```
/*
 * blt_ll_st17h26_loop.h
 *
 *  Created on: 2015-10-22
 *      Author: Lenze
 */

#ifndef BLT_LL_st17h26_LOOP_H_
#define BLT_LL_st17h26_LOOP_H_

/
********************************************************************
*
 * Public Functions
 */

/
********************************************************************
*
 * @fn      blt_push_notify
 *
 * @brief   API for notify small data to master.
 *
 * @param   handle    [in]       - The handle which used to notify
data.
 *          val           [in]       - The data value.
 *          len           [in]           - The size of data. The max
size is 4.
 *
 * @return      Status.
 * @retval      1 : notification successfully.
 * @retval  0 : notification fail.
 */
u8 blt_push_notify (u16 handle, u32 val, int len);

/
********************************************************************
*
 * @fn      blt_push_notify_data
 *
 * @brief   API for notify large data to master.
 *
 * @param   handle    [in]       - The handle which used to notify
data.
 *          p             [in]       - The start pointer of data.
 *          len           [in]           - The size of data. The max
size is 20.If over 20 bytes,
 *                                               then notify the
first 20 bytes,the extra data will be ignord.
 *
 * @return      Status.
 * @retval      1 : notification successfully.
 * @retval  0 : notification fail.
 */
```

```
u8 blt_push_notify_data (u16 handle, u8 *p, int len);

/
*******************************************************************
*
 * @fn      blt_adv_init
 *
 * @brief   API for initializing advertising data.Calibration
advertising time.
 *
 * @param   None.
 *
 * @return      None.
 */
void    blt_adv_init ();

/
*******************************************************************
*
 * @fn      blt_init
 *
 * @brief   API for initializing data of advertising stage.Include
advertising mac address,advertising data and scan response data.
 *
 * @param   p_mac       [in]    - The pointer of mac address buffer.
Currently, the size of the buffer is 6 bytes.
 *                      p_adv   [in]    - The pointer of advertising
data buffer.This buffer must be based on format of st17h26 adv data.
 *                      p_rsp   [in]    - The pointer of scan
response data buffer. This buffer must be based on format of st17h26
scan response.
 *
 * @return      None.
 */
void blt_init (u8 *p_mac, u8 *p_adv, u8 *p_rsp);

/
*******************************************************************
*
 * @fn      blt_init_timing_hid
 *
 * @brief   API for timing calibration when interval lower than
100ms, could be called in user_init.
 *
 * @param   None
 *
 * @return      None.
 */
void blt_init_timing_hid ();

/
*******************************************************************
*
 * @fn      blt_init_timing
 *
 * @brief   API for timing calibration when interval longer than
100ms, could be called in user_init.
 *
 * @param   None
 *
 * @return      None.
 */
void blt_init_timing();

/
*******************************************************************
*
 * @fn      blt_set_adv_interval
 *
 * @brief   API for setting advertising interval.
 *
 * @param   t_us        [in]            - The advertising interval
in microseconds.
 *
 * @return      None.
 */
void    blt_set_adv_interval (u32 t_us);

/
*******************************************************************
*
 * @fn      blt_set_adv_type
 *
 * @brief   API for setting advertising type.
 *
 * @param   type        [in] - The advertising type.
 *
 * @return      None.
 *
 * @note        The type value of parameter must one of the
following four options.
 *                      -0 connected undirected advertising event
 *                      -1 connected directed advertising event
 *                      -2 non-connected undirected advertising
 *                      -6 connected discovery advertising event
 */
void    blt_set_adv_type (u8 type);

/
*******************************************************************
*
 * @fn      blt_send_adv
 *
 * @brief   API for sending advertising packet.
 *
 * @param   mask [in]   - The advertising channel mask.
 *
 * @return      None.
 */
u8*             blt_send_adv (int mask);

/
********************************************************************
*
 * @fn      blt_fifo_num
 *
 * @brief   API for getting the current number of already used
fifo.The total fifo number is 4.
 *
 * @param   None.
 *
 * @return      The current number of already used fifo.
 */
u8 blt_fifo_num ();

/
********************************************************************
*
 * @fn      blt_fifo_empty
 *
 * @brief   API for getting the fifo status,empty or not empty.
 *
 * @param   None.
 *
 * @return      status *
 * @retval      1 : The current fifo state is Empty
 * @retval  0 : Not empty.
 */
int blt_fifo_empty ();

/
********************************************************************
*
 * @fn      blt_terminate
 *
 * @brief   API for sending the terminate command to end current
connection.
 *
 * @param   None.
 *
 * @return      None
 */
void blt_terminate ();

/
********************************************************************
*
 * @fn      blt_brx
 *
 * @brief   API for handling the status in ble connection status.
 * @note        Device must call this function if it need a normal
connection .
```

```
 *
 * @param   None.
 *
 * @return      Status
 */
u8      blt_brx ();

/
********************************************************************
*
 * @fn      blt_brx_sleep
 *
 * @brief   API for handling the status in ble connection status and
setting the connect parameters.
 *
 * @note        Device must call this function if it need a normal
connection .
 *                      In order to keep a relative accurate
timing, we should call
 *                      blt_brx() or blt_send_adv() without other
functions after called this.
 * @param   [in]        app_wakeup_tick         - Wakeup tick of
application layer required,or it should be 0 while application
didn't need special time.
 *
 * @return      None
 */
void blt_brx_sleep (u32 app_wakeup_tick);

/
********************************************************************
*
 * @fn      blt_enable_suspend
 *
 * @brief   API for setting the status of idle time after setting
it.
 *
 * @param   [in]        en              - The idle time status.
 * @note        Generally, the idle time should be suspend state or
deepsleep state for low energy. And we would not to set it while we
are debug the code.
 *                      The parameter [en] can be a combination of
any one of following advertising states and any one of following
connect states:
 *                      Advertising state:      0
        :system will not enter suspend or deepsleep mode.
 *
SUSPEND_ADV     :system will enter suspend mode .
 *
DEEPSLEEP_CONN :system will enter deepsleep mode .
 *
 *                      Connection state:       0
        :system will not enter suspend or deepsleep mode.
 *
```

```
SUSPEND_CONN:system will enter suspend mode.
 * @return   The idle time state before setting.
 *
 */
u8      blt_enable_suspend (u8 en);

/
********************************************************************
*
 * @fn      blt_set_wakeup_source
 *
 * @brief   API for setting the wakeup source while system enter
suspend mode or deepsleep mode.
 *
 * @param   [in]        src             - wakeup src while system
enter suspend/deepsleep mode.(System has setted a default wakeup
source, wakeup from timer)
 * @note        The parameter is select one or more from follows:
 *                      PM_WAKEUP_CORE : suspend wakeup from gpio.
 *                      PM_WAKEUP_PAD  : deepsleep wakeup from gpio.
 *                      PM_WAKEUP_TIMER : suspend/deepsleep wakeup
from timer.
 *
 * @return   None
 */
void    blt_set_wakeup_source (int src);

/
********************************************************************
*
 * @fn      blt_update_parameter_request
 *
 * @brief   API for sending the command LL_CONNECTION_PARAM_REQ, and
get command parameter from the function blt_update_conn_para used
recently, if system never called blt_update_conn_para, and it will
used default parameter.
 *
 * @param       None.
 *
 * @return  Status.
 * @retval  1 - Send successfully.
 * @retval  0 - send fail.
 */
u8 blt_update_parameter_request ();

/
********************************************************************
*
 * @fn      blt_update_conn_para
 *
 * @brief   API for setting the parameter of connection without
sending it;
 *
 * @param   [in]        min_interval            - The Interval_Min
```

```
shall be set to indicate the minimum value of connInteravl.
connInterval = Interval_Min * 1.25 Ms. The default Interval_Min
value is 104.
 *                      [in]    max_interval            - The
Interval_Max shall be set to indicate the maximum value of
connInteravl. connInterval = Interval_Min * 1.25 Ms. The default
Interval_Max is value 120.
 *                      [in]    latency                         -
The Latency shall be set to indicate the connSlaveLatency.
connSlaveLatency = Latency. The default Latency value is 0.
 *                      [in]    timeout                         -
The Timeout shall be set to indicate the connSupervisionTimeout
Value.connSupervisionTimeout = timeout * 10 Ms. The default Timeout
value is 600.
 * @note        Refer to command LL_CONNECTION_PARAM_REQ in <Core
4.1> VOL6.PartB.2.4.2.16:
 *
 * @return   None
 */
void blt_update_conn_para (u16 min_interval, u16 max_interval, u16
latency, u16 timeout);

/
********************************************************************
*
 * @fn      blt_register_event_callback
 *
 * @brief   API for register callback function, and used to
application layer/
 *
 * @param   [in]        e       - The event flag.
 *                      [in]    p       - The callback function
pointer.
 *
 * @return   None
 */
void blt_register_event_callback (u8 e, blt_event_callback_t p);
#endif /* BLT_LL_st17h26_LOOP_H_ */
```