ThreadX? Performance Analysis

Author : William Lamie, Express Logic, Inc

Real-Time Computing and Communications Lab.
Hanyang University
Woojong Kim

2006.11.20

Contents

Overview

- ThreadX?
  - Express Logic`s advanced RTOS designed specifically for embedded applications.
- ThreadX`s Feature
  - Small Memory Footprint
  - Fast Context Switch
  - Fast Interrupt Response
  - Preemption-Threshold Technology
  - Picokernel Design
  - Supports All Leading 32/64-bit Processors
  - Broad Tools Support
  - Easy To Use
  - Full Source Code
  - Royalty-Free

Goals and Major Contributions

- Goals
  - Introduce How to Performance analysis on ThreadX.
  - Benefit of Performanace analysis
    - Debug application deadline problem.
    - Useful in pre-production system tuning and verification.

Non-Intrusive Performance Analysis

- Feature
  - Requires no additional code on the target system
  - Not effect on the timming of the target system
- Logic Analyzer Analysis
  - Variable
    - _tx_thread_current_ptr
      - When a thread is executing, the address of it`s control block
      - If NULL, no thread is running and the system is idle.
    - _tx_thread_system_state
      - If non-zero, interrupt processig is in effect.
      - If zero, thread or idle system processing.

5

Non-Intrusive Performance Analysis

- Schedule Count
  - Variable
    - tx_run_count
      - Each time a thread is sheduled a counter is incremented in its control bolck.
      - Useful in detecting excessive preemption conditions.

6

Intrusive Performance Analysis

- Feature
  - Requires additional code(ex: assembly) on the target system
  - Overhead induced is probably on the order of 5% per context switch or interrupt.

File	Function
TX_TS.S	_tx_thread_schedule
TX_TSR.S	_tx_thread_system_return
TX_TCS.S	_tx_thread_context_save
TX_TCR.S	_tx_thread_context_restore

Summary

- Non-intrusive approaches enable developers to analyze system performance of their actual production software running at full speed.

RTOS Real-Time Performance vs. Ease of Use : Assessing Performance Needs of an Application vs. Other Considerations

Author : John.A Carbone, VP, Marketing

Real-Time Computing and Communications Lab.
Hanyang University
Woojong Kim

2006.11.20

Contents

Overview

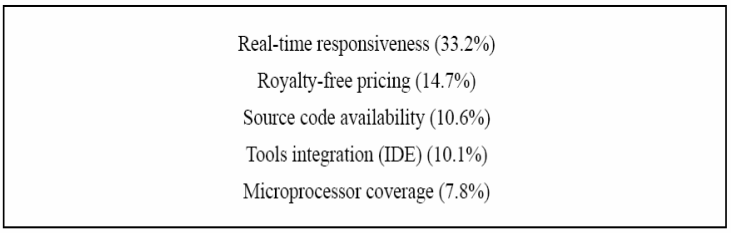


Figure-1. Evans Data Corporation's December, 2002 Survey revealed the Top 5 RTOS Features most valued by developers:

- Given that "real-time performance" is often cited by developers as paramount in their list of criteria for selection of an RTOS.
- Developers are encouraged to choose an RTOS with low overhead or risk missing the performance requirements of their application.

11

What performance is needed?

- Before evaluating an RTOS, it`s important to determine what performance the application actually requires.
- Developers demand the highest level of RTOS performance.
- Once response requirements are met, faster response is of little additional value. (ex : keyboard)
- Knowing what is sufficient for an application will enable the developer to avoid paying more for an RTOS.

12

Developers speak

- What interrupt response and context switch requirements does your application need?

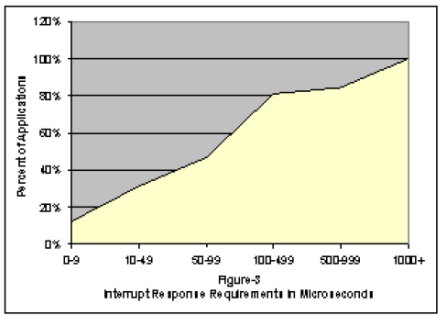


Figure-3. Only 30% of real-time applications require response faster than 50 microseconds, and more than half of all real-time applications can be satisfied with response of 100 microseconds

13

What`s really important?

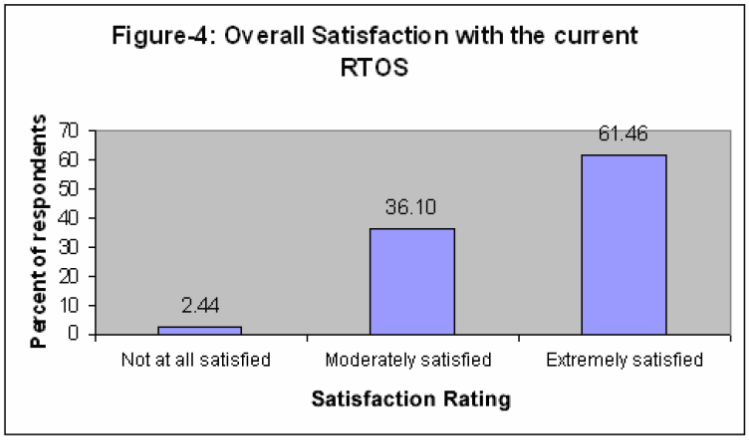


Figure-4. Satisfaction rating given to commercial real-time operating systems used by respondents (Gartner Dataquest, 2004)

14

What`s really important?

	Importance	Satisfaction	Gap
Run Time Royalties	6.7	6.6	-0.1
Features	7.6	7.5	-0.1
Processor Support	7.6	7.6	0.0
Development Tools Support	7.9	7.3	-0.6
Customer Support	7.8	7.1	-0.7

Customer Importance/Satisfaction Scores Gartner Dataquest,2004)

- Rather than look for an RTOS with 1 microsecond interrupt response and 50 nanosecond context switching, it might be more useful to fine one that is easy to learn and use.

15

Ease of use

- Ease of use can have very significant benefits for developers and can help to bring products to market more quickly, with lower development cost, resulting in greater profitability.

- RTOS characteristics for ease of use
  - Simple RTOS system services
  - Good documentation
  - Intuitive naming convention
    - Ex: rtos_message_send(GOOD), xyxConstructBridge_435(Bad)
  - Availability
  - Responsive technical support

16

Conclusion

- Once you understand your application`s performance needs, it`s likely that many RTOSes will meet them.
- Greater ease of use just might enable you to complete development and bring a product to market more quickly, more economically, and more profitably.

Efficient Memory Protection for Embedded System

Author : John Carbone, Express Logic

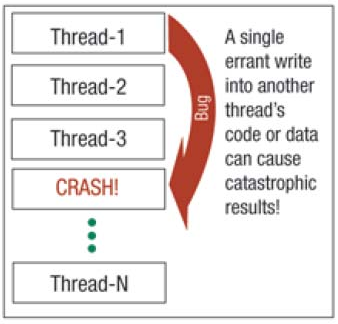
Real-Time Computing and Communications Lab.
Hanyang University
Woojong Kim

2006.11.20

Contents

Overview

- Many techniques exist for protecting from the errant actions of other programs on the same system.
  - In desktops and servers, this is effectively done through memory management hardware and multi-user OS that gives each user an independent virtual machine environment.



Overview

- Desktop and server approaches require more resources and may be impractical for an embedded system.

- Embedded Memory Protection(EMP) is specifically designed to provide memory protection for embedded system.
  - EMP provides Memory protection without the typical overhead or complexity of desktop approaches.

21

Memory Protection Background

- In desktop, memory protection has involved virtual memory and a process model programming environment.
  - Each process has an independent virtual address range mapped to physical memory.
  - Overhead to do the complex bookkeeping of mapped page table
  - Requires additional memory for page table.
  - Performance penalties for accesses outside of cached table entries.
  - Performance penalties for copying data during transfers between processes.

- Some embedded operation systems have employed process model memory protection schemes.
  - Such systems have ample memory and processor resources
    - ex) military, aerospace and high-end telecommunications applications.

22

Memory Protection Background

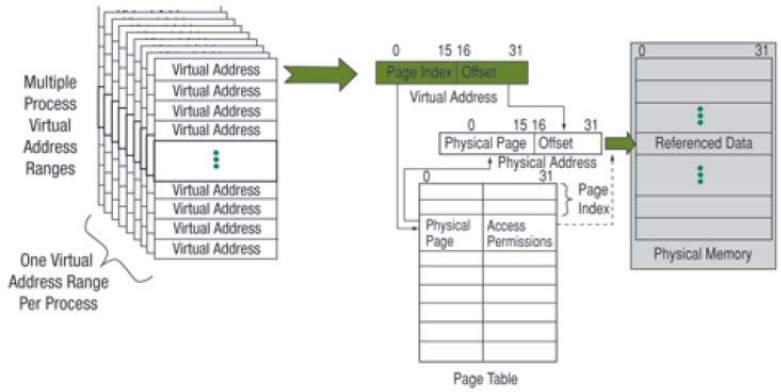


Figure 2 In a virtual memory system, addresses in each user's virtual address space are translated to reference code and data in physical memory.

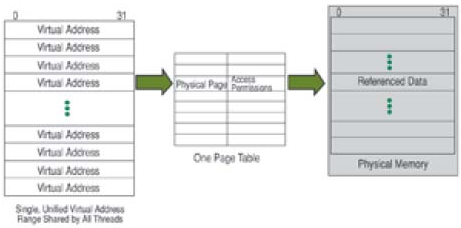
23

The Cost of Virtual Memory

- While process model virtual memory system deliver added security for embedded applications,
  - increase overhead in memory consumption and processor performance
  - programming complexity is greater for the developer
- As a result, most embedded applications have opted to
  - live without the benefits of MMU support in the OS
  - suffer the additional costs associated with use of an OS offering a process model approach

24

A New Approach



- All kernel and application memory pages are mapped directly to physical memory pages in a one-to-one.
- In EMP, a program`s virtual address pages are translated into physical address page via pointers in the MMU, with access permissions for each page.

A New Approach

- This can be implemented with a few simple RTOS services that set MMU protection registers to permit or prevent access to designated data and to activate or deactivate those protections.

How This Differs from Other Protection Approahes

- Protection should only be used when needed.
  - Protection is left off by default.
- It reduces the need for additional memory.
  - Very little code is added to provide memory protection services
- Memory protection using EMP requires only a handful of API calls.
  - Programming simplicity and operational efficiency are retained by using a simple API and only a handful of service call.

Conclusion

- EMP results in more efficient performance.
  - Single "logical equals physical" address space is used, there is no overhead introduced at context switch time.
  - Page translation is done completely in hardware, within the address-generation stage of the processor pipeline.
  - Messages can be sent from one thread to another without the need to copy the data, as is required when multiple virtual address ranges are used.