CE-235 Sistemas Embarcados de Tempo Real

Instituto Tecnológico de Aeronáutica - ITA

Laboratório
2.5

Sprint 01

LUCAS NICOLLI TOSI

Sembro de 2019

1. **Ao iniciar o debug o Processor Expert é inicializado:**

```
Debug
lab2_Debug_PNE [GDB PEMicro Interface Debugging]
  lab2.elf
    Thread #1 (Suspended : Breakpoint)
      main() at main.c:55 0x7a8
  C:\Freescale\KDS_v3\eclipse\plugins\com.pemicro.debug.gdbjtag.pne_2.3.6.201602211227\win32\pegdbserver_console
  arm-none-eabi-gdb
```

```c
main.c
/*lint -save  -e970 Disable MISRA rule (6.3) checking. */
int main(void)
/*lint -restore Enable MISRA rule (6.3) checking. */
{
  /* Write your local variable definition here */

  /*** Processor Expert internal initialization. DON'T REMOVE THIS CODE!!! ***/
  PE_low_level_init();
  /*** End of Processor Expert internal initialization.                    ***/

  /* Write your code here */
  /* For example: for(;;) { } */

  /*** Don't write any code pass this line, or it will be deleted during code generation. ***/
  /*** RTOS startup code. Macro PEX_RTOS_START is defined by the RTOS component. DON'T MODIFY THIS CODE!!! ***/
  #ifdef PEX_RTOS_START
    PEX_RTOS_START();                  /* Startup of the selected RTOS. Macro is defined by the RTOS component. */
  #endif
  /*** End of RTOS startup code.  ***/
  /*** Processor Expert end of main routine. DON'T MODIFY THIS CODE!!! ***/
  for(;;){}
  /*** Processor Expert end of main routine. DON'T WRITE CODE BELOW!!! ***/
```

2. **O método que inicializa a tarefa é utilizado pelo Processor Expert:**

```c
tasks.c   Task1.c
**     This component module is generated by Processor Expert. Do not modify it.
/*!
** @file Task1.c
** @version 01.00
*/
/*!
**   @addtogroup Task1_module Task1 module documentation
**   @{
*/

/* MODULE Task1. */

#include "os_tasks.h"
#include "Task1.h"

/* Define resources for a task statically */
OSA_TASK_DEFINE(Task1, TASK1_TASK_STACK_SIZE);

/*
** ===================================================================
**     Method      :  Init (component OS_Task)
**
**     Description :
**         The method creates and starts task defined by OS_Task
**         component.
**         This method is internal. It is used by Processor Expert only.
** ===================================================================
*/
osa_status_t Task1_Init(void)
{
  /* Create "Task1" task */
  if (OSA_TaskCreate(Task1_task,       /* The task function entry */
                     (uint8_t *)TASK1_TASK_NAME, /* The name of this task */
                     TASK1_TASK_STACK_SIZE, /* The stack size in byte */
                     Task1_stack,      /* Pointer to the stack */
                     TASK1_TASK_PRIORITY, /* Initial priority of the task */
                     (task_param_t)(NULL), /* Pointer to be passed to the task when it is created */
                     false,            /* This task will use not float register */
                     &Task1_task_handler) /* Pointer to the task handler */
                                != kStatus_OSA_Success) {
    return kStatus_OSA_Error;
  }
  return kStatus_OSA_Success;
}
```

3. **De acordo com a descrição da API, quando uma task é criada, a mesma é adicionada a uma lista de tarefas prontas para serem executadas:**

C/C++ - lab2/SDK/rtos/FreeRTOS/include/task.h - Kinetis Design Studio - C:\Users\Intos\workspace.kds

File Edit Source Refactor Navigate Search Project Run Processor Expert Window Help

task.h

lab2/SDK/rtos/FreeRTOS/include/task.h

```
/*-----------------------------------------------------------
 * TASK CREATION API
 *-----------------------------------------------------------*/

/**
 * task. h
 *<pre>
 BaseType_t xTaskCreate(
                            TaskFunction_t pvTaskCode,
                            const char * const pcName,
                            uint16_t usStackDepth,
                            void *pvParameters,
                            UBaseType_t uxPriority,
                            TaskHandle_t *pvCreatedTask
                        );</pre>
 *
 * Create a new task and add it to the list of tasks that are ready to run.
 *
 * xTaskCreate() can only be used to create a task that has unrestricted
 * access to the entire microcontroller memory map.  Systems that include MPU
 * support can alternatively create an MPU constrained task using
 * xTaskCreateRestricted().
 *
 * @param pvTaskCode Pointer to the task entry function.  Tasks
 * must be implemented to never return (i.e. continuous loop).
 *
 * @param pcName A descriptive name for the task.  This is mainly used to
 * facilitate debugging.  Max length defined by configMAX_TASK_NAME_LEN - default
 * is 16.
 *
 * @param usStackDepth The size of the task stack specified as the number of
 * variables the stack can hold - not the number of bytes.  For example, if
 * the stack is 16 bits wide and usStackDepth is defined as 100, 200 bytes
 * will be allocated for stack storage.
 *
 * @param pvParameters Pointer that will be used as the parameter for the task
 * being created.
 *
 * @param uxPriority The priority at which the task should run.  Systems that
 * include MPU support can optionally create tasks in a privileged (system)
 * mode by setting bit portPRIVILEGE_BIT of the priority parameter.  For
 * example, to create a privileged task at priority 2 the uxPriority parameter
 * should be set to ( 2 | portPRIVILEGE_BIT ).
 *
 * @param pvCreatedTask Used to pass back a handle by which the created task
 * can be referenced.
 *
 * @return pdPASS if the task was successfully created and added to a ready
 * list, otherwise an error code defined in the file projdefs.h
 *
 * Example usage:
   <pre>
 // Task to be created.
 void vTaskCode( void * pvParameters )
 {
     for( ;; )
     {
```

4. **Upload do LAB2.5 no GitHub: https://github.com/Intosi/CE-235**