The ITRON Project : Overview and Recent Results

28th Oct. 1998

Hiroaki Takada

Dept. of Information and Computer Sciences
Toyohashi Univ. of Technology

Yukikazu Nakamoto

NEC Corporation

Kiichiro Tamaru

TOSHIBA Corporation

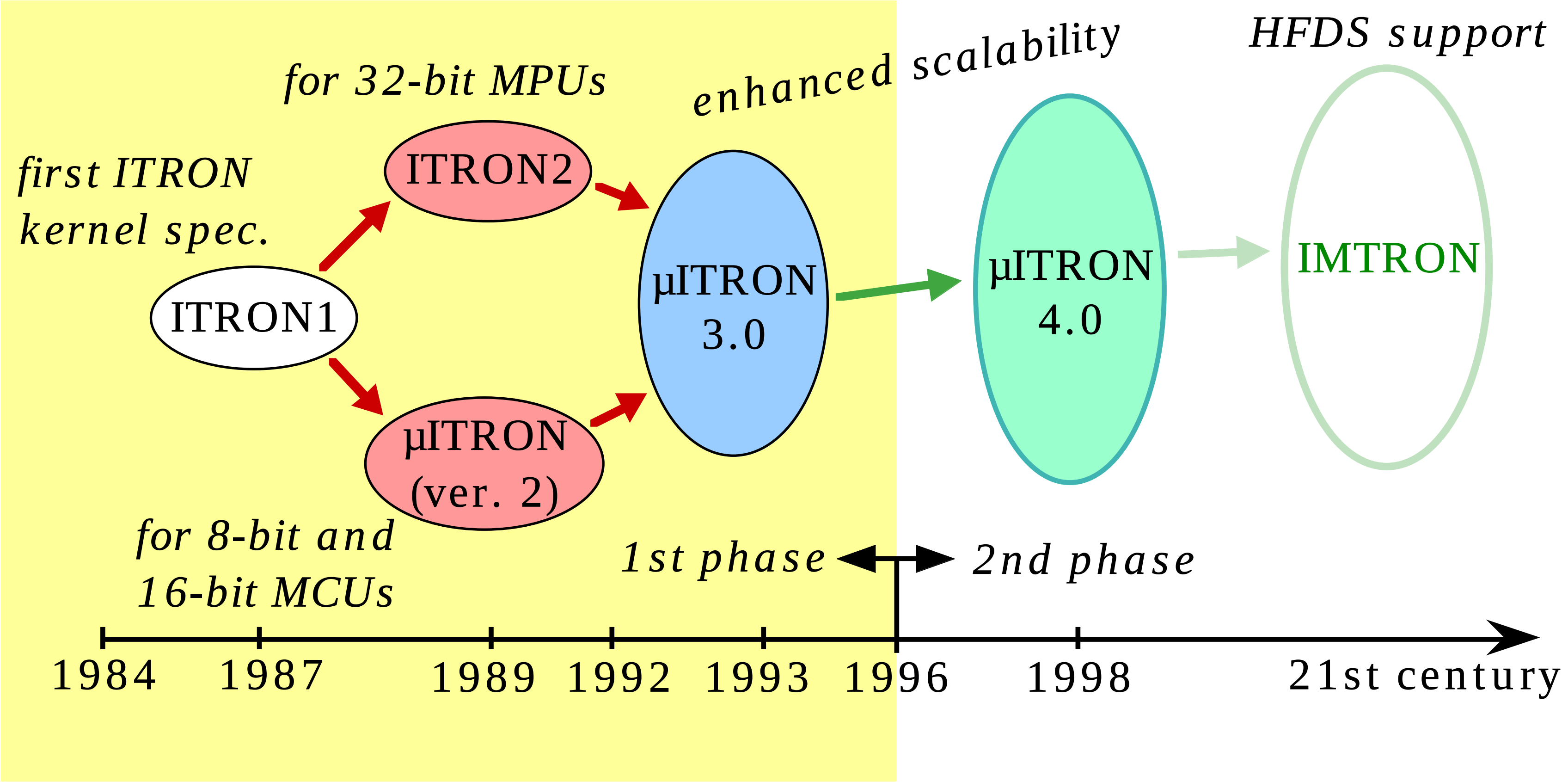
ITRON Project

- one of the subprojects of the TRON Project
- a project to standardize real-time operating system and related specifications for embedded systems (*esp*. small-scale embedded systems)
- a joint project of industry and academia (*non-government !*)

  core members:
  Fujitsu, Hitachi, Mitsubishi Electric,
  NEC, Toshiba, Oki Electric Industry,
  Univ. of Tokyo, Toyohashi Univ. of Technology
- open specification policy
- *2nd phase* started recently
  - 1st phase (1984 –)
  - 2nd phase (1996 –)

ITRON Project – *1st Phase*

- focused on *real-time kernel* specifications
  - ← Only kernel functions are necessary for many small-scale embedded systems.



Requirements on Standard RTOS Specification

- deriving maximum performance from hardware

  reducing the cost of final products

- improving software productivity

  easy training of software engineers

  facilitating the reuse of software components

- applicable to various scales and types of processors

  scalability *8-bit to 32-bit MCUs/MPUs*

- truly open standard

The ITRON specifications have been designed to meet these requirements.

Design Principles of the ITRON Specifications

- design concept: *loose standardization*

  maximum performance cannot be obtained with strict standardization

- design principles
  - allow for adaptation to hardware, avoiding excessive hardware virtualization
  - allow for adaptation to the application
  - emphasize software engineer training ease
  - organize specification series and divide into levels
  - provide a wealth of functions

Functions Supported in µITRON3.0 Specification

- task management
- task-dependent synchronization
- basic synchronization and communication
  (semaphore, eventflag, mailbox)
- extended synchronization and communication
  (message buffer, rendezvous)
- interrupt management
- memory pool management
- time management
- system management
- *no I/O management functions*

ref) K. Sakamura Ed., "*µITRON3.0: An Open and Portable Real-Time Operating System for Embedded Systems*", IEEE CS Press, 1997.

Implementation Status

! *We do not know how many kernels are implemented based on the ITRON specifications.*

- about 45 registered implementations for about 35 processors
- several non-registered commercial implementations

→ *implemented for almost all major processors*

8-bit to 32-bit MCUs/MPUs

- many in-house implementations
- some freely distributed implementations

recently announced product

eCos : a free real-time kernel conformant to µITRON specification by Cygus Solutions

Implementation Examples

- Two µITRON-specification kernels implemented for a 16-bit MCU

OS type	Single-chip	General-purpose
No. of system calls	Task part: 29 Non-task part: 15	Task part: 36 Non-task part: 27
Scheduling	Fixed priority 1 task per priority	Variable priority
System call interface	Subroutine call	Software interrupt
Exception management	None	Exit exception, CPU exception
Wakeup request count	Max. 15	Max. 255
Semaphore count	Max. 255	Max. 65,535
System timer	32-bit	48-bit
Program size	0.6 – 4.4 KB	1.9 – 5.3 KB
Typical RAM use*	200 Bytes	640 Bytes
Task switching time**	17µS	32.5µS
Max. interrupt masking time**	9µS	9.5µS

* OS work area and various stack areas in the following configuration
tasks: 10, semaphores: 2, eventflags: 2, mailboxes: 2, external interrupts: 2 levels

Typical ITRON-specification Kernel Applications

Audio/Visual Equipment, Home Appliance

TVs, VCRs, digital cameras, settop box, audio components, microwave ovens, rice cookers, air-conditioners, washing machines, ...

Personal Information Appliance, Entertainment/Education

PDAs (Personal Digital Assistants), personal organizers, car navigation systems, game gear, electronic musical instruments

PC Peripheral, Office Equipment

printers, scanners, disk drives, CD-ROM drives, copiers, FAX, word processors, ...

Communication Equipment

answer phones, ISDN telephones, cellular phones, PCS terminals, ATM switches, broadcasting equipment, wireless systems, satellites, ...

Transportation, Industrial Control, and Others

automobiles, plant control, industrial robots, elevators, vending machines, medical equipment, ...

Application Status

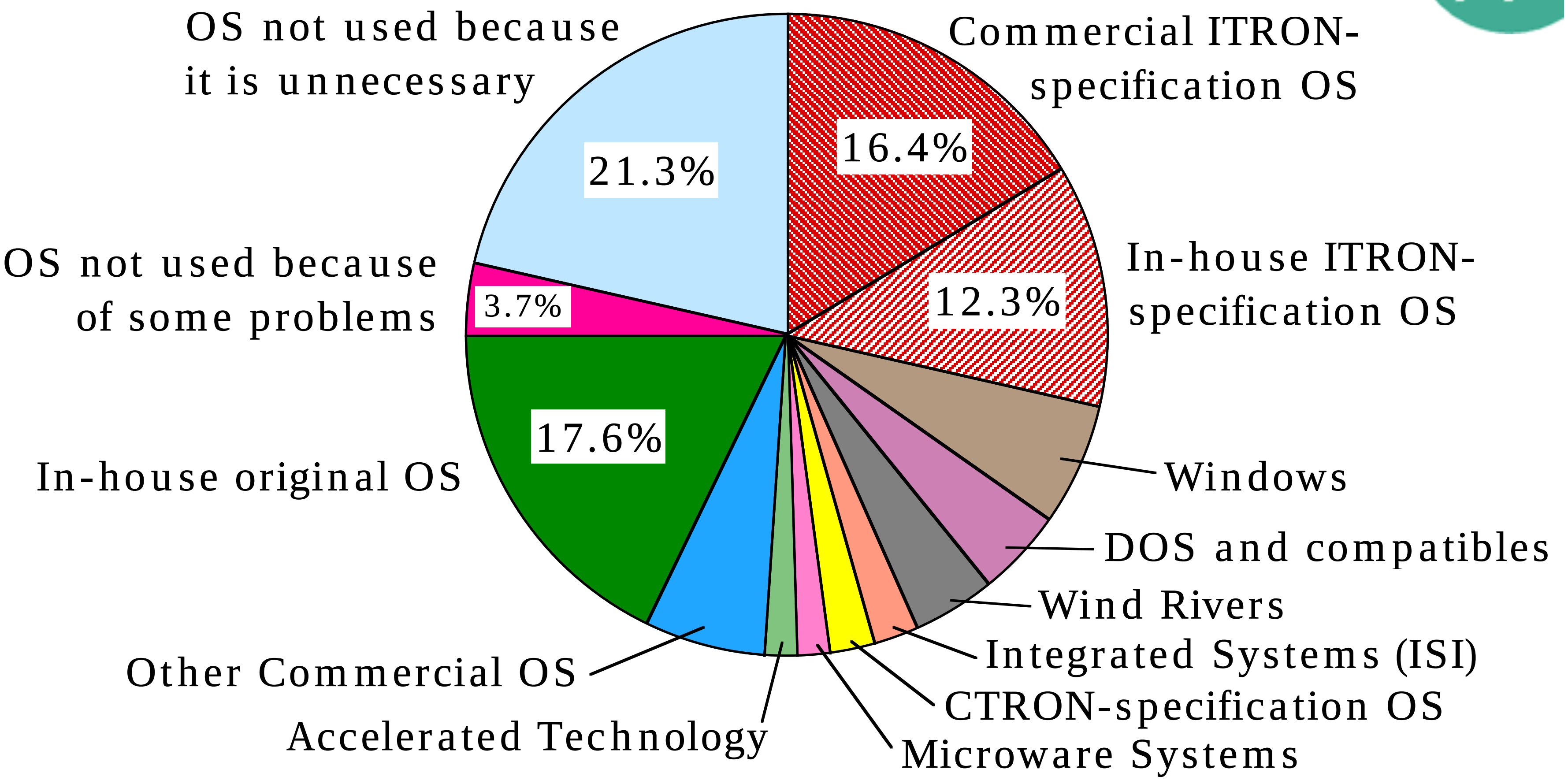
- widely used for various application fields
- de-facto real-time kernel standard in Japan

Application Examples

- two application examples to small-scale systems

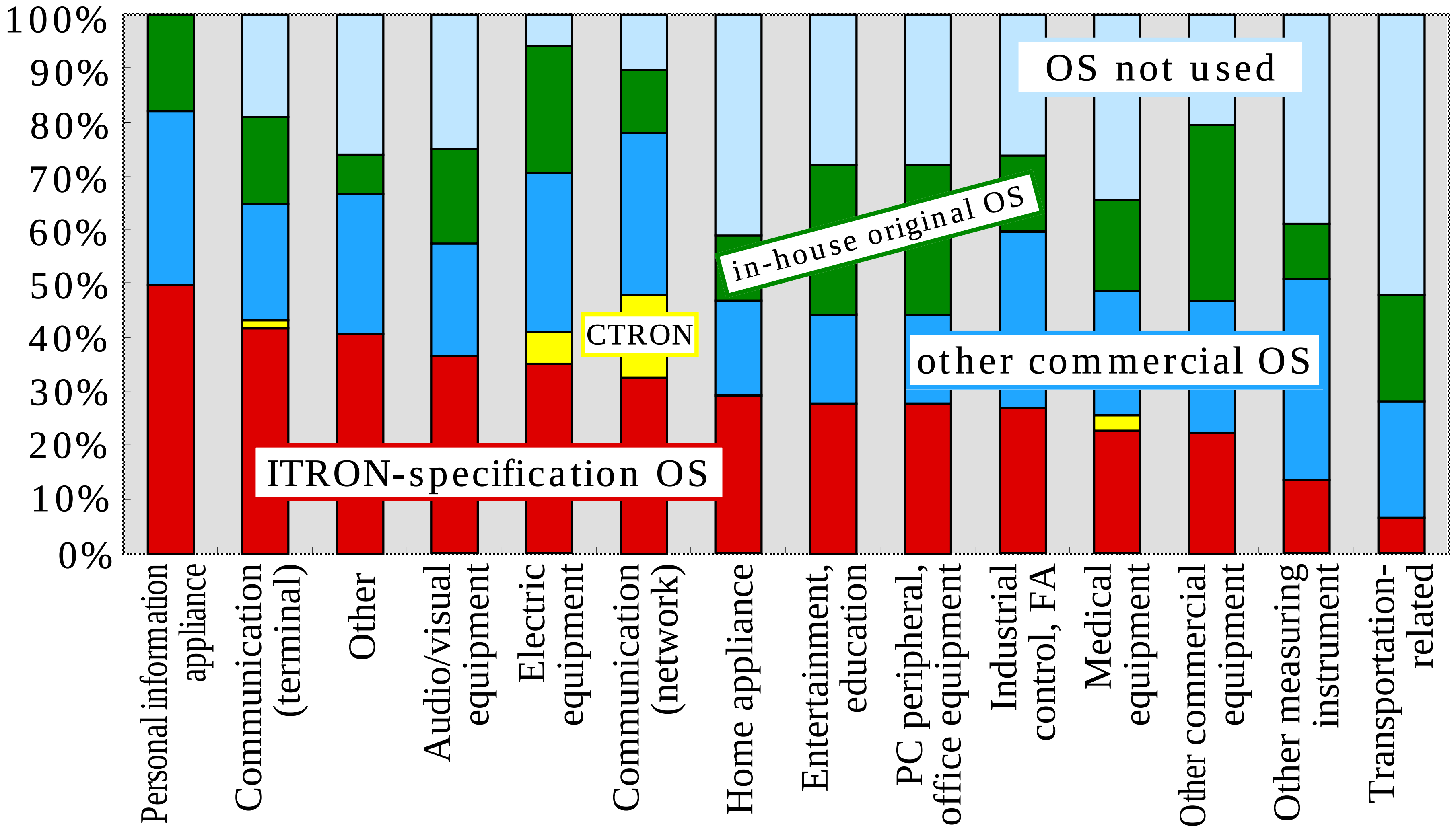
Application	FAX machine	CD player
MCU Type	16-bit	8-bit
RAM size	2 KB	512 Bytes
ROM size	32 KB	32 KB
Used Memory RAM	1346 Bytes	384 Bytes
ROM	28.8 KB	17.8 KB
No. of Tasks	6	9
No. of Interrupt Handlers	6	6
No. of Used System Calls	12	7
Kernel Size RAM (ratio)	250 Bytes (19%)	146 Bytes (38%)
ROM (ratio)	2.5 KB (8.7%)	2.3 KB (13%)

Real-Time OS Used in Embedded Systems

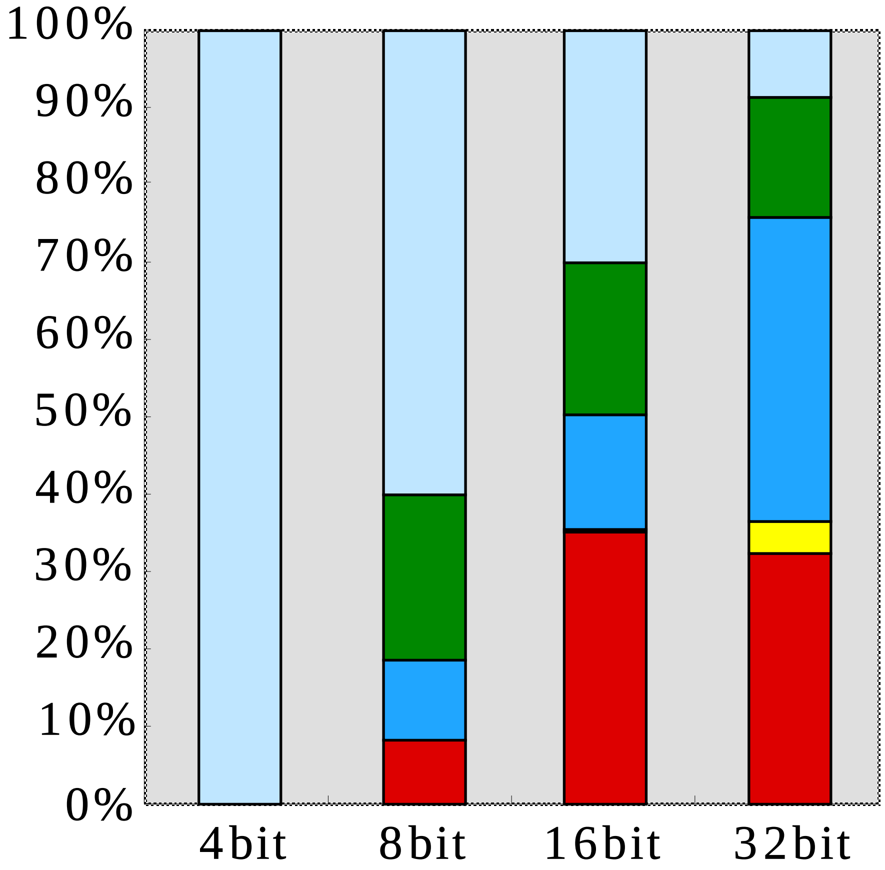


Real-Time OS used in Embedded Systems

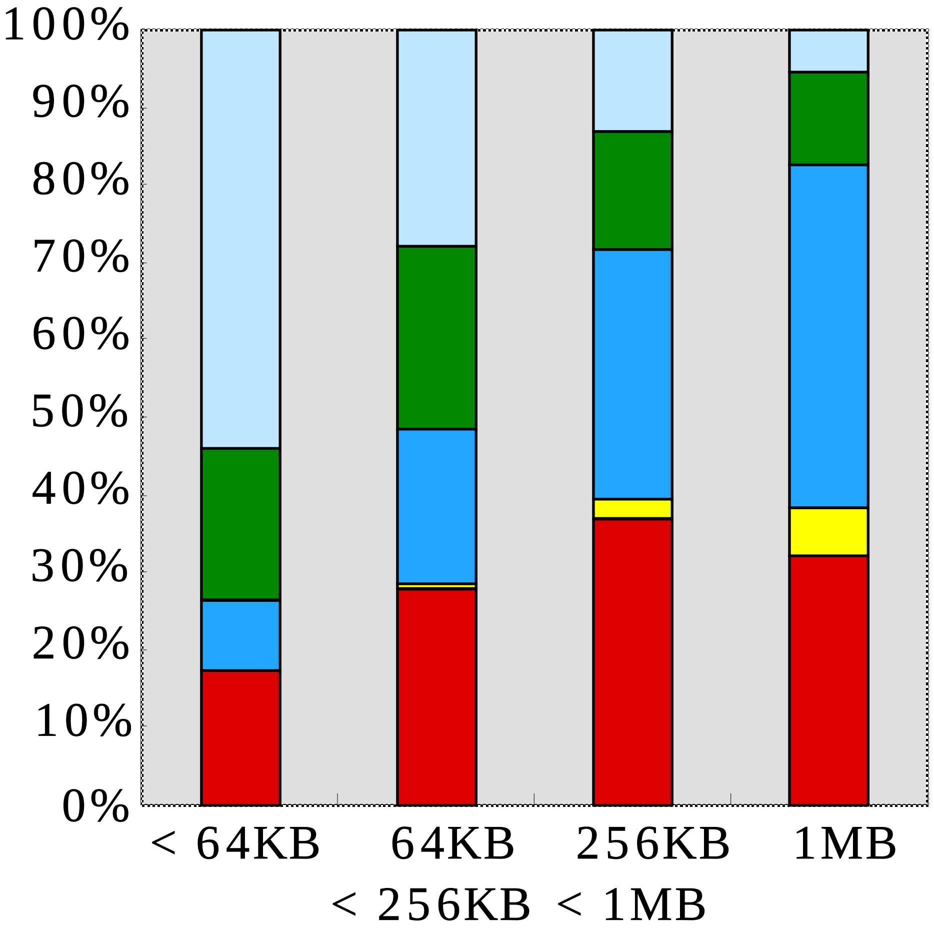
(TRON Association Survey, 1997–1998, in Japan)



Real-Time OS Usage vs. Application Fields



OS Usage vs. CPU Type



OS Usage vs. Program Size

(TRON Association Survey, 1997-1998, Japan)

ITRON Project – *2nd Phase*

- broaden the scope of the standardization to related aspects listed below
- 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

→ *several standardization activities are in progress*

Importance of Software Components

- Embedded systems is growing larger and more complex.

  eg) digital camera
  automotive applications

- Some hardware components can now be implemented with software.

  eg) software modem
  voice compression/decompression
  JPEG, MPEG

↓

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

Standardization *for* Software Components

(1) satisfying the preconditions for the circulation and use of software components

(2) standard interface for software components in specific fields

Standard Interface for Software Components

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

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

- *started from most important fields*
  - ITRON TCP/IP API Specification
  - JTRON Specification (Java on ITRON)

Preconditions for the Circulation and Use

! *"Loose standardization"* policy 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 constraints

→ *application design guidelines for R-T systems*

µITRON4.0: next generation µITRON kernel Spec.

issue:

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

standard profile:

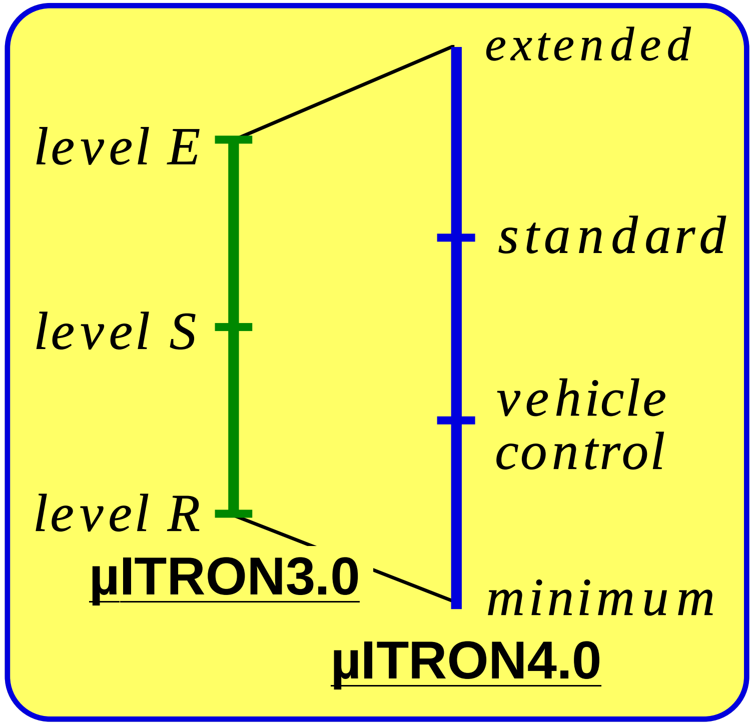
- the set of kernel functions *strictly* defined for raising software portability

extended functions:

- optional features

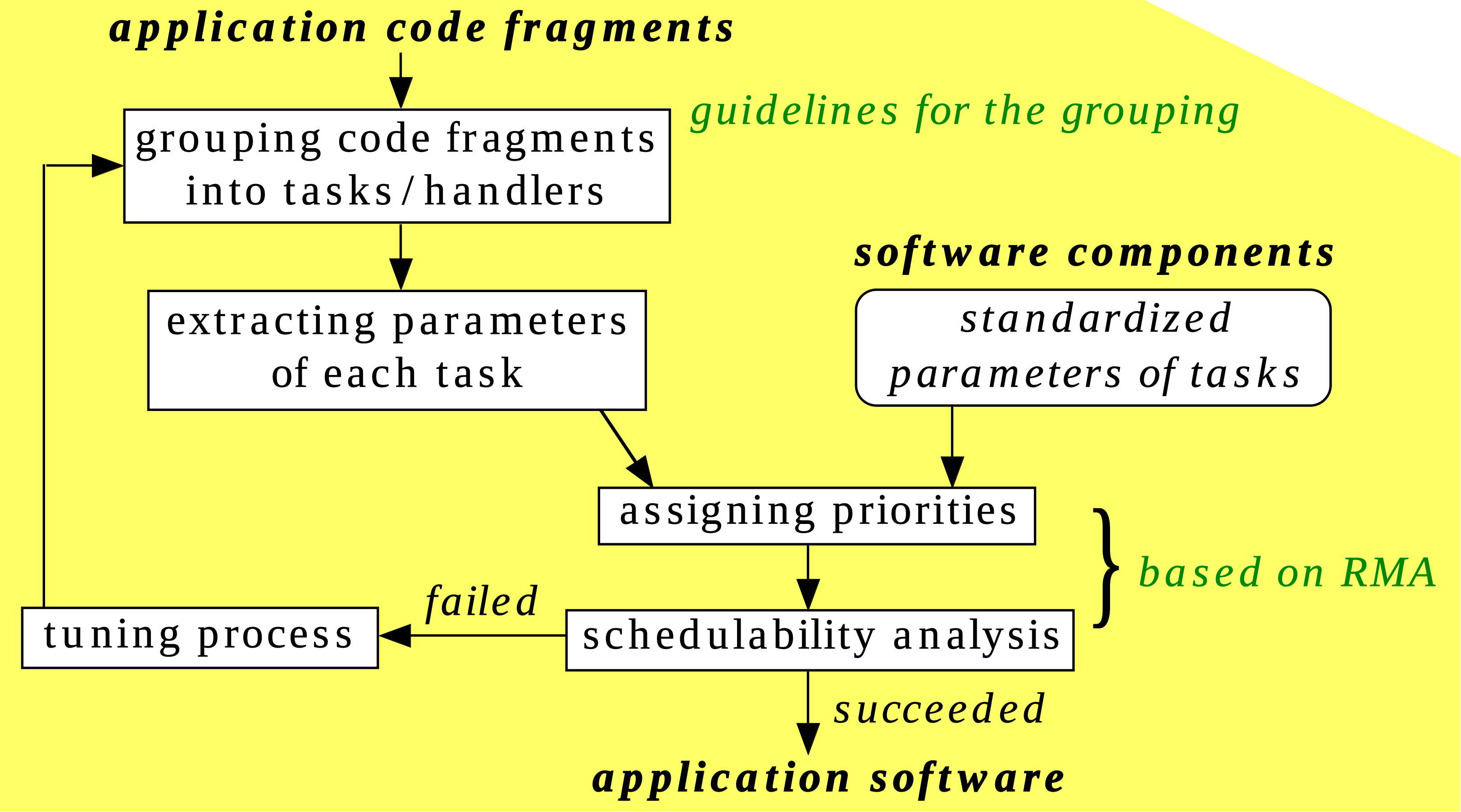
subsetting:

- vehicle control profile
- real-time kernel specification without wait state



Application Design Guidelines

- a framework to satisfy the real-time constraints of software components


application code fragments
grouping code fragments into tasks/handlers
guidelines for the grouping
extracting parameters of each task
software components
standardized parameters of tasks
assigning priorities
based on RMA
failed
tuning process
schedulability analysis
succeeded
application software

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 (*or* real-time kernel) task model
- Standard TCP/IP API suitable for embedded system is required.

  ITRON TCP/IP API Specification

approach:

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

differences with 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 is used for receiving UDP packets.

etc.

JTRON Specification

- a practical approach for applying Java technology to embedded real-time systems
  - implementing Java runtime environment on real-time OS
  - taking the advantages of both environment

modules requiring real-time property

… *implemented on real-time OS*

downloadable module, GUI

… *implemented on Java runtime environment*

- Communication interface between real-time tasks and Java applications should be standardized.

JTRON Specification

three types of communication interfaces:

Type 1: *attach classes*

- Java applications can access real-time OS resources through attach classes.

Type 2: *shared object*

- Real-time tasks can access shared objects exported from the Java application.
- explicit locking/unlocking mechanism
- Java application must explicitly call the unshare method on the object.

Type 3: *stream interface*

- Real-time tasks and Java applications can communicate through stream interface.

Other Activities

- standardization activities targeted for automotive control applications *finished*

  ↓ *the first application-specific activity*

  Requiremens on real-time kernel in automotive control applications have been clarified.

  → *reflected to μITRON4.0*

- device driver design guidelines *current*
  - hierarchical model of DIC (device interface component)
- debugging interface of real-time kernel *future*
  - standard interface between ITRON-specification kernel and debugging tools, including software debuggers, ICE, and logic analyzer
- C++ (*incl.* EC++) language binding *future*

Summary

- μITRON real-time kernel is a de-facto industry standard in Japan.
- several 2nd phase activities

  ITRON TCP/IP API Specification
  JTRON2.0 Specification
  μITRON4.0 Real-Time Kernel Specification *etc.*

↓

- continue the effort to meet industry's needs

  ITRON Project is an open activity and is waiting for your contributions.

ITRON Home Page
http://www.itron.gr.jp/