T-ENGINE – KIẾN TRÚC PHÁT TRIỂN TIÊU CHUẨN MỞ CHO CÁC HỆ THỐNG NHÚNG THỜI GIAN THỰC

T-ENGINE - OPEN STANDARDIZED DEVELOPMENT PLATFORM FOR REAL-TIME EMBEDDED SYSTEMS

Nguyen Minh Phong, Nguyen Tuan Thanh and Pham Tuong Hai

Faculty of Information Technology, Ho Chi Minh City University of Technology

BẢN TÓM TẮT

Chủ đề của bài báo là T-Engine, một hệ nhúng mở đang được phát triển và chuẩn hóa. Hệ thống bao gồm phần cứng chuẩn và một nhân hệ điều hành thời gian thực. Dự án T-Engine được đề xuất bởi giáo sư Ken Sakamura (http//:tron.um.u-tokyo.ac.jp/) trường Đại học Tokyo. Việc quảng bá hệ nhúng này là nhằm vào mục đích đưa ra một hệ thống phát triển chuẩn cho nhiều ứng dụng khác nhau. Việc chuẩn hóa bao gồm những nâng cấp về cấu hình phần cứng và về môi trường phát triển ứng dụng, cộng thêm với việc phân phối và cho phép sử dụng mã nguồn mở của các khối chương trình phần mềm. Nội dung bài báo giới thiệu về lịch sử của T-Engine, các thế hệ sản phẩm, kiến trúc, thành tựu và các thủ tục phát triển phần mềm. Nội dung bài báo cũng giới thiệu một số ứng dụng mẫu mà nhóm tác giả đã thực hiện trong thời gian qua trên T-Engine/SH7760 và môi trường phát triển Personal Media Corporation.

ABSTRACT

The topic of the term paper is *T-Engine*. T-Engine is an open, standardized real-time operating system (RTOS) development platform for embedded systems. It is composed of standardized hardware (T-Engine platform) and standard real-time kernel (T-Kernel). The T-Engine project was proposed by Professor Ken Sakamura (http://tron.um.u-tokyo.ac.jp/) from The University of Tokyo. It was launched with the aim of standardizing the development platform of embedded systems. This includes improvements to the hardware configuration and the development environment, plus distribution and portability of software components. The term paper would elaborate the history of T-Engine, the generations of its development, its architecture, its achievements, and software development procedures. Then this paper would also include a demo with some small embedded applications, developed on T-Engine/SH7760 development kit of Personal Media Corporation (http://www.personal-media.co.jp/welcome-e.html).

1. INTRODUCTION

These are the days when the term like embedded is increasingly becoming more and more popular in the world. We are flooded with embedded systems that seem to be everywhere. We can define an embedded system as "*A combination of computer hardware and software, and perhaps additional mechanical or other parts, designed to perform a dedicated function*" [1], for example, washing machines, robots, hand-held telephones, modem and automobiles. Each of these devices contains a processor and software and is designed to perform a specific function. For example, the modem is designed to send and receive digital data over an analog telephone line. That's it. And all of the other devices can be summarized in a single sentence as well. If an embedded system is designed well, the existence of the processor and software could be completely unnoticed by a user of the device. Such is the case for a microwave oven, alarm clock, or camcorder.

When developing embedded systems, there are some following challenges:

- Limited operating system support for programming.
- Limited processing time of devices.
- No standardized architecture.

For these reasons, a system architecture standard for real-time embedded systems called T-Engine was developed.

The TRON Project (http://www.tron.org) has established the T-Engine Project in order to promote an open, real-time standardized development environment with the aim of achieving a "ubiquitous computing environment" where everything has a computer incorporated in it and is connected to a network. T-Engine offers an efficient development environment for the development of portable information devices, home electronic appliances and other network devices in a short period of time. The TRON Project's network security architecture, eTRON, has also been incorporated in T-Engine. This eTRON sub-architecture is intended to prevent tapping, falsification, and disguise of malicious users so that electronic information can be safely delivered to the other party through insecure network channels such as the Internet.

For efficient development, the hardware (T-Engine board) and real-time operating system (T-Kernel) are standardized and distribution of middleware is encouraged. Moreover, T-Engine is able to smooth cooperation among chip makers, hardware makers, software makers and system manufacturers, encourage mutual business dealings, reduce development time and cost, thus enabling high value added product offerings in a short period of time.

T-Engine is a standard architecture for next generation, real-time embedded systems aimed at improving software productivity for these systems. An open consortium, the T-Engine Forum (http://www.t-engine.org) in the TRON Project, developed this architecture. Consortium members include computer hardware and software vendors, telecommunication carriers, and computer-using companies. T-Engine development followed a strong philosophy that recognizes the importance of an open standard, middleware distribution, a good balance between virtualization and adaptation, chip-free hardware standardization, and security [2].

T-Engine is designed to provide some following advantages:

- Open standard for designing devices.
- Middleware distribution to develop different kinds of device.
- Chip-free hardware standard.
- More security for devices.

2. T-ENGINE ARCHITECTURE

T-Engine reference system model consists of a target system and a development environment system. The target system is where the developed software runs. The development environment system is where developers build the software. In embedded systems, the target system and the development environment are usually different. The layered design of these systems defines specifications for each layer. Figure 1 shows a model of this architecture.

2.1. Target System's Layered Architecture

In the T-Engine target system's layered architecture, each set of hardware specifications includes

• Functional specifications describing circuit operations, and

• Physical specifications defining the sizes and positions of connectors and other components.

2.1.1. The Hardware Layer

The hardware layer handles system specifications separately for development and products, but the development and product systems share the standardized functional specifications. This sharing assures that the same software works on both systems. They strictly define functional specifications for peripheral hardware, but, for reasons mentioned previously, they leave them unspecified for the CPU to expedite the adaptation of T-Engine on application systems.

Physical specifications for the development and product systems might not be the same. For development systems, they strictly define and standardize the board size, connector type and position, electrical properties, and other characteristics. This enhances the reusability of expansion boards. On the other hand, our approach permits and actually recommends installing product systems based on different physical specifications, provided that the systems satisfy the functional specifications. For instance, when installing a target system on a potentially popular product, it's better to make custom LSI chips and install lower-cost and more compact hardware based on the same functional specifications as those of T-Engine. In the case of products in smaller lots, not requiring downsizing, it's efficient to use the development system as the product system. To support this, they have made the board size in T-Engine's physical specification as compact as possible.

They've developed several system architectures, each aimed at a different target system size. There are four architectures available now: standard T-Engine, micro- or μT-Engine, nano- or nT-Engine, and pico- or pT-Engine.

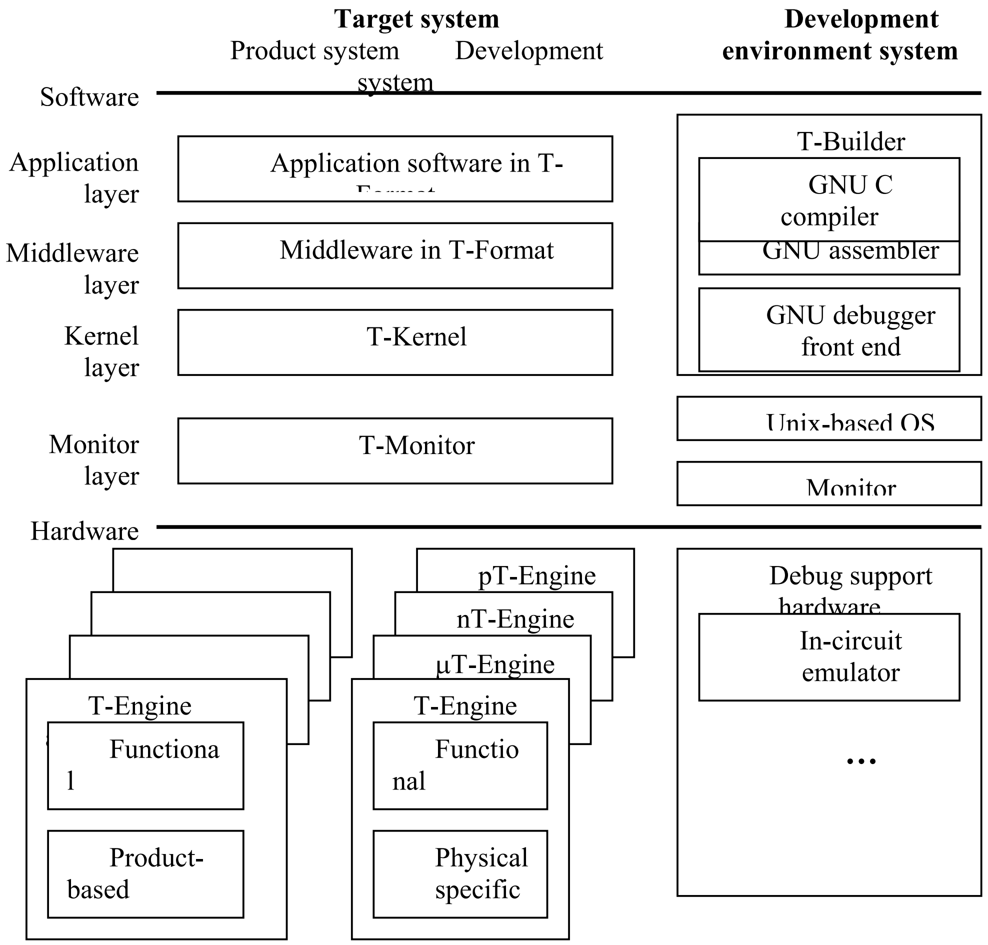


Figure 1: Model of the T-Engine architecture.

Standard Hardware

The standard T-Engine hardware is comprised of a 75mm x 120mm CPU board (see Figure 2), which combines with an LCD board, a power supply board, and an expansion board, making it possible to build the target system hardware out of it. The μT-Engine CPU board is a lot smaller, being 60mm x

85mm in size. The mechanical size and arrangement of the external connectors on the CPU board are also being standardized. The type of CPU applicable is not restricted to any specific one. A unique feature of the T-Engine hardware is that with a miniature board configuration, it is possible to turn it into something closer to the image of the target system [3].

The standard T-Engine platform is suited for mobile information appliances such as next-generation mobile phones and electronic-book readers. Suitable devices would have a graphic display and an input device, providing an advanced user interface; run on a battery; and have a wireless communication function.

The µT-Engine is a standard platform for computers embedded in such units as home appliances and in measuring equipment and it does not necessarily need a memory management unit (MMU). They apply the same specifications used for T-Engine to the expansion connector to enable the sharing of hardware components between both engines. Unlike T-Engine, a graphic display is not required because µT-Engine devices would use a simple user interface. This platform also provides an eTRON chip as standard.

The nT-Engine is an inexpensive, coin-sized hardware platform intended for such nodes as lighting fixtures, sensors, and window controllers. It consists of a processor core, network interface, and peripheral functions. A hardware library integrates the necessary peripheral functions. This platform serves as a processor core, combining functions to create a target system.

The pT-Engine is a chip-shaped platform with a sensor function and a wireless or optical communication function. Only several millimeters in size, pT-Engine fits into a large variety of non-powered objects, including clothing, desks, chairs, paintings (hanging on walls), dishes, and drug bottles. Rather than just a simple radio frequency identification (RFID) device, pT-Engine is also a computer with processing capability. For example, this chip could attach to a wine bottle in transit to monitor and record temperature and vibration data for quality assurance. To run on non-powered objects, the chip might use electromagnetic energy from a communication medium or power generated by micro electro-mechanical systems using micro pulsation.

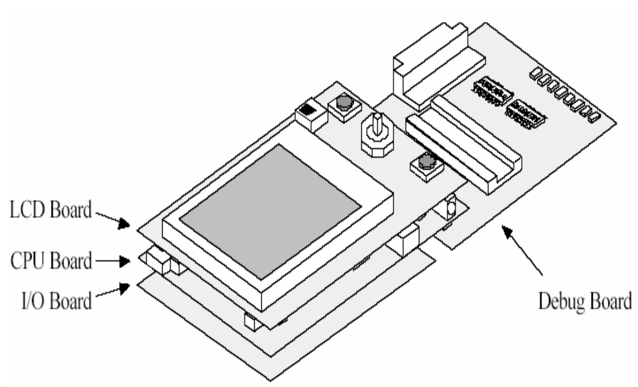
	- CPU board: Interfaces, such as the standard board and serial IO which can operate alone, PCMCIA, a sound codec, USB host, eTRON SIM, LCD, and an expansion bus connector, are carried. - LCD board: The display devices PDA, an Electronic Book, for cellular phones, etc. It consists of user interface, such as a touch panel and a key input. - Extension board: It consists of extended functions, such as various networks, an extended memory, and mass memory. - Debugging board: It consists of debugging functions such as JTAG interface etc., and a writing of Flash Memory function on the CPU board.

Figure 2: The shape of a standard board T-Engine

2.1.2. The software layer

The software layer contains standard sets of specifications (Figure 3). Following these specifications ensures that software is suited to run on hardware satisfying T-Engine's functional specifications. Unlike the hardware specifications, the development and product systems share the software specifications except for a debug support function, which only the development system supports. So, the software layer is divided into the following four layers:

• Monitor: This software enables basic operations on T-Engine hardware, including functions that run application software, load it into memory, and read from and write to memory. This layer defines a standard specification called T-Monitor.

• Kernel: This is the real-time kernel of the T-Engine system architecture, as defined in the T-Kernel specifications.

• Middleware: This layer provides services to application software with user-defined system calls, application tasks, and libraries offered by T-Kernel. The T-Format specification standardizes formats for middleware source and execution code.

• Application: This layer enables applications based on the T-Engine architecture. As in the middleware layer, T-Format defines the code formats for application software.

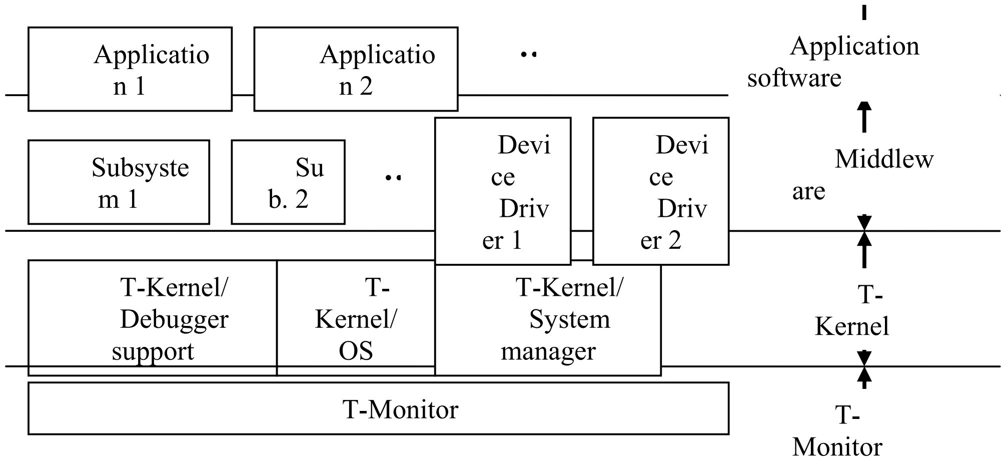


Figure 3: T-Engine runtime software architecture.

T-Monitor

This monitoring software is used for booting OS and debugging. It specification is defined, and the interface with the development environment is taken. T-Monitor is the basic monitor program in T-Engine, intended to be resident in ROM for provision of the following functions.

T-Kernel

The standard real-time OS that runs on T-Engine is called T-Kernel. T-Kernel acts as the implementation platform for various middleware and applications that run on T-Engine, and is used as the common kernel in a ubiquitous computing environment. The T-Kernel/OS with general real-time OS functions such as tasks and semaphores, etc., T-Kernel/SM (System Manager), which improves the distribution of device drivers and middleware, and T-Kernel/DS (Debugger Support) with debugging functions are all being included in T-Kernel. The T-Kernel/SM determines a standardized interface for embedding OS extended functions (managers) and device drivers, and is devised to enable independent development and distribution of middleware and/or applications, and device drivers. On the other hand, if a developer has been using ITRON up to now, the shift to adopting T-Kernel will be smooth as the T-Kernel/OS basic functions are the same as past ITRON-specification OS.

T-Kernel generally refers to all of:

- T-Kernel Operating System (T-Kernel/OS)
- T-Kernel System Manager (T-Kernel/SM)
- T-Kernel Debugger Support (T-Kernel/DS)

But in some cases T-Kernel/OS only (narrow definition) is called T-Kernel.

In T-Kernel and ITRON, we call the resources that we make into the objects of operations kernel objects, or simply objects. Tasks and cyclic handlers, semaphores, mailboxes, etc., are all kernel objects. What we call creating programs on T-Kernel we could also call creating kernel objects.

On T-Kernel, they dynamically manage each type of resource. This is important in order for T-Engine to function as a distribution platform for middleware, and it is a great difference with ITRON. What one must first pay attention to in T-Engine programming is the fact that kernel object IDs are automatically assigned dynamically.

On both T-Kernel and ITRON, kernel objects are identified according to numerical values called IDs. What identifies a task is a task ID, and what identifies a semaphore is a semaphore ID.

On ITRON, normally, these IDs were determined statically at the time the program was created, and it was possible for the user to assign arbitrary values. For example, in the case of task IDs, it was possible to determine at the time of programming that "the ID of Task A is 1, and the ID of Task B is 2."

On T-Kernel, object IDs are all automatically assigned dynamically at the time of program execution. For example, task IDs are assigned through T-Kernel internal processing at the time of task generation. Accordingly, the user cannot assign arbitrary IDs. Also, it becomes necessary to put down in variables inside programs IDs that are assigned by the kernel.

In ITRON, the user decided on the allocation of memory, and this was determined statically at the time the program was created. It was a situation in which the user created a so-called memory map. For example, in a case where one created a memory pool, the user had to allocate the memory region of that memory pool, and he/she had to transmit that to the ITRON system. In T-Kernel, when one creates a memory pool, if one only specifies the size of the memory pool, afterward when T-Kernel executes the program, it will allocate the memory region for you. It is the same also with the regions of message buffers, task stack regions, and so on.

Furthermore, T-Kernel is compatible with high-level memory management in which an MMU is used. Memory management also is based on dynamic allocation in T-Kernel.

The basic logical unit of concurrent program execution is called a "task". In T-Kernel, task states are classified primarily into the five states (Ready, Run, Wait, Dormant, Non-Existent). Of these, Wait state in the board sense is further classified into three states (Wait, Wait-Suspended, Suspended). The task state transitions will be detailed in the figure 4.

Device Drivers

The development of device drivers for general or specific applications is simplified by disclosing the source code of a standard device driver to the public. The system makers can create target device drivers by referring to it. Now, there are many standard device driver specifications for the following devices:

- Serial communications device
- LAN network interface device
- System disk (PC card type, USB storage, RAM/ROM disk)
- eTRON chip SIM interface device
- Clock device
- Keyboard/pointing device
- Screen (display) device
- Audio/voice device
- MIDI device

Besides these, they have drawn up device manager specifications for USB and PCMCIA. These device managers control the USB Host Controller and the PCMCIA Controller, and they are things that provide services to device drivers that deal with each USB device and each PCMCIA device located at a higher level (higher level device drivers).

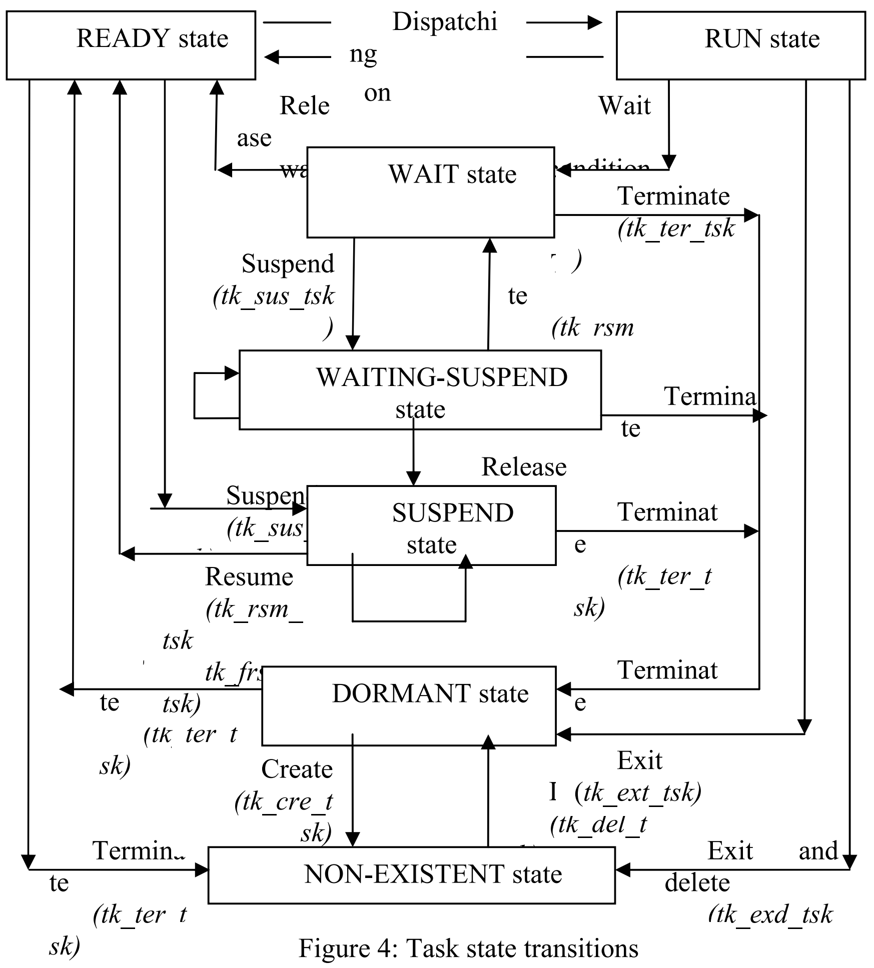


Figure 4: Task state transitions

2.2. Development environment system's layered architecture

The hardware layer for the development environment includes several types of hardware the in-circuit emulator, for example for hardware emulation and testing. The T-Engine architecture does not specifically define hardware specifications except for the interface between the hardware and the target system, and the interface with higher-level development environments.

The software layer includes a development environment layer and an operating system or kernel layer. The development environment software includes compilers, linkers, assemblers, and front-end systems for cross debuggers. T-Builder is the standard development environment specification for these systems and components. T-Builder's specifications conform to those of GNU development systems. For development, a programmer could use any operating system or kernel that T-Builder (the standard development environment) can run on the system.

2.2.1. Middleware and software development

The T-Engine Project is constructing a standard development environment (T-Builder), a standard format of distributed software (T-Format), a basic collection of standard library middleware (T-Collection) that runs on T-Kernel, and a framework for distributing and selling software (T-Dist) that runs on T-Engine.

2.2.2. T-Builder

Specifications for a standard development environment ensure the portability of middleware and application software. Due to its nature, T-Engine first requires a common development environment compatible with multiple CPUs, and, secondly, an open development environment. They have defined T-Builder, a minimum and basic standard development environment, and adopted the GNU development environment. T-Builder is simply a reference standard environment intended for building T-Engine middleware. Development environment vendors will most likely offer better development environments that feature compatibility with T-Builder.

2.2.3. T-Format

T-Format is a set of standard software formats for T-Engine middleware distribution. T-Format consists of a binary-code format, source code style guidelines, and a document format. Software formatted according to T-Format will run on target systems that satisfy T-Engine and T-Kernel specifications. The T-Builder basic software development environment handles source code and binary code based on T-Format. We use C, C++, and Java as basic languages for standardization.

We have employed the executable and linking format (ELF) as the binary code format and Dwarf as the debug symbol format. To prevent symbol collisions when linking multiple libraries, T-Format defines the naming rules and header file format for such global symbols as global variables and exported functions. Regarding source code style, it's crucial to avoid symbol collision when linking libraries. The naming rules defined in binary format for global symbols apply directly to the source code format.

2.2.4. T-Collection

T-Collection is a basic set of software that runs on T-Engine and T-Kernel. For example, the collection might contain a small library consisting of tree programs, a sorting function, a searching function, various encryptions and signatures, data compression algorithms, and data format translations. It could also include basic resident programs, such as those for TCP/IP, and file systems based on file allocation tables. T-Collection is similar to the ACM Collected Algorithms (http:// www.acm.org/calgo) collection and, more importantly, is in a standard format that makes it easy to run code immediately. Normally, the T-Engine Forum will provide these algorithms in source code format but will offer some in binary format. The T-Engine Forum collects the content of T- Collection, referees it, and releases it.

2.2.5. T-Dist

The T-Dist framework aids Internet distribution of software components built by software vendors using T-Format. When a vendor registers a middleware package at the T-Engine Forum, other users can search for it on the T-Dist software database. T-Dist's licensing mechanism also enables license management operations, such as for settling payments related to distribution. To provide this licensing mechanism, middleware handles billing procedures and manages protected execution in coordination with an eTRON secure chip. By using the micropayment, cipher, and authentication functions offered by eTRON, this middleware provides a framework for various middleware licenses. For example, implementations can include settings that permit execution only on eTRON systems that store a certain encryption key. The key is delivered according to the source code license, thus preventing unlicensed system use. As another example, we can realize a payment mechanism in which the price depends on the number of times particular software runs.

2.2.6. Middleware

This represents the various middleware that run on T-kernel. Network protocol stacks, filing systems, Japanese language processing, Kana-to-Kanji conversion, eTRON-specified security software, GUI, audio processing, Java etc., are available. Combining this software makes it possible to develop a stable product within a short period of time. For the purpose of encouraging more middleware distribution, information on the possible uses and combinations of middleware will be managed by T-Engine Project database, and will be accessible to the public. This database system will strongly support the distribution of middleware. It is scheduled to be employed jointly with the eTRON-specified accounting system software in the near future.

2.2.7. Development Environment

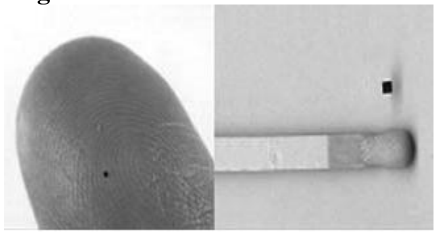
In order to smooth the software development and middleware distribution for T-Engine, all object code formats are standardized using GNU. The GNU development environment running on PC-based Linux, followed by the monitor (T-Monitor) for transferring object programs onto the T-Engine, etc., are all features being included in the T-Engine Development Kit. Moreover, the file management functions for storing the transferred programs onto an ATA card etc., and its command line interpreter (CLI) for commanding the functions interactively are also being supplied to improve the efficiency of software development on T-Engine.

The T-Engine project does not specify the type of CPU used and is able to absorb variations in CPU by its development environment. More specifically, careful consideration has been given so that most middleware, device drivers, and applications developed may be corrected and made able to operate even on a different CPU by the simple act of recompiling. Moreover, the T-Engine project standardizes the object format of programs by referring to the GNU development environment model so that improvement on the distribution of middleware can be achieved.

3. ACHIEVEMENTS OF T-ENGINE

3.1. Various Products based on T-Engine Architecture

Tags



In 2003, the T-Engine Forum adopted standards for ultrasmall (0.4 mm^2) tags capable of holding 6,000 times more data than bar codes. The chips, part of the T-Engine family, are equipped with tiny antennae that can transmit data to a reading device about 30 cm away.

A growing library of middleware is available for T-Engine. According to the T-Engine Forum Web site it includes network protocol stacks, filing systems, Japanese language processing, eTRON-specified security software, a GUI, and audio processing. Other middleware will soon be added to support speech recognition, MP3, and digital watermarks. The platform has also attracted support from American software makers. Sun Micro systems ported Java to T-Engine, MontaVista did the same with its real-time version of Linux, and Microsoft followed suit by porting Windows CE to T-Engine. The ports are referred to as non-native kernel extensions or, informally, as “guest operating systems.”

Tags Everywhere

The UID Center provides the infra-structure for managing electronic tags embedded in or attached to objects in a ubiquitous environment. The center developed the ucode, a multicode tag that automatically identifies information stored in bar codes, RFID chips, smart cards, and electronic tags embedded in virtual entities such as software and electronic money. Comparable to the ISBN (International Standard Book Numbering) code used in the publishing industry, the UID Center assigns unique numbers to each tag and stores data relating to the object in database servers. The ucode tags use a 128-bit code that can be extended in 128-bit units, creating a virtually limitless string of numbers.

The UID Center is conducting verification tests of the technology in several industries, among them transportation, housing, agriculture, and health care. Earlier this year, Japanese farmers, wholesalers, and retailers implemented a food traceability scheme using T-Engine's UID technology. Interest in the technology has increased in recent years, in part due to concerns about mad cow disease, bird flu, and the mislabeling of food items. UID technology lets retailers and consumers trace every phase of the production process. A tag on a packet of soybeans provides information about where the beans were grown, what chemicals were used, and when.

To navigate this tagged environment, the UID Center developed the Ubiquitous Communicator, a PDA-like device that reads ucode tags and retrieves the relevant data from the UID Center's server database. The standard UC has a host of features, including wireless LAN, Voice over Internet Protocol, infrared data communication, and a biometric reader. Apart from the PDA-like version, the UID Center developed a cell phone model and a watch style. All are built with standard T-Engine boards.

The multifunctional UC anticipates an environment in which electronic tags are embedded in every conceivable object. In a restaurant, the UC might read a tag attached to a menu and display today's through insecure network channels, including the Internet. eTRON has a flexible cryptographic architecture and an ID protection protocol to prevent third parties from tracking the tag owner's activities.

Ubiquitous Communicators

A Ubiquitous Communicator (UC) is a new kind of terminal for providing information that is completely different from a PC or PDA. The greatest feature of the UC is its use as a tool for people to communicate with a Ubiquitous computing environment, which is why we call it a "Ubiquitous Communicator."

There are three kinds of communication: communication with physical objects, communication with people, and communication with the environment. When communicating with physical objects, the UC acquires information about the physical objects by communicating with an ultra tiny chip and others attached to each physical object in the user's environment. Communicating with people is, just as the term suggests, communication between people using UCs as mobile phones or VoIP. Communicating with the environment is when the UC gathers information via a computer network installed as LAN or in appliances. The UC then becomes aware of the environment or conditions of a given space and controls the facilities or appliances [5].

Ubiquitous Security – eTRON

eTRON (entity TRON) is a wide-area distributed system architecture designed to securely store and distribute "information" which plays a central role in a computerized society on a digital information infrastructure. Important and valuable information in modern society includes money, various certificates, tickets and keys. To maintain their effectiveness as valuable information, it is required to assure sufficient resistance to forgery, reproduction and modification with physical protection measures, such as special quality of paper and ink and sophisticated manufacturing and printing technologies.

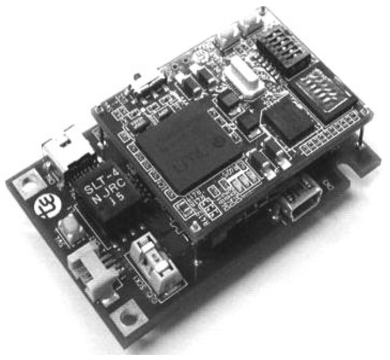
eTRON realizes special digital information which has properties that are made possible with these physical entities, such as unity, difficulty of manufacturing, unreproducible, difficulty of falsification and portability, and this digital information is called "Electronic entity". Electronic entities are stored only in tamper-resistant hardware devices, called "eTRON devices", and transmitted only between these eTRON devices. In addition, security of information transmitted via such communications is firmly protected with cryptographic technologies. In this way, eTRON provides a versatile framework to prevent or detect reproduction, eavesdropping and modification of electronic entities and to securely handle a wide range of valuable information.

For eTRON, various types of eTRON devices are available according to applications. eTRON/8 card is an eTRON device card equipped with an 8 bit micro controller, and a contactless interface compliant with the ISO/IEC 14443, and operates with a weak induced current without external power supply. On the other hand, the currently implemented eTRON/16 chip, which uses a 16 bit micro controller, is a dual-interface eTRON device that is equipped with both contact communication interface compliant with the ISO/IEC7816 and contactless interface compliant with the ISO/IEC 14443.

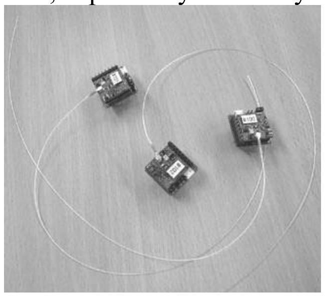
An eTRON/16 chip is designed on the assumption that it is embedded into various computer nodes, such as T-Engine, and is equipped with complex instruction to support many types of applications that handle electronic entities described above. One of the implementation examples is a SIM card.

Sensor Network

In a ubiquitous society, a sensor network is one of the services that makes "Context Awareness" possible. Specifically, a sensor network is a network in which nT-Engines and pT-Engines connected to various sensors provide environment information, collected from each sensor, to UCs (Ubiquitous Communicators), other nT-Engines and pT-Engines. It also enables control of actuators connected to nT-Engines and pT-Engines based on the environment information.



nT-Engine is a node that is connected to a sensor to collect environment information and controls the environment to enable a comfortable (optimum) condition based on this information. As nT-Engine is required to function as a sensor node or actuator node, it is equipped with various interfaces. The prototype nT-Engine that was made by the YRP Ubiquitous Networking Laboratory this time is equipped with serial, versatile I/O, AD, DA, PWM, various timers, interruption, and DMX (lighting control) interfaces. UNP (Ubiquitous Network Protocol) was newly developed as a network protocol and implemented on nT-Engine. UNP is a protocol for collecting information and controlling devices under an ubiquitous environment. UNPs form a protocol stack. UNP is a protocol appropriate for the Ubiquitous era, supporting the automatic network establishment function and network security function. nT-Engine can be applied to lighting, air conditioning, door and blind control within houses, building management systems, auto body-related systems, expressway-related systems, etc.



pT-Engine is an ultra-tiny active wireless node in a ubiquitous society equipped with power supply, a micro computer, and various sensors. By attaching pT-Engines to various physical objects and environments, the sensors and microcomputers in the pT-Engines continuously collect information on the environment and store it internally.

Furthermore, nT-Engines that function as a single base station and 1000 node pT-Engines can communicate with each other to use information on the environment from a sensor network of nT-Engines. pT-Engine enables control of secure communications and secure information with eTRON identification and by encrypting and processing stored information. The prototype pT-Engine that was made by the YRP Ubiquitous Networking Laboratory this time adopts the low power wireless communication method, but in the future, we are thinking about adopting the UWB (Ultra Wide Band) communication method that enables high-precision positioning, which will make it possible to relate position information acquired with three-point measurements using multiple nT-Engine base stations, and information on environments, such as temperatures.

TRON House

TRON Project Leader Ken Sakamura, in cooperation with Toyota Home K.K., has designed and developed a new intelligent home based on TRON and other leading edge technologies. Called "Toyota Dream House PAPI," this new intelligent home is designed to reflect the ubiquitous computing technologies that will be available for intelligent home construction in the year 2010 [6].

The main goals of this project were to design and realize an environmentally friendly, energy saving intelligent house design in which the latest ubiquitous network computing technologies created by the T-Engine project could be tested and further developed. This is important to keep in mind, because the natural tendency among some members of the press is to heap criticism on a house like this as too expensive, too impractical, or too intrusive to be sold in Japan. Not everything developed in Toyota Dream House PAPI is going to be incorporated into everyone's house in the future.

In Toyota Dream House PAPI, the already established concept of the home theater has been developed one step further. This home theater knows where the human occupants are and adjusts the lighting and sound to their locations and preferences. If any changes have to be made to any devices in the room, such as the air conditioning or the ventilation, the Ubiquitous Communicator can easily make them.

Toyota Dream House PAPI will be available for viewing by the public in groups with a maximum of eight persons from March 25 through September 25, 2005.

3.2. Various Middleware

A growing library of middleware is available for T-Engine. According to the T-Engine Forum Web site, it includes network protocol stacks, filing systems, Japanese language processing, eTRON-specified security software, a GUI, and audio processing. Other middleware will soon be added to support speech recognition, MP3, and digital watermarks. The platform has also attracted support from American software makers. Sun Microsystems ported Java to T-Engine, MontaVista did the same with its real-time version of Linux, and Microsoft followed suit by porting Windows CE to T-Engine. The ports are referred to as non-native kernel extensions or, informally, as “guest operating systems.”

3.3. Growing support

The T-Engine Forum has grown from 22 members when it was founded in 2002 to nearly 500 members today. Among them are nearly all of Japan’s blue chip companies and a growing number of global tech giants such as Sony, Toyota, Microsoft, and IBM. The Forum has development centers in China, Korea, and Singapore.

3.4. Multilingual environment

One of the beneficiaries of growing support for T-Engine is Personal Media Corp., producer of T-Engine development kits and a long-time developer of TRON products. In the 1990s, PMC developed Cho Kanji, a multilingual environment that supports Japanese, Chinese, Korean, and about 40 other languages. Cho Kanji is an alternative for Unicode. The latter uses four-byte encoding for pictographic characters. Cho Kanji uses two-byte encoding, a considerable advantage in network environments and when using devices with limited memory. Cho Kanji currently supports approximately 170,000 characters, which can be used inside the application and in file names.

4. EMBEDDED APPLICATION DEVELOPMENT

4.1. Categories of Software to be developed

The software to be developed can be classified broadly into three categories, which differ in development method, program object format and other ways [7].

- Monitor-based software (Programs that run on T-Monitor)

Software that does not use T-Kernel functions, running directly on hardware in a non-MMU environment and loaded into memory and run by T-Monitor. In T-Monitor, a program load and execute function is furnished as a debug function. By utilizing this, one can run programs on top of T-Monitor.

However, basically, one cannot utilize other system functions, such as T-Kernel and devices drivers, from programs that run on T-Monitor. Accordingly, programs that run on top of T-Monitor are mainly used for things like testing hardware and debugging, and they are not appropriate for regular applications

- T-Kernel-based software (Programs that run on T-Kernel)

These programs are the basic form of user applications that we run on T-Engine, and they are also the closest in form to the conventional embedded programs that used ITRON. User applications consist of one or multiple tasks, and we can use T-Kernel system calls from each task. A device driver or other T-Kernel-based software runs in an environment using an MMU, as a resident program in system memory space.

System space is the addressable memory space from 0x40000000 to 0x7fffffff.

After using the CLI recv command to save the program as a file to the work disk, the program is loaded and run using the CLI or IMS lodspg command. When the program is finished running, it continues to occupy the memory area. The memory must be freed using the unlspg command.

When a program is loaded, relocation takes place automatically. The actual address to which it is loaded is displayed by the lodspg command, and can be confirmed by the CLI ref spg command.

- Process-based software (Programs that run on Extensions)

User applications that run on top of T-Kernel Extensions differ greatly from user applications that run on top of T-Kernel.

User applications on T-Kernel Standard Extension run in terms of units called processes, and they can utilize system calls that the extension provides rather than T-Kernel system calls. Among the system calls the extension provides, there are things such as process control, file control, and interprocess communication.

Process-based software runs as an ordinary application in an environment using an MMU, and is loaded into local memory space.

Local space is the addressable memory space from 0x00000000 to 0x10000000.

After using the CLI recv command to save the program as a file to the work disk, the program is loaded and run using the CLI or IMS program run command.

Process-based software runs at the user-level protection level and therefore cannot make direct use of T-Kernel functions, nor can it access I/O space directly.

What forms the core of T-Kernel is T-Kernel/OS. Functions that correspond to ITRON are mainly under the charge of T-Kernel/OS. In the programming of applications that run on T-Kernel, what one first must understand is this T-Kernel/OS.

What T-Kernel/SM provides are extended functions in T-Kernel that did not exist in ITRON. In cases when we embed in a system middleware on T-Kernel, it is necessary for one to understand T-Kernel/SM.

What T-Kernel/DS provides are functions for development tools, such as debuggers. Accordingly, it is not necessary to be conscious of T-Kernel/DS in carrying out normal programming.

4.2. Preparations for Program Development

- A T-Engine development kit, use the products of Personal Media Corporation, assuming that the reader is using standard T-Engine.
- Prepare a Linux PC
- Install the development environment in the Linux PC and set the environment variables
- Connect the Linux PC to T-Engine
- Create a work disk to be used with T-Engine
- Write program, compile and build
- Download and execute program on T-Engine kit, debug if needed.

4.3. The Structure of a Program that Runs on T-Engine

The structure of the main function is basically as follows:

```
EXPORT ER main(INT ac, UB *AV[ ])
{
if(ac<0)
{  /*call at the time of unlspg*/
      Program           termination
   processing
}
else
{  /*call at the time of lodspg*/
      Program startup processing
}
return 0;
}
```

4.4. T-Engine Development Kit – An Open Development Platform for Embedded Systems

The T-Engine Development Kit has been released by Personal Media Corporation (PMC) as the first commercialized product of the T-Engine specification. T-Engine has been placed in the spotlight as the super development platform for ubiquitous computing. The T-Engine Development Kit is comprised of a T-Engine CPU board, a real-time operating system (called T-Kernel), specifications and other documentation on CD-ROM, as well as a PC-based Linux development environment [8].

PMC T-Shell development kit

PMC T-Shell development kit is middleware for T-Engine that enables provision of a GUI system with rich character support. It includes screen drawing functions, GUI parts, a windowing system and other GUI functions, as well as kana-kanji transform, TrueType fonts with more than 170,000 characters, and a TCP/IP manager. It can be used to develop GUI-based applications quickly and easily. Moreover, it comes with the visual language MicroScript to be run on T-Engine with little or no modification.

4.5. WideStudio for T-Engine

WideStudio is an open source, Integrated Development Environment for desktop applications purely made in Japan. This enables you to develop GUI applications that can run on Windows95/98/Me/NT/2000/Xp, WindowsCE, Linux, FreeBSD, SOLARIS, MacOSX(w/X11), BTRON, T-Engine, mu-CLinux(wo/X11) in various programming languages such as C/C++, Java, Perl, Ruby, Python, Objective Caml.

Since an application is build on MWT (Multi-Platform Widget Toolkit) which runs on multiple platforms, WideStudio applications are all source compatible between these platforms. If you developed an application in C/C++ language, you only need to re-compile the source code to run on a different platform in as native code [9].

WideStudio for T-Engine is one of WideStudio Suite that enables you to develop a desktop application on T-Engine. WideStudio for T-Engine offers the following features to develop T-Engine applications:

- Generate T-Engine native binaries that runs much faster
- Source codes can be shared among other platforms.

WideStudio for T-Engine uses Linux or Windows Operating Systems as its development environment for T-Engine applications. Applications developed on Linux or Windows are transmitted to T-Engine to run or for debugging.

Required software is show below:

- (For Linux) Compilers such as gcc/g++
- Your favorite editor to edit source codes
- T-Kernel Development Kit (from Personal Media Co.)
- T-Shell Development Kit (from Personal Media Co.) *As required*

Also, gcc compiler for T-Engine come with T-Kernel development kit is required for both Linux/Window.

5. SOME SMALL EMBEDDED APPLICATIONS

- Calculator: a simple calculator can do most common functions (add, subtract, multiply, and divide integer or floating point numbers). It uses touch screen to input numbers and displays result on LCD.

- Snake: snake-hunting game like one in cell-phone.

- Image viewer: can view images (in BMP, JPEG, GIF format) on LCD. Images can be stored on USB flash disk or CF card.

- Watch: emulate a sport watch,

- Drawsamp: users can use touch screen to draw anything on LCD.

6. CONCLUSION

T-Engine will be an important technology and platform in the world of embedded systems. By now, several computer vendors have released more than ten hardware systems based on T-Engine specification. Upon these, software vendor companies are developing major middleware components for embedded systems.

T-Engine also enjoys growing support in other East Asian countries. For instance, Renesas Technology Corp., Nanyang Technological University, and the Singapore government co-founded Singapore's T-Engine Application Development Center (TEADEC), which provides English and Chinese tutorials for T- Engine development. TEADEC also supplies low-cost training material to training centers in Thailand, Malaysia, and Vietnam.

REFERENCES

1. The NetBSD Foundation URL: www.netbsd.org/Misc/embed.html#a1
2. Ken Sakamura & Noboru Koshizuka, University of Tokyo, "The Open, Real-time Embedded-Systems Platform", page 1-3.
3. http://www.t-engine.org/english/whatis.html
4. T-Kernel specification of PMC T-Engine development kit.
5. TRON Show 2005 Report: http://www.tron.org/tronshow/2006-e/ts2005-03-01.html
6. Toyota Dream House PAPI: http://tronweb.super-nova.co.jp/toyotadreamhousepapi.html
7. T-Engine programming appeared on pages 17-25 in Vol. 81 of *TRONWARE*.
8. http://www.personal-media.co.jp/te/en/tekit.html
9. http://www.widestudio.org/EE/