Introduction to the ITRON Specifications

Open Real-Time Operating System Standards for Embedded Systems

The ITRON specifications are the standards for real-time operating systems and related specifications for embedded systems. Since the project started, we have published a series of ITRON real-time kernel specifications. Of these, the μITRON real-time kernel specification, which was designed for consumer products and other small-scale embedded systems, has been implemented for numerous 8-bit, 16-bit and 32-bit MCUs (Microcontroller Units) and adopted in countless end products, making it an industry standard in this field. Based on these achievements, we have broadened the scope of our standardization efforts beyond the kernel specifications to related aspects, working to expand the ITRON specifications as well as embedded system design technologies.

http://www.itron.gr.jp/

The Status and Features of Embedded Systems

Advances in microprocessor technology continue to open up new application fields for embedded systems. Originally they were used mainly for factory production line control and other industrial applications. Their use spread to communications and office equipment, then on to automotive systems, audio and video products, TVs, cellular phones, synthesizers, game machines, and household appliances such as washing machines, airconditioners and lighting systems. Today nearly all the electrical and electronic products around us are controlled by embedded systems.

In parallel with this trend, the equipment controlled by embedded systems has become more sophisticated, often incorporating many functions in one product. Embedded systems have grown in scale and complexity as a result. Moreover, as products increasingly adopt digital technology, advanced microprocessors have enabled more of the processing to be implemented in software, making embedded systems all the more important.

As a general rule, the small-scale embedded systems typical of consumer products are produced in much greater number and more cheaply than the large-scale systems used mostly in industrial applications. The emphasis in large-scale systems is therefore on reducing development cost, while in the case of small-scale systems the emphasis tends to fall on lowering the cost of producing the end product. In the consumer products field, especially, there is a strong need to shorten system development time in order to keep up with the fierce competition for new products; and once a product has been put on sale, its software is almost never modified. In other words, the system development life cycle is extremely short.

In the field of small-scale embedded systems, wide use is made of a single-chip MCU (Micro Controller Unit) incorporating the processor core, ROM, RAM, general I/O devices and application-specific devices. Developing software for an MCU is made difficult by the hardware resource constraints resulting from the need to keep down the end product cost. Memory, in particular, is likely to be severely limited. On a typical 16-bit MCU there might be 64 KB of ROM and around 1 KB of RAM available, or perhaps 128 KB and 4 KB on a slightly larger system. Another notable feature is the extremely large number of processor cores in use, since the MCU is often optimized to a particular application for the sake of cost performance.

Even in the small-scale embedded system field, the growing scale and complexity of software and the need for fast development turnaround time have made improving software productivity a pressing need. The use of C and other high-level languages, along with the use of a µITRON-specification OS or other real-time OS, have become increasingly common for this reason.

What is the TRON Project?

TRON (The Real-time Operating system Nucleus) is a project started by Dr. Ken Sakamura of the University of Tokyo in 1984. The project aims to create an ideal computer architecture with a view toward the future computerized society. In the computerized society, all the objects making up our living environment will come to have computer chips embedded in them. These objects, moreover, will be capable of interacting with each other via world-wide computer networks. The long-term goal of the TRON Project is to develop highly functionally distributed system (HFDS), in which each of these computerized objects is able to work in cooperation with other objects. These computer-embedded networked objects are called intelligent objects.

Requirements of a Real-time OS for Use in Embedded Systems

As microprocessors continue to advance in performance, their application to consumer goods and other mass-produced items is growing. The demand for improved cost performance in embedded systems is thus as strong as ever. Moreover, the expanding application of embedded systems means that an increasing number of software engineers are coming into contact with a real-time OS, making it highly important to train system designers and programmers in the requisite skills.

Evidence of these trends is seen in the results of TRON Association survey in Japan, Nov. 2000. As shown in Figure 1, a large number of respondents, when asked about problems with using a real-time OS in embedded systems, pointed to such issues as the lack of trained engineers and the large differences in specifications from one OS to another. These relate to training and standardization.

Against this backdrop, we undertook the TRON Project based on the recognized need for standardization of real-time OS specifications for common use across a wide range of embedded systems. The emphasis in this standardization is first of all on consistency of concepts and terminology, in order to make training easier.

The most difficult problem faced in attempting to standardize real-time OS specifications for embedded systems is how to resolve the tradeoff between the need to optimize systems to hardware and the need for improved software productivity. In an MCU-based system with its severe resource constraints, a precondition for adopting a real-time OS is that maximum advantage can be derived from the available hardware. On the other hand, the main incentive for using a real-time OS is to improve software productivity. Yet, when it is attempted to raise the level of abstraction of the services provided by the OS, and to achieve full portability at the source code level across different hardware architectures, the gap between the OS-provided services and the hardware architecture creates a runtime overhead, making it difficult to derive the maximum performance from the hardware.

The optimum tradeoff between these two needs depends largely on the nature of the embedded system. In the case of a small-scale system, it makes little sense to sacrifice runtime performance for the sake of portability, given the need to keep down the cost of the end product. On the other hand, if existing software components

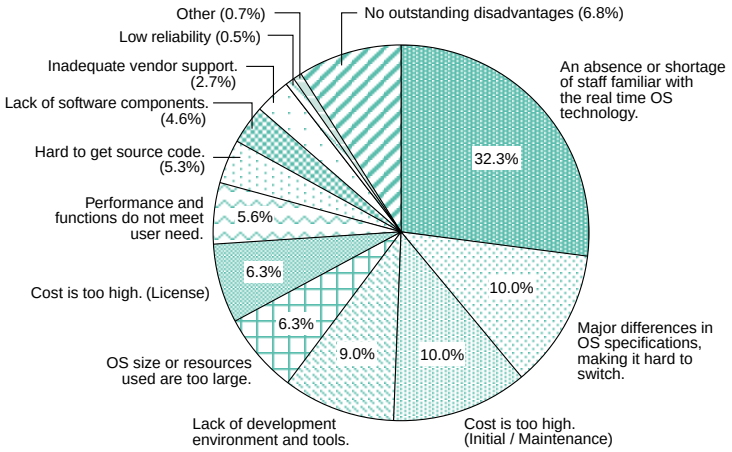

Figure 1: Problems with a real-time OS
(TRON Association survey in Japan, Nov. 2000)

are to be used, or if a system is being developed on a scale that requires reuse of software, achieving software portability is a very important need. Also, the proper balance between the two needs changes as microprocessor technology advances.

There is likewise a big difference in the functions demanded of small-scale and large-scale embedded systems. An OS equipped with advanced functions that are of little use in a small-scale embedded system will only increase the size of the system and lower its performance. A large-scale system, however, can benefit from an OS with advanced functions in the form of improved software productivity.

The demands of a real-time OS, in other words, can vary greatly depending on the scale and nature of the embedded system in which it is used. It would be possible to define separate real-time OS specifications for different system scales and types; but from the standpoints of software engineer training ease, portability of software components, and availability of development support tools, it is preferable to define a real-time OS specification with the scalability to enable common application across a wide range of embedded systems.

The above requirements of a real-time OS specification for embedded system use can be summarized briefly along the following lines.

- Being able to derive maximum performance from hardware.
- Contributing to improved software productivity.
- Being scalable across a range of systems.

In addition to these technical requirements, it is important that the specifications be open in the true sense. Considering that embedded systems are used in all kinds of electrical and electronic equipment around us, the specifications need to be offered in the public domain so that anyone can obtain them, and anyone should be able to implement them freely in commercial products, without paying royalties to the specification developers.

ITRON Specification Design Policy

The ITRON OS specifications were designed according to the following principles in order to meet the requirements discussed above.

- **Allow for adaptation to hardware, avoiding excessive hardware virtualization**

  In order for an OS to take maximum advantage of the performance built into the MCU or other hardware and deliver excellent real-time response, the specifications must avoid excessive virtualization of hardware features. Adaptation to hardware means changing the real-time OS specifications and internal implementation methods as necessary based on the kind of hardware and its performance needs, raising the overall system performance.
  Specifically, the ITRON specifications make a clear distinction between aspects that should be standardized across hardware architectures and matters that should be decided optimally based on the nature of the hardware and its performance. Among the aspects that are standardized are task scheduling rules, system call names and functions, parameter names, order and meaning, and error code names and meanings. In other areas, standardization and virtualization are avoided because they might lower runtime performance. These include parameter bit size and interrupt handler starting methods, which are decided separately for each implementation.

- **Allow for adaptation to the application**

  Adaptation to the application means changing the kernel specifications and internal implementation methods based on the kernel functions and performance required by the application, in order to raise the overall system performance. In the case of an embedded system, the OS object code is generated separately for each application, so adaptation to the application works especially well. (See Figure 2.)
  In designing the ITRON specifications, this adaptation has been accomplished by making the functions provided by the kernel as independent of each other as possible, allowing a selection of just the functions required by each application. In practice, most µITRON-specification kernels are supplied with the kernel itself in library format. The selection of kernel functions is made simply by linking to the application program, at which time only the required functions are incorporated in the system. In addition, each system call provides a single function, making it easy to incorporate only the necessary functions.

- **Emphasize software engineer training ease**

  The ITRON specifications employ standardization as a way of making it easier for software developers to acquire the necessary skills. Consistency in use of terminology, system call naming and the like help ensure that once something is learned, it will have wide applicability thereafter. Another way training is emphasized is by making available educational text materials.

- **Specification series organization and division into levels**

  To enable adaptation to a wide diversity of hardware, the specifications are organized into a series and divided into levels. Of the real-time kernel specifications developed to date, the µITRON specification (Ver. 2.0) was designed chiefly for use with small-scale systems using an 8- or 16-bit MCU, while the ITRON2 specification was intended for 32-bit processors.

Each specification is further divided into levels based on the degree of need for each function. When the kernel is implemented, the level can be chosen based on the kinds of applications aimed for and their required functions. (See Figure 2.) The µITRON3.0 specification divides the system calls into levels, enabling this one specification to cover the range from small-scale to large-scale processors.
Specifications for distributed systems connected to networks and for multiprocessor systems are also being standardized within the ITRON specification series.

- **Provide a wealth of functions**

  The primitives that the kernel provides are not limited to a small number but cover a wide range of different functions. By making use of the primitives that match the type of application and hardware, system implementers should be able to achieve high runtime performance and write programs more easily.

A theme common to several of these design principles is "loose standardization." This refers to the approach of leaving room for hardware-specific and application-specific features rather than trying to apply blanket standards to the extent that runtime performance would be harmed. Loose standardization makes it possible to derive the maximum performance benefits from a diversity of hardware.

ITRON-specification OS Status

Since this project started in 1984, we have studied standard real-time OS specifications for embedded systems, and have developed and made available the series of ITRON real-time kernel specifications as a result. The reason for putting the emphasis on the kernel specifications in this standardization work is that many small-scale embedded systems use only the kernel specifications.

The first ITRON specification was issued in 1987 as the ITRON1 specification. A number of real-time kernels were developed on this specification and applied to systems, mainly by way of proving the applicability of the specification. Thereafter the µITRON specification (Ver. 2.0) was developed with a smaller set of functions geared to 8- and 16-bit MCUs, as well as the ITRON2 specification for 32-bit processors. Both were released in 1989. Of these, the µITRON specification was able to function practically on an MCU with severely limited processing power and memory. It found ready application to a large number of systems running on many different types of MCUs. In fact, µITRON-specification kernels have been developed for the MCUs of nearly all of Japan's computer and chip manufacturers.

During the course of the µITRON application to a wide range of fields, we were able to get a better idea of the need for each function and the performance demands. Also, the expanding use of MCUs in different applications resulted in the µITRON specification being implemented for 32-bit processors, which was not anticipated when the specification was designed. We therefore decided to revise the specification approach by drawing up a scalable specification, able to be used with MCUs across the range from 8-bit to 32-bit processors. The result of this work was issued in 1993 as the µITRON3.0 specification. The English version of this specification can be downloaded from the ITRON Web site. The main functions of the µITRON3.0 specification are summarized in Table 3.

The ITRON-specification real-time kernels registered with the TRON Association as of March 1, 2002 are listed in Table 4. They consist of more than 60 products implemented for around 40 different processors. U.S. Software vendors also have µITRON-specification kernel products. Moreover, because the µITRON-specification kernel is small in size and relatively easy to

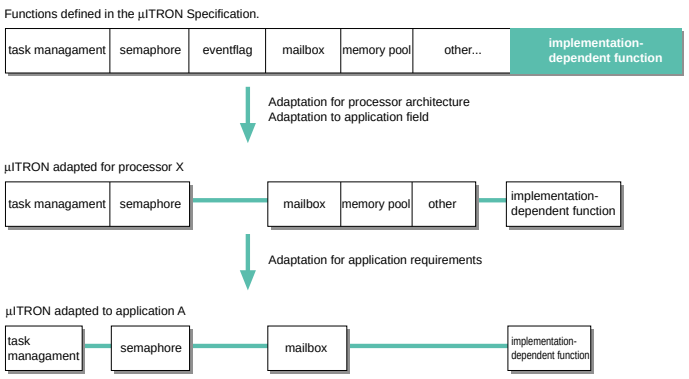

Figure 2: Adaptation in the µITRON specification

Table3: Functions supported in µITRON3.0-specification kernel

1. Task management • Direct manipulation and referencing of task status.
2. Task-dependent synchronization • Task synchronization function in the task itself.
3. Synchronization and communication • Three synchronization and communication functions independent of tasks, namely, semaphore, eventflag and mailbox functions.
4. Extended synchronization and communication • Two advanced task-independent synchronization and communication functions, namely, message buffer and rendezvous.
5. Interrupt management • Function for defining a handler for external interrupts. • Function for disabling and enabling external interrupts.
6. Memory pool management • Functions for software management of memory pools and memory block allocation.
7. Time management • Functions for system clock setting and reference. • Task delay function. • Timer handler functions, for time-triggered starting.
8. System management • Functions for setting and referencing the system environment as a whole.
9. Network support • Management and support functions for a loosely coupled network.

implement, many companies have built kernels for their own in-house use in addition to the products listed here. There are also various µITRON-specification kernels available as free software.

Obviously, with so many ITRON-specification kernels having been implemented, they are being used in many different application fields. Table 5 gives some examples of the huge number of applications making use of an ITRON-specification kernel. As the earlier mentioned survey by the TRON Association also shows, the ITRON specifications are in especially wide use in consumer products fields, where they are a de facto industry standard (see Figure 6). Among the cases where an ITRON-specification kernel is used, very many of these use an in-house implementation of the kernel, attesting to the true openness of this standard specification.

Recent Results and Current Activities

As noted earlier, the TRON Project up to now has concentrated on standardizing real-time kernel specifications. As embedded systems grow large and more complex, however, the need has increased for standardization efforts that take into account software components (or middleware), development tools and other aspects of the larger real-time kernel environment. In the TRON Project today we are first of all putting an emphasis on software component-related standardization. To this end we are working to bring together the conditions promoting the development and porting of software components, and to draw up standard software component interfaces in specific fields.

The studies aimed at bringing together the conditions for promoting software component development and porting are focused primarily on the following two themes. The first is to resolve the issue that porting of software components up to now has been difficult due to the large difference in specifications among ITRON-specification kernel implementations. This requires that the level of standardization of the kernel specifications be raised while retaining the benefits of loose standardization. The second theme is supporting software components with hard real-time characteristics. Software components include many that demand real-time response. What is needed is a framework that allows coexistence of software components with applications while satisfying their real-time constraints, and enabling use of multiple software components each with their own real-time needs. As the result of these studies, we published the µITRON4.0 specification, in June, 1999. The µITRON4.0 specification defines the standard profile which strictly defines the kernel functions for raising software portability. At the same time, the µITRON4.0 specification follows the loose standardization policy as a whole. Namely, a subset of the standard profile is permitted for small-scale systems. Also, many extended functions that are not included in the standard profile are defined in the specification.

The standardization of software component interfaces in specific fields has being taking place for the API (application program interface) of the TCP/IP protocol stack and the Java runtime environment.

The TCP/IP protocol stack has taken on increasing significance in the field of embedded systems, recently. Though the socket interface is in wide use today as a TCP/IP API, it is not appropriate for embedded systems (particularly small-scale ones) because of such problems as its large overhead and the necessity of dynamic memory management within the protocol stack. The ITRON TCP/IP API Specification, which is a standard TCP/IP API for embedded systems, has been designed to solve these problems of the socket interface and to enable a compact and efficient implementation of the TCP/IP protocol stack. The ITRON TCP/IP API Specification has been published on May, 1998.

Java technology is also drawing interest these days. A practical approach for applying Java technology to embedded real-time systems is to implement the Java runtime environment on an ITRON-specification kernel, then build an application system whereby the parts for which Java is best suited are implemented as Java programs, and the parts taking advantage of the ITRON-specification kernel strengths are implemented as ITRON tasks. A key issue here is the standardization of the communication interface between Java programs and ITRON tasks. The JTRON2.1

Table 4: ITRON-specification kernel implementations
(Products registered with the TRON Association as of March. 1, 2002)

Company	Processor	Specification
ACCESS CO., LTD.	SH Family, ARM7TDMI, L7200	µITRON3.0
	SH Family, Strong ARM	µITRON4.0
A.I. Corporation / US SOFTWARE	TRON Task! 4.0	µITRON4.0
ELMIC SYSTEMS, INC.	SH3	µITRON3.0
eSOL Co., Ltd.	V20	ITRON1
	V33A	
	V25	
	V55PI	
	SH2, VR4100/VR4300, TMS470R1x, SR320	µITRON2.0
	Vr 4300, SH-3, ARM7TDMI	µITRON4.0
Firmware Systems Inc.	ARM7TDMI Series	µITRON3.0
	Strong ARM	
FUJITSU LIMITED	F^{2}MC-16LX/16L/16/16H/16F Family	µITRON2.0
	F^{2}MC-8L Family	
	FR Family	µITRON3.0
	SPARClite Series	
GRAPE SYSTEMS INC. / Accelerated Technology Incorporated	MPC68K, PPC860, ARM7 V850e, SH3 SH4	µITRON4.0
Hitachi, Ltd.	H8/300	µITRON2.0
	H8/500	
	H8/300H	
	SH	
	H8S	
	SH-1, SH-2 Series	µITRON3.0
	SH2-DSP Series	
	SH3 Series	
Mitsubishi Electric Semiconductor Application Engineering Corporation	M32 Family	µITRON2.0
	7700 Family	
	M16 Family	
	38000 Series	
	M16C/60 Series	µITRON3.0
	M32R/D	
	7900 Family	
MiSPO Co., Ltd.	8086	µITRON2.0
	H8/300H	
	Z80	
	SH1, SH2	
	H8/500	
	68000, 68010, CPU32	
Morson Japan	MC68020	µITRON2.0
	MC68000	
	8086 Series	
NEC Corporation	78K/III Series	µITRON2.0
	78K/II Series	
	78K/0 Series	
	78K/IV Series	
	RX850	µITRON3.0
	RX850 Pro	
	RX4000	
	RX4000 v4	µITRON4.0
Sony Corp.	SPC900	µITRON3.0
Three Ace Computer Corp.	8086 Series	µITRON2.0
TOSHIBA CORP.	TLCS-90	µITRON2.0
	TLCS-900	
	TX19 Series	µITRON3.0
Toshiba Information Systems (Japan) Corporation	8086 Series	µITRON2.0
	68000 Series	
	TLCS-R3900 Family	µITRON3.0
	Pentium, i486	
Masayuki Kawakami	Z80	µITRON3.0
	Z80	µITRON4.0

* Products not supported outside of Japan are included.

Specification has been designed to define this interface standard and published on Nov., 2000.

Besides software component-related work, another area of field-specific research and standardization which has been undertaken is application of the ITRON-specification kernel to the automotive field. The result has been reflected to the µITRON4.0 Specification.

Other current activities include the definition of the ITRON debugging interface, the interface specification between ITRON-specification kernels and debugging environments, the device driver design guidelines, and the application design guidelines. We are also conducting a project that is to implement some software components running on ITRON-specification kernels and make them free. Other themes we would like to undertake over the course of the coming months and years include standards for C++/EC++ language binding of the ITRON-specification kernel.

Promotional Activities

Even though the ITRON specifications have come to be called an industry standard, the activity of the TRON Project is not so well known, prompting us to step up our promotional work further. At the same time we are making efforts to promote the global acceptance of the ITRON specifications.

Among the specific activities, we have established an ITRON Web site on the Internet. We also take part in trade shows and seminars in the embedded systems field, actively promoting awareness and acceptance of the TRON Project.

As global promotional activities. We promotioned TRON project at the Embedded Systems Conference, the world's largest trade show in the embedded systems field. Base on these activities, we plan to establish the North American Chapter of the TRON Association.

Conclusion

The ITRON-specification kernel has been adopted by many Japanese manufacturers including the leading semiconductor venders, has been implemented for a wide range of different processors and applied in a large number of products in a diversity of fields. The µITRON-specification kernel, in particular, continues to find application to single-chip MCUs that previously could not use a real-time OS due to the memory and execution speed constraints. In the process, it is assuming a position as the world's first standard kernel specification in this field.

Based on these achievements, we are broadening our standardization focus from kernel specifications to related areas such as software components and the development environment. At the same time we are going ahead with surveys and standardization work in specific application fields. As a future direction we are moving toward realization of the HFDS (highly functionally distributed system) that is the goal of the TRON Project as a whole.

Table 5: 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

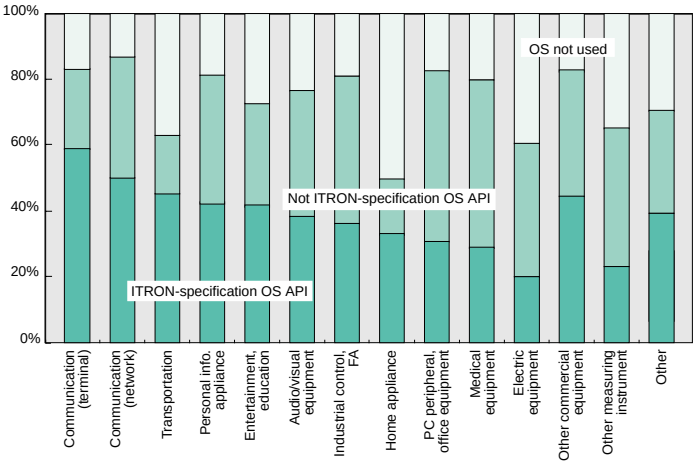

Figure 6. Real-time OS use in embedded systems
(TRON Association survey in Japan, Nov. 2000)

TRON Association

Katsuta Building 5F, 3-39, Mita 1-chome, Minato-ku, Tokyo 108-0073, Japan
TEL: +81-3-3454-3191 FAX: +81-3-3454-3224