User Manual

for S32K1 LIN Driver

Document Number: UM2LINASR4.4 Rev0000R1.0.1 Rev. 1.0

1 Revision History 2

2 Introduction 3

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

3 Driver 7

3.1 Requirements . . . 7
3.2 Driver Design Summary . . . 7
3.3 Hardware Resources . . . 8
3.3.1 Low Power Universal Asynchronous Receiver/Transmitter (LPUART): . . . 8
3.4 Deviations from Requirements . . . 10
3.5 Driver Limitations . . . 13
3.6 Driver usage and configuration tips . . . 13
3.6.1 Dual Clock Feature . . . 13
3.6.2 Frame Timeout Disable Feature . . . 14
3.6.3 Auto-detect baudrate feature . . . 16
3.6.4 How to configure a FLEXIO channel . . . 18
3.7 Runtime errors . . . 19
3.8 Symbolic Names Disclaimer . . . 20

4 Tresos Configuration Plug-in 21

4.1 Module Lin . . . 22
4.2 Container AutosarExt . . . 22
4.3 Parameter LinDisableDemReportErrorStatus . . . 23
4.4 Parameter LinFrameTimeoutDisable . . . 23
4.5 Parameter LinEnableUserModeSupport . . . 25
4.6 Parameter LinLpuartStartTimerNotification . . . 26
4.7 Parameter LinLpuartStopTimerNotification . . . 26
4.8 Parameter LinFlexioStartTimerNotification . . . 27
4.9 Parameter LinFlexioStopTimerNotification . . . 27
4.10 Container LinGeneral . . . 28
4.11 Parameter LinMulticoreSupport . . . 28
4.12 Parameter LinDevErrorDetect . . . 29
4.13 Parameter LinIndex . . . 29
4.14 Parameter LinTimeoutMethod . . . 30
4.15 Parameter LinTimeoutDuration . . . 30
4.16 Parameter LinVersionInfoApi . . . 31

4.17 Reference LinEcucPartitionRef . . . 31
4.18 Container LinDemEventParameterRefs . . . 32
4.19 Reference LIN_E_TIMEOUT . . . 32
4.20 Container LinGlobalConfig . . . 33
4.21 Container LinChannel . . . 33
4.22 Parameter LinChannelId . . . 34
4.23 Parameter LinNodeType . . . 34
4.24 Parameter LinChannelBaudRate . . . 35
4.25 Parameter BreakLength . . . 35
4.26 Parameter DetectedBreakLength . . . 36
4.27 Parameter LinResponseTimeout . . . 36
4.28 Parameter LinHeaderTimeout . . . 37
4.29 Parameter LinHwChannel . . . 38
4.30 Parameter LinChannelWakeupSupport . . . 38
4.31 Reference LinClockRef . . . 39
4.32 Reference LinClockRef_Alternate . . . 39
4.33 Reference LinChannelEcuMWakeupSource . . . 39
4.34 Reference LinChannelEcucPartitionRef . . . 40
4.35 Reference LinFlexioRxControllerRef . . . 40
4.36 Reference LinFlexioTxControllerRef . . . 41
4.37 Container CommonPublishedInformation . . . 41
4.38 Parameter ArReleaseMajorVersion . . . 42
4.39 Parameter ArReleaseMinorVersion . . . 42
4.40 Parameter ArReleaseRevisionVersion . . . 43
4.41 Parameter ModuleId . . . 43
4.42 Parameter SwMajorVersion . . . 44
4.43 Parameter SwMinorVersion . . . 44
4.44 Parameter SwPatchVersion . . . 45
4.45 Parameter VendorApiInfix . . . 45
4.46 Parameter VendorId . . . 46
4.47 Configuration elements of Lin . . . 46
4.48 Form POST_BUILD_VARIANT_USED . . . 46
4.49 Attribute IMPLEMENTATION_CONFIG_VARIANT detailed description . . . 46
4.50 Form IMPLEMENTATION_CONFIG_VARIANT . . . 47
4.51 Attribute IMPLEMENTATION_CONFIG_VARIANT detailed description . . . 47
4.52 Form AutosarExt . . . 47
4.52.1 LinDisableDemReportErrorStatus (AutosarExt) . . . 47
4.53 Attribute LinDisableDemReportErrorStatus (AutosarExt) detailed description . . . 47
4.53.1 LinEnableUserModeSupport (AutosarExt) . . . 48
4.54 Attribute LinEnableUserModeSupport (AutosarExt) detailed description . . . 48

4.55 Form LinGeneral . . . 48
4.55.1 LinMulticoreSupport (LinGeneral) . . . 48
4.56 Attribute LinMulticoreSupport (LinGeneral) detailed description . . . 48
4.56.1 LinDevErrorDetect (LinGeneral) . . . 49
4.57 Attribute LinDevErrorDetect (LinGeneral) detailed description . . . 49
4.57.1 LinIndex (LinGeneral) . . . 49
4.58 Attribute LinIndex (LinGeneral) detailed description . . . 49
4.58.1 LinTimeoutDuration (LinGeneral) . . . 49
4.59 Attribute LinTimeoutDuration (LinGeneral) detailed description . . . 49
4.59.1 LinVersionInfoApi (LinGeneral) . . . 50
4.60 Attribute LinVersionInfoApi (LinGeneral) detailed description . . . 50
4.60.1 LinEcucPartitionRef (LinGeneral) . . . 50
4.61 Attribute LinVersionInfoApi (LinGeneral) detailed description . . . 50
4.62 Form LinDemEventParameterRefs . . . 50
4.62.1 LIN_E_TIMEOUT (LinDemEventParameterRefs) . . . 51
4.63 Attribute LIN_E_TIMEOUT (LinDemEventParameterRefs) detailed description . . . 51
4.64 Form LinGlobalConfig . . . 51
4.64.1 Form LinChannel . . . 51
4.65 Attribute LinChannelId (LinChannel) detailed description . . . 51
4.66 Attribute LinChannelBaudRate (LinChannel) detailed description . . . 52
4.67 Attribute LinHwChannel (LinChannel) detailed description . . . 52
4.68 Attribute LinClockRef (LinChannel) detailed description . . . 52
4.69 Attribute LinClockRef_Alternate (LinChannel) detailed description . . . 53
4.70 Attribute LinChannelWakeupSupport (LinChannel) detailed description . . . 53
4.71 Attribute LinChannelEcuMWakeupSource (LinChannel) detailed description . . . 53
4.72 Attribute LinChannelEcucPartitionRef (LinChannel) detailed description . . . 54

5 Module Index **55**
5.1 Software Specification . . . 55

6 Module Documentation **56**
6.1 FLEXIO_IP . . . 56
6.1.1 Detailed Description . . . 56
6.1.2 Data Structure Documentation . . . 57
6.1.3 Macro Definition Documentation . . . 60
6.1.4 Types Reference . . . 60
6.1.5 Enum Reference . . . 60
6.1.6 Function Reference . . . 63
6.2 LIN Driver . . . 69
6.2.1 Detailed Description . . . 69
6.2.2 Data Structure Documentation . . . 70

6.2.3 Macro Definition Documentation 71
6.2.4 Function Reference 75
6.3 LIN 79
6.3.1 Detailed Description 79
6.4 Lpuart Lin IPL 80
6.4.1 Detailed Description 80
6.4.2 Data Structure Documentation 81
6.4.3 Macro Definition Documentation 85
6.4.4 Types Reference 86
6.4.5 Enum Reference 86
6.4.6 Function Reference 89

Chapter 1

Revision History

Revision	Date	Author	Description
1.0	24.02.2022	NXP RTD Team	Prepared for release RTD S32K1 Version 1.0.1

Chapter 2

Introduction

- Supported Derivatives
- Overview
- About This Manual
- Acronyms and Definitions
- Reference List

This User Manual describes NXP Semiconductor AUTOSAR LIN for S32K1XX. 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:

- s32k116_qfn32
- s32k116_lqfp48
- s32k118_lqfp48
- s32k118_lqfp64
- s32k142_lqfp48
- s32k142_lqfp64
- s32k142_lqfp100
- s32k142w_lqfp48
- s32k142w_lqfp64
- s32k144_lqfp48

- s32k144_lqfp64
- s32k144_lqfp100
- s32k144_mapbga100
- s32k144w_lqfp48
- s32k144w_lqfp64
- s32k146_lqfp64
- s32k146_lqfp100
- s32k146_mapbga100
- s32k146_lqfp144
- s32k148_lqfp100
- s32k148_mapbga100
- s32k148_lqfp144
- s32k148_lqfp176

All of the above microcontroller devices are collectively named as S32K1.

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 Signifigant 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 4.4.0
2	S32K1XX Reference Manual	S32K1xx Series Reference Manual, Rev. 14, 09/2021
3	Errata	S32K116_0N96V Rev. 22/OCT/2021
		S32K118_0N97V Rev. 22/OCT/2021
		S32K142_0N33V Rev. 22/OCT/2021
		S32K144_0N57U Rev. 22/OCT/2021
		S32K144W_0P64A Rev. 22/OCT/2021
		S32K146_0N73V Rev. 22/OCT/2021
		S32K148_0N20V Rev. 22/OCT/2021
4	Datasheet	S32K1xx Data Sheet, Rev. 14, 08/2021

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 S32K1XX 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 S32K1XX family supports the following derivatives S32K116, S32K118, S32K142, S32K142W, S32K144, S32↩ K144W, S32K146, and S32K148.

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

3.3.1 Low Power Universal Asynchronous Receiver/Transmitter (LPUART):

The L↩ PUART Module Hardware Instances on the S32K1XX family derivatives are listed in the following table.

Chip	Instances
S32K116	LPUART0
	LPUART1
S32K118	LPUART0
	LPUART1
S32K142	LPUART0
	LPUART1
S32K144	LPUART0
	LPUART1
	LPUART2
S32K144W	LPUART0
	LPUART1
	LPUART2
S32K146	LPUART0
	LPUART1
	LPUART2
S32K148	LPUART0
	LPUART1
	LPUART2

Figure 3.1 LPUART configuration in Reference manual

In order to configure a LPUART or FLEXIO LIN channel the following steps must be followed:

3.3.1.1 Autosar configuration:

Go to Mcu component and enable the LPUART or FLEXIO Clock which is used by the desired Lpuart hardware channel. Create a refference to this clock.

Go to Lin component in your configurator. Select the desired Lpuart hardware channel. Use the Mcu refference previously configured.

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.

Go to Platform component and enable the Lpuart hardware channel interrupt and configure the ISR Handler.

Go to Port component and create two Pins in the Port Container. Those pins must be configured as RX and TX, specific for the hardware instance used.

For example:

In EB tresos and also Design Studio, *LPUART_0* channel has named *LPUART_0*, correspondingly as below:

Figure 3.2 Lin hardware channel configuration in EB Tresos.

The peripheral clocks which must be enabled are LPUARTX_CLK where X is the number of hw instance used and FLEXIO_CLK.

In order to configure the pins, the files from Reference Manual shall be interogated in order to determine the PCR value of the required pin.

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, not fully 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, not implemented or out of scope for the LIN driver.

Requirement	Status	Description	Notes
SWS_Lin_00177	N/S	In case several LIN driver instances (of same or different vendor) are implemented in one ECU the file names, API names, and published parameters must be modified such that no two definitions with the same name are generated. The name shall be extended according to SRS_BSW_00347 with a Vendor Id (needed to distinguish LIN drivers from different vendors) and a Vendor specific name (needed to distinguish different hardware units implemented by one Vendor).	Rejection reason: There is only one LIN driver instance.
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.

Requirement	Status	Description	Notes
SWS_Lin_00999	N/S	These requirements are not applicable to this specification. (SRS_BSW_00307, SRS_BSW_00312, SRS_BSW_00325, SRS_BSW_00326, SRS_BSW_00328, SRS_BSW_00329, SRS_BSW_00330, SRS_BSW_00331, SRS_BSW_00336, SRS_BSW_00339, SRS_BSW_00342, SRS_BSW_00343, SRS_BSW_00353, SRS_BSW_00357, SRS_BSW_00359, SRS_BSW_00360, SRS_BSW_00361, SRS_BSW_00373, SRS_BSW_00376, SRS_BSW_00378, SRS_BSW_00383, SRS_BSW_00395, SRS_BSW_00397, SRS_BSW_00398, SRS_BSW_00399, SRS_BSW_00400, SRS_BSW_00413, SRS_BSW_00415, SRS_BSW_00416, SRS_BSW_00417, BSW00420, SRS_BSW_00422, SRS_BSW_00423, SRS_BSW_00424, SRS_BSW_00425, SRS_BSW_00426, SRS_BSW_00427, SRS_BSW_00428, SRS_BSW_00429, BSW00431, SRS_BSW_00432, SRS_BSW_00433, BSW00434, SRS_BSW_00005, SRS_BSW_00007, SRS_BSW_00162, SRS_BSW_00168, SRS_SPAL_12056, SRS_SPAL_12267, SRS_SPAL_12163, SRS_SPAL_12463, SRS_SPAL_12075, SRS_SPAL_12078, SRS_SPAL_12092, SRS_Lin_01551, SRS_Lin_01568, SRS_Lin_01569, SRS_Lin_01570, SRS_Lin_01564, SRS_Lin_01546, SRS_Lin_01561, SRS_Lin_01549, SRS_Lin_01571, SRS_Lin_01514, SRS_Lin_01515, SRS_Lin_01502, SRS_Lin_01558, BSW01527, SRS_Lin_01523, SRS_Lin_01540, SRS_Lin_01545, SRS_Lin_01534, SRS_Lin_01574, SRS_Lin_01539, SRS_Lin_01544, SRS_Lin_01590).	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.

Requirement	Status	Description	Notes
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
- In Master mode, the break byte length is not configurable. The only length supported is 13 bits long.
- Break length detection is not configurable (we only support 11-bits break length detect).
- Autobaud feature is not implemented over FLEXIO module.
- When using the Autobaudrate feature for S32K11X platforms, the driver is able to detect only baudrate values less than 19200 bps.

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.

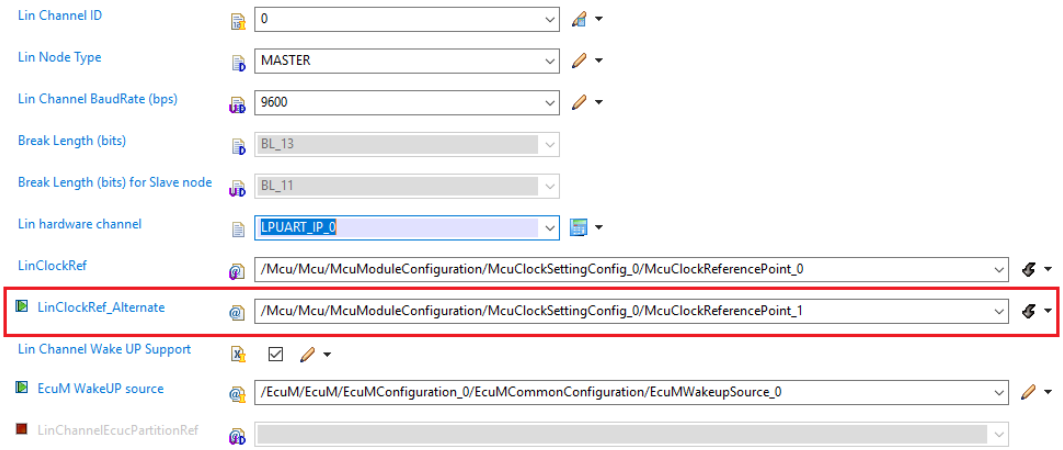

Figure 3.3 Dual Clock Feature configuration node in EB Tresos.

3.6.2 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.

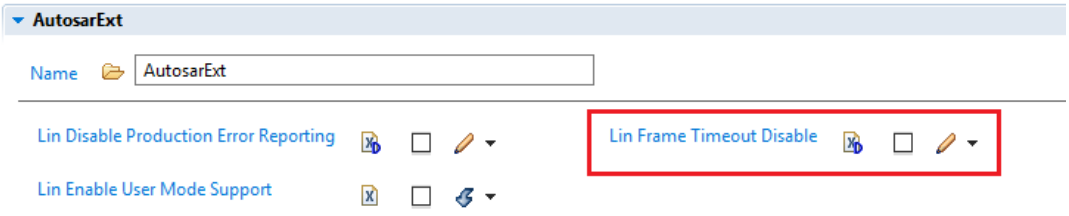

Figure 3.4 Frame Timeout Disable Feature configuration node in EB Tresos.

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.

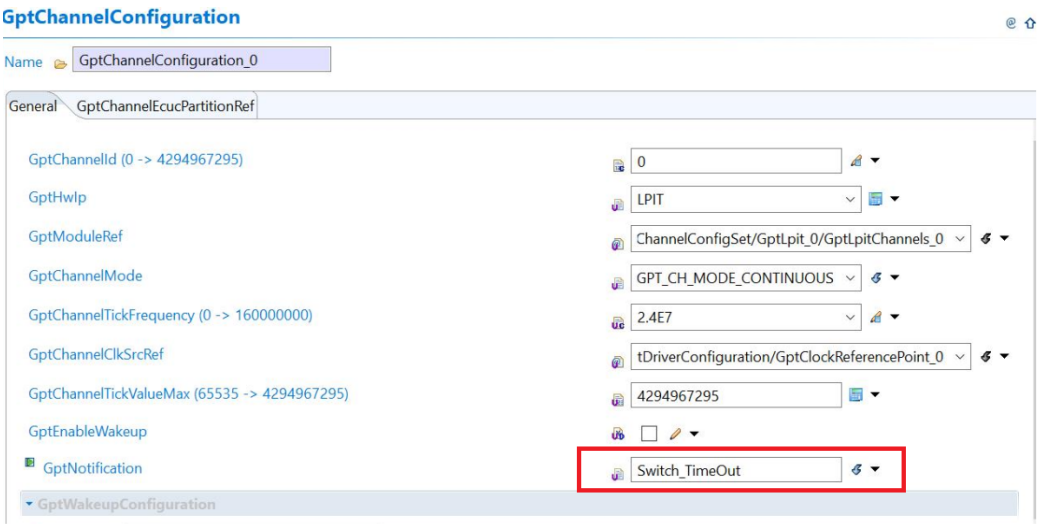

Figure 3.5 Timeout notification function configured in EB Tresos.

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.

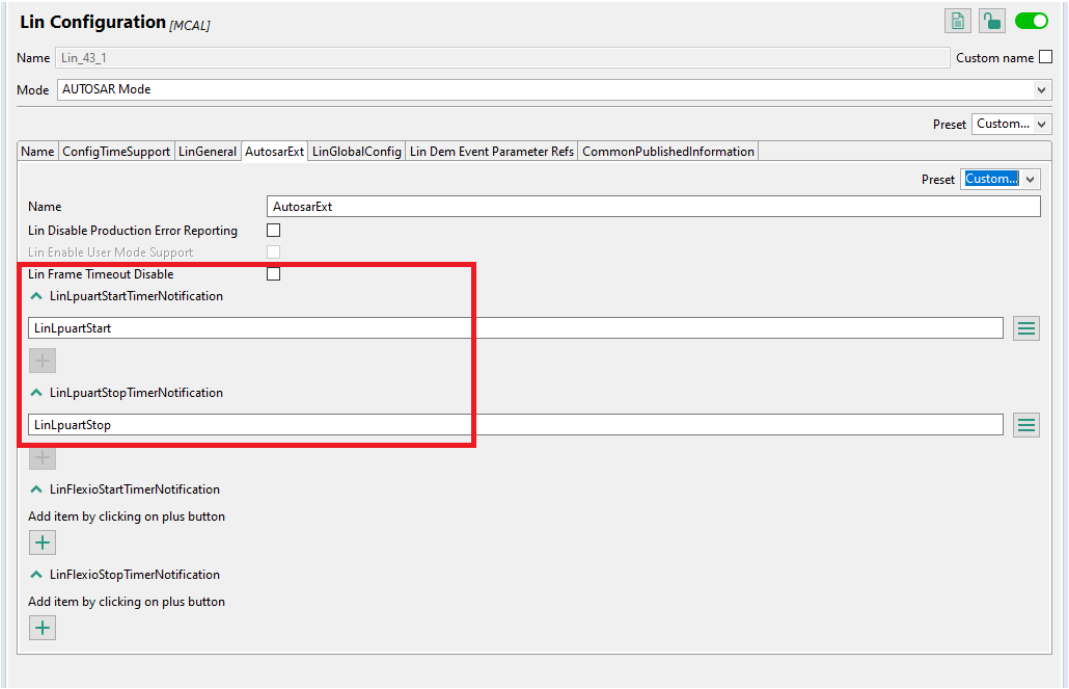

Figure 3.6 Frame Timeout Enabled Feature and notification functions configured in Design Studio.

3.6.3 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 triger 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.

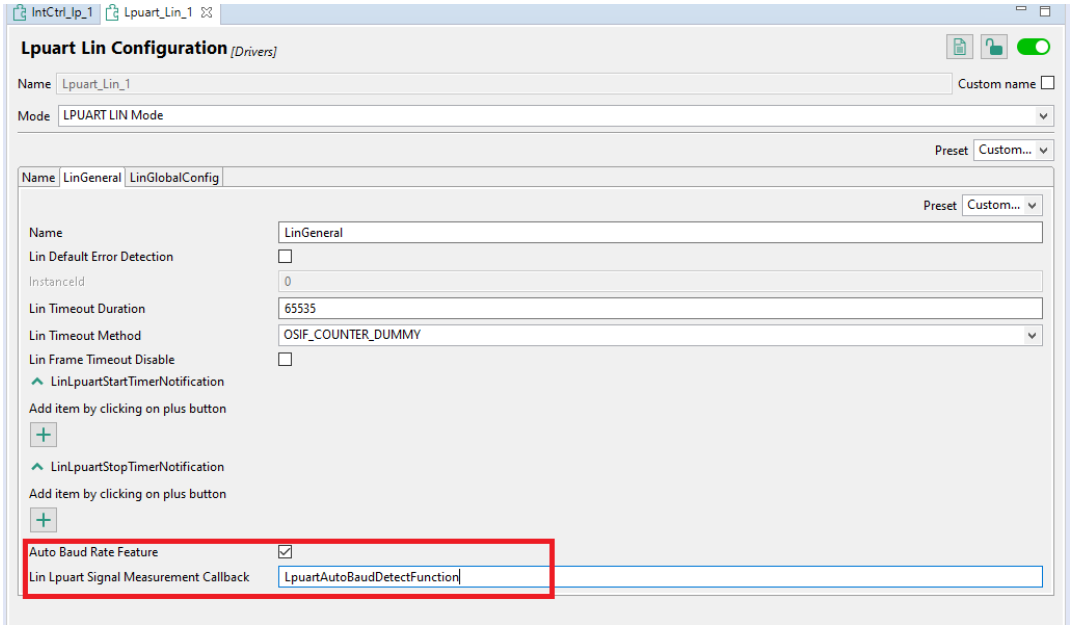

Figure 3.7 Autobaud Feature and notification functions configured in Design Studio

Enable Auto baud rate in LinChannel container:

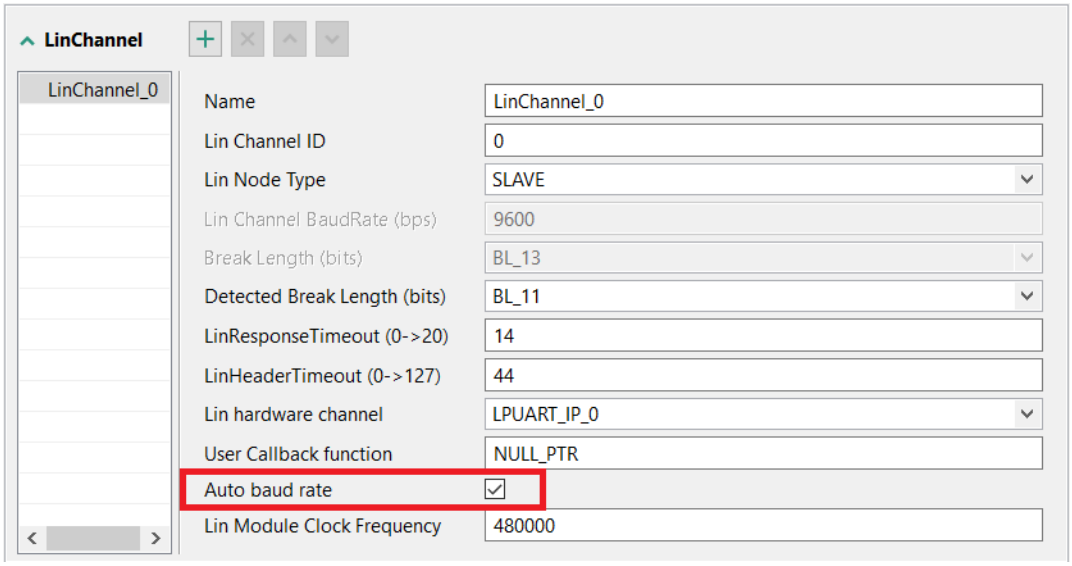

Figure 3.8 Autobaud Feature configured for channel in Design Studio

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.

3.6.4 How to configure a FLEXIO channel

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

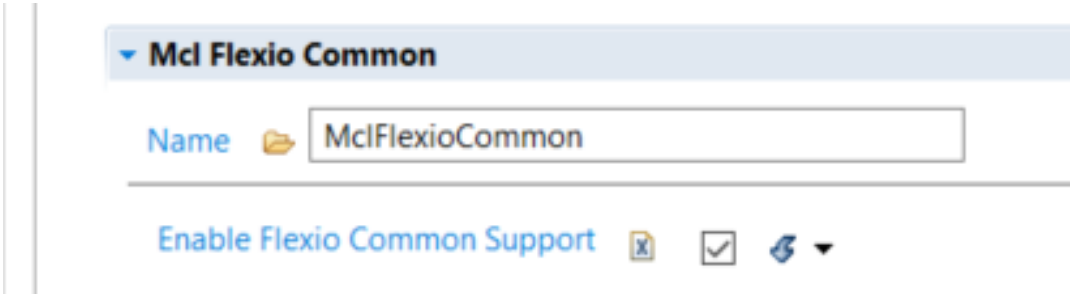

Figure 3.9 Mcl Flexio Common check.

After enabling the Mcl Flexio Common, go to Mcl/MclConfig/FlexioCommon/FlexioCommon_0/FlexioMclLogic↩ Channels 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.

Figure 3.10 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 selectthe Mcl Flexio Logic Channels configurted in the Mcl component. The same channel must **NOT** be configured for both Tx and Rx.

Note

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

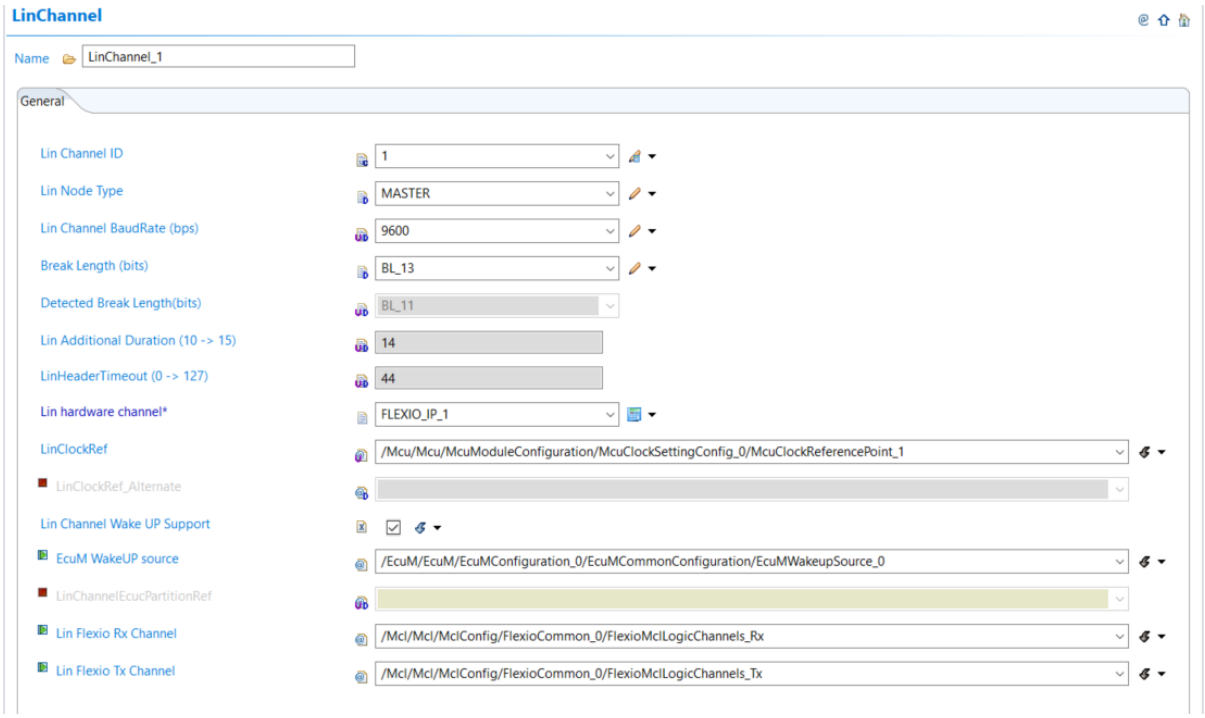

Figure 3.11 Configuration for LIN channel using FLEXIO hardware unit.

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 AutosarExt
    - Parameter LinDisableDemReportErrorStatus
    - Parameter LinFrameTimeoutDisable
    - Parameter LinEnableUserModeSupport
    - Parameter LinLpuartStartTimerNotification
    - Parameter LinLpuartStopTimerNotification
    - Parameter LinFlexioStartTimerNotification
    - Parameter LinFlexioStopTimerNotification
  - Container LinGeneral
    - Parameter LinMulticoreSupport
    - Parameter LinDevErrorDetect
    - Parameter LinIndex
    - Parameter LinTimeoutMethod
    - Parameter LinTimeoutDuration
    - Parameter LinVersionInfoApi
    - Reference LinEcucPartitionRef
  - Container LinDemEventParameterRefs
    - Reference LIN_E_TIMEOUT
  - 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 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:

- AutosarExt
- LinGeneral
- LinDemEventParameterRefs
- LinGlobalConfig
- CommonPublishedInformation

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

4.2 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

4.3 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

4.4 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

4.5 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

4.6 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

4.7 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

4.8 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

4.9 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

4.10 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

4.11 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).

Note: S32K1XX does not support multi-core feature.

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

4.12 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
defaultValue	false

4.13 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
defaultValue	0
max	255
min	0

4.14 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_COU↩ NTER_CUSTOM']

4.15 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

Property	Value
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	1000
max	65535
min	0

4.16 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
defaultValue	false

4.17 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

Property	Value
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

4.18 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

4.19 Reference LIN_E_TIMEOUT

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

Property	Value
type	ECUC-REFERENCE-DEF
origin	AUTOSAR_ECUC
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE

Property	Value
	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

4.20 Container LinGlobalConfig

This container contains the global configuration parameter of the Lin driver. This container is a MultipleConfigurationContaine
i.e. this container and its sub-containers exit 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

4.21 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

4.22 Parameter LinChannelId

Identifies the LIN channel. Replaces LIN_CHANNEL_INDEX_NAME from the LIN SWS.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
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

4.23 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']

4.24 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
defaultValue	9600
max	20000
min	1000

4.25 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
defaultValue	BL_13
literals	['BL_13']

4.26 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']

4.27 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 x 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

Property	Value
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	14
max	20
min	0

4.28 Parameter LinHeaderTimeout

Header timeout value.

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

Example:

- Header_nominal = 13 + 2 + 10 + 10 = 35 (Bit time)

- Additional 40% duration compared to the nominal transmission time, so Header_max_time = 1.4 * 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

4.29 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', 'FLEXIO_IP_0', 'F↩ LEXIO_IP_1']

4.30 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

4.31 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/EcucDefs/Mcu/McuModuleConfiguration/McuClockSetting↩ Config/McuClockReferencePoint

4.32 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/EcucDefs/Mcu/McuModuleConfiguration/McuClockSetting↩ Config/McuClockReferencePoint

4.33 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
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/EcuM/EcuMConfiguration/EcuMCommon↩ Configuration/EcuMWakeupSource

4.34 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/EcucDefs/EcuC/EcucPartitionCollection/EcucPartition

4.35 Reference LinFlexioRxControllerRef

Lin Flexio Rx Channel Reference

Reference to the Flexio Controller configure for the Reception

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_T40D2M10I1R0/Mcl/MclConfig/FlexioCommon/FlexioMclLogicChannels']

4.36 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_T40D2M10I1R0/Mcl/MclConfig/FlexioCommon/FlexioMclLogicChannels']

4.37 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

4.38 Parameter ArReleaseMajorVersion

Major 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	4
max	4
min	4

4.39 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

Property	Value
	VARIANT-PRE-COMPILE: PUBLISHED-INFORMATION
defaultValue	4
max	4
min	4

4.40 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
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

4.41 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

Property	Value
max	82
min	82

4.42 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
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PUBLISHED-INFORMATION
	VARIANT-PRE-COMPILE: PUBLISHED-INFORMATION
defaultValue	1
max	1
min	1

4.43 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

4.44 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
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PUBLISHED-INFORMATION
	VARIANT-PRE-COMPILE: PUBLISHED-INFORMATION
defaultValue	1
max	1
min	1

4.45 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	

4.46 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.

4.47 Configuration elements of Lin

Included forms:

- POST_BUILD_VARIANT_USED
- IMPLEMENTATION_CONFIG_VARIANT
- AutosarExt
- LinGeneral
- LinDemEventParameterRefs
- LinGlobalConfig
- CommonPublishedInformation

4.48 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.

4.49 Attribute IMPLEMENTATION_CONFIG_VARIANT detailed description

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

4.50 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.

4.51 Attribute IMPLEMENTATION_CONFIG_VARIANT detailed description

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

4.52 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.

4.52.1 LinDisableDemReportErrorStatus (AutosarExt)

Switches the Diagnostic Error Reporting and Notification OFF

4.53 Attribute LinDisableDemReportErrorStatus (AutosarExt) detailed description

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

4.53.1 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.

4.54 Attribute LinEnableUserModeSupport (AutosarExt) detailed description

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

4.55 Form LinGeneral

Autosar Requirements: ECUC_Lin_00183.

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

4.55.1 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).

4.56 Attribute LinMulticoreSupport (LinGeneral) detailed description

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

4.56.1 LinDevErrorDetect (LinGeneral) Autosar Requirements: ECUC_Lin_00066.

Switches the Default Error Detection and Notification ON or OFF.

4.57 Attribute LinDevErrorDetect (LinGeneral) detailed description

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

4.57.1 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.

4.58 Attribute LinIndex (LinGeneral) detailed description

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

4.58.1 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

4.59 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

4.59.1 LinVersionInfoApi (LinGeneral) Autosar Requirements: ECUC_Lin_00067.

Switches the Lin_GetVersionInfo function ON or OFF.

4.60 Attribute LinVersionInfoApi (LinGeneral) detailed description

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

4.60.1 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.

4.61 Attribute LinVersionInfoApi (LinGeneral) detailed description

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

4.62 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.

4.62.1 LIN_E_TIMEOUT (LinDemEventParameterRefs)

4.63 Attribute LIN_E_TIMEOUT (LinDemEventParameterRefs) detailed description

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

4.64 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.

4.64.1 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.

4.64.1.1 LinChannelId (LinChannel)

Identifies the LIN channel. Replaces LIN_CHANNEL_INDEX_NAME from the LIN SWS.

4.65 Attribute LinChannelId (LinChannel) detailed description

Property	Value
Label	Lin Channel ID
Type	INTEGER
Origin	AUTOSAR_ECUC
Symbolic Name	true
Invalid	Range >= 0 <= (N - 1) (N is number of configured channels)

4.65.0.1 LinChannelBaudRate (LinChannel)

Autosar Requirements: ECUC_Lin_00180.

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

4.66 Attribute LinChannelBaudRate (LinChannel) detailed description

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

4.66.0.1 LinHwChannel (LinChannel)

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

4.67 Attribute LinHwChannel (LinChannel) detailed description

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

4.67.0.1 LinClockRef (LinChannel)

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

4.68 Attribute LinClockRef (LinChannel) detailed description

Property	Value
Label	LinClockRef
Type	REFERENCE
Origin	AUTOSAR_ECUC
Symbolic Name	false

4.68.0.1 LinClockRef_Alternate (LinChannel)

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

4.69 Attribute LinClockRef_Alternate (LinChannel) detailed description

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

4.69.0.1 LinChannelWakeupSupport (LinChannel)

Autosar Requirements: ECUC_Lin_00182,

Specifies if the LIN hardware channel supports wake up functionality.

4.70 Attribute LinChannelWakeupSupport (LinChannel) detailed description

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

4.70.0.1 LinChannelEcuMWakeupSource (LinChannel)

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

4.71 Attribute LinChannelEcuMWakeupSource (LinChannel) detailed description

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

4.71.0.1 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.

4.72 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 56
LIN Driver 69
LIN 79
Lpuart Lin IPL 80

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_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_ST↩ ATE_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, 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 LinLpuartStartTimerNotification. 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_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.
Flexio_Lin_Ip_CallbackType	Callback	Callback function to invoke after receiving a byte or transmitting a byte.
Flexio_Lin_Ip_StateStructType *	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_ER↩ 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.
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_ER↩ ROR	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_FL↩ EXIO 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

Channel	LIN_FLEXIO channel number
UserConfig	user configuration structure of type Flexio_Lin_Ip_UserConfigType

Returns

Flexio_Lin_Ip_StatusType

Return values

FLEXIO_LIN_IP_STATUS_SUCCESS	: Init command has been accepted.
FLEXIO_LIN_IP_STATUS_ERROR	: 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_ST↩ ATE_IDLE.

Parameters

channel	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

channel	LIN_FLEXIO channel number

Returns

operation status:

Return values

FLEXIO_LIN_IP_STATUS_SUCCESS	: The frame was aborted
FLEXIO_LIN_IP_STATUS_ERROR	: 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

channel	LIN_FLEXIO channel number

Returns

Flexio_Lin_Ip_StatusType

Return values

FLEXIO_LIN_IP_STATUS_SUCCESS	: 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

channel	LIN_FLEXIO channel number

Returns

operation status:

Return values

FLEXIO_LIN_IP_STATUS_SUCCESS	: Command has been accepted.
FLEXIO_LIN_IP_STATUS_ERROR	: 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

channel	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,
            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

channel	LIN_FLEXIO instance number.
Buffer	Pointer to the received data buffer.

Returns

operation status: Flexio_Lin_Ip_TransferStatusType

Return values

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_ERR↩ OR,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.
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_RESPON↩ SE,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_BU↩ SY,Slave	Ongoing header reception.
FLEXIO_LIN_IP_STATUS_RX_HEADER_ERR↩ OR,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_OPERATION↩ AL,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.

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.

Parameters

channel	LIN_FLEXIO channel number
PduInfo	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

FLEXIO_LIN_IP_STATUS_SUCCESS	: Command has not been accepted .
FLEXIO_LIN_IP_STATUS_BUSY	: Driver is busy with another transfer.
FLEXIO_LIN_IP_STATUS_ERROR	: 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 LinLpuartStartTimerNotification. In timer IRQ handler, call this function.

6.2 LIN Driver

6.2.1 Detailed Description

Data Structures

- struct Lin_ChannelConfigType

  LIN channel configuration type structure. *More...*
- struct Lin_ConfigType

  LIN driver configuration type structure. *More...*

Macros

- #define LIN_UNINIT

  LIN driver states.
- #define LIN_INIT

  LIN driver states.
- #define LIN_CH_SLEEP_PENDING

  LIN Channel states.
- #define LIN_CH_SLEEP_STATE

  LIN Channel states.
- #define LIN_CH_OPERATIONAL

  LIN Channel states.
- #define LIN_GETSTATUS_ID

  API service ID for Lin_GetStatus() function.
- #define LIN_GETVERSIONINFO_ID

  API service ID for Lin_GetVersionInfo() function.
- #define LIN_GOTOSLEEP_ID

  API service ID for Lin_GoToSleep() function.
- #define LIN_GOTOSLEEPINTERNAL_ID

  API service ID for Lin_GoToSleepInternal() function.
- #define LIN_INIT_ID

  API service ID for Lin_Init() function.
- #define LIN_SENDFRAME_ID

  API service ID for Lin_SendFrame() function.
- #define LIN_WAKEUP_ID

  API service ID for Lin_WakeUp() function.
- #define LIN_WAKEUPINTERNAL_ID

  API service ID for Lin_WakeupInternal() function.
- #define LIN_CHECKWAKEUP_ID

  API service ID for Lin_CheckWakeup() function.
- #define LIN_TIMEOUT_ERROR

  Return code for timeout error.

Function Reference

- Std_ReturnType Lin_CheckWakeup (uint8 Channel)
  Validates for upper layers the wake up of LIN channel.
- void Lin_Init (const Lin_ConfigType *Config)
  Initializes the LIN module.
- Std_ReturnType Lin_GoToSleepInternal (uint8 Channel)
  Same function as Lin_Ipw_GoToSleep() but without sending a go-to-sleep-command on the bus.
- Std_ReturnType Lin_Wakeup (uint8 Channel)
  Generates a wake up pulse.
- Std_ReturnType Lin_WakeupInternal (uint8 Channel)
  Wake up the LIN channel.

6.2.2 Data Structure Documentation

6.2.2.1 struct Lin_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_Types.h.

Data Fields

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

6.2.2.2 struct Lin_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_Types.h.

Data Fields

- const Lin_ChannelConfigType * Lin_ChannelPtr [LIN_HW_MAX_MODULES]
  Partition core id is assigned for this configuration.

6.2.2.2.1 Field Documentation

6.2.2.2.1.1 Lin_ChannelPtr `const Lin_ChannelConfigType* Lin_ChannelPtr[LIN_HW_MAX_MODULES]`

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_Types.h.

6.2.3 Macro Definition Documentation

6.2.3.1 LIN_UNINIT

```
#define LIN_UNINIT
```

LIN driver states.

The state LIN_UNINIT means that the Lin module has not been initialized yet and cannot be used.

Definition at line 229 of file Lin.h.

6.2.3.2 LIN_INIT

```
#define LIN_INIT
```

LIN driver states.

The LIN_INIT state indicates that the LIN driver has been initialized, making each available channel ready for service.

Definition at line 238 of file Lin.h.

6.2.3.3 LIN_CH_SLEEP_PENDING

```
#define LIN_CH_SLEEP_PENDING
```

LIN Channel states.

go-to-sleep-command has been issued on the bus, LIN channel stay at this state until Lin_GetStatus() is called

Definition at line 247 of file Lin.h.

6.2.3.4 LIN_CH_SLEEP_STATE

```
#define LIN_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 257 of file Lin.h.

6.2.3.5 LIN_CH_OPERATIONAL

```
#define LIN_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 267 of file Lin.h.

6.2.3.6 LIN_GETSTATUS_ID

```
#define LIN_GETSTATUS_ID
```

API service ID for Lin_GetStatus() function.

Parameters used when raising an error or exception.

Definition at line 277 of file Lin.h.

6.2.3.7 LIN_GETVERSIONINFO_ID

```
#define LIN_GETVERSIONINFO_ID
```

API service ID for Lin_GetVersionInfo() function.

Parameters used when raising an error or exception.

Definition at line 285 of file Lin.h.

6.2.3.8 LIN_GOTOSLEEP_ID

```
#define LIN_GOTOSLEEP_ID
```

API service ID for Lin_GoToSleep() function.

Parameters used when raising an error or exception.

Definition at line 293 of file Lin.h.

6.2.3.9 LIN_GOTOSLEEPINTERNAL_ID

```
#define LIN_GOTOSLEEPINTERNAL_ID
```

API service ID for Lin_GoToSleepInternal() function.

Parameters used when raising an error or exception.

Definition at line 301 of file Lin.h.

6.2.3.10 LIN_INIT_ID

```
#define LIN_INIT_ID
```

API service ID for Lin_Init() function.

Parameters used when raising an error or exception.

Definition at line 309 of file Lin.h.

6.2.3.11 LIN_SENDFRAME_ID

```
#define LIN_SENDFRAME_ID
```

API service ID for Lin_SendFrame() function.

Parameters used when raising an error or exception.

Definition at line 317 of file Lin.h.

6.2.3.12 LIN_WAKEUP_ID

```
#define LIN_WAKEUP_ID
```

API service ID for Lin_WakeUp() function.

Parameters used when raising an error or exception.

Definition at line 325 of file Lin.h.

6.2.3.13 LIN_WAKEUPINTERNAL_ID

```
#define LIN_WAKEUPINTERNAL_ID
```

API service ID for Lin_WakeupInternal() function.

Parameters used when raising an error or exception.

Definition at line 333 of file Lin.h.

6.2.3.14 LIN_CHECKWAKEUP_ID

```
#define LIN_CHECKWAKEUP_ID
```

API service ID for Lin_CheckWakeup() function.

Parameters used when raising an error or exception.

Definition at line 341 of file Lin.h.

6.2.3.15 LIN_TIMEOUT_ERROR

```
#define LIN_TIMEOUT_ERROR
```

Return code for timeout error.

Definition at line 120 of file Lin_Types.h.

6.2.4 Function Reference

6.2.4.1 Lin_CheckWakeup()

```
Std_ReturnType Lin_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	*Channel*	LIN channel to be addressed.

Return values

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

6.2.4.2 Lin_Init()

```
void Lin_Init (
            const Lin_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_CH_SLEEP_STATE.
- Set driver state machine to LIN_INIT.

Parameters

in	*Config*	- Pointer to LIN driver configuration set.

Returns

void

Precondition

-

6.2.4.3 Lin_GoToSleepInternal()

```
Std_ReturnType Lin_GoToSleepInternal (
            uint8 Channel )
```

Same function as Lin_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	*Channel*	LIN channel to be addressed.

Returns

Std_ReturnType.

Return values

E_NOT_OK	In case of a timeout situation only.
E_OK	Otherwise.

Precondition

: Lin_Init function must be called before this API.

6.2.4.4 Lin_Wakeup()

```
Std_ReturnType Lin_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_CH_OPERATIONAL state.

Parameters

in	*Channel*	LIN channel to be addressed.

Returns

Std_ReturnType.

Return values

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

Precondition

: Lin_Init function must be called before this API.

6.2.4.5 Lin_WakeupInternal()

```
Std_ReturnType Lin_WakeupInternal (
            uint8 Channel )
```

Wake up the LIN channel.

This function shall put the LIN channel in LIN_CH_OPERATIONAL state without sending a wake up signal to the LIN bus

Parameters

in	*Channel*	LIN channel to be addressed.

Returns

Std_ReturnType.

Return values

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

Precondition

: Lin_Init function must be called before this API.

6.3 LIN

6.3.1 Detailed Description

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_UserConfigType

  User configuration structure of the LIN driver. More...

Macros

- #define LPUART_LIN_IP_SLAVE

  Macro that specifies usage of a SLAVE note.
- #define LPUART_LIN_IP_MASTER

  Macro that specifies usage of a SLAVE note.
- #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, uint8 **Buffer)

  Returns the status of an on going transmission.
- void Lpuart_Lin_Ip_TimerExpiredService (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.
- void Lpuart_Lin_Ip_IRQHandler (const uint8 Instance)

  LIN_LPUART interrupt handler for RX_TX and Error interrupts.

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_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 [450us->5ms].
- uint8 BaseWakeupByteDetectInverted

  Byte use to check detection of wake up pulse longer than 150us.
- uint32 ChannelClock

  Channel clock.

6.4.2.2.1 Field Documentation

6.4.2.2.1.1 Instance `uint8 Instance`

Hardware Instance.

Definition at line 302 of file Lpuart_Lin_Ip_Types.h.

6.4.2.2.1.2 BaudRateDivisor `uint32 BaudRateDivisor`

Baudrate divider to be configured in hardware.

Definition at line 303 of file Lpuart_Lin_Ip_Types.h.

6.4.2.2.1.3 OverSamplingRatio `uint32 OverSamplingRatio`

baudrate of LIN Hardware Interface to configure

Definition at line 304 of file Lpuart_Lin_Ip_Types.h.

6.4.2.2.1.4 NodeFunction `boolean NodeFunction`

Node function as Master or Slave.

Definition at line 305 of file Lpuart_Lin_Ip_Types.h.

6.4.2.2.1.5 BreakLength `uint8 BreakLength`

Length of break character will be transmitted.

Definition at line 306 of file Lpuart_Lin_Ip_Types.h.

6.4.2.2.1.6 BreakLengthDetect `uint8 BreakLengthDetect`

Length of break character can be detected.

Definition at line 307 of file Lpuart_Lin_Ip_Types.h.

6.4.2.2.1.7 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.

6.4.2.2.1.8 WakeupByte `uint8 WakeupByte`

Byte will be sent to generate wakeup pulse [450us->5ms].

Definition at line 319 of file Lpuart_Lin_Ip_Types.h.

6.4.2.2.1.9 BaseWakeupByteDetectInverted `uint8 BaseWakeupByteDetectInverted`

Byte use to check detection of wake up pulse longer than 150us.

Definition at line 320 of file Lpuart_Lin_Ip_Types.h.

6.4.2.2.1.10 ChannelClock `uint32 ChannelClock`

Channel clock.

Definition at line 321 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 note.

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 note.

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 ok.
LPUART_LIN_IP_SEND_HEADER_OK	Sync byte received with errors.
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

LPUART_LIN_IP_ENHANCED_CS	Enhanced CheckSum model.
LPUART_LIN_IP_CLASSIC_CS	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

LPUART_LIN_IP_FRAMERESPONSE_TX	Response is generated from this (master) node.
LPUART_LIN_IP_FRAMERESPONSE_RX	Response is generated from a remote slave node.
LPUART_LIN_IP_FRAMERESPONSE_IGNORE	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

LPUART_LIN_IP_STATUS_FAIL	Command has not been accepted.
LPUART_LIN_IP_STATUS_TX_OK	Transmission successfully.
LPUART_LIN_IP_STATUS_TX_BUSY	Transmission is ongoing.
LPUART_LIN_IP_STATUS_TX_HEADER_E↩ RROR	Erroneous header transmission.
LPUART_LIN_IP_STATUS_TX_ERROR	Erroneous response transmission.
LPUART_LIN_IP_STATUS_RX_OK	Reception of correct response.
LPUART_LIN_IP_STATUS_RX_BUSY	Ongoing reception. At least one response byte has been received.
LPUART_LIN_IP_STATUS_RX_ERROR	Erroneous response reception.

Enumerator

LPUART_LIN_IP_STATUS_RX_NO_RESPONSE	No response byte has been received so far(timeout occurred).
LPUART_LIN_IP_STATUS_RX_HEADER_OK	Slave received a correct header.
LPUART_LIN_IP_STATUS_RX_HEADER_BUSY	Slave is receiving header.
LPUART_LIN_IP_STATUS_RX_HEADER_ERROR	Erroneous header reception of slave.
LPUART_LIN_IP_STATUS_OPERATIONAL	Normal operation.
LPUART_LIN_IP_STATUS_SLEEP	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

Instance	LIN_LPUART instance number
UserConfig	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

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

Returns

operation status:

Return values

LPUART_LIN_IP_STATUS_SUCCESS	- Send command has been accepted.
LPUART_LIN_IP_STATUS_BUSY	- Operation failed due to hardware instance busy.
LPUART_LIN_IP_STATUS_ERROR	- 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

Instance	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

Instance	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

Instance	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

Instance	LIN_LPUART instance number

Returns

operation status:

Return values

LPUART_LIN_IP_STATUS_SUCCESS	: Command has been accepted.
LPUART_LIN_IP_STATUS_ERROR	: 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,
            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

channel	LIN_LPUART instance number.
Buffer	Pointer to the received data buffer.

Returns

operation status: LPUART_Lin_Ip_TransferStatusType

Return values

LPUART_LIN_IP_STATUS_FAIL	Command has not been accepted .
LPUART_LIN_IP_STATUS_TX_OK,Master	Header or Response successful transmission. Slave: Response successful transmission.
LPUART_LIN_IP_STATUS_TX_BUSY,Master	Ongoing transmission (Header or Response). Slave: Ongoing transmission response.
LPUART_LIN_IP_STATUS_TX_HEADER_ER↩ ROR,Master	Erroneous header transmission such as: Mismatch between sent and read back data or Physical bus error./
LPUART_LIN_IP_STATUS_TX_ERROR,Master	or Slave: Erroneous response transmission such as mismatch between sent and read back data, Physical bus error
LPUART_LIN_IP_STATUS_RX_OK,Master	or Slave: Reception of correct response.
LPUART_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.
LPUART_LIN_IP_STATUS_RX_ERROR,Master	or Slave: Erroneous response reception such as: - Framing error - Overrun error - Checksum error or Short response(timeout occurred).
LPUART_LIN_IP_STATUS_RX_NO_RESPON↩ SE,Master	or Slave: No response byte has been received so far(timeout occurred).
LPUART_LIN_IP_STATUS_RX_HEADER_↩ OK,Slave	Reception of correct header.
LPUART_LIN_IP_STATUS_RX_HEADER_BU↩ SY,Slave	Ongoing header reception.
LPUART_LIN_IP_STATUS_RX_HEADER_ER↩ ROR,Slave	Erroneous header reception such as: Break field error, Sync field error, Identifier parity error, Framing error, timeout occurred or Physical bus error.
LPUART_LIN_IP_STATUS_OPERATION↩ AL,Normal	operation; there is no data has been sent or received after channel initialized or switched to IDLE from SLEEP.
LPUART_LIN_IP_STATUS_SLEEP	Sleep state operation.

6.4.6.9 Lpuart_Lin_Ip_TimerExpiredService()

```
void Lpuart_Lin_Ip_TimerExpiredService (
            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

Instance	LIN_LPUART instance number

Returns

None

6.4.6.10 Lpuart_Lin_Ip_IRQHandler()

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

LIN_LPUART interrupt handler for RX_TX and Error interrupts.

Parameters

Instance	LIN_LPUART instance number

Returns

void

How to Reach Us:

Home Page:
nxp.com

Web Support:
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.