**Date Submitted: 12/13/19**

**Youtube Link:**

https://www.youtube.com/watch?v=ESH3\_bZ6c8Y

**------------------------------------------------------------------------------------**

**CC1350 TI-RTOS/RF Lab 05**

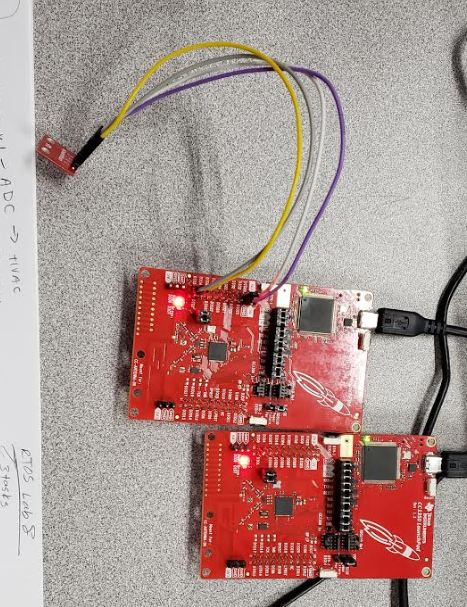
In this lab, my partner and I were to program a star network of sensor nodes. That is have one CC1350 board be the sensor board which reads values from the sensor and the other CC1350 board will be the Collector, receiving temperature data. By following the lab tutorials both my partner and I were able to program our boards.

The steps were as followed:

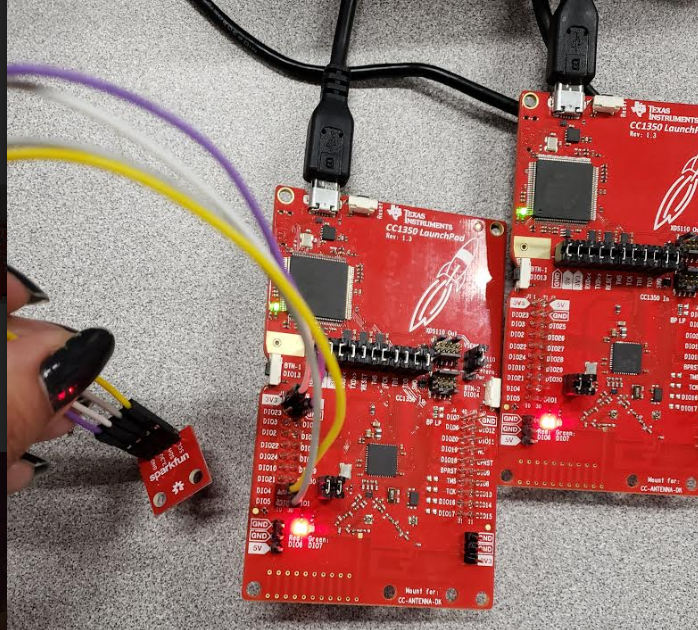
1. Build and load the collector example
2. Build and load the sensor example
3. Use the collector and sensor
4. Update sensor data
5. Combine portable app with TI 15.4 Stack app
6. Use stack to send temperature to portable app
7. We can then see our CC1350s’ working successfully

For this lab, you just had to follow the instructions.

**Modified Schematic:**

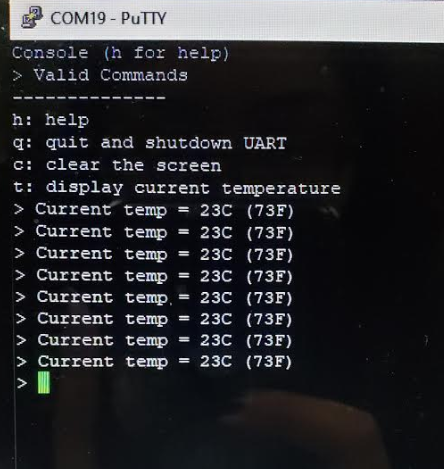


**CC1350 Board Connections**

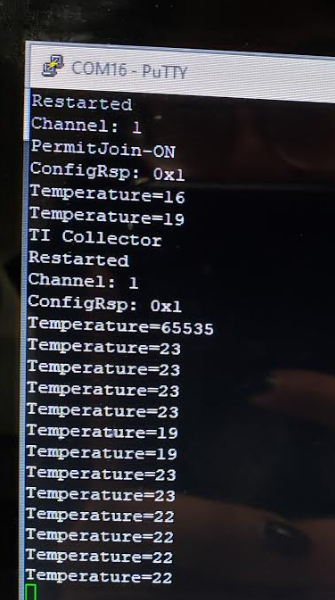


**As one can see this is the sensor board. The temperature sensor is connected to it.**

**Results:**



**This is the sensor data that will be sending it to the collector**



**This is what the collector board is receiving from the sensor board**

**Modified Code:**

**Several files were involved see GitHub for better formatting**

**Sensor Code:**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**@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**

**/\* Interval for delay between orphan notifications \*/**

**#define CONFIG\_ORPHAN\_BACKOFF\_INTERVAL 300000**

**/\*! Setting for Phy ID \*/**

**#define CONFIG\_PHY\_ID (APIMAC\_STD\_US\_915\_PHY\_1)**

**/\*! MAC Parameter \*/**

**/\*! Min BE - Minimum Backoff Exponent \*/**

**#define CONFIG\_MIN\_BE 3**

**/\*! Max BE - Maximum Backoff Exponent \*/**

**#define CONFIG\_MAX\_BE 5**

**/\*! MAC MAX CSMA Backoffs \*/**

**#define CONFIG\_MAC\_MAX\_CSMA\_BACKOFFS 4**

**/\*! macMaxFrameRetries - Maximum Frame Retries \*/**

**#define CONFIG\_MAX\_RETRIES 3**

**#if ((CONFIG\_PHY\_ID >= APIMAC\_MRFSK\_STD\_PHY\_ID\_BEGIN) && (CONFIG\_PHY\_ID <= APIMAC\_MRFSK\_STD\_PHY\_ID\_END))**

**/\*! Setting for channel page \*/**

**#define CONFIG\_CHANNEL\_PAGE (APIMAC\_CHANNEL\_PAGE\_9)**

**#elif ((CONFIG\_PHY\_ID >= APIMAC\_MRFSK\_GENERIC\_PHY\_ID\_BEGIN) && (CONFIG\_PHY\_ID <= APIMAC\_MRFSK\_GENERIC\_PHY\_ID\_END))**

**/\*! Setting for channel page \*/**

**#define CONFIG\_CHANNEL\_PAGE (APIMAC\_CHANNEL\_PAGE\_10)**

**#else**

**#error "PHY ID is wrong."**

**#endif**

**#if (defined(CC1312R1\_LAUNCHXL))**

**#if((CONFIG\_PHY\_ID == APIMAC\_GENERIC\_CHINA\_433\_PHY\_128) || (CONFIG\_PHY\_ID == APIMAC\_GENERIC\_CHINA\_LRM\_433\_PHY\_130))**

**#error "Error: 433 MHz Operation is not supported on 1312 board!"**

**#endif**

**#endif**

**/\*! scan duration in seconds\*/**

**#define CONFIG\_SCAN\_DURATION 5**

**/\*!**

**Coordinator Short Address When Operating with FH Enabled.**

**\*/**

**#define FH\_COORD\_SHORT\_ADDR 0xAABB**

**/\*!**

**Range Extender Mode setting.**

**The following modes are available.**

**APIMAC\_NO\_EXTENDER - does not have PA/LNA**

**APIMAC\_HIGH\_GAIN\_MODE - high gain mode**

**To enable CC1190, use**

**#define CONFIG\_RANGE\_EXT\_MODE APIMAC\_HIGH\_GAIN\_MODE**

**\*/**

**#define CONFIG\_RANGE\_EXT\_MODE APIMAC\_NO\_EXTENDER**

**/\*! Setting Default Key\*/**

**#define KEY\_TABLE\_DEFAULT\_KEY {0x12, 0x34, 0x56, 0x78, 0x9a, 0xbc, 0xde, 0xf0,\**

**0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00}**

**/\*!**

**Channel mask used when CONFIG\_FH\_ENABLE is false.**

**Each bit indicates if the corresponding channel is to be scanned**

**First byte represents channels 0 to 7 and the last byte represents**

**channels 128 to 135.**

**For byte zero in the bit mask, LSB representing Ch0.**

**For byte 1, LSB represents Ch8 and so on.**

**e.g., 0x01 0x10 represents Ch0 and Ch12 are included.**

**The default of 0x0F represents channels 0-3 are selected.**

**APIMAC\_STD\_US\_915\_PHY\_1 (50kbps/2-FSK/915MHz band) has channels 0 - 128.**

**APIMAC\_STD\_ETSI\_863\_PHY\_3 (50kbps/2-FSK/863MHz band) has channels 0 - 33.**

**APIMAC\_GENERIC\_CHINA\_433\_PHY\_128 (50kbps/2-FSK/433MHz band) has channels 0 - 6.**

**\*/**

**#define CONFIG\_CHANNEL\_MASK { 0x0F, 0x00, 0x00, 0x00, 0x00, 0x00, \**

**0x00, 0x00, 0x00, 0x00, 0x00, 0x00, \**

**0x00, 0x00, 0x00, 0x00, 0x00 }**

**/\*!**

**Channel mask used when CONFIG\_FH\_ENABLE is true.**

**Represents the list of channels on which the device can hop.**

**When CONFIG\_RX\_ON\_IDLE is true, the actual sequence will**

**be based on DH1CF function. When it is set to false, the sequence**

**shall be a linear hopping over available channels in ascending order and**

**shall be used to change channel during the join phase.**

**It is represented as a bit string with LSB representing Ch0.**

**e.g., 0x01 0x10 represents Ch0 and Ch12 are included.**

**\*/**

**#define CONFIG\_FH\_CHANNEL\_MASK { 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, \**

**0xFF, 0x00, 0x00, 0x00, 0x00, 0x00, \**

**0x00, 0x00, 0x00, 0x00, 0x00,}**

**/\* FH related config variables \*/**

**/\*!**

**List of channels to target the Async frames**

**It is represented as a bit string with LSB representing Ch0**

**e.g., 0x01 0x10 represents Ch0 and Ch12 are included**

**It should cover all channels that could be used by a target device in its**

**hopping sequence. Channels marked beyond number of channels supported by**

**PHY Config will be excluded by stack. To avoid interference on a channel,**

**it should be removed from Async Mask and added to exclude channels**

**(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 500**

**#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 6000 //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 \*/**

**#define CONFIG\_TRANSMIT\_POWER 12**

**#endif**

**#ifndef DeviceFamily\_CC13X2**

**#if CONFIG\_RANGE\_EXT\_MODE**

**#if (CCFG\_FORCE\_VDDR\_HH == 1)**

**#error "CCFG\_FORCE\_VDDR\_HH should be 0"**

**#endif**

**#else**

**#if ((CONFIG\_PHY\_ID == APIMAC\_GENERIC\_CHINA\_433\_PHY\_128) || (CONFIG\_PHY\_ID == APIMAC\_GENERIC\_CHINA\_LRM\_433\_PHY\_130))**

**#if (CCFG\_FORCE\_VDDR\_HH == 0)**

**#if (CONFIG\_TRANSMIT\_POWER >= 15)**

**#error "CONFIG\_TRANSMIT\_POWER should be less than 15"**

**#endif**

**#else**

**#if (CONFIG\_TRANSMIT\_POWER < 15)**

**/\* In 433 MHz band when CCFG\_FORCE\_VDDR\_HH = 1, only possible value of transmit power is 15 \*/**

**#error "CONFIG\_TRANSMIT\_POWER should be 15"**

**#endif**

**#endif**

**#else**

**#if (CCFG\_FORCE\_VDDR\_HH == 0)**

**#if (CONFIG\_TRANSMIT\_POWER >= 14)**

**#error "CONFIG\_TRANSMIT\_POWER should be less than 14"**

**#endif**

**#else**

**#if (CONFIG\_TRANSMIT\_POWER < 14)**

**/\* In US and ETSI band when CCFG\_FORCE\_VDDR\_HH = 1, only possible value of transmit power is 14 \*/**

**#error "CONFIG\_TRANSMIT\_POWER should be 14"**

**#endif**

**#endif**

**#endif**

**#endif**

**#else**

**#if (CCFG\_FORCE\_VDDR\_HH == 1)**

**#if (CONFIG\_TRANSMIT\_POWER != 14)**

**/\* In US and ETSI band when CCFG\_FORCE\_VDDR\_HH = 1, only possible value of transmit power is 14 \*/**

**#error "CONFIG\_TRANSMIT\_POWER should be 14"**

**#endif**

**#endif**

**#endif**

**/\*!**

**\* Enable this mode for certfication.**

**\* For FH certification, CONFIG\_FH\_ENABLE should**

**\* also be enabled**

**\*/**

**#define CERTIFICATION\_TEST\_MODE false**

**#ifdef POWER\_MEAS**

**/\*!**

**Power profile to be used when Power MEAS is enabled.**

**Profile 1 - POLL\_ACK - Polling Only**

**Profile 2 - DATA\_ACK - 20 byte application data + ACK from sensor to collector**

**Profile 3 - POLL\_DATA - Poll + received Data from collector**

**Profile 4 - SLEEP - No Poll or Data. In Beacon mode, beacon RX would occur**

**\*/**

**#define POWER\_TEST\_PROFILE DATA\_ACK**

**#endif**

**/\* Check if all the necessary parameters have been set for FH mode \*/**

**#if CONFIG\_FH\_ENABLE**

**#if !defined(FEATURE\_ALL\_MODES) && !defined(FEATURE\_FREQ\_HOP\_MODE)**

**#error "Do you want to build image with frequency hopping mode? \**

**Define either FEATURE\_FREQ\_HOP\_MODE or FEATURE\_ALL\_MODES in features.h"**

**#endif**

**#endif**

**/\* Check if stack level security is enabled if application security is enabled \*/**

**#if CONFIG\_SECURE**

**#if !defined(FEATURE\_MAC\_SECURITY)**

**#error "Define FEATURE\_MAC\_SECURITY or FEATURE\_ALL\_MODES in features.h to \**

**be able to use security at application level"**

**#endif**

**#endif**

**/\* Set beacon order and superframe order to 15 for FH mode to avoid user error \*/**

**#if CONFIG\_FH\_ENABLE**

**#if (CONFIG\_MAC\_BEACON\_ORDER != 15) && (CONFIG\_MAC\_SUPERFRAME\_ORDER != 15)**

**#error "Do you want to build image with frequency hopping mode? \**

**If yes, CONFIG\_MAC\_BEACON\_ORDER and CONFIG\_MAC\_SUPERFRAME\_ORDER \**

**should both be set to 15"**

**#endif**

**#if (FH\_NUM\_NON\_SLEEPY\_HOPPING\_NEIGHBORS < 2) || (FH\_NUM\_NON\_SLEEPY\_FIXED\_CHANNEL\_NEIGHBORS < 2)**

**#error "You have an invalid value for FH neighbors. Set the values \**

**for FH\_NUM\_NON\_SLEEPY\_HOPPING\_NEIGHBORS and FH\_NUM\_NON\_SLEEPY\_FIXED\_CHANNEL\_NEIGHBORS to at least 2"**

**#endif**

**#endif**

**#ifdef \_\_cplusplus**

**}**

**#endif**

**#endif /\* CONFIG\_H \*/**

**Maintirtos.c**

**/\***

**\* 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.**

**\*/**

**/\***

**\* ======== main\_tirtos.c ========**

**\*/**

**#include <stdint.h>**

**/\* POSIX Header files \*/**

**#include <pthread.h>**

**/\* RTOS header files \*/**

**#include <ti/sysbios/BIOS.h>**

**/\* Driver header files \*/**

**#include <ti/drivers/GPIO.h>**

**/\* Example/Board Header files \*/**

**#include <ti/drivers/Board.h>**

**/\* Mutex to protect the reading/writing of the temperature variables \*/**

**pthread\_mutex\_t temperatureMutex;**

**extern void \*temperatureThread(void \*arg0);**

**extern void \*consoleThread(void \*arg0);**

**/\* Stack size in bytes. Large enough in case debug kernel is used. \*/**

**#define THREADSTACKSIZE 1024**

**/\***

**\* ======== main ========**

**\*/**

**int main\_app(void)**

**{**

**pthread\_t thread;**

**pthread\_attr\_t attrs;**

**struct sched\_param priParam;**

**int retc;**

**/\* Call driver init functions \*/**

**// Board\_init();**

**/\* Initialize the attributes structure with default values \*/**

**pthread\_attr\_init(&attrs);**

**/\* Set priority, detach state, and stack size attributes \*/**

**priParam.sched\_priority = 1;**

**retc = pthread\_attr\_setschedparam(&attrs, &priParam);**

**retc |= pthread\_attr\_setdetachstate(&attrs, PTHREAD\_CREATE\_DETACHED);**

**retc |= pthread\_attr\_setstacksize(&attrs, THREADSTACKSIZE);**

**if (retc != 0) {**

**/\* failed to set attributes \*/**

**while (1) {}**

**}**

**retc = pthread\_create(&thread, &attrs, consoleThread, NULL);**

**if (retc != 0) {**

**/\* pthread\_create() failed \*/**

**while (1) {}**

**}**

**/\***

**\* Let's make the temperature thread a higher priority .**

**\* Higher number means higher priority in TI-RTOS.**

**\*/**

**priParam.sched\_priority = 2;**

**retc = pthread\_attr\_setschedparam(&attrs, &priParam);**

**if (retc != 0) {**

**/\* failed to set priority \*/**

**while (1) {}**

**}**

**retc = pthread\_create(&thread, &attrs, temperatureThread, NULL);**

**if (retc != 0) {**

**/\* pthread\_create() failed \*/**

**while (1) {}**

**}**

**/\* Create a mutex that will protect temperature variables \*/**

**retc = pthread\_mutex\_init(&temperatureMutex, NULL);**

**if (retc != 0) {**

**/\* pthread\_mutex\_init() failed \*/**

**while (1) {}**

**}**

**/\* Initialize the GPIO since multiple threads are using it \*/**

**// GPIO\_init();**

**/\* Start the TI-RTOS scheduler \*/**

**// BIOS\_start();**

**return (0);**

**}**

**console.c**

**/\***

**\* 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.**

**\*/**

**/\***

**\* ======== console.c ========**

**\*/**

**#include <stdint.h>**

**#include <string.h>**

**#include <stdbool.h>**

**/\* POSIX Header files \*/**

**#include <pthread.h>**

**#include <semaphore.h>**

**/\* Driver Header files \*/**

**#include <ti/drivers/GPIO.h>**

**#include <ti/drivers/UART.h>**

**#ifdef CC32XX**

**#include <ti/drivers/Power.h>**

**#include <ti/drivers/power/PowerCC32XX.h>**

**#endif**

**/\* Example/Board Header files \*/**

**#include "Board.h"**

**#include "smsgs.h"**

**#include "mac\_util.h"**

**#include "api\_mac.h"**

**#include "sensor.h"**

**extern Smsgs\_tempSensorField\_t tempSensor;**

**/\* Console display strings \*/**

**const char consoleDisplay[] = "\fConsole (h for help)\r\n";**

**const char helpPrompt[] = "Valid Commands\r\n" \**

**"--------------\r\n" \**

**"h: help\r\n" \**

**"q: quit and shutdown UART\r\n" \**

**"c: clear the screen\r\n" \**

**"t: display current temperature\r\n";**

**const char byeDisplay[] = "Bye! Hit button1 to start UART again\r\n";**

**const char tempStartDisplay[] = "Current temp = ";**

**const char tempMidDisplay[] = "C (";**

**const char tempEndDisplay[] = "F)\r\n";**

**const char cleanDisplay[] = "\f";**

**const char userPrompt[] = "> ";**

**const char readErrDisplay[] = "Problem read UART.\r\n";**

**/\* Used to determine whether to have the thread block \*/**

**volatile bool uartEnabled = true;**

**sem\_t semConsole;**

**/\* Temperature written by the temperature thread and read by console thread \*/**

**extern volatile float temperature;**

**extern volatile float temperaturef;**

**/\* Mutex to protect the reading/writing of the float temperature \*/**

**extern pthread\_mutex\_t temperatureMutex;**

**/\* Used itoa instead of sprintf to help minimize the size of the stack \*/**

**static void itoa(int n, char s[]);**

**/\***

**\* ======== gpioButtonFxn ========**

**\* Callback function for the GPIO interrupt on Board\_GPIO\_BUTTON1.**

**\* There is no debounce logic here since we are just looking for**

**\* a button push. The uartEnabled variable protects use against any**

**\* additional interrupts cased by the bouncing of the button.**

**\*/**

**void gpioButtonFxn(uint\_least8\_t index)**

**{**

**/\* If disabled, enable and post the semaphore \*/**

**if (uartEnabled == false) {**

**uartEnabled = true;**

**sem\_post(&semConsole);**

**}**

**}**

**/\***

**\* ======== simpleConsole ========**

**\* Handle the user input. Currently this console does not handle**

**\* user back-spaces or other "hard" characters.**

**\*/**

**void simpleConsole(UART\_Handle uart)**

**{**

**char cmd;**

**int status;**

**char tempStr[8];**

**int localTemperatureC;**

**int localTemperatureF;**

**UART\_write(uart, consoleDisplay, sizeof(consoleDisplay));**

**/\* Loop until read fails or user quits \*/**

**while (1) {**

**UART\_write(uart, userPrompt, sizeof(userPrompt));**

**status = UART\_read(uart, &cmd, sizeof(cmd));**

**if (status == 0) {**

**UART\_write(uart, readErrDisplay, sizeof(readErrDisplay));**

**cmd = 'q';**

**}**

**switch (cmd) {**

**case 't':**

**tempSensor.objectTemp = localTemperatureC;**

**tempSensor.ambienceTemp = localTemperatureC;**

**Util\_setEvent(&Sensor\_events, EXT\_SENSOR\_READING\_TIMEOUT\_EVT);**

**UART\_write(uart, tempStartDisplay, sizeof(tempStartDisplay));**

**/\***

**\* Make sure we are accessing the global float temperature variables**

**\* in a thread-safe manner.**

**\*/**

**pthread\_mutex\_lock(&temperatureMutex);**

**localTemperatureC = (int)temperature;**

**localTemperatureF = (int)temperaturef;**

**pthread\_mutex\_unlock(&temperatureMutex);**

**itoa((int)localTemperatureC, tempStr);**

**UART\_write(uart, tempStr, strlen(tempStr));**

**UART\_write(uart, tempMidDisplay, sizeof(tempMidDisplay));**

**itoa((int)localTemperatureF, tempStr);**

**UART\_write(uart, tempStr, strlen(tempStr));**

**UART\_write(uart, tempEndDisplay, sizeof(tempEndDisplay));**

**break;**

**case 'c':**

**UART\_write(uart, cleanDisplay, sizeof(cleanDisplay));**

**break;**

**case 'q':**

**UART\_write(uart, byeDisplay, sizeof(byeDisplay));**

**return;**

**case 'h':**

**default:**

**UART\_write(uart, helpPrompt, sizeof(helpPrompt));**

**break;**

**}**

**}**

**}**

**/\***

**\* ======== consoleThread ========**

**\*/**

**void \*consoleThread(void \*arg0)**

**{**

**UART\_Params uartParams;**

**UART\_Handle uart;**

**int retc;**

**#ifdef CC32XX**

**/\***

**\* The CC3220 examples by default do not have power management enabled.**

**\* This allows a better debug experience. With the power management**

**\* enabled, if the device goes into a low power mode the emulation**

**\* session is lost.**

**\* Let's enable it and also configure the button to wake us up.**

**\*/**

**PowerCC32XX\_Wakeup wakeup;**

**PowerCC32XX\_getWakeup(&wakeup);**

**wakeup.wakeupGPIOFxnLPDS = gpioButtonFxn;**

**PowerCC32XX\_configureWakeup(&wakeup);**

**Power\_enablePolicy();**

**#endif**

**/\* Configure the button pin \*/**

**GPIO\_setConfig(Board\_GPIO\_BUTTON1, GPIO\_CFG\_IN\_PU | GPIO\_CFG\_IN\_INT\_FALLING);**

**/\* install Button callback and enable it \*/**

**GPIO\_setCallback(Board\_GPIO\_BUTTON1, gpioButtonFxn);**

**GPIO\_enableInt(Board\_GPIO\_BUTTON1);**

**retc = sem\_init(&semConsole, 0, 0);**

**if (retc == -1) {**

**while (1);**

**}**

**UART\_init();**

**/\***

**\* Initialize the UART parameters outside the loop. Let's keep**

**\* most of the defaults (e.g. baudrate = 115200) and only change the**

**\* following.**

**\*/**

**UART\_Params\_init(&uartParams);**

**uartParams.writeDataMode = UART\_DATA\_BINARY;**

**uartParams.readDataMode = UART\_DATA\_BINARY;**

**uartParams.readReturnMode = UART\_RETURN\_FULL;**

**/\* Loop forever to start the console \*/**

**while (1) {**

**if (uartEnabled == false) {**

**retc = sem\_wait(&semConsole);**

**if (retc == -1) {**

**while (1);**

**}**

**}**

**/\* Create a UART for the console \*/**

**uart = UART\_open(Board\_UART0, &uartParams);**

**if (uart == NULL) {**

**while (1);**

**}**

**simpleConsole(uart);**

**/\***

**\* Since we returned from the console, we need to close the UART.**

**\* The Power Manager will go into a lower power mode when the UART**

**\* is closed.**

**\*/**

**UART\_close(uart);**

**uartEnabled = false;**

**}**

**}**

**/\***

**\* The following function is from good old K & R.**

**\*/**

**static void reverse(char s[])**

**{**

**int i, j;**

**char c;**

**for (i = 0, j = strlen(s)-1; i<j; i++, j--) {**

**c = s[i];**

**s[i] = s[j];**

**s[j] = c;**

**}**

**}**

**/\***

**\* The following function is from good old K & R.**

**\*/**

**static void itoa(int n, char s[])**

**{**

**int i, sign;**

**if ((sign = n) < 0) /\* record sign \*/**

**n = -n; /\* make n positive \*/**

**i = 0;**

**do { /\* generate digits in reverse order \*/**

**s[i++] = n % 10 + '0'; /\* get next digit \*/**

**} while ((n /= 10) > 0); /\* delete it \*/**

**if (sign < 0)**

**s[i++] = '-';**

**s[i] = '\0';**

**reverse(s);**

**}**

**temperature.c**

**/\***

**\* 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.**

**\*/**

**/\***

**\* ======== temperature.c ========**

**\*/**

**#include <stdint.h>**

**#include <stddef.h>**

**#include <unistd.h>**

**#include <ti/display/Display.h>**

**#include <ti/display/Display.h>**

**/\* POSIX Header files \*/**

**#include <pthread.h>**

**#include <semaphore.h>**

**#include <signal.h>**

**#include <time.h>**

**/\* Driver Header files \*/**

**#include <ti/drivers/GPIO.h>**

**#include <ti/drivers/I2C.h>**

**/\* Example/Board Header files \*/**

**#include "Board.h"**

**/\* ======== Si7021 Registers ======== \*/**

**#define Si7021\_TMP\_REG 0xE3**

**#define Si7021\_HUM\_REG 0xE5**

**#define Si7021\_ADDR 0x40**

**/\***

**\* ======== HIGH\_TEMP ========**

**\* Send alert when this temperature (in Celsius) is exceeded**

**\*/**

**#define HIGH\_TEMP 30**

**/\***

**\* ======== TMP Registers ========**

**\*/**

**#define TMP006\_REG 0x0001 /\* Die Temp Result Register for TMP006 \*/**

**#define TMP116\_REG 0x0000 /\* Die Temp Result Register for TMP116 \*/**

**/\***

**\* The CC32XX LaunchPads come with an on-board TMP006 or TMP116 temperature**

**\* sensor depending on the revision. Newer revisions come with the TMP116.**

**\* The Build Automation Sensors (BOOSTXL-BASSENSORS) BoosterPack**

**\* contains a TMP116.**

**\***

**\* We are using the DIE temperature because it's cool!**

**\***

**\* Additionally: no calibration is being done on the TMPxxx device to simplify**

**\* the example code.**

**\*/**

**#define TMP006\_ADDR 0x41;**

**#define TMP116\_BP\_ADDR 0x48;**

**#define TMP116\_LP\_ADDR 0x49;**

**/\* Temperature written by the temperature thread and read by console thread \*/**

**volatile float temperatureC;**

**volatile float temperatureF;**

**volatile float temperaturef;**

**volatile float temperature;**

**volatile float temp;**

**volatile float sample;**

**Display\_Handle display;**

**/\* Mutex to protect the reading/writing of the temperature variables \*/**

**extern pthread\_mutex\_t temperatureMutex;**

**/\***

**\* ======== clearAlert ========**

**\* Clear the LED**

**\*/**

**//static void clearAlert(float temperature)**

**//{**

**// GPIO\_write(Board\_GPIO\_LED0, Board\_GPIO\_LED\_OFF);**

**//}**

**/\***

**\* ======== sendAlert ========**

**\* Okay, just light a LED in this example, but with the SimpleLink SDK,**

**\* you could send it out over the radio to something cool!**

**\*/**

**//static void sendAlert(float temperature)**

**//{**

**// GPIO\_write(Board\_GPIO\_LED0, Board\_GPIO\_LED\_ON);**

**//}**

**/\***

**\* ======== postSem ========**

**\* Function called when the timer (created in setupTimer) expires.**

**\*/**

**static void postSem(union sigval val)**

**{**

**sem\_t \*sem = (sem\_t\*)(val.sival\_ptr);**

**sem\_post(sem);**

**}**

**/\***

**\* ======== setupTimer ========**

**\* Create a timer that will expire at the period specified by the**

**\* time arguments. When the timer expires, the passed in semaphore**

**\* will be posted by the postSem function.**

**\***

**\* A non-zero return indicates a failure.**

**\*/**

**int setupTimer(sem\_t \*sem, timer\_t \*timerid, time\_t sec, long nsec)**

**{**

**struct sigevent sev;**

**struct itimerspec its;**

**int retc;**

**retc = sem\_init(sem, 0, 0);**

**if (retc != 0) {**

**return(retc);**

**}**

**/\* Create the timer that wakes up the thread that will pend on the sem. \*/**

**sev.sigev\_notify = SIGEV\_SIGNAL;**

**sev.sigev\_value.sival\_ptr = sem;**

**sev.sigev\_notify\_function = &postSem;**

**sev.sigev\_notify\_attributes = NULL;**

**retc = timer\_create(CLOCK\_MONOTONIC, &sev, timerid);**

**if (retc != 0) {**

**return(retc);**

**}**

**/\* Set the timer to go off at the specified period \*/**

**its.it\_interval.tv\_sec = sec;**

**its.it\_interval.tv\_nsec = nsec;**

**its.it\_value.tv\_sec = sec;**

**its.it\_value.tv\_nsec = nsec;**

**retc = timer\_settime(\*timerid, 0, &its, NULL);**

**if (retc != 0) {**

**timer\_delete(\*timerid);**

**return(retc);**

**}**

**return(0);**

**}**

**/\***

**\* ======== temperatureThread ========**

**\* This thread reads the temperature every second via I2C and sends an**

**\* alert if it goes above HIGH\_TEMP.**

**\*/**

**void \*temperatureThread(void \*arg0)**

**{**

**uint8\_t txBuffer[1];**

**uint8\_t rxBuffer[2];**

**I2C\_Handle i2c;**

**I2C\_Params i2cParams;**

**I2C\_Transaction i2cTransaction;**

**sem\_t semTimer;**

**// timer\_t timerid;**

**// int retc;**

**/\* Configure the LED and if applicable, the TMP116\_EN pin \*/**

**GPIO\_setConfig(Board\_GPIO\_LED0, GPIO\_CFG\_OUT\_STD | GPIO\_CFG\_OUT\_LOW);**

**#ifdef Board\_GPIO\_TMP116\_EN**

**GPIO\_setConfig(Board\_GPIO\_TMP116\_EN, GPIO\_CFG\_OUT\_STD | GPIO\_CFG\_OUT\_HIGH);**

**/\* 1.5 ms reset time for the TMP116 \*/**

**sleep(1);**

**#endif**

**/\***

**\* Create/Open the I2C that talks to the TMP sensor**

**\*/**

**I2C\_init();**

**Display\_init();**

**I2C\_Params\_init(&i2cParams);**

**i2cParams.bitRate = I2C\_400kHz;**

**i2c = I2C\_open(Board\_I2C\_TMP, &i2cParams);**

**if (i2c == NULL) {**

**while (1);**

**}**

**/\* Common I2C transaction setup \*/**

**i2cTransaction.writeBuf = txBuffer;**

**i2cTransaction.writeCount = 1;**

**i2cTransaction.readBuf = rxBuffer;**

**i2cTransaction.readCount = 2;**

**/\***

**\* Determine which I2C sensor is present.**

**\* We will prefer sensors in this order: TMP116 (on BoosterPacks),**

**\* TMP116 (on-board CC32XX LaunchPads), and last TMP006**

**\* (on older CC32XX LaunchPads).**

**\*/**

**// Try Si7021**

**txBuffer[0] = Si7021\_TMP\_REG;**

**i2cTransaction.slaveAddress = Si7021\_ADDR;**

**if (!I2C\_transfer(i2c, &i2cTransaction))**

**{**

**// Could not resolve a sensor, error**

**Display\_printf(display, 0, 0, "Error. No TMP sensor found!");**

**while(1);**

**}**

**else**

**{**

**Display\_printf(display, 0, 0, "Detected Si7021 sensor.");**

**}**

**// Take 20 samples and print them out onto the console**

**for (sample = 0; sample < 100; sample++)**

**{**

**if (I2C\_transfer(i2c, &i2cTransaction))**

**{**

**//**

**// Extract degrees C from the received data;**

**// see Si7021 datasheet**

**//**

**temp = (rxBuffer[0] << 8) | (rxBuffer[1]);**

**temperature = (((175.72 \* temp)/ 65536) - 46.85); // celsius**

**temperaturef = (temperature \* (1.8)) + 32; //farenheit**

**Display\_printf(display, 0, 0, "Sample %u: %d (C)", sample, temperaturef);**

**}**

**else**

**{**

**Display\_printf(display, 0, 0, "I2C Bus fault.");**

**}**

**}**

**/\***

**\* The temperature thread blocks on the semTimer semaphore, which the**

**\* timerId timer will post every second. The timer is created in the**

**\* setupTimer function. It's returned so the thread could change the**

**\* period or delete it if desired.**

**\*/**

**// retc = setupTimer(&semTimer, &timerid, 1, 0);**

**// if (retc != 0) {**

**// while (1);**

**// }**

**// while (1)**

**// {**

**// if (I2C\_transfer(i2c, &i2cTransaction)) {**

**// /\***

**// \* Extract degrees C from the received data; see sensor datasheet.**

**// \* Make sure we are updating the global temperature variables**

**// \* in a thread-safe manner.**

**// \*/**

**// pthread\_mutex\_lock(&temperatureMutex);**

**// temperatureC = (rxBuffer[0] << 6) | (rxBuffer[1] >> 2);**

**// temperatureC \*= 0.03125;**

**// temperatureF = temperatureC \* 9 / 5 + 32;**

**// pthread\_mutex\_unlock(&temperatureMutex);**

**//**

**// /\* Send an alert if the temperature is too high!! \*/**

**// if ((int)temperatureC >= HIGH\_TEMP) {**

**// sendAlert(temperatureC);**

**// }**

**// else {**

**// clearAlert(temperatureC);**

**// }**

**// }**

**//---------------------------------------------------------------------------**

**/\***

**// Common I2C transaction setup**

**i2cTransaction.writeBuf = txBuffer;**

**i2cTransaction.writeCount = 1;**

**i2cTransaction.readBuf = rxBuffer;**

**i2cTransaction.readCount = 2;**

**\*/**

**/\***

**// Try Si7021**

**txBuffer[0] = Si7021\_TMP\_REG;**

**i2cTransaction.slaveAddress = Si7021\_ADDR;**

**if (!I2C\_transfer(i2c, &i2cTransaction))**

**{**

**// Could not resolve a sensor, error**

**Display\_printf(display, 0, 0, "Error. No TMP sensor found!");**

**while(1);**

**}**

**else**

**{**

**Display\_printf(display, 0, 0, "Detected Si7021 sensor.");**

**}**

**// Take 20 samples and print them out onto the console**

**for (sample = 0; sample < 20; sample++)**

**{**

**if (I2C\_transfer(i2c, &i2cTransaction))**

**{**

**//**

**// Extract degrees C from the received data;**

**// see Si7021 datasheet**

**//**

**temperature = (rxBuffer[0] << 8) | (rxBuffer[1]);**

**temperaturef = (((175.72 \* temperature)/ 65536) - 46.85);**

**Display\_printf(display, 0, 0, "Sample %u: %d (C)", sample, temperaturef);**

**}**

**else**

**{**

**Display\_printf(display, 0, 0, "I2C Bus fault.");**

**}**

**}**

**\*/**

**//---------------------------------------------------------------------------**

**// /\* Block until the timer posts the semaphore. \*/**

**// retc = sem\_wait(&semTimer);**

**// if (retc == -1) {**

**// while (1);**

**// }**

**// }**

**}**

**Collector Code:**

**Config.h**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**@file config.h**

**@brief TI-15.4 Stack configuration parameters for Collector 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**

**/\*! Coordinator short address \*/**

**#define CONFIG\_COORD\_SHORT\_ADDR 0xAABB**

**/\*! FH disabled as default \*/**

**#define CONFIG\_FH\_ENABLE false**

**/\*! maximum beacons possibly received \*/**

**#define CONFIG\_MAX\_BEACONS\_RECD 200**

**/\*! maximum devices in association table \*/**

**#define CONFIG\_MAX\_DEVICES 50**

**/\*!**

**Setting beacon order to 15 will disable the beacon, 8 is a good value for**

**beacon mode**

**\*/**

**#define CONFIG\_MAC\_BEACON\_ORDER 15**

**/\*!**

**Setting superframe order to 15 will disable the superframe, 8 is a good value**

**for beacon mode**

**\*/**

**#define CONFIG\_MAC\_SUPERFRAME\_ORDER 15**

**/\*! Setting for Phy ID \*/**

**#define CONFIG\_PHY\_ID (APIMAC\_STD\_US\_915\_PHY\_1)**

**#if ((CONFIG\_PHY\_ID >= APIMAC\_MRFSK\_STD\_PHY\_ID\_BEGIN) && (CONFIG\_PHY\_ID <= APIMAC\_MRFSK\_STD\_PHY\_ID\_END))**

**/\*! Setting for channel page \*/**

**#define CONFIG\_CHANNEL\_PAGE (APIMAC\_CHANNEL\_PAGE\_9)**

**#elif ((CONFIG\_PHY\_ID >= APIMAC\_MRFSK\_GENERIC\_PHY\_ID\_BEGIN) && (CONFIG\_PHY\_ID <= APIMAC\_MRFSK\_GENERIC\_PHY\_ID\_END))**

**/\*! Setting for channel page \*/**

**#define CONFIG\_CHANNEL\_PAGE (APIMAC\_CHANNEL\_PAGE\_10)**

**#else**

**#error "PHY ID is wrong."**

**#endif**

**#if (defined(CC1312R1\_LAUNCHXL))**

**#if((CONFIG\_PHY\_ID == APIMAC\_GENERIC\_CHINA\_433\_PHY\_128) || (CONFIG\_PHY\_ID == APIMAC\_GENERIC\_CHINA\_LRM\_433\_PHY\_130))**

**#error "Error: 433 MHz Operation is not supported on 1312 board!"**

**#endif**

**#endif**

**/\*! MAC Parameter \*/**

**/\*! Min BE - Minimum Backoff Exponent \*/**

**#define CONFIG\_MIN\_BE 3**

**/\*! Max BE - Maximum Backoff Exponent \*/**

**#define CONFIG\_MAX\_BE 5**

**/\*! MAC MAX CSMA Backoffs \*/**

**#define CONFIG\_MAC\_MAX\_CSMA\_BACKOFFS 4**

**/\*! macMaxFrameRetries - Maximum Frame Retries \*/**

**#define CONFIG\_MAX\_RETRIES 3**

**/\*! Application traffic profile \*/**

**#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)))**

**/\*!**

**Reporting Interval - in milliseconds to be set on connected devices using**

**configuration request messages**

**\*/**

**#define CONFIG\_REPORTING\_INTERVAL 1000**

**/\*!**

**Polling interval in milliseconds to be set on connected devices using**

**configuration request messages. Must be greater than or equal to default**

**polling interval set on sensor devices**

**\*/**

**#define CONFIG\_POLLING\_INTERVAL 100**

**/\*!**

**Time interval in ms between tracking message intervals**

**\*/**

**#define TRACKING\_DELAY\_TIME 60000**

**#else**

**/\*!**

**Reporting Interval - in milliseconds to be set on connected devices using**

**configuration request messages**

**\*/**

**#define CONFIG\_REPORTING\_INTERVAL 300000**

**/\*!**

**Polling interval in milliseconds to be set on connected devices using**

**configuration request messages. Must be greater than or equal to default**

**polling interval set on sensor devices**

**\*/**

**#define CONFIG\_POLLING\_INTERVAL 60000**

**/\*!**

**Time interval in ms between tracking message intervals**

**\*/**

**#define TRACKING\_DELAY\_TIME 300000**

**#endif**

**/\*! scan duration**

**\* scan type = MAC\_MPM\_SCAN\_NBPAN (see mac\_api.h):**

**\* scan duration = aBaseSlotDuration \* 2 \* CONFIG\_SCAN\_DURATION**

**\***

**\* scan type = MAC\_MPM\_SCAN\_BPAN (see mac\_api.h):**

**\* scan duration = aBaseSuperframeDuration \* 2 \* CONFIG\_SCAN\_DURATION**

**\* other types**

**\* scan duration = aBaseSuperframeDuration \* (1 + 2 \* CONFIG\_SCAN\_DURATION)**

**\*/**

**#define CONFIG\_SCAN\_DURATION 5**

**/\*!**

**Range Extender Mode setting.**

**The following modes are available.**

**APIMAC\_NO\_EXTENDER - does not have PA/LNA**

**APIMAC\_HIGH\_GAIN\_MODE - high gain mode**

**To enable CC1190, use**

**#define CONFIG\_RANGE\_EXT\_MODE APIMAC\_HIGH\_GAIN\_MODE**

**\*/**

**#define CONFIG\_RANGE\_EXT\_MODE APIMAC\_NO\_EXTENDER**

**/\*! Setting Default Key\*/**

**#define KEY\_TABLE\_DEFAULT\_KEY {0x12, 0x34, 0x56, 0x78, 0x9a, 0xbc, 0xde, 0xf0,\**

**0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00}**

**/\*!**

**Channel mask used when CONFIG\_FH\_ENABLE is false.**

**Each bit indicates if the corresponding channel is to be scanned**

**First byte represents channels 0 to 7 and the last byte represents**

**channels 128 to 135.**

**For byte zero in the bit mask, LSB representing Ch0.**

**For byte 1, LSB represents Ch8 and so on.**

**e.g., 0x01 0x10 represents Ch0 and Ch12 are included.**

**The default of 0x0F represents channels 0-3 are selected.**

**APIMAC\_STD\_US\_915\_PHY\_1 (50kbps/2-FSK/915MHz band) has channels 0 - 128.**

**APIMAC\_STD\_ETSI\_863\_PHY\_3 (50kbps/2-FSK/863MHz band) has channels 0 - 33.**

**APIMAC\_GENERIC\_CHINA\_433\_PHY\_128 (50kbps/2-FSK/433MHz band) has channels 0 - 6.**

**\*/**

**#define CONFIG\_CHANNEL\_MASK { 0x0F, 0x00, 0x00, 0x00, 0x00, 0x00, \**

**0x00, 0x00, 0x00, 0x00, 0x00, 0x00, \**

**0x00, 0x00, 0x00, 0x00, 0x00 }**

**/\*!**

**Channel mask used when CONFIG\_FH\_ENABLE is true.**

**Represents the list of channels on which the device can hop.**

**The actual sequence used shall be based on DH1CF function.**

**It is represented as a bit string with LSB representing Ch0.**

**e.g., 0x01 0x10 represents Ch0 and Ch12 are included.**

**\*/**

**#define CONFIG\_FH\_CHANNEL\_MASK { 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, \**

**0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, \**

**0xFF, 0xFF, 0xFF, 0xFF, 0xFF }**

**/\*!**

**List of channels to target the Async frames**

**It is represented as a bit string with LSB representing Ch0**

**e.g., 0x01 0x10 represents Ch0 and Ch12 are included**

**It should cover all channels that could be used by a target device in its**

**hopping sequence. Channels marked beyond number of channels supported by**

**PHY Config will be excluded by stack. To avoid interference on a channel,**

**it should be removed from Async Mask and added to exclude channels**

**(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 }**

**/\* FH related config variables \*/**

**/\*!**

**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 5**

**/\*!**

**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 5**

**/\*!**

**Dwell time: The duration for which the collector will**

**stay on a specific channel before hopping to next channel.**

**\*/**

**#define CONFIG\_DWELL\_TIME 250**

**/\*!**

**FH Application Broadcast Msg generation interval in ms.**

**Value should be set at least greater than 200 ms,**

**\*/**

**#define FH\_BROADCAST\_INTERVAL 10000**

**/\*! FH Broadcast dwell time. If set to 0, it shall disable broadcast hopping and**

**\* broadcast message transmissions in FH Mode \*/**

**#define FH\_BROADCAST\_DWELL\_TIME 100**

**#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)))**

**/\*!**

**The minimum trickle timer window for PAN Advertisement,**

**and PAN Configuration frame transmissions.**

**Recommended to set this to half of PAS/PCS MIN Timer**

**\*/**

**#define CONFIG\_TRICKLE\_MIN\_CLK\_DURATION 3000**

**/\*!**

**The maximum trickle timer window for PAN Advertisement,**

**and PAN Configuration frame transmissions.**

**\*/**

**#define CONFIG\_TRICKLE\_MAX\_CLK\_DURATION 6000**

**#else**

**/\*!**

**The minimum trickle timer window for PAN Advertisement,**

**and PAN Configuration frame transmissions.**

**Recommended to set this to half of PAS/PCS MIN Timer**

**\*/**

**#define CONFIG\_TRICKLE\_MIN\_CLK\_DURATION 30000**

**/\*!**

**The maximum trickle timer window for PAN Advertisement,**

**and PAN Configuration frame transmissions.**

**\*/**

**#define CONFIG\_TRICKLE\_MAX\_CLK\_DURATION 60000**

**#endif**

**/\*!**

**To enable Doubling of PA/PC trickle time,**

**useful when network has non sleepy nodes and**

**there is a requirement to use PA/PC to convey updated**

**PAN information. Note that when using option the CONFIG\_TRICKLE\_MIN\_CLK\_DURATION**

**and CONFIG\_TRICKLE\_MAX\_CLK\_DURATION should be set to a sufficiently large value.**

**Recommended values are 1 min and 16 min respectively.**

**\*/**

**#define CONFIG\_DOUBLE\_TRICKLE\_TIMER false**

**/\*! value for ApiMac\_FHAttribute\_netName \*/**

**#define CONFIG\_FH\_NETNAME {"FHTest"}**

**/\*!**

**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 \*/**

**#define CONFIG\_TRANSMIT\_POWER 12**

**#endif**

**#ifndef DeviceFamily\_CC13X2**

**#if CONFIG\_RANGE\_EXT\_MODE**

**#if (CCFG\_FORCE\_VDDR\_HH == 1)**

**#error "CCFG\_FORCE\_VDDR\_HH should be 0"**

**#endif**

**#else**

**#if ((CONFIG\_PHY\_ID == APIMAC\_GENERIC\_CHINA\_433\_PHY\_128) || (CONFIG\_PHY\_ID == APIMAC\_GENERIC\_CHINA\_LRM\_433\_PHY\_130))**

**#if (CCFG\_FORCE\_VDDR\_HH == 0)**

**#if (CONFIG\_TRANSMIT\_POWER >= 15)**

**#error "CONFIG\_TRANSMIT\_POWER should be less than 15"**

**#endif**

**#else**

**#if (CONFIG\_TRANSMIT\_POWER < 15)**

**/\* In 433 MHz band when CCFG\_FORCE\_VDDR\_HH = 1, only possible value of transmit power is 15 \*/**

**#error "CONFIG\_TRANSMIT\_POWER should be 15"**

**#endif**

**#endif**

**#else**

**#if (CCFG\_FORCE\_VDDR\_HH == 0)**

**#if (CONFIG\_TRANSMIT\_POWER >= 14)**

**#error "CONFIG\_TRANSMIT\_POWER should be less than 14"**

**#endif**

**#else**

**#if (CONFIG\_TRANSMIT\_POWER < 14)**

**/\* In US and ETSI band when CCFG\_FORCE\_VDDR\_HH = 1, only possible value of transmit power is 14 \*/**

**#error "CONFIG\_TRANSMIT\_POWER should be 14"**

**#endif**

**#endif**

**#endif**

**#endif**

**#else**

**#if (CCFG\_FORCE\_VDDR\_HH == 1)**

**#if (CONFIG\_TRANSMIT\_POWER != 14)**

**/\* In US and ETSI band when CCFG\_FORCE\_VDDR\_HH = 1, only possible value of transmit power is 14 \*/**

**#error "CONFIG\_TRANSMIT\_POWER should be 14"**

**#endif**

**#endif**

**#endif**

**/\*!**

**\* Enable this mode for certfication.**

**\* For FH certification, CONFIG\_FH\_ENABLE should**

**\* also be enabled.**

**\*/**

**#define CERTIFICATION\_TEST\_MODE false**

**#ifdef POWER\_MEAS**

**/\*! Size of RAMP Data to be sent when POWER Test is enabled \*/**

**#define COLLECTOR\_TEST\_RAMP\_DATA\_SIZE 20**

**/\*!**

**Power profile to be used when Power MEAS is enabled.**

**Profile 1 - POLL\_ACK - Polling Only**

**Profile 2 - DATA\_ACK - 20 byte application data + ACK from sensor to collector**

**Profile 3 - POLL\_DATA - Poll + received Data from collector**

**Profile 4 - SLEEP - No Poll or Data. In Beacon mode, beacon RX would occur**

**\*/**

**#define POWER\_TEST\_PROFILE DATA\_ACK**

**#endif**

**/\* Check if all the necessary parameters have been set for FH mode \*/**

**#if CONFIG\_FH\_ENABLE**

**#if !defined(FEATURE\_ALL\_MODES) && !defined(FEATURE\_FREQ\_HOP\_MODE)**

**#error "Do you want to build image with frequency hopping mode? \**

**Define either FEATURE\_FREQ\_HOP\_MODE or FEATURE\_ALL\_MODES in features.h"**

**#endif**

**#endif**

**/\* Check if stack level security is enabled if application security is enabled \*/**

**#if CONFIG\_SECURE**

**#if !defined(FEATURE\_MAC\_SECURITY)**

**#error "Define FEATURE\_MAC\_SECURITY or FEATURE\_ALL\_MODES in features.h to \**

**be able to use security at application level"**

**#endif**

**#endif**

**/\* Set beacon order and superframe order to 15 for FH mode to avoid user error \*/**

**#if CONFIG\_FH\_ENABLE**

**#if (CONFIG\_MAC\_BEACON\_ORDER != 15) && (CONFIG\_MAC\_SUPERFRAME\_ORDER != 15)**

**#error "Do you want to build image with frequency hopping mode? \**

**If yes, CONFIG\_MAC\_BEACON\_ORDER and CONFIG\_MAC\_SUPERFRAME\_ORDER \**

**should both be set to 15"**

**#endif**

**#endif**

**#ifdef \_\_cplusplus**

**}**

**#endif**

**#endif /\* CONFIG\_H \*/**

**Collector.c**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**@file collector.c**

**@brief TIMAC 2.0 Collector Example Application**

**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.**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Includes**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

**#include <string.h>**

**#include <stdint.h>**

**#include "mac\_util.h"**

**#include "api\_mac.h"**

**#include "cllc.h"**

**#include "csf.h"**

**#include "smsgs.h"**

**#include "collector.h"**

**#include "board\_lcd.h"**

**#ifdef FEATURE\_SECURE\_COMMISSIONING**

**#include "sm\_ti154.h"**

**#endif**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Constants and definitions**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

**#if !defined(STATIC)**

**/\* make local \*/**

**#define STATIC static**

**#endif**

**#if !defined(CONFIG\_AUTO\_START)**

**#if defined(AUTO\_START)**

**#define CONFIG\_AUTO\_START 1**

**#else**

**#define CONFIG\_AUTO\_START 0**

**#endif**

**#endif**

**/\* Beacon order for non beacon network \*/**

**#define NON\_BEACON\_ORDER 15**

**/\* Default MSDU Handle rollover \*/**

**#define MSDU\_HANDLE\_MAX 0x1F**

**/\* App marker in MSDU handle \*/**

**#define APP\_MARKER\_MSDU\_HANDLE 0x80**

**/\* App Config request marker for the MSDU handle \*/**

**#define APP\_CONFIG\_MSDU\_HANDLE 0x40**

**/\* Ramp data request marker for the MSDU handle \*/**

**#define RAMP\_DATA\_MSDU\_HANDLE 0x20**

**/\* App Broadcast Cmd Msg marker for the MSDU Handle \*/**

**#define APP\_BROADCAST\_MSDU\_HANDLE 0x20**

**/\* Default configuration frame control \*/**

**#define CONFIG\_FRAME\_CONTROL (Smsgs\_dataFields\_tempSensor | \**

**Smsgs\_dataFields\_lightSensor | \**

**Smsgs\_dataFields\_humiditySensor | \**

**Smsgs\_dataFields\_msgStats | \**

**Smsgs\_dataFields\_configSettings)**

**/\* Delay for config request retry in busy network \*/**

**#define CONFIG\_DELAY 2000**

**#define CONFIG\_RESPONSE\_DELAY 3\*CONFIG\_DELAY**

**/\* Tracking timeouts \*/**

**#define TRACKING\_CNF\_DELAY\_TIME 2000 /\* in milliseconds \*/**

**#if (CONFIG\_PHY\_ID == APIMAC\_PHY\_ID\_NONE) && (CONFIG\_MAC\_BEACON\_ORDER != NON\_BEACON\_ORDER)**

**#define TRACKING\_TIMEOUT\_TIME ((1<<CONFIG\_MAC\_BEACON\_ORDER) \* 960 \* 16 \* 3 / 1000) /\*in milliseconds\*/**

**#else**

**#define TRACKING\_TIMEOUT\_TIME (CONFIG\_POLLING\_INTERVAL \* 3) /\*in milliseconds\*/**

**#endif**

**/\* Initial delay before broadcast transmissions are started in FH mode \*/**

**#define BROADCAST\_CMD\_START\_TIME 60000**

**/\* Number of superframe periods to hold a indirect packet at collector for**

**Sensor to poll and get the frame\*/**

**#define BCN\_MODE\_INDIRECT\_PERSISTENT\_TIME 3**

**#if ((CONFIG\_PHY\_ID >= APIMAC\_MRFSK\_STD\_PHY\_ID\_BEGIN) && (CONFIG\_PHY\_ID <= APIMAC\_MRFSK\_GENERIC\_PHY\_ID\_BEGIN))**

**/\* MAC Indirect Persistent Timeout \*/**

**#define INDIRECT\_PERSISTENT\_TIME ((5 \* 1000 \* CONFIG\_POLLING\_INTERVAL / 2)/ \**

**(BASE\_SUPER\_FRAME\_DURATION \* \**

**SYMBOL\_DURATION\_50\_kbps))**

**#elif ((CONFIG\_PHY\_ID >= APIMAC\_GENERIC\_US\_915\_PHY\_132) && (CONFIG\_PHY\_ID <= APIMAC\_GENERIC\_ETSI\_863\_PHY\_133))**

**/\* MAC Indirect Persistent Timeout \*/**

**#define INDIRECT\_PERSISTENT\_TIME ((5 \* 1000 \* CONFIG\_POLLING\_INTERVAL / 2)/ \**

**(BASE\_SUPER\_FRAME\_DURATION \* \**

**SYMBOL\_DURATION\_200\_kbps))**

**#elif (CONFIG\_PHY\_ID == APIMAC\_PHY\_ID\_NONE)**

**/\* MAC Indirect Persistent Timeout \*/**

**#define INDIRECT\_PERSISTENT\_TIME ((5 \* 1000 \* CONFIG\_POLLING\_INTERVAL / 2)/ \**

**(BASE\_SUPER\_FRAME\_DURATION \* \**

**SYMBOL\_DURATION\_250\_kbps))**

**#else**

**/\* MAC Indirect Persistent Timeout \*/**

**#define INDIRECT\_PERSISTENT\_TIME ((5 \* 1000 \* CONFIG\_POLLING\_INTERVAL / 2)/ \**

**(BASE\_SUPER\_FRAME\_DURATION \* \**

**SYMBOL\_DURATION\_LRM))**

**#endif**

**/\* Assoc Table (CLLC) status settings \*/**

**#define ASSOC\_CONFIG\_SENT 0x0100 /\* Config Req sent \*/**

**#define ASSOC\_CONFIG\_RSP 0x0200 /\* Config Rsp received \*/**

**#define ASSOC\_CONFIG\_MASK 0x0300 /\* Config mask \*/**

**#define ASSOC\_TRACKING\_SENT 0x1000 /\* Tracking Req sent \*/**

**#define ASSOC\_TRACKING\_RSP 0x2000 /\* Tracking Rsp received \*/**

**#define ASSOC\_TRACKING\_RETRY 0x4000 /\* Tracking Req retried \*/**

**#define ASSOC\_TRACKING\_ERROR 0x8000 /\* Tracking Req error \*/**

**#define ASSOC\_TRACKING\_MASK 0xF000 /\* Tracking mask \*/**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Global variables**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

**/\* Task pending events \*/**

**uint16\_t Collector\_events = 0;**

**/\*! Collector statistics \*/**

**Collector\_statistics\_t Collector\_statistics;**

**/\* Permit join setting \*/**

**extern bool permitJoining;**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Local variables**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

**static void \*sem;**

**/\*! true if the device was restarted \*/**

**static bool restarted = false;**

**/\*! CLLC State \*/**

**STATIC Cllc\_states\_t cllcState = Cllc\_states\_initWaiting;**

**/\*! Device's PAN ID \*/**

**STATIC uint16\_t devicePanId = 0xFFFF;**

**/\*! Device's Outgoing MSDU Handle values \*/**

**STATIC uint8\_t deviceTxMsduHandle = 0;**

**STATIC bool fhEnabled = false;**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Local function prototypes**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

**static void initializeClocks(void);**

**static void cllcStartedCB(Llc\_netInfo\_t \*pStartedInfo);**

**static ApiMac\_assocStatus\_t cllcDeviceJoiningCB(**

**ApiMac\_deviceDescriptor\_t \*pDevInfo,**

**ApiMac\_capabilityInfo\_t \*pCapInfo);**

**static void cllcStateChangedCB(Cllc\_states\_t state);**

**static void dataCnfCB(ApiMac\_mcpsDataCnf\_t \*pDataCnf);**

**static void dataIndCB(ApiMac\_mcpsDataInd\_t \*pDataInd);**

**static void processStartEvent(void);**

**static void processConfigResponse(ApiMac\_mcpsDataInd\_t \*pDataInd);**

**static void processTrackingResponse(ApiMac\_mcpsDataInd\_t \*pDataInd);**

**static void processToggleLedResponse(ApiMac\_mcpsDataInd\_t \*pDataInd);**

**static void processSensorData(ApiMac\_mcpsDataInd\_t \*pDataInd);**

**static Cllc\_associated\_devices\_t \*findDevice(ApiMac\_sAddr\_t \*pAddr);**

**static Cllc\_associated\_devices\_t \*findDeviceStatusBit(uint16\_t mask, uint16\_t statusBit);**

**static uint8\_t getMsduHandle(Smsgs\_cmdIds\_t msgType);**

**static bool sendMsg(Smsgs\_cmdIds\_t type, uint16\_t dstShortAddr, bool rxOnIdle,**

**uint16\_t len,**

**uint8\_t \*pData);**

**static void generateConfigRequests(void);**

**static void generateTrackingRequests(void);**

**static void generateBroadcastCmd(void);**

**static void sendTrackingRequest(Cllc\_associated\_devices\_t \*pDev);**

**static void commStatusIndCB(ApiMac\_mlmeCommStatusInd\_t \*pCommStatusInd);**

**static void pollIndCB(ApiMac\_mlmePollInd\_t \*pPollInd);**

**static void processDataRetry(ApiMac\_sAddr\_t \*pAddr);**

**static void processConfigRetry(void);**

**static void processIdendifyLedRequest(ApiMac\_mcpsDataInd\_t \*pDataInd);**

**static void orphanIndCb(ApiMac\_mlmeOrphanInd\_t \*pData);**

**#ifdef FEATURE\_SECURE\_COMMISSIONING**

**/\* Security Manager callback functions \*/**

**static void smFailCMProcessCb(ApiMac\_deviceDescriptor\_t \*devInfo, bool rxOnIdle, bool keyRefreshment);**

**static void smSuccessCMProcessCb(ApiMac\_deviceDescriptor\_t \*devInfo, bool keyRefreshment);**

**#endif /\* FEATURE\_SECURE\_COMMISSIONING \*/**

**#ifdef POWER\_MEAS**

**void generateIndirectRampMsg(void);**

**#endif**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Callback tables**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

**/\*! API MAC Callback table \*/**

**ApiMac\_callbacks\_t Collector\_macCallbacks =**

**{**

**/\*! Associate Indicated callback \*/**

**NULL,**

**/\*! Associate Confirmation callback \*/**

**NULL,**

**/\*! Disassociate Indication callback \*/**

**NULL,**

**/\*! Disassociate Confirmation callback \*/**

**NULL,**

**/\*! Beacon Notify Indication callback \*/**

**NULL,**

**/\*! Orphan Indication callback \*/**

**orphanIndCb,**

**/\*! Scan Confirmation callback \*/**

**NULL,**

**/\*! Start Confirmation callback \*/**

**NULL,**

**/\*! Sync Loss Indication callback \*/**

**NULL,**

**/\*! Poll Confirm callback \*/**

**NULL,**

**/\*! Comm Status Indication callback \*/**

**commStatusIndCB,**

**/\*! Poll Indication Callback \*/**

**pollIndCB,**

**/\*! Data Confirmation callback \*/**

**dataCnfCB,**

**/\*! Data Indication callback \*/**

**dataIndCB,**

**/\*! Purge Confirm callback \*/**

**NULL,**

**/\*! WiSUN Async Indication callback \*/**

**NULL,**

**/\*! WiSUN Async Confirmation callback \*/**

**NULL,**

**/\*! Unprocessed message callback \*/**

**NULL**

**};**

**STATIC Cllc\_callbacks\_t cllcCallbacks =**

**{**

**/\*! Coordinator Started Indication callback \*/**

**cllcStartedCB,**

**/\*! Device joining callback \*/**

**cllcDeviceJoiningCB,**

**/\*! The state has changed callback \*/**

**cllcStateChangedCB**

**};**

**#ifdef FEATURE\_SECURE\_COMMISSIONING**

**STATIC SM\_callbacks\_t SMCallbacks =**

**{**

**/\*! Security authentication failed callback \*/**

**smFailCMProcessCb,**

**/\* Security authentication successful callback \*/**

**smSuccessCMProcessCb**

**};**

**#endif /\* FEATURE\_SECURE\_COMMISSIONING \*/**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Public Functions**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

**/\*!**

**Initialize this application.**

**Public function defined in collector.h**

**\*/**

**#ifdef OSAL\_PORT2TIRTOS**

**void Collector\_init(uint8\_t macTaskId)**

**#else**

**void Collector\_init(void)**

**#endif**

**{**

**/\* Initialize the collector's statistics \*/**

**memset(&Collector\_statistics, 0, sizeof(Collector\_statistics\_t));**

**/\* Initialize the MAC \*/**

**#ifdef OSAL\_PORT2TIRTOS**

**sem = ApiMac\_init(macTaskId,CONFIG\_FH\_ENABLE);**

**#else**

**sem = ApiMac\_init(CONFIG\_FH\_ENABLE);**

**#endif**

**ApiMac\_mlmeSetReqUint8(ApiMac\_attribute\_phyCurrentDescriptorId,**

**(uint8\_t)CONFIG\_PHY\_ID);**

**ApiMac\_mlmeSetReqUint8(ApiMac\_attribute\_channelPage,**

**(uint8\_t)CONFIG\_CHANNEL\_PAGE);**

**/\* Initialize the Coordinator Logical Link Controller \*/**

**Cllc\_init(&Collector\_macCallbacks, &cllcCallbacks);**

**/\* Register the MAC Callbacks \*/**

**ApiMac\_registerCallbacks(&Collector\_macCallbacks);**

**/\* Initialize the platform specific functions \*/**

**Csf\_init(sem);**

**#ifdef FEATURE\_SECURE\_COMMISSIONING**

**/\* Intialize the security manager and register callbacks \*/**

**SM\_init();**

**SM\_registerCallback(&SMCallbacks);**

**#endif /\* FEATURE\_SECURE\_COMMISSIONING \*/**

**/\* Set the indirect persistent timeout \*/**

**#if (CONFIG\_MAC\_BEACON\_ORDER != NON\_BEACON\_ORDER)**

**ApiMac\_mlmeSetReqUint16(ApiMac\_attribute\_transactionPersistenceTime,**

**BCN\_MODE\_INDIRECT\_PERSISTENT\_TIME);**

**#else**

**ApiMac\_mlmeSetReqUint16(ApiMac\_attribute\_transactionPersistenceTime,**

**INDIRECT\_PERSISTENT\_TIME);**

**#endif**

**ApiMac\_mlmeSetReqUint8(ApiMac\_attribute\_phyTransmitPowerSigned,**

**(uint8\_t)CONFIG\_TRANSMIT\_POWER);**

**/\* Set Min BE \*/**

**ApiMac\_mlmeSetReqUint8(ApiMac\_attribute\_backoffExponent,**

**(uint8\_t)CONFIG\_MIN\_BE);**

**/\* Set Max BE \*/**

**ApiMac\_mlmeSetReqUint8(ApiMac\_attribute\_maxBackoffExponent,**

**(uint8\_t)CONFIG\_MAX\_BE);**

**/\* Set MAC MAX CSMA Backoffs \*/**

**ApiMac\_mlmeSetReqUint8(ApiMac\_attribute\_maxCsmaBackoffs,**

**(uint8\_t)CONFIG\_MAC\_MAX\_CSMA\_BACKOFFS);**

**/\* Set MAC MAX Frame Retries \*/**

**ApiMac\_mlmeSetReqUint8(ApiMac\_attribute\_maxFrameRetries,**

**(uint8\_t)CONFIG\_MAX\_RETRIES);**

**#ifdef FCS\_TYPE16**

**/\* Set the fcs type \*/**

**ApiMac\_mlmeSetReqBool(ApiMac\_attribute\_fcsType,**

**(bool)1);**

**#endif**

**/\* Initialize the app clocks \*/**

**initializeClocks();**

**if(CONFIG\_FH\_ENABLE && (FH\_BROADCAST\_DWELL\_TIME > 0))**

**{**

**/\* Start broadcast frame transmissions in FH mode if broadcast dwell time**

**\* is greater than zero \*/**

**Csf\_setBroadcastClock(BROADCAST\_CMD\_START\_TIME);**

**}**

**if(CONFIG\_AUTO\_START)**

**{**

**/\* Start the device \*/**

**Util\_setEvent(&Collector\_events, COLLECTOR\_START\_EVT);**

**}**

**}**

**/\*!**

**Application task processing.**

**Public function defined in collector.h**

**\*/**

**void Collector\_process(void)**

**{**

**/\* Start the collector device in the network \*/**

**if(Collector\_events & COLLECTOR\_START\_EVT)**

**{**

**if(cllcState == Cllc\_states\_initWaiting)**

**{**

**processStartEvent();**

**}**

**/\* Clear the event \*/**

**Util\_clearEvent(&Collector\_events, COLLECTOR\_START\_EVT);**

**}**

**/\* Is it time to send the next tracking message? \*/**

**if(Collector\_events & COLLECTOR\_TRACKING\_TIMEOUT\_EVT)**

**{**

**/\* Process Tracking Event \*/**

**generateTrackingRequests();**

**/\* Clear the event \*/**

**Util\_clearEvent(&Collector\_events, COLLECTOR\_TRACKING\_TIMEOUT\_EVT);**

**}**

**/\***

**The generate a config request for all associated devices that need one**

**\*/**

**if(Collector\_events & COLLECTOR\_CONFIG\_EVT)**

**{**

**#ifdef FEATURE\_SECURE\_COMMISSIONING**

**if(SM\_Last\_State != SM\_CM\_InProgress) {**

**generateConfigRequests();**

**}**

**#else**

**generateConfigRequests();**

**#endif**

**/\* Clear the event \*/**

**Util\_clearEvent(&Collector\_events, COLLECTOR\_CONFIG\_EVT);**

**}**

**/\***

**Collector generate a broadcast command message for FH mode**

**\*/**

**if(Collector\_events & COLLECTOR\_BROADCAST\_TIMEOUT\_EVT)**

**{**

**/\* Clear the event \*/**

**Util\_clearEvent(&Collector\_events, COLLECTOR\_BROADCAST\_TIMEOUT\_EVT);**

**if(FH\_BROADCAST\_INTERVAL > 0 && (!CERTIFICATION\_TEST\_MODE))**

**{**

**generateBroadcastCmd();**

**/\* set clock for next broadcast command \*/**

**Csf\_setBroadcastClock(FH\_BROADCAST\_INTERVAL);**

**}**

**}**

**/\* Process LLC Events \*/**

**Cllc\_process();**

**/\* Allow the Specific functions to process \*/**

**Csf\_processEvents();**

**#ifdef FEATURE\_SECURE\_COMMISSIONING**

**/\* Allow the security manager specific functions to process \*/**

**SM\_process();**

**#endif /\* FEATURE\_SECURE\_COMMISSIONING \*/**

**/\***

**Don't process ApiMac messages until all of the collector events**

**are processed.**

**\*/**

**if(Collector\_events == 0)**

**{**

**/\* Wait for response message or events \*/**

**ApiMac\_processIncoming();**

**}**

**}**

**/\*!**

**Build and send the configuration message to a device.**

**Public function defined in collector.h**

**\*/**

**Collector\_status\_t Collector\_sendConfigRequest(ApiMac\_sAddr\_t \*pDstAddr,**

**uint16\_t frameControl,**

**uint32\_t reportingInterval,**

**uint32\_t pollingInterval)**

**{**

**Collector\_status\_t status = Collector\_status\_invalid\_state;**

**/\* Are we in the right state? \*/**

**if(cllcState >= Cllc\_states\_started)**

**{**

**Llc\_deviceListItem\_t item;**

**/\* Is the device a known device? \*/**

**if(Csf\_getDevice(pDstAddr, &item))**

**{**

**uint8\_t buffer[SMSGS\_CONFIG\_REQUEST\_MSG\_LENGTH];**

**uint8\_t \*pBuf = buffer;**

**/\* Build the message \*/**

**\*pBuf++ = (uint8\_t)Smsgs\_cmdIds\_configReq;**

**\*pBuf++ = Util\_loUint16(frameControl);**

**\*pBuf++ = Util\_hiUint16(frameControl);**

**\*pBuf++ = Util\_breakUint32(reportingInterval, 0);**

**\*pBuf++ = Util\_breakUint32(reportingInterval, 1);**

**\*pBuf++ = Util\_breakUint32(reportingInterval, 2);**

**\*pBuf++ = Util\_breakUint32(reportingInterval, 3);**

**\*pBuf++ = Util\_breakUint32(pollingInterval, 0);**

**\*pBuf++ = Util\_breakUint32(pollingInterval, 1);**

**\*pBuf++ = Util\_breakUint32(pollingInterval, 2);**

**\*pBuf = Util\_breakUint32(pollingInterval, 3);**

**if((sendMsg(Smsgs\_cmdIds\_configReq, item.devInfo.shortAddress,**

**item.capInfo.rxOnWhenIdle,**

**(SMSGS\_CONFIG\_REQUEST\_MSG\_LENGTH),**

**buffer)) == true)**

**{**

**status = Collector\_status\_success;**

**Collector\_statistics.configRequestAttempts++;**

**/\* set timer for retry in case response is not received \*/**

**Csf\_setConfigClock(CONFIG\_DELAY);**

**}**

**else**

**{**

**processConfigRetry();**

**}**

**}**

**}**

**return (status);**

**}**

**/\*!**

**Update the collector statistics**

**Public function defined in collector.h**

**\*/**

**void Collector\_updateStats( void )**

**{**

**/\* update the stats from the MAC \*/**

**ApiMac\_mlmeGetReqUint32(ApiMac\_attribute\_diagRxSecureFail,**

**&Collector\_statistics.rxDecryptFailures);**

**ApiMac\_mlmeGetReqUint32(ApiMac\_attribute\_diagTxSecureFail,**

**&Collector\_statistics.txEncryptFailures);**

**}**

**/\*!**

**Build and send the toggle led message to a device.**

**Public function defined in collector.h**

**\*/**

**Collector\_status\_t Collector\_sendToggleLedRequest(ApiMac\_sAddr\_t \*pDstAddr)**

**{**

**Collector\_status\_t status = Collector\_status\_invalid\_state;**

**/\* Are we in the right state? \*/**

**if(cllcState >= Cllc\_states\_started)**

**{**

**Llc\_deviceListItem\_t item;**

**/\* Is the device a known device? \*/**

**if(Csf\_getDevice(pDstAddr, &item))**

**{**

**uint8\_t buffer[SMSGS\_TOGGLE\_LED\_REQUEST\_MSG\_LEN];**

**/\* Build the message \*/**

**buffer[0] = (uint8\_t)Smsgs\_cmdIds\_toggleLedReq;**

**sendMsg(Smsgs\_cmdIds\_toggleLedReq, item.devInfo.shortAddress,**

**item.capInfo.rxOnWhenIdle,**

**SMSGS\_TOGGLE\_LED\_REQUEST\_MSG\_LEN,**

**buffer);**

**status = Collector\_status\_success;**

**}**

**else**

**{**

**status = Collector\_status\_deviceNotFound;**

**}**

**}**

**return(status);**

**}**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Local Functions**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

**/\*!**

**\* @brief Initialize the clocks.**

**\*/**

**static void initializeClocks(void)**

**{**

**/\* Initialize the tracking clock \*/**

**Csf\_initializeTrackingClock();**

**Csf\_initializeConfigClock();**

**Csf\_initializeBroadcastClock();**

**Csf\_initializeIdentifyClock();**

**}**

**/\*!**

**\* @brief CLLC Started callback.**

**\***

**\* @param pStartedInfo - pointer to network information**

**\*/**

**static void cllcStartedCB(Llc\_netInfo\_t \*pStartedInfo)**

**{**

**devicePanId = pStartedInfo->devInfo.panID;**

**if(pStartedInfo->fh == true)**

**{**

**fhEnabled = true;**

**}**

**/\* updated the user \*/**

**Csf\_networkUpdate(restarted, pStartedInfo);**

**/\* Start the tracking clock \*/**

**Csf\_setTrackingClock(TRACKING\_DELAY\_TIME);**

**#ifdef FEATURE\_SECURE\_COMMISSIONING**

**/\* Coordinator has started \*/**

**readySMToRun = true;**

**#endif**

**}**

**/\*!**

**\* @brief Device Joining callback from the CLLC module (ref.**

**\* Cllc\_deviceJoiningFp\_t in cllc.h). This function basically**

**\* gives permission that the device can join with the return**

**\* value.**

**\***

**\* @param pDevInfo - device information**

**\* @param capInfo - device's capability information**

**\***

**\* @return ApiMac\_assocStatus\_t**

**\*/**

**static ApiMac\_assocStatus\_t cllcDeviceJoiningCB(**

**ApiMac\_deviceDescriptor\_t \*pDevInfo,**

**ApiMac\_capabilityInfo\_t \*pCapInfo)**

**{**

**ApiMac\_assocStatus\_t status;**

**/\* Make sure the device is in our PAN \*/**

**if(pDevInfo->panID == devicePanId)**

**{**

**/\* Update the user that a device is joining \*/**

**status = Csf\_deviceUpdate(pDevInfo, pCapInfo);**

**if(status==ApiMac\_assocStatus\_success)**

**{**

**#ifdef FEATURE\_MAC\_SECURITY**

**/\* Add device to security device table \*/**

**Cllc\_addSecDevice(pDevInfo->panID,**

**pDevInfo->shortAddress,**

**&pDevInfo->extAddress, 0);**

**#endif /\* FEATURE\_MAC\_SECURITY \*/**

**#ifdef FEATURE\_SECURE\_COMMISSIONING**

**/\* Strictly do not set the event for sending collector**

**\* config event as this has to be sent only after the**

**\* secure commissioning is finished successfully**

**\*/**

**#else**

**/\* Set event for sending collector config packet \*/**

**Util\_setEvent(&Collector\_events, COLLECTOR\_CONFIG\_EVT);**

**#endif**

**}**

**}**

**else**

**{**

**status = ApiMac\_assocStatus\_panAccessDenied;**

**}**

**return (status);**

**}**

**/\*!**

**\* @brief CLLC State Changed callback.**

**\***

**\* @param state - CLLC new state**

**\*/**

**static void cllcStateChangedCB(Cllc\_states\_t state)**

**{**

**/\* Save the state \*/**

**cllcState = state;**

**/\* Notify the user interface \*/**

**Csf\_stateChangeUpdate(cllcState);**

**}**

**/\*!**

**\* @brief MAC Data Confirm callback.**

**\***

**\* @param pDataCnf - pointer to the data confirm information**

**\*/**

**static void dataCnfCB(ApiMac\_mcpsDataCnf\_t \*pDataCnf)**

**{**

**/\* Record statistics \*/**

**if(pDataCnf->status == ApiMac\_status\_channelAccessFailure)**

**{**

**Collector\_statistics.channelAccessFailures++;**

**}**

**else if(pDataCnf->status == ApiMac\_status\_noAck)**

**{**

**Collector\_statistics.ackFailures++;**

**}**

**else if(pDataCnf->status == ApiMac\_status\_transactionExpired)**

**{**

**Collector\_statistics.txTransactionExpired++;**

**}**

**else if(pDataCnf->status == ApiMac\_status\_transactionOverflow)**

**{**

**Collector\_statistics.txTransactionOverflow++;**

**}**

**else if(pDataCnf->status == ApiMac\_status\_success)**

**{**

**Csf\_updateFrameCounter(NULL, pDataCnf->frameCntr);**

**}**

**else if(pDataCnf->status != ApiMac\_status\_success)**

**{**

**Collector\_statistics.otherTxFailures++;**

**}**

**#ifdef POWER\_MEAS**

**/\* Back to back data messages to ensure a response for every poll message \*/**

**if((pDataCnf->msduHandle & RAMP\_DATA\_MSDU\_HANDLE) &&**

**(POWER\_TEST\_PROFILE == POLL\_DATA))**

**{**

**generateIndirectRampMsg();**

**}**

**#endif**

**/\* Make sure the message came from the app \*/**

**if(pDataCnf->msduHandle & APP\_MARKER\_MSDU\_HANDLE)**

**{**

**/\* What message type was the original request? \*/**

**if(pDataCnf->msduHandle & APP\_CONFIG\_MSDU\_HANDLE)**

**{**

**/\* Config Request \*/**

**Cllc\_associated\_devices\_t \*pDev;**

**pDev = findDeviceStatusBit(ASSOC\_CONFIG\_MASK, ASSOC\_CONFIG\_SENT);**

**if(pDev != NULL)**

**{**

**if(pDataCnf->status != ApiMac\_status\_success)**

**{**

**/\* Try to send again \*/**

**pDev->status &= ~ASSOC\_CONFIG\_SENT;**

**Csf\_setConfigClock(CONFIG\_DELAY);**

**}**

**else**

**{**

**pDev->status |= ASSOC\_CONFIG\_SENT;**

**pDev->status |= ASSOC\_CONFIG\_RSP;**

**pDev->status |= CLLC\_ASSOC\_STATUS\_ALIVE;**

**Csf\_setConfigClock(CONFIG\_RESPONSE\_DELAY);**

**}**

**}**

**/\* Update stats \*/**

**if(pDataCnf->status == ApiMac\_status\_success)**

**{**

**Collector\_statistics.configReqRequestSent++;**

**}**

**}**

**else if(pDataCnf->msduHandle & APP\_BROADCAST\_MSDU\_HANDLE)**

**{**

**if(pDataCnf->status == ApiMac\_status\_success)**

**{**

**Collector\_statistics.broadcastMsgSentCnt++;**

**}**

**}**

**else**

**{**

**/\* Tracking Request \*/**

**Cllc\_associated\_devices\_t \*pDev;**

**pDev = findDeviceStatusBit(ASSOC\_TRACKING\_SENT,**

**ASSOC\_TRACKING\_SENT);**

**if(pDev != NULL)**

**{**

**if(pDataCnf->status == ApiMac\_status\_success)**

**{**

**/\* Make sure the retry is clear \*/**

**pDev->status &= ~ASSOC\_TRACKING\_RETRY;**

**}**

**else**

**{**

**if(pDev->status & ASSOC\_TRACKING\_RETRY)**

**{**

**/\* We already tried to resend \*/**

**pDev->status &= ~ASSOC\_TRACKING\_RETRY;**

**pDev->status |= ASSOC\_TRACKING\_ERROR;**

**}**

**else**

**{**

**/\* Go ahead and retry \*/**

**pDev->status |= ASSOC\_TRACKING\_RETRY;**

**}**

**pDev->status &= ~ASSOC\_TRACKING\_SENT;**

**/\* Try to send again or another \*/**

**Csf\_setTrackingClock(TRACKING\_CNF\_DELAY\_TIME);**

**}**

**}**

**/\* Update stats \*/**

**if(pDataCnf->status == ApiMac\_status\_success)**

**{**

**Collector\_statistics.trackingReqRequestSent++;**

**}**

**}**

**}**

**}**

**/\*!**

**\* @brief MAC Data Indication callback.**

**\***

**\* @param pDataInd - pointer to the data indication information**

**\*/**

**static void dataIndCB(ApiMac\_mcpsDataInd\_t \*pDataInd)**

**{**

**if((pDataInd != NULL) && (pDataInd->msdu.p != NULL)**

**&& (pDataInd->msdu.len > 0))**

**{**

**Smsgs\_cmdIds\_t cmdId = (Smsgs\_cmdIds\_t)\*(pDataInd->msdu.p);**

**#ifdef FEATURE\_MAC\_SECURITY**

**{**

**if(Cllc\_securityCheck(&(pDataInd->sec)) == false)**

**{**

**/\* Reject the message \*/**

**return;**

**}**

**}**

**#endif /\* FEATURE\_MAC\_SECURITY \*/**

**if(pDataInd->srcAddr.addrMode == ApiMac\_addrType\_extended)**

**{**

**uint16\_t shortAddr = Csf\_getDeviceShort(**

**&pDataInd->srcAddr.addr.extAddr);**

**if(shortAddr != CSF\_INVALID\_SHORT\_ADDR)**

**{**

**/\* Switch to the short address for internal tracking \*/**

**pDataInd->srcAddr.addrMode = ApiMac\_addrType\_short;**

**pDataInd->srcAddr.addr.shortAddr = shortAddr;**

**}**

**else**

**{**

**/\* Can't accept the message - ignore it \*/**

**return;**

**}**

**}**

**switch(cmdId)**

**{**

**case Smsgs\_cmdIds\_configRsp:**

**processConfigResponse(pDataInd);**

**break;**

**case Smsgs\_cmdIds\_trackingRsp:**

**processTrackingResponse(pDataInd);**

**break;**

**case Smsgs\_cmdIds\_IdentifyLedReq:**

**processIdendifyLedRequest(pDataInd);**

**break;**

**case Smsgs\_cmdIds\_toggleLedRsp:**

**processToggleLedResponse(pDataInd);**

**break;**

**case Smsgs\_cmdIds\_sensorData:**

**processSensorData(pDataInd);**

**break;**

**case Smsgs\_cmdIds\_rampdata:**

**Collector\_statistics.sensorMessagesReceived++;**

**break;**

**#ifdef FEATURE\_SECURE\_COMMISSIONING**

**case Smgs\_cmdIds\_CommissionMsg:**

**/\* Process Security manager commissioning data \*/**

**SM\_processCommData(pDataInd);**

**break;**

**#endif /\* FEATURE\_SECURE\_COMMISSIONING \*/**

**default:**

**/\* Should not receive other messages \*/**

**break;**

**}**

**}**

**}**

**/\*!**

**\* @brief Process the start event**

**\*/**

**static void processStartEvent(void)**

**{**

**Llc\_netInfo\_t netInfo;**

**uint32\_t frameCounter = 0;**

**Csf\_getFrameCounter(NULL, &frameCounter);**

**/\* See if there is existing network information \*/**

**if(Csf\_getNetworkInformation(&netInfo))**

**{**

**uint16\_t numDevices = 0;**

**#ifdef FEATURE\_MAC\_SECURITY**

**/\* Initialize the MAC Security \*/**

**Cllc\_securityInit(frameCounter);**

**#endif /\* FEATURE\_MAC\_SECURITY \*/**

**numDevices = Csf\_getNumDeviceListEntries();**

**/\* Restore with the network and device information \*/**

**Cllc\_restoreNetwork(&netInfo, (uint8\_t)numDevices, NULL);**

**restarted = true;**

**#ifdef FEATURE\_SECURE\_COMMISSIONING**

**if((numDevices >0 )&&(!CONFIG\_FH\_ENABLE))**

**{**

**fCommissionRequired = true;**

**keyRecoverDeviceNumber = numDevices;**

**}**

**#endif**

**}**

**else**

**{**

**restarted = false;**

**#ifdef FEATURE\_MAC\_SECURITY**

**/\* Initialize the MAC Security \*/**

**Cllc\_securityInit(frameCounter);**

**#endif /\* FEATURE\_MAC\_SECURITY \*/**

**/\* Start a new netork \*/**

**Cllc\_startNetwork();**

**}**

**}**

**/\*!**

**\* @brief Process the Config Response message.**

**\***

**\* @param pDataInd - pointer to the data indication information**

**\*/**

**static void processConfigResponse(ApiMac\_mcpsDataInd\_t \*pDataInd)**

**{**

**/\* Make sure the message is the correct size \*/**

**if(pDataInd->msdu.len == SMSGS\_CONFIG\_RESPONSE\_MSG\_LENGTH)**

**{**

**Cllc\_associated\_devices\_t \*pDev;**

**Smsgs\_configRspMsg\_t configRsp;**

**uint8\_t \*pBuf = pDataInd->msdu.p;**

**/\* Parse the message \*/**

**configRsp.cmdId = (Smsgs\_cmdIds\_t)\*pBuf++;**

**configRsp.status = (Smsgs\_statusValues\_t)Util\_buildUint16(pBuf[0],**

**pBuf[1]);**

**pBuf += 2;**

**configRsp.frameControl = Util\_buildUint16(pBuf[0], pBuf[1]);**

**pBuf += 2;**

**configRsp.reportingInterval = Util\_buildUint32(pBuf[0], pBuf[1],**

**pBuf[2],**

**pBuf[3]);**

**pBuf += 4;**

**configRsp.pollingInterval = Util\_buildUint32(pBuf[0], pBuf[1], pBuf[2],**

**pBuf[3]);**

**pDev = findDevice(&pDataInd->srcAddr);**

**if(pDev != NULL)**

**{**

**/\* Clear the sent flag and set the response flag \*/**

**pDev->status &= ~ASSOC\_CONFIG\_SENT;**

**pDev->status |= ASSOC\_CONFIG\_RSP;**

**}**

**/\* Report the config response \*/**

**Csf\_deviceConfigUpdate(&pDataInd->srcAddr, pDataInd->rssi,**

**&configRsp);**

**Util\_setEvent(&Collector\_events, COLLECTOR\_CONFIG\_EVT);**

**Collector\_statistics.configResponseReceived++;**

**}**

**}**

**/\*!**

**\* @brief Process the Tracking Response message.**

**\***

**\* @param pDataInd - pointer to the data indication information**

**\*/**

**static void processTrackingResponse(ApiMac\_mcpsDataInd\_t \*pDataInd)**

**{**

**/\* Make sure the message is the correct size \*/**

**if(pDataInd->msdu.len == SMSGS\_TRACKING\_RESPONSE\_MSG\_LENGTH)**

**{**

**Cllc\_associated\_devices\_t \*pDev;**

**pDev = findDevice(&pDataInd->srcAddr);**

**if(pDev != NULL)**

**{**

**if(pDev->status & ASSOC\_TRACKING\_SENT)**

**{**

**pDev->status &= ~ASSOC\_TRACKING\_SENT;**

**pDev->status |= ASSOC\_TRACKING\_RSP;**

**/\* Setup for next tracking \*/**

**Csf\_setTrackingClock( TRACKING\_DELAY\_TIME);**

**/\* Retry config request \*/**

**processConfigRetry();**

**}**

**}**

**/\* Update stats \*/**

**Collector\_statistics.trackingResponseReceived++;**

**}**

**}**

**/\*!**

**\* @brief Process the Toggle Led Response message.**

**\***

**\* @param pDataInd - pointer to the data indication information**

**\*/**

**static void processIdendifyLedRequest(ApiMac\_mcpsDataInd\_t \*pDataInd)**

**{**

**/\* Make sure the message is the correct size \*/**

**if(pDataInd->msdu.len == SMSGS\_INDENTIFY\_LED\_REQUEST\_MSG\_LEN)**

**{**

**Llc\_deviceListItem\_t item;**

**/\* Is the device a known device? \*/**

**if(Csf\_getDevice(&(pDataInd->srcAddr), &item))**

**{**

**uint8\_t cmdBytes[SMSGS\_INDENTIFY\_LED\_RESPONSE\_MSG\_LEN];**

**Csf\_identifyLED((pDataInd->msdu.p[1]) \* 1000);**

**/\* send the response message directly \*/**

**cmdBytes[0] = (uint8\_t) Smsgs\_cmdIds\_IdentifyLedRsp;**

**cmdBytes[1] = 0;**

**sendMsg(Smsgs\_cmdIds\_toggleLedRsp,**

**item.devInfo.shortAddress,**

**item.capInfo.rxOnWhenIdle,**

**SMSGS\_INDENTIFY\_LED\_RESPONSE\_MSG\_LEN,**

**cmdBytes);**

**}**

**}**

**}**

**/\*!**

**\* @brief Process the Toggle Led Response message.**

**\***

**\* @param pDataInd - pointer to the data indication information**

**\*/**

**static void processToggleLedResponse(ApiMac\_mcpsDataInd\_t \*pDataInd)**

**{**

**/\* Make sure the message is the correct size \*/**

**if(pDataInd->msdu.len == SMSGS\_TOGGLE\_LED\_RESPONSE\_MSG\_LEN)**

**{**

**bool ledState;**

**uint8\_t \*pBuf = pDataInd->msdu.p;**

**/\* Skip past the command ID \*/**

**pBuf++;**

**ledState = (bool)\*pBuf;**

**/\* Notify the user \*/**

**Csf\_toggleResponseReceived(&pDataInd->srcAddr, ledState);**

**}**

**}**

**/\*!**

**\* @brief Process the Sensor Data message.**

**\***

**\* @param pDataInd - pointer to the data indication information**

**\*/**

**static void processSensorData(ApiMac\_mcpsDataInd\_t \*pDataInd)**

**{**

**Smsgs\_sensorMsg\_t sensorData;**

**uint8\_t \*pBuf = pDataInd->msdu.p;**

**memset(&sensorData, 0, sizeof(Smsgs\_sensorMsg\_t));**

**/\* Parse the message \*/**

**sensorData.cmdId = (Smsgs\_cmdIds\_t)\*pBuf++;**

**memcpy(sensorData.extAddress, pBuf, SMGS\_SENSOR\_EXTADDR\_LEN);**

**pBuf += SMGS\_SENSOR\_EXTADDR\_LEN;**

**sensorData.frameControl = Util\_buildUint16(pBuf[0], pBuf[1]);**

**pBuf += 2;**

**/\* Parse data in order of frameControl mask, starting with LSB \*/**

**if(sensorData.frameControl & Smsgs\_dataFields\_tempSensor)**

**{**

**sensorData.tempSensor.ambienceTemp = Util\_buildUint16(pBuf[0], pBuf[1]);**

**pBuf += 2;**

**sensorData.tempSensor.objectTemp = Util\_buildUint16(pBuf[0], pBuf[1]);**

**pBuf += 2;**

**}**

**if(sensorData.frameControl & Smsgs\_dataFields\_lightSensor)**

**{**

**sensorData.lightSensor.rawData = Util\_buildUint16(pBuf[0], pBuf[1]);**

**pBuf += 2;**

**}**

**if(sensorData.frameControl & Smsgs\_dataFields\_humiditySensor)**

**{**

**sensorData.humiditySensor.temp = Util\_buildUint16(pBuf[0], pBuf[1]);**

**pBuf += 2;**

**sensorData.humiditySensor.humidity = Util\_buildUint16(pBuf[0], pBuf[1]);**

**pBuf += 2;**

**}**

**if(sensorData.frameControl & Smsgs\_dataFields\_msgStats)**

**{**

**sensorData.msgStats.joinAttempts = Util\_buildUint16(pBuf[0], pBuf[1]);**

**pBuf += 2;**

**sensorData.msgStats.joinFails = Util\_buildUint16(pBuf[0], pBuf[1]);**

**pBuf += 2;**

**sensorData.msgStats.msgsAttempted = Util\_buildUint16(pBuf[0], pBuf[1]);**

**pBuf += 2;**

**sensorData.msgStats.msgsSent = Util\_buildUint16(pBuf[0], pBuf[1]);**

**pBuf += 2;**

**sensorData.msgStats.trackingRequests = Util\_buildUint16(pBuf[0],**

**pBuf[1]);**

**pBuf += 2;**

**sensorData.msgStats.trackingResponseAttempts = Util\_buildUint16(**

**pBuf[0],**

**pBuf[1]);**

**pBuf += 2;**

**sensorData.msgStats.trackingResponseSent = Util\_buildUint16(pBuf[0],**

**pBuf[1]);**

**pBuf += 2;**

**sensorData.msgStats.configRequests = Util\_buildUint16(pBuf[0],**

**pBuf[1]);**

**pBuf += 2;**

**sensorData.msgStats.configResponseAttempts = Util\_buildUint16(**

**pBuf[0],**

**pBuf[1]);**

**pBuf += 2;**

**sensorData.msgStats.configResponseSent = Util\_buildUint16(pBuf[0],**

**pBuf[1]);**

**pBuf += 2;**

**sensorData.msgStats.channelAccessFailures = Util\_buildUint16(pBuf[0],**

**pBuf[1]);**

**pBuf += 2;**

**sensorData.msgStats.macAckFailures = Util\_buildUint16(pBuf[0], pBuf[1]);**

**pBuf += 2;**

**sensorData.msgStats.otherDataRequestFailures = Util\_buildUint16(**

**pBuf[0],**

**pBuf[1]);**

**pBuf += 2;**

**sensorData.msgStats.syncLossIndications = Util\_buildUint16(pBuf[0],**

**pBuf[1]);**

**pBuf += 2;**

**sensorData.msgStats.rxDecryptFailures = Util\_buildUint16(pBuf[0],**

**pBuf[1]);**

**pBuf += 2;**

**sensorData.msgStats.txEncryptFailures = Util\_buildUint16(pBuf[0],**

**pBuf[1]);**

**pBuf += 2;**

**sensorData.msgStats.resetCount = Util\_buildUint16(pBuf[0],**

**pBuf[1]);**

**pBuf += 2;**

**sensorData.msgStats.lastResetReason = Util\_buildUint16(pBuf[0],**

**pBuf[1]);**

**pBuf += 2;**

**sensorData.msgStats.joinTime = Util\_buildUint16(pBuf[0],**

**pBuf[1]);**

**pBuf += 2;**

**sensorData.msgStats.interimDelay = Util\_buildUint16(pBuf[0],**

**pBuf[1]);**

**pBuf += 2;**

**sensorData.msgStats.numBroadcastMsgRcvd = Util\_buildUint16(pBuf[0],**

**pBuf[1]);**

**pBuf += 2;**

**sensorData.msgStats.numBroadcastMsglost = Util\_buildUint16(pBuf[0],**

**pBuf[1]);**

**pBuf += 2;**

**sensorData.msgStats.avgE2EDelay = Util\_buildUint16(pBuf[0],pBuf[1]);**

**pBuf += 2;**

**sensorData.msgStats.worstCaseE2EDelay = Util\_buildUint16(pBuf[0],pBuf[1]);**

**pBuf += 2;**

**}**

**if(sensorData.frameControl & Smsgs\_dataFields\_configSettings)**

**{**

**sensorData.configSettings.reportingInterval = Util\_buildUint32(pBuf[0],**

**pBuf[1],**

**pBuf[2],**

**pBuf[3]);**

**pBuf += 4;**

**sensorData.configSettings.pollingInterval = Util\_buildUint32(pBuf[0],**

**pBuf[1],**

**pBuf[2],**

**pBuf[3]);**

**}**

**Collector\_statistics.sensorMessagesReceived++;**

**/\* Report the sensor data \*/**

**Csf\_deviceSensorDataUpdate(&pDataInd->srcAddr, pDataInd->rssi,**

**&sensorData);**

**processDataRetry(&(pDataInd->srcAddr));**

**}**

**/\*!**

**\* @brief Find the associated device table entry matching pAddr.**

**\***

**\* @param pAddr - pointer to device's address**

**\***

**\* @return pointer to the associated device table entry,**

**\* NULL if not found.**

**\*/**

**static Cllc\_associated\_devices\_t \*findDevice(ApiMac\_sAddr\_t \*pAddr)**

**{**

**int x;**

**Cllc\_associated\_devices\_t \*pItem = NULL;**

**/\* Check for invalid parameters \*/**

**if((pAddr == NULL) || (pAddr->addrMode == ApiMac\_addrType\_none))**

**{**

**return (NULL);**

**}**

**for(x = 0; x < CONFIG\_MAX\_DEVICES; x++)**

**{**

**/\* Make sure the entry is valid. \*/**

**if(Cllc\_associatedDevList[x].shortAddr != CSF\_INVALID\_SHORT\_ADDR)**

**{**

**if(pAddr->addrMode == ApiMac\_addrType\_short)**

**{**

**if(pAddr->addr.shortAddr == Cllc\_associatedDevList[x].shortAddr)**

**{**

**pItem = &Cllc\_associatedDevList[x];**

**break;**

**}**

**}**

**}**

**}**

**return (pItem);**

**}**

**/\*!**

**\* @brief Find the associated device table entry matching status bit.**

**\***

**\* @param statusBit - what status bit to find**

**\***

**\* @return pointer to the associated device table entry,**

**\* NULL if not found.**

**\*/**

**static Cllc\_associated\_devices\_t \*findDeviceStatusBit(uint16\_t mask, uint16\_t statusBit)**

**{**

**int x;**

**Cllc\_associated\_devices\_t \*pItem = NULL;**

**for(x = 0; x < CONFIG\_MAX\_DEVICES; x++)**

**{**

**/\* Make sure the entry is valid. \*/**

**if(Cllc\_associatedDevList[x].shortAddr != CSF\_INVALID\_SHORT\_ADDR)**

**{**

**if((Cllc\_associatedDevList[x].status & mask) == statusBit)**

**{**

**pItem = &Cllc\_associatedDevList[x];**

**break;**

**}**

**}**

**}**

**return (pItem);**

**}**

**/\*!**

**\* @brief Get the next MSDU Handle**

**\* <BR>**

**\* The MSDU handle has 3 parts:<BR>**

**\* - The MSBit(7), when set means the the application sent the message**

**\* - Bit 6, when set means that the app message is a config request**

**\* - Bits 0-5, used as a message counter that rolls over.**

**\***

**\* @param msgType - message command id needed**

**\***

**\* @return msdu Handle**

**\*/**

**static uint8\_t getMsduHandle(Smsgs\_cmdIds\_t msgType)**

**{**

**uint8\_t msduHandle = deviceTxMsduHandle;**

**/\* Increment for the next msdu handle, or roll over \*/**

**if(deviceTxMsduHandle >= MSDU\_HANDLE\_MAX)**

**{**

**deviceTxMsduHandle = 0;**

**}**

**else**

**{**

**deviceTxMsduHandle++;**

**}**

**/\* Add the message type bit for ramp data \*/**

**if(msgType == Smsgs\_cmdIds\_rampdata)**

**{**

**msduHandle |= RAMP\_DATA\_MSDU\_HANDLE;**

**return (msduHandle);**

**}**

**/\* Add the App specific bit \*/**

**msduHandle |= APP\_MARKER\_MSDU\_HANDLE;**

**/\* Add the message type bit \*/**

**if(msgType == Smsgs\_cmdIds\_configReq)**

**{**

**msduHandle |= APP\_CONFIG\_MSDU\_HANDLE;**

**}**

**else if(msgType == Smgs\_cmdIds\_broadcastCtrlMsg)**

**{**

**msduHandle |= APP\_BROADCAST\_MSDU\_HANDLE;**

**}**

**return (msduHandle);**

**}**

**/\*!**

**\* @brief Send MAC data request**

**\***

**\* @param type - message type**

**\* @param dstShortAddr - destination short address**

**\* @param rxOnIdle - true if not a sleepy device**

**\* @param len - length of payload**

**\* @param pData - pointer to the buffer**

**\***

**\* @return true if sent, false if not**

**\*/**

**static bool sendMsg(Smsgs\_cmdIds\_t type, uint16\_t dstShortAddr, bool rxOnIdle,**

**uint16\_t len,**

**uint8\_t \*pData)**

**{**

**ApiMac\_mcpsDataReq\_t dataReq;**

**/\* Fill the data request field \*/**

**memset(&dataReq, 0, sizeof(ApiMac\_mcpsDataReq\_t));**

**dataReq.dstAddr.addrMode = ApiMac\_addrType\_short;**

**dataReq.dstAddr.addr.shortAddr = dstShortAddr;**

**dataReq.srcAddrMode = ApiMac\_addrType\_short;**

**if(fhEnabled && rxOnIdle)**

**{**

**Llc\_deviceListItem\_t item;**

**if(Csf\_getDevice(&(dataReq.dstAddr), &item))**

**{**

**/\* Switch to the long address \*/**

**dataReq.dstAddr.addrMode = ApiMac\_addrType\_extended;**

**memcpy(&dataReq.dstAddr.addr.extAddr, &item.devInfo.extAddress,**

**(APIMAC\_SADDR\_EXT\_LEN));**

**dataReq.srcAddrMode = ApiMac\_addrType\_extended;**

**}**

**else**

**{**

**/\* Can't send the message \*/**

**return (false);**

**}**

**}**

**dataReq.dstPanId = devicePanId;**

**dataReq.msduHandle = getMsduHandle(type);**

**dataReq.txOptions.ack = true;**

**if(rxOnIdle == false)**

**{**

**dataReq.txOptions.indirect = true;**

**}**

**dataReq.msdu.len = len;**

**dataReq.msdu.p = pData;**

**#ifdef FEATURE\_MAC\_SECURITY**

**#ifdef FEATURE\_SECURE\_COMMISSIONING**

**{**

**Llc\_deviceListItem\_t deviceItem;**

**if(Csf\_getDevice(&(dataReq.dstAddr), &deviceItem)) {**

**SM\_getSrcDeviceSecurityInfo(deviceItem.devInfo.extAddress,deviceItem.devInfo.shortAddress, &dataReq.sec);**

**}**

**else {**

**/\* Can't send the message \*/**

**return (false);**

**}**

**}**

**#else**

**/\* Fill in the appropriate security fields \*/**

**Cllc\_securityFill(&dataReq.sec);**

**#endif /\* FEATURE\_SECURE\_COMMISSIONING \*/**

**#endif /\* FEATURE\_MAC\_SECURITY \*/**

**/\* Send the message \*/**

**if(ApiMac\_mcpsDataReq(&dataReq) != ApiMac\_status\_success)**

**{**

**/\* Transaction overflow occurred \*/**

**return (false);**

**}**

**else**

**{**

**return (true);**

**}**

**}**

**/\*!**

**\* @brief Send MAC broadcast data request. Only supported in FH mode.**

**\***

**\* @param type - message type**

**\* @param len - length of payload**

**\* @param pData - pointer to the buffer**

**\*/**

**static void sendBroadcastMsg(Smsgs\_cmdIds\_t type, uint16\_t len,**

**uint8\_t \*pData)**

**{**

**ApiMac\_mcpsDataReq\_t dataReq;**

**/\* Only supported for FH mode \*/**

**if(!fhEnabled)**

**{**

**return;**

**}**

**/\* Fill the data request field \*/**

**memset(&dataReq, 0, sizeof(ApiMac\_mcpsDataReq\_t));**

**dataReq.dstAddr.addrMode = ApiMac\_addrType\_none;**

**dataReq.srcAddrMode = ApiMac\_addrType\_short;**

**dataReq.dstPanId = devicePanId;**

**dataReq.msduHandle = getMsduHandle(type);**

**dataReq.txOptions.ack = false;**

**dataReq.txOptions.indirect = false;**

**dataReq.msdu.len = len;**

**dataReq.msdu.p = pData;**

**#ifdef FEATURE\_MAC\_SECURITY**

**/\* Fill in the appropriate security fields \*/**

**Cllc\_securityFill(&dataReq.sec);**

**#endif /\* FEATURE\_MAC\_SECURITY \*/**

**/\* Send the message \*/**

**ApiMac\_mcpsDataReq(&dataReq);**

**}**

**/\*!**

**@brief Build and send fixed size ramp data to the first associated device.**

**\*/**

**#ifdef POWER\_MEAS**

**void generateIndirectRampMsg()**

**{**

**uint8\_t \*pMsgBuf;**

**uint16\_t index;**

**uint16\_t sensorShortAddr;**

**sensorShortAddr = Cllc\_associatedDevList[0].shortAddr;**

**if ((sensorShortAddr == CSF\_INVALID\_SHORT\_ADDR) ||**

**(!COLLECTOR\_TEST\_RAMP\_DATA\_SIZE))**

**{**

**return;**

**}**

**pMsgBuf = (uint8\_t \*)Csf\_malloc(COLLECTOR\_TEST\_RAMP\_DATA\_SIZE);**

**if(pMsgBuf)**

**{**

**uint8\_t \*pBuf = pMsgBuf;**

**\*pBuf++ = (uint8\_t)Smsgs\_cmdIds\_rampdata;**

**for(index = 1; index < COLLECTOR\_TEST\_RAMP\_DATA\_SIZE; index++)**

**{**

**\*pBuf++ = (uint8\_t) (index & 0xFF);**

**}**

**sendMsg(Smsgs\_cmdIds\_rampdata, sensorShortAddr, false,**

**COLLECTOR\_TEST\_RAMP\_DATA\_SIZE, pMsgBuf);**

**Csf\_free(pMsgBuf);**

**}**

**}**

**#endif**

**/\*!**

**\* @brief Generate Config Requests for all associate devices**

**\* that need one.**

**\*/**

**static void generateConfigRequests(void)**

**{**

**#ifndef POWER\_MEAS**

**int x;**

**if(CERTIFICATION\_TEST\_MODE)**

**{**

**/\* In Certification mode only back to back uplink**

**\* data traffic shall be supported\*/**

**return;**

**}**

**/\* Clear any timed out transactions \*/**

**for(x = 0; x < CONFIG\_MAX\_DEVICES; x++)**

**{**

**if((Cllc\_associatedDevList[x].shortAddr != CSF\_INVALID\_SHORT\_ADDR)**

**&& (Cllc\_associatedDevList[x].status & CLLC\_ASSOC\_STATUS\_ALIVE))**

**{**

**if((Cllc\_associatedDevList[x].status &**

**(ASSOC\_CONFIG\_SENT | ASSOC\_CONFIG\_RSP))**

**== (ASSOC\_CONFIG\_SENT | ASSOC\_CONFIG\_RSP))**

**{**

**Cllc\_associatedDevList[x].status &= ~(ASSOC\_CONFIG\_SENT**

**| ASSOC\_CONFIG\_RSP);**

**}**

**}**

**}**

**/\* Make sure we are only sending one config request at a time \*/**

**if(findDeviceStatusBit(ASSOC\_CONFIG\_MASK, ASSOC\_CONFIG\_SENT) == NULL)**

**{**

**/\* Run through all of the devices \*/**

**for(x = 0; x < CONFIG\_MAX\_DEVICES; x++)**

**{**

**/\* Make sure the entry is valid. \*/**

**if((Cllc\_associatedDevList[x].shortAddr != CSF\_INVALID\_SHORT\_ADDR)**

**&& (Cllc\_associatedDevList[x].status & CLLC\_ASSOC\_STATUS\_ALIVE))**

**{**

**uint16\_t status = Cllc\_associatedDevList[x].status;**

**/\***

**Has the device been sent or already received a config request?**

**\*/**

**if(((status & (ASSOC\_CONFIG\_SENT | ASSOC\_CONFIG\_RSP)) == 0))**

**{**

**ApiMac\_sAddr\_t dstAddr;**

**Collector\_status\_t stat;**

**/\* Set up the destination address \*/**

**dstAddr.addrMode = ApiMac\_addrType\_short;**

**dstAddr.addr.shortAddr =**

**Cllc\_associatedDevList[x].shortAddr;**

**/\* Send the Config Request \*/**

**stat = Collector\_sendConfigRequest(**

**&dstAddr, (CONFIG\_FRAME\_CONTROL),**

**(CONFIG\_REPORTING\_INTERVAL),**

**(CONFIG\_POLLING\_INTERVAL));**

**if(stat == Collector\_status\_success)**

**{**

**/\***

**Mark as the message has been sent and expecting a response**

**\*/**

**Cllc\_associatedDevList[x].status |= ASSOC\_CONFIG\_SENT;**

**Cllc\_associatedDevList[x].status &= ~ASSOC\_CONFIG\_RSP;**

**}**

**/\* Only do one at a time \*/**

**break;**

**}**

**}**

**}**

**}**

**#endif**

**}**

**/\*!**

**\* @brief Generate Tracking Requests for all associate devices**

**\* that need one.**

**\*/**

**static void generateTrackingRequests(void)**

**{**

**#ifndef POWER\_MEAS**

**int x;**

**/\* Run through all of the devices, looking for previous activity \*/**

**for(x = 0; x < CONFIG\_MAX\_DEVICES; x++)**

**{**

**if(CERTIFICATION\_TEST\_MODE)**

**{**

**/\* In Certification mode only back to back uplink**

**\* data traffic shall be supported\*/**

**return;**

**}**

**/\* Make sure the entry is valid. \*/**

**if((Cllc\_associatedDevList[x].shortAddr != CSF\_INVALID\_SHORT\_ADDR)**

**&& (Cllc\_associatedDevList[x].status & CLLC\_ASSOC\_STATUS\_ALIVE))**

**{**

**uint16\_t status = Cllc\_associatedDevList[x].status;**

**/\***

**Has the device been sent a tracking request or received a**

**tracking response?**

**\*/**

**if(status & ASSOC\_TRACKING\_RETRY)**

**{**

**sendTrackingRequest(&Cllc\_associatedDevList[x]);**

**return;**

**}**

**else if((status & (ASSOC\_TRACKING\_SENT | ASSOC\_TRACKING\_RSP**

**| ASSOC\_TRACKING\_ERROR)))**

**{**

**Cllc\_associated\_devices\_t \*pDev = NULL;**

**int y;**

**if(status & (ASSOC\_TRACKING\_SENT | ASSOC\_TRACKING\_ERROR))**

**{**

**ApiMac\_deviceDescriptor\_t devInfo;**

**Llc\_deviceListItem\_t item;**

**ApiMac\_sAddr\_t devAddr;**

**/\***

**Timeout occured, notify the user that the tracking**

**failed.**

**\*/**

**memset(&devInfo, 0, sizeof(ApiMac\_deviceDescriptor\_t));**

**devAddr.addrMode = ApiMac\_addrType\_short;**

**devAddr.addr.shortAddr =**

**Cllc\_associatedDevList[x].shortAddr;**

**if(Csf\_getDevice(&devAddr, &item))**

**{**

**memcpy(&devInfo.extAddress,**

**&item.devInfo.extAddress,**

**sizeof(ApiMac\_sAddrExt\_t));**

**}**

**devInfo.shortAddress = Cllc\_associatedDevList[x].shortAddr;**

**devInfo.panID = devicePanId;**

**Csf\_deviceNotActiveUpdate(&devInfo,**

**((status & ASSOC\_TRACKING\_SENT) ? true : false));**

**/\* Not responding, so remove the alive marker \*/**

**Cllc\_associatedDevList[x].status**

**&= ~(CLLC\_ASSOC\_STATUS\_ALIVE**

**| ASSOC\_CONFIG\_SENT | ASSOC\_CONFIG\_RSP);**

**}**

**/\* Clear the tracking bits \*/**

**Cllc\_associatedDevList[x].status &= ~(ASSOC\_TRACKING\_ERROR**

**| ASSOC\_TRACKING\_SENT | ASSOC\_TRACKING\_RSP);**

**/\* Find the next valid device \*/**

**y = x;**

**while(pDev == NULL)**

**{**

**/\* Check for rollover \*/**

**if(y == (CONFIG\_MAX\_DEVICES-1))**

**{**

**/\* Move to the beginning \*/**

**y = 0;**

**}**

**else**

**{**

**/\* Move the the next device \*/**

**y++;**

**}**

**if(y == x)**

**{**

**/\* We've come back around \*/**

**break;**

**}**

**/\***

**Is the entry valid and active \*/**

**if((Cllc\_associatedDevList[y].shortAddr**

**!= CSF\_INVALID\_SHORT\_ADDR)**

**&& (Cllc\_associatedDevList[y].status**

**& CLLC\_ASSOC\_STATUS\_ALIVE))**

**{**

**pDev = &Cllc\_associatedDevList[y];**

**}**

**}**

**if(pDev == NULL)**

**{**

**/\* Another device wasn't found, send to same device \*/**

**pDev = &Cllc\_associatedDevList[x];**

**}**

**sendTrackingRequest(pDev);**

**/\* Only do one at a time \*/**

**return;**

**}**

**}**

**}**

**/\* If no activity found, find the first active device \*/**

**for(x = 0; x < CONFIG\_MAX\_DEVICES; x++)**

**{**

**/\* Make sure the entry is valid. \*/**

**if((Cllc\_associatedDevList[x].shortAddr != CSF\_INVALID\_SHORT\_ADDR)**

**&& (Cllc\_associatedDevList[x].status & CLLC\_ASSOC\_STATUS\_ALIVE))**

**{**

**sendTrackingRequest(&Cllc\_associatedDevList[x]);**

**break;**

**}**

**}**

**if(x == CONFIG\_MAX\_DEVICES)**

**{**

**/\* No device found, Setup delay for next tracking message \*/**

**Csf\_setTrackingClock(TRACKING\_DELAY\_TIME);**

**}**

**#endif**

**}**

**/\*!**

**\* @brief Generate Broadcast Cmd Request Message**

**\*/**

**static void generateBroadcastCmd(void)**

**{**

**uint8\_t buffer[SMSGS\_BROADCAST\_CMD\_LENGTH];**

**uint8\_t \*pBuf = buffer;**

**/\* Build the message \*/**

**\*pBuf++ = (uint8\_t)Smgs\_cmdIds\_broadcastCtrlMsg;**

**\*pBuf++ = Util\_loUint16(Collector\_statistics.broadcastMsgSentCnt);**

**\*pBuf++ = Util\_hiUint16(Collector\_statistics.broadcastMsgSentCnt);**

**sendBroadcastMsg(Smgs\_cmdIds\_broadcastCtrlMsg, SMSGS\_BROADCAST\_CMD\_LENGTH,**

**buffer);**

**}**

**/\*!**

**\* @brief Generate Tracking Requests for a device**

**\***

**\* @param pDev - pointer to the device's associate device table entry**

**\*/**

**static void sendTrackingRequest(Cllc\_associated\_devices\_t \*pDev)**

**{**

**uint8\_t cmdId = Smsgs\_cmdIds\_trackingReq;**

**/\* Send the Tracking Request \*/**

**if((sendMsg(Smsgs\_cmdIds\_trackingReq, pDev->shortAddr,**

**pDev->capInfo.rxOnWhenIdle,**

**(SMSGS\_TRACKING\_REQUEST\_MSG\_LENGTH),**

**&cmdId)) == true)**

**{**

**/\* Mark as Tracking Request sent \*/**

**pDev->status |= ASSOC\_TRACKING\_SENT;**

**/\* Setup Timeout for response \*/**

**Csf\_setTrackingClock(TRACKING\_TIMEOUT\_TIME);**

**/\* Update stats \*/**

**Collector\_statistics.trackingRequestAttempts++;**

**}**

**else**

**{**

**ApiMac\_sAddr\_t devAddr;**

**devAddr.addrMode = ApiMac\_addrType\_short;**

**devAddr.addr.shortAddr = pDev->shortAddr;**

**processDataRetry(&devAddr);**

**}**

**}**

**/\*!**

**\* @brief Process the MAC Comm Status Indication Callback**

**\***

**\* @param pCommStatusInd - Comm Status indication**

**\*/**

**static void commStatusIndCB(ApiMac\_mlmeCommStatusInd\_t \*pCommStatusInd)**

**{**

**if(pCommStatusInd->reason == ApiMac\_commStatusReason\_assocRsp)**

**{**

**if(pCommStatusInd->status != ApiMac\_status\_success)**

**{**

**Cllc\_associated\_devices\_t \*pDev;**

**pDev = findDevice(&pCommStatusInd->dstAddr);**

**if(pDev)**

**{**

**/\* Mark as inactive and clear config and tracking states \*/**

**pDev->status = 0;**

**}**

**}**

**}**

**}**

**/\*!**

**\* @brief Process the MAC Poll Indication Callback**

**\***

**\* @param pPollInd - poll indication**

**\*/**

**static void pollIndCB(ApiMac\_mlmePollInd\_t \*pPollInd)**

**{**

**ApiMac\_sAddr\_t addr;**

**addr.addrMode = ApiMac\_addrType\_short;**

**if (pPollInd->srcAddr.addrMode == ApiMac\_addrType\_short)**

**{**

**addr.addr.shortAddr = pPollInd->srcAddr.addr.shortAddr;**

**}**

**else**

**{**

**addr.addr.shortAddr = Csf\_getDeviceShort(**

**&pPollInd->srcAddr.addr.extAddr);**

**}**

**processDataRetry(&addr);**

**}**

**/\*!**

**\* @brief Process retries for config and tracking messages**

**\***

**\* @param addr - MAC address structure \*/**

**static void processDataRetry(ApiMac\_sAddr\_t \*pAddr)**

**{**

**if(pAddr->addr.shortAddr != CSF\_INVALID\_SHORT\_ADDR)**

**{**

**Cllc\_associated\_devices\_t \*pItem;**

**pItem = findDevice(pAddr);**

**if(pItem)**

**{**

**/\* Set device status to alive \*/**

**pItem->status |= CLLC\_ASSOC\_STATUS\_ALIVE;**

**/\* Check to see if we need to send it a config \*/**

**if((pItem->status & (ASSOC\_CONFIG\_RSP | ASSOC\_CONFIG\_SENT)) == 0)**

**{**

**processConfigRetry();**

**}**

**/\* Check to see if we need to send it a tracking message \*/**

**if((pItem->status & (ASSOC\_TRACKING\_SENT| ASSOC\_TRACKING\_RETRY)) == 0)**

**{**

**/\* Make sure we aren't already doing a tracking message \*/**

**if(((Collector\_events & COLLECTOR\_TRACKING\_TIMEOUT\_EVT) == 0)**

**&& (Csf\_isTrackingTimerActive() == false)**

**&& (findDeviceStatusBit(ASSOC\_TRACKING\_MASK,**

**ASSOC\_TRACKING\_SENT) == NULL))**

**{**

**/\* Setup for next tracking \*/**

**Csf\_setTrackingClock(TRACKING\_DELAY\_TIME);**

**}**

**}**

**}**

**}**

**}**

**/\*!**

**\* @brief Process retries for config messages**

**\*/**

**static void processConfigRetry(void)**

**{**

**/\* Retry config request if not already sent \*/**

**if(((Collector\_events & COLLECTOR\_CONFIG\_EVT) == 0)**

**&& (Csf\_isConfigTimerActive() == false))**

**{**

**/\* Set config event \*/**

**Csf\_setConfigClock(CONFIG\_DELAY);**

**}**

**}**

**#ifdef FEATURE\_SECURE\_COMMISSIONING**

**/\*!**

**\* @brief Security manager failure processing function**

**\*/**

**void smFailCMProcessCb(ApiMac\_deviceDescriptor\_t \*devInfo, bool rxOnIdle,**

**bool keyRefreshment)**

**{**

**/\* Re-enable joining if it was enabled prior to key refreshment process, or**

**\* if commissioning was a result of device association as network must have**

**\* been open for association to occur**

**\*/**

**if (keyRefreshment) {**

**if (permitJoining) {**

**// the permitJoining was allowed before, so let's allow it**

**uint32\_t duration = 0xFFFFFFFF;**

**Cllc\_setJoinPermit(duration);**

**}**

**}**

**else if (fCommissionRequired == false) {**

**// secure commissioning. The permitJoining was allowed before, so let's allow it**

**uint32\_t duration = 0xFFFFFFFF;**

**Cllc\_setJoinPermit(duration);**

**}**

**if (keyRefreshment == true)**

**{ /\* Device failed key refreshment process \*/**

**LCD\_WRITE\_STRING\_VALUE("Key Refresh Failed: 0x",**

**devInfo->shortAddress, 16, 6);**

**}**

**else**

**{ /\* Device failed commissioning process \*/**

**LCD\_WRITE\_STRING\_VALUE("Commissioning Failed: 0x",**

**devInfo->shortAddress, 16, 6);**

**/\* Send a disassociation request to the device \*/**

**Cllc\_sendDisassociationRequest(devInfo->shortAddress, rxOnIdle);**

**/\* Remove device from the NV list \*/**

**Cllc\_removeDevice(&devInfo->extAddress);**

**Util\_setEvent(&Collector\_events, COLLECTOR\_CONFIG\_EVT);**

**}**

**}**

**/\*!**

**\* @brief Security manager success processing function**

**\*/**

**void smSuccessCMProcessCb(ApiMac\_deviceDescriptor\_t \*devInfo, bool keyRefreshment)**

**{**

**/\* Re-enable joining if it was enabled prior to key refreshment process, or**

**\* if commissioning was a result of device association as network must have**

**\* been open for association to occur**

**\*/**

**if (keyRefreshment) {**

**if (permitJoining) {**

**// the permitJoining was allowed before, so let's allow it**

**uint32\_t duration = 0xFFFFFFFF;**

**Cllc\_setJoinPermit(duration);**

**}**

**}**

**else if (fCommissionRequired == false) {**

**// secure commissioning. The permitJoining was allowed before, so let's allow it**

**uint32\_t duration = 0xFFFFFFFF;**

**Cllc\_setJoinPermit(duration);**

**}**

**if (keyRefreshment == true)**

**{ /\* Device successfully finished key refreshment process \*/**

**LCD\_WRITE\_STRING\_VALUE("Key Refreshed: 0x",**

**devInfo->shortAddress, 16, 6);**

**}**

**else**

**{ /\* Device successfully finished commissioning process \*/**

**LCD\_WRITE\_STRING\_VALUE("Commissioned: 0x",**

**devInfo->shortAddress, 16, 6);**

**Util\_setEvent(&Collector\_events, COLLECTOR\_CONFIG\_EVT);**

**}**

**}**

**#endif /\* FEATURE\_SECURE\_COMMISSIONING \*/**

**/\*!**

**\* @brief Process Orphan indication callback**

**\***

**\* @param pData - pointer to orphan indication callback structure**

**\*/**

**static void orphanIndCb(ApiMac\_mlmeOrphanInd\_t \*pData)**

**{**

**uint16\_t shortAddr = Csf\_getDeviceShort(&pData->orphanAddress);**

**/\* get the short address of the device \*/**

**if(CSF\_INVALID\_SHORT\_ADDR != shortAddr)**

**{**

**Csf\_IndicateOrphanReJoin(shortAddr);**

**}**

**}**

**Main.c**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**@file main.c**

**@brief main entry of the example application**

**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.**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Includes**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

**#include <xdc/std.h>**

**#include <xdc/runtime/Error.h>**

**#include <xdc/runtime/System.h>**

**#include <ti/sysbios/BIOS.h>**

**#include <ti/sysbios/knl/Task.h>**

**#include <ioc.h>**

**#include "sys\_ctrl.h"**

**#include "Board.h"**

**#include <inc/hw\_ccfg.h>**

**#include <inc/hw\_ccfg\_simple\_struct.h>**

**/\* Header files required for the temporary idle task function \*/**

**#include <ti/drivers/Power.h>**

**#include <ti/drivers/power/PowerCC26XX.h>**

**#include <aon\_rtc.h>**

**#include <prcm.h>**

**#if defined(FEATURE\_OAD)**

**#include <ti/drivers/SPI.h>**

**#endif**

**/\* Header files required to enable instruction fetch cache \*/**

**#include <ti/sysbios/hal/Hwi.h>**

**#include "cpu.h"**

**#include "collector.h"**

**#ifdef NV\_RESTORE**

**#include "macconfig.h"**

**#ifdef ONE\_PAGE\_NV**

**#include "nvocop.h"**

**#else**

**#include "nvoctp.h"**

**#endif**

**#endif**

**#include <string.h>**

**#ifdef OSAL\_PORT2TIRTOS**

**#include "macTask.h"**

**#include"rom\_jt.h"**

**#else**

**#include "api\_mac.h"**

**#include "icall.h"**

**#endif**

**#include "uart\_printf.h"**

**#if defined(RESET\_ASSERT)**

**#include "csf.h"**

**#endif**

**#ifndef USE\_DEFAULT\_USER\_CFG**

**#include "mac\_user\_config.h"**

**/\* MAC user defined configuration \*/**

**macUserCfg\_t macUser0Cfg[] = MAC\_USER\_CFG;**

**#endif /\* USE\_DEFAULT\_USER\_CFG \*/**

**#if ((CONFIG\_RANGE\_EXT\_MODE == APIMAC\_HIGH\_GAIN\_MODE) && \**

**defined(DeviceFamily\_CC13X0) && !defined(FREQ\_2\_4G))**

**#include "board\_palna.h"**

**#endif**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Constants**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

**/\* Assert Reasons \*/**

**#define MAIN\_ASSERT\_ICALL 2**

**#define MAIN\_ASSERT\_MAC 3**

**#define MAIN\_ASSERT\_HWI\_TIRTOS 4**

**#define RFC\_MODE\_BLE PRCM\_RFCMODESEL\_CURR\_MODE1**

**#define RFC\_MODE\_IEEE PRCM\_RFCMODESEL\_CURR\_MODE2**

**#define RFC\_MODE\_ANT PRCM\_RFCMODESEL\_CURR\_MODE4**

**#define RFC\_MODE\_EVERYTHING\_BUT\_ANT PRCM\_RFCMODESEL\_CURR\_MODE5**

**#define RFC\_MODE\_EVERYTHING PRCM\_RFCMODESEL\_CURR\_MODE6**

**/\* Extended Address offset in FCFG (LSB..MSB) \*/**

**#define EXTADDR\_OFFSET 0x2F0**

**#if defined(DeviceFamily\_CC13X2) || (DeviceFamily\_CC26X2)**

**#define APP\_TASK\_STACK\_SIZE 1536**

**#else**

**#define APP\_TASK\_STACK\_SIZE 960**

**#endif**

**#define APP\_TASK\_PRIORITY 1**

**#define SET\_RFC\_MODE(mode) HWREG( PRCM\_BASE + PRCM\_O\_RFCMODESEL ) = (mode)**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**External Variables**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

**extern ApiMac\_sAddrExt\_t ApiMac\_extAddr;**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Global Variables**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

**Task\_Struct appTask; /\* not static so you can see in ROV \*/**

**static uint8\_t appTaskStack[APP\_TASK\_STACK\_SIZE];**

**#ifdef OSAL\_PORT2TIRTOS**

**static uint8\_t \_macTaskId;**

**#endif**

**/\***

**When assert happens, this field will be filled with the reason:**

**MAIN\_ASSERT\_HWI\_TIRTOS,**

**MAIN\_ASSERT\_ICALL,**

**MAIN\_ASSERT\_MAC**

**\*/**

**uint8 Main\_assertReason = 0;**

**#ifdef NV\_RESTORE**

**mac\_Config\_t Main\_user1Cfg =**

**{ 0 };**

**#endif**

**#if defined(BOARD\_DISPLAY\_USE\_UART)**

**UART\_Params uartParams;**

**#endif**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Local Variables**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

**/\* Used to check for a valid extended address \*/**

**static const uint8\_t dummyExtAddr[] =**

**{ 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF };**

**#ifdef NV\_RESTORE**

**#ifdef ONE\_PAGE\_NV**

**/\* NVOCOP load API pointers \*/**

**static void NVOCOP\_loadApiPtrs(NVINTF\_nvFuncts\_t \*pfn)**

**{**

**// Load caller's structure with pointers to the NV API functions**

**pfn->initNV = &NVOCOP\_initNV;**

**pfn->compactNV = &NVOCOP\_compactNV;**

**pfn->createItem = NULL;**

**pfn->deleteItem = &NVOCOP\_deleteItem;**

**pfn->readItem = &NVOCOP\_readItem;**

**pfn->writeItem = &NVOCOP\_writeItem;**

**pfn->writeItemEx = NULL;**

**pfn->getItemLen = NULL;**

**}**

**#endif**

**#endif**

**/\*!**

**\* @brief Fill in your own assert function.**

**\***

**\* @param assertReason - reason: MAIN\_ASSERT\_HWI\_TIRTOS,**

**\* MAIN\_ASSERT\_ICALL, or**

**\* MAIN\_ASSERT\_MAC**

**\*/**

**void Main\_assertHandler(uint8\_t assertReason)**

**{**

**Main\_assertReason = assertReason;**

**#if defined(RESET\_ASSERT)**

**Csf\_assertInd(assertReason);**

**/\* Pull the plug and start over \*/**

**SysCtrlSystemReset();**

**#else**

**Hwi\_disable();**

**while(1)**

**{**

**/\* Put you code here to do something if in assert \*/**

**}**

**#endif**

**}**

**/\*!**

**\* @brief Main task function**

**\***

**\* @param a0 -**

**\* @param a1 -**

**\*/**

**Void appTaskFxn(UArg a0, UArg a1)**

**{**

**#ifdef TIMAC\_AGAMA\_FPGA**

**/\* FPGA build disables POWER constraints \*/**

**Power\_setConstraint(PowerCC26XX\_IDLE\_PD\_DISALLOW);**

**Power\_setConstraint(PowerCC26XX\_SB\_DISALLOW);**

**IOCPortConfigureSet(IOID\_20, IOC\_PORT\_RFC\_GPO0, IOC\_STD\_OUTPUT);**

**IOCPortConfigureSet(IOID\_18, IOC\_PORT\_RFC\_GPI0, IOC\_STD\_INPUT);**

**// configure RF Core SMI Command Link**

**IOCPortConfigureSet(IOID\_22, IOC\_IOCFG0\_PORT\_ID\_RFC\_SMI\_CL\_OUT, IOC\_STD\_OUTPUT);**

**IOCPortConfigureSet(IOID\_21, IOC\_IOCFG0\_PORT\_ID\_RFC\_SMI\_CL\_IN, IOC\_STD\_INPUT);**

**#else**

**/\***

**Disallow shutting down JTAG, VIMS, SYSBUS during idle state**

**since TIMAC requires SYSBUS during idle.**

**\*/**

**#endif**

**#ifndef OSAL\_PORT2TIRTOS**

**/\* Initialize ICall module \*/**

**ICall\_init();**

**#endif**

**/\* Copy the extended address from the CCFG area \*/**

**memcpy(ApiMac\_extAddr, (uint8\_t \*)&(\_\_ccfg.CCFG\_IEEE\_MAC\_0),**

**(APIMAC\_SADDR\_EXT\_LEN / 2));**

**memcpy(ApiMac\_extAddr + (APIMAC\_SADDR\_EXT\_LEN / 2), (uint8\_t \*)&(\_\_ccfg.CCFG\_IEEE\_MAC\_1),**

**(APIMAC\_SADDR\_EXT\_LEN / 2));**

**/\* Check to see if the CCFG IEEE is valid \*/**

**if(memcmp(ApiMac\_extAddr, dummyExtAddr, APIMAC\_SADDR\_EXT\_LEN) == 0)**

**{**

**/\* No, it isn't valid. Get the Primary IEEE Address \*/**

**memcpy(ApiMac\_extAddr, (uint8\_t \*)(FCFG1\_BASE + EXTADDR\_OFFSET),**

**(APIMAC\_SADDR\_EXT\_LEN));**

**}**

**#ifdef NV\_RESTORE**

**/\* Setup the NV driver \*/**

**#ifdef ONE\_PAGE\_NV**

**NVOCOP\_loadApiPtrs(&Main\_user1Cfg.nvFps);**

**#else**

**NVOCTP\_loadApiPtrs(&Main\_user1Cfg.nvFps);**

**#endif**

**if(Main\_user1Cfg.nvFps.initNV)**

**{**

**Main\_user1Cfg.nvFps.initNV( NULL);**

**}**

**#endif**

**#ifdef OSAL\_PORT2TIRTOS**

**/\* Initialize the application \*/**

**Collector\_init(\_macTaskId);**

**#else**

**ICall\_createRemoteTasks();**

**/\* Initialize the application \*/**

**Collector\_init();**

**#endif**

**/\* Kick off application - Forever loop \*/**

**while(1)**

**{**

**Collector\_process();**

**}**

**}**

**/\*!**

**\* @brief TIRTOS HWI Handler. The name of this function is set to**

**\* M3Hwi.excHandlerFunc in app.cfg, you can disable this by**

**\* setting it to null.**

**\***

**\* @param excStack - TIROS variable**

**\* @param lr - TIROS variable**

**\*/**

**xdc\_Void Main\_excHandler(UInt \*excStack, UInt lr)**

**{**

**/\* User defined function \*/**

**Main\_assertHandler(MAIN\_ASSERT\_HWI\_TIRTOS);**

**}**

**/\*!**

**\* @brief HAL assert handler required by OSAL memory module.**

**\*/**

**void halAssertHandler(void)**

**{**

**/\* User defined function \*/**

**Main\_assertHandler(MAIN\_ASSERT\_ICALL);**

**}**

**/\*!**

**\* @brief MAC HAL assert handler.**

**\*/**

**void macHalAssertHandler(void)**

**{**

**/\* User defined function \*/**

**Main\_assertHandler(MAIN\_ASSERT\_MAC);**

**}**

**/\*!**

**\* @brief "main()" function - starting point**

**\*/**

**int main(void)**

**{**

**Task\_Params taskParams;**

**#ifndef USE\_DEFAULT\_USER\_CFG**

**macUser0Cfg[0].pAssertFP = macHalAssertHandler;**

**#endif**

**#if ((CONFIG\_RANGE\_EXT\_MODE == APIMAC\_HIGH\_GAIN\_MODE) && \**

**defined(DeviceFamily\_CC13X0) && !defined(FREQ\_2\_4G))**

**macUser0Cfg[0].pSetRE = Board\_Palna\_initialize;**

**#endif**

**/\***

**Initialization for board related stuff such as LEDs**

**following TI-RTOS convention**

**\*/**

**PIN\_init(BoardGpioInitTable);**

**#if defined(FEATURE\_OAD)**

**SPI\_init();**

**#endif**

**#if defined(BOARD\_DISPLAY\_USE\_UART)**

**/\* Enable System\_printf(..) UART output \*/**

**UART\_init();**

**UART\_Params\_init(&uartParams);**

**#ifndef TIMAC\_AGAMA\_FPGA**

**uartParams.baudRate = 115200;**

**#else**

**uartParams.baudRate = 460800;**

**#endif**

**UartPrintf\_init(UART\_open(Board\_UART0, &uartParams));**

**#endif /\* BOARD\_DISPLAY\_USE\_UART \*/**

**#ifdef OSAL\_PORT2TIRTOS**

**\_macTaskId = macTaskInit(macUser0Cfg);**

**#endif**

**/\* Configure task. \*/**

**Task\_Params\_init(&taskParams);**

**taskParams.stack = appTaskStack;**

**taskParams.stackSize = APP\_TASK\_STACK\_SIZE;**

**taskParams.priority = APP\_TASK\_PRIORITY;**

**Task\_construct(&appTask, appTaskFxn, &taskParams, NULL);**

**#ifdef DEBUG\_SW\_TRACE**

**IOCPortConfigureSet(IOID\_8, IOC\_PORT\_RFC\_TRC, IOC\_STD\_OUTPUT**

**| IOC\_CURRENT\_4MA | IOC\_SLEW\_ENABLE);**

**#endif /\* DEBUG\_SW\_TRACE \*/**

**BIOS\_start(); /\* enable interrupts and start SYS/BIOS \*/**

**return (0);**

**}**