```
/******************************************************************************

 @file config.h

 @brief TI-15.4 Stack configuration parameters for Sensor applications

 Group: WCS LPC
 Target Device: cc13x0

 ******************************************************************************

 Copyright (c) 2016-2019, Texas Instruments Incorporated
 All rights reserved.

 Redistribution and use in source and binary forms, with or without
 modification, are permitted provided that the following conditions
 are met:

 *  Redistributions of source code must retain the above copyright
    notice, this list of conditions and the following disclaimer.

 *  Redistributions in binary form must reproduce the above copyright
    notice, this list of conditions and the following disclaimer in the
    documentation and/or other materials provided with the distribution.

 *  Neither the name of Texas Instruments Incorporated nor the names of
    its contributors may be used to endorse or promote products derived
    from this software without specific prior written permission.

 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
 AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO,
 THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR
 CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
 EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
 PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS;
 OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,
 WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR
 OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE,
 EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

 ******************************************************************************


 *****************************************************************************/
#ifndef CONFIG_H
#define CONFIG_H

/******************************************************************************
 Includes
 *****************************************************************************/
#include "api_mac.h"

#ifdef __cplusplus
extern "C"
{
#endif

/******************************************************************************
 Constants and definitions
 *****************************************************************************/
/* config parameters */
/*! Security Enable - set to true to turn on security */
#define CONFIG_SECURE                true
/*! PAN ID */
#define CONFIG_PAN_ID                0xFFFF
/*! FH disabled as default */
#define CONFIG_FH_ENABLE             false
/*! link quality */
#define CONFIG_LINKQUALITY           1
/*! percent filter */
#define CONFIG_PERCENTFILTER         0xFF

/*!
 Beacon order, value of 15 indicates non beacon mode,
 8 is a good value for beacon mode
*/
#define CONFIG_MAC_BEACON_ORDER          15
/*!
 Superframe order, value of 15 indicates non beacon mode,
 8 is a good value for beacon mode
*/
#define CONFIG_MAC_SUPERFRAME_ORDER      15
/*! Maximum number of message failure, to indicate sync loss */
#define CONFIG_MAX_DATA_FAILURES     3
/*!
 Maximum number of attempts for association in FH mode
 after reception of a PAN Config frame
 */
#define CONFIG_FH_MAX_ASSOCIATION_ATTEMPTS    3
/* Interval for scan backoff */
#define CONFIG_SCAN_BACKOFF_INTERVAL  5000
```

```
 93 /* Interval for delay between orphan notifications */
 94 #define CONFIG_ORPHAN_BACKOFF_INTERVAL 300000
 95
 96 /*! Setting for Phy ID */
 97 #define CONFIG_PHY_ID                (APIMAC_STD_US_915_PHY_1)
 98
 99 /*! MAC Parameter */
100 /*! Min BE - Minimum Backoff Exponent */
101 #define CONFIG_MIN_BE   3
102 /*! Max BE - Maximum Backoff Exponent */
103 #define CONFIG_MAX_BE   5
104 /*! MAC MAX CSMA Backoffs */
105 #define CONFIG_MAC_MAX_CSMA_BACKOFFS   4
106 /*! macMaxFrameRetries - Maximum Frame Retries */
107 #define CONFIG_MAX_RETRIES   3
108
109 #if ((CONFIG_PHY_ID >= APIMAC_MRFSK_STD_PHY_ID_BEGIN) && (CONFIG_PHY_ID <= APIMAC_MRFSK_STD_PHY_ID_END))
110 /*! Setting for channel page */
111 #define CONFIG_CHANNEL_PAGE          (APIMAC_CHANNEL_PAGE_9)
112 #elif ((CONFIG_PHY_ID >= APIMAC_MRFSK_GENERIC_PHY_ID_BEGIN) && (CONFIG_PHY_ID <= APIMAC_MRFSK_GENERIC_PHY_ID_END))
113 /*! Setting for channel page */
114 #define CONFIG_CHANNEL_PAGE          (APIMAC_CHANNEL_PAGE_10)
115 #else
116 #error "PHY ID is wrong."
117 #endif
118
119 #if (defined(CC1312R1_LAUNCHXL))
120 #if((CONFIG_PHY_ID == APIMAC_GENERIC_CHINA_433_PHY_128) || (CONFIG_PHY_ID == APIMAC_GENERIC_CHINA_LRM_433_PHY_130))
121 #error "Error: 433 MHz Operation is not supported on 1312 board!"
122 #endif
123 #endif
124
125 /*! scan duration in seconds*/
126 #define CONFIG_SCAN_DURATION         5
127
128 /*!
129  Coordinator Short Address When Operating with FH Enabled.
130  */
131 #define FH_COORD_SHORT_ADDR 0xAABB
132 /*!
133  Range Extender Mode setting.
134  The following modes are available.
135  APIMAC_NO_EXTENDER - does not have PA/LNA
136  APIMAC_HIGH_GAIN_MODE - high gain mode
137  To enable CC1190, use
138  #define CONFIG_RANGE_EXT_MODE       APIMAC_HIGH_GAIN_MODE
139 */
140 #define CONFIG_RANGE_EXT_MODE       APIMAC_NO_EXTENDER
141
142 /*! Setting Default Key*/
143 #define KEY_TABLE_DEFAULT_KEY {0x12, 0x34, 0x56, 0x78, 0x9a, 0xbc, 0xde, 0xf0,\
144                                0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00}
145
146 /*!
147  Channel mask used when CONFIG_FH_ENABLE is false.
148  Each bit indicates if the corresponding channel is to be scanned
149  First byte represents channels 0 to 7 and the last byte represents
150  channels 128 to 135.
151  For byte zero in the bit mask, LSB representing Ch0.
152  For byte 1, LSB represents Ch8 and so on.
153  e.g., 0x01 0x10 represents Ch0 and Ch12 are included.
154  The default of 0x0F represents channels 0-3 are selected.
155  APIMAC_STD_US_915_PHY_1 (50kbps/2-FSK/915MHz band) has channels 0 - 128.
156  APIMAC_STD_ETSI_863_PHY_3 (50kbps/2-FSK/863MHz band) has channels 0 - 33.
157  APIMAC_GENERIC_CHINA_433_PHY_128 (50kbps/2-FSK/433MHz band) has channels 0 - 6.
158 */
159 #define CONFIG_CHANNEL_MASK          { 0x0F, 0x00, 0x00, 0x00, 0x00, 0x00, \
160                                        0x00, 0x00, 0x00, 0x00, 0x00, 0x00, \
161                                        0x00, 0x00, 0x00, 0x00, 0x00 }
162 /*!
163  Channel mask used when CONFIG_FH_ENABLE is true.
164  Represents the list of channels on which the device can hop.
165  When CONFIG_RX_ON_IDLE is true, the actual sequence will
166  be based on DH1CF function. When it is set to false, the sequence
167  shall be a linear hopping over available channels in ascending order and
168  shall be used to change channel during the join phase.
169  It is represented as a bit string with LSB representing Ch0.
170  e.g., 0x01 0x10 represents Ch0 and Ch12 are included.
171  */
172 #define CONFIG_FH_CHANNEL_MASK       { 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, \
173                                        0xFF, 0x00, 0x00, 0x00, 0x00, 0x00, \
174                                        0x00, 0x00, 0x00, 0x00, 0x00,}
175 /* FH related config variables */
176 /*!
177  List of channels to target the Async frames
178  It is represented as a bit string with LSB representing Ch0
179  e.g., 0x01 0x10 represents Ch0 and Ch12 are included
180  It should cover all channels that could be used by a target device in its
181  hopping sequence. Channels marked beyond number of channels supported by
182  PHY Config will be excluded by stack. To avoid interference on a channel,
183  it should be removed from Async Mask and added to exclude channels
184  (CONFIG_CHANNEL_MASK).
```

```
 */
#define FH_ASYNC_CHANNEL_MASK         { 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, \
                                        0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, \
                                        0xFF, 0xFF, 0xFF, 0xFF, 0xFF }

/*! Rx on when idle, false for sleepy device, true for non sleepy device */
#define CONFIG_RX_ON_IDLE          false

/*!
 The number of non sleepy channel hopping end devices to be supported.
 It is to be noted that the total number of non sleepy devices supported
  must be less than 50. Stack will allocate memory proportional
 to the number of end devices requested.
 */
#define FH_NUM_NON_SLEEPY_HOPPING_NEIGHBORS  2
/*!
 The number of non sleepy fixed channel end devices to be supported.
 It is to be noted that the total number of non sleepy devices supported
  must be less than 50. Stack will allocate memory proportional
 to the number of end devices requested.
 */
#define FH_NUM_NON_SLEEPY_FIXED_CHANNEL_NEIGHBORS  2

/*!
 Dwell Time: The duration for which a non sleepy end device shall
 stay on a specific channel before hopping to next channel.
 */
#define CONFIG_DWELL_TIME            250

#if (((CONFIG_PHY_ID >= APIMAC_MRFSK_STD_PHY_ID_BEGIN) && (CONFIG_PHY_ID <= APIMAC_MRFSK_GENERIC_PHY_ID_BEGIN)) || \
    ((CONFIG_PHY_ID >= APIMAC_GENERIC_US_915_PHY_132) && (CONFIG_PHY_ID <= APIMAC_GENERIC_ETSI_863_PHY_133)))
/*! Default Polling interval in milliseconds. It will get updated upon reception
 of a config request message */
#define CONFIG_POLLING_INTERVAL      6000
/*! PAN Advertisement Solicit trickle timer duration in milliseconds */
#define CONFIG_PAN_ADVERT_SOLICIT_CLK_DURATION    6000
/*! PAN Config Solicit trickle timer duration in milliseconds */
#define CONFIG_PAN_CONFIG_SOLICIT_CLK_DURATION    6000
/*! Default Reporting Interval - in milliseconds. It will get updated upon
 reception of a config request message */
#define CONFIG_REPORTING_INTERVAL  180000
#else
/*! Default Polling interval in milliseconds. It will get updated upon reception
 of a config request message */
#define CONFIG_POLLING_INTERVAL      60000
/*! PAN Advertisement Solicit trickle timer duration in milliseconds */
#define CONFIG_PAN_ADVERT_SOLICIT_CLK_DURATION    60000
/*! PAN Config Solicit trickle timer duration in milliseconds */
#define CONFIG_PAN_CONFIG_SOLICIT_CLK_DURATION    60000
/*! Default Reporting Interval - in milliseconds. It will get updated upon
 reception of a config request message */
#define CONFIG_REPORTING_INTERVAL  600000
#endif

/*! FH Poll/Sensor msg start time randomization window */
#define CONFIG_FH_START_POLL_DATA_RAND_WINDOW   10000

/*! If enabled, the periodic sensor message shall be sent as a fixed size
 * packet of specified size. If set to 0, the periodic sensor message shall be
 * of type sensor data specified in smsgs.h
 */
#define SENSOR_TEST_RAMP_DATA_SIZE   0

/*! value for ApiMac_FHAttribute_netName */
#define CONFIG_FH_NETNAME            {"FHTest"}

/*! Range Extender is not supported in uBLE project  */
#ifdef FEATURE_UBLE
#if CONFIG_RANGE_EXT_MODE
#error "CONFIG_RANGE_EXT_MODE should be APIMAC_NO_EXTENDER"
#endif
#endif

/*!
 Value for Transmit Power in dBm
 For US and ETSI band, Default value is 10, allowed values are
 -10, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12 and 14dBm.
 For China band, allowed values are 6, 10, 13, 14 and 15dBm.
 For CC1190, allowed values are between 18, 23, 25, 26 and 27dBm.
 When the nodes in the network are close to each other
 lowering this value will help reduce saturation */
#ifndef DeviceFamily_CC13X2
#if CONFIG_RANGE_EXT_MODE
#define CONFIG_TRANSMIT_POWER        26
#else
#if ((CONFIG_PHY_ID == APIMAC_GENERIC_CHINA_433_PHY_128) || (CONFIG_PHY_ID == APIMAC_GENERIC_CHINA_LRM_433_PHY_130))
#define CONFIG_TRANSMIT_POWER        14
#else
#define CONFIG_TRANSMIT_POWER        12
#endif
#endif
#else /* DeviceFamily_CC13X2 */
```

```
277 #define CONFIG_TRANSMIT_POWER        12
278 #endif
279
280 #ifndef DeviceFamily_CC13X2
281 #if CONFIG_RANGE_EXT_MODE
282 #if (CCFG_FORCE_VDDR_HH == 1)
283 #error "CCFG_FORCE_VDDR_HH should be 0"
284 #endif
285 #else
286 #if ((CONFIG_PHY_ID == APIMAC_GENERIC_CHINA_433_PHY_128) || (CONFIG_PHY_ID == APIMAC_GENERIC_CHINA_LRM_433_PHY_130))
287 #if (CCFG_FORCE_VDDR_HH == 0)
288 #if (CONFIG_TRANSMIT_POWER >= 15)
289 #error "CONFIG_TRANSMIT_POWER should be less than 15"
290 #endif
291 #else
292 #if (CONFIG_TRANSMIT_POWER < 15)
293 /* In 433 MHz band when CCFG_FORCE_VDDR_HH = 1, only possible value of transmit power is 15 */
294 #error "CONFIG_TRANSMIT_POWER should be 15"
295 #endif
296 #endif
297 #else
298 #if (CCFG_FORCE_VDDR_HH == 0)
299 #if (CONFIG_TRANSMIT_POWER >= 14)
300 #error "CONFIG_TRANSMIT_POWER should be less than 14"
301 #endif
302 #else
303 #if (CONFIG_TRANSMIT_POWER < 14)
304 /* In US and ETSI band when CCFG_FORCE_VDDR_HH = 1, only possible value of transmit power is 14 */
305 #error "CONFIG_TRANSMIT_POWER should be 14"
306 #endif
307 #endif
308 #endif
309 #endif
310 #else
311 #if (CCFG_FORCE_VDDR_HH == 1)
312 #if (CONFIG_TRANSMIT_POWER != 14)
313 /* In US and ETSI band when CCFG_FORCE_VDDR_HH = 1, only possible value of transmit power is 14 */
314 #error "CONFIG_TRANSMIT_POWER should be 14"
315 #endif
316 #endif
317 #endif
318
319 /*!
320 * Enable this mode for certfication.
321 * For FH certification, CONFIG_FH_ENABLE should
322 * also be enabled
323 */
324 #define CERTIFICATION_TEST_MODE     false
325
326 #ifdef POWER_MEAS
327 /*!
328  Power profile to be used when Power MEAS is enabled.
329  Profile 1 - POLL_ACK - Polling Only
330  Profile 2 - DATA_ACK - 20 byte application data + ACK from sensor to collector
331  Profile 3 - POLL_DATA - Poll + received Data from collector
332  Profile 4 - SLEEP - No Poll or Data. In Beacon mode, beacon RX would occur
333  */
334 #define POWER_TEST_PROFILE  DATA_ACK
335 #endif
336
337 /* Check if all the necessary parameters have been set for FH mode */
338 #if CONFIG_FH_ENABLE
339 #if !defined(FEATURE_ALL_MODES) && !defined(FEATURE_FREQ_HOP_MODE)
340 #error "Do you want to build image with frequency hopping mode? \
341         Define either FEATURE_FREQ_HOP_MODE or FEATURE_ALL_MODES in features.h"
342 #endif
343 #endif
344
345 /* Check if stack level security is enabled if application security is enabled */
346 #if CONFIG_SECURE
347 #if !defined(FEATURE_MAC_SECURITY)
348 #error "Define FEATURE_MAC_SECURITY or FEATURE_ALL_MODES in features.h to \
349         be able to use security at application level"
350 #endif
351 #endif
352
353 /* Set beacon order and superframe order to 15 for FH mode to avoid user error */
354 #if CONFIG_FH_ENABLE
355 #if (CONFIG_MAC_BEACON_ORDER != 15) && (CONFIG_MAC_SUPERFRAME_ORDER != 15)
356 #error "Do you want to build image with frequency hopping mode? \
357     If yes, CONFIG_MAC_BEACON_ORDER and CONFIG_MAC_SUPERFRAME_ORDER \
358     should both be set to 15"
359 #endif
360 #if (FH_NUM_NON_SLEEPY_HOPPING_NEIGHBORS < 2) || (FH_NUM_NON_SLEEPY_FIXED_CHANNEL_NEIGHBORS < 2)
361 #error "You have an invalid value for FH neighbors. Set the values \
362         for FH_NUM_NON_SLEEPY_HOPPING_NEIGHBORS and FH_NUM_NON_SLEEPY_FIXED_CHANNEL_NEIGHBORS to at least 2"
363 #endif
364 #endif
365
366 #ifdef __cplusplus
367 }
368 #endif
```

```
369
370 #endif /* CONFIG_H */
371
372
```