# **High Resolution Timers**

## **Contents**

* [1 Description](http://elinux.org/High_Resolution_Timers#Description)
* [2 Rationale](http://elinux.org/High_Resolution_Timers#Rationale)
* [3 Resources](http://elinux.org/High_Resolution_Timers#Resources)
  + [3.1 Projects](http://elinux.org/High_Resolution_Timers#Projects)
    - [3.1.1 hrtimers - Thomas Gleixner's patch](http://elinux.org/High_Resolution_Timers#hrtimers_-_Thomas_Gleixner.27s_patch)
    - [3.1.2 HRT - Geoge Anzinger's patch](http://elinux.org/High_Resolution_Timers#HRT_-_Geoge_Anzinger.27s_patch)
* [4 Downloads](http://elinux.org/High_Resolution_Timers#Downloads)
  + [4.1 Patch](http://elinux.org/High_Resolution_Timers#Patch)
* [5 Utility programs](http://elinux.org/High_Resolution_Timers#Utility_programs)
* [6 How To Use](http://elinux.org/High_Resolution_Timers#How_To_Use)
* [7 How to detect if your timer system supports high resolution](http://elinux.org/High_Resolution_Timers#How_to_detect_if_your_timer_system_supports_high_resolution)
* [8 How to validate](http://elinux.org/High_Resolution_Timers#How_to_validate)
* [9 Sample Results](http://elinux.org/High_Resolution_Timers#Sample_Results)
* [10 Case Study 1](http://elinux.org/High_Resolution_Timers#Case_Study_1)
* [11 Case Study 2](http://elinux.org/High_Resolution_Timers#Case_Study_2)
* [12 Status](http://elinux.org/High_Resolution_Timers#Status)
* [13 Future Work/Action Items](http://elinux.org/High_Resolution_Timers#Future_Work.2FAction_Items)
* [14 Old information (for 2.4 kernel)](http://elinux.org/High_Resolution_Timers#Old_information_.28for_2.4_kernel.29)

## **Description**

The objective of the high resolution timers project is to implement the POSIX 1003.1b Section 14 (Clocks and Timers) API in Linux. This includes support for high resolution timers - that is, timers with accuracy better than 1 jiffy.

When the project started, the POSIX clocks and timers APIs were not supported by Linux. Over time, the clocks and timers APIs have been adopted, and core infrastructure support for high resolution timers has been accepted into the mainline kernel (in 2.6.21). However, as of this writing, not all embedded platforms has support for high resolution timers, and even when support is present in the kernel code, it can be tricky to configure it for the kernel.

## **Rationale**

Currently, timers in Linux are only supported at a resolution of 1 jiffy. The length of a jiffy is dependent on the value of HZ in the Linux kernel, and is 1 millisecond on i386 and some other platforms, and 10 milliseconds on most embedded platforms.

Higher resolution timers are needed to allow the system to wake up and process data at more accurate intervals.

## **Resources**

### **Projects**

#### **hrtimers - Thomas Gleixner's patch**

One project to support high resolution timers is Thomas Gleixner's hrtimers.

Thomas gave a presentation at the Ottawa Linux Symposium, July 2006, presenting the current status of hrtimers. The presentation is here: [OLS hrtimers](https://docs.google.com/file/d/0BzuiPiVvL63cNTYwYWE1YTgtODFhMS00NzM1LTlkMTItYWVlNzU3MWQ1NzA5/edit?sort=name&layout=list&pli=1&pid=0BzuiPiVvL63cNzJlODhmYWEtYWY1MS00Yjc1LTg5YzUtODViMDUzOTZjNzUz&cindex=89#)

As of July 2006, "generic clock sources" was accepted into Linus' mainline kernel tree (2.6.18-rc??). This means it should be appear in the mainline 2.6.18 kernel version, when that is available. hrtimers should soon follow, likely appearing in 2.6.19.

In February of 2006, James Perkins of WindRiver wrote:

ktimers has been obsoleted by hrtimers, and the core of hrtimers was merged and is present in Linus' 2.6.16-rc2. hrtimers is used as the base for itimers, nanosleep, and posix-timers. hrtimers are well-described by Jonathan Corbet at<http://lwn.net/Articles/167897/>

Since only the core of hrtimers is in 2.6.16-rc2, the hrtimers generally use the system timer as their tick source and run at HZ. John Stultz' generalized time source code has not yet been merged. Thomas Gleixner is maintaining his git tree and has graciously published patches at<http://www.tglx.de/projects/hrtimers/> that include generalized clocksource, new timeofday patches, and get you the real "high resolution" timers for a subset of architectures.

High-res timers work is experimental and shifting and has been focusing on getting x86 working first, if this is adequate for you and you can use 2.6.16 kernels it's recommended, and let us all know of any problems or improvements. In contrast, the previous implementation that George Anzinger lead provides a fairly comprehensive set of functionality, back in the 2.6.8-2.6.10 era, but it isn't an active project at this time.

*Note that the current HRT maintainers objected to this characterization.*

#### **HRT - Geoge Anzinger's patch**

Prior to hrtimers, the main patch which provided high resolution timers was George Anzinger's patch.The official HRT site for this patch is at:

* [high-res-timers](http://sourceforge.net/projects/high-res-timers/)

## **Downloads**

### **Patch**

* See [Patch Archive](http://elinux.org/Patch_Archive)
* Tom Rini has posted some patches for earlier 2.6 kernels at:
  + [trini patches](http://source.mvista.com/%7Etrini/hrt/hrt_07Dec2004_001_2.6.10-rc3.patch)

## **Utility programs**

## **How To Use**

In order to use high resolution timers, you need to verify that the kernel has support for this feature for your target processor (and board). Also, you need to configure support for it in the Linux kernel.

Set CONFIG\_HIGH\_RES\_TIMERS=y in your kernel config.

Compile your kernel and install it on your target board.

To use the Posix Timers API, see this online resource [[1]](http://www.opengroup.org/onlinepubs/009695399/basedefs/time.h.html)

## **How to detect if your timer system supports high resolution**

Here are several ways you can identify if your system supports high resolution timers.

* Examine kernel startup messages

Watch the kernel boot messages, or use dmesg. If the kernel successfully turns on the high resolution timer feature, it will print the message "Switched to high resolution mode on CPU0" (or something similar) during startup.

* Examine /proc/timer\_list

You can also examine the timer\_list, and see whether specific clocks are listed as supporting high resolution. Here is a dump of /proc/timer\_list on an [OSK](http://elinux.org/OSK) (ARM-based) development board, showing the clocks configured for high resolution.

* + cat /proc/timer\_list

Timer List Version: v0.3  
HRTIMER\_MAX\_CLOCK\_BASES: 2  
now at 294115539550 nsecs  
  
cpu: 0  
 clock 0:  
 .index: 0  
 .resolution: 1 nsecs  
 .get\_time: ktime\_get\_real  
 .offset: 0 nsecs  
active timers:  
 clock 1:  
 .index: 1  
 .resolution: 1 nsecs  
 .get\_time: ktime\_get  
 .offset: 0 nsecs  
active timers:  
 #0: <c1e39e38>, tick\_sched\_timer, S:01, tick\_nohz\_restart\_sched\_tick, swapper/0  
 # expires at 294117187500 nsecs [in 1647950 nsecs]  
 #1: <c1e39e38>, it\_real\_fn, S:01, do\_setitimer, syslogd/796  
 # expires at 1207087219238 nsecs [in 912971679688 nsecs]  
 .expires\_next : 294117187500 nsecs  
 .hres\_active : 1  
 .nr\_events : 1635  
 .nohz\_mode : 2  
 .idle\_tick : 294078125000 nsecs  
 .tick\_stopped : 0  
 .idle\_jiffies : 4294966537  
 .idle\_calls : 2798  
 .idle\_sleeps : 1031  
 .idle\_entrytime : 294105407714 nsecs  
 .idle\_sleeptime : 286135498094 nsecs  
 .last\_jiffies : 4294966541  
 .next\_jiffies : 4294966555  
 .idle\_expires : 294179687500 nsecs  
jiffies: 4294966542  
  
  
Tick Device: mode: 1  
Clock Event Device: 32k-timer  
 max\_delta\_ns: 2147483647  
 min\_delta\_ns: 30517  
 mult: 140737  
 shift: 32  
 mode: 3  
 next\_event: 294117187500 nsecs  
 set\_next\_event: omap\_32k\_timer\_set\_next\_event  
 set\_mode: omap\_32k\_timer\_set\_mode  
 event\_handler: hrtimer\_interrupt

Here are some things to check:

1. Check the resolution reported for your clocks. If your clock supports high resolution, it will have a .resolution value of 1 nsecs. If it does not, then it will have a .resolution value that equals the number of nanoseconds in a jiffy (usually 10000 nsecs, on embedded platforms).

2. Check the event\_handler for the Tick Device. If the event handlers is 'hrtimer\_interrupt' then the clock is set up for high resolution handling. If the event handler is 'tick\_handle\_periodic', then the device is set up for regular tick-based handling.

3. Check the list of timers, and see if the attribute .hres\_active has a value of 1. If so, then the high resolution timer feature is active.

* Run a test program

You can run a small test program, and actually measure that the timers are returning in less than the period of a jiffy. If they are, this is the most definitive proof that your kernel supports high resolution timers. One example program you can try is [cyclictest](http://rt.wiki.kernel.org/index.php/Cyclictest). Here is a sample command line which will test timers using nanosleep:

* + cyclictest -n -p 80 -i 500 -l 5000

This does a test of clock\_nanosleep, with priority 80, at 500 microsecond intervals, running the 5000 iterations of the test.

## **How to validate**

See above with regard to cyclictest

## **Sample Results**

[Examples of use with measurement of the effects.]

## **Case Study 1**

## **Case Study 2**

## **Status**

* Status: implemented
* Architecture Support:

(for each arch, one of: unknown, patches apply, compiles, runs, works, accepted)

* + i386: works
  + ARM: unknown
  + PPC: works
  + MIPS: unknown
  + SH: unknown

## **Future Work/Action Items**

Here is a list of things that could be worked on for this feature:

* Documentation
* Testing

## **Old information (for 2.4 kernel)**

The High Resolution Timers system allows a user space program to be wake up from a timer event with better accuracy, when using the POSIX timer APIs. Without this system, the best accuracy that can be obtained for timer events is 1 jiffy. This depends on the setting of HZ in the kernel. In the 2.4 kernel, HZ was set to 100, which means that the best accuracy you could get on a timer wakeup in user space was 10 milliseconds.

Put differently, if you asked for a timer event in 500 microseconds, you would wake up in 10 milliseconds (at least).

To support this feature on a particular board, you have to add a kernel driver that uses a timer on the system and supports the interface documented in: include/linux/hrtime.h (in the CELF tree) Additional documentation about this feature is available in Documentation/high-res-timers/

Patches for high-res timers were first presented at the time of kernel version 2.5.47, in November, 2002. See early patches