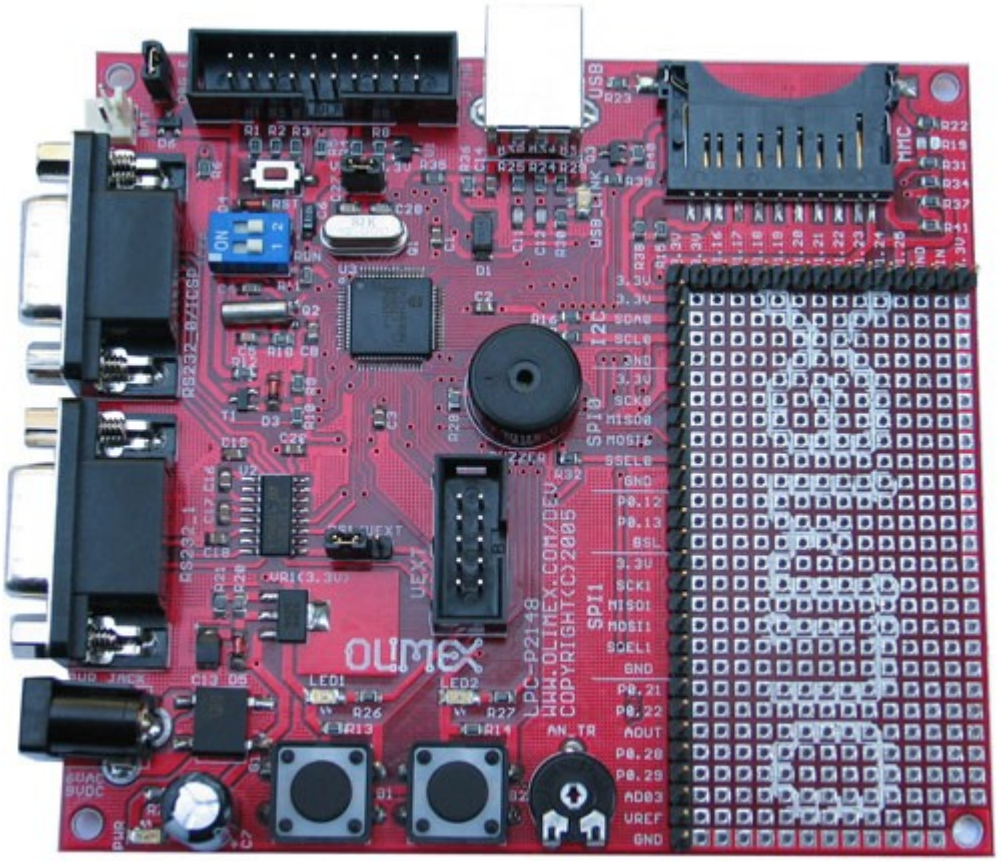
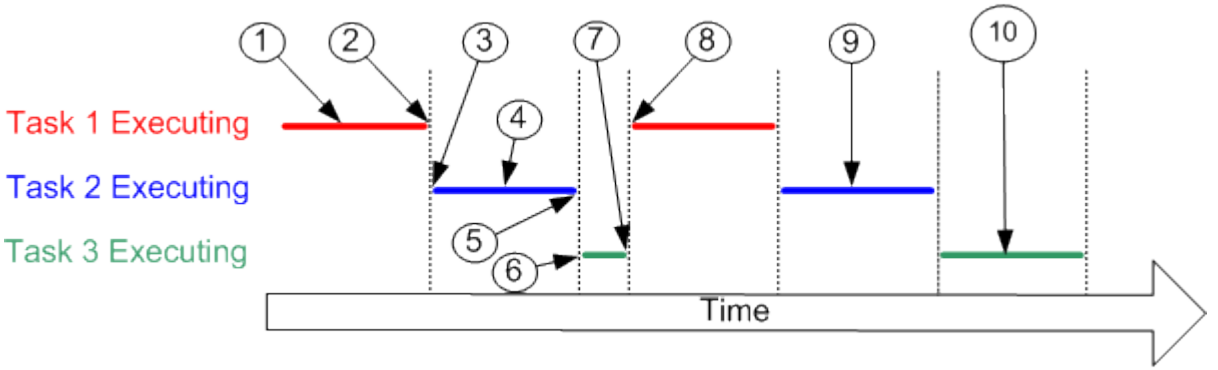
LPC2148's RTOS

Bruce Chhuon
4/10/07



What is a Real Time Operating System?

- A Real Time Operating System (RTOS) manages hardware and software resources.
- Deterministic - guarantees task completion at a set deadline.
- Real time operations are possible through multitasking. Task are broken into threads and scheduled to be processed based on priority



RTOS Importance

- An high priority signal such as one from an ABS system of an car should not be queued.
- Critical processes should be addressed immediately.
- A slow response may result in Death.

eCos

- Currently running on the LPC2148
- Drawbacks:
  - Can't shut off superfluous functions
  - C++ dependencies
  - Crazy build system
  - Overhead

FreeRTOS

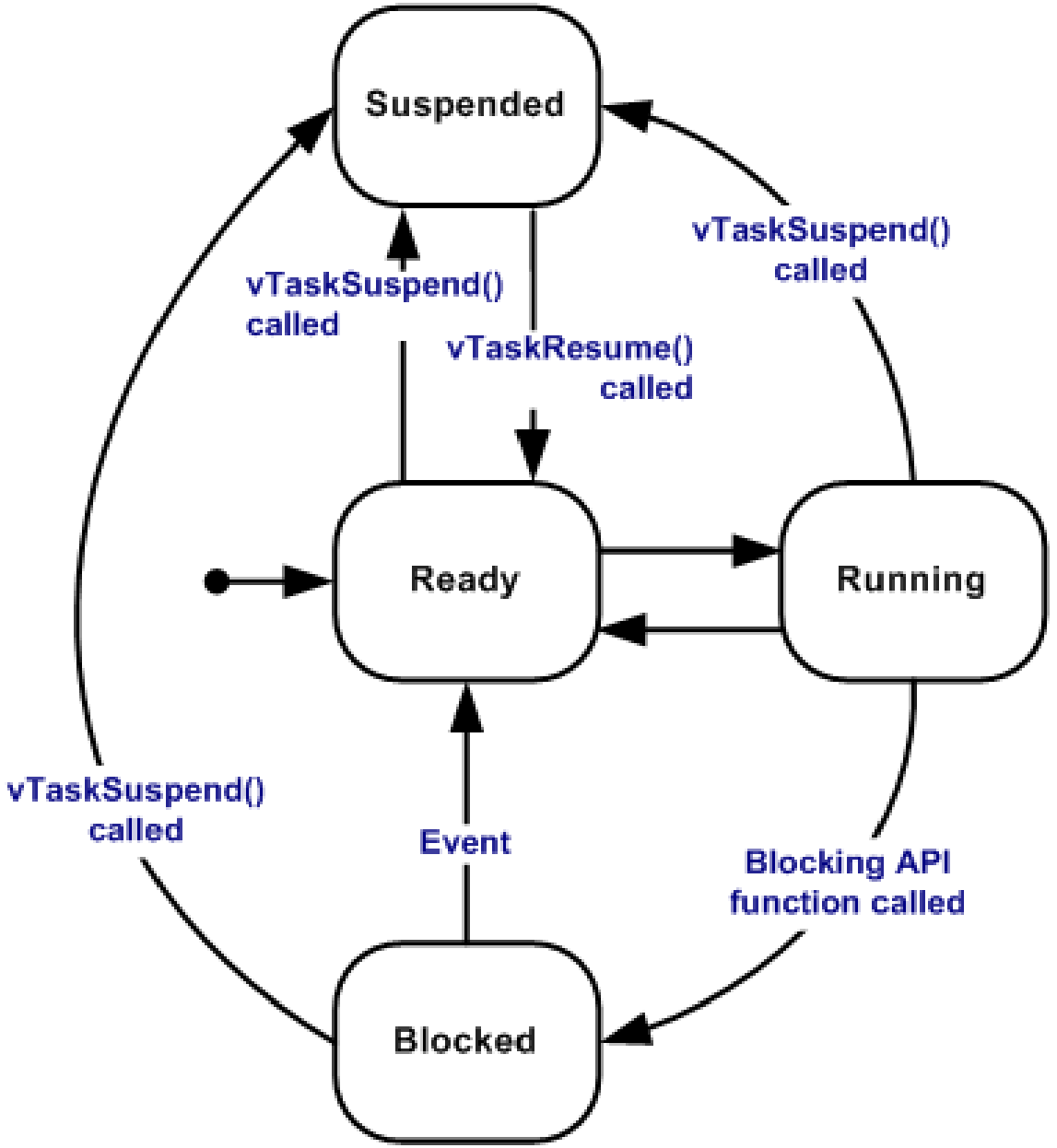
- Design to be:
  - Simple
  - Portable
  - Concise
- Primarily written in C.
- Few assembler functions
- Thumb mode supported
- Has been ported to the LPC2148
- Run both task and co-routines.

Task

- Real time application can be structure as a set of independent tasks.
- Tasks have their own stack.
- The scheduler is responsible for starting, stop, swapping in, and swapping out tasks.
- The scheduler is responsible for managing the processor context:
  - Registers values
  - Stack contents

Task have 4 states

- Running State
  - Task is utilising the processor
- Ready State
  - Task able to run
- Block State
  - Task waiting on an event
- Suspended State
  - Task are unavailable for scheduling


Suspended
vTaskSuspend()
called
vTaskResume()
called
vTaskSuspend()
called
Ready
Running
vTaskSuspend()
called
Event
Blocking API
function called
Blocked

Tasks Priorities

- The range of priorities is configurable 0 - desired amount
- Modify configMAX_PRIORITIES in FreeRTOSConfig.h
- Higher value means more RAM usage
- Higher priorities task run before lower priority task

Implementing a Task

- Task creation is done by xTaskCreate()
- Task deletion is done by vTaskDelete()
- Functions that implement a task should return void and take void pointers as its only parameter.

Template for a function implementing a task

```
void vATaskFunction( void *pvParameters )
{
    for( ;; )
    {
        -- Task application code here. --
    }
}
```

The Idle Task

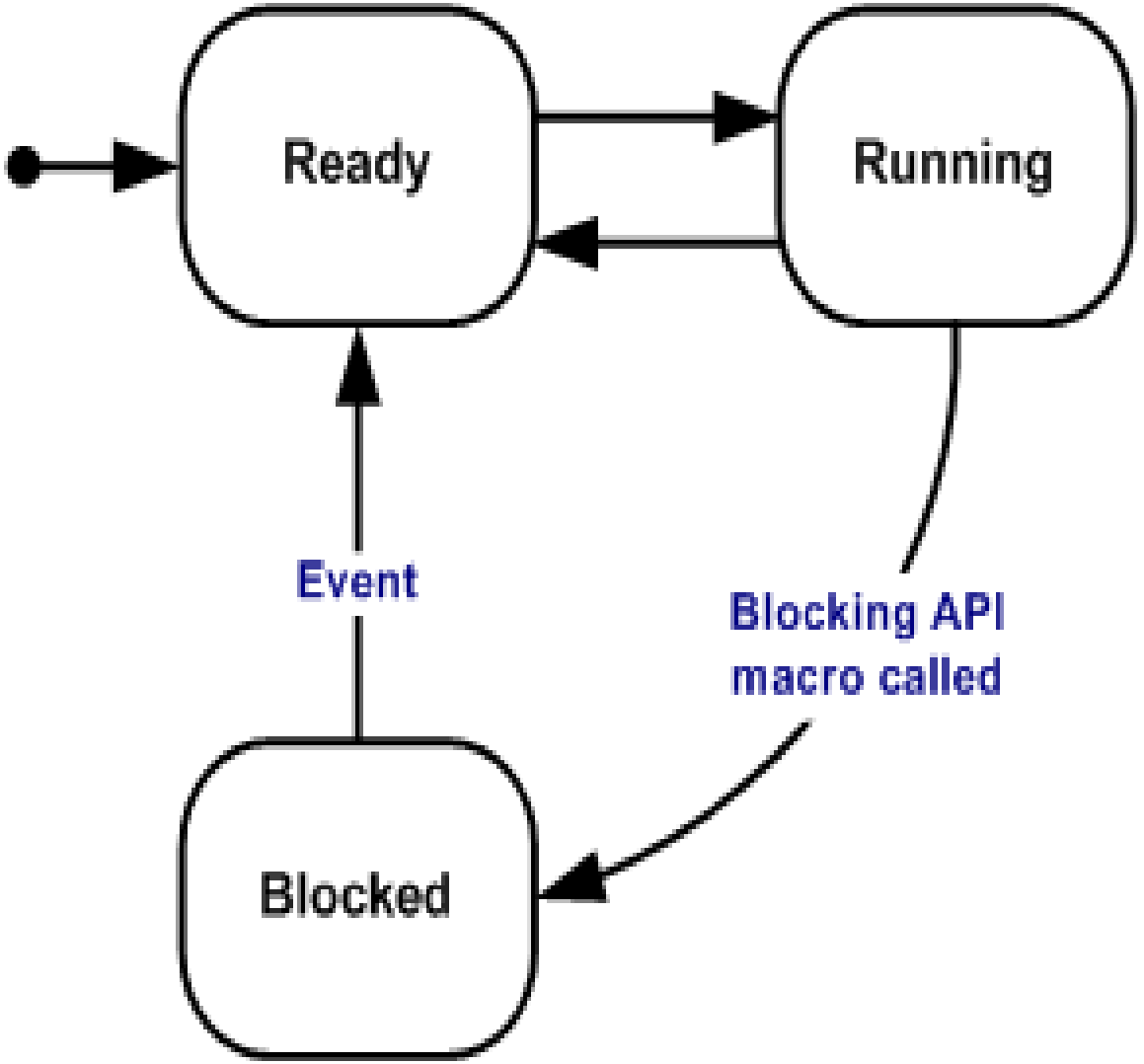
- Idle tasks are created automatically when the scheduler is started
- Responsible for freeing memory by removing deleted task
- Idle task should not be starve out of time

Co-routines

- Co-routine share a stack
- Co-routines are prioritized and scheduled with respect to other co-routines.
- Co-routine are implemented through a set of macros
- Restrictions on where the API call can be made

Co-routine States

- Running State
  - Task is utilising the processor
- Ready State
  - Task able to run
- Block State
  - Task waiting on an event
- Suspended State
  - May exist in future releases


Ready
Running
Event
Blocking API
macro called
Blocked

Co-routine Priorities

- The range of priorities is configurable 0 - desired amount
- Modify configMAX_PRIORITIES in FreeRTOSConfig.h
- Higher value means more RAM usage
- Tasks have priority over co-routine

Implementing a Co-routine

- Create by calling xCoRoutineCreate()
- Must start with crSTART()
- Must end with crEND()
- Usually implemented as a continuous loop
- A co-routine function can create other co-routines
- Functions which implement a co-routine should return void, have xCoRoutineHandle and an index as parameters.

Template for functions implementing a co-routine

```
void vACoRoutineFunction( xCoRoutineHandle xHandle,
unsigned portBASE_TYPE uxIndex )
    {
        crSTART( xHandle );

        for( ;; )
        {
            -- Co-routine application code here. --
        }

        crEND();
    }
```

Mixing Task and Co-routines

- Co-routine should be schedule when idle.
- Co-routines are executed when there are no tasks queued to be in the running state.
- Co-routines consume less memory, but are more restrictive and more complex than a task.

Limitation and Restrictions

- When a co-routine is blocked, its stack is not protected.
- Variable on the stack can changed
- To overcome this problem the variable should be declared as static
- API function that call to block co-routines can only be made from the co-routine itself.
- Blocking calls can't be made from switch statements

Memory Usage for STR7x ARM7 Port

- Kernel
  - 4 KB of ROM
- Scheduler
  - 236 bytes
- Queue
  - 76 bytes + queue storage area
- Task
  - 64 bytes + stack size

Community

- Embedded Systems Discussion Groups
  - http://www.embeddedrelated.com/groups/lpc2000/1.php
- Yahoo groups
  - http://tech.groups.yahoo.com/group/lpc2000/
- Source Forge
  - http://sourceforge.net/forum/forum.php?forum_id=382005