

Generic FSK Link Layer

API Reference Manual

Rev. 1
Apr 2019



Contents

Chapter 1 Genfsk

1.1	Overview	1
1.2	Data Structure Documentation	6
1.2.1	struct GENFSK_nwk_addr_match_t	6
1.2.2	struct GENFSK_TimeEvent_t	6
1.2.3	struct GENFSK_radio_config_t	7
1.2.4	struct GENFSK_mode_config_t	7
1.2.5	struct GENFSK_packet_config_t	7
1.2.6	struct GENFSK_crc_config_t	8
1.2.7	struct GENFSK_whitener_config_t	9
1.2.8	struct GENFSK_bitproc_t	10
1.2.9	struct GENFSK_packet_header_t	10
1.2.10	struct GENFSK_packet_t	11
1.3	Macro Definition Documentation	11
1.3.1	GENFSK_DRIVER_VERSION	11
1.3.2	gGENFSK_IrqPriority_c	11
1.3.3	gGENFSK_TaskStackSize_c	11
1.3.4	gGENFSK_TaskPriority_c	11
1.3.5	gGENFSK_InstancesCnt_c	11
1.3.6	gGENFSK_InvalidIdx_c	12
1.3.7	gGENFSK_InvalidTimerId_c	12
1.4	Typedef Documentation	12
1.4.1	GENFSK_nwk_addr_t	12
1.4.2	GENFSK_timestamp_t	12
1.4.3	genfskTimerId_t	12
1.4.4	genfskTimeCallback_t	12
1.4.5	genfskPacketReceivedCallBack_t	12
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

Section number	Title	Page
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	16
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	18
1.5.20	genfskManchesterInv_t	18
1.5.21	genfskManchesterStart_t	18
1.5.22	genfskPacketType_t	19
1.5.23	genfskMode_t	19
1.6	Function Documentation	19
1.6.1	GENFSK_Init(void)	19
1.6.2	GENFSK_AllocInstance(uint8_t *pInstanceId, const GENFSK_radio_config_t *radioConfig, const GENFSK_packet_config_t *packetConfig, const GENFSK_bitproc_t *bitProcConfig, const GENFSK_mode_config_t *modeConfig)	20
1.6.3	GENFSK_RadioConfig(uint8_t instanceId, const GENFSK_radio_config_t *radioConfig)	20
1.6.4	GENFSK_SetPacketConfig(uint8_t instanceId, const GENFSK_packet_config_t *packetConfig)	21
1.6.5	GENFSK_GetPacketConfig(uint8_t instanceId, GENFSK_packet_config_t *packetConfig)	21
1.6.6	GENFSK_SetModeConfig(uint8_t instanceId, const GENFSK_mode_config_t *modeConfig)	21
1.6.7	GENFSK_GetModeConfig(uint8_t instanceId, GENFSK_mode_config_t *modeConfig)	22
1.6.8	GENFSK_SetCrcConfig(uint8_t instanceId, const GENFSK_crc_config_t *crcConfig)	22
1.6.9	GENFSK_GetCrcConfig(uint8_t instanceId, GENFSK_crc_config_t *crcConfig)	22
1.6.10	GENFSK_SetWhitenerConfig(uint8_t instanceId, const GENFSK_whitener_config_t *whitenerConfig)	23
1.6.11	GENFSK_GetWhitenerConfig(uint8_t instanceId, GENFSK_whitener_config_t *whitenerConfig)	23
1.6.12	GENFSK_FreeInstance(uint8_t instanceId)	23
1.6.13	GENFSK_ResetToDefaults(uint8_t instanceId)	24

Section number	Title	Page
1.6.14	GENFSK_SetNetworkAddress(uint8_t instanceId, uint8_t location, const GENFSK_nwk_addr_match_t *nwkAddressSettings)	24
1.6.15	GENFSK_SetEventMask(uint8_t instanceId, uint32_t mask)	24
1.6.16	GENFSK_GetEventMask(uint8_t instanceId)	25
1.6.17	GENFSK_GetNetworkAddress(uint8_t instanceId, uint8_t location, GENFSK_nwk_addr_match_t *nwkAddressSettings)	25
1.6.18	GENFSK_EnableNetworkAddress(uint8_t instanceId, uint8_t location)	25
1.6.19	GENFSK_DisableNetworkAddress(uint8_t instanceId, uint8_t location)	27
1.6.20	GENFSK_SetChannelNumber(uint8_t instanceId, uint8_t channelNum)	27
1.6.21	GENFSK_GetChannelNumber(uint8_t instanceId)	27
1.6.22	GENFSK_SetTxPowerLevel(uint8_t instanceId, uint8_t txPowerLevel)	29
1.6.23	GENFSK_GetTxPowerLevel(uint8_t instanceId)	29
1.6.24	GENFSK_StartTx(uint8_t instanceId, const uint8_t *pBuffer, uint16_t bufLengthBytes, GENFSK_timestamp_t txStartTime, bool followedByRx, GENFSK_timestamp_t rxDuration)	29
1.6.25	GENFSK_CancelPendingTx(void)	30
1.6.26	GENFSK_StartRx(uint8_t instanceId, uint8_t *pRxBuffer, uint16_t maxBufLengthBytes, GENFSK_timestamp_t rxStartTime, GENFSK_timestamp_t rxDuration, const uint8_t *pTxBuffer, uint16_t txBufLengthBytes)	30
1.6.27	GENFSK_CancelPendingRx(void)	31
1.6.28	GENFSK_AbortAll(void)	31
1.6.29	GENFSK_GetTimestamp(void)	31
1.6.30	GENFSK_TimeScheduleEvent(GENFSK_TimeEvent_t *pEvent)	32
1.6.31	GENFSK_TimeCancelEvent(genfskTimerId_t *pTimerId)	32
1.6.32	GENFSK_PacketToByteArray(uint8_t instanceId, const GENFSK_packet_t *pPacket, uint8_t *pBuffer)	32
1.6.33	GENFSK_ByteArrayToPacket(uint8_t instanceId, const uint8_t *pBuffer, GENFSK_packet_t *pPacket)	33
1.6.34	GENFSK_RegisterCallbacks(uint8_t instanceId, genfskPacketReceivedCallback_t packetReceivedCallback, genfskEventNotifyCallback_t eventCallback)	33
1.7	Variable Documentation	34
1.7.1	nwkAddrSizeBytes	34
1.7.2	nwkAddrThrBits	34
1.7.3	nwkAddr	34
1.7.4	timestamp	34
1.7.5	callback	34
1.7.6	radioMode	34
1.7.7	dataRate	34
1.7.8	mode	34
1.7.9	stayInRx	35
1.7.10	turnaroundTime	35
1.7.11	preambleSizeBytes	35
1.7.12	packetType	35
1.7.13	lengthSizeBits	35

Section number	Title	Page
1.7.14	lengthBitOrder	35
1.7.15	syncAddrSizeBytes	35
1.7.16	lengthAdjBytes	35
1.7.17	h0SizeBits	36
1.7.18	h1SizeBits	36
1.7.19	h0Match	36
1.7.20	h0Mask	36
1.7.21	h1Match	36
1.7.22	h1Mask	36
1.7.23	crcEnable	36
1.7.24	crcRecvInvalid	36
1.7.25	crcSize	37
1.7.26	crcStartByte	37
1.7.27	crcRefIn	37
1.7.28	crcRefOut	37
1.7.29	crcByteOrder	37
1.7.30	crcSeed	37
1.7.31	crcPoly	37
1.7.32	crcXorOut	37
1.7.33	whitenEnable	38
1.7.34	whitenStart	38
1.7.35	whitenEnd	38
1.7.36	whitenB4Crc	38
1.7.37	whitenPolyType	38
1.7.38	whitenRefIn	38
1.7.39	whitenPayloadReinit	38
1.7.40	whitenSize	39
1.7.41	whitenInit	39
1.7.42	whitenPoly	39
1.7.43	whitenSizeThr	39
1.7.44	manchesterEn	39
1.7.45	manchesterStart	39
1.7.46	manchesterInv	39
1.7.47	crcConfig	40
1.7.48	whitenerConfig	40
1.7.49	h0Field	40
1.7.50	lengthField	40
1.7.51	h1Field	40
1.7.52	addr	40
1.7.53	header	40
1.7.54	payload	40

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_mode_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](#)
- #define [gGENFSK_InvalidTimerId_c](#)

Typedefs

- typedef uint32_t [GENFSK_nwk_addr_t](#)
- typedef 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)
- typedef void(* [genfskEventNotifyCallBack_t](#)) ([genfskEvent_t](#) event, [genfskEventStatus_t](#) eventStatus)

Enumerations

- enum [genfskDataRate_t](#) {
 [gGenfskDR1Mbps](#),
 [gGenfskDR500Kbps](#),
 [gGenfskDR250Kbps](#) }

Overview

- 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,
gGenfskDefaultEnabledEvents,
gGenfskAllEvents,
gGenfskPromiscuousEvent }
- 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,
 gFibonacciPolyType }
- 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 }
- enum genfskMode_t {

Overview

```
gGenfskGllMode,  
gGenfskPanMode,  
gGenfskFanMode,  
gGenfskHybridDualPanMode,  
gGenfskWmbusAMode,  
gGenfskWmbusBMode,  
gGenfskAntMode,  
gGenfskLeg0LiMode,  
gGenfskLeg0noLiMode,  
gGenfskBleUncodedMode,  
gGenfskBleLrMode,  
gGenfskBleConcurentMode,  
gGenfskGtmMode }
```

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
- genfskMode_t GENFSK_mode_config_t::mode
- uint8_t GENFSK_mode_config_t::stayInRx
- uint32_t GENFSK_mode_config_t::turnaroundTime
- uint16_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
- int16_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, const `GENFSK_radio_config_t` *radioConfig, const `GENFSK_packet_config_t` *packetConfig, const `GENFSK_bitproc_t` *bitProcConfig, const `GENFSK_mode_config_t` *modeConfig)
- `genfskStatus_t` `GENFSK_RadioConfig` (uint8_t instanceId, const `GENFSK_radio_config_t` *radioConfig)
- `genfskStatus_t` `GENFSK_SetPacketConfig` (uint8_t instanceId, const `GENFSK_packet_config_t` *packetConfig)
- `genfskStatus_t` `GENFSK_GetPacketConfig` (uint8_t instanceId, `GENFSK_packet_config_t` *packetConfig)
- `genfskStatus_t` `GENFSK_SetModeConfig` (uint8_t instanceId, const `GENFSK_mode_config_t` *modeConfig)
- `genfskStatus_t` `GENFSK_GetModeConfig` (uint8_t instanceId, `GENFSK_mode_config_t` *modeConfig)
- `genfskStatus_t` `GENFSK_SetCrcConfig` (uint8_t instanceId, const `GENFSK_crc_config_t` *crcConfig)
- `genfskStatus_t` `GENFSK_GetCrcConfig` (uint8_t instanceId, `GENFSK_crc_config_t` *crcConfig)
- `genfskStatus_t` `GENFSK_SetWhitenerConfig` (uint8_t instanceId, const `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, const `GENFSK_nwk_addr_match_t` *nwkAddressSettings)
- `genfskStatus_t` `GENFSK_SetEventMask` (uint8_t instanceId, uint32_t mask)
- `uint32_t` `GENFSK_GetEventMask` (uint8_t instanceId)

Data Structure Documentation

- [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, const uint8_t *pBuffer, uint16_t bufLengthBytes, [GENFSK_timestamp_t](#) txStartTime, bool followedByRx, [GENFSK_timestamp_t](#) rxDuration)
- [genfskStatus_t GENFSK_CancelPendingTx](#) (void)
- [genfskStatus_t GENFSK_StartRx](#) (uint8_t instanceId, uint8_t *pRxBuffer, uint16_t maxBufLengthBytes, [GENFSK_timestamp_t](#) rxStartTime, [GENFSK_timestamp_t](#) rxDuration, const uint8_t *pTxBuffer, uint16_t txBufLengthBytes)
- [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)
- [genfskStatus_t GENFSK_PacketToByteArray](#) (uint8_t instanceId, const [GENFSK_packet_t](#) *pPacket, uint8_t *pBuffer)
- [genfskStatus_t GENFSK_ByteArrayToPacket](#) (uint8_t instanceId, const 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	nwkAddrSizeBytes	Sync (network) address length in bytes, allowed range is 0..3 representing 1 to 4 bytes long sync addresses.
uint8_t	nwkAddrThresholdBits	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_TimeEvent_t

GENFSK time structure.

Data Fields

GENFSK_timestamp_t	timestamp	
genfskTimeCallback_t	callback	

1.2.3 struct GENFSK_radio_config_t

GENFSK radio configure structure.

Data Fields

genfskRadioMode_t	radioMode	Radio mode for GENFSK radio. See "genfskRadioMode_t".
genfskDataRate_t	dataRate	Data rate for GENFSK radio. See "genfskDataRate_t".

1.2.4 struct GENFSK_mode_config_t

GENFSK mode configure structure.

Data Fields

genfskMode_t	mode	Mode for GENFSK link layer. See "genfskMode_t".
uint8_t	stayInRx	Enable the ability to recycle and stay in receive even after an RX_IRQ , to avoid the full RX re-warmup.
uint32_t	turnaroundTime	Tx to Rx or Rx to Tx delay in us. Only applies to combined RX/TX sequences. Not used otherwise

1.2.5 struct GENFSK_packet_config_t

GENFSK packet format configure structure.

Data Fields

uint16_t	preambleSizeBytes	Preamble length in bytes, allowed range is 0..511 representing 1 to 512 bytes long preambles.
genfskPacketType_t	packetType	Packet type. See "genfskPacketType_t".

Data Structure Documentation

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.
int16_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.

1.2.6 struct GENFSK_crc_config_t

GENFSK CRC module configure structure.

Data Fields

genfskCrc← Compute← Mode_t	crcEnable	Software override of the HW-computed CRC for TX. See "genfsk← CrcComputeMode_t".
genfskCrc← RecvInvalid_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.
genfskCrc← CfgCrcRefIn← _t	crcRefIn	CRC reflect input. See "genfskCrcCfgCrcRefIn_t".
genfskCrc← CfgCrcRef← Out_t	crcRefOut	CRC reflect output. See "genfskCrcCfgCrcRefOut_t".

genfskCrcCfcrcByteOrd_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).

1.2.7 struct GENFSK_whitener_config_t

GENFSK Whitener module configure structure.

Note

Whitening and Manchester encoding are mutually exclusive.

Data Fields

genfskWhitenMode_t	whitenEnable	Enable/Disable HW (de)whitening on RX and TX packets. See "genfskWhitenMode_t".
genfskWhitenStart_t	whitenStart	Configure Whitener start point. See "genfskWhitenStart_t".
genfskWhitenEnd_t	whitenEnd	Configure end-of-whitening. See "genfskWhitenEnd_t".
genfskWhitenB4Crc_t	whitenB4Crc	Configure for whitening-before-CRC. See "genfskWhitenB4Crc_t".
genfskWhitenPolyType_t	whitenPolyType	Whiten polynomial type. See "genfskWhitenPolyType_t".
genfskWhitenCfgRefIn_t	whitenRefIn	Whiten reflect input. See "genfskWhitenCfgRefIn_t".
genfskWhitenCfgPayloadReinit_t	whitenPayloadReinit	Configure for whitener re-initialization. See "genfskWhitenCfgPayloadReinit_t".

Data Structure Documentation

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 ".

1.2.8 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.

1.2.9 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.
----------	---------	-----------------

1.2.10 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.

Typedef Documentation

1.3.6 **#define gGENFSK_InvalidIdx_c**

GENFSK LL invalid instance ID.

1.3.7 **#define gGENFSK_InvalidTimerId_c**

GENFSK Timer invalid 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 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.
<i>packet_length</i>	The allocated pkt_buffer size for the maximum packet length that can be received.
<i>timestamp</i>	The timestamp for the received packet in microseconds.
<i>rssi</i>	The RSSI for the received packet.
<i>crcValid</i>	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

<i>event</i>	Reason the callback is being invoked. See "genfskEvent_t".
<i>eventStatus</i>	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.

Enumeration Type Documentation

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.
- gGenfskDefaultEnabledEvents*** default enabled events.
- gGenfskAllEvents*** All events.
- gGenfskPromiscuousEvent*** Promiscuous event .

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.

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.

Enumerator

- 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.

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).

1.5.23 enum genfskMode_t

Link layer Mode definition.

Enumerator

gGenfskGllMode 0000b - GLL Mode

gGenfskPanMode 0001b - PAN Mode

gGenfskFanMode 0010b - FAN Mode

gGenfskHybridDualPanMode 0011b - Hybrid Dual PAN Mode

gGenfskWmbusAMode 0100b - WMBUSA Mode

gGenfskWmbusBMode 0101b - WMBUSB Mode

gGenfskAntMode 0110b - ANT Mode

gGenfskLeg0LiMode 0111b - LEG0 Mode, packet has Length-field

gGenfskLeg0noLiMode 1000b - LEG0 Mode, packet has no Length-field

gGenfskBleUncodedMode 1001b - BLE Uncoded Mode

gGenfskBleLrMode 1010b - BLE LR Mode

gGenfskBleConcurrentMode 1011b - BLE Concurrent Mode (RX configuration only; TX uses either BLE_UNCODED or BLE_LR configuration)

gGenfskGtmMode 1111b - GTM Mode

1.6 Function Documentation

1.6.1 genfskStatus_t GENFSK_Init (void)

Initializes the GENFSK LL.

This function initializes the GENFSK LL.

Function Documentation

Return values

<i>gGenfskSuccess_c</i>	if success or the failure reason.
-------------------------	-----------------------------------

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

Allocates a GENFSK LL instance.

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

Parameters

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

Return values

<i>gGenfskSuccess_c</i>	if success or the failure reason.
-------------------------	-----------------------------------

Warning

Should be called after [GENFSK_Init\(\)](#);

1.6.3 genfskStatus_t GENFSK_RadioConfig (uint8_t *instanceId*, const 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

<i>instanceId</i>	The ID of the instance.
-------------------	-------------------------

<i>radioConfig</i>	The radio configuration to be set for GENFSK LL.
--------------------	--

Return values

<i>gGenfskSuccess_c</i>	if success or the failure reason.
-------------------------	-----------------------------------

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

Sets the packet configuration for the current GENFSK LL instance.

This function sets the packet configuration.

Parameters

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

Return values

<i>gGenfskSuccess_c</i>	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

<i>instanceId</i>	The ID of the instance.
<i>packetConfig</i>	The stored packet configuration.

Return values

<i>gGenfskSuccess_c</i>	if success or the failure reason.
-------------------------	-----------------------------------

1.6.6 **genfskStatus_t GENFSK_SetModeConfig (uint8_t *instanceId*, const GENFSK_mode_config_t * *modeConfig*)**

Sets the mode configuration for the current GENFSK LL instance.

This function sets the mode configuration.

Function Documentation

Parameters

<i>instanceId</i>	The ID of the instance.
<i>modeConfig</i>	The mode configuration to be set in GENFSK LL.

Return values

<i>gGenfskSuccess_c</i>	if success or the failure reason.
-------------------------	-----------------------------------

1.6.7 **genfskStatus_t GENFSK_GetModeConfig (uint8_t *instanceId*, GENFSK_mode_config_t * *modeConfig*)**

Returns the mode configuration currently set in GENFSK LL.

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

Parameters

<i>instanceId</i>	The ID of the instance.
<i>modeConfig</i>	The stored mode configuration.

Return values

<i>gGenfskSuccess_c</i>	if success or the failure reason.
-------------------------	-----------------------------------

1.6.8 **genfskStatus_t GENFSK_SetCrcConfig (uint8_t *instanceId*, const GENFSK_crc_config_t * *crcConfig*)**

Sets the CRC configuration for the current GENFSK LL instance.

This function sets the CRC configuration.

Parameters

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

Return values

<i>gGenfskSuccess_c</i>	if success or the failure reason.
-------------------------	-----------------------------------

1.6.9 **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

<i>instanceId</i>	The ID of the instance.
<i>crcConfig</i>	The stored CRC configuration.

Return values

<i>gGenfskSuccess_c</i>	if success or the failure reason.
-------------------------	-----------------------------------

1.6.10 **genfskStatus_t** GENFSK_SetWhitenerConfig (uint8_t *instanceId*, const GENFSK_whitener_config_t * *whitenerConfig*)

Sets whitening configuration for the current GENFSK LL instance.

This function sets the whitening configuration.

Parameters

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

Return values

<i>gGenfskSuccess_c</i>	if success or the failure reason.
-------------------------	-----------------------------------

1.6.11 **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

<i>instanceId</i>	The ID of the instance.
<i>whitenerConfig</i>	The stored whitening configuration.

Return values

<i>gGenfskSuccess_c</i>	if success or the failure reason.
-------------------------	-----------------------------------

1.6.12 **genfskStatus_t** GENFSK_FreelInstance (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.

Function Documentation

Parameters

<i>instanceId</i>	The ID of the instance.
-------------------	-------------------------

Return values

<i>gGenfskSuccess_c</i>	if success or the failure reason.
-------------------------	-----------------------------------

1.6.13 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

<i>instanceId</i>	The ID of the instance.
-------------------	-------------------------

1.6.14 genfskStatus_t GENFSK_SetNetworkAddress (uint8_t *instanceId*, uint8_t *location*, const GENFSK_nwk_addr_match_t * *nwkAddressSettings*)

Controls setting one of the network address match locations.

This function set the network address matching.

Parameters

<i>instanceId</i>	The ID of the instance.
<i>location</i>	the location number to set, valid range is 0..3. This location will be enabled if there are no errors during the setting process.
<i>nwkAddressSettings</i>	the settings to be applied.

Return values

<i>gGenfskSuccess_c</i>	if success or the failure reason.
-------------------------	-----------------------------------

1.6.15 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

<i>instanceId</i>	The ID of the instance.
<i>mask</i>	The event mask specifies which notification events are sent by genfskEventNotify↔ CallBack_t. See "genfskEvent_t".

Return values

<i>gGenfskSuccess_c</i>	if success or the failure reason.
-------------------------	-----------------------------------

1.6.16 uint32_t GENFSK_GetEventMask (uint8_t *instanceId*)

Returns the event mask for genfskEventNotifyCallBack_t callback.

Parameters

<i>instanceId</i>	The ID of the instance. Returns the current enabled events for genfskEventNotify↔ CallBack_t.
-------------------	--

Return values

<i>genfskEvent_t.</i>	
-----------------------	--

1.6.17 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

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

Return values

<i>gGenfskSuccess_c</i>	if success or the failure reason.
-------------------------	-----------------------------------

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

Controls enabling one of the network address match locations.



Function Documentation

This function enables one of the network address matching.

Parameters

<i>instanceId</i>	The ID of the instance.
<i>location</i>	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

<i>gGenfskSuccess_c</i>	if success or the failure reason.
-------------------------	-----------------------------------

1.6.19 **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

<i>instanceId</i>	The ID of the instance.
<i>location</i>	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

<i>gGenfskSuccess_c</i>	if success or the failure reason.
-------------------------	-----------------------------------

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

Sets the channel number.

This function sets the channel number.

Parameters

<i>instanceId</i>	The ID of the instance.
<i>channelNum</i>	The channel number on which to transmit and receive, $0 \leq \text{channelNum} \leq 127$; Formula: $F = (2360 + \text{channelNum})$ [in MHz].

Return values

<i>gGenfskSuccess_c</i>	if success or the failure reason.
-------------------------	-----------------------------------

1.6.21 **uint8_t GENFSK_GetChannelNumber (uint8_t *instanceId*)**

Returns the channel number currently set in GENFSK LL.



Function Documentation

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

Parameters

<i>instanceId</i>	The ID of the instance.
-------------------	-------------------------

Return values

<i>The</i>	channel number currently set, $0 \leq \text{channelNum} \leq 127$; Formula: $F = (2360 + \text{channelNum})$ [in MHz].
------------	---

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

Sets the power level for transmission.

This function sets power level for transmission.

Parameters

<i>instanceId</i>	The ID of the instance.
<i>txPowerLevel</i>	The power level for transmission, $0 \leq \text{txPowerLevel} \leq 32$.

Return values

<i>gGenfskSuccess_c</i>	if success or the failure reason.
-------------------------	-----------------------------------

1.6.23 **uint8_t** GENFSK_GetTxPowerLevel (uint8_t *instanceId*)

Returns the power level currently set in GENFSK LL.

Parameters

<i>instanceId</i>	The ID of the instance. This function returns the power level currently set in GENFSK LL.
-------------------	---

Return values

<i>The</i>	power level for transmission, $0 \leq \text{txPowerLevel} \leq 32$.
------------	--

1.6.24 **genfskStatus_t** GENFSK_StartTx (uint8_t *instanceId*, const uint8_t * *pBuffer*, uint16_t *bufLengthBytes*, GENFSK_timestamp_t *txStartTime*, bool *followedByRx*, GENFSK_timestamp_t *rxDuration*)

Performs a transmission.

This function performs a transmission of GENFSK LL Packet.

Function Documentation

Parameters

<i>instanceId</i>	The ID of the instance.
<i>pBuffer</i>	The pointer to a buffer containing the packet body compliant to the previously configured settings.
<i>bufLengthBytes</i>	The buffer length in bytes.
<i>txStartTime</i>	The time at which to start transmission. Set 0 for immediate transmission.
<i>followedByRx</i>	If set to 1, RX sequence will automatically start after Tx completion.
<i>rxDuration</i>	The duration of the receive operation. Set 0 for continuous reception. Only valid if followedByRx is set to 1

Return values

<i>gGenfskSuccess_c</i>	if success or the failure reason.
-------------------------	-----------------------------------

Warning

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

1.6.25 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

<i>gGenfskSuccess_c</i>	if success or the failure reason.
-------------------------	-----------------------------------

1.6.26 genfskStatus_t GENFSK_StartRx (uint8_t instanceId, uint8_t * pRxBuffer, uint16_t maxBufLengthBytes, GENFSK_timestamp_t rxStartTime, GENFSK_timestamp_t rxDuration, const uint8_t * pTxBuffer, uint16_t txBufLengthBytes)

Performs a receive operation.

This function performs a receive operation.

Parameters

<i>instanceId</i>	The ID of the instance.
-------------------	-------------------------

<i>pRxBuffer</i>	The pointer to a buffer used for reception.
<i>maxBuf← LengthBytes</i>	The allocated pBuffer size for the maximum packet length that can be received.
<i>rxStartTime</i>	The time at which to start receive. Set 0 for immediate receive.
<i>rxDuration</i>	The duration of the receive operation. Set 0 for continuous reception.
<i>pTxBuffer</i>	The pointer to a buffer used for transmission in case of AutoAck. Set it to NULL to disable AutoAck.
<i>txBufLength← Bytes</i>	The buffer length in bytes. Valid only when AutoAck enabled.

Return values

<i>gGenfskSuccess_c</i>	if success or the failure reason.
-------------------------	-----------------------------------

Warning

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

1.6.27 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

<i>gGenfskSuccess_c</i>	if success or the failure reason.
-------------------------	-----------------------------------

1.6.28 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

<i>gGenfskSuccess_c</i>	if success or the failure reason.
-------------------------	-----------------------------------

1.6.29 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.

Function Documentation

Return values

<i>The</i>	value of the timebase, in microseconds.
------------	---

Warning

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

1.6.30 **genfskTimerId_t GENFSK_TimeScheduleEvent (GENFSK_TimeEvent_t * *pEvent*)**

Schedules an event.

Parameters

<i>pEvent</i>	event to be scheduled.
---------------	------------------------

Return values

<i>genfskTimerId_t</i>	the ID of the allocated timer.
------------------------	--------------------------------

1.6.31 **void GENFSK_TimeCancelEvent (genfskTimerId_t * *pTimerId*)**

Cancels an event.

Parameters

<i>pTimerId</i>	pointer to the ID of the timer. Will be reset to gGENFSK_InvalidTimerId_c.
-----------------	--

1.6.32 **genfskStatus_t GENFSK_PacketToByteArray (uint8_t *instanceId*, const 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 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

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

Return values

<i>gGenfskSuccess_c</i>	if success or the failure reason.
-------------------------	-----------------------------------

1.6.33 **genfskStatus_t GENFSK_ByteArrayToPacket (uint8_t *instanceId*, const 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

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

Return values

<i>gGenfskSuccess_c</i>	if success or the failure reason.
-------------------------	-----------------------------------

1.6.34 **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

<i>instanceId</i>	The ID of the instance.
<i>packetReceivedCallback</i>	Packet received callback.
<i>eventCallback</i>	

Variable Documentation

<i>eventCallback</i>	Event callback.
----------------------	-----------------

Return values

<i>gGenfskSuccess_c</i>	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 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 genfskMode_t GENFSK_mode_config_t::mode

Mode for GENFSK link layer.

See "genfskMode_t".

1.7.9 uint8_t GENFSK_mode_config_t::stayInRx

Enable the ability to recycle and stay in receive even after an RX_IRQ, to avoid the full RX re-warmup.

1.7.10 uint32_t GENFSK_mode_config_t::turnaroundTime

Tx to Rx or Rx to Tx delay in us.

Only applies to combined RXX/Tx sequences. Not used otherwise

1.7.11 uint16_t GENFSK_packet_config_t::preambleSizeBytes

Preamble length in bytes, allowed range is 0..511 representing 1 to 512 bytes long preambles.

1.7.12 genfskPacketType_t GENFSK_packet_config_t::packetType

Packet type.

See "genfskPacketType_t".

1.7.13 uint8_t GENFSK_packet_config_t::lengthSizeBits

Number of bits in the LENGTH field.

1.7.14 genfskPacketCfgLengthBitOrd_t GENFSK_packet_config_t::lengthBitOrder

Bit order for the LENGTH field of the header.

See "genfskPacketCfgLengthBitOrd_t".

1.7.15 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.

Variable Documentation

1.7.16 int16_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.

1.7.17 uint8_t GENFSK_packet_config_t::h0SizeBits

Number of bits in the H0 field.

1.7.18 uint8_t GENFSK_packet_config_t::h1SizeBits

Number of bits in the H1 field.

1.7.19 uint16_t GENFSK_packet_config_t::h0Match

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

1.7.20 uint16_t GENFSK_packet_config_t::h0Mask

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

1.7.21 uint16_t GENFSK_packet_config_t::h1Match

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

1.7.22 uint16_t GENFSK_packet_config_t::h1Mask

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

1.7.23 genfskCrcComputeMode_t GENFSK_crc_config_t::crcEnable

Software override of the HW-computed CRC for TX.

See "genfskCrcComputeMode_t".

1.7.24 genfskCrcRecvInvalid_t GENFSK_crc_config_t::crcRecvInvalid

Receive packets with invalid CRC.

See "genfskRecvInvalidCrc_t".

1.7.25 uint8_t GENFSK_crc_config_t::crcSize

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

1.7.26 uint8_t GENFSK_crc_config_t::crcStartByte

Start CRC with this byte position.

Byte #0 is the first byte of Sync Address.

1.7.27 genfskCrcCfgCrcRefIn_t GENFSK_crc_config_t::crcRefIn

CRC reflect input.

See "genfskCrcCfgCrcRefIn_t".

1.7.28 genfskCrcCfgCrcRefOut_t GENFSK_crc_config_t::crcRefOut

CRC reflect output.

See "genfskCrcCfgCrcRefOut_t".

1.7.29 genfskCrcCfgCrcByteOrd_t GENFSK_crc_config_t::crcByteOrder

CRC byte order.

See "genfskCrcCfgCrcByteOrd_t".

1.7.30 uint32_t GENFSK_crc_config_t::crcSeed

CRC Seed value.

Initial value for CRC LFSR.

Variable Documentation

1.7.31 uint32_t GENFSK_crc_config_t::crcPoly

CRC Polynomial value.

1.7.32 uint32_t GENFSK_crc_config_t::crcXorOut

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

1.7.33 genfskWhitenMode_t GENFSK_whitener_config_t::whitenEnable

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

See "genfskWhitenMode_t".

1.7.34 genfskWhitenStart_t GENFSK_whitener_config_t::whitenStart

Configure Whitener start point.

See "genfskWhitenStart_t".

1.7.35 genfskWhitenEnd_t GENFSK_whitener_config_t::whitenEnd

Configure end-of-whitening.

See "genfskWhitenEnd_t".

1.7.36 genfskWhitenB4Crc_t GENFSK_whitener_config_t::whitenB4Crc

Configure for whitening-before-CRC.

See "genfskWhitenB4Crc_t".

1.7.37 genfskWhitenPolyType_t GENFSK_whitener_config_t::whitenPolyType

Whiten polynomial type.

See "genfskWhitenPolyType_t".

1.7.38 genfskWhitenCfgRefIn_t GENFSK_whitener_config_t::whitenRefIn

Whiten reflect input.

See "genfskWhitenCfgRefIn_t".

1.7.39 genfskWhitenCfgPayloadReinit_t GENFSK_whitener_config_t::whitenPayloadReinit

Configure for whitener re-initialization.

See "genfskWhitenCfgPayloadReinit_t".

1.7.40 uint8_t GENFSK_whitener_config_t::whitenSize

Length of whitener LFSR.

Maximum value 9.

1.7.41 uint16_t GENFSK_whitener_config_t::whitenInit

Initialization value for Whitening/De-whitening.

Maximum 9 bits.

1.7.42 uint16_t GENFSK_whitener_config_t::whitenPoly

Whitener polynomial.

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

1.7.43 uint16_t GENFSK_whitener_config_t::whitenSizeThr

Whitener size threshold.

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

1.7.44 genfskManchesterEn_t GENFSK_whitener_config_t::manchesterEn

Configure for Manchester Encoding/Decoding.

See "genfskManchesterEn_t".

1.7.45 **genfskManchesterStart_t GENFSK_whitener_config_t::manchesterStart**

Configure for inverted Manchester Encoding.

See "genfskManchesterStart_t".

1.7.46 **genfskManchesterInv_t GENFSK_whitener_config_t::manchesterInv**

Configure Manchester Encoding start point.

See "genfskManchesterInv_t".

1.7.47 **GENFSK_crc_config_t* GENFSK_bitproc_t::crcConfig**

CRC module configuration structure.

1.7.48 **GENFSK_whitener_config_t* GENFSK_bitproc_t::whitenerConfig**

Whitener module configuration structure.

1.7.49 **uint16_t GENFSK_packet_header_t::h0Field**

H0 field value.

1.7.50 **uint16_t GENFSK_packet_header_t::lengthField**

LENGTH field value.

1.7.51 **uint16_t GENFSK_packet_header_t::h1Field**

H1 field value.

1.7.52 **GENFSK_nwk_addr_t GENFSK_packet_t::addr**

Network address.

1.7.53 GENFSK_packet_header_t GENFSK_packet_t::header

Packet header data structure.

1.7.54 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, QorIQ, QorIQ Qonverge, Ready Play, SafeAssure, the SafeAssure logo, StarCore, Symphony, VortiQa, Vybrid, Airfast, BeeKit, BeeStack, CoreNet, Flexis, MXC, Platform in a Package, QUICC Engine, SMARTMOS, Tower, TurboLink, and UMEMS are trademarks of NXP B.V. All other product or service names are the property of their respective owners. ARM, AMBA, ARM Powered, Artisan, Cortex, Jazelle, Keil, SecurCore, Thumb, TrustZone, and Vision are registered trademarks of ARM Limited (or its subsidiaries) in the EU and/or elsewhere. ARM7, ARM9, ARM11, big.LITTLE, CoreLink, CoreSight, DesignStart, Mali, mbed, NEON, POP, Sensinode, Socrates, ULINK and Versatile are trademarks of ARM Limited (or its subsidiaries) in the EU and/or elsewhere. All rights reserved. Oracle and Java are registered trademarks of Oracle and/or its affiliates. The Power Architecture and Power.org word marks and the Power and Power.org logos and related marks are trademarks and service marks licensed by Power.org.

© 2019 NXP B.V.