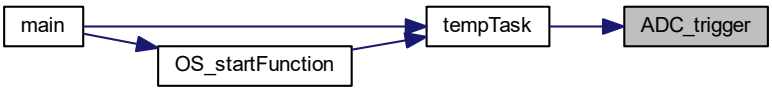
Dio_Init

ADC_Init

Definition at line 67 of file adc.c.

Here is the call graph for this function:

Here is the caller graph for this function:



5.18.1.3 getADC_value()

```
ERROR_STATE getADC_value (
            uint16_t * u16p_adcValue )
```

Description: getADC_value return last value of ADC

Parameters

out	*u16p_adcValue*	pointer to hold data of last conversion

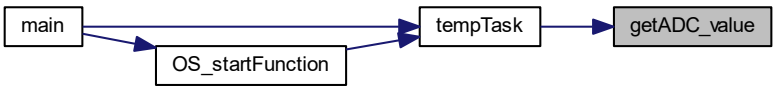
Returns

: ERROR_STATUS [OK,NOK]

Definition at line 106 of file adc.c.

Here is the call graph for this function:

Here is the caller graph for this function:



5.18.2 Variable Documentation

5.18.2.1 gu8_ADC_ADCValue

```
uint16_t gu8_ADC_ADCValue
```

Definition at line 36 of file adc.c.

5.18.2.2 gu8_ADC_State

```
uint8_t gu8_ADC_State =ADC_IDLE_STATE
```

Definition at line 37 of file adc.c.

5.19 F:/Carier/Embedded/PIC/ElectricalWaterHeater/MCAL/ADC/adc.h File Reference

Data Structures

- struct gstr_ADC_ConfigParam_t

Functions

- ERROR_STATE ADC_Init (void)
- ERROR_STATE ADC_trigger (uint8_t u8_canal)
- ERROR_STATE getADC_value (uint16_t *u16p_adcValue)

Variables

- uint16_t gu8_ADC_ADCValue

5.19.1 Function Documentation

5.19.1.1 ADC_Init()

```
ERROR_STATE ADC_Init (
             void  )
```

Description:

This function is used to initialize the adc based on the configuration in adc_cfg module.

PRE-CONDITION: Module must be idle *not initalized befor*, DIO must be intalized

POST-CONDITION:

Returns

ERROR_STATE{OK,NOK}.

Example Example:

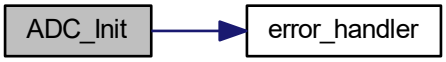
```
ADC_Init();
```

See also

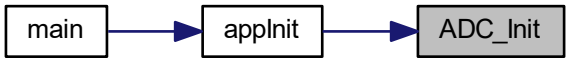
Dio_Init

Definition at line 41 of file adc.c.

Here is the call graph for this function:



Here is the caller graph for this function:



5.19.1.2 ADC_trigger()

```
ERROR_STATE ADC_trigger (
              uint8_t u8_canal )
```

Description:

This function is used to trigger adc on specified chanal

PRE-CONDITION: Module must be initalized, DIO must be intalized

POST-CONDITION: when converstion done gu8_ADC_State upadate with the conversion value

Returns

ERROR_STATE{OK,NOK}.

Example Example:

```
ADC_UpdateValue(2);
```

See also

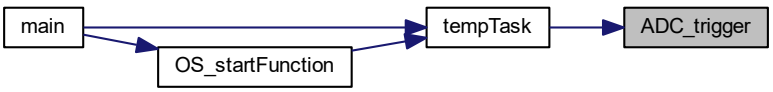
Dio_Init

ADC_Init

Definition at line 67 of file adc.c.

Here is the call graph for this function:

Here is the caller graph for this function:



5.19.1.3 getADC_value()

```
ERROR_STATE getADC_value (
            uint16_t * u16p_adcValue )
```

Description: getADC_value return last value of ADC

Parameters

out	*u16p_adcValue*	pointer to hold data of last conversion

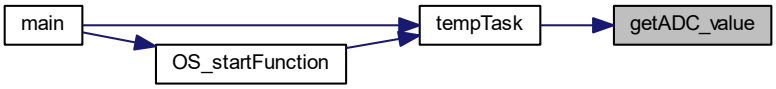
Returns

: ERROR_STATUS [OK,NOK]

Definition at line 106 of file adc.c.

Here is the call graph for this function:

Here is the caller graph for this function:



5.19.2 Variable Documentation

5.19.2.1 gu8_ADC_ADCValue

```
uint16_t gu8_ADC_ADCValue
```

Definition at line 36 of file adc.c.

5.20 F:/Carier/Embedded/PIC/ElectricalWaterHeater/MCAL/ADC/adc_Cfg.c File Reference

Variables

- const gstr_ADC_ConfigParam_t gstr_ADC_Config

5.20.1 Variable Documentation

5.20.1.1 gstr_ADC_Config

```
const gstr_ADC_ConfigParam_t gstr_ADC_Config
```

Initial value:

```
= {
                    0x02
                  ,0x00
}
```

Definition at line 10 of file adc_Cfg.c.

5.21 F:/Carier/Embedded/PIC/ElectricalWaterHeater/MCAL/ADC/adc_Cfg.h File Reference

Variables

- const gstr_ADC_ConfigParam_t gstr_ADC_Config

5.21.1 Variable Documentation

5.21.1.1 gstr_ADC_Config

```
const gstr_ADC_ConfigParam_t gstr_ADC_Config
```

Definition at line 10 of file adc_Cfg.c.

5.22 F:/Carier/Embedded/PIC/ElectricalWaterHeater/MCAL/DIO/DIO.c File Reference

Functions

- ERROR_STATE DIO_Init (void)
- ERROR_STATE DIO_Write (uint8_t u8_GroupId, uint8_t u8_Data)
- ERROR_STATE DIO_Toggle (uint8_t u8_GroupId)
- ERROR_STATE DIO_Read (uint8_t u8_GroupId, uint8_t *u8p_DataPtr)

5.22.1 Function Documentation

5.22.1.1 DIO_Init()

```
ERROR_STATE DIO_Init (
            void  )
```

Function prototype: void DIO_Init(void)

Summary: initialized all project pins for init values

Description: loop through the array of pin configuration and configure each pin as specified

Precondition: No preconditions required

Parameters: void

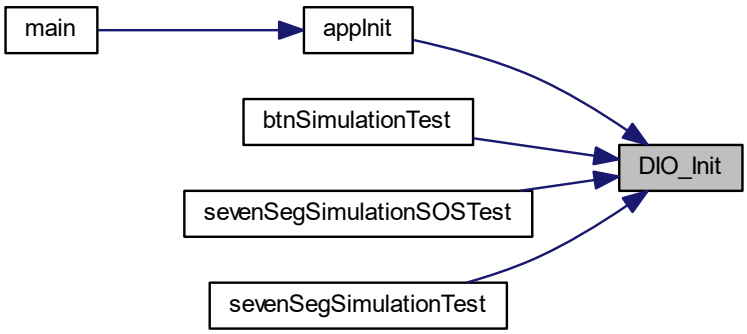
Returns: void

Example: DIO_Init()

Remarks:

Definition at line 23 of file DIO.c.

Here is the caller graph for this function:



5.22.1.2 DIO_Read()

```
ERROR_STATE DIO_Read (
            uint8_t GroupId,
            uint8_t * DataPtr )
```

Description:

This function is used to read the value of specified group_id

PRE-CONDITION: Module must be initialized

POST-CONDITION:

Returns

ERROR_STATE{OK,NOK}.

Example Example:

```
DIO_Read(3,&retValue);
```

See also

Dio_Init

Definition at line 135 of file DIO.c.

5.22.1.3 DIO_Toggle()

```
ERROR_STATE DIO_Toggle (
            uint8_t u8_GroupId )
```

Description:

This function is used to flip the value of specified group_id

PRE-CONDITION: Module must be initialized

POST-CONDITION:

Returns

ERROR_STATE{OK,NOK}.

Example Example:

```
DIO_Toggle(3);
```

See also

Dio_Init

Definition at line 100 of file DIO.c.

5.22.1.4 DIO_Write()

```
ERROR_STATE DIO_Write (
             uint8_t GroupId,
             uint8_t Data )
```

Description:

This function write certain value to specified group configuration in the DioConfigParam_t struct

PRE-CONDITION: Module must be idle initialized,

POST-CONDITION:

Returns

ERROR_STATE{OK,NOK}.

Example Example:

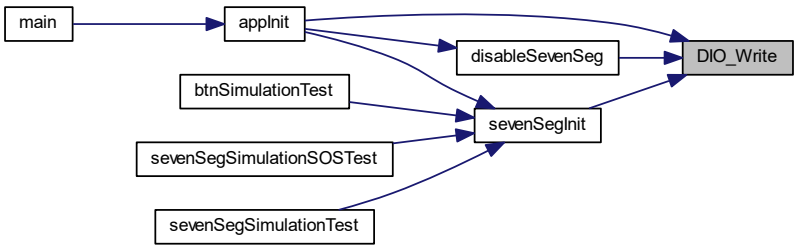
```
DIO_Write(0,0x3D);
```

See also

Dio_Init

Definition at line 63 of file DIO.c.

Here is the caller graph for this function:



5.23 F:/Carier/Embedded/PIC/ElectricalWaterHeater/MCAL/DIO/DIO.h File Reference

Data Structures

- struct DioConfigParam_t

Functions

- ERROR_STATE DIO_Init (void)
- ERROR_STATE DIO_Write (uint8_t GroupId, uint8_t Data)
- ERROR_STATE DIO_Toggle (uint8_t u8_GroupId)
- ERROR_STATE DIO_Read (uint8_t GroupId, uint8_t *DataPtr)

5.23.1 Function Documentation

5.23.1.1 DIO_Init()

```
ERROR_STATE DIO_Init (
            void  )
```

Function prototype: void DIO_Init(void)

Summary: initialized all project pins for init values

Description: loop through the array of pin configuration and configure each pin as specified

Precondition: No preconditions required

Parameters: void

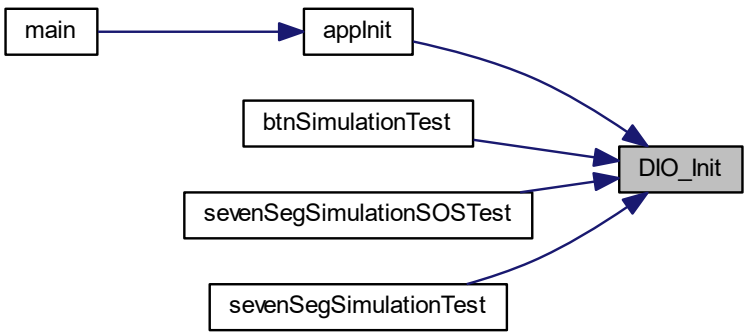
Returns: void

Example: DIO_Init()

Remarks:

Definition at line 23 of file DIO.c.

Here is the caller graph for this function:



5.23.1.2 DIO_Read()

```
ERROR_STATE DIO_Read (
            uint8_t GroupId,
            uint8_t * DataPtr )
```

Description:

This function is used to read the value of specified group_id

PRE-CONDITION: Module must be initialized

POST-CONDITION:

Returns

ERROR_STATE{OK,NOK}.

Example Example:

```
DIO_Read(3,&retValue);
```

See also

Dio_Init

Definition at line 135 of file DIO.c.

5.23.1.3 DIO_Toggle()

```
ERROR_STATE DIO_Toggle (
            uint8_t u8_GroupId )
```

Description:

This function is used to flip the value of specified group_id

PRE-CONDITION: Module must be initialized

POST-CONDITION:

Returns

ERROR_STATE{OK,NOK}.

Example Example:

```
DIO_Toggle(3);
```

See also

Dio_Init

Definition at line 100 of file DIO.c.

5.23.1.4 DIO_Write()

```
ERROR_STATE DIO_Write (
             uint8_t GroupId,
             uint8_t Data )
```

Description:

This function write certain value to specified group configuration in the DioConfigParam_t struct

PRE-CONDITION: Module must be idle initialized,

POST-CONDITION:

Returns

ERROR_STATE{OK,NOK}.

Example Example:

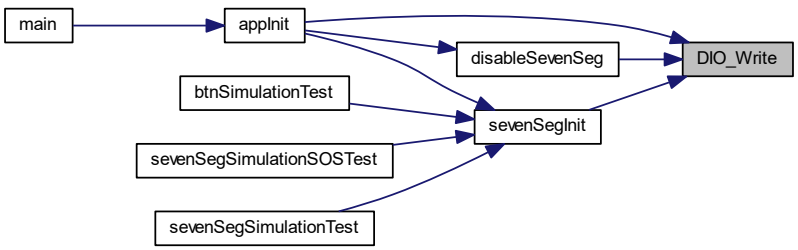
```
DIO_Write(0,0x3D);
```

See also

Dio_Init

Definition at line 63 of file DIO.c.

Here is the caller graph for this function:



5.24 F:/Carier/Embedded/PIC/ElectricalWaterHeater/MCAL/DIO/DIO_Cfg.c File Reference

Variables

- const DioConfigParam_t gstr_DIO_ConfigParam [DIO_NUM_OF_GROUPS]

5.24.1 Variable Documentation

5.24.1.1 gstr_DIO_ConfigParam

```
const DioConfigParam_t gstr_DIO_ConfigParam[DIO_NUM_OF_GROUPS]
```

Definition at line 4 of file DIO_Cfg.c.

5.25 F:/Carier/Embedded/PIC/ElectricalWaterHeater/MCAL/DIO/DIO_Cfg.h File Reference

Variables

- const DioConfigParam_t gstr_DIO_ConfigParam [DIO_NUM_OF_GROUPS]

5.25.1 Variable Documentation

5.25.1.1 gstr_DIO_ConfigParam

```
const DioConfigParam_t gstr_DIO_ConfigParam[DIO_NUM_OF_GROUPS]
```

Definition at line 4 of file DIO_Cfg.c.

5.26 F:/Carier/Embedded/PIC/ElectricalWaterHeater/MCAL/DIO/unitTest/DIO_test.h File Reference

5.27 F:/Carier/Embedded/PIC/ElectricalWaterHeater/MCAL/I2C/i2c.c File Reference

Functions

- void i2c_init (void)
- void i2c_start (void)
- void i2c_stop (void)
- void i2c_wb (unsigned char val)
- unsigned char i2c_rb (unsigned char ack)

5.27.1 Function Documentation

5.27.1.1 i2c_init()

```
void i2c_init (
             void  )
```

Definition at line 43 of file i2c.c.

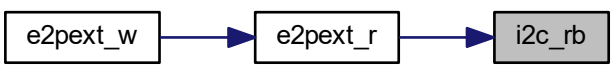
Here is the caller graph for this function:

5.27.1.2 i2c_rb()

```
unsigned char i2c_rb (
             unsigned char ack )
```

Definition at line 86 of file i2c.c.

Here is the caller graph for this function:

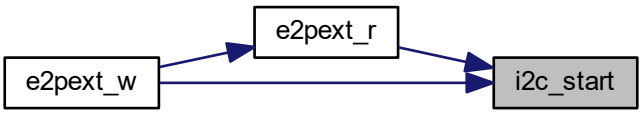


5.27.1.3 i2c_start()

```
void i2c_start (
             void  )
```

Definition at line 50 of file i2c.c.

Here is the caller graph for this function:

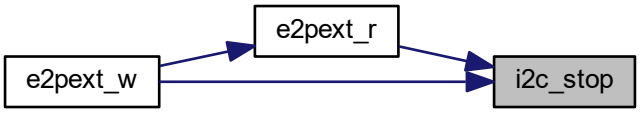


5.27.1.4 i2c_stop()

```
void i2c_stop (
             void  )
```

Definition at line 59 of file i2c.c.

Here is the caller graph for this function:

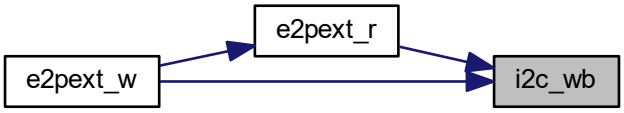


5.27.1.5 i2c_wb()

```
void i2c_wb (
            unsigned char val )
```

Definition at line 68 of file i2c.c.

Here is the caller graph for this function:



5.28 F:/Carier/Embedded/PIC/ElectricalWaterHeater/MCAL/I2C/i2c.h File Reference

Functions

- void i2c_init (void)
- void i2c_start (void)
- void i2c_stop (void)
- void i2c_wb (unsigned char val)
- unsigned char i2c_rb (unsigned char ack)
- void i2c_acktst (unsigned char val)

5.28.1 Function Documentation

5.28.1.1 i2c_acktst()

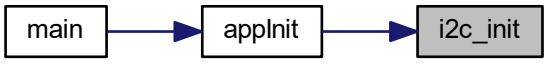
```
void i2c_acktst (
            unsigned char val )
```

5.28.1.2 i2c_init()

```
void i2c_init (
            void  )
```

Definition at line 43 of file i2c.c.

Here is the caller graph for this function:

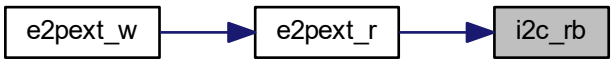


5.28.1.3 i2c_rb()

```
unsigned char i2c_rb (
            unsigned char ack )
```

Definition at line 86 of file i2c.c.

Here is the caller graph for this function:

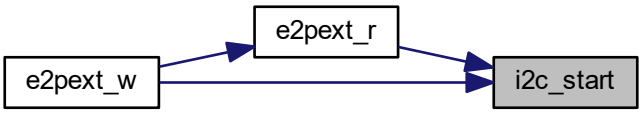


5.28.1.4 i2c_start()

```
void i2c_start (
            void  )
```

Definition at line 50 of file i2c.c.

Here is the caller graph for this function:

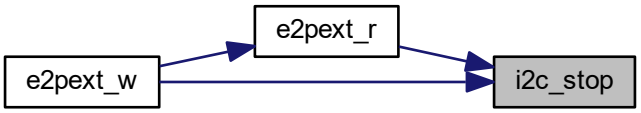


5.28.1.5 i2c_stop()

```
void i2c_stop (
            void  )
```

Definition at line 59 of file i2c.c.

Here is the caller graph for this function:

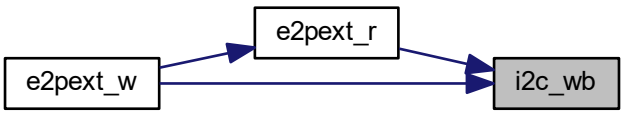


5.28.1.6 i2c_wb()

```
void i2c_wb (
            unsigned char val )
```

Definition at line 68 of file i2c.c.

Here is the caller graph for this function:



5.29 F:/Carier/Embedded/PIC/ElectricalWaterHeater/MCAL/ISR/isr.c File Reference

Functions

- void __interrupt () __ISR(void)
- void interruptsInit (void)

5.29.1 Function Documentation

5.29.1.1 __interrupt()

```
void __interrupt ( )
```

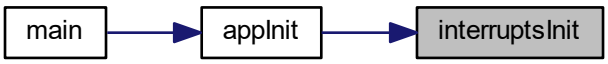
Definition at line 4 of file isr.c.

5.29.1.2 interruptsInit()

```
void interruptsInit (
             void  )
```

Definition at line 16 of file isr.c.

Here is the caller graph for this function:



5.30 F:/Carier/Embedded/PIC/ElectricalWaterHeater/MCAL/ISR/isr.h File Reference

Functions

- void interruptsInit (void)

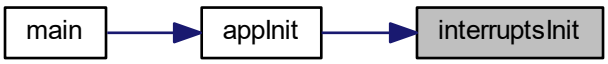
5.30.1 Function Documentation

5.30.1.1 interruptsInit()

```
void interruptsInit (
            void  )
```

Definition at line 16 of file isr.c.

Here is the caller graph for this function:



5.31 F:/Carier/Embedded/PIC/ElectricalWaterHeater/MCAL/Registers.h File Reference

5.32 F:/Carier/Embedded/PIC/ElectricalWaterHeater/MCAL/TIMER/timer.c File Reference

Functions

- ERROR_STATE system_timer_init (void)
- ERROR_STATE start_system_timer (void)

Variables

- volatile uint8_t gu8_timer_ticks

5.32.1 Function Documentation

5.32.1.1 start_system_timer()

```
ERROR_STATE start_system_timer (
            void  )
```

Description: start_system_timer start system timer

Parameters

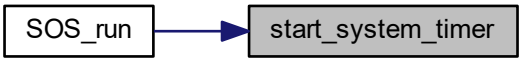
in	*void*	

Returns

: ERROR_STATUS [OK,NOK]

Definition at line 43 of file timer.c.

Here is the caller graph for this function:



5.32.1.2 system_timer_init()

```
ERROR_STATE system_timer_init (
             void  )
```

Description: system_timer_init system timer

Parameters

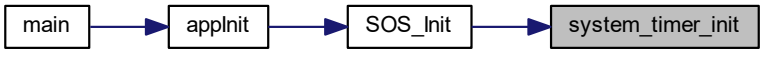
in	*void*	

Returns

: ERROR_STATUS [OK,NOK]

Definition at line 19 of file timer.c.

Here is the caller graph for this function:



5.32.2 Variable Documentation

5.32.2.1 gu8_timer_ticks

```
volatile uint8_t gu8_timer_ticks
```

Definition at line 16 of file timer.c.

5.33 F:/Carier/Embedded/PIC/ElectricalWaterHeater/MCAL/TIMER/timer.h File Reference

Functions

- ERROR_STATE system_timer_init (void)
- ERROR_STATE start_system_timer (void)

Variables

- volatile uint8_t gu8_timer_ticks

5.33.1 Function Documentation

5.33.1.1 start_system_timer()

```
ERROR_STATE start_system_timer (
            void  )
```

Description: start_system_timer start system timer

Parameters

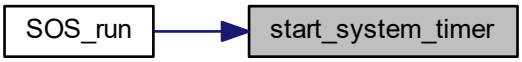
in	*void*	

Returns

: ERROR_STATUS [OK,NOK]

Definition at line 43 of file timer.c.

Here is the caller graph for this function:



5.33.1.2 system_timer_init()

```
ERROR_STATE system_timer_init (
            void  )
```

Description: system_timer_init system timer

Parameters

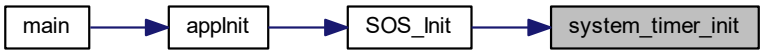
in	*void*	

Returns

: ERROR_STATUS [OK,NOK]

Definition at line 19 of file timer.c.

Here is the caller graph for this function:



5.33.2 Variable Documentation

5.33.2.1 gu8_timer_ticks

```
volatile uint8_t gu8_timer_ticks
```

Definition at line 16 of file timer.c.

5.34 F:/Carier/Embedded/PIC/ElectricalWaterHeater/ServiceLayer/common_macros.h File Reference

5.35 F:/Carier/Embedded/PIC/ElectricalWaterHeater/ServiceLayer/Config.h File Reference

5.36 F:/Carier/Embedded/PIC/ElectricalWaterHeater/ServiceLayer/Error_Handler/ErrorHandler.c File Reference

Functions

- void error_handler (sint16_t error_ID)

Variables

- STATIC uint16_t error_buffer_head = -1
- STATIC uint16_t error_Buffer [ERROR_BUFFER_SIZE]

5.36.1 Function Documentation

5.36.1.1 error_handler()

```
void error_handler (
            sint16_t error_ID )
```

Description: error_handler save error id to error buffer

Parameters

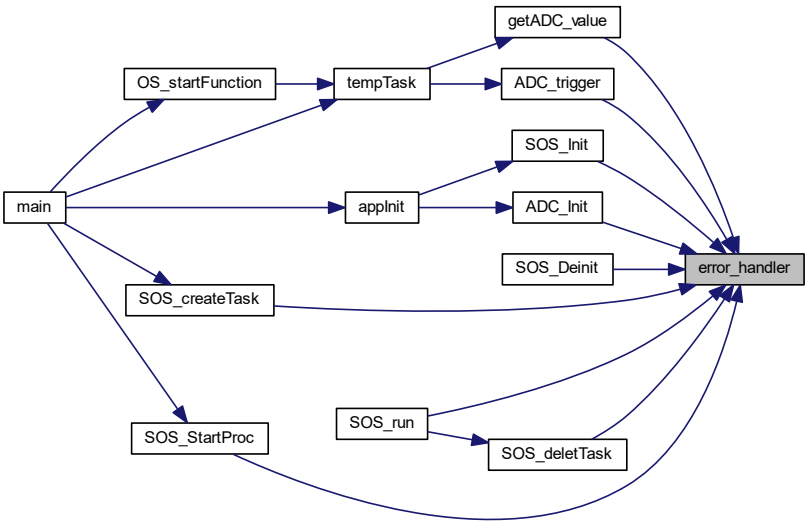
in	*error_ID*	[MODULE ERROR + ERROR_TYPE]

Returns

: void

Definition at line 14 of file ErrorHandler.c.

Here is the caller graph for this function:



5.36.2 Variable Documentation

5.36.2.1 error_Buffer

```
STATIC uint16_t error_Buffer[ERROR_BUFFER_SIZE]
```

Definition at line 12 of file ErrorHandler.c.

5.36.2.2 error_buffer_head

```
STATIC uint16_t error_buffer_head = -1
```

Definition at line 11 of file ErrorHandler.c.

5.37 F:/Carier/Embedded/PIC/ElectricalWaterHeater/ServiceLayer/Error_Handler/SystemErrors.h File Reference

Enumerations

- enum ERROR_STATE { OK =0, NOK =-1 }

Functions

- void error_handler (sint16_t error_ID)

5.37.1 Enumeration Type Documentation

5.37.1.1 ERROR_STATE

```
enum ERROR_STATE
```

Enumerator

OK	
NOK	

Definition at line 31 of file SystemErrors.h.

5.37.2 Function Documentation

5.37.2.1 error_handler()

```
void error_handler (
            sint16_t error_ID )
```

Description: error_handler save error id to error buffer

Parameters

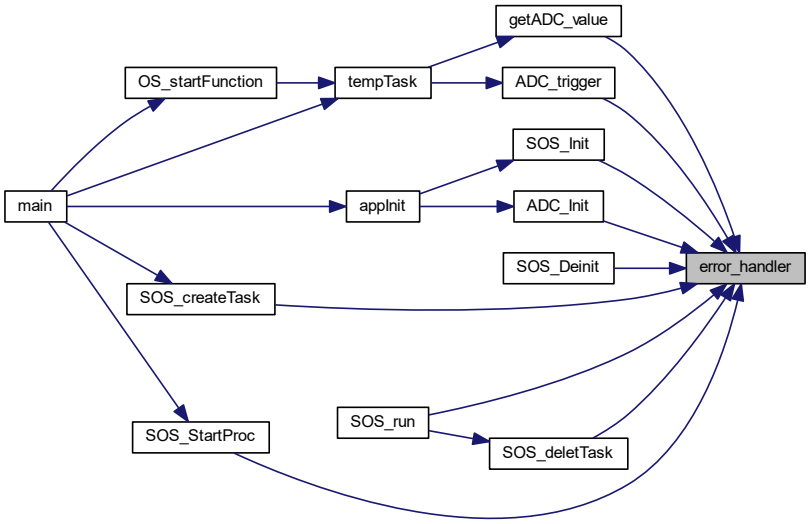
in	*error_ID*	[MODULE ERROR + ERROR_TYPE]

Returns

: void

Definition at line 14 of file ErrorHandler.c.

Here is the caller graph for this function:



5.38 F:/Carier/Embedded/PIC/ElectricalWaterHeater/ServiceLayer/SOS/SOS.c File Reference

Data Structures

- struct SOS_obj_str

Typedefs

- typedef struct SOS_obj_str gstr_SOS_obj_t

Functions

- ERROR_STATE SOS_Init (void)
- ERROR_STATE SOS_createTask (uint8_t Id, void(*callB_fun_ptr)(void), uint16_t lap_time, uint8_t type, uint8_t periority)
- ERROR_STATE SOS_deletTask (uint8_t Id)
- ERROR_STATE SOS_run (void)
- ERROR_STATE SOS_Deinit (void)
- ERROR_STATE SOS_StartProc (CBF callBackFun)

Variables

- STATIC uint8_t SOS_Init_flag = FALSE
- STATIC uint8_t SOS_Timer_ch
- STATIC gstr_SOS_obj_t gastr_SOS_ObjBuffer [SOS_OBJ_BUFFER_SIZE]
- STATIC sint8_t u8_SOS_objBufferHead
- STATIC uint8_t taken_Ids [SOS_OBJ_BUFFER_SIZE+_ONE]
- STATIC CBF gp_OS_StartProc

5.38.1 Typedef Documentation

5.38.1.1 gstr_SOS_obj_t

```
typedef struct SOS_obj_str gstr_SOS_obj_t
```

5.38.2 Function Documentation

5.38.2.1 SOS_createTask()

```
ERROR_STATE SOS_createTask (
            uint8_t Id,
            void(*)(void) callB_fun_ptr,
            uint16_t lap_time,
            uint8_t type,
            uint8_t periority )
```

Description: SOS_start add new SOS_object to be served on periodic time

Parameters

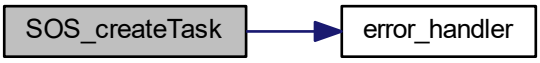
in	*id*	from 0 to SOS_OBJ_BUFFER_SIZE.
in	*callB_fun_ptr*	addresss of the rotuen to be called.
in	*lap_time*	form 0 to SOS_MAX_LAP_TIME
in	*type*	[PERIODIC,ONE_SHOT]

Returns

: ERROR_STATUS [OK,NOK]

Definition at line 65 of file SOS.c.

Here is the call graph for this function:



Here is the caller graph for this function:

5.38.2.2 SOS_Deinit()

```
ERROR_STATE SOS_Deinit (
             void  )
```

Description: SOS_DeInit Danit SOS.

Parameters

	void

Returns

: ERROR_STATUS [OK,NOK]

Definition at line 276 of file SOS.c.

Here is the call graph for this function:

5.38.2.3 SOS_deletTask()

```
ERROR_STATE SOS_deletTask (
            uint8_t Id )
```

Description: SOS_stop delete task with specified id number by removing it from SOS Buffer.

Parameters

in	*Id*	id of the task to be removed from SOS buffer 0<=id<SOS_BUFFER_MAX_SIZE

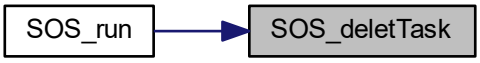
Returns

: ERROR_STATUS [OK,NOK]

Definition at line 156 of file SOS.c.

Here is the call graph for this function:

Here is the caller graph for this function:



5.38.2.4 SOS_Init()

```
ERROR_STATE SOS_Init (
            void  )
```

Description: SOS_init initialize SOS module with timer canal specified and and time tick reslution as multiple of timer tick.

Parameters

	void **NOTE: use linking configuration of sruct SOS_cfg**

Returns

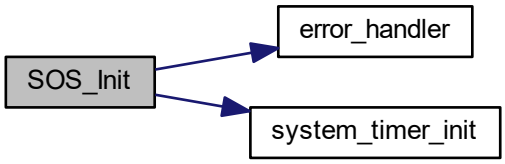
: ERROR_STATUS {OK,NOK}

See also

SOS_cfg in SOS_Cfg.c

Definition at line 31 of file SOS.c.

Here is the call graph for this function:



Here is the caller graph for this function:

5.38.2.5 SOS_run()

```
ERROR_STATE SOS_run (
             void  )
```

Description: SOS_dispatcher is the main SOS procedure, dispatcher check for new time tick then loop through the SOS_objBuffer and execute tasks at their specified time.

Parameters

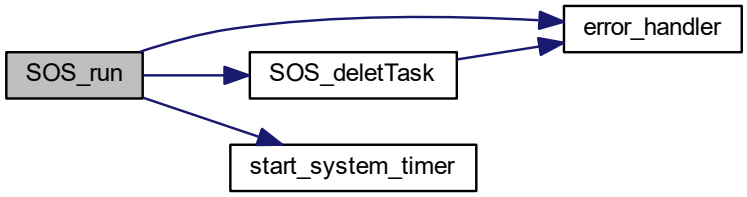
	void

Returns

: ERROR_STATUS [OK,NOK]

Definition at line 198 of file SOS.c.

Here is the call graph for this function:



5.38.2.6 SOS_StartProc()

```
ERROR_STATE SOS_StartProc (
            CBF callBackFun )
```

Description: SOS_StartProc run the start of os procedure by calling the call back function pointer passed as a paramter from the application

Parameters

in	*CBF*	function to call at the start of the os

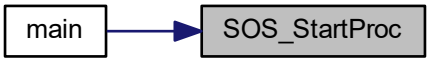
Returns

: ERROR_STATUS [OK,NOK]

Definition at line 312 of file SOS.c.

Here is the call graph for this function:

Here is the caller graph for this function:



5.38.3 Variable Documentation

5.38.3.1 gastr_SOS_ObjBuffer

```
STATIC gstr_SOS_obj_t gastr_SOS_ObjBuffer[SOS_OBJ_BUFFER_SIZE]
```

Definition at line 27 of file SOS.c.

5.38.3.2 gp_OS_StartProc

```
STATIC CBF gp_OS_StartProc
```

Definition at line 30 of file SOS.c.

5.38.3.3 SOS_Init_flag

```
STATIC uint8_t SOS_Init_flag = FALSE
```

Definition at line 25 of file SOS.c.

5.38.3.4 SOS_Timer_ch

```
STATIC uint8_t SOS_Timer_ch
```

Definition at line 26 of file SOS.c.

5.38.3.5 taken_Ids

```
STATIC uint8_t taken_Ids[SOS_OBJ_BUFFER_SIZE+_ONE]
```

Definition at line 29 of file SOS.c.

5.38.3.6 u8_SOS_objBufferHead

```
STATIC sint8_t u8_SOS_objBufferHead
```

Definition at line 28 of file SOS.c.

5.39 F:/Carier/Embedded/PIC/ElectricalWaterHeater/ServiceLayer/SOS/↩ SOS.h File Reference

Data Structures

- struct SOS_cfg_str

Typedefs

- typedef void(∗ CBF) (void)
- typedef struct SOS_cfg_str gstr_SOS_cfg_t

Functions

- ERROR_STATE SOS_Init (void)
- ERROR_STATE SOS_createTask (uint8_t Id, void(∗callB_fun_ptr)(void), uint16_t lap_time, uint8_t type, uint8_t periority)
- ERROR_STATE SOS_deletTask (uint8_t Id)
- ERROR_STATE SOS_run (void)
- ERROR_STATE SOS_Deinit (void)
- ERROR_STATE SOS_StartProc (CBF callBackFun)

5.39.1 Typedef Documentation

5.39.1.1 CBF

```
typedef void(* CBF) (void)
```

Definition at line 26 of file SOS.h.

5.39.1.2 gstr_SOS_cfg_t

```
typedef struct SOS_cfg_str gstr_SOS_cfg_t
```

5.39.2 Function Documentation

5.39.2.1 SOS_createTask()

```
ERROR_STATE SOS_createTask (
            uint8_t Id,
            void(*)(void) callB_fun_ptr,
            uint16_t lap_time,
            uint8_t type,
            uint8_t periority )
```

Description: SOS_start add new SOS_object to be served on periodic time

Parameters

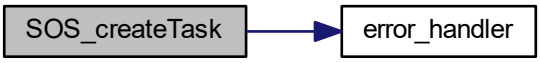
in	*id*	from 0 to SOS_OBJ_BUFFER_SIZE.
in	*callB_fun_ptr*	addresss of the rotuen to be called.
in	*lap_time*	form 0 to SOS_MAX_LAP_TIME
in	*type*	[PERIODIC,ONE_SHOT]

Returns

: ERROR_STATUS [OK,NOK]

Definition at line 65 of file SOS.c.

Here is the call graph for this function:



Here is the caller graph for this function:

5.39.2.2 SOS_Deinit()

```
ERROR_STATE SOS_Deinit (
             void  )
```

Description: SOS_DeInit Danit SOS.

Parameters

	void

Returns

: ERROR_STATUS [OK,NOK]

Definition at line 276 of file SOS.c.

Here is the call graph for this function:

5.39.2.3 SOS_deletTask()

```
ERROR_STATE SOS_deletTask (
            uint8_t Id )
```

Description: SOS_stop delete task with specified id number by removing it from SOS Buffer.

Parameters

in	*Id*	id of the task to be removed from SOS buffer 0<=id<SOS_BUFFER_MAX_SIZE

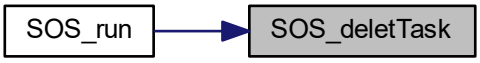
Returns

: ERROR_STATUS [OK,NOK]

Definition at line 156 of file SOS.c.

Here is the call graph for this function:

Here is the caller graph for this function:



5.39.2.4 SOS_Init()

```
ERROR_STATE SOS_Init (
            void  )
```

Description: SOS_init initialize SOS module with timer canal specified and and time tick reslution as multiple of timer tick.

Parameters

	void **NOTE: use linking configuration of sruct SOS_cfg**

Returns

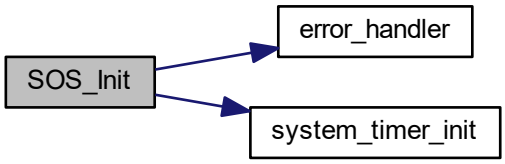
: ERROR_STATUS {OK,NOK}

See also

SOS_cfg in SOS_Cfg.c

Definition at line 31 of file SOS.c.

Here is the call graph for this function:



Here is the caller graph for this function:

5.39.2.5 SOS_run()

```
ERROR_STATE SOS_run (
              void  )
```

Description: SOS_dispatcher is the main SOS procedure, dispatcher check for new time tick then loop through the SOS_objBuffer and execute tasks at their specified time.

Parameters

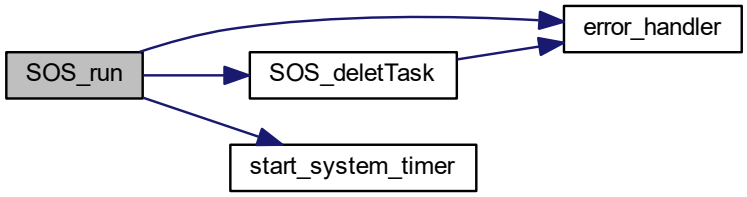
	void

Returns

: ERROR_STATUS [OK,NOK]

Definition at line 198 of file SOS.c.

Here is the call graph for this function:



5.39.2.6 SOS_StartProc()

```
ERROR_STATE SOS_StartProc (
            CBF callBackFun )
```

Description: SOS_StartProc run the start of os procedure by calling the call back function pointer passed as a paramter from the application

Parameters

in	*CBF*	function to call at the start of the os

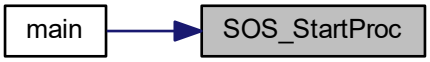
Returns

: ERROR_STATUS [OK,NOK]

Definition at line 312 of file SOS.c.

Here is the call graph for this function:

Here is the caller graph for this function:



5.40 F:/Carier/Embedded/PIC/ElectricalWaterHeater/ServiceLayer/SOS/↩ SOS_cfg.c File Reference

Variables

- gstr_SOS_cfg_t SOS_linkCfg = {SOS_TIMER_CH0,OS_TICK}

5.40.1 Variable Documentation

5.40.1.1 SOS_linkCfg

```
gstr_SOS_cfg_t SOS_linkCfg = {SOS_TIMER_CH0,OS_TICK}
```

Definition at line 12 of file SOS_cfg.c.

5.41 F:/Carier/Embedded/PIC/ElectricalWaterHeater/ServiceLayer/SOS/↩ SOS_cfg.h File Reference

Variables

- gstr_SOS_cfg_t SOS_linkCfg

5.41.1 Variable Documentation

5.41.1.1 SOS_linkCfg

```
gstr_SOS_cfg_t SOS_linkCfg
```

Definition at line 12 of file SOS_cfg.c.

5.42 F:/Carier/Embedded/PIC/ElectricalWaterHeater/ServiceLayer/std_types.h File Reference

Typedefs

- typedef unsigned char bool
- typedef unsigned char uint8_t
- typedef signed char sint8_t
- typedef unsigned int uint16_t
- typedef signed int sint16_t
- typedef unsigned long uint32_t
- typedef signed long sint32_t
- typedef unsigned long long uint64_t
- typedef signed long long sint64_t
- typedef float float32_t
- typedef double float64_t

5.42.1 Typedef Documentation

5.42.1.1 bool

```
typedef unsigned char bool
```

Definition at line 36 of file std_types.h.

5.42.1.2 float32_t

```
typedef float float32_t
```

Definition at line 72 of file std_types.h.

5.42.1.3 float64_t

```
typedef double float64_t
```

Definition at line 73 of file std_types.h.

5.42.1.4 sint16_t

```
typedef signed int sint16_t
```

Definition at line 67 of file std_types.h.

5.42.1.5 sint32_t

```
typedef signed long sint32_t
```

Definition at line 69 of file std_types.h.

5.42.1.6 sint64_t

```
typedef signed long long sint64_t
```

Definition at line 71 of file std_types.h.

5.42.1.7 sint8_t

```
typedef signed char sint8_t
```

Definition at line 65 of file std_types.h.

5.42.1.8 uint16_t

```
typedef unsigned int uint16_t
```

Definition at line 66 of file std_types.h.

5.42.1.9 uint32_t

```
typedef unsigned long uint32_t
```

Definition at line 68 of file std_types.h.

5.42.1.10 uint64_t

```
typedef unsigned long long uint64_t
```

Definition at line 70 of file std_types.h.

5.42.1.11 uint8_t

```
typedef unsigned char uint8_t
```

Definition at line 64 of file std_types.h.

Index