

Reducing task jitter in shared-clock embedded systems using CAN

Mouaaz Nahas, Michael J. Pont & Aman Jain
Embedded Systems Laboratory
University of Leicester

Presentation at Embedded Systems Show
Birmingham, UK, 13 October 2004

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

Overview

- In this presentation, I will briefly explain the “shared-clock scheduling” (S-C) approach used in distributed systems.
- I will also demonstrate the levels of jitter result from the use of a S-C design running over CAN.
- I will then describe a modification to the original S-C algorithm which is intended to reduce the jitter levels.

Background

- Many modern embedded systems are based on a multi-processor architecture
- CAN is a popular and cost-effective choice for such systems
- CAN is very popular in the automotive sector and is also becoming widely used in process control and many other areas

3

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

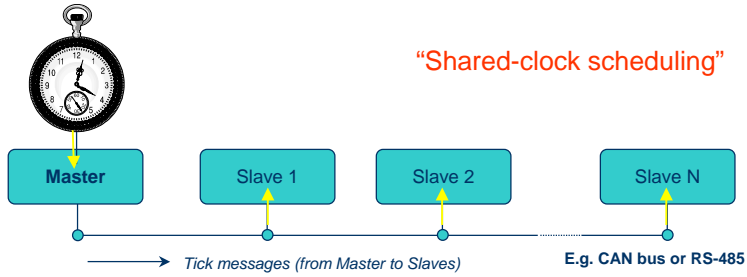
Background

- Some CAN features:
 - High speed
 - Low cost
 - Short data length
 - Fast reaction time
 - Broadcast and peer-to-peer communication
 - Error handling
- CAN is a message-object based
- Many microcontroller families have support for CAN

4

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

Shared-clock (S-C) architecture



On the Master node, the TTC scheduler is driven by interrupts from an on-chip timer

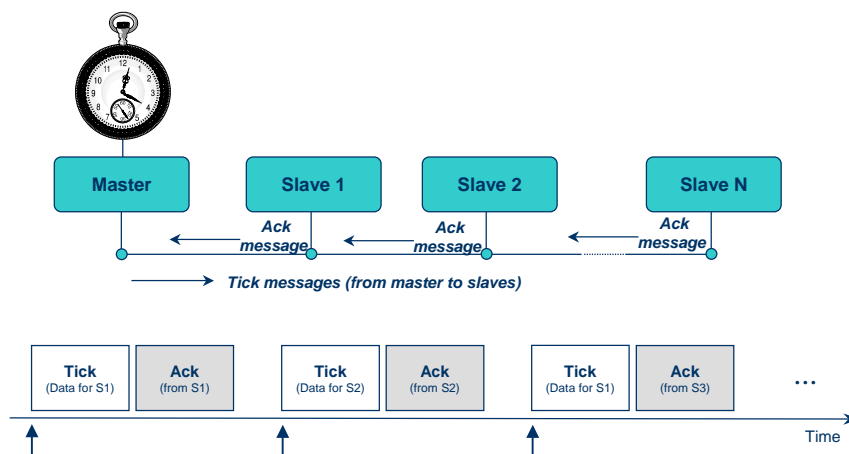
The Master sends periodic "Tick" messages to the slaves

On the Slave nodes, the scheduler is driven by interrupts generated through the arrival of messages from the Master

5

Embedded Systems Laboratory
<http://www.10.ac.uk/ereg/embedded>

Shared-clock (S-C) architecture

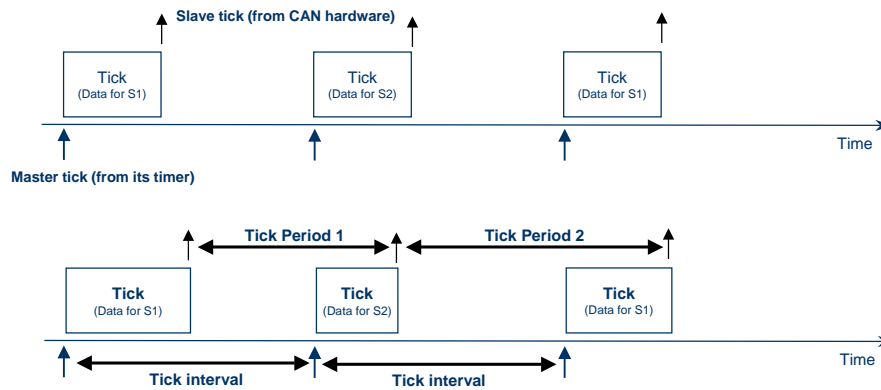


6

Embedded Systems Laboratory
<http://www.10.ac.uk/ereg/embedded>

Task jitter in S-C architecture

- In S-C network, tasks on the slaves will suffer from timing jitter if there is any variation in the time taken to send “tick” messages between the Master and the Slaves.



7

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

Task jitter in S-C architecture

- Sources of jitter:
 - Software (schedulers)
 - Hardware (CAN transmission)
- Jitter affects the performance of embedded systems.
- The aim of the present study was to reduce jitter levels

8

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

Sources of jitter (CAN-hardware)

- CAN incorporates a 'bit-stuffing' mechanism to avoid drift in the receiver's clock.
- What is bit-stuffing?

100000101111110001.....
100000**1**1011111**0**10001.....
 ↑ ↑
 Stuffed bits

9

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

Sources of jitter (CAN-hardware)

- Bit stuffing can have an impact on the message length, and hence – in a S-C design - on task jitter.
- Using 64 data bits, the worst case scenario (Nolte, 2001) is when the data looks like:

11111000011110000.....

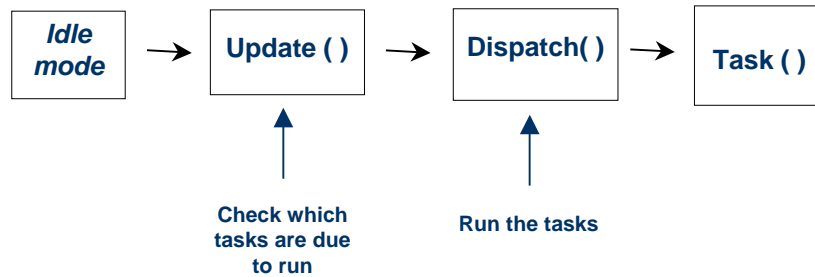
- In this case, there will be 15 stuff bits inserted by the CAN hardware.

10

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

Sources of jitter (Schedulers)

- In the Slave node, the time taken between the interrupts and the task executions will vary if there is more than one scheduled task.



11

Embedded Systems Laboratory
<http://www.10.ac.uk/ereg/embedded>

System hardware setup

- Two nodes: one Master and one Slave.
- Infineon C515C boards (8051 architecture).
- 10 MHz crystal oscillator.
- The measurement tools:
 - NI DAQ 'NI PCI-6035E'
 - LabVIEW 6.1 software
- 1000 successive samples measured

12

Embedded Systems Laboratory
<http://www.10.ac.uk/ereg/embedded>

S-C CAN scheduler setup

- Standard S-C scheduler (see: Pont, 2001)
- 333.33 kbaud
- 8 data bytes (one byte for Slave ID)
- 6 ms tick interval
- 1 task scheduled in the Master
- 1 – 10 tasks scheduled in the slave

13

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

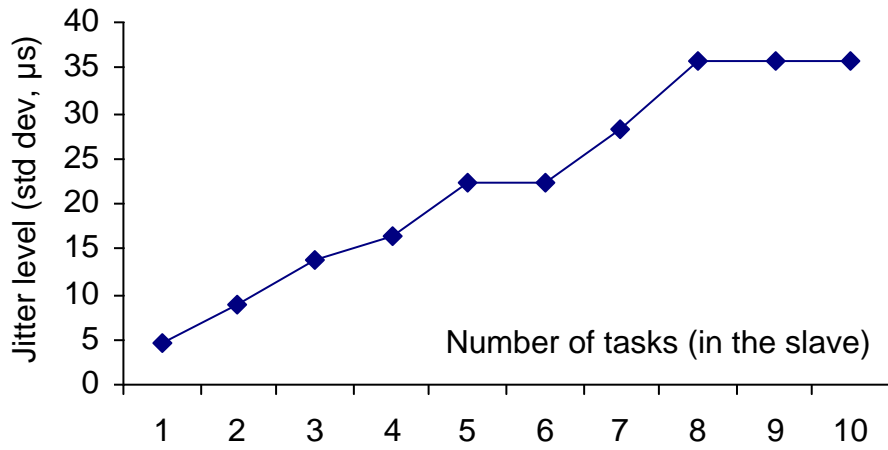
Data selection

- In this study, the jitter measured was due to the hardware bit-stuffing as well as the schedulers.
- The best case, worst case and random data were considered:
 - Best case data is 1010101010.....
 - Worst case data is 11111000011110000.....
 - Random is pseudo-random values (purely random)

14

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

Jitter from the original S-C scheduler (random)

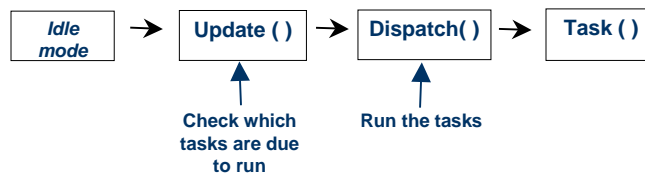


15

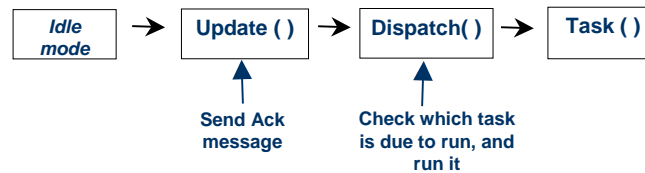
Embedded Systems Laboratory
<http://www.10.ac.uk/ereg/embedded>

Design of reduced-jitter scheduler

In the original scheduler



In the modified scheduler



16

Embedded Systems Laboratory
<http://www.10.ac.uk/ereg/embedded>

Design of reduced-jitter S-C scheduler

- In the modified scheduler:
 - Check activities in the dispatcher
 - 'Update' has fixed duration

An example:

```
void SCH_Update(void)
{
    // Note that an interrupt has occurred
    Tick_count_G++;

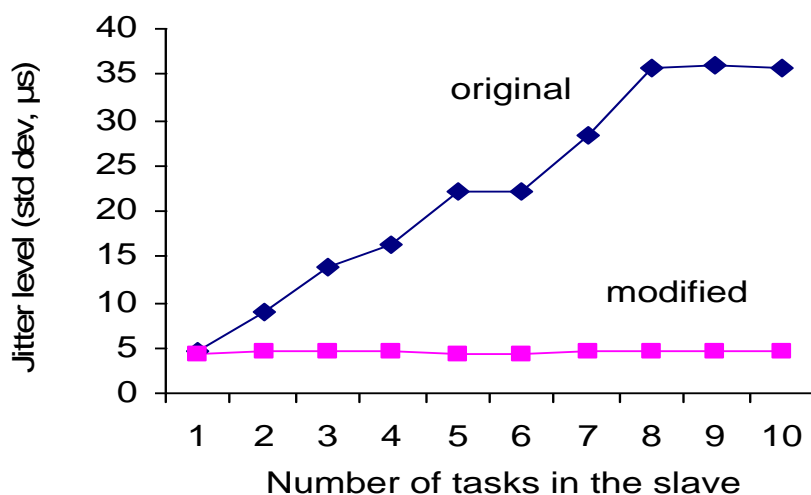
    // Clear interrupt flag (if necessary)

    // Send Ack message to the Master
}
```

17

Embedded Systems Laboratory
<http://www.16.ac.uk/ereg/embedded>

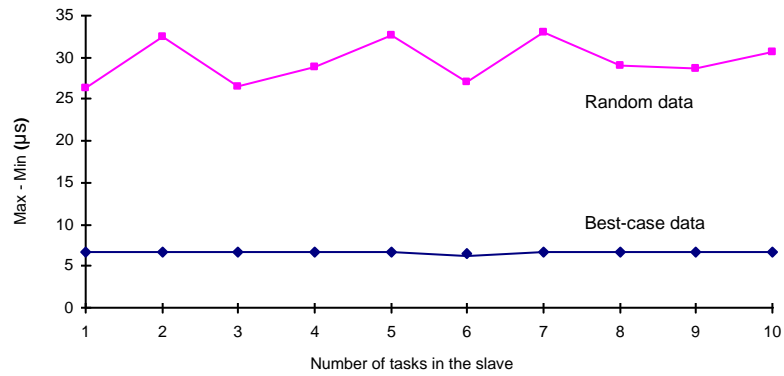
Jitter from the modified S-C scheduler (random)



18

Embedded Systems Laboratory
<http://www.16.ac.uk/ereg/embedded>

Best and random data (modified scheduler)



19

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

Conclusions

- S-C scheduler was briefly described.
- Jitter from the original S-C design was demonstrated.
- A reduced-jitter scheduler was described.
- Modified jitter levels were demonstrated.

- Still some “room for improvement” which we are currently considering.

20

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

References

Cottet, F. and David, L. (1999) "A solution to the time jitter removal in deadline based scheduling of real-time applications", 5th IEEE Real-Time Technology and Applications Symposium - WIP, Vancouver, Canada, pp. 33-38

Nolte, T.; Hansson, H.; Norström, C. and Punnekkat, S. (2001) "Using Bit-stuffing Distributions in CAN Analysis", IEEE/IEE Real-Time Embedded Systems Workshop (Satellite of the IEEE Real-Time Systems Symposium) London

Pont, M.J. (2001) "Patterns for time-triggered embedded systems: Building reliable applications with the 8051 family of microcontrollers", ACM Press / Addison-Wesley. ISBN: 0-201-331381.