ITRON Newsletter No.6

ITRON Technical Committee, TRON Association
Katsuta Building 5F, 3-39, Mita 1-chome, Minato-ku, Tokyo 108, JAPAN
TEL: (03) 3454-3191 FAX: (03) 3454-3224

The ITRON Registration System for Products and Applications

The products listed in another page were newly registered in the period from November 1, 1993 through January 1, 1994. Details of the product registration system, and an updated list of registered products, can be obtained by contacting the TRON Association. Previously registered products are also listed in ITRON Newsletter Nos.1 to 3, 5.

ITRON-related Publications

Listed in another page are the publications prepared and issued by the ITRON Technical Committee as of December 1, 1993. The ITRON-μITRON Standard Handbook is a one-volume compilation of the μITRON (Ver 2.0) and ITRON2 specifications. Each of the publications can be obtained directly from the sources indicated.

The latest version of μITRON3.0 is now Ver 3.01.00. Changes made since the μITRON3.0 Standard Handbook was released (Ver 3.00.00) are noted in Newsletter No.5.

New Products

The two products recently registered in the ITRON Registration System for Products and Applications are introduced here.

MORTOS/n68

MORTOS/n68 is a μITRON-specification OS for use in embedded systems, supporting the MC68000 processor.

System call support

System calls through level 4 are supported except for `frsm_tsk` and `get_blk`. The level 5 system call `ret_rst` is also supported, for a total of 52 system calls.

Maximum number of system resources

Tasks:	511
Task priorities:	32 levels
Flags:	127
Semaphores:	127
Mailboxes:	127
Memory pools:	127
Cyclic handlers:	127
Alarm handlers:	127

C language is used for system definition, so when the system is activated an external switch or other means is used to read and convert the system definitions.

A C language system call interface library is provided.

System call parameter checking can be omitted, in order to improve real-time performance.

For system development a C compiler and assembler are required in addition to MORTOS/n68.

Development procedure

1. Creation of a source file (C and/or assembly language)
2. Creation of a system definition file (C language)
3. Compiling
4. Linkage with the MORTOS kernel library
5. Debugging
6. ROM encapsulation

MORTOS/n98

MORTOS/n98 is a μITRON-specification OS for the 8086 processor family. It runs on the PC-9800 Series of personal computers (NEC) or can be used in embedded systems.

† This newsletter is reprinted from TRONWARE vol.25 and TRON PROJECT BIMONTHLY No.30.

Newly Registered Products (Nov. 1, 1993 – Jan. 1, 1993)

Specification	Product Name	Supported Processor	Company
μITRON2.0	MORTOS/n68	MC68000	MORSON JAPAN
	MORTOS/n98	8086 family	MORSON JAPAN

ITRON-related Publications

Name	Type	Price	Publisher	ISBN No.
ITRON-μITRON Standard Handbook	Specification (Japanese)	4,800Yen	Personal Media Co.	4-89362-079-7
μITRON3.0 Standard Handbook	Specification (Japanese)	4,000Yen	Personal Media Co.	4-89362-106-8
ITRON/FILE Standard Handbook	Specification (Japanese)	3,000Yen	Personal Media Co.	4-89362-092-4
ITRON Standard Guidebook '92-'93	Textbook (Japanese)	3,500Yen	Personal Media Co.	4-89362-197-6
μITRON Specification Ver 2.01.00.00	Specification (English)	12,000Yen	TRON Association	–
ITRON2 Specification Ver 2.02.00.10	Specification (English)	15,000Yen	TRON Association	–

NOTES:
- Prices do not include consumption tax.
- TRON Association members may purchase Association publications at a special discount price.
- English-language specifications are also distributed free of charge via the Internet. The method for downloading is explained in Newsletter No.2.

System call support

A total of 25 system calls are supported, for task management (5), task-dependent synchronization (7), synchronization and communication (9), interrupt management (1), and time management/timer handler functions (3). Support for other system calls is planned. These system calls are used from a C language interface.

Maximum number of system resources

Tasks:	511
Task priorities:	32 levels
Flags:	127
Semaphores:	127
Mailboxes:	127
Cyclic handlers:	127

C language is used for system definition, so when the system is activated an external switch or other means is used to read and convert the system definitions.

A system call C language interface library is provided.

System call parameter checking can be omitted, in order to improve real-time performance.

For system development a C compiler is necessary in addition to MORTOS/n98.

Development procedure

1. Creation of a source file (C language)
2. Creation of a system definition file (C language)
3. Compiling
4. Linkage with the MORTOS kernel library
5. Debugging
6. ROM encapsulation (not required when used on a personal computer)

MORTOS versions for other CPUs are also being planned. System developers interested in such an OS are asked to contact MORSON JAPAN.

Recent Works on ITRON

The monthly magazine Interface in its December 1993 issue carries a special feature article on μITRON, entitled, "A real-time OS introduction and μITRON implementation." It starts by describing what a real-time OS is, then explains how to make and use a μITRON-specification OS. A simple OS, for which a source code list is provided, is used in the explanation. This feature should be of interest to anyone desiring an introduction to the world of real-time operating systems and μITRON.

The feature does, however, contain a number of errors or statements that might be misunderstood, so care is required when reading it.

1. Section 3 states that with Hitachi's μITRON-specification kernel only one task of a given priority level can be created. Although a few of Hitachi's μITRON implementations adopt such a specification for the sake of improving performance and saving memory, most of their μITRON-specification OSs, like those of other firms, do not have this restriction.

2. Section 3 also states that the only real-time OS debugger that can be used along with an ICE is that of Yokogawa-Hewlett-Packard Ltd. In fact, however, a number of other firms have released such debuggers for a μITRON-specification kernel.

3. Section 4 incorrectly states that when `pget_blk` and `pget_blf` are executed from the task-independent part, the acquired memory block must be returned before the task-independent part is terminated. However, it is also possible for a memory block acquired in the task-independent part to be returned in the task part, which is the more usual practice.