Linux Mips Documentation

The kernel development community

Jan 15, 2023

CONTENTS

CHAPTER

ONE

BMIPS DEVICETREE BOOTING

Some bootloaders only support a single entry point, at the start of the kernel image. Other bootloaders will jump to the ELF start address. Both schemes are supported; CONFIG_BOOT_RAW=y and CONFIG_NO_EXCEPT_FILL=y, so the first instruction immediately jumps to kernel_entry().

Similar to the arch/arm case (b), a DT-aware bootloader is expected to set up the following registers:

a0 : 0

a1 : 0xffffffff

a2 : Physical pointer to the device tree block (defined in chapter II) in RAM. The device tree can be located anywhere in the first 512MB of the physical address space (0x00000000 - 0x1fffffff), aligned on a 64 bit boundary.

Legacy bootloaders do not use this convention, and they do not pass in a DT block. In this case, Linux will look for a builtin DTB, selected via CONFIG_DT_*.

This convention is defined for 32-bit systems only, as there are not currently any 64-bit BMIPS implementations.

CHAPTER

TWO

INGENIC JZ47XX SOCS TIMER/COUNTER UNIT HARDWARE

The Timer/Counter Unit (TCU) in Ingenic JZ47xx SoCs is a multi-function hardware block. It features up to eight channels, that can be used as counters, timers, or PWM.

- JZ4725B, JZ4750, JZ4755 only have six TCU channels. The other SoCs all have eight channels.
- JZ4725B introduced a separate channel, called Operating System Timer (OST). It is a 32-bit programmable timer. On JZ4760B and above, it is 64-bit.
- Each one of the TCU channels has its own clock, which can be reparented to three different clocks (pclk, ext, rtc), gated, and reclocked, through their TCSR register.
  - The watchdog and OST hardware blocks also feature a TCSR register with the same format in their register space.
  - The TCU registers used to gate/ungate can also gate/ungate the watchdog and OST clocks.
- Each TCU channel works in one of two modes:
  - mode TCU1: channels cannot work in sleep mode, but are easier to operate.
  - mode TCU2: channels can work in sleep mode, but the operation is a bit more complicated than with TCU1 channels.
- The mode of each TCU channel depends on the SoC used:
  - On the oldest SoCs (up to JZ4740), all of the eight channels operate in TCU1 mode.
  - On JZ4725B, channel 5 operates as TCU2, the others operate as TCU1.
  - On newest SoCs (JZ4750 and above), channels 1-2 operate as TCU2, the others operate as TCU1.
- Each channel can generate an interrupt. Some channels share an interrupt line, some don’t, and this changes between SoC versions:
  - on older SoCs (JZ4740 and below), channel 0 and channel 1 have their own interrupt line; channels 2-7 share the last interrupt line.
  - On JZ4725B, channel 0 has its own interrupt; channels 1-5 share one interrupt line; the OST uses the last interrupt line.
  - on newer SoCs (JZ4750 and above), channel 5 has its own interrupt; channels 0-4 and (if eight channels) 6-7 all share one interrupt line; the OST uses the last interrupt line.

2.1 Implementation

The functionalities of the TCU hardware are spread across multiple drivers:

clocks	drivers/clk/ingenic/tcu.c
interrupts	drivers/irqchip/irq-ingenic-tcu.c
timers	drivers/clocksource/ingenic-timer.c
OST	drivers/clocksource/ingenic-ost.c
PWM	drivers/pwm/pwm-jz4740.c
watchdog	drivers/watchdog/jz4740_wdt.c

Because various functionalities of the TCU that belong to different drivers and frameworks can be controlled from the same registers, all of these drivers access their registers through the same regmap.

For more information regarding the devicetree bindings of the TCU drivers, have a look at Documentation/devicetree/bindings/timer/ingenic,tcu.yaml.

CHAPTER

THREE

FEATURE STATUS ON MIPS ARCHITECTURE

Subsystem	Feature	Kconfig	Status	Desc
core	cBPF-JIT	HAVE_CBPF_JIT	ok	arch
core	eBPF-JIT	HAVE_EBPF_JIT	ok	arch
core	generic-idle-thread	GENERIC_SMP_IDLE_THREAD	ok	arch
core	jump-labels	HAVE_ARCH_JUMP_LABEL	ok	arch
core	thread-info-in-task	THREAD_INFO_IN_TASK	TODO	arch
core	tracehook	HAVE_ARCH_TRACEHOOK	ok	arch
debug	debug-vm-pgtable	ARCH_HAS_DEBUG_VM_PGTABLE	TODO	arch
debug	gcov-profile-all	ARCH_HAS_GCOV_PROFILE_ALL	ok	arch
debug	KASAN	HAVE_ARCH_KASAN	TODO	arch
debug	kcov	ARCH_HAS_KCOV	ok	arch
debug	kgdb	HAVE_ARCH_KGDB	ok	arch
debug	kmemleak	HAVE_DEBUG_KMEMLEAK	ok	arch
debug	kprobes	HAVE_KPROBES	ok	arch
debug	kprobes-on-ftrace	HAVE_KPROBES_ON_FTRACE	TODO	arch
debug	kretprobes	HAVE_KRETPROBES	ok	arch
debug	optprobes	HAVE_OPTPROBES	TODO	arch
debug	stackprotector	HAVE_STACKPROTECTOR	ok	arch
debug	uprobes	ARCH_SUPPORTS_UPROBES	ok	arch
debug	user-ret-profiler	HAVE_USER_RETURN_NOTIFIER	TODO	arch
io	dma-contiguous	HAVE_DMA_CONTIGUOUS	ok	arch
locking	cmpxchg-local	HAVE_CMPXCHG_LOCAL	TODO	arch
locking	lockdep	LOCKDEP_SUPPORT	ok	arch
locking	queued-rwlocks	ARCH_USE_QUEUED_RWLOCKS	ok	arch
locking	queued-spinlocks	ARCH_USE_QUEUED_SPINLOCKS	ok	arch
perf	kprobes-event	HAVE_REGS_AND_STACK_ACCESS_API	ok	arch
perf	perf-regs	HAVE_PERF_REGS	ok	arch
perf	perf-stackdump	HAVE_PERF_USER_STACK_DUMP	ok	arch
sched	membarrier-sync-core	ARCH_HAS_MEMBARRIER_SYNC_CORE	TODO	arch
sched	numa-balancing	ARCH_SUPPORTS_NUMA_BALANCING	TODO	arch
seccomp	seccomp-filter	HAVE_ARCH_SECCOMP_FILTER	ok	arch
time	arch-tick-broadcast	ARCH_HAS_TICK_BROADCAST	ok	arch
time	clockevents	!LEGACY_TIMER_TICK	ok	arch
time	context-tracking	HAVE_CONTEXT_TRACKING	ok	arch
time	irq-time-acct	HAVE_IRQ_TIME_ACCOUNTING	ok	arch
time	virt-cpuacct	HAVE_VIRT_CPU_ACCOUNTING	ok	arch

Table 1 – continued from p

Subsystem	Feature	Kconfig	Status	Desc
vm	batch-unmap-tlb-flush	ARCH_WANT_BATCHED_UNMAP_TLB_FLUSH	TODO	arch
vm	ELF-ASLR	ARCH_HAS_ELF_RANDOMIZE	ok	arch
vm	huge-vmap	HAVE_ARCH_HUGE_VMAP	TODO	arch
vm	ioremap_prot	HAVE_IOREMAP_PROT	ok	arch
vm	PG_uncached	ARCH_USES_PG_UNCACHED	TODO	arch
vm	pte_special	ARCH_HAS_PTE_SPECIAL	ok	arch
vm	THP	HAVE_ARCH_TRANSPARENT_HUGEPAGE	ok	arch