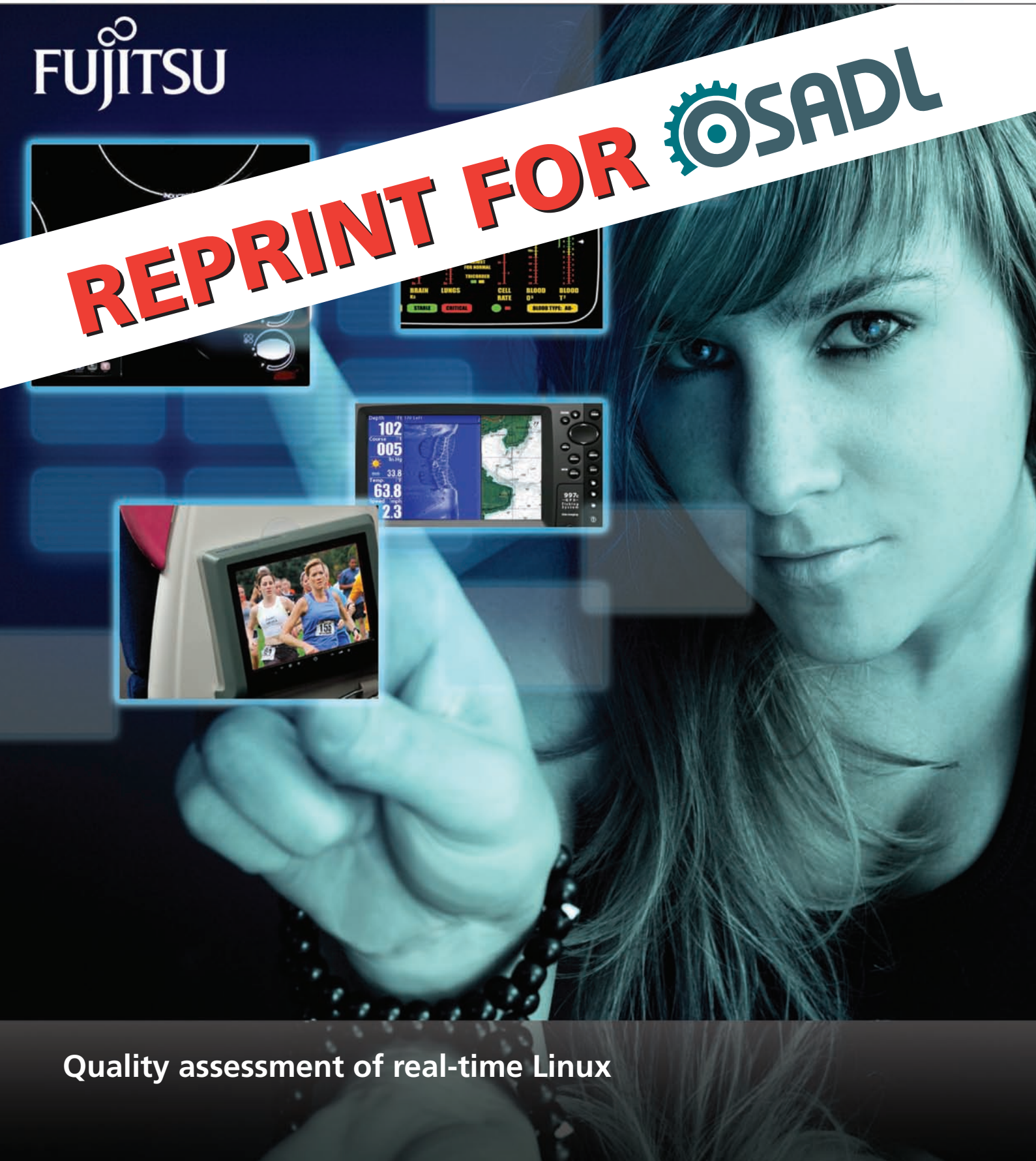
boards & solutions + ECE

Combined Print Magazine for the European Embedded Market

Sept 06/11

REPRINT FOR OSADL



Quality assessment of real-time Linux

Quality assessment of real-time Linux

By Carsten Emde, Open Source Automation Development Lab (OSADL)
and Thomas Gleixner, Linutronix

This article describes in detail the real-time capabilities and built-in testing features of Linux. It also explains the advantages that OSADL brings to the Linux community by testing real-time Linux exhaustively in the OSADL QA farm.

Figure 1. Each test rack in the OSADL QA farm has eight power switches with a network connection, eight network switches (10/100/1000 Mbit/s) with port mirroring, and eight serial network adapters.

Since the highly positive reception for furthering the real-time capabilities of the Linux kernel at the 2006 Kernel Summit in Ottawa, mainline Linux maintainer Linus Torvalds has regularly been accepting related patches. Over the last five years and 22 versions from version 2.6.18 to 2.6.40, many of the real-time components have been included in the Linux kernel. There will, by the way, be no version 2.6.40 since it has been renumbered to version 3.0. The remaining patches that are not yet available in mainline Linux have been combined in an archive and need to be included before the configuration variable CONFIG_PREEMPT_RT_FULL becomes available and can be set to enable production of a realtime Linux kernel. The archive is called PREEMPT_RT patch, and the Linux version is called PREEMPT_RT Linux after the former name of the configuration variable.

The PREEMPT_RT patch will turn Linux into a classic real-time operating system with a POSIX programming interface. The real-time tasks or threads will not have to run in kernel space or in the context of the interrupt service handler; they can instead run as normal user programs in user space with a given priority. A total of 99 priority levels are available where a value of 1 represents the lowest, and a value of 99 represents the highest priority. To set the priority of a process, the POSIX call sched_setparam() is available that can also be used to simultaneously select the desired scheduler policy. In case of a multiprocessor system, it can become necessary to assign a specific processor or group of processors to a process. This is done using the POSIX call sched_setaffinity(). Apart from the calls mlock() and mlockall() that are used to prevent the memory of a process from being swapped to disk (which is unacceptable for a real-time process), no other calls to the operating system are necessary to define the real-time characteristics of an application.

The real-time capability of an operating system can be measured both externally and internally. For external measurement, a voltage change is triggered at the input of an interrupt-capable I/O controller, and the time is measured from this voltage change until a waiting user space program is scheduled and continues execution. In a simplified form, this measurement can also be executed using the timer interrupt and waiting for the expiration of a sleep() call. In any case, one sample is obtained per interrupt; that is, the duration from an external event to the execution of the user space program. This time is called the preemption latency or the total latency. Normally, the individual samples are entered in a histogram with a range width of 1 µs. To graphically represent this, it is recommended to use a linear x-scale with the latency values and a logarithmic y-scale with the sample frequencies per range. In this way, it is also possible to visualize measurements that occur only very rarely. This is necessary, since the highest measured value is the most important part of the measurement, and may occur very rarely, sometimes only once. In addition, it is important to take a very high number of individual measurements in order for the result to be sufficiently reliable – at least 100 million or preferably 1 billion measurements. At a measuring frequency of 5 kHz, obtaining 100 million measurements takes about 5 hours and 33 minutes. The worst-case latency ever measured represents the characteristic property of a real-time system.

With internal measurements, the Linux kernel (as part of the trace subsystem) records the current time at the beginning of interrupt processing and immediately before the user space regains control. The differences between these two time stamps are then saved in a histogram that can be read out using the sysfs virtual file system. In this internal measurement, several stations are run through, the delays of which also are calculated and recorded individually: delay until starting the timer interrupt, delay from the end of the timer interrupt to enqueuing the user space program, delay between enqueuing the user space program and the actual context change, and delay between

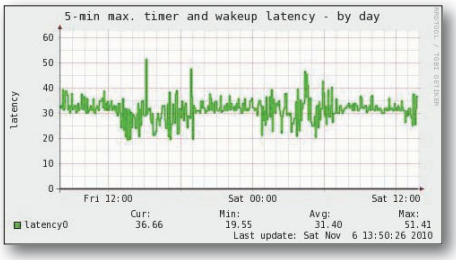


Figure 2. The example shows a 30-hour section of a measurement, but similar plots can be generated for a month or even for a year.

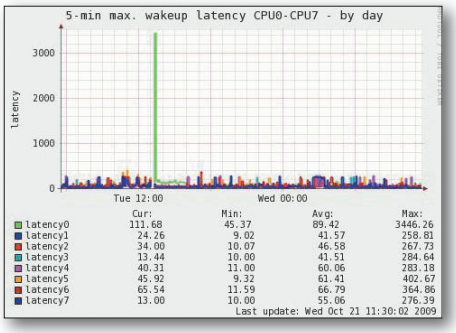


Figure 3. Example of a latency that was triggered for calibration purposes with a driver that intentionally generates latencies in a system.

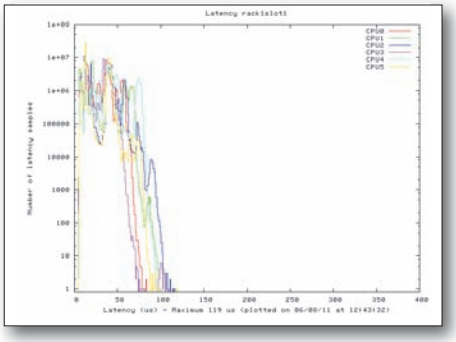


Figure 4. Latency measurement on a 6-core processor. The plots are based on 100 million individual measurements per CPU core.

the context change and returning from system call. In addition, sums of intermediate periods of time such as the combined timer delay and wakeup latency are calculated and saved in histograms. The sysfs virtual file system contains a mechanism to reset the histograms. This can be used to let the measurement cover only selected periods of time. Last not least, characteristic data of the process for which the maximum latency was measured are saved and made available through the virtual file system. These data are the name, ID and priority of the process. In addition, such data are also saved of the process from which the switch was made to the delayed process. This is done to obtain information about why a longer latency occurred in an individual case.

In contrast to proprietary operating systems, the technical data of Linux, and in particular the real-time capability of specific processor families and processors, are not presented in glossy sales brochures; rather, each user is responsible for dealing with the data himself. In addition, and this is even more important, everyone must guarantee that these data are maintained. Since this is difficult or even impossible for individual users such as an average machine company, the Open Source Automation Development Lab was founded in 2005 which is tasked with functioning like a Linux company. Its responsibilities include providing the real-time capability data of the Linux kernel and guaranteeing that these data are maintained. OSADL is a registered cooperative and financed by the contributions of its member companies. All users of open source software in the fields of machine construction and automation and in general all companies that use such software in industrial environments and pass it on are invited to join OSADL. To generate relevant measurements, the OSADL test center or OSADL QA farm was set up. The systems to be tested are placed on individual test tablets, and eight systems form a test rack as shown in figure 1. Each tablet is equipped with uniform plug-in connections for power (220 V, 50 Hz), a network connection (RJ45), and serial connection (RJ45). Each test rack has eight power switches with a network connection, eight network switches (10/100/1000 Mbit/s) with port mirroring, and eight serial network adapters. The tablets can be easily removed and temporarily operated in a workplace, e.g. for error analysis. Depending on the project definition, the test system can be released with the Linux kernel provided by the hardware manufacturer, or with the current Latest Stable real-time kernel released by OSADL.

All the systems undergo a 12-hour test cycle in which they are measured continuously – one-half of the time under continuous idle conditions, and one-half of the time under load. In the first two hours of the six-hour load period, timer interrupts are triggered at a frequency of 5 kHz as the only load. During the following four hours, the systems are additionally exposed to a specific load with memory allocations, network accesses and file system calls. For the cyclic timer interrupts, the test program cyclictest is used that is contained in the rt-tests package provided by many Linux distributions.

The processor families and processors listed in table 1 are tested in the OSADL QA farm (as of June 2011).

The continuous internal kernel latency measurement is reset after every five minutes, and the measurements that were taken up to that time are saved. This yields a continuous flow of consecutive five-minute maxima that can be displayed as a data curve over time. The example in figure 2 shows a 30-hour section of a measurement, but similar plots can be generated for a month or even for a year. The maximum value arising during the entire measuring period which is displayed below the curve on the right corresponds to the measurement total and is termed the worst-case latency. In this mode of representation, we can easily see abnormally high latencies that have only occurred once.

Figure 3 shows an example of a latency that was triggered for calibration purposes with a driver that intentionally generates latencies in a system. In this case, only the first of the eight processor cores was affected by this particularly high latency. To generate latency plots, latency was measured twice every day for five hours and 33 minutes in the OSADL test center.

Figure 4 shows an example of such a measurement on a 6-core processor. The plots are based on 100 million individual measurements per CPU core. In evaluating the real-time capability, the bottom right section of the latency plot is particularly important. The steeper the curve the more reliable the results. As already mentioned, time-related information is lost in latency histograms. In addition, it is desirable to record and depict longer periods of time and a larger number of measured values. For this reason, the latency plots recorded in the OSADL test center are also combined into a single figure, and several hundred plots can be easily portrayed sequentially. The time axis runs backwards, i.e. the most recent measurement is at the front of the drawing. The scale selected for the individual latency plot is left logarithmic so that infrequent events still can be identified.

Figure 5 shows such a long-term latency plot comprising more than 20 billion individual measurements. One elevated (in this case known) latency value is clearly identifiable. Latency plots that are repeatedly recorded over long periods of time are also important in the analysis of the real-time behavior with reference to changes in the run-time conditions. These include the intentional removal of inappropriate hardware components and specifically parametrizing the kernel to modifying dynamic kernel variables. Figure 6 shows the effect of the kernel parameter processor.max_cstate=1 on a system's real-time behavior. Apparently, this processor is a particularly energy-saving version in which energy is saved by largely turning off the processor when idling. This is conversely disadvantageous for the real-time behavior since, given an asynchronous external event, the processor has to restart which apparently takes approximately 100µs in this instance. If a C state higher than 1 is prevented by the cited kernel parameter, the energy savings are canceled, but the real-time behavior of the system improves due to the reduced latency. Whereas the effect described above would have been identifiable in individual latency plots, there are other

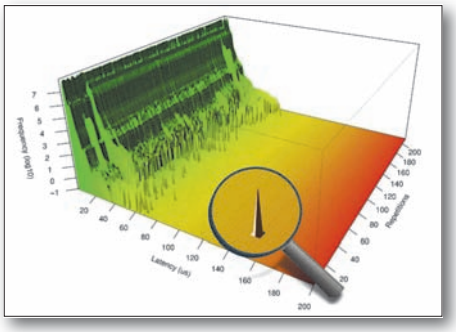


Figure 5. Long-term latency plot comprising more than 20 billion individual measurements. One elevated latency value of known origin is clearly identifiable.

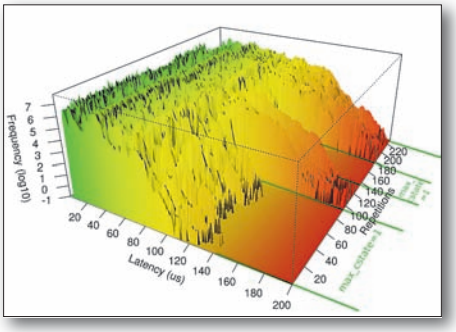


Figure 6. Effect of the kernel parameter processor.max_cstate=1 on a system real-time behavior

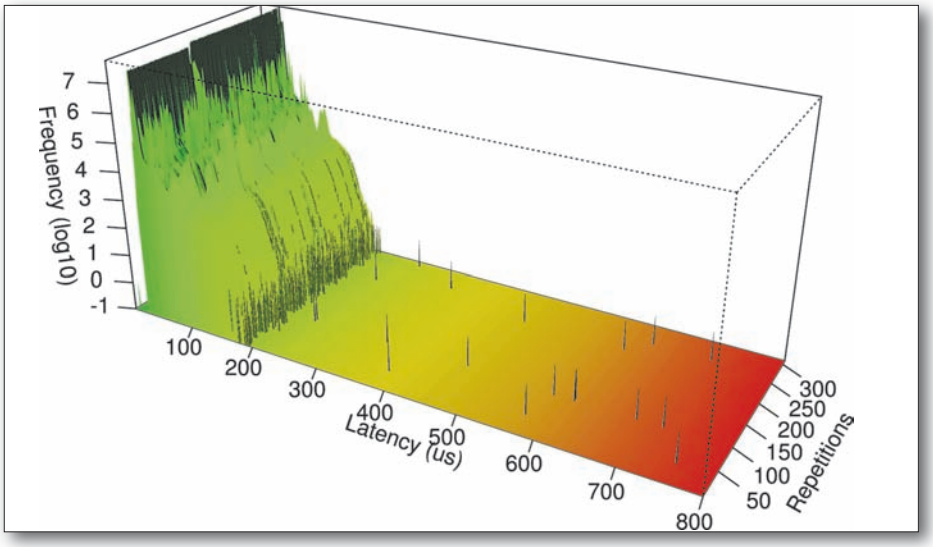


Figure 7. Advantage and the necessity of long-term measurements: The very rare yet important latency exceptions are only depicted in the extremely high number of individual measurements

phenomena that are only identifiable in a long-term depiction of a large number of repeatedly recorded latency plots.

The example in figure 7 is one of the cases that show the advantage and the necessity of long-term measurements: the very rare yet important latency exceptions are only depicted in the extremely high number of individual measurements. This is probably a very rare SMI (system management interrupt) or another design flaw. This finding has not been analyzed further since the relatively slow processor under test from the mid-1990s is no longer being manufactured, and is only running as a reference on the OSADL QA farm. The subsequent versions by the same manufacturer do not manifest this problem. In any case, it would be inadvisable to use this processor or chipset in a real-time project. With the exception of the repeatedly recorded latency plots that only OSADL member companies can access, the data and graphics shown here are available on the OSADL website for anyone, along with a great deal of additional information. The continuous latency measurements are updated every five minutes. The latency plots and profiles of all the systems are updated twice a day after the end of the cyclictest measuring phases. For this purpose, the OSADL test center is connected to the OSADL web server via a high-speed VPN connection. Among others, the profiles include the complete kernel configuration, kernel parameters and other information that is required to build identical systems at other places to challenge the results obtained at the OSADL QA farm. The URL of the OSADL test center is http://www.osadl.org/QA. ■

OSADL Open Source Automation Development Lab

OSADL eG
Aichhalder Str. 39
D-78713 Schramberg • Germany

Phone: +49(7422)515-8820
Fax.: +49(7422)515-8822
E-Mail: info@osadl.org

x86 processor family

AMD

K6 3D, @333 MHz, 32 bit
LX800 @500 MHz, 32 bit
Athlon XP 2000+, 32 bit
Athlon 64 2800+, 64 bit
G-Series T56N @1400 MHz, 64 bit
Phenom II X6 @3200 MHz, 64 bit

Intel

Pentium @133 MHz, 32 bit
Pentium II Klamath @233 MHz, 32 bit
Atom N270 @1600 MHz, 32 bit
Atom D510 @1667 MHz, 64 bit
Atom D525 @1800 MHz, 64 bit
Atom Z530 @1600 MHz, 32 bit (standard kernel as a reference)
Celeron M @1500 MHz, 32 bit
Pentium M dual-core @2300 MHz, 32 bit
Xeon (single socket) @2000 MHz, 32 bit
Xeon (dual socket) @2800 MHz, 32 bit
Core 2 Duo @2400 MHz, 64 bit
Core 2 Quad @2400 MHz, 32 bit
i7 Nehalem 975 @3333 MHz, 32 bit
i7 Gulftown X980 @3333 MHz, 64 bit
Core i3-2100T @2500 MHz, 64 bit
Core i7-2600K @3400 MHz, 64 bit

VIA

C3 Samuel 2 @533, 32 bit
C3 Nehemiah @533 MHz, 32 bit
C7 @1000 MHz, 32 bit

ARM processor family

Freescale

i.MX27 ARM9 @400 MHz, 32 bit
i.MX35 ARM11 @532 MHz, 32 bit

Marvell

SheevaPlug @1200 MHz, 32 bit

Texas Instruments

AM3517 @600 MHz, 32 bit
OMAP3525 ARM cortex-a8 @720 MHz, 32 bit
AM4430 @1000 MHz, 32 bit

PowerPC processor family

Freescale

MPC 5200 @396 MHz, 32 bit
MPC 5121 @400 MHz, 32 bit

MIPS processor family

ICT

Loongson 2F @800 MHz, 64 bit

Table 1. The processor families and processors tested in the OSADL QA farm