

# User Manual

for S32K3 LIN Driver

Document Number: UM34LINASRR21-11 Rev0000R3.0.0 Rev. 1.0

<b>1 Revision History</b>	<b>2</b>
<b>2 Introduction</b>	<b>3</b>
2.1 Supported Derivatives	3
2.2 Overview	4
2.3 About This Manual	5
2.4 Acronyms and Definitions	6
2.5 Reference List	6
<b>3 Driver</b>	<b>7</b>
3.1 Requirements	7
3.2 Driver Design Summary	7
3.3 Hardware Resources	8
3.4 Deviations from Requirements	12
3.5 Driver Limitations	13
3.6 Driver usage and configuration tips	13
3.6.1 Dual Clock Feature	13
3.6.2 How to configure to detect the wake-up signal	14
3.6.3 Frame Timeout Disable Feature	14
3.6.4 How to configure multicore	16
3.6.5 Auto-detect baudrate feature	18
3.6.6 What need to do with callback	20
3.6.7 How to configure a FLEXIO channel	21
3.7 Runtime errors	22
3.8 Symbolic Names Disclaimer	23
<b>4 Tresos Configuration Plug-in</b>	<b>24</b>
4.1 Module Lin	25
4.2 Container LinGlobalConfig	26
4.3 Container LinChannel	26
4.4 Parameter LinChannelId	26
4.5 Parameter LinNodeType	27
4.6 Parameter LinChannelBaudRate	27
4.7 Parameter BreakLength	28
4.8 Parameter DetectedBreakLength	28
4.9 Parameter LinResponseTimeout	29
4.10 Parameter LinHeaderTimeout	30
4.11 Parameter LinHwChannel	30
4.12 Parameter LinChannelWakeupSupport	31
4.13 Reference LinClockRef	31
4.14 Reference LinClockRef_Alternate	32

4.15 Reference LinChannelEcuMWakeupSource . . . . .	32
4.16 Reference LinChannelEcucPartitionRef . . . . .	33
4.17 Reference LinFlexioRxControllerRef . . . . .	33
4.18 Reference LinFlexioTxControllerRef . . . . .	34
4.19 Container LinGeneral . . . . .	34
4.20 Parameter LinMulticoreSupport . . . . .	35
4.21 Parameter LinDevErrorDetect . . . . .	35
4.22 Parameter LinIndex . . . . .	36
4.23 Parameter LinTimeoutMethod . . . . .	36
4.24 Parameter LinTimeoutDuration . . . . .	37
4.25 Parameter LinVersionInfoApi . . . . .	38
4.26 Reference LinEcucPartitionRef . . . . .	38
4.27 Container AutosarExt . . . . .	39
4.28 Parameter LinDisableDemReportErrorStatus . . . . .	39
4.29 Parameter LinFrameTimeoutDisable . . . . .	40
4.30 Parameter LinEnableUserModeSupport . . . . .	40
4.31 Parameter LinLpuartStartTimerNotification . . . . .	41
4.32 Parameter LinLpuartStopTimerNotification . . . . .	41
4.33 Parameter LinFlexioStartTimerNotification . . . . .	42
4.34 Parameter LinFlexioStopTimerNotification . . . . .	42
4.35 Parameter LinLpuartWakeupTimerNotification . . . . .	43
4.36 Container LinDemEventParameterRefs . . . . .	43
4.37 Reference LIN_E_TIMEOUT . . . . .	44
4.38 Container CommonPublishedInformation . . . . .	44
4.39 Parameter ArReleaseMajorVersion . . . . .	44
4.40 Parameter ArReleaseMinorVersion . . . . .	45
4.41 Parameter ArReleaseRevisionVersion . . . . .	45
4.42 Parameter ModuleId . . . . .	46
4.43 Parameter SwMajorVersion . . . . .	46
4.44 Parameter SwMinorVersion . . . . .	47
4.45 Parameter SwPatchVersion . . . . .	47
4.46 Parameter VendorApiInfix . . . . .	48
4.47 Parameter VendorId . . . . .	48
4.48 Configuration elements of Lin . . . . .	49
4.49 Form POST_BUILD_VARIANT_USED . . . . .	49
4.50 Attribute IMPLEMENTATION_CONFIG_VARIANT detailed description . . . . .	49
4.51 Form IMPLEMENTATION_CONFIG_VARIANT . . . . .	50
4.52 Attribute IMPLEMENTATION_CONFIG_VARIANT detailed description . . . . .	50
4.53 Form AutosarExt . . . . .	50
4.53.1 LinDisableDemReportErrorStatus (AutosarExt) . . . . .	50

4.54	Attribute LinDisableDemReportErrorStatus (AutosarExt) detailed description . . . . .	50
4.54.1	LinEnableUserModeSupport (AutosarExt) . . . . .	51
4.55	Attribute LinEnableUserModeSupport (AutosarExt) detailed description . . . . .	51
4.56	Form LinGeneral . . . . .	51
4.56.1	LinMulticoreSupport (LinGeneral) . . . . .	51
4.57	Attribute LinMulticoreSupport (LinGeneral) detailed description . . . . .	51
4.57.1	LinDevErrorDetect (LinGeneral) . . . . .	52
4.58	Attribute LinDevErrorDetect (LinGeneral) detailed description . . . . .	52
4.58.1	LinIndex (LinGeneral) . . . . .	52
4.59	Attribute LinIndex (LinGeneral) detailed description . . . . .	52
4.59.1	LinTimeoutDuration (LinGeneral) . . . . .	52
4.60	Attribute LinTimeoutDuration (LinGeneral) detailed description . . . . .	52
4.60.1	LinVersionInfoApi (LinGeneral) . . . . .	53
4.61	Attribute LinVersionInfoApi (LinGeneral) detailed description . . . . .	53
4.61.1	LinEcucPartitionRef (LinGeneral) . . . . .	53
4.62	Attribute LinVersionInfoApi (LinGeneral) detailed description . . . . .	53
4.63	Form LinDemEventParameterRefs . . . . .	53
4.63.1	LIN_E_TIMEOUT (LinDemEventParameterRefs) . . . . .	54
4.64	Attribute LIN_E_TIMEOUT (LinDemEventParameterRefs) detailed description . . . . .	54
4.65	Form LinGlobalConfig . . . . .	54
4.65.1	Form LinChannel . . . . .	54
4.66	Attribute LinChannelId (LinChannel) detailed description . . . . .	54
4.67	Attribute LinChannelBaudRate (LinChannel) detailed description . . . . .	55
4.68	Attribute LinHwChannel (LinChannel) detailed description . . . . .	55
4.69	Attribute LinClockRef (LinChannel) detailed description . . . . .	55
4.70	Attribute LinClockRef_Alternate (LinChannel) detailed description . . . . .	56
4.71	Attribute LinChannelWakeupSupport (LinChannel) detailed description . . . . .	56
4.72	Attribute LinChannelEcuMWakeupSource (LinChannel) detailed description . . . . .	56
4.73	Attribute LinChannelEcucPartitionRef (LinChannel) detailed description . . . . .	57
<b>5</b>	<b>Module Index</b>	<b>58</b>
5.1	Software Specification . . . . .	58
<b>6</b>	<b>Module Documentation</b>	<b>59</b>
6.1	FLEXIO_IP . . . . .	59
6.1.1	Detailed Description . . . . .	59
6.1.2	Data Structure Documentation . . . . .	60
6.1.3	Macro Definition Documentation . . . . .	63
6.1.4	Types Reference . . . . .	64
6.1.5	Enum Reference . . . . .	64
6.1.6	Function Reference . . . . .	67

6.2 LIN Driver . . . . .	73
6.2.1 Detailed Description . . . . .	73
6.2.2 Data Structure Documentation . . . . .	74
6.2.3 Macro Definition Documentation . . . . .	75
6.2.4 Function Reference . . . . .	81
6.3 LIN . . . . .	88
6.3.1 Detailed Description . . . . .	88
6.3.2 Macro Definition Documentation . . . . .	88
6.3.3 Enum Reference . . . . .	88
6.3.4 Function Reference . . . . .	89
6.4 Lpuart Lin IPL . . . . .	90
6.4.1 Detailed Description . . . . .	90
6.4.2 Data Structure Documentation . . . . .	91
6.4.3 Macro Definition Documentation . . . . .	95
6.4.4 Types Reference . . . . .	96
6.4.5 Enum Reference . . . . .	96
6.4.6 Function Reference . . . . .	99



## Chapter 1

### Revision History

Revision	Date	Author	Description
1.0	31.03.2023	NXP RTD Team	S32K3 Real-Time Drivers AUTOSAR 4.4 & R21-11 Version 3.0.0

## Chapter 2

### Introduction

- [Supported Derivatives](#)
- [Overview](#)
- [About This Manual](#)
- [Acronyms and Definitions](#)
- [Reference List](#)

This User Manual describes NXP Semiconductor AUTOSAR LIN for S32K3XX. AUTOSAR LIN driver configuration parameters and deviations from the specification are described in Driver chapter of this document. AUTOSAR LIN driver requirements and APIs are described in the AUTOSAR LIN driver software specification document.

### 2.1 Supported Derivatives

The software described in this document is intended to be used with the following microcontroller devices of NXP Semiconductors:

- s32k310\_mqfp100
- s32k310\_lqfp48
- s32k311\_mqfp100 / MWCT2015S\_mqfp100
- s32k311\_lqfp48
- s32k312\_mqfp100 / MWCT2016S\_mqfp100
- s32k312\_mqfp172 / MWCT2016S\_mqfp172
- s32k314\_mqfp172
- s32k314\_mapbga257
- s32k322\_mqfp100 / MWCT2D16S\_mqfp100
- s32k322\_mqfp172 / MWCT2D16S\_mqfp172

- s32k324\_mqfp172 / MWCT2D17S\_mqfp172
- s32k324\_mapbga257
- s32k341\_mqfp100
- s32k341\_mqfp172
- s32k342\_mqfp100
- s32k342\_mqfp172
- s32k344\_mqfp172
- s32k344\_mapbga257
- s32k394\_mapbga289
- s32k396\_mapbga289
- s32k358\_mqfp172
- s32k358\_mapbga289
- s32k328\_mqfp172
- s32k328\_mapbga289
- s32k338\_mqfp172
- s32k338\_mapbga289
- s32k348\_mqfp172
- s32k348\_mapbga289
- s32m274\_lqfp64
- s32m276\_lqfp64

All of the above microcontroller devices are collectively named as S32K3.

Note: MWCT part numbers contain NXP confidential IP for Qi Wireless Power.

## 2.2 Overview

**AUTOSAR (AUTomotive Open System ARchitecture)** is an industry partnership working to establish standards for software interfaces and software modules for automobile electronic control systems.

AUTOSAR:

- paves the way for innovative electronic systems that further improve performance, safety and environmental friendliness.
- is a strong global partnership that creates one common standard: "Cooperate on standards, compete on implementation".
- is a key enabling technology to manage the growing electrics/electronics complexity. It aims to be prepared for the upcoming technologies and to improve cost-efficiency without making any compromise with respect to quality.
- facilitates the exchange and update of software and hardware over the service life of the vehicle.



## 2.3 About This Manual

This Technical Reference employs the following typographical conventions:

- **Boldface** style: Used for important terms, notes and warnings.
- *Italic* style: Used for code snippets in the text. Note that C language modifiers such "const" or "volatile" are sometimes omitted to improve readability of the presented code.

Notes and warnings are shown as below:

Note

This is a note.

Warning

This is a warning

## 2.4 Acronyms and Definitions

Term	Definition
API	Application Programming Interface
ASM	Assembler
BSMI	Basic Software Make file Interface
C/CPP	C and C++ Source Code
DEM	Diagnostic Event Manager
DET	Development Error Tracer
ECUM	ECU state Manager
ECU	Electronic Control Unit
ISR	Interrupt Service Routine
LIN	Local Interconnect Network
LSB	Least Significant Bit
MCU	Micro Controller Unit
MSB	Most Significant Bit
N/A	Not Applicable
RAM	Random Access Memory
SIU	Systems Integration Unit
SWS	Software Specification
VLE	Variable Length Encoding
XML	Extensible Markup Language

## 2.5 Reference List

#	Title	Version
1	Specification of LIN Driver	AUTOSAR Release R21-11
2	Reference Manual	S32K3xx Reference Manual, Rev.6, Draft B, 01/2023
3	Reference Manual for S32K39 and S32K37	S32K39 and S32K37 Reference Manual, Rev. 2 Draft A, 11/2022
4	Reference Manual for S32M27x	S32M27x Reference Manual, Rev.2, Draft A, — 02/2023
5	Datasheet	S32K3xx Data Sheet, Rev. 6, 11/2022
6	Datasheet for S32K39 and S32K37	S32K396 Data Sheet, Rev. 1.1 — 08/2022
7	Datasheet for S32M27x	S32M2xx Data Sheet, Rev. 2 RC — 12/2022
8	Errata	S32K358_0P14E Mask Set Errata – Rev. 28, 9/2022
		S32K396_0P40E Mask Set Errata, Rev. DEC2022, 12/2022
		S32K311_0P98C Mask Set Errata, Rev. 6/March/2023, 3/2023
		S32K312: Mask Set Errata for Mask 0P09C, Rev. 25/↔ April/2022
		S32K342: Mask Set Errata for Mask 0P97C, Rev. 10, 11/2022
		S32K3x4: Mask Set Errata for Mask 0P55A/1P55A, Rev. 14/Oct/2022

## Chapter 3

### Driver

- [Requirements](#)
- [Driver Design Summary](#)
- [Hardware Resources](#)
- [Deviations from Requirements](#)
- [Driver Limitations](#)
- [Driver usage and configuration tips](#)
- [Runtime errors](#)
- [Symbolic Names Disclaimer](#)

### 3.1 Requirements

Requirements for this driver are detailed in the Autosar Driver Software Specification document (See [Table Reference List](#)).

### 3.2 Driver Design Summary

The LIN driver is part of the Real Time Drivers, performs the hardware access and offers a hardware independent API to the upper layer.

The only upper layer, which has access to the LIN driver, is the LIN Interface.

A LIN driver can support more than one channel over LPUART and FLEXIO hardware units.

This LIN driver supports both MASTER and SLAVE mode over all the hardware units.

The LIN Driver for S32K3XX uses the LPUART and FLEXIO on-chip hardware modules which provides special support for the LIN protocol.

It can be used to automate most tasks of a LIN master and slave node.

It is possible to transmit entire frames (or sequences of frames) and receive data from LIN slaves.

The LIN physical interface should be connected to the LPUART module pins in order to get the LIN bus voltage levels. The same requirement applies also for FLEXIO module.

The LPUART hardware unit has the following major features:

- Transmit and receive baud rate can operate asynchronous to the bus clock
- Programmable baud rates (13-bit modulo divider) with configurable oversampling ratio from 4x to 32x
- Receive data register full, transmit data register empty and transmission complete interrupts
- Receive overrun, framing error, and noise error detection
- Optional 13-bit break character generation

The FLEXIO hardware unit has the following major features:

- Array of 32-bit shift registers with transmit, receive and data match modes
- Double buffered shifter operation for continuous data transfer
- Shifter concatenation to support large transfer sizes
- Automatic start/stop bit generation
- Programmable baud rates independent of bus clock frequency
- Interrupt, DMA or polled transmit/receive operation
- Integrated general purpose input/output registers and pin rising/falling edge interrupts to simplify software support

### 3.3 Hardware Resources

The device family includes S32K311, S32K312, S32K314, S32K324, S32K344, S32K342, S32K341, S32K322, S32K328, S32K338, S32K348, S32K358, S32K39, S32M276, S32M274 derivatives.

The hardware configured by the Lin driver are LPUART and FLEXIO.

Lin Physical Channels:

- For S32K310/S32K311 has 8 channels (4 LPUART channels and 4 FLEXIO channels): LPUART\_0, LPUART\_1, LPUART\_2, LPUART\_3, FLEXIO\_0, FLEXIO\_1, FLEXIO\_2, FLEXIO\_3.
- For S32K312 has 12 channels (8 LPUART channels and 4 FLEXIO channels): LPUART\_0, LPUART\_1, LPUART\_2, LPUART\_3, LPUART\_4, LPUART\_5, LPUART\_6, LPUART\_7, FLEXIO\_0, FLEXIO\_1, FLEXIO\_2, FLEXIO\_3.
- For S32K314 has 20 channels (16 LPUART channels and 4 FLEXIO channels): LPUART\_0, LPUART\_1, LPUART\_2, LPUART\_3, LPUART\_4, LPUART\_5, LPUART\_6, LPUART\_7, LPUART\_8, LPUART\_9, LPUART\_10, LPUART\_11, LPUART\_12, LPUART\_13, LPUART\_14, LPUART\_15, FLEXIO\_0, FLEXIO\_1, FLEXIO\_2, FLEXIO\_3.
- For S32K322/S32K324/S32K328/ has 20 channels (16 LPUART channels and 4 FLEXIO channels): LPUART\_0, LPUART\_1, LPUART\_2, LPUART\_3, LPUART\_4, LPUART\_5, LPUART\_6, LPUART\_7, LPUART\_8, LPUART\_9, LPUART\_10, LPUART\_11, LPUART\_12, LPUART\_13, LPUART\_14, LPUART\_15, FLEXIO\_0, FLEXIO\_1, FLEXIO\_2, FLEXIO\_3.

- For S32K328/S32K338/S32K348/S32K358 has 20 channels (16 LPUART channels and 4 FLEXIO channels): LPUART\_0, LPUART\_1, LPUART\_2, LPUART\_3, LPUART\_4, LPUART\_5, LPUART\_6, LPUART\_7, LPUART\_8, LPUART\_9, LPUART\_10, LPUART\_11, LPUART\_12, LPUART\_13, LPUART\_14, LPUART\_15, FLEXIO\_0, FLEXIO\_1, FLEXIO\_2, FLEXIO\_3.
- For S32K39 has 9 channels (5 LPUART channels and 4 FLEXIO channels): LPUART\_0, LPUART\_1, LPUART\_2, LPUART\_3, LPUART\_4, LPUART\_IP\_MSC, FLEXIO\_0, FLEXIO\_1, FLEXIO\_2, FLEXIO\_3.
- For S32M276/S32M274 has 8 channels (4 LPUART channels and 4 FLEXIO channels): LPUART\_0, LPUART\_1, LPUART\_2, LPUART\_3, FLEXIO\_0, FLEXIO\_1, FLEXIO\_2, FLEXIO\_3.

For example:

In EB tresos and also Design Studio, *LPUART\_0* channel has named *LPUART\_0*, correspondingly as below:



**Figure 3.1 Lin hardware channel configuration in EB Tresos.**

In case of the FLEXIO module there is only one hardware instance on the chip. The FLEXIO\_0, FLEXIO\_1, FLEXIO\_2, FLEXIO\_3 are an abstractization of the resources that the FLEXIO unit uses in order to implement the Lin protocol. For more details about the FLEXIO hardware unit, please read "Flexible I/O (FlexIO)" subchapter of the Communication chapter of the Reference Manual.

The LIN channel to S32K328, S32K338, S32K348, S32K358 microcontroller pin mapping can be done using file *S32K358\_S32K348\_S32K38\_S32K328\_IOMUX.xlsx* from Reference manual. The LIN channel to S32M276, S32M274 microcontroller pin mapping can be done using file *S32M27x\_IOMUX.xlsx* from Reference manual. The LIN channel to S32K39 microcontroller pin mapping can be done using file *S32K39\_S32K37\_IOMUX.xlsx* from Reference manual. The LIN channel to S32K310, S32K311 microcontroller pin mapping can be done using file *S32K311\_S32K310\_IOMUX.xlsx* from Reference manual. The LIN channel to S32K342, S32K341, S32K322 microcontroller pin mapping can be done using file *S32K342\_S32K341\_S32K322\_IOMUX.xlsx* from Reference manual. The LIN channel to S32K344, S32K324, S32K314 microcontroller pin mapping can be done using file *S32K344\_S32K324\_S32K314\_IOMUX.xlsx* from Reference manual.

The RM chapter stated above has a table as below:

Table 628. LPUART instances

Instance	S32K314/S32K324/S32K44/S32K358/S32K348/ S32K338/S32K328/S32K388	S32K311/S32K322/ S32K342/S32K341	S32K312
LPUART_0	Yes	Yes	Yes
LPUART_1	Yes	Yes	Yes
LPUART_2	Yes	Yes	Yes
LPUART_3	Yes	Yes	Yes
LPUART_4	Yes	No	Yes
LPUART_5	Yes	No	Yes
LPUART_6	Yes	No	Yes
LPUART_7	Yes	No	Yes
LPUART_8	Yes	No	No
LPUART_9	Yes	No	No
LPUART_10	Yes	No	No
LPUART_11	Yes	No	No
LPUART_12	Yes	No	No
LPUART_13	Yes	No	No
LPUART_14	Yes	No	No
LPUART_15	Yes	No	No

Figure 3.2 LPUART configuration in Reference manual

*LPUART\_0* channel can be found in the *S32K358\_S32K348\_S32K38\_S32K328\_IOMUX.xlsx* file from Reference manual with naming is LPUART0. The Pin-Muxing is:

PTA2	SIUL_MSCR2	0000_0000	GPIO[2]	SIUL		I/O
PTA2		0000_0001	FCCU_ERR0	FCCU	FCCU Error 0 Output	O
PTA2		0000_0010	eMIOS_1_CH[19]_Y	eMIOS_1	eMIOS Channel	O
PTA2		0000_0101	FXIO_D4	FXIO	FlexIO Bi-directional Shift/timer I/O	O
PTA2		0000_0110	LCU0_OUT3	LCU0	LCU Output	O
PTA2		0000_0111	LPSP15_SIN	LPSP15	LPSP1 Serial Data Input	O
PTA2	-	-	ADC1_X[0]	ADC1	ADC External Input	I
PTA2	-	-	WKPU[0]	WKPU	Wakeup Input	I
PTA2	-	-	CMP1_IN2	CMP1	Comparator Input Signal	I
PTA2	SIUL_IMCR530	0000_0001	EIRQ[2]	SIUL	External Interrupt	I
PTA2	SIUL_IMCR611	0000_0100	eMIOS_1_CH[19]_Y	eMIOS_1	eMIOS Channel	I
PTA2	SIUL_IMCR660	0000_0001	FCCU_ERR_IN0	FCCU	FCCU Error 0 Output	I
PTA2	SIUL_IMCR668	0000_0011	FXIO_D4	FXIO	FlexIO Bi-directional Shift/timer I/O	I
PTA2	SIUL_IMCR699	0000_0001	LPUART0_RX	LPUART0	Receive	I
PTA2	SIUL_IMCR751	0000_0010	LPSP11_SIN	LPSP11	LPSP1 Serial Data Input	I
PTA2	SIUL_IMCR779	0000_0010	LPSP15_SIN	LPSP15	LPSP1 Serial Data Input	I
PTA3	SIUL_MSCR3	0000_0000	GPIO[3]	SIUL		I/O
PTA3		0000_0001	FCCU_ERR1	FCCU	FCCU Error 1 Output	O
PTA3		0000_0010	eMIOS_1_CH[20]_Y	eMIOS_1	eMIOS Channel	O
PTA3		0000_0011	LPSP11_SCK	LPSP11	LPSP1 Serial Clock I/O	O
PTA3		0000_0100	LCU0_OUT2	LCU0	LCU Output	O
PTA3		0000_0101	FXIO_D5	FXIO	FlexIO Bi-directional Shift/timer I/O	O
PTA3		0000_0110	LPUART0_TX	LPUART0	Transmit	O
PTA3		0000_0111	LPSP15_SCK	LPSP15	LPSP1 Serial Clock I/O	O
PTA3	-	-	ADC1_S17	ADC1	ADC Standard Input	I
PTA3	SIUL_IMCR531	0000_0001	EIRQ[3]	SIUL	External Interrupt	I
PTA3	SIUL_IMCR612	0000_0100	eMIOS_1_CH[20]_Y	eMIOS_1	eMIOS Channel	I
PTA3	SIUL_IMCR661	0000_0001	FCCU_ERR_IN1	FCCU	FCCU Error 1 Input	I
PTA3	SIUL_IMCR669	0000_0011	FXIO_D5	FXIO	FlexIO Bi-directional Shift/timer I/O	I
PTA3	SIUL_IMCR750	0000_0001	LPSP11_SCK	LPSP11	LPSP1 Serial Clock I/O	I
PTA3	SIUL_IMCR778	0000_0010	LPSP15_SCK	LPSP15	LPSP1 Serial Clock I/O	I
PTA3	SIUL_IMCR875	0000_0001	LPUART0_TX	LPUART0	Transmit	I

Figure 3.3 Pin-Muxing of LPUART\_0

*FLEXIO\_0* channel can be found in the *S32K358\_S32K348\_S32K38\_S32K328\_IOMUX.xlsx* file from Reference manual using the name of the pin configured in the Mcl component. For example, if the Lin FLEXIO\_0 channel uses a reference from Mcl which has Flexio PIN 5 for Rx and PIN 4 for TX, the pins can be found in the file *S32K358\_S32K348\_S32K38\_S32K328\_IOMUX.xlsx* as the picture below.

PTD2	SIUL_MSCR98	0000_0000	GPIO[98]	SIUL		I/O
PTD2		0000_0001	LCU0_OUT1	LCU0	LCU Output	O
PTD2		0000_0010	eMIOS_1_CH[21]_Y	eMIOS_1	eMIOS Channel	O
PTD2		0000_0011	LPSP1_SOUT	LPSP1	LPSP1 Serial Data Output	O
PTD2		0000_0100	FXIO_D4	FXIO	FlexIO Bi-directional Shift/timer I/O	O
PTD2		0000_0101	FXIO_D6	FXIO	FlexIO Bi-directional Shift/timer I/O	O
PTD2		0000_0110	LPUART3_TX	LPUART3	Transmit	O
PTD2		0000_0111	LPSP15_SOUT	LPSP15	LPSP1 Serial Data Output	O
PTD2	-	-	WKPU[9]	WKPU	Wakeup Input	I
PTD2	-	-	ADC0_S16	ADC0	ADC Standard Input	I
PTD2	SIUL_IMCR538	0000_0011	EIRQ[10]	SIUL	External Interrupt	I
PTD2	SIUL_IMCR613	0000_0100	eMIOS_1_CH[21]_Y	eMIOS_1	eMIOS Channel	I
PTD2	SIUL_IMCR668	0000_0001	FXIO_D4	FXIO	FlexIO Bi-directional Shift/timer I/O	I
PTD2	SIUL_IMCR670	0000_0011	FXIO_D6	FXIO	FlexIO Bi-directional Shift/timer I/O	I
PTD2	SIUL_IMCR752	0000_0001	LPSP1_SOUT	LPSP1	LPSP1 Serial Data Output	I
PTD2	SIUL_IMCR780	0000_0010	LPSP15_SOUT	LPSP15	LPSP1 Serial Data Output	I
PTD2	SIUL_IMCR861	0000_0001	TRGMUX_IN5	TRGMUX	Trigger Mux Input	I
PTD2	SIUL_IMCR878	0000_0010	LPUART3_TX	LPUART3	Transmit	I
PTD3	SIUL_MSCR99	0000_0000	GPIO[99]	SIUL		I/O
PTD3		0000_0010	eMIOS_1_CH[22]_X	eMIOS_1	eMIOS Channel	O
PTD3		0000_0011	LPSP1_PCS0	LPSP1	Peripheral Chip Select 0	O
PTD3		0000_0100	FXIO_D5	FXIO	FlexIO Bi-directional Shift/timer I/O	O
PTD3		0000_0101	FXIO_D7	FXIO	FlexIO Bi-directional Shift/timer I/O	O
PTD3		0000_0110	LCU0_OUT0	LCU0	LCU Output	O
PTD3	-	-	NMI_b	SYSTEM	Non Maskable Interrupt	I
PTD3	-	-	WKPU[1]	WKPU	Wakeup Input	I
PTD3	-	-	ADC0_S10	ADC0	ADC Standard Input	I
PTD3	SIUL_IMCR539	0000_0011	EIRQ[11]	SIUL	External Interrupt	I
PTD3	SIUL_IMCR614	0000_0100	eMIOS_1_CH[22]_X	eMIOS_1	eMIOS Channel	I
PTD3	SIUL_IMCR669	0000_0010	FXIO_D5	FXIO	FlexIO Bi-directional Shift/timer I/O	I
PTD3	SIUL_IMCR671	0000_0011	FXIO_D7	FXIO	FlexIO Bi-directional Shift/timer I/O	I
PTD3	SIUL_IMCR702	0000_0011	LPUART3_RX	LPUART3	Receive	I
PTD3	SIUL_IMCR744	0000_0001	LPSP1_PCS0	LPSP1	Peripheral Chip Select 0	I
PTD3	SIUL_IMCR860	0000_0001	TRGMUX_IN4	TRGMUX	Trigger Mux Input	I

Figure 3.4 Pin-Muxing of FLEXIO Module

The Pin-Muxing for LPUART is: For transmit data signal TXD connected to pin PTA[3]. And for receive data signal RXD connected to pin PTA[2].

The Pin-Muxing for FLEXIO is: There are several FLEXIO pins which can be set as INPUT or OUTPUT. An example is PTD[2] for RX and PTD[3] for TX.

### 3.4 Deviations from Requirements

The driver deviates from the AUTOSAR LIN Driver software specification in some places. The table below identifies the AUTOSAR requirements that are not implemented or out of scope for the LIN Driver.

Term	Definition
N/S	Out of scope
N/I	Not implemented
N/F	Not fully implemented

Below table identifies the AUTOSAR requirements that are not fully implemented, implemented differently or out of scope for the LIN driver.



Requirement	Status	Description	Notes
SWS_Lin_00055	N/S	The Lin module shall fulfill all design and implementation guidelines as described in Specification of C Implementation Rules AUTOSAR_TR_C - ImplementationRules.pdf.	Requirement already covered by process.
SWS_Lin_00026	N/S	If the LIN hardware unit cannot queue the bytes for transmission or reception (e.g. simple UART implementation), the LIN driver shall provide a temporary communication buffer.	The LIN hardware already has a data buffer built-in.
SWS_Lin_00039	N/S	Values that can be configured are hardware dependent. Therefore, the rules and constraints cannot be given in the standard.	This is not a requirement.
SWS_Lin_00999	N/S	These requirements are not applicable to this specification.	This is not a requirement.
SWS_Lin_00201	N/S	For different LIN hardware units a separate LIN driver needs to be implemented. It is up to the implementer to adapt the driver to the different instances of similar LIN channels.	Rejection reason: LIN driver is designed to adapt to different LIN hardware units since.
SWS_Lin_00099	N/S	If development error detection for the Lin module is enabled: the function Lin_Init shall check the parameter Config for being within the allowed range. If Config is not in the allowed range, the function Lin_Init shall raise the development error LIN_E_INVA - LID_POINTER.	Requirement is marked Rejected, as it is replaced by CPR_RTD_00255

## 3.5 Driver Limitations

The limitations of this driver are:

- In Slave mode, Lin over Flexio can't distinguish frame error from overrun error, so LIN\_ERR\_INC\_RESP will be reported to LinIf instead of LIN\_ERR\_RESP\_STOPBIT
- Break length detection is not configurable (we only support 11-bits break length detect).
- Autobaud feature is not implemented over FLEXIO module.
- On K39x, LPUART\_MSC is a special resource and Lin Driver doesn't support.

## 3.6 Driver usage and configuration tips

### 3.6.1 Dual Clock Feature

Lin driver allows dynamic change of working frequency. The LinClockRef\_Alternate node can be found at Lin/↔ LinGlobalConfig/[Configuration\_name]/LinChannel/General. This node should be enabled in order to have this feature active. The frequency can be changed if Lin\_SetClockMode function is call, after calling Lin\_Init function.

## Note

- Our recommended usage for this API is to call it when the driver is in a lower power state but still in active use.

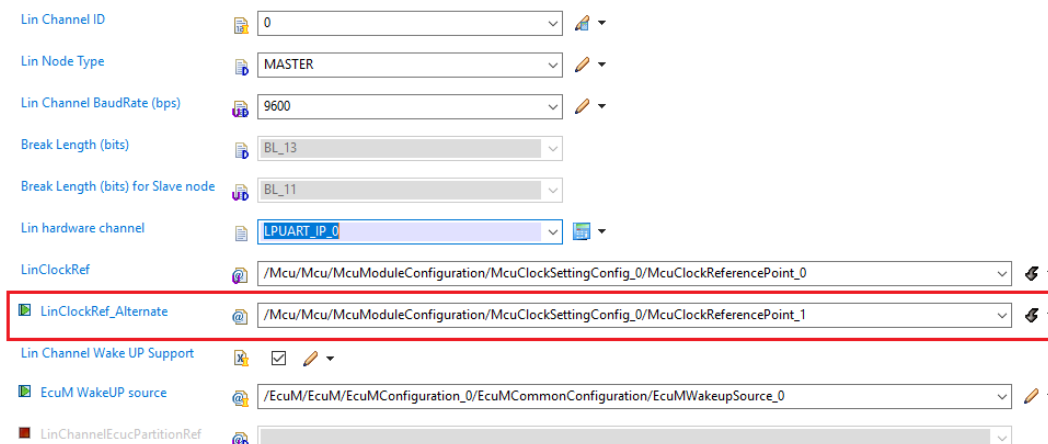


Figure 3.5 Dual Clock Feature configuration node in EB Tresos.

### 3.6.2 How to configure to detect the wake-up signal

Because wake-up signal detection is not support by LPUART hardware, so driver will use the timer and Lpuart edge detection interrupts to detect and measure the wake up signal. The wake-up signal feature uses a timer which must be initialized in the application/test cases. Lin driver uses a notification functions which are intended to start and stop the measurement of the timeout period. These functions must be implemented by user. The notification will be called by interrupt of Lpuart when Lpuart detects both falling and rising edge of wake up signal. User need to calculate time get from twice of wake-up and change it to nanosecond. Driver only handles the nanosecond.

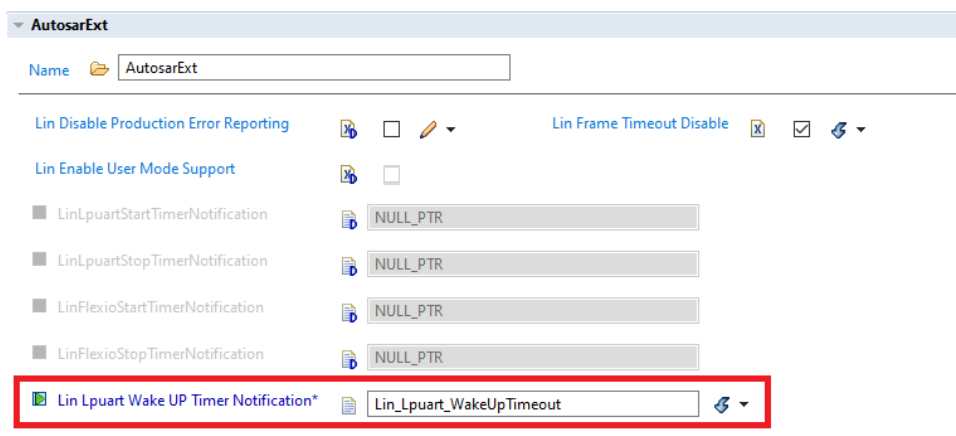
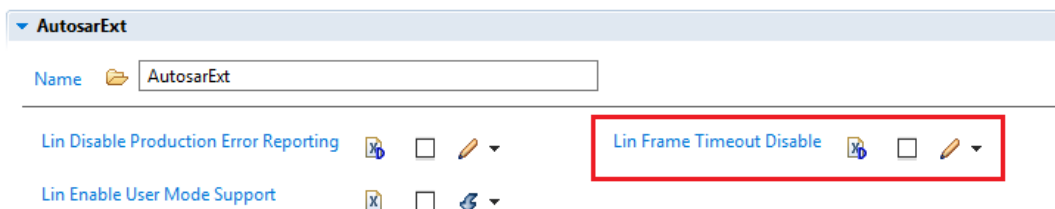


Figure 3.6 Callback function configuration.

**3.6.3 Frame Timeout Disable Feature** In Autosar mode, the 'Lin Frame Timeout Disable' checkbox in AutosarExt container should be enabled in order to have this feature active. If LinFrameTimeoutDisable is ON

then Lin driver will accept the frame that is longer than Maximal Frame Length. For the LIN driver, timeout feature is applied to both LIN 2.1 master and slave nodes, in response frame reception.



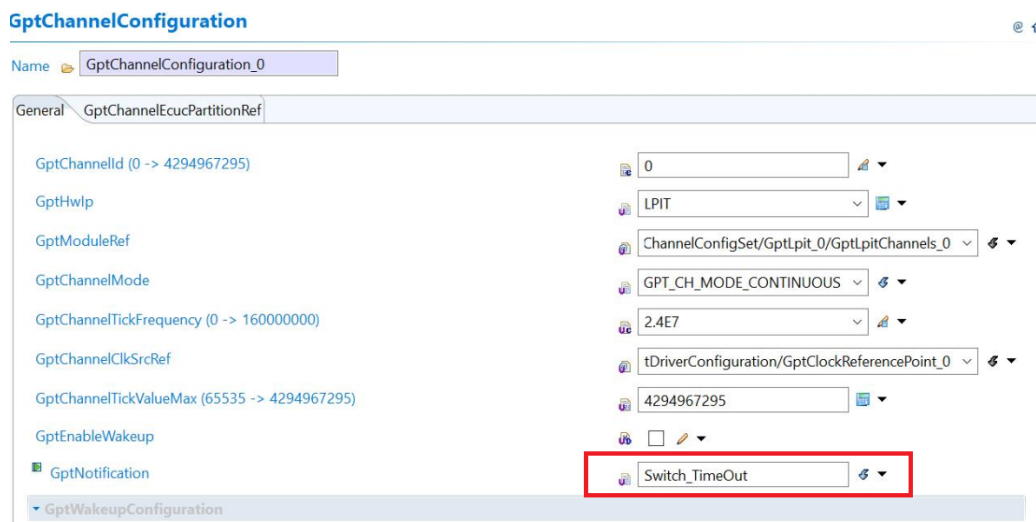
**Figure 3.7 Frame Timeout Disable Feature configuration node in EB Tresos.**

The frame timeout feature uses a timer which must be initialized in the application/test cases. Lin driver uses two notification functions which are intended to start and stop the measurement of the timeout period. These functions must be implemented by user.

The start function has the following prototype : `void func (uint8 Channel, uint32 MicroSeconds);` This function must start the timer to count the period of time provided via the second parameter.

The stop function has the following prototype : `void func (uint8 Channel);` This function must stop the timer counting.

Note: User should configure timeout notification function in timer IRQ handler, which must be called when timeout occurs via GptNotification node of GPT module.



**Figure 3.8 Timeout notification function configured in EB Tresos.**

Switch\_TimeOut, as shown in the diagram above, is a user-written timeout notification function that calls the Lin driver's corresponding Lpuart\_Lin\_Ip\_TimerExpiredService or Flexio\_Lin\_Ip\_TimerExpiredService functions when using Lpuart or Flexio.

If this feature is enabled, LinFrameTimeoutDisable is OFF (not scored-out in the configurators). The two notification functions must be also provided in the configurator. In the next figure, there is an example of a configuration in

Design Studio for a Lpuart channel. In case of a Flexio channel, notification functions must be added separately in the Flexio specific fields.

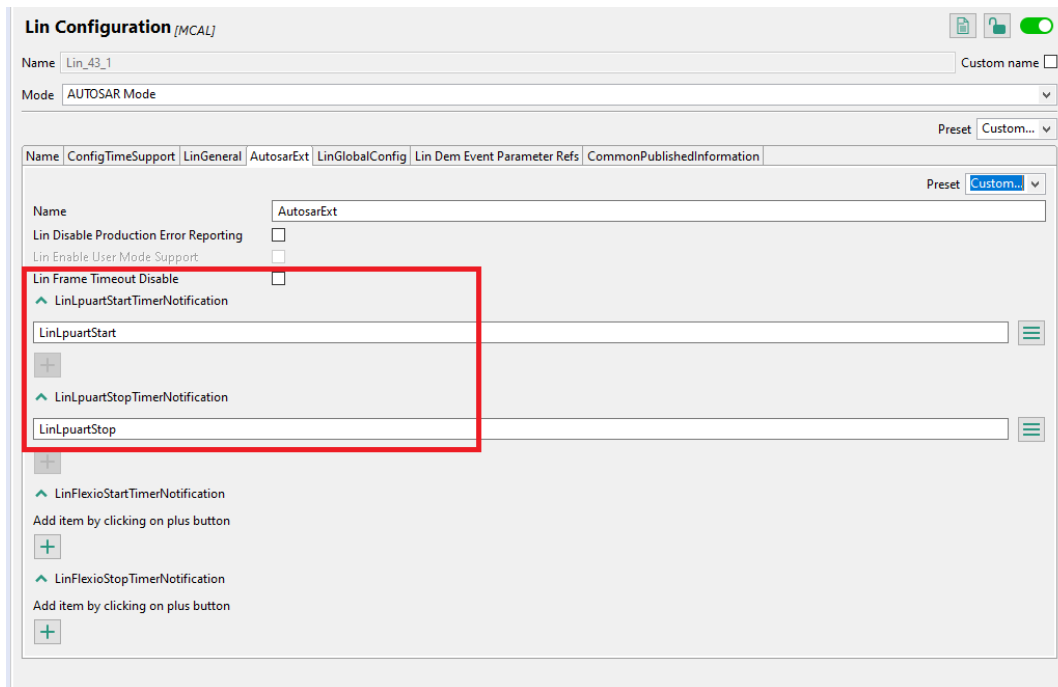


Figure 3.9 Frame Timeout Enabled Feature and notification functions configured in Design Studio.

## 3.6.4 How to configure multicore

Set multicore support node of Lin module is ON to partitions can be referred.

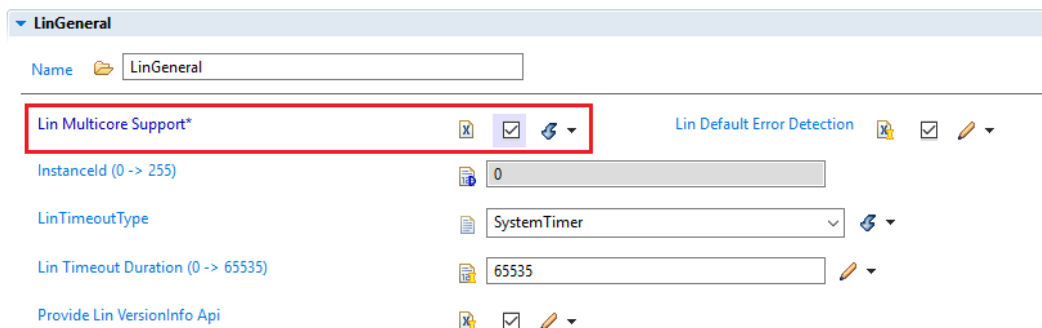


Figure 3.10 Enable Multicore support for Lin module.

A partition will be assigned for each Lin channel. Each LinChannel selects only a partition (reference from ECUC).

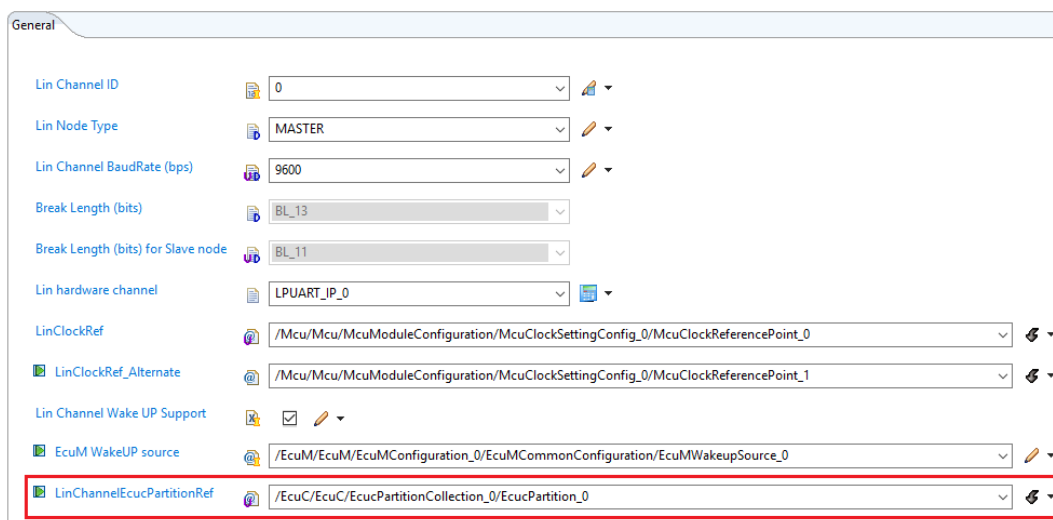


Figure 3.11 Select partition for Lin channel.

Each partition will be referred to a core by Os. Each OsApplication will be set to one partition maps to one core.

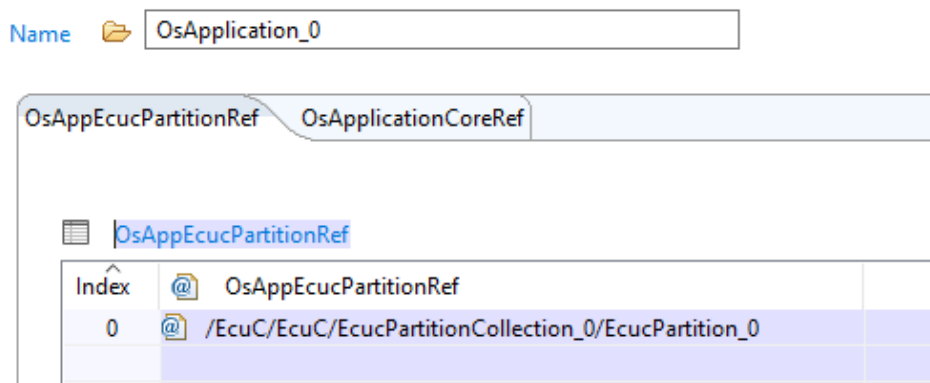


Figure 3.12 Partition referred in Os.

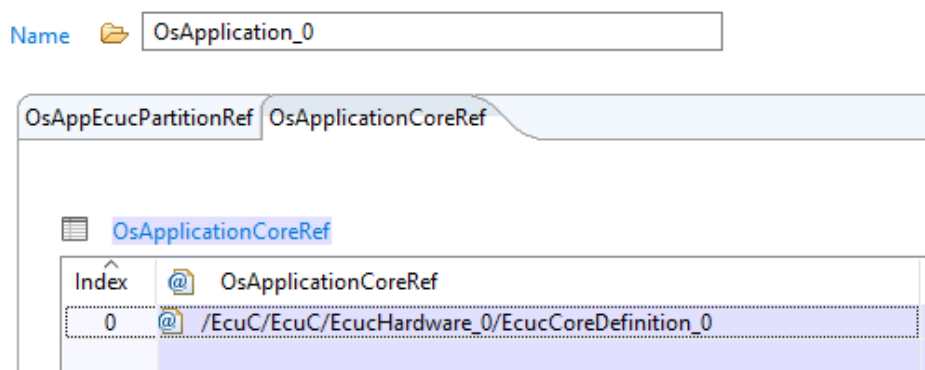
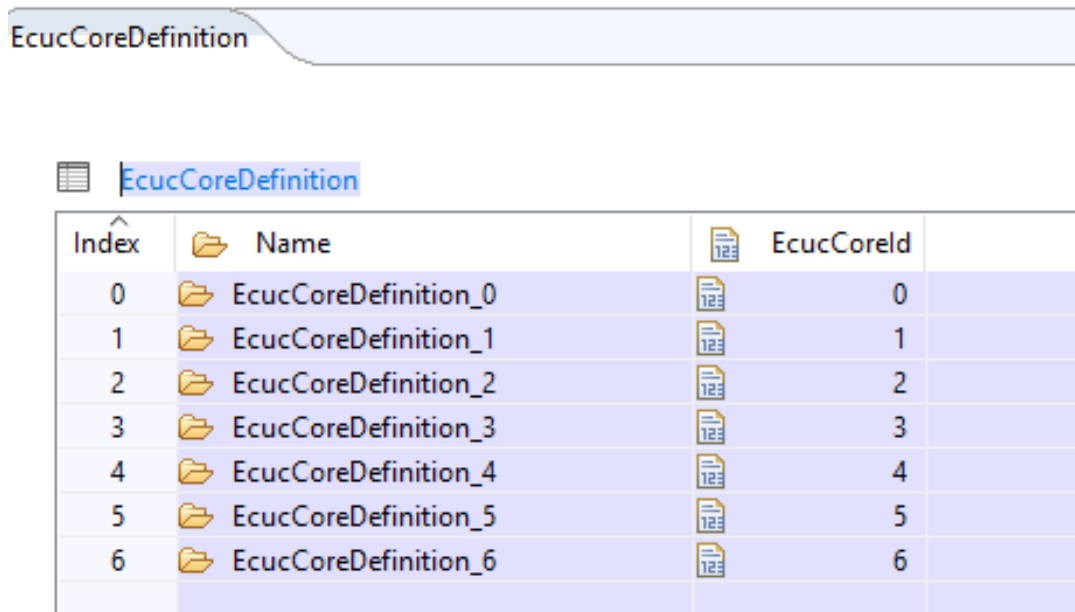


Figure 3.13 Core referred in Os.

In EcuC configuration, each EcucCoreDefinition will selected a Core.

EcucCoreDefinition



Index	Name	EcucCoreId
0	EcucCoreDefinition_0	0
1	EcucCoreDefinition_1	1
2	EcucCoreDefinition_2	2
3	EcucCoreDefinition_3	3
4	EcucCoreDefinition_4	4
5	EcucCoreDefinition_5	5
6	EcucCoreDefinition_6	6

Figure 3.14 EcucCoreDefinition selected a core.

Note

- The multi-core feature is only available for S32K324 device
- When using multi-core for Flexio, all Flexio channels must be assigned to the same core.

3.6.5 Auto-detect baudrate feature

Autobaud is an extensive feature in Lpuart Lin Driver which allows a slave node to automatically detect baudrate of LIN bus and adapt its original baudrate to bus value. Auto Baud is applied when the baudrate of the incoming data is unknown.

This feature is only available for Lpuart Lin Ip driver.

This feature needs one Pin and one Timer. For Pin setting, we have 2 ways to implement:

- First of all (recommended), user uses Rx pin of current Lpuart channel, initializes and sets up it as GPIO mode with interrupt enable on both falling and rising edge. After autobaud process has finished successfully, user need to set up this Rx pin as Uart Rx mode.
- The other one, user can use any Pin, set up it as GPIO mode with interrupt enable on both falling and rising edge (or can set up output trigger timer pin E.g FTM pin) and physically wires this pin to Rx pin. For Timer: This feature uses a timer (ex: FTM) in Input Capture Mode. The timer must be initialized in the application/test cases.

In order to use this feature, the Autobaud Feature must be enable in the configurator and a function must be provided and implemented.

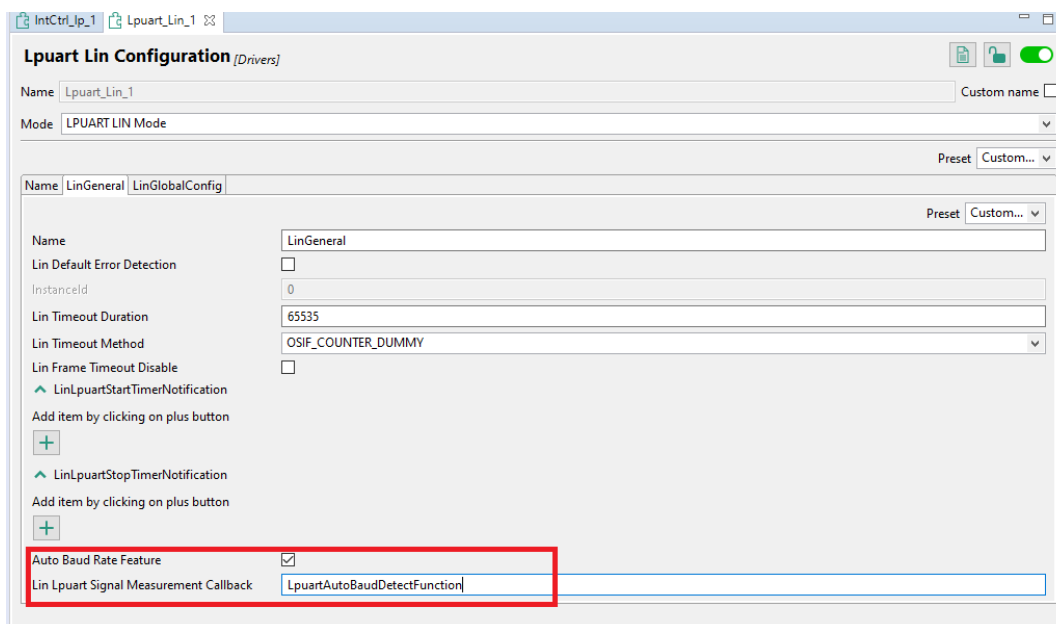


Figure 3.15 Autobaud Feature and notification functions configured in Design Studio

Enable Auto baud rate in LinChannel container:

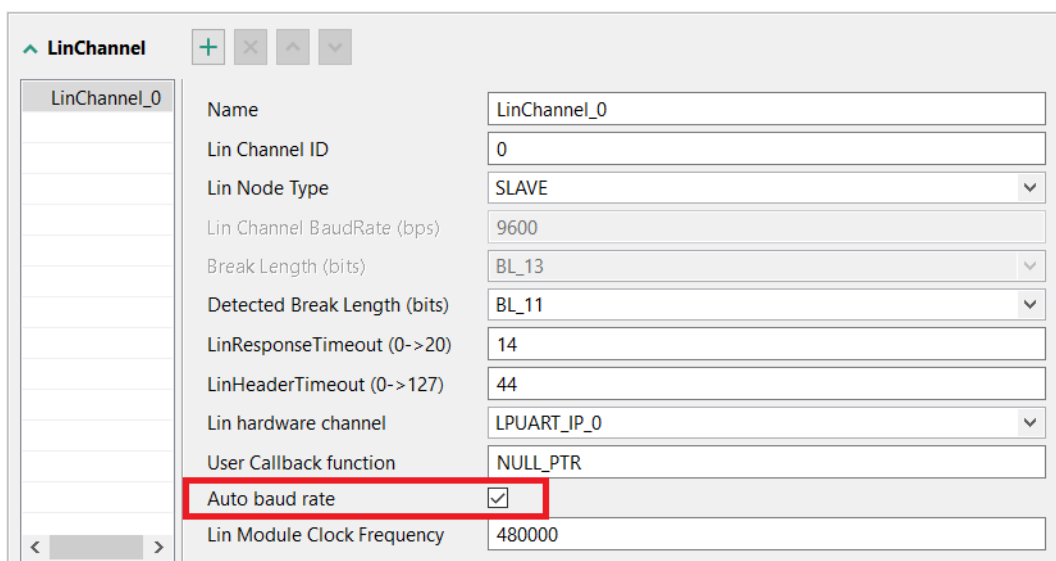


Figure 3.16 Autobaud Feature configured for channel in Design Studio

Users shall assign measurement callback function pointer in *Lin Lpuart Signal Measurement Callback* field in S32↔DS. In the application/test cases, the function has the following prototype: "void func(uint8 Instance, uint32 \*↔Nanoseconds);". This function must assign into Nanoseconds parameter time period between two consecutive pin

active edges in nano seconds. If this function is called for the first time, it will start the timer to measure time. When an event (such as detecting a falling edge of a dominant signal while node is in sleep mode) occurs, LIN driver will call this callback to start time measurement. Then on rising edge of that signal, LIN driver will call this callback function to get time interval of that dominant signal in nano seconds. If Autobaud feature is enabled, LIN driver uses this callback to measure two bit time length between two consecutive falling edges of the sync byte in order to evaluate Master's baudrate. Users can implement this function in their applications.

The application should use a Port Pin interrupt or (external pin trigger timer interrupt of both rising and falling edges(E.g FTM)), call `Lpuart_Lin_Ip_AutoBaudCapture(uint8 Instance)` function to calculate and set Slave's baudrate like Master's baudrate. When receiving a frame header, the slave detect LIN bus's baudrate based on the synchronization byte and adapts its baudrate accordingly. On changing baudrate, the slave set current event ID to `LPUART_LIN_IP_BAUDRATE_ADJUSTED` and call the callback function. In that callback function users might change the pin mode as Uart Rx mode.

Note: Baudrate evaluation process is executed until autobaud successfully, do not call any function else during this process.

### 3.6.6 What need to do with callback

The callback function will be called if at least one of the below events occurred:

- A wake-up signal was detected
  - Baudrate was adjusted
  - At least one of the below errors happened:
    - Sync field error,
    - Pid error,
    - Frame error,
    - Read-back error,
    - Checksum error,
    - Overrun error,
    - Timeout error.
  - Slave received a valid header
  - Master/Slave transmission completed
- The callback function, which is implemented by user, handles each event case by case following user concept. Its type is `<IpName>_Lin_Ip_CallbackType`, the exact type name depends on peripheral used. Ex: If Lpuart peripheral is being used, the type name is `Lpuart_Lin_Ip_CallbackType`. The callback function is a void (non-return) function and has 2 input parameter: Instance and LinState. Instance is offset of the hardware peripheral. LinState is a pointer to structure `<IpName>_Lin_Ip_StateStructType`, that contains the current state of the Lin instance (please refer to `<IpName>_Lin_Ip_StateStructType` structure description on Types Reference chapter of each IPV).

Name of the callback function can be modified on field "User Callback function".

The image shows a configuration window for the LIN driver. It has several settings:
 

- Lin Channel Wake Up Support**: A checkbox that is currently unchecked.
- User Callback function**: A text field containing the value 'callback'. This row is highlighted in yellow.
- Auto baud rate**: A checkbox that is currently unchecked.
- Count stop bit**: A dropdown menu showing 'ONE\_STOP\_BIT'.
- Checksum Calculate Disable**: A checkbox that is currently unchecked.
- LinClockFunctionalGroupRef**: A dropdown menu showing 'BOARD\_BootClockRUN'.

Figure 3.17 Callback function configuration.



### 3.6.7 How to configure a FLEXIO channel

In the Mcl plugin the Mcl Flexio Common must be checked.

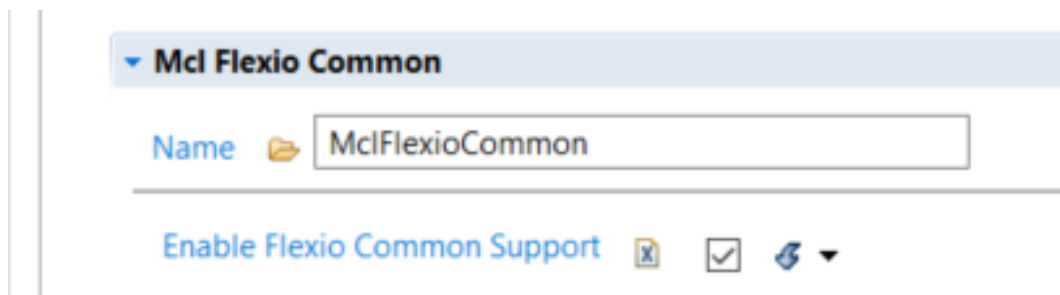


Figure 3.18 Mcl Flexio Common check.

After enabling the Mcl Flexio Common, go to Mcl/MclConfig/FlexioCommon/FlexioCommon\_0/FlexioMclLogicChannels and configure two Flexio Logic Channels. Make sure the selection of pins and channels is not the same between the two logic channels. One good helpful piece of advice is to use the Name field to keep easily track which channel is configured for Reception (Rx) and which one is used for Transmission (Tx). The pin chosen is the pin of the FLEXIO module which must be also added in the Port / Siul configuration.

Flexio Logic Channels			
Index	Name	Flexio Channel	Flexio P...
0	FlexioMclLogicChannels_Rx	CHANNEL_0	PIN_5
1	FlexioMclLogicChannels_Tx	CHANNEL_1	PIN_4

Figure 3.19 Mcl Flexio Logic channels.

In the Lin component you must choose the Lin hardware channel one of the FLEXIO channels. The next step is to enable the Lin Flexio Rx Channel and Lin Flexio Tx Channel and select the Mcl Flexio Logic Channels configured in the Mcl component. The same channel must **NOT** be configured for both Tx and Rx.

## Driver

### Note

Not enabling the Flexio Rx and Tx channel when using FLEXIO hardware unit is leading to build errors.

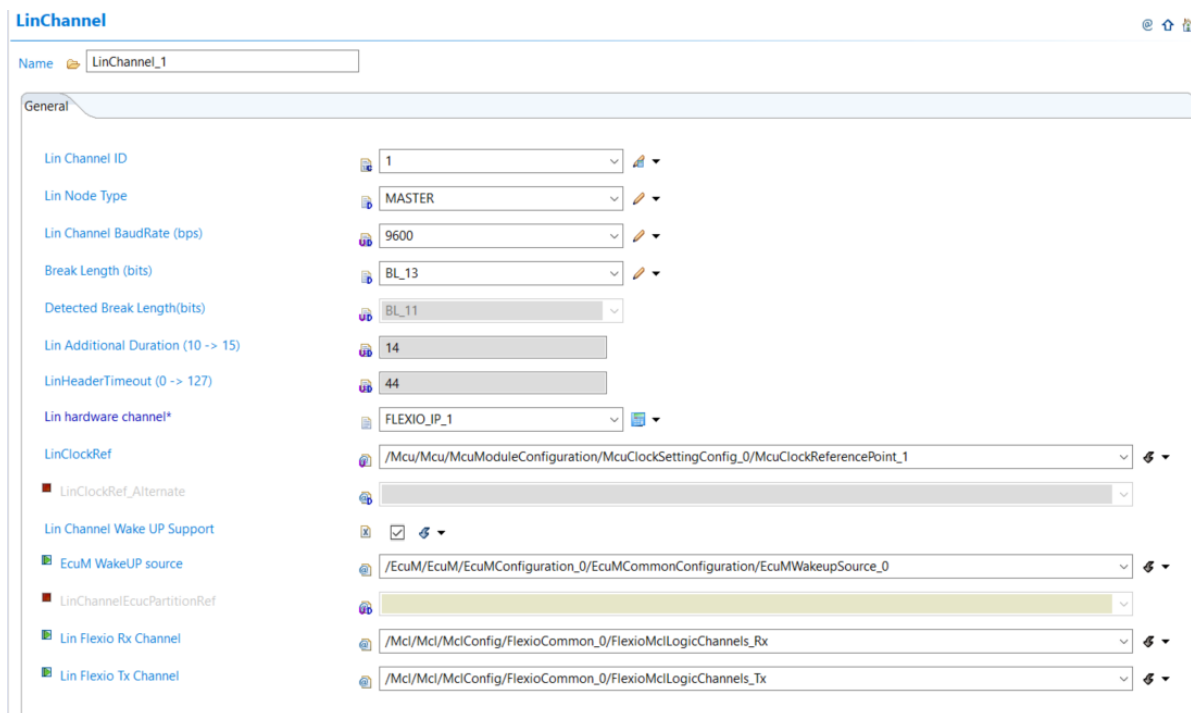


Figure 3.20 Configuration for LIN channel using FLEXIO hardware unit.

### Note

All the above tips apply also for Design Studio configuration.

## 3.7 Runtime errors

The Lin driver generates the following DEM errors at runtime.

Function	Error Code	Condition triggering the error
Lin_GoToSleep()	Lin_E_TIMEOUT	Timeout caused by hardware error waiting for cancellation of current byte. No sleep command will be sent , and Lin driver will not enter sleep state.
Lin_GoToSleepInternal()	Lin_E_TIMEOUT	Timeout caused by hardware error waiting for cancellation of current byte. Lin driver will not enter sleep state.
Lin_SendFrame()	Lin_E_TIMEOUT	Timeout caused by hardware error waiting for cancellation of current byte. New frame is not sent.

### 3.8 Symbolic Names Disclaimer

All containers having `symbolicNameValue` set to `TRUE` in the AUTOSAR schema will generate defines like:

```
#define <Mip>Conf_<Container_ShortName>_<Container_ID>
```

For this reason it is forbidden to duplicate the names of such containers across the RTD configurations or to use names that may trigger other compile issues (e.g. match existing `#ifdefs` arguments).

## Chapter 4

### Tresos Configuration Plug-in

This chapter describes the Tresos configuration plug-in for the driver. All the parameters are described below.

- Module [Lin](#)
  - Container [LinGlobalConfig](#)
    - \* Container [LinChannel](#)
      - Parameter [LinChannelId](#)
      - Parameter [LinNodeType](#)
      - Parameter [LinChannelBaudRate](#)
      - Parameter [BreakLength](#)
      - Parameter [DetectedBreakLength](#)
      - Parameter [LinResponseTimeout](#)
      - Parameter [LinHeaderTimeout](#)
      - Parameter [LinHwChannel](#)
      - Parameter [LinChannelWakeupSupport](#)
      - Reference [LinClockRef](#)
      - Reference [LinClockRef\\_Alternate](#)
      - Reference [LinChannelEcuMWakeupSource](#)
      - Reference [LinChannelEcucPartitionRef](#)
      - Reference [LinFlexioRxControllerRef](#)
      - Reference [LinFlexioTxControllerRef](#)
  - Container [LinGeneral](#)
    - \* Parameter [LinMulticoreSupport](#)
    - \* Parameter [LinDevErrorDetect](#)
    - \* Parameter [LinIndex](#)
    - \* Parameter [LinTimeoutMethod](#)
    - \* Parameter [LinTimeoutDuration](#)
    - \* Parameter [LinVersionInfoApi](#)
    - \* Reference [LinEcucPartitionRef](#)
  - Container [AutosarExt](#)
    - \* Parameter [LinDisableDemReportErrorStatus](#)
    - \* Parameter [LinFrameTimeoutDisable](#)
    - \* Parameter [LinEnableUserModeSupport](#)

- \* Parameter [LinLpuartStartTimerNotification](#)
- \* Parameter [LinLpuartStopTimerNotification](#)
- \* Parameter [LinFlexioStartTimerNotification](#)
- \* Parameter [LinFlexioStopTimerNotification](#)
- \* Parameter [LinLpuartWakeupTimerNotification](#)
- Container [LinDemEventParameterRefs](#)
  - \* Reference [LIN\\_E\\_TIMEOUT](#)
- Container [CommonPublishedInformation](#)
  - \* Parameter [ArReleaseMajorVersion](#)
  - \* Parameter [ArReleaseMinorVersion](#)
  - \* Parameter [ArReleaseRevisionVersion](#)
  - \* Parameter [ModuleId](#)
  - \* Parameter [SwMajorVersion](#)
  - \* Parameter [SwMinorVersion](#)
  - \* Parameter [SwPatchVersion](#)
  - \* Parameter [VendorApiInfix](#)
  - \* Parameter [VendorId](#)

## 4.1 Module Lin

Configuration of the Lin (Local Interconnect Network) module.

Included containers:

- [LinGlobalConfig](#)
- [LinGeneral](#)
- [AutosarExt](#)
- [LinDemEventParameterRefs](#)
- [CommonPublishedInformation](#)

Property	Value
type	ECUC-MODULE-DEF
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantSupport	true
supportedConfigVariants	VARIANT-POST-BUILD, VARIANT-PRE-COMPILE

## 4.2 Container LinGlobalConfig

This container contains the global configuration parameter of the Lin driver. This container is a MultipleConfigurationContainer i.e. this container and its sub-containers exist once per configuration set.

Included subcontainers:

- [LinChannel](#)

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A

## 4.3 Container LinChannel

This container contains the configuration (parameters) of the LIN Controller(s).

Note: "User should use unique names for naming the LIN channels across different LinGlobalConfig Sets."

Included subcontainers:

- None

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF
lowerMultiplicity	1
upperMultiplicity	Infinite
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE

## 4.4 Parameter LinChannelId

Identifies the LIN channel. Replaces LIN\_CHANNEL\_INDEX\_NAME from the LIN SWS.

Property	Value
type	ECUC-INTEGER-PARAM-DEF

Property	Value
origin	AUTOSAR_ECUC
symbolicNameValue	true
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	1
max	65535
min	0

## 4.5 Parameter LinNodeType

LinNodeType

Autosar Requirements: ECUC\_Lin\_00191

Specifies the LIN node type of this channel: Master or Slave node

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	MASTER
literals	['MASTER', 'SLAVE']

## 4.6 Parameter LinChannelBaudRate

LinChannelBaudRate

Autosar Requirements: ECUC\_Lin\_00180

Specifies the baud rate of the LIN channel in 'bps'. Valid range: 1000..20000.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
default Value	9600
max	20000
min	1000

## 4.7 Parameter BreakLength

Defines the break length in bits.

Note:

This Parameter is an Implementation Specific Parameter.

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: PRE-COMPILE
default Value	BL_13
literals	['BL_13']

## 4.8 Parameter DetectedBreakLength

Defines the break length in bits which can detect by Lin node in Slave mode.



Note:

This Parameter is an Implementation Specific Parameter.

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
defaultValue	BL_11
literals	['BL_11']

## 4.9 Parameter LinResponseTimeout

Response timeout value.

This is the response timeout duration (in bit time) for 1 byte.

The default value is 14, corresponding to  $T\_Response\_Maximum = 1.4 \times T\_Response\_Nominal$ . Here, 1.4 corresponds to  $LinResponseTimeout/10$ .

This node is only configured if AutosarExt/LinFrameTimeoutDisable is false.

Note:

This Parameter is an Implementation Specific Parameter.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	14
max	20
min	3

## 4.10 Parameter LinHeaderTimeout

Header timeout value.

This field contains the header timeout duration (in bit time) after slave detect break character.

Example:

- $\text{Header\_nominal} = 13 + 2 + 10 + 10 = 35$  (Bit time)
- Additional 40% duration compared to the nominal transmission time, so  $\text{Header\_max\_time} = 1.4 * \text{Header\_nominal} = 49$  (Bit time)
- Taking into account a possible 14% clock deviation, header\_max seen is  $49 * 1.14 = 56$  (Bit time)
- The counter start after detection break - 11 Bit time, the header timeout value is  $56 - 11 = 45$

This node is only configured if AutosarExt/LinFrameTimeoutDisable is false and LinNodeType is SLAVE.

Note:

This Parameter is an Implementation Specific Parameter.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	44
max	127
min	0

## 4.11 Parameter LinHwChannel

Selects the Lin Hardware Channel.

Note:

This Parameter is an Implementation Specific Parameter.

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: PRE-COMPILE
defaultValue	LPUART_IP_0
literals	['LPUART_IP_0', 'LPUART_IP_1', 'LPUART_IP_2', 'LPUART_IP_3', 'LPUART_IP_4', 'LPUART_IP_5', 'LPUART_IP_6', 'LPUART_IP_7', 'LPUART_IP_8', 'LPUART_IP_9', 'LPUART_IP_10', 'LPUART_IP_11', 'LPUART_IP_12', 'LPUART_IP_13', 'LPUART_IP_14', 'LPUART_IP_15', 'FLEXIO_IP_0', 'FLEXIO_IP_1', 'FLEXIO_IP_2', 'FLEXIO_IP_3']

## 4.12 Parameter LinChannelWakeupSupport

LinChannelWakeupSupport

Autosar Requirements: LIN182\_Conf

Specifies if the LIN hardware channel supports wake up functionality.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	false

## 4.13 Reference LinClockRef

Reference to the LIN clock source configuration, which is set in the MCU driver configuration.

Property	Value
type	ECUC-REFERENCE-DEF
origin	AUTOSAR_ECUC
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
requiresSymbolicNameValue	False
destination	/AUTOSAR/EcuDefs/Mcu/McuModuleConfiguration/McuClockSetting↔ Config/McuClockReferencePoint

#### 4.14 Reference LinClockRef\_Alternate

Alternate reference to the LIN clock source configuration, which is set in the MCU driver configuration, used in Low Power Mode.

Property	Value
type	ECUC-REFERENCE-DEF
origin	NXP
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
requiresSymbolicNameValue	False
destination	/AUTOSAR/EcuDefs/Mcu/McuModuleConfiguration/McuClockSetting↔ Config/McuClockReferencePoint

#### 4.15 Reference LinChannelEcuMWakeupSource

This parameter contains a reference to the Wakeup Source for this controller as defined in the ECU State Manager.

Property	Value
type	ECUC-REFERENCE-DEF
origin	AUTOSAR_ECUC

Property	Value
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
requiresSymbolicNameValue	False
destination	/AUTOSAR/EcuDefs/EcuM/EcuMConfiguration/EcuMCommon← Configuration/EcuMWakeupSource

## 4.16 Reference LinChannelEcucPartitionRef

Maps one single Lin channel to zero or one ECUC partitions.

The ECUC partition referenced is a subset of the ECUC partitions where the Lin driver is mapped to.

Property	Value
type	ECUC-REFERENCE-DEF
origin	AUTOSAR_ECUC
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	true
multiplicityConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: PRE-COMPILE
postBuildVariantValue	true
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: PRE-COMPILE
requiresSymbolicNameValue	False
destination	/AUTOSAR/EcuDefs/EcuC/EcucPartitionCollection/EcucPartition

## 4.17 Reference LinFlexioRxControllerRef

Lin Flexio Rx Channel Reference

Reference to the Flexio Controller configure for the Reception

Property	Value
type	ECUC-CHOICE-REFERENCE-DEF

Property	Value
origin	NXP
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
requiresSymbolicNameValue	False
destinations	['/TS_T40D34M30I0R0/Mcl/MclConfig/FlexioCommon/FlexioMclLogicChannels']

## 4.18 Reference LinFlexioTxControllerRef

Lin Flexio Tx Channel Reference

Reference to the Flexio Controller configure for the Transmission

Property	Value
type	ECUC-CHOICE-REFERENCE-DEF
origin	NXP
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
requiresSymbolicNameValue	False
destinations	['/TS_T40D34M30I0R0/Mcl/MclConfig/FlexioCommon/FlexioMclLogicChannels']

## 4.19 Container LinGeneral

LinGeneral

Autosar Requirements: ECUC\_Lin\_00183

This container contains the parameters related to each LIN Driver Unit.

This container is a MultipleConfigurationContainer, i.e. this container and

its sub-containers exit once per configuration set.

Included subcontainers:

- None

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A

## 4.20 Parameter LinMulticoreSupport

This parameter determine multi-core feature will be used in Lin driver.

If LinMulticoreSupport is disabled, then for all the variants no partition shall be defined.

If LinMulticoreSupport is enabled, at least one EcucPartition needs to be defined (in all variants).

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	false

## 4.21 Parameter LinDevErrorDetect

LinDevErrorDetect

Autosar Requirements: ECUC\_Lin\_00066

Switches the Default Error Detection and Notification ON or OFF.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
default Value	false

## 4.22 Parameter LinIndex

Autosar Requirements: ECUC\_Lin\_00179

Specifies the InstanceId of this module instance. If only one instance is present it shall have the Id 0.

Note, this parameter is not used in the current implementation.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
default Value	0
max	255
min	0

## 4.23 Parameter LinTimeoutMethod

LinTimeoutMethod

Configures the timeout method.



Based on this selection a certain timeout method from OsIf will be used in the driver.

This parameter is used to select between different OsIf counter implementations. For additional details, please refer to the OsIf documentation.

Note: If SystemTimer or CustomTimer are selected make sure the corresponding timer is enabled in OsIf General configuration.

Note: Implementation Specific Parameter.

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	OSIF_COUNTER_DUMMY
literals	['OSIF_COUNTER_DUMMY', 'OSIF_COUNTER_SYSTEM', 'OSIF_COUNTER_CUSTOM']

## 4.24 Parameter LinTimeoutDuration

Specifies the maximum number of loops for blocking function until a timeout is raised in short term wait loops. If LinTimeoutMethod is OSIF\_COUNTER\_SYSTEM or OSIF\_COUNTER\_CUSTOM, LinTimeoutDuration is in microsecond value. If LinTimeoutMethod is OSIF\_COUNTER\_DUMMY, the LinTimeoutDuration is number of wait loop.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	1000
max	65535
min	0

## 4.25 Parameter LinVersionInfoApi

LinVersionInfoApi

Autosar Requirements: ECUC\_Lin\_00067

Switches the Lin\_GetVersionInfo function ON or OFF.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
default Value	false

## 4.26 Reference LinEcucPartitionRef

Maps the Lin driver to zero or multiple ECUC partitions to make the modules API available in this partition.

The Lin driver will operate as an independent instance in each of the partitions.

Property	Value
type	ECUC-REFERENCE-DEF
origin	AUTOSAR_ECUC
lowerMultiplicity	0
upperMultiplicity	Infinite
postBuildVariantMultiplicity	true
multiplicityConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: PRE-COMPILE
postBuildVariantValue	true
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: PRE-COMPILE
requiresSymbolicNameValue	False
destination	/AUTOSAR/EcucDefs/EcuC/EcucPartitionCollection/EcucPartition

## 4.27 Container AutosarExt

AutosarExt

Autosar Requirements:

This container contains the global configuration parameters of the Non-Autosar Lin driver.

This container is a MultipleConfigurationContainer, i.e. this container and

its sub-containers exist once per configuration set.

Included subcontainers:

- None

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A

## 4.28 Parameter LinDisableDemReportErrorStatus

LinDisableDemReportErrorStatus

Switches the Diagnostic Error Reporting and Notification OFF.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	false

## 4.29 Parameter LinFrameTimeoutDisable

LinFrameTimeoutDisable

The master will accept the frame that is longer than TFrame\_Maximum.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	true

## 4.30 Parameter LinEnableUserModeSupport

When this parameter is enabled, the MDL module will adapt to run from User Mode, with the following measures:

- a) configuring REG\_PROT for ABC1, ABC2 IPs so that the registers under protection can be accessed from user mode by setting UAA bit in REG\_PROT\_GCR to 1
- b) using 'call trusted function' stubs for all internal function calls that access registers requiring supervisor mode.
- c) other module specific measures

for more information, please see chapter 5.7 User Mode Support in IM

Note: Implementation Specific Parameter.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	false

### 4.31 Parameter LinLpuartStartTimerNotification

Lin Start Timer Notification for the checking frame timeout error when Lin over Lpuart used.

This parameter is a reference to a notification function to start timer when reception has just begun.

Note: This parameter should be configured when LinFrameTimeoutDisable is OFF

Property	Value
type	ECUC-FUNCTION-NAME-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	NULL_PTR

### 4.32 Parameter LinLpuartStopTimerNotification

Lin Stop Timer Notification for the checking frame timeout error when Lin over Lpuart used.

This parameter is a reference to a notification function to stop timer.

Note: This parameter should be configured when LinFrameTimeoutDisable is OFF

Property	Value
type	ECUC-FUNCTION-NAME-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	NULL_PTR

### 4.33 Parameter LinFlexioStartTimerNotification

Lin Start Timer Notification for the checking frame timeout error when Lin over Flexio used.

This parameter is a reference to a notification function to start timer when reception has just begun.

Note: This parameter should be configured when LinFrameTimeoutDisable is OFF

Property	Value
type	ECUC-FUNCTION-NAME-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	NULL_PTR

### 4.34 Parameter LinFlexioStopTimerNotification

Lin Stop Timer Notification for the checking frame timeout error when Lin over Flexio used.

This parameter is a reference to a notification function to stop timer when reception finished or timeout occurred.

Note: This parameter should be configured when LinFrameTimeoutDisable is OFF

Property	Value
type	ECUC-FUNCTION-NAME-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	NULL_PTR

## 4.35 Parameter LinLpuartWakeupTimerNotification

Lin Lpuart Wake UP Timer Notification for the checking Wake Up pulse when Lin over Lpuart used.

This parameter is a reference to a notification function to start/stop timer corresponding falling/rising of Wakeup pulse.

Note: This parameter have to be configured when sending/receiving the wakeup pulse.

Property	Value
type	ECUC-FUNCTION-NAME-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	NULL_PTR

## 4.36 Container LinDemEventParameterRefs

Container for the references to DemEventParameter elements which shall be invoked using the API Dem\_SetEventStatus API in case the corresponding error occurs. The EventId is taken from the referenced DemEventParameter's DemEventId value. The standardized errors are provided in the container and can be extended by vendor specific error references.

Included subcontainers:

- None

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE

### 4.37 Reference LIN\_E\_TIMEOUT

Reference to the DemEventParameter which shall be issued when the error "Timeout caused by hardware error" has occurred.

Property	Value
type	ECUC-REFERENCE-DEF
origin	AUTOSAR_ECUC
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
requiresSymbolicNameValue	False
destination	/AUTOSAR/EcucDefs/Dem/DemConfigSet/DemEventParameter

### 4.38 Container CommonPublishedInformation

Common container, aggregated by all modules. It contains published information about vendor and versions.

Included subcontainers:

- None

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A

### 4.39 Parameter ArReleaseMajorVersion

Major version number of AUTOSAR specification on which the appropriate implementation is based on.

Property	Value
type	ECUC-INTEGER-PARAM-DEF



Property	Value
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PUBLISHED-INFORMATION
	VARIANT-PRE-COMPILE: PUBLISHED-INFORMATION
defaultValue	4
max	4
min	4

#### 4.40 Parameter ArReleaseMinorVersion

Minor version number of AUTOSAR specification on which the appropriate implementation is based on.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PUBLISHED-INFORMATION
	VARIANT-PRE-COMPILE: PUBLISHED-INFORMATION
defaultValue	7
max	7
min	7

#### 4.41 Parameter ArReleaseRevisionVersion

Revision version number of AUTOSAR specification on which the appropriate implementation is based on.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false

Property	Value
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PUBLISHED-INFORMATION
	VARIANT-PRE-COMPILE: PUBLISHED-INFORMATION
defaultValue	0
max	0
min	0

## 4.42 Parameter ModuleId

Module ID of this module from Module List.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PUBLISHED-INFORMATION
	VARIANT-PRE-COMPILE: PUBLISHED-INFORMATION
defaultValue	82
max	82
min	82

## 4.43 Parameter SwMajorVersion

Major version number of the vendor specific implementation of the module. The numbering is vendor specific.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1

Property	Value
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PUBLISHED-INFORMATION
	VARIANT-PRE-COMPILE: PUBLISHED-INFORMATION
defaultValue	3
max	3
min	3

#### 4.44 Parameter SwMinorVersion

Minor version number of the vendor specific implementation of the module. The numbering is vendor specific.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PUBLISHED-INFORMATION
	VARIANT-PRE-COMPILE: PUBLISHED-INFORMATION
defaultValue	0
max	0
min	0

#### 4.45 Parameter SwPatchVersion

Patch level version number of the vendor specific implementation of the module. The numbering is vendor specific.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A

Property	Value
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PUBLISHED-INFORMATION
	VARIANT-PRE-COMPILE: PUBLISHED-INFORMATION
defaultValue	0
max	0
min	0

## 4.46 Parameter VendorApiInfix

In driver modules which can be instantiated several times on a single ECU, BSW00347 requires that the name of APIs is extended by the VendorId and a vendor specific name.

This parameter is used to specify the vendor specific name. In total, the implementation specific name is generated as follows:

<ModuleName>\_\_>VendorId>\_\_<VendorApiInfix>.

E.g. assuming that the VendorId of the implementor is 123 and the implementer chose a VendorApiInfix of "v11r456" a api name Can\_Write defined in the SWS will translate to Can\_123\_v11r456Write.

This parameter is mandatory for all modules with upper multiplicity > 1. It shall not be used for modules with upper multiplicity =1.

Property	Value
type	ECUC-STRING-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-POST-BUILD: PUBLISHED-INFORMATION
	VARIANT-PRE-COMPILE: PUBLISHED-INFORMATION
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PUBLISHED-INFORMATION
	VARIANT-PRE-COMPILE: PUBLISHED-INFORMATION
defaultValue	LPUART_FLEXIO

## 4.47 Parameter VendorId

Vendor ID of the dedicated implementation of this module according to the AUTOSAR vendor list.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PUBLISHED-INFORMATION
	VARIANT-PRE-COMPILE: PUBLISHED-INFORMATION
defaultValue	43
max	43
min	43

This chapter describes the Tresos configuration plug-in for the *driver* Driver. The most of the parameters are described below.

## 4.48 Configuration elements of Lin

Included forms:

- POST\_BUILD\_VARIANT\_USED
- IMPLEMENTATION\_CONFIG\_VARIANT
- AutosarExt
- LinGeneral
- LinDemEventParameterRefs
- LinGlobalConfig
- CommonPublishedInformation

## 4.49 Form POST\_BUILD\_VARIANT\_USED

Indicates whether a module implementation has or plans to have (i.e., introduced at link or post-build time) new post-build variation points.

## 4.50 Attribute IMPLEMENTATION\_CONFIG\_VARIANT detailed description

Property	Value
Label	Post Build Variant Used
Type	BOOLEAN
Origin	AUTOSAR_ECUC
Symbolic Name	true
Default	false

## 4.51 Form IMPLEMENTATION\_CONFIG\_VARIANT

There are two variant mode to choose:

- **VARIANT-PRE-COMPILE:** Only parameters with Pre-compile time configuration are allowed in this variant.
- **VARIANT-POST-BUILD:** Parameters with Pre-compile time, Link time and Postbuild time are allowed in this variant.

## 4.52 Attribute IMPLEMENTATION\_CONFIG\_VARIANT detailed description

Property	Value
Label	Configuration Variant
Type	ENUMERATION
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	VariantPostBuild
Range	VariantPostBuild VariantPreCompile

## 4.53 Form AutosarExt

This container contains the global configuration parameters of the AutosarExt *driver* driver. The container is a MultipleConfigurationContainer. It and its subcontainers exist once per configuration set.

**4.53.1 LinDisableDemReportErrorStatus (AutosarExt)** Switches the Diagnostic Error Reporting and Notification OFF

## 4.54 Attribute LinDisableDemReportErrorStatus (AutosarExt) detailed description

Property	Value
Label	Lin Disable Production Error Reporting
Type	BOOLEAN
Origin	Custom
Symbolic Name	false
Default	false

**4.54.1 LinEnableUserModeSupport (AutosarExt)** When LinEnableUserModeSupport is ON, the Lin module will adapt to run from User Mode. Note Lin module does not include registers protection. So, it is accessible to all registered in any public mode.

## 4.55 Attribute LinEnableUserModeSupport (AutosarExt) detailed description

Property	Value
Label	Lin Enable User Mode Support
Type	BOOLEAN
Origin	Custom
Symbolic Name	false
Default	false

## 4.56 Form LinGeneral

Autosar Requirements: ECUC\_Lin\_00183.

This container contains the parameters related to each LIN Driver Unit.

**4.56.1 LinMulticoreSupport (LinGeneral)** This parameter determine multi-core feature will be used in Lin driver.

- If LinMulticoreSupport is disabled, then for all the variants no partition shall be defined.
- If LinMulticoreSupport is enabled, at least one EcucPartition needs to be defined (in all variants).

## 4.57 Attribute LinMulticoreSupport (LinGeneral) detailed description

Property	Value
Label	Lin Multicore Support
Type	BOOLEAN
Origin	Custom
Symbolic Name	false
Default	false

#### 4.57.1 LinDevErrorDetect (LinGeneral) Autosar Requirements: ECUC\_Lin\_00066.

Switches the Default Error Detection and Notification ON or OFF.

### 4.58 Attribute LinDevErrorDetect (LinGeneral) detailed description

Property	Value
Label	Lin Development Error Detection
Type	BOOLEAN
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	true

#### 4.58.1 LinIndex (LinGeneral) Autosar Requirements: ECUC\_Lin\_00179.

Specifies the InstanceId of this module instance. If only one instance is present it shall have the Id 0.

Note

This parameter is not used in the current implementation.

### 4.59 Attribute LinIndex (LinGeneral) detailed description

Property	Value
Label	InstanceId
Type	INTEGER
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	0
Invalid	Range <=255 >=0

#### 4.59.1 LinTimeoutDuration (LinGeneral) Autosar Requirements: ECUC\_Lin\_00093.

Specifies the maximum number of loops for blocking function until a timeout is raised in short term wait loops

### 4.60 Attribute LinTimeoutDuration (LinGeneral) detailed description

Property	Value
Label	Lin Timeout Duration



Property	Value
Type	INTEGER
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	1000
Invalid	Range <=65535 >=0

#### 4.60.1 LinVersionInfoApi (LinGeneral) Autosar Requirements: ECUC\_Lin\_00067.

Switches the Lin\_GetVersionInfo function ON or OFF.

### 4.61 Attribute LinVersionInfoApi (LinGeneral) detailed description

Property	Value
Label	Provide Lin VersionInfo Api
Type	BOOLEAN
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	true

#### 4.61.1 LinEcucPartitionRef (LinGeneral) Autosar Requirements: ECUC\_Lin\_00192.

Maps the Lin driver to zero or multiple ECUC partitions to make the modules API available in this partition. The Lin driver will operate as an independent instance in each of the partitions.

### 4.62 Attribute LinVersionInfoApi (LinGeneral) detailed description

Property	Value
Label	Provide Lin VersionInfo Api
Type	BOOLEAN
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	true

### 4.63 Form LinDemEventParameterRefs

Container for the references to DemEventParameter elements which shall be invoked using the API Dem\_SetEvent↔ Status i n case the corresponding error occurs. The EventId is taken from the referenced DemEventParameter's DemEventId symbolic value. The standardized errors are provided in this container and can be extended by vendor-specific error references.

#### 4.63.1 LIN\_E\_TIMEOUT (LinDemEventParameterRefs)

### 4.64 Attribute LIN\_E\_TIMEOUT (LinDemEventParameterRefs) detailed description

Property	Value
Label	Lin Timeout Dem Error
Type	SYMBOLIC-NAME-REFERENCE
Origin	AUTOSAR_ECUC
Enable	true

### 4.65 Form LinGlobalConfig

The LinGlobalConfig container contains the global configuration parameter of the Lin driver. The container is a MultipleConfigurationContainer, i.e. container and its subcontainers exist once per configuration set.

#### 4.65.1 Form LinChannel This container contains the configuration (parameters) of the LIN Controller(s).

Note User should use unique names for naming the LIN channels across different LinGlobalConfig Sets.

##### 4.65.1.1 LinChannelId (LinChannel)

Identifies the LIN channel. Replaces LIN\_CHANNEL\_INDEX\_NAME from the LIN SWS.

### 4.66 Attribute LinChannelId (LinChannel) detailed description

Property	Value
Label	Lin Channel ID
Type	INTEGER
Origin	AUTOSAR_ECUC
Symbolic Name	true
Invalid	Range $\geq 0 \leq (N - 1)$ (N is number of configured channels)

#### 4.66.0.1 LinChannelBaudRate (LinChannel)

Autosar Requirements: ECUC\_Lin\_00180.

Specifies the baud rate of the LIN channel in 'bps'. Valid range: 1000..20000.

## 4.67 Attribute LinChannelBaudRate (LinChannel) detailed description

Property	Value
Label	Lin Channel BaudRate (bps)
Type	INTEGER
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	9600
Invalid	Range $\geq 0 \leq (N - 1)$ (N is number of configured channels)

### 4.67.0.1 LinHwChannel (LinChannel)

Selects the physical LIN Channel. This Parameter is an Implementation Specific Parameter

## 4.68 Attribute LinHwChannel (LinChannel) detailed description

Property	Value
Label	Lin hardware channel
Type	ENUMERATION
Origin	Custom
Symbolic Name	false

### 4.68.0.1 LinClockRef (LinChannel)

Reference to the LIN clock source configuration, which is set in the MCU driver configuration.

## 4.69 Attribute LinClockRef (LinChannel) detailed description

Property	Value
Label	LinClockRef
Type	REFERENCE
Origin	AUTOSAR_ECUC
Symbolic Name	false

### 4.69.0.1 LinClockRef\_Alternate (LinChannel)

Alternate reference to the LIN clock source configuration, which is set in the MCU driver configuration, used in Low Power Mode.

## 4.70 Attribute LinClockRef\_Alternate (LinChannel) detailed description

Property	Value
Label	LinClockRef_Alternate
Type	REFERENCE
Origin	Custom
Symbolic Name	false
Enable	true

### 4.70.0.1 LinChannelWakeupSupport (LinChannel)

Autosar Requirements: ECUC\_Lin\_00182,

Specifies if the LIN hardware channel supports wake up functionality.

## 4.71 Attribute LinChannelWakeupSupport (LinChannel) detailed description

Property	Value
Label	Lin Channel Wake UP support
Type	BOOLEAN
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	false

### 4.71.0.1 LinChannelEcuMWakeupSource (LinChannel)

This parameter contains a reference to the Wakeup Source for this controller as defined in the ECU State Manager.

## 4.72 Attribute LinChannelEcuMWakeupSource (LinChannel) detailed description

Property	Value
Label	EcuM WakeUP source
Type	SYMBOLIC-NAME-REFERENCE
Origin	AUTOSAR_ECUC
Symbolic Name	false
Enable	true

#### 4.72.0.1 LinChannelEcucPartitionRef (LinChannel)

Maps one single Lin channel to zero or one ECUC partitions. The ECUC partition referenced is a subset of the ECUC partitions where the Lin driver is mapped to.

### 4.73 Attribute LinChannelEcucPartitionRef (LinChannel) detailed description

Property	Value
Label	LinChannelEcucPartitionRef
Type	SYMBOLIC-NAME-REFERENCE
Origin	AUTOSAR_ECUC
Symbolic Name	false
Enable	true



# Chapter 5

## Module Index

### 5.1 Software Specification

Here is a list of all modules:

FLEXIO_IP . . . . .	59
LIN Driver . . . . .	73
LIN . . . . .	88
Lpuart Lin IPL . . . . .	90

## Chapter 6

### Module Documentation

#### 6.1 FLEXIO\_IP

##### 6.1.1 Detailed Description

##### Data Structures

- struct [Flexio\\_Lin\\_Ip\\_PduType](#)  
*This Type is used to provide PID, checksum model, response type of the frame, data length and SDU pointer from the LIN Interface to the Flexio Lin Ip driver. [More...](#)*
- struct [Flexio\\_Lin\\_Ip\\_StateStructType](#)  
*Flexio Lin Ip driver state structure type. It is internal for the driver use only. Implements : Flexio\_Lin\_Ip\_StateStructType\_Class. [More...](#)*
- struct [Flexio\\_Lin\\_Ip\\_UserConfigType](#)  
*Flexio Lin Ip driver User configuration structure type @Implements : Flexio\_Lin\_Ip\_UserConfigType\_Class. [More...](#)*

##### Macros

- [#define FLEXIO\\_LIN\\_IP\\_SLAVE](#)  
*Macro that specifies usage of a SLAVE note.*
- [#define FLEXIO\\_LIN\\_IP\\_MASTER](#)  
*Macro that specifies usage of a SLAVE note.*

##### Types Reference

- typedef void(\* [Flexio\\_Lin\\_Ip\\_CallbackType](#)) (const uint8 Instance, const [Flexio\\_Lin\\_Ip\\_StateStructType](#) \*LinState)  
*LIN Driver callback function type.*

## Enum Reference

- enum [Flexio\\_Lin\\_Ip\\_EventIdType](#)  
*Defines types for an enumerating event related to an Identifier.*
- enum [Flexio\\_Lin\\_Ip\\_NodeStateType](#)  
*Define type for an enumerating LIN Node state.*
- enum [Flexio\\_Lin\\_Ip\\_StatusType](#)  
*Define type function possible return values.*
- enum [Flexio\\_Lin\\_Ip\\_FrameCsModelType](#)  
*Define type for check sum type of the frame.*
- enum [Flexio\\_Lin\\_Ip\\_FrameResponseType](#)  
*Define type of the response part of frame.*
- enum [Flexio\\_Lin\\_Ip\\_TransferStatusType](#)  
*Description: This type defines a range of transfer status.*

## Function Reference

- [Flexio\\_Lin\\_Ip\\_StatusType Flexio\\_Lin\\_Ip\\_Init](#) (const uint8 Channel, const [Flexio\\_Lin\\_Ip\\_UserConfigType](#) \*UserConfig)  
*Initializes an LIN\_FLEXIO channel for LIN Network.*
- void [Flexio\\_Lin\\_Ip\\_GoToIdleState](#) (const uint8 Channel)  
*Puts current LIN node to Idle state This function changes current node state to FLEXIO\_LIN\_IP\_NODE\_STATE\_IDLE.*
- [Flexio\\_Lin\\_Ip\\_StatusType Flexio\\_Lin\\_Ip\\_AbortTransferData](#) (const uint8 Channel)  
*Aborts an on-going non-blocking transmission/reception.*
- [Flexio\\_Lin\\_Ip\\_StatusType Flexio\\_Lin\\_Ip\\_GoToSleepMode](#) (const uint8 Channel)  
*This function puts current node to sleep mode.*
- [Flexio\\_Lin\\_Ip\\_StatusType Flexio\\_Lin\\_Ip\\_SendWakeupSignal](#) (const uint8 Channel)  
*Sends a wakeup signal through the LIN\_FLEXIO interface.*
- [Flexio\\_Lin\\_Ip\\_StatusType Flexio\\_Lin\\_Ip\\_Deinit](#) (const uint8 Channel)  
*De-initialize a FLEXIO LIN channel.*
- [Flexio\\_Lin\\_Ip\\_TransferStatusType Flexio\\_Lin\\_Ip\\_GetStatus](#) (const uint8 Channel, const uint8 \*\*Buffer)  
*Returns the status of an on going transmission.*
- [Flexio\\_Lin\\_Ip\\_StatusType Flexio\\_Lin\\_Ip\\_SendFrame](#) (const uint8 Channel, const [Flexio\\_Lin\\_Ip\\_PduType](#) \*PduInfo)  
*Sends a LIN frame as master or a response as a slave.*
- void [Flexio\\_Lin\\_Ip\\_TimerExpiredService](#) (const uint8 Channel)  
*This is callback function for Timer Interrupt Handler. Users shall initialize a timer (for example FTM) and the time period in microseconds will be set by the driver via LinFlexioStartTimerNotification. In timer IRQ handler, call this function.*

### 6.1.2 Data Structure Documentation



### 6.1.2.1 struct Flexio\_Lin\_Ip\_PduType

This Type is used to provide PID, checksum model, response type of the frame, data length and SDU pointer from the LIN Interface to the Flexio Lin Ip driver.

Definition at line 206 of file Flexio\_Lin\_Ip\_Types.h.

#### Data Fields

- uint8 [Pid](#)  
*LIN frame identifier.*
- [Flexio\\_Lin\\_Ip\\_FrameCsModelType](#) Cs  
*Checksum model type.*
- [Flexio\\_Lin\\_Ip\\_FrameResponseType](#) Drc  
*Response type.*
- uint8 [Dl](#)  
*Data length.*
- uint8 \* [SduPtr](#)  
*Pointer to Sdu.*

#### 6.1.2.1.1 Field Documentation

##### 6.1.2.1.1.1 Pid uint8 Pid

LIN frame identifier.

Definition at line 208 of file Flexio\_Lin\_Ip\_Types.h.

##### 6.1.2.1.1.2 Cs [Flexio\\_Lin\\_Ip\\_FrameCsModelType](#) Cs

Checksum model type.

Definition at line 209 of file Flexio\_Lin\_Ip\_Types.h.

##### 6.1.2.1.1.3 Drc [Flexio\\_Lin\\_Ip\\_FrameResponseType](#) Drc

Response type.

Definition at line 210 of file Flexio\_Lin\_Ip\_Types.h.

### 6.1.2.1.1.4 Dl uint8 Dl

Data length.

Definition at line 211 of file Flexio\_Lin\_Ip\_Types.h.

### 6.1.2.1.1.5 SduPtr uint8\* SduPtr

Pointer to Sdu.

Definition at line 212 of file Flexio\_Lin\_Ip\_Types.h.

### 6.1.2.2 struct Flexio\_Lin\_Ip\_StateStructType

Flexio Lin Ip driver state structure type. It is internal for the driver use only. Implements : Flexio\_Lin\_Ip\_StateStructType\_Class.

Definition at line 220 of file Flexio\_Lin\_Ip\_Types.h.

Data Fields

Type	Name	Description
uint8	CntByte	To count number of bytes already transmitted or received.
uint8	Checksum	Checksum byte.
volatile boolean	IsBusBusy	True if there are data, frame headers being transferred on bus.
uint8 *	SduBuf	The buffer of received data.
volatile uint8	Dl	The remaining number of bytes to be transmitted.
uint8	CurrentId	Current ID.
uint8	CurrentPid	Current PID.
<a href="#">Flexio_Lin_Ip_FrameResponseType</a>	FrameCommand	
<a href="#">Flexio_Lin_Ip_FrameCsModelType</a>	Cs	Current Checksum type.
volatile <a href="#">Flexio_Lin_Ip_EventIdType</a>	CurrentEventId	Current ID Event.
volatile <a href="#">Flexio_Lin_Ip_NodeStateType</a>	CurrentNodeState	Current Node state.
volatile <a href="#">Flexio_Lin_Ip_NodeStateType</a>	PreviousNodeState	Store previous node state when set Lin channel to idle for further processing.

### 6.1.2.3 struct Flexio\_Lin\_Ip\_UserConfigType

Flexio Lin Ip driver User configuration structure type @Implements : Flexio\_Lin\_Ip\_UserConfigType\_Class.

Definition at line 251 of file Flexio\_Lin\_Ip\_Types.h.

## Data Fields

Type	Name	Description
uint8	Instance	Flexio hardware channel for this configuration.
uint8	TxShifterId	Flexio shifter use for transmission(Tx).
uint8	TxTimerId	Flexio timer use for transmission(Tx).
uint8	TxPin	Flexio pin use for transmission(Tx).
uint8	RxShifterId	Flexio shifter use for reception(Rx).
uint8	RxTimerId	Flexio timer use for reception(Rx).
uint8	RxPin	Flexio pin use for reception(Rx).
uint8	FlexioHwInstance	Flexio hardware module number.
uint8	MasterBreakLength	Number of bits for break length.
uint8	SlaveSyncBreakLength	Number of bits detected by slave node(break length).
uint8	TimerClkSrc	Timer clock source.
uint16	Baudratedivider	Baudrate divider.
boolean	NodeFunction	Type of Lin node. It can be Master node or Slave node.
<a href="#">Flexio_Lin_Ip_CallbackType</a>	Callback	Callback function to invoke after receiving a byte or transmitting a byte.
<a href="#">Flexio_Lin_Ip_StateStructType</a> *	StateStruct	Pointer for the internal state structure of the driver.
uint32	Baudrate	Baudrate value configured.
uint8	WakeupByte	Byte will be sent to generate wakeup pulse [450us->5ms].
uint8	BaseWakeupByteDetectInverted	Byte use to check detection of wake up pulse longer than 150us.

### 6.1.3 Macro Definition Documentation

#### 6.1.3.1 FLEXIO\_LIN\_IP\_SLAVE

```
#define FLEXIO_LIN_IP_SLAVE
```

Macro that specifies usage of a SLAVE note.

This macro is used in user configuration structure to set the node type as SLAVE.

Definition at line 91 of file Flexio\_Lin\_Ip\_Types.h.

### 6.1.3.2 FLEXIO\_LIN\_IP\_MASTER

```
#define FLEXIO_LIN_IP_MASTER
```

Macro that specifies usage of a SLAVE note.

This macro is used in user configuration structure to set the node type as MASTER.

Definition at line 99 of file Flexio\_Lin\_Ip\_Types.h.

### 6.1.4 Types Reference

#### 6.1.4.1 Flexio\_Lin\_Ip\_CallbackType

```
typedef void(* Flexio_Lin_Ip_CallbackType) (const uint8 Instance, const Flexio_Lin_Ip_StateStructType *LinState)
```

LIN Driver callback function type.

Definition at line 245 of file Flexio\_Lin\_Ip\_Types.h.

### 6.1.5 Enum Reference

#### 6.1.5.1 Flexio\_Lin\_Ip\_EventIdType

```
enum Flexio_Lin_Ip_EventIdType
```

Defines types for an enumerating event related to an Identifier.

Enumerator

FLEXIO_LIN_IP_NO_EVENT	No event.
FLEXIO_LIN_IP_WAKEUP_SIGNAL	Wakeup signal detected.
FLEXIO_LIN_IP_BAUDRATE_ADJUSTED	Baudrate adjusted.
FLEXIO_LIN_IP_RECV_BREAK_FIELD_OK	Break field received.
FLEXIO_LIN_IP_SYNC_ERROR	Sync byte received with errors.
FLEXIO_LIN_IP_RECV_HEADER_OK	PID byte received ok.
FLEXIO_LIN_IP_PID_ERROR	PID byte received with errors.
FLEXIO_LIN_IP_TX_UNDERRUN_ERROR	Tx underrun error.
FLEXIO_LIN_IP_READBACK_ERROR	Readback error.
FLEXIO_LIN_IP_CHECKSUM_ERROR_EVENT	Checksum error.
FLEXIO_LIN_IP_TX_COMPLETED	Tx completed.
FLEXIO_LIN_IP_RX_COMPLETED	Rx completed.
FLEXIO_LIN_IP_RX_OVERRUN_ERROR	Rx overrun error.
FLEXIO_LIN_IP_TIMEOUT_ERROR	Timeout error.

Definition at line 106 of file Flexio\_Lin\_Ip\_Types.h.

#### 6.1.5.2 Flexio\_Lin\_Ip\_NodeStateType

```
enum Flexio_Lin_Ip_NodeStateType
```

Define type for an enumerating LIN Node state.

Enumerator

FLEXIO_LIN_IP_NODE_STATE_UNINIT	Uninitialized state.
FLEXIO_LIN_IP_NODE_STATE_SLEEP_MODE	Sleep mode state.
FLEXIO_LIN_IP_NODE_STATE_IDLE	Idle state.
FLEXIO_LIN_IP_NODE_STATE_SEND_BREAK_FIELD	Send break field state.
FLEXIO_LIN_IP_NODE_STATE_RECV_SYNC	Receive the synchronization byte state.
FLEXIO_LIN_IP_NODE_STATE_SEND_PID	Send PID state.
FLEXIO_LIN_IP_NODE_STATE_RECV_PID	Receive PID state.
FLEXIO_LIN_IP_NODE_STATE_RECV_DATA	Receive data state.
FLEXIO_LIN_IP_NODE_STATE_RECV_DATA_COMPLETED	Receive data completed state.
FLEXIO_LIN_IP_NODE_STATE_SEND_DATA	Send data state.
FLEXIO_LIN_IP_NODE_STATE_SEND_DATA_COMPLETED	Send data completed state.
FLEXIO_LIN_IP_NODE_STATE_SEND_SYNC	Send data completed state.

Definition at line 128 of file Flexio\_Lin\_Ip\_Types.h.

#### 6.1.5.3 Flexio\_Lin\_Ip\_StatusType

```
enum Flexio_Lin_Ip_StatusType
```

Define type function possible return values.

Definition at line 149 of file Flexio\_Lin\_Ip\_Types.h.

#### 6.1.5.4 Flexio\_Lin\_Ip\_FrameCsModelType

```
enum Flexio_Lin_Ip_FrameCsModelType
```

Define type for check sum type of the frame.

Enumerator

FLEXIO_LIN_IP_ENHANCED_CS	Enhanced CheckSum model.
FLEXIO_LIN_IP_CLASSIC_CS	Classic CheckSum model.

Definition at line 160 of file Flexio\_Lin\_Ip\_Types.h.

### 6.1.5.5 Flexio\_Lin\_Ip\_FrameResponseType

```
enum Flexio_Lin_Ip_FrameResponseType
```

Define type of the response part of frame.

Enumerator

FLEXIO_LIN_IP_FRAMERESPONSE_TX	When response is generated from master.
FLEXIO_LIN_IP_FRAMERESPONSE_RX	When response is generated from a remote slave.
FLEXIO_LIN_IP_FRAMERESPONSE_IGNORE	When response is generated from one slave to another slave.

Definition at line 171 of file Flexio\_Lin\_Ip\_Types.h.

### 6.1.5.6 Flexio\_Lin\_Ip\_TransferStatusType

```
enum Flexio_Lin_Ip_TransferStatusType
```

Description: This type defines a range of transfer status.

Enumerator

FLEXIO_LIN_IP_STATUS_FAIL	Command has not been accepted .
FLEXIO_LIN_IP_STATUS_TX_OK	Master: Header or Response successful transmission. Slave: Response successful transmission.
FLEXIO_LIN_IP_STATUS_TX_BUSY	Master: Ongoing transmission (Header or Response). Slave: Ongoing transmission response.
FLEXIO_LIN_IP_STATUS_TX_HEADER_ERROR ROR	Master: Erroneous header transmission such as: Mismatch between sent and read back data or Physical bus error.
FLEXIO_LIN_IP_STATUS_TX_ERROR	Master or Slave: Erroneous response transmission such as mismatch between sent and read back data, Physical bus error.
FLEXIO_LIN_IP_STATUS_RX_OK	Master or Slave: Reception of correct response.

## Enumerator

FLEXIO_LIN_IP_STATUS_RX_BUSY	Master or Slave: Ongoing reception. At least one response byte has been received, but the checksum byte has not been received.
FLEXIO_LIN_IP_STATUS_RX_ERROR	Master or Slave: Erroneous response reception such as: - Framing error - Overrun error - Checksum error or Short response(timeout occurred).
FLEXIO_LIN_IP_STATUS_RX_NO_RESPONSE	Master or Slave: No response byte has been received so far(timeout occurred).
FLEXIO_LIN_IP_STATUS_RX_HEADER_OK	Slave: Reception of correct header.
FLEXIO_LIN_IP_STATUS_RX_HEADER_BUSY	Slave: Ongoing header reception.
FLEXIO_LIN_IP_STATUS_RX_HEADER_ERROR	Slave: Erroneous header reception such as: Break field error, Sync field error, Identifier parity error, Framing error, timeout occurred or Physical bus error.
FLEXIO_LIN_IP_STATUS_OPERATIONAL	Normal operation; there is no data has been sent or received after channel initialized or switched to IDLE from SLEEP.
FLEXIO_LIN_IP_STATUS_SLEEP	Sleep state operation.

Definition at line 182 of file Flexio\_Lin\_Ip\_Types.h.

## 6.1.6 Function Reference

### 6.1.6.1 Flexio\_Lin\_Ip\_Init()

```
Flexio_Lin_Ip_StatusType Flexio_Lin_Ip_Init (
    const uint8 Channel,
    const Flexio_Lin_Ip_UserConfigType * UserConfig )
```

Initializes an LIN\_FLEXIO channel for LIN Network.

The caller provides memory for the driver state structures during initialization. The user must select the LIN\_FLEXIO clock source in the application to initialize the LIN\_FLEXIO. This function initializes a FLEXIO channel for operation. This function will initialize the run-time state structure to keep track of the on-going transfers, initialize the module to user defined settings and default settings. It will configure two timers, two shifters and two pins for Rx and Tx transmissions and at the end will enable the FLEXIO module.

## Parameters

<i>Channel</i>	LIN_FLEXIO channel number
<i>UserConfig</i>	user configuration structure of type <a href="#">Flexio_Lin_Ip_UserConfigType</a>

Returns

Flexio\_Lin\_Ip\_StatusType

Return values

<i>FLEXIO_LIN_IP_STATUS_SUCCESS</i>	: Init command has been accepted.
<i>FLEXIO_LIN_IP_STATUS_ERROR</i>	: Flexio module is not enabled, thus flexio channel cannot be initialized.

6.1.6.2 Flexio\_Lin\_Ip\_GoToIdleState()

```
void Flexio_Lin_Ip_GoToIdleState (
    const uint8 Channel )
```

Puts current LIN node to Idle state This function changes current node state to FLEXIO\_LIN\_IP\_NODE\_STATE\_IDLE.

Parameters

<i>channel</i>	LIN_FLEXIO channel number
----------------	---------------------------

Returns

void

6.1.6.3 Flexio\_Lin\_Ip\_AbortTransferData()

```
Flexio_Lin_Ip_StatusType Flexio_Lin_Ip_AbortTransferData (
    const uint8 Channel )
```

Aborts an on-going non-blocking transmission/reception.

While performing a non-blocking transferring data, users can call this function to terminate immediately the transferring.

Parameters

<i>channel</i>	LIN_FLEXIO channel number
----------------	---------------------------



Returns

operation status:

Return values

<i>FLEXIO_LIN_IP_STATUS_SUCCESS</i>	: The frame was aborted
<i>FLEXIO_LIN_IP_STATUS_ERROR</i>	: Operation failed due to transmission/reception could not stopped in timeout.

#### 6.1.6.4 Flexio\_Lin\_Ip\_GoToSleepMode()

```
Flexio_Lin_Ip_StatusType Flexio_Lin_Ip_GoToSleepMode (
    const uint8 Channel )
```

This function puts current node to sleep mode.

This function changes current node state to FLEXIO\_LIN\_IP\_NODE\_STATE\_SLEEP\_MODE

Parameters

<i>channel</i>	LIN_FLEXIO channel number
----------------	---------------------------

Returns

Flexio\_Lin\_Ip\_StatusType

Return values

<i>FLEXIO_LIN_IP_STATUS_SUCCESS</i>	: Gotosleep command has been accepted.
-------------------------------------	--

#### 6.1.6.5 Flexio\_Lin\_Ip\_SendWakeupSignal()

```
Flexio_Lin_Ip_StatusType Flexio_Lin_Ip_SendWakeupSignal (
    const uint8 Channel )
```

Sends a wakeup signal through the LIN\_FLEXIO interface.

Parameters

<i>channel</i>	LIN_FLEXIO channel number
----------------	---------------------------

Returns

operation status:

Return values

<i>FLEXIO_LIN_IP_STATUS_SUCCESS</i>	: Command has been accepted.
<i>FLEXIO_LIN_IP_STATUS_ERROR</i>	: Command has not been accepted, error occurred.

### 6.1.6.6 Flexio\_Lin\_Ip\_Deinit()

```
Flexio_Lin_Ip_StatusType Flexio_Lin_Ip_Deinit (  
    const uint8 Channel )
```

De-initialize a FLEXIO LIN channel.

This function shall de initialize a flexio lin channel. If no channel is available, disables also the FLEXIO module.

Parameters

<i>channel</i>	LIN_FLEXIO instance number.
----------------	-----------------------------

Returns

operation status: VOID

### 6.1.6.7 Flexio\_Lin\_Ip\_GetStatus()

```
Flexio_Lin_Ip_TransferStatusType Flexio_Lin_Ip_GetStatus (  
    const uint8 Channel,  
    const uint8 ** Buffer )
```

Returns the status of an on going transmission.

This function returns the status of the current non-blocking transfer. If a response reception has been successfully received, Buffer will be referenced to receive buffer.

Parameters

<i>channel</i>	LIN_FLEXIO instance number.
<i>Buffer</i>	Pointer to the received data buffer.

Returns

operation status: Flexio\_Lin\_Ip\_TransferStatusType

Return values

<i>FLEXIO_LIN_IP_STATUS_FAIL</i>	Command has not been accepted .
<i>FLEXIO_LIN_IP_STATUS_TX_OK,Master</i>	Header or Response successful transmission. Slave: Response successful transmission.
<i>FLEXIO_LIN_IP_STATUS_TX_BUSY,Master</i>	Ongoing transmission (Header or Response). Slave: Ongoing transmission response.
<i>FLEXIO_LIN_IP_STATUS_TX_HEADER_ERR↔OR,Master</i>	Erroneous header transmission such as: Mismatch between sent and read back data or Physical bus error./
<i>FLEXIO_LIN_IP_STATUS_TX_ERROR,Master</i>	or Slave: Erroneous response transmission such as mismatch between sent and read back data, Physical bus error
<i>FLEXIO_LIN_IP_STATUS_RX_OK,Master</i>	or Slave: Reception of correct response.
<i>FLEXIO_LIN_IP_STATUS_RX_BUSY,Master</i>	or Slave: Ongoing reception. At least one response byte has been received, but the checksum byte has not been received.
<i>FLEXIO_LIN_IP_STATUS_RX_ERROR,Master</i>	or Slave: Erroneous response reception such as: - Framing error - Overrun error - Checksum error or Short response(timeout occurred).
<i>FLEXIO_LIN_IP_STATUS_RX_NO_RESPON↔SE,Master</i>	or Slave: No response byte has been received so far(timeout occurred).
<i>FLEXIO_LIN_IP_STATUS_RX_HEADER_↔OK,Slave</i>	Reception of correct header.
<i>FLEXIO_LIN_IP_STATUS_RX_HEADER_BU↔SY,Slave</i>	Ongoing header reception.
<i>FLEXIO_LIN_IP_STATUS_RX_HEADER_ERR↔OR,Slave</i>	Erroneous header reception such as: Break field error, Sync field error, Identifier parity error, Framing error, timeout occurred or Physical bus error.
<i>FLEXIO_LIN_IP_STATUS_OPERATION↔AL,Normal</i>	operation; there is no data has been sent or received after channel initialized or switched to IDLE from SLEEP.
<i>FLEXIO_LIN_IP_STATUS_SLEEP</i>	Sleep state operation.

#### 6.1.6.8 Flexio\_Lin\_Ip\_SendFrame()

```
Flexio_Lin_Ip_StatusType Flexio_Lin_Ip_SendFrame (
    const uint8 Channel,
    const Flexio_Lin_Ip_PduType * PduInfo )
```

Sends a LIN frame as master or a response as a slave.

## Module Documentation

### Parameters

<i>channel</i>	LIN_FLEXIO channel number
<i>PduInfo</i>	Structure which contains information about the current frame which is to be transfered such as PID, Data Length, Type of checksum, Data buffer and Type of the response expected.

### Returns

operation status:

### Return values

<i>FLEXIO_LIN_IP_STATUS_SUCCESS</i>	: Command has not been accepted .
<i>FLEXIO_LIN_IP_STATUS_BUSY</i>	: Driver is busy with another transfer.
<i>FLEXIO_LIN_IP_STATUS_ERROR</i>	: Parameters such as Data Length and Pid are not correct or the current node is in sleep mode.

#### 6.1.6.9 Flexio\_Lin\_Ip\_TimerExpiredService()

```
void Flexio_Lin_Ip_TimerExpiredService (  
    const uint8 Channel )
```

This is callback function for Timer Interrupt Handler. Users shall initialize a timer (for example FTM) and the time period in microseconds will be set by the driver via LinFlexioStartTimerNotification. In timer IRQ handler, call this function.

## 6.2 LIN Driver

### 6.2.1 Detailed Description

#### Data Structures

- struct [Lin\\_43\\_LPUART\\_FLEXIO\\_ChannelConfigType](#)  
*LIN channel configuration type structure. [More...](#)*
- struct [Lin\\_43\\_LPUART\\_FLEXIO\\_ConfigType](#)  
*LIN driver configuration type structure. [More...](#)*

#### Macros

- `#define LIN_43_LPUART_FLEXIO_E_UNINIT`  
*API service used without module initialization.*
- `#define LIN_43_LPUART_FLEXIO_E_INVALID_CHANNEL`  
*API service used with an invalid or inactive channel parameter.*
- `#define LIN_43_LPUART_FLEXIO_E_INIT_FAILED`  
*API service called with invalid configuration pointer.*
- `#define LIN_43_LPUART_FLEXIO_E_STATE_TRANSITION`  
*Invalid state transition for the current state.*
- `#define LIN_43_LPUART_FLEXIO_E_PARAM_POINTER`  
*API service called with a NULL pointer.*
- `#define LIN_43_LPUART_FLEXIO_E_TIMEOUT`  
*Timeout caused by hardware error.*
- `#define LIN_43_LPUART_FLEXIO_E_PARAM_CONFIG`  
*API service called with the requested resource is not configured to be available on the current core.*
- `#define LIN_43_LPUART_FLEXIO_UNINIT`  
*LIN driver states.*
- `#define LIN_43_LPUART_FLEXIO_INIT`  
*LIN driver states.*
- `#define LIN_43_LPUART_FLEXIO_CH_SLEEP_PENDING`  
*LIN Channel states.*
- `#define LIN_43_LPUART_FLEXIO_CH_SLEEP_STATE`  
*LIN Channel states.*
- `#define LIN_43_LPUART_FLEXIO_CH_OPERATIONAL`  
*LIN Channel states.*
- `#define LIN_43_LPUART_FLEXIO_GETSTATUS_ID`  
*API service ID for [Lin\\_43\\_LPUART\\_FLEXIO\\_GetStatus\(\)](#) function.*
- `#define LIN_43_LPUART_FLEXIO_GETVERSIONINFO_ID`  
*API service ID for [Lin\\_43\\_LPUART\\_FLEXIO\\_GetVersionInfo\(\)](#) function.*
- `#define LIN_43_LPUART_FLEXIO_GOTOSLEEP_ID`  
*API service ID for [Lin\\_43\\_LPUART\\_FLEXIO\\_GoToSleep\(\)](#) function.*
- `#define LIN_43_LPUART_FLEXIO_GOTOSLEEPINTERNAL_ID`

- *API service ID for [Lin\\_43\\_LPUART\\_FLEXIO\\_GoToSleepInternal\(\)](#) function.*
- `#define LIN_43_LPUART_FLEXIO_INIT_ID`
- *API service ID for [Lin\\_43\\_LPUART\\_FLEXIO\\_Init\(\)](#) function.*
- `#define LIN_43_LPUART_FLEXIO_SENDFRAME_ID`
- *API service ID for [Lin\\_43\\_LPUART\\_FLEXIO\\_SendFrame\(\)](#) function.*
- `#define LIN_43_LPUART_FLEXIO_WAKEUP_ID`
- *API service ID for [Lin\\_43\\_LPUART\\_FLEXIO\\_WakeUp\(\)](#) function.*
- `#define LIN_43_LPUART_FLEXIO_WAKEUPINTERNAL_ID`
- *API service ID for [Lin\\_43\\_LPUART\\_FLEXIO\\_WakeupInternal\(\)](#) function.*
- `#define LIN_43_LPUART_FLEXIO_CHECKWAKEUP_ID`
- *API service ID for [Lin\\_43\\_LPUART\\_FLEXIO\\_CheckWakeup\(\)](#) function.*
- `#define LIN_43_LPUART_FLEXIO_TIMEOUT_ERROR`
- *Return code for timeout error.*

## Function Reference

- `Std_ReturnType Lin\_43\_LPUART\_FLEXIO\_CheckWakeup (uint8 Channel)`  
*Validates for upper layers the wake up of LIN channel.*
- `void Lin\_43\_LPUART\_FLEXIO\_Init (const Lin\_43\_LPUART\_FLEXIO\_ConfigType *Config)`  
*Initializes the LIN module.*
- `Lin_StatusType Lin\_43\_LPUART\_FLEXIO\_GetStatus (uint8 Channel, const uint8 **Lin_43_LPUART_FLEXIO_SduPtr)`  
*Gets the status of the LIN driver when Channel is operating.*
- `Std_ReturnType Lin\_43\_LPUART\_FLEXIO\_SendFrame (uint8 Channel, const Lin_PduType *PduInfoPtr)`  
*Sends a LIN frame.*
- `Std_ReturnType Lin\_43\_LPUART\_FLEXIO\_GoToSleep (uint8 Channel)`  
*Prepares and send a go-to-sleep-command frame on Channel.*
- `Std_ReturnType Lin\_43\_LPUART\_FLEXIO\_GoToSleepInternal (uint8 Channel)`  
*Same function as [Lin\\_43\\_LPUART\\_FLEXIO\\_Ipw\\_GoToSleep\(\)](#) but without sending a go-to-sleep-command on the bus.*
- `Std_ReturnType Lin\_43\_LPUART\_FLEXIO\_Wakeup (uint8 Channel)`  
*Generates a wake up pulse.*
- `Std_ReturnType Lin\_43\_LPUART\_FLEXIO\_WakeupInternal (uint8 Channel)`  
*Wake up the LIN channel.*
- `void Lin\_43\_LPUART\_FLEXIO\_GetVersionInfo (Std_VersionInfoType *versioninfo)`  
*Returns the version information of this module.*

## 6.2.2 Data Structure Documentation

### 6.2.2.1 struct [Lin\\_43\\_LPUART\\_FLEXIO\\_ChannelConfigType](#)

LIN channel configuration type structure.

This is the type of the external data structure containing the overall initialization data for one LIN Channel. A pointer to such a structure is provided to the LIN channel initialization routine for configuration of the LIN hardware channel.

Definition at line 144 of file [Lin\\_43\\_LPUART\\_FLEXIO\\_Types.h](#).

## Data Fields

	Type	Name	Description
	uint8	LinChannelID	
Lin_43_LPUART_FLEXIO_HwConfigType *	const	ChannelConfigPtr	Lin Channel ID. !<
	uint32	ChannelCoreId	LIN Hardware configuration pointer. !<
	boolean	AllocatedPartition	LIN Channel core id. !<

### 6.2.2.2 struct Lin\_43\_LPUART\_FLEXIO\_ConfigType

LIN driver configuration type structure.

This is the type of the pointer to the external data LIN Channels. A pointer of such a structure is provided to the LIN driver initialization routine for configuration of the LIN hardware channel.

Definition at line 164 of file Lin\_43\_LPUART\_FLEXIO\_Types.h.

## Data Fields

- const [Lin\\_43\\_LPUART\\_FLEXIO\\_ChannelConfigType](#) \* [Lin\\_43\\_LPUART\\_FLEXIO\\_ChannelPtr](#) [(2U)]  
*Partition core id is assigned for this configuration.*

#### 6.2.2.2.1 Field Documentation

**6.2.2.2.1.1 Lin\_43\_LPUART\_FLEXIO\_ChannelPtr** const [Lin\\_43\\_LPUART\\_FLEXIO\\_ChannelConfigType](#)\*  
[Lin\\_43\\_LPUART\\_FLEXIO\\_ChannelPtr](#) [(2U)]

Partition core id is assigned for this configuration.

!<

Hardware channel.

Constant pointer of the constant external data structure containing the overall initialization data for all the configured LIN Channels.

Definition at line 173 of file Lin\_43\_LPUART\_FLEXIO\_Types.h.

## 6.2.3 Macro Definition Documentation

### 6.2.3.1 LIN\_43\_LPUART\_FLEXIO\_E\_UNINIT

```
#define LIN_43_LPUART_FLEXIO_E_UNINIT
```

API service used without module initialization.

The LIN Driver module shall report the development error "LIN\_43\_LPUART\_FLEXIO\_E\_UNINIT (0x00)", when the API Service is used without module initialization.

Definition at line 148 of file Lin\_43\_LPUART\_FLEXIO.h.

### 6.2.3.2 LIN\_43\_LPUART\_FLEXIO\_E\_INVALID\_CHANNEL

```
#define LIN_43_LPUART_FLEXIO_E_INVALID_CHANNEL
```

API service used with an invalid or inactive channel parameter.

The LIN Driver module shall report the development error "LIN\_43\_LPUART\_FLEXIO\_E\_INVALID\_CHANNEL (0x02)", when API Service used with an invalid or inactive channel parameter.

Definition at line 158 of file Lin\_43\_LPUART\_FLEXIO.h.

### 6.2.3.3 LIN\_43\_LPUART\_FLEXIO\_E\_INIT\_FAILED

```
#define LIN_43_LPUART_FLEXIO_E_INIT_FAILED
```

API service called with invalid configuration pointer.

The LIN Driver module shall report the development error "LIN\_43\_LPUART\_FLEXIO\_E\_INIT\_FAILED (0x03)", when API Service is called with invalid configuration pointer.

Definition at line 168 of file Lin\_43\_LPUART\_FLEXIO.h.

### 6.2.3.4 LIN\_43\_LPUART\_FLEXIO\_E\_STATE\_TRANSITION

```
#define LIN_43_LPUART_FLEXIO_E_STATE_TRANSITION
```

Invalid state transition for the current state.

The LIN Driver module shall report the development error "LIN\_43\_LPUART\_FLEXIO\_E\_STATE\_TRANSITION (0x04)", when Invalid state transition occurs from the current state.

Definition at line 178 of file Lin\_43\_LPUART\_FLEXIO.h.



### 6.2.3.5 LIN\_43\_LPUART\_FLEXIO\_E\_PARAM\_POINTER

```
#define LIN_43_LPUART_FLEXIO_E_PARAM_POINTER
```

API service called with a NULL pointer.

The LIN Driver module shall report the development error "LIN\_43\_LPUART\_FLEXIO\_E\_PARAM\_POINTER (0x05)", when API Service is called with a NULL pointer. In case of this error, the API service shall return immediately without any further action, beside reporting this development error.

Definition at line 190 of file Lin\_43\_LPUART\_FLEXIO.h.

### 6.2.3.6 LIN\_43\_LPUART\_FLEXIO\_E\_TIMEOUT

```
#define LIN_43_LPUART_FLEXIO_E_TIMEOUT
```

Timeout caused by hardware error.

The LIN Driver module shall report the development error "LIN\_43\_LPUART\_FLEXIO\_E\_TIMEOUT (0x06)", when the error "Timeout caused by hardware error" has occurred and the reference LinDemEventParameterRefs/LIN\_E\_TIMEOUT is not configured.

Definition at line 201 of file Lin\_43\_LPUART\_FLEXIO.h.

### 6.2.3.7 LIN\_43\_LPUART\_FLEXIO\_E\_PARAM\_CONFIG

```
#define LIN_43_LPUART_FLEXIO_E_PARAM_CONFIG
```

API service called with the requested resource is not configured to be available on the current core.

The LIN will check upon each API call if the requested resource is configured to be available on the current core, and in case of error will return LIN\_43\_LPUART\_FLEXIO\_E\_PARAM\_CONFIG

Definition at line 210 of file Lin\_43\_LPUART\_FLEXIO.h.

### 6.2.3.8 LIN\_43\_LPUART\_FLEXIO\_UNINIT

```
#define LIN_43_LPUART_FLEXIO_UNINIT
```

LIN driver states.

The state LIN\_43\_LPUART\_FLEXIO\_UNINIT means that the Lin module has not been initialized yet and cannot be used.

Definition at line 221 of file Lin\_43\_LPUART\_FLEXIO.h.

### 6.2.3.9 LIN\_43\_LPUART\_FLEXIO\_INIT

```
#define LIN_43_LPUART_FLEXIO_INIT
```

LIN driver states.

The LIN\_43\_LPUART\_FLEXIO\_INIT state indicates that the LIN driver has been initialized, making each available channel ready for service.

Definition at line 230 of file Lin\_43\_LPUART\_FLEXIO.h.

### 6.2.3.10 LIN\_43\_LPUART\_FLEXIO\_CH\_SLEEP\_PENDING

```
#define LIN_43_LPUART_FLEXIO_CH_SLEEP_PENDING
```

LIN Channel states.

go-to-sleep-command has been issued on the bus, LIN channel stay at this state until [Lin\\_43\\_LPUART\\_FLEXIO\\_GetStatus\(\)](#) is called

Definition at line 239 of file Lin\_43\_LPUART\_FLEXIO.h.

### 6.2.3.11 LIN\_43\_LPUART\_FLEXIO\_CH\_SLEEP\_STATE

```
#define LIN_43_LPUART_FLEXIO_CH_SLEEP_STATE
```

LIN Channel states.

The detection of a wake-up pulse is enabled. The LIN hardware is into a low power mode if such a mode is provided by the hardware.

Definition at line 249 of file Lin\_43\_LPUART\_FLEXIO.h.

### 6.2.3.12 LIN\_43\_LPUART\_FLEXIO\_CH\_OPERATIONAL

```
#define LIN_43_LPUART_FLEXIO_CH_OPERATIONAL
```

LIN Channel states.

The individual channel has been initialized (using at least one statically configured data set) and is able to participate in the LIN cluster.

Definition at line 259 of file Lin\_43\_LPUART\_FLEXIO.h.

### 6.2.3.13 LIN\_43\_LPUART\_FLEXIO\_GETSTATUS\_ID

```
#define LIN_43_LPUART_FLEXIO_GETSTATUS_ID
```

API service ID for [Lin\\_43\\_LPUART\\_FLEXIO\\_GetStatus\(\)](#) function.

Parameters used when raising an error or exception.

Definition at line 269 of file Lin\_43\_LPUART\_FLEXIO.h.

### 6.2.3.14 LIN\_43\_LPUART\_FLEXIO\_GETVERSIONINFO\_ID

```
#define LIN_43_LPUART_FLEXIO_GETVERSIONINFO_ID
```

API service ID for [Lin\\_43\\_LPUART\\_FLEXIO\\_GetVersionInfo\(\)](#) function.

Parameters used when raising an error or exception.

Definition at line 277 of file Lin\_43\_LPUART\_FLEXIO.h.

### 6.2.3.15 LIN\_43\_LPUART\_FLEXIO\_GOTOSLEEP\_ID

```
#define LIN_43_LPUART_FLEXIO_GOTOSLEEP_ID
```

API service ID for [Lin\\_43\\_LPUART\\_FLEXIO\\_GoToSleep\(\)](#) function.

Parameters used when raising an error or exception.

Definition at line 285 of file Lin\_43\_LPUART\_FLEXIO.h.

### 6.2.3.16 LIN\_43\_LPUART\_FLEXIO\_GOTOSLEEPINTERNAL\_ID

```
#define LIN_43_LPUART_FLEXIO_GOTOSLEEPINTERNAL_ID
```

API service ID for [Lin\\_43\\_LPUART\\_FLEXIO\\_GoToSleepInternal\(\)](#) function.

Parameters used when raising an error or exception.

Definition at line 293 of file Lin\_43\_LPUART\_FLEXIO.h.

### 6.2.3.17 LIN\_43\_LPUART\_FLEXIO\_INIT\_ID

```
#define LIN_43_LPUART_FLEXIO_INIT_ID
```

API service ID for [Lin\\_43\\_LPUART\\_FLEXIO\\_Init\(\)](#) function.

Parameters used when raising an error or exception.

Definition at line 301 of file Lin\_43\_LPUART\_FLEXIO.h.

### 6.2.3.18 LIN\_43\_LPUART\_FLEXIO\_SENDFRAME\_ID

```
#define LIN_43_LPUART_FLEXIO_SENDFRAME_ID
```

API service ID for [Lin\\_43\\_LPUART\\_FLEXIO\\_SendFrame\(\)](#) function.

Parameters used when raising an error or exception.

Definition at line 309 of file Lin\_43\_LPUART\_FLEXIO.h.

### 6.2.3.19 LIN\_43\_LPUART\_FLEXIO\_WAKEUP\_ID

```
#define LIN_43_LPUART_FLEXIO_WAKEUP_ID
```

API service ID for [Lin\\_43\\_LPUART\\_FLEXIO\\_WakeUp\(\)](#) function.

Parameters used when raising an error or exception.

Definition at line 317 of file Lin\_43\_LPUART\_FLEXIO.h.

### 6.2.3.20 LIN\_43\_LPUART\_FLEXIO\_WAKEUPINTERNAL\_ID

```
#define LIN_43_LPUART_FLEXIO_WAKEUPINTERNAL_ID
```

API service ID for [Lin\\_43\\_LPUART\\_FLEXIO\\_WakeupInternal\(\)](#) function.

Parameters used when raising an error or exception.

Definition at line 325 of file Lin\_43\_LPUART\_FLEXIO.h.

6.2.3.21 LIN\_43\_LPUART\_FLEXIO\_CHECKWAKEUP\_ID

#define LIN\_43\_LPUART\_FLEXIO\_CHECKWAKEUP\_ID

API service ID for [Lin\\_43\\_LPUART\\_FLEXIO\\_CheckWakeup\(\)](#) function.

Parameters used when raising an error or exception.

Definition at line 333 of file Lin\_43\_LPUART\_FLEXIO.h.

6.2.3.22 LIN\_43\_LPUART\_FLEXIO\_TIMEOUT\_ERROR

#define LIN\_43\_LPUART\_FLEXIO\_TIMEOUT\_ERROR

Return code for timeout error.

Definition at line 120 of file Lin\_43\_LPUART\_FLEXIO\_Types.h.

6.2.4 Function Reference

6.2.4.1 Lin\_43\_LPUART\_FLEXIO\_CheckWakeup()

```
Std_ReturnType Lin_43_LPUART_FLEXIO_CheckWakeup (
    uint8 Channel )
```

Validates for upper layers the wake up of LIN channel.

This function identifies if the addressed LIN channel has been woken up by the LIN bus transceiver. It checks the wake up flag from the addressed LIN channel which must be in sleep mode and have the wake up signal.

Parameters

in	<i>Channel</i>	LIN channel to be addressed.
----	----------------	------------------------------

Return values

<i>E_NOT_OK</i>	If the LIN Channel is not valid or LIN driver is not initialized or the addressed LIN Channel is not in sleep state.
<i>E_OK</i>	Otherwise.

6.2.4.2 Lin\_43\_LPUART\_FLEXIO\_Init()

```
void Lin_43_LPUART_FLEXIO_Init (
    const Lin_43_LPUART_FLEXIO_ConfigType * Config )
```

Initializes the LIN module.

This function performs software initialization of LIN driver:

- Clears the shadow buffer of all available Lin channels
- Sets all the available channels to sleep mode and configures their state machine to LIN\_43\_LPUART\_FLEXIO\_CH\_SLEEP\_STATE.
- Set driver state machine to LIN\_43\_LPUART\_FLEXIO\_INIT.

Parameters

in	<i>Config</i>	- Pointer to LIN driver configuration set.
----	---------------	--

Returns

void

Precondition

-

6.2.4.3 Lin\_43\_LPUART\_FLEXIO\_GetStatus()

```
Lin_StatusType Lin_43_LPUART_FLEXIO_GetStatus (
    uint8 Channel,
    const uint8 ** Lin_43_LPUART_FLEXIO_SduPtr )
```

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

This function returns the state of the current transmission, reception or operation status. If the reception of a Slave response was successful then this service provides a pointer to the buffer where the data is stored.

Parameters

in	<i>Channel</i>	LIN channel to be addressed
out	<i>LinSduPtr</i>	pointer to pointer to a shadow buffer or memory mapped LIN Hardware receive buffer where the current SDU is stored

Returns

Lin\_StatusType.

Return values

<i>LIN_NOT_OK</i>	Development or production error rised none of the below conditions.
<i>LIN_TX_OK</i>	Successful transmission.
<i>LIN_TX_BUSY</i>	Ongoing transmission of header or response.
<i>LIN_TX_HEADER_ERROR</i>	Error occurred during header transmission.
<i>LIN_TX_ERROR</i>	Error occurred during response transmission.
<i>LIN_RX_OK</i>	Reception of correct response.
<i>LIN_RX_BUSY</i>	Ongoing reception where at least one byte has been received.
<i>LIN_RX_ERROR</i>	Error occurred during reception.
<i>LIN_RX_NO_REPONSE</i>	No data byte has been received yet.
<i>LIN_OPERATIONAL</i>	Channel is ready for next header. transmission and no data are available.

Precondition

: This API is available only if at least one node is configured as MASTER.

#### 6.2.4.4 Lin\_43\_LPUART\_FLEXIO\_SendFrame()

```
Std_ReturnType Lin_43_LPUART_FLEXIO_SendFrame (
    uint8 Channel,
    const Lin_PduType * PduInfoPtr )
```

Sends a LIN frame.

Sends a LIN header and a LIN response, if necessary. The direction of the frame response (master response, slave response, slave-to-slave communication) is provided by the PduInfoPtr.

Parameters

in	<i>Channel</i>	LIN channel to be addressed.
in	<i>PduInfoPtr</i>	pointer to PDU containing the PID, Checksum model, Response type, Dl and SDU data pointer.

Returns

Std\_ReturnType.

### Return values

<i>E_NOT_OK</i>	If the LIN Channel is not valid or LIN driver is not initialized or PduInfoPtr is NULL or a timeout occurs or LIN Channel is in sleep state.
<i>E_OK</i>	Otherwise.

### Precondition

: Lin\_43\_LPUART\_FLEXIO\_Init function must be called before this API. This API is available only if at least one node is configured as MASTER.

#### 6.2.4.5 Lin\_43\_LPUART\_FLEXIO\_GoToSleep()

```
Std_ReturnType Lin_43_LPUART_FLEXIO_GoToSleep (  
    uint8 Channel )
```

Prepares and send a go-to-sleep-command frame on Channel.

This function stops any ongoing transmission and initiates the transmission of the sleep command (master command frame with ID = 0x3C and data = (0x00, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF)).

### Parameters

in	<i>Channel</i>	LIN channel to be addressed.
----	----------------	------------------------------

### Returns

Std\_ReturnType.

### Return values

<i>E_NOT_OK</i>	In case of a timeout situation only.
<i>E_OK</i>	Otherwise.

### Precondition

: Lin\_43\_LPUART\_FLEXIO\_Init function must be called before this API. This API is available only if at least one node is configured as MASTER.



#### 6.2.4.6 Lin\_43\_LPUART\_FLEXIO\_GoToSleepInternal()

```
Std_ReturnType Lin_43_LPUART_FLEXIO_GoToSleepInternal (
    uint8 Channel )
```

Same function as Lin\_43\_LPUART\_FLEXIO\_Ipw\_GoToSleep() but without sending a go-to-sleep-command on the bus.

This function stops any ongoing transmission and put the Channel in sleep mode (then LIN hardware enters a reduced power operation mode).

Parameters

in	<i>Channel</i>	LIN channel to be addressed.
----	----------------	------------------------------

Returns

Std\_ReturnType.

Return values

<i>E_NOT_OK</i>	In case of a timeout situation only.
<i>E_OK</i>	Otherwise.

Precondition

: Lin\_43\_LPUART\_FLEXIO\_Init function must be called before this API.

#### 6.2.4.7 Lin\_43\_LPUART\_FLEXIO\_Wakeup()

```
Std_ReturnType Lin_43_LPUART_FLEXIO_Wakeup (
    uint8 Channel )
```

Generates a wake up pulse.

This function shall sent a wake up signal to the LIN bus and put the LIN channel in LIN\_43\_LPUART\_FLEXIO\_O\_CH\_OPERATIONAL state.

Parameters

in	<i>Channel</i>	LIN channel to be addressed.
----	----------------	------------------------------

Returns

Std\_ReturnType.

Return values

<i>E_NOT_OK</i>	If the LIN driver is not in sleep state or LIN Channel is not valid or LIN driver is not initialized.
<i>E_OK</i>	Otherwise.

Precondition

: Lin\_43\_LPUART\_FLEXIO\_Init function must be called before this API.

### 6.2.4.8 Lin\_43\_LPUART\_FLEXIO\_WakeupInternal()

```
Std_ReturnType Lin_43_LPUART_FLEXIO_WakeupInternal (
    uint8 Channel )
```

Wake up the LIN channel.

This function shall put the LIN channel in LIN\_43\_LPUART\_FLEXIO\_CH\_OPERATIONAL state without sending a wake up signal to the LIN bus

Parameters

in	<i>Channel</i>	LIN channel to be addressed.
----	----------------	------------------------------

Returns

Std\_ReturnType.

Return values

<i>E_NOT_OK</i>	If the LIN driver is not in sleep state or LIN Channel is not valid or LIN driver is not initialized.
<i>E_OK</i>	Otherwise.

Precondition

: Lin\_43\_LPUART\_FLEXIO\_Init function must be called before this API.

#### 6.2.4.9 Lin\_43\_LPUART\_FLEXIO\_GetVersionInfo()

```
void Lin_43_LPUART_FLEXIO_GetVersionInfo (
    Std_VersionInfoType * versioninfo )
```

Returns the version information of this module.

The version information includes:

- Two bytes for the Vendor ID
- Two bytes for the Module ID
- One byte for the Instance ID
- Three bytes version number. The numbering shall be vendor specific: it consists of:
  - The major, the minor and the patch version number of the module;
  - The AUTOSAR specification version number shall not be included. The AUTOSAR specification version number is checked during compile time and therefore not required in this API.

Parameters

in, out	<i>versioninfo</i>	Pointer for storing the version information of this module.
---------	--------------------	---

Returns

void.

## 6.3 LIN

### 6.3.1 Detailed Description

#### Macros

- `#define LIN_43_LPUART_FLEXIO_SETCLOCKMODE_ID`  
*API service ID for `Lin_43_LPUART_FLEXIO_SetClockMode()` function.*

#### Enum Reference

- enum `Lin_43_LPUART_FLEXIO_ClockModesType`  
*Clock modes.*

#### Function Reference

- Std\_ReturnType `Lin_43_LPUART_FLEXIO_SetClockMode` (`Lin_43_LPUART_FLEXIO_ClockModesType` ClockMode)  
*Lin\_43\_LPUART\_FLEXIO\_SetClockMode.*

### 6.3.2 Macro Definition Documentation

#### 6.3.2.1 LIN\_43\_LPUART\_FLEXIO\_SETCLOCKMODE\_ID

```
#define LIN_43_LPUART_FLEXIO_SETCLOCKMODE_ID
```

API service ID for `Lin_43_LPUART_FLEXIO_SetClockMode()` function.

Parameters used when raising an error or exception.

Definition at line 95 of file `Lin_43_LPUART_FLEXIO_ASRExt.h`.

### 6.3.3 Enum Reference

#### 6.3.3.1 Lin\_43\_LPUART\_FLEXIO\_ClockModesType

```
enum Lin_43_LPUART_FLEXIO_ClockModesType
```

Clock modes.

Precondition

`LIN_43_LPUART_FLEXIO_DUAL_CLOCK_MODE` must be defined and its value must be `STD_ON`.

Note

Possible clock modes: `LIN_43_LPUART_FLEXIO_NORMAL` (normal execution mode), `LIN_43_LPUART_FLEXIO_ALTERNATE` (low power mode).

Enumerator

<code>LIN_43_LPUART_FLEXIO_NORMAL</code>	<code>LIN_43_LPUART_FLEXIO_NORMAL</code> mode.
<code>LIN_43_LPUART_FLEXIO_ALTERNATE</code>	<code>LIN_43_LPUART_FLEXIO_ALTERNATE</code> mode.

Definition at line 111 of file `Lin_43_LPUART_FLEXIO_ASRExt.h`.

## 6.3.4 Function Reference

### 6.3.4.1 `Lin_43_LPUART_FLEXIO_SetClockMode()`

```
Std_ReturnType Lin_43_LPUART_FLEXIO_SetClockMode (
    Lin_43_LPUART_FLEXIO_ClockModesType ClockMode )
```

`Lin_43_LPUART_FLEXIO_SetClockMode`.

Switch the clock mode to the alternate clock mode on all the Lin channels.

Parameters

in	<i>ClockMode</i>	New clock mode.
----	------------------	-----------------

Returns

The result of the switching clock operation.

Return values

<i>E_OK</i>	The switching operation was OK.
<i>E_NOT_OK</i>	The switching operation has failed caused by the wrong driver state.

Switch the clock mode to the alternate clock mode on all the Lin channels.

Precondition

`LIN_43_LPUART_FLEXIO_DUAL_CLOCK_MODE` must be defined and its value must be `STD_ON`.

Note

Switch the clock mode to the alternate clock mode on all the Lin channels.

## 6.4 Lpuart Lin IPL

### 6.4.1 Detailed Description

#### Data Structures

- struct [Lpuart\\_Lin\\_Ip\\_PduType](#)  
*Define type for an structure containing information regarding the LIN Frame . [More...](#)*
- struct [Lpuart\\_Lin\\_Ip\\_StateStructType](#)  
*Runtime state of the LIN driver. [More...](#)*
- struct [Lpuart\\_Lin\\_Ip\\_UserConfigType](#)  
*User configuration structure of the LIN driver. [More...](#)*

#### Macros

- `#define LPUART_LIN_IP_SLAVE`  
*Macro that specifies usage of a SLAVE node.*
- `#define LPUART_LIN_IP_MASTER`  
*Macro that specifies usage of a SLAVE node.*
- `#define LPUART_LIN_IP_BREAK_CHAR_10_BIT_MINIMUM_U8`  
*Macro LPUART break char length 10 bit times.*
- `#define LPUART_LIN_IP_BREAK_CHAR_13_BIT_MINIMUM_U8`  
*Macro LPUART break char length 10 bit times.*

#### Types Reference

- typedef void(\* [Lpuart\\_Lin\\_Ip\\_CallbackType](#)) (const uint8 Instance, const [Lpuart\\_Lin\\_Ip\\_StateStructType](#) \*LinState)  
*LIN Driver callback function type.*

#### Enum Reference

- enum [Lpuart\\_Lin\\_Ip\\_EventIdType](#)  
*Enum containing the events related to a ID.*
- enum [Lpuart\\_Lin\\_Ip\\_NodeStateType](#)  
*Define type for an enumerating LIN Node state.*
- enum [Lpuart\\_Lin\\_Ip\\_StatusType](#)  
*LPUART status type.*
- enum [Lpuart\\_Lin\\_Ip\\_FrameCsModelType](#)  
*Define type for checksum type of the frame.*
- enum [Lpuart\\_Lin\\_Ip\\_FrameResponseType](#)  
*Define type for Response type of the frame.*
- enum [Lpuart\\_Lin\\_Ip\\_TransferStatusType](#)  
*Description: This type defines a range of transfer status.*

## Function Reference

- [Lpuart\\_Lin\\_Ip\\_StatusType Lpuart\\_Lin\\_Ip\\_Init](#) (const uint8 Instance, const [Lpuart\\_Lin\\_Ip\\_UserConfigType](#) \*UserConfig)  
*Initializes an LIN\_LPUART instance for LIN Network.*
- [Lpuart\\_Lin\\_Ip\\_StatusType Lpuart\\_Lin\\_Ip\\_Deinit](#) (const uint8 Instance)  
*Shuts down the LIN\_LPUART by disabling interrupts and transmitter/receiver.*
- [Lpuart\\_Lin\\_Ip\\_StatusType Lpuart\\_Lin\\_Ip\\_SendFrame](#) (const uint8 Instance, const [Lpuart\\_Lin\\_Ip\\_PduType](#) \*PduInfo)  
*Sends frame out through the LIN\_LPUART module using non-blocking method.*
- [Lpuart\\_Lin\\_Ip\\_StatusType Lpuart\\_Lin\\_Ip\\_AbortTransferData](#) (const uint8 Instance)  
*Aborts an on-going non-blocking transmission/reception.*
- [Lpuart\\_Lin\\_Ip\\_StatusType Lpuart\\_Lin\\_Ip\\_GoToSleepMode](#) (const uint8 Instance)  
*This function puts current node to sleep mode.*
- void [Lpuart\\_Lin\\_Ip\\_GoToIdleState](#) (const uint8 Instance)  
*Puts current LIN node to Idle state This function changes current node state to LIN\_NODE\_STATE\_IDLE.*
- [Lpuart\\_Lin\\_Ip\\_StatusType Lpuart\\_Lin\\_Ip\\_SendWakeupSignal](#) (const uint8 Instance)  
*Sends a wakeup signal through the LIN\_LPUART interface.*
- [Lpuart\\_Lin\\_Ip\\_TransferStatusType Lpuart\\_Lin\\_Ip\\_GetStatus](#) (const uint8 Instance, const uint8 \*\*Buffer)  
*Returns the status of an on going transmission.*
- void [Lpuart\\_Lin\\_Ip\\_IRQHandler](#) (const uint8 Instance)  
*This is callback function for Timer Interrupt Handler. Users shall initialize a timer (for example FTM) and the time period in microseconds will be set by the driver via LinLpuartStartTimerNotification. In timer IRQ handler, call this function.*

## 6.4.2 Data Structure Documentation

### 6.4.2.1 struct Lpuart\_Lin\_Ip\_PduType

Define type for an structure containing information regarding the LIN Frame .

Definition at line 223 of file Lpuart\_Lin\_Ip\_Types.h.

#### Data Fields

- uint8 [Pid](#)  
*LIN frame identifier.*
- [Lpuart\\_Lin\\_Ip\\_FrameCsModelType](#) Cs  
*Checksum model type.*
- [Lpuart\\_Lin\\_Ip\\_FrameResponseType](#) Drc  
*Response type.*
- uint8 [Dl](#)  
*Data length.*
- uint8 \* [SduPtr](#)  
*Pointer to Sdu.*

### 6.4.2.1.1 Field Documentation

#### 6.4.2.1.1.1 **Pid** `uint8 Pid`

LIN frame identifier.

Definition at line 225 of file `Lpuart_Lin_Ip_Types.h`.

#### 6.4.2.1.1.2 **Cs** `Lpuart_Lin_Ip_FrameCsModelType Cs`

Checksum model type.

Definition at line 226 of file `Lpuart_Lin_Ip_Types.h`.

#### 6.4.2.1.1.3 **Drc** `Lpuart_Lin_Ip_FrameResponseType Drc`

Response type.

Definition at line 227 of file `Lpuart_Lin_Ip_Types.h`.

#### 6.4.2.1.1.4 **Dl** `uint8 Dl`

Data length.

Definition at line 228 of file `Lpuart_Lin_Ip_Types.h`.

#### 6.4.2.1.1.5 **SduPtr** `uint8* SduPtr`

Pointer to Sdu.

Definition at line 229 of file `Lpuart_Lin_Ip_Types.h`.

### 6.4.2.2 **struct Lpuart\_Lin\_Ip\_StateStructType**

Runtime state of the LIN driver.

Definition at line 261 of file `Lpuart_Lin_Ip_Types.h`.



## Data Fields

Type	Name	Description
const uint8 *	TxBuff	The buffer of data being sent.
uint8 *	RxBuff	The buffer of received data.
uint8	CntByte	To count number of bytes already transmitted or received.
volatile uint8	TxSize	The remaining number of bytes to be transmitted.
volatile uint8	RxSize	The remaining number of bytes to be received.
uint8	Checksum	Checksum byte.
volatile boolean	IsBusBusy	True if there are data, frame headers being transferred on bus.
uint8	CurrentPid	Current PID.
volatile Lpuart_Lin_Ip_EventIdType	CurrentEventId	Current ID Event.
volatile Lpuart_Lin_Ip_NodeStateType	CurrentNodeState	Current Node state.
uint32	LinSourceClockFreq	Frequency of the source clock for LIN.
volatile Lpuart_Lin_Ip_NodeStateType	PreviousNodeState	Store previous node state when set Lin channel to idle for further processing.

## 6.4.2.3 struct Lpuart\_Lin\_Ip\_UserConfigType

User configuration structure of the LIN driver.

Definition at line 300 of file Lpuart\_Lin\_Ip\_Types.h.

## Data Fields

- uint8 [Instance](#)  
*Hardware Instance.*
- uint32 [BaudRateDivisor](#)  
*Baudrate divider to be configured in hardware.*
- uint32 [OverSamplingRatio](#)  
*baudrate of LIN Hardware Interface to configure*
- boolean [NodeFunction](#)  
*Node function as Master or Slave.*
- uint8 [BreakLength](#)  
*Length of break character will be transmitted.*
- uint8 [BreakLengthDetect](#)  
*Length of break character can be detected.*
- [Lpuart\\_Lin\\_Ip\\_CallbackType](#) [Callback](#)  
*Callback function to invoke after receiving a byte or transmitting a byte.*
- uint8 [WakeupByte](#)  
*Byte will be sent to generate wakeup pulse [400us->4ms].*
- uint32 [ChannelClock](#)  
*Channel clock.*

### 6.4.2.3.1 Field Documentation

#### 6.4.2.3.1.1 Instance `uint8 Instance`

Hardware Instance.

Definition at line 302 of file `Lpuart_Lin_Ip_Types.h`.

#### 6.4.2.3.1.2 BaudRateDivisor `uint32 BaudRateDivisor`

Baudrate divider to be configured in hardware.

Definition at line 303 of file `Lpuart_Lin_Ip_Types.h`.

#### 6.4.2.3.1.3 OverSamplingRatio `uint32 OverSamplingRatio`

baudrate of LIN Hardware Interface to configure

Definition at line 304 of file `Lpuart_Lin_Ip_Types.h`.

#### 6.4.2.3.1.4 NodeFunction `boolean NodeFunction`

Node function as Master or Slave.

Definition at line 305 of file `Lpuart_Lin_Ip_Types.h`.

#### 6.4.2.3.1.5 BreakLength `uint8 BreakLength`

Length of break character will be transmitted.

Definition at line 306 of file `Lpuart_Lin_Ip_Types.h`.

#### 6.4.2.3.1.6 BreakLengthDetect `uint8 BreakLengthDetect`

Length of break character can be detected.

Definition at line 307 of file `Lpuart_Lin_Ip_Types.h`.

**6.4.2.3.1.7 Callback** `Lpuart_Lin_Ip_CallbackType` Callback

Callback function to invoke after receiving a byte or transmitting a byte.

Definition at line 311 of file `Lpuart_Lin_Ip_Types.h`.

**6.4.2.3.1.8 WakeupByte** `uint8` WakeupByte

Byte will be sent to generate wakeup pulse [400us->4ms].

Definition at line 319 of file `Lpuart_Lin_Ip_Types.h`.

**6.4.2.3.1.9 ChannelClock** `uint32` ChannelClock

Channel clock.

Definition at line 320 of file `Lpuart_Lin_Ip_Types.h`.

**6.4.3 Macro Definition Documentation****6.4.3.1 LPUART\_LIN\_IP\_SLAVE**

```
#define LPUART_LIN_IP_SLAVE
```

Macro that specifies usage of a SLAVE node.

This macro is used in user configuration structure to set the node type as SLAVE.

Definition at line 94 of file `Lpuart_Lin_Ip_Types.h`.

**6.4.3.2 LPUART\_LIN\_IP\_MASTER**

```
#define LPUART_LIN_IP_MASTER
```

Macro that specifies usage of a SLAVE node.

This macro is used in user configuration structure to set the node type as MASTER.

Definition at line 102 of file `Lpuart_Lin_Ip_Types.h`.

### 6.4.3.3 LPUART\_LIN\_IP\_BREAK\_CHAR\_10\_BIT\_MINIMUM\_U8

```
#define LPUART_LIN_IP_BREAK_CHAR_10_BIT_MINIMUM_U8
```

Macro LPUART break char length 10 bit times.

LPUART break char length 10 bit times (if M = 0, SBNS = 0) or 11 (if M = 1, SBNS = 0 or M = 0, SBNS = 1) or 12 (if M = 1, SBNS = 1 or M10 = 1, SNBS = 0) or 13 (if M10 = 1, SNBS = 1)

Definition at line 111 of file Lpuart\_Lin\_Ip\_Types.h.

### 6.4.3.4 LPUART\_LIN\_IP\_BREAK\_CHAR\_13\_BIT\_MINIMUM\_U8

```
#define LPUART_LIN_IP_BREAK_CHAR_13_BIT_MINIMUM_U8
```

Macro LPUART break char length 10 bit times.

LPUART break char length 13 bit times (if M = 0, SBNS = 0 or M10 = 0, SBNS = 1) or 14 (if M = 1, SBNS = 0 or M = 1, SBNS = 1) or 15 (if M10 = 1, SBNS = 1 or M10 = 1, SNBS = 0)

Definition at line 120 of file Lpuart\_Lin\_Ip\_Types.h.

## 6.4.4 Types Reference

### 6.4.4.1 Lpuart\_Lin\_Ip\_CallbackType

```
typedef void(* Lpuart_Lin_Ip_CallbackType) (const uint8 Instance, const Lpuart\_Lin\_Ip\_StateStructType *LinState)
```

LIN Driver callback function type.

Definition at line 291 of file Lpuart\_Lin\_Ip\_Types.h.

## 6.4.5 Enum Reference

### 6.4.5.1 Lpuart\_Lin\_Ip\_EventIdType

```
enum Lpuart\_Lin\_Ip\_EventIdType
```

Enum containing the events related to a ID.

This enum defines types for an enumerating event related to an Identifier.

Enumerator

LPUART_LIN_IP_NO_EVENT	No event.
LPUART_LIN_IP_WAKEUP_SIGNAL	Wakeup signal detected.
LPUART_LIN_IP_BAUDRATE_ADJUSTED	Baudrate adjusted.
LPUART_LIN_IP_RECV_BREAK_FIELD_OK	Break field received.
LPUART_LIN_IP_SYNC_ERROR	Sync byte received with errors.
LPUART_LIN_IP_SEND_HEADER_OK	Sync byte received ok.
LPUART_LIN_IP_RECV_HEADER_OK	PID byte received ok.
LPUART_LIN_IP_PID_ERROR	PID byte received with errors.
LPUART_LIN_IP_FRAME_ERROR	Frame transfer has errors.
LPUART_LIN_IP_READBACK_ERROR	Readback error.
LPUART_LIN_IP_CHECKSUM_ERROR_EVENT	Checksum error.
LPUART_LIN_IP_TX_COMPLETED	Tx completed.
LPUART_LIN_IP_RX_COMPLETED	Rx completed.
LPUART_LIN_IP_RX_OVERRUN_ERROR	Rx overrun error.
LPUART_LIN_IP_TIMEOUT_ERROR	Timeout error.

Definition at line 134 of file Lpuart\_Lin\_Ip\_Types.h.

#### 6.4.5.2 Lpuart\_Lin\_Ip\_NodeStateType

```
enum Lpuart_Lin_Ip_NodeStateType
```

Define type for an enumerating LIN Node state.

Enumerator

LPUART_LIN_IP_NODE_STATE_UNINIT	Uninitialized state.
LPUART_LIN_IP_NODE_STATE_SLEEP_MODE	Sleep mode state.
LPUART_LIN_IP_NODE_STATE_IDLE	Idle state.
LPUART_LIN_IP_NODE_STATE_SEND_BREAK_FIELD	Send break field state.
LPUART_LIN_IP_NODE_STATE_SEND_SYNC	Send the synchronization byte state.
LPUART_LIN_IP_NODE_STATE_RECV_SYNC	Receive the synchronization byte state.
LPUART_LIN_IP_NODE_STATE_SEND_PID	Send PID state.
LPUART_LIN_IP_NODE_STATE_RECV_PID	Receive PID state.
LPUART_LIN_IP_NODE_STATE_RECV_DATA	Receive data state.
LPUART_LIN_IP_NODE_STATE_RECV_DATA_COMPLETED	Receive data completed state.
LPUART_LIN_IP_NODE_STATE_SEND_DATA	Send data state.
LPUART_LIN_IP_NODE_STATE_SEND_DATA_COMPLETED	Send data completed state.

Definition at line 160 of file Lpuart\_Lin\_Ip\_Types.h.

### 6.4.5.3 Lpuart\_Lin\_Ip\_StatusType

enum `Lpuart_Lin_Ip_StatusType`

LPUART status type.

Definition at line 180 of file `Lpuart_Lin_Ip_Types.h`.

### 6.4.5.4 Lpuart\_Lin\_Ip\_FrameCsModelType

enum `Lpuart_Lin_Ip_FrameCsModelType`

Define type for checksum type of the frame.

Enumerator

<code>LPUART_LIN_IP_ENHANCED_CS</code>	Enhanced CheckSum model.
<code>LPUART_LIN_IP_CLASSIC_CS</code>	Classic CheckSum model.

Definition at line 191 of file `Lpuart_Lin_Ip_Types.h`.

### 6.4.5.5 Lpuart\_Lin\_Ip\_FrameResponseType

enum `Lpuart_Lin_Ip_FrameResponseType`

Define type for Response type of the frame.

Enumerator

<code>LPUART_LIN_IP_FRAMERESPONSE_TX</code>	Response is generated from this (master) node.
<code>LPUART_LIN_IP_FRAMERESPONSE_RX</code>	Response is generated from a remote slave node.
<code>LPUART_LIN_IP_FRAMERESPONSE_IGNORE</code>	Response is generated from one slave to another slave. For the master the response will be anonymous, it does not have to receive the response.

Definition at line 201 of file `Lpuart_Lin_Ip_Types.h`.

### 6.4.5.6 Lpuart\_Lin\_Ip\_TransferStatusType

enum `Lpuart_Lin_Ip_TransferStatusType`

Description: This type defines a range of transfer status.

Enumerator

<code>LPUART_LIN_IP_STATUS_FAIL</code>	Command has not been accepted.
<code>LPUART_LIN_IP_STATUS_TX_OK</code>	Transmission successfully.
<code>LPUART_LIN_IP_STATUS_TX_BUSY</code>	Transmission is ongoing.
<code>LPUART_LIN_IP_STATUS_TX_HEADER_ERROR</code>	Erroneous header transmission.
<code>LPUART_LIN_IP_STATUS_TX_ERROR</code>	Erroneous response transmission.
<code>LPUART_LIN_IP_STATUS_RX_OK</code>	Reception of correct response.
<code>LPUART_LIN_IP_STATUS_RX_BUSY</code>	Ongoing reception. At least one response byte has been received.
<code>LPUART_LIN_IP_STATUS_RX_ERROR</code>	Erroneous response reception.
<code>LPUART_LIN_IP_STATUS_RX_NO_RESPONSE</code>	No response byte has been received so far(timeout occurred).
<code>LPUART_LIN_IP_STATUS_RX_HEADER_OK</code>	Slave received a correct header.
<code>LPUART_LIN_IP_STATUS_RX_HEADER_BUSY</code>	Slave is receiving header.
<code>LPUART_LIN_IP_STATUS_RX_HEADER_ERROR</code>	Erroneous header reception of slave.
<code>LPUART_LIN_IP_STATUS_OPERATIONAL</code>	Normal operation.
<code>LPUART_LIN_IP_STATUS_SLEEP</code>	Sleep state operation.

Definition at line 236 of file `Lpuart_Lin_Ip_Types.h`.

## 6.4.6 Function Reference

### 6.4.6.1 Lpuart\_Lin\_Ip\_Init()

```
Lpuart_Lin_Ip_StatusType Lpuart_Lin_Ip_Init (
    const uint8 Instance,
    const Lpuart_Lin_Ip_UserConfigType * UserConfig )
```

Initializes an LIN\_LPUART instance for LIN Network.

The caller provides memory for the driver state structures during initialization. The user must select the LIN\_LPUART clock source in the application to initialize the LIN\_LPUART. This function initializes a LPUART instance for operation. This function will initialize the run-time state structure to keep track of the on-going transfers, initialize the module to user defined settings and default settings, set break field length to be 13 bit times minimum, enable the break detect interrupt, Rx complete interrupt, frame error detect interrupt, and enable the LPUART module transmitter and receiver

### Parameters

<i>Instance</i>	LIN_LPUART instance number
<i>UserConfig</i>	user configuration structure of type #lin_user_config_t

### Returns

LPUART\_LIN\_IP\_STATUS\_SUCCESS - Initialization command has been accepted. LPUART\_LIN\_IP↵  
\_STATUS\_ERROR - Initialization command has not been accepted.

#### 6.4.6.2 Lpuart\_Lin\_Ip\_Deinit()

```
Lpuart_Lin_Ip_StatusType Lpuart_Lin_Ip_Deinit (  
    const uint8 Instance )
```

Shuts down the LIN\_LPUART by disabling interrupts and transmitter/receiver.

### Parameters

<i>Instance</i>	LIN_LPUART instance number
-----------------	----------------------------

### Returns

LPUART\_LIN\_IP\_STATUS\_SUCCESS - De-initialization command has been accepted. LPUART\_LIN↵  
\_IP\_STATUS\_ERROR - De-initialization command has not been accepted. \*

#### 6.4.6.3 Lpuart\_Lin\_Ip\_SendFrame()

```
Lpuart_Lin_Ip_StatusType Lpuart_Lin_Ip_SendFrame (  
    const uint8 Instance,  
    const Lpuart_Lin_Ip_PduType * PduInfo )
```

Sends frame out through the LIN\_LPUART module using non-blocking method.

This enables an a-sync method for transmitting lin frame. Non-blocking means that the function returns immediately. The application has to get the transferring status to know when the frame is complete. This function will calculate the checksum byte and send it with the frame data. If txSize is equal to 0 or greater than 8 or node's current state is in SLEEP mode then the function will return LPUART\_LIN\_IP\_STATUS\_ERROR. If isBusBusy is currently true then the function will return LPUART\_LIN\_IP\_STATUS\_BUSY.



## Parameters

in	<i>Instance</i>	LIN_LPUART instance number
in	<i>PduInfo</i>	PDU containing the ID, Checksum model, Response type, Data Length and SDU data pointer.

## Returns

operation status:

## Return values

<i>LPUART_LIN_IP_STATUS_SUCCESS</i>	- Send command has been accepted.
<i>LPUART_LIN_IP_STATUS_BUSY</i>	- Operation failed due to hardware instance busy.
<i>LPUART_LIN_IP_STATUS_ERROR</i>	- Send command has not been accepted.

#### 6.4.6.4 Lpuart\_Lin\_Ip\_AbortTransferData()

```
Lpuart_Lin_Ip_StatusType Lpuart_Lin_Ip_AbortTransferData (
    const uint8 Instance )
```

Aborts an on-going non-blocking transmission/reception.

While performing a non-blocking transferring data, users can call this function to terminate immediately the transferring.

## Parameters

<i>Instance</i>	LIN_LPUART instance number
-----------------	----------------------------

## Returns

Lpuart\_Lin\_Ip\_StatusType

#### 6.4.6.5 Lpuart\_Lin\_Ip\_GoToSleepMode()

```
Lpuart_Lin_Ip_StatusType Lpuart_Lin_Ip_GoToSleepMode (
    const uint8 Instance )
```

This function puts current node to sleep mode.

This function changes current node state to LIN\_NODE\_STATE\_SLEEP\_MODE

### Parameters

<i>Instance</i>	LIN_LPUART instance number
-----------------	----------------------------

### Returns

function always return LPUART\_LIN\_IP\_STATUS\_SUCCESS

#### 6.4.6.6 Lpuart\_Lin\_Ip\_GoToIdleState()

```
void Lpuart_Lin_Ip_GoToIdleState (  
    const uint8 Instance )
```

Puts current LIN node to Idle state This function changes current node state to LIN\_NODE\_STATE\_IDLE.

### Parameters

<i>Instance</i>	LIN_LPUART instance number
-----------------	----------------------------

### Returns

function always return LPUART\_LIN\_IP\_STATUS\_SUCCESS

#### 6.4.6.7 Lpuart\_Lin\_Ip\_SendWakeupSignal()

```
Lpuart_Lin_Ip_StatusType Lpuart_Lin_Ip_SendWakeupSignal (  
    const uint8 Instance )
```

Sends a wakeup signal through the LIN\_LPUART interface.

### Parameters

<i>Instance</i>	LIN_LPUART instance number
-----------------	----------------------------

### Returns

operation status:

Return values

<i>LPUART_LIN_IP_STATUS_SUCCESS</i>	: Command has been accepted.
<i>LPUART_LIN_IP_STATUS_ERROR</i>	: Command has not been accepted, error occurred.

#### 6.4.6.8 Lpuart\_Lin\_Ip\_GetStatus()

```
Lpuart_Lin_Ip_TransferStatusType Lpuart_Lin_Ip_GetStatus (
    const uint8 Instance,
    const uint8 ** Buffer )
```

Returns the status of an on going transmission.

This function returns the status of the current non-blocking transfer. If a response reception has been successfully received, Buffer will be referenced to receive buffer.

Parameters

<i>channel</i>	LIN_LPUART instance number.
<i>Buffer</i>	Pointer to the received data buffer.

Returns

operation status: LPUART\_Lin\_Ip\_TransferStatusType

Return values

<i>LPUART_LIN_IP_STATUS_FAIL</i>	Command has not been accepted .
<i>LPUART_LIN_IP_STATUS_TX_OK,Master</i>	Header or Response successful transmission. Slave: Response successful transmission.
<i>LPUART_LIN_IP_STATUS_TX_BUSY,Master</i>	Ongoing transmission (Header or Response). Slave: Ongoing transmission response.
<i>LPUART_LIN_IP_STATUS_TX_HEADER_ERROR,Master</i>	Erroneous header transmission such as: Mismatch between sent and read back data or Physical bus error./
<i>LPUART_LIN_IP_STATUS_TX_ERROR,Master</i>	or Slave: Erroneous response transmission such as mismatch between sent and read back data, Physical bus error
<i>LPUART_LIN_IP_STATUS_RX_OK,Master</i>	or Slave: Reception of correct response.
<i>LPUART_LIN_IP_STATUS_RX_BUSY,Master</i>	or Slave: Ongoing reception. At least one response byte has been received, but the checksum byte has not been received.
<i>LPUART_LIN_IP_STATUS_RX_ERROR,Master</i>	or Slave: Erroneous response reception such as: - Framing error - Overrun error - Checksum error or Short response(timeout occurred).

Return values

<i>LPUART_LIN_IP_STATUS_RX_NO_RESPONSE, Master</i>	or Slave: No response byte has been received so far(timeout occurred).
<i>LPUART_LIN_IP_STATUS_RX_HEADER_OK, Slave</i>	Reception of correct header.
<i>LPUART_LIN_IP_STATUS_RX_HEADER_BUSY, Slave</i>	Ongoing header reception.
<i>LPUART_LIN_IP_STATUS_RX_HEADER_ERROR, Slave</i>	Erroneous header reception such as: Break field error, Sync field error, Identifier parity error, Framing error, timeout occurred or Physical bus error.
<i>LPUART_LIN_IP_STATUS_OPERATION_IDLE, Normal</i>	operation; there is no data has been sent or received after channel initialized or switched to IDLE from SLEEP.
<i>LPUART_LIN_IP_STATUS_SLEEP</i>	Sleep state operation.

### 6.4.6.9 Lpuart\_Lin\_Ip\_IRQHandler()

```
void Lpuart_Lin_Ip_IRQHandler (
    const uint8 Instance )
```

This is callback function for Timer Interrupt Handler. Users shall initialize a timer (for example FTM) and the time period in microseconds will be set by the driver via LinLpuartStartTimerNotification. In timer IRQ handler, call this function.

Parameters

<i>Instance</i>	LIN_LPUART instance number
-----------------	----------------------------

Returns

None

LIN\_LPUART interrupt handler for RX\_TX and Error interrupts.

Parameters

<i>Instance</i>	LIN_LPUART instance number
-----------------	----------------------------

Returns

void

**How to Reach Us:**

**Home Page:**

[nxp.com](http://nxp.com)

**Web Support:**

[nxp.com/support](http://nxp.com/support)

Information in this document is provided solely to enable system and software implementers to use NXP products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document. NXP reserves the right to make changes without further notice to any products herein.

NXP makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does NXP assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters that may be provided in NXP data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including "typicals," must be validated for each customer application by customer's technical experts. NXP does not convey any license under its patent rights nor the rights of others. NXP sells products pursuant to standard terms and conditions of sale, which can be found at the following address: [nxp.com/SalesTermsandConditions](http://nxp.com/SalesTermsandConditions).

NXP, the NXP logo, NXP SECURE CONNECTIONS FOR A SMARTER WORLD, COOLFLUX, EMBRACE, GREENCHIP, HITAG, I2C BUS, ICODE, JCOP, LIFE VIBES, MIFARE, MIFARE CLASSIC, MIFARE DESFire, MIFARE PLUS, MIFARE FLEX, MANTIS, MIFARE ULTRALIGHT, MIFARE4MOBILE, MIGLO, NTAG, ROADLINK, SMARTLX, SMARTMX, STARPLUG, TOPFET, TRENCHMOS, UCODE, Freescale, the Freescale logo, Altivec, C-5, CodeTEST, CodeWarrior, ColdFire, ColdFire+, C-Ware, the Energy Efficient Solutions logo, Kinetis, Layerscape, MagniV, mobileGT, PEG, PowerQUICC, Processor Expert, QorIQ, QorIQ Qonverge, Ready Play, SafeAssure, the SafeAssure logo, StarCore, Symphony, VortiQa, Vybrid, Airfast, BeeKit, BeeStack, CoreNet, Flexis, MXC, Platform in a Package, QUICC Engine, SMARTMOS, Tower, TurboLink, and UMEMS are trademarks of NXP B.V. All other product or service names are the property of their respective owners. ARM, AMBA, ARM Powered, Artisan, Cortex, Jazelle, Keil, SecurCore, Thumb, TrustZone, and Vision are registered trademarks of ARM Limited (or its subsidiaries) in the EU and/or elsewhere. ARM7, ARM9, ARM11, big.LITTLE, CoreLink, CoreSight, DesignStart, Mali, mbed, NEON, POP, Sensinode, Socrates, ULINK and Versatile are trademarks of ARM Limited (or its subsidiaries) in the EU and/or elsewhere. All rights reserved. Oracle and Java are registered trademarks of Oracle and/or its affiliates. The Power Architecture and Power.org word marks and the Power and Power.org logos and related marks are trademarks and service marks licensed by Power.org.

© 2023 NXP B.V.

