µITRON 4.0 Specification

Nicholas J Witchey, Ph.D.
Engineering Manager

U S Software Corporation

µITRON 4.0 - What and Why

µITRON 4.0 is the next generation µITRON real time kernel specification

Why it is necessary?

- Software portability
  - Our "loose standardization" policy often contradicts with software portability"
- Functions for independently-developed software components

  Incorporating the results of recent investigations
  - Hard real time systems supports
  - Requirements for automotive control application
- Following the advancement of microprocessor technology

Portability vs. Adaptability

- Portability of software components built on μITRON can be raised if we define the kernel functions more strictly
- Adaptability (incl. scalability) is the most important advantage of μITRON, so it should be kept

Standard Profile

- The set of kernel functions strictly defines for raising software portability

μITRON 4.0	**- loose standardization**
standard profile	**- strict standardization**

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

Standard Profile - Overview

Target System

- Target processor: high-end 16 bit and 32 bit
- Kernel size: 10kb to 20kb with all functions
- The whole software is linked to one module
- Kernel objects are statically defined

Function Overview (See http://www.itron.gr.jp)

- Includes almost all level S functions of µITRON 3.0
- Incorporates some level E functions of µITRON 3.0
- Includes newly introduced functions
- Several µITRON 3.0 function have been modified; others more strictly defined

Standard Profile - Function Overview (cont)

Level S of µITRON 3.0

- Basic task management and synchronization
- Semaphore, eventflag, mailbox
- Interrupt management, basic time management

From Level E of µITRON 3.0

- Fixed-sized memory pool, cyclic handlers
- Service calls with timeout

Major Modifications / More Strict Definitions

- `act_tsk` with queuing instead of `sta_tsk`
- Some terminology and service call names
- How to write an interrupt handler in C
- Service calls used in an interrupt handler

Standard Profile - Function Overview (cont)

Newly Introduced Functions

- Data queue (queue one word messages)
- Exception handling mechanism
  - task exception routine, CPU exception handler
- System state reference
- `can_act, isig_tim`

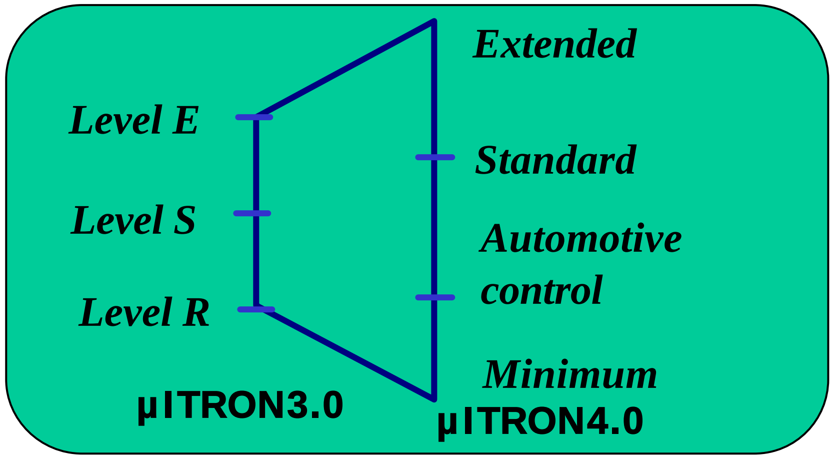
Static API

- 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
  - Both of these have common parameters

Broader Scalability

New Functions not Included in μITRON 3.0

- Data queues
- Task exception handling
- System state reference
- Interrupt service routine
- Hard real-time support
- Automatic ID assignment



Automotive Control Profile

- Smaller profile definition especially suitable for automotive control application

Minimum Requirements

- Dormant state instead of waiting state is mandatory

Functions Supported in µITRON 4.0 Spec

- Task management
- Task-dependent synchronization
- Task exception management
- Basic synchronization and communication
  - (Semaphore, eventflag, data queue, mailbox
- Extended synchronization and communication
  - (mutex, message buffer, rendezvous)
- Memory pool management
  - (fixed-sized, variable-sized)
- Time management
  - (cyclic handler, alarm handler, overrun handler)
- System state management
- Interrupt management
- Service call management
- System configuration management