## **Generic FSK Link Layer**

## **API Reference Manual**

Rev. 14 Jun 2020



## **Contents**

## Chapter 1 Genfsk

1.1	Overview
1.2	Data Structure Documentation
1.2.1	struct GENFSK_nwk_addr_match_t
1.2.2	struct GENFSK_TimeEvent_t
1.2.3	struct GENFSK_radio_config_t
1.2.4	struct GENFSK_packet_config_t
1.2.5	struct GENFSK_crc_config_t
1.2.6	struct GENFSK_whitener_config_t
1.2.7	struct GENFSK_bitproc_t
1.2.8	struct GENFSK_packet_header_t
1.2.9	struct GENFSK_packet_t
1.3	Macro Definition Documentation
1.3.1	gGenFskVerMajor_c
1.3.2	GENFSK_DRIVER_VERSION
1.3.3	gGENFSK_IrqPriority_c
1.3.4	gGENFSK_TaskStackSize_c
1.3.5	gGENFSK_TaskPriority_c
1.3.6	gGENFSK_InstancesCnt_c
1.3.7	gGENFSK_InvalidIdx_c
1.3.8	gGENFSK_XcvrInitRetryCount_c
1.3.9	gGENFSK_InvalidTimerId_c
1.3.10	gGENFSK_NoMwsControl_c
1.3.11	gGENFSK_LLMwsControl_c
1.3.12	gGENFSK_AppMwsControl_c
1.3.13	gGENFSK_MwsControl_c
1.4	Typedef Documentation
1.4.1	GENFSK_nwk_addr_t
1.4.2	GENFSK_timestamp_t
1.4.3	genfskTimerId_t
1.4.4	genfskTimeCallback_t
1.4.5	genfskPacketReceivedCallBack_t

Section nur	mber Title	Page
1.4.6	genfskEventNotifyCallBack_t	13
1.5	Enumeration Type Documentation	13
1.5.1	genfskDataRate_t	13
1.5.2	genfskRadioMode_t	13
1.5.3	genfskStatus_t	14
1.5.4	genfskEvent_t	14
1.5.5	genfskEventStatus_t	14
1.5.6	genfskPacketCfgLengthBitOrd_t	15
1.5.7	genfskCrcComputeMode_t	15
1.5.8	genfskCrcRecvInvalid_t	15
1.5.9	genfskCrcCfgCrcRefIn_t	15
1.5.10	genfskCrcCfgCrcRefOut_t	16
1.5.11	genfskCrcCfgCrcByteOrd_t	16
1.5.12	genfskWhitenMode_t	
1.5.13	genfskWhitenStart_t	16
1.5.14	genfskWhitenEnd_t	17
1.5.15	genfskWhitenB4Crc_t	17
1.5.16	genfskWhitenPolyType_t	17
1.5.17	genfskWhitenCfgRefIn_t	17
1.5.18	genfskWhitenCfgPayloadReinit_t	18
1.5.19	genfskManchesterEn_t	
1.5.20	genfskManchesterInv_t	18
1.5.21	genfskManchesterStart_t	18
1.5.22	genfskPacketType_t	19
1.6	Function Documentation	19
1.6.1	GENFSK_Init(void)	
1.6.2	GENFSK_AllocInstance(uint8_t *pInstanceId, GENFSK_radio_config_←	
	t *radioConfig, GENFSK_packet_config_t *packetConfig, GENFSK_bitproc_t	
	*bitProcConfig)	19
1.6.3	GENFSK_RadioConfig(uint8_t instanceId, GENFSK_radio_config_t *radio←	
	Config)	20
1.6.4	GENFSK_SetPacketConfig(uint8_t instanceId, GENFSK_packet_config_←	
	t *packetConfig)	20
1.6.5	GENFSK_GetPacketConfig(uint8_t instanceId, GENFSK_packet_config_←	
	t *packetConfig)	20
1.6.6	GENFSK_SetCrcConfig(uint8_t instanceId, GENFSK_crc_config_t *crcConfig) .	21
1.6.7	GENFSK_GetCrcConfig(uint8_t instanceId, GENFSK_crc_config_t *crcConfig).	21
1.6.8	GENFSK_SetWhitenerConfig(uint8_t instanceId, GENFSK_whitener_config_	
	t *whitenerConfig)	21
1.6.9	GENFSK_GetWhitenerConfig(uint8_t instanceId, GENFSK_whitener_config_ \infty	
	t *whitenerConfig)	22
1.6.10	GENFSK_FreeInstance(uint8_t instanceId)	
1.6.11	GENFSK_ResetToDefaults(uint8_t instanceId)	

NXP Semiconductors v

Section number	Title	Page
1.7.10	lengthSizeBits	. 33
1.7.11	lengthBitOrder	. 33
1.7.12	syncAddrSizeBytes	. 33
1.7.13	lengthAdjBytes	. 33
1.7.14	hOSizeBits	. 33
1.7.15	h1SizeBits	. 33
1.7.16	h0Match	. 33
1.7.17	h0Mask	. 33
1.7.18	h1Match	. 34
1.7.19	h1Mask	. 34
1.7.20	crcEnable	. 34
1.7.21	crcRecvInvalid	. 34
1.7.22	crcSize	. 34
1.7.23	crcStartByte	. 34
1.7.24	crcRefIn	. 34
1.7.25	crcRefOut	. 34
1.7.26	crcByteOrder	. 35
1.7.27	crcSeed	. 35
1.7.28	crcPoly	. 35
1.7.29	crcXorOut	. 35
1.7.30	whitenEnable	. 35
1.7.31	whitenStart	. 35
1.7.32	whitenEnd	. 35
1.7.33	whitenB4Crc	. 35
1.7.34	whitenPolyType	. 36
1.7.35	whitenRefIn	. 36
1.7.36	whitenPayloadReinit	. 36
1.7.37	whitenSize	. 36
1.7.38	whitenInit	. 36
1.7.39	whitenPoly	. 36
1.7.40	whitenSizeThr	. 37
1.7.41	manchesterEn	
1.7.42	manchesterStart	. 37
1.7.43	manchesterInv	. 37
1.7.44	crcConfig	. 37
1.7.45	whitenerConfig	. 37
1.7.46	h0Field	
1.7.47	lengthField	
1.7.48	h1Field	
1.7.49	addr	
1.7.50	header	
1.7.51	payload	

## Chapter 1 Genfsk

#### 1.1 **Overview**

### **Files**

file genfsk\_interface.h

### **Data Structures**

- struct GENFSK\_nwk\_addr\_match\_t
- struct GENFSK\_TimeEvent\_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 gGenFskVerMajor\_c
- #define **gGenFskVerMinor\_c**
- #define **gGenFskVerPatch\_c**
- #define **gGenFskBuildNo\_c**
- #define **gGenFskModuleId\_c**
- #define gGenFskVerString\_c
- #define gGENFSK\_IrqPriority\_c
- #define gGENFSK\_TaskStackSize\_c
- #define gGENFSK\_TaskPriority\_c
- #define gGENFSK\_InstancesCnt\_c
- #define gGENFSK\_InvalidIdx\_c
  #define gGENFSK\_XcvrInitRetryCount\_c
  #define gGENFSK\_InvalidTimerId\_c
- #define gGENFSK\_NoMwsControl\_c
- #define gGENFSK\_LLMwsControl\_c
- #define gGENFSK\_AppMwsControl\_c
- #define gGENFSK MwsControl c

## **Typedefs**

- typedef uint32\_t GENFSK\_nwk\_addr\_ttypedef uint64\_t GENFSK\_timestamp\_t
- typedef uint8\_t genfskTimerId\_t
- typedef void(\* genfskTimeCallback\_t) (void)
- typedef void(\* genfskPacketReceivedCallBack\_t) (uint8\_t \*pBuffer, uint16\_t bufferLength, uint64\_t timestamp, int8\_t rssi, uint8\_t crcValid)

### Overview

### **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,
 gGenfskGapEvent,
 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 {
```

### **Overview**

```
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
   GENFSK\_timestamp\_t GENFSK\_TimeEvent\_t::timestamp
- genfskTimeCallback t GENFSK TimeEvent t::callback
- 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::packetTypeuint8\_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 tGENFSK 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 \*radio← Config, GENFSK\_packet\_config\_t \*packetConfig, GENFSK\_bitproc\_t \*bitProcConfig)
- genfskStatus\_t GENFSK\_RadioConfig (uint8\_t instanceId, GENFSK\_radio\_config\_t \*radio← Config)
- 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 maxBufLength← Bytes, GENFSK\_timestamp\_t rxStartTime, GENFSK\_timestamp\_t rxDuration)
- genfskStatus\_t GENFSK\_RestartRx (uint8\_t instanceId, uint8\_t \*pBuffer, uint16\_t maxBuf

  LengthBytes)
- genfskStatus\_t GENFSK\_CancelPendingRx (void)
- genfskStatus\_t GENFSK\_AbortAll (void)
- GENFSK\_timestamp\_t GENFSK\_GetTimestamp (void)
- genfskTimerId\_t GENFSK\_TimeScheduleEvent (GENFSK\_TimeEvent\_t \*pEvent)
- void GENFSK TimeCancelEvent (genfskTimerId t \*pTimerId)

NXP Semiconductors

### **Data Structure Documentation**

- genfskStatus\_t GENFSK\_PacketToByteArray (uint8\_t instanceId, GENFSK\_packet\_t \*pPacket, uint8 t \*pBuffer)
- genfskStatus\_t GENFSK\_RegisterCallbacks (uint8\_t instanceId, genfskPacketReceivedCallBack\_t packetReceivedCallback, genfskEventNotifyCallBack\_t eventCallback)
- genfskStatus\_t GENFSK\_RegisterRealTimeCallback (uint8\_t instanceId, genfskPacketReceived Callback\_t packetReceivedCallback)
- genfskStatus t GENFSK GetState (uint8 t instanceId)

### 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←	Sync (network) address length in bytes, allowed range is 03 rep-
	Bytes	resenting 1 to 4 bytes long sync addresses.
uint8_t	nwkAddrThr⊷	Sync (network) address matching threshold, number of bits that
	Bits	can mismatch and still be considered a match.
GENFSK_←	nwkAddr	The network address to be matched.
nwk_addr_t		

## 1.2.2 struct GENFSK\_TimeEvent\_t

GENFSK time structure.

Data Fields

GENFSK_←	timestamp	
timestamp_t		
genfskTime←	callback	
Callback_t		

## 1.2.3 struct GENFSK\_radio\_config\_t

GENFSK radio configure structure.

### Data Fields

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

## 1.2.4 struct GENFSK\_packet\_config\_t

GENFSK packet format configure structure.

Data Fields

uint8_t	preambleSize←	Preamble length in bytes, allowed range is 07 representing 1 to 8
	Bytes	bytes long preambles.
genfskPacket←	packetType	Packet type. See "genfskPacketType_t".
Type_t		
uint8_t	lengthSizeBits	Number of bits in the LENGTH field.
genfskPacket←	lengthBitOrder	Bit order for the LENGTH field of the header. See "genfskPacket←
<b>CfgLengthBit</b> ←		CfgLengthBitOrd_t".
Ord_t		
uint8_t	syncAddr←	Sync (network) address length in bytes, allowed range is 03 rep-
	SizeBytes	resenting 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.

## 1.2.5 struct GENFSK\_crc\_config\_t

GENFSK CRC module configure structure.

Data Fields

## **Data Structure Documentation**

genfskCrc←	crcEnable	Software override of the HW-computed CRC for TX. See "genfsk⊷
Compute←		CrcComputeMode_t".
Mode_t		
genfskCrc←	crcRecvInvalid	Receive packets with invalid CRC. See "genfskRecvInvalidCrc_t".
RecvInvalid_t		
uint8_t	crcSize	Number of CRC octets, allowed range is 04.
uint8_t	crcStartByte	Start CRC with this byte position. Byte #0 is the first byte of Sync
		Address.
genfskCrc←	crcRefIn	CRC reflect input. See "genfskCrcCfgCrcRefIn_t".
CfgCrcRefIn←		
_t		
genfskCrc←	crcRefOut	CRC reflect output. See "genfskCrcCfgCrcRefOut_t".
CfgCrcRef←		
Out_t		
genfskCrc←	crcByteOrder	CRC byte order. See "genfskCrcCfgCrcByteOrd_t".
CfgCrcByte ←		
Ord_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).

## 1.2.6 struct GENFSK\_whitener\_config\_t

GENFSK Whitener module configure structure.

Note

Whitening and Manchester encoding are mutually exclusive.

### Data Fields

genfsk⇔	whitenEnable	Enable/Disable HW (de)whitening on RX and TX packets. S	See
WhitenMode←		"genfskWhitenMode_t".	
_t			
genfsk↔	whitenStart	Configure Whitener start point. See "genfskWhitenStart_t".	
WhitenStart_t			
genfsk↔	whitenEnd	Configure end-of-whitening. See "genfskWhitenEnd_t".	
WhitenEnd_t			

## **Data Structure Documentation**

genfsk← WhitenB4Crc←	whitenB4Crc	Configure for whitening-before-CRC. See "genfskWhitenB4Crc  _t".
_t		
genfskض	whitenPoly⇔	Whiten polynomial type. See "genfskWhitenPolyType_t".
WhitenPoly⊷	Type	
Type_t		
genfskض	whitenRefIn	Whiten reflect input. See "genfskWhitenCfgRefIn_t".
WhitenCfg←		
RefIn_t		
genfskض	whiten←	Configure for whitener re-initialization. See "genfskWhitenCfg↔
WhitenCfg←	PayloadReinit	PayloadReinit_t.
Payload←		
Reinit_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 en-
		able whiten. Requires WHITEN_START 2 or 3.
genfsk←	manchesterEn	Configure for Manchester Encoding/Decoding. See "genfsk⊷
Manchester←		ManchesterEn_t".
En_t		
genfsk⊷	manchester←	Configure for inverted Manchester Encoding. See "genfsk⊷
Manchester←	Start	ManchesterStart_t".
Start_t		
genfsk⊷	manchesterInv	Configure Manchester Encoding start point. See "genfsk⊷
Manchester←		ManchesterInv_t".
Inv_t		

## 1.2.7 struct GENFSK\_bitproc\_t

GENFSK bitstream processing configuration.

Configuration of the bitstream processing to be done for GENFSK.

Data Fields

GENFSK_←	crcConfig	CRC module configuration structure.
crc_config_t		
*		

NXP Semiconductors 9

### **Macro Definition Documentation**

GENFSK_←	whitenerConfig	Whitener module configuration structure.
whitener_←		
config_t		
*		

## 1.2.8 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.

## 1.2.9 struct GENFSK\_packet\_t

Data structure for GENFSK packet.

Data Fields

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

## 1.3 Macro Definition Documentation

## 1.3.1 #define gGenFskVerMajor\_c

Generic FSK build version.

## 1.3.2 #define GENFSK\_DRIVER\_VERSION

GENFSK Link Layer driver version 0.0.1.

## 1.3.3 #define gGENFSK\_IrqPriority\_c

GENFSK Protocol Engine interrupt.

GENFSK Protocol Engine interrupt priority.

### 1.3.4 #define gGENFSK TaskStackSize c

GENFSK LL Task stack size.

### 1.3.5 #define gGENFSK TaskPriority c

GENFSK LL Task priority.

## 1.3.6 #define gGENFSK\_InstancesCnt\_c

GENFSK LL total number of available instances.

## 1.3.7 #define gGENFSK\_InvalidIdx\_c

GENFSK LL invalid instance ID.

## 1.3.8 #define gGENFSK\_XcvrInitRetryCount\_c

maximum GENFSK XCVR\_Init retry count.

## 1.3.9 #define gGENFSK\_InvalidTimerId\_c

GENFSK Timer invalid ID.

## 1.3.10 #define gGENFSK\_NoMwsControl\_c

GENFSK does not usw MWS.

## 1.3.11 #define gGENFSK\_LLMwsControl\_c

The GENFSK Link Layer controls the MWS.

## 1.3.12 #define gGENFSK\_AppMwsControl\_c

The Application Layer controls the MWS.

### **Typedef Documentation**

### 1.3.13 #define gGENFSK MwsControl c

GENFSK MWS control option.

## 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.

## $1.4.3 \quad typedef \ uint8\_t \ genfskTimerId\_t$

GENFSK timer ID type.

## 1.4.4 typedef void(\* genfskTimeCallback\_t) (void)

GENFSK timeout callback.

## 1.4.5 typedef void( \* genfskPacketReceivedCallBack\_t) (uint8\_t \*pBuffer, uint16\_t bufferLength, uint64\_t timestamp, int8\_t rssi, uint8\_t crcValid)

Packet Received callback function pointer type.

#### **Parameters**

<i>pBuffer</i> 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.

## 1.4.6 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".

#### **Enumeration Type Documentation** 1.5

#### 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.

```
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.
```

### **Enumeration Type Documentation**

### 1.5.3 enum genfskStatus\_t

Error codes for the GENESK driver.

#### Enumerator

## 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.
    gGenfskGapEvent Gap packet processing.
    gGenfskAllEvents All events.
```

## 1.5.5 enum genfskEventStatus\_t

GENFSK notification events status.

```
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.

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

### **Enumeration Type Documentation**

## 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.

```
gWhitenStartNoWhitening No whitening.
gWhitenStartWhiteningAtH0 Start whitening at start-of-H0.
gWhitenStartWhiteningAtH1 Start whitening at start-of-H1 but only if LENGTH > WHITEN_S← Z_THR.
gWhitenStartWhiteningAtPayload Start whitening at start-of-payload but only if LENGTH > W← HITEN_SZ_THR.
```

17

## 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.

## **Enumeration Type Documentation**

## 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 re	ason.
---	-------

## 1.6.2 genfskStatus\_t GENFSK AllocInstance ( uint8 t \* plnstanceld, **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();

### **Function Documentation**

## 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 instanceld, 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.
Ī	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 instanceld, 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 instanceld, 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 instanceld, 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_o	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.

### **Function Documentation**

### **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 fail	ure reason
g Genjskbuccess_c   II success of the fair	uic icason.
8 - 1 3 - 1 - 1 - 1 - 1 - 1 - 1 - 1 - 1 -	

## 1.6.9 genfskStatus\_t GENFSK\_GetWhitenerConfig ( uint8\_t instanceld, 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

0 010	1 0 11
al'antal Juggara	if success or the failure reason.
9 GENISK SUCCESS C	H SUCCESS OF THE TAILING TEASON.
goenjananeeess_e	ii baccess of the fallare reason.
0 9 =	

## 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_o	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 instanceld, 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 03. This location will be enabled if there
	are no errors during the setting process.
nwkAddress⇔	the settings to be applied.
Settings	

### 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.

NXP Semiconductors 23

### **Function Documentation**

### Parameters

instanceId	The ID of the instance. Re	urns the current enabled events for genfskEventNotify-
	CallBack_t.	

### Return values

genfskEvent_t.	

## 1.6.15 genfskStatus\_t GENFSK\_GetNetworkAddress ( uint8\_t instanceld, 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 03. This location will be enabled if there
	are no errors during the setting process.
nwkAddress⇔	the stored network address settings at the specified location.
Settings	

### 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

insta	ınceId	The ID of the instance.
loc	cation	the location number to disable, valid range is 03. This location will be enabled if
		there are no errors during the setting process.

### Return values

## 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 03. This location will be disabled if
	there are no errors during the setting process.

### Return values

gGenfskSuccess_o	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, $0 \le \text{channelNum} \le 127$ ;
	Formula: $F = (2360 + channelNum) [in MHz].$

### Return values

C (10	1.0
gGentskSuccess c	if success or the failure reason.
Z O CH J S N S H C C C S S _ C	ii success of 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.

### **Function Documentation**

### Parameters

instanceId	The ID of the instance.
------------	-------------------------

### Return values

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

## 1.6.20 genfskStatus\_t GENFSK\_SetTxPowerLevel ( uint8\_t instanceld, 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, $0 \le \text{txPowerLevel} \le 32$ .

### Return values

1		
	gGenfskSuccess c	if success or the failure reason.
	g Gerijski success_c	in success of the failure reason.

## 1.6.21 uint8\_t GENFSK\_GetTxPowerLevel ( uint8\_t instanceld )

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, $0 \le \text{txPowerLevel} \le 32$ .
-----	--

## 1.6.22 genfskStatus\_t GENFSK\_StartTx ( uint8\_t instanceld, 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 con-
	figured 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 ( $\sim$ 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

0 010	10 .1 0.11
gGentskSuccess c	it success or the failure reason.
g Geniskbaccess c	ii success of the faiture reason.
0 9 =	

# 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

instance	Id The ID of the instance.
pBufj	The pointer to a buffer used for reception.
тахВи	The allocated pBuffer size for the maximum packet length that can be received.
LengthBy	es

NXP Semiconductors 27

### **Function Documentation**

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.
0 7 -	

### Warning

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

## 1.6.25 genfskStatus\_t GENFSK\_RestartRx ( uint8\_t instanceId, uint8\_t \* pBuffer, uint16\_t maxBufLengthBytes )

Restarts receive operation.

This function restarts receive operation.

### **Parameters**

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

### Return values

$gGenfskSuccess\_c$ if success or the failure reason.
---

## 1.6.26 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

0 010	1. 0.11
gGentskSuccess c	if success or the failure reason.
g Genjskouccess c	ii success of the faiture reason.
8 9	

## 1.6.27 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.

## 1.6.28 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.

## 1.6.29 genfskTimerId\_t GENFSK\_TimeScheduleEvent ( GENFSK\_TimeEvent\_t \* pEvent )

Schedules an event.

**Parameters** 

*pEvent* | event to be scheduled.

Return values

genfskTimerId t the ID of the allocated timer.

## 1.6.30 void GENFSK\_TimeCancelEvent ( genfskTimerId\_t \* pTimerId )

Cancels an event.

**Parameters** 

*pTimerId* pointer to the ID of the timer. Will be reset to gGENFSK\_InvalidTimerId\_c.

## 1.6.31 genfskStatus\_t GENFSK\_PacketToByteArray ( uint8\_t instanceld, GENFSK\_packet\_t \* pPacket, uint8\_t \* pBuffer )

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

NXP Semiconductors 29

### **Function Documentation**

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 This API operates in two modes: "copy" mode and "no copy" mode.

- "copy" mode is assumed when pPacket->payload is specified: packet header and payload are copied into pBuffer
- "no copy" is assumed when pPacket->payload is NULL. Only packet header is formed into pBuffer, payload has to be copied by the caller.

### 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.
------------------	-----------------------------------

## 1.6.32 genfskStatus\_t GENFSK\_ByteArrayToPacket ( uint8\_t instanceld, 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.
	pВuffer	Pointer to the byte array formatted buffer.
pPacket   Pointer to the packet structure to store the formatted packet		Pointer to the packet structure to store the formatted packet.

### Return values

gGenfskSuccess_o	if success or the failure reason.
------------------	-----------------------------------

# 1.6.33 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.
packet←	Packet received callback.
$Received \leftarrow$	
Callback	
eventCallback	Event callback.

### Return values

gGenfskSuccess_c	if success or the failure reason.

## 1.6.34 genfskStatus\_t GENFSK\_RegisterRealTimeCallback ( uint8\_t instanceId, genfskPacketReceivedCallBack\_t packetReceivedCallback )

Registers the real time callback functions for packet received.

This function register the real time callback functions for packet received

### **Parameters**

instanceId	The ID of the instance.
packet←	Packet received callback.
$Received \leftarrow$	
Callback	

### Return values

$C = C \cdot 1 \cdot C$	:C
gGentskSuccess c	if success or the failure reason.
8 9 c. 19 s. 18 te c c s s _c	II SWOODS OF WIN TWING TOWN OFF

Warning: packetReceivedCallback will be called in ISR context (when RX\_IRQ fires)

## 1.6.35 genfskStatus\_t GENFSK\_GetState ( uint8\_t instanceId )

Retreives the GENFSK LL state receive operation.

### **Parameters**

instanceId	The ID of the instance.

### Return values

genfskStatus_t	The status of the GENFSK instance
----------------	-----------------------------------

NXP Semiconductors 31

### Variable Documentation

### 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 GENFSK\_timestamp\_t GENFSK TimeEvent t::timestamp

## 1.7.5 genfskTimeCallback\_t GENFSK\_TimeEvent\_t::callback

## 1.7.6 genfskRadioMode\_t GENFSK radio config t::radioMode

Radio mode for GENFSK radio.

See "genfskRadioMode\_t".

## 1.7.7 genfskDataRate\_t GENFSK\_radio\_config\_t::dataRate

Data rate for GENFSK radio.

See "genfskDataRate\_t".

## 1.7.8 uint8 t GENFSK packet config t::preambleSizeBytes

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

## 1.7.9 genfskPacketType\_t GENFSK\_packet\_config\_t::packetType

Packet type.

See "genfskPacketType\_t".

### 1.7.10 uint8 t GENFSK packet config t::lengthSizeBits

Number of bits in the LENGTH field.

### 1.7.11 genfskPacketCfgLengthBitOrd\_t GENFSK packet config t::lengthBitOrder

Bit order for the LENGTH field of the header.

See "genfskPacketCfgLengthBitOrd\_t".

## 1.7.12 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.

## 1.7.13 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.

## 1.7.14 uint8\_t GENFSK\_packet\_config\_t::h0SizeBits

Number of bits in the H0 field.

## 1.7.15 uint8\_t GENFSK\_packet\_config\_t::h1SizeBits

Number of bits in the H1 field.

## 1.7.16 uint16\_t GENFSK\_packet\_config\_t::h0Match

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

## 1.7.17 uint16\_t GENFSK\_packet\_config\_t::h0Mask

Mask to select which bits of H0 must match the h0\_match field.

### Variable Documentation

## 1.7.18 uint16\_t GENFSK\_packet\_config\_t::h1Match

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

### 1.7.19 uint16\_t GENFSK\_packet\_config\_t::h1Mask

Mask to select which bits of H1 must match the h1\_match field.

### 1.7.20 genfskCrcComputeMode\_t GENFSK crc config t::crcEnable

Software override of the HW-computed CRC for TX.

See "genfskCrcComputeMode\_t".

### 1.7.21 genfskCrcRecvInvalid\_t GENFSK crc config t::crcRecvInvalid

Receive packets with invalid CRC.

See "genfskRecvInvalidCrc\_t".

### 1.7.22 uint8\_t GENFSK\_crc\_config\_t::crcSize

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

## 1.7.23 uint8\_t GENFSK\_crc\_config\_t::crcStartByte

Start CRC with this byte position.

Byte #0 is the first byte of Sync Address.

## 1.7.24 genfskCrcCfgCrcRefIn\_t GENFSK\_crc\_config\_t::crcRefIn

CRC reflect input.

See "genfskCrcCfgCrcRefIn\_t".

## 1.7.25 genfskCrcCfgCrcRefOut\_t GENFSK\_crc\_config\_t::crcRefOut

CRC reflect output.

See "genfskCrcCfgCrcRefOut\_t".

## 1.7.26 genfskCrcCfgCrcByteOrd\_t GENFSK\_crc\_config\_t::crcByteOrder

CRC byte order.

See "genfskCrcCfgCrcByteOrd\_t".

### 1.7.27 uint32 t GENFSK crc config t::crcSeed

CRC Seed value.

Initial value for CRC LFSR.

### 1.7.28 uint32 t GENFSK crc config t::crcPoly

CRC Polynomial value.

## 1.7.29 uint32 t GENFSK crc config t::crcXorOut

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

## 1.7.30 genfskWhitenMode\_t GENFSK\_whitener\_config\_t::whitenEnable

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

See "genfskWhitenMode\_t".

## 1.7.31 genfskWhitenStart\_t GENFSK\_whitener\_config\_t::whitenStart

Configure Whitener start point.

See "genfskWhitenStart\_t".

## 1.7.32 genfskWhitenEnd\_t GENFSK\_whitener\_config\_t::whitenEnd

Configure end-of-whitening.

See "genfskWhitenEnd\_t".

## 1.7.33 genfskWhitenB4Crc\_t GENFSK\_whitener\_config\_t::whitenB4Crc

Configure for whitening-before-CRC.

### Variable Documentation

See "genfskWhitenB4Crc\_t".

### 1.7.34 genfskWhitenPolyType\_t GENFSK\_whitener\_config\_t::whitenPolyType

Whiten polynomial type.

See "genfskWhitenPolyType\_t".

## 1.7.35 genfskWhitenCfgRefIn\_t GENFSK\_whitener\_config\_t::whitenRefIn

Whiten reflect input.

See "genfskWhitenCfgRefIn\_t".

## 1.7.36 genfskWhitenCfgPayloadReinit\_t GENFSK\_whitener\_config\_t::whiten ← PayloadReinit

Configure for whitener re-initialization.

See "genfskWhitenCfgPayloadReinit\_t.

## 1.7.37 uint8\_t GENFSK\_whitener\_config\_t::whitenSize

Length of whitener LFSR.

Maximum value 9.

## 1.7.38 uint16 t GENFSK whitener config t::whitenInit

Initialization value for Whitening/De-whitening.

Maximum 9 bits.

## 1.7.39 uint16\_t GENFSK\_whitener\_config\_t::whitenPoly

Whitener polynomial.

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

### 1.7.40 uint16 t GENFSK whitener config t::whitenSizeThr

Whitener size threshold.

Maximum packet length required to enable whiten. Requires WHITEN\_START 2 or 3.

## 1.7.41 genfskManchesterEn\_t GENFSK\_whitener\_config\_t::manchesterEn

Configure for Manchester Encoding/Decoding.

See "genfskManchesterEn\_t".

## 1.7.42 genfskManchesterStart\_t GENFSK\_whitener\_config\_t::manchesterStart

Configure for inverted Manchester Encoding.

See "genfskManchesterStart\_t".

### 1.7.43 genfskManchesterInv t GENFSK whitener config t::manchesterInv

Configure Manchester Encoding start point.

See "genfskManchesterInv\_t".

## 1.7.44 GENFSK\_crc\_config\_t\* GENFSK\_bitproc\_t::crcConfig

CRC module configuration structure.

## 1.7.45 GENFSK\_whitener\_config\_t\* GENFSK\_bitproc\_t::whitenerConfig

Whitener module configuration structure.

### 1.7.46 uint16 t GENFSK packet header t::h0Field

H0 field value.

### 1.7.47 uint16 t GENFSK packet header t::lengthField

LENGTH field value.

### **Variable Documentation**

## 1.7.48 uint16\_t GENFSK\_packet\_header\_t::h1Field

H1 field value.

## 1.7.49 GENFSK\_nwk\_addr\_t GENFSK\_packet\_t::addr

Network address.

## 1.7.50 GENFSK\_packet\_header\_t GENFSK\_packet\_t::header

Packet header data structure.

## 1.7.51 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.

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

© 2019 NXP B.V.