Generic FSK Link Layer

API Reference Manual

Rev. 12
Jul 2019

Contents

Chapter 1
Genfsk

Chapter 1
Genfsk

1.1 Overview

Files

- file genfsk_interface.h

Data Structures

- struct GENFSK_nwk_addr_match_t
- struct GENFSK_radio_config_t
- struct GENFSK_packet_config_t
- struct GENFSK_crc_config_t
- struct GENFSK_whitener_config_t
- struct GENFSK_bitproc_t
- struct GENFSK_packet_header_t
- struct GENFSK_packet_t

Macros

- #define gGENFSK_IrqPriority_c
- #define gGENFSK_TaskStackSize_c
- #define gGENFSK_TaskPriority_c
- #define gGENFSK_InstancesCnt_c
- #define gGENFSK_InvalidIdx_c

Typedefs

- typedef uint32_t GENFSK_nwk_addr_t
- typedef uint64_t GENFSK_timestamp_t
- typedef void(* genfskPacketReceivedCallBack_t) (uint8_t *pBuffer, uint16_t bufferLength, uint64_t timestamp, uint8_t rssi, uint8_t crcValid)
- typedef void(* genfskEventNotifyCallBack_t) (genfskEvent_t event, genfskEventStatus_t event↩ Status)

Enumerations

- enum genfskDataRate_t {
  gGenfskDR1Mbps,
  gGenfskDR500Kbps,
  gGenfskDR250Kbps }
- enum genfskRadioMode_t {

gGenfskGfskBt0p5h0p5,
gGenfskGfskBt0p5h0p32,
gGenfskGfskBt0p5h0p7,
gGenfskGfskBt0p5h1p0,
gGenfskGfskBt0p3h0p5,
gGenfskGfskBt0p7h0p5,
gGenfskFsk,
gGenfskMsk }

- enum genfskStatus_t {
  gGenfskSuccess_c,
  gGenfskInvalidParameters_c,
  gGenfskFail_c,
  gGenfskNotInitialized_c,
  gGenfskAlreadyInit_c,
  gGenfskBusyRx_c,
  gGenfskBusyTx_c,
  gGenfskBusyPendingRx_c,
  gGenfskBusyPendingTx_c,
  gGenfskInstantPassed_c,
  gGenfskAllocInstanceFailed }
- enum genfskEvent_t {
  gGenfskTxEvent,
  gGenfskRxEvent,
  gGenfskNwkAddressMatch,
  gGenfskWakeEvent,
  gGenfskAllEvents }
- enum genfskEventStatus_t {
  gGenfskSuccess,
  gGenfskRxAllocLengthFail,
  gGenfskTimeout,
  gGenfskSyncLost,
  gGenfskCRCInvalid,
  gGenfskH0Fail,
  gGenfskH1Fail,
  gGenfskLengthFail }
- enum genfskPacketCfgLengthBitOrd_t {
  gGenfskLengthBitLsbFirst,
  gGenfskLengthBitMsbFirst }
- enum genfskCrcComputeMode_t {
  gGenfskCrcDisable,
  gGenfskCrcEnable }
- enum genfskCrcRecvInvalid_t {
  gGenfskCrcSupressInvalid,
  gGenfskCrcRecvInvalid }
- enum genfskCrcCfgCrcRefIn_t {
  gGenfskCrcInputNoRef,

gGenfskCrcRefInput }
- enum genfskCrcCfgCrcRefOut_t { gGenfskCrcOutputNoRef, gGenfskCrcRefOutput }
- enum genfskCrcCfgCrcByteOrd_t { gGenfskCrcLSByteFirst, gGenfskCrcMSByteFirst }
- enum genfskWhitenMode_t { gGenfskWhitenDisable, gGenfskWhitenEnable }
- enum genfskWhitenStart_t { gWhitenStartNoWhitening, gWhitenStartWhiteningAtH0, gWhitenStartWhiteningAtH1, gWhitenStartWhiteningAtPayload }
- enum genfskWhitenEnd_t { gWhitenEndAtEndOfPayload, gWhitenEndAtEndOfCrc }
- enum genfskWhitenB4Crc_t { gCrcB4Whiten, gWhitenB4Crc }
- enum genfskWhitenPolyType_t { gGaloisPolyType, gFibonnaciPolyType }
- enum genfskWhitenCfgRefIn_t { gGenfskWhitenInputNoRef, gGenfskWhitenRefInput }
- enum genfskWhitenCfgPayloadReinit_t { gGenfskWhitenNoPayloadReinit, gGenfskWhitenPayloadReinit }
- enum genfskManchesterEn_t { gGenfskManchesterDisable, gGenfskManchesterEnable }
- enum genfskManchesterInv_t { gGenfskManchesterNoInv, gGenfskManchesterInverted }
- enum genfskManchesterStart_t { gGenfskManchesterStartAtPayload, gGenfskManchesterStartAtHeader }
- enum genfskPacketType_t { gGenfskFormattedPacket, gGenfskRawPacket }

Variables

- uint8_t GENFSK_nwk_addr_match_t::nwkAddrSizeBytes
- uint8_t GENFSK_nwk_addr_match_t::nwkAddrThrBits
- GENFSK_nwk_addr_t GENFSK_nwk_addr_match_t::nwkAddr

- genfskRadioMode_t GENFSK_radio_config_t::radioMode
- genfskDataRate_t GENFSK_radio_config_t::dataRate
- uint8_t GENFSK_packet_config_t::preambleSizeBytes
- genfskPacketType_t GENFSK_packet_config_t::packetType
- uint8_t GENFSK_packet_config_t::lengthSizeBits
- genfskPacketCfgLengthBitOrd_t GENFSK_packet_config_t::lengthBitOrder
- uint8_t GENFSK_packet_config_t::syncAddrSizeBytes
- int8_t GENFSK_packet_config_t::lengthAdjBytes
- uint8_t GENFSK_packet_config_t::h0SizeBits
- uint8_t GENFSK_packet_config_t::h1SizeBits
- uint16_t GENFSK_packet_config_t::h0Match
- uint16_t GENFSK_packet_config_t::h0Mask
- uint16_t GENFSK_packet_config_t::h1Match
- uint16_t GENFSK_packet_config_t::h1Mask
- genfskCrcComputeMode_t GENFSK_crc_config_t::crcEnable
- genfskCrcRecvInvalid_t GENFSK_crc_config_t::crcRecvInvalid
- uint8_t GENFSK_crc_config_t::crcSize
- uint8_t GENFSK_crc_config_t::crcStartByte
- genfskCrcCfgCrcRefIn_t GENFSK_crc_config_t::crcRefIn
- genfskCrcCfgCrcRefOut_t GENFSK_crc_config_t::crcRefOut
- genfskCrcCfgCrcByteOrd_t GENFSK_crc_config_t::crcByteOrder
- uint32_t GENFSK_crc_config_t::crcSeed
- uint32_t GENFSK_crc_config_t::crcPoly
- uint32_t GENFSK_crc_config_t::crcXorOut
- genfskWhitenMode_t GENFSK_whitener_config_t::whitenEnable
- genfskWhitenStart_t GENFSK_whitener_config_t::whitenStart
- genfskWhitenEnd_t GENFSK_whitener_config_t::whitenEnd
- genfskWhitenB4Crc_t GENFSK_whitener_config_t::whitenB4Crc
- genfskWhitenPolyType_t GENFSK_whitener_config_t::whitenPolyType
- genfskWhitenCfgRefIn_t GENFSK_whitener_config_t::whitenRefIn
- genfskWhitenCfgPayloadReinit_t GENFSK_whitener_config_t::whitenPayloadReinit
- uint8_t GENFSK_whitener_config_t::whitenSize
- uint16_t GENFSK_whitener_config_t::whitenInit
- uint16_t GENFSK_whitener_config_t::whitenPoly
- uint16_t GENFSK_whitener_config_t::whitenSizeThr
- genfskManchesterEn_t GENFSK_whitener_config_t::manchesterEn
- genfskManchesterStart_t GENFSK_whitener_config_t::manchesterStart
- genfskManchesterInv_t GENFSK_whitener_config_t::manchesterInv
- GENFSK_crc_config_t * GENFSK_bitproc_t::crcConfig
- GENFSK_whitener_config_t * GENFSK_bitproc_t::whitenerConfig
- uint16_t GENFSK_packet_header_t::h0Field
- uint16_t GENFSK_packet_header_t::lengthField
- uint16_t GENFSK_packet_header_t::h1Field
- GENFSK_nwk_addr_t GENFSK_packet_t::addr
- GENFSK_packet_header_t GENFSK_packet_t::header
- uint8_t * GENFSK_packet_t::payload

Driver version

- #define GENFSK_DRIVER_VERSION

GENFSK functional Operation

- genfskStatus_t GENFSK_Init (void)

- genfskStatus_t GENFSK_AllocInstance (uint8_t *pInstanceId, GENFSK_radio_config_t *radioConfig, GENFSK_packet_config_t *packetConfig, GENFSK_bitproc_t *bitProcConfig)
- genfskStatus_t GENFSK_RadioConfig (uint8_t instanceId, GENFSK_radio_config_t *radioConfig)
- genfskStatus_t GENFSK_SetPacketConfig (uint8_t instanceId, GENFSK_packet_config_t *packetConfig)
- genfskStatus_t GENFSK_GetPacketConfig (uint8_t instanceId, GENFSK_packet_config_t *packetConfig)
- genfskStatus_t GENFSK_SetCrcConfig (uint8_t instanceId, GENFSK_crc_config_t *crcConfig)
- genfskStatus_t GENFSK_GetCrcConfig (uint8_t instanceId, GENFSK_crc_config_t *crcConfig)
- genfskStatus_t GENFSK_SetWhitenerConfig (uint8_t instanceId, GENFSK_whitener_config_t *whitenerConfig)
- genfskStatus_t GENFSK_GetWhitenerConfig (uint8_t instanceId, GENFSK_whitener_config_t *whitenerConfig)
- genfskStatus_t GENFSK_FreeInstance (uint8_t instanceId)
- void GENFSK_ResetToDefaults (uint8_t instanceId)
- genfskStatus_t GENFSK_SetNetworkAddress (uint8_t instanceId, uint8_t location, GENFSK_nwk_addr_match_t *nwkAddressSettings)
- genfskStatus_t GENFSK_SetEventMask (uint8_t instanceId, uint32_t mask)
- uint32_t GENFSK_GetEventMask (uint8_t instanceId)
- genfskStatus_t GENFSK_GetNetworkAddress (uint8_t instanceId, uint8_t location, GENFSK_nwk_addr_match_t *nwkAddressSettings)
- genfskStatus_t GENFSK_EnableNetworkAddress (uint8_t instanceId, uint8_t location)
- genfskStatus_t GENFSK_DisableNetworkAddress (uint8_t instanceId, uint8_t location)
- genfskStatus_t GENFSK_SetChannelNumber (uint8_t instanceId, uint8_t channelNum)
- uint8_t GENFSK_GetChannelNumber (uint8_t instanceId)
- genfskStatus_t GENFSK_SetTxPowerLevel (uint8_t instanceId, uint8_t txPowerLevel)
- uint8_t GENFSK_GetTxPowerLevel (uint8_t instanceId)
- genfskStatus_t GENFSK_StartTx (uint8_t instanceId, uint8_t *pBuffer, uint16_t bufLengthBytes, GENFSK_timestamp_t txStartTime)
- genfskStatus_t GENFSK_CancelPendingTx (void)
- genfskStatus_t GENFSK_StartRx (uint8_t instanceId, uint8_t *pBuffer, uint16_t maxBufLengthBytes, GENFSK_timestamp_t rxStartTime, GENFSK_timestamp_t rxDuration)
- genfskStatus_t GENFSK_CancelPendingRx (void)
- genfskStatus_t GENFSK_AbortAll (void)
- GENFSK_timestamp_t GENFSK_GetTimestamp (void)
- genfskStatus_t GENFSK_PacketToByteArray (uint8_t instanceId, GENFSK_packet_t *pPacket, uint8_t *pBuffer)
- genfskStatus_t GENFSK_ByteArrayToPacket (uint8_t instanceId, uint8_t *pBuffer, GENFSK_packet_t *pPacket)
- genfskStatus_t GENFSK_RegisterCallbacks (uint8_t instanceId, genfskPacketReceivedCallBack_t packetReceivedCallback, genfskEventNotifyCallBack_t eventCallback)

1.2 Data Structure Documentation

1.2.1 struct GENFSK_nwk_addr_match_t

GENFSK network address matching settings.

Definition of the settings for matching a network address.

Data Fields

uint8_t	nwkAddrSize↩ Bytes	Sync (network) address length in bytes, allowed range is 0..3 representing 1 to 4 bytes long sync addresses.
uint8_t	nwkAddrThr↩ Bits	Sync (network) address matching threshold, number of bits that can mismatch and still be considered a match.
GENFSK_↩ nwk_addr_t	nwkAddr	The network address to be matched.

1.2.2 struct GENFSK_radio_config_t

GENFSK radio configure structure.

Data Fields

genfskRadio↩ Mode_t	radioMode	Radio mode for GENFSK radio. See "genfskRadioMode_t".
genfskData↩ Rate_t	dataRate	Data rate for GENFSK radio. See "genfskDataRate_t".

1.2.3 struct GENFSK_packet_config_t

GENFSK packet format configure structure.

Data Fields

uint8_t	preambleSize↩ Bytes	Preamble length in bytes, allowed range is 0..7 representing 1 to 8 bytes long preambles.
genfskPacket↩ Type_t	packetType	Packet type. See "genfskPacketType_t".
uint8_t	lengthSizeBits	Number of bits in the LENGTH field.
genfskPacket↩ CfgLengthBit↩ Ord_t	lengthBitOrder	Bit order for the LENGTH field of the header. See "genfskPacket↩ CfgLengthBitOrd_t".
uint8_t	syncAddr↩ SizeBytes	Sync (network) address length in bytes, allowed range is 0..3 representing 1 to 4 bytes long sync addresses.
int8_t	lengthAdjBytes	Signed adjustment to the length field for TX and RX. A value of 0 (default) means LENGTH is interpreted as PAYLOAD + CRC.
uint8_t	h0SizeBits	Number of bits in the H0 field.
uint8_t	h1SizeBits	Number of bits in the H1 field.

uint16_t	h0Match	Bits which must match the H0 portion of a received packet for valid packet reception.
uint16_t	h0Mask	Mask to select which bits of H0 must match the h0_match field.
uint16_t	h1Match	Bits which must match the H1 portion of a received packet for valid packet reception.
uint16_t	h1Mask	Mask to select which bits of H1 must match the h1_match field.

1.2.4 struct GENFSK_crc_config_t

GENFSK CRC module configure structure.

Data Fields

genfskCrcComputeMode_t	crcEnable	Software override of the HW-computed CRC for TX. See "genfskCrcComputeMode_t".
genfskCrcRecvInvalid_t	crcRecvInvalid	Receive packets with invalid CRC. See "genfskRecvInvalidCrc_t".
uint8_t	crcSize	Number of CRC octets, allowed range is 0..4.
uint8_t	crcStartByte	Start CRC with this byte position. Byte #0 is the first byte of Sync Address.
genfskCrcCfgCrcRefIn_t	crcRefIn	CRC reflect input. See "genfskCrcCfgCrcRefIn_t".
genfskCrcCfgCrcRefOut_t	crcRefOut	CRC reflect output. See "genfskCrcCfgCrcRefOut_t".
genfskCrcCfgCrcByteOrd_t	crcByteOrder	CRC byte order. See "genfskCrcCfgCrcByteOrd_t".
uint32_t	crcSeed	CRC Seed value. Initial value for CRC LFSR.
uint32_t	crcPoly	CRC Polynomial value.
uint32_t	crcXorOut	XOR mask for CRC result (for no mask, should be 0).

1.2.5 struct GENFSK_whitener_config_t

GENFSK Whitener module configure structure.

Note

Whitening and Manchester encoding are mutually exclusive.

Data Fields

genfsk↩ WhitenMode↩ _t	whitenEnable	Enable/Disable HW (de)whitening on RX and TX packets. See "genfskWhitenMode_t".
genfsk↩ WhitenStart_t	whitenStart	Configure Whitener start point. See "genfskWhitenStart_t".
genfsk↩ WhitenEnd_t	whitenEnd	Configure end-of-whitening. See "genfskWhitenEnd_t".
genfsk↩ WhitenB4Crc↩ _t	whitenB4Crc	Configure for whitening-before-CRC. See "genfskWhitenB4Crc↩ _t".
genfsk↩ WhitenPoly↩ Type_t	whitenPoly↩ Type	Whiten polynomial type. See "genfskWhitenPolyType_t".
genfsk↩ WhitenCfg↩ RefIn_t	whitenRefIn	Whiten reflect input. See "genfskWhitenCfgRefIn_t".
genfsk↩ WhitenCfg↩ Payload↩ Reinit_t	whiten↩ PayloadReinit	Configure for whitener re-initialization. See "genfskWhitenCfg↩ PayloadReinit_t.
uint8_t	whitenSize	Length of whitener LFSR. Maximum value 9.
uint16_t	whitenInit	Initialization value for Whitening/De-whitening. Maximum 9 bits.
uint16_t	whitenPoly	Whitener polynomial. The polynomial value must be right-justified if smaller than 9-bits. Maximum 9 bits.
uint16_t	whitenSizeThr	Whitener size threshold. Maximum packet length required to enable whiten. Requires WHITEN_START 2 or 3.
genfsk↩ Manchester↩ En_t	manchesterEn	Configure for Manchester Encoding/Decoding. See "genfsk↩ ManchesterEn_t".
genfsk↩ Manchester↩ Start_t	manchester↩ Start	Configure for inverted Manchester Encoding. See "genfsk↩ ManchesterStart_t".
genfsk↩ Manchester↩ Inv_t	manchesterInv	Configure Manchester Encoding start point. See "genfsk↩ ManchesterInv_t".

1.2.6 struct GENFSK_bitproc_t

GENFSK bitstream processing configuration.

Configuration of the bitstream processing to be done for GENFSK.

Data Fields

GENFSK_↩ crc_config_t *	crcConfig	CRC module configuration structure.
GENFSK_↩ whitener_↩ config_t *	whitenerConfig	Whitener module configuration structure.

1.2.7 struct GENFSK_packet_header_t

Data structure for GENFSK packet header.

Data Fields

uint16_t	h0Field	H0 field value.
uint16_t	lengthField	LENGTH field value.
uint16_t	h1Field	H1 field value.

1.2.8 struct GENFSK_packet_t

Data structure for GENFSK packet.

Data Fields

GENFSK_↩ nwk_addr_t	addr	Network address.
GENFSK_↩ packet_↩ header_t	header	Packet header data structure.
uint8_t *	payload	Payload (+ CRC if reception) buffer.

1.3 Macro Definition Documentation

1.3.1 #define GENFSK_DRIVER_VERSION

GENFSK Link Layer driver version 0.0.1.

1.3.2 #define gGENFSK_IrqPriority_c

GENFSK Protocol Engine interrupt.

GENFSK Protocol Engine interrupt priority.

1.3.3 #define gGENFSK_TaskStackSize_c

GENFSK LL Task stack size.

1.3.4 #define gGENFSK_TaskPriority_c

GENFSK LL Task priority.

1.3.5 #define gGENFSK_InstancesCnt_c

GENFSK LL total number of available instances.

1.3.6 #define gGENFSK_InvalidIdx_c

GENFSK LL invalid instance ID.

1.4 Typedef Documentation

1.4.1 typedef uint32_t GENFSK_nwk_addr_t

GENFSK network address type.

Network address.

Note

The LS bytes of this type are used when network address length is less than 4 bytes.

1.4.2 typedef uint64_t GENFSK_timestamp_t

GENFSK timestamp type.

64 bits of timestamp.

Note

The timestamp based on a 1us timer tick.

1.4.3 typedef void(∗ genfskPacketReceivedCallBack_t) (uint8_t ∗pBuffer, uint16_t bufferLength, uint64_t timestamp, uint8_t rssi, uint8_t crcValid)

Packet Received callback function pointer type.

Parameters

pBuffer	The pointer to the buffer used for reception.
packet_length	The allocated pkt_buffer size for the maximum packet length that can be received.
timestamp	The timestamp for the received packet in microseconds.
rssi	The RSSI for the received packet.
crcValid	If set, the CRC for the received packet is valid. Else CRC is invalid.

1.4.4 typedef void(* genfskEventNotifyCallBack_t) (genfskEvent_t event, genfskEventStatus_t eventStatus)

Event notification callback function pointer type.

Parameters

event	Reason the callback is being invoked. See "genfskEvent_t".
eventStatus	The status of the event. See "genfskEventStatus_t".

1.5 Enumeration Type Documentation

1.5.1 enum genfskDataRate_t

GENFSK Data Rate selections.

Enumerator

gGenfskDR1Mbps GENFSK 1 MBit datarate.
gGenfskDR500Kbps GENFSK 500 KBit datarate.
gGenfskDR250Kbps GENFSK 250 KBit datarate.

1.5.2 enum genfskRadioMode_t

GENFSK Radio Mode selections.

Enumerator

gGenfskGfskBt0p5h0p5 BT=0.5, h=0.5 [BLE at 1MBPS data rate; CS4 at 250KBPS data rate].
gGenfskGfskBt0p5h0p32 BT=0.5, h=0.32.
gGenfskGfskBt0p5h0p7 BT=0.5, h=0.7 [CS1 at 500KBPS data rate].
gGenfskGfskBt0p5h1p0 BT=0.5, h=1.0 [CS4 at 250KBPS data rate].
gGenfskGfskBt0p3h0p5 BT=0.3, h=0.5 [CS2 at 1MBPS data rate].
gGenfskGfskBt0p7h0p5 BT=0.7, h=0.5.
gGenfskFsk FSK.
gGenfskMsk MSK.

1.5.3 enum genfskStatus_t

Error codes for the GENFSK driver.

Enumerator

gGenfskSuccess_c Execution successful.
gGenfskInvalidParameters_c Invalid parameters.
gGenfskFail_c Execution failure.
gGenfskNotInitialized_c The GENFSK module was not initialized.
gGenfskAlreadyInit_c Issued by GENFSK_AllocInstance() if the GENFSK module is already initialized.
gGenfskBusyRx_c Transceiver has an active RX sequence.
gGenfskBusyTx_c Transceiver has an active TX sequence.
gGenfskBusyPendingRx_c Transceiver has a pending RX sequence.
gGenfskBusyPendingTx_c Transceiver has a pending TX sequence.
gGenfskInstantPassed_c Issued by GENFSK_StartRx()/GENFSK_StartTx() when an event is programmed too close or in the past.
gGenfskAllocInstanceFailed Issued by GENFSK_AllocInstance() when the instance allocation failed.

1.5.4 enum genfskEvent_t

GENFSK notification events.

Enumerator

gGenfskTxEvent TX sequence has completed with a successful packet transmission.
gGenfskRxEvent RX sequence has completed with a successful packet reception.
gGenfskNwkAddressMatch Network address match has occurred.
gGenfskWakeEvent The SLEEP_TMR has matched GENERIC_FSK_WAKE and DSM exited.
gGenfskAllEvents All events.

1.5.5 enum genfskEventStatus_t

GENFSK notification events status.

Enumerator

gGenfskSuccess Success status.
gGenfskRxAllocLengthFail Allocated RX buffer length is smaller than the received packet length.
gGenfskTimeout RX sequence timeout.
gGenfskSyncLost RX/TX PLL unlock.

gGenfskCRCInvalid CRC invalid for RX packet. Promiscuous mode only!
gGenfskH0Fail H0 violated status. Promiscuous mode only!
gGenfskH1Fail H1 violated status. Promiscuous mode only!
gGenfskLengthFail Length field violated status. Promiscuous mode only!

1.5.6 enum genfskPacketCfgLengthBitOrd_t

LENGTH_BIT_ORD bit definitions.

Enumerator

gGenfskLengthBitLsbFirst Bit order of the LENGTH field of the header LSB first.
gGenfskLengthBitMsbFirst Bit order of the LENGTH field of the header MSB first.

1.5.7 enum genfskCrcComputeMode_t

CRC enable bit definitions.

Enumerator

gGenfskCrcDisable CRC functionality disabled.
gGenfskCrcEnable CRC functionality enabled.

1.5.8 enum genfskCrcRecvInvalid_t

Enumerator

gGenfskCrcSupressInvalid Supress reception of packets with invalid CRC reception.
gGenfskCrcRecvInvalid Receive packets with invalid CRC.

1.5.9 enum genfskCrcCfgCrcRefIn_t

CRC_REF_IN bit definitions.

Enumerator

gGenfskCrcInputNoRef Do not manipulate input data stream.
gGenfskCrcRefInput Reflect each byte in the input stream bitwise.

1.5.10 enum genfskCrcCfgCrcRefOut_t

CRC_REF_OUT bit definitions.

Enumerator

gGenfskCrcOutputNoRef Do not manipulate CRC result.
gGenfskCrcRefOutput CRC result is to be reflected bitwise (operated on entire word).

1.5.11 enum genfskCrcCfgCrcByteOrd_t

CRC_BYTE_ORD bit definitions.

Enumerator

gGenfskCrcLSByteFirst Byte order of the CRC LS Byte first.
gGenfskCrcMSByteFirst Bit order of the CRC MS Byte first.

1.5.12 enum genfskWhitenMode_t

Whitener enable bit definitions.

Enumerator

gGenfskWhitenDisable Whitener functionality disabled.
gGenfskWhitenEnable Whitener functionality enabled.

1.5.13 enum genfskWhitenStart_t

WHITEN_START bit definitions.

Enumerator

gWhitenStartNoWhitening No whitening.
gWhitenStartWhiteningAtH0 Start whitening at start-of-H0.
gWhitenStartWhiteningAtH1 Start whitening at start-of-H1 but only if LENGTH > WHITEN_SZ_THR.
gWhitenStartWhiteningAtPayload Start whitening at start-of-payload but only if LENGTH > WHITEN_SZ_THR.

1.5.14 enum genfskWhitenEnd_t

WHITEN_END bit definitions.

Enumerator

gWhitenEndAtEndOfPayload End whiten at end-of-payload.
gWhitenEndAtEndOfCrc End whiten at end-of-CRC.

1.5.15 enum genfskWhitenB4Crc_t

WHITEN_B4_CRC bit definitions.

Enumerator

gCrcB4Whiten CRC before whiten/de-whiten.
gWhitenB4Crc Whiten/de-whiten before CRC.

1.5.16 enum genfskWhitenPolyType_t

WHITEN_POLY_TYPE bit definitions.

Enumerator

gGaloisPolyType A Galois type LFSR is used with the whiten polynomial.
gFibonnaciPolyType A Fibonacci type LFSR is used with the whiten polynomial.

1.5.17 enum genfskWhitenCfgRefIn_t

WHITEN_REF_IN bit definitions.

Note

The input data stream is reflected bit-wise, per byte. Bit 7 becomes bit 0, bit 6 becomes bit 1, etc. Will only cause the reflection of the payload data bits as they are used in the whiten calculation and will not cause any change in the output bit order.

Enumerator

gGenfskWhitenInputNoRef Do not manipulate input data stream.
gGenfskWhitenRefInput Reflect each byte in the input stream bitwise.

1.5.18 enum genfskWhitenCfgPayloadReinit_t

WHITEN_PAYLOAD_REINIT bit definitions.

Enumerator

gGenfskWhitenNoPayloadReinit Do not re-initialize whitener LFSR at start-of-payload.
gGenfskWhitenPayloadReinit Re-initialize whitener LFSR at start-of-payload.

1.5.19 enum genfskManchesterEn_t

MANCHESTER_EN bit definitions.

Enumerator

gGenfskManchesterDisable Disable Manchester encoding (TX) and decoding (RX).
gGenfskManchesterEnable Enable Manchester encoding (TX) and decoding (RX).

1.5.20 enum genfskManchesterInv_t

MANCHESTER_INV bit definitions.

Enumerator

gGenfskManchesterNoInv Manchester coding as per 802.3.
gGenfskManchesterInverted Manchester coding as per 802.3 but with the encoding signal inverted.

1.5.21 enum genfskManchesterStart_t

MANCHESTER_START bit definitions.

Enumerator

gGenfskManchesterStartAtPayload Start Manchester coding at start-of-payload.
gGenfskManchesterStartAtHeader Start Manchester coding at start-of-header.

1.5.22 enum genfskPacketType_t

Data packet type bit definitions.

Enumerator

gGenfskFormattedPacket The packets sent or received are formatted, all HW accelerations are available.

gGenfskRawPacket The packets sent or received are RAW, all HW acceleration is bypassed (limited to 35bytes of payload).

1.6 Function Documentation

1.6.1 genfskStatus_t GENFSK_Init (void)

Initializes the GENFSK LL.

This function initializes the GENFSK LL.

Return values

gGenfskSuccess_c	if success or the failure reason.

1.6.2 genfskStatus_t GENFSK_AllocInstance (uint8_t * *pInstanceId,* GENFSK_radio_config_t * *radioConfig,* GENFSK_packet_config_t * *packetConfig,* GENFSK_bitproc_t * *bitProcConfig*)

Allocates a GENFSK LL instance.

This function allocates the GENFSK LL module and initilizes the instance according to the protocol and processing chain settings.

Parameters

pInstanceId	The pointer which will save the allocated instance. gGENFSK_InvalidIdx_c if the allocation failed.
radioConfig	The radio configuration for which the GENFSK LL should be configured.
packetConfig	The packet configuration for which the GENFSK LL should be configured.
bitProcConfig	The bitstream processing for which the GENFSK LL should be configured.

Return values

gGenfskSuccess_c	if success or the failure reason.

Warning

Should be called after GENFSK_Init();

1.6.3 genfskStatus_t GENFSK_RadioConfig (uint8_t *instanceId,* GENFSK_radio_config_t * *radioConfig*)

Sets the radio configuration for the current GENFSK LL instance.

This function initialize the radio and sets the radio configuration.

Parameters

instanceId	The ID of the instance.
radioConfig	The radio configuration to be set for GENFSK LL.

Return values

gGenfskSuccess_c	if success or the failure reason.

1.6.4 genfskStatus_t GENFSK_SetPacketConfig (uint8_t *instanceId,* GENFSK_packet_config_t * *packetConfig*)

Sets the packet configuration for the current GENFSK LL instance.

This function sets the packet configuration.

Parameters

instanceId	The ID of the instance.
packetConfig	The packet configuration to be set in GENFSK LL.

Return values

gGenfskSuccess_c	if success or the failure reason.

1.6.5 genfskStatus_t GENFSK_GetPacketConfig (uint8_t *instanceId,* GENFSK_packet_config_t * *packetConfig*)

Returns the packet configuration currently set in GENFSK LL.

This function returns the packet configuration currently set in GENFSK LL.

Parameters

instanceId	The ID of the instance.
packetConfig	The stored packet configuration.

Return values

gGenfskSuccess_c	if success or the failure reason.

1.6.6 genfskStatus_t GENFSK_SetCrcConfig (uint8_t *instanceId,* GENFSK_crc_config_t * *crcConfig*)

Sets the CRC configuration for the current GENFSK LL instance.

This function sets the CRC configuration.

Parameters

instanceId	The ID of the instance.
crcConfig	The CRC configuration to be set in GENFSK LL.

Return values

gGenfskSuccess_c	if success or the failure reason.

1.6.7 genfskStatus_t GENFSK_GetCrcConfig (uint8_t *instanceId,* GENFSK_crc_config_t * *crcConfig*)

Returns the CRC configuration currently set in GENFSK LL.

This function returns the CRC configuration currently set in GENFSK LL.

Parameters

instanceId	The ID of the instance.
crcConfig	The stored CRC configuration.

Return values

gGenfskSuccess_c	if success or the failure reason.

1.6.8 genfskStatus_t GENFSK_SetWhitenerConfig (uint8_t *instanceId,* GENFSK_whitener_config_t * *whitenerConfig*)

Sets whitening configuration for the current GENFSK LL instance.

This function sets the whitening configuration.

Parameters

instanceId	The ID of the instance.
whitenerConfig	The whitening configuration to be set in GENFSK LL.

Return values

gGenfskSuccess_c	if success or the failure reason.

1.6.9 genfskStatus_t GENFSK_GetWhitenerConfig (uint8_t *instanceId,* GENFSK_whitener_config_t * *whitenerConfig*)

Returns the whitening configuration currently set in GENFSK LL.

This function returns the whitening configuration currently set in GENFSK LL.

Parameters

instanceId	The ID of the instance.
whitenerConfig	The stored whitening configuration.

Return values

gGenfskSuccess_c	if success or the failure reason.

1.6.10 genfskStatus_t GENFSK_FreeInstance (uint8_t *instanceId*)

De-initializes the GENFSK LL instance.

This function sets all GENFSK registers values to reset values and disables GENFSK LL interrupt if no other instance is initialized.

Parameters

instanceId	The ID of the instance.

Return values

gGenfskSuccess_c	if success or the failure reason.

1.6.11 void GENFSK_ResetToDefaults (uint8_t *instanceId*)

Reset the GENFSK LL to default values.

This function reset the GENFSK LL registers values to the reset values.

Parameters

instanceId	The ID of the instance.

1.6.12 genfskStatus_t GENFSK_SetNetworkAddress (uint8_t *instanceId,* uint8_t *location,* GENFSK_nwk_addr_match_t * *nwkAddressSettings*)

Controls setting one of the network address match locations.

This function set the network address matching.

Parameters

instanceId	The ID of the instance.
location	the location number to set, valid range is 0..3. This location will be enabled if there are no errors during the setting process.
nwkAddress↩ Settings	the settings to be applied.

Return values

gGenfskSuccess_c	if success or the failure reason.

1.6.13 genfskStatus_t GENFSK_SetEventMask (uint8_t *instanceId,* uint32_t *mask*)

Sets the event mask for genfskEventNotifyCallBack_t callback.

Sets the event mask for genfskEventNotifyCallBack_t.

Parameters

instanceId	The ID of the instance.
mask	The event mask specifies which notification events are sent by genfskEventNotify↩ CallBack_t. See "genfskEvent_t".

Return values

gGenfskSuccess_c	if success or the failure reason.

1.6.14 uint32_t GENFSK_GetEventMask (uint8_t *instanceId*)

Returns the event mask for genfskEventNotifyCallBack_t callback.

Parameters

instanceId	The ID of the instance. Returns the current enabled events for genfskEventNotify↩ CallBack_t.

Return values

genfskEvent_t.	

1.6.15 genfskStatus_t GENFSK_GetNetworkAddress (uint8_t *instanceId,* uint8_t *location,* GENFSK_nwk_addr_match_t * *nwkAddressSettings*)

Returns the network address set at location.

This function enables setting the network address matching.

Parameters

instanceId	The ID of the instance.
location	the location number to set, valid range is 0..3. This location will be enabled if there are no errors during the setting process.
nwkAddress↩ Settings	the stored network address settings at the specified location.

Return values

gGenfskSuccess_c	if success or the failure reason.

1.6.16 genfskStatus_t GENFSK_EnableNetworkAddress (uint8_t *instanceId,* uint8_t *location*)

Controls enabling one of the network address match locations.

This function enables one of the network address matching.

Parameters

instanceId	The ID of the instance.
location	the location number to disable, valid range is 0..3. This location will be enabled if there are no errors during the setting process.

Return values

gGenfskSuccess_c	if success or the failure reason.

1.6.17 genfskStatus_t GENFSK_DisableNetworkAddress (uint8_t *instanceId,* uint8_t *location*)

Controls disabling one of the network address match locations.

This function disables one of the network address matching.

Parameters

instanceId	The ID of the instance.
location	the location number to disable, valid range is 0..3. This location will be disabled if there are no errors during the setting process.

Return values

gGenfskSuccess_c	if success or the failure reason.

1.6.18 genfskStatus_t GENFSK_SetChannelNumber (uint8_t *instanceId,* uint8_t *channelNum*)

Sets the channel number.

This function sets the channel number.

Parameters

instanceId	The ID of the instance.
channelNum	The channel number on which to transmit and receive, 0 <= channelNum <= 127; Formula: F = (2360 + channelNum) [in MHz].

Return values

gGenfskSuccess_c	if success or the failure reason.

1.6.19 uint8_t GENFSK_GetChannelNumber (uint8_t *instanceId*)

Returns the channel number currently set in GENFSK LL.

This function returns the channel number currently set in GENFSK LL.

Parameters

instanceId	The ID of the instance.

Return values

The	channel number currently set, 0 <= channelNum <= 127; Formula: F = (2360 + channelNum) [in MHz].

1.6.20 genfskStatus_t GENFSK_SetTxPowerLevel (uint8_t *instanceId,* uint8_t *txPowerLevel*)

Sets the power level for transmission.

This function sets power level for transmission.

Parameters

instanceId	The ID of the instance.
txPowerLevel	The power level for transmission, 0 <= txPowerLevel <= 32.

Return values

gGenfskSuccess_c	if success or the failure reason.

1.6.21 uint8_t GENFSK_GetTxPowerLevel (uint8_t *instanceId*)

Returns the power level currently set in GENFSK LL.

Parameters

instanceId	The ID of the instance. This function returns the power level currently set in GENFSK LL.

Return values

The	power level for transmission, 0 <= txPowerLevel <= 32.

1.6.22 genfskStatus_t GENFSK_StartTx (uint8_t *instanceId,* uint8_t * *pBuffer,* uint16_t *bufLengthBytes,* GENFSK_timestamp_t *txStartTime*)

Performs a transmission.

This function performs a transmission of GENFSK LL Packet.

Parameters

instanceId	The ID of the instance.
pBuffer	The pointer to a buffer containing the packet body compliant to the previously configured settings.
bufLengthBytes	The buffer length in bytes.
txStartTime	The time at which to start transmission. Set 0 for immediate transmission.

Return values

gGenfskSuccess_c	if success or the failure reason.

Warning

Timebase roll over at 24 bits (∼16.7 seconds) must be considered in setting the txStartTime.

1.6.23 genfskStatus_t GENFSK_CancelPendingTx (void)

Cancels pending TX events.

This function cancels pending TX events for the current active instance but do not abort a TX-in-progress.

Return values

gGenfskSuccess_c	if success or the failure reason.

1.6.24 genfskStatus_t GENFSK_StartRx (uint8_t *instanceId,* uint8_t * *pBuffer,* uint16_t *maxBufLengthBytes,* GENFSK_timestamp_t *rxStartTime,* GENFSK_timestamp_t *rxDuration*)

Performs a receive operation.

This function performs a receive operation.

Parameters

instanceId	The ID of the instance.
pBuffer	The pointer to a buffer used for reception.
maxBuf↩ LengthBytes	The allocated pBuffer size for the maximum packet length that can be received.

rxStartTime	The time at which to start receive. Set 0 for immediate receive.
rxDuration	The duration of the receive operation. Set 0 for continuous reception.

Return values

gGenfskSuccess_c	if success or the failure reason.

Warning

Timebase roll over at 24 bits (∼16.7 seconds) must be considered in setting the rxStartTime.

1.6.25 genfskStatus_t GENFSK_CancelPendingRx (void)

Cancels pending RX events.

This function cancels pending RX events for the current active instance but do not abort a RX-in-progress.

Return values

gGenfskSuccess_c	if success or the failure reason.

1.6.26 genfskStatus_t GENFSK_AbortAll (void)

Cancels all pending events.

This function cancels all pending events for the current active instance and abort any sequence-in-progress.

Return values

gGenfskSuccess_c	if success or the failure reason.

1.6.27 GENFSK_timestamp_t GENFSK_GetTimestamp (void)

Fetches the current value of the GENFSK LL timebase.

This function fetches the current value of the timebase for the LL.

Return values

The	value of the timebase, in microseconds.

Warning

Any use of the timestamp value must allow for processing delays.

1.6.28 genfskStatus_t GENFSK_PacketToByteArray (uint8_t *instanceId*, GENFSK_packet_t * *pPacket*, uint8_t * *pBuffer*)

Converts a packet buffer to a byte array format to be sent by GENFSK LL.

This function is used before GENFSK_StartTx() in order to convert the formatted packet in a byte array to be sent over the air. The byte array will have the format : NWK_ADDRESS | H0 | LENGTH | H1 | PAYLOAD | CRC

Parameters

instanceId	The ID of the instance for which the packet to be formatted.
pPacket	Pointer to the packet structure to be formatted.
pBuffer	Pointer to the byte array formatted buffer.

Return values

gGenfskSuccess_c	if success or the failure reason.

1.6.29 genfskStatus_t GENFSK_ByteArrayToPacket (uint8_t *instanceId*, uint8_t * *pBuffer*, GENFSK_packet_t * *pPacket*)

Converts a received byte array formatted packet in GENFSK_packet_t format.

This function is used after a packet is received in order to convert the byte array received over the air in GENFSK_packet_t format.

Parameters

instanceId	The ID of the instance for which the packet to be formatted.
pBuffer	Pointer to the byte array formatted buffer.
pPacket	Pointer to the packet structure to store the formatted packet.

Return values

gGenfskSuccess_c	if success or the failure reason.

1.6.30 genfskStatus_t GENFSK_RegisterCallbacks (uint8_t *instanceId*, genfskPacketReceivedCallBack_t *packetReceivedCallback*, genfskEventNotifyCallBack_t *eventCallback*)

Registers the callback functions packet received and event notifications.

This function register the callback functions for packet received and event notifications.

Parameters

instanceId	The ID of the instance.
packetReceivedCallback	Packet received callback.
eventCallback	Event callback.

Return values

gGenfskSuccess_c	if success or the failure reason.

1.7 Variable Documentation

1.7.1 uint8_t GENFSK_nwk_addr_match_t::nwkAddrSizeBytes

Sync (network) address length in bytes, allowed range is 0..3 representing 1 to 4 bytes long sync addresses.

1.7.2 uint8_t GENFSK_nwk_addr_match_t::nwkAddrThrBits

Sync (network) address matching threshold, number of bits that can mismatch and still be considered a match.

1.7.3 GENFSK_nwk_addr_t GENFSK_nwk_addr_match_t::nwkAddr

The network address to be matched.

1.7.4 genfskRadioMode_t GENFSK_radio_config_t::radioMode

Radio mode for GENFSK radio.

See "genfskRadioMode_t".

1.7.5 genfskDataRate_t GENFSK_radio_config_t::dataRate

Data rate for GENFSK radio.

See "genfskDataRate_t".

1.7.6 uint8_t GENFSK_packet_config_t::preambleSizeBytes

Preamble length in bytes, allowed range is 0..7 representing 1 to 8 bytes long preambles.

1.7.7 genfskPacketType_t GENFSK_packet_config_t::packetType

Packet type.
See "genfskPacketType_t".

1.7.8 uint8_t GENFSK_packet_config_t::lengthSizeBits

Number of bits in the LENGTH field.

1.7.9 genfskPacketCfgLengthBitOrd_t GENFSK_packet_config_t::lengthBitOrder

Bit order for the LENGTH field of the header.
See "genfskPacketCfgLengthBitOrd_t".

1.7.10 uint8_t GENFSK_packet_config_t::syncAddrSizeBytes

Sync (network) address length in bytes, allowed range is 0..3 representing 1 to 4 bytes long sync addresses.

1.7.11 int8_t GENFSK_packet_config_t::lengthAdjBytes

Signed adjustment to the length field for TX and RX.
A value of 0 (default) means LENGTH is interpreted as PAYLOAD + CRC.

1.7.12 uint8_t GENFSK_packet_config_t::h0SizeBits

Number of bits in the H0 field.

1.7.13 uint8_t GENFSK_packet_config_t::h1SizeBits

Number of bits in the H1 field.

1.7.14 uint16_t GENFSK_packet_config_t::h0Match

Bits which must match the H0 portion of a received packet for valid packet reception.

1.7.15 uint16_t GENFSK_packet_config_t::h0Mask

Mask to select which bits of H0 must match the h0_match field.

1.7.16 uint16_t GENFSK_packet_config_t::h1Match

Bits which must match the H1 portion of a received packet for valid packet reception.

1.7.17 uint16_t GENFSK_packet_config_t::h1Mask

Mask to select which bits of H1 must match the h1_match field.

1.7.18 genfskCrcComputeMode_t GENFSK_crc_config_t::crcEnable

Software override of the HW-computed CRC for TX.
See "genfskCrcComputeMode_t".

1.7.19 genfskCrcRecvInvalid_t GENFSK_crc_config_t::crcRecvInvalid

Receive packets with invalid CRC.
See "genfskRecvInvalidCrc_t".

1.7.20 uint8_t GENFSK_crc_config_t::crcSize

Number of CRC octets, allowed range is 0..4.

1.7.21 uint8_t GENFSK_crc_config_t::crcStartByte

Start CRC with this byte position.
Byte #0 is the first byte of Sync Address.

1.7.22 genfskCrcCfgCrcRefIn_t GENFSK_crc_config_t::crcRefIn

CRC reflect input.
See "genfskCrcCfgCrcRefIn_t".

1.7.23 genfskCrcCfgCrcRefOut_t GENFSK_crc_config_t::crcRefOut

CRC reflect output.

See "genfskCrcCfgCrcRefOut_t".

1.7.24 genfskCrcCfgCrcByteOrd_t GENFSK_crc_config_t::crcByteOrder

CRC byte order.

See "genfskCrcCfgCrcByteOrd_t".

1.7.25 uint32_t GENFSK_crc_config_t::crcSeed

CRC Seed value.

Initial value for CRC LFSR.

1.7.26 uint32_t GENFSK_crc_config_t::crcPoly

CRC Polynomial value.

1.7.27 uint32_t GENFSK_crc_config_t::crcXorOut

XOR mask for CRC result (for no mask, should be 0).

1.7.28 genfskWhitenMode_t GENFSK_whitener_config_t::whitenEnable

Enable/Disable HW (de)whitening on RX and TX packets.

See "genfskWhitenMode_t".

1.7.29 genfskWhitenStart_t GENFSK_whitener_config_t::whitenStart

Configure Whitener start point.

See "genfskWhitenStart_t".

1.7.30 genfskWhitenEnd_t GENFSK_whitener_config_t::whitenEnd

Configure end-of-whitening.

See "genfskWhitenEnd_t".

1.7.31 genfskWhitenB4Crc_t GENFSK_whitener_config_t::whitenB4Crc

Configure for whitening-before-CRC.

See "genfskWhitenB4Crc_t".

1.7.32 genfskWhitenPolyType_t GENFSK_whitener_config_t::whitenPolyType

Whiten polynomial type.

See "genfskWhitenPolyType_t".

1.7.33 genfskWhitenCfgRefIn_t GENFSK_whitener_config_t::whitenRefIn

Whiten reflect input.

See "genfskWhitenCfgRefIn_t".

1.7.34 genfskWhitenCfgPayloadReinit_t GENFSK_whitener_config_t::whitenPayloadReinit

Configure for whitener re-initialization.

See "genfskWhitenCfgPayloadReinit_t.

1.7.35 uint8_t GENFSK_whitener_config_t::whitenSize

Length of whitener LFSR.

Maximum value 9.

1.7.36 uint16_t GENFSK_whitener_config_t::whitenInit

Initialization value for Whitening/De-whitening.

Maximum 9 bits.

1.7.37 uint16_t GENFSK_whitener_config_t::whitenPoly

Whitener polynomial.

The polynomial value must be right-justified if smaller than 9-bits. Maximum 9 bits.

1.7.38 uint16_t GENFSK_whitener_config_t::whitenSizeThr

Whitener size threshold.

Maximum packet length required to enable whiten. Requires WHITEN_START 2 or 3.

1.7.39 genfskManchesterEn_t GENFSK_whitener_config_t::manchesterEn

Configure for Manchester Encoding/Decoding.

See "genfskManchesterEn_t".

1.7.40 genfskManchesterStart_t GENFSK_whitener_config_t::manchesterStart

Configure for inverted Manchester Encoding.

See "genfskManchesterStart_t".

1.7.41 genfskManchesterInv_t GENFSK_whitener_config_t::manchesterInv

Configure Manchester Encoding start point.

See "genfskManchesterInv_t".

1.7.42 GENFSK_crc_config_t* GENFSK_bitproc_t::crcConfig

CRC module configuration structure.

1.7.43 GENFSK_whitener_config_t* GENFSK_bitproc_t::whitenerConfig

Whitener module configuration structure.

1.7.44 uint16_t GENFSK_packet_header_t::h0Field

H0 field value.

1.7.45 uint16_t GENFSK_packet_header_t::lengthField

LENGTH field value.

1.7.46 uint16_t GENFSK_packet_header_t::h1Field

H1 field value.

1.7.47 GENFSK_nwk_addr_t GENFSK_packet_t::addr

Network address.

1.7.48 GENFSK_packet_header_t GENFSK_packet_t::header

Packet header data structure.

1.7.49 uint8_t* GENFSK_packet_t::payload

Payload (+ CRC if reception) buffer.

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.