User Manual

for S32K3 ZIPWIRE Driver

Document Number: UM34ZIPWIREASRR21-11 Rev0000R3.0.0 Rev. 1.0

Chapter 1

Revision History

Revision	Date	Author	Description
1.0	31.03.2023	NXP RTD Team	Prepared for release RTD S32K3 3.0.0

Chapter 2

Introduction

This User Manual describes NXP Semiconductor AUTOSAR ZIPWIRE for S32K3 microontrollers. AUTOSAR ZI↩PWIRE driver configuration parameters and deviations from the specification are described in Driver chapter of this document. AUTOSAR ZIPWIRE driver requirements and APIs are described in the AUTOSAR ZIPWIRE driver software specification document.

2.1 Supported Derivatives

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

- s32k310_mqfp100
- s32k310_lqfp48
- s32k311_mqfp100 / MWCT2015S_mqfp100
- s32k311_lqfp48
- s32k312_mqfp100 / MWCT2016S_mqfp100
- s32k312_mqfp172 / MWCT2016S_mqfp172
- s32k314_mqfp172
- s32k314_mapbga257
- s32k322_mqfp100 / MWCT2D16S_mqfp100
- s32k322_mqfp172 / MWCT2D16S_mqfp172

- s32k324_mqfp172 / MWCT2D17S_mqfp172
- s32k324_mapbga257
- s32k341_mqfp100
- s32k341_mqfp172
- s32k342_mqfp100
- s32k342_mqfp172
- s32k344_mqfp172
- s32k344_mapbga257
- s32k394_mapbga289
- s32k396_mapbga289
- s32k358_mqfp172
- s32k358_mapbga289
- s32k328_mqfp172
- s32k328_mapbga289
- s32k338_mqfp172
- s32k338_mapbga289
- s32k348_mqfp172
- s32k348_mapbga289
- s32m274_lqfp64
- s32m276_lqfp64 All of the above microcontroller devices are collectively named as S32K3. Note: MWCT part numbers contain NXP confidential IP for Qi Wireless Power.

2.2 Overview

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

AUTOSAR:

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

2.3 About This Manual

This Technical Reference employs the following typographical conventions:

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

Notes and warnings are shown as below:

Note

This is a note.

Warning

This is a warning

2.4 Acronyms and Definitions

Term	Definition
API	Application Programming Interface
ASM	Assembler
BSMI	Basic Software Make file Interface
CAN	Controller Area Network
C/CPP	C and C++ Source Code
CS	Chip Select
CTU	Cross Trigger Unit
DEM	Diagnostic Event Manager
DET	Development Error Tracer
DMA	Direct Memory Access
ECU	Electronic Control Unit
FIFO	First In First Out
LSB	Least Signifigant Bit
MCU	Micro Controller Unit
MIDE	Multi Integrated Development Environment
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	S32K3XX Reference Manual	Rev.6, Draft B, 01/2023
2	S32K396 Reference Manual	Rev. 2 Draft A, 11/2022
3	S32K396 Data Sheet	Rev. 1.1 — 08/2022
4	S32K396_0P40E Mask Set Errata	Rev. DEC2022, 12/2022

Chapter 3

Driver

3.1 Requirements

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

For CDD: Zipwire is a Complex Device Driver (CDD).

It has vendor-specific requirements and implementation.

3.2 Driver Design Summary

- The ZIPWIRE driver is implemented as an complex device driver. It uses the LFAST and SPIP hardware peripherals which provides support for implementing the ZIPWIRE transfers.
- The driver offers: Read, Write ReadBlocking, WriteBlocking, ReadDma, WriteDma, ReadDmaBlocking, WriteDmaBlocking and StreamWrite.
- Hardware and software settings can be configured using an Autosar standard configuration tool. The information required for a ZIPWIRE data transfers will be configured in a data structure that will be sent as parameter to the API of the driver.

3.3 Hardware Resources

The ZIPWIRE Driver consists of:

1. ZIPWIRE IP

3.4 Deviations from Requirements

None.

3.5 Driver Limitations

The ZIPWIRE Driver has the following limitations:
- Post Build Compiler not implemented.

3.6 Driver usage and configuration tips

This driver is an Complex Device Driver. Complete driver functionality together with API description can be found below.

3.6.1 ZIPWIRE Calculation Type

- Autosar Library supports only the Autosar Protocols.
- Polynomial configuration is enabled when the following Protocols are selected.
- Note: Each Calculation Type supports a specific list of Protocols. If out of range there will be an error message.

3.6.2 ZIPWIRE Initialization.

- The Zipwire_Init() function shall initialize the ZIPWIRE hardware peripheral(s) and the internal driver context, according to the input configuration data. The application shall ensure that the Zipwire_Init() function is called first. Only the Zipwire_GetVersionInfo() can be called before Zipwire_Init().

3.6.3 ZIPWIRE DeInitialization.

- The Zipwire_DeInit() function shall de-initialize the ZIPWIRE hardware peripheral(s) and the internal driver context, according to the input configuration data.

3.6.4 ZIPWIRE Instance Configuration.

- The function receives pointer to a configuration structure that shall be loaded into the Logic Instance. According to input parameters this function configures channel's ZIPWIRE Error Callback Function, Timeout Prescaler, Address Offset, LFAST Role, LFAST Speed Mode, LFAST Low Speed Clock Division, LFAST Syncronisation Attempts and LFAST Timeout.

3.6.5 ZIPWIRE Channel Configuration.

- The function receives pointer to a configuration structure that shall be loaded into the Logic Channel. According to input parameters this function configures channel's ZIPWIRE Interrupts Enable (Timeout, Acknowledge and Transfer Id), Dma Enable and the links to the DMA channels.

3.7 Runtime errors

The driver generates the following DET errors at runtime.

Function	Error Code	Condition triggering the error
Zipwire_Init()	ZIPWIRE_E_INIT_FAILED	API is called with a NULL pointer as parameter.
Zipwire_Init()	ZIPWIRE_E_ALREADY_INIT↩IALIZED	API is called while the driver was already initialized.
Zipwire_DeInit()	ZIPWIRE_E_DEINIT_FAILED	API is called with a NULL pointer as parameter.
Zipwire_Read()	ZIPWIRE_E_UNINIT	API is called before the Init function is called or after the DeInit function is called
Zipwire_ReadBlocking()	ZIPWIRE_E_UNINIT	API is called before the Init function is called or after the DeInit function is called
Zipwire_ReadDma()	ZIPWIRE_E_UNINIT	API is called before the Init function is called or after the DeInit function is called
Zipwire_ReadDmaBlocking()	ZIPWIRE_E_UNINIT	API is called before the Init function is called or after the DeInit function is called
Zipwire_Write()	ZIPWIRE_E_UNINIT	API is called before the Init function is called or after the DeInit function is called
Zipwire_WriteBlocking()	ZIPWIRE_E_UNINIT	API is called before the Init function is called or after the DeInit function is called
Zipwire_WriteDma()	ZIPWIRE_E_UNINIT	API is called before the Init function is called or after the DeInit function is called
Zipwire_WriteDmaBlocking()	ZIPWIRE_E_UNINIT	API is called before the Init function is called or after the DeInit function is called

Function	Error Code	Condition triggering the error
Zipwire_StreamWrite()	ZIPWIRE_E_UNINIT	API is called before the Init function is called or after the DeInit function is called
Zipwire_RequestId()	ZIPWIRE_E_UNINIT	API is called before the Init function is called or after the DeInit function is called
Zipwire_Trigger()	ZIPWIRE_E_UNINIT	API is called before the Init function is called or after the DeInit function is called
Zipwire_GetChannelStatus()	ZIPWIRE_E_UNINIT	API is called before the Init function is called or after the DeInit function is called
Zipwire_InstallGlobalCallback()	ZIPWIRE_E_UNINIT	API is called before the Init function is called or after the DeInit function is called
Zipwire_InstallChannelCallback()	ZIPWIRE_E_UNINIT	API is called before the Init function is called or after the DeInit function is called

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 Zipwire
  - Container ZipwireGeneral
    - Parameter ZipwireDetectError
    - Parameter ZipwireEnableUserModeSupport
    - Parameter ZipwireDmaSupportEnable
    - Parameter ZipwireVersionInfoApi
    - Parameter ZipwireTimeoutMethod
    - Parameter ZipwireTimeoutDuration
  - Container ZipwireInstanceConfig
    - Parameter ZipwireLogicInstanceName
    - Parameter ZipwireHwInstance
    - Parameter ZipwireErrorInterruptCallback
    - Parameter ZipwireEnableMaxCountReachedIrq
    - Parameter ZipwireTimeoutPrescaler
    - Parameter ZipwireAddressOffset
    - Container ZipwireLfastConfig
      - Parameter ZipwireLfastRole
      - Parameter ZipwireLfastSpeedMode
      - Parameter ZipwireLfastLowSpeedClock
      - Parameter ZipwireLfastSyncAttempts
      - Parameter ZipwireLfastSyncTimeout
  - Container ZipwireChannelConfig
    - Parameter ZipwireLogicChannelName
    - Parameter ZipwireHwInstance
    - Parameter ZipwireHwChannel
    - Parameter ZipwireErrorInterruptCallback
    - Parameter ZipwireTimeout
    - Parameter ZipwireDmaChannelEnable
    - Parameter ZipwireEnableTimeoutErrIrq
    - Parameter ZipwireEnableAckErrIrq

        * Parameter ZipwireEnableTransferIdErrIrq
        * Container ZipwireDmaConfig
            · Reference ZipwireDataDmaLogicChannelName
            · Reference ZipwireAddressDmaLogicChannelName
    - Container CommonPublishedInformation
        * Parameter ArReleaseMajorVersion
        * Parameter ArReleaseMinorVersion
        * Parameter ArReleaseRevisionVersion
        * Parameter ModuleId
        * Parameter SwMajorVersion
        * Parameter SwMinorVersion
        * Parameter SwPatchVersion
        * Parameter VendorApiInfix
        * Parameter VendorId

4.1 Module Zipwire

Vendor specific: Configuration of the Zipwire module.

Included containers:

- ZipwireGeneral
- ZipwireInstanceConfig
- ZipwireChannelConfig
- CommonPublishedInformation

Property	Value
type	ECUC-MODULE-DEF
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantSupport	false
supportedConfigVariants	VARIANT-PRE-COMPILE

4.2 Container ZipwireGeneral

Zipwire General

All general parameters of the Zipwire driver are collected here.

Included subcontainers:

- None

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

4.3 Parameter ZipwireDetectError

ZIPWIRE Development Error Detect

Compile switch to enable/disable development error detection for this module.

Unchecked: Zipwire Development error detection disabled

Checked : Zipwire Development error detection enabled

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-PRE-COMPILE: PRE-COMPILE
defaultValue	true

4.4 Parameter ZipwireEnableUserModeSupport

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-PRE-COMPILE: PRE-COMPILE
defaultValue	false

4.5 Parameter ZipwireDmaSupportEnable

Dma Read/Write Support

Check this in order to be able to use DMA in the Zipwire driver. Leaving this unchecked will allow the Zipwire driver to compile with no dependencies from the Mcl driver.

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-PRE-COMPILE: PRE-COMPILE
defaultValue	false

4.6 Parameter ZipwireVersionInfoApi

Zipwire VersionInfo Api

Compile switch to enable/disable the version information API.

Checked : API enabled

Unchecked: API disabled

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-PRE-COMPILE: PRE-COMPILE
defaultValue	false

4.7 Parameter ZipwireTimeoutMethod

ZipwireTimeoutMethod

Configures the timeout method.

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

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-PRE-COMPILE: POST-BUILD
defaultValue	OSIF_COUNTER_DUMMY
literals	['OSIF_COUNTER_SYSTEM', 'OSIF_COUNTER_CUSTOM', 'OSIF_CO↩ UNTER_DUMMY']

4.8 Parameter ZipwireTimeoutDuration

The unit of measurement is given in number of microseconds. This is a timeout value which is used to wait till the blocking operation is finished

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: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	800
max	65535
min	1

4.9 Container ZipwireInstanceConfig

Zipwire Instance Configuration

Configuration of an individual Zipwire(SIPI and LFAST) Instance. Symbolic names will be generated for each instance.

Included subcontainers:

- ZipwireLfastConfig

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

4.10 Parameter ZipwireLogicInstanceName

Logic Instance Name

Instance used for SIPI and LFAST operations

Property	Value
type	ECUC-STRING-PARAM-DEF
origin	NXP
symbolicNameValue	False
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	ZIPWIRE_LOGIC_INSTANCE_0

4.11 Parameter ZipwireHwInstance

Hardware Instance

Identifies the Zipwire Hardware Instance for LFAST and SIPI .

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-PRE-COMPILE: PRE-COMPILE
defaultValue	ZIPWIRE_LOGIC_INSTANCE_0
literals	['ZIPWIRE_LOGIC_INSTANCE_0']

4.12 Parameter ZipwireErrorInterruptCallback

Vendor specific:

User callback function

NOTE: Use NULL_PTR w/o quotes. If the used string is different from NULL_PTR it will be used as the configured function name.

Property	Value
type	ECUC-FUNCTION-NAME-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	NULL_PTR

4.13 Parameter ZipwireEnableMaxCountReachedIrq

Max Count Reached Interrupt Enable

Checked : Enabled

Unchecked: Disabled

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-PRE-COMPILE: PRE-COMPILE
defaultValue	false

4.14 Parameter ZipwireTimeoutPrescaler

Configures the timeout clock prescaler value.

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false

Property	Value
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	ZIPWIRE_DIV_64
literals	['ZIPWIRE_DIV_64', 'ZIPWIRE_DIV_128', 'ZIPWIRE_DIV_256', 'ZIPWI↩ RE_DIV_512', 'ZIPWIRE_DIV_1024']

4.15 Parameter ZipwireAddressOffset

Configures address increment/decrement field.

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	ZIPWIRE_ADDR_NO_CHANGE
literals	['ZIPWIRE_ADDR_NO_CHANGE', 'ZIPWIRE_ADDR_INC_4', 'ZIPWIR↩ E_ADDR_DEC_4']

4.16 Container ZipwireLfastConfig

This container contains the hardware configuration parameters of the Zipwire LFAST module.

Included subcontainers:

- None

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF
lowerMultiplicity	1

Property	Value
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A

4.17 Parameter ZipwireLfastRole

Configures the LFAST role: MASTER/SLAVE.

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

4.18 Parameter ZipwireLfastSpeedMode

Configures the LFAST speed mode: high-speed/low-speed.

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

4.19 Parameter ZipwireLfastLowSpeedClock

Configures the LFAST clock division factor in Low Speed Select mode.

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	LFAST_LOW_SPEED_CLK_DIV_2
literals	['LFAST_LOW_SPEED_CLK_DIV_2', 'LFAST_LOW_SPEED_CLK_DI↩ V_4']

4.20 Parameter ZipwireLfastSyncAttempts

LFAST Syncronisation Attempts

Number of attempts for the master to synchronize with the slave;

this field is only used by the master node.

A value of zero for this parameter is equivalent to an infinite

number of attempts; the LFAST master will try forever to synchronize

with the slave

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-PRE-COMPILE: PRE-COMPILE
defaultValue	0
max	255
min	0

4.21 Parameter ZipwireLfastSyncTimeout

LFAST Timeout

Timeout used for the LFAST master-slave synchronization;

A value of zero for this parameter is equivalent to timeout

being disregarded by the driver; the LFAST initialization

will wait forever for commands/responses from the other node.

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-PRE-COMPILE: PRE-COMPILE
defaultValue	0
max	255
min	0

4.22 Container ZipwireChannelConfig

Zipwire Channels Configuration

Configuration of an individual Zipwire(SIPI) channel. Symbolic names will be generated for each channel.

Included subcontainers:

- ZipwireDmaConfig

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

4.23 Parameter ZipwireLogicChannelName

Logic Channel Name

Channel used for SIPI operations

Property	Value
type	ECUC-STRING-PARAM-DEF
origin	NXP
symbolicNameValue	False
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	ZIPWIRE_LOGIC_CHANNEL_0

4.24 Parameter ZipwireHwInstance

Hardware Instance

Identifies the Zipwire Hardware Instance for LFAST and SIPI .

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-PRE-COMPILE: PRE-COMPILE
defaultValue	ZIPWIRE_LOGIC_INSTANCE_0
literals	['ZIPWIRE_LOGIC_INSTANCE_0']

4.25 Parameter ZipwireHwChannel

Hardware Channel

Selects one of the Zipwire hardware channels available on the device.

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
defaultValue	ZIPWIRE_LOGIC_CHANNEL_0
literals	['ZIPWIRE_LOGIC_CHANNEL_0', 'ZIPWIRE_LOGIC_CHANNEL_1', 'Z↩ IPWIRE_LOGIC_CHANNEL_2', 'ZIPWIRE_LOGIC_CHANNEL_3']

4.26 Parameter ZipwireErrorInterruptCallback

Vendor specific:

User callback function

NOTE: Use NULL_PTR w/o quotes. If the used string is different from NULL_PTR it will be used as the configured function name.

Property	Value
type	ECUC-FUNCTION-NAME-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	NULL_PTR

4.27 Parameter ZipwireTimeout

Timeout

Timeout value for requests

A value of zero for this parameter is equivalent to timeout

being disregarded by the driver

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-PRE-COMPILE: PRE-COMPILE
defaultValue	0
max	255
min	0

4.28 Parameter ZipwireDmaChannelEnable

Dma Channel Enable

Checked : Enabled

Unchecked: Disabled

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-PRE-COMPILE: PRE-COMPILE
defaultValue	false

4.29 Parameter ZipwireEnableTimeoutErrIrq

Timeout Error Interrupt Enable

Checked : Enabled

Unchecked: Disabled

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-PRE-COMPILE: PRE-COMPILE
defaultValue	false

4.30 Parameter ZipwireEnableAckErrIrq

Acknowledge Error Interrupt Enable

Checked : Enabled

Unchecked: Disabled

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-PRE-COMPILE: PRE-COMPILE
defaultValue	false

4.31 Parameter ZipwireEnableTransferIdErrIrq

Transfer Id Error Interrupt Enable

Checked : Enabled

Unchecked: Disabled

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-PRE-COMPILE: PRE-COMPILE
defaultValue	false

4.32 Container ZipwireDmaConfig

This container contains the hardware configuration parameters of the Zipwire module.

Included subcontainers:

- None

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

4.33 Reference ZipwireDataDmaLogicChannelName

DMA Logic Channel Name

DMA Logic Channel is used to transfer data.

Property	Value
type	ECUC-REFERENCE-DEF
origin	NXP
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
requiresSymbolicNameValue	False
destination	/AUTOSAR/EcucDefs/Mcl/MclConfig/dmaLogicChannel_Type

4.34 Reference ZipwireAddressDmaLogicChannelName

DMA Logic Channel Name

DMA Logic Channel is used to transfer addresses.

Property	Value
type	ECUC-REFERENCE-DEF
origin	NXP
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
requiresSymbolicNameValue	False
destination	/AUTOSAR/EcucDefs/Mcl/MclConfig/dmaLogicChannel_Type

4.35 Container CommonPublishedInformation

Common Published Information

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

4.36 Parameter ArReleaseMajorVersion

AUTOSAR Release Major Version

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-PRE-COMPILE: PUBLISHED-INFORMATION
defaultValue	4
max	4
min	4

4.37 Parameter ArReleaseMinorVersion

AUTOSAR Release Minor Version

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

Property	Value
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PUBLISHED-INFORMATION
defaultValue	7
max	7
min	7

4.38 Parameter ArReleaseRevisionVersion

AUTOSAR Release Revision Version

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-PRE-COMPILE: PUBLISHED-INFORMATION
defaultValue	0
max	0
min	0

4.39 Parameter ModuleId

Module ID

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

Property	Value
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PUBLISHED-INFORMATION
defaultValue	255
max	255
min	255

4.40 Parameter SwMajorVersion

Software Major Version

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-PRE-COMPILE: PUBLISHED-INFORMATION
defaultValue	3
max	3
min	3

4.41 Parameter SwMinorVersion

Software Minor Version

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

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

4.42 Parameter SwPatchVersion

Software Patch Version

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-PRE-COMPILE: PUBLISHED-INFORMATION
defaultValue	0
max	0
min	0

4.43 Parameter VendorApiInfix

Vendor Api Infix

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-PRE-COMPILE: PUBLISHED-INFORMATION
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PUBLISHED-INFORMATION
defaultValue	

4.44 Parameter VendorId

Vendor Id

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-PRE-COMPILE: PUBLISHED-INFORMATION
defaultValue	43
max	43
min	43

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

Chapter 5

Module Index

5.1 Software Specification

Here is a list of all modules:

Chapter 6

Module Documentation

6.1 ZIPWIRE HLD Driver

6.1.1 Detailed Description

Macros

- #define CDD_ZIPWIRE_MODULE_ID
  Parameters that shall be published within the Zipwire driver header file and also in the module's description file.
- #define ZIPWIRE_INSTANCE_ID
  ID of ZIPWIRE Instance.
- #define ZIPWIRE_INIT_ID
  API service ID for Zipwire_Init function.
- #define ZIPWIRE_DEINIT_ID
  API service ID for Zipwire_DeInit function.
- #define ZIPWIRE_READ_ID
  API service ID for Zipwire_Read function.
- #define ZIPWIRE_READBLOCKING_ID
  API service ID for Zipwire_ReadBlocking function.
- #define ZIPWIRE_READDMA_ID
  API service ID for Zipwire_ReadDma function.
- #define ZIPWIRE_READDMABLOCKING_ID
  API service ID for Zipwire_ReadDmaBlocking function.
- #define ZIPWIRE_WRITE_ID
  API service ID for Zipwire_Write function.
- #define ZIPWIRE_WRITEBLOCKING_ID
  API service ID for Zipwire_WriteBlocking function.
- #define ZIPWIRE_WRITEDMA_ID
  API service ID for Zipwire_WriteDma function.
- #define ZIPWIRE_WRITEDMABLOCKING_ID
  API service ID for Zipwire_WriteDmaBlocking function.
- #define ZIPWIRE_STREAMWRITE_ID

API service ID for Zipwire_StreamWrite function.
- #define ZIPWIRE_REQUESTID_ID

  API service ID for Zipwire_RequestId function.
- #define ZIPWIRE_TRIGGER_ID

  API service ID for Zipwire_Triger function.
- #define ZIPWIRE_GETCHANNELSTATUS_ID

  API service ID for Zipwire_GetChannelStatus function.
- #define ZIPWIRE_INSTALLGLOBALCALLBACK_ID

  API service ID for Zipwire_InstallGlobalCallback function.
- #define ZIPWIRE_INSTALLCHANNELCALLBACK_ID

  API service ID for Zipwire_InstallChannelCallback function.
- #define ZIPWIRE_E_INIT_FAILED

  The ZIPWIRE module is not properly initialized.
- #define ZIPWIRE_E_DEINIT_FAILED

  The ZIPWIRE module is not properly deinitialized.
- #define ZIPWIRE_E_ALREADY_INITIALIZED

  The ZIPWIRE module is already initialized.
- #define ZIPWIRE_E_UNINIT

  The ZIPWIRE module is was never initialized.
- #define ZIPWIRE_TYPES_VENDOR_ID

  Parameters that shall be published within the Zipwire driver header file and also in the module's description file.

Types Reference

- typedef Zipwire_Ip_ConfigType Zipwire_InitType

  This type contains the Zipwire Initialization.

Enum Reference

- enum Zipwire_StatusType

  This type contains the Zipwire driver states.

Function Reference

- void Zipwire_Init (const Zipwire_InitType ∗ConfigPtr)

  This service will store the Zipwire driver installation configuration based on user configuration.
- void Zipwire_DeInit (const Zipwire_InitType ∗ConfigPtr)

  This service will store the Zipwire driver installation configuration based on user configuration.
- Zipwire_Ip_StatusType Zipwire_Read (uint8 HwInstance, uint8 HwChannel, Zipwire_Ip_TransferDescriptor ∗DataArray, uint32 DataArrayLength)

  Performs multiple read transfers.
- Zipwire_Ip_StatusType Zipwire_ReadBlocking (uint8 HwInstance, uint8 HwChannel, Zipwire_Ip_TransferDescriptor ∗DataArray, uint32 DataArrayLength)

  Performs multiple read transfers synchronously.

- Zipwire_Ip_StatusType Zipwire_ReadDma (uint8 HwInstance, uint8 HwChannel, uint32 ∗DataArray, const uint32 ∗AddressArray, uint32 DataArrayLength)

  Performs multiple read transfers with DMA.
- Zipwire_Ip_StatusType Zipwire_ReadDmaBlocking (uint8 HwInstance, uint8 HwChannel, uint32 ∗Data↩ Array, const uint32 ∗AddressArray, uint32 DataArrayLength)

  Performs multiple read transfers.
- Zipwire_Ip_StatusType Zipwire_Write (uint8 HwInstance, uint8 HwChannel, Zipwire_Ip_TransferDescriptor ∗DataArray, uint32 DataArrayLength)

  Performs multiple write transfers.
- Zipwire_Ip_StatusType Zipwire_WriteBlocking (uint8 HwInstance, uint8 HwChannel, Zipwire_Ip_TransferDescriptor ∗DataArray, uint32 DataArrayLength)

  Performs multiple write transfers synchronously.
- Zipwire_Ip_StatusType Zipwire_WriteDma (uint8 HwInstance, uint8 HwChannel, const uint32 ∗DataArray, const uint32 ∗AddressArray, uint32 DataArrayLength)

  Performs multiple write transfers using DMA.
- Zipwire_Ip_StatusType Zipwire_WriteDmaBlocking (uint8 HwInstance, uint8 HwChannel, const uint32 ∗DataArray, const uint32 ∗AddressArray, uint32 DataArrayLength)

  Performs multiple write transfers using DMA, synchronously.
- Zipwire_Ip_StatusType Zipwire_StreamWrite (uint8 HwInstance, uint8 HwChannel, uint32 DataAddress, uint32 TargetAcrRegAddress, const uint32 ∗Data)

  Performs a synchronous stream write.
- Zipwire_Ip_StatusType Zipwire_RequestId (uint8 HwInstance, uint8 HwChannel, uint32 ∗Id)

  Performs an ID request transfer.
- Zipwire_Ip_StatusType Zipwire_Trigger (uint8 HwInstance, uint8 HwChannel)

  Sends a trigger command to the target.
- Zipwire_Ip_StatusType Zipwire_GetChannelStatus (uint8 HwInstance, uint8 HwChannel)

  Returns the channel status.
- Zipwire_Ip_Callback Zipwire_InstallGlobalCallback (uint8 HwInstance, Zipwire_Ip_Callback Callback↩ Function, void ∗CallbackParam)

  Installs a global driver callback.
- Zipwire_Ip_ChannelCallback Zipwire_InstallChannelCallback (uint8 HwInstance, uint8 HwChannel, Zipwire_Ip_ChannelCallback CallbackFunction, void ∗CallbackParam)

  Installs a channel callback.

6.1.2 Macro Definition Documentation

6.1.2.1 CDD_ZIPWIRE_MODULE_ID

```
#define CDD_ZIPWIRE_MODULE_ID
```

Parameters that shall be published within the Zipwire driver header file and also in the module's description file.

Definition at line 62 of file CDD_Zipwire.h.

6.1.2.2 ZIPWIRE_INSTANCE_ID

```
#define ZIPWIRE_INSTANCE_ID
```

ID of ZIPWIRE Instance.

Parameters used when raising an error/exception

Definition at line 154 of file CDD_Zipwire.h.

6.1.2.3 ZIPWIRE_INIT_ID

```
#define ZIPWIRE_INIT_ID
```

API service ID for Zipwire_Init function.

Parameters used when raising an error/exception

Definition at line 168 of file CDD_Zipwire.h.

6.1.2.4 ZIPWIRE_DEINIT_ID

```
#define ZIPWIRE_DEINIT_ID
```

API service ID for Zipwire_DeInit function.

Parameters used when raising an error/exception

Definition at line 174 of file CDD_Zipwire.h.

6.1.2.5 ZIPWIRE_READ_ID

```
#define ZIPWIRE_READ_ID
```

API service ID for Zipwire_Read function.

Parameters used when raising an error/exception

Definition at line 180 of file CDD_Zipwire.h.

6.1.2.6 ZIPWIRE_READBLOCKING_ID

```
#define ZIPWIRE_READBLOCKING_ID
```

API service ID for Zipwire_ReadBlocking function.

Parameters used when raising an error/exception

Definition at line 186 of file CDD_Zipwire.h.

6.1.2.7 ZIPWIRE_READDMA_ID

```
#define ZIPWIRE_READDMA_ID
```

API service ID for Zipwire_ReadDma function.

Parameters used when raising an error/exception

Definition at line 192 of file CDD_Zipwire.h.

6.1.2.8 ZIPWIRE_READDMABLOCKING_ID

```
#define ZIPWIRE_READDMABLOCKING_ID
```

API service ID for Zipwire_ReadDmaBlocking function.

Parameters used when raising an error/exception

Definition at line 198 of file CDD_Zipwire.h.

6.1.2.9 ZIPWIRE_WRITE_ID

```
#define ZIPWIRE_WRITE_ID
```

API service ID for Zipwire_Write function.

Parameters used when raising an error/exception

Definition at line 204 of file CDD_Zipwire.h.

6.1.2.10 ZIPWIRE_WRITEBLOCKING_ID

```
#define ZIPWIRE_WRITEBLOCKING_ID
```

API service ID for Zipwire_WriteBlocking function.

Parameters used when raising an error/exception

Definition at line 210 of file CDD_Zipwire.h.

6.1.2.11 ZIPWIRE_WRITEDMA_ID

```
#define ZIPWIRE_WRITEDMA_ID
```

API service ID for Zipwire_WriteDma function.

Parameters used when raising an error/exception

Definition at line 216 of file CDD_Zipwire.h.

6.1.2.12 ZIPWIRE_WRITEDMABLOCKING_ID

```
#define ZIPWIRE_WRITEDMABLOCKING_ID
```

API service ID for Zipwire_WriteDmaBlocking function.

Parameters used when raising an error/exception

Definition at line 222 of file CDD_Zipwire.h.

6.1.2.13 ZIPWIRE_STREAMWRITE_ID

```
#define ZIPWIRE_STREAMWRITE_ID
```

API service ID for Zipwire_StreamWrite function.

Parameters used when raising an error/exception

Definition at line 228 of file CDD_Zipwire.h.

6.1.2.14 ZIPWIRE_REQUESTID_ID

```
#define ZIPWIRE_REQUESTID_ID
```

API service ID for Zipwire_RequestId function.

Parameters used when raising an error/exception

Definition at line 234 of file CDD_Zipwire.h.

6.1.2.15 ZIPWIRE_TRIGGER_ID

```
#define ZIPWIRE_TRIGGER_ID
```

API service ID for Zipwire_Triger function.

Parameters used when raising an error/exception

Definition at line 240 of file CDD_Zipwire.h.

6.1.2.16 ZIPWIRE_GETCHANNELSTATUS_ID

```
#define ZIPWIRE_GETCHANNELSTATUS_ID
```

API service ID for Zipwire_GetChannelStatus function.

Parameters used when raising an error/exception

Definition at line 246 of file CDD_Zipwire.h.

6.1.2.17 ZIPWIRE_INSTALLGLOBALCALLBACK_ID

```
#define ZIPWIRE_INSTALLGLOBALCALLBACK_ID
```

API service ID for Zipwire_InstallGlobalCallback function.

Parameters used when raising an error/exception

Definition at line 252 of file CDD_Zipwire.h.

6.1.2.18 ZIPWIRE_INSTALLCHANNELCALLBACK_ID

```
#define ZIPWIRE_INSTALLCHANNELCALLBACK_ID
```

API service ID for Zipwire_InstallChannelCallback function.

Parameters used when raising an error/exception

Definition at line 258 of file CDD_Zipwire.h.

6.1.2.19 ZIPWIRE_E_INIT_FAILED

```
#define ZIPWIRE_E_INIT_FAILED
```

The ZIPWIRE module is not properly initialized.

Parameter is used when raising a Det error

Definition at line 267 of file CDD_Zipwire.h.

6.1.2.20 ZIPWIRE_E_DEINIT_FAILED

```
#define ZIPWIRE_E_DEINIT_FAILED
```

The ZIPWIRE module is not properly deinitialized.

Parameter is used when raising a Det error

Definition at line 274 of file CDD_Zipwire.h.

6.1.2.21 ZIPWIRE_E_ALREADY_INITIALIZED

```
#define ZIPWIRE_E_ALREADY_INITIALIZED
```

The ZIPWIRE module is already initialized.

Parameter is used when raising a Det error

Definition at line 281 of file CDD_Zipwire.h.

6.1.2.22 ZIPWIRE_E_UNINIT

```
#define ZIPWIRE_E_UNINIT
```

The ZIPWIRE module is was never initialized.

Parameter is used when raising a Det error

Definition at line 288 of file CDD_Zipwire.h.

6.1.2.23 ZIPWIRE_TYPES_VENDOR_ID

```
#define ZIPWIRE_TYPES_VENDOR_ID
```

Parameters that shall be published within the Zipwire driver header file and also in the module's description file.

Definition at line 60 of file Zipwire_Types.h.

6.1.3 Types Reference

6.1.3.1 Zipwire_InitType

```
typedef Zipwire_Ip_ConfigType Zipwire_InitType
```

This type contains the Zipwire Initialization.

The Zipwire Ip Initialization contains all the information required to initialize the ZIPWIRE Channels Internal driver structure.

Definition at line 112 of file Zipwire_Types.h.

6.1.4 Enum Reference

6.1.4.1 Zipwire_StatusType

```
enum Zipwire_StatusType
```

This type contains the Zipwire driver states.

The role is identified by the following structure. Internal driver enumeration.

Definition at line 119 of file Zipwire_Types.h.

6.1.5 Function Reference

6.1.5.1 Zipwire_Init()

```
void Zipwire_Init (
            const Zipwire_InitType * ConfigPtr )
```

This service will store the Zipwire driver installation configuration based on user configuration.

This service is a non-reentrant function that shall store user configuration. The initialization is applied for the enabled IPs, configured statically.

Parameters

in	*ConfigPtr*	- Pointer to the Zipwire specific configuration structure that contains static configuration.

Returns

void

6.1.5.2 Zipwire_DeInit()

```
void Zipwire_DeInit (
            const Zipwire_InitType * ConfigPtr )
```

This service will store the Zipwire driver installation configuration based on user configuration.

This service is a non-reentrant function that shall store user configuration. The initialization is applied for the enabled IPs, configured statically.

Parameters

in	*ConfigPtr*	- Pointer to the Zipwire specific configuration structure that contains static configuration.

Returns

void

6.1.5.3 Zipwire_Read()

```
Zipwire_Ip_StatusType Zipwire_Read (
            uint8 HwInstance,
            uint8 HwChannel,
            Zipwire_Ip_TransferDescriptor * DataArray,
            uint32 DataArrayLength )
```

Performs multiple read transfers.

This function performs multiple reads from the addresses supplied by the user within the array parameter. It returns once the first transfer is launched. If a callback is installed, the user will be notified when the last read transfer is done; otherwise, transfer status can be checked by calling 'Zipwire_Ip_GetChannelStatus'.

Parameters

in	*HwInstance*	- Instance number
in	*HwChannel*	- Channel number
in	*DataArray*	- Array of transfer descriptors (address, size, data)
in	*DataArrayLength*	- Length of the data array

Returns

An error code or ZIPWIRE_IP_STATUS_SUCCESS

6.1.5.4 Zipwire_ReadBlocking()

```
Zipwire_Ip_StatusType Zipwire_ReadBlocking (
             uint8 HwInstance,
             uint8 HwChannel,
             Zipwire_Ip_TransferDescriptor * DataArray,
             uint32 DataArrayLength )
```

Performs multiple read transfers synchronously.

This function performs multiple reads from the addresses supplied by the user within the array parameter. It does not return until all the read requests are served or an error occurs. Read data is stored in the array elements.

Parameters

in	*HwInstance*	- Instance number
in	*HwChannel*	- Channel number
in	*DataArray*	- Array of transfer descriptors (address, size, data)
in	*DataArrayLength*	- Length of the data array

Returns

An error - code or ZIPWIRE_STATUS_SUCCESS

6.1.5.5 Zipwire_ReadDma()

```
Zipwire_Ip_StatusType Zipwire_ReadDma (
             uint8 HwInstance,
             uint8 HwChannel,
             uint32 * DataArray,
             const uint32 * AddressArray,
             uint32 DataArrayLength )
```

Performs multiple read transfers with DMA.

This function performs multiple reads from the addresses supplied by the user within the array parameter. It returns once the first transfer is launched. If a callback is installed, the user will be notified when the last read transfer is done; otherwise, transfer status can be checked by calling 'Zipwire_Ip_GetChannelStatus'.

Parameters

in	*HwInstance*	- Instance number
in	*HwChannel*	- Channel number
in	*DataArray*	- Array of transfer descriptors (address, size, data)
in	*AddressArray*	- Array containing target addresses where data will be read from
in	*DataArrayLength*	- Length of the data & address buffers

Returns

An error code or ZIPWIRE_IP_STATUS_SUCCESS

6.1.5.6 Zipwire_ReadDmaBlocking()

```
Zipwire_Ip_StatusType Zipwire_ReadDmaBlocking (
            uint8 HwInstance,
            uint8 HwChannel,
            uint32 * DataArray,
            const uint32 * AddressArray,
            uint32 DataArrayLength )
```

Performs multiple read transfers.

This function performs multiple reads from the addresses supplied by the user within the array parameter. It returns once the first transfer is launched. If a callback is installed, the user will be notified when the last read transfer is done; otherwise, transfer status can be checked by calling 'Zipwire_Ip_GetChannelStatus'.

Parameters

in	*HwInstance*	- Instance number
in	*HwChannel*	- Channel number
in	*DataArray*	- Array of transfer descriptors (address, size, data)
in	*AddressArray*	- Array containing target addresses where data will be read from
in	*DataArrayLength*	- Length of the data & address buffers

Returns

An error code or ZIPWIRE_IP_STATUS_SUCCESS

6.1.5.7 Zipwire_Write()

```
Zipwire_Ip_StatusType Zipwire_Write (
            uint8 HwInstance,
            uint8 HwChannel,
            Zipwire_Ip_TransferDescriptor * DataArray,
            uint32 DataArrayLength )
```

Performs multiple write transfers.

This function performs multiple write operations at the addresses supplied by the user within the array parameter. It returns once the first transfer is launched. If a callback is installed, the user will be notified when the last write transfer is done; otherwise, transfer status can be checked with by calling 'Zipwire_Ip_GetChannelStatus'.

Parameters

in	*HwInstance*	- Instance number
in	*HwChannel*	- Channel number
in	*DataArray*	- Array of transfer descriptors (address, size, data)
in	*DataArrayLength*	- Length of the data array

Returns

An error - code or ZIPWIRE_IP_STATUS_SUCCESS

6.1.5.8 Zipwire_WriteBlocking()

```
Zipwire_Ip_StatusType Zipwire_WriteBlocking (
            uint8 HwInstance,
            uint8 HwChannel,
            Zipwire_Ip_TransferDescriptor * DataArray,
            uint32 DataArrayLength )
```

Performs multiple write transfers synchronously.

This function performs multiple write operations at the addresses supplied by the user within the array parameter. It does not return until the last write operation is completed or an error occurred.

Parameters

in	*HwInstance*	- Instance number
in	*HwChannel*	- Channel number
in	*DataArray*	- Array of transfer descriptors (address, size, data)
in	*DataArrayLength*	- Length of the data array

Returns

An error - code or ZIPWIRE_IP_STATUS_SUCCESS

6.1.5.9 Zipwire_WriteDma()

```
Zipwire_Ip_StatusType Zipwire_WriteDma (
            uint8 HwInstance,
            uint8 HwChannel,
            const uint32 * DataArray,
            const uint32 * AddressArray,
            uint32 DataArrayLength )
```

Performs multiple write transfers using DMA.

This function performs multiple write transfers from the address supplied by the user, using DMA requests. The DMA engine automatically copies data from the data buffer. The function does not return until all the write requests are served or an error occurs. @Note: Only 32-bits transfers are supported in DMA mode.

Parameters

in	*HwInstance*	- Instance number
in	*HwChannel*	- Channel number
in	*DataArray*	- Array of transfer descriptors (address, size, data)
in	*AddressArray*	- Array containing target addresses where data will be read from
in	*DataArrayLength*	- Length of the data & address buffers

Returns

An error - code or ZIPWIRE_IP_STATUS_SUCCESS

6.1.5.10 Zipwire_WriteDmaBlocking()

```
Zipwire_Ip_StatusType Zipwire_WriteDmaBlocking (
            uint8 HwInstance,
            uint8 HwChannel,
            const uint32 * DataArray,
            const uint32 * AddressArray,
            uint32 DataArrayLength )
```

Performs multiple write transfers using DMA, synchronously.

This function performs multiple write operations at the addresses supplied by the user within the array parameter. It returns once the first transfer is launched. If a callback is installed, the user will be notified when the last write transfer is done; otherwise, transfer status can be checked with by calling 'Zipwire_Ip_GetChannelStatus'.

Parameters

in	*HwInstance*	- Instance number
in	*HwChannel*	- Channel number
in	*DataArray*	- Array of transfer descriptors (address, size, data)
in	*AddressArray*	- Array containing target addresses where data will be read from
in	*DataArrayLength*	- Length of the data & address buffers

Returns

An error - code or ZIPWIRE_IP_STATUS_SUCCESS

6.1.5.11 Zipwire_StreamWrite()

```
Zipwire_Ip_StatusType Zipwire_StreamWrite (
            uint8 HwInstance,
            uint8 HwChannel,
            uint32 DataAddress,
            uint32 TargetAcrRegAddress,
            const uint32 * Data )
```

Performs a synchronous stream write.

This function performs a streaming write operation. It does not return until all the bytes are transferred.

Parameters

in	*HwInstance*	- Instance number
in	*HwChannel*	- The HwChannel number
in	*DataAddress*	- Target address where the data will be written
in	*TargetAcrRegAddress*	- Address of the SIPI_ACR register on the target node
in	*Data*	- Array of data bytes to be streamed; it should point to an array of minimum 8 bytes (SIPI stream transfer size). It is application responsibility to correctly allocate memory before passing this reference, driver is unaware of memory allocation at application level.

Returns

An error code or ZIPWIRE_IP_STATUS_SUCCESS

6.1.5.12 Zipwire_RequestId()

```
Zipwire_Ip_StatusType Zipwire_RequestId (
            uint8 HwInstance,
            uint8 HwChannel,
            uint32 * Id )
```

Performs an ID request transfer.

This requests the device ID from the target node. The target ID will be saved in the output parameter provided by application.

Parameters

in	*HwInstance*	- Instance number
in	*HwChannel*	- The channel number
in	*Id*	- Reference to user variable where the target ID is stored

Returns

An error code or ZIPWIRE_IP_STATUS_SUCCESS

6.1.5.13 Zipwire_Trigger()

```
Zipwire_Ip_StatusType Zipwire_Trigger (
            uint8 HwInstance,
            uint8 HwChannel )
```

Sends a trigger command to the target.

This function sends a trigger transfer command to the target.

Parameters

in	*HwInstance*	- Instance number
in	*HwChannel*	- The channel number

Returns

An error code or ZIPWIRE_IP_STATUS_SUCCESS

6.1.5.14 Zipwire_GetChannelStatus()

```
Zipwire_Ip_StatusType Zipwire_GetChannelStatus (
            uint8 HwInstance,
            uint8 HwChannel )
```

Returns the channel status.

This function returns the status of the last transfer executed by the channel.

Parameters

in	*HwInstance*	- Instance number
in	*HwChannel*	- The channel number

Returns

The current channel status, or the status of the latest command; ZIPWIRE_IP_STATUS_BUSY is returned if a non-blocking command is in progress; ZIPWIRE_IP_STATUS_SUCCESS is returned if the last command completed successfully; If an error occurred in the last command, an appropriate error code is returned; please check the zipwire error codes descriptions.

6.1.5.15 Zipwire_InstallGlobalCallback()

```
Zipwire_Ip_Callback Zipwire_InstallGlobalCallback (
            uint8 HwInstance,
            Zipwire_Ip_Callback CallbackFunction,
            void * CallbackParam )
```

Installs a global driver callback.

This function installs a driver callback that will treat 'max count reached' and 'global CRC error' events.

Parameters

in	*HwInstance*	- Instance number
in	*CallbackFunction*	- The new callback
in	*CallbackParam*	- The new callback parameter

Returns

Reference to the current callback.

6.1.5.16 Zipwire_InstallChannelCallback()

```
Zipwire_Ip_ChannelCallback Zipwire_InstallChannelCallback (
            uint8 HwInstance,
            uint8 HwChannel,
            Zipwire_Ip_ChannelCallback CallbackFunction,
            void * CallbackParam )
```

Installs a channel callback.

This function installs a callback for a zipwire channel. It will be called on successful read/write, or in case of errors in the transfer.

Parameters

in	*HwInstance*	- Instance number
in	*HwChannel*	- The channel number
in	*CallbackFunction*	- The new callback
in	*CallbackParam*	- The new callback parameter

Returns

Reference to the current channel callback.

6.2 ZIPWIRE IPL Driver

6.2.1 Detailed Description

Data Structures

- struct Zipwire_Ip_SIPI_Channel_Type

  This type contains the Zipwire Ip SIPI Channel Registers. More...
- struct Zipwire_Ip_SIPI_Stream_Channel_Type

  This type contains the Zipwire Ip SIPI Stream Channel Registers. More...
- struct Zipwire_Ip_LfastChannelConfigType

  This type contains the Zipwire Ip LFAST configuration. More...
- struct Zipwire_Ip_TransferDescriptor

  This type contains the Zipwire transfer descriptor. More...
- struct Zipwire_Ip_LogicInstanceConfigType

  This type contains the Zipwire user configuration. More...
- struct Zipwire_Ip_LogicChannelConfigType

  This type contains the Zipwire user configuration. More...
- struct Zipwire_Ip_InstancesInitType

  This type contains the Zipwire Ip Initialization. More...
- struct Zipwire_Ip_ChannelsInitType

  This type contains the Zipwire Ip Initialization. More...
- struct Zipwire_Ip_ConfigType

  This type contains the Zipwire Ip Initialization. More...
- struct Zipwire_Ip_InstanceState

  This type contains the Zipwire configuration for global callback function. More...
- struct Zipwire_Ip_ChannelState

  This type contains the Zipwire channel state structure. More...

Macros

- #define ZIPWIRE_IP_DEVASSERT_VENDOR_ID

  Parameters that shall be published within the standard types header file and also in the module's description file.

Types Reference

- typedef void(∗ Zipwire_Ip_Callback) (uint8 HwInstance, Zipwire_Ip_Events Event, void ∗UserData)

  This type contains the global callback for the zipwire driver .
- typedef void(∗ Zipwire_Ip_ChannelCallback) (uint8 HwInstance, uint8 HwChannel, Zipwire_Ip_Events Event, void ∗UserData)

  This type contains the channel callback for zipwire driver.

Enum Reference

- enum Zipwire_Ip_Sipi_ChannelInterrupt

  Channel interrupts.
- enum Zipwire_Ip_Sipi_ChannelFlag

  Channel interrupt flags.
- enum Zipwire_Ip_Events

  This type contains the Zipwire Ip Events Type.
- enum Zipwire_Ip_StatusType

  This type contains the Zipwire Ip LFAST driver states.
- enum Zipwire_Ip_LfastRole

  This type contains the Zipwire Ip LFAST Role(MASTER/SLAVE).
- enum Zipwire_Ip_LfastSpeedMode

  This type contains the Zipwire Ip LFAST Speed Mode(LOW/HIGH).
- enum Zipwire_Ip_LfastLowSpeedClk

  This type contains the Zipwire Ip LFAST clock division in low speed.
- enum Zipwire_Ip_AddressOffset

  This type contains the Zipwire Ip address offset.
- enum Zipwire_Ip_TransferSize

  This type contains the Zipwire Ip address offset.
- enum Zipwire_Ip_TimeoutPrescaler

  This type contains the Zipwire Ip timeout counter prescaler.

Function Reference

- void Zipwire_Ip_Init (const Zipwire_Ip_ConfigType ∗const pxZipwireConfig)

  This function initializes the Zipwire Driver in IP Layer.
- void Zipwire_Ip_DeInit (const Zipwire_Ip_ConfigType ∗const pxZipwireConfig)

  This function initializes the Zipwire Driver in IP Layer.
- Zipwire_Ip_StatusType Zipwire_Ip_InitInstance (const Zipwire_Ip_LogicInstanceConfigType ∗pxZipwire←
InstanceConfig)

  Initializes the driver.
- Zipwire_Ip_StatusType Zipwire_Ip_DeInitInstance (uint8 HwInstance)

  De-initializes the ZIPWIRE driver.
- Zipwire_Ip_StatusType Zipwire_Ip_InitChannel (const Zipwire_Ip_LogicChannelConfigType ∗pxZipwire←
ChannelConfig)

  Initializes a ZIPWIRE HwChannel.
- Zipwire_Ip_StatusType Zipwire_Ip_DeInitChannel (uint8 HwInstance, uint8 HwChannel)

  De-initializes a ZIPWIRE HwChannel.
- Zipwire_Ip_StatusType Zipwire_Ip_Read (uint8 HwInstance, uint8 HwChannel, Zipwire_Ip_TransferDescriptor
∗DataArray, uint32 DataArrayLength)

  Performs multiple read transfers.
- Zipwire_Ip_StatusType Zipwire_Ip_ReadBlocking (uint8 HwInstance, uint8 HwChannel, Zipwire_Ip_TransferDescriptor
∗DataArray, uint32 DataArrayLength)

  Performs multiple read transfers synchronously.

- Zipwire_Ip_StatusType Zipwire_Ip_Write (uint8 HwInstance, uint8 HwChannel, Zipwire_Ip_TransferDescriptor ∗DataArray, uint32 DataArrayLength)

  Performs multiple write transfers.
- Zipwire_Ip_StatusType Zipwire_Ip_WriteBlocking (uint8 HwInstance, uint8 HwChannel, Zipwire_Ip_TransferDescriptor ∗DataArray, uint32 DataArrayLength)

  Performs multiple write transfers synchronously.
- Zipwire_Ip_StatusType Zipwire_Ip_StreamWrite (uint8 HwInstance, uint8 HwChannel, uint32 DataAddress, uint32 TargetAcrRegAddress, const uint32 ∗Data)

  Performs a synchronous stream write.
- Zipwire_Ip_StatusType Zipwire_Ip_RequestId (uint8 HwInstance, uint8 HwChannel, uint32 ∗Id)

  Performs an ID request transfer.
- Zipwire_Ip_StatusType Zipwire_Ip_Trigger (uint8 HwInstance, uint8 HwChannel)

  Sends a trigger command to the target.
- Zipwire_Ip_StatusType Zipwire_Ip_GetChannelStatus (uint8 HwInstance, uint8 HwChannel)

  Returns the channel status.
- Zipwire_Ip_Callback Zipwire_Ip_InstallGlobalCallback (uint8 HwInstance, Zipwire_Ip_Callback Callback←Function, void ∗CallbackParam)

  Installs a global driver callback.
- Zipwire_Ip_ChannelCallback Zipwire_Ip_InstallChannelCallback (uint8 HwInstance, uint8 HwChannel, Zipwire_Ip_ChannelCallback CallbackFunction, void ∗CallbackParam)

  Installs a channel callback.
- Zipwire_Ip_StatusType Zipwire_Ip_MasterInit (DIGRF_TOP_Type ∗pxZipwireBase, Zipwire_Ip_LfastSpeedMode SpeedMode, Zipwire_Ip_LfastLowSpeedClk LowSpeedClkDiv, uint32 Timeout, uint32 Attempts)

  LFAST Master initialization.
- Zipwire_Ip_StatusType Zipwire_Ip_SlaveInit (DIGRF_TOP_Type ∗pxZipwireBase, Zipwire_Ip_LfastSpeedMode SpeedMode, Zipwire_Ip_LfastLowSpeedClk LowSpeedClkDiv, uint32 Timeout)

  LFAST Slave initialization.

6.2.2 Data Structure Documentation

6.2.2.1 struct Zipwire_Ip_SIPI_Channel_Type

This type contains the Zipwire Ip SIPI Channel Registers.

The Channel is identified by the following structure: Configure registers for each SIPI Channel.

Definition at line 257 of file Zipwire_Ip_Types.h.

Data Fields

Type	Name	Description
volatile uint32	CCR	SIPI Channel Control Register
volatile uint32	CSR	SIPI Channel Status Register
uint8	RESERVED_0[4]	
volatile uint32	CIR	SIPI Channel Interrupt Register
volatile uint32	CTOR	SIPI Channel Timeout Register
volatile const uint32	CCRC	SIPI Channel ZIPWIRE Register
volatile uint32	CAR	SIPI Channel Address Register
volatile uint32	CDR	SIPI Channel Data Register

6.2.2.2 struct Zipwire_Ip_SIPI_Stream_Channel_Type

This type contains the Zipwire Ip SIPI Stream Channel Registers.

The Channel is identified by the following structure Configure registers for each SIPI Stream Channel.

Definition at line 274 of file Zipwire_Ip_Types.h.

Data Fields

Type	Name	Description
volatile uint32	CCR	SIPI Channel Control Register
volatile uint32	CSR	SIPI Channel Status Register
uint8	RESERVED_0[4]	
volatile uint32	CIR	SIPI Channel Interrupt Register
volatile uint32	CTOR	SIPI Channel Timeout Register
volatile const uint32	CCRC	SIPI Channel ZIPWIRE Register
volatile uint32	CAR	SIPI Channel Address Register
volatile uint32	CDR[SIPI1_CDR2_COUNT]	SIPI Channel Data Register

6.2.2.3 struct Zipwire_Ip_LfastChannelConfigType

This type contains the Zipwire Ip LFAST configuration.

The LFAST configuration is identified by the following structure.

Definition at line 290 of file Zipwire_Ip_Types.h.

Data Fields

Type	Name	Description
Zipwire_Ip_LfastRole	Role	LFAST role: MASTER/SLAVE
Zipwire_Ip_LfastSpeedMode	SpeedMode	LFAST speed mode: high-speed/low-speed
uint32	SyncTimeout	Timeout used for the LFAST master-slave synchronization; @Note: A value of zero for this parameter is equivalent to timeout being disregarded by the driver; the LFAST initialization will wait forever for commands/responses from the other node.
uint32	SyncAttempts	Number of attempts for the master to synchronize with the slave; this field is only used by the master node. @Note: A value of zero for this parameter is equivalent to an infinite number of attempts; the LFAST master will try forever to synchronize with the slave
Zipwire_Ip_LfastLowSpeedClk	LowSpeedClkDiv	LFAST clock division factor in Low Speed Select mode.

6.2.2.4 struct Zipwire_Ip_TransferDescriptor

This type contains the Zipwire transfer descriptor.

The zipwire transfer descriptor is identified by the following structure.

Definition at line 311 of file Zipwire_Ip_Types.h.

6.2.2.5 struct Zipwire_Ip_LogicInstanceConfigType

This type contains the Zipwire user configuration.

The zipwire instance user configuration is identified by the following structure.

Definition at line 323 of file Zipwire_Ip_Types.h.

Data Fields

Type	Name	Description
uint8	HwInstance	SIPI instance number
Zipwire_Ip_LfastChannelConfigType ∗	LfastConfig	LFAST configuration
Zipwire_Ip_AddressOffset	AddrOffset	Address increment/decrement for stream transfers
Zipwire_Ip_TimeoutPrescaler	TimeoutClkDiv	SIPI timeout clock prescaler
boolean	MaxCountReachedInt	Maximum address reached interrupt enable
Zipwire_Ip_Callback	Callback	Global callback (max count reached/global ZIPWIRE error)
void ∗	CallbackParam	Global callback parameter

6.2.2.6 struct Zipwire_Ip_LogicChannelConfigType

This type contains the Zipwire user configuration.

The zipwire channel user configuration is identified by the following structure.

Definition at line 339 of file Zipwire_Ip_Types.h.

Data Fields

Type	Name	Description
uint8	HwChannel	SIPI channel number (0-3)
uint8	HwInstance	SIPI instance number
uint8	Timeout	Timeout value for requests.
boolean	DmaEnable	Enable channel DMA functionality
uint8	DmaDataChn	DMA channel number used to transfer data

Data Fields

Type	Name	Description
uint8	DmaAddrChn	DMA channel number used to transfer addresses
boolean	TimeoutErrIrq	Enables/disables timeout error handling for the channel
boolean	AckErrIrq	Enables/disables ACK error handling for the channel
boolean	TransIdErrIrq	Enables/disables transaction ID error handling for the channel
Zipwire_Ip_ChannelCallback	Callback	Channel error callback
void *	CallbackParam	Channel error callback parameter

6.2.2.7 struct Zipwire_Ip_InstancesInitType

This type contains the Zipwire Ip Initialization.

The Zipwire Ip Initialization contains all the information required to initialize the ZIPWIRE Instances Internal driver structure.

Definition at line 362 of file Zipwire_Ip_Types.h.

Data Fields

Type	Name	Description
const Zipwire_Ip_LogicInstanceConfigType *const *	LogicInstanceConfigList	Pointer to list LogicInstanceConfig

6.2.2.8 struct Zipwire_Ip_ChannelsInitType

This type contains the Zipwire Ip Initialization.

The Zipwire Ip Initialization contains all the information required to initialize the ZIPWIRE Channels Internal driver structure.

Definition at line 374 of file Zipwire_Ip_Types.h.

Data Fields

Type	Name	Description
const Zipwire_Ip_LogicChannelConfigType *const *	LogicChannelConfigList	Pointer to list LogicChannelConfig

6.2.2.9 struct Zipwire_Ip_ConfigType

This type contains the Zipwire Ip Initialization.

The Zipwire Ip Initialization contains all the information required to initialize the ZIPWIRE Channels Internal driver structure.

Definition at line 386 of file Zipwire_Ip_Types.h.

Data Fields

Type	Name	Description
const Zipwire_Ip_InstancesInitType ∗	ZipwireIpInstanceCfg	Pointer to list LogicChannelConfig
const Zipwire_Ip_ChannelsInitType ∗	ZipwireIpChannelCfg	Pointer to list LogicInstanceConfig

6.2.2.10 struct Zipwire_Ip_InstanceState

This type contains the Zipwire configuration for global callback function.

The zipwire global instance state is identified by the following structure.

Definition at line 397 of file Zipwire_Ip_Types.h.

Data Fields

Type	Name	Description
Zipwire_Ip_Callback	Callback	Global callback (max count reached/global ZIPWIRE error)
void ∗	CallbackParam	Global callback parameter

6.2.2.11 struct Zipwire_Ip_ChannelState

This type contains the Zipwire channel state structure.

The zipwire channel state structure is identified by the following structure.

Definition at line 408 of file Zipwire_Ip_Types.h.

Data Fields

Type	Name	Description
uint8	HwInstance	SIPI instance number
uint8	HwChannel	SIPI channel number
Zipwire_Ip_ChannelCallback	Callback	Channel callback
void ∗	CallbackParam	Channel callback parameter
volatile Zipwire_Ip_StatusType	ChannelStatus	Channel status
volatile boolean	IsBlocking	Flag used for channel blocking operation
Zipwire_Ip_TransferDescriptor ∗	TransferBuffer	Internal array holding the buffer for continuous transfers
volatile uint32	RemainingTransfers	Number of continuous transfers still to be served

Data Fields

Type	Name	Description
boolean	DmaEnable	Enable DMA functionality for this channel
boolean	DmaWriteTransfer	Flag used to differentiate between read and write DMA transfers
uint8	DmaDataChn	DMA channel number used to transfer data
uint8	DmaAddrChn	DMA channel number used to transfer addresses
volatile boolean	IdRequest	Flag used to mark ID request transfers
boolean	DisableNotification	Flag used when the application must not be notified of an event
uint32 *	TargetId	Internal variable storing the requested target ID

6.2.3 Macro Definition Documentation

6.2.3.1 ZIPWIRE_IP_DEVASSERT_VENDOR_ID

```
#define ZIPWIRE_IP_DEVASSERT_VENDOR_ID
```

Parameters that shall be published within the standard types header file and also in the module's description file.

Definition at line 61 of file Zipwire_Ip_DevAssert.h.

6.2.4 Types Reference

6.2.4.1 Zipwire_Ip_Callback

```
typedef void(* Zipwire_Ip_Callback) (uint8 HwInstance, Zipwire_Ip_Events Event, void *UserData)
```

This type contains the global callback for the zipwire driver .

The Callback is defined by the user and installed by the driver in the corresponding IRQ.

Returns

void

Definition at line 237 of file Zipwire_Ip_Types.h.

6.2.4.2 Zipwire_Ip_ChannelCallback

```
typedef void(* Zipwire_Ip_ChannelCallback) (uint8 HwInstance, uint8 HwChannel, Zipwire_Ip_Events Event,
void *UserData)
```

This type contains the channel callback for zipwire driver.

The Callback is defined by the user and installed by the driver in the corresponding IRQ.

Returns

void

Definition at line 248 of file Zipwire_Ip_Types.h.

6.2.5 Enum Reference

6.2.5.1 Zipwire_Ip_Sipi_ChannelInterrupt

```
enum Zipwire_Ip_Sipi_ChannelInterrupt
```

Channel interrupts.

Definition at line 108 of file Zipwire_Ip_Sipi_Hw_Access.h.

6.2.5.2 Zipwire_Ip_Sipi_ChannelFlag

```
enum Zipwire_Ip_Sipi_ChannelFlag
```

Channel interrupt flags.

Definition at line 121 of file Zipwire_Ip_Sipi_Hw_Access.h.

6.2.5.3 Zipwire_Ip_Events

```
enum Zipwire_Ip_Events
```

This type contains the Zipwire Ip Events Type.

The Event State type provides information about the Zipwire general state.

Enumerator

ZIPWIRE_EVENT_GLOBAL_CRC_ERR	Global ZIPWIRE error
ZIPWIRE_EVENT_MAX_COUNT_REACHED	Address maximum count reached
ZIPWIRE_EVENT_TIMEOUT_ERR	Channel timeout
ZIPWIRE_EVENT_TRANSACTION_ID_ERR	Transaction ID error
ZIPWIRE_EVENT_ACK_ERR	Error on received ACK
ZIPWIRE_EVENT_WRITE_COMPLETE	ACK received correctly
ZIPWIRE_EVENT_READ_COMPLETE	Read answer received
ZIPWIRE_EVENT_TARGET_ID_RECEIVED	ID request served; received target ID
ZIPWIRE_EVENT_TRIGGER_COMMAND	Trigger command received
ZIPWIRE_EVENT_DMA_ERR	DMA error

Definition at line 120 of file Zipwire_Ip_Types.h.

6.2.5.4 Zipwire_Ip_StatusType

```
enum Zipwire_Ip_StatusType
```

This type contains the Zipwire Ip LFAST driver states.

The role is identified by the following structure. Internal driver enumeration.

Definition at line 140 of file Zipwire_Ip_Types.h.

6.2.5.5 Zipwire_Ip_LfastRole

```
enum Zipwire_Ip_LfastRole
```

This type contains the Zipwire Ip LFAST Role(MASTER/SLAVE).

The role is identified by the following structure. Internal driver enumeration.

Definition at line 157 of file Zipwire_Ip_Types.h.

6.2.5.6 Zipwire_Ip_LfastSpeedMode

```
enum Zipwire_Ip_LfastSpeedMode
```

This type contains the Zipwire Ip LFAST Speed Mode(LOW/HIGH).

The speed mode is identified by the following structure. Internal driver enumeration.

Definition at line 168 of file Zipwire_Ip_Types.h.

6.2.5.7 Zipwire_Ip_LfastLowSpeedClk

```
enum Zipwire_Ip_LfastLowSpeedClk
```

This type contains the Zipwire Ip LFAST clock division in low speed.

The clock division in low speed is identified by the following structure. Internal driver enumeration.

Definition at line 179 of file Zipwire_Ip_Types.h.

6.2.5.8 Zipwire_Ip_AddressOffset

```
enum Zipwire_Ip_AddressOffset
```

This type contains the Zipwire Ip address offset.

The address offset is identified by the following structure. Internal driver enumeration.

Enumerator

ZIPWIRE_ADDR_NO_CHANGE	No change. Address stays the same after the transfer is done
ZIPWIRE_ADDR_INC_4	Increment address by 4
ZIPWIRE_ADDR_DEC_4	Decrement address by 4

Definition at line 190 of file Zipwire_Ip_Types.h.

6.2.5.9 Zipwire_Ip_TransferSize

```
enum Zipwire_Ip_TransferSize
```

This type contains the Zipwire Ip address offset.

The transfer size is identified by the following structure. Internal driver enumeration.

Enumerator

ZIPWIRE_8_BITS	8-bit transfer
ZIPWIRE_16_BITS	16-bit transfer
ZIPWIRE_32_BITS	32-bit transfer

Definition at line 203 of file Zipwire_Ip_Types.h.

6.2.5.10 Zipwire_Ip_TimeoutPrescaler

```
enum Zipwire_Ip_TimeoutPrescaler
```

This type contains the Zipwire Ip timeout counter prescaler.

The timeout counter prescaler is identified by the following structure. Internal driver enumeration.

Enumerator

ZIPWIRE_DIV_64	Timeout counter clock = system clock/64
ZIPWIRE_DIV_128	Timeout counter clock = system clock/128
ZIPWIRE_DIV_256	Timeout counter clock = system clock/256
ZIPWIRE_DIV_512	Timeout counter clock = system clock/512
ZIPWIRE_DIV_1024	Timeout counter clock = system clock/1024

Definition at line 216 of file Zipwire_Ip_Types.h.

6.2.6 Function Reference

6.2.6.1 Zipwire_Ip_Init()

```
void Zipwire_Ip_Init (
            const Zipwire_Ip_ConfigType *const pxZipwireConfig )
```

This function initializes the Zipwire Driver in IP Layer.

This service is a non-reentrant function that shall initialize the Zipwire driver. The user must make sure that the clock is enabled.

Parameters

in	*pxZipwireConfig*	- Pointer to the configuration structure.

Returns

void

6.2.6.2 Zipwire_Ip_DeInit()

```
void Zipwire_Ip_DeInit (
            const Zipwire_Ip_ConfigType *const pxZipwireConfig )
```

This function initializes the Zipwire Driver in IP Layer.

This service is a non-reentrant function that shall initialize the Zipwire driver. The user must make sure that the clock is enabled.

Parameters

in	*pxZipwireConfig*	- Pointer to the configuration structure.

Returns

void

6.2.6.3 Zipwire_Ip_InitInstance()

```
Zipwire_Ip_StatusType Zipwire_Ip_InitInstance (
            const Zipwire_Ip_LogicInstanceConfigType * pxZipwireInstanceConfig )
```

Initializes the driver.

This function initializes the appropriate SIPI and LFAST interfaces, according to the configuration passed by the user.

Parameters

in	*pxZipwireInstanceConfig*	- Specifies the Logic Instance Configuration defined by the user

Returns

An error code or ZIPWIRE_IP_STATUS_SUCCESS

6.2.6.4 Zipwire_Ip_DeInitInstance()

```
Zipwire_Ip_StatusType Zipwire_Ip_DeInitInstance (
            uint8 HwInstance )
```

De-initializes the ZIPWIRE driver.

This function shuts down the communication interfaces and brings the driver state machine back to the uninitialized state.

Parameters

in	*pxZipwireInstanceConfig*	- Specifies the Logic Instance Configuration defined by the user
in	*pxZipwireChannelInit*	- Specifies the Logic Channel Configuration defined by the user

Returns

An error code or ZIPWIRE_IP_STATUS_SUCCESS

6.2.6.5 Zipwire_Ip_InitChannel()

```
Zipwire_Ip_StatusType Zipwire_Ip_InitChannel (
            const Zipwire_Ip_LogicChannelConfigType * pxZipwireChannelConfig )
```

Initializes a ZIPWIRE HwChannel.

This function initializes a HwChannel with the settings provided by the user.

Parameters

in	*pxZipwireChannelConfig*	- Specifies the Logic Channel Config defined by the user

Returns

An error code or ZIPWIRE_IP_STATUS_SUCCESS

6.2.6.6 Zipwire_Ip_DeInitChannel()

```
Zipwire_Ip_StatusType Zipwire_Ip_DeInitChannel (
            uint8 HwInstance,
            uint8 HwChannel )
```

De-initializes a ZIPWIRE HwChannel.

This function de-initializes a HwChannel.

Parameters

in	*pxZipwireChannelConfig*	- Specifies the Logic Channel Config defined by the user

Returns

An error code or ZIPWIRE_IP_STATUS_SUCCESS

6.2.6.7 Zipwire_Ip_Read()

```
Zipwire_Ip_StatusType Zipwire_Ip_Read (
            uint8 HwInstance,
            uint8 HwChannel,
            Zipwire_Ip_TransferDescriptor * DataArray,
            uint32 DataArrayLength )
```

Performs multiple read transfers.

This function performs multiple reads from the addresses supplied by the user within the array parameter. It returns once the first transfer is launched. If a callback is installed, the user will be notified when the last read transfer is done; otherwise, transfer status can be checked by calling 'Zipwire_Ip_GetChannelStatus'.

Parameters

in	*HwInstance*	- Instance number
in	*HwChannel*	- The HwChannel number
	[in/out]	DataArray - Array of transfer descriptors (address, size, data)
in	*DataArrayLength*	- Length of the data array

Returns

An error code or ZIPWIRE_IP_STATUS_SUCCESS

6.2.6.8 Zipwire_Ip_ReadBlocking()

```
Zipwire_Ip_StatusType Zipwire_Ip_ReadBlocking (
            uint8 HwInstance,
            uint8 HwChannel,
            Zipwire_Ip_TransferDescriptor * DataArray,
            uint32 DataArrayLength )
```

Performs multiple read transfers synchronously.

This function performs multiple reads from the addresses supplied by the user within the array parameter. It does not return until all the read requests are served or an error occurs. Read data is stored in the array elements.

Parameters

in	*HwInstance*	- Instance number
in	*HwChannel*	- The HwChannel number
	[in/out]	DataArray - Array of transfer descriptors (address, size, data)
in	*DataArrayLength*	- Length of the data array

Returns

An error code or ZIPWIRE_IP_STATUS_SUCCESS

6.2.6.9 Zipwire_Ip_Write()

```
Zipwire_Ip_StatusType Zipwire_Ip_Write (
            uint8 HwInstance,
            uint8 HwChannel,
            Zipwire_Ip_TransferDescriptor * DataArray,
            uint32 DataArrayLength )
```

Performs multiple write transfers.

This function performs multiple write operations at the addresses supplied by the user within the array parameter. It returns once the first transfer is launched. If a callback is installed, the user will be notified when the last write transfer is done; otherwise, transfer status can be checked with by calling 'ZIPWIRE_DRV_GetChannelStatus'.

Parameters

in	*HwInstance*	- Instance number
in	*HwChannel*	- The HwChannel number
in	*DataArray*	- Array of transfer descriptors (address, size, data)
in	*DataArrayLength*	- Length of the data array

Returns

An error code or ZIPWIRE_IP_STATUS_SUCCESS

6.2.6.10 Zipwire_Ip_WriteBlocking()

```
Zipwire_Ip_StatusType Zipwire_Ip_WriteBlocking (
            uint8 HwInstance,
            uint8 HwChannel,
            Zipwire_Ip_TransferDescriptor * DataArray,
            uint32 DataArrayLength )
```

Performs multiple write transfers synchronously.

This function performs multiple write operations at the addresses supplied by the user within the array parameter. It does not return until the last write operation is completed or an error occurred.

Parameters

in	*HwInstance*	- Instance number
in	*HwChannel*	- The HwChannel number
in	*DataArray*	- Array of transfer descriptors (address, size, data)
in	*DataArrayLength*	- Length of the data array

Returns

An error code or ZIPWIRE_IP_STATUS_SUCCESS

6.2.6.11 Zipwire_Ip_StreamWrite()

```
Zipwire_Ip_StatusType Zipwire_Ip_StreamWrite (
            uint8 HwInstance,
            uint8 HwChannel,
            uint32 DataAddress,
            uint32 TargetAcrRegAddress,
            const uint32 * Data )
```

Performs a synchronous stream write.

This function performs a streaming write operation. It does not return until all the bytes are transferred.

Parameters

in	*HwInstance*	- Instance number
in	*HwChannel*	- The HwChannel number
in	*DataAddress*	- Target address where the data will be written
in	*TargetAcrRegAddress*	- Address of the SIPI_ACR register on the target node
in	*Data*	- Array of data bytes to be streamed; it should point to an array of minimum 8 bytes (SIPI stream transfer size). It is application responsibility to correctly allocate memory before passing this reference, driver is unaware of memory allocation at application level.

Returns

An error code or ZIPWIRE_IP_STATUS_SUCCESS

6.2.6.12 Zipwire_Ip_RequestId()

```
Zipwire_Ip_StatusType Zipwire_Ip_RequestId (
            uint8 HwInstance,
            uint8 HwChannel,
            uint32 * Id )
```

Performs an ID request transfer.

This requests the device ID from the target node. The target ID will be saved in the output parameter provided by application.

Parameters

in	*HwInstance*	- Instance number
in	*HwChannel*	- The channel number
in	*Id*	- Reference to user variable where the target ID is stored

Returns

An error code or ZIPWIRE_IP_STATUS_SUCCESS

6.2.6.13 Zipwire_Ip_Trigger()

```
Zipwire_Ip_StatusType Zipwire_Ip_Trigger (
            uint8 HwInstance,
            uint8 HwChannel )
```

Sends a trigger command to the target.

This function sends a trigger transfer command to the target.

Parameters

in	*HwInstance*	- Instance number
in	*HwChannel*	- The channel number

Returns

An error code or ZIPWIRE_IP_STATUS_SUCCESS

6.2.6.14 Zipwire_Ip_GetChannelStatus()

```
Zipwire_Ip_StatusType Zipwire_Ip_GetChannelStatus (
            uint8 HwInstance,
            uint8 HwChannel )
```

Returns the channel status.

This function returns the status of the last transfer executed by the channel.

Parameters

in	*HwInstance*	- Instance number
in	*HwChannel*	- The channel number

Returns

The current channel status, or the status of the latest command; ZIPWIRE_IP_STATUS_BUSY is returned if a non-blocking command is in progress; ZIPWIRE_IP_STATUS_SUCCESS is returned if the last command completed successfully; If an error occurred in the last command, an appropriate error code is returned; please check the zipwire error codes descriptions.

6.2.6.15 Zipwire_Ip_InstallGlobalCallback()

```
Zipwire_Ip_Callback Zipwire_Ip_InstallGlobalCallback (
            uint8 HwInstance,
            Zipwire_Ip_Callback CallbackFunction,
            void * CallbackParam )
```

Installs a global driver callback.

This function installs a driver callback that will treat 'max count reached' and 'global CRC error' events.

Parameters

in	*HwInstance*	- Instance number
in	*CallbackFunction*	- The new callback
in	*CallbackParam*	- The new callback parameter

Returns

Reference to the current callback.

6.2.6.16 Zipwire_Ip_InstallChannelCallback()

```
Zipwire_Ip_ChannelCallback Zipwire_Ip_InstallChannelCallback (
             uint8 HwInstance,
             uint8 HwChannel,
             Zipwire_Ip_ChannelCallback CallbackFunction,
             void * CallbackParam )
```

Installs a channel callback.

This function installs a callback for a zipwire channel. It will be called on successful read/write, or in case of errors in the transfer.

Parameters

in	*HwInstance*	- Instance number
in	*HwChannel*	- The channel number
in	*CallbackFunction*	- The new callback
in	*CallbackParam*	- The new callback parameter

Returns

Reference to the current channel callback.

6.2.6.17 Zipwire_Ip_MasterInit()

```
Zipwire_Ip_StatusType Zipwire_Ip_MasterInit (
             DIGRF_TOP_Type * pxZipwireBase,
             Zipwire_Ip_LfastSpeedMode SpeedMode,
             Zipwire_Ip_LfastLowSpeedClk LowSpeedClkDiv,
             uint32 Timeout,
             uint32 Attempts )
```

LFAST Master initialization.

Initializes the LFAST master interface

Parameters

in	*pxZipwireBase*	- LFAST pxZipwireBase pointer.

Parameters

in	*PllClkDiv*	- LFAST PLL reference clock divider.
in	*FeedbackDiv*	- Feedback Division factor for LFAST PLL VCO output clock.
in	*SpeedMode*	- low-speed/high-speed.
in	*LowSpeedClkDiv*	- low-speed clock input (sysclk/2 or sysclk/4).
in	*Timeout*	- Cycles allowed for the synchronization to complete. A value of zero passed for the timeout parameter is disregarded by the driver; the master will wait forever for the responses from the slave.
in	*Attempts*	- Number of Attempts for the master to synchronize with the slave; A value of zero for this parameter is equivalent to an infinite number of Attempts; the LFAST master will wait forever for the slave to confirm it's status.

Returns

- error code

6.2.6.18 Zipwire_Ip_SlaveInit()

```
Zipwire_Ip_StatusType Zipwire_Ip_SlaveInit (
            DIGRF_TOP_Type * pxZipwireBase,
            Zipwire_Ip_LfastSpeedMode SpeedMode,
            Zipwire_Ip_LfastLowSpeedClk LowSpeedClkDiv,
            uint32 Timeout )
```

LFAST Slave initialization.

Initializes the LFAST slave interface

Parameters

in	*pxZipwireBase*	- LFAST pxZipwireBase pointer.
in	*PllClkDiv*	- LFAST PLL reference clock divider.
in	*FeedbackDiv*	- Feedback Division factor for LFAST PLL VCO output clock.
in	*SpeedMode*	- low-speed/high-speed.
in	*LowSpeedClkDiv*	- low-speed clock input (sysclk/2 or sysclk/4).
in	*Timeout*	- cycles allowed for the synchronization to complete. A value of zero passed for the timeout parameter is disregarded by the driver; the slave will wait forever for the commands from the master.

Returns

- error code