
# Overview of the µITRON4.0 and ITRON TCP/IP API Specification

***4th Nov. 1998***

**Hiroaki Takada**

ITRON Technical Committee
Toyohashi Univ. of Technology

# ITRON Project – *2nd Phase*


- a project to standardize RTOS and related specifications for embedded systems

*1st Phase* (1984– )

- focused on *real-time kernel* specifications

*2nd Phase* (1996– )

- broaden the scope of the standardization effort to related aspects
  - *software components* (software IP)
  - *development environments*
  - *application-specific standards*

*Several standardization activities are in progress.*

# µITRON4.0 – What and Why?


- µITRON4.0 is the next generation µITRON real-time kernel specification.

*Why it is necessary?*

- raising software portability
  - Software portability becomes significant as embedded software is getting larger.
  - Our "*loose standardization*" policy often condradicts with software portability.
- incorporating the results of our recent investigations
  - hard real-time system supports
  - investigation on *automotive applications*
- following the advancement of microprocessors

# Portability *vs.* Adaptability


- Software portability can be raised if we define the kernel functions more strictly.
- Adaptability (*incl.* scalability) is the most important advantage of µITRON, and we should keep it.

*standard profile*

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

| | |
|---|---|
| µITRON4.0 | — loose standardization |
| standard profile | — strict standardiation |

- *Subsetting* is still acceptable for small systems.
- *Extended functions* are also defined.

# Standard Profile – Concept


## *Target System*

- target processor: high-end 16bit to 32bit
- kernel size: 10KB to 20KB with all functions
- The whole software is linked to one module.
- Kernel objects are statically defined.

## *Function Overview*

- including almost all level S functions of µITRON3.0
- incorporating some level E functions of µITRON3.0
- some modifications and more strict definitions based on µITRON3.0
- defining several new functions and APIs

# Standard Profile – Function Overview (1)


*Level S of µITRON3.0*

- basic task management and synchronizations
- semaphore, eventflag, mailbox
- interrupt management, basic time management

*From Level E of µITRON3.0*

- fixed-sized memory pool, cyclic handler
- service calls with timeout

*Major Modifications / More Strict Definitions*

- `act_tsk` with queueing instead of `sta_tsk`
- some terminologies and service call names
- how to write an interrupt handler in C language
- service calls used in an interrupt handler

# Standard Profile – Function Overview (2)


*Newly Defined Functions*

- data queue (queue for one word messages)
- exception handling mechanism
  task exception routine, CPU exception handler
- referencing system status
- `can_act`, `isig_tim`

*Static API*

- Standard description (in a system configuration file) for defining kernel objects statically.
  cre_tsk(...) — service call for creating a task
  CRE_TSK(...) — *static API* for creating a task

*common parameters*

# Broader Scalability

## *Extended Functions*

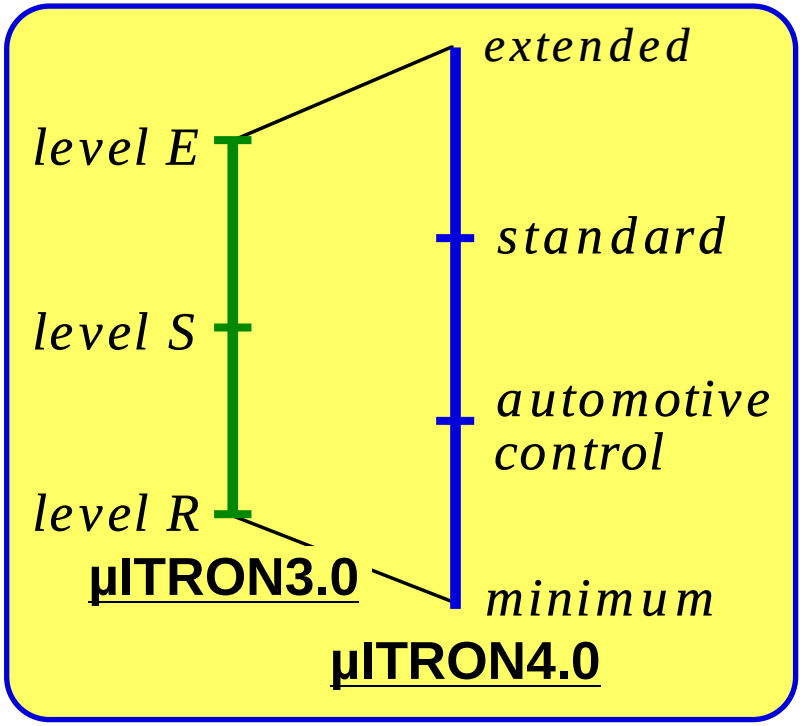
- the other level E functions of µITRON3.0
- hard real-time support
- auto ID assignment

## *Automotive Control Profile*

- a bit smaller subset than level S of µITRON3.0 with *stack sharing mechanism*

## *Minimum Requirements*

- Dormant state instead of waiting state is mandatory.
- All tasks can share a stack without waiting state.



# Standardization Process


Hard Real-Time Study Group

RTOS Automotive Application Technical Committee

µITRON4.0 Study Group

- kernel specification WG
- debugging interface specification WG
- application design guidelines WG
- device driver design guidelines WG

TCP/IP API

JTRON

- *open study group*
- started in April, 1998
- µITRON4.0 real-time kernel specification will be publish in near future.

# 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


## *Design Approaches*

- based on the socket interface
  The socket interface can be implemented as a library on the new API.
- laying importance on understandability
- minimizing the necessity of dynamic memory management within the protocol stack
- optimized API for each protocol (TCP and UDP)
- API for reducing the number of data copies
- considerations for real-time applications
- exploiting the use of static configuration
- harmonized with ITRON conventions, but applicable to other RTOS

## *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.
- Reduced copy API, TCP service calls for reducing the number of data copies, is also defined.

```
tcp_get_buf   tcp_rcv_buf
tcp_snd_buf   tcp_rel_buf
```

- Non-blocking calls and callbacks are supported.
- The callback routine is used for receiving UDP packets.
- *etc.*

## *Standardization Process and Status*


- March 1997 Embedded TCP/IP Techincal Committee is launched.
- April 1998 The specification (Ver. 1.0) is fixed.
- May 1998 approved by the ITRON Technical Committee as an ITRON specification.
- within a few month
  A minor update is planned (Ver. 1.1).
- Several companies (in Japan) are developping protocol stack products conformant to the specification.

# Summary


## *Our Recent Results*

- ITRON TCP/IP API Specification
- JTRON2.0 Specification

## *Current Activities*

- µITRON4.0 Real-Time Kernel Specification
- Application Design Guidelines
- Device Driver Design Guidelines

## *Future Activities*

- Interface specification between µITRON real-time kernel and debugging tools

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