ITRON Supporters' Meeting

Current Activities of the ITRON Project

1–Oct.–1997

Hiroaki Takada

(ITRON Technical Committee / University of Tokyo)

hiro@is.s.u-tokyo.ac.jp

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

What is the ITRON Project?

- a project to standardize real-time operating system and related specifications for embedded systems

  ◄ *one of the subproject of the TRON project*

- A series of the ITRON real-time kernel specifications have been published and are widely used.

  ► *de-facto industry standard in Japan*

- μITRON specifications are designed for small-scale embedded systems using MCUs with limited hardware resources.
- The ITRON specifications are open in that anyone is free to implement and sell products based on them.

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

↓

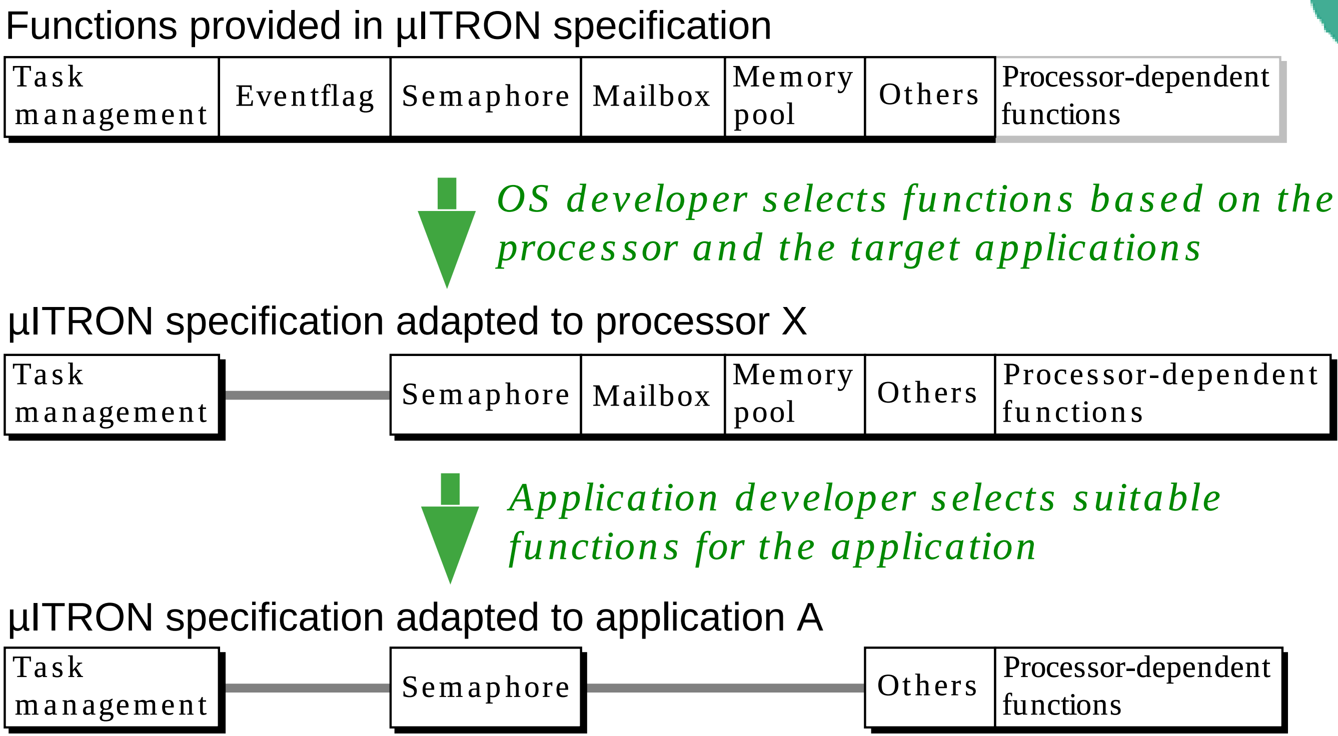
The ITRON specifications have been designed to meet these requirements.

Design Principles of the ITRON Specifications

- design concept: *loose standardization*

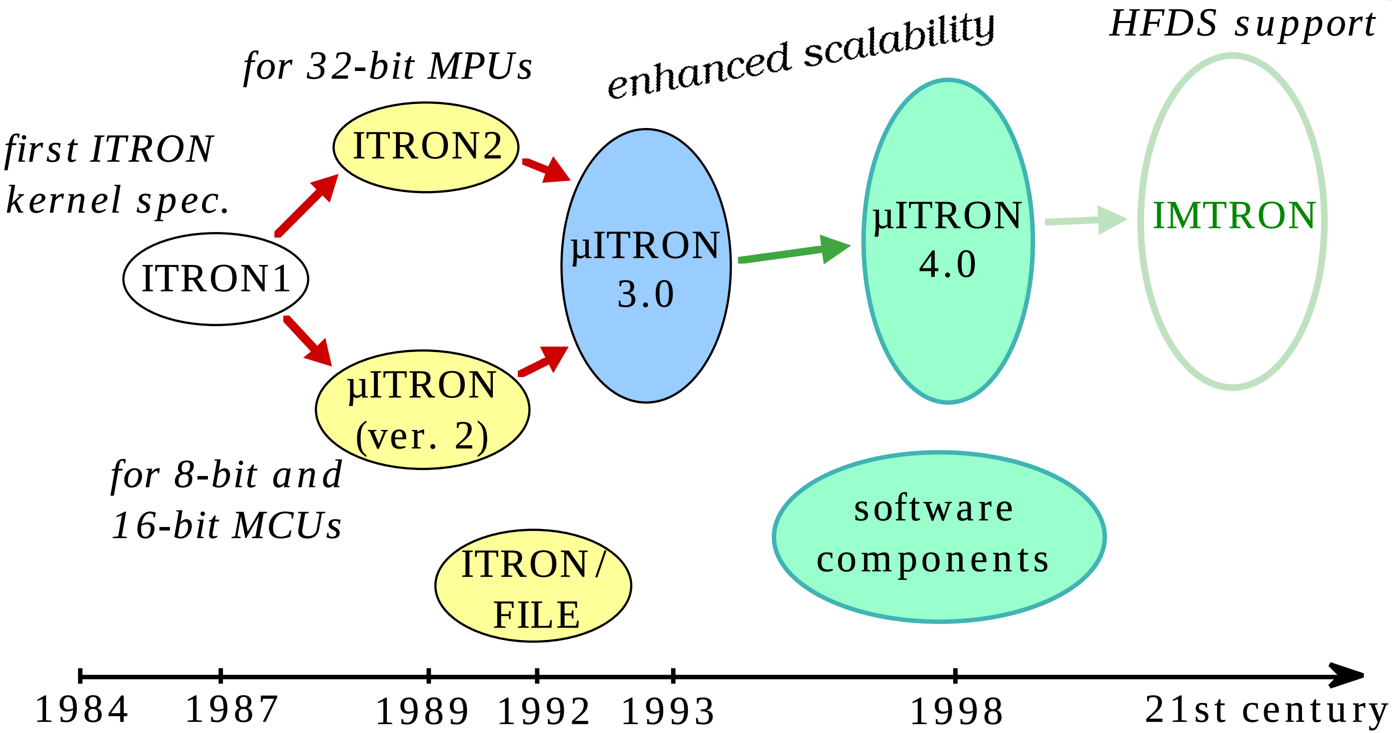
  maximum performance cannot be obtained with strict standardization

- 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



Two-Step Adaptation in µITRON Specifications

History of the ITRON Specifications



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
- (network support)

➡ *The whole specification can be downloaded from the ITRON Home Page.*

Implementation Status

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

- more than 40 registered implementations for about 30 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

  incl. an implementation by Univ. of Tokyo

  (for research and educational use)

Implementation Examples

- Two µITRON-specification kernels for an 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

** Clock 16 MHz, using on-chip memory

Application Status

- widely used for various application areas
- most popular RTOS specification in Japan

Application Examples

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%)

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

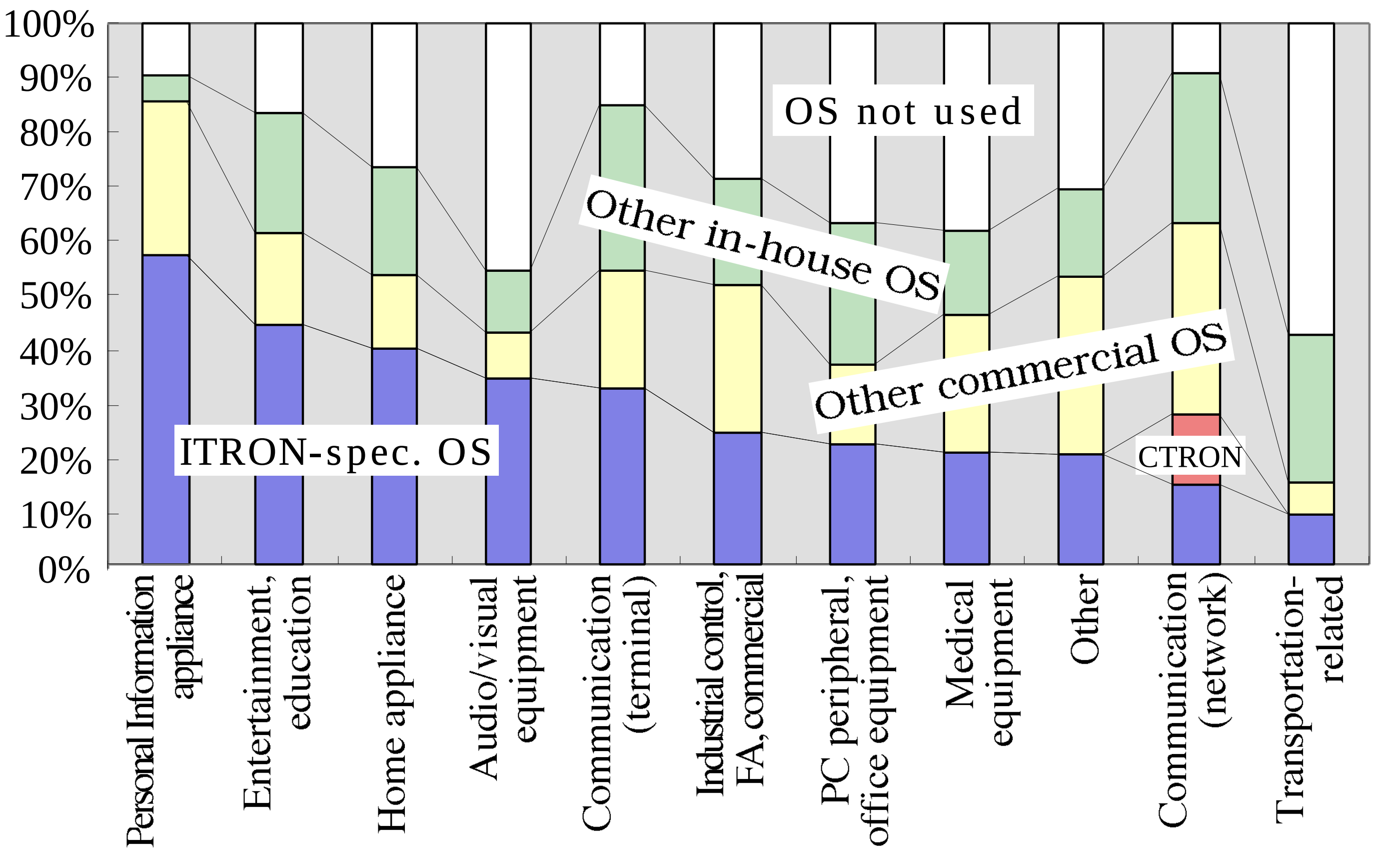
printers, scanners, disk drives, CD-ROM drives, copiers, FAX, word processors, ...

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, ...



RTOS used in Embedded Systems

(TRON Association Survey, 1996–1997, in Japan)

ITRON Project – *2nd Stage*

1st stage: real-time kernel specification

2nd stage: *related* standards for embedded systems

- software components (software IP)
  - infrastructure for the use of software components
  - standard API for software components
- development tools
  - interface between kernel and development tools

    eg) language binding, debugger support
- application-specific standards
  - satisfying application-specific requirements

➡ *several standardization activities are in progress*

Necessity of Software Components

- Embedded systems is growing large and more complex.

  eg) digital camera

- Some hardware components can be implemented with software.

  eg) software modem
  voice compression/decompression
  JPEG, MPEG

↓

- Lack of expertise is the largest problem.
- Development from scratch becomes more and more difficult.

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

  ➡ *begun from the most important field*

Promoting the Use of Software Components

- Loose standardization is an obstacle for the portability of sofware components.

↓

- The level of standardization is necessary to 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

Next Generation µITRON Kernel Spec.

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

issue: the *tradeoff* between performance and software portability

observation:

larger system … *larger* software size
larger processing power (32 or 64-bit)

→ *Software portability* is relatively important.

smaller system … *smaller* software size
smaller processing power (8 or 16-bit)

→ *Performance* is relatively important.

solution:

- defining some *profiles* (for *larger* system)

  profile = a standard set of kernel functions
  for a specific range of applications

  at first ... *standard profile*
  later ... *extended profile*,
  profile for vehicle control applications

- subsetting is still acceptable (for *smaller* system)

standard profile:

- Application systems in which the whole software is linked to one module are assumed.
- Kernel objects (task, semaphore, etc.) are statically defined.

Next Generation μITRON Kernel (cont.)

- *hard real-time support* (out of the standard profile)
  - priority inheritance
  - overrun detection and exception handling
- *standard performance metric* of the kernel for hard real-time systems
- *standard description for kernel configurations*

  `cre_tsk( ... )` ... system call (*dynamic API*) to create task

  `CRE_TSK( ... )` ... kernel configuration description (*static API*) to create task

➡ μITRON4.0 real-time kernel specification

expected to be completed in 1998

Design Guidelines for Real-Time Applications

two purposes:

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

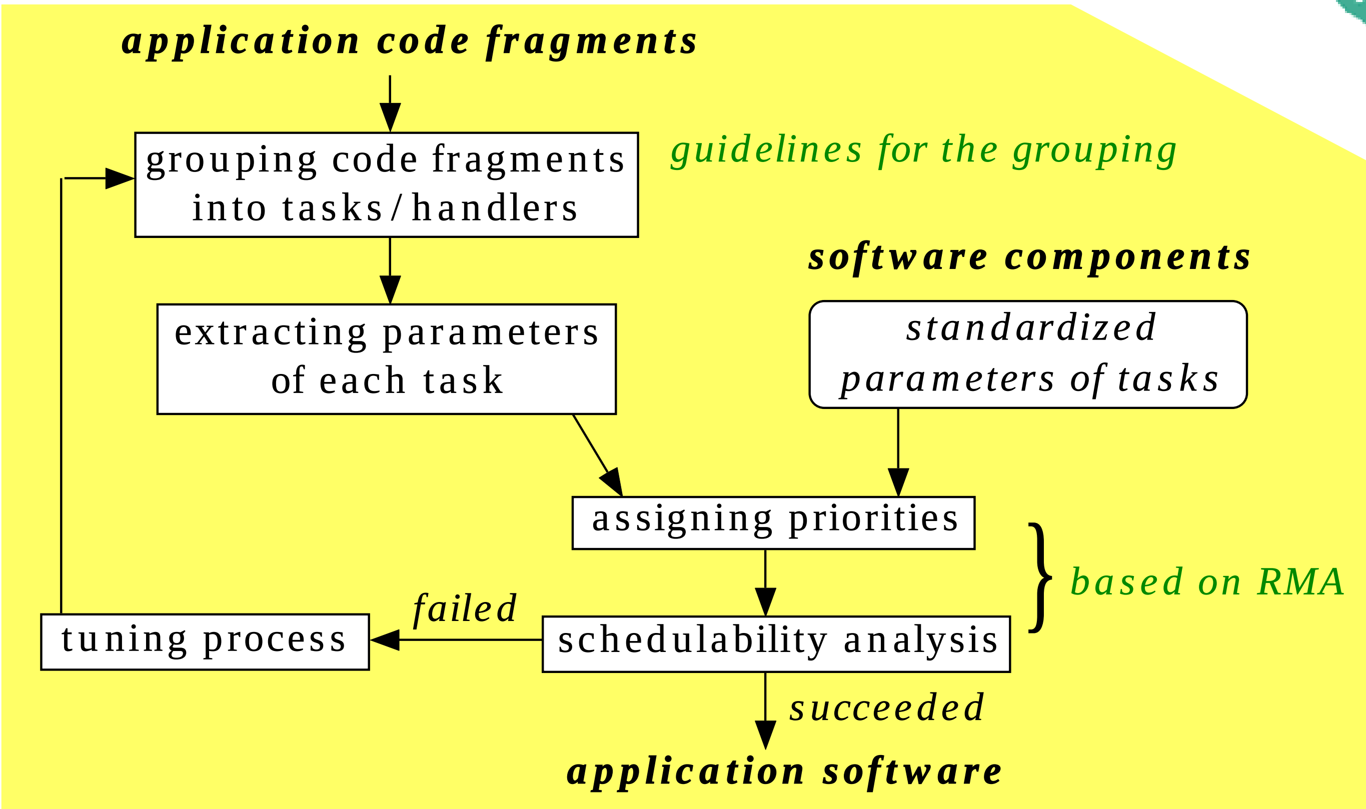
  RMA (*rate monotonic analysis*) is adopted.

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

  How to devide a system into tasks?

  How to assign priorities to tasks?

Framwork of the Design Guidelines



Standard TCP/IP API for Embedded Systems

the first standardization activity in specific fields

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

TCP/IP API under Discussion (subject to change)

- based on the *socket interface* with following *modifications*

UDP:

- Received packets are handled with callback.

TCP:

- Multiple tasks can wait connections on a port.

  "fork" is not necessary!

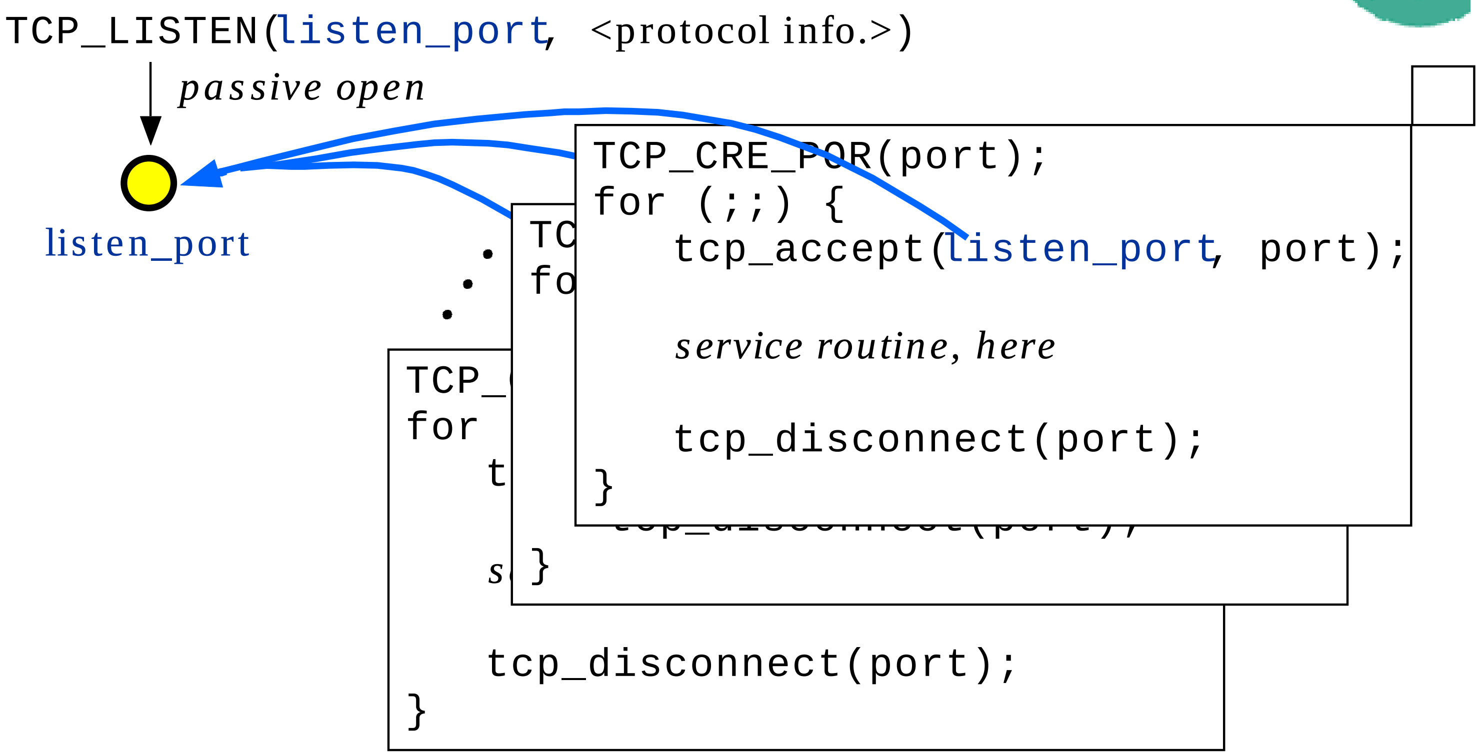
- Application program is allowed to access the window buffer directly (faster version of read/write).

  One copy may be saved.

- Callbacks are used for non-blocking calls.

  "select" can be realized with callbacks!

How to Implement Multi-threaded Server?



* API names will be harmonized with ITRON kernel specification.

Standardization for Automotive Applications

the first application-specific activity

- widely used for *car navigation systems*, already
- Some car makers/suppliers are investigating its application to *engine management systems*.
  - ➡ μITRON is still too large to vehicle control.
    - ➡ *μITRON profile for vehicle control applications*
- standard API for software components for automotive applications
  - ITRON API for *OSEK/VDX COM and NM protocols*
  - angle management, etc.

Activities which will be started shortly

- standard C++ language binding for ITRON
- standardization for “Java on ITRON”
- standard interface between ITRON-specification kernel and debugging tools
    - software debuggers
    - ICE
    - logic analyzer

➡ *Tool support becomes easy!*

In Conclusion

ITRON Project is an open activity.

We are waiting for your contributions!

- ITRON Technical Committee (in TRON Association)
  - Hard Real-Time Support Study Group
    - Kernel Specification WG
    - Application Design Guidelines WG
  - "Java on ITRON" Technical Committee
- Embedded TCP/IP Technical Committee
- RTOS Automotive Application Technical Committee