```
/*!
 * \file      SX1278.h
 *
 * \brief     SX1278 driver implementation
 *
 * \copyright Revised BSD License, see section \ref LICENSE.
 *
 * \code
 *                ______                              _
 *               / _____)             _              | |
 *              ( (____  _____ ____ _| |_ _____  ____| |__
 *               \____ \| ___ |    (_   _) ___ |/ ___)  _ \
 *               _____) ) ____| | | || |_| ____( (___| | | |
 *              (______/|_____)_|_|_| \__)_____)\____)_| |_|
 *              (C)2013-2017 Semtech
 *
 * \endcode
 *
 * \author    Miguel Luis ( Semtech )
 *
 * \author    Gregory Cristian ( Semtech )
 */
#ifndef __SX1278_H__
#define __SX1278_H__

#include <stdint.h>
#include <stdbool.h>

#include "radio.h"
#include "SX1278Regs-Fsk.h"
#include "SX1278Regs-LoRa.h"

/*!
 * Radio wake-up time from sleep
 */
#define RADIO_WAKEUP_TIME                           1 // [ms]

/*!
 * Sync word for Private LoRa networks
 */
#define LORA_MAC_PRIVATE_SYNCWORD                   0x12

/*!
 * Sync word for Public LoRa networks
 */
#define LORA_MAC_PUBLIC_SYNCWORD                    0x34

#define RSSI_OFFSET_LF                              -164
#define RSSI_OFFSET_HF                              -157

/*!
 * Radio FSK modem parameters
 */
typedef struct
{
```

```
    int8_t   Power;
    uint32_t Fdev;
    uint32_t Bandwidth;
    uint32_t BandwidthAfc;
    uint32_t Datarate;
    uint16_t PreambleLen;
    bool     FixLen;
    uint8_t  PayloadLen;
    bool     CrcOn;
    bool     IqInverted;
    bool     RxContinuous;
    uint32_t TxTimeout;
    uint32_t RxSingleTimeout;
}RadioFskSettings_t;

/*!
 * Radio FSK packet handler state
 */
typedef struct
{
    uint8_t  PreambleDetected;
    uint8_t  SyncWordDetected;
    int8_t   RssiValue;
    int32_t  AfcValue;
    uint8_t  RxGain;
    uint16_t Size;
    uint16_t NbBytes;
    uint8_t  FifoThresh;
    uint8_t  ChunkSize;
}RadioFskPacketHandler_t;

/*!
 * Radio LoRa modem parameters
 */
typedef struct
{
    int8_t   Power;
    uint32_t Bandwidth;
    uint32_t Datarate;
    bool     LowDatarateOptimize;
    uint8_t  Coderate;
    uint16_t PreambleLen;
    bool     FixLen;
    uint8_t  PayloadLen;
    bool     CrcOn;
    bool     FreqHopOn;
    uint8_t  HopPeriod;
    bool     IqInverted;
    bool     RxContinuous;
    uint32_t TxTimeout;
    bool     PublicNetwork;
}RadioLoRaSettings_t;

/*!
 * Radio LoRa packet handler state
 */
typedef struct
{
```

```
    int8_t SnrValue;
    int16_t RssiValue;
    uint8_t Size;
}RadioLoRaPacketHandler_t;

/*!
 * Gpio
 */
typedef struct
{
  uint8_t port;
  uint8_t pin;
} Gpio_t;


/*!
 * Radio Settings
 */
typedef struct
{
    RadioState_t              State;
    RadioModems_t             Modem;
    uint32_t                  Channel;
    RadioFskSettings_t        Fsk;
    RadioFskPacketHandler_t   FskPacketHandler;
    RadioLoRaSettings_t       LoRa;
    RadioLoRaPacketHandler_t  LoRaPacketHandler;
}RadioSettings_t;

/*!
 * Radio hardware and global parameters
 */
typedef struct SX1278_s
{
    Gpio_t        Reset;
    Gpio_t        DIO0;
    Gpio_t        DIO1;
    Gpio_t        DIO2;
    Gpio_t        DIO3;
    Gpio_t        DIO4;
    Gpio_t        DIO5;
    RadioSettings_t Settings;
}SX1278_t;

/*!
 * Hardware IO IRQ callback function definition
 */
typedef void ( DioIrqHandler )();

/*!
 * SX1278 definitions
 */
#define XTAL_FREQ                                   32000000
#define FREQ_STEP                                   61.03515625

#define RX_BUFFER_SIZE                              256

/*!
```

```
 * =============================================================================
 * Public functions prototypes
 * =============================================================================
 */
void SX1278Reset(void);
/*!
 * \brief Initializes the radio
 *
 * \param [IN] events Structure containing the driver callback functions
 */
bool SX1278Init(bool clkout_switch);


/*!
 * \brief Initializes the radio, but no chip reset!!!
 *
 * \param [IN] events Structure containing the driver callback functions
 */
bool SX1278InitWithoutReset(bool clkout_switch);


/*!
 * Return current radio status
 *
 * \param status Radio status.[RF_IDLE, RF_RX_RUNNING, RF_TX_RUNNING]
 */
RadioState_t SX1278GetStatus( void );

/*!
 * \brief Configures the radio with the given modem
 *
 * \param [IN] modem Modem to be used [0: FSK, 1: LoRa]
 */
void SX1278SetModem( RadioModems_t modem );

/*!
 * \brief Sets the channel configuration
 *
 * \param [IN] freq         Channel RF frequency
 */
void SX1278SetChannel( uint32_t freq );

/*!
 * \brief Checks if the channel is free for the given time
 *
 * \param [IN] modem      Radio modem to be used [0: FSK, 1: LoRa]
 * \param [IN] freq       Channel RF frequency
 * \param [IN] rssiThresh RSSI threshold
 * \param [IN] maxCarrierSenseTime Max time while the RSSI is measured
 *
 * \retval isFree         [true: Channel is free, false: Channel is not free]
 */
bool SX1278IsChannelFree( RadioModems_t modem, uint32_t freq, int16_t
rssiThresh, uint32_t maxCarrierSenseTime );

/*!
 * \brief Generates a 32 bits random value based on the RSSI readings
 *
```

```
 * \remark This function sets the radio in LoRa modem mode and disables
 *         all interrupts.
 *         After calling this function either SX1278SetRxConfig or
 *         SX1278SetTxConfig functions must be called.
 *
 * \retval randomValue    32 bits random value
 */
uint32_t SX1278Random( void );

/*!
 * \brief Sets the reception parameters
 *
 * \remark When using LoRa modem only bandwidths 125, 250 and 500 kHz are
supported
 *
 * \param [IN] modem        Radio modem to be used [0: FSK, 1: LoRa]
 * \param [IN] bandwidth    Sets the bandwidth
 *                          FSK : >= 2600 and <= 250000 Hz
 *                          LoRa: [0: 125 kHz, 1: 250 kHz,
 *                                 2: 500 kHz, 3: Reserved]
 * \param [IN] datarate     Sets the Datarate
 *                          FSK : 600..300000 bits/s
 *                          LoRa: [6: 64, 7: 128, 8: 256, 9: 512,
 *                                10: 1024, 11: 2048, 12: 4096  chips]
 * \param [IN] coderate     Sets the coding rate (LoRa only)
 *                          FSK : N/A ( set to 0 )
 *                          LoRa: [1: 4/5, 2: 4/6, 3: 4/7, 4: 4/8]
 * \param [IN] bandwidthAfc Sets the AFC Bandwidth (FSK only)
 *                          FSK : >= 2600 and <= 250000 Hz
 *                          LoRa: N/A ( set to 0 )
 * \param [IN] preambleLen  Sets the Preamble length
 *                          FSK : Number of bytes
 *                          LoRa: Length in symbols (the hardware adds 4 more
symbols)
 * \param [IN] symbTimeout  Sets the RxSingle timeout value
 *                          FSK : timeout number of bytes
 *                          LoRa: timeout in symbols
 * \param [IN] fixLen       Fixed length packets [0: variable, 1: fixed]
 * \param [IN] payloadLen   Sets payload length when fixed length is used
 * \param [IN] crcOn        Enables/Disables the CRC [0: OFF, 1: ON]
 * \param [IN] freqHopOn    Enables disables the intra-packet frequency hopping
 *                          FSK : N/A ( set to 0 )
 *                          LoRa: [0: OFF, 1: ON]
 * \param [IN] hopPeriod    Number of symbols between each hop
 *                          FSK : N/A ( set to 0 )
 *                          LoRa: Number of symbols
 * \param [IN] iqInverted   Inverts IQ signals (LoRa only)
 *                          FSK : N/A ( set to 0 )
 *                          LoRa: [0: not inverted, 1: inverted]
 * \param [IN] rxContinuous Sets the reception in continuous mode
 *                          [false: single mode, true: continuous mode]
 */
void SX1278SetRxConfig( RadioModems_t modem, uint32_t bandwidth,
                        uint32_t datarate, uint8_t coderate,
                        uint32_t bandwidthAfc, uint16_t preambleLen,
                        uint16_t symbTimeout, bool fixLen,
                        uint8_t payloadLen,
                        bool crcOn, bool freqHopOn, uint8_t hopPeriod,
```

```
                          bool iqInverted, bool rxContinuous );

/*!
 * \brief Sets the transmission parameters
 *
 * \remark When using LoRa modem only bandwidths 125, 250 and 500 kHz are
supported
 *
 * \param [IN] modem        Radio modem to be used [0: FSK, 1: LoRa]
 * \param [IN] power        Sets the output power [dBm]
 * \param [IN] fdev         Sets the frequency deviation (FSK only)
 *                          FSK : [Hz]
 *                          LoRa: 0
 * \param [IN] bandwidth    Sets the bandwidth (LoRa only)
 *                          FSK : 0
 *                          LoRa: [0: 125 kHz, 1: 250 kHz,
 *                                 2: 500 kHz, 3: Reserved]
 * \param [IN] datarate     Sets the Datarate
 *                          FSK : 600..300000 bits/s
 *                          LoRa: [6: 64, 7: 128, 8: 256, 9: 512,
 *                                10: 1024, 11: 2048, 12: 4096  chips]
 * \param [IN] coderate     Sets the coding rate (LoRa only)
 *                          FSK : N/A ( set to 0 )
 *                          LoRa: [1: 4/5, 2: 4/6, 3: 4/7, 4: 4/8]
 * \param [IN] preambleLen  Sets the preamble length
 *                          FSK : Number of bytes
 *                          LoRa: Length in symbols (the hardware adds 4 more
symbols)
 * \param [IN] fixLen       Fixed length packets [0: variable, 1: fixed]
 * \param [IN] crcOn        Enables disables the CRC [0: OFF, 1: ON]
 * \param [IN] freqHopOn    Enables disables the intra-packet frequency hopping
 *                          FSK : N/A ( set to 0 )
 *                          LoRa: [0: OFF, 1: ON]
 * \param [IN] hopPeriod    Number of symbols between each hop
 *                          FSK : N/A ( set to 0 )
 *                          LoRa: Number of symbols
 * \param [IN] iqInverted   Inverts IQ signals (LoRa only)
 *                          FSK : N/A ( set to 0 )
 *                          LoRa: [0: not inverted, 1: inverted]
 * \param [IN] timeout      Transmission timeout [ms]
 */
void SX1278SetTxConfig( RadioModems_t modem, int8_t power, uint32_t fdev,
                        uint32_t bandwidth, uint32_t datarate,
                        uint8_t coderate, uint16_t preambleLen,
                        bool fixLen, bool crcOn, bool freqHopOn,
                        uint8_t hopPeriod, bool iqInverted, uint32_t timeout,
uint8_t paFlag );

/*!
 * \brief Computes the packet time on air in ms for the given payload
 *
 * \Remark Can only be called once SetRxConfig or SetTxConfig have been called
 *
 * \param [IN] modem      Radio modem to be used [0: FSK, 1: LoRa]
 * \param [IN] pktLen     Packet payload length
 *
 * \retval airTime        Computed airTime (ms) for the given packet payload
length
```

```
 */
uint32_t SX1278GetTimeOnAir( RadioModems_t modem, uint8_t pktLen );

/*!
 * \brief Sends the buffer of size. Prepares the packet to be sent and sets
 *        the radio in transmission
 *
 * \param [IN]: buffer     Buffer pointer
 * \param [IN]: size       Buffer size
 */
void SX1278Send( uint8_t *buffer, uint8_t size, uint8_t flag);

/*!
 * \brief Sets the radio in sleep mode
 */
void SX1278SetSleep( void );

/*!
 * \brief Sets the radio in standby mode
 */
void SX1278SetStby( void );

/*!
 * \brief Sets the radio in reception mode for the given time
 * \param [IN] timeout Reception timeout [ms] [0: continuous, others timeout]
 */
void SX1278SetRx( uint32_t timeout, uint8_t flag);

/*!
 * \brief Start a Channel Activity Detection
 */
void SX1278StartCad( void );

/*!
 * \brief Sets the radio in continuous wave transmission mode
 *
 * \param [IN]: freq       Channel RF frequency
 * \param [IN]: power      Sets the output power [dBm]
 * \param [IN]: time       Transmission mode timeout [s]
 */
void SX1278SetTxContinuousWave( uint32_t freq, int8_t power, uint16_t time );

/*!
 * \brief Reads the current RSSI value
 *
 * \retval rssiValue Current RSSI value in [dBm]
 */
int16_t SX1278ReadRssi( RadioModems_t modem );

/*!
 * \brief Writes the radio register at the specified address
 *
 * \param [IN]: addr Register address
 * \param [IN]: data New register value
 */
void SX1278WriteReg( uint8_t addr, uint8_t data );

/*!
```

```
 * \brief Reads the radio register at the specified address
 *
 * \param [IN]: addr Register address
 * \retval data Register value
 */
uint8_t SX1278ReadReg( uint8_t addr );

/*!
 * \brief Writes multiple radio registers starting at address
 *
 * \param [IN] addr   First Radio register address
 * \param [IN] buffer Buffer containing the new register's values
 * \param [IN] size   Number of registers to be written
 */
void SX1278WriteBuffer( uint8_t addr, uint8_t *buffer, uint8_t size );

/*!
 * \brief Reads multiple radio registers starting at address
 *
 * \param [IN] addr First Radio register address
 * \param [OUT] buffer Buffer where to copy the registers data
 * \param [IN] size Number of registers to be read
 */
void SX1278ReadBuffer( uint8_t addr, uint8_t *buffer, uint8_t size );

/*!
 * \brief Sets the maximum payload length.
 *
 * \param [IN] modem      Radio modem to be used [0: FSK, 1: LoRa]
 * \param [IN] max        Maximum payload length in bytes
 */
void SX1278SetMaxPayloadLength( RadioModems_t modem, uint8_t max );

/*!
 * \brief Sets the network to public or private. Updates the sync byte.
 *
 * \remark Applies to LoRa modem only
 *
 * \param [IN] enable if true, it enables a public network
 */
void SX1278SetPublicNetwork( bool enable );

/*!
 * \brief Gets the time required for the board plus radio to get out of sleep.
[ms]
 *
 * \retval time Radio plus board wakeup time in ms.
 */
uint32_t SX1278GetWakeupTime( void );

void SX1278ReadFifo(uint8_t *buffer, uint8_t size);

void SX1278OnDio0Irq(void);

void SX1278OnDio3Irq(void);

void SX1278SendCommit(void);
```

```
void SX1278SetClockOut(RadioModems_t modem, uint8_t clk_freq);

uint32_t SX1278GetTimeOnAir( RadioModems_t modem, uint8_t pktLen );

void SX1278ClearRXandValidheaderIrq(void);
#endif // __SX1278_H__
```