EXHIBIT

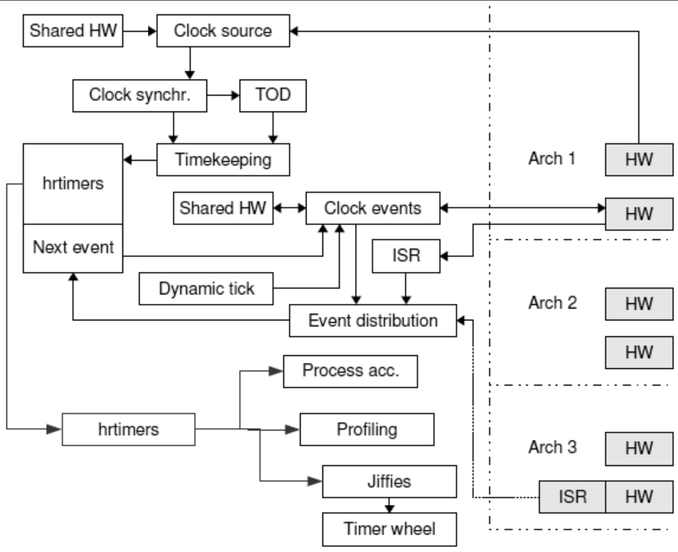
J

<u>‘460 Patent Infringement Contentions</u>

Motorola’s infringing products (“Accused Devices”) include mobile devices, such as smartphones, associated software, and components thereof. The Accused Devices include Motorola’s Android based phones which include, but are not limited to, the Motorola Droid X, Droid 2, Droid 2 Global, Cliq 2, Defy, Bravo, Droid Pro, Droid 2 R2-D2, Droid X 2, Charm, Droid, Flipside, Flipout, Atrix, Droid Bionic, Xoom, Devour A555, Backflip, Cliq/Dext, Cliq XT/Quench, Citrus, Spice, i1 and other Motorola Android based phones incorporating hardware and/or software that is substantially similar. The figures and illustrations in the infringement chart below display exemplary devices.

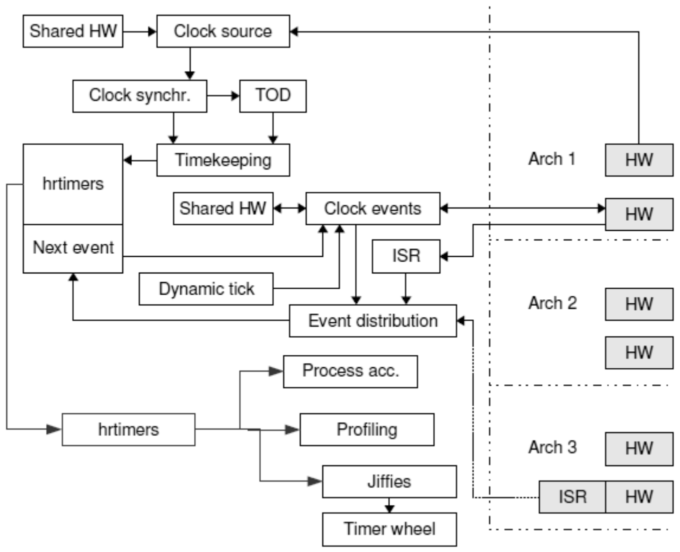
U.S. Patent No. 7,383,460 (‘460 Patent)	**Accused Devices**
7. A system to configure a timer in a computing device, the system comprising:	**<u>Each Accused Device provides a system to configure a timer in a computing device</u>** To the extent that this claim's preamble is construed as limiting, the Motorola Droid X (hereinafter, "Droid X") is a "smartphone" computing device that runs the Android operating system. (*See* http://www.motorola.com/Consumers/US-EN/Consumer-Product-and-Services/Mobile-Phones/ci.Motorola-DROID-X-US-EN.alt, last visited April 12, 2011.) The Droid X operates to configure a timer. By way of example, applications provided on the Droid X use the Android MediaPlayer class. (*See* http://developer.android.com/reference/android/media/MediaPlayer.html, last visited April 12, 2011.) The Android MediaPlayer configures a timer by use of the *hrtimer* component, which is described in further detail below.
a timer substantially guaranteed to expire at a time certain;	**<u>Each Accused Device provides a timer substantially guaranteed to expire at a time certain</u>** By way of example, the Droid X includes a Texas Instruments Open Multimedia Application Platform ("OMAP") processor. (*See http://news.cnet.com/8301-13924_3-20008725-64.html*, last visited April 12, 2011.) OMAP includes timers substantially guaranteed to expire at a time certain. By way of example, OMAP includes 11 general purpose timers, 2 watchdog timers, and a synchronized timer. (*See Texas Instruments Technical*

U.S. Patent No. 7,383,460 (‘460 Patent)	**Accused Devices**
	Reference Manual, OMAP36xx Multimedia Device, Silicon Revision 1.x, Version Q, Public Version, at 2702) (available at http://focus.ti.com/pdfs/wtbu/OMAP36xx_ES1.x_PUBLIC_TRM_vQ.zip, last visited April 12, 2011) ("OMAP documentation.") These timers are registered with a high resolution *hrtimer* component because they are substantially guaranteed to expire at a time certain. Other Accused Devices use other types of processors. By way of example, some Accused Devices — such as the Motorola Xoom — make use of an Nvidia Tegra processor. (*See* http://developer.motorola.com/products/xoom/, last visited April 13, 2011.) Some Accused Devices — such as the Motorola Cliq — make use of a Qualcomm processor. (*See* http://developer.motorola.com/products/cliq/, last visited April 13, 2011.) Other Accused Devices — such as the Motorola i1 — make use of a Freescale Zeus processor. (*See* http://developer.motorola.com/products/i1/.) The processors of all these devices include a timer which registers with a high resolution *hrtimer* component and is substantially guaranteed to expire at a time certain, as shown in further detail below.
a hardware-independent interface to the timer,	**Each Accused Device provides a hardware-independent interface to the timer** The Android operating system provides a hardware-independent component called *hrtimer*. (*See, e.g.*, *kernel/hrtimer.c*; *include/linux/hrtimer.h*) The Android documentation references the following block diagram and indicates that it illustrates *hrtimer*:

U.S. Patent No. 7,383,460 ('460 Patent)	Accused Devices
	 Shared HW Clock source Clock synchr. TOD Timekeeping hrtimers Next event Shared HW Clock events ISR Dynamic tick Event distribution Process acc. hrtimers Profiling Jiffies Timer wheel Arch 1 HW HW Arch 2 HW HW Arch 3 HW ISR HW (*See* kernel/documentation/timers/highres.txt) (citing Thomas Gleixner and Douglas Neihaus presentation, "hrtimers and beyond – transformation of the Linux time(r) system," OLS 2006, at slide 22, currently available at http://www.kernel.org/pub/linux/kernel/people/tglx/hrtimers/ols2006-hrtimers.pdf, last visited April 12, 2011) (*see also* slides 15, 18, 20.) This figure depicts several hardware architectures on the right-hand side that can interface with a common *hrtimer* component, showing that *hrtimer* is hardware-independent. In addition, as will be described in more detail below, *hrtimer* includes program code that enables a variety of distinct types of hardware-dependent components to register with the *hrtimer* component, further showing that *hrtimer* is hardware-independent. The Accused Devices provide access to *hrtimer* through hardware-independent interfaces to the timer. By way of example, the Accused Devices provide timer access for applications residing in the user mode through hardware-independent interfaces such as *nanosleep* (*see bionic/libc/include/sys/linux-unistd.h*; *see also include/linux/syscalls.h*.) and *setitimer* (*see platform/bionic/libc/include/sys/linux-unistd.h*; *see also include/linux/syscalls.h.)* Likewise, the Accused Devices provide timer access for applications residing in the kernel mode through *hrtimer* routines such as *hrtimer_start(), hrtimer_restart(), hrtimer_start_expires()*, and *hrtimer_start_range_ns()*. (*See kernel/hrtimer.c*; *include/linux/hrtimer.h*.)

U.S. Patent No. 7,383,460 (‘460 Patent)	**Accused Devices**
	The Android documentation indicates that applications residing in the user mode or applications residing in the kernel mode use these interfaces to set the timer: "[t]he primary users of precision timers are user-space applications that utilize nanosleep, posix-timers and itimer interfaces. Also, in-kernel users like drivers and subsystems which require precise timed events can benefit from the availability of a separate high-resolution timer subsystem as well." (*See kernel/documentation/timers/hrtimers.txt.*) The manner by which applications interoperate with the *hrtimer* component and by which the *hrtimer* component interoperates with hardware-dependent components is described in more detail below.
wherein the hardware-independent interface is a kernel mode routine having a set interrupt timer application programming interface (API) for receiving parameters associated with a request from the application to set the timer, and	**In Each Accused Device, the hardware-independent interface is a kernel mode routine having a set interrupt timer application programming interface (API) for receiving parameters associated with a request from the application to set the timer** The Android documentation states that *nanosleep*, *setitimer*, and *hrtimer* operate in the kernel mode. The *hrtimer* component is implemented by program logic located in the file *hrtimer.c,* which is a part of the Android operating system's kernel. (*See kernel/hrtimer.c.*) The Android states that the *hrtimer* component is currently used for "precise in-kernel timing." (*See kernel/hrtimer.c.*) The Android documentation further states that *hrtimer* provides "kernel logic" that works at a nanosecond-level resolution and provides a simplification of "timing related kernel code." (*See kernel/documentation/hrtimers.txt.*) The Android documentation also states that "The hrtimer patch converts the following kernel functionality to use hrtimers: nanosleep[,] itimers[,] posix-timers[.]" (*See kernel/documentation/timers/hrtimers.txt.*) The hardware-independent interfaces have a set interrupt timer API for receiving parameters associated with a request to set the timer. The *nanosleep* routine receives such parameters from applications residing in user mode, including a parameter providing an indication of an expiry interval for the timer. (*See bionic/libc/include/sys/linux-unistd.h*); (*see also include/linux/syscalls.h.*) The *setitimer* routine, as another

U.S. Patent No. 7,383,460 (‘460 Patent)	**Accused Devices**
	example, receives such parameters from applications residing in user mode, including a parameter providing an indication of an expiry interval for the timer. (*See platform/bionic/libc/include/sys/linux-unistd.h*); (*see also include/linux/syscalls.h.)* Routines provided by the *hrtimer* component receive such parameters from applications residing in kernel mode. For example, *hrtimer_start_expires* (*see include/linux/hrtimer.h*) and *hrtimer_start_range_ns()* (*see kernel/hrtimer.c*) receive parameters associated with requests from *nanosleep*, including a parameter providing an indication of an expiry interval. Also by way of example, *hrtimer_start* (*see kernel/hrtimer.c*) and *hrtimer_restart* (*see include/linux/hrtimer.h*) receive parameters associated with requests from *setitimer*, including a parameter providing an indication of an expiry interval.
validating the request,	**<u>Each Accused Device provides for validating the request</u>** By way of example, a request passed to *nanosleep* is validated using the *timespec_valid* function. (*See kernel/hrtimer.c*.) By way of another example, a request passed to *setitimer* is validated using the *timeval_valid* function. (*See kernel/itimer.c*.) As yet another example, the *hrtimer* component validates requests through program logic in the routine *clockevents_program_event()*. This routine computes a *delta* value based on the requested expiry time and performs comparisons on *delta* in order to validate the requested expiry interval. (*See kernel/time/clockevents.c*.)
wherein validating the request includes validating the parameters by the hardware-independent interface;	**<u>In each Accused Device, validating the request includes validating the parameters by the hardware-independent interface</u>** Continuing with the examples discussed above, *timespec_valid* (*see kernel/hrtimer.c*), *timeval_valid* (*see kernel/itimer.c*), and *clockevents_program_event()* (*kernel/time/clockevents.c*) are all hardware-independent. This program logic is used regardless of the hardware timer that the *hrtimer* component interoperates with.

U.S. Patent No. 7,383,460 (‘460 Patent)	Accused Devices
a hardware-dependent interface to the timer;	<u>**Each Accused Device provides a hardware-dependent interface to the timer**</u> The Android documentation references the following block diagram and states that it illustrates the *hrtimer* component:  Shared HW Clock source Clock synchr. TOD Timekeeping hrtimers Next event Shared HW Clock events ISR Dynamic tick Event distribution Process acc. hrtimers Profiling Jiffies Timer wheel Arch 1 HW HW Arch 2 HW HW Arch 3 HW ISR HW (*See* kernel/documentation/timers/highres.txt) (citing Thomas Gleixner and Douglas Neihaus presentation, "hrtimers and beyond – transformation of the Linux time(r) system," OLS 2006, at slide 22, currently available at http://www.kernel.org/pub/linux/kernel/people/tglx/hrtimers/ols2006-hrtimers.pdf, last visited March 30, 2011) (*see also* slides 15, 18, 20.) This figure depicts several hardware architectures on the right-hand side and shows that each must interoperate with hardware-dependent components. The kernel directory */arch/* includes hardware-dependent components of various types. Interfaces by which the hardware-independent *hrtimer* component interoperates with hardware-dependent components will depend on the hardware being used. By way of example, exemplary Accused Devices using a TI OMAP processor (such as Droid X) include a hardware-dependent interface. (*See, e.g.*, */arch/arm/mach-omap2/timer-gp.c*.) By way of another example, exemplary Accused Devices using an Nvidia Tegra processor (such as Xoom) include a hardware-

U.S. Patent No. 7,383,460 ('460 Patent)	Accused Devices
	dependent interface. (*See, e.g., arch/arm/kernel/smp_twd.c*.) By way of another example, exemplary Accused Devices using a Qualcomm MSM processor (such as Cliq) include a hardware-independent interface. (*See, e.g., arch/arm/mach-msm/timer.c*.) Accused Devices using other processors (e.g., a Freescale Zeus processor as used by i1) will likewise include a hardware-dependent interface.
and a processor in which the hardware-independent interface operates to validate a request from an application to set the timer	**<u>Each Accused Device provides a processor in which the hardware-independent interface operates to validate a request from an application to set the timer</u>** As discussed above, the Accused Devices include processors, such as TI OMAP processors, Nvidia Tegra processors, Qualcomm processors, and Freescale Zeus processors. These processors execute the program logic described above to validate a request from an application to set a timer.
and to relay the validated request to the hardware-dependent process, and	**<u>Each Accused Device provides for relaying the validated request to the hardware-dependent process</u>** Hardware-specific information is registered with hardware-independent components using the data structure *clock_event_device*. (*See include/clockchips.h*.) The *set_next_event* member of that data structure provides an indication of a hardware-dependent process to which the hardware-independent component should relay the request. At those times when it is appropriate to set the timer, the hardware-independent *hrtimer* component will relay the request to the hardware-dependent process. By way of example, in exemplary Accused Devices using a TI OMAP processor (such as Droid X) the request is relayed to the hardware-dependent process *omap2_gp_timer_set_next_event()*. (*See /arch/arm/mach-omap2/timer-gp.c*.) By way of another example, in exemplary Accused Devices using an Nvidia Tegra processor (such as Motorola Xoom), the request is relayed to the hardware-dependent process *twd_set_next_event()*. (*See arch/arm/kernel/smp_twd.c*.)

U.S. Patent No. 7,383,460 (‘460 Patent)	**Accused Devices**
	By way of another example, in exemplary Accused Devices using a Qualcomm MSM processor (such as Cliq X), the request is relayed to the hardware-dependent process *msm_timer_set_next_event()*. (*See arch/arm/mach-msm/timer.c*.) In Accused Devices using other processors (e.g., a Freescale Zeus processor as used by i1), the request is likewise relayed to hardware-dependent processes (*e.g.*, *clkev_set_next_event()*).
further in which the hardware-dependent interface operates to set the timer to expire in accordance with the validated request and	**<u>In each Accused Device, the hardware-dependent interface operates to set the timer to expire in accordance with the validated request</u>** By way of example, in exemplary Accused Devices using a TI OMAP processor (such as Droid X) the hardware-dependent interface operates to set the timer through functionality set forth in */arch/arm/mach-omap2/timer-gp.c* and *arm/plat-omap/dmtimer.c*. By way of example, in exemplary Accused Devices using a Nvidia Tegra processor (such as Motorola Xoom) the hardware-dependent interface operates to set the timer through functionality set forth in *arch/arm/kernel/smp_twd.c* and *arch/arm/include/asm/io.h*. By way of example, in exemplary Accused Devices using a Qualcomm MSM processor (such as Cliq X) the hardware-dependent interface operates to set the timer through functionality set forth in *arch/arm/mach-msm/timer.c* and *arch/arm/include/asm/io.h*. In Accused Devices using other processors (e.g., a Freescale Zeus processor as used in i1), the hardware-dependent interface likewise operates to set the timer.
to execute a timer interrupt service routine upon expiration of the timer.	**<u>Each Accused Device executes a timer interrupt service routine upon expiration of the timer</u>** Upon expiration of the timer, the Accused Devices execute a timer interrupt service routine. By way of example, in exemplary Accused Devices running a TI

U.S. Patent No. 7,383,460 (‘460 Patent)	**Accused Devices**
	OMAP processor (such as the Droid X), handlers process the interrupt generated by the timer as to invoke the *hrtimer_interrupt()* service routine (*see kernel/hrtimer.c*) via a call to *omap2_gp_timer_interrupt()* (*see /arch/arm/mach-omap2/timer-gp.c*.) Exemplary Accused Devices running an Nvidia Tegra processor, Qualcomm MSM processor, or other processors (e.g., a Freescale Zeus processor) similarly handle the interrupt generated by the timer as to invoke the *hrtimer_interrupt()* service routine.
8. The system of claim 7, wherein the timer is a high precision event timer (HPET).	**<u>In each Accused Device, the timer is a high precision event timer</u>** The Accused Devices use timers that are high resolution timers and that interoperate with the high resolution *hrtimer* component, as discussed above. The timers operate at a nanosecond-based resolution. (*See kernel/documentation/timers/hrtimers.txt*.)
9. The system of claim 8, wherein the hardware-dependent interface operates to set the timer by writing an actual time at which the HPET should expire to a comparator register associated with the HPET, the actual [time] being determined by the hardware-dependent interface in accordance with the validated request.	**<u>In each Accused Device, the hardware-dependent interface operates to set the timer by writing an actual time at which the HPET should expire to a comparator register associated with the HPET, the actual tune being determined by the hardware-dependent interface in accordance with the validated request</u>** In the Accused Devices, the hardware-dependent interface operates to set the timer by writing an actual time at which the timer should expire to a register associated with the timer. The actual time is determined by functions in the hardware-dependent interface. Specifically, the *hrtimer* component adds the expiry interval specified by the application to a value representing the current time. (*See, e.g., ktime_add_safe()*in *__hrtimer_start_range_ns* in *kernel/hrtimer.c*.) The resulting value is passed to hardware-dependent components, which convert it from nanosecond-based units into units representing the actual clock cycle value at which the timer should expire. By way of example, exemplary Accused Devices running a TI OMAP processor (such as Droid X) covert the value into units representing the actual clock cycle value. (*See, e.g., kernel/timer-gp.c* and *arch/arm/plat-omap/dmtimer.c*.)

U.S. Patent No. 7,383,460 (‘460 Patent)	**Accused Devices**
	By way of another example, exemplary Accused Devices running an Nvidia Tegra processor (such as Motorola Xoom) covert the value into units representing the actual clock cycle value. (*See, e.g.*, *arch/arm/kernel/smp_twd.c* and *arch/arm/include/asm/io.h*.) By way of another example, exemplary Accused Devices running a Qualcomm MSM processor (such as Cliq X) convert the value into units representing the actual clock cycle value. (*See, e.g.*, *arch/arm/mach-msm/timer.c* and *arch/arm/include/asm/io.h*.) Exemplary Accused Devices using other processors (e.g., a Freescale Zeus processor) likewise convert the value into units representing the actual clock cycle value. The time at which the timer should expire is written to a comparator register. By way of example, according to the OMAP documentation, GP timers can operate in "compare mode." "When the compare enable register GPTi.TCLR[6] CE bit is set to 1, the timer value (GPTi.TCRR[31:0] TIMER_COUNTER field) is continuously compared to the value held in the timer match register (GPTi.TMAR). The GPTi.TMAR[31:0] COMPARE_VALUE value can be loaded at any time (timer counting or stopped). When the GPTi.TCRR and the GPTi.TMAR values match, an interrupt is issued, if the GPTi.TIER[0] MAT_IT_ENA bit is set." (*OMAP documentation* at 2718.)
10. The system of claim 7, wherein the parameters specify an interval representing a period of time after which the hardware interrupt timer is requested to expire, and wherein the processor operates to validate the request by determining that the interval is of substantially sufficient duration to set the timer.	**In each Accused Device, the parameters specify an interval representing a period of time after which the hardware interrupt timer is requested to expire, and wherein the processor operates to validate the request by determining that the interval is of substantially sufficient duration to set the timer** By way of example, the *rqtp* parameter passed into *nanosleep* specifies an interval representing a period of time after which the hardware interrupt timer is requested to expire. (*See bionic/libc/include/sys/linux-unistd.h*); (*see also include/linux/syscalls.h*.) Also by way of example, the *itimerval* parameter passed into *setitimer* includes the *it_value* parameter which specifies an interval representing a period of time after which the hardware interrupt timer is requested to expire. (*See platform/bionic/libc/include/sys/linux-unistd.h*); (*see also*

U.S. Patent No. 7,383,460 (‘460 Patent)	**Accused Devices**
	include/linux/syscalls.h.) The parameters passed into routines provided by the *hrtimer* component also include a representation of a period of time after which the hardware interrupt timer is requested to expire. (*See, e.g.*, *timer* and *mode* parameter passed into *hrtimer_start_expires()*; *timer*, *tim*, and *mode* parameters passed into *hrtimer_start_range_ns()*; and the *timer*, *tim*, and *mode* parameters passed into *hrtimer_start()*. (*See kernel/hrtimer.c*; *include/linux/hrtimer.h*.) The Accused Devices' processors operate to validate the request by determining that the interval is of substantially sufficient duration to set the timer. By way of example, the *timespec_valid* function determines that the expiry interval is of substantially sufficient duration. (*See kernel/hrtimer.c*.) By way of another example, the *timeval_valid* function determines whether the expiry interval is of substantially sufficient duration. (*See kernel/itimer.c*.) By way of further example, *clockevents_program_event()* checks that the expiry interval is of substantially sufficient duration by computing a *delta* value based in part on the requested expiry time and checking that the *delta* value is greater than 0. This function further checks that the interval is of substantially sufficient duration by checking that *delta* is greater than a constant *min_delta_ns*. (*See kernel/time/clockevents.c*.)
11. The system of claim 7, wherein the parameters specify a mode in which the timer is requested to operate, and wherein the processor operates to validate the request by determining that the mode is one of periodic and aperiodic.	**<u>In each Accused Device, the parameters specify a mode in which the timer is requested to operate, and the processor operates to validate the request by determining that the mode is one of periodic and aperiodic</u>** By way of example, the *setitimer* routine accepts the parameter *itimerval *value*, which includes the parameters *timeval it_value* and *timeval it_interval*. (*See platform/bionic/libc/include/sys/linux-unistd.h*; *include/linux/syscalls.h*.) These parameters specify that the mode is one of periodic and aperiodic based on whether the *it_interval* parameter is greater than 0. If *it_interval* is greater than 0, the

U.S. Patent No. 7,383,460 (‘460 Patent)	Accused Devices
	processor determines that the mode is periodic. If *it_interval* is 0, the processor determines that the mode is aperiodic. (*See kernel/itimer.c*; *kernel/hrtimer.c*.)
12. The system of claim 7, wherein the hardware-dependent interface is a hardware application layer (HAL) routine having an interface to receive the validated parameters associated with the request relayed from the hardware-independent interface.	**<u>In each Accused Device, the hardware-dependent interface is a hardware application layer (HAL) routine having an interface to receive the validated parameters associated with the request relayed from the hardware-independent interface</u>** As described above, the *clock_event_device()* data structure includes a member *set_next_event* that provides an indication of a hardware-dependent process to which the hardware-independent component should relay the request. The hardware-dependent processes identified above serve as hardware application layers. They further receive as arguments indications of a validated expiry time associated with the request relayed from the *hrtimer* component. By way of example, *omap2_gp_timer_set_next_event()* receives the validated parameter *cycles*. (*See /arch/arm/mach-omap2/timer-gp.c*.) By way of another example, *twd_set_next_event()* receives the validated parameter *evt*. (*See arch/arm/kernel/smp_twd.c*.) By way of another example, *msm_timer_set_next_event()* receives the validated parameter *cycles*. (*See arch/arm/mach-msm/timer.c*.) Likewise, Accused Devices using other processors, such as a Freescale Zeus processor, will include a hardware-dependent interface to receive validated parameters associated with the request related from the hardware-independent interface.
13. The system of claim 7, wherein the hardware-dependent interface further operates to execute an application service routine upon expiration of the timer.	**<u>In each Accused Device, the hardware-dependent interface further operates to execute an application service routine upon expiration of the timer</u>** The hardware-dependent components operate to execute an application service routine upon expiration of the timer. As discussed above, the Accused Devices handle the interrupt generated by the timer as to invoke *hrtimer_interrupt()*. *Hrtimer_interrupt()* invokes *__run_hrtimer* which references the *function* member of an *hrtimer* structure. The *function* member provides an indication of the application service routine that should be executed upon expiration of the timer. (*See kernel/hrtimer.c*.)