wlan_apdocumentation

Generated by Doxygen 1.8.8

Thu Oct 2 2014 21:16:36

Contents

Chapter 1

Module Index

1.1 Modules

Here is a list of all modules:

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

Chapter 3

Module Documentation

3.1 Getting_started_ap

Macros

- #define APP_NAME "WLAN AP"
- #define APPLICATION_VERSION "1.1.0"
- #define OSI_STACK_SIZE 2048
- #define PING_INTERVAL 1000 /∗ In msecs ∗/
- #define PING_TIMEOUT 3000 /∗ In msecs ∗/
- #define PING_PKT_SIZE 20 /∗ In bytes ∗/
- #define NO_OF_ATTEMPTS 3
- #define PING_FLAG 0

Enumerations

- enum e_AppStatusCodes { LAN_CONNECTION_FAILED = -0x7D0, CLIENT_CONNECTION_FAILED = L↩
AN_CONNECTION_FAILED - 1, DEVICE_NOT_IN_STATION_MODE = CLIENT_CONNECTION_FAILED -
1, STATUS_CODE_MAX = -0xBB8 }

Functions

- void SimpleLinkWlanEventHandler (SlWlanEvent_t ∗pSlWlanEvent)
- void SimpleLinkNetAppEventHandler (SlNetAppEvent_t ∗pNetAppEvent)

  This function handles network events such as IP acquisition, IP leased, IP released etc.
- void SimpleLinkHttpServerCallback (SlHttpServerEvent_t ∗pHttpEvent, SlHttpServerResponse_t ∗pHttp↩
Response)

  This function handles HTTP server events.
- void SimpleLinkGeneralEventHandler (SlDeviceEvent_t ∗pDevEvent)

  This function handles General Events.
- void SimpleLinkSockEventHandler (SlSockEvent_t ∗pSock)
- void SimpleLinkPingReport (SlPingReport_t ∗pPingReport)

  This function handles ping report events.
- void WlanAPMode (void ∗pvParameters)

  start simplelink, wait for the sta to connect to the device and run the ping test for that sta
- void main ()

Variables

- unsigned char g_ulStatus = 0
- unsigned long g_ulStaIp = 0
- unsigned long g_ulPingPacketsRecv = 0
- unsigned long g_uiGatewayIP = 0

3.1.1 Detailed Description

3.1.2 Macro Definition Documentation

3.1.2.1 #define APP_NAME "WLAN AP"

Definition at line 83 of file main.c.

Referenced by main().

3.1.2.2 #define APPLICATION_VERSION "1.1.0"

Definition at line 84 of file main.c.

3.1.2.3 #define NO_OF_ATTEMPTS 3

Definition at line 94 of file main.c.

3.1.2.4 #define OSI_STACK_SIZE 2048

Definition at line 85 of file main.c.

Referenced by main().

3.1.2.5 #define PING_FLAG 0

Definition at line 95 of file main.c.

3.1.2.6 #define PING_INTERVAL 1000 /∗ In msecs ∗/

Definition at line 91 of file main.c.

3.1.2.7 #define PING_PKT_SIZE 20 /∗ In bytes ∗/

Definition at line 93 of file main.c.

3.1.2.8 #define PING_TIMEOUT 3000 /∗ In msecs ∗/

Definition at line 92 of file main.c.

3.1.3 Enumeration Type Documentation

3.1.3.1 enum e_AppStatusCodes

Enumerator

LAN_CONNECTION_FAILED

CLIENT_CONNECTION_FAILED

DEVICE_NOT_IN_STATION_MODE

STATUS_CODE_MAX

Definition at line 98 of file main.c.

3.1.4 Function Documentation

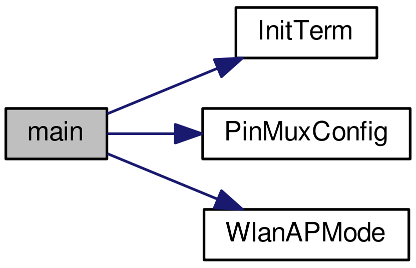
3.1.4.1 void main (void)

Definition at line 932 of file main.c.

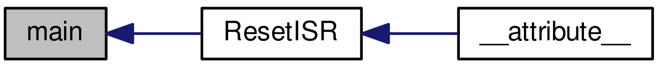
References APP_NAME, ERR_PRINT, InitTerm(), LOOP_FOREVER, OSI_STACK_SIZE, PinMuxConfig(), SPA↩ WN_TASK_PRIORITY, and WlanAPMode().

Referenced by ResetISR().

Here is the call graph for this function:



Here is the caller graph for this function:



3.1.4.2 void SimpleLinkGeneralEventHandler (SlDeviceEvent_t ∗ *pDevEvent*)

This function handles General Events.

Parameters

in	*pDevEvent*	- Pointer to General Event Info

Returns

None

Definition at line 395 of file main.c.

References UART_PRINT.

3.1.4.3 void SimpleLinkHttpServerCallback (SlHttpServerEvent_t ∗ *pHttpEvent,* SlHttpServerResponse_t ∗ *pHttpResponse*)

This function handles HTTP server events.

Parameters

in	*pServerEvent*	- Contains the relevant event information
in	*pServer←Response*	- Should be filled by the user with the relevant response information

Returns

None

Definition at line 380 of file main.c.

3.1.4.4 void SimpleLinkNetAppEventHandler (SlNetAppEvent_t ∗ *pNetAppEvent*)

This function handles network events such as IP acquisition, IP leased, IP released etc.

Parameters

in	*pNetAppEvent*	- Pointer to NetApp Event Info

Returns

None

Definition at line 325 of file main.c.

References CLR_STATUS_BIT, g_ulStaIp, g_ulStatus, SET_STATUS_BIT, STATUS_BIT_IP_AQUIRED, STATU←S_BIT_IP_LEASED, and UART_PRINT.

3.1.4.5 void SimpleLinkPingReport (SlPingReport_t ∗ *pPingReport*)

This function handles ping report events.

Parameters

in	*pPingReport*	- Ping report statistics

Returns

None

Definition at line 452 of file main.c.

References g_ulPingPacketsRecv, g_ulStatus, SET_STATUS_BIT, and STATUS_BIT_PING_DONE.

3.1.4.6 void SimpleLinkSockEventHandler (SlSockEvent_t ∗ *pSock*)

This function handles socket events indication

Parameters

`in`	*pSock*	- Pointer to Socket Event Info

Returns

None

Definition at line 416 of file main.c.

References UART_PRINT.

3.1.4.7 void SimpleLinkWlanEventHandler (SlWlanEvent_t ∗ *pSlWlanEvent*)

On Successful completion of Wlan Connect, This function triggers Connection status to be set.

Parameters

pSlWlanEvent	pointer indicating Event type

Returns

None

Definition at line 229 of file main.c.

References CLR_STATUS_BIT, g_ulStatus, SET_STATUS_BIT, STATUS_BIT_CONNECTION, STATUS_BIT_I↩
P_AQUIRED, STATUS_BIT_IP_LEASED, and UART_PRINT.

3.1.4.8 void WlanAPMode (void ∗ *pvParameters*)

start simplelink, wait for the sta to connect to the device and run the ping test for that sta

Parameters

pvparameters	is the pointer to the list of parameters that can be passed to the task while creating it

Returns

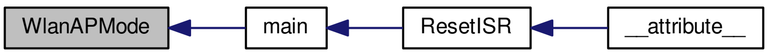
None

Definition at line 771 of file main.c.

References DEVICE_NOT_IN_STATION_MODE, ERR_PRINT, g_ulStaIp, g_ulStatus, IS_IP_ACQUIRED, IS_IP↩
_LEASED, LOOP_FOREVER, SL_STOP_TIMEOUT, UART_PRINT, and UNUSED.

Referenced by main().

Here is the caller graph for this function:



3.1.5 Variable Documentation

3.1.5.1 unsigned long g_uiGatewayIP = 0

Definition at line 114 of file main.c.

3.1.5.2 unsigned long g_ulPingPacketsRecv = 0

Definition at line 113 of file main.c.

Referenced by SimpleLinkPingReport().

3.1.5.3 unsigned long g_ulStaIp = 0

Definition at line 112 of file main.c.

Referenced by SimpleLinkNetAppEventHandler(), and WlanAPMode().

3.1.5.4 unsigned char g_ulStatus = 0

Definition at line 111 of file main.c.

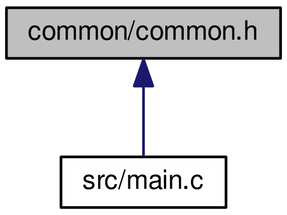
Referenced by SimpleLinkNetAppEventHandler(), SimpleLinkPingReport(), SimpleLinkWlanEventHandler(), and WlanAPMode().

Chapter 4

File Documentation

4.1 common/common.h File Reference

This graph shows which files directly or indirectly include this file:



Macros

- #define SSID_NAME "YOUR_WIFI_SSID" /∗ AP SSID ∗/
- #define SECURITY_TYPE SL_SEC_TYPE_WPA_WPA2/∗ Security type (OPEN or WEP or WPA∗/
- #define SECURITY_KEY "password" /∗ Password of the secured AP ∗/
- #define SSID_LEN_MAX 32
- #define BSSID_LEN_MAX 6
- #define UART_PRINT Report
- #define DBG_PRINT Report
- #define ERR_PRINT(x) Report("Error [%d] at line [%d] in function [%s] \n\r",x,__LINE__,__FUNCTION__)
- #define LOOP_FOREVER()
- #define ASSERT_ON_ERROR(error_code)
- #define SPAWN_TASK_PRIORITY 9
- #define SL_STOP_TIMEOUT 200
- #define UNUSED(x) ((x) = (x))
- #define SUCCESS 0
- #define FAILURE -1
- #define CLR_STATUS_BIT_ALL(status_variable) (status_variable = 0)
- #define SET_STATUS_BIT(status_variable, bit) status_variable |= (1<<(bit))
- #define CLR_STATUS_BIT(status_variable, bit) status_variable &= ∼(1<<(bit))

- #define CLR_STATUS_BIT_ALL(status_variable) (status_variable = 0)
- #define GET_STATUS_BIT(status_variable, bit) (0 != (status_variable & (1<<(bit))))
- #define IS_NW_PROCSR_ON(status_variable)
- #define IS_CONNECTED(status_variable)
- #define IS_IP_LEASED(status_variable)
- #define IS_IP_ACQUIRED(status_variable)
- #define IS_SMART_CFG_START(status_variable)
- #define IS_P2P_DEV_FOUND(status_variable)
- #define IS_P2P_REQ_RCVD(status_variable)
- #define IS_CONNECT_FAILED(status_variable)
- #define IS_PING_DONE(status_variable)

Enumerations

- enum e_StatusBits {
  STATUS_BIT_NWP_INIT = 0, STATUS_BIT_CONNECTION, STATUS_BIT_IP_LEASED, STATUS_BIT_↵
  IP_AQUIRED,
  STATUS_BIT_SMARTCONFIG_START, STATUS_BIT_P2P_DEV_FOUND, STATUS_BIT_P2P_REQ_R↵
  ECEIVED, STATUS_BIT_CONNECTION_FAILED,
  STATUS_BIT_PING_DONE }

4.1.1 Macro Definition Documentation

4.1.1.1 #define ASSERT_ON_ERROR(*error_code*)

Value:

```
{\
                 if(error_code < 0) \
                   {\
                        ERR_PRINT(error_code);\
                        return error_code;\
                 }\
            }
```

Definition at line 84 of file common.h.

4.1.1.2 #define BSSID_LEN_MAX 6

Definition at line 64 of file common.h.

4.1.1.3 #define CLR_STATUS_BIT(*status_variable*, *bit*) status_variable &= ∼(1<<(bit))

Definition at line 134 of file common.h.

Referenced by SimpleLinkNetAppEventHandler(), and SimpleLinkWlanEventHandler().

4.1.1.4 #define CLR_STATUS_BIT_ALL(*status_variable*) (status_variable = 0)

Definition at line 135 of file common.h.

4.1.1.5 #define CLR_STATUS_BIT_ALL(*status_variable*) (status_variable = 0)

Definition at line 135 of file common.h.

4.1.1.6 #define DBG_PRINT Report

Definition at line 73 of file common.h.

4.1.1.7 #define ERR_PRINT(*x*) Report("Error [%d] at line [%d] in function [%s] \n\r",x,__LINE__,__FUNCTION__)

Definition at line 74 of file common.h.

Referenced by main(), and WlanAPMode().

4.1.1.8 #define FAILURE -1

Definition at line 97 of file common.h.

4.1.1.9 #define GET_STATUS_BIT(*status_variable,* *bit*) (0 != (status_variable & (1<<(bit))))

Definition at line 136 of file common.h.

4.1.1.10 #define IS_CONNECT_FAILED(*status_variable*)

Value:

```
GET_STATUS_BIT(status_variable,\
                                                            STATUS_BIT_CONNECTION_FAILED
       )
```

Definition at line 152 of file common.h.

4.1.1.11 #define IS_CONNECTED(*status_variable*)

Value:

```
GET_STATUS_BIT(status_variable,\
                                                                   STATUS_BIT_CONNECTION)
```

Definition at line 140 of file common.h.

4.1.1.12 #define IS_IP_ACQUIRED(*status_variable*)

Value:

```
GET_STATUS_BIT(status_variable,\
                                                                    STATUS_BIT_IP_AQUIRED)
```

Definition at line 144 of file common.h.

Referenced by WlanAPMode().

4.1.1.13 #define IS_IP_LEASED(*status_variable*)

Value:

```
GET_STATUS_BIT(status_variable,\
                                                                     STATUS_BIT_IP_LEASED)
```

Definition at line 142 of file common.h.

Referenced by WlanAPMode().

4.1.1.14 #define IS_NW_PROCSR_ON(*status_variable*)

Value:

```
GET_STATUS_BIT(status_variable,\
                                                        STATUS_BIT_NWP_INIT)
```

Definition at line 138 of file common.h.

4.1.1.15 #define IS_P2P_DEV_FOUND(*status_variable*)

Value:

```
GET_STATUS_BIT(status_variable,\
                                                   STATUS_BIT_P2P_DEV_FOUND)
```

Definition at line 148 of file common.h.

4.1.1.16 #define IS_P2P_REQ_RCVD(*status_variable*)

Value:

```
GET_STATUS_BIT(status_variable,\
                                                STATUS_BIT_P2P_REQ_RECEIVED)
```

Definition at line 150 of file common.h.

4.1.1.17 #define IS_PING_DONE(*status_variable*)

Value:

```
GET_STATUS_BIT(status_variable,\
                                                       STATUS_BIT_PING_DONE)
```

Definition at line 154 of file common.h.

4.1.1.18 #define IS_SMART_CFG_START(*status_variable*)

Value:

```
GET_STATUS_BIT(status_variable,\
                                               STATUS_BIT_SMARTCONFIG_START
      )
```

Definition at line 146 of file common.h.

4.1.1.19 #define LOOP_FOREVER( )

Value:

```
{\
                while(1); \
            }
```

Definition at line 78 of file common.h.

Referenced by main(), and WlanAPMode().

4.1.1.20 #define SECURITY_KEY "password" /∗ Password of the secured AP ∗/

Definition at line 62 of file common.h.

4.1.1.21 #define SECURITY_TYPE SL_SEC_TYPE_WPA_WPA2/∗ Security type (OPEN or WEP or WPA∗/

Definition at line 61 of file common.h.

4.1.1.22 #define SET_STATUS_BIT(*status_variable, bit*) status_variable \|= (1<<(bit))

Definition at line 133 of file common.h.

Referenced by SimpleLinkNetAppEventHandler(), SimpleLinkPingReport(), and SimpleLinkWlanEventHandler().

4.1.1.23 #define SL_STOP_TIMEOUT 200

Definition at line 94 of file common.h.

Referenced by WlanAPMode().

4.1.1.24 #define SPAWN_TASK_PRIORITY 9

Definition at line 93 of file common.h.

Referenced by main().

4.1.1.25 #define SSID_LEN_MAX 32

Definition at line 63 of file common.h.

4.1.1.26 #define SSID_NAME "YOUR_WIFI_SSID" /∗ AP SSID ∗/

Definition at line 60 of file common.h.

4.1.1.27 #define SUCCESS 0

Definition at line 96 of file common.h.

4.1.1.28 #define UART_PRINT Report

Definition at line 72 of file common.h.

Referenced by SimpleLinkGeneralEventHandler(), SimpleLinkNetAppEventHandler(), SimpleLinkSockEvent↩ Handler(), SimpleLinkWlanEventHandler(), and WlanAPMode().

4.1.1.29 #define UNUSED(*x*) ((x) = (x))

Definition at line 95 of file common.h.

Referenced by WlanAPMode().

4.1.2 Enumeration Type Documentation

4.1.2.1 enum e_StatusBits

Enumerator

STATUS_BIT_NWP_INIT

STATUS_BIT_CONNECTION

STATUS_BIT_IP_LEASED

STATUS_BIT_IP_AQUIRED

STATUS_BIT_SMARTCONFIG_START

STATUS_BIT_P2P_DEV_FOUND

STATUS_BIT_P2P_REQ_RECEIVED

STATUS_BIT_CONNECTION_FAILED

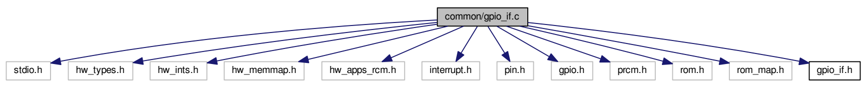
STATUS_BIT_PING_DONE

Definition at line 102 of file common.h.

4.2 common/gpio_if.c File Reference

```
#include <stdio.h>
#include "hw_types.h"
#include "hw_ints.h"
#include "hw_memmap.h"
#include "hw_apps_rcm.h"
#include "interrupt.h"
#include "pin.h"
#include "gpio.h"
#include "prcm.h"
#include "rom.h"
#include "rom_map.h"
#include "gpio_if.h"
```

Include dependency graph for gpio_if.c:



Macros

- #define PIN_LED1 9
- #define PIN_LED2 10
- #define PIN_LED3 11

Functions

- void GPIO_IF_LedConfigure (unsigned char ucPins)
- void GPIO_IF_LedOn (char ledNum)

  Turns a specific LED Off.
- void GPIO_IF_LedOff (char ledNum)

Turns a specific LED Off.
- unsigned char GPIO_IF_LedStatus (unsigned char ucGPIONum)

  This function returns LED current Status.
- void GPIO_IF_LedToggle (unsigned char ucLedNum)

  Toggles a board LED.
- void GPIO_IF_GetPortNPin (unsigned char ucPin, unsigned int ∗puiGPIOPort, unsigned char ∗pucGPIOPin)
- void GPIO_IF_ConfigureNIntEnable (unsigned int uiGPIOPort, unsigned char ucGPIOPin, unsigned int ui↩ IntType, void(∗pfnIntHandler)(void))
- void GPIO_IF_Set (unsigned char ucPin, unsigned int uiGPIOPort, unsigned char ucGPIOPin, unsigned char ucGPIOValue)
- unsigned char GPIO_IF_Get (unsigned char ucPin, unsigned int uiGPIOPort, unsigned char ucGPIOPin)

Variables

- unsigned int g_uiLED1Port = 0
- unsigned int g_uiLED2Port = 0
- unsigned int g_uiLED3Port = 0
- unsigned char g_ucLED1Pin
- unsigned char g_ucLED2Pin
- unsigned char g_ucLED3Pin

4.2.1 Macro Definition Documentation

4.2.1.1 #define PIN_LED1 9

Definition at line 82 of file gpio_if.c.

Referenced by GPIO_IF_LedConfigure(), GPIO_IF_LedOff(), and GPIO_IF_LedOn().

4.2.1.2 #define PIN_LED2 10

Definition at line 83 of file gpio_if.c.

Referenced by GPIO_IF_LedConfigure(), GPIO_IF_LedOff(), and GPIO_IF_LedOn().

4.2.1.3 #define PIN_LED3 11

Definition at line 84 of file gpio_if.c.

Referenced by GPIO_IF_LedConfigure(), GPIO_IF_LedOff(), and GPIO_IF_LedOn().

4.2.2 Function Documentation

4.2.2.1 void GPIO_IF_ConfigureNIntEnable (unsigned int *uiGPIOPort,* unsigned char *ucGPIOPin,* unsigned int *uiIntType,* void(∗)(void) *pfnIntHandler*)

Configures the GPIO selected as input to generate interrupt on activity

Parameters

uiGPIOPort	is the GPIO port address

ucGPIOPin	is the GPIO pin of the specified port
uiIntType	is the type of the interrupt (refer gpio.h)
pfnIntHandler	is the interrupt handler to register

This function

1. Sets GPIO interrupt type
2. Registers Interrupt handler
3. Enables Interrupt

Returns

None

Definition at line 372 of file gpio_if.c.

4.2.2.2 unsigned char GPIO_IF_Get (unsigned char *ucPin,* unsigned int *uiGPIOPort,* unsigned char *ucGPIOPin*)

Set a value to the specified GPIO pin

Parameters

ucPin	is the GPIO pin to be set (0:39)
uiGPIOPort	is the GPIO port address
ucGPIOPin	is the GPIO pin of the specified port

This function

1. Gets a value of the specified GPIO pin

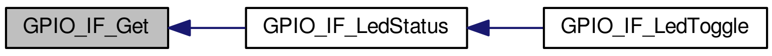
Returns

value of the GPIO pin

Definition at line 447 of file gpio_if.c.

Referenced by GPIO_IF_LedStatus().

Here is the caller graph for this function:



4.2.2.3 void GPIO_IF_GetPortNPin (unsigned char *ucPin,* unsigned int ∗ *puiGPIOPort,* unsigned char ∗ *pucGPIOPin*)

Get the port and pin of a given GPIO

Parameters

ucPin	is the pin to be set-up as a GPIO (0:39)
puiGPIOPort	is the pointer to store GPIO port address return value
pucGPIOPin	is the pointer to store GPIO pin return value

This function

1. Return the GPIO port address and pin for a given external pin number

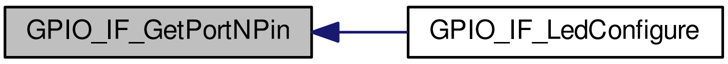
Returns

None.

Definition at line 338 of file gpio_if.c.

Referenced by GPIO_IF_LedConfigure().

Here is the caller graph for this function:



4.2.2.4 void GPIO_IF_LedConfigure (unsigned char *ucPins*)

GPIO Enable & Configuration

Parameters

ucPins	is the bit-pack representation of 3 LEDs LSB:GP09-GP10-GP11:MSB

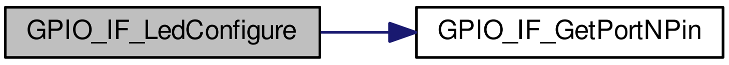
Returns

None

Definition at line 123 of file gpio_if.c.

References g_ucLED1Pin, g_ucLED2Pin, g_ucLED3Pin, g_uiLED1Port, g_uiLED2Port, g_uiLED3Port, GPIO_IF←_GetPortNPin(), LED1, LED2, LED3, PIN_LED1, PIN_LED2, and PIN_LED3.

Here is the call graph for this function:



4.2.2.5 void GPIO_IF_LedOff (char *ledNum*)

Turns a specific LED Off.

Turn LED Off

Parameters

ledNum	is the LED Number

Returns

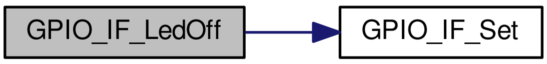
none

Definition at line 217 of file gpio_if.c.

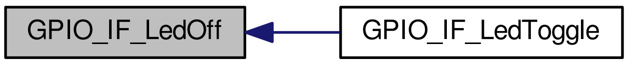
References g_ucLED1Pin, g_ucLED2Pin, g_ucLED3Pin, g_uiLED1Port, g_uiLED2Port, g_uiLED3Port, GPIO_I↩F_Set(), MCU_ALL_LED_IND, MCU_ASSOCIATED_IND, MCU_CLIENT_CONNECTED_IND, MCU_EXECUTE↩_FAIL_IND, MCU_EXECUTE_SUCCESS_IND, MCU_GREEN_LED_GPIO, MCU_IP_ALLOC_IND, MCU_ON_IN↩D, MCU_ORANGE_LED_GPIO, MCU_RED_LED_GPIO, MCU_SENDING_DATA_IND, MCU_SERVER_INIT_IND, PIN_LED1, PIN_LED2, and PIN_LED3.

Referenced by GPIO_IF_LedToggle().

Here is the call graph for this function:



Here is the caller graph for this function:



4.2.2.6 void GPIO_IF_LedOn (char *ledNum*)

Turns a specific LED Off.

Turn LED On

Parameters

ledNum	is the LED Number

Returns

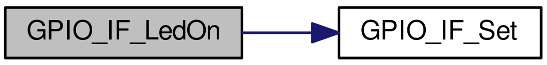
none

Definition at line 162 of file gpio_if.c.

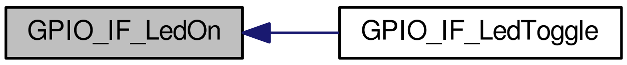
References g_ucLED1Pin, g_ucLED2Pin, g_ucLED3Pin, g_uiLED1Port, g_uiLED2Port, g_uiLED3Port, GPIO_I↩F_Set(), MCU_ALL_LED_IND, MCU_ASSOCIATED_IND, MCU_CLIENT_CONNECTED_IND, MCU_EXECUTE↩_FAIL_IND, MCU_EXECUTE_SUCCESS_IND, MCU_GREEN_LED_GPIO, MCU_IP_ALLOC_IND, MCU_ON_IN↩D, MCU_ORANGE_LED_GPIO, MCU_RED_LED_GPIO, MCU_SENDING_DATA_IND, MCU_SERVER_INIT_IND, PIN_LED1, PIN_LED2, and PIN_LED3.

Referenced by GPIO_IF_LedToggle().

Here is the call graph for this function:



Here is the caller graph for this function:



4.2.2.7 unsigned char GPIO_IF_LedStatus (unsigned char *ucGPIONum*)

This function returns LED current Status.

Parameters

`in`	*ucGPIONum*	is the GPIO to which the LED is connected MCU_GREEN_LED_GPIO

Returns

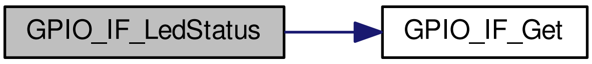
1: LED ON, 0: LED OFF

Definition at line 272 of file gpio_if.c.

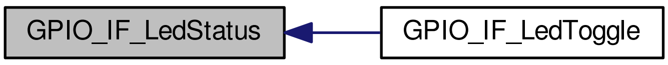
References g_ucLED1Pin, g_ucLED2Pin, g_ucLED3Pin, g_uiLED1Port, g_uiLED2Port, g_uiLED3Port, GPIO_IF↩_Get(), MCU_GREEN_LED_GPIO, MCU_ORANGE_LED_GPIO, and MCU_RED_LED_GPIO.

Referenced by GPIO_IF_LedToggle().

Here is the call graph for this function:



Here is the caller graph for this function:



4.2.2.8 void GPIO_IF_LedToggle (unsigned char *ucLedNum*)

Toggles a board LED.

Toggle the Led state

Parameters

ledNum	is the LED Number

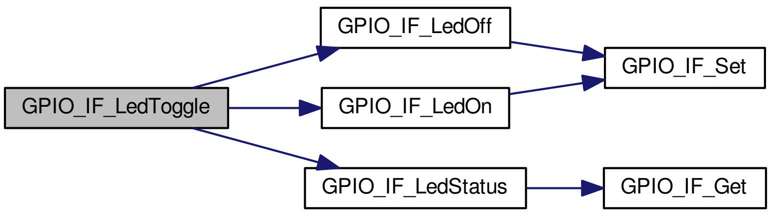
Returns

none

Definition at line 309 of file gpio_if.c.

References GPIO_IF_LedOff(), GPIO_IF_LedOn(), and GPIO_IF_LedStatus().

Here is the call graph for this function:



4.2.2.9 void GPIO_IF_Set (unsigned char *ucPin,* unsigned int *uiGPIOPort,* unsigned char *ucGPIOPin,* unsigned char *ucGPIOValue*)

Set a value to the specified GPIO pin

Parameters

ucPin	is the GPIO pin to be set (0:39)
uiGPIOPort	is the GPIO port address
ucGPIOPin	is the GPIO pin of the specified port
ucGPIOValue	is the value to be set

This function

1. Sets a value to the specified GPIO pin

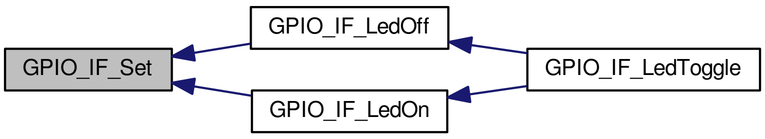
Returns

None.

Definition at line 416 of file gpio_if.c.

Referenced by GPIO_IF_LedOff(), and GPIO_IF_LedOn().

Here is the caller graph for this function:



4.2.3 Variable Documentation

4.2.3.1 unsigned char g_ucLED1Pin

Definition at line 80 of file gpio_if.c.

Referenced by GPIO_IF_LedConfigure(), GPIO_IF_LedOff(), GPIO_IF_LedOn(), and GPIO_IF_LedStatus().

4.2.3.2 unsigned char g_ucLED2Pin

Definition at line 80 of file gpio_if.c.

Referenced by GPIO_IF_LedConfigure(), GPIO_IF_LedOff(), GPIO_IF_LedOn(), and GPIO_IF_LedStatus().

4.2.3.3 unsigned char g_ucLED3Pin

Definition at line 80 of file gpio_if.c.

Referenced by GPIO_IF_LedConfigure(), GPIO_IF_LedOff(), GPIO_IF_LedOn(), and GPIO_IF_LedStatus().

4.2.3.4 unsigned int g_uiLED1Port = 0

Definition at line 79 of file gpio_if.c.

Referenced by GPIO_IF_LedConfigure(), GPIO_IF_LedOff(), GPIO_IF_LedOn(), and GPIO_IF_LedStatus().

4.2.3.5 unsigned int g_uiLED2Port = 0

Definition at line 79 of file gpio_if.c.

Referenced by GPIO_IF_LedConfigure(), GPIO_IF_LedOff(), GPIO_IF_LedOn(), and GPIO_IF_LedStatus().

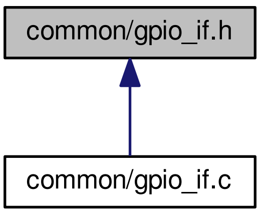
4.2.3.6 unsigned int g_uiLED3Port = 0

Definition at line 79 of file gpio_if.c.

Referenced by GPIO_IF_LedConfigure(), GPIO_IF_LedOff(), GPIO_IF_LedOn(), and GPIO_IF_LedStatus().

4.3 common/gpio_if.h File Reference

This graph shows which files directly or indirectly include this file:



Enumerations

- enum ledEnum { NO_LED, LED1 = 0x1, LED2 = 0x2, LED3 = 0x4 }
- enum ledNames {
  NO_LED_IND = NO_LED, MCU_SENDING_DATA_IND = LED1, MCU_ASSOCIATED_IND, MCU_IP_AL↩
  LOC_IND,
  MCU_SERVER_INIT_IND, MCU_CLIENT_CONNECTED_IND, MCU_ON_IND, MCU_EXECUTE_SUCCE↩
  SS_IND,
  MCU_EXECUTE_FAIL_IND, MCU_RED_LED_GPIO, MCU_ORANGE_LED_GPIO, MCU_GREEN_LED_↩
  GPIO,
  MCU_ALL_LED_IND }

Functions

- void GPIO_IF_GetPortNPin (unsigned char ucPin, unsigned int ∗puiGPIOPort, unsigned char ∗pucGPIOPin)
- void GPIO_IF_ConfigureNIntEnable (unsigned int uiGPIOPort, unsigned char ucGPIOPin, unsigned int ui↩
  IntType, void(∗pfnIntHandler)(void))
- void GPIO_IF_Set (unsigned char ucPin, unsigned int uiGPIOPort, unsigned char ucGPIOPin, unsigned char ucGPIOValue)
- unsigned char GPIO_IF_Get (unsigned char ucPin, unsigned int uiGPIOPort, unsigned char ucGPIOPin)
- void GPIO_IF_LedConfigure (unsigned char ucPins)
- void GPIO_IF_LedOn (char ledNum)

  Turns a specific LED Off.

- void GPIO_IF_LedOff (char ledNum)

  Turns a specific LED Off.
- unsigned char GPIO_IF_LedStatus (unsigned char ucGPIONum)

  This function returns LED current Status.
- void GPIO_IF_LedToggle (unsigned char ucLedNum)

  Toggles a board LED.

4.3.1 Enumeration Type Documentation

4.3.1.1 enum **ledEnum**

Enumerator

NO_LED

LED1

LED2

LED3

Definition at line 53 of file gpio_if.h.

4.3.1.2 enum **ledNames**

Enumerator

NO_LED_IND

MCU_SENDING_DATA_IND

MCU_ASSOCIATED_IND

MCU_IP_ALLOC_IND

MCU_SERVER_INIT_IND

MCU_CLIENT_CONNECTED_IND

MCU_ON_IND

MCU_EXECUTE_SUCCESS_IND

MCU_EXECUTE_FAIL_IND

MCU_RED_LED_GPIO

MCU_ORANGE_LED_GPIO

MCU_GREEN_LED_GPIO

MCU_ALL_LED_IND

Definition at line 62 of file gpio_if.h.

4.3.2 Function Documentation

4.3.2.1 void GPIO_IF_ConfigureNIntEnable (unsigned int *uiGPIOPort,* unsigned char *ucGPIOPin,* unsigned int *uiIntType,* void(∗)(void) *pfnIntHandler*)

Configures the GPIO selected as input to generate interrupt on activity

Parameters

uiGPIOPort	is the GPIO port address
ucGPIOPin	is the GPIO pin of the specified port
uiIntType	is the type of the interrupt (refer gpio.h)
pfnIntHandler	is the interrupt handler to register

This function

1. Sets GPIO interrupt type
2. Registers Interrupt handler
3. Enables Interrupt

Returns

None

Definition at line 372 of file gpio_if.c.

4.3.2.2 unsigned char GPIO_IF_Get (unsigned char *ucPin*, unsigned int *uiGPIOPort*, unsigned char *ucGPIOPin*)

Set a value to the specified GPIO pin

Parameters

ucPin	is the GPIO pin to be set (0:39)
uiGPIOPort	is the GPIO port address
ucGPIOPin	is the GPIO pin of the specified port

This function

1. Gets a value of the specified GPIO pin

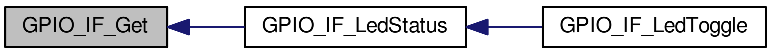
Returns

value of the GPIO pin

Definition at line 447 of file gpio_if.c.

Referenced by GPIO_IF_LedStatus().

Here is the caller graph for this function:



4.3.2.3 void GPIO_IF_GetPortNPin (unsigned char *ucPin*, unsigned int ∗ *puiGPIOPort*, unsigned char ∗ *pucGPIOPin*)

Get the port and pin of a given GPIO

Parameters

ucPin	is the pin to be set-up as a GPIO (0:39)
puiGPIOPort	is the pointer to store GPIO port address return value
pucGPIOPin	is the pointer to store GPIO pin return value

This function

1. Return the GPIO port address and pin for a given external pin number

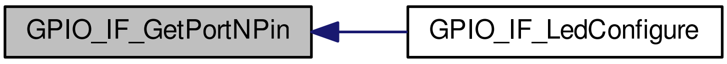
Returns

None.

Definition at line 338 of file gpio_if.c.

Referenced by GPIO_IF_LedConfigure().

Here is the caller graph for this function:



4.3.2.4 void GPIO_IF_LedConfigure (unsigned char *ucPins*)

GPIO Enable & Configuration

Parameters

ucPins	is the bit-pack representation of 3 LEDs LSB:GP09-GP10-GP11:MSB

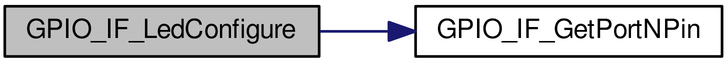
Returns

None

Definition at line 123 of file gpio_if.c.

References g_ucLED1Pin, g_ucLED2Pin, g_ucLED3Pin, g_uiLED1Port, g_uiLED2Port, g_uiLED3Port, GPIO_IF↔_GetPortNPin(), LED1, LED2, LED3, PIN_LED1, PIN_LED2, and PIN_LED3.

Here is the call graph for this function:



4.3.2.5 void GPIO_IF_LedOff (char *ledNum*)

Turns a specific LED Off.

Turn LED Off

Parameters

ledNum	is the LED Number

Returns

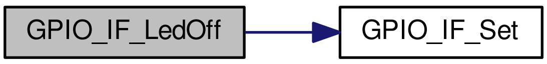
none

Definition at line 217 of file gpio_if.c.

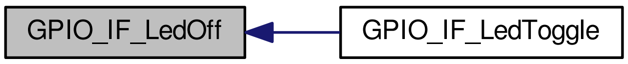
References g_ucLED1Pin, g_ucLED2Pin, g_ucLED3Pin, g_uiLED1Port, g_uiLED2Port, g_uiLED3Port, GPIO_I↩
F_Set(), MCU_ALL_LED_IND, MCU_ASSOCIATED_IND, MCU_CLIENT_CONNECTED_IND, MCU_EXECUTE↩
_FAIL_IND, MCU_EXECUTE_SUCCESS_IND, MCU_GREEN_LED_GPIO, MCU_IP_ALLOC_IND, MCU_ON_IN↩
D, MCU_ORANGE_LED_GPIO, MCU_RED_LED_GPIO, MCU_SENDING_DATA_IND, MCU_SERVER_INIT_IND, PIN_LED1, PIN_LED2, and PIN_LED3.

Referenced by GPIO_IF_LedToggle().

Here is the call graph for this function:



Here is the caller graph for this function:



4.3.2.6 void GPIO_IF_LedOn (char *ledNum*)

Turns a specific LED Off.

Turn LED On

Parameters

ledNum	is the LED Number

Returns

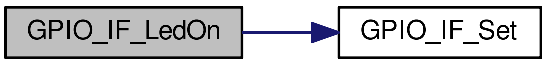
none

Definition at line 162 of file gpio_if.c.

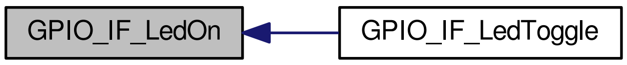
References g_ucLED1Pin, g_ucLED2Pin, g_ucLED3Pin, g_uiLED1Port, g_uiLED2Port, g_uiLED3Port, GPIO_I↵F_Set(), MCU_ALL_LED_IND, MCU_ASSOCIATED_IND, MCU_CLIENT_CONNECTED_IND, MCU_EXECUTE↵_FAIL_IND, MCU_EXECUTE_SUCCESS_IND, MCU_GREEN_LED_GPIO, MCU_IP_ALLOC_IND, MCU_ON_IN↵D, MCU_ORANGE_LED_GPIO, MCU_RED_LED_GPIO, MCU_SENDING_DATA_IND, MCU_SERVER_INIT_IND, PIN_LED1, PIN_LED2, and PIN_LED3.

Referenced by GPIO_IF_LedToggle().

Here is the call graph for this function:



Here is the caller graph for this function:



4.3.2.7 unsigned char GPIO_IF_LedStatus (unsigned char *ucGPIONum*)

This function returns LED current Status.

Parameters

`in`	*ucGPIONum*	is the GPIO to which the LED is connected MCU_GREEN_LED_GPIO

Returns

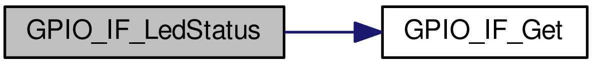
1: LED ON, 0: LED OFF

Definition at line 272 of file gpio_if.c.

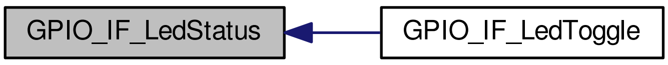
References g_ucLED1Pin, g_ucLED2Pin, g_ucLED3Pin, g_uiLED1Port, g_uiLED2Port, g_uiLED3Port, GPIO_IF↵_Get(), MCU_GREEN_LED_GPIO, MCU_ORANGE_LED_GPIO, and MCU_RED_LED_GPIO.

Referenced by GPIO_IF_LedToggle().

Here is the call graph for this function:



Here is the caller graph for this function:



4.3.2.8 void GPIO_IF_LedToggle (unsigned char *ucLedNum*)

Toggles a board LED.

Toggle the Led state

Parameters

ledNum	is the LED Number

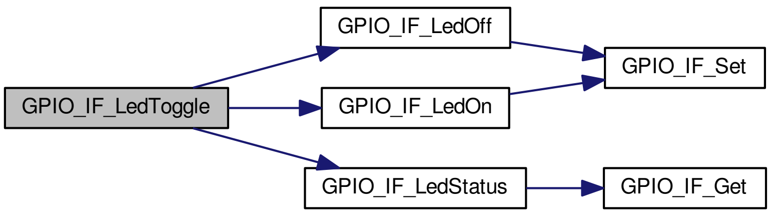
Returns

none

Definition at line 309 of file gpio_if.c.

References GPIO_IF_LedOff(), GPIO_IF_LedOn(), and GPIO_IF_LedStatus().

Here is the call graph for this function:



4.3.2.9 void GPIO_IF_Set (unsigned char *ucPin,* unsigned int *uiGPIOPort,* unsigned char *ucGPIOPin,* unsigned char *ucGPIOValue*)

Set a value to the specified GPIO pin

Parameters

ucPin	is the GPIO pin to be set (0:39)
uiGPIOPort	is the GPIO port address
ucGPIOPin	is the GPIO pin of the specified port
ucGPIOValue	is the value to be set

This function

1. Sets a value to the specified GPIO pin

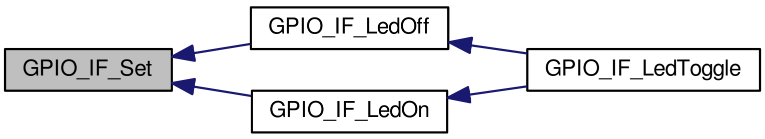
Returns

None.

Definition at line 416 of file gpio_if.c.

Referenced by GPIO_IF_LedOff(), and GPIO_IF_LedOn().

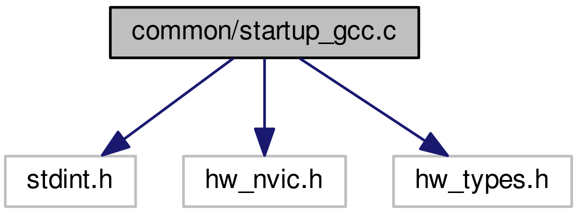
Here is the caller graph for this function:



4.4 common/startup_gcc.c File Reference

```
#include <stdint.h>
#include "hw_nvic.h"
#include "hw_types.h"
```

Include dependency graph for startup_gcc.c:



Functions

- void ResetISR (void)

- void _c_int00 (void)
- void vPortSVCHandler (void)
- void xPortPendSVHandler (void)
- void xPortSysTickHandler (void)
- int main (void)
- __attribute__ ((section(".intvecs")))
- void ∗ _sbrk (unsigned int incr)

Variables

- unsigned long _heap
- unsigned long _eheap
- uint32_t _etext
- uint32_t _data
- uint32_t _edata
- uint32_t _bss
- uint32_t _ebss
- uint32_t __init_data

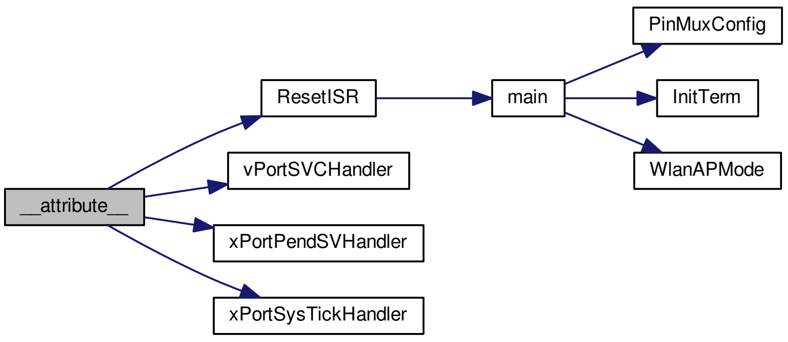
4.4.1 Function Documentation

4.4.1.1 __attribute__ ((section(".intvecs")))

Definition at line 94 of file startup_gcc.c.

References __init_data, _bss, _data, _ebss, _edata, _etext, ResetISR(), vPortSVCHandler(), xPortPendSV↩Handler(), and xPortSysTickHandler().

Here is the call graph for this function:



4.4.1.2 void _c_int00 (void)

4.4.1.3 void∗ _sbrk (unsigned int *incr*)

Definition at line 325 of file startup_gcc.c.

References _eheap, and _heap.

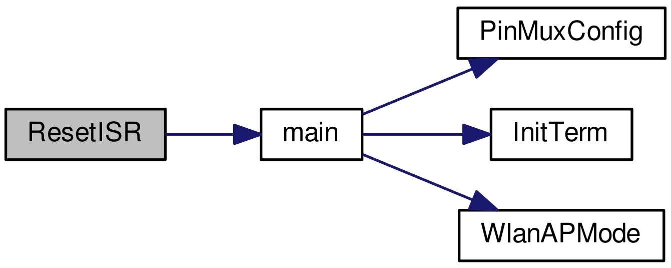
4.4.1.4 void ResetISR (void)

Definition at line 214 of file startup_gcc.c.

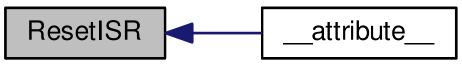
References __init_data, _edata, and main().

Referenced by __attribute__().

Here is the call graph for this function:



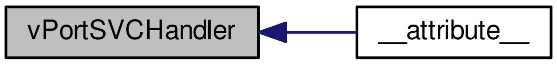
Here is the caller graph for this function:



4.4.1.5 void vPortSVCHandler (void)

Referenced by __attribute__().

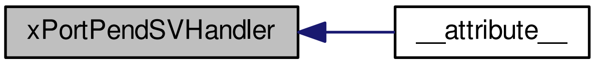
Here is the caller graph for this function:



4.4.1.6 void xPortPendSVHandler (void)

Referenced by __attribute__().

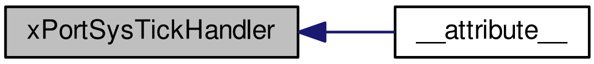
Here is the caller graph for this function:



4.4.1.7 void xPortSysTickHandler (void)

Referenced by __attribute__().

Here is the caller graph for this function:



4.4.2 Variable Documentation

4.4.2.1 uint32_t __init_data

Referenced by __attribute__(), and ResetISR().

4.4.2.2 uint32_t _bss

Referenced by __attribute__().

4.4.2.3 uint32_t _data

Referenced by __attribute__().

4.4.2.4 uint32_t _ebss

Referenced by __attribute__().

4.4.2.5 uint32_t _edata

Referenced by __attribute__(), and ResetISR().

4.4.2.6 unsigned long _eheap

Referenced by _sbrk().

4.4.2.7 uint32_t _etext

Referenced by __attribute__().

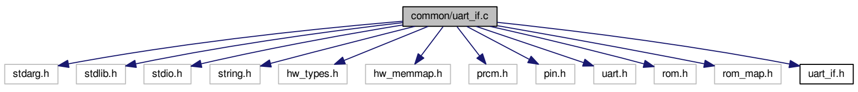
4.4.2.8 unsigned long _heap

Referenced by _sbrk().

4.5 common/uart_if.c File Reference

```
#include <stdarg.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include "hw_types.h"
#include "hw_memmap.h"
#include "prcm.h"
#include "pin.h"
#include "uart.h"
#include "rom.h"
#include "rom_map.h"
#include "uart_if.h"
```

Include dependency graph for uart_if.c:



Macros

- #define IS_SPACE(x) (x == 32 ? 1 : 0)

Functions

- void InitTerm ()
- void Message (const char ∗str)
- void ClearTerm ()
- void Error (char ∗pcFormat,...)
- int GetCmd (char ∗pcBuffer, unsigned int uiBufLen)
- int TrimSpace (char ∗pcInput)
- int Report (const char ∗pcFormat,...)

Variables

- unsigned int ilen =1

4.5.1 Macro Definition Documentation

4.5.1.1 #define IS_SPACE(*x*) (x == 32 ? 1 : 0)

Definition at line 56 of file uart_if.c.

Referenced by TrimSpace().

4.5.2 Function Documentation

4.5.2.1 void ClearTerm (void)

Clear the console window

This function

1. clears the console window.

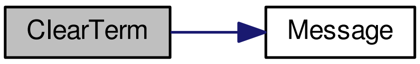
Returns

none

Definition at line 127 of file uart_if.c.

References Message().

Here is the call graph for this function:



4.5.2.2 void Error (char ∗ *pcFormat,* ...)

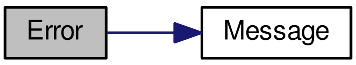
Error Function

Parameters

Definition at line 142 of file uart_if.c.

References Message().

Here is the call graph for this function:



4.5.2.3 int GetCmd (char ∗ *pcBuffer,* unsigned int *uiBufLen*)

Get the Command string from UART

Parameters

pucBuffer	is the command store to which command will be populated
ucBufLen	is the length of buffer store available

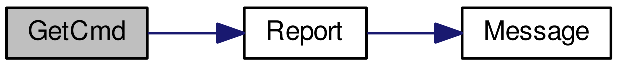
Returns

Length of the bytes received. -1 if buffer length exceeded.

Definition at line 165 of file uart_if.c.

References CONSOLE, and Report().

Here is the call graph for this function:



4.5.2.4 void InitTerm (void)

Initialization

This function

1. Configures the UART to be used.

Returns

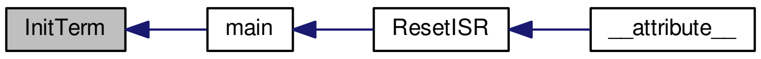
none

Definition at line 80 of file uart_if.c.

References CONSOLE, CONSOLE_PERIPH, and UART_BAUD_RATE.

Referenced by main().

Here is the caller graph for this function:



4.5.2.5 void Message (const char * *str*)

Outputs a character string to the console

Parameters

str	is the pointer to the string to be printed

This function

1. prints the input string character by character on to the console.

Returns

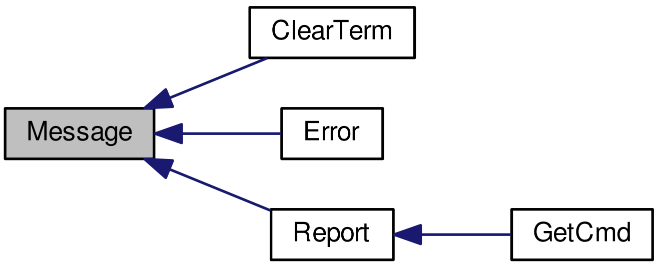
none

Definition at line 103 of file uart_if.c.

References CONSOLE.

Referenced by ClearTerm(), Error(), and Report().

Here is the caller graph for this function:



4.5.2.6 int Report (const char ∗ *pcFormat*, ...)

prints the formatted string on to the console

Parameters

format	is a pointer to the character string specifying the format in the following arguments need to be interpreted.
[variable	number of] arguments according to the format in the first parameters This function 1. prints the formatted error statement.

Returns

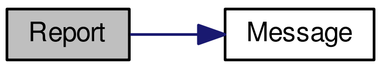
count of characters printed

Definition at line 278 of file uart_if.c.

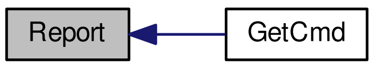
References Message().

Referenced by GetCmd().

Here is the call graph for this function:



Here is the caller graph for this function:



4.5.2.7 int TrimSpace (char ∗ *pcInput*)

Trim the spaces from left and right end of given string

Parameters

Input	string on which trimming happens

Returns

length of trimmed string

Definition at line 239 of file uart_if.c.

References IS_SPACE.

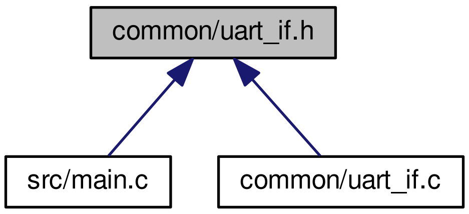
4.5.3 Variable Documentation

4.5.3.1 unsigned int ilen =1

Definition at line 66 of file uart_if.c.

4.6 common/uart_if.h File Reference

This graph shows which files directly or indirectly include this file:



Macros

- #define UART_BAUD_RATE 115200
- #define SYSCLK 80000000
- #define CONSOLE UARTA0_BASE
- #define CONSOLE_PERIPH PRCM_UARTA0
- #define UART_IF_BUFFER 64

Functions

- void DispatcherUARTConfigure (void)
- void DispatcherUartSendPacket (unsigned char ∗inBuff, unsigned short usLength)
- int GetCmd (char ∗pcBuffer, unsigned int uiBufLen)
- void InitTerm (void)
- void ClearTerm (void)
- void Message (const char ∗format)
- void Error (char ∗format,...)
- int TrimSpace (char ∗pcInput)
- int Report (const char ∗format,...)

Variables

- unsigned char g_ucUARTBuffer [ ]

4.6.1 Macro Definition Documentation

4.6.1.1 #define CONSOLE UARTA0_BASE

Definition at line 58 of file uart_if.h.

Referenced by GetCmd(), InitTerm(), and Message().

4.6.1.2 #define CONSOLE_PERIPH PRCM_UARTA0

Definition at line 59 of file uart_if.h.

Referenced by InitTerm().

4.6.1.3 #define SYSCLK 80000000

Definition at line 57 of file uart_if.h.

4.6.1.4 #define UART_BAUD_RATE 115200

Definition at line 56 of file uart_if.h.

Referenced by InitTerm().

4.6.1.5 #define UART_IF_BUFFER 64

Definition at line 63 of file uart_if.h.

4.6.2 Function Documentation

4.6.2.1 void ClearTerm (void)

Clear the console window

This function

1. clears the console window.

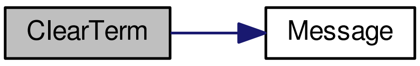
Returns

none

Definition at line 127 of file uart_if.c.

References Message().

Here is the call graph for this function:



4.6.2.2 void DispatcherUARTConfigure (void)

4.6.2.3 void DispatcherUartSendPacket (unsigned char ∗ *inBuff,* unsigned short *usLength*)

4.6.2.4 void Error (char ∗ *pcFormat,* ...)

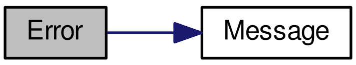
Error Function

Parameters

Definition at line 142 of file uart_if.c.

References Message().

Here is the call graph for this function:



4.6.2.5 int GetCmd (char ∗ *pcBuffer,* unsigned int *uiBufLen*)

Get the Command string from UART

Parameters

pucBuffer	is the command store to which command will be populated
ucBufLen	is the length of buffer store available

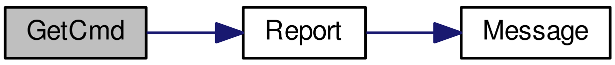
Returns

Length of the bytes received. -1 if buffer length exceeded.

Definition at line 165 of file uart_if.c.

References CONSOLE, and Report().

Here is the call graph for this function:



4.6.2.6 void InitTerm (void)

Initialization

This function

1. Configures the UART to be used.

Returns

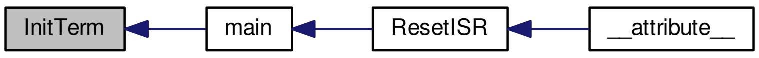
none

Definition at line 80 of file uart_if.c.

References CONSOLE, CONSOLE_PERIPH, and UART_BAUD_RATE.

Referenced by main().

Here is the caller graph for this function:



4.6.2.7 void Message (const char ∗ *str*)

Outputs a character string to the console

Parameters

str	is the pointer to the string to be printed

This function

1. prints the input string character by character on to the console.

Returns

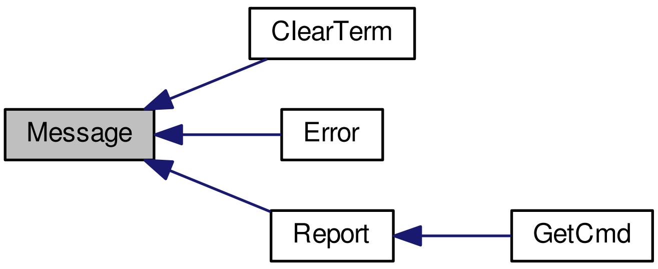
none

Definition at line 103 of file uart_if.c.

References CONSOLE.

Referenced by ClearTerm(), Error(), and Report().

Here is the caller graph for this function:



4.6.2.8 int Report (const char ∗ *pcFormat,* *...*)

prints the formatted string on to the console

Parameters

format	is a pointer to the character string specifying the format in the following arguments need to be interpreted.
[variable	number of] arguments according to the format in the first parameters This function 1. prints the formatted error statement.

Returns

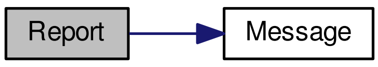
count of characters printed

Definition at line 278 of file uart_if.c.

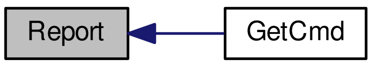
References Message().

Referenced by GetCmd().

Here is the call graph for this function:



Here is the caller graph for this function:



4.6.2.9 int TrimSpace (char ∗ *pcInput*)

Trim the spaces from left and right end of given string

Parameters

Input	string on which trimming happens

Returns

length of trimmed string

Definition at line 239 of file uart_if.c.

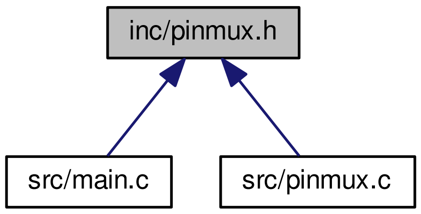
References IS_SPACE.

4.6.3 Variable Documentation

4.6.3.1 unsigned char g_ucUARTBuffer[ ]

4.7 inc/pinmux.h File Reference

This graph shows which files directly or indirectly include this file:



Functions

- void PinMuxConfig (void)

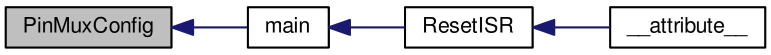
4.7.1 Function Documentation

4.7.1.1 void PinMuxConfig (void)

Definition at line 56 of file pinmux.c.

Referenced by main().

Here is the caller graph for this function:

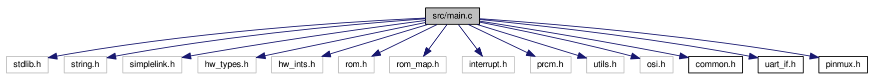


4.8 src/main.c File Reference

```
#include <stdlib.h>
```

```
#include <string.h>
#include "simplelink.h"
#include "hw_types.h"
#include "hw_ints.h"
#include "rom.h"
#include "rom_map.h"
#include "interrupt.h"
#include "prcm.h"
#include "utils.h"
#include "osi.h"
#include "common.h"
#include "uart_if.h"
#include "pinmux.h"
```

Include dependency graph for main.c:



Macros

- #define APP_NAME "WLAN AP"
- #define APPLICATION_VERSION "1.1.0"
- #define OSI_STACK_SIZE 2048
- #define PING_INTERVAL 1000 /∗ In msecs ∗/
- #define PING_TIMEOUT 3000 /∗ In msecs ∗/
- #define PING_PKT_SIZE 20 /∗ In bytes ∗/
- #define NO_OF_ATTEMPTS 3
- #define PING_FLAG 0

Enumerations

- enum e_AppStatusCodes { LAN_CONNECTION_FAILED = -0x7D0, CLIENT_CONNECTION_FAILED = L↩ AN_CONNECTION_FAILED - 1, DEVICE_NOT_IN_STATION_MODE = CLIENT_CONNECTION_FAILED - 1, STATUS_CODE_MAX = -0xBB8 }

Functions

- void SimpleLinkWlanEventHandler (SlWlanEvent_t ∗pSlWlanEvent)
- void SimpleLinkNetAppEventHandler (SlNetAppEvent_t ∗pNetAppEvent)

  This function handles network events such as IP acquisition, IP leased, IP released etc.
- void SimpleLinkHttpServerCallback (SlHttpServerEvent_t ∗pHttpEvent, SlHttpServerResponse_t ∗pHttp↩ Response)

  This function handles HTTP server events.
- void SimpleLinkGeneralEventHandler (SlDeviceEvent_t ∗pDevEvent)

  This function handles General Events.
- void SimpleLinkSockEventHandler (SlSockEvent_t ∗pSock)
- void SimpleLinkPingReport (SlPingReport_t ∗pPingReport)

  This function handles ping report events.
- void WlanAPMode (void ∗pvParameters)

  start simplelink, wait for the sta to connect to the device and run the ping test for that sta
- void main ()

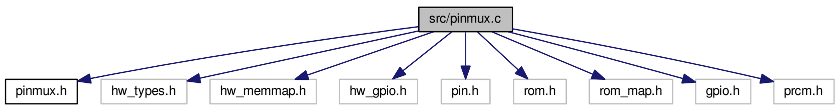
Variables

- unsigned char g_ulStatus = 0
- unsigned long g_ulStaIp = 0
- unsigned long g_ulPingPacketsRecv = 0
- unsigned long g_uiGatewayIP = 0

4.9 src/pinmux.c File Reference

```
#include "pinmux.h"
#include "hw_types.h"
#include "hw_memmap.h"
#include "hw_gpio.h"
#include "pin.h"
#include "rom.h"
#include "rom_map.h"
#include "gpio.h"
#include "prcm.h"
```

Include dependency graph for pinmux.c:



Functions

- void PinMuxConfig (void)

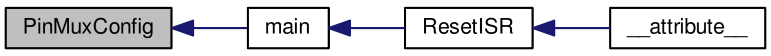
4.9.1 Function Documentation

4.9.1.1 void PinMuxConfig (void)

Definition at line 56 of file pinmux.c.

Referenced by main().

Here is the caller graph for this function:



Index