

## Design and test of a task guardian for use in TTCS embedded systems

**Zemian M. Hughes and Michael J. Pont**  
Embedded Systems Laboratory  
University of Leicester

Presentation at UK Embedded Forum  
Birmingham, UK, 13 October 2004

Embedded Systems Laboratory  
<http://www.le.ac.uk/eg/embedded>

### Overview

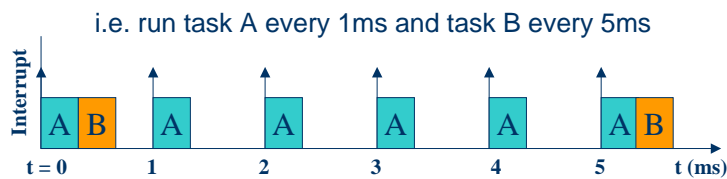
---

- In this presentation, I will briefly explain the normal operation of a “time-triggered co-operatively scheduled” (TTCS) system.
- I will then describe the impacts that a “task overrun” can have on a TTCS system
- My aim is then to discuss a basic solution to the “task overrun” problem, including various features to determine the required behaviour of the “task guardian”
- I will then describe implementation costs and a simple case study to show, and prove a working solution to the problem.

## The TTCS system

---

- Can be viewed as:
  - A simple operating system that allows tasks to be run periodically (Pont 2001)or as
  - A single shared timer interrupt service routine



3

Embedded Systems Laboratory  
<http://www.le.ac.uk/esrg/embedded>

## Features of a TTCS system

---

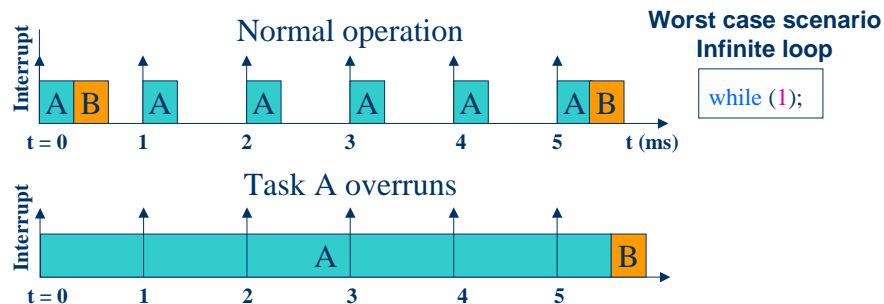
- Advantages of Co-operative Scheduling as identified by (Bates 2000)
  - The scheduler is simpler
  - Overheads are reduced
  - Testing is easier
  - Certification authorities tend to support this form of scheduling
- Overall, TTCS is known to provide a simple, low-cost and highly predictable platform which makes it ideal for safety-critical systems
- However there is a failure mode which can greatly impair system performance

4

Embedded Systems Laboratory  
<http://www.le.ac.uk/esrg/embedded>

## The Task Overrun

- Due to the architecture of single-tasking co-operatively scheduling – tasks must run to completion
- Assume two tasks, A and B. Where task A runs every millisecond and task B is “slack stealing” every 5ms



5

Embedded Systems Laboratory  
<http://www.le.ac.uk/esrg/embedded>

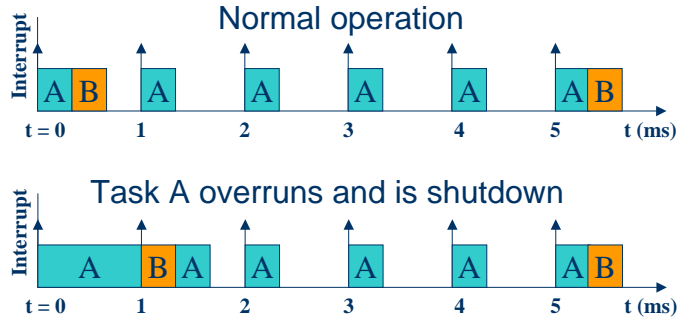
## Effects of “task overruns”

- A minimal task overrun, may only generate a tick offset error where the scheduler sequencing has been shifted a millisecond or more, and may not be apparent to the user
- A task overrun that is longer but still returns to the scheduler, often results in the system appearing very slow and sluggish
- A task overrun that does not return causes the system to hang indefinitely

6

Embedded Systems Laboratory  
<http://www.le.ac.uk/esrg/embedded>

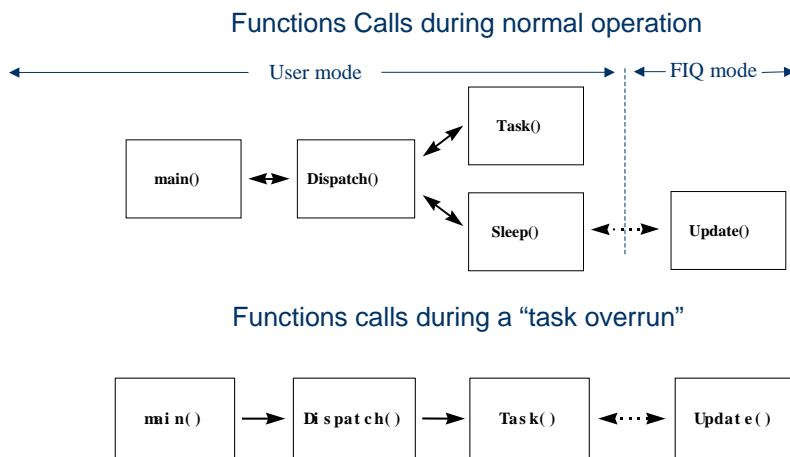
## Basic solution - "killing the task"



- Slack-stealing task B is forced to catch-up in the next tick interval
- The remaining tasks remain unaffected

7

## Analyzing the function calls of the scheduler



8

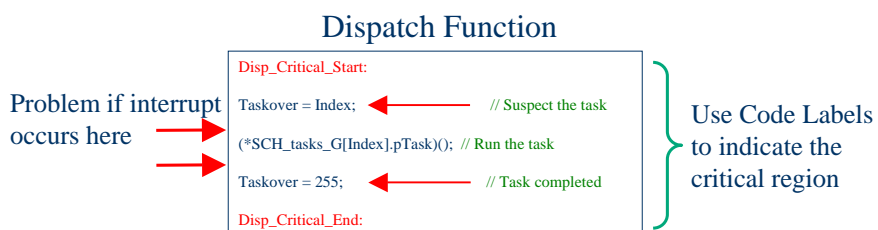
## Summary of “task guardian” behaviour

- The task guardian is mainly a revised version of Update – which shuts down any task found executing whilst the Update ISR is invoked
- This is carried out as follows:
  - The Update ISR will detect the task overrun
  - Update will return control to End\_Task (rather than the task), which locates the return address to Dispatch.
  - Control is passed to dispatch where – as far as possible – normal program operation continues

9

Embedded Systems Laboratory  
<http://www.le.ac.uk/esrg/embedded>

## Detecting task overruns



- Use a ‘Taskover’ variable to store the overrun Task ID
- Code labels prevents incorrect task overrun detection
- Use if statements in update task function to check that the return address is not within the critical code region.

10

Embedded Systems Laboratory  
<http://www.le.ac.uk/esrg/embedded>

## Returning from Update

---

- Due to ARM 7 architecture there are 3 ways in which the return address can be stored

### ATPCS Frame saved return address

```
// Set frame stored r14 return address
*(fp - 1) = (int)End_Task + 4;
```

### Stack saved return address

```
// Set stored r14 return address
*(ampFIQ_Stack_Base - 1) = (int)End_Task;
```

### Register saved return address

```
// Set r14 register
asm ("ldr r14, =End_Task + 4");
```

---

11

Embedded Systems Laboratory  
<http://www.le.ac.uk/esrg/embedded>

## Shutting down the task

---

- End\_Task does the following:
  - Determines if Task was a leaf function or not by comparing frame pointers
  - If not leaf, function calls are back traced to the base function in Task
  - The contents of the stack is located and returned to the appropriate registers
  - A return to Dispatch is then Issued
- Dispatch
  - Dispatch will run a backup task, if one exists.
  - Normal operation now continues

---

12

Embedded Systems Laboratory  
<http://www.le.ac.uk/esrg/embedded>

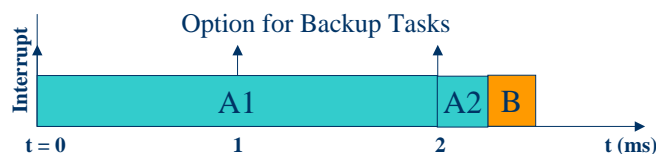
## Adding support of backup tasks

- The ability to shutdown overrunning tasks can improve the reliability of TTCS design
- However avoiding scheduler “jams” does not always solve the problem
- Therefore the use of backup tasks allows the user to specify there own recovery methods and decisions

13

Embedded Systems Laboratory  
<http://www.le.ac.uk/esrg/embedded>

## Backup tasks solution



```
// Setup backup task if one exists and not already running backup task
if ((SCH_tasks_G[Task_Ovrrun_G].pTask_2 > 0) & (SCH_tasks_G[Task_Ovrrun_G].pTask
    != SCH_tasks_G[Task_Ovrrun_G].pTask_2))
{
    // Set backup task
    SCH_tasks_G[Task_Ovrrun_G].pTask = SCH_tasks_G[Task_Ovrrun_G].pTask_2;
    // Set it to run immediately
    Backup_G = 2;
}
```

14

Embedded Systems Laboratory  
<http://www.le.ac.uk/esrg/embedded>

## Task Priority solution

---

- If the overrunning task has no backup task then it may be best to set it to the lowest priority
- This is achieved using an array of the task indexes
- The values are shifted around to indicate their priority level
- This saves on shifting all the data within the task array itself

---

15

## Implementation costs

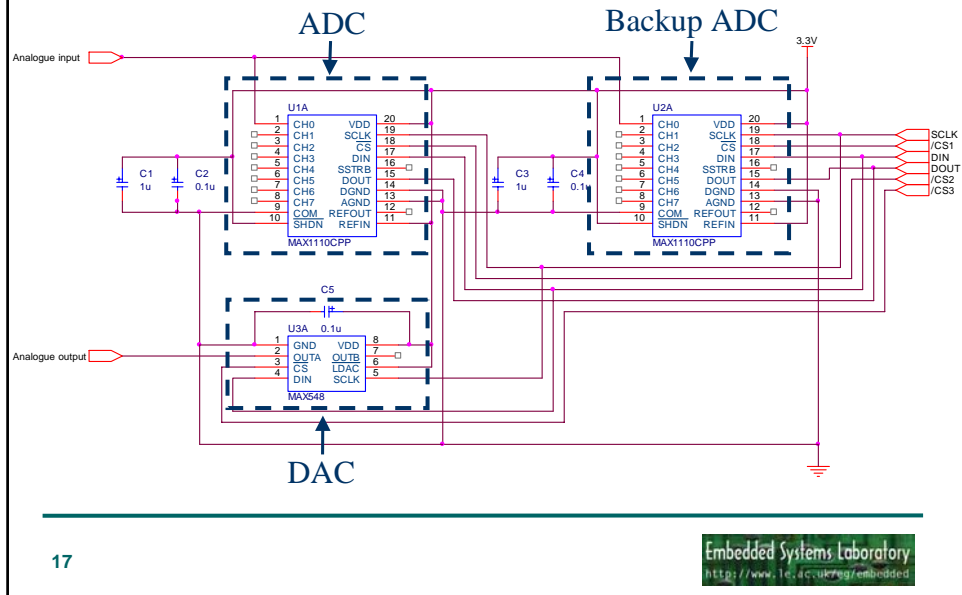
---

Feature	Instructions	Increase in instructions
Base Scheduler (reduced jitter)	163	-
Basic Task Guardian	214	31%
+ Update modification protection	259	59%
+ Recover stack variables	277	70%
+ Backup tasks feature	359	120%
+ Priority tasks feature	421	158%

---

16

## Case study (ADC DAC)



17

Embedded Systems Laboratory  
http://www.le.ac.uk/ereg/embedded

## Use of critical and backup functions

- This critical function will overrun every 10ms

```
// ADC and DAC critical function
int ADC_DAC_CRTL(void)
{
    static tByte Count;
    tByte data;

    // Overrun every 10 times the function is called
    if (++Count == 10)
    {
        Count = 0;
        // Generate an overrun
        while(1);
    }

    // Read analogue value from first ADC
    data = SPI_MAX1110_Read_Byte(0);

    // Send value to DAC
    SPI_MAX548A_Send_Byte(data);
    return 0;
}
```

18

Embedded Systems Laboratory  
http://www.le.ac.uk/ereg/embedded

## Backup function

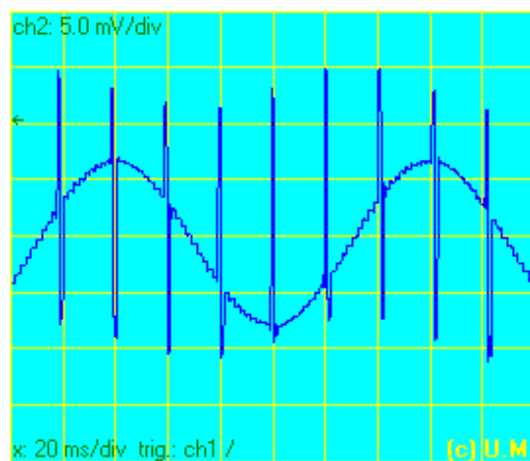
- Functions ending in 'CRTL' will be returned one function call up in the event of an overrun
- The return value of the function will be -1

```
// ADC and DAC calling function
int ADC_DAC_CALL(void)
{
    //Call critical function
    if (ADC_DAC_CRTL() == OVERRUN)
    {
        //If overrun call backup
        ADC_DAC_Backup();
    }
}
```

19

Embedded Systems Laboratory  
<http://www.ie.ac.uk/eresg/embedded>

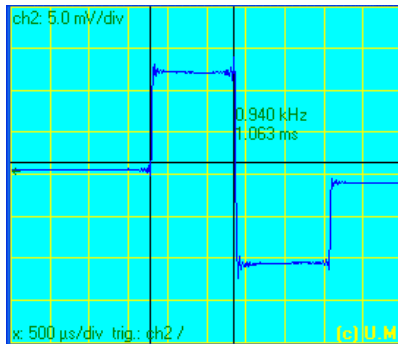
## Results: ADC DAC Test



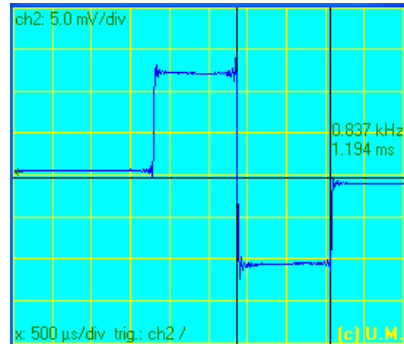
20

Embedded Systems Laboratory  
<http://www.ie.ac.uk/eresg/embedded>

## Results: Proof of Recovery



Task Overrun time



Backup Task time

21

Embedded Systems Laboratory  
<http://www.ie.ac.uk/eresg/embedded>

## Conclusions

- The TTCS scheduler was briefly described.
- The impact of task overruns explained
- Introduced the “task guardian” and features
- Illustrated the use of the task guardian in a simple case study

22

Embedded Systems Laboratory  
<http://www.ie.ac.uk/eresg/embedded>

## References

---

- Pont, M.J. (2001), "Patterns for time-triggered embedded systems: Building reliable applications with the 8051 family of micro controllers", Addison-Wesley / ACM Press.
- Bate, I. (2000) "Introduction to scheduling and timing analysis", in *"The Use of Ada in Real-Time System"* (6 April, 2000). IEE Conference Publication 00/034.
- ARM (1998), "The ARM-THUMB Procedure Call Standard", SWS ESPC 0002 A-05, <http://www.arm.com> .