```
#include  <includes.h>

/************************************************************
*                LOCAL GLOBAL VARIABLES
************************************************************/

static  OS_TCB        AppTaskStartTCB;
static  OS_TCB        AppTaskLED2TCB;
static  OS_TCB        AppTaskLED3TCB;
static  OS_TCB        AppTaskLED4TCB;

static  CPU_STK       AppTaskStartStk[APP_CFG_TASK_START_STK_SIZE];

static  CPU_STK       AppTaskLED2Stk[APP_CFG_TASK_LED2_STK_SIZE];
static  CPU_STK       AppTaskLED3Stk[APP_CFG_TASK_LED3_STK_SIZE];
static  CPU_STK       AppTaskLED4Stk[APP_CFG_TASK_LED4_STK_SIZE];

OS_TMR       AppLED3Tmr;
OS_SEM       AppLED3Sem;

OS_TMR       AppLED4Tmr;
OS_Q         AppLED4MsgQ;

CPU_INT08U  AppLEDNbr;



/************************************************************
*                     FUNCTION PROTOTYPES
************************************************************/

static  void  AppTaskStart (void  *p_arg);

static  void  AppTaskLED2 (void  *p_arg);

static  void  AppTaskLED3 (void  *p_arg);

static  void  AppTaskLED4  (void  *p_arg);

static  void  AppObjCreate (void);

static  void  AppTaskCreate (void);

/* Timer callback functions for LED applications */

void  AppLED3TmrCallBack (OS_TMR *p_tmr, void   *p_arg);

void  AppLED4TmrCallBack (OS_TMR *p_tmr, void   *p_arg);
```

```
/*********************************************************************
*                              main()
*********************************************************************/
int main (void){
   OS_ERR      err;
#if (CPU_CFG_NAME_ERR == DEF_ENABLED)
   CPU_ERR     cpu_err;
#endif

   CPU_Init();

   BSP_IntDisAll();   /* Disable all interrupts. */

   OSInit(&err);   /* Initialize "uC/OS-III, The Real-Time Kernel" */

   OSTaskCreate((OS_TCB *) &AppTaskStartTCB,  /* Create the start task */
             (CPU_CHAR    *) "App Task Start",
             (OS_TASK_PTR  ) AppTaskStart,
             (void        *)0,
             (OS_PRIO      )1,
             (CPU_STK     *)&AppTaskStartStk[0],
             (CPU_STK_SIZE )APP_CFG_TASK_START_STK_SIZE_LIMIT,
             (CPU_STK_SIZE )APP_CFG_TASK_START_STK_SIZE,
             (OS_MSG_QTY   )0,
             (OS_TICK      )0,
             (void        *)0,
             (OS_OPT       )(OS_OPT_TASK_STK_CHK | OS_OPT_TASK_STK_CLR),
             (OS_ERR      *)&err);

   OSStart(&err);     /* Start multitasking */

   return (1);
}
```

```
/*********************************************************************
*                    STARTUP TASK
*********************************************************************/

static  void  AppTaskStart (void *p_arg){
    CPU_INT32U  cpu_clk_freq;
    CPU_INT32U  cnts;

    OS_ERR      os_err;

    (void)p_arg;        /* Note #1 */

    BSP_Init();         /* Initialize BSP functions */
    CPU_Init();         /* Initialize the uC/CPU services */

    cpu_clk_freq = BSP_CPU_ClkFreq(); /* Determine SysTick reference freq. */

/* Determine nbr SysTick increments  */
cnts = cpu_clk_freq / (CPU_INT32U)OSCfg_TickRate_Hz;

    OS_CPU_SysTickInit(cnts);  /* Init uC/OS periodic time src (SysTick). */

#if OS_CFG_STAT_TASK_EN > 0u
    OSStatTaskCPUUsageInit(&os_err);  /* Compute CPU capacity with no task running */
#endif

    AppObjCreate();      /* Create Application Events */

    AppTaskCreate();     /* Create application tasks */

    while (DEF_TRUE) {   /* Task body, always written as an infinite loop. */

        BSP_LED_Toggle(1);

        OSTimeDlyHMSM(0, 0, 0, 200, OS_OPT_TIME_HMSM_STRICT, &os_err);
    }
}
```

```
/************************************************************
*                    AppObjCreate()
************************************************************/

static  void  AppObjCreate (void)
{
   CPU_INT08U  os_err;


   OSSemCreate((OS_SEM *) &AppLED3Sem,
            (CPU_CHAR    *) "LED4 semaphore",
            (OS_SEM_CTR   ) 0,
            (OS_ERR *) &os_err);

   OSQCreate((OS_Q *) &AppLED4MsgQ,
           (CPU_CHAR  *) "App LED4 MsgQ",
           (OS_MSG_QTY ) 10,
           (OS_ERR *) &os_err);

        /* Create software timer event */
   OSTmrCreate((OS_TMR              *)&AppLED3Tmr,
            (CPU_CHAR           *)"Blink LED3 timer",
            (OS_TICK             ) 1,
            (OS_TICK             ) 50,
            (OS_OPT              ) OS_OPT_TMR_PERIODIC,
            (OS_TMR_CALLBACK_PTR  ) AppLED3TmrCallBack,
            (void               *) 0,
            (OS_ERR             *)&os_err);

        /* Create software timer event */
   OSTmrCreate((OS_TMR *) &AppLED4Tmr,
            (CPU_CHAR *) "Blink LED 4 timer",
            (OS_TICK) 1,
            (OS_TICK ) 25,
            (OS_OPT) OS_OPT_TMR_PERIODIC,
            (OS_TMR_CALLBACK_PTR  ) AppLED4TmrCallBack,
            (void *) &AppLEDNbr,
            (OS_ERR *)&os_err);

}
```

```
/*****************************************************************
*                    AppTaskCreate()
*****************************************************************/

static  void  AppTaskCreate (void)
{
   OS_ERR  err;

   OSTaskCreate ((OS_TCB *) &AppTaskLED2TCB,
             (CPU_CHAR *) "App Task LED2",
             (OS_TASK_PTR ) AppTaskLED2,
             (void *) 0,
             (OS_PRIO ) 1,
             (CPU_STK  *)&AppTaskLED2Stk[0],
             (CPU_STK_SIZE ) APP_CFG_TASK_LED2_STK_SIZE_LIMIT,
             (CPU_STK_SIZE ) APP_CFG_TASK_LED2_STK_SIZE,
             (OS_MSG_QTY   ) 0,
             (OS_TICK ) 0,
             (void  *) 0,
             (OS_OPT ) (OS_OPT_TASK_STK_CHK | OS_OPT_TASK_STK_CLR),
             (OS_ERR *) &err);

   OSTaskCreate ((OS_TCB  *) &AppTaskLED3TCB,
             (CPU_CHAR  *) "App Task LED3",
             (OS_TASK_PTR ) AppTaskLED3,
             (void  *) 0,
             (OS_PRIO ) 1,
             (CPU_STK *) &AppTaskLED3Stk[0],
             (CPU_STK_SIZE ) APP_CFG_TASK_LED3_STK_SIZE_LIMIT,
             (CPU_STK_SIZE ) APP_CFG_TASK_LED3_STK_SIZE,
             (OS_MSG_QTY) 0,
             (OS_TICK ) 0,
             (void  *) 0,
             (OS_OPT ) (OS_OPT_TASK_STK_CHK | OS_OPT_TASK_STK_CLR),
             (OS_ERR *) &err);

   OSTaskCreate ((OS_TCB *) &AppTaskLED4TCB,
             (CPU_CHAR *) "App Task LED4",
             (OS_TASK_PTR ) AppTaskLED4,
             (void  *) 0,
             (OS_PRIO ) 2,
             (CPU_STK *) &AppTaskLED4Stk[0],
             (CPU_STK_SIZE ) APP_CFG_TASK_LED4_STK_SIZE_LIMIT,
             (CPU_STK_SIZE ) APP_CFG_TASK_LED4_STK_SIZE,
             (OS_MSG_QTY ) 0,
             (OS_TICK ) 0,
             (void  *) 0,
             (OS_OPT ) (OS_OPT_TASK_STK_CHK | OS_OPT_TASK_STK_CLR),
             (OS_ERR *) &err);

}
```

```
/*********************************************************************
*                      AppTaskLED2()
*********************************************************************/

static  void  AppTaskLED2(void *p_arg)
{
   OS_ERR   os_err;

   (void) p_arg;

   while(DEF_TRUE) {
      BSP_LED_Toggle(2);
      OSTimeDlyHMSM(0, 0, 0, 100, OS_OPT_TIME_HMSM_STRICT, &os_err);
   }
}

/*********************************************************************
*                  AppTaskLED3()
*********************************************************************/

static  void  AppTaskLED3(void *p_arg)
{
   CPU_BOOLEAN  status;
   OS_ERR  os_err;

   (void) p_arg;

   while (DEF_TRUE) {
      status = OSTmrStart(&AppLED3Tmr, &os_err);

      if (status != DEF_OK) {
         while (DEF_TRUE) {
            ;
         }
      } else {
         OSSemPend((OS_SEM   *)&AppLED3Sem,
                   (OS_TICK   ) 0,
                   (OS_OPT    ) OS_OPT_PEND_BLOCKING,
                   (CPU_TS   *) 0,
                   (OS_ERR   *)&os_err);
      }

      BSP_LED_Toggle(3);
   }
}



void  AppLED3TmrCallBack (OS_TMR   *p_tmr, void *p_arg)
{
   OS_ERR  os_err;


   OSSemPost((OS_SEM   *)&AppLED3Sem, (OS_OPT    )OS_OPT_POST_ALL, (OS_ERR   *)&os_err);
}
```

```
/************************************************************************
*                AppTaskLED4()
************************************************************************/


static  void  AppTaskLED4(void *p_arg)
{
   CPU_INT32U      *p_msg;
   CPU_INT08U       led_nbr;
   OS_MSG_SIZE      msg_size;
   CPU_BOOLEAN      status;
   OS_ERR           os_err;

   (void) p_arg;

   AppLEDNbr = 2;

   status = OSTmrStart(&AppLED4Tmr, &os_err);

   while (DEF_TRUE) {
      if (AppLEDNbr == 5) {
        AppLEDNbr = 3;
      }

      p_msg = (CPU_INT32U *) OSQPend((OS_Q        *)&AppLED4MsgQ,
                                     (OS_TICK      ) 0,
                                      (OS_OPT       ) OS_OPT_PEND_BLOCKING,
                                     (OS_MSG_SIZE *)&msg_size,
                                     (CPU_TS      *) 0,
                                      (OS_ERR      *)&os_err));

      if (os_err == OS_ERR_NONE) {
         led_nbr = (CPU_INT08U) *p_msg;
      }

   BSP_LED_Toggle(led_nbr);
   AppLEDNbr++;
   }
}



void  AppLED4TmrCallBack(OS_TMR *p_tmr, void   *p_arg){

   OS_ERR      os_err;

  OSQPost((OS_Q *) &AppLED4MsgQ,
         (void  *) p_arg,
         (OS_MSG_SIZE ) sizeof(CPU_INT08U),
         (OS_OPT ) OS_OPT_POST_FIFO,
         (OS_ERR  *) &os_err);

}
```