Code

Node.c – code for node's microcontroller

```
#include <stdint.h>
#include <stdbool.h>
#include <avr/io.h>
#include <util/delay.h>
#include <avr/eeprom.h>

#include "zigbeemega1281/config_uart_extended.h" // See this file for all project options.
#include "zigbeemega1281/compiler.h"
#include "zigbeemega1281/at86rf230_registermap.h"

#include "zigbeemega1281/hal_avr_mega1281.h"
#include "zigbeemega1281/hal_avr_mega1281.c"

#include "zigbeemega1281/tat.h"
#include "zigbeemega1281/tat.c"

#include "mac_node.h"
#include "mac_node.c"

#include "usbincludebasestationv2.c"

#include "LCD.h"
#include "SampleFont8.h"
#include "LCDfunctions.c"

/*============================ INCLDUE
=========================================*/
// See above (ADC 2/24/2008)
/*============================ MACROS
==========================================*/
#define RX_POOL_SIZE       ( 4 ) //MUST BE GREATER THAN ZERO.
#define OPERATING_CHANNEL ( 17 ) // !< Channel to transmit on. Must be between
11 and 26!
/*============================ TYPEDEFS
=======================================*/

/*============================ VARIABLES
=======================================*/
static uint8_t tx_frame[ 127 ]; //!< Buffer used to build TX frames. (Size must be max
PSDU length.)
static uint8_t cmd_frame[127 ];
static hal_rx_frame_t rx_pool[ RX_POOL_SIZE ]; //!< Pool of hal_rx_frame_t's.
static hal_rx_frame_t *rx_pool_start; //!< Pointer to start of pool.
static hal_rx_frame_t *rx_pool_end; //!< Pointer to end of pool.
static hal_rx_frame_t *rx_pool_head; //!< Pointer to next hal_rx_frame_t it is possible to
write.
static hal_rx_frame_t *rx_pool_tail; //!< Pointer to next hal_rx_frame_t that can be read
from the pool.
static uint8_t rx_pool_items_free; //!< Number of free items (hal_rx_frame_t) in the pool.
static uint8_t rx_pool_items_used; // !< Number of used items.
static bool rx_pool_overflow_flag; //!< Flag that is used to signal a pool overflow.

static bool rx_flag; //!< Flag used to mask between the two possible TRX_END events.

/*============================ PROTOTYPES
=======================================*/
static bool trx_init( void );
static void trx_end_handler( uint32_t time_stamp );
static void rx_pool_init( void );

/*! \brief This function is used to initialize the TRX.
 *
 * The TAT will be set up to run on the chosen operating channel, with CLKM diabled,
 * and then configure the RX_AACK and TX_ARET modes.
 *
 *  \retval true if the TRX was successfully configured.
 *  \retval false if the TRX was not configured properly.
 */
static bool trx_init( void ){

   static bool status;

   if (tat_init( ) != TAT_SUCCESS) {
      status = false;
   } else if (tat_set_operating_channel( OPERATING_CHANNEL ) != TAT_SUCCESS)
{
      status = false;
   } else if (tat_set_clock_speed( true, CLKM_DISABLED ) != TAT_SUCCESS) {
      status = false;
   } else{

      /*Set up the extended modes:*/
      //RX_AACK:
      tat_set_short_address( SHORT_ADDRESS ); //Short Address.
                  //tat_set_extended_address( (uint8_t *)&my_addr->long_addr[0] );
      tat_set_pan_id( PAN_ID ); //PAN ID.
      tat_set_device_role( false ); // No Coordintor support is necessary.
```

```
        //TX_ARET:
        tat_configure_csma( 234, 0xE2 ); // Default CSMA_SEED_0, MIN_BE = 3,
MAX_CSMA_RETRIES = , and CSMA_SEED_1 =

        //Both Modes:
        tat_use_auto_tx_crc( true ); //Automatic CRC must be enabled.
        hal_set_trx_end_event_handler( trx_end_handler ); // Event handler for TRX_END
events.
                    hal_enable_trx_interrupt( );    //Enable interrupts from the radio
transceiver.

        status = true;
    } // end: if (tat_init( ) != TAT_SUCCESS) ...

    return status;
}

/*! \brief This function initialize the rx_pool. The rx_pool is in essence a FIFO.
 */
static void rx_pool_init( void ){

    rx_pool_start = rx_pool;
    rx_pool_end = &rx_pool[ RX_POOL_SIZE - 1 ];

    rx_pool_head = rx_pool_start;
    rx_pool_tail = rx_pool_end;

    rx_pool_items_free = RX_POOL_SIZE;
    rx_pool_items_used = 0;

    rx_pool_overflow_flag = false;
}

/*! \brief This function is the TRX_END event handler that is called from the
 *        TRX isr if assigned.
 *
 *  \param[in] time_stamp Interrupt timestamp in IEEE 802.15.4 symbols.
 */
static void trx_end_handler( uint32_t time_stamp ){

    if (rx_flag == true) {

        //Check if these is space left in the rx_pool.
        if (rx_pool_items_free == 0) {
            rx_pool_overflow_flag = true;
```

```
      } else {

         //Space left, so upload the received frame.
         hal_frame_read( rx_pool_head );

         //Then check the CRC. Will not store frames with invalid CRC.
         if (rx_pool_head->crc == true) {

            //Handle wrapping of rx_pool.
            if (rx_pool_head == rx_pool_end) {
               rx_pool_head = rx_pool_start;
            } else {
               ++rx_pool_head;
            } // end: if (rx_pool_head == rx_pool_end) ...

            --rx_pool_items_free;
            ++rx_pool_items_used;
         } // end: if (rx_pool_head->crc == true) ...
      } // end: if (rx_pool_items_free == 0) ...
   } // end:  if (rx_flag == true) ...
}

void main( void ){
         /* Turn USB chip off */
         initialize_MAX();

         //unsigned int county = 0;
         char ans = 'j';
   static uint8_t length_of_rcv_data = 0;
   static uint8_t frame_sequence_number = 0;
         uint8_t tx_length;
         uint8_t cmd_length;
         uint8_t frame_type;
         uint8_t payload_index;
         unsigned int i;
         char display_buf[30];
   rx_flag = true;

         // Initialization functions
   rx_pool_init( );
   trx_init( );
         initialize( );
         ScanKeyMatrixInit();
         mac_init();

         // Initialize Address Entry for Node
```

```
        address_entry my_addr;
        my_addr.pan_id = 0x00;
        my_addr.coord_addr = 0x00;
        my_addr.long_addr[7] = 0x23;
        my_addr.long_addr[6] = 0x87;
        my_addr.long_addr[5] = 0x12;
        my_addr.long_addr[4] = 0xAB;
        my_addr.long_addr[3] = 0xD8;
        my_addr.long_addr[2] = 0xC4;
        my_addr.long_addr[1] = 0x89;
        my_addr.long_addr[0] = 0x1B;
        my_addr.associated = 0;

        uint16_t shortaddr;
        shortaddr = 0x8723;
        tat_set_short_address( shortaddr ); //Short Address.
        tat_set_extended_address((uint8_t *)&my_addr.long_addr[0]);

        uint8_t ext_addr[8];
        char ext_addr_char[32];
        tat_get_extended_address((uint8_t *)&ext_addr);

        itoa(ext_addr[4], ext_addr_char, 2);
        String_T addr_disp= {4,L00J,0,ext_addr_char};
        //write_text(&addr_disp);
        String_T answer = {4,L00J,0, "Your Answer:"};
        write_text(&answer);
        // mac_build_data_frame(tx_frame,&my_addr);

/*      uint8_t ext_add1;uint8_t ext_add2;
        uint8_t ext_add3;uint8_t ext_add4;

        ext_add1 = eeprom_read_byte( (uint8_t*)1);
        ext_add2 = eeprom_read_byte( (uint8_t*)2);
        ext_add3 = eeprom_read_byte( (uint8_t*)3);
        ext_add4 = eeprom_read_byte( (uint8_t*)4);*/

   //Set system state to RX_AACK_ON
   if (tat_set_trx_state( RX_AACK_ON ) == TAT_SUCCESS) {
   }
        sei( ); // Turn interupts on

   /*Enter Normal Program Flow:
     - Check for newly received frames.
     - Notify on rx_pool overflow.
     - Try to send data on air interface, if something is received on UART/USB.
```

```
      - Notify if the typed message was too long.
    */

         String_T rcv_pkt   = {7, L00J, 0, "Received Packet"};
         String_T err_tx    = {7, L00J, 0, "Error: Packet not TX'ed"};
         String_T rx_beacon = {7, L00J, 0, "Received Beacon"};
         String_T cmd       = {7, L00J, 0, "Received Command Frame"};
         String_T data_fr   = {7, L00J, 0, "Received Data Frame"};
         String_T testing   = {7, L00J, 0, "Testing LCD"};
         String_T assoc     = {6, L00J, 0, "Association Successful"};
         write_text(&testing);

    while (true) {
                //county++;

      //Check if we have received something on the air interface.
      if (rx_pool_items_used != 0) {

         //Handle wrapping of rx_pool.
         if (rx_pool_tail == rx_pool_end) {
            rx_pool_tail = rx_pool_start;
         } else {
            ++rx_pool_tail;
         }

         //Turn interrupts off for a short while to protect when status
         //information about the rx_pool is updated.
         cli( );
         ++rx_pool_items_free;
         --rx_pool_items_used;
         sei( );

                        page_clear(7); write_text(&rcv_pkt);// Tell us we received a pkt
                        length_of_rcv_data = (rx_pool_tail->length);

                        frame_type = ( (rx_pool_tail->data[0]) & 0x07 );

                        if (frame_type == 0x00) {      // Frame type = beacon
                                page_clear(7); write_text(&rx_beacon);
                                my_addr.pan_id = ( (rx_pool_tail->data[3] & 0xFFFF) |
((rx_pool_tail->data[4] & 0xFFFF) << 8) );
                                // tat_set_pan_id( my_addr.pan_id);
                                my_addr.coord_addr = ( (rx_pool_tail->data[5] & 0xFFFF)
| ((rx_pool_tail->data[6] & 0xFFFF) << 8) );
```

```
                                if ( ((rx_pool_tail->data[8] & 0xC0) == 0xC0) &&
(my_addr.associated == 0)) {
                                        // Try to associate with PAN coord
                                        cmd_length =
mac_build_command_frame(cmd_frame, &my_addr);
                                        cmd_frame[cmd_length++] = 0x01;  // Assiciation
Request
                                        // Capability Info Field
                                        // bits: 00010001 (0-7)
                                        cmd_frame[cmd_length++] = 0x88;  //10001000

                                        cmd_length += 2;

                                        if (tat_set_trx_state( TX_ARET_ON ) ==
TAT_SUCCESS){
                                                cmd_frame[ 2 ] = macFSN++; //Sequence
Number.

                        rx_flag = false; // Set the flag false, so that the TRX_END event is
not misinterpreted.

                        if (tat_send_data_with_retry( cmd_length, cmd_frame, 1 ) ==
TAT_SUCCESS) {

                        } else {
                                                        page_clear(7); write_text(&err_tx);
                        }
                                        }// End Set TX state to send frame
                                }
                        }// End if received a beacon

                        else if (frame_type == 0x03) { // If a command frame is sent
                                payload_index = 15;
                                page_clear(7); write_text(&cmd);
                                if (rx_pool_tail->data[payload_index++] == 0x02) {
                                        payload_index++;
                                        my_addr.short_addr = ( (rx_pool_tail-
>data[payload_index-1] & 0xFFFF) | ((rx_pool_tail->data[payload_index] & 0xFFFF)
<< 8) );
                                        payload_index++;
                                        if (rx_pool_tail->data[payload_index++] == 0x00)
{ // Association successful
                                                my_addr.associated = true;
                                        }
                                page_clear(6); write_text(&assoc);
                                }
                        }// End if Receive a command

                        else if (frame_type == 0x01) {// if Receive a data frame
                                page_clear(7); write_text(&data_fr);
                                if (rx_pool_tail->data[9] == 0x01) {
                                        for (i=0;i<rx_pool_tail->data[10];i++) {
                                                display_buf[i] = rx_pool_tail->data[i+11];
                                        }
                                        String_T msg = {0, L00J, 0, display_buf};
                                        page_clear(0);page_clear(1);page_clear(2);
write_text(&msg);
                                }
                                else if (rx_pool_tail->data[9] == 0x02) {
                                        for (i=0;i<rx_pool_tail->data[10];i++) {
                                                display_buf[i] = rx_pool_tail->data[i+11];
                                        }
                                        String_T msg = {3, L00J, 0, display_buf};
                                        page_clear(3); write_text(&msg);
                                }
                        }// End receive a data frame

                        if (tat_set_trx_state( RX_AACK_ON ) == TAT_SUCCESS){ }
                        rx_flag = true; // Set the flag back again. Only used to protec the
frame transmission.
        } // end: check for packet on air interface ...

        //Check for rx_pool overflow.
        if (rx_pool_overflow_flag == true) {
           cli();
           rx_pool_init( );
                        length_of_rcv_data = 0;
           sei();
        }

        //Check if data is ready to be sent.
                String_T num = { 4, ABSJ, 64, ScanKeyMatrix() };
                if (num.text[0] == 'y') ans = 'j';
        if (num.text[0] != 'y') && num.text[0] != ans ){
                        write_text(&num);
                        ans = num.text[0];
                        tx_length = mac_build_data_frame(tx_frame,&my_addr);

                        tx_frame[ tx_length++ ] = num.text[0];
                        tx_frame[ tx_length++ ] = 0x00;
                        tx_frame[ tx_length++ ] = 0x00;
                        tx_frame[ tx_length++ ] = 0x00;
                        tx_frame[ tx_length++ ] = 0x00;

            //Change state to TX_ARET_ON and send data if the state transition was
successful.
            if (tat_set_trx_state( TX_ARET_ON ) == TAT_SUCCESS) {

               tx_frame[ 2 ] = frame_sequence_number++; //Sequence Number.

               rx_flag = false; // Set the flag false, so that the TRX_END event is not
misinterpreted.

               if (tat_send_data_with_retry( tx_length, tx_frame, 1 ) == TAT_SUCCESS) {

               } else {
                                    page_clear(0);
                                    write_text(&err_tx);
               }
            } else {

            }
            if (tat_set_trx_state( RX_AACK_ON ) != TAT_SUCCESS) { }
            rx_flag = true; // Set the flag back again. Only used to protec the frame
transmission.
        } // end:
    } // emd: while (true) ...
}
/*EOF*/
```

Basestation.c – code for base station's microcontroller

```
#include <stdint.h>
#include <stdbool.h>
#include <avr/io.h>
#include <util/delay.h>

#include "zigbeemega1281/config_uart_extended.h" // See this file for all project options.
#include "zigbeemega1281/compiler.h"
#include "zigbeemega1281/at86rf230_registermap.h"

#include "zigbeemega1281/hal_avr_mega1281.h"
#include "zigbeemega1281/hal_avr_mega1281.c"
#include "zigbeemega1281/tat.h"
#include "zigbeemega1281/tat.c"
#include "mac.h"
#include "mac.c"
```

```
#include "usbIncludeBasestationV2.c"

#include "LCD.h"
#include "SampleFont8.h"
#include "LCDfunctions.c"

/*============================== INCLDUE
========================================*/
// See above (ADC 2/24/2008)
/*============================== MACROS
=========================================*/
#define OPERATING_CHANNEL ( 17 ) // !< Channel to transmit on. Must be between
11 and 26!
#define PAN_ID          ( 0xBEEF ) //!< System PAN ID.

#define MAX_ASSOCIATED_NODES ( 100 )
#define START_ADDRESS  ( 0xAA01 )

/*============================== TYPEDEFS
=======================================*/

/*============================== VARIABLES
======================================*/
static uint8_t tx_frame[ 127 ]; //!< Buffer used to build TX frames. (Size must be max
PSDU length.)
static uint8_t beacon_frame[ 127 ];
static uint8_t cmd_frame[ 127 ];

static hal_rx_frame_t rx_pool[ RX_POOL_SIZE ]; //!< Pool of hal_rx_frame_t's.
static hal_rx_frame_t *rx_pool_start; //!< Pointer to start of pool.
static hal_rx_frame_t *rx_pool_end; //!< Pointer to end of pool.
static hal_rx_frame_t *rx_pool_head; //!< Pointer to next hal_rx_frame_t it is possible to
write.
static hal_rx_frame_t *rx_pool_tail; //!< Pointer to next hal_rx_frame_t that can be read
from the pool.
static uint8_t rx_pool_items_free; //!< Number of free items (hal_rx_frame_t) in the pool.
static uint8_t rx_pool_items_used; // !< Number of used items.
static bool rx_pool_overflow_flag; //!< Flag that is used to signal a pool overflow.

static bool rx_flag; //!< Flag used to mask between the two possible TRX_END events.

static uint8_t num_assoc_nodes;

uint8_t *rssi_lvl_ptr;
```

```
/*============================ PROTOTYPES
=====================================*/
static bool trx_init( void );
static void trx_end_handler( uint32_t time_stamp );
static void rx_pool_init( void );


/*! \brief This function is used to initialize the TRX.
 *
 * The TAT will be set up to run on the chosen operating channel, with CLKM diabled,
 * and then configure the RX_AACK and TX_ARET modes.
 *
 *  \retval true if the TRX was successfully configured.
 *  \retval false if the TRX was not configured properly.
 */
static bool trx_init( void ){

   static bool status;

   if (tat_init( ) != TAT_SUCCESS) {
      status = false;
   } else if (tat_set_operating_channel( OPERATING_CHANNEL ) != TAT_SUCCESS)
{
      status = false;
   } else if (tat_set_clock_speed( true, CLKM_DISABLED ) != TAT_SUCCESS) {
      status = false;
   } else{

      /*Set up the extended modes:*/
      //RX_AACK:
      tat_set_short_address( SHORT_ADDRESS ); //Short Address.
      tat_set_pan_id( PAN_ID ); //PAN ID.
      tat_set_device_role( true ); // Coordinator

      //TX_ARET:
      tat_configure_csma( 234, 0xE2 ); // Default CSMA_SEED_0, MIN_BE = 3,
MAX_CSMA_RETRIES = , and CSMA_SEED_1 =

      //Both Modes:
      tat_use_auto_tx_crc( true ); //Automatic CRC must be enabled.
      hal_set_trx_end_event_handler( trx_end_handler ); // Event handler for TRX_END
events.
                  hal_enable_trx_interrupt( );    //Enable interrupts from the radio
transceiver.

      status = true;
```

```
    } // end: if (tat_init( ) != TAT_SUCCESS) ...

    return status;
}

/*! \brief This function initialize the rx_pool. The rx_pool is in essence a FIFO.
 */
static void rx_pool_init( void ){

    rx_pool_start = rx_pool;
    rx_pool_end = &rx_pool[ RX_POOL_SIZE - 1 ];

    rx_pool_head = rx_pool_start;
    rx_pool_tail = rx_pool_end;

    rx_pool_items_free = RX_POOL_SIZE;
    rx_pool_items_used = 0;

    rx_pool_overflow_flag = false;
}

/*! \brief This function is the TRX_END event handler that is called from the
 *         TRX isr if assigned.
 *
 *  \param[in] time_stamp Interrupt timestamp in IEEE 802.15.4 symbols.
 */
static void trx_end_handler( uint32_t time_stamp ){

    if (rx_flag == true) {

        //Check if these is space left in the rx_pool.
        if (rx_pool_items_free == 0) {
            rx_pool_overflow_flag = true;
        } else {

            //Space left, so upload the received frame.
            hal_frame_read( rx_pool_head );

            //Then check the CRC. Will not store frames with invalid CRC.
            if (rx_pool_head->crc == true) {

                //Handle wrapping of rx_pool.
                if (rx_pool_head == rx_pool_end) {
                    rx_pool_head = rx_pool_start;
                } else {
                    ++rx_pool_head;
            } // end: if (rx_pool_head == rx_pool_end) ...

            --rx_pool_items_free;
            ++rx_pool_items_used;
         } // end: if (rx_pool_head->crc == true) ...
      } // end: if (rx_pool_items_free == 0) ...
   } // end:  if (rx_flag == true) ...
}

int main( void ){

        //DDRB |=  0b00010000;
        //PORTB |= 0b00010000;

   static uint8_t length_of_received_data = 0;
        uint8_t tx_length;
        uint8_t beacon_len;
        uint8_t cmd_length;
        uint16_t count = 0;
        num_assoc_nodes = 0;
        static assoc_node_t assoc_pool[ MAX_ASSOCIATED_NODES ];

        uint16_t next_available_short_addr;
        uint16_t new_addr;

        unsigned int i;
        unsigned char flag;
        //char rssi_char[16];
   rx_flag = true;

        String_T rx_pkt        = {7, L00J, 0, "Received Packet"          };
        String_T rx_usb        = {7, L00J, 0, "Received USB"             };
        String_T tx_mode       = {7, L00J, 0, "Entering TX mode"         };
        String_T rx_mode       = {7, L00J, 0, "Entering RX mode"         };
        String_T send_beacon   = {7, L00J, 0, "Sending Beacon"           };
        String_T assn_rqst     = {7, L00J, 0, "Received Assn. Request"   };
        String_T err_tx        = {7, L00J, 0, "Error: Packet not TX'ed"  };
        String_T data_pkt      = {7, L00J, 0, "Received a Data Packet"   };
        String_T data_frm_node = {7, L00J, 0, "Received from joined node"};

   /*Pre Build Header of IEEE 802.15.4 Data frame.*/
   /*tx_frame[ 0 ] = 0x61; //FCF.
   tx_frame[ 1 ] = 0x88; //FCF.
                  //Sequence number set during frame transmission.
   tx_frame[ 3 ] = PAN_ID & 0xFF; //Dest. PANID.
   tx_frame[ 4 ] = (PAN_ID >> 8 ) & 0xFF; //Dest. PANID.
```

```
tx_frame[ 5 ] = DEST_ADDRESS & 0xFF; //Dest. Addr.
tx_frame[ 6 ] = (DEST_ADDRESS >> 8 ) & 0xFF; //Dest. Addr.
tx_frame[ 7 ] = SHORT_ADDRESS & 0xFF; //Source Addr.
tx_frame[ 8 ] = (SHORT_ADDRESS >> 8 ) & 0xFF; //Source Addr.
*/
rx_pool_init( );
      initialize_MAX ();
trx_init( );
      initialize( );
      mac_init( );

      // set up attributes
      address_entry self_addr;
      self_addr.pan_id = tat_get_pan_id();
      self_addr.short_addr = 0xACDC;//tat_get_short_address();
      self_addr.long_addr[0] = 0x99;
      self_addr.long_addr[1] = 0x88;
      self_addr.long_addr[2] = 0x77;
      self_addr.long_addr[3] = 0x66;
      self_addr.long_addr[4] = 0x55;
      self_addr.long_addr[5] = 0x44;
      self_addr.long_addr[6] = 0x33;
      self_addr.long_addr[7] = 0x22;
      next_available_short_addr = START_ADDRESS;

      tat_set_extended_address( &(self_addr.long_addr[0]) );

//Set system state to RX_AACK_ON
if (tat_set_trx_state( RX_AACK_ON ) == TAT_SUCCESS){
               write_text(&rx_mode);
      }

      sei( );

/*Enter Normal Program Flow:
   - Check for newly received frames. Output to LCD and send to USB.
   - Notify on rx_pool overflow.
   - Try to send data on air interface, if something is received on USB.
   - Notify if the typed message was too long.
 */
while (true) {

               // If host suspends, sleep
               if (Suspended) {
                        check_for_resume ();
               }
```

```
                // Handles interupts from USB chip
                if (MAX_Int_Pending()) {
                        service_irqs ();
                }

        //Check if we have received something on the air interface.
        if (rx_pool_items_used != 0) {

           //Handle wrapping of rx_pool.
           if (rx_pool_tail == rx_pool_end) {
              rx_pool_tail = rx_pool_start;
           } else {
              ++rx_pool_tail;
           }

           //Turn interrupts off for a short while to protect when status
           //information about the rx_pool is updated.
           cli( );
           ++rx_pool_items_free;
           --rx_pool_items_used;
           sei( );

                        length_of_received_data = (rx_pool_tail->length);
                        page_clear(7); write_text(&rx_pkt);

                        // If an association request is received
                        if ((rx_pool_tail->data[0] & 0x07) == 0x03) {
                                page_clear(7); write_text(&assn_rqst);
                                num_assoc_nodes++;
                                new_addr = next_available_short_addr++;
                                assoc_pool[new_addr-START_ADDRESS].short_addr =
new_addr;
                                assoc_pool[new_addr-START_ADDRESS].long_addr[0] =
(rx_pool_tail->data[9 ] & 0xFF);
                                assoc_pool[new_addr-START_ADDRESS].long_addr[1] =
(rx_pool_tail->data[10] & 0xFF);
                                assoc_pool[new_addr-START_ADDRESS].long_addr[2] =
(rx_pool_tail->data[11] & 0xFF);
                                assoc_pool[new_addr-START_ADDRESS].long_addr[3] =
(rx_pool_tail->data[12] & 0xFF);
                                assoc_pool[new_addr-START_ADDRESS].long_addr[4] =
(rx_pool_tail->data[13] & 0xFF);
                                assoc_pool[new_addr-START_ADDRESS].long_addr[5] =
(rx_pool_tail->data[14] & 0xFF);
                                assoc_pool[new_addr-START_ADDRESS].long_addr[6] =
(rx_pool_tail->data[15] & 0xFF);
```

```
                                        assoc_pool[new_addr-START_ADDRESS].long_addr[7] =
(rx_pool_tail->data[16] & 0xFF);

                                        cmd_length = mac_build_command_frame(cmd_frame,
&self_addr, (uint8_t *) &(assoc_pool[new_addr-START_ADDRESS].long_addr) );
                                        cmd_frame[cmd_length++] = 0x02;  // Association
Response
                                                  // Short Address
                                        cmd_frame[cmd_length++] = new_addr & 0xFF;
                                        cmd_frame[cmd_length++] = (new_addr >> 8 ) & 0xFF;
                                                  // Association Status
                                        cmd_frame[cmd_length++] = 0x00; // Assn. Successful

                                        cmd_length += 2;

                                        if (tat_set_trx_state( TX_ARET_ON ) ==
TAT_SUCCESS){
                                                  cmd_frame[ 2 ] = macFSN++; //Sequence Number.

                    rx_flag = false; // Set the flag false, so that the TRX_END event is not
misinterpreted.

                    if (tat_send_data_with_retry( cmd_length, cmd_frame, 1 ) ==
TAT_SUCCESS) {
                                                            TO_COMPUTER[ 0 ] = 0x00;          //
Registered msg type
                              TO_COMPUTER[ 1 ] = assoc_pool[new_addr-
START_ADDRESS].long_addr[0];
                                                            TO_COMPUTER[ 2 ] =
assoc_pool[new_addr-START_ADDRESS].long_addr[1];
                                                            TO_COMPUTER[ 3 ] =
assoc_pool[new_addr-START_ADDRESS].long_addr[2];
                                                            TO_COMPUTER[ 4 ] =
assoc_pool[new_addr-START_ADDRESS].long_addr[3];
                                                            TO_COMPUTER[ 5 ] =
assoc_pool[new_addr-START_ADDRESS].long_addr[4];
                                                            TO_COMPUTER[ 6 ] =
assoc_pool[new_addr-START_ADDRESS].long_addr[5];
                                                            TO_COMPUTER[ 7 ] =
assoc_pool[new_addr-START_ADDRESS].long_addr[6];
                                                            TO_COMPUTER[ 8 ] =
assoc_pool[new_addr-START_ADDRESS].long_addr[7];
                                                            TO_COMPUTER[ 9 ] = new_addr & 0xFF;
                                                            TO_COMPUTER[ 10] = (new_addr >> 8 )
& 0xFF;
```

```
                                        flag =
device_to_computer(TO_COMPUTER, 11);

                } else {
                                        page_clear(7);
                                        write_text(&err_tx);
                }
                                }

                        }
                        // If we receive a data packet
                        if ((rx_pool_tail->data[0] & 0x07) == 0x01) {
                                page_clear(7);
                                write_text(&data_pkt);
                                uint8_t node_is_assoc = 0;
                                uint16_t rcv_addr;
                                rcv_addr = ( (rx_pool_tail->data[7] && 0xFFFF) |
((rx_pool_tail->data[8] && 0xFFFF) << 8) );
                                for (i=0;i<num_assoc_nodes;i++) { // Check
                                        if (assoc_pool[i].short_addr == rcv_addr){
                                                node_is_assoc = 1;
                                        }
                                }
                                if (node_is_assoc){ }
                                page_clear(7); write_text(&data_frm_node);

                                TO_COMPUTER[0] = rx_pool_tail->data[9];
                                TO_COMPUTER[1] = '\0';

                                String_T line7 = {7, L00J, 0, (char *)TO_COMPUTER};
                                write_text(&line7);

                                TO_COMPUTER[0] = 0x01; //Set to answer message type
                                TO_COMPUTER[1] = rx_pool_tail->data[7];
                                TO_COMPUTER[2] = rx_pool_tail->data[8];
                                TO_COMPUTER[3] = rx_pool_tail->data[9];
                                flag = device_to_computer(TO_COMPUTER, 4);
                        }
                        if (tat_set_trx_state( RX_AACK_ON ) == TAT_SUCCESS){
                                //String_T state = {4,L00J,0,"State is RX_AACK_ON"};
                                //write_text(&state);
                        }
                        rx_flag = true;
        } // end: if (rx_pool_items_used != 0) ...

        //Check for rx_pool overflow.
        if (rx_pool_overflow_flag == true) {
           cli();
           rx_pool_init( );
                        length_of_received_data = 0;
           sei();
        }

        //Check for new data on the serial interface.
        //Check if data is ready to be sent.
        if (OUT1_NEWDATA){                    // Check USB
                        write_text(&rx_usb);

                        OUT1_NEWDATA = 0;
                        flag = device_to_computer(OUT1_BUFFER, 1);

                        tx_length = mac_build_data_frame(tx_frame, &self_addr,
0xFFFF);

                        // tx_length += OUT1_BUFFER[0];

                        for (i=0; i<OUT1_BUFFER[0];i++) {
                                TO_COMPUTER[i] = OUT1_BUFFER[i+1];
                                tx_frame [tx_length++] = OUT1_BUFFER[i+1];
                        }
                        flag = device_to_computer(TO_COMPUTER,
OUT1_BUFFER[0]);

                        tx_frame[tx_length++] = 0x00;
                        tx_frame[tx_length++] = 0x00;


                        String_T line2 = {4, L00J, 0, (char *)(OUT1_BUFFER+1)};
                        write_text(&line2);

           //Change state to TX_ARET_ON and send data if the state transition was
successful.
           if (tat_set_trx_state( TX_ARET_ON ) == TAT_SUCCESS) {

                                write_text(&tx_mode);

              tx_frame[ 2 ] = macFSN++; //Sequence Number.

              rx_flag = false; // Set the flag false, so that the TRX_END event is not
misinterpreted.

              if (tat_send_data_with_retry( tx_length, tx_frame, 1 ) == TAT_SUCCESS) {
```

```
                                        // Successful transmission
                                }
        }

        if (tat_set_trx_state( RX_AACK_ON ) != TAT_SUCCESS) { }

        rx_flag = true; // Set the flag back again. Only used to protect the frame
transmission.

      } // end "Check USB"

                if (count == 65000) {
                        write_text(&send_beacon);
                        page_clear(3);
                        count = 0;
                        beacon_len = mac_build_beacon_frame(beacon_frame,
&self_addr);
                        beacon_frame[ 2 ] = macBSN++;
                        if (tat_set_trx_state( TX_ARET_ON ) == TAT_SUCCESS) {
                                if (tat_send_data_with_retry( beacon_len, beacon_frame, 1
) == TAT_SUCCESS) {}
                        }
                        if (tat_set_trx_state( RX_AACK_ON ) == TAT_SUCCESS) {
                                write_text(&rx_mode);
                        }
                }
                count++;
   } // emd: while (true)
}
/*EOF*/
```

Mac.c – Zigbee mac implementation

```
// MAC Implementation

#include "mac.h"

// Functions

void mac_init (void) {
        macBSN = 0;             // Initialize the beacon sequence
        macFSN = 0;             // Initialize the frame sequence

};

uint8_t mac_build_beacon_frame (uint8_t *tx_frame, address_entry *curr_addr) {
        uint8_t tx_len;
        //Frame format

        //FCF
        //bits 0-2: 000 - beacon frame
        //     3  : 0   - security disabled
        //     4  : 1   - frame pending (broadcast frame)
        //     5  : 1   - ACK requested
        //     6  : 0   - unused/ignored (PAN ID compression)
        //    7-9 : 000 - reserved
        //   10-11: 00  - unused/ignored
        //   12-13: 00  - only set to 'one' if security enabled
        //   14-15: 10  - src addr is 16-bit short addr
        tx_frame[ 0 ] = 0x30; // 00110000 - FCF
        tx_frame[ 1 ] = 0x80; // 01000000 - FCF
        // tx_frame[2] set when frame is sent
        tx_frame[ 3 ] = curr_addr->pan_id & 0xFF;                    //Source PAN
ID
   tx_frame[ 4 ] = (curr_addr->pan_id >> 8 ) & 0xFF;            //Source PAN ID
        tx_frame[ 5 ] = curr_addr->short_addr & 0xFF;                //Source Addr.
   tx_frame[ 6 ] = (curr_addr->short_addr >> 8 ) & 0xFF;   //Source Addr.

        // SuperFrame Specification
        //bits 3-0: 1000- Beacon Order
        //     7-4: 1000- SuperFrame Order
        //    11-8: 1000- Final CAP Slot
        //     12 : 0   - BLE
        //     13 : 0   - Reserved
        //     14 : 1   - PAN Coordinator
        //     15 : 1   - Assn Permit
        tx_frame[ 7 ] = 0x88; //10001000
        tx_frame[ 8 ] = 0xC8; //11001000

        // GTX Spec Field
        //bits 0-2: 000 - GTS Descriptor Count
        //     3-6: 0000- Reserved
        //      7 : 0   - GTS permit
        tx_frame[ 9 ] = 0x00; //00000000 - No GTS

        // Pending Address
        // bits 0-2: 000 - # short addr pending
        //       3 : 0   - Reserved
        //      4-6: 000 - # long addr pending
        //       7 : 0   - Reserved
        tx_frame[10 ] = 0x00; //00000000 - No Pending Addresses
        // Beacon Payload
        tx_frame[11 ] = 0x00;
        tx_frame[12 ] = 0x00;

        tx_len = 13;
        return tx_len;
};

uint8_t mac_build_data_frame (uint8_t *tx_frame, address_entry *curr_addr, uint16_t
dest){
        uint8_t tx_len;
        // Frame format
           /*Pre Build Header of IEEE 802.15.4 Data frame.*/
   tx_frame[ 0 ] = 0x61;         //FCF.
   tx_frame[ 1 ] = 0x88;         //FCF.
                 //Sequence number set during frame transmission.
   tx_frame[ 3 ] = curr_addr->pan_id & 0xFF;                         //Dest.
PANID.
   tx_frame[ 4 ] = (curr_addr->pan_id >> 8 ) & 0xFF;            //Dest. PANID.
   tx_frame[ 5 ] = dest & 0xFF;                                               //Dest.
Addr.
   tx_frame[ 6 ] = (dest >> 8 ) & 0xFF;                                  //Dest. Addr.
   tx_frame[ 7 ] = curr_addr->short_addr & 0xFF;                //Source Addr.
   tx_frame[ 8 ] = (curr_addr->short_addr >> 8 ) & 0xFF;   //Source Addr.

        tx_len = 9;
        return tx_len;
};

uint8_t mac_build_command_frame (uint8_t *tx_frame, address_entry *curr_addr,
uint8_t *dest){
        uint8_t tx_len = 0;
        //                      Frame format - Assn. Response
        //FCF
        //bits 0-2: 110 - command frame
        //     3  : 0   - security disabled
        //     4  : 0   - frame pending
        //     5  : 1   - ACK requested
        //     6  : 1   - PAN ID compression enabled (send only dest)
        //    7-9 : 000 - reserved
        //   10-11: 01  - dst addr is 64-bit long addr
        //   12-13: 00  - only set to 'one' if security enabled
        //   14-15: 11  - src addr is 64-bit long addr
        tx_frame[ 0 ] = 0x63; //01100011 - FCF
        tx_frame[ 1 ] = 0xC8; //11001000 - FCF
                // tx_frame[ 2 ] = sequency number is set when sent
   tx_frame[ 3 ] = curr_addr->pan_id & 0xFF;                                    //Dest.
PANID.
   tx_frame[ 4 ] = (curr_addr->pan_id >> 8 ) & 0xFF;                      //Dest. PANID.
   tx_frame[ 5 ] =  *dest & 0xFF;                                                        //Dest.
Addr.
   tx_frame[ 6 ] = (*(dest+1) ) & 0xFF;                                          //Dest. Addr.
   tx_frame[ 7 ] = curr_addr->long_addr[ 0 ]  & 0xFF;                   //Source Addr.
   tx_frame[ 8 ] = (curr_addr->long_addr[1] ) & 0xFF;                   //Source Addr.
         tx_frame[ 9 ] = (curr_addr->long_addr[2] ) & 0xFF;                //Source Addr.
         tx_frame[ 10] = (curr_addr->long_addr[3] ) & 0xFF;               //Source Addr.
         tx_frame[ 11] = (curr_addr->long_addr[4] ) & 0xFF;               //Source Addr.
         tx_frame[ 12] = (curr_addr->long_addr[5] ) & 0xFF;               //Source Addr.
         tx_frame[ 13] = (curr_addr->long_addr[6] ) & 0xFF;               //Source Addr.
         tx_frame[ 14] = (curr_addr->long_addr[7] ) & 0xFF;               //Source Addr.

         tx_len = 15;
         return tx_len;
};
```

Mac.h – Zigbee mac implementation header file

```
// MAC Specification File

#ifndef MAC_BASE_H
#define MAC_BASE_H

// Variables

typedef struct address_entry {
         uint16_t pan_id;
         uint16_t short_addr;
         uint8_t long_addr[ 8 ];
} address_entry;

uint8_t macBSN;
uint8_t macFSN;

typedef struct assoc_node {
         uint16_t short_addr;
         uint8_t long_addr[ 8 ];
} assoc_node_t;


// Prototypes

void mac_init (void);
```

```
uint8_t mac_build_beacon_frame (uint8_t *, address_entry *);
uint8_t mac_build_data_frame (uint8_t *, address_entry *, uint16_t);
uint8_t mac_build_command_frame (uint8_t *, address_entry *, uint8_t *);
void mac_add_to_tx_buffer (void);

#endif //MAC_BASE_H
```

MAX3420E_BF1.h – Contains pound defines for register locations and bit masks, also macros

```
// MAX3420E_BF1.h
// Macros
// See the single bug fix below.
//
#define SETBIT(reg,val) wreg(reg,(rreg(reg)|val));
#define CLRBIT(reg,val) wreg(reg,(rreg(reg)&~val));

// ************ BUG FIX ************
//#define STALL_EP0 wreg(9,0x23); // Set all three EP0 stall bits--data stage IN/OUT
and status stage
// BUG FIX 2-9-06. The above statement hard-codes the register number to 9, ignoring
the fact that
// the wreg function expects the register numbers to be pre-shifted 3 bits to put them into
the 5 MSB's of
// the SPI command byte. Here is the correction:

#define STALL_EP0 wreg(rEPSTALLS,0x23);      // Set all three EP0 stall bits--data
stage IN/OUT and status stage

// ******** END OF BUG FIX**********

#define MSB(word) (unsigned char)(((unsigned short)(word) >> 8) & 0xff)
#define LSB(word) (unsigned char)((unsigned short)(word) & 0xff)

// MAX3420E Registers
#define rEP0FIFO    0<<3
#define rEP1OUTFIFO 1<<3
#define rEP2INFIFO  2<<3
#define rEP3INFIFO  3<<3
#define rSUDFIFO    4<<3
#define rEP0BC      5<<3
#define rEP1OUTBC   6<<3
#define rEP2INBC    7<<3
#define rEP3INBC    8<<3
#define rEPSTALLS   9<<3
#define rCLRTOGS    10<<3
```

```
#define rEPIRQ      11<<3
#define rEPIEN      12<<3
#define rUSBIRQ     13<<3
#define rUSBIEN     14<<3
#define rUSBCTL     15<<3
#define rCPUCTL     16<<3
#define rPINCTL     17<<3
#define rRevision   18<<3
#define rFNADDR     19<<3
#define rGPIO       20<<3

// MAX3420E bit masks for register bits
// R9 EPSTALLS Register
#define bmACKSTAT   0x40
#define bmSTLSTAT   0x20
#define bmSTLEP3IN  0x10
#define bmSTLEP2IN  0x08
#define bmSTLEP1OUT 0x04
#define bmSTLEP0OUT 0x02
#define bmSTLEP0IN  0x01

// R10 CLRTOGS Register
#define bmEP3DISAB  0x80
#define bmEP2DISAB  0x40
#define bmEP1DISAB  0x20
#define bmCTGEP3IN  0x10
#define bmCTGEP2IN  0x08
#define bmCTGEP1OUT 0x04

// R11 EPIRQ register bits
#define bmSUDAVIRQ  0x20
#define bmIN3BAVIRQ 0x10
#define bmIN2BAVIRQ 0x08
#define bmOUT1DAVIRQ 0x04
#define bmOUT0DAVIRQ 0x02
#define bmIN0BAVIRQ 0x01

// R12 EPIEN register bits
#define bmSUDAVIE   0x20
#define bmIN3BAVIE  0x10
#define bmIN2BAVIE  0x08
#define bmOUT1DAVIE 0x04
#define bmOUT0DAVIE 0x02
#define bmIN0BAVIE  0x01

// R13 USBIRQ register bits
```

```
#define bmURESDNIRQ 0x80
#define bmVBUSIRQ   0x40
#define bmNOVBUSIRQ 0x20
#define bmSUSPIRQ   0x10
#define bmURESIRQ   0x08
#define bmBUSACTIRQ 0x04
#define bmRWUDNIRQ  0x02
#define bmOSCOKIRQ  0x01

// R14 USBIEN register bits
#define bmURESDNIE  0x80
#define bmVBUSIE    0x40
#define bmNOVBUSIE  0x20
#define bmSUSPIE    0x10
#define bmURESIE    0x08
#define bmBUSACTIE  0x04
#define bmRWUDNIE   0x02
#define bmOSCOKIE   0x01

// R15 USBCTL Register
#define bmHOSCSTEN  0x80
#define bmVBGATE    0x40
#define bmCHIPRES   0x20
#define bmPWRDOWN   0x10
#define bmCONNECT   0x08
#define bmSIGRWU    0x04

// R16 CPUCTL Register
#define bmIE        0x01

// R17 PINCTL Register
#define bmFDUPSPI   0x10
#define bmINTLEVEL  0x08
#define bmPOSINT    0x04
#define bmGPOB      0x02
#definebmGPOA      0x01

//
// GPX[B:A] settings (PINCTL register)
#define gpxOPERATE  0x00
#define gpxVBDETECT 0x01
#define gpxBUSACT   0x02
#define gpxSOF      0x03

// ***********************
// Standard USB Requests
```

```
#define SR_GET_STATUS            0x00   // Get Status
#define SR_CLEAR_FEATURE         0x01   // Clear Feature
#define SR_RESERVED              0x02   // Reserved
#define SR_SET_FEATURE           0x03   // Set Feature
#define SR_SET_ADDRESS           0x05   // Set Address
#define SR_GET_DESCRIPTOR        0x06   // Get Descriptor
#define SR_SET_DESCRIPTOR        0x07   // Set Descriptor
#define SR_GET_CONFIGURATION            0x08   // Get Configuration
#define SR_SET_CONFIGURATION            0x09   // Set Configuration
#define SR_GET_INTERFACE         0x0a   // Get Interface
#define SR_SET_INTERFACE         0x0b   // Set Interface

// Get Descriptor codes
#define GD_DEVICE          0x01   // Get device descriptor: Device
#define GD_CONFIGURATION         0x02   // Get device descriptor: Configuration
#define GD_STRING          0x03   // Get device descriptor: String
#define GD_HID                   0x21   // Get descriptor: HID
#define GD_REPORT              0x22     // Get descriptor: Report

// SETUP packet offsets
#define bmRequestType        0
#definebRequest            1
#define wValueL                  2
#define wValueH                  3
#define wIndexL                  4
#define wIndexH                  5
#define wLengthL           6
#define wLengthH           7

// HID bRequest values
#define GET_REPORT               1
#define GET_IDLE           2
#define GET_PROTOCOL           3
#define SET_REPORT               9
#define SET_IDLE           0x0A
#define SET_PROTOCOL          0x0B
#define INPUT_REPORT          1
```

Usbinclude.c – Main USB code

```
#include "MAX3420E_BF1BasestationV2.h"      // MAX3420E registers (rREGNAME),
bits (bmBITNAME), and some handy macros
#include "enumerationBasestationV2.h"  // HID keyboard enumeration data

#include <avr/io.h>
```

```
// function prototypes
void SPI_Init(void);           // Configure MAXQ2000 and MAX3420E IO pins for SPI
void Reset_MAX(void);           // Reset the MAX3420E
void wreg(unsigned char r,unsigned char v);       // Write a MAX3420E register byte
void wregAS(unsigned char r,unsigned char v);     // Same as 'wreg' but also set the
ACKSTAT bit in the SPI command byte
unsigned char rreg(unsigned char r);             // Read a MAX3420E register byte
unsigned char rregAS(unsigned char r);            // Same as 'rreg' but also set the
ACKSTAT bit
void readbytes(unsigned char reg, unsigned char N, unsigned char *p);  // Read N
MAX3420E FIFO bytes into the array p
void writebytes(unsigned char reg, unsigned char N, const unsigned char *p); // Write N
MAX3420E FIFO bytes into the array p
unsigned char MAX_Int_Pending(void);     // Poll the MAX3420E INT pin (set for active
low level)

// USB functions
void std_request(void);
void class_request(void);
void vendor_request(void);
void send_descriptor(void);
void send_keystroke(unsigned char);
void feature(unsigned char);
void get_status(void);
void set_interface(void);
void get_interface(void);
void set_configuration(void);
void get_configuration(void);

// Application code
void do_SETUP(void);       // Handle a USB SETUP transfer
void do_IN3(void);         // Send keyboard characters over Endpoint 3-IN
void do_OUT1(void);
void check_for_resume(void);
void service_irqs(void);
void initialize_MAX(void);
unsigned char device_to_computer(unsigned char*, unsigned char);

//Global variables
unsigned char SUD[8];                    // Local copy of the 8 setup data read from the
MAX3420E SUDFIFO
unsigned char msgidx,msglen;             // Text string in EnumApp_enum_data.h--
index and length
unsigned char configval;                 // Set/Get_Configuration value
unsigned char ep3stall;                          // Flag for EP3 Stall, set by Set_Feature,
reported back in Get_Status
unsigned char interfacenum;          // Set/Get interface value
unsigned char RWU_enabled;          // Set by Set/Clear_Feature RWU request, sent back
for Get_Status-RWU
unsigned char Suspended;           // Tells the main loop to look for host resume and
RWU pushbutton
unsigned char TO_COMPUTER[64];
unsigned char IN3_BUFFER[6][65];             // Buffer for data to be written to endpoint
3
unsigned char IN3_BUFFER_LOWER;
unsigned char IN3_BUFFER_UPPER;
unsigned char OUT1_BUFFER[65];
unsigned char OUT1_NEWDATA;
//
#define ENABLE_IRQS wreg(rEPIEN,(bmSUDAVIE+bmIN3BAVIE));
wreg(rUSBIEN,(bmURESIE+bmURESDNIE));
// Note: the SUSPEND IRQ will be enabled later, when the device is configured.
// This prevents repeated SUSPEND IRQ's

void initialize_MAX(void){
        ep3stall=0;                          // EP3 inintially un-halted (no stall) (CH9
testing)
        msgidx = 0;                          // start of KB Message[]
        msglen = sizeof(Message);   // so we can check for the end of the message
        IN3_BUFFER_LOWER = 0;
        IN3_BUFFER_UPPER = 0;
        OUT1_NEWDATA = 0;
        // software flags
        configval=0;              // at pwr on OR bus reset we're unconfigured
        Suspended=0;
        RWU_enabled=0;             // Set by host Set_Feature(enable RWU) request
        //
        SPI_Init();              // set up MAXQ2000 to use its SPI port as a master
        //
        // Always set the FDUPSPI bit in the PINCTL register FIRST if you are using the
SPI port in
        // full duplex mode. This configures the port properly for subsequent SPI
accesses.
        //
        wreg(rPINCTL,(bmFDUPSPI+bmINTLEVEL+gpxSOF)); // MAX3420:
SPI=full-duplex, INT=neg level, GPX=SOF
        Reset_MAX();
        wreg(rGPIO,0x0F);                 // lites off (Active HIGH)
        // This is a self-powered design, so the host could turn off Vbus while we are
powered.
```

```
	// Therefore set the VBGATE bit to have the MAX3420E automatically
disconnect the D+
	// pullup resistor in the absense of Vbus. Note: the VBCOMP pin must be
connected to Vbus
	// or pulled high for this code to work--a low on VBCOMP will prevent USB
connection.
	wreg(rUSBCTL,(bmCONNECT+bmVBGATE)); // VBGATE=1 disconnects D+
pullup if host turns off VBUS
	ENABLE_IRQS
	wreg(rCPUCTL,bmIE);		// Enable the INT pin
}



//
***********************************************************************
************
// This endless loop checks for two high priority events (every time through the loop):
// 1. USB suspend ("Suspended" flag = 1). If suspended, checks for resume signaling.
// 2. A MAX3420E pending interrupt.
//
// Every 20 msec, it reads the "SEND" pushbutton. Every half second, it blinks
// the "Loop Active" light.
//
// ********************************** MAIN
*****************************************



void check_for_resume(void){

	if(rreg(rUSBIRQ) & bmBUSACTIRQ){	// THE HOST RESUMED BUS
TRAFFIC

		Suspended=0;			// no longer suspended
	}
	else if(RWU_enabled){			// Only if the host enabled RWU

	if((rreg(rGPIO)&0x40)==0){	// See if the Remote Wakeup button was pressed

	Suspended=0;			// no longer suspended
	SETBIT(rUSBCTL,bmSIGRWU)	// signal RWU
	while ((rreg(rUSBIRQ)&bmRWUDNIRQ)==0) ;	// spin until RWU signaling
done
	CLRBIT(rUSBCTL,bmSIGRWU)	// remove the RESUME signal
	wreg(rUSBIRQ,bmRWUDNIRQ);	// clear the IRQ
```

```
          while((rreg(rGPIO)&0x40)==0) ;  // hang until RWU button released
          wreg(rUSBIRQ,bmBUSACTIRQ);    // wait for bus traffic -- clear the BUS
Active IRQ
          while((rreg(rUSBIRQ) & bmBUSACTIRQ)==0) ; // & hang here until it's set
again...
       }
          }
}

void service_irqs(void){

          unsigned char itest1,itest2;
          itest1 = rreg(rEPIRQ);            // Check the EPIRQ bits
          itest2 = rreg(rUSBIRQ);           // Check the USBIRQ bits
          if(itest1 & bmSUDAVIRQ){
          wreg(rEPIRQ,bmSUDAVIRQ);     // clear the SUDAV IRQ
          do_SETUP();
   }
          if(itest1 & bmIN3BAVIRQ){          // Was an EP3-IN packet just dispatched to
the host?
       do_IN3();
   }                         // NOTE: don't clear the IN3BAVIRQ bit here--loading the EP3-IN
byte
                         // count register in the do_IN3() function does it.
          if(itest1 & bmOUT1DAVIRQ){
          do_OUT1();
          }

          if((configval != 0) && (itest2&bmSUSPIRQ)){   // HOST suspended bus for 3
msec

          wreg(rUSBIRQ,(bmSUSPIRQ+bmBUSACTIRQ));  // clear the IRQ and bus
activity IRQ
          Suspended=1;                 // signal the main loop
   }

   if(rreg(rUSBIRQ)& bmURESIRQ){

          wreg(rUSBIRQ,bmURESIRQ);      // clear the IRQ
   }
          if(rreg(rUSBIRQ) & bmURESDNIRQ){

          wreg(rUSBIRQ,bmURESDNIRQ);    // clear the IRQ bit
          Suspended=0;                 // in case we were suspended
          ENABLE_IRQS                  // ...because a bus reset clears the IE bits
   }
```

```
}

void do_SETUP(void){

          readbytes(rSUDFIFO,8,SUD);          // got a SETUP packet. Read 8 SETUP bytes
          switch(SUD[bmRequestType]&0x60){     // Parse the SETUP packet. For request
type, look only at b6&b5

          case 0x00:      std_request();            break;
          case 0x20:      class_request();          break;  // just a stub in this program
          case 0x40:      vendor_request();         break;  // just a stub in this program
          default:        STALL_EP0                    // unrecognized request type
   }
}

void do_IN3(void){

   unsigned char index;
   unsigned char count;

   if( IN3_BUFFER_LOWER == IN3_BUFFER_UPPER ){
      wreg(rEP3INFIFO,0);
      wreg(rEP3INBC,1);
   }
   else{
      count = IN3_BUFFER[IN3_BUFFER_LOWER][0];
      for(index=0;index<count;index++){
         wreg(rEP3INFIFO,IN3_BUFFER[IN3_BUFFER_LOWER][index+1]);
      }
      wreg(rEP3INBC,count);
      if (IN3_BUFFER_LOWER == 5){
         IN3_BUFFER_LOWER = 0;
      }
      else{
         IN3_BUFFER_LOWER++;
      }
   }
}

void do_OUT1(void){

   unsigned char count;
   unsigned char index;

   count = rreg(rEP1OUTBC);
   OUT1_BUFFER[0] = count;
   for(index=0;index<count;index++){
      OUT1_BUFFER[index+1] = rreg(rEP1OUTFIFO);
   }
   OUT1_NEWDATA = 1;
   wreg(rEPIRQ, bmOUT1DAVIRQ);
}


unsigned char device_to_computer(unsigned char *ptr, unsigned char count){

   unsigned char index;

   if( IN3_BUFFER_UPPER+1 == IN3_BUFFER_LOWER || (IN3_BUFFER_UPPER
== 5 && IN3_BUFFER_LOWER == 0)){
      return 1;
   }
   else{
      IN3_BUFFER[IN3_BUFFER_UPPER][0] = count;
      for(index=0;index<count;index++){
         IN3_BUFFER[IN3_BUFFER_UPPER][index+1] = *ptr;
         ptr++;
      }
      if(IN3_BUFFER_UPPER == 5){
         IN3_BUFFER_UPPER = 0;
      }
      else{
         IN3_BUFFER_UPPER++;
      }
      return 0;
   }
}



//******************
void std_request(void){

         switch(SUD[bRequest]){

                  case    SR_GET_DESCRIPTOR:      send_descriptor();    break;
                  case    SR_SET_FEATURE:         feature(1);           break;
                  case    SR_CLEAR_FEATURE:       feature(0);           break;
                  case    SR_GET_STATUS:          get_status();         break;
                  case    SR_SET_INTERFACE:       set_interface();      break;
                  case    SR_GET_INTERFACE:       get_interface();      break;
                  case    SR_GET_CONFIGURATION:   get_configuration();  break;
```

```
                case    SR_SET_CONFIGURATION:   set_configuration();  break;
                case    SR_SET_ADDRESS:         rregAS(rFNADDR);      break;  //
discard return value
                default:  STALL_EP0
        }
}

//**************************
void set_configuration(void){

        configval=SUD[wValueL];          // Store the config value
        if(configval != 0){              // If we are configured,
                SETBIT(rUSBIEN,bmSUSPIE);      // start looking for SUSPEND
interrupts
        }
        rregAS(rFNADDR);                 // dummy read to set the ACKSTAT bit
}

void get_configuration(void){

        wreg(rEP0FIFO,configval);        // Send the config value
        wregAS(rEP0BC,1);
}

//**********************
void set_interface(void){       // All we accept are Interface=0 and AlternateSetting=0,
otherwise send STALL

        unsigned char dumval;
        if((SUD[wValueL]==0)                  // wValueL=Alternate Setting index
        &&(SUD[wIndexL]==0)){                 // wIndexL=Interface index

                dumval=rregAS(rFNADDR);// dummy read to set the ACKSTAT bit
        }
        else STALL_EP0
}

//**********************
void get_interface(void){       // Check for Interface=0, always report AlternateSetting=0

        if(SUD[wIndexL]==0){                  // wIndexL=Interface index

                wreg(rEP0FIFO,0);             // AS=0
                wregAS(rEP0BC,1);             // send one byte, ACKSTAT
        }
        else STALL_EP0
```

```
}

//*******************
void get_status(void)
{
unsigned char testbyte;
testbyte=SUD[bmRequestType];
switch(testbyte)
        {
        case 0x80:                      // directed to DEVICE
                wreg(rEP0FIFO,(RWU_enabled+1));         // first byte is 000000rs where
r=enabled for RWU and s=self-powered.
                wreg(rEP0FIFO,0x00);            // second byte is always 0
                wregAS(rEP0BC,2);       // load byte count, arm the IN transfer, ACK
the status stage of the CTL transfer
                break;
        case 0x81:                      // directed to INTERFACE
                wreg(rEP0FIFO,0x00);            // this one is easy--two zero bytes
                wreg(rEP0FIFO,0x00);
                wregAS(rEP0BC,2);       // load byte count, arm the IN transfer, ACK
the status stage of the CTL transfer
                break;
        case 0x82:                      // directed to ENDPOINT
                if(SUD[wIndexL]==0x83)          // We only reported ep3, so it's the
only one the host can stall IN3=83
            {
            wreg(rEP0FIFO,ep3stall);// first byte is 0000000h where h is the halt (stall)
bit
            wreg(rEP0FIFO,0x00);        // second byte is always 0
            wregAS(rEP0BC,2);           // load byte count, arm the IN transfer, ACK
the status stage of the CTL transfer
            break;
            }
                else  STALL_EP0         // Host tried to stall an invalid endpoint (not
3)
        default:    STALL_EP0           // don't recognize the request
        }
}

//
*****************************************************************
*********************
// FUNCTION: Set/Get_Feature. Call as feature(1) for Set_Feature or feature(0) for
Clear_Feature.
// There are two set/clear feature requests:
//      To a DEVICE:        Remote Wakeup (RWU).
//        To an ENDPOINT:    Stall (EP3 only for this app)
//
void feature(unsigned char sc)
{
unsigned char mask;
  if((SUD[bmRequestType]==0x02)  // dir=h->p, recipient = ENDPOINT
  &&  (SUD[wValueL]==0x00)       // wValueL is feature selector, 00 is EP Halt
  &&  (SUD[wIndexL]==0x83))      // wIndexL is endpoint number IN3=83
      {
      mask=rreg(rEPSTALLS);   // read existing bits
      if(sc==1)              // set_feature
        {
        mask += bmSTLEP3IN;       // Halt EP3IN
        ep3stall=1;
        }
      else                   // clear_feature
        {
        mask &= ~bmSTLEP3IN;      // UnHalt EP3IN
        ep3stall=0;
        wreg(rCLRTOGS,bmCTGEP3IN);  // clear the EP3 data toggle
        }
      wreg(rEPSTALLS,(mask|bmACKSTAT)); // Don't use wregAS for this--directly
writing the ACKSTAT bit
      }
  else if ((SUD[bmRequestType]==0x00)     // dir=h->p, recipient = DEVICE
         &&  (SUD[wValueL]==0x01))        // wValueL is feature selector, 01 is
Device_Remote_Wakeup
          {
          RWU_enabled = sc<<1;        // =2 for set, =0 for clear feature. The shift puts it in
the get_status bit position.
          rregAS(rFNADDR);            // dummy read to set ACKSTAT
          }
  else STALL_EP0
}

//***********************
void send_descriptor(void)
{
unsigned short reqlen,sendlen,desclen;
const unsigned char *pDdata;                            // pointer to ROM Descriptor
data to send
//
// NOTE This function assumes all descriptors are 64 or fewer bytes and can be sent in a
single packet
//
desclen = 0;                                    // check for zero as error condition (no case
statements satisfied)
reqlen = SUD[wLengthL] + 256*SUD[wLengthH]; // 16-bit
        switch (SUD[wValueH])                           // wValueH is descriptor type
        {
        case  GD_DEVICE:
          desclen = DD[0];      // descriptor length
          pDdata = DD;
          break;
        case  GD_CONFIGURATION:
          desclen = CD[2];      // Config descriptor includes interface, HID, report and ep
descriptors
          pDdata = CD;
          break;
        case  GD_STRING:
          desclen = strDesc[SUD[wValueL]][0];   // wValueL=string index, array[0] is the
length
          pDdata = strDesc[SUD[wValueL]];       // point to first array element
          break;
        case  GD_HID:
          desclen = CD[18];
          pDdata = &CD[18];
          break;
        case  GD_REPORT:
          desclen = CD[25];
          pDdata = RepD;
      break;
        }       // end switch on descriptor type
//
if (desclen!=0)                 // one of the case statements above filled in a value
        {
        sendlen = (reqlen <= desclen) ? reqlen : desclen; // send the smaller of requested
and avaiable
      writebytes(rEP0FIFO,sendlen,pDdata);
        wregAS(rEP0BC,sendlen);   // load EP0BC to arm the EP0-IN transfer &
ACKSTAT
        }
else STALL_EP0  // none of the descriptor types match
}

void class_request(void)
{
STALL_EP0
}

void vendor_request(void)
```

```
{
STALL_EP0
}

// ******************* END of ENUMERATION CODE ********************
//
void Reset_MAX(void)
{
unsigned char dum;
wreg(rUSBCTL,0x20);         // chip reset
wreg(rUSBCTL,0x00);         // remove the reset
   do                // Chip reset stops the oscillator. Wait for it to stabilize.
   {
   dum=rreg(rUSBIRQ);
   dum &= bmOSCOKIRQ;
   }
   while (dum==0);
}
//
// -----------------------------------------------------------
// The code below customizes this app for the ATMEGA128
// microprocessor and the AVR studio most likely gcc compiler. Only this
// section changes if you use a different uP and/or compiler.
// -----------------------------------------------------------
//
// The MAX3420E is wired to the ATMEGA128 as follows:
//
// MISO PB3
// SCLK PB1
// MOSI PB2
// SS#  PB4
// GPX  PF0
// INT  PE4



// Register SPCR bit masks
#define bmSPR0  0x01
#define bmSPR1  0x02
#define bmCPHA  0x04
#define bmCPOL  0x08
#define bmMSTR  0x10
#define bmDORD  0x20
#define bmSPE   0x40
#define bmSPIE  0x80
```

```
// Register SPSR bit masks
#define bmSPI2X 0x01
#define bmWCOL  0x40
#define bmSPIF  0x80

// PortB bit masks


//
#define SS_HI PORTB |= (1<<PB4);    // SS# connected to Port5 bit4 in this app
#define SS_LO PORTB &= ~(1<<PB4);

unsigned char MAX_Int_Pending(void)
{
return (unsigned char)((PINE&0x10)==0);
}

void SPI_Init(void)
{
// Set up the ATMEGA128 SPI port

  DDRB |= (1<<DDB4)|(1<<DDB2)|(1<<DDB1);    // Set SS#, MOSI, SCLK as output
ports
  SPSR = bmSPI2X;                                          // Enable SPI 2x
speed mode
  SS_HI                                       // SS# high
  DDRB &= ~(1<<DDB3);                         // Set SPI MISO as input
  DDRF &= ~(1<<DDF0);                                      // Set GPX as input
  DDRE &= ~(1<<DDE4);                         // Set PE4 (INT) as input
  SPCR = bmSPE | bmMSTR;                                // Enable SPI as master
}

void wreg(unsigned char reg, unsigned char dat)
{
   cli();
   DDRB |= (1<<DDB0);
   PORTB |= ( 1<<0x00);



  SS_LO                               // Set SS# low
  SPDR = reg+2;                         // send the register number with the DIR bit (b1) set to
WRITE
  while (!(SPSR & (1<<SPIF)));            // loop if data still being sent
  SPDR = dat;                           // send the data
  while (!(SPSR & (1<<SPIF)));            // loop if data still being sent
  SS_HI                               // set SS# high
  sei();
}


// Write a MAX3410E register with the "ACK STATUS" bit set in the command byte
void wregAS(unsigned char reg, unsigned char dat)
{

    cli();
    DDRB |= (1<<DDB0);
    PORTB |= ( 1<<0x00);


  SS_LO                                    // Set SS# low
  SPDR = reg+3;                            // reg number with DIR=1 (write) and ACKSTAT=1
  while (!(SPSR & (1<<SPIF)));             // loop if data still being sent
  SPDR = dat;                              // send the data
  while (!(SPSR & (1<<SPIF)));             // loop if data still being sent
  SS_HI                                    // set SS# high
  sei();
}

// Read a register, return its value.
unsigned char rreg(unsigned char reg)
{
unsigned char dum;

    cli();
    DDRB |= (1<<DDB0);
    PORTB |= ( 1<<0x00);


  SS_LO
  SPDR = reg;                              // reg number w. dir=0 (IN)
  while (!(SPSR & (1<<SPIF)));             // loop if data still being sent
  dum = SPDR;                              // NECESSARY TO RE-ENABLE THE INPUT
BUFFER in unsigned char MODE
  SPDR=0x00;                               // data is don't care, we're clocking in MISO bits
  while (!(SPSR & (1<<SPIF)));             // loop if data still being sent
  SS_HI
  sei();
  return(SPDR);
}

// Read a byte (as rreg), but also set the AckStat bit in the command byte.
unsigned char rregAS(unsigned char reg)
```

```
{
unsigned char dum;

   cli();
   DDRB |= (1<<DDB0);
   PORTB |= ( 1<<0x00);


  SS_LO
  SPDR = reg+1;                          // reg number w. dir=0 (IN) and ACKSTAT=1
  while (!(SPSR & (1<<SPIF)));              // loop if data still being sent
  dum = SPDR;                            // NECESSARY TO RE-ENABLE THE INPUT
BUFFER in unsigned char MODE
  SPDR=0x00;                             // data is don't care, we're clocking in MISO bits
  while (!(SPSR & (1<<SPIF)));              // loop if data still being sent
  SS_HI
  sei();
  return(SPDR);
}

void readbytes(unsigned char reg, unsigned char N, unsigned char *p)
{
unsigned char j;

   cli();
   DDRB |= (1<<DDB0);
   PORTB |= ( 1<<0x00);



   SS_LO
   SPDR = reg;                          // write bit b1=0 to command a read operation
   while (!(SPSR & (1<<SPIF)));           // loop if data still being sent
   j = SPDR;                          // NECESSARY TO RE-ENABLE THE INPUT
BUFFER in unsigned char MODE
   for(j=0; j<N; j++)
   {
      SPDR = 0x00;                       // dummy value to get the next read byte
      while (!(SPSR & (1<<SPIF)));        // loop if data still being sent
      *p = SPDR;                        // store it in the data array
      p++;                           // bump the pointer
   }
  SS_HI
  sei();
}
void writebytes(unsigned char reg, unsigned char N, const unsigned char *p)
```

```
{
unsigned char j,wd;

    cli();
    DDRB |= (1<<DDB0);
    PORTB |= ( 1<<0x00);


    SS_LO
    SPDR = reg+2;                          // write bit b1=1 to command a write operation
    while (!(SPSR & (1<<SPIF)));           // loop if data still being sent
    for(j=0; j<N; j++)
    {
        wd = *p;                           // write the array value
        SPDR = wd;
        while (!(SPSR & (1<<SPIF)));       // loop if data still being sent
        p++;                               // bump the pointer
    }
    SS_HI
  sei();
}
//
// Diagnostic Aid:
// Call this function from main() to verify operation of your SPI port.
//
void test_SPI(void)        // Use this to check your versions of the rreg and wreg functions
{
unsigned char j,wr,rd;
SPI_Init();                // Configure and initialize the uP's SPI port
wreg(rPINCTL,bmFDUPSPI);   // MAX3420: SPI=full-duplex
wreg(rUSBCTL,bmCHIPRES);   // reset the MAX3420E
wreg(rUSBCTL,0);           // remove the reset
wr=0x01;                   // initial register write value
for(j=0; j<8; j++)
  {
  wreg(rUSBIEN,wr);
  rd = rreg(rUSBIEN);
  wr <<= 1;      // Put a breakpoint here. Values of 'rd' should be 01,02,04,08,10,20,40,80
  }
}

/* ISR( INT4_vect ) {

        EIFR  |= (1 << INTF4);

// Put crap here.
}*/
```

enumerationBasestation.h – Contains data that is sent during enumeration

```
// EnumApp_enum_data.h
// Enumeration tables & HID keyboard data
//
const unsigned char DD[]=   // DEVICE Descriptor
      {0x12,              // bLength = 18d
       0x01,              // bDescriptorType = Device (1)
       0x00,0x01,         // bcdUSB(L/H) USB spec rev (BCD)
       0x00,0x00,0x00,    // bDeviceClass, bDeviceSubClass, bDeviceProtocol
       0x40,              // bMaxPacketSize0 EP0 is 64 bytes
       0x6A,0x0B,         // idVendor(L/H)--Maxim is 0B6A
       0x46,0x53,         // idProduct(L/H)--5346
       0x34,0x12,         // bcdDevice--1234
       1,2,3,             // iManufacturer, iProduct, iSerialNumber
       1};                // bNumConfigurations

const unsigned char CD[]=   // CONFIGURATION Descriptor
      {0x09,              // bLength
       0x02,              // bDescriptorType = Config
       0x29,0x00,         // wTotalLength(L/H) = 41 bytes
       0x01,              // bNumInterfaces
       0x01,              // bConfigValue
       0x00,              // iConfiguration
       0xA0,              // bmAttributes. b7=1 b6=self-powered b5=RWU supported
       100,               // MaxPower is 250 ma
// INTERFACE Descriptor
       0x09,              // length = 9
       0x04,              // type = IF
       0x00,              // IF #0
       0x00,              // bAlternate Setting
       0x02,              // bNum Endpoints
       0x03,              // bInterfaceClass = HID
       0x00,0x00,         // bInterfaceSubClass, bInterfaceProtocol
       0x00,              // iInterface
// HID Descriptor--It's at CD[18]
       0x09,              // bLength
       0x21,              // bDescriptorType = HID
       0x10,0x01,         // bcdHID(L/H) Rev 1.1
       0x00,              // bCountryCode (none)
       0x01,              // bNumDescriptors (one report descriptor)
       0x22,              // bDescriptorType  (report)
       19,0,              // CD[25]: wDescriptorLength(L/H) (report descriptor size is 43
bytes)
```

```
// Endpoint Descriptor IN3
        0x07,           // bLength
        0x05,           // bDescriptorType (Endpoint)
        0x83,           // bEndpointAddress (EP3-IN)
        0x03,           // bmAttributes (interrupt)
        64,0,           // wMaxPacketSize (64)
        10,             // bInterval (poll every 10 msec)
// Endpoint Descriptor OUT1
        0x07,           // bLength
        0x05,           // bDescriptorType (Endpoint)
        0x01,           // bEndpointAddress (EP1-OUT)
        0x03,           // bmAttributes (interrupt)
        64,0,           // wMaxPacketSize (64)
        10};            // bInterval (poll every 10 msec)


const unsigned char RepD[]=   // Report descriptor
        {
        0x05,0x0C,      // Usage Page (consumer)
        0x09,0xE0,      // Usage ID (volume)
        0xA1,0x01,      // Collection
        0x09,0xE0,      //        Usage Page 7 (keyboard/keypad)
        //0x19,0xE0,    //        Usage Minimum = 224
        //0x29,0xE7,    //        Usage Maximum = 231
        0x15,0x00,      //        Logical Minimum = 0
        0x25,0xFF,      //        Logical Maximum = 255
        0x75,0x08,      //        Report Size = 8
        0x95,0x01,      //        Report Count = 1
        0x81,0x02,      //    Input(Data,Variable,Absolute)
        //0x95,0x01,    //        Report Count = 1
        //0x75,0x08,    //        Report Size = 8
        //0x81,0x01,    //    Input(Constant)
        //0x19,0x00,    //        Usage Minimum = 0
        //0x29,0x65,    //        Usage Maximum = 101
        //0x15,0x00,    //        Logical Minimum = 0,
        //0x25,0x65,    //        Logical Maximum = 101
        //0x75,0x08,    //        Report Size = 8
        //0x95,0x01,    //        Report Count = 1
        //0x81,0x00,    //    Input(Data,Variable,Array)
        0xC0};          // End Collection

// STRING descriptors. An array of string arrays

const unsigned char strDesc[][64]= {
// STRING descriptor 0--Language string
{
        0x04,             // bLength
    0x03,            // bDescriptorType = string
    0x09,0x04       // wLANGID(L/H) = English-United Sates
},
// STRING descriptor 1--Manufacturer ID
{
        12,          // bLength
    0x03,            // bDescriptorType = string
    'M',0,'a',0,'x',0,'i',0,'m',0 // text in Unicode
},
// STRING descriptor 2 - Product ID
{   24,          // bLength
    0x03,            // bDescriptorType = string
    'M',0,'A',0,'X',0,'3',0,'4',0,'2',0,'0',0,'E',0,' ',0,
        'E',0,'n',0,'u',0,'m',0,' ',0,'C',0,'o',0,'d',0,'e',0
},

// STRING descriptor 3 - Serial Number ID
{       20,          // bLength
    0x03,            // bDescriptorType = string
    'S',0,
    '/',0,
    'N',0,
    ' ',0,
    '3',0,
    '4',0,
    '2',0,
    '0',0,
        'E',0,
}};

const unsigned char Message[]={ // each letter is 3 bytes--shiftcode, 00, HID keycode
    0x00,0x00,0x28,     // (cr)
    0x02,0x00,0x17,     // T (02 is shift)
    0x00,0x00,0x0B,     // h
    0x00,0x00,0x08,     // e
    0x00,0x00,0x2C,     // (sp)
    0x02,0x00,0x07,     // D
    0x00,0x00,0x15,     // r
    0x00,0x00,0x08,     // e
    0x00,0x00,0x04,     // a
    0x00,0x00,0x10,     // m
    0x02,0x00,0x17,     // T
    0x00,0x00,0x08,     // e
    0x00,0x00,0x08,     // a
    0x00,0x00,0x10,     // m
    0x00,0x00,0x2C,     // (sp)
    0x00,0x00,0x0C,     // i
    0x00,0x00,0x16,     // s
    0x00,0x00,0x2C,     // (sp)
    0x00,0x00,0x0C,     // i
    0x00,0x00,0x11,     // n
    0x00,0x00,0x2C,     // (sp)
    0x00,0x00,0x1C,     // y
    0x00,0x00,0x12,     // o
    0x00,0x00,0x18,     // u
    0x00,0x00,0x15,     // r
    0x00,0x00,0x2C,     // (sp)
    0x00,0x00,0x10,     // m
    0x00,0x00,0x04,     // a
    0x00,0x00,0x06,     // c
    0x00,0x00,0x0B,     // h
    0x00,0x00,0x0C,     // i
    0x00,0x00,0x11,     // n
    0x00,0x00,0x08,     // e
    0x00,0x00,0x28};    // !
```

LCDfunctions.c – Contains all functions used by LCD and Keypad

```
#include <avr/io.h>
#include <util/delay.h>
#include "LCD.h"
#include "SampleFont8.h"

//initialize
#define reset 0xE2
#define display_on 0xAF
#define display_off 0xAE
#define display_all_eon 0xA5
#define display_normal_eon 0xA4
#define display_normal 0xA6
#define display_reverse 0xA7
#define adc_normal 0xA0
#define adc_reverse    0xA1
#define shl_normal 0xC0
#define shl_reverse 0xC8
#define lcd_biashi 0xA3
#define vc_on 0x2C
#define vr_on 0x2E
#define vf_on 0x2F
```

```
#define reg_res 0x26
#define ref_volt_reg_mode 0x81
#define ref_volt_reg 0x08

//display
#define num_columns 128
#define num_rows 64
#define num_pages 8
#define display_page 0xB0
#define display_column_l 0x00
#define display_column_h 0x10

static const BYTE JustTbl[8] = {
        0, num_columns/4,                    // L00J,L25J
        num_columns/3, num_columns/2,            // L33J,C50J
        (num_columns*2)/3, (num_columns*3)/4,   // R66J,R75J
        num_columns, 0                       // R100J,ABSJ
};


BYTE invertflg = 0;      // normally 0, to invert data set to 0xFF
BYTE CJ_code = 0;        // control/justify code for text display
BYTE startcol;       // save starting col for later use
BYTE column;
BYTE pagenr;

char *keyPadMatrix[17] =
{
            "D","3","2","1",
    "A","6","5","4",
    "B","9","8","7",
    "C","F","0","T","y"
};

void CMD(int command, int delay)
{
            clear_RS();
            clear_E();
            clear_RW();
            PORTC = command;
            set_E();
            clear_E();
            _delay_us(delay);
}

void initialize()
```

```
{
	//Hard Inputs
	//PS hi for parallel
	//MI lo for 8080
	//CS1B
	//CS2 hi
	//RESETB hi

	//Set Output Pins
	DDRC |= 0xFF;
	DDRD |= 0b11100000;

	//Initialization Sequence
	CMD(reset,100);
	CMD(display_normal,10);
	CMD(adc_reverse,10);
	CMD(shl_normal,10);
	CMD(lcd_biashi,10);
	CMD(vc_on,10);
	CMD(vr_on,10);
	CMD(vf_on,10);
	CMD(reg_res,100);
	CMD(ref_volt_reg_mode,100);
	CMD(ref_volt_reg,100);
	lcd_clear();
	CMD(display_on,100);
	CMD(display_all_eon,500);
	_delay_ms(500);
	CMD(display_normal_eon,10);
}

void write_byte(BYTE arg)
{
	set_RS();
	clear_RW();
	PORTC = arg;
	set_E();
	clear_E();
}

BYTE read_byte( void )
{
	BYTE data;
	set_RW();
	set_RS();
	DDRC = 0;
```

```
            set_E();
            data = PINC;
            _delay_ms(1);
            clear_E();
            return data;
}

void lcd_clear(void)
{
   BYTE page, col;

   for (page = 0; page < num_pages; page++)
   {
      CMD(display_page + page,1);  // set page number
      CMD(display_column_h,1);        // set column 0
      CMD(display_column_l,1);        // set column 0

      for (col = 0; col < num_columns + 4; col++)
      {
         write_byte(0);       // write blank to display
      }
   }
}

void clear_RS()
{
            PORTD &= ~(1 << RS);
}

void set_RS()
{
            PORTD |= (1 << RS);
}

void clear_RW()
{
            PORTD &= ~(1 << RW);
}

void set_RW()
{
            PORTD |= (1 << RW);
}

void clear_E()
{
        PORTD &= ~(1 << E);
}

void set_E()
{
        PORTD |= (1 << E);
}

void tdt(void)
{       set_position(0x1C,0x3B);

        write_byte(0x18);
        write_byte(0x03);
        write_byte(0xC2);
        write_byte(0x8A);
        write_byte(0xA3);
        write_byte(0xA3);
        write_byte(0x8A);
        write_byte(0xC2);
        write_byte(0x03);
        write_byte(0x18);

        _delay_ms(500);
        _delay_ms(500);
        _delay_ms(500);
        _delay_ms(500);
        _delay_ms(500);
        _delay_ms(500);
        _delay_ms(500);
        _delay_ms(500);

        while(1)
        {
                CMD(display_reverse,500);
                _delay_ms(250);
                CMD(display_normal,500);
                _delay_ms(250);
        }
}

void set_position( BYTE page, BYTE col )
{
        CMD(display_page + page,1);
        CMD(display_column_h + ((col + 4) >> 4),1);
        CMD(display_column_l + ((col + 4) & 0xF),1);
}
```

```
void write_small_char(char chr)
{
   BYTE len, a;
   BYTE *bits;

   if (chr < firstchr_S) return;   // char code out of range
   a = chr - firstchr_S;
   if (a >= nr_chrs_S) return;     // char code out of range

   len = lentbl_S[a];              // number of columns
   bits = chrtbl_S[a];             // bitmap data pointer

         if (len == 0) return;
         if (column + len >= 128) {
                  column = 0;
                  pagenr++;
         }
   set_position(pagenr, column);   //set starting position

   for (a = 0; a < len; a++)
   {
      write_byte(*(bits + a));     // bitmap data
   }

   if (! (CJ_code & NOSPC))
   {
      write_byte(0);               // blank pixel column between chars
   }

   column += len;                  // update current column

   if (! (CJ_code & NOSPC))
   {
      column++;                    // include blank between chars
   }
}

void write_text(String_T *strg)
{
   BYTE a, b;
   char *cp;

   pagenr = strg->page;            // save page number
   CJ_code = strg->cj_code;        // save CJ code
   b = CJ_code & JUSTIFY;          // isolate justify code
```

```
    if (b == L00J)
    {
        column = 0;
    }
    else if (b == ABSJ)
    {
        column = strg->abs;
    }
    else
    {
        a = char_len(strg->text);// get display length in pixels
        if (b != R100J)              // if not right justify
        {
            a /= 2;                  // use half length for centering
        }
        column = JustTbl[b] - a;    // set starting column
    }

    if (column >= num_columns)          // if over/underflow
    {
        column = 0;                 // use left justify
    }
    startcol = column;               // save start column
    cp = strg->text;                // get pointer to the text

    while (*cp)
    {
         write_small_char(*cp++);
    }
}

BYTE char_len(char *chr)
{
    BYTE len, first, nrc, *lentbl;

        first = firstchr_S;
        lentbl = lentbl_S;
                    nrc = nr_chrs_S;

    len = 0;

    while (*chr)
    {
                    if (*chr >= first && ((*chr) - first) < nrc)
                    {
                        len += *(lentbl + (*chr) - first);

                        if (! (CJ_code & NOSPC))
                        {
                              len++;             // include blank between chars
                        }
                  }
      chr++;
   }
   return len;
}

void ScanKeyMatrixInit()
{
        DDRA = 0xFF;
        PORTA = 0b11110000;
        DDRA = 0b11110000;        // rows as outputs and columns as inputs
}

char *ScanKeyMatrix()
{
int key;


        key = 0;

        do{

        PORTA = 0b11100000;
      if( !(PINA & 0b00000001) )
        break;
      key++;
      if( !(PINA & 0b00000010) )
        break;
      key++;
      if( !(PINA & 0b00000100) )
        break;
      key++;
                  if( !(PINA & 0b00001000) )
        break;
      key++;

        PORTA = 0b11010000;
      if( !(PINA & 0b00000001) )
        break;
      key++;
```

```
        if( !(PINA & 0b00000010) )
          break;
        key++;
        if( !(PINA & 0b00000100) )
          break;
        key++;
                    if( !(PINA & 0b00001000) )
          break;
        key++;

          PORTA = 0b10110000;
        if( !(PINA & 0b00000001) )
          break;
        key++;
        if( !(PINA & 0b00000010) )
          break;
        key++;
        if( !(PINA & 0b00000100) )
          break;
        key++;
                    if( !(PINA & 0b00001000) )
          break;
        key++;

          PORTA = 0b01110000;
        if( !(PINA & 0b00000001) )
          break;
        key++;
        if( !(PINA & 0b00000010) )
          break;
        key++;
        if( !(PINA & 0b00000100) )
          break;
        key++;
                    if( !(PINA & 0b00001000) )
          break;
        key++;
      } while(0);

    return keyPadMatrix[key];
}

void page_clear(BYTE page)
{
                BYTE col;
        CMD(display_page + page,1);  // set page number
        CMD(display_column_h,1);        // set column 0
        CMD(display_column_l,1);        // set column 0

        for (col = 0; col < num_columns + 4; col++)
        {
           write_byte(0);       // write blank to display
        }
}
```

LCD.h – header file for LCD

```
#if !defined(LCD_H)
#define LCD_H

typedef unsigned char BYTE;
typedef unsigned short WORD;
typedef unsigned char BITS;

#define E           7
#define RS          5
#define RW          6

typedef struct      // by putting text string definitions
{                   // in a structure, you don't have to
   BYTE page;       // mess with string lengths in the
   BYTE cj_code;    // declaration.
   BYTE abs;
   char *text;      // pointer to null terminated string
} String_T;

/* Examples:
String_T str1 = { 2, C50J+FONT16, 0, "Display text" };  // pages 2 & 3, large font,
centered
String_T str2 = { 0, ABSJ, 45, "Display text" };        // page 0, small font, start in col 45
String_T str3 = { 5, L33J, 0, buffer };                 // page 5, small font, centered @ 1/3,
refering
char buffer[128];                                       // to string that's not constant
*/

//  text string display control/justification codes

#define NOSPC       0x10    // no space between characters
#define JUSTIFY     7       // bit mask
#define L00J        0       // left, starting @ col 0
#define L25J        1       // left, centered @ 1/4
```

```
#define L33J        2       // left, centered @ 1/3
#define C50J        3       // centered @ 1/2
#define R66J        4       // right, centered @ 2/3
#define R75J        5       // right, centered @ 3/4
#define R100J       6       // right, ending @ last col
#define ABSJ        7       // starting @ abs col #

#define CLEAR       (String_T *)0xFFFF  // code to clear display
#define EOL         (String_T *)0       // end of list of String_Ts

extern BYTE invertflg;      // normally 0, to invert data set to 0xFF
extern BYTE CJ_code;        // control/justify code for text display
extern BYTE contrast;       // user contrast setting
extern BYTE startcol;       // save starting col for later use
extern BYTE column;         // current col

void CMD(int command, int delay);
void initialize();
void write_byte(BYTE arg);
void lcd_clear(void);
void clear_RS();
void set_RS();
void clear_RW();
void set_RW();
void clear_E();
void set_E();
void tdt(void); //graphic
void set_position( BYTE page, BYTE col );
BYTE read_byte( void );
void write_text(String_T *text);
void write_small_char(char chr);
BYTE char_len(char *chr);
void page_clear(BYTE page);
#endif
```

SampleFont8.c – small fond bitmap data

```
#include "SampleFont8.h"

static BYTE chr20[3] = { 0x00, 0x00, 0x00 };
static BYTE chr21[3] = { 0x00, 0x5F, 0x00 };
static BYTE chr22[3] = { 0x07, 0x00, 0x07 };
static BYTE chr23[5] = { 0x14, 0x7F, 0x14, 0x7F, 0x14 };
static BYTE chr24[5] = { 0x26, 0x49, 0x7F, 0x49, 0x32 };
static BYTE chr25[5] = { 0x63, 0x13, 0x08, 0x64, 0x63 };
static BYTE chr26[5] = { 0x36, 0x49, 0x00, 0x22, 0x50 };
static BYTE chr27[3] = { 0x04, 0x03, 0x00 };
static BYTE chr28[3] = { 0x1C, 0x22, 0x41 };
static BYTE chr29[3] = { 0x41, 0x22, 0x1C };
static BYTE chr2A[5] = { 0x14, 0x2A, 0x1C, 0x2A, 0x14 };
static BYTE chr2B[5] = { 0x08, 0x08, 0x3E, 0x08, 0x08 };
static BYTE chr2C[2] = { 0x80, 0x60 };
static BYTE chr2D[5] = { 0x08, 0x08, 0x08, 0x08, 0x08 };
static BYTE chr2E[1] = { 0x40 };
static BYTE chr2F[5] = { 0x60, 0x10, 0x08, 0x04, 0x03 };
static BYTE chr30[5] = { 0x3E, 0x51, 0x49, 0x45, 0x3E };
static BYTE chr31[5] = { 0x00, 0x42, 0x7F, 0x40, 0x00 };
static BYTE chr32[5] = { 0x62, 0x51, 0x49, 0x49, 0x46 };
static BYTE chr33[5] = { 0x22, 0x41, 0x49, 0x49, 0x36 };
static BYTE chr34[5] = { 0x18, 0x14, 0x12, 0x7F, 0x10 };
static BYTE chr35[5] = { 0x27, 0x49, 0x49, 0x49, 0x31 };
static BYTE chr36[5] = { 0x3C, 0x4A, 0x49, 0x49, 0x30 };
static BYTE chr37[5] = { 0x01, 0x71, 0x09, 0x05, 0x03 };
static BYTE chr38[5] = { 0x36, 0x49, 0x49, 0x49, 0x36 };
static BYTE chr39[5] = { 0x06, 0x49, 0x49, 0x29, 0x1E };
static BYTE chr3A[1] = { 0x14 };
static BYTE chr3B[2] = { 0x80, 0x68 };
static BYTE chr3C[4] = { 0x08, 0x14, 0x22, 0x41 };
static BYTE chr3D[4] = { 0x14, 0x14, 0x14, 0x14 };
static BYTE chr3E[4] = { 0x41, 0x22, 0x14, 0x08 };
static BYTE chr3F[5] = { 0x02, 0x01, 0x51, 0x09, 0x06 };
static BYTE chr40[5] = { 0x3E, 0x41, 0x5D, 0x00, 0x4E };
static BYTE chr41[5] = { 0x7C, 0x12, 0x11, 0x12, 0x7C };
static BYTE chr42[5] = { 0x7F, 0x49, 0x49, 0x49, 0x36 };
static BYTE chr43[5] = { 0x3E, 0x41, 0x41, 0x41, 0x22 };
static BYTE chr44[5] = { 0x7F, 0x41, 0x41, 0x22, 0x1C };
static BYTE chr45[5] = { 0x7F, 0x49, 0x49, 0x49, 0x41 };
static BYTE chr46[5] = { 0x7F, 0x09, 0x09, 0x09, 0x01 };
static BYTE chr47[5] = { 0x3E, 0x41, 0x49, 0x29, 0x72 };
static BYTE chr48[5] = { 0x7F, 0x08, 0x08, 0x08, 0x7F };
static BYTE chr49[3] = { 0x41, 0x7F, 0x41 };
static BYTE chr4A[5] = { 0x20, 0x40, 0x41, 0x3F, 0x01 };
static BYTE chr4B[5] = { 0x7F, 0x08, 0x14, 0x22, 0x41 };
static BYTE chr4C[4] = { 0x7F, 0x40, 0x40, 0x40 };
static BYTE chr4D[5] = { 0x7F, 0x02, 0x0C, 0x02, 0x7F };
static BYTE chr4E[5] = { 0x7F, 0x06, 0x08, 0x30, 0x7F };
static BYTE chr4F[5] = { 0x3E, 0x41, 0x41, 0x41, 0x3E };
static BYTE chr50[5] = { 0x7F, 0x09, 0x09, 0x09, 0x06 };
static BYTE chr51[5] = { 0x3E, 0x41, 0x51, 0x21, 0x5E };
static BYTE chr52[5] = { 0x7F, 0x09, 0x19, 0x29, 0x46 };
static BYTE chr53[5] = { 0x26, 0x49, 0x49, 0x49, 0x32 };
static BYTE chr54[5] = { 0x01, 0x01, 0x7F, 0x01, 0x01 };
static BYTE chr55[5] = { 0x3F, 0x40, 0x40, 0x40, 0x3F };
static BYTE chr56[5] = { 0x0F, 0x30, 0x40, 0x30, 0x0F };
static BYTE chr57[7] = { 0x0F, 0x30, 0x40, 0x38, 0x40, 0x30, 0x0F };
static BYTE chr58[5] = { 0x63, 0x14, 0x08, 0x14, 0x63 };
static BYTE chr59[5] = { 0x07, 0x08, 0x70, 0x08, 0x07 };
static BYTE chr5A[5] = { 0x61, 0x51, 0x49, 0x45, 0x43 };
static BYTE chr5B[3] = { 0x7F, 0x41, 0x41 };
static BYTE chr5C[5] = { 0x03, 0x04, 0x08, 0x10, 0x60 };
static BYTE chr5D[3] = { 0x41, 0x41, 0x7F };
static BYTE chr5E[5] = { 0x08, 0x04, 0x02, 0x04, 0x08 };
static BYTE chr5F[5] = { 0x40, 0x40, 0x40, 0x40, 0x40 };
static BYTE chr60[3] = { 0x03, 0x04, 0x00 };
static BYTE chr61[4] = { 0x20, 0x54, 0x54, 0x78 };
static BYTE chr62[5] = { 0x7F, 0x28, 0x44, 0x44, 0x38 };
static BYTE chr63[4] = { 0x38, 0x44, 0x44, 0x28 };
static BYTE chr64[5] = { 0x38, 0x44, 0x44, 0x28, 0x7F };
static BYTE chr65[4] = { 0x38, 0x54, 0x54, 0x48 };
static BYTE chr66[4] = { 0x08, 0x7E, 0x09, 0x02 };
static BYTE chr67[4] = { 0x98, 0xA4, 0xA4, 0x58 };
static BYTE chr68[5] = { 0x7F, 0x08, 0x04, 0x04, 0x78 };
static BYTE chr69[2] = { 0x3D, 0x40 };
static BYTE chr6A[3] = { 0x80, 0x84, 0x7D };
static BYTE chr6B[4] = { 0x7F, 0x10, 0x28, 0x44 };
static BYTE chr6C[3] = { 0x01, 0x7F, 0x00 };
static BYTE chr6D[6] = { 0x7C, 0x08, 0x04, 0x78, 0x04, 0x78 };
static BYTE chr6E[5] = { 0x7C, 0x08, 0x04, 0x04, 0x78 };
static BYTE chr6F[4] = { 0x38, 0x44, 0x44, 0x38 };
static BYTE chr70[5] = { 0xFC, 0x18, 0x24, 0x24, 0x18 };
static BYTE chr71[5] = { 0x18, 0x24, 0x24, 0x18, 0xFC };
static BYTE chr72[5] = { 0x7C, 0x08, 0x04, 0x04, 0x08 };
static BYTE chr73[4] = { 0x48, 0x54, 0x54, 0x24 };
static BYTE chr74[3] = { 0x04, 0x3E, 0x44 };
static BYTE chr75[5] = { 0x3C, 0x40, 0x40, 0x20, 0x7C };
static BYTE chr76[5] = { 0x1C, 0x20, 0x40, 0x20, 0x1C };
static BYTE chr77[5] = { 0x3C, 0x40, 0x30, 0x40, 0x3C };
static BYTE chr78[5] = { 0x44, 0x28, 0x10, 0x28, 0x44 };
static BYTE chr79[4] = { 0x1C, 0xA0, 0xA0, 0x7C };
static BYTE chr7A[4] = { 0x64, 0x54, 0x54, 0x4C };
static BYTE chr7B[3] = { 0x08, 0x36, 0x41 };
static BYTE chr7C[3] = { 0x00, 0x7F, 0x00 };
static BYTE chr7D[3] = { 0x41, 0x36, 0x08 };
static BYTE chr7E[2] = { 0x00, 0x00 };
static BYTE chr7F[1] = {0x00};
static BYTE chr80[7] = { 0x1C, 0x22, 0x41, 0x4F, 0x41, 0x22, 0x1C };
static BYTE chr81[7] = { 0x1C, 0x22, 0x41, 0x49, 0x45, 0x22, 0x1C };
static BYTE chr82[7] = { 0x1C, 0x22, 0x41, 0x49, 0x49, 0x2A, 0x1C };
static BYTE chr83[7] = { 0x1C, 0x22, 0x41, 0x49, 0x51, 0x22, 0x1C };
static BYTE chr84[7] = { 0x1C, 0x22, 0x41, 0x79, 0x41, 0x22, 0x1C };
static BYTE chr85[7] = { 0x1C, 0x22, 0x51, 0x49, 0x41, 0x22, 0x1C };
static BYTE chr86[7] = { 0x1C, 0x2A, 0x49, 0x49, 0x41, 0x22, 0x1C };
static BYTE chr87[7] = { 0x1C, 0x22, 0x45, 0x49, 0x41, 0x22, 0x1C };

BYTE lentbl_S[104] = { 3, 3, 3, 5, 5, 5, 5, 3, 3,
                       3, 5, 5, 2, 5, 1, 5, 5, 5,
                       5, 5, 5, 5, 5, 5, 5, 5, 1,
                       2, 4, 4, 4, 5, 5, 5, 5, 5,
                       5, 5, 5, 5, 5, 3, 5, 5, 4,
                       5, 5, 5, 5, 5, 5, 5, 5, 5,
                       5, 7, 5, 5, 5, 3, 5, 3, 5,
                       5, 3, 4, 5, 4, 5, 4, 4, 4,
                       5, 2, 3, 4, 3, 6, 5, 4, 5,
                       5, 5, 4, 3, 5, 5, 5, 5, 4,
                       4, 3, 3, 3, 2, 1, 7, 7, 7,
                       7, 7, 7, 7, 7 };

BYTE * chrtbl_S[104] = { chr20, chr21, chr22, chr23, chr24, chr25, chr26,
                         chr27, chr28, chr29, chr2A, chr2B, chr2C, chr2D,
                         chr2E, chr2F, chr30, chr31, chr32, chr33, chr34,
                         chr35, chr36, chr37, chr38, chr39, chr3A, chr3B,
                         chr3C, chr3D, chr3E, chr3F, chr40, chr41, chr42,
                         chr43, chr44, chr45, chr46, chr47, chr48, chr49,
                         chr4A, chr4B, chr4C, chr4D, chr4E, chr4F, chr50,
                         chr51, chr52, chr53, chr54, chr55, chr56, chr57,
                         chr58, chr59, chr5A, chr5B, chr5C, chr5D, chr5E,
                         chr5F, chr60, chr61, chr62, chr63, chr64, chr65,
                         chr66, chr67, chr68, chr69, chr6A, chr6B, chr6C,
                         chr6D, chr6E, chr6F, chr70, chr71, chr72, chr73,
                         chr74, chr75, chr76, chr77, chr78, chr79, chr7A,
                         chr7B, chr7C, chr7D, chr7E, chr7F, chr80, chr81,
                         chr82, chr83, chr84, chr85, chr86, chr87 };
```

SampleFont8.h – small font header file

```
#include "LCD.h"

#define nr_chrs_S 104
#define firstchr_S 32

extern BYTE lentbl_S[104];
extern BYTE * chrtbl_S[104];
```

zigbee_repeater.c

```
#include <stdint.h>
#include <stdbool.h>
#include <avr/io.h>
#include <util/delay.h>

#include "zigbee/config_uart_extended.h" // See this file for all project options.
#include "zigbee/compiler.h"
#include "zigbee/at86rf230_registermap.h"

#include "zigbee/hal_avr_mega128.h"
#include "zigbee/hal_avr_mega128.c"

#include "zigbee/tat.h"
#include "zigbee/tat.c"

#include "zigbee/com.h"
#include "zigbee/com.c"

#include "LCD.h"
#include "SampleFont8.h"
#include "SampleFont16.h"
#include "LCDfunctions.c"

/*============================ INCLDUE
=========================================*/
// See above (ADC 2/24/2008)
/*============================ MACROS
==========================================*/
/*============================ TYPEDEFS
=========================================*/

/*============================ VARIABLES
=======================================*/
static uint8_t tx_frame[ 127 ]; //!< Buffer used to build TX frames. (Size must be max
PSDU length.)

static hal_rx_frame_t rx_pool[ RX_POOL_SIZE ]; //!< Pool of hal_rx_frame_t's.
static hal_rx_frame_t *rx_pool_start; //!< Pointer to start of pool.
static hal_rx_frame_t *rx_pool_end; //!< Pointer to end of pool.
static hal_rx_frame_t *rx_pool_head; //!< Pointer to next hal_rx_frame_t it is possible to
write.
static hal_rx_frame_t *rx_pool_tail; //!< Pointer to next hal_rx_frame_t that can be read
from the pool.
static uint8_t rx_pool_items_free; //!< Number of free items (hal_rx_frame_t) in the pool.
static uint8_t rx_pool_items_used; // !< Number of used items.
static bool rx_pool_overflow_flag; //!< Flag that is used to signal a pool overflow.

static bool rx_flag; //!< Flag used to mask between the two possible TRX_END events.

/*static uint8_t debug_pll_transition[] = "State transition failed\r\n"; //!< Debug Text.
static uint8_t debug_type_message[] = "\r<---Type Message:\r\n"; //!< Debug Text.
static uint8_t debug_data_sent[] = "<---TX OK.\r\n"; //!< Debug Text.
static uint8_t debug_data_received[] = "\r--->Rx:\r"; //!< Debug Text.
static uint8_t debug_lqi[] = "LQI: "; //!< Debug Text.
static uint8_t debug_rx_pool_overflow[] = "RX Buffer Overflow!\r\n"; //!< Debug Text.
static uint8_t debug_transmission_failed[] = "TX Failed!\r\n"; //!< Debug Text.
static uint8_t debug_transmission_length[] = "Typed Message too long!!\r\n"; //!< Debug
Text.
static uint8_t debug_fatal_error[] = "A fatal error. System must be reset.\r\n"; //!< Debug
Text.*/
/*============================ PROTOTYPES
====================================*/
static bool trx_init( void );
static void trx_end_handler( uint32_t time_stamp );
static void rx_pool_init( void );

/*! \brief This function is used to initialize the TRX.
 *
 * The TAT will be set up to run on the chosen operating channel, with CLKM diabled,
 * and then configure the RX_AACK and TX_ARET modes.
 *
 *  \retval true if the TRX was successfully configured.
 *  \retval false if the TRX was not configured properly.
 */
static bool trx_init( void ){

   static bool status;

   if (tat_init( ) != TAT_SUCCESS) {
      status = false;
   } else if (tat_set_operating_channel( OPERATING_CHANNEL ) != TAT_SUCCESS)
{
      status = false;
   } else if (tat_set_clock_speed( true, CLKM_DISABLED ) != TAT_SUCCESS) {
      status = false;
   } else{

      /*Set up the extended modes:*/
      //RX_AACK:
      tat_set_short_address( SHORT_ADDRESS ); //Short Address.
      tat_set_pan_id( PAN_ID ); //PAN ID.
```

```
        tat_set_device_role( false ); // No Coordintor support is necessary.

        //TX_ARET:
        tat_configure_csma( 234, 0xE2 ); // Default CSMA_SEED_0, MIN_BE = 3,
MAX_CSMA_RETRIES = , and CSMA_SEED_1 =

        //Both Modes:
        tat_use_auto_tx_crc( true ); //Automatic CRC must be enabled.
        hal_set_trx_end_event_handler( trx_end_handler ); // Event handler for TRX_END
events.
                hal_enable_trx_interrupt( );    //Enable interrupts from the radio
transceiver.

        status = true;
    } // end: if (tat_init( ) != TAT_SUCCESS) ...

    return status;
}

/*! \brief This function initialize the rx_pool. The rx_pool is in essence a FIFO.
 */
static void rx_pool_init( void ){

    rx_pool_start = rx_pool;
    rx_pool_end = &rx_pool[ RX_POOL_SIZE - 1 ];

    rx_pool_head = rx_pool_start;
    rx_pool_tail = rx_pool_end;

    rx_pool_items_free = RX_POOL_SIZE;
    rx_pool_items_used = 0;

    rx_pool_overflow_flag = false;
}

/*! \brief This function is the TRX_END event handler that is called from the
 *         TRX isr if assigned.
 *
 *  \param[in] time_stamp Interrupt timestamp in IEEE 802.15.4 symbols.
 */
static void trx_end_handler( uint32_t time_stamp ){

    if (rx_flag == true) {

        //Check if these is space left in the rx_pool.
        if (rx_pool_items_free == 0) {
            rx_pool_overflow_flag = true;
        } else {

            //Space left, so upload the received frame.
            hal_frame_read( rx_pool_head );

            //Then check the CRC. Will not store frames with invalid CRC.
            if (rx_pool_head->crc == true) {

                //Handle wrapping of rx_pool.
                if (rx_pool_head == rx_pool_end) {
                    rx_pool_head = rx_pool_start;
                } else {
                    ++rx_pool_head;
                } // end: if (rx_pool_head == rx_pool_end) ...

                --rx_pool_items_free;
                ++rx_pool_items_used;
            } // end: if (rx_pool_head->crc == true) ...
        } // end: if (rx_pool_items_free == 0) ...
    } // end:  if (rx_flag == true) ...
}

void main( void ){

            DDRB |=  0b00010000;
            PORTB |= 0b00010000;

    static uint8_t length_of_received_data = 0;
    static uint8_t frame_sequence_number = 0;
            int i;
            char tmpbuf[30];
    rx_flag = true;

    /*Pre Build Header of IEEE 802.15.4 Data frame.*/
    tx_frame[ 0 ] = 0x61; //FCF.
    tx_frame[ 1 ] = 0x88; //FCF.
                       //Sequence number set during frame transmission.
    tx_frame[ 3 ] = PAN_ID & 0xFF; //Dest. PANID.
    tx_frame[ 4 ] = (PAN_ID >> 8 ) & 0xFF; //Dest. PANID.
    tx_frame[ 5 ] = DEST_ADDRESS & 0xFF; //Dest. Addr.
    tx_frame[ 6 ] = (DEST_ADDRESS >> 8 ) & 0xFF; //Dest. Addr.
    tx_frame[ 7 ] = SHORT_ADDRESS & 0xFF; //Source Addr.
    tx_frame[ 8 ] = (SHORT_ADDRESS >> 8 ) & 0xFF; //Source Addr.

    rx_pool_init( );
```

```
trx_init( );
      initialize( );

//Set system state to RX_AACK_ON
if (tat_set_trx_state( RX_AACK_ON ) == TAT_SUCCESS) {

} // end: if (tat_set_trx_state( RX_AACK_ON ) != TAT_SUCCESS) ...

      sei( );

/*Enter Normal Program Flow:
   - Check for newly received frames. Double the number if received.
   - Notify on rx_pool overflow.
   - Try to send data on air interface, if something is received on UART/USB.
   - Notify if the typed message was too long.
 */
while (true) {

   //Check if we have received something on the air interface.
   if (rx_pool_items_used != 0) {

      //Handle wrapping of rx_pool.
      if (rx_pool_tail == rx_pool_end) {
         rx_pool_tail = rx_pool_start;
      } else {
         ++rx_pool_tail;
      } // end: if (rx_pool_tail == rx_pool_end) ...

      //Turn interrupts off for a short while to protect when status
      //information about the rx_pool is updated.
      cli( );

      ++rx_pool_items_free;
      --rx_pool_items_used;

      sei( );

                        length_of_received_data = (rx_pool_tail->length);
                        String_T line3 = {1, L00J, 0, "received packet"};
                        write_text(&line3);

   } // end: if (rx_pool_items_used != 0) ...
            else {}//{length_of_received_data = 0;}

   //Check for rx_pool overflow.
   if (rx_pool_overflow_flag == true) {
```

```
            cli();
            rx_pool_init( );
                        length_of_received_data = 0;
            sei();
        } // end: if (rx_pool_overflow_flag == true) ...

                String_T line1 = {0, L00J, 0, "Testing"};
                write_text(&line1);

        //Check for new data on the serial interface.
        //Check if data is ready to be sent.
        if ((length_of_received_data) > 0 ){

                                // Output data on LCD
                        for (i = 0;i<length_of_received_data-10;i++){
                                tmpbuf[i] = rx_pool_tail->data[i+10];
                        }
                        String_T line2 = {3/*rx_pool_tail->data[9]*/, L00J, 0, tmpbuf};
                        write_text(&line2);
                }

                if (1) {

            //Change state to TX_ARET_ON and send data if the state transition was
    successful.
            if (tat_set_trx_state( TX_ARET_ON ) == TAT_SUCCESS) {

                                tx_frame [ 9 ] = 0x03;
                                tx_frame [ 10] = 0x42;
                                tx_frame [ 11] = 0x4F;
                                tx_frame [ 12] = 0x4F;
                                tx_frame [ 13] = 0x42;
                                tx_frame [ 14] = 0x00;

              uint8_t *rx_frame = &(rx_pool_tail->data[9]);

              uint8_t tx_frame_length = 15; // Length of prebuilt frame header.
              tx_frame[ 2 ] = frame_sequence_number++; //Sequence Number.

              //Copy data into the TX frame buffer.
              /*do {
                 tx_frame[ tx_frame_length++ ] = *rx_frame++;
              } while (--length_of_received_data > 0);*/
```

```
                rx_flag = false; // Set the flag false, so that the TRX_END event is not
misinterpreted.

                if (tat_send_data_with_retry( tx_frame_length, tx_frame, 1 ) ==
TAT_SUCCESS) {
                    _delay_ms(100);
                } else {

                } // end:  if (tat_send_data_with_retry( tx_frame_length, tx_frame, 1 ) ...
            } else {

            } // end: if (tat_set_trx_state( TX_ARET_ON ) == TAT_SUCCESS) ...

            if (tat_set_trx_state( RX_AACK_ON ) != TAT_SUCCESS) {

            } // end: if (tat_set_trx_state( RX_AACK_ON ) != TAT_SUCCESS) ...

            rx_flag = true; // Set the flag back again. Only used to protec the frame
transmission.

                            for (i=0;i<65530;i++) { }
                            for (i=0;i<65530;i++) { }
                            for (i=0;i<65530;i++) { }

        } // end:
        //} // end: if (length_of_received_data == 1) ...
    } // emd: while (true) ...
}
/*EOF*/
```

At86rf230_registermap.h

```
/*
 * This file is autogenerated from regxml2include.py
 * Do not modify it, changes will be lost after rebuild.
 */
/**
 * @file
 * generated register definition file
 * Inputfile:  phy230_registermap_external.xml
 * Version:    1.9    for external use
 * Created at: Thu Jan 25 17:07:33 2007
 *
 */
/*
 * Copyright (c) 2006, Atmel Corporation All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions are met:
 *
 * 1. Redistributions of source code must retain the above copyright notice,
 * this list of conditions and the following disclaimer.
 *
 * 2. 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.
 *
 * 3. The name of ATMEL may not be used to endorse or promote products derived
 * from this software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY ATMEL ``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
EXPRESSLY AND
 * SPECIFICALLY DISCLAIMED. IN NO EVENT SHALL ATMEL 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.
 */
/**
 *  @author
 *      Atmel Corporation: http://www.atmel.com
 *      Support email: avr@atmel.com
 */

#ifndef PHY230_REGISTERMAP_EXTERNAL_H
#define PHY230_REGISTERMAP_EXTERNAL_H

#define HAVE_REGISTER_MAP (1)
/** Offset for register TRX_STATUS
```

```
 * @ingroup apiHalPHY230Reg
 */
#define RG_TRX_STATUS                   (0x01)
  /** Access parameters for sub-register CCA_DONE in register @ref
RG_TRX_STATUS
   * @ingroup apiHalPHY230Sreg
   */
# define SR_CCA_DONE                  0x01, 0x80, 7
  /** Access parameters for sub-register CCA_STATUS in register @ref
RG_TRX_STATUS
   * @ingroup apiHalPHY230Sreg
   */
# define SR_CCA_STATUS                0x01, 0x40, 6
# define SR_reserved_01_3            0x01, 0x20, 5
  /** Access parameters for sub-register TRX_STATUS in register @ref
RG_TRX_STATUS
   * @ingroup apiHalPHY230Sreg
   */
# define SR_TRX_STATUS                0x01, 0x1f, 0
   /** Constant P_ON for sub-register @ref SR_TRX_STATUS
    * @ingroup apiHalPHY230Const
    */
#  define P_ON                   (0)
   /** Constant BUSY_RX for sub-register @ref SR_TRX_STATUS
    * @ingroup apiHalPHY230Const
    */
#  define BUSY_RX                (1)
   /** Constant BUSY_TX for sub-register @ref SR_TRX_STATUS
    * @ingroup apiHalPHY230Const
    */
#  define BUSY_TX                (2)
   /** Constant RX_ON for sub-register @ref SR_TRX_STATUS
    * @ingroup apiHalPHY230Const
    */
#  define RX_ON                  (6)
   /** Constant TRX_OFF for sub-register @ref SR_TRX_STATUS
    * @ingroup apiHalPHY230Const
    */
#  define TRX_OFF                (8)
   /** Constant PLL_ON for sub-register @ref SR_TRX_STATUS
    * @ingroup apiHalPHY230Const
    */
#  define PLL_ON                 (9)
   /** Constant SLEEP for sub-register @ref SR_TRX_STATUS
    * @ingroup apiHalPHY230Const
    */
```

```
#  define SLEEP                    (15)
  /** Constant BUSY_RX_AACK for sub-register @ref SR_TRX_STATUS
   * @ingroup apiHalPHY230Const
   */
#  define BUSY_RX_AACK             (17)
  /** Constant BUSY_TX_ARET for sub-register @ref SR_TRX_STATUS
   * @ingroup apiHalPHY230Const
   */
#  define BUSY_TX_ARET             (18)
  /** Constant RX_AACK_ON for sub-register @ref SR_TRX_STATUS
   * @ingroup apiHalPHY230Const
   */
#  define RX_AACK_ON               (22)
  /** Constant TX_ARET_ON for sub-register @ref SR_TRX_STATUS
   * @ingroup apiHalPHY230Const
   */
#  define TX_ARET_ON               (25)
  /** Constant RX_ON_NOCLK for sub-register @ref SR_TRX_STATUS
   * @ingroup apiHalPHY230Const
   */
#  define RX_ON_NOCLK              (28)
  /** Constant RX_AACK_ON_NOCLK for sub-register @ref SR_TRX_STATUS
   * @ingroup apiHalPHY230Const
   */
#  define RX_AACK_ON_NOCLK         (29)
  /** Constant BUSY_RX_AACK_NOCLK for sub-register @ref SR_TRX_STATUS
   * @ingroup apiHalPHY230Const
   */
#  define BUSY_RX_AACK_NOCLK       (30)

/** Offset for register TRX_STATE
 * @ingroup apiHalPHY230Reg
 */
#define RG_TRX_STATE                    (0x02)
 /** Access parameters for sub-register TRAC_STATUS in register @ref
RG_TRX_STATE
  * @ingroup apiHalPHY230Sreg
  */
# define SR_TRAC_STATUS                 0x02, 0xe0, 5
 /** Access parameters for sub-register TRX_CMD in register @ref RG_TRX_STATE
  * @ingroup apiHalPHY230Sreg
  */
# define SR_TRX_CMD                 0x02, 0x1f, 0
  /** Constant CMD_NOP for sub-register @ref SR_TRX_CMD
   * @ingroup apiHalPHY230Const
   */
```

```
#  define CMD_NOP                  (0)
  /** Constant CMD_TX_START for sub-register @ref SR_TRX_CMD
   * @ingroup apiHalPHY230Const
   */
#  define CMD_TX_START             (2)
  /** Constant CMD_FORCE_TRX_OFF for sub-register @ref SR_TRX_CMD
   * @ingroup apiHalPHY230Const
   */
#  define CMD_FORCE_TRX_OFF        (3)
  /** Constant CMD_RX_ON for sub-register @ref SR_TRX_CMD
   * @ingroup apiHalPHY230Const
   */
#  define CMD_RX_ON                (6)
  /** Constant CMD_TRX_OFF for sub-register @ref SR_TRX_CMD
   * @ingroup apiHalPHY230Const
   */
#  define CMD_TRX_OFF              (8)
  /** Constant CMD_PLL_ON for sub-register @ref SR_TRX_CMD
   * @ingroup apiHalPHY230Const
   */
#  define CMD_PLL_ON               (9)
  /** Constant CMD_RX_AACK_ON for sub-register @ref SR_TRX_CMD
   * @ingroup apiHalPHY230Const
   */
#  define CMD_RX_AACK_ON           (22)
  /** Constant CMD_TX_ARET_ON for sub-register @ref SR_TRX_CMD
   * @ingroup apiHalPHY230Const
   */
#  define CMD_TX_ARET_ON           (25)

/** Offset for register TRX_CTRL_0
 * @ingroup apiHalPHY230Reg
 */
#define RG_TRX_CTRL_0                    (0x03)
 /** Access parameters for sub-register PAD_IO in register @ref RG_TRX_CTRL_0
  * @ingroup apiHalPHY230Sreg
  */
# define SR_PAD_IO                 0x03, 0xc0, 6
 /** Access parameters for sub-register PAD_IO_CLKM in register @ref
RG_TRX_CTRL_0
  * @ingroup apiHalPHY230Sreg
  */
# define SR_PAD_IO_CLKM            0x03, 0x30, 4
  /** Constant CLKM_2mA for sub-register @ref SR_PAD_IO_CLKM
   * @ingroup apiHalPHY230Const
   */
```

```
#  define CLKM_2mA                (0)
  /** Constant CLKM_4mA for sub-register @ref SR_PAD_IO_CLKM
   * @ingroup apiHalPHY230Const
   */
#  define CLKM_4mA                (1)
  /** Constant CLKM_6mA for sub-register @ref SR_PAD_IO_CLKM
   * @ingroup apiHalPHY230Const
   */
#  define CLKM_6mA                (2)
  /** Constant CLKM_8mA for sub-register @ref SR_PAD_IO_CLKM
   * @ingroup apiHalPHY230Const
   */
#  define CLKM_8mA                (3)
 /** Access parameters for sub-register CLKM_SHA_SEL in register @ref
RG_TRX_CTRL_0
  * @ingroup apiHalPHY230Sreg
  */
# define SR_CLKM_SHA_SEL              0x03, 0x08, 3
 /** Access parameters for sub-register CLKM_CTRL in register @ref
RG_TRX_CTRL_0
  * @ingroup apiHalPHY230Sreg
  */
# define SR_CLKM_CTRL                 0x03, 0x07, 0
  /** Constant CLKM_no_clock for sub-register @ref SR_CLKM_CTRL
   * @ingroup apiHalPHY230Const
   */
#  define CLKM_no_clock            (0)
  /** Constant CLKM_1MHz for sub-register @ref SR_CLKM_CTRL
   * @ingroup apiHalPHY230Const
   */
#  define CLKM_1MHz               (1)
  /** Constant CLKM_2MHz for sub-register @ref SR_CLKM_CTRL
   * @ingroup apiHalPHY230Const
   */
#  define CLKM_2MHz               (2)
  /** Constant CLKM_4MHz for sub-register @ref SR_CLKM_CTRL
   * @ingroup apiHalPHY230Const
   */
#  define CLKM_4MHz               (3)
  /** Constant CLKM_8MHz for sub-register @ref SR_CLKM_CTRL
   * @ingroup apiHalPHY230Const
   */
#  define CLKM_8MHz               (4)
  /** Constant CLKM_16MHz for sub-register @ref SR_CLKM_CTRL
   * @ingroup apiHalPHY230Const
   */
```

```
#  define CLKM_16MHz              (5)

/** Offset for register PHY_TX_PWR
 * @ingroup apiHalPHY230Reg
 */
#define RG_PHY_TX_PWR                   (0x05)
  /** Access parameters for sub-register TX_AUTO_CRC_ON in register @ref
RG_PHY_TX_PWR
   * @ingroup apiHalPHY230Sreg
   */
# define SR_TX_AUTO_CRC_ON           0x05, 0x80, 7
# define SR_reserved_05_2            0x05, 0x70, 4
  /** Access parameters for sub-register TX_PWR in register @ref RG_PHY_TX_PWR
   * @ingroup apiHalPHY230Sreg
   */
# define SR_TX_PWR                   0x05, 0x0f, 0

/** Offset for register PHY_RSSI
 * @ingroup apiHalPHY230Reg
 */
#define RG_PHY_RSSI                  (0x06)
# define SR_reserved_06_1            0x06, 0xe0, 5
  /** Access parameters for sub-register RSSI in register @ref RG_PHY_RSSI
   * @ingroup apiHalPHY230Sreg
   */
# define SR_RSSI                     0x06, 0x1f, 0

/** Offset for register PHY_ED_LEVEL
 * @ingroup apiHalPHY230Reg
 */
#define RG_PHY_ED_LEVEL                 (0x07)
  /** Access parameters for sub-register ED_LEVEL in register @ref
RG_PHY_ED_LEVEL
   * @ingroup apiHalPHY230Sreg
   */
# define SR_ED_LEVEL                 0x07, 0xff, 0

/** Offset for register PHY_CC_CCA
 * @ingroup apiHalPHY230Reg
 */
#define RG_PHY_CC_CCA                   (0x08)
  /** Access parameters for sub-register CCA_REQUEST in register @ref
RG_PHY_CC_CCA
   * @ingroup apiHalPHY230Sreg
   */
# define SR_CCA_REQUEST              0x08, 0x80, 7
```

```
  /** Access parameters for sub-register CCA_MODE in register @ref
RG_PHY_CC_CCA
   * @ingroup apiHalPHY230Sreg
   */
# define SR_CCA_MODE                0x08, 0x60, 5
  /** Access parameters for sub-register CHANNEL in register @ref RG_PHY_CC_CCA
   * @ingroup apiHalPHY230Sreg
   */
# define SR_CHANNEL                 0x08, 0x1f, 0

/** Offset for register CCA_THRES
 * @ingroup apiHalPHY230Reg
 */
#define RG_CCA_THRES                (0x09)
  /** Access parameters for sub-register CCA_CS_THRES in register @ref
RG_CCA_THRES
   * @ingroup apiHalPHY230Sreg
   */
# define SR_CCA_CS_THRES            0x09, 0xf0, 4
  /** Access parameters for sub-register CCA_ED_THRES in register @ref
RG_CCA_THRES
   * @ingroup apiHalPHY230Sreg
   */
# define SR_CCA_ED_THRES            0x09, 0x0f, 0

/** Offset for register IRQ_MASK
 * @ingroup apiHalPHY230Reg
 */
#define RG_IRQ_MASK                 (0x0e)
  /** Access parameters for sub-register IRQ_MASK in register @ref RG_IRQ_MASK
   * @ingroup apiHalPHY230Sreg
   */
# define SR_IRQ_MASK                0x0e, 0xff, 0

/** Offset for register IRQ_STATUS
 * @ingroup apiHalPHY230Reg
 */
#define RG_IRQ_STATUS               (0x0f)
  /** Access parameters for sub-register IRQ_7_BAT_LOW in register @ref
RG_IRQ_STATUS
   * @ingroup apiHalPHY230Sreg
   */
# define SR_IRQ_7_BAT_LOW           0x0f, 0x80, 7
  /** Access parameters for sub-register IRQ_6_TRX_UR in register @ref
RG_IRQ_STATUS
   * @ingroup apiHalPHY230Sreg
   */
# define SR_IRQ_6_TRX_UR             0x0f, 0x40, 6
  /** Access parameters for sub-register IRQ_5 in register @ref RG_IRQ_STATUS
   * @ingroup apiHalPHY230Sreg
   */
# define SR_IRQ_5                    0x0f, 0x20, 5
  /** Access parameters for sub-register IRQ_4 in register @ref RG_IRQ_STATUS
   * @ingroup apiHalPHY230Sreg
   */
# define SR_IRQ_4                    0x0f, 0x10, 4
  /** Access parameters for sub-register IRQ_3_TRX_END in register @ref
RG_IRQ_STATUS
   * @ingroup apiHalPHY230Sreg
   */
# define SR_IRQ_3_TRX_END            0x0f, 0x08, 3
  /** Access parameters for sub-register IRQ_2_RX_START in register @ref
RG_IRQ_STATUS
   * @ingroup apiHalPHY230Sreg
   */
# define SR_IRQ_2_RX_START           0x0f, 0x04, 2
  /** Access parameters for sub-register IRQ_1_PLL_UNLOCK in register @ref
RG_IRQ_STATUS
   * @ingroup apiHalPHY230Sreg
   */
# define SR_IRQ_1_PLL_UNLOCK         0x0f, 0x02, 1
  /** Access parameters for sub-register IRQ_0_PLL_LOCK in register @ref
RG_IRQ_STATUS
   * @ingroup apiHalPHY230Sreg
   */
# define SR_IRQ_0_PLL_LOCK           0x0f, 0x01, 0

/** Offset for register VREG_CTRL
 * @ingroup apiHalPHY230Reg
 */
#define RG_VREG_CTRL                 (0x10)
  /** Access parameters for sub-register AVREG_EXT in register @ref
RG_VREG_CTRL
   * @ingroup apiHalPHY230Sreg
   */
# define SR_AVREG_EXT                0x10, 0x80, 7
  /** Access parameters for sub-register AVDD_OK in register @ref RG_VREG_CTRL
   * @ingroup apiHalPHY230Sreg
   */
# define SR_AVDD_OK                  0x10, 0x40, 6
  /** Access parameters for sub-register AVREG_TRIM in register @ref
RG_VREG_CTRL
```

```
  * @ingroup apiHalPHY230Sreg
  */
# define SR_AVREG_TRIM              0x10, 0x30, 4
  /** Constant AVREG_1_80V for sub-register @ref SR_AVREG_TRIM
   * @ingroup apiHalPHY230Const
   */
#  define AVREG_1_80V           (0)
  /** Constant AVREG_1_75V for sub-register @ref SR_AVREG_TRIM
   * @ingroup apiHalPHY230Const
   */
#  define AVREG_1_75V           (1)
  /** Constant AVREG_1_84V for sub-register @ref SR_AVREG_TRIM
   * @ingroup apiHalPHY230Const
   */
#  define AVREG_1_84V           (2)
  /** Constant AVREG_1_88V for sub-register @ref SR_AVREG_TRIM
   * @ingroup apiHalPHY230Const
   */
#  define AVREG_1_88V           (3)
 /** Access parameters for sub-register DVREG_EXT in register @ref
RG_VREG_CTRL
  * @ingroup apiHalPHY230Sreg
  */
# define SR_DVREG_EXT               0x10, 0x08, 3
 /** Access parameters for sub-register DVDD_OK in register @ref RG_VREG_CTRL
  * @ingroup apiHalPHY230Sreg
  */
# define SR_DVDD_OK                 0x10, 0x04, 2
 /** Access parameters for sub-register DVREG_TRIM in register @ref
RG_VREG_CTRL
  * @ingroup apiHalPHY230Sreg
  */
# define SR_DVREG_TRIM              0x10, 0x03, 0
  /** Constant DVREG_1_80V for sub-register @ref SR_DVREG_TRIM
   * @ingroup apiHalPHY230Const
   */
#  define DVREG_1_80V           (0)
  /** Constant DVREG_1_75V for sub-register @ref SR_DVREG_TRIM
   * @ingroup apiHalPHY230Const
   */
#  define DVREG_1_75V           (1)
  /** Constant DVREG_1_84V for sub-register @ref SR_DVREG_TRIM
   * @ingroup apiHalPHY230Const
   */
#  define DVREG_1_84V           (2)
  /** Constant DVREG_1_88V for sub-register @ref SR_DVREG_TRIM
```

```
   * @ingroup apiHalPHY230Const
   */
#  define DVREG_1_88V            (3)

/** Offset for register BATMON
 * @ingroup apiHalPHY230Reg
 */
#define RG_BATMON                      (0x11)
# define SR_reserved_11_1           0x11, 0xc0, 6
  /** Access parameters for sub-register BATMON_OK in register @ref RG_BATMON
   * @ingroup apiHalPHY230Sreg
   */
# define SR_BATMON_OK               0x11, 0x20, 5
  /** Access parameters for sub-register BATMON_HR in register @ref RG_BATMON
   * @ingroup apiHalPHY230Sreg
   */
# define SR_BATMON_HR               0x11, 0x10, 4
  /** Access parameters for sub-register BATMON_VTH in register @ref
RG_BATMON
   * @ingroup apiHalPHY230Sreg
   */
# define SR_BATMON_VTH              0x11, 0x0f, 0

/** Offset for register XOSC_CTRL
 * @ingroup apiHalPHY230Reg
 */
#define RG_XOSC_CTRL                   (0x12)
  /** Access parameters for sub-register XTAL_MODE in register @ref
RG_XOSC_CTRL
   * @ingroup apiHalPHY230Sreg
   */
# define SR_XTAL_MODE               0x12, 0xf0, 4
  /** Access parameters for sub-register XTAL_TRIM in register @ref
RG_XOSC_CTRL
   * @ingroup apiHalPHY230Sreg
   */
# define SR_XTAL_TRIM               0x12, 0x0f, 0

/** Offset for register FTN_CTRL
 * @ingroup apiHalPHY230Reg
 */
#define RG_FTN_CTRL                    (0x18)
  /** Access parameters for sub-register FTN_START in register @ref RG_FTN_CTRL
   * @ingroup apiHalPHY230Sreg
   */
# define SR_FTN_START               0x18, 0x80, 7
# define SR_reserved_18_2            0x18, 0x40, 6
  /** Access parameters for sub-register FTNV in register @ref RG_FTN_CTRL
    * @ingroup apiHalPHY230Sreg
    */
# define SR_FTNV                     0x18, 0x3f, 0

/** Offset for register PLL_CF
 * @ingroup apiHalPHY230Reg
 */
#define RG_PLL_CF                    (0x1a)
  /** Access parameters for sub-register PLL_CF_START in register @ref RG_PLL_CF
    * @ingroup apiHalPHY230Sreg
    */
# define SR_PLL_CF_START             0x1a, 0x80, 7
# define SR_reserved_1a_2            0x1a, 0x70, 4
  /** Access parameters for sub-register PLL_CF in register @ref RG_PLL_CF
    * @ingroup apiHalPHY230Sreg
    */
# define SR_PLL_CF                   0x1a, 0x0f, 0

/** Offset for register PLL_DCU
 * @ingroup apiHalPHY230Reg
 */
#define RG_PLL_DCU                   (0x1b)
  /** Access parameters for sub-register PLL_DCU_START in register @ref
RG_PLL_DCU
    * @ingroup apiHalPHY230Sreg
    */
# define SR_PLL_DCU_START            0x1b, 0x80, 7
# define SR_reserved_1b_2            0x1b, 0x40, 6
  /** Access parameters for sub-register PLL_DCUW in register @ref RG_PLL_DCU
    * @ingroup apiHalPHY230Sreg
    */
# define SR_PLL_DCUW                 0x1b, 0x3f, 0

/** Offset for register PART_NUM
 * @ingroup apiHalPHY230Reg
 */
#define RG_PART_NUM                  (0x1c)
  /** Access parameters for sub-register PART_NUM in register @ref RG_PART_NUM
    * @ingroup apiHalPHY230Sreg
    */
# define SR_PART_NUM                 0x1c, 0xff, 0
   /** Constant RF230 for sub-register @ref SR_PART_NUM
     * @ingroup apiHalPHY230Const
     */
```

```
#  define RF230                    (2)

/** Offset for register VERSION_NUM
 * @ingroup apiHalPHY230Reg
 */
#define RG_VERSION_NUM                  (0x1d)
  /** Access parameters for sub-register VERSION_NUM in register @ref
RG_VERSION_NUM
   * @ingroup apiHalPHY230Sreg
   */
# define SR_VERSION_NUM              0x1d, 0xff, 0

/** Offset for register MAN_ID_0
 * @ingroup apiHalPHY230Reg
 */
#define RG_MAN_ID_0                   (0x1e)
  /** Access parameters for sub-register MAN_ID_0 in register @ref RG_MAN_ID_0
   * @ingroup apiHalPHY230Sreg
   */
# define SR_MAN_ID_0                0x1e, 0xff, 0

/** Offset for register MAN_ID_1
 * @ingroup apiHalPHY230Reg
 */
#define RG_MAN_ID_1                   (0x1f)
  /** Access parameters for sub-register MAN_ID_1 in register @ref RG_MAN_ID_1
   * @ingroup apiHalPHY230Sreg
   */
# define SR_MAN_ID_1                0x1f, 0xff, 0

/** Offset for register SHORT_ADDR_0
 * @ingroup apiHalPHY230Reg
 */
#define RG_SHORT_ADDR_0                (0x20)
  /** Access parameters for sub-register SHORT_ADDR_0 in register @ref
RG_SHORT_ADDR_0
   * @ingroup apiHalPHY230Sreg
   */
# define SR_SHORT_ADDR_0            0x20, 0xff, 0

/** Offset for register SHORT_ADDR_1
 * @ingroup apiHalPHY230Reg
 */
#define RG_SHORT_ADDR_1                (0x21)
  /** Access parameters for sub-register SHORT_ADDR_1 in register @ref
RG_SHORT_ADDR_1
```

```
   * @ingroup apiHalPHY230Sreg
   */
# define SR_SHORT_ADDR_1            0x21, 0xff, 0

/** Offset for register PAN_ID_0
 * @ingroup apiHalPHY230Reg
 */
#define RG_PAN_ID_0                 (0x22)
  /** Access parameters for sub-register PAN_ID_0 in register @ref RG_PAN_ID_0
   * @ingroup apiHalPHY230Sreg
   */
# define SR_PAN_ID_0                0x22, 0xff, 0

/** Offset for register PAN_ID_1
 * @ingroup apiHalPHY230Reg
 */
#define RG_PAN_ID_1                 (0x23)
  /** Access parameters for sub-register PAN_ID_1 in register @ref RG_PAN_ID_1
   * @ingroup apiHalPHY230Sreg
   */
# define SR_PAN_ID_1                0x23, 0xff, 0

/** Offset for register IEEE_ADDR_0
 * @ingroup apiHalPHY230Reg
 */
#define RG_IEEE_ADDR_0              (0x24)
  /** Access parameters for sub-register IEEE_ADDR_0 in register @ref
RG_IEEE_ADDR_0
   * @ingroup apiHalPHY230Sreg
   */
# define SR_IEEE_ADDR_0             0x24, 0xff, 0

/** Offset for register IEEE_ADDR_1
 * @ingroup apiHalPHY230Reg
 */
#define RG_IEEE_ADDR_1              (0x25)
  /** Access parameters for sub-register IEEE_ADDR_1 in register @ref
RG_IEEE_ADDR_1
   * @ingroup apiHalPHY230Sreg
   */
# define SR_IEEE_ADDR_1             0x25, 0xff, 0

/** Offset for register IEEE_ADDR_2
 * @ingroup apiHalPHY230Reg
 */
#define RG_IEEE_ADDR_2              (0x26)
```

```
  /** Access parameters for sub-register IEEE_ADDR_2 in register @ref
RG_IEEE_ADDR_2
   * @ingroup apiHalPHY230Sreg
   */
# define SR_IEEE_ADDR_2              0x26, 0xff, 0

/** Offset for register IEEE_ADDR_3
 * @ingroup apiHalPHY230Reg
 */
#define RG_IEEE_ADDR_3                 (0x27)
  /** Access parameters for sub-register IEEE_ADDR_3 in register @ref
RG_IEEE_ADDR_3
   * @ingroup apiHalPHY230Sreg
   */
# define SR_IEEE_ADDR_3              0x27, 0xff, 0

/** Offset for register IEEE_ADDR_4
 * @ingroup apiHalPHY230Reg
 */
#define RG_IEEE_ADDR_4                 (0x28)
  /** Access parameters for sub-register IEEE_ADDR_4 in register @ref
RG_IEEE_ADDR_4
   * @ingroup apiHalPHY230Sreg
   */
# define SR_IEEE_ADDR_4              0x28, 0xff, 0

/** Offset for register IEEE_ADDR_5
 * @ingroup apiHalPHY230Reg
 */
#define RG_IEEE_ADDR_5                 (0x29)
  /** Access parameters for sub-register IEEE_ADDR_5 in register @ref
RG_IEEE_ADDR_5
   * @ingroup apiHalPHY230Sreg
   */
# define SR_IEEE_ADDR_5              0x29, 0xff, 0

/** Offset for register IEEE_ADDR_6
 * @ingroup apiHalPHY230Reg
 */
#define RG_IEEE_ADDR_6                 (0x2a)
  /** Access parameters for sub-register IEEE_ADDR_6 in register @ref
RG_IEEE_ADDR_6
   * @ingroup apiHalPHY230Sreg
   */
# define SR_IEEE_ADDR_6              0x2a, 0xff, 0
```

```
/** Offset for register IEEE_ADDR_7
 * @ingroup apiHalPHY230Reg
 */
#define RG_IEEE_ADDR_7                 (0x2b)
  /** Access parameters for sub-register IEEE_ADDR_7 in register @ref
RG_IEEE_ADDR_7
   * @ingroup apiHalPHY230Sreg
   */
# define SR_IEEE_ADDR_7              0x2b, 0xff, 0

/** Offset for register XAH_CTRL
 * @ingroup apiHalPHY230Reg
 */
#define RG_XAH_CTRL                    (0x2c)
  /** Access parameters for sub-register MAX_FRAME_RETRIES in register @ref
RG_XAH_CTRL
   * @ingroup apiHalPHY230Sreg
   */
# define SR_MAX_FRAME_RETRIES        0x2c, 0xf0, 4
  /** Access parameters for sub-register MAX_CSMA_RETRIES in register @ref
RG_XAH_CTRL
   * @ingroup apiHalPHY230Sreg
   */
# define SR_MAX_CSMA_RETRIES         0x2c, 0x0e, 1
# define SR_reserved_2c_3            0x2c, 0x01, 0

/** Offset for register CSMA_SEED_0
 * @ingroup apiHalPHY230Reg
 */
#define RG_CSMA_SEED_0                 (0x2d)
  /** Access parameters for sub-register CSMA_SEED_0 in register @ref
RG_CSMA_SEED_0
   * @ingroup apiHalPHY230Sreg
   */
# define SR_CSMA_SEED_0              0x2d, 0xff, 0

/** Offset for register CSMA_SEED_1
 * @ingroup apiHalPHY230Reg
 */
#define RG_CSMA_SEED_1                 (0x2e)
  /** Access parameters for sub-register MIN_BE in register @ref RG_CSMA_SEED_1
   * @ingroup apiHalPHY230Sreg
   */
# define SR_MIN_BE                   0x2e, 0xc0, 6
# define SR_reserved_2e_2            0x2e, 0x30, 4
```

```
  /** Access parameters for sub-register I_AM_COORD in register @ref
RG_CSMA_SEED_1
   * @ingroup apiHalPHY230Sreg
   */
# define SR_I_AM_COORD              0x2e, 0x08, 3
  /** Access parameters for sub-register CSMA_SEED_1 in register @ref
RG_CSMA_SEED_1
   * @ingroup apiHalPHY230Sreg
   */
# define SR_CSMA_SEED_1             0x2e, 0x07, 0

#endif /* PHY230_REGISTERMAP_EXTERNAL_H */
```

Com.c

```
/*
 * This file is autogenerated from regxml2include.py
 * Do not modify it, changes will be lost after rebuild.
 */
/**
 * @file
 * generated register definition file
 * Inputfile:  phy230_registermap_external.xml
 * Version:    1.9    for external use
 * Created at: Thu Jan 25 17:07:33 2007
 *
 */
/*
 * Copyright (c) 2006, Atmel Corporation All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions are met:
 *
 * 1. Redistributions of source code must retain the above copyright notice,
 * this list of conditions and the following disclaimer.
 *
 * 2. 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.
 *
 * 3. The name of ATMEL may not be used to endorse or promote products derived
 * from this software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY ATMEL ``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
EXPRESSLY AND
 * SPECIFICALLY DISCLAIMED. IN NO EVENT SHALL ATMEL 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.
 */
/**
 *  @author
 *      Atmel Corporation: http://www.atmel.com
 *      Support email: avr@atmel.com
 */

#ifndef PHY230_REGISTERMAP_EXTERNAL_H
#define PHY230_REGISTERMAP_EXTERNAL_H

#define HAVE_REGISTER_MAP (1)
/** Offset for register TRX_STATUS
 * @ingroup apiHalPHY230Reg
 */
#define RG_TRX_STATUS                   (0x01)
  /** Access parameters for sub-register CCA_DONE in register @ref
RG_TRX_STATUS
   * @ingroup apiHalPHY230Sreg
   */
# define SR_CCA_DONE                0x01, 0x80, 7
  /** Access parameters for sub-register CCA_STATUS in register @ref
RG_TRX_STATUS
   * @ingroup apiHalPHY230Sreg
   */
# define SR_CCA_STATUS              0x01, 0x40, 6
# define SR_reserved_01_3           0x01, 0x20, 5
  /** Access parameters for sub-register TRX_STATUS in register @ref
RG_TRX_STATUS
```

```
  * @ingroup apiHalPHY230Sreg
  */
# define SR_TRX_STATUS                0x01, 0x1f, 0
  /** Constant P_ON for sub-register @ref SR_TRX_STATUS
   * @ingroup apiHalPHY230Const
   */
#  define P_ON                   (0)
  /** Constant BUSY_RX for sub-register @ref SR_TRX_STATUS
   * @ingroup apiHalPHY230Const
   */
#  define BUSY_RX                 (1)
  /** Constant BUSY_TX for sub-register @ref SR_TRX_STATUS
   * @ingroup apiHalPHY230Const
   */
#  define BUSY_TX                 (2)
  /** Constant RX_ON for sub-register @ref SR_TRX_STATUS
   * @ingroup apiHalPHY230Const
   */
#  define RX_ON                   (6)
  /** Constant TRX_OFF for sub-register @ref SR_TRX_STATUS
   * @ingroup apiHalPHY230Const
   */
#  define TRX_OFF                 (8)
  /** Constant PLL_ON for sub-register @ref SR_TRX_STATUS
   * @ingroup apiHalPHY230Const
   */
#  define PLL_ON                  (9)
  /** Constant SLEEP for sub-register @ref SR_TRX_STATUS
   * @ingroup apiHalPHY230Const
   */
#  define SLEEP                   (15)
  /** Constant BUSY_RX_AACK for sub-register @ref SR_TRX_STATUS
   * @ingroup apiHalPHY230Const
   */
#  define BUSY_RX_AACK            (17)
  /** Constant BUSY_TX_ARET for sub-register @ref SR_TRX_STATUS
   * @ingroup apiHalPHY230Const
   */
#  define BUSY_TX_ARET            (18)
  /** Constant RX_AACK_ON for sub-register @ref SR_TRX_STATUS
   * @ingroup apiHalPHY230Const
   */
#  define RX_AACK_ON              (22)
  /** Constant TX_ARET_ON for sub-register @ref SR_TRX_STATUS
   * @ingroup apiHalPHY230Const
   */
```

```
#  define TX_ARET_ON              (25)
  /** Constant RX_ON_NOCLK for sub-register @ref SR_TRX_STATUS
   * @ingroup apiHalPHY230Const
   */
#  define RX_ON_NOCLK             (28)
  /** Constant RX_AACK_ON_NOCLK for sub-register @ref SR_TRX_STATUS
   * @ingroup apiHalPHY230Const
   */
#  define RX_AACK_ON_NOCLK         (29)
  /** Constant BUSY_RX_AACK_NOCLK for sub-register @ref SR_TRX_STATUS
   * @ingroup apiHalPHY230Const
   */
#  define BUSY_RX_AACK_NOCLK       (30)

/** Offset for register TRX_STATE
 * @ingroup apiHalPHY230Reg
 */
#define RG_TRX_STATE                   (0x02)
 /** Access parameters for sub-register TRAC_STATUS in register @ref
RG_TRX_STATE
  * @ingroup apiHalPHY230Sreg
  */
# define SR_TRAC_STATUS              0x02, 0xe0, 5
 /** Access parameters for sub-register TRX_CMD in register @ref RG_TRX_STATE
  * @ingroup apiHalPHY230Sreg
  */
# define SR_TRX_CMD                 0x02, 0x1f, 0
  /** Constant CMD_NOP for sub-register @ref SR_TRX_CMD
   * @ingroup apiHalPHY230Const
   */
#  define CMD_NOP                (0)
  /** Constant CMD_TX_START for sub-register @ref SR_TRX_CMD
   * @ingroup apiHalPHY230Const
   */
#  define CMD_TX_START            (2)
  /** Constant CMD_FORCE_TRX_OFF for sub-register @ref SR_TRX_CMD
   * @ingroup apiHalPHY230Const
   */
#  define CMD_FORCE_TRX_OFF        (3)
  /** Constant CMD_RX_ON for sub-register @ref SR_TRX_CMD
   * @ingroup apiHalPHY230Const
   */
#  define CMD_RX_ON              (6)
  /** Constant CMD_TRX_OFF for sub-register @ref SR_TRX_CMD
   * @ingroup apiHalPHY230Const
   */
```

```
#  define CMD_TRX_OFF             (8)
  /** Constant CMD_PLL_ON for sub-register @ref SR_TRX_CMD
   * @ingroup apiHalPHY230Const
   */
#  define CMD_PLL_ON              (9)
  /** Constant CMD_RX_AACK_ON for sub-register @ref SR_TRX_CMD
   * @ingroup apiHalPHY230Const
   */
#  define CMD_RX_AACK_ON          (22)
  /** Constant CMD_TX_ARET_ON for sub-register @ref SR_TRX_CMD
   * @ingroup apiHalPHY230Const
   */
#  define CMD_TX_ARET_ON          (25)

/** Offset for register TRX_CTRL_0
 * @ingroup apiHalPHY230Reg
 */
#define RG_TRX_CTRL_0                   (0x03)
 /** Access parameters for sub-register PAD_IO in register @ref RG_TRX_CTRL_0
  * @ingroup apiHalPHY230Sreg
  */
# define SR_PAD_IO                  0x03, 0xc0, 6
 /** Access parameters for sub-register PAD_IO_CLKM in register @ref
RG_TRX_CTRL_0
  * @ingroup apiHalPHY230Sreg
  */
# define SR_PAD_IO_CLKM             0x03, 0x30, 4
  /** Constant CLKM_2mA for sub-register @ref SR_PAD_IO_CLKM
   * @ingroup apiHalPHY230Const
   */
#  define CLKM_2mA                (0)
  /** Constant CLKM_4mA for sub-register @ref SR_PAD_IO_CLKM
   * @ingroup apiHalPHY230Const
   */
#  define CLKM_4mA                (1)
  /** Constant CLKM_6mA for sub-register @ref SR_PAD_IO_CLKM
   * @ingroup apiHalPHY230Const
   */
#  define CLKM_6mA                (2)
  /** Constant CLKM_8mA for sub-register @ref SR_PAD_IO_CLKM
   * @ingroup apiHalPHY230Const
   */
#  define CLKM_8mA                (3)
 /** Access parameters for sub-register CLKM_SHA_SEL in register @ref
RG_TRX_CTRL_0
  * @ingroup apiHalPHY230Sreg
```

```
   */
# define SR_CLKM_SHA_SEL            0x03, 0x08, 3
  /** Access parameters for sub-register CLKM_CTRL in register @ref
RG_TRX_CTRL_0
   * @ingroup apiHalPHY230Sreg
   */
# define SR_CLKM_CTRL               0x03, 0x07, 0
   /** Constant CLKM_no_clock for sub-register @ref SR_CLKM_CTRL
    * @ingroup apiHalPHY230Const
    */
#  define CLKM_no_clock           (0)
   /** Constant CLKM_1MHz for sub-register @ref SR_CLKM_CTRL
    * @ingroup apiHalPHY230Const
    */
#  define CLKM_1MHz               (1)
   /** Constant CLKM_2MHz for sub-register @ref SR_CLKM_CTRL
    * @ingroup apiHalPHY230Const
    */
#  define CLKM_2MHz               (2)
   /** Constant CLKM_4MHz for sub-register @ref SR_CLKM_CTRL
    * @ingroup apiHalPHY230Const
    */
#  define CLKM_4MHz               (3)
   /** Constant CLKM_8MHz for sub-register @ref SR_CLKM_CTRL
    * @ingroup apiHalPHY230Const
    */
#  define CLKM_8MHz               (4)
   /** Constant CLKM_16MHz for sub-register @ref SR_CLKM_CTRL
    * @ingroup apiHalPHY230Const
    */
#  define CLKM_16MHz               (5)

/** Offset for register PHY_TX_PWR
 * @ingroup apiHalPHY230Reg
 */
#define RG_PHY_TX_PWR                   (0x05)
  /** Access parameters for sub-register TX_AUTO_CRC_ON in register @ref
RG_PHY_TX_PWR
   * @ingroup apiHalPHY230Sreg
   */
# define SR_TX_AUTO_CRC_ON           0x05, 0x80, 7
# define SR_reserved_05_2            0x05, 0x70, 4
  /** Access parameters for sub-register TX_PWR in register @ref RG_PHY_TX_PWR
   * @ingroup apiHalPHY230Sreg
   */
# define SR_TX_PWR                   0x05, 0x0f, 0
```

```
/** Offset for register PHY_RSSI
 * @ingroup apiHalPHY230Reg
 */
#define RG_PHY_RSSI                    (0x06)
# define SR_reserved_06_1            0x06, 0xe0, 5
  /** Access parameters for sub-register RSSI in register @ref RG_PHY_RSSI
   * @ingroup apiHalPHY230Sreg
   */
# define SR_RSSI                     0x06, 0x1f, 0

/** Offset for register PHY_ED_LEVEL
 * @ingroup apiHalPHY230Reg
 */
#define RG_PHY_ED_LEVEL                (0x07)
  /** Access parameters for sub-register ED_LEVEL in register @ref
RG_PHY_ED_LEVEL
   * @ingroup apiHalPHY230Sreg
   */
# define SR_ED_LEVEL                 0x07, 0xff, 0

/** Offset for register PHY_CC_CCA
 * @ingroup apiHalPHY230Reg
 */
#define RG_PHY_CC_CCA                  (0x08)
  /** Access parameters for sub-register CCA_REQUEST in register @ref
RG_PHY_CC_CCA
   * @ingroup apiHalPHY230Sreg
   */
# define SR_CCA_REQUEST              0x08, 0x80, 7
  /** Access parameters for sub-register CCA_MODE in register @ref
RG_PHY_CC_CCA
   * @ingroup apiHalPHY230Sreg
   */
# define SR_CCA_MODE                 0x08, 0x60, 5
  /** Access parameters for sub-register CHANNEL in register @ref RG_PHY_CC_CCA
   * @ingroup apiHalPHY230Sreg
   */
# define SR_CHANNEL                  0x08, 0x1f, 0

/** Offset for register CCA_THRES
 * @ingroup apiHalPHY230Reg
 */
#define RG_CCA_THRES                   (0x09)
  /** Access parameters for sub-register CCA_CS_THRES in register @ref
RG_CCA_THRES
```

```
   * @ingroup apiHalPHY230Sreg
   */
# define SR_CCA_CS_THRES             0x09, 0xf0, 4
  /** Access parameters for sub-register CCA_ED_THRES in register @ref
RG_CCA_THRES
   * @ingroup apiHalPHY230Sreg
   */
# define SR_CCA_ED_THRES             0x09, 0x0f, 0

/** Offset for register IRQ_MASK
 * @ingroup apiHalPHY230Reg
 */
#define RG_IRQ_MASK                  (0x0e)
  /** Access parameters for sub-register IRQ_MASK in register @ref RG_IRQ_MASK
   * @ingroup apiHalPHY230Sreg
   */
# define SR_IRQ_MASK                 0x0e, 0xff, 0

/** Offset for register IRQ_STATUS
 * @ingroup apiHalPHY230Reg
 */
#define RG_IRQ_STATUS                (0x0f)
  /** Access parameters for sub-register IRQ_7_BAT_LOW in register @ref
RG_IRQ_STATUS
   * @ingroup apiHalPHY230Sreg
   */
# define SR_IRQ_7_BAT_LOW            0x0f, 0x80, 7
  /** Access parameters for sub-register IRQ_6_TRX_UR in register @ref
RG_IRQ_STATUS
   * @ingroup apiHalPHY230Sreg
   */
# define SR_IRQ_6_TRX_UR             0x0f, 0x40, 6
  /** Access parameters for sub-register IRQ_5 in register @ref RG_IRQ_STATUS
   * @ingroup apiHalPHY230Sreg
   */
# define SR_IRQ_5                    0x0f, 0x20, 5
  /** Access parameters for sub-register IRQ_4 in register @ref RG_IRQ_STATUS
   * @ingroup apiHalPHY230Sreg
   */
# define SR_IRQ_4                    0x0f, 0x10, 4
  /** Access parameters for sub-register IRQ_3_TRX_END in register @ref
RG_IRQ_STATUS
   * @ingroup apiHalPHY230Sreg
   */
# define SR_IRQ_3_TRX_END            0x0f, 0x08, 3
```

```
 /** Access parameters for sub-register IRQ_2_RX_START in register @ref
RG_IRQ_STATUS
   * @ingroup apiHalPHY230Sreg
   */
# define SR_IRQ_2_RX_START           0x0f, 0x04, 2
 /** Access parameters for sub-register IRQ_1_PLL_UNLOCK in register @ref
RG_IRQ_STATUS
   * @ingroup apiHalPHY230Sreg
   */
# define SR_IRQ_1_PLL_UNLOCK         0x0f, 0x02, 1
 /** Access parameters for sub-register IRQ_0_PLL_LOCK in register @ref
RG_IRQ_STATUS
   * @ingroup apiHalPHY230Sreg
   */
# define SR_IRQ_0_PLL_LOCK           0x0f, 0x01, 0

/** Offset for register VREG_CTRL
 * @ingroup apiHalPHY230Reg
 */
#define RG_VREG_CTRL                 (0x10)
 /** Access parameters for sub-register AVREG_EXT in register @ref
RG_VREG_CTRL
   * @ingroup apiHalPHY230Sreg
   */
# define SR_AVREG_EXT                0x10, 0x80, 7
 /** Access parameters for sub-register AVDD_OK in register @ref RG_VREG_CTRL
   * @ingroup apiHalPHY230Sreg
   */
# define SR_AVDD_OK                  0x10, 0x40, 6
 /** Access parameters for sub-register AVREG_TRIM in register @ref
RG_VREG_CTRL
   * @ingroup apiHalPHY230Sreg
   */
# define SR_AVREG_TRIM               0x10, 0x30, 4
  /** Constant AVREG_1_80V for sub-register @ref SR_AVREG_TRIM
    * @ingroup apiHalPHY230Const
    */
#  define AVREG_1_80V           (0)
  /** Constant AVREG_1_75V for sub-register @ref SR_AVREG_TRIM
    * @ingroup apiHalPHY230Const
    */
#  define AVREG_1_75V           (1)
  /** Constant AVREG_1_84V for sub-register @ref SR_AVREG_TRIM
    * @ingroup apiHalPHY230Const
    */
#  define AVREG_1_84V           (2)
```

```
   /** Constant AVREG_1_88V for sub-register @ref SR_AVREG_TRIM
    * @ingroup apiHalPHY230Const
    */
#  define AVREG_1_88V             (3)
  /** Access parameters for sub-register DVREG_EXT in register @ref
RG_VREG_CTRL
   * @ingroup apiHalPHY230Sreg
   */
# define SR_DVREG_EXT                0x10, 0x08, 3
  /** Access parameters for sub-register DVDD_OK in register @ref RG_VREG_CTRL
   * @ingroup apiHalPHY230Sreg
   */
# define SR_DVDD_OK                  0x10, 0x04, 2
  /** Access parameters for sub-register DVREG_TRIM in register @ref
RG_VREG_CTRL
   * @ingroup apiHalPHY230Sreg
   */
# define SR_DVREG_TRIM               0x10, 0x03, 0
   /** Constant DVREG_1_80V for sub-register @ref SR_DVREG_TRIM
    * @ingroup apiHalPHY230Const
    */
#  define DVREG_1_80V             (0)
   /** Constant DVREG_1_75V for sub-register @ref SR_DVREG_TRIM
    * @ingroup apiHalPHY230Const
    */
#  define DVREG_1_75V             (1)
   /** Constant DVREG_1_84V for sub-register @ref SR_DVREG_TRIM
    * @ingroup apiHalPHY230Const
    */
#  define DVREG_1_84V             (2)
   /** Constant DVREG_1_88V for sub-register @ref SR_DVREG_TRIM
    * @ingroup apiHalPHY230Const
    */
#  define DVREG_1_88V             (3)

/** Offset for register BATMON
 * @ingroup apiHalPHY230Reg
 */
#define RG_BATMON                    (0x11)
# define SR_reserved_11_1            0x11, 0xc0, 6
  /** Access parameters for sub-register BATMON_OK in register @ref RG_BATMON
   * @ingroup apiHalPHY230Sreg
   */
# define SR_BATMON_OK                0x11, 0x20, 5
  /** Access parameters for sub-register BATMON_HR in register @ref RG_BATMON
   * @ingroup apiHalPHY230Sreg
   */
# define SR_BATMON_HR                0x11, 0x10, 4
  /** Access parameters for sub-register BATMON_VTH in register @ref
RG_BATMON
   * @ingroup apiHalPHY230Sreg
   */
# define SR_BATMON_VTH               0x11, 0x0f, 0

/** Offset for register XOSC_CTRL
 * @ingroup apiHalPHY230Reg
 */
#define RG_XOSC_CTRL                 (0x12)
  /** Access parameters for sub-register XTAL_MODE in register @ref
RG_XOSC_CTRL
   * @ingroup apiHalPHY230Sreg
   */
# define SR_XTAL_MODE                0x12, 0xf0, 4
  /** Access parameters for sub-register XTAL_TRIM in register @ref
RG_XOSC_CTRL
   * @ingroup apiHalPHY230Sreg
   */
# define SR_XTAL_TRIM                0x12, 0x0f, 0

/** Offset for register FTN_CTRL
 * @ingroup apiHalPHY230Reg
 */
#define RG_FTN_CTRL                  (0x18)
  /** Access parameters for sub-register FTN_START in register @ref RG_FTN_CTRL
   * @ingroup apiHalPHY230Sreg
   */
# define SR_FTN_START                0x18, 0x80, 7
# define SR_reserved_18_2            0x18, 0x40, 6
  /** Access parameters for sub-register FTNV in register @ref RG_FTN_CTRL
   * @ingroup apiHalPHY230Sreg
   */
# define SR_FTNV                     0x18, 0x3f, 0

/** Offset for register PLL_CF
 * @ingroup apiHalPHY230Reg
 */
#define RG_PLL_CF                    (0x1a)
  /** Access parameters for sub-register PLL_CF_START in register @ref RG_PLL_CF
   * @ingroup apiHalPHY230Sreg
   */
# define SR_PLL_CF_START             0x1a, 0x80, 7
# define SR_reserved_1a_2            0x1a, 0x70, 4
```

```
  /** Access parameters for sub-register PLL_CF in register @ref RG_PLL_CF
   * @ingroup apiHalPHY230Sreg
   */
# define SR_PLL_CF                  0x1a, 0x0f, 0

/** Offset for register PLL_DCU
 * @ingroup apiHalPHY230Reg
 */
#define RG_PLL_DCU                      (0x1b)
  /** Access parameters for sub-register PLL_DCU_START in register @ref
RG_PLL_DCU
   * @ingroup apiHalPHY230Sreg
   */
# define SR_PLL_DCU_START            0x1b, 0x80, 7
# define SR_reserved_1b_2            0x1b, 0x40, 6
  /** Access parameters for sub-register PLL_DCUW in register @ref RG_PLL_DCU
   * @ingroup apiHalPHY230Sreg
   */
# define SR_PLL_DCUW                 0x1b, 0x3f, 0

/** Offset for register PART_NUM
 * @ingroup apiHalPHY230Reg
 */
#define RG_PART_NUM                     (0x1c)
  /** Access parameters for sub-register PART_NUM in register @ref RG_PART_NUM
   * @ingroup apiHalPHY230Sreg
   */
# define SR_PART_NUM                 0x1c, 0xff, 0
   /** Constant RF230 for sub-register @ref SR_PART_NUM
    * @ingroup apiHalPHY230Const
    */
#  define RF230                   (2)

/** Offset for register VERSION_NUM
 * @ingroup apiHalPHY230Reg
 */
#define RG_VERSION_NUM                  (0x1d)
  /** Access parameters for sub-register VERSION_NUM in register @ref
RG_VERSION_NUM
   * @ingroup apiHalPHY230Sreg
   */
# define SR_VERSION_NUM              0x1d, 0xff, 0

/** Offset for register MAN_ID_0
 * @ingroup apiHalPHY230Reg
 */
```

```
#define RG_MAN_ID_0                    (0x1e)
  /** Access parameters for sub-register MAN_ID_0 in register @ref RG_MAN_ID_0
   * @ingroup apiHalPHY230Sreg
   */
# define SR_MAN_ID_0                 0x1e, 0xff, 0

/** Offset for register MAN_ID_1
 * @ingroup apiHalPHY230Reg
 */
#define RG_MAN_ID_1                    (0x1f)
  /** Access parameters for sub-register MAN_ID_1 in register @ref RG_MAN_ID_1
   * @ingroup apiHalPHY230Sreg
   */
# define SR_MAN_ID_1                 0x1f, 0xff, 0

/** Offset for register SHORT_ADDR_0
 * @ingroup apiHalPHY230Reg
 */
#define RG_SHORT_ADDR_0                (0x20)
  /** Access parameters for sub-register SHORT_ADDR_0 in register @ref
RG_SHORT_ADDR_0
   * @ingroup apiHalPHY230Sreg
   */
# define SR_SHORT_ADDR_0             0x20, 0xff, 0

/** Offset for register SHORT_ADDR_1
 * @ingroup apiHalPHY230Reg
 */
#define RG_SHORT_ADDR_1                (0x21)
  /** Access parameters for sub-register SHORT_ADDR_1 in register @ref
RG_SHORT_ADDR_1
   * @ingroup apiHalPHY230Sreg
   */
# define SR_SHORT_ADDR_1             0x21, 0xff, 0

/** Offset for register PAN_ID_0
 * @ingroup apiHalPHY230Reg
 */
#define RG_PAN_ID_0                    (0x22)
  /** Access parameters for sub-register PAN_ID_0 in register @ref RG_PAN_ID_0
   * @ingroup apiHalPHY230Sreg
   */
# define SR_PAN_ID_0                 0x22, 0xff, 0

/** Offset for register PAN_ID_1
 * @ingroup apiHalPHY230Reg
 */
#define RG_PAN_ID_1                    (0x23)
  /** Access parameters for sub-register PAN_ID_1 in register @ref RG_PAN_ID_1
   * @ingroup apiHalPHY230Sreg
   */
# define SR_PAN_ID_1                 0x23, 0xff, 0

/** Offset for register IEEE_ADDR_0
 * @ingroup apiHalPHY230Reg
 */
#define RG_IEEE_ADDR_0                 (0x24)
  /** Access parameters for sub-register IEEE_ADDR_0 in register @ref
RG_IEEE_ADDR_0
   * @ingroup apiHalPHY230Sreg
   */
# define SR_IEEE_ADDR_0              0x24, 0xff, 0

/** Offset for register IEEE_ADDR_1
 * @ingroup apiHalPHY230Reg
 */
#define RG_IEEE_ADDR_1                 (0x25)
  /** Access parameters for sub-register IEEE_ADDR_1 in register @ref
RG_IEEE_ADDR_1
   * @ingroup apiHalPHY230Sreg
   */
# define SR_IEEE_ADDR_1              0x25, 0xff, 0

/** Offset for register IEEE_ADDR_2
 * @ingroup apiHalPHY230Reg
 */
#define RG_IEEE_ADDR_2                 (0x26)
  /** Access parameters for sub-register IEEE_ADDR_2 in register @ref
RG_IEEE_ADDR_2
   * @ingroup apiHalPHY230Sreg
   */
# define SR_IEEE_ADDR_2              0x26, 0xff, 0

/** Offset for register IEEE_ADDR_3
 * @ingroup apiHalPHY230Reg
 */
#define RG_IEEE_ADDR_3                 (0x27)
  /** Access parameters for sub-register IEEE_ADDR_3 in register @ref
RG_IEEE_ADDR_3
   * @ingroup apiHalPHY230Sreg
   */
# define SR_IEEE_ADDR_3              0x27, 0xff, 0
```

```
/** Offset for register IEEE_ADDR_4
 * @ingroup apiHalPHY230Reg
 */
#define RG_IEEE_ADDR_4                  (0x28)
  /** Access parameters for sub-register IEEE_ADDR_4 in register @ref
RG_IEEE_ADDR_4
   * @ingroup apiHalPHY230Sreg
   */
# define SR_IEEE_ADDR_4              0x28, 0xff, 0

/** Offset for register IEEE_ADDR_5
 * @ingroup apiHalPHY230Reg
 */
#define RG_IEEE_ADDR_5                  (0x29)
  /** Access parameters for sub-register IEEE_ADDR_5 in register @ref
RG_IEEE_ADDR_5
   * @ingroup apiHalPHY230Sreg
   */
# define SR_IEEE_ADDR_5              0x29, 0xff, 0

/** Offset for register IEEE_ADDR_6
 * @ingroup apiHalPHY230Reg
 */
#define RG_IEEE_ADDR_6                  (0x2a)
  /** Access parameters for sub-register IEEE_ADDR_6 in register @ref
RG_IEEE_ADDR_6
   * @ingroup apiHalPHY230Sreg
   */
# define SR_IEEE_ADDR_6              0x2a, 0xff, 0

/** Offset for register IEEE_ADDR_7
 * @ingroup apiHalPHY230Reg
 */
#define RG_IEEE_ADDR_7                  (0x2b)
  /** Access parameters for sub-register IEEE_ADDR_7 in register @ref
RG_IEEE_ADDR_7
   * @ingroup apiHalPHY230Sreg
   */
# define SR_IEEE_ADDR_7              0x2b, 0xff, 0

/** Offset for register XAH_CTRL
 * @ingroup apiHalPHY230Reg
 */
#define RG_XAH_CTRL                     (0x2c)
```

```
  /** Access parameters for sub-register MAX_FRAME_RETRIES in register @ref
RG_XAH_CTRL
   * @ingroup apiHalPHY230Sreg
   */
# define SR_MAX_FRAME_RETRIES        0x2c, 0xf0, 4
  /** Access parameters for sub-register MAX_CSMA_RETRIES in register @ref
RG_XAH_CTRL
   * @ingroup apiHalPHY230Sreg
   */
# define SR_MAX_CSMA_RETRIES         0x2c, 0x0e, 1
# define SR_reserved_2c_3            0x2c, 0x01, 0

/** Offset for register CSMA_SEED_0
 * @ingroup apiHalPHY230Reg
 */
#define RG_CSMA_SEED_0                  (0x2d)
  /** Access parameters for sub-register CSMA_SEED_0 in register @ref
RG_CSMA_SEED_0
   * @ingroup apiHalPHY230Sreg
   */
# define SR_CSMA_SEED_0              0x2d, 0xff, 0

/** Offset for register CSMA_SEED_1
 * @ingroup apiHalPHY230Reg
 */
#define RG_CSMA_SEED_1                  (0x2e)
  /** Access parameters for sub-register MIN_BE in register @ref RG_CSMA_SEED_1
   * @ingroup apiHalPHY230Sreg
   */
# define SR_MIN_BE                   0x2e, 0xc0, 6
# define SR_reserved_2e_2            0x2e, 0x30, 4
  /** Access parameters for sub-register I_AM_COORD in register @ref
RG_CSMA_SEED_1
   * @ingroup apiHalPHY230Sreg
   */
# define SR_I_AM_COORD               0x2e, 0x08, 3
  /** Access parameters for sub-register CSMA_SEED_1 in register @ref
RG_CSMA_SEED_1
   * @ingroup apiHalPHY230Sreg
   */
# define SR_CSMA_SEED_1              0x2e, 0x07, 0

#endif /* PHY230_REGISTERMAP_EXTERNAL_H */
```

Com.h

```
/*
 * This file is autogenerated from regxml2include.py
 * Do not modify it, changes will be lost after rebuild.
 */
/**
 * @file
 * generated register definition file
 * Inputfile:  phy230_registermap_external.xml
 * Version:    1.9    for external use
 * Created at: Thu Jan 25 17:07:33 2007
 *
 */
/*
 * Copyright (c) 2006, Atmel Corporation All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions are met:
 *
 * 1. Redistributions of source code must retain the above copyright notice,
 * this list of conditions and the following disclaimer.
 *
 * 2. 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.
 *
 * 3. The name of ATMEL may not be used to endorse or promote products derived
 * from this software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY ATMEL ``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
EXPRESSLY AND
 * SPECIFICALLY DISCLAIMED. IN NO EVENT SHALL ATMEL 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.
 */
/**
 *  @author
 *      Atmel Corporation: http://www.atmel.com
 *      Support email: avr@atmel.com
 */

#ifndef PHY230_REGISTERMAP_EXTERNAL_H
#define PHY230_REGISTERMAP_EXTERNAL_H

#define HAVE_REGISTER_MAP (1)
/** Offset for register TRX_STATUS
 * @ingroup apiHalPHY230Reg
 */
#define RG_TRX_STATUS                    (0x01)
  /** Access parameters for sub-register CCA_DONE in register @ref
RG_TRX_STATUS
   * @ingroup apiHalPHY230Sreg
   */
# define SR_CCA_DONE                  0x01, 0x80, 7
  /** Access parameters for sub-register CCA_STATUS in register @ref
RG_TRX_STATUS
   * @ingroup apiHalPHY230Sreg
   */
# define SR_CCA_STATUS                0x01, 0x40, 6
# define SR_reserved_01_3             0x01, 0x20, 5
  /** Access parameters for sub-register TRX_STATUS in register @ref
RG_TRX_STATUS
   * @ingroup apiHalPHY230Sreg
   */
# define SR_TRX_STATUS                0x01, 0x1f, 0
   /** Constant P_ON for sub-register @ref SR_TRX_STATUS
    * @ingroup apiHalPHY230Const
    */
#  define P_ON                   (0)
   /** Constant BUSY_RX for sub-register @ref SR_TRX_STATUS
    * @ingroup apiHalPHY230Const
    */
#  define BUSY_RX                (1)
   /** Constant BUSY_TX for sub-register @ref SR_TRX_STATUS
    * @ingroup apiHalPHY230Const
    */
#  define BUSY_TX                (2)
   /** Constant RX_ON for sub-register @ref SR_TRX_STATUS
```

```
    * @ingroup apiHalPHY230Const
    */
#  define RX_ON                   (6)
   /** Constant TRX_OFF for sub-register @ref SR_TRX_STATUS
    * @ingroup apiHalPHY230Const
    */
#  define TRX_OFF                 (8)
   /** Constant PLL_ON for sub-register @ref SR_TRX_STATUS
    * @ingroup apiHalPHY230Const
    */
#  define PLL_ON                  (9)
   /** Constant SLEEP for sub-register @ref SR_TRX_STATUS
    * @ingroup apiHalPHY230Const
    */
#  define SLEEP                   (15)
   /** Constant BUSY_RX_AACK for sub-register @ref SR_TRX_STATUS
    * @ingroup apiHalPHY230Const
    */
#  define BUSY_RX_AACK            (17)
   /** Constant BUSY_TX_ARET for sub-register @ref SR_TRX_STATUS
    * @ingroup apiHalPHY230Const
    */
#  define BUSY_TX_ARET            (18)
   /** Constant RX_AACK_ON for sub-register @ref SR_TRX_STATUS
    * @ingroup apiHalPHY230Const
    */
#  define RX_AACK_ON              (22)
   /** Constant TX_ARET_ON for sub-register @ref SR_TRX_STATUS
    * @ingroup apiHalPHY230Const
    */
#  define TX_ARET_ON              (25)
   /** Constant RX_ON_NOCLK for sub-register @ref SR_TRX_STATUS
    * @ingroup apiHalPHY230Const
    */
#  define RX_ON_NOCLK             (28)
   /** Constant RX_AACK_ON_NOCLK for sub-register @ref SR_TRX_STATUS
    * @ingroup apiHalPHY230Const
    */
#  define RX_AACK_ON_NOCLK        (29)
   /** Constant BUSY_RX_AACK_NOCLK for sub-register @ref SR_TRX_STATUS
    * @ingroup apiHalPHY230Const
    */
#  define BUSY_RX_AACK_NOCLK      (30)

/** Offset for register TRX_STATE
 * @ingroup apiHalPHY230Reg
 */
#define RG_TRX_STATE                    (0x02)
  /** Access parameters for sub-register TRAC_STATUS in register @ref
RG_TRX_STATE
   * @ingroup apiHalPHY230Sreg
   */
# define SR_TRAC_STATUS              0x02, 0xe0, 5
  /** Access parameters for sub-register TRX_CMD in register @ref RG_TRX_STATE
   * @ingroup apiHalPHY230Sreg
   */
# define SR_TRX_CMD                  0x02, 0x1f, 0
   /** Constant CMD_NOP for sub-register @ref SR_TRX_CMD
    * @ingroup apiHalPHY230Const
    */
#  define CMD_NOP                (0)
   /** Constant CMD_TX_START for sub-register @ref SR_TRX_CMD
    * @ingroup apiHalPHY230Const
    */
#  define CMD_TX_START            (2)
   /** Constant CMD_FORCE_TRX_OFF for sub-register @ref SR_TRX_CMD
    * @ingroup apiHalPHY230Const
    */
#  define CMD_FORCE_TRX_OFF        (3)
   /** Constant CMD_RX_ON for sub-register @ref SR_TRX_CMD
    * @ingroup apiHalPHY230Const
    */
#  define CMD_RX_ON               (6)
   /** Constant CMD_TRX_OFF for sub-register @ref SR_TRX_CMD
    * @ingroup apiHalPHY230Const
    */
#  define CMD_TRX_OFF              (8)
   /** Constant CMD_PLL_ON for sub-register @ref SR_TRX_CMD
    * @ingroup apiHalPHY230Const
    */
#  define CMD_PLL_ON              (9)
   /** Constant CMD_RX_AACK_ON for sub-register @ref SR_TRX_CMD
    * @ingroup apiHalPHY230Const
    */
#  define CMD_RX_AACK_ON          (22)
   /** Constant CMD_TX_ARET_ON for sub-register @ref SR_TRX_CMD
    * @ingroup apiHalPHY230Const
    */
#  define CMD_TX_ARET_ON          (25)

/** Offset for register TRX_CTRL_0
 * @ingroup apiHalPHY230Reg
 */
#define RG_TRX_CTRL_0                   (0x03)
  /** Access parameters for sub-register PAD_IO in register @ref RG_TRX_CTRL_0
   * @ingroup apiHalPHY230Sreg
   */
# define SR_PAD_IO                   0x03, 0xc0, 6
  /** Access parameters for sub-register PAD_IO_CLKM in register @ref
RG_TRX_CTRL_0
   * @ingroup apiHalPHY230Sreg
   */
# define SR_PAD_IO_CLKM              0x03, 0x30, 4
   /** Constant CLKM_2mA for sub-register @ref SR_PAD_IO_CLKM
    * @ingroup apiHalPHY230Const
    */
#  define CLKM_2mA                (0)
   /** Constant CLKM_4mA for sub-register @ref SR_PAD_IO_CLKM
    * @ingroup apiHalPHY230Const
    */
#  define CLKM_4mA                (1)
   /** Constant CLKM_6mA for sub-register @ref SR_PAD_IO_CLKM
    * @ingroup apiHalPHY230Const
    */
#  define CLKM_6mA                (2)
   /** Constant CLKM_8mA for sub-register @ref SR_PAD_IO_CLKM
    * @ingroup apiHalPHY230Const
    */
#  define CLKM_8mA                (3)
  /** Access parameters for sub-register CLKM_SHA_SEL in register @ref
RG_TRX_CTRL_0
   * @ingroup apiHalPHY230Sreg
   */
# define SR_CLKM_SHA_SEL             0x03, 0x08, 3
  /** Access parameters for sub-register CLKM_CTRL in register @ref
RG_TRX_CTRL_0
   * @ingroup apiHalPHY230Sreg
   */
# define SR_CLKM_CTRL                0x03, 0x07, 0
   /** Constant CLKM_no_clock for sub-register @ref SR_CLKM_CTRL
    * @ingroup apiHalPHY230Const
    */
#  define CLKM_no_clock           (0)
   /** Constant CLKM_1MHz for sub-register @ref SR_CLKM_CTRL
    * @ingroup apiHalPHY230Const
    */
#  define CLKM_1MHz               (1)
   /** Constant CLKM_2MHz for sub-register @ref SR_CLKM_CTRL
    * @ingroup apiHalPHY230Const
    */
#  define CLKM_2MHz              (2)
   /** Constant CLKM_4MHz for sub-register @ref SR_CLKM_CTRL
    * @ingroup apiHalPHY230Const
    */
#  define CLKM_4MHz              (3)
   /** Constant CLKM_8MHz for sub-register @ref SR_CLKM_CTRL
    * @ingroup apiHalPHY230Const
    */
#  define CLKM_8MHz              (4)
   /** Constant CLKM_16MHz for sub-register @ref SR_CLKM_CTRL
    * @ingroup apiHalPHY230Const
    */
#  define CLKM_16MHz              (5)

/** Offset for register PHY_TX_PWR
 * @ingroup apiHalPHY230Reg
 */
#define RG_PHY_TX_PWR                  (0x05)
  /** Access parameters for sub-register TX_AUTO_CRC_ON in register @ref
RG_PHY_TX_PWR
   * @ingroup apiHalPHY230Sreg
   */
# define SR_TX_AUTO_CRC_ON           0x05, 0x80, 7
# define SR_reserved_05_2            0x05, 0x70, 4
  /** Access parameters for sub-register TX_PWR in register @ref RG_PHY_TX_PWR
   * @ingroup apiHalPHY230Sreg
   */
# define SR_TX_PWR                   0x05, 0x0f, 0

/** Offset for register PHY_RSSI
  * @ingroup apiHalPHY230Reg
  */
#define RG_PHY_RSSI                    (0x06)
# define SR_reserved_06_1            0x06, 0xe0, 5
  /** Access parameters for sub-register RSSI in register @ref RG_PHY_RSSI
   * @ingroup apiHalPHY230Sreg
   */
# define SR_RSSI                     0x06, 0x1f, 0

/** Offset for register PHY_ED_LEVEL
  * @ingroup apiHalPHY230Reg
  */
#define RG_PHY_ED_LEVEL                (0x07)
```

```
  /** Access parameters for sub-register ED_LEVEL in register @ref
RG_PHY_ED_LEVEL
   * @ingroup apiHalPHY230Sreg
   */
# define SR_ED_LEVEL                 0x07, 0xff, 0

/** Offset for register PHY_CC_CCA
 * @ingroup apiHalPHY230Reg
 */
#define RG_PHY_CC_CCA                   (0x08)
  /** Access parameters for sub-register CCA_REQUEST in register @ref
RG_PHY_CC_CCA
   * @ingroup apiHalPHY230Sreg
   */
# define SR_CCA_REQUEST              0x08, 0x80, 7
  /** Access parameters for sub-register CCA_MODE in register @ref
RG_PHY_CC_CCA
   * @ingroup apiHalPHY230Sreg
   */
# define SR_CCA_MODE                 0x08, 0x60, 5
  /** Access parameters for sub-register CHANNEL in register @ref RG_PHY_CC_CCA
   * @ingroup apiHalPHY230Sreg
   */
# define SR_CHANNEL                  0x08, 0x1f, 0

/** Offset for register CCA_THRES
 * @ingroup apiHalPHY230Reg
 */
#define RG_CCA_THRES                    (0x09)
  /** Access parameters for sub-register CCA_CS_THRES in register @ref
RG_CCA_THRES
   * @ingroup apiHalPHY230Sreg
   */
# define SR_CCA_CS_THRES             0x09, 0xf0, 4
  /** Access parameters for sub-register CCA_ED_THRES in register @ref
RG_CCA_THRES
   * @ingroup apiHalPHY230Sreg
   */
# define SR_CCA_ED_THRES             0x09, 0x0f, 0

/** Offset for register IRQ_MASK
 * @ingroup apiHalPHY230Reg
 */
#define RG_IRQ_MASK                     (0x0e)
  /** Access parameters for sub-register IRQ_MASK in register @ref RG_IRQ_MASK
   * @ingroup apiHalPHY230Sreg
  */
# define SR_IRQ_MASK                0x0e, 0xff, 0

/** Offset for register IRQ_STATUS
 * @ingroup apiHalPHY230Reg
 */
#define RG_IRQ_STATUS                (0x0f)
 /** Access parameters for sub-register IRQ_7_BAT_LOW in register @ref
RG_IRQ_STATUS
  * @ingroup apiHalPHY230Sreg
  */
# define SR_IRQ_7_BAT_LOW           0x0f, 0x80, 7
 /** Access parameters for sub-register IRQ_6_TRX_UR in register @ref
RG_IRQ_STATUS
  * @ingroup apiHalPHY230Sreg
  */
# define SR_IRQ_6_TRX_UR            0x0f, 0x40, 6
 /** Access parameters for sub-register IRQ_5 in register @ref RG_IRQ_STATUS
  * @ingroup apiHalPHY230Sreg
  */
# define SR_IRQ_5                   0x0f, 0x20, 5
 /** Access parameters for sub-register IRQ_4 in register @ref RG_IRQ_STATUS
  * @ingroup apiHalPHY230Sreg
  */
# define SR_IRQ_4                   0x0f, 0x10, 4
 /** Access parameters for sub-register IRQ_3_TRX_END in register @ref
RG_IRQ_STATUS
  * @ingroup apiHalPHY230Sreg
  */
# define SR_IRQ_3_TRX_END           0x0f, 0x08, 3
 /** Access parameters for sub-register IRQ_2_RX_START in register @ref
RG_IRQ_STATUS
  * @ingroup apiHalPHY230Sreg
  */
# define SR_IRQ_2_RX_START          0x0f, 0x04, 2
 /** Access parameters for sub-register IRQ_1_PLL_UNLOCK in register @ref
RG_IRQ_STATUS
  * @ingroup apiHalPHY230Sreg
  */
# define SR_IRQ_1_PLL_UNLOCK        0x0f, 0x02, 1
 /** Access parameters for sub-register IRQ_0_PLL_LOCK in register @ref
RG_IRQ_STATUS
  * @ingroup apiHalPHY230Sreg
  */
# define SR_IRQ_0_PLL_LOCK          0x0f, 0x01, 0
```

```
/** Offset for register VREG_CTRL
 * @ingroup apiHalPHY230Reg
 */
#define RG_VREG_CTRL                    (0x10)
 /** Access parameters for sub-register AVREG_EXT in register @ref
RG_VREG_CTRL
   * @ingroup apiHalPHY230Sreg
   */
# define SR_AVREG_EXT               0x10, 0x80, 7
 /** Access parameters for sub-register AVDD_OK in register @ref RG_VREG_CTRL
   * @ingroup apiHalPHY230Sreg
   */
# define SR_AVDD_OK                 0x10, 0x40, 6
 /** Access parameters for sub-register AVREG_TRIM in register @ref
RG_VREG_CTRL
   * @ingroup apiHalPHY230Sreg
   */
# define SR_AVREG_TRIM              0x10, 0x30, 4
  /** Constant AVREG_1_80V for sub-register @ref SR_AVREG_TRIM
    * @ingroup apiHalPHY230Const
    */
#  define AVREG_1_80V            (0)
  /** Constant AVREG_1_75V for sub-register @ref SR_AVREG_TRIM
    * @ingroup apiHalPHY230Const
    */
#  define AVREG_1_75V            (1)
  /** Constant AVREG_1_84V for sub-register @ref SR_AVREG_TRIM
    * @ingroup apiHalPHY230Const
    */
#  define AVREG_1_84V            (2)
  /** Constant AVREG_1_88V for sub-register @ref SR_AVREG_TRIM
    * @ingroup apiHalPHY230Const
    */
#  define AVREG_1_88V            (3)
 /** Access parameters for sub-register DVREG_EXT in register @ref
RG_VREG_CTRL
   * @ingroup apiHalPHY230Sreg
   */
# define SR_DVREG_EXT               0x10, 0x08, 3
 /** Access parameters for sub-register DVDD_OK in register @ref RG_VREG_CTRL
   * @ingroup apiHalPHY230Sreg
   */
# define SR_DVDD_OK                 0x10, 0x04, 2
 /** Access parameters for sub-register DVREG_TRIM in register @ref
RG_VREG_CTRL
   * @ingroup apiHalPHY230Sreg
   */
# define SR_DVREG_TRIM              0x10, 0x03, 0
   /** Constant DVREG_1_80V for sub-register @ref SR_DVREG_TRIM
    * @ingroup apiHalPHY230Const
    */
#  define DVREG_1_80V           (0)
   /** Constant DVREG_1_75V for sub-register @ref SR_DVREG_TRIM
    * @ingroup apiHalPHY230Const
    */
#  define DVREG_1_75V           (1)
   /** Constant DVREG_1_84V for sub-register @ref SR_DVREG_TRIM
    * @ingroup apiHalPHY230Const
    */
#  define DVREG_1_84V           (2)
   /** Constant DVREG_1_88V for sub-register @ref SR_DVREG_TRIM
    * @ingroup apiHalPHY230Const
    */
#  define DVREG_1_88V           (3)

/** Offset for register BATMON
 * @ingroup apiHalPHY230Reg
 */
#define RG_BATMON                     (0x11)
# define SR_reserved_11_1           0x11, 0xc0, 6
 /** Access parameters for sub-register BATMON_OK in register @ref RG_BATMON
   * @ingroup apiHalPHY230Sreg
   */
# define SR_BATMON_OK               0x11, 0x20, 5
 /** Access parameters for sub-register BATMON_HR in register @ref RG_BATMON
   * @ingroup apiHalPHY230Sreg
   */
# define SR_BATMON_HR               0x11, 0x10, 4
 /** Access parameters for sub-register BATMON_VTH in register @ref
RG_BATMON
   * @ingroup apiHalPHY230Sreg
   */
# define SR_BATMON_VTH              0x11, 0x0f, 0

/** Offset for register XOSC_CTRL
 * @ingroup apiHalPHY230Reg
 */
#define RG_XOSC_CTRL                  (0x12)
 /** Access parameters for sub-register XTAL_MODE in register @ref
RG_XOSC_CTRL
   * @ingroup apiHalPHY230Sreg
   */
```

```
# define SR_XTAL_MODE                0x12, 0xf0, 4
  /** Access parameters for sub-register XTAL_TRIM in register @ref
RG_XOSC_CTRL
   * @ingroup apiHalPHY230Sreg
   */
# define SR_XTAL_TRIM                0x12, 0x0f, 0

/** Offset for register FTN_CTRL
 * @ingroup apiHalPHY230Reg
 */
#define RG_FTN_CTRL                  (0x18)
  /** Access parameters for sub-register FTN_START in register @ref RG_FTN_CTRL
   * @ingroup apiHalPHY230Sreg
   */
# define SR_FTN_START                0x18, 0x80, 7
# define SR_reserved_18_2            0x18, 0x40, 6
  /** Access parameters for sub-register FTNV in register @ref RG_FTN_CTRL
   * @ingroup apiHalPHY230Sreg
   */
# define SR_FTNV                     0x18, 0x3f, 0

/** Offset for register PLL_CF
 * @ingroup apiHalPHY230Reg
 */
#define RG_PLL_CF                    (0x1a)
  /** Access parameters for sub-register PLL_CF_START in register @ref RG_PLL_CF
   * @ingroup apiHalPHY230Sreg
   */
# define SR_PLL_CF_START             0x1a, 0x80, 7
# define SR_reserved_1a_2            0x1a, 0x70, 4
  /** Access parameters for sub-register PLL_CF in register @ref RG_PLL_CF
   * @ingroup apiHalPHY230Sreg
   */
# define SR_PLL_CF                   0x1a, 0x0f, 0

/** Offset for register PLL_DCU
 * @ingroup apiHalPHY230Reg
 */
#define RG_PLL_DCU                   (0x1b)
  /** Access parameters for sub-register PLL_DCU_START in register @ref
RG_PLL_DCU
   * @ingroup apiHalPHY230Sreg
   */
# define SR_PLL_DCU_START            0x1b, 0x80, 7
# define SR_reserved_1b_2            0x1b, 0x40, 6
  /** Access parameters for sub-register PLL_DCUW in register @ref RG_PLL_DCU
```

```
   * @ingroup apiHalPHY230Sreg
   */
# define SR_PLL_DCUW                0x1b, 0x3f, 0

/** Offset for register PART_NUM
 * @ingroup apiHalPHY230Reg
 */
#define RG_PART_NUM                    (0x1c)
  /** Access parameters for sub-register PART_NUM in register @ref RG_PART_NUM
   * @ingroup apiHalPHY230Sreg
   */
# define SR_PART_NUM                0x1c, 0xff, 0
   /** Constant RF230 for sub-register @ref SR_PART_NUM
    * @ingroup apiHalPHY230Const
    */
#  define RF230                 (2)

/** Offset for register VERSION_NUM
 * @ingroup apiHalPHY230Reg
 */
#define RG_VERSION_NUM                 (0x1d)
  /** Access parameters for sub-register VERSION_NUM in register @ref
RG_VERSION_NUM
   * @ingroup apiHalPHY230Sreg
   */
# define SR_VERSION_NUM             0x1d, 0xff, 0

/** Offset for register MAN_ID_0
 * @ingroup apiHalPHY230Reg
 */
#define RG_MAN_ID_0                    (0x1e)
  /** Access parameters for sub-register MAN_ID_0 in register @ref RG_MAN_ID_0
   * @ingroup apiHalPHY230Sreg
   */
# define SR_MAN_ID_0                0x1e, 0xff, 0

/** Offset for register MAN_ID_1
 * @ingroup apiHalPHY230Reg
 */
#define RG_MAN_ID_1                    (0x1f)
  /** Access parameters for sub-register MAN_ID_1 in register @ref RG_MAN_ID_1
   * @ingroup apiHalPHY230Sreg
   */
# define SR_MAN_ID_1                0x1f, 0xff, 0

/** Offset for register SHORT_ADDR_0
```

```
 * @ingroup apiHalPHY230Reg
 */
#define RG_SHORT_ADDR_0                (0x20)
  /** Access parameters for sub-register SHORT_ADDR_0 in register @ref
RG_SHORT_ADDR_0
   * @ingroup apiHalPHY230Sreg
   */
# define SR_SHORT_ADDR_0            0x20, 0xff, 0

/** Offset for register SHORT_ADDR_1
 * @ingroup apiHalPHY230Reg
 */
#define RG_SHORT_ADDR_1                (0x21)
  /** Access parameters for sub-register SHORT_ADDR_1 in register @ref
RG_SHORT_ADDR_1
   * @ingroup apiHalPHY230Sreg
   */
# define SR_SHORT_ADDR_1            0x21, 0xff, 0

/** Offset for register PAN_ID_0
 * @ingroup apiHalPHY230Reg
 */
#define RG_PAN_ID_0                  (0x22)
  /** Access parameters for sub-register PAN_ID_0 in register @ref RG_PAN_ID_0
   * @ingroup apiHalPHY230Sreg
   */
# define SR_PAN_ID_0                0x22, 0xff, 0

/** Offset for register PAN_ID_1
 * @ingroup apiHalPHY230Reg
 */
#define RG_PAN_ID_1                  (0x23)
  /** Access parameters for sub-register PAN_ID_1 in register @ref RG_PAN_ID_1
   * @ingroup apiHalPHY230Sreg
   */
# define SR_PAN_ID_1                0x23, 0xff, 0

/** Offset for register IEEE_ADDR_0
 * @ingroup apiHalPHY230Reg
 */
#define RG_IEEE_ADDR_0                (0x24)
  /** Access parameters for sub-register IEEE_ADDR_0 in register @ref
RG_IEEE_ADDR_0
   * @ingroup apiHalPHY230Sreg
   */
# define SR_IEEE_ADDR_0             0x24, 0xff, 0
```

```
/** Offset for register IEEE_ADDR_1
 * @ingroup apiHalPHY230Reg
 */
#define RG_IEEE_ADDR_1                  (0x25)
  /** Access parameters for sub-register IEEE_ADDR_1 in register @ref
RG_IEEE_ADDR_1
   * @ingroup apiHalPHY230Sreg
   */
# define SR_IEEE_ADDR_1               0x25, 0xff, 0

/** Offset for register IEEE_ADDR_2
 * @ingroup apiHalPHY230Reg
 */
#define RG_IEEE_ADDR_2                  (0x26)
  /** Access parameters for sub-register IEEE_ADDR_2 in register @ref
RG_IEEE_ADDR_2
   * @ingroup apiHalPHY230Sreg
   */
# define SR_IEEE_ADDR_2               0x26, 0xff, 0

/** Offset for register IEEE_ADDR_3
 * @ingroup apiHalPHY230Reg
 */
#define RG_IEEE_ADDR_3                  (0x27)
  /** Access parameters for sub-register IEEE_ADDR_3 in register @ref
RG_IEEE_ADDR_3
   * @ingroup apiHalPHY230Sreg
   */
# define SR_IEEE_ADDR_3               0x27, 0xff, 0

/** Offset for register IEEE_ADDR_4
 * @ingroup apiHalPHY230Reg
 */
#define RG_IEEE_ADDR_4                  (0x28)
  /** Access parameters for sub-register IEEE_ADDR_4 in register @ref
RG_IEEE_ADDR_4
   * @ingroup apiHalPHY230Sreg
   */
# define SR_IEEE_ADDR_4               0x28, 0xff, 0

/** Offset for register IEEE_ADDR_5
 * @ingroup apiHalPHY230Reg
 */
#define RG_IEEE_ADDR_5                  (0x29)
```

```
  /** Access parameters for sub-register IEEE_ADDR_5 in register @ref
RG_IEEE_ADDR_5
   * @ingroup apiHalPHY230Sreg
   */
# define SR_IEEE_ADDR_5              0x29, 0xff, 0

/** Offset for register IEEE_ADDR_6
 * @ingroup apiHalPHY230Reg
 */
#define RG_IEEE_ADDR_6                  (0x2a)
  /** Access parameters for sub-register IEEE_ADDR_6 in register @ref
RG_IEEE_ADDR_6
   * @ingroup apiHalPHY230Sreg
   */
# define SR_IEEE_ADDR_6              0x2a, 0xff, 0

/** Offset for register IEEE_ADDR_7
 * @ingroup apiHalPHY230Reg
 */
#define RG_IEEE_ADDR_7                  (0x2b)
  /** Access parameters for sub-register IEEE_ADDR_7 in register @ref
RG_IEEE_ADDR_7
   * @ingroup apiHalPHY230Sreg
   */
# define SR_IEEE_ADDR_7              0x2b, 0xff, 0

/** Offset for register XAH_CTRL
 * @ingroup apiHalPHY230Reg
 */
#define RG_XAH_CTRL                     (0x2c)
  /** Access parameters for sub-register MAX_FRAME_RETRIES in register @ref
RG_XAH_CTRL
   * @ingroup apiHalPHY230Sreg
   */
# define SR_MAX_FRAME_RETRIES        0x2c, 0xf0, 4
  /** Access parameters for sub-register MAX_CSMA_RETRIES in register @ref
RG_XAH_CTRL
   * @ingroup apiHalPHY230Sreg
   */
# define SR_MAX_CSMA_RETRIES         0x2c, 0x0e, 1
# define SR_reserved_2c_3            0x2c, 0x01, 0

/** Offset for register CSMA_SEED_0
 * @ingroup apiHalPHY230Reg
 */
#define RG_CSMA_SEED_0                  (0x2d)
```

```
  /** Access parameters for sub-register CSMA_SEED_0 in register @ref
RG_CSMA_SEED_0
   * @ingroup apiHalPHY230Sreg
   */
# define SR_CSMA_SEED_0             0x2d, 0xff, 0

/** Offset for register CSMA_SEED_1
 * @ingroup apiHalPHY230Reg
 */
#define RG_CSMA_SEED_1                 (0x2e)
  /** Access parameters for sub-register MIN_BE in register @ref RG_CSMA_SEED_1
   * @ingroup apiHalPHY230Sreg
   */
# define SR_MIN_BE                  0x2e, 0xc0, 6
# define SR_reserved_2e_2           0x2e, 0x30, 4
  /** Access parameters for sub-register I_AM_COORD in register @ref
RG_CSMA_SEED_1
   * @ingroup apiHalPHY230Sreg
   */
# define SR_I_AM_COORD              0x2e, 0x08, 3
  /** Access parameters for sub-register CSMA_SEED_1 in register @ref
RG_CSMA_SEED_1
   * @ingroup apiHalPHY230Sreg
   */
# define SR_CSMA_SEED_1             0x2e, 0x07, 0

#endif /* PHY230_REGISTERMAP_EXTERNAL_H */
```

Compiler.h

```
/*
 * This file is autogenerated from regxml2include.py
 * Do not modify it, changes will be lost after rebuild.
 */
/**
 * @file
 * generated register definition file
 * Inputfile:  phy230_registermap_external.xml
 * Version:    1.9    for external use
 * Created at: Thu Jan 25 17:07:33 2007
 *
 */
/*
 * Copyright (c) 2006, Atmel Corporation All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
```

```
/**
 *  @author
 *      Atmel Corporation: http://www.atmel.com
 *      Support email: avr@atmel.com
 */

#ifndef PHY230_REGISTERMAP_EXTERNAL_H
#define PHY230_REGISTERMAP_EXTERNAL_H

#define HAVE_REGISTER_MAP (1)
/** Offset for register TRX_STATUS
 * @ingroup apiHalPHY230Reg
 */
```

```
#define RG_TRX_STATUS                    (0x01)
  /** Access parameters for sub-register CCA_DONE in register @ref
RG_TRX_STATUS
   * @ingroup apiHalPHY230Sreg
   */
# define SR_CCA_DONE                  0x01, 0x80, 7
  /** Access parameters for sub-register CCA_STATUS in register @ref
RG_TRX_STATUS
   * @ingroup apiHalPHY230Sreg
   */
# define SR_CCA_STATUS                0x01, 0x40, 6
# define SR_reserved_01_3             0x01, 0x20, 5
  /** Access parameters for sub-register TRX_STATUS in register @ref
RG_TRX_STATUS
   * @ingroup apiHalPHY230Sreg
   */
# define SR_TRX_STATUS                0x01, 0x1f, 0
   /** Constant P_ON for sub-register @ref SR_TRX_STATUS
    * @ingroup apiHalPHY230Const
    */
#  define P_ON                    (0)
   /** Constant BUSY_RX for sub-register @ref SR_TRX_STATUS
    * @ingroup apiHalPHY230Const
    */
#  define BUSY_RX                 (1)
   /** Constant BUSY_TX for sub-register @ref SR_TRX_STATUS
    * @ingroup apiHalPHY230Const
    */
#  define BUSY_TX                 (2)
   /** Constant RX_ON for sub-register @ref SR_TRX_STATUS
    * @ingroup apiHalPHY230Const
    */
#  define RX_ON                   (6)
   /** Constant TRX_OFF for sub-register @ref SR_TRX_STATUS
    * @ingroup apiHalPHY230Const
    */
#  define TRX_OFF                 (8)
   /** Constant PLL_ON for sub-register @ref SR_TRX_STATUS
    * @ingroup apiHalPHY230Const
    */
#  define PLL_ON                  (9)
   /** Constant SLEEP for sub-register @ref SR_TRX_STATUS
    * @ingroup apiHalPHY230Const
    */
#  define SLEEP                   (15)
   /** Constant BUSY_RX_AACK for sub-register @ref SR_TRX_STATUS
```

```
    * @ingroup apiHalPHY230Const
    */
#  define BUSY_RX_AACK             (17)
   /** Constant BUSY_TX_ARET for sub-register @ref SR_TRX_STATUS
    * @ingroup apiHalPHY230Const
    */
#  define BUSY_TX_ARET             (18)
   /** Constant RX_AACK_ON for sub-register @ref SR_TRX_STATUS
    * @ingroup apiHalPHY230Const
    */
#  define RX_AACK_ON               (22)
   /** Constant TX_ARET_ON for sub-register @ref SR_TRX_STATUS
    * @ingroup apiHalPHY230Const
    */
#  define TX_ARET_ON               (25)
   /** Constant RX_ON_NOCLK for sub-register @ref SR_TRX_STATUS
    * @ingroup apiHalPHY230Const
    */
#  define RX_ON_NOCLK              (28)
   /** Constant RX_AACK_ON_NOCLK for sub-register @ref SR_TRX_STATUS
    * @ingroup apiHalPHY230Const
    */
#  define RX_AACK_ON_NOCLK         (29)
   /** Constant BUSY_RX_AACK_NOCLK for sub-register @ref SR_TRX_STATUS
    * @ingroup apiHalPHY230Const
    */
#  define BUSY_RX_AACK_NOCLK       (30)

/** Offset for register TRX_STATE
 * @ingroup apiHalPHY230Reg
 */
#define RG_TRX_STATE                 (0x02)
  /** Access parameters for sub-register TRAC_STATUS in register @ref
RG_TRX_STATE
   * @ingroup apiHalPHY230Sreg
   */
# define SR_TRAC_STATUS              0x02, 0xe0, 5
  /** Access parameters for sub-register TRX_CMD in register @ref RG_TRX_STATE
   * @ingroup apiHalPHY230Sreg
   */
# define SR_TRX_CMD                0x02, 0x1f, 0
   /** Constant CMD_NOP for sub-register @ref SR_TRX_CMD
    * @ingroup apiHalPHY230Const
    */
#  define CMD_NOP               (0)
   /** Constant CMD_TX_START for sub-register @ref SR_TRX_CMD
```

```
    * @ingroup apiHalPHY230Const
    */
#  define CMD_TX_START            (2)
   /** Constant CMD_FORCE_TRX_OFF for sub-register @ref SR_TRX_CMD
    * @ingroup apiHalPHY230Const
    */
#  define CMD_FORCE_TRX_OFF       (3)
   /** Constant CMD_RX_ON for sub-register @ref SR_TRX_CMD
    * @ingroup apiHalPHY230Const
    */
#  define CMD_RX_ON               (6)
   /** Constant CMD_TRX_OFF for sub-register @ref SR_TRX_CMD
    * @ingroup apiHalPHY230Const
    */
#  define CMD_TRX_OFF             (8)
   /** Constant CMD_PLL_ON for sub-register @ref SR_TRX_CMD
    * @ingroup apiHalPHY230Const
    */
#  define CMD_PLL_ON              (9)
   /** Constant CMD_RX_AACK_ON for sub-register @ref SR_TRX_CMD
    * @ingroup apiHalPHY230Const
    */
#  define CMD_RX_AACK_ON          (22)
   /** Constant CMD_TX_ARET_ON for sub-register @ref SR_TRX_CMD
    * @ingroup apiHalPHY230Const
    */
#  define CMD_TX_ARET_ON          (25)

/** Offset for register TRX_CTRL_0
 * @ingroup apiHalPHY230Reg
 */
#define RG_TRX_CTRL_0                   (0x03)
  /** Access parameters for sub-register PAD_IO in register @ref RG_TRX_CTRL_0
   * @ingroup apiHalPHY230Sreg
   */
# define SR_PAD_IO                  0x03, 0xc0, 6
  /** Access parameters for sub-register PAD_IO_CLKM in register @ref
RG_TRX_CTRL_0
   * @ingroup apiHalPHY230Sreg
   */
# define SR_PAD_IO_CLKM             0x03, 0x30, 4
   /** Constant CLKM_2mA for sub-register @ref SR_PAD_IO_CLKM
    * @ingroup apiHalPHY230Const
    */
#  define CLKM_2mA                (0)
   /** Constant CLKM_4mA for sub-register @ref SR_PAD_IO_CLKM
```

```
    * @ingroup apiHalPHY230Const
    */
#  define CLKM_4mA                (1)
   /** Constant CLKM_6mA for sub-register @ref SR_PAD_IO_CLKM
    * @ingroup apiHalPHY230Const
    */
#  define CLKM_6mA                (2)
   /** Constant CLKM_8mA for sub-register @ref SR_PAD_IO_CLKM
    * @ingroup apiHalPHY230Const
    */
#  define CLKM_8mA                (3)
  /** Access parameters for sub-register CLKM_SHA_SEL in register @ref
RG_TRX_CTRL_0
   * @ingroup apiHalPHY230Sreg
   */
# define SR_CLKM_SHA_SEL             0x03, 0x08, 3
  /** Access parameters for sub-register CLKM_CTRL in register @ref
RG_TRX_CTRL_0
   * @ingroup apiHalPHY230Sreg
   */
# define SR_CLKM_CTRL                0x03, 0x07, 0
   /** Constant CLKM_no_clock for sub-register @ref SR_CLKM_CTRL
    * @ingroup apiHalPHY230Const
    */
#  define CLKM_no_clock           (0)
   /** Constant CLKM_1MHz for sub-register @ref SR_CLKM_CTRL
    * @ingroup apiHalPHY230Const
    */
#  define CLKM_1MHz               (1)
   /** Constant CLKM_2MHz for sub-register @ref SR_CLKM_CTRL
    * @ingroup apiHalPHY230Const
    */
#  define CLKM_2MHz               (2)
   /** Constant CLKM_4MHz for sub-register @ref SR_CLKM_CTRL
    * @ingroup apiHalPHY230Const
    */
#  define CLKM_4MHz               (3)
   /** Constant CLKM_8MHz for sub-register @ref SR_CLKM_CTRL
    * @ingroup apiHalPHY230Const
    */
#  define CLKM_8MHz               (4)
   /** Constant CLKM_16MHz for sub-register @ref SR_CLKM_CTRL
    * @ingroup apiHalPHY230Const
    */
#  define CLKM_16MHz              (5)
```

```
/** Offset for register PHY_TX_PWR
 * @ingroup apiHalPHY230Reg
 */
#define RG_PHY_TX_PWR                   (0x05)
  /** Access parameters for sub-register TX_AUTO_CRC_ON in register @ref
RG_PHY_TX_PWR
   * @ingroup apiHalPHY230Sreg
   */
# define SR_TX_AUTO_CRC_ON            0x05, 0x80, 7
# define SR_reserved_05_2            0x05, 0x70, 4
  /** Access parameters for sub-register TX_PWR in register @ref RG_PHY_TX_PWR
   * @ingroup apiHalPHY230Sreg
   */
# define SR_TX_PWR                   0x05, 0x0f, 0

/** Offset for register PHY_RSSI
 * @ingroup apiHalPHY230Reg
 */
#define RG_PHY_RSSI                    (0x06)
# define SR_reserved_06_1            0x06, 0xe0, 5
  /** Access parameters for sub-register RSSI in register @ref RG_PHY_RSSI
   * @ingroup apiHalPHY230Sreg
   */
# define SR_RSSI                     0x06, 0x1f, 0

/** Offset for register PHY_ED_LEVEL
 * @ingroup apiHalPHY230Reg
 */
#define RG_PHY_ED_LEVEL                 (0x07)
  /** Access parameters for sub-register ED_LEVEL in register @ref
RG_PHY_ED_LEVEL
   * @ingroup apiHalPHY230Sreg
   */
# define SR_ED_LEVEL                 0x07, 0xff, 0

/** Offset for register PHY_CC_CCA
 * @ingroup apiHalPHY230Reg
 */
#define RG_PHY_CC_CCA                   (0x08)
  /** Access parameters for sub-register CCA_REQUEST in register @ref
RG_PHY_CC_CCA
   * @ingroup apiHalPHY230Sreg
   */
# define SR_CCA_REQUEST              0x08, 0x80, 7
  /** Access parameters for sub-register CCA_MODE in register @ref
RG_PHY_CC_CCA
```

```
  * @ingroup apiHalPHY230Sreg
  */
# define SR_CCA_MODE                0x08, 0x60, 5
 /** Access parameters for sub-register CHANNEL in register @ref RG_PHY_CC_CCA
  * @ingroup apiHalPHY230Sreg
  */
# define SR_CHANNEL                 0x08, 0x1f, 0

/** Offset for register CCA_THRES
 * @ingroup apiHalPHY230Reg
 */
#define RG_CCA_THRES                  (0x09)
 /** Access parameters for sub-register CCA_CS_THRES in register @ref
RG_CCA_THRES
  * @ingroup apiHalPHY230Sreg
  */
# define SR_CCA_CS_THRES            0x09, 0xf0, 4
 /** Access parameters for sub-register CCA_ED_THRES in register @ref
RG_CCA_THRES
  * @ingroup apiHalPHY230Sreg
  */
# define SR_CCA_ED_THRES            0x09, 0x0f, 0

/** Offset for register IRQ_MASK
 * @ingroup apiHalPHY230Reg
 */
#define RG_IRQ_MASK                   (0x0e)
 /** Access parameters for sub-register IRQ_MASK in register @ref RG_IRQ_MASK
  * @ingroup apiHalPHY230Sreg
  */
# define SR_IRQ_MASK                0x0e, 0xff, 0

/** Offset for register IRQ_STATUS
 * @ingroup apiHalPHY230Reg
 */
#define RG_IRQ_STATUS                 (0x0f)
 /** Access parameters for sub-register IRQ_7_BAT_LOW in register @ref
RG_IRQ_STATUS
  * @ingroup apiHalPHY230Sreg
  */
# define SR_IRQ_7_BAT_LOW           0x0f, 0x80, 7
 /** Access parameters for sub-register IRQ_6_TRX_UR in register @ref
RG_IRQ_STATUS
  * @ingroup apiHalPHY230Sreg
  */
# define SR_IRQ_6_TRX_UR            0x0f, 0x40, 6
```

```
 /** Access parameters for sub-register IRQ_5 in register @ref RG_IRQ_STATUS
   * @ingroup apiHalPHY230Sreg
   */
# define SR_IRQ_5                    0x0f, 0x20, 5
 /** Access parameters for sub-register IRQ_4 in register @ref RG_IRQ_STATUS
   * @ingroup apiHalPHY230Sreg
   */
# define SR_IRQ_4                    0x0f, 0x10, 4
 /** Access parameters for sub-register IRQ_3_TRX_END in register @ref
RG_IRQ_STATUS
   * @ingroup apiHalPHY230Sreg
   */
# define SR_IRQ_3_TRX_END            0x0f, 0x08, 3
 /** Access parameters for sub-register IRQ_2_RX_START in register @ref
RG_IRQ_STATUS
   * @ingroup apiHalPHY230Sreg
   */
# define SR_IRQ_2_RX_START           0x0f, 0x04, 2
 /** Access parameters for sub-register IRQ_1_PLL_UNLOCK in register @ref
RG_IRQ_STATUS
   * @ingroup apiHalPHY230Sreg
   */
# define SR_IRQ_1_PLL_UNLOCK         0x0f, 0x02, 1
 /** Access parameters for sub-register IRQ_0_PLL_LOCK in register @ref
RG_IRQ_STATUS
   * @ingroup apiHalPHY230Sreg
   */
# define SR_IRQ_0_PLL_LOCK           0x0f, 0x01, 0

/** Offset for register VREG_CTRL
 * @ingroup apiHalPHY230Reg
 */
#define RG_VREG_CTRL                 (0x10)
 /** Access parameters for sub-register AVREG_EXT in register @ref
RG_VREG_CTRL
   * @ingroup apiHalPHY230Sreg
   */
# define SR_AVREG_EXT                0x10, 0x80, 7
 /** Access parameters for sub-register AVDD_OK in register @ref RG_VREG_CTRL
   * @ingroup apiHalPHY230Sreg
   */
# define SR_AVDD_OK                  0x10, 0x40, 6
 /** Access parameters for sub-register AVREG_TRIM in register @ref
RG_VREG_CTRL
   * @ingroup apiHalPHY230Sreg
   */
```

```
# define SR_AVREG_TRIM              0x10, 0x30, 4
  /** Constant AVREG_1_80V for sub-register @ref SR_AVREG_TRIM
   * @ingroup apiHalPHY230Const
   */
#  define AVREG_1_80V            (0)
  /** Constant AVREG_1_75V for sub-register @ref SR_AVREG_TRIM
   * @ingroup apiHalPHY230Const
   */
#  define AVREG_1_75V            (1)
  /** Constant AVREG_1_84V for sub-register @ref SR_AVREG_TRIM
   * @ingroup apiHalPHY230Const
   */
#  define AVREG_1_84V            (2)
  /** Constant AVREG_1_88V for sub-register @ref SR_AVREG_TRIM
   * @ingroup apiHalPHY230Const
   */
#  define AVREG_1_88V            (3)
 /** Access parameters for sub-register DVREG_EXT in register @ref
RG_VREG_CTRL
  * @ingroup apiHalPHY230Sreg
  */
# define SR_DVREG_EXT               0x10, 0x08, 3
 /** Access parameters for sub-register DVDD_OK in register @ref RG_VREG_CTRL
  * @ingroup apiHalPHY230Sreg
  */
# define SR_DVDD_OK                 0x10, 0x04, 2
 /** Access parameters for sub-register DVREG_TRIM in register @ref
RG_VREG_CTRL
  * @ingroup apiHalPHY230Sreg
  */
# define SR_DVREG_TRIM              0x10, 0x03, 0
  /** Constant DVREG_1_80V for sub-register @ref SR_DVREG_TRIM
   * @ingroup apiHalPHY230Const
   */
#  define DVREG_1_80V            (0)
  /** Constant DVREG_1_75V for sub-register @ref SR_DVREG_TRIM
   * @ingroup apiHalPHY230Const
   */
#  define DVREG_1_75V            (1)
  /** Constant DVREG_1_84V for sub-register @ref SR_DVREG_TRIM
   * @ingroup apiHalPHY230Const
   */
#  define DVREG_1_84V            (2)
  /** Constant DVREG_1_88V for sub-register @ref SR_DVREG_TRIM
   * @ingroup apiHalPHY230Const
   */
```

```
#  define DVREG_1_88V              (3)

/** Offset for register BATMON
 * @ingroup apiHalPHY230Reg
 */
#define RG_BATMON                        (0x11)
# define SR_reserved_11_1            0x11, 0xc0, 6
  /** Access parameters for sub-register BATMON_OK in register @ref RG_BATMON
   * @ingroup apiHalPHY230Sreg
   */
# define SR_BATMON_OK                0x11, 0x20, 5
  /** Access parameters for sub-register BATMON_HR in register @ref RG_BATMON
   * @ingroup apiHalPHY230Sreg
   */
# define SR_BATMON_HR                0x11, 0x10, 4
  /** Access parameters for sub-register BATMON_VTH in register @ref
RG_BATMON
   * @ingroup apiHalPHY230Sreg
   */
# define SR_BATMON_VTH               0x11, 0x0f, 0

/** Offset for register XOSC_CTRL
 * @ingroup apiHalPHY230Reg
 */
#define RG_XOSC_CTRL                     (0x12)
  /** Access parameters for sub-register XTAL_MODE in register @ref
RG_XOSC_CTRL
   * @ingroup apiHalPHY230Sreg
   */
# define SR_XTAL_MODE                0x12, 0xf0, 4
  /** Access parameters for sub-register XTAL_TRIM in register @ref
RG_XOSC_CTRL
   * @ingroup apiHalPHY230Sreg
   */
# define SR_XTAL_TRIM                0x12, 0x0f, 0

/** Offset for register FTN_CTRL
 * @ingroup apiHalPHY230Reg
 */
#define RG_FTN_CTRL                      (0x18)
  /** Access parameters for sub-register FTN_START in register @ref RG_FTN_CTRL
   * @ingroup apiHalPHY230Sreg
   */
# define SR_FTN_START                0x18, 0x80, 7
# define SR_reserved_18_2            0x18, 0x40, 6
  /** Access parameters for sub-register FTNV in register @ref RG_FTN_CTRL
```

```
   * @ingroup apiHalPHY230Sreg
   */
# define SR_FTNV                    0x18, 0x3f, 0

/** Offset for register PLL_CF
 * @ingroup apiHalPHY230Reg
 */
#define RG_PLL_CF                   (0x1a)
  /** Access parameters for sub-register PLL_CF_START in register @ref RG_PLL_CF
   * @ingroup apiHalPHY230Sreg
   */
# define SR_PLL_CF_START            0x1a, 0x80, 7
# define SR_reserved_1a_2           0x1a, 0x70, 4
  /** Access parameters for sub-register PLL_CF in register @ref RG_PLL_CF
   * @ingroup apiHalPHY230Sreg
   */
# define SR_PLL_CF                  0x1a, 0x0f, 0

/** Offset for register PLL_DCU
 * @ingroup apiHalPHY230Reg
 */
#define RG_PLL_DCU                  (0x1b)
  /** Access parameters for sub-register PLL_DCU_START in register @ref
RG_PLL_DCU
   * @ingroup apiHalPHY230Sreg
   */
# define SR_PLL_DCU_START           0x1b, 0x80, 7
# define SR_reserved_1b_2           0x1b, 0x40, 6
  /** Access parameters for sub-register PLL_DCUW in register @ref RG_PLL_DCU
   * @ingroup apiHalPHY230Sreg
   */
# define SR_PLL_DCUW                0x1b, 0x3f, 0

/** Offset for register PART_NUM
 * @ingroup apiHalPHY230Reg
 */
#define RG_PART_NUM                 (0x1c)
  /** Access parameters for sub-register PART_NUM in register @ref RG_PART_NUM
   * @ingroup apiHalPHY230Sreg
   */
# define SR_PART_NUM                0x1c, 0xff, 0
   /** Constant RF230 for sub-register @ref SR_PART_NUM
    * @ingroup apiHalPHY230Const
    */
#  define RF230                 (2)
```

```
/** Offset for register VERSION_NUM
 * @ingroup apiHalPHY230Reg
 */
#define RG_VERSION_NUM                  (0x1d)
  /** Access parameters for sub-register VERSION_NUM in register @ref
RG_VERSION_NUM
   * @ingroup apiHalPHY230Sreg
   */
# define SR_VERSION_NUM              0x1d, 0xff, 0

/** Offset for register MAN_ID_0
 * @ingroup apiHalPHY230Reg
 */
#define RG_MAN_ID_0                   (0x1e)
  /** Access parameters for sub-register MAN_ID_0 in register @ref RG_MAN_ID_0
   * @ingroup apiHalPHY230Sreg
   */
# define SR_MAN_ID_0                0x1e, 0xff, 0

/** Offset for register MAN_ID_1
 * @ingroup apiHalPHY230Reg
 */
#define RG_MAN_ID_1                   (0x1f)
  /** Access parameters for sub-register MAN_ID_1 in register @ref RG_MAN_ID_1
   * @ingroup apiHalPHY230Sreg
   */
# define SR_MAN_ID_1                0x1f, 0xff, 0

/** Offset for register SHORT_ADDR_0
 * @ingroup apiHalPHY230Reg
 */
#define RG_SHORT_ADDR_0                 (0x20)
  /** Access parameters for sub-register SHORT_ADDR_0 in register @ref
RG_SHORT_ADDR_0
   * @ingroup apiHalPHY230Sreg
   */
# define SR_SHORT_ADDR_0             0x20, 0xff, 0

/** Offset for register SHORT_ADDR_1
 * @ingroup apiHalPHY230Reg
 */
#define RG_SHORT_ADDR_1                 (0x21)
  /** Access parameters for sub-register SHORT_ADDR_1 in register @ref
RG_SHORT_ADDR_1
   * @ingroup apiHalPHY230Sreg
   */
```

```
# define SR_SHORT_ADDR_1            0x21, 0xff, 0

/** Offset for register PAN_ID_0
 * @ingroup apiHalPHY230Reg
 */
#define RG_PAN_ID_0                  (0x22)
  /** Access parameters for sub-register PAN_ID_0 in register @ref RG_PAN_ID_0
   * @ingroup apiHalPHY230Sreg
   */
# define SR_PAN_ID_0               0x22, 0xff, 0

/** Offset for register PAN_ID_1
 * @ingroup apiHalPHY230Reg
 */
#define RG_PAN_ID_1                  (0x23)
  /** Access parameters for sub-register PAN_ID_1 in register @ref RG_PAN_ID_1
   * @ingroup apiHalPHY230Sreg
   */
# define SR_PAN_ID_1               0x23, 0xff, 0

/** Offset for register IEEE_ADDR_0
 * @ingroup apiHalPHY230Reg
 */
#define RG_IEEE_ADDR_0               (0x24)
  /** Access parameters for sub-register IEEE_ADDR_0 in register @ref
RG_IEEE_ADDR_0
   * @ingroup apiHalPHY230Sreg
   */
# define SR_IEEE_ADDR_0            0x24, 0xff, 0

/** Offset for register IEEE_ADDR_1
 * @ingroup apiHalPHY230Reg
 */
#define RG_IEEE_ADDR_1               (0x25)
  /** Access parameters for sub-register IEEE_ADDR_1 in register @ref
RG_IEEE_ADDR_1
   * @ingroup apiHalPHY230Sreg
   */
# define SR_IEEE_ADDR_1            0x25, 0xff, 0

/** Offset for register IEEE_ADDR_2
 * @ingroup apiHalPHY230Reg
 */
#define RG_IEEE_ADDR_2               (0x26)
  /** Access parameters for sub-register IEEE_ADDR_2 in register @ref
RG_IEEE_ADDR_2
```

```
   * @ingroup apiHalPHY230Sreg
   */
# define SR_IEEE_ADDR_2             0x26, 0xff, 0

/** Offset for register IEEE_ADDR_3
 * @ingroup apiHalPHY230Reg
 */
#define RG_IEEE_ADDR_3                (0x27)
  /** Access parameters for sub-register IEEE_ADDR_3 in register @ref
RG_IEEE_ADDR_3
   * @ingroup apiHalPHY230Sreg
   */
# define SR_IEEE_ADDR_3             0x27, 0xff, 0

/** Offset for register IEEE_ADDR_4
 * @ingroup apiHalPHY230Reg
 */
#define RG_IEEE_ADDR_4                (0x28)
  /** Access parameters for sub-register IEEE_ADDR_4 in register @ref
RG_IEEE_ADDR_4
   * @ingroup apiHalPHY230Sreg
   */
# define SR_IEEE_ADDR_4             0x28, 0xff, 0

/** Offset for register IEEE_ADDR_5
 * @ingroup apiHalPHY230Reg
 */
#define RG_IEEE_ADDR_5                (0x29)
  /** Access parameters for sub-register IEEE_ADDR_5 in register @ref
RG_IEEE_ADDR_5
   * @ingroup apiHalPHY230Sreg
   */
# define SR_IEEE_ADDR_5             0x29, 0xff, 0

/** Offset for register IEEE_ADDR_6
 * @ingroup apiHalPHY230Reg
 */
#define RG_IEEE_ADDR_6                (0x2a)
  /** Access parameters for sub-register IEEE_ADDR_6 in register @ref
RG_IEEE_ADDR_6
   * @ingroup apiHalPHY230Sreg
   */
# define SR_IEEE_ADDR_6             0x2a, 0xff, 0

/** Offset for register IEEE_ADDR_7
 * @ingroup apiHalPHY230Reg
 */
#define RG_IEEE_ADDR_7                  (0x2b)
  /** Access parameters for sub-register IEEE_ADDR_7 in register @ref
RG_IEEE_ADDR_7
   * @ingroup apiHalPHY230Sreg
   */
# define SR_IEEE_ADDR_7              0x2b, 0xff, 0

/** Offset for register XAH_CTRL
 * @ingroup apiHalPHY230Reg
 */
#define RG_XAH_CTRL                     (0x2c)
  /** Access parameters for sub-register MAX_FRAME_RETRIES in register @ref
RG_XAH_CTRL
   * @ingroup apiHalPHY230Sreg
   */
# define SR_MAX_FRAME_RETRIES        0x2c, 0xf0, 4
  /** Access parameters for sub-register MAX_CSMA_RETRIES in register @ref
RG_XAH_CTRL
   * @ingroup apiHalPHY230Sreg
   */
# define SR_MAX_CSMA_RETRIES         0x2c, 0x0e, 1
# define SR_reserved_2c_3            0x2c, 0x01, 0

/** Offset for register CSMA_SEED_0
 * @ingroup apiHalPHY230Reg
 */
#define RG_CSMA_SEED_0                  (0x2d)
  /** Access parameters for sub-register CSMA_SEED_0 in register @ref
RG_CSMA_SEED_0
   * @ingroup apiHalPHY230Sreg
   */
# define SR_CSMA_SEED_0              0x2d, 0xff, 0

/** Offset for register CSMA_SEED_1
 * @ingroup apiHalPHY230Reg
 */
#define RG_CSMA_SEED_1                  (0x2e)
  /** Access parameters for sub-register MIN_BE in register @ref RG_CSMA_SEED_1
   * @ingroup apiHalPHY230Sreg
   */
# define SR_MIN_BE                   0x2e, 0xc0, 6
# define SR_reserved_2e_2            0x2e, 0x30, 4
  /** Access parameters for sub-register I_AM_COORD in register @ref
RG_CSMA_SEED_1
   * @ingroup apiHalPHY230Sreg
   */
# define SR_I_AM_COORD              0x2e, 0x08, 3
  /** Access parameters for sub-register CSMA_SEED_1 in register @ref
RG_CSMA_SEED_1
   * @ingroup apiHalPHY230Sreg
   */
# define SR_CSMA_SEED_1              0x2e, 0x07, 0

#endif /* PHY230_REGISTERMAP_EXTERNAL_H */
```

Compiler_avr32.h

```
/*
 * This file is autogenerated from regxml2include.py
 * Do not modify it, changes will be lost after rebuild.
 */
/**
 * @file
 * generated register definition file
 * Inputfile:  phy230_registermap_external.xml
 * Version:    1.9    for external use
 * Created at: Thu Jan 25 17:07:33 2007
 *
 */
/*
 * Copyright (c) 2006, Atmel Corporation All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions are met:
 *
 * 1. Redistributions of source code must retain the above copyright notice,
 * this list of conditions and the following disclaimer.
 *
 * 2. 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.
 *
 * 3. The name of ATMEL may not be used to endorse or promote products derived
 * from this software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY ATMEL ``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
EXPRESSLY AND
```

```
 * SPECIFICALLY DISCLAIMED. IN NO EVENT SHALL ATMEL 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.
 */
/**
 *  @author
 *      Atmel Corporation: http://www.atmel.com
 *      Support email: avr@atmel.com
 */

#ifndef PHY230_REGISTERMAP_EXTERNAL_H
#define PHY230_REGISTERMAP_EXTERNAL_H

#define HAVE_REGISTER_MAP (1)
/** Offset for register TRX_STATUS
 * @ingroup apiHalPHY230Reg
 */
#define RG_TRX_STATUS                   (0x01)
  /** Access parameters for sub-register CCA_DONE in register @ref
RG_TRX_STATUS
   * @ingroup apiHalPHY230Sreg
   */
# define SR_CCA_DONE                0x01, 0x80, 7
  /** Access parameters for sub-register CCA_STATUS in register @ref
RG_TRX_STATUS
   * @ingroup apiHalPHY230Sreg
   */
# define SR_CCA_STATUS               0x01, 0x40, 6
# define SR_reserved_01_3            0x01, 0x20, 5
  /** Access parameters for sub-register TRX_STATUS in register @ref
RG_TRX_STATUS
   * @ingroup apiHalPHY230Sreg
   */
# define SR_TRX_STATUS               0x01, 0x1f, 0
   /** Constant P_ON for sub-register @ref SR_TRX_STATUS
```

```
    * @ingroup apiHalPHY230Const
    */
#  define P_ON                    (0)
   /** Constant BUSY_RX for sub-register @ref SR_TRX_STATUS
    * @ingroup apiHalPHY230Const
    */
#  define BUSY_RX                 (1)
   /** Constant BUSY_TX for sub-register @ref SR_TRX_STATUS
    * @ingroup apiHalPHY230Const
    */
#  define BUSY_TX                 (2)
   /** Constant RX_ON for sub-register @ref SR_TRX_STATUS
    * @ingroup apiHalPHY230Const
    */
#  define RX_ON                   (6)
   /** Constant TRX_OFF for sub-register @ref SR_TRX_STATUS
    * @ingroup apiHalPHY230Const
    */
#  define TRX_OFF                 (8)
   /** Constant PLL_ON for sub-register @ref SR_TRX_STATUS
    * @ingroup apiHalPHY230Const
    */
#  define PLL_ON                  (9)
   /** Constant SLEEP for sub-register @ref SR_TRX_STATUS
    * @ingroup apiHalPHY230Const
    */
#  define SLEEP                   (15)
   /** Constant BUSY_RX_AACK for sub-register @ref SR_TRX_STATUS
    * @ingroup apiHalPHY230Const
    */
#  define BUSY_RX_AACK            (17)
   /** Constant BUSY_TX_ARET for sub-register @ref SR_TRX_STATUS
    * @ingroup apiHalPHY230Const
    */
#  define BUSY_TX_ARET            (18)
   /** Constant RX_AACK_ON for sub-register @ref SR_TRX_STATUS
    * @ingroup apiHalPHY230Const
    */
#  define RX_AACK_ON              (22)
   /** Constant TX_ARET_ON for sub-register @ref SR_TRX_STATUS
    * @ingroup apiHalPHY230Const
    */
#  define TX_ARET_ON              (25)
   /** Constant RX_ON_NOCLK for sub-register @ref SR_TRX_STATUS
    * @ingroup apiHalPHY230Const
    */
```

```
#  define RX_ON_NOCLK              (28)
   /** Constant RX_AACK_ON_NOCLK for sub-register @ref SR_TRX_STATUS
    * @ingroup apiHalPHY230Const
    */
#  define RX_AACK_ON_NOCLK         (29)
   /** Constant BUSY_RX_AACK_NOCLK for sub-register @ref SR_TRX_STATUS
    * @ingroup apiHalPHY230Const
    */
#  define BUSY_RX_AACK_NOCLK       (30)

/** Offset for register TRX_STATE
 * @ingroup apiHalPHY230Reg
 */
#define RG_TRX_STATE                    (0x02)
  /** Access parameters for sub-register TRAC_STATUS in register @ref
RG_TRX_STATE
   * @ingroup apiHalPHY230Sreg
   */
# define SR_TRAC_STATUS              0x02, 0xe0, 5
  /** Access parameters for sub-register TRX_CMD in register @ref RG_TRX_STATE
   * @ingroup apiHalPHY230Sreg
   */
# define SR_TRX_CMD                  0x02, 0x1f, 0
   /** Constant CMD_NOP for sub-register @ref SR_TRX_CMD
    * @ingroup apiHalPHY230Const
    */
#  define CMD_NOP               (0)
   /** Constant CMD_TX_START for sub-register @ref SR_TRX_CMD
    * @ingroup apiHalPHY230Const
    */
#  define CMD_TX_START            (2)
   /** Constant CMD_FORCE_TRX_OFF for sub-register @ref SR_TRX_CMD
    * @ingroup apiHalPHY230Const
    */
#  define CMD_FORCE_TRX_OFF        (3)
   /** Constant CMD_RX_ON for sub-register @ref SR_TRX_CMD
    * @ingroup apiHalPHY230Const
    */
#  define CMD_RX_ON              (6)
   /** Constant CMD_TRX_OFF for sub-register @ref SR_TRX_CMD
    * @ingroup apiHalPHY230Const
    */
#  define CMD_TRX_OFF             (8)
   /** Constant CMD_PLL_ON for sub-register @ref SR_TRX_CMD
    * @ingroup apiHalPHY230Const
    */
```

```
#  define CMD_PLL_ON              (9)
  /** Constant CMD_RX_AACK_ON for sub-register @ref SR_TRX_CMD
   * @ingroup apiHalPHY230Const
   */
#  define CMD_RX_AACK_ON          (22)
  /** Constant CMD_TX_ARET_ON for sub-register @ref SR_TRX_CMD
   * @ingroup apiHalPHY230Const
   */
#  define CMD_TX_ARET_ON          (25)

/** Offset for register TRX_CTRL_0
 * @ingroup apiHalPHY230Reg
 */
#define RG_TRX_CTRL_0                 (0x03)
 /** Access parameters for sub-register PAD_IO in register @ref RG_TRX_CTRL_0
  * @ingroup apiHalPHY230Sreg
  */
# define SR_PAD_IO                  0x03, 0xc0, 6
 /** Access parameters for sub-register PAD_IO_CLKM in register @ref
RG_TRX_CTRL_0
  * @ingroup apiHalPHY230Sreg
  */
# define SR_PAD_IO_CLKM             0x03, 0x30, 4
  /** Constant CLKM_2mA for sub-register @ref SR_PAD_IO_CLKM
   * @ingroup apiHalPHY230Const
   */
#  define CLKM_2mA               (0)
  /** Constant CLKM_4mA for sub-register @ref SR_PAD_IO_CLKM
   * @ingroup apiHalPHY230Const
   */
#  define CLKM_4mA               (1)
  /** Constant CLKM_6mA for sub-register @ref SR_PAD_IO_CLKM
   * @ingroup apiHalPHY230Const
   */
#  define CLKM_6mA               (2)
  /** Constant CLKM_8mA for sub-register @ref SR_PAD_IO_CLKM
   * @ingroup apiHalPHY230Const
   */
#  define CLKM_8mA               (3)
 /** Access parameters for sub-register CLKM_SHA_SEL in register @ref
RG_TRX_CTRL_0
  * @ingroup apiHalPHY230Sreg
  */
# define SR_CLKM_SHA_SEL            0x03, 0x08, 3
 /** Access parameters for sub-register CLKM_CTRL in register @ref
RG_TRX_CTRL_0
```

```
   * @ingroup apiHalPHY230Sreg
   */
# define SR_CLKM_CTRL                0x03, 0x07, 0
  /** Constant CLKM_no_clock for sub-register @ref SR_CLKM_CTRL
   * @ingroup apiHalPHY230Const
   */
#  define CLKM_no_clock           (0)
  /** Constant CLKM_1MHz for sub-register @ref SR_CLKM_CTRL
   * @ingroup apiHalPHY230Const
   */
#  define CLKM_1MHz               (1)
  /** Constant CLKM_2MHz for sub-register @ref SR_CLKM_CTRL
   * @ingroup apiHalPHY230Const
   */
#  define CLKM_2MHz               (2)
  /** Constant CLKM_4MHz for sub-register @ref SR_CLKM_CTRL
   * @ingroup apiHalPHY230Const
   */
#  define CLKM_4MHz               (3)
  /** Constant CLKM_8MHz for sub-register @ref SR_CLKM_CTRL
   * @ingroup apiHalPHY230Const
   */
#  define CLKM_8MHz               (4)
  /** Constant CLKM_16MHz for sub-register @ref SR_CLKM_CTRL
   * @ingroup apiHalPHY230Const
   */
#  define CLKM_16MHz              (5)

/** Offset for register PHY_TX_PWR
 * @ingroup apiHalPHY230Reg
 */
#define RG_PHY_TX_PWR                  (0x05)
 /** Access parameters for sub-register TX_AUTO_CRC_ON in register @ref
RG_PHY_TX_PWR
  * @ingroup apiHalPHY230Sreg
  */
# define SR_TX_AUTO_CRC_ON           0x05, 0x80, 7
# define SR_reserved_05_2            0x05, 0x70, 4
 /** Access parameters for sub-register TX_PWR in register @ref RG_PHY_TX_PWR
  * @ingroup apiHalPHY230Sreg
  */
# define SR_TX_PWR                   0x05, 0x0f, 0

/** Offset for register PHY_RSSI
 * @ingroup apiHalPHY230Reg
 */
```

```
#define RG_PHY_RSSI                     (0x06)
# define SR_reserved_06_1            0x06, 0xe0, 5
  /** Access parameters for sub-register RSSI in register @ref RG_PHY_RSSI
   * @ingroup apiHalPHY230Sreg
   */
# define SR_RSSI                     0x06, 0x1f, 0

/** Offset for register PHY_ED_LEVEL
 * @ingroup apiHalPHY230Reg
 */
#define RG_PHY_ED_LEVEL                 (0x07)
  /** Access parameters for sub-register ED_LEVEL in register @ref
RG_PHY_ED_LEVEL
   * @ingroup apiHalPHY230Sreg
   */
# define SR_ED_LEVEL                 0x07, 0xff, 0

/** Offset for register PHY_CC_CCA
 * @ingroup apiHalPHY230Reg
 */
#define RG_PHY_CC_CCA                   (0x08)
  /** Access parameters for sub-register CCA_REQUEST in register @ref
RG_PHY_CC_CCA
   * @ingroup apiHalPHY230Sreg
   */
# define SR_CCA_REQUEST              0x08, 0x80, 7
  /** Access parameters for sub-register CCA_MODE in register @ref
RG_PHY_CC_CCA
   * @ingroup apiHalPHY230Sreg
   */
# define SR_CCA_MODE                 0x08, 0x60, 5
  /** Access parameters for sub-register CHANNEL in register @ref RG_PHY_CC_CCA
   * @ingroup apiHalPHY230Sreg
   */
# define SR_CHANNEL                  0x08, 0x1f, 0

/** Offset for register CCA_THRES
 * @ingroup apiHalPHY230Reg
 */
#define RG_CCA_THRES                    (0x09)
  /** Access parameters for sub-register CCA_CS_THRES in register @ref
RG_CCA_THRES
   * @ingroup apiHalPHY230Sreg
   */
# define SR_CCA_CS_THRES             0x09, 0xf0, 4
```

```
  /** Access parameters for sub-register CCA_ED_THRES in register @ref
RG_CCA_THRES
   * @ingroup apiHalPHY230Sreg
   */
# define SR_CCA_ED_THRES              0x09, 0x0f, 0

/** Offset for register IRQ_MASK
 * @ingroup apiHalPHY230Reg
 */
#define RG_IRQ_MASK                   (0x0e)
  /** Access parameters for sub-register IRQ_MASK in register @ref RG_IRQ_MASK
   * @ingroup apiHalPHY230Sreg
   */
# define SR_IRQ_MASK                  0x0e, 0xff, 0

/** Offset for register IRQ_STATUS
 * @ingroup apiHalPHY230Reg
 */
#define RG_IRQ_STATUS                 (0x0f)
  /** Access parameters for sub-register IRQ_7_BAT_LOW in register @ref
RG_IRQ_STATUS
   * @ingroup apiHalPHY230Sreg
   */
# define SR_IRQ_7_BAT_LOW             0x0f, 0x80, 7
  /** Access parameters for sub-register IRQ_6_TRX_UR in register @ref
RG_IRQ_STATUS
   * @ingroup apiHalPHY230Sreg
   */
# define SR_IRQ_6_TRX_UR              0x0f, 0x40, 6
  /** Access parameters for sub-register IRQ_5 in register @ref RG_IRQ_STATUS
   * @ingroup apiHalPHY230Sreg
   */
# define SR_IRQ_5                     0x0f, 0x20, 5
  /** Access parameters for sub-register IRQ_4 in register @ref RG_IRQ_STATUS
   * @ingroup apiHalPHY230Sreg
   */
# define SR_IRQ_4                     0x0f, 0x10, 4
  /** Access parameters for sub-register IRQ_3_TRX_END in register @ref
RG_IRQ_STATUS
   * @ingroup apiHalPHY230Sreg
   */
# define SR_IRQ_3_TRX_END             0x0f, 0x08, 3
  /** Access parameters for sub-register IRQ_2_RX_START in register @ref
RG_IRQ_STATUS
   * @ingroup apiHalPHY230Sreg
   */
```

```
# define SR_IRQ_2_RX_START           0x0f, 0x04, 2
  /** Access parameters for sub-register IRQ_1_PLL_UNLOCK in register @ref
RG_IRQ_STATUS
   * @ingroup apiHalPHY230Sreg
   */
# define SR_IRQ_1_PLL_UNLOCK         0x0f, 0x02, 1
  /** Access parameters for sub-register IRQ_0_PLL_LOCK in register @ref
RG_IRQ_STATUS
   * @ingroup apiHalPHY230Sreg
   */
# define SR_IRQ_0_PLL_LOCK           0x0f, 0x01, 0

/** Offset for register VREG_CTRL
 * @ingroup apiHalPHY230Reg
 */
#define RG_VREG_CTRL                  (0x10)
  /** Access parameters for sub-register AVREG_EXT in register @ref
RG_VREG_CTRL
   * @ingroup apiHalPHY230Sreg
   */
# define SR_AVREG_EXT               0x10, 0x80, 7
  /** Access parameters for sub-register AVDD_OK in register @ref RG_VREG_CTRL
   * @ingroup apiHalPHY230Sreg
   */
# define SR_AVDD_OK                 0x10, 0x40, 6
  /** Access parameters for sub-register AVREG_TRIM in register @ref
RG_VREG_CTRL
   * @ingroup apiHalPHY230Sreg
   */
# define SR_AVREG_TRIM              0x10, 0x30, 4
   /** Constant AVREG_1_80V for sub-register @ref SR_AVREG_TRIM
    * @ingroup apiHalPHY230Const
    */
#  define AVREG_1_80V            (0)
   /** Constant AVREG_1_75V for sub-register @ref SR_AVREG_TRIM
    * @ingroup apiHalPHY230Const
    */
#  define AVREG_1_75V            (1)
   /** Constant AVREG_1_84V for sub-register @ref SR_AVREG_TRIM
    * @ingroup apiHalPHY230Const
    */
#  define AVREG_1_84V            (2)
   /** Constant AVREG_1_88V for sub-register @ref SR_AVREG_TRIM
    * @ingroup apiHalPHY230Const
    */
#  define AVREG_1_88V            (3)
```

```
  /** Access parameters for sub-register DVREG_EXT in register @ref
RG_VREG_CTRL
   * @ingroup apiHalPHY230Sreg
   */
# define SR_DVREG_EXT               0x10, 0x08, 3
  /** Access parameters for sub-register DVDD_OK in register @ref RG_VREG_CTRL
   * @ingroup apiHalPHY230Sreg
   */
# define SR_DVDD_OK                 0x10, 0x04, 2
  /** Access parameters for sub-register DVREG_TRIM in register @ref
RG_VREG_CTRL
   * @ingroup apiHalPHY230Sreg
   */
# define SR_DVREG_TRIM              0x10, 0x03, 0
   /** Constant DVREG_1_80V for sub-register @ref SR_DVREG_TRIM
    * @ingroup apiHalPHY230Const
    */
#  define DVREG_1_80V            (0)
   /** Constant DVREG_1_75V for sub-register @ref SR_DVREG_TRIM
    * @ingroup apiHalPHY230Const
    */
#  define DVREG_1_75V            (1)
   /** Constant DVREG_1_84V for sub-register @ref SR_DVREG_TRIM
    * @ingroup apiHalPHY230Const
    */
#  define DVREG_1_84V            (2)
   /** Constant DVREG_1_88V for sub-register @ref SR_DVREG_TRIM
    * @ingroup apiHalPHY230Const
    */
#  define DVREG_1_88V            (3)

/** Offset for register BATMON
 * @ingroup apiHalPHY230Reg
 */
#define RG_BATMON                   (0x11)
# define SR_reserved_11_1          0x11, 0xc0, 6
  /** Access parameters for sub-register BATMON_OK in register @ref RG_BATMON
   * @ingroup apiHalPHY230Sreg
   */
# define SR_BATMON_OK               0x11, 0x20, 5
  /** Access parameters for sub-register BATMON_HR in register @ref RG_BATMON
   * @ingroup apiHalPHY230Sreg
   */
# define SR_BATMON_HR               0x11, 0x10, 4
  /** Access parameters for sub-register BATMON_VTH in register @ref
RG_BATMON
```

```
   * @ingroup apiHalPHY230Sreg
   */
# define SR_BATMON_VTH               0x11, 0x0f, 0

/** Offset for register XOSC_CTRL
 * @ingroup apiHalPHY230Reg
 */
#define RG_XOSC_CTRL                    (0x12)
  /** Access parameters for sub-register XTAL_MODE in register @ref
RG_XOSC_CTRL
   * @ingroup apiHalPHY230Sreg
   */
# define SR_XTAL_MODE                0x12, 0xf0, 4
  /** Access parameters for sub-register XTAL_TRIM in register @ref
RG_XOSC_CTRL
   * @ingroup apiHalPHY230Sreg
   */
# define SR_XTAL_TRIM                0x12, 0x0f, 0

/** Offset for register FTN_CTRL
 * @ingroup apiHalPHY230Reg
 */
#define RG_FTN_CTRL                     (0x18)
  /** Access parameters for sub-register FTN_START in register @ref RG_FTN_CTRL
   * @ingroup apiHalPHY230Sreg
   */
# define SR_FTN_START                0x18, 0x80, 7
# define SR_reserved_18_2            0x18, 0x40, 6
  /** Access parameters for sub-register FTNV in register @ref RG_FTN_CTRL
   * @ingroup apiHalPHY230Sreg
   */
# define SR_FTNV                     0x18, 0x3f, 0

/** Offset for register PLL_CF
 * @ingroup apiHalPHY230Reg
 */
#define RG_PLL_CF                       (0x1a)
  /** Access parameters for sub-register PLL_CF_START in register @ref RG_PLL_CF
   * @ingroup apiHalPHY230Sreg
   */
# define SR_PLL_CF_START             0x1a, 0x80, 7
# define SR_reserved_1a_2            0x1a, 0x70, 4
  /** Access parameters for sub-register PLL_CF in register @ref RG_PLL_CF
   * @ingroup apiHalPHY230Sreg
   */
# define SR_PLL_CF                   0x1a, 0x0f, 0
```

```
/** Offset for register PLL_DCU
 * @ingroup apiHalPHY230Reg
 */
#define RG_PLL_DCU                     (0x1b)
  /** Access parameters for sub-register PLL_DCU_START in register @ref
RG_PLL_DCU
   * @ingroup apiHalPHY230Sreg
   */
# define SR_PLL_DCU_START            0x1b, 0x80, 7
# define SR_reserved_1b_2            0x1b, 0x40, 6
  /** Access parameters for sub-register PLL_DCUW in register @ref RG_PLL_DCU
   * @ingroup apiHalPHY230Sreg
   */
# define SR_PLL_DCUW                 0x1b, 0x3f, 0

/** Offset for register PART_NUM
 * @ingroup apiHalPHY230Reg
 */
#define RG_PART_NUM                    (0x1c)
  /** Access parameters for sub-register PART_NUM in register @ref RG_PART_NUM
   * @ingroup apiHalPHY230Sreg
   */
# define SR_PART_NUM                 0x1c, 0xff, 0
   /** Constant RF230 for sub-register @ref SR_PART_NUM
    * @ingroup apiHalPHY230Const
    */
#  define RF230                   (2)

/** Offset for register VERSION_NUM
 * @ingroup apiHalPHY230Reg
 */
#define RG_VERSION_NUM                 (0x1d)
  /** Access parameters for sub-register VERSION_NUM in register @ref
RG_VERSION_NUM
   * @ingroup apiHalPHY230Sreg
   */
# define SR_VERSION_NUM              0x1d, 0xff, 0

/** Offset for register MAN_ID_0
 * @ingroup apiHalPHY230Reg
 */
#define RG_MAN_ID_0                    (0x1e)
  /** Access parameters for sub-register MAN_ID_0 in register @ref RG_MAN_ID_0
   * @ingroup apiHalPHY230Sreg
   */
```

```
# define SR_MAN_ID_0                0x1e, 0xff, 0

/** Offset for register MAN_ID_1
 * @ingroup apiHalPHY230Reg
 */
#define RG_MAN_ID_1                    (0x1f)
  /** Access parameters for sub-register MAN_ID_1 in register @ref RG_MAN_ID_1
   * @ingroup apiHalPHY230Sreg
   */
# define SR_MAN_ID_1                0x1f, 0xff, 0

/** Offset for register SHORT_ADDR_0
 * @ingroup apiHalPHY230Reg
 */
#define RG_SHORT_ADDR_0                (0x20)
  /** Access parameters for sub-register SHORT_ADDR_0 in register @ref
RG_SHORT_ADDR_0
   * @ingroup apiHalPHY230Sreg
   */
# define SR_SHORT_ADDR_0            0x20, 0xff, 0

/** Offset for register SHORT_ADDR_1
 * @ingroup apiHalPHY230Reg
 */
#define RG_SHORT_ADDR_1                (0x21)
  /** Access parameters for sub-register SHORT_ADDR_1 in register @ref
RG_SHORT_ADDR_1
   * @ingroup apiHalPHY230Sreg
   */
# define SR_SHORT_ADDR_1            0x21, 0xff, 0

/** Offset for register PAN_ID_0
 * @ingroup apiHalPHY230Reg
 */
#define RG_PAN_ID_0                    (0x22)
  /** Access parameters for sub-register PAN_ID_0 in register @ref RG_PAN_ID_0
   * @ingroup apiHalPHY230Sreg
   */
# define SR_PAN_ID_0                0x22, 0xff, 0

/** Offset for register PAN_ID_1
 * @ingroup apiHalPHY230Reg
 */
#define RG_PAN_ID_1                    (0x23)
  /** Access parameters for sub-register PAN_ID_1 in register @ref RG_PAN_ID_1
   * @ingroup apiHalPHY230Sreg
   */
# define SR_PAN_ID_1                 0x23, 0xff, 0

/** Offset for register IEEE_ADDR_0
 * @ingroup apiHalPHY230Reg
 */
#define RG_IEEE_ADDR_0                 (0x24)
  /** Access parameters for sub-register IEEE_ADDR_0 in register @ref
RG_IEEE_ADDR_0
   * @ingroup apiHalPHY230Sreg
   */
# define SR_IEEE_ADDR_0              0x24, 0xff, 0

/** Offset for register IEEE_ADDR_1
 * @ingroup apiHalPHY230Reg
 */
#define RG_IEEE_ADDR_1                 (0x25)
  /** Access parameters for sub-register IEEE_ADDR_1 in register @ref
RG_IEEE_ADDR_1
   * @ingroup apiHalPHY230Sreg
   */
# define SR_IEEE_ADDR_1              0x25, 0xff, 0

/** Offset for register IEEE_ADDR_2
 * @ingroup apiHalPHY230Reg
 */
#define RG_IEEE_ADDR_2                 (0x26)
  /** Access parameters for sub-register IEEE_ADDR_2 in register @ref
RG_IEEE_ADDR_2
   * @ingroup apiHalPHY230Sreg
   */
# define SR_IEEE_ADDR_2              0x26, 0xff, 0

/** Offset for register IEEE_ADDR_3
 * @ingroup apiHalPHY230Reg
 */
#define RG_IEEE_ADDR_3                 (0x27)
  /** Access parameters for sub-register IEEE_ADDR_3 in register @ref
RG_IEEE_ADDR_3
   * @ingroup apiHalPHY230Sreg
   */
# define SR_IEEE_ADDR_3              0x27, 0xff, 0

/** Offset for register IEEE_ADDR_4
 * @ingroup apiHalPHY230Reg
 */
```

```
#define RG_IEEE_ADDR_4                  (0x28)
  /** Access parameters for sub-register IEEE_ADDR_4 in register @ref
RG_IEEE_ADDR_4
   * @ingroup apiHalPHY230Sreg
   */
# define SR_IEEE_ADDR_4               0x28, 0xff, 0

/** Offset for register IEEE_ADDR_5
 * @ingroup apiHalPHY230Reg
 */
#define RG_IEEE_ADDR_5                  (0x29)
  /** Access parameters for sub-register IEEE_ADDR_5 in register @ref
RG_IEEE_ADDR_5
   * @ingroup apiHalPHY230Sreg
   */
# define SR_IEEE_ADDR_5               0x29, 0xff, 0

/** Offset for register IEEE_ADDR_6
 * @ingroup apiHalPHY230Reg
 */
#define RG_IEEE_ADDR_6                  (0x2a)
  /** Access parameters for sub-register IEEE_ADDR_6 in register @ref
RG_IEEE_ADDR_6
   * @ingroup apiHalPHY230Sreg
   */
# define SR_IEEE_ADDR_6               0x2a, 0xff, 0

/** Offset for register IEEE_ADDR_7
 * @ingroup apiHalPHY230Reg
 */
#define RG_IEEE_ADDR_7                  (0x2b)
  /** Access parameters for sub-register IEEE_ADDR_7 in register @ref
RG_IEEE_ADDR_7
   * @ingroup apiHalPHY230Sreg
   */
# define SR_IEEE_ADDR_7               0x2b, 0xff, 0

/** Offset for register XAH_CTRL
 * @ingroup apiHalPHY230Reg
 */
#define RG_XAH_CTRL                     (0x2c)
  /** Access parameters for sub-register MAX_FRAME_RETRIES in register @ref
RG_XAH_CTRL
   * @ingroup apiHalPHY230Sreg
   */
# define SR_MAX_FRAME_RETRIES         0x2c, 0xf0, 4
```

```
  /** Access parameters for sub-register MAX_CSMA_RETRIES in register @ref
RG_XAH_CTRL
   * @ingroup apiHalPHY230Sreg
   */
# define SR_MAX_CSMA_RETRIES         0x2c, 0x0e, 1
# define SR_reserved_2c_3            0x2c, 0x01, 0

/** Offset for register CSMA_SEED_0
 * @ingroup apiHalPHY230Reg
 */
#define RG_CSMA_SEED_0                (0x2d)
  /** Access parameters for sub-register CSMA_SEED_0 in register @ref
RG_CSMA_SEED_0
   * @ingroup apiHalPHY230Sreg
   */
# define SR_CSMA_SEED_0              0x2d, 0xff, 0

/** Offset for register CSMA_SEED_1
 * @ingroup apiHalPHY230Reg
 */
#define RG_CSMA_SEED_1                (0x2e)
  /** Access parameters for sub-register MIN_BE in register @ref RG_CSMA_SEED_1
   * @ingroup apiHalPHY230Sreg
   */
# define SR_MIN_BE                   0x2e, 0xc0, 6
# define SR_reserved_2e_2            0x2e, 0x30, 4
  /** Access parameters for sub-register I_AM_COORD in register @ref
RG_CSMA_SEED_1
   * @ingroup apiHalPHY230Sreg
   */
# define SR_I_AM_COORD               0x2e, 0x08, 3
  /** Access parameters for sub-register CSMA_SEED_1 in register @ref
RG_CSMA_SEED_1
   * @ingroup apiHalPHY230Sreg
   */
# define SR_CSMA_SEED_1              0x2e, 0x07, 0

#endif /* PHY230_REGISTERMAP_EXTERNAL_H */
```

Compiler_avr.h

```
/* This file has been prepared for Doxygen automatic documentation generation.*/
/*! \file
*********************************************************************
 *
 * \brief This file implements some macros that makes the IAR C-compiler and
 *       avr-gcc work with the same code base for the AVR architecture.
 *
 * \par Application note:
 *     AVR2001: AT86RF230 Software Programmer's Guide
 *
 * \par Documentation
 *     For comprehensive code documentation, supported compilers, compiler
 *     settings and supported devices see readme.html
 *
 * \author
 *     Atmel Corporation: http://www.atmel.com \n
 *     Support email: avr@atmel.com
 *
 * $Name$
 * $Revision: 613 $
 * $RCSfile$
 * $Date: 2006-04-07 14:40:07 +0200 (fr, 07 apr 2006) $  \n
 *
 * Copyright (c) 2006, Atmel Corporation All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions are met:
 *
 * 1. Redistributions of source code must retain the above copyright notice,
 * this list of conditions and the following disclaimer.
 *
 * 2. 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.
 *
 * 3. The name of ATMEL may not be used to endorse or promote products derived
 * from this software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY ATMEL ``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
EXPRESSLY AND
 * SPECIFICALLY DISCLAIMED. IN NO EVENT SHALL ATMEL 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 COMPILER_AVR_H
#define COMPILER_AVR_H

/** This macro will protect the following code from interrupts.*/
#define AVR_ENTER_CRITICAL_REGION( ) {uint8_t volatile saved_sreg = SREG;
cli( );

/** This macro must always be used in conjunction with
AVR_ENTER_CRITICAL_REGION
    so that interrupts are enabled again.*/
#define AVR_LEAVE_CRITICAL_REGION( ) SREG = saved_sreg;}

#if defined( __ICCAVR__ )

#include <inavr.h>
#include <ioavr.h>
#include <intrinsics.h>

#include "crc16.h"

#define FLASH_DECLARE(x) __flash x
#define FLASH_DECLARE(x) __flash x
#define FLASH_STRING(x) ((__flash const char *)(x))
#define PGM_READ_BYTE(x) *(x)
/**
   Perform a delay of \c us microseconds.

   The macro F_CPU is supposed to be defined to a constant defining the CPU
   clock frequency (in Hertz).

   The maximal possible delay is 262.14 ms / F_CPU in MHz.

   \note For the IAR compiler, currently F_CPU must be a
   multiple of 1000000UL (1 MHz).
 */
#define delay_us( us )   ( __delay_cycles( ( F_CPU / 1000000UL ) * ( us ) ) )

/*
 * Some preprocessor magic to allow for a header file abstraction of
 * interrupt service routine declarations for the IAR compiler.  This
 * requires the use of the C99 _Pragma() directive (rather than the
 * old #pragma one that could not be used as a macro replacement), as
 * well as two different levels of preprocessor concetanations in
 * order to do both, assign the correct interrupt vector name, as well
 * as construct a unique function name for the ISR.
 *
 * Do *NOT* try to reorder the macros below, or you'll suddenly find
 * out about all kinds of IAR bugs...
 */
#define PRAGMA(x) _Pragma( #x )
#define ISR(vec) PRAGMA( vector=vec ) __interrupt void handler_##vec(void)
#define sei( ) (__enable_interrupt( ))
#define cli( ) (__disable_interrupt( ))

#define watchdog_reset( ) (__watchdog_reset( ))

#define INLINE PRAGMA( inline=forced ) static

#elif defined( __GNUC__ )

#include <avr/io.h>
#include <avr/interrupt.h>
# include <avr/pgmspace.h>

#include <util/crc16.h>
#include <util/delay.h>

#define delay_us( us )   (_delay_us( us ))

#define INLINE static inline
#define crc_ccitt_update( crc, data ) _crc_ccitt_update( crc, data )

#define __x
#define __z

#else
#error Compiler not supported.
#endif
#endif
```

Compiler_sam7.h

```
/* This file has been prepared for Doxygen automatic documentation generation.*/
/*! \file
*********************************************************************
 *
 * \brief This file implements some macros that makes the IAR C-compiler and
 *       avr-gcc work with the same code base for the AVR architecture.
 *
 * \par Application note:
 *     AVR2001: AT86RF230 Software Programmer's Guide
 *
 * \par Documentation
 *     For comprehensive code documentation, supported compilers, compiler
 *     settings and supported devices see readme.html
 *
 * \author
 *     Atmel Corporation: http://www.atmel.com \n
 *     Support email: avr@atmel.com
 *
 * $Name$
 * $Revision: 613 $
 * $RCSfile$
 * $Date: 2006-04-07 14:40:07 +0200 (fr, 07 apr 2006) $  \n
 *
 * Copyright (c) 2006, Atmel Corporation All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions are met:
 *
 * 1. Redistributions of source code must retain the above copyright notice,
 * this list of conditions and the following disclaimer.
 *
 * 2. 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.
 *
 * 3. The name of ATMEL may not be used to endorse or promote products derived
 * from this software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY ATMEL ``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
EXPRESSLY AND
```

```
 * SPECIFICALLY DISCLAIMED. IN NO EVENT SHALL ATMEL 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 COMPILER_SAM7_H
#define COMPILER_SAM7_H

/** This macro will protect the following code from interrupts.*/
#define ARM_ENTER_CRITICAL_REGION( )

/** This macro must always be used in conjunction with
ARM_ENTER_CRITICAL_REGION
    so that interrupts are enabled again.*/
#define ARM_LEAVE_CRITICAL_REGION( )

/*These are dummy defines. Only used for AVR.*/
#define __x
#define __z

#if defined( __ICCARM__ )

#include "include/AT91SAM7S64.h" //Change to correct device.

/**
   Perform a delay of \c us microseconds.

   The macro F_CPU is supposed to be defined to a constant defining the CPU
   clock frequency (in Hertz).

   The maximal possible delay is 262.14 ms / F_CPU in MHz.

   \note For the IAR compiler, currently F_CPU must be a
```

```
  multiple of 1000000UL (1 MHz).
 */
#define delay_us( us )   ( __delay_cycles( ( F_CPU / 1000000UL ) * ( us ) ) )

/*
 * Some preprocessor magic to allow for a header file abstraction of
 * interrupt service routine declarations for the IAR compiler.  This
 * requires the use of the C99 _Pragma() directive (rather than the
 * old #pragma one that could not be used as a macro replacement), as
 * well as two different levels of preprocessor concetanations in
 * order to do both, assign the correct interrupt vector name, as well
 * as construct a unique function name for the ISR.
 *
 * Do *NOT* try to reorder the macros below, or you'll suddenly find
 * out about all kinds of IAR bugs...
 */
#define PRAGMA(x) _Pragma( #x )
#define ISR(vec) PRAGMA( vector=vec ) __interrupt void handler_##vec(void)
#define sei( ) (__enable_interrupt( ))
#define cli( ) (__disable_interrupt( ))

#define INLINE PRAGMA( inline=forced ) static

#elif defined( __GNUC__ )

#include <avr/io.h>
#include <avr/interrupt.h>
# include <avr/pgmspace.h>

#include <util/crc16.h>
#include <util/delay.h>

#define delay_us( us )   (_delay_us( us ))

#define INLINE static inline
#define crc_ccitt_update( crc, data ) _crc_ccitt_update( crc, data )

#define __x
#define __z

#else
#error Compiler not supported.
#endif
#endif
```

Config_uart.h

```
/* This file has been prepared for Doxygen automatic documentation generation.*/
/*! \file
*********************************************************************
 *
 * \brief This files contains the different build options for the wireless uart
 *        example.
 *
 * \par Application note:
 *      AVR2001: AT86RF230 Software Programmer's Guide
 *
 * \par Documentation
 *      For comprehensive code documentation, supported compilers, compiler
 *      settings and supported devices see readme.html
 *
 * \author
 *      Atmel Corporation: http://www.atmel.com \n
 *      Support email: avr@atmel.com
 *
 * $Name$
 * $Revision: 613 $
 * $RCSfile$
 * $Date: 2006-04-07 14:40:07 +0200 (fr, 07 apr 2006) $  \n
 *
 * Copyright (c) 2006, Atmel Corporation All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions are met:
 *
 * 1. Redistributions of source code must retain the above copyright notice,
 * this list of conditions and the following disclaimer.
 *
 * 2. 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.
 *
 * 3. The name of ATMEL may not be used to endorse or promote products derived
 * from this software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY ATMEL ``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
EXPRESSLY AND
```

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

#define OPERATING_CHANNEL ( 17 ) // !< Channel to transmit on. Must be between
11 and 26!

/*Chose one of these options according to the kit you want to use.*/
//#define STK541
#define RZ502

#define RX_POOL_SIZE       ( 4 ) //MUST BE GREATER THAN ZERO.
/*EOF*/
```

Config_uart_extended.h

```
/* This file has been prepared for Doxygen automatic documentation generation.*/
/*! \file
*******************************************************************
 *
 * \brief This files contains the different build options for the extended
 *        wireless uart example.
 *
 * \par Application note:
 *      AVR2001: AT86RF230 Software Programmer's Guide
 *
 * \par Documentation
 *      For comprehensive code documentation, supported compilers, compiler
 *      settings and supported devices see readme.html
 *
 * \author
 *      Atmel Corporation: http://www.atmel.com \n
 *      Support email: avr@atmel.com
 *
 * $Name$
 * $Revision: 613 $
 * $RCSfile$
 * $Date: 2006-04-07 14:40:07 +0200 (fr, 07 apr 2006) $  \n
 *
 * Copyright (c) 2006, Atmel Corporation All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions are met:
 *
 * 1. Redistributions of source code must retain the above copyright notice,
 * this list of conditions and the following disclaimer.
 *
 * 2. 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.
 *
 * 3. The name of ATMEL may not be used to endorse or promote products derived
 * from this software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY ATMEL ``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
EXPRESSLY AND
 * SPECIFICALLY DISCLAIMED. IN NO EVENT SHALL ATMEL 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_UART_EXTENDED
```

```
#define CONFIG_UART_EXTENDED

#define OPERATING_CHANNEL ( 17 ) // !< Channel to transmit on. Must be between
11 and 26!
#define PAN_ID           ( 0xBEEF ) //!< System PAN ID.

//Make sure that the two nodes are programmed with different short address
//(SHORT_ADDRESS) and destination address (DEST_ADDRESS). Do this by
commenting
//out one of the sections below. First compile for Node1 and then for Node2.

//Node1
#define SHORT_ADDRESS_NODE1 ( 0xBAAD ) //!< Short Addres for node 1 of the
link.

//Node2
#define SHORT_ADDRESS_NODE2 ( 0xACDC ) //!< Short Addres for node 2 of the
link.

/*Compile and program once for each of these options.*/
//#define NODE1

#define SHORT_ADDRESS ( SHORT_ADDRESS_NODE2 )
#define DEST_ADDRESS  ( SHORT_ADDRESS_NODE1 )

/*Chose one of these options according to the kit you want to use.*/
//#define STK541
#define RZ502

#define COM_RX_BUFFER_SIZE ( 118 ) //DO NOT ALTER!!!
#define RX_POOL_SIZE      ( 4 ) //MUST BE GREATER THAN ZERO.
#endif
/*EOF*/
```

Crc16.h

```
/* This file has been prepared for Doxygen automatic documentation generation.*/
/*! \file
*******************************************************************
 *
 * \brief
 *     This file implements the API for the implemented CCITT 16-bit CRC used
 *     by IEEE 802.15.4. This is function is included since IAR lacks this
 *     implementation. For avr-gcc it is available in avr-libc: <util/crc16.h>
 *
 * \par Application note:
```

```
 *      AVR2001: AT86RF230 Software Programmer's Guide.
 *
 * \par Documentation
 *      For comprehensive code documentation, supported compilers, compiler
 *      settings and supported devices see readme.html
 *
 * \author
 *      Atmel Corporation: http://www.atmel.com \n
 *      Support email: avr@atmel.com
 *
 * $Name$
 * $Revision: 613 $
 * $RCSfile$
 * $Date: 2006-04-07 14:40:07 +0200 (fr, 07 apr 2006) $  \n
 *
 * Copyright (c) 2006, Atmel Corporation All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions are met:
 *
 * 1. Redistributions of source code must retain the above copyright notice,
 * this list of conditions and the following disclaimer.
 *
 * 2. 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.
 *
 * 3. The name of ATMEL may not be used to endorse or promote products derived
 * from this software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY ATMEL ``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
EXPRESSLY AND
 * SPECIFICALLY DISCLAIMED. IN NO EVENT SHALL ATMEL 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 CRC_16_H
#define CRC_16_H
/*============================ INCLUDE
=========================================*/
#include <stdint.h>
#include <stdbool.h>

/*============================ MACROS
==========================================*/
/*============================ TYPDEFS
=========================================*/
/*============================ PROTOTYPES
=====================================*/
#if defined( __ICCAVR__ )
uint16_t crc_ccitt_update( uint16_t crc, uint8_t data );
#endif /* __ICCAVR__ */
#endif
/*EOF*/
```

Hal.h

```
/* This file has been prepared for Doxygen automatic documentation generation.*/
/*! \file
*****************************************************************
 *
 * \brief This file includes the correct HAL given the selected architecture.
 *
 * \par Application note:
 *      AVR2001: AT86RF230 Software Programmer's Guide
 *
 * \par Documentation
 *      For comprehensive code documentation, supported compilers, compiler
 *      settings and supported devices see readme.html
 *
 * \author
 *      Atmel Corporation: http://www.atmel.com \n
 *      Support email: avr@atmel.com
 *
 * $Name$
 * $Revision: 613 $
 * $RCSfile$
 * $Date: 2006-04-07 14:40:07 +0200 (fr, 07 apr 2006) $  \n
 *
 * Copyright (c) 2006, Atmel Corporation All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions are met:
 *
 * 1. Redistributions of source code must retain the above copyright notice,
 * this list of conditions and the following disclaimer.
 *
 * 2. 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.
 *
 * 3. The name of ATMEL may not be used to endorse or promote products derived
 * from this software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY ATMEL ``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
EXPRESSLY AND
 * SPECIFICALLY DISCLAIMED. IN NO EVENT SHALL ATMEL 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 HAL_H
#define HAL_H
/*============================ INCLUDE
=========================================*/
```

```
#if defined( AVR )
      #include "hal_avr_mega1281.h"
   #elif defined( AVR32 )
      #error Architecture not supported yet.
   #elif defined( SAM7 )
      #include "hal_sam7.h"
      //#error Architecture not supported yet.
   #else
      #error This files should only be compiled with the ARCHITECTURE set.
   #endif

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

#include "compiler.h"
/*============================ MACROS
========================================*/
#define HAL_BAT_LOW_MASK       ( 0x80 ) //!< Mask for the BAT_LOW interrupt.
#define HAL_TRX_UR_MASK        ( 0x40 ) //!< Mask for the TRX_UR interrupt.
#define HAL_TRX_END_MASK       ( 0x08 ) //!< Mask for the TRX_END interrupt.
#define HAL_RX_START_MASK      ( 0x04 ) //!< Mask for the RX_START interrupt.
#define HAL_PLL_UNLOCK_MASK    ( 0x02 ) //!< Mask for the PLL_UNLOCK
interrupt.
#define HAL_PLL_LOCK_MASK      ( 0x01 ) //!< Mask for the PLL_LOCK interrupt.

#define HAL_MIN_FRAME_LENGTH   ( 0x03 ) //!< A frame should be at least 3 bytes.
#define HAL_MAX_FRAME_LENGTH   ( 0x7F ) //!< A frame should no more than
127 bytes.
/*============================ TYPDEFS
=======================================*/
/*! \brief  This struct defines the rx data container.
 *
 *  \see hal_frame_read
 *
 *  \ingroup hal
 */
typedef struct{
   uint8_t length;
   uint8_t data[ HAL_MAX_FRAME_LENGTH ];
   uint8_t lqi;
   bool crc;
} hal_rx_frame_t;

//! RX_START event handler callback type. Is called with timestamp in IEEE 802.15.4
symbols and frame length.
```

```
typedef void (*hal_rx_start_isr_event_handler_t)(uint32_t const isr_timestamp, uint8_t
const frame_length);

//! RRX_END event handler callback type. Is called with timestamp in IEEE 802.15.4
symbols and frame length.
typedef void (*hal_trx_end_isr_event_handler_t)(uint32_t const isr_timestamp);
/*============================ PROTOTYPES
===================================*/
void hal_init( void );

void hal_reset_flags( void );
uint8_t hal_get_bat_low_flag( void );
void hal_clear_bat_low_flag( void );

uint8_t hal_get_trx_ur_flag( void );
void hal_clear_trx_ur_flag( void );

uint8_t hal_get_trx_end_flag( void );
void hal_clear_trx_end_flag( void );
hal_trx_end_isr_event_handler_t hal_get_trx_end_event_handler( void );
void hal_set_trx_end_event_handler( hal_trx_end_isr_event_handler_t
trx_end_callback_handle );
void hal_clear_trx_end_event_handler( void );

uint8_t hal_get_rx_start_flag( void );
void hal_clear_rx_start_flag( void );
hal_rx_start_isr_event_handler_t hal_get_rx_start_event_handler( void );
void hal_set_rx_start_event_handler( hal_rx_start_isr_event_handler_t
rx_start_callback_handle );
void hal_clear_rx_start_event_handler( void );

uint8_t hal_get_unknown_isr_flag( void );
void hal_clear_unknown_isr_flag( void );

uint8_t hal_get_pll_unlock_flag( void );
void hal_clear_pll_unlock_flag( void );

uint8_t hal_get_pll_lock_flag( void );
void hal_clear_pll_lock_flag( void );

uint8_t hal_register_read( uint8_t address );
void hal_register_write( uint8_t address, uint8_t value );
uint8_t hal_subregister_read( uint8_t address, uint8_t mask, uint8_t position );
void hal_subregister_write( uint8_t address, uint8_t mask, uint8_t position,
                            uint8_t value );
__z void hal_frame_read( hal_rx_frame_t *rx_frame );
__z void hal_frame_write( uint8_t *write_buffer, uint8_t length );
__z void hal_sram_read( uint8_t address, uint8_t length, uint8_t *data );
__z void hal_sram_write( uint8_t address, uint8_t length, uint8_t *data );
uint32_t hal_get_system_time( void );
#endif
/*EOF*/
```

Hal_avr_mega1281.c

```
/* This file has been prepared for Doxygen automatic documentation generation.*/
/*! \file
*********************************************************************
 *
 * \brief This file implements the HAL for the AT86RF230 radio transceiver.
 *
 *       The Hardware Abstraction Layer implements all the necessary
 *       functionality to interact with the radio transceiver:
 *        -# Register, Frame Buffer and SRAM access functions (SPI).
 *        -# Control of IO pins (SLP_TR and RST).
 *        -# Interrupt handler.
 *       This particular implementation is for the AVR micrcontroller.
 *
 * \par Application note:
 *     AVR2001: AT86RF230 Software Programmer's Guide
 *
 * \par Documentation
 *     For comprehensive code documentation, supported compilers, compiler
 *     settings and supported devices see readme.html
 *
 * \author
 *     Atmel Corporation: http://www.atmel.com \n
 *     Support email: avr@atmel.com
 *
 * $Name$
 * $Revision: 613 $
 * $RCSfile$
 * $Date: 2006-04-07 14:40:07 +0200 (fr, 07 apr 2006) $  \n
 *
 * Copyright (c) 2006, Atmel Corporation All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions are met:
 *
 * 1. Redistributions of source code must retain the above copyright notice,
 * this list of conditions and the following disclaimer.
 *
 * 2. 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.
 *
 * 3. The name of ATMEL may not be used to endorse or promote products derived
 * from this software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY ATMEL ``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
EXPRESSLY AND
 * SPECIFICALLY DISCLAIMED. IN NO EVENT SHALL ATMEL 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.

************************************************************************
******/

/*============================ INCLUDE
=========================================*/
#include <stdlib.h>

#include "at86rf230_registermap.h"

#include "compiler.h"
#include "hal.h"
/*============================ MACROS
==========================================*/

/*
 * Macros defined for the radio transceiver's access modes.
 *
 * These functions are implemented as macros since they are used very often and
 * we want to remove the function call overhead.
 */
#define HAL_DUMMY_READ         ( 0x00 ) //!< Dummy value for the SPI.

#define HAL_TRX_CMD_RW         ( 0xC0 ) //!< Register Write (short mode).
#define HAL_TRX_CMD_RR         ( 0x80 ) //!< Register Read (short mode).
#define HAL_TRX_CMD_FW         ( 0x60 ) //!< Frame Transmit Mode (long mode).
#define HAL_TRX_CMD_FR         ( 0x20 ) //!< Frame Receive Mode (long mode).
#define HAL_TRX_CMD_SW         ( 0x40 ) //!< SRAM Write.
#define HAL_TRX_CMD_SR         ( 0x00 ) //!< SRAM Read.
#define HAL_TRX_CMD_RADDRM     ( 0x7F ) //!< Register Address Mask.

#define HAL_CALCULATED_CRC_OK   ( 0 ) //!< CRC calculated over the frame
including the CRC field should be 0.
/*============================ TYPDEFS
=========================================*/
/*============================ VARIABLES
========================================*/
/*! \brief This is a file internal variable that contains the 16 MSB of the
 *        system time.
 *
 *        The system time (32-bit) is the current time in microseconds. For the
 *        AVR microcontroller implementation this is solved by using a 16-bit
 *        timer (Timer1) with a clock frequency of 1MHz. The hal_system_time is
 *        incremented when the 16-bit timer overflows, representing the 16 MSB.
 *        The timer value it self (TCNT1) is then the 16 LSB.
 *
 *  \see hal_get_system_time
 */
static uint16_t hal_system_time = 0;

/*Flag section.*/
static uint8_t volatile hal_bat_low_flag; //!< BAT_LOW flag.
static uint8_t volatile hal_trx_ur_flag; //!< TRX_UR flag.
static uint8_t volatile hal_trx_end_flag; //!< TRX_END flag.
static uint8_t volatile hal_rx_start_flag; //!< RX_START falg;
static uint8_t volatile hal_unknown_isr_flag; //!< Error, unknown interrupt event signaled
from the radio transceiver.
static uint8_t volatile hal_pll_unlock_flag; //!< PLL_UNLOCK flag.
static uint8_t volatile hal_pll_lock_flag;   //!< PLL_LOCK flag.

/*Callbacks.*/

/*! \brief This function is called when a rx_start interrupt is signaled.
 *
 *        If this function pointer is set to something else than NULL, it will
 *        be called when a RX_START event is signaled. The function takes two
 *        parameters: timestamp in IEEE 802.15.4 symbols (16 us resolution) and
 *        frame length. The event handler will be called in the interrupt domain,
 *        so the function must be kept short and not be blocking! Otherwise the
 *        system performance will be greatly degraded.
 *
 *  \see hal_set_rx_start_event_handler
 */
static hal_rx_start_isr_event_handler_t rx_start_callback;

/*! \brief This function is called when a trx_end interrupt is signaled.
 *
 *        If this function pointer is set to something else than NULL, it will
 *        be called when a TRX_END event is signaled. The function takes two
 *        parameters: timestamp in IEEE 802.15.4 symbols (16 us resolution) and
 *        frame length. The event handler will be called in the interrupt domain,
 *        so the function must be kept short and not be blocking! Otherwise the
 *        interrupt performance will be greatly degraded.
 *
 *  \see hal_set_trx_end_event_handler
 */
static hal_trx_end_isr_event_handler_t trx_end_callback;
/*============================ PROTOTYPES
===================================*/
/*============================ IMPLEMENTATION
================================*/

/*! \brief  This function initializes the Hardware Abstraction Layer.
 *
 *  \ingroup hal_avr_api
 */
void hal_init( void ){

   /*Reset variables used in file.*/
   hal_system_time = 0;
   hal_reset_flags( );

   /*IO Specific Initialization.*/
   DDR_SLP_TR |= (1 << SLP_TR); //Enable SLP_TR as output.
   DDR_RST    |= (1 << RST);    //Enable RST as output.

   /*SPI Specific Initialization.*/
   //Set SS, CLK and MOSI as output.
   HAL_DDR_SPI  |= (1 << HAL_DD_SS) | (1 << HAL_DD_SCK) | (1 <<
HAL_DD_MOSI);
```

```
   HAL_PORT_SPI |= (1 << HAL_DD_SS) | (1 << HAL_DD_SCK); //Set SS and CLK
high
   SPCR        = (1 << SPE) | (1 << MSTR); //Enable SPI module and master operation.
   SPSR        = (1 << SPI2X); //Enable doubled SPI speed in master mode.

   /*TIMER1 Specific Initialization.*/
   // TCCR1B = HAL_TCCR1B_CONFIG;       //Set clock prescaler
   // TIFR |= (1 << ICF1);             //Clear Input Capture Flag.
        //         HAL_ENABLE_OVERFLOW_INTERRUPT( ); //Enable Timer1
overflow interrupt.
   // hal_enable_trx_interrupt( );    //Enable interrupts from the radio transceiver.

        // Rewrite for using hardware interrupts (ADC 2/24/08)

        SREG  |= 0x10000000;                    // Global Interrupt Enable

        TCCR1B = HAL_TCCR1B_CONFIG;
        // TIFR |= (1 << ICF1);

        EIMSK =  0b00000000;                    // Disable INT5
        EICRB =  0b00000100;                    // Any logical Change
        EIMSK =  0b00100000;                    // Enable INT5

        EIFR  |= (1 << INTF5);                  // Clear the flag

        HAL_ENABLE_OVERFLOW_INTERRUPT( ); //Enable Timer1 overflow
interrupt.
   hal_enable_trx_interrupt( );    //Enable interrupts from the radio transceiver.
}

/*! \brief  This function reset the interrupt flags and interrupt event handlers
 *         (Callbacks) to their default value.
 *
 *  \ingroup hal_avr_api
 */
void hal_reset_flags( void ){

   AVR_ENTER_CRITICAL_REGION( );

   //Reset Flags.
   hal_bat_low_flag     = 0;
   hal_trx_ur_flag      = 0;
   hal_trx_end_flag     = 0;
   hal_rx_start_flag    = 0;
   hal_unknown_isr_flag = 0;
   hal_pll_unlock_flag  = 0;
```

```
    hal_pll_lock_flag    = 0;

    //Reset Associated Event Handlers.
    rx_start_callback = NULL;
    trx_end_callback  = NULL;

    AVR_LEAVE_CRITICAL_REGION( )
}

/*! \brief  This function returns the current value of the BAT_LOW flag.
 *
 *  The BAT_LOW flag is incremented each time a BAT_LOW event is signaled from
the
 *  radio transceiver. This way it is possible for the end user to poll the flag
 *  for new event occurances.
 *
 *  \ingroup hal_avr_api
 */
uint8_t hal_get_bat_low_flag( void ){
    return hal_bat_low_flag;
}

/*! \brief  This function clears the BAT_LOW flag.
 *
 *  \ingroup hal_avr_api
 */
void hal_clear_bat_low_flag( void ){

    AVR_ENTER_CRITICAL_REGION( );
    hal_bat_low_flag = 0;
    AVR_LEAVE_CRITICAL_REGION( );
}

/*! \brief  This function returns the current value of the TRX_UR flag.
 *
 *  The TRX_UR flag is incremented each time a TRX_UR event is signaled from the
 *  radio transceiver. This way it is possible for the end user to poll the flag
 *  for new event occurances.
 *
 *  \ingroup hal_avr_api
 */
uint8_t hal_get_trx_ur_flag( void ){
    return hal_trx_ur_flag;
}

/*! \brief  This function clears the TRX_UR flag.
```

```
 *
 *  \ingroup hal_avr_api
 */
void hal_clear_trx_ur_flag( void ){

   AVR_ENTER_CRITICAL_REGION( );
   hal_trx_ur_flag = 0;
   AVR_LEAVE_CRITICAL_REGION( );
}

/*! \brief  This function returns the current value of the TRX_END flag.
 *
 *  The TRX_END flag is incremented each time a TRX_END event is signaled from the
 *  radio transceiver. This way it is possible for the end user to poll the flag
 *  for new event occurances.
 *
 *  \ingroup hal_avr_api
 */
uint8_t hal_get_trx_end_flag( void ){
   return hal_trx_end_flag;
}

/*! \brief  This function clears the TRX_END flag.
 *
 *  \ingroup hal_avr_api
 */
void hal_clear_trx_end_flag( void ){

   AVR_ENTER_CRITICAL_REGION( );
   hal_trx_end_flag = 0;
   AVR_LEAVE_CRITICAL_REGION( );
}

/*! \brief  This function is used to set new TRX_END event handler, overriding
 *          old handler reference.
 *
 *  \ingroup hal_avr_api
 */
hal_trx_end_isr_event_handler_t hal_get_trx_end_event_handler( void ){
   return trx_end_callback;
}

/*! \brief  This function is used to set new TRX_END event handler, overriding
 *          old handler reference.
 *
 *  \ingroup hal_avr_api
 */
void hal_set_trx_end_event_handler( hal_trx_end_isr_event_handler_t
trx_end_callback_handle ){

   AVR_ENTER_CRITICAL_REGION( );
   trx_end_callback = trx_end_callback_handle;
   AVR_LEAVE_CRITICAL_REGION( );
}

/*! \brief  Remove event handler reference.
 *
 *  \ingroup hal_avr_api
 */
void hal_clear_trx_end_event_handler( void ){

   AVR_ENTER_CRITICAL_REGION( );
   trx_end_callback = NULL;
   AVR_LEAVE_CRITICAL_REGION( );
}

/*! \brief  This function returns the current value of the RX_START flag.
 *
 *  The RX_START flag is incremented each time a RX_START event is signaled from
the
 *  radio transceiver. This way it is possible for the end user to poll the flag
 *  for new event occurances.
 *
 *  \ingroup hal_avr_api
 */
uint8_t hal_get_rx_start_flag( void ){
   return hal_rx_start_flag;
}

/*! \brief  This function clears the RX_START flag.
 *
 *  \ingroup hal_avr_api
 */
void hal_clear_rx_start_flag( void ){

   AVR_ENTER_CRITICAL_REGION( );
   hal_rx_start_flag = 0;
   AVR_LEAVE_CRITICAL_REGION( );
}

/*! \brief  This function returns the active RX_START event handler
 *
 *  \return Current RX_START event handler registered.
 *  \ingroup hal_avr_api
 */
hal_rx_start_isr_event_handler_t hal_get_rx_start_event_handler( void ){
   return rx_start_callback;
}

/*! \brief  This function is used to set new RX_START event handler, overriding
 *          old handler reference.
 *
 *  \ingroup hal_avr_api
 */
void hal_set_rx_start_event_handler( hal_rx_start_isr_event_handler_t
rx_start_callback_handle ){

   AVR_ENTER_CRITICAL_REGION( );
   rx_start_callback = rx_start_callback_handle;
   AVR_LEAVE_CRITICAL_REGION( );
}

/*! \brief  Remove event handler reference.
 *
 *  \ingroup hal_avr_api
 */
void hal_clear_rx_start_event_handler( void ){

   AVR_ENTER_CRITICAL_REGION( );
   rx_start_callback = NULL;
   AVR_LEAVE_CRITICAL_REGION( );
}

/*! \brief  This function returns the current value of the UNKNOWN_ISR flag.
 *
 *  The UNKNOWN_ISR flag is incremented each time an UNKNOWN_ISR event is
signaled from the
 *  radio transceiver. This way it is possible for the end user to poll the flag
 *  for new event occurances.
 *
 *  \ingroup hal_avr_api
 */
uint8_t hal_get_unknown_isr_flag( void ){
   return hal_unknown_isr_flag;
}

/*! \brief  This function clears the UNKNOWN_ISR flag.
 *
 *  \ingroup hal_avr_api
 */
void hal_clear_unknown_isr_flag( void ){

   AVR_ENTER_CRITICAL_REGION( );
   hal_unknown_isr_flag = 0;
   AVR_LEAVE_CRITICAL_REGION( );
}

/*! \brief  This function returns the current value of the PLL_UNLOCK flag.
 *
 *  The PLL_UNLOCK flag is incremented each time a PLL_UNLOCK event is signaled
from the
 *  radio transceiver. This way it is possible for the end user to poll the flag
 *  for new event occurances.
 *
 *  \ingroup hal_avr_api
 */
uint8_t hal_get_pll_unlock_flag( void ){
   return hal_pll_unlock_flag;
}

/*! \brief  This function clears the PLL_UNLOCK flag.
 *
 *  \ingroup hal_avr_api
 */
void hal_clear_pll_unlock_flag( void ){

   AVR_ENTER_CRITICAL_REGION( );
   hal_pll_unlock_flag = 0;
   AVR_LEAVE_CRITICAL_REGION( );
}

/*! \brief  This function returns the current value of the PLL_LOCK flag.
 *
 *  The PLL_LOCK flag is incremented each time a PLL_LOCK event is signaled from
the
 *  radio transceiver. This way it is possible for the end user to poll the flag
 *  for new event occurances.
 *
 *  \ingroup hal_avr_api
 */
uint8_t hal_get_pll_lock_flag( void ){
   return hal_pll_lock_flag;
}
```

```
/*! \brief  This function clears the PLL_LOCK flag.
 *
 *  \ingroup hal_avr_api
 */
void hal_clear_pll_lock_flag( void ){

    AVR_ENTER_CRITICAL_REGION( );
    hal_pll_lock_flag = 0;
    AVR_LEAVE_CRITICAL_REGION( );
}

/*! \brief  This function reads data from one of the radio transceiver's registers.
 *
 *  \param  address Register address to read from. See datasheet for register
 *                  map.
 *
 *  \see Look at the at86rf230_registermap.h file for register address definitions.
 *
 *  \returns The actual value of the read register.
 *
 *  \ingroup hal_avr_api
 */
uint8_t hal_register_read( uint8_t address ){

    //Add the register read command to the register address.
    address &= HAL_TRX_CMD_RADDRM;
    address |= HAL_TRX_CMD_RR;

    uint8_t register_value = 0;

    AVR_ENTER_CRITICAL_REGION( );

    HAL_SS_LOW( ); //Start the SPI transaction by pulling the Slave Select low.

    /*Send Register address and read register content.*/
    SPDR = address;
    while ((SPSR & (1 << SPIF)) == 0) {;}
    register_value = SPDR;

    SPDR = register_value;
    while ((SPSR & (1 << SPIF)) == 0) {;}
    register_value = SPDR;

    HAL_SS_HIGH( ); //End the transaction by pulling the Slave Select High.

    AVR_LEAVE_CRITICAL_REGION( );
```

```
   return register_value;
}

/*! \brief  This function writes a new value to one of the radio transceiver's
 *         registers.
 *
 *  \see Look at the at86rf230_registermap.h file for register address definitions.
 *
 *  \param  address Address of register to write.
 *  \param  value   Value to write.
 *
 *  \ingroup hal_avr_api
 */
void hal_register_write( uint8_t address, uint8_t value ){

   //Add the Register Write command to the address.
   address = HAL_TRX_CMD_RW | (HAL_TRX_CMD_RADDRM & address);

   AVR_ENTER_CRITICAL_REGION( );

   HAL_SS_LOW( ); //Start the SPI transaction by pulling the Slave Select low.

   /*Send Register address and write register content.*/
   SPDR = address;
   while ((SPSR & (1 << SPIF)) == 0) {;}
   uint8_t dummy_read = SPDR;

   SPDR = value;
   while ((SPSR & (1 << SPIF)) == 0) {;}
   dummy_read = SPDR;

   HAL_SS_HIGH( ); //End the transaction by pulling the Slave Slect High.

   AVR_LEAVE_CRITICAL_REGION( );
}

/*! \brief  This function reads the value of a specific subregister.
 *
 *  \see Look at the at86rf230_registermap.h file for register and subregister
 *      definitions.
 *
 *  \param  address  Main register's address.
 *  \param  mask  Bit mask of the subregister.
 *  \param  position   Bit position of the subregister
 *  \retval Value of the read subregister.
 *
 *  \ingroup hal_avr_api
 */
uint8_t hal_subregister_read( uint8_t address, uint8_t mask, uint8_t position ){

    //Read current register value and mask out subregister.
    uint8_t register_value = hal_register_read( address );
    register_value &= mask;
    register_value >>= position; //Align subregister value.

    return register_value;
}

/*! \brief  This function writes a new value to one of the radio transceiver's
 *          subregisters.
 *
 *  \see Look at the at86rf230_registermap.h file for register and subregister
 *       definitions.
 *
 *  \param  address  Main register's address.
 *  \param  mask  Bit mask of the subregister.
 *  \param  position  Bit position of the subregister
 *  \param  value  Value to write into the subregister.
 *
 *  \ingroup hal_avr_api
 */
void hal_subregister_write( uint8_t address, uint8_t mask, uint8_t position,
                            uint8_t value ){

    //Read current register value and mask area outside the subregister.
    uint8_t register_value = hal_register_read( address );
    register_value &= ~mask;

    //Start preparing the new subregister value. shift in place and mask.
    value <<= position;
    value &= mask;

    value |= register_value; //Set the new subregister value.

    //Write the modified register value.
    hal_register_write( address, value );
}

/*! \brief  This function will upload a frame from the radio transceiver's frame
 *          buffer.
 *
 *         If the frame currently available in the radio transceiver's frame buffer
 *         is out of the defined bounds. Then the frame length, lqi value and crc
 *         be set to zero. This is done to indicate an error.
 *
 *  \param  rx_frame    Pointer to the data structure where the frame is stored.
 *
 *  \ingroup hal_avr_api
 */
__z void hal_frame_read( hal_rx_frame_t *rx_frame ){

    AVR_ENTER_CRITICAL_REGION( );

    HAL_SS_LOW( );

    /*Send frame read command.*/
    SPDR = HAL_TRX_CMD_FR;
    while ((SPSR & (1 << SPIF)) == 0) {;}
    uint8_t frame_length = SPDR;

    /*Read frame length.*/
    SPDR = frame_length;
    while ((SPSR & (1 << SPIF)) == 0) {;}
    frame_length = SPDR;

    /*Check for correct frame length.*/
    if ((frame_length >= HAL_MIN_FRAME_LENGTH) && (frame_length <=
HAL_MAX_FRAME_LENGTH)) {

        uint16_t crc = 0;
        uint8_t *rx_data = (rx_frame->data);

        rx_frame->length = frame_length; //Store frame length.

        /*Upload frame buffer to data pointer. Calculate CRC.*/
        SPDR = frame_length;
        while ((SPSR & (1 << SPIF)) == 0) {;}

        do {

            uint8_t const tempData = SPDR;
            SPDR = tempData; // Any data will do, and tempData is readily available. Saving
cycles.

            *rx_data++ = tempData;

            crc = crc_ccitt_update( crc, tempData );
            while ((SPSR & (1 << SPIF)) == 0) {;}
        } while (--frame_length > 0);

        /*Read LQI value for this frame.*/
        rx_frame->lqi = SPDR;

        HAL_SS_HIGH( );

        /*Check calculated crc, and set crc field in hal_rx_frame_t accordingly.*/
        if (crc == HAL_CALCULATED_CRC_OK) {
            rx_frame->crc = true;
        } else { rx_frame->crc = false; }
    } else {

        HAL_SS_HIGH( );

        rx_frame->length = 0;
        rx_frame->lqi    = 0;
        rx_frame->crc    = false;
    }

    AVR_LEAVE_CRITICAL_REGION( );
}

/*! \brief  This function will download a frame to the radio transceiver's frame
 *          buffer.
 *
 *  \param  write_buffer    Pointer to data that is to be written to frame buffer.
 *  \param  length          Length of data. The maximum length is 127 bytes.
 *
 *  \ingroup hal_avr_api
 */
__z void hal_frame_write( uint8_t *write_buffer, uint8_t length ){

    length &= HAL_TRX_CMD_RADDRM; //Truncate length to maximum frame length.

    AVR_ENTER_CRITICAL_REGION( );

    HAL_SS_LOW( ); //Initiate the SPI transaction.

    /*SEND FRAME WRITE COMMAND AND FRAME LENGTH.*/
    SPDR = HAL_TRX_CMD_FW;
    while ((SPSR & (1 << SPIF)) == 0) {;}
    uint8_t dummy_read = SPDR;
    SPDR = length;
    while ((SPSR & (1 << SPIF)) == 0) {;}
    dummy_read = SPDR;

    //Download to the Frame Buffer.
    do {

        SPDR = *write_buffer++;
        --length;

        while ((SPSR & (1 << SPIF)) == 0) {;}

        dummy_read = SPDR;
    } while (length > 0);

    HAL_SS_HIGH( ); //Terminate SPI transaction.

    AVR_LEAVE_CRITICAL_REGION( );
}

/*! \brief Read SRAM
 *
 * This function reads from the SRAM of the radio transceiver.
 *
 * \param address Address in the TRX's SRAM where the read burst should start
 * \param length Length of the read burst
 * \param data Pointer to buffer where data is stored.
 *
 * \ingroup hal_avr_api
 */
__z void hal_sram_read( uint8_t address, uint8_t length, uint8_t *data ){

    AVR_ENTER_CRITICAL_REGION( );

    HAL_SS_LOW( ); //Initiate the SPI transaction.

    /*Send SRAM read command.*/
    SPDR = HAL_TRX_CMD_SR;
    while ((SPSR & (1 << SPIF)) == 0) {;}
    uint8_t dummy_read = SPDR;

    /*Send address where to start reading.*/
    SPDR = address;
    while ((SPSR & (1 << SPIF)) == 0) {;}
    dummy_read = SPDR;
    /*Upload the chosen memory area.*/
    do {

        SPDR = HAL_DUMMY_READ;
        while ((SPSR & (1 << SPIF)) == 0) {;}
        *data++ = SPDR;
    } while (--length > 0);

    HAL_SS_HIGH( );

    AVR_LEAVE_CRITICAL_REGION( );
}

/*! \brief Write SRAM
 *
 * This function writes into the SRAM of the radio transceiver.
 *
 * \param address Address in the TRX's SRAM where the write burst should start
 * \param length  Length of the write burst
 * \param data    Pointer to an array of bytes that should be written
 *
 * \ingroup hal_avr_api
 */
__z void hal_sram_write( uint8_t address, uint8_t length, uint8_t *data ){

    AVR_ENTER_CRITICAL_REGION( );

    HAL_SS_LOW( );

    /*Send SRAM write command.*/
    SPDR = HAL_TRX_CMD_SW;
    while ((SPSR & (1 << SPIF)) == 0) {;}
    uint8_t dummy_read = SPDR;

    /*Send address where to start writing to.*/
    SPDR = address;
    while ((SPSR & (1 << SPIF)) == 0) {;}
    dummy_read = SPDR;

    /*Upload the chosen memory area.*/
    do {

        SPDR = *data++;
        while ((SPSR & (1 << SPIF)) == 0) {;}
        dummy_read = SPDR;
    } while (--length > 0);
    HAL_SS_HIGH( );

    AVR_LEAVE_CRITICAL_REGION( );
}

/*! \brief This function returns the system time in symbols, as defined in the
 *        IEEE 802.15.4 standard.
 *
 * \returns The system time with symbol resolution.
 *
 * \ingroup hal_avr_api
 */
uint32_t hal_get_system_time( void ){

    uint32_t system_time = hal_system_time; //Read current system time (MSB).
    system_time <<= 16;

    /*Disable interrupts. This is done since the TCNT1 value should not
      be read when the timer is running.*/
    AVR_ENTER_CRITICAL_REGION( );

    system_time |= TCNT1; //Add 16 LSB.

    AVR_LEAVE_CRITICAL_REGION( );

    //Return the system time in symbols.
    return ((system_time / HAL_US_PER_SYMBOL) & HAL_SYMBOL_MASK);
}

//This #if compile switch is used to provide a "standard" function body for the
//doxygen documentation.

ISR( INT5_vect ) {

        EIFR  |= (1 << INTF5);          // Clear INT5 interrupt (explicit)

            /*The following code reads the current system time. This is done by first
      reading the hal_system_time and then adding the 16 LSB directly from the
      TCNT1 register.
     */
    uint32_t isr_timestamp = hal_system_time;
    isr_timestamp <<= 16;
    isr_timestamp |= TCNT1;

    /*Read Interrupt source.*/
    HAL_SS_LOW( );

    /*Send Register address and read register content.*/
    SPDR = RG_IRQ_STATUS | HAL_TRX_CMD_RR;

    /* This is the second part of the convertion of system time to a 16 us time
       base. The division is moved here so we can spend less time waiting for SPI
       data.
     */
    isr_timestamp /= HAL_US_PER_SYMBOL; //Divide so that we get time in 16us
resolution.
    isr_timestamp &= HAL_SYMBOL_MASK;

    while ((SPSR & (1 << SPIF)) == 0) {;}
    uint8_t interrupt_source = SPDR; //The interrupt variable is used as a dummy read.

    SPDR = interrupt_source;
    while ((SPSR & (1 << SPIF)) == 0) {;}
    interrupt_source = SPDR; //The interrupt source is read.

    HAL_SS_HIGH( );

    /*Handle the incomming interrupt. Prioritized.*/
    if ((interrupt_source & HAL_RX_START_MASK)) {

        hal_rx_start_flag++; //Increment RX_START flag.

        if( rx_start_callback != NULL ){

            /*Read Frame length and call rx_start callback.*/
            HAL_SS_LOW( );

            SPDR = HAL_TRX_CMD_FR;
            while ((SPSR & (1 << SPIF)) == 0) {;}
            uint8_t frame_length = SPDR;

            SPDR = frame_length; //Any data will do, so frame_length is used.
            while ((SPSR & (1 << SPIF)) == 0) {;}
            frame_length = SPDR;

            HAL_SS_HIGH( );

            rx_start_callback( isr_timestamp, frame_length );
        }
    } else if (interrupt_source & HAL_TRX_END_MASK) {
        hal_trx_end_flag++; //Increment TRX_END flag.

        if( trx_end_callback != NULL ){
           trx_end_callback( isr_timestamp );
        }
    } else if (interrupt_source & HAL_TRX_UR_MASK) {
        hal_trx_ur_flag++; //Increment TRX_UR flag.
    } else if (interrupt_source & HAL_PLL_UNLOCK_MASK) {
        hal_pll_unlock_flag++; //Increment PLL_UNLOCK flag.
    } else if (interrupt_source & HAL_PLL_LOCK_MASK) {
        hal_pll_lock_flag++; //Increment PLL_LOCK flag.
    } else if (interrupt_source & HAL_BAT_LOW_MASK) {

        //Disable BAT_LOW interrupt to prevent interrupt storm. The interrupt
        //will continously be signaled when the supply voltage is less than the
        //user defined voltage threshold.
        uint8_t trx_isr_mask = hal_register_read( RG_IRQ_MASK );
        trx_isr_mask &= ~HAL_BAT_LOW_MASK;
        hal_register_write( RG_IRQ_MASK, trx_isr_mask );
        hal_bat_low_flag++; //Increment BAT_LOW flag.
    } else {
        hal_unknown_isr_flag++;  //Increment UNKNOWN_ISR flag.
    }

}
#if defined( DOXYGEN )
/*! \brief ISR for the radio IRQ line, triggered by the input capture.
 *  This is the interrupt service routine for timer1.ICIE1 input capture.
 *  It is triggered of a rising edge on the radio transceivers IRQ line.
 */


void TIMER1_CAPT_vect( void );

#else  /* !DOXYGEN */ // REWRITE FOR INT5

ISR( TIMER1_CAPT_vect ){

    /*The following code reads the current system time. This is done by first
      reading the hal_system_time and then adding the 16 LSB directly from the
      TCNT1 register.
     */
    uint32_t isr_timestamp = hal_system_time;
    isr_timestamp <<= 16;
    isr_timestamp |= TCNT1;
    /*Read Interrupt source.*/
    HAL_SS_LOW( );

    /*Send Register address and read register content.*/
    SPDR = RG_IRQ_STATUS | HAL_TRX_CMD_RR;

    /* This is the second part of the convertion of system time to a 16 us time
       base. The division is moved here so we can spend less time waiting for SPI
       data.
     */
    isr_timestamp /= HAL_US_PER_SYMBOL; //Divide so that we get time in 16us
resolution.
    isr_timestamp &= HAL_SYMBOL_MASK;

    while ((SPSR & (1 << SPIF)) == 0) {;}
    uint8_t interrupt_source = SPDR; //The interrupt variable is used as a dummy read.

    SPDR = interrupt_source;
    while ((SPSR & (1 << SPIF)) == 0) {;}
    interrupt_source = SPDR; //The interrupt source is read.

    HAL_SS_HIGH( );

    /*Handle the incomming interrupt. Prioritized.*/
    if ((interrupt_source & HAL_RX_START_MASK)) {

        hal_rx_start_flag++; //Increment RX_START flag.

        if( rx_start_callback != NULL ){

            /*Read Frame length and call rx_start callback.*/
            HAL_SS_LOW( );

            SPDR = HAL_TRX_CMD_FR;
            while ((SPSR & (1 << SPIF)) == 0) {;}
            uint8_t frame_length = SPDR;

            SPDR = frame_length; //Any data will do, so frame_length is used.
            while ((SPSR & (1 << SPIF)) == 0) {;}
            frame_length = SPDR;

            HAL_SS_HIGH( );

            rx_start_callback( isr_timestamp, frame_length );
        }
    } else if (interrupt_source & HAL_TRX_END_MASK) {
        hal_trx_end_flag++; //Increment TRX_END flag.

        if( trx_end_callback != NULL ){
            trx_end_callback( isr_timestamp );
        }
    } else if (interrupt_source & HAL_TRX_UR_MASK) {
        hal_trx_ur_flag++; //Increment TRX_UR flag.
    } else if (interrupt_source & HAL_PLL_UNLOCK_MASK) {
        hal_pll_unlock_flag++; //Increment PLL_UNLOCK flag.
    } else if (interrupt_source & HAL_PLL_LOCK_MASK) {
        hal_pll_lock_flag++; //Increment PLL_LOCK flag.
    } else if (interrupt_source & HAL_BAT_LOW_MASK) {

        //Disable BAT_LOW interrupt to prevent interrupt storm. The interrupt
        //will continously be signaled when the supply voltage is less than the
        //user defined voltage threshold.
        uint8_t trx_isr_mask = hal_register_read( RG_IRQ_MASK );
        trx_isr_mask &= ~HAL_BAT_LOW_MASK;
        hal_register_write( RG_IRQ_MASK, trx_isr_mask );
        hal_bat_low_flag++; //Increment BAT_LOW flag.
    } else {
        hal_unknown_isr_flag++;  //Increment UNKNOWN_ISR flag.
    }

}
#   endif /* defined(DOXYGEN) */

//This #if compile switch is used to provide a "standard" function body for the
//doxygen documentation.
#if defined( DOXYGEN )
/*! \brief Timer Overflow ISR
 * This is the interrupt service routine for timer1 overflow.
 */
void TIMER1_OVF_vect( void );
#else  /* !DOXYGEN */
ISR( TIMER1_OVF_vect ){
    hal_system_time++;
}
#endif
/*EOF*/
```

Hal_avr_mega1281.h

```
/* This file has been prepared for Doxygen automatic documentation generation.*/
```

```
/*! \file
*****************************************************************************
 *
 * \brief This file implements the HAL API for the AT86RF230 radio
 *        transceiver.
 *
 * \defgroup hal_avr_api Hardware Abstraction Layer API Functions
 *      This set of functions (Some defined as macros) is the API for the
 *      Hardware Abstraction Layer. These functions gives complete access to
 *      all of the low level functionality of the radio transceiver
 *      (IO, SPI and ISR).
 *
 * \defgroup hal_avr_board Hardware Abstraction Layer Board Specific Configuration
 *      This set of macros are provided so that the hal is easy to port to any
 *      AVR device and board configuration. The porting is simply done by changing
 *      the macros accordingly to the schematic and selected device. The clock
 *      settings ensure that any of the supported clock frequencies are supported
 *      seamlessly during compile time.
 *      Default pin configuration (RCBs):
 *      -#  Chip Select: PB0
 *      -#  SPI Clock Signal: PB1
 *      -#  MOSI: PB2
 *      -#  MISO: PB3
 *      -#  SLP_TR: PB4
 *      -#  RST: PB5
 *      -#  CLKM: Not Used
 *      -#  IRQ: Timer1 Input Capture pin. Could also use pin change or external
 *          interrupt.
 *
 * \par Application note:
 *      AVR2001: AT86RF230 Software Programmer's Guide
 *
 * \par Documentation
 *      For comprehensive code documentation, supported compilers, compiler
 *      settings and supported devices see readme.html
 *
 * \author
 *      Atmel Corporation: http://www.atmel.com \n
 *      Support email: avr@atmel.com
 *
 * $Name$
 * $Revision: 613 $
 * $RCSfile$
 * $Date: 2006-04-07 14:40:07 +0200 (fr, 07 apr 2006) $  \n
 *
 * Copyright (c) 2006, Atmel Corporation All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions are met:
 *
 * 1. Redistributions of source code must retain the above copyright notice,
 * this list of conditions and the following disclaimer.
 *
 * 2. 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.
 *
 * 3. The name of ATMEL may not be used to endorse or promote products derived
 * from this software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY ATMEL ``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
EXPRESSLY AND
 * SPECIFICALLY DISCLAIMED. IN NO EVENT SHALL ATMEL 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 HAL_AVR_H
#define HAL_AVR_H
/*============================ INCLUDE
======================================*/
#include <stdint.h>
#include <stdbool.h>
/*============================ MACROS
=======================================*/
```

```
/*! \brief Pin number that corresponds to the SLP_TR pin.
 *
 *  \ingroup hal_avr_board
 */
#define SLP_TR              ( 0x01 )

/*! \brief Data Direction Register that corresponds to the port where SLP_TR is
 *        connected.
 *  \ingroup hal_avr_board
 */
#define DDR_SLP_TR          ( DDRE )

/*! \brief Port (Write Access) where SLP_TR is connected.
 *  \ingroup hal_avr_board
 */
#define PORT_SLP_TR         ( PORTE )

/*! \brief Pin (Read Access) where SLP_TR is connected.
 *  \ingroup hal_avr_board
 */
#define PIN_SLP_TR          ( PINE )

/*! \brief This macro pulls the SLP_TR pin high.
 *
 *  \ingroup hal_avr_api
 */
#define hal_set_slptr_high( )         ( PORT_SLP_TR |= ( 1 << SLP_TR ) )

/*! \brief This macro pulls the SLP_TR pin low.
 *
 *  \ingroup hal_avr_api
 */
#define hal_set_slptr_low( )          ( PORT_SLP_TR &= ~( 1 << SLP_TR ) )

/*! \brief  Read current state of the SLP_TR pin (High/Low).
 *
 *  \retval 0 if the pin is low, 1 is the pin is high.
 *
 *  \ingroup hal_avr_api
 */
#define hal_get_slptr( ) ( ( PIN_SLP_TR & ( 1 << SLP_TR ) ) >> SLP_TR )

/*! \brief Pin number that corresponds to the RST pin.
 *  \ingroup hal_avr_board
 */
#define RST                 ( 0x00 )
```

```
/*! \brief Data Direction Register that corresponds to the port where RST is
 *         connected.
 *  \ingroup hal_avr_board
 */
#define DDR_RST              ( DDRE )

/*! \brief Port (Write Access) where RST is connected.
 *  \ingroup hal_avr_board
 */
#define PORT_RST             ( PORTE )

/*! \brief Pin (Read Access) where RST is connected.
 *  \ingroup hal_avr_board
 */
#define PIN_RST              ( PINE )

/*! \brief This macro pulls the RST pin high.
 *
 *  \ingroup hal_avr_api
 */
#define hal_set_rst_high( )           ( PORT_RST |= ( 1 << RST ) )

/*! \brief This macro pulls the RST pin low.
 *
 *  \ingroup hal_avr_api
 */
#define hal_set_rst_low( )            ( PORT_RST &= ~( 1 << RST ) )

/*! \brief  Read current state of the RST pin (High/Low).
 *
 *  \retval 0 if the pin is low, 1 if the pin is high.
 *
 *  \ingroup hal_avr_api
 */
#define hal_get_rst( ) ( ( PIN_RST & ( 1 << RST )  ) >> RST )

/*! \brief The slave select pin is PB0.
 *  \ingroup hal_avr_board
 */
#define HAL_SS_PIN              ( 0x00 )

/*! \brief The SPI module is located on PORTB.
 *  \ingroup hal_avr_board
 */
#define HAL_PORT_SPI            ( PORTB )
```

```
/*! \brief Data Direction Register for PORTB.
 *  \ingroup hal_avr_board
 */
#define HAL_DDR_SPI            ( DDRB )

/*! \brief Data Direction bit for SS.
 *  \ingroup hal_avr_board
 */
#define HAL_DD_SS              ( 0x00 )

/*! \brief Data Direction bit for SCK.
 *  \ingroup hal_avr_board
 */
#define HAL_DD_SCK             ( 0x01 )

/*! \brief Data Direction bit for MOSI.
 *  \ingroup hal_avr_board
 */
#define HAL_DD_MOSI            ( 0x02 )

/*! \brief Data Direction bit for MISO.
 *  \ingroup hal_avr_board
 */
#define HAL_DD_MISO            ( 0x03 )

#define HAL_SS_HIGH( ) (HAL_PORT_SPI |= ( 1 << HAL_SS_PIN )) //!< MACRO
for pulling SS high.
#define HAL_SS_LOW( )  (HAL_PORT_SPI &= ~( 1 << HAL_SS_PIN )) //!< MACRO
for pulling SS low.

/*! \brief Macros defined for HAL_TIMER1.
 *
 *  These macros are used to define the correct setupt of the AVR's Timer1, and
 *  to ensure that the hal_get_system_time function returns the system time in
 *  symbols (16 us ticks).
 *
 *  \ingroup hal_avr_board
 */

#if ( F_CPU == 16000000UL )
   #define HAL_TCCR1B_CONFIG ( ( 1 << ICES1 ) | ( 1 << CS12 ) )
   #define HAL_US_PER_SYMBOL ( 1 )
   #define HAL_SYMBOL_MASK   ( 0xFFFFffff )
#elif ( F_CPU == 8000000UL )
   #define HAL_TCCR1B_CONFIG ( ( 1 << ICES1 ) | ( 1 << CS11 ) | ( 1 << CS10 ) )
   #define HAL_US_PER_SYMBOL ( 2 )
   #define HAL_SYMBOL_MASK   ( 0x7FFFffff )
#elif ( F_CPU == 4000000UL )
   #define HAL_TCCR1B_CONFIG ( ( 1 << ICES1 ) | ( 1 << CS11 ) | ( 1 << CS10 ) )
   #define HAL_US_PER_SYMBOL ( 1 )
   #define HAL_SYMBOL_MASK   ( 0xFFFFffff )
#elif ( F_CPU == 1000000UL )
   #define HAL_TCCR1B_CONFIG ( ( 1 << ICES1 ) | ( 1 << CS11 ) )
   #define HAL_US_PER_SYMBOL ( 2 )
   #define HAL_SYMBOL_MASK   ( 0x7FFFffff )
#else
   #error "Clock speed not supported."
#endif

#define HAL_ENABLE_INPUT_CAPTURE_INTERRUPT( ) ( EIMSK =  0b00100000
)
#define HAL_DISABLE_INPUT_CAPTURE_INTERRUPT( ) ( EIMSK =  0b00000000
)

#define HAL_ENABLE_OVERFLOW_INTERRUPT( ) ( TIMSK0 |= ( 1 << TOIE1 ) )
#define HAL_DISABLE_OVERFLOW_INTERRUPT( ) ( TIMSK0 &= ~( 1 << TOIE1 )
)

/*! \brief  Enable the interrupt from the radio transceiver.
 *
 *  \ingroup hal_avr_api
 */
#define hal_enable_trx_interrupt( ) (
HAL_ENABLE_INPUT_CAPTURE_INTERRUPT( ) )

/*! \brief  Disable the interrupt from the radio transceiver.
 *
 *  \retval 0 if the pin is low, 1 if the pin is high.
 *
 *  \ingroup hal_avr_api
 */
#define hal_disable_trx_interrupt( ) (
HAL_DISABLE_INPUT_CAPTURE_INTERRUPT( ) )
/*============================ TYPDEFS
========================================*/
/*============================ PROTOTYPES
====================================*/

#endif
/*EOF*/
```

Tat.c

```
/* This file has been prepared for Doxygen automatic documentation generation.*/
/*! \file
*********************************************************************
 *
 * \brief This files implements the Transceiver Access Toolbox.
 *
 * \par Application note:
 *      AVR2001:  AT86RF230 Software Programmer's Guide
 *
 * \par Documentation
 *      For comprehensive code documentation, supported compilers, compiler
 *      settings and supported devices see readme.html
 *
 * \author
 *      Atmel Corporation: http://www.atmel.com \n
 *      Support email: avr@atmel.com
 *
 * $Name$
 * $Revision: 613 $
 * $RCSfile$
 * $Date: 2006-04-07 14:40:07 +0200 (fr, 07 apr 2006) $  \n
 *
 * Copyright (c) 2006, Atmel Corporation All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions are met:
 *
 * 1. Redistributions of source code must retain the above copyright notice,
 * this list of conditions and the following disclaimer.
 *
 * 2. 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.
 *
 * 3. The name of ATMEL may not be used to endorse or promote products derived
 * from this software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY ATMEL ``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
EXPRESSLY AND
```

```
 * SPECIFICALLY DISCLAIMED. IN NO EVENT SHALL ATMEL 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.

*********************************************************************
******/
/*============================ INCLDUE
=========================================*/
#include <stdint.h>
#include <stdbool.h>

#include "at86rf230_registermap.h"

#include "compiler.h"
#include "tat.h"
#include "hal.h"
/*============================ MACROS
==========================================*/
#define TAT_CCA_DONE_MASK     ( 1 << 7 ) //!< Mask used to check the
CCA_DONE bit.
#define TAT_CCA_IDLE_MASK     ( 1 << 6 ) //!< Mask used to check the
CCA_STATUS bit.

#define TAT_START_CCA ( 1 ) //!< Value in the CCA_REQUEST subregister that
initiate a cca.

#define TAT_TRANSMISSION_SUCCESS  ( 0 )
#define TAT_BUSY_CHANNEL         ( 3 )
#define TAT_MIN_IEEE_FRAME_LENGTH ( 5 )
/*============================ TYPEDEFS
=========================================*/

/*! \brief  This enumeration defines the necessary timing information for the
 *          AT86RF230 radio transceiver. All times are in microseconds.
 *
 *         These constants are extracted from the datasheet.
 *
 *  \ingroup tat
 */
typedef enum{

   TIME_TO_ENTER_P_ON             = 510, //!< Transition time from VCC is applied
to P_ON.
   TIME_P_ON_TO_TRX_OFF           = 510, //!< Transition time from P_ON to
TRX_OFF.
   TIME_SLEEP_TO_TRX_OFF          = 880, //!< Transition time from SLEEP to
TRX_OFF.
   TIME_RESET                     = 6,   //!< Time to hold the RST pin low during reset
   TIME_ED_MEASUREMENT            = 140, //!< Time it takes to do a ED
measurement.
   TIME_CCA                       = 140, //!< Time it takes to do a CCA.
   TIME_PLL_LOCK                  = 150, //!< Maximum time it should take for the PLL
to lock.
   TIME_FTN_TUNING                = 25,  //!< Maximum time it should take to do the
filter tuning.
   TIME_NOCLK_TO_WAKE             = 6,   //!< Transition time from *_NOCLK to
being awake.
   TIME_CMD_FORCE_TRX_OFF         = 1,    //!< Time it takes to execute the
FORCE_TRX_OFF command.
   TIME_TRX_OFF_TO_PLL_ACTIVE     = 180, //!< Transition time from TRX_OFF
to: RX_ON, PLL_ON, TX_ARET_ON and RX_AACK_ON.
   TIME_STATE_TRANSITION_PLL_ACTIVE = 1, //!< Transition time from PLL
active state to another.
}tat_trx_timing_t;
/*============================ VARIABLES
======================================*/
/*============================ PROTOTYPES
=====================================*/
static bool is_sleeping( void );

/*! \brief  Initialize the Transceiver Access Toolbox and lower layers.
 *
 *         If the initialization is successful the radio transceiver will be in
 *         TRX_OFF state.
 *
 *  \note  This function must be called prior to any of the other functions in
 *        this file! Can be called from any transceiver state.
 *
 *  \retval TAT_SUCCESS     The radio transceiver was successfully initialized
 *                         and put into the TRX_OFF state.
 *  \retval TAT_UNSUPPORTED_DEVICE  The connected device is not an Atmel
 *                          AT86RF230 radio transceiver.
 *  \retval TAT_TIMED_OUT   The radio transceiver was not able to initialize and
 *                     enter TRX_OFF state within the specified time.
 *
 *  \ingroup tat
 */
tat_status_t tat_init( void ){

   tat_status_t init_status = TAT_SUCCESS;

   delay_us( TIME_TO_ENTER_P_ON );

   //Initialize Hardware Abstraction Layer.
   hal_init( );

   tat_reset_trx( ); //Do HW reset of radio transeiver.

   //Force transition to TRX_OFF.
   hal_subregister_write( SR_TRX_CMD, CMD_FORCE_TRX_OFF );
   delay_us( TIME_P_ON_TO_TRX_OFF ); //Wait for the transition to be complete.

   if (tat_get_trx_state( ) != TRX_OFF) {
      init_status = TAT_TIMED_OUT;
   } else {

      //Read Version Number
      uint8_t version_number = hal_register_read( RG_VERSION_NUM );

      if ((version_number != RF230_REVA ) && (version_number != RF230_REVB)) {
         init_status = TAT_UNSUPPORTED_DEVICE;
      } else {

         if (hal_register_read( RG_MAN_ID_0 ) !=
SUPPORTED_MANUFACTURER_ID) {
            init_status = TAT_UNSUPPORTED_DEVICE;
         } else {
            hal_register_write( RG_IRQ_MASK,
RF230_SUPPORTED_INTERRUPT_MASK );
         } // end: if (hal_register_read( RG_MAN_ID_0 ) != ...
      } // end: if ((version_number != RF230_REVA ) ...
   } // end: if (tat_get_trx_state( ) ...

   return init_status;
}

/*! \brief  This function will return the channel used by the radio transceiver.
```

```
 *
 *  \return Current channel, 11 to 26.
 *
 *  \ingroup tat
 */
uint8_t tat_get_operating_channel( void ){
   return hal_subregister_read( SR_CHANNEL );
}

/*! \brief This function will change the operating channel.
 *
 *  \param  channel New channel to operate on. Must be between 11 and 26.
 *
 *  \retval TAT_SUCCESS New channel set.
 *  \retval TAT_WRONG_STATE Transceiver is in a state where the channel cannot
 *                      be changed (SLEEP).
 *  \retval TAT_INVALID_ARGUMENT Channel argument is out of bounds.
 *  \retval TAT_TIMED_OUT The PLL did not lock within the specified time.
 *
 *  \ingroup tat
 */
tat_status_t tat_set_operating_channel( uint8_t channel ){

   /*Do function parameter and state check.*/
   if ((channel < RF230_MIN_CHANNEL) || (channel > RF230_MAX_CHANNEL)) {
      return TAT_INVALID_ARGUMENT;
   }

   if (is_sleeping( ) == true) { return TAT_WRONG_STATE; }

   if (tat_get_operating_channel( ) == channel) { return TAT_SUCCESS; }

   /*Set new operating channel.*/
   hal_subregister_write( SR_CHANNEL, channel );

   //Read current state and wait for the PLL_LOCK interrupt if the
   //radio transceiver is in either RX_ON or PLL_ON.
   uint8_t trx_state = tat_get_trx_state( );

   if ((trx_state == RX_ON) || (trx_state == PLL_ON)) {
      delay_us( TIME_PLL_LOCK );
   }

   tat_status_t channel_set_status = TAT_TIMED_OUT;

   //Check that the channel was set properly.
```

```
    if (tat_get_operating_channel( ) == channel) {
        channel_set_status = TAT_SUCCESS;
    }

    return channel_set_status;
}

/*! \brief This function will read and return the output power level.
 *
 *  \returns 0 to 15 Current output power in "TX power settings" as defined in
 *          the radio transceiver's datasheet
 *
 *  \ingroup tat
 */
uint8_t tat_get_tx_power_level( void ){
    return hal_subregister_read( SR_TX_PWR );
}

/*! \brief This function will change the output power level.
 *
 *  \param  power_level New output power level in the "TX power settings"
 *                      as defined in the radio transceiver's datasheet.
 *
 *  \retval TAT_SUCCESS New output power set successfully.
 *  \retval TAT_INVALID_ARGUMENT The supplied function argument is out of
bounds.
 *  \retval TAT_WRONG_STATE It is not possible to change the TX power when the
 *                          device is sleeping.
 *
 *  \ingroup tat
 */
tat_status_t tat_set_tx_power_level( uint8_t power_level ){

    /*Check function parameter and state.*/
    if (power_level > TX_PWR_17_2DBM) { return TAT_INVALID_ARGUMENT; }

    if (is_sleeping( ) == true) { return TAT_WRONG_STATE; }

    /*Set new power level*/
    hal_subregister_write( SR_TX_PWR, power_level );

    return TAT_SUCCESS;
}

/*! \brief This function will measure the amount of energy sensed on the antenna
 *         pins.
 *
 *  \param ed_level  This is a pointer used to return the measured energy. The
 *                   value is only written if the function returns with
 *                   TAT_SUCCESS. The written value is 0 to 84.
 *
 *  \retval TAT_SUCCESS The detection was successful.
 *  \retval TAT_WRONG_STATE The energy detection can only be done if the radio
 *                          transceiver's state is RX_ON or BUSY_RX.
 *  \ingroup tat
 */
tat_status_t tat_do_ed_scan( uint8_t *ed_level ){

   uint8_t current_state = tat_get_trx_state( );

   /*Check state. Only possible to do ed measurement from RX_ON or BUSY_RX.*/
   if ((current_state != RX_ON ) && (current_state != BUSY_RX)) {
      return TAT_WRONG_STATE;
   }

   /*Do ED measurement.*/
   //Initiate the measurement by writing to the PHY_ED_LEVEL register.
   hal_register_write( RG_PHY_ED_LEVEL, 0 );
   delay_us( TIME_ED_MEASUREMENT ); //Wait 128 us + 12 us computation time.
   *ed_level = hal_register_read( RG_PHY_ED_LEVEL ); //Write ED level to pointer.

   return TAT_SUCCESS;
}

/*! \brief This function returns the current CCA mode used.
 *
 *  \return CCA mode currently used, 0 to 3.
 *
 *  \ingroup tat
 */
uint8_t tat_get_cca_mode( void ){
   return hal_subregister_read( SR_CCA_MODE );
}

/*! \brief This function returns the current ED threshold used by the CCA algorithm.
 *
 *  \return Current ED threshold, 0 to 15.
 *
 *  \ingroup tat
 */
uint8_t tat_get_ed_threshold( void ){
   return hal_subregister_read( SR_CCA_ED_THRES );
}

/*! \brief This function will configure the Clear Channel Assessment algorithm.
 *
 *  \param  mode Three modes are available: Energy above threshold, carrier
 *              sense only and carrier sense with energy above threshold.
 *  \param  ed_threshold Above this energy threshold the channel is assumed to be
 *                   busy. The threshold is given in positive dBm values.
 *                   Ex. -91 dBm gives a csThreshold of 91. Value range for
 *                   the variable is [61 to 91]. Only valid for the CCA_ED
 *                   and CCA_CARRIER_SENSE_ED modes.
 *
 *  \retval TAT_SUCCESS Mode and its parameters successfully changed.
 *  \retval TAT_WRONG_STATE This function cannot be called in the SLEEP state.
 *  \retval TAT_INVALID_ARGUMENT If one of the three function arguments are out
 *                           of bounds.
 *
 *  \ingroup tat
 */
tat_status_t tat_set_cca_mode( uint8_t mode, uint8_t ed_threshold ){

   /*Check function parameters and state.*/
   if ((mode != CCA_ED) && (mode != CCA_CARRIER_SENSE) &&
      (mode != CCA_CARRIER_SENSE_WITH_ED)) {
      return TAT_INVALID_ARGUMENT;
   }

   //Ensure that the ED threshold is within bounds.
   if (ed_threshold > RF230_MAX_ED_THRESHOLD) { return
TAT_INVALID_ARGUMENT; }

   //Ensure that the radio transceiver is not sleeping.
   if (is_sleeping( ) == true) { return TAT_WRONG_STATE; }

   /*Change cca mode and ed threshold.*/
   hal_subregister_write( SR_CCA_MODE, mode );
   hal_subregister_write( SR_CCA_ED_THRES, ed_threshold );

   return TAT_SUCCESS;
}

/*! \brief  This function will perform a Clear Channel Assessment.
 *
 *  \note The state after the cca mesasurement will be RX_ON.
 *
 *  \retval TAT_CCA_IDLE  Channel is ready for transmission.
 *  \retval TAT_CCA_BUSY  Channel is not ready for transmission.
 *  \retval TAT_WRONG_STATE CCA can only be done in PLL_ON.
 *  \retval TAT_TIMED_OUT CCA algorithm timed out.
 *
 *  \ingroup tat
 */
tat_status_t tat_do_cca( void ){

   /*Check state. CCA measurement only possible form PLL_ON state.*/
   if (tat_get_trx_state( ) != PLL_ON) { return TAT_WRONG_STATE; }

   /*Ensure that it is possible to enter RX_ON*/
   if (tat_set_trx_state( RX_ON ) != TAT_SUCCESS) { return TAT_TIMED_OUT; }

   //The CCA is initiated by writing 1 to the CCA_REQUEST subregister.
   hal_subregister_write( SR_CCA_REQUEST, TAT_START_CCA );

   //The CCA is measured over 128 us + 12 us computation time.
   delay_us( TIME_CCA );

   uint8_t status = hal_register_read( RG_TRX_STATUS ); //Read cca status.
   tat_status_t cca_status = TAT_CCA_BUSY; //Return variable.

   //Check if the algorithm finished -> CCA_DONE == 1.
   if ((status & TAT_CCA_DONE_MASK) != TAT_CCA_DONE_MASK) {
      cca_status = TAT_TIMED_OUT;
   } else {

      //CCA done, but check if the channel is busy or not.
      if ((status & TAT_CCA_IDLE_MASK) != TAT_CCA_IDLE_MASK) {
         cca_status = TAT_CCA_BUSY;
      } else {
         cca_status = TAT_CCA_IDLE;
      } // end: if ((status & TAT_CCA_IDLE_MASK) ...
   } // end: if ((status & TAT_CCA_DONE_MASK) ...

   return cca_status;
}

/*! \brief This function returns the Received Signal Strength Indication.
 *
 *  \note This function should only be called from the: RX_ON and BUSY_RX. This
 *        can be ensured by reading the current state of the radio transceiver
 *        before executing this function!
 *  \param rssi Pointer to memory location where RSSI value should be written.
 *  \retval TAT_SUCCESS The RSSI measurement was successful.
 *  \retval TAT_WRONG_STATE The radio transceiver is not in RX_ON or BUSY_RX.
 *
 *  \ingroup tat
 */
tat_status_t tat_get_rssi_value( uint8_t *rssi ){

   uint8_t current_state = tat_get_trx_state( );
   tat_status_t retval = TAT_WRONG_STATE;

   /*The RSSI measurement should only be done in RX_ON or BUSY_RX.*/
   if ((current_state == RX_ON) || (current_state == BUSY_RX)) {

      *rssi = hal_subregister_read( SR_RSSI );
      retval = TAT_SUCCESS;
   }

   return retval;
}

/*! \brief This function returns the current threshold volatge used by the
 *         battery monitor (BATMON_VTH).
 *
 *  \note This function can not be called from P_ON or SLEEP. This is ensured
 *        by reading the device state before calling this function.
 *
 *  \return Current threshold voltage, 0 to 15.
 *
 *  \ingroup tat
 */
uint8_t tat_batmon_get_voltage_threshold( void ){
   return hal_subregister_read( SR_BATMON_VTH );
}

/*! \brief This function returns if high or low voltage range is used.
 *
 *  \note This function can not be called from P_ON or SLEEP. This is ensured
 *        by reading the device state before calling this function.
 *
 *  \retval 0 Low voltage range selected.
 *  \retval 1 High voltage range selected.
 *
 *  \ingroup tat
 */
uint8_t tat_batmon_get_voltage_range( void ){
   return hal_subregister_read( SR_BATMON_HR );
}
```

```
/*! \brief This function is used to configure the battery monitor module
 *
 *  \param range True means high voltage range and false low voltage range.
 *  \param voltage_threshold The datasheet defines 16 voltage levels for both
 *                       low and high range.
 *  \retval TAT_SUCCESS Battery monitor configured
 *  \retval TAT_WRONG_STATE The device is sleeping.
 *  \retval TAT_INVALID_ARGUMENT The voltage_threshold parameter is out of
 *                            bounds (Not within [0 - 15]).
 *  \ingroup tat
 */
tat_status_t tat_batmon_configure( bool range, uint8_t voltage_threshold ){

    /*Check function parameters and state.*/
    if (voltage_threshold > BATTERY_MONITOR_HIGHEST_VOLTAGE) {
        return TAT_INVALID_ARGUMENT;
    }

    if (is_sleeping( ) == true) { return TAT_WRONG_STATE; }

    /*Write new voltage range and voltage level.*/
    if (range == true) {
        hal_subregister_write( SR_BATMON_HR,
BATTERY_MONITOR_HIGH_VOLTAGE );
    } else {
        hal_subregister_write( SR_BATMON_HR,
BATTERY_MONITOR_LOW_VOLTAGE );
    } // end: if (range == true) ...

    hal_subregister_write( SR_BATMON_VTH, voltage_threshold );

    return TAT_SUCCESS;
}

/*! \brief This function returns the status of the Battery Monitor module.
 *
 *  \note This function can not be called from P_ON or SLEEP. This is ensured
 *        by reading the device state before calling this function.
 *
 *  \retval TAT_BAT_LOW Battery voltage is below the programmed threshold.
 *  \retval TAT_BAT_OK Battery voltage is above the programmed threshold.
 *
 *  \ingroup tat
 */
tat_status_t tat_batmon_get_status( void ){
    tat_status_t batmon_status = TAT_BAT_LOW;

    if (hal_subregister_read( SR_BATMON_OK ) !=
        BATTERY_MONITOR_VOLTAGE_UNDER_THRESHOLD) {
        batmon_status = TAT_BAT_OK;
    }

    return batmon_status;
}

/*! \brief This function returns the current clock setting for the CLKM pin.
 *
 *  \retval CLKM_DISABLED CLKM pin is disabled.
 *  \retval CLKM_1MHZ CLKM pin is prescaled to 1 MHz.
 *  \retval CLKM_2MHZ CLKM pin is prescaled to 2 MHz.
 *  \retval CLKM_4MHZ CLKM pin is prescaled to 4 MHz.
 *  \retval CLKM_8MHZ CLKM pin is prescaled to 8 MHz.
 *  \retval CLKM_16MHZ CLKM pin is not prescaled. Output is 16 MHz.
 *
 *  \ingroup tat
 */
uint8_t tat_get_clock_speed( void ){
    return hal_subregister_read( SR_CLKM_CTRL );
}

/*! \brief This function changes the prescaler on the CLKM pin.
 *
 *  \param direct   This boolean variable is used to determine if the frequency
 *                  of the CLKM pin shall be changed directly or not. If direct
 *                  equals true, the frequency will be changed directly. This is
 *                  fine if the CLKM signal is used to drive a timer etc. on the
 *                  connected microcontroller. However, the CLKM signal can also
 *                  be used to clock the microcontroller itself. In this situation
 *                  it is possible to change the CLKM frequency indirectly
 *                  (direct == false). When the direct argument equlas false, the
 *                  CLKM frequency will be changed first after the radio transceiver
 *                  has been taken to SLEEP and awaken again.
 *  \param clock_speed This parameter can be one of the following constants:
 *                    CLKM_DISABLED, CLKM_1MHZ, CLKM_2MHZ, CLKM_4MHZ,
CLKM_8MHZ
 *                    or CLKM_16MHZ.
 *
 *  \retval TAT_SUCCESS Clock speed updated. New state is TRX_OFF.
 *  \retval TAT_INVALID_ARGUMENT Requested clock speed is out of bounds.
 *
 * \ingroup tat
 */
tat_status_t tat_set_clock_speed( bool direct, uint8_t clock_speed ){

    /*Check function parameter and current clock speed.*/
    if (clock_speed > CLKM_16MHZ) { return TAT_INVALID_ARGUMENT; }

    if (tat_get_clock_speed( ) == clock_speed) { return TAT_SUCCESS; }

    /*Select to change the CLKM frequency directly or after returning from SLEEP.*/
    if (direct == false) {
        hal_subregister_write( SR_CLKM_SHA_SEL, 1 );
    } else {
        hal_subregister_write( SR_CLKM_SHA_SEL, 0 );
    } // end: if (direct == false) ...

    hal_subregister_write( SR_CLKM_CTRL, clock_speed );

    return TAT_SUCCESS;
}

/*! \brief  This function calibrates the Single Side Band Filter.
 *
 *  \retval     TAT_SUCCESS    Filter is calibrated.
 *  \retval     TAT_TIMED_OUT  The calibration could not be completed within time.
 *  \retval     TAT_WRONG_STATE This function can only be called from TRX_OFF or
 *             PLL_ON.
 *
 *  \ingroup tat
 */
tat_status_t tat_calibrate_filter( void ){

    /*Check current state. Only possible to do filter calibration from TRX_OFF or
PLL_ON.*/
    uint8_t trx_state = tat_get_trx_state( );

    if ((trx_state != TRX_OFF ) && (trx_state != PLL_ON)) { return
TAT_WRONG_STATE; }

    //Start the tuning algorithm by writing one to the FTN_START subregister.
    hal_subregister_write( SR_FTN_START, 1 );
    delay_us( TIME_FTN_TUNING ); //Wait for the calibration to finish.

    tat_status_t filter_calibration_status = TAT_TIMED_OUT;

    //Verify the calibration result.
```

```
    if (hal_subregister_read( SR_FTN_START ) == FTN_CALIBRATION_DONE) {
        filter_calibration_status = TAT_SUCCESS;
    }

    return filter_calibration_status;
}

/*! \brief  This function calibrates the PLL.
 *
 *  \retval     TAT_SUCCESS    PLL Center Frequency and Delay Cell is calibrated.
 *  \retval     TAT_TIMED_OUT  The calibration could not be completed within time.
 *  \retval     TAT_WRONG_STATE This function can only be called from PLL_ON.
 *
 *  \ingroup tat
 */
tat_status_t tat_calibrate_pll( void ){

    /*Check current state. Only possible to calibrate PLL from PLL_ON state*/
    if (tat_get_trx_state( ) != PLL_ON) { return TAT_WRONG_STATE; }

    //Initiate the DCU and CF calibration loops.
    hal_subregister_write( SR_PLL_DCU_START, 1 );
    hal_subregister_write( SR_PLL_CF_START, 1 );

    //Wait maximum 150 us for the PLL to lock.
    hal_clear_pll_lock_flag( );
    delay_us( TIME_PLL_LOCK );

    tat_status_t pll_calibration_status = TAT_TIMED_OUT;

    if (hal_get_pll_lock_flag( ) > 0) {

        if (hal_subregister_read( SR_PLL_DCU_START ) ==
PLL_DCU_CALIBRATION_DONE) {

            if (hal_subregister_read( SR_PLL_CF_START ) ==
PLL_CF_CALIBRATION_DONE) {
                pll_calibration_status = TAT_SUCCESS;
            } // end: if (hal_subregister_read( SR_PLL_CF_START ) ...
        } // end: if (hal_subregister_read( SR_PLL_DCU_START ) ...
    } // end: if ((hal_get_pll_lock_flag( ) ...

    return pll_calibration_status;
}

/*! \brief  This function return the Radio Transceivers current state.
```

```
 *
 *  \retval    P_ON             When the external supply voltage (VDD) is
 *                        first supplied to the transceiver IC, the
 *                        system is in the P_ON (Poweron) mode.
 *  \retval    BUSY_RX          The radio transceiver is busy receiving a
 *                        frame.
 *  \retval    BUSY_TX          The radio transceiver is busy transmitting a
 *                        frame.
 *  \retval    RX_ON            The RX_ON mode enables the analog and digital
 *                        receiver blocks and the PLL frequency
 *                        synthesizer.
 *  \retval    TRX_OFF          In this mode, the SPI module and crystal
 *                        oscillator are active.
 *  \retval    PLL_ON           Entering the PLL_ON mode from TRX_OFF will
 *                        first enable the analog voltage regulator. The
 *                        transceiver is ready to transmit a frame.
 *  \retval    BUSY_RX_AACK     The radio was in RX_AACK_ON mode and
received
 *                        the Start of Frame Delimiter (SFD). State
 *                        transition to BUSY_RX_AACK is done if the SFD
 *                        is valid.
 *  \retval    BUSY_TX_ARET     The radio transceiver is busy handling the
 *                        auto retry mechanism.
 *  \retval    RX_AACK_ON       The auto acknowledge mode of the radio is
 *                        enabled and it is waiting for an incomming
 *                        frame.
 *  \retval    TX_ARET_ON       The auto retry mechanism is enabled and the
 *                        radio transceiver is waiting for the user to
 *                        send the TX_START command.
 *  \retval    RX_ON_NOCLK      The radio transceiver is listening for
 *                        incomming frames, but the CLKM is disabled so
 *                        that the controller could be sleeping.
 *                        However, this is only true if the controller
 *                        is run from the clock output of the radio.
 *  \retval    RX_AACK_ON_NOCLK   Same as the RX_ON_NOCLK state, but with
the
 *                        auto acknowledge module turned on.
 *  \retval    BUSY_RX_AACK_NOCLK Same as BUSY_RX_AACK, but the controller
 *                        could be sleeping since the CLKM pin is
 *                        disabled.
 *  \retval    STATE_TRANSITION   The radio transceiver's state machine is in
 *                        transition between two states.
 *
 *  \ingroup tat
 */
uint8_t tat_get_trx_state( void ){
    return hal_subregister_read( SR_TRX_STATUS );
}

/*! \brief  This function checks if the radio transceiver is sleeping.
 *
 *  \retval     true    The radio transceiver is in SLEEP or one of the *_NOCLK
 *                  states.
 *  \retval     false   The radio transceiver is not sleeping.
 *
 *  \ingroup tat
 */
static bool is_sleeping( void ){

    bool sleeping = false;

    //The radio transceiver will be at SLEEP or one of the *_NOCLK states only if
    //the SLP_TR pin is high.
    if (hal_get_slptr( ) != 0) {
        sleeping = true;
    }

    return sleeping;
}

/*! \brief  This function will change the current state of the radio
 *          transceiver's internal state machine.
 *
 *  \param     new_state        Here is a list of possible states:
 *           - RX_ON        Requested transition to RX_ON state.
 *           - TRX_OFF      Requested transition to TRX_OFF state.
 *           - PLL_ON       Requested transition to PLL_ON state.
 *           - RX_AACK_ON   Requested transition to RX_AACK_ON state.
 *           - TX_ARET_ON   Requested transition to TX_ARET_ON state.
 *
 *  \retval    TAT_SUCCESS          Requested state transition completed
 *                           successfully.
 *  \retval    TAT_INVALID_ARGUMENT Supplied function parameter out of bounds.
 *  \retval    TAT_WRONG_STATE      Illegal state to do transition from.
 *  \retval    TAT_BUSY_STATE       The radio transceiver is busy.
 *  \retval    TAT_TIMED_OUT        The state transition could not be completed
 *                           within resonable time.
 *
 *  \ingroup tat
 */
tat_status_t tat_set_trx_state( uint8_t new_state ){
```

```
/*Check function paramter and current state of the radio transceiver.*/
if (!((new_state == TRX_OFF ) || (new_state == RX_ON) || (new_state == PLL_ON) ||
   (new_state == RX_AACK_ON ) || (new_state == TX_ARET_ON ))) {

   return TAT_INVALID_ARGUMENT;
}

if (is_sleeping( ) == true) { return TAT_WRONG_STATE; }

uint8_t original_state = tat_get_trx_state( );

if ((original_state == BUSY_RX ) || (original_state == BUSY_TX) ||
   (original_state == BUSY_RX_AACK) || (original_state == BUSY_TX_ARET)) {
   return TAT_BUSY_STATE;
}

if (new_state == original_state) { return TAT_SUCCESS; }

//At this point it is clear that the requested new_state is:
//TRX_OFF, RX_ON, PLL_ON, RX_AACK_ON or TX_ARET_ON.

//The radio transceiver can be in one of the following states:
//TRX_OFF, RX_ON, PLL_ON, RX_AACK_ON, TX_ARET_ON.
if( new_state == TRX_OFF ){
   tat_reset_state_machine( ); //Go to TRX_OFF from any state.
} else {

   //It is not allowed to go from RX_AACK_ON or TX_AACK_ON and directly to
   //TX_AACK_ON or RX_AACK_ON respectively. Need to go via RX_ON or
PLL_ON.
   if ((new_state == TX_ARET_ON) && (original_state == RX_AACK_ON)) {

      //First do intermediate state transition to PLL_ON, then to TX_ARET_ON.
      //The final state transition to TX_ARET_ON is handled after the if-else if.
      hal_subregister_write( SR_TRX_CMD, PLL_ON );
      delay_us( TIME_STATE_TRANSITION_PLL_ACTIVE );
   } else if ((new_state == RX_AACK_ON) && (original_state == TX_ARET_ON)) {

      //First do intermediate state transition to RX_ON, then to RX_AACK_ON.
      //The final state transition to RX_AACK_ON is handled after the if-else if.
      hal_subregister_write( SR_TRX_CMD, RX_ON );
      delay_us( TIME_STATE_TRANSITION_PLL_ACTIVE );
   }

   //Any other state transition can be done directly.
   hal_subregister_write( SR_TRX_CMD, new_state );
```

```
        //When the PLL is active most states can be reached in 1us. However, from
        //TRX_OFF the PLL needs time to activate.
        if (original_state == TRX_OFF) {
            delay_us( TIME_TRX_OFF_TO_PLL_ACTIVE );
        } else {
            delay_us( TIME_STATE_TRANSITION_PLL_ACTIVE );
        } // end: if (original_state == TRX_OFF) ...
    } // end: if( new_state == TRX_OFF ) ...

    /*Verify state transition.*/
    tat_status_t set_state_status = TAT_TIMED_OUT;

    if( tat_get_trx_state( ) == new_state ){ set_state_status = TAT_SUCCESS; }

    return set_state_status;
}

/*! \brief  This function will put the radio transceiver to sleep.
 *
 *  \retval    TAT_SUCCESS          Sleep mode entered successfully.
 *  \retval    TAT_TIMED_OUT        The transition to TRX_OFF took too long.
 *
 *  \ingroup tat
 */
tat_status_t tat_enter_sleep_mode( void ){

    if (is_sleeping( ) == true) { return TAT_SUCCESS; }

    tat_reset_state_machine( ); //Force the device into TRX_OFF.

    tat_status_t enter_sleep_status = TAT_TIMED_OUT;

    if (tat_get_trx_state( ) == TRX_OFF) {

        //Enter Sleep.
        hal_set_slptr_high( );
        enter_sleep_status = TAT_SUCCESS;
    }

    return enter_sleep_status;
}

/*! \brief  This function will take the radio transceiver from sleep mode and
 *          put it into the TRX_OFF state.
 *
 *  \retval    TAT_SUCCESS          Left sleep mode and entered TRX_OFF state.
 *  \retval    TAT_TIMED_OUT        Transition to TRX_OFF state timed out.
 *
 *  \ingroup tat
 */
tat_status_t tat_leave_sleep_mode( void ){

    //Check if the radio transceiver is actually sleeping.
    if (is_sleeping( ) == false) { return TAT_SUCCESS; }

    hal_set_slptr_low( );
    delay_us( TIME_SLEEP_TO_TRX_OFF );

    tat_status_t leave_sleep_status = TAT_TIMED_OUT;

    //Ensure that the radio transceiver is in the TRX_OFF state.
    if (tat_get_trx_state( ) == TRX_OFF) {
        leave_sleep_status = TAT_SUCCESS;
    }

    return leave_sleep_status;
}

/*! \brief  This function will reset the state machine (to TRX_OFF) from any of
 *          its states, except for the SLEEP state.
 *
 *  \ingroup tat
 */
void tat_reset_state_machine( void ){

    hal_set_slptr_low( );
    delay_us( TIME_NOCLK_TO_WAKE );
    hal_subregister_write( SR_TRX_CMD, CMD_FORCE_TRX_OFF );
    delay_us( TIME_CMD_FORCE_TRX_OFF );
}

/*! \brief  This function will reset all the registers and the state machine of
 *          the radio transceiver.
 *
 *  \ingroup tat
 */
void tat_reset_trx( void ){

    hal_set_rst_low( );
    hal_set_slptr_low( );
    delay_us( TIME_RESET );
```

```
   hal_set_rst_high( );
}

/*! \brief  This function will enable or disable automatic CRC during frame
 *         transmission.
 *
 *  \param  auto_crc_on If this parameter equals true auto CRC will be used for
 *                  all frames to be transmitted. The framelength must be
 *                  increased by two bytes (16 bit CRC). If the parameter equals
 *                  false, the automatic CRC will be disabled.
 *
 *  \ingroup tat
 */
void tat_use_auto_tx_crc( bool auto_crc_on ){

   if (auto_crc_on == true) {
      hal_subregister_write( SR_TX_AUTO_CRC_ON, 1 );
   } else {
      hal_subregister_write( SR_TX_AUTO_CRC_ON, 0 );
   } // end: if (auto_crc_on == true) ...
}

/*! \brief  This function will download a frame to the radio transceiver's
 *         transmit buffer and send it.
 *
 *  \param  data_length Length of the frame to be transmitted. 1 to 128 bytes are the valid
lengths.
 *  \param  *data   Pointer to the data to transmit
 *
 *  \retval TAT_SUCCESS Frame downloaded and sent successfully.
 *  \retval TAT_INVALID_ARGUMENT If the dataLength is 0 byte or more than 127
 *                         bytes the frame will not be sent.
 *  \retval TAT_WRONG_STATE It is only possible to use this function in the
 *                      PLL_ON and TX_ARET_ON state. If any other state is
 *                      detected this error message will be returned.
 *
 *  \ingroup tat
 */
__x tat_status_t tat_send_data( uint8_t data_length, uint8_t *data ){

   /*Check function parameters and current state.*/
   if (data_length > RF230_MAX_TX_FRAME_LENGTH) { return
TAT_INVALID_ARGUMENT; }

   if ((tat_get_trx_state( ) != PLL_ON)) { return TAT_WRONG_STATE; }
```

```
    /*Do frame transmission.*/
    //Toggle the SLP_TR pin to initiate the frame transmission.
    hal_set_slptr_high( );
    hal_set_slptr_low( );

    hal_frame_write( data, data_length ); //Then write data to the frame buffer.

    return TAT_SUCCESS;
}

/*! \brief  This function will read the I_AM_COORD sub register.
 *
 *  \retval 0 Not coordinator.
 *  \retval 1 Coordinator role enabled.
 *
 *  \ingroup tat
 */
uint8_t tat_get_device_role( void ){
    return hal_subregister_read( SR_I_AM_COORD);
}

/*! \brief  This function will set the I_AM_COORD sub register.
 *
 *  \param[in] i_am_coordinator If this parameter is true, the associated
 *                              coordinator role will be enabled in the radio
 *                              transceiver's address filter.
 *                              False disables the same feature.
 *  \ingroup tat
 */
void tat_set_device_role( bool i_am_coordinator ){

    if (i_am_coordinator == true) {
        hal_subregister_write( SR_I_AM_COORD, 0);
    } else {
        hal_subregister_write( SR_I_AM_COORD, 0);
    } // end: if (i_am_coordinator == true) ...
}

/*! \brief  This function will return the PANID used by the address filter.
 *
 *  \retval Any value from 0 to 0xFFFF.
 *
 *  \ingroup tat
 */
uint16_t tat_get_pan_id( void ){
    uint8_t pan_id_15_8 = hal_register_read( RG_PAN_ID_1 ); // Read pan_id_15_8.
    uint8_t pan_id_7_0 = hal_register_read( RG_PAN_ID_0 ); // Read pan_id_7_0.

    uint16_t pan_id = ((uint16_t)(pan_id_15_8 << 8)) | pan_id_7_0;

    return pan_id;
}

/*! \brief  This function will set the PANID used by the address filter.
 *
 *  \param  new_pan_id Desired PANID. Can be any value from 0x0000 to 0xFFFF
 *
 *  \ingroup tat
 */
void tat_set_pan_id( uint16_t new_pan_id ){

    uint8_t pan_byte = new_pan_id & 0xFF; // Extract new_pan_id_7_0.
    hal_register_write( RG_PAN_ID_0, pan_byte );

    pan_byte = (new_pan_id >> 8*1) & 0xFF;  // Extract new_pan_id_15_8.
    hal_register_write( RG_PAN_ID_1, pan_byte );
}

/*! \brief  This function will return the current short address used by the
 *          address filter.
 *
 *  \retval Any value from 0x0000 to 0xFFFF
 *
 *  \ingroup tat
 */
uint16_t tat_get_short_address( void ){

    uint8_t short_address_15_8 = hal_register_read( RG_SHORT_ADDR_1 ); // Read
short_address_15_8.
    uint8_t short_address_7_0  = hal_register_read( RG_SHORT_ADDR_1 ); // Read
short_address_7_0.

    uint16_t short_address = ((uint16_t)(short_address_15_8 << 8)) | short_address_7_0;

    return short_address;
}

/*! \brief  This function will set the short address used by the address filter.
 *
 *  \param  new_short_address Short address to be used by the address filter.
 *
 *  \ingroup tat
 */
void tat_set_short_address( uint16_t new_short_address ){

    uint8_t short_address_byte = new_short_address & 0xFF; // Extract
short_address_7_0.
    hal_register_write( RG_SHORT_ADDR_0, short_address_byte );

    short_address_byte = (new_short_address >> 8*1) & 0xFF; // Extract
short_address_15_8.
    hal_register_write( RG_SHORT_ADDR_1, short_address_byte );
}

/*! \brief  This function will read the extended address used by the address
 *          filter.
 *
 *  \note In this function a pointer is used to convey the 64-bit result, since
 *        it is very inefficient to use the stack for this.
 *
 *  \return Extended Address, any 64-bit value.
 *
 *  \ingroup tat
 */
__x void tat_get_extended_address( uint8_t *extended_address ){

    *extended_address++ = hal_register_read( RG_IEEE_ADDR_7 ); // Read
ieee_address_63_56.
    *extended_address++ = hal_register_read( RG_IEEE_ADDR_6 ); // Read
ieee_address_55_48.
    *extended_address++ = hal_register_read( RG_IEEE_ADDR_5 ); // Read
ieee_address_47_40.
    *extended_address++ = hal_register_read( RG_IEEE_ADDR_4 ); // Read
ieee_address_39_32.
    *extended_address++ = hal_register_read( RG_IEEE_ADDR_3 ); // Read
ieee_address_31_24.
    *extended_address++ = hal_register_read( RG_IEEE_ADDR_2 ); // Read
ieee_address_23_16.
    *extended_address++ = hal_register_read( RG_IEEE_ADDR_1 ); // Read
ieee_address_15_8.
    *extended_address   = hal_register_read( RG_IEEE_ADDR_0 ); // Read
ieee_address_7_0.
}

/*! \brief  This function will set a new extended address to be used by the
 *          address filter.
 *
 *  \param  new_extended_address Extended address to be used by the address filter.
 *
 *  \ingroup tat
 */
__x void tat_set_extended_address( uint8_t *extended_address ){

    hal_register_write( RG_IEEE_ADDR_7, *extended_address++ );
    hal_register_write( RG_IEEE_ADDR_6, *extended_address++ );
    hal_register_write( RG_IEEE_ADDR_5, *extended_address++ );
    hal_register_write( RG_IEEE_ADDR_4, *extended_address++ );
    hal_register_write( RG_IEEE_ADDR_3, *extended_address++ );
    hal_register_write( RG_IEEE_ADDR_2, *extended_address++ );
    hal_register_write( RG_IEEE_ADDR_1, *extended_address++ );
    hal_register_write( RG_IEEE_ADDR_0, *extended_address++ );
}

/*! \brief  This function will configure the CSMA algorithm used by the radio
 *          transceiver when transmitting data from TX_ARET_ON state.
 *
 *  \param  seed0 Lower 8 bits of the seed used for the random number generator
 *                in the CSMA algorithm. Value range: 0 to 255.
 *  \param  be_csma_seed1 Is a combined argument of the MIN_BE,
MAX_CSMA_RETRIES
 *                      and SEED1 variables:
 *                      -# MIN_BE: Bit[7:6] Minimum back-off exponent in the
 *                         CSMA/CA algorithm.
 *                      -# MAX_CSMA_RETRIES: Bit[5:3] Number of retries in
 *                        TX_ARET_ON mode to repeat the CSMA/CA procedures
 *                        before the ARET procedure gives up.
 *                      -# SEED1: Bits[2:0] Higher 3 bits of CSMA_SEED, bits[10:8]
 *                         Seed for the random number generator in the
 *                         CSMA/CA algorithm.
 *  \retval TAT_SUCCESS The CSMA algorithm was configured successfully.
 *  \retval TAT_WRONG_STATE This function should not be called in the
 *                        SLEEP state.
 *
 *  \ingroup tat
 */
tat_status_t tat_configure_csma( uint8_t seed0, uint8_t be_csma_seed1 ){

    /*Check state.*/
    if (is_sleeping( ) == true) { return TAT_WRONG_STATE; }

    /*Extract parameters, and configure the CSMA-CA algorithm.*/
    uint8_t back_off_exponent = ( be_csma_seed1 & 0xC0 ) >> 6;
    uint8_t csma_retries      = ( be_csma_seed1 & 0x38 ) >> 3;
    uint8_t seed1           = ( be_csma_seed1 & 0x07 );

    hal_subregister_write( SR_MAX_FRAME_RETRIES, 0 ); //AT86RF230 rev A errata.
    hal_subregister_write( SR_MAX_CSMA_RETRIES, csma_retries );
    hal_subregister_write( SR_MIN_BE, back_off_exponent );
    hal_register_write( RG_CSMA_SEED_0, seed0 );
    hal_subregister_write( SR_CSMA_SEED_1, seed1 );

    return TAT_SUCCESS;
}

/*! \brief  This function uses the .
 *
 *  \note This function can only be executed after tat_configure_csma has been
 *        called!
 *  \note This function can only send valid IEEE 802.15.4 Frames.
 *
 *  \param frame_length Length of frame to transmit.
 *  \param frame Pointer to the frame to transmit.
 *  \param retries Number of times to retry frame transmission (Zero means that
 *                 the frame will be sent once.).
 *  \retval TAT_SUCCESS if the frame was sent successfully within the defined
 *                      number of retries.
 *  \retval TAT_CHANNEL_ACCESS_FAILURE if the channel was found to be busy on
 *                      the last retry.
 *  \retval TAT_NO_ACK if an IEEE 802.15.4 acknowledge was not received in time.
 *  \retval TAT_INVALID_ARGUMENT if the frame_length is too long.
 *  \retval TAT_WRONG_STATE if the radio transceiver is not in TX_ARET_ON.
 *
 *  \ingroup tat
 */
__x tat_status_t tat_send_data_with_retry( uint8_t frame_length, uint8_t *frame,
                                   uint8_t retries ){

    tat_status_t task_status = TAT_CHANNEL_ACCESS_FAILURE;

    /*Do sanity check on function parameters and current state.*/
    if ((frame_length > RF230_MAX_TX_FRAME_LENGTH) ||
        (frame_length < TAT_MIN_IEEE_FRAME_LENGTH)) {
        return TAT_INVALID_ARGUMENT;
    }

    if (tat_get_trx_state( ) != TX_ARET_ON) { return TAT_WRONG_STATE; }

    hal_clear_trx_end_flag( );
```

```
/*Do initial frame transmission.*/
hal_set_slptr_high( );
hal_set_slptr_low( );
hal_frame_write( frame, frame_length ); //Then write data to the frame buffer.

bool retry = false; // Variable used to control the retry loop.

/*Do retry if requested.*/
do{

   //Wait for TRX_END interrupt.
   while (hal_get_trx_end_flag( ) == 0) {;}

   //Check status.
   uint8_t transaction_status = hal_subregister_read( SR_TRAC_STATUS );

   //Check for failure.
   if ((transaction_status != TAT_TRANSMISSION_SUCCESS)) {

      if (transaction_status == TAT_BUSY_CHANNEL) {
         task_status = TAT_CHANNEL_ACCESS_FAILURE;
      } else {
         task_status = TAT_NO_ACK;
      }

      if ((retries--) > 0) {

         retry = true;

         //Wait for the TRX to go back to TX_ARET_ON.
         while (tat_get_trx_state() != TX_ARET_ON) {;}

         hal_clear_trx_end_flag( );
         hal_set_slptr_high( );
         hal_set_slptr_low( );
      } else {
         retry = false;
      }
   } else{

      task_status = TAT_SUCCESS;
      retry = false;
   } // end: if ((transaction_status != TAT_TRANSMISSION_SUCCESS)) ...
} while (retry == true);

return task_status;
```

```
}
/*EOF*/
```

Tat.h

```
/* This file has been prepared for Doxygen automatic documentation generation.*/
/*! \file
*********************************************************************
 *
 * \brief This files defines the API for the Transceiver Access Toolbox.
 *
 *      This file contains the function prototypes for the Transceiver Access
 *      Toolbox, hence it is an API. The Transceiver Access Toolbox is an
 *      abstraction layer that hides the details of the radio transceiver from
 *      the end-user. The goal for the Transceiver Access Toolbox is to wrap the
 *      services that the radio transceiver can perform into easy to use functions.
 *
 * \par Application note:
 *      AVR2001: AT86RF230 Software Programmer's Guide
 *
 * \par Documentation
 *      For comprehensive code documentation, supported compilers, compiler
 *      settings and supported devices see readme.html
 *
 * \author
 *      Atmel Corporation: http://www.atmel.com \n
 *      Support email: avr@atmel.com
 *
 * $Name$
 * $Revision: 613 $
 * $RCSfile$
 * $Date: 2006-04-07 14:40:07 +0200 (fr, 07 apr 2006) $  \n
 *
 * Copyright (c) 2006, Atmel Corporation All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions are met:
 *
 * 1. Redistributions of source code must retain the above copyright notice,
 * this list of conditions and the following disclaimer.
 *
 * 2. 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.
 *
 * 3. The name of ATMEL may not be used to endorse or promote products derived
```

```
*********************************************************************
******/
/*! \mainpage Transceiver Access Toolbox
 *
 * \section secIntroduction Introduction
 * The Transceiver Access Toolbox is implemented as an easy-to-use library that
 * covers most of the functionality provided by the AT86RF230 radio transceiver.
 * The library is completely written in the C programming language.
 *
 * \subsection secMisra MISRA Compliance
 * The Transceiver Access Toolbox is MISRA Compliant to the required rule set,
 * with the following exceptions:
 *   -# Rule 1: No compiler extensions allowed. Comment: Not possible to meet,
 *           since we are on an embedded target where interrupts are needed.
 *   -# Rule 37: Bitwise operations is not allowed on signed integer types.
 *            Comment: This is an error in the MISRA parser, since no singned
 *            variable is used. Problem seams to be in stdint.h.
 *   -# Rule 54: Null statement can only be on a line by itself. Comment: Triggered
 *            by the AVR_ENTER_CRITICAL_REGION macro. This macro can be
 *            written out so that the error goes away. The macro is only
 *            used for convenience.
 *   -# Rule 71: No prototype seen. Comment: This error orginates from how
 *            interrupt handlers are implemented in IAR.
 *   -# Rule 96: Function like macro shall be enclosed in parenthesis. Comment:
 *             Triggered by the AVR_ENTER_CRITICAL_REGION macro.
 *             This macro can be written out so that the error goes away.
 *             The macro is only used for convenience and readability of the
 *             code.
 *
 * \subsection secCompileOptions Compile Options
 *
 *   -# F_CPU: The clock speed of the CPU must be defined. The current
 *           implementation supports 1, 4, 8 and 16 MHz operation. Example:
 *           # define F_CPU=8000000UL
 *   -# ARCHITECTURE: AVR is the only currently supported architecture.
 *                Example: # define AVR
 *
 *   See the examples directory for further information on how to set up the
 *   IAR or avr-gcc toolchains to define these symbols.
 *
 * \defgroup tat Transceiver Access Toolbox API
 *     The Transceiver Access Toolbox API is a set of functions that provides the
 *     end user with total access to all functionality in the radio transceiver,
 *     without dealing with its registers and internal dynamics.
 *
 */

#ifndef TAT_H
#define TAT_H
/*============================ INCLUDE
=========================================*/
#include <stdint.h>
#include <stdbool.h>

#include "compiler.h"
/*============================ MACROS
==========================================*/
#define SUPPORTED_PART_NUMBER                 ( 2 )
#define RF230_REVA                            ( 1 )
#define RF230_REVB                            ( 2 )
#define SUPPORTED_MANUFACTURER_ID             ( 31 )
#define RF230_SUPPORTED_INTERRUPT_MASK        ( 0xCF )

#define RF230_MIN_CHANNEL                     ( 11 )
#define RF230_MAX_CHANNEL                     ( 26 )
#define RF230_MIN_ED_THRESHOLD                ( 0 )
#define RF230_MAX_ED_THRESHOLD                ( 15 )
#define RF230_MAX_TX_FRAME_LENGTH             ( 127 ) //!< 127 Byte PSDU.
```

```
#define TX_PWR_3DBM                     ( 0 )
#define TX_PWR_17_2DBM                  ( 15 )

#define BATTERY_MONITOR_HIGHEST_VOLTAGE         ( 15 )
#define BATTERY_MONITOR_VOLTAGE_UNDER_THRESHOLD ( 0 )
#define BATTERY_MONITOR_HIGH_VOLTAGE            ( 1 )
#define BATTERY_MONITOR_LOW_VOLTAGE             ( 0 )

#define FTN_CALIBRATION_DONE                ( 0 )
#define PLL_DCU_CALIBRATION_DONE            ( 0 )
#define PLL_CF_CALIBRATION_DONE             ( 0 )
/*============================ TYPEDEFS
=========================================*/

/*! \brief  This macro defines the start value for the TAT_* status constants.
 *
 *         It was chosen to have this macro so that the user can define where
 *         the status returned from the TAT starts. This can be useful in a
 *         system where numerous drivers are used, and some range of status codes
 *         are occupied.
 *
 *  \see tat_status_t
 *  \ingroup tat
 */
#define TAT_STATUS_START_VALUE              ( 0x40 )

/*! \brief  This enumeration defines the possible return values for the TAT API
 *         functions.
 *
 *         These values are defined so that they should not collide with the
 *         return/status codes defined in the IEEE 802.15.4 standard.
 *
 *  \ingroup tat
 */
typedef enum{
   //!< The requested service was performed successfully.
   TAT_SUCCESS = TAT_STATUS_START_VALUE,
   //!< The connected device is not an Atmel AT86RF230.
   TAT_UNSUPPORTED_DEVICE,
   //!< One or more of the supplied function arguments are invalid.
   TAT_INVALID_ARGUMENT,
   //!< The requested service timed out.
   TAT_TIMED_OUT,
   //!< The end-user tried to do an invalid state transition.
   TAT_WRONG_STATE,
   //!< The radio transceiver is busy receiving or transmitting.
```

```
    TAT_BUSY_STATE,
    //!< The requested state transition could not be completed.
    TAT_STATE_TRANSITION_FAILED,
    //!< Channel in idle. Ready to transmit a new frame.
    TAT_CCA_IDLE,
    //!< Channel busy.
    TAT_CCA_BUSY,
    //!< Transceiver is busy receiving or transmitting data.
    TAT_TRX_BUSY,
    //!< Measured battery voltage is lower than voltage threshold.
    TAT_BAT_LOW,
    //!< Measured battery voltage is above the voltage threshold.
    TAT_BAT_OK,
    //!< The CRC failed for the actual frame.
    TAT_CRC_FAILED,
    //!< The channel access failed during the auto mode.
    TAT_CHANNEL_ACCESS_FAILURE,
    //!< No acknowledge frame was received.
    TAT_NO_ACK,
}tat_status_t;

/*! \brief  This enumeration defines the possible modes available for the
 *          Clear Channel Assessment algorithm.
 *
 *          These constants are extracted from the datasheet.
 *
 *  \ingroup tat
 */
typedef enum{
    //!< Use energy detection above threshold mode.
    CCA_ED                  = 0,
    //!< Use carrier sense mode.
    CCA_CARRIER_SENSE       = 1,
    //!< Use a combination of both energy detection and carrier sense.
    CCA_CARRIER_SENSE_WITH_ED = 2
}tat_cca_mode_t;

/*! \brief  This enumeration defines the possible CLKM speeds.
 *
 *          These constants are extracted from the datasheet.
 *
 *  \ingroup tat
 */
typedef enum{

    CLKM_DISABLED      = 0,
   CLKM_1MHZ        = 1,
   CLKM_2MHZ        = 2,
   CLKM_4MHZ        = 3,
   CLKM_8MHZ        = 4,
   CLKM_16MHZ        = 5
}tat_clkm_speed_t;
/*============================ PROTOTYPES
====================================*/
tat_status_t tat_init( void );
uint8_t tat_get_operating_channel( void );
tat_status_t tat_set_operating_channel( uint8_t channel );
uint8_t tat_get_tx_power_level( void );
tat_status_t tat_set_tx_power_level( uint8_t power_level );

tat_status_t tat_do_ed_scan( uint8_t *ed_level );
uint8_t tat_get_cca_mode( void );
uint8_t tat_get_ed_threshold( void );
tat_status_t tat_set_cca_mode( uint8_t mode, uint8_t ed_threshold );
tat_status_t tat_do_cca( void );
tat_status_t  tat_get_rssi_value( uint8_t *rssi );

uint8_t tat_batmon_get_voltage_threshold( void );
uint8_t tat_batmon_get_voltage_range( void );
tat_status_t tat_batmon_configure( bool range, uint8_t voltage_threshold );
tat_status_t tat_batmon_get_status( void );

uint8_t tat_get_clock_speed( void );
tat_status_t tat_set_clock_speed( bool direct, uint8_t clock_speed );
tat_status_t tat_calibrate_filter( void );
tat_status_t tat_calibrate_pll( void );

uint8_t tat_get_trx_state( void );
tat_status_t tat_set_trx_state( uint8_t new_state );
tat_status_t tat_enter_sleep_mode( void );
tat_status_t tat_leave_sleep_mode( void );
void tat_reset_state_machine( void );
void tat_reset_trx( void );

void tat_use_auto_tx_crc( bool auto_crc_on );
__x tat_status_t tat_send_data( uint8_t data_length, uint8_t *data );

uint8_t tat_get_device_role( void );
void tat_set_device_role( bool i_am_coordinator );
uint16_t tat_get_pan_id( void );
void tat_set_pan_id( uint16_t new_pan_id );
uint16_t tat_get_short_address( void );
```

```
void tat_set_short_address( uint16_t new_short_address );
__x void tat_get_extended_address( uint8_t *extended_address );
__x void tat_set_extended_address( uint8_t *extended_address );
tat_status_t tat_configure_csma( uint8_t seed0, uint8_t be_csma_seed1 );
__x tat_status_t tat_send_data_with_retry( uint8_t frame_length, uint8_t *frame,
                                  uint8_t retries );
#endif
/*EOF*/

/* This file has been prepared for Doxygen automatic documentation generation.*/
/*! \file
*********************************************************************
 *
 * \brief This files implements a very simple serial communication link between
 *        two nodes.
 *
 * \section wireless_uart The Simple Wireless Uart Example
 *      This particular example implements a very simple and crude wireless uart.
 *      It does not use any of the mechanisms described by the IEEE 802.15.4
 *      standard. It will simply send the last message received on the serial
 *      interface, and not check for a busy channel or wait for an
 *      acknowledgement. This is just to show how simple it can be to
 *      communicate with the AT86RF230 radio transceiver. Data received on the
 *      air interface will be pushed onto the serial stream (USB or RS232).
 *
 * \par Application note:
 *      AVR2001: Transceiver Access Toolbox for the AT86RF230
 *
 * \par Documentation
 *      For comprehensive code documentation, supported compilers, compiler
 *      settings and supported devices see readme.html
 *
 * \author
 *      Atmel Corporation: http://www.atmel.com \n
 *      Support email: avr@atmel.com
 *
 * $Name$
 * $Revision: 613 $
 * $RCSfile$
 * $Date: 2006-04-07 14:40:07 +0200 (fr, 07 apr 2006) $  \n
 *
 * Copyright (c) 2006, Atmel Corporation All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions are met:
 *
 * 1. Redistributions of source code must retain the above copyright notice,
 * this list of conditions and the following disclaimer.
 *
 * 2. 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.
 *
 * 3. The name of ATMEL may not be used to endorse or promote products derived
 * from this software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY ATMEL ``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
EXPRESSLY AND
 * SPECIFICALLY DISCLAIMED. IN NO EVENT SHALL ATMEL 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.

*******************************************************************
******/
/*============================ INCLDUE
=========================================*/
#include <stdint.h>
#include <stdbool.h>

#include "config_uart.h" // See this file for all project options.

#include "compiler.h"

#include "at86rf230_registermap.h"
#include "hal.h"
#include "tat.h"
#include "com.h"
```

```
/*============================= MACROS
=========================================*/
/*============================= TYPEDEFS
=======================================*/
/*============================= VARIABLES
======================================*/
static hal_rx_frame_t rx_pool[ RX_POOL_SIZE ]; //!< Pool of hal_rx_frame_t's.
static hal_rx_frame_t *rx_pool_start; //!< Pointer to start of pool.
static hal_rx_frame_t *rx_pool_end; //!< Pointer to end of pool.
static hal_rx_frame_t *rx_pool_head; //!< Pointer to next hal_rx_frame_t it is possible to
write.
static hal_rx_frame_t *rx_pool_tail; //!< Pointer to next hal_rx_frame_t that can be read
from the pool.
static uint8_t rx_pool_items_free; //!< Number of free items (hal_rx_frame_t) in the pool.
static uint8_t rx_pool_items_used; // !< Number of used items.
static bool rx_pool_overflow_flag; //!< Flag that is used to signal a pool overflow.

static bool rx_flag; //!< Flag used to mask between the two possible TRX_END events.

static uint8_t debug_pll_transition[] = "State transition failed\r\n"; //!< Debug Text.
static uint8_t debug_type_message[] = "\r<---Type Message:\r\n"; //!< Debug Text.
static uint8_t debug_data_sent[] = "<---TX OK.\r\n"; //!< Debug Text.
static uint8_t debug_data_received[] = "\r--->Rx:\r"; //!< Debug Text.
static uint8_t debug_lqi[] = "LQI: "; //!< Debug Text.
static uint8_t debug_rx_pool_overflow[] = "RX Buffer Overflow!\r\n"; //!< Debug Text.
static uint8_t debug_transmission_failed[] = "TX Failed!\r\n"; //!< Debug Text.
static uint8_t debug_transmission_length[] = "Typed Message too long!!\r\n"; //!< Debug
Text.
static uint8_t debug_fatal_error[] = "A fatal error. System must be reset.\r\n"; //!< Debug
Text.
/*============================= PROTOTYPES
======================================*/
static bool trx_init( void );
static void avr_init( void );
static void trx_end_handler( uint32_t time_stamp );
static void rx_pool_init( void );

/*! \brief This function is used to initialize the TRX.
 *
 *  \retval true if the TRX was successfully configured.
 *  \retval false if the TRX was not configured properly.
 */
static bool trx_init( void ){

    static bool status;
    if (tat_init( ) != TAT_SUCCESS) {
        status = false;
    } else if (tat_set_operating_channel( OPERATING_CHANNEL ) != TAT_SUCCESS)
{
        status = false;
    } else if (tat_set_clock_speed( true, CLKM_DISABLED ) != TAT_SUCCESS) {
        status = false;
    } else{

        tat_use_auto_tx_crc( true ); //Automatic CRC must be enabled.
        hal_set_trx_end_event_handler( trx_end_handler ); // Event handler for TRX_END
events.

        status = true;
    } // end: if (tat_init( ) != TAT_SUCCESS) ...

    return status;
}

/*! \brief This function configure the necessary IO modules on the AVR.
 */
static void avr_init( void ){
    com_init( BR_38400 );
}

/*! \brief This function initialize the rx_pool. The rx_pool is in essence a FIFO.
 */
static void rx_pool_init( void ){

    rx_pool_start = rx_pool;
    rx_pool_end = &rx_pool[ RX_POOL_SIZE - 1 ];

    rx_pool_head = rx_pool_start;
    rx_pool_tail = rx_pool_end;

    rx_pool_items_free = RX_POOL_SIZE;
    rx_pool_items_used = 0;

    rx_pool_overflow_flag = false;
}

/*! \brief This function is the TRX_END event handler that is called from the
 *        TRX isr if assigned.
 *
 *  \param[in] time_stamp Interrupt timestamp in IEEE 802.15.4 symbols.
 */
```

```
static void trx_end_handler( uint32_t time_stamp ){

    if (rx_flag == true) {

        //Check if these is space left in the rx_pool.
        if (rx_pool_items_free == 0) {
            rx_pool_overflow_flag = true;
        } else {

            //Space left, so upload the received frame.
            hal_frame_read( rx_pool_head );

            //Then check the CRC. Will not store frames with invalid CRC.
            if (rx_pool_head->crc == true) {

                //Handle wrapping of rx_pool.
                if (rx_pool_head == rx_pool_end) {
                    rx_pool_head = rx_pool_start;
                } else {
                    ++rx_pool_head;
                } // end: if (rx_pool_head == rx_pool_end) ...

                --rx_pool_items_free;
                ++rx_pool_items_used;
            } // end: if (rx_pool_head->crc == true) ...
        } // end: if (rx_pool_items_free == 0) ...
    } // end:  if (rx_flag == true) ...
}

void main( void ){

    static uint8_t length_of_received_data = 0;
    rx_flag = true;

    rx_pool_init( );
    avr_init( );
    trx_init( );

    //Set system state to RX_ON
    if (tat_set_trx_state( RX_ON ) != TAT_SUCCESS) {
        com_send_string( debug_fatal_error, sizeof( debug_fatal_error ) );
    } // end: if (tat_set_trx_state( RX_ON ) != TAT_SUCCESS) ...

    sei( );

    //Give the user an indication that the system is ready.
```

```
com_send_string( debug_type_message, sizeof( debug_type_message ) );

/*Enter Normal Program Flow:
   - Check for newly received frames. Print them if something is received.
   - Notify on rx_pool overflow.
   - Try to send data on air interface, if something is received on UART/USB.
   - Notify if the typed message was too long.
 */
while (true) {

   //Check if we have received something on the air interface.
   if (rx_pool_items_used != 0) {

      //Handle wrapping of rx_pool.
      if (rx_pool_tail == rx_pool_end) {
         rx_pool_tail = rx_pool_start;
      } else {
         ++rx_pool_tail;
      } // end: if (rx_pool_tail == rx_pool_end) ...

      //Turn interrupts off for a short while to protect when status
      //information about the rx_pool is updated.
      cli( );

      ++rx_pool_items_free;
      --rx_pool_items_used;

      sei( );

      //Send the frame to the user:
      com_send_string( debug_data_received, sizeof( debug_data_received ) );
      com_send_string( rx_pool_tail->data, ((rx_pool_tail->length) - 2 ) );
      com_send_string( debug_lqi, sizeof( debug_lqi ) );
      com_send_hex( rx_pool_tail->lqi );
      com_send_string( debug_type_message, sizeof( debug_type_message ) );
   } // end: if (rx_pool_items_used != 0) ...

   //Check for rx_pool overflow.
   if (rx_pool_overflow_flag == true) {

      cli();
      rx_pool_init( );
      com_send_string( debug_rx_pool_overflow, sizeof( debug_rx_pool_overflow ) );
      sei();
   } // end: if (rx_pool_overflow_flag == true) ...
```

```
    //Check for new data on the serial interface.
    //Check if data is ready to be sent.
    length_of_received_data = com_get_number_of_received_bytes( );

    if (length_of_received_data == 1) {

       //length_of_received_data == 1 indicates a buffer overflow. Received data too
long.
       com_send_string( debug_transmission_length, sizeof( debug_transmission_length
) );
       com_reset_receiver( );
    } else {

       if ((length_of_received_data >= 3) && (length_of_received_data <=
COM_RX_BUFFER_SIZE)) {

          //Change state to PLL_ON and send data if the state transition was successful.
          if (tat_set_trx_state( PLL_ON ) == TAT_SUCCESS) {

             uint8_t *rx_frame = com_get_received_data( );

             rx_flag = false; // Set the flag false, so that the TRX_END event is not
misinterpreted.

             if (tat_send_data( length_of_received_data, rx_frame ) == TAT_SUCCESS)
{
                com_send_string( debug_data_sent, sizeof( debug_data_sent ) );
             } else {
                com_send_string( debug_transmission_failed, sizeof(
debug_transmission_failed ) );
             } // end:  if (tat_send_data_with_retry( tx_frame_length, tx_frame, 1 ) ...

             //Wait for the TRX FSM to go back to PLL_ON.
             while (tat_get_trx_state( ) != PLL_ON) {;}
          } else {
             com_send_string( debug_pll_transition, sizeof( debug_pll_transition ) );
          } // end: if (tat_set_trx_state( PLL_ON ) == TAT_SUCCESS) ...

          if (tat_set_trx_state( RX_ON ) != TAT_SUCCESS) {
             com_send_string( debug_fatal_error, sizeof( debug_fatal_error ) );
          } // end: if (tat_set_trx_state( RX_ON ) != TAT_SUCCESS) ...

          rx_flag = true; // Set the flag back again. Only used to protec the frame
transmission.
          com_reset_receiver( );
          com_send_string( debug_type_message, sizeof( debug_type_message ) );
```

```
            } // end:
        } // end: if (length_of_received_data == 1) ...
    } // emd: while (true) ...
}
/*EOF*/
```