|  |  |
| --- | --- |
| Document Title | Specification of LFDriver |
| Document Owner | Joint-Lab |
| Document Responsibility | Joint-Lab |
| Document Identification No | T.B.D |
| Document Classification | Standard |
|  |  |
| Document Status | Evaluation |
| Part of AUTOSAR Release | T.B.D |

|  |  |  |
| --- | --- | --- |
| Document Change History | | |
| Realease |  |  |
| 1.0 | Song Chen Hong | * Initial Release |

Table of Contents

[Introduction and functional overview 3](#_Toc408409197)

[Acronyms and abbreviations 3](#_Toc408409198)

[Related documentation 3](#_Toc408409199)

[Constraints and assumptions 3](#_Toc408409200)

[API Specification 3](#_Toc408409201)

[Imported types 3](#_Toc408409202)

[Type definitions 3](#_Toc408409203)

[Function specifications 4](#_Toc408409204)

[LFDrv\_Init 4](#_Toc408409205)

[LFDrv\_Deinit 4](#_Toc408409206)

[LFDrv\_SetDataBuffer 5](#_Toc408409207)

[LFDrv\_ClearDataBuffer 6](#_Toc408409208)

[LFDrv\_RestCmd 6](#_Toc408409209)

[LFDrv\_AsyncTrsmStart 6](#_Toc408409210)

[LFDrv\_AsyncTrsmStop 7](#_Toc408409211)

[LFDrv\_GetStatus 7](#_Toc408409212)

[LFDrv\_DMACmpInterrupt 8](#_Toc408409213)

[Sequence diagrams 9](#_Toc408409214)

[Initialization 9](#_Toc408409215)

[De-initialization 9](#_Toc408409216)

[Sending PWM 10](#_Toc408409217)

[Getting the LF Driver state 11](#_Toc408409218)

[Stop the LF Driver 11](#_Toc408409219)

# Introduction and functional overview

This specification specifies the functionality, API and the configuration of the software module LF driver.

The driver provides functions for initialization and control of the microcontroller internal PWM stage and DMA.

The DMA shall change the PWM’s period, duty and polarity as the rule of Manchester and the special requirement.

# Acronyms and abbreviations

|  |  |
| --- | --- |
| ***Acronym:*** | ***Description:*** |
| None | None |

|  |  |
| --- | --- |
| ***Abbreviation:*** | ***Description:*** |
| LFDrv | Low Frequency Driver |

# Related documentation

Input documents

[1] LFDriver\_Requirement.doc

# Constraints and assumptions

Limitations

The LFDriver SWS does not cover all PWM channels to send PWM.

# API Specification

## Imported types

None

## Type definitions

LFDrvStatusReturnType

[SWS\_LFD\_00109]⌈

|  |  |  |
| --- | --- | --- |
| ***Name:*** | LFDrvStatusReturnType | |
| ***Type:*** | Enumeration | |
| ***Range:*** | LFDrv\_IDLE | The LF Driver is in idle state. |
| LFDrv\_BUSY | The LF Driver is in busy state. |
| ***Description:*** | Return the state of LF Driver. | |

⌋()

LFDrvStatusWord

[SWS\_LFD\_00110]⌈

|  |  |  |
| --- | --- | --- |
| ***Name:*** | LFDrvStatusWord | |
| ***Type:*** | Enumeration | |
| ***Range:*** | LFDrvIsIdle | Set the LF Driver state to idle. |
| LFDrvIsBusy | Set the LF Driver state to busy. |
| ***Description:*** | Set the state of LF Driver. | |

⌋()

## Function specifications

### LFDrv\_Init

[ SWS\_LFD\_00100 ]

|  |  |
| --- | --- |
| ***Service name:*** | LFDrv\_Init |
| ***Syntax:*** | void LFDrv\_Init(void) |
| ***Service ID[hex]:*** | 0x00 |
| ***Sync/Async:*** | Synchronous |
| ***Reentrancy:*** | Non Reentrant |
| ***Parameters (in):*** | None |
| ***Parameters (inout):*** | None |
| ***Parameters (out):*** | None |
| ***Return value:*** | None |
| ***Description:*** | Service for LF Driver initialization |

**[SWS-LFD-00001]** ⌈The function LFDrv\_Init shall initialize the clock used by TAU module, the PIN used by PWM output and TAU module’s configuration as PWM output function. ⌋ (SRS\_ LFDRIVER\_009, SRS\_ LFDRIVER\_011)

**[SWS-LFD-00002]** ⌈The function LFDrv\_Init shall set the communication state to IDLE. ⌋ ()

### LFDrv\_Deinit

[ SWS\_LFD\_00101 ]

|  |  |
| --- | --- |
| ***Service name:*** | LFDrv\_Deinit |
| ***Syntax:*** | void LFDrv\_Deinit(void) |
| ***Service ID[hex]:*** | 0x01 |
| ***Sync/Async:*** | Synchronous |
| ***Reentrancy:*** | Non Reentrant |
| ***Parameters (in):*** | None |
| ***Parameters (inout):*** | None |
| ***Parameters (out):*** | None |
| ***Return value:*** | None |
| ***Description:*** | Service for LF Driver de-initialization |

**[SWS-LFD-00003]** ⌈The function LFDrv\_Deinit shall release the clock setting to the POR value, release the PIN setting to the POR value and release the TAU module’s configuration to the POR value. ⌋ (SRS\_ LFDRIVER\_010)

### LFDrv\_SetDataBuffer

**[ SWS\_LFD\_00102 ]**

|  |  |  |
| --- | --- | --- |
| ***Service name:*** | LFDrv\_SetDataBuffer | |
| ***Syntax:*** | void LFDrv\_SetDataBuffer(uint8 \*databuff,uint8 Siz) | |
| ***Service ID[hex]:*** | 0x03 | |
| ***Sync/Async:*** | Synchronous | |
| ***Reentrancy:*** | Non Reentrant | |
| ***Parameters (in):*** | \*databuff | Pointer to the data buffer |
| Siz | The data buffer’s size |
| ***Parameters (inout):*** | None | |
| ***Parameters (out):*** | None | |
| ***Return value:*** | rv | TURE: function execute right  FALSE: function execute wrong |
| ***Description:*** | Service store data to the data buffers which as the DMA transferring’s source address and configure DMA | |

**[SWS-LFD-004]** ⌈The function LFDrv\_DmaConfig shall configure 4 DMA channels for transferring. ⌋

**[SWS-LFD-005]** ⌈DMA0 shall be configured to transfer data from RAM to the TAU’s RDT register. ⌋

**[SWS-LFD-006]** ⌈DMA1 shall be configured to transfer period data from RAM to the TAU master channel’s CDR register. ⌋

**[SWS-LFD-007]** ⌈DMA2 shall be configured to transfer duty data from RAM to the TAU slave channel’s CDR register. ⌋

**[SWS-LFD-008]** ⌈DMA3 shall be configured to transfer data from RAM to the TAU’s TOL register. ⌋

**[SWS-LFD-009]** ⌈The function LFDrv\_SetDataBuffer shall get the contents and size of communication to store into local buffer. ⌋ (SRS\_ LFDRIVER\_012)

**[SWS-LFD-010]** ⌈The function LFDrv\_SetDataBuffer shall reconfigure the data bit by bit as the communication requirements. ⌋ (SRS\_ LFDRIVER\_005,SRS\_ LFDRIVER\_006,SRS\_ LFDRIVER\_007,SRS\_ LFDRIVER\_008)

### LFDrv\_ClearDataBuffer

**[ SWS\_LFD\_00103 ]**

|  |  |
| --- | --- |
| ***Service name:*** | LFDrv\_ClearDataBuffer |
| ***Syntax:*** | void LFDrv\_ClearDataBuffer(void) |
| ***Service ID[hex]:*** | 0x04 |
| ***Sync/Async:*** | Synchronous |
| ***Reentrancy:*** | Non Reentrant |
| ***Parameters (in):*** | None |
| ***Parameters (inout):*** | None |
| ***Parameters (out):*** | None |
| ***Return value:*** | None |
| ***Description:*** | Service clear the data buffers |

**[SWS-LFD-011]** ⌈The function LFDrv\_ClearDataBuffer shall clear the data buffers. ⌋

### LFDrv\_RestCmd

**[ SWS\_LFD\_00104 ]**

|  |  |
| --- | --- |
| ***Service name:*** | LFDrv\_RestCmd |
| ***Syntax:*** | void LFDrv\_RestCmd(void) |
| ***Service ID[hex]:*** | 0x05 |
| ***Sync/Async:*** | Synchronous |
| ***Reentrancy:*** | Non Reentrant |
| ***Parameters (in):*** | None |
| ***Parameters (inout):*** | None |
| ***Parameters (out):*** | None |
| ***Return value:*** | None |
| ***Description:*** | Service reset the data buffer’s index |

**[SWS-LFD-012]** ⌈The function LFDrv\_RestCmd shall reset the data buffer’s index. ⌋

### LFDrv\_AsyncTrsmStart

**[ SWS-LFD-00105 ]**

|  |  |
| --- | --- |
| ***Service name:*** | LFDrv\_AsyncTrsmStart |
| ***Syntax:*** | void LFDrv\_AsyncTrsmStart(void) |
| ***Service ID[hex]:*** | 0x06 |
| ***Sync/Async:*** | Synchronous |
| ***Reentrancy:*** | Non Reentrant |
| ***Parameters (in):*** | None |
| ***Parameters (inout):*** | None |
| ***Parameters (out):*** | None |
| ***Return value:*** | None |
| ***Description:*** | Service start the TAU modules to send PWM |

**[SWS-LFD-012]** ⌈The function LFDrv\_AsyncTrsmStart shall start the TAU modules and set

communication sate to BUSY. ⌋ (SRS\_ LFDRIVER\_013)

### LFDrv\_AsyncTrsmStop

**[ SWS-LFD-00106 ]**

|  |  |
| --- | --- |
| ***Service name:*** | LFDrv\_AsyncTrsmStop |
| ***Syntax:*** | void LFDrv\_AsyncTrsmStop(void) |
| ***Service ID[hex]:*** | 0x07 |
| ***Sync/Async:*** | Synchronous |
| ***Reentrancy:*** | Non Reentrant |
| ***Parameters (in):*** | None |
| ***Parameters (inout):*** | None |
| ***Parameters (out):*** | None |
| ***Return value:*** | None |
| ***Description:*** | Service stop the TAU modules |

**[SWS-LFD-00014]** ⌈The function LFDrv\_AsyncTrsmStop shall stop the TAU modules and set communication sate to IDLE. ⌋ (SRS\_ LFDRIVER\_014)

### LFDrv\_GetStatus

**[ SWS-LFD-00107 ]**

|  |  |  |
| --- | --- | --- |
| ***Service name:*** | LFDrv\_GetStatus | |
| ***Syntax:*** | LFDrvStatusReturnType LFDrv\_GetStatus(void) | |
| ***Service ID[hex]:*** | 0x08 | |
| ***Sync/Async:*** | Synchronous | |
| ***Reentrancy:*** | Non Reentrant | |
| ***Parameters (in):*** | None | |
| ***Parameters (inout):*** | None | |
| ***Parameters (out):*** | None | |
| ***Return value:*** | LFDrvRetState | LFDrv\_IDLE : LFDrv state is IDLE  LFDrv\_BUSY : LFDrv state is BUSY |
| ***Description:*** | Service get the communication status | |

**[SWS-LFD-00015]** ⌈The function LFDrv\_GetStatus shall get the status of LFDrv module. ⌋ (SRS\_ LFDRIVER\_015)

### LFDrv\_DMACmpInterrupt

**[ SWS-LFD-00108 ]**

|  |  |
| --- | --- |
| ***Service name:*** | LFDrv\_DMACmpInterrupt |
| ***Syntax:*** | void LFDrv\_DMACmpInterrupt(void) |
| ***Service ID[hex]:*** | 0x09 |
| ***Sync/Async:*** | Synchronous |
| ***Reentrancy:*** | Non Reentrant |
| ***Parameters (in):*** | None |
| ***Parameters (inout):*** | None |
| ***Parameters (out):*** | None |
| ***Return value:*** | None |
| ***Description:*** | Service complete the communication |

**[SWS-LFD-00016]** ⌈The function LFDrv\_DMACmpInterrupt shall complete the communication and stop the TAU counter and clear DMA transfer request flag and completed flag. ⌋ (SRS\_ LFDRIVER\_016)

# Sequence diagrams

## Initialization

LF Driver User

<module>

LF Driver

LFDrv\_Init()

LFDrv\_Init()

Description:

LF Driver Initialization

Set the clock for TAU counter, set the pin function for PWM output and configure TAU module as PWM output function.

Comments:

Figure 1: LF Driver Initialization

## De-initialization

<module>

LF Driver

LF Driver User

LFDrv\_Deinit()

LFDrv\_Deinit()

Figure 2: LF Driver De-Initialization

Description:

LF Driver De-Initialization

Comments:

## Sending PWM

<module>

LF Driver

LF Driver User

Figure 3: Sending PWM

LFDrv\_DMACmpInterrupt()

LFDrv\_DMACmpInterrupt()

LFDrv\_RestCmd()

LFDrv\_ClearDataBuffer()

LFDrv\_RestCmd()

LFDrv\_ClearDataBuffer()

LFDrv\_SetDataBuffer(uint8 \*databuff,uint8 Siz)

LFDrv\_SetDataBuffer()

LFDrv\_AsyncTrsmStart()

LFDrv\_AsyncTrsmStart()

Description:

Get the communication data and reconfigure the data to store into the buffer. Configure DMA.

Description:

Clear the data buffer.

Description:

If setting data buffer execute true, then start the TAU counter to send PWM.

Description:

PWM transmission completed, use it to handle something that users want to do.

Description:

Sending PWM

Comments:

## Getting the LF Driver state

Description:

Get the LF Driver state.

Comments:

LFDrv\_GetStatus()

LFDrv\_GetStatus()

<module>

LF Driver

LF Driver User

Figure 4: Getting the LF Driver state

## Stop the LF Driver

Figure 5: Stop the LF Driver

Description:

When the LF Driver is sending PWM, users can call this function to stop it.

Comments:

LFDrv\_AsyncTrsmStop()

LFDrv\_AsyncTrsmStop()

<module>

LF Driver

LF Driver User