REAL TIME OPERATING SYSTEM PROGRAMMING-I: µC/OS-II and VxWorks

Lesson-3: µC/OS-II System level and task Functions

Chapter-11 L03: "Embedded Systems - Architecture, Programming and Design", Raj Kamal, Publs.: McGraw-Hill Education

1. System Level Functions



Chapter-11 L03: "Embedded Systems - Architecture, Programming and Design", Raj Kamal, Publs.: McGraw-Hill Education

OSInit()

void OSInit (void) At the beginning prior to the OSStart ()

Function <u>void</u> OSInit (<u>void</u>) to initiate the operating system

 Use is compulsory before calling any OS kernel functions

• Refer Example 11.1- Step 2.

OSStart () and OSTickInit ()

• void OSStart (void)

After OSInit () and task-creating function(s)
void OSTickInit (void)
In first task function that executes once. Initializes the system timer ticks (RTC interrupts)

OSStart ()

- Function <u>void</u> OSStart (<u>void</u>) to start the initiated operating system and created tasks Its use is compulsory for the multitasking OS kernel operations
- Refer Example 11.2- Step 4.

2. Programming Examples— OS Init and OS Start

Chapter-11 L03: "Embedded Systems - Architecture, Programming and Design", Raj Kamal, Publs.: McGraw-Hill Education Step *i*: Initiating the RTOS void main (void) { OSInit (); /* Create a task */

۰

٠

•

• •

Step j: Starting the RTOS

void main (void) {
OSInit ();
/* Create a task */

./*Start the RTOS */ OSStart ()

۲

٠

•

•

Chapter-11 L03: "Embedded Systems - Architecture, Programming and Design", Raj Kamal, Publs.: McGraw-Hill Education

3. Interrupt Service Task (ISR) Start and End

OSIntEnter () and OSIntExit ()

• void OSIntEnter (void) Just after start of the ISR codes OSIntExit must call just before the return from ISR • void OSIntExit (void) After the OSIntEnter () is called just after the start of the ISR codes and OSIntExit is called just before the return from ISR.

OSIntEnter()

- Function void OSIntEnter (void) - used at the start of ISR For sending a message to RTOS kernel for taking control—compulsory to let OS kernel control the nesting of the ISRs in case of occurrences of multiple interrupts of varying priorities
- Refer Example 11.3- Step 2.

OSIntExit()

Function void OSIntExit (void)
used just before the return from the running ISR

For sending a message to RTOS kernel for quitting control of presently running ISRRefer Example 11.4- Step 4.

4. Critical Section Start and End

Critical Section

OS ENTER CRITICAL – Macro to disable interrupts before a critical section • OS EXIT CRITICAL – Macro to enable interrupts. [ENTER] and EXIT functions form a pair in the critical section]

OS_ENTER_CRITICAL

• OS_ENTER_CRITICAL

2015

- used at the start of a ISR or task - for sending a message to RTOS kernel and disabling the interrupts

use compulsory when the OS kernel is to take note of and disable the interrupts of the system
Refer Example 11.5- Step 3.

OS_EXIT_CRITICAL

• OS_EXIT_CRITICAL

- used at the end of critical section

- for sending a message to RTOS kernel and enabling the interrupts
- Use is compulsory to OS kernel for taking note
- of and enable the disabled interrupts.
- •Refer Example 11.6- Step 5.

5. System Clock Tick Initiate

OSTickInit () and OS_TICKS_PER_SEC

Function <u>void</u> OSTickInit (void)

 is used to initiate the system clock
 ticks and interrupts at regular intervals as
 per OS_TICKS_PER_SEC predefined
 when defining configuration of MUCOS

Refer Example 11.7- Steps 2 and 10.

2015

Chapter-11 L03: "Embedded Systems - Architecture, Programming and Design", Raj Kamal, Publs.: McGraw-Hill Education

6. Task Service Functions

Task Service Functions

 Service functions mean the functions of multitasking service (task create, suspend or resume), time setting and time retrieving (getting) functions.

OSTaskCreate

• unsigned byte **OSTaskCreate** (void (*task) (void *taskPointer), void *pmdata, OS_STK *taskStackPointer, unsigned byte taskPriority) Called for creating a task • Refer Example 11.7 Steps 1, 2, 5 and 8

OSTaskCreate...

*taskPointer

2015

a pointer to the codes of the task being created

• *pmdata— pointer for an optional message data reference passed to the task. If none, assign as NULL Exemplary use of the pointers and stack for task creation

• Refer Example 11.7 step 5 • OS TASK CREATE EN Must be preprocessor directive to enable inclusion of task management functions by MUCOS • Refer Example 11.7- Step 1 Statement 1

Specifying Maximum Number of Tasks

• Assume that system tasks are of high priority 0 to 7. Each user-task is to be assigned a *priority*, which must be set between 8 and OS MAX TASKS +7 [or 8 and OS_LOWEST_PRIORITY - 8]. If maximum number of user tasks OS MAX TASKS is 8, the priority can be set between 8 and 15. Refer Example 11.7- Step 1 Statement 1

Specifying OS_LOWEST_PRIO

OS_LOWEST_PRIO must be set at 23 for eight user tasks of priority between 8 and 15, because MUCOS will assign priority = 15 to lowest priority task
System low priority tasks will be assigned priorities between 16 and 23 .
Refer Example 11.7- Step 1 Statement 2.

OSTaskSuspend and OSTaskResume

- unsigned byte OSTaskSuspend (unsigned byte taskPriority)
- Called for blocking a task (Example 11.8 Step 12)
- unsigned byte OSTaskResume (unsigned byte taskPriority)
- Called for resuming a blocked task (Example 11.9 Step 20)

7. Programming Method and Example Preprocessor commands, main and Task Creation

Programming Method

• Method: RTOS after start first runs FirstTask, and then FirstTask *creates* all the application tasks and *initiates* system clock ticks

• later suspend itself in infinite while loop

Step A: Program preprocessor commands #define OS MAX TASKS 8 #define OS LOWESTORIO 15 #define OS MAX TASKS #define OS LOWESTPRIO 15 #define OS TASK CREATE EN 1 #define OS TASK DEL EN 1 #define OS TASK SUSPEND EN 1 #define OS TASK RESUME EN 1 #define OS TICS PER SEC 100

> Chapter-11 L03: "Embedded Systems - Architecture, Programming and Design", Raj Kamal, Publs.: McGraw-Hill Education

Step B: Global functions and their parameters declarations #define FirstTaskPriority 8 #define FirstTaskStackSize 100 /* Define other task-priorities & stacksizes*/ static void FirstTask(*taskPointer); static OS_STK FirstTask [FirstTaskStackSize] Step B: Global functions and their parameters declarations... #define task1Priority 9 #define task1StackSize 100 /* Define other task-priorities & stacksizes*/ static void task1(*taskPointer); static OS_STK task1 [task1StackSize] Step B: Global functions and their parameters declarations...

#define task2Priority 10
#define task2StackSize 100
/* Define other task-priorities & stacksizes*/
static void task2(*taskPointer);
static OS_STK task2 [task2StackSize]

Step C: Main function

void main (void) { OSInit (); /* Create First task */ OSTaskCreate (FirstTask, void (*) 0,(void *)&FirstTaskStack[FirstTaskStackSize], FirstTaskPriority); OSStart ();

Step D: First task for starting system clock and creating application tasks static void FirstTask (void *taskPointer) { /*System clock time set */ OSTaskCreate (task1, void (*) 0,(void *)&task1Stack [task1StackSize], task1Priority); OSTaskCreate (task2, void (*) 0,(void *)&task2Stack [task2StackSize], task2Priority); /* Create All application related remaining tasks */

Step D: First task for starting system clock and creating application tasks...

OSTimeSet (presetTime); OSTickInit (); /* Initiate system timer ticking*/ /* Create application related highest priority tasks */

...; ...; ..; while (1) {...;

Step E: First task suspending indefinitely itself in while loop static void FirstTask (void *taskPointer) {

. while (1) { OSTaskSuspend (FirstTaskPriority); }

۲

Step F: Application Tasks— task1 and task2

static void task1 (void *taskPointer) {

. while (1) {

٠

٠

٠

2015

Chapter-11 L03: "Embedded Systems - Architecture, Programming and Design", Raj Kamal, Publs.: McGraw-Hill Education

Step F: Application Tasks— task1 and task2... static void task2 (void *taskPointer) {

. while (1) {

•

٠

2015

Chapter-11 L03: "Embedded Systems - Architecture, Programming and Design", Raj Kamal, Publs.: McGraw-Hill Education



Chapter-11 L03: "Embedded Systems - Architecture, Programming and Design", Raj Kamal, Publs.: McGraw-Hill Education

We learnt

- Initiating OS, starting OS, creating tasks, setting system clock tick rate and tick initiate functions and
- An example in which the OS on start, first runs FirstTask and then FirstTask creates application tasks and initiate system clock ticks and later suspend itself in infinite while loop.

We learnt

2015

• MUCOS creating, defining priority and stack-size, deleting, suspending and resuming functions for task servicing

End of Lesson-2 of Chapter 11 on µC/OS-II System level and task Functions

