

AC784xx\_DFP LIN

5.1.0

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	Lin_ChannelConfigType Struct Reference . . . . .	3
3.1.1	Detailed Description . . . . .	3
3.1.2	Member Data Documentation . . . . .	3
3.1.2.1	AutoBaudEnable . . . . .	3
3.1.2.2	BaudRate . . . . .	4
3.1.2.3	BreakLength . . . . .	4
3.1.2.4	BreakThreshold . . . . .	4
3.1.2.5	Callback . . . . .	4
3.1.2.6	ModeType . . . . .	4
3.2	Lin_ChannelInfoType Struct Reference . . . . .	5
3.2.1	Detailed Description . . . . .	5
3.2.2	Member Data Documentation . . . . .	5
3.2.2.1	CurrentEventId . . . . .	5
3.2.2.2	CurrentPid . . . . .	5
3.2.2.3	RxBuff . . . . .	5
3.3	Lin_ChannelStateType Struct Reference . . . . .	6
3.3.1	Detailed Description . . . . .	6
3.3.2	Member Data Documentation . . . . .	6

3.3.2.1	Callback	6
3.3.2.2	ChannelInfo	6
3.3.2.3	Checksum	7
3.3.2.4	CntByte	7
3.3.2.5	CurrentNodeState	7
3.3.2.6	IsBusBusy	7
3.3.2.7	PreviousNodeState	7
3.3.2.8	RxSize	8
3.3.2.9	TimeoutCounter	8
3.3.2.10	TxBuff	8
3.3.2.11	TxSize	8
3.4	Lin_PduType Struct Reference	8
3.4.1	Detailed Description	9
3.4.2	Member Data Documentation	9
3.4.2.1	Cs	9
3.4.2.2	DI	9
3.4.2.3	Drc	9
3.4.2.4	Pid	10
3.4.2.5	SduPtr	10
<b>4</b>	<b>File Documentation</b>	<b>11</b>
4.1	AC784xx_API_Reference_Manual_LIN.pdf File Reference	11
4.2	Lin_Hal.c File Reference	11
4.2.1	Detailed Description	12
4.2.2	Macro Definition Documentation	12
4.2.2.1	LIN_DATA_LENGTH_8	12
4.2.2.2	LIN_ID_MASK	12
4.2.2.3	LIN_MAX_CHANNEL_BAUDRATE	13
4.2.2.4	LIN_MIN_CHANNEL_BAUDRATE	13
4.2.2.5	LIN_SAMPLE_CNT_16_VALUE	13
4.2.2.6	LIN_SMP_CNT16	13
4.2.2.7	LIN_SYNC_DATA	13

4.2.3	Function Documentation	13
4.2.3.1	Lin_Hal_AbortTransferData()	13
4.2.3.2	Lin_Hal_DeInit()	14
4.2.3.3	Lin_Hal_GetStatus()	14
4.2.3.4	Lin_Hal_GoToIdleState()	15
4.2.3.5	Lin_Hal_GoToSleepMode()	15
4.2.3.6	Lin_Hal_Init()	16
4.2.3.7	Lin_Hal_ProcessParity()	16
4.2.3.8	Lin_Hal_SendFrameData()	17
4.2.3.9	Lin_Hal_SendWakeupSignal()	17
4.2.3.10	Lin_Hal_SetTimeoutCounter()	18
4.2.3.11	Lin_Hal_TimeoutService()	18
4.3	Lin_Hal.h File Reference	18
4.3.1	Detailed Description	19
4.3.2	Function Documentation	19
4.3.2.1	Lin_Hal_AbortTransferData()	19
4.3.2.2	Lin_Hal_DeInit()	20
4.3.2.3	Lin_Hal_GetStatus()	20
4.3.2.4	Lin_Hal_GoToIdleState()	21
4.3.2.5	Lin_Hal_GoToSleepMode()	21
4.3.2.6	Lin_Hal_Init()	22
4.3.2.7	Lin_Hal_ProcessParity()	22
4.3.2.8	Lin_Hal_SendFrameData()	23
4.3.2.9	Lin_Hal_SendWakeupSignal()	23
4.3.2.10	Lin_Hal_SetTimeoutCounter()	24
4.3.2.11	Lin_Hal_TimeoutService()	24
4.4	Lin_Hal_Types.h File Reference	24
4.4.1	Detailed Description	26
4.4.2	Macro Definition Documentation	26
4.4.2.1	LIN_CHECK_PARITY	26
4.4.2.2	LIN_MAKE_PARITY	26
4.4.3	Typedef Documentation	26
4.4.3.1	Lin_CallbackType	26
4.4.4	Enumeration Type Documentation	26
4.4.4.1	Lin_BreakLengthType	26
4.4.4.2	Lin_BreakThresholdType	27
4.4.4.3	Lin_EventIdType	28
4.4.4.4	Lin_FrameCsModelType	28
4.4.4.5	Lin_FrameResponseType	28
4.4.4.6	Lin_ModeType	29
4.4.4.7	Lin_NodeStateType	29
4.4.4.8	Lin_StatusType	29

# Chapter 1

## Class Index

### 1.1 Class List

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

<a href="#">Lin_ChannelConfigType</a>	Configuration struction of the LIN driver . . . . .	3
<a href="#">Lin_ChannelInfoType</a>	LIN channel info structure . . . . .	5
<a href="#">Lin_ChannelStateType</a>	LIN current state structure . . . . .	6
<a href="#">Lin_PduType</a>	LIN frame type used to provide PID,checksum model, data length and SDU pointer . . . . .	8

# Chapter 2

## File Index

### 2.1 File List

Here is a list of all files with brief descriptions:

<a href="#">AC784xx_API_Reference_Manual_LIN.pdf</a>	11
<a href="#">Lin_Hal.c</a>	
This file provides Hal lin api	11
<a href="#">Lin_Hal.h</a>	
This file provides extern Hal lin api	18
<a href="#">Lin_Hal_Types.h</a>	
This file provides lin module used hardware Types	24

## Chapter 3

# Class Documentation

### 3.1 Lin\_ChannelConfigType Struct Reference

Configuration struction of the LIN driver.

```
#include <Lin_Hal_Types.h>
```

#### Public Attributes

- uint32 [BaudRate](#)
- boolean [AutoBaudEnable](#)
- [Lin\\_ModeType](#) [ModeType](#)
- [Lin\\_BreakLengthType](#) [BreakLength](#)
- [Lin\\_BreakThresholdType](#) [BreakThreshold](#)
- [Lin\\_CallbackType](#) [Callback](#)

#### 3.1.1 Detailed Description

Configuration struction of the LIN driver.

Definition at line 211 of file `Lin_Hal_Types.h`.

#### 3.1.2 Member Data Documentation

##### 3.1.2.1 AutoBaudEnable

```
boolean Lin_ChannelConfigType::AutoBaudEnable
```

Autobaud function enable

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

### 3.1.2.2 BaudRate

`uint32 Lin_ChannelConfigType::BaudRate`

LIN Baudrate value

Definition at line 213 of file `Lin_Hal_Types.h`.

### 3.1.2.3 BreakLength

`Lin_BreakLengthType Lin_ChannelConfigType::BreakLength`

LIN break length for master

Definition at line 216 of file `Lin_Hal_Types.h`.

### 3.1.2.4 BreakThreshold

`Lin_BreakThresholdType Lin_ChannelConfigType::BreakThreshold`

LIN break detect threshold for slave

Definition at line 217 of file `Lin_Hal_Types.h`.

### 3.1.2.5 Callback

`Lin_CallbackType Lin_ChannelConfigType::Callback`

Callback funtion

Definition at line 218 of file `Lin_Hal_Types.h`.

### 3.1.2.6 ModeType

`Lin_ModeType Lin_ChannelConfigType::ModeType`

Node mode as Master or Slave

Definition at line 215 of file `Lin_Hal_Types.h`.

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

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



## 3.2 Lin\_ChannelInfoType Struct Reference

LIN channel info structure.

```
#include <Lin_Hal_Types.h>
```

### Public Attributes

- `uint8 * RxBuff`
- `uint8 CurrentPid`
- `Lin_EventIdType CurrentEventId`

### 3.2.1 Detailed Description

LIN channel info structure.

Definition at line 196 of file `Lin_Hal_Types.h`.

### 3.2.2 Member Data Documentation

#### 3.2.2.1 CurrentEventId

```
Lin_EventIdType Lin_ChannelInfoType::CurrentEventId
```

Current event ID

Definition at line 200 of file `Lin_Hal_Types.h`.

#### 3.2.2.2 CurrentPid

```
uint8 Lin_ChannelInfoType::CurrentPid
```

Current PID

Definition at line 199 of file `Lin_Hal_Types.h`.

#### 3.2.2.3 RxBuff

```
uint8* Lin_ChannelInfoType::RxBuff
```

The buffer of received data

Definition at line 198 of file `Lin_Hal_Types.h`.

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

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

## 3.3 Lin\_ChannelStateType Struct Reference

LIN current state structure.

### Public Attributes

- const uint8 \* [TxBuff](#)
- uint8 [CntByte](#)
- uint8 [TxSize](#)
- uint8 [RxSize](#)
- uint8 [Checksum](#)
- boolean [IsBusBusy](#)
- volatile uint32 [TimeoutCounter](#)
- [Lin\\_NodeStateType](#) [CurrentNodeState](#)
- [Lin\\_NodeStateType](#) [PreviousNodeState](#)
- [Lin\\_CallbackType](#) [Callback](#)
- [Lin\\_ChannelInfoType](#) [ChannelInfo](#)

### 3.3.1 Detailed Description

LIN current state structure.

Definition at line 110 of file [Lin\\_Hal.c](#).

### 3.3.2 Member Data Documentation

#### 3.3.2.1 Callback

[Lin\\_CallbackType](#) [Lin\\_ChannelStateType::Callback](#)

Callback function

Definition at line 122 of file [Lin\\_Hal.c](#).

#### 3.3.2.2 ChannelInfo

[Lin\\_ChannelInfoType](#) [Lin\\_ChannelStateType::ChannelInfo](#)

LIN channel information

Definition at line 123 of file [Lin\\_Hal.c](#).

### 3.3.2.3 CheckSum

```
uint8 Lin_ChannelStateType::Checksum
```

Checksum byte

Definition at line 116 of file Lin\_Hal.c.

### 3.3.2.4 CntByte

```
uint8 Lin_ChannelStateType::CntByte
```

The count size of bytes already transmitted or received

Definition at line 113 of file Lin\_Hal.c.

### 3.3.2.5 CurrentNodeState

```
Lin_NodeStateType Lin_ChannelStateType::CurrentNodeState
```

Current node state

Definition at line 119 of file Lin\_Hal.c.

### 3.3.2.6 IsBusBusy

```
boolean Lin_ChannelStateType::IsBusBusy
```

Bus busy state

Definition at line 117 of file Lin\_Hal.c.

### 3.3.2.7 PreviousNodeState

```
Lin_NodeStateType Lin_ChannelStateType::PreviousNodeState
```

Store previous node state when set Lin channel to idle for further processing

Definition at line 121 of file Lin\_Hal.c.

### 3.3.2.8 RxSize

```
uint8 Lin_ChannelStateType::RxSize
```

Byte size need to be received

Definition at line 115 of file Lin\_Hal.c.

### 3.3.2.9 TimeoutCounter

```
volatile uint32 Lin_ChannelStateType::TimeoutCounter
```

Timeout counter value

Definition at line 118 of file Lin\_Hal.c.

### 3.3.2.10 TxBuff

```
const uint8* Lin_ChannelStateType::TxBuff
```

The buffer of transmitted data

Definition at line 112 of file Lin\_Hal.c.

### 3.3.2.11 TxSize

```
uint8 Lin_ChannelStateType::TxSize
```

Byte size need to be transmitted

Definition at line 114 of file Lin\_Hal.c.

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

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

## 3.4 Lin\_PduType Struct Reference

LIN frame type used to provide PID,checksum model, data length and SDU pointer.

```
#include <Lin_Hal_Types.h>
```

## Public Attributes

- uint8 [Pid](#)
- [Lin\\_FrameCsModelType](#) Cs
- [Lin\\_FrameResponseType](#) Drc
- uint8 [Dl](#)
- uint8 \* [SduPtr](#)

### 3.4.1 Detailed Description

LIN frame type used to provide PID,checksum model, data length and SDU pointer.

Definition at line 224 of file [Lin\\_Hal\\_Types.h](#).

### 3.4.2 Member Data Documentation

#### 3.4.2.1 Cs

```
Lin\_FrameCsModelType Lin_PduType::Cs
```

Checksum model type.

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

#### 3.4.2.2 Dl

```
uint8 Lin_PduType::Dl
```

Data length.

Definition at line 229 of file [Lin\\_Hal\\_Types.h](#).

#### 3.4.2.3 Drc

```
Lin\_FrameResponseType Lin_PduType::Drc
```

Response type.

Definition at line 228 of file [Lin\\_Hal\\_Types.h](#).

#### 3.4.2.4 Pid

```
uint8 Lin_PduType::Pid
```

LIN frame identifier.

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

#### 3.4.2.5 SduPtr

```
uint8* Lin_PduType::SduPtr
```

Pointer to Sdu(Servie Data Unit).

Definition at line 230 of file Lin\_Hal\_Types.h.

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

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

## Chapter 4

# File Documentation

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

### 4.2 Lin\_Hal.c File Reference

This file provides Hal lin api.

```
#include "AC784xx_Uart_Reg.h"
#include "Lin_Hal.h"
#include "Ckgen_Hal.h"
#include "Rcm_Hal.h"
#include "Core_Hal.h"
```

#### Classes

- struct [Lin\\_ChannelStateType](#)  
*LIN current state structure.*

#### Macros

- #define [LIN\\_SMP\\_CNT16](#) (0U)
- #define [LIN\\_SAMPLE\\_CNT\\_16\\_VALUE](#) (16UL)
- #define [LIN\\_MIN\\_CHANNEL\\_BAUDRATE](#) (1000U)
- #define [LIN\\_MAX\\_CHANNEL\\_BAUDRATE](#) (20000U)
- #define [LIN\\_SYNC\\_DATA](#) (0x55UL)
- #define [LIN\\_DATA\\_LENGTH\\_8](#) ((uint8)8U)
- #define [LIN\\_ID\\_MASK](#) (0x3FU)

## Functions

- void [Lin\\_Hal\\_Init](#) (uint8 Instance, const [Lin\\_ChannelConfigType](#) \*ChannelConfigPtr)  
*Hal Initialize a LIN channel.*
- void [Lin\\_Hal\\_DeInit](#) (uint8 Instance)  
*Deinitializes the Lin module.*
- Hal\_StatusType [Lin\\_Hal\\_GoToSleepMode](#) (uint8 Instance)  
*Lin channel go to sleep.*
- void [Lin\\_Hal\\_GoToIdleState](#) (uint8 Instance)  
*Lin channel go to idle state.*
- Hal\_StatusType [Lin\\_Hal\\_SendWakeupSignal](#) (uint8 Instance)  
*Send LIN wakeup signal.*
- uint8 [Lin\\_Hal\\_ProcessParity](#) (uint8 Pid, uint8 Type)  
*Process identifier parity.*
- Hal\_StatusType [Lin\\_Hal\\_SendFrameData](#) (uint8 Instance, const [Lin\\_PduType](#) \*PduInfo)  
*Send LIN frame data.*
- void [Lin\\_Hal\\_AbortTransferData](#) (uint8 Instance)  
*This function is abort lin transfer.*
- [Lin\\_StatusType](#) [Lin\\_Hal\\_GetStatus](#) (uint8 Instance, uint8 \*LinSduPtr)  
*Gets the status of the LIN driver when Channel is operating.*
- void [Lin\\_Hal\\_SetTimeoutCounter](#) (uint8 Instance, uint32 Timeout)  
*Set timeout counter for timer interrupt.*
- void [Lin\\_Hal\\_TimeoutService](#) (uint8 Instance)  
*Timer interrupt callback function.*

### 4.2.1 Detailed Description

This file provides Hal lin api.

### 4.2.2 Macro Definition Documentation

#### 4.2.2.1 LIN\_DATA\_LENGTH\_8

```
#define LIN_DATA_LENGTH_8 ((uint8)8U)
```

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

#### 4.2.2.2 LIN\_ID\_MASK

```
#define LIN_ID_MASK (0x3FU)
```

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



#### 4.2.2.3 LIN\_MAX\_CHANNEL\_BAUDRATE

```
#define LIN_MAX_CHANNEL_BAUDRATE (20000U)
```

Definition at line 96 of file Lin\_Hal.c.

#### 4.2.2.4 LIN\_MIN\_CHANNEL\_BAUDRATE

```
#define LIN_MIN_CHANNEL_BAUDRATE (1000U)
```

Definition at line 95 of file Lin\_Hal.c.

#### 4.2.2.5 LIN\_SAMPLE\_CNT\_16\_VALUE

```
#define LIN_SAMPLE_CNT_16_VALUE (16UL)
```

Definition at line 93 of file Lin\_Hal.c.

#### 4.2.2.6 LIN\_SMP\_CNT16

```
#define LIN_SMP_CNT16 (0U)
```

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

#### 4.2.2.7 LIN\_SYNC\_DATA

```
#define LIN_SYNC_DATA (0x55UL)
```

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

### 4.2.3 Function Documentation

#### 4.2.3.1 Lin\_Hal\_AbortTransferData()

```
void Lin_Hal_AbortTransferData (  
    uint8 Instance )
```

This function is abort lin transfer.

##### Note

Function ID: DES\_LIN\_API\_004

**Parameters**

in	<i>Instance</i>	LIN module Instance.
----	-----------------	----------------------

**Returns**

void

Definition at line 832 of file Lin\_Hal.c.

**4.2.3.2 Lin\_Hal\_DeInit()**

```
void Lin_Hal_DeInit (  
    uint8 Instance )
```

Deinitializes the Lin module.

**Note**

Function ID: DES\_LIN\_API\_001

**Parameters**

in	<i>Instance</i>	LIN module Instance
----	-----------------	---------------------

**Returns**

void

Definition at line 563 of file Lin\_Hal.c.

**4.2.3.3 Lin\_Hal\_GetStatus()**

```
Lin_StatusType Lin_Hal_GetStatus (  
    uint8 Instance,  
    uint8 * LinSduPtr )
```

Gets the status of the LIN driver when Channel is operating.

**Note**

Function ID: DES\_LIN\_API\_008

**Parameters**

in	<i>Instance</i>	LIN module Instance.
out	<i>LinSduPtr</i>	Pointer to the buffer where the received SDU(Service Data Unit) will be stored.

**Returns**

Lin\_StatusType: Lin current channel status

Definition at line 861 of file Lin\_Hal.c.

**4.2.3.4 Lin\_Hal\_GoToIdleState()**

```
void Lin_Hal_GoToIdleState (  
    uint8 Instance )
```

Lin channel go to idle state.

**Note**

Function ID: DES\_LIN\_API\_007

**Parameters**

in	<i>Instance</i>	LIN module Instance.
----	-----------------	----------------------

**Returns**

void

Definition at line 651 of file Lin\_Hal.c.

**4.2.3.5 Lin\_Hal\_GoToSleepMode()**

```
Hal_StatusType Lin_Hal_GoToSleepMode (  
    uint8 Instance )
```

Lin channel go to sleep.

**Note**

Function ID: DES\_LIN\_API\_005

**Parameters**

in	<i>Instance</i>	LIN module Instance.
----	-----------------	----------------------

**Returns**

Hal\_StatusType: return STATUS\_SUCCESS if successful, STATUS\_BUSY if busy.

Definition at line 605 of file Lin\_Hal.c.

#### 4.2.3.6 Lin\_Hal\_Init()

```
void Lin_Hal_Init (
    uint8 Instance,
    const Lin_ChannelConfigType * ChannelConfigPtr )
```

Hal Initialize a LIN channel.

##### Note

Function ID: DES\_LIN\_API\_000

##### Parameters

in	<i>Instance</i>	LIN module Instance.
in	<i>ChannelConfigPtr</i>	LIN Channel Config pointer.

##### Returns

void

Definition at line 468 of file Lin\_Hal.c.

#### 4.2.3.7 Lin\_Hal\_ProcessParity()

```
uint8 Lin_Hal_ProcessParity (
    uint8 Pid,
    uint8 Type )
```

Process identifier parity.

##### Note

Function ID: DES\_LIN\_API\_003

##### Parameters

in	<i>Pid</i>	ID or Pid.
in	<i>Type</i>	Specifies the operation type: <ul style="list-style-type: none"><li>• LIN_MAKE_PARITY : Compute the parity bits and return the PID</li><li>• LIN_CHECK_PARITY: Check the parity bits return PID if correct,otherwise return 0xFF.</li></ul>

##### Returns

uint8: If successful, return PID, otherwise return 0xFF.

Definition at line 728 of file Lin\_Hal.c.

#### 4.2.3.8 Lin\_Hal\_SendFrameData()

```
Hal_StatusType Lin_Hal_SendFrameData (
    uint8 Instance,
    const Lin_PduType * PduInfo )
```

Send LIN frame data.

##### Note

Function ID: DES\_LIN\_API\_002

##### Parameters

in	<i>Instance</i>	LIN module Instance
in	<i>PduInfo</i>	Pointer to <a href="#">Lin_PduType</a> containing the PID,Checksum model, Response type, Data Length and SDU data

##### Returns

Hal\_StatusType:if successful return STATUS\_SUCCESS or return STATUS\_ERROR/STATUS\_BUSY.

Definition at line 773 of file Lin\_Hal.c.

#### 4.2.3.9 Lin\_Hal\_SendWakeupSignal()

```
Hal_StatusType Lin_Hal_SendWakeupSignal (
    uint8 Instance )
```

Send LIN wakeup signal.

##### Note

Function ID: DES\_LIN\_API\_006

##### Parameters

in	<i>Instance</i>	LIN module Instance.
----	-----------------	----------------------

##### Returns

Hal\_StatusType: return STATUS\_SUCCESS if successful, STATUS\_BUSY if busy.

Definition at line 686 of file Lin\_Hal.c.

#### 4.2.3.10 Lin\_Hal\_SetTimeoutCounter()

```
void Lin_Hal_SetTimeoutCounter (
    uint8 Instance,
    uint32 Timeout )
```

Set timeout counter for timer interrupt.

##### Note

Function ID: DES\_LIN\_API\_017

##### Parameters

in	<i>Instance</i>	LIN module instance
in	<i>Timeout</i>	timeout counter

##### Returns

void

Definition at line 948 of file Lin\_Hal.c.

#### 4.2.3.11 Lin\_Hal\_TimeoutService()

```
void Lin_Hal_TimeoutService (
    uint8 Instance )
```

Timer interrupt callback function.

##### Note

Function ID: DES\_LIN\_API\_018

##### Parameters

in	<i>Instance</i>	LIN module instance
----	-----------------	---------------------

##### Returns

void

Definition at line 969 of file Lin\_Hal.c.

## 4.3 Lin\_Hal.h File Reference

This file provides extern Hal lin api.

```
#include "Lin_Hal_Types.h"
```

## Functions

- void [Lin\\_Hal\\_Init](#) (uint8 Instance, const [Lin\\_ChannelConfigType](#) \*ChannelConfigPtr)  
*Hal Initialize a LIN channel.*
- void [Lin\\_Hal\\_DeInit](#) (uint8 Instance)  
*Deinitializes the Lin module.*
- Hal\_StatusType [Lin\\_Hal\\_GoToSleepMode](#) (uint8 Instance)  
*Lin channel go to sleep.*
- void [Lin\\_Hal\\_GoToIdleState](#) (uint8 Instance)  
*Lin channel go to idle state.*
- Hal\_StatusType [Lin\\_Hal\\_SendWakeupSignal](#) (uint8 Instance)  
*Send LIN wakeup signal.*
- uint8 [Lin\\_Hal\\_ProcessParity](#) (uint8 Pid, uint8 Type)  
*Process identifier parity.*
- Hal\_StatusType [Lin\\_Hal\\_SendFrameData](#) (uint8 Instance, const [Lin\\_PduType](#) \*PduInfo)  
*Send LIN frame data.*
- void [Lin\\_Hal\\_AbortTransferData](#) (uint8 Instance)  
*This function is abort lin transfer.*
- [Lin\\_StatusType](#) [Lin\\_Hal\\_GetStatus](#) (uint8 Instance, uint8 \*LinSduPtr)  
*Gets the status of the LIN driver when Channel is operating.*
- void [Lin\\_Hal\\_SetTimeoutCounter](#) (uint8 Instance, uint32 Timeout)  
*Set timeout counter for timer interrupt.*
- void [Lin\\_Hal\\_TimeoutService](#) (uint8 Instance)  
*Timer interrupt callback function.*

### 4.3.1 Detailed Description

This file provides extern Hal lin api.

### 4.3.2 Function Documentation

#### 4.3.2.1 Lin\_Hal\_AbortTransferData()

```
void Lin_Hal_AbortTransferData (
    uint8 Instance )
```

This function is abort lin transfer.

#### Note

Function ID: DES\_LIN\_API\_004

#### Parameters

in	Instance	LIN module Instance.
----	----------	----------------------

**Returns**

void

Definition at line 832 of file Lin\_Hal.c.

**4.3.2.2 Lin\_Hal\_DeInit()**

```
void Lin_Hal_DeInit (
    uint8 Instance )
```

Deinitializes the Lin module.

**Note**

Function ID: DES\_LIN\_API\_001

**Parameters**

in	<i>Instance</i>	LIN module Instance
----	-----------------	---------------------

**Returns**

void

Definition at line 563 of file Lin\_Hal.c.

**4.3.2.3 Lin\_Hal\_GetStatus()**

```
Lin_StatusType Lin_Hal_GetStatus (
    uint8 Instance,
    uint8 * LinSduPtr )
```

Gets the status of the LIN driver when Channel is operating.

**Note**

Function ID: DES\_LIN\_API\_008

**Parameters**

in	<i>Instance</i>	LIN module Instance.
out	<i>LinSduPtr</i>	Pointer to the buffer where the received SDU(Service Data Unit) will be stored.

**Returns**

Lin\_StatusType: Lin current channel status



Definition at line 861 of file Lin\_Hal.c.

#### 4.3.2.4 Lin\_Hal\_GoToIdleState()

```
void Lin_Hal_GoToIdleState (
    uint8 Instance )
```

Lin channel go to idle state.

##### Note

Function ID: DES\_LIN\_API\_007

##### Parameters

in	<i>Instance</i>	LIN module Instance.
----	-----------------	----------------------

##### Returns

void

Definition at line 651 of file Lin\_Hal.c.

#### 4.3.2.5 Lin\_Hal\_GoToSleepMode()

```
Hal_StatusType Lin_Hal_GoToSleepMode (
    uint8 Instance )
```

Lin channel go to sleep.

##### Note

Function ID: DES\_LIN\_API\_005

##### Parameters

in	<i>Instance</i>	LIN module Instance.
----	-----------------	----------------------

##### Returns

Hal\_StatusType: return STATUS\_SUCCESS if successful, STATUS\_BUSY if busy.

Definition at line 605 of file Lin\_Hal.c.

#### 4.3.2.6 Lin\_Hal\_Init()

```
void Lin_Hal_Init (
    uint8 Instance,
    const Lin_ChannelConfigType * ChannelConfigPtr )
```

Hal Initialize a LIN channel.

##### Note

Function ID: DES\_LIN\_API\_000

##### Parameters

in	<i>Instance</i>	LIN module Instance.
in	<i>ChannelConfigPtr</i>	LIN Channel Config pointer.

##### Returns

void

Definition at line 468 of file Lin\_Hal.c.

#### 4.3.2.7 Lin\_Hal\_ProcessParity()

```
uint8 Lin_Hal_ProcessParity (
    uint8 Pid,
    uint8 Type )
```

Process identifier parity.

##### Note

Function ID: DES\_LIN\_API\_003

##### Parameters

in	<i>Pid</i>	ID or Pid.
in	<i>Type</i>	Specifies the operation type: <ul style="list-style-type: none"><li>• LIN_MAKE_PARITY : Compute the parity bits and return the PID</li><li>• LIN_CHECK_PARITY: Check the parity bits return PID if correct,otherwise return 0xFF.</li></ul>

##### Returns

uint8: If successful, return PID, otherwise return 0xFF.

Definition at line 728 of file Lin\_Hal.c.

#### 4.3.2.8 Lin\_Hal\_SendFrameData()

```
Hal_StatusType Lin_Hal_SendFrameData (
    uint8 Instance,
    const Lin_PduType * PduInfo )
```

Send LIN frame data.

##### Note

Function ID: DES\_LIN\_API\_002

##### Parameters

in	<i>Instance</i>	LIN module Instance
in	<i>PduInfo</i>	Pointer to <a href="#">Lin_PduType</a> containing the PID,Checksum model, Response type, Data Length and SDU data

##### Returns

Hal\_StatusType:if successful return STATUS\_SUCCESS or return STATUS\_ERROR/STATUS\_BUSY.

Definition at line 773 of file Lin\_Hal.c.

#### 4.3.2.9 Lin\_Hal\_SendWakeupSignal()

```
Hal_StatusType Lin_Hal_SendWakeupSignal (
    uint8 Instance )
```

Send LIN wakeup signal.

##### Note

Function ID: DES\_LIN\_API\_006

##### Parameters

in	<i>Instance</i>	LIN module Instance.
----	-----------------	----------------------

##### Returns

Hal\_StatusType: return STATUS\_SUCCESS if successful, STATUS\_BUSY if busy.

Definition at line 686 of file Lin\_Hal.c.

#### 4.3.2.10 Lin\_Hal\_SetTimeoutCounter()

```
void Lin_Hal_SetTimeoutCounter (
    uint8 Instance,
    uint32 Timeout )
```

Set timeout counter for timer interrupt.

##### Note

Function ID: DES\_LIN\_API\_017

##### Parameters

in	<i>Instance</i>	LIN module instance
in	<i>Timeout</i>	timeout counter

##### Returns

void

Definition at line 948 of file Lin\_Hal.c.

#### 4.3.2.11 Lin\_Hal\_TimeoutService()

```
void Lin_Hal_TimeoutService (
    uint8 Instance )
```

Timer interrupt callback function.

##### Note

Function ID: DES\_LIN\_API\_018

##### Parameters

in	<i>Instance</i>	LIN module instance
----	-----------------	---------------------

##### Returns

void

Definition at line 969 of file Lin\_Hal.c.

## 4.4 Lin\_Hal\_Types.h File Reference

This file provides lin module used hardware Types.

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

## Classes

- struct [Lin\\_ChannelInfoType](#)  
*LIN channel info structure.*
- struct [Lin\\_ChannelConfigType](#)  
*Configuration struction of the LIN driver.*
- struct [Lin\\_PduType](#)  
*LIN frame type used to provide PID,checksum model, data length and SDU pointer.*

## Macros

- #define [LIN\\_MAKE\\_PARITY](#) (0U)
- #define [LIN\\_CHECK\\_PARITY](#) (1U)

## Typedefs

- typedef void(\* [Lin\\_CallbackType](#)) (uint8 Instance, [Lin\\_ChannelInfoType](#) \*LinInfo)  
*LIN callback function.*

## Enumerations

- enum [Lin\\_ModeType](#) { [LIN\\_MASTER](#) = 0U, [LIN\\_SLAVE](#) = 1U }  
*LIN Mode.*
- enum [Lin\\_BreakLengthType](#) {  
[BREAK\\_LENGTH\\_13BIT](#) = 0U, [BREAK\\_LENGTH\\_14BIT](#), [BREAK\\_LENGTH\\_15BIT](#), [BREAK\\_LENGTH\\_16BIT](#),  
[BREAK\\_LENGTH\\_17BIT](#), [BREAK\\_LENGTH\\_18BIT](#), [BREAK\\_LENGTH\\_19BIT](#), [BREAK\\_LENGTH\\_20BIT](#),  
[BREAK\\_LENGTH\\_21BIT](#), [BREAK\\_LENGTH\\_22BIT](#), [BREAK\\_LENGTH\\_23BIT](#), [BREAK\\_LENGTH\\_24BIT](#),  
[BREAK\\_LENGTH\\_25BIT](#), [BREAK\\_LENGTH\\_26BIT](#), [BREAK\\_LENGTH\\_27BIT](#), [BREAK\\_LENGTH\\_28BIT](#) }  
*LIN break length for LIN master send.*
- enum [Lin\\_BreakThresholdType](#) { [BREAK\\_THRESHOLD\\_10BIT](#) = 0U, [BREAK\\_THRESHOLD\\_11BIT](#) = 1U }  
*LIN break threshold for LIN slave detect.*
- enum [Lin\\_EventIdType](#) {  
[LIN\\_NO\\_EVENT](#) = 0x00U, [LIN\\_WAKEUP\\_SIGNAL](#) = 0x01U, [LIN\\_SYNC\\_OK](#) = 0x02U, [LIN\\_SYNC\\_ERROR](#) =  
0x03U,  
[LIN\\_PID\\_OK](#) = 0x04U, [LIN\\_PID\\_ERROR](#) = 0x05U, [LIN\\_FRAME\\_ERROR](#) = 0x06U, [LIN\\_READBACK\\_ERROR](#) =  
0x07U,  
[LIN\\_CHECKSUM\\_ERROR](#) = 0x08U, [LIN\\_TX\\_COMPLETED](#) = 0x09U, [LIN\\_RX\\_COMPLETED](#) = 0x0AU, [LIN\\_RX←  
\\_OVERRUN](#) = 0x0BU,  
[LIN\\_NOISE\\_ERROR](#) = 0x0CU, [LIN\\_TIMEOUT\\_ERROR](#) = 0x0DU }  
*LIN event identifier enumeration define.*
- enum [Lin\\_NodeStateType](#) {  
[LIN\\_NODE\\_STATE\\_UNINIT](#) = 0x00U, [LIN\\_NODE\\_STATE\\_SLEEP\\_MODE](#) = 0x01U, [LIN\\_NODE\\_STATE\\_IDLE](#) =  
0x02U, [LIN\\_NODE\\_STATE\\_SEND\\_BREAK\\_FIELD](#) = 0x03U,  
[LIN\\_NODE\\_STATE\\_SEND\\_SYNC](#) = 0x04U, [LIN\\_NODE\\_STATE\\_RECV\\_SYNC](#) = 0x05U, [LIN\\_NODE\\_STATE\\_S←  
END\\_PID](#) = 0x06U, [LIN\\_NODE\\_STATE\\_RECV\\_PID](#) = 0x07U,  
[LIN\\_NODE\\_STATE\\_RECV\\_DATA](#) = 0x08U, [LIN\\_NODE\\_STATE\\_RECV\\_DATA\\_COMPLETED](#) = 0x09U, [LIN\\_NO←  
DE\\_STATE\\_SEND\\_DATA](#) = 0x0AU, [LIN\\_NODE\\_STATE\\_SEND\\_DATA\\_COMPLETED](#) = 0x0BU }  
*LIN node state enumeration define.*
- enum [Lin\\_FrameCsModelType](#) { [LIN\\_ENHANCED\\_CS](#), [LIN\\_CLASSIC\\_CS](#) }  
*Define type for checksum type of the frame.*
- enum [Lin\\_FrameResponseType](#) { [LIN\\_FRAMERESPONSE\\_TX](#), [LIN\\_FRAMERESPONSE\\_RX](#), [LIN\\_FRAMERES←  
PONSE\\_IGNORE](#) }  
*Define type for response type of the frame.*
- enum [Lin\\_StatusType](#) {  
[LIN\\_NOT\\_OK](#) = 0, [LIN\\_TX\\_OK](#), [LIN\\_TX\\_BUSY](#), [LIN\\_TX\\_HEADER\\_ERROR](#),  
[LIN\\_TX\\_ERROR](#), [LIN\\_RX\\_OK](#), [LIN\\_RX\\_BUSY](#), [LIN\\_RX\\_ERROR](#),  
[LIN\\_RX\\_NO\\_RESPONSE](#), [LIN\\_OPERATIONAL](#), [LIN\\_CH\\_SLEEP](#) }  
*Define a range of transfer status.*

### 4.4.1 Detailed Description

This file provides lin module used hardware Types.

### 4.4.2 Macro Definition Documentation

#### 4.4.2.1 LIN\_CHECK\_PARITY

```
#define LIN_CHECK_PARITY (1U)
```

Check parity for PID

Definition at line 52 of file Lin\_Hal\_Types.h.

#### 4.4.2.2 LIN\_MAKE\_PARITY

```
#define LIN_MAKE_PARITY (0U)
```

Make parity for PID

Definition at line 51 of file Lin\_Hal\_Types.h.

### 4.4.3 Typedef Documentation

#### 4.4.3.1 Lin\_CallbackType

```
typedef void(* Lin_CallbackType) (uint8 Instance, Lin\_ChannelInfoType *LinInfo)
```

LIN callback function.

Definition at line 206 of file Lin\_Hal\_Types.h.

### 4.4.4 Enumeration Type Documentation

#### 4.4.4.1 Lin\_BreakLengthType

```
enum Lin\_BreakLengthType
```

LIN break length for LIN master send.

## Enumerator

BREAK_LENGTH_13BIT	break length 13bit
BREAK_LENGTH_14BIT	break length 14bit
BREAK_LENGTH_15BIT	break length 15bit
BREAK_LENGTH_16BIT	break length 16bit
BREAK_LENGTH_17BIT	break length 17bit
BREAK_LENGTH_18BIT	break length 18bit
BREAK_LENGTH_19BIT	break length 19bit
BREAK_LENGTH_20BIT	break length 20bit
BREAK_LENGTH_21BIT	break length 21bit
BREAK_LENGTH_22BIT	break length 22bit
BREAK_LENGTH_23BIT	break length 23bit
BREAK_LENGTH_24BIT	break length 24bit
BREAK_LENGTH_25BIT	break length 25bit
BREAK_LENGTH_26BIT	break length 26bit
BREAK_LENGTH_27BIT	break length 27bit
BREAK_LENGTH_28BIT	break length 28bit

Definition at line 73 of file Lin\_Hal\_Types.h.

## 4.4.4.2 Lin\_BreakThresholdType

```
enum Lin_BreakThresholdType
```

LIN break threshold for LIN slave detect.

## Enumerator

BREAK_THRESHOLD_10BIT	10-bit length
BREAK_THRESHOLD_11BIT	11-bit length

Definition at line 96 of file Lin\_Hal\_Types.h.

## 4.4.4.3 Lin\_EventIdType

```
enum Lin_EventIdType
```

LIN event identifier enumeration define.

## Enumerator

LIN_NO_EVENT	No event
LIN_WAKEUP_SIGNAL	Received a wakeup signal
LIN_SYNC_OK	Sync byte is correct
LIN_SYNC_ERROR	Sync byte is error
LIN_PID_OK	PID correct
LIN_PID_ERROR	PID error
LIN_FRAME_ERROR	Frame error
LIN_READBACK_ERROR	Readback data is error
LIN_CHECKSUM_ERROR	Checksum is error
LIN_TX_COMPLETED	Send data completed
LIN_RX_COMPLETED	Receive data completed
LIN_RX_OVERRUN	Receive overrun flag
LIN_NOISE_ERROR	Noise byte is error
LIN_TIMEOUT_ERROR	Timeout error

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

## 4.4.4.4 Lin\_FrameCsModelType

```
enum Lin_FrameCsModelType
```

Define type for checksum type of the frame.

## Enumerator

LIN_ENHANCED_CS	Enhanced checksum mode
LIN_CLASSIC_CS	Classic checksum mode

Definition at line 145 of file Lin\_Hal\_Types.h.

## 4.4.4.5 Lin\_FrameResponseType

```
enum Lin_FrameResponseType
```

Define type for response type of the frame.

## Enumerator

LIN_FRAMERESPONSE_TX	Response is generated from this node.
LIN_FRAMERESPONSE_RX	Response is generated from another node and is relevant for this node.
LIN_FRAMERESPONSE_IGNORE	Response is generated from one slave to another slave.



Definition at line 154 of file Lin\_Hal\_Types.h.

#### 4.4.4.6 Lin\_ModeType

enum [Lin\\_ModeType](#)

LIN Mode.

Enumerator

LIN_MASTER	set the node type as MASTER
LIN_SLAVE	set the node type as SLAVE

Definition at line 64 of file Lin\_Hal\_Types.h.

#### 4.4.4.7 Lin\_NodeStateType

enum [Lin\\_NodeStateType](#)

LIN node state enumeration define.

Enumerator

LIN_NODE_STATE_UNINIT	Uninitialized state
LIN_NODE_STATE_SLEEP_MODE	Sleep mode state
LIN_NODE_STATE_IDLE	Idle state
LIN_NODE_STATE_SEND_BREAK_FIELD	Send break field state
LIN_NODE_STATE_SEND_SYNC	Send sync byte state
LIN_NODE_STATE_RECV_SYNC	Receive sync byte state
LIN_NODE_STATE_SEND_PID	Send PID state
LIN_NODE_STATE_RECV_PID	Receive PID state
LIN_NODE_STATE_RECV_DATA	Receive data state
LIN_NODE_STATE_RECV_DATA_COMPLETED	Receive data completed state
LIN_NODE_STATE_SEND_DATA	Send data state
LIN_NODE_STATE_SEND_DATA_COMPLETED	Send data completed state

Definition at line 126 of file Lin\_Hal\_Types.h.

#### 4.4.4.8 Lin\_StatusType

enum [Lin\\_StatusType](#)

Define a range of transfer status.

## Enumerator

LIN_NOT_OK	Development or production error occurred.
LIN_TX_OK	Successful transmission.
LIN_TX_BUSY	Ongoing transmission (Header or Response).
LIN_TX_HEADER_ERROR	Erroneous header transmission such as: <ul style="list-style-type: none"><li>• Mismatch between sent and read back data</li><li>• Identifier parity error</li><li>• Physical bus error.</li></ul>
LIN_TX_ERROR	Erroneous transmission such as: <ul style="list-style-type: none"><li>• Mismatch between sent and read back data</li><li>• Physical bus error.</li></ul>
LIN_RX_OK	Reception of correct response.
LIN_RX_BUSY	Ongoing reception: at least one response byte has been received, but the checksum byte has not been received.
LIN_RX_ERROR	Erroneous reception such as: <ul style="list-style-type: none"><li>• Framing error</li><li>• Overrun error</li><li>• Checksum error.</li></ul>
LIN_RX_NO_RESPONSE	No response byte has been received so far.
LIN_OPERATIONAL	Normal operation: <ul style="list-style-type: none"><li>• The related LIN channel is ready to transmit next header</li><li>• No data from previous frame available (e.g. after initialization).</li></ul>
LIN_CH_SLEEP	Sleep mode operation; <ul style="list-style-type: none"><li>• In this mode wake-up detection from slave nodes is enabled.</li></ul>

Definition at line 164 of file Lin\_Hal\_Types.h.

# Index

AC784xx\_API\_Reference\_Manual\_LIN.pdf, [11](#)

AutoBaudEnable

Lin\_ChannelConfigType, [3](#)

BaudRate

Lin\_ChannelConfigType, [3](#)

BreakLength

Lin\_ChannelConfigType, [4](#)

BreakThreshold

Lin\_ChannelConfigType, [4](#)

Callback

Lin\_ChannelConfigType, [4](#)

Lin\_ChannelStateType, [6](#)

ChannelInfo

Lin\_ChannelStateType, [6](#)

Checksum

Lin\_ChannelStateType, [6](#)

CntByte

Lin\_ChannelStateType, [7](#)

Cs

Lin\_PduType, [9](#)

CurrentEventId

Lin\_ChannelInfoType, [5](#)

CurrentNodeState

Lin\_ChannelStateType, [7](#)

CurrentPid

Lin\_ChannelInfoType, [5](#)

DI

Lin\_PduType, [9](#)

Drc

Lin\_PduType, [9](#)

IsBusBusy

Lin\_ChannelStateType, [7](#)

LIN\_CHECK\_PARITY

Lin\_Hal\_Types.h, [26](#)

LIN\_DATA\_LENGTH\_8

Lin\_Hal.c, [12](#)

LIN\_ID\_MASK

Lin\_Hal.c, [12](#)

LIN\_MAKE\_PARITY

Lin\_Hal\_Types.h, [26](#)

LIN\_MAX\_CHANNEL\_BAUDRATE

Lin\_Hal.c, [12](#)

LIN\_MIN\_CHANNEL\_BAUDRATE

Lin\_Hal.c, [13](#)

LIN\_SAMPLE\_CNT\_16\_VALUE

Lin\_Hal.c, [13](#)

LIN\_SMP\_CNT16

Lin\_Hal.c, [13](#)

LIN\_SYNC\_DATA

Lin\_Hal.c, [13](#)

Lin\_BreakLengthType

Lin\_Hal\_Types.h, [26](#)

Lin\_BreakThresholdType

Lin\_Hal\_Types.h, [27](#)

Lin\_CallbackType

Lin\_Hal\_Types.h, [26](#)

Lin\_ChannelConfigType, [3](#)

AutoBaudEnable, [3](#)

BaudRate, [3](#)

BreakLength, [4](#)

BreakThreshold, [4](#)

Callback, [4](#)

ModeType, [4](#)

Lin\_ChannelInfoType, [5](#)

CurrentEventId, [5](#)

CurrentPid, [5](#)

RxBuff, [5](#)

Lin\_ChannelStateType, [6](#)

Callback, [6](#)

ChannelInfo, [6](#)

Checksum, [6](#)

CntByte, [7](#)

CurrentNodeState, [7](#)

IsBusBusy, [7](#)

PreviousNodeState, [7](#)

RxSize, [7](#)

TimeoutCounter, [8](#)

TxBuff, [8](#)

TxSize, [8](#)

Lin\_EventIdType

Lin\_Hal\_Types.h, [27](#)

Lin\_FrameCsModelType

Lin\_Hal\_Types.h, [28](#)

Lin\_FrameResponseType

Lin\_Hal\_Types.h, [28](#)

Lin\_Hal.c, [11](#)

LIN\_DATA\_LENGTH\_8, [12](#)

LIN\_ID\_MASK, [12](#)

LIN\_MAX\_CHANNEL\_BAUDRATE, [12](#)

LIN\_MIN\_CHANNEL\_BAUDRATE, [13](#)

LIN\_SAMPLE\_CNT\_16\_VALUE, [13](#)

LIN\_SMP\_CNT16, [13](#)

LIN\_SYNC\_DATA, [13](#)

Lin\_Hal\_AbortTransferData, [13](#)

Lin\_Hal\_DelInit, [14](#)

Lin\_Hal\_GetStatus, [14](#)

Lin\_Hal\_GoToIdleState, [15](#)

Lin\_Hal\_GoToSleepMode, [15](#)

Lin\_Hal\_Init, [15](#)

Lin\_Hal\_ProcessParity, [16](#)

- Lin\_Hal\_SendFrameData, [16](#)
- Lin\_Hal\_SendWakeupSignal, [17](#)
- Lin\_Hal\_SetTimeoutCounter, [17](#)
- Lin\_Hal\_TimeoutService, [18](#)
- Lin\_Hal.h, [18](#)
  - Lin\_Hal\_AbortTransferData, [19](#)
  - Lin\_Hal\_DelNit, [20](#)
  - Lin\_Hal\_GetStatus, [20](#)
  - Lin\_Hal\_GoToIdleState, [21](#)
  - Lin\_Hal\_GoToSleepMode, [21](#)
  - Lin\_Hal\_Init, [21](#)
  - Lin\_Hal\_ProcessParity, [22](#)
  - Lin\_Hal\_SendFrameData, [22](#)
  - Lin\_Hal\_SendWakeupSignal, [23](#)
  - Lin\_Hal\_SetTimeoutCounter, [23](#)
  - Lin\_Hal\_TimeoutService, [24](#)
- Lin\_Hal\_AbortTransferData
  - Lin\_Hal.c, [13](#)
  - Lin\_Hal.h, [19](#)
- Lin\_Hal\_DelNit
  - Lin\_Hal.c, [14](#)
  - Lin\_Hal.h, [20](#)
- Lin\_Hal\_GetStatus
  - Lin\_Hal.c, [14](#)
  - Lin\_Hal.h, [20](#)
- Lin\_Hal\_GoToIdleState
  - Lin\_Hal.c, [15](#)
  - Lin\_Hal.h, [21](#)
- Lin\_Hal\_GoToSleepMode
  - Lin\_Hal.c, [15](#)
  - Lin\_Hal.h, [21](#)
- Lin\_Hal\_Init
  - Lin\_Hal.c, [15](#)
  - Lin\_Hal.h, [21](#)
- Lin\_Hal\_ProcessParity
  - Lin\_Hal.c, [16](#)
  - Lin\_Hal.h, [22](#)
- Lin\_Hal\_SendFrameData
  - Lin\_Hal.c, [16](#)
  - Lin\_Hal.h, [22](#)
- Lin\_Hal\_SendWakeupSignal
  - Lin\_Hal.c, [17](#)
  - Lin\_Hal.h, [23](#)
- Lin\_Hal\_SetTimeoutCounter
  - Lin\_Hal.c, [17](#)
  - Lin\_Hal.h, [23](#)
- Lin\_Hal\_TimeoutService
  - Lin\_Hal.c, [18](#)
  - Lin\_Hal.h, [24](#)
- Lin\_Hal\_Types.h, [24](#)
  - LIN\_CHECK\_PARITY, [26](#)
  - LIN\_MAKE\_PARITY, [26](#)
  - Lin\_BreakLengthType, [26](#)
  - Lin\_BreakThresholdType, [27](#)
  - Lin\_CallbackType, [26](#)
  - Lin\_EventIdType, [27](#)
  - Lin\_FrameCsModelType, [28](#)
  - Lin\_FrameResponseType, [28](#)
  - Lin\_ModeType, [29](#)
  - Lin\_NodeStateType, [29](#)
  - Lin\_StatusType, [29](#)
- Lin\_ModeType
  - Lin\_Hal\_Types.h, [29](#)
- Lin\_NodeStateType
  - Lin\_Hal\_Types.h, [29](#)
- Lin\_PduType, [8](#)
  - Cs, [9](#)
  - DI, [9](#)
  - Drc, [9](#)
  - Pid, [9](#)
  - SduPtr, [10](#)
- Lin\_StatusType
  - Lin\_Hal\_Types.h, [29](#)
- ModeType
  - Lin\_ChannelConfigType, [4](#)
- Pid
  - Lin\_PduType, [9](#)
- PreviousNodeState
  - Lin\_ChannelStateType, [7](#)
- RxBuff
  - Lin\_ChannelInfoType, [5](#)
- RxSize
  - Lin\_ChannelStateType, [7](#)
- SduPtr
  - Lin\_PduType, [10](#)
- TimeoutCounter
  - Lin\_ChannelStateType, [8](#)
- TxBuff
  - Lin\_ChannelStateType, [8](#)
- TxSize
  - Lin\_ChannelStateType, [8](#)