```
/*
**  hypos-tuto-227.c Sample code for HyperpanelOS ========================== **
**                                                                           **
**  This simple code is located into the Application Container, it is run    **
**  by the VMK sub-operating system. On the other hand, the I/O container    **
**  runs all the drivers that are VMIO Finite State machines.                **
**                                                                           **
**  This code demonstrate how to use the Remote Control Unit (RCU) driver    **
**  in receive mode.                                                         **
**                                                                           **
**  The receiver module is the 433Mhz RF MX-05V wireless module.             **
**  The RCU is a DIO 54740                                                   **
**                                                                           **
**  The software behaves as follows:                                         **
**                                                                           **
**  - Keys 1 and 0 of line "1" will respectively switch ON and OFF the       **
**    RED LED of the daughter board                                          **
**                                                                           **
**  - Keys 1 and 0 of line "2" will respectively switch ON and OFF the       **
**    GREEN LED of the daughter board                                        **
**                                                                           **
**  - Keys 1 and 0 of line "3" will respectively switch ON and OFF the       **
**    YELLOW LED of the daughter board                                       **
**                                                                           **
**  - Keys 1 and 0 of line "G" will respectively switch ON and OFF all       **
**    the 3 RED/GREEN/YELLOW les of the daugther board                       **
**                                                                           **
**  ========================================================================= **
*/


/* Documentations of devices ------------------------------------------------

   RF 433MHz wireless - Where to buy:
   https://makeradvisor.com/tools/433mhz-receiver-transmitter-module/

   RF 433MHz receiver - Specifications
   Frequency Range: 433.92 MHz
   Modulation: ASK
   Input Voltage: 5V


*/

/* Includes files and external references .................................*/

#include <hypos.h>                    // Hyperpanel OS basic interfaces
#include <drv_asy.h>                  // Prototype of "asy_write"
#include <drv_rcu.h>                  // Prototype of RCU procedures

#include "./myio.h"                   // Interface of "myio.c"
#include "./mytsk.h"                  // Interface of "mytsk.c"


/* Alternate names for the DIO 54760 UHF RCU keys .........................*/

#define KEY_1_OFF        KEY_SHIFT_F1  // Device 1: OFF
#define KEY_1_ON         KEY_SHIFT_F2  // Device 1: ON
#define KEY_2_OFF        KEY_SHIFT_F3  // Device 2: OFF
#define KEY_2_ON         KEY_SHIFT_F4  // Device 2: ON
#define KEY_3_OFF        KEY_SHIFT_F5  // Device 3: OFF
#define KEY_3_ON         KEY_SHIFT_F6  // Device 3: ON
#define KEY_GRP_OFF      KEY_SHIFT_F11 // Group of devices: OFF
#define KEY_GRP_ON       KEY_SHIFT_F12 // Group of devices: ON


/* Internal data types of the module ......................................*/
```

```
    static void     open_gpio (               );
    static void     open_led  (               );
    static void     open_kbd  (               );
    static void     set_led   (int, const char*);


/* Internal global variables of this module --------------------------------*/

    int             gpio_iod        ; // IOD of device GPIO\DEV0
    int             led_iod         ; // IOD of device LED\DEV0
    int             led_iods[3]     ; // IOD of subchan RED/GREEN/YELLOW
    int             kbd_iod         ; // IOD of device KBDITF\DEV0
    int             rcurcv_id       ; // ID for RCU RCV
    char            mess[32]        ; // Message for trace on ASY0


/* Static data of this module ----------------------------------------------*/

    static const char *const led_tag_pop  = "CMD=POPALL"                    ;
    static const char *const led_tag_off =  "PATTERN=OFF:0\nCMD=PUSH"       ;
    static const char *const led_tag_on =   "PATTERN=ON:0\nCMD=PUSH"        ;


/* Message -----------------------------------------------------------------*/

    static const char *text[]       = // Messages on scree
      {
        "DIO Key 1 OFF     \r\n"    , // Receive Key 1 OFF from the DIO RCU.
        "DIO Key 1 ON      \r\n"    , // Receive Key 1 ON  from the DIO RCU.
        "DIO Key 2 OFF     \r\n"    , // Receive Key 2 OFF from the DIO RCU.
        "DIO Key 2 ON      \r\n"    , // Receive Key 2 ON  from the DIO RCU.
        "DIO Key 3 OFF     \r\n"    , // Receive Key 3 OFF from the DIO RCU.
        "DIO Key 3 ON      \r\n"    , // Receive Key 3 ON  from the DIO RCU.
        "DIO Key Group OFF \r\n"    , // Receive Key G OFF from the DIO RCU.
        "DIO Key Group ON  \r\n"    , // Receive Key G ON  from the DIO RCU.
      }                             ;


/* Beginning of the code ---------------------------------------------------

loop_app_tsk        Main loop of the Application container task
open_gpio           Open the GPIO driver
open_led            Open the LED driver
open_kbd            Open the KBDITF driver
set_led             Set a pattern to one LED
*/


/*  Procedure loop_app_tsk ------------------------------------------------

    Purpose : This is our task main loop.
*/

int loop_app_tsk (void *param)
  {
    int             keycode         ; // Received RCU keycode
    char            msg[32]         ; // Trace message

/*****************************************************************************
 * Initialisations
 ****************************************************************************/

    asy_write(0,(unsigned char*)"DIO Init\r\n",10);

    open_gpio()                     ; // Open the GPIO driver
    open_led()                      ; // Open the LED driver
    open_kbd()                      ; // Open the KBDITF driver
```

```
/*****************************************************************************
    Main loop of the receiver function. The loop receives from "drv_rcu"
    DRV_KEY_PRESS and DRV_KEY_RELEASE events. With DRV_KEY_PRESS events, we
    change the local LED state according to the received keycode. The
    DRV_KEY_RELEASE events are not used.
 *****************************************************************************/

    loop                            : // Main loop.

    wait_allevent()                 ; // Wait for events.
    keycode = task_evt.reserve      ; // Get recevied event (keycode).

    if (task_evt.code==DRV_KEY_PRESS)
      {
        hsprintf(msg                ,
             "DIO Keycode=0x%04X\r\n",
                  keycode          );
        asy_write(0                 ,
                  (unsigned char*)msg,
                  strlen(msg))      ;

/*****************************************************************************
    Update the led according to a DIO keycode. The DIO Ref 54760 UHF RCU has 8
    buttons lebelled I1/O1, I2/O2, I3/O3 and IG/OG. According to the keycode,
    what we do with the 3 LEDs is as follows:

    I1 -> RED ON    O1 -> RED OFF
    I2 -> RED ON    O2 -> RED OFF
    I3 -> RED ON    O3 -> RED OFF
    IG -> ALL ON    OG -> ALL OFF
 *****************************************************************************/

        switch ( keycode )
          {
            case KEY_1_OFF   :          // Key 1 OFF
              asy_write(0,              // Send a trace
              (unsigned char*)text[0],20); //
              set_led(0,led_tag_off)    ; // Set RED OFF
              break                     ; //

            case KEY_1_ON    :          // Key 1 ON
              asy_write(0,              // Send a trace
              (unsigned char*)text[1],20); //
              set_led(0,led_tag_on )    ; // Set RED ON
              break                     ; //

            case KEY_2_OFF   :          // Key 2 OFF
              asy_write(0,              // Send a trace
              (unsigned char*)text[2],20); //
              set_led(1,led_tag_off)    ; // Set GREEN OFF
              break                     ; //

            case KEY_2_ON    :          // Key 2 ON
              asy_write(0,              // Send a trace
              (unsigned char*)text[3],20); //
              set_led(1,led_tag_on )    ; // Set GREEN ON
              break                     ; //

            case KEY_3_OFF   :          // Key 3 OFF
              asy_write(0,              // Send a trace
              (unsigned char*)text[4],20); //
              set_led(2,led_tag_off)    ; // Set YELLOW OFF
              break                     ; //

            case KEY_3_ON    :          // Key 3 ON
              asy_write(0,              // Send a trace
              (unsigned char*)text[5],20); //
```

```
              set_led(2,led_tag_on )     ; // Set YELLOW ON
              break                      ; //

            case KEY_GRP_OFF :             // Key GROUP OFF
              asy_write(0,                 // Send a trace
              (unsigned char*)text[6],20); //
              set_led(0,led_tag_off)     ; // Set RED OFF
              set_led(1,led_tag_off)     ; // Set GREEN OFF
              set_led(2,led_tag_off)     ; // Set YELLOW OFF
              break                      ;

            case KEY_GRP_ON  :             // Key GROUP ON
              asy_write(0,                 // Send a trace
              (unsigned char*)text[7],20); //
              set_led(0,led_tag_on )     ; // Set RED ON
              set_led(1,led_tag_on )     ; // Set GREEN ON
              set_led(2,led_tag_on )     ; // Set YELLOW ON
              break                      ;
          }
    }

    goto loop                            ; // Next loop

    return 0                             ;
  }

/*  Procedure open_gpio ----------------------------------------------------

    Purpose : Initialize the code of the GPIO driver module.
*/

static void open_gpio()
  {
    myio_open(DRVGPIO,DEV0,"",&gpio_iod); // Open GPIO\DEV0
 }

/*  Procedure open_led -----------------------------------------------------

    Purpose : Initializes the LED driver, opens the LED controller, allocates
              the three LED (RED, GREEN, YELLOW) subchannels.
*/

static void open_led()
  {

/*************************************************************************
 * Step 1 : We call the "myio_open" that calls "drv_init_driver", then       *
 * ------   "add_uroute", then "drv_alloc_device" and "drv_open_device".     *
 *          It has to be noted that the LED driver is a "software device",   *
 *          because there is no LED controller dedicated device. Most of the *
 *          time "hidden" GPIO (not visible by the application) are used.    *
 *************************************************************************/

    myio_open(DRVLED,DEV0,"",&led_iod)  ; // Open LED\DEV0


/*************************************************************************
 * Step 2 : The contoller is ready. We are going to allocate the LED         *
 * ------   subchannels, one for each LED. We call the "myio_alloc_sub"      *
 *          subroutine that calls the "drv_alloc_subchan" procedure. This    *
 *          last procedure sends a REQ_ALLOC_SUB event to the VMIO AUT_LED   *
 *          automaton. This automaton will then send us a RESP_ALLOC_SUB     *
 *          response event. Here we unschedule until this event is received. *
 *************************************************************************/

    myio_alloc_sub(led_iod,"LEDNAME=RED"    ,&led_iods[0]); // Allocates RED
    myio_alloc_sub(led_iod,"LEDNAME=GREEN"  ,&led_iods[1]); // Allocates GREEN
    myio_alloc_sub(led_iod,"LEDNAME=YELLOW" ,&led_iods[2]); // Allocates YELLOW
```

```
  }

/*  Procedure open_kbd ----------------------------------------------------

    Purpose : Initialize the code of the KBDITF driver and then open the
              device to get key press events from the remote control unit
              (RCU). The opened device descriptor (Input/Output Descriptor) is
              kept in "kbd_iod". Open the "KBDITF\DEV0" device. We call the
              "myio_open" simple API procedure. In fact the subroutine calls
              "drv_init_driver", "add_uroute", "drv_alloc_device", and then
              the "drvopen_device". The "KBDITF\DEV0" driver is a software
              driver that does not handles any hardware by itself. With this
              sample application, the underlying hardware is an infra-red
             receiver. This hardware is managed by the "drv_rcu" driver, that
             in our case sends events to the KBDITF module, that sends
             DRV_KEY_PRESS and DRV_KEY_RELEASE events to us.
*/

static void open_kbd()
  {
    int             ret              ; // Procedures returned code

    myio_open(DRVKBD,DEV0,""         , // Opens KBDITF\DEV0
                           &kbd_iod) ; //

    rcu_get_id(0                     , // 0:Receiver 1:Transmitter
               0                     , // numdev = the first one
               &rcurcv_id            , // Corresponding id
               &ret                 ); // Error code

    rcu_set(rcurcv_id                , // id
            1                        , // swpos = the selected input (IR/UHF)
            "DIO 54760"              , // name  =
            0xFFFFFFFF               , // idval =
            &ret                     ); // ret   = error code
  }

/*  Procedure set_led -----------------------------------------------------

    Purpose : Sets a parten for LED "n" (0:RED, 1:GREEN, 2:YELLOW)
*/

static void set_led(int n, const char *pat)
  {
    myio_setval(led_iods[n]          , // Clear
                          led_tag_pop);
    myio_setval(led_iods[n],pat      ); // Send pattern
  }
```