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
## Functions provided in μITRON specification

<table>
<thead>
<tr>
<th>Task management</th>
<th>Eventflag</th>
<th>Semaphore</th>
<th>Mailbox</th>
<th>Memory pool</th>
<th>Others</th>
<th>Processor-dependent functions</th>
</tr>
</thead>
</table>

### Two-Step Adaptation in μITRON Specifications

**OS developer selects functions based on the processor and the target applications**

**Application developer selects suitable functions for the application**

Hiroaki Takada

ITRON Technical Committee
History of the ITRON Specifications

- ITRON1
- ITRON2
- µITRON
  - for 32-bit MPUs
  - first ITRON kernel spec.
  - for 8-bit and 16-bit MCUs
- µITRON 3.0
- µITRON 4.0
- IMTRON
- software components
- HFDS support
- enhanced scalability

Timeline:
- 1984
- 1987
- 1989
- 1992
- 1993
- 1998
- 21st century
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**

<table>
<thead>
<tr>
<th></th>
<th>Single-chip</th>
<th>General-purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>OS type</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Task part</strong></td>
<td>Task part: 29</td>
<td>Task part: 36</td>
</tr>
<tr>
<td><strong>Non-task part</strong></td>
<td>Non-task part: 15</td>
<td>Non-task part: 27</td>
</tr>
<tr>
<td><strong>Fixed priority</strong></td>
<td>Fixed priority 1</td>
<td>Variable priority</td>
</tr>
<tr>
<td><strong>1 task per priority</strong></td>
<td>Subroutine call</td>
<td></td>
</tr>
<tr>
<td><strong>None</strong></td>
<td></td>
<td>Software interrupt</td>
</tr>
<tr>
<td><strong>Exception management</strong></td>
<td></td>
<td>Exit exception, CPU exception</td>
</tr>
<tr>
<td><strong>Wakeup request count</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Semaphore count</strong></td>
<td>Max. 15</td>
<td>Max. 255</td>
</tr>
<tr>
<td><strong>System timer</strong></td>
<td>Max. 255</td>
<td>Max. 65,535</td>
</tr>
<tr>
<td><strong>Program size</strong></td>
<td>0.6 – 4.4 KB</td>
<td>1.9 – 5.3 KB</td>
</tr>
<tr>
<td><strong>Typical RAM use</strong></td>
<td>0.6 – 4.4 KB</td>
<td></td>
</tr>
<tr>
<td><strong>Task part</strong></td>
<td>200 Bytes</td>
<td>640 Bytes</td>
</tr>
<tr>
<td><strong>Non-task part</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Task switching time</strong></td>
<td>17μS</td>
<td>32.5μS</td>
</tr>
<tr>
<td><strong>Max. interrupt masking time</strong></td>
<td>9μS</td>
<td>9.5μS</td>
</tr>
</tbody>
</table>

* 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

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

Hiroaki Takada
ITRON Technical Committee
RTOS used in Embedded Systems
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
    e.g. 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.
  
  \textit{eg)} digital camera

- Some hardware components can be implemented with software.
  
  \textit{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 software 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)
  
  \[\text{profile} = \text{a standard set of kernel functions for a specific range of applications}\]

  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 divide a system into tasks?
  How to assign priorities to tasks?
Framework of the Design Guidelines

application code fragments

guidelines for the grouping

software components

standardized parameters of tasks

assigning priorities

based on RMA

failed

tuning process

tuned

schedulability analysis

succeeded

application software
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 occurred within the protocol stack is not notified to the application.
  - 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?

TCP_LISTEN( listen_port, <protocol info.>)

passive open

listen_port

TCP_CRE_POR(port);
for (;;) {
    tcp_accept(listen_port, port);
    service routine, here
    tcp_disconnect(port);
}

TCP_CRE_POR(port);
for (;;) {
    tcp_accept(listen_port, port);
    service routine, here
    tcp_disconnect(port);
}

tcp_disconnect(port);
}

* 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