

AC784xx\_DFP I2C

7.1.0

Generated by Doxygen 1.8.13

# Contents

|          |  |          |
|----------|--|----------|
| <b>1</b> | <b>Class Index</b>                                   | <b>1</b> |
| 1.1      | Class List . . . . .                                 | 1        |
| <b>2</b> | <b>File Index</b>                                    | <b>2</b> |
| 2.1      | File List . . . . .                                  | 2        |
| <b>3</b> | <b>Class Documentation</b>                           | <b>3</b> |
| 3.1      | DataTransmitType Struct Reference . . . . .          | 3        |
| 3.1.1    | Detailed Description . . . . .                       | 3        |
| 3.1.2    | Member Data Documentation . . . . .                  | 3        |
| 3.1.2.1  | DataBufferPtr . . . . .                              | 3        |
| 3.1.2.2  | DataLength . . . . .                                 | 4        |
| 3.1.2.3  | DirType . . . . .                                    | 4        |
| 3.1.2.4  | Is10bitAddr . . . . .                                | 4        |
| 3.1.2.5  | SendStop . . . . .                                   | 4        |
| 3.1.2.6  | SlaveAddress . . . . .                               | 4        |
| 3.2      | I2c_Hal_ChannelConfigType Struct Reference . . . . . | 5        |
| 3.2.1    | Detailed Description . . . . .                       | 5        |
| 3.2.2    | Member Data Documentation . . . . .                  | 5        |
| 3.2.2.1  | Callback . . . . .                                   | 5        |
| 3.2.2.2  | I2cMode . . . . .                                    | 5        |
| 3.2.2.3  | I2cType . . . . .                                    | 5        |
| 3.2.2.4  | MasterConfigPtr . . . . .                            | 6        |
| 3.2.2.5  | SlaveConfigPtr . . . . .                             | 6        |
| 3.3      | I2c_Hal_MasterConfigType Struct Reference . . . . .  | 6        |

|          |  |    |
|----------|--|----|
| 3.3.1    | Detailed Description                     | 6  |
| 3.3.2    | Member Data Documentation                | 7  |
| 3.3.2.1  | Arbitration                              | 7  |
| 3.3.2.2  | BaudRate                                 | 7  |
| 3.3.2.3  | PinLow                                   | 7  |
| 3.3.2.4  | RxDmaChannel                             | 7  |
| 3.3.2.5  | SclPin                                   | 7  |
| 3.3.2.6  | SclSdaLowEn                              | 8  |
| 3.3.2.7  | SdaPin                                   | 8  |
| 3.3.2.8  | SendStop                                 | 8  |
| 3.3.2.9  | SlaveAddress                             | 8  |
| 3.3.2.10 | SyncEn                                   | 8  |
| 3.3.2.11 | TimeCfgTypes                             | 9  |
| 3.3.2.12 | TransferType                             | 9  |
| 3.3.2.13 | TxDmaChannel                             | 9  |
| 3.4      | I2c_Hal_MasterStateType Struct Reference | 9  |
| 3.4.1    | Detailed Description                     | 10 |
| 3.4.2    | Member Data Documentation                | 10 |
| 3.4.2.1  | DirType                                  | 10 |
| 3.4.2.2  | Is10bitAddr                              | 10 |
| 3.4.2.3  | RxBuffPtr                                | 10 |
| 3.4.2.4  | RxDmaChannel                             | 10 |
| 3.4.2.5  | RxPointer                                | 11 |
| 3.4.2.6  | RxSize                                   | 11 |
| 3.4.2.7  | SendStop                                 | 11 |
| 3.4.2.8  | SlaveAddress                             | 11 |
| 3.4.2.9  | SourceClock                              | 11 |
| 3.4.2.10 | Status                                   | 12 |
| 3.4.2.11 | TransferType                             | 12 |
| 3.4.2.12 | TxBuffPtr                                | 12 |
| 3.4.2.13 | TxDmaChannel                             | 12 |

|          |  |    |
|----------|--|----|
| 3.4.2.14 | TxSize                                   | 12 |
| 3.5      | I2c_Hal_SlaveConfigType Struct Reference | 13 |
| 3.5.1    | Detailed Description                     | 13 |
| 3.5.2    | Member Data Documentation                | 13 |
| 3.5.2.1  | Is10bitAddr                              | 13 |
| 3.5.2.2  | RxDmaChannel                             | 13 |
| 3.5.2.3  | SlaveAddress                             | 14 |
| 3.5.2.4  | slaveListening                           | 14 |
| 3.5.2.5  | StretchEn                                | 14 |
| 3.5.2.6  | TransferType                             | 14 |
| 3.5.2.7  | TxDmaChannel                             | 14 |
| 3.5.2.8  | WakeupEn                                 | 15 |
| 3.6      | I2c_Hal_SlaveStateType Struct Reference  | 15 |
| 3.6.1    | Detailed Description                     | 15 |
| 3.6.2    | Member Data Documentation                | 15 |
| 3.6.2.1  | RxBuffPtr                                | 15 |
| 3.6.2.2  | RxDmaChannel                             | 16 |
| 3.6.2.3  | RxPointer                                | 16 |
| 3.6.2.4  | RxSize                                   | 16 |
| 3.6.2.5  | SlaveAddress                             | 16 |
| 3.6.2.6  | slaveListening                           | 16 |
| 3.6.2.7  | Status                                   | 17 |
| 3.6.2.8  | TransferType                             | 17 |
| 3.6.2.9  | TxBuffPtr                                | 17 |
| 3.6.2.10 | TxDmaChannel                             | 17 |
| 3.6.2.11 | TxSize                                   | 17 |

|   |           |
|---|-----------|
| <b>4 File Documentation</b>                             | <b>18</b> |
| 4.1 AC784xx_API_Reference_Manual_I2C.pdf File Reference | 18        |
| 4.2 AC784xx_I2c_Reg.h File Reference                    | 18        |
| 4.2.1 Detailed Description                              | 21        |
| 4.2.2 Macro Definition Documentation                    | 21        |
| 4.2.2.1 I2C_HW_DEADLINE_TIMEOUT                         | 21        |
| 4.2.3 Function Documentation                            | 21        |
| 4.2.3.1 I2c_Reg_ClearPltieFlag()                        | 21        |
| 4.2.3.2 I2c_Reg_ClearStartFlag()                        | 22        |
| 4.2.3.3 I2c_Reg_ClearStatus0()                          | 22        |
| 4.2.3.4 I2c_Reg_ClearStatus1()                          | 23        |
| 4.2.3.5 I2c_Reg_ClearStopFlag()                         | 23        |
| 4.2.3.6 I2C_Reg_GetSampleCnt()                          | 24        |
| 4.2.3.7 I2c_Reg_GetStatus0()                            | 24        |
| 4.2.3.8 I2C_Reg_GetStepCnt()                            | 24        |
| 4.2.3.9 I2C_Reg_IsADEXT()                               | 25        |
| 4.2.3.10 I2C_Reg_IsBND()                                | 25        |
| 4.2.3.11 I2C_Reg_IsBNDDMAInterrupt()                    | 26        |
| 4.2.3.12 I2C_Reg_IsBusy()                               | 26        |
| 4.2.3.13 I2C_Reg_IsDMARxEnable()                        | 26        |
| 4.2.3.14 I2c_Reg_IsDMATxEnable()                        | 27        |
| 4.2.3.15 I2C_Reg_IsMaster()                             | 27        |
| 4.2.3.16 I2C_Reg_IsNack()                               | 28        |
| 4.2.3.17 I2c_Reg_IsPltie()                              | 28        |
| 4.2.3.18 I2C_Reg_IsReady()                              | 29        |
| 4.2.3.19 I2c_Reg_IsRxOF()                               | 29        |
| 4.2.3.20 I2C_Reg_IsSMBusAlertResponse()                 | 29        |
| 4.2.3.21 I2c_Reg_IsSSIntEnable()                        | 30        |
| 4.2.3.22 I2c_Reg_IsStart()                              | 30        |
| 4.2.3.23 I2c_Reg_IsStop()                               | 31        |
| 4.2.3.24 I2c_Reg_IsTx()                                 | 31        |

|          |                              |    |
|----------|------------------------------|----|
| 4.2.3.25 | I2c_Reg_IsTxUF()             | 32 |
| 4.2.3.26 | I2c_Reg_ReadDataReg()        | 32 |
| 4.2.3.27 | I2c_Reg_ReceiveLastOneByte() | 33 |
| 4.2.3.28 | I2c_Reg_RxEn()               | 33 |
| 4.2.3.29 | I2c_Reg_SendAck()            | 34 |
| 4.2.3.30 | I2c_Reg_SendNack()           | 34 |
| 4.2.3.31 | I2c_Reg_SendStart()          | 35 |
| 4.2.3.32 | I2c_Reg_SendStop()           | 35 |
| 4.2.3.33 | I2C_Reg_SetADEXT()           | 36 |
| 4.2.3.34 | I2c_Reg_SetARB()             | 36 |
| 4.2.3.35 | I2C_Reg_SetBNDDMAInterrupt() | 37 |
| 4.2.3.36 | I2C_Reg_SetDebug()           | 37 |
| 4.2.3.37 | I2C_Reg_SetDGL()             | 38 |
| 4.2.3.38 | I2C_Reg_SetDGLCnt()          | 38 |
| 4.2.3.39 | I2c_Reg_SetDMARx()           | 39 |
| 4.2.3.40 | I2c_Reg_SetDMATx()           | 39 |
| 4.2.3.41 | I2C_Reg_SetGCA()             | 40 |
| 4.2.3.42 | I2c_Reg_SetInterrupt()       | 40 |
| 4.2.3.43 | I2C_Reg_SetMNT()             | 41 |
| 4.2.3.44 | I2c_Reg_SetModuleEnable()    | 41 |
| 4.2.3.45 | I2c_Reg_SetMSTR()            | 42 |
| 4.2.3.46 | I2c_Reg_SetNackInterrupt()   | 42 |
| 4.2.3.47 | I2c_Reg_SetPltie()           | 43 |
| 4.2.3.48 | I2C_Reg_SetRAD()             | 44 |
| 4.2.3.49 | I2C_Reg_SetRxFInterrupt()    | 44 |
| 4.2.3.50 | I2C_Reg_SetRxOFInterrupt()   | 45 |
| 4.2.3.51 | I2c_Reg_SetSampleStep()      | 45 |
| 4.2.3.52 | I2c_Reg_SetSlaveAddr()       | 46 |
| 4.2.3.53 | I2C_Reg_SetSlaveRangeAddr()  | 46 |
| 4.2.3.54 | I2C_Reg_SetSMBusAlert()      | 47 |
| 4.2.3.55 | I2C_Reg_SetSoftwareReset()   | 47 |

|          |                                |    |
|----------|--------------------------------|----|
| 4.2.3.56 | I2c_Reg_SetSSInterrupt()       | 47 |
| 4.2.3.57 | I2c_Reg_SetStretch()           | 48 |
| 4.2.3.58 | I2c_Reg_SetSYNC()              | 49 |
| 4.2.3.59 | I2C_Reg_SetTxInterrupt()       | 49 |
| 4.2.3.60 | I2C_Reg_SetTxUFInterrupt()     | 50 |
| 4.2.3.61 | I2c_Reg_SetWakeup()            | 50 |
| 4.2.3.62 | I2c_Reg_TransmitOneByte()      | 51 |
| 4.2.3.63 | I2c_Reg_TxEn()                 | 51 |
| 4.2.3.64 | I2c_Reg_WriteDataReg()         | 52 |
| 4.3      | I2c_Hal.c File Reference       | 52 |
| 4.3.1    | Detailed Description           | 53 |
| 4.3.2    | Macro Definition Documentation | 53 |
| 4.3.2.1  | I2C_WAIT_BND                   | 53 |
| 4.3.2.2  | I2C_WAIT_STATUS                | 54 |
| 4.3.3    | Function Documentation         | 54 |
| 4.3.3.1  | I2c_Hal_AbortTransceive()      | 54 |
| 4.3.3.2  | I2c_Hal_AsyncTransceive()      | 54 |
| 4.3.3.3  | I2c_Hal_DeInit()               | 55 |
| 4.3.3.4  | I2c_Hal_GetBase()              | 55 |
| 4.3.3.5  | I2c_Hal_GetStatus()            | 56 |
| 4.3.3.6  | I2c_Hal_Init()                 | 56 |
| 4.3.3.7  | I2c_Hal_MasterGetBaudRate()    | 57 |
| 4.3.3.8  | I2c_Hal_MasterSetBaudRate()    | 57 |
| 4.3.3.9  | I2C_Hal_SlaveGetRxSize()       | 58 |
| 4.3.3.10 | I2C_Hal_SlaveSetRxBuffer()     | 58 |
| 4.3.3.11 | I2C_Hal_SlaveSetTxBuffer()     | 59 |
| 4.3.3.12 | I2c_Hal_SyncTransceive()       | 59 |
| 4.3.3.13 | ISR()                          | 60 |
| 4.4      | I2c_Hal.h File Reference       | 60 |
| 4.4.1    | Detailed Description           | 60 |
| 4.4.2    | Function Documentation         | 61 |

|          |                                |    |
|----------|--------------------------------|----|
| 4.4.2.1  | I2c_Hal_AbortTransceive()      | 61 |
| 4.4.2.2  | I2c_Hal_AsyncTransceive()      | 61 |
| 4.4.2.3  | I2c_Hal_DeInit()               | 62 |
| 4.4.2.4  | I2c_Hal_GetBase()              | 62 |
| 4.4.2.5  | I2c_Hal_GetStatus()            | 63 |
| 4.4.2.6  | I2c_Hal_Init()                 | 63 |
| 4.4.2.7  | I2c_Hal_MasterGetBaudRate()    | 64 |
| 4.4.2.8  | I2c_Hal_MasterSetBaudRate()    | 64 |
| 4.4.2.9  | I2C_Hal_SlaveGetRxSize()       | 65 |
| 4.4.2.10 | I2C_Hal_SlaveSetRxBuffer()     | 65 |
| 4.4.2.11 | I2C_Hal_SlaveSetTxBuffer()     | 66 |
| 4.4.2.12 | I2c_Hal_SyncTransceive()       | 66 |
| 4.5      | I2c_Hal_Types.h File Reference | 67 |
| 4.5.1    | Detailed Description           | 68 |
| 4.5.2    | Macro Definition Documentation | 68 |
| 4.5.2.1  | I2C_ADDEXT_PRIMARY_BYTE_FIX    | 68 |
| 4.5.2.2  | I2C_INSTANCE_ID                | 68 |
| 4.5.3    | Typedef Documentation          | 68 |
| 4.5.3.1  | I2c_Hal_CallbackType           | 69 |
| 4.5.3.2  | I2c_TransmitData               | 69 |
| 4.5.4    | Enumeration Type Documentation | 69 |
| 4.5.4.1  | anonymous enum                 | 69 |
| 4.5.4.2  | anonymous enum                 | 69 |
| 4.5.4.3  | I2c_Hal_ChannelStatusType      | 70 |
| 4.5.4.4  | I2c_Hal_DirType                | 70 |
| 4.5.4.5  | I2c_Hal_MasterStatusType       | 70 |
| 4.5.4.6  | I2c_Hal_SlaveStatusType        | 71 |
| 4.5.4.7  | I2c_Hal_TimeCfgTypes           | 71 |
| 4.5.4.8  | I2c_Hal_TransferType           | 71 |
| 4.5.4.9  | I2c_HwTypes                    | 72 |
| 4.5.4.10 | I2c_ModeTypes                  | 72 |



# Chapter 1

## Class Index

### 1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

|  |    |
|--|----|
| <a href="#">DataTransmitType</a>                             |    |
| <a href="#">DataTransmitType</a> . . . . .                   | 3  |
| <a href="#">I2c_Hal_ChannelConfigType</a>                    |    |
| I2C configuration structure . . . . .                        | 5  |
| <a href="#">I2c_Hal_MasterConfigType</a>                     |    |
| Configuration structure that the user needs to set . . . . . | 6  |
| <a href="#">I2c_Hal_MasterStateType</a>                      |    |
| Master internal context structure . . . . .                  | 9  |
| <a href="#">I2c_Hal_SlaveConfigType</a>                      |    |
| Slave configuration structure . . . . .                      | 13 |
| <a href="#">I2c_Hal_SlaveStateType</a>                       |    |
| Slave internal context structure . . . . .                   | 15 |

# Chapter 2

## File Index

### 2.1 File List

Here is a list of all files with brief descriptions:

|  |    |
|--|----|
| <a href="#">AC784xx_API_Reference_Manual_I2C.pdf</a> | 18 |
| <a href="#">AC784xx_I2c_Reg.h</a>                    |    |
| This file provides Reg i2c api                       | 18 |
| <a href="#">I2c_Hal.c</a>                            |    |
| This file provides I2c hal function                  | 52 |
| <a href="#">I2c_Hal.h</a>                            |    |
| This file provides cdd I2c hal function extern       | 60 |
| <a href="#">I2c_Hal_Types.h</a>                      |    |
| This file provides i2c config                        | 67 |

## Chapter 3

# Class Documentation

### 3.1 DataTransmitType Struct Reference

[DataTransmitType](#).

```
#include <I2c_Hal_Types.h>
```

#### Public Attributes

- uint16 [SlaveAddress](#)
- [I2c\\_Hal\\_DirType](#) [DirType](#)
- boolean [SendStop](#)
- [I2c\\_TransmitData](#) \* [DataBufferPtr](#)
- uint32 [DataLength](#)
- boolean [Is10bitAddr](#)

#### 3.1.1 Detailed Description

[DataTransmitType](#).

Definition at line 222 of file [I2c\\_Hal\\_Types.h](#).

#### 3.1.2 Member Data Documentation

##### 3.1.2.1 DataBufferPtr

[I2c\\_TransmitData](#)\* [DataTransmitType::DataBufferPtr](#)

Definition at line 227 of file [I2c\\_Hal\\_Types.h](#).

### 3.1.2.2 DataLength

```
uint32 DataTransmitType::DataLength
```

Definition at line 228 of file I2c\_Hal\_Types.h.

### 3.1.2.3 DirType

```
I2c_Hal_DirType DataTransmitType::DirType
```

Type of I2C transfer direction

Definition at line 225 of file I2c\_Hal\_Types.h.

### 3.1.2.4 Is10bitAddr

```
boolean DataTransmitType::Is10bitAddr
```

i2c slave addr is 10bit or not

Definition at line 229 of file I2c\_Hal\_Types.h.

### 3.1.2.5 SendStop

```
boolean DataTransmitType::SendStop
```

Specifies if STOP condition must be generated after current transfer

Definition at line 226 of file I2c\_Hal\_Types.h.

### 3.1.2.6 SlaveAddress

```
uint16 DataTransmitType::SlaveAddress
```

Slave address, 7-bit or 10-bit

Definition at line 224 of file I2c\_Hal\_Types.h.

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

- [I2c\\_Hal\\_Types.h](#)

## 3.2 I2c\_Hal\_ChannelConfigType Struct Reference

I2C configuration structure.

```
#include <I2c_Hal_Types.h>
```

### Public Attributes

- [I2c\\_HwTypes](#) I2cType
- [I2c\\_ModeTypes](#) I2cMode
- [I2c\\_Hal\\_CallbackType](#) Callback
- const [I2c\\_Hal\\_MasterConfigType](#) \* MasterConfigPtr
- const [I2c\\_Hal\\_SlaveConfigType](#) \* SlaveConfigPtr

### 3.2.1 Detailed Description

I2C configuration structure.

Definition at line 235 of file I2c\_Hal\_Types.h.

### 3.2.2 Member Data Documentation

#### 3.2.2.1 Callback

[I2c\\_Hal\\_CallbackType](#) I2c\_Hal\_ChannelConfigType::Callback

callback function

Definition at line 239 of file I2c\_Hal\_Types.h.

#### 3.2.2.2 I2cMode

[I2c\\_ModeTypes](#) I2c\_Hal\_ChannelConfigType::I2cMode

Type of I2C mode

Definition at line 238 of file I2c\_Hal\_Types.h.

#### 3.2.2.3 I2cType

[I2c\\_HwTypes](#) I2c\_Hal\_ChannelConfigType::I2cType

Type of I2C mode

Definition at line 237 of file I2c\_Hal\_Types.h.

#### 3.2.2.4 MasterConfigPtr

```
const I2c_Hal_MasterConfigType* I2c_Hal_ChannelConfigType::MasterConfigPtr
```

Master internal context structure

Definition at line 240 of file I2c\_Hal\_Types.h.

#### 3.2.2.5 SlaveConfigPtr

```
const I2c_Hal_SlaveConfigType* I2c_Hal_ChannelConfigType::SlaveConfigPtr
```

Slave internal context structure

Definition at line 241 of file I2c\_Hal\_Types.h.

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

- [I2c\\_Hal\\_Types.h](#)

## 3.3 I2c\_Hal\_MasterConfigType Struct Reference

Configuration structure that the user needs to set.

```
#include <I2c_Hal_Types.h>
```

### Public Attributes

- uint8 [TxDmaChannel](#)
- uint8 [RxDmaChannel](#)
- uint8 [SdaPin](#)
- uint8 [SclPin](#)
- uint16 [SlaveAddress](#)
- uint16 [PinLow](#)
- uint32 [BaudRate](#)
- boolean [SendStop](#)
- boolean [SclSdaLowEn](#)
- boolean [SyncEn](#)
- boolean [Arbitration](#)
- [I2c\\_Hal\\_TransferType](#) [TransferType](#)
- [I2c\\_Hal\\_TimeCfgTypes](#) [TimeCfgTypes](#)

### 3.3.1 Detailed Description

Configuration structure that the user needs to set.

Definition at line 187 of file I2c\_Hal\_Types.h.

### 3.3.2 Member Data Documentation

#### 3.3.2.1 Arbitration

```
boolean I2c_Hal_MasterConfigType::Arbitration
```

Enable/disable arbitration

Definition at line 199 of file I2c\_Hal\_Types.h.

#### 3.3.2.2 BaudRate

```
uint32 I2c_Hal_MasterConfigType::BaudRate
```

Baud rate

Definition at line 195 of file I2c\_Hal\_Types.h.

#### 3.3.2.3 PinLow

```
uint16 I2c_Hal_MasterConfigType::PinLow
```

Pinlow \* Clock

Definition at line 194 of file I2c\_Hal\_Types.h.

#### 3.3.2.4 RxDmaChannel

```
uint8 I2c_Hal_MasterConfigType::RxDmaChannel
```

Rx Channel number for DMA

Definition at line 190 of file I2c\_Hal\_Types.h.

#### 3.3.2.5 SclPin

```
uint8 I2c_Hal_MasterConfigType::SclPin
```

Eio pin to use as I2C SCL pin

Definition at line 192 of file I2c\_Hal\_Types.h.

#### 3.3.2.6 SclSdaLowEn

```
boolean I2c_Hal_MasterConfigType::SclSdaLowEn
```

PltieEn enable

Definition at line 197 of file I2c\_Hal\_Types.h.

#### 3.3.2.7 SdaPin

```
uint8 I2c_Hal_MasterConfigType::SdaPin
```

Eio pin to use as I2C SDA pin

Definition at line 191 of file I2c\_Hal\_Types.h.

#### 3.3.2.8 SendStop

```
boolean I2c_Hal_MasterConfigType::SendStop
```

Specifies if STOP condition must be generated after current transfer

Definition at line 196 of file I2c\_Hal\_Types.h.

#### 3.3.2.9 SlaveAddress

```
uint16 I2c_Hal_MasterConfigType::SlaveAddress
```

Slave address, 7-bit or 10-bit

Definition at line 193 of file I2c\_Hal\_Types.h.

#### 3.3.2.10 SyncEn

```
boolean I2c_Hal_MasterConfigType::SyncEn
```

Enable/disable SCL sync

Definition at line 198 of file I2c\_Hal\_Types.h.



### 3.3.2.11 TimeCfgTypes

[I2c\\_Hal\\_TimeCfgTypes](#) I2c\_Hal\_MasterConfigType::TimeCfgTypes

Type of I2C TimeCfg, SCL or SCL/SDA

Definition at line 201 of file I2c\_Hal\_Types.h.

### 3.3.2.12 TransferType

[I2c\\_Hal\\_TransferType](#) I2c\_Hal\_MasterConfigType::TransferType

Type of I2C transfer, DMA or interrupt

Definition at line 200 of file I2c\_Hal\_Types.h.

### 3.3.2.13 TxDmaChannel

uint8 I2c\_Hal\_MasterConfigType::TxDmaChannel

Tx Channel number for DMA

Definition at line 189 of file I2c\_Hal\_Types.h.

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

- [I2c\\_Hal\\_Types.h](#)

## 3.4 I2c\_Hal\_MasterStateType Struct Reference

Master internal context structure.

### Public Attributes

- uint8 [TxDmaChannel](#)
- uint8 [RxDmaChannel](#)
- uint16 [SlaveAddress](#)
- uint32 [RxSize](#)
- uint32 [TxSize](#)
- uint32 [RxPointer](#)
- uint32 [SourceClock](#)
- uint8 \* [RxBuffPtr](#)
- const uint8 \* [TxBuffPtr](#)
- [I2c\\_Hal\\_DirType](#) DirType
- [I2c\\_Hal\\_TransferType](#) TransferType
- [I2c\\_Hal\\_MasterStatusType](#) Status
- boolean [SendStop](#)
- boolean [Is10bitAddr](#)

### 3.4.1 Detailed Description

Master internal context structure.

Definition at line 76 of file I2c\_Hal.c.

### 3.4.2 Member Data Documentation

#### 3.4.2.1 DirType

```
I2c_Hal_DirType I2c_Hal_MasterStateType::DirType
```

Type of I2C transferdirection

Definition at line 87 of file I2c\_Hal.c.

#### 3.4.2.2 Is10bitAddr

```
boolean I2c_Hal_MasterStateType::Is10bitAddr
```

i2c slave addr is 10bit or not

Definition at line 91 of file I2c\_Hal.c.

#### 3.4.2.3 RxBuffPtr

```
uint8* I2c_Hal_MasterStateType::RxBuffPtr
```

Pointer to receive data buffer

Definition at line 85 of file I2c\_Hal.c.

#### 3.4.2.4 RxDmaChannel

```
uint8 I2c_Hal_MasterStateType::RxDmaChannel
```

Rx Channel number for DMA

Definition at line 79 of file I2c\_Hal.c.

#### 3.4.2.5 RxPointer

```
uint32 I2c_Hal_MasterStateType::RxPointer
```

position of receive data in user defined buffer

Definition at line 83 of file I2c\_Hal.c.

#### 3.4.2.6 RxSize

```
uint32 I2c_Hal_MasterStateType::RxSize
```

Size of receive data buffer

Definition at line 81 of file I2c\_Hal.c.

#### 3.4.2.7 SendStop

```
boolean I2c_Hal_MasterStateType::SendStop
```

Specifies if STOP condition must be generated after current transfer

Definition at line 90 of file I2c\_Hal.c.

#### 3.4.2.8 SlaveAddress

```
uint16 I2c_Hal_MasterStateType::SlaveAddress
```

Slave address

Definition at line 80 of file I2c\_Hal.c.

#### 3.4.2.9 SourceClock

```
uint32 I2c_Hal_MasterStateType::SourceClock
```

I2c master mode source clock

Definition at line 84 of file I2c\_Hal.c.

#### 3.4.2.10 Status

```
I2c_Hal_MasterStatusType I2c_Hal_MasterStateType::Status
```

Status of last driver operation

Definition at line 89 of file I2c\_Hal.c.

#### 3.4.2.11 TransferType

```
I2c_Hal_TransferType I2c_Hal_MasterStateType::TransferType
```

Type of I2C transfer, DMA or interrupt

Definition at line 88 of file I2c\_Hal.c.

#### 3.4.2.12 TxBuffPtr

```
const uint8* I2c_Hal_MasterStateType::TxBuffPtr
```

Pointer to transmit data buffer

Definition at line 86 of file I2c\_Hal.c.

#### 3.4.2.13 TxDmaChannel

```
uint8 I2c_Hal_MasterStateType::TxDmaChannel
```

Tx Channel number for DMA

Definition at line 78 of file I2c\_Hal.c.

#### 3.4.2.14 TxSize

```
uint32 I2c_Hal_MasterStateType::TxSize
```

Size of transmit data buffer

Definition at line 82 of file I2c\_Hal.c.

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

- [I2c\\_Hal.c](#)

## 3.5 I2c\_Hal\_SlaveConfigType Struct Reference

Slave configuration structure.

```
#include <I2c_Hal_Types.h>
```

### Public Attributes

- uint8 [TxDMAChannel](#)
- uint8 [RxDMAChannel](#)
- uint16 [SlaveAddress](#)
- boolean [StretchEn](#)
- boolean [WakeupEn](#)
- boolean [Is10bitAddr](#)
- boolean [slaveListening](#)
- [I2c\\_Hal\\_TransferType](#) [TransferType](#)

### 3.5.1 Detailed Description

Slave configuration structure.

Definition at line 207 of file `I2c_Hal_Types.h`.

### 3.5.2 Member Data Documentation

#### 3.5.2.1 Is10bitAddr

```
boolean I2c_Hal_SlaveConfigType::Is10bitAddr
```

Select 7-bit or 10-bit slave address

Definition at line 214 of file `I2c_Hal_Types.h`.

#### 3.5.2.2 RxDMAChannel

```
uint8 I2c_Hal_SlaveConfigType::RxDMAChannel
```

Rx Channel number for DMA

Definition at line 210 of file `I2c_Hal_Types.h`.

### 3.5.2.3 SlaveAddress

```
uint16 I2c_Hal_SlaveConfigType::SlaveAddress
```

Slave address, 7-bit or 10-bit

Definition at line 211 of file I2c\_Hal\_Types.h.

### 3.5.2.4 slaveListening

```
boolean I2c_Hal_SlaveConfigType::slaveListening
```

Slave mode (always listening or on demand only)

Definition at line 215 of file I2c\_Hal\_Types.h.

### 3.5.2.5 StretchEn

```
boolean I2c_Hal_SlaveConfigType::StretchEn
```

Enable/disable slave SCL stretch

Definition at line 212 of file I2c\_Hal\_Types.h.

### 3.5.2.6 TransferType

```
I2c_Hal_TransferType I2c_Hal_SlaveConfigType::TransferType
```

Type of I2C transfer, DMA or interrupt

Definition at line 216 of file I2c\_Hal\_Types.h.

### 3.5.2.7 TxDmaChannel

```
uint8 I2c_Hal_SlaveConfigType::TxDmaChannel
```

Tx Channel number for DMA

Definition at line 209 of file I2c\_Hal\_Types.h.

### 3.5.2.8 WakeupEn

```
boolean I2c_Hal_SlaveConfigType::WakeupEn
```

Enable/disable slave address match wakeup

Definition at line 213 of file I2c\_Hal\_Types.h.

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

- [I2c\\_Hal\\_Types.h](#)

## 3.6 I2c\_Hal\_SlaveStateType Struct Reference

Slave internal context structure.

### Public Attributes

- boolean [slaveListening](#)
- uint8 [TxDmaChannel](#)
- uint8 [RxDmaChannel](#)
- uint16 [SlaveAddress](#)
- uint32 [RxSize](#)
- uint32 [TxSize](#)
- uint32 [RxPointer](#)
- uint8 \* [RxBuffPtr](#)
- const uint8 \* [TxBuffPtr](#)
- [I2c\\_Hal\\_TransferType](#) [TransferType](#)
- [I2c\\_Hal\\_SlaveStatusType](#) [Status](#)

### 3.6.1 Detailed Description

Slave internal context structure.

Definition at line 97 of file I2c\_Hal.c.

### 3.6.2 Member Data Documentation

#### 3.6.2.1 RxBuffPtr

```
uint8* I2c_Hal_SlaveStateType::RxBuffPtr
```

Pointer to receive data buffer

Definition at line 106 of file I2c\_Hal.c.

### 3.6.2.2 RxDmaChannel

```
uint8 I2c_Hal_SlaveStateType::RxDmaChannel
```

Rx Channel number for DMA

Definition at line 101 of file I2c\_Hal.c.

### 3.6.2.3 RxPointer

```
uint32 I2c_Hal_SlaveStateType::RxPointer
```

position of receive data in user defined buffer

Definition at line 105 of file I2c\_Hal.c.

### 3.6.2.4 RxSize

```
uint32 I2c_Hal_SlaveStateType::RxSize
```

Size of receive data buffer

Definition at line 103 of file I2c\_Hal.c.

### 3.6.2.5 SlaveAddress

```
uint16 I2c_Hal_SlaveStateType::SlaveAddress
```

Slave address, 7-bit or 10-bit

Definition at line 102 of file I2c\_Hal.c.

### 3.6.2.6 slaveListening

```
boolean I2c_Hal_SlaveStateType::slaveListening
```

Slave mode (always listening or on demand only)

Definition at line 99 of file I2c\_Hal.c.



### 3.6.2.7 Status

`I2c_Hal_SlaveStatusType` `I2c_Hal_SlaveStateType::Status`

Status of last driver operation

Definition at line 109 of file `I2c_Hal.c`.

### 3.6.2.8 TransferType

`I2c_Hal_TransferType` `I2c_Hal_SlaveStateType::TransferType`

Type of I2C transfer, DMA or interrupt

Definition at line 108 of file `I2c_Hal.c`.

### 3.6.2.9 TxBuffPtr

`const uint8*` `I2c_Hal_SlaveStateType::TxBuffPtr`

Pointer to transmit data buffer

Definition at line 107 of file `I2c_Hal.c`.

### 3.6.2.10 TxDmaChannel

`uint8` `I2c_Hal_SlaveStateType::TxDmaChannel`

Tx Channel number for DMA

Definition at line 100 of file `I2c_Hal.c`.

### 3.6.2.11 TxSize

`uint32` `I2c_Hal_SlaveStateType::TxSize`

Size of transmit data buffer

Definition at line 104 of file `I2c_Hal.c`.

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

- [I2c\\_Hal.c](#)

## Chapter 4

# File Documentation

### 4.1 AC784xx\_API\_Reference\_Manual\_I2C.pdf File Reference

### 4.2 AC784xx\_I2c\_Reg.h File Reference

This file provides Reg i2c api.

```
#include "I2c_Hal_Types.h"
```

#### Macros

- #define [I2C\\_HW\\_DEADLINE\\_TIMEOUT](#) (0x80000UL)  
*I2C hardware timeout count macro.*

#### Functions

- LOCAL\_INLINE void [I2c\\_Reg\\_SetSYNC](#) (I2C\_Type \*const Base, boolean IsEnable)  
*Enable/disable I2C master synchronization.*
- LOCAL\_INLINE void [I2c\\_Reg\\_SetARB](#) (I2C\_Type \*const Base, boolean IsEnable)  
*Enable/disable I2C master arbitration.*
- LOCAL\_INLINE void [I2c\\_Reg\\_SetMSTR](#) (I2C\_Type \*const Base, [I2c\\_ModeTypes](#) Mode)  
*Set i2c master/slave.*
- LOCAL\_INLINE void [I2c\\_Reg\\_SetModuleEnable](#) (I2C\_Type \*const Base, boolean IsEnable)  
*Set i2c module IsEnable.*
- LOCAL\_INLINE void [I2c\\_Reg\\_SetSlaveAddr](#) (I2C\_Type \*const Base, uint16 SlaveAddr)  
*Set slave address.*
- LOCAL\_INLINE void [I2C\\_Reg\\_SetSlaveRangeAddr](#) (I2C\_Type \*const base, uint8 rangeAddr)  
*Set slave range address.*
- LOCAL\_INLINE void [I2C\\_Reg\\_SetRAD](#) (I2C\_Type \*const base, boolean enable)  
*Enable/disable slave range address.*
- LOCAL\_INLINE void [I2C\\_Reg\\_SetMNT](#) (I2C\_Type \*const base, boolean enable)  
*Enable/disable slave monitor function.*
- LOCAL\_INLINE void [I2C\\_Reg\\_SetGCA](#) (I2C\_Type \*const base, boolean enable)  
*Enable/disable slave general call.*
- LOCAL\_INLINE uint32 [I2C\\_Reg\\_IsADEXT](#) (const I2C\_Type \*const base)

*Get the slave extend address status.*

- LOCAL\_INLINE uint32 [I2C\\_Reg\\_IsBND](#) (const I2C\_Type \*const base)

*Get the BND status.*

- LOCAL\_INLINE uint32 [I2C\\_Reg\\_IsNack](#) (const I2C\_Type \*const base)

*Get the NACK status.*

- LOCAL\_INLINE uint32 [I2C\\_Reg\\_IsBusy](#) (const I2C\_Type \*const base)

*Get the BUS busy status.*

- LOCAL\_INLINE uint32 [I2C\\_Reg\\_IsReady](#) (const I2C\_Type \*const base)

*Get the core ready status.*

- LOCAL\_INLINE uint32 [I2C\\_Reg\\_IsMaster](#) (const I2C\_Type \*const base)

*Get the device role.*

- LOCAL\_INLINE void [I2C\\_Reg\\_SetADEXT](#) (I2C\_Type \*const Base, boolean IsEnable)

*Enable/disable slave address extention.*

- LOCAL\_INLINE void [I2c\\_Reg\\_SetWakeup](#) (I2C\_Type \*const Base, boolean IsEnable)

*Enable/disable I2C wakeup.*

- LOCAL\_INLINE void [I2c\\_Reg\\_SetStretch](#) (I2C\_Type \*const Base, boolean IsEnable)

*Enable/dosable I2C slave stretch.*

- LOCAL\_INLINE void [I2c\\_Reg\\_SetSampleStep](#) (I2C\_Type \*const Base, uint32 SampleCnt, uint32 StepCnt)

*Set i2c SCL sample step cnt.*

- LOCAL\_INLINE void [I2c\\_Reg\\_SetInterrupt](#) (I2C\_Type \*const Base, boolean IsEnable)

*Set i2c interrupt.*

- LOCAL\_INLINE void [I2c\\_Reg\\_ClearStatus0](#) (I2C\_Type \*const Base, uint32 ClearMsk)

*Clear the status1 mask bit.*

- LOCAL\_INLINE void [I2c\\_Reg\\_SetNackInterrupt](#) (I2C\_Type \*const Base, boolean IsEnable)

*Clear the status1 mask bit.*

- LOCAL\_INLINE void [I2c\\_Reg\\_SetDMATx](#) (I2C\_Type \*const Base, boolean IsEnable)

*Enable/disable dma tx.*

- LOCAL\_INLINE void [I2c\\_Reg\\_TxEn](#) (I2C\_Type \*const Base)

*Set transfer tx direction.*

- LOCAL\_INLINE void [I2c\\_Reg\\_WriteDataReg](#) (I2C\_Type \*const Base, uint8 Data)

*Write Data register.*

- LOCAL\_INLINE void [I2c\\_Reg\\_SetDMARx](#) (I2C\_Type \*const Base, boolean IsEnable)

*Enable/disable dma rx.*

- LOCAL\_INLINE void [I2c\\_Reg\\_RxEn](#) (I2C\_Type \*const Base)

*Set transfer rx direction.*

- LOCAL\_INLINE uint8 [I2c\\_Reg\\_ReadDataReg](#) (const I2C\_Type \*Base)

*Read data register.*

- LOCAL\_INLINE void [I2c\\_Reg\\_SendStop](#) (I2C\_Type \*const Base)

*Enable transfer stop.*

- LOCAL\_INLINE void [I2c\\_Reg\\_SetSSInterrupt](#) (I2C\_Type \*Base, boolean IsEnable)

*Set start stop interrupt.*

- LOCAL\_INLINE uint32 [I2c\\_Reg\\_GetStatus0](#) (const I2C\_Type \*const Base)

*Get status0 register.*

- LOCAL\_INLINE uint32 [I2c\\_Reg\\_IsStart](#) (const I2C\_Type \*const Base)

*Get the start flag.*

- LOCAL\_INLINE uint32 [I2c\\_Reg\\_IsSSIntEnable](#) (const I2C\_Type \*const Base)

*Get the start/stop int enable.*

- LOCAL\_INLINE uint32 [I2c\\_Reg\\_IsStop](#) (const I2C\_Type \*const Base)

*Get the stop flag.*

- LOCAL\_INLINE void [I2c\\_Reg\\_ClearStartFlag](#) (I2C\_Type \*const Base)

*Clear the start flag bit.*

- LOCAL\_INLINE void [I2c\\_Reg\\_ClearStopFlag](#) (I2C\_Type \*const Base)

*Clear the stop flag bit.*

- LOCAL\_INLINE uint32 [I2c\\_Reg\\_IsDMATxEnable](#) (const I2C\_Type \*const Base)  
*Get whether the Tx is DMA or not.*
- LOCAL\_INLINE uint32 [I2C\\_Reg\\_IsDMARxEnable](#) (const I2C\_Type \*const base)  
*Get whether the Rx is DMA or not.*
- LOCAL\_INLINE void [I2C\\_Reg\\_SetDebug](#) (I2C\_Type \*const base, boolean enable)  
*Enable I2C debug mode or not.*
- LOCAL\_INLINE void [I2C\\_Reg\\_SetSoftwareReset](#) (I2C\_Type \*const base, boolean enable)  
*Enable I2C software reset of master and slave or not.*
- LOCAL\_INLINE void [I2C\\_Reg\\_SetSMBusAlert](#) (I2C\_Type \*const base, boolean enable)  
*Enable SMBus Alert or not.*
- LOCAL\_INLINE void [I2C\\_Reg\\_SetBNDDMAInterrupt](#) (I2C\_Type \*const base, boolean enable)  
*Set BND\_DMA interrupt.*
- LOCAL\_INLINE uint32 [I2C\\_Reg\\_IsBNDDMAInterrupt](#) (const I2C\_Type \*const base)  
*Get BND\_DMA interrupt status.*
- LOCAL\_INLINE void [I2C\\_Reg\\_SetTxEmptyInterrupt](#) (I2C\_Type \*const base, boolean enable)  
*Enable/disable slave tx buff empty interrupt.*
- LOCAL\_INLINE void [I2C\\_Reg\\_SetRxFInterrupt](#) (I2C\_Type \*const base, boolean enable)  
*Enable/disable slave rx buff full interrupt.*
- LOCAL\_INLINE void [I2C\\_Reg\\_SetTxUFInterrupt](#) (I2C\_Type \*const base, boolean enable)  
*Enable/disable slave tx under flow interrupt.*
- LOCAL\_INLINE void [I2C\\_Reg\\_SetRxOFInterrupt](#) (I2C\_Type \*const base, boolean enable)  
*Enable/disable slave rx over flow interrupt.*
- LOCAL\_INLINE uint32 [I2c\\_Reg\\_IsTx](#) (const I2C\_Type \*const Base)  
*Get the transfer direction.*
- LOCAL\_INLINE void [I2c\\_Reg\\_SendNack](#) (I2C\_Type \*const Base)  
*Set transfer nack.*
- LOCAL\_INLINE void [I2c\\_Reg\\_SendAck](#) (I2C\_Type \*const Base)  
*Set transfer ack.*
- LOCAL\_INLINE void [I2c\\_Reg\\_ReceiveLastOneByte](#) (I2C\_Type \*Base, uint8 \*Data)  
*Read one byte without send next clock for master.*
- LOCAL\_INLINE uint32 [I2c\\_Reg\\_IsTxUF](#) (const I2C\_Type \*const Base)  
*Get slave tx under flow bit.*
- LOCAL\_INLINE void [I2c\\_Reg\\_ClearStatus1](#) (I2C\_Type \*const Base, uint32 ClearMsk)  
*Clear the status2 mask bit.*
- LOCAL\_INLINE uint32 [I2c\\_Reg\\_IsRxOF](#) (const I2C\_Type \*const Base)  
*Get slave rx over flow bit.*
- LOCAL\_INLINE void [I2c\\_Reg\\_SendStart](#) (I2C\_Type \*const Base)  
*Enable transfer start.*
- LOCAL\_INLINE uint8 [I2C\\_Reg\\_GetStepCnt](#) (const I2C\_Type \*const base)  
*Get i2c SCL step cnt.*
- LOCAL\_INLINE uint8 [I2C\\_Reg\\_GetSampleCnt](#) (const I2C\_Type \*const base)  
*Get i2c SCL sample cnt.*
- LOCAL\_INLINE void [I2c\\_Reg\\_SetPltie](#) (I2C\_Type \*const Base, boolean IsEnable, uint16 Value, [I2c\\_Hal\\_TimeCfg](#)↔  
[Types](#) Types)  
*Set Pltie enable.*
- LOCAL\_INLINE uint32 [I2c\\_Reg\\_IsPltie](#) (const I2C\_Type \*const Base)  
*Get the Pltie flag.*
- LOCAL\_INLINE void [I2c\\_Reg\\_ClearPltieFlag](#) (I2C\_Type \*const Base)  
*Clear the Pltie flag bit.*
- LOCAL\_INLINE void [I2c\\_Reg\\_TransmitOneByte](#) (I2C\_Type \*Base, uint8 Data)  
*Write one byte with polling BND flag, make sure to disable interrupt when use the interface.*
- LOCAL\_INLINE void [I2C\\_Reg\\_SetDGLCnt](#) (I2C\_Type \*const base, uint8 DGLCnt)  
*Set deglitch DGL cnt.*

- LOCAL\_INLINE void [I2C\\_Reg\\_SetDGL](#) (I2C\_Type \*const base, boolean enable)  
*Enable/disable deglitch DGL function.*
- LOCAL\_INLINE uint32 [I2C\\_Reg\\_IsSMBusAlertResponse](#) (const I2C\_Type \*const base)  
*Get SMBus Alert Response flag.*

### 4.2.1 Detailed Description

This file provides Reg i2c api.

### 4.2.2 Macro Definition Documentation

#### 4.2.2.1 I2C\_HW\_DEADLINE\_TIMEOUT

```
#define I2C_HW_DEADLINE_TIMEOUT (0x80000UL)
```

I2C hardware timeout count macro.

Definition at line 59 of file AC784xx\_I2c\_Reg.h.

### 4.2.3 Function Documentation

#### 4.2.3.1 I2c\_Reg\_ClearPltieFlag()

```
LOCAL_INLINE void I2c_Reg_ClearPltieFlag (  
    I2C_Type *const Base )
```

Clear the Pltie flag bit.

#### Note

Service ID: NA

#### Parameters

|     |             |                  |
|-----|-------------|------------------|
| in  | <i>Base</i> | I2C base pointer |
| out | <i>None</i> |                  |

#### Returns

void

Definition at line 922 of file AC784xx\_I2c\_Reg.h.

#### 4.2.3.2 I2c\_Reg\_ClearStartFlag()

```
LOCAL_INLINE void I2c_Reg_ClearStartFlag (
    I2C_Type *const Base )
```

Clear the start flag bit.

##### Note

Function ID:DES\_I2C\_API\_267  
Service ID: NA

##### Parameters

|     |             |                  |
|-----|-------------|------------------|
| in  | <i>Base</i> | I2C base pointer |
| out | <i>None</i> |                  |

##### Returns

void

Definition at line 567 of file AC784xx\_I2c\_Reg.h.

#### 4.2.3.3 I2c\_Reg\_ClearStatus0()

```
LOCAL_INLINE void I2c_Reg_ClearStatus0 (
    I2C_Type *const Base,
    uint32 ClearMsk )
```

Clear the status1 mask bit.

##### Note

Function ID:DES\_I2C\_API\_249  
Service ID: NA

##### Parameters

|     |                 |                       |
|-----|-----------------|-----------------------|
| in  | <i>Base</i>     | I2C Base pointer      |
| in  | <i>ClearMsk</i> | the mask bit to clear |
| out | <i>None</i>     |                       |

##### Returns

void

Definition at line 365 of file AC784xx\_I2c\_Reg.h.

#### 4.2.3.4 I2c\_Reg\_ClearStatus1()

```
LOCAL_INLINE void I2c_Reg_ClearStatus1 (
    I2C_Type *const Base,
    uint32 ClearMsk )
```

Clear the status2 mask bit.

##### Note

Function ID:DES\_I2C\_API\_275

Service ID: NA

##### Parameters

|     |                 |                       |
|-----|-----------------|-----------------------|
| in  | <i>Base</i>     | The I2C base pointer  |
| in  | <i>ClearMsk</i> | the mask bit to clear |
| out | <i>None</i>     |                       |

##### Returns

void

Definition at line 826 of file AC784xx\_I2c\_Reg.h.

#### 4.2.3.5 I2c\_Reg\_ClearStopFlag()

```
LOCAL_INLINE void I2c_Reg_ClearStopFlag (
    I2C_Type *const Base )
```

Clear the stop flag bit.

##### Note

Function ID:DES\_I2C\_API\_268

Service ID: NA

##### Parameters

|     |             |                  |
|-----|-------------|------------------|
| in  | <i>Base</i> | I2C base pointer |
| out | <i>None</i> |                  |

##### Returns

void

Definition at line 583 of file AC784xx\_I2c\_Reg.h.

#### 4.2.3.6 I2C\_Reg\_GetSampleCnt()

```
LOCAL_INLINE uint8 I2C_Reg_GetSampleCnt (
    const I2C_Type *const base )
```

Get i2c SCL sample cnt.

##### Parameters

|    |             |                  |
|----|-------------|------------------|
| in | <i>base</i> | I2C base pointer |
|----|-------------|------------------|

##### Returns

Sample cnt

Definition at line 878 of file AC784xx\_I2c\_Reg.h.

#### 4.2.3.7 I2c\_Reg\_GetStatus0()

```
LOCAL_INLINE uint32 I2c_Reg_GetStatus0 (
    const I2C_Type *const Base )
```

Get status0 register.

##### Note

Function ID:DES\_I2C\_API\_263  
Service ID: NA

##### Parameters

|     |             |                  |
|-----|-------------|------------------|
| in  | <i>Base</i> | I2C base pointer |
| out | <i>None</i> |                  |

##### Returns

uint32: status0 register

Definition at line 515 of file AC784xx\_I2c\_Reg.h.

#### 4.2.3.8 I2C\_Reg\_GetStepCnt()

```
LOCAL_INLINE uint8 I2C_Reg_GetStepCnt (
    const I2C_Type *const base )
```

Get i2c SCL step cnt.



**Parameters**

|           |             |                  |
|-----------|-------------|------------------|
| <i>in</i> | <i>base</i> | I2C base pointer |
|-----------|-------------|------------------|

**Returns**

Step cnt

Definition at line 863 of file AC784xx\_I2c\_Reg.h.

**4.2.3.9 I2C\_Reg\_IsADEXT()**

```
LOCAL_INLINE uint32 I2C_Reg_IsADEXT (
    const I2C_Type *const base )
```

Get the slave extend address status.

**Parameters**

|           |             |                  |
|-----------|-------------|------------------|
| <i>in</i> | <i>base</i> | I2C base pointer |
|-----------|-------------|------------------|

**Returns**

slave extend address enabling bit

Definition at line 210 of file AC784xx\_I2c\_Reg.h.

**4.2.3.10 I2C\_Reg\_IsBND()**

```
LOCAL_INLINE uint32 I2C_Reg_IsBND (
    const I2C_Type *const base )
```

Get the BND status.

**Parameters**

|           |             |                  |
|-----------|-------------|------------------|
| <i>in</i> | <i>base</i> | I2C base pointer |
|-----------|-------------|------------------|

**Returns**

BND flag

Definition at line 221 of file AC784xx\_I2c\_Reg.h.

#### 4.2.3.11 I2C\_Reg\_IsBNDDMAInterrupt()

```
LOCAL_INLINE uint32 I2C_Reg_IsBNDDMAInterrupt (
    const I2C_Type *const base )
```

Get BND\_DMA interrupt status.

##### Parameters

|    |               |  |
|----|---------------|--|
| in | <i>base</i>   | I2C base pointer   |
| in | <i>enable</i> | enabling state <ul style="list-style-type: none"><li>• true to enable</li><li>• false to disable</li></ul> |

##### Returns

BND\_DMA interrupt enable status

Definition at line 680 of file AC784xx\_I2c\_Reg.h.

#### 4.2.3.12 I2C\_Reg\_IsBusy()

```
LOCAL_INLINE uint32 I2C_Reg_IsBusy (
    const I2C_Type *const base )
```

Get the BUS busy status.

##### Parameters

|    |             |                  |
|----|-------------|------------------|
| in | <i>base</i> | I2C base pointer |
|----|-------------|------------------|

##### Returns

busy flag

Definition at line 243 of file AC784xx\_I2c\_Reg.h.

#### 4.2.3.13 I2C\_Reg\_IsDMARxEnable()

```
LOCAL_INLINE uint32 I2C_Reg_IsDMARxEnable (
    const I2C_Type *const base )
```

Get whether the Rx is DMA or not.

**Parameters**

|    |             |                  |
|----|-------------|------------------|
| in | <i>base</i> | I2C base pointer |
|----|-------------|------------------|

**Returns**

DMARX status

Definition at line 610 of file AC784xx\_I2c\_Reg.h.

**4.2.3.14 I2c\_Reg\_IsDMATxEnable()**

```
LOCAL_INLINE uint32 I2c_Reg_IsDMATxEnable (  
    const I2C_Type *const Base )
```

Get whether the Tx is DMA or not.

**Note**

Function ID:DES\_I2C\_API\_269

Service ID: NA

**Parameters**

|     |             |                  |
|-----|-------------|------------------|
| in  | <i>Base</i> | I2C base pointer |
| out | <i>None</i> |                  |

**Returns**

uint32: DMATX status

Definition at line 599 of file AC784xx\_I2c\_Reg.h.

**4.2.3.15 I2C\_Reg\_IsMaster()**

```
LOCAL_INLINE uint32 I2C_Reg_IsMaster (  
    const I2C_Type *const base )
```

Get the device role.

**Parameters**

|    |             |                  |
|----|-------------|------------------|
| in | <i>base</i> | I2C base pointer |
|----|-------------|------------------|

**Returns**

whether is master or not

Definition at line 265 of file AC784xx\_I2c\_Reg.h.

**4.2.3.16 I2C\_Reg\_IsNack()**

```
LOCAL_INLINE uint32 I2C_Reg_IsNack (  
    const I2C_Type *const base )
```

Get the NACK status.

**Parameters**

|    |             |                  |
|----|-------------|------------------|
| in | <i>base</i> | I2C base pointer |
|----|-------------|------------------|

**Returns**

NACK flag

Definition at line 232 of file AC784xx\_I2c\_Reg.h.

**4.2.3.17 I2c\_Reg\_IsPltie()**

```
LOCAL_INLINE uint32 I2c_Reg_IsPltie (  
    const I2C_Type *const Base )
```

Get the Pltie flag.

**Note**

Service ID: NA

**Parameters**

|     |             |                  |
|-----|-------------|------------------|
| in  | <i>Base</i> | I2C base pointer |
| out | <i>None</i> |                  |

**Returns**

uint32: the Pltie flag

Definition at line 910 of file AC784xx\_I2c\_Reg.h.

#### 4.2.3.18 I2C\_Reg\_IsReady()

```
LOCAL_INLINE uint32 I2C_Reg_IsReady (
    const I2C_Type *const base )
```

Get the core ready status.

##### Parameters

|    |             |                  |
|----|-------------|------------------|
| in | <i>base</i> | I2C base pointer |
|----|-------------|------------------|

##### Returns

ready flag

Definition at line 254 of file AC784xx\_I2c\_Reg.h.

#### 4.2.3.19 I2c\_Reg\_IsRxOF()

```
LOCAL_INLINE uint32 I2c_Reg_IsRxOF (
    const I2C_Type *const Base )
```

Get slave rx over flow bit.

##### Note

Function ID:DES\_I2C\_API\_276

Service ID: NA

##### Parameters

|     |             |                      |
|-----|-------------|----------------------|
| in  | <i>Base</i> | The I2C base pointer |
| out | <i>None</i> |                      |

##### Returns

uint32: rx over flow bit

Definition at line 839 of file AC784xx\_I2c\_Reg.h.

#### 4.2.3.20 I2C\_Reg\_IsSMBusAlertResponse()

```
LOCAL_INLINE uint32 I2C_Reg_IsSMBusAlertResponse (
    const I2C_Type *const base )
```

Get SMBus Alert Response flag.

**Parameters**

|    |             |                  |
|----|-------------|------------------|
| in | <i>base</i> | I2C base pointer |
|----|-------------|------------------|

**Returns**

the SMBus Alert Response flag

Definition at line 991 of file AC784xx\_I2c\_Reg.h.

**4.2.3.21 I2c\_Reg\_IsSSIntEnable()**

```
LOCAL_INLINE uint32 I2c_Reg_IsSSIntEnable (
    const I2C_Type *const Base )
```

Get the start/stop int enable.

**Note**

Function ID:DES\_I2C\_API\_265  
Service ID: NA

**Parameters**

|     |             |                  |
|-----|-------------|------------------|
| in  | <i>Base</i> | I2C base pointer |
| out | <i>None</i> |                  |

**Returns**

uint32: the start/stop int enable flag

Definition at line 541 of file AC784xx\_I2c\_Reg.h.

**4.2.3.22 I2c\_Reg\_IsStart()**

```
LOCAL_INLINE uint32 I2c_Reg_IsStart (
    const I2C_Type *const Base )
```

Get the start flag.

**Note**

Function ID:DES\_I2C\_API\_264  
Service ID: NA

**Parameters**

|     |             |                  |
|-----|-------------|------------------|
| in  | <i>Base</i> | I2C base pointer |
| out | <i>None</i> |                  |

**Returns**

uint32: the start flag

Definition at line 528 of file AC784xx\_I2c\_Reg.h.

**4.2.3.23 I2c\_Reg\_IsStop()**

```
LOCAL_INLINE uint32 I2c_Reg_IsStop (
    const I2C_Type *const Base )
```

Get the stop flag.

**Note**

Function ID:DES\_I2C\_API\_266  
Service ID: NA

**Parameters**

|     |             |                  |
|-----|-------------|------------------|
| in  | <i>Base</i> | I2C base pointer |
| out | <i>None</i> |                  |

**Returns**

uint32: the stop flag

Definition at line 554 of file AC784xx\_I2c\_Reg.h.

**4.2.3.24 I2c\_Reg\_IsTx()**

```
LOCAL_INLINE uint32 I2c_Reg_IsTx (
    const I2C_Type *const Base )
```

Get the transfer direction.

**Note**

Function ID:DES\_I2C\_API\_270  
Service ID: NA

**Parameters**

|     |             |                  |
|-----|-------------|------------------|
| in  | <i>Base</i> | I2C base pointer |
| out | <i>None</i> |                  |

**Returns**

uint32: transfer direction flag

Definition at line 749 of file AC784xx\_I2c\_Reg.h.

**4.2.3.25 I2c\_Reg\_IsTxUF()**

```
LOCAL_INLINE uint32 I2c_Reg_IsTxUF (
    const I2C_Type *const Base )
```

Get slave tx under flow bit.

**Note**

Function ID:DES\_I2C\_API\_274  
Service ID: NA

**Parameters**

|     |             |                      |
|-----|-------------|----------------------|
| in  | <i>Base</i> | The I2C base pointer |
| out | <i>None</i> |                      |

**Returns**

uint32: tx under flow bit

Definition at line 812 of file AC784xx\_I2c\_Reg.h.

**4.2.3.26 I2c\_Reg\_ReadDataReg()**

```
LOCAL_INLINE uint8 I2c_Reg_ReadDataReg (
    const I2C_Type * Base )
```

Read data register.

**Note**

Function ID:DES\_I2C\_API\_260  
Service ID: NA



**Parameters**

|     |             |                  |
|-----|-------------|------------------|
| in  | <i>Base</i> | I2C Base pointer |
| out | <i>None</i> |                  |

**Returns**

uint8: Read data

Definition at line 469 of file AC784xx\_I2c\_Reg.h.

**4.2.3.27 I2c\_Reg\_ReceiveLastOneByte()**

```
LOCAL_INLINE void I2c_Reg_ReceiveLastOneByte (
    I2C_Type * Base,
    uint8 * Data )
```

Read one byte without send next clock for master.

**Note**

Function ID:DES\_I2C\_API\_273

Service ID: NA

**Parameters**

|     |             |                               |
|-----|-------------|-------------------------------|
| in  | <i>Base</i> | The I2C base pointer          |
| in  | <i>Data</i> | last one byte data to be read |
| out | <i>None</i> |                               |

**Returns**

void

Definition at line 790 of file AC784xx\_I2c\_Reg.h.

**4.2.3.28 I2c\_Reg\_RxEn()**

```
LOCAL_INLINE void I2c_Reg_RxEn (
    I2C_Type *const Base )
```

Set transfer rx direction.

**Note**

Function ID:DES\_I2C\_API\_259

Service ID: NA

**Parameters**

|     |             |                  |
|-----|-------------|------------------|
| in  | <i>Base</i> | I2C Base pointer |
| out | <i>None</i> |                  |

**Returns**

void

Definition at line 456 of file AC784xx\_I2c\_Reg.h.

**4.2.3.29 I2c\_Reg\_SendAck()**

```
LOCAL_INLINE void I2c_Reg_SendAck (  
    I2C_Type *const Base )
```

Set transfer ack.

**Note**

Function ID:DES\_I2C\_API\_272  
Service ID: NA

**Parameters**

|     |             |                  |
|-----|-------------|------------------|
| in  | <i>Base</i> | I2C base pointer |
| out | <i>None</i> |                  |

**Returns**

void

Definition at line 775 of file AC784xx\_I2c\_Reg.h.

**4.2.3.30 I2c\_Reg\_SendNack()**

```
LOCAL_INLINE void I2c_Reg_SendNack (  
    I2C_Type *const Base )
```

Set transfer nack.

**Note**

Function ID:DES\_I2C\_API\_271  
Service ID: NA

**Parameters**

|     |             |                  |
|-----|-------------|------------------|
| in  | <i>Base</i> | I2C base pointer |
| out | <i>None</i> |                  |

**Returns**

void

Definition at line 762 of file AC784xx\_I2c\_Reg.h.

**4.2.3.31 I2c\_Reg\_SendStart()**

```
LOCAL_INLINE void I2c_Reg_SendStart (
    I2C_Type *const Base )
```

Enable transfer start.

**Note**

Function ID:DES\_I2C\_API\_277  
Service ID: NA

**Parameters**

|     |             |                      |
|-----|-------------|----------------------|
| in  | <i>Base</i> | The I2C base pointer |
| out | <i>None</i> |                      |

**Returns**

void

Definition at line 852 of file AC784xx\_I2c\_Reg.h.

**4.2.3.32 I2c\_Reg\_SendStop()**

```
LOCAL_INLINE void I2c_Reg_SendStop (
    I2C_Type *const Base )
```

Enable transfer stop.

**Note**

Function ID:DES\_I2C\_API\_261  
Service ID: NA

**Parameters**

|     |             |                  |
|-----|-------------|------------------|
| in  | <i>Base</i> | I2C Base pointer |
| out | <i>None</i> |                  |

**Returns**

void

Definition at line 482 of file AC784xx\_I2c\_Reg.h.

**4.2.3.33 I2C\_Reg\_SetADEXT()**

```
LOCAL_INLINE void I2C_Reg_SetADEXT (
    I2C_Type *const Base,
    boolean IsEnable )
```

Enable/disable slave address extention.

**Parameters**

|    |                 |  |
|----|-----------------|--|
| in | <i>base</i>     | I2C base pointer   |
| in | <i>IsEnable</i> | enabling state <ul style="list-style-type: none"><li>• true to enable</li><li>• false to disable</li></ul> |

**Returns**

void

Definition at line 278 of file AC784xx\_I2c\_Reg.h.

**4.2.3.34 I2c\_Reg\_SetARB()**

```
LOCAL_INLINE void I2c_Reg_SetARB (
    I2C_Type *const Base,
    boolean IsEnable )
```

Enable/disable I2C master arbitration.

**Note**

Function ID:DES\_I2C\_API\_240  
Service ID: NA

## Parameters

|     |                 |  |
|-----|-----------------|--|
| in  | <i>Base</i>     | I2C Base pointer   |
| in  | <i>IsEnable</i> | enabling state <ul style="list-style-type: none"><li>• true to enable</li><li>• false to disable</li></ul> |
| out | <i>None</i>     |  |

## Returns

void

Definition at line 96 of file AC784xx\_I2c\_Reg.h.

## 4.2.3.35 I2C\_Reg\_SetBNDDMAInterrupt()

```
LOCAL_INLINE void I2C_Reg_SetBNDDMAInterrupt (  
    I2C_Type *const base,  
    boolean enable )
```

Set BND\_DMA interrupt.

## Parameters

|    |               |  |
|----|---------------|--|
| in | <i>base</i>   | I2C base pointer   |
| in | <i>enable</i> | enabling state <ul style="list-style-type: none"><li>• true to enable</li><li>• false to disable</li></ul> |

## Returns

None

Definition at line 666 of file AC784xx\_I2c\_Reg.h.

## 4.2.3.36 I2C\_Reg\_SetDebug()

```
LOCAL_INLINE void I2C_Reg_SetDebug (  
    I2C_Type *const base,  
    boolean enable )
```

Enable I2C debug mode or not.

**Parameters**

|    |               |  |
|----|---------------|--|
| in | <i>base</i>   | I2C base pointer   |
| in | <i>enable</i> | enabling state <ul style="list-style-type: none"><li>• true to enable</li><li>• false to disable</li></ul> |

**Returns**

None

Definition at line 624 of file AC784xx\_I2c\_Reg.h.

**4.2.3.37 I2C\_Reg\_SetDGL()**

```
LOCAL_INLINE void I2C_Reg_SetDGL (
    I2C_Type *const base,
    boolean enable )
```

Enable/disable deglitch DGL function.

**Parameters**

|    |               |  |
|----|---------------|--|
| in | <i>base</i>   | I2C base pointer   |
| in | <i>enable</i> | enabling state <ul style="list-style-type: none"><li>• true to enable</li><li>• false to disable</li></ul> |

**Returns**

None

Definition at line 973 of file AC784xx\_I2c\_Reg.h.

**4.2.3.38 I2C\_Reg\_SetDGLCnt()**

```
LOCAL_INLINE void I2C_Reg_SetDGLCnt (
    I2C_Type *const base,
    uint8 DGLCnt )
```

Set deglitch DGL cnt.

**Parameters**

|    |               |                  |
|----|---------------|------------------|
| in | <i>base</i>   | I2C base pointer |
| in | <i>DGLCnt</i> | deglitch cnt     |

**Returns**

None

Definition at line 955 of file AC784xx\_I2c\_Reg.h.

**4.2.3.39 I2c\_Reg\_SetDMARx()**

```
LOCAL_INLINE void I2c_Reg_SetDMARx (
    I2C_Type *const Base,
    boolean IsEnable )
```

Enable/disable dma rx.

**Note**

Function ID:DES\_I2C\_API\_256

Service ID: NA

**Parameters**

|     |                 |  |
|-----|-----------------|--|
| in  | <i>Base</i>     | I2C Base pointer   |
| in  | <i>IsEnable</i> | enabling state <ul style="list-style-type: none"><li>• true to enable</li><li>• false to disable</li></ul> |
| out | <i>None</i>     |  |

**Returns**

void

Definition at line 442 of file AC784xx\_I2c\_Reg.h.

**4.2.3.40 I2c\_Reg\_SetDMATx()**

```
LOCAL_INLINE void I2c_Reg_SetDMATx (
    I2C_Type *const Base,
    boolean IsEnable )
```

Enable/disable dma tx.

**Note**

Function ID:DES\_I2C\_API\_251

Service ID: NA

**Parameters**

|     |                 |  |
|-----|-----------------|--|
| in  | <i>Base</i>     | I2C Base pointer   |
| in  | <i>IsEnable</i> | enabling state <ul style="list-style-type: none"><li>• true to enable</li><li>• false to disable</li></ul> |
| out | <i>None</i>     |  |

**Returns**

void

Definition at line 398 of file AC784xx\_I2c\_Reg.h.

**4.2.3.41 I2C\_Reg\_SetGCA()**

```
LOCAL_INLINE void I2C_Reg_SetGCA (  
    I2C_Type *const base,  
    boolean enable )
```

Enable/disable slave general call.

**Parameters**

|    |               |  |
|----|---------------|--|
| in | <i>base</i>   | I2C base pointer   |
| in | <i>enable</i> | enabling state <ul style="list-style-type: none"><li>• true to enable</li><li>• false to disable</li></ul> |

**Returns**

None

Definition at line 199 of file AC784xx\_I2c\_Reg.h.

**4.2.3.42 I2c\_Reg\_SetInterrupt()**

```
LOCAL_INLINE void I2c_Reg_SetInterrupt (  
    I2C_Type *const Base,  
    boolean IsEnable )
```

Set i2c interrupt.

**Note**

Function ID:DES\_I2C\_API\_248  
Service ID: NA



**Parameters**

|     |                 |  |
|-----|-----------------|--|
| in  | <i>Base</i>     | I2C Base pointer   |
| in  | <i>IsEnable</i> | enabling state <ul style="list-style-type: none"><li>• true to enable</li><li>• false to disable</li></ul> |
| out | <i>None</i>     |  |

**Returns**

void

Definition at line 350 of file AC784xx\_I2c\_Reg.h.

**4.2.3.43 I2C\_Reg\_SetMNT()**

```
LOCAL_INLINE void I2C_Reg_SetMNT (
    I2C_Type *const base,
    boolean enable )
```

Enable/disable slave monitor function.

**Parameters**

|    |               |  |
|----|---------------|--|
| in | <i>base</i>   | I2C base pointer   |
| in | <i>enable</i> | enabling state <ul style="list-style-type: none"><li>• true to enable</li><li>• false to disable</li></ul> |

**Returns**

None

Definition at line 185 of file AC784xx\_I2c\_Reg.h.

**4.2.3.44 I2c\_Reg\_SetModuleEnable()**

```
LOCAL_INLINE void I2c_Reg_SetModuleEnable (
    I2C_Type *const Base,
    boolean IsEnable )
```

Set i2c module IsEnable.

**Note**

Function ID:DES\_I2C\_API\_242  
Service ID: NA

**Parameters**

|     |                 |  |
|-----|-----------------|--|
| in  | <i>Base</i>     | I2C Base pointer   |
| in  | <i>IsEnable</i> | enabling state <ul style="list-style-type: none"><li>• true to enable</li><li>• false to disable</li></ul> |
| out | <i>None</i>     |  |

**Returns**

void

Definition at line 129 of file AC784xx\_I2c\_Reg.h.

**4.2.3.45 I2c\_Reg\_SetMSTR()**

```
LOCAL_INLINE void I2c_Reg_SetMSTR (
    I2C_Type *const Base,
    I2c_ModeTypes Mode )
```

Set i2c master/slave.

**Note**

Function ID:DES\_I2C\_API\_241  
Service ID: NA

**Parameters**

|     |             |   |
|-----|-------------|---|
| in  | <i>Base</i> | I2C Base pointer  |
| in  | <i>Mode</i> | i2c mode <ul style="list-style-type: none"><li>• I2C_SLAVE</li><li>• I2C_MASTER</li></ul> |
| out | <i>None</i> |   |

**Returns**

void

Definition at line 113 of file AC784xx\_I2c\_Reg.h.

**4.2.3.46 I2c\_Reg\_SetNackInterrupt()**

```
LOCAL_INLINE void I2c_Reg_SetNackInterrupt (
    I2C_Type *const Base,
    boolean IsEnable )
```

Clear the status1 mask bit.

**Note**

Function ID:DES\_I2C\_API\_250  
Service ID: NA

**Parameters**

|     |                 |  |
|-----|-----------------|--|
| in  | <i>Base</i>     | I2C Base pointer   |
| in  | <i>IsEnable</i> | enabling state <ul style="list-style-type: none"><li>• true to enable</li><li>• false to disable</li></ul> |
| out | <i>None</i>     |  |

**Returns**

void

Definition at line 381 of file AC784xx\_I2c\_Reg.h.

**4.2.3.47 I2c\_Reg\_SetPltie()**

```
LOCAL_INLINE void I2c_Reg_SetPltie (
    I2C_Type *const Base,
    boolean IsEnable,
    uint16 Value,
    I2c_Hal_TimeCfgTypes Types )
```

Set Pltie enable.

**Note**

Service ID: NA

**Parameters**

|     |             |                  |
|-----|-------------|------------------|
| in  | <i>Base</i> | I2C Base pointer |
| out | <i>None</i> |                  |

**Returns**

void

Definition at line 894 of file AC784xx\_I2c\_Reg.h.

#### 4.2.3.48 I2C\_Reg\_SetRAD()

```
LOCAL_INLINE void I2C_Reg_SetRAD (
    I2C_Type *const base,
    boolean enable )
```

Enable/disable slave range address.

##### Parameters

|    |               |  |
|----|---------------|--|
| in | <i>base</i>   | I2C base pointer   |
| in | <i>enable</i> | enabling state <ul style="list-style-type: none"><li>• true to enable</li><li>• false to disable</li></ul> |

##### Returns

None

Definition at line 171 of file AC784xx\_I2c\_Reg.h.

#### 4.2.3.49 I2C\_Reg\_SetRxFInterrupt()

```
LOCAL_INLINE void I2C_Reg_SetRxFInterrupt (
    I2C_Type *const base,
    boolean enable )
```

Enable/disable slave rx buff full interrupt.

##### Parameters

|    |               |  |
|----|---------------|--|
| in | <i>base</i>   | I2C base pointer   |
| in | <i>enable</i> | enabling state <ul style="list-style-type: none"><li>• true to enable</li><li>• false to disable</li></ul> |

##### Returns

None

Definition at line 708 of file AC784xx\_I2c\_Reg.h.

#### 4.2.3.50 I2C\_Reg\_SetRxOFInterrupt()

```
LOCAL_INLINE void I2C_Reg_SetRxOFInterrupt (
    I2C_Type *const base,
    boolean enable )
```

Enable/disable slave rx over flow interrupt.

##### Parameters

|    |               |  |
|----|---------------|--|
| in | <i>base</i>   | I2C base pointer   |
| in | <i>enable</i> | enabling state <ul style="list-style-type: none"><li>• true to enable</li><li>• false to disable</li></ul> |

##### Returns

None

Definition at line 736 of file AC784xx\_I2c\_Reg.h.

#### 4.2.3.51 I2c\_Reg\_SetSampleStep()

```
LOCAL_INLINE void I2c_Reg_SetSampleStep (
    I2C_Type *const Base,
    uint32 SampleCnt,
    uint32 StepCnt )
```

Set i2c SCL sample step cnt.

##### Note

Function ID:DES\_I2C\_API\_247  
Service ID: NA

##### Parameters

|     |                  |  |
|-----|------------------|--|
| in  | <i>Base</i>      | I2C Base pointer   |
| in  | <i>SampleCnt</i> | sample cnt   |
| in  | <i>StepCnt</i>   | step cnt <ul style="list-style-type: none"><li>• <math>\text{sample\_width} = (\text{SampleCnt} + 1) * \text{APB period}</math></li><li>• <math>\text{half\_pulse\_width} = (\text{StepCnt} + 1) * \text{sample\_width}</math></li></ul> |
| out | <i>None</i>      |  |

##### Returns

void

Definition at line 330 of file AC784xx\_I2c\_Reg.h.

#### 4.2.3.52 I2c\_Reg\_SetSlaveAddr()

```
LOCAL_INLINE void I2c_Reg_SetSlaveAddr (
    I2C_Type *const Base,
    uint16 SlaveAddr )
```

Set slave address.

##### Note

Function ID:DES\_I2C\_API\_243

Service ID: NA

##### Parameters

|     |                  |                             |
|-----|------------------|-----------------------------|
| in  | <i>Base</i>      | I2C Base pointer            |
| in  | <i>SlaveAddr</i> | slave 7~10bit address value |
| out | <i>None</i>      |                             |

##### Returns

void

Definition at line 144 of file AC784xx\_I2c\_Reg.h.

#### 4.2.3.53 I2C\_Reg\_SetSlaveRangeAddr()

```
LOCAL_INLINE void I2C_Reg_SetSlaveRangeAddr (
    I2C_Type *const base,
    uint8 rangeAddr )
```

Set slave range address.

##### Parameters

|    |                  |                           |
|----|------------------|---------------------------|
| in | <i>base</i>      | I2C base pointer          |
| in | <i>rangeAddr</i> | slave range address value |

##### Returns

None

Definition at line 157 of file AC784xx\_I2c\_Reg.h.

#### 4.2.3.54 I2C\_Reg\_SetSMBusAlert()

```
LOCAL_INLINE void I2C_Reg_SetSMBusAlert (
    I2C_Type *const base,
    boolean enable )
```

Enable SMBus Alert or not.

##### Parameters

|    |               |  |
|----|---------------|--|
| in | <i>base</i>   | I2C base pointer   |
| in | <i>enable</i> | enabling state <ul style="list-style-type: none"><li>• true to enable</li><li>• false to disable</li></ul> |

##### Returns

None

Definition at line 652 of file AC784xx\_I2c\_Reg.h.

#### 4.2.3.55 I2C\_Reg\_SetSoftwareReset()

```
LOCAL_INLINE void I2C_Reg_SetSoftwareReset (
    I2C_Type *const base,
    boolean enable )
```

Enable I2C software reset of master and slave or not.

##### Parameters

|    |               |  |
|----|---------------|--|
| in | <i>base</i>   | I2C base pointer   |
| in | <i>enable</i> | enabling state <ul style="list-style-type: none"><li>• true to enable</li><li>• false to disable</li></ul> |

##### Returns

None

Definition at line 638 of file AC784xx\_I2c\_Reg.h.

#### 4.2.3.56 I2c\_Reg\_SetSSInterrupt()

```
LOCAL_INLINE void I2c_Reg_SetSSInterrupt (
    I2C_Type * Base,
    boolean IsEnable )
```

Set start stop interrupt.

**Note**

Function ID:DES\_I2C\_API\_262

Service ID: NA

**Parameters**

|     |               |  |
|-----|---------------|--|
| in  | <i>Base</i>   | I2C base pointer   |
| in  | <i>Enable</i> | enabling state <ul style="list-style-type: none"><li>• TRUE to enable</li><li>• FALSE to disable</li></ul> |
| out | <i>None</i>   |  |

**Returns**

void

Definition at line 498 of file AC784xx\_I2c\_Reg.h.

**4.2.3.57 I2c\_Reg\_SetStretch()**

```
LOCAL_INLINE void I2c_Reg_SetStretch (
    I2C_Type *const Base,
    boolean IsEnable )
```

Enable/dosable I2C slave stretch.

**Note**

Function ID:DES\_I2C\_API\_245

Service ID: NA

**Parameters**

|     |                 |  |
|-----|-----------------|--|
| in  | <i>Base</i>     | I2C Base pointer   |
| in  | <i>IsEnable</i> | enabling state <ul style="list-style-type: none"><li>• true to enable</li><li>• false to disable</li></ul> |
| out | <i>None</i>     |  |

**Returns**

void

Definition at line 312 of file AC784xx\_I2c\_Reg.h.



#### 4.2.3.58 I2c\_Reg\_SetSYNC()

```
LOCAL_INLINE void I2c_Reg_SetSYNC (
    I2C_Type *const Base,
    boolean IsEnable )
```

Enable/disable I2C master synchronization.

##### Note

Function ID:DES\_I2C\_API\_239  
Service ID: NA

##### Parameters

|     |                 |  |
|-----|-----------------|--|
| in  | <i>Base</i>     | I2C Base pointer   |
| in  | <i>IsEnable</i> | enabling state <ul style="list-style-type: none"><li>• true to enable</li><li>• false to disable</li></ul> |
| out | <i>None</i>     |  |

##### Returns

void

Definition at line 79 of file AC784xx\_I2c\_Reg.h.

#### 4.2.3.59 I2C\_Reg\_SetTxInterrupt()

```
LOCAL_INLINE void I2C_Reg_SetTxInterrupt (
    I2C_Type *const base,
    boolean enable )
```

Enable/disable slave tx buff empty interrupt.

##### Parameters

|    |               |  |
|----|---------------|--|
| in | <i>base</i>   | I2C base pointer   |
| in | <i>enable</i> | enabling state <ul style="list-style-type: none"><li>• true to enable</li><li>• false to disable</li></ul> |

##### Returns

None

Definition at line 694 of file AC784xx\_I2c\_Reg.h.

#### 4.2.3.60 I2C\_Reg\_SetTxUFInterrupt()

```
LOCAL_INLINE void I2C_Reg_SetTxUFInterrupt (
    I2C_Type *const base,
    boolean enable )
```

Enable/disable slave tx under flow interrupt.

##### Parameters

|    |               |  |
|----|---------------|--|
| in | <i>base</i>   | I2C base pointer   |
| in | <i>enable</i> | enabling state <ul style="list-style-type: none"><li>• true to enable</li><li>• false to disable</li></ul> |

##### Returns

None

Definition at line 722 of file AC784xx\_I2c\_Reg.h.

#### 4.2.3.61 I2c\_Reg\_SetWakeup()

```
LOCAL_INLINE void I2c_Reg_SetWakeup (
    I2C_Type *const Base,
    boolean IsEnable )
```

Enable/disable I2C wakeup.

##### Note

Function ID:DES\_I2C\_API\_244  
Service ID: NA

##### Parameters

|     |                 |  |
|-----|-----------------|--|
| in  | <i>Base</i>     | I2C Base pointer   |
| in  | <i>IsEnable</i> | enabling state <ul style="list-style-type: none"><li>• true to enable</li><li>• false to disable</li></ul> |
| out | <i>None</i>     |  |

##### Returns

void

Definition at line 295 of file AC784xx\_I2c\_Reg.h.

#### 4.2.3.62 I2c\_Reg\_TransmitOneByte()

```
LOCAL_INLINE void I2c_Reg_TransmitOneByte (
    I2C_Type * Base,
    uint8 Data )
```

Write one byte with polling BND flag, make sure to disable interrupt when use the interface.

##### Note

Function ID:DES\_I2C\_API\_258  
Service ID: NA

##### Parameters

|     |             |                       |
|-----|-------------|-----------------------|
| in  | <i>Base</i> | I2C Base pointer      |
| in  | <i>Data</i> | the value to DATA_Reg |
| out | <i>None</i> |                       |

##### Returns

Hal\_StatusType: i2c hardware status

Definition at line 938 of file AC784xx\_I2c\_Reg.h.

#### 4.2.3.63 I2c\_Reg\_TxEn()

```
LOCAL_INLINE void I2c_Reg_TxEn (
    I2C_Type *const Base )
```

Set transfer tx direction.

##### Note

Function ID:DES\_I2C\_API\_254  
Service ID: NA

##### Parameters

|     |             |                  |
|-----|-------------|------------------|
| in  | <i>Base</i> | I2C Base pointer |
| out | <i>None</i> |                  |

##### Returns

void

Definition at line 412 of file AC784xx\_I2c\_Reg.h.

## 4.2.3.64 I2c\_Reg\_WriteDataReg()

```
LOCAL_INLINE void I2c_Reg_WriteDataReg (
    I2C_Type *const Base,
    uint8 Data )
```

Write Data register.

## Note

Function ID:DES\_I2C\_API\_255  
Service ID: NA

## Parameters

|     |             |                   |
|-----|-------------|-------------------|
| in  | <i>Base</i> | I2C Base pointer  |
| in  | <i>Data</i> | the Data to write |
| out | <i>None</i> |                   |

## Returns

void

Definition at line 426 of file AC784xx\_I2c\_Reg.h.

## 4.3 I2c\_Hal.c File Reference

This file provides I2c hal function.

```
#include "AC784xx_I2c_Reg.h"
#include "Ckgen_Hal.h"
#include "Dma_Hal.h"
#include "I2c_Hal.h"
#include "Rcm_Hal.h"
#include "Core_Hal.h"
```

## Classes

- struct [I2c\\_Hal\\_MasterStateType](#)  
*Master internal context structure.*
- struct [I2c\\_Hal\\_SlaveStateType](#)  
*Slave internal context structure.*

## Macros

- #define [I2C\\_WAIT\\_BND](#)(CH, STATUS, TIMEOUT)
- #define [I2C\\_WAIT\\_STATUS](#)(CH, STATUS, MASK)

## Functions

- [ISR](#) (I2C0\_IRQHandler)
- void [I2c\\_Hal\\_Init](#) (uint8 Instance, const [I2c\\_Hal\\_ChannelConfigType](#) \*ChannelConfigPtr)  
*Initializes the I2C module.*
- void [I2c\\_Hal\\_DeInit](#) (uint8 Instance)  
*DeInitializes the I2C module.*
- uint32 [I2c\\_Hal\\_MasterGetBaudRate](#) (uint8 Instance)  
*Get the currently configured baud rate.*
- Hal\_StatusType [I2c\\_Hal\\_MasterSetBaudRate](#) (uint8 Instance, uint32 BaudRate)  
*Set the currently configured baud rate.*
- Hal\_StatusType [I2c\\_Hal\\_SyncTransceive](#) (uint8 Instance, const [DataTransmitType](#) \*DataTransmitPtr)  
*Starts an synchronous transmission on the I2C bus.*
- Hal\_StatusType [I2c\\_Hal\\_AsyncTransceive](#) (uint8 Instance, const [DataTransmitType](#) \*DataTransmitPtr)  
*Starts an asynchronous transmission on the I2C bus.*
- Hal\_StatusType [I2c\\_Hal\\_AbortTransceive](#) (uint8 Instance)  
*Stop an asynchronous transmission on the I2C bus.*
- [I2c\\_Hal\\_ChannelStatusType](#) [I2c\\_Hal\\_GetStatus](#) (uint8 Instance)  
*Gets the status of an I2C channel.*
- I2C\_Type \* [I2c\\_Hal\\_GetBase](#) (uint8 Instance)  
*Get the I2C base.*
- Hal\_StatusType [I2C\\_Hal\\_SlaveSetTxBuffer](#) (uint8 Instance, const uint8 \*TxBuff, uint32 TxSize)  
*Provide a buffer for transmitting data.*
- Hal\_StatusType [I2C\\_Hal\\_SlaveSetRxBuffer](#) (uint8 Instance, uint8 \*RxBuff, uint32 RxSize)  
*Provide a buffer for receiving data.*
- uint32 [I2C\\_Hal\\_SlaveGetRxSize](#) (uint8 Instance)  
*Get the listening receiving data.*

### 4.3.1 Detailed Description

This file provides I2c hal function.

### 4.3.2 Macro Definition Documentation

#### 4.3.2.1 I2C\_WAIT\_BND

```
#define I2C_WAIT_BND(  
    CH,  
    STATUS,  
    TIMEOUT )
```

#### Value:

```
{ \
    TIMEOUT = 0U; \
    STATUS = I2c_Reg_GetStatus0(CH); \
    while ((0U == (STATUS & I2C_STATUS0_BND_Msk)) && (TIMEOUT < \
        I2C_HW_DEADLINE_TIMEOUT)) \
    { \
        STATUS = I2c_Reg_GetStatus0(CH); \
        TIMEOUT++; \
    } \
}
```

Definition at line 52 of file I2c\_Hal.c.

### 4.3.2.2 I2C\_WAIT\_STATUS

```
#define I2C_WAIT_STATUS (
    CH,
    STATUS,
    MASK )
```

#### Value:

```
{ \
    STATUS = I2c_Reg_GetStatus0(CH); \
    while (0U == (STATUS & MASK)) \
    { \
        STATUS = I2c_Reg_GetStatus0(CH); \
    } \
}
```

Definition at line 63 of file I2c\_Hal.c.

## 4.3.3 Function Documentation

### 4.3.3.1 I2c\_Hal\_AbortTransceive()

```
Hal_StatusType I2c_Hal_AbortTransceive (
    uint8 Instance )
```

Stop an asynchronous transmission on the I2C bus.

Starts an asynchronous transmission on the I2C bus.

#### Note

Function ID:DES\_I2C\_API\_242  
Service ID: NA

#### Parameters

|    |                 |                           |
|----|-----------------|---------------------------|
| in | <i>Instance</i> | : I2C hardware channel ID |
|----|-----------------|---------------------------|

#### Returns

Hal\_StatusType: STATUS\_SUCCESS: Success to Send or receive STATUS\_ERROR: Otherwise

Definition at line 2603 of file I2c\_Hal.c.

### 4.3.3.2 I2c\_Hal\_AsyncTransceive()

```
Hal_StatusType I2c_Hal_AsyncTransceive (
    uint8 Instance,
    const DataTransmitType * DataTransmitPtr )
```

Starts an asynchronous transmission on the I2C bus.

**Note**

Function ID:DES\_I2C\_API\_202  
Service ID: NA

**Parameters**

|                |                        |                           |
|----------------|------------------------|---------------------------|
| <i>in</i>      | <i>Instance</i>        | : I2C hardware channel ID |
| <i>in, out</i> | <i>DataTransmitPtr</i> | transmit data used        |

**Returns**

Hal\_StatusType: STATUS\_SUCCESS: Success to Send or receive STATUS\_ERROR: Otherwise

Definition at line 2546 of file I2c\_Hal.c.

**4.3.3.3 I2c\_Hal\_DeInit()**

```
void I2c_Hal_DeInit (
    uint8 Instance )
```

Deinitializes the I2C module.

**Note**

Function ID:DES\_I2C\_API\_201  
Service ID: NA

**Parameters**

|           |                 |                           |
|-----------|-----------------|---------------------------|
| <i>in</i> | <i>Instance</i> | : I2C hardware channel ID |
|-----------|-----------------|---------------------------|

**Returns**

void

Definition at line 2305 of file I2c\_Hal.c.

**4.3.3.4 I2c\_Hal\_GetBase()**

```
I2C_Type* I2c_Hal_GetBase (
    uint8 Instance )
```

Get the I2C base.

**Note**

Function ID:DES\_I2C\_API\_243

**Parameters**

|    |                 |                            |
|----|-----------------|----------------------------|
| in | <i>Instance</i> | : I2C hardware channel ID. |
|----|-----------------|----------------------------|

**Returns**

I2C\_Type\*: the I2C base addr.

Definition at line 2730 of file I2c\_Hal.c.

**4.3.3.5 I2c\_Hal\_GetStatus()**

```
I2c_Hal_ChannelStatusType I2c_Hal_GetStatus (
    uint8 Instance )
```

Gets the status of an I2C channel.

**Note**

Function ID:DES\_I2C\_API\_203

Service ID: NA

**Parameters**

|    |                 |                           |
|----|-----------------|---------------------------|
| in | <i>Instance</i> | : I2C hardware channel ID |
|----|-----------------|---------------------------|

**Returns**

I2c\_ChannelStatusType: I2c channel currenr status

Definition at line 2639 of file I2c\_Hal.c.

**4.3.3.6 I2c\_Hal\_Init()**

```
void I2c_Hal_Init (
    uint8 Instance,
    const I2c_Hal_ChannelConfigType * ChannelConfigPtr )
```

Initializes the I2C module.

**Note**

Function ID:DES\_I2C\_API\_200

Service ID: NA



**Parameters**

|    |                         |                                    |
|----|-------------------------|------------------------------------|
| in | <i>Instance</i>         | : I2C hardware channel ID          |
| in | <i>ChannelConfigPtr</i> | : Pointer to I2c_ChannelConfigType |

**Returns**

Hal\_StatusType: STATUS\_SUCCESS Initializes Success STATUS\_ERROR Initializes Failed

Definition at line 2270 of file I2c\_Hal.c.

**4.3.3.7 I2c\_Hal\_MasterGetBaudRate()**

```
uint32 I2c_Hal_MasterGetBaudRate (
    uint8 Instance )
```

Get the currently configured baud rate.

**Note**

Function ID: DES\_I2C\_API\_240  
Service ID: NA

**Parameters**

|    |                 |                           |
|----|-----------------|---------------------------|
| in | <i>Instance</i> | : I2C hardware channel ID |
|----|-----------------|---------------------------|

**Returns**

uint32: Get the currently baud rate

Definition at line 2357 of file I2c\_Hal.c.

**4.3.3.8 I2c\_Hal\_MasterSetBaudRate()**

```
Hal_StatusType I2c_Hal_MasterSetBaudRate (
    uint8 Instance,
    uint32 BaudRate )
```

Set the currently configured baud rate.

**Note**

Function ID: DES\_I2C\_API\_204  
Service ID: NA

**Parameters**

|    |                 |                           |
|----|-----------------|---------------------------|
| in | <i>Instance</i> | : I2C hardware channel ID |
| in | <i>BaudRate</i> | I2C channel Set baudRate  |

**Returns**

Hal\_StatusType: STATUS\_SUCCESS Set the currently configured baud rate. Success STATUS\_ERROR Set the currently configured baud rate Failed

Definition at line 2391 of file I2c\_Hal.c.

**4.3.3.9 I2C\_Hal\_SlaveGetRxSize()**

```
uint32 I2C_Hal_SlaveGetRxSize (  
    uint8 Instance )
```

Get the listening receiving data.

**Parameters**

|    |                 |                         |
|----|-----------------|-------------------------|
| in | <i>Instance</i> | The I2C instance number |
|----|-----------------|-------------------------|

**Returns**

The receive data len

Definition at line 3039 of file I2c\_Hal.c.

**4.3.3.10 I2C\_Hal\_SlaveSetRxBuffer()**

```
Hal_StatusType I2C_Hal_SlaveSetRxBuffer (  
    uint8 Instance,  
    uint8 * RxBuff,  
    uint32 RxSize )
```

Provide a buffer for receiving data.

**Parameters**

|    |                 |  |
|----|-----------------|--|
| in | <i>Instance</i> | The I2C instance number                            |
| in | <i>RxBuff</i>   | Pointer to the buffer where to store received data |
| in | <i>RxSize</i>   | Length of the data to be transferred               |

**Returns**

The result of execution

Definition at line 3015 of file I2c\_Hal.c.

**4.3.3.11 I2C\_Hal\_SlaveSetTxBuffer()**

```
Hal_StatusType I2C_Hal_SlaveSetTxBuffer (
    uint8 Instance,
    const uint8 * TxBuff,
    uint32 TxSize )
```

Provide a buffer for transmitting data.

**Parameters**

|    |                 |                                       |
|----|-----------------|---------------------------------------|
| in | <i>Instance</i> | The I2C instance number               |
| in | <i>TxBuff</i>   | Pointer to the data to be transferred |
| in | <i>TxSize</i>   | Length of the data to be transferred  |

**Returns**

The result of execution

Definition at line 2988 of file I2c\_Hal.c.

**4.3.3.12 I2c\_Hal\_SyncTransceive()**

```
Hal_StatusType I2c_Hal_SyncTransceive (
    uint8 Instance,
    const DataTransmitType * DataTransmitPtr )
```

Starts an synchronous transmission on the I2C bus.

**Note**

Function ID:DES\_I2C\_API\_241

Service ID: NA

**Parameters**

|         |                        |                           |
|---------|------------------------|---------------------------|
| in      | <i>Instance</i>        | : I2C hardware channel ID |
| in, out | <i>DataTransmitPtr</i> | transmit data used        |

**Returns**

Hal\_StatusType: STATUS\_SUCCESS: Success to Send or receive STATUS\_ERROR: Otherwise

Definition at line 2496 of file I2c\_Hal.c.

#### 4.3.3.13 ISR()

```
ISR (
    I2C0_IRQHandler )
```

Definition at line 3085 of file I2c\_Hal.c.

## 4.4 I2c\_Hal.h File Reference

This file provides cdd I2c hal function extern.

```
#include "I2c_Hal_Types.h"
```

### Functions

- void [I2c\\_Hal\\_Init](#) (uint8 Instance, const [I2c\\_Hal\\_ChannelConfigType](#) \*ChannelConfigPtr)  
*Initializes the I2C module.*
- void [I2c\\_Hal\\_DeInit](#) (uint8 Instance)  
*DeInitializes the I2C module.*
- uint32 [I2c\\_Hal\\_MasterGetBaudRate](#) (uint8 Instance)  
*Get the currently configured baud rate.*
- Hal\_StatusType [I2c\\_Hal\\_MasterSetBaudRate](#) (uint8 Instance, uint32 BaudRate)  
*Set the currently configured baud rate.*
- Hal\_StatusType [I2c\\_Hal\\_SyncTransceive](#) (uint8 Instance, const [DataTransmitType](#) \*DataTransmitPtr)  
*Starts an synchronous transmission on the I2C bus.*
- Hal\_StatusType [I2c\\_Hal\\_AsyncTransceive](#) (uint8 Instance, const [DataTransmitType](#) \*DataTransmitPtr)  
*Starts an asynchronous transmission on the I2C bus.*
- Hal\_StatusType [I2c\\_Hal\\_AbortTransceive](#) (uint8 Instance)  
*Starts an asynchronous transmission on the I2C bus.*
- [I2c\\_Hal\\_ChannelStatusType](#) [I2c\\_Hal\\_GetStatus](#) (uint8 Instance)  
*Gets the status of an I2C channel.*
- I2C\_Type \* [I2c\\_Hal\\_GetBase](#) (uint8 Instance)  
*Get the I2C base.*
- Hal\_StatusType [I2C\\_Hal\\_SlaveSetTxBuffer](#) (uint8 Instance, const uint8 \*TxBuff, uint32 TxSize)  
*Provide a buffer for transmitting data.*
- Hal\_StatusType [I2C\\_Hal\\_SlaveSetRxBuffer](#) (uint8 Instance, uint8 \*RxBuff, uint32 RxSize)  
*Provide a buffer for receiving data.*
- uint32 [I2C\\_Hal\\_SlaveGetRxSize](#) (uint8 Instance)  
*Get the listening receiving data.*

### 4.4.1 Detailed Description

This file provides cdd I2c hal function extern.

## 4.4.2 Function Documentation

### 4.4.2.1 I2c\_Hal\_AbortTransceive()

```
Hal_StatusType I2c_Hal_AbortTransceive (
    uint8 Instance )
```

Starts an asynchronous transmission on the I2C bus.

#### Note

Function ID:DES\_I2C\_API\_242  
Service ID: NA

#### Parameters

|    |                 |                           |
|----|-----------------|---------------------------|
| in | <i>Instance</i> | : I2C hardware channel ID |
|----|-----------------|---------------------------|

#### Returns

Hal\_StatusType: STATUS\_SUCCESS: Success to Send or receive STATUS\_ERROR: Otherwise

Starts an asynchronous transmission on the I2C bus.

#### Note

Function ID:DES\_I2C\_API\_242  
Service ID: NA

#### Parameters

|    |                 |                           |
|----|-----------------|---------------------------|
| in | <i>Instance</i> | : I2C hardware channel ID |
|----|-----------------|---------------------------|

#### Returns

Hal\_StatusType: STATUS\_SUCCESS: Success to Send or receive STATUS\_ERROR: Otherwise

Definition at line 2603 of file I2c\_Hal.c.

### 4.4.2.2 I2c\_Hal\_AsyncTransceive()

```
Hal_StatusType I2c_Hal_AsyncTransceive (
    uint8 Instance,
    const DataTransmitType * DataTransmitPtr )
```

Starts an asynchronous transmission on the I2C bus.

#### Note

Function ID:DES\_I2C\_API\_202  
Service ID: NA

**Parameters**

|                |                        |                           |
|----------------|------------------------|---------------------------|
| <i>in</i>      | <i>Instance</i>        | : I2C hardware channel ID |
| <i>in, out</i> | <i>DataTransmitPtr</i> | transmit data used        |

**Returns**

Hal\_StatusType: STATUS\_SUCCESS: Success to Send or receive STATUS\_ERROR: Otherwise

Definition at line 2546 of file I2c\_Hal.c.

**4.4.2.3 I2c\_Hal\_DeInit()**

```
void I2c_Hal_DeInit (
    uint8 Instance )
```

DeInitializes the I2C module.

**Note**

Function ID:DES\_I2C\_API\_201  
Service ID: NA

**Parameters**

|           |                 |                           |
|-----------|-----------------|---------------------------|
| <i>in</i> | <i>Instance</i> | : I2C hardware channel ID |
|-----------|-----------------|---------------------------|

**Returns**

void

Definition at line 2305 of file I2c\_Hal.c.

**4.4.2.4 I2c\_Hal\_GetBase()**

```
I2C_Type* I2c_Hal_GetBase (
    uint8 Instance )
```

Get the I2C base.

**Note**

Function ID:DES\_I2C\_API\_243

**Parameters**

|           |                 |                            |
|-----------|-----------------|----------------------------|
| <i>in</i> | <i>Instance</i> | : I2C hardware channel ID. |
|-----------|-----------------|----------------------------|

Returns

I2C\_Type\*: the I2C base addr.

Definition at line 2730 of file I2c\_Hal.c.

4.4.2.5 I2c\_Hal\_GetStatus()

```
I2c_Hal_ChannelStatusType I2c_Hal_GetStatus (
    uint8 Instance )
```

Gets the status of an I2C channel.

Note

Function ID:DES\_I2C\_API\_203  
Service ID: NA

Parameters

|    |          |                           |
|----|----------|---------------------------|
| in | Instance | : I2C hardware channel ID |
|----|----------|---------------------------|

Returns

I2c\_ChannelStatusType: I2c channel currenr status

Definition at line 2639 of file I2c\_Hal.c.

4.4.2.6 I2c\_Hal\_Init()

```
void I2c_Hal_Init (
    uint8 Instance,
    const I2c_Hal_ChannelConfigType * ChannelConfigPtr )
```

Initializes the I2C module.

Note

Function ID:DES\_I2C\_API\_200  
Service ID: NA

Parameters

|    |                  |                                    |
|----|------------------|------------------------------------|
| in | Instance         | : I2C hardware channel ID          |
| in | ChannelConfigPtr | : Pointer to I2c_ChannelConfigType |

**Returns**

void

**Note**

Function ID:DES\_I2C\_API\_200  
Service ID: NA

**Parameters**

|    |                         |                                    |
|----|-------------------------|------------------------------------|
| in | <i>Instance</i>         | : I2C hardware channel ID          |
| in | <i>ChannelConfigPtr</i> | : Pointer to I2c_ChannelConfigType |

**Returns**

Hal\_StatusType: STATUS\_SUCCESS Initializes Success STATUS\_ERROR Initializes Failed

Definition at line 2270 of file I2c\_Hal.c.

**4.4.2.7 I2c\_Hal\_MasterGetBaudRate()**

```
uint32 I2c_Hal_MasterGetBaudRate (
    uint8 Instance )
```

Get the currently configured baud rate.

**Note**

Function ID: DES\_I2C\_API\_240  
Service ID: NA

**Parameters**

|    |                 |                           |
|----|-----------------|---------------------------|
| in | <i>Instance</i> | : I2C hardware channel ID |
|----|-----------------|---------------------------|

**Returns**

uint32: Get the currently baud rate

Definition at line 2357 of file I2c\_Hal.c.

**4.4.2.8 I2c\_Hal\_MasterSetBaudRate()**

```
Hal_StatusType I2c_Hal_MasterSetBaudRate (
    uint8 Instance,
    uint32 BaudRate )
```

Set the currently configured baud rate.



**Note**

Function ID: DES\_I2C\_API\_204  
Service ID: NA

**Parameters**

|    |                 |                           |
|----|-----------------|---------------------------|
| in | <i>Instance</i> | : I2C hardware channel ID |
| in | <i>BaudRate</i> | I2C channel Set baudRate  |

**Returns**

Hal\_StatusType: STATUS\_SUCCESS Set the currently configured baud rate. Success STATUS\_ERROR Set the currently configured baud rate Failed

Definition at line 2391 of file I2c\_Hal.c.

**4.4.2.9 I2C\_Hal\_SlaveGetRxSize()**

```
uint32 I2C_Hal_SlaveGetRxSize (  
    uint8 Instance )
```

Get the listening receiving data.

**Parameters**

|    |                 |                         |
|----|-----------------|-------------------------|
| in | <i>Instance</i> | The I2C instance number |
|----|-----------------|-------------------------|

**Returns**

The receive data len

Definition at line 3039 of file I2c\_Hal.c.

**4.4.2.10 I2C\_Hal\_SlaveSetRxBuffer()**

```
Hal_StatusType I2C_Hal_SlaveSetRxBuffer (  
    uint8 Instance,  
    uint8 * RxBuff,  
    uint32 RxSize )
```

Provide a buffer for receiving data.

**Parameters**

|    |                 |  |
|----|-----------------|--|
| in | <i>Instance</i> | The I2C instance number                            |
| in | <i>RxBuff</i>   | Pointer to the buffer where to store received data |
| in | <i>RxSize</i>   | Length of the data to be transferred               |

**Returns**

The result of execution

Definition at line 3015 of file I2c\_Hal.c.

**4.4.2.11 I2C\_Hal\_SlaveSetTxBuffer()**

```
Hal_StatusType I2C_Hal_SlaveSetTxBuffer (
    uint8 Instance,
    const uint8 * TxBuff,
    uint32 TxSize )
```

Provide a buffer for transmitting data.

**Parameters**

|    |                 |                                       |
|----|-----------------|---------------------------------------|
| in | <i>Instance</i> | The I2C instance number               |
| in | <i>TxBuff</i>   | Pointer to the data to be transferred |
| in | <i>TxSize</i>   | Length of the data to be transferred  |

**Returns**

The result of execution

Definition at line 2988 of file I2c\_Hal.c.

**4.4.2.12 I2c\_Hal\_SyncTransceive()**

```
Hal_StatusType I2c_Hal_SyncTransceive (
    uint8 Instance,
    const DataTransmitType * DataTransmitPtr )
```

Starts an synchronous transmission on the I2C bus.

**Note**

Function ID:DES\_I2C\_API\_241

Service ID: NA

**Parameters**

|         |                        |                           |
|---------|------------------------|---------------------------|
| in      | <i>Instance</i>        | : I2C hardware channel ID |
| in, out | <i>DataTransmitPtr</i> | transmit data used        |

**Returns**

Hal\_StatusType: STATUS\_SUCCESS: Success to Send or receive STATUS\_ERROR: Otherwise

Definition at line 2496 of file I2c\_Hal.c.

## 4.5 I2c\_Hal\_Types.h File Reference

This file provides i2c config.

```
#include "Device_Register.h"
#include "Platform_Types.h"
```

### Classes

- struct [I2c\\_Hal\\_MasterConfigType](#)  
*Configuration structure that the user needs to set.*
- struct [I2c\\_Hal\\_SlaveConfigType](#)  
*Slave configuration structure.*
- struct [DataTransmitType](#)  
*Data TransmitType.*
- struct [I2c\\_Hal\\_ChannelConfigType](#)  
*I2C configuration structure.*

### Macros

- #define [I2C\\_INSTANCE\\_ID](#) 0
- #define [I2C\\_ADDEXT\\_PRIMARY\\_BYTE\\_FIX](#) (0xF8U)  
*I2C 10bit address reserve mask bit macro.*

### Typedefs

- typedef uint8 [I2c\\_TransmitData](#)
- typedef void(\* [I2c\\_Hal\\_CallbackType](#)) (uint8 Instance, uint32 Event)

### Enumerations

- enum { [I2C\\_ENABLE\\_RX](#) = 0x0U, [I2C\\_ENABLE\\_TX](#) }  
*The enum contains the I2c TX or RX.*
- enum { [I2C\\_SLAVE\\_EVENT\\_RX\\_FULL](#) = 0x00U, [I2C\\_SLAVE\\_EVENT\\_TX\\_EMPTY](#) = 0x01U, [I2C\\_SLAVE\\_EVENT\\_TX\\_REQ](#) = 0x02U, [I2C\\_SLAVE\\_EVENT\\_RX\\_REQ](#) = 0x03U, [I2C\\_SLAVE\\_EVENT\\_STOP](#) = 0x04U, [I2C\\_SLAVE\\_EVENT\\_ADDRESS\\_MATCH](#) = 0x05U, [I2C\\_MASTER\\_EVENT\\_T\\_END\\_TRANSFER](#) = 0x06U, [I2C\\_MASTER\\_EVENT\\_PLTIE](#) = 0x07U }  
*Define the enum of the events which can trigger I2C callback.*
- enum [I2c\\_ModeTypes](#) { [I2C\\_SLAVE](#) = 0x00U, [I2C\\_MASTER](#) = 0x01U }  
*I2C master/slave mode enum.*
- enum [I2c\\_HwTypes](#) { [HW\\_I2C](#) = 0x00U, [EIO\\_I2C](#) = 0x01U }  
*I2C HW Type.*
- enum [I2c\\_Hal\\_DirType](#) { [I2C\\_WRITE](#) = 0x00U, [I2C\\_READ](#) = 0x01U }  
*Type of I2C transfer direction (write or read)*
- enum [I2c\\_Hal\\_TransferType](#) { [I2C\\_USING\\_DMA](#) = 0x00U, [I2C\\_USING\\_INTERRUPTS](#) = 0x01U }

*Type of I2C transfer(based on interrupts or DMA)*

- enum `I2c_Hal_TimeCfgTypes` { `I2C_SCL` = 0x00U, `I2C_SDA_SCL` = 0x01U }

*I2C timecfg mode enum.*

- enum `I2c_Hal_MasterStatusType` {  
`I2C_MASTER_CHANNEL_IDLE` = 0U, `I2C_MASTER_CHANNEL_BUSY_SEND`, `I2C_MASTER_CHANNEL_BUSY_RECEIVE`, `I2C_MASTER_CHANNEL_FINISHED`,  
`I2C_MASTER_CHANNEL_ARBITRATION_LOST`, `I2C_MASTER_CHANNEL_RECEIVE_NACK`, `I2C_MASTER_CHANNEL_ABORTED`, `I2C_MASTER_CHANNEL_TIMEOUT`,  
`I2C_MASTER_CHANNEL_DMA_ERROR` }
- enum `I2c_Hal_SlaveStatusType` {  
`I2C_SLAVE_CHANNEL_IDLE` = 0U, `I2C_SLAVE_CHANNEL_BUSY_SEND`, `I2C_SLAVE_CHANNEL_BUSY_RECEIVE`, `I2C_SLAVE_CHANNEL_BUSY_TRANSMIT`,  
`I2C_SLAVE_CHANNEL_FINISHED`, `I2C_SLAVE_CHANNEL_TX_EMPTY`, `I2C_SLAVE_CHANNEL_TX_UNDERRUN`, `I2C_SLAVE_CHANNEL_RX_OVERRUN`,  
`I2C_SLAVE_CHANNEL_ABORTED`, `I2C_SLAVE_CHANNEL_DMA_ERROR` }
- enum `I2c_Hal_ChannelStatusType` {  
`I2C_CHANNEL_IDLE` = 0U, `I2C_CHANNEL_BUSY_TRANSMIT`, `I2C_CHANNEL_FINISHED`, `I2C_CHANNEL_ERROR_PRESENT`,  
`I2C_CHANNEL_ABORTED_SUCCESS` }

### 4.5.1 Detailed Description

This file provides i2c config.

### 4.5.2 Macro Definition Documentation

#### 4.5.2.1 I2C\_ADDEXT\_PRIMARY\_BYTE\_FIX

```
#define I2C_ADDEXT_PRIMARY_BYTE_FIX (0xF8U)
```

I2C 10bit address reserve mask bit macro.

Definition at line 60 of file `I2c_Hal_Types.h`.

#### 4.5.2.2 I2C\_INSTANCE\_ID

```
#define I2C_INSTANCE_ID 0
```

Definition at line 51 of file `I2c_Hal_Types.h`.

### 4.5.3 Typedef Documentation

#### 4.5.3.1 I2c\_Hal\_CallbackType

```
typedef void(* I2c_Hal_CallbackType) (uint8 Instance, uint32 Event)
```

Definition at line 76 of file I2c\_Hal\_Types.h.

#### 4.5.3.2 I2c\_TransmitData

```
typedef uint8 I2c_TransmitData
```

< I2c transmit Data Callback for I2C master mode

Definition at line 74 of file I2c\_Hal\_Types.h.

### 4.5.4 Enumeration Type Documentation

#### 4.5.4.1 anonymous enum

```
anonymous enum
```

The enum contains the I2c TX or RX.

##### Enumerator

|               |               |
|---------------|---------------|
| I2C_ENABLE_RX | Enable I2C RX |
| I2C_ENABLE_TX | Enable I2C TX |

Definition at line 81 of file I2c\_Hal\_Types.h.

#### 4.5.4.2 anonymous enum

```
anonymous enum
```

Define the enum of the events which can trigger I2C callback.

##### Enumerator

|                               |                          |
|-------------------------------|--------------------------|
| I2C_SLAVE_EVENT_RX_FULL       | Rx buffer is full        |
| I2C_SLAVE_EVENT_TX_EMPTY      | Tx buffer is empty       |
| I2C_SLAVE_EVENT_TX_REQ        | Tx request               |
| I2C_SLAVE_EVENT_RX_REQ        | Rx request               |
| I2C_SLAVE_EVENT_STOP          | Slave detected stop      |
| I2C_SLAVE_EVENT_ADDRESS_MATCH | Slave address match      |
| I2C_MASTER_EVENT_END_TRANSFER | Master transferend event |
| I2C_MASTER_EVENT_PLTIE        | Master pltie event       |

Definition at line 90 of file I2c\_Hal\_Types.h.

#### 4.5.4.3 I2c\_Hal\_ChannelStatusType

```
enum I2c_Hal_ChannelStatusType
```

##### Enumerator

|                             |   |
|-----------------------------|---|
| I2C_CHANNEL_IDLE            | brief Status Indication I2C channel is idle           |
| I2C_CHANNEL_BUSY_TRANSMIT   | brief Status Indication Transmit operation is ongoing |
| I2C_CHANNEL_FINISHED        | brief Status Indication operation is finished         |
| I2C_CHANNEL_ERROR_PRESENT   | brief Status Indication an error is present           |
| I2C_CHANNEL_ABORTED_SUCCESS | brief Status Indication operation abort success       |

Definition at line 174 of file I2c\_Hal\_Types.h.

#### 4.5.4.4 I2c\_Hal\_DirType

```
enum I2c_Hal_DirType
```

Type of I2C transfer direction (write or read)

##### Enumerator

|           |                    |
|-----------|--------------------|
| I2C_WRITE | I2C write transfer |
| I2C_READ  | I2C read transfer  |

Definition at line 123 of file I2c\_Hal\_Types.h.

#### 4.5.4.5 I2c\_Hal\_MasterStatusType

```
enum I2c_Hal_MasterStatusType
```

##### Enumerator

|                                     |   |
|-------------------------------------|---|
| I2C_MASTER_CHANNEL_IDLE             | brief Status Indication I2C channel is idle             |
| I2C_MASTER_CHANNEL_BUSY_SEND        | brief Status Indication send operation is ongoing       |
| I2C_MASTER_CHANNEL_BUSY_RECEIVE     | brief Status Indication receiving operation is ongoing  |
| I2C_MASTER_CHANNEL_FINISHED         | brief Status Indication operation is finished           |
| I2C_MASTER_CHANNEL_ARBITRATION_LOST | brief Status Indication operation is arbitration lost   |
| I2C_MASTER_CHANNEL_RECEIVE_NACK     | brief Status Indication operation is receive NACK       |
| I2C_MASTER_CHANNEL_ABORTED          | brief Status Indication operation is aborted            |
| I2C_MASTER_CHANNEL_TIMEOUT          | brief Status Indication operation is timeout            |
| I2C_MASTER_CHANNEL_DMA_ERROR        | brief Status Indication operation is Dma transmit error |

Definition at line 147 of file I2c\_Hal\_Types.h.

#### 4.5.4.6 I2c\_Hal\_SlaveStatusType

```
enum I2c_Hal_SlaveStatusType
```

##### Enumerator

|                                 |   |
|---------------------------------|---|
| I2C_SLAVE_CHANNEL_IDLE          | brief Status Indication I2C channel is idle             |
| I2C_SLAVE_CHANNEL_BUSY_SEND     | brief Status Indication send operation is ongoing       |
| I2C_SLAVE_CHANNEL_BUSY_RECEIVE  | brief Status Indication receiving operation is ongoing  |
| I2C_SLAVE_CHANNEL_BUSY_TRANSMIT | brief Status Indication Transmit operation is ongoing   |
| I2C_SLAVE_CHANNEL_FINISHED      | brief Status Indication operation is finished           |
| I2C_SLAVE_CHANNEL_TX_EMPTY      | brief Status Indication an error is tx empty            |
| I2C_SLAVE_CHANNEL_TX_UNDERRUN   | brief Status Indication an error is under run           |
| I2C_SLAVE_CHANNEL_RX_OVERRUN    | brief Status Indication an error is over run            |
| I2C_SLAVE_CHANNEL_ABORTED       | brief Status Indication operation is aborted            |
| I2C_SLAVE_CHANNEL_DMA_ERROR     | brief Status Indication operation is Dma transmit error |

Definition at line 160 of file I2c\_Hal\_Types.h.

#### 4.5.4.7 I2c\_Hal\_TimeCfgTypes

```
enum I2c_Hal_TimeCfgTypes
```

I2C timecfg mode enum.

##### Enumerator

|             |                  |
|-------------|------------------|
| I2C_SCL     | I2C SCL mode     |
| I2C_SDA_SCL | I2C SDA/SCL mode |

Definition at line 141 of file I2c\_Hal\_Types.h.

#### 4.5.4.8 I2c\_Hal\_TransferType

```
enum I2c_Hal_TransferType
```

Type of I2C transfer(based on interrupts or DMA)

##### Enumerator

|                      |   |
|----------------------|---|
| I2C_USING_DMA        | The driver will use DMA to perform I2C transfer       |
| I2C_USING_INTERRUPTS | The driver will use interrupt to perform I2C transfer |

Definition at line 132 of file I2c\_Hal\_Types.h.

#### 4.5.4.9 I2c\_HwTypes

enum [I2c\\_HwTypes](#)

I2C HW Type.

Enumerator

|         |         |
|---------|---------|
| HW_I2C  | HW I2C  |
| EIO_I2C | EIO I2C |

Definition at line 114 of file I2c\_Hal\_Types.h.

#### 4.5.4.10 I2c\_ModeTypes

enum [I2c\\_ModeTypes](#)

I2C master/slave mode enum.

Enumerator

|            |                 |
|------------|-----------------|
| I2C_SLAVE  | I2C slave mode  |
| I2C_MASTER | I2C master mode |

Definition at line 105 of file I2c\_Hal\_Types.h.



# Index

AC784xx\_API\_Reference\_Manual\_I2C.pdf, 18

AC784xx\_I2c\_Reg.h, 18

I2C\_HW\_DEADLINE\_TIMEOUT, 21

I2C\_Reg\_GetSampleCnt, 23

I2C\_Reg\_GetStepCnt, 24

I2C\_Reg\_IsADEXT, 25

I2C\_Reg\_IsBNDDMAInterrupt, 25

I2C\_Reg\_IsBND, 25

I2C\_Reg\_IsBusy, 26

I2C\_Reg\_IsDMARxEnable, 26

I2C\_Reg\_IsMaster, 27

I2C\_Reg\_IsNack, 28

I2C\_Reg\_IsReady, 28

I2C\_Reg\_IsSMBusAlertResponse, 29

I2C\_Reg\_SetADEXT, 36

I2C\_Reg\_SetBNDDMAInterrupt, 37

I2C\_Reg\_SetDGLCnt, 38

I2C\_Reg\_SetDGL, 38

I2C\_Reg\_SetDebug, 37

I2C\_Reg\_SetGCA, 40

I2C\_Reg\_SetMNT, 41

I2C\_Reg\_SetRAD, 43

I2C\_Reg\_SetRxFInterrupt, 44

I2C\_Reg\_SetRxOFInterrupt, 44

I2C\_Reg\_SetSMBusAlert, 46

I2C\_Reg\_SetSlaveRangeAddr, 46

I2C\_Reg\_SetSoftwareReset, 47

I2C\_Reg\_SetTxElInterrupt, 49

I2C\_Reg\_SetTxUFInterrupt, 49

I2c\_Reg\_ClearPltieFlag, 21

I2c\_Reg\_ClearStartFlag, 21

I2c\_Reg\_ClearStatus0, 22

I2c\_Reg\_ClearStatus1, 22

I2c\_Reg\_ClearStopFlag, 23

I2c\_Reg\_GetStatus0, 24

I2c\_Reg\_IsDMATxEnable, 27

I2c\_Reg\_IsPltie, 28

I2c\_Reg\_IsRxOF, 29

I2c\_Reg\_IsSSIntEnable, 30

I2c\_Reg\_IsStart, 30

I2c\_Reg\_IsStop, 31

I2c\_Reg\_IsTx, 31

I2c\_Reg\_IsTxUF, 32

I2c\_Reg\_ReadDataReg, 32

I2c\_Reg\_ReceiveLastOneByte, 33

I2c\_Reg\_RxEn, 33

I2c\_Reg\_SendAck, 34

I2c\_Reg\_SendNack, 34

I2c\_Reg\_SendStart, 35

I2c\_Reg\_SendStop, 35

I2c\_Reg\_SetARB, 36

I2c\_Reg\_SetDMARx, 39

I2c\_Reg\_SetDMATx, 39

I2c\_Reg\_SetInterrupt, 40

I2c\_Reg\_SetMSTR, 42

I2c\_Reg\_SetModuleEnable, 41

I2c\_Reg\_SetNackInterrupt, 42

I2c\_Reg\_SetPltie, 43

I2c\_Reg\_SetSSInterrupt, 47

I2c\_Reg\_SetSYNC, 48

I2c\_Reg\_SetSampleStep, 45

I2c\_Reg\_SetSlaveAddr, 46

I2c\_Reg\_SetStretch, 48

I2c\_Reg\_SetWakeUp, 50

I2c\_Reg\_TransmitOneByte, 50

I2c\_Reg\_TxEn, 51

I2c\_Reg\_WriteDataReg, 51

Arbitration

I2c\_Hal\_MasterConfigType, 7

BaudRate

I2c\_Hal\_MasterConfigType, 7

Callback

I2c\_Hal\_ChannelConfigType, 5

DataBufferPtr

DataTransmitType, 3

DataLength

DataTransmitType, 3

DataTransmitType, 3

DataBufferPtr, 3

DataLength, 3

DirType, 4

Is10bitAddr, 4

SendStop, 4

SlaveAddress, 4

DirType

DataTransmitType, 4

I2c\_Hal\_MasterStateType, 10

I2C\_ADDEXT\_PRIMARY\_BYTE\_FIX

I2c\_Hal\_Types.h, 68

I2C\_HW\_DEADLINE\_TIMEOUT

AC784xx\_I2c\_Reg.h, 21

I2C\_Hal\_SlaveGetRxSize

I2c\_Hal.c, 58

I2c\_Hal.h, 65

I2C\_Hal\_SlaveSetRxBuffer

I2c\_Hal.c, 58

I2c\_Hal.h, 65

I2C\_Hal\_SlaveSetTxBuffer

I2c\_Hal.c, 59

I2c\_Hal.h, 66

I2C\_INSTANCE\_ID

- I2c\_Hal\_Types.h, 68
- I2C\_Reg\_GetSampleCnt
  - AC784xx\_I2c\_Reg.h, 23
- I2C\_Reg\_GetStepCnt
  - AC784xx\_I2c\_Reg.h, 24
- I2C\_Reg\_IsADEXT
  - AC784xx\_I2c\_Reg.h, 25
- I2C\_Reg\_IsBNDDMAInterrupt
  - AC784xx\_I2c\_Reg.h, 25
- I2C\_Reg\_IsBND
  - AC784xx\_I2c\_Reg.h, 25
- I2C\_Reg\_IsBusy
  - AC784xx\_I2c\_Reg.h, 26
- I2C\_Reg\_IsDMARxEnable
  - AC784xx\_I2c\_Reg.h, 26
- I2C\_Reg\_IsMaster
  - AC784xx\_I2c\_Reg.h, 27
- I2C\_Reg\_IsNack
  - AC784xx\_I2c\_Reg.h, 28
- I2C\_Reg\_IsReady
  - AC784xx\_I2c\_Reg.h, 28
- I2C\_Reg\_IsSMBusAlertResponse
  - AC784xx\_I2c\_Reg.h, 29
- I2C\_Reg\_SetADEXT
  - AC784xx\_I2c\_Reg.h, 36
- I2C\_Reg\_SetBNDDMAInterrupt
  - AC784xx\_I2c\_Reg.h, 37
- I2C\_Reg\_SetDGLCnt
  - AC784xx\_I2c\_Reg.h, 38
- I2C\_Reg\_SetDGL
  - AC784xx\_I2c\_Reg.h, 38
- I2C\_Reg\_SetDebug
  - AC784xx\_I2c\_Reg.h, 37
- I2C\_Reg\_SetGCA
  - AC784xx\_I2c\_Reg.h, 40
- I2C\_Reg\_SetMNT
  - AC784xx\_I2c\_Reg.h, 41
- I2C\_Reg\_SetRAD
  - AC784xx\_I2c\_Reg.h, 43
- I2C\_Reg\_SetRxFlInterrupt
  - AC784xx\_I2c\_Reg.h, 44
- I2C\_Reg\_SetRxOFInterrupt
  - AC784xx\_I2c\_Reg.h, 44
- I2C\_Reg\_SetSMBusAlert
  - AC784xx\_I2c\_Reg.h, 46
- I2C\_Reg\_SetSlaveRangeAddr
  - AC784xx\_I2c\_Reg.h, 46
- I2C\_Reg\_SetSoftwareReset
  - AC784xx\_I2c\_Reg.h, 47
- I2C\_Reg\_SetTxElInterrupt
  - AC784xx\_I2c\_Reg.h, 49
- I2C\_Reg\_SetTxUFIInterrupt
  - AC784xx\_I2c\_Reg.h, 49
- I2C\_WAIT\_BND
  - I2c\_Hal.c, 53
- I2C\_WAIT\_STATUS
  - I2c\_Hal.c, 53
- I2c\_Hal.c, 52
  - I2C\_Hal\_SlaveGetRxSize, 58
  - I2C\_Hal\_SlaveSetRxBuffer, 58
  - I2C\_Hal\_SlaveSetTxBuffer, 59
- I2C\_WAIT\_BND, 53
- I2C\_WAIT\_STATUS, 53
- I2c\_Hal\_AbortTransceive, 54
- I2c\_Hal\_AsyncTransceive, 54
- I2c\_Hal\_DelInit, 55
- I2c\_Hal\_GetBase, 55
- I2c\_Hal\_GetStatus, 56
- I2c\_Hal\_Init, 56
- I2c\_Hal\_MasterGetBaudRate, 57
- I2c\_Hal\_MasterSetBaudRate, 57
- I2c\_Hal\_SyncTransceive, 59
- ISR, 60
- I2c\_Hal.h, 60
  - I2C\_Hal\_SlaveGetRxSize, 65
  - I2C\_Hal\_SlaveSetRxBuffer, 65
  - I2C\_Hal\_SlaveSetTxBuffer, 66
  - I2c\_Hal\_AbortTransceive, 61
  - I2c\_Hal\_AsyncTransceive, 61
  - I2c\_Hal\_DelInit, 62
  - I2c\_Hal\_GetBase, 62
  - I2c\_Hal\_GetStatus, 63
  - I2c\_Hal\_Init, 63
  - I2c\_Hal\_MasterGetBaudRate, 64
  - I2c\_Hal\_MasterSetBaudRate, 64
  - I2c\_Hal\_SyncTransceive, 66
- I2c\_Hal\_AbortTransceive
  - I2c\_Hal.c, 54
  - I2c\_Hal.h, 61
- I2c\_Hal\_AsyncTransceive
  - I2c\_Hal.c, 54
  - I2c\_Hal.h, 61
- I2c\_Hal\_CallbackType
  - I2c\_Hal\_Types.h, 68
- I2c\_Hal\_ChannelConfigType, 5
  - Callback, 5
  - I2cMode, 5
  - I2cType, 5
  - MasterConfigPtr, 5
  - SlaveConfigPtr, 6
- I2c\_Hal\_ChannelStatusType
  - I2c\_Hal\_Types.h, 70
- I2c\_Hal\_DelInit
  - I2c\_Hal.c, 55
  - I2c\_Hal.h, 62
- I2c\_Hal\_DirType
  - I2c\_Hal\_Types.h, 70
- I2c\_Hal\_GetBase
  - I2c\_Hal.c, 55
  - I2c\_Hal.h, 62
- I2c\_Hal\_GetStatus
  - I2c\_Hal.c, 56
  - I2c\_Hal.h, 63
- I2c\_Hal\_Init
  - I2c\_Hal.c, 56
  - I2c\_Hal.h, 63
- I2c\_Hal\_MasterConfigType, 6
  - Arbitration, 7
  - BaudRate, 7
  - PinLow, 7
  - RxDmaChannel, 7
  - SclPin, 7

- SclSdaLowEn, [7](#)
- SdaPin, [8](#)
- SendStop, [8](#)
- SlaveAddress, [8](#)
- SyncEn, [8](#)
- TimeCfgTypes, [8](#)
- TransferType, [9](#)
- TxDmaChannel, [9](#)
- I2c\_Hal\_MasterGetBaudRate
  - I2c\_Hal.c, [57](#)
  - I2c\_Hal.h, [64](#)
- I2c\_Hal\_MasterSetBaudRate
  - I2c\_Hal.c, [57](#)
  - I2c\_Hal.h, [64](#)
- I2c\_Hal\_MasterStateType, [9](#)
  - DirType, [10](#)
  - Is10bitAddr, [10](#)
  - RxBuffPtr, [10](#)
  - RxDmaChannel, [10](#)
  - RxPointer, [10](#)
  - RxSize, [11](#)
  - SendStop, [11](#)
  - SlaveAddress, [11](#)
  - SourceClock, [11](#)
  - Status, [11](#)
  - TransferType, [12](#)
  - TxBuffPtr, [12](#)
  - TxDmaChannel, [12](#)
  - TxSize, [12](#)
- I2c\_Hal\_MasterStatusType
  - I2c\_Hal\_Types.h, [70](#)
- I2c\_Hal\_SlaveConfigType, [13](#)
  - Is10bitAddr, [13](#)
  - RxDmaChannel, [13](#)
  - SlaveAddress, [13](#)
  - slaveListening, [14](#)
  - StretchEn, [14](#)
  - TransferType, [14](#)
  - TxDmaChannel, [14](#)
  - WakeupEn, [14](#)
- I2c\_Hal\_SlaveStateType, [15](#)
  - RxBuffPtr, [15](#)
  - RxDmaChannel, [15](#)
  - RxPointer, [16](#)
  - RxSize, [16](#)
  - SlaveAddress, [16](#)
  - slaveListening, [16](#)
  - Status, [16](#)
  - TransferType, [17](#)
  - TxBuffPtr, [17](#)
  - TxDmaChannel, [17](#)
  - TxSize, [17](#)
- I2c\_Hal\_SlaveStatusType
  - I2c\_Hal\_Types.h, [71](#)
- I2c\_Hal\_SyncTransceive
  - I2c\_Hal.c, [59](#)
  - I2c\_Hal.h, [66](#)
- I2c\_Hal\_TimeCfgTypes
  - I2c\_Hal\_Types.h, [71](#)
- I2c\_Hal\_TransferType
  - I2c\_Hal\_Types.h, [71](#)
- I2c\_Hal\_Types.h, [67](#)
  - I2C\_ADDEXT\_PRIMARY\_BYTE\_FIX, [68](#)
  - I2C\_INSTANCE\_ID, [68](#)
  - I2c\_Hal\_CallbackType, [68](#)
  - I2c\_Hal\_ChannelStatusType, [70](#)
  - I2c\_Hal\_DirType, [70](#)
  - I2c\_Hal\_MasterStatusType, [70](#)
  - I2c\_Hal\_SlaveStatusType, [71](#)
  - I2c\_Hal\_TimeCfgTypes, [71](#)
  - I2c\_Hal\_TransferType, [71](#)
  - I2c\_HwTypes, [72](#)
  - I2c\_ModeTypes, [72](#)
  - I2c\_TransmitData, [69](#)
- I2c\_HwTypes
  - I2c\_Hal\_Types.h, [72](#)
- I2c\_ModeTypes
  - I2c\_Hal\_Types.h, [72](#)
- I2c\_Reg\_ClearPltieFlag
  - AC784xx\_I2c\_Reg.h, [21](#)
- I2c\_Reg\_ClearStartFlag
  - AC784xx\_I2c\_Reg.h, [21](#)
- I2c\_Reg\_ClearStatus0
  - AC784xx\_I2c\_Reg.h, [22](#)
- I2c\_Reg\_ClearStatus1
  - AC784xx\_I2c\_Reg.h, [22](#)
- I2c\_Reg\_ClearStopFlag
  - AC784xx\_I2c\_Reg.h, [23](#)
- I2c\_Reg\_GetStatus0
  - AC784xx\_I2c\_Reg.h, [24](#)
- I2c\_Reg\_IsDMATxEnable
  - AC784xx\_I2c\_Reg.h, [27](#)
- I2c\_Reg\_IsPltie
  - AC784xx\_I2c\_Reg.h, [28](#)
- I2c\_Reg\_IsRxOF
  - AC784xx\_I2c\_Reg.h, [29](#)
- I2c\_Reg\_IsSSIntEnable
  - AC784xx\_I2c\_Reg.h, [30](#)
- I2c\_Reg\_IsStart
  - AC784xx\_I2c\_Reg.h, [30](#)
- I2c\_Reg\_IsStop
  - AC784xx\_I2c\_Reg.h, [31](#)
- I2c\_Reg\_IsTx
  - AC784xx\_I2c\_Reg.h, [31](#)
- I2c\_Reg\_IsTxUF
  - AC784xx\_I2c\_Reg.h, [32](#)
- I2c\_Reg\_ReadDataReg
  - AC784xx\_I2c\_Reg.h, [32](#)
- I2c\_Reg\_ReceiveLastOneByte
  - AC784xx\_I2c\_Reg.h, [33](#)
- I2c\_Reg\_RxEn
  - AC784xx\_I2c\_Reg.h, [33](#)
- I2c\_Reg\_SendAck
  - AC784xx\_I2c\_Reg.h, [34](#)
- I2c\_Reg\_SendNack
  - AC784xx\_I2c\_Reg.h, [34](#)
- I2c\_Reg\_SendStart
  - AC784xx\_I2c\_Reg.h, [35](#)
- I2c\_Reg\_SendStop
  - AC784xx\_I2c\_Reg.h, [35](#)
- I2c\_Reg\_SetARB
  - AC784xx\_I2c\_Reg.h, [36](#)

- I2c\_Reg\_SetDMARx
  - AC784xx\_I2c\_Reg.h, [39](#)
- I2c\_Reg\_SetDMATx
  - AC784xx\_I2c\_Reg.h, [39](#)
- I2c\_Reg\_SetInterrupt
  - AC784xx\_I2c\_Reg.h, [40](#)
- I2c\_Reg\_SetMSTR
  - AC784xx\_I2c\_Reg.h, [42](#)
- I2c\_Reg\_SetModuleEnable
  - AC784xx\_I2c\_Reg.h, [41](#)
- I2c\_Reg\_SetNackInterrupt
  - AC784xx\_I2c\_Reg.h, [42](#)
- I2c\_Reg\_SetPltie
  - AC784xx\_I2c\_Reg.h, [43](#)
- I2c\_Reg\_SetSSInterrupt
  - AC784xx\_I2c\_Reg.h, [47](#)
- I2c\_Reg\_SetSYNC
  - AC784xx\_I2c\_Reg.h, [48](#)
- I2c\_Reg\_SetSampleStep
  - AC784xx\_I2c\_Reg.h, [45](#)
- I2c\_Reg\_SetSlaveAddr
  - AC784xx\_I2c\_Reg.h, [46](#)
- I2c\_Reg\_SetStretch
  - AC784xx\_I2c\_Reg.h, [48](#)
- I2c\_Reg\_SetWakeup
  - AC784xx\_I2c\_Reg.h, [50](#)
- I2c\_Reg\_TransmitOneByte
  - AC784xx\_I2c\_Reg.h, [50](#)
- I2c\_Reg\_TxEn
  - AC784xx\_I2c\_Reg.h, [51](#)
- I2c\_Reg\_WriteDataReg
  - AC784xx\_I2c\_Reg.h, [51](#)
- I2c\_TransmitData
  - I2c\_Hal\_Types.h, [69](#)
- I2cMode
  - I2c\_Hal\_ChannelConfigType, [5](#)
- I2cType
  - I2c\_Hal\_ChannelConfigType, [5](#)
- ISR
  - I2c\_Hal.c, [60](#)
- Is10bitAddr
  - DataTransmitType, [4](#)
  - I2c\_Hal\_MasterStateType, [10](#)
  - I2c\_Hal\_SlaveConfigType, [13](#)
- MasterConfigPtr
  - I2c\_Hal\_ChannelConfigType, [5](#)
- PinLow
  - I2c\_Hal\_MasterConfigType, [7](#)
- RxBuffPtr
  - I2c\_Hal\_MasterStateType, [10](#)
  - I2c\_Hal\_SlaveStateType, [15](#)
- RxDmaChannel
  - I2c\_Hal\_MasterConfigType, [7](#)
  - I2c\_Hal\_MasterStateType, [10](#)
  - I2c\_Hal\_SlaveConfigType, [13](#)
  - I2c\_Hal\_SlaveStateType, [15](#)
- RxPointer
  - I2c\_Hal\_MasterStateType, [10](#)
- I2c\_Hal\_SlaveStateType, [16](#)
- RxSize
  - I2c\_Hal\_MasterStateType, [11](#)
  - I2c\_Hal\_SlaveStateType, [16](#)
- SclPin
  - I2c\_Hal\_MasterConfigType, [7](#)
- SclSdaLowEn
  - I2c\_Hal\_MasterConfigType, [7](#)
- SdaPin
  - I2c\_Hal\_MasterConfigType, [8](#)
- SendStop
  - DataTransmitType, [4](#)
  - I2c\_Hal\_MasterConfigType, [8](#)
  - I2c\_Hal\_MasterStateType, [11](#)
- SlaveAddress
  - DataTransmitType, [4](#)
  - I2c\_Hal\_MasterConfigType, [8](#)
  - I2c\_Hal\_MasterStateType, [11](#)
  - I2c\_Hal\_SlaveConfigType, [13](#)
  - I2c\_Hal\_SlaveStateType, [16](#)
- SlaveConfigPtr
  - I2c\_Hal\_ChannelConfigType, [6](#)
- slaveListening
  - I2c\_Hal\_SlaveConfigType, [14](#)
  - I2c\_Hal\_SlaveStateType, [16](#)
- SourceClock
  - I2c\_Hal\_MasterStateType, [11](#)
- Status
  - I2c\_Hal\_MasterStateType, [11](#)
  - I2c\_Hal\_SlaveStateType, [16](#)
- StretchEn
  - I2c\_Hal\_SlaveConfigType, [14](#)
- SyncEn
  - I2c\_Hal\_MasterConfigType, [8](#)
- TimeCfgTypes
  - I2c\_Hal\_MasterConfigType, [8](#)
- TransferType
  - I2c\_Hal\_MasterConfigType, [9](#)
  - I2c\_Hal\_MasterStateType, [12](#)
  - I2c\_Hal\_SlaveConfigType, [14](#)
  - I2c\_Hal\_SlaveStateType, [17](#)
- TxBuffPtr
  - I2c\_Hal\_MasterStateType, [12](#)
  - I2c\_Hal\_SlaveStateType, [17](#)
- TxDmaChannel
  - I2c\_Hal\_MasterConfigType, [9](#)
  - I2c\_Hal\_MasterStateType, [12](#)
  - I2c\_Hal\_SlaveConfigType, [14](#)
  - I2c\_Hal\_SlaveStateType, [17](#)
- TxSize
  - I2c\_Hal\_MasterStateType, [12](#)
  - I2c\_Hal\_SlaveStateType, [17](#)
- WakeupEn
  - I2c\_Hal\_SlaveConfigType, [14](#)