

AC784xx_DFP UART

5.1.0

Generated by Doxygen 1.8.13

Contents

1	Class Index	1
1.1	Class List	1
2	File Index	2
2.1	File List	2
3	Class Documentation	3
3.1	Uart_ChannelConfigType Struct Reference	3
3.1.1	Detailed Description	3
3.1.2	Member Data Documentation	3
3.1.2.1	BaudRate	3
3.1.2.2	BitCountPerChar	4
3.1.2.3	ParityMode	4
3.1.2.4	RxCallback	4
3.1.2.5	RxDmaChannel	4
3.1.2.6	SampleCnt	4
3.1.2.7	StopBitCount	5
3.1.2.8	TransferType	5
3.1.2.9	TxCallback	5
3.1.2.10	TxDmaChannel	5
3.2	Uart_ChannelStateType Struct Reference	5
3.2.1	Detailed Description	6
3.2.2	Member Data Documentation	6
3.2.2.1	BitCountPerChar	6
3.2.2.2	InitState	6

3.2.2.3	IsRxBlocking	7
3.2.2.4	IsRxBusy	7
3.2.2.5	IsTxBlocking	7
3.2.2.6	IsTxBusy	7
3.2.2.7	ReceiveStatus	7
3.2.2.8	RxBuff	8
3.2.2.9	RxCallback	8
3.2.2.10	RxComplete	8
3.2.2.11	RxDmaChannel	8
3.2.2.12	RxSize	8
3.2.2.13	SendStatus	9
3.2.2.14	TransferType	9
3.2.2.15	TxBuff	9
3.2.2.16	TxCallback	9
3.2.2.17	TxComplete	9
3.2.2.18	TxDmaChannel	10
3.2.2.19	TxSize	10
3.3	Uart_IrDAConfigType Struct Reference	10
3.3.1	Detailed Description	10
3.3.2	Member Data Documentation	10
3.3.2.1	Enable	11
3.3.2.2	RxWidth	11
3.3.2.3	TxMode	11
3.3.2.4	TxWidth	11
3.4	Uart_RS485ConfigType Struct Reference	11
3.4.1	Detailed Description	12
3.4.2	Member Data Documentation	12
3.4.2.1	DelayCnt	12
3.4.2.2	DelayEnable	12
3.4.2.3	Enable	12
3.4.2.4	GuardTime	13
3.4.2.5	InvpolEnable	13

4 File Documentation	14
4.1 AC784xx_API_Reference_Manual_UART.pdf File Reference	14
4.2 AC784xx_Uart_Reg.h File Reference	14
4.2.1 Detailed Description	17
4.2.2 Macro Definition Documentation	17
4.2.2.1 UART_DIV_FRAC_VAL	17
4.2.2.2 UART_DIV_H_POS	17
4.2.2.3 UART_DIV_MASK	17
4.2.2.4 UART_IDLE_REG_ID	17
4.2.2.5 UART_IER_REG_ID	17
4.2.2.6 UART_INDEX_MAX	18
4.2.2.7 UART_LINCR_REG_ID	18
4.2.2.8 UART_LSR0_REG_ID	18
4.2.2.9 UART_LSR1_REG_ID	18
4.2.2.10 UART_MATCHCR_REG_ID	18
4.2.2.11 UART_SHIFT	18
4.2.3 Enumeration Type Documentation	18
4.2.3.1 Uart_InterruptConfigType	18
4.2.3.2 Uart_StatusFlagType	19
4.2.4 Function Documentation	19
4.2.4.1 Uart_Reg_ClearErrorFlags()	20
4.2.4.2 Uart_Reg_ClearStatusFlag()	20
4.2.4.3 Uart_Reg_GetBase()	20
4.2.4.4 Uart_Reg_GetBaudRateDivisor()	21
4.2.4.5 Uart_Reg_GetChar()	21
4.2.4.6 Uart_Reg_GetChar9()	21
4.2.4.7 Uart_Reg_GetIntMode()	23
4.2.4.8 Uart_Reg_GetLSR0()	23
4.2.4.9 Uart_Reg_GetLSR1()	24
4.2.4.10 Uart_Reg_GetSampleCounter()	24
4.2.4.11 Uart_Reg_GetStatusFlag()	24

4.2.4.12	Uart_Reg_IsErrorInterruptEnable()	25
4.2.4.13	Uart_Reg_PutChar()	25
4.2.4.14	Uart_Reg_PutChar9()	25
4.2.4.15	Uart_Reg_ReadLinSleep()	27
4.2.4.16	Uart_Reg_SendLINBreak()	27
4.2.4.17	Uart_Reg_SetAddrFilter()	28
4.2.4.18	Uart_Reg_SetBaudRateDivisor()	28
4.2.4.19	Uart_Reg_SetBitCountPerChar()	28
4.2.4.20	Uart_Reg_SetCtsRts()	29
4.2.4.21	Uart_Reg_SetDataMatch()	29
4.2.4.22	Uart_Reg_SetErrorInterrupts()	30
4.2.4.23	Uart_Reg_SetFIFO()	30
4.2.4.24	Uart_Reg_SetIdleInterrupt()	31
4.2.4.25	Uart_Reg_SetInterruptEn()	31
4.2.4.26	Uart_Reg_SetIntMode()	31
4.2.4.27	Uart_Reg_SetInvRx()	32
4.2.4.28	Uart_Reg_SetInvTx()	32
4.2.4.29	Uart_Reg_SetIrDA()	33
4.2.4.30	Uart_Reg_SetIrDARxWidth()	33
4.2.4.31	Uart_Reg_SetIrDATxWidth()	34
4.2.4.32	Uart_Reg_SetLinBreakLength()	34
4.2.4.33	Uart_Reg_SetLinBreakThreshold()	34
4.2.4.34	Uart_Reg_SetLinCtrl()	35
4.2.4.35	Uart_Reg_SetLinSleep()	35
4.2.4.36	Uart_Reg_SetLoop()	36
4.2.4.37	Uart_Reg_SetParityMode()	36
4.2.4.38	Uart_Reg_SetReceiverCmd()	36
4.2.4.39	Uart_Reg_SetRS485()	37
4.2.4.40	Uart_Reg_SetRS485Delay()	37
4.2.4.41	Uart_Reg_SetRS485DelayCnt()	38
4.2.4.42	Uart_Reg_SetRS485GuardEnable()	38

4.2.4.43	Uart_Reg_SetRS485GuardTime()	38
4.2.4.44	Uart_Reg_SetRS485Invpol()	39
4.2.4.45	Uart_Reg_SetRxDmaCmd()	39
4.2.4.46	Uart_Reg_SetSampleCounter()	40
4.2.4.47	Uart_Reg_SetSingleWire()	40
4.2.4.48	Uart_Reg_SetStopBitCount()	41
4.2.4.49	Uart_Reg_SetTransmitterCmd()	41
4.2.4.50	Uart_Reg_SetTxDir()	41
4.2.4.51	Uart_Reg_SetTxDmaCmd()	42
4.3	Uart_Hal.c File Reference	42
4.3.1	Detailed Description	44
4.3.2	Macro Definition Documentation	44
4.3.2.1	UART_SAMPLE_CNT_16_VALUE	44
4.3.2.2	UART_SAMPLE_CNT_4_VALUE	44
4.3.2.3	UART_SAMPLE_CNT_8_VALUE	44
4.3.2.4	UART_USE_DMA_TRANSMIT_LEN_MAX	44
4.3.3	Function Documentation	45
4.3.3.1	Uart_Hal_AbortReceivingData()	45
4.3.3.2	Uart_Hal_AbortSendingData()	45
4.3.3.3	Uart_Hal_CheckTimeout()	46
4.3.3.4	Uart_Hal_DeInit()	46
4.3.3.5	Uart_Hal_GetBaudRate()	47
4.3.3.6	Uart_Hal_GetReceiveStatus()	47
4.3.3.7	Uart_Hal_GetSendStatus()	48
4.3.3.8	Uart_Hal_Init()	48
4.3.3.9	Uart_Hal_ReceiveData()	49
4.3.3.10	Uart_Hal_ReceiveDataBlocking()	49
4.3.3.11	Uart_Hal_ReceiveDataPolling()	50
4.3.3.12	Uart_Hal_SendData()	50
4.3.3.13	Uart_Hal_SendDataBlocking()	51
4.3.3.14	Uart_Hal_SendDataPolling()	51

4.3.3.15	Uart_Hal_SetAddrFilter()	52
4.3.3.16	Uart_Hal_SetBaudRate()	52
4.3.3.17	Uart_Hal_SetCtsRts()	53
4.3.3.18	Uart_Hal_SetDataMatch()	53
4.3.3.19	Uart_Hal_SetIdleInterrupt()	54
4.3.3.20	Uart_Hal_SetIrDA()	54
4.3.3.21	Uart_Hal_SetMatchInterrupt()	55
4.3.3.22	Uart_Hal_SetRS485()	55
4.3.3.23	Uart_Hal_StartTimeout()	56
4.4	Uart_Hal.h File Reference	56
4.4.1	Function Documentation	57
4.4.1.1	Uart_Hal_AbortReceivingData()	57
4.4.1.2	Uart_Hal_AbortSendingData()	58
4.4.1.3	Uart_Hal_DeInit()	58
4.4.1.4	Uart_Hal_GetBaudRate()	59
4.4.1.5	Uart_Hal_GetReceiveStatus()	59
4.4.1.6	Uart_Hal_GetSendStatus()	60
4.4.1.7	Uart_Hal_Init()	60
4.4.1.8	Uart_Hal_ReceiveData()	61
4.4.1.9	Uart_Hal_ReceiveDataBlocking()	61
4.4.1.10	Uart_Hal_ReceiveDataPolling()	62
4.4.1.11	Uart_Hal_SendData()	62
4.4.1.12	Uart_Hal_SendDataBlocking()	63
4.4.1.13	Uart_Hal_SendDataPolling()	63
4.4.1.14	Uart_Hal_SetAddrFilter()	64
4.4.1.15	Uart_Hal_SetBaudRate()	64
4.4.1.16	Uart_Hal_SetCtsRts()	65
4.4.1.17	Uart_Hal_SetDataMatch()	65
4.4.1.18	Uart_Hal_SetIdleInterrupt()	66
4.4.1.19	Uart_Hal_SetIrDA()	66
4.4.1.20	Uart_Hal_SetMatchInterrupt()	67
4.4.1.21	Uart_Hal_SetRS485()	67
4.5	Uart_Hal_Types.h File Reference	68
4.5.1	Typedef Documentation	69
4.5.1.1	Uart_CallbackType	69
4.5.2	Enumeration Type Documentation	69
4.5.2.1	Uart_EventType	69
4.5.2.2	Uart_ParityModeConfigType	69
4.5.2.3	Uart_PerCharConfigType	70
4.5.2.4	Uart_RtsCtsType	70
4.5.2.5	Uart_SampleCntType	70
4.5.2.6	Uart_StopBitConfigType	71
4.5.2.7	Uart_TransferType	71
4.5.2.8	Uart_TxPinDirType	71

CONTENTS	vii
Index	72

Chapter 1

Class Index

1.1 Class List

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

Uart_ChannelConfigType	
UART configuration structure	3
Uart_ChannelStateType	
UART runtime state	5
Uart_IrDAConfigType	
Configuration structure for IrDA settings	10
Uart_RS485ConfigType	
Configuration structure for RS485 settings	11

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

AC784xx_API_Reference_Manual_UART.pdf	14
AC784xx_Uart_Reg.h	
This file provides extern Reg lin api	14
Uart_Hal.c	
This file provides Uart hal function extern	42
Uart_Hal.h	56
Uart_Hal_Types.h	68

Chapter 3

Class Documentation

3.1 Uart_ChannelConfigType Struct Reference

UART configuration structure.

```
#include <Uart_Hal_Types.h>
```

Public Attributes

- uint32 [BaudRate](#)
- [Uart_ParityModeConfigType](#) [ParityMode](#)
- [Uart_StopBitConfigType](#) [StopBitCount](#)
- [Uart_PerCharConfigType](#) [BitCountPerChar](#)
- [Uart_TransferType](#) [TransferType](#)
- [Uart_SampleCntType](#) [SampleCnt](#)
- uint8 [RxDmaChannel](#)
- uint8 [TxDmaChannel](#)
- [Uart_CallbackType](#) [RxCallback](#)
- [Uart_CallbackType](#) [TxCallback](#)

3.1.1 Detailed Description

UART configuration structure.

Definition at line 136 of file `Uart_Hal_Types.h`.

3.1.2 Member Data Documentation

3.1.2.1 BaudRate

```
uint32 Uart_ChannelConfigType::BaudRate
```

UART baud rate

Definition at line 138 of file `Uart_Hal_Types.h`.

3.1.2.2 BitCountPerChar

`Uart_PerCharConfigType` `Uart_ChannelConfigType::BitCountPerChar`

number of bits in a character (7, 8 or 9)

Definition at line 141 of file `Uart_Hal_Types.h`.

3.1.2.3 ParityMode

`Uart_ParityModeConfigType` `Uart_ChannelConfigType::ParityMode`

parity mode, disabled, even, odd

Definition at line 139 of file `Uart_Hal_Types.h`.

3.1.2.4 RxCallback

`Uart_CallbackType` `Uart_ChannelConfigType::RxCallback`

Callback for data receive

Definition at line 146 of file `Uart_Hal_Types.h`.

3.1.2.5 RxDmaChannel

`uint8` `Uart_ChannelConfigType::RxDmaChannel`

DMA rx channel for DMA transfer

Definition at line 144 of file `Uart_Hal_Types.h`.

3.1.2.6 SampleCnt

`Uart_SampleCntType` `Uart_ChannelConfigType::SampleCnt`

UART sample counter

Definition at line 143 of file `Uart_Hal_Types.h`.

3.1.2.7 StopBitCount

`Uart_StopBitConfigType` `Uart_ChannelConfigType::StopBitCount`

number of stop bits, 1 or 2 stop bits

Definition at line 140 of file `Uart_Hal_Types.h`.

3.1.2.8 TransferType

`Uart_TransferType` `Uart_ChannelConfigType::TransferType`

Type of UART transfer (interrupt or dma)

Definition at line 142 of file `Uart_Hal_Types.h`.

3.1.2.9 TxCallback

`Uart_CallbackType` `Uart_ChannelConfigType::TxCallback`

Callback for data send

Definition at line 147 of file `Uart_Hal_Types.h`.

3.1.2.10 TxDmaChannel

`uint8` `Uart_ChannelConfigType::TxDmaChannel`

DMA tx channel for DMA transfer

Definition at line 145 of file `Uart_Hal_Types.h`.

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

- [Uart_Hal_Types.h](#)

3.2 Uart_ChannelStateType Struct Reference

UART runtime state.

Public Attributes

- boolean [InitState](#)
- const uint8 * [TxBuff](#)
- uint8 * [RxBuff](#)
- uint32 [TxSize](#)
- uint32 [RxSize](#)
- boolean [IsTxBusy](#)
- boolean [IsRxBusy](#)
- boolean [IsTxBlocking](#)
- boolean [IsRxBlocking](#)
- [Uart_PerCharConfigType](#) [BitCountPerChar](#)
- [Uart_CallbackType](#) [RxCallback](#)
- [Uart_CallbackType](#) [TxCallback](#)
- [Uart_TransferType](#) [TransferType](#)
- uint8 [RxDmaChannel](#)
- uint8 [TxDmaChannel](#)
- volatile boolean [RxComplete](#)
- volatile boolean [TxComplete](#)
- [Hal_StatusType](#) [SendStatus](#)
- [Hal_StatusType](#) [ReceiveStatus](#)

3.2.1 Detailed Description

UART runtime state.

Definition at line 68 of file `Uart_Hal.c`.

3.2.2 Member Data Documentation

3.2.2.1 BitCountPerChar

`Uart_PerCharConfigType Uart_ChannelStateType::BitCountPerChar`

number of bits(7/8/9 bits) in a char

Definition at line 79 of file `Uart_Hal.c`.

3.2.2.2 InitState

`boolean Uart_ChannelStateType::InitState`

Definition at line 70 of file `Uart_Hal.c`.

3.2.2.3 IsRxBlocking

```
boolean Uart_ChannelStateType::IsRxBlocking
```

True if receive is blocking

Definition at line 78 of file Uart_Hal.c.

3.2.2.4 IsRxBusy

```
boolean Uart_ChannelStateType::IsRxBusy
```

True if receive is active

Definition at line 76 of file Uart_Hal.c.

3.2.2.5 IsTxBlocking

```
boolean Uart_ChannelStateType::IsTxBlocking
```

True if transmit is blocking

Definition at line 77 of file Uart_Hal.c.

3.2.2.6 IsTxBusy

```
boolean Uart_ChannelStateType::IsTxBusy
```

True if transmit is active

Definition at line 75 of file Uart_Hal.c.

3.2.2.7 ReceiveStatus

```
Hal_StatusType Uart_ChannelStateType::ReceiveStatus
```

Status of last receive

Definition at line 88 of file Uart_Hal.c.

3.2.2.8 RxBuff

```
uint8* Uart_ChannelStateType::RxBuff
```

The buffer of received data

Definition at line 72 of file Uart_Hal.c.

3.2.2.9 RxCallback

```
Uart_CallbackType Uart_ChannelStateType::RxCallback
```

Callback for data receive

Definition at line 80 of file Uart_Hal.c.

3.2.2.10 RxComplete

```
volatile boolean Uart_ChannelStateType::RxComplete
```

Flag to indicate the completion of a blocking receive operation

Definition at line 85 of file Uart_Hal.c.

3.2.2.11 RxDmaChannel

```
uint8 Uart_ChannelStateType::RxDmaChannel
```

DMA channel for UART receive.

Definition at line 83 of file Uart_Hal.c.

3.2.2.12 RxSize

```
uint32 Uart_ChannelStateType::RxSize
```

The remaining number of bytes to be received

Definition at line 74 of file Uart_Hal.c.

3.2.2.13 SendStatus

```
Hal_StatusType Uart_ChannelStateType::SendStatus
```

Status of last transmit

Definition at line 87 of file Uart_Hal.c.

3.2.2.14 TransferType

```
Uart_TransferType Uart_ChannelStateType::TransferType
```

Type of UART transfer (interrupt or dma)

Definition at line 82 of file Uart_Hal.c.

3.2.2.15 TxBuff

```
const uint8* Uart_ChannelStateType::TxBuff
```

The buffer of transmit data

Definition at line 71 of file Uart_Hal.c.

3.2.2.16 TxCallback

```
Uart_CallbackType Uart_ChannelStateType::TxCallback
```

Callback for data send

Definition at line 81 of file Uart_Hal.c.

3.2.2.17 TxComplete

```
volatile boolean Uart_ChannelStateType::TxComplete
```

Flag to indicate the completion of a blocking transmit operation

Definition at line 86 of file Uart_Hal.c.

3.2.2.18 TxDmaChannel

```
uint8 Uart_ChannelStateType::TxDmaChannel
```

DMA channel for UART transmit.

Definition at line 84 of file Uart_Hal.c.

3.2.2.19 TxSize

```
uint32 Uart_ChannelStateType::TxSize
```

The remaining number of bytes to be transmitted

Definition at line 73 of file Uart_Hal.c.

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

- [Uart_Hal.c](#)

3.3 Uart_IrDAConfigType Struct Reference

Configuration structure for IrDA settings.

```
#include <Uart_Hal_Types.h>
```

Public Attributes

- boolean [Enable](#)
- boolean [TxMode](#)
- uint16 [RxWidth](#)
- uint8 [TxWidth](#)

3.3.1 Detailed Description

Configuration structure for IrDA settings.

Definition at line 167 of file Uart_Hal_Types.h.

3.3.2 Member Data Documentation

3.3.2.1 Enable

```
boolean Uart_IrDAConfigType::Enable
```

Enable or disable IrDA mode

Definition at line 169 of file Uart_Hal_Types.h.

3.3.2.2 RxWidth

```
uint16 Uart_IrDAConfigType::RxWidth
```

Receive pulse width

Definition at line 171 of file Uart_Hal_Types.h.

3.3.2.3 TxMode

```
boolean Uart_IrDAConfigType::TxMode
```

Set IrDA mode to transmit(TRUE) or receive(FALSE)

Definition at line 170 of file Uart_Hal_Types.h.

3.3.2.4 TxWidth

```
uint8 Uart_IrDAConfigType::TxWidth
```

Transmit pulse width

Definition at line 172 of file Uart_Hal_Types.h.

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

- [Uart_Hal_Types.h](#)

3.4 Uart_RS485ConfigType Struct Reference

Configuration structure for RS485 settings.

```
#include <Uart_Hal_Types.h>
```

Public Attributes

- boolean [Enable](#)
- boolean [InvpolEnable](#)
- boolean [DelayEnable](#)
- uint8 [DelayCnt](#)
- uint8 [GuardTime](#)

3.4.1 Detailed Description

Configuration structure for RS485 settings.

Definition at line 155 of file Uart_Hal_Types.h.

3.4.2 Member Data Documentation

3.4.2.1 DelayCnt

```
uint8 Uart_RS485ConfigType::DelayCnt
```

RS485 delay time

Definition at line 160 of file Uart_Hal_Types.h.

3.4.2.2 DelayEnable

```
boolean Uart_RS485ConfigType::DelayEnable
```

enable or disable RS485 delay function

Definition at line 159 of file Uart_Hal_Types.h.

3.4.2.3 Enable

```
boolean Uart_RS485ConfigType::Enable
```

Enable or disable RS485 mode

Definition at line 157 of file Uart_Hal_Types.h.

3.4.2.4 GuardTime

```
uint8 Uart_RS485ConfigType::GuardTime
```

RS484 guard Time

Definition at line 161 of file Uart_Hal_Types.h.

3.4.2.5 InvpolEnable

```
boolean Uart_RS485ConfigType::InvpolEnable
```

Enable or disable polarity inversion

Definition at line 158 of file Uart_Hal_Types.h.

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

- [Uart_Hal_Types.h](#)

Chapter 4

File Documentation

4.1 AC784xx_API_Reference_Manual_UART.pdf File Reference

4.2 AC784xx_Uart_Reg.h File Reference

This file provides extern Reg lin api.

```
#include "Device_Register.h"
#include "Uart_Hal_Types.h"
```

Macros

- #define [UART_DIV_MASK](#) (0xFFU)
- #define [UART_DIV_H_POS](#) (0x8U)
- #define [UART_DIV_FRAC_VAL](#) (0x20U)
- #define [UART_INDEX_MAX](#) (4UL)
- #define [UART_SHIFT](#) (16U)
- #define [UART_LSR0_REG_ID](#) (1U)
- #define [UART_LSR1_REG_ID](#) (2U)
- #define [UART_IER_REG_ID](#) (3U)
- #define [UART_MATCHCR_REG_ID](#) (4U)
- #define [UART_LINCR_REG_ID](#) (5U)
- #define [UART_IDLE_REG_ID](#) (6U)

Enumerations

- enum [Uart_InterruptConfigType](#) {
 [UART_INT_RX_NOT_EMPTY](#), [UART_INT_TX_NOT_FULL](#), [UART_INT_TX_COMPLETE](#), [UART_INT_PARITY_ERR](#),
 [UART_INT_FRAME_ERR](#), [UART_INT_NOISE_ERR](#), [UART_INT_RX_OVERRUN](#), [UART_INT_CTS_CHANGE](#),
 [UART_INT_TX_FULL](#), [UART_INT_DATA_MATCH](#), [UART_INT_LIN_WAKEUP](#), [UART_INT_LIN_SYNC_ERR](#),
 [UART_INT_LIN_BREAK](#), [UART_INT_IDLE_LINE](#) }

 UART interrupt configuration structure.
- enum [Uart_StatusFlagType](#) {
 [UART_RX_DATA_READY](#), [UART_RX_OVERRUN](#), [UART_PARITY_ERR](#), [UART_FRAME_ERR](#),
 [UART_BREAK_ERR](#), [UART_TX_DATA_NOT_FULL](#), [UART_TX_COMPLETE](#), [UART_NOISE_ERR](#),
 [UART_TX_DATA_FULL](#), [UART_IDLE_LINE](#), [UART_LIN_SYNC_ERR](#), [UART_LIN_BREAK](#),
 [UART_CTS_CHANGE](#), [UART_LIN_WAKEUP](#), [UART_IDLE](#), [UART_CTS](#),
 [UART_RTS](#), [UART_DATA_MATCH](#), [UART_ADDR_FILTER_MATCH](#) }

Functions

- LOCAL_INLINE UART_Type * [Uart_Reg_GetBase](#) (uint8 Instance)
Get UART base.
- LOCAL_INLINE void [Uart_Reg_SetInterruptEn](#) (UART_Type *Base, uint16 InterruptEn)
uart interrupt disable or enable
- LOCAL_INLINE boolean [Uart_Reg_GetStatusFlag](#) (const UART_Type *Base, [Uart_StatusFlagType](#) StatusFlag)
Get UART status flags.
- LOCAL_INLINE void [Uart_Reg_PutChar](#) (UART_Type *Base, uint8 Data)
Sends the UART 7/8-bit character.
- LOCAL_INLINE void [Uart_Reg_PutChar9](#) (UART_Type *Base, uint16 Data)
UART send 9-bits data.
- LOCAL_INLINE void [Uart_Reg_GetChar](#) (const UART_Type *Base, uint8 *ReadData)
Gets the UART 7/8-bit character.
- LOCAL_INLINE void [Uart_Reg_GetChar9](#) (const UART_Type *Base, uint16 *ReadData)
UART get 9-bits data.
- LOCAL_INLINE void [Uart_Reg_SendLINBreak](#) (UART_Type *Base)
Send LIN break.
- LOCAL_INLINE uint32 [Uart_Reg_GetLSR0](#) (const UART_Type *Base)
Get LSR0 register value.
- LOCAL_INLINE uint32 [Uart_Reg_GetLSR1](#) (const UART_Type *Base)
Get LSR1 register value.
- LOCAL_INLINE void [Uart_Reg_SetIntMode](#) (UART_Type *Base, [Uart_InterruptConfigType](#) IntSrc, boolean Is↔Enable)
Set UART interrupt mode.
- LOCAL_INLINE void [Uart_Reg_SetErrorInterrupts](#) (UART_Type *Base, boolean IsEnable)
IsEnable or disable the UART error interrupts.
- LOCAL_INLINE boolean [Uart_Reg_IsErrorInterruptEnable](#) (const UART_Type *Base)
Check if error interrupts are enabled.
- LOCAL_INLINE void [Uart_Reg_SetIdleInterrupt](#) (UART_Type *Base, boolean IsEnable)
Set UART idle interrupt. Note: Please enable idle function first by called function UART_SetIdleFun, otherwise idle interrupt is invalid.
- LOCAL_INLINE void [Uart_Reg_SetTxDmaCmd](#) (UART_Type *Base, boolean IsEnable)
Configures UART transmit DMA request.
- LOCAL_INLINE void [Uart_Reg_SetRxDmaCmd](#) (UART_Type *Base, boolean IsEnable)
Configures UART receive DMA request.
- LOCAL_INLINE void [Uart_Reg_ClearErrorFlags](#) (UART_Type *Base)
Clears the error status flags.
- LOCAL_INLINE boolean [Uart_Reg_GetIntMode](#) (const UART_Type *Base, [Uart_InterruptConfigType](#) IntSrc)
Get UART interrupt source is enabled or disabled.
- LOCAL_INLINE void [Uart_Reg_SetSampleCounter](#) (UART_Type *Base, uint32 SampleCnt)
This function is used to set sample counter.
- LOCAL_INLINE uint32 [Uart_Reg_GetSampleCounter](#) (const UART_Type *Base)
Get UART sample counter value.
- LOCAL_INLINE void [Uart_Reg_SetBaudRateDivisor](#) (UART_Type *Base, float32 BaudRateDivisor)
This function is used to set baud rate divisor.
- LOCAL_INLINE float32 [Uart_Reg_GetBaudRateDivisor](#) (const UART_Type *Base)
Gets the UART baud rate divisor.
- LOCAL_INLINE void [Uart_Reg_SetBitCountPerChar](#) (UART_Type *Base, [Uart_PerCharConfigType](#) BitCountPer↔Char)
UART set bit count per char.
- LOCAL_INLINE void [Uart_Reg_SetParityMode](#) (UART_Type *Base, [Uart_ParityModeConfigType](#) Parity)
UART set parity mode.

- LOCAL_INLINE void [Uart_Reg_SetStopBitCount](#) (UART_Type *Base, [Uart_StopBitConfigType](#) StopBitCount)
Configures the number of stop bits.
- LOCAL_INLINE void [Uart_Reg_SetFIFO](#) (UART_Type *Base, boolean IsEnable)
IsEnable or disable UART fifo.
- LOCAL_INLINE void [Uart_Reg_SetTransmitterCmd](#) (UART_Type *Base, boolean IsEnable)
This function is used to set transmitter cmd.
- LOCAL_INLINE void [Uart_Reg_SetReceiverCmd](#) (UART_Type *Base, boolean IsEnable)
This function is used to set receiver cmd.
- LOCAL_INLINE void [Uart_Reg_SetLinBreakThreshold](#) (UART_Type *Base, uint32 Length)
Set LIN break threshold length for slave detect.
- LOCAL_INLINE void [Uart_Reg_SetLinBreakLength](#) (UART_Type *Base, uint32 Length)
This function is used to set LIN break length.
- LOCAL_INLINE void [Uart_Reg_SetLinCtrl](#) (UART_Type *Base, uint32 LinCtrl)
This function is used to set LIN ctrl.
- LOCAL_INLINE void [Uart_Reg_SetLinSleep](#) (UART_Type *Base, boolean IsEnable)
This function is used to set LIN sleep.
- LOCAL_INLINE uint32 [Uart_Reg_ReadLinSleep](#) (const UART_Type *Base)
This function is used to read LIN sleep status.
- LOCAL_INLINE void [Uart_Reg_ClearStatusFlag](#) (UART_Type *Base, [Uart_StatusFlagType](#) StatusFlag)
Clear UART status flags.
- LOCAL_INLINE void [Uart_Reg_SetDataMatch](#) (UART_Type *Base, uint16 Data, boolean IsEnable)
Set UART data match function.
- LOCAL_INLINE void [Uart_Reg_SetAddrFilter](#) (UART_Type *Base, uint16 Addr, boolean IsEnable)
Set UART address filter function.
- LOCAL_INLINE void [Uart_Reg_SetCtsRts](#) (UART_Type *Base, [Uart_RtsCtsType](#) RtsCts)
Set UART hardware flow control function.
- LOCAL_INLINE void [Uart_Reg_SetRS485](#) (UART_Type *Base, boolean Enable)
Enable or disable RS485 function.
- LOCAL_INLINE void [Uart_Reg_SetRS485Invpol](#) (UART_Type *Base, boolean Enable)
Set RS485 RTS polarity.
- LOCAL_INLINE void [Uart_Reg_SetRS485Delay](#) (UART_Type *Base, boolean Enable)
Set RS485 delay function.
- LOCAL_INLINE void [Uart_Reg_SetRS485DelayCnt](#) (UART_Type *Base, uint8 Counter)
Set RS485 delay time.
- LOCAL_INLINE void [Uart_Reg_SetRS485GuardEnable](#) (UART_Type *Base, boolean Enable)
Set RS484 Guart Time enable.
- LOCAL_INLINE void [Uart_Reg_SetRS485GuardTime](#) (UART_Type *Base, uint8 GuardTime)
Set RS484 Guart Time.
- LOCAL_INLINE void [Uart_Reg_SetIrDA](#) (UART_Type *Base, boolean Enable)
Enable/disable UART IrDA function.
- LOCAL_INLINE void [Uart_Reg_SetIrDARxWidth](#) (UART_Type *Base, uint16 Pulse)
Set IrDA receiver minimum pulse width.
- LOCAL_INLINE void [Uart_Reg_SetIrDATxWidth](#) (UART_Type *Base, uint8 Pulse)
Set IrDA transmit minimum pulse width.
- LOCAL_INLINE void [Uart_Reg_SetInvTx](#) (UART_Type *Base, boolean Enable)
UART inverse polarity for tx output.
- LOCAL_INLINE void [Uart_Reg_SetInvRx](#) (UART_Type *Base, boolean Enable)
UART inverse polarity for the rx input.
- LOCAL_INLINE void [Uart_Reg_SetSingleWire](#) (UART_Type *Base, boolean Enable)
Set UART single-wire mode.
- LOCAL_INLINE void [Uart_Reg_SetTxDir](#) (UART_Type *base, [Uart_TxPinDirType](#) Direction)
UART TX pin direction config in single-wire mode.
- LOCAL_INLINE void [Uart_Reg_SetLoop](#) (UART_Type *Base, boolean Enable)
Set UART loop mode.

4.2.1 Detailed Description

This file provides extern Reg lin api.

4.2.2 Macro Definition Documentation

4.2.2.1 UART_DIV_FRAC_VAL

```
#define UART_DIV_FRAC_VAL (0x20U)
```

Definition at line 60 of file AC784xx_Uart_Reg.h.

4.2.2.2 UART_DIV_H_POS

```
#define UART_DIV_H_POS (0x8U)
```

Definition at line 59 of file AC784xx_Uart_Reg.h.

4.2.2.3 UART_DIV_MASK

```
#define UART_DIV_MASK (0xFFU)
```

Definition at line 58 of file AC784xx_Uart_Reg.h.

4.2.2.4 UART_IDLE_REG_ID

```
#define UART_IDLE_REG_ID (6U)
```

Definition at line 69 of file AC784xx_Uart_Reg.h.

4.2.2.5 UART_IER_REG_ID

```
#define UART_IER_REG_ID (3U)
```

Definition at line 66 of file AC784xx_Uart_Reg.h.

4.2.2.6 UART_INDEX_MAX

```
#define UART_INDEX_MAX (4UL)
```

Definition at line 61 of file AC784xx_Uart_Reg.h.

4.2.2.7 UART_LINCR_REG_ID

```
#define UART_LINCR_REG_ID (5U)
```

Definition at line 68 of file AC784xx_Uart_Reg.h.

4.2.2.8 UART_LSR0_REG_ID

```
#define UART_LSR0_REG_ID (1U)
```

Definition at line 64 of file AC784xx_Uart_Reg.h.

4.2.2.9 UART_LSR1_REG_ID

```
#define UART_LSR1_REG_ID (2U)
```

Definition at line 65 of file AC784xx_Uart_Reg.h.

4.2.2.10 UART_MATCHCR_REG_ID

```
#define UART_MATCHCR_REG_ID (4U)
```

Definition at line 67 of file AC784xx_Uart_Reg.h.

4.2.2.11 UART_SHIFT

```
#define UART_SHIFT (16U)
```

Definition at line 63 of file AC784xx_Uart_Reg.h.

4.2.3 Enumeration Type Documentation

4.2.3.1 Uart_InterruptConfigType

```
enum Uart_InterruptConfigType
```

UART interrupt configuration structure.

Enumerator

UART_INT_RX_NOT_EMPTY	
UART_INT_TX_NOT_FULL	
UART_INT_TX_COMPLETE	
UART_INT_PARITY_ERR	
UART_INT_FRAME_ERR	
UART_INT_NOISE_ERR	
UART_INT_RX_OVERRUN	
UART_INT_CTS_CHANGE	
UART_INT_TX_FULL	
UART_INT_DATA_MATCH	
UART_INT_LIN_WAKEUP	
UART_INT_LIN_SYNC_ERR	
UART_INT_LIN_BREAK	
UART_INT_IDLE_LINE	

Definition at line 75 of file AC784xx_Uart_Reg.h.

4.2.3.2 Uart_StatusFlagType

```
enum Uart_StatusFlagType
```

Enumerator

UART_RX_DATA_READY	
UART_RX_OVERRUN	
UART_PARITY_ERR	
UART_FRAME_ERR	
UART_BREAK_ERR	
UART_TX_DATA_NOT_FULL	
UART_TX_COMPLETE	
UART_NOISE_ERR	
UART_TX_DATA_FULL	
UART_IDLE_LINE	
UART_LIN_SYNC_ERR	
UART_LIN_BREAK	
UART_CTS_CHANGE	
UART_LIN_WAKEUP	
UART_IDLE	
UART_CTS	
UART_RTS	
UART_DATA_MATCH	
UART_ADDR_FILTER_MATCH	

Definition at line 107 of file AC784xx_Uart_Reg.h.

4.2.4 Function Documentation

4.2.4.1 Uart_Reg_ClearErrorFlags()

```
LOCAL_INLINE void Uart_Reg_ClearErrorFlags (
    UART_Type * Base )
```

Clears the error status flags.

Parameters

in	<i>Base</i>	UART Base pointer
----	-------------	-------------------

Returns

void

Definition at line 416 of file AC784xx_Uart_Reg.h.

4.2.4.2 Uart_Reg_ClearStatusFlag()

```
LOCAL_INLINE void Uart_Reg_ClearStatusFlag (
    UART_Type * Base,
    Uart_StatusFlagType StatusFlag )
```

Clear UART status flags.

Parameters

in	<i>Base</i>	UART interrupt sources
in	<i>StatusFlag</i>	Select which status flag to be cleared

Returns

void

Definition at line 699 of file AC784xx_Uart_Reg.h.

4.2.4.3 Uart_Reg_GetBase()

```
LOCAL_INLINE UART_Type* Uart_Reg_GetBase (
    uint8 Instance )
```

Get UART base.

Parameters

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

Returns

UART_Type*: the UART base addr.

Definition at line 162 of file AC784xx_Uart_Reg.h.

4.2.4.4 Uart_Reg_GetBaudRateDivisor()

```
LOCAL_INLINE float32 Uart_Reg_GetBaudRateDivisor (
    const UART_Type * Base )
```

Gets the UART baud rate divisor.

Parameters

in	<i>Base</i>	UART base pointer
----	-------------	-------------------

Returns

The baud rate divisor

Definition at line 515 of file AC784xx_Uart_Reg.h.

4.2.4.5 Uart_Reg_GetChar()

```
LOCAL_INLINE void Uart_Reg_GetChar (
    const UART_Type * Base,
    uint8 * ReadData )
```

Gets the UART 7/8-bit character.

Parameters

in	<i>Base</i>	UART Instance
in	<i>ReadData</i>	Data read from receiver (7/8-bit)

Returns

void

Definition at line 250 of file AC784xx_Uart_Reg.h.

4.2.4.6 Uart_Reg_GetChar9()

```
LOCAL_INLINE void Uart_Reg_GetChar9 (
    const UART_Type * Base,
    uint16 * ReadData )
```

UART get 9-bits data.

Parameters

in	<i>Base</i>	UART Instance
in	<i>ReadData</i>	Data read from receiver (7/8-bit)

Returns

void

Definition at line 264 of file AC784xx_Uart_Reg.h.

4.2.4.7 Uart_Reg_GetIntMode()

```
LOCAL_INLINE boolean Uart_Reg_GetIntMode (
    const UART_Type * Base,
    Uart_InterruptConfigType IntSrc )
```

Get UART interrupt source is enabled or disabled.

Parameters

in	<i>Base</i>	UART Base pointer
in	<i>IntSrc</i>	UART interrupt sources

Returns

Whether the interrupt source is enabled or disabled

Definition at line 434 of file AC784xx_Uart_Reg.h.

4.2.4.8 Uart_Reg_GetLSR0()

```
LOCAL_INLINE uint32 Uart_Reg_GetLSR0 (
    const UART_Type * Base )
```

Get LSR0 register value.

Parameters

in	<i>Base</i>	UART Base pointer
----	-------------	-------------------

Returns

LSR0 register value

Definition at line 287 of file AC784xx_Uart_Reg.h.

4.2.4.9 Uart_Reg_GetLSR1()

```
LOCAL_INLINE uint32 Uart_Reg_GetLSR1 (
    const UART_Type * Base )
```

Get LSR1 register value.

Parameters

in	<i>Base</i>	UART Base pointer
----	-------------	-------------------

Returns

LSR1 register value

Definition at line 298 of file AC784xx_Uart_Reg.h.

4.2.4.10 Uart_Reg_GetSampleCounter()

```
LOCAL_INLINE uint32 Uart_Reg_GetSampleCounter (
    const UART_Type * Base )
```

Get UART sample counter value.

Parameters

in	<i>Base</i>	UART base pointer
----	-------------	-------------------

Returns

UART over sample counter value

Definition at line 482 of file AC784xx_Uart_Reg.h.

4.2.4.11 Uart_Reg_GetStatusFlag()

```
LOCAL_INLINE boolean Uart_Reg_GetStatusFlag (
    const UART_Type * Base,
    Uart_StatusFlagType StatusFlag )
```

Get UART status flags.

Parameters

in	<i>Base</i>	UART Base pointer
in	<i>statusFlag</i>	Select which status flag

Returns

The selected status flag is true or false

Definition at line 196 of file AC784xx_Uart_Reg.h.

4.2.4.12 Uart_Reg_IsErrorInterruptEnable()

```
LOCAL_INLINE boolean Uart_Reg_IsErrorInterruptEnable (
    const UART_Type * Base )
```

Check if error interrupts are enabled.

Parameters

in	<i>Base</i>	UART Base pointer
----	-------------	-------------------

Returns

TRUE means enabled, FLASE means disabled

Definition at line 362 of file AC784xx_Uart_Reg.h.

4.2.4.13 Uart_Reg_PutChar()

```
LOCAL_INLINE void Uart_Reg_PutChar (
    UART_Type * Base,
    uint8 Data )
```

Sends the UART 7/8-bit character.

Parameters

in	<i>Base</i>	UART Instance
in	<i>Data</i>	data to send (7/8-bit)

Returns

void

Definition at line 226 of file AC784xx_Uart_Reg.h.

4.2.4.14 Uart_Reg_PutChar9()

```
LOCAL_INLINE void Uart_Reg_PutChar9 (
    UART_Type * Base,
    uint16 Data )
```

UART send 9-bits data.

Parameters

in	<i>Base</i>	UART Instance
in	<i>Data</i>	The data which will be send

Returns

void

Definition at line 238 of file AC784xx_Uart_Reg.h.

4.2.4.15 Uart_Reg_ReadLinSleep()

```
LOCAL_INLINE uint32 Uart_Reg_ReadLinSleep (
    const UART_Type * Base )
```

This function is used to read LIN sleep status.

Parameters

in	<i>Base</i>	UART interrupt sources
----	-------------	------------------------

Returns

void

Definition at line 685 of file AC784xx_Uart_Reg.h.

4.2.4.16 Uart_Reg_SendLINBreak()

```
LOCAL_INLINE void Uart_Reg_SendLINBreak (
    UART_Type * Base )
```

Send LIN break.

Parameters

in	<i>Base</i>	UART Base pointer
----	-------------	-------------------

Returns

void

Definition at line 276 of file AC784xx_Uart_Reg.h.

4.2.4.17 Uart_Reg_SetAddrFilter()

```
LOCAL_INLINE void Uart_Reg_SetAddrFilter (
    UART_Type * Base,
    uint16 Addr,
    boolean IsEnable )
```

Set UART address filter function.

Parameters

in	<i>BaseAddress</i>	UART base pointer
in	<i>Addr</i>	UART match address
in	<i>IsEnable</i>	Enable or disable address filter function

Returns

void

Definition at line 757 of file AC784xx_Uart_Reg.h.

4.2.4.18 Uart_Reg_SetBaudRateDivisor()

```
LOCAL_INLINE void Uart_Reg_SetBaudRateDivisor (
    UART_Type * Base,
    float32 BaudRateDivisor )
```

This function is used to set baud rate divisor.

Parameters

in	<i>BaseAddress</i>	UART base pointer
in	<i>BaudRateDivisor</i>	the baudRate what will to set

Returns

void

Definition at line 494 of file AC784xx_Uart_Reg.h.

4.2.4.19 Uart_Reg_SetBitCountPerChar()

```
LOCAL_INLINE void Uart_Reg_SetBitCountPerChar (
    UART_Type * Base,
    Uart_PerCharConfigType BitCountPerChar )
```

UART set bit count per char.

Parameters

in	<i>Base</i>	UART base pointer
in	<i>BitCountPerChar</i>	7, 8 or 9 bits per char

Returns

void

Definition at line 535 of file AC784xx_Uart_Reg.h.

4.2.4.20 Uart_Reg_SetCtsRts()

```
LOCAL_INLINE void Uart_Reg_SetCtsRts (
    UART_Type * Base,
    Uart_RtsCtsType RtsCts )
```

Set UART hardware flow control function.

Parameters

in	<i>BaseAddress</i>	UART base pointer
in	<i>RtsCts</i>	Set UART hardware flow control function -UART_RTSCTS_NONE: NO RTS or CTS -UART_RTS_ONLY: RTS only -UART_CTS_ONLY: CTS only -UART_RTSCTS_ALL: Enable RTS and CTS

Returns

void

Definition at line 781 of file AC784xx_Uart_Reg.h.

4.2.4.21 Uart_Reg_SetDataMatch()

```
LOCAL_INLINE void Uart_Reg_SetDataMatch (
    UART_Type * Base,
    uint16 Data,
    boolean IsEnable )
```

Set UART data match function.

Parameters

in	<i>Base</i>	UART base pointer
in	<i>Data</i>	UART match data
in	<i>IsEnable</i>	Enable or disable data match function

Returns

void

Definition at line 736 of file AC784xx_Uart_Reg.h.

4.2.4.22 Uart_Reg_SetErrorInterrupts()

```
LOCAL_INLINE void Uart_Reg_SetErrorInterrupts (
    UART_Type * Base,
    boolean IsEnable )
```

IsEnable or disable the UART error interrupts.

Parameters

in	<i>Base</i>	UART Base pointer
in	<i>IsEnable</i>	set error interrupt

Returns

void

Definition at line 347 of file AC784xx_Uart_Reg.h.

4.2.4.23 Uart_Reg_SetFIFO()

```
LOCAL_INLINE void Uart_Reg_SetFIFO (
    UART_Type * Base,
    boolean IsEnable )
```

IsEnable or disable UART fifo.

Parameters

in	<i>Base</i>	UART base pointer
in	<i>IsEnable</i>	enable or disable UART fifo

Returns

void

Definition at line 602 of file AC784xx_Uart_Reg.h.

4.2.4.24 Uart_Reg_SetIdleInterrupt()

```
LOCAL_INLINE void Uart_Reg_SetIdleInterrupt (
    UART_Type * Base,
    boolean IsEnable )
```

Set UART idle interrupt. Note: Please enable idle function first by called function UART_SetIdleFun, otherwise idle interrupt is invalid.

Parameters

in	<i>Base</i>	UART base pointer
in	<i>IsEnable</i>	IsEnable/disable idle interrupt

Returns

void

Definition at line 378 of file AC784xx_Uart_Reg.h.

4.2.4.25 Uart_Reg_SetInterruptEn()

```
LOCAL_INLINE void Uart_Reg_SetInterruptEn (
    UART_Type * Base,
    uint16 InterruptEn )
```

uart interrupt disable or enable

Parameters

in	<i>UARTx</i>	UART interrupt sources
in	<i>InterruptEn</i>	interrupt en bits

Returns

void

Definition at line 183 of file AC784xx_Uart_Reg.h.

4.2.4.26 Uart_Reg_SetIntMode()

```
LOCAL_INLINE void Uart_Reg_SetIntMode (
    UART_Type * Base,
    Uart_InterruptConfigType IntSrc,
    boolean IsEnable )
```

Set UART interrupt mode.

Parameters

in	<i>Base</i>	UART interrupt sources
in	<i>IntSrc</i>	LINCX reg data
in	<i>IsEnable</i>	TRUE or FALSE

Returns

void

Definition at line 311 of file AC784xx_Uart_Reg.h.

4.2.4.27 Uart_Reg_SetInvRx()

```
LOCAL_INLINE void Uart_Reg_SetInvRx (
    UART_Type * Base,
    boolean Enable )
```

UART inverse polarity for the rx input.

Parameters

in	<i>Base</i>	UART base pointer
in	<i>Enable</i>	enable or disable rx inverse function

Returns

void

Definition at line 950 of file AC784xx_Uart_Reg.h.

4.2.4.28 Uart_Reg_SetInvTx()

```
LOCAL_INLINE void Uart_Reg_SetInvTx (
    UART_Type * Base,
    boolean Enable )
```

UART inverse polarity for tx output.

Parameters

in	<i>Base</i>	UART base pointer
in	<i>Enable</i>	Enable or disable tx inverse function

Returns

void

Definition at line 938 of file AC784xx_Uart_Reg.h.

4.2.4.29 Uart_Reg_SetIrDA()

```
LOCAL_INLINE void Uart_Reg_SetIrDA (
    UART_Type * Base,
    boolean Enable )
```

Enable/disable UART IrDA function.

Parameters

in	<i>BaseAddress</i>	UART base pointer
in	<i>Enable</i>	Enable/disable UART IrDA function

Returns

void

Definition at line 901 of file AC784xx_Uart_Reg.h.

4.2.4.30 Uart_Reg_SetIrDARxWidth()

```
LOCAL_INLINE void Uart_Reg_SetIrDARxWidth (
    UART_Type * Base,
    uint16 Pulse )
```

Set IrDA receiver minimum pulse width.

Parameters

in	<i>BaseAddress</i>	UART base pointer
in	<i>Pulse</i>	IrDA receiver minimum pulse width(range: 0~65535)

Returns

void

Definition at line 913 of file AC784xx_Uart_Reg.h.

4.2.4.31 Uart_Reg_SetIrDATxWidth()

```
LOCAL_INLINE void Uart_Reg_SetIrDATxWidth (
    UART_Type * Base,
    uint8 Pulse )
```

Set IrDA transmit minimum pulse width.

Parameters

in	<i>BaseAddress</i>	UART base pointer
in	<i>Pulse</i>	IrDA receiver minimum pulse width(range: 0~7)

Returns

void

Definition at line 926 of file AC784xx_Uart_Reg.h.

4.2.4.32 Uart_Reg_SetLinBreakLength()

```
LOCAL_INLINE void Uart_Reg_SetLinBreakLength (
    UART_Type * Base,
    uint32 Length )
```

This function is used to set LIN break lenght.

Parameters

in	<i>Base</i>	UART base pointer
in	<i>Length</i>	BRKLGH length

Returns

void

Definition at line 650 of file AC784xx_Uart_Reg.h.

4.2.4.33 Uart_Reg_SetLinBreakThreshold()

```
LOCAL_INLINE void Uart_Reg_SetLinBreakThreshold (
    UART_Type * Base,
    uint32 Length )
```

Set LIN break threshold length for slave detect.

Parameters

in	<i>Base</i>	UART base pointer
in	<i>Length:LIN</i>	break threshold length

Returns

void

Definition at line 638 of file AC784xx_Uart_Reg.h.

4.2.4.34 Uart_Reg_SetLinCtrl()

```
LOCAL_INLINE void Uart_Reg_SetLinCtrl (
    UART_Type * Base,
    uint32 LinCtrl )
```

This function is used to set LIN ctrl.

Parameters

in	<i>Base</i>	UART base pointer
in	<i>LinCtrl</i>	LINCR reg data

Returns

void

Definition at line 662 of file AC784xx_Uart_Reg.h.

4.2.4.35 Uart_Reg_SetLinSleep()

```
LOCAL_INLINE void Uart_Reg_SetLinSleep (
    UART_Type * Base,
    boolean IsEnable )
```

This function is used to set LIN sleep.

Parameters

in	<i>Base</i>	UART interrupt sources
in	<i>IsEnable</i>	TREU or FALSE

Returns

void

Definition at line 674 of file AC784xx_Uart_Reg.h.

4.2.4.36 Uart_Reg_SetLoop()

```
LOCAL_INLINE void Uart_Reg_SetLoop (
    UART_Type * Base,
    boolean Enable )
```

Set UART loop mode.

Parameters

in	<i>base</i>	UART base pointer
in	<i>enable</i>	Enable or disable loop mode

Returns

void

Definition at line 997 of file AC784xx_Uart_Reg.h.

4.2.4.37 Uart_Reg_SetParityMode()

```
LOCAL_INLINE void Uart_Reg_SetParityMode (
    UART_Type * Base,
    Uart_ParityModeConfigType Parity )
```

UART set parity mode.

Parameters

in	<i>Base</i>	UART base pointer
in	<i>Parity</i>	parity disable, odd or even

Returns

void

Definition at line 550 of file AC784xx_Uart_Reg.h.

4.2.4.38 Uart_Reg_SetReceiverCmd()

```
LOCAL_INLINE void Uart_Reg_SetReceiverCmd (
    UART_Type * Base,
    boolean IsEnable )
```

This function is used to set receiver cmd.

Parameters

in	<i>Base</i>	UART base pointer
in	<i>IsEnable</i>	enable or disable set receive cmd

Returns

void

Definition at line 626 of file AC784xx_Uart_Reg.h.

4.2.4.39 Uart_Reg_SetRS485()

```
LOCAL_INLINE void Uart_Reg_SetRS485 (
    UART_Type * Base,
    boolean Enable )
```

Enable or disable RS485 function.

Parameters

in	<i>BaseAddress</i>	UART base pointer
in	<i>Enable</i>	Enable or disable RS485 function -true: enable RS485 function -false: disable RS485 function

Returns

void

Definition at line 824 of file AC784xx_Uart_Reg.h.

4.2.4.40 Uart_Reg_SetRS485Delay()

```
LOCAL_INLINE void Uart_Reg_SetRS485Delay (
    UART_Type * Base,
    boolean Enable )
```

Set RS485 delay function.

Parameters

in	<i>BaseAddress</i>	UART base pointer
in	<i>Enable</i>	enable or disable RS485 delay function -true: enable RS485 delay -false: disable RS485 delay

Returns

void

Definition at line 853 of file AC784xx_Uart_Reg.h.

4.2.4.41 Uart_Reg_SetRS485DelayCnt()

```
LOCAL_INLINE void Uart_Reg_SetRS485DelayCnt (
    UART_Type * Base,
    uint8 Counter )
```

Set RS485 delay time.

Parameters

in	<i>BaseAddress</i>	UART base pointer
in	<i>Counter</i>	RS485 delay time[range 0~255]

Returns

void

Definition at line 865 of file AC784xx_Uart_Reg.h.

4.2.4.42 Uart_Reg_SetRS485GuardEnable()

```
LOCAL_INLINE void Uart_Reg_SetRS485GuardEnable (
    UART_Type * Base,
    boolean Enable )
```

Set RS484 Guart Time enable.

Parameters

in	<i>Base</i>	UART base pointer
in	<i>enable</i>	Enable/disable/ UART RS485 guard time

Returns

none

Definition at line 877 of file AC784xx_Uart_Reg.h.

4.2.4.43 Uart_Reg_SetRS485GuardTime()

```
LOCAL_INLINE void Uart_Reg_SetRS485GuardTime (
    UART_Type * Base,
    uint8 GuardTime )
```

Set RS484 Guart Time.

Parameters

in	<i>base</i>	UART base pointer
in	<i>guardTime</i>	UART set guard time 0 to 15

Returns

none

Definition at line 889 of file AC784xx_Uart_Reg.h.

4.2.4.44 Uart_Reg_SetRS485Invpol()

```
LOCAL_INLINE void Uart_Reg_SetRS485Invpol (
    UART_Type * Base,
    boolean Enable )
```

Set RS485 RTS polarity.

Parameters

in	<i>BaseAddress</i>	UART base pointer
in	<i>Enable</i>	Enable/disable/ UART RS485 RTS invert polarity -true: enable RS485 INVPOL -false: disable RS485 INVPOL

Returns

void

Definition at line 839 of file AC784xx_Uart_Reg.h.

4.2.4.45 Uart_Reg_SetRxDmaCmd()

```
LOCAL_INLINE void Uart_Reg_SetRxDmaCmd (
    UART_Type * Base,
    boolean IsEnable )
```

Configures UART receive DMA request.

Parameters

in	<i>Base</i>	UART Base pointer
in	<i>IsEnable</i>	Receive DMA request configuration (enable: 1/disable: 0)

Returns

void

Definition at line 404 of file AC784xx_Uart_Reg.h.

4.2.4.46 Uart_Reg_SetSampleCounter()

```
LOCAL_INLINE void Uart_Reg_SetSampleCounter (
    UART_Type * Base,
    uint32 SampleCnt )
```

This function is used to set sample counter.

Parameters

in	<i>Base</i>	UART base pointer
in	<i>SampleCnt</i>	Uart_SampleCNTConfigType

Returns

void

Definition at line 471 of file AC784xx_Uart_Reg.h.

4.2.4.47 Uart_Reg_SetSingleWire()

```
LOCAL_INLINE void Uart_Reg_SetSingleWire (
    UART_Type * Base,
    boolean Enable )
```

Set UART single-wire mode.

Parameters

in	<i>Base</i>	UART base pointer
in	<i>Enable</i>	Enable or disable single-wire mode

Returns

void

Definition at line 962 of file AC784xx_Uart_Reg.h.

4.2.4.48 Uart_Reg_SetStopBitCount()

```
LOCAL_INLINE void Uart_Reg_SetStopBitCount (
    UART_Type * Base,
    Uart_StopBitConfigType StopBitCount )
```

Configures the number of stop bits.

Parameters

in	<i>Base</i>	UART base pointer
in	<i>StopBitCount</i>	Number of stop bits

Returns

void

Definition at line 590 of file AC784xx_Uart_Reg.h.

4.2.4.49 Uart_Reg_SetTransmitterCmd()

```
LOCAL_INLINE void Uart_Reg_SetTransmitterCmd (
    UART_Type * Base,
    boolean IsEnable )
```

This function is used to set transmitter cmd.

Parameters

in	<i>Base</i>	UART base pointer
in	<i>IsEnable</i>	enable or disable set transmit cmd

Returns

void

Definition at line 614 of file AC784xx_Uart_Reg.h.

4.2.4.50 Uart_Reg_SetTxDir()

```
LOCAL_INLINE void Uart_Reg_SetTxDir (
    UART_Type * base,
    Uart_TxPinDirType Direction )
```

UART TX pin direction config in single-wire mode.

Parameters

in	<i>Base</i>	UART base pointer
in	<i>Direction</i>	Set tx pin direction in single-wire mode <ul style="list-style-type: none">• UART_TX_PIN_DIR_INPUT• UART_TX_PIN_DIR_OUTPUT

Returns

void

Definition at line 985 of file AC784xx_Uart_Reg.h.

4.2.4.51 Uart_Reg_SetTxDmaCmd()

```
LOCAL_INLINE void Uart_Reg_SetTxDmaCmd (  
    UART_Type * Base,  
    boolean IsEnable )
```

Configures UART transmit DMA request.

Parameters

in	<i>Base</i>	UART Base pointer
in	<i>IsEnable</i>	Transmit DMA request configuration

Returns

void

Definition at line 391 of file AC784xx_Uart_Reg.h.

4.3 Uart_Hal.c File Reference

This file provides Uart hal function extern.

```
#include "AC784xx_Uart_Reg.h"  
#include "Uart_Hal.h"  
#include "Dma_Hal.h"  
#include "Ckgen_Hal.h"  
#include "Rcm_Hal.h"  
#include "Core_Hal.h"  
#include "OsIf_Time.h"  
#include "OsIf_Irq.h"  
#include "OsIf_Critical.h"
```

Classes

- struct [Uart_ChannelStateType](#)
UART runtime state.

Macros

- #define [UART_SAMPLE_CNT_4_VALUE](#) (4U)
- #define [UART_SAMPLE_CNT_8_VALUE](#) (8U)
- #define [UART_SAMPLE_CNT_16_VALUE](#) (16U)
- #define [UART_USE_DMA_TRANSMIT_LEN_MAX](#) (32768U)

Functions

- LOCAL_INLINE void [Uart_Hal_StartTimeout](#) (uint32 *CounterOut, uint32 *TicksOut, uint32 Timeout)
Prepare for timeout checking.
- LOCAL_INLINE boolean [Uart_Hal_CheckTimeout](#) (uint32 *Counter, uint32 *ElapsedTicks, uint32 TimeoutTicks)
Check if the timeout has elapsed.
- void [Uart_Hal_Init](#) (uint8 Instance, const [Uart_ChannelConfigType](#) *ChannelConfigPtr)
Initializes the Uart module.
- void [Uart_Hal_DeInit](#) (uint8 Instance)
Deinitializes the Uart module.
- Hal_StatusType [Uart_Hal_SendData](#) (uint8 Instance, const uint8 *TxBuff, uint32 TxSize)
UART send data using non-blocking method (interrupt or DMA).
- Hal_StatusType [Uart_Hal_SendDataPolling](#) (uint8 Instance, const uint8 *TxBuff, uint32 TxSize)
UART send data using polling mode.
- Hal_StatusType [Uart_Hal_SendDataBlocking](#) (uint8 Instance, const uint8 *TxBuff, uint32 TxSize, uint32 TimeoutUs)
UART send data using blocking method, which will not return until the transmit is complete.
- Hal_StatusType [Uart_Hal_GetSendStatus](#) (uint8 Instance, uint32 *BytesRemaining)
Get send status.
- Hal_StatusType [Uart_Hal_AbortSendingData](#) (uint8 Instance)
Terminates an non-blocking UART transmission early.
- Hal_StatusType [Uart_Hal_ReceiveData](#) (uint8 Instance, uint8 *RxBuff, uint32 RxSize)
UART receive data using non-blocking method(interrupt or DMA).
- Hal_StatusType [Uart_Hal_ReceiveDataPolling](#) (uint8 Instance, uint8 *RxBuff, uint32 RxSize)
UART receive data using polling mode.
- Hal_StatusType [Uart_Hal_ReceiveDataBlocking](#) (uint8 Instance, uint8 *RxBuff, uint32 RxSize, uint32 TimeoutUs)
UART receive data using blocking method, which will not return until the receive is complete.
- Hal_StatusType [Uart_Hal_GetReceiveStatus](#) (uint8 Instance, uint32 *BytesRemaining)
Get receive status.
- Hal_StatusType [Uart_Hal_AbortReceivingData](#) (uint8 Instance)
Terminates a non-blocking receive early.
- Hal_StatusType [Uart_Hal_SetBaudRate](#) (uint8 Instance, uint32 DesiredBaudRate, [Uart_SampleCntType](#) SampleCnt)
Set UART baud rate.
- void [Uart_Hal_GetBaudRate](#) (uint8 Instance, uint32 *ConfiguredBaudRate)
Get UART baudrate.
- void [Uart_Hal_SetIdleInterrupt](#) (uint8 Instance, boolean IsEnable)
Enable or disable the UART idle interrupt.
- void [Uart_Hal_SetDataMatch](#) (uint8 Instance, uint16 Data, boolean IsEnable)
Configure data match interrupt.
- void [Uart_Hal_SetAddrFilter](#) (uint8 Instance, uint16 Addr, boolean IsEnable)

Configure address filter.

- void [Uart_Hal_SetMatchInterrupt](#) (uint8 Instance, boolean IsEnable)
Enable or disable the UART match interrupt.
- void [Uart_Hal_SetCtsRts](#) (uint8 Instance, [Uart_RtsCtsType](#) RtsCts)
Configure the RTS/CTS flow control.
- void [Uart_Hal_SetRS485](#) (uint8 Instance, const [Uart_RS485ConfigType](#) *Config)
Configure RS485 settings.
- void [Uart_Hal_SetIrDA](#) (uint8 Instance, const [Uart_IrDAConfigType](#) *Config)
Configure IrDA settings.

4.3.1 Detailed Description

This file provides Uart hal function extern.

This file provides Uart hal function.

4.3.2 Macro Definition Documentation

4.3.2.1 UART_SAMPLE_CNT_16_VALUE

```
#define UART_SAMPLE_CNT_16_VALUE (16U)
```

Definition at line 58 of file Uart_Hal.c.

4.3.2.2 UART_SAMPLE_CNT_4_VALUE

```
#define UART_SAMPLE_CNT_4_VALUE (4U)
```

Definition at line 56 of file Uart_Hal.c.

4.3.2.3 UART_SAMPLE_CNT_8_VALUE

```
#define UART_SAMPLE_CNT_8_VALUE (8U)
```

Definition at line 57 of file Uart_Hal.c.

4.3.2.4 UART_USE_DMA_TRANSMIT_LEN_MAX

```
#define UART_USE_DMA_TRANSMIT_LEN_MAX (32768U)
```

Definition at line 60 of file Uart_Hal.c.

4.3.3 Function Documentation

4.3.3.1 Uart_Hal_AbortReceivingData()

```
Hal_StatusType Uart_Hal_AbortReceivingData (
    uint8 Instance )
```

Terminates a non-blocking receive early.

Note

Function ID: DES_UART_API_011

Parameters

in	<i>Instance</i>	UART hardware channel ID
----	-----------------	--------------------------

Returns

Hal_StatusType. if successful return STATUS_SUCCESS

Definition at line 1084 of file Uart_Hal.c.

4.3.3.2 Uart_Hal_AbortSendingData()

```
Hal_StatusType Uart_Hal_AbortSendingData (
    uint8 Instance )
```

Terminates an non-blocking UART transmission early.

Note

Function ID: DES_UART_API_010

Parameters

in	<i>Instance</i>	UART hardware channel ID
----	-----------------	--------------------------

Returns

Hal_StatusType: return STATUS_SUCCESS if successful

Definition at line 747 of file Uart_Hal.c.

4.3.3.3 Uart_Hal_CheckTimeout()

```
LOCAL_INLINE boolean Uart_Hal_CheckTimeout (
    uint32 * Counter,
    uint32 * ElapsedTicks,
    uint32 TimeoutTicks )
```

Check if the timeout has elapsed.

Note

Function ID: DES_UART_API_052

Parameters

in, out	<i>Counter</i>	Pointer to counter value, which is updated with the current counter.
in, out	<i>ElapsedTicks</i>	Pointer to store the accumulated elapsed ticks.
in	<i>TimeoutTicks</i>	The total number of ticks representing the timeout.

Returns

TRUE if the timeout has elapsed, FALSE otherwise

Definition at line 1513 of file Uart_Hal.c.

4.3.3.4 Uart_Hal_DeInit()

```
void Uart_Hal_DeInit (
    uint8 Instance )
```

Deinitializes the Uart module.

Note

Function ID: DES_UART_API_001

Parameters

in	<i>Instance</i>	UART hardware channel ID
----	-----------------	--------------------------

Returns

void

Definition at line 476 of file Uart_Hal.c.

4.3.3.5 Uart_Hal_GetBaudRate()

```
void Uart_Hal_GetBaudRate (
    uint8 Instance,
    uint32 * ConfiguredBaudRate )
```

Get UART baudrate.

Note

Function ID: DES_UART_API_013

Parameters

in	<i>Instance</i>	UART hardware channel ID
out	<i>ConfiguredBaudRate</i>	Return the UART configured baudrate

Returns

void

Definition at line 1194 of file Uart_Hal.c.

4.3.3.6 Uart_Hal_GetReceiveStatus()

```
Hal_StatusType Uart_Hal_GetReceiveStatus (
    uint8 Instance,
    uint32 * BytesRemaining )
```

Get receive status.

Note

Function ID: DES_UART_API_009

Parameters

in	<i>Instance</i>	UART hardware channel ID
out	<i>BytesRemaining</i>	Remaining bytes to receive

Returns

Hal_StatusType: Receive status

Definition at line 1040 of file Uart_Hal.c.

4.3.3.7 Uart_Hal_GetSendStatus()

```
Hal_StatusType Uart_Hal_GetSendStatus (
    uint8 Instance,
    uint32 * BytesRemaining )
```

Get send status.

Note

Function ID: DES_UART_API_008

Parameters

in	<i>Instance</i>	UART hardware channel ID
out	<i>BytesRemaining</i>	Remaining bytes to be sent

Returns

Hal_StatusType: Transmit status

Definition at line 703 of file Uart_Hal.c.

4.3.3.8 Uart_Hal_Init()

```
void Uart_Hal_Init (
    uint8 Instance,
    const Uart_ChannelConfigType * ChannelConfigPtr )
```

Initializes the Uart module.

Note

Function ID: DES_UART_API_000

Parameters

in	<i>Instance</i>	UART hardware channel ID
in	<i>ChannelConfigPtr</i>	Pointer to Uart_ChannelConfigType

Returns

void

Definition at line 424 of file Uart_Hal.c.

4.3.3.9 Uart_Hal_ReceiveData()

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

UART receive data using non-blocking method(interrupt or DMA).

Note

Function ID: DES_UART_API_005

Parameters

in	<i>Instance</i>	UART hardware channel ID
out	<i>RxBuff</i>	The rx data buffer pointer
in	<i>RxSize</i>	The rx data buffer bytes size

Returns

Hal_StatusType: if successful return STATUS_SUCCESS or return STATUS_ERROR/STATUS_BUSY.

Definition at line 786 of file Uart_Hal.c.

4.3.3.10 Uart_Hal_ReceiveDataBlocking()

```
Hal_StatusType Uart_Hal_ReceiveDataBlocking (
    uint8 Instance,
    uint8 * RxBuff,
    uint32 RxSize,
    uint32 TimeoutUs )
```

UART receive data using blocking method, which will not return until the receive is complete.

Note

Function ID: DES_UART_API_007

Parameters

in	<i>Instance</i>	UART hardware channel ID
out	<i>RxBuff</i>	The rx data buffer pointer
in	<i>RxSize</i>	The rx data buffer bytes size
in	<i>TimeoutUs</i>	Maximum time in microseconds to wait for the reception to complete

Returns

Hal_StatusType: The receive status

Definition at line 966 of file Uart_Hal.c.

4.3.3.11 Uart_Hal_ReceiveDataPolling()

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

UART receive data using polling mode.

Note

Function ID: DES_UART_API_006

Parameters

in	<i>Instance</i>	UART hardware channel ID
out	<i>RxBuff</i>	The rx data buffer pointer
in	<i>RxSize</i>	The rx data buffer bytes size

Returns

Hal_StatusType: Receive status

Definition at line 838 of file Uart_Hal.c.

4.3.3.12 Uart_Hal_SendData()

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

UART send data using non-blocking method (interrupt or DMA).

Note

Function ID: DES_UART_API_002

Parameters

in	<i>Instance</i>	UART hardware channel ID
in	<i>TxBuff</i>	The tx data buffer pointer
in	<i>TxSize</i>	The tx data buffer bytes size

Returns

Hal_StatusType: if successful return STATUS_SUCCESS or return STATUS_ERROR/STATUS_BUSY.

Definition at line 508 of file Uart_Hal.c.

4.3.3.13 Uart_Hal_SendDataBlocking()

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

UART send data using blocking method, which will not return until the transmit is complete.

Note

Function ID: DES_UART_API_004

Parameters

in	<i>Instance</i>	UART hardware channel ID
in	<i>TxBuff</i>	The tx data buffer pointer
in	<i>TxSize</i>	The tx data buffer bytes size
in	<i>TimeoutUs</i>	Maximum time in microseconds to wait for the transmission

Returns

Hal_StatusType: Transmit status

Definition at line 628 of file Uart_Hal.c.

4.3.3.14 Uart_Hal_SendDataPolling()

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

UART send data using polling mode.

Note

Function ID: DES_UART_API_003

Parameters

in	<i>Instance</i>	UART hardware channel ID
in	<i>TxBuff</i>	The tx data buffer pointer
in	<i>TxSize</i>	The tx data buffer bytes size

Returns

Hal_StatusType: if successful return STATUS_SUCCESS or return STATUS_BUSY.

Definition at line 560 of file Uart_Hal.c.

4.3.3.15 Uart_Hal_SetAddrFilter()

```
void Uart_Hal_SetAddrFilter (
    uint8 Instance,
    uint16 Addr,
    boolean IsEnable )
```

Configure address filter.

Note

Function ID: DES_UART_API_015

Parameters

in	<i>Instance</i>	UART hardware channel ID
in	<i>Addr</i>	The address value to be used for filtering incoming data
in	<i>IsEnable</i>	Use TRUE to enable and FALSE to disable address filter

Returns

void

Definition at line 1287 of file Uart_Hal.c.

4.3.3.16 Uart_Hal_SetBaudRate()

```
Hal_StatusType Uart_Hal_SetBaudRate (
    uint8 Instance,
    uint32 DesiredBaudRate,
    Uart_SampleCntType SampleCnt )
```

Set UART baud rate.

Note

Function ID: DES_UART_API_012

Parameters

in	<i>Instance</i>	UART hardware channel ID
in	<i>DesiredBaudRate</i>	The desired baudrate to be set
in	<i>SampleCnt</i>	The sample count value to be used

Returns

Hal_StatusType: The status of the operation STATUS_SUCCESS or STATUS_BUSY

Definition at line 1124 of file Uart_Hal.c.

4.3.3.17 Uart_Hal_SetCtsRts()

```
void Uart_Hal_SetCtsRts (
    uint8 Instance,
    Uart_RtsCtsType RtsCts )
```

Configure the RTS/CTS flow control.

Note

Function ID: DES_UART_API_014

Parameters

in	<i>Instance</i>	UART hardware channel ID
in	<i>RtsCts</i>	Configuration for RTS/CTS flow control

Returns

void

Definition at line 1323 of file Uart_Hal.c.

4.3.3.18 Uart_Hal_SetDataMatch()

```
void Uart_Hal_SetDataMatch (
    uint8 Instance,
    uint16 Data,
    boolean IsEnable )
```

Configure data match interrupt.

Note

Function ID: DES_UART_API_016

Parameters

in	<i>Instance</i>	UART hardware channel ID
in	<i>Data</i>	The data value to match against received data
in	<i>IsEnable</i>	Use TRUE to enable and FALSE to disable data match

Returns

void

Definition at line 1268 of file Uart_Hal.c.

4.3.3.19 Uart_Hal_SetIdleInterrupt()

```
void Uart_Hal_SetIdleInterrupt (
    uint8 Instance,
    boolean IsEnable )
```

Enable or disable the UART idle interrupt.

Note

Function ID: DES_UART_API_019

Parameters

in	<i>Instance</i>	UART hardware channel ID
in	<i>IsEnable</i>	Use TRUE to enable and FALSE to disable the idle interrupt

Returns

void

Definition at line 1248 of file Uart_Hal.c.

4.3.3.20 Uart_Hal_SetIrDA()

```
void Uart_Hal_SetIrDA (
    uint8 Instance,
    const Uart_IrDAConfigType * Config )
```

Configure IrDA settings.

Note

Function ID: DES_UART_API_017

Parameters

in	<i>Instance</i>	UART hardware channel ID
in	<i>Config</i>	Config Pointer to the IrDA configuration structure

Returns

void

Definition at line 1367 of file Uart_Hal.c.

4.3.3.21 Uart_Hal_SetMatchInterrupt()

```
void Uart_Hal_SetMatchInterrupt (
    uint8 Instance,
    boolean IsEnable )
```

Enable or disable the UART match interrupt.

Note

Function ID: DES_UART_API_020

Parameters

in	<i>Instance</i>	UART hardware channel ID
in	<i>IsEnable</i>	Use TRUE to enable and FALSE to disable match interrupt

Returns

void

Definition at line 1305 of file Uart_Hal.c.

4.3.3.22 Uart_Hal_SetRS485()

```
void Uart_Hal_SetRS485 (
    uint8 Instance,
    const Uart_RS485ConfigType * Config )
```

Configure RS485 settings.

Note

Function ID: DES_UART_API_018

Parameters

in	<i>Instance</i>	UART hardware channel ID
in	<i>Config</i>	Config Pointer to the RS485 configuration structure

Returns

void

Definition at line 1341 of file Uart_Hal.c.

4.3.3.23 Uart_Hal_StartTimeout()

```
LOCAL_INLINE void Uart_Hal_StartTimeout (
    uint32 * CounterOut,
    uint32 * TicksOut,
    uint32 Timeout )
```

Prepare for timeout checking.

Note

Function ID: DES_UART_API_051

Parameters

out	<i>CounterOut</i>	Pointer to store the current counter value.
out	<i>TicksOut</i>	Pointer to store the number of ticks corresponding to the timeout.
in	<i>Timeout</i>	Timeout in microseconds.

Returns

void

Definition at line 1507 of file Uart_Hal.c.

4.4 Uart_Hal.h File Reference

#include "Uart_Hal_Types.h"

Functions

- void [Uart_Hal_Init](#) (uint8 Instance, const [Uart_ChannelConfigType](#) *ChannelConfigPtr)
Initializes the Uart module.
- void [Uart_Hal_DeInit](#) (uint8 Instance)

Deinitializes the Uart module.

- Hal_StatusType [Uart_Hal_SendData](#) (uint8 Instance, const uint8 *TxBuff, uint32 TxSize)
UART send data using non-blocking method (interrupt or DMA).
- Hal_StatusType [Uart_Hal_SendDataPolling](#) (uint8 Instance, const uint8 *TxBuff, uint32 TxSize)
UART send data using polling mode.
- Hal_StatusType [Uart_Hal_SendDataBlocking](#) (uint8 Instance, const uint8 *TxBuff, uint32 TxSize, uint32 TimeoutUs)
UART send data using blocking method, which will not return until the transmit is complete.
- Hal_StatusType [Uart_Hal_GetSendStatus](#) (uint8 Instance, uint32 *BytesRemaining)
Get send status.
- Hal_StatusType [Uart_Hal_AbortSendingData](#) (uint8 Instance)
Terminates an non-blocking UART transmission early.
- Hal_StatusType [Uart_Hal_ReceiveData](#) (uint8 Instance, uint8 *RxBuff, uint32 RxSize)
UART receive data using non-blocking method(interrupt or DMA).
- Hal_StatusType [Uart_Hal_ReceiveDataPolling](#) (uint8 Instance, uint8 *RxBuff, uint32 RxSize)
UART receive data using polling mode.
- Hal_StatusType [Uart_Hal_ReceiveDataBlocking](#) (uint8 Instance, uint8 *RxBuff, uint32 RxSize, uint32 TimeoutUs)
UART receive data using blocking method, which will not return until the receive is complete.
- Hal_StatusType [Uart_Hal_GetReceiveStatus](#) (uint8 Instance, uint32 *BytesRemaining)
Get receive status.
- Hal_StatusType [Uart_Hal_AbortReceivingData](#) (uint8 Instance)
Terminates a non-blocking receive early.
- Hal_StatusType [Uart_Hal_SetBaudRate](#) (uint8 Instance, uint32 DesiredBaudRate, [Uart_SampleCntType](#) SampleCnt)
Set UART baud rate.
- void [Uart_Hal_SetIdleInterrupt](#) (uint8 Instance, boolean IsEnable)
Enable or disable the UART idle interrupt.
- void [Uart_Hal_GetBaudRate](#) (uint8 Instance, uint32 *ConfiguredBaudRate)
Get UART baudrate.
- void [Uart_Hal_SetDataMatch](#) (uint8 Instance, uint16 Data, boolean IsEnable)
Configure data match interrupt.
- void [Uart_Hal_SetAddrFilter](#) (uint8 Instance, uint16 Addr, boolean IsEnable)
Configure address filter.
- void [Uart_Hal_SetMatchInterrupt](#) (uint8 Instance, boolean IsEnable)
Enable or disable the UART match interrupt.
- void [Uart_Hal_SetCtsRts](#) (uint8 Instance, [Uart_RtsCtsType](#) RtsCts)
Configure the RTS/CTS flow control.
- void [Uart_Hal_SetRS485](#) (uint8 Instance, const [Uart_RS485ConfigType](#) *Config)
Configure RS485 settings.
- void [Uart_Hal_SetIrDA](#) (uint8 Instance, const [Uart_IrDAConfigType](#) *Config)
Configure IrDA settings.

4.4.1 Function Documentation

4.4.1.1 Uart_Hal_AbortReceivingData()

```
Hal_StatusType Uart_Hal_AbortReceivingData (
    uint8 Instance )
```

Terminates a non-blocking receive early.

Note

Function ID: DES_UART_API_011

Parameters

in	<i>Instance</i>	UART hardware channel ID
----	-----------------	--------------------------

Returns

Hal_StatusType: return STATUS_SUCCESS if successful

Note

Function ID: DES_UART_API_011

Parameters

in	<i>Instance</i>	UART hardware channel ID
----	-----------------	--------------------------

Returns

Hal_StatusType. if successful return STATUS_SUCCESS

Definition at line 1084 of file Uart_Hal.c.

4.4.1.2 Uart_Hal_AbortSendingData()

```
Hal_StatusType Uart_Hal_AbortSendingData (  
    uint8 Instance )
```

Terminates an non-blocking UART transmission early.

Note

Function ID: DES_UART_API_010

Parameters

in	<i>Instance</i>	UART hardware channel ID
----	-----------------	--------------------------

Returns

Hal_StatusType: return STATUS_SUCCESS if successful

Definition at line 747 of file Uart_Hal.c.

4.4.1.3 Uart_Hal_DeInit()

```
void Uart_Hal_DeInit (  
    uint8 Instance )
```

Deinitializes the Uart module.

Note

Function ID: DES_UART_API_001

Parameters

in	<i>Instance</i>	UART hardware channel ID
----	-----------------	--------------------------

Returns

void

Definition at line 476 of file Uart_Hal.c.

4.4.1.4 Uart_Hal_GetBaudRate()

```
void Uart_Hal_GetBaudRate (
    uint8 Instance,
    uint32 * ConfiguredBaudRate )
```

Get UART baudrate.

Note

Function ID: DES_UART_API_013

Parameters

in	<i>Instance</i>	UART hardware channel ID
out	<i>ConfiguredBaudRate</i>	Return the UART configured baudrate

Returns

void

Definition at line 1194 of file Uart_Hal.c.

4.4.1.5 Uart_Hal_GetReceiveStatus()

```
Hal_StatusType Uart_Hal_GetReceiveStatus (
    uint8 Instance,
    uint32 * BytesRemaining )
```

Get receive status.

Note

Function ID: DES_UART_API_009

Parameters

in	<i>Instance</i>	UART hardware channel ID
out	<i>BytesRemaining</i>	Remaining bytes to receive

Returns

Hal_StatusType: Receive status

Definition at line 1040 of file Uart_Hal.c.

4.4.1.6 Uart_Hal_GetSendStatus()

```
Hal_StatusType Uart_Hal_GetSendStatus (
    uint8 Instance,
    uint32 * BytesRemaining )
```

Get send status.

Note

Function ID: DES_UART_API_008

Parameters

in	<i>Instance</i>	UART hardware channel ID
out	<i>BytesRemaining</i>	Remaining bytes to be sent

Returns

Hal_StatusType: Transmit status

Definition at line 703 of file Uart_Hal.c.

4.4.1.7 Uart_Hal_Init()

```
void Uart_Hal_Init (
    uint8 Instance,
    const Uart_ChannelConfigType * ChannelConfigPtr )
```

Initializes the Uart module.

Note

Function ID: DES_UART_API_000

Parameters

in	<i>Instance</i>	UART hardware channel ID
in	<i>ChannelConfigPtr</i>	Pointer to Uart_ChannelConfigType

Returns

void

Definition at line 424 of file Uart_Hal.c.

4.4.1.8 Uart_Hal_ReceiveData()

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

UART receive data using non-blocking method(interrupt or DMA).

Note

Function ID: DES_UART_API_005

Parameters

in	<i>Instance</i>	UART hardware channel ID
out	<i>RxBuff</i>	The rx data buffer pointer
in	<i>RxSize</i>	The rx data buffer bytes size

Returns

Hal_StatusType: if successful return STATUS_SUCCESS or return STATUS_ERROR/STATUS_BUSY.

Definition at line 786 of file Uart_Hal.c.

4.4.1.9 Uart_Hal_ReceiveDataBlocking()

```
Hal_StatusType Uart_Hal_ReceiveDataBlocking (
    uint8 Instance,
    uint8 * RxBuff,
    uint32 RxSize,
    uint32 TimeoutUs )
```

UART receive data using blocking method, which will not return until the receive is complete.

Note

Function ID: DES_UART_API_007

Parameters

in	<i>Instance</i>	UART hardware channel ID
out	<i>RxBuff</i>	The rx data buffer pointer
in	<i>RxSize</i>	The rx data buffer bytes size
in	<i>TimeoutUs</i>	Maximum time in microseconds to wait for the reception to complete

Returns

Hal_StatusType: The receive status

Definition at line 966 of file Uart_Hal.c.

4.4.1.10 Uart_Hal_ReceiveDataPolling()

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

UART receive data using polling mode.

Note

Function ID: DES_UART_API_006

Parameters

in	<i>Instance</i>	UART hardware channel ID
out	<i>RxBuff</i>	The rx data buffer pointer
in	<i>RxSize</i>	The rx data buffer bytes size

Returns

Hal_StatusType: Receive status

Definition at line 838 of file Uart_Hal.c.

4.4.1.11 Uart_Hal_SendData()

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

UART send data using non-blocking method (interrupt or DMA).

Note

Function ID: DES_UART_API_002

Parameters

in	<i>Instance</i>	UART hardware channel ID
in	<i>TxBuff</i>	The tx data buffer pointer
in	<i>TxSize</i>	The tx data buffer bytes size

Returns

Hal_StatusType: if successful return STATUS_SUCCESS or return STATUS_ERROR/STATUS_BUSY.

Definition at line 508 of file Uart_Hal.c.

4.4.1.12 Uart_Hal_SendDataBlocking()

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

UART send data using blocking method, which will not return until the transmit is complete.

Note

Function ID: DES_UART_API_004

Parameters

in	<i>Instance</i>	UART hardware channel ID
in	<i>TxBuff</i>	The tx data buffer pointer
in	<i>TxSize</i>	The tx data buffer bytes size
in	<i>TimeoutUs</i>	Maximum time in microseconds to wait for the transmission

Returns

Hal_StatusType: Transmit status

Definition at line 628 of file Uart_Hal.c.

4.4.1.13 Uart_Hal_SendDataPolling()

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

UART send data using polling mode.

Note

Function ID: DES_UART_API_003

Parameters

in	<i>Instance</i>	UART hardware channel ID
in	<i>TxBuff</i>	The tx data buffer pointer
in	<i>TxSize</i>	The tx data buffer bytes size

Returns

Hal_StatusType: if successful return STATUS_SUCCESS or return STATUS_BUSY.

Definition at line 560 of file Uart_Hal.c.

4.4.1.14 Uart_Hal_SetAddrFilter()

```
void Uart_Hal_SetAddrFilter (
    uint8 Instance,
    uint16 Addr,
    boolean IsEnable )
```

Configure address filter.

Note

Function ID: DES_UART_API_015

Parameters

in	<i>Instance</i>	UART hardware channel ID
in	<i>Addr</i>	The address value to be used for filtering incoming data
in	<i>IsEnable</i>	Use TRUE to enable and FALSE to disable address filter

Returns

void

Definition at line 1287 of file Uart_Hal.c.

4.4.1.15 Uart_Hal_SetBaudRate()

```
Hal_StatusType Uart_Hal_SetBaudRate (
    uint8 Instance,
    uint32 DesiredBaudRate,
    Uart_SampleCntType SampleCnt )
```

Set UART baud rate.

Note

Function ID: DES_UART_API_012

Parameters

in	<i>Instance</i>	UART hardware channel ID
in	<i>DesiredBaudRate</i>	The desired baudrate to be set
in	<i>SampleCnt</i>	The sample count value to be used

Returns

Hal_StatusType: The status of the operation STATUS_SUCCESS or STATUS_BUSY

Definition at line 1124 of file Uart_Hal.c.

4.4.1.16 Uart_Hal_SetCtsRts()

```
void Uart_Hal_SetCtsRts (
    uint8 Instance,
    Uart_RtsCtsType RtsCts )
```

Configure the RTS/CTS flow control.

Note

Function ID: DES_UART_API_014

Parameters

in	<i>Instance</i>	UART hardware channel ID
in	<i>RtsCts</i>	Configuration for RTS/CTS flow control

Returns

void

Definition at line 1323 of file Uart_Hal.c.

4.4.1.17 Uart_Hal_SetDataMatch()

```
void Uart_Hal_SetDataMatch (
    uint8 Instance,
    uint16 Data,
    boolean IsEnable )
```

Configure data match interrupt.

Note

Function ID: DES_UART_API_016

Parameters

in	<i>Instance</i>	UART hardware channel ID
in	<i>Data</i>	The data value to match against received data
in	<i>IsEnable</i>	Use TRUE to enable and FALSE to disable data match

Returns

void

Definition at line 1268 of file Uart_Hal.c.

4.4.1.18 Uart_Hal_SetIdleInterrupt()

```
void Uart_Hal_SetIdleInterrupt (
    uint8 Instance,
    boolean IsEnable )
```

Enable or disable the UART idle interrupt.

Note

Function ID: DES_UART_API_019

Parameters

in	<i>Instance</i>	UART hardware channel ID
in	<i>IsEnable</i>	Use TRUE to enable and FALSE to disable the idle interrupt

Returns

void

Definition at line 1248 of file Uart_Hal.c.

4.4.1.19 Uart_Hal_SetIrDA()

```
void Uart_Hal_SetIrDA (
    uint8 Instance,
    const Uart_IrDAConfigType * Config )
```

Configure IrDA settings.

Note

Function ID: DES_UART_API_017

Parameters

in	<i>Instance</i>	UART hardware channel ID
in	<i>Config</i>	Config Pointer to the IrDA configuration structure

Returns

void

Definition at line 1367 of file Uart_Hal.c.

4.4.1.20 Uart_Hal_SetMatchInterrupt()

```
void Uart_Hal_SetMatchInterrupt (
    uint8 Instance,
    boolean IsEnable )
```

Enable or disable the UART match interrupt.

Note

Function ID: DES_UART_API_020

Parameters

in	<i>Instance</i>	UART hardware channel ID
in	<i>IsEnable</i>	Use TRUE to enable and FALSE to disable match interrupt

Returns

void

Definition at line 1305 of file Uart_Hal.c.

4.4.1.21 Uart_Hal_SetRS485()

```
void Uart_Hal_SetRS485 (
    uint8 Instance,
    const Uart_RS485ConfigType * Config )
```

Configure RS485 settings.

Note

Function ID: DES_UART_API_018

Parameters

in	<i>Instance</i>	UART hardware channel ID
in	<i>Config</i>	Config Pointer to the RS485 configuration structure

Returns

void

Definition at line 1341 of file Uart_Hal.c.

4.5 Uart_Hal_Types.h File Reference

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

Classes

- struct [Uart_ChannelConfigType](#)
UART configuration structure.
- struct [Uart_RS485ConfigType](#)
Configuration structure for RS485 settings.
- struct [Uart_IrDAConfigType](#)
Configuration structure for IrDA settings.

Typedefs

- typedef void(* [Uart_CallbackType](#)) (uint8 Instance, [Uart_EventType](#) Event)

Enumerations

- enum [Uart_EventType](#) {
[UART_EVENT_RX_FULL](#) = 0x00U, [UART_EVENT_TX_EMPTY](#) = 0x01U, [UART_EVENT_END_TRANSFER](#) = 0x02U, [UART_EVENT_ERROR](#) = 0x03U,
[UART_EVENT_DATA_MATCH](#) = 0x04U, [UART_EVENT_IDLE_LINE](#) = 0x05U }
Define the enum of the events which can trigger UART callback.
- enum [Uart_TransferType](#) { [UART_USING_DMA](#) = 0U, [UART_USING_INTERRUPTS](#) }
no-blocking transfer type.
- enum [Uart_SampleCntType](#) { [UART_SMP_CNT16](#) = 0U, [UART_SMP_CNT8](#), [UART_SMP_CNT4](#) }
UART sample counter enumeration.
- enum [Uart_PerCharConfigType](#) { [UART_7_BITS_PER_CHAR](#) = 0x0U, [UART_8_BITS_PER_CHAR](#) = 0x1U, [UART_9_BITS_PER_CHAR](#) = 0x2U }
UART number of bits in a character.
- enum [Uart_StopBitConfigType](#) { [UART_ONE_STOP_BIT](#) = 0x0U, [UART_TWO_STOP_BIT](#) = 0x1U }
- enum [Uart_ParityModeConfigType](#) { [UART_PARITY_DISABLED](#) = 0x0U, [UART_PARITY_EVEN](#) = 0x1U, [UART_PARITY_ODD](#) = 0x2U }
- enum [Uart_RtsCtsType](#) { [UART_RTSCTS_NONE](#) = 0U, [UART_RTS_ONLY](#), [UART_CTS_ONLY](#), [UART_RTSCTS_ALL](#) }
UART hardware flow enumeration.
- enum [Uart_TxPinDirType](#) { [UART_TX_PIN_DIR_INPUT](#) = 0U, [UART_TX_PIN_DIR_OUTPUT](#) }
UART TX PIN direction enumeration.

4.5.1 Typedef Documentation

4.5.1.1 Uart_CallbackType

```
typedef void(* Uart_CallbackType) (uint8 Instance, Uart_EventType Event)
```

Definition at line 67 of file Uart_Hal_Types.h.

4.5.2 Enumeration Type Documentation

4.5.2.1 Uart_EventType

```
enum Uart_EventType
```

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

Enumerator

UART_EVENT_RX_FULL	Rx buffer received completed
UART_EVENT_TX_EMPTY	Tx buffer transfer completed
UART_EVENT_END_TRANSFER	The current transfer is ending
UART_EVENT_ERROR	An error occurred during transfer
UART_EVENT_DATA_MATCH	Rx data match
UART_EVENT_IDLE_LINE	Idle line

Definition at line 56 of file Uart_Hal_Types.h.

4.5.2.2 Uart_ParityModeConfigType

```
enum Uart_ParityModeConfigType
```

Enumerator

UART_PARITY_DISABLED	parity disabled
UART_PARITY_EVEN	parity enabled, type even
UART_PARITY_ODD	parity enabled, type odd

Definition at line 104 of file Uart_Hal_Types.h.

4.5.2.3 Uart_PerCharConfigType

enum [Uart_PerCharConfigType](#)

UART number of bits in a character.

Enumerator

UART_7_BITS_PER_CHAR	7-bit data characters
UART_8_BITS_PER_CHAR	8-bit data characters
UART_9_BITS_PER_CHAR	9-bit data characters

Definition at line 91 of file Uart_Hal_Types.h.

4.5.2.4 Uart_RtsCtsType

enum [Uart_RtsCtsType](#)

UART hardware flow enumeration.

Enumerator

UART_RTSCTS_NONE	RTS CTS: all disable
UART_RTS_ONLY	RTS only
UART_CTS_ONLY	CTS only
UART_RTSCTS_ALL	RTS CTS: all Enable

Definition at line 114 of file Uart_Hal_Types.h.

4.5.2.5 Uart_SampleCntType

enum [Uart_SampleCntType](#)

UART sample counter enumeration.

Enumerator

UART_SMP_CNT16	SAM_CNT0: based on 16*baud_pulse
UART_SMP_CNT8	SAM_CNT1: based on 8*baud_pulse
UART_SMP_CNT4	SAM_CNT2: based on 4*baud_pulse

Definition at line 81 of file Uart_Hal_Types.h.

4.5.2.6 Uart_StopBitConfigType

```
enum Uart_StopBitConfigType
```

Enumerator

UART_ONE_STOP_BIT	one stop bit
UART_TWO_STOP_BIT	two stop bits

Definition at line 98 of file Uart_Hal_Types.h.

4.5.2.7 Uart_TransferType

```
enum Uart_TransferType
```

no-blocking transfer type.

Enumerator

UART_USING_DMA	Use DMA to perform UART transfer
UART_USING_INTERRUPTS	Use interrupts to perform UART transfer

Definition at line 72 of file Uart_Hal_Types.h.

4.5.2.8 Uart_TxPinDirType

```
enum Uart_TxPinDirType
```

UART TX PIN direction enumeration.

Enumerator

UART_TX_PIN_DIR_INPUT	TX PIN as input in single-wire mode
UART_TX_PIN_DIR_OUTPUT	TX PIN as output in single-wire mode

Definition at line 125 of file Uart_Hal_Types.h.

Index

AC784xx_API_Reference_Manual_UART.pdf, [14](#)

AC784xx_Uart_Reg.h, [14](#)

 UART_DIV_FRAC_VAL, [17](#)

 UART_DIV_H_POS, [17](#)

 UART_DIV_MASK, [17](#)

 UART_IDLE_REG_ID, [17](#)

 UART_IER_REG_ID, [17](#)

 UART_INDEX_MAX, [17](#)

 UART_LINCR_REG_ID, [18](#)

 UART_LSR0_REG_ID, [18](#)

 UART_LSR1_REG_ID, [18](#)

 UART_MATCHCR_REG_ID, [18](#)

 UART_SHIFT, [18](#)

 Uart_InterruptConfigType, [18](#)

 Uart_Reg_ClearErrorFlags, [19](#)

 Uart_Reg_ClearStatusFlag, [20](#)

 Uart_Reg_GetBase, [20](#)

 Uart_Reg_GetBaudRateDivisor, [21](#)

 Uart_Reg_GetChar, [21](#)

 Uart_Reg_GetChar9, [21](#)

 Uart_Reg_GetIntMode, [23](#)

 Uart_Reg_GetLSR0, [23](#)

 Uart_Reg_GetLSR1, [23](#)

 Uart_Reg_GetSampleCounter, [24](#)

 Uart_Reg_GetStatusFlag, [24](#)

 Uart_Reg_IsErrorInterruptEnable, [25](#)

 Uart_Reg_PutChar, [25](#)

 Uart_Reg_PutChar9, [25](#)

 Uart_Reg_ReadLinSleep, [27](#)

 Uart_Reg_SendLINBreak, [27](#)

 Uart_Reg_SetAddrFilter, [27](#)

 Uart_Reg_SetBaudRateDivisor, [28](#)

 Uart_Reg_SetBitCountPerChar, [28](#)

 Uart_Reg_SetCtsRts, [29](#)

 Uart_Reg_SetDataMatch, [29](#)

 Uart_Reg_SetErrorInterrupts, [30](#)

 Uart_Reg_SetFIFO, [30](#)

 Uart_Reg_SetIdleInterrupt, [30](#)

 Uart_Reg_SetIntMode, [31](#)

 Uart_Reg_SetInterruptEn, [31](#)

 Uart_Reg_SetInvRx, [32](#)

 Uart_Reg_SetInvTx, [32](#)

 Uart_Reg_SetLrDARxWidth, [33](#)

 Uart_Reg_SetLrDATxWidth, [33](#)

 Uart_Reg_SetLrDA, [33](#)

 Uart_Reg_SetLinBreakLength, [34](#)

 Uart_Reg_SetLinBreakThreshold, [34](#)

 Uart_Reg_SetLinCtrl, [35](#)

 Uart_Reg_SetLinSleep, [35](#)

 Uart_Reg_SetLoop, [36](#)

 Uart_Reg_SetParityMode, [36](#)

 Uart_Reg_SetRS485, [37](#)

 Uart_Reg_SetRS485Delay, [37](#)

 Uart_Reg_SetRS485DelayCnt, [38](#)

 Uart_Reg_SetRS485GuardEnable, [38](#)

 Uart_Reg_SetRS485GuardTime, [38](#)

 Uart_Reg_SetRS485Invpol, [39](#)

 Uart_Reg_SetReceiverCmd, [36](#)

 Uart_Reg_SetRxDMAcmd, [39](#)

 Uart_Reg_SetSampleCounter, [40](#)

 Uart_Reg_SetSingleWire, [40](#)

 Uart_Reg_SetStopBitCount, [40](#)

 Uart_Reg_SetTransmitterCmd, [41](#)

 Uart_Reg_SetTxDir, [41](#)

 Uart_Reg_SetTxDMAcmd, [42](#)

 Uart_StatusFlagType, [19](#)

BaudRate

 Uart_ChannelConfigType, [3](#)

BitCountPerChar

 Uart_ChannelConfigType, [3](#)

 Uart_ChannelStateType, [6](#)

DelayCnt

 Uart_RS485ConfigType, [12](#)

DelayEnable

 Uart_RS485ConfigType, [12](#)

Enable

 Uart_IrDAConfigType, [10](#)

 Uart_RS485ConfigType, [12](#)

GuardTime

 Uart_RS485ConfigType, [12](#)

InitState

 Uart_ChannelStateType, [6](#)

InvpolEnable

 Uart_RS485ConfigType, [13](#)

IsRxBlocking

 Uart_ChannelStateType, [6](#)

IsRxBusy

 Uart_ChannelStateType, [7](#)

IsTxBlocking

 Uart_ChannelStateType, [7](#)

IsTxBusy

 Uart_ChannelStateType, [7](#)

ParityMode

 Uart_ChannelConfigType, [4](#)

ReceiveStatus

 Uart_ChannelStateType, [7](#)

RxBuff

 Uart_ChannelStateType, [7](#)

- RxCallback
 - Uart_ChannelConfigType, [4](#)
 - Uart_ChannelStateType, [8](#)
- RxComplete
 - Uart_ChannelStateType, [8](#)
- RxDmaChannel
 - Uart_ChannelConfigType, [4](#)
 - Uart_ChannelStateType, [8](#)
- RxSize
 - Uart_ChannelStateType, [8](#)
- RxWidth
 - Uart_IrDAConfigType, [11](#)
- SampleCnt
 - Uart_ChannelConfigType, [4](#)
- SendStatus
 - Uart_ChannelStateType, [8](#)
- StopBitCount
 - Uart_ChannelConfigType, [4](#)
- TransferType
 - Uart_ChannelConfigType, [5](#)
 - Uart_ChannelStateType, [9](#)
- TxBuff
 - Uart_ChannelStateType, [9](#)
- TxCallback
 - Uart_ChannelConfigType, [5](#)
 - Uart_ChannelStateType, [9](#)
- TxComplete
 - Uart_ChannelStateType, [9](#)
- TxDmaChannel
 - Uart_ChannelConfigType, [5](#)
 - Uart_ChannelStateType, [9](#)
- TxMode
 - Uart_IrDAConfigType, [11](#)
- TxSize
 - Uart_ChannelStateType, [10](#)
- TxWidth
 - Uart_IrDAConfigType, [11](#)
- UART_DIV_FRAC_VAL
 - AC784xx_Uart_Reg.h, [17](#)
- UART_DIV_H_POS
 - AC784xx_Uart_Reg.h, [17](#)
- UART_DIV_MASK
 - AC784xx_Uart_Reg.h, [17](#)
- UART_IDLE_REG_ID
 - AC784xx_Uart_Reg.h, [17](#)
- UART_IER_REG_ID
 - AC784xx_Uart_Reg.h, [17](#)
- UART_INDEX_MAX
 - AC784xx_Uart_Reg.h, [17](#)
- UART_LINCR_REG_ID
 - AC784xx_Uart_Reg.h, [18](#)
- UART_LSR0_REG_ID
 - AC784xx_Uart_Reg.h, [18](#)
- UART_LSR1_REG_ID
 - AC784xx_Uart_Reg.h, [18](#)
- UART_MATCHCR_REG_ID
 - AC784xx_Uart_Reg.h, [18](#)
- UART_SAMPLE_CNT_16_VALUE
- Uart_Hal.c, [44](#)
- UART_SAMPLE_CNT_4_VALUE
 - Uart_Hal.c, [44](#)
- UART_SAMPLE_CNT_8_VALUE
 - Uart_Hal.c, [44](#)
- UART_SHIFT
 - AC784xx_Uart_Reg.h, [18](#)
- UART_USE_DMA_TRANSMIT_LEN_MAX
 - Uart_Hal.c, [44](#)
- Uart_CallbackType
 - Uart_Hal_Types.h, [69](#)
- Uart_ChannelConfigType, [3](#)
 - BaudRate, [3](#)
 - BitCountPerChar, [3](#)
 - ParityMode, [4](#)
 - RxCallback, [4](#)
 - RxDmaChannel, [4](#)
 - SampleCnt, [4](#)
 - StopBitCount, [4](#)
 - TransferType, [5](#)
 - TxCallback, [5](#)
 - TxDmaChannel, [5](#)
- Uart_ChannelStateType, [5](#)
 - BitCountPerChar, [6](#)
 - InitState, [6](#)
 - IsRxBlocking, [6](#)
 - IsRxBusy, [7](#)
 - IsTxBlocking, [7](#)
 - IsTxBusy, [7](#)
 - ReceiveStatus, [7](#)
 - RxBuff, [7](#)
 - RxCallback, [8](#)
 - RxComplete, [8](#)
 - RxDmaChannel, [8](#)
 - RxSize, [8](#)
 - SendStatus, [8](#)
 - TransferType, [9](#)
 - TxBuff, [9](#)
 - TxCallback, [9](#)
 - TxComplete, [9](#)
 - TxDmaChannel, [9](#)
 - TxSize, [10](#)
- Uart_EventType
 - Uart_Hal_Types.h, [69](#)
- Uart_Hal.c, [42](#)
 - UART_SAMPLE_CNT_16_VALUE, [44](#)
 - UART_SAMPLE_CNT_4_VALUE, [44](#)
 - UART_SAMPLE_CNT_8_VALUE, [44](#)
 - UART_USE_DMA_TRANSMIT_LEN_MAX, [44](#)
 - Uart_Hal_AbortReceivingData, [45](#)
 - Uart_Hal_AbortSendingData, [45](#)
 - Uart_Hal_CheckTimeout, [45](#)
 - Uart_Hal_DeInit, [46](#)
 - Uart_Hal_GetBaudRate, [46](#)
 - Uart_Hal_GetReceiveStatus, [47](#)
 - Uart_Hal_GetSendStatus, [47](#)
 - Uart_Hal_Init, [48](#)
 - Uart_Hal_ReceiveData, [48](#)
 - Uart_Hal_ReceiveDataBlocking, [49](#)
 - Uart_Hal_ReceiveDataPolling, [50](#)
 - Uart_Hal_SendData, [50](#)

- Uart_Hal_SendDataBlocking, 51
- Uart_Hal_SendDataPolling, 51
- Uart_Hal_SetAddrFilter, 52
- Uart_Hal_SetBaudRate, 52
- Uart_Hal_SetCtsRts, 53
- Uart_Hal_SetDataMatch, 53
- Uart_Hal_SetIdleInterrupt, 54
- Uart_Hal_SetIrDA, 54
- Uart_Hal_SetMatchInterrupt, 55
- Uart_Hal_SetRS485, 55
- Uart_Hal_StartTimeout, 56
- Uart_Hal.h, 56
 - Uart_Hal_AbortReceivingData, 57
 - Uart_Hal_AbortSendingData, 58
 - Uart_Hal_DeInit, 58
 - Uart_Hal_GetBaudRate, 59
 - Uart_Hal_GetReceiveStatus, 59
 - Uart_Hal_GetSendStatus, 60
 - Uart_Hal_Init, 60
 - Uart_Hal_ReceiveData, 61
 - Uart_Hal_ReceiveDataBlocking, 61
 - Uart_Hal_ReceiveDataPolling, 62
 - Uart_Hal_SendData, 62
 - Uart_Hal_SendDataBlocking, 63
 - Uart_Hal_SendDataPolling, 63
 - Uart_Hal_SetAddrFilter, 64
 - Uart_Hal_SetBaudRate, 64
 - Uart_Hal_SetCtsRts, 65
 - Uart_Hal_SetDataMatch, 65
 - Uart_Hal_SetIdleInterrupt, 66
 - Uart_Hal_SetIrDA, 66
 - Uart_Hal_SetMatchInterrupt, 67
 - Uart_Hal_SetRS485, 67
- Uart_Hal_AbortReceivingData
 - Uart_Hal.c, 45
 - Uart_Hal.h, 57
- Uart_Hal_AbortSendingData
 - Uart_Hal.c, 45
 - Uart_Hal.h, 58
- Uart_Hal_CheckTimeout
 - Uart_Hal.c, 45
- Uart_Hal_DeInit
 - Uart_Hal.c, 46
 - Uart_Hal.h, 58
- Uart_Hal_GetBaudRate
 - Uart_Hal.c, 46
 - Uart_Hal.h, 59
- Uart_Hal_GetReceiveStatus
 - Uart_Hal.c, 47
 - Uart_Hal.h, 59
- Uart_Hal_GetSendStatus
 - Uart_Hal.c, 47
 - Uart_Hal.h, 60
- Uart_Hal_Init
 - Uart_Hal.c, 48
 - Uart_Hal.h, 60
- Uart_Hal_ReceiveData
 - Uart_Hal.c, 48
 - Uart_Hal.h, 61
- Uart_Hal_ReceiveDataBlocking
 - Uart_Hal.c, 49
- Uart_Hal.h, 61
- Uart_Hal_ReceiveDataPolling
 - Uart_Hal.c, 50
 - Uart_Hal.h, 62
- Uart_Hal_SendData
 - Uart_Hal.c, 50
 - Uart_Hal.h, 62
- Uart_Hal_SendDataBlocking
 - Uart_Hal.c, 51
 - Uart_Hal.h, 63
- Uart_Hal_SendDataPolling
 - Uart_Hal.c, 51
 - Uart_Hal.h, 63
- Uart_Hal_SetAddrFilter
 - Uart_Hal.c, 52
 - Uart_Hal.h, 64
- Uart_Hal_SetBaudRate
 - Uart_Hal.c, 52
 - Uart_Hal.h, 64
- Uart_Hal_SetCtsRts
 - Uart_Hal.c, 53
 - Uart_Hal.h, 65
- Uart_Hal_SetDataMatch
 - Uart_Hal.c, 53
 - Uart_Hal.h, 65
- Uart_Hal_SetIdleInterrupt
 - Uart_Hal.c, 54
 - Uart_Hal.h, 66
- Uart_Hal_SetIrDA
 - Uart_Hal.c, 54
 - Uart_Hal.h, 66
- Uart_Hal_SetMatchInterrupt
 - Uart_Hal.c, 55
 - Uart_Hal.h, 67
- Uart_Hal_SetRS485
 - Uart_Hal.c, 55
 - Uart_Hal.h, 67
- Uart_Hal_StartTimeout
 - Uart_Hal.c, 56
- Uart_Hal_Types.h, 68
 - Uart_CallbackType, 69
 - Uart_EventType, 69
 - Uart_ParityModeConfigType, 69
 - Uart_PerCharConfigType, 69
 - Uart_RtsCtsType, 70
 - Uart_SampleCntType, 70
 - Uart_StopBitConfigType, 70
 - Uart_TransferType, 71
 - Uart_TxPinDirType, 71
- Uart_InterruptConfigType
 - AC784xx_Uart_Reg.h, 18
- Uart_IrDAConfigType, 10
 - Enable, 10
 - RxWidth, 11
 - TxMode, 11
 - TxWidth, 11
- Uart_ParityModeConfigType
 - Uart_Hal_Types.h, 69
- Uart_PerCharConfigType
 - Uart_Hal_Types.h, 69
- Uart_RS485ConfigType, 11

- DelayCnt, [12](#)
- DelayEnable, [12](#)
- Enable, [12](#)
- GuardTime, [12](#)
- InvpolEnable, [13](#)
- Uart_Reg_ClearErrorFlags
 - AC784xx_Uart_Reg.h, [19](#)
- Uart_Reg_ClearStatusFlag
 - AC784xx_Uart_Reg.h, [20](#)
- Uart_Reg_GetBase
 - AC784xx_Uart_Reg.h, [20](#)
- Uart_Reg_GetBaudRateDivisor
 - AC784xx_Uart_Reg.h, [21](#)
- Uart_Reg_GetChar
 - AC784xx_Uart_Reg.h, [21](#)
- Uart_Reg_GetChar9
 - AC784xx_Uart_Reg.h, [21](#)
- Uart_Reg_GetIntMode
 - AC784xx_Uart_Reg.h, [23](#)
- Uart_Reg_GetLSR0
 - AC784xx_Uart_Reg.h, [23](#)
- Uart_Reg_GetLSR1
 - AC784xx_Uart_Reg.h, [23](#)
- Uart_Reg_GetSampleCounter
 - AC784xx_Uart_Reg.h, [24](#)
- Uart_Reg_GetStatusFlag
 - AC784xx_Uart_Reg.h, [24](#)
- Uart_Reg_IsErrorInterruptEnable
 - AC784xx_Uart_Reg.h, [25](#)
- Uart_Reg_PutChar
 - AC784xx_Uart_Reg.h, [25](#)
- Uart_Reg_PutChar9
 - AC784xx_Uart_Reg.h, [25](#)
- Uart_Reg_ReadLinSleep
 - AC784xx_Uart_Reg.h, [27](#)
- Uart_Reg_SendLINBreak
 - AC784xx_Uart_Reg.h, [27](#)
- Uart_Reg_SetAddrFilter
 - AC784xx_Uart_Reg.h, [27](#)
- Uart_Reg_SetBaudRateDivisor
 - AC784xx_Uart_Reg.h, [28](#)
- Uart_Reg_SetBitCountPerChar
 - AC784xx_Uart_Reg.h, [28](#)
- Uart_Reg_SetCtsRts
 - AC784xx_Uart_Reg.h, [29](#)
- Uart_Reg_SetDataMatch
 - AC784xx_Uart_Reg.h, [29](#)
- Uart_Reg_SetErrorInterrupts
 - AC784xx_Uart_Reg.h, [30](#)
- Uart_Reg_SetFIFO
 - AC784xx_Uart_Reg.h, [30](#)
- Uart_Reg_SetIdleInterrupt
 - AC784xx_Uart_Reg.h, [30](#)
- Uart_Reg_SetIntMode
 - AC784xx_Uart_Reg.h, [31](#)
- Uart_Reg_SetInterruptEn
 - AC784xx_Uart_Reg.h, [31](#)
- Uart_Reg_SetInvRx
 - AC784xx_Uart_Reg.h, [32](#)
- Uart_Reg_SetInvTx
 - AC784xx_Uart_Reg.h, [32](#)
- Uart_Reg_SetIrDARxWidth
 - AC784xx_Uart_Reg.h, [33](#)
- Uart_Reg_SetIrDATxWidth
 - AC784xx_Uart_Reg.h, [33](#)
- Uart_Reg_SetIrDA
 - AC784xx_Uart_Reg.h, [33](#)
- Uart_Reg_SetLinBreakLength
 - AC784xx_Uart_Reg.h, [34](#)
- Uart_Reg_SetLinBreakThreshold
 - AC784xx_Uart_Reg.h, [34](#)
- Uart_Reg_SetLinCtrl
 - AC784xx_Uart_Reg.h, [35](#)
- Uart_Reg_SetLinSleep
 - AC784xx_Uart_Reg.h, [35](#)
- Uart_Reg_SetLoop
 - AC784xx_Uart_Reg.h, [36](#)
- Uart_Reg_SetParityMode
 - AC784xx_Uart_Reg.h, [36](#)
- Uart_Reg_SetRS485
 - AC784xx_Uart_Reg.h, [37](#)
- Uart_Reg_SetRS485Delay
 - AC784xx_Uart_Reg.h, [37](#)
- Uart_Reg_SetRS485DelayCnt
 - AC784xx_Uart_Reg.h, [38](#)
- Uart_Reg_SetRS485GuardEnable
 - AC784xx_Uart_Reg.h, [38](#)
- Uart_Reg_SetRS485GuardTime
 - AC784xx_Uart_Reg.h, [38](#)
- Uart_Reg_SetRS485Invpol
 - AC784xx_Uart_Reg.h, [39](#)
- Uart_Reg_SetReceiverCmd
 - AC784xx_Uart_Reg.h, [36](#)
- Uart_Reg_SetRxDmaCmd
 - AC784xx_Uart_Reg.h, [39](#)
- Uart_Reg_SetSampleCounter
 - AC784xx_Uart_Reg.h, [40](#)
- Uart_Reg_SetSingleWire
 - AC784xx_Uart_Reg.h, [40](#)
- Uart_Reg_SetStopBitCount
 - AC784xx_Uart_Reg.h, [40](#)
- Uart_Reg_SetTransmitterCmd
 - AC784xx_Uart_Reg.h, [41](#)
- Uart_Reg_SetTxDir
 - AC784xx_Uart_Reg.h, [41](#)
- Uart_Reg_SetTxDmaCmd
 - AC784xx_Uart_Reg.h, [42](#)
- Uart_RtsCtsType
 - Uart_Hal_Types.h, [70](#)
- Uart_SampleCntType
 - Uart_Hal_Types.h, [70](#)
- Uart_StatusFlagType
 - AC784xx_Uart_Reg.h, [19](#)
- Uart_StopBitConfigType
 - Uart_Hal_Types.h, [70](#)
- Uart_TransferType
 - Uart_Hal_Types.h, [71](#)
- Uart_TxPinDirType
 - Uart_Hal_Types.h, [71](#)