OSEK Standard and experiments on microcontroller devices

Paolo Gai
Evidence Srl
pj@evidence.eu.com

summary

- the hardware
- example 1 – ISR2 and tasks
- example 2 – application modes and resources
- example 3 – events, alarms, ErrorHook, ORTI

the hardware

- the evaluation board used is a FLEX board (Light or Full) with a Demo Daughter board

- during the examples, we'll use the following devices:
  - the DSPIC MCU
  - 1 timer
  - a button
    - used to generate interrupts when pressed or released
    - also used as external input
  - leds
  - 16x2 LCD

Example 1 – Tasks and ISR2

- The demo shows the usage of the following primitives: DeclareTask – ActivateTask – TerminateTask - Schedule

- Demo structure
  - The demo is consists of two tasks, Task1 and Task2.
  - Task1 repeatedly puts on and off a sequence of LEDs
  - Task2 simply turns on and off a LED, and
    - is activated by the press of a button. Task2 is de facto a disturbing task that, depending on the configuration parameters,
    - may preempt Task1

Ex. 1 Configuration 1: Full preemptive

- This configuration is characterized by the following properties:
  - periodic interrupt → Task1 activation → LED 0 to 5 blink
  - button → Task2 activation → Task2 always preempts Task1, blinks LED 6/7 and prints a message

Notes:

- Task2 is automatically activated by StartOS
  - AUTOSTART=TRUE
- Conformance Class is BCC1
  - lost activations if the button pressed too fast!

Ex. 1 Configuration 2: Non preemptive

- Task1 is NON preemptive
- Task2 runs only when Task1 does not run
  - LEDs 6 and 7 does not interrupt the ChristmasTree
- IRQs are not lost, but task activations may be

Ex. 1 Configuration 3: Preemption points

- Task1 calls Schedule in the middle of the Christmas tree

- Result:
  - Task2 can now preempt Task1 in the middle of the Christmas tree

Ex. 4 Configuration 4: Multiple Activations.

- BCC2 Conformance class
- Task2 can now store pending activations, which are executed whenever possible

Example 2 - Resources and App. modes

- The demo shows the usage of the following primitives: GetActiveApplicationMode, GetResource, ReleaseResource

- Demo structure
  - Two tasks, LowTask and HighTask. They share a resource.
  - LowTask is a periodic low priority task, activated by a timer, with a long execution time.
  - Almost all its execution time is spent inside a critical section. LED 0 is turned on when LowTask is inside the critical section.
  - HighTask is a high priority task that increments (decrements) a counter depending on the application mode being ModeIncrement (ModeDecrement). The task is aperiodic, and is activated by the ISR linked to the button.

Example 2 - Resources and App. modes (2)

- Application Modes are used to implement a task behavior dependent on a startup condition
- (ERIKA specific) HighTask and LowTask are configured to share the same stack by setting the following line inside the OIL task properties:

  STACK = SHARED;

Example 3 - Event and Alarm API Example

- The demo shows the usage of the following primitives:

  WaitEvent, Getevent, ClearEvent, SetEvent, ErrorHook, StartupHook, SetRelAlarm, CounterTick

- Demo structure:
  - The demo consists of two tasks, Task1 and Task2.
  - Task1 is an extended task. Extended tasks are tasks that:
    - can call blocking primitives (WaitEvent)
    - must have a separate stack
  - A task is considered an Extended Task when the OIL file includes events inside the task properties.
  - Task1 waits for two events:
    - Timer → CounterTick → AlarmTask1 → TimerEvent → LED 1
    - Button IRQ → SetEvent(ButtonEvent) → LED 2

Example 3 - Event and Alarm API Example (2)

- Button press → ISR2 →SetRelAlarm(AlarmTask2) → Task2 activation → LED 3 on.
- ErrorHook → when the button is pressed rapidly twice
  - SetRelAlarm primitive called by the Button IRQ on an already armed alarm
- The alarm support is basically a wakeup mechanism that can be attached to application or external events (such as timer interrupts) by calling CounterTick to implement an asynchronous notification.
- (ERIKA Enterprise specific) Task1 needs a separate stack because it uses WaitEvent.

Example 3 - Event and Alarm API Example (3)

- Running the example
  - Timer Interrupt → Counter1 incremented.
  - AlarmTask1 → TimerEvent event set on Task1 → Task1 wakes up, get the event, and blinks LED 1.
  - The visible result is that LED 1 periodically blinks on the board.

  - button press → Task1 runs and LED 3 goes on and off
  - rapid button press → ErrorHook due to multiple calls of SetRelAlarm

  - ORTI Informations are available for this demo