TRON Project Int'l Symposium '98

Recent Results of the ITRON Subproject

12th Mar. 1998

Hiroaki Takada
Toyohashi Univ. of Technology
hiro@ertl.ics.tut.ac.jp

Kiichiro Tamaru
TOSHIBA Corporation

§ TRON is an abbreviation of “The Real-time Operating system Nucleus.”
§ ITRON is an abbreviation of “Industrial TRON.”

ITRON Subproject *in the 2nd Stage*

1st stage: real-time kernel specification

2nd stage: *related standards* for embedded systems

- software components (*software IP*)
  - satisfying the preconditions for promoting the development and circulation of software components
  - standard API for software components
- development environments
  - interface between real-time kernel and development environments
    eg) language binding, debugging support
- application-specific standards
  - satisfying application-specific requirements

Standardization Activities

- ITRON Hard Real-Time Support Study Group
  (Nov. 1996 – Mar. 1998)
  - Kernel Specification WG
  - Application Design Guidelines WG
- Embedded TCP/IP Technical Committee
  (Apr. 1997 –)
- RTOS Automotive Application Technical Committee
  (Jun. 1997 – Mar. 1998)
- Java Technology on ITRON-Specification OS Technical Committee
  (Nov. 1997 –)

Importance of Software Components

- Embedded systems is growing larger and more complex.

  eg) digital camera
  automotive applications

- Some hardware components can be implemented with software.

  eg) software modem
  voice compression/decompression
  JPEG, MPEG

- Development from scratch becomes more and more difficult.
- Lack of expertise is a serious problem.

Standardization *for* Software Components

(1) promoting the development, circulation, and use of software components

(2) standard API for software components in specific fields

Standard API for Software Components

- Standardization should be done for each kind of software components.

  eg) communication protocols (TCP/IP)
  file system, MPEG

→ *should be started from most important fields*

TCP/IP protocol stack

Promoting the Use of Software Components

! Loose standardization is an obstacle for the portability of software components.

⇓

- The standardization level should be raised.
  → *next generation µITRON kernel specification*

! Software components with hard real-time constraints should be supported.

eg) software modem, MPEG

⇓

- coexistence of software components with applications while satisfying their real-time constraints
- enabling use of multiple software components with their own real-time needs
  → *application design guidelines for R-T systems*

Next Generation µITRON Kernel Spec.

µITRON4.0

motivations:

- improving software portability
- incorporating new kernel functions
  - *hard real-time support*

issue:

- *improving software portability* while keeping the advantage of "*loose standardization*"

  performance vs. *software portability*

approach:

- defining several *profiles*

  profile = a standard set of kernel functions for a specific range of applications
- subsetting is still acceptable (for small systems)

Standard Profile

concept:

- a set of kernel functions defined for raising software portability
  - Software components should use only the functions included in the standard profile.
  - Kernels should implement all functions included in the standard profile.

system assumptions:

- The whole software is linked to one module.
- Kernel objects (task, semaphore, etc.) are statically defined.

Standard Profile – Function Overview

still under discussions

- including almost all level S functions of µITRON3.0

incorporating from level E:

- fixed-sized memorypool
- cyclic handler

  detailed specification is revised
- system calls with timeout

  tslp_tsk, twai_sem, ...

strict standardization:

- System calls invoked from interrupt handlers should be iXXX_YYY.
- Mailbox should be implemented with linked-list.

modifications:

- ient_int and isig_tim are added.
- Some system calls are renamed.

  preq_sem → pol_sem
- Some terminologies are changed or clarified.

  suspend → *suspended*

static API:

- System calls for creating and deleting kernel objects (task, semaphore, ...) are not included.
- Kernel objects should be defined with static API.

  cre_tsk(...) ..... system call (*dynamic API*) for creating a task

  CRE_TSK(...) ... kernel configuration description (*static API*) for creating a task

exception handlings:

Exception handling architecture is defined!

- CPU exception handler
  - Exceptions are raised by the processor.
  - Handlers are invoked immediately.
  - Handlers are executed in non-task context.
- task exception routine
  - Exceptions are raised with ras_tex system call.
  - Handlers are executed in the task context.
  - Handlers are invoked when the task is scheduled in the next time.

similar to UNIX signal handler, but much lighter (and simpler) mechanism

Extended Functions

- Level E functions of µITRON3.0 which are not included in the standard profile are defined as extended functions.
  - system calls for creating, deleting, and referring kernel objects
  - messagebuffer and rendezvous
  - variable-sized memorypool
  - alarm handler

  etc.
- Hard real-time support functions are incorporated.
  - mutual exclusion with priority ceiling and priority inheritance support
  - overrun detection

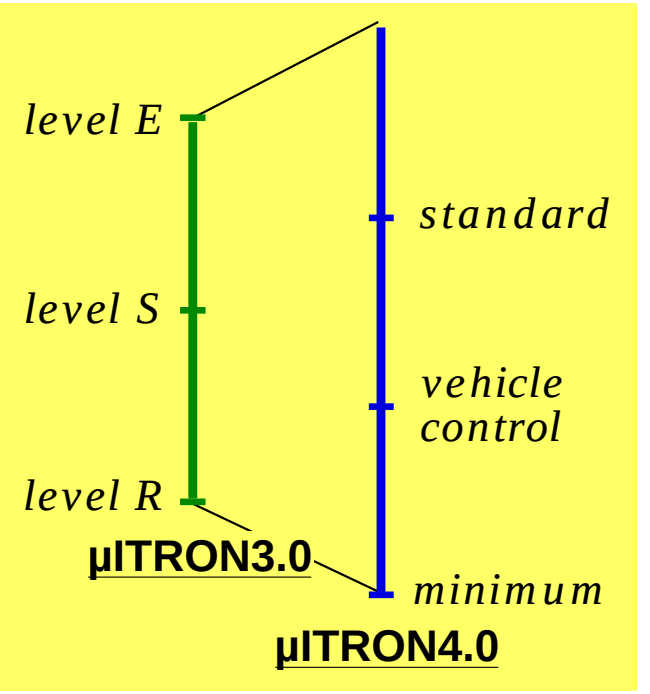
Vehicle Control Profile

- A subset definition of µITRON for vehicle control applications has been proposed by automotive engineers
- a bit smaller subset than level S of µITRON3.0

with
modified mailbox functions
+ stack sharing mechanism

↓

incorporated as another profile



Minimum Profile without Wait States

- Wait state is mandatody with the existing µITRON specifiations, and dormant state is optional.

  ↓

  should be exchanged!

  - Dormant state is useful for saving stack space.
  - All tasks can share one stack space if wait states are unnecessary to be supported.
  - Many application systems do not require wait state.

  ↓

- A profile without wait state should be defined as an *introductory specification*.

Application Design Guidelines

for Real-Time Systems

two purposes:

- guaranteeing real-time constraints of both software components and application based on real-time scheduling theories

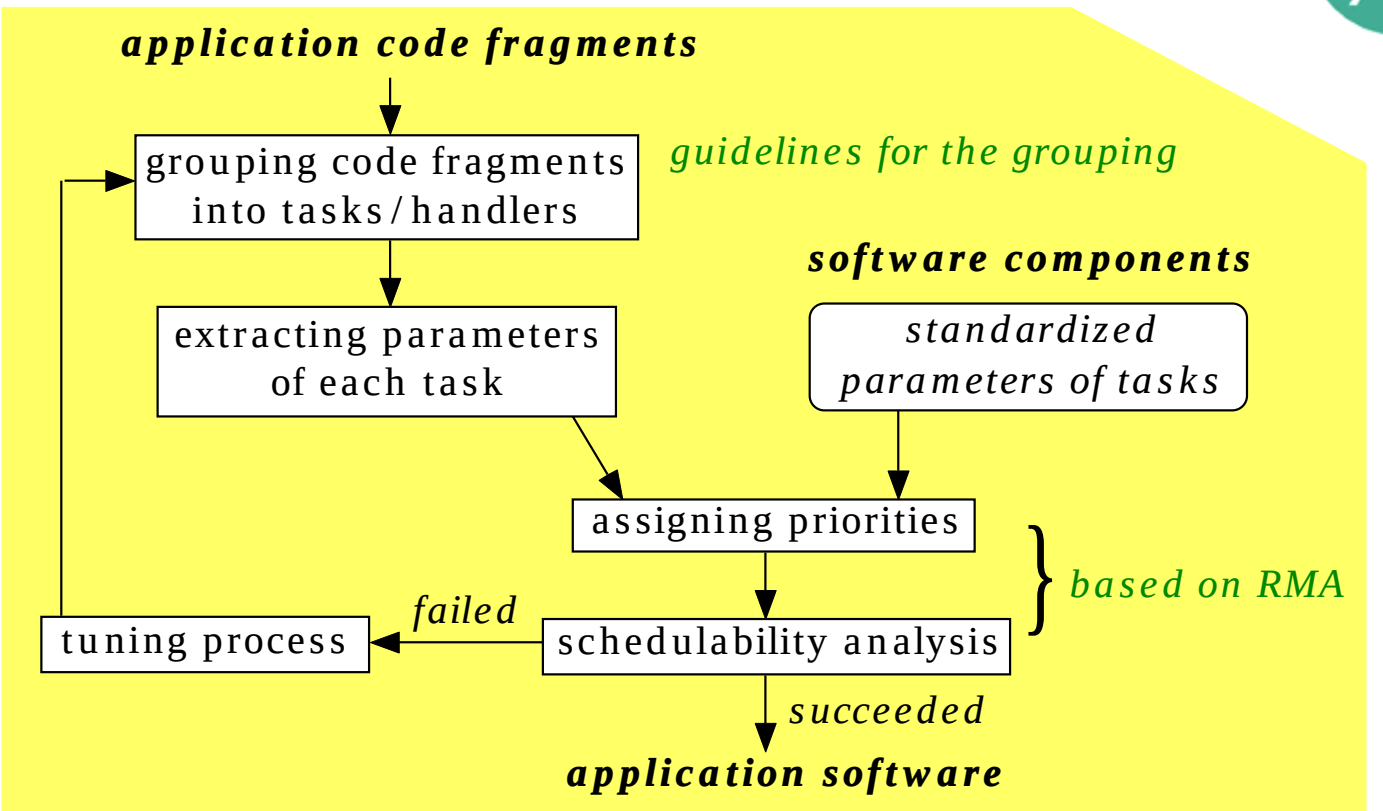
  RMA (*rate monotonic analysis*) is adopted.

- providing novice system designers a good guidelines to design real-time applications using a real-time kernel

  How to divide a system into tasks?

  How to assign priorities to tasks?

Framework of the Design Guidelines



ITRON TCP/IP API Specification

- TCP/IP protocol stack is one of the most important software components, today.
- The socket interface is *not suitable* for (*esp.* small-scale) embedded systems.
  - necessity of dynamic memory management within the protocol stack
    - → *Errors occurred within the protocol stack is not notified to the application.*
  - difference between UNIX process model and ITRON (RTOS) task model

status:

- *in final discussion stage and will be published soon*

approach:

- based on the socket interface
- The socket interface can be implemented as a library on the proposed API.

difference from the socket interface:

- TCP API and UDP API are separately defined.
- "*End point*" abstraction is adopted instead of "*socket*" abstraction. TCP end point for waiting for connection requests and TCP connection end point are handled as different objects.
- TCP APIs for reducing data copies are also defined.
- Non-blocking calls and callbacks are supported.
- The callback routine can be used for receiving UDP packets.

etc.

Future Plan

- µITRON4.0 Real-Time Kernel Specification
  planned to be published within 1998
- Real-Time Kernel Debugging Interface
  new standardization activity in 1998
  - standard interface between ITRON-specification kernel and debugging environments
- Application Design Guidelines
- Device Driver Design Guidelines
  continued to be investigated

- µITRON4.0 Specification Study Group
  open study group for non-members

Summary

- documents to be published in 1998
  - ITRON TCP/IP API Specification
  - JTRON ver.2 Specification
  - µITRON4.0 Specification
  - Application Design Guidelines
- standardization activities in 1998
  - µITRON4.0 Specification Study Group
  - Embedded TCP/IP Technical Committee
  - Java Technology on ITRON-Specification OS Technical Committee
  - RTOS Automotive Application Study Group