

wlan\_apdocumentation

Generated by Doxygen 1.8.8

Thu Oct 2 2014 21:16:36



# Contents

<b>1</b>	<b>Module Index</b>	<b>1</b>
1.1	Modules . . . . .	1
<b>2</b>	<b>File Index</b>	<b>3</b>
2.1	File List . . . . .	3
<b>3</b>	<b>Module Documentation</b>	<b>5</b>
3.1	Getting_started_ap . . . . .	5
3.1.1	Detailed Description . . . . .	6
3.1.2	Macro Definition Documentation . . . . .	6
3.1.2.1	APP_NAME . . . . .	6
3.1.2.2	APPLICATION_VERSION . . . . .	6
3.1.2.3	NO_OF_ATTEMPTS . . . . .	6
3.1.2.4	OSI_STACK_SIZE . . . . .	6
3.1.2.5	PING_FLAG . . . . .	6
3.1.2.6	PING_INTERVAL . . . . .	6
3.1.2.7	PING_PKT_SIZE . . . . .	6
3.1.2.8	PING_TIMEOUT . . . . .	6
3.1.3	Enumeration Type Documentation . . . . .	7
3.1.3.1	e_AppStatusCodes . . . . .	7
3.1.4	Function Documentation . . . . .	7
3.1.4.1	main . . . . .	7
3.1.4.2	SimpleLinkGeneralEventHandler . . . . .	8
3.1.4.3	SimpleLinkHttpServerCallback . . . . .	9
3.1.4.4	SimpleLinkNetAppEventHandler . . . . .	9
3.1.4.5	SimpleLinkPingReport . . . . .	9
3.1.4.6	SimpleLinkSockEventHandler . . . . .	10
3.1.4.7	SimpleLinkWlanEventHandler . . . . .	11
3.1.4.8	WlanAPMode . . . . .	11
3.1.5	Variable Documentation . . . . .	12
3.1.5.1	g_uiGatewayIP . . . . .	12
3.1.5.2	g_ulPingPacketsRecv . . . . .	12

3.1.5.3	<a href="#">g_ulStalp</a>	12
3.1.5.4	<a href="#">g_ulStatus</a>	12
<b>4</b>	<b>File Documentation</b>	<b>13</b>
4.1	<a href="#">common/common.h File Reference</a>	13
4.1.1	<a href="#">Macro Definition Documentation</a>	14
4.1.1.1	<a href="#">ASSERT_ON_ERROR</a>	14
4.1.1.2	<a href="#">BSSID_LEN_MAX</a>	14
4.1.1.3	<a href="#">CLR_STATUS_BIT</a>	14
4.1.1.4	<a href="#">CLR_STATUS_BIT_ALL</a>	14
4.1.1.5	<a href="#">CLR_STATUS_BIT_ALL</a>	14
4.1.1.6	<a href="#">DBG_PRINT</a>	15
4.1.1.7	<a href="#">ERR_PRINT</a>	15
4.1.1.8	<a href="#">FAILURE</a>	15
4.1.1.9	<a href="#">GET_STATUS_BIT</a>	15
4.1.1.10	<a href="#">IS_CONNECT_FAILED</a>	15
4.1.1.11	<a href="#">IS_CONNECTED</a>	15
4.1.1.12	<a href="#">IS_IP_ACQUIRED</a>	15
4.1.1.13	<a href="#">IS_IP_LEASED</a>	15
4.1.1.14	<a href="#">IS_NW_PROCSR_ON</a>	16
4.1.1.15	<a href="#">IS_P2P_DEV_FOUND</a>	16
4.1.1.16	<a href="#">IS_P2P_REQ_RCVD</a>	16
4.1.1.17	<a href="#">IS_PING_DONE</a>	16
4.1.1.18	<a href="#">IS_SMART_CFG_START</a>	16
4.1.1.19	<a href="#">LOOP_FOREVER</a>	16
4.1.1.20	<a href="#">SECURITY_KEY</a>	17
4.1.1.21	<a href="#">SECURITY_TYPE</a>	17
4.1.1.22	<a href="#">SET_STATUS_BIT</a>	17
4.1.1.23	<a href="#">SL_STOP_TIMEOUT</a>	17
4.1.1.24	<a href="#">SPAWN_TASK_PRIORITY</a>	17
4.1.1.25	<a href="#">SSID_LEN_MAX</a>	17
4.1.1.26	<a href="#">SSID_NAME</a>	17
4.1.1.27	<a href="#">SUCCESS</a>	17
4.1.1.28	<a href="#">UART_PRINT</a>	17
4.1.1.29	<a href="#">UNUSED</a>	17
4.1.2	<a href="#">Enumeration Type Documentation</a>	18
4.1.2.1	<a href="#">e_StatusBits</a>	18
4.2	<a href="#">common/gpio_if.c File Reference</a>	18
4.2.1	<a href="#">Macro Definition Documentation</a>	19
4.2.1.1	<a href="#">PIN_LED1</a>	19

4.2.1.2	PIN_LED2	19
4.2.1.3	PIN_LED3	19
4.2.2	Function Documentation	19
4.2.2.1	GPIO_IF_ConfigureNIntEnable	19
4.2.2.2	GPIO_IF_Get	20
4.2.2.3	GPIO_IF_GetPortNPin	20
4.2.2.4	GPIO_IF_LedConfigure	21
4.2.2.5	GPIO_IF_LedOff	22
4.2.2.6	GPIO_IF_LedOn	22
4.2.2.7	GPIO_IF_LedStatus	23
4.2.2.8	GPIO_IF_LedToggle	24
4.2.2.9	GPIO_IF_Set	25
4.2.3	Variable Documentation	26
4.2.3.1	g_ucLED1Pin	26
4.2.3.2	g_ucLED2Pin	26
4.2.3.3	g_ucLED3Pin	26
4.2.3.4	g_uiLED1Port	26
4.2.3.5	g_uiLED2Port	27
4.2.3.6	g_uiLED3Port	27
4.3	common/gpio_if.h File Reference	27
4.3.1	Enumeration Type Documentation	28
4.3.1.1	ledEnum	28
4.3.1.2	ledNames	28
4.3.2	Function Documentation	28
4.3.2.1	GPIO_IF_ConfigureNIntEnable	28
4.3.2.2	GPIO_IF_Get	29
4.3.2.3	GPIO_IF_GetPortNPin	29
4.3.2.4	GPIO_IF_LedConfigure	30
4.3.2.5	GPIO_IF_LedOff	31
4.3.2.6	GPIO_IF_LedOn	31
4.3.2.7	GPIO_IF_LedStatus	32
4.3.2.8	GPIO_IF_LedToggle	33
4.3.2.9	GPIO_IF_Set	34
4.4	common/startup_gcc.c File Reference	35
4.4.1	Function Documentation	36
4.4.1.1	__attribute__	36
4.4.1.2	_c_int00	36
4.4.1.3	_sbrk	36
4.4.1.4	ResetISR	37
4.4.1.5	vPortSVCHandler	37

4.4.1.6	xPortPendSVHandler	38
4.4.1.7	xPortSysTickHandler	38
4.4.2	Variable Documentation	38
4.4.2.1	__init_data	38
4.4.2.2	_bss	38
4.4.2.3	_data	38
4.4.2.4	_ebss	38
4.4.2.5	_edata	38
4.4.2.6	_eheap	39
4.4.2.7	_etext	39
4.4.2.8	_heap	39
4.5	common/uart_if.c File Reference	39
4.5.1	Macro Definition Documentation	40
4.5.1.1	IS_SPACE	40
4.5.2	Function Documentation	40
4.5.2.1	ClearTerm	40
4.5.2.2	Error	40
4.5.2.3	GetCmd	41
4.5.2.4	InitTerm	41
4.5.2.5	Message	42
4.5.2.6	Report	43
4.5.2.7	TrimSpace	44
4.5.3	Variable Documentation	44
4.5.3.1	ilen	44
4.6	common/uart_if.h File Reference	45
4.6.1	Macro Definition Documentation	45
4.6.1.1	CONSOLE	45
4.6.1.2	CONSOLE_PERIPH	46
4.6.1.3	SYSCLK	46
4.6.1.4	UART_BAUD_RATE	46
4.6.1.5	UART_IF_BUFFER	46
4.6.2	Function Documentation	46
4.6.2.1	ClearTerm	46
4.6.2.2	DispatcherUARTConfigure	46
4.6.2.3	DispatcherUartSendPacket	46
4.6.2.4	Error	46
4.6.2.5	GetCmd	47
4.6.2.6	InitTerm	47
4.6.2.7	Message	48
4.6.2.8	Report	49

---

4.6.2.9	TrimSpace	50
4.6.3	Variable Documentation	51
4.6.3.1	g_ucUARTBuffer	51
4.7	inc/pinmux.h File Reference	51
4.7.1	Function Documentation	51
4.7.1.1	PinMuxConfig	51
4.8	src/main.c File Reference	51
4.9	src/pinmux.c File Reference	53
4.9.1	Function Documentation	53
4.9.1.1	PinMuxConfig	53
<b>Index</b>		<b>55</b>





# Chapter 1

## Module Index

### 1.1 Modules

Here is a list of all modules:

Getting_started_ap . . . . .	5
------------------------------	---



## Chapter 2

# File Index

### 2.1 File List

Here is a list of all files with brief descriptions:

common/common.h . . . . .	13
common/gpio_if.c . . . . .	18
common/gpio_if.h . . . . .	27
common/startup_gcc.c . . . . .	35
common/uart_if.c . . . . .	39
common/uart_if.h . . . . .	45
inc/pinmux.h . . . . .	51
src/main.c . . . . .	51
src/pinmux.c . . . . .	53



## 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 = LAN_CONNECTION_FAILED - 1, DEVICE_NOT_IN_STATION_MODE = CLIENT_CONNECTION_FAILED - 1, STATUS_CODE_MAX = -0xBB8 }`

#### Functions

- `void SimpleLinkWlanEventHandler (SIWlanEvent_t *pSIWlanEvent)`
- `void SimpleLinkNetAppEventHandler (SINetAppEvent_t *pNetAppEvent)`  
*This function handles network events such as IP acquisition, IP leased, IP released etc.*
- `void SimpleLinkHttpServerCallback (SIHttpServerEvent_t *pHttpEvent, SIHttpServerResponse_t *pHttpResponse)`  
*This function handles HTTP server events.*
- `void SimpleLinkGeneralEventHandler (SIDeviceEvent_t *pDevEvent)`  
*This function handles General Events.*
- `void SimpleLinkSockEventHandler (SISockEvent_t *pSock)`
- `void SimpleLinkPingReport (SIPingReport_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_ulStalp` = 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 msec */`

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 msec */`

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

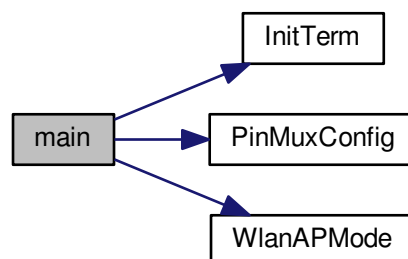
#### 3.1.4.1 void main ( void )

Definition at line 932 of file main.c.

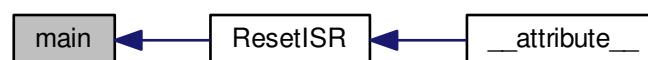
References APP\_NAME, ERR\_PRINT, InitTerm(), LOOP\_FOREVER, OSI\_STACK\_SIZE, PinMuxConfig(), SPA↔WN\_TASK\_PRIORITY, and WlanAPMode().

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 ( SDeviceEvent\_t \* *pDevEvent* )

This function handles General Events.



**Parameters**

in	<i>pDevEvent</i>	- Pointer to General Event Info
----	------------------	---------------------------------

**Returns**

None

Definition at line 395 of file main.c.

References UART\_PRINT.

**3.1.4.3 void SimpleLinkHttpServerCallback ( SIHttpServerEvent\_t \* *pHttpEvent*, SIHttpServerResponse\_t \* *pHttpResponse* )**

This function handles HTTP server events.

**Parameters**

in	<i>pServerEvent</i>	- Contains the relevant event information
in	<i>pServerResponse</i>	- 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 ( SNetAppEvent\_t \* *pNetAppEvent* )**

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

**Parameters**

in	<i>pNetAppEvent</i>	- Pointer to NetApp Event Info
----	---------------------	--------------------------------

**Returns**

None

Definition at line 325 of file main.c.

References CLR\_STATUS\_BIT, g\_ulStalp, g\_ulStatus, SET\_STATUS\_BIT, STATUS\_BIT\_IP\_AQUIRED, STATUS\_BIT\_IP\_LEASED, and UART\_PRINT.

**3.1.4.5 void SimpleLinkPingReport ( SPingReport\_t \* *pPingReport* )**

This function handles ping report events.

**Parameters**

in	<i>pPingReport</i>	- 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 ( SISockEvent\_t \* *pSock* )

This function handles socket events indication

**Parameters**

in	<i>pSock</i>	- Pointer to Socket Event Info
----	--------------	--------------------------------

**Returns**

None

Definition at line 416 of file main.c.

References UART\_PRINT.

**3.1.4.7 void SimpleLinkWlanEventHandler ( SIWlanEvent\_t \* *pSIWlanEvent* )**

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

**Parameters**

<i>pSIWlanEvent</i>	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\_IP\_ACQUIRED, 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**

<i>pvparameters</i>	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\_ulStalp = 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.

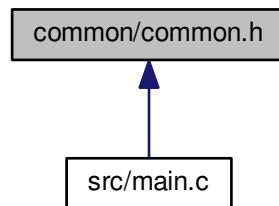
Referenced by SimpleLinkNetAppEventHandler(), SimpleLinkPingReport(), SimpleLinkWlanEventHandler(), and WlanAPMode().

## 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_RECEIVED, 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

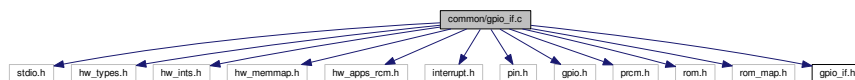
Enumerator

**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 uiIntType, 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

<i>uiGPIOPort</i>	is the GPIO port address
-------------------	--------------------------

<i>ucGPIOPin</i>	is the GPIO pin of the specified port
<i>uiIntType</i>	is the type of the interrupt (refer gpio.h)
<i>pfnIntHandler</i>	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

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

This function

1. Gets a value of the specified GPIO pin

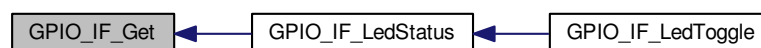
#### Returns

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

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

<i>ucPins</i>	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

<i>ledNum</i>	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\_IF\_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\_IND, 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

<i>ledNum</i>	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_IF_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_IND`, `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**

<i>in</i>	<i>ucGPIONum</i>	is the GPIO to which the LED is connected <code>MCU_GREEN_LED_GPIO</code>
-----------	------------------	---

**Returns**

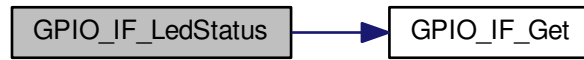
1: LED ON, 0: LED OFF

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

<i>ledNum</i>	is the LED Number
---------------	-------------------

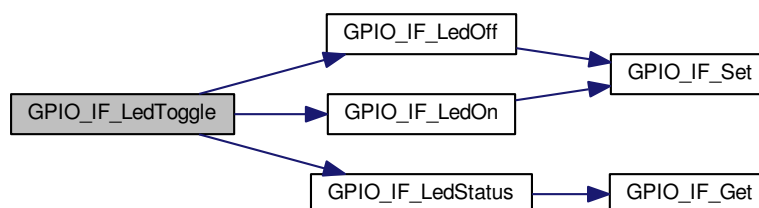
##### Returns

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**

<i>ucPin</i>	is the GPIO pin to be set (0:39)
<i>uiGPIOPort</i>	is the GPIO port address
<i>ucGPIOPin</i>	is the GPIO pin of the specified port
<i>ucGPIOValue</i>	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().

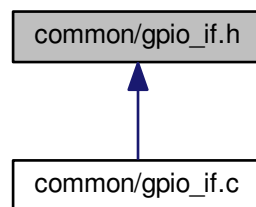
## 4.2.3.6 unsigned int g\_uiLED3Port = 0

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

<i>uiGPIOPort</i>	is the GPIO port address
<i>ucGPIOPin</i>	is the GPIO pin of the specified port
<i>uiIntType</i>	is the type of the interrupt (refer gpio.h)
<i>pfnIntHandler</i>	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

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

This function

1. Gets a value of the specified GPIO pin

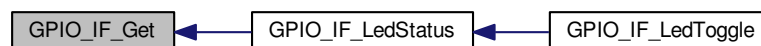
## Returns

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**

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

<i>ucPins</i>	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

<i>ledNum</i>	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\_IF\_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\_IND, 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

<i>ledNum</i>	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\_IF\_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\_IND, 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**

<i>in</i>	<i>ucGPIONum</i>	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

<i>ledNum</i>	is the LED Number
---------------	-------------------

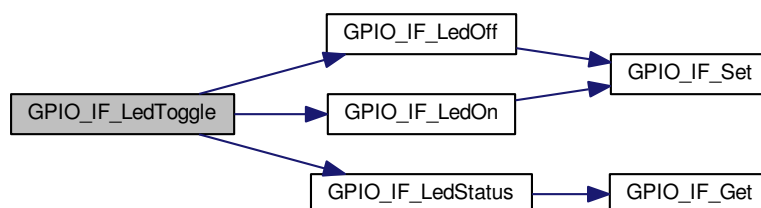
##### Returns

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

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

This function

1. Sets a value to the specified GPIO pin

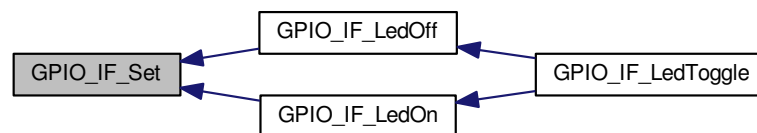
## Returns

None.

Definition at line 416 of file gpio\_if.c.

Referenced by GPIO\_IF\_LedOff(), and GPIO\_IF\_LedOn().

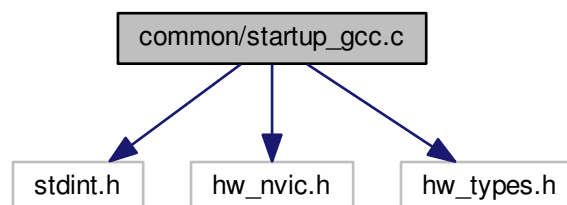
Here is the caller graph for this function:



## 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`

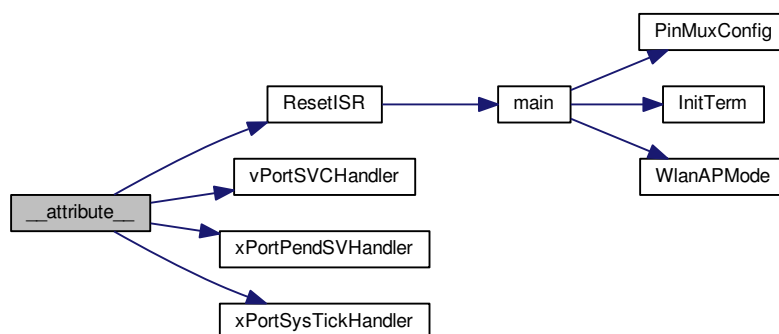
### 4.4.1 Function Documentation

#### 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()`, `xPortPendSVHandler()`, 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`.

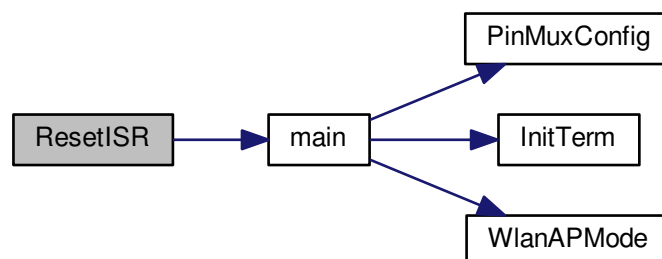
#### 4.4.1.4 void ResetISR ( void )

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

<i>pcBuffer</i>	is the command store to which command will be populated
<i>ucBufLen</i>	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

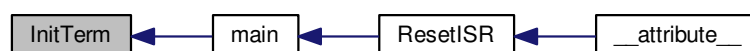
none

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**

<i>str</i>	is the pointer to the string to be printed
------------	--

This function

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

**Returns**

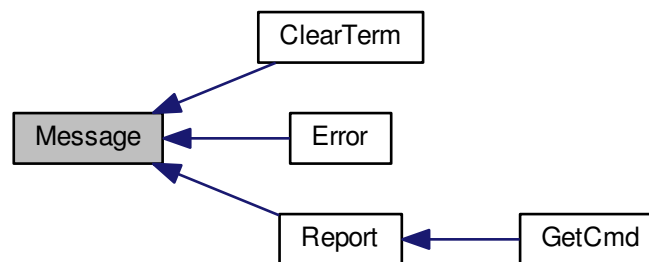
none

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**

<i>format</i>	is a pointer to the character string specifying the format in the following arguments need to be interpreted.
[ <i>variable</i> ]	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

<i>Input</i>	string on which trimming happens
--------------	----------------------------------

##### Returns

length of trimmed string

Definition at line 239 of file uart\_if.c.

References IS\_SPACE.

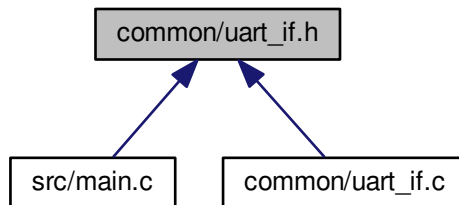
### 4.5.3 Variable Documentation

#### 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 PRM_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

<i>pcBuffer</i>	is the command store to which command will be populated
<i>ucBufLen</i>	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**

<i>str</i>	is the pointer to the string to be printed
------------	--

This function

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

**Returns**

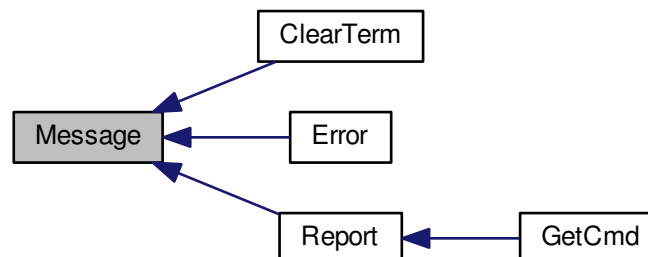
none

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**

<i>format</i>	is a pointer to the character string specifying the format in the following arguments need to be interpreted.
<i>[variable</i>	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 \* *pclnput* )**

Trim the spaces from left and right end of given string

**Parameters**

<i>Input</i>	string on which trimming happens
--------------	----------------------------------

**Returns**

length of trimmed string

Definition at line 239 of file uart\_if.c.

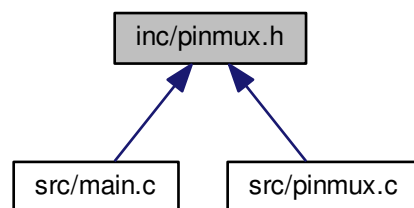
References IS\_SPACE.

### 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

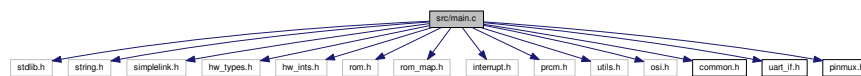
```
#include <stdlib.h>
```

```

#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` = LAN\_CONNECTION\_FAILED - 1, `DEVICE_NOT_IN_STATION_MODE` = CLIENT\_CONNECTION\_FAILED - 1, `STATUS_CODE_MAX` = -0xBB8 }

## Functions

- void `SimpleLinkWlanEventHandler` (SIWlanEvent\_t \*pSIWlanEvent)
- void `SimpleLinkNetAppEventHandler` (SINetAppEvent\_t \*pNetAppEvent)  
*This function handles network events such as IP acquisition, IP leased, IP released etc.*
- void `SimpleLinkHttpServerCallback` (SIHttpServerEvent\_t \*pHttpEvent, SIHttpServerResponse\_t \*pHttpResponse)  
*This function handles HTTP server events.*
- void `SimpleLinkGeneralEventHandler` (SIDeviceEvent\_t \*pDevEvent)  
*This function handles General Events.*
- void `SimpleLinkSockEventHandler` (SISockEvent\_t \*pSock)
- void `SimpleLinkPingReport` (SIPingReport\_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` ()

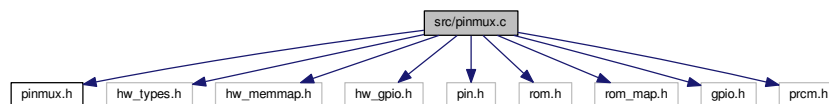
## Variables

- unsigned char `g_ulStatus` = 0
- unsigned long `g_ulStalp` = 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

- `__attribute__`
  - `startup_gcc.c`, 36
- `__init_data`
  - `startup_gcc.c`, 38
- `_bss`
  - `startup_gcc.c`, 38
- `_c_int00`
  - `startup_gcc.c`, 36
- `_data`
  - `startup_gcc.c`, 38
- `_ebss`
  - `startup_gcc.c`, 38
- `_edata`
  - `startup_gcc.c`, 38
- `_eheap`
  - `startup_gcc.c`, 38
- `_etext`
  - `startup_gcc.c`, 39
- `_heap`
  - `startup_gcc.c`, 39
- `_sbrk`
  - `startup_gcc.c`, 36
- APP\_NAME
  - `Getting_started_ap`, 6
- APPLICATION\_VERSION
  - `Getting_started_ap`, 6
- ASSERT\_ON\_ERROR
  - `common.h`, 14
- BSSID\_LEN\_MAX
  - `common.h`, 14
- CLIENT\_CONNECTION\_FAILED
  - `Getting_started_ap`, 7
- CLR\_STATUS\_BIT
  - `common.h`, 14
- CLR\_STATUS\_BIT\_ALL
  - `common.h`, 14
- CONSOLE
  - `uart_if.h`, 45
- CONSOLE\_PERIPH
  - `uart_if.h`, 45
- ClearTerm
  - `uart_if.c`, 40
  - `uart_if.h`, 46
- `common.h`
  - ASSERT\_ON\_ERROR, 14
  - BSSID\_LEN\_MAX, 14
  - CLR\_STATUS\_BIT, 14
  - CLR\_STATUS\_BIT\_ALL, 14
  - DBG\_PRINT, 14
  - e\_StatusBits, 18
  - ERR\_PRINT, 15
  - FAILURE, 15
  - GET\_STATUS\_BIT, 15
  - IS\_CONNECT\_FAILED, 15
  - IS\_CONNECTED, 15
  - IS\_IP\_ACQUIRED, 15
  - IS\_IP\_LEASED, 15
  - IS\_NW\_PROCSR\_ON, 15
  - IS\_P2P\_DEV\_FOUND, 16
  - IS\_P2P\_REQ\_RCVD, 16
  - IS\_PING\_DONE, 16
  - IS\_SMART\_CFG\_START, 16
  - LOOP\_FOREVER, 16
  - SECURITY\_KEY, 16
  - SECURITY\_TYPE, 17
  - SET\_STATUS\_BIT, 17
  - SL\_STOP\_TIMEOUT, 17
  - SPAWN\_TASK\_PRIORITY, 17
  - SSID\_LEN\_MAX, 17
  - SSID\_NAME, 17
  - STATUS\_BIT\_CONNECTION, 18
  - STATUS\_BIT\_CONNECTION\_FAILED, 18
  - STATUS\_BIT\_IP\_AQUIRED, 18
  - STATUS\_BIT\_IP\_LEASED, 18
  - STATUS\_BIT\_NWP\_INIT, 18
  - STATUS\_BIT\_P2P\_DEV\_FOUND, 18
  - STATUS\_BIT\_P2P\_REQ\_RECEIVED, 18
  - STATUS\_BIT\_PING\_DONE, 18
  - STATUS\_BIT\_SMARTCONFIG\_START, 18
  - SUCCESS, 17
  - UART\_PRINT, 17
  - UNUSED, 17
- `common/common.h`, 13
- `common/gpio_if.c`, 18
- `common/gpio_if.h`, 27
- `common/startup_gcc.c`, 35
- `common/uart_if.c`, 39
- `common/uart_if.h`, 45
- DBG\_PRINT
  - `common.h`, 14
- DEVICE\_NOT\_IN\_STATION\_MODE
  - `Getting_started_ap`, 7
- DispatcherUARTConfigure
  - `uart_if.h`, 46
- DispatcherUartSendPacket
  - `uart_if.h`, 46

- e\_AppStatusCodes
  - Getting\_started\_ap, 7
- e\_StatusBits
  - common.h, 18
- ERR\_PRINT
  - common.h, 15
- Error
  - uart\_if.c, 40
  - uart\_if.h, 46
- FAILURE
  - common.h, 15
- g\_ucLED1Pin
  - gpio\_if.c, 26
- g\_ucLED2Pin
  - gpio\_if.c, 26
- g\_ucLED3Pin
  - gpio\_if.c, 26
- g\_ucUARTBuffer
  - uart\_if.h, 51
- g\_uiGatewayIP
  - Getting\_started\_ap, 12
- g\_uiLED1Port
  - gpio\_if.c, 26
- g\_uiLED2Port
  - gpio\_if.c, 26
- g\_uiLED3Port
  - gpio\_if.c, 27
- g\_ulPingPacketsRecv
  - Getting\_started\_ap, 12
- g\_ulStalp
  - Getting\_started\_ap, 12
- g\_ulStatus
  - Getting\_started\_ap, 12
- GET\_STATUS\_BIT
  - common.h, 15
- GPIO\_IF\_ConfigureNIntEnable
  - gpio\_if.c, 19
  - gpio\_if.h, 28
- GPIO\_IF\_Get
  - gpio\_if.c, 20
  - gpio\_if.h, 29
- GPIO\_IF\_GetPortNPin
  - gpio\_if.c, 20
  - gpio\_if.h, 29
- GPIO\_IF\_LedConfigure
  - gpio\_if.c, 21
  - gpio\_if.h, 30
- GPIO\_IF\_LedOff
  - gpio\_if.c, 21
  - gpio\_if.h, 30
- GPIO\_IF\_LedOn
  - gpio\_if.c, 22
  - gpio\_if.h, 31
- GPIO\_IF\_LedStatus
  - gpio\_if.c, 23
  - gpio\_if.h, 32
- GPIO\_IF\_LedToggle
  - gpio\_if.c, 24
  - gpio\_if.h, 33
- GPIO\_IF\_Set
  - gpio\_if.c, 24
  - gpio\_if.h, 33
- GetCmd
  - uart\_if.c, 41
  - uart\_if.h, 47
- Getting\_started\_ap, 5
  - APP\_NAME, 6
  - APPLICATION\_VERSION, 6
  - CLIENT\_CONNECTION\_FAILED, 7
  - DEVICE\_NOT\_IN\_STATION\_MODE, 7
  - e\_AppStatusCodes, 7
  - g\_uiGatewayIP, 12
  - g\_ulPingPacketsRecv, 12
  - g\_ulStalp, 12
  - g\_ulStatus, 12
  - LAN\_CONNECTION\_FAILED, 7
  - main, 7
  - NO\_OF\_ATTEMPTS, 6
  - OSI\_STACK\_SIZE, 6
  - PING\_FLAG, 6
  - PING\_INTERVAL, 6
  - PING\_PKT\_SIZE, 6
  - PING\_TIMEOUT, 6
  - STATUS\_CODE\_MAX, 7
  - SimpleLinkGeneralEventHandler, 7
  - SimpleLinkHttpServerCallback, 9
  - SimpleLinkNetAppEventHandler, 9
  - SimpleLinkPingReport, 9
  - SimpleLinkSockEventHandler, 9
  - SimpleLinkWlanEventHandler, 11
  - WlanAPMode, 11
- gpio\_if.c
  - g\_ucLED1Pin, 26
  - g\_ucLED2Pin, 26
  - g\_ucLED3Pin, 26
  - g\_uiLED1Port, 26
  - g\_uiLED2Port, 26
  - g\_uiLED3Port, 27
  - GPIO\_IF\_ConfigureNIntEnable, 19
  - GPIO\_IF\_Get, 20
  - GPIO\_IF\_GetPortNPin, 20
  - GPIO\_IF\_LedConfigure, 21
  - GPIO\_IF\_LedOff, 21
  - GPIO\_IF\_LedOn, 22
  - GPIO\_IF\_LedStatus, 23
  - GPIO\_IF\_LedToggle, 24
  - GPIO\_IF\_Set, 24
  - PIN\_LED1, 19
  - PIN\_LED2, 19
  - PIN\_LED3, 19
- gpio\_if.h
  - GPIO\_IF\_ConfigureNIntEnable, 28
  - GPIO\_IF\_Get, 29
  - GPIO\_IF\_GetPortNPin, 29
  - GPIO\_IF\_LedConfigure, 30



- GPIO\_IF\_LedOff, [30](#)
- GPIO\_IF\_LedOn, [31](#)
- GPIO\_IF\_LedStatus, [32](#)
- GPIO\_IF\_LedToggle, [33](#)
- GPIO\_IF\_Set, [33](#)
- LED1, [28](#)
- LED2, [28](#)
- LED3, [28](#)
- ledEnum, [28](#)
- ledNames, [28](#)
- MCU\_ALL\_LED\_IND, [28](#)
- MCU\_ASSOCIATED\_IND, [28](#)
- MCU\_CLIENT\_CONNECTED\_IND, [28](#)
- MCU\_EXECUTE\_FAIL\_IND, [28](#)
- MCU\_EXECUTE\_SUCCESS\_IND, [28](#)
- MCU\_GREEN\_LED\_GPIO, [28](#)
- MCU\_IP\_ALLOC\_IND, [28](#)
- MCU\_ON\_IND, [28](#)
- MCU\_ORANGE\_LED\_GPIO, [28](#)
- MCU\_RED\_LED\_GPIO, [28](#)
- MCU\_SENDING\_DATA\_IND, [28](#)
- MCU\_SERVER\_INIT\_IND, [28](#)
- NO\_LED, [28](#)
- NO\_LED\_IND, [28](#)
- IS\_CONNECT\_FAILED
  - common.h, [15](#)
- IS\_CONNECTED
  - common.h, [15](#)
- IS\_IP\_ACQUIRED
  - common.h, [15](#)
- IS\_IP\_LEASED
  - common.h, [15](#)
- IS\_NW\_PROCSR\_ON
  - common.h, [15](#)
- IS\_P2P\_DEV\_FOUND
  - common.h, [16](#)
- IS\_P2P\_REQ\_RCVD
  - common.h, [16](#)
- IS\_PING\_DONE
  - common.h, [16](#)
- IS\_SMART\_CFG\_START
  - common.h, [16](#)
- IS\_SPACE
  - uart\_if.c, [40](#)
- ilen
  - uart\_if.c, [44](#)
- inc/pinmux.h, [51](#)
- InitTerm
  - uart\_if.c, [41](#)
  - uart\_if.h, [47](#)
- LAN\_CONNECTION\_FAILED
  - Getting\_started\_ap, [7](#)
- LED1
  - gpio\_if.h, [28](#)
- LED2
  - gpio\_if.h, [28](#)
- LED3
  - gpio\_if.h, [28](#)
- LOOP\_FOREVER
  - common.h, [16](#)
- ledEnum
  - gpio\_if.h, [28](#)
- ledNames
  - gpio\_if.h, [28](#)
- MCU\_ALL\_LED\_IND
  - gpio\_if.h, [28](#)
- MCU\_ASSOCIATED\_IND
  - gpio\_if.h, [28](#)
- MCU\_CLIENT\_CONNECTED\_IND
  - gpio\_if.h, [28](#)
- MCU\_EXECUTE\_FAIL\_IND
  - gpio\_if.h, [28](#)
- MCU\_EXECUTE\_SUCCESS\_IND
  - gpio\_if.h, [28](#)
- MCU\_GREEN\_LED\_GPIO
  - gpio\_if.h, [28](#)
- MCU\_IP\_ALLOC\_IND
  - gpio\_if.h, [28](#)
- MCU\_ON\_IND
  - gpio\_if.h, [28](#)
- MCU\_ORANGE\_LED\_GPIO
  - gpio\_if.h, [28](#)
- MCU\_RED\_LED\_GPIO
  - gpio\_if.h, [28](#)
- MCU\_SENDING\_DATA\_IND
  - gpio\_if.h, [28](#)
- MCU\_SERVER\_INIT\_IND
  - gpio\_if.h, [28](#)
- main
  - Getting\_started\_ap, [7](#)
- Message
  - uart\_if.c, [41](#)
  - uart\_if.h, [48](#)
- NO\_LED
  - gpio\_if.h, [28](#)
- NO\_LED\_IND
  - gpio\_if.h, [28](#)
- NO\_OF\_ATTEMPTS
  - Getting\_started\_ap, [6](#)
- OSI\_STACK\_SIZE
  - Getting\_started\_ap, [6](#)
- PIN\_LED1
  - gpio\_if.c, [19](#)
- PIN\_LED2
  - gpio\_if.c, [19](#)
- PIN\_LED3
  - gpio\_if.c, [19](#)
- PING\_FLAG
  - Getting\_started\_ap, [6](#)
- PING\_INTERVAL
  - Getting\_started\_ap, [6](#)
- PING\_PKT\_SIZE

- Getting\_started\_ap, 6
- PING\_TIMEOUT
  - Getting\_started\_ap, 6
- PinMuxConfig
  - pinmux.c, 53
  - pinmux.h, 51
- pinmux.c
  - PinMuxConfig, 53
- pinmux.h
  - PinMuxConfig, 51
- Report
  - uart\_if.c, 43
  - uart\_if.h, 48
- ResetISR
  - startup\_gcc.c, 36
- SECURITY\_KEY
  - common.h, 16
- SECURITY\_TYPE
  - common.h, 17
- SET\_STATUS\_BIT
  - common.h, 17
- SL\_STOP\_TIMEOUT
  - common.h, 17
- SPAWN\_TASK\_PRIORITY
  - common.h, 17
- SSID\_LEN\_MAX
  - common.h, 17
- SSID\_NAME
  - common.h, 17
- STATUS\_BIT\_CONNECTION
  - common.h, 18
- STATUS\_BIT\_CONNECTION\_FAILED
  - common.h, 18
- STATUS\_BIT\_IP\_AQUIRED
  - common.h, 18
- STATUS\_BIT\_IP\_LEASED
  - common.h, 18
- STATUS\_BIT\_NWP\_INIT
  - common.h, 18
- STATUS\_BIT\_P2P\_DEV\_FOUND
  - common.h, 18
- STATUS\_BIT\_P2P\_REQ\_RECEIVED
  - common.h, 18
- STATUS\_BIT\_PING\_DONE
  - common.h, 18
- STATUS\_BIT\_SMARTCONFIG\_START
  - common.h, 18
- STATUS\_CODE\_MAX
  - Getting\_started\_ap, 7
- SUCCESS
  - common.h, 17
- SYSCLK
  - uart\_if.h, 46
- SimpleLinkGeneralEventHandler
  - Getting\_started\_ap, 7
- SimpleLinkHttpServerCallback
  - Getting\_started\_ap, 9
- SimpleLinkNetAppEventHandler
  - Getting\_started\_ap, 9
- SimpleLinkPingReport
  - Getting\_started\_ap, 9
- SimpleLinkSockEventHandler
  - Getting\_started\_ap, 9
- SimpleLinkWlanEventHandler
  - Getting\_started\_ap, 11
- src/main.c, 51
- src/pinmux.c, 53
- startup\_gcc.c
  - \_\_attribute\_\_, 36
  - \_\_init\_data, 38
  - \_bss, 38
  - \_c\_int00, 36
  - \_data, 38
  - \_ebss, 38
  - \_edata, 38
  - \_ehheap, 38
  - \_etext, 39
  - \_heap, 39
  - \_sbrk, 36
  - ResetISR, 36
  - vPortSVCHandler, 37
  - xPortPendSVHandler, 37
  - xPortSysTickHandler, 38
- TrimSpace
  - uart\_if.c, 44
  - uart\_if.h, 50
- UART\_BAUD\_RATE
  - uart\_if.h, 46
- UART\_IF\_BUFFER
  - uart\_if.h, 46
- UART\_PRINT
  - common.h, 17
- UNUSED
  - common.h, 17
- uart\_if.c
  - ClearTerm, 40
  - Error, 40
  - GetCmd, 41
  - IS\_SPACE, 40
  - ilen, 44
  - InitTerm, 41
  - Message, 41
  - Report, 43
  - TrimSpace, 44
- uart\_if.h
  - CONSOLE, 45
  - CONSOLE\_PERIPH, 45
  - ClearTerm, 46
  - DispatcherUARTConfigure, 46
  - DispatcherUartSendPacket, 46
  - Error, 46
  - g\_ucUARTBuffer, 51
  - GetCmd, 47
  - InitTerm, 47

- Message, [48](#)
- Report, [48](#)
- SYSCLK, [46](#)
- TrimSpace, [50](#)
- UART\_BAUD\_RATE, [46](#)
- UART\_IF\_BUFFER, [46](#)

vPortSVCHandler

- startup\_gcc.c, [37](#)

WlanAPMode

- Getting\_started\_ap, [11](#)

xPortPendSVHandler

- startup\_gcc.c, [37](#)

xPortSysTickHandler

- startup\_gcc.c, [38](#)