```
/*! \file mainEjemplo_1.c
    \author Carlos Augusto Centeno
    \version    1.00
    \brief  Pagina Principal del Proyecto
    \mainpage   En el siguiente ejemplo se evaluan las capacidades de
uCOS-II.
    \tableofcontents
    \section sec    Caracteristicas Generales
    \n Se ejemplifica el uso de:
    \n EVENTS, Semaforos, MailBox,
    \n Suspension y Reasuncion de Tareas.
    \n Se puede verificar la ocurrencia de Deadlock.

  \subsection subsection1 RTOS
     uCOS-II version 2.52
     \n Autor: Jean J. Labrosse

  \subsection subsection2 PORT PIC18
     Autor: Nathan Brown.

  \subsection subsection3 Proyecto de Test

     Carlos A. Centeno
     \n mail: ccenteno@gmail.com
     \n G.In.T.E.A - UTN FRC -
http://www.investigacion.frc.utn.edu.ar/gintea/


*/

#include    "includes.h"
#include    "prototipos.h"
#include    "timers.h"
#include    "io_define.h"



OS_STK  Start_TaskStk[50L];     /*!< Stack de la Tarea Task_START   */
OS_STK  Task1Stk[100L]; /*!< Stack de la Tarea Task1 MUTEX  */
OS_STK  Task2Stk[100L]; /*!< Stack de la Tarea Task2 MUTEX  */
OS_STK  TaskAdcStk[50L];        /*!< Stack de la Tarea Task ADC     */
OS_STK  TaskReleStk[100L];      /*!< Stack de la Tarea Task RELE    */


OS_EVENT    *ADC_Mbox_1;        /*!< ACD_Mbox 1 */
OS_EVENT    *ADC_Mbox_2;        /*!< ACD_Mbox 2 */
OS_EVENT    *ADC_Sem_1;     /*!< ACD_Mbox 2 */
OS_EVENT    *ADC_Sem_2;     /*!< ACD_Mbox 2 */
```

```
//-------------------------------------------------------------------
------------
void Task_START(void *pdata)
{
    char err;
    UserInit();

    // configure the input/output pins
    ADCON1 = 0b00001110;        // set the A/D register
    INTCON2 = 0b01111111;
    INTCON3 = 0b00000000;

    // enable interrupts
    OpenTimer0(TIMER_INT_ON & T0_16BIT & T0_SOURCE_INT & T0_PS_1_1);
    TMR0H = 0xD8;
    TMR0L = 0xA0;
    INTCONbits.GIEH = 1;

    // Start up other tasks

    OSTaskCreate(Task1,     (void *)0, &Task1Stk[0], TASK_1_PRIORITY);
    OSTaskCreate(Task2,     (void *)0, &Task2Stk[0], TASK_2_PRIORITY);
    OSTaskCreate(TaskADC,       (void *)0, &TaskAdcStk[0],
TASK_ADC_PRIORITY);
    OSTaskCreate(TaskRele,      (void *)0, &TaskReleStk[0],
TASK_RELE_PRIORITY);

    OSTaskSuspend(TASK_ADC_PRIORITY);
    // task loop
    for(;;)
    {
        salidaLed3 = 0;
        OSTimeDly(3);
        salidaLed3 = 1;
        OSTimeDly(4);
        OSSemPend(ADC_Sem_2, 10, err);
        if(err == OS_TIMEOUT)
        {
            salidaLed3 = 1;
        }else if(err == OS_NO_ERR)
        {
            OSTaskResume(TASK_ADC_PRIORITY);

        }
    }
}

//-------------------------------------------------------------------
------------
```

```
    void main (void)
    {
        INT8U   err;

        INTCONbits.GIEH = 0;
        OSInit();

    //  ADC_Mutex = OSMutexCreate(ADC_MUTEX_PRIORITY, &err);
        ADC_Mbox_1=OSMboxCreate(0);
        ADC_Mbox_2=OSMboxCreate(0);
        ADC_Sem_1 = OSSemCreate(1);
        ADC_Sem_2 = OSSemCreate(0);



        OSTaskCreate(Task_START, (void *)0, &Start_TaskStk[0], 0);
        OSStart();
    }

    //-------------------------------------------------------------------
    ------------
    void UserInit(void)
    {
        // Puertos IO

        TRISA = 0b00000000;
        TRISB = 0b00000000;
        TRISC = 0b10000000;     // Pines 6 Y 7 para puerto RS232; pin 4
    como SDI

        TRISD = 0b00000000;
        TRISE = 0b00000000;

        LATA   = 0b11111111;
        LATB  =  0b11111111;
        ADCON1 = 0b00001111;    // RA5 (SS) configurada como I/O DIGITAL
        CMCON  = 0b00000111;    // comparador apagado

        PORTA=0;
        PORTB = 0x00;
        PORTC=0;
        PORTD=0;
        PORTE=0;

        PIR1 = 0;   /* clear any pending interrupts */

        //resistencias de pull - up  DESCONECTADAS (RBPU)
        INTCON2 = 0b10000000;
```

```
    }//end UserInit


    //-------------------------------------------------------------------
    ------------
    void Task1(void *pdata)
    {
        INT8U   err;
        INT8U   uiValorADC;
    #if OS_CRITICAL_METHOD == 3     /* Allocate storage for CPU status
    register      */
        OS_CPU_SR  cpu_sr;
    #endif

        for(;;)
        {
            OSSemPend(ADC_Sem_1,0, &err);
            uiValorADC = convertirADC();
            OSMboxPost(ADC_Mbox_1, (void *)&uiValorADC);
            OSSemPost(ADC_Sem_1);

            OSTimeDly(1);
        }
    }

    //-------------------------------------------------------------------
    ------------
    void Task2(void *pdata)
    {
        INT8U   err;
        INT8U uiValorADC;
    #if OS_CRITICAL_METHOD == 3                       /* Allocate storage
    for CPU status register           */
        OS_CPU_SR  cpu_sr;
    #endif

        for(;;)
        {
            OSSemPend(ADC_Sem_1,0, &err);
            uiValorADC = convertirADC();
            OSMboxPost(ADC_Mbox_2, (void *)&uiValorADC);
            OSSemPost(ADC_Sem_1);

            OSTimeDly(1);
        }
    }


    //-------------------------------------------------------------------
```

```
------------
void TaskRele(void *pdata)
{
#if OS_CRITICAL_METHOD == 3                       /* Allocate storage
for CPU status register           */
    OS_CPU_SR  cpu_sr;
#endif

    INT8U  *err;
    INT8U   sumatoria;
    INT8U *rxmsg;
    INT8U inc_P;

    for(;;)
    {
        // Uso de MBOX
        rxmsg = OSMboxPend(ADC_Mbox_1, 0, err);   // Recibo la
potencia del ADC
        sumatoria = *rxmsg;
        rxmsg = OSMboxPend(ADC_Mbox_2, 0, err);   // Recibo la
potencia del ADC
        sumatoria += *rxmsg;
        sumatoria /= 2;


        if(sumatoria >= 250)
            salidaRELE = 1;
        else if(sumatoria <= 200)
            salidaRELE = 0;

        OSTimeDly(5);
    }
}

//-------------------------------------------------------------------
------------
void TaskADC(void *pdata)
{

#if OS_CRITICAL_METHOD == 3                       /* Allocate storage
for CPU status register           */
    OS_CPU_SR  cpu_sr;
#endif

    for(;;)
    {
        salidaLed1 = 0;
        OSTimeDly(3);
        salidaLed1 = 1;
        OSTimeDly(4);
```

```
    }
}



//-------------------------------------------------------------------------
------------
long int convertirADC(void)
{
    union{
        int Alto;
        char Bajo;
    }valorADC;

    static INT16U valorTest = 122;
    static INT8S valorIncremental = 13;


    ADCON0bits.GO = 1;
    while(ADCON0bits.GO);
    valorADC.Alto = ADRESH*256;
    valorADC.Bajo = ADRESL;

    valorADC.Alto = valorTest;

    // codigo para emular conversion
    valorTest += valorIncremental;
    if(valorTest >= 768)
        valorIncremental = -27;
    else if(valorTest <= 100)
        valorIncremental = +13;


    return valorADC.Alto;
}
```