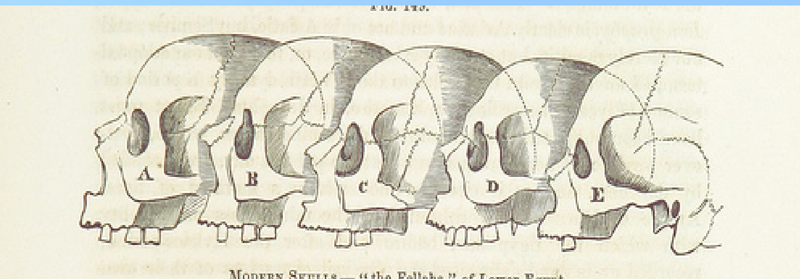


Viller Hsiao <villerhsiao@gmail.com>

F9 Microkernel ktimer

Rights to Copy

copyright © 2014 Viller Hsiao

F9 Microkernel [1]

- New open source implementation built from scratch, which deploys modern kernel techniques, derived from L4 microkernel designs, to deep embedded devices.
  - BSD License
- Characteristics of F9 microkernel
  - Efficiency: performance + power consumption
    - Tickless
  - Security: memory protection + isolated execution
  - Flexible development environment
- Repository
  - https://github.com/f9micro

Agenda

- Concept about operating system kernel time subsystem
- F9 Microkernel ktimer introduction

Concept about Operating System Kernel Time Subsystem

OS Kernel Time Subsystem

- Current Time
  - System time, UTC/GMP time, ... etc
- Alarm, the time event
- Process timeslice
- Timeout
- Time delay

System Time and Tick

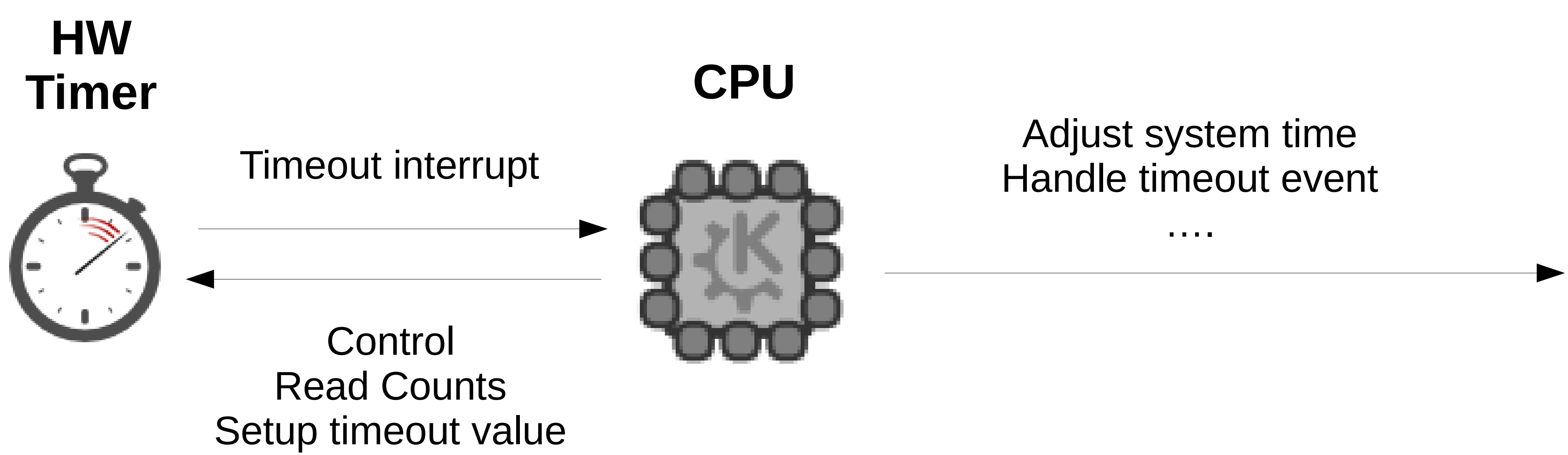
- From Wikipedia [2],

  “System time represents a computer system's notion of the passing of time.“

  “System time is measured by a system clock, which is typically implemented as a simple count of the number of ticks that have transpired since some arbitrary starting date, called the epoch.”

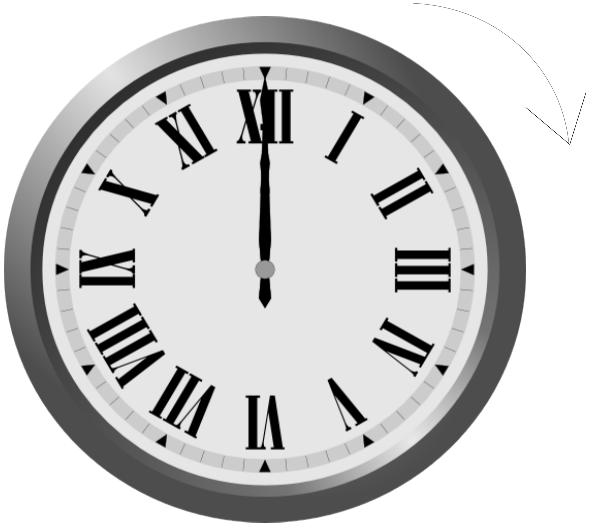
How Is Tick Implemented?

- Hardware timer device
  - Assert interrupt after a programmable inteval
    - Handling tick stuff in Timeout Interrupt Service Routine (ISR)



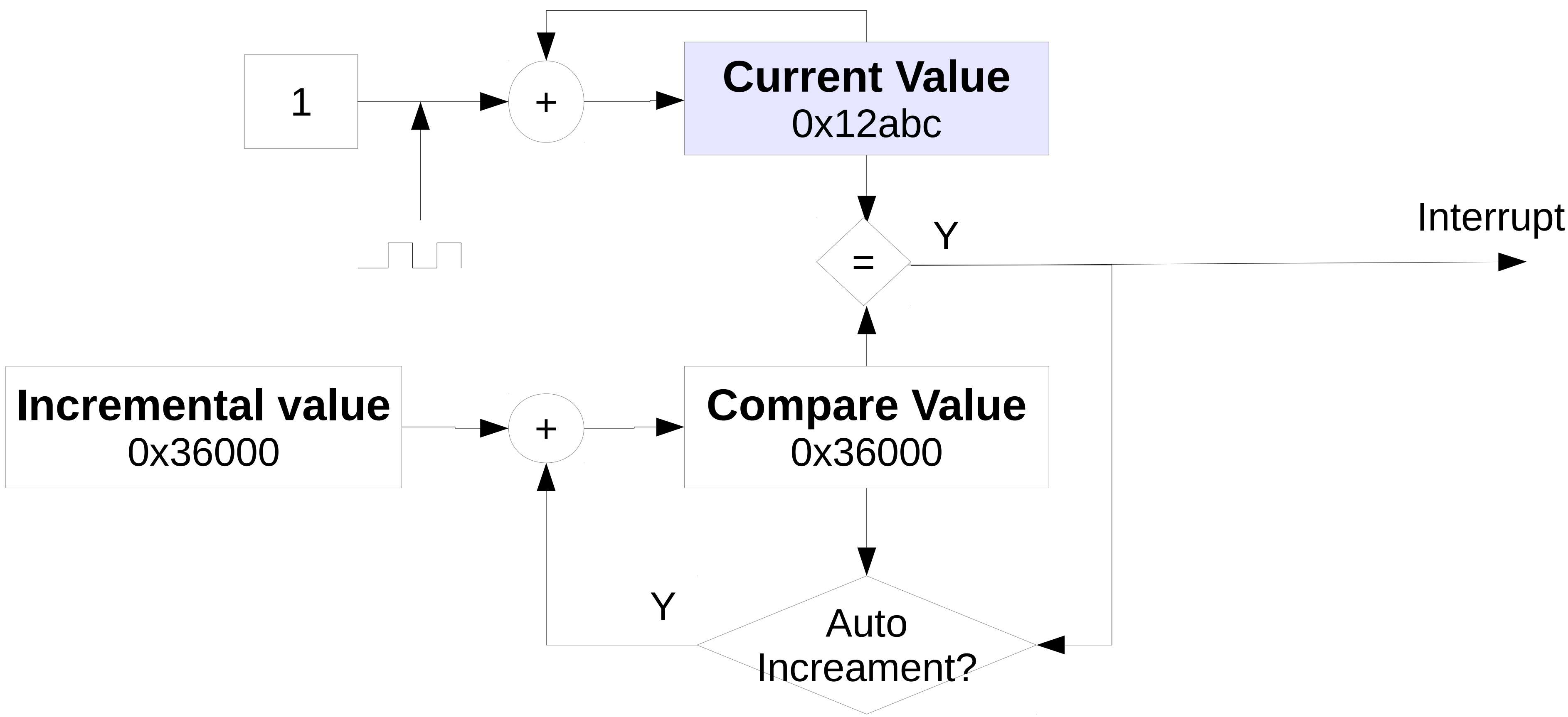
How Is Tick Implemented? (Cont.)

- Type 1: Incremental timer
  - Example: Cortex-A9 MP Global Timer



How Is Tick Implemented? (Cont.)

- Incremental timer functionality

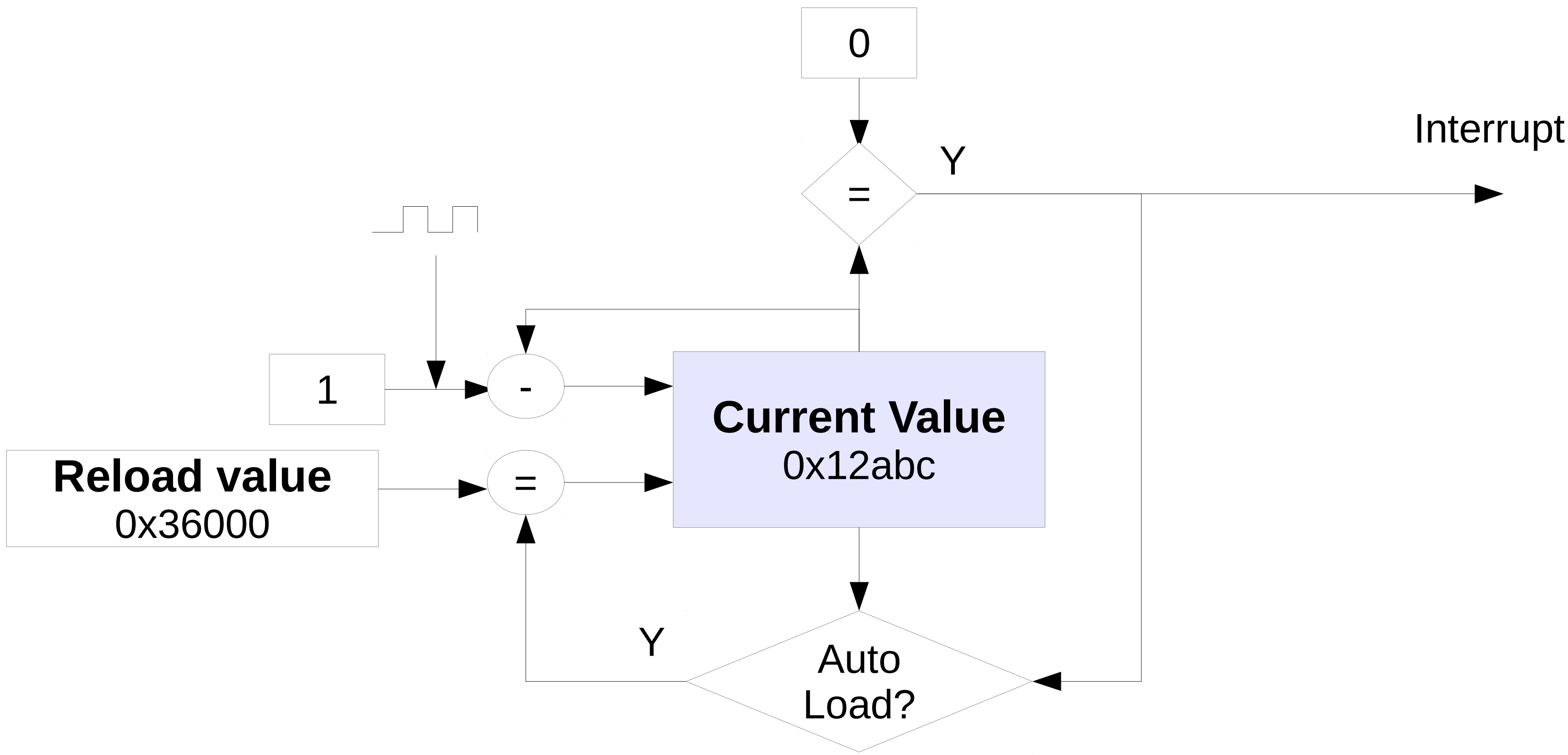


How Is Tick Implemented? (Cont.)

- Type 2: Counting down timer
  - Example: Cortex-M4 SysTick

How Is Tick Implemented? (Cont.)

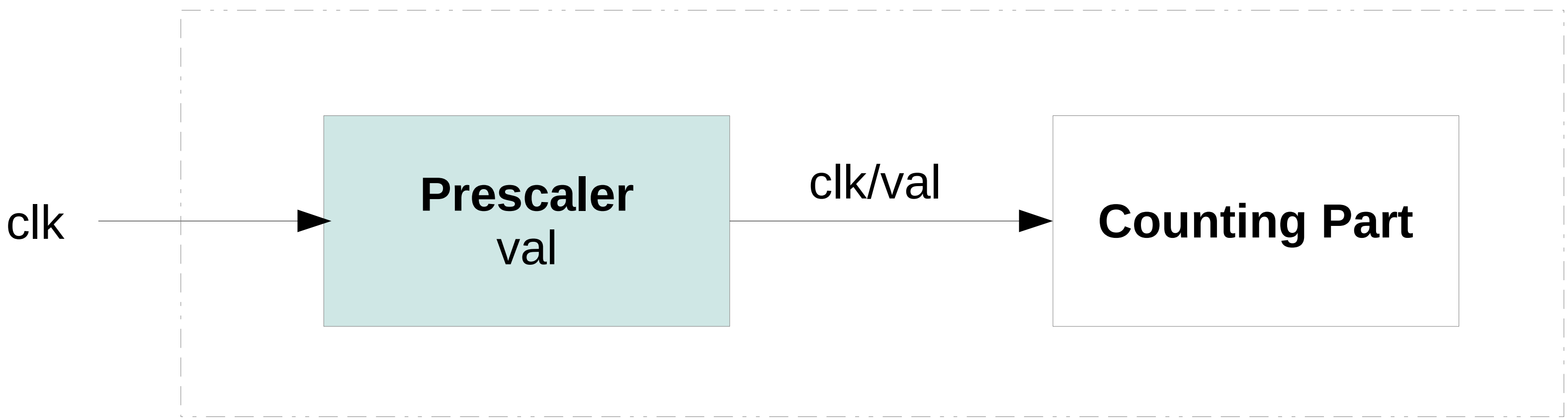
- Counting down timer functionality



How Is Tick Implemented? (Cont.)

- Prescaler
  - Tweak HW tick period

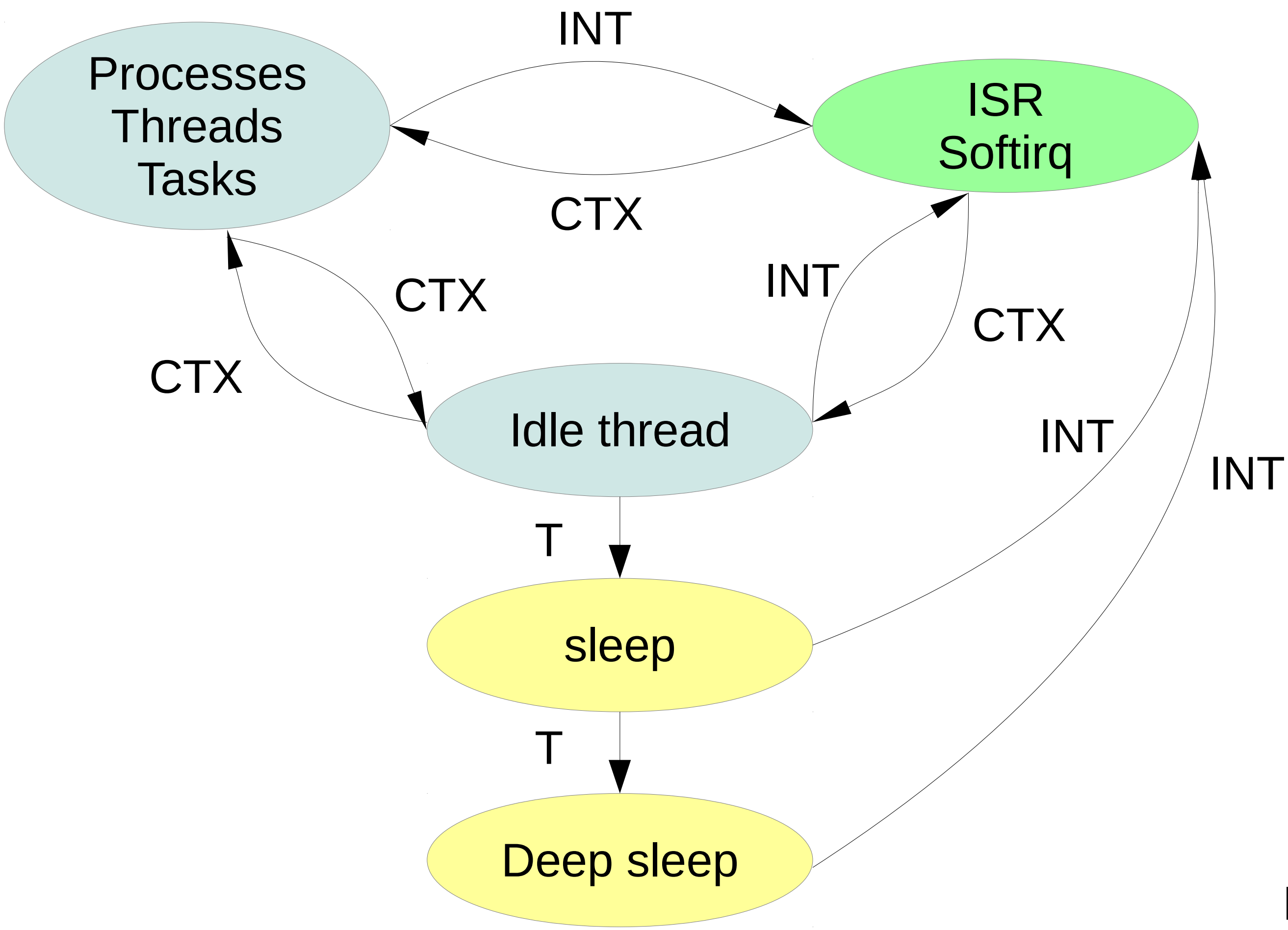
Timer Module



How Is Tick Implemented? (cont.)

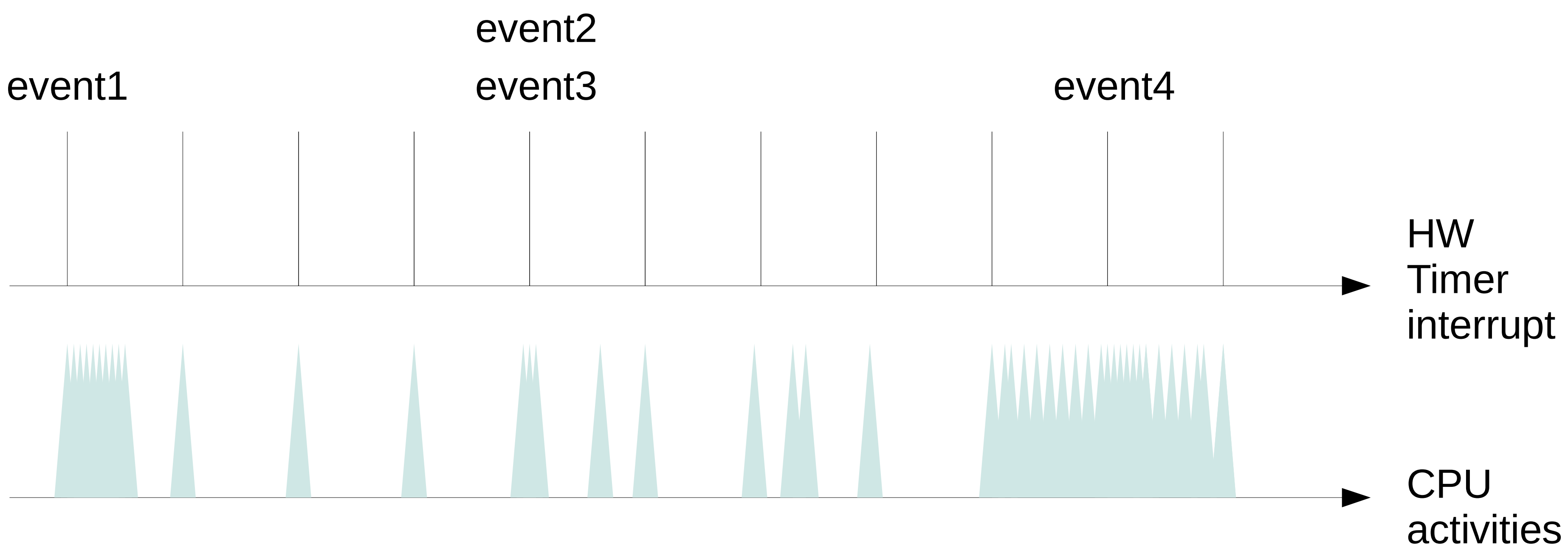
- Timeout ISR
  - Increase system ticks
  - Execute handler of timeout event
  - Re-schedule if required

Simple Graph of CPU Operating States



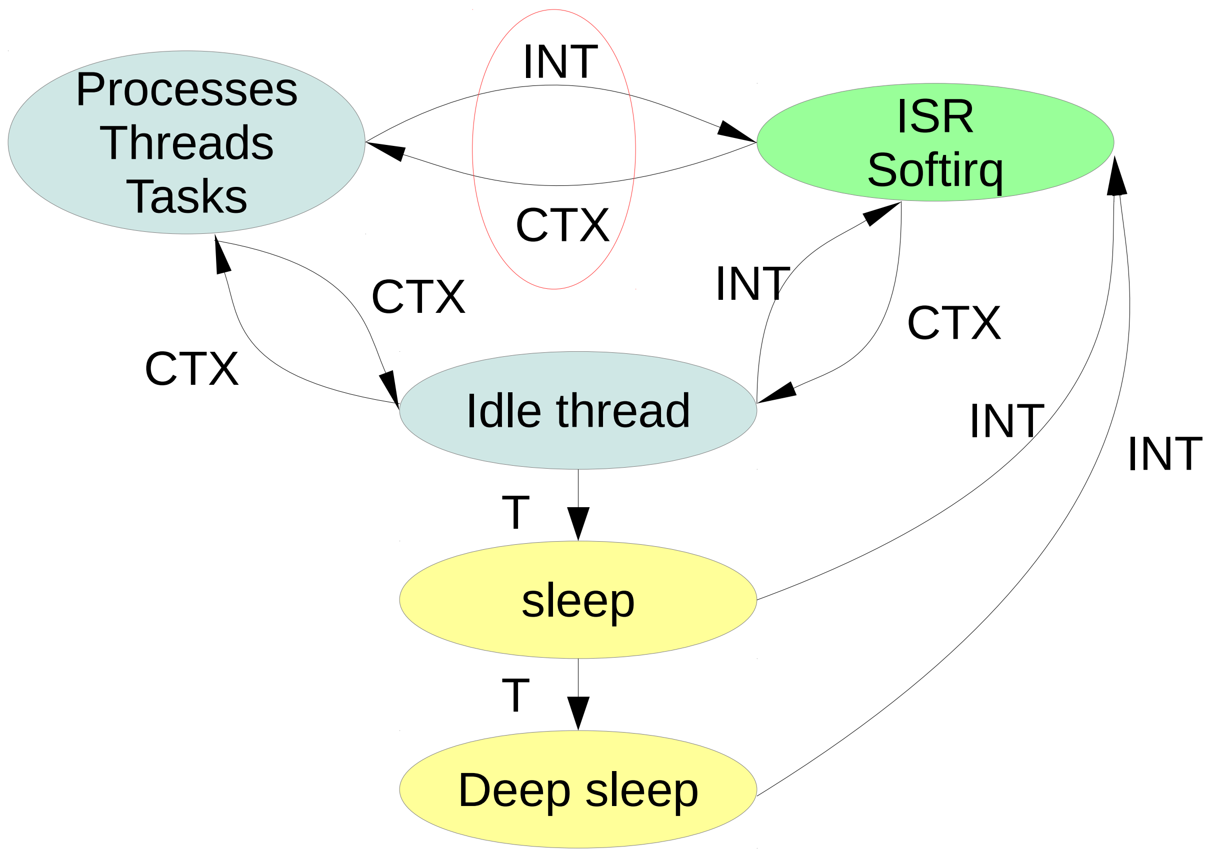
INT : interrupt
CTX: context switch
T : after a while

Time Diagram of Legacy Ticks



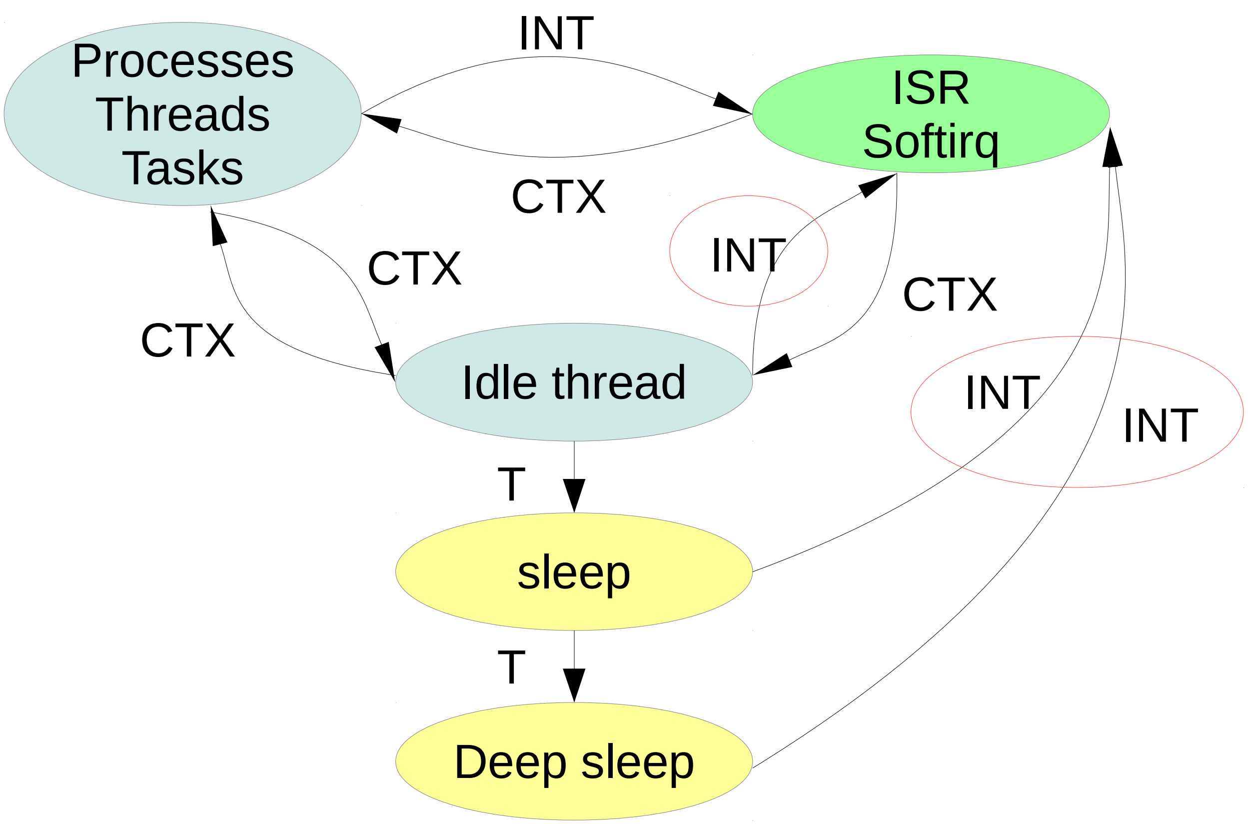
Where Is the Problem?

- Context switch overhead



Where Is the Problem?

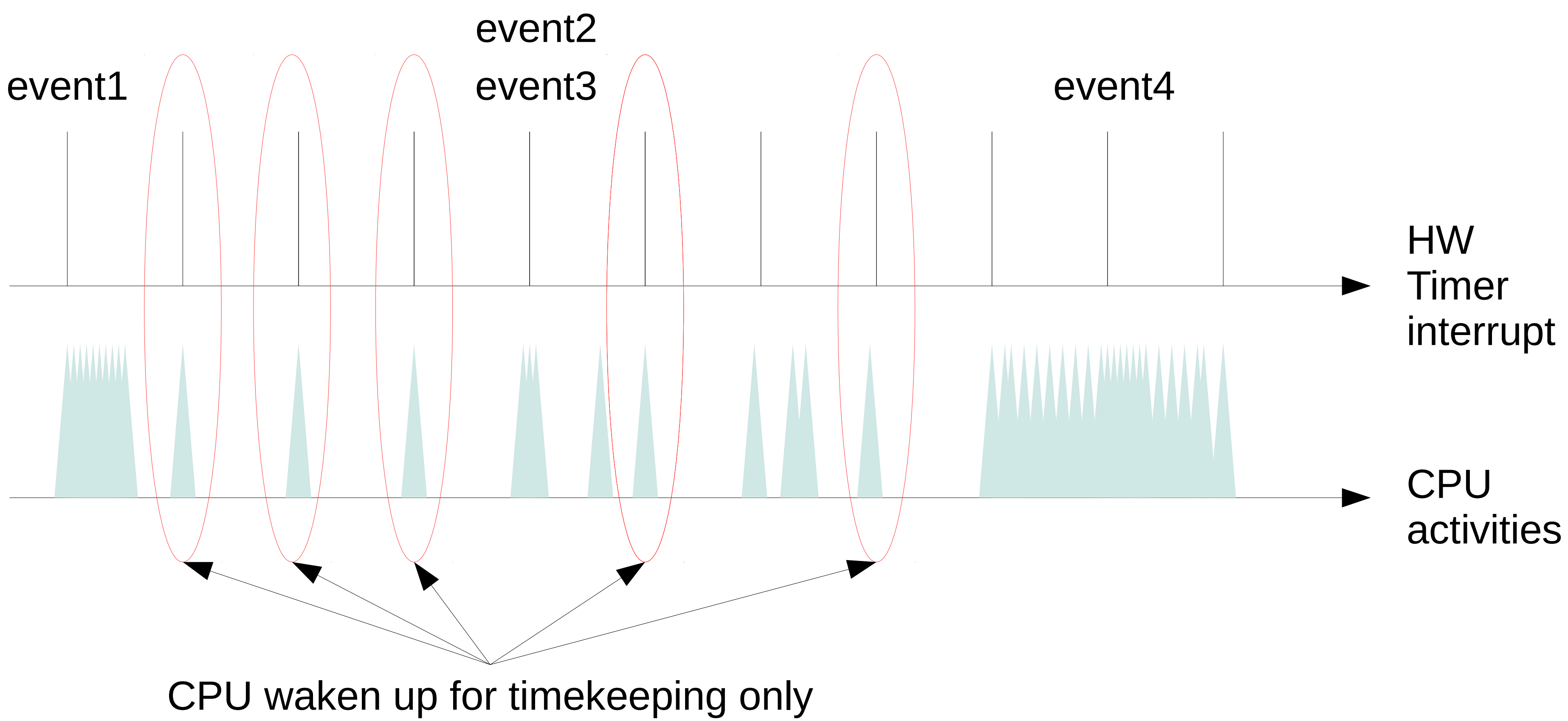
- Unecessary power consumed



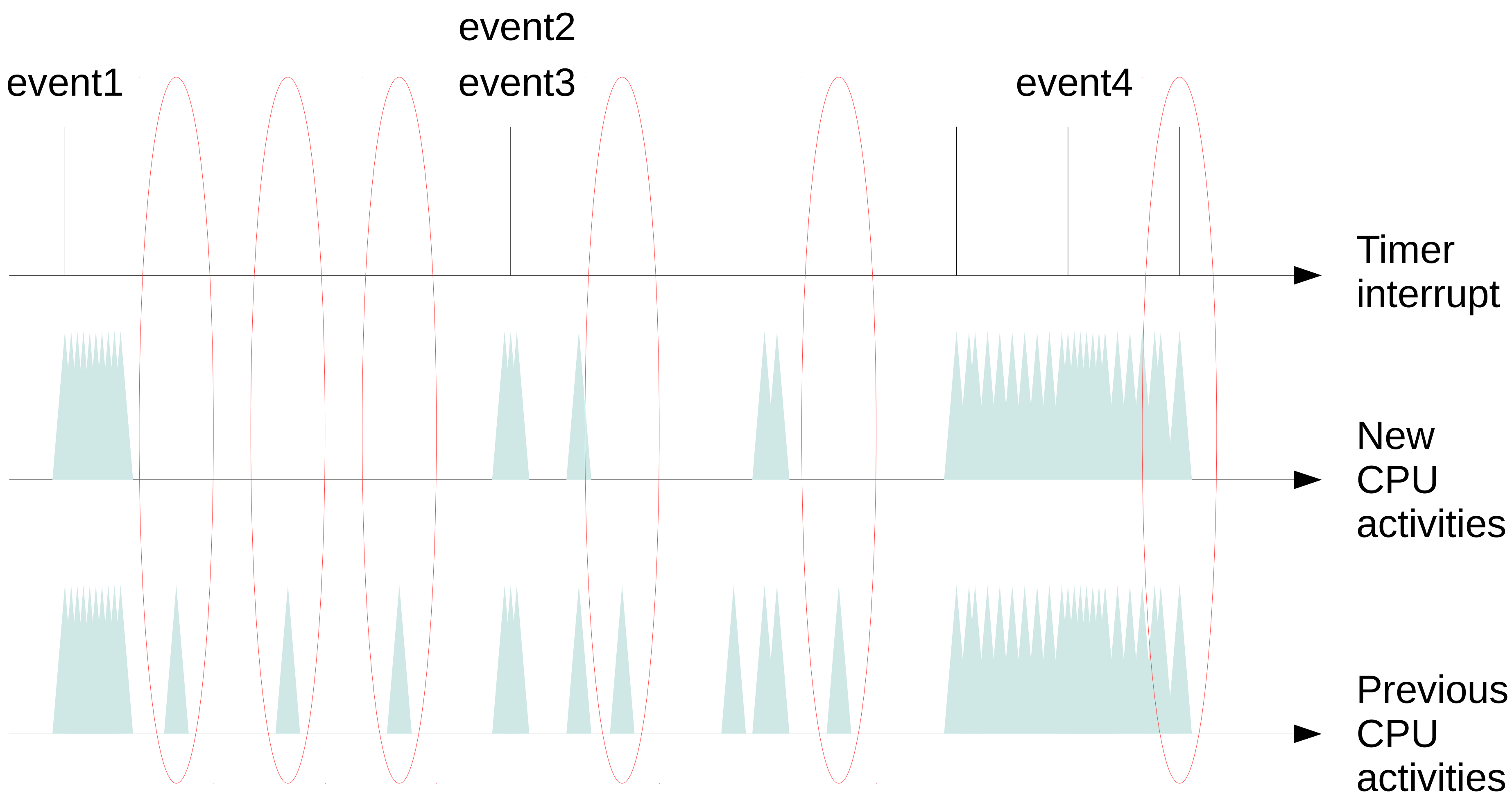
Where Is the Problem?

- Tick resolution
  - Performance, power v.s. timer precision

Time Diagram of Legacy Ticks



Solution: Tickless



Drawback of Tickless

- Tickless is not free [3],

  “It increases the number of instructions executed on the path to and from the idle loop.”

  “On many architectures, dyntick-idle mode also increases the number of expensive clock-reprogramming operations”

- Systems with aggressive real-time response constraints often run periodic tick

Example: Linux Time Subsystem

- Device level
  - clocksource
    - jiffies
  - clockevent

- High resolution kernel timer
  - hrtimers

Example: Linux Time Subsystem (Cont.)

- Tickless
  - (Nearly) Full tickless operation after 3.10
  - **dyntick-idle**
    - CONFIG_NO_HZ_IDLE
  - **adaptive-tick**
    - CONFIG_NO_HZ_FULL
  - **periodic**
    - CONFIG_HZ_PERIODIC
    - Consumes battery power 2-3 times as fast as **dyntick-idle** one. [3]

F9 Microkernel ktimer Introduction

Example Hardware Device

- STM32F4Discovery board
  - Cortex-M4 EVB
    - Use Cortex-M4 SysTick as the tick HW
      - 24-bit system timer
      - Counting down
      - Please refer to [4] for more detail

ktimer Features

- System time
- Time event
- Tickless

ktimer Configurations

- CONFIG_KTIMER_HEARTBEAT
  - HW cycles per ktimer tick
- CONFIG_KTIMER_MINTICKS
  - Minimum ktimer ticks unit for time event
- CONFIG_KTIMER_TICKLESS
  - Enable tickless

ktimer Event Utility

- **ktimer_event_create()**
  - Arguments
    - Timeout ticks
    - Timeout handler
    - Pointer to arguments for timeout handler
  - Return Non-zero to reschedule event again
    - Freed automatically if not be re-scheduled

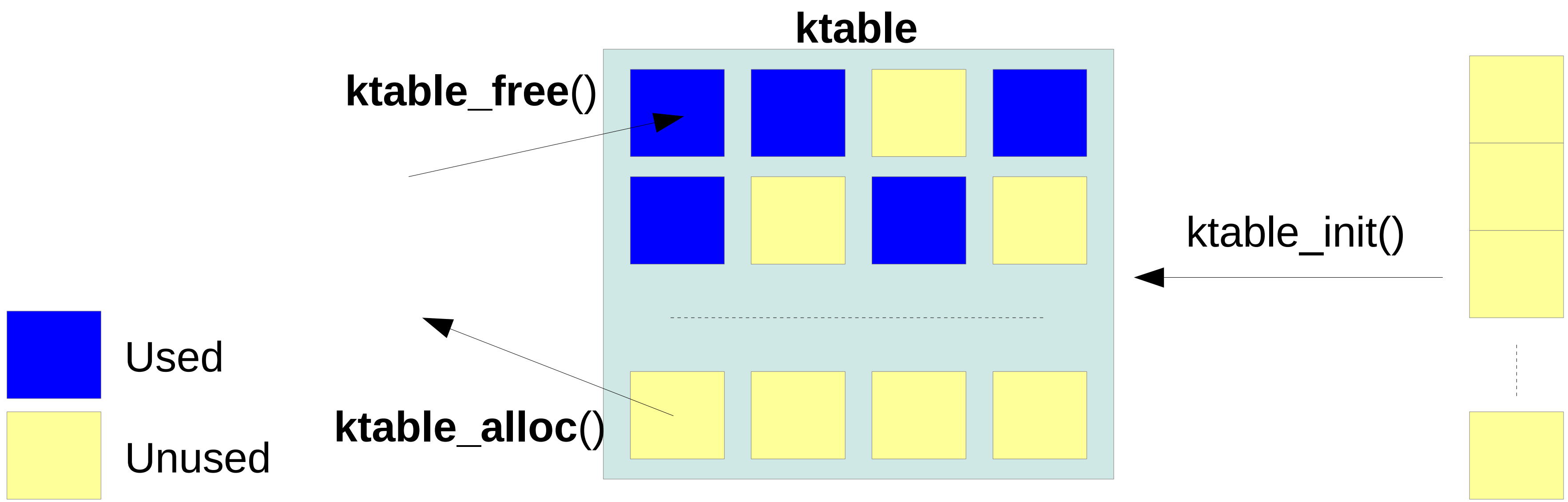
ktimer Anotomy

- System time
  - **ktime_now**

- ktimer event list
  - **event_queue**
  - Managed by **ktable**

ktimer Anotomy (Cont.)

- **ktable**
  - Basic kernel library in F9 Microkernel
  - Fast objects pool management
    - To allocate/free objects of pre-defined size and numbers easiler



ktimer Anotomy (Cont.)

- Life cycle of time event
  - One-shot
  - Continuous
- How to control life cycle in ktimer?
  - The return value of handler is treated as next timeout delta

ktimer Anotomy (Cont.)

- HW Timer ISR
  - **__ktimer_handler()**
    - Increase tick
    - Arrange Softirq for handling timer events

ktimer Anotomy (Cont.)

- Two-stage interrupt handling
  - ISR
    - IRQ context
  - Softirq
    - Thread context
      - Real time preemptive characteristic.
      - Can be scheduled like any other threads in the system [5].
    - Handled in **kernel thread**

ktimer Anotomy (Cont.)

- HW Timer ISR
  - **__ktimer_handler()**
    - Increase tick
    - Arrange Softirq for handling timer events

ktimer Anotomy (Cont.)

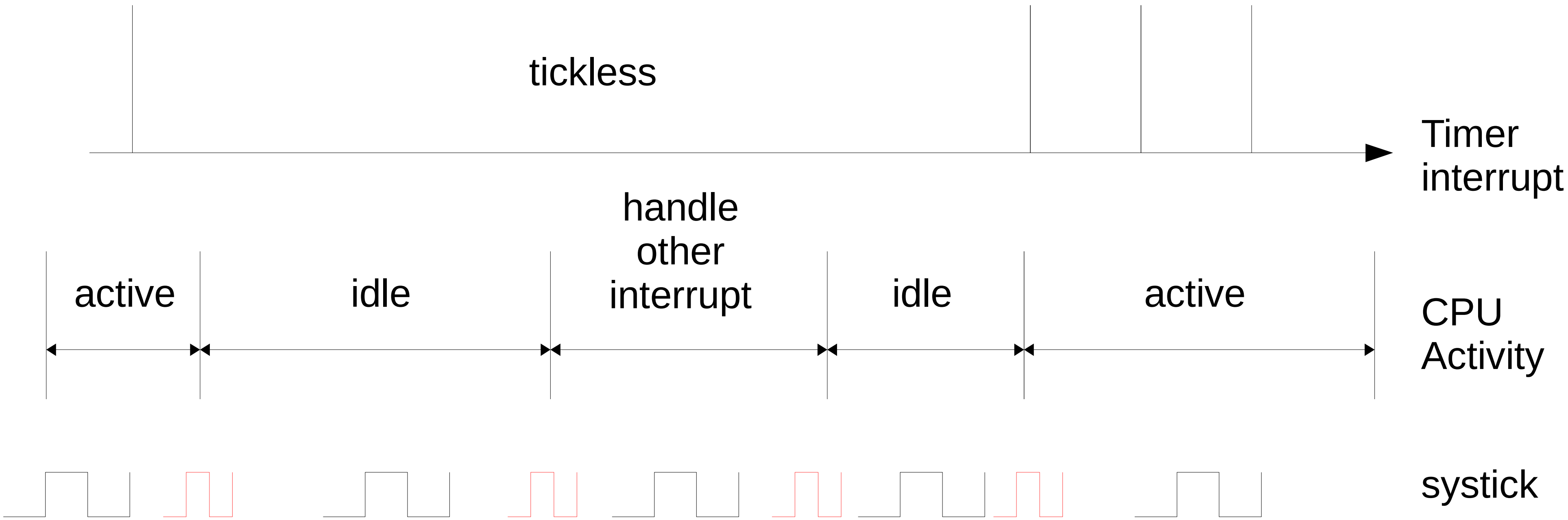
- HW timer softirq handler
  - **ktimer_event_handler()**
    - Find timeout event and executing its handler
    - Re-insert event if required

ktimer Anotomy (Cont.)

- Tickless
  - Enter tickless right before going to CPU idle state
  - Set interval of next timer interrupt as delta of next event
    - Or **KTIMER_MAXTICKS**
  - Adjust system time after waked up

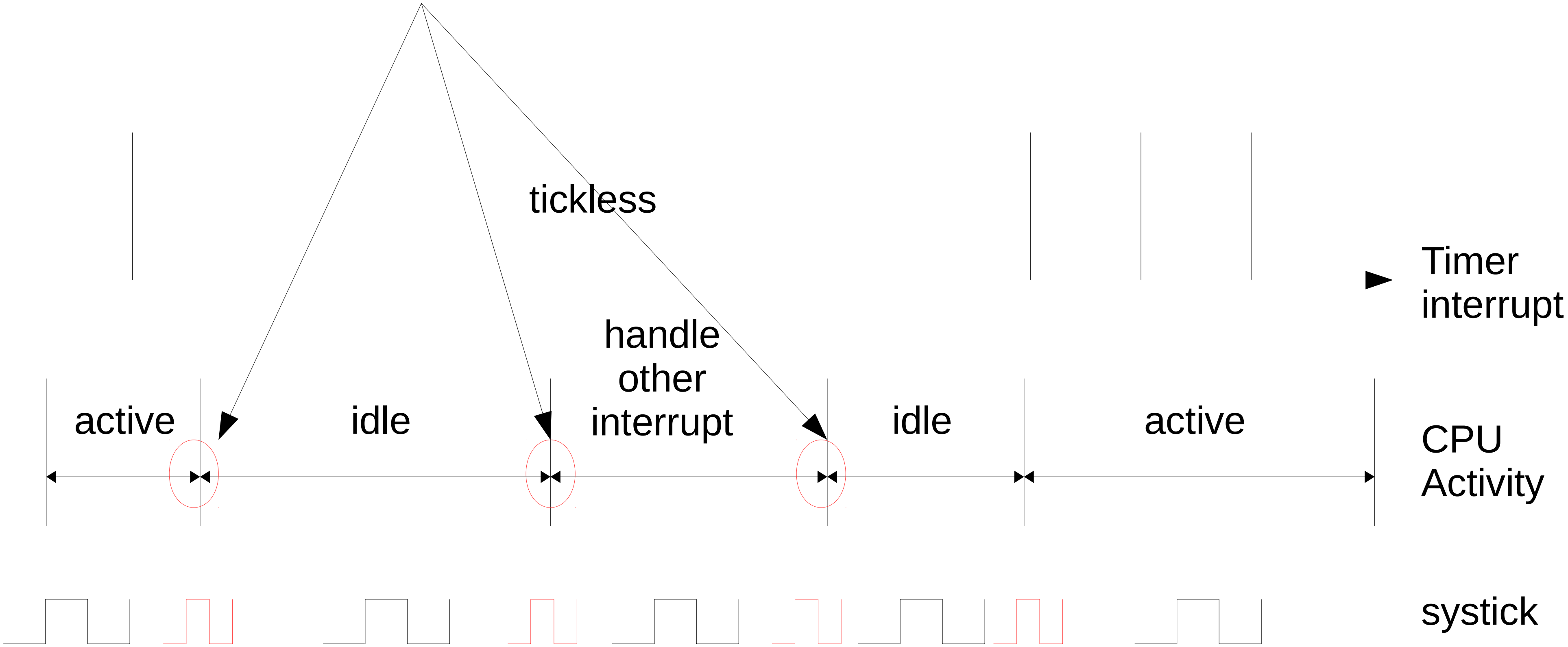
ktimer Anotomy (Cont.)

- Tickless Compensation
  - SysTick frequency distortion when enter/exit standby mode



ktimer Anotomy (Cont.)

- Tickless Compensation

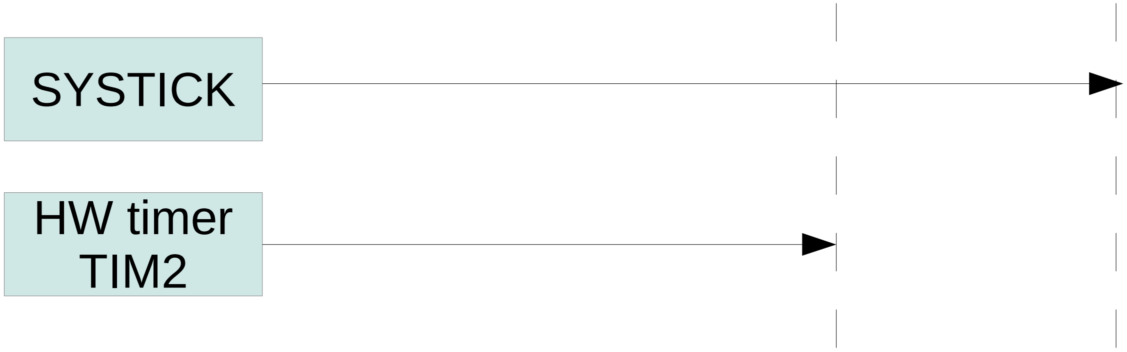


ktimer Anotomy (Cont.)

- Tickless Compensation
  - Pre-defined value
    - **CONFIG_KTIMER_TICKLESS_COMPENSATION**
      - Compensation value before entering idle
    - **CONFIG_KTIMER_TICKLESS_INT_COMPENSATION**
      - Compensation value when CPU is waked up by non-timer interrupt.
  - But how much to set these value to?

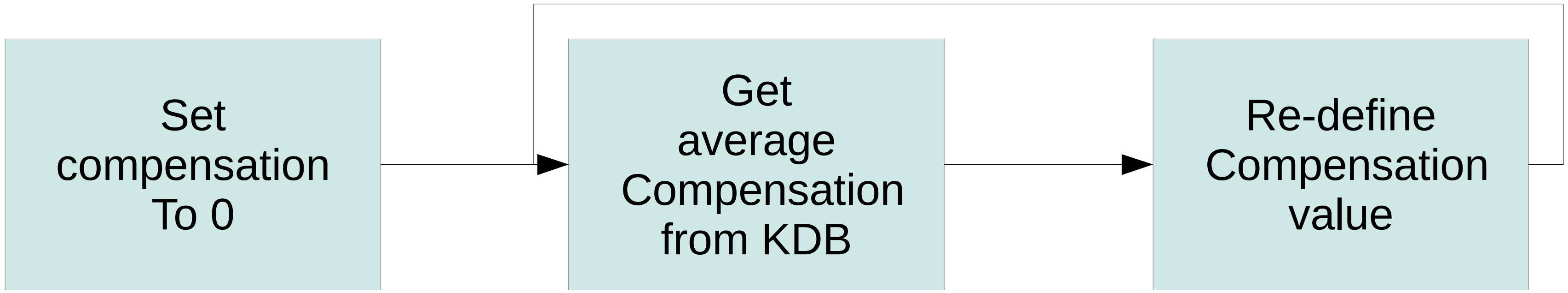
ktimer Anotomy (Cont.)

- Tickless Compensation
  - But how much to set these value to?
    - Get average compensation from general pupose timer
      - (total diff / count of entering tickless)



ktimer Anotomy (Cont.)

- Tickless Compensation
  - How to calibrate the compensation value?
    - **CONFIG_KTIMER_TICKLESS_VERIFY**
    - **CONFIG_KDB**



ktimer Anotomy (Cont.)

- KDB
  - Kernel debugger

```
## KDB ##
commands:
K: print kernel tables
e: dump ktimer events
n: show timer (now)
s: show softirqs
t: dump threads
m: dump memory pools
a: dump address spaces
p: show sampling
v: show tickless scheduling stat
----------------
```

```
## KDB ##
-------NOW------
Now is 70993
Ktimer T=1297 D=0
--------------------
```

Reference:

[1] Jim Huang (Dec 9, 2013), “F9: A Secure and Efficient Microkernel Built for Deep Embedded Systems”

[2] Wikipedia, “System time”

[3] P. E. McKenney (May 14, 2013), “NO_HZ: Reducing Scheduling-Clock Ticks”

[4] ST Mcroelectronics, STM32F3xxx and STM32F4xxx Cortex-M4 programming manual

[5] J. Altenberg, “Using the Realtime Preemption Patch on ARM CPUs”

THE END