

Introduction to Linux-based solution for embedded software development

**Section 1 Eddy Real-Time Linux, Lemonix™**

Section 2 Eddy Integrated Development Environment, LemonIDE™

Section 3 Eddy Utility Programs

# **Eddy Real-Time Linux, Lemonix™**

## **: the basis for embedded software standard platform**

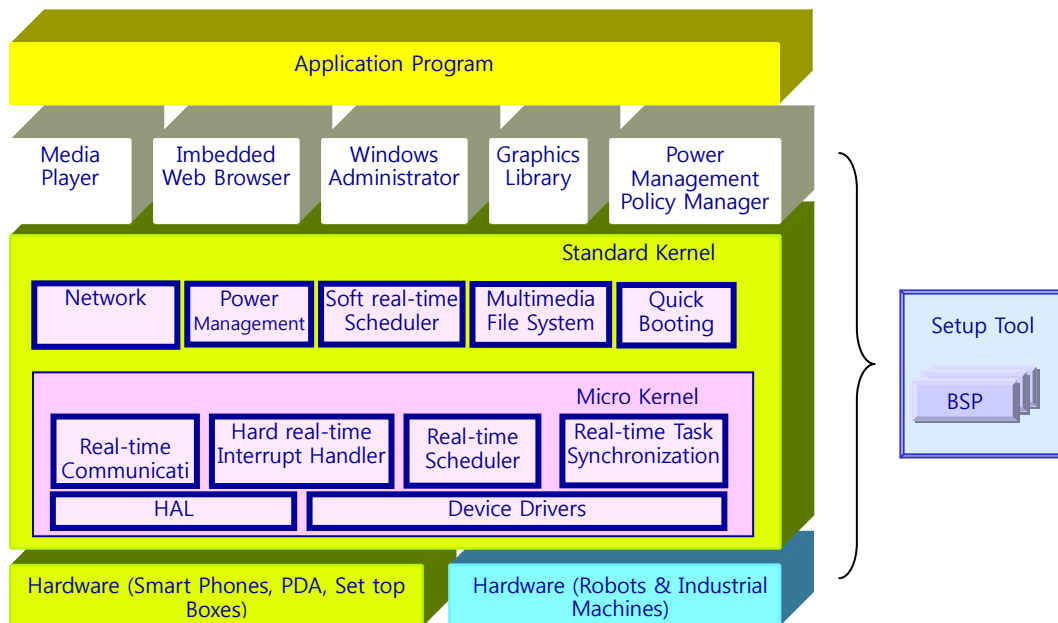
Lemonix™ is a Linux-based standard operating system based designed to be applied on diverse next-generation embedded software platforms. To introduce a Linux based solution for developing embedded software systems, we would like to summarize and introduce special features, primary capabilities and system requirements of Lemonix™.

### **1. Summary**

Lemonix™, unlike classic embedded operating systems, supports network-based operations and ubiquitous network system to utilize standardized, miniaturized, real time embedded systems in ubiquitous computing environment. Furthermore, it is provided as embedded software standard platform with diverse middleware and increased convenience.

While embedded software with various scales and properties are in demand, application software programmers call for more coherent and unified developing system. Lemonix™ accepts all these requests.

The Lemonix™ operating system implants hardware optimization technology such as miniaturization, low-power support, resource management since it is limited in size, price and resources and has a structure of the following diagram. Moreover, it provides network support, real-time scheduler, multimedia file system and quick booting capability all of which can be easily installed to target embedded systems through setup tool.



[\* Structure of Lemonix™ Operating System]

Kernel's HAL (Hardware Abstraction Layer) is an abstract layer provided to the operating system for handling a variety of hardware. Target builder's QSP (Lemonix™ Support Package) not only provides operating system kernels but also organizes and sets up multimedia player, middleware such as Java Virtual Machine and application programs.

Next, we will demonstrate the precise capabilities of Lemonix™ and target builder and the overall support system for applying Lemonix™ as a standard platform.

## 2. Features

Lemonix™ comes up with following features.

### Real-time support

Lemonix™ kernel implants all the positive features of standard Linux kernel while satisfying user's needs for real-time applications by reorganizing the standard Linux kernel's structural problem. As standard embedded Linux kernel is a classic monolithic based kernel, it emphasizes more on throughput than responsibility, the indicator of real-time aspect. Thus, in order to support real-time operations by increasing responsibility, Lemonix™ kernels are composed of preemptive kernels by modifying lock mechanism of the kernel and related codes and furthermore, introduce the lock break technology.

### Power Managing

Power managing capability is crucial in embedded systems to use batteries efficiently and is developed with active power management technology, a technology that minimizes energy consumption while minimizing the

performance decrease by adjusting each device's power condition, as standard technology. Lemonix™ is designed to maximize the efficiency with active power management structure.

## **Multimedia File System**

Low-spec systems are used in embedded systems for price competition creating limits in handling multimedia data. To overcome these restraints, Lemonix™ assures the multimedia data's QoS while supporting journaling capability that is indispensable in embedded system at the same time.

## **Communication**

For real-time support in embedded systems, Lemonix™ revises the RTP communication protocol in network module, thereby developing network protocol stack in kernel level instead of in the system library range. Moreover, it is combined with BSD socket API which uses system calls widely in POSIX based operating systems.

## **Setups**

Since Lemonix™ operating system consists of complex and diverse components with complicated dependence between them, tools are provided for easier modification and installation of the system. Users can easily modify kernels and systems for many ways like installing Lemonix™ to target systems, checking the system application device's memory usage to see if the device can be installed to the target with current options or installing the final image to the target in many different ways.

## **3. Primary Capabilities**

Lemonix™ is a Linux kernel based operating system that provides real-time scheduling, low-power managing, multimedia data handling and related device drivers. Details to primary features are as follows.

### **Real-Time Support**

As is in all other operating system's real-time support, the most important aspect of real-time support in embedded operating system kernel is to raise preemption. Because, for an operating system to have real-time aspect, it should guarantee correctness of time and operation process in beginning and end of real-time task that occurs in limited time when a task that has higher priority over normal task occurs in the operating system. Of all these necessities, the responsibility, which indicates how fast each portion of the total system can respond to synchronous and asynchronous events, becomes important for timing guarantee.

When a normal task enters into the kernel using system calls in a standard embedded Linux kernel, real-time task cannot be executed immediately even if a real-time task with higher priority occurs since Linux resembles the classic UNIX monolithic kernel structure and emphasizes more on throughput. Therefore, structure of the kernel should be modified into a more preemptive structure in order to apply real-time support. In other words, preemption should be applied to all areas leaving some kernel codes that must be protected.

Lemonix™ operating system kernel allows real-time support with some features added and improved. The two main features added are Preemptive Kernel support and Lock Breaking. With these features, Lemonix™

operating system kernel can provide both the real-time support which the standard Linux kernel lacks and the benefits of using standard Linux kernel. The kernel used is based on standard embedded Linux 2.6.x kernel.

- **Preemptive Kernel Support**

To provide preemptive kernel, fundamental codes inside kernel must be protected even when a task enters into the kernel after another task, and preemptive kernel becomes associated with lock mechanism. In other words, the preemption of the task itself must be protected through lock mechanism. However, classic standard Linux kernels did not support these features. So Lemonix™ kernel has modified and developed lock mechanism to provide preemptive kernels. The following shows provided features and notes.

- **Lock Breaking Support**

Although the embedded operating system kernel itself can be preemptive with above features, spin rock area cannot be preemptive. When this spin rock area is short, it would not affect the response time. However if this area gets long, it can greatly decelerate the responsibility of the system. Lock breaking is used to resolve this problem. Lock breaking technique improves responsibility of the system by revising long rock areas with short ones.

- **Response Time Analysis**

Lemonix™ RTOS, designed to have maximized real-time performance by minimizing the kernel response time through preemptive feature and lock breaking technique, guarantees superb response time compared to other RTOS.

	Max. latency	Min.Latency	Avg.Latency
Lemonix™ RTOS	36.83	5.58	8.16
Linux by M company	42.95	6.52	7.86
General Linux	7,021.25	2.81	29.89

[ \* Kernel response latency (usec) ]

## **Power Management**

As more mobile instruments such as PDA, smart phones and etc. are taking larger importance in embedded systems and they all depend on batteries to function, it is essential to develop techniques to manage batteries in embedded operating systems efficiently. Power management is a technique that minimizes performance decrease and energy consumption by adjusting each device's power condition according to accurate estimations on operation amount. Nevertheless, total system's power efficiency can decline even after each device's power consumption is minimized through power management and the management needs to be performed on system level. In addition, application's property and demands should be used to calculate operation amount accurately. The efficiency of power management can be maximized through these operations.

Lemonix™ Active Power Management Structure provides following features.

- Active power management on system level – Minimizes power consumption on CPU, LCD and memory through system monitoring
- Active power management for periodic applications.
- Decreased power consumption through dynamic voltage/frequency changing with demands on performance increase in MPEG players.
- Active power management for interactive applications
- Less power consumption in works such as Web browsing or Text editor.
- Power Management Policy Manager
- Low power Disk I/O (Free Space File System, FS2)

## **Embedded File System**

Low-spec systems are used in embedded systems for price competition creating limits in handling multimedia data. To overcome these restraints, Lemonix™ assures the QoS of multimedia data while supporting journaling capability that is indispensable in embedded system at the same time.

The primary objective of file system is quick access to data. For quick access of data, the file system stores data in structured form. Since, structured data form can be formed by another kind of data, the data stored for structuring data are called meta-data (in other words, data on data). The meta-data has its meaning as data when it guarantees integrity.

To have integrity means that the data itself does not contain errors, and the file system performs FSCK (File System Consistency Check) in order to maintain this integrity. Fortunately, the file system checks to keep integrity of meta-data on system shut down. When the system initiates, the file system receives the meta-data assured by the system. These actions can be performed on normal system shut down and initiation. These actions, however, can be quite a burden to the system on abnormal shut down and following initiation.

File system can use meta-data from the system after checking its integrity with FSCK. When the file system realizes that the meta-data is not the same data it passed to the system through normal shut down process, it checks almost all meta-data that were handled by the system to find a meta-data with integrity. The difficulty in this process is the required time.

Normally, the process takes from a couple of minutes to couple of hours depending on the system. To resolve this conflict, Journaling File System is used. It is a system in which the file system takes responsible for both the consistency and the resulting load.

Lemonix™ resolves this FSCK problem in Journaling File System by introducing a new data structure called Journal. By recording what the file system will do before making a change on meta-data, the Journaling File System can keep the log of recent changes in meta-data which can be used to check consistency on file system on abnormal shut down. Maintaining integrity is important as there are many abnormal shut downs in embedded systems.

An example of Journaling File System is Linux ext3. Ext3 introduces a block device driver called JBD (Journal Block Device Layer) that handles the journaling part for maintaining the structure of ext2. This receives the data block the file system wants to store and periodically stores the data in its own block device. It also restores the

stored data when it is necessary. Since JBD is not part of the file system, it can be used in any block device based file system in which journaling is needed and therefore, JBD is useful for construction of the journaling file system.

To use journaling in file system, the system only needs to call API provided by JBD on necessary parts. The actual journaling process is done by JBD afterwards. The disk format is identical as well since it is based on existing file system code. In other words, file system transform can be easily done as identical meta-data are used. Upgrading to journaling file system generally requires backing-up data, disk formatting, mounting journaling file system and restoring data. However, transforming to ext3 from ext2 can be done with backing-up data with ext2 mounted. Therefore, Lemonix™ provides journaling using JBD.

## **Micro Kernel**

Embedded Linux kernel based Lemonix™, as mentioned before, supports real-time capability. However, the real-time support, strictly speaking, only supports soft real-time, making it incapable of handling hard real-time embedded systems or devices that require small amount of footprints such as military control, robot control, medical devices or industrial control units. Thus, Lemonix™ supports task occupation time limit of 100µs or less, which is about as much as classical RTOS supports, allowing it to handle devices mentioned above.

There are two primary techniques used for this which are priority-based interrupt thread and voluntary occupation. Priority-based interrupt thread makes the interrupt management into a thread itself, since the nature of serialized interrupt management makes it hard to support real-time thereby allowing Lemonix™ to have priority order and handle interrupts in urgent tasks ahead. Voluntary occupying is a technique that allows scheduling in less than 1 ms, usual scheduling resolution, by inserting an occupation point in the kernel. Moreover, it provides kernels that have small amount of footprint about 100KB to increase usage.

## **Target Builder**

Target Builder is a tool-kit for conveniently setting and installing target image and supports the product's most efficient system organization. It helps modifying options on kernel, basic applications and target properties just by clicking on GUI while supporting following features at the same time.

- Simple creation of projects using wizard; variety of busyboxes that helps automatic reliance check and setup; precise setup support for about 30 basic packages such as tiny login.
- Creation of diverse target root file systems (ext2/ext3, ramdisk image, jffs2, cramfs etc.) and CD installer; target image creation and automatic loading capability through USB memory stick installer.

## **4. System Requirements**

Lemonix™ has the following system requirements.

### **\* Compiler**

- Eclipse based LemonIDE™ integrated development environment support
- GNU C/C++ 4.1.1 · arm-linux cross-compiler support

**\* Host System**

- Linux installed computer
- Install using virtualization software in Windows environment

**\* Development Hosts**

- Fedora Core 4, 5, 6
- Redhat 9.0
- Redhat Enterprise Linux
- Ubuntu Linux 6.x, 7.x
- Debian 4.0
- SUSE Linux
- CentOS 4.5
- Asianux

**\* Target Systems**

- Eddy-CPU V2.0
- Eddy-S1/PIN V2.0
- Eddy-S1/DB9 V2.0
- Eddy-S1/DB9-PoE V2.0
- Eddy S2M/PIN V2.0

**\* Hardware Support**

- ARM 9 cores